Collibra

Collibra Data Intelligence Cloud

Collibra Data Quality & Observability

Version 2023.04

Collibra

Collibra Collibra Data Quality & Observability guide

Release date: April 2, 2023 / Revision date: March 30, 2023

You can find the most up-to-date technical documentation on our documentation site at

https://productresources.collibra.com/doc/collibra/latest/Content/DataQuality/to_data-quality.htm

Contents

Contents ii

Introducing Collibra Data Quality v

Getting Started vii

Release Notes ix

Builds xciv

APIs cii

Collibra DQ Installation cv

Collibra DQ Connections ccxxiv

Collibra DQ Features cccix

Known Limitations cccxlvii

Cross-Data Set Rules cccxlviii

In-Clause (Single Column) cccxlviii

Except (Multi-Column) cccxlviii

Referencing secondary data sets cccxlix

Joins cccxlix

Sample Results cccl

Known Limitations cccl

Data Type ccclvi

Features ..... cdxv

Known Limitations ..... cdxvi

Steps ..... cdxvii

Shapes ..... cdxix

Replay ..... cdxix

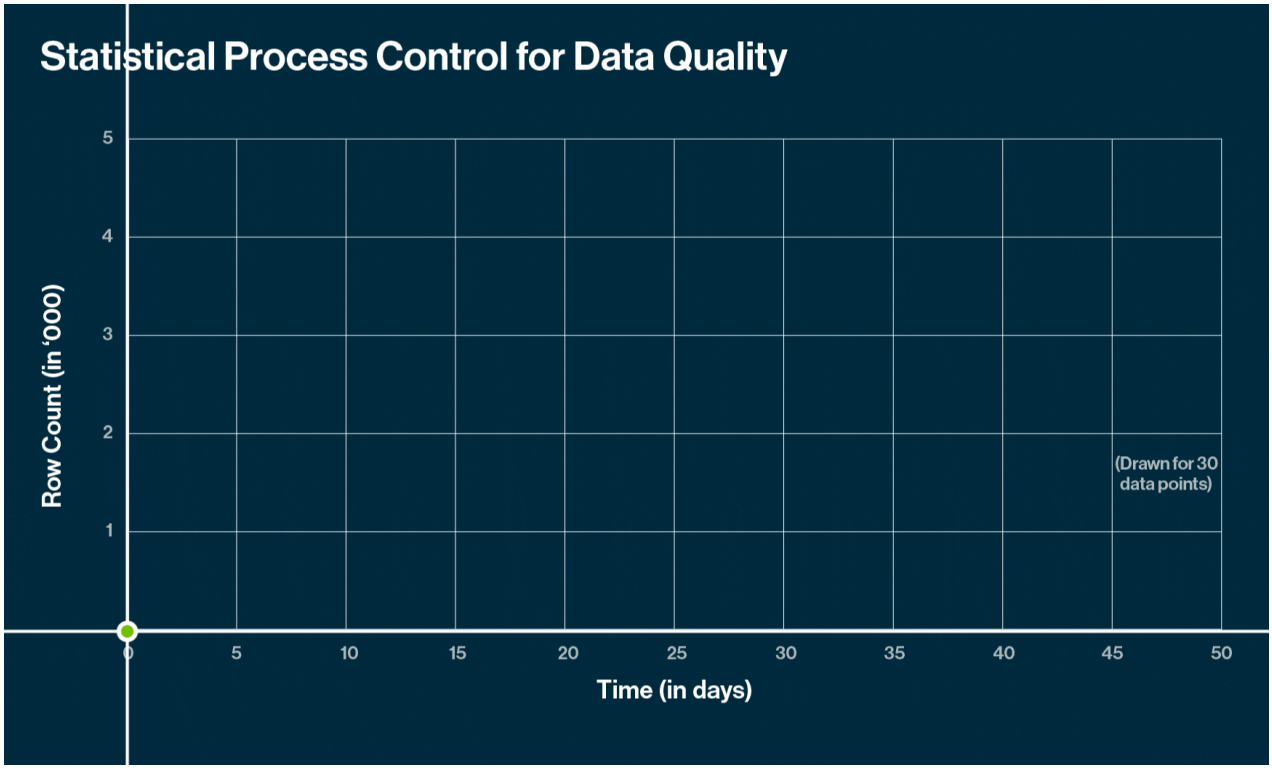
Outliers ..... cdxix
Running a Full Scan ..... cdxxii
Prerequisites ..... cdxxii
Steps ..... cdxxii
What's next? ..... cdxxiii
Running a Partial Scan ..... cdxxiv
Prerequisites ..... cdxxiv
Steps ..... cdxxiv
Time Slice ..... cdxxv
Row Filter ..... cdxxvi
Limit ..... cdxxvii
What's next? ..... cdxxvii
Scanning with SQL Query ..... cdxxviii
Prerequisites ..... cdxxviii
Steps ..... cdxxviii
What's next? ..... cdxxix
Collibra DQ Scorecards ..... cdxxxv
Collibra DQ Scheduler ..... cdxliv
Collibra DQ Alerts ..... cdxlix
Collibra DQ Reports ..... cdlxiii
Collibra DQ Workflows ..... dxiv
Collibra DQ DIC Integration ..... dxxx
Collibra DQ Catalog ..... dlviii
Collibra DQ Solutions ..... dlxxi
Background ..... dlxxxvii
What Did We Notice? ..... dlxxxviii

Traditional Approach ..... dlxxxviii
What Did We Need? ..... dlxxxix
DQ is the difference ..... dxc
What is CDQ ..... dxci
8 Ways to Add Value Using CDQ ..... dxci
What Savings Does CDQ Provide? ..... dxcii
How Can CDQ Help? ..... dxcii
What makes CDQ unique? ..... dxciii
Why? ..... dxciv
Collibra DQ Benchmarks ..... dcxxxii
Collibra DQ APIs ..... dcxliv
Rule types ..... dccxxv
Simple rule ..... dccxxvi
Example #1 ..... dccxxvi
Example #2 ..... dccxxix
Freeform SQL ..... dccxxxiii
Individual statement ..... dccxxxiii
Join statements ..... dccxxxiv
Rule types ..... dccxxxvi
Syntax ..... dccxxxvii
Collibra DQ Architecture ..... dccclxxvii
Collibra DQ Admin ..... dcccxciii
Collibra DQ Security Configuration ..... cmxx
Collibra DQ Legal ..... cmliii

Introducing Collibra Data Quality

Automatic data quality without the need for rules. Collibra Data Quality & Observability provides a fast and elegant way to manage your datasets by learning through observation rather than human input.

Collibra DQ applies the latest advancements in data science and machine learning to the problem of data quality, surfacing data issues in minutes instead of months.



Getting Started with Collibra DQ

A Pluggable and Complete Data Quality Framework

If you are adding data quality to your data pipelines the below visual illustrates the number of products and pieces you will need consider to successfully complete your overall governance program. The Collibra DQ suite allows you to use either native Collibra DQ components or integrate the 3rd party components of your choice. By using our best practice guide and framework you can easily complete the DQ lifecycle.

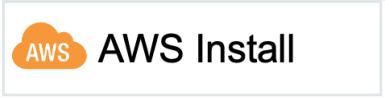
Data Pipelines that Tie into the DQ Framework

Collibra DQ offers a coding framework for developers or ETL designers that want to built real-time data quality into their broader data pipeline. This provides the same algos and DQ checks as the Collibra DQ UI Wizard but with direct access into your code points. Consistency is a must to have a program you trust.

Getting Started

Standard	The install script for a standard install. *All software and data resides in your VPC.* Install time: `7 minutes`	Click Here
Containerized	Uses Helm Charts and K8s. For users who want WebApp and Compute components in containers. Compute pools are supported. *All software and data resides in your VPC.* Install time: `12 minutes`	Click Here
Edge	Provides a cloud application in Collibra Cloud but an Edge component on your premise to safeguard your data. *Customer data and connection credentials residste in your VPC. Application data is stored on Collibra Cloud.* Install time: `1 hour`	Click Here

Marketplace	The Google Marketplace option is a simple, 1-click image installation. This is the quickest option for single-server install in GCP. *All software and data resides in your VPC.* Install time: `5 minutes`	Click Here
Google Cloud Deployment Manager	Hooks into GCP cloud services like GKE for ephemeral compute and RDS for cloud database. *All software and data resides in your VPC.*	Coming soon



Marketplace	The Amazon Marketplace option is a simple, 1-click image installation. This is the quickest option for single-server install in AWS Cloud. *All software and data resides in your VPC.* Install time: `5 minutes`	Click Here
CloudFormation	Hooks into AWS cloud services like EKS for ephemeral compute and RDS for cloud database. *All software and data resides in your VPC.* Install time: `12 minutes`	Click Here

Please see the Agreements for the terms and conditions on Collibra's evaluation offerings.

> Tip For more information, please contact **info@collibra.com**

Release Notes

> Important
> **Disclaimer** - Failure to upgrade to the most recent release of the Collibra Service may adversely impact the security, reliability, availability, integrity, performance or support (including Collibra's ability to meet its service levels) of the Service. Collibra hereby disclaims all liability, express or implied, for any reduction in the security, reliability, availability, integrity, performance or support of the Service to the extent the foregoing would have been avoided had you allowed Collibra to implement the most current release of the Service when scheduled by Collibra. Further, to the extent your failure to upgrade the Service impacts the security, reliability, availability, integrity or performance of the Service for other customers or users of the Service, Collibra may suspend your access to the Service until you have upgraded to the most recent release.

2023.03

- Highlights
- New Features
- Enhancements
- Fixes
- Known Limitations
- DQ Security Metrics

Highlights

- When your Collibra DQ license is about to expire, a counter in a warning banner on the Collibra DQ home page now counts down from 30 days until your license expires. You can update your license key at any time to avoid any disruptions.
    - A message now appears in the Description field of the Jobs activity logs and the Job Exception field of the Jobs log table view when a DQ Job fails because

of an expired or invalid license key.
    - An error message now appears on the Findings page when a job fails because of an expired or invalid license key.
    - When admin users update a license with a valid license key on the License page of the Admin Console, a success message appears.
    - When admin users update a license with an invalid license key on the License page of the Admin Console, an error message appears.
        - DQ Jobs will again run normally once a valid license key is saved.

> Warning
> Any DQ Jobs that run on accounts with expired or invalid license keys will no longer run until a valid license key is saved.

- To improve how Collibra DQ handles Jobs concurrently at lower memory agent configurations, the default agent batch size for concurrent Jobs is now 5. For example, when you run more than 5 Jobs concurrently, only 5 Jobs run at once. All additional Jobs remain in Staged status until the initial 5 Jobs are complete, which allows new Jobs to continue to enter the queue on the Jobs page.
    - The following are the default values in the Helm owl-agent-configmap.yaml file:
        - `DEFAULT_BATCH_ID: <base64-encoded-string>`
        - `DEFAULT_BATCH_SIZE: 5`
    - The following are the default values in the Helm values.yaml and values_test.yaml files:
        - `agent_batch_id: "default?batchSize=5"`
        - `agent_batch_size: 5`
    - It may be necessary to override the default values for high concurrency requirements. Increase the `DEFAULT_BATCH_ID` and `DEFAULT_BATCH_SIZE` values and update the corresponding Helm values for DQ Agent configuration accordingly.
- You can now convert log outputs to JSON formatting for ingestion by a third-party log consumer.
- You can now optionally include additional fields and hyperlinks to the environments from which alerts originate in email alerts.
    - Admin users can turn this option on or off from the Alerts page within the Admin Console.

> Note
> If you only install owlweb or owlagent, you need to copy the owlmanage.sh file from the installation package into the bin directory with the following command:
>
> ```
> cp $PACKAGE_LOCATION/owlmanage.sh $INSTALL_
> PATH/bin/owlmanage.sh
> ```

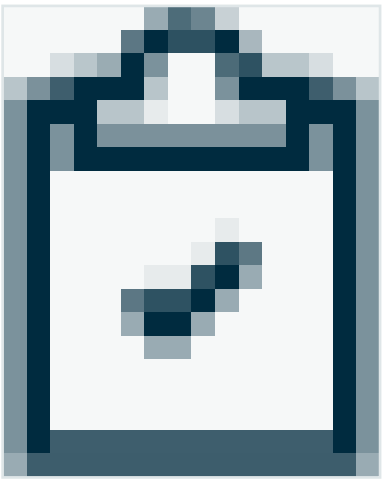
New Features

Platform

- Collibra DQ now displays in-app messages about expired, expiring, and invalid licenses, including a banner message on the Collibra DQ home screen that counts down from 30 days until your license expires.
    - You can update or edit your license from the License Management page in the Admin Console.

Enhancements

Jobs

- The icon for the Jobs page is now the  clipboard icon.

Rules

- Export LinkIds now has the same export capabilities as Export and Export with Details.
- You can now use keyboard shortcuts on the Rule Builder page to display dataset-level stat rules.

Alerts

- You can now optionally include additional fields and hyperlinks to the environments from which alerts originate in email alerts.

Agent

- To improve how Collibra DQ handles Jobs concurrently at lower memory agent configurations, the default agent batch size for concurrent jobs is now 5.

Admin

- You can now set default values on the Add External Service Configuration form on the Assignment Queues page when you want to override the values for individual finding validate action.

Security

- Collibra DQ now enforces a Spring CSRF token, as well as CORS and SameSite, as standards to mitigate against CSRF attacks.
    - Any cookie-based auth requests without this token are rejected with a 401 error.
    - This feature is active by default, but can be deactivated if necessary. Deactivating this feature does not compromise the security of the Collibra DQ app because we have other mitigation standards in place for CSRF.
    - All non-GET HTTP requests need to carry a CSRF token header when they call the API. This happens automatically when you use the app within Collibra DQ.

    - This implementation only impacts cookie-based authentication. Additionally, Swagger UI and APIs that use bearer token authentication are not impacted by this change.
- Server-side validation for username, first name, last name, and email is now included in the /v2/createlocaljdbcuser internal registration API.

Platform

- New User Request emails now include additional fields and hyperlinks to the environment from which the request originates for easier access.
- You can now convert log outputs to JSON formatting for ingestion by a third-party log consumer.

Snowflake Pushdown (beta)

- You can now scan for exact and fuzzy matches as part of the Dupes check layer.
- You can now use the Correlation tab on the Profile page to understand correlation levels between numerical columns.
- Minimum and maximum string length profiling is now available.

Fixes

Explorer

- Fixed an issue where a custom Transform statement was corrupted after running a DQ Job and clicking the Edit button. (ticket #104992)

Rules

- Fixed an issue where custom rules with primary and secondary datasets that shared similar naming conventions failed because of incorrect replacement of @Dataset or @secondary with the name of the dataset by Collibra DQ's replace logic. (tickets #105812, 108490)

Source

- Fixed an issue that resulted in an Authentication error when the source data loaded because both the source and target used the same Kerberos TGT-based connection. (ticket #102639)

Platform

- Fixed an issue where incorrect syntax in the owldq/templates/rbac.yaml manifest caused errors for some Kubernetes deployments on AKS.
- Fixed an issue where an upgrade to Release 2023.01 caused errors, which referred to the agent_displayname column on the agent_q table when the web pods launched.
- Fixed an issue where the Flyway script V188 did not run successfully after an upgrade from Release Version 2022.08 to 2023.01. (ticket #107222)

Connections

- Fixed an issue that prevented Collibra DQ from establishing a connection to Azure Data Lake after an upgrade to Release Version 2022.11. (tickets #104271, 107451)

Outliers

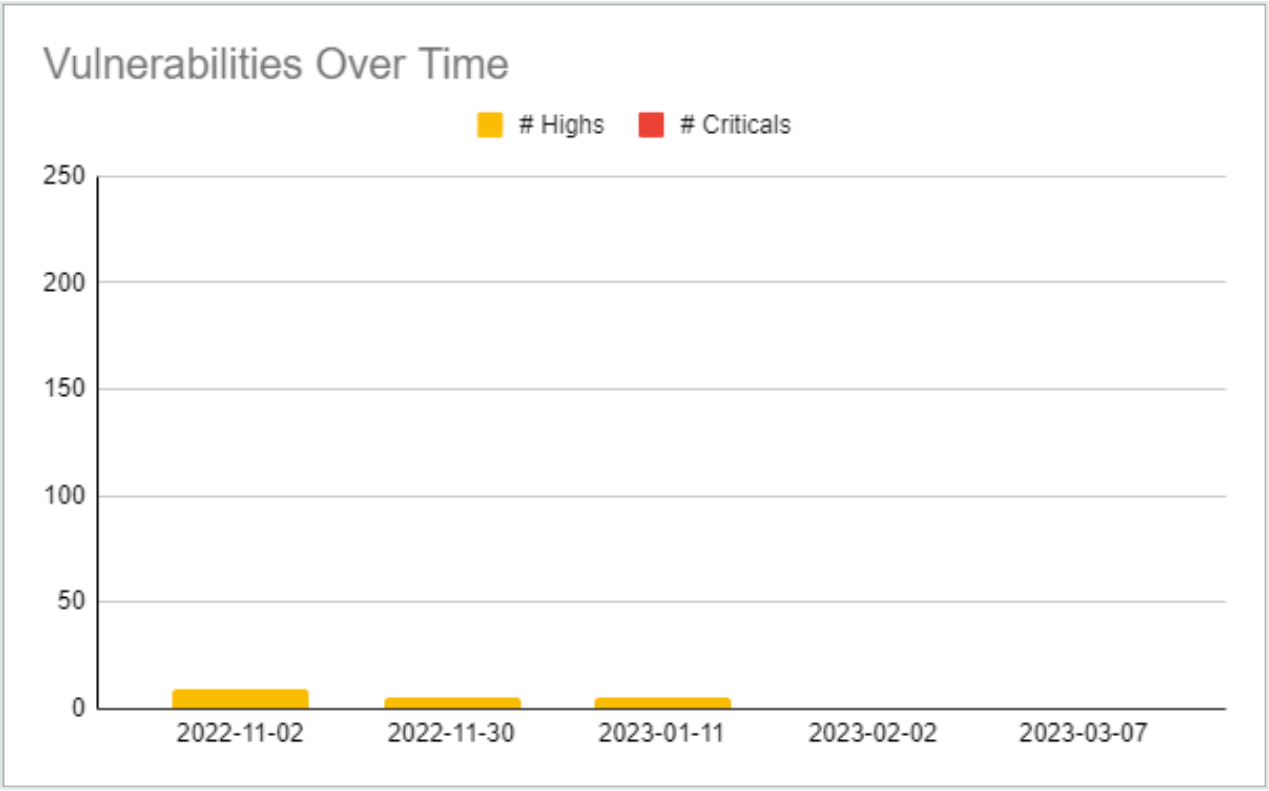
- Fixed an issue where jobs that used union lookback and whose findings were marked as ignored, off-peak, or removed, were incorrectly included in the outlier calculation. (ticket #104746)

Known Limitations

Snowflake Pushdown (beta)

- Mixed datatype rules are not supported in Pushdown.

DQ Security Metrics



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolved	# Highs	# Mediums	Artifact(s)
DQ	2022-11-02	2022.11	0	0	0	9	6	JFrog Artifact
DQ	2022-11-30	2022.12	0	0	0	5	2	JFrog Artifact
DQ	2023-01-11	2023.01	0	1	1	5	2	JFrog Artifact
DQ	2023-02-02	2023.02	0	0	0	0	0	JFrog Artifact
DQ	2023-03-07	2023.03	0	1	1	0	0	JFrog Artifact

2023.02

- Highlights
- New Features
- Enhancements
- Fixes
- Known Limitations
- DQ Security Metrics

Highlights

- To enhance user experience, the React MUI is now on by default across the Collibra DQ app. You can restore the legacy experience by following the steps on the React page.
- You can now use the new /v3/rules/{source}/copy API to bulk copy all rules from an existing dataset to another target dataset. With this new API, you can input a source dataset from which to copy rules, then choose one or more target datasets to which your rules copy.
- You can now leverage the Spring Framework to access controller-level logs and debug API calls.
- The new Oversized Job Report is now available by default in the Reports section. The Oversized Job Report lets you review usage analytics for jobs that use more hardware than what is required for their workloads.
- You can now view and edit your Agent Display Name from the Agent Management screen. This lets you customize your Agent Display Name to make it easier to identify your agent throughout Collibra DQ.
- When running large DQ Spark jobs, you can now bring in on-demand Persistent Volume Claims to use as Spark scratch disk space on Kubernetes deployments.

> Note
> If you only install owlweb or owlagent, you need to copy the owlmanage.sh file from the installation package into the bin directory with the following command:
>
> ```
> cp $PACKAGE_LOCATION/owlmanage.sh $INSTALL_
> PATH/bin/owlmanage.sh
> ```

New Features

Reports

- The new Oversized Job Report is now available by default.

Agent

- You can now view and edit your Agent Display Name from the Agent Management screen.
- When running large DQ Spark jobs, you can now bring in on-demand Persistent Volume Claims to use as Spark scratch disk space on Kubernetes deployments.

APIs

- You can now use the new /v3/rules/{source}/copy API to bulk copy all rules from an existing dataset to another target dataset.
- You can now leverage the Spring Framework to access controller-level logs and debug API calls.

Enhancements

Rules

- Native rules with invalid SQL queries now display an exception message instead of incorrectly displaying a passing status.
- You can now drill into a pulse view chart on the $daysWithoutData stat rules on the Findings page.

Platform

- React MUI is now on by default.
- To ensure proper compliance with Collibra's security protocol, new installations now require you to set up a password that meets Collibra's password policy and a default email address associated with the admin user account.
    - Instructions for providing these credentials are interactive as part of the setup script for Standalone installs.
    - For new installs in non-interactive mode, such as automated deployment, export the following variables, as they are required for the script to execute correctly:

        - `export DQ_ADMIN_USER_PASSWORD`
        - `export DQ_ADMIN_USER_EMAIL`
    - For Cloud native installs, you must use the following Helm variables to set the password and default admin email:
        - `--set global.web.admin.email=${email} \`
        - `--set global.web.admin.password=${password} \`
    - For multi-tenant instances, admin users are created via the UI, not with this process.

Security

- To ensure proper compliance with Collibra's security protocol, all deployments of Collibra DQ on Kubernetes now use a read-only root filesystem.
    - All ephemeral storage directories, such as /opt/owl/log, /tmp/scratch, and /opt/owl/cache, are still writable for Collibra DQ to function correctly.
    - The read-only root filesystem is enforced on all long-running pods except for Spark jobs and Livy sessions.
    - Ensure that you pick the corresponding Helm charts for a successful upgrade.
        - Any customizations must be ported to the corresponding Helm charts. For additional assistance, please contact Collibra Professional Services.
        - You can no longer upload custom JDBC drivers under the /opt/owl/drivers folder. Refer to the documentation for more details on how to configure custom JDBC drivers.

> Note Standalone deployments of Collibra DQ are not affected by these updates.

Snowflake Pushdown (beta)

- You can now scan for categorical outliers.
- You can now apply weighting and boundaries as part of the Outlier layer.

Fixes

Rules

- Fixed an issue where regex rules, which worked in previous DQ versions, appeared as exceptions after upgrading to the 2022.09 version. (ticket #102615)

Dupes

- Fixed an issue that prevented jobs from running with Dupes check switched on for tables with only one column. (ticket #104585)

Platform

- Fixed an issue with the print statement in the stage 3 logs where an excessive number of log records were written due to too many observations. (ticket #104081)

Security

- Fixed an issue with role mapping where invalid values for `LDAP_GROUP_RESULT_DN_ATTRIBUTE` and `LDAP_GROUP_RESULT_CONTAINER_BASE` set to the base OU of the AD groups caused the role mapping to fail.

Connections

- Fixed an issue with connections using Kerberos authentication that prevented jobs from running because of IT restrictions around writing to Openshift containers. (ticket #102964)
- Fixed an issue with Databricks cluster connections to allow `PWD={}` input in the connection string URL. (ticket #104845)
    - You are still required to enter `PWD={your-text-here}` in the URL.
    - This fix only applies to Databricks cluster connections, not to Databricks connections that use SQL-endpoints.

- The logging of entire ServiceNow response messages has been added to help identify errors.

Agent

- Fixed an issue where associating or disassociating connections from a remote Agent resulted in a PreparedStatementCallback error, which prevented connections from being deleted. (tickets #102322, 102763)

Snowflake Pushdown (beta)

- Fixed an issue with Min and Max on the Profile page that caused strings with numerical values to display invalid results. (ticket #105935)

Known Limitations

Rules

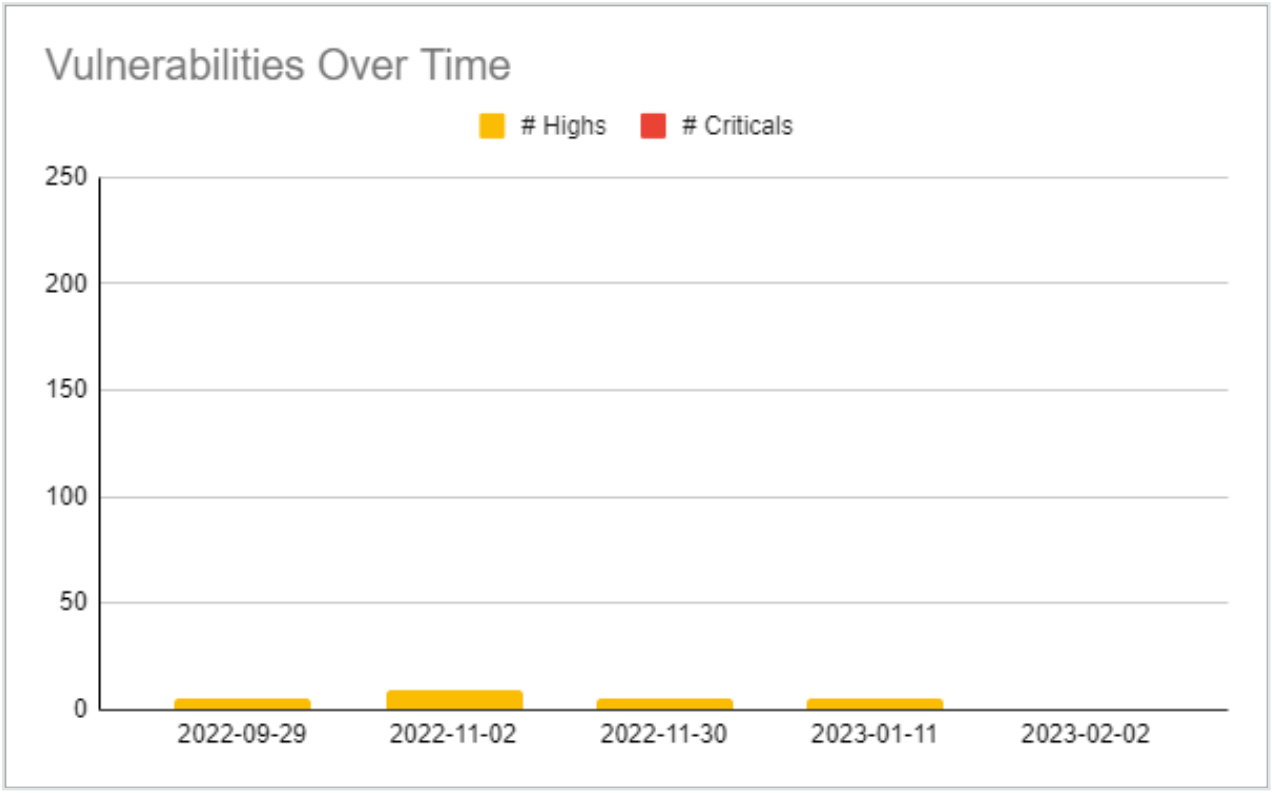
- The $daysWithoutData stat rule has a limitation where rules that are not named "daysWithoutData" do not display their Pulse View charts when you drill into the rule on the Findings page. A workaround for this is to name your stat rule exactly as it appears here: `daysWithoutData`.

Platform

The dollar sign `$` is a special character that must be escaped with a backslash `\` when running the setup script or Helm via the command line.

DQ Security Metrics

The following image shows a chart of Collibra DQ security vulnerabilities arranged by release version.



The following image shows a table of Collibra DQ security metrics arranged by release version.

Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolved	# Highs	# Mediums	Artifact(s)
DQ	2022-09-29	2022.10	0	0	0	5	13	JFrog Artifact
DQ	2022-11-02	2022.11	0	0	0	9	6	JFrog Artifact
DQ	2022-11-30	2022.12	0	0	0	5	2	JFrog Artifact
DQ	2023-01-11	2023.01	0	1	1	5	2	JFrog Artifact
DQ	2023-02-02	2023.02	0	0	0	0	0	JFrog Artifact

2023.01

New Features

Admin

- Admin users can now view and modify the DQ license key and license name from the new License page.

Platform

- You can now deploy new Helm charts on new and existing releases of Collibra DQ Cloud Native deployments without encountering character size limitations.

Reports

- Four new reports are now available by default:
  - The Missing Jobs Report shows jobs that were expected to run but didn't run as scheduled.
  - The Hardware Usage Report shows the datasets that required the most total cores to run and more general hardware usage statistics.
  - The Observability Score Roll-Up Report shows the aggregated scores of all AdaptiveRules (all datasets + all columns) and averages passing and breaking for all columns over 30 days.
  - The Rules Passing Fraction Roll-Up Report shows all the passing rows and total rows scanned for user-defined rules aggregated by dimensions over the past 30 days.

Enhancements

Rules

- The Export LinkIds button is now displayed within the Rules tab of the Findings page. This button was previously only available under the Breaks tab on the Rule Builder page.
  - You can enable LinkIDs from the Scope workflow on the Explorer page to export LinkIds for rule break records.
- The Copy Rules API (/v3/rules/copy) now has the following enhancements:
  - Copied rules now copy to a new dataset correctly regardless of their rule type.
  - Security logs for every rule copy request are now available for admin users to review in the Audit Trail section of the Admin Console.

Explorer

- When creating a DQ Job for a table with no rows, the columns are now shown in the Scope section.

Platform

- A new property, `LOCAL_REGISTRATION_ENABLED`, in the owl-env.sh script and K8s config map is now available to display or hide the registration link on the Sign in page for local users.
  - For owl-env.sh:
    - The command `export LOCAL_REGISTRATION_ENABLED=true` allows the registration link to display on the Sign in page.

      > Note Because the registration link is visible by default, this property is also set to true by default.

    - The command `export LOCAL_REGISTRATION_ENABLED=false` hides the registration link from the Sign in page.
  - For K8s:
    - The configuration `LOCAL_REGISTRATION_ENABLED:"true"` allows the registration link to display on the Sign in page.

      > Note Because the registration link is visible by default, this property is also set to true by default.

    - The configuration `LOCAL_REGISTRATION_ENABLED:"false"` hides the registration link from the Sign in page.

Connections

- SPARK322 and SPARK320 are now shipped with a Spark JDBC connection provider for Standalone deployments.
- When you bring Collibra DQ jars into Databricks, you are now required to set the property

`spark.sql.-sources.disabledJdbcConnProviderList='basic,oracle,mssql'` at either the Spark Cluster-level or the SparkSession-level before using Collibra DQ's set of functions for Spark profiles 3.2.1 and onwards.

Admin

- You can now sort the Date column on the Usage page. Dates now appear in descending order by default.

Fixes

Rules

- Fixed an issue that prevented the descriptions of saved rules from being edited when the rule name contained a greater than symbol after a single quote. (ticket #100114)
- Fixed an issue with rule builder validation that caused a rule syntax exception message to throw. (tickets #99735, 101165)
- Fixed an issue with rules with complex conditions (multiple rlike strings) for Freeform SQL rules that resulted in an exception message. (ticket #100116)

DQ Job

- Fixed an issue where behavioral observations made for a dataset did not subtract points from the data quality score. (ticket #98539)

Alerts

- Fixed an issue that limited the ability to edit or delete alerts with names containing apostrophes from the UI. (ticket #98864)

Outliers

- Fixed an issue where recalibrating an outlier would bulk apply downtrain labeling to it. (ticket #100085)

Reports

- Fixed an issue where Completeness Reports were not generated when the Custom Range filter was used. (ticket #99786)

DQ Connector

- Fixed an issue with the Collibra DQ - Collibra Data Intelligence Cloud integration that prevented Rules and Charts from importing. (ticket #104872)

Admin

- Fixed an issue with time-based data retention when using linkId that caused too many break records to store in the metastore. (ticket #99072, 102900)

Known Limitations

Rules

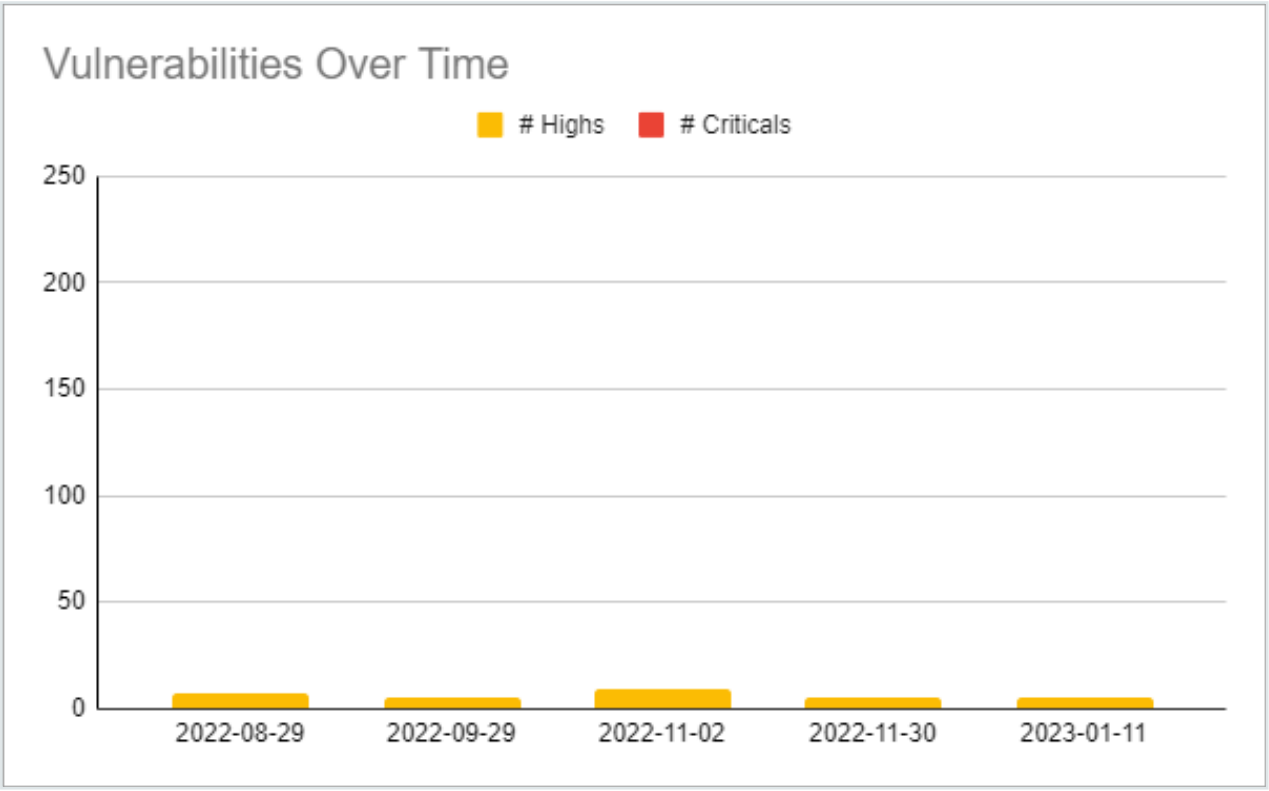
- The new Export LinkIds button generates a CSV file limited to viewing only via a spreadsheet program, like Excel.
    - A workaround is to Save/Export the CSV file from the spreadsheet program in order to allow viewing in general text editors.

DQ Job

- Remote file jobs with headers containing white spaces fail with a requirement failed exception message.

- A workaround is to edit the DQ Job command line in the Run CMD tab and place single quotes around the column name in `-q` and double quotes around the entire `-header` flag.

DQ Security Metrics



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolved	# Highs	# Mediums	Artifact(s)
DQ	2022-08-29	2022.09	0	0	0	7	12	JFrog Artifact
DQ	2022-09-29	2022.10	0	0	0	5	13	JFrog Artifact
DQ	2022-11-02	2022.11	0	0	0	9	6	JFrog Artifact
DQ	2022-11-30	2022.12	0	0	0	5	2	JFrog Artifact
DQ	2023-01-11	2023.01	0	1	1	5	2	JFrog Artifact

Hotfixes

Collibra Data Quality & Observability 2023.01.1

- Fixed an issue with Databricks cluster connections to allow `PWD={}` input in the connection string URL. (ticket #104845)

    - You are still required to enter `PWD={your-text-here}` in the URL.
    - This fix only applies to Databricks cluster connections, not to Databricks connections that use SQL-endpoints.

2022.12

New Features

Explorer

- The types of queries that can run from the View Data page are now restricted to read-only queries only.

APIs

- You can now copy SQLG- and SQLF-type rules from an existing dataset to another existing dataset with the /v3/rules/copy API call.

Connections

- You can now create a MongoDB connection with a CDATA driver.

Snowflake Pushdown (beta)

- You can now detect outliers when running a Pushdown job.

Enhancements

DQ Job

- All tables on the Jobs page now include pagination, dropdown filters, and the ability to export.

Rules

- Rules associated with datasets with zero rows now execute successfully.
- Stat rule evaluation on secondary datasets is now supported for SQLF rules.

Profile

- You can now view run execution details and stale data by toggling the box chart on the findings page.

APIs

- The getRecords notebook API function is now updated and the getGeneric query is renamed getDupesPreview.
- You can now obtain rules from a dataset and reassign them to another dataset with the following Databricks notebook API functions:
  - `def addRules(rules: List[Rule], dataset: String): Owl`
  - `def getRulesDfByDataset(dataset: String): DataFrame`
  - `def getRulesByDataset(dataset: String): List[Rule]`
  - `def getRuleNamesByDataset(dataset: String): DataFrame`

Platform

- TechPreview (TP) labels are now removed from the UI.

Connections

> Warning As of September 2022, Databricks JDBC driver version 2.6.27 is packaged as part of both standalone and Kubernetes download packages. The Databricks Simba driver (version 2.6.22) is no longer packaged for Kubernetes. As a result of this change, the Databricks connection template has changed, and any existing connection using the old driver (2.6.22) must be updated. For more information on updating your drivers, refer to Standalone Upgrade.

- The Databricks SQL endpoint is now supported for JDBC connections.
- 3DES and DES encryption cipher for Kerberos authentication types are no longer supported because of recent Red Hat OS (RHEL 8.7) cipher deprecation.

Fixes

Explorer

- Fixed an issue with the `-rdEnd` variable in the command line the variable in the query to be improperly escaped. (ticket #98702)

Profile

- Fixed an issue where the confidence score (Conf) displayed values greater than the threshold of 100. (ticket #99636)
- Fixed an issue where HTML in data fields was rendered on the Data Preview section of the findings page. (ticket #97883)
- Fixed an issue with Data Preview that resulted in an OOM error when the data_preview table contained a large number of records.

Rules

- Fixed an issue where values on the Rules tab did not correctly display in scientific notation format. (ticket #89738)
- Fixed an issue when using a secondary dataset that prevented @dataset for primary dataset from being supported.

Scorecards

- Profile is now removed from the DQ Scorecards submenu.

Security

- Fixed an issue with LDAP external groups to role mappings when there was no fully qualified path for the LDAP group that caused malformed API calls and did not save properly.
- Fixed an issue with the Dataset Security feature. (ticket #100317)
    - When the following security settings are configured, the system fully restricts access to the findings page for admin users:
        - Dataset security is turned on.
        - Default owner access is unchecked.
        - Dataset belongs to no roles or no roles to which the user has access.

APIs

- Fixed an issue with the /v2/gethints endpoint that prevented the Hints table from displaying correctly on the findings page. (ticket #98941)
- Fixed an issue with the getRecords and getGenerics APIs that prevented any information from being returned. (ticket #98820)

Alerts

- Fixed an issue with the SQL on the Alert Notifications page that prevented data from appearing in the DataTables error message.

Agent

- Fixed an issue with GKP deployments where job scans failed because the driver pod could not create connections to the metastore. (ticket #102175)

Validate Source

- Fixed an issue where the Source to Target scorecard incorrectly displayed a mismatch because of an unexpected column type checked during a schema order check. (ticket #98300)

Connections

- Resolved connection issues in certain cases by upgrading the Athena driver to version 2.0.33. (ticket #100340)
- Fixed an issue where HDFS connections could not rerun a job successfully because certain parameters were automatically appended to the Free Form (Appended) field of the Agent configuration. (ticket #95810)
- Fixed an issue with Dremio connection timeouts on Kubernetes deployments. (ticket #101221)
    - To prevent Dremio connection issues, set the following value in the Free Form (Appended) field of the Agent configuration:
      `-conf spark.driver.extraJavaOptions=-Dcd-jd.io.netty.tryReflectionSetAccessible=true`

Known Limitations

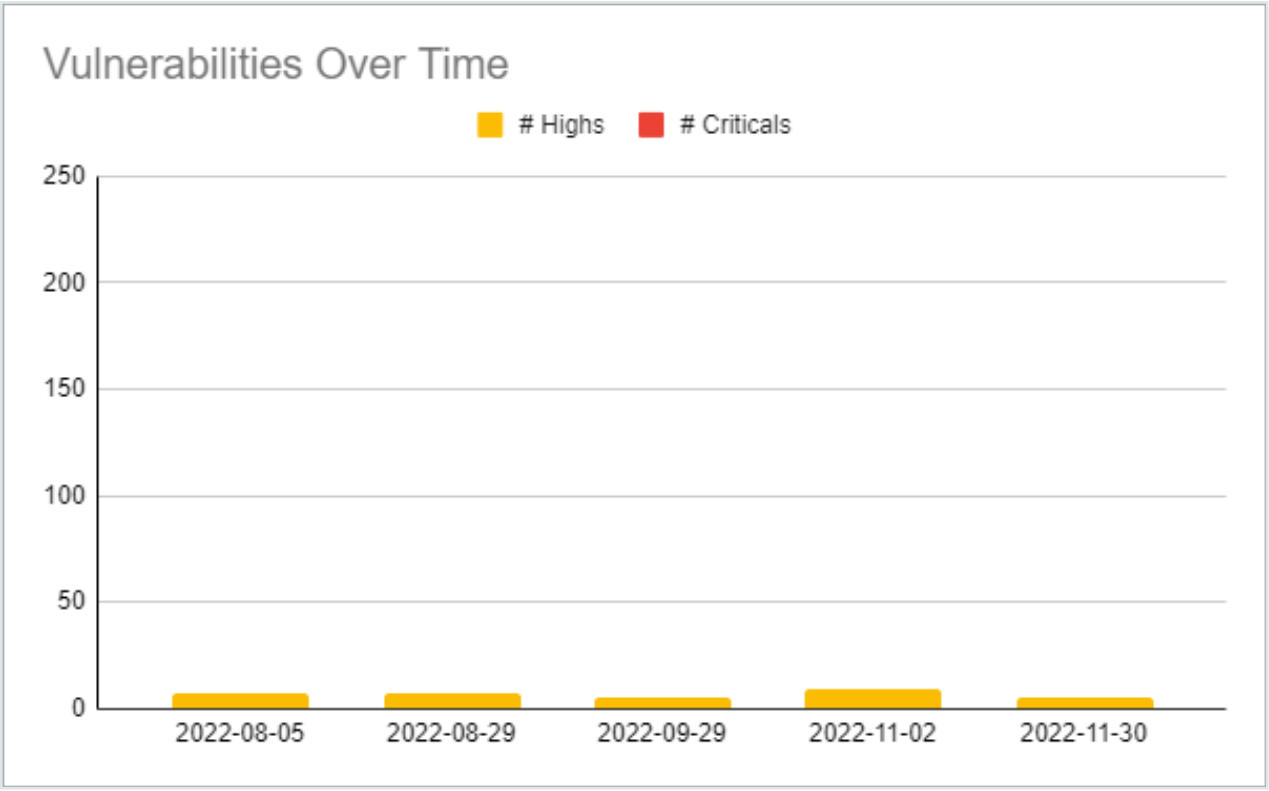
Rules

- Freeform rules with fully qualified column names are currently unsupported when they use the following syntax:
  `select <column name> FROM @<dataset name> WHERE @<dataset name>.<column name> condition`
    - A workaround to this limitation is to use aliasing instead.

APIs

- When using the new /v3/rules/copy API, the copied rule automatically appends "copied" to the rule name. After copying a rule, you may need to manually update the rule name.
    - If the copied rule is performed on a target dataset that does not conform to the compatible columns, then you need to manually update the rule to ensure the columns are compatible across datasets.
    - Dataset Security is not enforced when using the /v3/rules/copy API.

DQ Security Metrics



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolved	# Highs	# Mediums	Artifact(s)
DQ	2022-08-05	2022.08	0	0	0	7	14	JFrog Artifact
DQ	2022-08-29	2022.09	0	0	0	7	12	JFrog Artifact
DQ	2022-09-29	2022.10	0	0	0	5	13	JFrog Artifact
DQ	2022-11-02	2022.11	0	0	0	9	6	JFrog Artifact
DQ	2022-11-30	2022.12	0	0	0	5	2	JFrog Artifact

Hotfixes

Collibra Data Quality & Observability 2022.12.3

- Fixed an issue with Databricks cluster connections to allow `PWD={}` input in the connection string URL. (ticket #104845)
  - You are still required to enter `PWD={your-text-here}` in the URL.
  - This fix only applies to Databricks cluster connections, not to Databricks connections that use SQL-endpoints.

2022.11

Warning
The MS SQL driver that comes with JDK11 standalone packages does not currently work in the JDK11 environment. MSSQL requires a separate JAR for JDK11. Please contact your Customer Success Manager for the compatible driver.

Dremio is not currently supported for JDK11 standalone packages. If you plan to run JDK11, add `-Dcdjd.io.netty.tryReflectionSetAccessible=true` to owlmanage.sh as a JVM option for your Web and Spark instances. Please contact your Customer Success Manager for assistance.

Dremio jobs currently fail on both K8s and standalone JDK11 deployments. Add the following config to the Free Form (Appended) field of the Agent Configuration template: `-conf spark.driver.extraJavaOptions=-Dcdjd.io.netty.tryReflectionSetAccessible=true`.

As of October 18, 2022, all images for the 2022.10 release have a Critical CVE (CVE-2022-42889). If you picked up the 2022.10 release before October 18, 2022, there should be no issue with your scans. If issues persist, please contact your Customer Success Manager for a new build.

Note
After you complete an upgrade or a new installation of Collibra DQ, you are now required to enter a license name by following either a one-time prompt on the login page, entering the `LICENSE_NAME` environment variable in the environment variable file (owl-env.sh), or by entering the `global.configMap.data.license_name` Helm chart variable. Your license name is the value after `YOUR NAME IS =` found in the license provision email sent to you by Collibra. Customers who do not have this information due to being issued a license before March 2022 should input license information following the format below.

For a single instance: <yourcompanyname>
For multiple instances: <yourcompanyname>-dev, <yourcompanyname>-test, <yourcompanyname>-prod
No spaces or special characters are permitted except for hyphens -.

> Note Before you execute the setup.sh script, update the `SPARK_PACKAGE` variable to the desired Spark package, for example `spark-3.2.2-bin-hadoop3.2.tgz`. You can check the `../packages` folder for the Spark package that was previously downloaded. You do not need to update the setup.sh script if you use the following export statement:
> `export SPARK_PACKAGE="spark-3.2.2-bin-hadoop3.2.tgz"`

New Features

Platform

- The following pages now support the new React MUI:
  - Scorecards
  - List View
  - Assignments
  - Pulse View
  - Alerts

> Note React is turned off by default for the 2022.11 release. If you would like to try the new React pages, you can toggle it on from the Admin Console, or contact your Customer Success Manager for assistance.

DQ Job

- You can now terminate jobs from the Jobs page if they are in progress, incorrectly submitted, or stuck in Staged status. When you terminate a job, two alerts are generated.
  - Jobs in the Spark UI display Finished statuses, even though they are terminated from the DQ UI.

Alerts

- You can now generate alerts for the following stale data stat rules:
  - `$daysWithoutData`
  - `$runsWithoutData`

    - `$daysSinceLastRun`
- You can now generate alerts for jobs stuck in Staged status for more than one hour.

Admin

- You can now configure LDAP for user access in multi-tenant environments.

Connections

- You can now use key-pair authentication for Snowflake connections.
    - When you append to the Connection URL string, your entry must be comma separated.
    - When you manually modify the Driver Properties field, your entry must be semi-colon separated.
- CDATA connections are now supported in standalone deployments.
    - CDATA drivers are now included in the release package.

Cloud Storage

- Azure Blob Storage is now a supported target storage system.

Snowflake Pushdown (beta)

- Schema Change monitoring from the AdaptiveRules tab is now enabled by default.
    - Schema is now separated from basic profiling.
- The new DatasetDefDTO API now returns Pushdown information.
- Dataset security checks are now implemented for Pushdown jobs.

Enhancements

Explorer

- The Job Estimate dialogue now has improved guidance on executors and cores. The Job Estimate now estimates when a max core, max executor, and max memory is

reached.

DQ Job

- Job schedule time zone is now a read-only field and can no longer be configured. Existing scheduled jobs reflect their current settings, but all other scheduled jobs are now based on the time zone of the DQ server (UTC). (ticket #88797, 89736, 92611, 95231)

Dupes

- A new warning message now displays when increasing the duplicate check limit from the UI. (ticket #95604)

Security

- Kubernetes service accounts associated with AWS IAM pod roles for controlling access to AWS services for cloud native DQ deployments on AWS EKS are now supported.
- When DATASET SECURITY is enabled, DATASET ACCESS is now required to edit, map, or retrieve datasets or business units. (ticket #92934)

Fixes

Rules

- Fixed an issue that prevented freeform rules containing double backslashes from saving. (ticket #96636, 96640)
- Fixed an issue that caused rules containing open brackets ([ ) to display break records incorrectly. (ticket #94399)
- Fixed an issue that caused rules containing regex to throw out of range exceptions. (ticket #98435)

DQ Job

- Fixed an issue where run time was not displayed on the findings page because run_id column type in the metastore did not include time zone. (ticket #96050)
- Fixed an issue that caused Parquet files to fail during the LOAD activity. (ticket #96191)
    - Other NFS file types, including ORC, CSV, and Avro, also run successfully.

Alerts

- Fixed an issue when saving batch names that used spaces between delimiters, which caused an invalid error to occur. (ticket #97028)

Validate Source

- The Add Column Names feature is now removed from the Source tab. (ticket #96066)
    - Instead, use the query to edit/limit columns or use Update Scope.
- Fixed an issue where disabling source check on a cloned dataset resulted in an error. You can now disable source validation on cloned datasets. (ticket #97795)

Dupes

- The Advanced Filter is now hidden from the Dupes tab. (ticket #96065)

Shapes

- Fixed an issue when editing a dataset that reverted the Shape Detection setting (Off, Auto, or Manual) applied when it was created. (ticket #95471, 95473)

Schema

- Fixed an issue with schema detection on files where schema detection was performed on all columns when a subset of columns was selected. (ticket #92476)

    - Use the`headercheckoff` flag when it is necessary to see only when columns are added or dropped.
- Fixed an issue where schema changes were not correctly identified and updated. (ticket #96013)

Behavior

- Fixed an issue with behavior lookback(`-bhlb`) that caused Row Count changes to be misrepresented. (ticket #94840)

Connections

- Azure Blob connections in standalone environments require the following jars to be added to the `$SPARK_HOME/jars` folder:
    - hadoop-azure-3.2.0.jar
    - wildfly-openssl-1.1.3.Final.jar

API

- Fixed an issue with the DB import process to ensure JobSchedule records import without error. (ticket #98405)

Known Limitations

DQ Job

- Job termination is not supported for jobs in Unknown status.

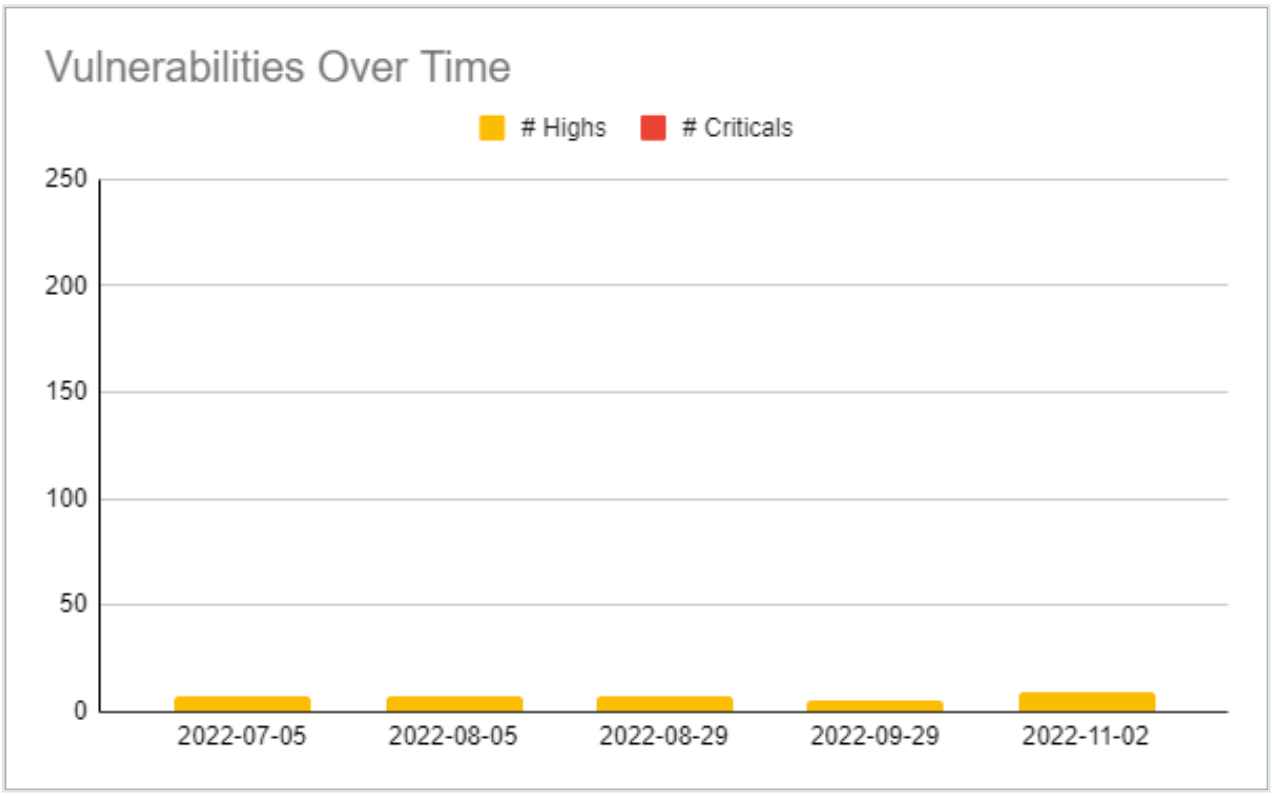
Validate Source

- Cloning and saving, enabling, or disabling the source tab is associated with the original dataset name and fails on the screen when an update is made, but does not affect the actual job run.

Connections

- When adding driver properties using the +Add Property option for Snowflake connections, semicolons are incorrectly appended to key values. Instead, use comma format to separate key values.

DQ Security Metrics



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolved	# Highs	# Mediums	Artifact(s)
DQ	2022-07-05	2022.07	0	0	0	7	13	JFrog Artifact
DQ	2022-08-05	2022.08	0	0	0	7	14	JFrog Artifact
DQ	2022-08-29	2022.09	0	0	0	7	12	JFrog Artifact
DQ	2022-09-29	2022.10	0	0	0	5	13	JFrog Artifact
DQ	2022-11-02	2022.11	0	0	0	9	6	JFrog Artifact

2022.10

New Features

> Warning For the Collibra Data Quality 2022.10 release, all Docker images run on JDK11. Standalone packages contain JDK8 and JDK11 options. If you are an existing customer who requires JDK11, please upgrade your runtime before upgrading to 2022.10. Most Hadoop environment versions (EMR/HDP/CDH) still run on JDK8, so customers using these environments can upgrade with the JDK8 packages. If you prefer to upgrade to JDK11, you must follow the documentation of your respective Hadoop environment to upgrade to JDK11 before deploying the 2022.10 release.
>
> The MS SQL driver that comes with JDK11 standalone packages does not currently work in the JDK11 environment. MSSQL requires a separate JAR for JDK11. Please contact your Customer Success Manager for the compatible driver.
>
> Dremio is not currently supported for JDK11 standalone packages. If you plan to run JDK11, add `-Dcdjd.io.netty.tryReflectionSetAccessible=true` to owlmanage.sh as a JVM option for your Web and Spark instances. Please contact your Customer Success Manager for assistance.
>
> As of October 18, 2022, all images for the 2022.10 release have a Critical CVE (CVE-2022-42889). If you picked up the 2022.10 release before October 18, 2022, there should be no issue with your scans. If issues persist, please contact your Customer Success Manager for a new build.

Rules

- You can now define a rule to detect the number of days a job runs without data by using `$daysWithoutData`.
- You can now define a rule to detect the number of days a job runs with 0 rows by using `$runsWithoutData`.
- You can now define a rule to detect the number of days since a job last ran by using `$daysSinceLastRun`.

Profile

- You can now use a string length feature by toggling the Profile String Length check-box when you create a data set.
    - When Profile String Length is checked, the min/max length of a string column is saved to table dataset_field

Validate Source

- You can now write rules against a loaded source data frame when `-post-clearcache` is configured in the agent.

> Note The DQ UI will be converted to the React MUI framework with the 2022.11 release. Prior to the 2022.11 release, you can turn the React flag on, but note that some features may be temporarily limited.

Enhancements

DQ Job

- Start Time and Update Time are now based on the server time zone of the DQ Web App.

Scheduler

- The Job Schedule page now has pagination.

Scorecards

- From Pulse View, you can now view missing runs, runs with 0 rows, and runs with failed scores.

Admin/Catalog

- Connection details are now masked when non-admin users attempt to view or modify database connection details from the Catalog page. Only users with role_admin or role_connection_manager have the ability to view connection details on this page. (ticket #94430)

API

- The /v2/getRunIdDetailsByDataset endpoint now provides the following:
  - The RunIDs for a given data set.
  - All completed DQ Jobs for a given data set.

Snowflake Pushdown (beta)

- You can now detect shapes that do not conform to a data field. Pushdown jobs scan all columns for shapes by default.
- You can now view Histogram and Data Preview details for the Profile activity.

Connections

- The Snowflake JDBC driver is now updated to 3.13.14.

Fixes

Rules

- Fixed an issue with the Rule Validator that resulted in missing table errors. The Validator now correctly detects columns. (ticket #93430)

DQ Job

- Fixed an issue that caused queries with joins to fail on the load activity when Full Profile Pushdown was enabled. Pushdown profiling now supports SQL joins. (ticket #92409)
- Fixed an issue that caused jobs to fail at the load activity when using the CTE query. Please note that CTE support is currently limited to Postgres connections. (ticket #88287, 89150)
- Fixed an issue that caused inconsistencies between the time zones represented in the Start Time and Update Time columns.

Agent

- Fixed the loadBalancerSourceRanges for web and spark_history services in EKS environments. (ticket #95398)
    - The helm property `global.ingress.*` has been removed to separate the config for web and spark_history. Please update the property as follows:__ `global.web.ingress.*``global.spark_history.ingress.*`
- Added support to specify the inbound CIDRs for the Ingress using the property `.global.web.service.loadBalancerSourceRanges`. (ticket #95398)
    - Though Ingress is supported as part of Helm charts, we recommend attaching your own Ingress to the deployment if you need further customization.
    - This requires a new Helm chart.
- Fixed an issue that caused Livy file estimates to fail for GCS on K8s deployments.
- Fixed an issue that caused jobs to fail for GCS on K8s deployments.

Validate Source

- The Add Column Names feature is scheduled for removal with the upcoming 2022.11 release. (ticket #96066)
    - This was a previous functionality before being able to limit the query directly (`srcq`) and Update Scope was added.
    - Use the query to edit/limit columns and also use Update Scope.

- Fixed an issue that caused the incorrect message to display for [VALUE_THRESHOLD] when validate source was specified for a matched case. (ticket #94435)

Dupes

- The Advanced Filter is scheduled for removal from the Dupes page with the upcoming 2022.11 release. (ticket #96065)

Explorer

- Fixed an issue that caused BigQuery connections to incorrectly update the library (`-lib`) path when a subset of columns was selected. (ticket #96768)

Scheduler

- Fixed an issue that prevented the scheduler from running certain scheduled jobs in multi-tenancy setups. Email server information is now captured from the correct tenant. (ticket #92898)

Known Limitations

Rules

- When a data set has 0 rows returned, stat rules applied to the data set are not executed. While a full fix is planned for a future release, this limitation is only partially fixed as of 2022.10.

DQ Job

- CTE query support is currently limited to Postgres connections. DB2 and MSSQL are currently unsupported.

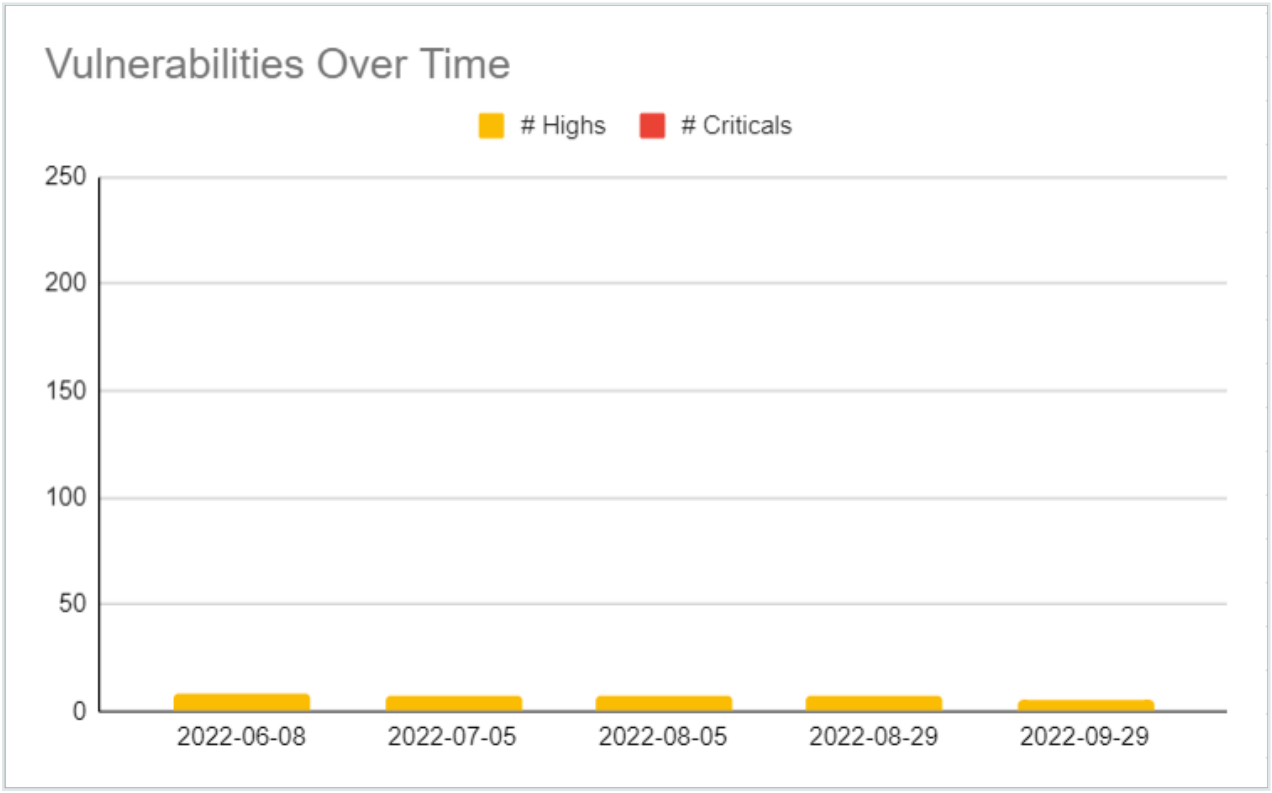
Catalog

- When using the new bulk actions feature, updates to your job are not immediately visible in the UI. Once you apply a rule, run a DQ Job against that data set. From the Rules tab, a row with the newly applied rule is visible.

Snowflake Pushdown (beta)

- Freeform (SQLF) rules cannot use a data set name but instead must use `@dataset` because Snowflake does not explicitly understand data set names.
- When using the SQL Query workflow, selecting a subset of columns in your SQL query must be enclosed in double quotes to prevent the job from running infinitely and without failing.
- Min/Max precision and scale are only calculated for `double` data types. All other data types are currently out of scope.

DQ Security Metrics



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolved	# Highs	# Mediums	Artifact(s)
DQ	2022-06-08	2022.06	0	0	0	8	13	JFrog Artifact
DQ	2022-07-05	2022.07	0	0	0	7	13	JFrog Artifact
DQ	2022-08-05	2022.08	0	0	0	7	14	JFrog Artifact
DQ	2022-08-29	2022.09	0	0	0	7	12	JFrog Artifact
DQ	2022-09-29	2022.10	0	0	0	5	13	JFrog Artifact

Collibra DQ release archive

2022.09

Enhancements

Rules

- The Conditions column on the Rules tab now displays SQLG and SQLF rule definitions on hover.

DQ Job

- The Jobs chart now shows a dotted gray line to represent jobs in Submitted status.
- The Jobs chart now supports an hourly view option.
- When you run a Pushdown Job that has a data set that returns 0 rows, an unclear message displays.

Schema

- From the Config tab in Explorer, a Check Header checkbox under DQ Job is now available for when column names contain special characters. The Check Header checkbox is checked by default.
    - When checked, schema findings do not display when detected.
    - When unchecked, schema findings display when detected.

Behavior

- Mean values are now rounded on the Findings page.

Explorer

- SOH delimiters for files are now supported.
- The Only checkbox on all Build Layer tabs is now removed.
- The Profile activity is now always enabled and no longer has an on/off switch.

Alerts

- Only one email per alert is now sent when alerts are set up for a scheduled job.
- You can now check the logs to see when an alert does not send in order to resend the email.

Scheduler

- The findings page now displays a green indicator next to the Schedule icon when you schedule a job to run automatically. If Scheduler is inactive, a red indicator displays.

API

- The v2/gethoot API now properly returns rule dimension information for data sets. (ticket #89973)

Connections

- The Databricks connection template has changed, due to an upgrade of the driver. Any existing connection that uses the old driver must be updated. Refer to the new template. (ticket #19950)
- The drivers for Athena, BigQuery, MongoDB, GCS, Hive/Impala were also upgraded but no connection change is required.

Spark

- The 2023.04 release uses Spark 3.2.2.

> Note We recommend using Spark 3.x for standalone installs/upgrades.

Fixes

Explorer

- Fixed an issue that prevented the Job Estimator from properly displaying row estimates when the run date was modified during a new job run. (ticket #90860)
- Fixed an issue that prevented DQ jobs created using NFS connection types from displaying under the Remote File Connections dropdown. (ticket #92479)
- Fixed an issue that caused the file type parser to throw an error message when the default comma delimiter was not detected. The parser now detects a file's delimiter and updates the delimiter type in the UI automatically. (ticket #89489, 92480)

Files

- The error message for Failed Merging Schema now has extra logging to clarify the cause of failed schema merges for both Livy sessions and non-Livy paths. (ticket #92694)

Security

- Fixed an issue with the v2/getcatalogtableshasrulesfromcxn API that triggered a 403 status code when Dataset Security was enabled. (ticket #93298, 94258)

Agent

- Fixed an issue that caused the Agent Check to no longer attempt check-ins to the metastore on K8s deployments, which resulted in red (unhealthy) status. (ticket #92055, 92963)
- Fixed an issue that prevented concurrent users from properly running Livy sessions. (ticket #92963, 90432)

Known Limitations

Rules

- The Rule Builder page becomes unusable if the user creates, validates, saves a new rule and then re-edits.
    - The workaround for this limitation is to do a full page refresh.
- When a user attempts to validate a rule that contains a stat, an exception error is returned.

Security

- The Assignments Queue feature is only available for local users. Support for externally connected users, such as SAML and AD connector, is not currently available.

Alerts

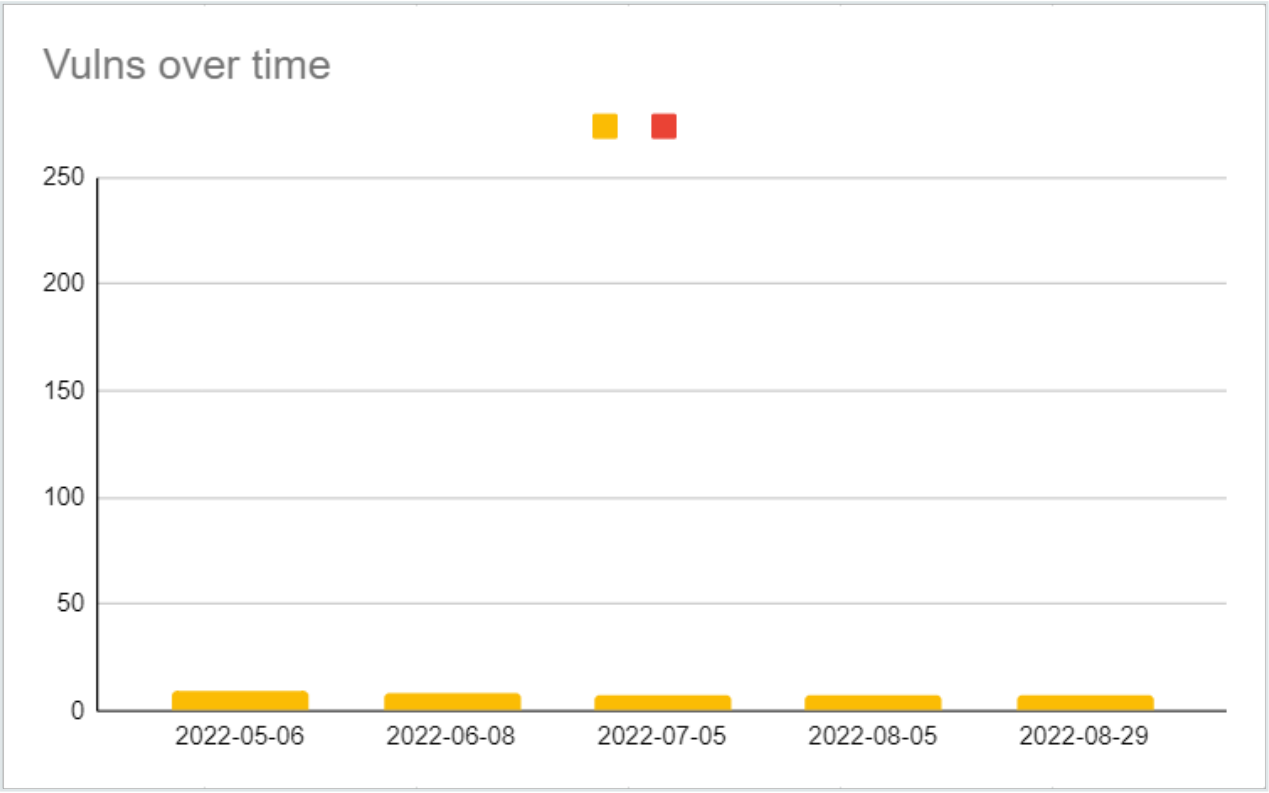
- When alert recipient email addresses are separated by semicolons `;`, alerts emails are not sent to the intended recipients.
    - A workaround for this limitation is to separate alert recipient email addresses with commas `,` instead of semicolons.

Snowflake Pushdown

- When a Job is run, which has a data set that returns 0 rows, an unclear message displays.
- When a native rule is created that contains an embedded stat, its calculated value will not display on the Job results page.
- Data Set security is not supported.
- Disabling autometrics will not take effect, therefore, all autometrics are executed.

- Creating a DQ job using only "SQL Query" workflow doesn't allow you to set the rundate value.

DQ Security Metrics



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resol	# Highs	# Mediums	Artifact(s)
DQ	2022-05-06	2022.05	0	0	2	9	19	JFrog Artifact
DQ	2022-06-08	2022.06	0	0	0	8	13	JFrog Artifact
DQ	2022-07-05	2022.07	0	0	0	7	13	JFrog Artifact
DQ	2022-08-05	2022.08	0	0	0	7	14	JFrog Artifact
DQ	2022-08-29	2022.09	0	0	0	7	12	JFrog Artifact

2022.08

New Features

Rules

- You can now write SQLG-type Stat Rules on mean.

Enhancements

Connections

- You can now authenticate Oracle JDBC connections with Kerberos TGT, Keytab, and Password. (tickets #75267, 76030)
- You can now authenticate SQL Server JDBC connections with Kerberos Keytab in addition to basic authentication.

Rules

- Rule Summary enhancements:
  - You can now select different time periods for analysis.
  - You can now view charts from three different pages, including Rule Detail Summary, Rule Breaks, and Rule Dimension Summary.

Security

- Vulnerabilities identified by Jfrog
  - Vulns 0, criticals 0, high severity 7
  - The majority of the current mediums are due to merging the dq-streaming module into core.
  - For a visual readout, see the DQ Security Metrics section below.

Agent

- You can now optionally configure individual time zones of DQ Job, Web, and Agent. You should only use this configuration when your instance and containers run in different system time zones. (tickets #87024, 87155)

Behavior

- The Behavior tab now has a new column, Delta Percent Change (Δ % Change).
- You can now hover over new tooltips in the following columns:
  - Baseline
  - % Change
  - Δ % Change
  - Zscore
  - Score

Outliers

- Outlier checks are now optimized to skip in certain circumstances. Outlier checks are only skipped when the history load of a specified date column is empty.
- You can now update and modify record flags from the command line with `-rc`, `-rcKeys`, `-rcDateCol`, and `-rcTbin`.

API

- The v2/gethoot API now properly returns rule dimension information for data sets.
- The v3/jobs/run API now has improvements to the 400 Bad Request error messages in specific circumstances.

Reports

- The PDF option is now removed from the Data Set Findingspage. To print dynamic column tables, use CSV or Excel options instead. (ticket #89739)

DQ Connector

- The version of Collibra Integration Library is now updated to 2.4.12.

Fixes

Connections

- The new GCS jars are required to use GCS spark-history-server. (ticket #90623)

DQ Job

- Fixed an issue that caused jobs using .TXT files to incorrectly render custom column names. (ticket #81808)
    - Files with .TXT extensions are now treated as delimited files. Files with .TXT extensions that are not delimited files should use their respective file type from the file type dropdown.
- Fixed an issue with deployments on K8s where jobs failed when the volume name exceeded 63 characters. (ticket #85372)

Agent

- Fixed an issue that caused the v2/updateagent API to fail when numCores was empty. (tickets #89737, 92404, 92680)
    - The numCores field is no longer a required field.

Validate Source

- Fixed an issue that caused validate source jobs to fail when the pkey was mapped to different column names. (ticket #88778)

Rules

- When using Freeform SQL rules with wild-card operators, rules again correctly pass validation. (ticket #89644)
- Fixed an issue with regex rules that use the characters `)`, `,` , and `;` in the rlike, which caused DQ to append spaces to those characters and prevented the regex from operating correctly. (tickets #89417, 92958)

- Fixed an issue that caused rules with column values containing parentheses ( ) to break due to the addition of padding before and after closing parentheses. (ticket #85176)
- Fixed an issue that caused rules with special characters such as @ to display incorrectly on the Rules page, Conditions tab, and when exported to Excel.
- Fixed an issue that prevented data sets with attached rules and roles from being renamed. (tickets #85731, 92059, 94315)

Profile

- Fixed an issue where certain results in TopN Values and Data Preview displayed in scientific notation. Scientific notation is now removed from the display. (tickets #82163, 89738)

Explorer

- Fixed an issue that allowed CLOB data types to be visible in the Drag Columns to Target map in the Source tab. (ticket #86902)

API

- The REST API endpoint v2/updateRoleDatasets again correctly saves roles to data sets.

Known Limitations

Rules

- The Findings page displays results from computational stat rules on mean as a single-quote string. For example, '573523.87' > 6763
- Column-level sorting for the Rule Summary feature is not currently available.

Admin

- When adding a Sensitive Label or a Data Category, the Edit and Update functions do not display the selected record. To properly display the record, you must first refresh the page before editing or updating.

Session Activity

- While the application UI is being redesigned, it is possible that when the application times out on the legacy side of the application, you might not be able to see it on the new React MUI side. This can happen when you have the DQ application open on multiple tabs.
    - We are not currently tracking session timeout from the legacy UI to React.

Beta features

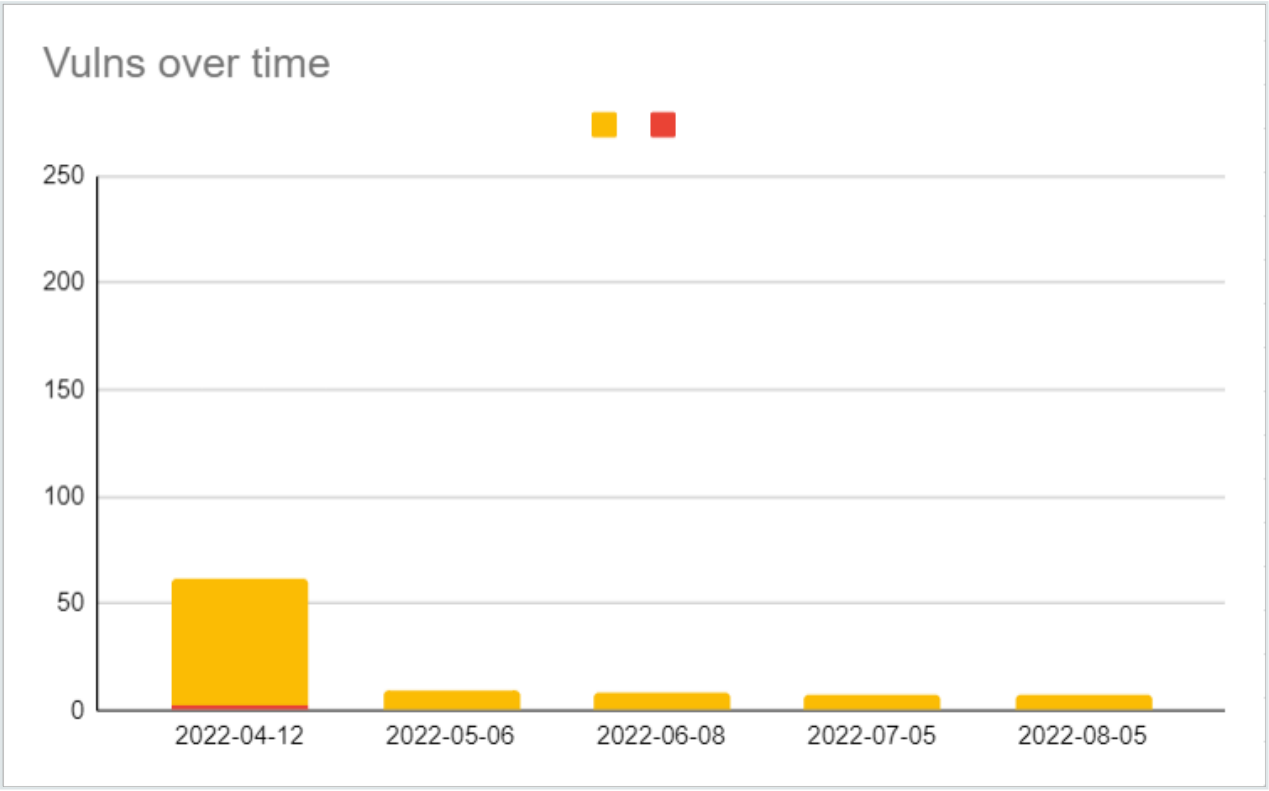
DQ Job

- Collibra is proud to launch a brand new feature, Snowflake Pushdown. Snowflake Pushdown allows for even faster processing and removes the need to set up a separate Spark compute platform to run Collibra Data Quality. Snowflake Pushdown is a private beta feature only available by request. Since this is a beta feature, some limitations are expected as we continue to improve its functionality. Contact your CSM to learn more about this feature.

DQ Security Metrics

> Warning There is a critical CVE `CVE-2016-1000027` that shows up in the image scan due to Spring version. This is a false positive and should be added to the exception list of the customer scan tools. We don't use `HttpInvokerServiceExporter` anywhere in the application and are not impacted by it.
>
> There is no fix version available for it from Spring. More details are available at

> Sonatype vulnerability CVE-2016-1000027 in Spring-web project · Issue #24434 · spring-projects/spring-framework



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resol	# Highs	# Mediums	Artifact(s)
DQ	2022-04-12	2022.04	2	2	0	60	136	JFrog Artifact
DQ	2022-05-06	2022.05	0	0	2	9	19	JFrog Artifact
DQ	2022-06-08	2022.06	0	0	0	8	13	JFrog Artifact
DQ	2022-07-05	2022.07	0	0	0	7	13	JFrog Artifact
DQ	2022-08-05	2022.08	0	0	0	7	14	JFrog Artifact

2022.07

> Note Standalone packages for the 2022.07 release have a version naming convention of `-RC`. This will revert back to the standard naming convention with the 2022.08 release, and has no impact on the safety or stability of standalone packages.

Fixes / Enhancements

- DQ Job
  - Fixed an issue that prevented data from appearing in the Source tab when Source Observation RunID was clicked from the Assignments page.
  - Fixed an issue that caused Annotations with special characters to be truncated in the Labels tab.
  - Fixed an issue that caused the Column (name) column of the Rules tab to display incorrectly when Run Discovery was used.
  - Fixed an issue where the Retrain button on the Record tab was disabled.
  - You can again invalidate observations with single quotes ' from the Shapes tab.
  - The Hints tab now displays any available data.
  - You can no longer change agents from the Scheduler modal.
- Rules
  - SQLF is now supported for Generic rules.
  - When running a custom rule through Rule Discovery, the column names Repo and Column again display correctly.
- Alerts
  - You can now send emails using unauthenticated SMTP servers.
- Security
  - Vulnerabilities identified by Jfrog
    - Vulns 0, criticals 0, high severity 7
    - For a visual readout, see the DQ Security Metrics section below.
  - Fixed an issue that allowed jobs to be run from the command line regardless of connection permissions.
    - When Connection Security is enabled, lock the SQL Editor to prevent unauthorized access to other connections. (#87916)
  - Fixed an issue that allowed View Only users to access some profile results and export the data to a CSV file.
    - Added an authorization check for data set access to the profile export feature, which allows only users with data set access to export the profile. (#87720)
  - Backslashes \ are no longer supported characters for AD usernames without disabling XSS for the /v2/updateadsecurityconfiguration API. (#88499)

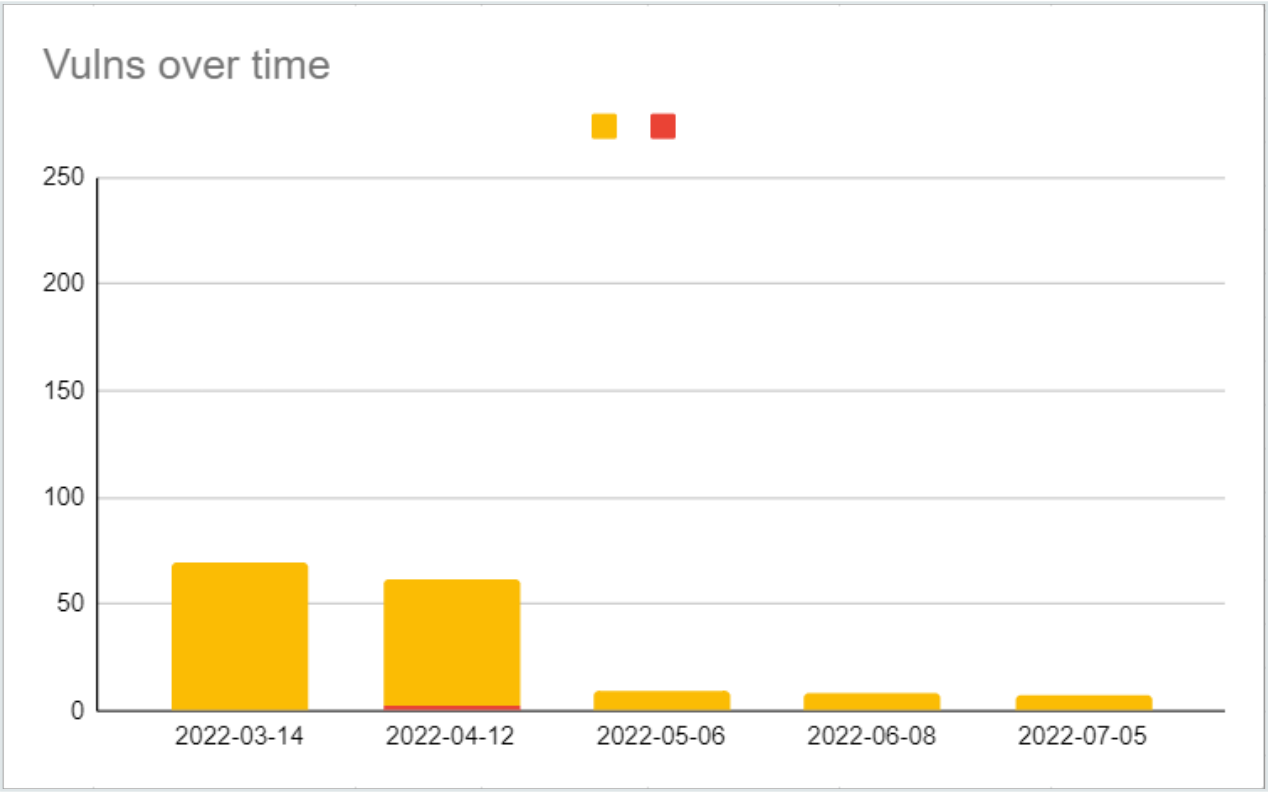
    - Fixed an issue that prevented navigation back to the log in page when tenant access was denied. (#89024)
- Profile
    - From the Labels tab, backslashes are now stripped from annotations when they are used for separation within strings.
- Admin
    - From Audit Trail, when administrators modify roles mapped to data sets or data sets mapped to roles, changes are now documented automatically, and display original and updated values.
    - The Agent Group (H/A) and its associated endpoints are now deprecated.
    - From Usage, you can now access a table and tiles reflective of your monthly usage metrics.
    - Salesforce account ID can now be configured for use with Pendo logs.
    - *Tech Preview* [TP] ServiceNow integration
        - You can now assign incidents (validate action) to ServiceNow groups and users with the following fields included in the same request: caller_id, description, short_description, cmdb_ci.
- Explorer
    - Fixed an issue with date range on Oracle connections, which caused end date to change to start date when Transform was selected.
    - The Job Estimate modal again displays the correct number of rows for Sybase connections.
    - Fixed an issue with Source to Target where double quotes " were removed from the source file in database to file targets.
- Scorecards
    - Enhanced the layout of the Assignment Queues page.
- API
    - v2/getallscheduledjobs is now available as an enhancement of the original, v2getscheduledjobs.
        - A UI integration is planned for a future release.
- Schedule
    - Added an Active column to the scheduler export.
        - The RunJob column was removed. (#88799)
- Reporting
    - Fixed an issue that created misalignment of column headers in PDF exports. (#89739)

Known Limitations

- Rules
  - To use the new SQLF feature for Generic rules, you must manually update the Generic rule type from SQLG to SQLF.
    - A UI feature for this is planned for a future release.
  - Stat rules such as $rowCount do not work for secondary data sets or previous runId of the same data set via @t1 syntax.
    - To work around this limitation, run a subquery to select count(*) from the secondary data set or the previous runId.
- Explorer
  - Drill-ins and jobs on Sybase connections run successfully, but connections to Sybase with encrypted passwords are currently unsupported.
- Files
  - When using CSV files, you cannot use a comma , in the name.
- Admin
  - *Tech Preview* [TP] ServiceNow integration
    - Special characters `!@#$%^&*()` in the description are not supported and will not persist to the ServiceNow assignment queue at this time.
    - Empty or invalid ServiceNow group name does not return an error in CDQ.
      - As a result, the ServiceNow assignment is generated with the default admin account as the owner if left empty or invalid.
      - You must have a valid ServiceNow group name or its related sys_id.
    - The new REACT UI is not yet supported for the ServiceNow Group integration.

DQ Security Metrics

> Warning There is a critical CVE `CVE-2016-1000027` that shows up in the image scan due to Spring version. This is a false positive and should be added to the exception list of the customer scan tools. We don't use `HttpInvokerServiceExporter` anywhere in the application and are not impacted by it.
>
> There is no fix version available for it from Spring. More details are available at Sonatype vulnerability CVE-2016-1000027 in Spring-web project · Issue #24434 · spring-projects/spring-framework



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolv	# Highs	# Mediums	Artifact(s)
DQ	2022-03-14	2022.03	0	0	6	70	213	JFrog Artifact
DQ	2022-04-12	2022.04	2	2	0	60	136	JFrog Artifact
DQ	2022-05-06	2022.05	0	0	2	9	19	JFrog Artifact
DQ	2022-06-08	2022.06	0	0	0	8	13	JFrog Artifact
DQ	2022-07-05	2022.07	0	0	0	7	13	JFrog Artifact

2022.06

Fixes / Enhancements

- DQ Job
  - Fixed an issue with the Learning Phase in the Behavior feature. (ticket #82907)
    - Once CDQ has the minimum number of completed successful scans, the learning status now changes to PASSING or BREAKING based on the results.
- Outliers
  - Fixed an issue where file lookback did not identify expected outliers. (#87967)
- Alerts
  - When configuring email alerts, SMTP Username and SMTP password fields are still required fields. (#86033)
    - Validation relaxation is planned for the 2022.07 release.
- Rules
  - Fixed an issue which caused rule breaks to report the opposite of what was defined when a Generic Rule utilizing regex/rlike was created. (#86977)
  - Fixed an issue where Data Classes with Date column types selected did not detect timestamps. (#83000)
  - Fixed an issue where Data Classes using the operators <, > or = caused the inverse rule created from this process to throw exceptions. (#83000)
  - When switching a data class from a regex to expression and then editing again, the regex checkbox is now correctly checked.
- Agent
  - The Explorer page and Scheduler modal now display the same agents. (#86175)
- Security
  - Vulnerabilities identified by Jfrog
    - Vulns 0, criticals 0, high severity 8
    - For a visual readout, see the DQ Security Metrics section below.
  - General advisory:
    - There is a critical CVE `CVE-2016-1000027` that shows up in the image scan due to Spring version. This is a false positive and should be added

to the exception list of the customer scan tools. We don't use `HttpInvokerServiceExporter` anywhere in the application and are not impacted by it.
    - There is no fix version available for it from Spring. More details are available at Sonatype vulnerability CVE-2016-1000027 in Spring-web project · Issue #24434 · spring-projects/spring-framework
- Major vulnerabilities related to Spring, ESAPI, and Swagger have been addressed.
- No cross DB reference is allowed in explorer while accessing SQL database connections.
- Sensitive UI fields such as username no longer allow autocomplete.
- If configured, the ENV variable `XSS_CANONICALIZE_INPUT_ENABLED` should be removed from configmap or owl-env.sh.
- When dataset security is turned on, you can now add role based authorization for editing existing datasets. (#87720)
- You can now override the following mail settings from the App Config page within the Configuration section of the Admin Console:
    - "mail.transport.protocol" -- default = smtp
    - "mail.smtp.auth" -- default = true: If true, attempt to authenticate the user using the AUTH command
    - "mail.smtp.auth.login.disable" -- default = false: If true, prevents use of the AUTH LOGIN command
    - "mail.smtp.starttls.enable" -- default = true: If true, enables the use of the STARTTLS command (if supported by the server) to switch the connection to a TLS-protected connection before issuing any login commands.
    - "mail.smtp.ssl.enable" -- default = false: If set to true, use SSL to connect and use the SSL port by default. Defaults to false for the "smtp" protocol and true for the "smtps" protocol.
    - "mail.smtp.ehlo" -- default = true
    - "mail.debug" -- default = true
    - "mail.smtp.ssl.trust" -- default = : If set, and a socket factory hasn't been specified, enables use of a MailSSLSocketFactory. If set to "*", all hosts are trusted. If set to a whitespace separated list of hosts, those hosts are trusted. Otherwise, trust depends on the certificate the server presents. (#76775, 88089)

- Profile
  - Mean value is now rounded appropriately within the Profile page.
    - For example: The value 2.4334334343345 is now rounded to 2.434.
- Connections
  - From the Athena driver, you can now use `MetadataRetrievalMethod=Query` for database queries from the Connection URL. (#86139)
  - Fixed an issue where error messages on failed connections did not display informational text. (#85527)
  - Fixed an issue where NFS file connections under Remote File connections caused jobs to fail. (#88156)
    - Added File protocol for Spark load for NFS file system.
    - Added nfs:// prefix wile adding a NFS connection.
      - This will prepend the URI with the file:// protocol when an NFS file connection is loaded via Spark.
- Catalog
  - The Graph option is no longer available in Quick links.
- Admin
  - The Pendo integration is now active by default.
    - No sensitive information is collected; only high-level usage stats are collected.
    - All new customers starting with 2022.06 onward will receive a new license.
    - If you install a standalone environment, modify the <install-dir>/-config/owl-env.sh file by adding your license name `export DQ_INTEGRATION_PENDO_ACCOUNTID=<your-license-name>`
    - This new integration will not block or impair the functionality of the app in any way.
    - For more information on Collibra's subprocessors, please review Collibra's Subprocessors page.
  - The Agent Group (H/A) and its associated endpoints are now deprecated. (#83086)
  - Fixed an issue where the "Add Data Category" button was missing without required permissions. (#86625)
  - When a session expires on an Admin page, you are now redirected to the login page.

    - The Admin Limits page now displays informational text indicating that only limits of Tenant - Admin type are displayed on the page.
    - Fixed an issue when editing an existing data category which caused the 'Add new' modal to open instead of the 'Edit' modal. (#89617)
    - From Configuration Settings, DB Limits is now called Data Retention Policy.
- Explorer
    - You can now view calculated views for SAP HANA when creating a DQ Job on the Explorer page. (#83147, 84328)
    - Fixed an issue which caused the Date range condition to incorrectly display results when using an Oracle connection. (#85802)
    - Fixed an issue which threw an error message when Transform was checked with Date Range condition when using a Postgres connection. (#85802)
    - Fixed an issue where an equals sign = used in a `-transform` expression from Run CMD caused jobs to fail. (#71547)
    - Fixed an issue where schema and table names containing underscores _ were not accepted.
    - Fixed an issue that allowed jobs to run with a row limit of less than 1.
    - Fixed an issue where incorrect files loaded for preview from BLOB containers with Livy enabled.
    - CLOB data types are unsupported. (#86902)
    - Improved performance and logic when drilling into a database and schema from the Explorer page.
- API
    - You can now access API quick links page from the Admin Console React page.
    - When using Swagger, UI text now indicates when a field is case sensitive.
- Reporting
    - *Tech Preview* [TP] Rule Summary page enhancements
        - You can now filter rule breaks by most frequent violations, most severe violations, and least violations.
        - You can now view interactive pie charts with rules and dimension summaries.
- UI
    - The styling of the expandable legacy navigation pane and the react menu are now updated.

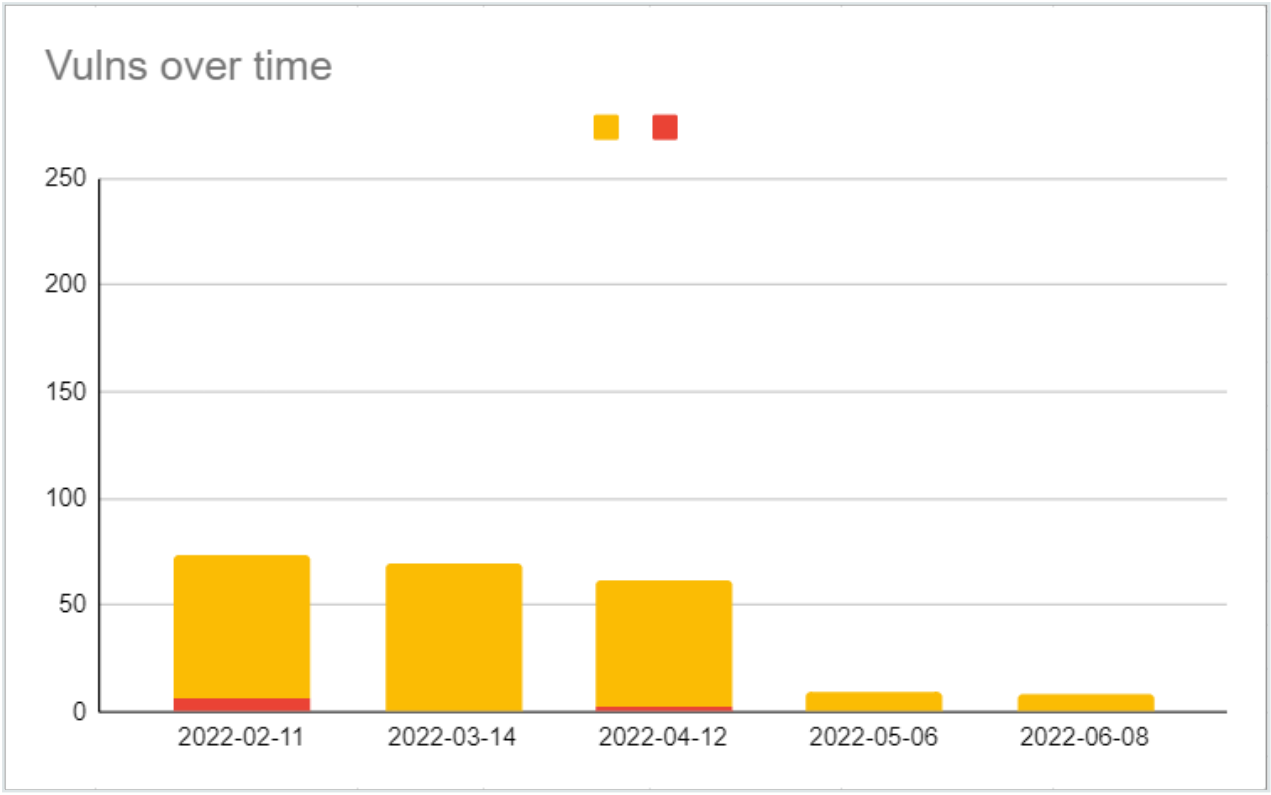
- Legal
  - Added a disclaimer to the DQ login page with a link to the Collibra Evaluation Agreement.

Known Limitations

- Validate Source
  - When comparing JDBC (target) to remote files such as S3 (source), there is a known Spark bug for "Recursive view detected".
    - This validate source combination is not possible in 2022.06 using Spark 3.2.
  - When using Bigquery as the source, the -libsrc needs to be manually modified to include the core (Spark Bigquery connector) directory.
    - For example, /home/centos/owl/drivers/bigquery**/core**
- Profile
  - Spark does not currently support varchar data types. All varchar data types are converted to String. Other unsupported data types may also be converted incorrectly.
- Security
  - Permissions on the Export task have not yet been addressed when dataset security is turned on and you add a role based authorization for editing existing datasets. (#87720)

DQ Security Metrics

> Warning There is a critical CVE `CVE-2016-1000027` that shows up in the image scan due to Spring version. This is a false positive and should be added to the exception list of the customer scan tools. We don't use `HttpInvokerServiceExporter` anywhere in the application and are not impacted by it.
>
> There is no fix version available for it from Spring. More details are available at Sonatype vulnerability CVE-2016-1000027 in Spring-web project · Issue #24434 · spring-projects/spring-framework



Product SKU	Date	Release	# Criticals	New Criticals	Criticals Resolved	# Highs	# Mediums	Artifact(s)
DQ	2022-02-11	2022.02	***6***	5	5	68	80	JFrog Artifact
DQ	2022-03-14	2022.03	0	0	6	70	213	JFrog Artifact
DQ	2022-04-12	2022.04	2	2	0	60	136	JFrog Artifact
DQ	2022-05-06	2022.05	0	0	2	9	19	JFrog Artifact
DQ	2022-06-08	2022.06	0	0	0	8	13	JFrog Artifact

2022.05

Fixes / Enhancements

- DQ Job
  - You can no longer update the dataset name (`-ds`) from the command line.
    - A helpful error message now appears if changes are made to `-ds`.
  - Stop Job action is no longer enabled for K8s.
  - Fixed an issue for Dremio jobs where jobs hang when editing or cloning an existing dataset.

- Outliers
  - Added "username" to outlier boundary table to track who creates the boundary.
    - The Outlier boundary again saves correctly after the addition of a username.
  - Fixed an issue that caused jobs to fail when Day from By dropdown was selected.
- Rules
  - Rule Preview drill-in capabilities are now improved:
    - You can now configure Preview Limits based on the individual rule.
      - Freeform and Simple rules are currently supported for the Preview Limit feature.
    - You can now set any positive number as the Rules Preview Limit.
      - When you update a Preview Limit value, you must re-run to apply the updated limit value.
    - On the DQ Job page, the details of an individual rule now displays a paginated sub-table of all the break records.
    - When a rule is labeled as BREAKING for rule types other than Freeform and SQL, UI text now displays, "Data preview records are only available for Freeform and Simple rules."
  - You can now hover over stat rules to see their conditions.
  - Data Concepts is renamed Data Categories.
  - Semantics is renamed Data Classes.
  - When a Data Class is assigned to a dataset via Profile controls, a rule is now created.
- Security
  - Vulnerabilities identified by Jfrog
    - Vulns 0, criticals 0, high severity 9
    - For a visual readout, see the DQ Security Metrics section below.
  - The OS vulnerabilities from the images of Collibra DQ 2022.04 have been resolved by using the base image of RHEL8 to build the images for Collibra DQ 2022.05. The following OS utilities will not be available in the 2022.05 release images:
    - Unified, OpenSSL crypto/stack
    - Full YUM stack
    - OS tools, including tar, gzip, and vi

    - AD users can again use auth/signin REST API.
    - The Highcharts CVSS2: 9.3/CVSS3: 9.8 vulnerability is resolved.
    - The LOGJAM (CVE-2015-400) SSL/TLS vulnerability is resolved.
    - The SpringShell (CVE-2022-22965) vulnerability is resolved.
    - TLS < 1.2 is no longer supported.
    - When Azure AD SSO sends a groups.link assertion, the application now tries to resolve the groups via the link.
        - You can now activate this setting by using the property, SAML_GROUP_LINK_PROP.
- Profile
    - You can now edit or delete semantics by clicking anywhere in the semantics cell of the Profile column table.
    - You can now save annotations with special characters.
        - Special characters that are not currently supported include percent sign `%`, backslash `\`, and caret `^`.
    - Fixed an issue where columns of broken rules were not highlighted.
- Connections
    - You can now view a list of all packaged and optionally packaged drivers on our new Builds page.
    - The Databricks JDBC driver is now available.
    - You can now add Databricks datasets using the Databricks Simba driver.
- Catalog
    - Fixed an issue where the deletion of a dataset caused orphaned links to datasets in other areas of Collibra DQ.
- Admin
    - *Tech Preview* [TP] You can now use the ServiceNow integration through a proxy server from the Assignment Queues screen.
    - You can now access the new Usage page to view monthly historical usage statistics.
    - AD users with Admin privileges can now add Business Units.
    - AD users with Admin privileges can now manage local users.
    - The Agent Groups (H/A) feature is marked for deprecation and will be removed from the app in the 2022.06 release.

- Explorer
  - You can again edit schema and table name from the Catalog page.
  - You can now navigate to a specific behavior tab directly from the Assignments page.
  - Fixed an issue when viewing Schemas in View Data wizard.
- Scorecard
  - Single-space `` , underscore _, and period . are now supported characters when saving Scorecard name.
- API
  - Improved API calls for the UserManagement Save function.
- Reporting
  - *Tech Preview* [TP] Rule Summary page enhancements
    - You can now filter rule breaks by a specified date range and view charts for Most Used Rule Types, Dataset with Most Rule, and Top Rules Run.

Known Limitations

> Warning Delta Files
>
> A bug was introduced as a result of removing CVEs in 2022.05. If you use Delta files -delta it is not advised to upgrade until an update is available.

- Explorer
  - Except for underscore _, special characters are not currently supported in schema or table names.
- Admin
  - *Tech Preview* [TP] ServiceNow integration
    - Only the local Docker container proxy has been tested and verified.
    - The Test Connection button's validating credentials capabilities is currently limited if the ServiceNow URL is valid.
    - The Validate All Rules function currently results in a failure.
    - You cannot edit an active ServiceNow assignment.
      - Invalidate/Validate or Resolve actions result in a failure.
    - You can assign a ServiceNow ticket with an embedded URL when escaped with double quotes.
      - No assignment is sent without this process.

- Multi-Tenant
  - Tenant names should be lower case. Use lower case characters, when creating a tenant from the multi tenant admin page. The current limitation is around the schema that is generated
- Reporting
  - *Tech Preview* Rule Summary page enhancements
    - Sorting any column returns an error.
    - User must use date picker as manual date entry is not honored.
    - The start and end date are out of order when navigating to the page.
    - The last page on the paginated list does not change when date criteria is updated.

2022.04

Install

> Note For standalone installations, within the setup.sh script find/replace the variable for **spark_package**.
> Change spark-3.0.1-bin-hadoop3.2.tgz to spark-3.1.2-bin-hadoop3.2.tgz

```
spark_package=${SPARK_PACKAGE:-"spark-3.0.1-bin-hadoop3.2.tgz"}

# replace with

spark_package=${SPARK_PACKAGE:-"spark-3.1.2-bin-hadoop3.2.tgz"}
```

Fixes / Enhancements

- DQ Job
  - Entering negative values for the downscore is no longer supported and will now produce an error message.
  - You can now invalidate schema with special characters.
  - Spark table names of historical dataset loaded and other spark tables are now available on Jobs Log table.

    - Long type values larger than `Integer.Max` no longer breaks the Profile.
    - View Findings now displays user's full name, if applicable, in Validate Modal. Assignment queue page also displays the full name of user, if applicable.
- Alerts
    - You can once again use the Cancel action button on the Alerts page.
    - You can now set up alerts to reach multiple email recipients.
    - If email_server table is not yet configured, a helpful message will now display in the Description column in the job log directing you to register an email Server under Admin - Alerts. The job will still run successfully.
- Rules
    - You can now modify Rules definitions from the primary DQ Job dashboard without loading the Rules page.
    - Mean value check once again triggers correctly for Integer and Long columns.
        - This fix triggers the mean value check for Integer and Long columns and shows an infinity percentage change in behavior for a period, depending on -bhlb. After this period, it should disappear.
    - For Native SQL rules, jobs now behave the same whether or not a semicolon ";" is included in the SQL query.
    - You can now use a hyphen "-" in a dataset name.
        - Acceptable special characters now include a hyphen "-", period ".", and underscore "_".
    - Added a tooltip that displays which condition is being checked in a DQ Job when using a Stat rule when you hover your cursor over a condition in the Condition column.
    - Improved the exception message for when there are no values for a specific column while using a Stat rule.
    - The WebUI passing boundaries range has been updated to ().
    - For Freeform rules, IS Null and IS NOT NULL no longer return invalid results in the Validation tab.
    - Added a pop-up success message for when the correct syntax rule passes for Freeform rules with secondary datasets after the Validate button is clicked.
- Security
    - Vulnerabilities identified by Jfrog
        - Vulns 2, criticals 2, high vulnerabilities
        - For a visual readout, see the DQ Security Metrics section below.

- Authorization restriction is now enforced for the following endpoints:
  - /v2/deletefiledir
  - /v2/getRunIdsByDataset
  - /v2/putDatasetWeight
  - /v2/checkListofFilesPath
  - /v2/getlistagents
  - /v2/checkDriver
  - /v2/getconnectionssensitive
  - /v2/getemailgroups
  - /v2/getemailserver
  - /v2/addemailgroup
  - /v2/validateEmailAddress
  - /v2/getlistoffiles
  - /v2/getlistoffilespath
  - /v2/getlistoffiles
  - /v2/getDriverDir
  - /v2/getlistrolesbydataset
  - /v2/getlistrolesbydistnctdatasets
  - /v2/getlistrolesbyfunctiontypename
  - /v2/getlistusersbyauthority
  - /v2/getlocalDBRoles
  - /v2/getsecuritysettingsbytype
  - /v2/getowlcheckinventory
  - /v2/getconnectionspwdmgrsensitive
  - /v2/getsecuritysettingsbycoltype
  - /v2/getdbuserlist
  - /v2/getdbuserdetailsbyuser
  - /v2/getexternaladgroupstointernalroles
  - /v2/getlistdatasets
  - /v2/getlistdatasetsbyrole
  - /v2/getaudittrailitems
  - /v2/get-all-audit
  - /v2/get-datasets-audit-trail-items
  - /v2/get-all-dataset-audit
  - /v2/getactivityaudit
  - /v2/getallactivityaudit
  - /v2/getlocaldbrolesbyuser

        - /v2/getdatasetaclsecurity
        - /v2/getexternaladgrouplist
        - /v2/getexternaladuserlist
        - /v2//external-service-configuration
    - Local user accounts now have an account lockout feature implemented with the following restrictions:
        - A user's account will be locked if a password is entered incorrectly more than 10 times (configurable via app config).
        - The locked account can only be unlocked by Admin user in user management screen.
        - If an Admin is locked, another Admin can unlock their account.
        - If all the Admins are locked, enable the account via DB (ubdate users table "accountNonLocked" colun to "1").
        - User cannot use forgot password to reset password while the account is locked.
    - CORS restriction is now enforced for SAML and multi-tenancy.
        - This breaks SAML unless the IDP is configured as a trusted origin in DQ, so the following property must be added to environment variables in order for DQ and SAML to work: CORS_ALLOWED_ORIGINS=\({IDP-BASE-URL},\)
            - Replace $ with the value of the actual IDP URL (For example: https://ping.auth.com)
            - Replace $ with the value of the actual DQ Base URL (For example: https://dq-env.com)
    - SAML login no longer automatically triggers on the login page during an existing session when accessing DQ base URL. For SAML login, you should instead use /saml/login.
        - API requests (v2/v3) return proper JSON response in case of failures.
        - auth/signin API is updated to provide JWT token for MT & local users.
- Profile
    - Mean value once again displays in the Volume column.
    - When connecting to MSSQL server on Windows from a Linux DQ environment, the connection no longer fails.
        - We recommend (not required) a TLS connection for MSSQL connections from a DQ Linux environment with a properly signed certificate setup on MSSQL server to connect only via TLS.

    - You can now edit annotations in the Labels tab.
- S3
    - Added an enhancement for -addlib flag.
- Connections
    - Added new Jconn4 driver for encrypted connections.
    - Tech Preview - You can now save a local (NFS) file directory as a connection type.
    - See our newest Supported Connections page for a definitive guide to driver support.
    - BigQuery is now certified for production, but removed from packaged install for K8s docker.
- Explorer
    - When toggling between fullfile and Union LookBack options, `-fullfile` and `-fllb` flags can no longer be generated together in the DQ Job command line.
    - Data Preview for Temp files loading in Explorer now correctly shows the order of columns of the original Temp file.
    - You can now drill in and search files within the connection.
    - You can now browse multiple local (NFS) file connections.
- Scorecard
    - You can now create scorecards with special characters "^[A-Za-z0-9]+$" in their names.
- Dupes
    - Added linkID column for exact match in both UI and REST API. linkID can now be either included or excluded from Dupes for exact match.
    - linkID is now shown at the aggregate level for Exact Match.
        - We recommend using this feature from a primary key perspective for its first iteration.
        - The aggregate function used is min().
            - For example: if you have 6 occurrences, you will get 1 example linkID, the min.
- API
    - Updated the /v2/getlistdataschemapreviewdbtablebycols API call method from GET to POST to support the long query (-q) or very large columns table.
    - Added a new SAML load balancer so the syestem picks the appropriate schema and SAML server URL for Swagger.

Known Limitations

- Profile
  - Special characters are not currently supported in annotations in the Label tab.
- Scorecard
  - Space " ", underscore "_", and period "." are not yet supported for scorecard edit.

2022.03

Fixes / Enhancements

- DQ Job
  - The -validatevaluesshowmissingkeys options now allows the extrapolation of missing keys between target and source.
  - Newly created jobs will no longer be marked incorrectly with enclosing double quotes.
  - File names with spaces are now handled with double quotes within the application.
- Alerts
  - Email notifications now have Collibra branding and terminology.
  - Fixed Cancel Action for Delete functionality on Alert page.
- Outliers
  - Fixed the issue where Numerical Outlier drill in graph wasn't displaying when perChange is NaN.
- Rules
  - Added additional HealthCare Data Classes to Rule Library.
  - Fixed input validation rule of POST - /v3/rules/ endpoints. The following validation rules have been applied to RuleDTO.ruleName field:
    - Maximum size is 100.
    - Must comply with the following regular expression: ^[a-zA-Z0-9_]+$
  - The rules on the Hoot page now show the correct exception data when expanded if there are two or more rules with exceptions attached to the dataset.

- Security
  - Vulnerabilities identified by Jfrog
    - Vulns 0, critical, 6 high vulnerabilities
  - Password length has increased to a maximum of 72 characters.
  - Forgot password screen will now always show success message in UI regardless of success or failure.
  - Fixed an issue of a throwing error message when adding/editing user roles.
  - Added error checks if the password manager script throws any errors.
  - Added the helper text "Enforce user roles to run the job" to DQ Job Security row.
  - User password field removed while updating user in user management screen.
    - Admin can only set password for another user wile creating new user, but not while updating/modifying them.
    - To change a password, users can now use either the profile page or the self-service (Forgot password) feature.
  - XSS security
    - Fixed the vulnerability on scorecard, jobs, rules and catalog pages.
    - Fixed the vulnerability via remote connection.
  - Mitigated the endpoint "/v2/getrawpreview" vulnerable to Local File inclusion attack.
  - DQ HTTP session cookie is now secured by default when HTTPS is enabled.

Rule Discovery Terminology Alignment

Data Concepts => Data Categories

Semantics => Data Classes

- Profile
  - Precision and Scale metrics are correct when using multi executors.
- Admin
  - Edge download page within Admin Console (for Cloud customers).
- Validate Source
  - *Tech Preview* [TP] Update Source Scope.
    - Added "Update Source Scope" in the Query section of the Source tab.
- Connection
  - Added handling for errors during log cleanup process.

- API
  - Improved API calls for the Save function.

Known Limitations

- Validate Source
  - *Tech Preview* [TP] Update Source Scope.
    - Only works for JDBC connections. Feature is hidden for remote, temp, local files.
    - Valsrc query won't be updated automatically when modifying column mappings. Use 'Preview' button to reset the feature if column mappings need to be changed.

2022.02

> Note For new Standalone Collibra DQ installations, please double check 'Number of Core(s)' field when setting up 'Edit Agent'.

Added UUIDs for Jobs may take additional time on initial startup after upgrade.

Enhancements

- DQ Job
  - Added UUIDs for jobs for better tracking between web and core
  - Improved DQ Job page load performance by optimizing calls
  - Fixed issue DQ jobs would fail when -rd is in "yyyy-mm-dd HH" format
- Outliers
  - *Tech Preview* [TP] Outlier Calibration
    - Feature flag can be set within owl-env.sh or configMap with `export outlier_calibration_enabled=true` (Default is off)
    - Ability to suppress Outlier observations for a user-determined length of time that would have otherwise surfaced as anomalies
    - Once feature is enabled, accessible within Outliers tab on DQ Job page

- Alerts
  - Ability to navigate to dataset specific Alerts from DQ Job page
  - Ability to test SMTP alert configurations when adding an email relay
  - Fixed issue where 'Reply Email' field did not properly accept user input value
    - Please note there are no (Collibra imposed) domain restrictions on Reply Email field
- Security
  - Stricter password policy is enforced on all user/tenant management screens/APIs.
    - The restriction is as follows: Minimum length of 8 characters
    - Maximum length of 20 characters.
    - At least one upper-case letter.
    - At least one numeric character.
    - At least one special character (supported are !,%,&,@,#,$,^,*,?,_,~)
    - User ID and password cannot be the same.
    - Password cannot contain user ID.
  - Change Password functionality on user profile requires a current password of the user.
  - Mitigated 64 critical, 15 high, and 12 medium vulnerabilities identified by JFrog
  - Upgrade Log4J to 2.17.1
    - Please follow *Note to Standalone Collibra DQ Customer Upgrades*: We have upgraded to Log4J 2.17, please refer to Standalone Upgrade for additional steps
  - Added connection security checks to users to prevent running jobs and query the tables that are not authorized per connection. This is applicable when `DB Connection Security` is enabled in the Admin Console under `General`.
  - Implemented stricter session management
  - Implemented CORS restriction to mitigate potential CSRF vulnerability
    - Enforced strict CORS policy by not allowing any domain. In order to allow other domains and tweak this behavior, we have exposed the following properties as environment variables in owl-env:
    - CORS_ALLOWED_ORIGINS=http://facebook.com,http://google.com
    - CORS_ALLOWED_METHODS-S=GET,POST,OPTIONS,DELETE,PUT,PATCH
    - CORS_ALLOWED_HEADERS=X-Requested-With,Origin,Content-Type,Accept,Authorization

        - CORS_EXPOSE_HEADERS=
        - CORS_ALLOW_CREDENTIALS=false
        - CORS_MAX_AGE=0
- *Tech Preview* [TP] DQ Connector
    - Fixed issue where tenant specified on DQ Connector configuration (issuer of the jwt token field within DGC Edge Management page) was not properly accepted; only rules that existed with 'public' schema were brought over; now the DQ Connector will accept the proper values
- Agent
    - Upon potential deletion of an agent, added server side validation to indicate number of scheduled jobs so that users can understand if jobs fail going forward
- Rules
    - Enhanced stability on Parallel Rule execution to ensure all rules load by reverting back to fixed thread counts
    - Display exceptions upon rule execution failure to improve rule management experience
    - Improvements to user experience in Rule Library tab (within Rules page) including filters and column alignment
    - Quick Rule dropdown within the Rules page will save with default severity of 1 point and a threshold of 1 percent
    - Enhanced validation for rules generated in Profile tab
    - Fixed issue where removing semantic tag may not have removed corresponding auto-generated rule
    - Rule name character limit of 100
    - Rule Builder page now returns error messages where the dataset contained 0 records
- Catalog
    - Renaming Dataset from Catalog page keeps associated rules
        - Clone only creates the dataset shell (with DQ job run configs, no additional rules, etc.) will be copied
    - Bulk actions support for Data Concepts
    - Fixed issue where child of business unit could be assigned as parent
    - Fixed issue where clearing individual filters were not functioning

- Validate Source
  - *Tech Preview* [TP] New collapsible section for Query in Source tab; enables users to use custom srcq, similar to query on section on Home tab so that users do not need to edit -srcq in cmd line editor on Run tab
  - Introducing new observation types via `-valscrshowmissingkey` flag
    - Key not in source
    - Key not in target
  - Source Name should be fetched as part of get-catalogandconnsrcnamebydataset API call for a given dataset
  - Fixed issue which prevented Hive from working as Target
- Export / Import
  - Fixed issue that import could not accommodate more than one table insert
  - Fixed bug where certain values were inadvertently inserted into RegEx rules upon Export
  - New endpoints added for Export and Import API
- Connection
  - Fixed Out Of Memory issue with Dremio
    - Explicitly added limit clause in the preview query within Update Scope
    - Dremio driver requires double quotes in Schema, Table, and Column names e.g. "SchemaName"."TableName"
  - Fixed Oracle TIMESTAMPLTZ conversion error
- Explorer
  - Fixed issue where 'Analyze Table' option did not populate for Hive
  - Fixed the static date values showing up in Managed Template and Run Check while running the job via v2/runtemplate API call from swagger UI
- Files
  - File names with spaces are now handled with double quotes t
  - Implemented Supported File Type Check at time of uploading the Temp Files via Explorer
    - Default supported file types are “csv,json,parquet,avro,delta".
    - In order to add/update the supported file types and ensure validation, a new environment variable needs to be added in owl-env.sh as below: `export ALLOWED_UPLOAD_FILE_TYPES-S="csv,json,parquet,avro,delta"`
    - Tip: For remote files with delimiter, please use the csv dropdown options for files with .txt extension

    - *Tech Preview* [TP] Users have ability to assign an agent when using temp file and local file Explorer paths without manually appending -master to agent or job (previous known limitation)
    - LIMIT values are now properly accepted on the Scope & Range query panel
- Dupes
    - Fixed issue where column selections were not retained from the original DQ Job with Dupes ON for future runs

Known Limitations

- Rules
    - Cannot currently create rule with API /v3/rules; will be fixed in future release
        - Please use /v2/createrule API
- Profile
    - Stat Rules
        - Tool tips will only generate when Max Precision and Max Scale are greater than 0
- DQ Job
    - /v2/runtemplate API still creates 'zombie' job
        - Please use /v3/jobs/run
- LinkID
    - LinkID column selection is case sensitive; breaks may not appear if case does not match
- Outliers
    - Outlier Calibrate
        - Outliers cannot retrain on-demand; to suppress existing Outliers, must rerun the DQ Job for those date(s)
        - In-app labels do not exist for Outliers which have been subject to past, current, or future calibration; references only exist within the `outlier_boundary` table in the metastore

[Informational Only] New Tables Introduced To Metastore In 2022.02

- `outlier_boundary`

[Informational Only] Changes To Metastore Made In 2022.02

```
ALTER TABLE validate_source_metadata ADD COLUMN IF NOT EXISTS
validate_values_show_missing_keys boolean DEFAULT false
ALTER TABLE opt_source ADD COLUMN IF NOT EXISTS validate_val-
ues_show_missing_keys boolean DEFAULT false

ALTER TABLE opt_source ADD COLUMN IF NOT EXISTS filter_cols
character varying[]

ALTER TABLE user_profile ADD COLUMN IF NOT EXISTS external_
user_id VARCHAR

ALTER TABLE owlcheck_q ADD COLUMN IF NOT EXISTS agent_job_uuid
UUID
ALTER TABLE job_log ADD COLUMN IF NOT EXISTS job_uuid UUID
ALTER TABLE platform_logs ADD COLUMN IF NOT EXISTS job_uuid
UUID
ALTER TABLE platform_logs DROP CONSTRAINT IF EXISTS platform_
logs_job_uuid_ux
ALTER TABLE platform_logs ADD CONSTRAINT platform_logs_job_
uuid_ux UNIQUE (job_uuid)
ALTER TABLE opt_owl ADD COLUMN IF NOT EXISTS job_uuid UUID
```

2022.01

Enhancements

- DQ Job
  - Fixed issue where backrun "-br" flag was inadvertently added on future runs (error contained in 2021.12) if the initial DQ Job setup Explorer selected backrun
  - Improved validation to not allow for slashes in dataset name
- Validate Source
  - Fixed potential DQ Job failure with Source turned on for some legacy installations when upgrading from older versions to 2021.11 and newer
- Explorer
  - `DB_VIEWS_ON` can be added with TRUE or FALSE values by adding new App Config (Add Custom within Admin -> Configuration)
  - -Addlib flag now working across JDBC connections
  - Update Scope now supports rdEnd

- Rules
  - When creating rules, run-time limit for each rule (in minutes) can be set on the Rule page UI and on the V3 API (by setting `runTimeLimit` property). The default is 30 minutes if not explicitly set. This 30 minute limit sets the overall timeout limit for all rules in a particular job. For example, if there are 10 rules with 9 rules with 30 min limit and 1 rule as 90 min limit, then the DQ Job will wait up to 90 min for all 10 rules to finish. This is because all rules must finish before the Rule stage in DQ Job to finish and move to the next stage. We do not support async stages where one long running rule is running while the job itself moves on to the next stage.
  - Added ability to specify score of 0 to a rule
  - Improvement to Stat Rules to fail without exception when result is not within range
- Profile
  - Fixed ability to remove a business unit from a dataset
  - Fixed issue where data concepts were not correctly displaying on a dataset's Profile page
  - Fixed sensitive labels not being assigned from Discovery
  - Treat certain doubles, floats, decimal types as Decimal format that preserves length and prevents Java from truncating to E11 format
  - Removed commas when displaying date columns
- Security
  - SAML Login fix for IDPs that use POST binding as default
- S3
  - Enhanced support where "." in column headers were causing large jobs to not complete
    - Underscores now replace periods and large jobs should no longer hang
- Connections
  - Updated default Snowflake template connection properties
    - Corrected 'db' parameter placeholder on connection string versus former 'databaseName'
  - Added Connectivity to BigQuery troubleshooting information

Known Limitations

- Local files using NO_AGENT require a valid $SPARK_HOME on the machine where the web server is running.
- Supported data types
  - CLOB datatypes are unsupported
- Explorer
  - -Addlib not yet supported for Remote Files e.g. S3

[Informational Only] Changes To Metastore Made In 2022.01

```
ALTER TABLE owl_rule ADD COLUMN IF NOT EXISTS run_time_limit
DOUBLE PRECISION NOT NULL DEFAULT 30.0;
ALTER TABLE owl_rule ADD COLUMN IF NOT EXISTS scoring_scheme
INT4 NOT NULL DEFAULT 0;

ALTER TABLE job_log ALTER COLUMN stage TYPE character varying;
-- stage set to varchar because RULE logs rule_nm into stage
ALTER TABLE job_log ALTER COLUMN log_desc TYPE character vary-
ing;
ALTER TABLE job_log ALTER COLUMN log_hint TYPE character
varying;
```

2021.12

Note to Standalone Collibra DQ Customer Upgrades: We have upgraded to Log4J 2.17, please refer to Standalone Upgrade for additional steps

Enhancements

- Rules
  - Semantic and data concept management: Run Discovery feature
    - Run Discovery feature can be accessed from Catalog by selecting 'Data Concept' option from Actions or clicking the 'Run Discovery' button on the Rules tab of the DQ Job page. This will run a DQ Scan to detect for the semantics assigned to the selected data concept

        - Algorithm now selects best match if column matches 2 or more data classes based on % match and name distance
    - *Tech Preview* [TP] Configurable rule break preview limit
        - Global default is 6 max rows per rule
        - Any change from 6 must be specified with previewLimit (API /v2/creat-erule) or in the Preview Limit field (UI)
        - Maximum of 50 from UI
    - Introducing additional Stat Rules including minPrecision, maxPrecision, minScale, maxScale
- Behavior
    - Min and max value checks are now triggered for all numeric columns when selected, even if column contains zeroes in lookback period
    - AR column view graph now shows theMean value for current day (runId). No re-run of DQ Job is necessary. The displayed Mean makes it clear that the % change is the % change from the mean, not runId - 1 day.
    - Findings in behaviors that were directly correlated to a row count shift as the root cause have been optimized, such that a major deviation in row count will no longer down-score related fields in the dataset to reduce noise
- Catalog
    - Catalog now features intelligent ranking based on Recency, Most Scanned, User
- Outliers
    - Outliers (advanced) allows for gaps in dates when establishing lookback period, which is established by history with row count > x (specified by user)
    - Fixed issue where outlier data preview graphics were not displayed
    - Fixed issue where outlier results did not honor the initial scope where clause, in particular for Remote Files (S3)
- Connections
    - BigQuery: Enhanced support for cataloging host name
- Pulse View
    - Pulse view can filter Connections and Users
    - Pulse view can serve as proxy verification on whether scheduled jobs were successfully completed
- Profile
    - Viewable precision and scale statistics for double, float, and decimal data types

- Shapes
  - Fixed issue where data shape preview not available when same shape is detected on the same row for different columns
- Files
  - *Tech Preview* [TP] Users have ability to assign an agent when using temp file and local file Explorer paths
    - Known limitation: -master must be freeform appended to the agent or to each job
  - Support for multicharacter delimiters
  - Improved delimiter support to distinguish string commas versus actual CSV commas to align data to respective columns
- Agent
  - Fixed issue where certain completed jobs could not be re-run on the DQ Jobs page. In other words, NO_AGENT was the only available option in the Agent dropdown. Now, users can select valid agents in the dropdown and this will persist for future scheduled jobs
- Schedule
  - Implemented validation to enforce user to choose days when picking schedule to avoid Java error messages
- Explorer
  - Fixed issue where '&' was not properly supported when adding additional parameters
- API
  - JSESSIONID session time is configurable
  - Bearer token and JSESSIONID authentication paths are properly forked
- Pattern
  - Patterns activity now shows Count (number of times the current dataset has the Pattern breaks). This Count is interpreted the same way as Outlier activity Count

2021.11

Enhancements

- Rules
  - *Tech Preview* [TP] Rule Discovery
    - The application now supports dynamic semantics checks. This allows you to create custom semantics that can be checked for when running a DQ check on a data set. Previously the application checked against a pre-defined set of semantics. You also have access to controls to organize and apply these semantics checks. The following is a list of changes:
      - There is a new data concepts management page. You can access it from Catalog or Admin Console. You can assign multiple semantics to a data concept.
      - When running a DQ check, you can select a data concept. The semantics assigned to this data concept will be checked against each column of dataset.
      - You have a list of predefined semantics that are not editable. You also have the ability to create/edit/delete custom semantics.
      - Repo on rules page has been added to Rules Library where semantics will be viewable.
- Resource Limits
  - You can edit the Performance Settings to supply limits to executors, cores, memory and cells so that a user can be warned if submitting a job that requires a lot of resources and admins can control maximum resources submitted.

Enhancements

- Explorer
  - *Tech Preview* [TP] Dynamic query reload allows you to view JOIN query columns in other activities.
    - User can update and reload the schema table with the custom query in the scope section by clicking the [Update Scope] button. It will enable using the new columns from the custom query in all activities (Profile,

Outlier, Dupes, Patterns, Source)
        - Since the first tab is for compositing the query, updating fields will change the user's custom query. Therefore, all areas are locked except the "query" field in the first tab to keep the query unchanged after updating the scope table
    - Support for some special characters in table name.
    - Fixed the ability to add additional libs that were previously not being properly saved on subsequent runs. Under DQ Job tag, please utilize -fllb boolean (union lookback) and libsrc input box for lib directory path (will materialize as -addlib).
- Connections
    - *Tech Preview* [TP] BigQuery Views and Joins
    - Please add the following to the BigQuery connection property

```
viewsEnabled=true
```

- API
    - You can perform multiple imports without conflicts.
    - You can have an incremental import such as updating matching records / insert new / leave existing. There is no requirement to delete tables first before running import.
- Profile
    - Fixed backrun timebin to work with weeks and quarters instead of days.
- Outliers
    - Split historical load to avoid historical query rounding up.
    - *Tech Preview* [TP] Outliers (advanced).
- Source
    - Fixed an issue where settings were not sticky for subsequent runs.
- Security
    - SAML Enhancements
        - New configuration settings are available when the Load Balancer is set for SSL Termination.
        - You can now set theMulti-tenancy support through SAML RelayState to route SSO to the proper tenant.

Patches

- 2021.11.1 Explorer
  - Allow ampersand in metastore host name for additional parameters
  - In below example, support for ampersand needed for required SSL flags

```
metastore01.us-east1-b.c.customer-dq-prod.in-
ternal:5432/dev?sslmode=required&currentSchema=public
```

Known Limitations

- Rules
  - Semantics and data concepts:
    - Not supported in pushdown mode
    - Exporting RegEx semantics not currently supported
  - While it is possible to create joins and cross-dataset rules using Freeform SQL, it is best practice to create a view and handle the join prior to running the DQ Job.
- Behavior
  - Schema is not eligible for invalidate
- Files
  - Local files using UPLOAD_PATH, UPLOAD_FILE_PATH, and temp files are only eligible to be deployed using the default NO_AGENT option. These are only intended for quick tests and not intended for production-scale use. Best practice is to use a remote file system connection (S3, Google storage or ADLS).
  - Delimiter support for special characters is limited. Supported file delimiters are comma, pipe, tab, semicolon, double quote and single quote. Custom delimiters will work for many characters, but not all combinations.
  - Temp files and NO_AGENT should have -master local[*] or -master spark://:7077 defined in freeform append of the agent options
- DQ Job
  - When submitting jobs via API from a different machine with a different timezone, timezone discrepancies are not accounted for automatically. Best

practice is to align each component to use UTC.
    - Jobs submitted via API with a run date that include HH:MM in the -rd (run date) will submit to the job queue and leave a remnant 'STAGED' job
- Connections
    - Postgres limits max connections per spark job. The default is 100. Please refer to Postgres official documentation how to increase max_connection and shared_buffers.
        - https://www.postgresql.org/docs/9.6/runtime-config-connection.html
    - BigQuery
        - Updating scope to include joins in BigQuery can only be materialized when tables are part of the same dataset collection
        - Should you receive an error for pre-existing BigQuery jobs, please add -dssafeoff to the cmd line or select 'Allow Overwrite' to enable this from Edit mode in the Explorer
- Alerts
    - After an upgrade to 2021.11, you may need to set the environment variable `ALERT_SCHEDULE_ENABLED=true` in **owl-env.sh** and restart **owl-web** to enable email alerts to work again.

2021.10

Enhancements

- DQ Job
    - Refactored DQ Job Score to Gauge Chart
- Explorer
    - Fixed issue where permissions are checked on datasets that do not yet exist
- Connections
    - Sybase 'Test / Preview' now available
    - Updated web model of saving additional connection properties
    - Fixed scenario where editing connection yields null instead of empty for multiple values
- Rules
    - Placeholder new searchable Rule Summary Page for Rule statistics / insights

- Alerts
  - Updated Alert Mailer to TLS 1.2 to resolve Third Party Error exception
  - Fixed issue where alerts are deleted even when clicking cancel button
- Behavior
  - Fixed issue where user must refresh to have invalidated item removed from UI
- Search
  - Fixed search on Audit Datasets and Dataset Management page
- Scorecards
  - Date ranges are now customizable
- Validate Source
  - Added feature that provides 'trim' option on String columns when running source-target validation, extra spaces in the cell are trimmed on both ends (left and right)
- Dupes
  - Resolved issue with white spaces in column headers blocking duplicate detection
- Security
  - Added configuration for setting the SAML_ENTITY_BASEURL, which sets the Consumer service url for the SP Metadata
- Shapes
  - Fixed issue where custom values override even after toggling Shapes back to auto or off
- Console
  - Fixed uncaught TypeError on login screen
  - Fixed GET timeout error on registration page
- Export/Import API
  - Users will be able to run the export/import API calls to conduct multiple promotions on the repo, schedule, and rule tables.

Patches

- 2021.10.1 Import / Export API without constraint conflicts
  - Import must match exactly to the format of our export in order to parse out columns and values to perform an update when existing records are already there

```
owl_rule
owl_check_repo
job_schedule
rule_repo
```

Known Limitations

- File sizes
  - Individual files greater than 5gb will experience performance degradation in Explorer for Standalone installs. Best practice is to save in smaller chunks and use bypass schema in the Explorer if needed.
  - Individual files greater than 25gb will experience performance degradation in Core for Standalone installs.
- Files
  - Explorer / browser will generally have difficulty supporting > 250 columns in files
- Profiling
  - Pushdown profiling on Bigquery, Redshift, Athena and Presto is available for specific datatypes.
  - Backrun option and flag will persist beyond the first run (-br). Please remove this flag if you do not want to backrun again.
- Explorer
  - QUARTER and WEEK are not supported time bins in this release.
  - On non-csv files, Explorer will not automatically infer file types. Users must change file type to the required value and click Step 2 "Load File". Nothing will change in Step 1 "File Information". A future enhancement will be added to automatically check filetypes by reading the first file
  - Dataset names should not contain special characters
- Rules
  - Out of the box semantic rules cannot be edited (STATECHECK, GENDERCHECK, etc). Users can still apply their own global rules which can be customized.
  - LinkId does not support alias columns that are not part of the -LinkId definition
- Connections
  - Connection names should not contain spaces

- Validate Source
    - Complex Validate Source queries can only be edited from the CMD line or JSON directly before hitting Run.
- Security
    - Active Directory in Azure SQL can connect via LDAP (basic auth) or Kerberos.
- S3 / GS / ADLS
    - Remote storage connections should be defined using the root bucket only.
- Estimate Job is only available for files when Livy is being used.
- Stop Job on jobs page is limited and does not work for all installation types.
- Bigquery connector does not work with views

Builds

This page provides information about Docker images and other information related to Collibra DQ build packages. Builds follow a naming convention to indicate which Optional Drivers drivers are packaged.

> Note **The optional drivers only impact container versions of Collibra DQ and do not impact Standalone installation packages.**

Available Containers

2023.03

Collibra Data Quality & Observability

- 2023.03-LM-2341
- 2023.03-ABDGCSILM-2340
- 2023.03-ABDGCSHILM-2339
- 2023.03-2342

Spark

- 3.2.2-2023.03-LM-2341
- 3.2.2-2023.03-ABDGCSILM-2340
- 3.2.2-2023.03-ABDGCSHILM-2339
- 3.2.2-2023.03-2342

2023.02

Collibra Data Quality & Observability

- 2023.02-LM-2188
- 2023.02-ABDGCSILM-2187
- 2023.02-ABDGCSHILM-2186
- 2023.02-2189

Spark

- 3.2.2-2023.02-LM-2188
- 3.2.2-2023.02-ABDGCSILM-2187
- 3.2.2-2023.02-ABDGCSHILM-2186
- 3.2.2-2023.02-2189

2023.01

Collibra Data Quality & Observability

- 2023.01-LM-2067
- 2023.01-ABDGCSILM-2066
- 2023.01-ABDGCSHILM-2065
- 2023.01-2068

Spark

- 3.2.2-2023.01-LM-2067
- 3.2.2-2023.01-ABDGCSILM-2066
- 3.2.2-2023.01-ABDGCSHILM-2065
- 3.2.2-2023.01-2068

2022.12

Collibra Data Quality & Observability

- 2022.12-LM-1960
- 2022.12-ABDGCSILM-1959
- 2022.12-ABDGCSHILM-1958
- 2022.12-1964

Spark

- 3.2.2-2022.12-LM-1960
- 3.2.2-2022.12-ABDGCSILM-1959
- 3.2.2-2022.12-ABDGCSHILM-1958
- 3.2.2-2022.12-1964

2022.11

Collibra Data Quality & Observability

- 2022.11-LM-1739
- 2022.11-ABDGCSILM-1738

- 2022.11-ABDGCSHILM-1737
- 2022.11-1736

Spark

- 3.2.2-2022.11-LM-1739
- 3.2.2-2022.11-ABDGCSILM-1738
- 3.2.2-2022.11-ABDGCSHILM-1737
- 3.2.2-2022.11-1736

2022.10

Collibra Data Quality & Observability

- 2022.10-ADGCSILM-1568
- 2022.10-ABDGCSILM-1569
- 2022.10-ABDGCSHILM-1570
- 2022.10-1572

Spark

- 3.2.2-2022.10-ADGCSILM-1568
- 3.2.2-2022.10-ABDGCSILM-1569
- 3.2.2-2022.10-ABDGCSHILM-1570
- 3.2.2-2022.10-1572

2022.09

Collibra Data Quality & Observability

- 2022.09-ADGCSILM-1386
- 2022.09-ABDGCSILM-1387
- 2022.09-ABDGCSHILM-1388
- 2022.09-1390

Spark

- 3.2.2-2022.09-ADGCSILM-1386
- 3.2.2-2022.09-ABDGCSILM-1387
- 3.2.2-2022.09-ABDGCSHILM-1388
- 3.2.2-2022.09-1390

2022.08

Collibra Data Quality & Observability

- 2022.08-L-1132
- 2022.08-AHM-1133
- 2022.08-H-1134
- 2022.08-HM-1135
- 2022.08-D-1136
- 2022.08-AL-1137
- 2022.08-AD-1138
- 2022.08-ABGCSHMS-1139
- 2022.08-AGCSHLM-1140
- 2022.08-M-1141
- 2022.08-GCSL-1142
- 2022.08-ADH-1143

Spark

- 3.2.0-2022.08-L-1132
- 3.2.0-2022.08-AHM-1133
- 3.2.0-2022.08-H-1134
- 3.2.0-2022.08-HM-1135
- 3.2.0-2022.08-D-1136
- 3.2.0-2022.08-AL-1137
- 3.2.0-2022.08-AD-1138
- 3.2.0-2022.08-ABGCSHMS-1139
- 3.2.0-2022.08-AGCSHLM-1140
- 3.2.0-2022.08-M-1141
- 3.2.0-2022.08-GCSL-1142
- 3.2.0-2022.08-ADH-1143

2022.07

Collibra Data Quality & Observability

- 2022.07-L-939
- 2022.07-AHM-940
- 2022.07-H-941
- 2022.07-HM-942
- 2022.07-D-943
- 2022.07-AL-944
- 2022.07-AD-945

- 2022.07-ABGCSHMS-947
- 2022.07-M-946

Spark

- 3.2.0-2022.07-L-939
- 3.2.0-2022.07-AHM-940
- 3.2.0-2022.07-H-946
- 3.2.0-2022.07-HM-942
- 3.2.0-2022.07-D-943
- 3.2.0-2022.07-AL-944
- 3.2.0-2022.07-AD-945
- 3.2.0-2022.07-ABGCSHMS-947
- 3.2.0-2022.07-M-946

2022.06

Collibra Data Quality & Observability

- 2022.06-L-819
- 2022.06-AHM-820
- 2022.06-H-821
- 2022.06-HM-822
- 2022.06-D-823
- 2022.06-AL-824
- 2022.06-AD-825
- 2022.06-ABGCSHMS-826
- 2022.06-M-830

Spark

- 3.2.0-2022.06-L-819
- 3.2.0-2022.06-AHM-820
- 3.2.0-2022.06-H-821
- 3.2.0-2022.06-HM-822
- 3.2.0-2022.06-D-823
- 3.2.0-2022.06-AL-824
- 3.2.0-2022.06-AD-825
- 3.2.0-2022.06-ABGCSHMS-826
- 3.2.0-2022.06-M-830

2022.05

Collibra Data Quality & Observability

- 2022.05-L-714
- 2022.05-AL-715
- 2022.05-H-716
- 2022.05-AHM-717
- 2022.05-ABGCSHMS-719
- 2022.05-D-721
- 2022.05-AD-723
- 2022.05-BDG-751
- 2022.05.2-L-737
- 2022.05.2-AHM-738
- 2022.05.2-HM-739
- 2022.05.2-H-740

Spark

- 3.2.0-2022.05-L-714
- 3.2.0-2022.05-AL-715
- 3.2.0-2022.05-H-716
- 3.2.0-2022.05-AHM-717
- 3.2.0-2022.05-ABGCSHMS-719
- 3.2.0-2022.05-D-721
- 3.2.0-2022.05-AD-723
- 3.2.0-2022.05-BDG-751
- 3.2.0-2022.05.2-L-737
- 3.2.0-2022.05.2-AHM-738
- 3.2.0-2022.05.2-HM-739
- 3.2.0-2022.05.2-H-740

2022.04

Collibra Data Quality & Observability

- 2022.04-L-303
- 2022.04-AL-302
- 2022.04-296
- 2022.04-A-295
- 2022.04-ALL-294 2
- 2022.04-ABHGCSGCRS-291

Spark

- 3.2.0-2022.04-L-303
- 3.2.0-2022.04-AL-302
- 3.2.0-2022.04-296
- 3.2.0-2022.04-A-295
- 3.2.0-2022.04-ALL-294
- 3.2.0-2022.04-ABHGCSGCRS-291

> Note The default build is considered the Secure Build with no optional drivers included and no critical vulnerabilities.

Config Map Example

These are the configs to change the versions. For a complete list of versions for Collibra DQ and Spark, refer to the Available Containers section above.

```
--set global.version.owl=2022.05-720 --set glob-
al.version.spark=3.2.0-2022.05-720
```

Pull Examples

docker pull https://gcr.io/owl-hadoop-cdh/owl-agent:2022.05-L-714

docker pull https://gcr.io/owl-hadoop-cdh/owl-livy:3.2.0-2022.05-L-714

docker pull https://gcr.io/owl-hadoop-cdh/owl-web:2022.05-L-714

docker pull https://gcr.io/owl-hadoop-cdh/spark:3.2.0-2022.05-L-714

Description

Example: 2022.05-L-714

2022-05: Release number (Year and Month)

L: Optional Livy package included **714** - A unique number appended to each build

Default Drivers

Always Packaged

- SQL Server
- Oracle
- Snowflake
- Redshift
- S3
- ADLS
- Postgres
- Mysql
- Teredata
- Sybase
- Db2
- Dremio

Optional Drivers

Please select drivers

- ABGCSHLMS (Drivers Initial in alphabetical order):
  - A : Athena
  - B : BigQuery
  - D = Databricks
  - GCS : Google Cloud Storage Connector
  - H : Hive
  - L : Livy
  - M : MongoDB
  - S : Solr

APIs

Rest

Please see the REST APIs section for more details.

```
import requests
import json

# Variables
owl = 'https://<ip_address>'              #Edit
user = '<user>'                           #Edit
password = '<password>'                   #Edit
tenant = 'public'                         #Edit
dataset = '<your_dataset_name>'           #Edit
runDate = '2021-08-08'                    #Edit
agentName = 'your_agent_name'             #Edit

# Authenticate
url = owl+'/auth/signin'
payload = json.dumps({"username": user, "password": password,
"iss": tenant })
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data-
a=payload, verify=False)
owl_header = {'Authorization': 'Bearer ' + response.json()
['token']}

# Run
response = requests.post(url = owl + '/v3/-
job-
s/run-
?agentName='+agentName+'&dataset='+dataset+'&runDate='+runDate,
headers=owl_header, verify=False)
jobId = str(response.json()['jobId'])

# Wait
waitForJobCompletion(100)

# Results
response = requests.get(url = owl + '/v3/-
jobs/'+jobId+'/findings', headers=owl_header,  verify=False)
```

Notebook

Please see the Databricks example for more information

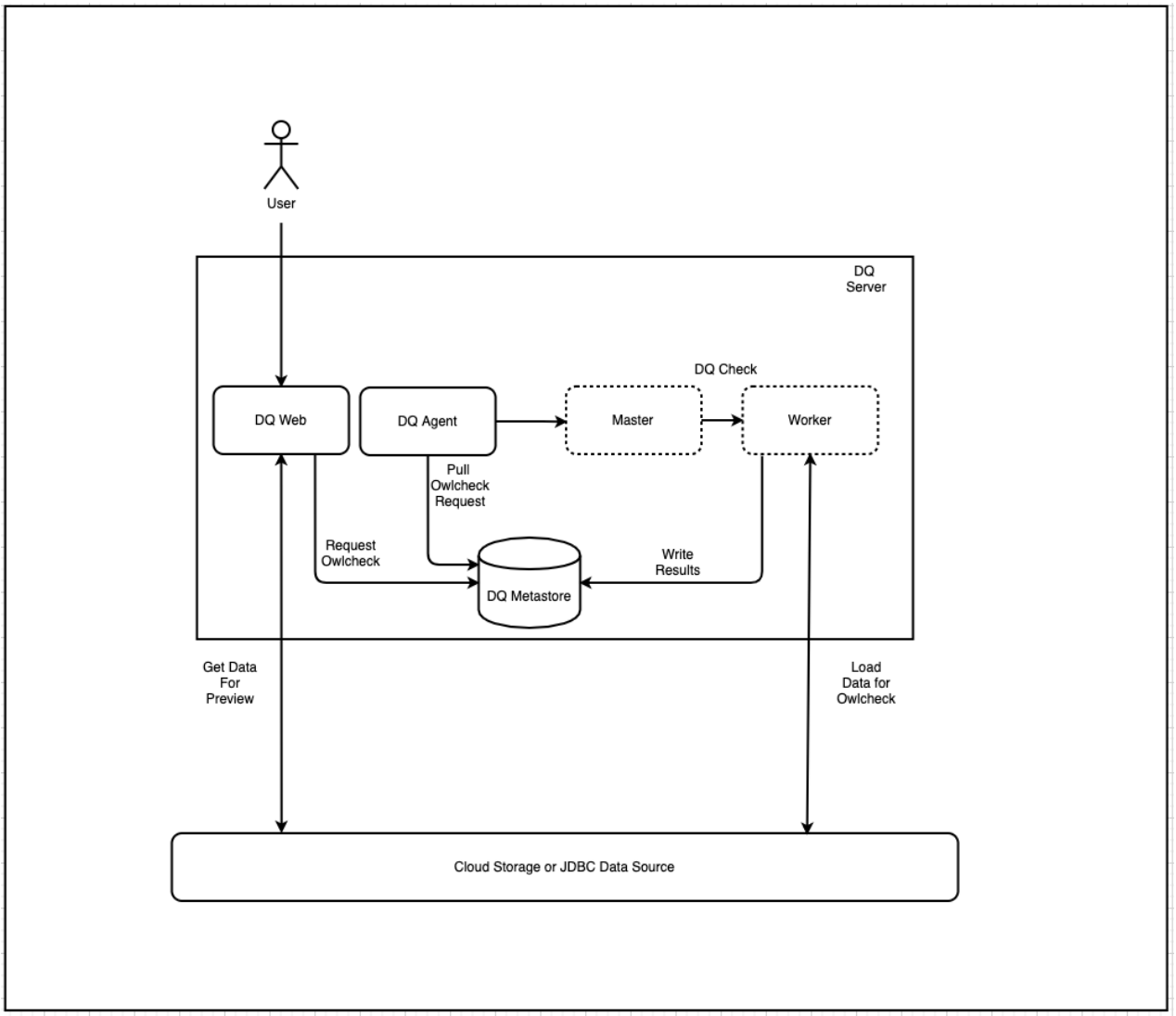
```
<MadCap:concept term="Data Quality" xmlns:MadCap-
p="http://www.madcapsoftware.com/Schemas/MadCap.xsd" />val data-
set = "cdq_notebook"
var date = "2018-01-11"

// Options
val opt = new OwlOptions()
opt.dataset = dataset
opt.runId = date
opt.host = pgHost
opt.port = pgPort
opt.pgUser = pgUser
opt.pgPassword = pgPass

// Pre Routine
val cdq = com.owl.core.util.OwlUtils.OwlContext(df, opt)
.register(opt)

// Scan
cdq.owlCheck()
val results =  cdq.hoot()
```

Chapter 5

Collibra DQ Installation

Standalone

When large scale and high concurrency checks are not required, DQ can be installed and operated entirely on a single host. In this mode, DQ will leverage a **Spark Standalone pseudo cluster** where the master and workers run and use resources from the same server. DQ also requires a Postgres database for storage and Java 8 for running the DQ web application. It is possible to install each of the Spark, Postgres, and Java 8 components separately and install DQ on top of existing components. However, we offer a full installation package that installs these components in off-line mode and install DQ in one server.



Standalone Install

1. Setup Tutorial Assumptions

We assume that a server running CentOS 7/CentOS 8 or RHEL 7/RHEL 8 is set up and ready to install Collibra DQ in the home directory (base path: `OWL_BASE`) under the subdirectory `owl` (install path: `$OWL_BASE/owl`). There is no requirement for Collibra DQ to be installed in the home directory, but the Collibra DQ Full Installation script may lead to permission-denied issues during local Postgres server installation if paths other than the home directory are used. If these issues occur, please adjust your directory permission to allow the installation script a write access to the Postgres data folder.

> Note Collibra DQ is supported to run on RHEL 8. You can use your PostgresSQL database instance.

This tutorial assumes that you are installing Collibra DQ on a brand new compute instance on Google Cloud Platform. Google Cloud SDK setup with proper user permission is assumed. This is optional, as you are free to create Full Standalone Installation setup on any cloud service provider or on-premise.

Please refer to the **GOAL** section for the intended outcome of each step and modify accordingly.

> Note The full install package supports CentOS 7/CentOS 8 and RHEL 7/RHEL 8. If another OS is required, please follow the basic install process. You must have a Postgres database installed and accessible, prior to running an RHEL 8 installation package. The Postgres repository is not compatible with RHEL 8.

> Note Collibra DQ Metastore supports AWS RDS or AWS Aurora.

```
# Create new GCP Compute Instance named "install"
gcloud compute instances create install \
    --image=centos-7-v20210701 \
    --image-project=centos-cloud \
    --machine-type=e2-standard-4

# SSH into the instance as user "centos"
gcloud compute ssh --zone "us-central1-a" --project "gcp-
example-project" "centos@full-standalone-installation"
```

GOAL

1. Create a new compute instance on a cloud provider (if applicable).
2. Access the server where Collibra DQ will be installed.

2. Download DQ Full Package

Download full package tarball using the signed link to the full package tarball provided by the Collibra DQ Team. Replace `<signed-link-to-full-package>` with the link provided.

```
### Go to the OWL_BASE (home directory of the user is most com-
mon)
### This example uses /home/owldq installing as the user owldq

cd /home/owldq

### Download & untar
curl -o dq-full-package.tar.gz "<signed-link-to-full-package>"
tar -xvf dq-full-package.tar.gz

### Clean-up unnecessary tarball (optional)
rm dq-full-package.tar.gz
```

GOAL

1. Download the full package tarball and place it in the `$OWL_BASE` (home directory). Download via `curl` or upload directly via FTP. The tarball name is assumed to be `dq-full-package.tar.gz` for the sake of simplicity.
2. Untar `dq-full-package.tar.gz` to `OWL_BASE`.

3. Install Collibra DQ + Postgres + Spark

First set some variables for `OWL_BASE` (where to install Collibra DQ. In this tutorial, you are already in the directory that you want to install), `OWL_METASTORE_USER` (the Postgres username used by DQ Web Application to access Postgres storage), and `OWL_METASTORE_PASS` (the Postgres password used by DQ Web Application to access Postgres storage).

```
### The base path that you want Collibra DQ installed under.
Trailing spaces are not permitted.

export OWL_BASE=$(pwd)
export OWL_METASTORE_USER=postgres
# minimum complexity recommended (18 length, upper, lower, num-
ber, symbol)
# example below
export OWL_METASTORE_PASS=H55Mt5EbXh1a%\$aiX6
# The default Spark version is 3.0.1, but you can optionally
install Spark 3.2.2 or 3.2.1 instead.
# Spark 3.2.2
export SPARK_PACKAGE=spark-3.2.2-bin-hadoop3.2.tgz
# Spark 3.1.2
export SPARK_PACKAGE=spark-3.1.2-bin-hadoop3.2.tgz
```

> Warning The `$` symbol is not a supported special character in your Postgres Metastore password.

`dq-package-full.tar.gz` that you untarred contains installation packages for Java 9 or Java 11, Postgres 11, and Spark. There is no need to download these components. These off-line installation components are located in `$(pwd)/package/install-packages`.

```
[centos@install ~]$ pwd
/home/centos
[centos@install ~]$ ls packages/install-packages/
postgresql11-11.9-1PGDG.rhel7.x86_64.rpm          postgresql11-libs-11.9-1PGDG.rhel7.x86_64.rpm    spark-3.0.1-bin-hadoop3.2.tgz
postgresql11-contrib-11.9-1PGDG.rhel7.x86_64.rpm  postgresql11-server-11.9-1PGDG.rhel7.x86_64.rpm  spark-extras.tar.gz
[centos@install ~]$
```

One of the files extracted from the tarball is `setup.sh`. This script installs Collibra DQ and the required components. If a component already exist (for example, Java 8 is already installed and `$JAVA_HOME` is set), then that component is not installed (i.e. Java 8 installation is skipped).

To control which components are installed, use the `-options=...` parameter. The argument provided should be a comma-delimited list of components to install (valid inputs: `spark`, `postgres`, `owlweb`, and `owlagent`. `-options=postgres,spark,owlweb,owlagent` means "install Postgres, Spark pseudo cluster, DQ Web Application, and DQ Agent". Note that Java is not part of the options. Java 8 or Java 11 installation is automatically checked and installed/skipped depending on availability.

At a minimum, you must specify `-options=spark,owlweb,owlagent` if you independently installed Postgres or use an external Postgres connection (as shown in Step 4, if you choose that installation route).

```
### The following installs PostgresDB locally as part of Col-
libra DQ install

./setup.sh \
    -owlbase=$OWL_BASE \
    -user=$OWL_METASTORE_USER \
    -pgpassword=$OWL_METASTORE_PASS \
    -options=postgres,spark,owlweb,owlagent
```

> Warning The `$` symbol is not a supported special character in your Postgres Metastore password.

> Note If you are prompted to install Java 8 or Java 11 because you do not have one of them installed, accept to install from a local package.

You are prompted to select a location to install Postgres, as shown below:

```
Postgres DB needs to be intialized. Default location = <OWL_
BASE>/postgres/data
to change path please enter a FULL valid path for Postgres and
hit <enter>
DB Path [ <OWL_BASE>/owl/postgres/data ] =
```

If the data files for the Postgres database need to be hosted at a specific location, provide it during this prompt. **Ensure the directory is writable**. Otherwise, press <Enter> to install

the data files into `$OWL_BASE/owl/postgres/data`. The default suggested path does not have permission issues if you use `OWL_BASE` as the home directory.

If no exceptions occur and the installation is successful, then the process completes with the following output:

```
installing owlweb
starting owlweb
starting owl-web
installing agent
not starting agent
install complete
please use owl owlmanage utility to configure license key and
start owl-agent after owl-web successfully starts up
```

GOAL

1. Specify `OWL_BASE` path where Collibra DQ will be installed and specify Postgres environment variables.
2. Install DQ Web with Postgres and Spark linked to DQ Agent (**all files are in the `$OWL_BASE/owl` sub-directory**) using `setup.sh` script provided. The location of `OWL_BASE` and Postgres are configurable, but we advise you take the defaults.

4. Install DQ + Spark and use existing Postgres (advanced)

> Note Skip Step 4 if you opted to install Postgres and performed Step 3 instead.
>
> We recommend Step 4 over Step 2 for advanced DQ Installer.

If you already installed DQ in the previous step, then skip this step. This is only for those who want to use external Postgres (e.g. use GCP Cloud SQL service as the Postgres metadata storage). If you have an existing Postgres installation, then everything in the previous step applies except the Postgres data path prompt and the `setup.sh` command.

Refer to Step 3 for details on `OWL_BASE`, `OWL_METASTORE_USER` , and `OWL_METASTORE_PASS`.

```
# The base path that you want Collibra DQ installed. No trailing

export OWL_BASE=$(pwd)
export OWL_METASTORE_USER=postgres

# minimum complexity recommended (18 length, upper, lower, num-
ber, symbol)
# example below
export OWL_METASTORE_PASS=H55Mt5EbXh1a%$aiX6
```

Run the following installation script. Note the missing "Postgres" in `-options` and new parameter `-pgserver`. This `-pgserver` could point to any URL that the standalone instance has access to.

```
# The following does not install PostgresDB and
# uses existing PostgresDB server located in localhost:5432 with
"postgres" database
./setup.sh \
    -owlbase=$OWL_BASE \
    -user=$OWL_METASTORE_USER \
    -pgpassword=$OWL_METASTORE_PASS \
    -options=spark,owlweb,owlagent \
    -pgserver="localhost:5432/postgres"
```

The database named `postgres` is used as the default DQ metadata storage. Changing this database name is out-of-scope for Full Standalone Installation. Contact the Collibra DQ team for assistance.

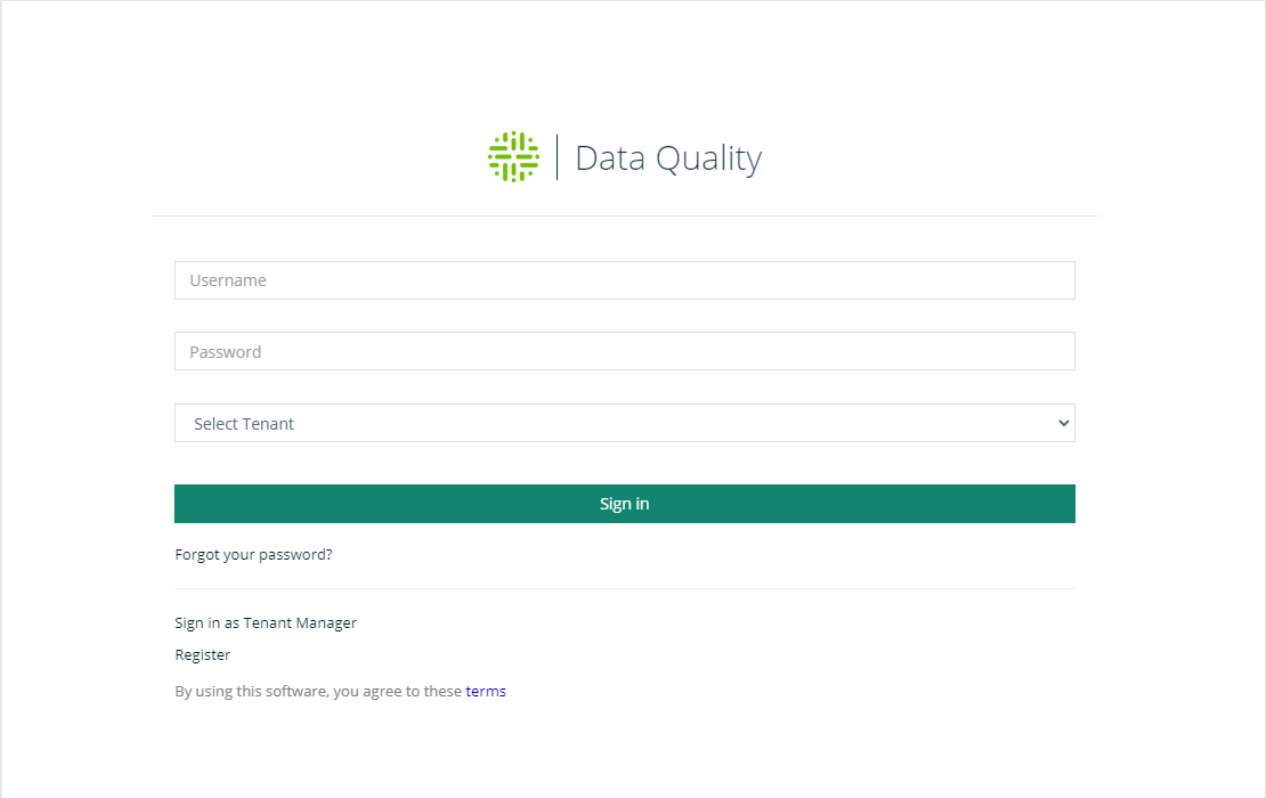
GOAL

1. Specify `OWL_BASE` path where Collibra DQ will be installed and specify Postgres environment variables
2. Install DQ Web and Spark linked to DQ Agent (**all files will be in `$OWL_BASE/owl` sub-directory**) using `setup.sh` script provided and link DQ Web to an existing Postgres server.

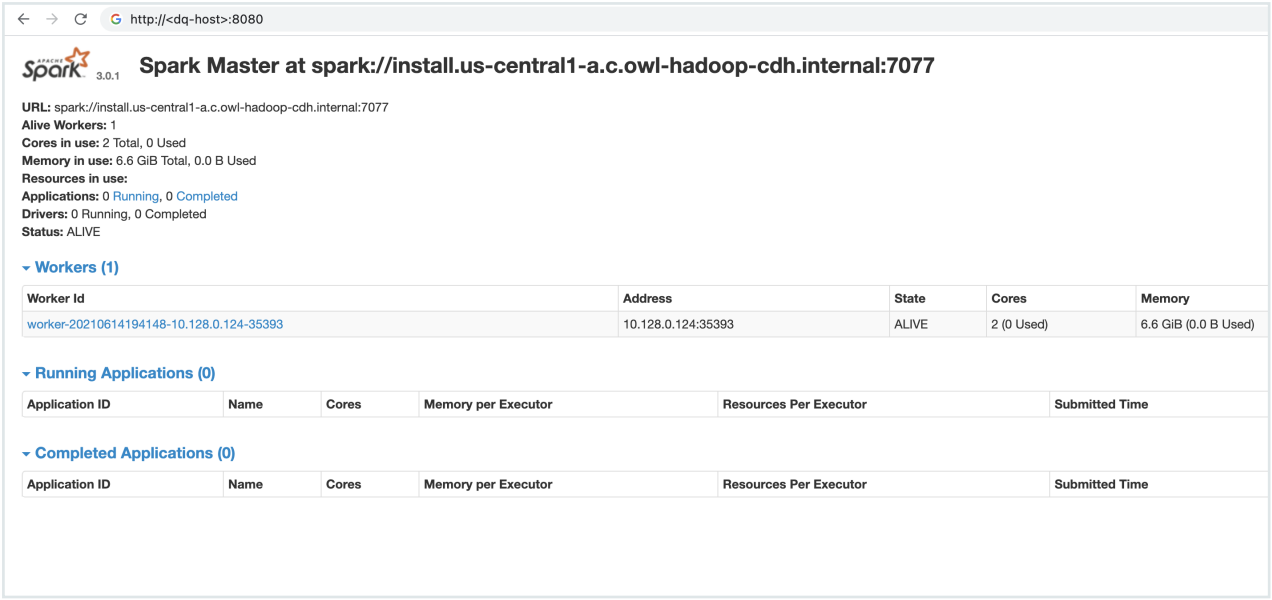
5. Verify DQ and Spark Installation

The installation process will start the DQ Web Application. This process initializes the Postgres metadata storage schema in Postgres under the database named `postgres`.

This process must complete successfully before the DQ Agent can be started. Wait approximately 1 minute for the Postgres metadata storage schema to be populated. If you can access DQ Web using `<url-to-dq-web>:9000` using a Web browser, then this means you have successfully installed DQ.



Next, verify that the Spark Cluster has started and is available to run DQ checks using `<url-to-dq-web>:8080`. Take note of the Spark Master URL starting with `spark://...`. This is required during DQ Agent configuration.



6. Set License Key

In order for Collibra DQ to run checks on data, the DQ Agent must be configured with a license key. Replace `<license-key>` with a valid license key provided by Collibra.

> Note Your license key is the value following `YOUR KEY IS =` in the license provision email.

```
cd $OWL_BASE/owl/bin
./owlmanage.sh setlic=<license-key>

# expected output:
# > License Accepted new date: <expiration-date>
```

7. Set License Name

It is required that you set a license name upon your initial deployment of Collibra DQ.

Replace `<your-license-name>` with a valid license name provided by Collibra.

> Note Your license name is the value following `YOUR NAME IS =` in the license provision email.

```
vi /<install-dir>/owl/config/owl-env.sh
export LICENSE_NAME=<your-license-name>
```

8. Set DQ Agent Configuration

Start the DQ Agent process to enable processing of DQ checks.

```
# 1 Start the agent to create the agent.properties file.
cd $OWL_BASE/owl/bin
./owlmanage.sh start=owlagent

# 2 Stop the agent and add this line to agent.properties:
./owlmanage.sh stop=owlagent

# 3 Add this line to agent.properties:
sparksubmitmode=native
sparkhome=</your/spark/home/folder>

# 4 Start the agent again.
./owlmanage.sh start=owlagent

# 5 Verify that agent.properties contains the correct details.
cd $OWL_BASE/owl/config
cat $OWL_BASE/owl/config/agent.properties
```

When the script successfully runs, the `$OWL_BASE/owl/config` folder contains a file called `agent.properties`. This file contains agent ID and the number of agents installed in your machine. Since this is the first non-default agent installed, the expected agent ID is 2. Verify that the `agent.properties` file is created. Your`agent.properties` is expected to have a different timestamp, but you should see `agentid=2`.

```
cd $OWL_BASE/owl/config
cat agent.properties

# expected output:
> #Tue Jul 13 22:26:19 UTC 2021
> agentid=2
```

Once the DQ Agent starts, it needs to be configured in DQ Web in order to successfully submit jobs to the local Spark (pseudo) cluster.

The new agent is set up with the template base path `/opt` and install path `/opt/owl`. The `owlmanage.sh start=owlagent` script does not respect `OWL_BASE` environment. **You must edit the Agent Configuration to follow the** `OWL_BASE`.

Follow the steps in the Agent section to configure the newly created DQ Agent and edit the following parameters in DQ Agent #2.

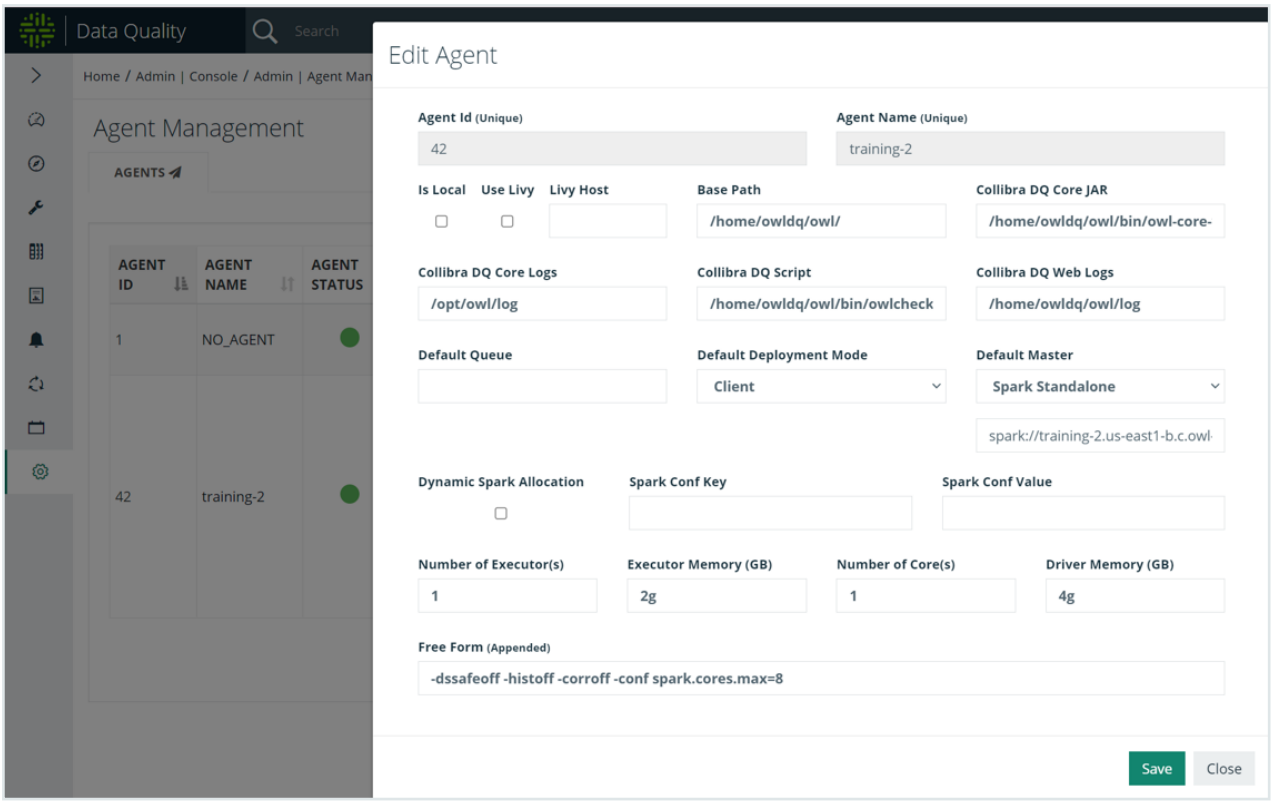
- Replace all occurrence of `/opt/owl` with your `$OWL_BASE/owl/`in **Base Path**, **Collibra DQ Core JAR**, **Collibra DQ Core Logs**, **Collibra DQ Script**, and **Collibra DQ Web Logs**.
    - Note that **Base Path** here does not refer to `OWL_BASE`
- Replace **Default Master** value with the Spark URL from Fig 3
- Replace **Number of Executors(s)**, **Executor Memory (GB)**, **Driver Memory (GB)** to a reasonable default based on the size of your instance.

Refer to the Agent section for descriptions of the parameters.

Specify the **Number of Core(s)**.

To limit Spark cores from being used for each job, a common configuration for the **Free Form (Appended)** field is `-conf spark.cores.max=8`.

Set the **Default Deployment Mode** option to **Client** for a Spark Standalone master.



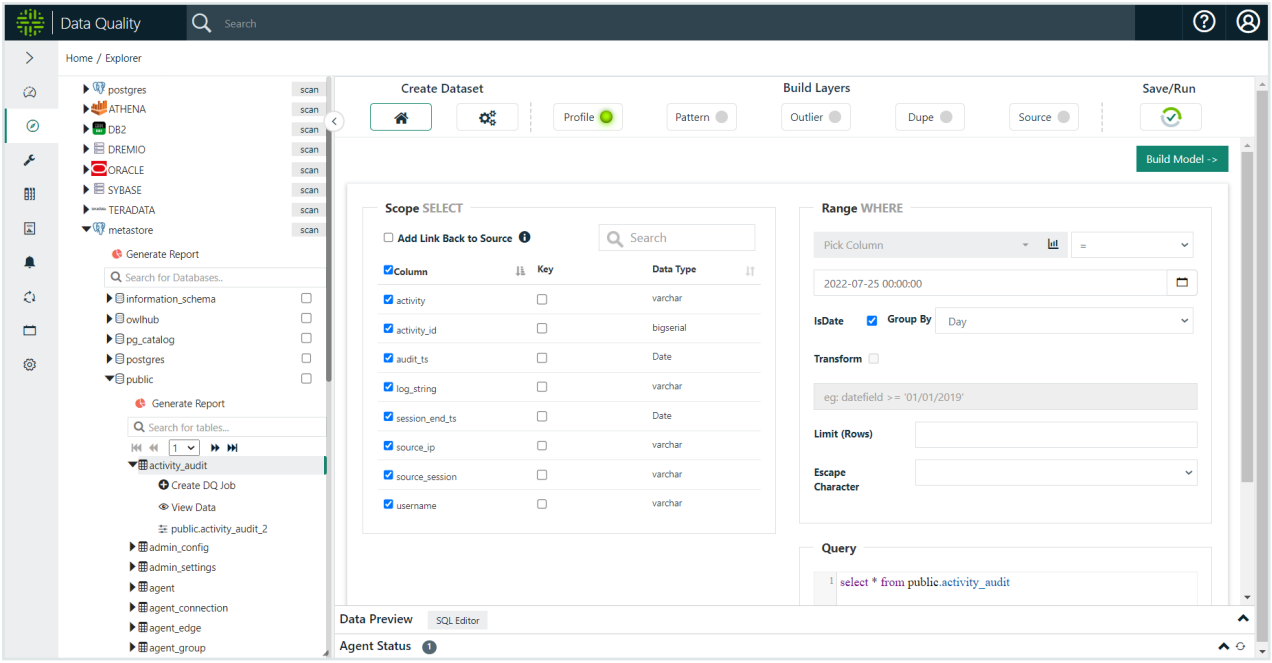
9. Create DB Connection for DQ Job

Follow the steps on Agent to add metastore database connection.

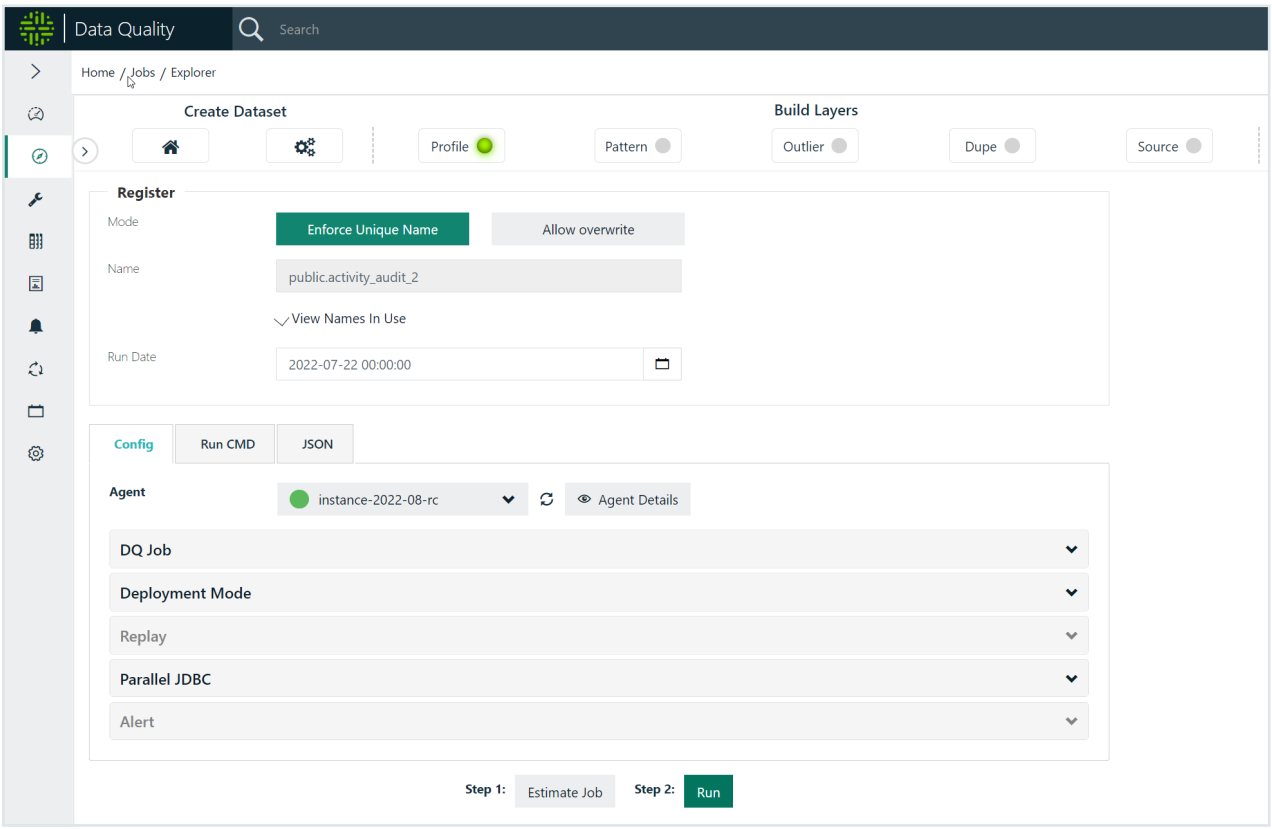
> Note In the following examples, a DQ Job is run against local DQ Metadata Storage.

Follow the steps on Agentsection to configure newly created DQ Agent.

Click the compass icon in the navigation pane to open the Explorer page. Click the **metastore** connection, select the **public** schema, and select the first table in the resulting list of tables. From the Preview and Scope page, click **Build Model**. When the Profile page populates, click **Save/Run**.

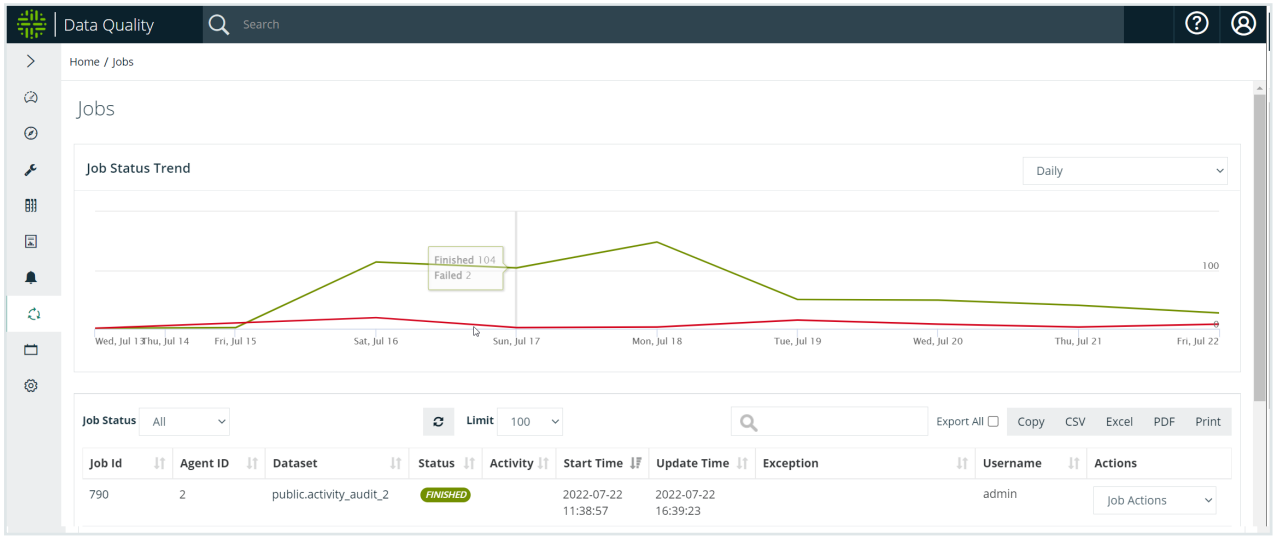


On the Run page, click **Estimate Job**, acknowledge the resource recommendations, and click **Run**.



Click the revolving arrows icon in the left navigation panel to open the Jobs page.

Wait 10 seconds and then click the refresh button above the Status column until the status shows that the DQ job is Finished. We recommend refreshing several times, pausing for a few seconds in between clicks. While a job runs, the Activity column tracks the sequence of activities DQ performs before it completes a job. A successful job shows its status as Finished last.



Troubleshooting + Helpful Commands

```
### Setting permissions on your pem file for ssh access

chmod 400 ~/Downloads/ssh_pem_key
```

Make sure working directory has permissions

For example, if I SSH into the machine with user `owldq` and use my default home directory location `/home/owldq/`

```
### Ensure appropriate permissions
### drwxr-xr-x

chmod -R 755 /home/owldq
```

Reinstall Postgres

```
### Postgres data directly initialization failed
### Postgres permission denied errors
### sed: can't read /home/owldq/owl/-
postgres/data/postgresql.conf: Permission denied

sudo rm -rf /home/owldq/owl/postgres
chmod -R 755 /home/owldq

### Reinstall just postgres
./setup.sh -owlbase=$OWL_BASE -user=$OWL_METASTORE_USER -pgpass-
word=$OWL_METASTORE_PASS -options=postgres
```

Changing Postgres password from SSH

```
### If you need to update your postgres password, you can lever-
age SSH into the VM
### Connect to your hosted instance of Postgres

sudo -i -u postgres
psql -U postgres
\password
#Enter new password: ### Enter Strong Password
#Enter it again: ### Re-enter Strong Password
\q
exit
```

> Warning The `$` symbol is not a supported special character in your Postgres Metastore password.

Permissions for ssh keys when starting Spark

```
### Spark standalone permission denied after using ./start-
all.sh

ssh-keygen -t rsa -N "" -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Permissions if log files are not writable

```
### Changing permissiongs on individual log files

sudo chmod 777 /home/owldq/owl/pids/owl-agent.pid
sudo chmod 777 /home/owldq/owl/pids/owl-web.pid
```

Getting the hostname of the instance

```
### Getting the hostname of the instance

hostname -f
```

Checking/deleting space of spark worker directory

```
### Checking worker nodes disk space

sudo du -ah | sort -hr | head -5
sudo find /home/owldq/owl/spark/work/* -mtime +1 -type f -delete
```

Increase Thread pool / Thread Pool Exhausted

Adding to the owl-env.sh script

```
# vi owl-env.sh
# modify these lines

export SPRING_DATASOURCE_POOL_MAX_WAIT=500
export SPRING_DATASOURCE_POOL_MAX_SIZE=30
export SPRING_DATASOURCE_POOL_INITIAL_SIZE=5

# restart web and agent
```

Adding to the Spark agent configurations

If you see the following message, update the agent configurations within load-spark-env.sh:

```
Failed to obtain JDBC Connection; nested exception is org.a-
pache.tomcat.jdbc.pool.PoolExhaustedException: [pool-29-thread-
2] Timeout: Pool empty. Unable to fetch a connection in 0
seconds, none available[size:2; busy:1; idle:0; lastwait:200].
```

Adjust the following configurations to modify the connection pool available:

```
export SPRING_DATASOURCE_POOL_MAX_WAIT=1000
export SPRING_DATASOURCE_POOL_MAX_SIZE=30
export SPRING_DATASOURCE_POOL_INITIAL_SIZE=5
```

> Note The load-spark-env.sh file is located in the $SPARK_HOME/bin folder.

Adding to the owl.properties file

Depending on client vs. cluster mode and cluster type, you may also need to add the following configurations in the owl.properties file:

```
spring.datasource.tomcat.initial-size=5
spring.datasource.tomcat.max-active=30
spring.datasource.tomcat.max-wait=1000
```

Active Database Queries

```
select * from pg_stat_activity where state='active'
```

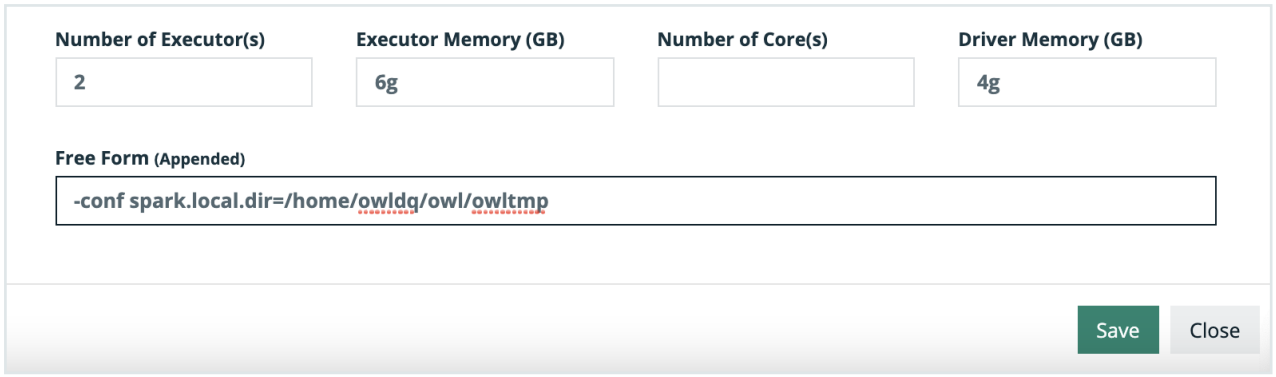
Too many open files configuration

```
### "Too many open files error message"
### check and modify that limits.conf file
### Do this on the machine where the agent is running for Spark
standalone version

ulimit -Ha
cat /etc/security/limits.conf

### Edit the limits.conf file
sudo vi /etc/security/limits.conf

### Increase the limit for example
### Add these 3 lines
fs.file-max=500000
*               soft    nofile          58192
*               hard    nofile          100000
### do not comment out the 3 lines (no '#'s in the 3 lines
above)
```

Redirecting Spark Scratch

```
### Redirect Spark scratch to another location
SPARK_LOCAL_DIRS=/mnt/disks/sdb/tmp

### Set Spark to delete older files
export SPARK_WORKER_OPTS="${SPARK_WORKER_OPTS} -Dspark.-
worker.cleanup.enabled=true -Dspark.worker.cleanup.interval=1800
-Dspark.worker.cleanup.appDataTtl=3600"
```

Or change Spark storage with an agent configuration `-conf spark.local.dir=/home/owldq/owl/owltmp`



Tip: Add Spark Home Environment Variables to Profile

```
### Adding ENV variables to bash profile

### Variable 'owldq' below should be updated wherever installed
e.g. centos

vi ~/.bash_profile
export SPARK_HOME=/home/owldq/owl/spark
export PATH=$SPARK_HOME/bin:$PATH

### Add to owl-env.sh for standalone install

vi /home/owldq/owl/config/owl-env.sh
export SPARK_HOME=/home/owldq/owl/spark
export PATH=$SPARK_HOME/bin:$PATH
```

Check Processes are Running

```
### Checking PIDS for different components

ps -aef|grep postgres
ps -aef|grep owl-web
ps -aef|grep owl-agent
ps -aef|grep spark
```

Starting Components

```
### Restart different components

cd /home/owldq/owl/bin/
./owlmanage.sh start=postgres
./owlmanage.sh start=owlagent
./owlmanage.sh start=owlweb

cd /home/owldq/owl/spark/sbin/
./stop-all.sh
./start-all.sh
```

Configuration Options

Setup.sh arguments

Argument	Description
-non-interactive	Skips asking to accept Java license agreement.
-skipSpark	Skips the extraction of Spark components.
-stop	Do not automatically start all components (Owl-Web, Zeppelin, Postgres).
-port=	Set DQ Web application to use the defined port.
-user=	Optional parameter to set the user to run Collibra DQ. The default is the current user.
-owlbase=	Sets the base path to where you want Collibra DQ installed.
-owlpackage=	Optional parameter to set the Collibra DQ package directory. The default is the current working directory.
-help	Display this help and exit.
-options=	The different Collibra DQ components to install in a comma-separated list format. For example, -options=owlagent,owlweb,postgres,spark

Argument	Description
-pgpassword=	The password used to set for the Postgres metastore. For unattended installs.
-pgserver=	The name of the Postgres server. For example, -pgserver=owl-postgres-host.example.com:5432/owldb. For unattended installs.

Example:

`./setup.sh -port=9000 -user=ec2-user -owlbase=/home/ec2-user -owlpackage=/home/ec2-user/packages`

- The tar ball extracted to this folder on my EC2 Instance: **** `/home/ec2-user-/packages/`
- Collibra DQ is running as the **** `ec2-user`
- The DQ Web application runs on port `9000`
- The base location for the setup.sh script to create the will be: `/home/ec2-user/`

Example installing just the agent (perhaps on an Edge node of a hadoop cluster):

`./setup.sh -user=ec2-user -owlbase=/home/ec2-user -owlpackage=/home/ec2-user/package -options=owlagent`

- The package extracted to this folder on my EC2 Instance: **** `/home/ec2-user-/packages/`
- Owl-agent is running as the **** `ec2-user`
- The base location for the setup.sh script to create the Collibra DQ folder and place all packages under Collibra DQ is: `/home/ec2-user/`

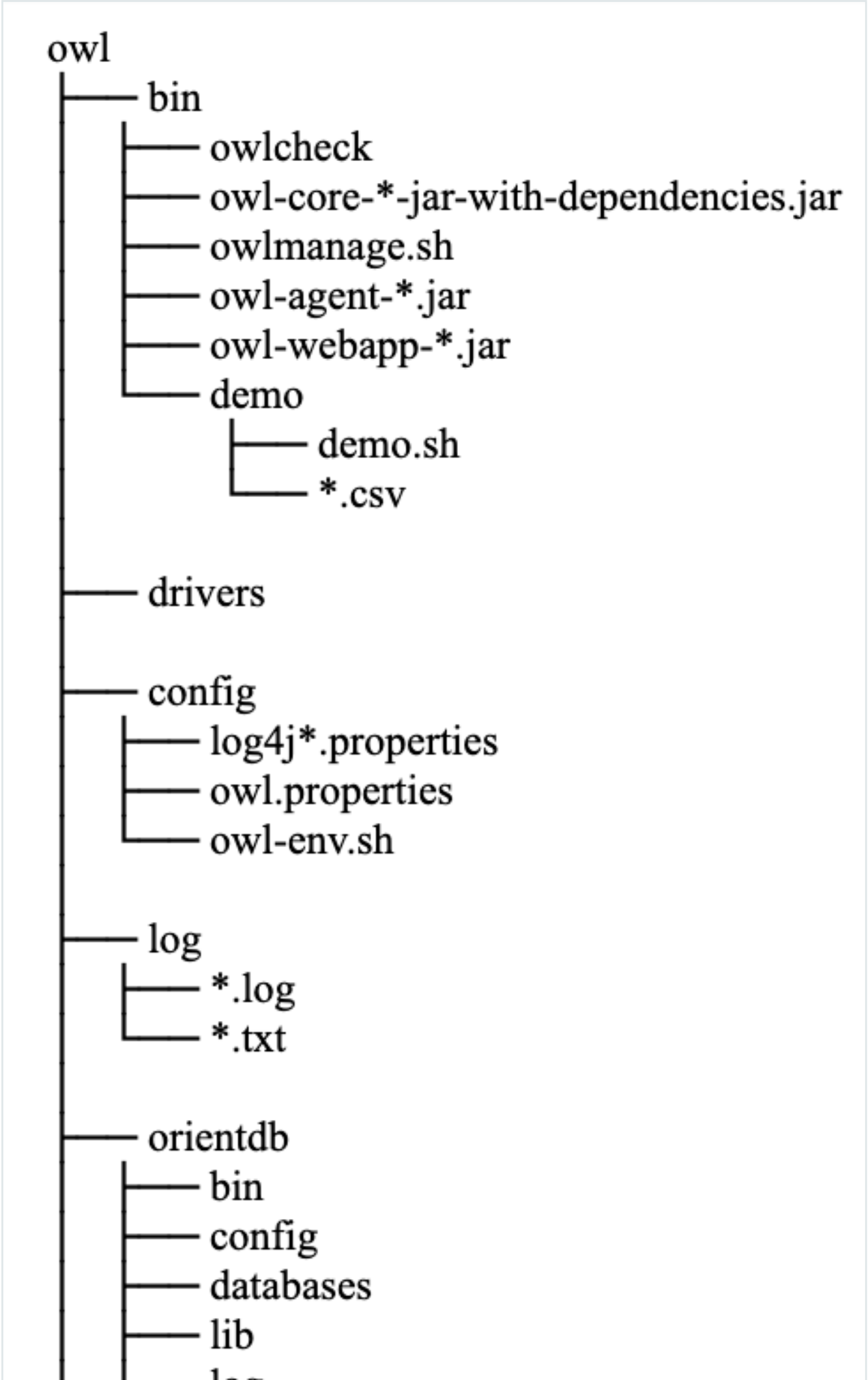
When installing different features, the following questions are asked:

- postgres = Postgres DBPassword needs to be supplied.
- If postgres is not being installed (such as agent install only) postgres metastore server name needs to be supplied.

Launching and Administering Collibra DQ

When the setup.sh script finishes by default software is automatically started. The setup.sh also creates the owlmanage.sh script which allows for stopping and starting of all owl services or some components of services. The setup script will also generate an owl-env.sh script that will hold the main variables that are reused across components (see owl-env.sh under the config directory).

Collibra DQ Directory Structure after running Setup.sh


owl
bin
owlcheck
owl-core-*-jar-with-dependencies.jar
owlmanage.sh
owl-agent-*.jar
owl-webapp-*.jar
demo
demo.sh
*.csv
drivers
config
log4j*.properties
owl.properties
owl-env.sh
log
*.log
*.txt
orientdb
bin
config
databases
lib

Configuration of ENV settings within owl-env.sh

Contents of the owl-env.sh script and what the script is used for.

owl-env.sh Scripts	Description
export SPARK_CONF_DIR="/home/collibra/owl/cdh-spark-conf"	The directory on your machine where the Spark conf directory resides.
export INSTALL_PATH="/home/collibra/owl"	The installation directory of Collibra DQ.
export JAVA_HOME="/home/collibra/jdk1.8.0_131"	Java Home for Collibra DQ to leverage.
export LOG_PATH="/home/collibra/owl/log"	The log path.
export BASE_PATH="/home/collibra"	The base location under which the Collibra DQ directory resides.
export SPARK_MAJOR_VERSION=2	Spark Major version. Collibra DQ only supports 2+ version of Spark.
export OWL_LIBS="/home/collibra/owl/libs"	Lib Directory to inject in spark-submit jobs.
export USE_LIBS=0	Use the lib directory or not. 0 is the default. A value of 0 means the lib directory is not used. A value of 1 means the lib directory is used.
export SPARKSUBMITEXE="spark-submit"	Spark submit executable command. Collibra DQ using spark-submit as an example.

owl-env.sh Scripts	Description
export ext_pass_manage=0	If using a password management system. You can enable for password to be pulled from it. A value of 0 disables an external password management system. A value of 1 enables an external password management system.
export ext_pass_script="/opt/owl/bin/getpassword.sh"	Leverage password script to execute a get password script from the vault.
TIMEOUT=900 #15 minutes in seconds	Owl-Web user time out limits.
PORT=9003 #owl-web port NUMBER	The default port to use for owl-web.
SPRING_LOG_LEVEL=ERROR	The logging level to be displayed in the owl-web.log
SPRING_DATASOURCE_DRIVER_CLASS_NAME-E=org.postgresql.Driver	The driver class name for postgres metastore (used by web).
export SPRING_DATASOURCE_URL-L=jdbc:postgresql://localhost:5432/postgres	JDBC connection string to Collibra DQ Postgres metastore.
export SPRING_DATASOURCE_USERNAME-E=collibra	Collibra DQ Postgres username.
export SPRING_DATASOURCE_PASSWORD-D=3+017wfY1l1vmsvGYAyUcw5zGL	Collibra DQPostgres password.
export AUTOCLEAN=TRUE/FALSE	TRUE/FALSE Enable/Disable automatically delete old datasets.
export DATASETS_PER_ROW=200000	Delete datasets after this threshold is hit (must be greater than the default to change).

owl-env.sh Scripts	Description
export ROW_COUNT_THRESHOLD=300000	Delete rows after this threshold is hit (must be greater than the default to change).
export SERVER_HTTP_ENABLED=true	Enabling HTTP to owl web
export OWL_ENC=OFF #JAVA for java encryption	Enable Encryption (NOTE need to add to owl.properties also). Has to be in form owl.enc=OFF within owl.properties file to disable, and in this form owl.enc=JAVA to enable. the owl.properties file is located in the owl install path /config folder (/opt/owl/config).
PGDBPATH=/home/collibra/owl/owl-postgres/bin/data	Path for Postgres DB
export RUNS_THRESHOLD=5000	Delete runs after this threshold is hit (must be greater than the default to change).
export HTTP_SECONDARY_PORT=9001	Secondary HTTP port to use which is useful when SSL is enabled.
export SERVER_PORT=9000	Same as PORT.
export SERVER_HTTPS_ENABLED=true	Enabling of SSL.
export SERVER_SSL_KEY_TYPE=PKCS12	Certificate trust store.
export SERVER_SSL_KEY_PASS-S=t2IMFWEHsQha3QaWnNaR8ALaFPH15Mh9	Certificate key password.
export SERVER_SSL_KEY_ALIAS=owl	Certificate key alias.
export SERVER_REQUIRE_SSL=true	Override HTTP on and force HTTPS regardless of HTTP settings.
export MULTITENANTMODE=FALSE	Flipping to TRUE will enable multi tenant support.

owl-env.sh Scripts	Description
export multiTenantSchemaHub=owlhub	Schema name used for multi tenancy.
export OWL_SPARKLOG_ENABLE=false	Enabling deeper spark logs when toggled to true.
export LDAP_GROUP_RESULT_DN_ATTRIBUTE	The attribute to the full path of the group object, for example, **CN=OwlAppAdmin ,OU=OwlGroups,OU=Groups,DC=owl, DC=com**. Default is **distinguishedname**.
export LDAP_GROUP_RESULT_NAME_ATTRIBUTE	The attribute to the simple name of the group, for example, **OwlAppAdmin**. Default is **CN**.
export LDAP_GROUP_RESULT_CONTAINER_BASE	Property used in the scenario where the LDAP_GROUP_RESULT_DN_ ATTRIBUTE does not return a value. In this case, the LDAP_GROUP_RESULT_ NAME_ATTRIBUTE prepends to this value, which creates a fully qualified LDAP path. For example, **OU=OwlGroups ,OU=Groups,DC=owl,DC=com**. Default is **<null>**.

Configuration of owl.properties file

Example	Description
key=XXXXXX	The license key.
spring.data-source.url=jdbc:postgresql://localhost:5432/postgres	The connection string to the Collibra DQ metastore (used by owl-core).

Example	Description
spring.datasource.password=xxxxxx	The password to the Collibra DQ metastore (used by owl-core).
spring.datasource.username=xxxxxx	The username to the Collibra DQ metastore (used by owl-core).
spring.datasource.driver-class-name-e=com.owl.org.postgresql.Driver	Shaded Postgres driver class name.
spring.agent.datasource.url	jdb-c:postgresql://$host:$port/owltrunk
spring.agent.datasource.username	{user}
spring.agent.datasource.passwords	{password}
spring.agent.datasource.driver-class-name	org.postgresql.Driver

Starting Spark

Spark Standalone Mode - Spark 3.2.0 Documentation

Launch Scripts

To launch a Spark standalone cluster with the launch scripts, you should create a file called conf/workers in your Spark directory, which must contain the hostnames of all the machines where you intend to start Spark workers, one per line. If conf/workers does not exist, the launch scripts defaults to a single machine (localhost), which is useful for testing. Note, the master machine accesses each of the worker machines via ssh. By default, ssh is run in parallel and requires password-less (using a private key) access to be setup. If you do not have a password-less setup, you can set the environment variable SPARK_SSH_FOREGROUND and serially provide a password for each worker.

Once you've set up this file, you can launch or stop your cluster with the following shell scripts, based on Hadoop's deploy scripts, and available in `SPARK_HOME/sbin`:

Shell Script	Description
`sbin/start-master.sh`	Starts a master instance on the machine the script is executed on.
`sbin/start-workers.sh`	Starts a worker instance on each machine specified in the `conf/workers` file.
`sbin/start-worker.sh`	Starts a worker instance on the machine the script is executed on.
`sbin/start-all.sh`	Starts both a master and a number of workers as described above.
`sbin/stop-master.sh`	Stops the master that was started via the `sbin/start-master.sh` script.
`sbin/stop-worker.sh`	Stops all worker instances on the machine the script is executed on.
`sbin/stop-workers.sh`	Stops all worker instances on the machines specified in the `conf/workers` file.
`sbin/stop-all.sh`	Stops both the master and the workers as described above.

> Note These scripts must be executed on the machine you want to run the Spark master on, not your local machine.

```
### Starting Spark Standalone

cd /home/owldq/owl/spark/sbin
./start-all.sh

### Stopping Spark
cd /home/owldq/owl/spark/sbin
./stop-all.sh
```

```
### Starting Spark with Separate Workers

SPARK_WORKER_OPTS=" -Dspark.worker.cleanup.enabled=true -Dspark.-
worker.cleanup.interval=1799 -Dspark.-
worker.cleanup.appDataTtl=3600"

### 1 start master
/home/owldq/owl/spark/sbin/start-master.sh

### 2 start workers
SPARK_WORKER_INSTANCES=3;/home/owldq/owl/spark/sbin/start-
slave.sh spark://$(hostname):7077 -c 5 -m 20g
```

Standalone Install (Script)

Requirements

Resource	Notes	Provided by
OS	Red Hat 7/Red Hat 8, or CentOS 7/CentOS 8 Note For RHEL 8 or CentOS 8, Postgres is automatically installed when you run the script.	Customer
Memory	16 GB Ram	
Cores	8	
Storage	50 GB Disk	
Permission	`sudo`	
Install Script	Download using the curl command below	Collibra

Access the machine through either a cloud shell or SSH. The snippet below is an example SSH command.

```
ssh -i <your_key> <user>@<public-ipv4-ip-address>
```

```
ssh -i ~/.ssh/abc.pem centos@22.000.111.3333
```

Your pem file should have correct permissions. Confirm you have the correct permissions with the command: `sudo chmod 400 <your_key>`

Steps

After you enter your SSH into the Centos or Redhat VM, run these commands from the command line:

Step 1: Download the script

```
curl -o cdq_install.sh https://owl-pack-
ages.s3.amazonaws.com/MP/cdq_install.sh
```

Step 2: Modify script permission

```
sudo chmod +x cdq_install.sh
```

Step 3: Run the script

```
echo | ./cdq_install.sh
```

> Note The most common directory to use for installation is your user directory (/home/<user>). You do not need to create any directories because the install script creates the correct directory structure. You should have `sudo` access to perform the installation and the directory should not be a restricted system directory.

> Note If you run the setup script in a non-interactive mode, such as an automated deployment, you must export the following variables for the script to run correctly:
>
> ```
> export DQ_ADMIN_USER_PASSWORD
> export DQ_ADMIN_USER_EMAIL
> ```

In this step, you are prompted to enter a default admin user email and password associated with the default admin user account.

The password must adhere to the following password policy:

- A minimum of 8 characters.
- A maximum of 72 characters.
- At least one upper-case character.
- At least one numeric character.
- At least one supported special character (`!@#%$^&*?_~`).
- Cannot contain the user ID (admin).

> Warning As part of Collibra's password policy, you cannot proceed with installation until you enter a valid password.

Step 4: Click the URL

Sign in to the application with the username `admin` and the default admin password you provided in the setup script.

```
http://<server_name/ip>:9000
```

> Note Make sure you have access to the server and port. Adding the correct security group or whitelisting your IP address is a common step to be able to access an application running on a cloud server.

BYOL (Optional)

Using the same command, you can bring your own license or use a different download link.

```
./cdq_install.sh "<Installer Download Link>" "<License Key>"
```

Confirmation (Optional)

Once the installation script is complete, check the details of the processes on the server.

```
ps -ef | grep -i spark
ps -ef | grep -i owl-web
ps -ef | grep -i owl-agent
ps -ef | grep -i postgres
```

When all of the processes are up and running, sign in to your standalone instance of Collibra Data Quality & Observability.

Configure your Agent and Adding Connections as normal.

Troubleshooting

Common error messages

If you receive an error with the message, "Application already running on port 8080," enter the following command:

```
sudo netstat -plten |grep java
```

You can then use a `kill` command to kill the process.

```
kill -9 <appId>
```

> Note If you receive a "permission denied" error message, make sure that you are using `sudo`.

Known Limitations

The dollar sign `$` is not a supported character for passwords created via Helm during the admin password creation process in Step 3. Because passwords require at least one special character, use one of the following supported characters instead: `!@#%^&*?_~`.

Getting Started with AWS

Prerequisites

To get started, you need a Collibra Data Quality account.

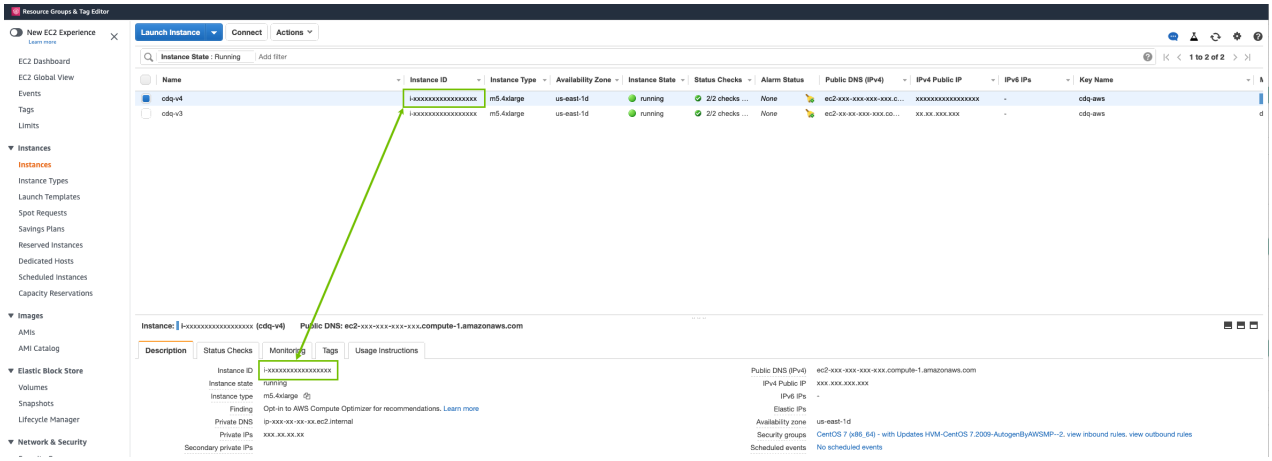
Connecting to Collibra Data Quality

1. To connect to your environment, enter your URL.

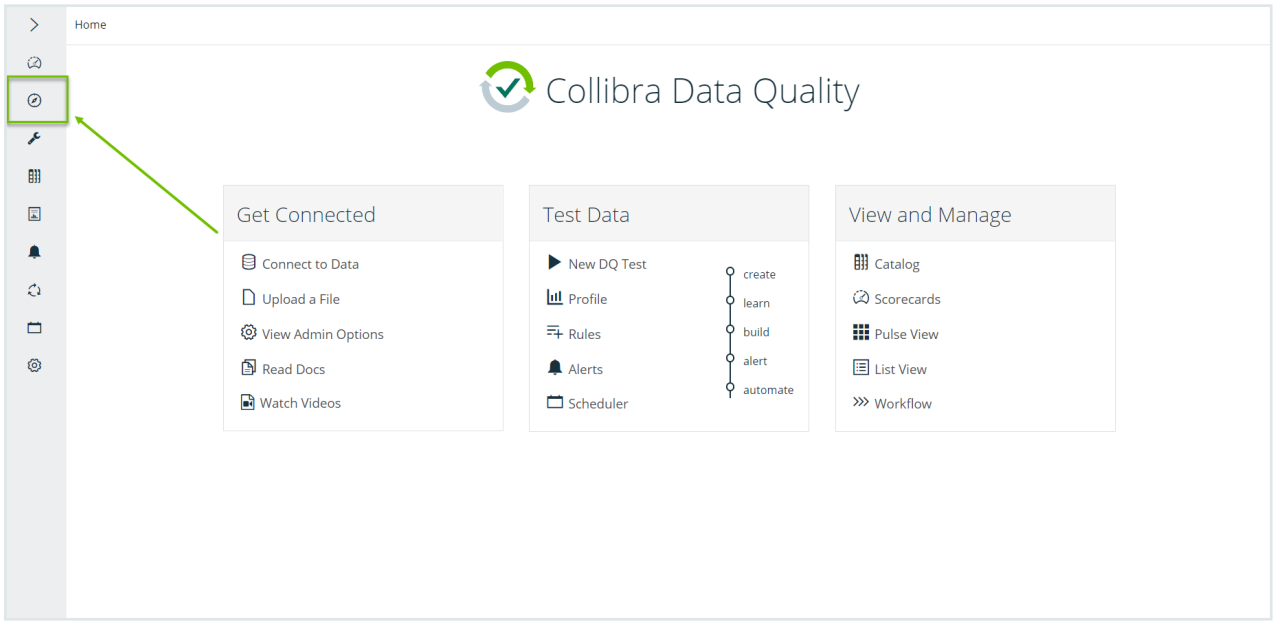
Example: https://<your_host_name>:9000

2. Enter "admin" as your **Username** and your unique Instance ID as your **Password**.

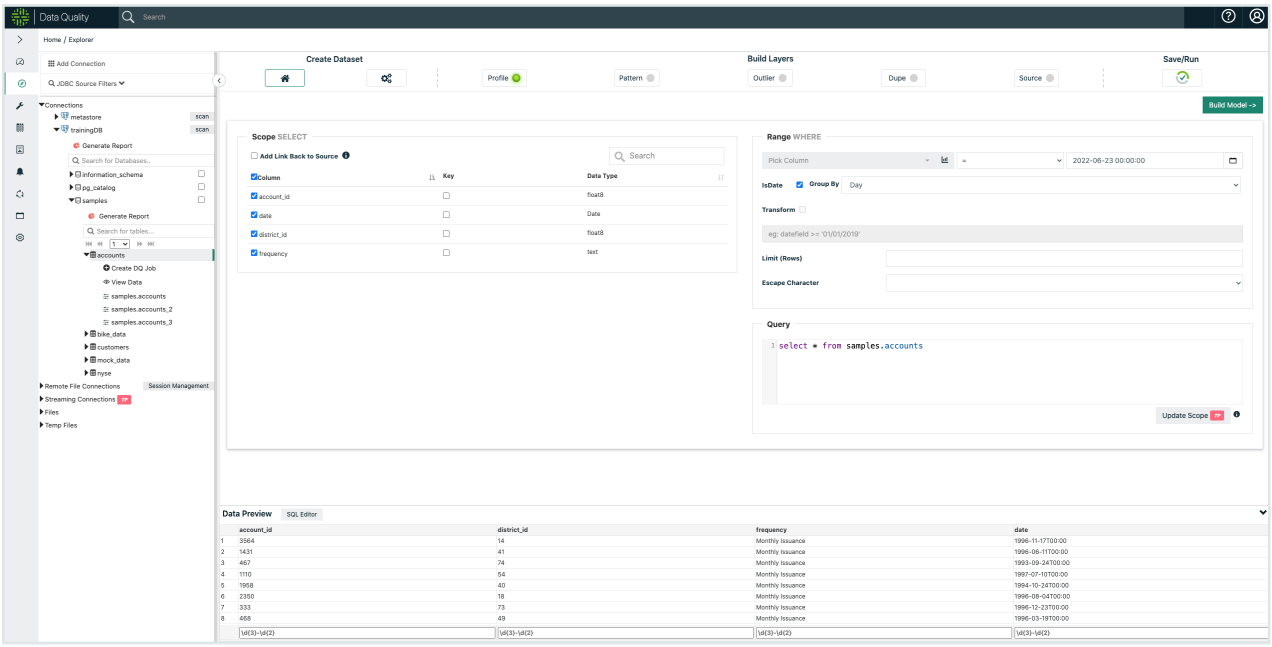
> Note Username: admin Password: CDQ<your_instance_id>



3. From the home page, click the **Explorer** tab.



4. From Explorer, select a connection from the **Connections** dropdown to access your data set.



What's next?

- Visit the Explorer (no-code) for more information on quickly connecting to your data sets.
- Visit Collibra Data Quality's Youtube channel for more tutorials.
- Register for Collibra Data Quality's Product Showcase to learn more about Collibra Data Quality.

Standalone Install (AWS CloudFormation)

This section describes how to install and configure Collibra Data Quality using the AWS CloudFormation stack.

Prerequisites

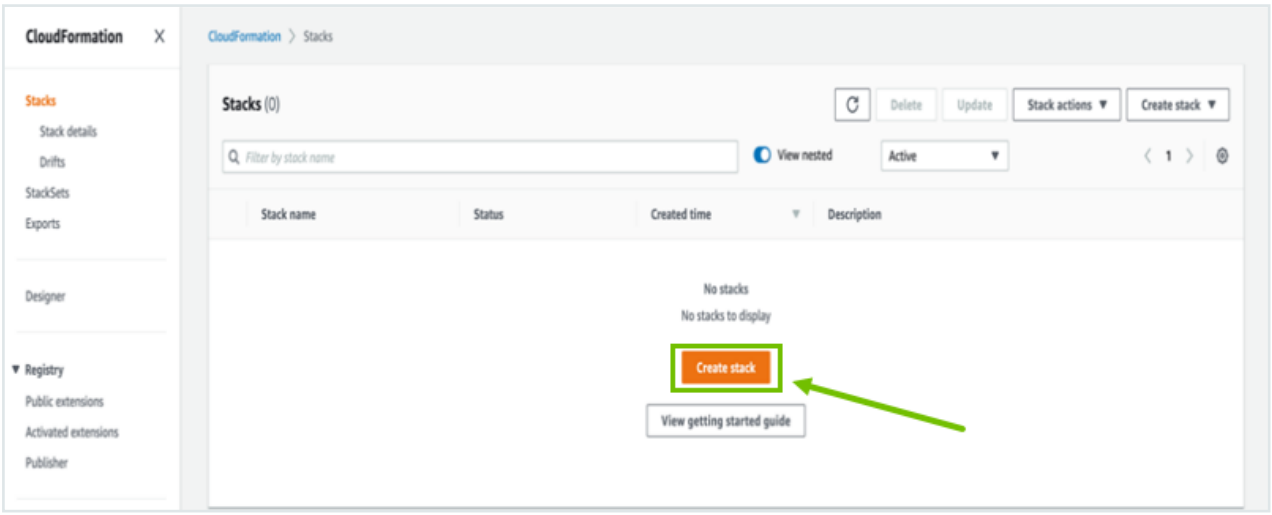
To install Collibra Data Quality using the AWS CloudFormation stack, you need an AWS user account with permissions to provision AWS resources.

Steps

1. Login to your AWS account and navigate to **CloudFormation** from the search bar.

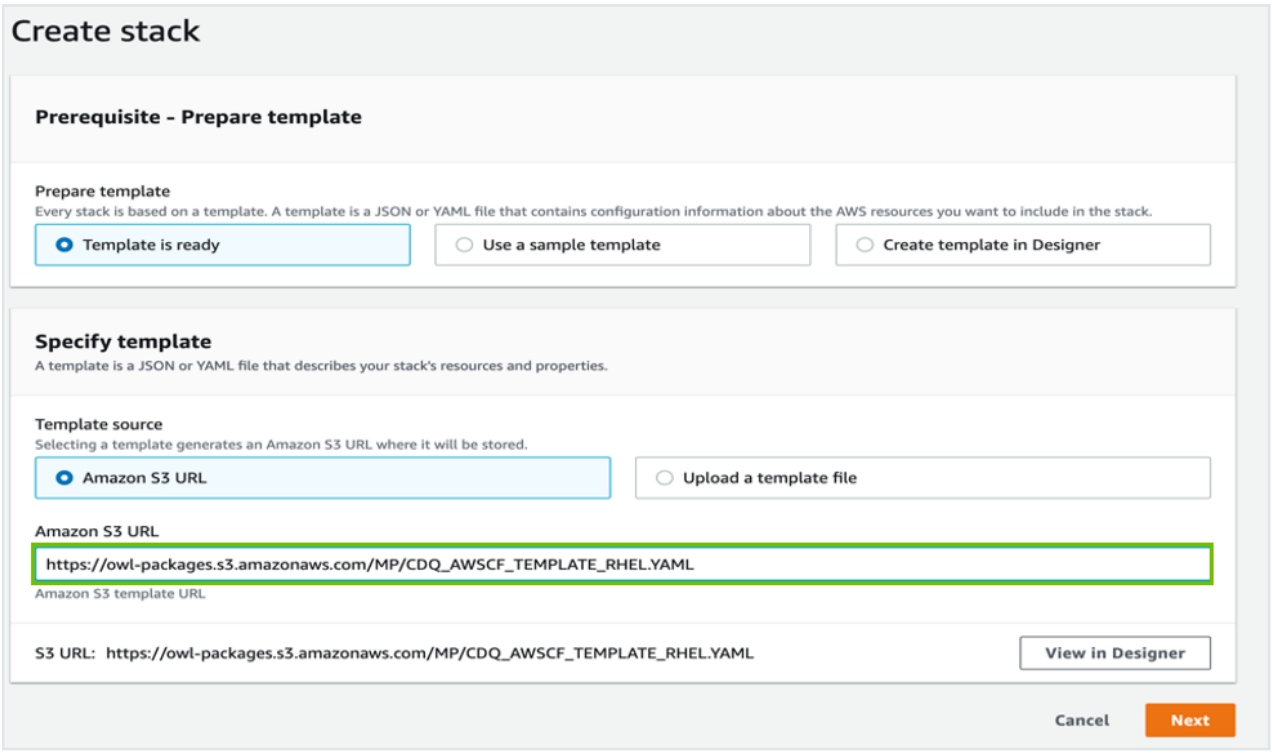
2. In CloudFormation, click the **Create stack** button and provide the S3 location for the template.

```
                <MadCap:concept term="Data Quality" xmlns:MadCap-
p="http://www.madcapsoftware.com/Schemas/MadCap.xsd" />ht-
tps://owl-packages.s3.amazonaws.com/MP/CDQ_AWSCF_TEMPLATE_
RHEL.YAML
```



3. Click **Next**.

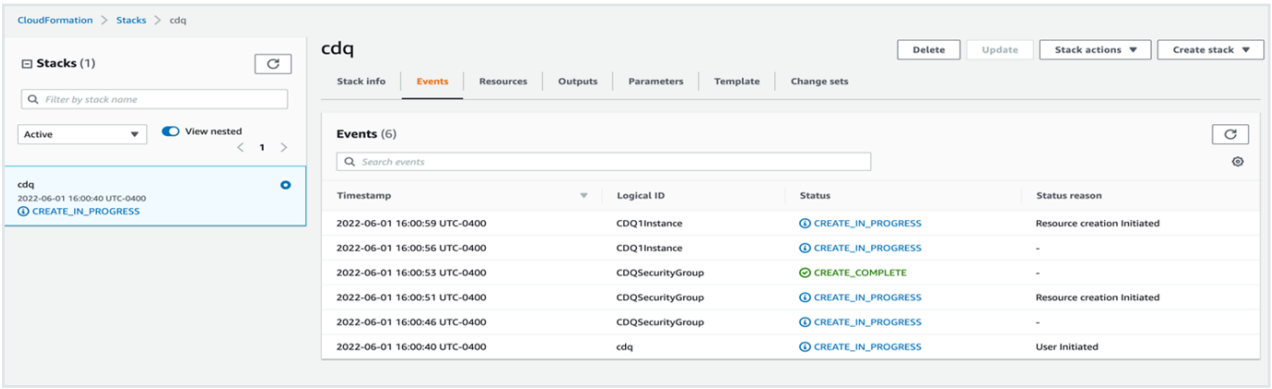
4. Follow the prompts and select the appropriate instance type, VPC, subnet, and key values based on your AWS account. Rest accepts everything by default.



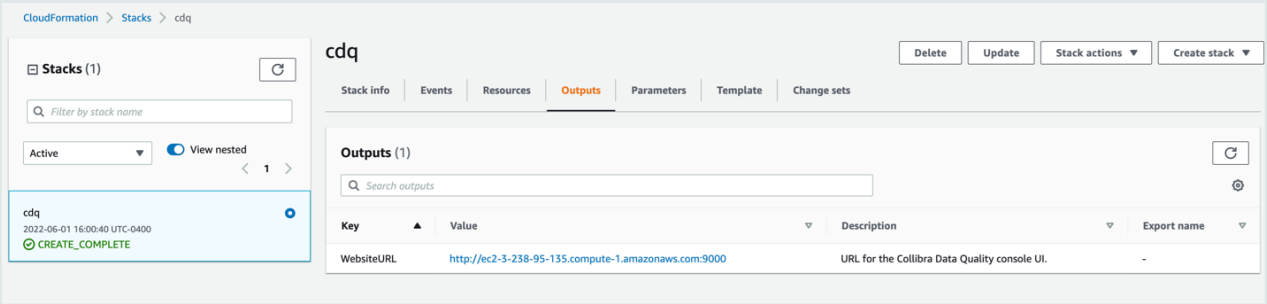
5. Click **Next**.

> Note This process takes 10-15 minutes to spin up the EC2 instance and deploy Collibra DQ.

6. In the **Events** tab, you can monitor status of your stack.



7. Once your stack is created, click the **Outputs** tab to access the CDQ Login URL.



8. In the **Value** column, click the CDQ Login URL.

> Note Your CDQ instance comes with sample data and prewired DB connections.

9. To uninstall and release the resource, delete the stack.

Standalone Install (Google Cloud Platform)

This section provides information on how to deploy Collibra Data Quality on Google Cloud Platform.

Prerequisites

You have:

- A Google user account.
- A project for Google Cloud Platform to deploy Collibra Data Quality.
- Deployment Manager permissions provisioned by an admin.

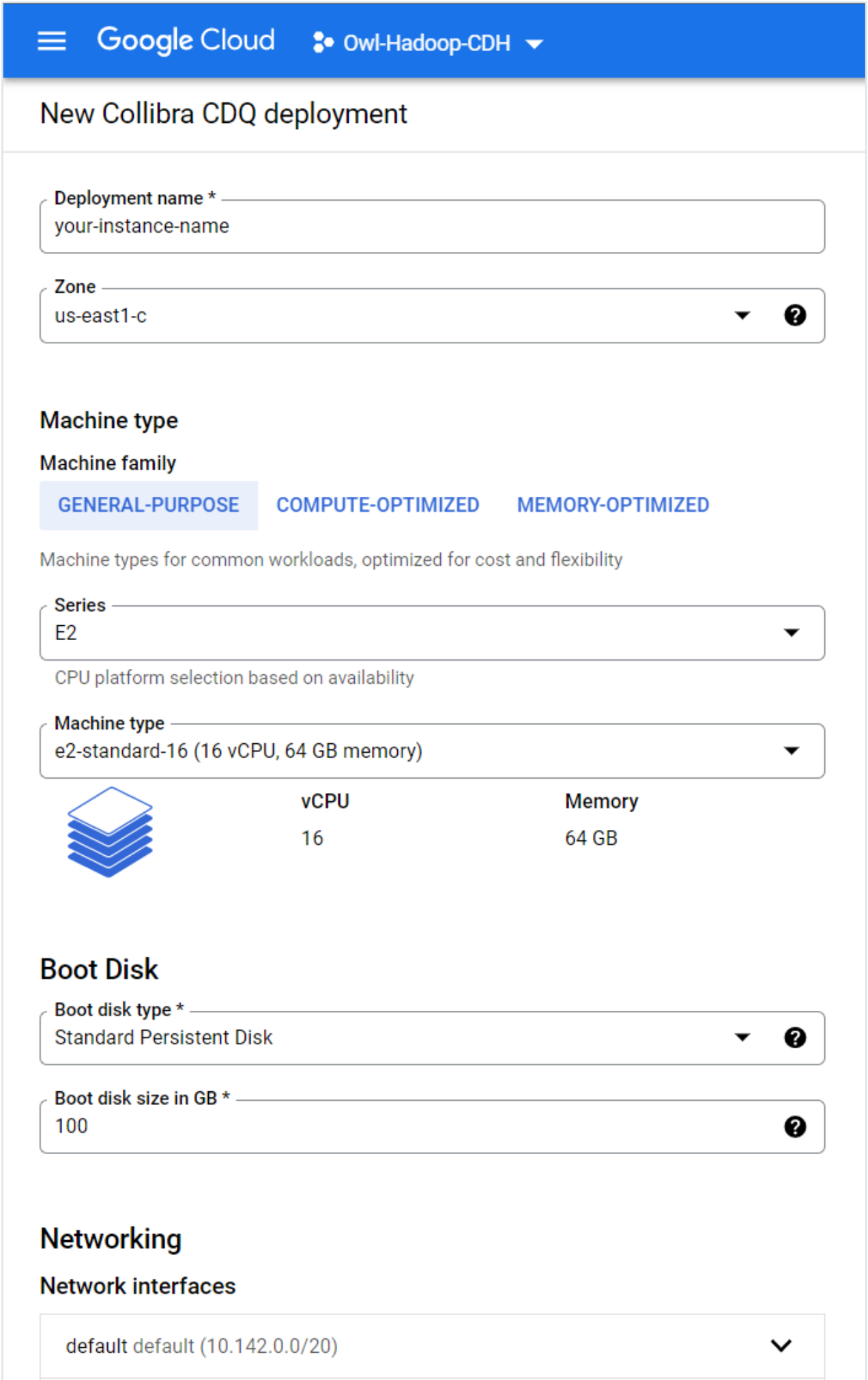
Steps

1. Step 1: Create a new Collibra Data Quality deployment
2. Step 2: Launch a new Collibra Data Quality deployment

Step 1: Create a new Collibra Data Quality deployment

1. Sign in to your Google Cloud Marketplace account and choose your **working project** for Collibra Data Quality deployment.

2. In the search bar, search for ***Collibra*** and press enter.
   >> The search results appear.
3. Select **Collibra CDQ**.
   >> The Collibra CDQ product page opens.
4. Select **Launch**.
   >> The New Collibra CDQ deployment page opens.
5. On the **New Collibra CDQ deployment** page, specify the following information:


Google Cloud
Owl-Hadoop-CDH
New Collibra CDQ deployment
Deployment name *
your-instance-name
Zone
us-east1-c
Machine type
Machine family
GENERAL-PURPOSE
COMPUTE-OPTIMIZED
MEMORY-OPTIMIZED
Machine types for common workloads, optimized for cost and flexibility
Series
E2
CPU platform selection based on availability
Machine type
e2-standard-16 (16 vCPU, 64 GB memory)
vCPU
16
Memory
64 GB
Boot Disk
Boot disk type *
Standard Persistent Disk
Boot disk size in GB *
100
Networking
Network interfaces
default default (10.142.0.0/20)

Field	Description
Deployment name	The name of your Collibra Data Quality deployment.
Zone	Select the zone closest to your region.
Series	The default is E2.
Machine type	The default is e2-standard-16 (16 vCPU, 64 GB memory).
Boot disk type	The default is Standard Persistent Disk.
Boot disk size in GB	The default is 100.

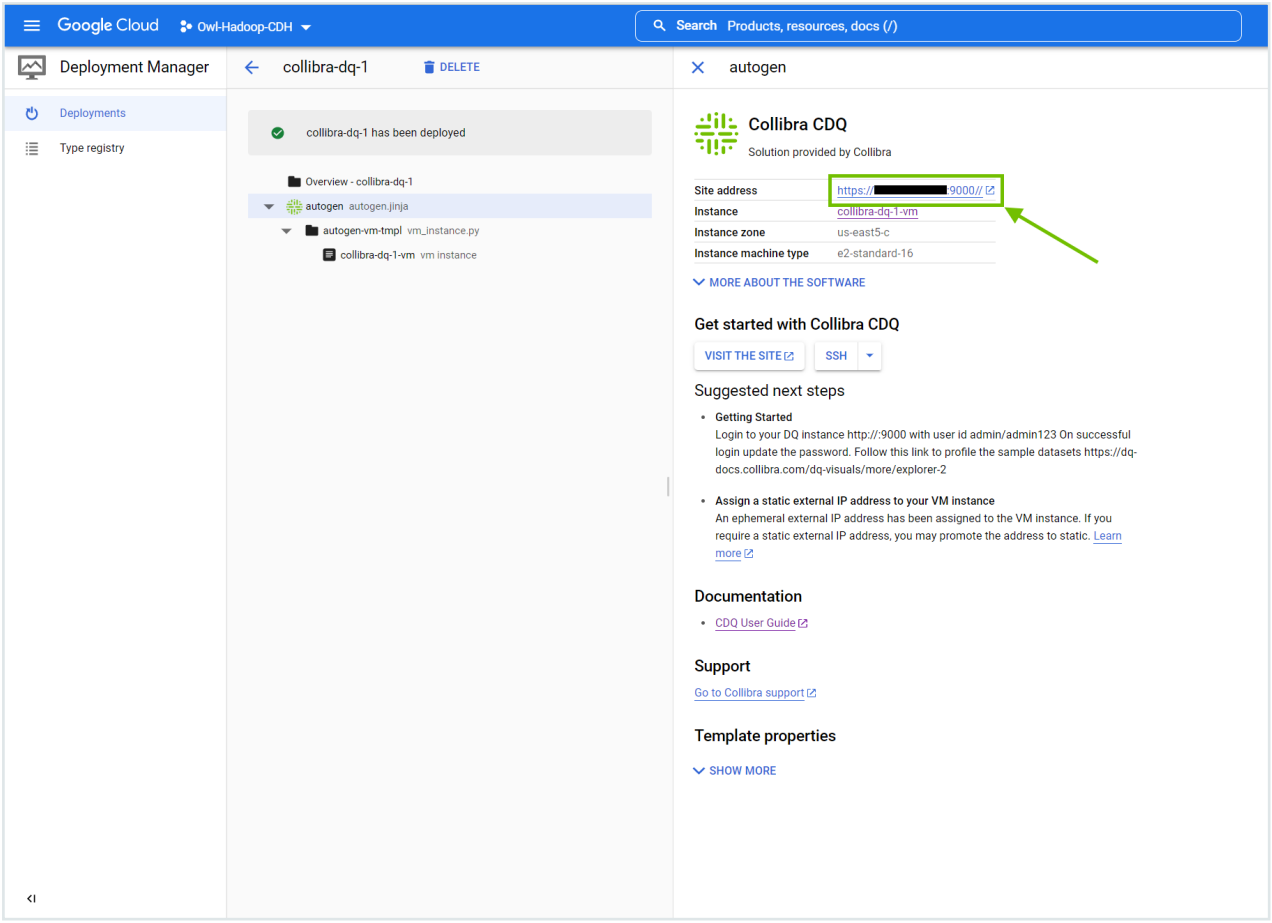
6. Read and accept the **Terms of Service**.

7. Select **Deploy**.
>> The **Deployment Manager** page opens.

> Note GCP also sends you a confirmation email containing a direct link to the Deployment Manager Page.

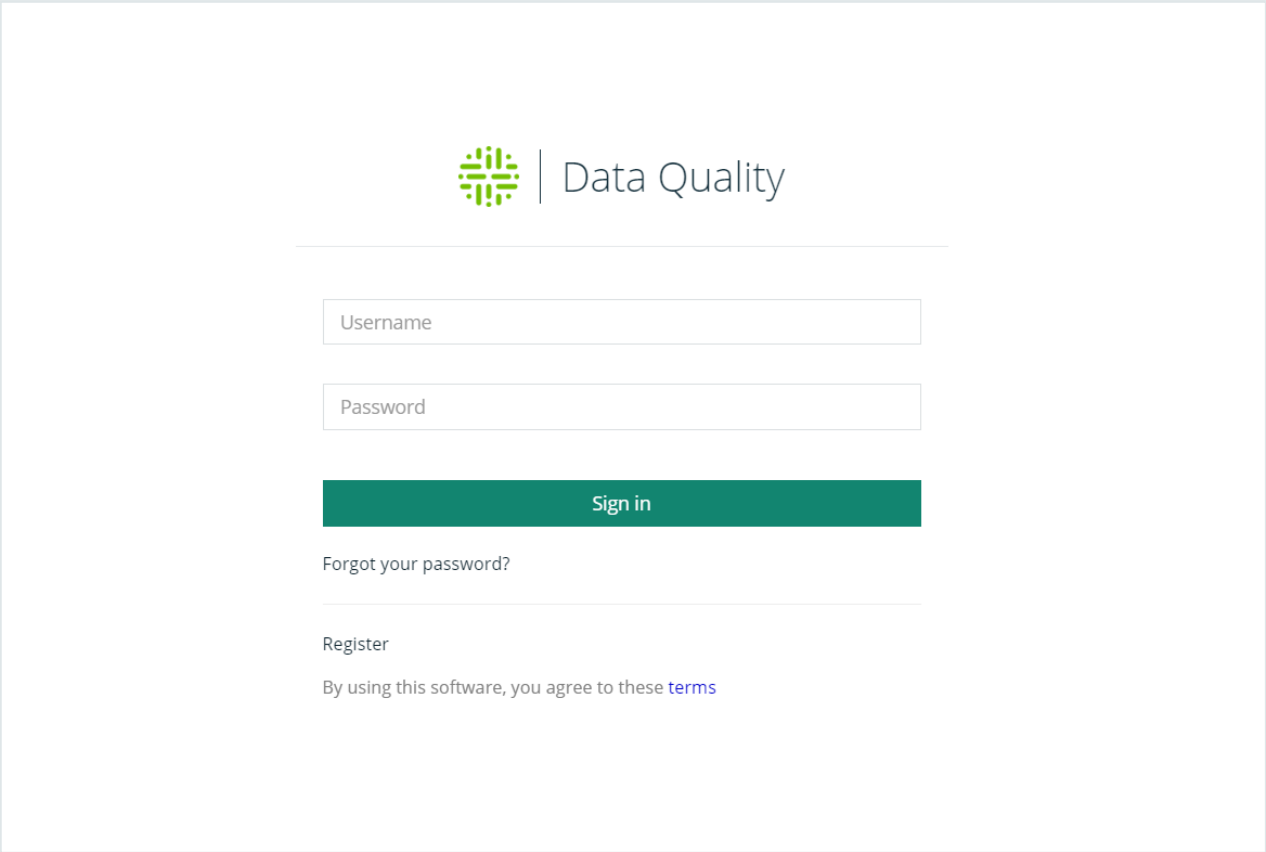
Step 2: Launch a new Collibra Data Quality deployment

1. From the **Deployment Manager**, select the **site address** to sign in to your Collibra Data Quality instance.



2. Sign in to your instance using the following one-time username and password: `admin / admin123` >> The Collibra Data Quality landing page opens.

> Note You must change your password after successfully signing in for the first time. Select the avatar in the upper right of your screen and select the Change Password tab and follow the prompts to change your password.



Troubleshooting your deployment

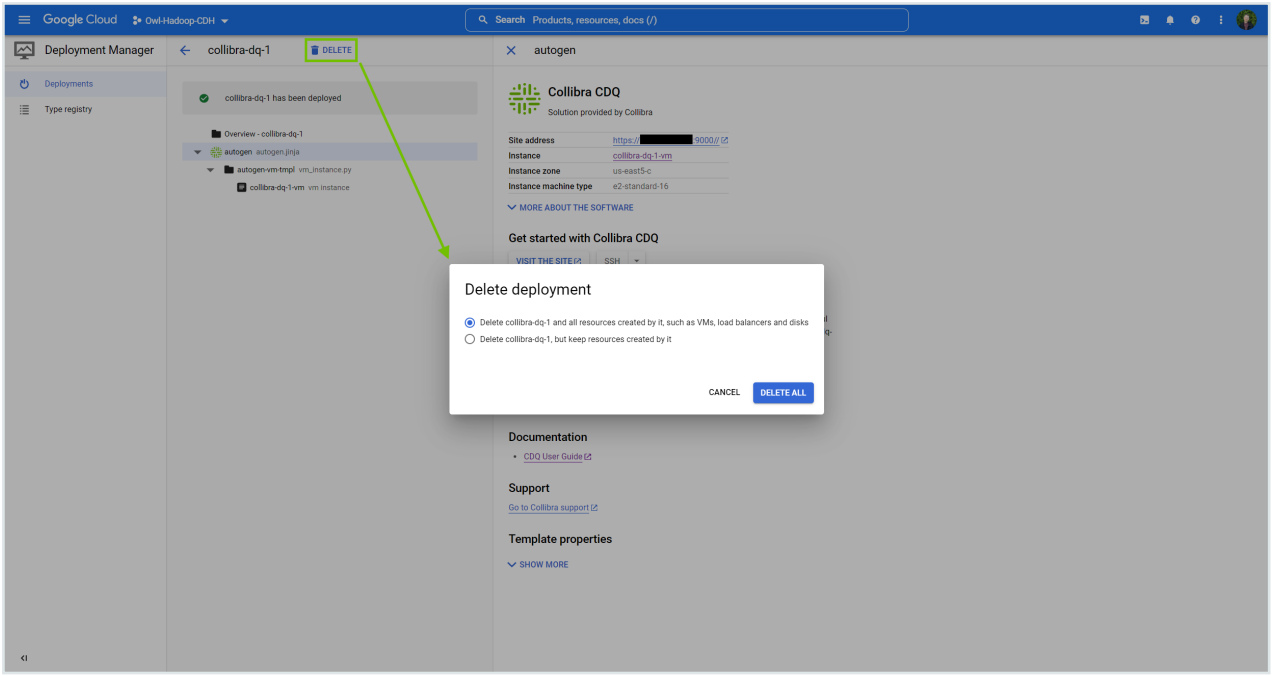
After a successful deployment of Collibra Data Quality on GCP, it is possible that you receive an error message when you select your site address. If this happens, you can:

- Check your network access and verify that you have the appropriate network tags.
- Check that the firewall entry is properly defined for your install.
- Check the URL and remove the **s** from http**s**. Also remove the second trailing forward slash / after :9000. Correct: http://<your.instance>:9000/

Deleting your deployment

To delete your deployment, go to the **Deployment Manager**. Select **Delete** at the top of the screen and then from the dialog box, choose to either

- Delete your deployment and all of its resources.
- Delete your deployment but keep its resources.



When you select a deletion method, your deployment is permanently removed from the list of deployments on the Deployment Manager page.

What's next?

- Visit the Explorer (no-code) for more information on quickly connecting to your data sets.
- Visit Collibra Data Quality's Youtube channel for more tutorials.

Standalone Upgrade

> Note Before proceeding with any upgrades, back up your Collibra DQ Metastore.

> Warning Rolling back Collibra DQ to a prior version is not supported. Please contact Collibra Support with any questions.

Download the Collibra DQ Upgrade Package

> Note As of the December 2021 release, all Collibra DQ customers upgrading or patching will receive the Full package (vs. the Base package) and should follow the same upgrade steps as below.

Download the tarball using the signed link to the full package tarball provided by Collibra. Replace `<signed-link-to-full-package>` with the link provided.

```
### Go to the OWL_BASE (home directory of the user is most com-
mon)
### This example we will use /home/owldq installing as the user
owldq

cd /home/owldq

### Download & untar
curl -o dq-full-package.tar.gz "<signed-link-to-full-package>"
tar -xvf dq-full-package.tar.gz

### Clean-up unnecessary tarball (optional)
rm dq-full-package.tar.gz
```

Upgrade Steps

1. Copy the contents of the provided package, for example, owl-<newversion>-<SPARK301>-package-full.tar.gz to the system being upgraded (extract contents).
    - Best practice: Untar the contents into a uniquely named folder, for example, 2022-10-dq-upgrade.
2. Stop the Collibra DQ Web process with the following commands:
    a. `cd /owlhome/owl/bin`
    b. `./owlmanage.sh stop=owlweb`
3. Stop the Collibra DQ Agent process with the following commands:
    a. `cd /owlhome/owl/bin`
    b. `./owlmanage.sh stop=owlagent`
4. Move the old jars from `owl/bin` with the following commands:
    a. `mv owl-webapp-<oldversion>-<spark301>.jar /tmp`
    b. `mv owl-agent-<oldversion>-<spark301>.jar /tmp`

   c. `mv owl-core-<oldversion>-<spark301>.jar /tmp`
5. Copy the new jars into the owl/bin folder from the extracted package with the following commands:
   a. `mv owl-webapp-<newversion>-<spark301>.jar /home/owldq/owl/bin`
   b. `mv owl-agent-<newversion>-<spark301>.jar /home/owldq/owl/bin`
   c. `mv owl-core-<newversion>-<spark301>.jar /home/owldq/owl/bin`
6. Start the Collibra DQ Web application with the following command:
   - `./owlmanage.sh start=owlweb`
7. Start the Collibra DQ Agent with the following command:
   - `./owlmanage.sh start=owlagent`
8. Validate the number of active services with the following command:
   - `ps -ef | grep owl`

Additional Notes / Steps Due To Log4J (December 2021)

Additional Step 1: Place Log4j-1.2-api-2.17.1.jar (as of 2022.02) into /<install-home>/owl/spark/jars.

> Note Was Log4j-1.2-api-2.17.0.jar in 2021.12 and 2022.01.

Who: All Collibra DQ customers, particularly those leveraging CLI mode.

1. Go to the same folder where the Collibra-provided upgrade package was extracted.
2. Go to <location of 2022-02-dq-upgrade>/packages/install-packages.
3. Extract the needed log4j-1.2-api-2.17.1.jar with the following command: `tar -xvf spark-extras.tar.gz spark-extras/log4j-1.2-api-2.17.1.-jar`.
4. Move the log4j-1.2-api-2.17.1.jar file into /<install-path>/spark/jars folder.

FAQ

Q: (When) do I need to move Log4j-1.2-api-2.17.1.jar before or after swapping the main Collibra DQ jars?

- A: The sequence does not matter.

Q: (What) if I don't follow these additional upgrade steps?

- A: If your `SPARK_SUBMIT_MODE` within owl-env.sh is set to `SPARK_SUBMIT_MODE-E=native`, Collibra DQ will function properly without the above additional upgrade step, with the exception of CLI mode.

Additional Step 2: Remove a legacy properties file.

Who: Only Collibra DQ customers upgrading Agents installed on Cloudera CDP Hadoop Edge Nodes.

1. Go to /<agenthome>/owl/config/.
2. Remove the `log4j-cluster.properties` file.

FAQ

Q: (When) do I need to remove log4j-cluster.properties before or after swapping the main Collibra DQ jars?

- A: Remove the file before restarting owl-agent. Otherwise, stop owl-agent again, remove the file, then restart owl-agent.

Q: (What) if I don't follow these additional steps?

- A:If you use agents on Hadoop edge nodes, you will receive errors when running DQ Jobs as a result of engaging a method that no longer exists.

Q: What should I do if I am not a vendor-supported Cloudera CDP version?

- A: Our testing and guidance mainly applies to vendor-supported (non-EOL) Cloudera CDP versions. Other Hadoop variants may handle logging differently and may require the legacy properties file. In short, feel free to first upgrade without this step, then remove the log4j-cluster.properties file if DQ Jobs are running into issues.

Upgrading data source drivers

When new data source drivers are available, they are listed in the Release Notes or recommended to you directly by Collibra. Determine which drivers need to be updated and follow these steps:

1. Confirm with Collibra Support which drivers need to be updated.
2. From the previously extracted tarball provided to you by Collibra, locate the driver-s.tar.gz file and extract the contents into a new directory called "drivers".
3. Replace the drivers:
    - Replace OWL_BASE/owl/drivers/<old-driver> with the new drivers extracted from the tarball OWL_BASE/owl/drivers/<new-driver>.
      For example, if you replace an old Databricks driver with a new one, the file path might look like OWL_BASE/owl/drivers/databricks.

Standalone Sizing

Small Tier - 16 Core, 128G RAM (r5.4xlarge / E16s v3)

Component	RAM	Cores
Web	2g	2
Postgres	2g	2
Spark	100g	10
Overhead	10g	2

Medium Tier - 32 Core, 256G RAM (r5.8xlarge / E32s v3)

Component	RAM	Cores
Web	2g	2
Postgres	2g	2
Spark	250g	26
Overhead	10g	2

Large Tier - 64 Core, 512G RAM (r5.16xlarge / E64s v3)

Component	RAM	Cores
Web	4g	3
Postgres	4g	3
Spark	486g	54
Overhead	18g	4

Estimates

Sizing should allow headroom and based on peak concurrency and peak volume requirements. If concurrency is not a requirement, you just need to size for peak volume (largest tables). Best practice to efficiently scan is to scope the job by selecting critical columns. See Performance Tuning for more information.

Bytes per Cell	Rows	Columns	Gigabytes	Gigabytes for Spark (3x)
16	1,000,000.00	25	0.4	1.2
16	10,000,000.00	25	4	12
16	100,000,000.00	25	40	120
16	1,000,000.00	50	0.8	2.4
16	10,000,000.00	50	8	24
16	100,000,000.00	50	80	240
16	1,000,000.00	100	1.6	4.8
16	10,000,000.00	100	16	48
16	1,000,000,000.00	100	1600	4800
16	100,000,000.00	100	160	480

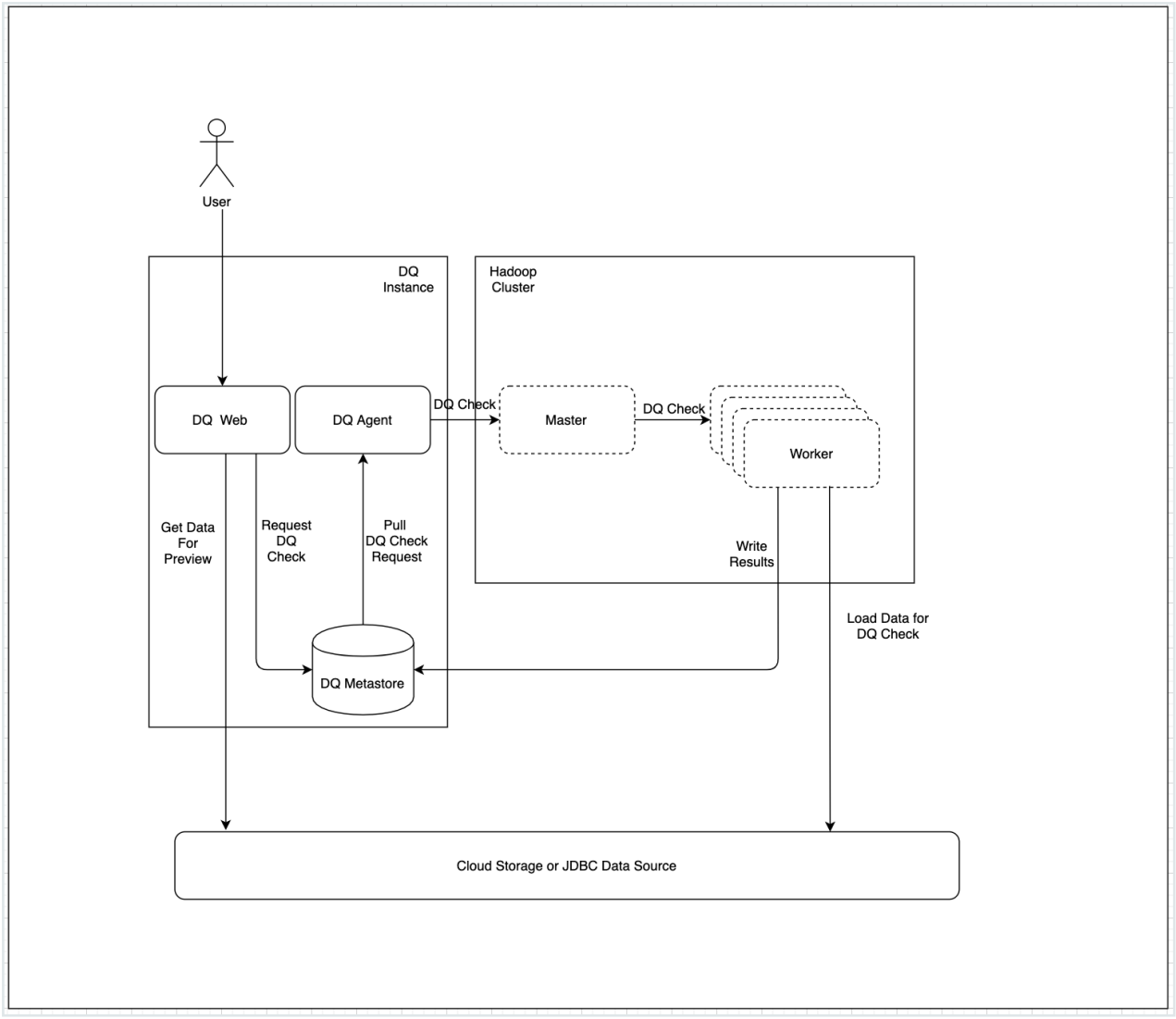
Bytes per Cell	Rows	Columns	Gigabytes	Gigabytes for Spark (3x)
16	1,000,000.00	200	3.2	9.6
16	10,000,000.00	200	32	96
16	100,000,000.00	200	320	960
16	1,000,000,000.00	200	3200	9600

Cluster

If your program requires more horsepower or (spark) workers than the example tiers above which is fairly common in Fortune 500 companies then you should consider the horizontal and ephemeral scale of a cluster. Common examples are Amazon EMR, Cloudera CDP, etc. Collibra DQ is built to scale up horizontally and can scale to hundreds of nodes.

Hadoop

For large scale processing and concurrency, a single vertically scaled Spark server is not enough. To address large scale processing, DQ has the ability to push compute to an external Hadoop cluster. This page describes the process by which the DQ Agent can be configured to push DQ jobs to Hadoop.



Hadoop Install

> Note In some cases, the required Hadoop client configuration requires the DQ Agent to run on a Hadoop Edge node within the cluster. This can happen for a number of reasons. Namely, native dependency packages are required, network isolation from the subnet that hosts the DQ server, complex security configuration, and so on. In these circumstances, you can deploy the DQ Agent on a cluster Edge Node that contains the required configurations and packages. In this setup, the DQ Agent uses the existing Hadoop configuration and packages to run DQ checks on the Hadoop cluster.

Hadoop Config Setup

Hadoop configuration can be incredibly complex. There can be hundreds of "knobs" across dozens of different components. However, DQ's goal is to simply leverage Hadoop to allocate compute resources in order to execute DQ checks (Spark jobs). This means that the only client side configurations required are:

- Security protocol definition
- Yarn Resource Manager endpoints
- Storage service (HDFS or Cloud storage).

Once the Hadoop client configuration is defined, it is only a matter of pointing the DQ Agent at the folder that contains the client configuration files. The DQ Agent is then able to use the Hadoop client configuration to submit jobs to the specified Hadoop cluster.

> Note DQ jobs running on Hadoop are Spark jobs. DQ will use the storage platform defined in the "fs.defaultFS" setting to distribute all of the required Spark libraries and specified dependency packages like drivers files. This allows DQ to use a version of Spark that is different than the one provided by the cluster. If it is a requirement to use the Spark version provided by the target Hadoop cluster, obtain and use a copy of the yarn-site.xml and core-site.xml from the cluster.

Create the Config Folder

```
cd $OWL_HOME
mkdir -p config/hadoop
echo "export HADOOP_CONF_DIR=$OWL_HOME/config/hadoop" >> con-
fig/owl-env.sh
bin/owlmanage.sh restart=owlagent
```

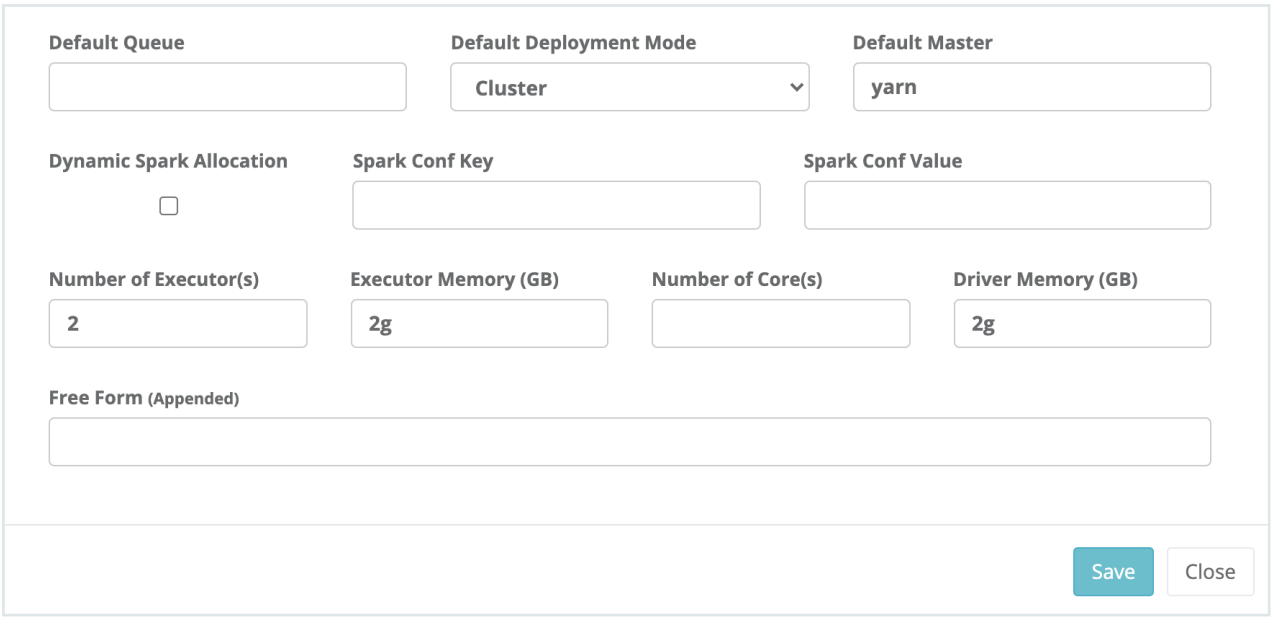
Minimum Config with Kerberos and TLS Disabled

This configuration would typical only be applicable in Cloud Hadoop scenarios (EMR/Dataproc/HDI). Cloud Hadoop clusters are ephemeral and do not store any data as the data is stored in and is secured by Cloud Storage.

```
export RESOURCE_MANAGER=<yarn-resoruce-manager-host>
export NAME_NODE=<namenode>

echo "
<configuration>
  <property>
    <name>hadoop.security.authentication</name>
    <value>simple</value>
  </property>
  <property>
    <name>hadoop.rpc.protection</name>
    <value>authentication</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://$NAME_NODE:8020</value>
  </property>
</configuration>
" >> $OWL_HOME/config/hadoop/core-site.xml

echo "
<configuration>
  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>$RESOURCE_MANAGER:8030</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>$RESOURCE_MANAGER:8032</value>
  </property>
  <property>
    <name>yarn.resourcemanager.webapp.address</name>
    <value>$RESOURCE_MANAGER:8088</value>
  </property>
</configuration>
" >> $OWL_HOME/config/hadoop/yarn-site.xml
```

Note When deploying a Cloud Service Hadoop cluster from any of the major Cloud platforms, it is possible to use Cloud Storage rather than HDFS for dependency package staging and distribution. To achieve this, create a new storage bucket and ensure that both the Hadoop cluster and the instance running DQ Agent have access to it. This is usually accomplished using a Role that is attached to the infrastructure. For example, AWS Instance Role with bucket access policies. Then, set "fs.defaultFS" in core-site.xml to the bucket path instead of HDFS.

Once the Hadoop client configuration is created, go to Agent Management console from the Admin Console and configure the agent to use Yarn (Hadoop resource scheduler) as the Default Master and set the Default Deployment Mode to "Cluster".

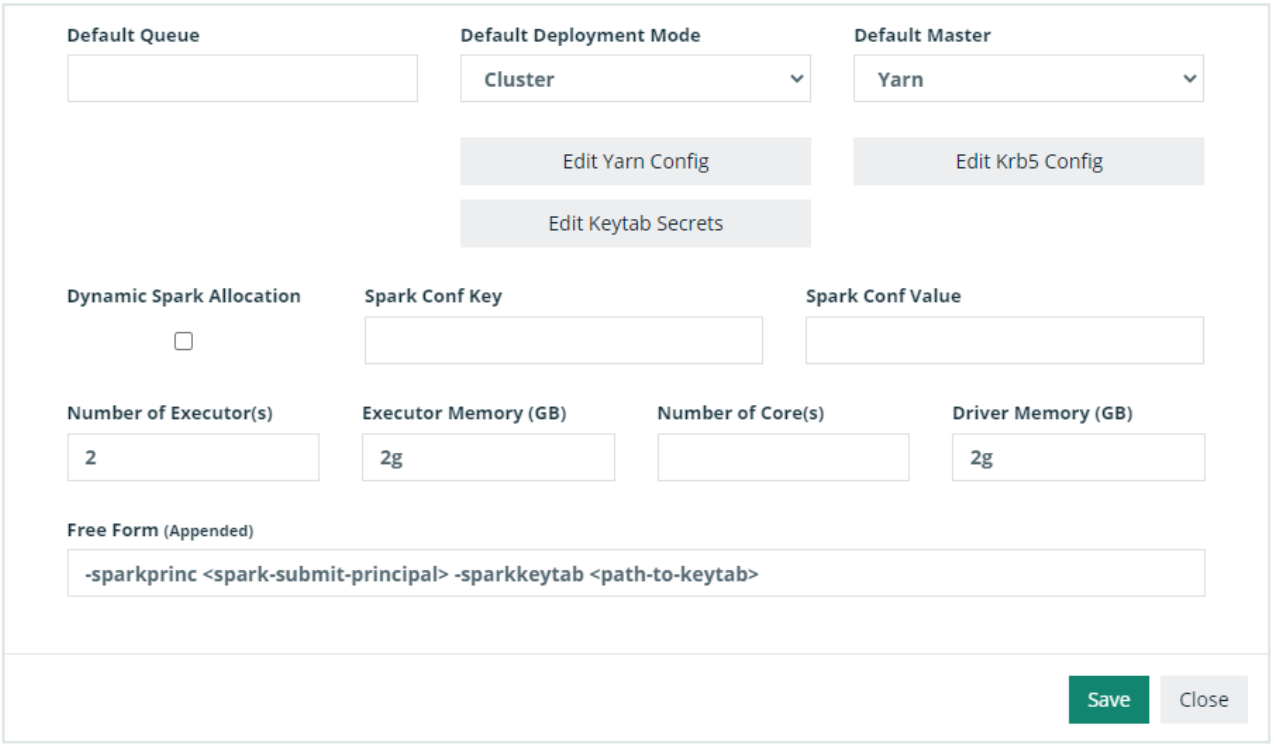


Kerberos Secured with Resource Manager TLS enabled

Typically, Hadoop clusters that are deployed on-premises are multi-tenant and not ephemeral. This means they must be secured using Kerberos. Additionally, all endpoints with HTTP endpoints will have TLS enabled. HDFS may also be configured for a more secure communication with additional RPC encryption.

```
export RESOURCE_MANAGER=<yarn-resoruce-manager-host>
export NAME_NODE=<namenode>
export KERBEROS_DOMAIN=<kerberos-domain-on-cluster>
export HDFS_RPC_PROTECTION=<authentication || privacy || integ-
rity>

echo "
<configuration>
  <property>
    <name>hadoop.security.authentication</name>
    <value>kerberos</value>
  </property>
  <property>
    <name>hadoop.rpc.protection</name>
    <value>$HDFS_RPC_PROTECTION</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://$NAME_NODE:8020</value>
  </property>
</configuration>
" >> $OWL_HOME/config/hadoop/core-site.xml

echo "
<configuration>
  <property>
    <name>hadoop.security.authentication</name>
    <value>HDFS/_HOST@$KERBEROS_DOMAIN</value>
  </property>
</configuration>
" >> $OWL_HOME/config/hadoop/hdfs-site.xml

echo "
<configuration>
  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value>$RESOURCE_MANAGER:8030</value>
  </property>
  <property>
    <name>yarn.resourcemanager.address</name>
    <value>$RESOURCE_MANAGER:8032</value>
  </property>
  <property>
    <name>yarn.resourcemanager.webapp.https.address</name>
    <value>$RESOURCE_MANAGER:8090</value>
  </property>
</configuration>
" >> $OWL_HOME/config/hadoop/yarn-site.xml
```

When the target Hadoop cluster is secured by Kerberos, DQ checks require a Kerberos credential. This typically means that the DQ Agent will need to be configured to include a Kerberos keytab with each DQ check. Access the DQ Agent configuration page from the Admin Console and configure the "Freeform Append" setting with the -sparkprinc <spark-submit-principal> -sparkkeytab <path-to-keytab>.



EMR / Dataproc / HDI

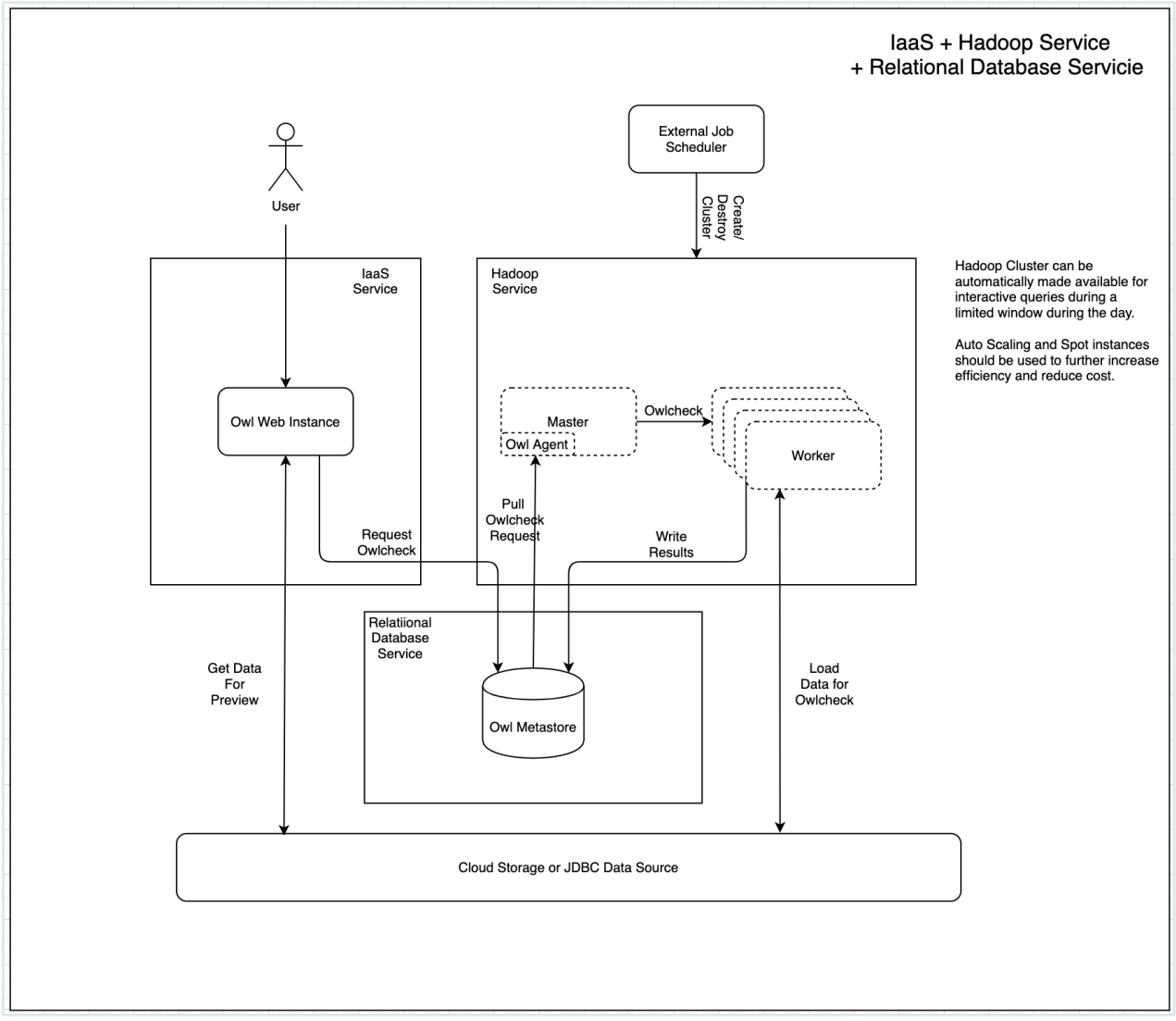
Running Apache Spark on Kubernetes differs from running this on virtual machine-based Hadoop clusters, which is the current mechanism provided by the existing CloudProc Dataproc service or competitive offerings like Amazon Web Services (AWS) Elastic MapReduce (EMR) and Microsoft's Azure HDInsight (HDI).

Each cloud provider has unique steps and configuration options. For more details on enabling agents for this deployment option refer to the Hadoop Integration section.

A detailed guide for EMR is provided below.

Collibra Data Quality & Observability on EMR Architecture

Collibra DQ is able to use EMR as the compute space for data quality jobs (DQ Jobs). While it is possible to simply operate a long running EMR cluster, EMR's intended operating model is ephemeral infrastructure. Using EMR as an ephemeral compute space is the most cost effective approach both in terms of operational effort and infrastructure costs. Collibra DQ makes it possible to seamlessly leverage EMR in this operating model. When there is not an EMR cluster available, Collibra DQ users are still able to browse datasets and DQ results in DQ Web. However, if a user attempts to deploy a DQ Job, they will see a red light icon next to the target agent. If the user still wants to request a DQ Job, it will simply wait in queue until the target agent comes back online the next time an EMR cluster is available.



Prepare for Deployment

> Note Before allowing Collibra DQ to use EMR as the compute space, make sure that DQ Web and the DQ Metastore are already deployed. For more installation details, see Standalone Install.

1. Create a "bootstrap bucket" location in S3 where Collibra DQ binaries and bootstrap script (`install-agent-emr.sh`) will be staged. The EMR cluster instances will need to include an attached Role that has access to this location in order to bootstrap the cluster. This location should not contain any data or any kind of sensitive materials and thus should not require any special permissions. It just needs to be accessible by EMR clusters for bootstrap purposes.
2. Create or modify an instance Profile Role that will be attached to EMR clusters so that it enables read access to the bootstrap bucket location. This Role is separate from the EMR service role that EMR will use to deploy the infrastructure.
3. Stage the bootstrap script and the Collibra DQ binary package in the bootstrap location created above.
4. Make sure that the VPC where the Collibra DQ Metastore is deployed is accessible from the VPC where EMR clusters will be deployed.
5. Make sure that Security Groups applied to the Collibra DQ Metastore are configured to allow access from EMR master and worker Security Groups.
6. Decide whether to use EMR 5.x or EMR 6.x. This is important because EMR 6 introduces Spark 3 and Scala 2.12. If EMR 6 is chosen, make sure Collibra DQ binaries were compiled for Spark 3 and Scala 2.12.
7. (OPTIONAL) Create and store a private key to access EMR instances.

Deploy an EMR Cluster

There are several ways to deploy EMR, but for dev-ops purposes, the best path is usually to use the AWS CLI utility. The example below will deploy an EMR cluster bootstrapped with Collibra DQ binaries and a functioning agent to deploy DQ Jobs.

> Note When defining the Bootstrap Location argument, do not include "s3://". For example: If Bootstrap Location is s3://bucket/prefix then BOOTSTRAP_LOCATION="bucket/prefix".

Warning Before you deploy EMR, edit `install-agent-emr.sh` and add the following export variables to it for the setup script to run correctly:

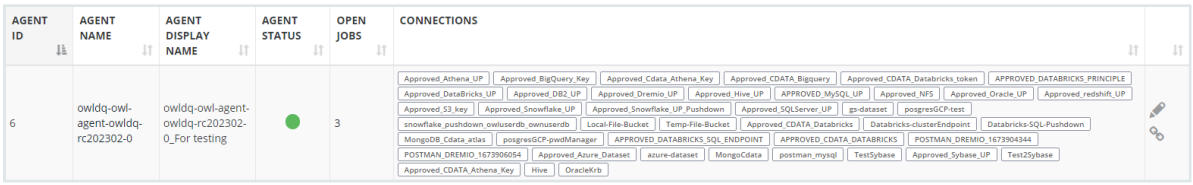
```
export DQ_ADMIN_USER_PASSWORD
export DQ_ADMIN_USER_EMAIL
```

```
aws emr create-cluster \
--auto-scaling-role EMR_AutoScaling_DefaultRole \
--applications Name=Hadoop Name=Spark Name=Hive Name=Tez \
--name owl-emr \
--release-label emr-6.2.0 \
--region ${EMR_REGION} \
--ebs-root-volume-size 10 \
--scale-down-behavior TERMINATE_AT_TASK_COMPLETION \
--enable-debugging \
--bootstrap-actions \
"[{\"Path\":\"s3://${BOOTSTRAP_LOCATION}/install-agent-emr.sh\",
\
\"Args\":[ \
\"${OWL_VERSION}\", \
\"${OWL_AGENT_ID}\", \
\"${METASTORE_HOST}:${METASTORE_PORT}/${METASTORE_DB}?-
currentSchema=owlhub\", \
\"${METASTORE_USER}\", \
\"${METASTORE_PASSWORD}\", \
\"${BOOTSTRAP_LOCATION}\", \
\"${LICENSE_KEY}\", \
\"native\"], \"Name\":\"install-owl-agent\"}]" \
--ec2-attributes "{ \
\"KeyName\":\"${EMR_INSTANCE_PRIVATE_KEY_NAME}\", \
\"InstanceProfile\":\"${BOOTSTRAP_ACCESS_ROLE}\", \
\"SubnetId\":\"${EMR_SUBNET}\", \
\"EmrManagedSlaveSecurityGroup\":\"${EMR_WORKER_SECURITY_
GROUP}\", \
\"EmrManagedMasterSecurityGroup\":\"${EMR_WORKER_SECURITY_
GROUP}\" \
}" \
--service-role ${EMR_SERVICE_ROLE} \
--log-uri s3n://${EMR_LOG_LOCATION} \
--instance-groups "[ \
{\"In-
stanceCoun-
t\":1,\"InstanceGroupType\":\"MASTER\",\"InstanceType\":\"${EMR_
MASTER_INSTANCE_TYPE}\",\"Name\":\"Master - 1\"}, \
{\"In-
stanceCoun-
t\":3,\"InstanceGroupType\":\"CORE\",\"InstanceType\":\"${EMR_
CORE_INSTANCE_TYPE}\",\"Name\":\"Core - 2\"} \
]"
```

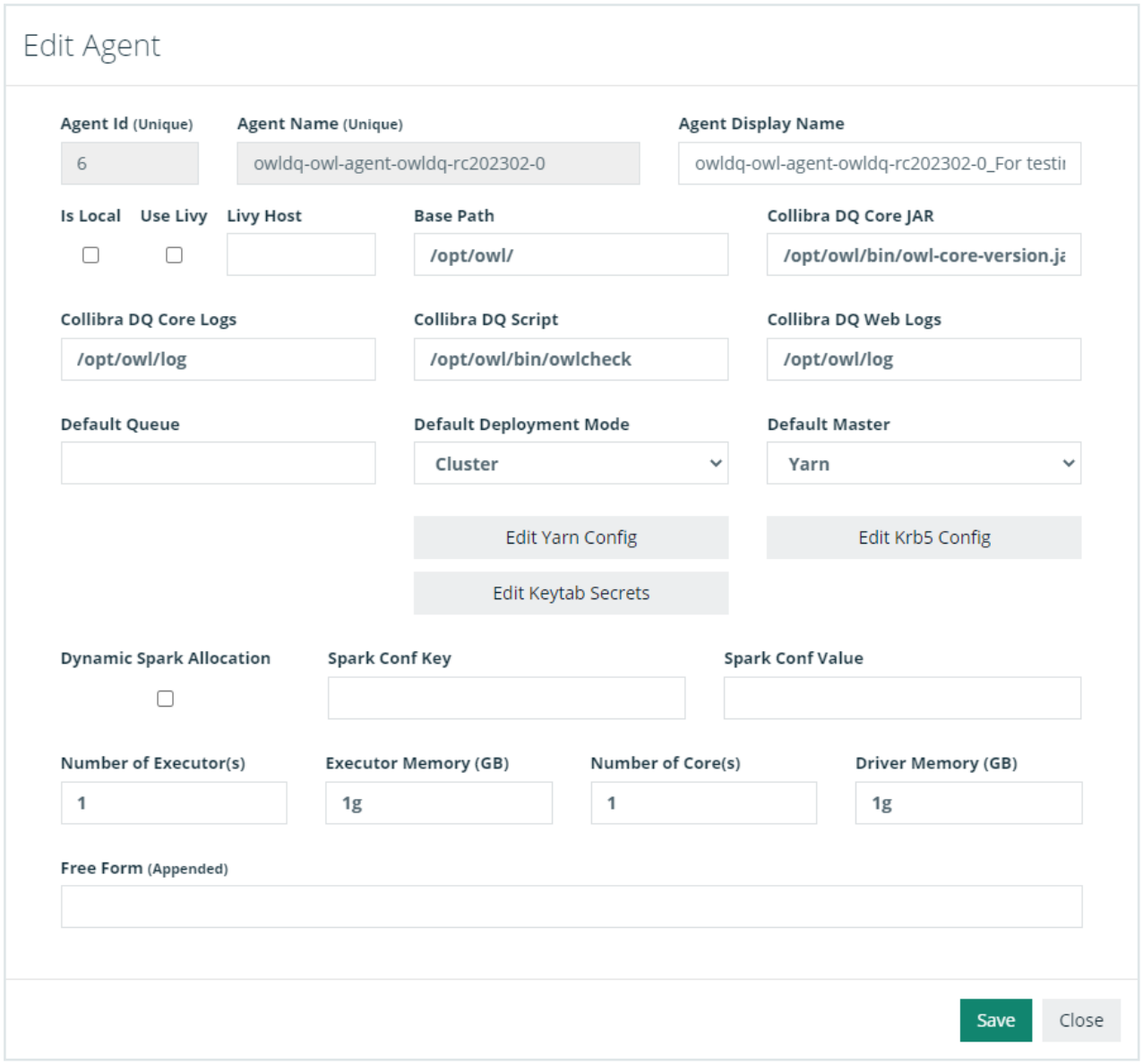
Configure Agent

Once the EMR cluster and the DQ Agent is deployed, it needs to be configured in DQ Web.

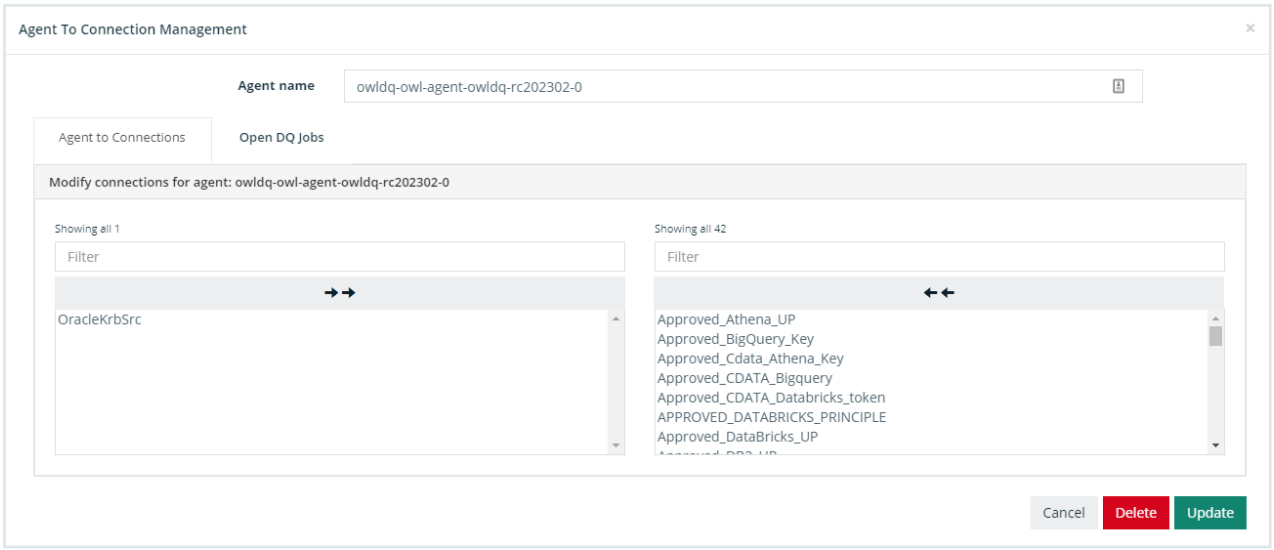
1. Sign in to your web instance of Collibra DQ.
2. Hover your cursor over the ⚙ icon and click **Admin Console**.
   » The Admin Console opens.
3. Click the **Remote Agent** tile.
   » The **Agent Management** page opens.
   » In the following example, the newly created agent has a green indicator to show that it is healthy and active.



4. Click the pencil icon to the right of the Connections column.
   » The **Edit Agent** modal appears.
5. From the **Default Deploy Mode** drop-down list, select **Cluster**.
6. From the **Default Master** drop-down list, select **Yarn**.



7. Click **Save**.
8. On the Agent Management page, click the chain link icon next to the Connections column to map the DQ Agent to Connections.
   » The **Edit Agent** modal appears.



> Note Any data sources that are not listed in the right hand pane are not visible to the agent. Refer to the Agent section for more details.

9. Click **Update**.

Run DQ Jobs

You can now use EMR to run DQ Jobs on your data. Refer to the Explorer (no-code) section for more details.

Cloud native

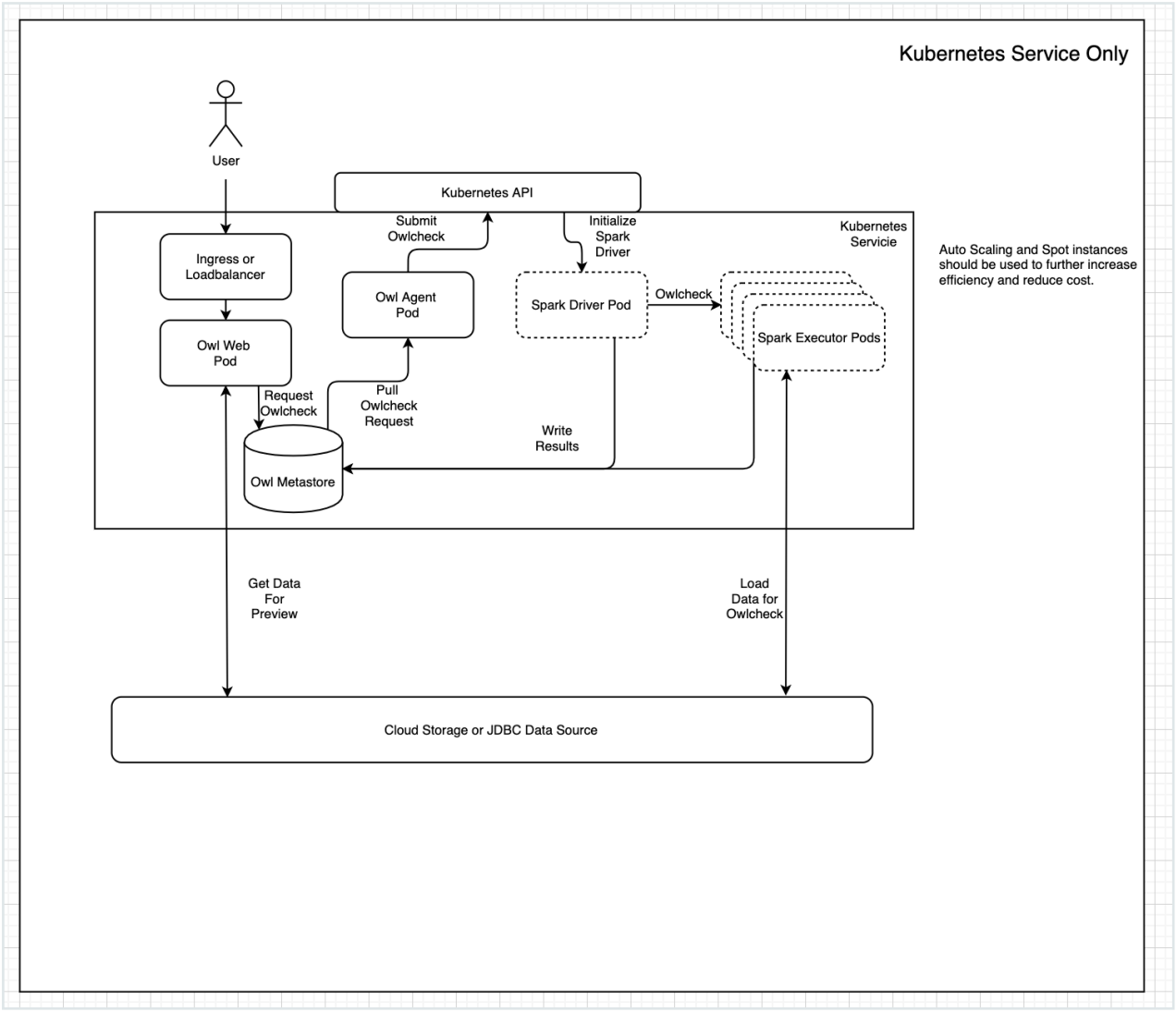
Introduction to cloud native architecture

According to the Cloud Native Computing Foundation ("CNCF") Charter:

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

Collibra Data Quality wholeheartedly embraces these principles in its design and deployment. The diagram below depicts Collibra Data Quality & Observability's cloud native deployment architecture:



In this "form factor", you can deploy Collibra DQ in any public or private cloud while maintaining a consistent experience, performance, and management runbook.

Collibra DQ microservices

To achieve cloud native architecture, Collibra DQ is decomposed into several components, each of which is deployed as a microservice in a container.

- **Owl Web** - The main point of entry and interaction between Collibra DQ and end users or integrated applications. Owl Web provides both a rich, interactive user experience and a robust set of APIs for automated integration.
- **Owl Agent** - You can think of the Agent as the "foreman" of Collibra DQ. When a user or application requests a data quality check through Owl Web, Owl Agent will marshal compute resources to perform the work. Owl Agent does not actually do any of the data quality work. Instead, it translates the request submitted by Owl Web into a technical descriptor of the work that needs to be done and then launches the requested DQ job.
- **Owl Metastore** - This is where Collibra DQ stores all the metadata, statistics, and results of DQ jobs. It is also then main point of communication between Owl Web and Owl Agent. The metastore also contains the results of DQ jobs performed by transient containers (workers) in the compute space.
- **History Server** - Collibra DQ relies on Apache Spark to actually scan data and perform the bulk of data quality activities. To facilitate troubleshooting and performance tuning of DQ jobs, Collibra DQ uses an instance of Spark History Server to enable easy access to Spark logs.
- **Spark** - Apache Spark is the distributed compute framework that powers the Collibra DQ data quality engine. Spark enables DQ jobs to rise to the task of data quality on Terabyte scale datasets. Spark containers are completely ephemeral and only live for as long as necessary to complete a given DQ job.

Containerization

The binaries and instruction sets described in each of the Collibra DQ microservices are encompassed within Docker container images. Each of the images is versioned and maintained in a secured cloud container registry repository. To initiate a Collibra DQ cloud native deployment, you must first obtain credentials to either pull the containers directly or download them to a private container registry.

> Warning Support for Collibra DQ cloud native deployment is limited to deployments using the containers provided from the Collibra container registry.
>
> Reach out to your customer contact for access to pull the Collibra containers.

Kubernetes

Kubernetes is a distributed container scheduler and has become synonymous with cloud native architecture. While Docker containers provide the logic and runtime at the application layer, most applications still require network, storage, and orchestration between multiple hosts in order to function. Kubernetes provides all of these facilities while abstracting away all of the complexity of the various technologies that power the public or private cloud hosting the application.

> Note For complete details on how to install Collibra DQ on Kubernetes with Docker containers, see Cloud native install.

> Note Collibra DQ supports Kubernetes versions v1.21 - v1.24.

Collibra DQ Helm chart

While Kubernetes currently provides the clearest path to gaining the benefits of a cloud native architecture, it is also one of the more complex technologies in existence. This has less to do with Kubernetes itself and more with the complexity of the constituent technologies it is trying to abstract. Technologies like attached distributed storage and software defined networks are entire areas of specialization that require extensive expertise to navigate. Well implemented Kubernetes platforms hide all of this complexity and make it possible for anyone to leverage these powerful concepts. However, a robust application like Collibra DQ requires many descriptors (K8s manifests) to deploy its various components and all of the required supporting resources like network and storage.

This is where Helm comes in. Helm is a client side utility (since v3) that automatically generates all the descriptors needed to deploy a cloud native application. Helm receives instructions in the form of a **Helm chart** that includes templated and parameterized versions of Kubernetes manifests. Along with the Helm chart, you can also pass arguments like names of artifacts, connection details, enable and disable commands, and so on. Helm resolves the user defined parameters within the manifests and submits them to Kubernetes for deployment. This enables you to deploy the application without

necessarily having a detailed understanding of the networking, storage or compute that underpins the application.

For example, the command below deploys Collibra DQ with all of the components depicted in the Introduction to cloud native architecture into Google Kubernetes Engine with Google Cloud Storage (GCS) as the storage location for Spark logs. The only prerequisite is that the image pull secret, representing credentials to access the container registry, and secret containing the credentials for a service account with access to GCS are already deployed to the namespace.

```
helm upgrade --install --namespace <namespace> \
--set global.version.owl=<owl-version> \
--set global.version.spark=<owl-spark-version> \
--set global.configMap.data.license_key=<owl-license-key> \
--set global.spark_history.enabled=true \
--set global.spark_history.logDirectory=gs://logs/spark-history/
\
--set global.cloudStorage.gcs.enableGCS=true \
<deployment-name> \
/path/to/chart/owldq
```

> Note The full universe of possible customizations is quite extensive and provides a great deal of flexibility in order to be applicable in a wide variety of platforms. However, when deploying on a known platform (EKS, GKE, AKS), the number of required inputs is quite limited. In common cases, you run a single CLI command including basic parameters like disable history server, configure the storage bucket for logs, specify the image repository, and so on.

Cloud native requirements

Minimum requirements

You need a machine with the following files and packages to run the installation. You can run these from a laptop or separate VM and they do not need to be issued on the Kubernetes cluster itself.

> Note For complete details on how to install Collibra DQ on Kubernetes with Docker containers, see Cloud native install.

Prerequisites

- Kubernetes cluster -- EKS, GKE, AKS, Openshift, Rancher
- Helm(v3)
- kubectl
- Cloud command line SDK, such as gcloud CLI, AWS CLI or similar
- External Postgres DB version 11.9 and above, storage size 100GB, cores 4 to 8 memory to 4 to 8 GB
- Private container registry -- to store images
- LoadBalancer -- IngressController -- Ingress
- Egress networking access
- Helm chart
- Images, image access key
- Minimum pod requirement -- 2 cores, 2GB RAM

Files

- The helm chart.
- JKS files with secrets created in kubectl:
  - *owldq-ssl-secret*
  - *owldq-pull-secret**
- A *spark-gcs-secret* you create from your service account file or token.

> Note * Available upon request from Collibra.

Application system requirements

Component	Processor	Memory	Storage
Collibra DQ Web	1 core	2 GB	10 MB PVC
Owl Agent	1 core	1 GB	100 MB PVC

Component	Processor	Memory	Storage
Owl Metastore	1 core	2 GB	10 GB PVC
Spark*	2 cores	2 GB	-

> Note * This is the minimum quantity of resources required to run an a Spark job in Kubernetes. This amount of resources would only provide the ability to scan a few megabytes of data with no more than a single job running at a given time. Proper sizing of the compute space must take into account the largest dataset that may be scanned, as well as the desired concurrency.

Network service considerations

Owl Web is the only required component that needs to be directly accessed from outside of Kubernetes. History Server is the only other component that can be accessed directly by users, however, it is optional.

If the target Kubernetes platform supports a LoadBalancer service type, you can configure the Helm chart to directly deploy the externally accessible endpoint.

> Note For testing purposes, you can also configure the Helm chart to deploy a NodePort service type.

For the Ingress service type, deploy OwlDQ without an externally accessible service and then attach the Ingress service separately. This applies when you use a third-party Ingress controller such as NGINX, Contour, etc.

> Note The Helm chart is able to deploy an Ingress on GKE and EKS platforms, however, there is a wide variety of possible Ingress configurations that have not been tested.

Obtaining credentials

Kubernetes stores credentials in the form of secrets. Secrets are base64 encoded files that you can mount into application containers and that application components can

reference at runtime. You use *pull secrets* to access secured container registries to obtain application containers.

> Note Although it is not recommended, you should only access the Collibra image registry for the initial download and validation of Docker images. After this, images should be uploaded and stored on your private registry, which allows you to control when the images are updated, as well as eliminate any operational dependencies on Collibra's repository.

SSL certificates

To enable SSL for secure access to Owl Web, a keystore that contains a signed certificate, keychain, and private key is required. This keystore must be available in the target namespace before you deploy Collibra DQ.

> Note By default, Collibra DQ looks for a secret called `owldq-ssl-secret` to find the keystore.

> Note Although it is possible to deploy with SSL disabled, is not recommended.

Cloud storage credentials

If you enable History Server, a distributed filesystem is required. Currently, Collibra DQ supports S3 and GCS for Spark history log storage.

> Note Azure Blob and HDFS on the near term roadmap.

Target storage system	Credentials requirements
S3	An IAM Role with access to the target bucket needs to be attached to the Kubernetes nodes of the namespace where Collibra DQ is being deployed.

Target storage system	Credentials requirements
GCS	You must create a secret from the JSON key file of a service account with access to the log bucket. The secret must be available in the namespace before you deploy Collibra DQ. By default, Collibra DQ looks for a secret called `spark-gcs-secret`, if GCS is enabled for Spark history logs. You can change this via a helm chart argument.

Container pull secret

Collibra Data Quality & Observability containers are stored in a secured repository in Google Container Registry. For Collibra DQ to successfully pull the containers when deployed, a pull secret with access to the container registry must be available in the target namespace.

> Note By default, Collibra DQ looks for a pull secret named `owldq-pull-secret`. You can change this via a helm chart argument.

Spark service account

To enable Owl Agent and the Spark driver to create and destroy compute containers, you must have a service account with a role that allows get/list/create/delete operations on pods/services/secrets/configMaps in the target namespace. By default, Collibra DQ attempts to create the required service account and the required RoleBinding to the default Edit role. Edit is a role that is generally available in a Kubernetes cluster. If the Edit role is not available, you can manually create it.

Accessing the platform

To deploy anything to a Kubernetes cluster, the first step is to install the required client utilities and configure access:

- **kubectl**: The main method of communication with a Kubernetes cluster. All configuration or introspection tasks will be preformed using kubectl.
- **helm v3**: Used to deploy the OwlDQ helm chart without hand coding manifests.

After you install the utilities, the next step is to configure a kube-context that points to and authenticates to the target platform. On cloud platforms like GKE and EKS, this process is completely automated through their respective CLI utilities.

```
aws eks --region <region-code> update-kubeconfig --name
<cluster_name>
```

```
gcloud container clusters get-credentials <cluster-name>
```

In private clouds, this process will vary from organization to organization, however, the platform infrastructure team should be able to provide the target kube-context entry.

Preparing secrets

Once access to the target platform is confirmed, you can begin the preparation of the namespace. Typically the namespace that Collibra DQ is going to be deployed into is pre-allocated by the platform team.

```
kubectl create namespace <namespace>
```

> Note There is a lot more that can go into namespace creation such as resource quota allocation, but that is generally a task for the platform team.

Create an SSL keystore secret

> Note For complete details on how to install Collibra DQ on Kubernetes with Docker containers, see Cloud native install.

Create a container pull secret

> Note For complete details on how to install Collibra DQ on Kubernetes with Docker containers, see Cloud native install.

JSON key file credential

```
kubectl create secret docker-registry owldq-pull-secret \
--docker-server=<owldq-registry-server> \
--docker-username=_json_key \
--docker-email=<service-account-email> \
--docker-password="$(cat /path/to/key.json)" \
--namespace <namespace>
```

Short lived access token

```
kubectl create secret docker-registry owldq-pull-secret \
--docker-server=<owldq-registry-server> \
--docker-username=oauth3accesstoken \
--docker-email=<service-account-email> \
--docker-password="<access-token-text>" \
--namespace <namespace>
```

> Warning GCP Oauth tokens are usually only good for 1 hour. This type of credential is excellent if the goal is to pull containers into a private registry. It can be used as the pull secret to access containers directly, however, the secret would have to be recreated with a fresh token before restarting any of the Colbra DQ components.

Create a GCS credential secret

```
kubectl create secret generic spark-gcs-secret \
--from-file /path/to/keystore.jks \
--namespace <namespace>
```

Warning The file name that you use in the `--from-file` argument should be *spark-gcs-secret*. If the file name is anything else, you must include an additional argument specifying the gcs secret name in the Helm command.

Note For complete details on how to install Collibra DQ on Kubernetes with Docker containers, see Cloud native install.

Cloud native install

Install Collibra DQ on Kubernetes with Docker Containers

Collibra DQ provides the following Core Docker containers:

- **Owl-Agent**: Launches the Apache Spark Jobs
- **Owl-Web**: The Collibra DQ web application itself
- **Apache Spark**: The runtime analytics engine
- **Postgres** (persistent volume needed): The Collibra DQ metastore
- **Apache Livy**:
  - Session Manager: How Collibra DQ can browse the HDFS, S3, GCS, or Azure Data Lake (ADL)
  - Interacts with Object Stores, similar to JDBC sources Explorer (estimate Jobs, get days with data, Filtergrams, etc.)

Prerequisites

- Kubernetes cluster -- EKS, GKE, AKS, Openshift, Rancher
- Helm(v3)
- kubectl
- Cloud command line SDK, such as gcloud CLI, AWS CLI or similar
- External Postgres DB version 11.9 and above, storage size 100GB, cores 4 to 8 memory to 4 to 8 GB
- Private container registry -- to store images
- LoadBalancer -- IngressController -- Ingress
- Egress networking access
- Helm chart

- Images, image access key
- Minimum pod requirement -- 2 cores, 2GB RAM

Steps

To install Collibra DQ on Kubernetes with Docker containers, follow these steps.

Sign in to the Kubernetes cluster

1. Sign in to the Kubernetes cluster from a Linux compatible terminal.
2. Create a namespace in the cluster using the following code snippet:

```
kubectl create namespace <owldq>
```

Pull images from the Collibra registry

Collibra DQ containers are located in the Docker Hub registry. Collibra provides a repo-key to access Collibra images in a .json file, which can be stored locally and used to login.

1. Download the Docker .json repo-key.
2. Run the following command:

```
docker login -u _json_key -p "$(cat repo-key.json)"
https://gcr.io
```

> Note Image names with their versions are provided by Collibra.

3. To pull the images, run the following Docker pull commands:

```
docker pull gcr.io/owl-hadoop-cdh/owl-agent:2021.09
docker pull gcr.io/owl-hadoop-cdh/owl-web:2021.09
docker pull gcr.io/owl-hadoop-cdh/spark:3.0.1-2021.09
docker pull gcr.io/owl-hadoop-cdh/owl-livy:3.0.1-2021.09
```

Push images into your private registry

1. Sign in to your private Docker container registry.
2. Tag and push the images from Collibra to your private registry, by using the following commands:

```
docker tag gcr.io/owl-hadoop-cdh/owl-web:2021.10.3
<registryURL>/owl-web:2021.10.3
docker push <registryURL>/owl-web:2021.10.3
```

Example:
docker tag [OPTIONS] IMAGE [:TAG][REGISTRYHOST/][USERNAME/]NAME[:TAG]

docker tag push NAME[:TAG]

Create an SSL keystore secret

```
kubectl create secret generic owldq-ssl-secret \
--from-file /path/to/keystore.jks \
--namespace <namespace>
```

> Warning The file name that you use in the `--from-file` argument should be *keystore.jks*. If the file name is anything else, you must include an additional argument specifying the keystore file name in the Helm command.

Create a pull secret

> Note Although it is not recommended, you should only access the Collibra image registry for the initial download and validation of Docker images. After this, images should be uploaded and stored on your private registry, which allows you to control when the images are updated, as well as eliminate any operational dependencies on Collibra's repository.

1. To create a pull secret, use the following code snippet:

```
kubectl create secret docker-registry owldq-pull-secret
--docker-server=https://gcr.io
--docker-username=_json_key
--docker-email=<email of customer>
--docker-password="$(cat repo-key.json)"
--namespace <owldq>
```

> Note If your private registry is used for images and if they are accessible from within the Kubernetes cluster, this secret need not be created. If credentials are required to access your private registry, create this secret by modifying the docker-server URL and docker-password.

Helm chart

For more detailed information about the Helm Chart, see Cloud native.

Unzip and store the Helm charts provided by Collibra on a Linux compatible deployment location.

> Note Once you have your Collibra DQ license, you will receive an email from Collibra that includes the Helm Charts as zip files.

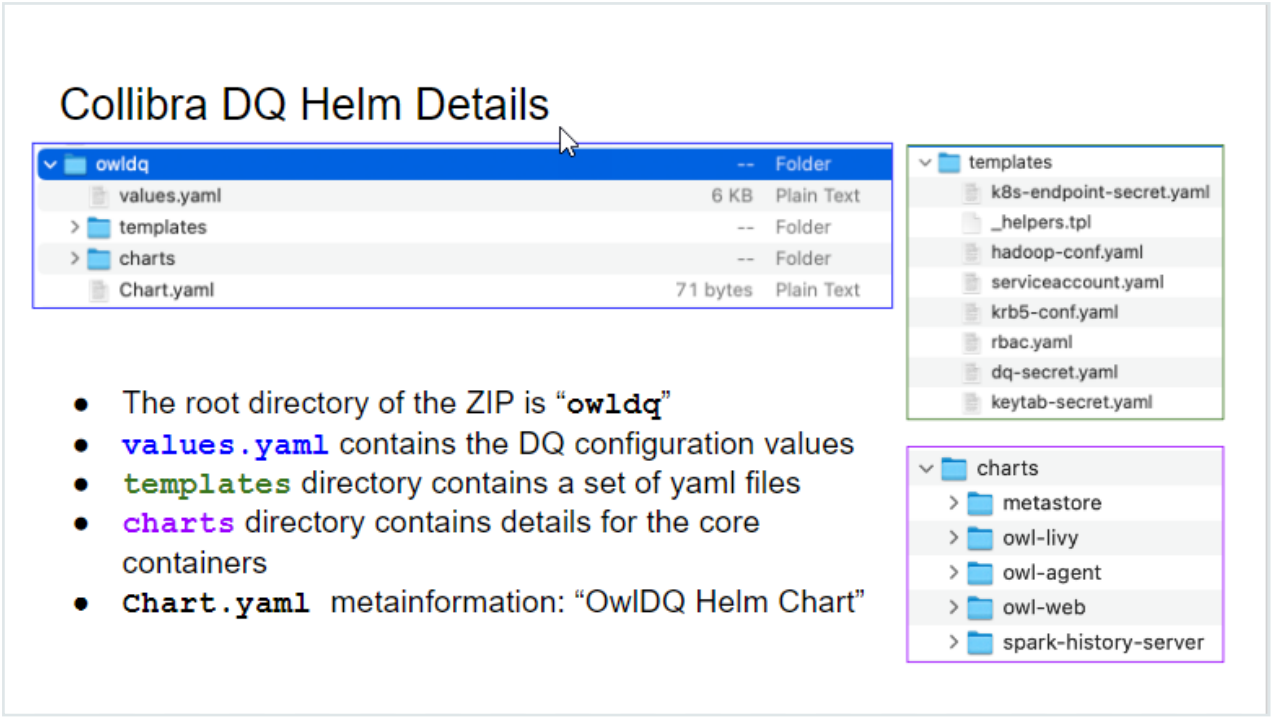
There should be two folders and two files:

- drwxrwxr-x -- templates
- drwxrwxr-x -- charts
- -rw-rw-r-- Chart.yaml
- -rw-rw-r-- values.yaml

There are two ways of passing parameters. While deploying, parameter values can be passed:

2. Using the values.yaml file or,
3. Using the Helm set commands.

> Note The set commands take precedence over the values.yaml file.



Deploy Collibra Data Quality & Observability

Once you have created a namespace and added all of the required secrets, you can begin the deployment of Collibra DQ.

Minimal install

Install Web, Agent, and metastore. Collibra DQ is inaccessible until you manually add an Ingress or another type of externally accessible service.

> Warning All of the following examples will pull containers directly from the Collibra DQ secured container registry. In most cases, InfoSec policies require that containers are sourced from a private container repository controlled by the local Cloud Ops team. Make sure to to add `--set global.image.repo=</url/of/private-repo>` so that you use only approved containers.

> Note The metastore container must start first as the other containers use it to write data. On your initial deployment, the other containers might start before the metastore and fail.

```
helm upgrade --install --namespace <namespace> \
--set global.version.owl=<owl-version> \
--set global.version.spark=<owl-spark-version> \
--set global.configMap.data.license_key=<owl-license-key> \
--set global.configMap.data.license_name=<your-license-name> \
--set global.web.admin.email=${email} \
--set global.web.admin.password=${password} \
--set global.web.service.type=ClusterIP \
--set global.image.repo=<pathTolmageRepo> \
<deployment-name> \
/path to the helm chart root folder>
```

Value	Description
<namespace>	Enter the namespace that you created in the Sign in to the Kubernetes cluster step.
<owl-version>	Enter the version from the web image suffix. For example, **2023.02** from the image, **owl-web:2023.02**.
<spark-version>	Enter the Spark version from the Spark image suffix. For example, **3.2.2-2023.02** from the image, **spark:3.2.2-2023.02**.
<owl-license-key>	Enter the license key provided to you by Collibra.
<your-license-name>	Enter the license name provided to you by Collibra.
${email}	Enter the default admin user email associated with the admin account.

Value	Description
${password}	Enter the default admin user password for the admin account. The password must adhere to the following password policy: • A minimum of 8 characters. • A maximum of 72 characters. • At least one upper-case character. • At least one numeric character. • At least one supported special character (`!@#%$^&*?_~`). • Cannot contain the user ID (admin). Note If a password that does not meet the password policy is entered, the install process proceeds as though the password is accepted, but the admin user becomes locked out. If this occurs, rerun the Helm command with a password that meets the password policy and restart the web pod.
<pathToImageRepo>	This is your private registry key, where the Collibra images are available. When this is not provided, you will pull the images from the Collibra image registry, for which you should create a pull secret with the repo key provided by Collibra. See Create a pull secret for more details.
<deployment-name>	Any name of your choice for this deployment.

Note If you optionally pass credentials for the Postgres Metastore, ensure that you do not use the `$` symbol in the `global.metastore.pass` variable, as it is an unsupported special character for Postgres Metastore passwords.

Externally accessible service

Perform the Minimal install and add a preconfigured NodePort or LoadBalancer service to provide access to the Web.

Warning A LoadBalancer service type requires that the Kubernetes platform is integrated with a Software Defined Network solution. This will generally be true for the Kubernetes services offered by major cloud vendors. Private cloud platforms more commonly use Ingress controllers. Check with the infrastructure team before attempting to use LoadBalancer service type.

```
helm upgrade --install --namespace <namespace> \
--set global.version.owl=<owl-version> \
--set global.version.spark=<owl-spark-version> \
--set global.configMap.data.license_key=<owl-license-key> \
--set global.configMap.data.license_name=<your-license-name> \
--set global.web.admin.email=${email} \
--set global.web.admin.password=${password} \
--set global.web.service.type=<NodePort || LoadBalancer> \
<deployment-name> \
/path/to/chart/owldq
```

Externally accessible with SSL enabled

Perform the install with external service but with SSL enabled.

> Note Ensure you have already deployed a keystore containing a key to the target namespace with a secret name that matches the `global.web.tls.key.secretName` argument (*owldq-ssl-secret* by default). Also, ensure that the secret's key name matches the `global.web.tls.key.store.name` argument (*dqkeystore.jks* by default).

```
helm upgrade --install --namespace <namespace> \
--set global.version.owl=<owl-version> \
--set global.version.spark=<owl-spark-version> \
--set global.configMap.data.license_key=<owl-license-key> \
--set global.configMap.data.license_name=<your-license-name> \
--set global.web.admin.email=${email} \
--set global.web.admin.password=${password} \
--set global.web.service.type=<NodePort || LoadBalancer> \
--set global.web.tls.enabled=true \
--set global.web.tls.key.secretName=owldq-ssl-secret \
--set global.web.tls.key.alias=<key-alias> \
--set global.web.tls.key.type=<JKS || PKCS12> \
--set global.web.tls.key.pass=<keystore-pass> \
--set global.web.tls.key.store.name=keystore.jks \
<deployment-name> \
/path/to/chart/owldq
```

Externally accessible and History Server for GCS Log Storage

Perform the install with external service and Spark History Server enabled. In the following example, the target log storage system is GCS.

> Note For Collibra DQ to be able to write Spark logs to GCS, create a secret from the JSON key file of a service account that has access to the log bucket. For more detailed information, see Cloud storage credentials.

```
helm upgrade --install --namespace <namespace> \
--set global.version.owl=<owl-version> \
--set global.version.spark=<owl-spark-version> \
--set global.configMap.data.license_key=<owl-license-key> \
--set global.configMap.data.license_name=<your-license-name> \
--set global.web.admin.email=${email} \
--set global.web.admin.password=${password} \
--set global.web.service.type=<NodePort || LoadBalancer> \
--set global.spark_history.enabled=true \
--set global.spark_history.logDirectory=gs://logs/spark-history/
\
--set global.spark_history.service.type=<NodePort || LoadBal-
ancer> \
--set global.cloudStorage.gcs.enableGCS=true \
<deployment-name> \
/path/to/chart/owldq
```

Externally Accessible and History Server for S3 Log Storage

Perform the install with external service and Spark History Server enabled. In this example, the target log storage system is S3.

> Note For Collibra DQ to be able to write Spark logs to S3, makes sure that an Instance Profile IAM Role with access to the log bucket is attached to all nodes serving the target namespace. For more detailed information, see Cloud storage credentials.

```
helm upgrade --install --namespace <namespace> \
--set global.version.owl=<owl-version> \
--set global.version.spark=<owl-spark-version> \
--set global.configMap.data.license_key=<owl-license-key> \
--set global.configMap.data.license_name=<your-license-name> \
--set global.web.admin.email=${email} \
--set global.web.admin.password=${password} \
--set global.web.service.type=<NodePort || LoadBalancer> \
--set global.spark_history.enabled=true \
--set global.spark_history.logDirectory=s3a://logs/spark-
history/ \
--set global.spark_history.service.type=<NodePort || LoadBal-
ancer> \
--set global.cloudStorage.s3.enableS3=true \
<deployment-name> \
/path/to/chart/owldq
```

Externally accessible with external metastore

Perform the install with external service and an external metastore, for example AWS RDS, Google Cloud SQL, or just PostgresSQL on its own instance.

> Warning Collibra DQ currently supports PostgreSQL 9.6 and newer.

```
helm upgrade --install --namespace <namespace> \
--set global.version.owl=<owl-version> \
--set global.version.spark=<owl-spark-version> \
--set global.configMap.data.license_key=<owl-license-key> \
--set global.configMap.data.license_name=<your-license-name> \
--set global.web.admin.email=${email} \
--set global.web.admin.password=${password} \
--set global.web.service.type=<NodePort || LoadBalancer> \
--set global.metastore.enabled=false
--set global.configMap.data.metastore_url-
l=jdbc:postgresql://<host>:<port>/<database>
--set global.configMap.data.metastore_user=<user> \
--set global.configMap.data.metastore_pass=<password> \
<deployment-name> \
/path/to/chart/owldq
```

> Warning The `$` symbol is not a supported special character in your Postgres Metastore password.

Troubleshooting and Helpful Commands

This guide is to provide the most common commands to run when troubleshooting a DQ environment that is deployed on Kubernetes. For a basic overview of Kubernetes and other relevant knowledge.

```
### Provide documentation on syntax and flags in the terminal

kubectl help

### To see how to use Kubernetes resources

kubectl api-resources -o wide
```

Viewing Kubernetes Resources

```
### Get Pods, their names & details in all Namespaces

kubectl get pods -A -o wide

### Get all Namespaces in a cluster

kubectl get namespaces

### Get Services in all Namespaces

kubectl get services -A -o wide

### List all deployments in all namespaces:

kubectl get deployments -A -o wide
```

Logs & Events

```
### List Events sorted by timestamp in all namespaces

kubectl get events -A --sort-by=.metadata.creationTimestamp

### Get logs from a specific pod:

kubectl logs [my-pod-name]
```

Resource Allocation

```
### If the Kubernetes Metrics Server is installed,
### the top command allows you to see the resource consumption
for nodes or podscode

kubectl top node
kubectl top pod

### If the Kubernetes Metrics Server is NOT installed, use

kubectl describe nodes | grep Allocated -A 10
```

Configuration

```
### Get current-context

kubectl config current-context

### See all configs for the entire cluster

kubectl config view
```

Authorization Issues

```
### Check to see if I can read pod logs for current user & con-
text

kubectl auth can-i get pods --subresource=log

### Check to see if I can do everything in my current namespace
("*" means all)

kubectl auth can-i '*' '*'
```

Configure custom JDBC data source drivers

This section shows you how to set up persistent external volumes on existing cloud native deployments of Collibra DQ to support the use of custom JDBC data source drivers.

Steps

1. Create a Persistent Volume (PV) and a Persistent Volume Claim (PVC) in the same Kubernetes cluster namespace where your cloud native deployment of Collibra DQ is running with `ReadWriteMany` access mode, as shown in the following example:

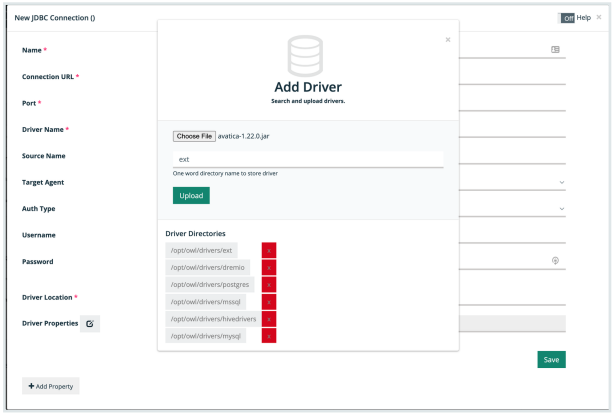
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: nfs-server.nfs.svc.cluster.local
    path: "/"

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 10Gi
```

> Tip You can choose any storage class that supports ReadWriteMany access mode only.

2. Patch the existing DQ Web and Agent StatefulSets with an additional Volume Mount, as shown in the following example:

```
    spec:
      volumes:
        - name: owldq-ext-jdbc-jars
          persistentVolumeClaim:
            claimName: nfs-pvc
    ...
    ...
      containers:
        - volumeMounts:
            - name: owldq-ext-jdbc-jars
              mountPath: /opt/owl/drivers/ext
```

3. From the Connections page, click **Add** on the **Generic JDBC Connection** tile.
   » The **New JDBC Conection** modal appears.
4. Enter the required information.
5. Click the folder icon in the **Driver Location** option.
   » The **Add Driver** modal appears.
6. Click **Choose File** and select the JAR file to upload.
7. Enter a one-word directory name to store the driver, for example, *ext*.
8. Select `/opt/owl/drivers/ext` under **Driver Directories** to upload your driver to that location, as shown in the image below.



9. Click **Upload**.
   » A success message appears when your driver is successfully added.

> Note
> Uploads from DQ Web only work if the PVC's storageclass supports the fsGroup security context of Kubernetes. If the fsGroup security context is not supported, create a temporary dummy pod running as a root user with the PVC mounted, and copy the required JDBC jars to the mount location with 544 permissions.

10. Go to the **Remote Agent** section of the Admin Console and click the pencil icon to edit your agent.
    » The **Edit Agent** modal appears.
11. Add the following properties to the **Free Form (Appended)** option, as shown in the following example:

```
-conf spark.kubernetes.driver.podTemplateFile=https://spark-
on-k8s-templates-public.s3.amazonaws.com/spark-on-k8s-tem-
plate.ym-
l,spark.kubernetes.executor.podTemplateFile=https://spark-
on-k8s-templates-public.s3.amazonaws.com/spark-on-k8s-
tem-
plate.ym-
l,s-
park.kubernetes.executor.volumes.persistentVolumeClaim.nfs-
pv.options.claimName=nfs-
pvc,s-
park.kubernetes.executor.volumes.persistentVolumeClaim.nfs-
pv.-
option-
s.s-
izeLim-
it=10Gi,-
spark.kuber-
netes.executor.volumes.persistentVolumeClaim.nfs-
pv.-
moun-
t.path-
=/o-
pt/owl/driver-
s/ex-
t,s-
park.kubernetes.executor.volumes.persistentVolumeClaim.nfs-
pv.-
moun-
t.readOnly-
=false,-
,spark.kubernetes.driver.volumes.persistentVolumeClaim.nfs-
pv.options.claimName=nfs-
pvc,s-
park.kubernetes.driver.volumes.persistentVolumeClaim.nfs-pv.-
option-
s.s-
izeLim-
it=10Gi,-
,spark.kubernetes.driver.volumes.persistentVolumeClaim.nfs-
pv.-
moun-
t.path-
=/o-
pt/owl/driver-
s/ex-
t,spark.kubernetes.driver.volumes.persistentVolumeClaim.nfs-
pv.mount.readOnly=false
```

> Note
> The pod templates induce security context to the Spark driver and executor pods, which allows you to mount the PVC dynamically. You can also download the template file and host it in your own public space.

12. Click **Save**.
13. Go to the **Connections** page to start using your new custom JDBC data source driver.

Configure Spark scratch disk space to handle large datasets

When you run large datasets, your Spark cluster's resources need to be sized appropriately to handle the computation of DQ rules. This section shows you how to configure on-demand Spark scratch disk space to handle huge Spark jobs.

Prerequisites

Before you configure Spark scratch disk space, ensure you have:

- A Collibra DQ deployment on a cloud native platform, such as EKS, GKS, and so on.
- A Kubernetes service account for Collibra DQ that meets the permission requirements to provision dynamic Persistent Volume Claims (PVCs) in ReadWriteOnce (RWO).

> Note This capability is only available for Collibra DQ versions 2023.02 or newer.

Steps

1. Open a terminal session.
2. Set the following values to deploy or update the Collibra DQ app with the provided Helm charts:

```
--set global.configMap.data.spark_scratch_type-
e="persistentVolumeClaim" --set global.configMap.data.spark_
scratch_storage_class="standard" --set glob-
al.configMap.data.spark_scratch_storage_size="20Gi" --set
global.configMap.data.spark_scratch_local_path-
h="/tmp/scratch"
```

> Note The values in the previous example are settings for GKE-based deployments.

a. If your Collibra DQ app is managed by an alternative Kubernetes package manager, ensure you update the owl-agent-configmap with the following variables:

```
KUBERNETES_SCRATCH_TYPE: persistentVolumeClaim
KUBERNETES_SCRATCH_STORAGE_CLASS: standard
KUBERNETES_SCRATCH_STORAGE_SIZE: 20Gi
KUBERNETES_SCRATCH_LOCAL_PATH: /tmp/scratch
```

> Note Pick the value of Storage Class, Storage Size and mouthPath specific to your requirements.

b. Restart the DQ agent pod to complete the updates.

3. Sign in to Collibra DQ.
4. Hover your cursor over the ⚙ icon and click **Admin Console**.
   » The Admin Console opens.
5. Click **Remote Agent**.
   » The **Agent Management** page opens.
6. Click the pencil icon.
   » The **Edit Agent** modal opens.
7. In the **Free Form (Appended)** option, enter the following string value:

```
-conf spark.kuber-
netes.driver.podTemplateFile=local:///opt/owl/config/k8s-
driver-
tem-
plate.ym-
l,s-
park.kuber-
netes.executor.podTemplateFile=local:///opt/owl/config/k8s-
executor-template.yml
```

> Tip If you add any additional configuration properties based on your specific requirements, separate them with commas.

8. Run a sample DQ job to process DQ rules against a dataset of your choice. The corresponding Spark job dynamically provisions PVCs for the driver and each executor to handle large datasets by juggling between Memory and Disk at the mouth path of the PVC.

What's next?

For more information, go to the official Spark documentation on:

- Spark on Kubernetes using Pod Templates.
- Spark on Kubernetes using Persistent Volume Claims.

EKS / GKE / AKS

For organizations that are familiar with containers and managed Kubernetes services, Collibra Data Quality & Observability offers a cloud native deployment option. Please refer to the Cloud Native section for more detail.

Cloud native

Cloud native requirements

Cloud native install

> Note The containers are part of a private repository which requires access. Please contact Collibra directly for more information.

Cloud

Collibra DQ Cloud is delivered as a Software-as-a-Service offering. This section describes the installation process and details the Edge component for Collibra DQ Cloud.

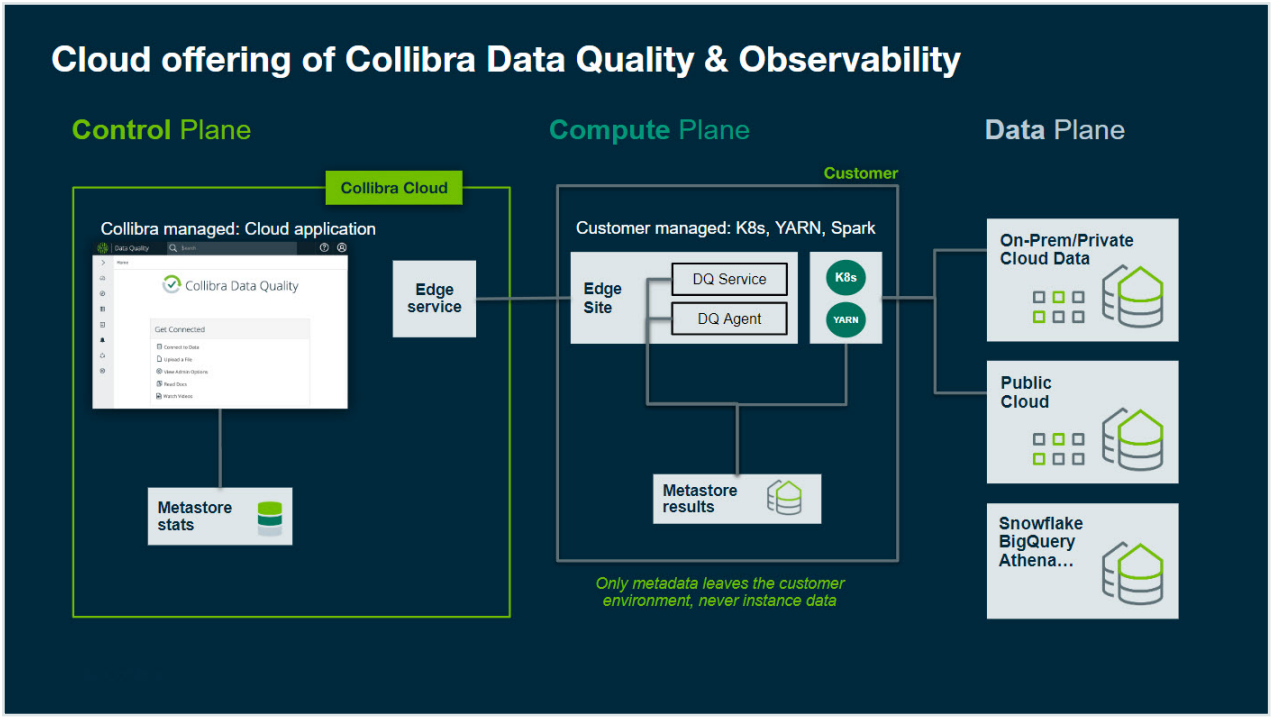
Requirements

To use Collibra DQ Cloud, you must ensure that the following system requirements are met.

Resource	Notes	Provisioned by
Collibra DQ	Version 2022.02 or newer with Edge mode enabled.	Collibra
Collibra Edge Site	Version 2022.02 or newer.	Customer
Postgres	Version 11 or newer.	Customer

Diagram

The following diagram shows the role of Edge and how it connects to Collibra DQ. Edge connects to your data sources and then provides data from them back to Collibra DQ, where you can scan and profile your data for quality issues.



Prerequisites

You need the following minimum hardware and software requirements:

VM

This is where your Edge site is installed. You need:

- RedHat 8 or CentOS 8
- SSH access
- 55 GB of free storage
  Verification command...
  Run the following command to verify:

```
df -B G
```

- 64 GB memory
  Verification command...
  Run the following command to verify:

```
free -m
```

- 16 cores
  Verification command...
  Run the following command to verify:

```
cat /proc/cpuinfo
```

- Egress (outbound) network access on port 443
  Verification command...
  Run the following command to verify:

```
wget https://google.com
```

- Network access to Postgres installed in step 2
  Verification command...
  Run the following command to verify:

```
telnet 10.64.2.3 5432
```

> Note For medium to large workloads of more than 100M rows by 100 columns, we recommend that your VM has a minimum of 32 cores, 128 GB memory, and 500 GB of free storage.

Edge installation requirements can be found **here**.

Postgres

This is where your DQ Job results are stored. You need:

- Version 11 or newer
- A minimum of 100 GB of free storage
- A minimum of 4 cores

- Network access to and from the VM where Edge is installed
- User with ownership rights over the target database

Steps

1. Obtain a Secure Collibra DQ Web URL

This is provisioned by Collibra. Along with the URL, credentials will be provided to access your instance.

2. Install Postgres

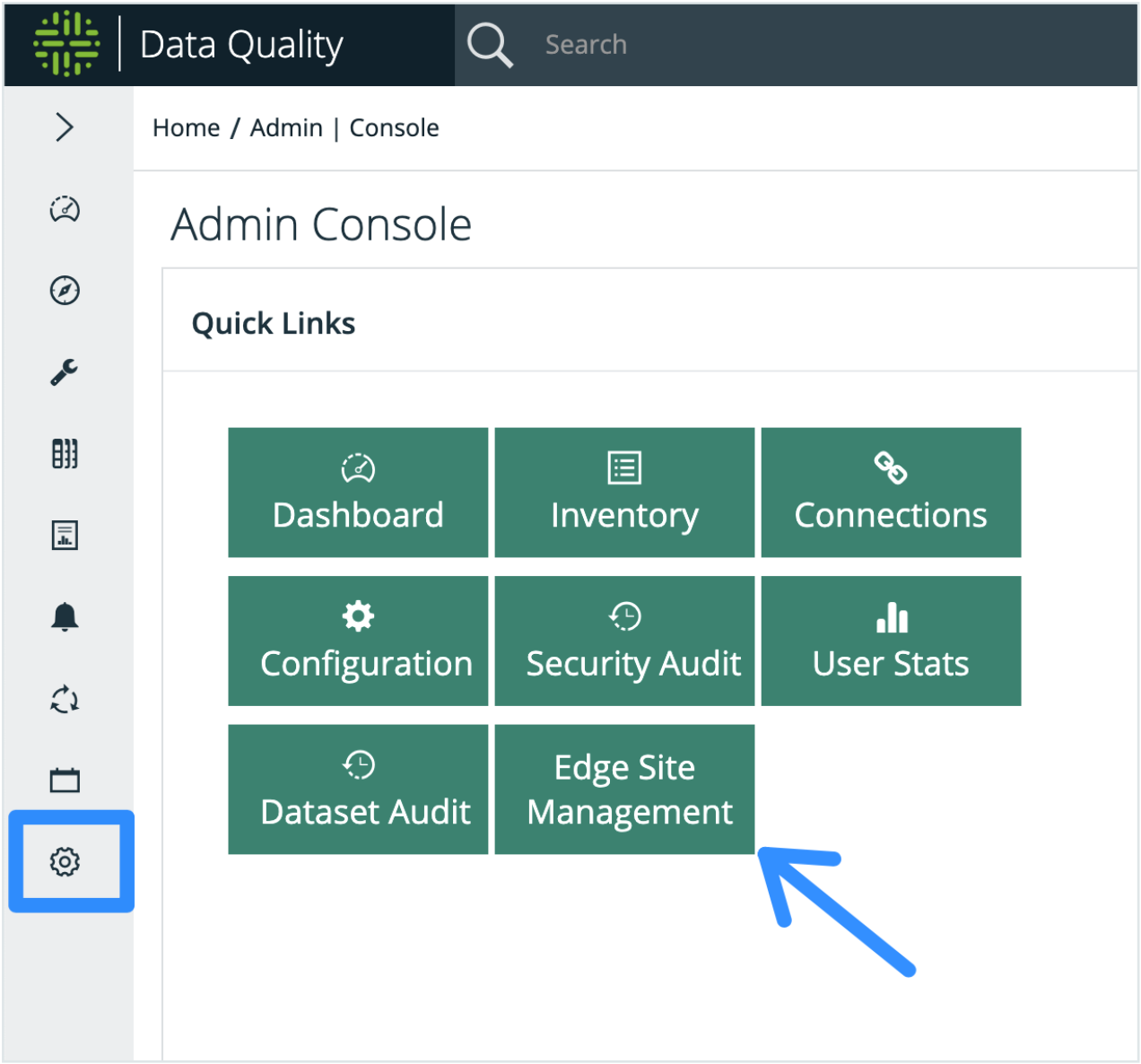
This is provisioned by the customer. There are several ways to install Postgres. You should follow your existing company process to provision a Postgres instance, such as RDS, Azure SQL, Cloud SQL, or standard install with a package manager. Ensure your version is 11 or newer.

> Important Remember your Postgres IP and login credentials. This is required when deploying the Edge site.

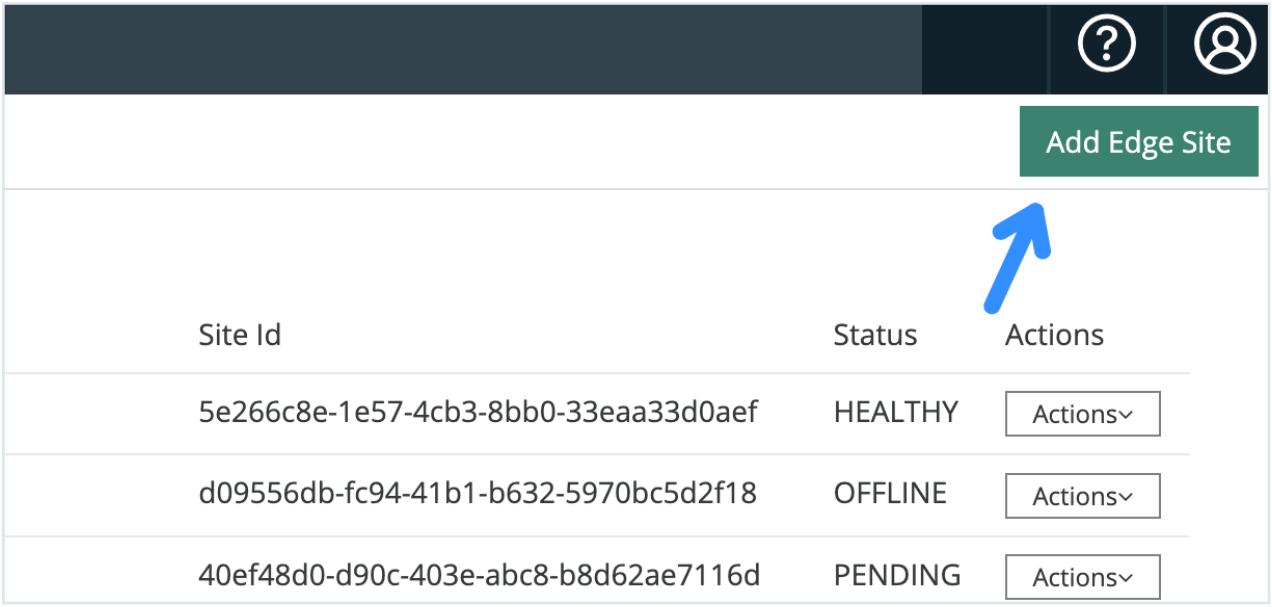
3. Install Edge

Refer to Edge documentation for system requirements.

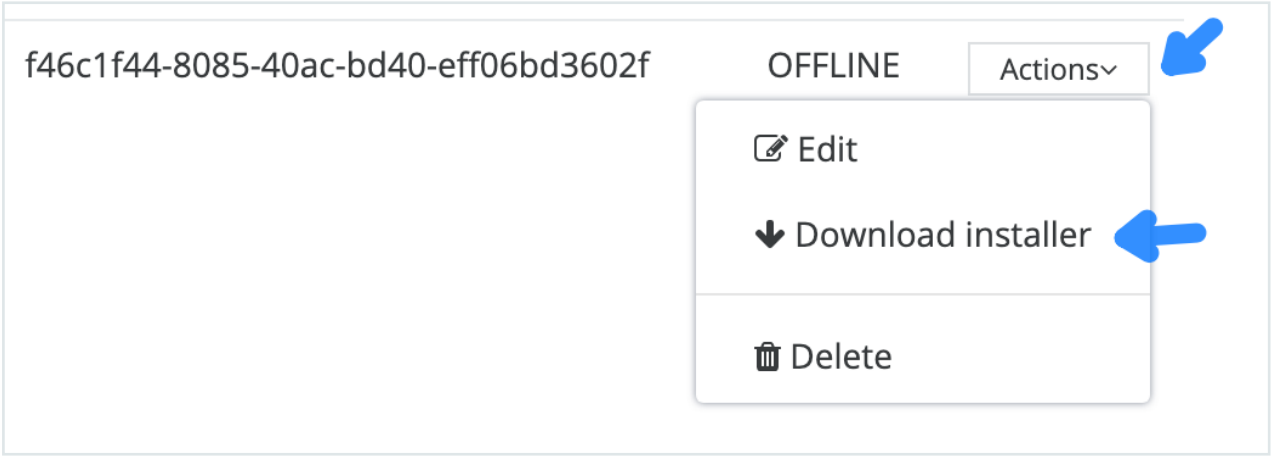
Navigate to the Edge Site Management panel in the Admin Console



Add an Edge Site and provide a name and description



Using the Actions drop-down, download the Edge installer package locally



> Warning Because connections must have an exact relationship between the Edge site and the datasource hostname, do not delete your Edge Site from the Edge Site Management page.

Upload the Edge installer package to your VM that meets the Edge system requirements above. An example scp command is below, but you can do this several ways.

```
scp -i ~/Downloads/vm-key.pem ~/Downloads/<installer>.tgz user-
@<host-or-ip>:/home/user/<installer>.tgz
```

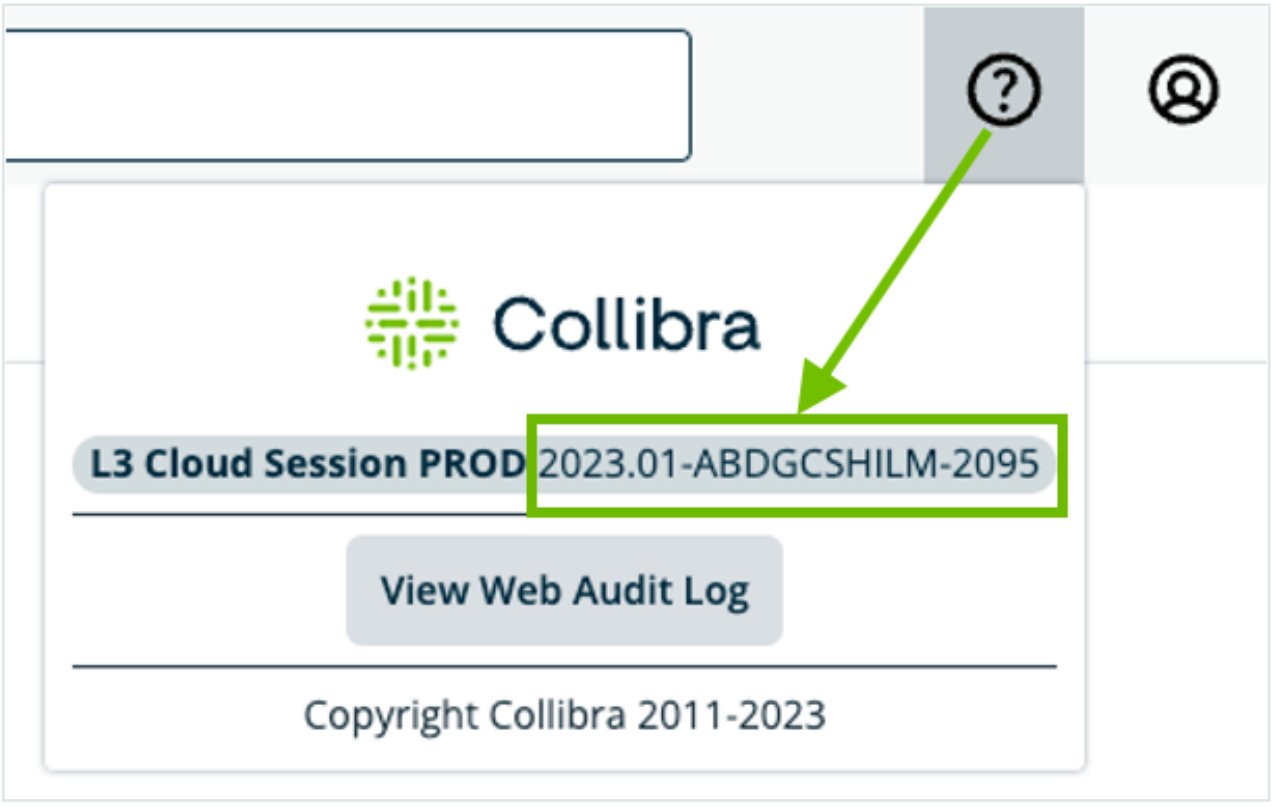
SSH to your VM after uploading the installer package. Untar the .tgz

```
tar -xvf <installer>.tgz
```

Install prerequisite Edge packages.

```
sudo yum install -y container-selinux selinux-policy-base

sudo yum install -y sudo yum install -y https://rp-
m.rancher.io/k3s/stable/common/centos/8/noarch/k3s-selinux-
1.2.2.el8.noarch.rpm

sudo firewall-cmd --zone=trusted --add-interface=lo --permanent

sudo firewall-cmd --zone=trusted --add-interface=cni0 --per-
manent

sudo firewall-cmd --reload
```

Confirm you have the correct Collibra DQ version pointer from your Cloud instance, for example **2023.01-ABDGCSHILM-2095**.



Remember your Postgres IP and credentials from the previous step.

Install Edge w/ DQ w/ the correct parameters

```
sudo /home/centos/install-master.sh --storage-path /var/edge
properties.yaml -r registries.yaml --set collibra_edge.-
collibra.dq.enabled=true,collibra_edge.-
collibra.dq.targetRevision=2023.01-ABDGCSHILM-2095,collibra_
edge.collibra.dq.sparkVersion=3.2.2,collibra_edge.-
collibra.dq.metastoreUrl=jdbc:postgresql://<your-postgres-
ip>:5432/postgres,collibra_edge.collibra.dq.metastoreUser=<your-
postgres-user>,collibra_edge.collibra.dq.metastorePass=<your-
postgres-password>
```

The snippet below is the same as the code block above.

The bold sections are the areas you will edit

sudo /home/**<your-directory>**/install-master.sh --storage-path /var/edge properties.yaml -r registries.yaml --set collibra_edge.collibra.dq.enabled=true,collibra_ edge.collibra.dq.targetRevision=**<version>**,collibra_ edge.collibra.dq.sparkVersion=**<spark-version>**,collibra_ edge.collibra.dq.metastoreUrl=jdbc:postgresql://**<postgres-ip>**:**<postgres-port>**/**<postgres_database_name>**,collibra_edge.collibra.dq.metastoreUser=**<postgres-user>**,collibra_edge.collibra.dq.metastorePass=**<postgres-password>**

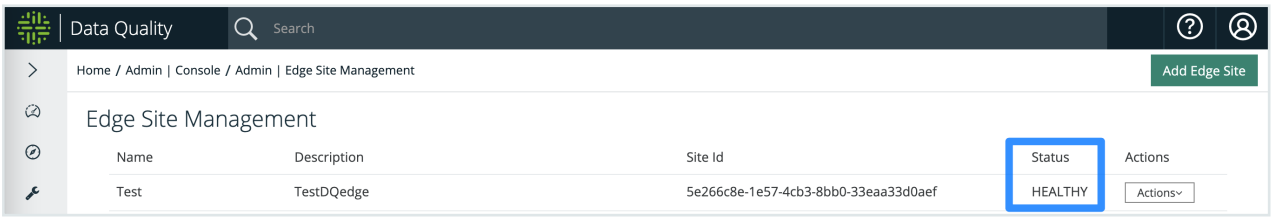
> Warning The $ symbol is not a supported special character in your Postgres Metastore password.

Check that all the processes are running / completed

```
sudo /usr/local/bin/kubectl get pods --all-namespaces
```

```
[centos@test-matt-mearns-edge ~]$ sudo /usr/local/bin/kubectl get pods --all-namespaces
NAMESPACE             NAME                                                READY   STATUS      RESTARTS   AGE
kube-system           local-path-provisioner-5ff76fc89d-b8vhj             1/1     Running     0          48m
kube-system           coredns-6488c6fcc6-vrwsd                            1/1     Running     0          48m
kube-system           metrics-server-86cbb8457f-z25lc                     1/1     Running     0          48m
collibra-edge         argo-cd-argocd-redis-648c968585-hzhlk               1/1     Running     0          47m
collibra-edge         argo-cd-argocd-application-controller-656bc485c9-w8jx6   1/1     Running     0          47m
collibra-edge         argo-cd-argocd-repo-server-5f796d76f4-ftrmx          1/1     Running     0          47m
collibra-edge         argo-cd-argocd-server-6bd4b4c7f7-8h62m               1/1     Running     0          47m
collibra-edge         collector-metrics-5989b8fc69-vgtm8                  1/1     Running     0          46m
collibra-edge         eventrouter-edge-eventrouter-58cf94d7f6-8c8gh       1/1     Running     0          46m
collibra-edge         metricscrape-fb6b6499c-jj2bm                        1/1     Running     0          46m
collibra-edge         agent-vkzd9                                         1/1     Running     0          46m
collibra-edge         signalfx-edge-signalfx-kvtcc                        1/1     Running     0          46m
collibra-edge         workflows-workflow-controller-6c9886bcdf-f7rv7      1/1     Running     0          45m
collibra-edge         minio-3                                             1/1     Running     0          45m
collibra-edge         minio-2                                             1/1     Running     0          45m
collibra-edge         minio-0                                             1/1     Running     0          45m
collibra-edge         minio-1                                             1/1     Running     0          45m
collibra-edge         workflows-server-5fcdbd6455-s59pl                   1/1     Running     0          45m
collibra-edge         collibra-edge-proxy-846dd4bdc5-9ssqj                1/1     Running     0          46m
collibra-edge         edge-agent-964pn                                    1/1     Running     0          45m
collibra-edge         owldq-owl-agent-owldq-96beafd3-0                    1/1     Running     0          45m
collibra-edge         datadog-54r7v                                       1/1     Running     0          45m
collibra-edge         owldq-owl-web-owldq-96beafd3-0                      1/1     Running     0          45m
collibra-edge         collibra-edge-controller-675d797cc8-rb458           1/1     Running     1          46m
edge-kube-installer   edge-kube-installer-job-2xgh5-fqvcc                 0/1     Completed   0          48m
```

Your Edge site will appear as HEALTHY upon successful installation



Uninstall Edge if there were mistakes or typos in the process

```
sudo /usr/local/bin/uninstall-edge.sh --force
```

> Note Do not delete an Edge Site from the UI. You can safely uninstall an Edge Site with this command.

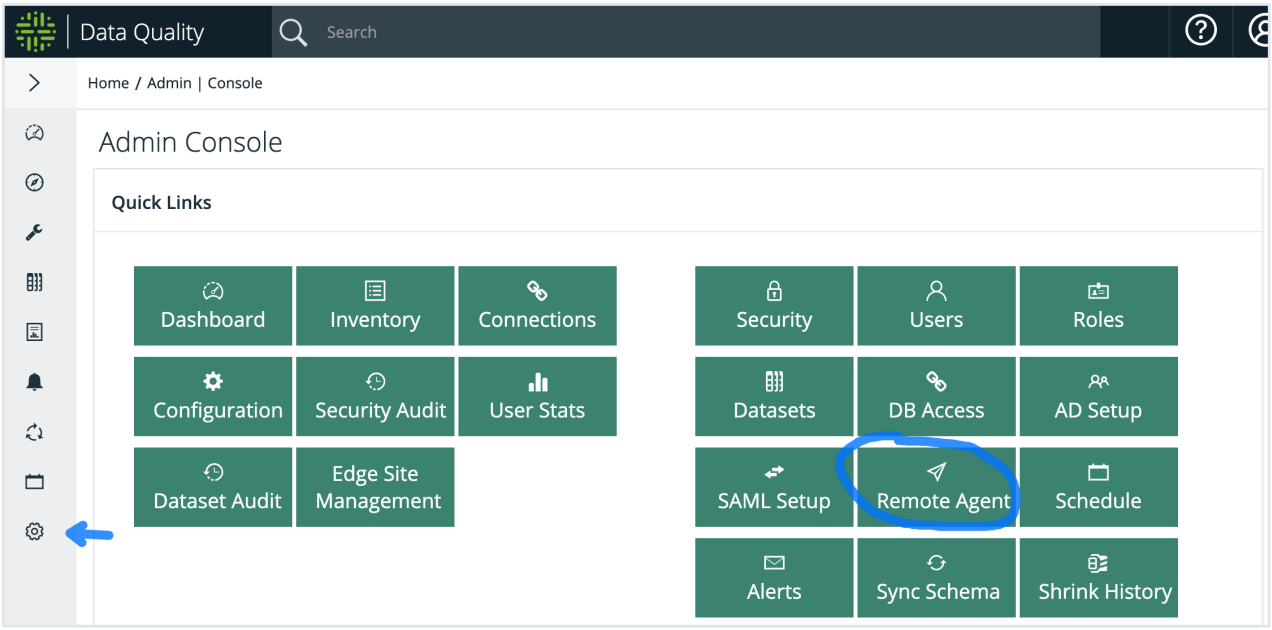
Reinstall the prerequisites if you perform the uninstall

```
sudo yum localinstall --skip-broken -y https://rp-
m.rancher.io/k3s/stable/common/centos/8/noarch/k3s-selinux-
1.2.2.el8.noarch.rpm
```

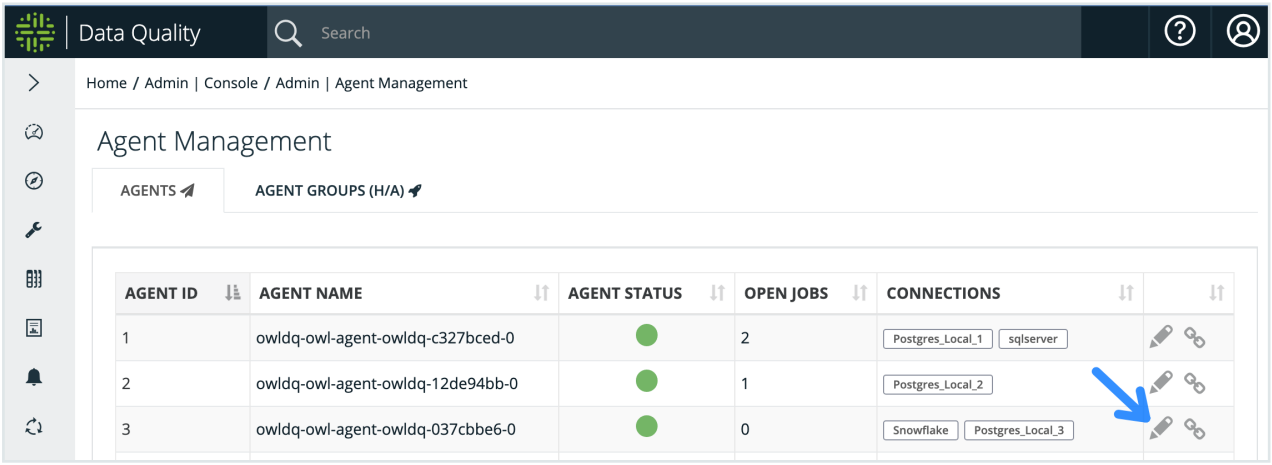
> Warning **To avoid orphaned records, do not delete an Edge using the UI.**

4. Configure an Agent

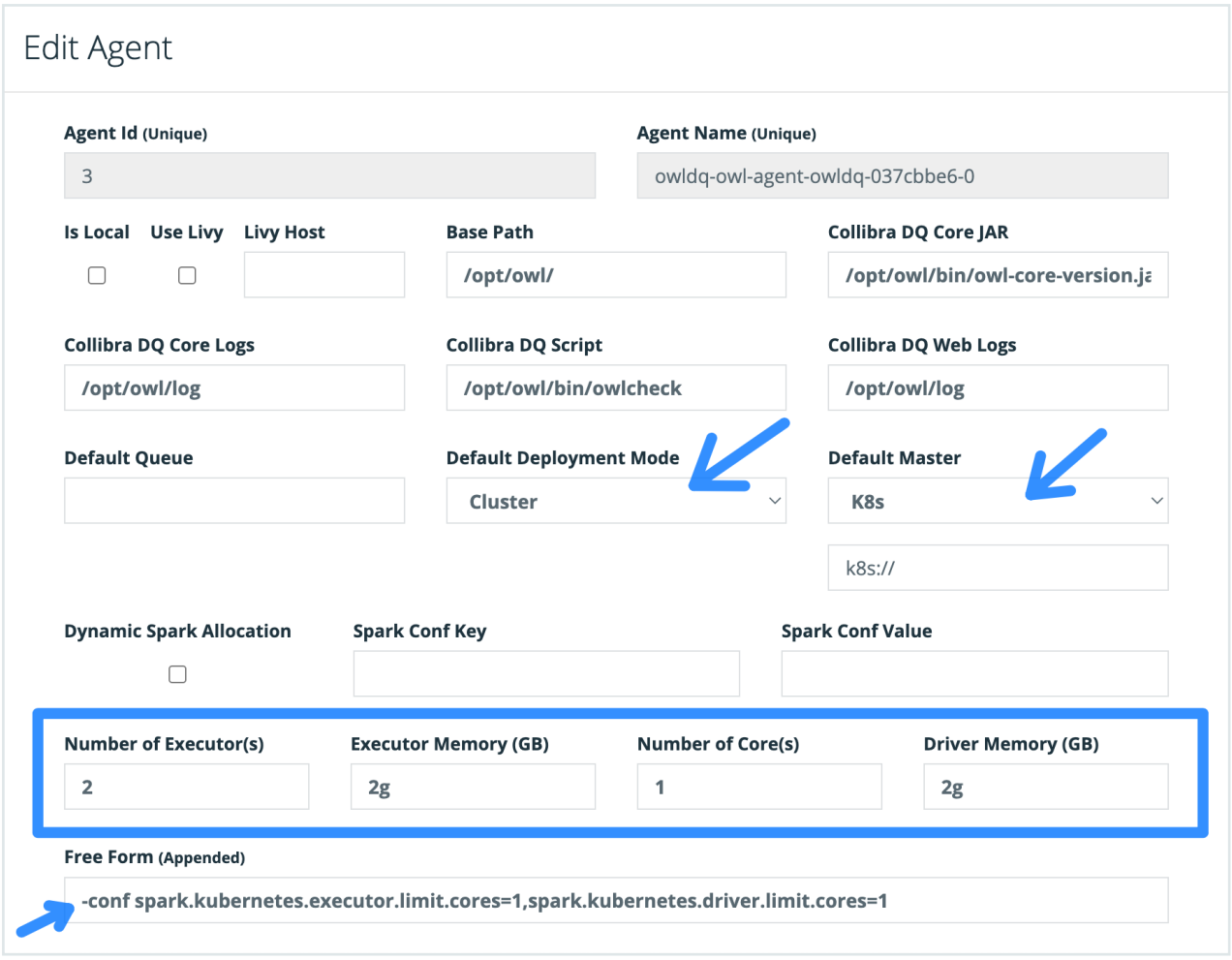
Go to the Remote Agent panel in the admin console



Upon completion of the Edge installation, you'll find an agent available from each respective Edge Site. Click the pencil icon to configure the agent.



Change the Default Deploy Mode to Cluster, the Default Masters to K8s and input defaults for resource assignment. Also add freeform append Spark confs as shown here.



Use the spark confs in the code block below.

```
-conf spark.kuber-
netes.ex-
ecutor.limit.cores=1,spark.kubernetes.driver.limit.cores=1
```

> Note The DQ Job (Spark) compute will take place locally on Edge K3s. Increase the size of your VM to vertically scale for more resources (.e.g. 32 cores, RAM, etc.). This is the preferred option in beta. Hadoop compute is supported if customer chooses that path and uses their Dataproc or EMR cluster.

> Note Make note of the agent name that as created. In the following step you will create a connection and select (link) the agent to your connection.

> Warning **Do not delete an Agent from the UI, to avoid any orphaned records.**

5. Set Job Limits

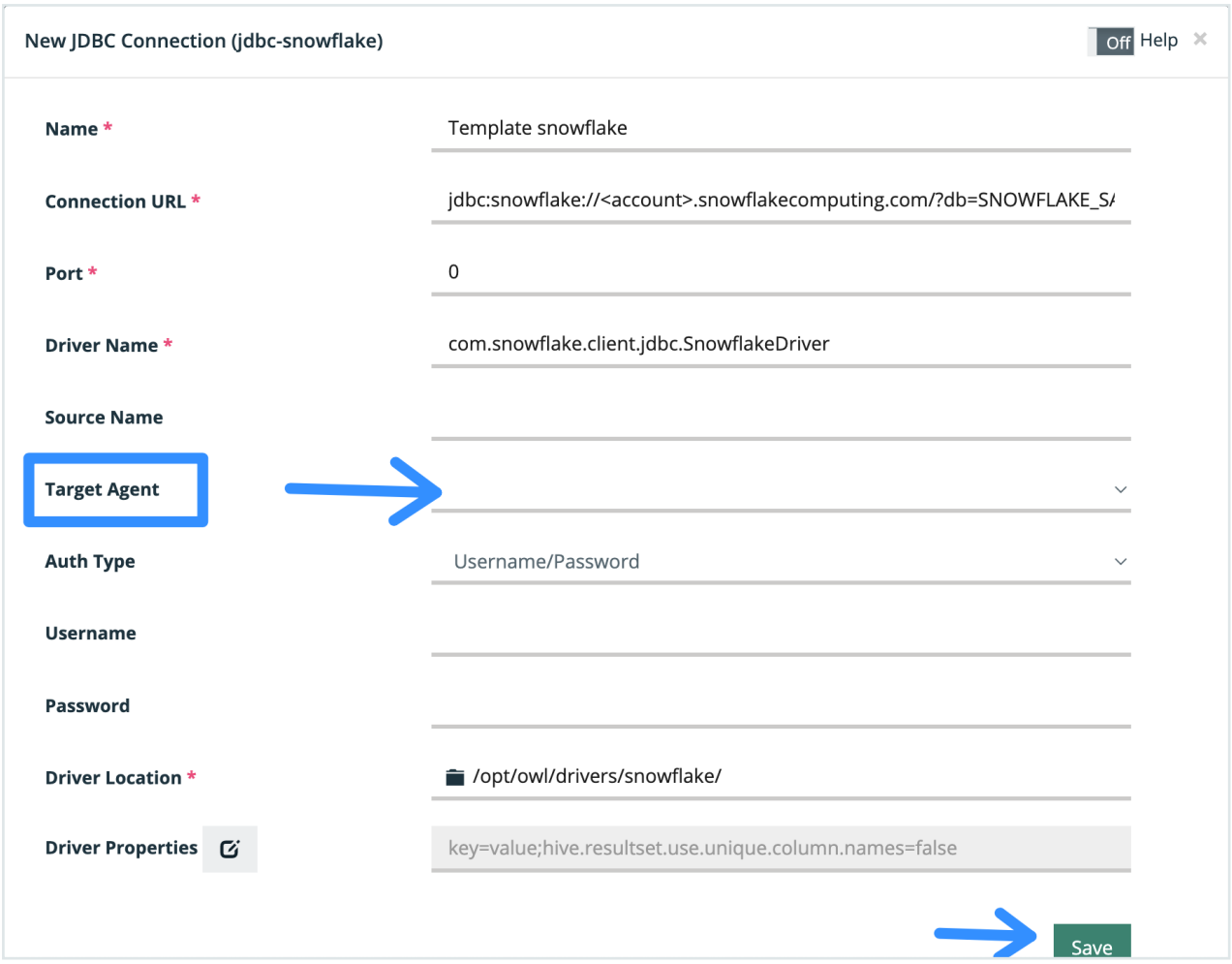
Set max cores to 1 in the job limit settings.

Refer to this link for configuring job limits.

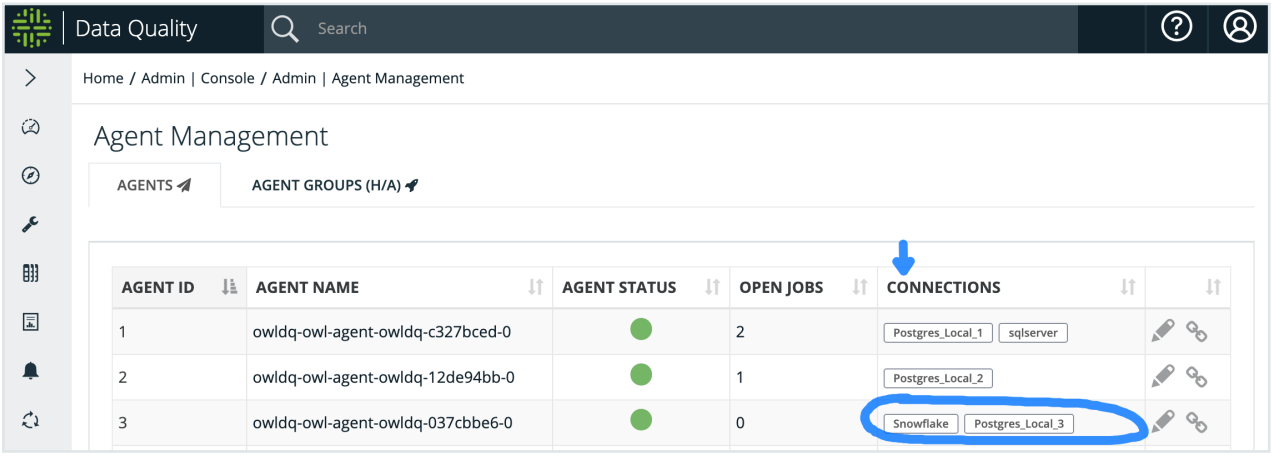
6. Add a Connection

This is the same process of adding a connection found Adding Connectionswith one difference. You will map the connection to your agent upon establishing a connection. **This is different than mapping a connection and an agent in the self-hosted application.**

Select your target agent using the **Target Agent drop-down**. This drop-down will populate with existing agents. Here is where you will select the agent name from the previous step.



Afterwards, you do not need to assign the connection to the agent. It will be automatically mapped.



> Note **To map a connection to another agent, you need to re-save the connection and select another agent from the drop-down list.**

7. Run a DQ Job

Run a DQ Job to validate the installation. Use the Explorer to onboard a table and check the Jobs page as normal to see the status.

> Note If the DQ Job does not succeed, please check your Agent settings and system prerequisites

Notes

Edge Capability Resource Requirements: If insufficient resources, your capabilities will not perform properly.

Installer: Please beware, downloading new installer will invalidate previous installer.

Volume: /var/lib/rancher/k3s path must have 50gb available

Root access: root access is needed, though future revisions will follow the least privileged user access policies.

The private beta is designed to let customers 1) complete the installation 2) confirm successful DQ jobs can be run and 3) validate their security requirements whereby no sensitive data is stored outside their custody.

Helpful Commands

```
# Get all pods running
sudo /usr/local/bin/kubectl get pods --all-namespaces

# Get shell access to pod
sudo /usr/local/bin/kubectl exec -it <dq-web-pod> -n collibra-
edge -- bash

# Get shell access to pod
sudo /usr/local/bin/kubectl exec -it collibra-edge-controller-
<pod-name> -n collibra-edge -- sh

# Check network connectivity to database
curl telnet://<rds-host>:<port>

# Delete jobs
sudo /usr/local/bin/kubectl delete pod <pod-name> -n collibra-
edge
```

FAQ

What network access is needed?

- The Edge Site and Postgres need to communicate with each other.
- Additionally, logging and heartbeat requires outbound access to several services. Please refer to Edge documentation for specific services that are used.

How can a user check the install?

- Time: The install should complete in around ~5 minutes; if not, there is likely an issue.
- Check that the pods
- `sudo /usr/local/bin/kubectl get pods --all-namespaces`

Is there a way to get more checks / more logs?

- `sudo /usr/local/bin kubectl describe`

How to verify successful install?

- In your Collibra DQ instance, navigate to the Edge Site Management panel in the Admin Console and confirm a HEALTHY status
- Support can confirm via Datadog, the edge site will send heartbeats

How to locate my Edge site in Datdog?

- Send your Edge Site ID to Support to check the health status.

Do customers have access to Datadog?

- Only Collibra has access to Datadog logging.

Can all my Collibra DQ and other capabilities run on the same Edge Site?

- There are not technical reasons preventing other capabilities and Collibra DQ from running on the same Edge Site.
- The guidance for the beta is to have DQ Edge separate from DGC Edge capabilities and simply use two Edge sites.

Are there any limitations with Collibra DQ Cloud in terms of features or functionality?

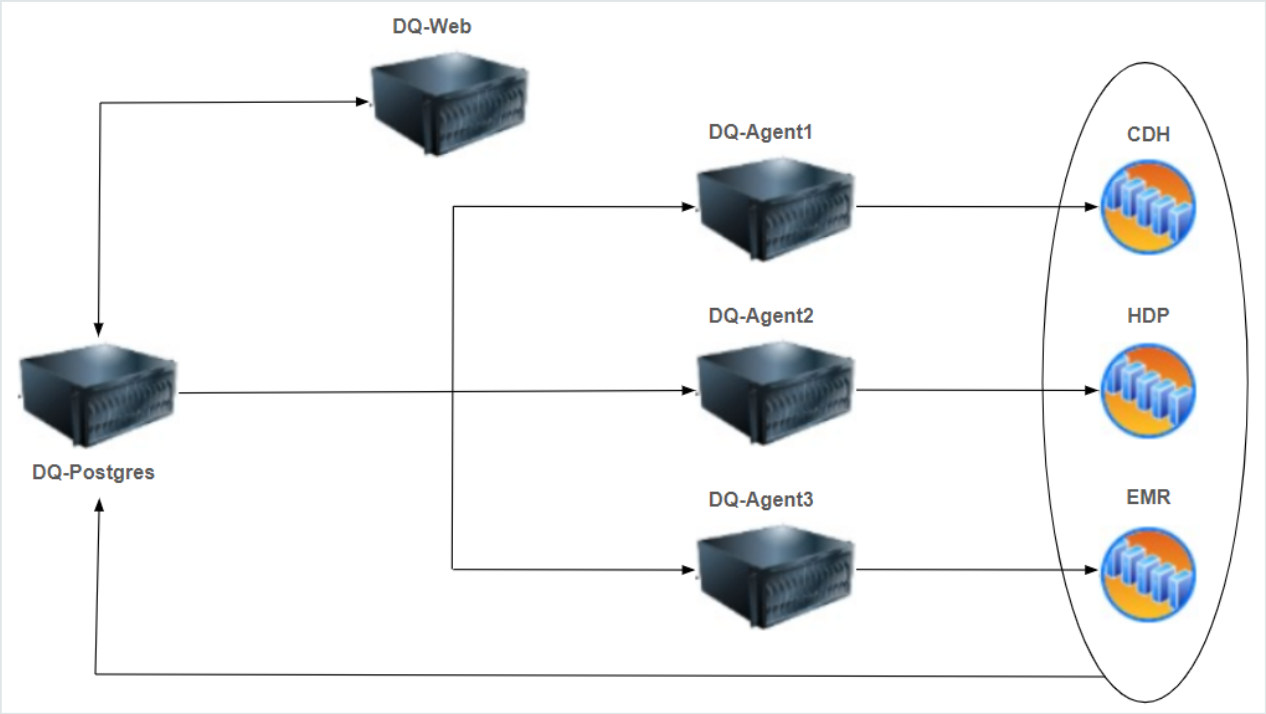
- While remote files are supported, local files and uploaded files are not supported due to security restrictions
- Specific drivers are not available in the beta, though the most common data sources are available.

What are the benefits of installing with Edge vs. a stand-alone, self-hosted application?

- The primary benefits are managed upgrades, maintenance, and reducing the ownership costs of an entirely self-hosted set of components.
- In addition, this design allows customers to take advantage of containers and cloud technologies without deep technical skillset requirements.
- This installation pattern was intentionally developed to not compromise any security requirements and give the customer complete custody of their data.
- Lastly, this aligns the Collibra architecture standards so support and services teams will benefit from normalized deployment models. In particular, when it comes to installation, configuration, and troubleshooting.

Agent

Diagram



The diagram above provides a high-level overview of how agents work within Collibra DQ. Job execution is driven by DQ Jobs that are written to an `agent_q` table inside the DQ Metastore (**DQ-Postgres**) via the Web App or REST API endpoint. Each active and available agent queries the DQ-Postgres table every 5 seconds to execute DQ Jobs for which the agent is responsible. For example, the EMR agent **DQ-Agent3** only executes DQ Jobs scheduled to run on EMR.

When an agent picks up a DQ Job, it launches the job either locally on the agent node itself or on a cluster as a Spark job (if the agent is set up as an edge node of the cluster). Depending on where the job launches, the results of the DQ Job will write back to the DQ Metastore. The results then display in the DQ Web App, are exposed as REST API, and become available for direct SQL query against the DQ Metastore.

Setting up a DQ Agent with `setup.sh` as part of the DQ package

Use the `setup.sh` script located in `/opt/owl/` (or other **Base Path** that your installation used). See the example in the code block below for setting up a DQ Agent with a Postgres server running `localhost` on port `5432` with database postgres and Postgres username/password combo `postgres/password`.

```
# PATH TO DIR THAT CONTAINS THE INSTALL DIR
export BASE_PATH=/opt

# PATH TO AGENT INSTALL DIR
export INSTALL_PATH=/opt/owl

# DQ Metadata Postgres Storage settings
export METASTORE_HOST=localhost
export METASTORE_PORT=5432
export METASTORE_DB=postgres
export METASTORE_USER=postgres
export METASTORE_PASSWORD=password

cd $INSTALL_PATH

# Install DQ Agent only
./setup.sh \
    -owlbase=$BASE_PATH \
    -options=owlagent \
    -pguser=$METASTORE_USER \
    -pgpassword=$METASTORE_PASSWORD \
    -pgserver=${METASTORE_HOST}:${METASTORE_PORT}/${METASTORE_
DB}
```

The setup script automatically generates the `/opt/owl/config/owl.properties` file and encrypts the provided password.

Setting up a DQ Agent manually

Steps

1. Open a terminal session and go to the directory with the installer.
2. Run the following command to encrypt your DQ Metastore password before it is stored in the `/opt/owl/config/owl.properties` file:

```
# PATH TO AGENT INSTALL DIR
export INSTALL_PATH=/opt/owl

cd $INSTALL_PATH

#Encrypt DQ Metadata Postgres Storage password
./owlmanage.sh encrypt=password
```

> Note `owlmanage.sh` generates an encrypted string for the plain text password input. You can use the encrypted string in the `/opt/owl/config/owl.properties` configuration file to avoid exposing the DQ Metadata Postgres Storage password.

3. Run the following command to open the `/opt/owl/config/owl.properties` configuration file:

```
vi $INSTALL_PATH/config/owl.properties
```

4. Add the following properties to the configuration file:

```
spring.datasource.url=jdbc:postgresql://{DB_HOST}:{DB_PORT}/
{METASTORE_DB}
spring.datasource.username={METASTORE_USER}
spring.datasource.password={METASTORE_PASSWORD}
spring.datasource.driver-class-name-
e=com.owl.org.postgresql.Driver

spring.agent.datasource.url=jdbc:postgresql://{DB_HOST}:{DB_
PORT}/{METASTORE_DB}
spring.agent.datasource.username={METASTORE_USER}
spring.agent.datasource.password={METASTORE_PASSWORD}
spring.agent.datasource.driver-class-name-
e=org.postgresql.Driver
```

5. Restart the DQ Web App.

Setting up the DQ Agent from the Admin Console

Steps

1. On the Collibra DQ home page, hover your cursor over ⚙ **Settings** and select **Admin Console**.
   » The Admin Console opens.
2. Click Remote Agent.
   » The **Agent Management** page opens.
3. In the last column of the Agents table, to the right, click the pencil icon to edit your agent.
   » The **Edit Agent** modal appears.
4. Enter the required information.

Field	Description
Agent Id	The numeric identifier of your agent. For example, *6*. This filed is pre-filled and cannot be edited.
Agent Name	The unique name of your agent. This field is pre-filled and cannot be edited.

Field	Description
Agent Display Name	The descriptive name of your agent that displays anywhere agent information is present in the DQ Web App. You can customize the Agent Display Name to make it easier to identify your agent. Tip There are no character restrictions for the Agent Display Name field, but it is best practice to use only alphanumeric characters, hyphens, and underscores.
Is Local	Select for Hadoop deployments only.
Is Livy	Deprecated. Not used.
Livy Host	The location where your Livy agent is hosted. This field is only applicable when Livy is in use.
Base Path	The installation folder path for DQ. All other paths in the DQ Agent are relative to this installation path. This is the location that is set as `OWL_BASE` in Full Standalone Setup and other installation setups followed by `owl/` folder. For example, if the setup command is `export OWL_BASE=/home/centos` then the **Base Path** in the Agent configuration should be set to `/home/centos/owl/`. Default: `/opt/owl/`.
Collibra DQ Core JAR	The file path to DQ Core jar file. Default `<Base Path>/owl/bin/`.
Collibra DQ Core Logs	The folder path where DQ Core logs are stored. Logs from DQ Jobs are stored in this folder. Default: `<Base Path>/owl/log`.
Collibra DQ Script	The file path to DQ execution script `owlcheck.sh`. This script is used to run DQ Job via command line without using agent. Using`owlcheck.sh`for running DQ Jobs is superseded by DQ Agent execution model. Default: `<Base Path>/owl/bin/owlcheck`.

Field	Description
Collibra DQ Web Logs	The folder path where DQ Web logs are stored. Logs from the DQ Web App are stored in this folder. Default: `<Base Path>/owl/log`.
Default Queue	The default resource queue for YARN.
Deploy Deployment Mode	The Spark deployment mode that takes one of `Client` or `Cluster`.
Default Master	The Spark Master URL copied from the Spark cluster verification screen. For example, `spark://...`.
Dynamic Spark Allocation	Deprecated. Not used.
Spark Conf Key	Deprecated. Not used.
Spark Conf Value	Deprecated. Not used.
Number of Executor(s)	The default number of executors allocated per DQ Job when using this Agent to run DQ Scans. The default is 1.
Executor Memory (GB)	The default RAM per executors allocated per DQ Job when using this Agent to run DQ Scans. The default is 1 gigabyte.
Number of Core(s)	The default number of cores per executors allocated per DQ Job when using this Agent to run DQ Scans. The default is 1.
Driver Memory (GB)	The default driver RAM allocated per DQ Job when using this Agent to run DQ Scans. The default is 1 gigabyte.
Free Form (Appended)	Other `spark-submit` parameters to append to each DQ Job when using this Agent to run DQ Scans.

5. Click **Save**.

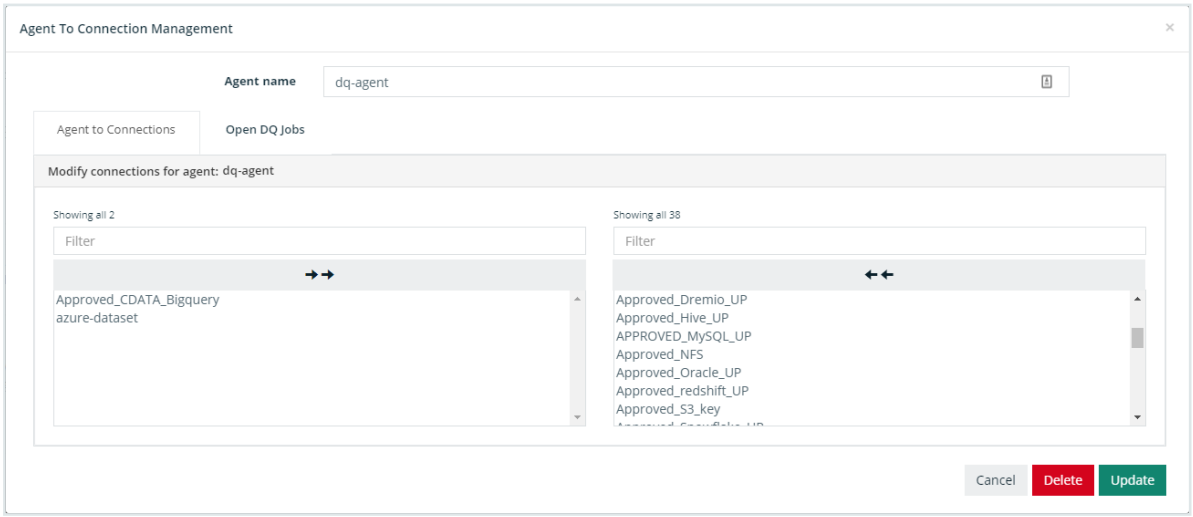
Linking data sources to the DQ Agent from the Admin Console

When you add new Data Sources, the DQ Agent requires permission to run DQ Jobs with them.

Steps

1. On the Collibra DQ home page, hover your cursor over ⚙ **Settings** and select **Admin Console**.
   » The Admin Console opens.
2. Click Remote Agent.
   » The **Agent Management** page opens.
3. In the last column of the Agents table, to the right, click the chain link icon to link your agent to data source connections.
   » The **Agent to Connection Management** wizard appears.

> Note The left panel contains a list of available connections that are not yet linked to the DQ Agent and do not yet have permission to run DQ Jobs. The right panel contains a list of connections that are linked to the DQ Agent and have permission to run DQ Jobs.



4. Click a connection in the left panel to link connections one at a time or click the double arrow icon to link all available connections at the same time.
5. Click **Update**.

> Tip You can unlink connections with the same methods listed above, but click the connections listed in the right panel instead of the left. Successfully unlinked connections appear in the left panel.

Adding a connection to a DQ Agent

Chapter 6

Collibra DQ Connections

Supported Connections

This page is a list of supported data source connection types. A supported data source is a data source that is shipped with the images or standalone bundles, and thus, eligible for support from the Collibra DQ team.

> Note Any data source that is compatible with the Java version and server to which you are connected can be used. However, if an issue occurs with an unsupported data source, we cannot guarantee support.

Production

The following is a list of drivers certified for production use.

Connections - Currently Supported

Connection	Certified	Tested	Packaged	Optionally Packaged	Push-down	Estimate job	Filtergram	Analyze Data	Schedule	Spark Agent	Yarn Agent	Parallel JDBC	Session State	Kerberos Password Manager	Kerberos Password Manager	Kerberos Keytab	Kerberos TGT	Standalone (non-Livy)	JDK8 Driver Compatibility	JDK11 Driver Compatibility
Athena	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Athena CDATA	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	No	Yes	Yes	Yes
BigQuery	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes

Connection	Certified	Tested	Packaged	Optionally Packaged	Pushdown	Estimate job	Filtergram	Analyze Data	Schedule	Spark Agent	Yarn Agent	Parallel JDBC	Session State	Kerberos Password	Kerberos Password Manager	Kerberos Keytab	Kerberos TGT	Standalone (non-Livy)	JDK8 Driver Compatibility	JDK11 Driver Compatibility
BigQuery CDATA	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	No	Yes	Yes	Yes
Databricks JDBC	Yes	Yes	No	Yes	No	No	No	No	No	No	No	No	No	No	No	No	No	No	Yes	Yes
Databricks CDATA	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	No	Yes	Yes	Yes

Connection	Certified	Tested	Packaged	Optionally Packaged	Pushdown	Estimate job	Filtergram	Analyze Data	Schedule	Spark Agent	Yarn Agent	Parallel JDBC	Session State	Kerberos Password	Kerberos Password Manager	Kerberos Keytab	Kerberos TGT	Standalone (non-Livy)	JDK8 Driver Compatibility	JDK11 Driver Compatibility
DB-2	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Dremio	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Hive	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hive CDATA	No	No	Yes	No	No	No	Yes	No	No	No	No	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes

Connection	Certified	Tested	Packaged	Optionally Packaged	Pushdown	Estimate job	Filtergram	Analyze Data	Schedule	Spark Agent	Yarn Agent	Parallel JDBC	Session State	Kerberos Password	Kerberos Password Manager	Kerberos Keytab	Kerberos TGT	Standalone (non-Livy)	JDK8 Driver Compatibility	JDK11 Driver Compatibility
Impala	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Impala CDATA	No	No	Yes	No	No	No	Yes	No	No	No	No	No	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Microsoft SQL	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	No

Connection	Certified	Tested	Packaged	Optionally Packaged	Pushdown	Estimate job	Filtergram	Analyze Data	Scheduler	Spark Agent	Yarn Agent	Parallel JDBC	Session State	Kerberos Password	Kerberos Password Manager	Kerberos Keytab	Kerberos TGT	Standalone (non-Livy)	JDK8 Driver Compatibility	JDK11 Driver Compatibility
MYSQL	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Oracle	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Postgres	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Presto	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes

Connection	Certified	Tested	Packaged	Optionally Packaged	Pushdown	Estimate job	Filtergram	Analyze Data	Schedule	Spark Agent	Yarn Agent	Parallel JDBC	Session State	Kerberos Password	Kerberos Password Manager	Kerberos Keytab	Kerberos TGT	Standalone (non-Livy)	JDK8 Driver Compatibility	JDK11 Driver Compatibility
Redshift	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Snowflake	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes
Sybase	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	Yes	Yes	Yes
Teradata	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes

> Tip A connection listed as Tested is one for which the Collibra DQ team has an environment and is included in regular regression testing.

> Note The Dremio connection is compatible with JDK11 if you add the following to owlmanage.sh as a JVM option for the web and Spark instance:
> `-Dcdjd.io.netty.tryReflectionSetAccessible=true`

Remote Connections - Currently Supported

Connection	Certified	Tested	Packaged	Optionally packaged	Push-down	Estimate job	Filtergram	Analyze data	Spark agent	Yarn agent
Azure Data Lake (Gen2)	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Google Cloud Storage	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes
HDFS	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes
S3	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes

Under Evaluation

The following is a list of drivers which are under evaluation (not certified yet for production usage). These connections are currently ineligible for escalated support services.

Connections - Tech Preview

Connection	Certified	Tested	Packaged	Optional packaging	Push-down	Estimate job	Filtergram	Analyze data	Schedule	Spark agent	Yarn agent	Parallel JDBC	Session state	Kerberos state	Kerberos password manager	Kerberos keytab	Kerberos T-GT	Standalone (non-Livy)
Cassandra	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No
MongoDB	✗ No	✗ No	✗ No	✗ No	✗ No	✓ Yes	✗ No	✓ Yes	✓ Yes	✓ Yes	✓ Yes	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✓ Yes
MongoDB CDATA	✓ Yes	✓ Yes	✓ Yes	✗ No	✗ No	✓ Yes	✓ Yes	✓ Yes	✗ No	✓ Yes	✗ No	✓ Yes	✗ No	✗ No	✗ No	✗ No	✗ No	✓ Yes
SAP HANA	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No	✗ No

Connection	Certified	Tested	Packaged	Optional packaging	Push-down	Estimate job	Filtergram	Analyze data	Schedule	Spark agent	Yarn agent	Parallel JDBC	Session state	Kerberos state	Kerberos password manager	Kerberos keytab	Kerberos T-GT	Standalone (non-Livy)
Solr	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No

Streaming - Tech Preview

Connection	Certified	Tested	Packaged	Optional packaging	Push-down	Estimate job	Filtergram	Analyze data	Schedule	Spark agent	Yarn agent	Parallel JDBC	Session state	Kerberos password	Kerberos password manager	Kerberos T-GT	CRDB metastore	Standalone (non-Livy)
Kafka	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No	No

Files

File type	Supported
CSV (and all delimiters)	Yes
Parquet	Yes
AVRO	Yes
JSON	Yes
DELTA	Yes

Limitations

Authentication

- DQ Jobs that require Kerberos TGT are not supported on Spark Standalone or Local deployments
    - Recommended to submit jobs via Yarn or K8s

File Limitations

File Sizes

- Files with more than 250 columns supported in File Explorer, unless you have Livy enabled.
- Files larger than 5gb are not supported in File Explorer, unless you have Livy enabled.
- Smaller file sizes will allow for skip scanning and more efficient processing
- Advanced features like replay, scheduling, and historical lookbacks require a date signature in the folder of file path

S3

- Please ensure no spaces in S3 connection name
- Please remember to select 'Save Credentials' checkbox upon establishing connection
- Please point to **root** bucket, not sub folders

Local Files

- Local files can only be run using NO_AGENT default
- This is for quick testing, smaller files, and demonstration purposes.
- Local file scanning is not intended for large scale production use.

Livy

- Livy is only supported for K8s environments

Spark Engine Support

- MapR is EOL and MapR spark engine not supported to run Collibra DQ jobs.

Databricks

Please refer to this page for more details on Databricks support

The only supported Databricks spark submit option is to use a notebook to initiate the job (Scala and Pyspark options). This is intended for pipeline developers and users knowledgeable with Databricks and notebooks. This form factor is ideal for incorporating data quality within existing Spark ETL data flows. The results are still available for business users to consume. The configuration is not intended for business users to implement. There are three ways that Databricks users can run DQ jobs using Databricks cluster or JDBC connection. 1. **Notebook** Users can directly open a notebook, upload Collibra DQjars and run a DQ job on Databricks cluster. The full steps are explained in below page. Collibra DQsupports this flow in production.

https://dq-docs.collibra.com/apis-1/notebook/cdq-+-databricks

2. Spark-Submit

There are two ways to run a spark submit job on Databricks's cluster. The first approach is to run a DQ spark submit job using Databricks UI and the second approach is by invoking

Databricks rest API. We have tested both approaches against different cluster versions of DataBricks (See below table). Below is the full documentation to demonstrate these paths. https://dq-docs.collibra.com/apis-1/notebook/cdq-+-databricks/dq-databricks-submit\

Please note that these are only examples to demonstrate how to achieve DQ spark submit to Databricks's cluster. These paths are **not** supported in production and the Collibra DQ team does **not** support any bug coverages or professional services or customer questions for these flows. \

3. JDBC

Collibra DQ users can create JDBC connections in CDQ UI and connect to their Databricks database. This is scheduled for 2022.05 release.

> Warning Delta Lake and JDBC connectivity has been validated against Spark 3.01 Collibra DQ package, Databricks 7.3 LTS and SparkJDBC41.jar. This is available as Preview. No other combinations have been certified at this time.

> Warning Spark submit using the Databricks spark master url is not supported.

	Spark 2.45	Spark 3.01	Spark 3.12	Spark 3.2
Notebook	YES	YES	YES	YES
DataBricks via JDBC	YES	YES	YES	YES
Spark-Submit API	NO	NO	NO	NO

Connectivity to Athena

Your host can connect to Athena with either an Athena public service endpoint or an Athena private endpoint. For more information on setting the endpoint, see Command line options and Boto3 documentation.

JDBC URL Example

jdbc:awsathena://AwsRegion=us-east-1;User=xxx;Password=xxx;S3OutputLocation=s3://data-

bucket;**MetadataRetrievalMethod=Query**

- Athena uses port 443 to connect to the host.
- Athena's streaming API uses port 444 to stream the query results. When you use a JDBC/ODBC driver, Athena uses this port to stream the query results to the JDBC/ODBC driver installed on the client host. Therefore, unblock this port when you use a JDBC/ODBC driver to connect to Athena. If this port is blocked, your business intelligence tool might time out or fail to show query results when you run a query.
- Use the appropriate JDBC connection URLs in your business tool configuration according to your private DNS configuration for your endpoint.
    - Use the following connection string if you turned off the private DNS: jdbc:awsathena://vpce-.athena.us-east-1.vpce.amazonaws.com:443
    - Use the following connection string if you turned on the private DNS: jdbc:awsathena://athena.us-east-1.amazonaws.com:443
- Be sure that the security group attached to your VPC endpoint allows traffic from the host where you installed the JDBC/ODBC driver.
- Be sure that port 444 isn't blocked. If you use an AWS PrivateLink endpoint to connect to Athena, then be sure that the security group attached to the AWS PrivateLink endpoint is open to inbound traffic on port 444. Athena uses port 444 to stream query results. If port 444 is blocked, then the results aren't streamed back to your client host. In such situations, you might receive an error message similar to "[Simba][AthenaJDBC](100123) An error has occurred. Exception during column initialization". This can also cause the business intelligence tool to stop responding and not display the query results.

```
telnet athena.us-east-1.amazonaws.com 443
telnet glue.us-east-1.amazonaws.com 443
```

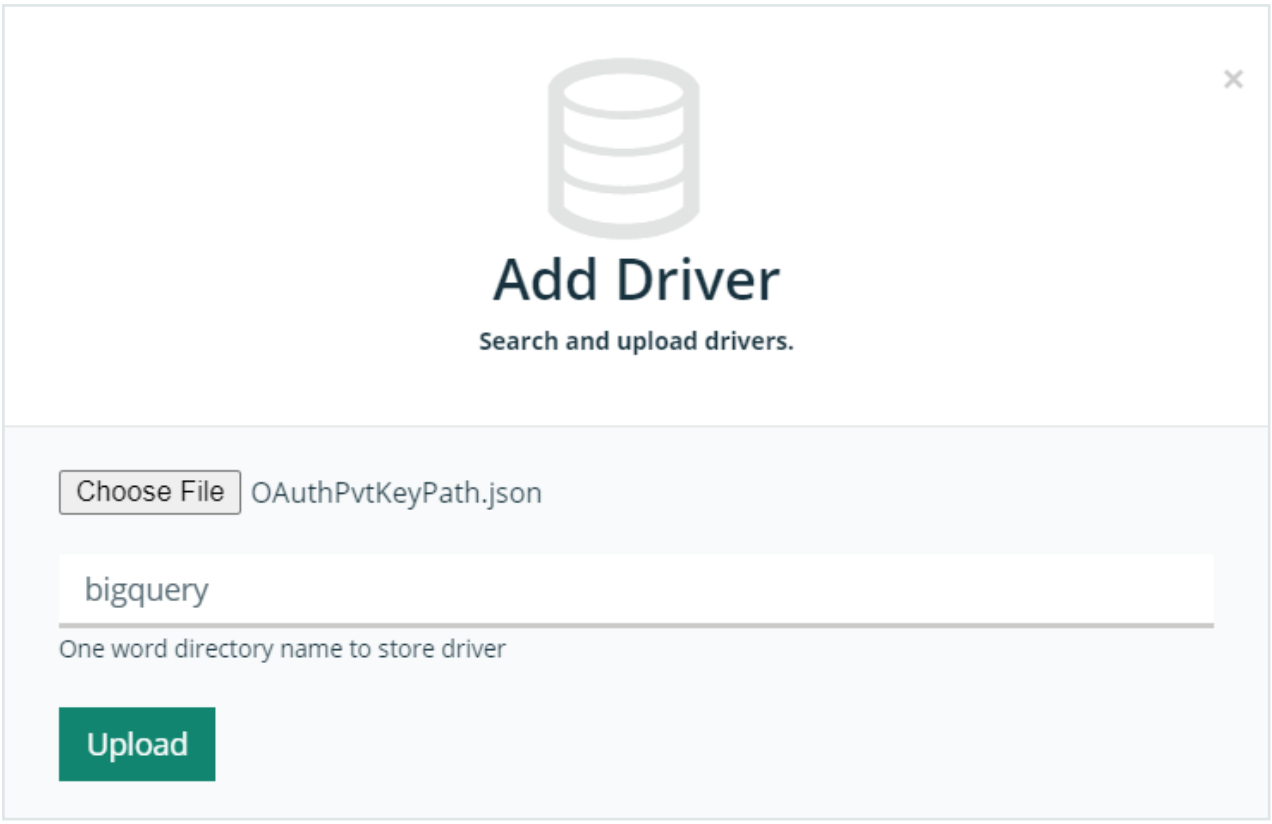
Minimum Permissions

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "athena:StartQueryExecution",
                "s3:ListBucketMultipartUploads",
                "athena:GetQueryResultsStream",
                "glue:GetTables",
                "glue:GetPartitions",
                "athena:GetQueryResults",
                "glue:BatchGetPartition",
                "s3:ListBucket",
                "glue:GetDatabases",
                "athena:ListQueryExecutions",
                "s3:ListMultipartUploadParts",
                "glue:GetTable",
                "glue:GetDatabase",
                "athena:GetWorkGroup",
                "s3:PutObject",
                "s3:GetObject",
                "glue:GetPartition",
                "glue:GetCatalogImportStatus",
                "athena:StopQueryExecution",
                "athena:GetQueryExecution",
                "s3:GetBucketLocation",
                "athena:BatchGetQueryExecution",
                "athena:DeletePreparedStatement",
                "athena:CreatePreparedStatement"
            ],
            "Resource": [
                "arn:aws:athena:*:<AWSAccountID>:workgroup/prima-
ry",
                "arn:aws:s3:::<S3 bucket name>/*",
                "arn:aws:s3:::<S3 bucket name>",
                "arn:aws:glue:*:<AWSAccountID>:catalog",
                "arn:aws:glue:*:<AWSAccountID>:database/<databas-
e name>",
                "arn:aws:glue:*:<AWSAccountID>:table/<database
name>/*"
            ]
        }
    ]
}
```

Connectivity to BigQuery

Steps for the BigQuery Connection

1. **Driver**: com.simba.googlebigquery.jdbc42.Driver
2. **Locate your service account owl-gcp.json** (your org auth key in JSON format)
3. **Create a JDBC connection** (for example only do not use this JDBC URL): jdbc:bigquery://https://www.googleapis.com/bigquery/v2:443;ProjectId=;OAuthType=0;OAuthServiceAcctEmail=<1234567890>-compute@developer.gserviceaccount.com;OAuthPvtKeyPath=/opt/ext/bq-gcp.json;Timeout=86400
4. **Requires a path to a JSON file** that contains the service account for authorization. That same file is provided to the Spark session to make a direct to storage connection for maximum parallelism once Core fires up."
    a. Helpful tip: This JSON file can be uploaded to your bigquery directory using the "add driver".



To succeed with the connection, you must follow these steps:

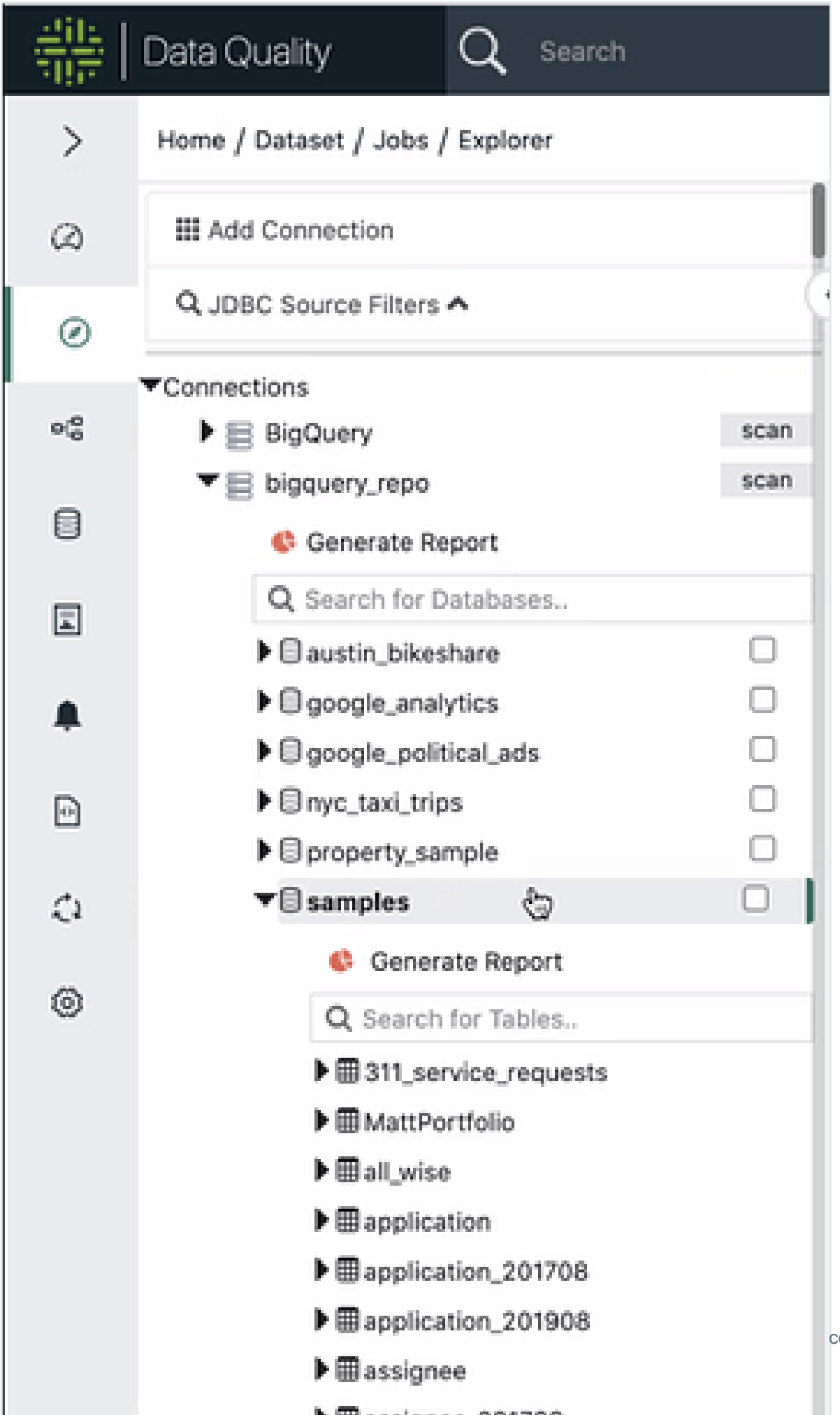
1. **Password for the BigQuery Connector form in Collibra DQ must be a base64 encoded string created from the json file (see step 3. above)** and input as password. For example: `base64 your_json.json -w 0` or `cat your_json.json | base64 -w 0`
2. **Check that this JAR exists and is on the path of the Collibra DQ Web UI server** (eg. <INSTALL_PATH>/owl/drivers/bigquery/core). Look at your driver directory location which contains this BigQuery JAR: spark-bigquery_2.12-0.18.1.jar
3. **Make sure these JARs present in <INSTALL_PATH>/owl/drivers/bigquery/:** ****animal-sniffer-annotations-1.19.jargoogle-api-services-bigquery-v2-rev20201030-1.30.10.jargrpc-google-cloud-bigquerystorage-v1beta1-0.106.4.jarlistenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jarannotations-4.1.1.4.jargoogle-auth-library-credentials-0.22.0.jargrpc-google-cloud-bigquerystorage-v1beta2-0.106.4.jaropencensus-api-0.24.0.jarapi-common-1.10.1.-jargoogle-auth-library-oauth2-http-0.22.0.jargrpc-grpclb-1.33.1.jaropencensus-contrib-http-util-0.24.0.jarauto-value-annotations-

1.7.4.jarGoogleBigQueryJDBC42.jargrpc-netty-shaded-1.33.1.jarperfmark-api-0.19.0.jaravro-1.10.0.jargoogle-cloud-bigquery-1.125.0.jargrpc-protobuf-1.33.1.jarprotobuf-java-3.13.0.jarchecker-compat-qual-2.5.5.jargoogle-cloud-bigquerystorage-1.6.4.jargrpc-protobuf-lite-1.33.1.jarprotobuf-java-util-3.13.0.jarcommons-codec-1.11.jargoogle-cloud-core-1.93.10.jargrpc-stub-1.33.1.jarproto-google-cloud-bigquerystorage-v1-1.6.4.jarcommons-compress-1.20.-jargoogle-cloud-core-http-1.93.10.jargson-2.8.6.jarproto-google-cloud-bigquerystorage-v1alpha2-0.106.4.jarcommons-lang3-3.5.jargoogle-http-client-1.38.0.jarguava-23.0.jarproto-google-cloud-bigquerystorage-v1beta1-0.106.4.jarcommons-logging-1.2.jargoogle-http-client-apache-v2-1.38.0.jarhttpclient-4.5.13.jarproto-google-cloud-bigquerystorage-v1beta2-0.106.4.jarconscrypt-openjdk-uber-2.5.1.jargoogle-http-client-appengine-1.38.0.jarhttpcore-4.4.13.jarproto-google-common-protos-2.0.1.jarcoregoogle-http-client-jackson2-1.38.0.jarj2objc-annotations-1.3.jarproto-google-iam-v1-1.0.3.jarerror_prone_annotations-2.4.0.jargoogle-oauth-client-1.31.1.jarjackson-annotations-2.11.0.jargrpc-alts-1.33.1.jarjackson-core-2.11.3.jarslf4j-api-1.7.30.jarfailureaccess-1.0.1.jargrpc-api-1.33.1.jarjackson-data-bind-2.11.0.jargax-1.60.0.jargrpc-auth-1.33.1.jarjavax.annotation-api-1.3.2.jarthreetenbp-1.5.0.jargax-grpc-1.60.0.jargrpc-context-1.33.1.jarjoda-time-2.10.1.jargax-httpjson-0.77.0.jargrpc-core-1.33.1.jarjson-20200518.jargoogle-api-client-1.31.1.jargrpc-google-cloud-bigquerystorage-v1-1.6.4.jarjsr305-3.0.2.jar

4. You may get a CLASSPATH conflict regarding the JAR files.

5. Make sure the BigQuery connector Scala version matches your Spark Scala version.

.


Data Quality
Search
Home / Dataset / Jobs / Explorer
Add Connection
JDBC Source Filters
Connections
BigQuery
scan
bigquery_repo
scan
Generate Report
Search for Databases..
austin_bikeshare
google_analytics
google_political_ads
nyc_taxi_trips
property_sample
samples
Generate Report
Search for Tables..
311_service_requests
MattPortfolio
all_wise
application
application_201708
application_201908
assignee

Networking

Please account for these urls from a networking and firewall perspective.

logging.googleapis.com

oauth2.googleapis.com

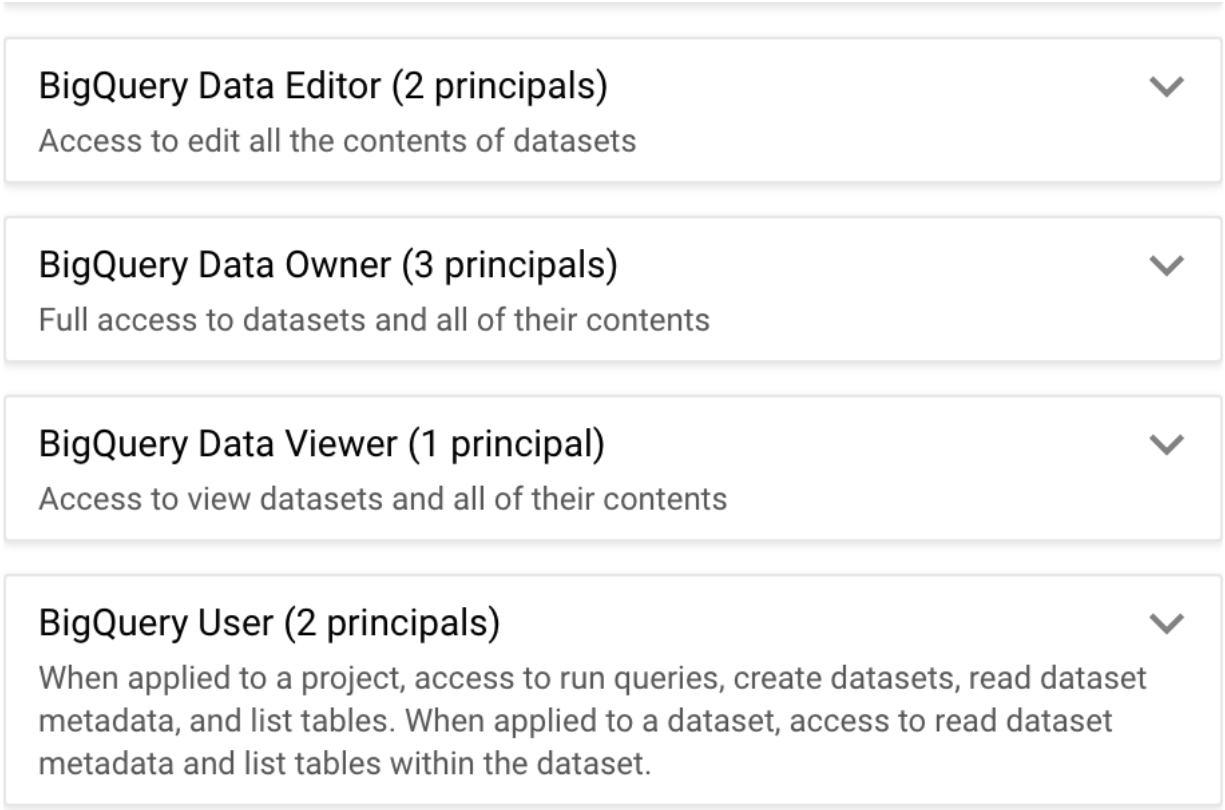
googleapis.com

bigquerystorage.googleapis.com

bigquery.googleapis.com

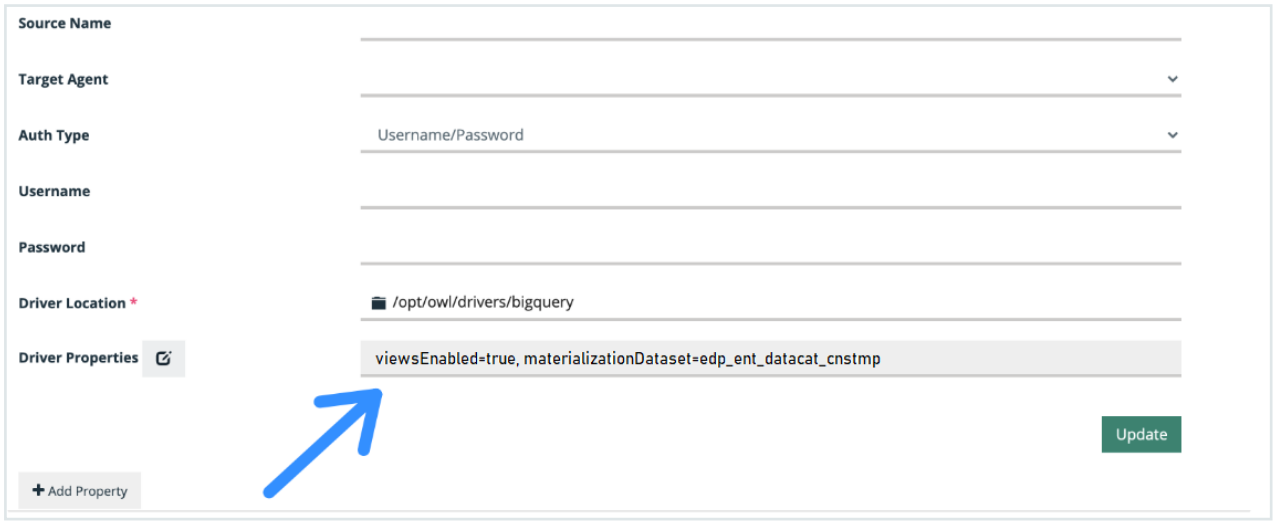
Permissions

Make sure the project and account have appropriate permissions. These are common permissions to provide to the account.



Views

Support for BigQuery views is available from the 2021.11 release onward. There are BigQuery limitations on creating views from different data sets (collections). Optionally, you can add the `viewsEnabled=true` parameter to the connection property when defining the connection.



> Note For read/write access to BigQuery, you can use the Spark BigQuery connector. To use this connector, ensure that the following configurations are set:
>
> `viewsEnabled` is set to true.
> `materializationDataset` is set to a data set where the GCP user has table creation permission.
> `materializationProject` is optional.

> Important The connector is implemented by running the query on BigQuery and the results are saved into a temporary table. Spark will then read the results, which could result in additional costs on your BigQuery account.

Spark Version 2

> Warning Be sure to use the Spark BigQuery connector that is compatible with your version of Spark.
>
> Also, when using Spark <3 and Scala 2.11, add the following props to the connection properties:

```
dq.bq.legacy=true,viewsEnabled=true
```

Connectivity to Databricks

There are three ways to utilize Databricks infrastructure with Collibra Data Quality:

- JDBC (Supported)
- Notebook/SDK (Supported)
- Spark Submit (Not Supported)

JDBC (Supported)

> Note As of May 2022, certification, support and optional packaging is available.

> Warning As of September 2022, Databricks JDBC driver version 2.6.27 is packaged as part of both standalone and Kubernetes download packages. The Databricks Simba driver (version 2.6.22) is no longer packaged for Kubernetes. As a result of this change, the Databricks connection template has changed, and any existing connection using the old driver (2.6.22) must be updated. For more information on updating your drivers, refer to Standalone Upgrade.

Connecting to a Databricks SQL warehouse

The following table shows the connection details for the required fields of a Databricks JDBC connection.

Property	Description
Name	The unique name used for your connection. For example, databricks-sql.
Connection URL	The connection string used to establish a connection to Databricks. This should adhere to the following format: jdbc:databricks://[Host]:[Port]/[Schema];[Property1]=[Value];[Property2]=[Value];... For example, jdbc:databricks://**<your-account-here>**.cloud.databrick-s.com:443/default;transportMode-e=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/warehouses/xxx;UID=token;PWD=<your-token-here>
Driver Name	The driver class name used for your connection. For example, com.databricks.client.jdbc.Driver
Port	The port used to establish a connection. The default is 0.

Connecting to Databricks Unity Catalog

> Note Connectivity to Databricks Unity Catalog is available in Collibra DQ versions 2023.01 or later.

Property	Description
Name	The unique name used for your connection. For example, databricks-unity-catalog.

Property	Description
Connection URL	The connection string used to establish a connection to Databricks. This should adhere to the following format and include the catalog name in the connection string: jdbc:databricks://[Host]:[Port]/[Schema];[Property1]=[Value]; [Property2]=[Value];...ConnCatalog=catalog_name For example, jdbc:databricks://**<your-account-here>**.cloud.databricks.com:443/default;transportMode=http;ssl=1;AuthMech=3;httpPath=/sql/1.0/warehouses/xxx;UID=token;PWD=xxxx;ConnCatalog=cdq
Driver Name	The driver class name used for your connection. For example, com.databricks.client.jdbc.Driver
Port	The port used to establish a connection. The default is 0.

Credentials

Property	Description
Username	The default value is **token**.
Password	The token you entered in the Connection URL. Tip To successfully establish a Databricks connection verify that your token is valid, because they generally expire in 90 days.

Jar

You can download the Databricks JDBC zip file by following one of the links below.

For the latest Databricks driver version, refer to the official Databricks JDBC Driver page.

For archived Databricks driver versions, refer to the official Databricks JDBC Driver archive.

> Note Databricks JDBC driver version 2.6.27 is packaged as part of both standalone and Kubernetes download packages.

Notebook (Supported)

- Pyspark SDK
- Scala SDK
- Collibra DQ + Databricks

Databricks no longer supports Runtime 6.5 or 10.3. Therefore, Collibra DQ Profile 2.45 is not runnable on Databricks.

https://docs.databricks.com/release-notes/runtime/10.3ml.html

The following table shows the latest supported versions of Collibra DQ Profiles and their matching Databricks Run times. (last updated: September 2022).

CDQ-DataBricks NoteBook status			
Databricks Cluster	7.3	9.1LTS	10.4
CDQ Profile	SPARK301	SPARK312	SPARK 320
Supported	YES	YES	YES

Spark Submit (Not Supported)

- Spark Master URL
- Databricks Jobs API
  - Rest
  - UI
  - DQ-Databricks Submit

> Note While these are not officially supported, there is a reference to architecture and implementation pattern for how to do a Databricks Job submission.

CDQ Connection to Databricks via JDBC

Make a Collibra DQ Connection to Databricks

Make a Collibra DQ connection to Databricks with the following steps.

> Note The Databricks Driver is packaged with the installation and can be found in the Driver folder.

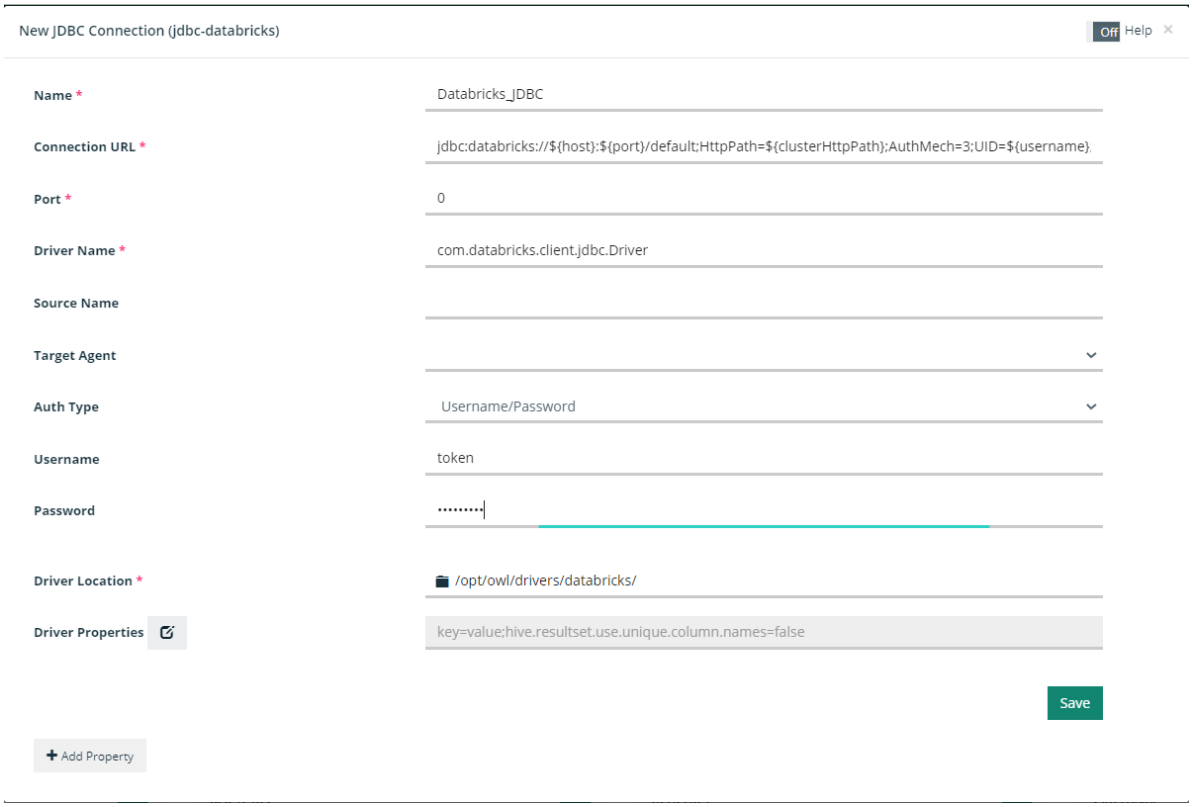
Steps

1. Login to your Collibra DQ instance.
2. Click the icon, then click **Connection**.
3. Locate the DATABRICKS driver DatabricksJDBC42.
4. Click **Add**.
5. Complete the following fields:
    - **Name** - Add a name for the connection (for example, Databricks_JDBC).
    - **Connection URL** - Enter the URL for the connection: jdbc:databricks://<your-account-here>..cloud.dat-abrick-s.com:443/de-fault;trans-portMode=http;ssl=1;httpPath=sql/protocolv1/o/3633393438801721/0915-195703-sh82m595;AuthMech=3;UID=token;PWD=<your-token-here>
    - **Port** - Enter the port for the connection.
    - **Driver Name** - Enter the name for the driver (for example, com.dat-abricks.client.jdbc.Driver)
    - **Auth Type** - Select the type of authorization from the drop-down list (for example, Username/Password).
        - **Username** - Username is **token**.
        - **Password** - The token you entered in the Connection URL.

          > Tip To successfully establish a Databricks connection verify that your token is valid, because they generally expire in 90 days.

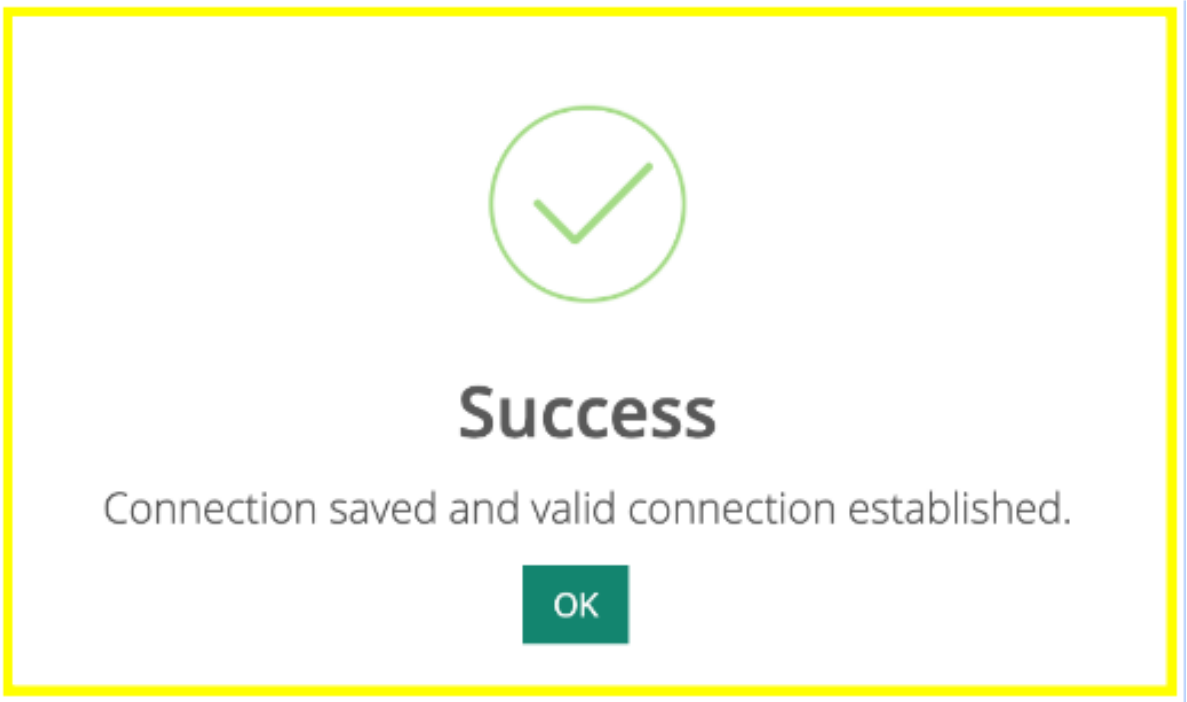
- **Driver Location** - Enter the location where the driver resides.

Your connection should look similar to the following screenshot:



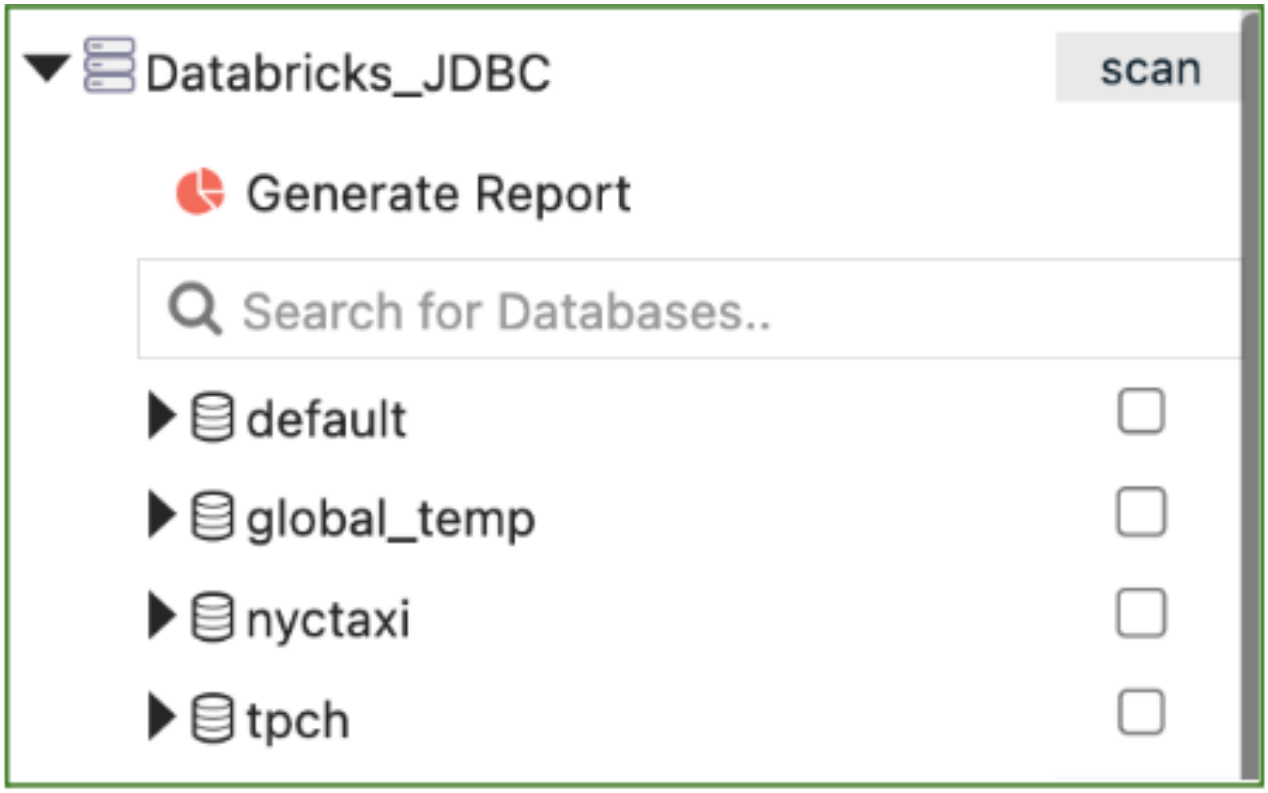
6. Click **Save**.

   If your connection information is valid, you receive the following message:

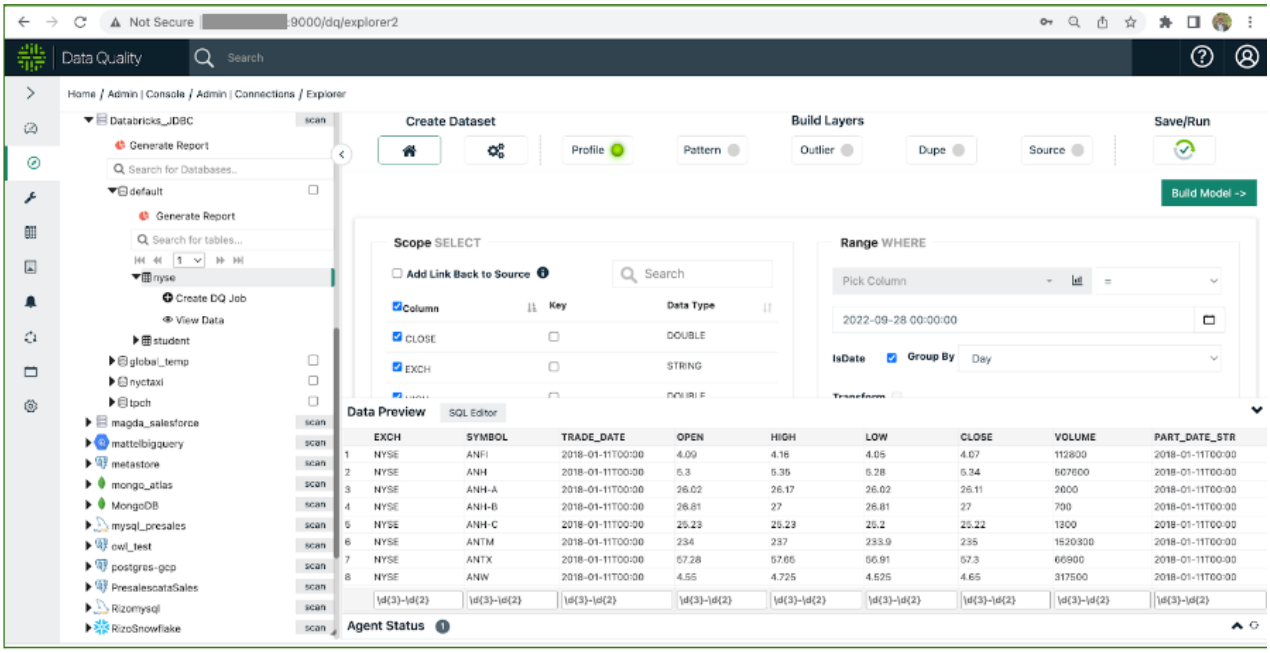


You have now successfully made a Collibra DQ connection to Databricks via JDBC.

> Tip To view the connection, you can navigate to the Collibra DQExplorer page and click Databricks JDBC connection.
> Choose your dataset and continue to create your DQ Job.

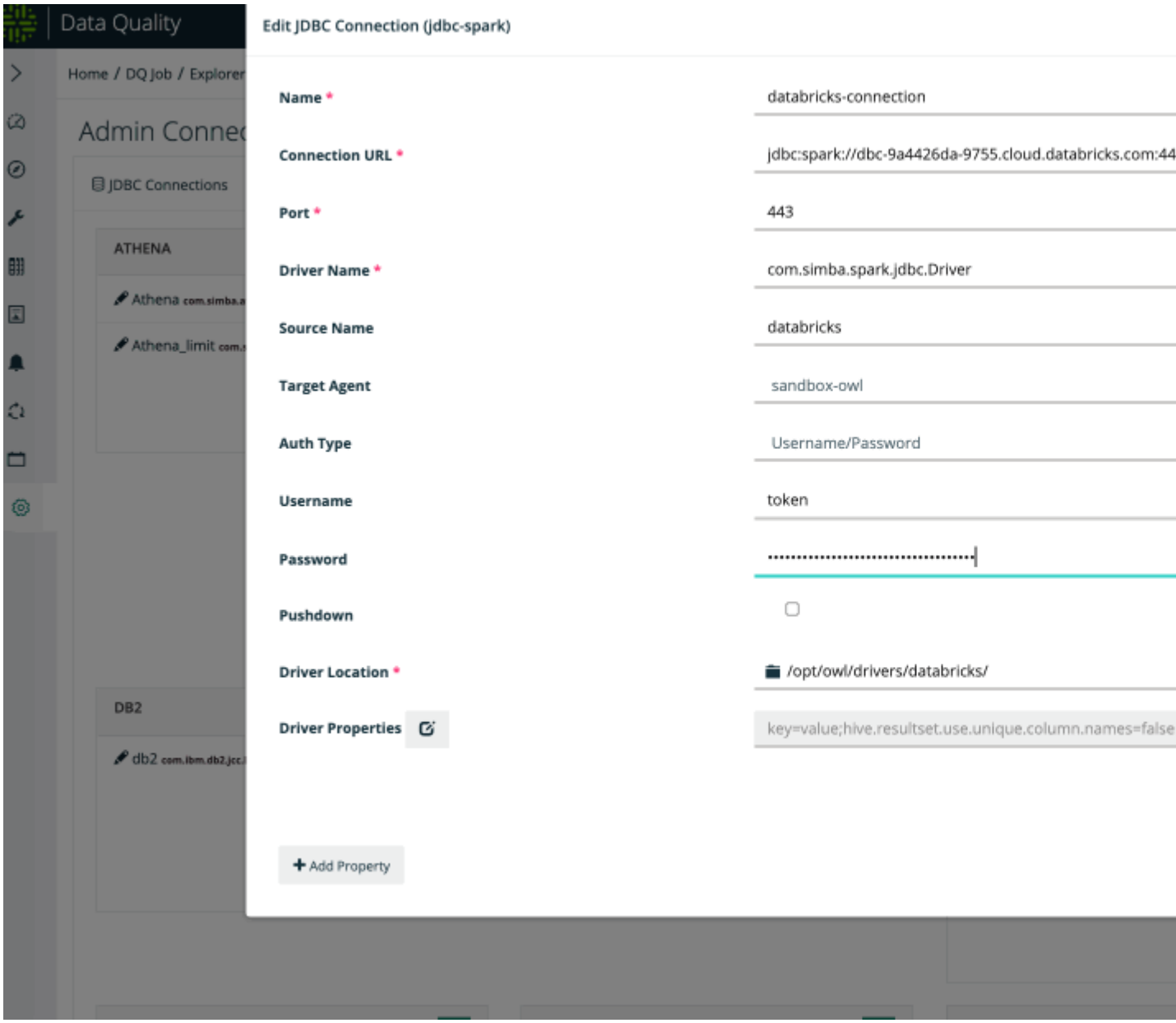


The following screenshot is an example of the **default > nyse** database.

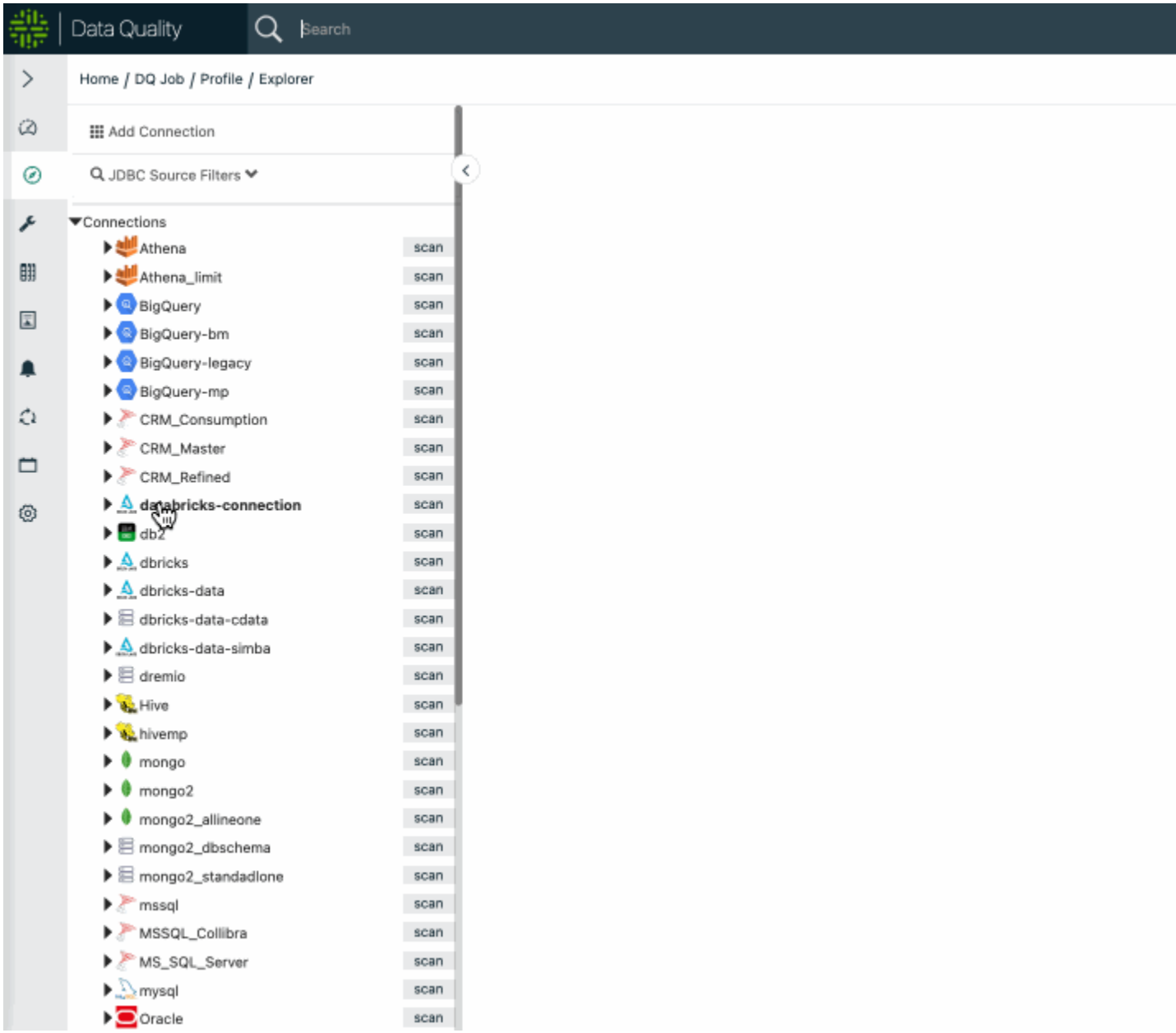


Recap

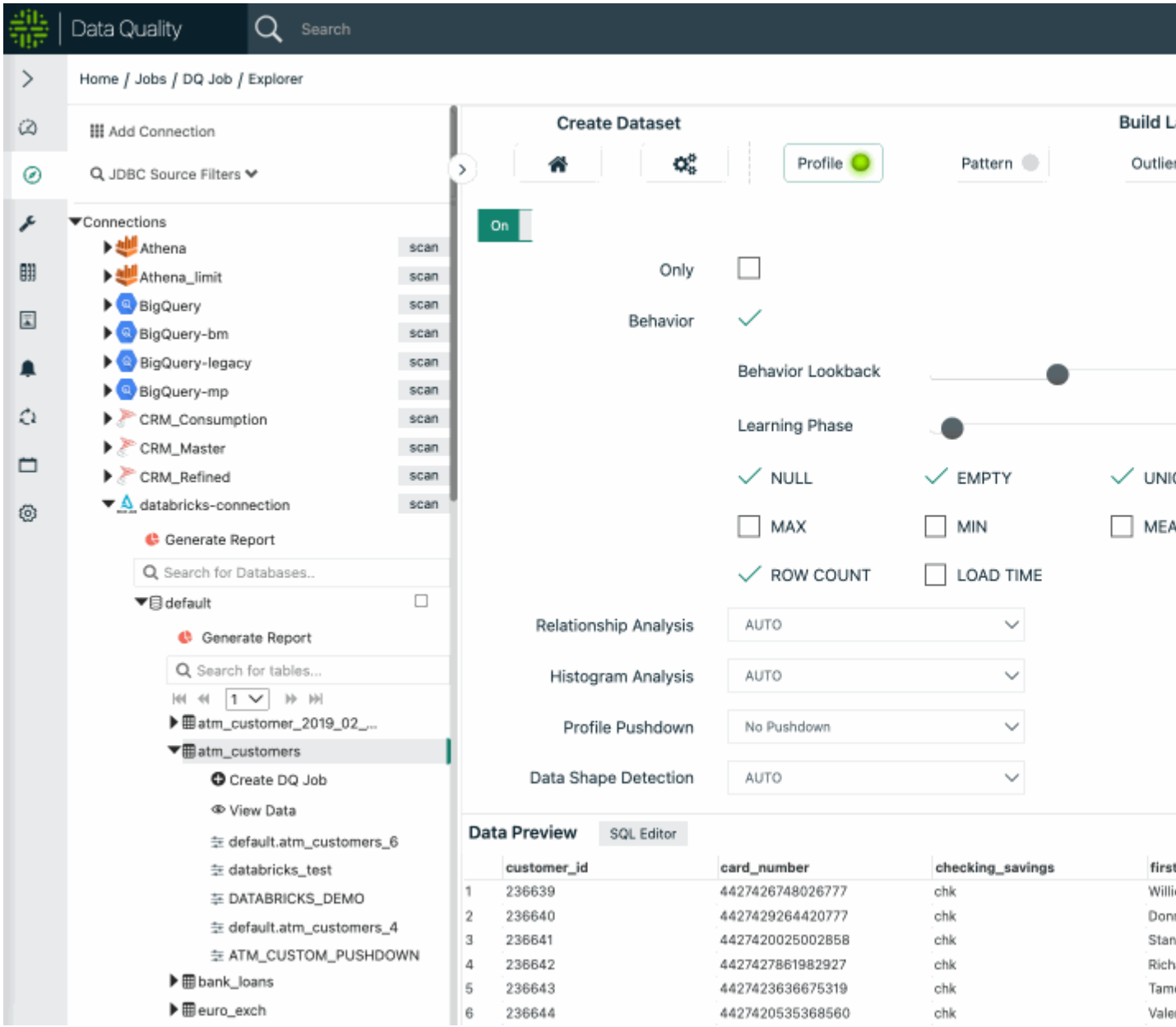
Input JDBC credentials:



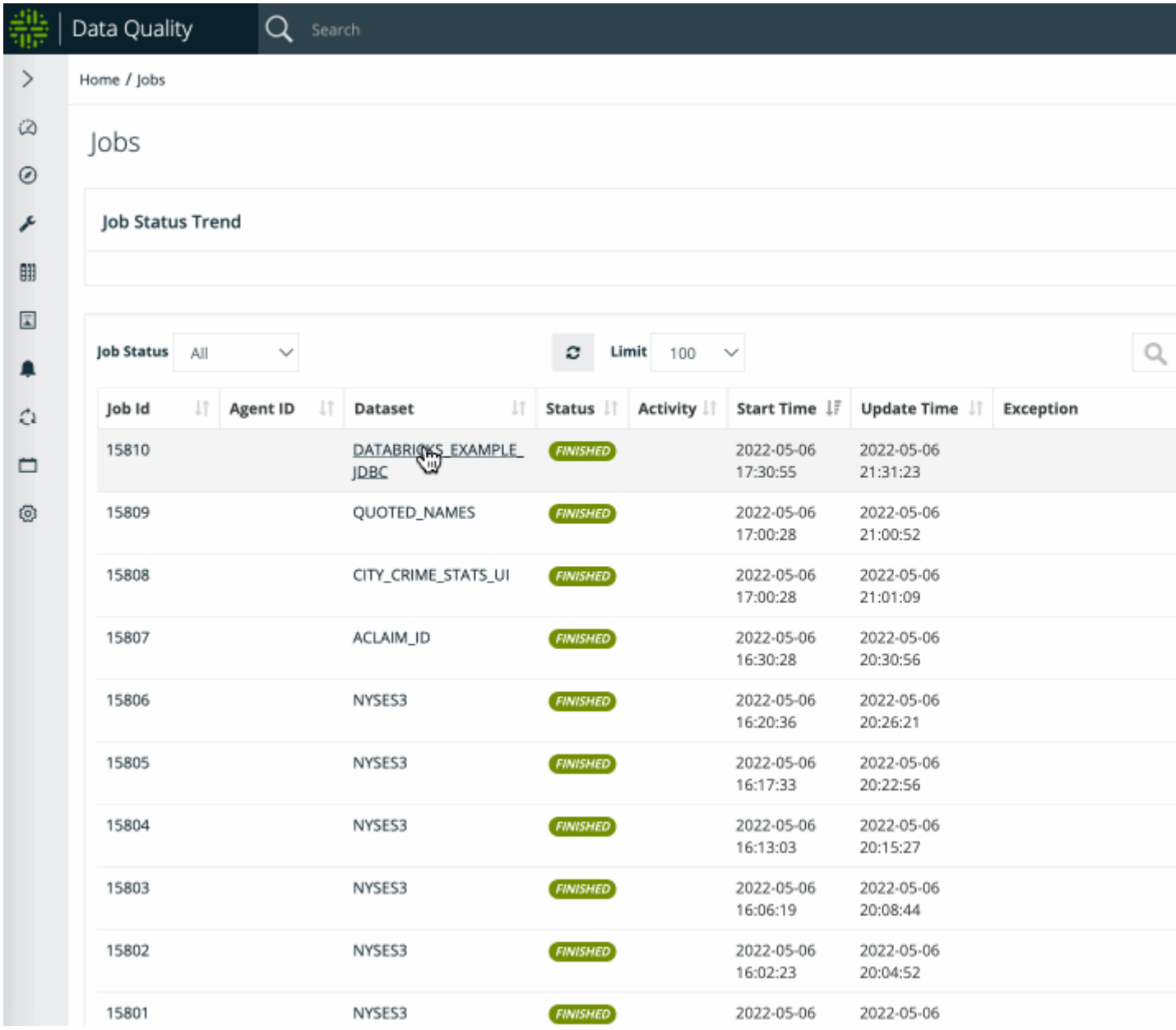
Browse connection:



Run Job:


Data Quality
Search
Home / Jobs / DQ Job / Explorer
Add Connection
JDBC Source Filters
Connections
Athena
Athena_limit
BigQuery
BigQuery-bm
BigQuery-legacy
BigQuery-mp
CRM_Consumption
CRM_Master
CRM_Refined
databricks-connection
scan
Generate Report
Search for Databases...
default
Search for tables...
atm_customer_2019_02_...
atm_customers
Create DQ Job
View Data
default.atm_customers_6
databricks_test
DATABRICKS_DEMO
default.atm_customers_4
ATM_CUSTOM_PUSHDOWN
bank_loans
euro_exch
Create Dataset
Profile
Pattern
On
Only
Behavior
Behavior Lookback
Learning Phase
NULL
EMPTY
MAX
MIN
ROW COUNT
LOAD TIME
Relationship Analysis
Histogram Analysis
Profile Pushdown
Data Shape Detection
AUTO
No Pushdown
Data Preview
SQL Editor
customer_id
card_number
checking_savings
1 236639 4427426748026777 chk
2 236640 4427429264420777 chk
3 236641 4427420025002858 chk
4 236642 4427427861982927 chk
5 236643 4427423636675319 chk
6 236644 4427420535368560 chk

See results:


Data Quality
Search
Home / Jobs
Jobs
Job Status Trend
Job Status All
Limit 100
Job Id | Agent ID | Dataset | Status | Activity | Start Time | Update Time | Exception
15810 DATABRICKS_EXAMPLE_JDBC FINISHED 2022-05-06 17:30:55 2022-05-06 21:31:23
15809 QUOTED_NAMES FINISHED 2022-05-06 17:00:28 2022-05-06 21:00:52
15808 CITY_CRIME_STATS_UI FINISHED 2022-05-06 17:00:28 2022-05-06 21:01:09
15807 ACLAIM_ID FINISHED 2022-05-06 16:30:28 2022-05-06 20:30:56
15806 NYSES3 FINISHED 2022-05-06 16:20:36 2022-05-06 20:26:21
15805 NYSES3 FINISHED 2022-05-06 16:17:33 2022-05-06 20:22:56
15804 NYSES3 FINISHED 2022-05-06 16:13:03 2022-05-06 20:15:27
15803 NYSES3 FINISHED 2022-05-06 16:06:19 2022-05-06 20:08:44
15802 NYSES3 FINISHED 2022-05-06 16:02:23 2022-05-06 20:04:52
15801 NYSES3 FINISHED 2022-05-06 2022-05-06

Connectivity to Hive

Example URL

```
jdbc:hive2://cdh-instance1.us-east1-b.c.owl-hadoop-cdh.in-
ternal:10000/default;AuthMech=1;KrbHostFQDN=cdh-instance1.us-
east1-b.c.owl-hadoop-cdh.in-
ternal;Kr-
bRealm-
=CW.COM;K-
rbSer-
viceName=hive;SSL=1;SSLKeyStore=/opt/cloudera/security/pki/cdh-
instance1.us-east1-b.c.owl-hadoop-cdh.internal-
serv-
er.jk-
s;Al-
lowSelfSignedCerts=1;SSLKeyStorePwd=password;principal=hive/cdh-
instance1.us-east1-b.c.owl-hadoop-cdh.internal@CW.COM
```

Driver Name

```
com.simba.hive.jdbc41.HS2Driver
```

Driver Properties

```
hive.resultset.use.unique.column.names=false
```

What Does each option mean

Base connection string = jdbc:hive2://cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal:10000/default

Authentication identifier = AuthMech=1 (which states Kerberos)

Kerberos Hive Server FQDN = KrbHostFQDN=cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal

Kerberos Realm used = KrbRealm=CW.COM *(Not necessarily needed)*

Kerberos Service name = KrbServiceName=hive

Enabling SSL = SSL=1

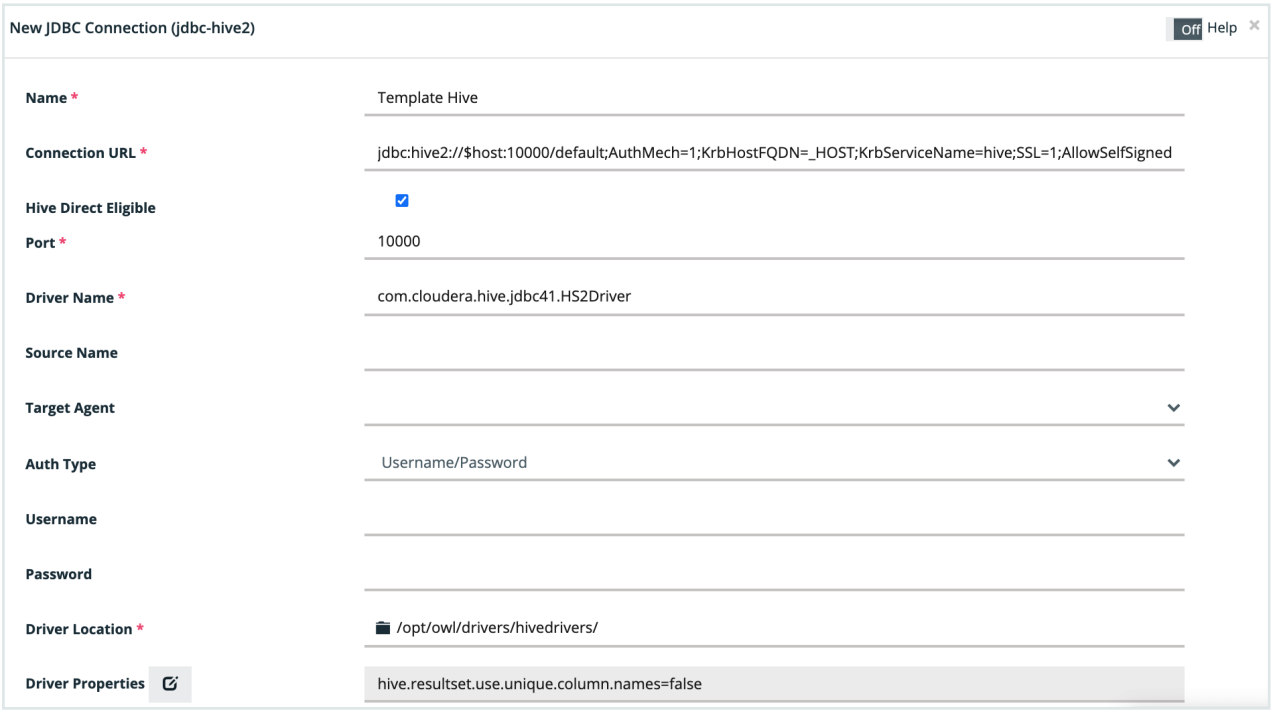
The SSL KeyStore to be used to = SSLKeyStore=/opt/cloudera/security/pki/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal-server.jks *(Could use SSL TrustStore also)*

Allow for Self Signed certifications to be OK = AllowSelfSignedCerts=1 *(our environment used self signed certs)*

Password to the KeyStore = SSLKeyStorePwd=password *(Not necessarily needed)*

Kerberos Principal to use to Authenticate with = principal=hive/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal@CW.COM

Example Connection



FAQ

It is very common to require this connection property when not using the default schema. Remember to add this to the connection properties when defining the connection.

```
hive.resultset.use.unique.column.names=false
```

Connecting to CDH 5.16 Hive SSL/TLS/Kerberos Setup

The Cloudera Hive JDBC drivers used
https://www.cloudera.com/downloads/connectors/hive/jdbc/2-5-16.html

JDBC Connection String used

jdbc:hive2://cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal:10000/default;AuthMech=1;KrbHostFQDN=cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal;KrbRealm=CW.COM;KrbServiceName=hive;SSL=1;SSLKeyStore=/opt/cloudera/security/pki/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal-server.jks;AllowSelfSignedCerts=1;SSLKeyStorePwd=password;principal=hive/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal@CW.COM

What Does each option mean

Base connection string = jdbc:hive2://cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal:10000/default

Authentication identifier = AuthMech=1 (which states Kerberos)

Kerberos Hive Server FQDN = KrbHostFQDN=cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal

Kerberos Realm used = KrbRealm=CW.COM *(Not necessarily needed)*

Kerberos Service name = KrbServiceName=hive

Enabling SSL = SSL=1

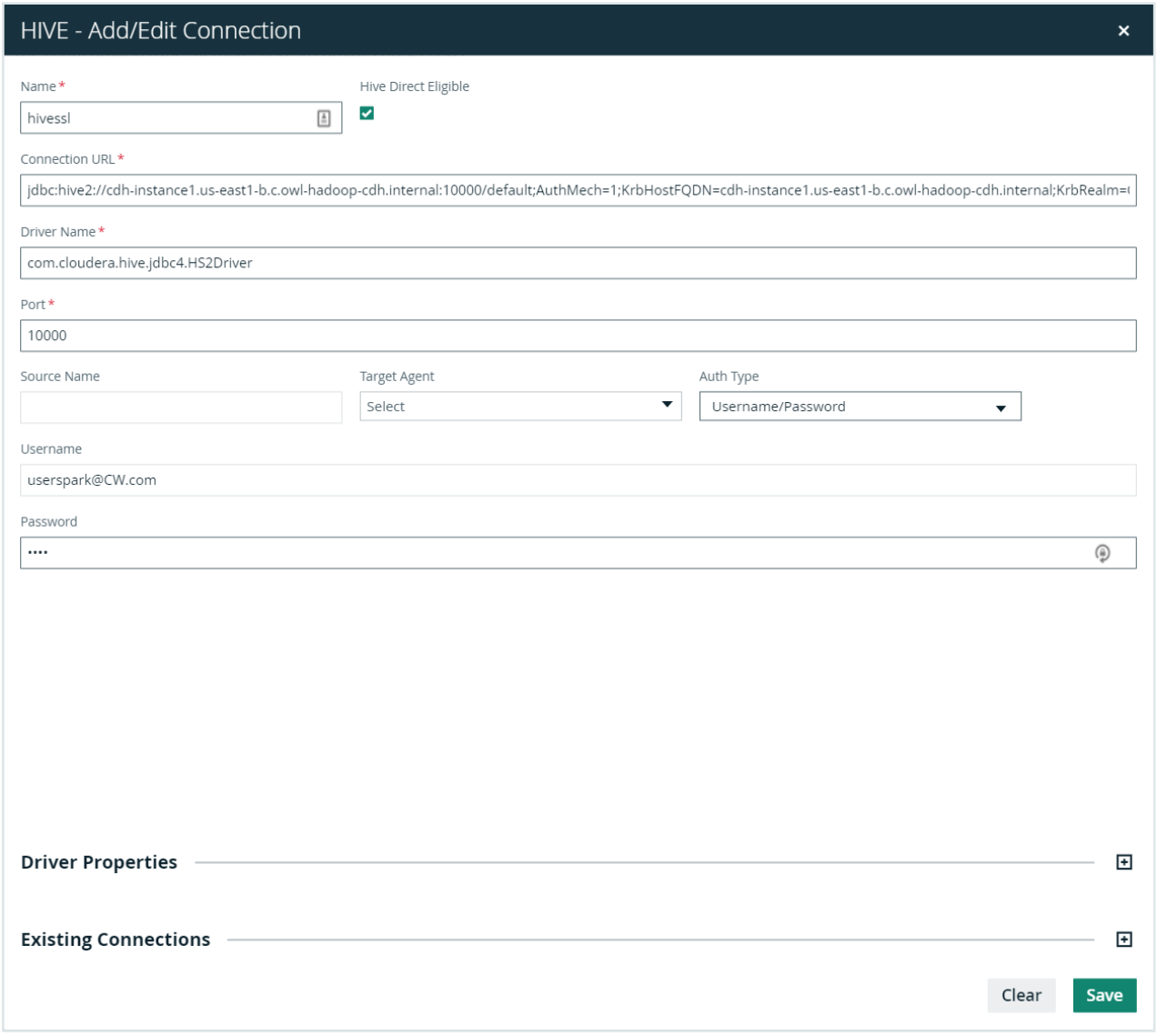
The SSL KeyStore to be used to = SSLKeyStore=/opt/cloudera/security/pki/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal-server.jks *(Could use SSLTrustStore also)*

Allow for Self Signed certifications to be OK = AllowSelfSignedCerts=1 *(our environment used self signed certs)*

Password to the KeyStore = SSLKeyStorePwd=password *(Not necessarily needed)*

Kerberos Principal to use to Authenticate with = principal=hive/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal@CW.COM

Within the Collibra DQ Web UI you have to specify the following (See ScreenShot below)



Name = hivessl

Connection URL = jdbc:hive2://cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal:10000/default;AuthMech=1;KrbHostFQDN=cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal;KrbRealm=CW.COM;KrbServiceName=hive;SSL=1;SSLKeyStore=/opt/cloudera/security/pki/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal-server.jks;AllowSelfSignedCerts=1;SSLKeyStorePwd=password;principal=hive/cdh-instance1.us-east1-b.c.owl-hadoop-cdh.internal@CW.COM

Port = 10000

Driver Name = com.cloudera.hive.jdbc4.HS2Driver

Username = userspark@CW.COM

Password = password

Connectivity to Oracle

Example URL

```
jdbc:oracle:thin:@10.589.0.31:1521:DEV
```

Driver Name

```
oracle.jdbc.OracleDriver
```

Connection Properties Recognized by Oracle JDBC Drivers

Name	Short Name	Type	Description
user	n/a	String	the user name for logging into the database
password	n/a	String	the password for logging into the database

Name	Short Name	Type	Description
database	server	String	the connect string for the database
internal_logon	n/a	String	a role, such as `sysdba` or `sysoper`, that allows you to log on as `sys`
defaultRowPrefetch	prefetch	String (containing integer value)	the default number of rows to prefetch from the server (default value is "10")
remarksReporting	remarks	String (containing boolean value)	"true" if getTables() and getColumns() should report TABLE_REMARKS; equivalent to using setRemarksReporting() (default value is "false")
defaultBatchValue	batchvalue	String (containing integer value)	the default batch value that triggers an execution request (default value is "10")
includeSynonyms	synonyms	String (containing boolean value)	"true" to include column information from predefined "synonym" SQL entities when you execute a `DataBaseMetaData getColumns()` call; equivalent to connection `setIncludeSynonyms()` call (default value is "false")

Name	Short Name	Type	Description
processEscapes	n/a	String (containing boolean value)	"false" to disable escape processing for statements (Statement or PreparedStatement) created from this connection. Set this to "false" if you want to avoid many calls to `Statement.setEscapeProcessing(false);`. This is espcially usefull for PreparedStatement where a call to `setEscapeProcessing (false)` would have no effect. The default is "true".
defaultNChar	n/a	String (containing boolean value)	"false" is the default. If set to "true", the default behavior for handling character datatypes is changed so that NCHAR/NVARCHAR2 become the default. This means that setFormOfUse() won't be needed anymore when using NCHAR/NVARCHAR2. This can also be set as a java property :`java -Doracle.jdbc.defaultNChar=true myApplication`
useFetchSizeWithLongColumn	n/a	String (containing boolean value)	"false" is the default. **THIS IS A THIN ONLY PROPERTY. IT SHOULD NOT BE USED WITH ANY OTHER DRIVERS.**If set to "true", the performance when retrieving data in a 'SELECT' will be improved but the default behavior for handling LONG columns will be changed to fetch multiple rows (prefetch size). It means that enough memory will be allocated to read this data. So if you want to use this property, make sure that the LONG columns you are retrieving are not too big or you may run out of memory. This property can also be set as a java property : `java -Doracle.jdbc.useFetchSizeWithLongColumn=true myApplication`

Name	Short Name	Type	Description
SetFloatAndDoubleUseBinary	n/a	String (containing boolean value)	"false" is the default. If set to "true", causes the java.sql.PreparedStatment setFloat and setDouble API's to use internal binary format as for BINARY_FLOAT and BINARY_DOUBLE parameters. See oracle.jdbc.OraclePreparedStatement setBinaryFloat and setBinaryDouble

https://docs.oracle.com/cd/E11882_01/appdev.112/e13995/oracle/jdbc/OracleDriver.html

Connectivity to Presto

Example URL

```
jdbc:presto://xyz.presto.svc.cluster.local:8080/database
```

Driver Name

```
com.facebook.presto.jdbc.PrestoDriver
```

Connection Properties

These methods may be mixed; some parameters may be specified in the URL while others are specified using properties. However, the same parameter may not be specified using both methods.

Parameter Reference

Name	Description
`user`	Username to use for authentication and authorization.

Name	Description
`password`	Password to use for LDAP authentication.
`socksProxy`	SOCKS proxy host and port. Example: `localhost:1080`
`httpProxy`	HTTP proxy host and port. Example: `localhost:8888`
`protocols`	Comma delineated list of HTTP protocols to use. Example: `protocols=http11`. Acceptable values: `http11,http10,http2`
`applicationNamePrefix`	Prefix to append to any specified `ApplicationName` client info property, which is used to set the source name for the Presto query. If neither this property nor `ApplicationName` are set, the source for the query will be `presto-jdbc`.
`accessToken`	Access token for token based authentication.
`SSL`	Use HTTPS for connections
`SSLKeyStorePath`	The location of the Java KeyStore file that contains the certificate and private key to use for authentication.
`SSLKeyStorePassword`	The password for the KeyStore.
`SSLTrustStorePath`	The location of the Java TrustStore file that will be used to validate HTTPS server certificates.
`SSLTrustStorePassword`	The password for the TrustStore.
`KerberosRemoteServiceName`	Presto coordinator Kerberos service name. This parameter is required for Kerberos authentication.
`KerberosPrincipal`	The principal to use when authenticating to the Presto coordinator.

Name	Description
`KerberosUseCanonicalHostname`	Use the canonical hostname of the Presto coordinator for the Kerberos service principal by first resolving the hostname to an IP address and then doing a reverse DNS lookup for that IP address. This is enabled by default.
`KerberosConfigPath`	Kerberos configuration file.
`KerberosKeytabPath`	Kerberos keytab file.
`KerberosCredentialCachePath`	Kerberos credential cache.
`extraCredentials`	Extra credentials for connecting to external services. The extraCredentials is a list of key-value pairs. Example: `foo:bar;abc:xyz` will create credentials `abc=xyz` and `foo=bar`
`customHeaders`	Custom headers to inject through JDBC driver. The customHeaders is a list of key-value pairs. Example: `testHeaderKey:testHeaderValue` will inject the header `testHeaderKey` with value `testHeaderValue`. Values should be percent encoded.

Connectivity to Redshift

Example URL

```
jdbc:redshift://redshift-cluster-name.kdkcis9g8.us-east-1.redshift.amazonaws.com:5439/dev
```

Driver Name

```
com.amazon.redshift.jdbc.Driver
```

Amazon Redshift offers drivers for tools that are compatible with the JDBC 4.2 API. For information about the functionality supported by these drivers, see the Amazon Redshift JDBC driver release notes.

For detailed information about how to install the JDBC driver version 1.0, reference the JDBC driver libraries, and register the driver class, see Amazon Redshift JDBC driver installation and configuration guide.

For each computer where you use the Amazon Redshift JDBC driver, make sure that Java Runtime Environment (JRE) 8.0 is installed.

If you use the Amazon Redshift JDBC driver for database authentication, make sure that you have AWS SDK for Java 1.11.118 or later in your Java class path. If you don't have AWS SDK for Java installed, download the ZIP file with JDBC 4.2-compatible driver (without the AWS SDK) and driver dependent libraries for the AWS SDK:

- JDBC 4.2-compatible driver (without the AWS SDK) and driver dependent libraries for AWS SDK files version 1.2.55.

  The class name for this driver is `com.amazon.redshift.jdbc42.Driver`.

  This ZIP file contains the JDBC4.2-compatible driver (without the AWS SDK) and its dependent library files. Unzip the dependent jar files to the same location as the JDBC driver. Only the JDBC driver needs to be in the CLASSPATH because the driver manifest file contains all dependent library file names which are located in the same directory as the JDBC driver. For more information about how to install the JDBC driver, see Amazon Redshift JDBC driver installation and configuration guide.

  Use this Amazon Redshift JDBC driver with the AWS SDK that is required for IAM database authentication.

- JDBC 4.2-compatible driver (without the AWS SDK) version 1.2.55.

  The class name for this driver is `com.amazon.redshift.jdbc42.Driver`.

  Be sure to use ANTLR version 4.8.1. The antlr4-runtime-4.8-1.jar is included in the ZIP download link above with the JDBC 4.2-compatible driver (without the AWS SDK) and driver dependent libraries for the AWS SDK.

For more information about previous driver versions, see Use previous JDBC driver versions with the AWS SDK for Java.

Then download and review the Amazon Redshift ODBC and JDBC driver license agreement.

If your tool requires a specific previous version of a driver, see Use previous JDBC driver version 1.0 driver versions in certain cases.

Getting the JDBC URL

Before you can connect to your Amazon Redshift cluster from a SQL client tool, you need to know the JDBC URL of your cluster. The JDBC URL has the following format: `jdbc:redshift://endpoint:port/database`.Note

A JDBC URL specified with the former format of `jdbc:postgresql://endpoint:port/database` still works.

The fields of the format shown preceding have the following values.

Field	Value
`jdbc`	The protocol for the connection.
`redshift`	The subprotocol that specifies to use the Amazon Redshift driver to connect to the database.
`endpoint`	The endpoint of the Amazon Redshift cluster.
`port`	The port number that you specified when you launched the cluster. If you have a firewall, make sure that this port is open for you to use.
`database`	The database that you created for your cluster.

The following is an example JDBC URL:
`jdbc:redshift://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/dev`

For information about how to get your JDBC connection, see Finding your cluster connection string.

If the client computer fails to connect to the database, you can troubleshoot possible issues. For more information, see Troubleshooting connection issues in Amazon Redshift.\

https://docs.aws.amazon.com/redshift/latest/mgmt/configure-jdbc-connection.html

Connectivity to Snowflake

Example URL

JDBC Driver Connection String

```
                <MadCap:concept term="Data Quality" xmlns:MadCap-
p="http://www.madcapsoftware.com/Schemas/MadCap.xsd" />jd-
bc:snowflake://accountname.us-east-
2.aws.snowflakecomputing.com?db=cdq&warehouse=cdqw
```

Driver Name

```
net.snowflake.client.jdbc.SnowflakeDriver
```

The previous driver class, `com.snowflake.client.jdbc.SnowflakeDriver`, is still supported but is deprecated (i.e. it will be removed in a future release, TBD).

Limit Databases Displayed

Add this connection property to limit the databases and tables displayed in the Explorer view. This will reduce the entries in the Explorer window to only those tables that the user can access.

```
CLIENT_METADATA_REQUEST_USE_CONNECTION_CTX=TRUE
```

Connection Parameters

For documentation on individual connection parameters, see the JDBC Driver Connection Parameter Reference.<account_identifier>

Specifies the account identifier for your Snowflake account. For details, see Account Identifiers. For examples of the account identifier used in a JDBC connection string, see Examples.<connection_params>

Specifies a series of one or more parameters, in the form of <param>=<value>, with each parameter separated by the ampersand character (&), and no spaces anywhere in the connection string.

For documentation on individual connection parameters, see the JDBC Driver Connection Parameter Reference.

Other Parameters

Any session parameter can be included in the connection string. For example:

> CLIENT_SESSION_KEEP_ALIVE=<Boolean>
>
> Specifies whether to keep the current session active after a period of inactivity, or to force the user to login again. If the value is true, Snowflake keeps the session active indefinitely, even if there is no activity from the user. If the value is false, the user must log in again after four hours of inactivity.
>
> Default is false.

For descriptions of all the session parameters, see Parameters.

Examples

The following is an example of the connection string that uses an account identifier that specifies the account myaccount in the organization myorganization.

```
jdbc:snowflake://myorganization-myac-
coun-
t.s-
now-
flake-
com-
put-
ing.-
com/?user=peter&warehouse=mywh&db=mydb&schema=public
```

The following is an example of a connection string that uses the account locator `xy12345` as the account identifier:

```
jdb-
c:s-
now-
flake://xy12345.s-
now-
flake-
com-
put-
ing.-
com/?user=peter&warehouse=mywh&db=mydb&schema=public
```

Note that this example uses an account in the AWS US West (Oregon) region. If the account is in a different region or if the account uses a different cloud provider, you need to specify additional segments after the account locator.

Configuring the JDBC Driver -- Snowflake Documentation

Private Link

Please let us know if you are using private link for Snowflake. Setup can vary depending on the endpoint that is created. In most cases, use the private endpoint as a normal JDBC connection.

Snowflake Community

Advanced Private Link and Proxy

Here is an example JDBC string connection we used that take into account the following setup:

- <ACCOUNT_NAME> is the full link to the Snowflake instance with the private link.
- DQ installed on-prem in a private IaaS and DQ is behind a proxy.
- If the Snowflake instance is using a private link, whitelist the private link URL to bypass the proxy.
- In addition to connectivity to the Snowflake instance, the JDBC driver tries to access Snowflake Blob storage by connecting directly to some S3 buckets managed by Snowflake.
- Those need to be whitelisted as well.

Example URL

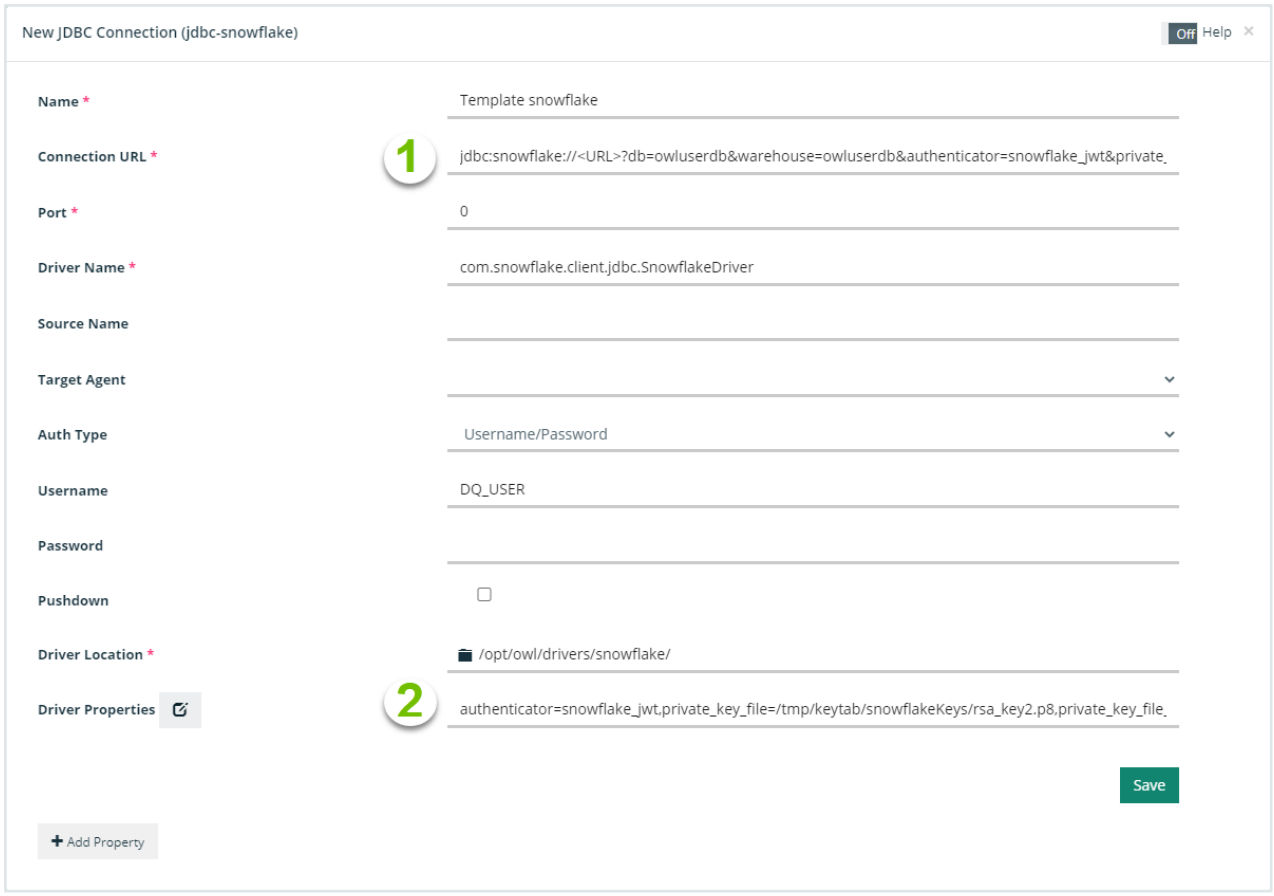
jdbc:snowflake://<ACCOUNT_NAME>/?tracing=all&useProxy=true&proxyHost=10.142.22.37&proxyPort=8080&proxyUser=xyz&proxyPassword=xyz&nonProxyHosts=*.privatelink.snowflakecomputing.com%7Csfc-eu-ds1-customer-stage.s3.eu-central-1.amazonaws.com

Configuring Key Pair Authentication

Collibra DQ supports key pair authentication, which is an alternative to basic authentication methods, like username and password, and provides enhanced authentication security.

There are two ways to use key pair authentication on Snowflake connections:

- Via the Connection URL.
- Via the Driver Properties.



Configuring key pair authentication via the Connection URL

The table below shows the properties used to configure key pair authentication with a Connection URL in the New JDBC Connection modal.

Property	Description
Connection URL	jdbc:snowflake://<URL>?db=owluserdb&warehouse=owluserdb&authenticator=snowflake_jwt&private_key_file=/<your-file-path>/keytab/snowflake_key.p8&private_key_file_pwd=<your-password> The file name should include a .p8 extension, as shown in the example above. > Note Connection URL parameters must be appended by ampersand (&).
Username	The username for which the private key is created. For example, DQ_User.
Password	The password field must be left empty.

Configuring key pair authentication via the Driver Properties

The table below shows the properties used to configure key pair authentication with the Driver Properties in the New JDBC Connection modal.

Property	Description
Username	The username for which the private key is created. For example, DQ_User.
Password	The password field must be left empty.
Driver Properties	authenticator=snowflake_jwt,private_key_file=/<your-file-path>/snowflakeKeys/rsa_key2.p8,private_key_file_pwd=<your-password> The file name should include a .p8 extension, as shown in the example above. Note Driver properties must be comma separated, as shown in the example above.

Setting up Snowflake Pushdown

> Note As of the 2022.11 release, Snowflake Pushdown is available as a public beta offering for participating customers.

To get started with Snowflake Pushdown, a user with Admin access must run the following script to set up users, roles, and the Collibra DQ virtual warehouse.

Step	Details

1	Update the following session variables:
	`set dq_username='SERVICE_ACCOUNT_USER';`
	`set dq_password='SERVICE_ACCOUNT_PASSWORD';`
	`set dq_warehouse_name='COLLIBRA_DQ_WH';`
	`set dq_warehouse_size='XSMALL';`
	`set user_database='TARGET_DB';`
	`set dq_role_name='COLLIBRA_DQ_ROLE';`
	Note `dq_password` must be in uppercase.
	Warning Do not update the variables for Collibra DQ.

2	Run as an admin user. `USE ROLE ACCOUNTADMIN;`

3	Create a user and role for Collibra DQ.
	`CREATE ROLE IF NOT EXISTS identifier($dq_role_name);`
	`CREATE USER IF NOT EXISTS identifier($dq_username) PASSWORD=$dq_password DEFAULT_ROLE=$dq_role_name;`
	`GRANT ROLE identifier($dq_role_name) TO USER identifier($dq_username);`

4	Create a virtual warehouse to run Collibra DQ. The virtual warehouse is referenced in the connection URL to direct DQ traffic to it.
	`CREATE WAREHOUSE IF NOT EXISTS identifier($dq_warehouse_name) WAREHOUSE_SIZE=$dq_warehouse_size INITIALLY_SUSPENDED=TRUE`
	`AUTO_SUSPEND = 5 AUTO_RESUME = TRUE;`

Step	Details
5	Assign privileges to the Collibra DQ warehouse. `GRANT OPERATE, USAGE, MONITOR ON WAREHOUSE identifier($dq_warehouse_name) TO ROLE identifier ($dq_role_name);`

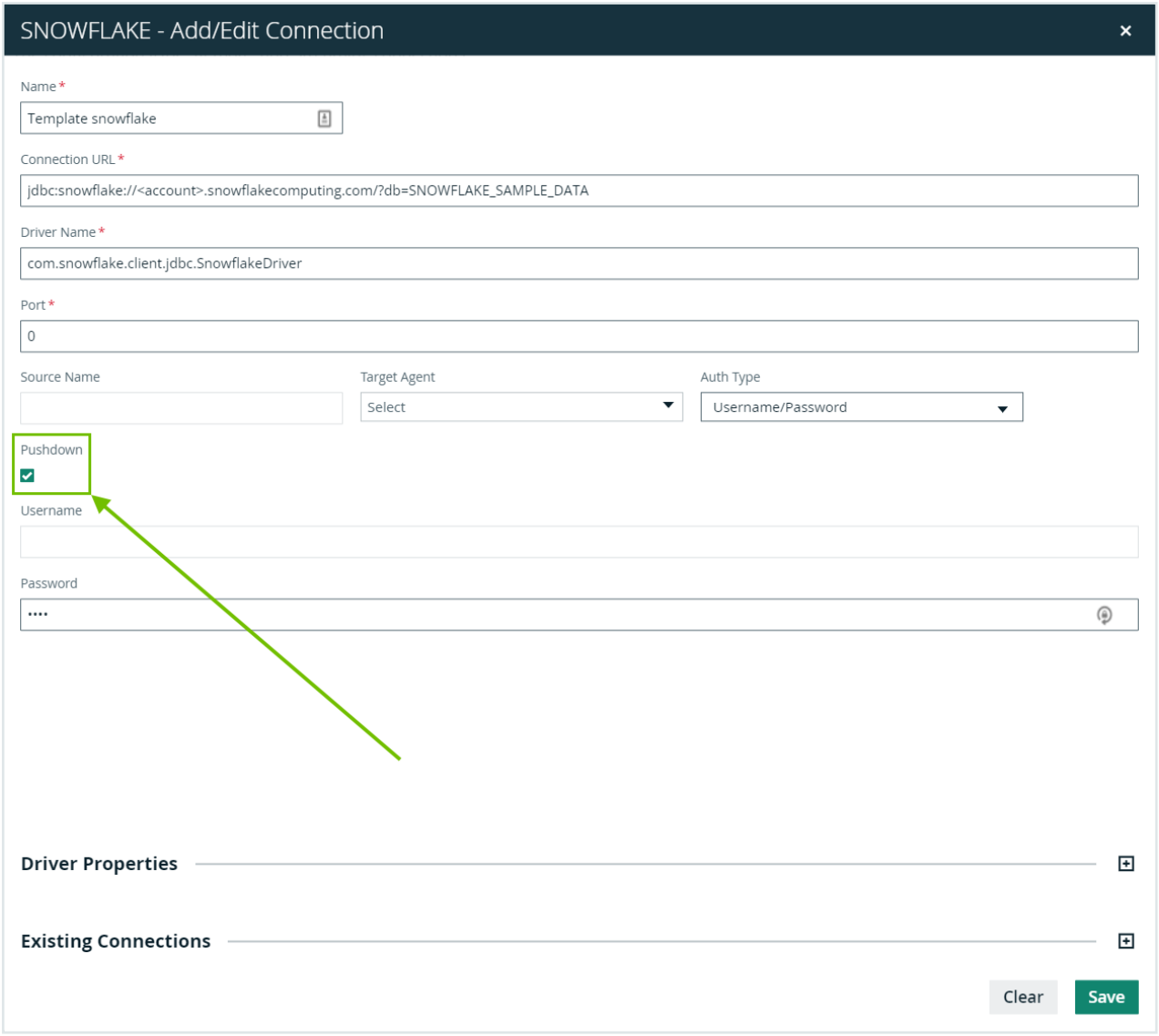
6	Assign metadata access to your Collibra DQ role.
	`GRANT USAGE,MONITOR on DATABASE identifier($user_database) to identifier($dq_role_name);`
	`GRANT USAGE,MONITOR ON ALL SCHEMAS IN DATABASE identifier($user_database) to identifier($dq_role_name);`

Step	Details

7	Update the session variable `user_database` and grant read access to objects in the user database. Run this portion for each target database within which you want to run DQ checks.
	
	`USE DATABASE identifier($user_database);`
	`GRANT SELECT ON ALL TABLES IN DATABASE identifier ($user_database) TO ROLE identifier($dq_role_ name);`
	`GRANT SELECT ON ALL VIEWS IN DATABASE identifier ($user_database) TO ROLE identifier($dq_role_ name);`
	`GRANT SELECT ON ALL EXTERNAL TABLES IN DATABASE identifier($user_database) TO ROLE identifier ($dq_role_name);`
	`GRANT SELECT ON ALL STREAMS IN DATABASE iden- tifier($user_database) TO ROLE identifier($dq_ role_name);`
	
	`GRANT SELECT ON FUTURE TABLES IN DATABASE iden- tifier($user_database) TO ROLE identifier($dq_ role_name);`
	`GRANT SELECT ON FUTURE VIEWS IN DATABASE iden- tifier($user_database) TO ROLE identifier($dq_ role_name);`
	`GRANT SELECT ON FUTURE EXTERNAL TABLES IN DATABASE identifier($user_database) TO ROLE iden- tifier($dq_role_name);`
	`GRANT SELECT ON FUTURE STREAMS IN DATABASE iden- tifier($user_database) TO ROLE identifier($dq_ role_name);`

> Warning Please ensure the SQL variables are updated in the above script before proceeding.

To run a Snowflake Pushdown job, you must opt in when setting up your Snowflake connection. To toggle Pushdown capabilities on, ensure that the Pushdown checkbox in the Snowflake connection modal is checked.



Connectivity to SQL Server

Example URL

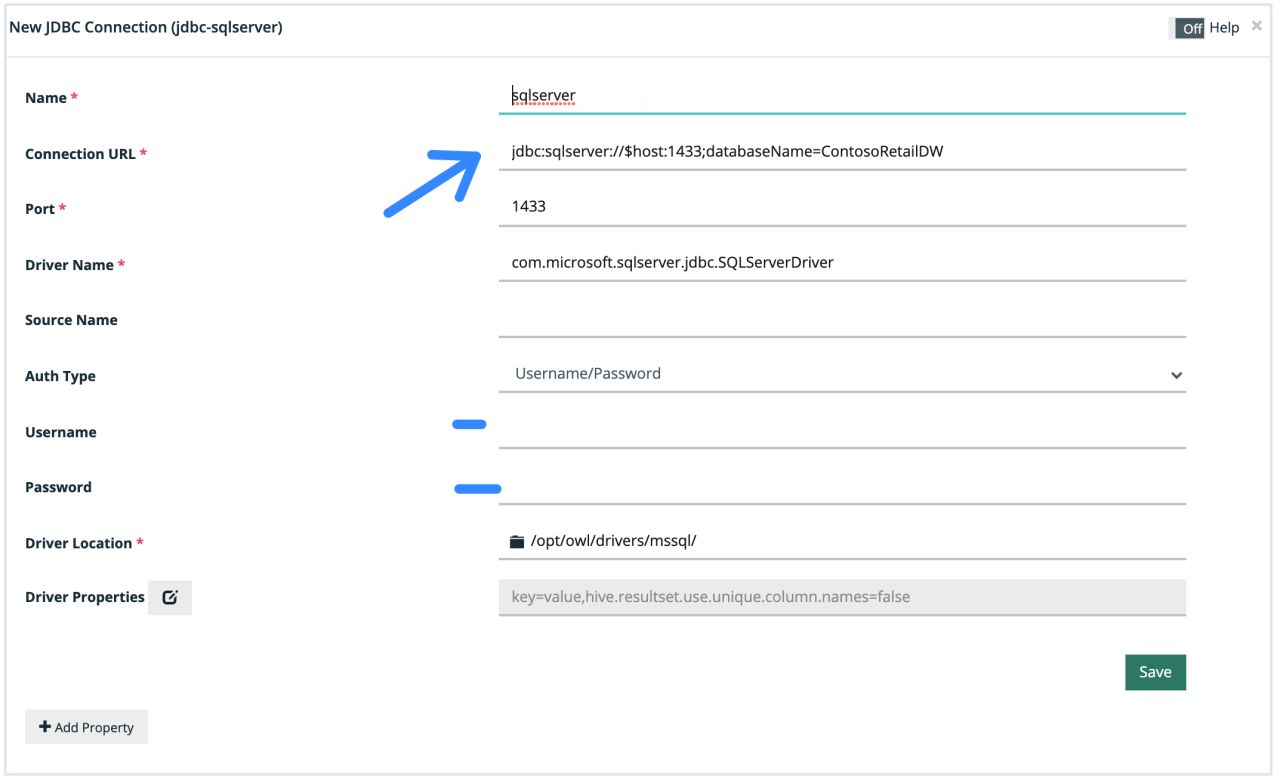
```
jdbc:sqlserver://$host:1433;databaseName=ContosoRetailDW
```

Driver Name

```
com.microsoft.sqlserver.jdbc.SQLServerDriver
```

Setup

1. Navigate to the connections page
2. Locate the SQL Server template
3. Fill in the required JDBC details
4. Click save to validate the connection



Microsoft JDBC Driver for SQL Server - JDBC Driver for SQL Server

Troubleshooting

The Microsoft JDBC Driver for SQL Server requires that TCP/IP be installed and running to communicate with your SQL Server database. You can use the SQL Server Configuration Manager to verify which network library protocols are installed.

A database connection attempt might fail for many reasons. These can include the following:

- TCP/IP is not enabled for SQL Server, or the server or port number specified is incorrect. Verify that SQL Server is listening with TCP/IP on the specified server and port. This might be reported with an exception similar to: "The login has failed. The TCP/IP connection to the host has failed." This indicates one of the following:
  - SQL Server is installed but TCP/IP has not been installed as a network protocol for SQL Server by using the SQL Server Network Utility for SQL Server 2000 (8.x), or the SQL Server Configuration Manager for SQL Server 2005 (9.x) and later.
  - TCP/IP is installed as a SQL Server protocol, but it is not listening on the port specified in the JDBC connection URL. The default port is 1433, but SQL Server can be configured at product installation to listen on any port. Make sure that SQL Server is listening on port 1433. Or, if the port has been changed, make sure that the port specified in the JDBC connection URL matches the changed port. For more information about JDBC connection URLs, see Building the connection URL.
  - The address of the computer that is specified in the JDBC connection URL does not refer to a server where SQL Server is installed and started.
  - The networking operation of TCP/IP between the client and server running SQL Server is not operable. You can check TCP/IP connectivity to SQL Server by using telnet. For example, at the command prompt, type `telnet 192.168.0.0 1433` where 192.168.0.0 is the address of the computer that is running SQL Server and 1433 is the port it is listening on. If you receive a message that states "Telnet cannot connect," TCP/IP is not listening on that port for SQL Server connections. Use the SQL Server Network Utility for SQL Server 2000 (8.x), or the SQL Server Configuration Manager for SQL Server 2005 (9.x) and later to make sure that SQL Server is configured to use TCP/IP on port 1433.
  - The port that is used by the server has not been opened in the firewall. This includes the port that is used by the server or optionally, the port associated with a named instance of the server.
- The specified database name is incorrect. Make sure that you are logging on to an existing SQL Server database.
- The user name or password is incorrect. Make sure that you have the correct values.

- When you use SQL Server Authentication, the JDBC driver requires that SQL Server is installed with SQL Server Authentication, which is not the default. Make sure that this option is included when you install or configure your instance of SQL Server.

See also

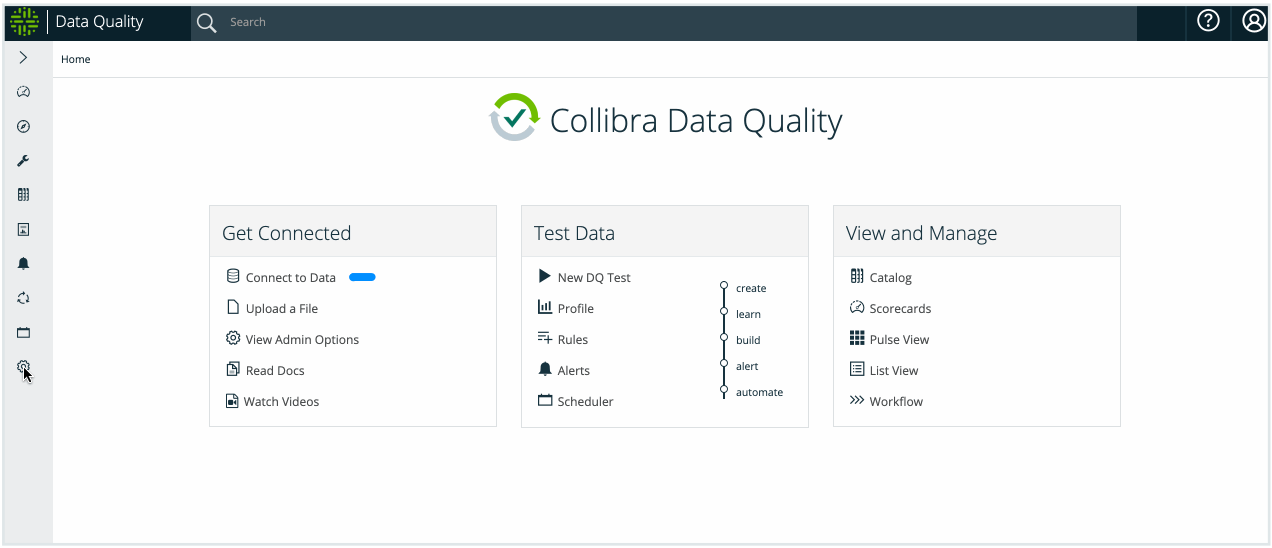
Diagnosing problems with the JDBC driver

Connecting to SQL Server with the JDBC driver

FAQ

SQL Server represents RDS, Azure SQL, and traditional SQL Server installations

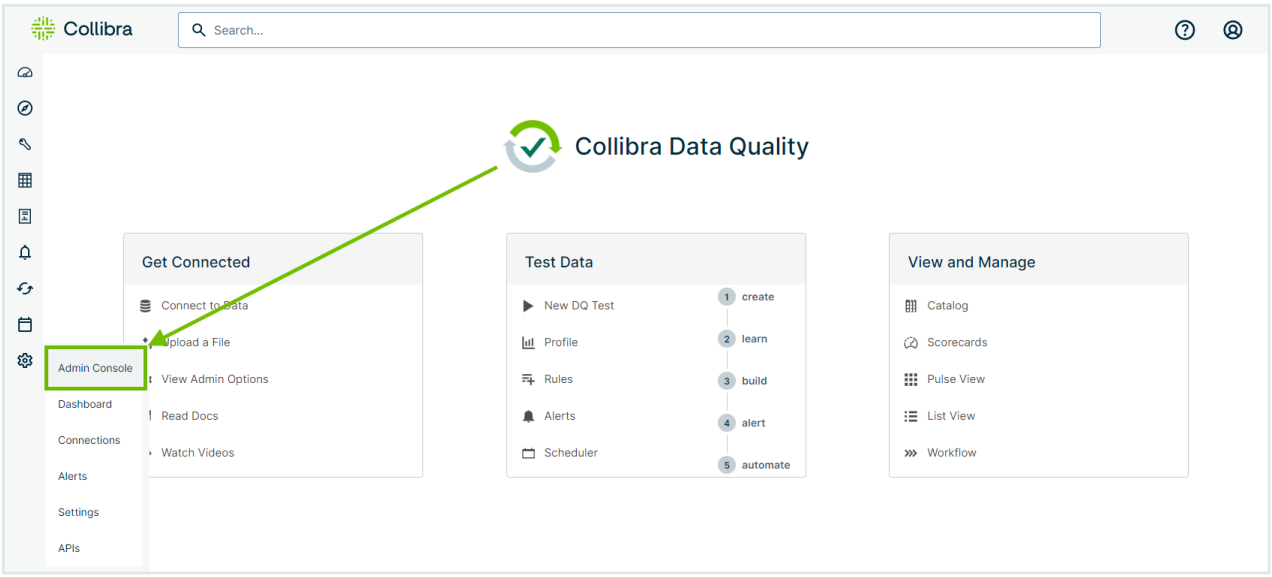
Adding Connections



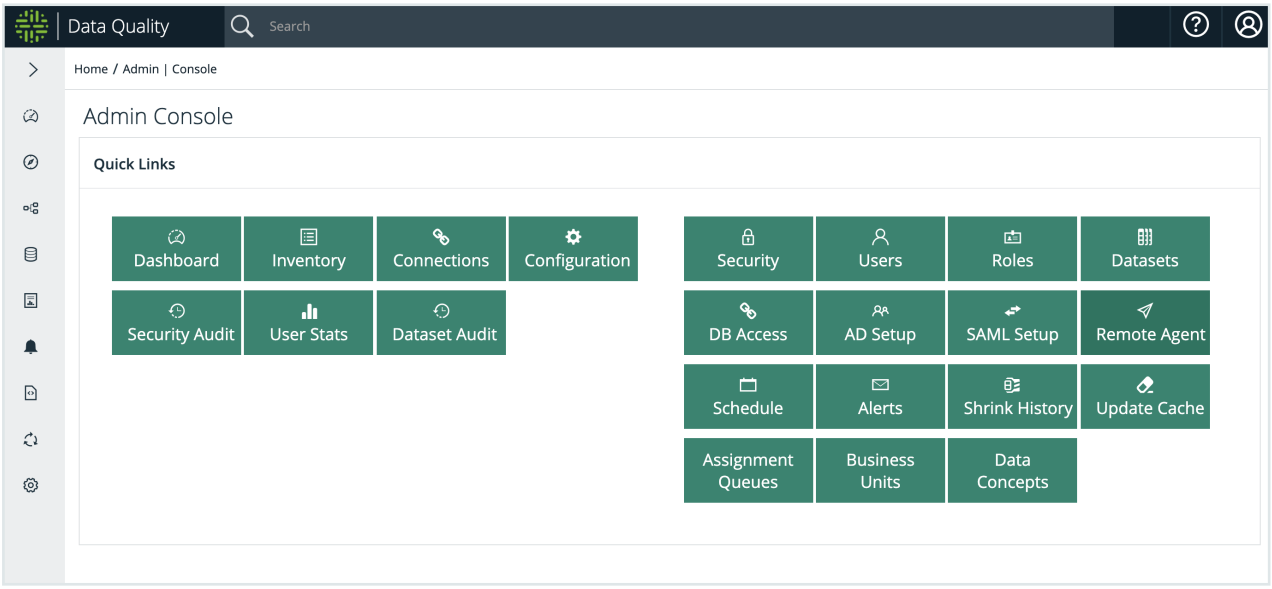
How to Add DB Connection via UI

We will add a connection named `metastore` that connects to local Postgres server (`localhost:5432/postgres`)

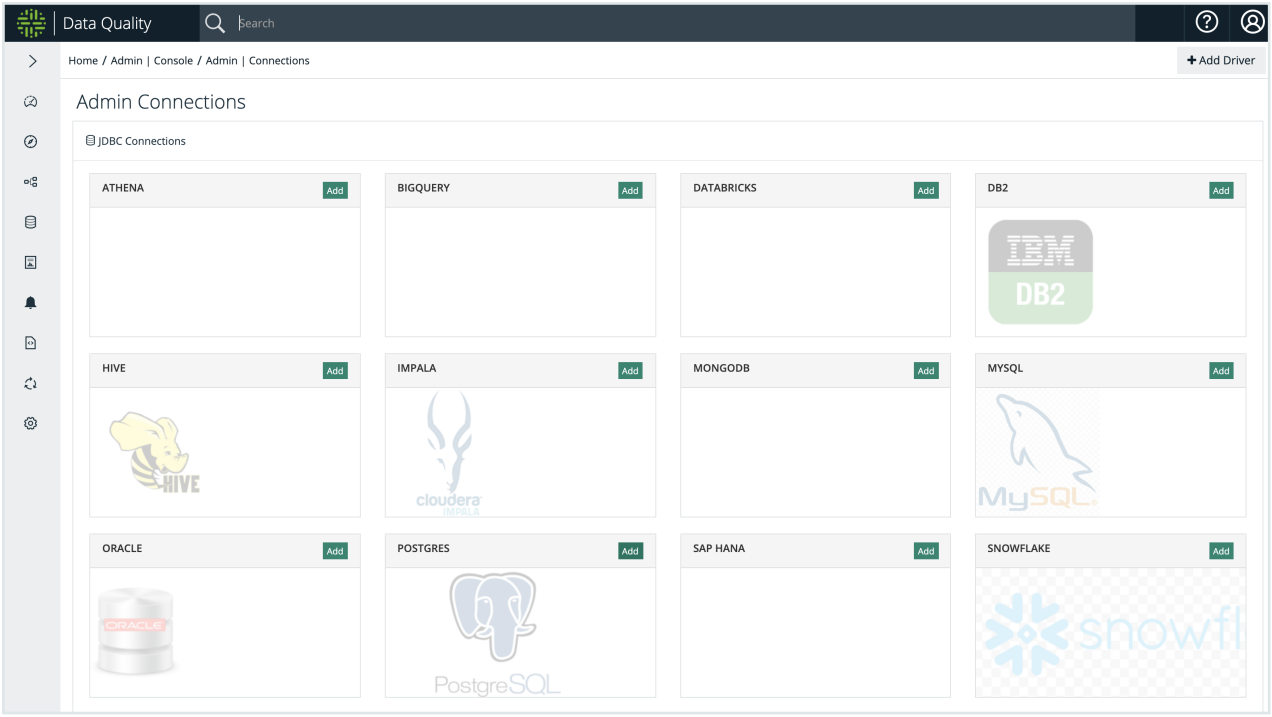
- Login to DQ Web and navigate to Admin Console.



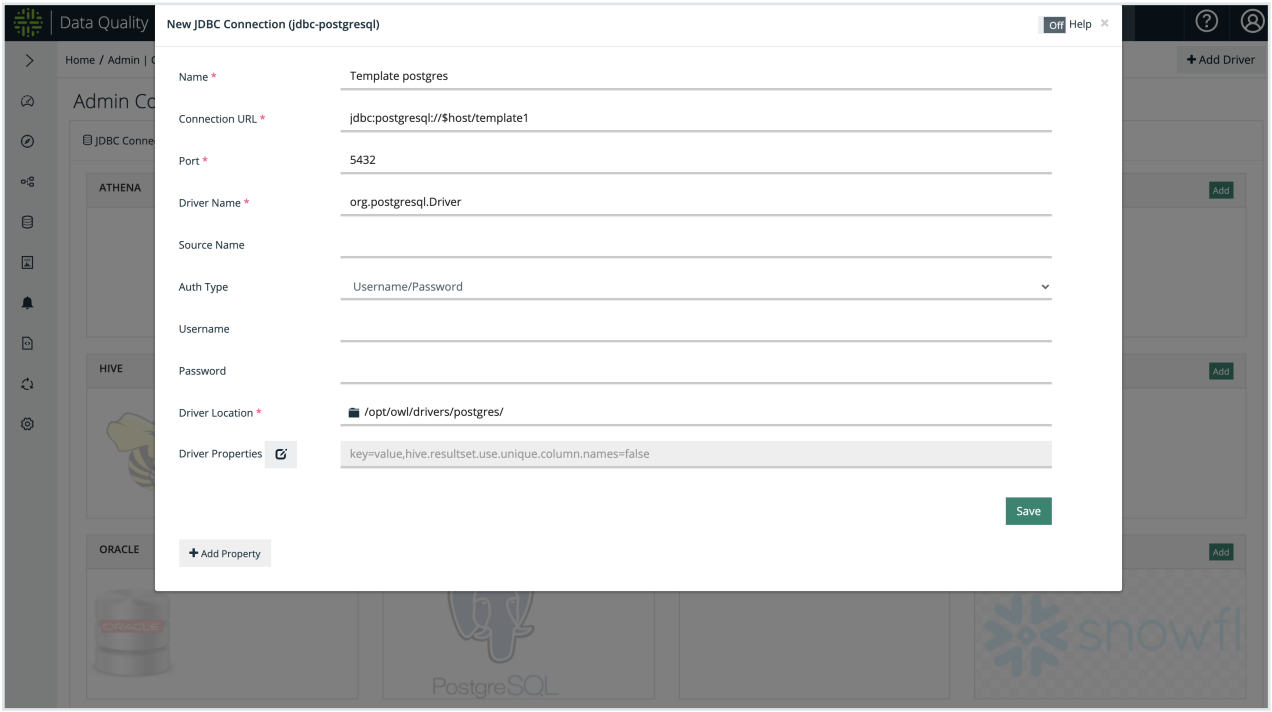
- From the Admin Console, click on the Connections tile.



- Click on Add button in Postgres box to add a Postgres connection.

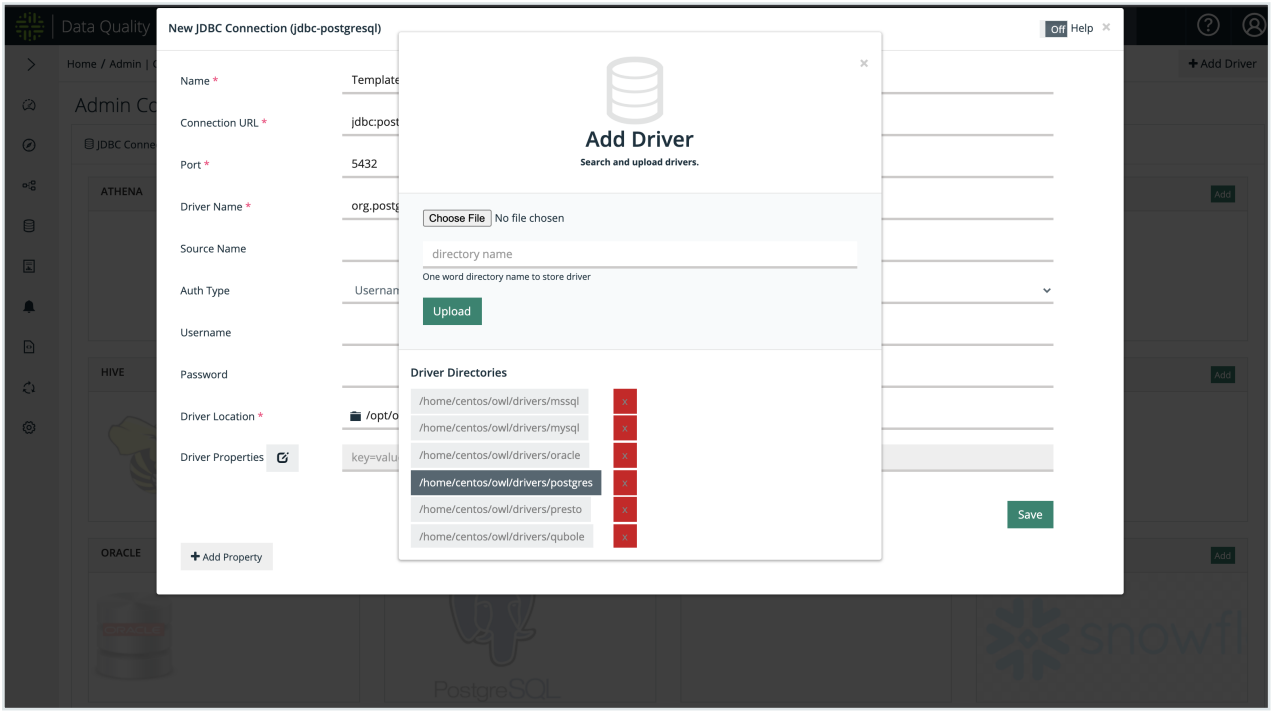


Default Postgres JDBC template connection is shown. This modal is populated with basic values what Postgres connection setting should look like.

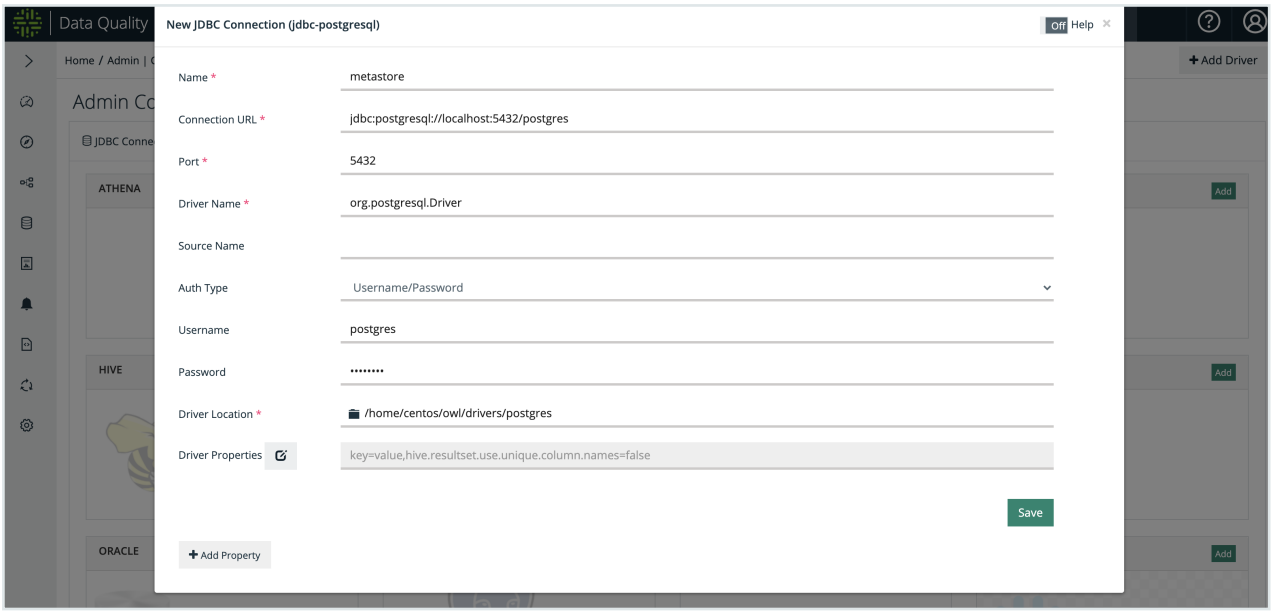


Replace the Connection URL to point to the Postgres server you want to run DQ Jobs against. In this example, `jdbc:postgresql://localhost:5432/postgres`

Also change Driver Location to the JDBC Driver for Postgres in your installation. Click on the folder icon and click on Postgres driver path. These Driver Directories are default JDBC Drivers provided by DQ installation (usually in `$OWL_BASE/owl/drivers/*`)



The following screenshot shows what the new connection setting should look like. Make sure to provide the correct Postgres Username and Password (if using Username/Password for authentication). Press Save to continue. ***This action will attempt to establish a connection.***



Link Connection to Agent

> Note Make sure to Agent to a DQ Agent, if required.

Remote file connections

About remote file connections

This section is an overview of the supported data file formats and the limitations of connecting to a remote file.

Supported file types

File formats differ in structure, so you might need to prepare your data before establishing a connection.

Type	File structure	Notes
Delimited (CSV, TSV, etc.)	Structured	The default delimiter is comma (for example, CSV).
Parquet	Structured	
Avro	Structured	
JSON	Semi-structured	
ORC	Semi-structured	
XML	Semi-structured	
Delta	Semi-structured	

Supported delimiters

The following table is a list of supported delimiters available in the Delimiter dropdown menu.

Type	Format	Description
Comma	CSV	`,` is used to separate values in the file. This is the default delimiter for files.
Tab	TSV	`tab` is used to separate values in the file.
Semicolon	CSV	`;` is used to separate values in the file.
Double Quote	CSV	`"` is used to separate values in the file.
Single Quote	CSV	`'` is used to separate values in the file.
Pipe	TXT	`|` is used to separate values in the text file.

Type	Format	Description
SOH	TXT	A Unicode character 'START OF HEADING' (U+0001) is an invisible control character.
Custom	N/A	Add a custom delimiter. Support for custom delimiters may vary.

Known Limitations

- DQ Jobs that run on remote file connections with headers that contain white spaces fail with a requirement failed exception message. A possible workaround is to edit the DQ Job command line in the Run CMD tab and place single quotes `''` around the column name in `-q` and double quotes `""` around the contents of the `-header` flag.
- Filtergram on the Data Preview tab is not available for any remote file connection. Currently, there is not a workaround for this limitation.
- Array and nested array datatypes in JSON files are not supported.

Connecting to Hadoop Distributed File System (HDFS)

Prerequisites

To configure the HDFS connector, you need:

- Admin permissions in your Collibra DQ instance.
- Access to an HDFS cluster.

Steps

1. In the main menu, hover over the gear icon and click **Connection**.
   » The Connections page opens.
2. Scroll down to the HDFS card.
3. Click the Add button to add a new HDFS connection.
   » The New Remote File Connection (HDFS) modal opens.

4. Enter the values for each property.

Property	Description
Name	The unique name of your HDFS connector.
Connection URL	The HDFS URL used for your connection.
Target Agent	The target agent lets you select an agent for your connection.
Auth Type	The method used to authorize your connection. **Note**: If you use an Unsecured Auth Type, no other authorization fields are required. This is not recommended.
Principal	The service principal used to let Collibra Data Quality access your connection.
Keytab	The keytab used to authorize your connection. **Note**: Only applicable when you select Keytab as the Auth Type.
TGT	The Ticket Granting Ticket used to authorize your connection. **Note**: Only applicable when you select TGT Cache as the Auth Type.
Driver Properties	The configurable driver properties for your connection. **Note**: This is an optional configuration.

5. Click **Save** to establish your connection.

What's next?

After you save your HDFS connection:

- A confirmation message tells you that your connection is saved and valid.
- You can immediately access your HDFS connection Explorer (no-code).
- Begin to Profile (automatic).

Connecting to Azure Data Lake Storage

This section contains an overview of Azure Data Lake Storage Gen2 (ADLS).

General information

Azure Data Lake Storage is designed for big data analytics workloads with support for POSIX permissions, hierarchical file systems, and faster read times. It is built on top of the Azure Blob Storage and leverages the ABFS driver.

Field	Description
Data source	Azure Data Lake Storage Gen2 (ADLS)
Supported versions	N/A
Connection string	`abfs://`
Packaged?	✓ Yes
Certified?	✓ Yes
Supported features	
Pushdown	✗ No
Estimate job	✓ Yes
Filtergram	✓ Yes
Analyze data	✓ Yes
Spark agent	✓ Yes
Yarn agent	✓ Yes

Recommended and required connection properties

Required	Connection Property	Type	Value
✓ Yes	Name	Text	The unique name used for your connection.

Required	Connection Property	Type	Value
✓ Yes	Connection URL	String	The connection string value of your ADLS connection. `abfs://<file system>@<account name>.d-fs.core.windows.net/<path>/file name` Tip See the Connection URL elements section for more information on the syntax.
✕ No	Target Agent	Text	The Agent used to submit your DQ Job.
✓ Yes	Auth Type	Option	The method to authenticate your connection. Note The configuration requirements are different depending on the Auth Type you select. See Authentication for more details on available authentication types.
✕ No	Driver Properties	String	The configurable driver properties for your connection.

Minimum user permissions

To create an Azure Data Lake Storage connection, you need to have ROLE_ADMIN assigned to you in Collibra DQ and access to Azure Data Lake Storage.

Authentication

Select one of the following authentication options:

Key
Service Principal

Field	Description
Storage Account	The Azure Storage account name.
Key	The authentication key for the storage account.
TenantId	The Microsoft Online tenant being used to access data. If not specified, your default tenant will be used.
ClientId	The client Id assigned when you register your application with an OAuth authorization server.
Secret	The client secret assigned when you register your application with an OAuth authorization server.

Connection URL elements

Field	Description
File scheme	The `abfs` protocol is used as the scheme identifier.
File system	The parent location that holds the files and folders. This is the same as containers in the Azure Storage Blob service.
Account name	The name given to your storage account during creation.
Path	A forward slash delimited `/` representation of the directory structure.
File name	The name of the individual file. This parameter is optional when you address a directory.

Connecting to Azure Blob Storage

This section contains an overview of Azure Blob Storage.

General information

Azure Blob Storage is the object store for different unstructured data such as text, video, and audio and uses flat object storage. The WASBS driver is used for Azure Blob Storage.

Field	Description
Data source	Azure Blob Storage
Supported versions	N/A
Connection string	`wasbs://`
Packaged?	✓ Yes
Certified?	✓ Yes
Supported features	
Pushdown	✗ No
Estimate job	✓ Yes
Filtergram	✓ Yes
Analyze data	✓ Yes
Spark agent	✓ Yes
Yarn agent	✓ Yes

Recommended and required connection properties

Required	Connection Property	Type	Value
✓ Yes	Name	Text	The unique name used for your connection.

Required	Connection Property	Type	Value
✓ Yes	Connection URL	String	The connection string value of your ADLS connection. `wasbs://<container>@<account name>.dfs.core.windows.net/<path>/file name` Tip See the Connection URL elements section for more information on the syntax.
✕ No	Target Agent	Text	The Agent used to submit your DQ Job.
✓ Yes	Auth Type	Option	The method to authenticate your connection. Note The configuration requirements are different depending on the Auth Type you select. See Authentication for more details on available authentication types.
✕ No	Driver Properties	String	The configurable driver properties for your connection.

Minimum user permissions

To create an Azure Blob Storage connection, you need to have ROLE_ADMIN assigned to you in Collibra DQ and access to Azure Blob Storage.

Authentication

Select one of the following authentication options:

Key
SAS

Field	Description
Storage Account	The Azure Storage account name.
Key	The authentication key for the storage account.
Storage Account	The Azure Storage account name.
Shared Access Signature	The string value of the Azure Blob shared access signature.

Connection URL elements

Field	Description
File scheme	The `wasbs` protocol is used as the scheme identifier.
Container	The parent location that holds the files and folders. This is the same as file system in the Azure Data Lake Storage service.
Account name	The name given to your storage account during creation.
Path	A forward slash delimited `/` representation of the directory structure.
File name	The name of the individual file. This parameter is optional when you address a directory.

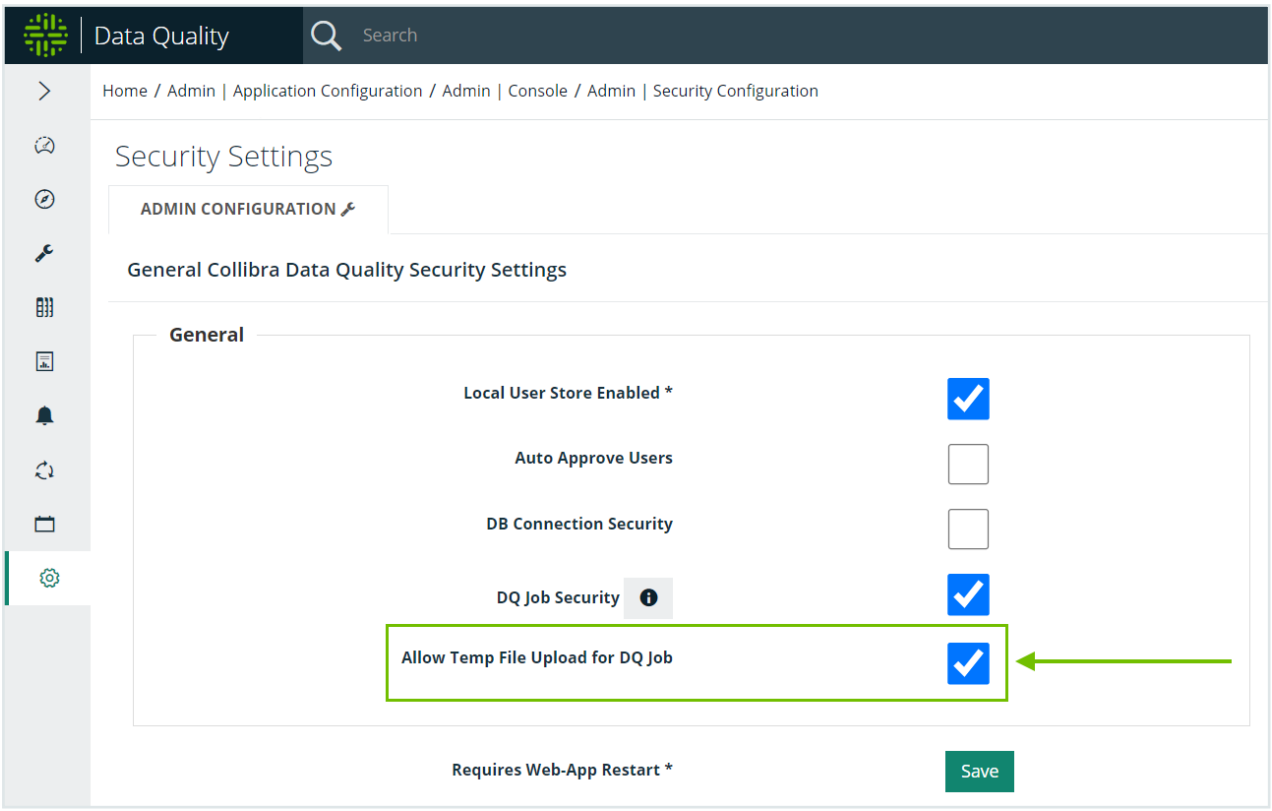
Connecting to Temp Files

Temp files allow you to upload and run a DQ job on files directly from your local machine. This is not a recommended connection for more advanced users but can be useful when you first get started in Collibra Data Quality.

Prerequisites

To connect to a temp file, you need:

- A file in a supported file format saved on your local drive.
- To verify that the Allow Temp File Upload for DQ Job checkbox under Admin Console > Security Configuration is checked.



Steps

1. In the main menu, click the **Explorer** button.
   » The Explorer expands.
2. Click **Temp Files**.
   » Temp Files expands.
3. Click **Add Temp File**.
   » Upload Temp File alert opens.
4. Click **Choose File**.
5. Select a file from your local drive.
6. Click **Open**.
   » Your file loads into the Temp Files folder.
7. In the Temp Files folder in the application, select your file.
8. Click **Create DQ Job**.

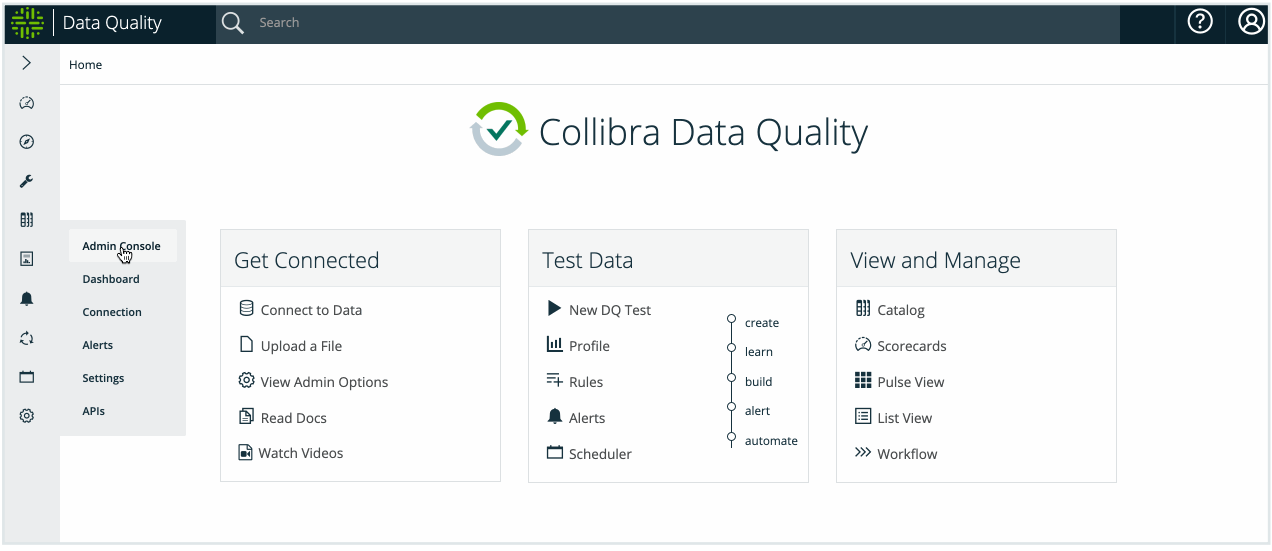
9. Verify your file information and enter the required information.
10. Click **Load File**.
    » The application automatically reads your file and opens Scope & Range.
11. Select your DQ layers.
    » You can also leave the defaults set.
12. Click **Save & Run**.
13. Enter a unique name and any additional information.
    » Temp files differ from other remote file connections insofar as they do not require an agent to run successfully. This is a legacy component.
14. Click the **Run CMD** tab.
    » This is a bypass step for temp files.
15. Click **Run**.
    » Your job is sent to the Jobs queue.

> Warning All temp files are only temporarily stored in the application. At 11:59 PM EST, all temp files uploaded on a given day are automatically removed.

Known Limitations

Temp files do not run successfully on Kubernetes deployments of Collibra DQ. This is a legacy known limitation wherein container environments do not support temp files.

Adding a connection to a DQ Agent

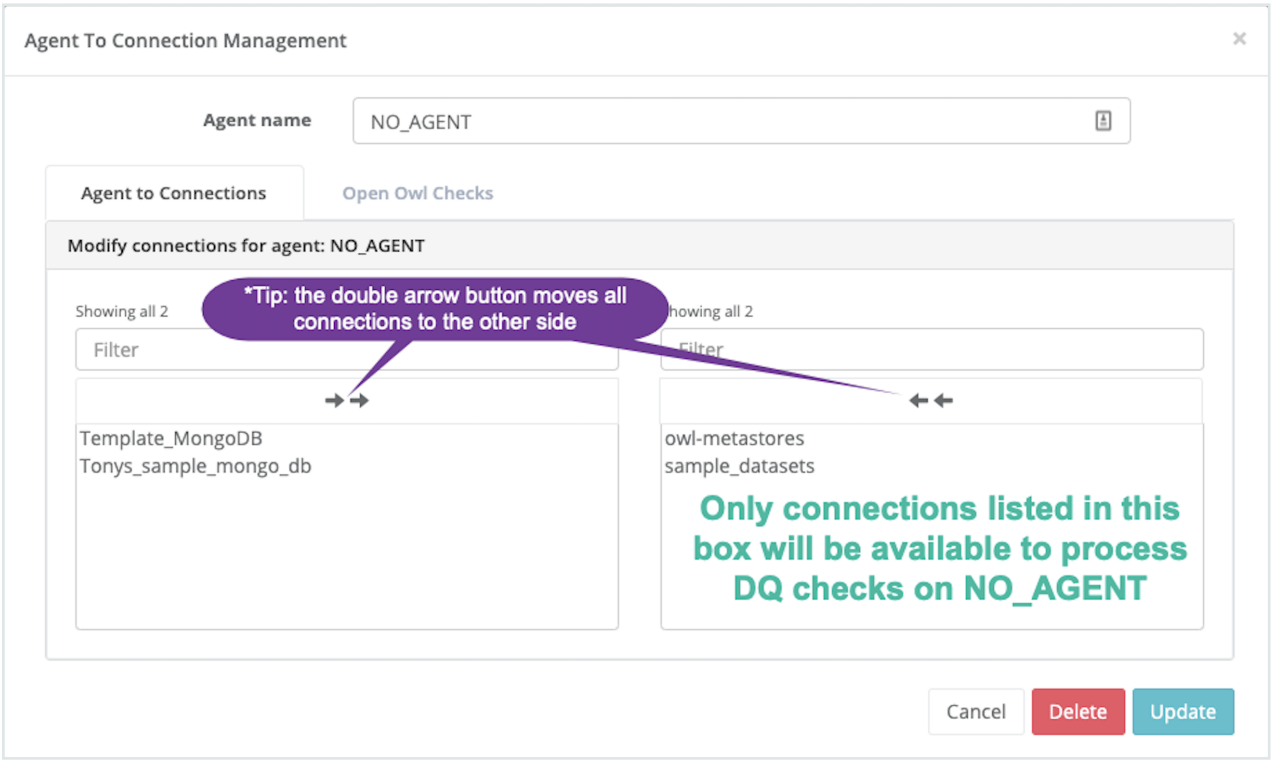


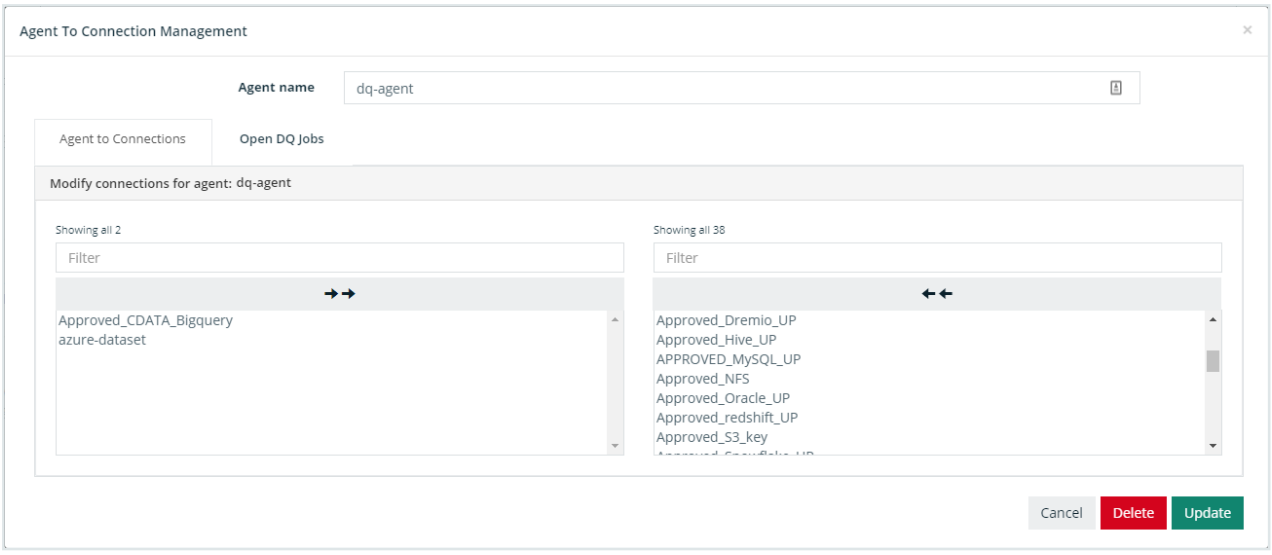
How To Link DB Connection to Agent via UI

When you add a new Database Connection, the DQ Agent must be given the permission to run DQ Job via the specified agent.

From Fig 3, select the chain link icon next to the DQ Agent to establish link to DB Connection. A modal to add give that agent permission to run DQ Jobs by DB Connection name will show (Fig 5). The left-side panel is the list DB Connection names that has not been linked to the DQ Agent. The right-side panel is the list of DB Connection names that has the permission to run DQ Job.

Double click the DQ Connection name to move from left to right. In Fig 5, DB Connection named "metastore" is being added to DQ Agent. Click the "Update" button to save the new list of DB Connections.Fig 5: Adding DB Connection named "metastore" to the DQ AgentFig 6: How to add all connections to the selected DQ Agent





Authentication

Kerberos authentication

Kerberos is a network authentication protocol that uses secret-key cryptography to provide secure authentication over unsecured networks.

You can use Kerberos authentication for connecting to a data source, for example Hive, on Kubernetes and YARN deployments of Collibra DQ.

> Note Kerberos authentication is not supported on Standalone deployments of Collibra DQ.

Authentication types

We support Kerberos password manager, keytab, password, and ticket granting ticket (TGT) for many of our supported data sources. For more information on which data sources support Kerberos authentication, see the supported data sources page.

> Note Ensure that you configure the krb5.conf file before you attempt to authenticate with Kerberos.

Create a krb5.conf file

Before you can authenticate connections with Kerberos, you need to create a storage space for your Kerberos configuration information. This section shows you how to create a krb5.conf file to store your Kerberos information.

Steps

1. Open a terminal session.
2. Run the following command:

```
sudo vi /etc/krb5.conf
```

3. Ensure that you have the correct information for the following, as shown also in the example screenshot below:
   a. default_realm
   b. default_ccache_name
   c. [realms]
   d. kdc
   e. [domain_realm]

```
[logging]
 default=FILE:/var/log/krb5kdc.log

[libdefaults]
 default_realm= CW.COM
 default_ccache_name = FILE:/<root-folder>/krb5/krb5.ccache
 dns_lookup_realm = false
 dns_lookup_kdc = true
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true

[realms]
 CW.COM = {
 kdc = <admin-ip-address>
 admin_server = <admin-ip-address>
}

[domain_realm]
 .cw.com = CW.COM
 cw.com = CW.COM
~
~
~
```

4. Run the following command to add your hostname:

```
sudo vi /etc/hosts
```

5. Enter a hostname, for example `localhost`.

> Note Ensure that the host where the krb5.conf is configured can reach the KDC server.

Create Kerberos authentication files

In order to authenticate with Kerberos, you need to create the files with which Collibra DQ will access to authenticate your datasource connections.

Prerequisites

You have:

- A krb5.conf file to store your Kerberos configuration information.
- Admin user permissions.

Steps

1. Open a terminal session.
2. Run the following command with your own username:

```
kinit <your-username>
```

3. Enter the password of your keytab file.

> Note If any encryption-related messages appear, you can safely ignore them.

4. Run the following command with your file name:

```
kinit -c /tmp/<your-file-name>.keytab
```

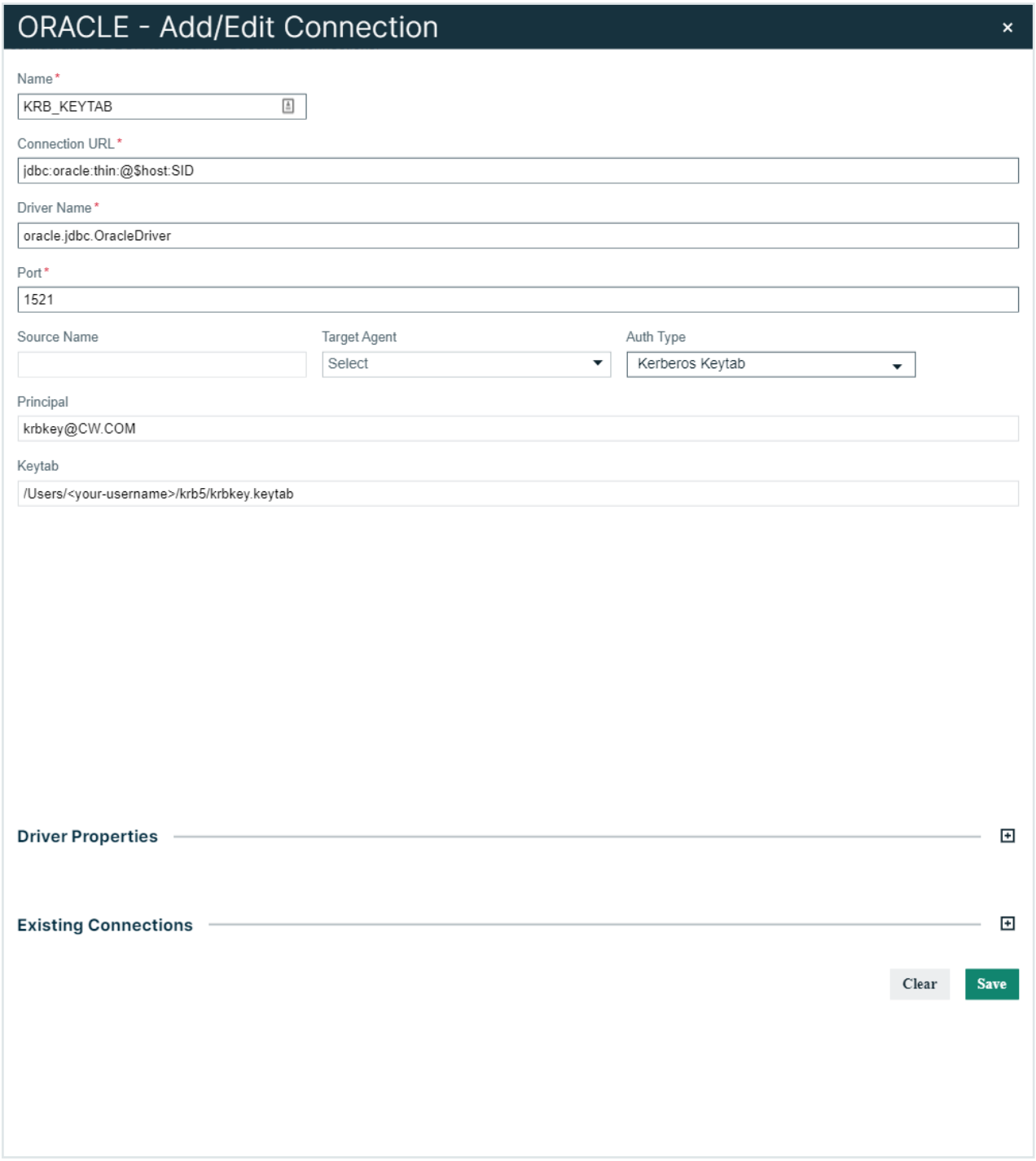
> Note You can define the location of your file, for example `/tmp/`.

5. Run the following command with your username to view the details of your keytab file:

```
klist <your-username>
```

View example screenshot...[1]

1



1. Open a terminal session.
2. Run the following command with your username:

```
kinit <your-username>
```

3. Enter the password of your TGT file.
4. Run the following command with your file name:

```
kinit -c /tmp/<your-file-name>.tgt
```

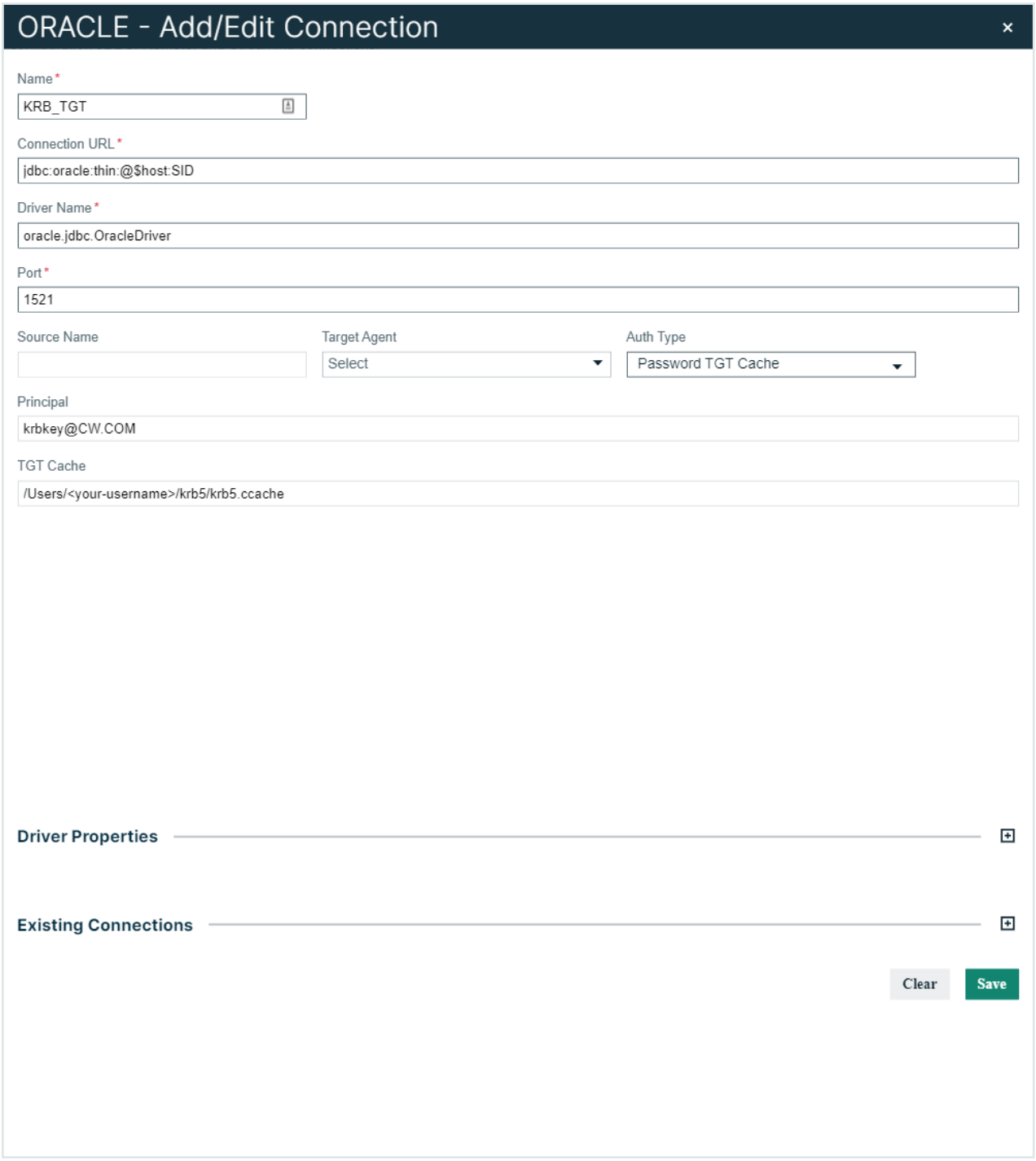
> Note You can define the location of your file, for example `/tmp/`.

5. Run the following command with your username to view the details of your TGT file:

```
klist <your-username>
```

View example screenshot...[1]

1



Collibra DQ Features

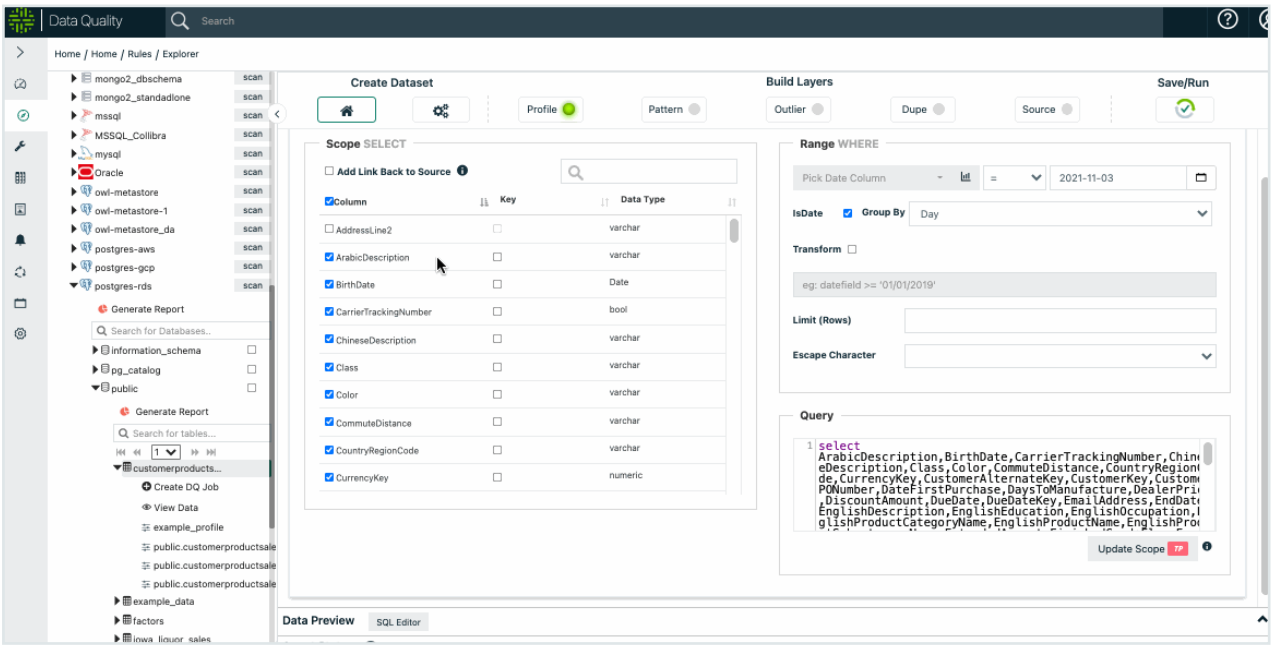
Profile (automatic)

Create profiles based on a table, view, or file.

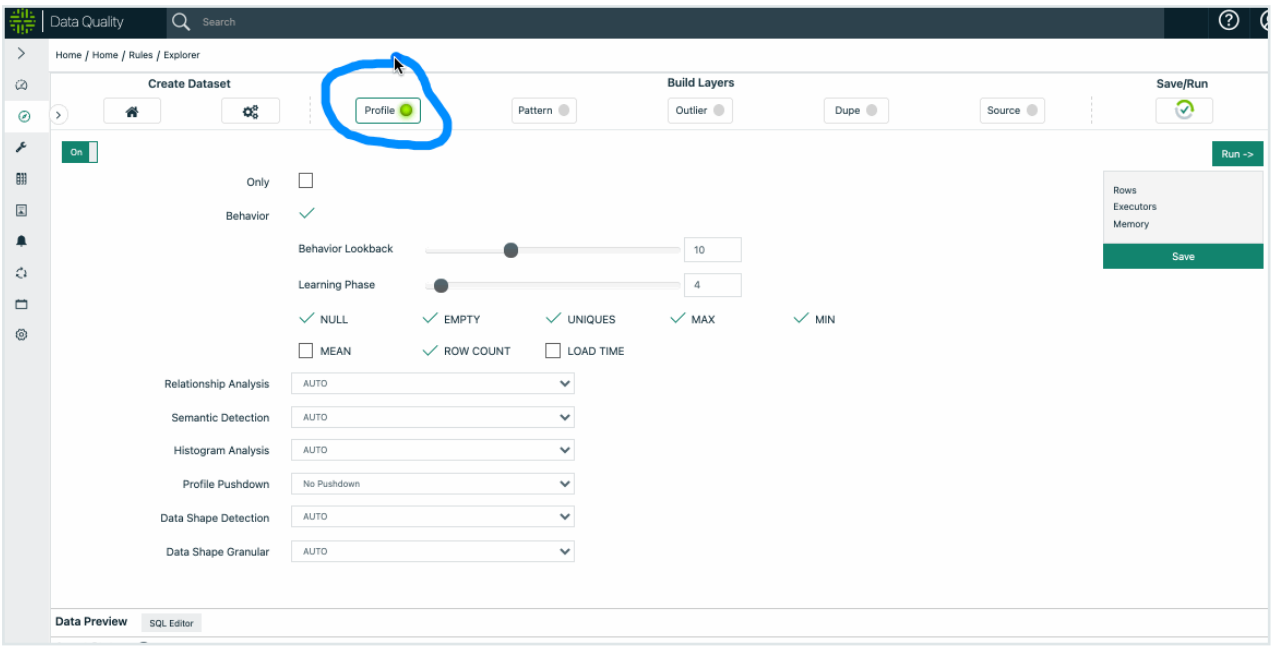
> Note Users have the option to scan the entire dataset or users can apply custom filtering to select the depth (row filtering) and width (columns).

Select the Scope

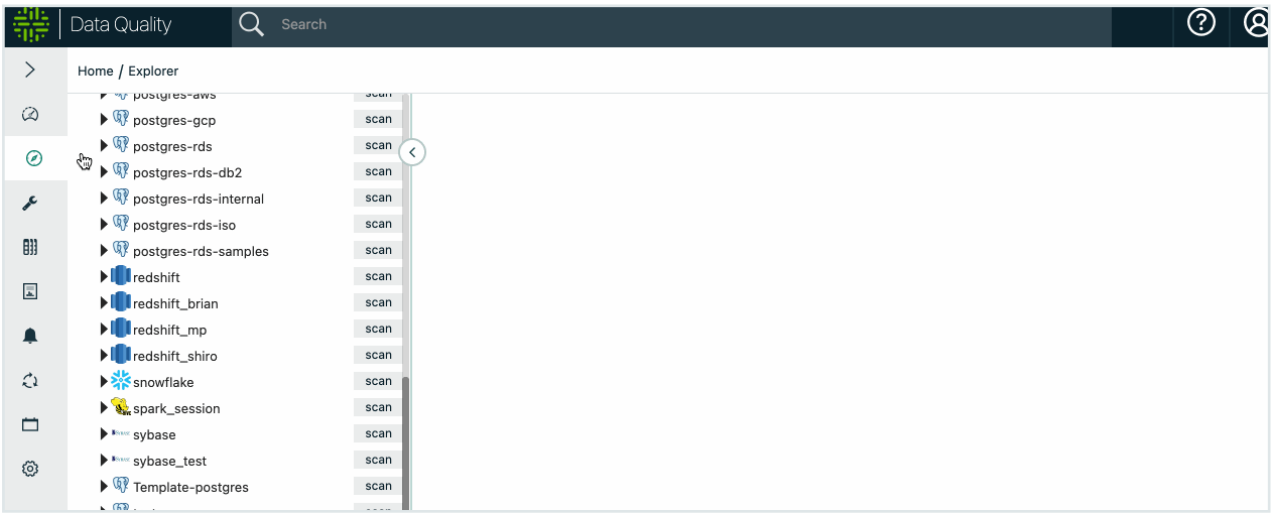
You can find detailed instructions about selecting the scope in the Explorer section. You can run limits, by time, or full table scans if you have enough resources.



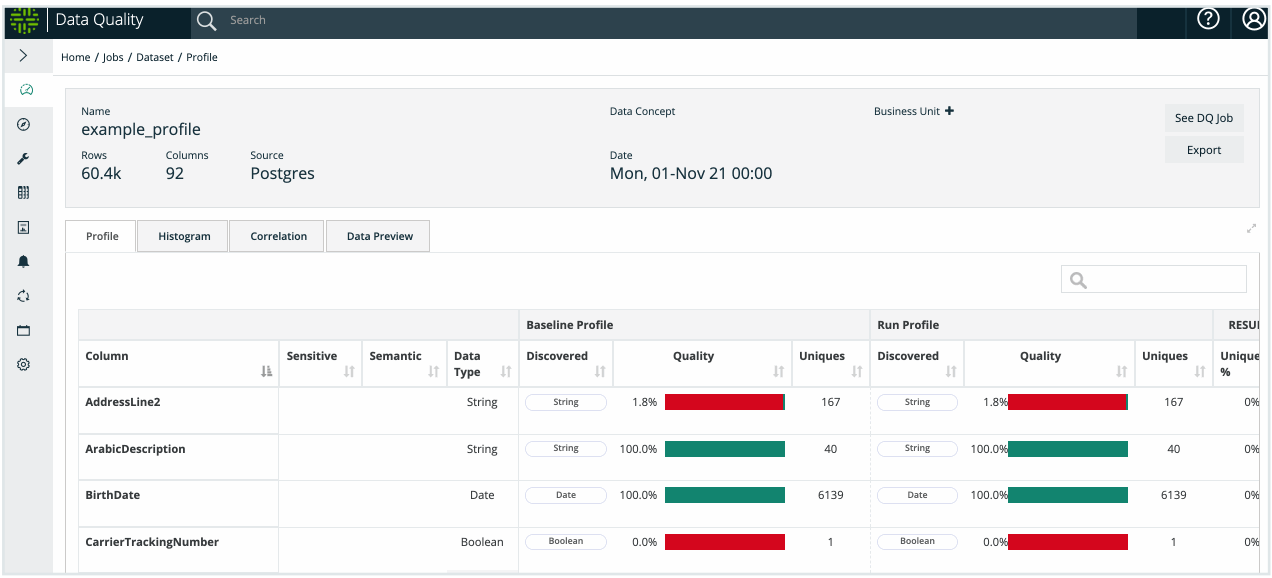
Select Options (or leave defaults)



Save / Run



View the Results



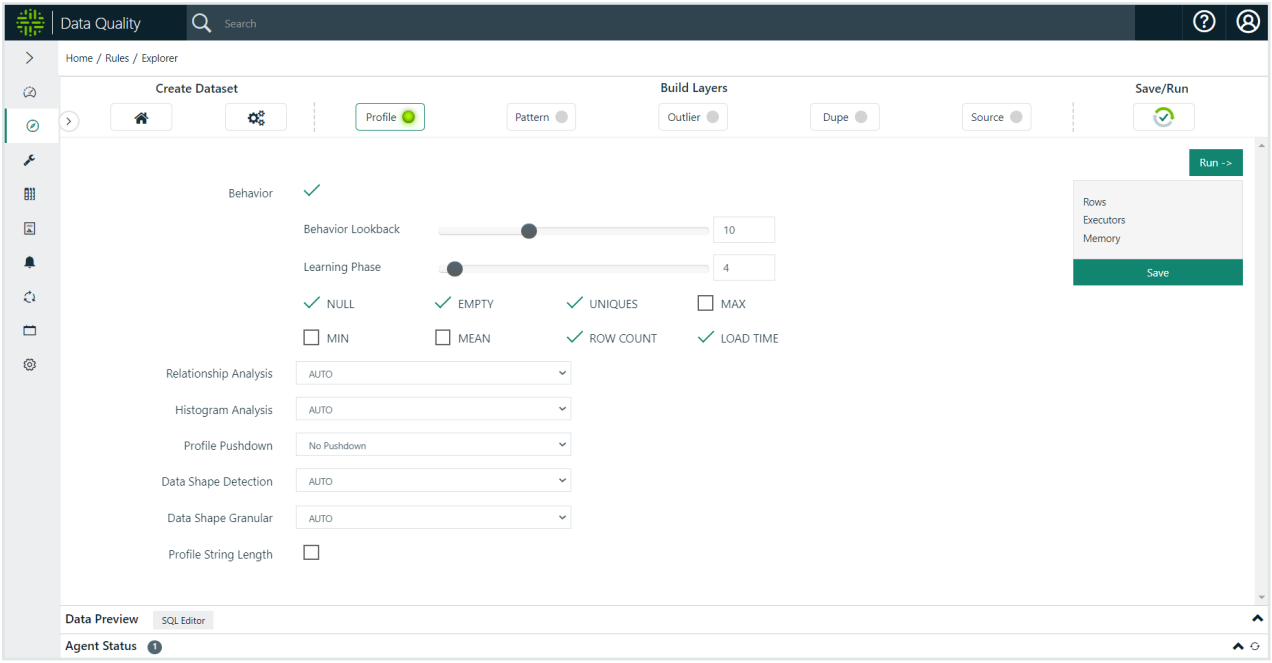
Automatically Profile

Collibra DQ automatically profiles data sets over time to enable drill-ins for detailed insights and automated data quality. A profile is just the first step towards an amazing amount of auto discovery. Visualize segments of the data set and how the data set changes over time.

Collibra DQ offers click or code options to run profiling.

Data Set Profile

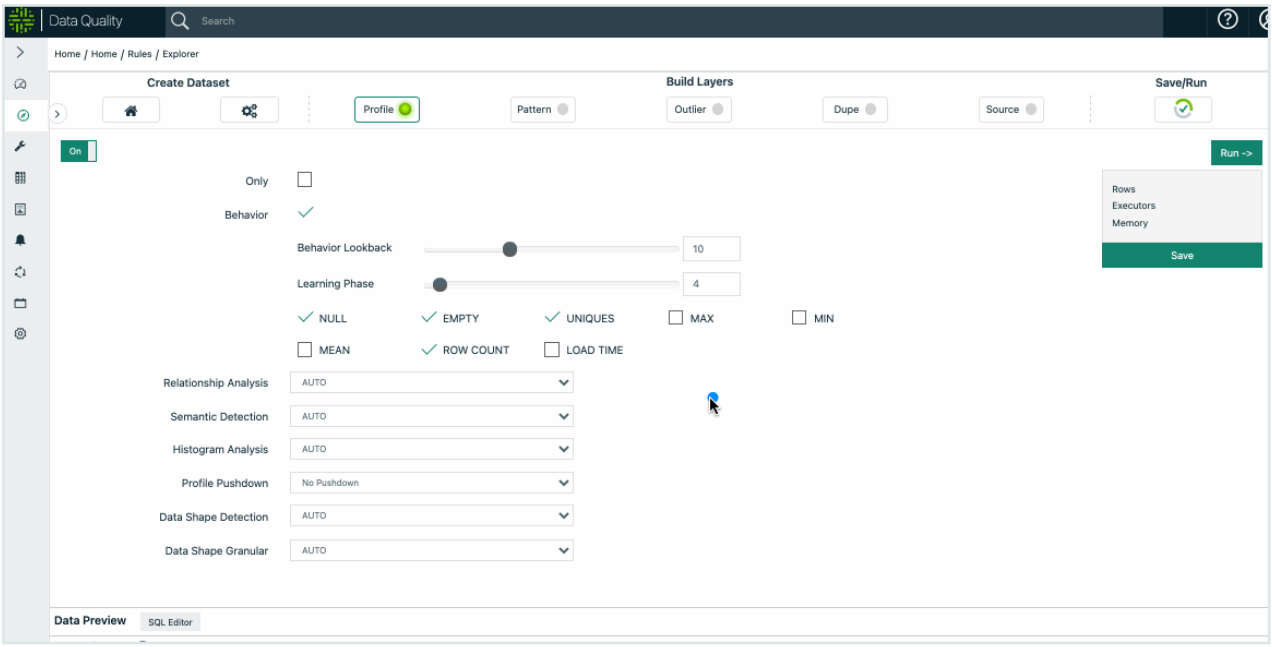
Collibra DQ creates a detailed profile of each dataset under management. This profile will later be used to both provide insight and automatically identify data quality issues.



Pushdown Profiling

Collibra DQ can compute the Profile of a data set either via Spark (default) or a Data Warehouse (Profile Pushdown) where the data lives as the engine. When the Profile is computed using the datasource DBMS the user can choose two levels of pushdown:

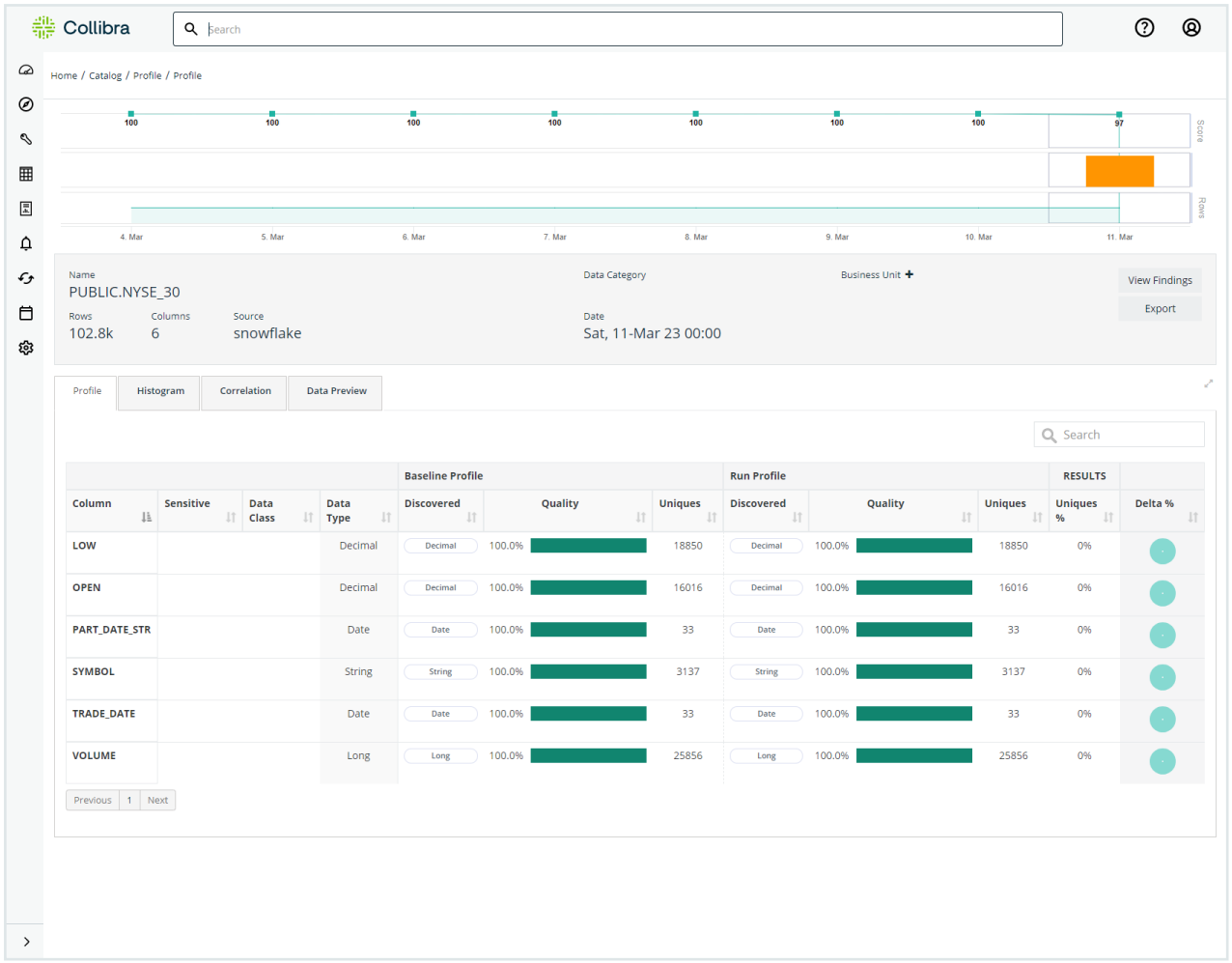
- Full Profile - Perform full profile calculation except for TopN
- Count - Only perform row and column counts



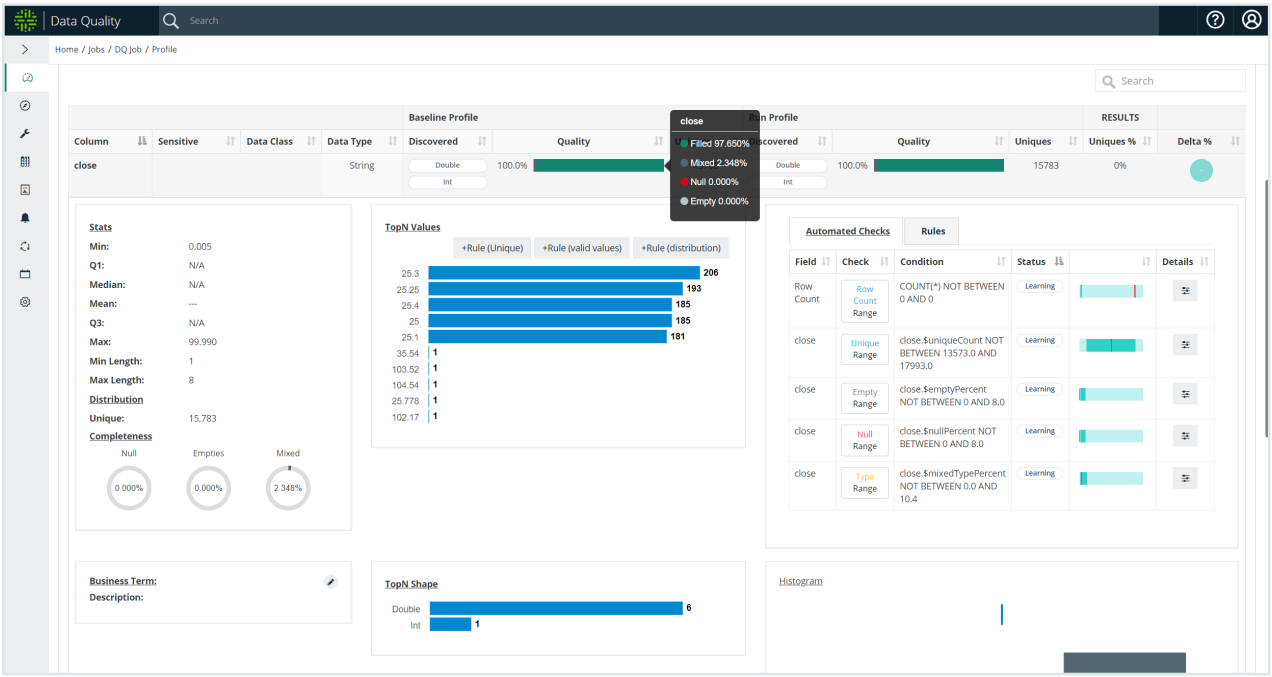
Note The following DBMS systems are supported for "Profile Pushdown":

- Impala
- Hive
- Snowflake
- Presto
- Teradata
- SQL Server
- PostgreSQL
- Redshift
- MySQL
- Oracle
- DB2

Warning Pushdown and parallel JDBC cannot be used together. If you are using pushdown, do not select the parallel JDBC option.


Collibra
Search
Home / Catalog / Profile / Profile
100 100 100 100 100 100 100 97
Score
Rows
4. Mar 5. Mar 6. Mar 7. Mar 8. Mar 9. Mar 10. Mar 11. Mar
Name
PUBLIC.NYSE_30
Rows
102.8k
Columns
6
Source
snowflake
Data Category
Date
Sat, 11-Mar 23 00:00
Business Unit
View Findings
Export
Profile
Histogram
Correlation
Data Preview
Search
Baseline Profile
Run Profile
RESULTS
Column
Sensitive
Data Class
Data Type
Discovered
Quality
Uniques
Discovered
Quality
Uniques
Uniques %
Delta %
LOW Decimal Decimal 100.0% 18850 Decimal 100.0% 18850 0%
OPEN Decimal Decimal 100.0% 16016 Decimal 100.0% 16016 0%
PART_DATE_STR Date Date 100.0% 33 Date 100.0% 33 0%
SYMBOL String String 100.0% 3137 String 100.0% 3137 0%
TRADE_DATE Date Date 100.0% 33 Date 100.0% 33 0%
VOLUME Long Long 100.0% 25856 Long 100.0% 25856 0%
Previous
1
Next

Profile Insights



By gathering a variety of different statistics, Collibra DQ's profile can provide a great deal of insight about a data set.

To see the difference between baseline (historical) and current values, Collibra DQ provides a **Delta %** change column. In the Delta % change column, data is represented in a pie chart for quick visualization of the changes.

To elaborate on the quality metrics:

The profile can discover attributes then helps delineate the relative metrics around numeric v. non-numeric discovered.

Metric	Type	Description
Filled	[1] Integer	The percentage of data that is numeric (or non-numeric) in a numeric (or non-numeric) discovered column.
Mixed	[String] Integer	The percentage of data that is non-numeric (or numeric) in a numeric (or non-numeric) discovered column.
Null	[]	The percentage of data that has no value at all.

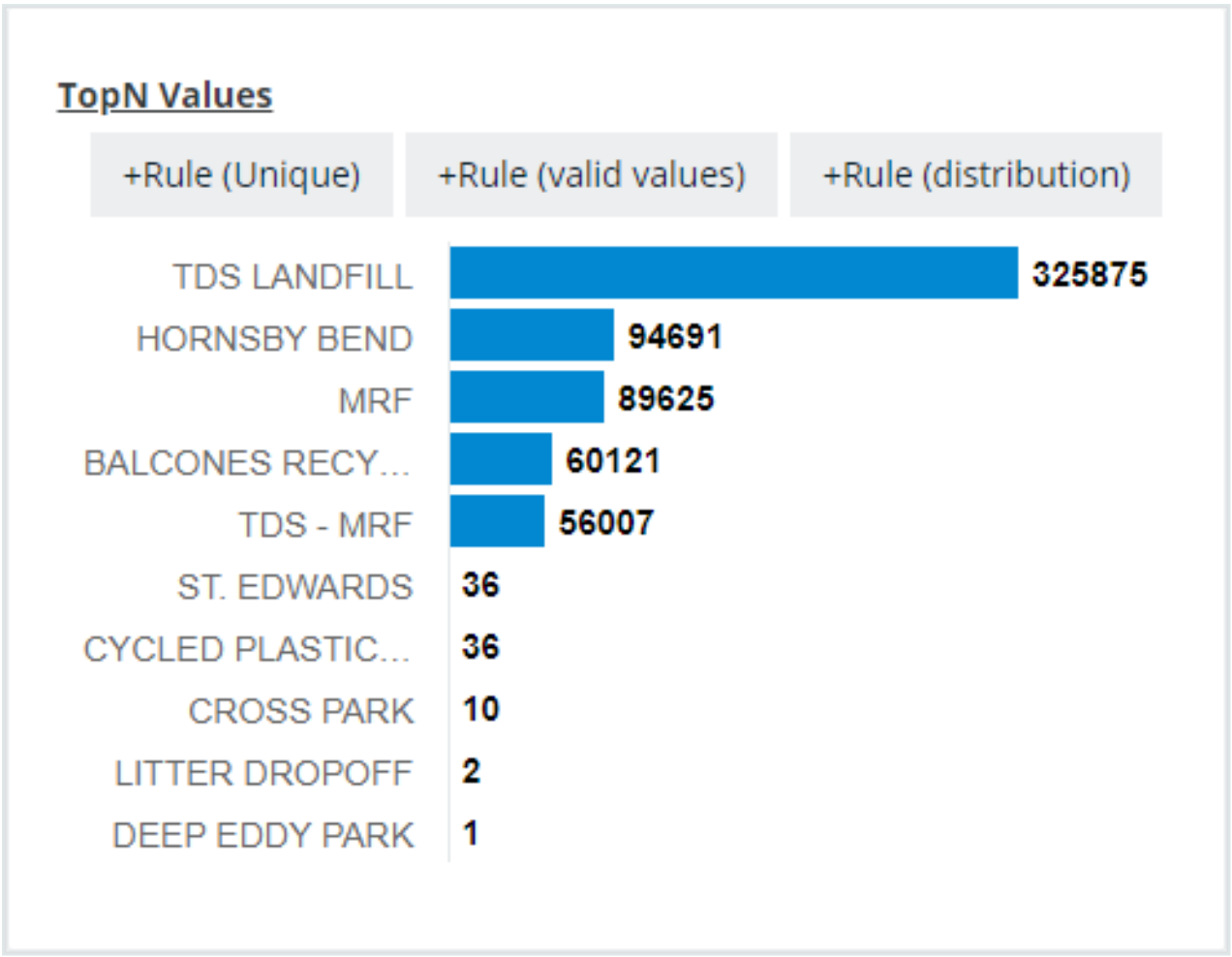
Metric	Type	Description
Empty	[""]	The percentage of data that has a string instance of zero length.

Profile includes the following statistics:

- Actual Datatype
- Discovered Datatypes
- Percent Null
- Percent Empty
- Percent Mixed Types
- Cardinality
- Minimum
- Maximum
- Mean
- TopN / BottomN
- Value Quartiles
- Minimum (String) Length
- Maximum (String) Length

TopN Values

From the **Profile** page in Catalog, you can view a **TopN Values** chart. The TopN Values chart represents the top 10 distinct values that appear most frequently.



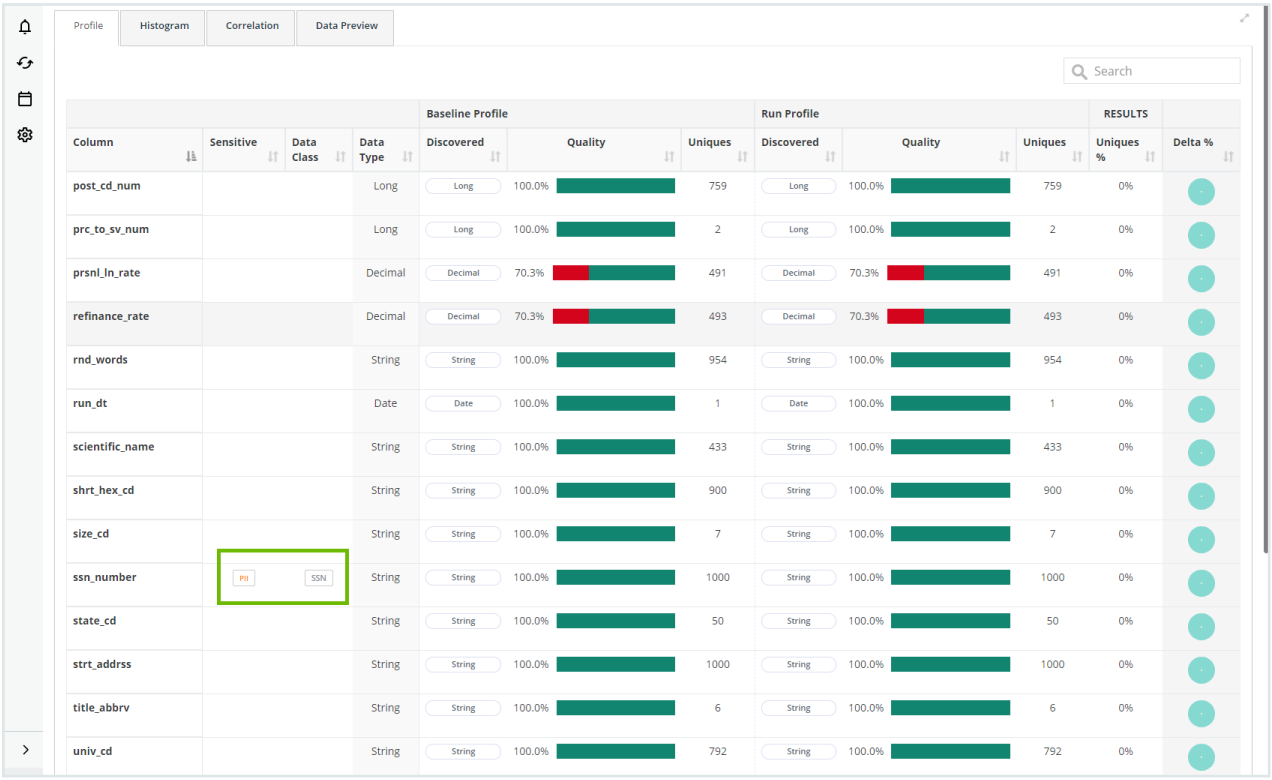
Sensitive Data Detection and Data Class

Collibra DQ can automatically identify any type of common PII columns.

> Note Collibra DQ is able to detect the following types of PII:

- EMAIL
- PHONE
- ZIP CODE
- STATE CD
- CREDIT CARD
- GENDER
- SSN

- IP ADDRESS
- EIN



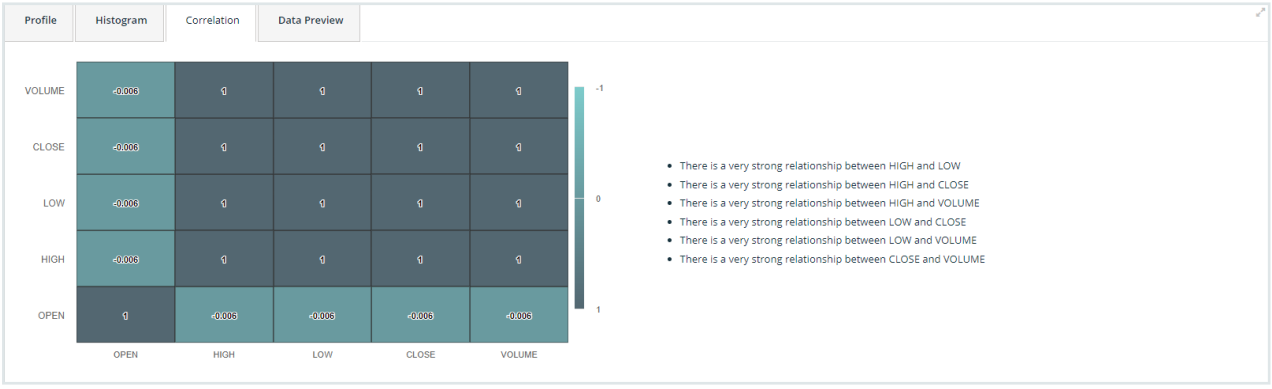
Once detected, you can tag the column from the Profile tab as the discovered sensitive data type and data class and automatically applies a rule. To remove a tag, click the tag and edit from the **Add Data Class/Sensitivity Label** modal.

Histograms

The first step in many data science projects is to segment the data. Collibra DQ automatically does this with histograms on the Histogram tab.

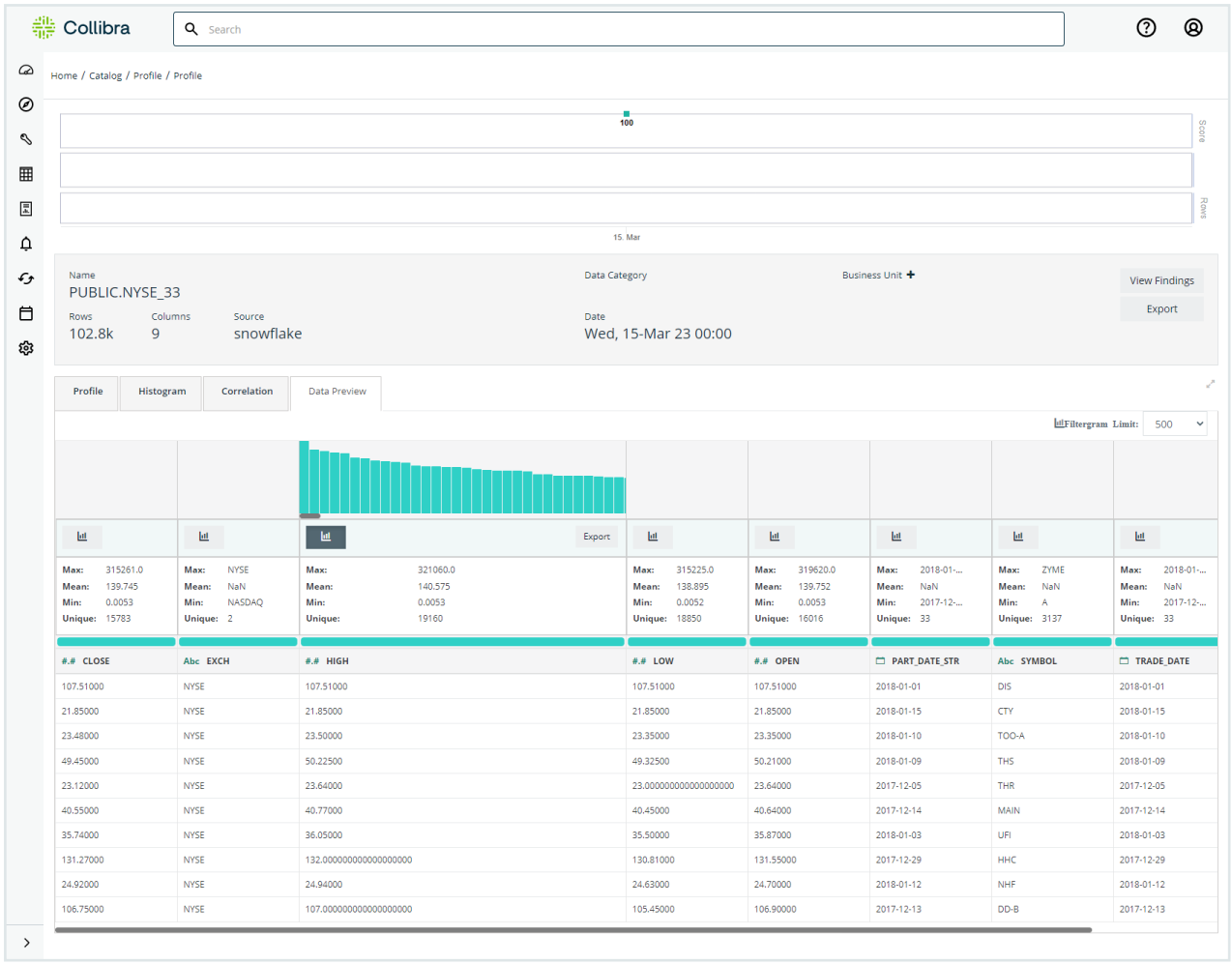
Correlation

The correlation matrix on the Correlation tab lets you discover hidden relationships and measure the strength of those relationships.



Data Preview

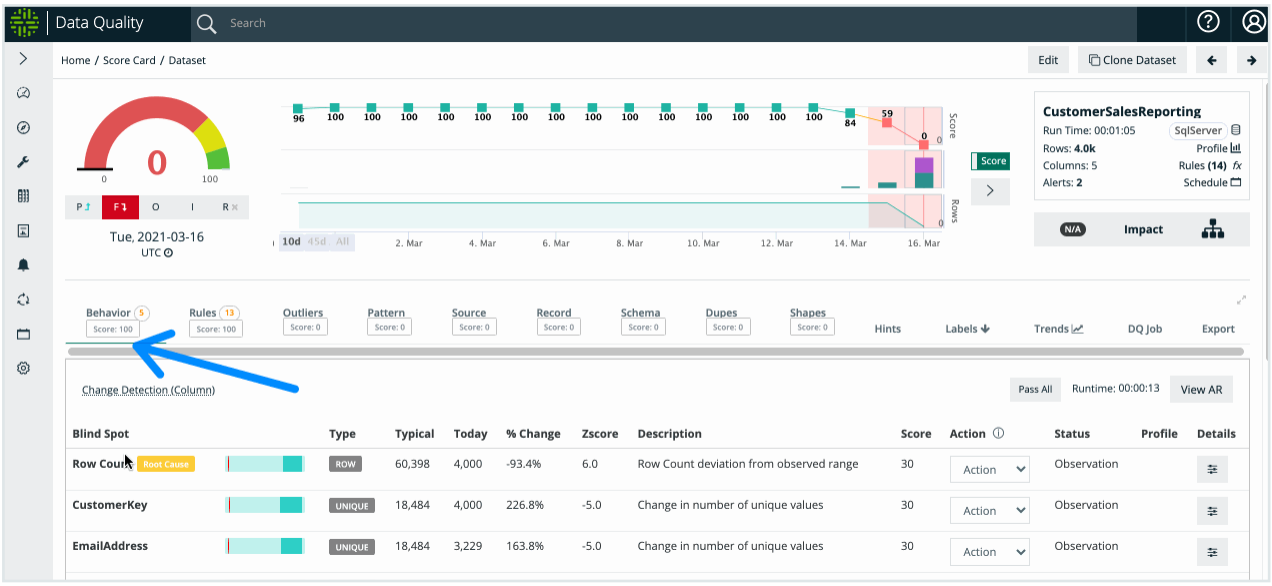
After profiling the data, for those users with appropriate rights, the Data Preview tab provides a glimpse of the dataset and basic insights such as highlights of Shape issues, Outliers (when enabled), and column Filtergram visualization.



Behavior (automatic)

This is commonly referred to as statistical change detection or data observability.

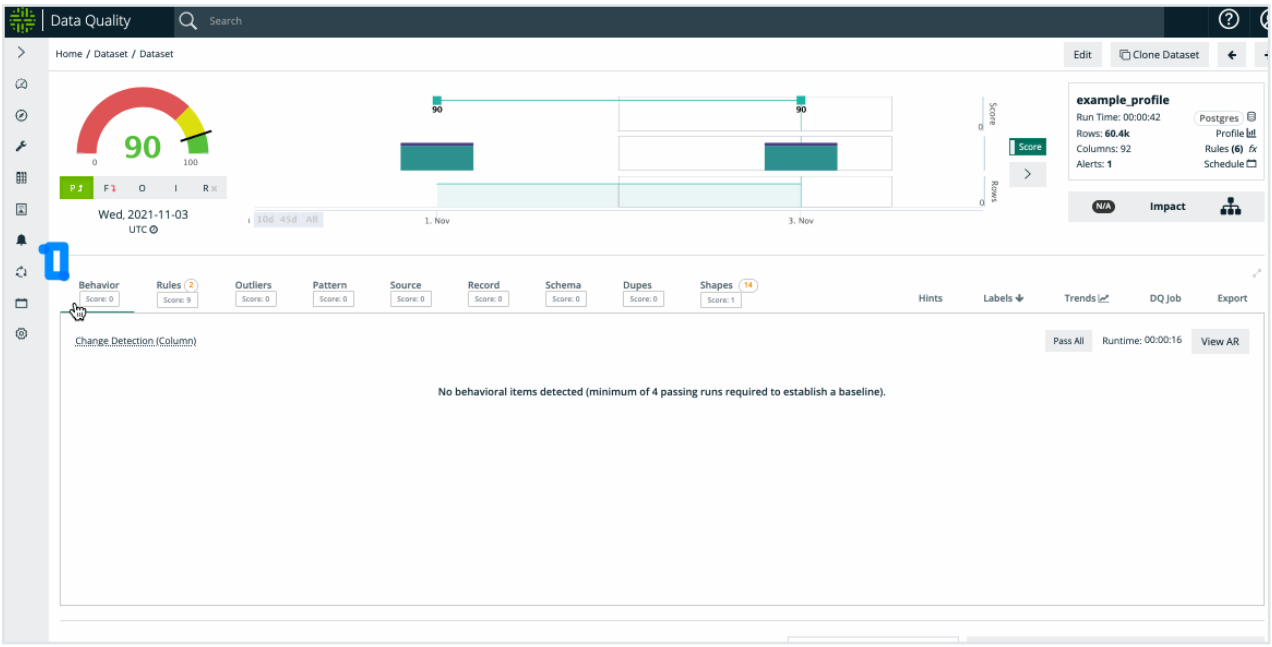
> Note Results are found under the Behavior tab (short for behavioral analytics). This tracks changes in the heuristics of the underlying data profiling metrics. The adaptive rules (AR) modal displays a complete list of monitoring types and criteria.



Evolution of Rule based Data Quality

The main goal of Collibra Data Quality is to provide enterprise data quality insight while greatly reducing the volume of Rules that need to be written manually. When a data set is brought under management, Collibra profiles the data and builds a model for each data set. This allows Collibra to learn what "normal" means within the context of each data set. As the data changes, the definition of "normal" also changes. Instead of requiring you to adjust rule settings, Collibra continues to adjust its model. This approach enables Collibra to provide automated, enterprise-grade data quality coverage that removes the need to write dozens or even hundreds of rules per data set.

> Note Behaviors is turned on by default. Monitoring will calibrate and detect DQ observations, based on the profiling activity.



Using Behavioral Analytics (Change Detection)

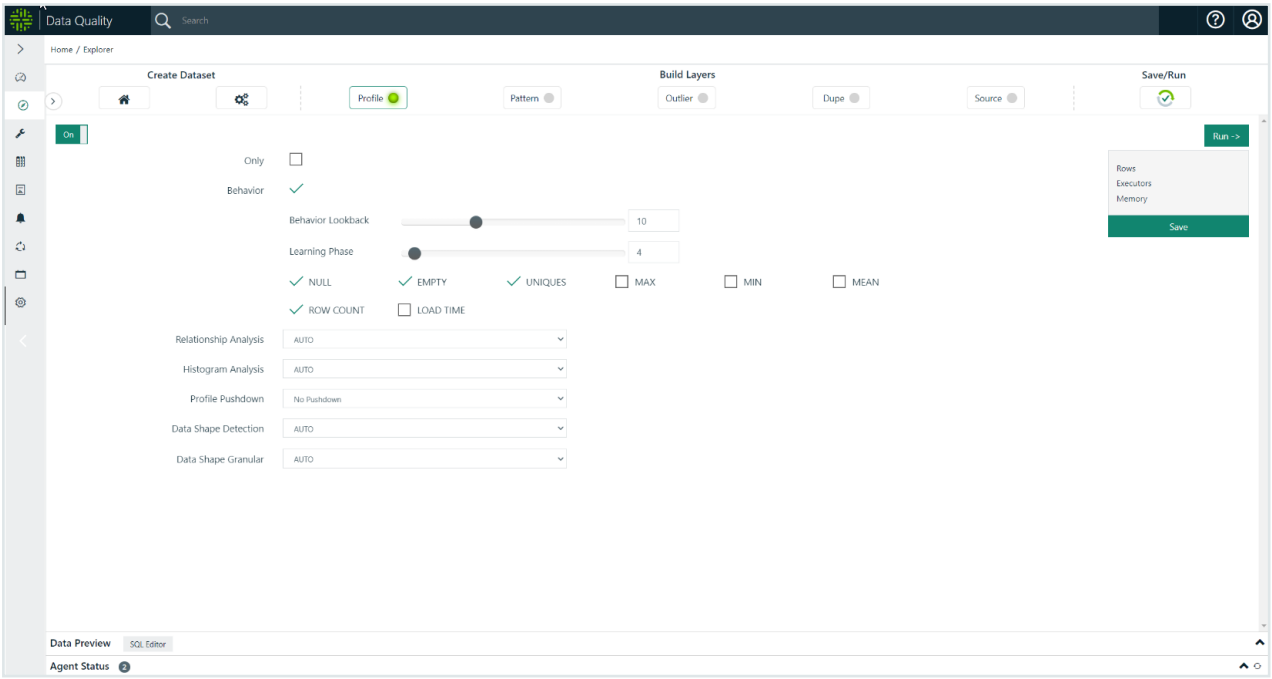
Typically, data quality checks are scheduled to run on a given data set daily. Behavior data quality, or change detection, is built on top of data calculated by the Profile activity. The default settings will often work just fine, however, Collibra lets you specify two key parameters:

Parameter	Description
Behavior Lookback	The number of DQ checks that the model encompasses. For example: A lookback of 10 means that the model is based on the combined statistics from the last 10 DQ checks of a data set.
Learning Phase	The minimum number of DQ checks that are required before behavioral scoring is applied. Collibra DQ does not attempt to apply Behavioral scoring to a data set until at least this many DQ checks are run on it.

Applicable Behavioral Factors

You can choose to forego the application of any of the above factors to the scoring of the model. For example, you can instruct Collibra Data Quality to not track MIN and MAX ranges of values in columns by unchecking the MIN and MAX checkbox. This prevents

Collibra from detecting any extreme values in any column of the dataset using the Behavioral model.

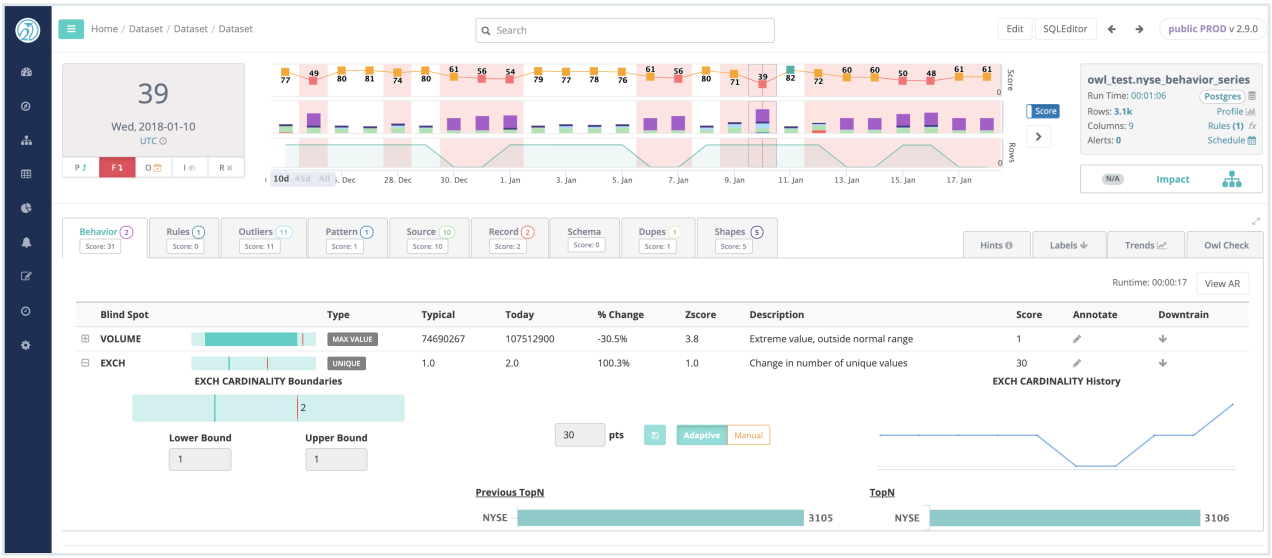


With each run, Collibra Data Quality profiles the data set at the column level and begins to establish a model for the data set. Initially, there is no need for any manual intervention, just keep the data coming. Within a few runs, the model becomes sufficiently robust to begin detecting data quality issues that are otherwise covered by manual rules. For example, Collibra may detect that a particular column experienced a spike in the number of NULL values (typically a manually defined rule).

> Note Collibra Data Quality's behavioral model consists of the following factors:
>
> - NULL values
> - Empty values
> - Cardinality
> - Datatype shifting
> - Row counts
> - Load time
> - Minimum value
> - Maximum value
> - Mean value

Over time, the definition of normal for any given column within the dataset can change. The data may legitimately become more sparse or decrease in volume. Collibra Data Quality continues to learn and adjust the model throughout the life of the data set. However, if there is a drastic (but legitimate) change in the data, this could still mean several days of unnecessary alerts while the model is adjusting. To accelerate model adjustment, Collibra DQ provides the ability to adjust the acceptable range for a given behavioral finding.

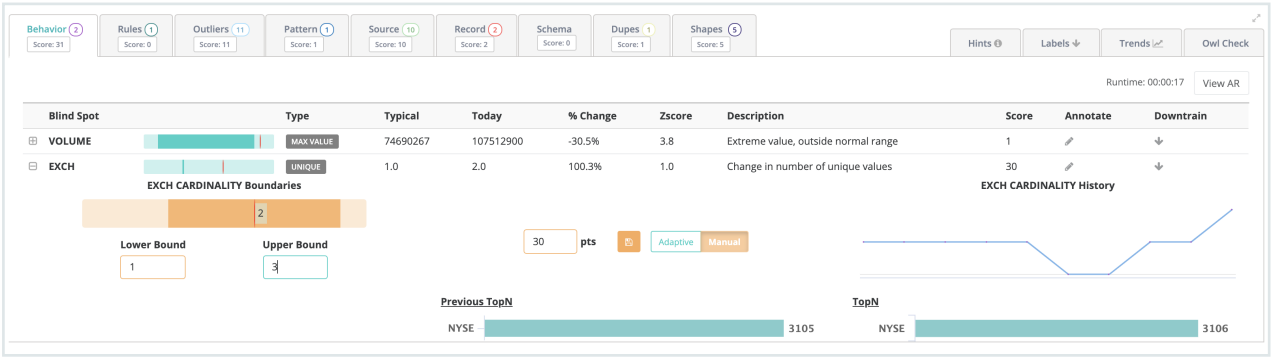
For example, Collibra DQ learned that a particular column typically has between 10% and 20% Empty values. Today, the column is 80% Empty values. Collibra raises a data quality issue and subtracts a proportional amount of points from the quality score of today's DQ run. You may review the finding and realize that there is a legitimate business reason for why that column has more empty values. With a few clicks, you can adjust the acceptable range for that finding. Collibra incorporates the user-defined inputs into the model and adjusts the current day's quality score. Collibra Data Quality would have eventually arrived at the correct range without any input, but without user input, it may have taken a few runs to get there.



Drill-in to see the predicted range of valid values

Automatic flagging of break records with erroneous data.

The screenshot above shows some of the controls and visualizations that can be used to tune the Behavioral model. In this specific example, Collibra has detected that the cardinality of the EXCH field has doubled from 1 to 2 unique values. However, you can instruct Collibra to disregard this finding and adjust the model by manually specifying the range of values acceptable in this column. To assist the user, Collibra provides a line chart and a historical topN visualization of this column's cardinality.



If you want to instruct Collibra that there can be as many as 3 valid values in the EXCH column, click the "Manual" button and adjust the upper bound from 1 to 3, then click the save button.

Collibra adjusts the Behavioral model's baseline, removes the finding, and adjusts the quality score. From that point on, Collibra knows the acceptable range for unique values in the EXCH column is between 1 and 3.

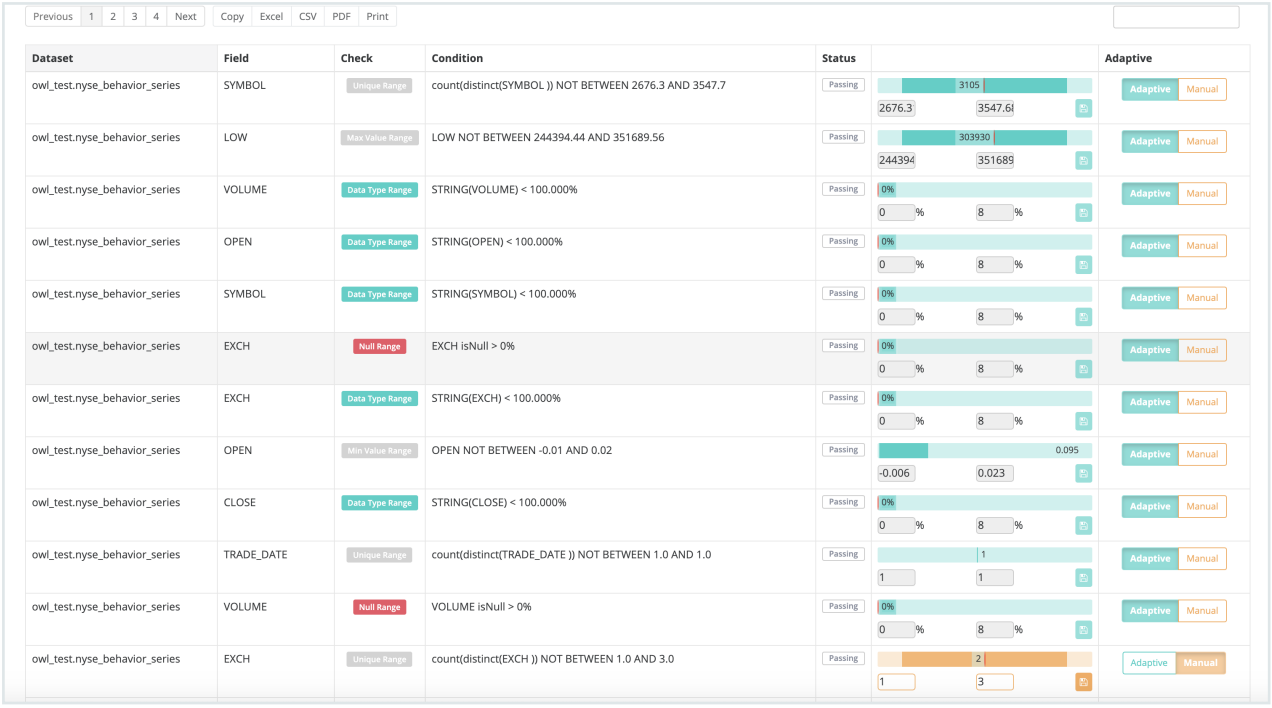
Item	Description
Blind Spot	The name of the column with a possible change detected during a DQ check.
Type	The type of DQ check performed on a given column. For example, Unique (Range) is the number of unique values that fall outside a given range.
Baseline	The baseline value is the mean of the preceding number of scans determined by the value selected for Behavior Lookback in the Profile section of the Explorer page.
Today	
% Change	The percent change from the value of one row to another. $x = \frac{V_2 - V_1}{\lvert V_1 \rvert} * 100$
Δ % Change	The delta percent change from the value of one row to another.
Zscore	The number of standard deviations away from the expected baseline value.
Description	The description of the type of DQ check performed on a given column.
Score	The value subtracted from your overall DQ score. The distance from the expected ranges set by the variance and boundaries of the baseline value. Expected ranges are also visible in the AR panel with graphs available in the Details panel for each line item.

Item	Description
Action	The item labels you can apply to an observation that let you train the behavioral model on future runs. Available options are Validate, Invalidate, and Resolve.
Status	The status of a DQ item, for example, Observation.
Profile	
Details	By clicking the Details button, a line graph lets you drill into changes over time.

Adaptive Rules

As Collibra Data Quality builds and evolves the behavioral model, it exposes all of the "Adaptive Rules" that it learns about. The above example demonstrates how Collibra learns and automatically applies rules. You have control but if left alone, Collibra learns what "Normal" means for a given data set and scores the data set accordingly. This results in a large set of rules that are automatically applied and adapted as the data set changes over time.

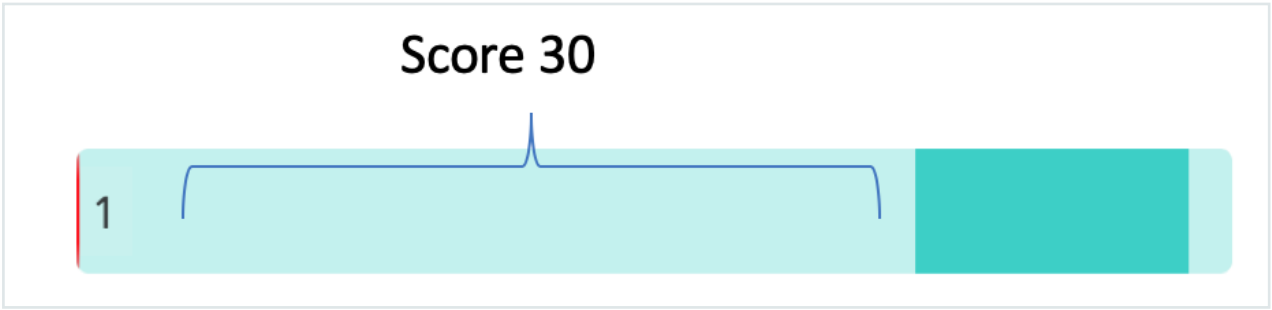
To view or modify Adaptive Rules, navigate to the Behavior tab on the findings page for the desired data set and click the "View AR" button on the right side of the screen. This brings up a full list of Adaptive Rules.



The Adaptive Rules also provide you with the ability to adjust ranges derived from the behavior model. You can manually adjust the tolerance range and score of any Adaptive Rule. While this may at times be convenient, it is also just fine to let Collibra Data Quality handle the model tuning through its own learning process.

Scoring

In adaptive mode Collibra Data Quality automatically generates a DQ item score based on the egregiousness of the line item. This measurement is directly proportional to the distance from the green range to the red line. Example below.



The score can range from 0-30. This ties to the percent change and Z-Score. In cases when the Z-Score ranges from 0.0 - 6.0.

FAQ

Q: Which Collibra DQ API contains all behavioral checks (passing and breaking)?

- /v2/getdqchecksdetails.

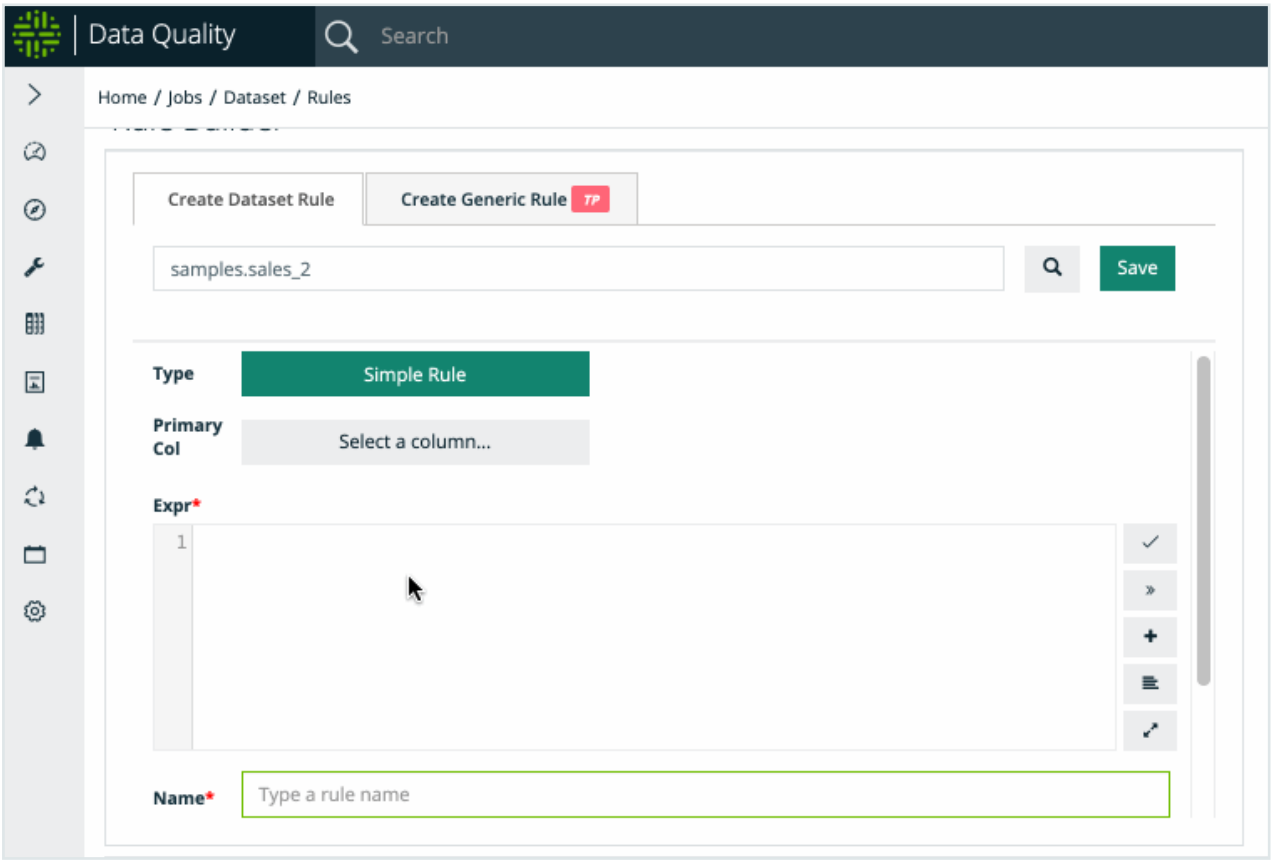
Q: How is 'Mean' defined in the Behavioral Modal chart?

- Mean represents the average of behavioral lookback window e.g. if today is the 11th of the month, and the bhlb is set at 10, the mean will be the average of the 1st to the 10th, and the statistic on the 11th day will represent the change to that mean.
- Also of note: the mean only includes passing rules, not failed runs.

Rules (user-defined)

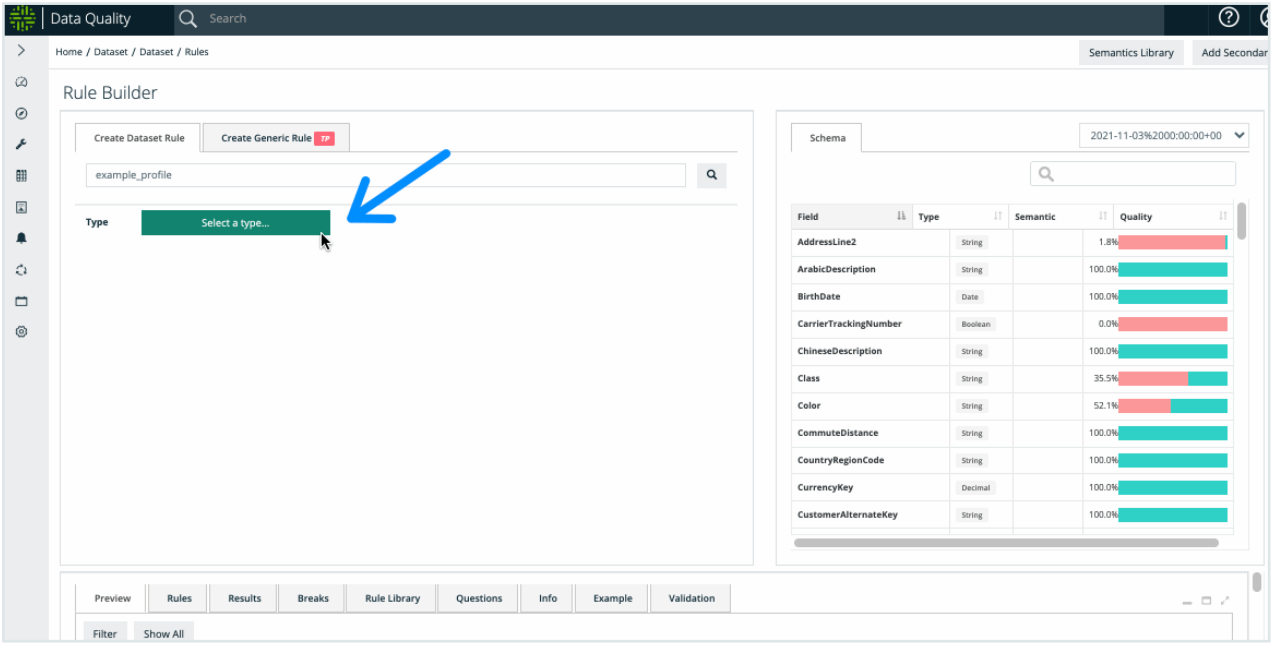
Apply custom monitoring with SQL.

SQL Rule Engine



Introduction

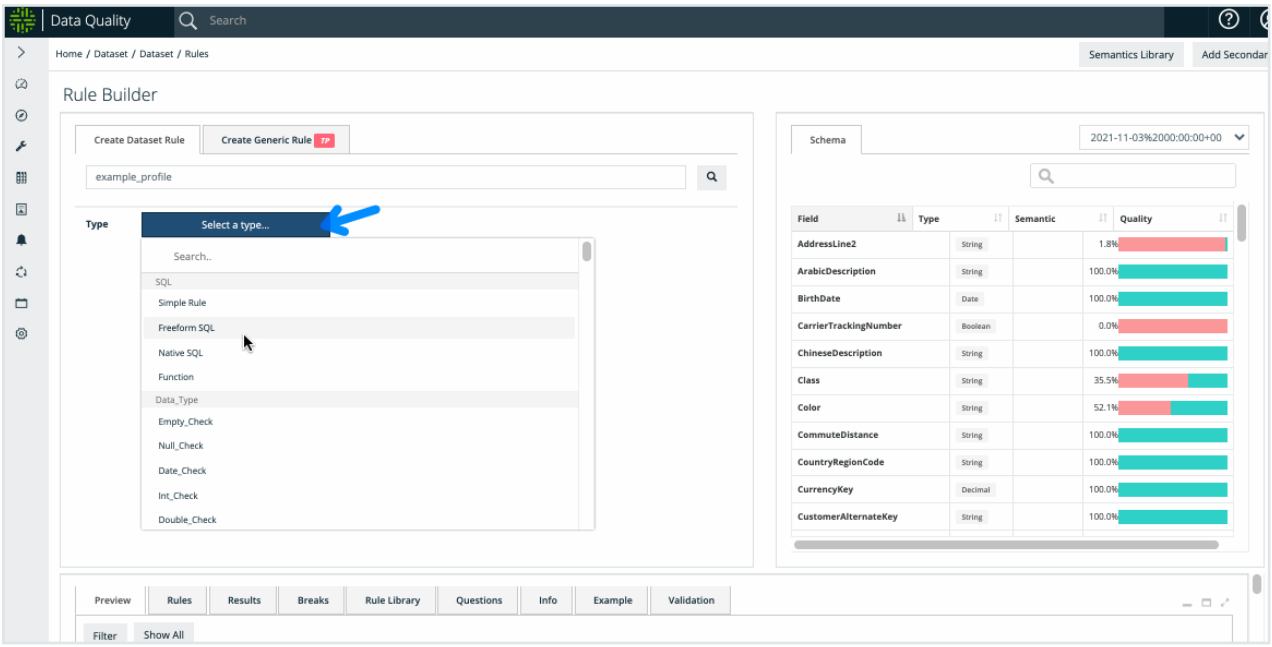
Collibra Data Quality takes a strong stance that data should first be profiled, auto-discovered and learned before applying basic rules. This methodology commonly removes thousands of rules that will never need to be written and evolve naturally overtime. However there are still many cases to add a simple rule, complex rule or domain specific rule. Simply search for any dataset and add a rule. You can use the optional Column Name/Category/Description to add meta-data to your rules for future reporting.



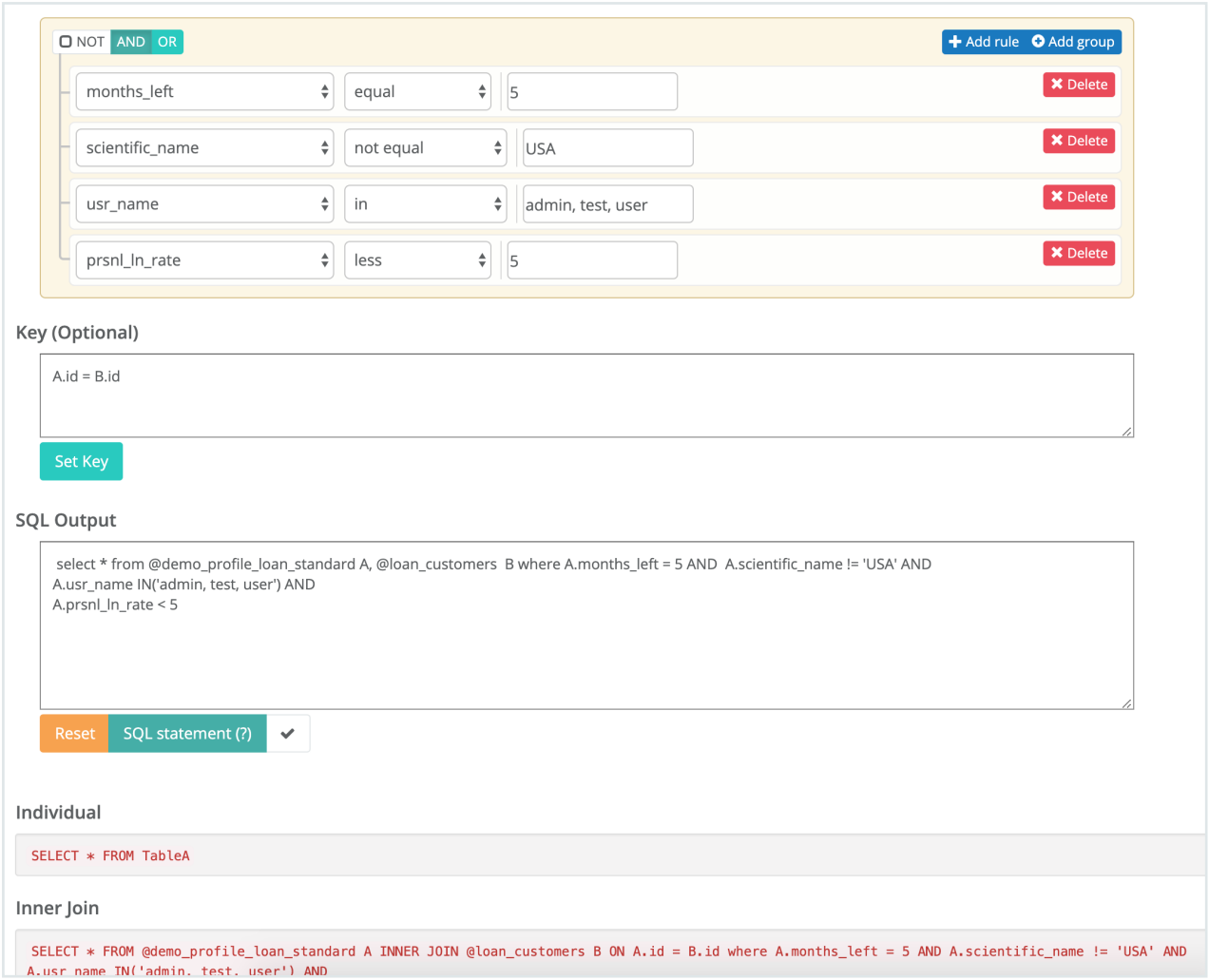
> Note Customized discovery routines can be run using the Rule Library together with Rule Discovery.

Query Builder

Query builder will help generate SQL for more complex rules. You can apply to one or two tables (Table A on left and Table B on right). The query builder can help build up multi-part conditions.


Data Quality
Search
Home / Dataset / Dataset / Rules
Semantics Library
Add Secondar
Rule Builder
Create Dataset Rule
Create Generic Rule
example_profile
Type
Select a type...
Search..
SQL
Simple Rule
Freeform SQL
Native SQL
Function
Data_Type
Empty_Check
Null_Check
Date_Check
Int_Check
Double_Check
Schema
2021-11-03%2000:00:00+00
Field
Type
Semantic
Quality
AddressLine2
ArabicDescription
BirthDate
CarrierTrackingNumber
ChineseDescription
Class
Color
CommuteDistance
CountryRegionCode
CurrencyKey
CustomerAlternateKey
Preview
Rules
Results
Breaks
Rule Library
Questions
Info
Example
Validation
Filter
Show All

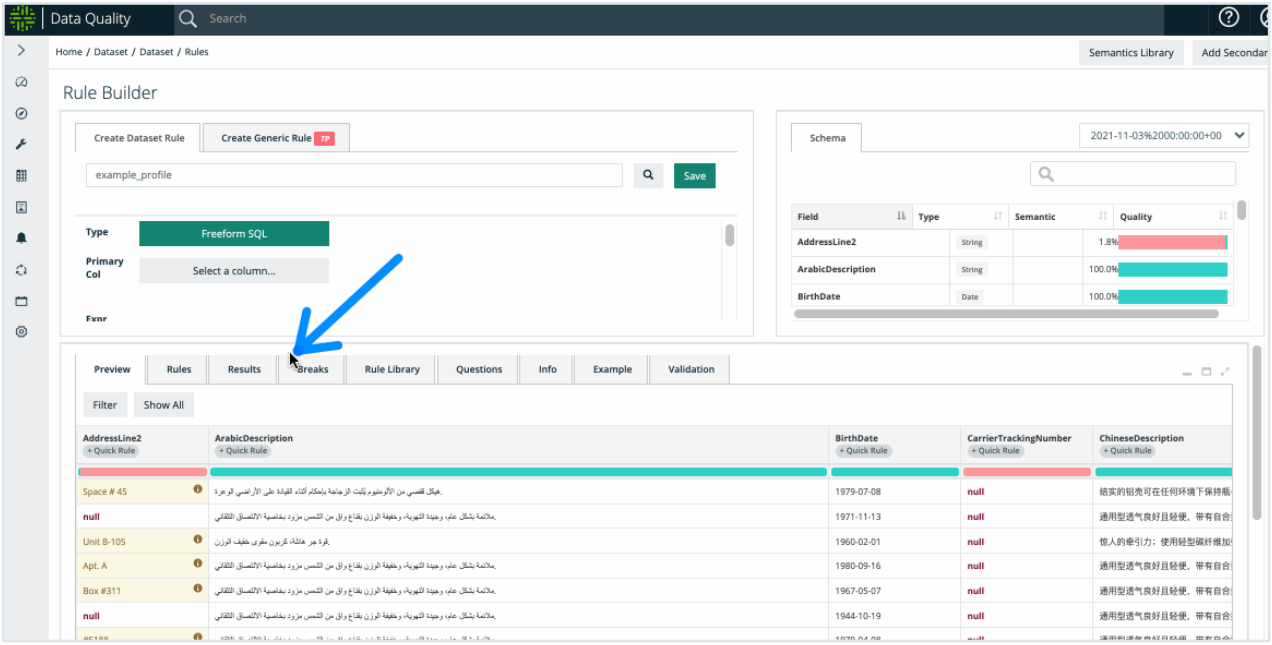
Builder
NOT
AND
OR
Add rule
Add group
id
equal
B.id
Delete
Key (Optional)
Use the builder to set a key, if joining more than one table
Set Key
loan_customers
id
Field
app_id
business_num_id
emplmt_industry
guid_num
id



> Note As with any SQL generator, there are limitations for more complex scenarios.

Break Records

Storing break records is only available for Freefrom and Simple rule types. Rule library rules uses one of these types as well.



Enable additional storage with the -linkid flag. This allows you to store complete sets of break records. See the DQ Job LinkID for more details.

> Note Stat, Native, and Data Type (global) rules are not eligible for storing exception records.

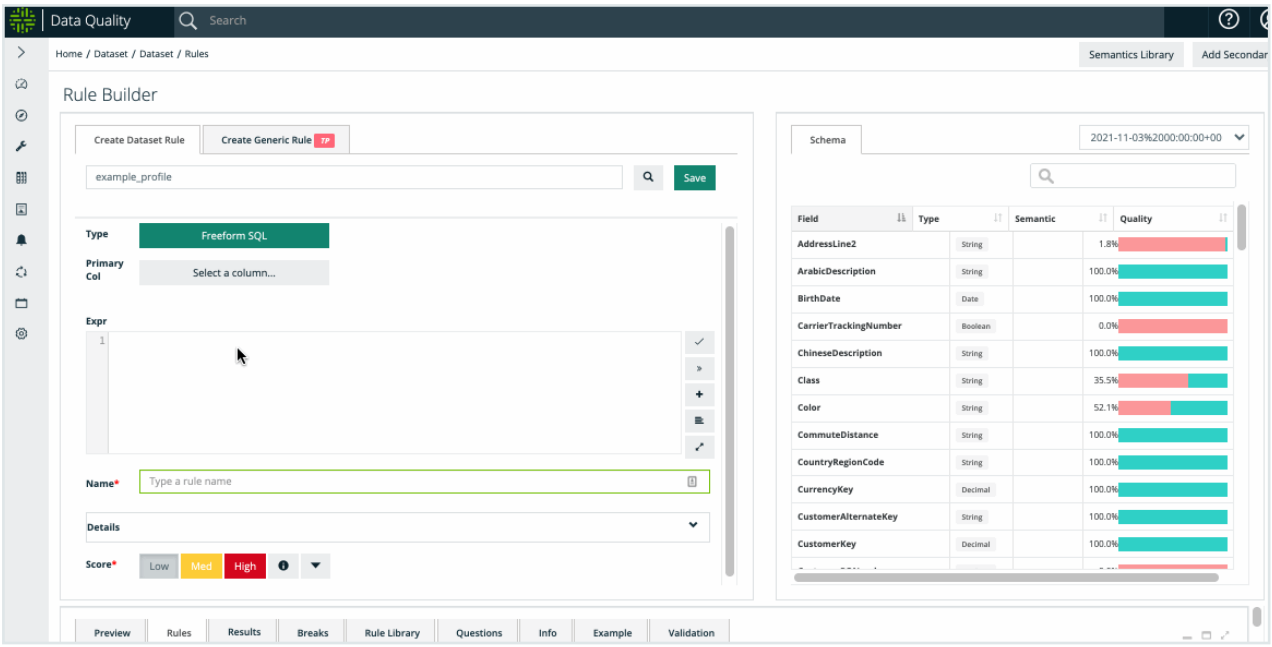
Quick Tips

If joining more than one data source, make sure both sets of drivers are in the -lib. Or separately supply a -libsrc pointing to the appropriate directory/jar file location. Versions later than 2021.11 use the -addlib for additional directories to add to the classpath.

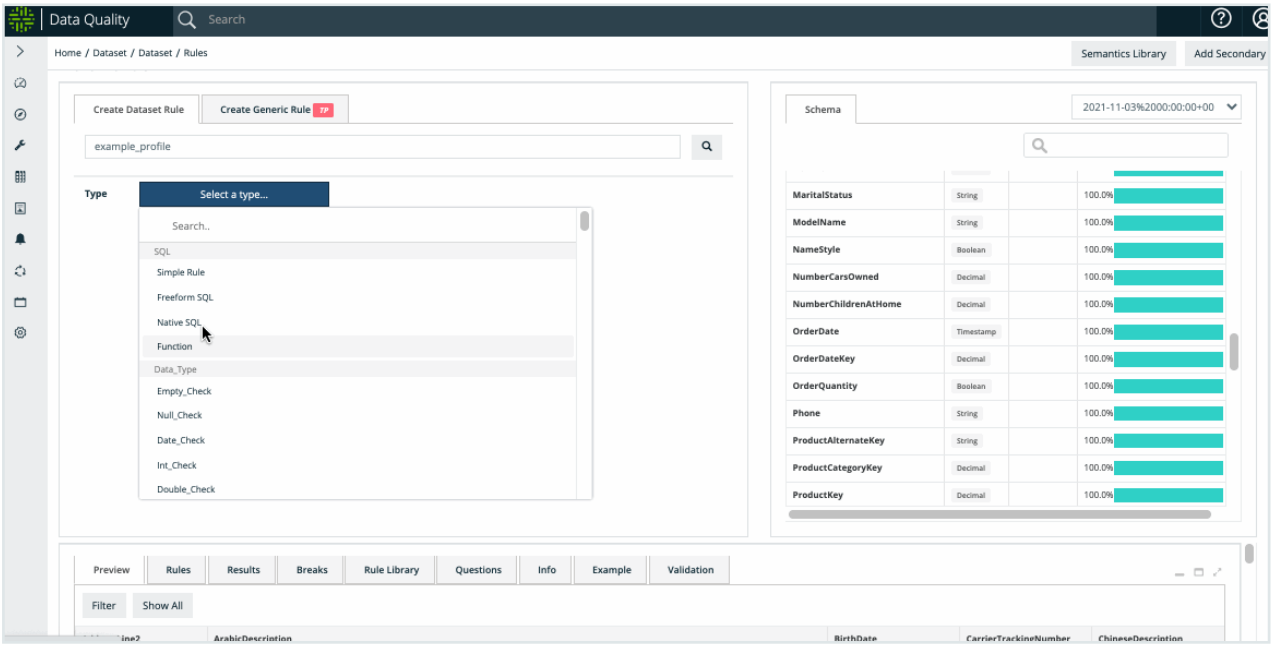
Native SQL uses your native DB syntax. The score is total break records / rows from the scope (query / -q) of the defined DQ job.

Spark SQL

This is a complete list of Spark SQL operators and functions available.
https://spark.apache.org/docs/latest/api/sql/index.html



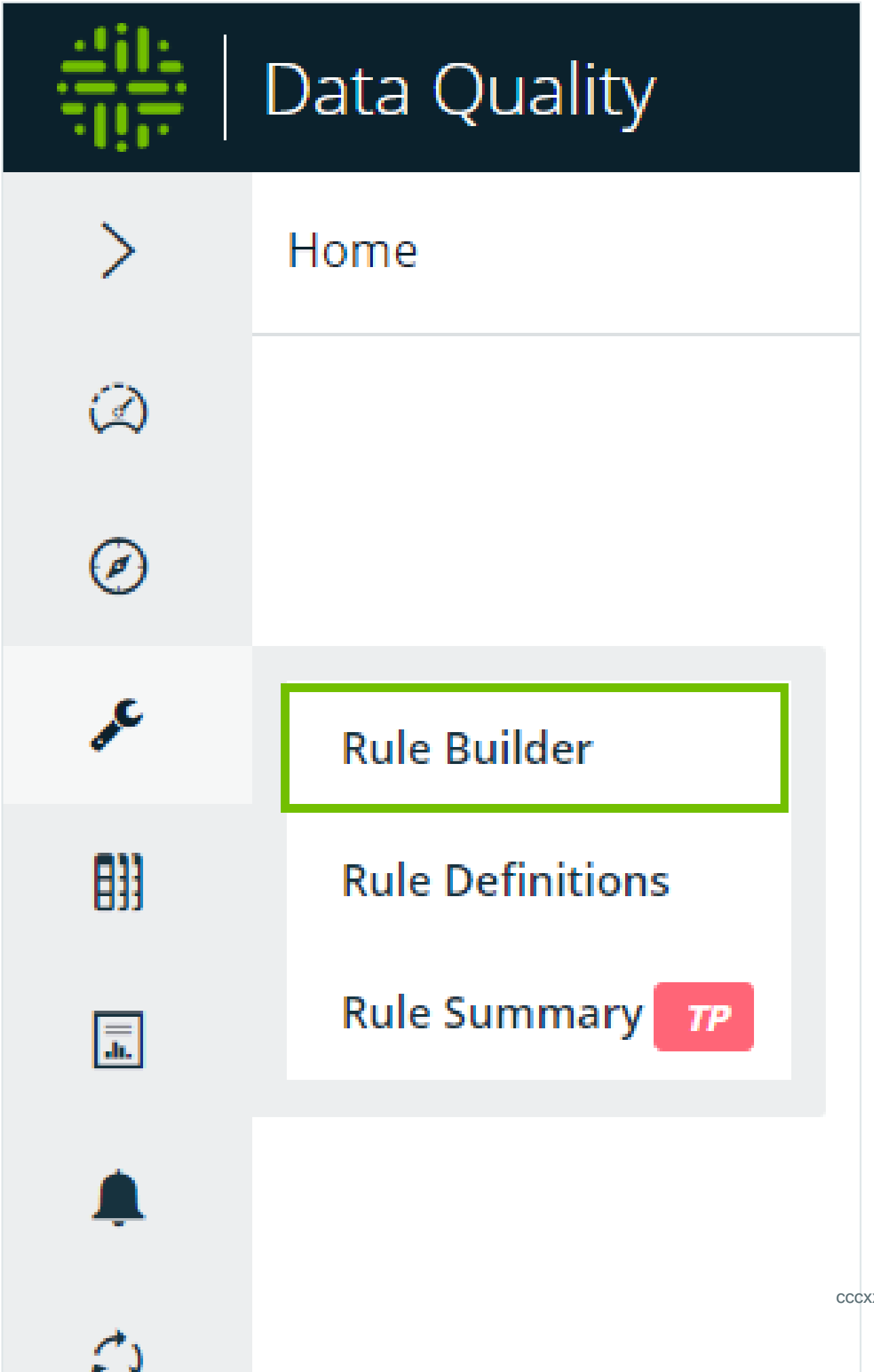
Adding a Rule



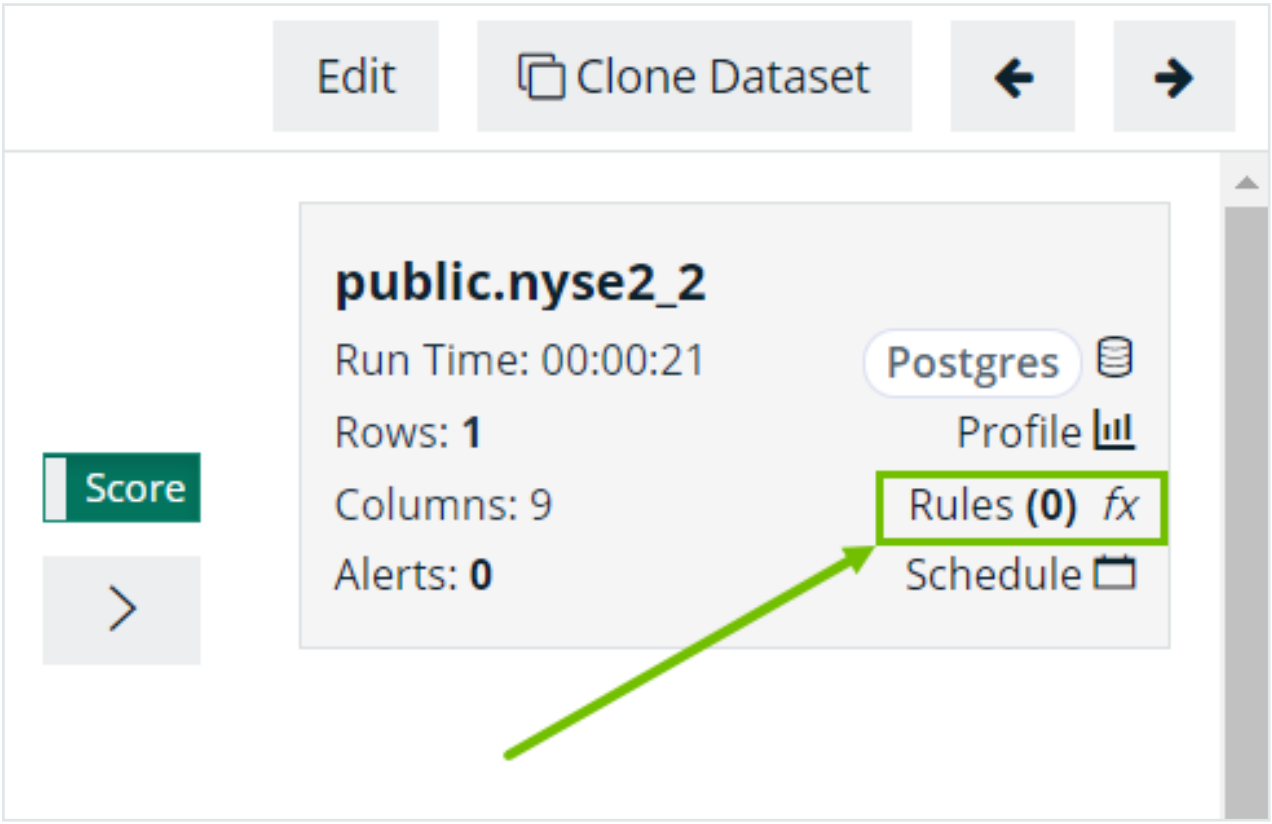
To add a rule, go to the Rules page. There are two ways to access the Rules page in Collibra DQ:

- From the **left navigation bar**.
- From the **findings page**.

To access the Rules page from the left navigation bar, click the wrench icon and then **Rule Builder**. From the Rule Builder page, select a data set and a rule type.


Data Quality
Home
Rule Builder
Rule Definitions
Rule Summary
TP

To access the Rules page from the findings page, open a DQ Job to display the findings page. From the findings page, click **Rules** in the metadata box in the upper right of the page. The Rule Builder opens. Since you're navigating to the Rule Builder from the findings page directly, you do not have to select a data set. In this case, select a rule type to get started.

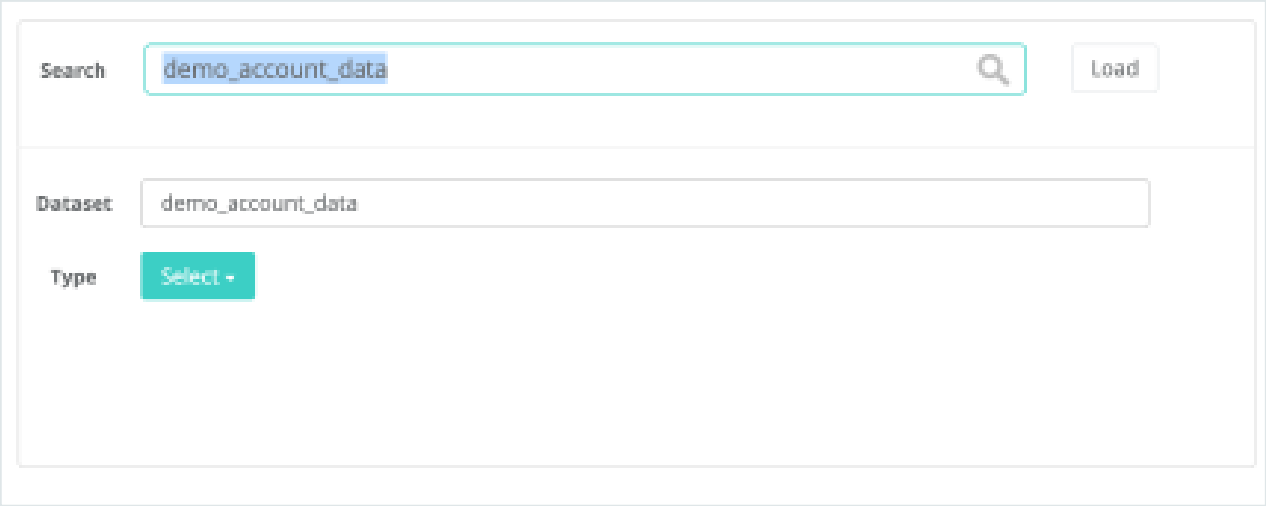


Instructions

1. Search for a data set or navigate to the Rule Builder page in the left navigation panel.
    - Rules can only be applied to data sets once a DQ job runs once
2. Click Load.
    - The schema and any previously saved rules populate.
3. Select a rule type with the dropdown next to the Type label
4. Select a rule name
    - If applying a preset rule, the rule name will be auto populated
5. Input a rule condition
    - Only if applying a simple, freeform sql, stat, or native rule type.
    - Provide a value in the condition/sql/function input field.

- Keystroke Ctrl+Space provides IntelliSense.

6. Select Low, Medium or High for scoring severity (optional).
7. Add any custom DQ dimensions for reporting (optional).
8. Click submit to save the rule.



The rule is measured on the next DQ job run for that particular data set.

Rule Types

Rule type	Description	Example
Simple rules	Simple rules are used when you want to filter a condition on a single column in a single table.	City = 'Baltimore'
Freeform SQL rules	Freeform SQL rules are used when you want to apply a condition across multiple tables/columns and generally when more flexibility or customization is desired.	select * from dataset where name = 'Collibra'
Preset rules	Preset rules are used for quickly adding strict condition checks. Commonly used conditions are available to add to any data set columns.	

All built-in Spark functions are available to use. Visit https://spark.apache.org/docs/2.3.0/api/sql/ for simple and freeform sql rules.

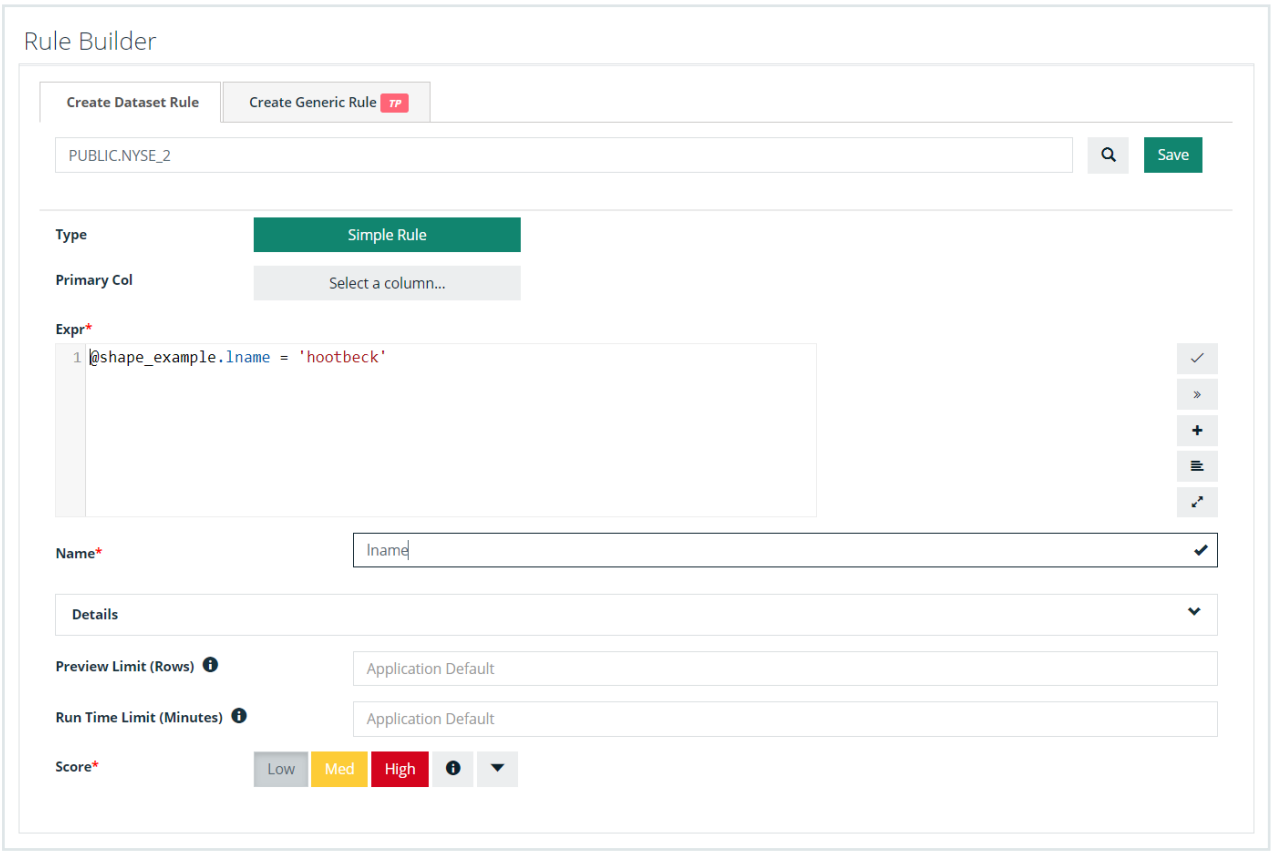
Points and Percentage

For every percentage the *x* condition occurs, deduct *y* points from the data quality score. If a rule was triggered 10 times out of 100 rows, break records occurred 10% of the time. If you input 1 point for every 1 percent, 10 points would be deducted from the overall score.

Creating Your First Rule

Let's create a simple rule using the below information. The data set name.

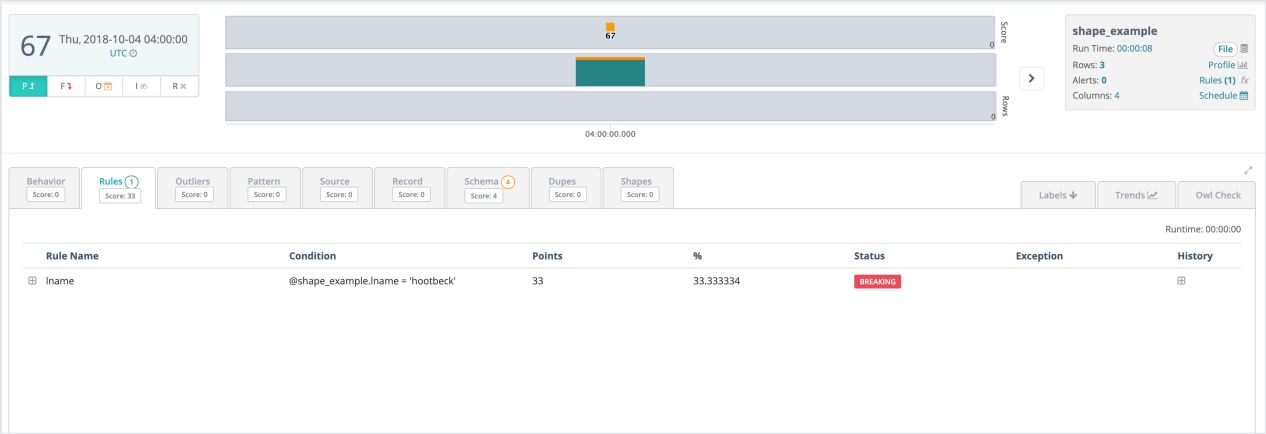
1. Search for "shape_example" and click "Load"
2. Select "Simple Rule"
3. Rule Name = lnametest
4. @shape_example.lname = "hootbeck" (should hit one time day over day).
5. Points = 1
6. Percentage = 1
7. Click "Submit"

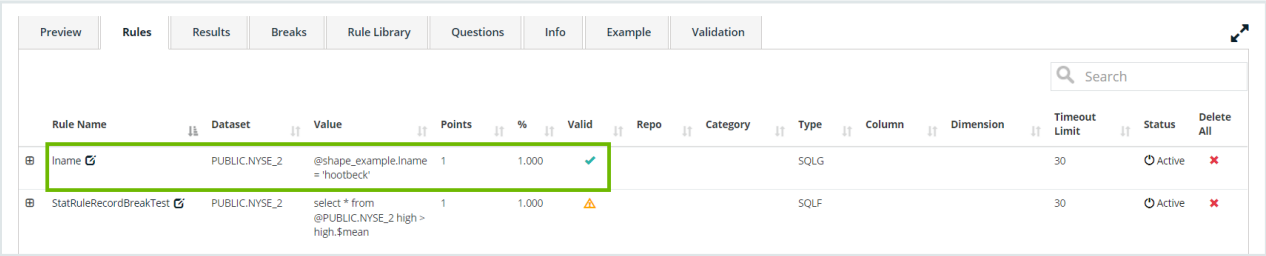


Once the rule has been submitted please find the below list of rules with the new rule we just defined as shown below.

Using the Rules tab

Rule scores will appear under the Rules tab on the findings page. You can also see more details in the bottom panel of the Rules page under the Rules and Results tabs.

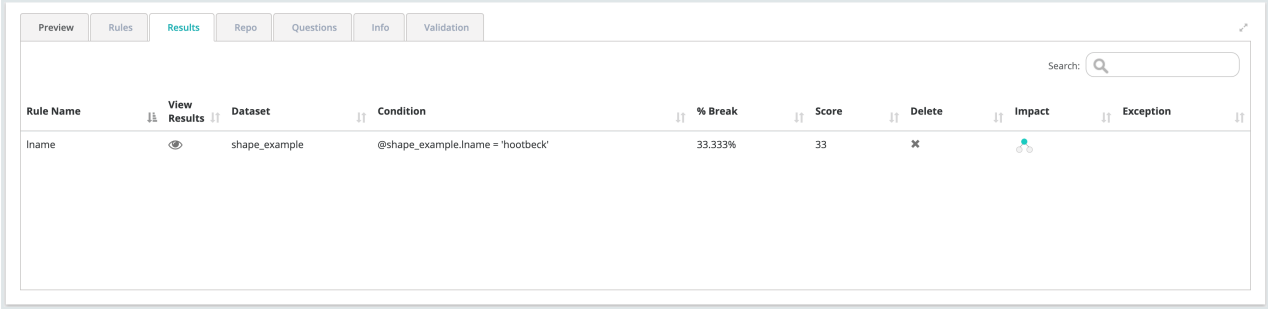




Click the plus icon next to the Rule Name to drill into any available rule. The following table describes the columns when you drill into a rule:

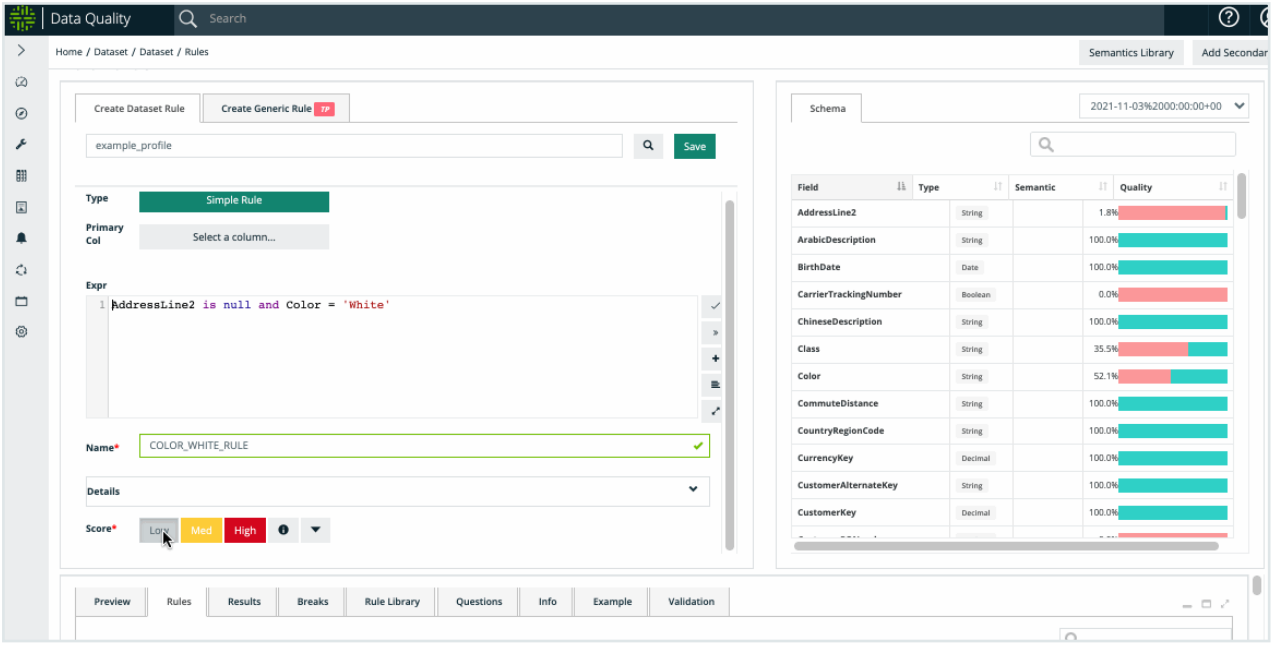
Column	Description
dataset	The dataset to which the rule applies.
ruleNm	The unique name of the rule.
ruleValue	The column of your table, view, or schema the rule queries against.

<table>
<tr><th>Column</th><th>Description</th></tr>
<tr><td>modType</td><td>There are four possible values that signify the audit sequence of a rule:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>i</td><td>Insert</td></tr>
<tr><td>u</td><td>Update</td></tr>
<tr><td>d</td><td>Delete</td></tr>
<tr><td>s</td><td>Status change</td></tr>
</table></td></tr>
<tr><td>updtTs</td><td>The timestamp of the last run of a rule.</td></tr>
<tr><td>isActive</td><td>The binary value of whether a rule is active or not. Active rules have values of 1 and inactive rules have values of 0.</td></tr>
<tr><td>userNm</td><td>The username of the user who generated the rule.</td></tr>
</table>



Scores

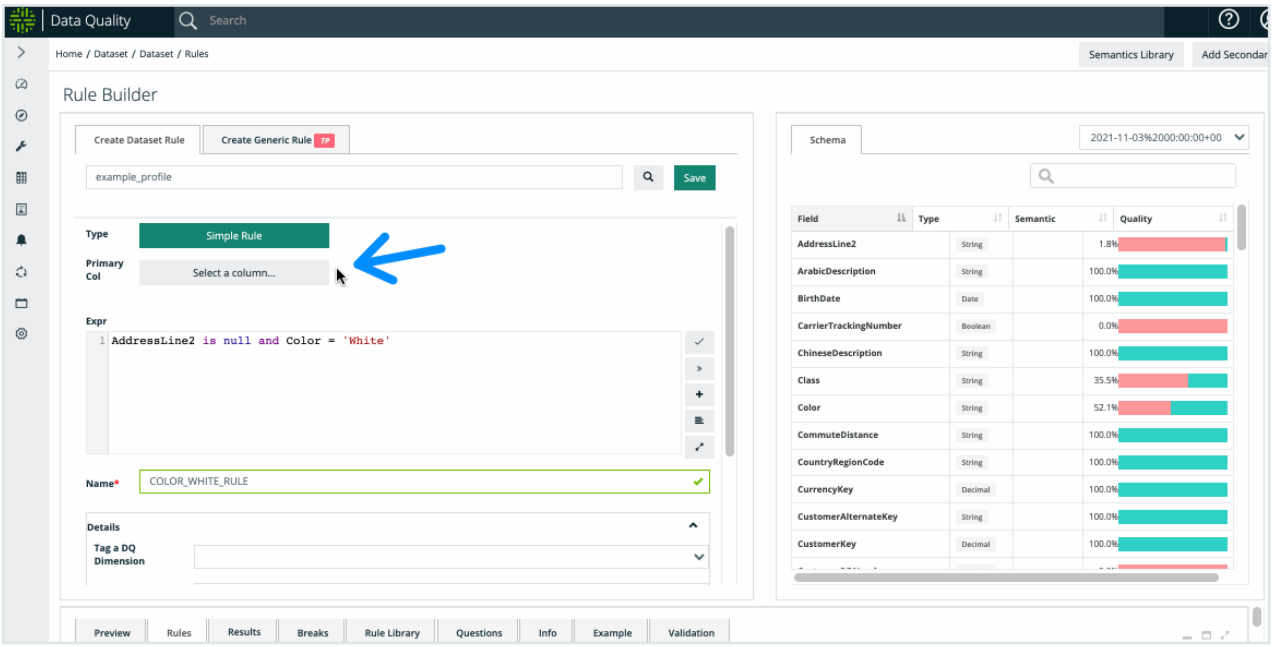
Each rule definition has a specific score. Depending on the severity, rules can be hard or soft, and can contribute a different weight that affects your overall DQ score.



> Note This is calculated by points per threshold. The ratio is calculated from the total rows of the associated DQ job scan.

Dimensions

Each rule can be associated with a dimension. You can also apply metadata like primary column and description. While not mandatory, these can be particularly useful when using the DQ Connector and Custom.



Rule Types

SQL-Based Rules

Depending on the complexity, you can choose from short-form or long-form rules.

Simple

Just the condition (short-form). For example, using the column email_address. This runs against the dataframe and uses Spark SQL syntax. Simple rules can be thought of as everything after the where clause.

```
email_address is not null and email_address != ''
```

Freeform

Where 'Simple' rules just use the condition, 'Freeform' rules use the complete SQL statement. When more complex SQL is required, you can express more with Freeform including joins, CTE's and window statements.

```
select * from @DATASET_NAME where
email_address is not null and email_address != ''
```

> Note All built-in spark functions are available to use. (https://spark.apache.org/docs/2.3.0/api/sql/) for simple and freeform sql rules.

Native

Native rules use the SQL dialect of the underlying connection and database. Files are not eligible for native SQL rules. This is ideal if you want to use pushdown profiling and you want to use existing SQL logic. When coupled with pushdown profiling, you can achieve a very minimal infrastructure footprint.

See the Native section for more details.

Stat rules

Write rules against meta data and profiling stats. Complex counts and ratios can be referenced with simple syntax.

See the Stat Rules section for more details.

Data Type

- **Empty check**
  - Rule type: EMPTYCHECK
  - Description: Checking whether the target column has empty values or not
- **Null check**
  - Rule type: NULLCHECK
  - Description: Checking whether the target column has *NULL* values or not
- **Date check**
  - Rule type: DATECHECK
  - Description: Checking whether the target column has only DATE values or not

- **Integer check**
  - Rule type: INTCHECK
  - Description: Checking whether the target column has only INTEGER values or not
- **Double check**
  - Rule type: DOUBLECHECK
  - Description: Checking whether the target column has only DOUBLE values or not
- **String check**
  - Rule type: STRINGCHECK
  - Description: Checking whether the target column has only STRING values or not.
- **Mixed datatype check**
  - Rule type: DATATYPECHECK
  - **Description:** ---

SQL Based Rules

Collibra Data Quality uses ANSI-compliant SQL and offers pushdown rules that leverage specific database syntax.

Simple

This is the condition after the "where" clause.

Depending on the complexity, users can choose from short form or long form rules.

Simple

Just the condition (short form). For example, using the column email_address. This runs against the dataframe and uses Spark SQL syntax. Simple rules can be thought of as everything after the where clause.

```
email_address is not null and email_address != ''
```

> Note All built-in spark functions are available to use. (https://spark.apache.org/docs/2.3.0/api/sql/) for simple and freeform sql rules.

Freeform

Fully defined SQL for more detailed checks.

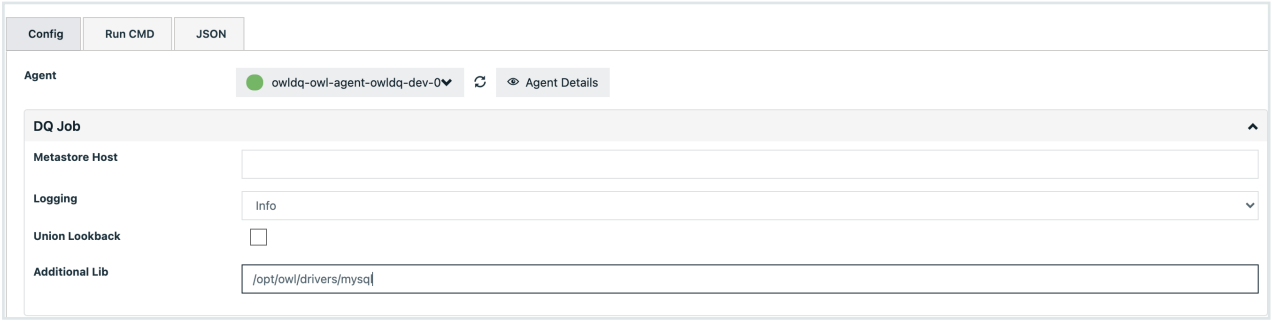
Where 'Simple' rules just use the condition, 'Freeform' rules use the complete SQL statement. When more complex SQL is required, you can express more with Freeform including joins, CTE's and window statements.

```
select * from @DATASET_NAME where
email_address is not null and email_address != ''
```

> Warning Freeform rules should include a semi-colon at the end of the rule value.

> Note All built-in Spark functions for non-Pushdown connections are available for use. See (https://spark.apache.org/docs/2.3.0/api/sql/) for Simple and Freeform SQL rules.

Cross-Connection Libraries



> Note When applying cross-connection rules please use the -addlib to submit the job with the appropriate jar files. In this example, a secondary set of jars is added through the Explorer. These files are located in the /opt/owl/drivers/mysql directory. The path should not contain double quotes or single quotes. It should point to a directory without spaces in the path.

Known Limitations

Freeform rules with fully qualified column names are currently unsupported when they use the following syntax:

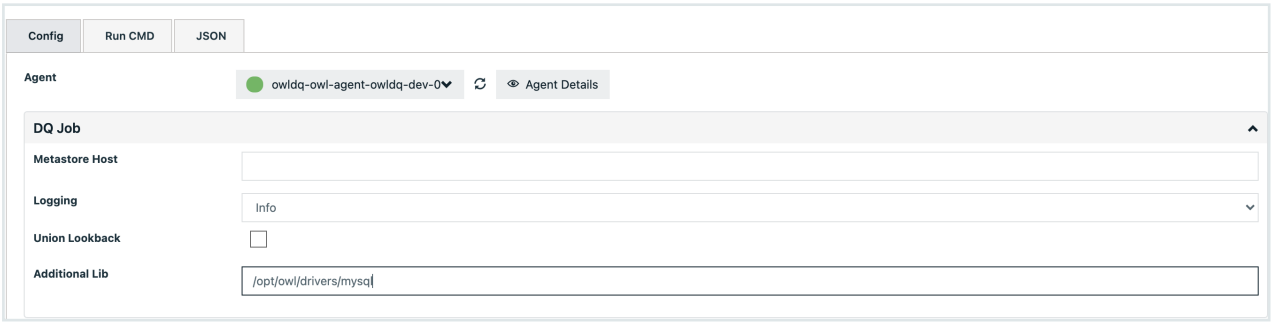
```
select <column name> FROM @<dataset name> WHERE @<dataset
name>.<column name> condition
```

A possible workaround to this limitation is to use aliasing instead.

Cross-Data Set Rules

> Warning Best practice for cross-table joins is to define a view and scan the view. Only in circumstances when a view cannot be created should you define cross-table joins with 2 separate data sets (DQ Jobs) and express the join in the rule.
>
> If you're doing multiple lookups, this will improve long-term performance and consolidate maintenance.

> Note Cross-dataset rules require -libsrc (prior to 2021.11) or -addlib (post 2022.01)



In-Clause (Single Column)

```
select * from @table1 where id
not in ( select id from @table2 )
```

Except (Multi-Column)

```
select id, app_id, email, guid_num from @table1
EXCEPT
select id, app_id, email, guid_num from @table2
```

Referencing secondary data sets

When you create a Freeform SQL rule with secondary data sets, there are three important points to consider in order to avoid missing data set exceptions.

- Ensure that the name of your first data set matches the name of the main data set on which your rule is created.
- The column names in your table must be exact matches with the syntax of your query. >> If the column name uses CAPS, then your query must also use CAPS.
- Ensure that any data set referenced in your rule maintains a score of greater than or equal to (>=) a passing score of 75.

> Note These three points apply to all rules, not just Freeform Rules.

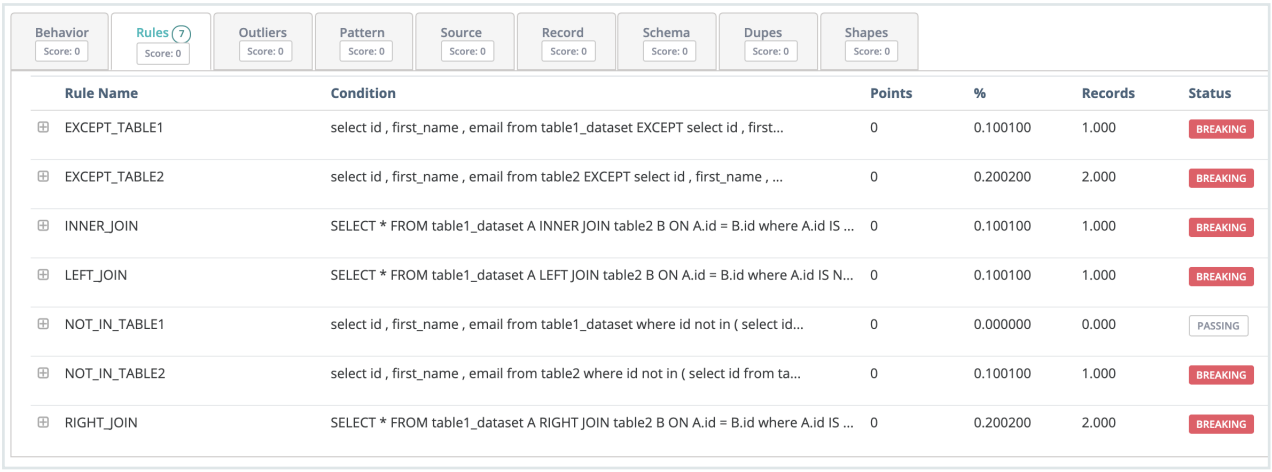
Joins

A **join** lets you reference two or more data sets in one rule. Joins are useful when you want to compare the values of a data set from a previous, or when you want to verify that all values are valid across your data sets.

Join example

```
SELECT
    *
FROM
    @table1 A
    LEFT JOIN @table2 B ON A.id = B.id
where
    B.id is null OR (A.email != B.email)
```

Sample Results



Cross-Table (Guided). Use our wizard to do ad-hoc analysis and visual setup.

Join Example Example (vs. cross-table guided seen above).

Multi-part condition rules with the rule builder. Combines profiling metrics & builder in one screen.

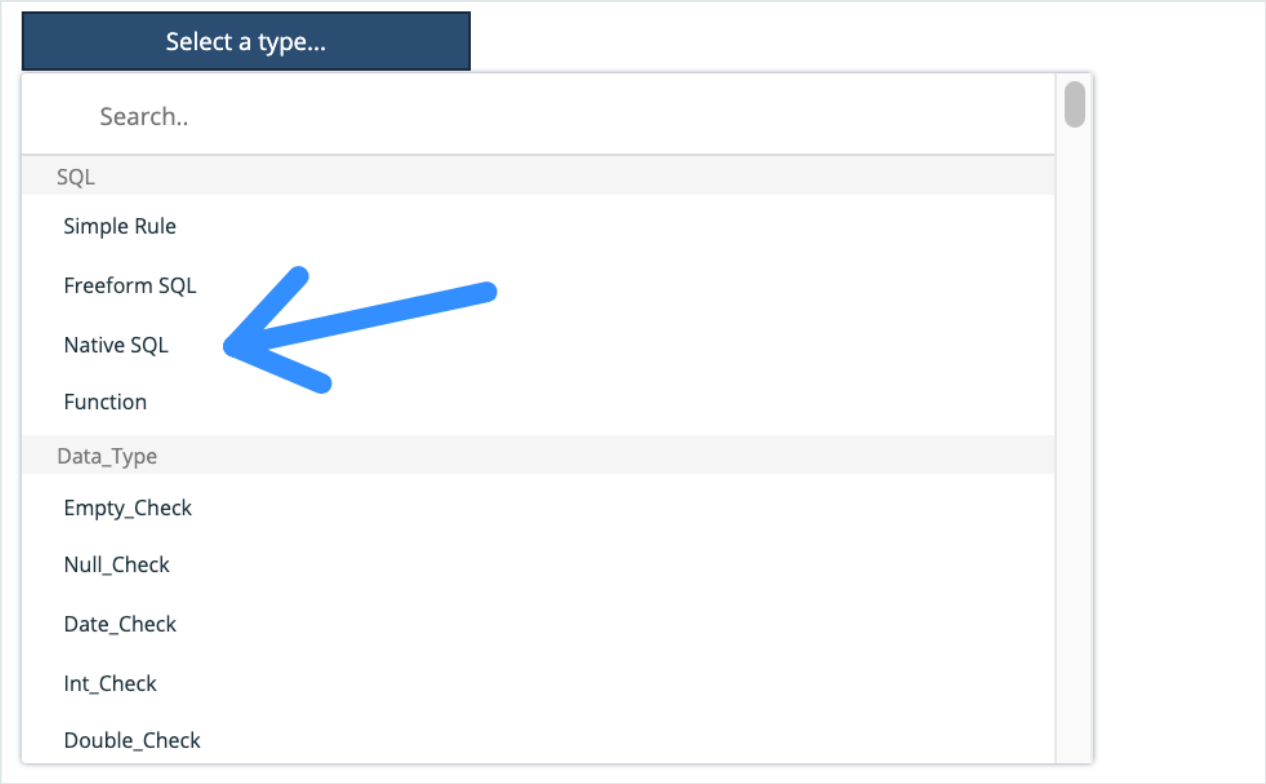
Known Limitations

There is a known limitation where a Cartesian join which produces a greater number of rows than the initial dataset shows a score greater than 100%. There is not a workaround other than the query for the DQ Job be the query that produces that larger number of rows.

Native

With Native SQL expressions we provide capability to use your existing validation statements, even if they use database-specific (Postgres/DB2/Oracle/MSSQL/etc.) functions or expressions.

This is commonly used when using pushdown profiling and/or pre-existing SQL validations exist in a specific syntax. Native rules are often referred to as pushdown rules.



If you have rules already written in Oracle, Sybase, or DB2 syntax - Copy/Paste the rule directly into the Native SQL section.

When creating a Native SQL rule, keep the following in mind:

- The SQL query must be a valid expression that can be run as a subquery. To avoid pulling large amounts of data into memory, Collibra Data Quality will wrap your expression so it only fetches the number of rows returned. Rules should be written such that the query returns the anomalous rows.
- The SQL query must take less than 30 minutes to run. It is recommended to use partitioned columns for efficiency.

- When testing the SQL query from the app, it is helpful if it takes less than 30 seconds to run. You can add a limit to reduce query time, or test the query in your SQL Editor.

> Note Testing native rules can be done quickly by limiting the results or using an external SQL IDE as well.

> Warning This rule type is not eligible to store break records, just the score itself! Additionally the rule will apply the logic directly to the JDBC connection. Native rules will run on the entire population of the data, regardless of any scope set for the DQ job (-q flag defined in the scope section of Explorer). Any filter or predicate clause entered in the -q flag should be added to the logic of the Native rule as well.

Stat Rules

A powerful technique is to access profile statistics in your rules. These are typically sub-second operations that do not require scanning or iterating. There are several cases where SQL struggles to support rules, such as: isNull but not "null count" or nullRatio or nullPercent. Or having access to types without doing crazy cast() operations. These are simplified below, for example `fname.$type != 'String'`.

```
select * from @dataset where
fname.$type != 'String' AND $rowCount < 800
```

Dataset-level stat rules

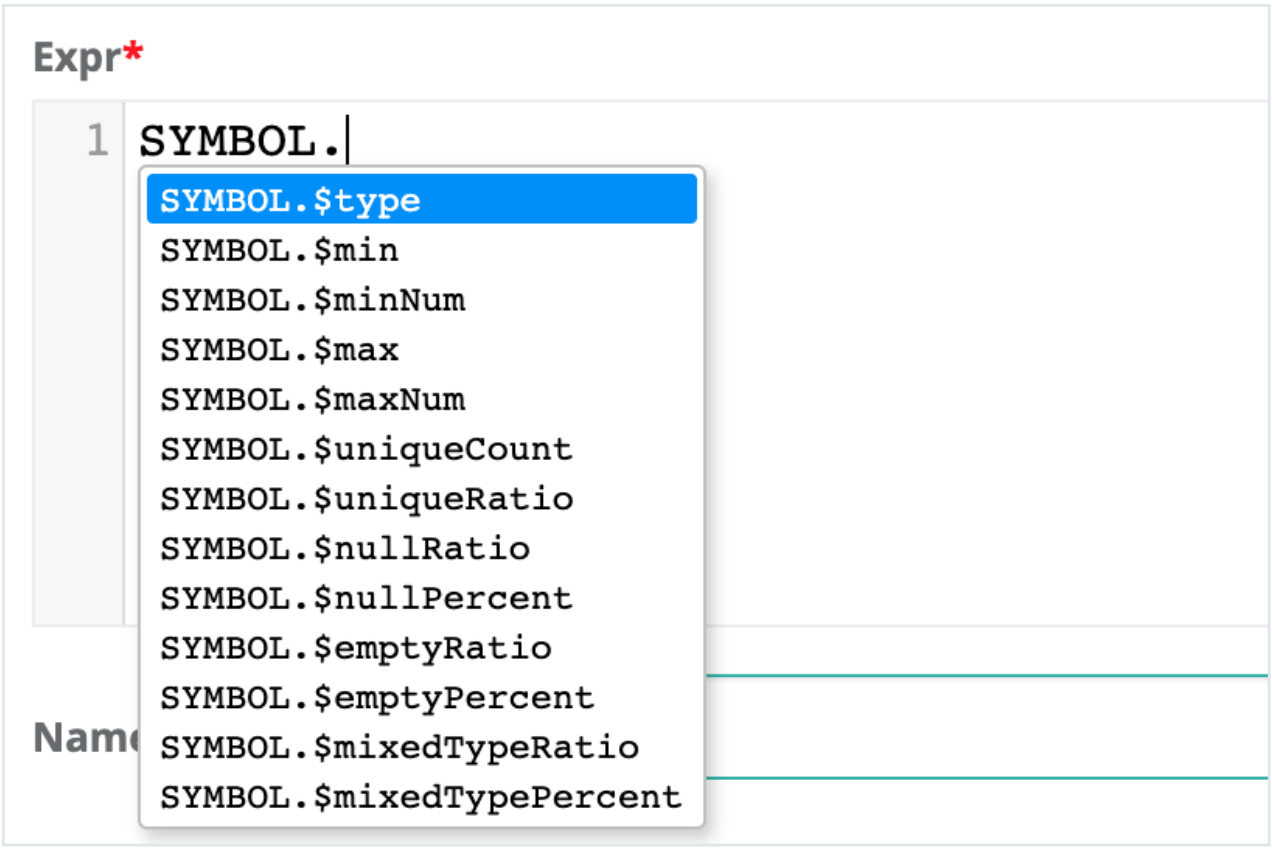
Dataset-level Stat Rule	Rule Example	Description
$totalTimeInSeconds	$totalTimeInSeconds > 25	Alert when DQ job runs longer than 25 seconds.
$totalTimeInMinutes	$totalTimeInMinutes > 5	Alert when DQ job runs longer than 5 minutes.
$totalTimeInHours	$totalTimeInHours > 1	Alert when DQ job runs longer than 1 hour.

Dataset-level Stat Rule	Rule Example	Description
$rowCount	$rowCount < 9000	Alert when row count less than 9,000.
$runDate	$runDate='2020-01-24'	Use the ${rd} variable in rules.
$daysWithoutData	$daysWithoutData > 4	Alert when a dataset has missing or no rows for 5 days.
$runsWithoutData	$runsWithoutData > 4	Alert when a dataset runs but has missing or no rows for 5 days.
$daysSinceLastRun	$daysSinceLastRun > 4	Alert when a dataset has not run for 5 days.

Column-level stat rules

Column-level Stat Rule	Rule Example	Description
.$type	fname.$type != 'String'	Alert when fname is not a string.
.$min	fname.$min > 'apple'	Lexicographical sort works for strings and numbers.
.$minNum	age.$minNum > 13	Type casted to a numeric for simple number checks.
.$mean	row_id.$mean > '4.500'	Alert when the mean is greater than a given value.
.$max	fname.$max > 'apple'	Alert when the max is greater than a given value.
.$maxNum	age.$maxNum > 13	Alert when the numeric value falls outside an acceptable range.
.$uniqueCount	id.$uniqueCount != $rowCount	Alert when the uniqueCount of a field doesn't match the rowCount.
.$uniqueRatio	gender.$uniqueRatio between .4 and .6	Alert when the ratio of uniqueCounts of a given field doesn't match the rowCount.

Column-level Stat Rule	Rule Example	Description
.$nullRatio	lname.$nullRatio not between .4 and .6	Alert when the ratio of nulls no longer falls within acceptable range.
.$nullPercent	lname.$nullPercent not between 40 and 60	Alert when the percent of nulls no longer falls within acceptable range
.$nullCount	lname.$nullCount >= 1	Test for a single null.
.$emptyRatio	nc.$emptyRatio > 0.2	Alert when the ratio of empties no longer falls within acceptable range.
.$emptyPercent	nc.$emptyPercent > 20	Alert when the percent of empties no longer falls within an acceptable range.
.$emptyCount		Alert when the emptyCounts of a field no longer fall within an acceptable range.
.$mixedTypeRatio	nc.$mixedTypeRatio > 0.2	Alert when the ratio of mixed data types no longer falls within an acceptable range. For example, Strings and Ints in the same field.
.$mixedTypePercent	nc.$mixedTypeRatio > 20	Alert when the percent of mixed data types no longer falls within an acceptable range. For example, Strings and Ints in the same field.
.$mixedTypeCount	id.$mixedTypeCount >= 1	Alerts when the mixed data typeCount no longer falls within an acceptable range. For example, Strings and Ints in the same field.

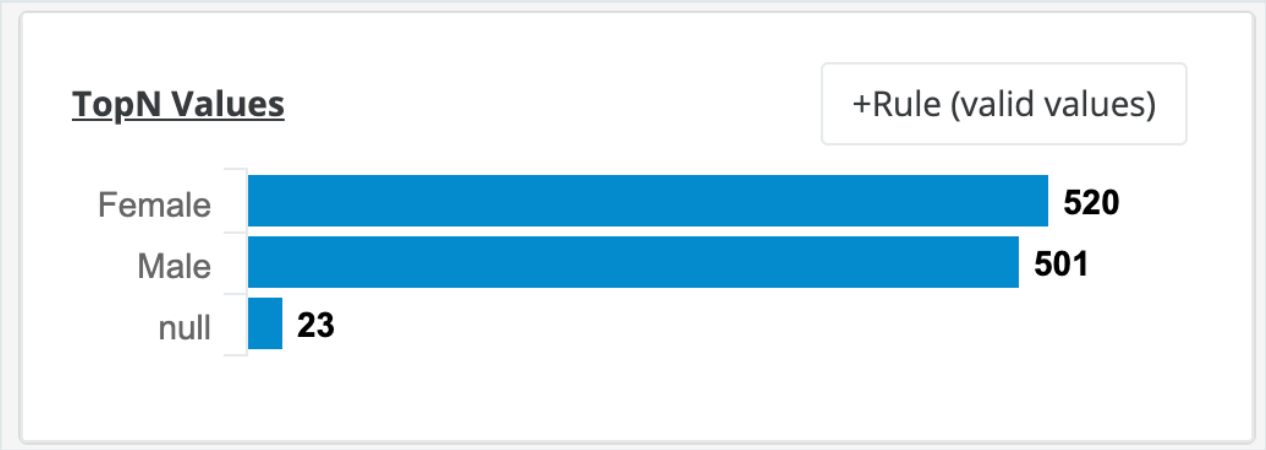


Distribution Rule

There is a common case in DQ where you want to know the distribution of a column's value. Consider gender. It can be expected that a column named gender consists of roughly 40-60% males and roughly 40-60% females if the data set is large and represents the population. This can be difficult to express in plain SQL, but it is very easy with the below syntax.

```
gender['Male'].$uniquePercent between 40 and 60
```

Column Value-level	Rule
.$uniqueCount	credit_rating['FAIR'].$uniqueCount > 7
.$uniquePercent	credit_rating['GOOD'].uniquePercent between 40 and 60



Known Limitations

- The $daysWithoutData stat rule has a limitation where rules that are not named "daysWithoutData" do not display their Pulse View charts when you drill into the rule on the Findings page. A possible workaround for this is to name your stat rule exactly as it appears here: daysWithoutData.
- You cannot combine stat rules or distribution rules with regex rules in the same rule. For example, `car_vin rlike '$[asdf][0-9]' and car_vin.$uniqueCount`

Data Type Rules

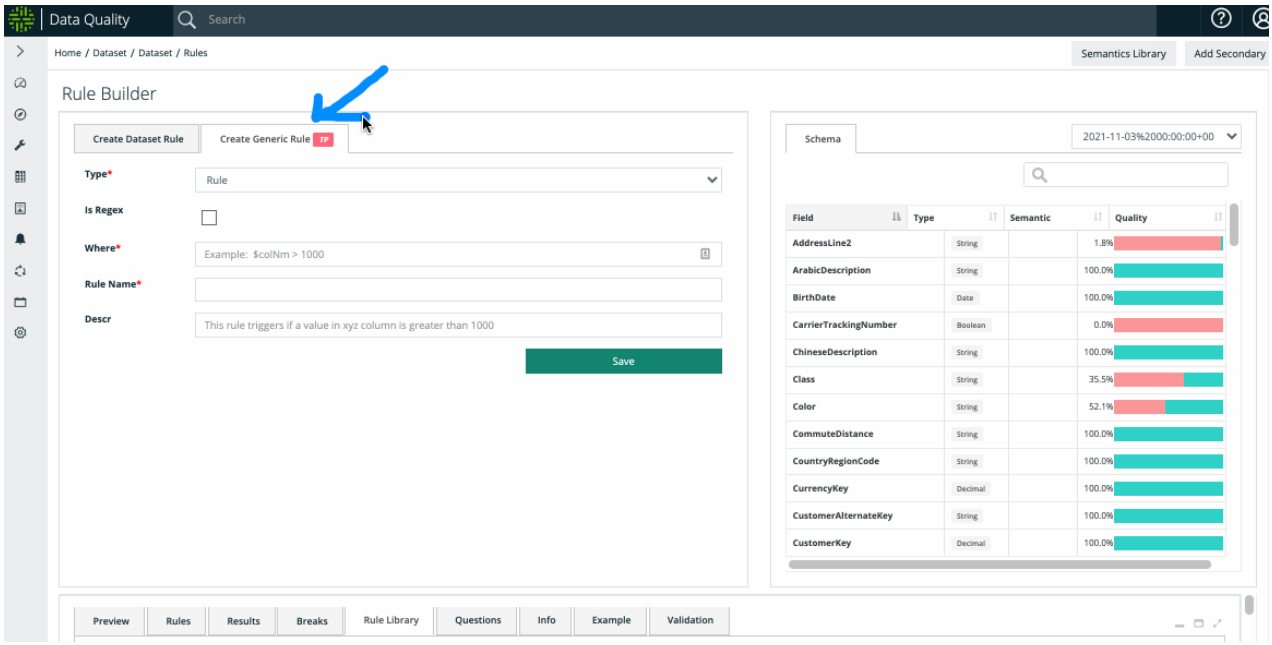
Data Type

Rule	Rule type	Description
Empty check	EMPTYCHECK	Checks whether the target column has empty values or not.
Null check	NULLCHECK	Checks whether the target column has *NULL* values or not.
Date check	DATECHECK	Checks whether the target column has **only** DATE values or not.
Integer check	INTCHECK	Checks whether the target column has **only** integer values or not.

Rule	Rule type	Description
Double check	DOUBLECHECK	Checks whether the target column has **only** DOUBLE values or not.
String check	STRINGCHECK	Checks whether the target column has **only** STRING values or not.
Mixed datatype check	DATATYPECHECK	Checks the dataType of the field. Note Mixed datatype rules are not supported in Pushdown.

Rule Templates

Ideal for rules that apply to more than one data set. Write once, apply many.



Templates are great for regex, format, and compliance checks.

A template rule substitutes the data set and column at runtime. This can save hundreds of redundant rules that do the same thing but on different column names.

Template rules are located in the **Type** dropdown as well as the **Quick Rules** dropdown. The complete list of template rules is located in the Rule Library section. These meant for

global rules that are ideal for code sets, compliance checks, and regex checks. These are ideal for checks that apply to many tables.

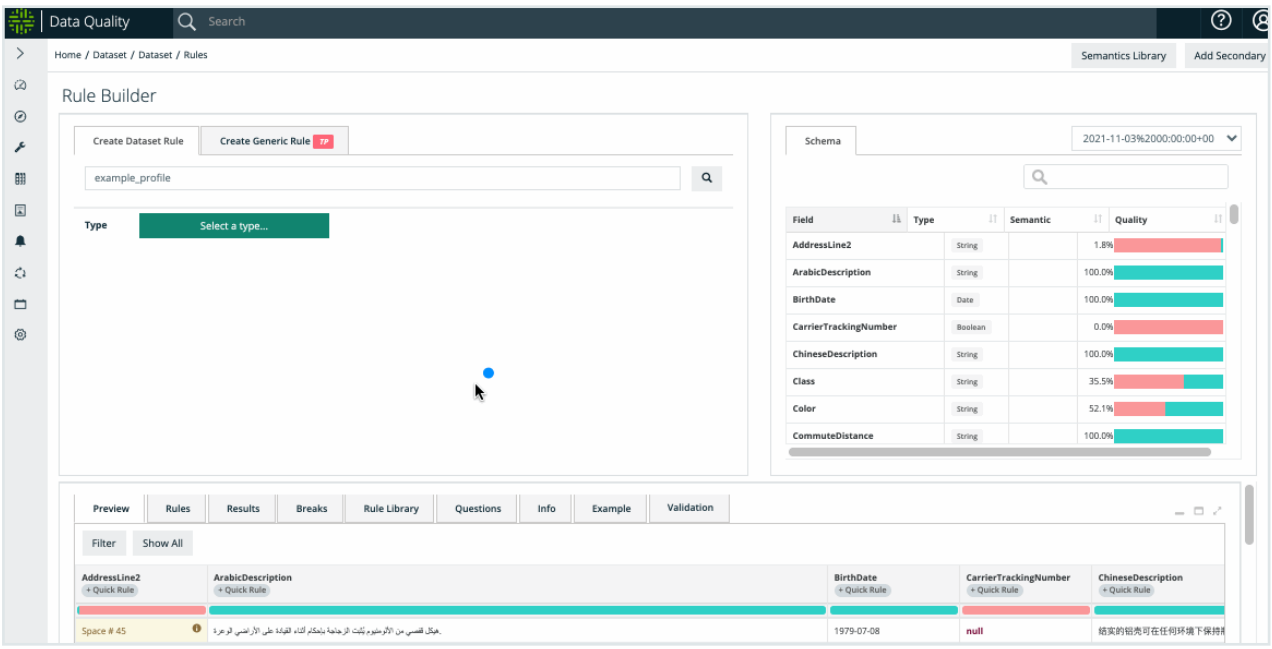
Rule templates use SQLG (simple) and occasionally SQLF (Freeform) types under the hood. Rule templates appear in the Rule Library once they are created.

See the Rule Library section for more details.

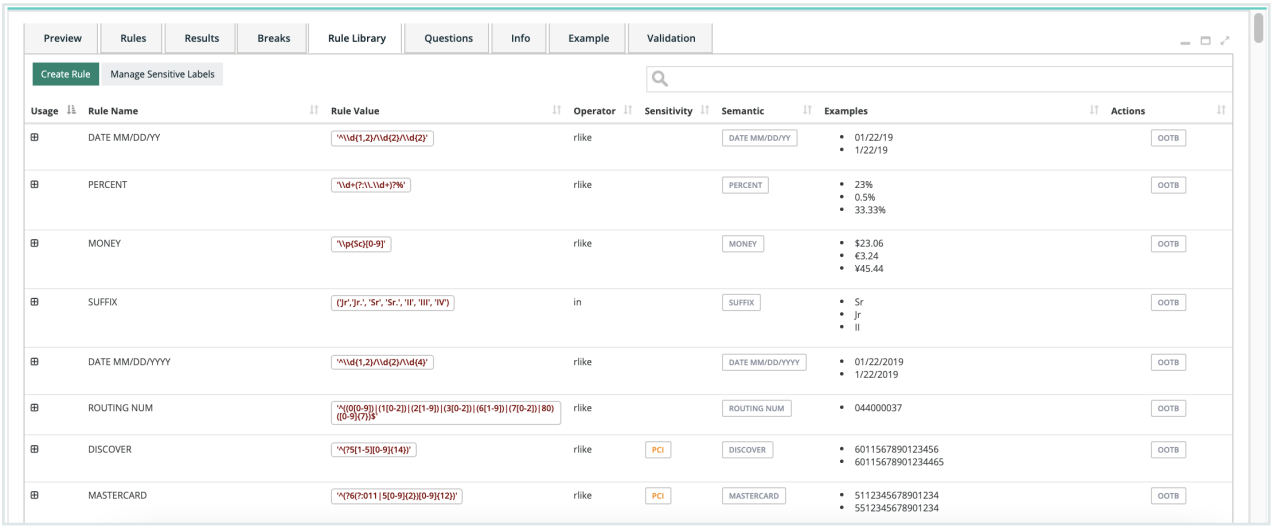
> Note Customized discovery routines can be run using the Rule Library together with Rule Discovery.

Rule Library

An organized repository of all your rule templates.



The rule library contains both OOTB and custom-built rule templates.



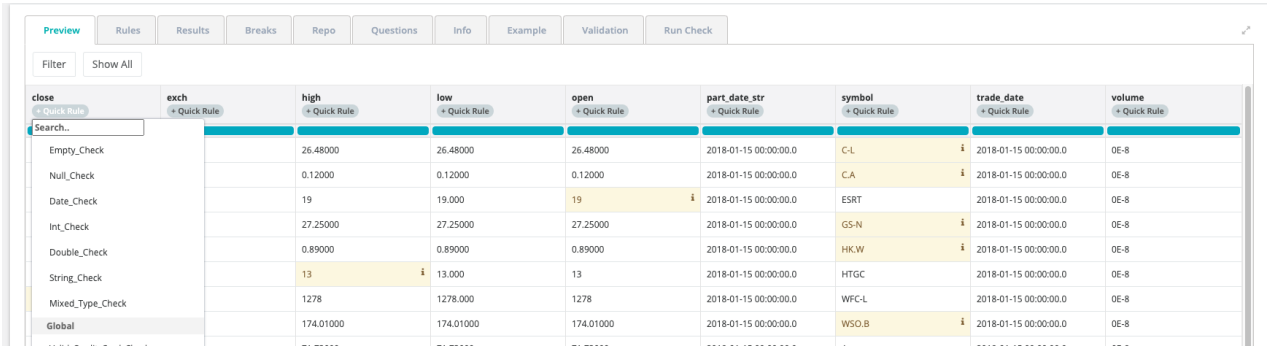
OOTB Rules

Collibra DQ shares all of its out-of-the-box rules with each user/tenant. This makes it easy to get started quickly and lets the team add common rules for specific use cases.

Below is a list of one-click rules that can be added to any data set. It is important to note that Collibra DQ often self-identifies these columns and automatically provides the proper protection.

- Email
- Zip
- Credit Card
- SSN
- EIN
- State Code
- Phone
- Gender
- IP Address
- Date
- Int
- Double

You can also apply common Data Type and Global rules from the Quick Rule dropdown under the Preview tab.



Customized Rules

Add to the Rule Library

Create a rule once using the Create Generic Rule builder and re-use the rule across any column on any data set. This is called a global rule, or a rule in the Rule Library that you can use for global use across many data sets. Collibra Data Quality substitutes the data set and column to which the rule applies at runtime. This commonly saves hundreds of redundant rules that do the same thing but on different column names.

To add a rule to the Rule Library:

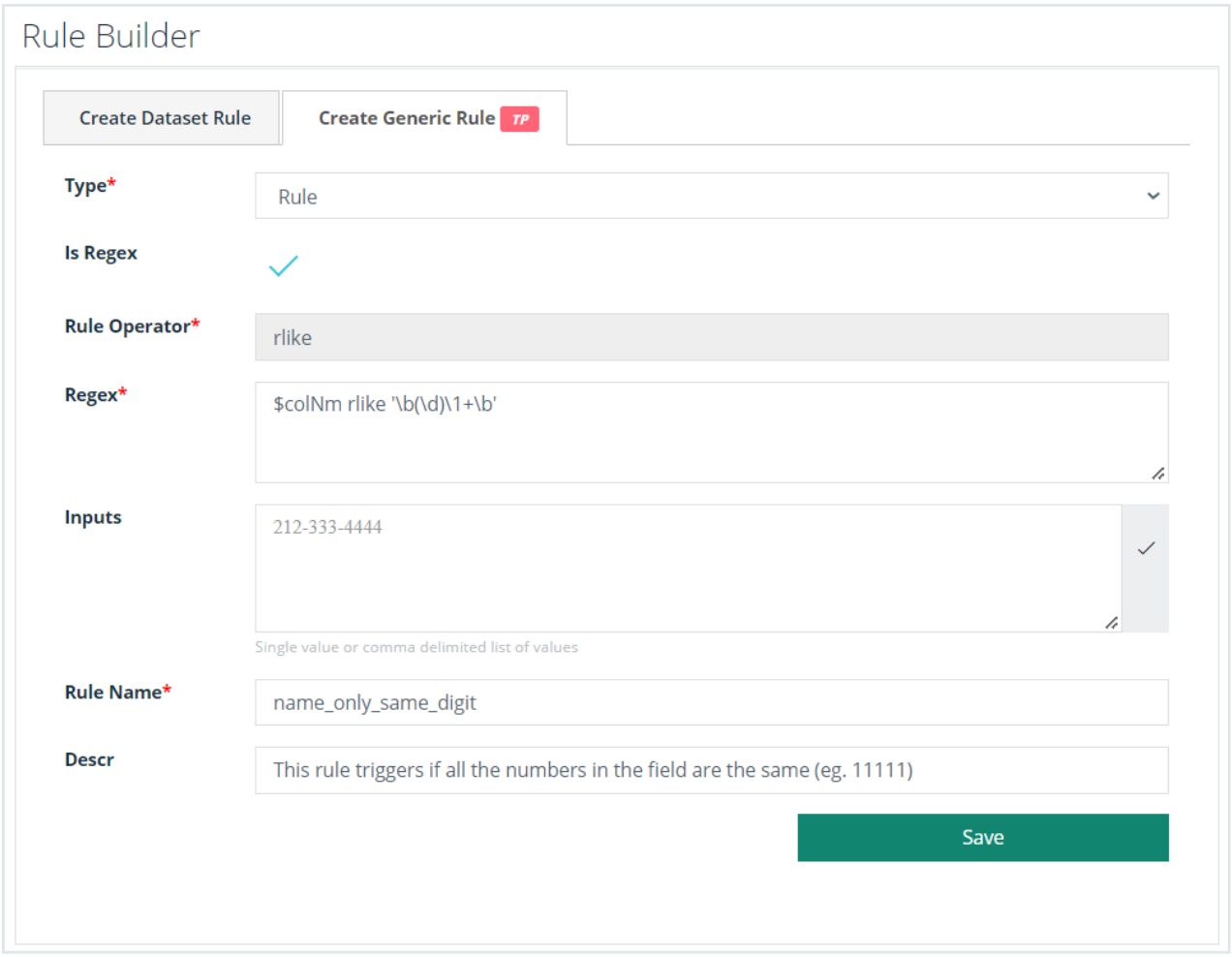
1. Select a data set from the search bar on the Rule Builder page.
2. Click the **Create Generic Rule** tab.

3. Enter the required information.

Option	Description
Type	The type of rule being created.
Is Regex	Select Is Regex if your rule is a Regex rule.
Rule Operator	rlike is the default operator.
Where	Enter your query. Only available for non-Regex rules.
Regex	Enter your Regex query. Only available for Regex rules.
Input	Enter your Regex input values as single values or a comma-delimited list of values. Optional field only available for Regex rules.
Rule name	Enter a unique name for your rule. This is stored in the Rule Library once the rule is saved. Required field for both Regex and non-Regex rules.
Descr	Enter a description for your rule. Optional field for both Regex and non-Regex rules.

4. Click **Save**.

The Rule Library hosts out of the box and custom global rules. See data concepts and semantics for advanced use of global rules.



The Rule Library hosts out-of-the-box and custom global rules. See Rule Discovery for advanced use of global rules.

Rule Discovery

Rule Preview

Set a Preview Limit

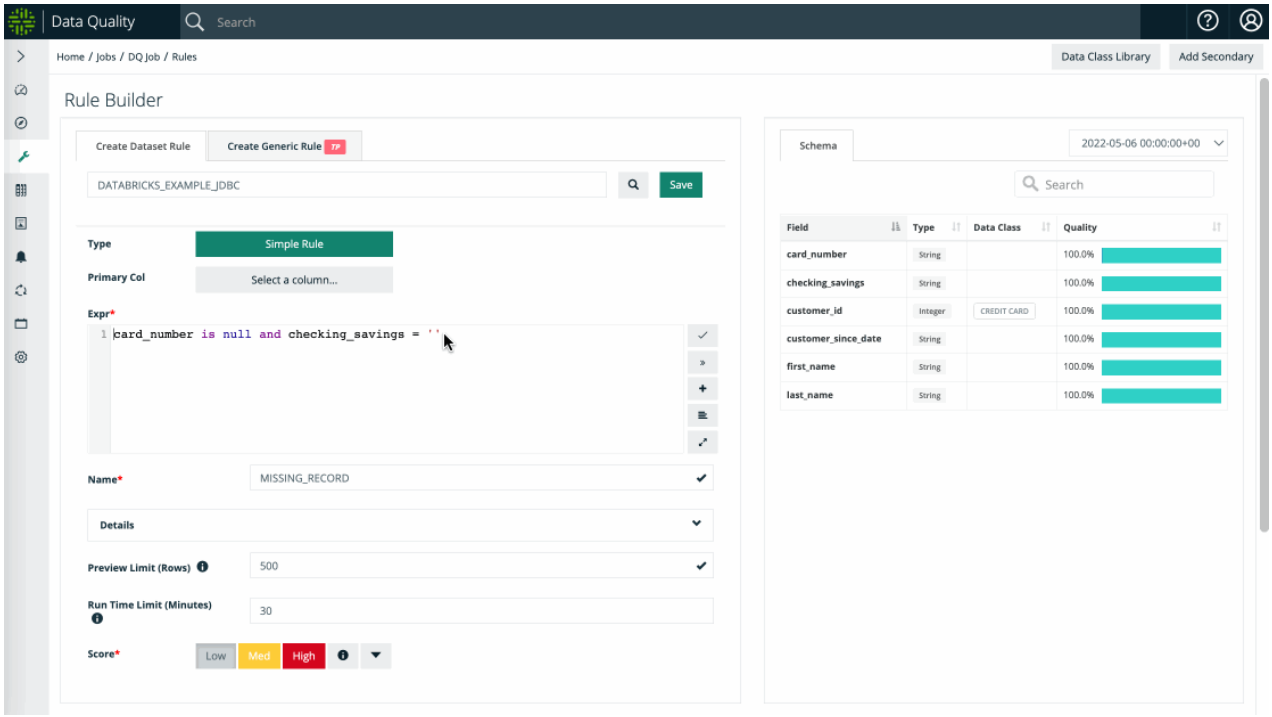
Rule Previews allow you to set Preview Limits by the row, enabling you to drill in to your data even further. Limits can be set to any positive number. 6 is default.

> Note Best practice is to apply a limit of 1000 or less. Increasing the limit above 1000 should only be done if your Postgres has sufficient CPU and memory.

Set a preview limit

1. Create a new **DQ Job** and navigate to the **Rule Builder** page.
2. Select your dataset and click the **Search** button.
3. Select a rule type from the **Select a type** button.
4. Create your SQL rule in the **Expr** field and enter a unique name for your rule in the **Name** field.
5. Enter an even number in **Preview Limit (Rows)**.
6. Click **Save**.
7. Run the **DQ Job** to execute the rules assigned to the dataset.

> Note Freeform and Simple rule are the only two rule types supported at this time.

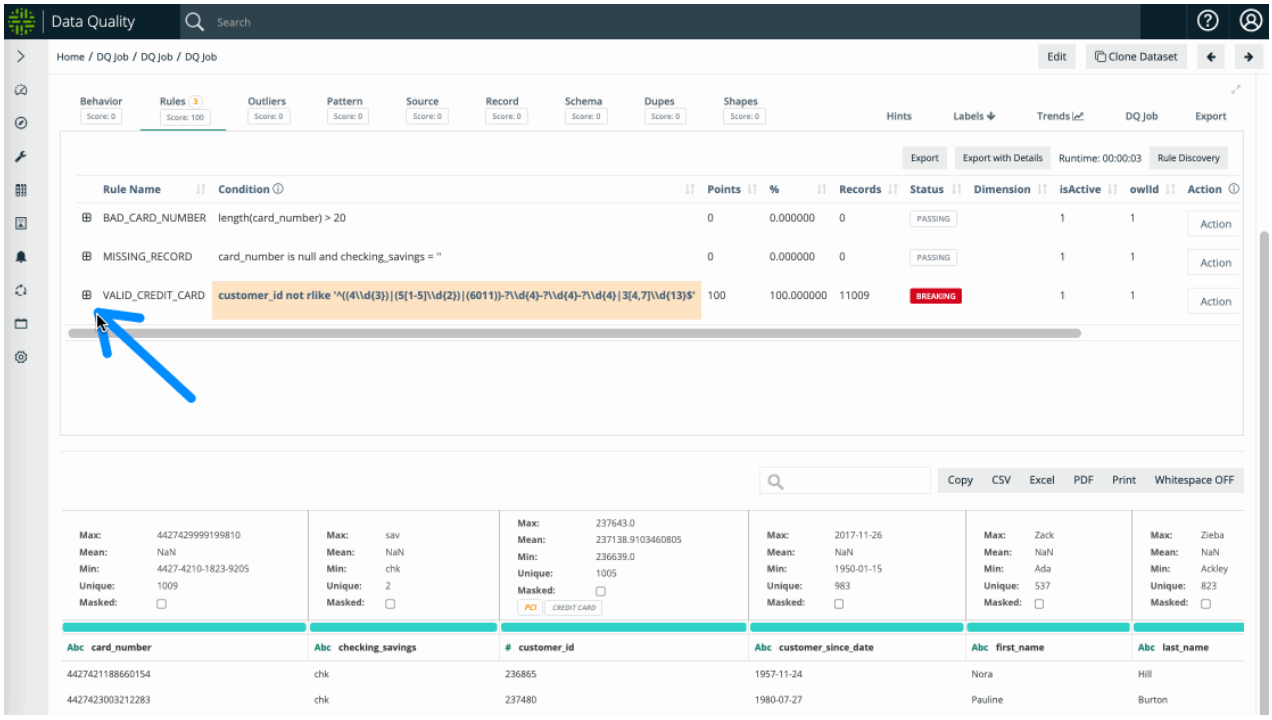


View and Export Results

Another key feature of Rule Previews is that you have the ability to easily export the details of your drill-in.

View and export your results

1. Click the + icon in the **Rule Name** column to expand the table to view your results.
2. Click **Export with Details** to download the break records to your local machine.



> Note Rule preview is currently available for Freeform and Simple rules.

Rule Discovery

Custom data discovery and enforcement using rule templates (data concepts and semantics).

Data Categories

A data category is the category or family of a data set, for example, stock data, interest rate data, etc. By giving data categories, or classifying data sets, we can transfer (apply) common understanding, rules, and ML to data sets. This allows data stewards to set up concepts once and enables organizations to unify and standardize common rules and terms, solving many metadata scale challenges.

> Note Data set level
> **Security reference data** - Bloomberg financial data - **home loan data** - mortgage application data.

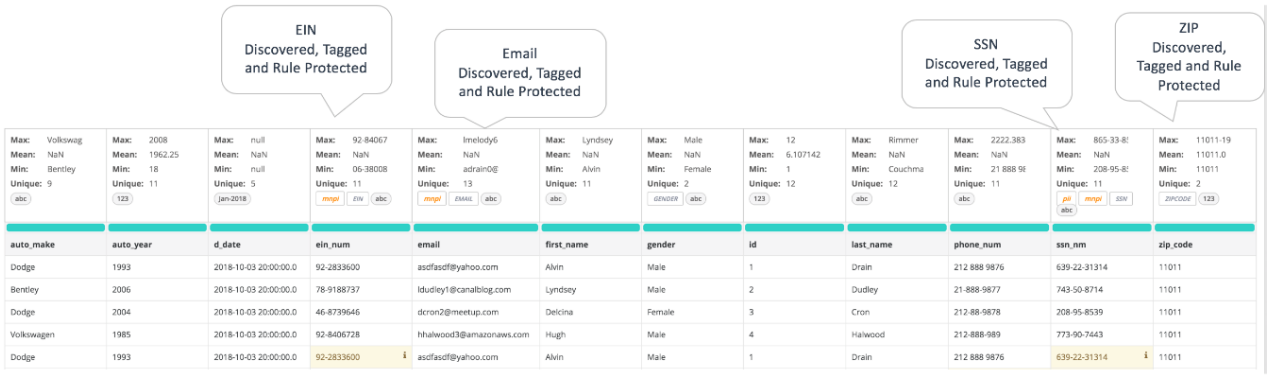
Data Classes

> Note Column Level
> EMAIL, ZIP CODE, SSN, CUSIP, GENDER, ADDRESS, CURRENCY CD, SKU, EIN, IP ADDRESS, PHONE, LICENSE, VIN, CREDIT CARD

A data class is the semantic type of a column or attribute of a data set, for example email, zip code, and so on. All columns have a physical type, such as String, Int, and Date, but the semantic understanding of what type of String is in the column can be very important. Data classes allow Collibra Data Quality to enforce DQ validation rules out of the box.

Collibra Data Quality's semantic scanning self-identifies standard columns and automatically provides the proper protection. This makes it easy to get started adding common rules for specific use cases.

Collibra Data Quality offers out of the box rules for 1-click rule creation



Run Discovery

With the **Run Discovery** modal, you can run a **DQ Scan** to detect for the semantics assigned to a selected data concept. The Run Discovery algorithm automatically selects

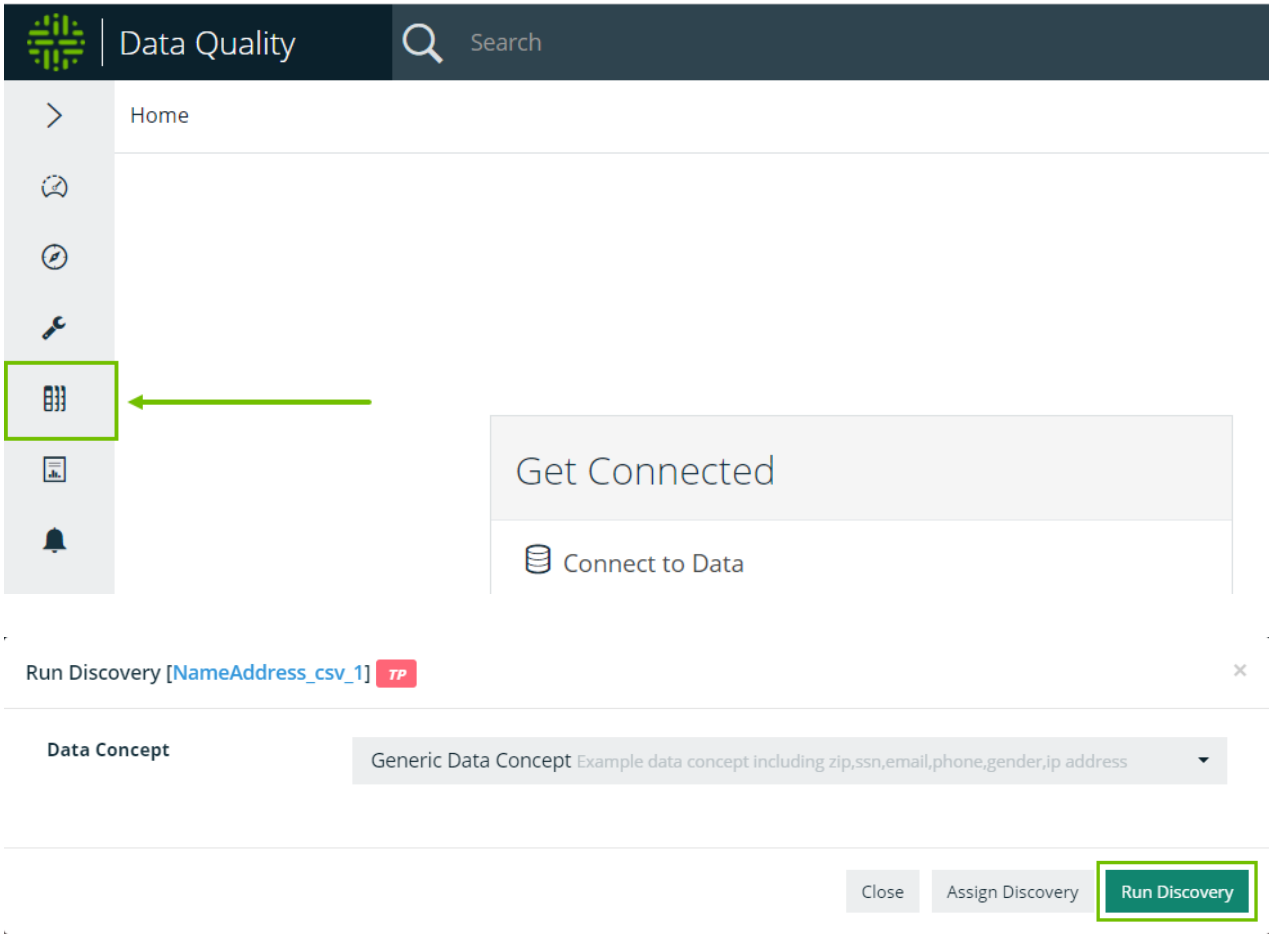
the best match if a column matches two or more data classes. Data class match criteria are determined by percent match and name distance.

You can access the Run Discovery feature via:

- **Catalog**
- **DQ Job**

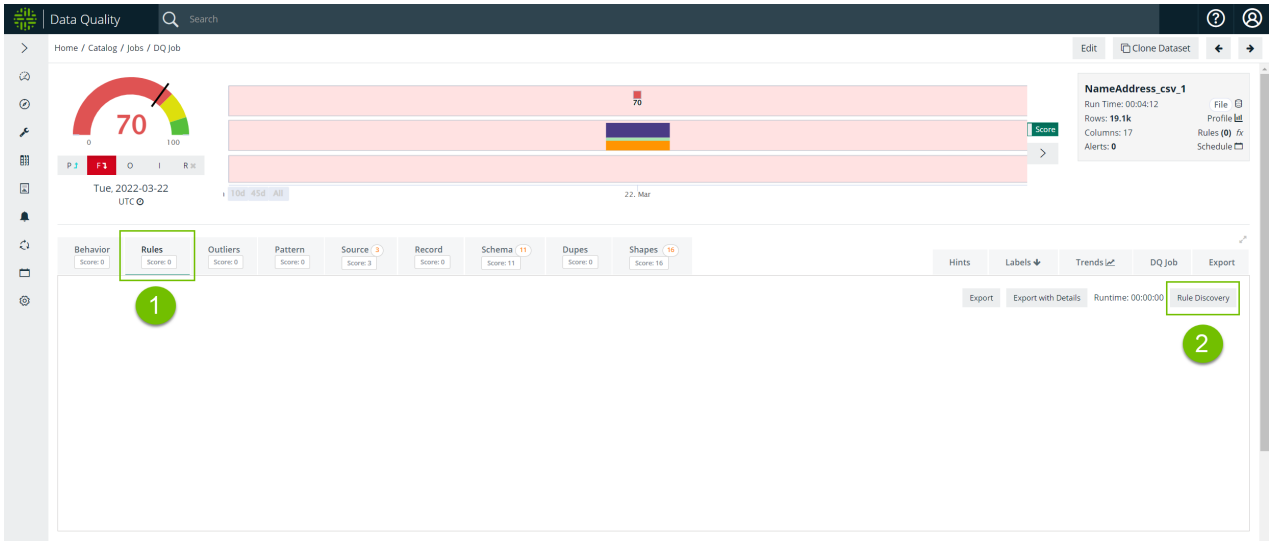
Via Catalog

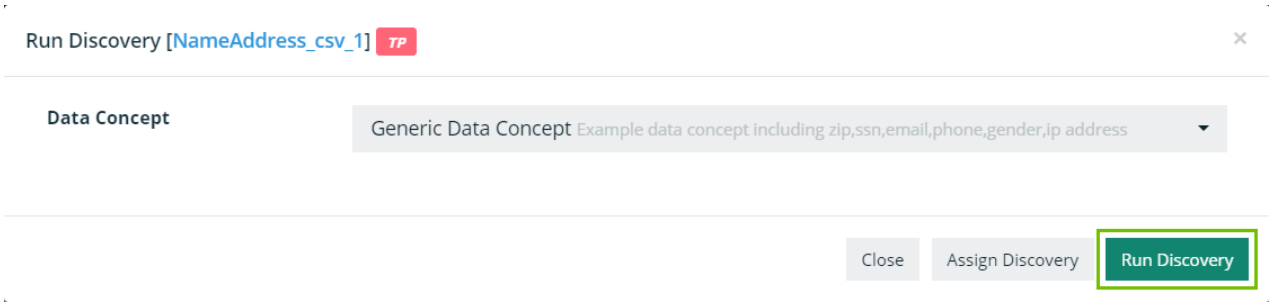
1. In **Catalog**, select your dataset.
2. In the **Actions** dropdown menu, click **Data Concept**.
3. Select an option from the **Data Concept** dropdown and click **Run Discovery**.



Via DQ Job

1. From the **DQ Job** page, select your DQ Job.
2. Click the **Rules** tab in your DQ Job.
3. Click the **Rule Discovery** button.
4. In the **Data Concept** window, select your Data Concept.
5. Click **Run Discovery**.





Sensitive Data

> Note Column Level
>
> **PII** - personally identifiable information **MNPI** - materially non public information
> **PCI** - credit information like a credit card number **PHI** - HIPAA medical information

Data Discovery: The Power of Combining All Three into One Domain

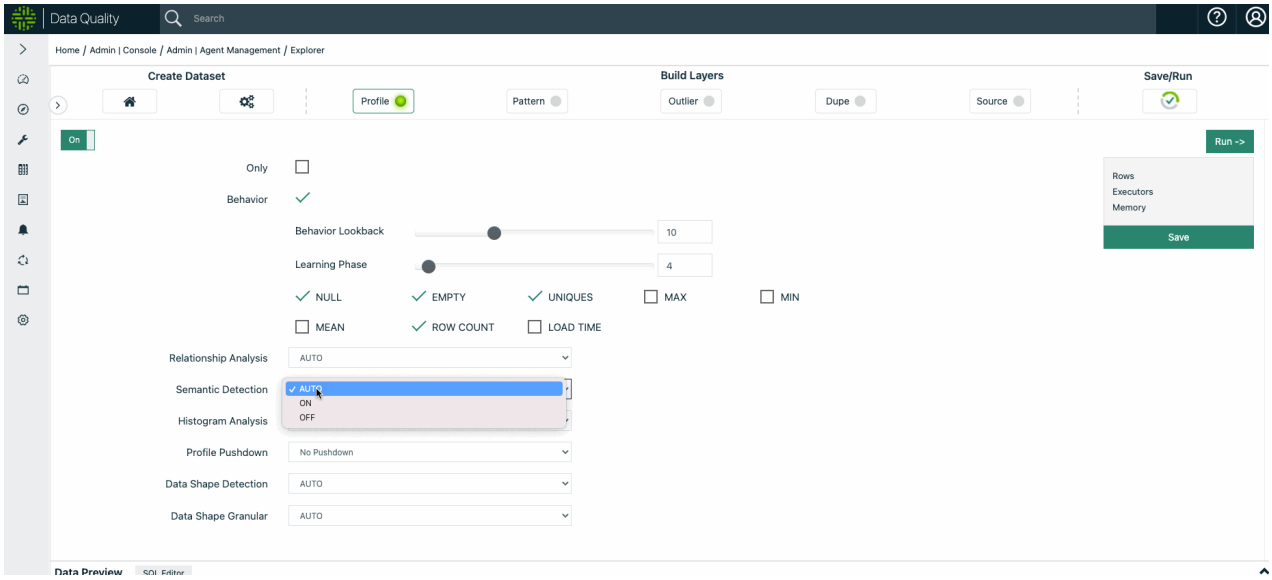
Now imagine if you could classify your datasets as concepts, then automatically have all the columns be recognized semantically(with validation rules in place) as well as have the columns labeled with sensitivity tags. It might look something like the following screenshot.

Max: zwiltshaw6c@ **Mean:** NaN **Min:** aabramow9n **Unique:** 1000 mnpi EMAIL	**Max:** n/a **Mean:** NaN **Min:** Accident &He **Unique:** 117	**Max:** 99-9985119 **Mean:** NaN **Min:** 00-0340993 **Unique:** 1000 mnpi EIN	**Max:** NULL **Mean:** NaN **Min:** NULL **Unique:** 1	**Max:** Zorine **Mean:** NaN **Min:** Abba **Unique:** 936	**Max:** Male **Mean:** NaN **Min:** Female **Unique:** 2 GENDER	**Max:** M **Mean:** NaN **Min:** F **Unique:** 2 GENDER
Abc email	**Abc emplmt_industry**	**Abc empl_num**	**Abc emp_nul_col**	**Abc first_name**	**Abc gender**	**Abc gndr_abbrv_cd**
eokynsillaghec1@domainmarket.com	Oil & Gas Production	78-1449687	NULL	Earvin	Male	F
abrocklesbyfa@dailymail.co.uk	n/a	83-2505150	NULL	Alec	Male	M
hlovedaypp@samsung.com	Computer Software: Prepackaged Software	27-3810518	NULL	Heidi	Female	M
bfrowengw@sfgate.com	Industrial Machinery/Components	03-8021700	NULL	Berkeley	Male	M
codayrl@istockphoto.com	n/a	38-3013395	NULL	Carmencita	Female	F
kcrankhornew@loc.gov	Business Services	94-0920083	NULL	Killy	Male	F
sglasner4d@github.io	Clothing/Shoe/Accessory Stores	31-7241123	NULL	Saunder	Male	F
cfollissfw@discovery.com	Semiconductors	43-9079862	NULL	Cinda	Female	F
nbartelsellis5r@wikimedia.org	n/a	37-8683740	NULL	Nevins	Male	M

Steps

Step 1: Create a DQ Job with Semantic Detection turned on.

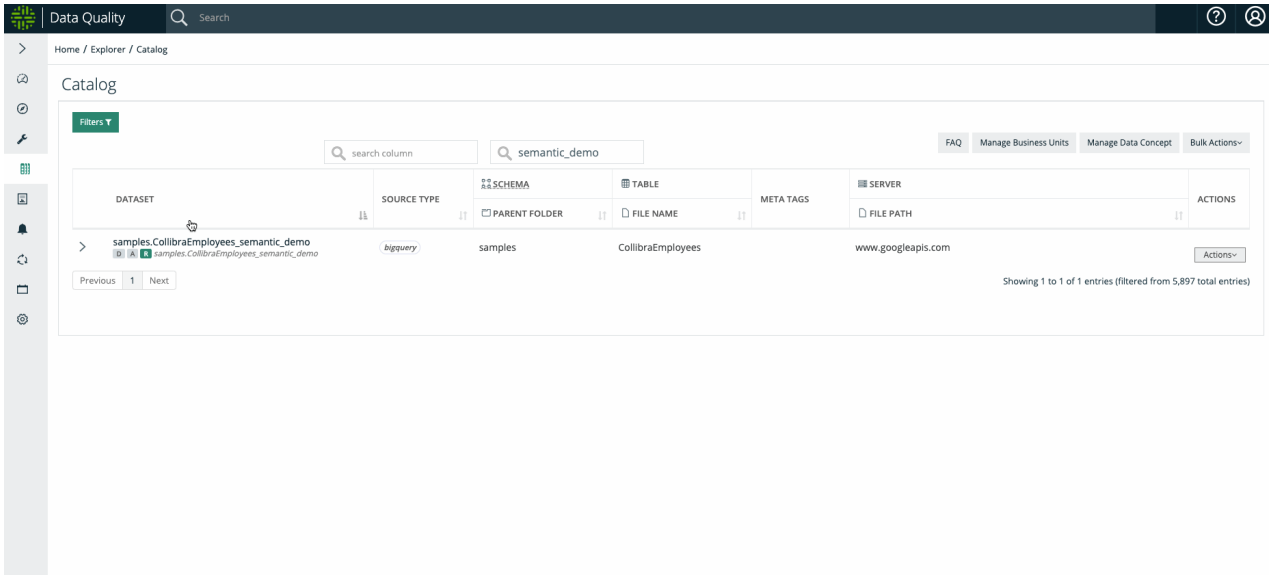
From the Profile options page, create a new DQ Job and select ON from the Semantic Detection dropdown.



Step 2: In Catalog, select and apply your Data Concept.

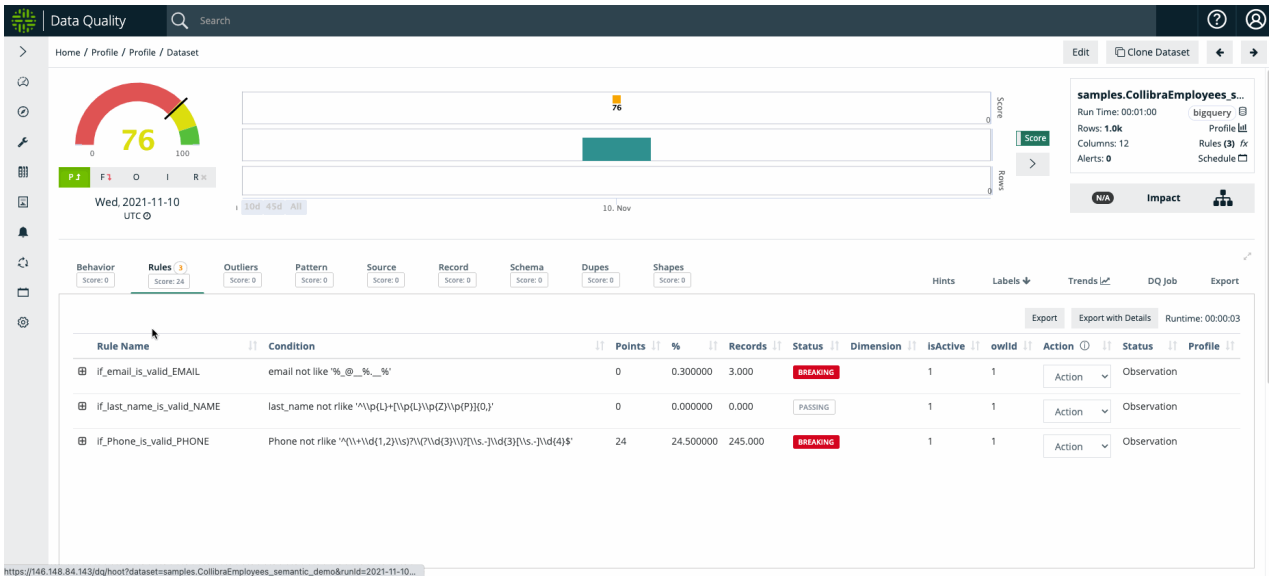
Navigate to your dataset in **Catalog** and select the Data Concept you would like to apply with the **Actions** dropdown menu.

See below sections on how to **Administer Data Concepts** as well as how to **Create and Manage Semantics**.



Step 3: Rerun your DQ Job with applied Data Concept.

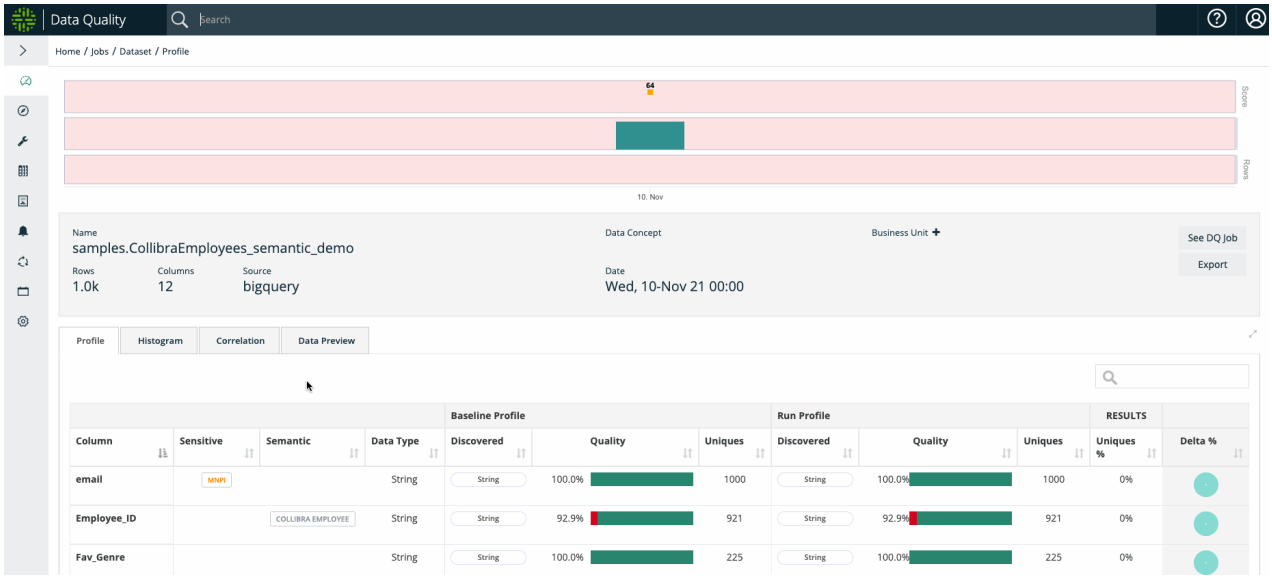
Please rerun your DQ Job so that Collibra Data Quality can 1) profile your data, 2) auto-generate the rules based on the Semantics under the Data Concept, and 3) highlight any break records.



Success! Review Findings

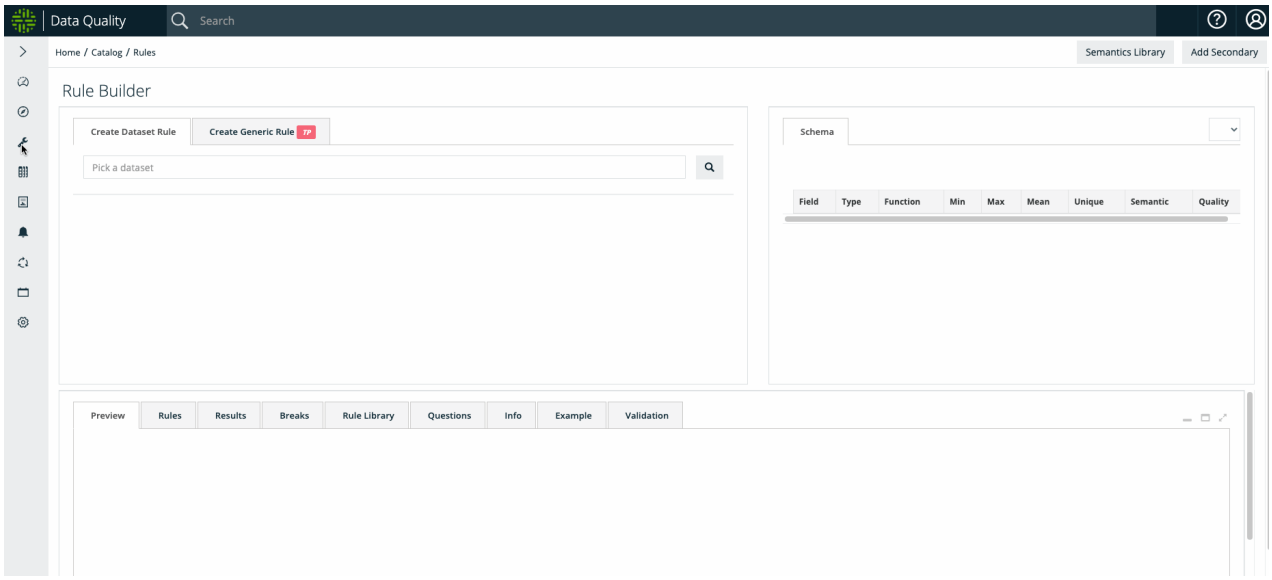
On the Profile page, observe the newly tagged Semantics on the applicable columns.

On the DQ Job page, browse your newly created rules based on Semantics, as well as any corresponding rule breaks.



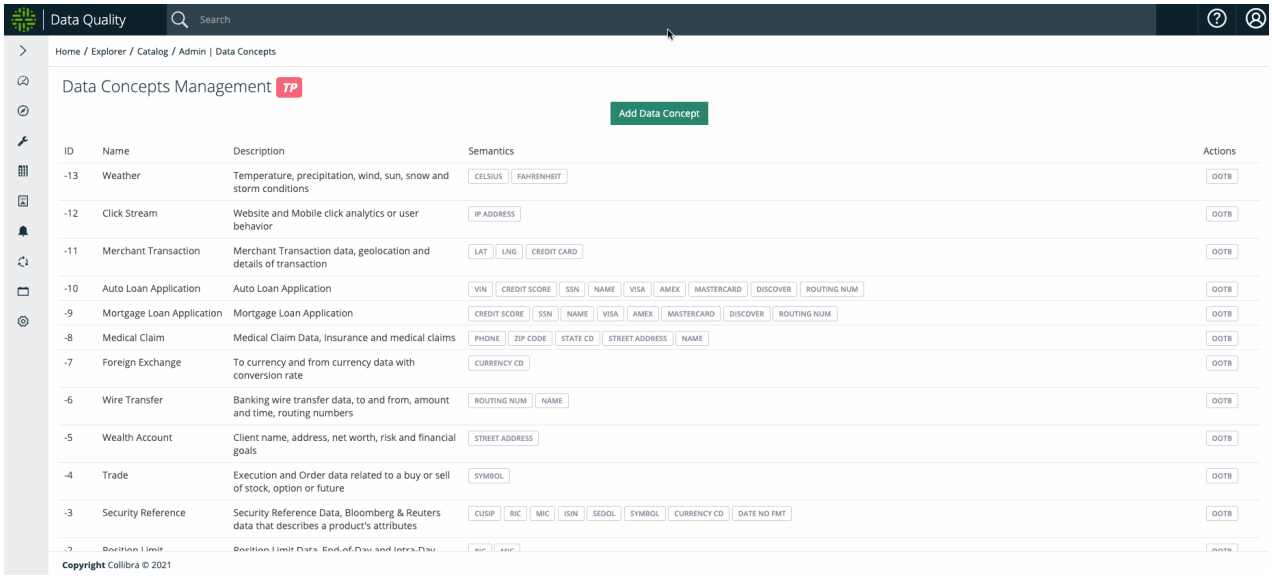
Creating and Managing Semantics

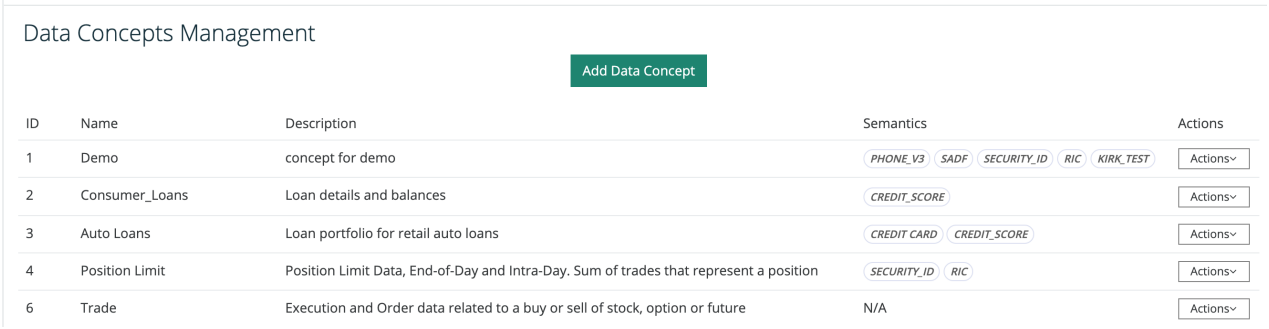
Create, test, and manage your Semantics in Collibra Data Quality in your **Rule Builder** wizard on the **Create Generic Rule** tab. Below is an example of creating a RegEx Semantic.



Administering Data Concepts

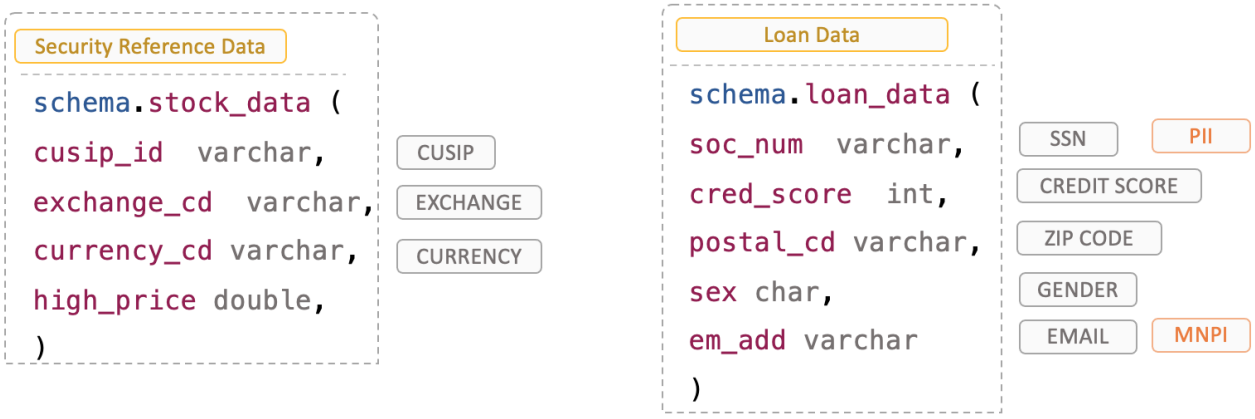
Setup your data concepts once and let the entire organization benefit by unifying all datasets to a common understanding in the admin Data Concepts page.





Physical Schemas to Semantics

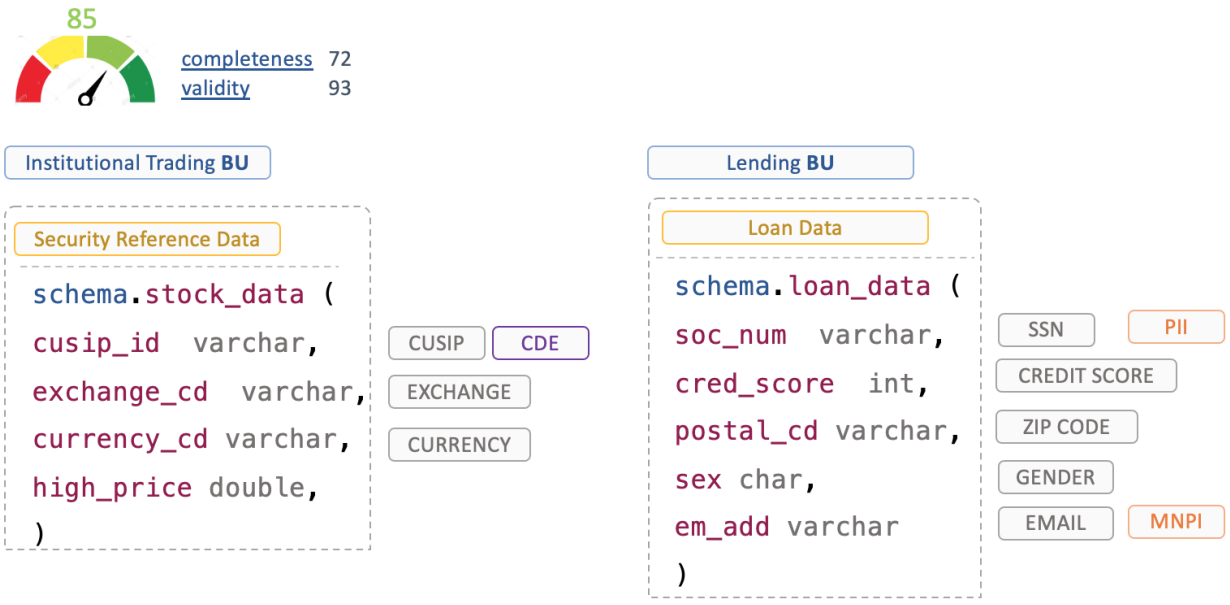
Below you can see the benefit of organized metadata. PDEs or `physical data elements` organized/tagged by semantics. This allows for sub-second searches while in catalog or searching for data to figure out where all your PII data lives, or what systems have "loan data".



Above you can see Data Concepts in Yellow, Semantics in Gray and Sensitive labels in Orange. Enabling you to organize all your data in classes, search and discover types no matter what system they live in or what the PDE column name is. Transforming technical types into business metadata.

Business Unit Roll up Reporting

Now that we have all PDEs discovered and tagged and rolled up into business terms, we can roll up technical assets like database tables and files into business reports across departments and non technical concepts.



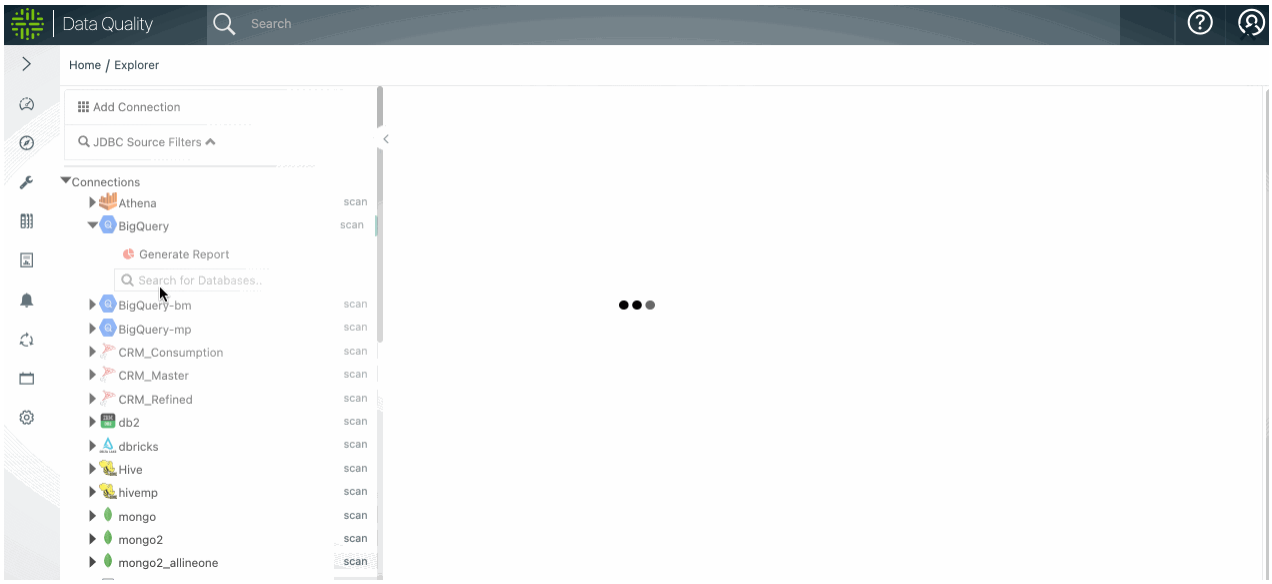
More

Collibra DQ advanced features.

When specific DQ challenges require specific DQ detection techniques, Collibra DQ offers a wide variety of advanced functionality. While Schema and Shapes utilize auto-discovery, other detection algorithms are best suited for users that understand their data and have specific use-cases in mind. Read more to understand if specific dimensions can be applied to your data.

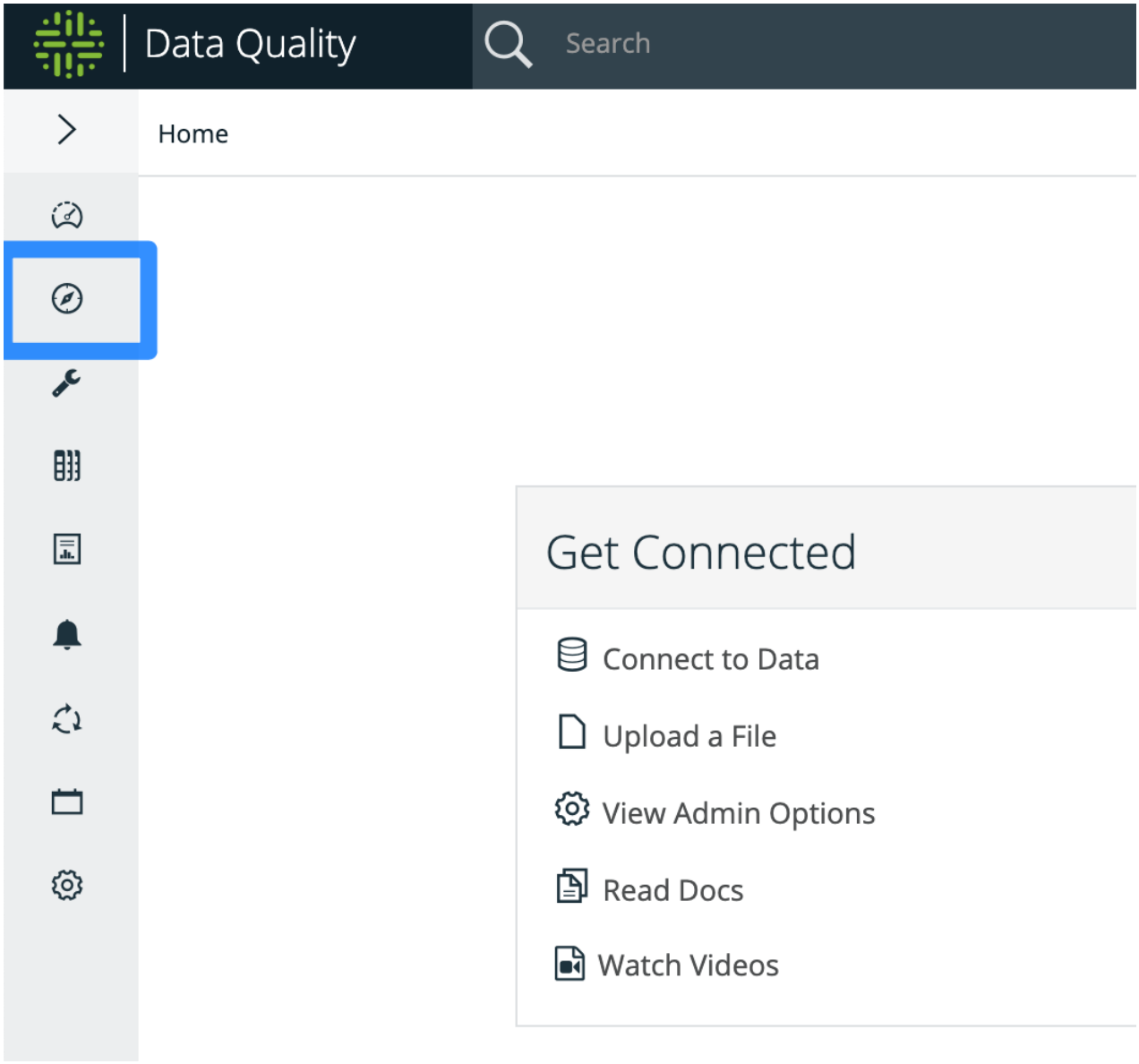
Explorer (no-code)

A no-code option to get started quickly and onboard a data set.



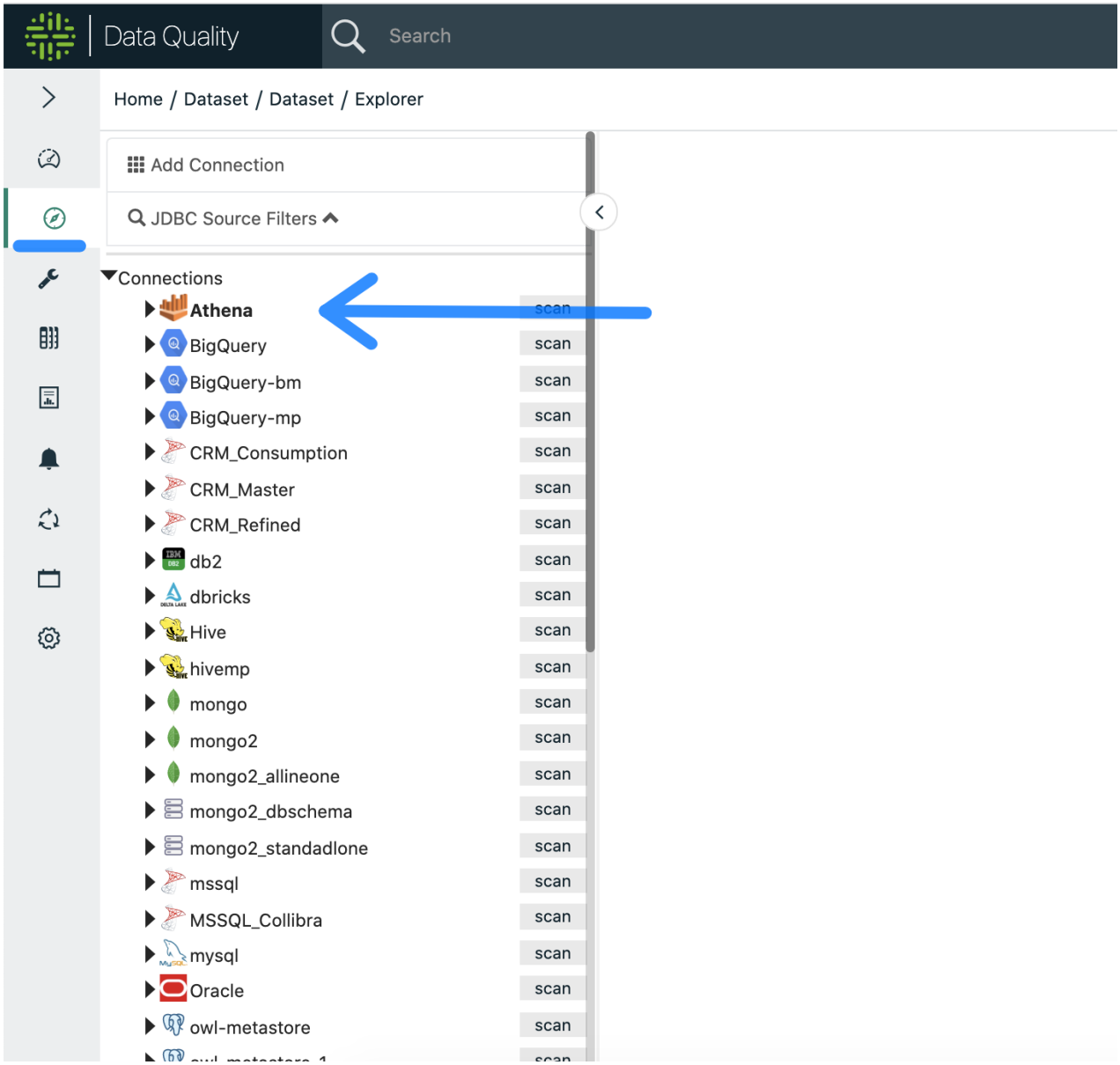
Getting Started

This page can be accessed by clicked the Explorer option (the compass icon).

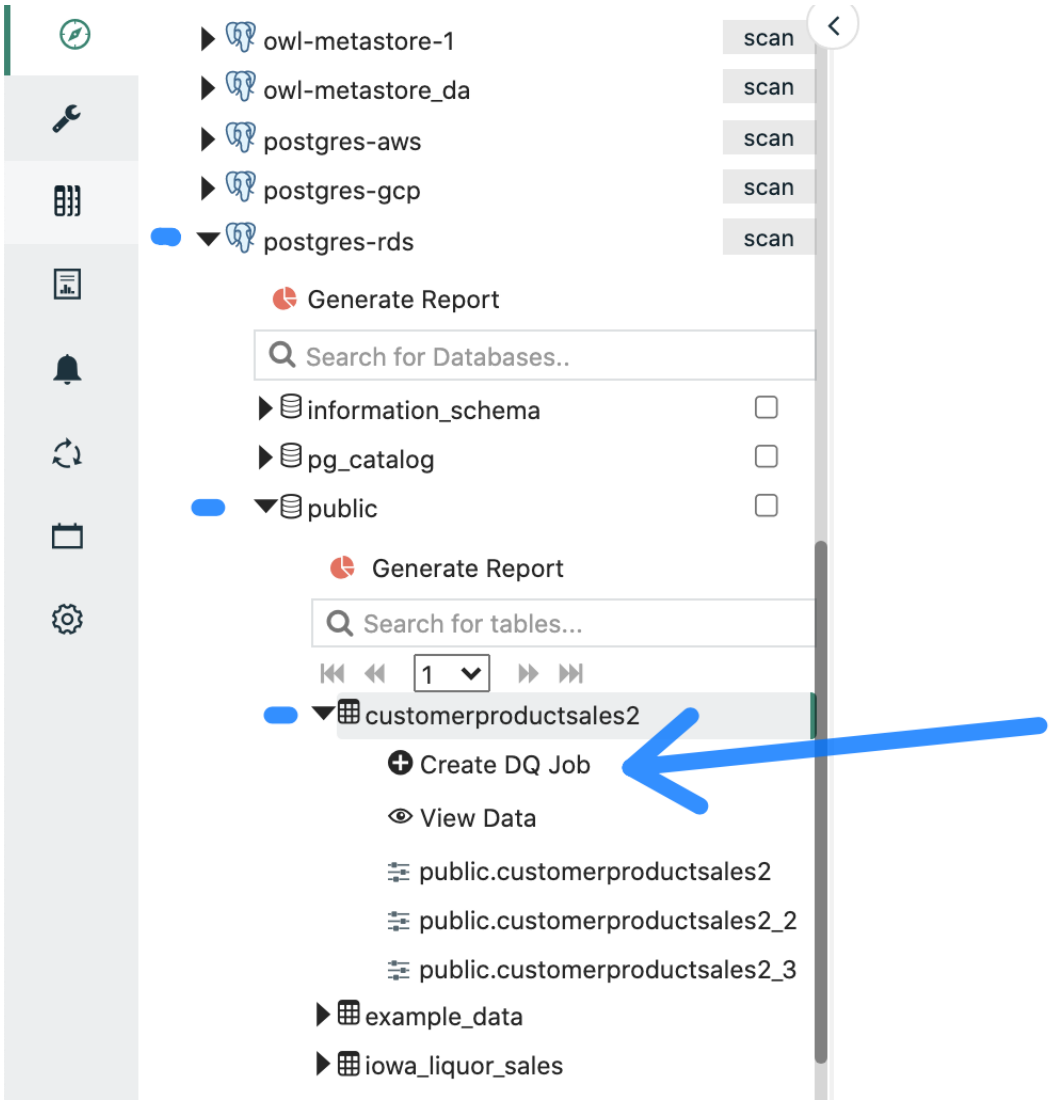


Note All UI functionality has corresponding API endpoints to define, run, and get results programmatically.

Select Your Data Source



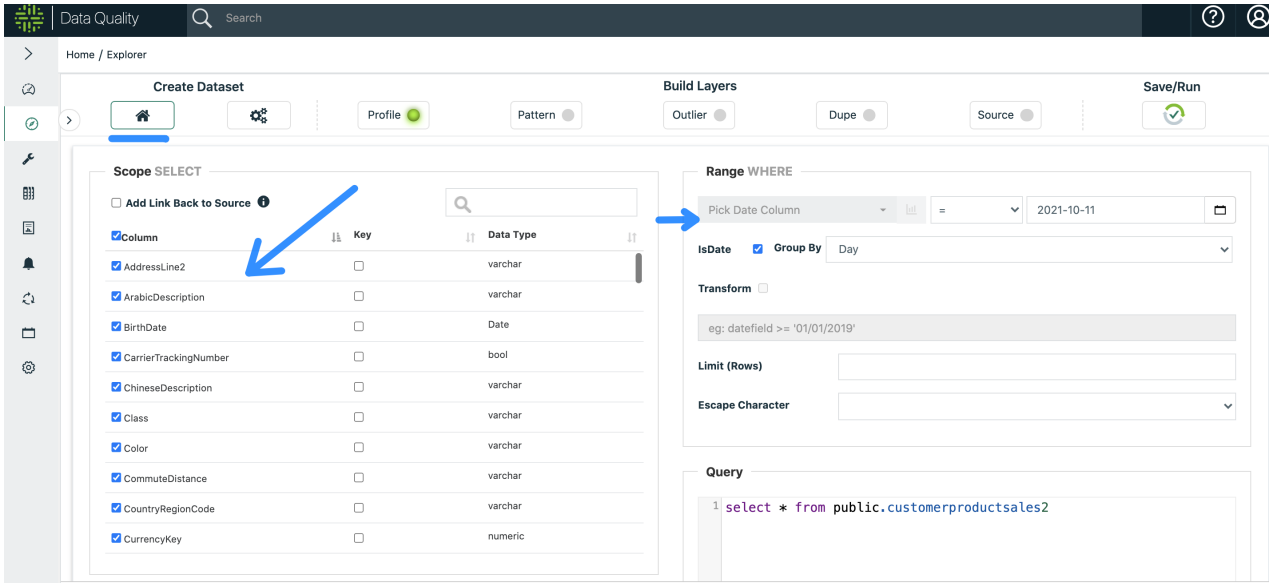
Create a new DQ Job by clicking +Create DQ Job



View Data is an interactive option to run queries and explore the data

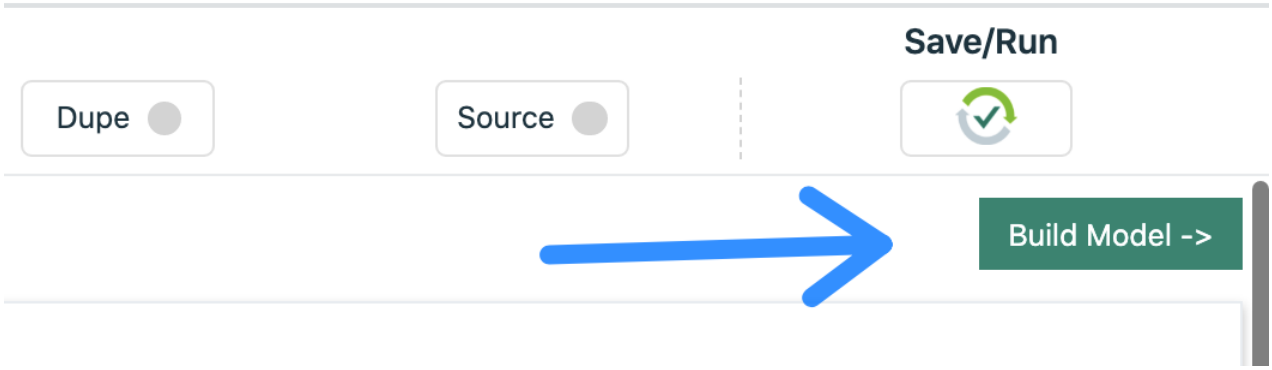
The bar chart icon will take you to a profile page of the dataset created prior to Explorer 2

Select The Scope and Define a Query



Pick Date Column if your dataset contains an appropriate time filter

Click Build Model -> to Save and Continue

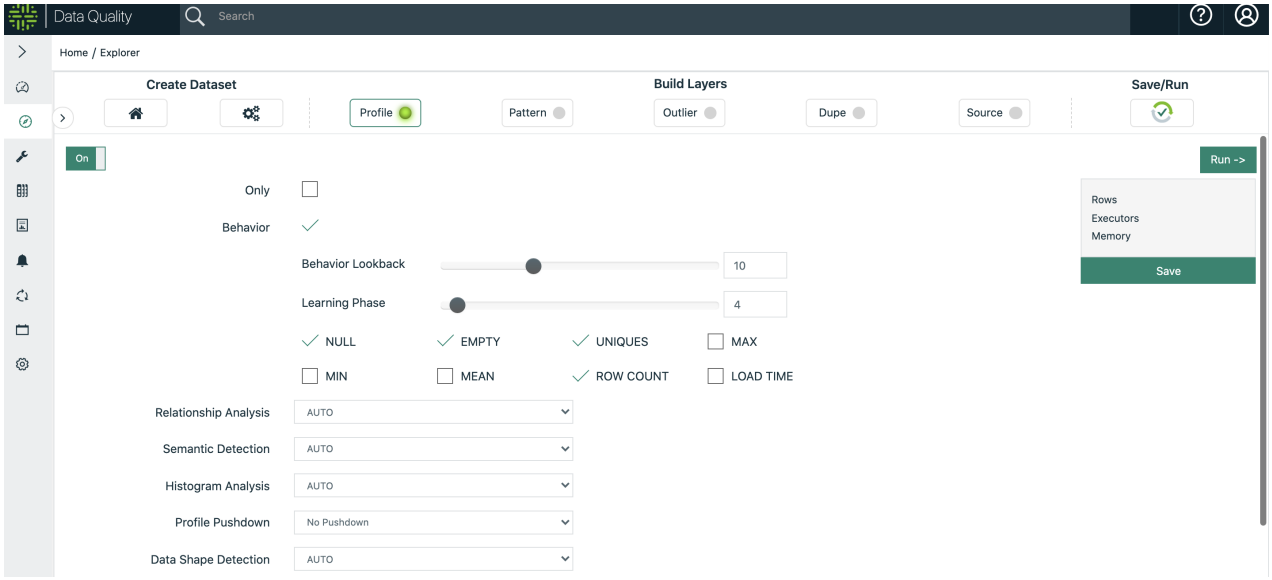


Transform Tab (advanced / optional)

Transform

Click Build Model -> to Save and Continue

Profile



Use the drop-downs to enable different analysis. Best practice is to leave the defaults.

Pattern (advanced / optional)

Toggle on Pattern to enable this layer.

Click +Add to define a group and series of columns.

Patterns (advanced)

Click Save to and Click Outlier to Continue

Outlier (advanced / optional)

Outliers (advanced)

Click Save to and Click Dupe to Continue

Dupe (advanced / optional)

Dupes (advanced)

Click Save to and Click Source to Continue

Source (advanced / optional)

Navigate to the source dataset.

Click Preview to interlace the columns.

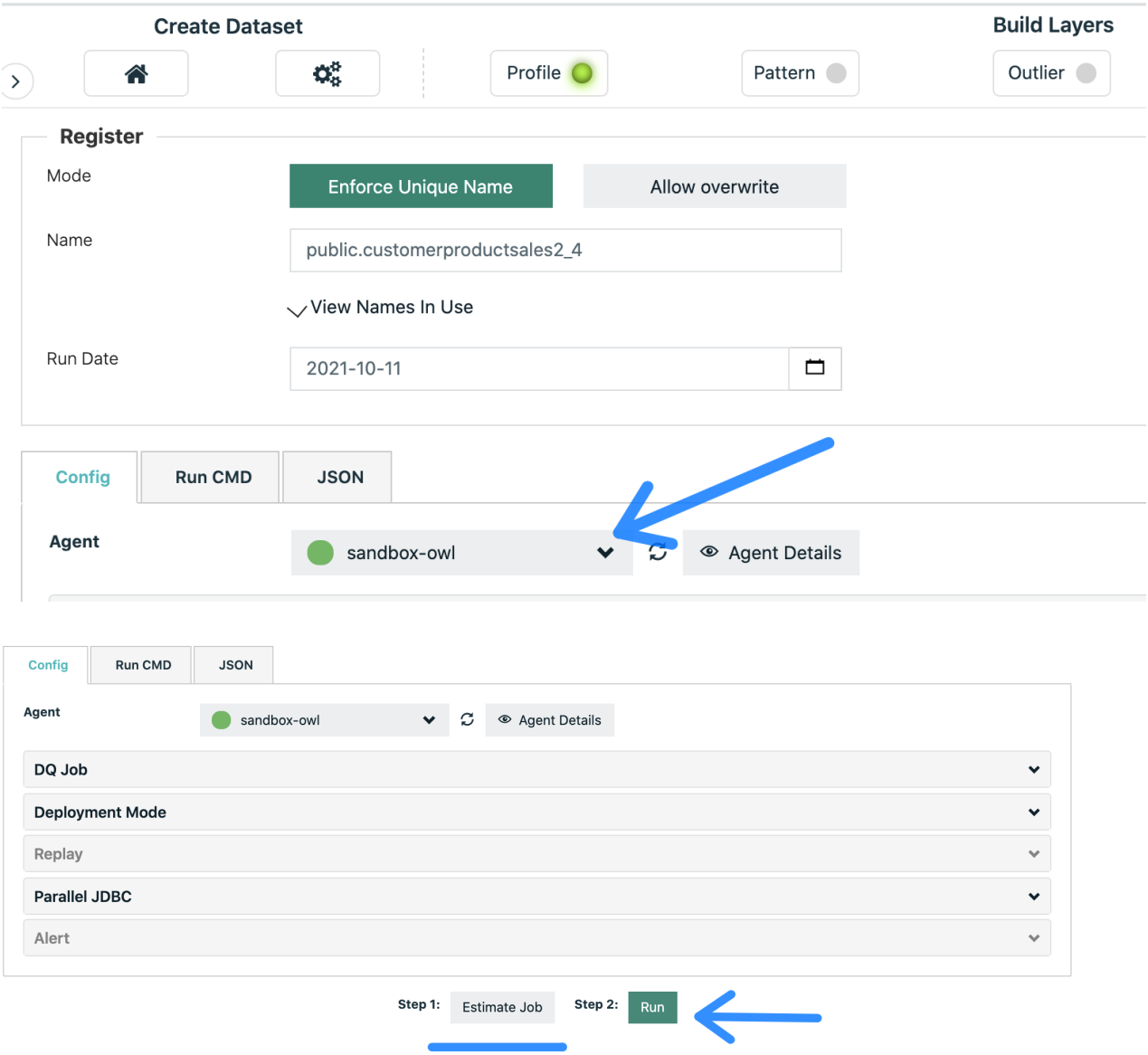
Manually map the columns by dragging left to right or deselect columns.

Source (advanced)

Click Save to and Click Save/Run to Continue

Run

1. Select an agent.
2. Click Estimate Job.
3. Click Run to start the job.



> Note *If you do not see your agent, please verify the agent has been assigned to your connection via Adding a connection to a DQ Agent.

Terminating a DQ Job

If a DQ Job is in progress, incorrectly submitted, or stuck in Staged status, you can terminate the job by clicking the Actions drop-down list and selecting **Terminate job**. If you

have an email address configured for alerts to be sent, two alerts are sent when a job is terminated.

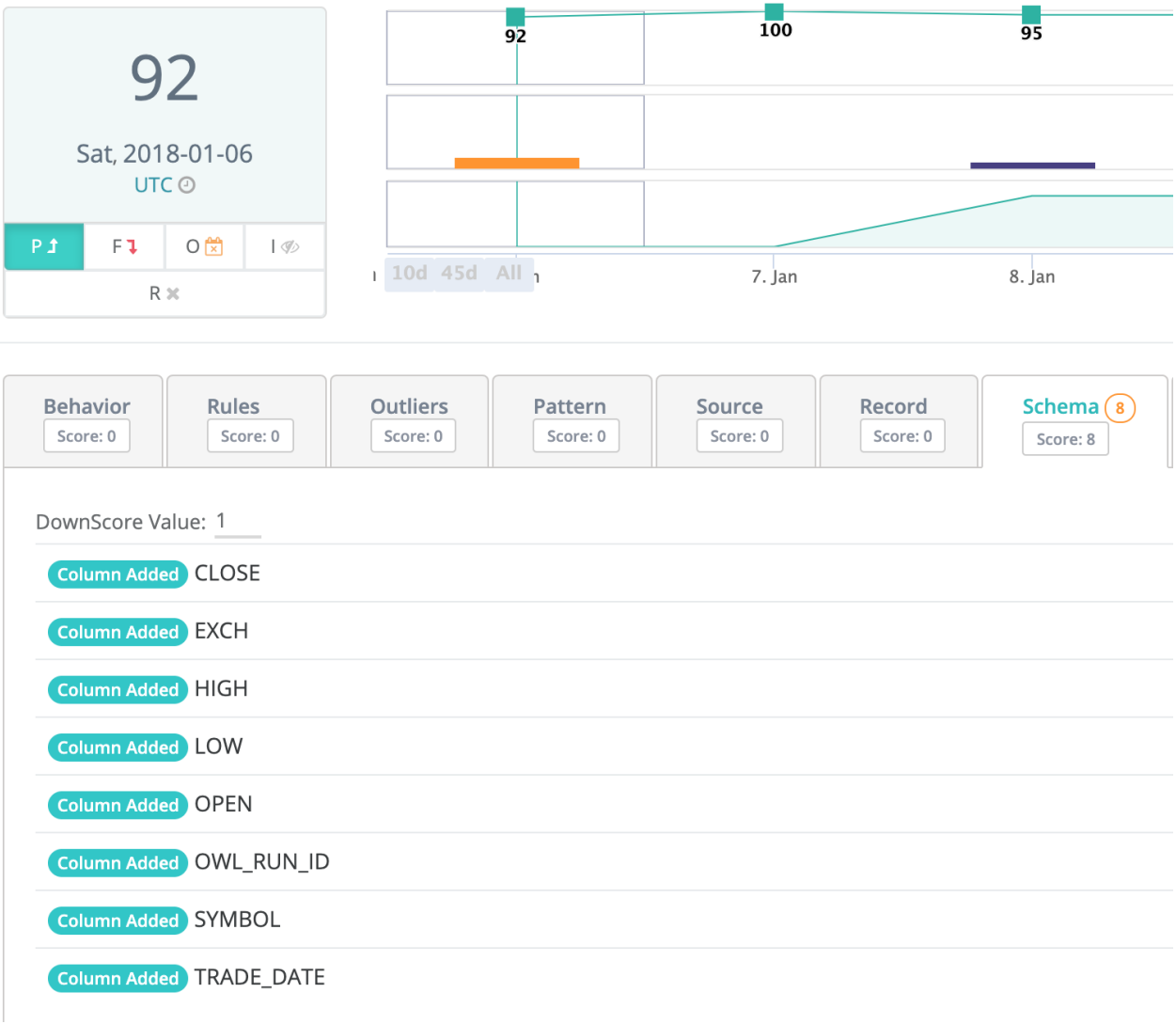
> Note Jobs in the Spark UI display a Finished status when terminated, even though they are terminated from the Collibra DQ UI.

Schema (automatic)

Detect schema evolution and unexpected schema changes.

Data set schemas are the columns or fields that define the dataset. They are often located in the header row of a tabular file or database table. However, JSON and XML are two examples of formats that include schema columns that are not in the header but rather nested throughout the document. OwlDQ automatically without needing to turn on any features detects the schema columns as well as reads or infers their data types (varchar, string, double, decimal, int, date, timestamp, etc.). Owl observes each data set so if a column is ever altered, removed or added it will automatically raise the event via its standard composite scoring system.

Scoring... Alerting... Schema Detection... Automatically



Schema Evolution is one of Owl's 9 DQ dimensions. It can be an important measurement for data stewards to understand how the data set is changing overtime. The orange bar on the chart shows a change in schema and allows for drilling in over time.

Shapes (automatic)

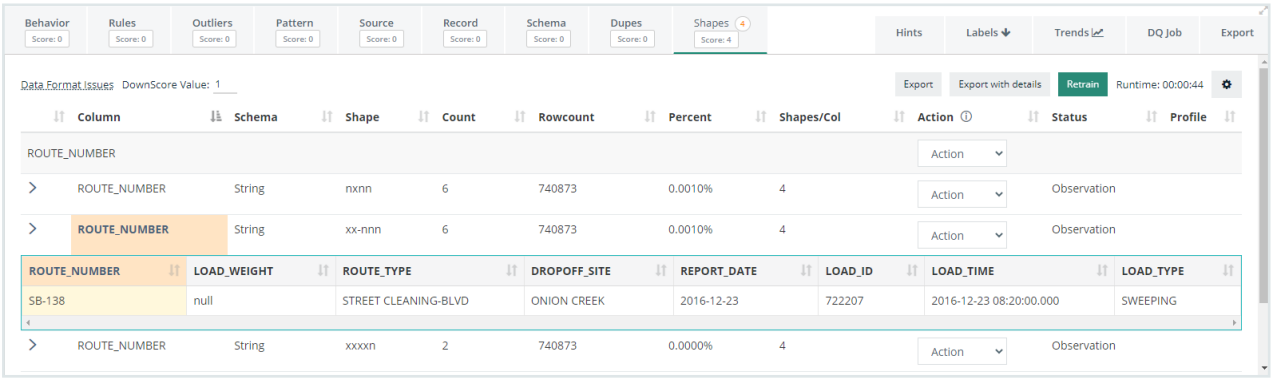
Collibra DQ automatically detects inconsistencies in data formats. These inconsistencies are where Data Scientists spend an enormous amount of time cleaning the data before building a ML model. Many reports have documented that over 80% of the time it takes to build a credible model comes from first understanding all the different formats and then

writing munging or ETL style code to clean it before processing. What about all the patterns the process or person doesn't even know about?

auto_year	phone_num	id
2004	212-88-9878	3
1994	212 88-9880	5
1992	212 888 882	7
1996	212.888.88888	10
1993	212 888 9876	1

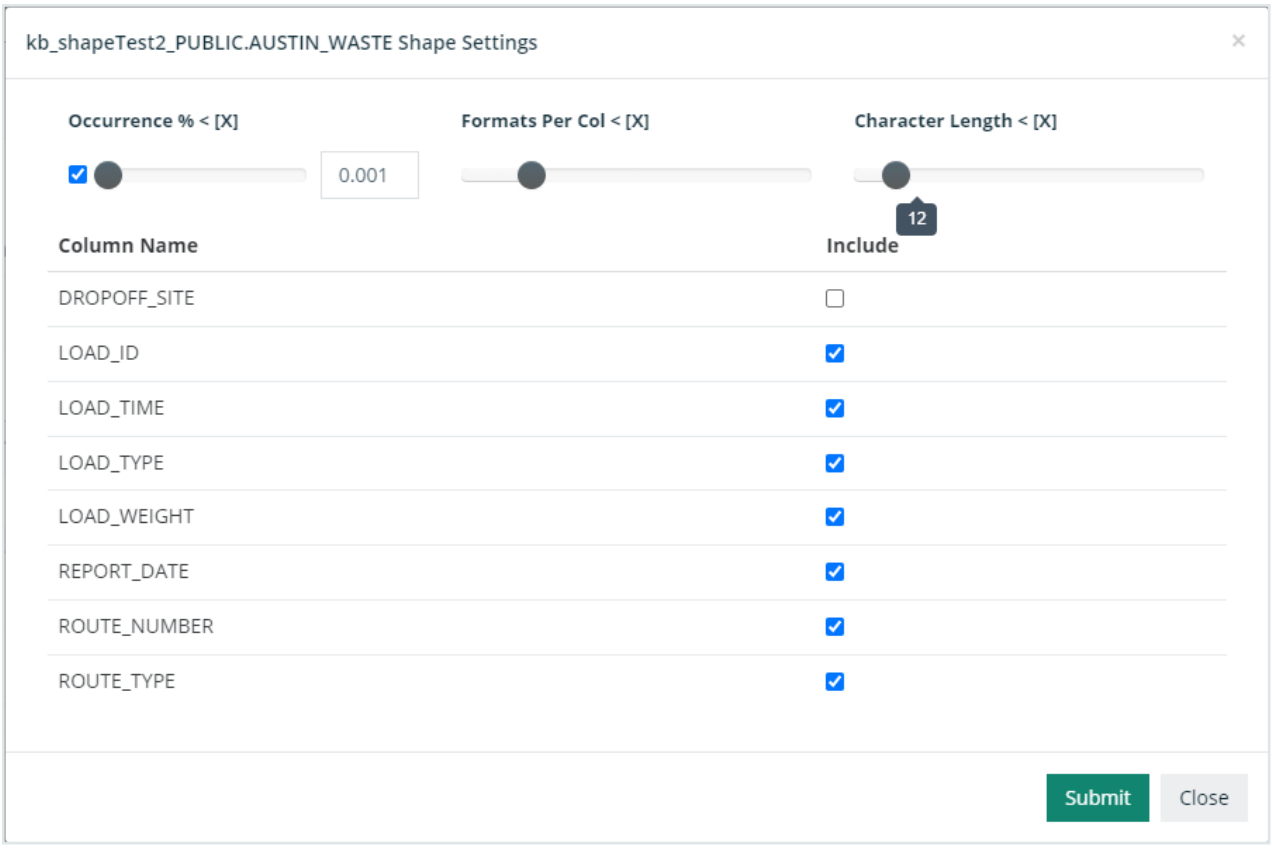
Drill-in to any Shape anomaly and see a visual example

See an itemized list view of the most infrequent or odd shapes in your datasets.



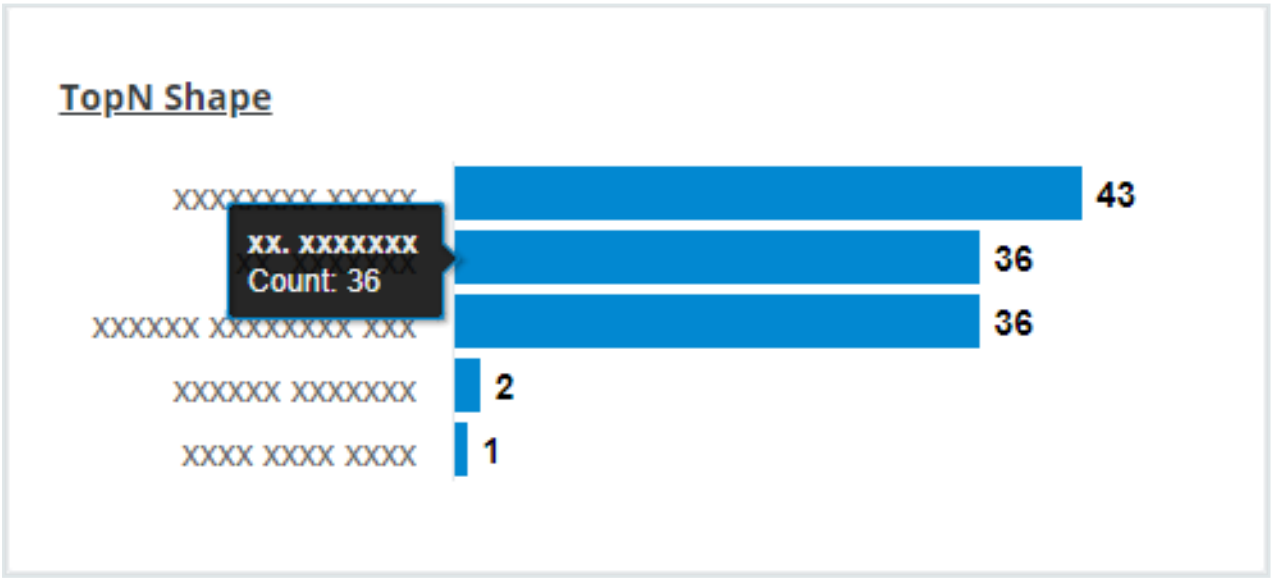
Shape Tuning

Shapes detection is on by default. Click the ⚙ on the Findings page to manually tune the Shape Settings. Click and drag the sliders to adjust the tuning of **Occurrence %**, **Format per Column**, and **Character Length**.



TopN Shape

Beneath the TopN Values chart on the Profile page is the **TopN Shape** chart. The TopN Shape chart represents the exceptional shapes for a particular column on the Shapes tab.



Dupes (advanced)

Dupes, or Duplicates, are values that match other existing values in columns and can be set to detect exact and fuzzy matches. Both exact and fuzzy support case-insensitive matching.

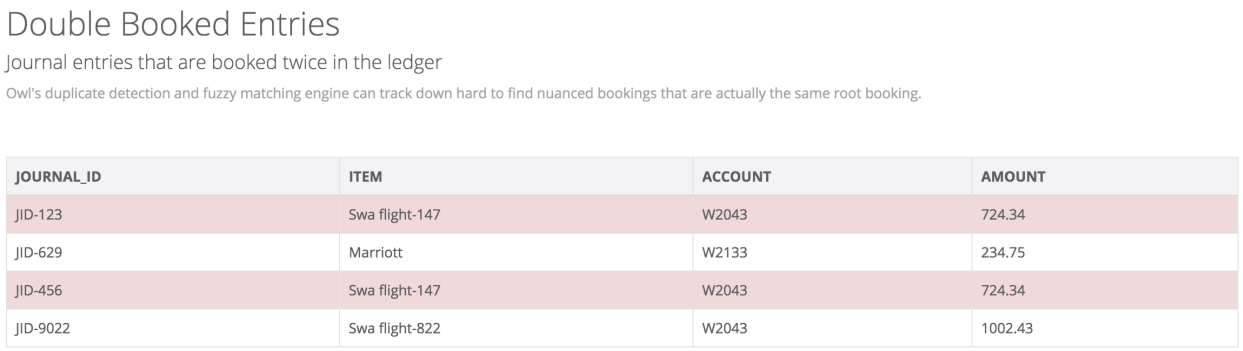
Exact matches detect entries that may match despite formatting issues. For example, Collibra DQ flags `John Doe`, `JOHN DOE`, and `john doe` as exact match dupes of each other. Exact match groups the same dupes and displays their counts in the **occurs** column on the Findings page.

Fuzzy matches detect entries that may match despite misspellings, common spelling variations, and more. For example, Collibra DQ flags `John Doe`, `Johnny Doe`, and `Johhnn Doe` as fuzzy match dupes of each other. The fuzzy match algorithm complexity is O^2 and compares every two fields in the dataset.

When fuzzy match runs on a dataset that is expected to have only exact matches, the dupes result will be greater than the size of the dataset because it compares every field with the rest of the fields on the list.

General Ledger. Accounting use-case

Whether you're looking for a fuzzy matching percent or single client cleanup, Collibra DQ's duplicate detection can help you sort and rank the likelihood of duplicate data.



Double Booked Entries

Journal entries that are booked twice in the ledger

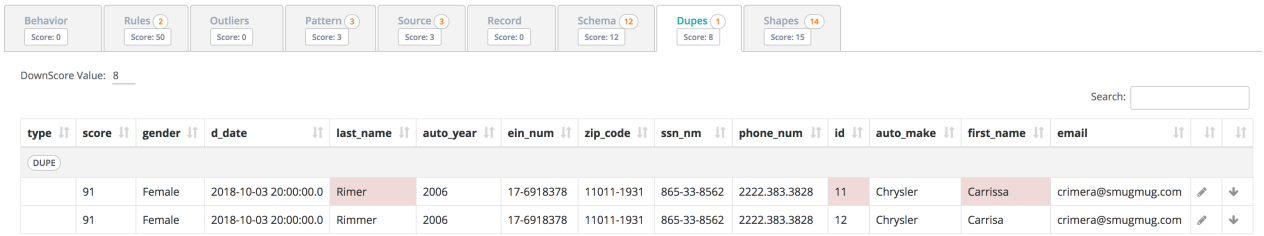
Owl's duplicate detection and fuzzy matching engine can track down hard to find nuanced bookings that are actually the same root booking.

JOURNAL_ID	ITEM	ACCOUNT	AMOUNT
JID-123	Swa flight-147	W2043	724.34
JID-629	Marriott	W2133	234.75
JID-456	Swa flight-147	W2043	724.34
JID-9022	Swa flight-822	W2043	1002.43

```
-f "file:///home/ec2-user/single_customer.csv" \
-d "," \
-ds customers \
-rd 2018-01-08 \
-dupe \
-dupenocase \
-depth 4
```

User Table has duplicate user entry

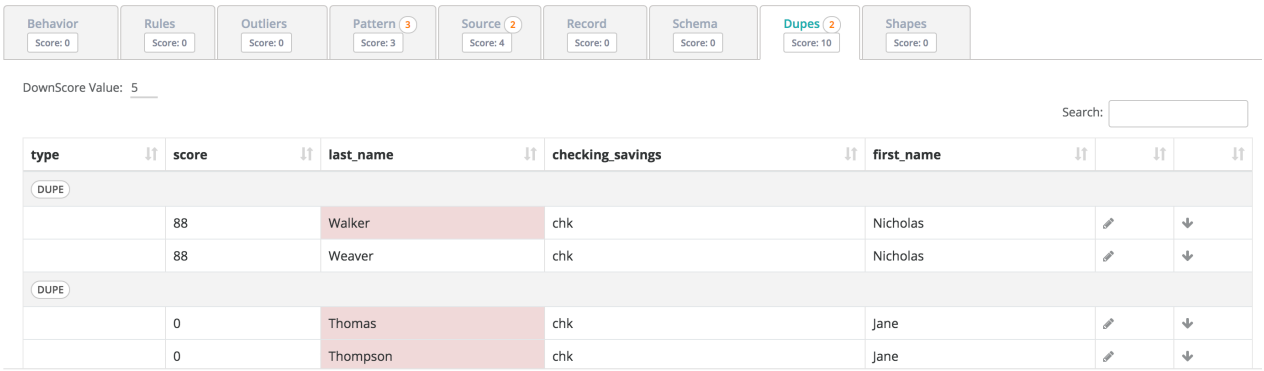
Carrisa Rimmer vs Carrissa Rimer



type	score	gender	d_date	last_name	auto_year	ein_num	zip_code	ssn_nm	phone_num	id	auto_make	first_name	email
DUPE													
	91	Female	2018-10-03 20:00:00.0	Rimer	2006	17-6918378	11011-1931	865-33-8562	2222.383.3828	11	Chrysler	Carrissa	crimera@smugmug.com
	91	Female	2018-10-03 20:00:00.0	Rimmer	2006	17-6918378	11011-1931	865-33-8562	2222.383.3828	12	Chrysler	Carrisa	crimera@smugmug.com

ATM customer data with only a 88% match

As you can see below, less than a 90% match in most cases is a false positive. Each dataset is a bit different, but in many cases you should tune your duplicates to roughly a 90+% match for interesting findings.



type	score	last_name	checking_savings	first_name
DUPE				
	88	Walker	chk	Nicholas
	88	Weaver	chk	Nicholas
DUPE				
	0	Thomas	chk	Jane
	0	Thompson	chk	Jane

Simple DataFrame Example

```
val l = Seq(
  ("Kirk", "Haslbeck", "2018-02-24 8:30:45", 18),
  ("Kirk", "Hasselbach", "2018-02-23 8:30:45", 11),
  ("Kirt", "Haslbeck", "2018-02-22 8:30:45", 10),
  ("Kirk", "hasselback", "2018-02-21 8:30:45", 12),
  ("kirk", "Haslbeck", "2018-02-20 8:30:45", 10)
)

val df = l.toDF("fname","lname", "app_date", "price")
val owl = new Owl(df).dupesDF
```

distance	records
0.16	DUPE [score: 83][Kirk,Hasselbach,2018-02-23 8:30:45] [Kirt,Haslbeck,2018-02-22 8:30:45]
0.083	DUPE [score: 91][Kirk,Haslbeck,2018-02-24 8:30:45] [Kirt,Haslbeck,2018-02-22 8:30:45]
0.083	DUPE [score: 91][Kirk,Haslbeck,2018-02-24 8:30:45] [Kirk,Hasselbach,2018-02-23 8:30:45]

Known Limitations

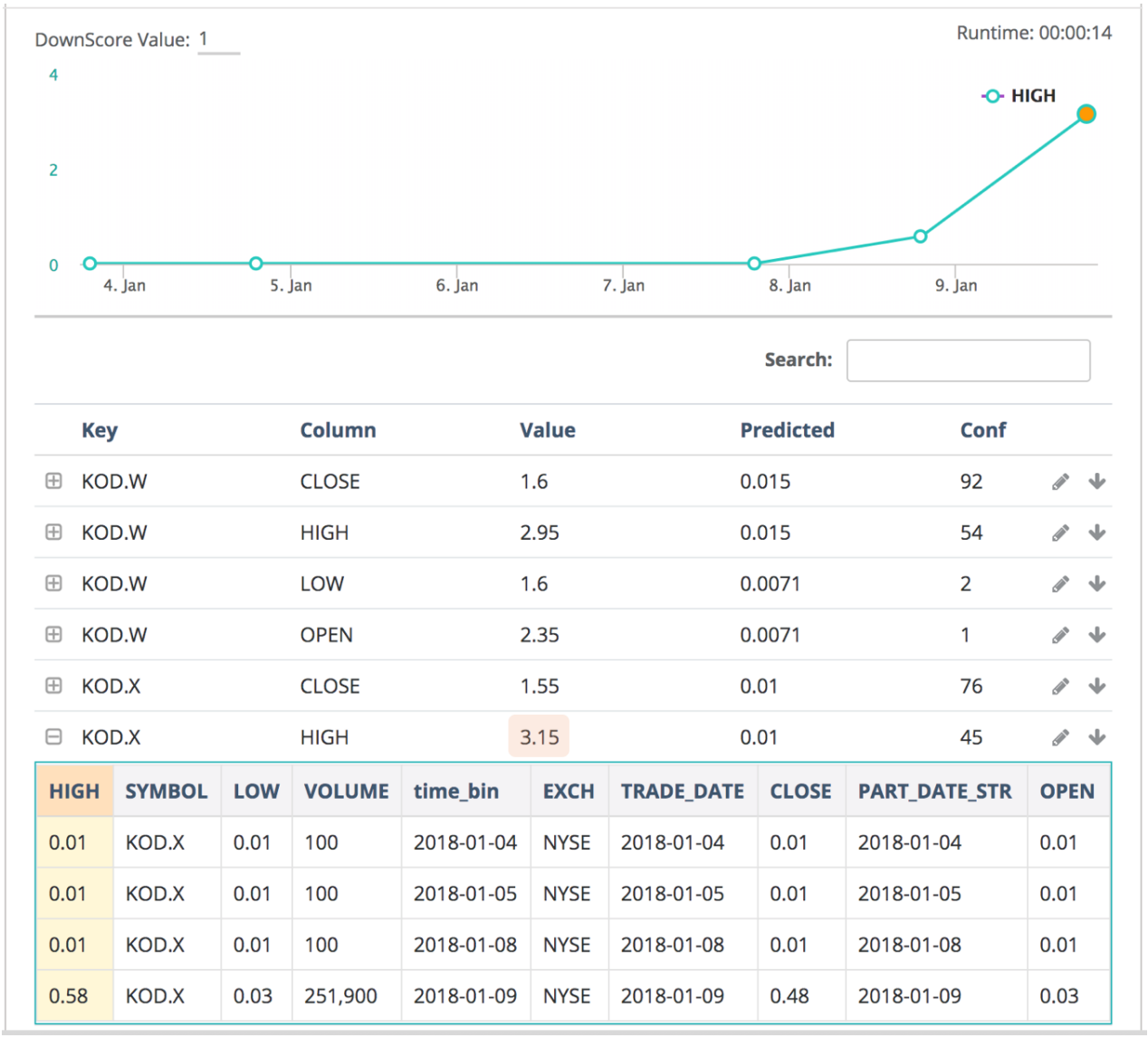
There is a known limitation where you cannot perform a dupes check on all columns or more than 10 columns at once.

Outliers (advanced)

> Note This is an advanced opt-in feature.

Numerical Outliers

Kodak Coin! In 2018 Kodak announced themselves as Kodak Coin and witnessed a steep change in their stock price. Collibra Data Quality automatically captured this event and provided the ability to drill into the item.

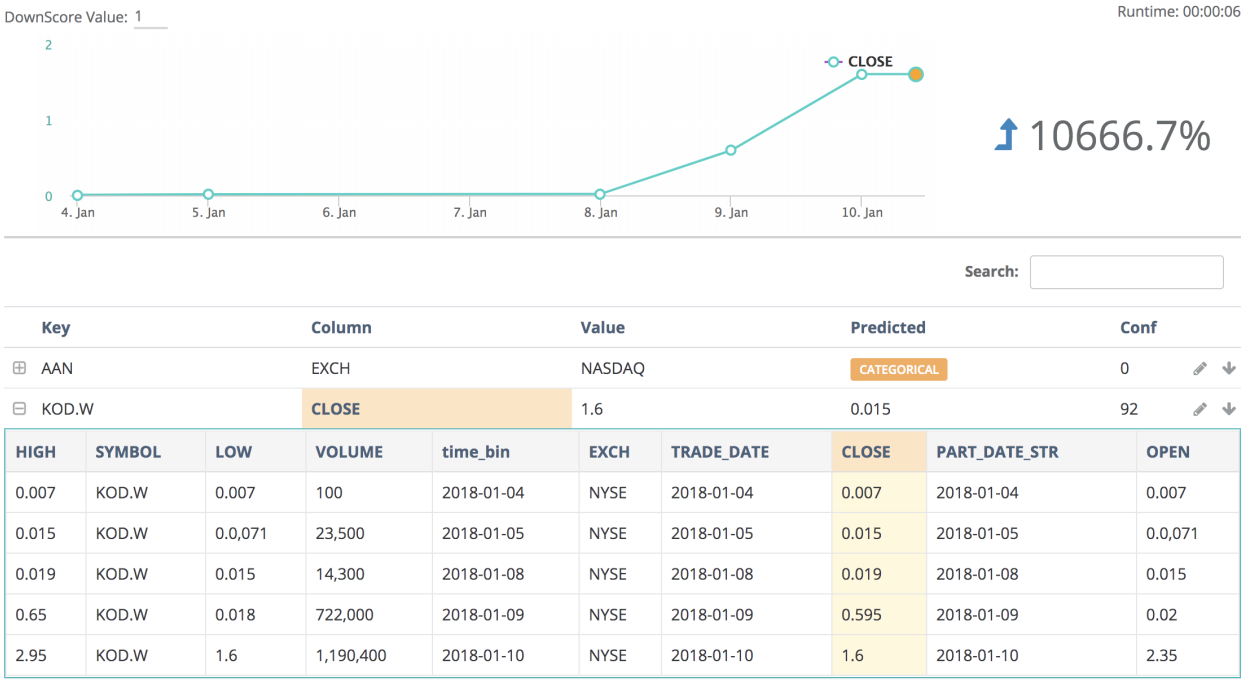


Key	Column	Value	Predicted	Conf
KOD.W	CLOSE	1.6	0.015	92
KOD.W	HIGH	2.95	0.015	54
KOD.W	LOW	1.6	0.0071	2
KOD.W	OPEN	2.35	0.0071	1
KOD.X	CLOSE	1.55	0.01	76
KOD.X	HIGH	3.15	0.01	45

HIGH	SYMBOL	LOW	VOLUME	time_bin	EXCH	TRADE_DATE	CLOSE	PART_DATE_STR	OPEN
0.01	KOD.X	0.01	100	2018-01-04	NYSE	2018-01-04	0.01	2018-01-04	0.01
0.01	KOD.X	0.01	100	2018-01-05	NYSE	2018-01-05	0.01	2018-01-05	0.01
0.01	KOD.X	0.01	100	2018-01-08	NYSE	2018-01-08	0.01	2018-01-08	0.01
0.58	KOD.X	0.03	251,900	2018-01-09	NYSE	2018-01-09	0.48	2018-01-09	0.03

Complex outliers made Simple

Even though DQ uses complex formulas to identify the correct outliers in a dataset, it uses simple terms when displaying them. If you notice below the change happened gradually, therefore if you only compared avgs or previous values you would not understand the full impact of this price change. 0% changed from yesterday and its moving/trailing avg would have caught up.



HIGH	SYMBOL	LOW	VOLUME	time_bin	EXCH	TRADE_DATE	CLOSE	PART_DATE_STR	OPEN
0.007	KOD.W	0.007	100	2018-01-04	NYSE	2018-01-04	0.007	2018-01-04	0.007
0.015	KOD.W	0.0,071	23,500	2018-01-05	NYSE	2018-01-05	0.015	2018-01-05	0.0,071
0.019	KOD.W	0.015	14,300	2018-01-08	NYSE	2018-01-08	0.019	2018-01-08	0.015
0.65	KOD.W	0.018	722,000	2018-01-09	NYSE	2018-01-09	0.595	2018-01-09	0.02
2.95	KOD.W	1.6	1,190,400	2018-01-10	NYSE	2018-01-10	1.6	2018-01-10	2.35

Dynamic history options

Data may not always enter your data pipeline on time and as expected due to weekend, holidays, errors, etc. To help capture outliers in spite of gaps, there are two main options:

- 1) Extend the lookback period (to 10 days from 5 days, for example)
- 2) Utilize additional flags per below (fllbminrow new as of 2021.11)

Flag	Description	Example
fllbminrow	File Lookback Minimum Rows: determines minimum number of rows that a previous file scan needs to be counted as file lookback	-fllbminrow 1 (counts nay DQ scans with 1 or more row in minimum history) -fllbminrow 0 (default behavior, row count does not matter)
dllb	Date Lookback: determines how many days of learning	-dllb 5 (5 days)

Upper & Lower Bound Limits

Collibra DQ automatically detects and flags data that falls outside of preset **Upper Bound** or **Lower Bound** limits. If data is detected outside of these limits, an alert is generated to notify of an outlier. Setting these limits allows you to finetune your ability to identify outliers.

Categorical Outliers

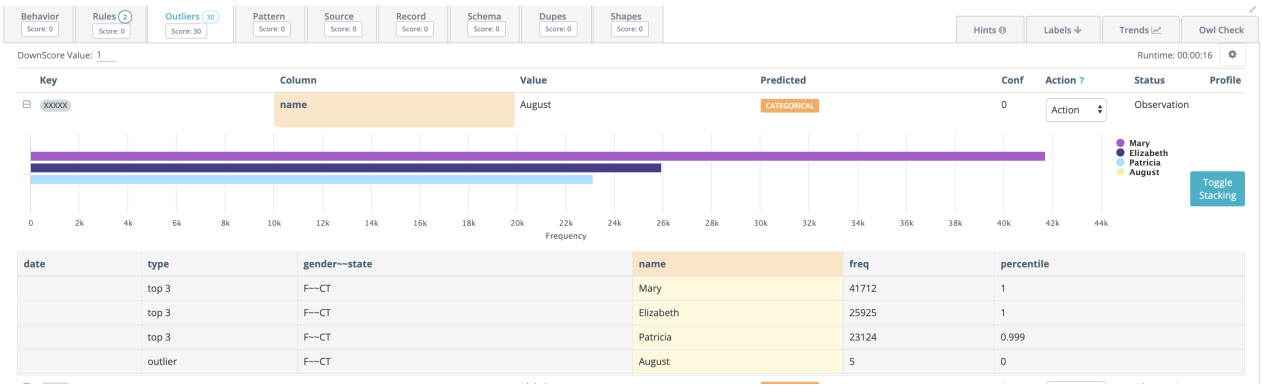
Categorical Outliers are much different than numerical outliers and require separate techniques to automatically capture meaningful anomalies. The details regarding Owl's methodology and testing can be found below, 3 minute read on the topic.

Categorical Outliers Don't Exist

DQ will automatically learn the normal behavior of your String and Categorical attributes such as STOCK,OPTION,FUTURE or state codes such as MD,NC,D.C. When a strange pattern occurs (e.g NYC instead of NY), DQ will show this as a categorical outlier.

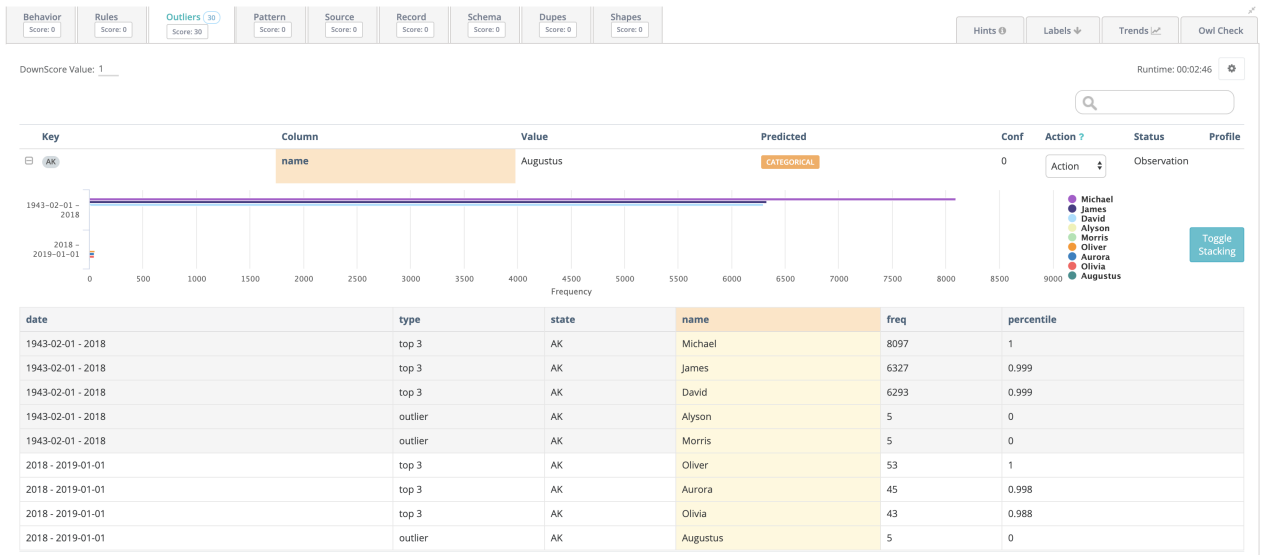
DQ is able to detect Categorical Outliers both with and without taking time into account. If a time dimension is not provided, DQ will calculate the distribution of categorical values

within the available data, and identify the values that fall into the most infrequent percentile (configurable).



date	type	gender~~state	name	freq	percentile
	top 3	F~~CT	Mary	41712	1
	top 3	F~~CT	Elizabeth	25925	1
	top 3	F~~CT	Patricia	23124	0.999
	outlier	F~~CT	August	5	0

If a time dimension is provided, DQ will first identify infrequent categories in the historical context and then in the context of the current Owlcheck. Only values that are historically infrequent or non-existent, and are infrequent in the current run will be considered Outliers.



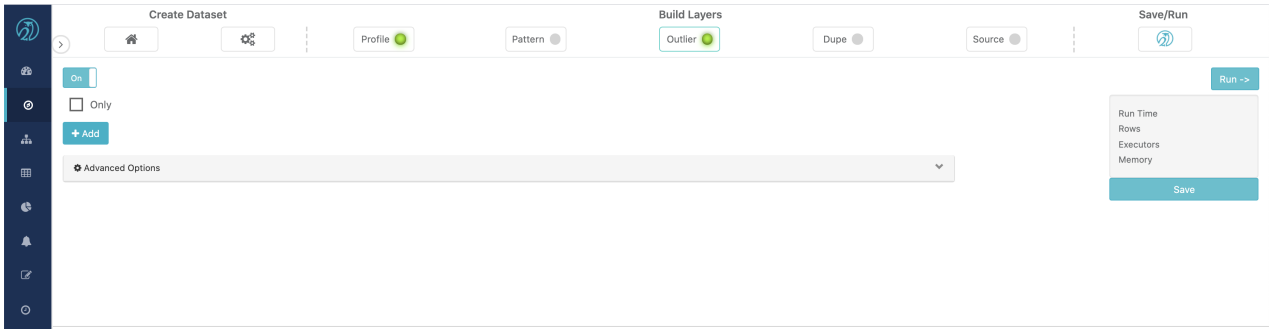
date	type	state	name	freq	percentile
1943-02-01 - 2018	top 3	AK	Michael	8097	1
1943-02-01 - 2018	top 3	AK	James	6327	0.999
1943-02-01 - 2018	top 3	AK	David	6293	0.999
1943-02-01 - 2018	outlier	AK	Alyson	5	0
1943-02-01 - 2018	outlier	AK	Morris	5	0
2018 - 2019-01-01	top 3	AK	Oliver	53	1
2018 - 2019-01-01	top 3	AK	Aurora	45	0.998
2018 - 2019-01-01	top 3	AK	Olivia	43	0.988
2018 - 2019-01-01	outlier	AK	Augustus	5	0

Training Outlier Detection Model

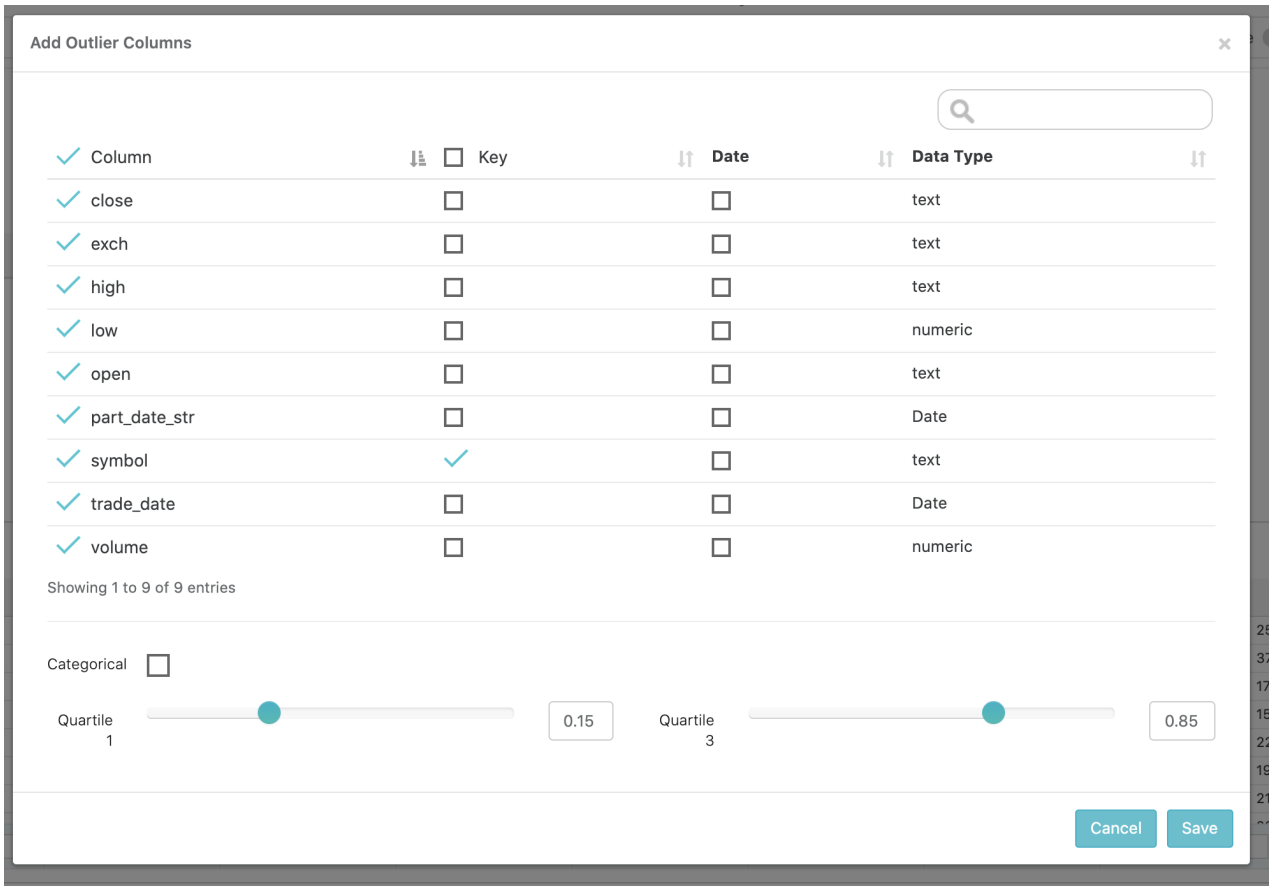
Although DQ uses different techniques to detect Numerical and Categorical Outliers, the training process is very similar.

At a minimum, DQ requires historical data that can be used as the training dataset. If no other input is provided, DQ will calculate the normal range for each selected column and look for numerical and categorical outliers within the training dataset without any further

context. The output will essentially consist of infrequent values that fall outside the normal range fo each column.

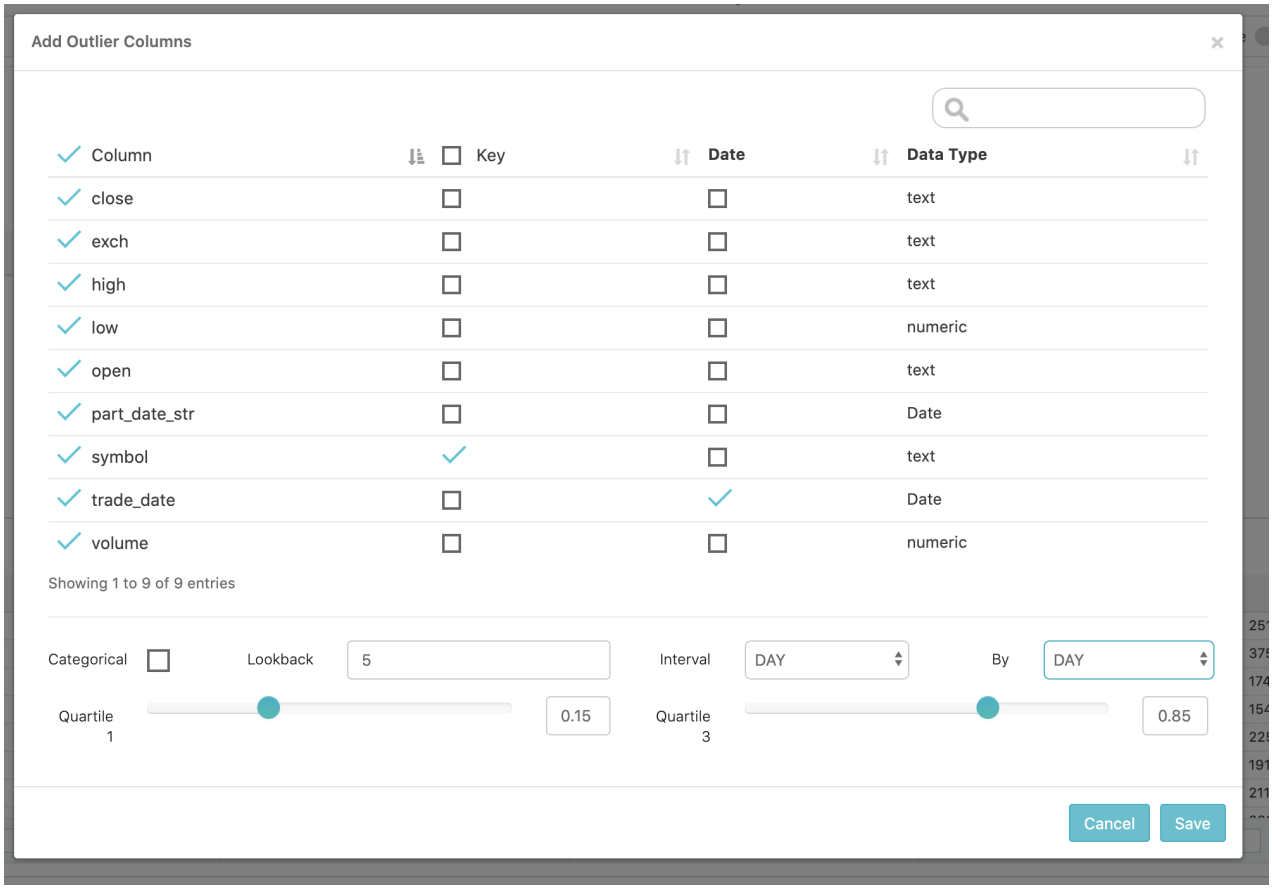


To obtain more targeted results, the DQ requires a "key" column. This column will be used to provide context by grouping each column by the key column. Defining a good key column tends to provide results that are a better indicators of actual data quality issues instead of simply infrequent values.

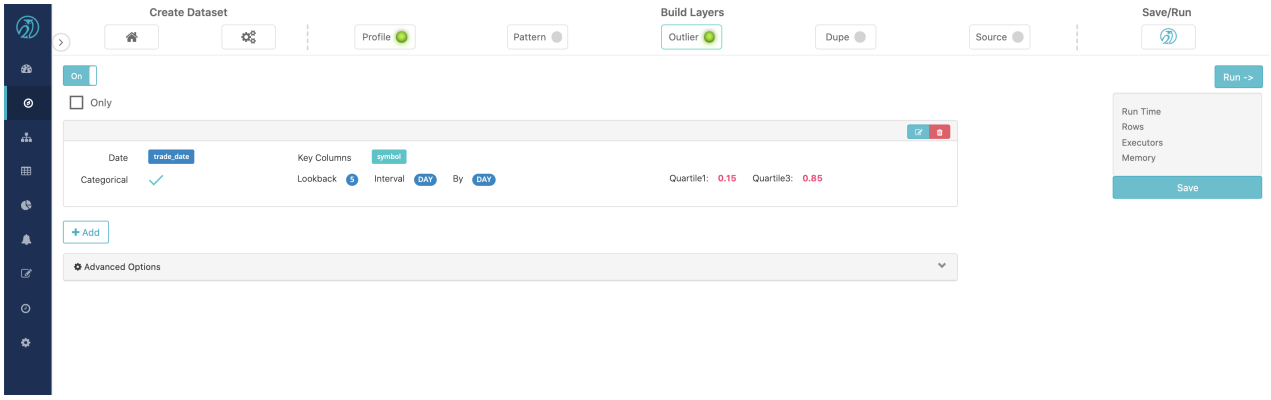


Another input that can make outlier detection more precise is a data/time column and a look back period. This enables a more precise calculation of the normal range for a column

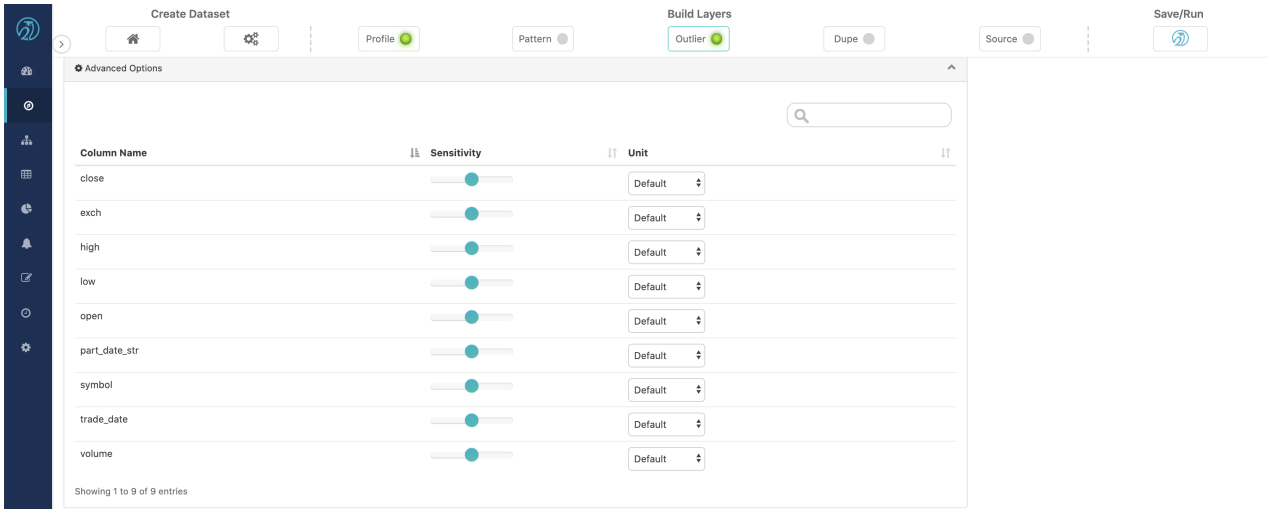
and in the case of numerical outliers, makes it possible for DQ to establish a trend. Given a time column and key column, DQ will not only identify numerical outliers, it will plot the historical trend of the column value trailing the outlier.



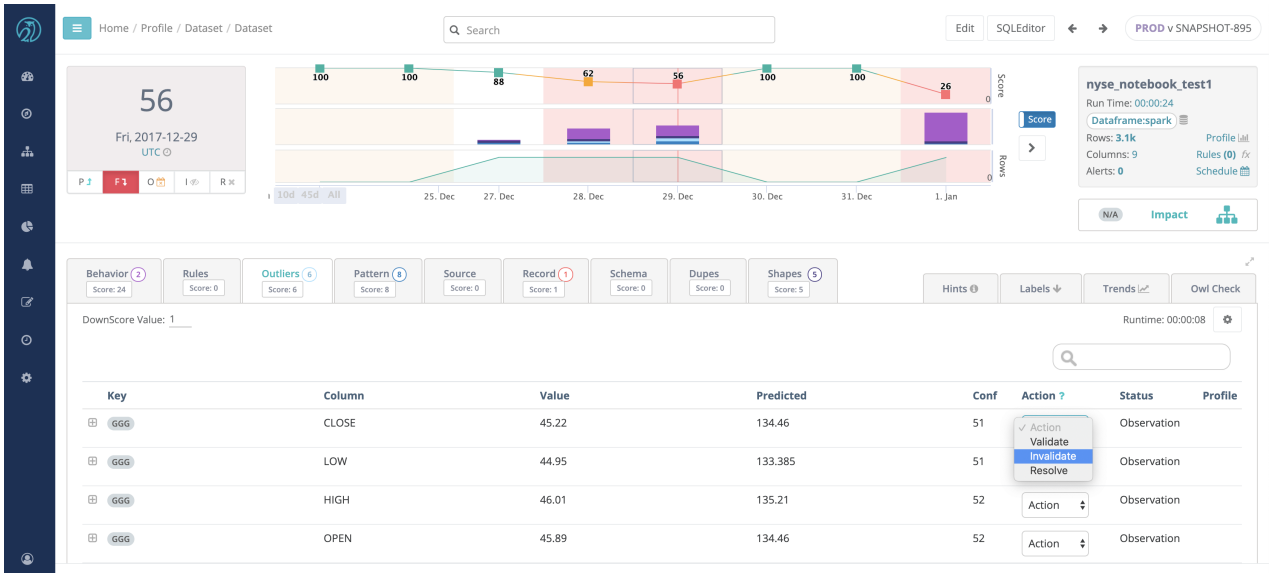
DQ also allows further refinement of the time dimension by defining time bins and processing intervals. By default, when given a time column, DQ will bin the data into days and process the data in daily interval. However, if the data is high frequency, day bins and day intervals might be too coarse grained. In this case, it might make more sense to group the data into bins on the minute and process the data in hour or minute intervals. The same concept applies in the other direction. What if the data is already aggregated on the month or year? In this case, it makes more sense to set the bins and intervals to month by month or month by year.



Some data may be measured in really small or large units or contain a lot of noise. In this case, DQ allows the user to adjust the sensitivity level and unit of measure for outlier detection on each column. Click the advanced tab to make these adjustments.



Once Outlier detection is complete for a given run, it's time to tune the scoring of the model. DQ allows the user to label any outlier findings as legitimate, thus preventing that outlier from being detected in the future or effecting the score of the current run. In addition, it is possible to define the significance of an outlier finding to a given dataset. This can be accomplished by setting how many quality points should be deducted for each outlier finding on any given run on that dataset. It is also possibly to adjust sensitivity and unit of measure of future runs by clicking on the small gear icon on the far left of the screen.



Spark DataFrame Example

Categorical Outliers

```
val assetClass =
Seq("STOCK","OPTION","FUTURE","STOCK","OPTION","FUTURE","STOCK","STOCK", "FUTURE",
    "STOCK","OPTION","FUTURE","STOCK","OPTION","FUTURE","STOCK","STOCK", "FUTURE",
    "STOCK","OPTION","FUTURE","STOCK","OPTION","FUTURE","STOCK","STOCK", "FUTURE",
    "STOCK","OPTION","FUTURE","STOCK","OPTION","FUTURE","STOCK","STOCK", "FUTURE",
    "STOCK","OPTION","FUTURE","STOCK","OPTION","FUTURE","STOCK","STOCK", "FUTURE",
    "STOCK","OPTION","FUTURE","STOCK","OPTION","FUTURE","STOCK","STOCK", "FUTURE",
    "SWAP", "OPTION", "STOCK","SWAP", "SWAP", "FUTRS", "STOCK", "OPTION","FUTURE",
    "OPTION", "STOCK" )

val df = assetClass.toDF("ASSET")
val outliers = new Owl(df).outliersDF("ASSET")
```

Key	Column	Value	Prediction	Confidence
	ASSET	FUTRS	FUTURES	

Real World Example

Imagine you are the data manager at Iowa Department of Commerce, Alcoholic Beverage Division. As part of the Department's open data initiative, the monthly Iowa liquor sales data are available to the public for analysis. (Thank you Iowa!)

An Iowan data analyst emails you about a data quality issue with **address** for store #2508 in the year 2016. You quickly run a SQL query on your data warehouse to see what is going on.

```
-- Assuming Postgres DB
select date_trunc('MONTH', "date") "date_month", address, count
(*) "sales_count"
from iowa_liquor_sales
where "date" >= '2016-01-01' and "date" < '2017-01-01' and
store_number = '2508'
group by date_trunc('MONTH', "date"), address
order by date_month, address
```

date_month	address	sales_count
2016-01-01 00:00:00	1843 JOHNSON AVENUE, N.W.	422
2016-02-01 00:00:00	1843 JOHNSON AVENUE, N.W.	451
2016-03-01 00:00:00	1843 JOHNSON AVENUE, N.W.	579
2016-04-01 00:00:00	1843 JOHNSON AVENUE, N.W.	404
2016-05-01 00:00:00	1843 Johnson Avenue, N.W.	625
2016-06-01 00:00:00	1843 Johnson Avenue, N.W.	695
2016-07-01 00:00:00	1843 Johnson Avenue, N.W.	457
2016-08-01 00:00:00	1843 Johnson Avenue, N.W.	744
2016-09-01 00:00:00	1843 Johnson Avenue, N.W.	681
2016-10-01 00:00:00	1843 Johnson Avenue, N.W.	728
2016-11-01 00:00:00	1843 Johnson Avenue, N.W.	1062
2016-12-01 00:00:00	1843 Johnson Avenue, N.W.	992

Because `store_number` is an unique number assigned to the store who ordered the liquor, the inconsistent `address` values for the same store pose data quality problem. But `address` is a string value that can take many forms. For store #2508, the reported address

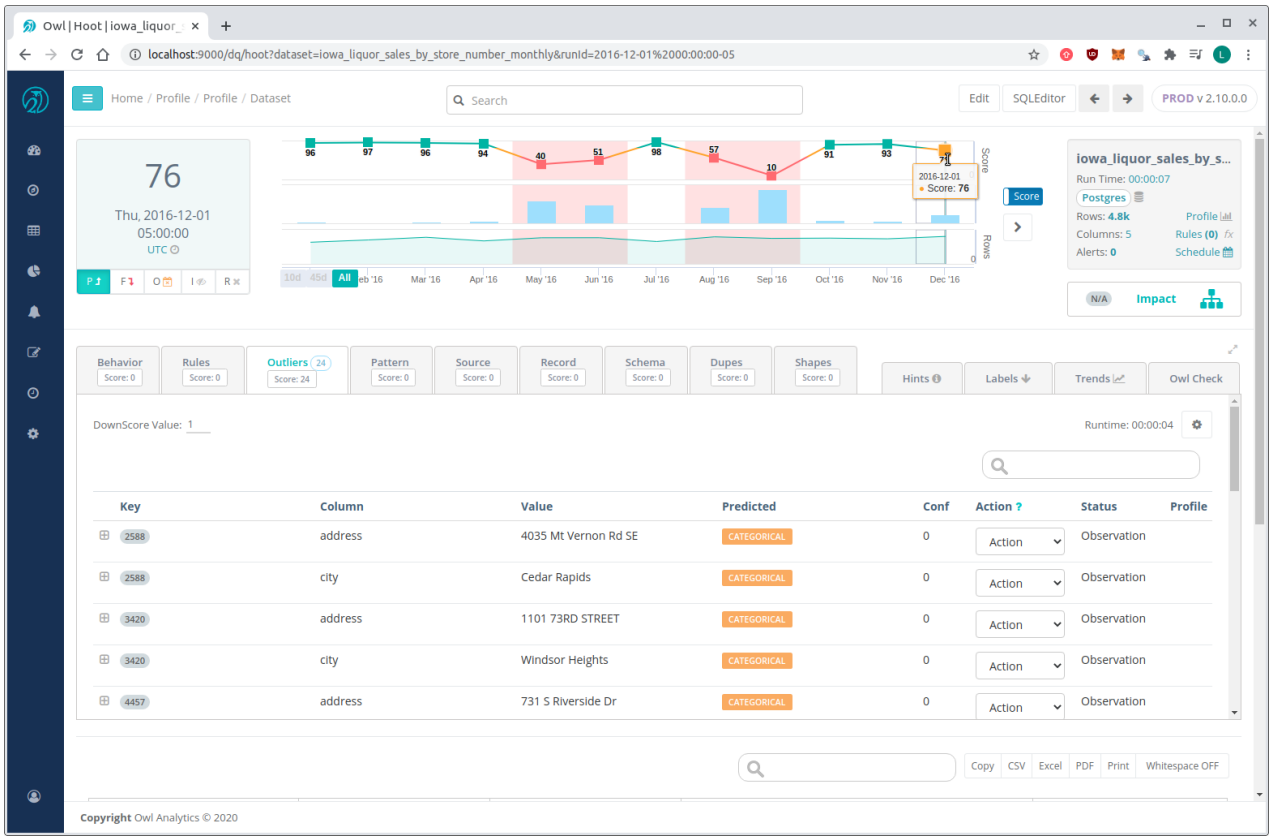
value has a shifted behavior from all capital letters starting on May 2016. For other cases, it might be completely different behavior change that you would have to manually check one by one. With over 2,000 unique stores, 19 million rows, and 8 years of data, you need an automated way to detect meaningful categorical outliers.

The following command shows an example of running monthly OwlDQ Checks, from the month of Jan 2016 to the month of December 2016. Each monthly run looks back 3 months of data to establish a baseline for categorical columns that you suspect would have similar data quality issues: `store_name`, `address`, and`city`.

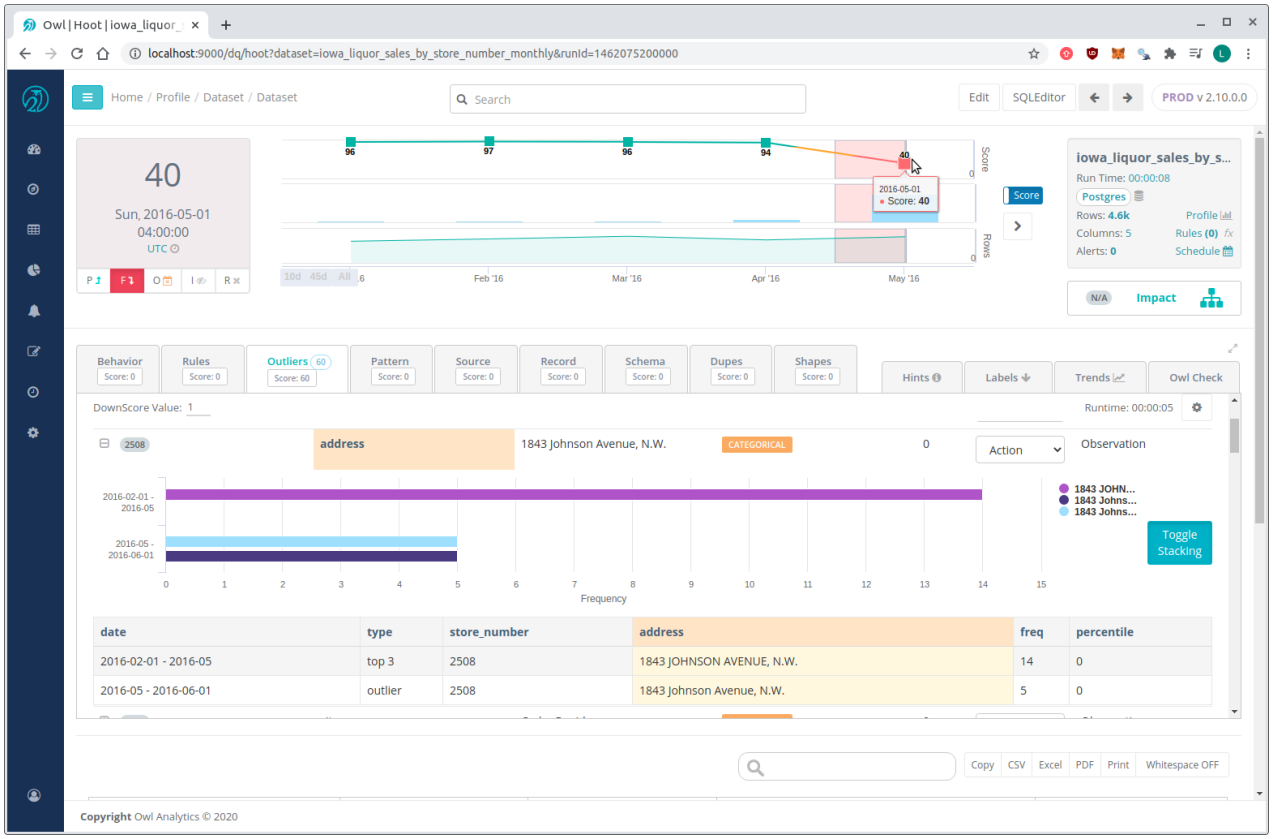
```
/opt/owl/bin/owlcheck
  # connection information to data
  -lib "/opt/owl/drivers/postgres/" -driver "org.-
postgresql.Driver"
  -c, "jdbc:postgresql://localhost:5432/postgres"
  -u, "postgres", "-p", "password"
  # Specify dataset name
  -ds "iowa_liquor_sales_by_store_number_monthly"
  # Specify date filter for the last filter, e.g. date >= '2016-
12-01' and date < '2017-01-01'
  -rd "2016-12-01" -rdEnd "2017-01-01"
  # SQL query template (${rd} and ${rdEnd} matches with -rd and
-rdEnd
  -q "select distinct on (date, store_number) date, store_num-
ber, store_name, address, city
       from iowa_liquor_sales where date >= '${rd}' and date <
'${rdEnd}' "
  # Turn on Outliers
  -dl
  # Group on store_number (optional if no grouping)
  -dlkey "store_number"
  # Specify column that is of date type (optional, if running
OwlCheck without time context)
  -dc "date"
  # Specify columns to run Outlier analysis (if not specified,
all the columns in query are included in analysis)
  -dlinc "store_name,address,city"
  # Specify 3 month lookback for each OwlCheck
  -dllb 3
  # Run Monthly OwlCheck
  -tbin "MONTH"
  # "backrun" Convenient way to run 12 preceding MONTHly owl
check
  -br 12
```

Results

The `-br 12` option ran 12 monthly OwlChecks for every month of 2016. The figure below shows OwlCheck Hoot page for the lastest run of dataset `iowa_liquor_sales_by_store_numbers_monthly`. The Hoot page shows that OwlCheck identified 24 Outliers among 4.8k rows of unique date x store_number for month of December, 2016.



Since the original data quality issue that inspired us to run OwlCheck is from May 2016, we can navigate to specific run date 2016-05-01 by click on the line graph on top. Then searching for store #2508 on the **key** column shows outlier detected for **column** `address`. Press [+] for that row to see contextual details about this detected value.



We can verify that OwlCheck identified the outlier of interest among other 60 data quality issues. Using OwlCheck, you can identify issues at scale for past data (using backrun), current (using simple OwlCheck), and future (using scheduled jobs).

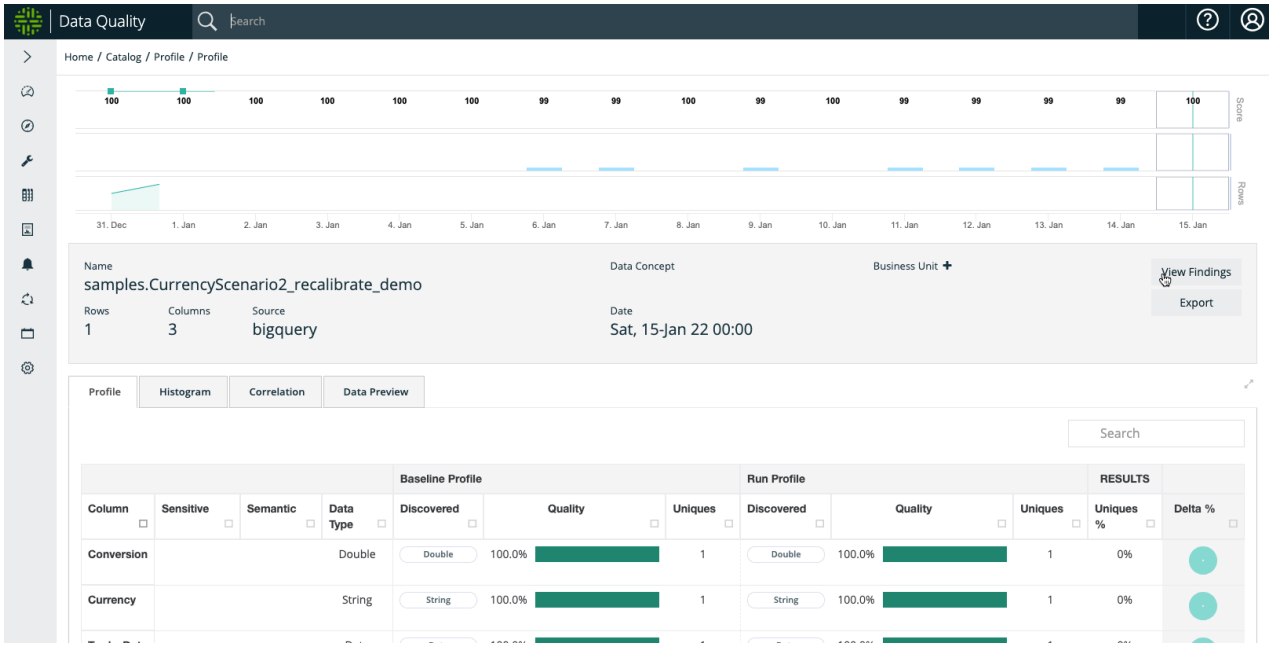
Tech Preview [TP] Outlier Calibration

Use Case

When A) using Outliers and B) faced with an event such as a stock split or currency devaluation, it may be helpful to calibrate the outlier boundaries within Collibra DQ to avoid surfacing non-issues for a period of time.

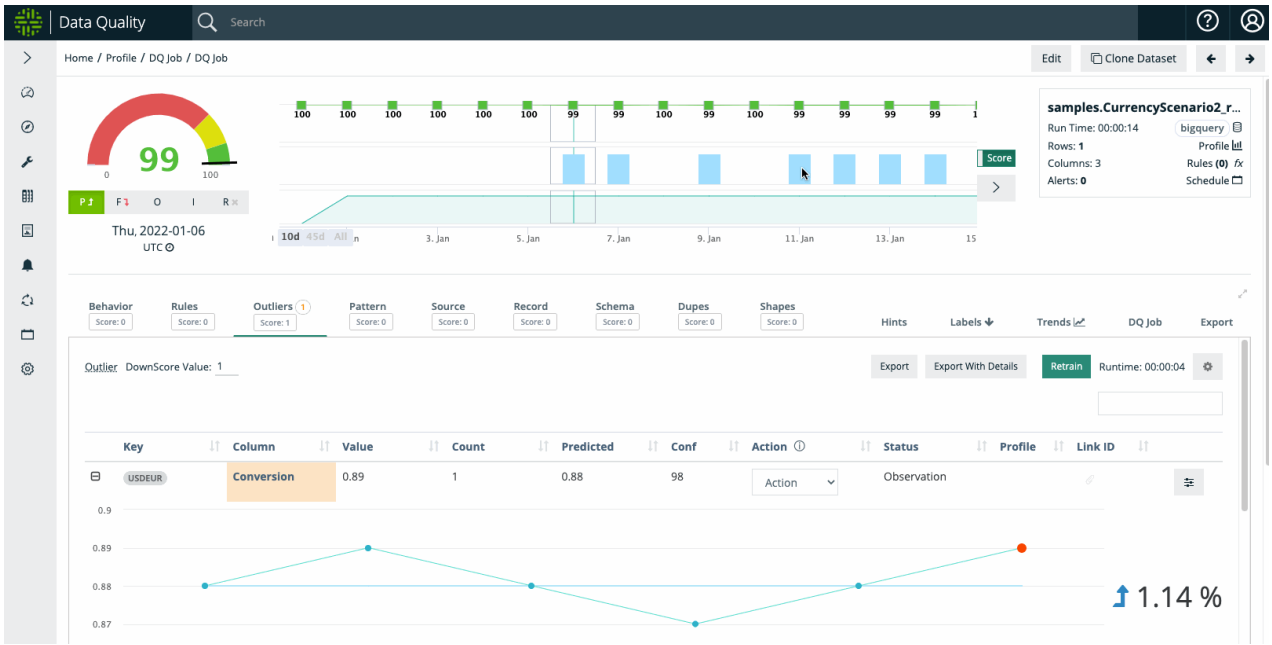
Example Step #1: No Action Necessary

In the video below, Collibra DQ Outliers were set to a high sensitivity. The USDEUR conversion rate on January 6th in the sample dataset may be considered reasonable and the user can 1) rerun the dataset with lower sensitivity or 2) downtrain the unintended Outlier anomalies.



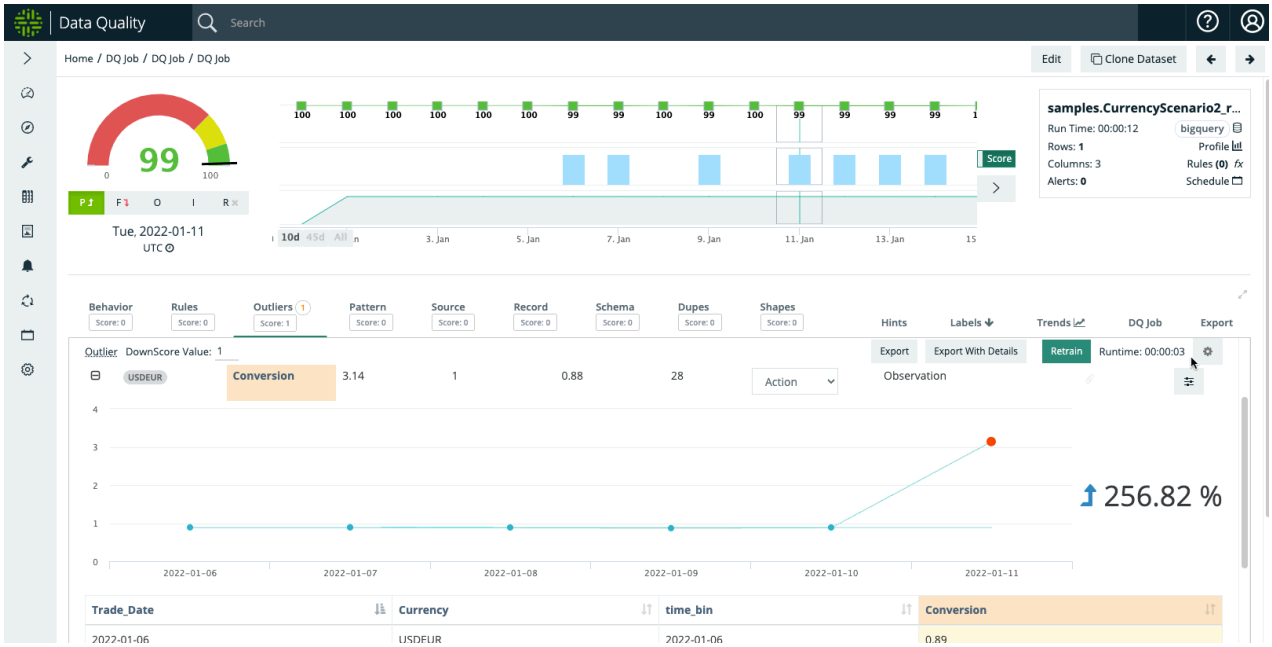
Example Step #2: Macro Event That User Understands e.g. Currency Devaluation or Stock Split

When examining the outlier on January 11th, the dataset depicts that the USDEUR conversion shot up to 3.14, which in our hypothetical example coincides with an explainable macroeconomic phenomenon. As such, the user may not want Outlier anomalies to trigger for a period of time.



Example Step #3: User Wants To Suppress Outliers

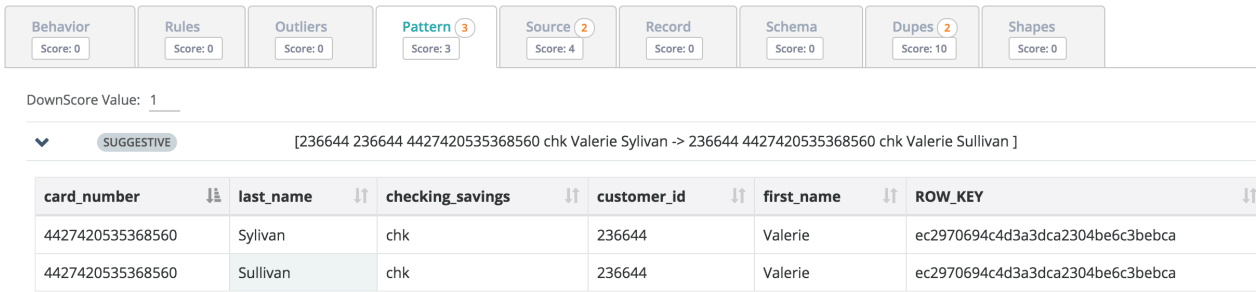
Once Outlier Calibration is enabled, a user can select the boundaries and duration of the 'suppression' period. And once the DQ Job is re-run for the selected date(s), the outliers will not trigger an anomaly / downscore.



Patterns (advanced)

> Note This is an advanced opt-in feature.

Owl uses the latest advancements in data science and ML to find deep patterns across millions of rows and columns. In the example below it noticed that Valerie is likely the same user as she has the same customer_id and card_number but recently showed up with a different last name. Possible misspelling or data quality issue?

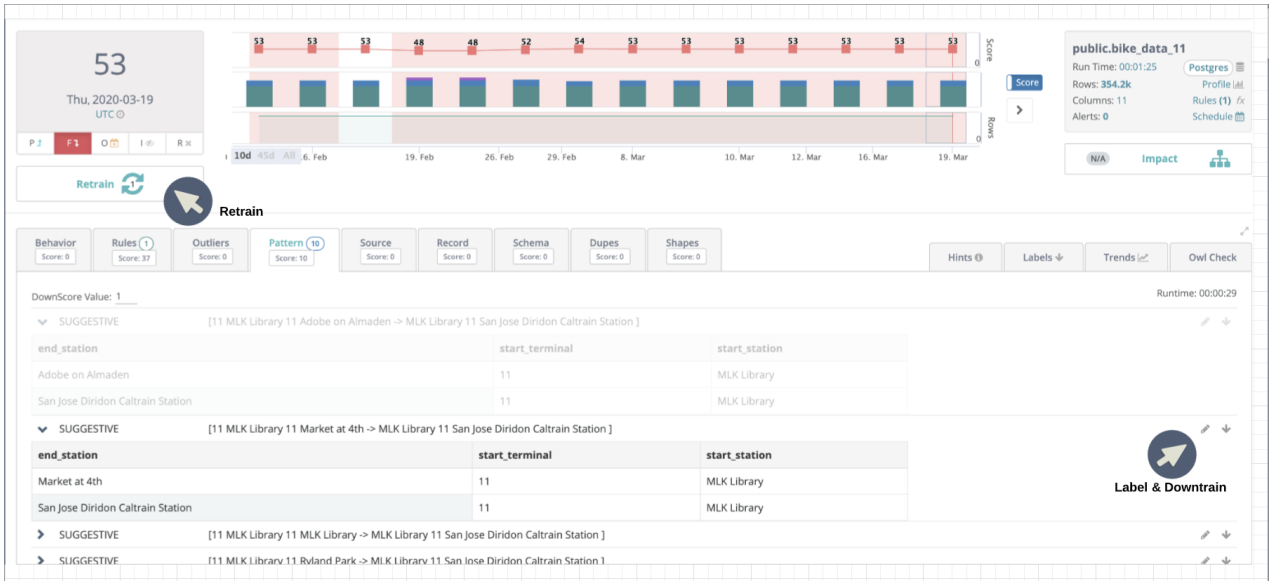


Training Anti-Pattern Detection Model

When the Patterns feature is enabled, DQ builds a collection of patterns that it identifies within the data. It will then use that collection to identify values that break established patterns. For example, in the image below, DQ learned that a bike route that starts at "MLK library" will end at "San Jose Diridon Caltrain Station". However, when the current day's data cross referenced against this pattern, DQ detects an anti-pattern where a trip starts at "MLK Library" but ends at "Market at 4th". DQ raises this anti-pattern as a data quality issue and highlights the what it believes the "end_station" value should have been.

To build a Pattern model, DQ requires historical data that contains the valid patterns and if possible, a date/time column. The user can then needs to define the date/time column, the look back period, and what columns make up the pattern. In the image below, the pattern was composed of "end_station", "start_terminal", "start_station".

It is possible that an apparent anti-pattern finding is actually valid data and not a data quality issue. In this case, DQ allows the user to further instruct the existing Patterns model on how to properly score and handle the findings. For example, if it turns out that "Market at 4th" is actually a valid "end_station" for a bike trip, the user can negate the identified anti-pattern by labeling it as valid. This action instructs DQ to not raise this particular anti-pattern again. DQ also rescores the current Owlcheck results to reflect the user's feedback. In addition, it is possible to define the weight of an anti-pattern finding on the current dataset by setting the numerical value to deduct per finding.



Fraud Detection?

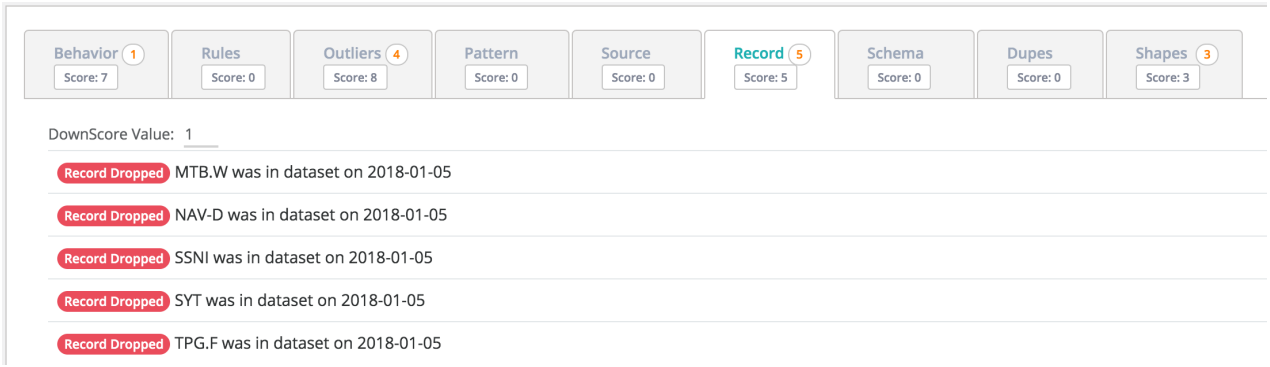
Think about a scenario where a dataset has a SSN column along with FNAME, LNAME and many others. What if your traditional rules engine passes because one of the rows has a valid SSN and a valid Name but the SSN doesn't belong to that person (his or her name and address, etc.)? This is where data mining can derive more sophisticated insights than a rules based approach.

Records (advanced)

> Note This is an advanced opt-in feature.

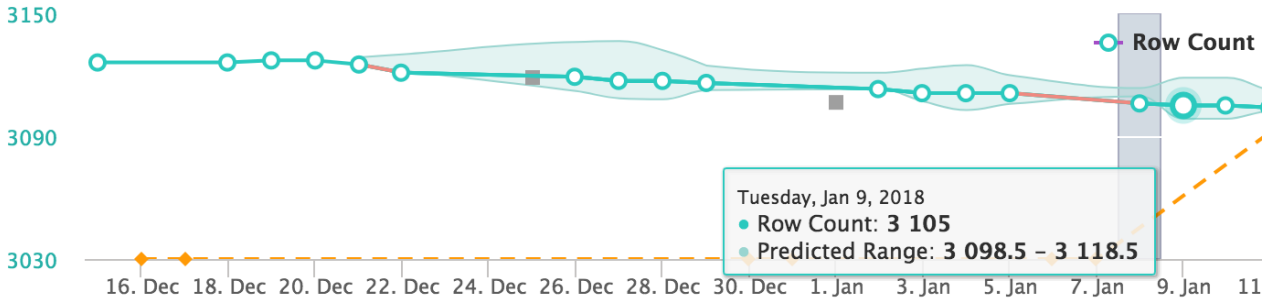
Where did my rows go?

Collibra DQ is constantly learning which records or rows in a dataset are most common. In the case below the NYSE had a reasonable dataset volume (row count).



Row Count Trend

We can see the rows dipping just slightly outside their predicted range. Arguably a subtle drop, yet abnormal to not represent these companies that typically do trade on the NYSE. Were they de-listed?



Source (advanced)

> Note This is an advanced opt-in feature.

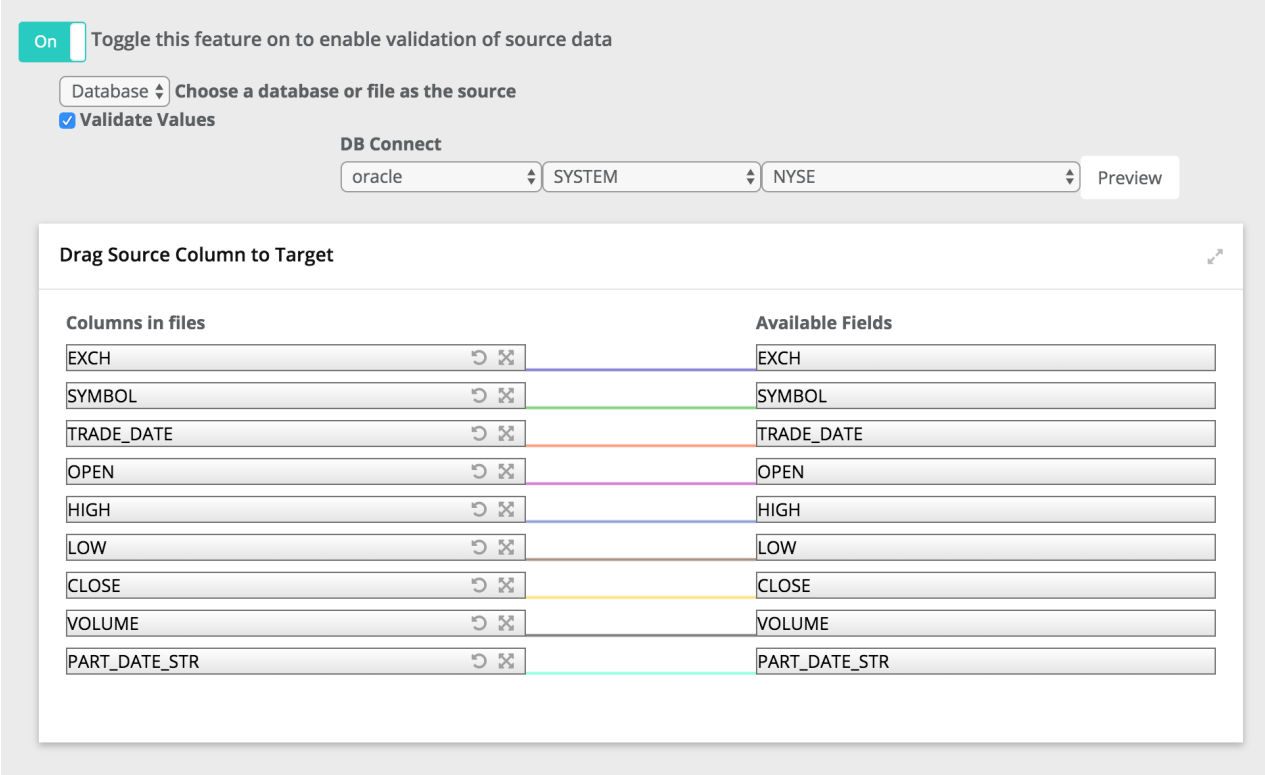
Does your data-lake reconcile with your upstream system?

Copying data from one system to another is probably the most common data activity to all organizations. Collibra DQ refers to this as source to target. As simple as this activity sounds, DQ has found that most of the time files and database tables are not being copied properly. To ensure and protect against target systems getting out of sync or not matching the originating source, turn on `-vs` to validate that the source matches the target.

A row count is not enough...

The most common check we encounter is a row count. However, a row count does not account for:

- Schema differences - Boolean to Int, Decimal to Double with precision loss, Timestamps and Dates.
- Value differences - Char or Varchars with whitespace vs Strings, null chars, delimiter fields that cause shifting, and much more.



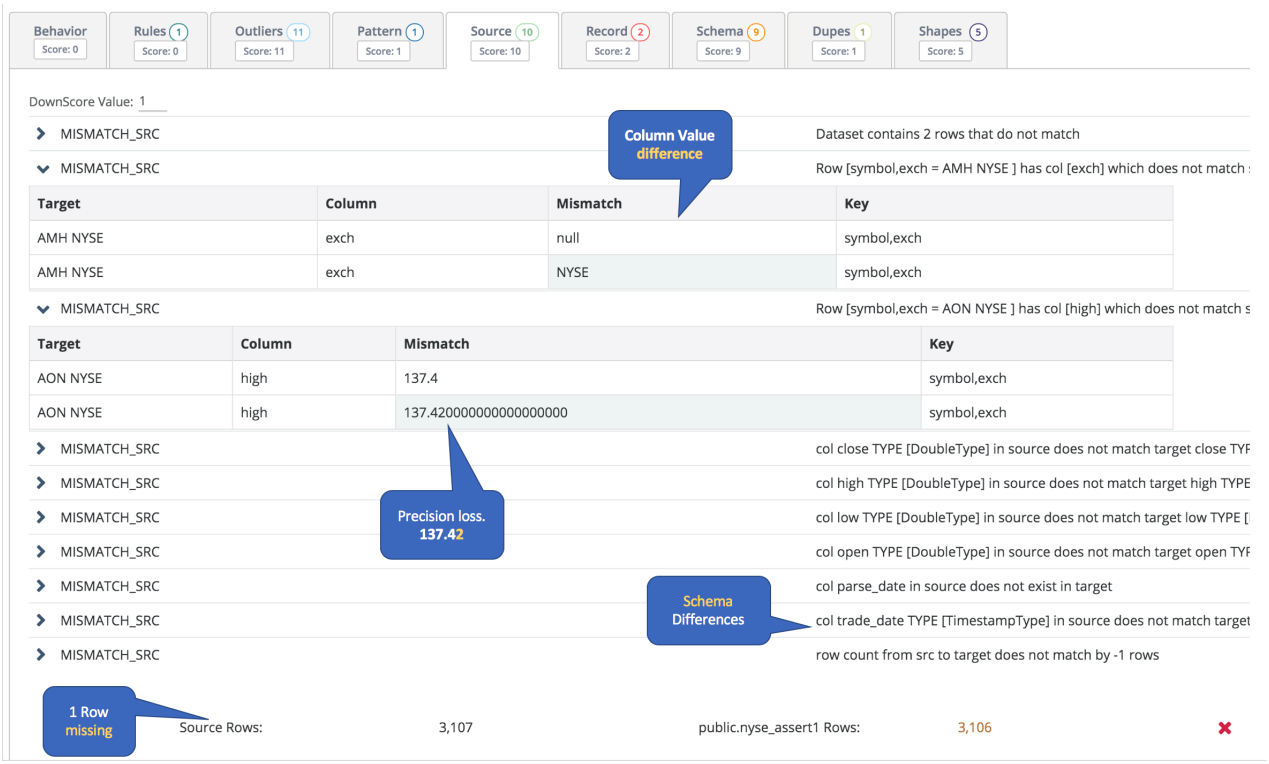
DQCheck Created from Wizard

The DQ Wizard GUI creates the below OwlCheck which it can execute from the GUI by clicking RUN or by pasting at the cmdline.

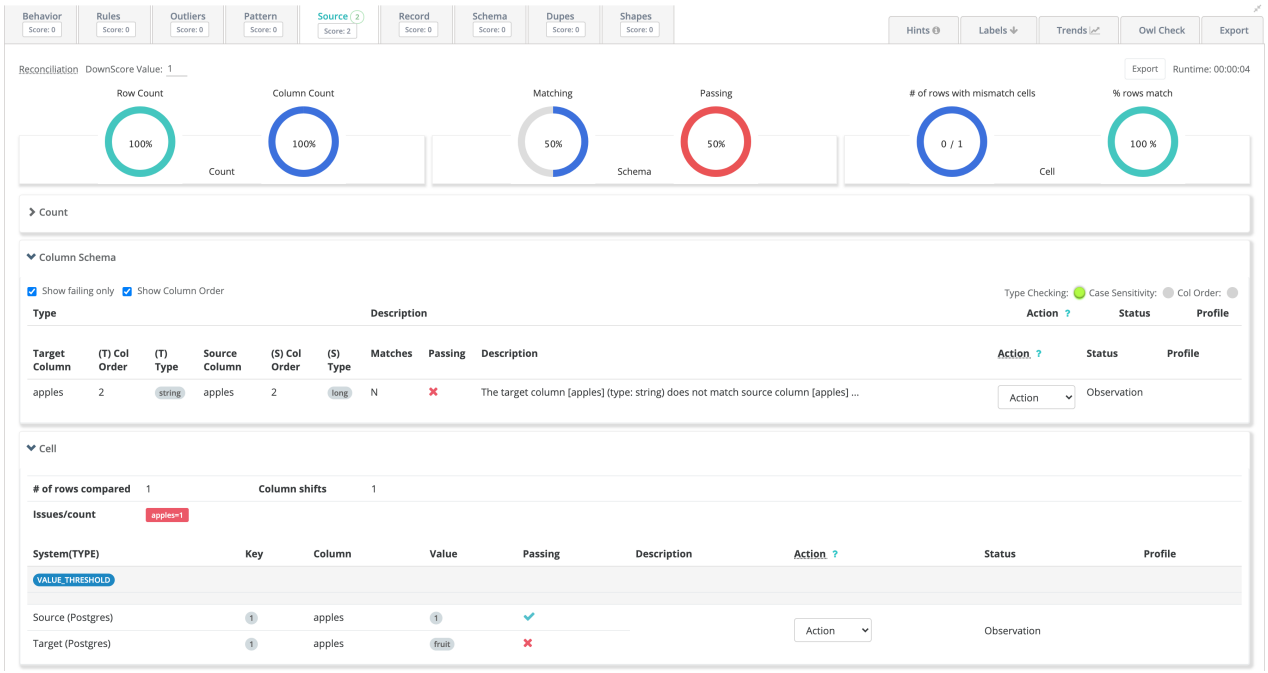
```
-lib /home/ec2-user/owl/drivers/valdrivers \
-driver org.postgresql.Driver \
-u user -p password \
-c "jdbc:postgresql://ec2-34-227-151-67.compute-1.amazon-
aws.com:5432/postgres" \
-q "select * from public.dateseries4" \
-ds psql_dateseries2 -rd 2018-11-07 \
-srcq select dz, sym as symz, high as highz, low as lowz, close
as closez, volume as volumez, changed as changedz, changep as
changepz, adjclose as adjclosez, open as openz from lake.d-
ateseries \
-srcu user \
-srcp password \
-srcds mysqlSYMZ \
-srcd com.mysql.cj.jdbc.Driver \
-srcc "jdbc:mysql://owldatalake.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com:3306/lake" \ -valsrckey "SYMZ" \
-vs \
-valsrcinc "dz,symz,open-
z,highz,lowz,closez,volumez,changedz,changepz,adjclosez"
```

End of Day Stock Data from Oracle to Mysql

In this example we loaded NYSE_EOD data in both Oracle and Mysql and then used Collibra DQ's Source Validation feature. We see 3 main classes of issues. 1) The row count is off by 1 row, this means a row was dropped or went missing when the data was copied. 2) The schemas don't exactly match. 3) In 2 cases the values are different at the cell level. NULL vs NYSE and 137.4 vs 137.42.



Latest View in 2.13+



Pushdown

Pushdown feature description

Some of the Collibra DQ features support pushdown to avoid transferring large dataset from data source (Database, Cloud storage, file systems etc.) into Spark. When pushdown is enabled and supported, the DQ Job will generate SQL queries to offload the compute to the data source, reducing the amount of data transfer and Spark computation of the DQ Job. Not all features support pushdown nor pushdown completely eliminate data transfer.

Why use pushdown

To analyze a dataset for DQ findings as part of a DQ Job, CDQ uses Spark as the compute engine to power the analysis. This requires the dataset to be loaded into Spark as a Spark DataFrame, and the DQ Job performance (job completion speed) is limited by Spark resources available and the complexity of the job. This data transfer from data source to Spark is dependent on two of the following factors:

1. Bandwidth limitation at the data source
   If the DQ Job requires pulling 100 million rows from a SQL Database, then the Input/Output limit of transferring data out of SQL Database into the Spark cluster will greatly increase the overall speed of the DQ Job.

2. Compute limitation at the data source for complex queries
   If the DQ Job requires complex queries to create the dataset (via `-q`), then this computation is done at the SQL Database level. Some examples of "complex" queries are:
   ```
   -q "SELECT * FROM public.very_long_table_of_transactions WHERE date = ${rd} and department = 'finance'"
   ```
   This query can take long time at the database level due to `WHERE` clause filtering a very long table on columns `date` and `department` that may not be indexed properly, leading to a full table scan without a `LIMIT` clause specified.
   ```
   -q "SELECT * FROM public.very_long_table_of_transactions transactions INNER JOIN public.departments departments ON transactions.department_id = departiments.id WHERE transactions.date = ${rd} and departments.name = 'finance'"
   ```
   This query can take long time at the database level due to a join between tables.

3. Resource limitation at the data source
   The data source may not have enough hardware resource available to efficiently fulfill the query and/or handle multiple DQ Jobs & other non-CDQ applications requesting data.

The bottleneck due to data transfer can be viewed in Jobs page > Job Logs > LOAD stage. This data loading step is the first step for all DQ Jobs. However, pushdown feature can be used to *reduce* (NOT eliminate) the data transfer & compute to Spark from the data source, if the DQ Job specified does not require loading all the data into Spark.

In summary, speed of loading the data from data source to Spark is a `-q query compute time at data source` + `network transfer of -q result between data source and Spark`. Pushdown can reduce both elements, but the efficiency gained is dependent on the complexity of `-q` query and how big the dataset from `-q`result would have been without pushdown.

How pushdown is efficient

Some of the Spark compute performed by the DQ Job can be translated into SQL queries that most relational databases support natively. In such a case, the DQ Job does not need to load all the rows of the dataset. Instead, the DQ Job can query the data source for the results of those SQL queries and reduce the amount of data transferring out of the data source. The results of these SQL queries are almost always lead to smaller amount of data compared to the full dataset defined by `-q`. Only some of the DQ Job features require the full dataset to be loaded into Spark. Therefore, pushdown can be a useful tool to speed up the overall DQ Job speed -- **provided that the speed of executing these SQL queries are faster than the speed of transferring the data out of the data source into Spark**. In most use cases, pushdown leads to faster DQ Job execution for large datasets. If the `-q` query is sufficiently complex, then the speed reduced by transferring less data into Spark can be cancelled out by the multiple frequent SQL queries made to the data source by the Pushdown process (because each query may have have redundant compute due to the complexity of `-q`).

How pushdown works in CDQ

Using pushdown only *reduces* the amount of data transferred out of the data source. It does NOT skip the LOAD stage in DQ Job. Every DQ Job requires a small sample of rows (10-20) of the dataset defined by `-q` in order to generate Data Preview and analyze schema information for the dataset run. This means the `-q` query may be fully computed at data source before the sampling can occur (depending on the complexity of the `-q`). In such a case, sampling 10-20 rows of data is not a quick and immediate LOAD stage and only efficiency gain comes from lack of transferring data between data source and Spark.

Therefore, pushdown feature would be most efficient if `-q` is a simple select query with simple where filtering. The benefit comes from the fact that if your dataset defined by `-q` results in 100 million rows, only 10-20 rows of the dataset defined by `-q` will be loaded into Spark.

Profile with pushdown will then generate a series of SQL queries and query the data source again for aggregate metric data. Depending on the dataset, these multiple SQL queries can be more efficient than loading all the data into Spark and computing these aggregate metric in Spark. The results between *Profile* with *pushdown* and *Profile* without *pushdown* are (practically) identical.

Profile: pushdown vs no pushdown

Here is the summary of Profile activity with details regarding pushdown support

Feature	Supports pushdown	Description
Row Count	Yes	Computes row count of the data-set.
Distinct Count	Yes	The number of distinct values in a column.
Mean	Yes	The average of all the values in the column. Supports numeric columns only.

Feature	Supports pushdown	Description
Min / Max	Yes	The minimum and maximum values of the column. Supports numeric and boolean columns only.
NULL Count	Yes	The number of null values in the column.
EMPTY Count	Yes	The number of empty values in the column. Supports string columns only.
TYPE Count	Yes	The number of different types inferred in a column (if any).
TopN / BottomN	No	Computes the top 5 most frequent (TopN) and least frequent (BottomN) values. Supports all types. This result is displayed as a frequency bar chart in Profile page. If pushdown is enabled, then TopN and BottomN values are not displayed. Related features like Stat Rules (Distribution) are also disabled.
Data Shape Detection	No	Detects shapes of the values based on Shape parameters provided (automatic or manual).
Histogram	No	Creates histogram of the values in the column.
Correlation Matrix	No	Creates correlation matrix. Supports only numeric columns.

Snowflake Pushdown

> Note As of 2022.11, Snowflake Pushdown is available as a public beta. Since this is a beta feature, some capabilities may be limited. For more information on our beta program, refer to Public betas.

Pushdown is an alternative computation method for running a DQ job, where all of the job's processing is submitted to a SQL data warehouse, such as Snowflake. Snowflake Pushdown jobs generate SQL queries to offload the compute to the data source, reducing the amount of data transfer and Spark computation of the DQ Job.

By running a Snowflake Pushdown job, you can:

- Reduce latency.
- Eliminate dependencies on Spark compute to run Collibra Data Quality, and increase processing speeds.
- Eliminate the egress costs for running DQ Jobs against large data sets.
- Auto-scale based on your processing requirements.

For more information on Snowflake, see the Snowflake documentation.

Prerequisites

Before running Snowflake Pushdown jobs, a user with Admin permissions must:

- Successfully run the Pushdown setup script.
- Enable Pushdown from the Collibra DQ UI.

Pushdown vs. Pull Up

Collibra DQ Pull Up is a DQ Job without pushdown, where all of the processing is executed inside the Apache Spark compute engine. Source data is stored inside a database, where Spark reads it out, and the parameters you set when you select a scope, define a range, and add build layers, are partitioned and sorted. The results of the profile job are then recorded in the DQ Metastore. Depending on the size of your data set and the number of DQ checks performed, this process can greatly slow run times because Spark has its own

compute resources, such as memory and CPUs. Pull up has limited support for profiling but you can't run it without setting up Spark.

With Snowflake Pushdown, the Collibra DQ Agent, which creates the Apache Spark DQ Job, is no longer needed. No agent is required to submit a Snowflake Pushdown job because all of the processing is sent directly to Snowflake. Therefore, Agent ID is always set to 0 for Snowflake Pushdown jobs.

With Snowflake, you can also scale your compute needs based on the specific requirements of your DQ Job. This is because Snowflake's architecture features auto-scaling, which allows you to automatically scale up, or *burst*, to 64 or 128 nodes when you require greater processing needs. Snowflake also automatically scales down when your DQ Job does not require robust processing. With auto-scaling, the processing of your data is enhanced, improving runtime performance and removing the egress costs of reading large amounts of data.

Features and Limitations of Snowflake Pushdown

> Note As of 2022.11, Snowflake Pushdown is available as a public beta. Since this is a beta feature, some capabilities may be limited. For more information on our beta program, refer to Public betas.

This section describes the features and limitations of using Collibra Data Quality & Observability's Snowflake Pushdown capability to run a DQ Job.

> Warning To use Snowflake Pushdown, you must be a participant in the public beta. This feature is currently unavailable for non-participants.

Features

The following table shows the features unique to DQ Jobs run using Snowflake Pushdown processing.

Feature	Description
Dynamic date filter	A togglable option that allows you to filter column data dynamically by run-Date $.
Configurable number of connections	Allows you to set the number of open connections between 1-5 so you can run jobs in parallel and improve the performance of profile jobs.
Cancel jobs	Unlike Spark compute jobs, you can cancel the SQL queries of Snowflake Pushdown jobs.
No agent	Pushdown runs the database engine to execute jobs directly, removing the need for agents. Agent ID = 0.
More control over AdaptiveRules	With minimal clicks, you can apply AdaptiveRules, such as row count and uniqueness, from the UI.

Known Limitations

Currently, there are some limitations with Snowflake Pushdown because it is in public beta as of 2022.11:

- Pattern detection is not yet supported.
- You cannot currently run a Job from the command line.
- Mixed data type rules are not supported in Pushdown.
- We do not currently support Okta integration. Support for this integration is planned for a future release.

Running a Snowflake Pushdown job

> Note As of 2022.11, Snowflake Pushdown is available as a public beta. Since this is a beta feature, some capabilities may be limited. For more information on our beta program, refer to Public betas.

This section shows you how to get started with the three scanning methods of a Snowflake Pushdown job.

Steps

1. From Explorer, select your Snowflake connection with the PD icon next to it.

   > Note For the PD icon to be visible, you need to enable Pushdown when you establish your Snowflake connection.

2. Select your schema.
3. Click Create DQ Job.
   The Job creator page opens.
4. Select a scanning method. Since the steps and procedures vary by scanning method, refer to the documentation associated with each scanning method for further instructions.

Scanning method	Description
Full Scan	Scans your full table to show all results. This is the standard scanning method.
Partial Scan	Scans a section of your table for more targeted results than a full scan.
SQL Query	Allows you to manually write and compile a SQL query for an advanced scan of your table.

AdaptiveRules

AdaptiveRules are common metrics used to observe changes to your data. These can be applied or removed from the AdaptiveRules tab on the Add Layers workflow by selecting or deselecting AdaptiveRules.

The following table shows a list of AdaptiveRules measured by Collibra DQ and whether they are applied by default.

AdaptiveRule type	Subtype	Description	Default?
Availability	N/A	Observe changes to the row count and loading time in your table.	N/A
	Row count	Monitor the row count change in your table.	True
	Loading time	Monitor loading time changes.	False
Distribution	N/A	Observe the number of unique values in a table.	N/A
	Uniqueness	Monitor a column's cardinality within the range of previous DQ Jobs	True
Conformity	N/A	Observe columns with values that fall outside of the normal range.	N/A
	Min	Monitor columns with min values outside the normal range.	False
	Mean	Monitor columns with mean values outside the normal range.	False
	Max	Monitor columns with max values outside the normal range.	False
Completeness	N/A	Observe columns in your table containing null values or empty fields.	N/A
	Null values	Monitor columns for null values.	True
	Empty values	Monitor columns for empty data.	True

Note You can always apply or remove AdaptiveRules, but if you bypass the configuration, the DQ Job will still run correctly with the default AdaptiveRules applied.

Shapes

A shape is the format of data in a string column. Enabling Shapes lets you discover inconsistencies in the data formats of a column. For example, when analyzing a date column, Collibra DQ may detect different string formats of the same meaning, such as 11-15-2022, 11/15/2022, and 11.15.2022.

The Shapes feature is on by default, but you can choose to toggle it off. You can also manually control the advanced options by checking the Manual checkbox.

Option	Description
Occurrences	Set occurrences between 0.001-5 to show shapes that occur less than the percentage you set.
Format per column	Set format per column between 0-100 to identify columns with more formats than the number you set. This helps identify columns that are too noisy, meaning they do not have a consistent format for detection.
Character length	Set character length between 0-100 to identify string lengths that are greater than the number you set. Character lengths beyond the set value are considered free from fields and do not have a format for detection.
Data shape granular	Considers the length of the shape and differentiates between numerical and letter formats. For the length of a shape to be detected, data shape granular must be enabled.

Replay

Replay gives you a historical behavior profile of your data by looking back into past runs to show how the data looked on a certain day, month, or year. To enable Replay, select the dropdown icon on the Run button and select **Replay**.

Outliers

Outliers are values that differ significantly from the rest of the data and may indicate bad or erroneous data. Numerical outliers are detected using the IQR and box plot methods.

You can detect for outliers by selecting the **Outliers** tab on the Add Layers screen. Click the **Add Outlier** button to open the new Outlier detection bucket's setup options. The following table shows the setup options and their descriptions:

Option	Description
Column	The column(s) used to detect outliers. You can select multiple columns for outlier detection. Required.
Type	The type of data in a given column. For example, VARCHAR.
Key	The column used for grouping detected outliers. When assigned, outliers are grouped to their assigned column when detected. Optional.
Date	The column used for bucketing. Optional.

Click **Save** to save your outlier bucket.

Optionally click the three dots menu icon to configure **Quartiles**, **Lookback**, and determine whether or not to detect for **Categorical** outliers. The following table shows the configuration options and their descriptions:

Option	Description
Quartile 1	The boundaries beyond which low (Q1) outliers are detected using the IQR formula. Use the slider or enter a value in the text box between 0.01 and 0.45.
Quartile 3	The boundaries beyond which high (Q3) outliers are detected using the IQR formula. Use the slider or enter a value in the text box between 0.55 and 0.99.
Lookback	The period used to look back through the dataset using the date column. Use the slider or enter a value in the text box between 0 and 30.
Interval	The unit of measurement of the lookback period, such as DAY.
Categorical	Select this option to detect for Categorical outliers instead of Numerical outliers.

Optionally click the pencil icon near **Advanced Options** to adjust outlier detection sensitivity. Use the slider to adjust outlier detection **Sensitivity** from Low to High for a given column. You can also change the **Unit** of outlier detection sensitivity with the dropdown menu. Click **Save**.

Click the pencil icon to edit a saved Outlier bucket, or click the trash can icon to remove it.

Running a Full Scan

> Note As of 2022.11, Snowflake Pushdown is available as a public beta. Since this is a beta feature, some capabilities may be limited. For more information on our beta program, refer to Public betas.

A full scan is a scan of your entire table. When running a full scan, you do not need to select columns or apply filters to rows because all columns are selected by default. Various DQ Layers, such as AdaptiveRules, are also applied by default. A full scan is commonly used to obtain a high-level overview of your data.

Prerequisites

You have an Connectivity to Snowflake and Pushdown enabled.

Steps

1. From Explorer, select your Snowflake connection with the PD icon next to it.

   > Note For the PD icon to be visible, you need to enable Pushdown when you establish your Snowflake connection.

2. Select your schema.
3. Click **Create DQ Job**.
   The Job creator page opens.
4. Select **Full Scan**.
5. Optionally add layers or toggle default settings.
   - Certain AdaptiveRules are enabled by default. You can optionally toggle them on and off.
   - Shapes are enabled by default. You can optionally toggle them on and off, and configure more advanced options by selecting the Manual checkbox.
6. Click **Next**.
   The Review page opens.
7. Review your DQ scan.

8. Select **Run**.
   A dialog appears and tracks the status of your Snowflake Pushdown job.

What's next?

After running a Full Scan, go to the Jobs page to Profile (automatic).

Running a Partial Scan

> Note As of 2022.11, Snowflake Pushdown is available as a public beta. Since this is a beta feature, some capabilities may be limited. For more information on our beta program, refer to Public betas.

A partial scan only scans the sections of your table that you specify Select columns, apply row filters, and add DQ layers to fine tune your DQ scan and obtain a targeted understanding of your data.

Prerequisites

You have an Connectivity to Snowflake and Connectivity to Snowflake.

Steps

1. From Explorer, select your Snowflake connection with the PD icon next to it.

   > Note For the PD icon to be visible, you need to enable Pushdown when you establish your Snowflake connection.

2. Select your schema.
3. Click **Create DQ Job**.
   The Job creator page opens.
4. Select **Partial Scan**.
   The Select columns view appears.
5. Select your columns by checking or unchecking the checkboxes associated with a particular column.
6. Click **Next**.
   Alternatively, you can click Select rows from the left workflow menu.
   The Select rows view appears.
7. Select a filter type to apply a specific filter.

Filter Type	Description
Time Slice	Scans a range of dates or times.
Row Filter	Scans a section of rows.
Limit	Scans a random sample of rows.

8. Click **Next**.
   Alternatively, you can click **Add layers** from the left workflow menu.
   The Add layers view appears.
9. Optionally add layers or toggle default settings.
   - Certain AdaptiveRules are enabled by default. You can optionally toggle them on and off.
   - Shapes are enabled by default. You can optionally toggle them on and off, and configure more advanced options by selecting the Manual checkbox.
10. Click **Next**.
    The Review page opens.
11. Review your DQ scan.
12. Select **Run**.
    A dialog appears and tracks the status of your Snowflake Pushdown job.

Time Slice

Time slice is a filter that lets you select a range of dates or times for your DQ scan. To apply a time slice filter, select **Add**, configure your preferences in the Add Time Slice modal, then select **Save** to save your filter. When a filter is successfully saved, a badge displays on the Time Slice tile and the number of applied filters is reflected in the left workflow menu.

Edit Time Slice modal

Component	Description
Column of reference	Select a column from your table for the application to analyze.
Date	When toggled on, the date filter lets you filter column data based on run date `${rd}`.
Visualize	Displays an autogenerated bar chart of days where data is available in your data set.
Operator	Select an operator, such as =or >, upon which the app bases its operation.
Run Date	Select a run date to allow the application to scan sections of your table, add a new date to it, and then run again. By selecting Run Date, a dynamic result is returned at runtime.

Row Filter

Row Filter scans a section of rows from your table. To add a Row Filter, select **Add**, configure your preferences in the Add Filter modal, then select **Save** to save your filter. When a filter is successfully saved, a badge displays on the Time Slice tile, and the number of applied filters is reflected in the left workflow menu.

Add Filter modal

Component	Description
Column of reference	Select a column from your table for the application to analyze.
Date	
Visualize	Displays an autogenerated bar chart of days where data is available in your data set.

Component	Description
Operator	Select an operator, such as =or >, upon which the app bases its operation.
Value	

Limit

Limit scans a random sample of rows based on the maximum number of rows you set. To apply a limit, specify the number of rows greater than 0 for DQ to scan.

What's next?

After running a Partial Scan, go to the Jobs page to Profile (automatic).

Scanning with SQL Query

> Note As of 2022.11, Snowflake Pushdown is available as a public beta. Since this is a beta feature, some capabilities may be limited. For more information on our beta program, refer to Public betas.

The SQL Query option lets you write and compile a query manually for an advanced scan of your table.

Prerequisites

You have an Connectivity to Snowflake and Connectivity to Snowflake.

Steps

1. From Explorer, select your Snowflake connection with the PD icon next to it.

   > Note For the PD icon to be visible, you need to enable Pushdown when you establish your Snowflake connection.

2. Select your schema.
3. Click **Create DQ Job**.
   The Job creator page opens.
4. Select **SQL Query**.
   The SQL view opens.
5. Write and compile a SQL query.

   > Note To switch to Standard view and return to the Running a Partial Scan workflow, select the three-dots menu icon and select **Standard view**. Any changes made to your SQL query are lost when you switch between views.

6. Click **Next**.
   Alternatively, you can click **Add layers** from the left workflow menu.
   The Add layers view appears.

7. Optionally add layers or toggle default settings.
    - Certain AdaptiveRules are enabled by default. You can optionally toggle them on and off.
    - Shapes are enabled by default. You can optionally toggle them on and off and configure more advanced options by selecting the Manual checkbox.
8. Click **Next**.
   The Review page opens.
9. Review your DQ scan.
10. Click **Run**.
    A dialog appears and tracks the status of your Snowflake Pushdown job.

What's next?

After scanning with SQL Query, go to the Jobs page to Profile (automatic).

Summary

Click or Code

Collibra DQ offers easy to use no (low) code options for getting started quickly. Alternatively, more technical users may prefer programmatic APIs.

Core Components

Collibra DQ offers a full DQ suite to cover the unique challenges of each data set.

9 Dimensions of DQ

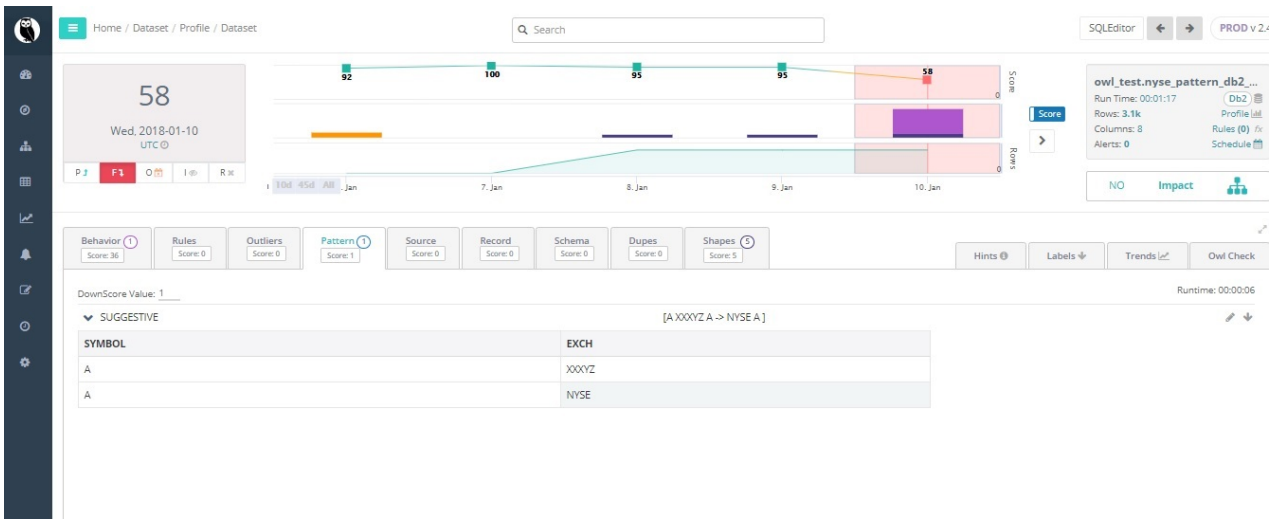
1. Behaviors - Data observability
2. Rules - SQL-based rules engine
3. Schema - When columns are added or dropped
4. Shapes - Typos and Formatting Anomalies
5. Duplicates - Fuzzy matching, Identify similar but not exact entries
6. Outliers - Anomalous records, clustering, time-series, categorical

7. Pattern - Classification, cross-column & parent/child anomalies
8. Record - Deltas for a given column(s)
9. Source - Source to target reconciliation

Check out our videos to learn more

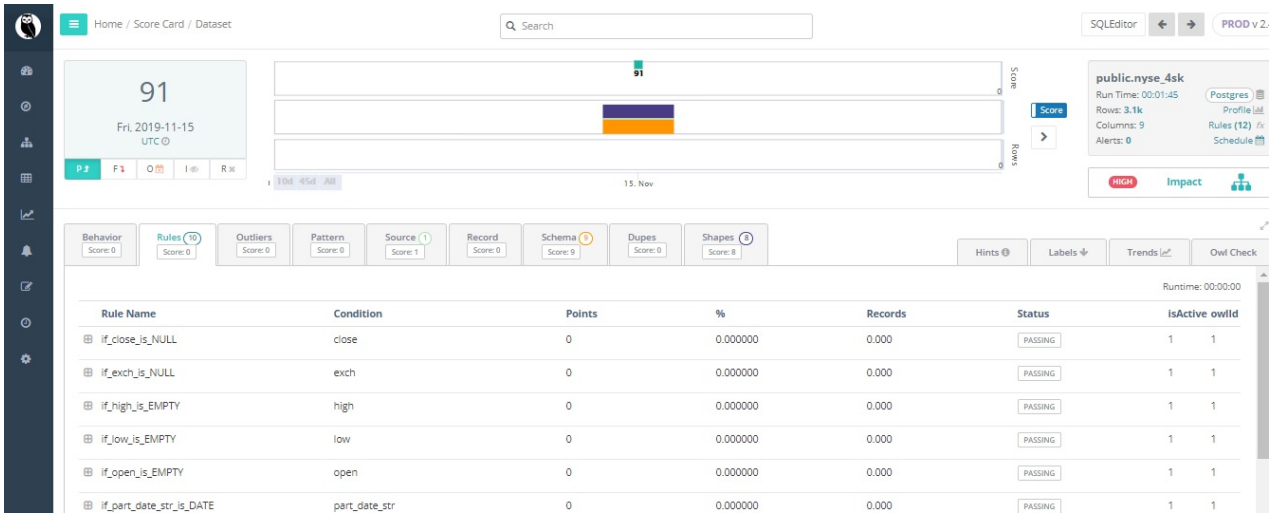
Behavior

Imagine a column going null, automatic row count checks - does your data behave/look/feel the same way it has in the past.



Rules

Assures only values compliant with your data rules are allowed within a data object.

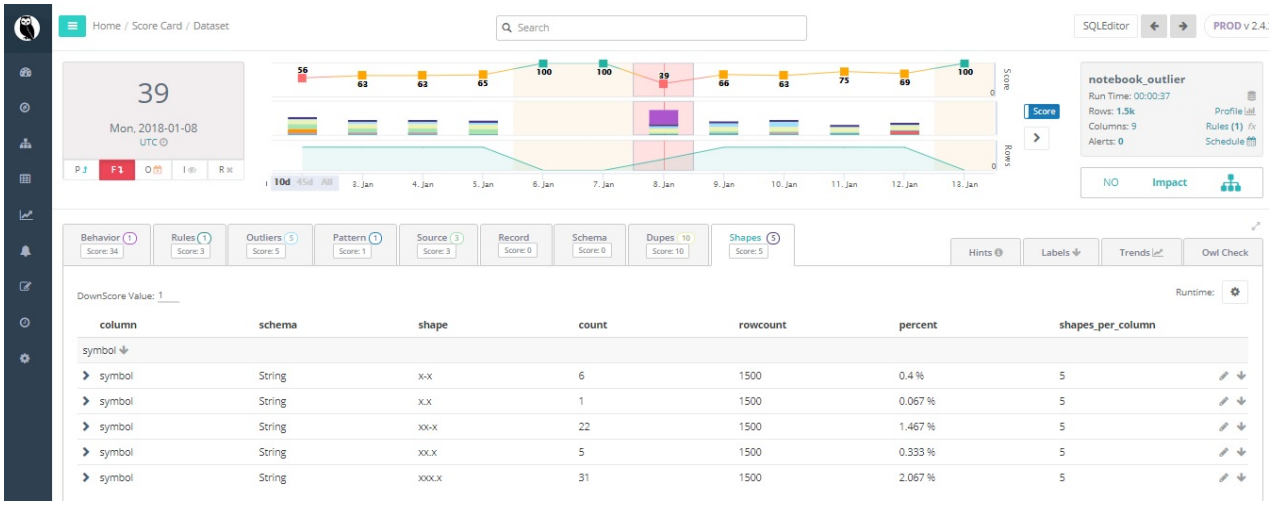


Schema

Columns add or dropped.

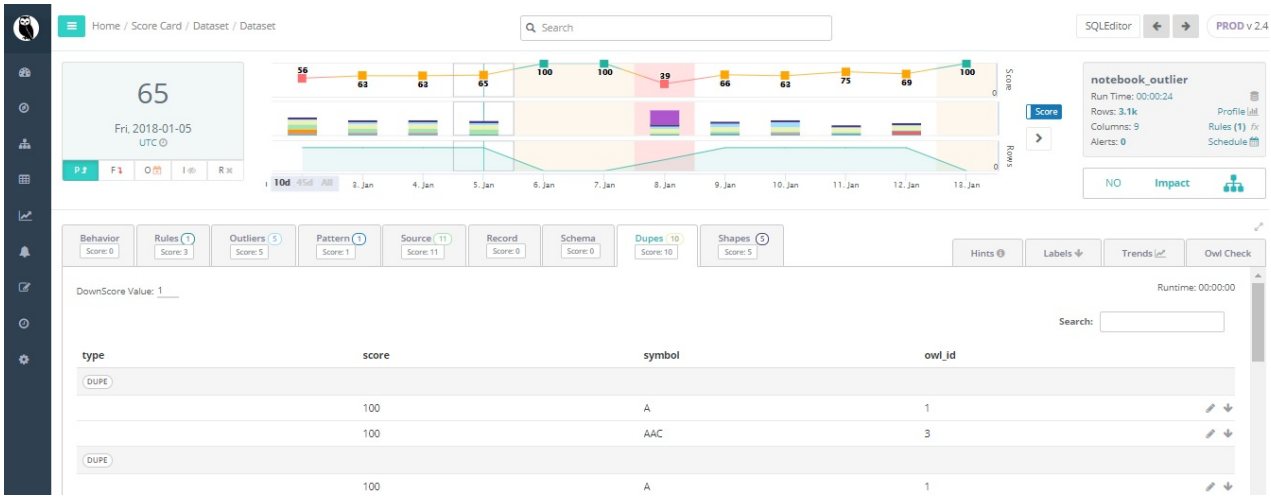
Shapes

Infrequent formats.



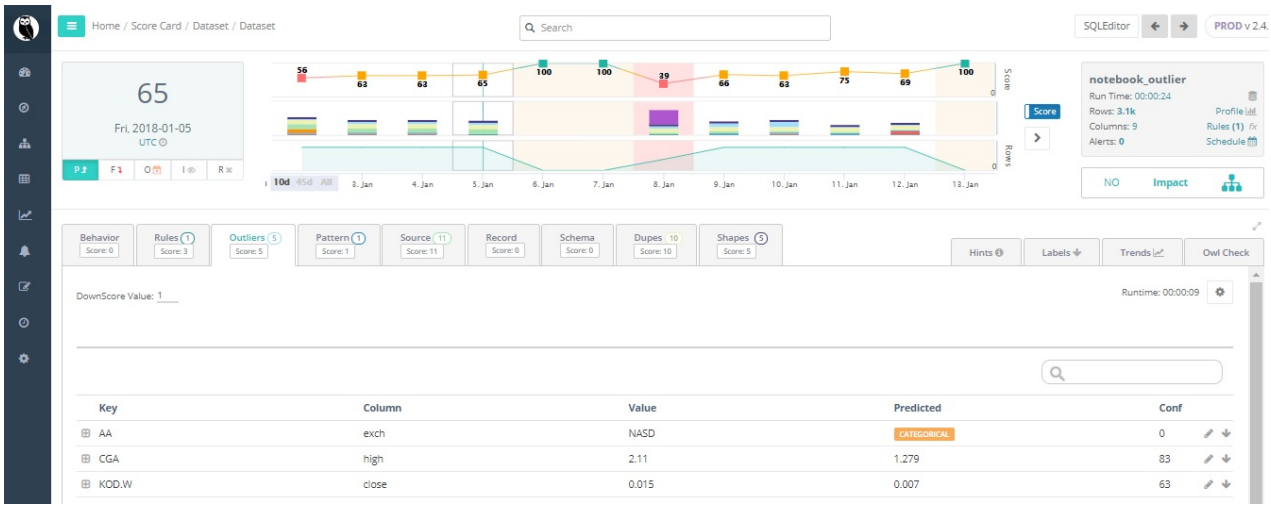
Dupes

Fuzzy matching to identify entries that have been added multiple times with similar but not exact detail.



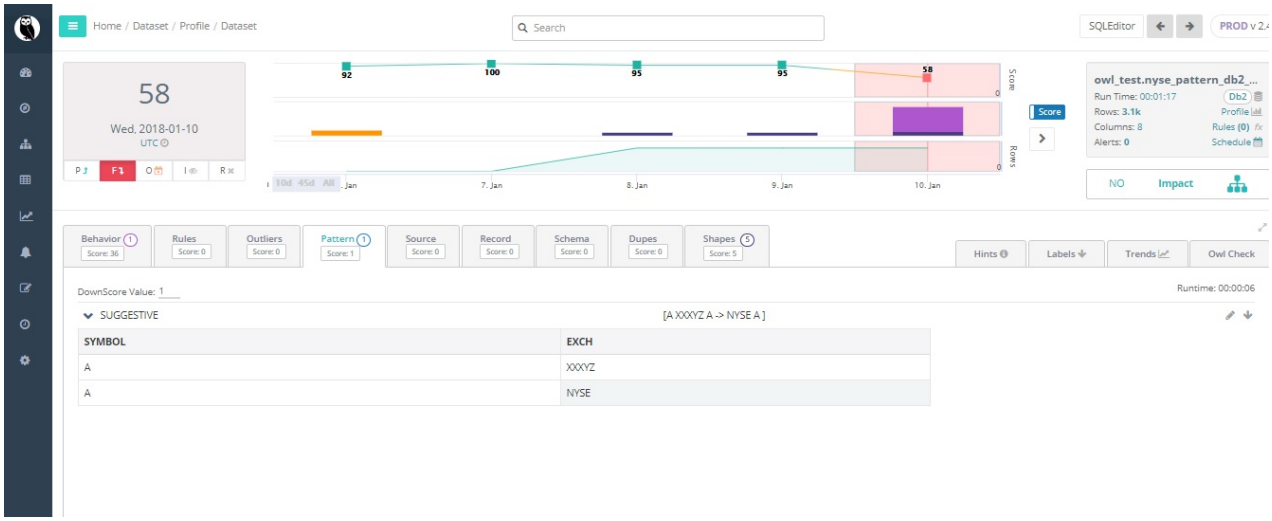
Outliers

Data points that differ significantly from other observations.



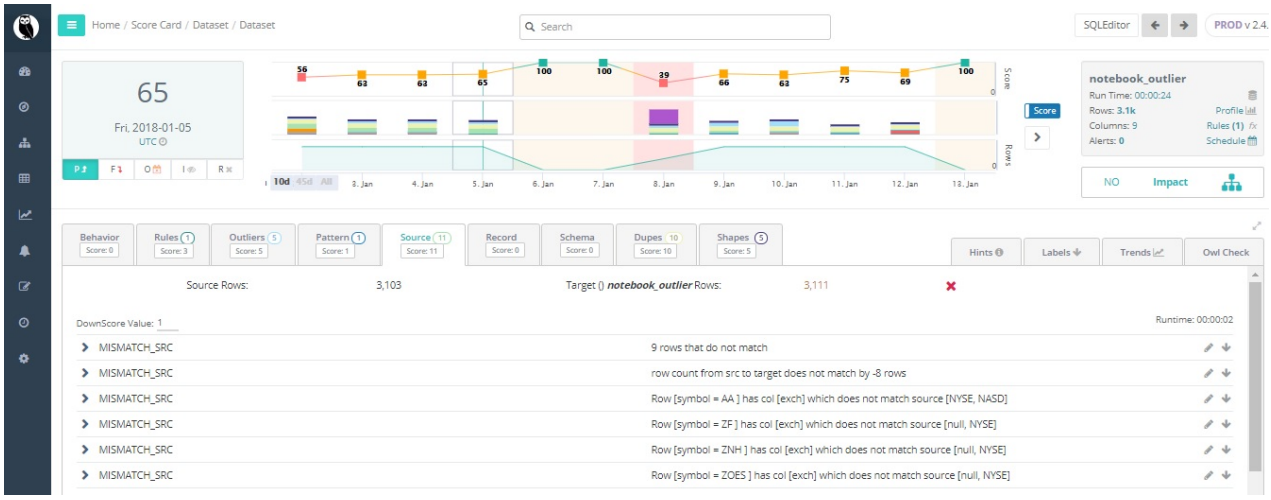
Pattern

Recognizing relevant patterns between data examples.



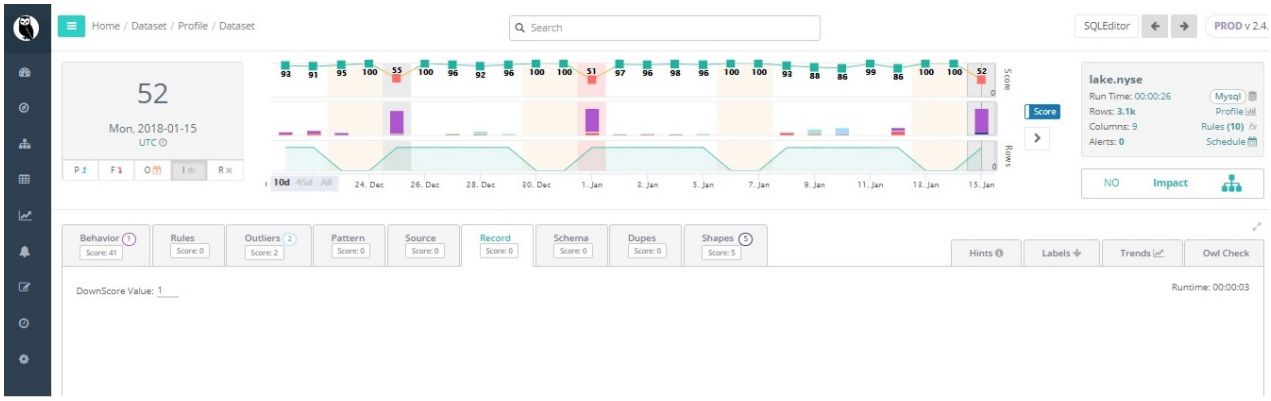
Source

Validating source to target accuracy.



Record

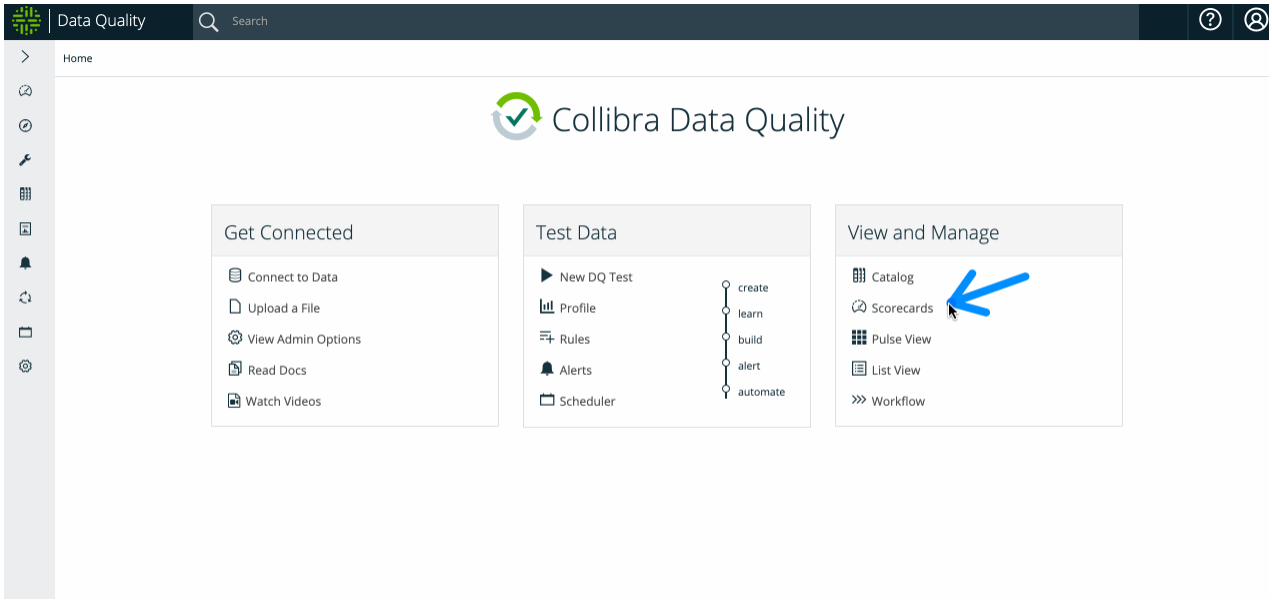
Deltas for a given column.

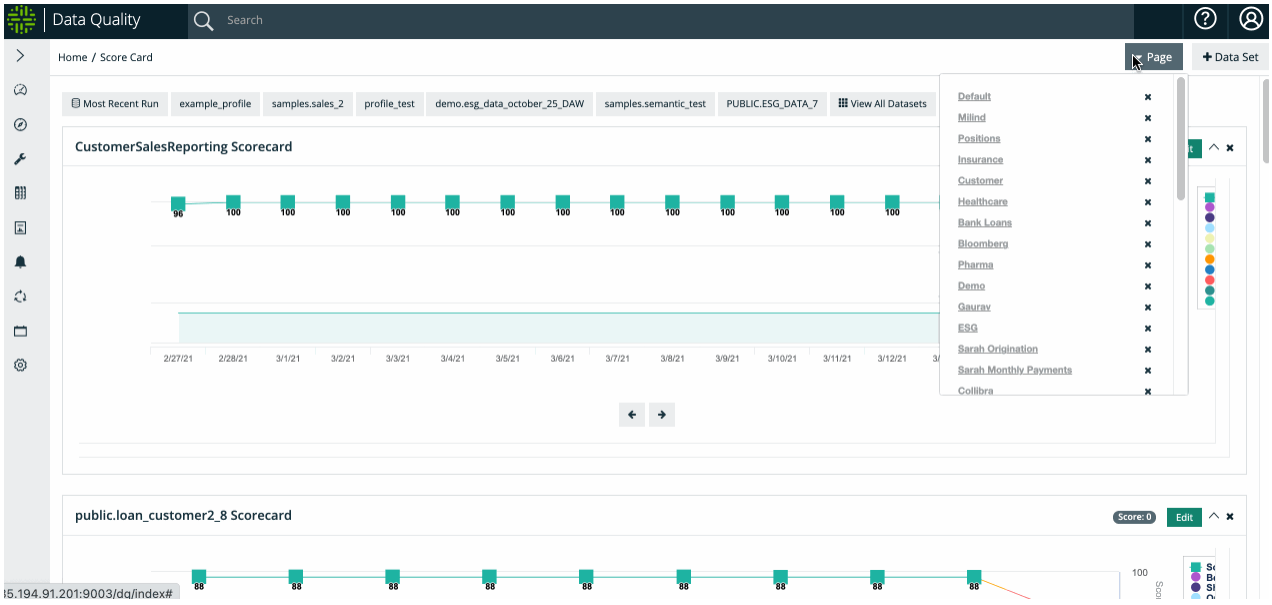

Home / Dataset / Profile / Dataset
Search
SQLEditor
PROD v 2.4
52
Mon, 2018-01-15
UTC
93 91 95 100 55 100 96 92 96 100 100 51 97 96 98 96 100 100 93 88 86 99 86 100 100 52
Score
Rows
10d 45d All
24. Dec 26. Dec 28. Dec 30. Dec 1. Jan 3. Jan 5. Jan 7. Jan 9. Jan 11. Jan 13. Jan 15. Jan
lake.nyse
Run Time: 00:00:26
Rows: 3.1k
Columns: 9
Alerts: 0
Mysql
Profile
Rules (10)
Schedule
NO
Impact
Behavior 1 Score: 41
Rules Score: 0
Outliers 2 Score: 2
Pattern Score: 0
Source Score: 0
Record Score: 0
Schema Score: 0
Dupes Score: 0
Shapes 5 Score: 5
Hints
Labels
Trends
Owl Check
DownScore Value: 1
Runtime: 00:00:03

Chapter 8

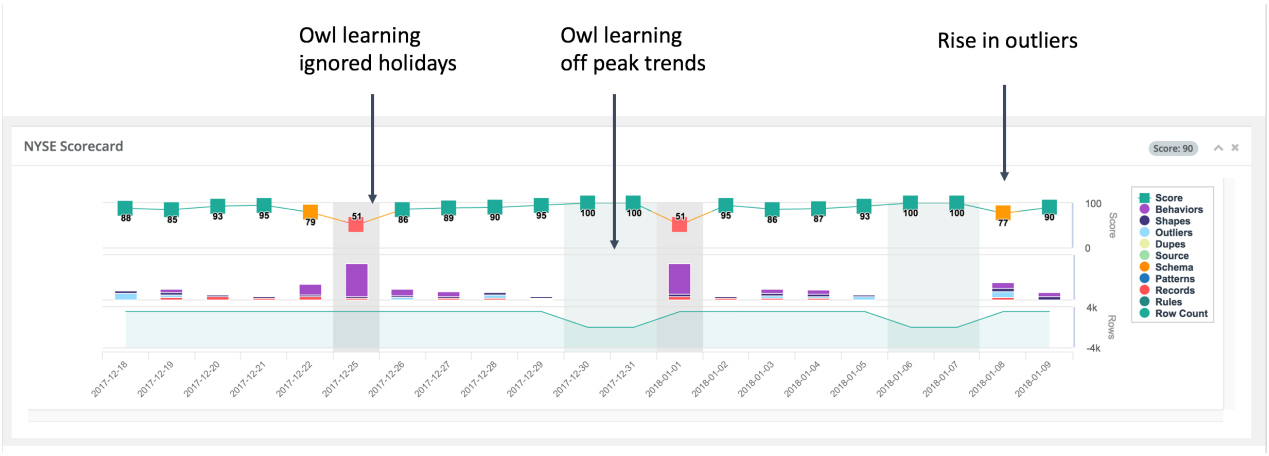
Collibra DQ Scorecards

Overview





Scorecards allow you to visualize the health and consistency of a data set over time. DQ highlights macro and micro trends, for example, weekend loads vs weekday loads or behavioral item changes per day, and display them on the dataset scorecard.

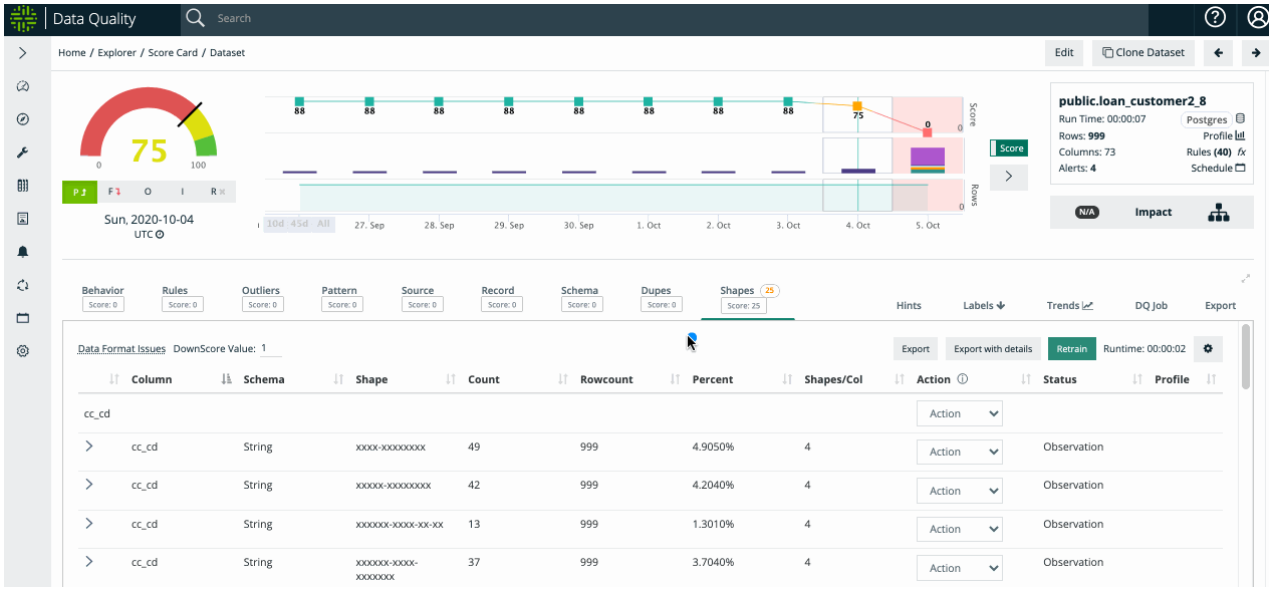


Data Quality Over Time, Drill-In and Roll-Up

Data quality doesn't mean a one time check or once a year project. Data is the life blood flowing through your organization. It's mandatory to know how your data is behaving right now, yesterday and over time to gain an understanding of the trends. For insights to be meaningful, we need to see both the lowest granularity and the big picture. DQ's approach lets you drill all the way into the exact moment the issue arose, as well as zoom out to see how your data is behaving month to month. This makes DQ useful at many different levels in your organization's heirarchy - a Data Steward might be more concerned with a recent change in data and want to correct it using Service Now immediately, whereas a Chief Data Officer might be more concerned with the overall health of the organization's data.

Scoring

Scoring can be completely controlled by the end-user with out of the box defaults.

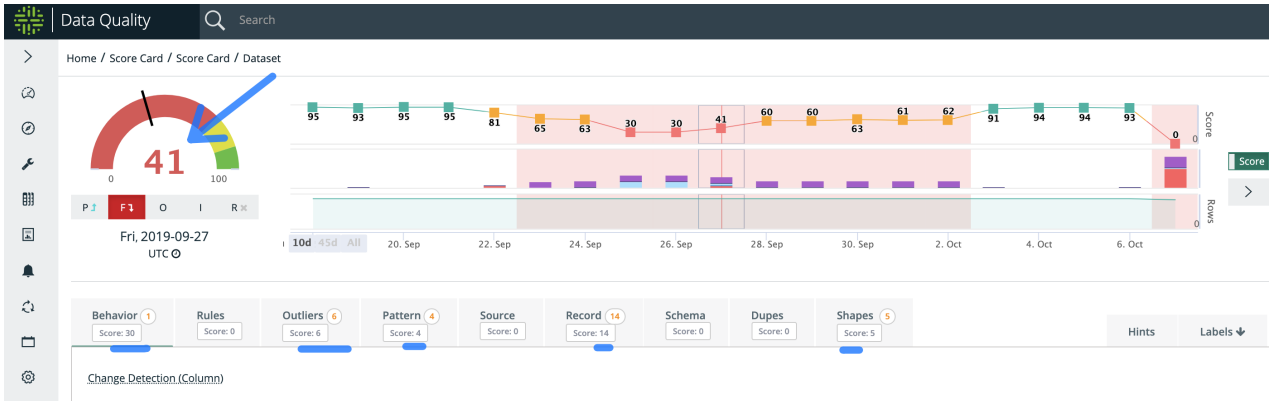


Collibra Data Quality provides a data quality assessment that scans nine dimensions of a data set to assure the integrity of that data. The nine dimensions include behavior, rules, outliers, pattern, source, record, schema, duplicates, and shapes.

OwlCheck produces a data quality score from 0-100. 100 represents no integrity issues found in the data set. The score numerically represents the integrity of that data. For example, the score of 100 tells the data analyst that there are zero data quality issues in that data set.

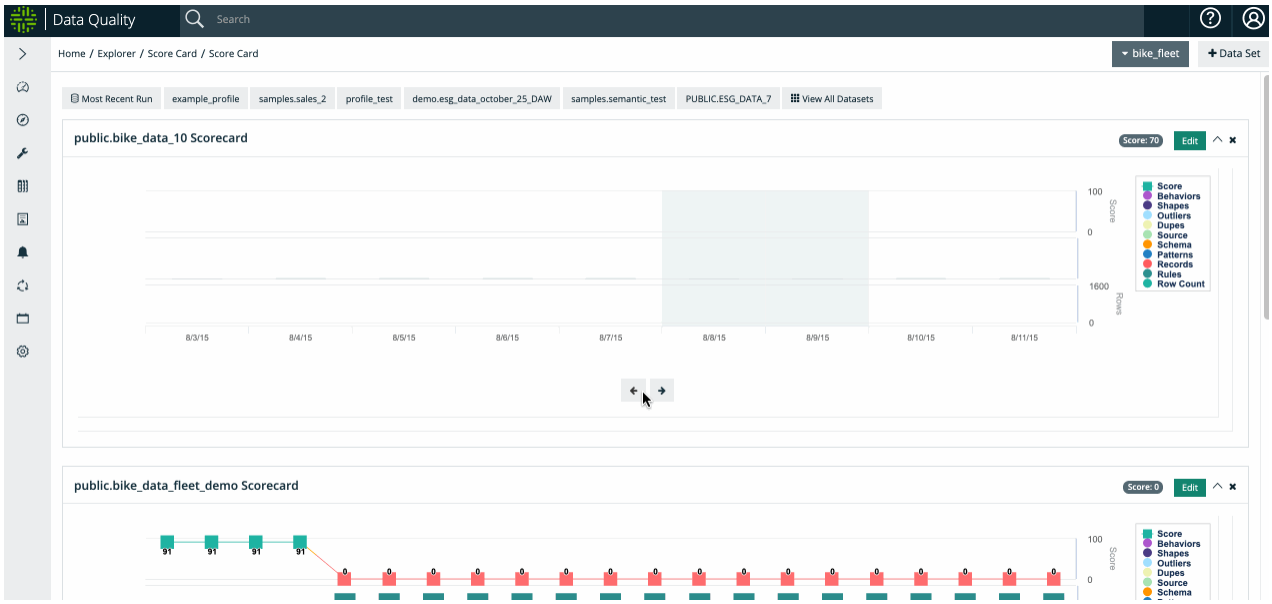
DQ scans your data with the same frequency. You load your data - Owl scans nine dimensions of DQ and summarizes the results into a score from 0-100.

Aggregate Score



> Note Each dimension can be custom weighted and rules can contain custom scoring severity. In this example, the deducted score (59) from the starting score (100) equals an overall score of 41.

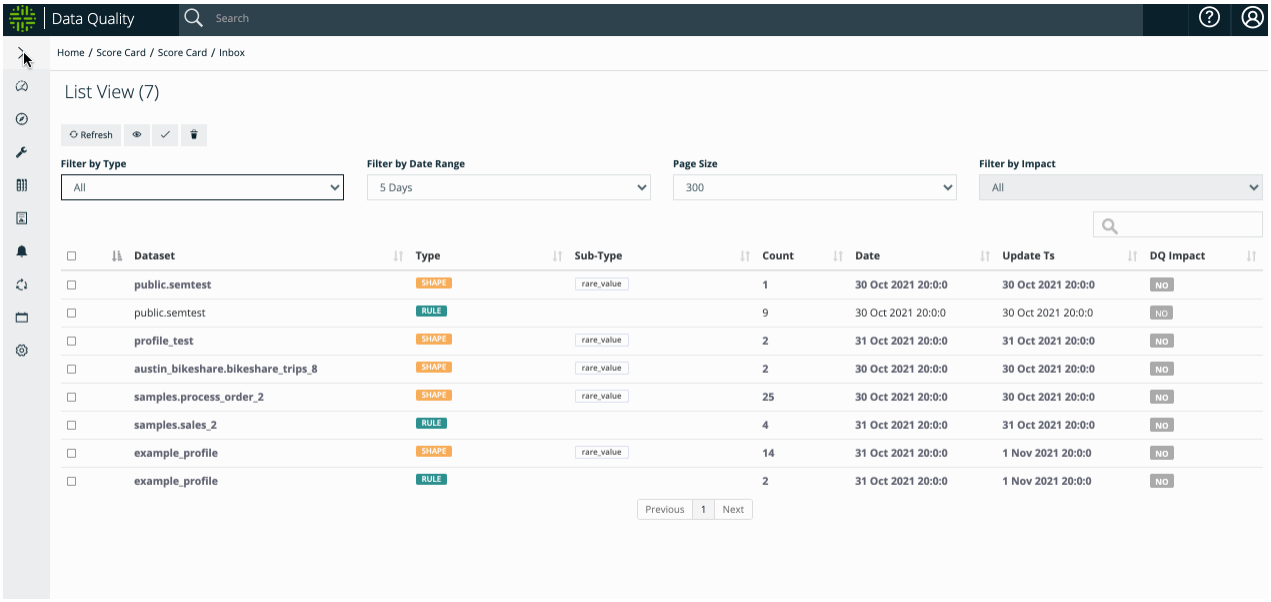
Page View



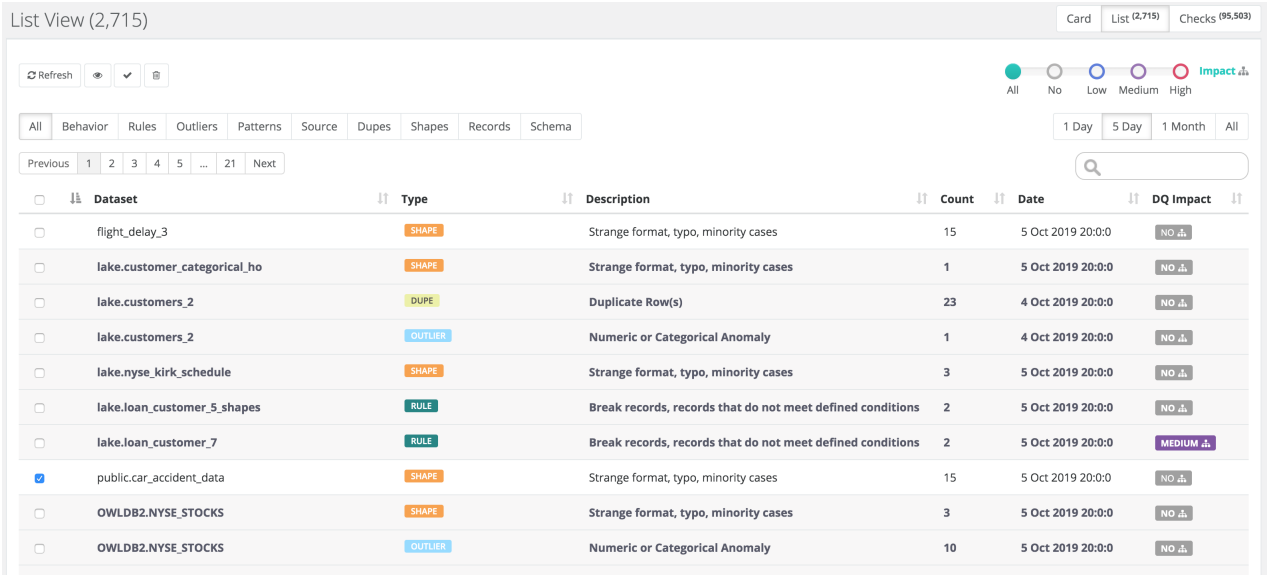
Visually and logically group data sets together to create a heat map of blindspots.

Similarly to job control and build frameworks like Jenkins, we always want to know the health of our data sets. Did it recently fail? Does it commonly fail on Mondays? What is the aggregate or composite score for multiple data sets? DQ allows you to define scheduled health checks that depend on the success/failure status of any number of data sets. This protects the downstream process consumers from pulling erroneous data into their models.

List View



Find DQ issues across all data sets in your data lake. Rank them, sort them, search them and limit them by time.



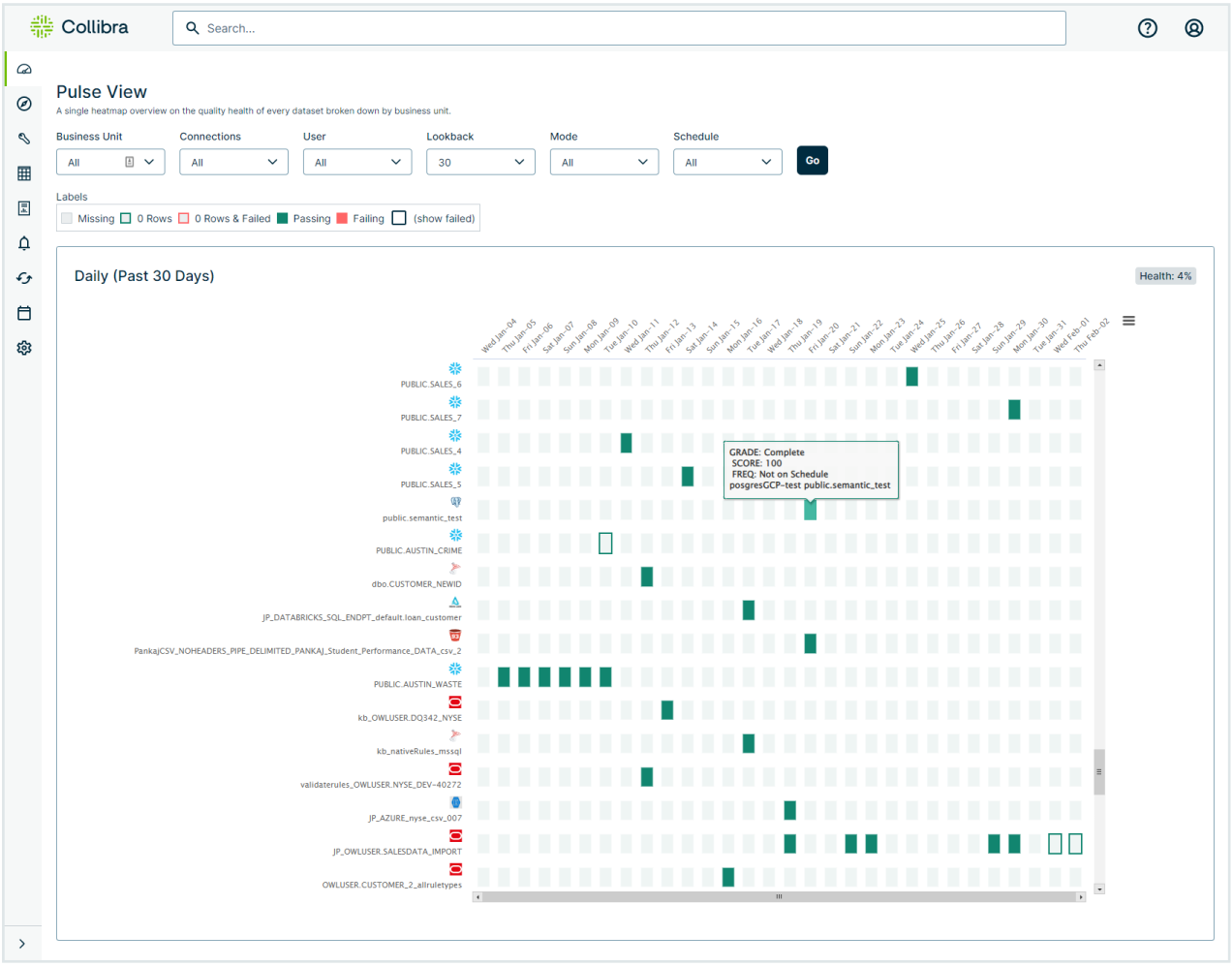
How Do I Take Action on Lots of Alerts?

One of the most frequently asked questions is how to operationalize and take action on all of the issues that are seemingly valid yet overwhelming. DQ seems to find many valid issues in your data sets, but there are many more issues in your data than expected. The list view helps by first limiting to a time range, for example, issues that have occurred less than five days, or possibly even just issues that occurred today. This will likely result in a large reduction in the issue count. In addition, you might limit issues to those DQ issues that have a business impact. This means other downstream processes or data sets are connected to this data set and field. You only get the business impact feature if you have enabled the DQ Graph module. This results in another drastic reduction in issues because now you are limited to issues that recently occurred and have impact to the business. Finally, you might filter by the "class" or "type" of DQ issue, such as Rules or Outliers. It is common in a large data lake that after taking these steps you are left with the one or five top ranking issues in your lake. These are likely the issues that should be prioritized and moved into a remediation queue.

Pulse View

The Pulse View provides a snapshot of Collibra DQ Jobs that were executed, whether manually or from a schedule. On the Pulse View dashboard, you can view heat maps of the health of your jobs, and filter by the following criteria:

- Business Unit
- Connections
- User
- Lookback
- Mode
- Schedule



The following screenshot shows an example of the type of data you can view when you hover your cursor over a heat map label, for example, a grade, score, and scheduled frequency.

The following table describes the heat map labels you can toggle for additional data about Jobs that ran for Collibra DQ. The charts are based on criteria you select from the drop-down lists on the Pulse View page.

Label	Label icon	Description
Missing		A run that is missing either failed at run time or was never created, started, or picked up.
0 Rows		A run that is passing or complete and has resulted in 0 rows of data.
0 Rows & Failing		A run that is failing or incomplete and has resulted in 0 rows of data.
Passing		A run that is passing or complete and has resulted in rows of data.
Failing		A run that is failing or incomplete and has resulted in rows of data.
(show failed)		When you select this check box, only the runs that failed display. When this option is not selected, every job that ran displays, whether the score was passing or failing.

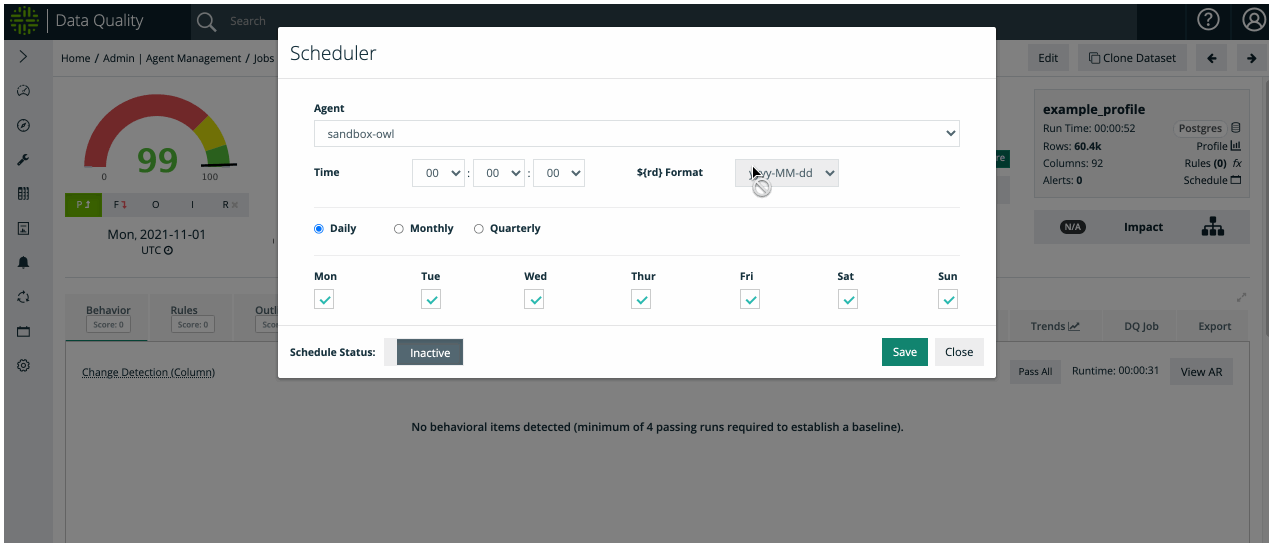
Note Jobs that run ad hoc or manually do not appear on the Pulse View page.

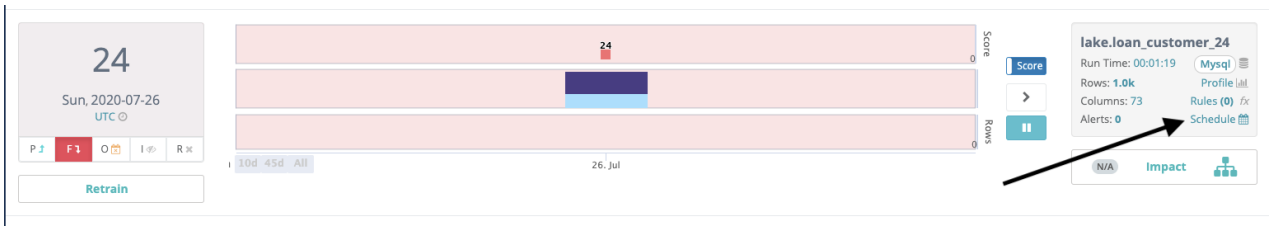
Chapter 9

Collibra DQ Scheduler

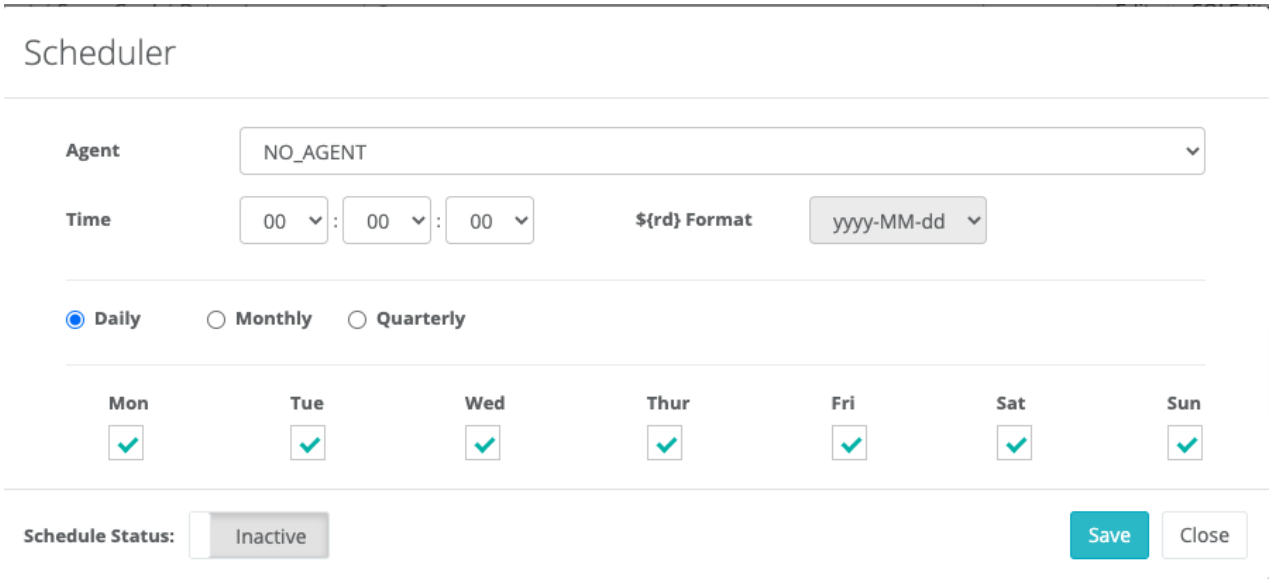
Schedule a Job

After you successfully run a job, you can schedule that job to run automatically. Do this by updating the template (if needed) and clicking the schedule icon in the hoot page. To change the template, you can use the -rd variable: $ in your query to set dynamic dates or date ranges for your scheduled job.

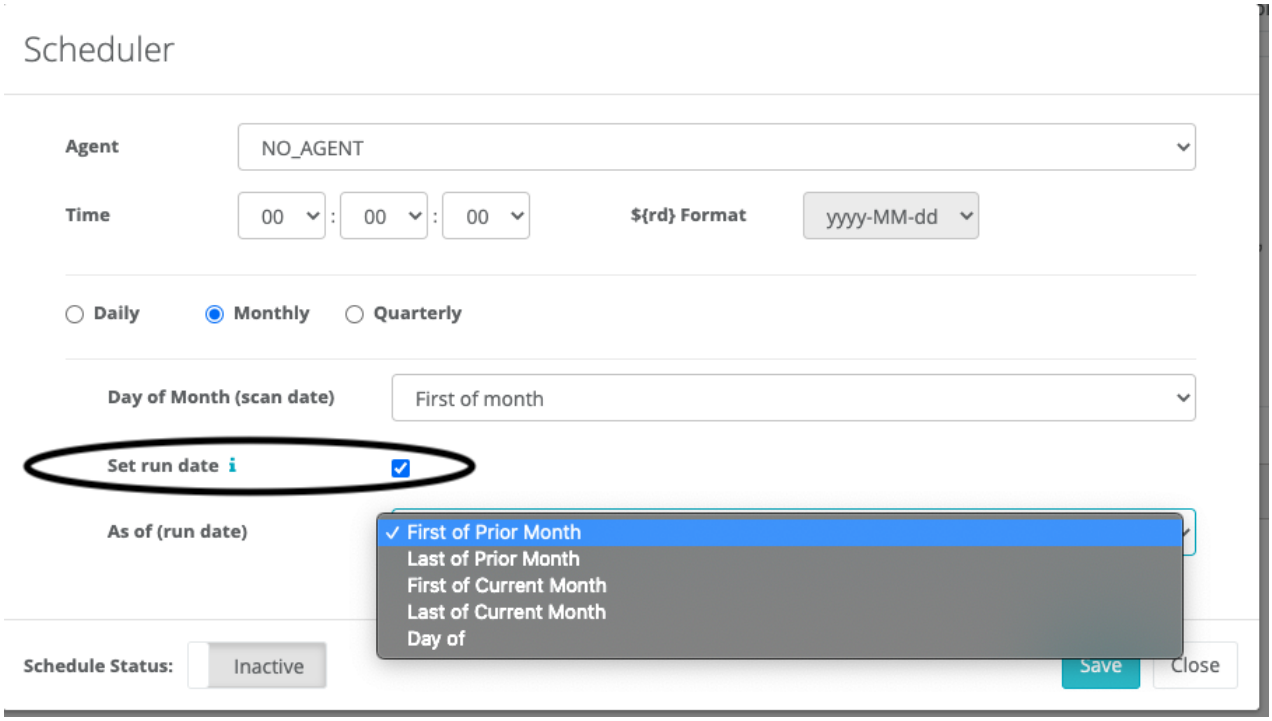




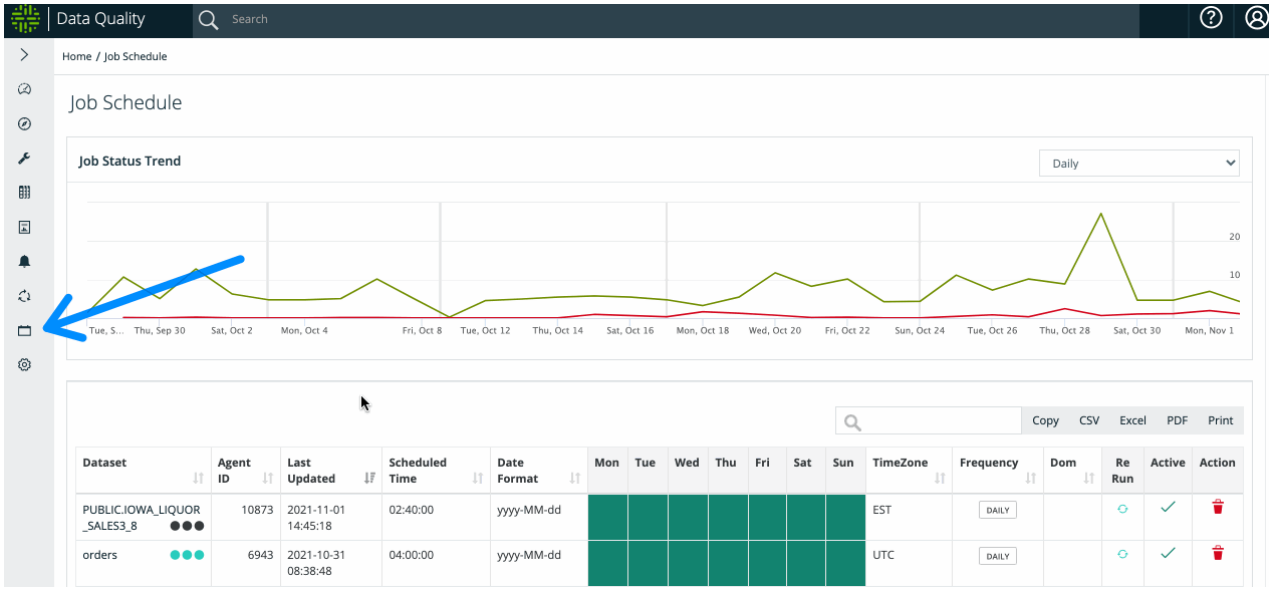
Here you can choose the Agent to run the job, the frequency (daily/monthly/quarterly) and the time of day:



If your monthly or quarterly jobs are loaded after the month or quarter has ended, you can schedule the job for the day when the data has landed, but set the offset to the proper run date required for charting/reporting.



Schedule Management

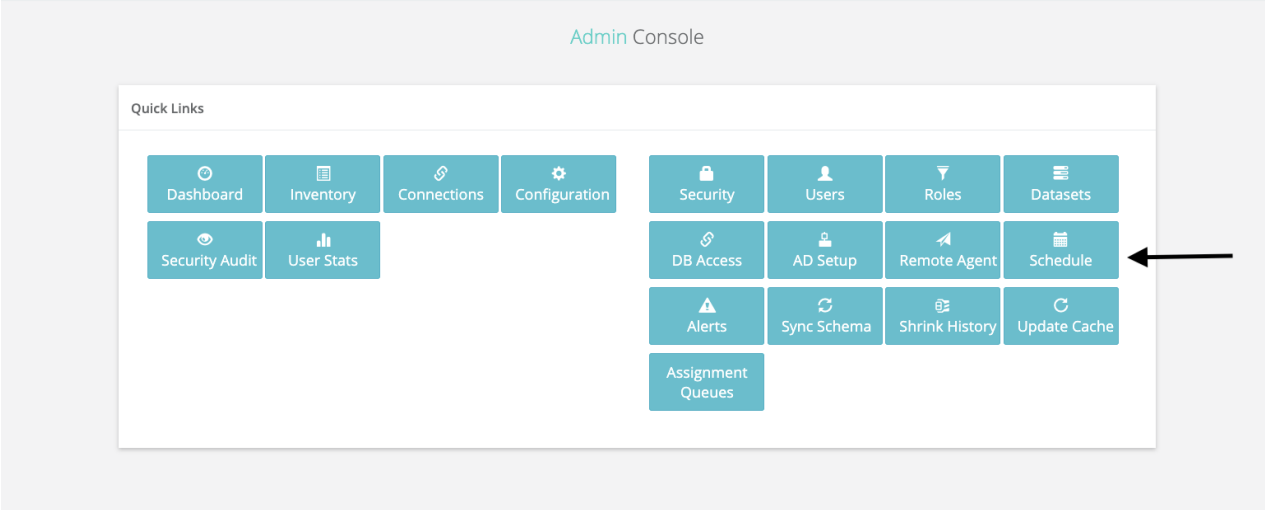


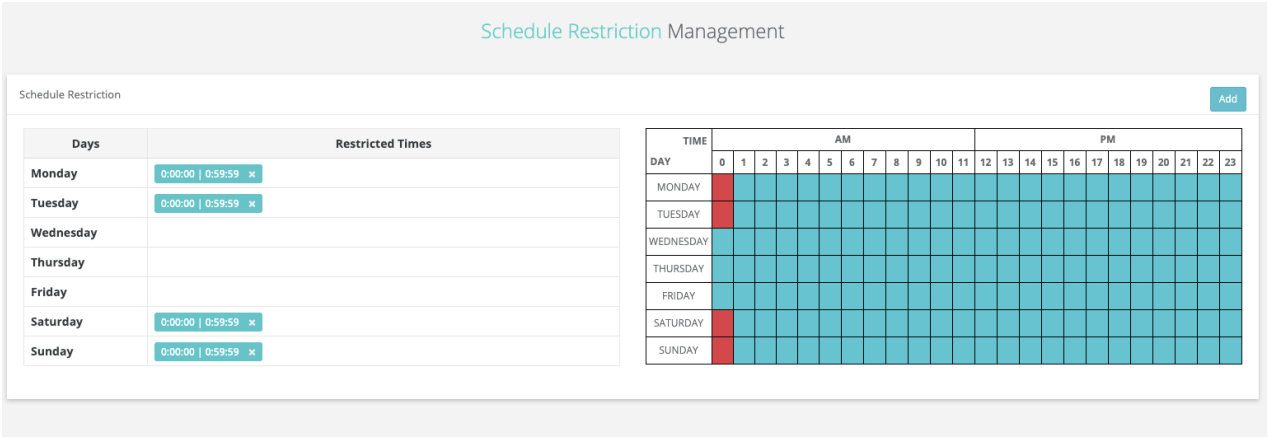
Enable Scheduled Jobs from your environment variable in owl-env.sh:

SCHEDULE_ENABLED = TRUE/FALSE (Default = TRUE)

Limit Scheduler Open Time-slots:

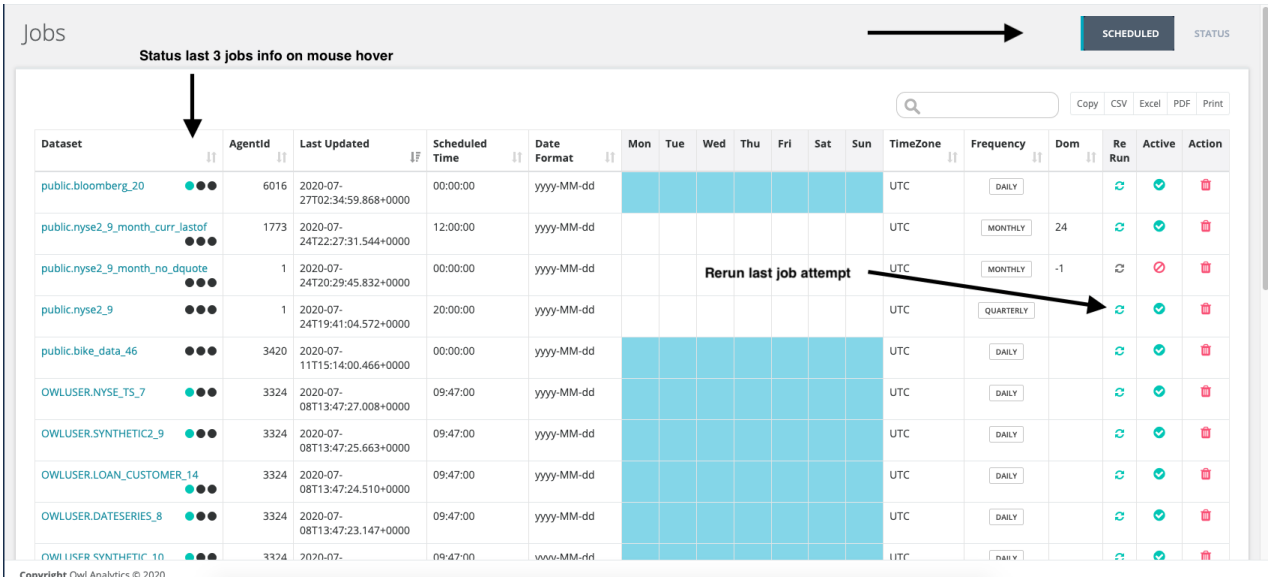
If you don't want automated jobs to be running during business hours, or for a particular day/time on any given day of the week, you can set "off-limit" times so authorized users don't select them when scheduling a job.





View/Re-Run Scheduled Jobs

You can view your schedule jobs from the scheduled tab on the jobs page.



Chapter 10

Collibra DQ Alerts

Email Alerts

Email alerts let you send emails to specified recipients when an alert condition is met for a given dataset.

Setting Up an Email Server Using the WebApp

You can set up the SMTP server using the DQ web application.

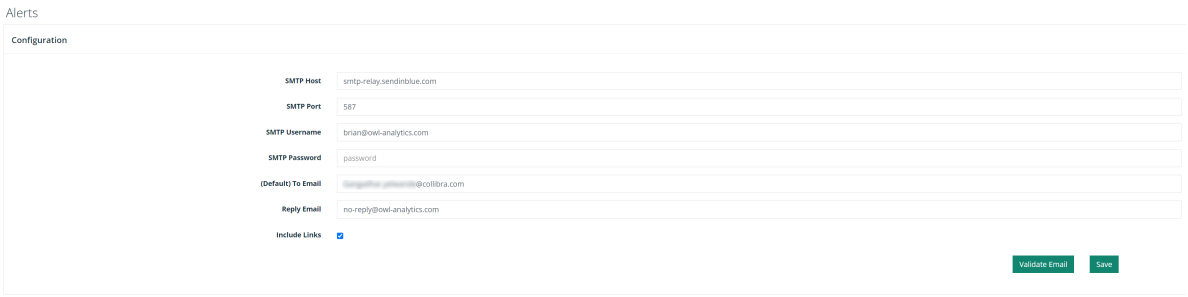
Steps

1. Login to the Collibra DQ instance and click the gear icon in the left navigation pane.
2. Select **Alerts**.
3. Update the following Alerts Configuration fields:

Option	Description
SMTP Host	The name or the IP address of the SMTP server.
SMTP Port	The port used by the SMTP server.
SMTP User-name	The username or the account that is configured on the SMTP server for use.
SMTP Pass-word	The password of the SMTP server username or account.
(Default) To Email	The sender email address.
Reply Email	The reply-to email address.
Include Links	Links to the specific instance(s)/page(s) where the alert came from in the DQ alert email.

4. Click **Save**.

> Note If you select the **Include Links** option, links to the specific instance (s)/page(s) where the alert came from display in the DQ alert email. This allows you to quickly access the environment(s) from which the alert originated.



> Important Be aware that some email servers, such as Microsoft Exchange, require the **SMTP username** and **Reply email** fields to be the same.

For more information about setting up an SMTP email server, go to Set up SMTP.

When you receive a new user request, the request includes the username, tenant name, and link to the instance where the request came from, for quick access.

New User Request

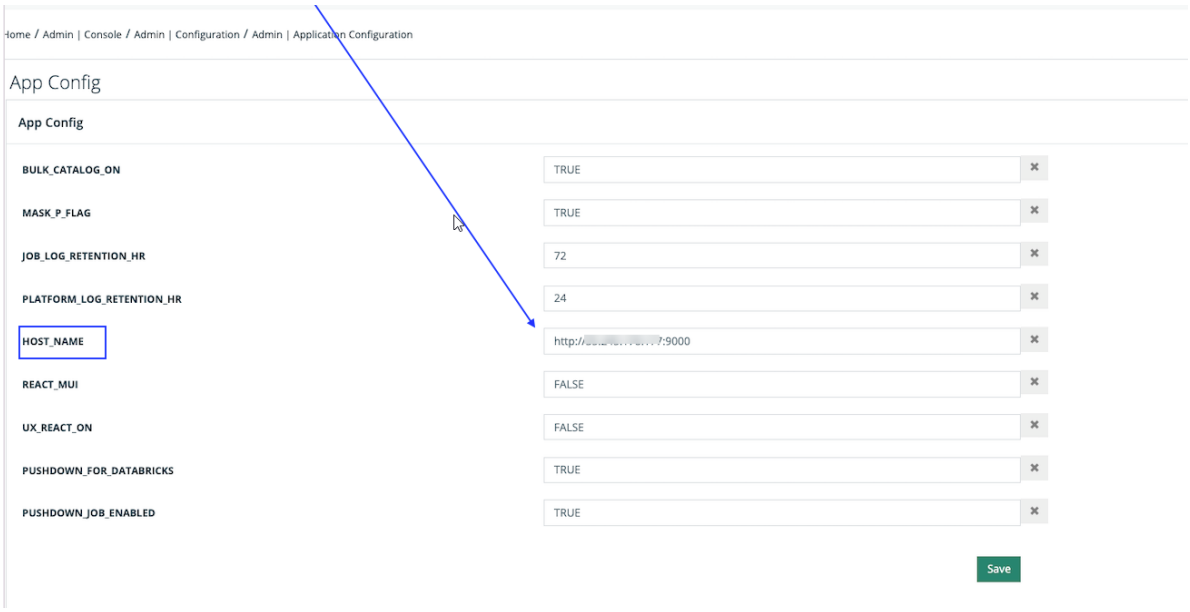
A new user is pending your approval. Please visit the user management link if you wish to approve this user.

User	Test123_5
Tenant Name	validation
Instance Link	https://[redacted]/usermanagement

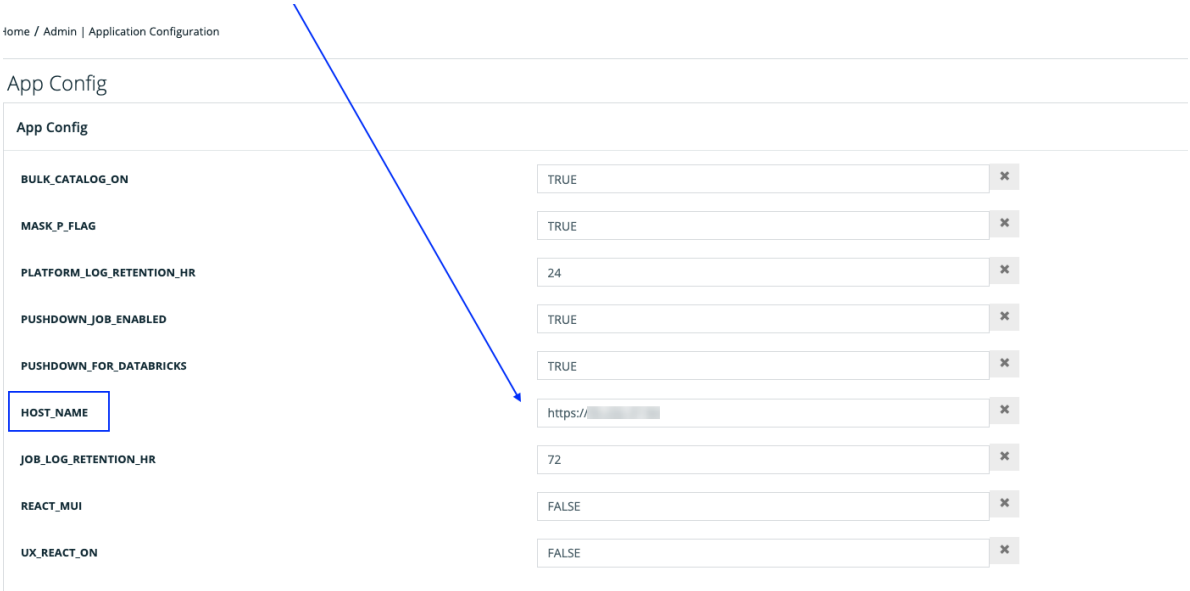
Thank you,
The Collibra Team

> Note If the instance comes from a standalone environment, the port number displays in the URL. If it is from another environment, such as Kubernetes, the port number is not included in the URL. See the following screenshots.

Hostname for Standalone Environment



Hostname for Other Environment



Setting a Condition to Send an Alert

You can set specific conditions so that an email alert is sent to recipients when those conditions are met.

Steps

1. Log in to the Collibra DQ instance.
2. Click the Alerts icon in the left navigation pane. The Alerts page opens.
3. Select **Alert Definitions**.
4. Select a dataset from the Alert Builder searchbar.
5. Enter an **Alert Name**, for example, score less than 75.
6. Define a **Condition**.

Condition Type	Description	Example
Built-in	Conditions that do not require any predefined rules to trigger alerts. To use built-in conditions, enter the condition, an operator, and a value. Available built-in conditions are: ◦ score ◦ row count	score < 75

Condition Type	Description	Example
Rules	Conditions that are tied to datasets as predefined, saved rules. To use rule conditions, enter the rule upon which the alert condition is based. Rule conditions can be configured based on: ◦ SQL Based Rules previously saved to a dataset. ◦ Stat Rules previously saved to a dataset. Tip You can configure dataset-level stat rules as conditions without previously saving them to a dataset. Note Because rule validation does not occur in the Alert Builder, it is not recommended to use rules that are not already saved to a dataset.	$rowCount > 1

7. Optionally enter a **Batch Name**.
8. Enter an **Alert Recipient** as a recipient of email alerts, for example, test.user-@collibra.com.
9. Optionally enter a **Custom Message**, for example, Alert when a score is less than 75.
10. Click **Save**.

For more information on creating rules to use as rule conditions, go to Adding a Rule.

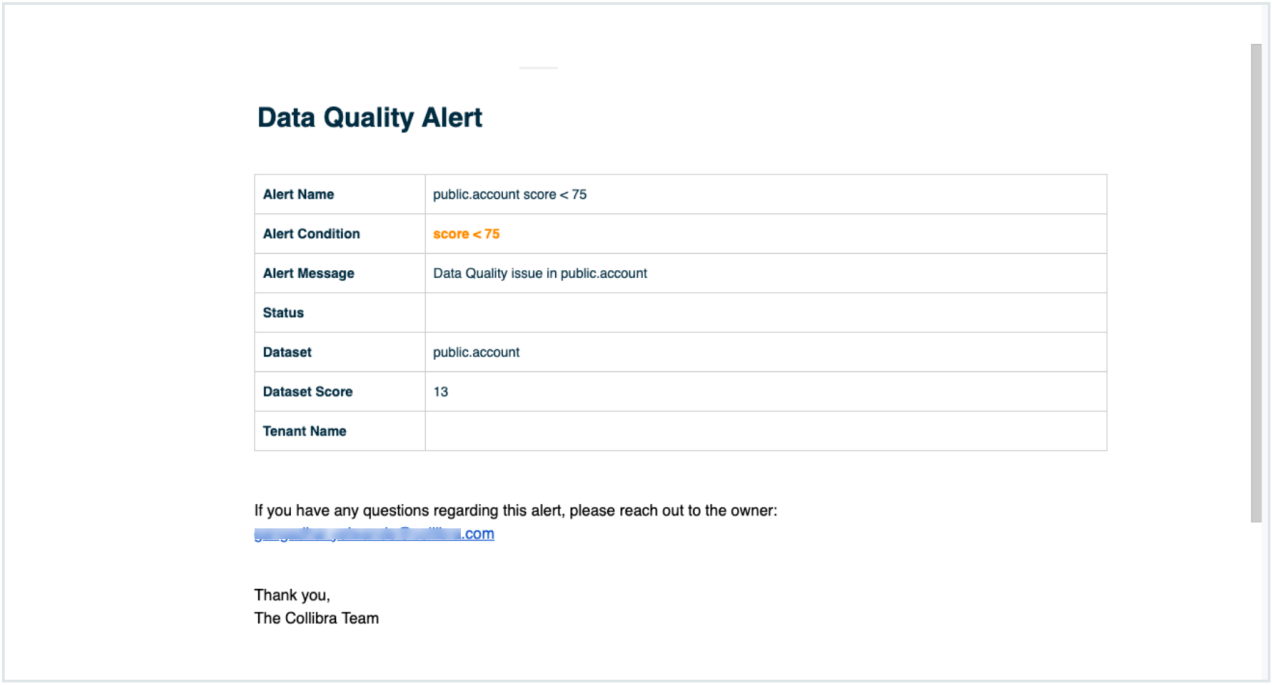
To use the batch name to create a consolidated list of alerts and distribution lists for a set of notifications per dataset, see Email Batch Alerts.

DQ Alerts for Datasets

You can set DQ alerts for datasets so that you are notified based on certain conditions that are triggered on the datasets. Below are some examples of what a dataset email looks like in your inbox.

> Note Ensure that the SMTP server was set up properly and that your email client didn't mark the email as spam.

The following example shows a DQ alert email without links from the instance(s)/page(s).



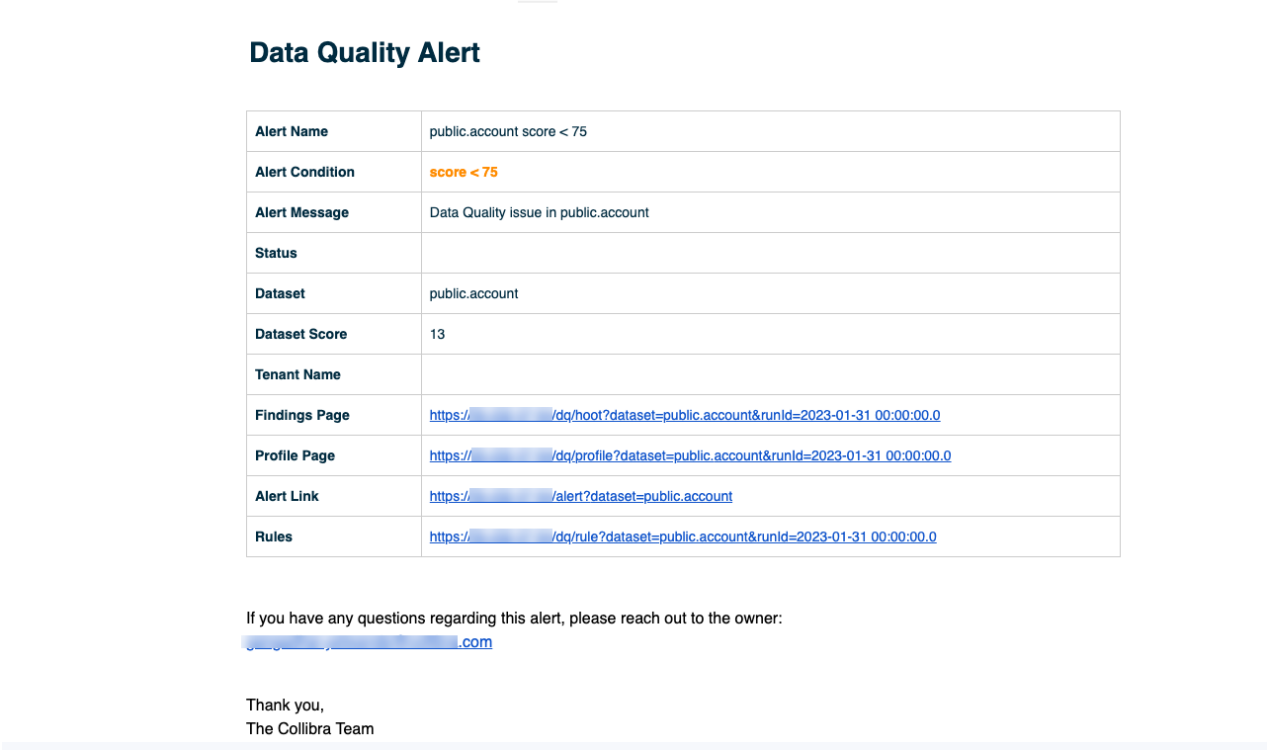
Data Quality Alert

Alert Name	public.account score < 75
Alert Condition	**score < 75**
Alert Message	Data Quality issue in public.account
Status	
Dataset	public.account
Dataset Score	13
Tenant Name	

If you have any questions regarding this alert, please reach out to the owner:
[redacted].com

Thank you,
The Collibra Team

The following example shows a DQ alert email with links from the instance(s)/page(s).



Data Quality Alert

Alert Name	public.account score < 75
Alert Condition	**score < 75**
Alert Message	Data Quality issue in public.account
Status	
Dataset	public.account
Dataset Score	13
Tenant Name	
Findings Page	https://[redacted]/dq/hoot?dataset=public.account&runId=2023-01-31 00:00:00.0
Profile Page	https://[redacted]/dq/profile?dataset=public.account&runId=2023-01-31 00:00:00.0
Alert Link	https://[redacted]/alert?dataset=public.account
Rules	https://[redacted]/dq/rule?dataset=public.account&runId=2023-01-31 00:00:00.0

If you have any questions regarding this alert, please reach out to the owner:
[redacted].com

Thank you,
The Collibra Team

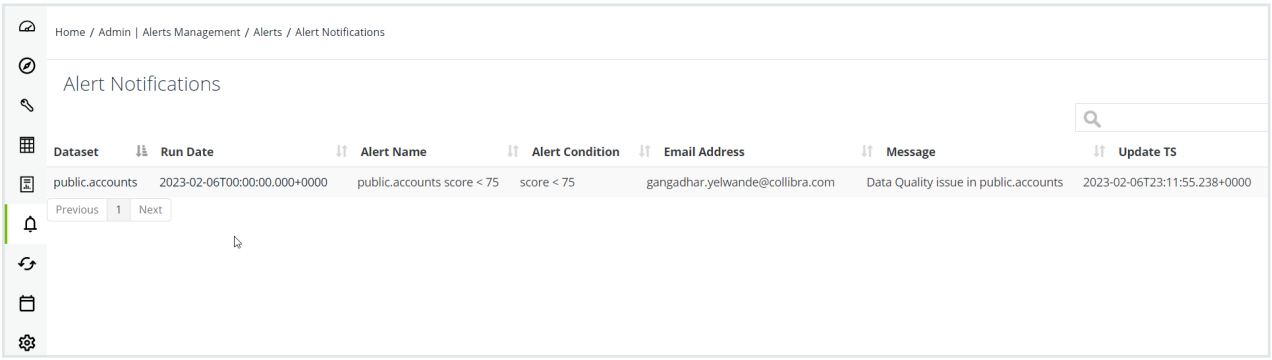
DQ Alerts for Failed Jobs

Another scenario is when the DQ Job fails to run or has an exception and, therefore, never gets the chance to score the data or run the alert condition. This is a failed alert that's automatically sent to the email address based on the Admin/SMTP settings defined in the To Email (Default) fields in the Admin console.

Alert Name:	default.infosec_2 (FAILED) **(FAILED)**
Dataset:	default.infosec_2
Condition:	
Msg:	DQ Job did not complete

Alert Notifications in Web UI

There are also alert notifications in the web UI. This can be helpful to confirm that the email alerts were sent out and who should have received the notifications.



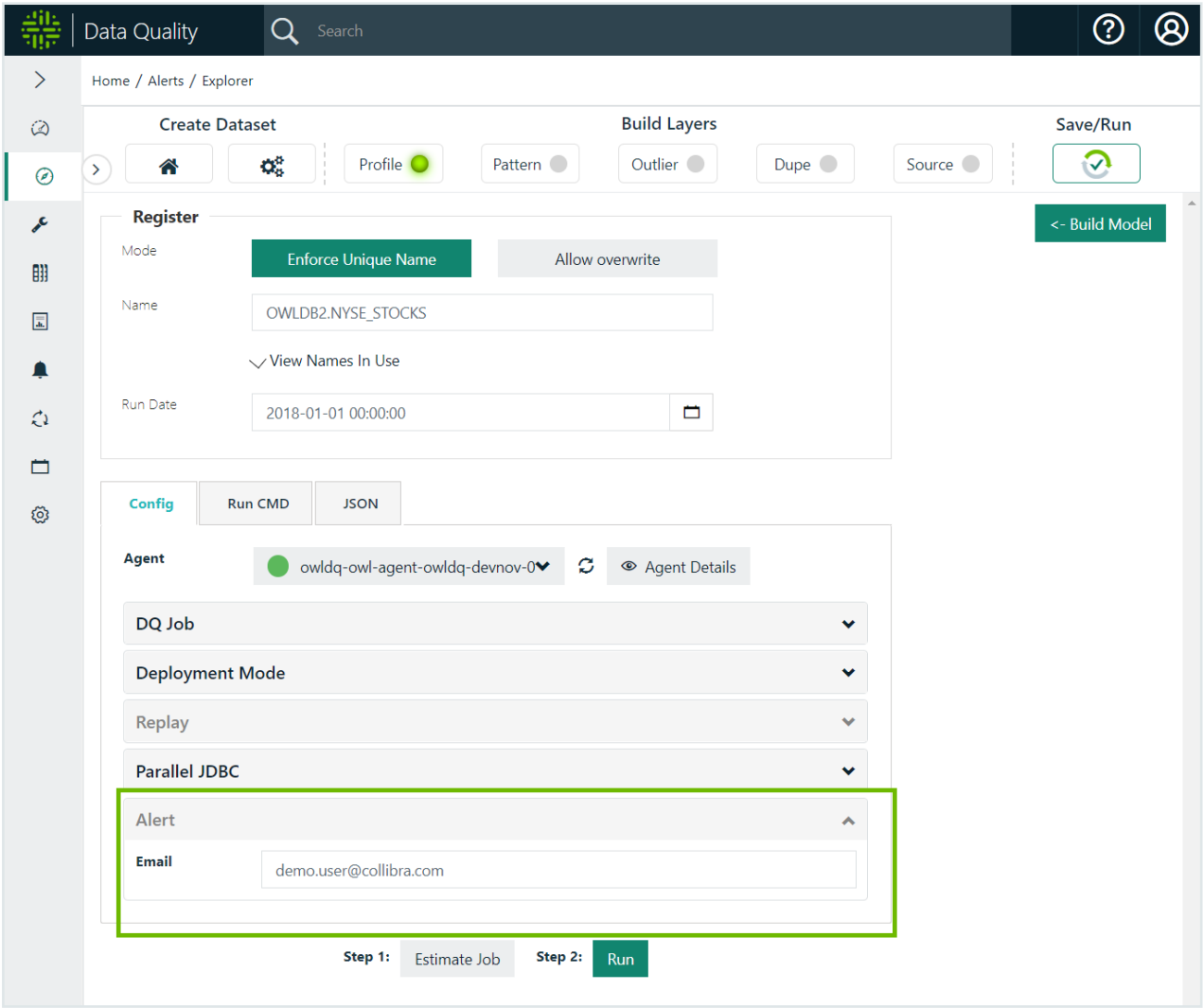
Setting up the Email Server Programmatically

If you are in a notebook or pipeline, you may prefer to use the Scala/Spark API to create the Email Server.

```
    val emailServer = OwlUtils.createEmailServer("smtp-
relay.sendinblue.com", 587)
    emailServer.setUsername("abc@owl-analytics.com")
    emailServer.setPassword("abc")
    emailServer.setCurrentSet(1)
    Util.EmailServerDaoFactory.delete(emailServer)
    Util.EmailServerDaoFactory.insert(emailServer)
```

Setting Up DQ Alerts for Jobs Stuck in Staged Status

Occasionally, jobs become stuck in Staged status after an attempted run. When you create a dataset in Explorer, you can set up alerts from the Config tab by entering an email address in the Email field before the first run of a newly created job.



When an email address is assigned to a dataset, an initial alert is sent 1 hour after a job becomes stuck in Staged. Additional alerts are sent every 24 hours a job remains in Staged, and these alerts persist until the job is no longer stuck. After resolution, previous runs of a job no longer in Staged are marked as Unknown.

Alerts for jobs stuck in Staged include:

- The Job ID.
- The name of the dataset.
- The agent status.
- A descriptive reason for why the job is stuck in staged and possible actions to take for remediation.

> Note If multiple alerts are configured for a particular job, an alert is sent for each one that is configured.

Setting Up SMTP Alerts Without a Username or Password

Some alert settings are configurable without requiring a username or password when you set up an email server.

Steps

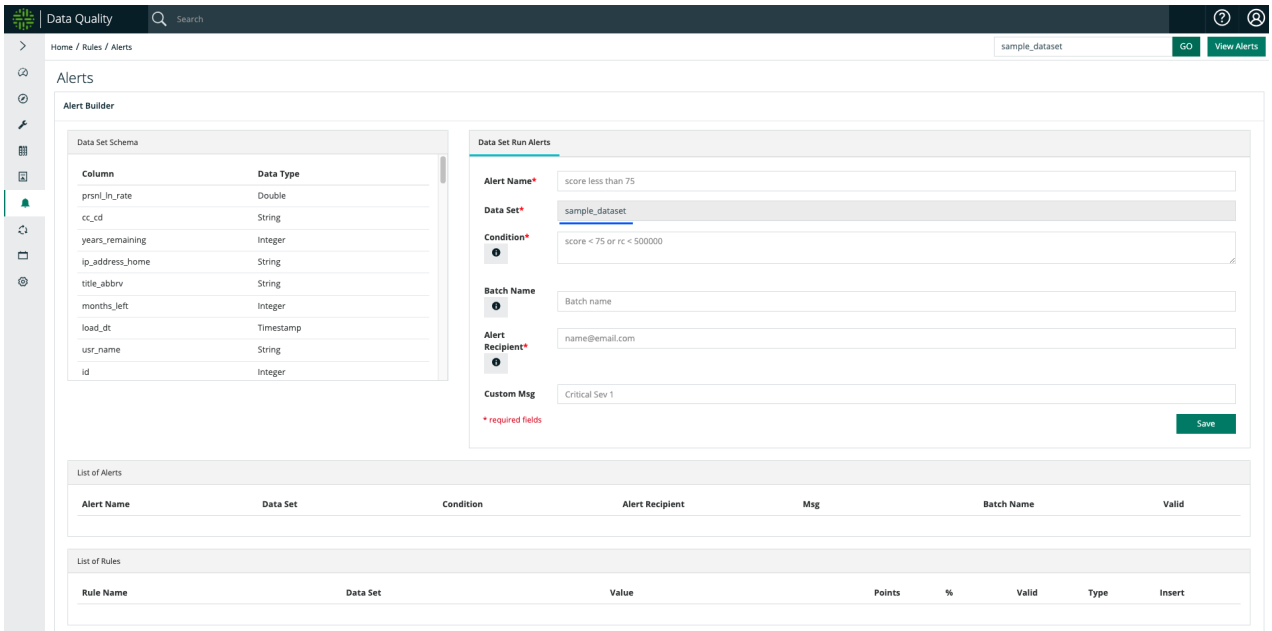
1. Login to the Collibra DQ instance and click the gear icon in the left navigation pane.
2. Click **Settings**.
   The Limit Settings page opens.
3. Select **App Config** in the upper right and then select **Add Custom**.
4. Enter a property in the **name** field and a value in the **value** field.
5. Select **Add**.

Property	Default Value	Description
mail.smtp.auth	True	When set to **True**, the server attempts to authenticate the user using the AUTH command. When set to **False**, username and password authentication are turned off.
mail.smtp.starttls.enable	True	When set to **True** and TLS is supported by the server, this enables the use of the STARTTLS command to switch the connection to a TLS-protected connection before issuing any login commands. When TLS is not supported by your mail server, this property must be set to **False**.

> Note These properties are preset to their default values. For example, mail.stmp.auth is preset to True.

Email Batch Alerts

The Batch Alerts functionality allows you to setup a single alert with multiple recipients. Click the Alerts icon in the left navigation pane and then click Alert Definitions.

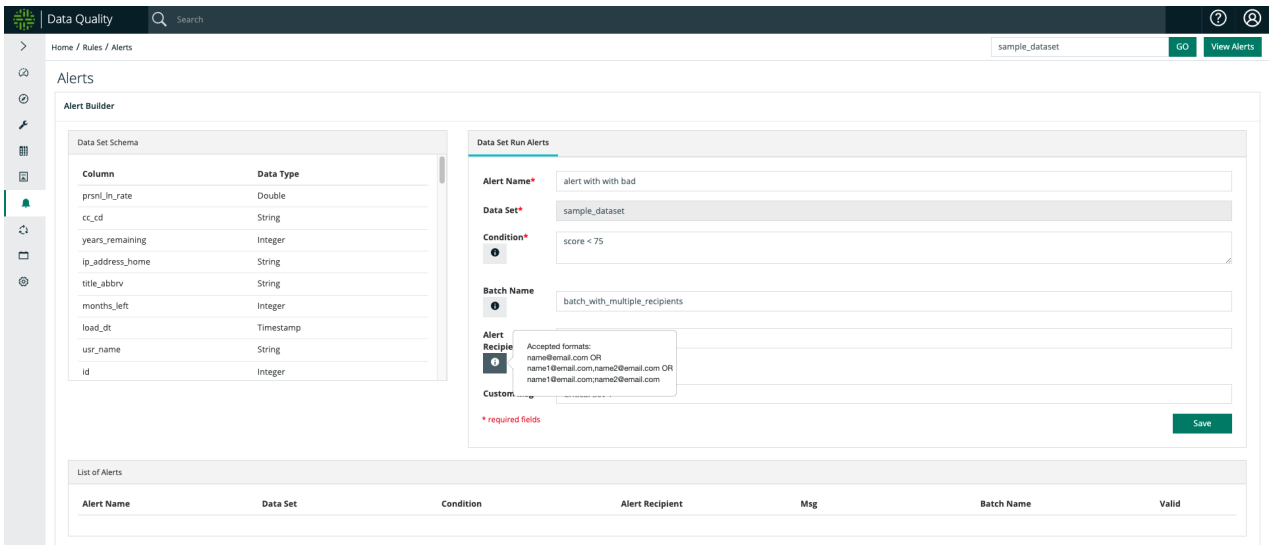


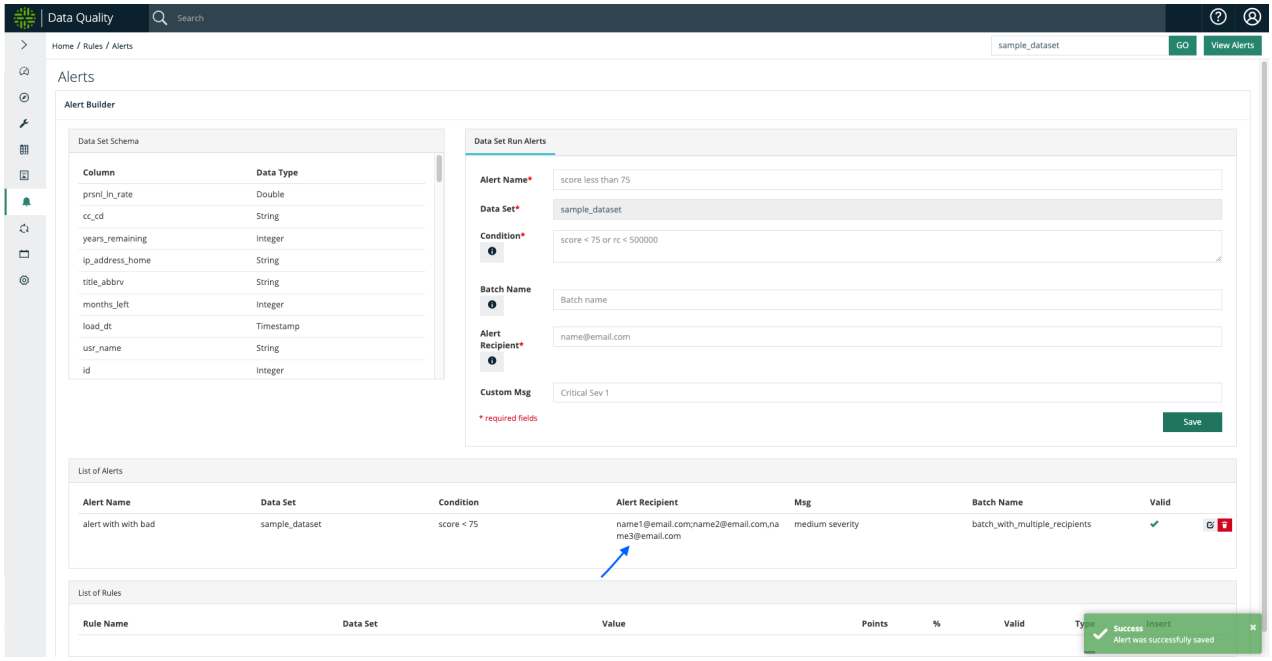
In the Batch Name field, specify a batch name for the multiple recipients. Multiple recipients are specified by comma (,) AND/OR semicolon (;) delimiters.

> Note You can specify either single or multiple recipients.

The accepted formats for email are "name@email.com" OR "name1@email.com,name2@email.com" OR "name1@email.com;name2@email.com".

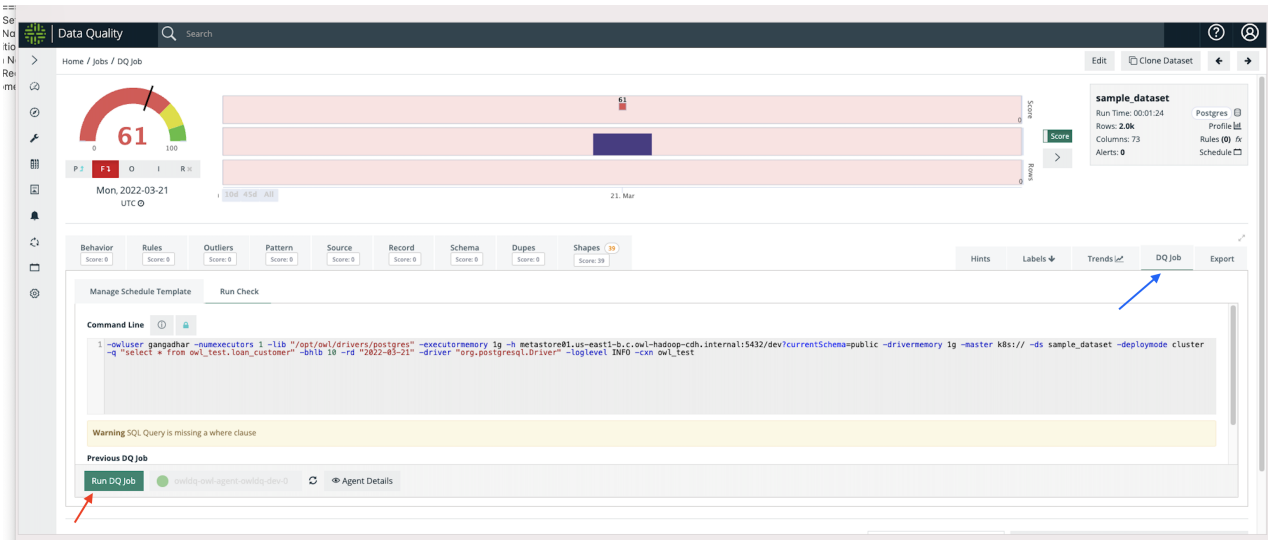
You can update the batch at any time.





After a job runs, it checks the data set with condition email not sent and batch name not empty. If this condition is met, an email is sent to all recipients in the batch.

You can also run the DQ jobs manually from **DQ Job** tab.



Alert for Batch Schedule

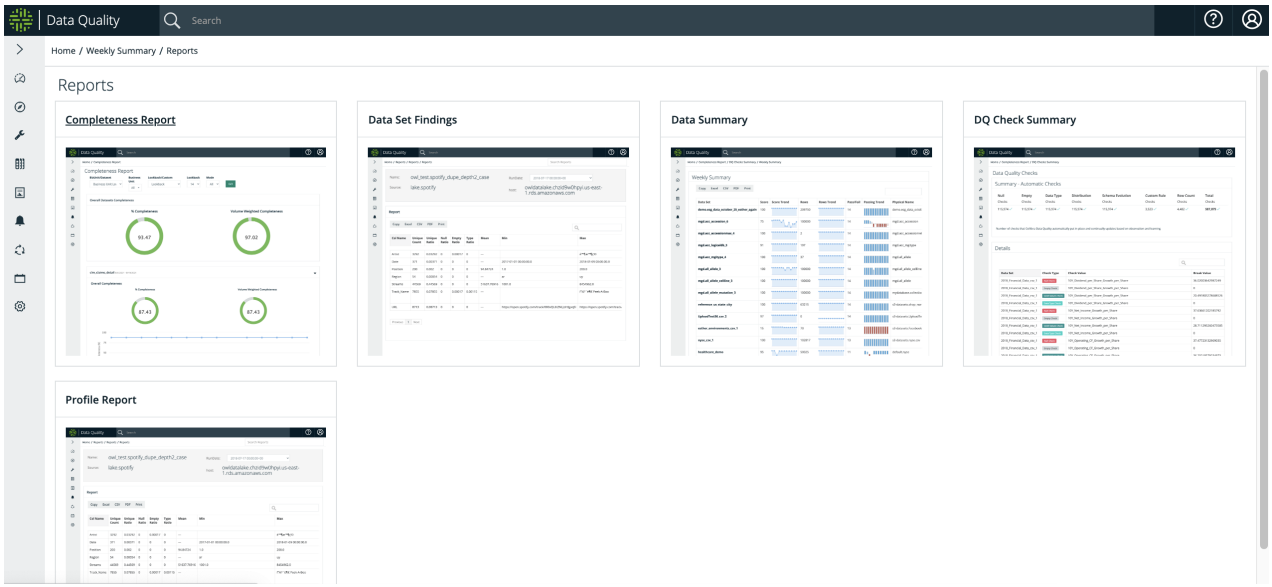
https://dq-docs.collibra.com/scheduler/schedule-owlchecks

Chapter 11

Collibra DQ Reports

Built-In

Collibra DQ provides various out of the box reports that allow you to access insights quickly.



Completeness Report

This section provides information on how to generate a Collibra DQ Completeness Report to determine what percentage of your data is complete.

What is Data Completeness?

Completeness answers the question of what percentage of your data is complete, or "filled in" (i.e., not `EMPTY` or `NULL`). Using this report, you can view the completeness of a column, a collection of tables, a business unit or data set (file or table), or almost any completeness query.

Completeness of your data is important because it shows whether data is inaccurate, invalid, the wrong type, or missing altogether, which can leave you without any data. Sometimes data values can be missing for valid reasons, which requires a better understanding of the context of whether the missing data is bad for business or acceptable.

You can generate the Completeness report from the Reports page.

Steps

Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
The Reports page opens.

1. Click the **Completeness Report** link.
2. From the BizUnit/Dataset drop-down list, select one of the following options:
   - **Business Unit Level**
     i. From the Business Unit drop-down list, select a business unit, or **All**.
     ii. Proceed to step 5.
   - **Dataset Level**
     i. In the Dataset search field, enter a data set for which you want to run the report. You can also enter a partial word to locate data sets in the system.
     ii. Proceed to step 5.
3. From the Lookback/Custom drop-down list, select one of the following options:
   - **Lookback**
     i. From the Lookback drop-down list, select the number of days back to include in the report. You can choose up to 30 days back.
     ii. From the Mode drop-down list, select one of the following options:
       - **All**: includes all jobs in the system.
       - **DRAFT**: includes only the jobs that are in draft mode.
       - **PUBLISHED**: includes the jobs that have been published.
     iii. Proceed to step 6.
   - **Custom Range**
     i. From the RunDate/UpdateTime drop-down list, select one of the following options:
       i. **Run Date**: date/time the data represents.
       ii. **Update Time**: time the DQ job ran.
     ii. In the Date Range field, select a date range by clicking in the from/to fields and choosing the dates using the interactive calendars.
     iii. From the Mode drop-down list, select one of the following options:
       - **All**: includes all reports in the system.
       - **DRAFT**: includes only the reports that are in draft mode.
       - **PUBLISHED**: includes the reports that have been published.

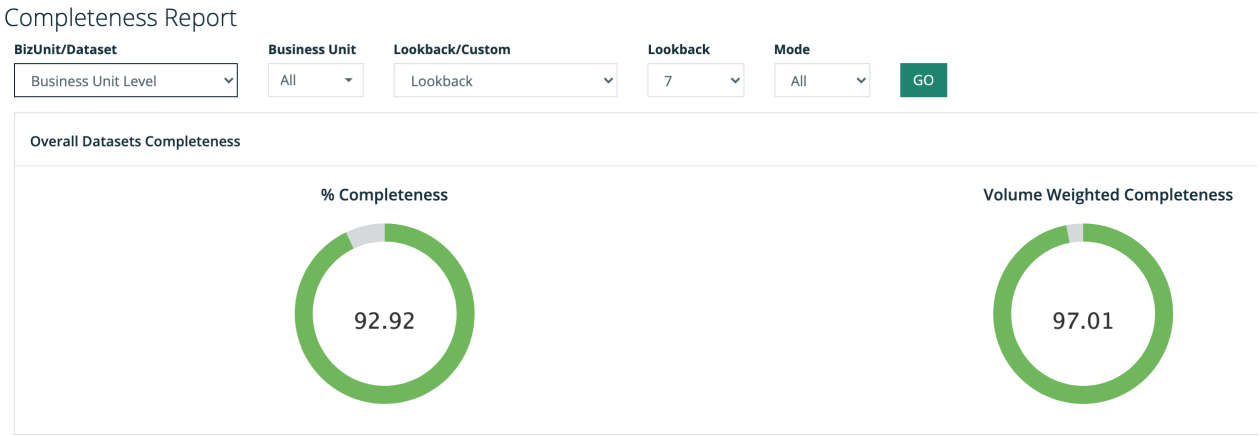
iv. Proceed to step 6.

4. Click **GO**.
   The results display based on your input.

> Note When looking at completeness over time, you should differentiate between the time the DQ job ran (update time) or the date/time the data represents (run date). For example, you could load stock data today but the data loaded was for last week.

All View

The **All** view represents the completeness of data sets throughout the entire Collibra DQ app. The **% Completeness** chart measures all the data, which shows around 93% complete in the following example. The **Volume Weighted Completeness** chart also measures the volume of the data, which shows around 97%.



Column View

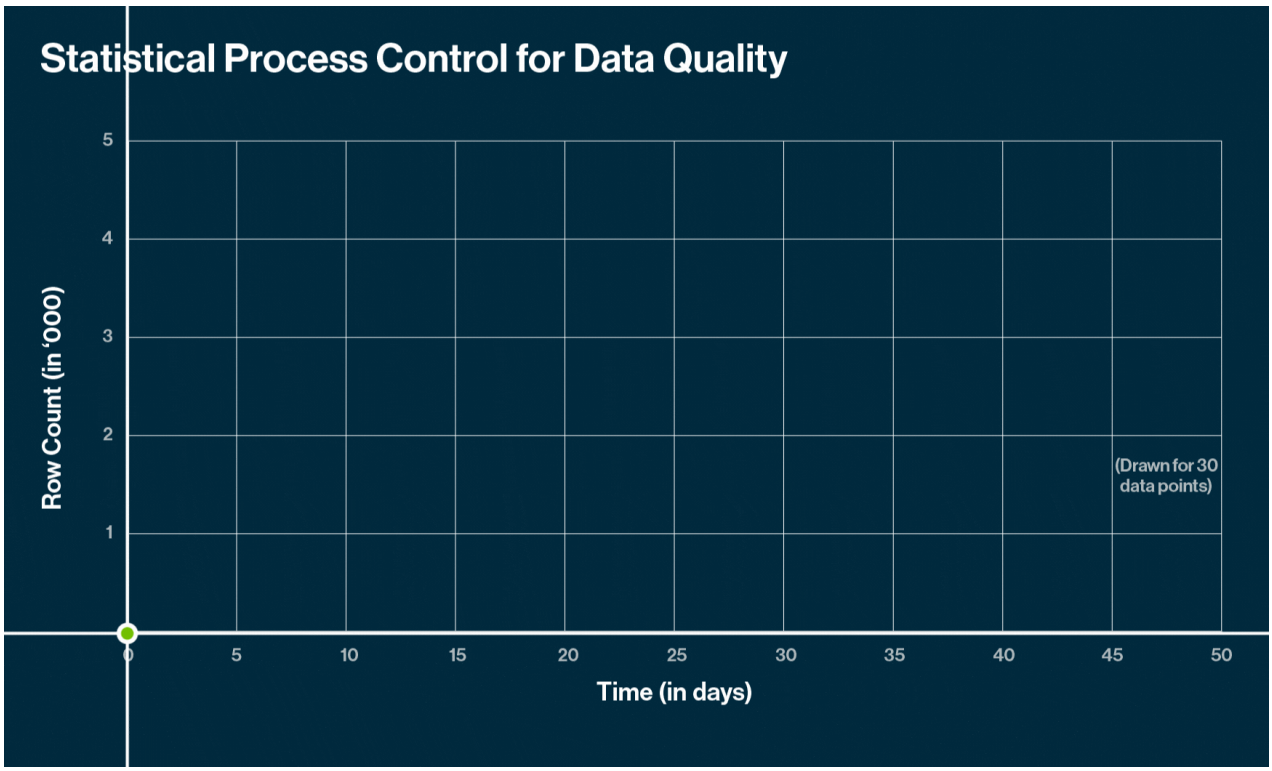
The column view shows the completeness of specific data sets, which makes it easy to see the columns that are least complete and, therefore, of possible concern. The columns range from 0% to 100% complete.

FieldNm	% completeness	volume weighted completeness
Employee_LeaveDate	31.8	31.8
Employee_Office	100	100
Report_Date	100	100
Employee_SSN	100	100
Employee_Email	100	100
Employee_Division	100	100
Employee_AddressStreet	100	100

Behavioral Analytics for Completeness

For a different approach to completeness management, see the Collibra DQ Behavior feature. This approach uses the data itself to create baselines and profiles to understand which completeness issues matter and, therefore, require you to take some kind of action.

To generate statistical process around completeness in the events you are most concerned about, and alert you to a change in slope (a drastic change in completeness), see the Collibra DQ Job Back Run and Profile (automatic) features.



Coverage Report

What is a Data Quality Coverage Report

The Coverage Report provides a view that shows DQ coverage across all your technical data sets (for example, schemas, files, tables, and items in the database) via a donut chart, as well as time-series bar charts that show the DQ run level information aggregated by month.

You can generate a Coverage report from the Reports page.

Steps

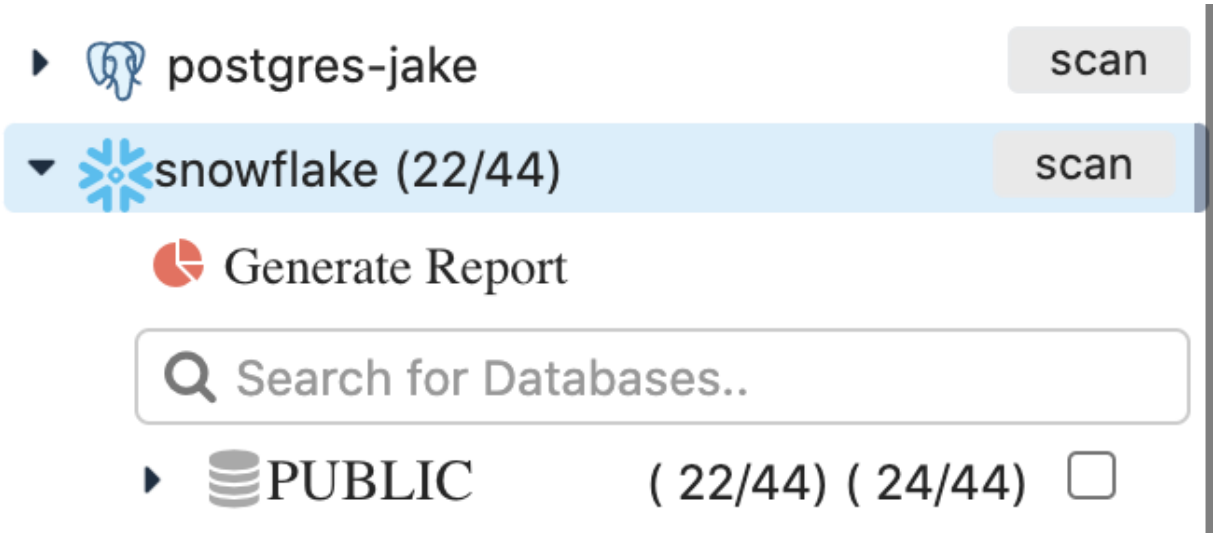
1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. In Connections, select one of the databases, for which you want to generate a report and click **Generate Report**. The Coverage Report displays. The report includes a donut chart showing the percentage of all schemas in the database that are running data quality on them and interactive bar charts showing which data sets are running data quality jobs on a monthly basis.
3. To see how many new data quality jobs have been added, click one of the following bar charts (the green color represents new DQ jobs and the gray are the existing jobs):

View	Description
1m	Shows new and existing jobs after one month.
3m	Shows new and existing jobs after three months.
6m	Shows new and existing jobs after six months.
YTD	Shows new and existing jobs year-to-date.
1y	Shows new and existing jobs after one year.

View	Description
All	Shows new and existing jobs for the entire range of months.

4. To generate a Coverage Report for a specific schema, expand the schema and click **Generate Report**. The Coverage Report displays for that schema. You can click into the charts to see specific information for DQ coverage for this schema.

   In the following screenshot, `(22/44)` next to the PUBLIC schema represents 22 out of 44 tables that have DQ jobs running on them and `(24/44)` represents 24 out of 44 jobs that have been cataloged, meaning they are registered and have metadata.



Data Set Findings

What is the Data Set Findings Report?

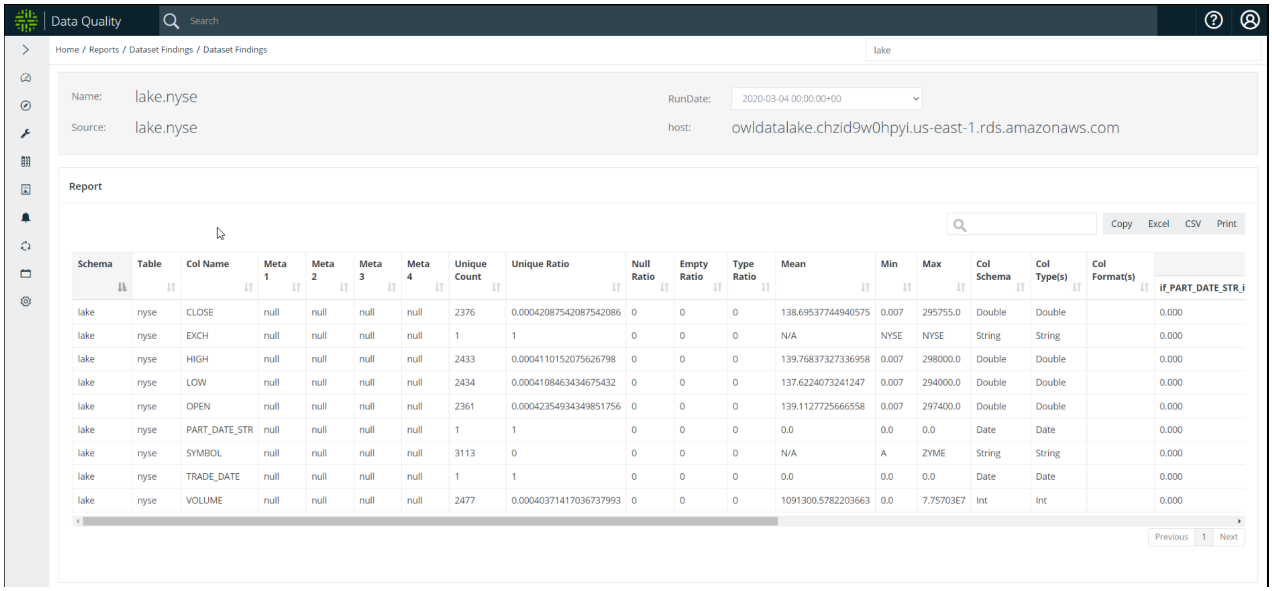
The Data Quality Data Set Findings Report allows you to search for a particular data set and generate, view, copy, print, and export the report to an Excel or CSV file format.

> Note As of the 2022.08 release, PDF is no longer a supported file format for exporting and printing reports. These functions are now restricted to the CSV file format only.

Steps

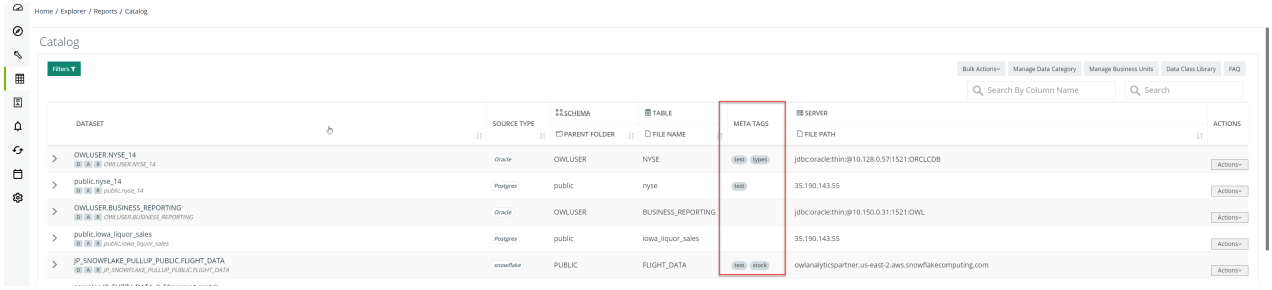
1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. Select the **Data Set Findings** tile.
3. In the search bar in the upper right corner, enter the name of your target data set.
   A list of available data sets populates in the drop-down menu.
4. Select the **data set** you want to view and export results.
   The Data Set Findings page displays for your data set.
   The report includes the name, source, RunDate, and host of your data set, as well as columns that highlight specific data about it.
5. Select the RunDate from the drop-down list to display the data in the table for that particular RunDate.
6. Toggle the icon at the top of the column to sort the data that displays in the columns in ascending or descending order.
7. Filter the report results by entering information in the search bar. For example, if you enter a number in the search field, any report result that includes the number displays.
8. Click **Copy**, **Excel**, **CSV**, or **Print** at the top right of the columns to copy, print, or export your reports.
9. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located bottom-right of the columns.

The following screenshot is an example of a Data Set Findings Report for the lake.nyse data set.



Meta fields can be used to classify data sets. You can assign Meta tags by clicking **Edit** in the Actions dropdown for a data set in the DQ Catalog. See Collibra DQ Catalog for more information.

The meta tags you added populate in the META TAGS field..



Summary Reports

Collibra Data Quality Summary reports provide information about your data sets rolled up in high level summaries. These reports include the Weekly Summary Report and Data Quality Checks Report.

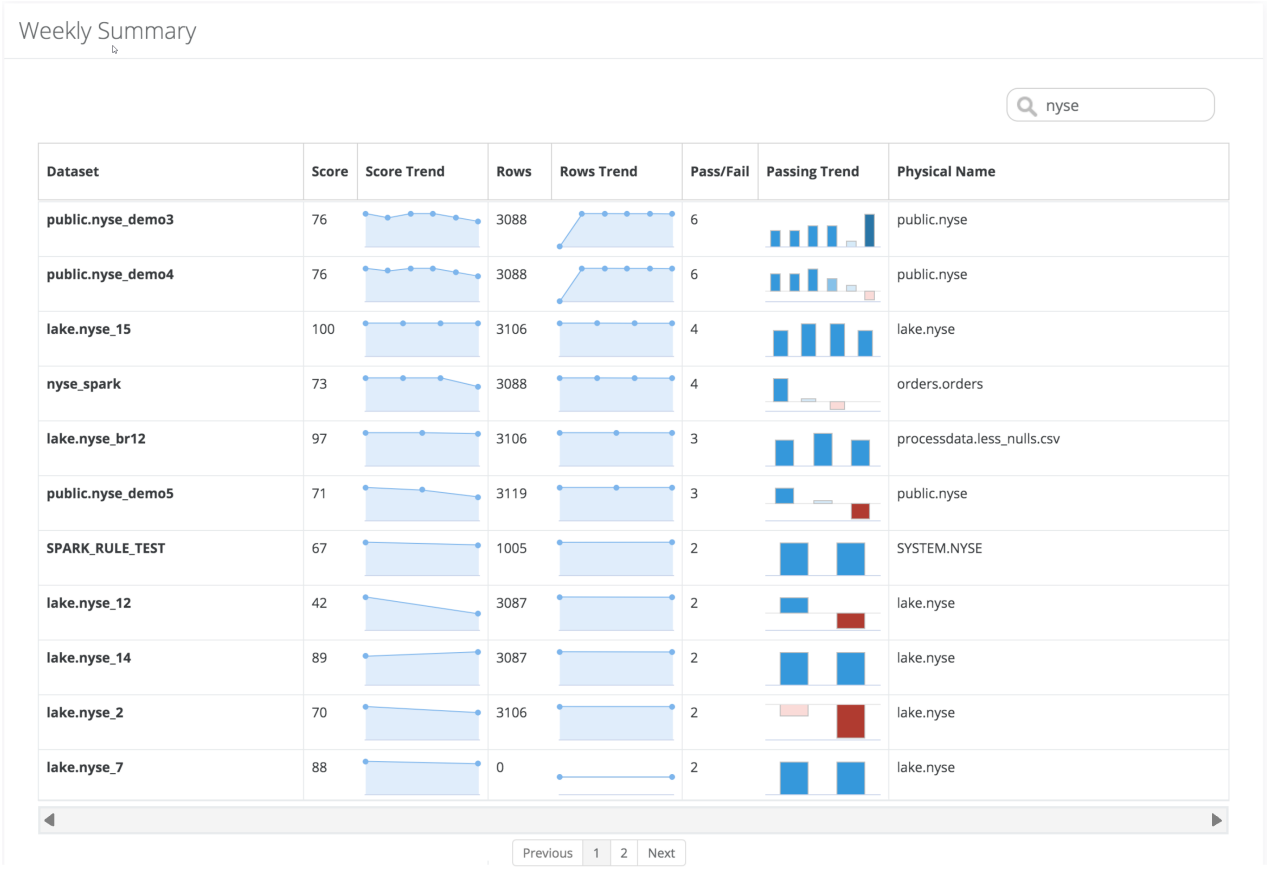
> Note These reports provide a look at the most simplistic form of the rule. However, this view does not associate itself with a run's score.

What is the Weekly Summary Report?

When you operate a large data lake or several large data environments, it's helpful to have a way to report across different dimensions at an executive summary level. You may want to know the health or coverage per line of business, department or tenet, or per database.

The Collibra Data Quality Weekly Summary Report automatically aggregates a simple series of the high level trends for each data set, which allows you to see the DQ scores and trends, as well as row counts and passing runs in a weekly report. You can copy, export, and print this report to an Excel or CSV file format.

> Note As of the 2022.08 release, PDF is no longer a supported file format for exporting and printing reports. These functions are now restricted to the CSV file format only.



Steps

To generate a Weekly Summary report, follow these steps.

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. Select the **Data Summary** tile.
   The Weekly Summary report displays.

The following table describes the report columns.

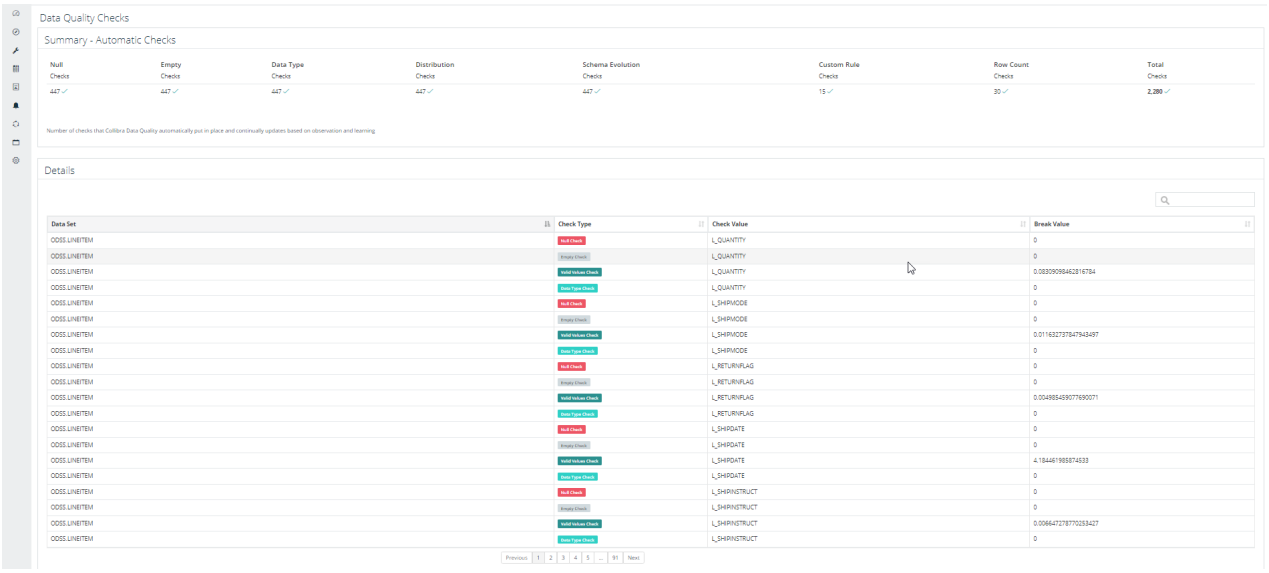
Column	Description
Dataset	The data sets scanned over one week.
Score	The average data quality score of a data set over one week.
Score Trend	A line graph representation of data quality scores over one week. Hover over the blue dots to see the score trend.
Rows	The total number of rows of a particular data set scanned over one week.
Rows Trend	A line graph representation of row count over one week. Hover over the blue dots to see the rows trend.
Pass/Fail	Total number of DQ scans, whether they pass or fail.
Passing Trend	A histogram of DQ scans that passed (blue) or failed (red) over one week. Hover over the blue dots to see the passing trend.
Table/File Name	The name of the table or file in use.

3. Toggle the icon at the top of the column to sort the data that displays in the columns in ascending or descending order.
4. Filter the report results by entering information in the search bar. For example, if you enter a number in the search field, any report result that includes that number displays.
5. Click **Copy**, **Excel**, **CSV**, or **Print** at the top right of the columns to copy, export, and print the report.
6. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located at the bottom of the columns.

What is the Data Quality Checks Report?

The Data Quality Checks Report provides a number of automatic checks of your data sets that are continually updated, based on observation and learning. This report includes the type of check performed and the check value and break value per data set. The top portion of this report rolls up the highest results of the checks.

> Note As of the 2022.08 release, PDF is no longer a supported file format for exporting and printing reports. These functions are now restricted to the CSV file format only.



> Note The Schema Evolution, located in the automatic checks summary at the top of the report, is a schema finding that shows how the schema has evolved from the last run.

Steps

To see the Data Quality Checks report, follow these steps.

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. Select the **DQ Check Summary** tile.
   The Data Quality Checks report displays.
   This report includes a summary of the number of checks that Collibra DQ automatically ran at the top of the report and the details, which are described in the following table.

Column	Description
Data Set	The name of the data set.
Check Type	The type of check that DQ ran.
Check Value	The value associated with this check.
Break Value	The number of points associated with a rule break, which are then subtracted from a data quality score.

3. Toggle the icon at the top of the column to sort the data that displays in the columns in ascending or descending order.
4. Filter the report results by entering information in the search bar. For example, if you enter a number in the search field, any report result that includes that number displays.

5. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located at the bottom of the columns.

Profile Report

What is the Profile Report

The Profile Report allows you to search for a particular data set and generate, view, copy, print, and export the report to an Excel or CSV file format.

> Note As of the 2022.08 release, PDF is no longer a supported file format for exporting and printing reports. These functions are now restricted to the CSV file format only.

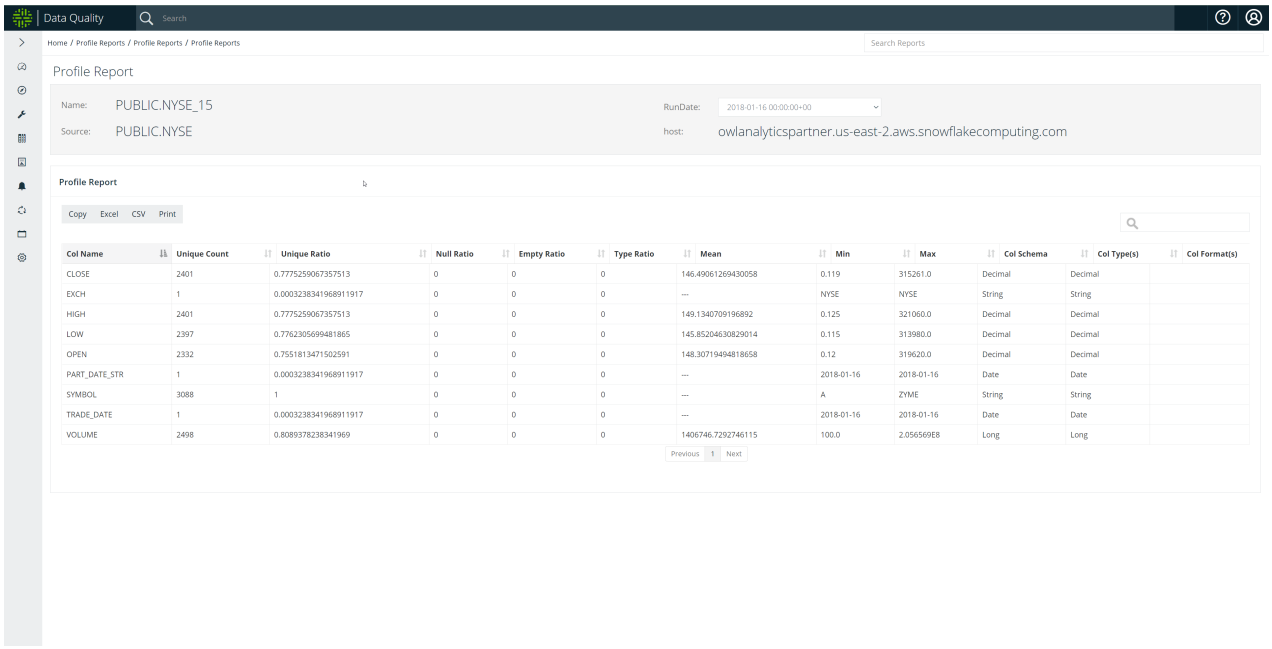
Steps

To generate a Profile Report, follow these steps.

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. Select the **Profile Report** tile.
3. In the search bar in the upper right corner, enter the name of your target data set.
   A list of available data sets populates in the drop-down menu.
4. Select the **data set** for which you want to view and export results.
   The Profile Report page displays for your data set.
   The report includes the name, source, RunDate, and host of your data set, as well as columns that highlight specific data about it.
5. Select the RunDate from the drop-down list to display the data in the table for that particular RunDate.
6. Toggle the icon at the top of the column to sort the data that displays in the columns in ascending or descending order.
7. Filter the report results by entering information in the search bar. For example, if you enter a number in the search field, any report result that includes the number displays.

8. Click **Copy**, **Excel**, **CSV**, or **Print** at the top left of the columns to copy, print, or export your reports.
9. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located at the bottom of the columns.

The following screenshot is an example of a Profile Report.



Missing Jobs Report

What is a Missing Jobs Report?

The Missing Jobs report provides a view of Collibra DQ jobs that were expected but didn't run on schedule.

You can generate a Missing Jobs report from the Reports page.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.

2. From the Reports page, select the **Missing Jobs Report** tile.
3. For x-axis, select one of the following options from the dropdown:
    - **count**
    - **updt_ts**
4. For y-axis, select one or more of the following options:
    - **count**
    - **updt_ts**
5. For the type of chart you want to display, select one of the following options from the dropdown:
    - **Line**
    - **Area**
    - **Column**
    - **Scatter**
6. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a Missing Jobs report.

```
SELECT jsa.updt_ts::date AS updt_ts,
        jsa.dataset,
        bus.name AS business_unit,
        js.freq,
        js.active,
        js.mon,
        js.tue,
        js.wed,
        js.thu,
        js.fri,
        js.sat,
        js.sun
    FROM job_schedule_attempt jsa
    JOIN job_schedule js ON js.dataset::text = jsa.dataset::text
    LEFT JOIN dataset_scan ds ON ds.dataset::text = jsa.data-
set::text AND ds.updt_ts::date = jsa.updt_ts::date
    LEFT JOIN business_unit_to_dataset bu ON jsa.dataset::text =
bu.dataset::text
    LEFT JOIN business_units bus ON bus.id = bu.id
    WHERE ds.updt_ts IS NULL AND jsa.updt_ts >= (now() - '20
days'::interval)
    ORDER BY (jsa.updt_ts::date) DESC
```

Hardware Usage Report

What is a Hardware Usage Report?

The Hardware Usage report provides a view of the Collibra DQ jobs that have consumed the most hardware.

You can generate a Hardware Usage report from the Reports page.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, click the **Hardware Usage Report** tile.

3. For x-axis, select one of the following options from the dropdown:
    - **num_executors**
    - **executor_memory**
    - **total_memory**
    - **total_cores**
    - **executor_cores**
    - **avg_row_count**
    - **dataset**
4. For y-axis, select one or more of the following options:
    - **num_executors**
    - **executor_memory**
    - **total_memory**
    - **total_cores**
    - **executor_cores**
    - **avg_row_count**
    - **dataset**
5. For the type of chart you want to display, select one of the following options from the drop-down:
    - **Line**
    - **Area**
    - **Column**
    - **Scatter**
6. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a Hardware Usage report.

```
with most_current_dataset_scan as ( select dataset, avg(rc)::Int
as avg_row_count from dataset_scan group by dataset )

     select opt.dataset, num_executors, executor_cores,
executor_memory, (executor_cores * num_executors) total_cores,
            (NULLIF(regexp_replace(executor_memory, '\D', '',
'g'), '')::numeric * num_executors) as total_memory,
            ds.avg_row_count
     from opt_spark opt, most_current_dataset_scan ds
     where opt.dataset = ds.dataset
     order by total_cores desc limit 75
```

Oversized Job Report

What is an Oversized Job Report?

The Oversized Job report provides a view of the Collibra DQ jobs that have more hardware assigned than is available to run. For example, say you have a scenario where 20 jobs all request 100 servers to kick off at the same time, but 2,000 servers are not available to run. The Oversized Job report identifies those 20 jobs, which are oversized. Those jobs can now be resized to the optimal amount of executors, cores, ram, etc.

Steps

To generate an Oversized Job report, follow these steps.

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, click the **Oversized Job Report** tile.
3. For x-axis, select one of the following options from the drop-down list:
   - **dataset**
   - **reason**
   - **recommended_total_memory**
   - **recommended_total_cores**
   - **recommended_num_executors**

- recommended_executor_memory
- num_executors
- executor_cores
- executor_memory
- total_cores
- total_memory
- avg_row_count
- column_count
- avg_total_time_in_minutes
- cell_count

4. For y-axis, select one or more of the following options:
    - dataset
    - reason
    - recommended_total_memory
    - recommended_total_cores
    - recommended_num_executors
    - recommended_executor_memory
    - num_executors
    - executor_cores
    - executor_memory
    - total_cores
    - total_memory
    - avg_row_count
    - column_count
    - avg_total_time_in_minutes
    - cell_count
5. For the type of chart you want to display, select one of the following options from the drop-down:
    - Line
    - Area
    - Column
    - Scatter
6. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL queries for an Oversized Job report.

```
DROP VIEW IF EXISTS report_oversized_jobs;
CREATE OR REPLACE VIEW report_oversized_jobs AS
(
WITH most_current_dataset_scan AS (
    SELECT dataset_scan.dataset,
           avg(dataset_scan.rc)::integer AS avg_row_count
    FROM dataset_scan
    GROUP BY dataset_scan.dataset
),
     dataset_schema_col AS (
         SELECT dataset_schema.dataset,
                count(*) AS column_count
         FROM dataset_schema
         GROUP BY dataset_schema.dataset
     ),
     dataset_activity_time AS (
         SELECT dataset_activity.dataset,
                round(avg(dataset_activity.total_time_in_
minutes), 2) AS avg_total_time_in_minutes
         FROM dataset_activity
         GROUP BY dataset_activity.dataset
     ),
     highest_hardware_usage AS (
         SELECT opt.dataset,
                opt.num_executors,
                opt.executor_cores,
                opt.executor_memory,
                opt.executor_cores * opt.num_executors AS total_
cores,
                NULLIF(regexp_replace(opt.executor_memory::text,
'\D'::text, ''::text, 'g'::text), ''::text)::numeric * opt.num_
executors::numeric AS total_memory,
                ds.avg_row_count,
                sch.column_count,
                dat.avg_total_time_in_minutes,
                ds.avg_row_count * sch.column_count AS cell_
count
         FROM opt_spark opt,
              most_current_dataset_scan ds,
              dataset_schema_col sch,
              dataset_activity_time dat
         WHERE opt.dataset::text = ds.dataset::text AND sch.data-
set::text = ds.dataset::text AND dat.dataset::text = ds.data-
set::text AND opt.executor_memory IS NOT NULL AND opt.executor_
memory::text NOT LIKE ''::text
         ORDER BY (opt.executor_cores * opt.num_executors) DESC
     )
```

```
SELECT
    highest_hardware_usage.dataset AS dataset,
    CASE
        WHEN (highest_hardware_usage.cell_count::numeric /
highest_hardware_usage.total_memory) > 20000000::numeric THEN
'Too Much Memory'::text
        WHEN (highest_hardware_usage.cell_count / highest_hard-
ware_usage.total_cores) > 190000000 THEN 'Too Many Cores'::text
        ELSE 'Too Much Memory & Cores'::text
        END AS reason,
    GREATEST(highest_hardware_usage.cell_count / 20000000, 1::-
decimal) AS recommended_total_memory,
    GREATEST(highest_hardware_usage.cell_count / 119000000, 1::-
decimal) AS recommended_total_cores,
    GREATEST(highest_hardware_usage.cell_count / 119000000 / 2,
1::decimal) AS recommended_num_executors,
    GREATEST(highest_hardware_usage.cell_count / 20000000, 1::-
decimal) / GREATEST(highest_hardware_usage.cell_count /
119000000 / 2, 1::decimal) AS recommended_executor_memory,
    highest_hardware_usage.num_executors,
    highest_hardware_usage.executor_cores,
    highest_hardware_usage.executor_memory,
    highest_hardware_usage.total_cores,
    highest_hardware_usage.total_memory,
    highest_hardware_usage.avg_row_count,
    highest_hardware_usage.column_count,
    highest_hardware_usage.avg_total_time_in_minutes,
    highest_hardware_usage.cell_count
FROM highest_hardware_usage
WHERE (
            (highest_hardware_usage.cell_count::numeric /
highest_hardware_usage.total_memory) < 20000000::numeric
        OR (highest_hardware_usage.cell_count / highest_hard-
ware_usage.total_cores) < 119000000)
  AND highest_hardware_usage.executor_cores > 1
  AND highest_hardware_usage.num_executors > 1
    );
```

Observability Score Roll-Up Report

What is an Observability Score Roll-Up Report?

The Observability Score Roll-Up report provides aggregated scores of all adaptive rules (includes all data sets and all columns), as well as averages passing and breaking for all

columns over 30 days. The scores are aggregated by check type and dimension.

You can generate an Observability Score Roll-Up report from the Reports page.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, click the **Observability Score Roll-Up Report** tile.
3. For x-axis, select one of the following options from the drop-down list:
   - **check_type**
   - **dimension**
   - **avg_observability_score**
4. For y-axis, select one or more of the following options:
   - **check_type**
   - **dimension**
   - **avg_observability_score**
5. For the type of chart you want to display, select one of the following options from the drop-down:
   - **Line**
   - **Area**
   - **Column**
   - **Scatter**
6. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for an Observability Score Roll-Up report.

```
with a as ( select dataset, col_nm from dataset_schema ),
     b as ( select distinct dataset, 'Row Count', 'ROW_COUNT',
'AVAILABILITY' from dataset_scan where updt_ts >= (NOW() -
INTERVAL '30 DAY') and updt_ts <= NOW()),
     c as ( select * FROM
               (VALUES
                    ('NULL', 'COMPLETENESS'),
                    ('EMPTY', 'COMPLETENESS'),
                    ('TYPE', 'SCHEMA'),
                    ('CARDINALITY', 'DISTRIBUTION'),
                    ('MIN', 'CONFORMITY'),
                    ('MAX', 'CONFORMITY'),
                    ('MEAN', 'CONFORMITY')) as t (type, dimension)
),
     d as ( select * from a,c  ),
     e as ( select * from d UNION (select * from b)),
     f as ( select dataset as be_dataset, be_score, field,
issue_type, run_id from behavior ),
     g as ( select * from owl_catalog ),
     h as ( select distinct dataset, run_id, rc from dataset_
scan where updt_ts >= (NOW() - INTERVAL '30 DAY') and updt_ts <
NOW()),
     i as ( select e.dataset, col_nm, type, dimension, run_id,
rc from e, h where e.dataset = h.dataset ),
     j as ( select distinct dataset, col_nm, col_semantic
from  dataset_schema ),
     k as ( select * from business_unit_to_dataset ),
     l as ( select * from business_units ),
     m as (
            select i.col_nm, type,
                   case when be_score is null then 100 else be_
score end as score, i.dimension,
                   i.run_id, rc, be_score, issue_type, i.data-
set, g.source_name, g.source, g.table_nm, g.run_mode, g.data_
concept_id, j.col_semantic,
                   l.name
            from i
                LEFT join f on f.be_dataset = i.dataset and
f.field = i.col_nm and f.issue_type = i.type and f.run_id =
i.run_id
                INNER JOIN g on i.dataset = g.dataset
                LEFT JOIN j on i.dataset = j.dataset and i.col_
nm = j.col_nm
                LEFT JOIN k on i.dataset = k.dataset
                LEFT JOIN l on l.id = k.id
            )
    select avg(score) as avg_observability_score, type as check_
type, dimension from m group by type, dimension order by
dimension
```

Rules Passing Fraction Roll-Up Report

What is a Rules Passing Fraction Roll-Up Report?

The Rules Passing Fraction Roll-Up report provides the passing and total rows scanned for user-defined rules over the past 30 days, which are aggregated by dimensions.

Steps

To generate a Rules Passing Fraction Roll-Up report, follow these steps.

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, click the **Rules Passing Fraction Roll-Up Report** tile.
3. For x-axis, select one of the following options from the drop-down list:
   - **avg_percent_rows_passing**
   - **dimension**
4. For y-axis, select one or more of the following options:
   - **dimension**
   - **avg_observability_score**
5. For the type of chart you want to display, select one of the following options from the drop-down:
   - **Line**
   - **Area**
   - **Column**
   - **Scatter**
6. Navigate the pages of your report by clicking the **Previous** and **Next** pagination buttons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a Rules Passing Fraction Roll-Up report.

```
                <MadCap:concept term="Data Quality" xmlns:MadCap-
p="http://www.madcapsoftware.com/Schemas/MadCap.xsd" />with a as
( select * from rule_output where updt_ts >= (NOW() - INTERVAL
'30 DAY') and updt_ts <= NOW()),
     b as ( select * from dataset_scan where rc > 1 ),
     c as ( select * from owl_rule ),
     e as ( select * from dq_dimension ),
     g as ( select * from owl_catalog ),
     h as ( select * from business_unit_to_dataset ),
     i as ( select * from business_units ),
     j as ( select distinct dataset, col_nm, col_semantic
from  dataset_schema ),
     f as (
     select
              a.dataset,
               a.rule_nm,
               CASE WHEN c.column_name = '' THEN 'global' WHEN
c.column_name is null THEN 'global'ELSE c.column_name END as
column_name,
               CASE WHEN e.dim_name is null THEN 'UNSPECIFIED'
else e.dim_name END as dim_name,
               c.dim_id,
               ROUND(a.perc) as perc,
               ROUND(a.perc * b.rc) as breaking_rows,
               (100 - (ROUND(a.perc * b.rc) / b.rc))  as score,
                b.rc as row_count,
                a.run_id,
                g.alias,
                g.catalog_rank,
                g.db_nm,
                g.run_mode,
                g.source_name,
                g.table_nm,
                i.name,
                j.col_nm,
                j.col_semantic
    from a
    LEFT JOIN b on a.dataset = b.dataset and a.run_id = b.run_id
    INNER JOIN c on a.dataset = c.dataset and a.rule_nm =
c.rule_nm
    LEFT JOIN e on e.dim_id = c.dim_id
    INNER JOIN g on g.dataset = a.dataset
    LEFT JOIN h on h.dataset = g.dataset
    LEFT JOIN i on i.id = h.id
    LEFT JOIN j on a.dataset = j.dataset and c.column_name =
j.col_nm
)
```

```
select case when avg(score) < 0 then 0 else  avg(score) end as
avg_percent_rows_passing, dim_name as dimension from f group by
dim_name
```

User Activity Report

What is a User Activity Report?

The User Activity report provides a view of the number of clicks in the Collibra DQ application by user. You can filter this report by various criteria from a Filter Type dropdown, as well as set a limit to how many users you want to view, based on the filter you choose. You can also configure the kind of chart you want to use to view the data.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, select the **User Activity** tile.
3. To display results by filter type, select one of the following options from the **Filter Type** dropdown:
   - **Last x Days** - filter by last number of days.
     i. Select one of the following options from the **Last x Days** dropdown:
     - **1 day**
     - **2 days**
     - **7 days**
     - **10 days**
     - **15 days**
     - **30 days**

ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

iii. Proceed to step 4.

- **Custom Range** - filter by configuring a custom range.
  i. Select a from and to date range in the Date Range fields.

> Tip You can also select the icon, which displays a calendar, in which you can select a specific date.

  ii. To limit the number of users that the chart displays, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

  iii. Proceed to step 4.
- **All** - filter by username and user clicks.
  i. To limit the number of users that the chart displays, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

  ii. Proceed to step 4.

4. Click **Go**.
5. To configure the type of chart configuration and data you want to see in the chart, click the ellipsis icon.
6. Click **Chart Configuration**.
   The Chart Configuration window displays.

> Note As a best practice, keep the Chart Configuration as the default.

7. For the x-axis column, select one of the following options from the **X-Axis Column** dropdown:
    - **username**
    - **userClicks**
8. For the y-axis columns, select one or more of the following options from the **Y-Axis Columns** dropdown:
    - **userClicks**
    - **username**
9. For the type of chart you want to display, select one of the following options from the **Chart Type** dropdown:
    - **Line**
    - **Area**
    - **Column**
    - **Scatter**
10. Click **Save**.
    The chart displays based on the criteria you selected. The table below the chart displays the specific usernames and user clicks, based on your selections.

> Note To close the Chart Configuration window, click **Cancel** or the **X** in the upper-right corner of the window. You can hide or show the filter options by clicking the ^ or v icons, located to the right of the filter options.

> Tip To narrow your results in the data table, you can enter information in the Search field.

11. Navigate the pages of your report by clicking the and icons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a User Activity report.

```
select username, count(*) user_clicks from activity_audit
where audit_ts > '2022-03-05'
and username <> 'admin'
group by username
order by user_clicks desc limit 22
```

Row Count by Dataset Report

What is a Row Count by Dataset Report?

The Row Count by Dataset report provides a view of the row count in the Collibra DQ application by dataset. You can filter this report by various criteria from a Filter Type dropdown, as well as set a limit to the number of datasets you want to view, based on the filter you choose. You can also configure the kind of chart you want to use to view the data.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, select the **Row Count by Dataset** tile.
3. To display results by filter type, select one of the following options from the **Filter Type** dropdown:

- **Last x Days** - filter by last number of days.
    i. Select one of the following options from the **Last x Days** dropdown:
        - **1 day**
        - **2 days**
        - **7 days**
        - **10 days**
        - **15 days**
        - **30 days**
    ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

        > Note The chart and table update, based on your selections.

    iii. Proceed to step 4.
- **Custom Range** - filter by configuring a custom range.
    i. Select a from and to date range in the Date Range fields.

        > Tip You can also select the icon, which displays a calendar, in which you can select a specific date.

    ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

        > Note The chart and table update, based on your selections.

    iii. Proceed to step 4.
- **All** - filter by datasets and row counts.

i. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

ii. Proceed to step 4.

4. Click **Go**.
5. To configure the type of chart configuration and data you want to see in the chart, click the ellipsis icon.
6. Click **Chart Configuration**.
   The Chart Configuration window displays.

> Note As a best practice, keep the Chart Configuration as the default.

7. For the x-axis column, select one of the following options from the **X-Axis Column** dropdown:
   - **rowCount**
   - **dataset**
8. For the y-axis columns, select one or more of the following options from the **Y-Axis Columns** dropdown:
   - **rowCount**
   - **dataset**
9. For the type of chart you want to display, select one of the following options from the **Chart Type** dropdown:
   - **Line**
   - **Area**
   - **Column**
   - **Scatter**

10. Click **Save**.
    The chart displays based on the criteria you selected. The table below the chart displays the specific datasets and row counts, based on your selections.

    > Note To close the Chart Configuration window, click **Cancel** or the **X** in the upper-right corner of the window. You can hide or show the filter options by clicking the ^ or v icons, located to the right of the filter options.

    > Tip To narrow your results in the data table, you can enter information in the Search field.

11. Navigate the pages of your report by clicking the and icons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a Row Count by Dataset report.

```
select dataset,rc as row_count from dataset_scan order by rc
desc
```

Alert Details Report

What is an Alert Details Report?

The Alert Details report provides a view of the details of all the alerts in the Collibra DQ application and the method by which they are sent. You can filter this report by various criteria from a Filter Type dropdown, as well as set a limit to the number of alert details you want to view, based on the filter you choose. You can also configure the kind of chart you want to use to view the data.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, select the **Alert Details** tile.
3. To display results by filter type, select one of the following options from the **Filter Type** dropdown:
   - **Last x Days** - filter by last number of days.
     i. Select one of the following options from the **Last x Days** dropdown:
        - **1 day**
        - **2 days**
        - **7 days**
        - **10 days**
        - **15 days**
        - **30 days**

     ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

     > Note The chart and table update, based on your selections.

     iii. Proceed to step 4.
   - **Custom Range** - filter by configuring a custom range.
     i. Select a from and to date range in the Date Range fields.

     > Tip You can also select the icon, which displays a calendar, in which you can select a specific date.

ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

iii. Proceed to step 4.

- **All** - filter by all alerts and alert value formats.

i. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

ii. Proceed to step 4.

4. Click **Go**.
5. To configure the type of chart configuration and data you want to see in the chart, click the ellipsis icon.
6. Click **Chart Configuration**.
   The Chart Configuration window displays.

> Note As a best practice, keep the Chart Configuration as the default.

7. For the x-axis column, select one of the following options from the **X-Axis Column** dropdown:
   - **alerts**
   - **alertFormatValue**
8. For the y-axis columns, select one or more of the following options from the **Y-Axis Columns** dropdown:
   - **alerts**
   - **alertFormatValue**

9. For the type of chart you want to display, select one of the following options from the **Chart Type** dropdown:
    - **Line**
    - **Area**
    - **Column**
    - **Scatter**
10. Click **Save**.
    The chart displays based on the criteria you selected. The table below the chart displays the specific alert details, based on your selections.

    > Note To close the Chart Configuration window, click **Cancel** or the **X** in the upper-right corner of the window. You can hide or show the filter options by clicking the ^ or v icons, located to the right of the filter options.

    > Tip To narrow your results in the data table, you can enter information in the Search field.

11. Navigate the pages of your report by clicking the and icons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for an Alert Details report.

```
select alert_format_value, count(*) alerts
from alert_cond
group by alert_format_value
order by alerts desc limit 10
```

Job Runs per User

What is a Job Runs per User Report?

The Job Runs per User report provides a view of the count of DQ Jobs run in the Collibra DQ application by user. You can filter this report by various criteria from a Filter Type dropdown, as well as set a limit to the number of usernames and job runs you want to view, based on the filter you choose. You can also configure the kind of chart you want to use to view the data.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, select the **Job Runs Per User** tile.
3. To display results by filter type, select one of the following options from the **Filter Type** dropdown:
   - **Last x Days** - filter by last number of days.
     i. Select one of the following options from the **Last x Days** dropdown:
        - **1 day**
        - **2 days**
        - **7 days**
        - **10 days**
        - **15 days**
        - **30 days**

     ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

iii. Proceed to step 4.

- **Custom Range** - filter by configuring a custom range.
  i. Select a from and to date range in the Date Range fields.

  > Tip You can also select the icon, which displays a calendar, in which you can select a specific date.

  ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

  > Note The chart and table update, based on your selections.

  iii. Proceed to step 4.
- **All** - filter by all job runs and users.
  i. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

  > Note The chart and table update, based on your selections.

  ii. Proceed to step 4.

4. Click **Go**.
5. To configure the type of chart configuration and data you want to see in the chart, click the ellipsis icon.
6. Click **Chart Configuration**.
   The Chart Configuration window displays.

   > Note As a best practice, keep the Chart Configuration as the default.

7. For the x-axis column, select one of the following options from the **X-Axis Column** dropdown:
    - **jobs**
    - **username**
8. For the y-axis columns, select one or more of the following options from the **Y-Axis Columns** dropdown:
    - **jobs**
    - **username**
9. For the type of chart you want to display, select one of the following options from the **Chart Type** dropdown:
    - **Line**
    - **Area**
    - **Column**
    - **Scatter**
10. Click **Save**.
    The chart displays based on the criteria you selected. The table below the chart displays the specific datasets and row counts, based on your selections.

    > Note To close the Chart Configuration window, click **Cancel** or the **X** in the upper-right corner of the window. You can hide or show the filter options by clicking the ^ or v icons, located to the right of the filter options.

    > Tip To narrow your results in the data table, you can enter information in the Search field.

11. Navigate the pages of your report by clicking the and icons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a Job Runs per User report.

```
select username, count(*) jobs
from owlcheck_q
where updt_ts > '2022-03-07'
and username <> 'admin'
group by username
order by jobs desc
```

Daily Job Runs per User

What is a Daily Job Runs per User Report?

The Daily Job Runs per User report provides a view of the count of DQ Jobs run in the Collibra DQ application by user, per day. You can filter this report by various criteria from a Filter Type dropdown, as well as set a limit to the number of usernames, dates, and job runs you want to view, based on the filter you choose. You can also configure the kind of chart you want to use to view the data.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, select the **Daily Job Runs Per User** tile.
3. To display results by filter type, select one of the following options from the **Filter Type** dropdown:
   - **Last x Days** - filter by last number of days.
     i. Select one of the following options from the **Last x Days** dropdown:
     - **1 day**

- **2 days**
- **7 days**
- **10 days**
- **15 days**
- **30 days**

ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

iii. Proceed to step 4.

◦ **Custom Range** - filter by configuring a custom range.

i. Select a from and to date range in the Date Range fields.

> Tip You can also select the icon, which displays a calendar, in which you can select a specific date.

ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

iii. Proceed to step 4.

◦ **All** - filter by all daily job runs and users.

i. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

ii. Proceed to step 4.

4. Click **Go**.
5. To configure the type of chart configuration and data you want to see in the chart, click the ellipsis icon.
6. Click **Chart Configuration**.
   The Chart Configuration window displays.

   > Note As a best practice, keep the Chart Configuration as the default.

7. For the x-axis column, select one of the following options from the **X-Axis Column** dropdown:
   - **jobs**
   - **updtTsStr**
   - **username**
8. For the y-axis columns, select one or more of the following options from the **Y-Axis Columns** dropdown:
   - **jobs**
   - **updtTsStr**
   - **username**
9. For the type of chart you want to display, select one of the following options from the **Chart Type** dropdown:
   - **Line**
   - **Area**
   - **Column**
   - **Scatter**
10. Click **Save**.
    The chart displays based on the criteria you selected. The table below the chart

displays the specific datasets and row counts, based on your selections.

> Note To close the Chart Configuration window, click **Cancel** or the **X** in the upper-right corner of the window. You can hide or show the filter options by clicking the ^ or v icons, located to the right of the filter options.

> Tip To narrow your results in the data table, you can enter information in the Search field.

11. Navigate the pages of your report by clicking the and icons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a Job Runs per User report.

```
select updt_ts::date, username, count(*)
from owlcheck_q
where updt_ts > '2022-3-09'
and username <> 'admin'
group by updt_ts::date, username
order by updt_ts asc
```

Rule Updates per User

What is a Rule Updates per User Report?

The Rule Updates per User report provides a view of when a rule is inserted, updated, or deleted and applied to the dataset in the Collibra DQ application, by user for a specific time. You can filter this report by various criteria from a Filter Type dropdown, as well as set a limit to how many users and rules you want to view, based on the filter you choose. You can also configure the kind of chart you want to use to view the data.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, select the **Rule Updates per User** tile.
3. To display results by filter type, select one of the following options from the **Filter Type** dropdown:
   - **Last x Days** - filter by last number of days.
     i. Select one of the following options from the **Last x Days** dropdown:
        - **1 day**
        - **2 days**
        - **7 days**
        - **10 days**
        - **15 days**
        - **30 days**

     ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

     > Note The chart and table update, based on your selections.

     iii. Proceed to step 4.
   - **Custom Range** - filter by configuring a custom range.
     i. Select a from and to date range in the Date Range fields.

     > Tip You can also select the icon, which displays a calendar, in which you can select a specific date.

ii. To limit the number of users that the chart displays, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

iii. Proceed to step 4.

- **All** - filter by username and rules.

i. To limit the number of users that the chart displays, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

ii. Proceed to step 4.

4. Click **Go**.
5. To configure the type of chart configuration and data you want to see in the chart, click the ellipsis icon.
6. Click **Chart Configuration**.
   The Chart Configuration window displays.

> Note As a best practice, keep the Chart Configuration as the default.

7. For the x-axis column, select one of the following options from the **X-Axis Column** dropdown:
   - **rules**
   - **username**
8. For the y-axis columns, select one or more of the following options from the **Y-Axis Columns** dropdown:
   - **rules**
   - **username**

9. For the type of chart you want to display, select one of the following options from the **Chart Type** dropdown:
    - **Line**
    - **Area**
    - **Column**
    - **Scatter**
10. Click **Save**.
    The chart displays based on the criteria you selected. The table below the chart displays the specific usernames and user clicks, based on your selections.

    > Note To close the Chart Configuration window, click **Cancel** or the **X** in the upper-right corner of the window. You can hide or show the filter options by clicking the ^ or v icons, located to the right of the filter options.

    > Tip To narrow your results in the data table, you can enter information in the Search field.

11. Navigate the pages of your report by clicking the and icons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a User Activity report.

```
select user_nm, count(*) rules from rule_history
where updt_ts > '2022-03-05'
and user_nm <> 'admin'
group by user_nm order by rules desc
```

Custom

Custom reports can be leveraged by connecting your favorite BI tool on the underlying reporting mart. Below are a few queries that can be used as inspiration for building your own reports. Please refer to the ERD diagram for a larger list of tables.

Long Running Jobs

```
select dataset,run_id,total_time from dataset_activity where total_
time is not null order by total_time desc
```

Jobs Submitted

```
select * from owlcheck_q
```

Jobs by User

```
select count(*) as owlchecks, username from owlcheck_q where updt_ts
< now() group by username order by owlchecks desc
```

Jobs by User, Dataset

```
select count(*), user_nm, dataset from dev.public.owl_check_history
group by user_nm, dataset order by count desc
```

Largest by Row Count

```
select dataset,rc as row_count from dataset_scan order by rc desc
```

Jobs by Month

```
with grp as ( select date_trunc('MONTH', run_id) as by_month from
dataset_scan where run_id < now() ) select count(*) as owlchecks,
by_month from grp group by by_month order by by_month desc
```

Rules by User

```
select count(*) as rules, user_nm from owl_rule group by user_nm
order by rules desc
```

By Spark(Cluster) Usage

```
select * from opt_spark order by num_executors desc
```

Jobs IDs from Agent

```
select remote_job_id from agent_q where remote_job_id is not null
```

Dataset Activity

```
select dataset,run_id,total_time from dataset_activity where total_
time is not null order by total_time desc
```

Jobs with Enriched Metrics

```
with activity as ( select dataset,run_id,total_time from dataset_
activity where total_time is not null order by total_time desc limit
100), scans as ( select * from dataset_scan where dataset in (select
dataset from activity) ), configs as ( select * from opt_spark where
dataset in (select dataset from activity)), schema as ( select count
(*) as col_cnt, dataset from dataset_schema where dataset in (select
dataset from activity) group by dataset ) SELECT A.dataset, A.run_
id, C.total_time, A.rc, D.col_cnt, B.driver_memory, B.num_
executors,B.executor_cores, B.executor_memory, B.master FROM scans A
INNER JOIN configs B ON A.dataset = B.dataset INNER JOIN activity C
ON A.dataset = C.dataset and A.run_id = C.run_id INNER JOIN schema D
on A.dataset = D.dataset ORDER BY C.total_time desc
```

Jobs. Load Times and Resources

```
with activity as ( select dataset,run_id,total_time from
public.dataset_activity where total_time is not null order by total_
time), scans as ( select * from public.dataset_scan where dataset in
(select dataset from activity) ), configs as ( select * from
public.opt_spark where dataset in (select dataset from activity)),
schema as ( select count(*) as col_cnt, dataset from public.dataset_
schema where dataset in (select dataset from activity) group by
dataset ) SELECT A.dataset, A.run_id, A.updt_ts, C.total_time, A.rc,
D.col_cnt, B.driver_memory, B.num_executors,B.executor_cores,
B.executor_memory, B.master FROM scans A INNER JOIN configs B ON
A.dataset = B.dataset INNER JOIN activity C ON A.dataset = C.dataset
and A.run_id = C.run_id INNER JOIN schema D on A.dataset = D.dataset
ORDER BY A.updt_ts desc limit 10
```

Dataset Scans and Scores By Schema

```
select * from public.dataset_scan where dataset like 'public.%';
```

Dataset Scans and Scores By Name

```
select * from public.dataset_scan where dataset ='public.atm_
customer';
```

Scans By Month By Schema - 'Public'

```
select dataset, DATE_TRUNC('MONTH', run_id) as run_id, count(*) as
Total_Scans from dataset_scan where dataset like 'public%' group by
dataset, run_id order by run_id asc
```

Rule Breaks Past 30 Days

```
select * from rule_output where run_id < NOW() - INTERVAL '30 DAY';
```

Scheduled Jobs Queue

```
select job_id,agent_id,dataset,run_id,status,activity,start_time
from public.owlcheck_q;
```

Column Counts from Dataset Schema

```
select dataset, count(*) from dataset_schema group by dataset;
```

Profiling Stats

```
select dataset, run_id, field_nm, (null_ratio * 100) as null_
percent, (empty_ratio * 100) as empty_percent, ROUND( CAST( ( 100 -
((null_ratio * 100) + (empty_ratio * 100)) ) as numeric), 3) as
completeness from public.dataset_field where updt_ts > '2020-06-01'
and dataset = 'ProcessOrder' and run_id > '2021-03-17 00:00:00+00'
order by completeness desc
```

Metadata / Schema / Datatypes

```
select * from public.dataset_schema;
```

Profile Stats

```
select * from public.dataset_field;
```

Locate Similar Columns

```
select distinct dataset, field_nm, max_abs from dataset_field where
max_abs = 'Wireless Telecommunications'
```

Same Column Names

```
select distinct dataset, field_nm from dataset_field where field_nm
= 'authenticated_user'
```

Similar Column Names

```
select distinct dataset,field_nm from dataset_field where field_nm
like '%id%'
```

Behavior Findings

```
select * from behavior where dataset='esg_data'
```

All Columns for Schema from Postgres Stats

```
SELECT table_name FROM information_schema.tables WHERE table_schema
= 'public' ORDER BY table_name
```

Chapter 12

Collibra DQ Workflows

Manage Collibra DQ Assignments

Collibra DQ allows you to assign DQ Job findings (observations) to other users to review those findings and make determinations based on whether the findings are legitimate.

Why Would I Assign DQ Job Findings to Users?

Sometimes a DQ Job run results in findings that include what seem to be anomalies in the data source. For example, when analyzing a job, you might notice a job finding with outliers that are not what you were expecting. In this case, you could assign this job finding to a user to either **validate**, **invalidate**, or **resolve** the finding.

When an assigned user validates a job finding, it means that the results are expected and that we can confirm that it is, in fact, what the behavior model (outlier, dupe, shape, etc.) is reporting. The result is legitimate and, therefore, we want it to continue reporting as is.

If the user invalidates a finding, it means that the results are not expected and it is not what the behavior model is reporting it to be. In this case, the finding could be reporting an actual problem so that finding would need to be modified so that, when fixed, it will not be shown on the next job run.

> Note Collibra recommends assigning the job finding validation to a user who has access to the source data.

You can assign job findings to internal Collibra DQ (CDQ) users or to external users with a third-party account, such as the ServiceNow tool. When you assign a user with a ServiceNow account, the assignments are sent as incidents, which can be tracked via the Service Now tool. You can select an individual or a group of users to manage the assignments for actions, such as escalations, de-escalations, and reassignments.

> Note The default configuration for assignments is CDQ, however, you can also configure external ServiceNow assignments in the Validate Finding window.

Assign Job Findings to Users

You can assign any job findings to a user who has previously logged into the Collibra DQ application (internal user) or one who has a matching ServiceNow account (external user).

> Note The default configuration for assignments is CDQ, however, you can also configure external ServiceNow assignments in the Validate Finding window.

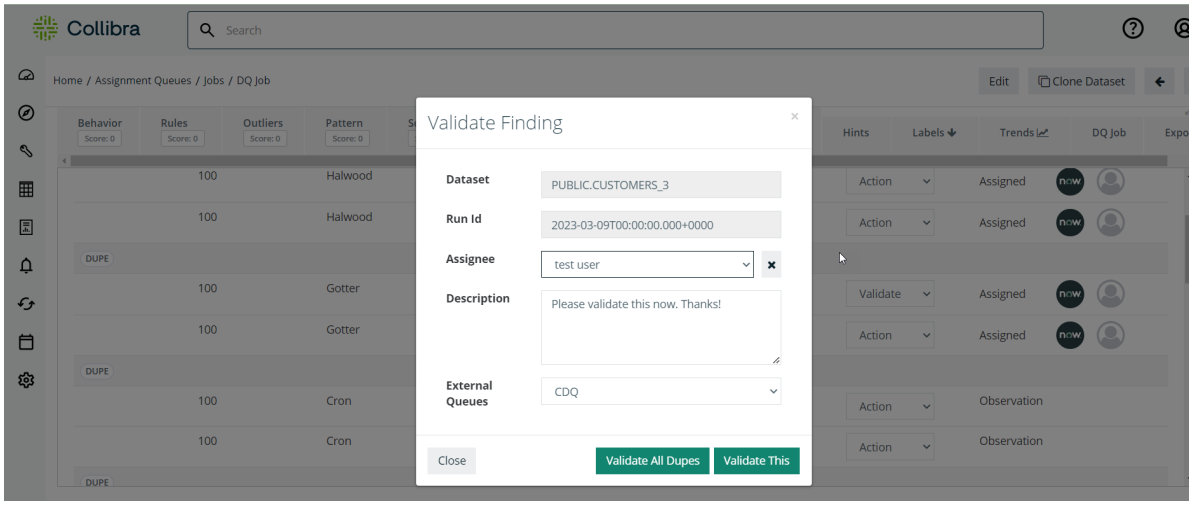
> Warning To configure a queue, you must have ROLE_ADMIN or ROLE_ CONNECTION_MANAGER.

Steps

1. Login to the Collibra DQ instance.
2. Select one of the findings of a previously run DQ Job.
3. From the **Actions** dropdown, select one of the following options.

Option	Description
Validate	Value is expected and represents the truth. Selecting this option means that you want the finding to be reported in the next job run.
Invalidate	Value is not what is expected and must be modified and fixed. You can set to Off-Peak, Ignore, or Remove so that the finding is removed from that day's run.
Resolve	Resolve the job finding. Resolving a finding retrains your Job's quality score if any points were deducted.

4. To validate a finding and assign it to an internal Collibra DQ user, select **Validate** from the **Actions** dropdown.
   The Validate Finding window displays.



5. Complete the following fields.

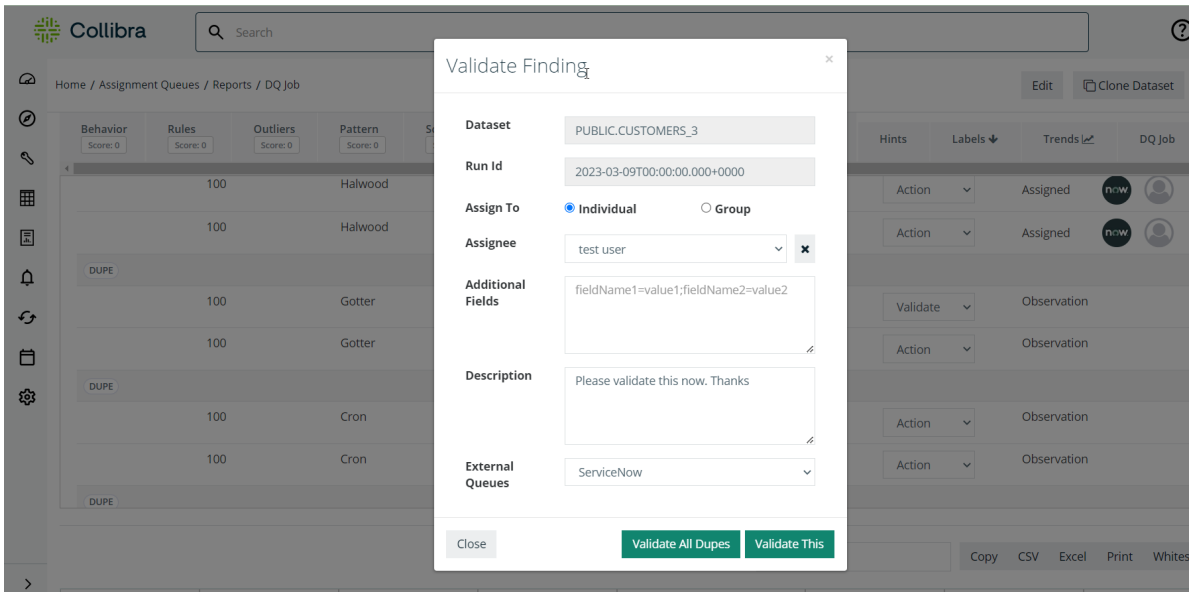
Field	Description
Dataset	Dataset of the finding. This field is already populated.
Run Id	ID of the DQ job run. This field is already populated.
Assignee	Collibra DQ user. Select the user from this dropdown.
Description	Enter a message for the user, such as "Please validate now".
External Queues	Select the internal/external user from this dropdown. Note CDQ is the default value.

6. To validate this finding, click **Validate This**.

> Note You can validate all the behaviors at once by selecting **Validate All <Behaviors>**. To close the window without saving, click **Close**.

7. To validate a finding and assign it to an external user (with a ServiceNow account), select **Validate** from the **Actions** dropdown.
   The Validate Finding window displays.
8. Select **ServiceNow** from the **External Queues** dropdown and complete the following fields.

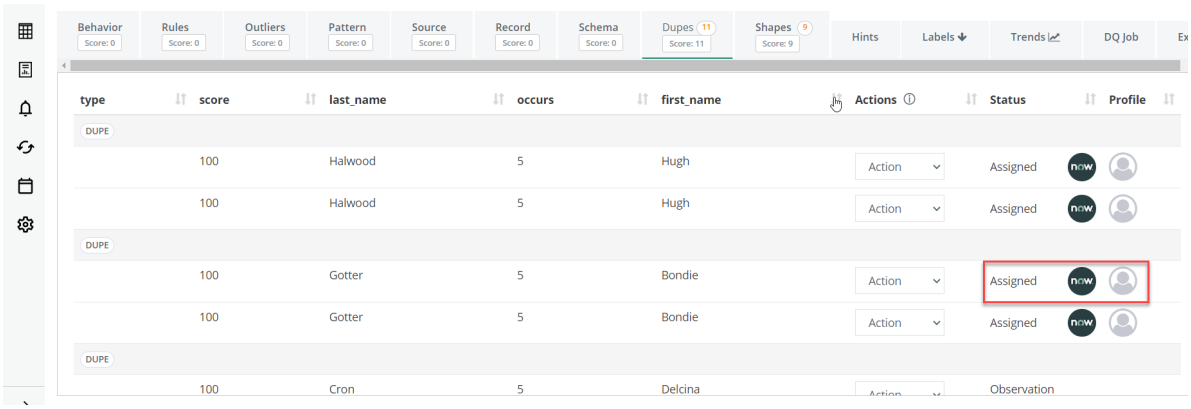
Field	Description
Dataset	Dataset of the finding. This field is already populated.
Run Id	ID of the DQ job run. This field is already populated.
Assign To	Select one of the following options: ○ **Individual** - Assign to one user ○ **Group** - Assign to a group of users
Assignee	Collibra DQ user. Select the user from this dropdown. Note If you do not select an assignee, the item is marked as valid but unassigned: **Acknowledged**.
Description	Enter a message for the user, such as "Please validate now", to provide details for the assignee. This field is optional.
External Queues	Select **ServiceNow** from this dropdown. Note CDQ is the default value.



9. To validate this finding, click **Validate This**.

> Note You can validate all the behaviors at once by selecting **Validate All <Behaviors>**. To close the window without saving, click **Close**.

You can distinguish whether the user is a CDQ or ServiceNow assignment in the Profile column. Users are represented by an icon, rather than their names or usernames.



10. To invalidate a finding, select **Invalidate** from the **Action** dropdown and add a note in the Annotation* field.

> Note The Annotation* field is a required field.

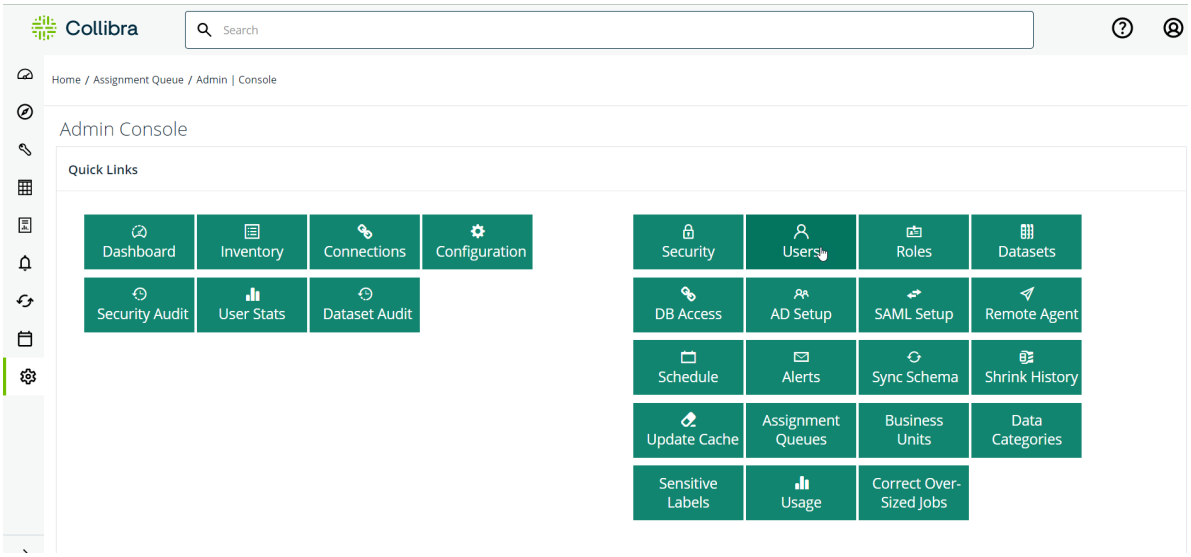
11. Click **Save**, **Save & Retrain**, or **Close** to close without saving.

Configure an External Assignment Connection

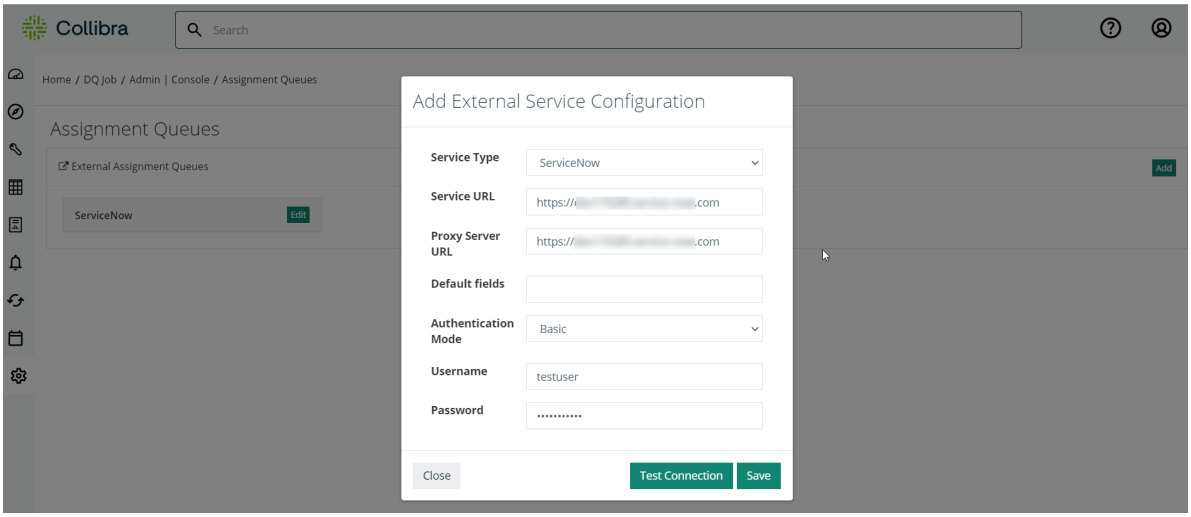
You can add or edit an external service configuration connection in the Admin Console.

Steps

1. Login to the Collibra DQ instance and click the gear icon in the left navigation pane.
2. Select **Admin Console** and click **Assignment Queues**.



3. Select **Add**, to add a new external service configuration connection, or **Edit**, to edit an existing connection.



4. Complete the following fields in the Edit External Service Configuration window.

Field	Description
Service Type	Type of assignment queue service you are configuring. Default is ServiceNow.
Service URL	URL of the service you are configuring.

Field	Description
Proxy Server URL	URL of the proxy server. Note This is only applicable based on infrastructure configuration.
Default Fields	Semicolon separated key=value pairs on the Admin page, which override related field values on created assignment service records. Note There is a hierarchy whereby Additional Fields take precedence, if there is a matching competing key=value pair between both fields.
Authentication Mode	Type of authentication.
Username	Authentication credentials to the service.
Password	Authentication credentials to the service.

5. Select **Save**.

> Note You can select **Test Connection**, to ensure that the configuration for your connection is valid. You will receive a message indicating if the connection test was successful.
> You can also click **Close** to close the window without saving.

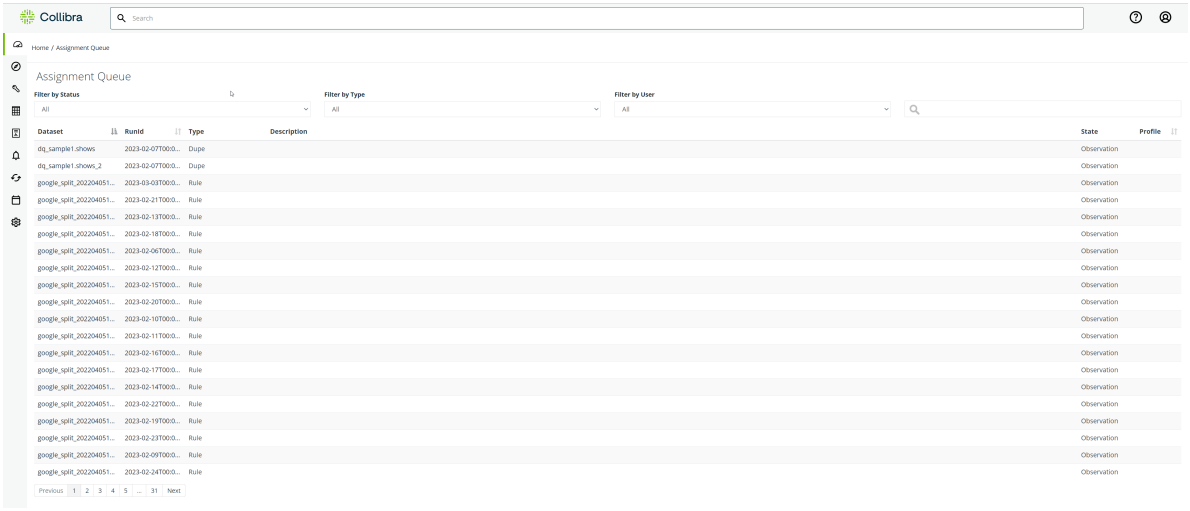
View the Assignment Queue

The Assignments Queue is a summary of existing DQ Job runs that can be filtered by behavior model assessment (outliers, dupes, shapes, etc.), users, and run status.

> Note You can view the Assignments Queue via the icon or via the Workflow option on the Collibra Data Quality & Observability Home page.

Steps

1. Login to the Collibra DQ instance and select the dashboard icon.
2. Select **Assignments**.
   The Assignment Queue displays.

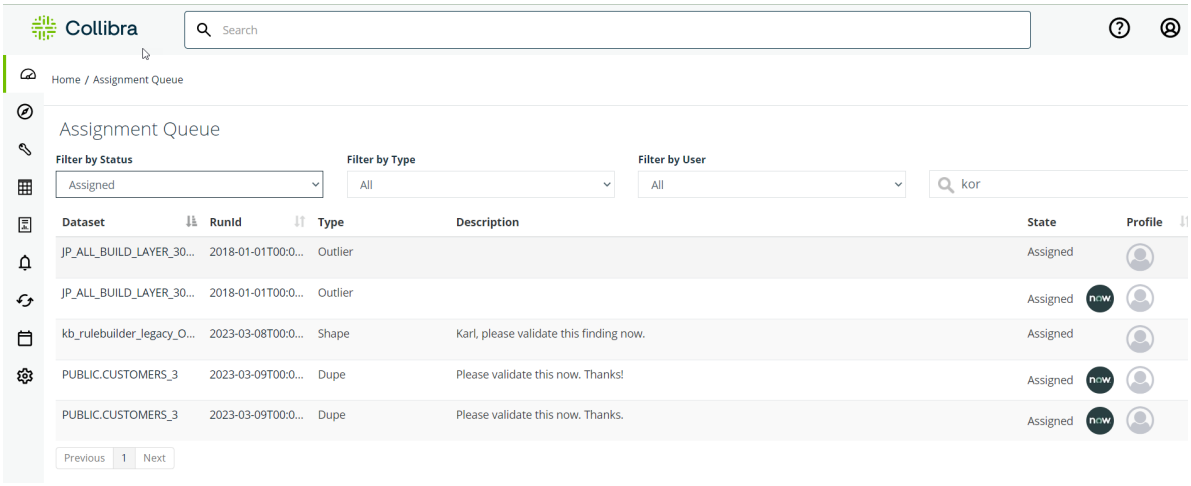


3. You can filter the Assignment Queue by selecting from the following criteria:
   - **Status** - Filter by status by selecting one of the following options in the **Filter by Status** dropdown:
     - **Unassigned** - Assignments that have not been assigned to a user to validate.
     - **Assigned** - Assignments that have been assigned to a user to validate.
     - **Verified** - Assignments that have been verified.
     - **Resolved** - Assignments that have been resolved.
   - **Filter by Type** - Filter by type of behavior model assessment by selecting one of the following options in the **Filter by Type** dropdown:
     - **Outlier**
     - **Shape**
     - **Rule**

- **Behavior**
- **Dupe**
- **Pattern**
- **Record Changes**
- **Schema Evolution**
- **Mismatch Source**

- **Filter by User** - Filter by user by selecting a user in the **Filter by User** dropdown.

You can also filter by entering your criteria in the Search field.



The Assignment Queue results include the following:

- **Dataset** - Link to a specific dataset.
- **RunID** - ID of the DQ job run.
- **Type** - Behavior model assessment of the Assignment.
- **Description** - Annotation or message added in the Validate Finding window. For more information, go to Manage Collibra DQ Assignments.
- **State** - State of the Assignment.
- **Profile** - User who was assigned the Job Finding.

FAQ

Can anyone assign an item?

Anyone with access to the dataset and TRAIN role (if enabled) can assign an item. Users who have been assigned an item can resolve without the TRAIN role.

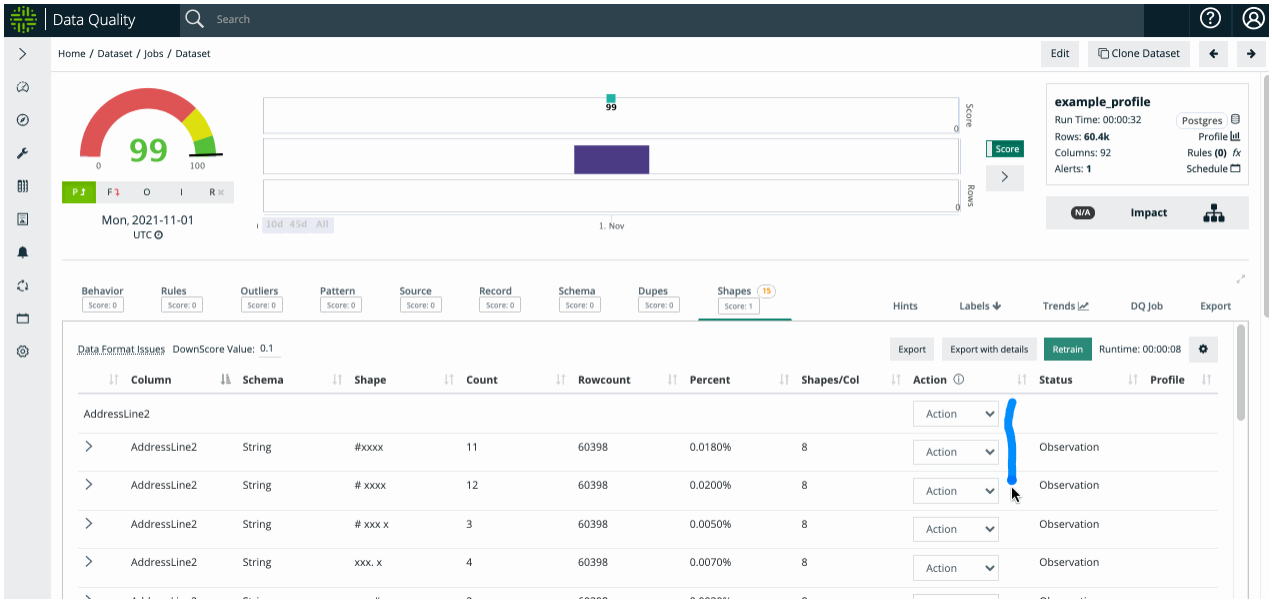
How do I add people to the assignment list

Once a person logs into the Collibra DQ Application they will get a profile and become eligible for assignment.

Labeling / Training

Item Labeling

Quickly click findings to trigger retraining.



Action labeling options

The following action labels instruct Collibra on how to handle a finding:

Action	Description
Validate	Instructs Collibra to either assign a finding to a specific user for review, which then appears in the View the Assignment Queue, or acknowledge without an assignee that the finding is a valid observation. **Note:** Validating a finding does not improve your score.
Invalidate	Instructs Collibra to ignore a finding and allow the value to pass. There are two invalidation options: Save and Save & Retrain. **Save:** Allows you to mark a finding as invalidated. **Save & Retrain:** Allows you to invalidate a finding and any previously saved invalidated findings (if any). **Note:** When you have many findings to invalidate, it may be best to use the Save option to invalidate them at the same time, once all findings are reviewed.
Resolve	Instructs Collibra to mark the finding as an observation and prevents it from appearing in future runs. Resolving a finding does not immediately affect data quality scores.

Available actions by feature

Feature	Available actions
Behaviors	Validate, Resolve
Rules	Validate, Resolve
Outliers	Validate, Invalidate, Resolve
Pattern	Validate, Invalidate, Resolve
Source	Validate, Invalidate, Resolve
Record	Validate, Resolve
Dupes	Validate, Invalidate, Resolve

> Warning Some findings are ineligible for all labeling options. For example, you can only apply Validate and Resolve labels to findings that result from Rules.

Validating a finding

When you apply a validate label to a finding, you can assign it to another DQ user to review. This marks the finding for future runs and sends it to the internal Assignments Queue. You can also configure DQ to send assignments to an external queue, such as the ServiceNow tool. For more information, go to Manage Collibra DQ Assignments.

Invalidating a finding

Sometimes the findings page flags issues with your data that DQ discovers during a job run, but maybe you want DQ to ignore certain flagged issues. The invalidate label allows you to do that. After you add a descriptive annotation of your action, you can then select either Save or Save & Retrain.

Save

If you have a large number of findings that DQ has flagged, and you want to invalidate all of them at once instead of clicking through one at a time, select Save for all of the findings you would like to bulk invalidate. On your last finding, select Save & Retrain. All previously saved invalidated findings are removed and DQ retrains your data set.

Save & Retrain

When you Save & Retrain your data set, any previously deducted points from a flagged finding are restored and reflected in your overall data quality score. If you do not have many findings to invalidate, you can Save & Retrain individually instead of in bulk.

Resolving a finding

Some features, such as Behavior and Rules, only permit Validate and Resolve actions. When you cannot Validate a finding but you want to apply a label, select Resolve. The

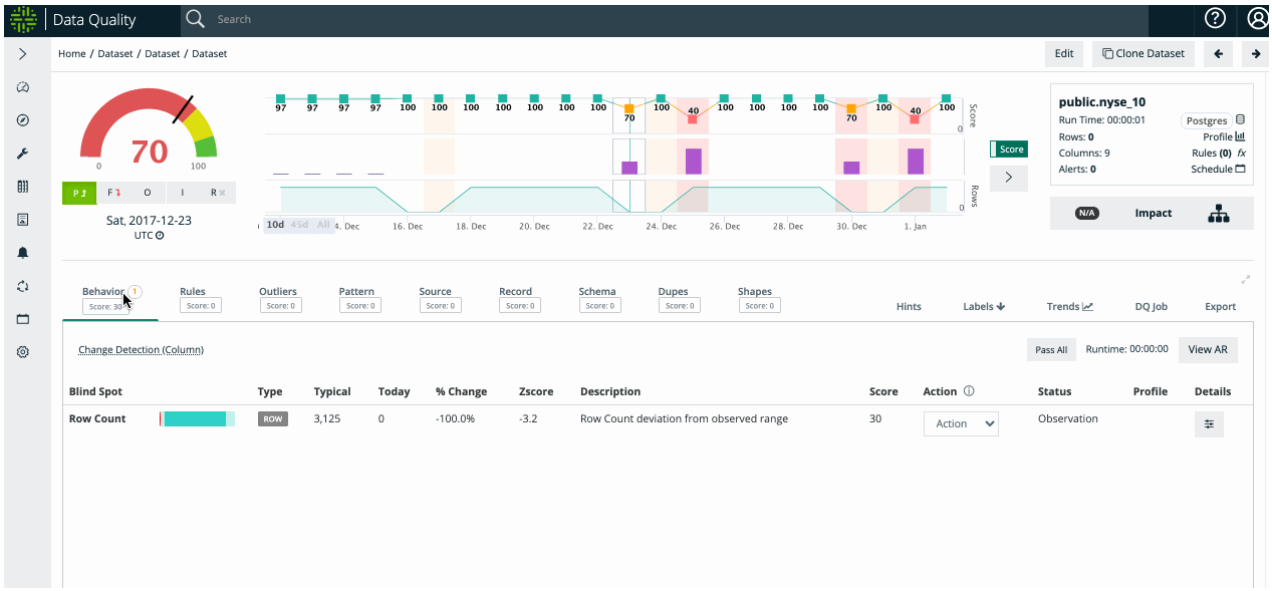
Resolve label prevents a finding from appearing in future runs of your data set, and does not immediately affect your data quality score when applied.

Recalling labeled findings

To modify a previously labeled finding, you can always access them through the Labels tab. Here you can edit an annotation or delete a label entirely. If you delete a label, it returns to the findings page, unlabeled. From there you can again choose to Validate, Invalidate, or Resolve it.

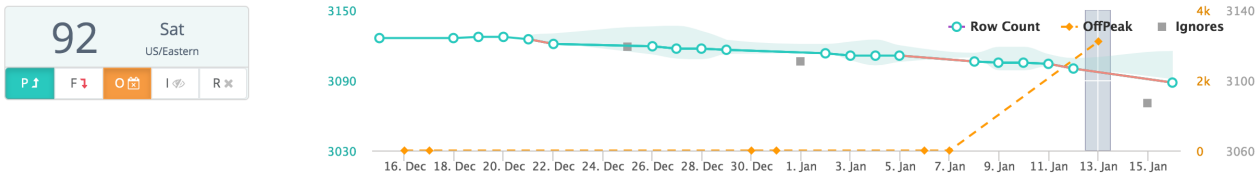
To closely analyze when a finding has received a label, who has applied it, and more, see also the Dataset Audit Trail.

Peak vs Off Peak



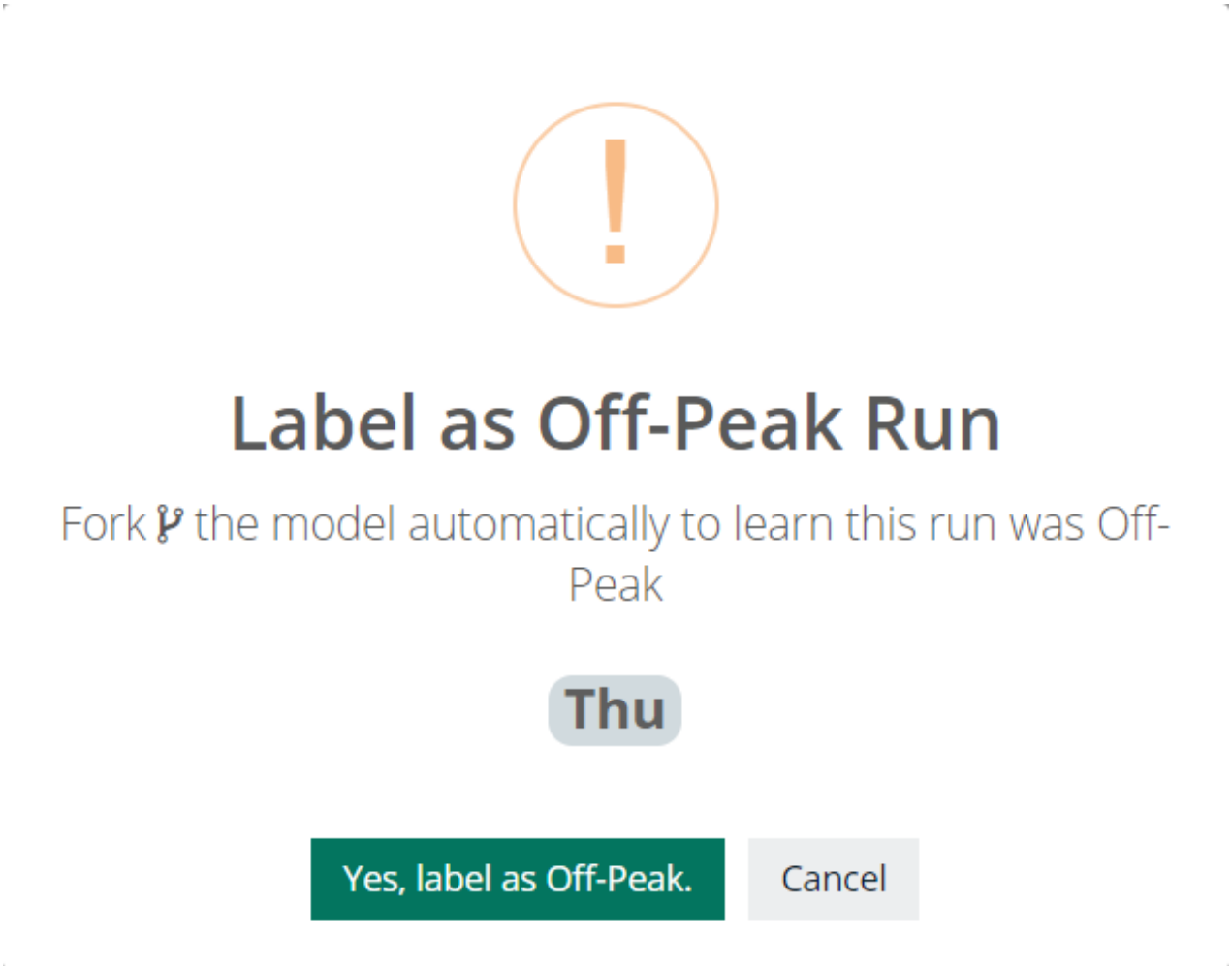
But my Weekend runs are not the Same

A common scenario that can fool behavioral analytics and machine learning is when you have a few different but normal patterns. Collibra has a rich labeling system that allows a user to fork the training model to learn these cycles individually without confusing the model.



Click the Green Button

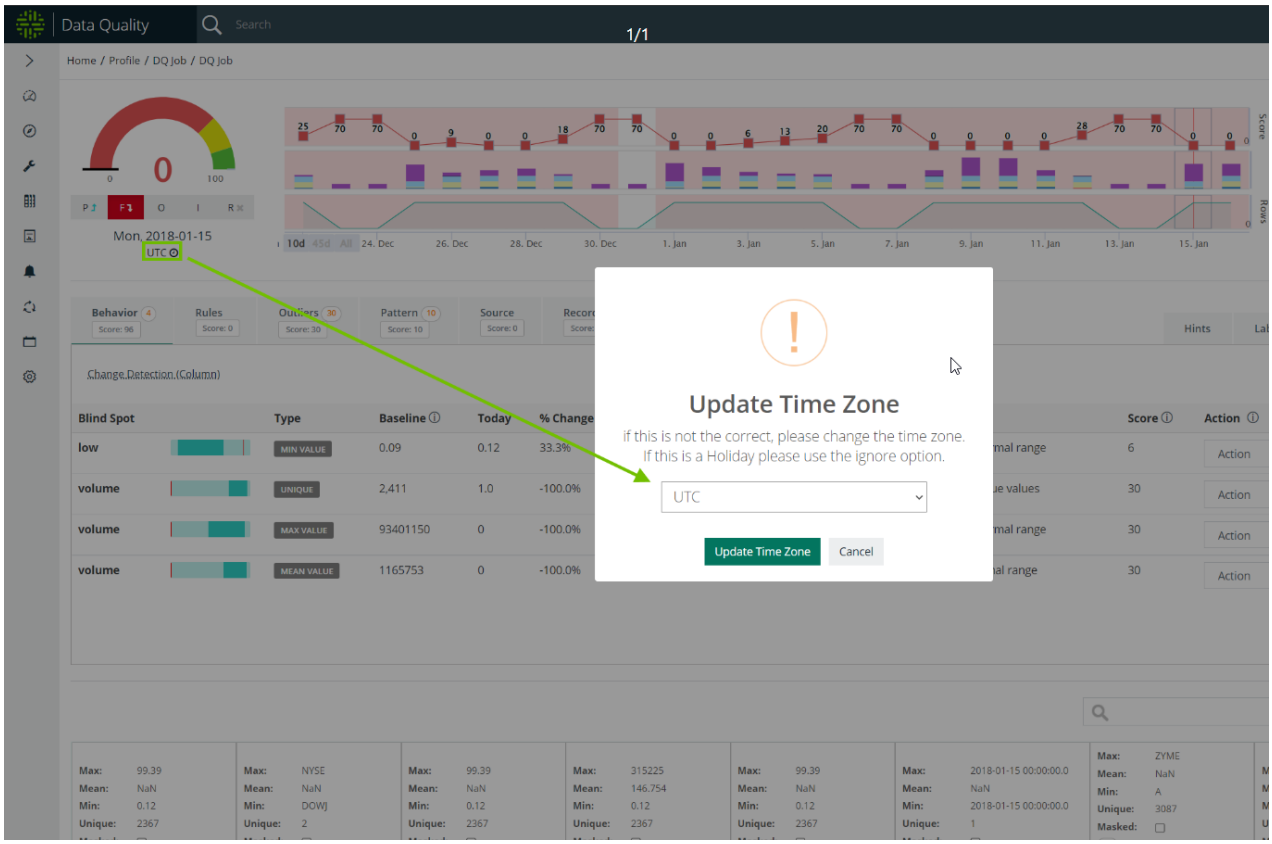
By clicking the green button, you can label the day of the week as peak vs off peak. You can also chose your time zone - this will help determine the day of the week accurately. You only need to click the peak scheduler once and the model will learn and forecast this understanding for every run in the future. This feature commonly prompts for a re-train.



Time Zones

Updating time zones

By default, Collibra Data Quality's time zone associated with the RunID is located in Coordinated Universal Time (UTC). To update the server time zone, select the Update Time Zone link from the findings page. An Update Time Zone dialog displays with the option to select your time zone from the dropdown menu. Click the Update Time Zone button to confirm your selection.



> Note Since the server time zone can differ from the configurable RunID time zone on the findings page, data sets in List View may have different dates than the date listed on the findings page of the same data set. For example, a data set with a RunID in the default UTC time zone may appear as 2022-01-15 00:00:00 on the findings page, but because the server is located in US/Central time, the date appears as 2022-01-14 19:00:00.

Chapter 13

Collibra DQ DIC Integration

DQ Connector

The Native DQ Connector brings intelligence from **Collibra Data Quality & Observability** into **Collibra Data Intelligence Cloud**. Once this integration is established, you will be able to bring in your Data Quality user-defined rules, metrics, and dimensions into **Collibra Data Catalog**.

> Note Only data sources ingested by both Collibra Data Catalog and Collibra Data Quality & Observability can synchronize data quality assets.

Prerequisites

Resource	Notes
Collibra Edge Site	DQ Connector is a capability of Edge
Collibra Data Intelligence Cloud	2021.07 Release (or newer)
Collibra Data Quality	2.15 (or newer)
Databases and Drivers	Proper Access and Credentials (Username / Password)

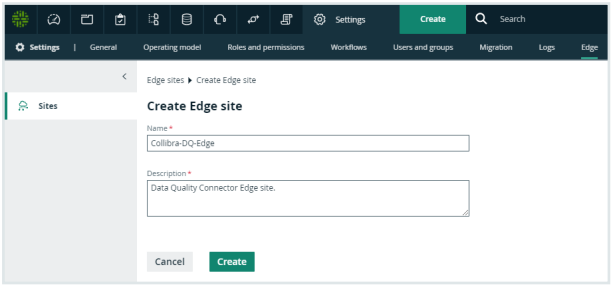
Because the DQ Connector is an Edge capability, you must be able to ingest data via Edge. For information about enabling and configuring Edge, see the Edge Configuration guide.

Create a Collibra Data Quality & Observability Edge site

Create an Edge site with the following properties:

Field	Description
Name	The name of the Edge site, for example **Collibra-DQ-Edge**. Do not use spaces or special characters. This field is mandatory and the name must be globally unique.

Field	Description
Description	The description of the Edge site. We recommend to put at least basic location information of the Edge site. This field is mandatory.



Install the Collibra Data Quality & Observability Edge site

Follow the instructions for your environment to Install an Edge site.

> Note This process automatically creates an Edge user, which you use later in the setup process.

Connect to your Collibra Data Quality & Observability source

Create a connection for each Collibra Data Quality & Observability data source you want to synchronize. The following table shows the available properties and their descriptions as they appear on the :

Section	Property	Description
Connection settings		

Section	Property	Description
	Name	The same name as the Collibra Data Quality & Observability connection name. Ensure that your connection name does not contain any white spaces, as they are not supported in Collibra DQ. Warning The connection name in Collibra Data Intelligence Cloud must be an exact match to the connection name used in Collibra DQ.
	Description	The description of the JDBC connection. This field is also visible when you register content.
	Connection provider	The connection provider, which determines the available connection parameters. Same as Collibra Data Quality & Observability.
Connection parameters		Example for **Username / Password JDBC driver**
	Username	The same username as the Collibra DQ connection username.
	Password	The same password as the Collibra DQ connection password.
	Driver class name	In most cases, this is the same driver name as the Collibra DQ connection driver name. If you select a different driver in Collibra Data Intelligence Cloud, the driver class name can be different from the Collibra DQ driver class name.

Section	Property	Description
	Driver Jar	In most cases, this is the same driver JAR file as from Collibra DQ. If you select a different driver in Collibra Data Intelligence Cloud, the driver jar can be different from the Collibra DQ driver jar. Ensure that the driver is supported in both Collibra Data Intelligence Cloud and Collibra DQ. Note Some CDATA drivers that are supported in Collibra Data Intelligence Cloud are not supported in Collibra DQ. It is best practice to use a CDATA driver in Collibra Data Intelligence Cloud, but you can use a different driver in Collibra DQ.
	Connection string	In most cases, this is the same URL as the Collibra DQ connection URL. If you select a different driver in Collibra Data Intelligence Cloud, the connection URL can be different from the Collibra DQ connection URL.

Add ingestion capabilities to your Collibra Data Quality & Observability connection

You must add a Catalog JDBC ingestion Edge capability template for each connection you have created to extract and process data for your data source.

Field	Description	Required
Capability	This section contains the general information about the capability.	
Name	The name of the Edge capability.	✓ Yes
Description	The description of the Edge capability.	✗ No
Capability template	The capability template, which determines the next available sections. Select the following Edge capability: `Catalog JDBC ingestion`	✓ Yes
Connection	This section contains information to connect to the data source.	

Field	Description	Required
JDBC connection	The connection to the data source.	✓ Yes
JDBC data source type (Deprecated)	Deprecated field. The field was used to indicate the type of the data source. You no longer need to change this field. The required value is automatically identified. Note The automatically identified value is not shown in this page.	✓ Yes
Supports schemas	A text field where you have to enter *True* to enable database registration of data sources that have no schema. If the data source has schemas, you can ignore this field. Tip If the data source does not have a schema, Data Catalog creates a Schema asset with the same name as the full name of the database.	✗ No

Field	Description	Required
Others	This section can contain additional capability properties. Warning Adding additional properties can have a significant impact on your Edge site. Only add or update them together with Collibra Support.	✕ No

Field	Description	Required

Field	Description	Required
General	This section contains general information about logging.	
Debug	An option to automatically send Edge infrastructure log files to Collibra Data Intelligence Cloud. By default, this option is set to *false*. Note We highly recommend to only send Edge infrastructure log files to Collibra Data Intelligence Cloud when you have issues with Edge. If you set it to *true*, it will automatically revert to *false* after 24h.	✕ No
Log level	An option to determine the verbosity level of Catalog connector log files. By default, this option is set to *No logging*.	✕ No

Configure destinations for Collibra Data Quality & Observability assets

Collibra Data Quality & Observability rules, metrics and dimensions require their own domains in Data Catalog. If you don't have existing domains for data quality or wish to use new ones for the quality extraction purpose, create a domain for each type of data quality asset:

- Rules: **Rulebook Domain**
- Metrics: **Business Asset Domain**
- Dimensions: **Business Asset Domain**

Assign permissions for Collibra Data Quality & Observability domains

Edge must have the correct resource permissions to manage assets inside the dedicated Collibra Data Quality & Observability domains. For each dedicated domain, assign the **Technical Steward** role to the Edge user.

> Note The Edge user is automatically created when you install the Edge site.

Add Collibra Data Quality & Observability characteristics to assets

To show Collibra Data Quality & Observability statistics for your data source, assign the following characteristic types to the **Table** and **Column** asset types:

Asset type	Characteristic type
Table	governed by Governance Asset
Column	is governed by Data Quality Rule

Add a DQ Connector capability

The DQ Connector facilitates the communication with Collibra Data Quality & Observability. Add a DQ Connector capability to your Collibra Data Quality & Observability Edge site:

Field	Description	Required
Capability	This section contains the general information about the capability.	
Name	The name of the Edge capability.	✓ Yes
Description	The description of the Edge capability.	✗ No
Capability template	The capability template, which determines the next available sections. Select the following capability template to ingest Collibra Data Quality & Observability user-defined rules, metrics, and dimensions into Collibra Data Catalog: `DQ Connector`	✓ Yes
DQ	This section contains information about the Collibra Data Quality & Observability connection.	
Base URL	Your Collibra Data Quality & Observability URL	✓ Yes

Field	Description	Required
Username	The Collibra Data Quality & Observability username for this connection.	✓ Yes
Password	The Collibra Data Quality & Observability password for this connection.	✓ Yes
Encryption options	Select the type of encryption to use. Default: *To be encrypted by Edge management server*.	
Issuer of the JWT	If you have selected *Encrypted with public key*, enter your JWT issuer.	✗ No
Collibra metadata model	This section contains information about where to ingest Collibra Data Quality & Observability assets.	
DQ Rules domain id	The UUID of the **Rulebook Domain** for the ingested Collibra Data Quality & Observability rules.	✓ Yes
DQ Metrics domain id	The UUID of the **Business Asset Domain** for the ingested Collibra Data Quality & Observability metrics.	✓ Yes
DQ Dimensions domain id	The UUID of the **Business Asset Domain** for the ingested Collibra Data Quality & Observability dimensions.	✓ Yes
Default DQ Dimension name	The default **Data Quality Dimension**, for example *Accuracy*, *Completeness*, *Consistency* and so on. Default: *Completeness*.	✓ Yes
DQ Metric classified by DQ Dimension relation type id	The UUID of the **Data Quality Metric classified by / classifies Data Quality Dimension** relation. If left unspecified, this relation will not be added.	✗ No
Assets are imported in batches of this size	The batch size of the ingestion. Default: *5000*.	✓ Yes
General	This section contains general information about logging.	

Field	Description	Required
Debug	An option to automatically send Edge infrastructure log files to Collibra Data Intelligence Cloud. By default, this option is set to *false*. Note We highly recommend to only send Edge infrastructure log files to Collibra Data Intelligence Cloud when you have issues with Edge. If you set it to *true*, it will automatically revert to *false* after 24h.	✕ No
Log level	An option to determine the verbosity level of Catalog connector log files. By default, this option is set to *No logging*.	✕ No

To make the Collibra Data Quality & Observability metadata available in Collibra Data Catalog, you must register the data source for each Collibra Data Quality & Observability data source you want to synchronize.

Create a Data Catalog System Asset

As a prerequisite to registering a data source in Data Catalog, you must create a System asset for each connected data source with the following properties:

Field	Value
Type	**System**
Domain	The domain to which the new assets will belong. You can only create a asset type in any domain of a domain type that is assigned to a selected asset type.
Name	The same name as the Collibra Data Quality & Observability connection name.

Register the Collibra Data Quality & Observability data source in Data Catalog

To make the Collibra Data Quality & Observability metadata available in Collibra Data Catalog, you must register the data source for each Collibra Data Quality & Observability data source you want to synchronize.

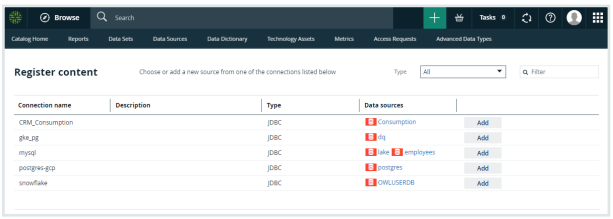
Create a Data Catalog System Asset

As a prerequisite to registering a data source in Data Catalog, you must create a System asset for each connected data source with the following properties:

Field	Value
Type	**System**
Domain	The domain to which the new assets will belong. You can only create a asset type in any domain of a domain type that is assigned to a selected asset type.
Name	The same name as the Collibra Data Quality & Observability connection name. Warning Connection name must be an exact match in both Collibra DQ and Collibra Data Intelligence Cloud. For example, if your connection name is **postgres-gcp** in Collibra DQ, it should also be **postgres-gcp** in Collibra Data Intelligence Cloud.

Register the Collibra Data Quality & Observability data source in Data Catalog

Register each Collibra Data Quality & Observability source in Data Catalog.



Extract Data Quality metadata

After you completed the DQ Connector configuration, you can start ingesting Collibra Data Quality & Observability metadata.

Prerequisites

- You have configured the metadata synchronization properties for the data source.

Steps

1. Open a Database asset page.
2. In the tab pane, click **Configuration**.
3. In the **Quality extraction** section, do one of the following:
    - To select schemas for data quality synchronization:
        i. Click **Edit**.
        » The **Data quality** column becomes editable.
        ii. Select whether to synchronize the available schemas.
        iii. Click **Save**.
    - To synchronize the selected schemas:
        i. Select the schema name to see its configuration.
        ii. Click **Synchronize**.
        » The synchronization job is started for the selected schemas.

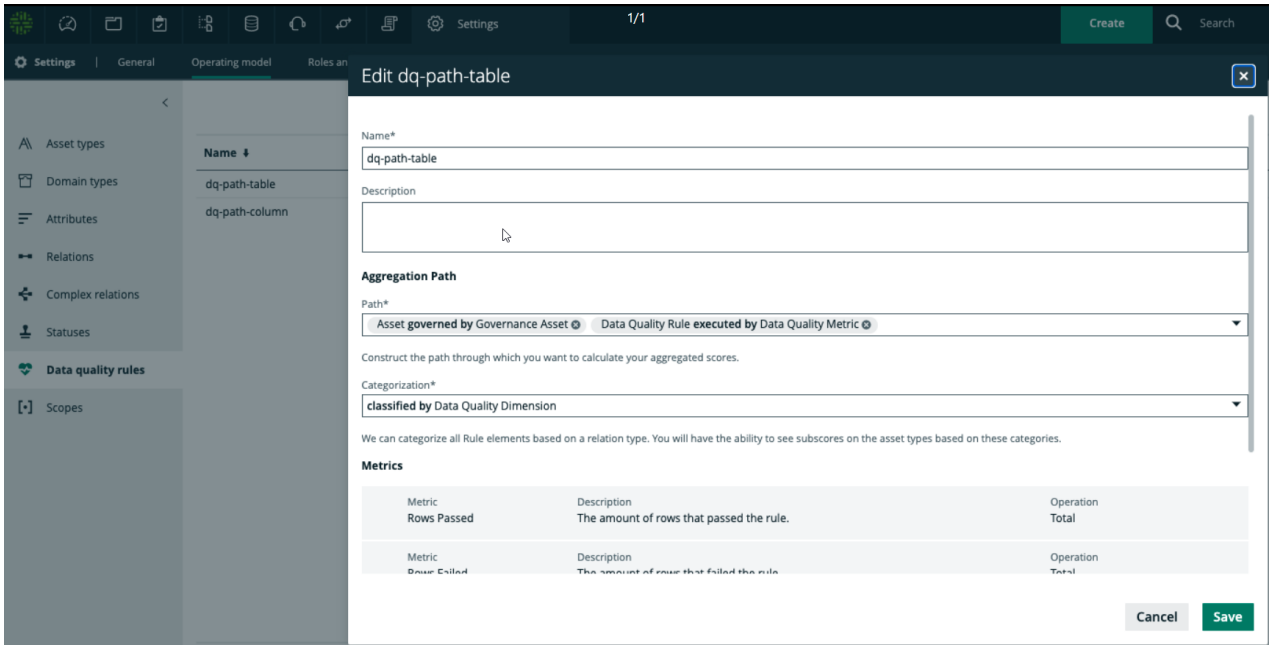
Known Limitations

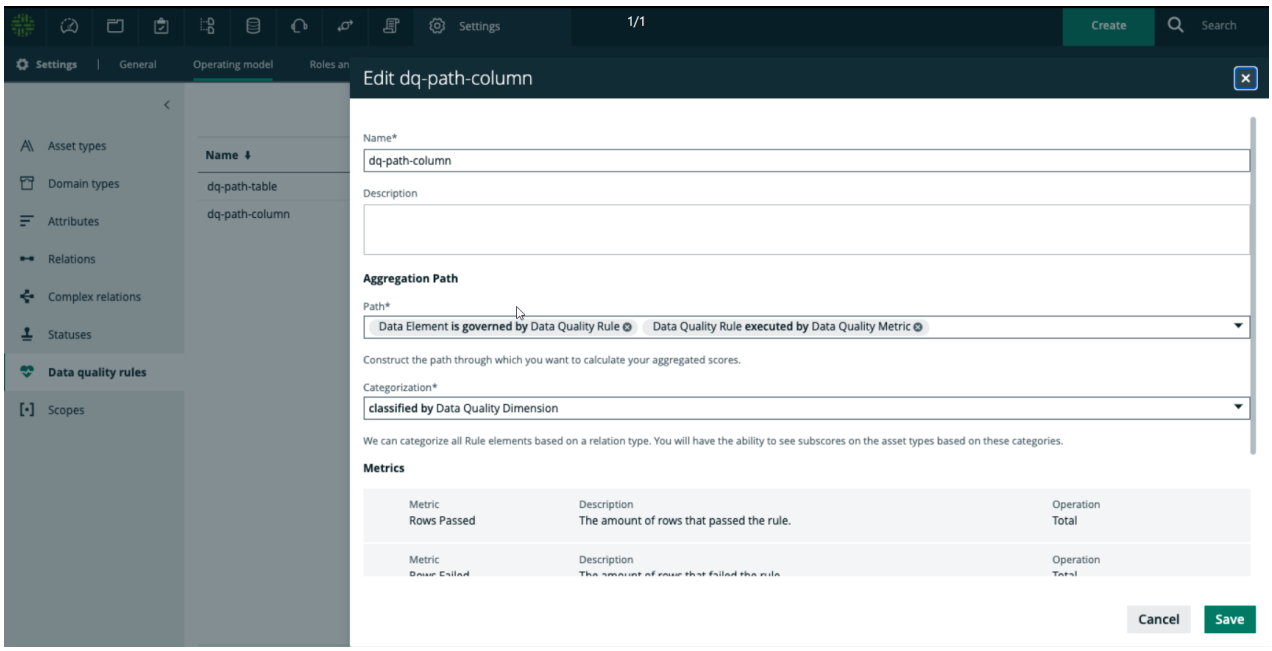
- Only 1 source tenant from Collibra DQ can be specified
- On-demand ingestion (vs. scheduled)
- Can only specify 1 domain destination for each of Rules, Metrics, and Dimensions
- Only JDBC sources supported (no file sources)

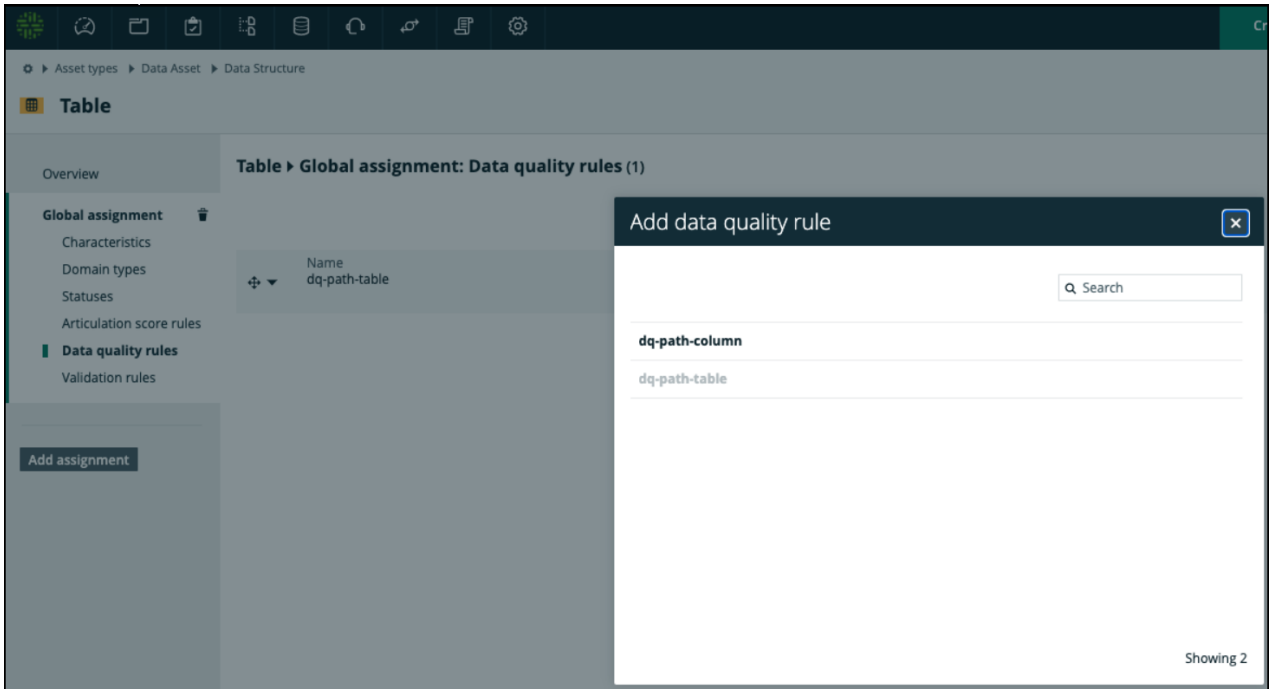
FAQ

Q: DQ Dashboard In DGC: I can verify the DQ Connector is synchronizing Data Quality Rules and Data Quality Metrics, but why don't Data Quality Dashboard Charts display?

A: Ensure correct **Aggregation Paths** and **Global Assignments** (or create, if none exist) for **Table** and **Column** below.

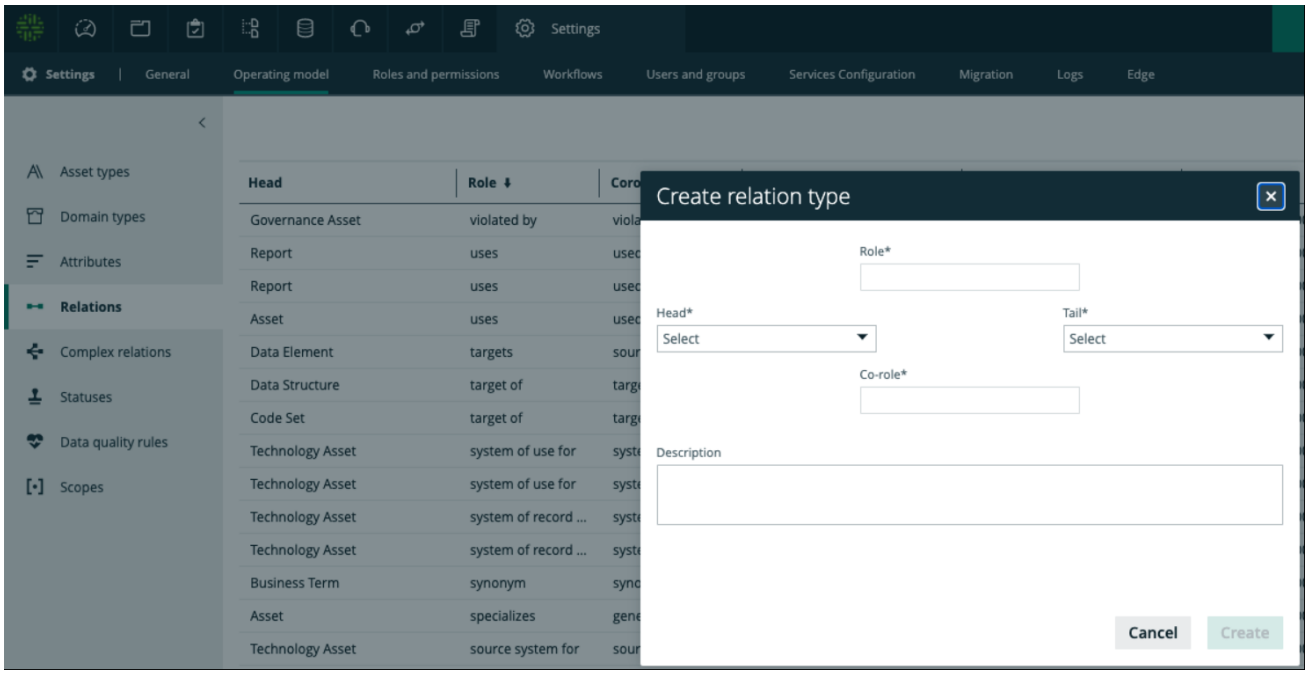

Settings
1/1
Create
Search
Settings | General | Operating model | Roles an
Edit dq-path-table
Asset types
Domain types
Attributes
Relations
Complex relations
Statuses
Data quality rules
Scopes
Name
dq-path-table
dq-path-column
Name*
dq-path-table
Description
Aggregation Path
Path*
Asset governed by Governance Asset
Data Quality Rule executed by Data Quality Metric
Construct the path through which you want to calculate your aggregated scores.
Categorization*
classified by Data Quality Dimension
We can categorize all Rule elements based on a relation type. You will have the ability to see subscores on the asset types based on these categories.
Metrics
Metric
Description
Operation
Rows Passed
The amount of rows that passed the rule.
Total
Cancel
Save

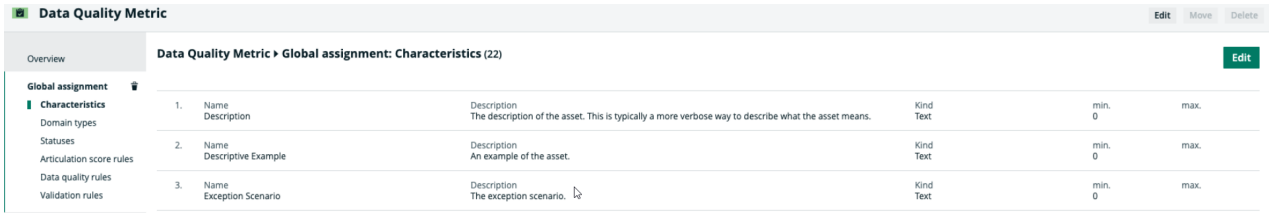

Settings
1/1
Create
Search
Settings | General | Operating model | Roles an
Edit dq-path-column
Asset types
Domain types
Attributes
Relations
Complex relations
Statuses
Data quality rules
Scopes
Name
dq-path-table
dq-path-column
Name*
dq-path-column
Description
Aggregation Path
Path*
Data Element is governed by Data Quality Rule
Data Quality Rule executed by Data Quality Metric
Construct the path through which you want to calculate your aggregated scores.
Categorization*
classified by Data Quality Dimension
We can categorize all Rule elements based on a relation type. You will have the ability to see subscores on the asset types based on these categories.
Metrics
Metric
Description
Operation
Rows Passed
The amount of rows that passed the rule.
Total
Cancel
Save

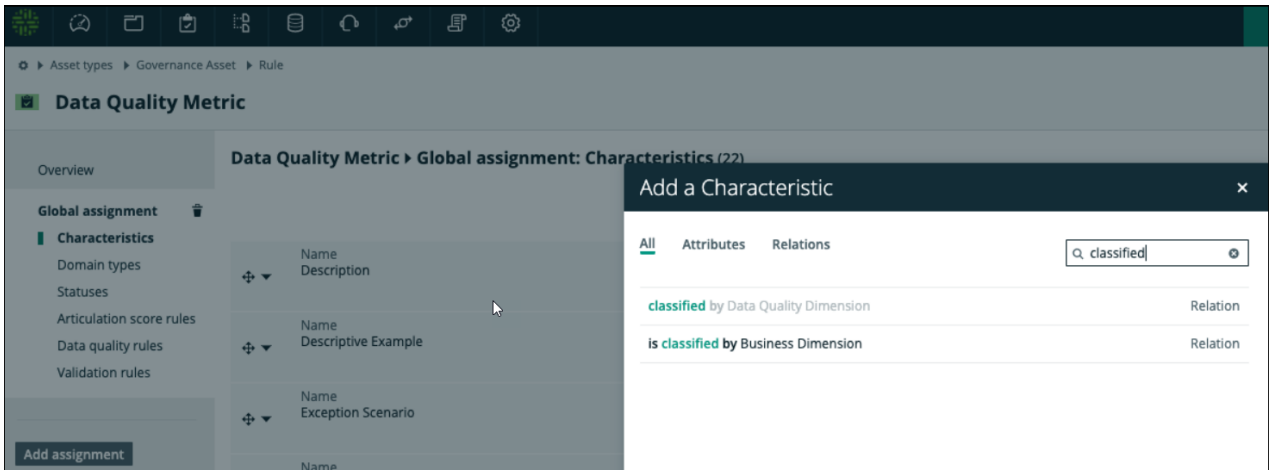


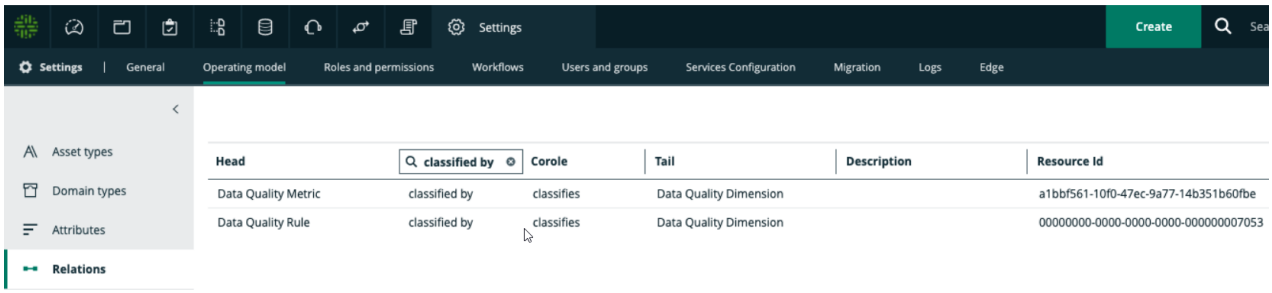
Q: DQ Dashboard In DGC: Why won't my DQ Dimension charts display in my Dashboard?

A: Please 1) add a new custom **Relation** 'Data Quality Metric classified by Data Quality Dimension', 2) **Global Assignment** for 'Data Quality Metric', 3) UUID of the new **Relation** into the **DQ Connector** setup in **Step 1G**, 4).









Q: I've connected and configured data sources correctly, why aren't DQ Rules and DQ Metrics being synchronized?

A: Please ensure Connection / System Names between Collibra DQ, Collibra, and Edge exactly match.

A: Please ensure Edge user has admin permissions to write the assets into Data Catalog.

A: Please ensure correct URL specified within the DQ Connector capability e.g. http://cdq.customer.com:9000/.

Q: Is DQ Connector unidirectional?

A: Yes, from Collibra DQ to Data Catalog in Collibra Data Intelligence Cloud.

Q: How many DQ Connectors can I run simultaneously?

A: Currently, one.

Q: Does the DQ Connector work with On-Prem Collibra DGC?

A: No, any work with on-prem Collibra DGC would be custom API development via Collibra Professional Services or a partner SI.

Q: If I delete a rule from Collibra DQ that I have already synchronized into Data Catalog, will it be deleted from Catalog in the next synchronization?

A: No, the DQ Connector only upserts into Data Catalog. If a rule is deleted from Collibra DQ, it will not be automatically deleted in Data Catalog.

Q: Why are my scores different in Collibra DQ and Data Catalog?

A: Currently, the DQ Connector pulls in the most recent user-defined rules from Collibra DQ. Other components that affect score such as Behaviors, Outliers, Patterns, Dupes, Source are not yet included.

Q: Getting errors when trying to delete both domain that Edge created for DB and the Connection?

A: Please delete Edge created domain via API.

Q: I've hit the synchronize button, how can I tell if my job is complete?

A: Check the Activities circle (button on top right of menu) for the status of your DQ Synchronization.

Q: Why did the rule with joins between two views that I created in Collibra DQ fail to import into Collibra Data Intelligence Cloud?

A: Because the column from the secondary view is flagged as primary, the rule maps the secondary column to the primary view. This causes the rule to import incorrectly, as the primary column does not exist as the primary view. A known workaround for this is to not select a primary column for this rule, and instead write the rule expression, including the columns required from both the primary and secondary views.

DQ Workflows

Benefits

The DQ Workflows package listed on Collibra Marketplace allows you to 1) create and manage Data Quality Issues, 2) receive Notifications on Rule Metrics, and 3) request Rule Creation and Modification within **Collibra Data Intelligence Cloud**. Data stewards will be able to organize and prioritize all requests within DIC before they take any action within **Collibra Data Quality**.

Once deployed, the workflows will facilitate quicker data issue remediation by involving business analysts and other personas who can now participate in your data quality workstreams.

Please note: DQ Workflows are listed on Collibra Marketplace and are templates to get customers started. Collibra-provided Marketplace listings are not subject to the same SLA obligations (https://marketplace.collibra.com/marketplace-terms/) In addition, they can only be leveraged within Collibra Data Intelligence Cloud. In the future, we will work towards releasing bi-directional workflows.

Step 0: Prerequisites

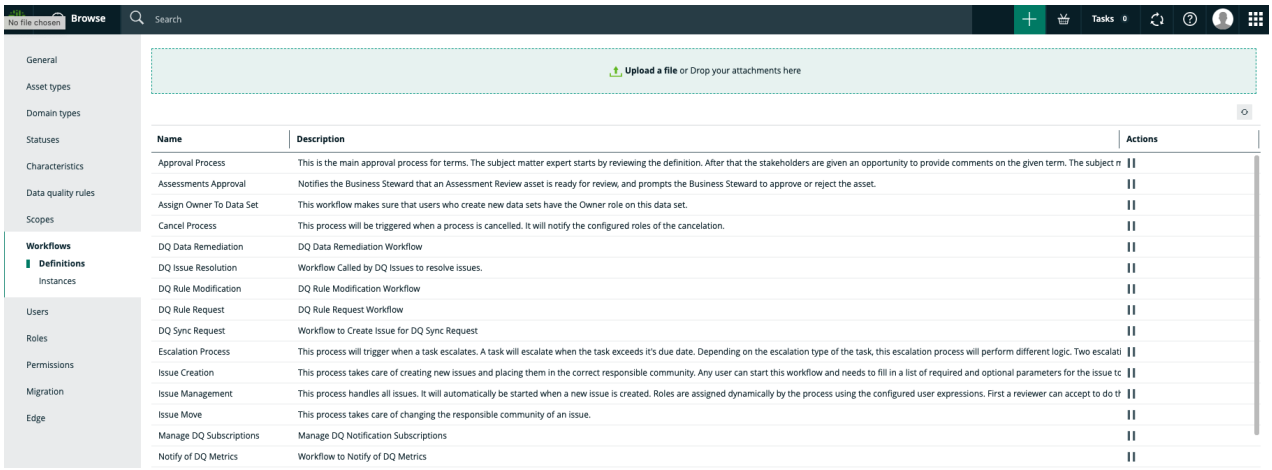
Resource	Notes
Collibra Edge Site	DQ Connector is a capability of Edge
Collibra Data Intelligence Cloud	2021.07 Release (or newer)
Collibra Data Quality	2.15 (or newer)
Collibra DQ Connector	Synchronized Rules from Data Quality to Catalog

> Note After gathering all the prerequisites, you can now proceed to the next step.

Step 1: Download, Deploy and Start DQ Workflows

1A. Download Package from Collibra Marketplace and Unzip Files

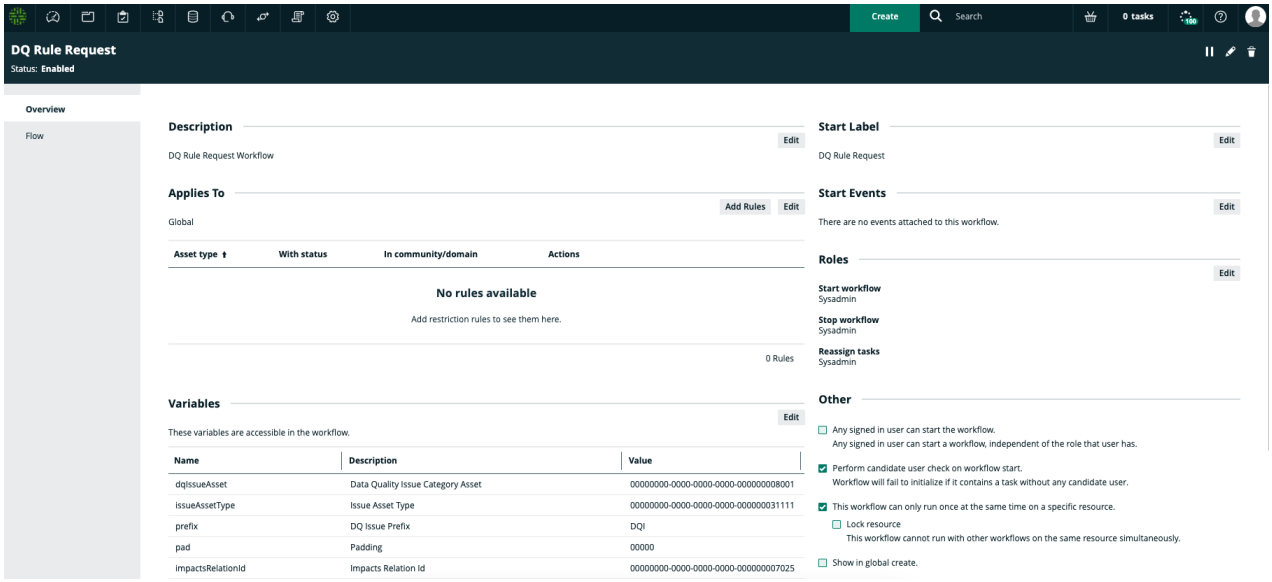
**1B. Deploy Workflows **

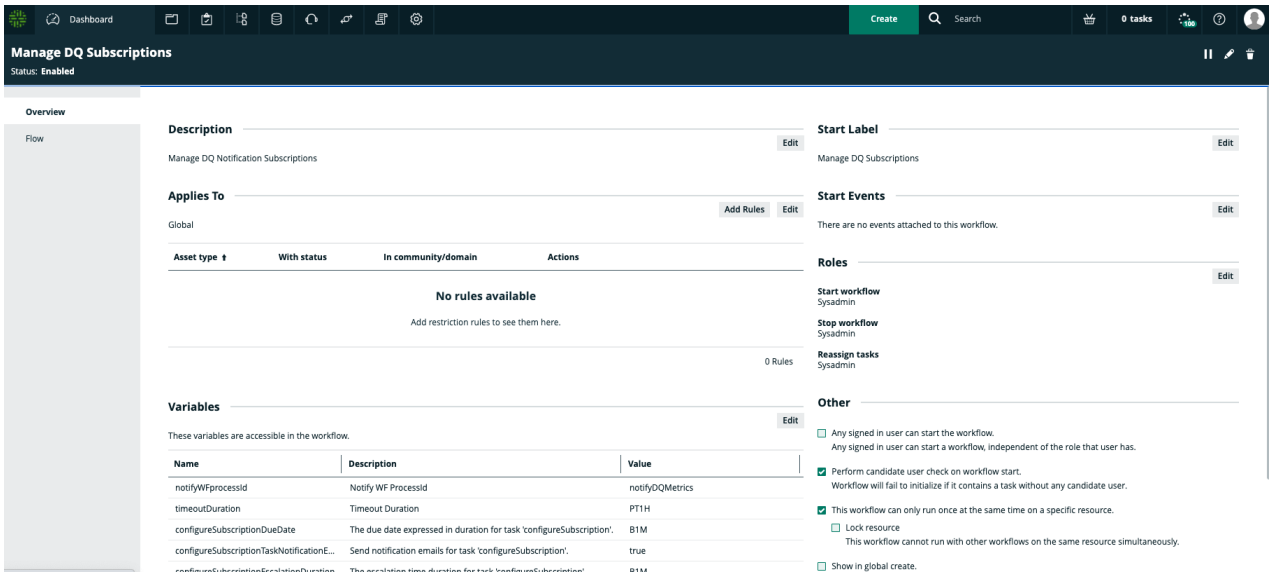


1C. Adjust Workflow Settings (One-Time Setup)

Workflow Configuration Setting	DQ Rule / DQ Sync Request	DQ Rule Modification	DQ Data Remediation	DQ Issue Resolution	Manage DQ Subscriptions	Notify of DQ Metrics
Applies To	Asset	Asset	Asset	Asset	Global	Global
Applies To Asset Type	Column, Table	Column, Table, Data Quality Rule	Column, Table	Issue		
Other: Any Signed In User Can Start Workflow	Y	Y			Y	Y

Workflow Configuration Setting	DQ Rule / DQ Sync Request	DQ Rule Modification	DQ Data Remediation	DQ Issue Resolution	Manage DQ Subscriptions	Notify of DQ Metrics
Other: Perform Candidate User Check on Workflow Start	Y	Y	Y		Y	Y
Other: This Workflow Can Only Run Once At Same Time on Specific Resource					Y	Y
Other: Show In Global Create					Y	Y
Roles: Start Workflow					Sysadmin	Sysadmin
Roles: Stop Workflow					Sysadmin	Sysadmin
Roles: Reassign Tasks					Sysadmin	Sysadmin

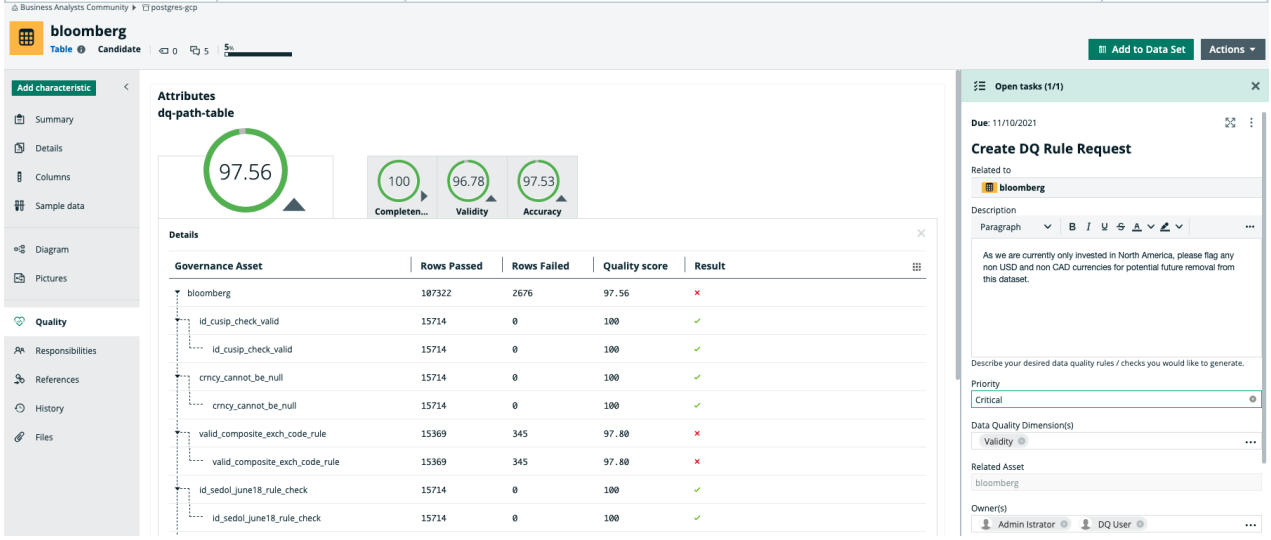

Create
Search
0 tasks
DQ Rule Request
Status: Enabled
Overview
Flow
Description
Edit
DQ Rule Request Workflow
Applies To
Add Rules
Edit
Global
Asset type
With status
In community/domain
Actions
No rules available
Add restriction rules to see them here.
0 Rules
Variables
Edit
These variables are accessible in the workflow.
Name
Description
Value
dqIssueAsset
Data Quality Issue Category Asset
00000000-0000-0000-0000-000000008001
issueAssetType
Issue Asset Type
00000000-0000-0000-0000-000000031111
prefix
DQ Issue Prefix
DQI
pad
Padding
00000
impactsRelationId
Impacts Relation Id
00000000-0000-0000-0000-000000007025
Start Label
Edit
DQ Rule Request
Start Events
Edit
There are no events attached to this workflow.
Roles
Edit
Start workflow
Sysadmin
Stop workflow
Sysadmin
Reassign tasks
Sysadmin
Other
Any signed in user can start the workflow.
Any signed in user can start a workflow, independent of the role that user has.
Perform candidate user check on workflow start.
Workflow will fail to initialize if it contains a task without any candidate user.
This workflow can only run once at the same time on a specific resource.
Lock resource
This workflow cannot run with other workflows on the same resource simultaneously.
Show in global create.

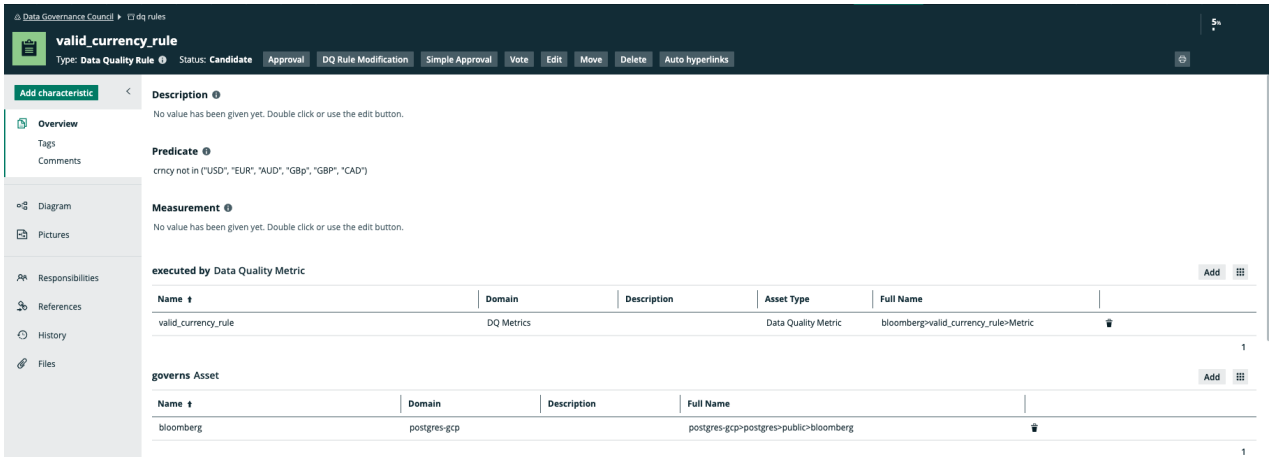

Dashboard
Create
Search
0 tasks
Manage DQ Subscriptions
Status: Enabled
Overview
Flow
Description
Edit
Manage DQ Notification Subscriptions
Applies To
Add Rules
Edit
Global
Asset type
With status
In community/domain
Actions
No rules available
Add restriction rules to see them here.
0 Rules
Variables
Edit
These variables are accessible in the workflow.
Name
Description
Value
notifyWFprocessId
Notify WF ProcessId
notifyDQMetrics
timeoutDuration
Timeout Duration
PT1H
configureSubscriptionDueDate
The due date expressed in duration for task 'configureSubscription'.
B1M
configureSubscriptionTaskNotificationE...
Send notification emails for task 'configureSubscription'.
true
Start Label
Edit
Manage DQ Subscriptions
Start Events
Edit
There are no events attached to this workflow.
Roles
Edit
Start workflow
Sysadmin
Stop workflow
Sysadmin
Reassign tasks
Sysadmin
Other
Any signed in user can start the workflow.
Any signed in user can start a workflow, independent of the role that user has.
Perform candidate user check on workflow start.
Workflow will fail to initialize if it contains a task without any candidate user.
This workflow can only run once at the same time on a specific resource.
Lock resource
This workflow cannot run with other workflows on the same resource simultaneously.
Show in global create.

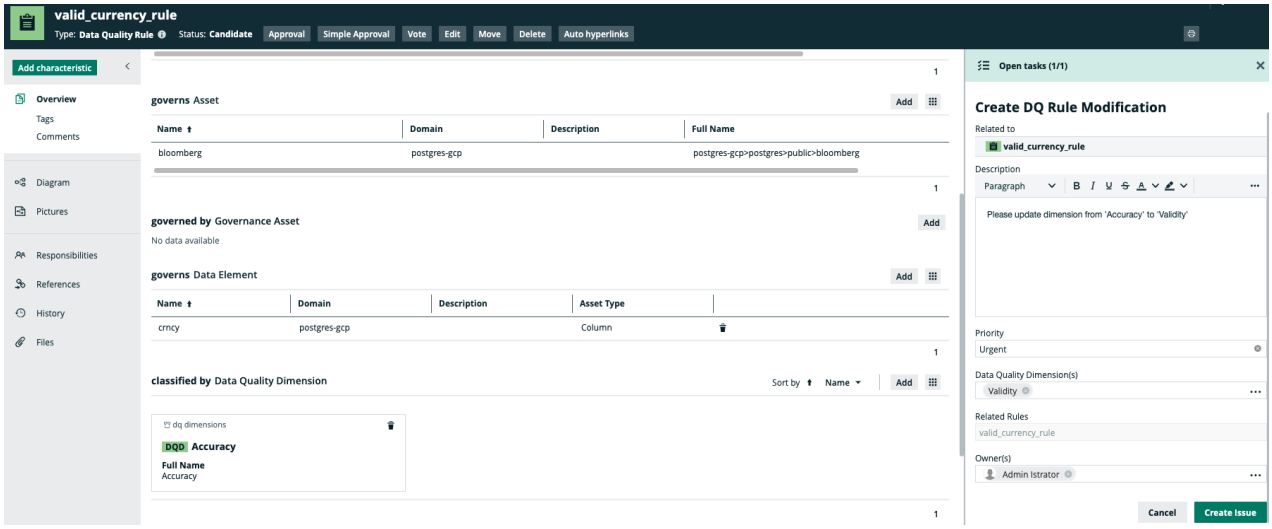
Step 2: Create Data Quality Requests / Issues

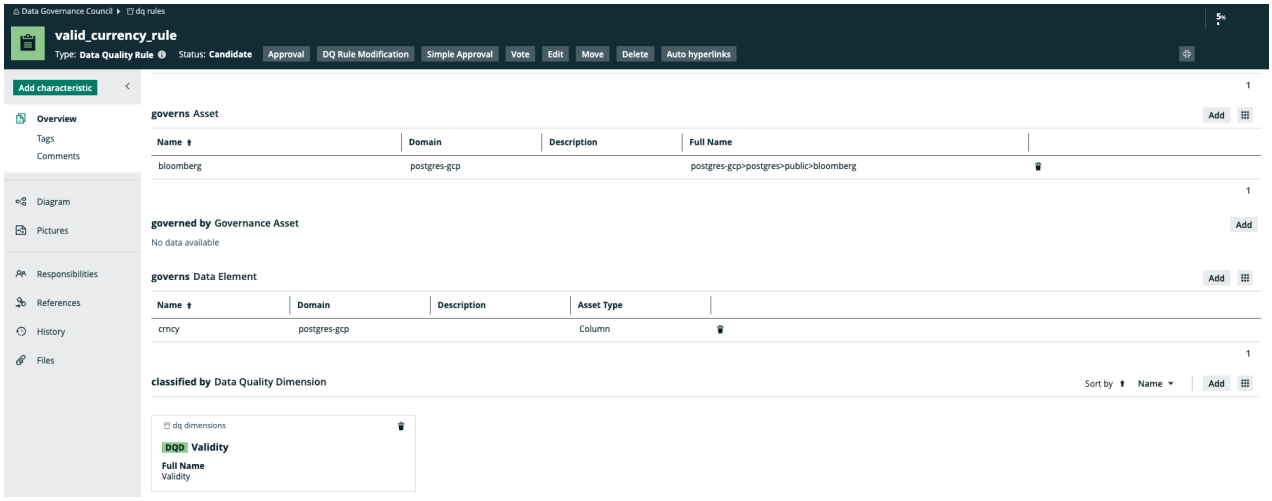
2A. Create Data Quality Issues

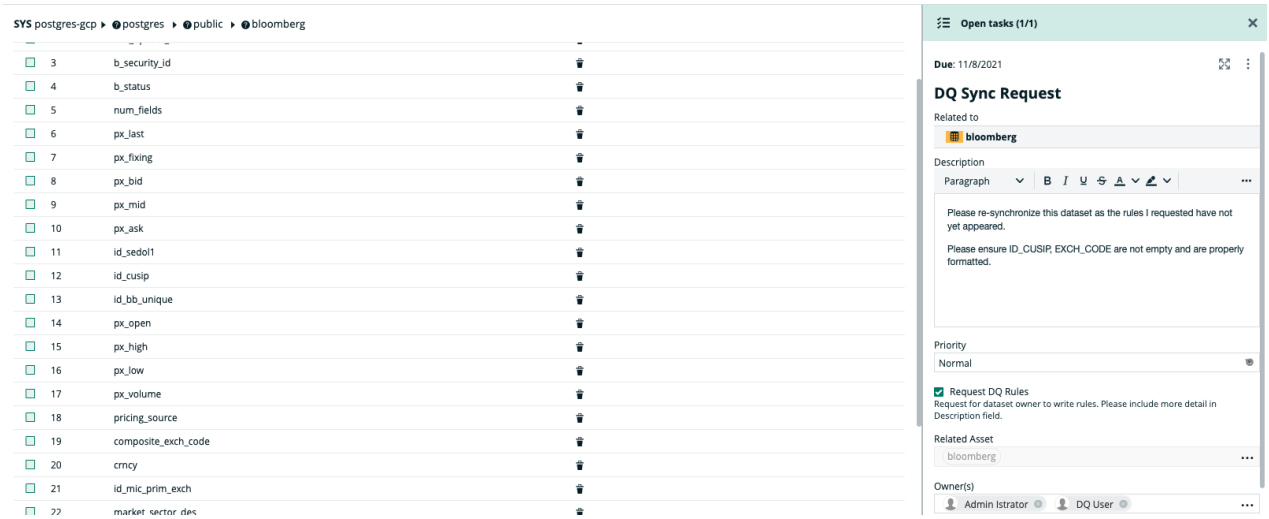
Workflow	Main Requestor Persona	Description	Steward Taking Action
DQ Data Remediation	Data Steward Business Analyst	Tracking / management for confirmed data issues which may require underlying data remediation	Data Lake Admin, ETL Engineer
DQ Rule Request	Business Analyst	Proposing data quality rules in plain language e.g. "flag any German phone numbers in this dataset" or "identify customers with churn risk based on engagement time with our platform"	Data Steward
DQ Rule Modification	Business Analyst	Proposing adjustments to existing rules e.g. values, dimensions, passing thresholds	Data Steward
DQ Sync Request	Business Analyst	Request for synchronization of the DQ Connector to synchronize and/or onboarding a new dataset with pre-populated rules	Data Steward




Data Governance Council
dq rules
valid_currency_rule
Type: Data Quality Rule
Status: Candidate
Approval
DQ Rule Modification
Simple Approval
Vote
Edit
Move
Delete
Auto hyperlinks
Add characteristic
Overview
Tags
Comments
Diagram
Pictures
Responsibilities
References
History
Files
Description
No value has been given yet. Double click or use the edit button.
Predicate
crncy not in ("USD", "EUR", "AUD", "GBp", "GBP", "CAD")
Measurement
No value has been given yet. Double click or use the edit button.
executed by Data Quality Metric
Add
Name
Domain
Description
Asset Type
Full Name
valid_currency_rule
DQ Metrics
Data Quality Metric
bloomberg>valid_currency_rule>Metric
1
governs Asset
Add
Name
Domain
Description
Full Name
bloomberg
postgres-gcp
postgres-gcp>postgres>public>bloomberg
1


valid_currency_rule
Type: Data Quality Rule
Status: Candidate
Approval
Simple Approval
Vote
Edit
Move
Delete
Auto hyperlinks
Add characteristic
Overview
Tags
Comments
Diagram
Pictures
Responsibilities
References
History
Files
governs Asset
Add
Name
Domain
Description
Full Name
bloomberg
postgres-gcp
postgres-gcp>postgres>public>bloomberg
governed by Governance Asset
Add
No data available
governs Data Element
Add
Name
Domain
Description
Asset Type
crncy
postgres-gcp
Column
classified by Data Quality Dimension
Sort by
Name
Add
dq dimensions
DQD Accuracy
Full Name
Accuracy
Open tasks (1/1)
Create DQ Rule Modification
Related to
valid_currency_rule
Description
Paragraph
Please update dimension from 'Accuracy' to 'Validity'
Priority
Urgent
Data Quality Dimension(s)
Validity
Related Rules
valid_currency_rule
Owner(s)
Admin Istrator
Cancel
Create Issue


Data Governance Council
dq rules
valid_currency_rule
Type: Data Quality Rule
Status: Candidate
Approval
DQ Rule Modification
Simple Approval
Vote
Edit
Move
Delete
Auto hyperlinks
Add characteristic
Overview
Tags
Comments
Diagram
Pictures
Responsibilities
References
History
Files
governs Asset
Add
Name
Domain
Description
Full Name
bloomberg
postgres-gcp
postgres-gcp>postgres>public>bloomberg
governed by Governance Asset
Add
No data available
governs Data Element
Add
Name
Domain
Description
Asset Type
crncy
postgres-gcp
Column
classified by Data Quality Dimension
Sort by
Name
Add
dq dimensions
DQD Validity
Full Name
Validity



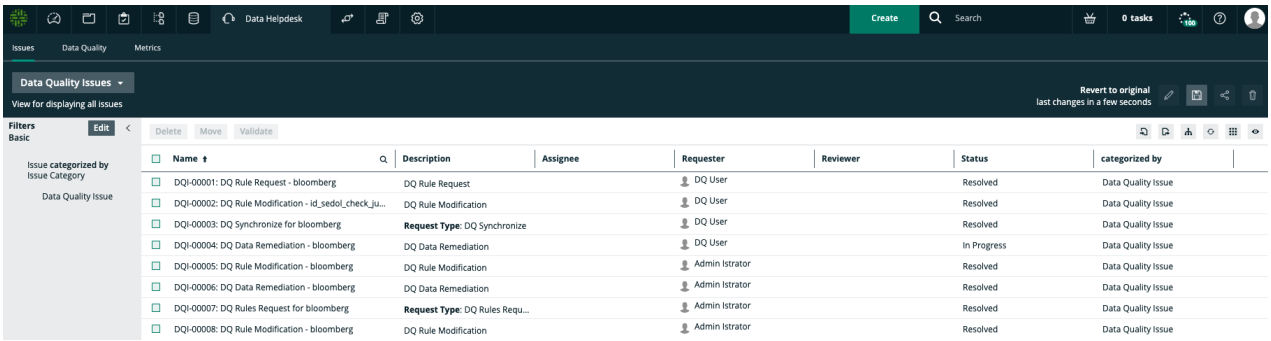
Step 3: Manage Data Quality Issues

3A. Setup Data Helpdesk Filter

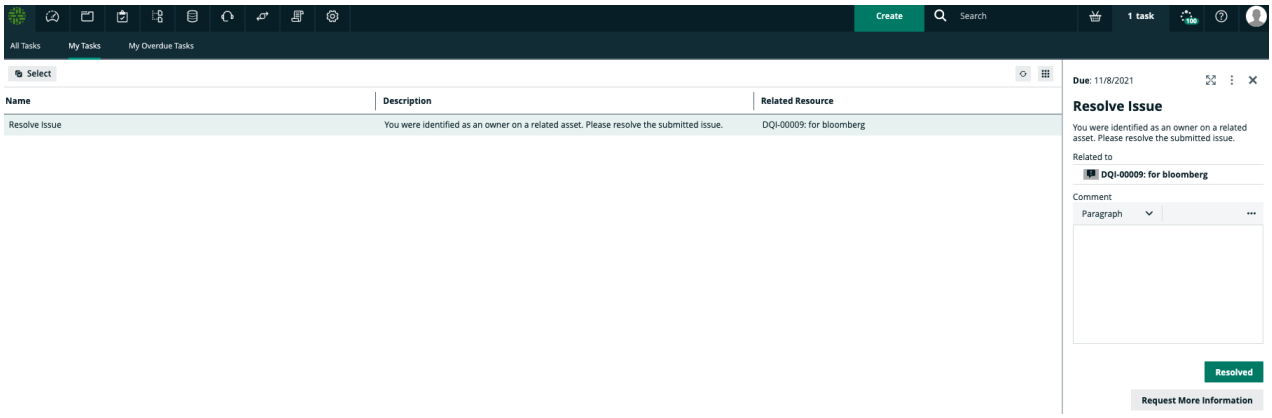
Data Helpdesk

- Select Issues
- Navigate to 'Filters'
- Properties > Attributes > Relations > Issue **categorized by **Issue Category > Input 'Data Quality Issue' > Apply
- Save button > Save View as > **'Data Quality Issues'**
- Optional settings for View: Can pin, promote, make public, make default

3B. Manage Issues From Data Helpdesk View



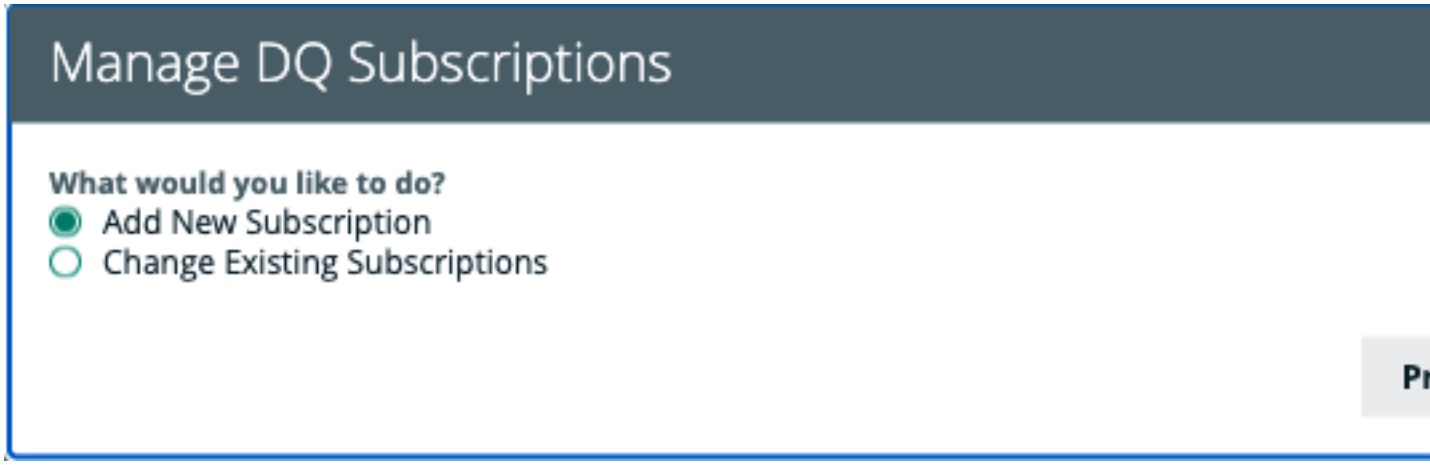
3C. Alternate: Manage Issues From Tasks



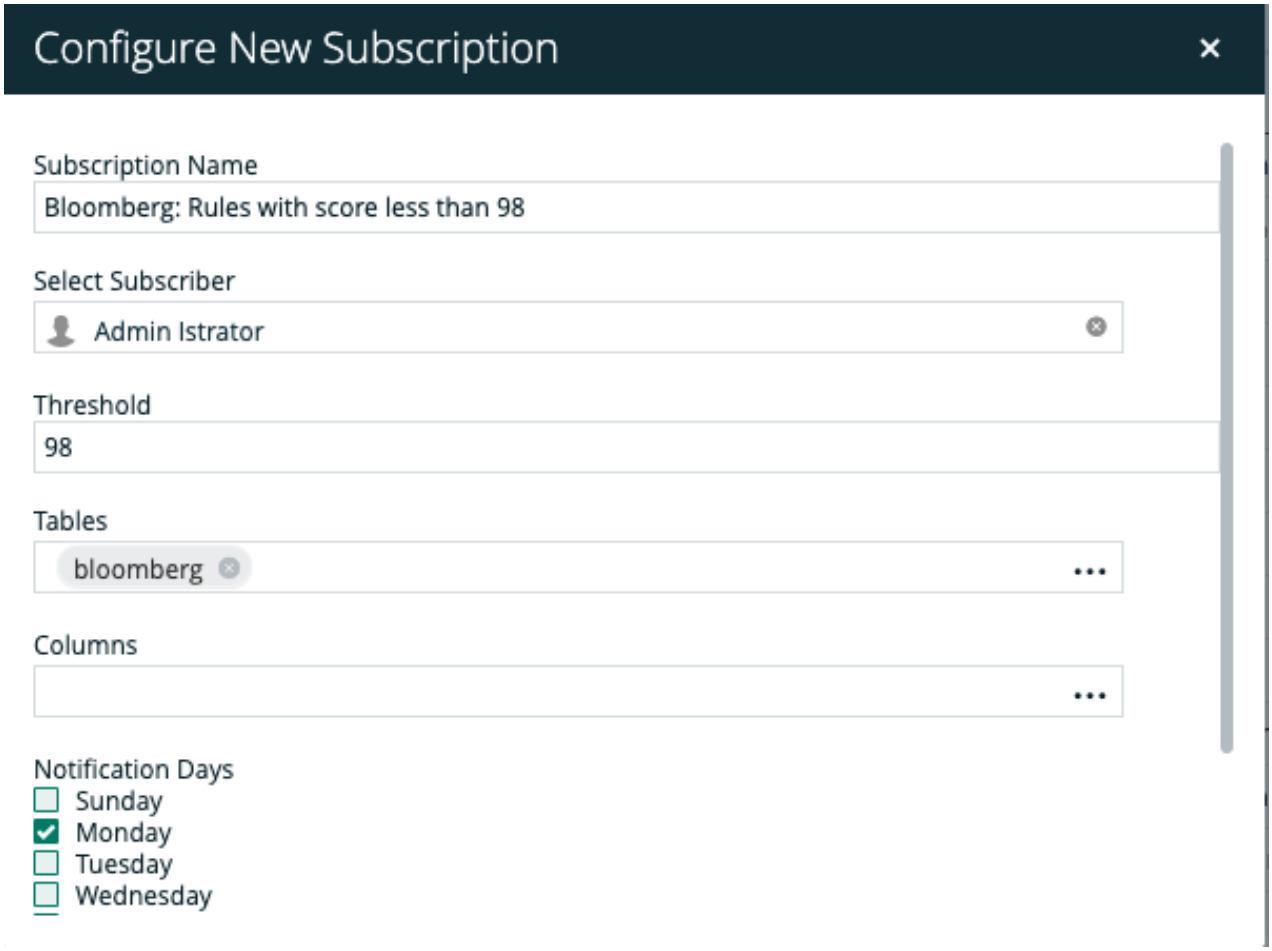
Step 4: Receive Notifications Of DQ Issues And Metrics

4A. Set Up DQ Metric Subscription

Who? Anyone can set up a DQ subscription, for yourself or for your teammates.



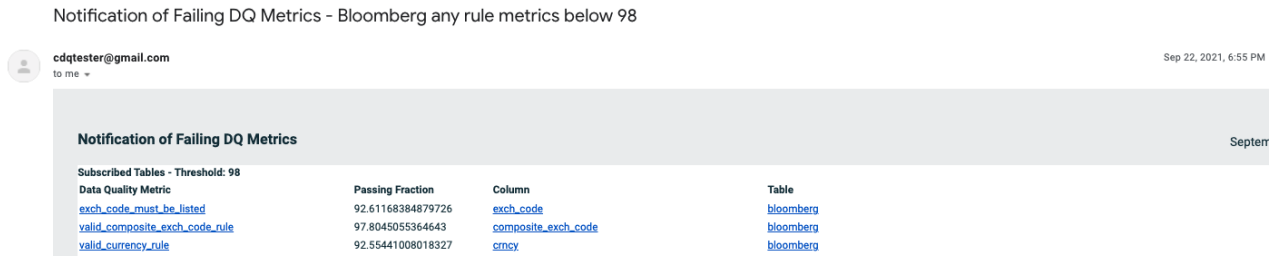
Alerts will be sent based on reviewing rules and metrics associated with **Tables** or **Columns** that violate the specified **Threshold**.



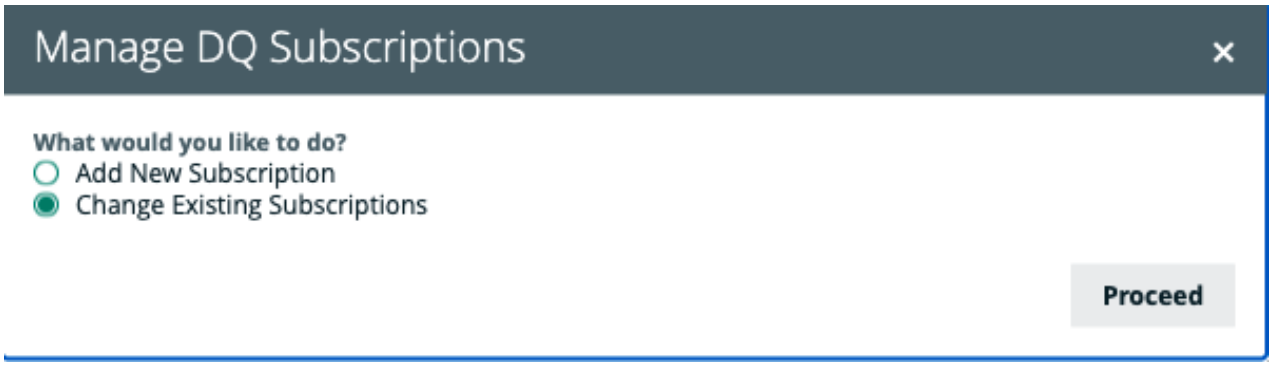
Assuming an e-mail is associated with the Subscriber within Collibra, the Subscriber will receive e-mail notifications by default at **12pm local server time**. This, along with other settings within the provided workflow, can be **adjusted** in **Eclipse**, Collibra's recommended workflow editor.

4B. Review DQ Metric Alerts

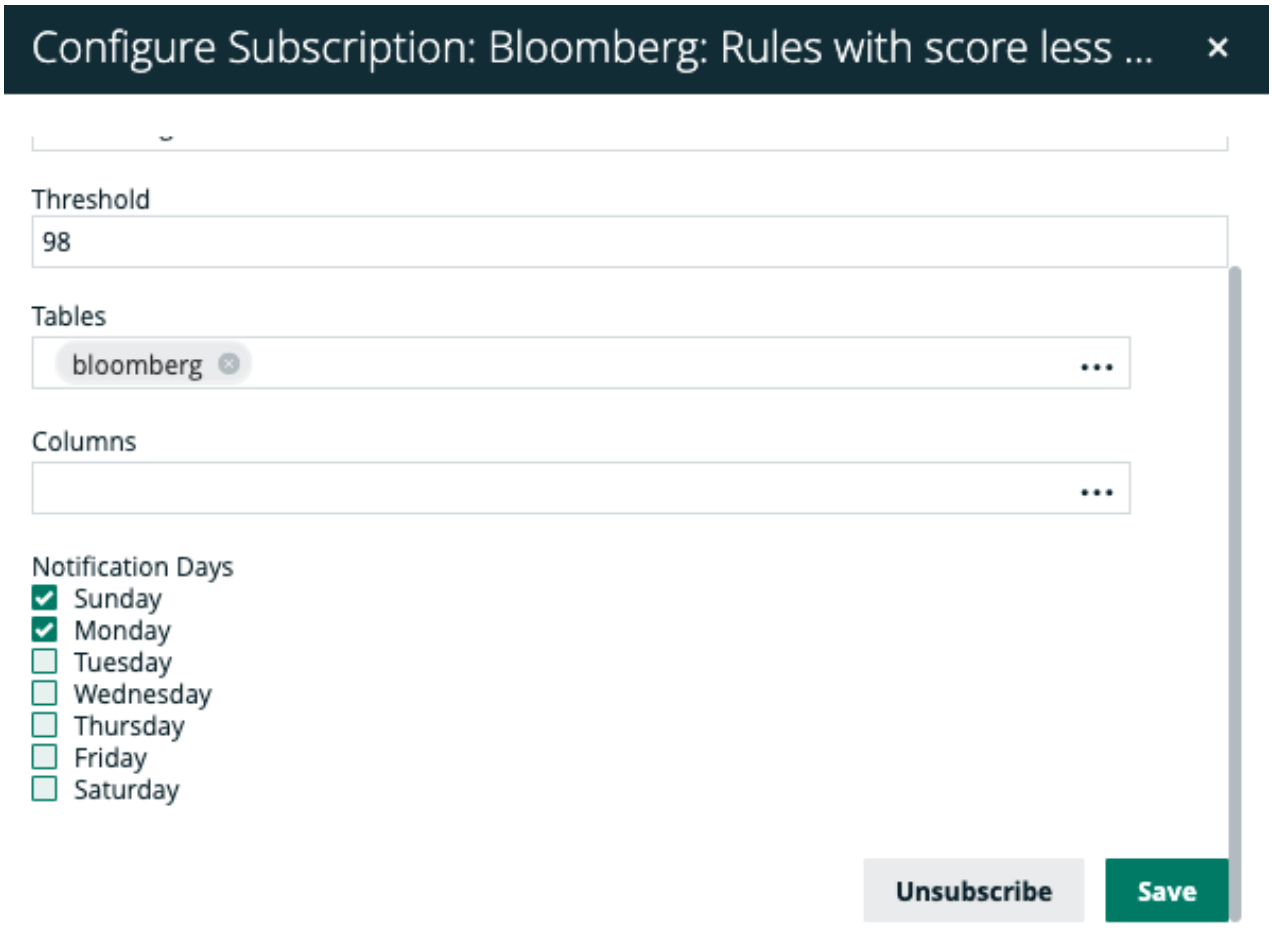
Ensure that the DQ alerts set for you are providing helpful details.



4C. Update Subscription Settings



For every subscription set up for a Subscriber, the Manage DQ Subscriptions modal will cycle through for your review. You can update **Threshold**, add or delete **Notification Days**, add or delete **Tables** or **Columns**, rename the **Subscription title**, **Save the new settings**, or simply **Unsubscribe**.



Chapter 14

Collibra DQ Catalog

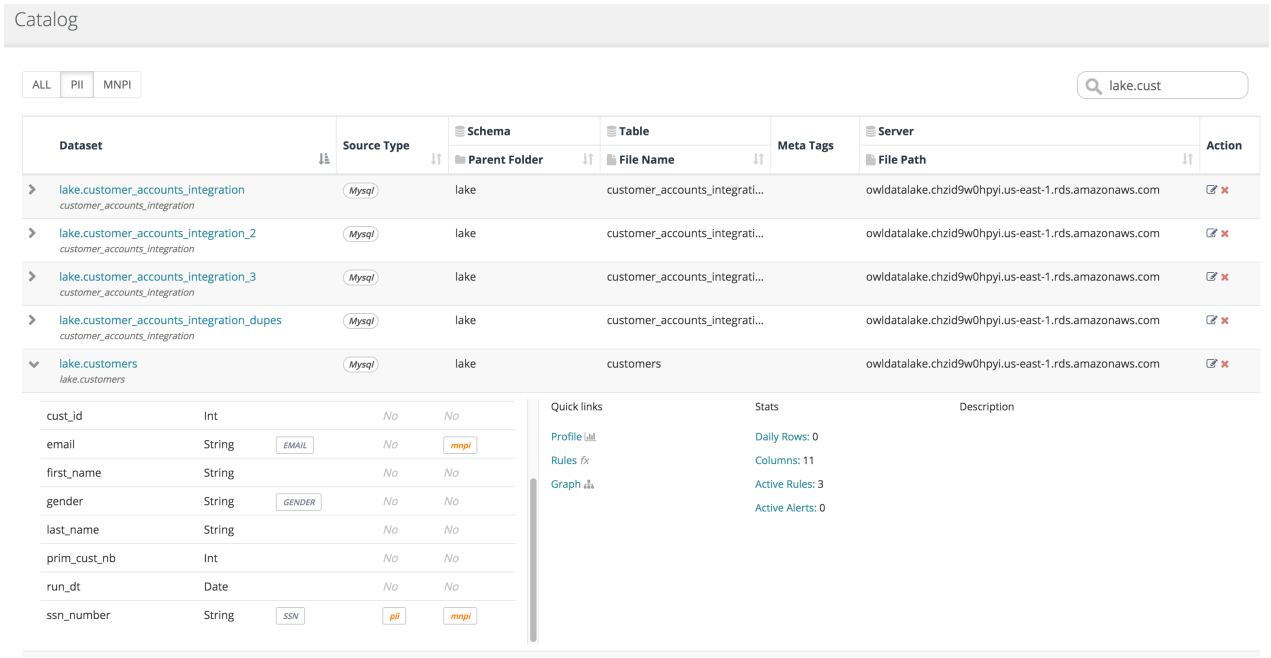
Overview

Smart Catalog - Bringing Data Science to Cataloging

While Collibra DQ does not pride itself on being a catalog tool it does automatically maintain a dataset and process catalog. It is a necessary control for DQ and helpful to the end user. Without a smart catalog a user could technically overwrite another user's OwlCheck (DQ check). For example -ds "Trade" and -ds "Trade". DQ believes a healthier habit is to store the full natural name of the dataset and allow the user to alias the name in the event that they wish to make a short-name. By doing this DQ protects users from mixing up their results and stops the constant renaming of common objects which leads to more unnecessary business level mapping. This approach makes it effortless for a user to create a new OwlCheck in the wizard because DQ will warn the user if there is a naming collision. DQ learns all the server hosts, the database schemas and table names and keeps things automatically organized. One less catalog to setup, manage and eventually untangle.

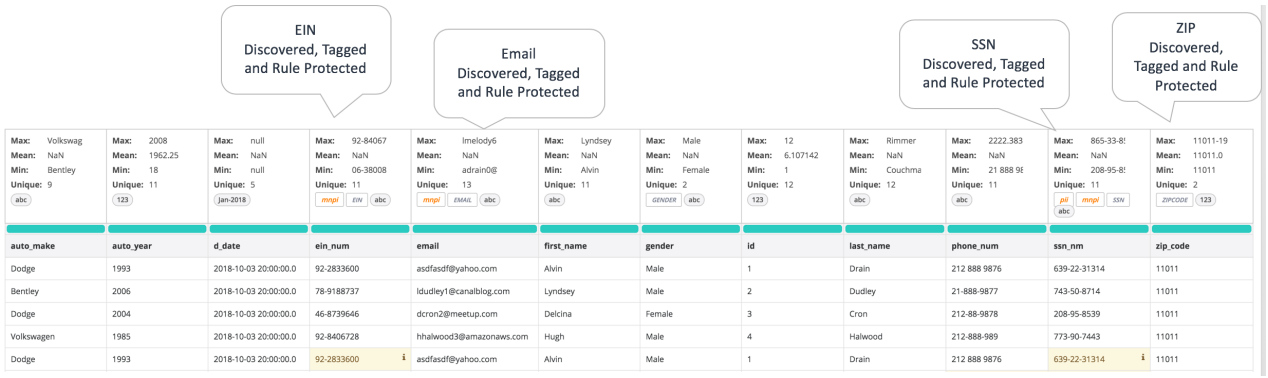
Automatic Sensitive Data (PII) Detection

DQ automatically understands the *semantic* schema of your data such as CREDIT CARD, EMAIL, SSN and much more. Additionally, DQ will label sensitive data with PII and MNPI classifications.

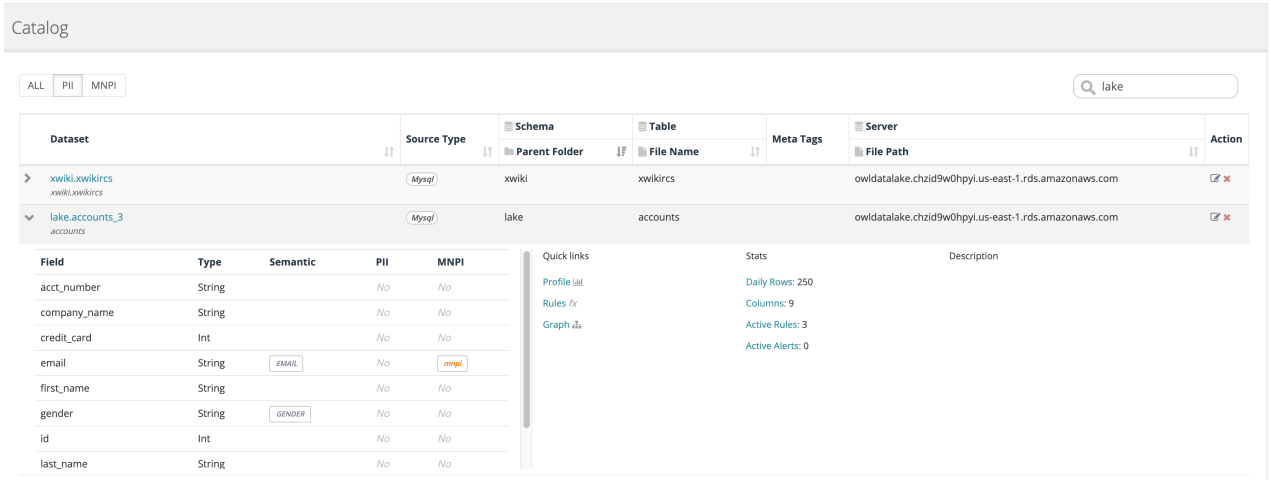


Data Table View of PII

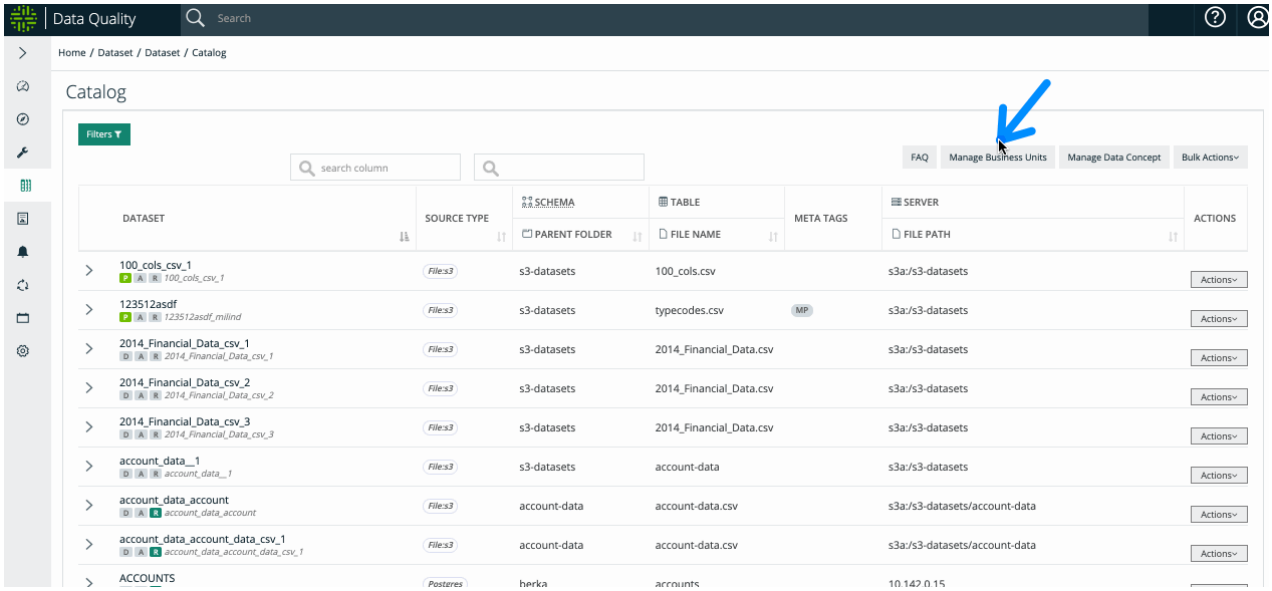
DQ applies many labels to the header of a field / column. These labels be seen in the data preview table with highlighted errors and findings.



The catalog offers a global view and filtering to see where PII exists

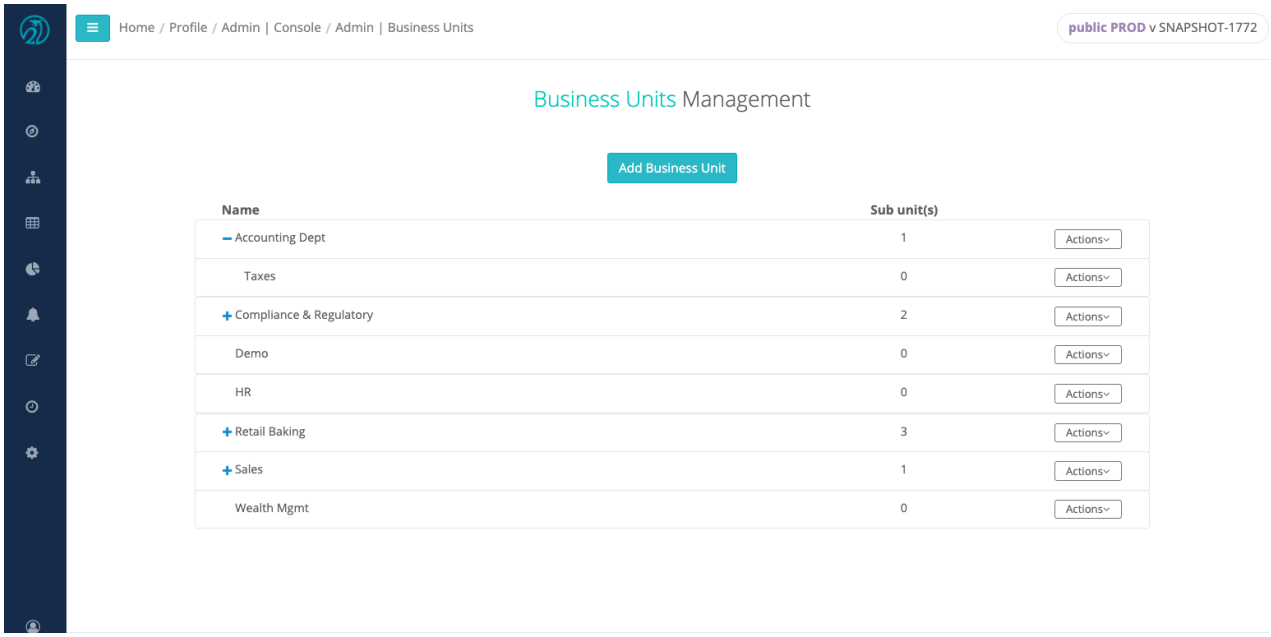


Business Units

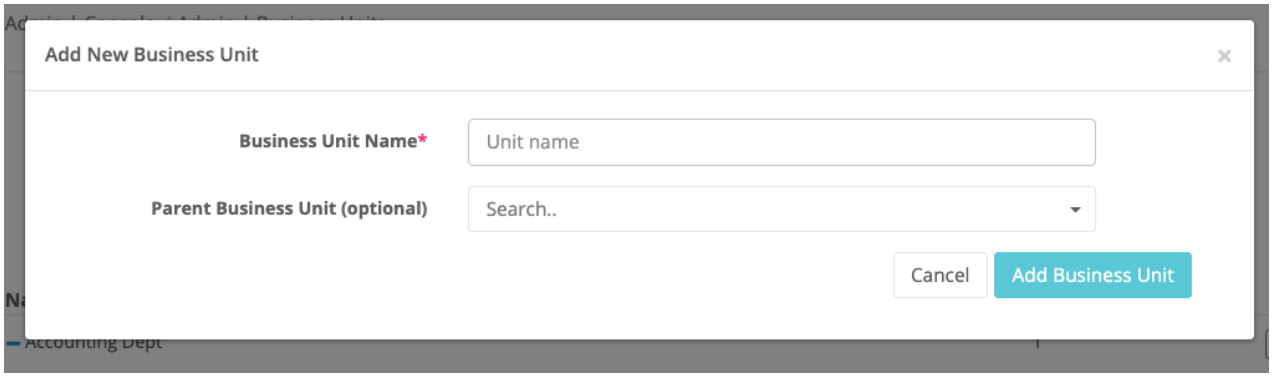


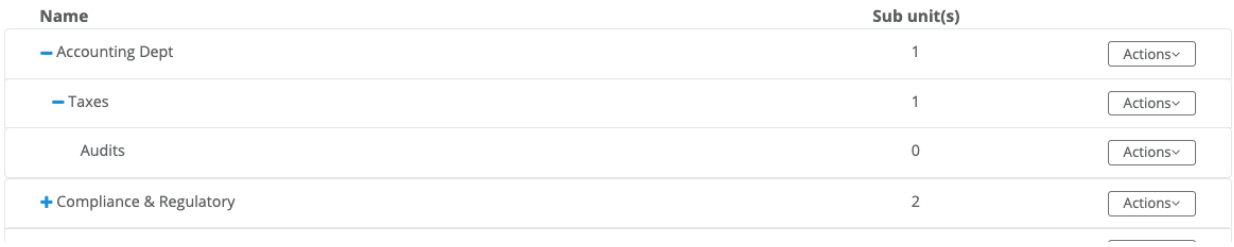
Collibra DQ allows its users to run large sets of DQ jobs. Business units provide a way to categorize and group these.

Through the admin console the user can navigate to the business units management page. This page will contain a table of your business units and a button to create new units.

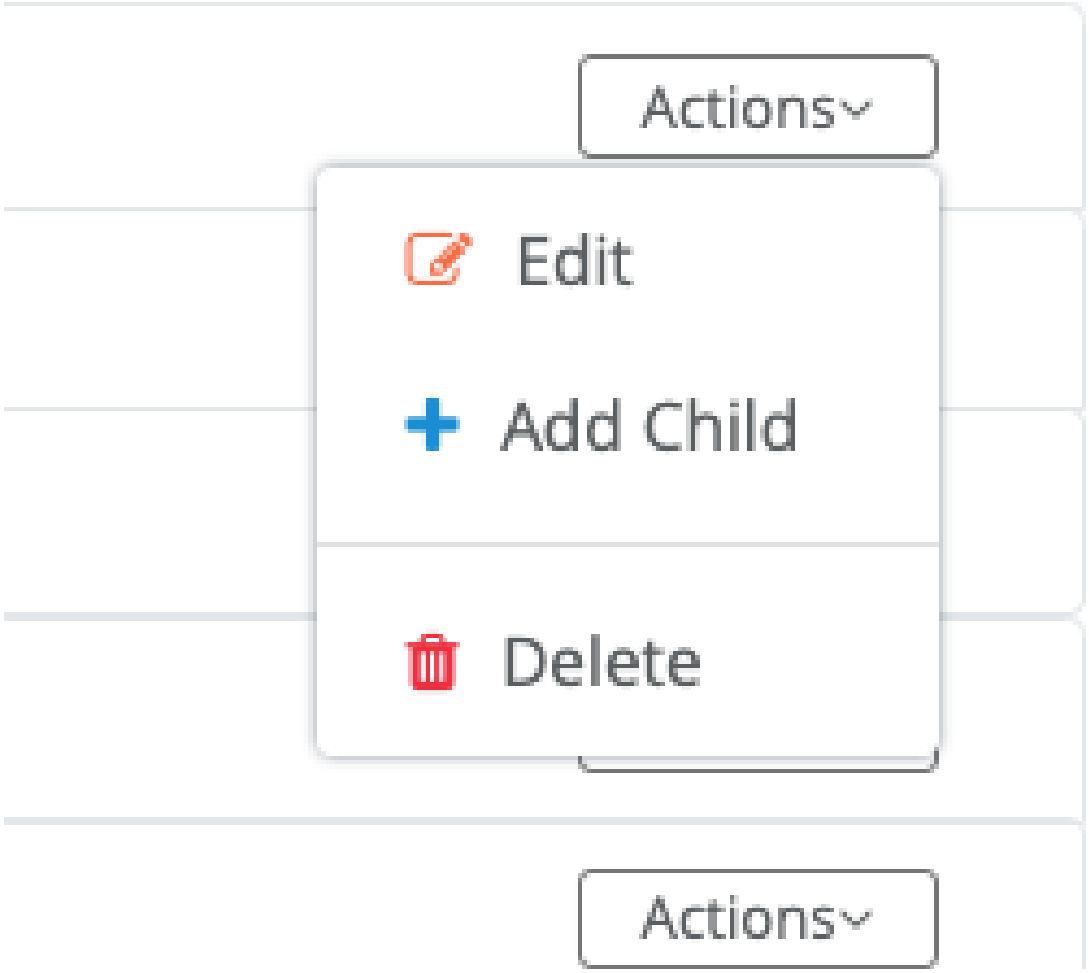


By clicking the Add Business Unit button the user will be able to fill out the business unit form. A unique name is required and a parent business unit can be optionally selected. Each unit can have one parent and many children. On the business units table you can click the blue plus icon to expand all children for that unit. If any children have children unit they will also have a plus sign to indicate the children rows can be expanded.

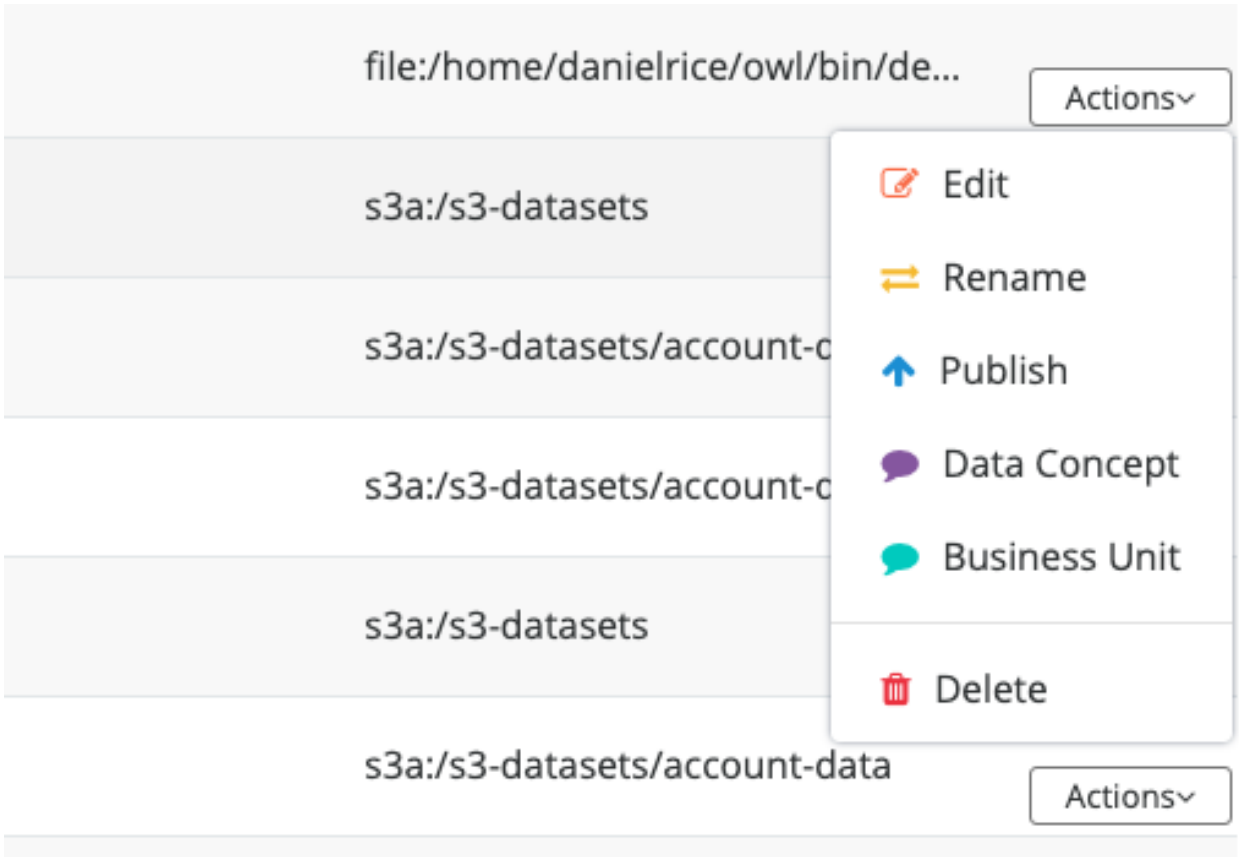


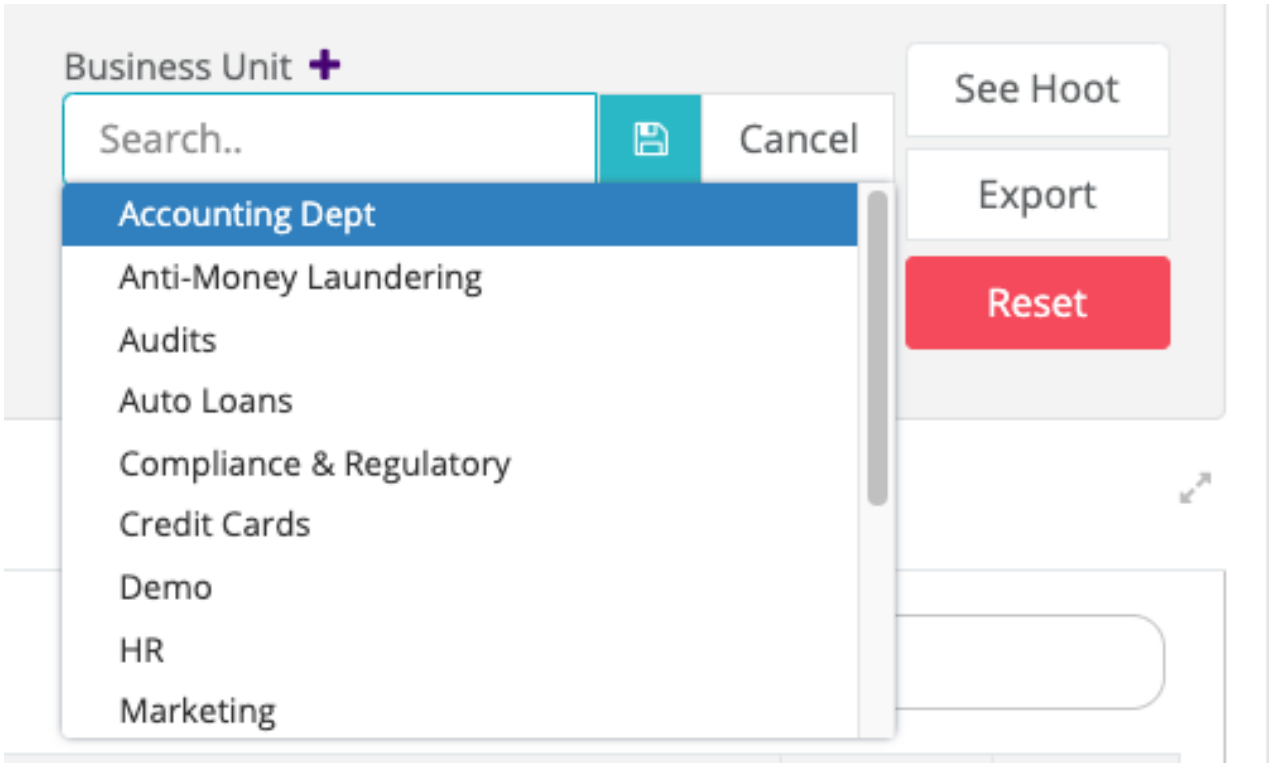


Each business unit has actions. The business unit can be edited or deleted. If the unit is assigned to at least one data set, it cannot be deleted. If the user wishes to create children of a unit they can choose the Add Child action to have the new business unit form pre-populated with the selected unit as the parent.

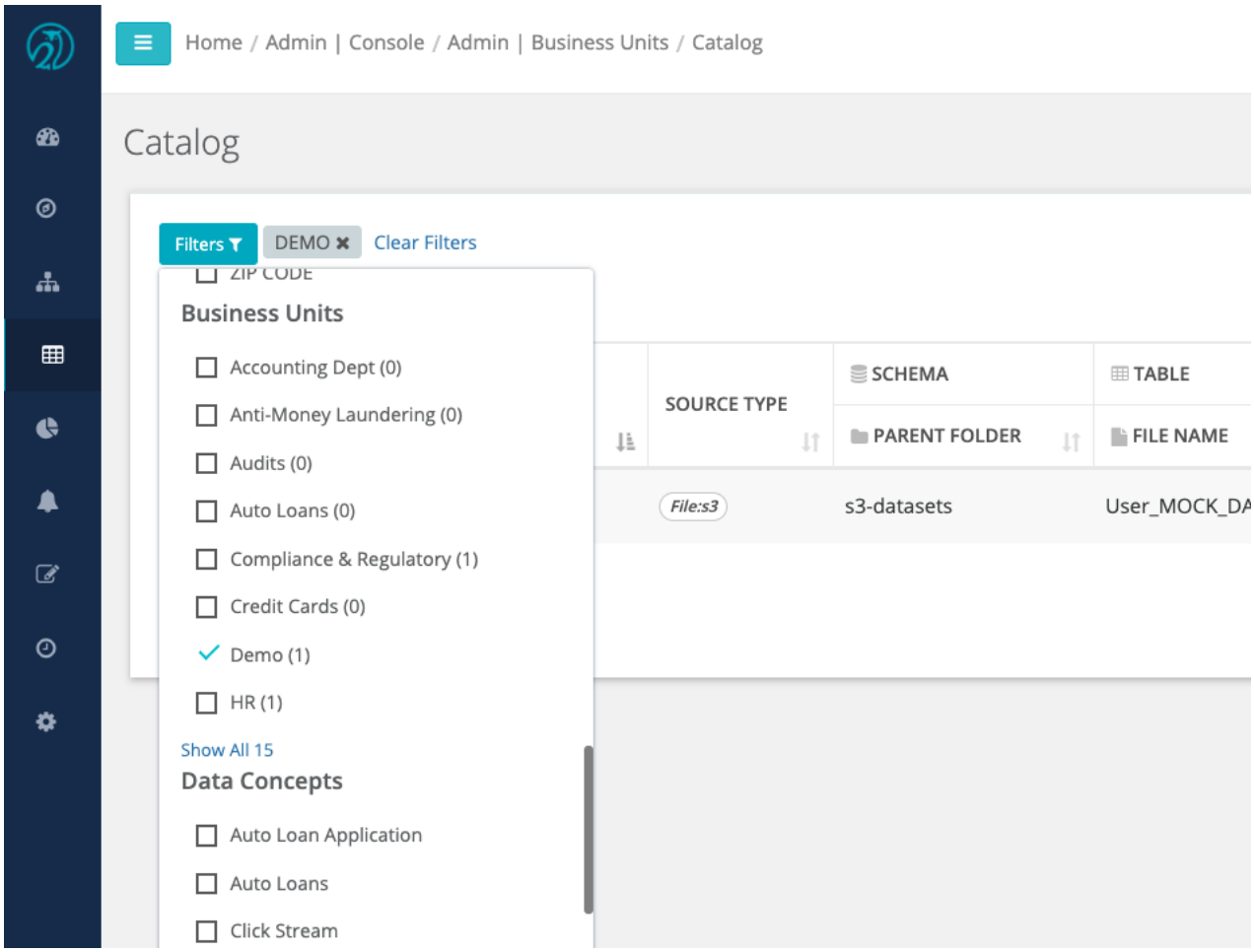


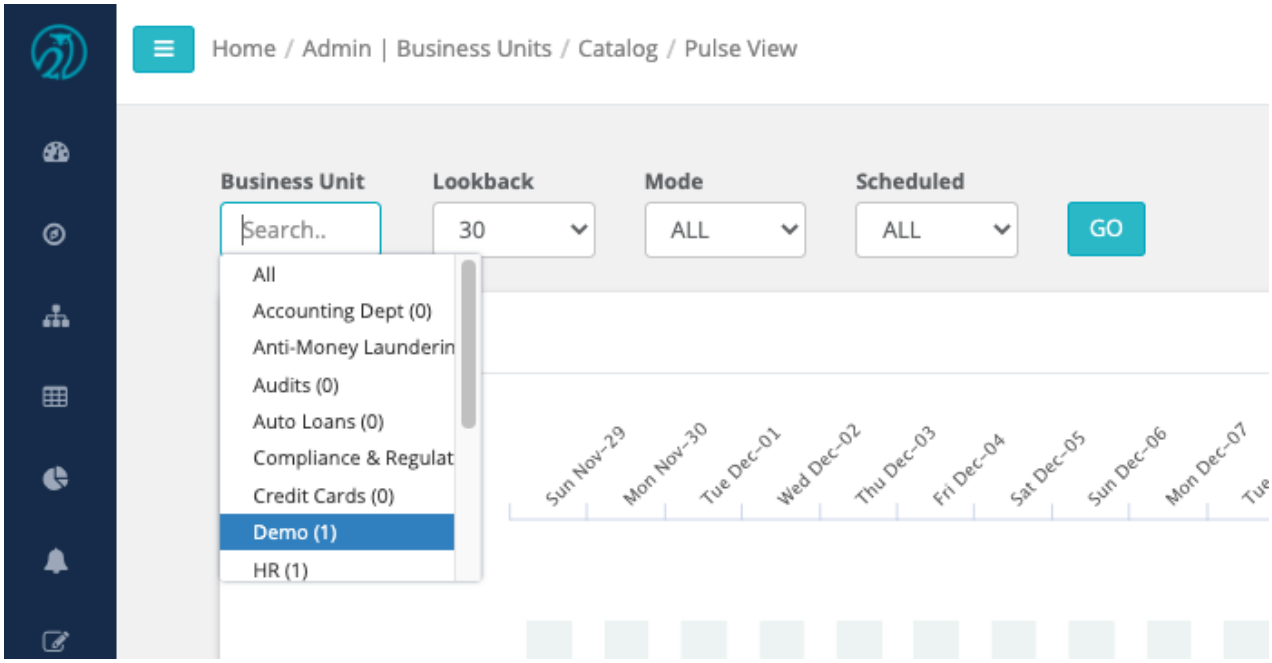
There are two different ways to assign a business unit to a data set. On the Catalog page there is an action option to manage the business unit. One the profile page the user can click the add/edit business units icon to open a control to manage the assigned business unit.


file:/home/danielrice/owl/bin/de...
Actions
s3a:/s3-datasets
s3a:/s3-datasets/account-c
s3a:/s3-datasets/account-c
s3a:/s3-datasets
s3a:/s3-datasets/account-data
Actions
Edit
Rename
Publish
Data Concept
Business Unit
Delete

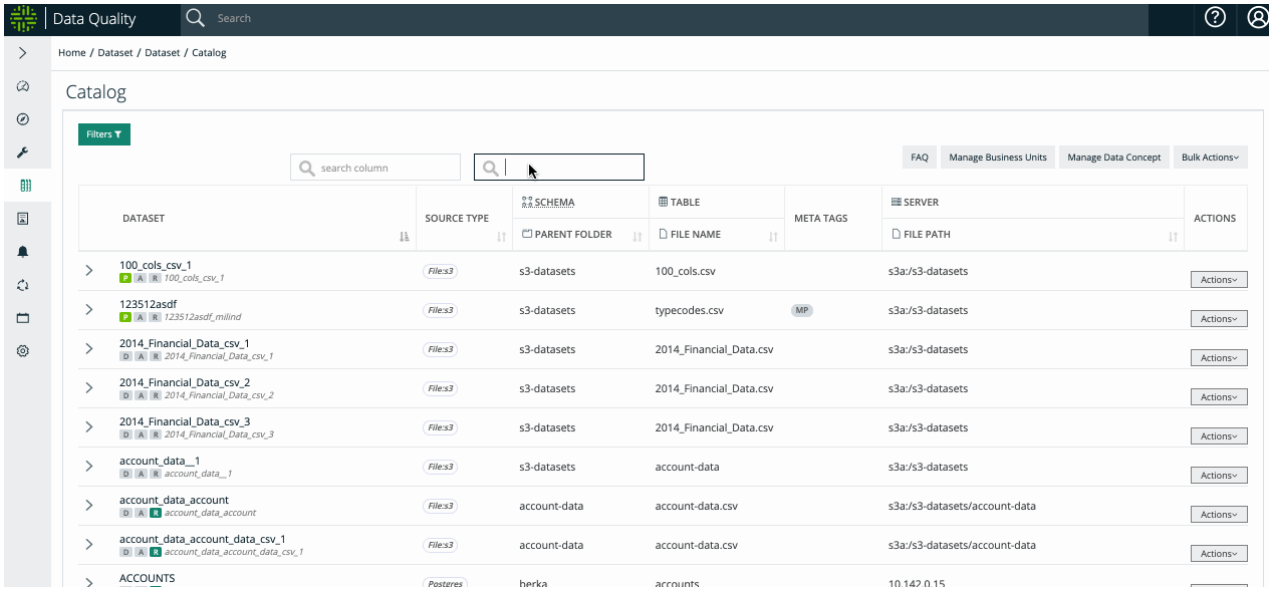

Business Unit
Search..
Cancel
See Hoot
Export
Reset
Accounting Dept
Anti-Money Laundering
Audits
Auto Loans
Compliance & Regulatory
Credit Cards
Demo
HR
Marketing

Once a business unit has been assigned to a data set the user can filter by that business unit on the Catalog and Pulse View.



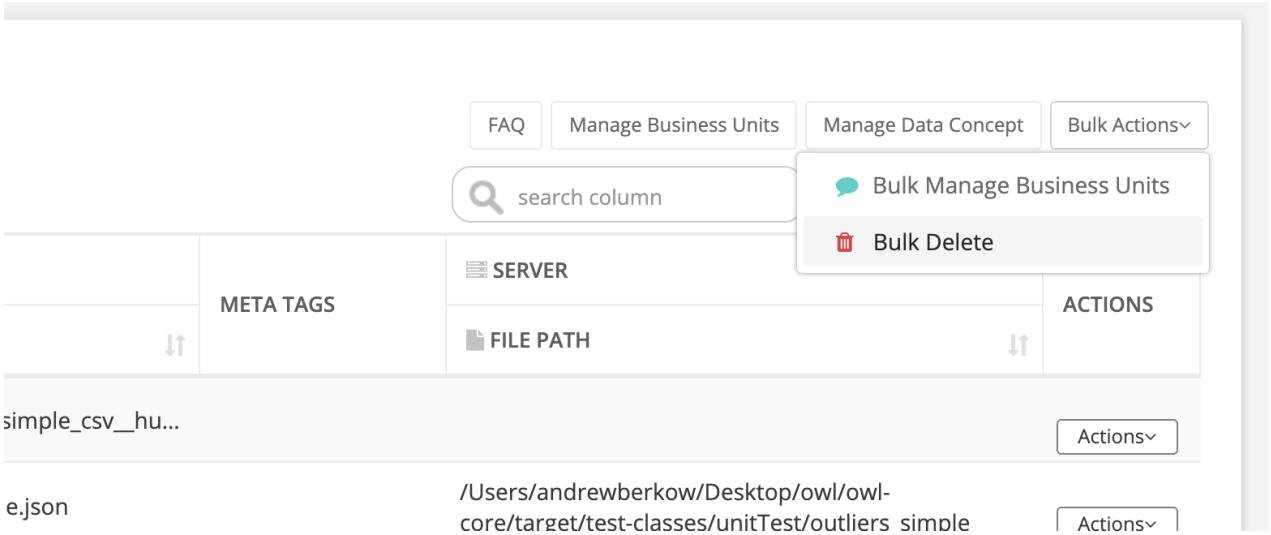


Catalog Bulk Actions

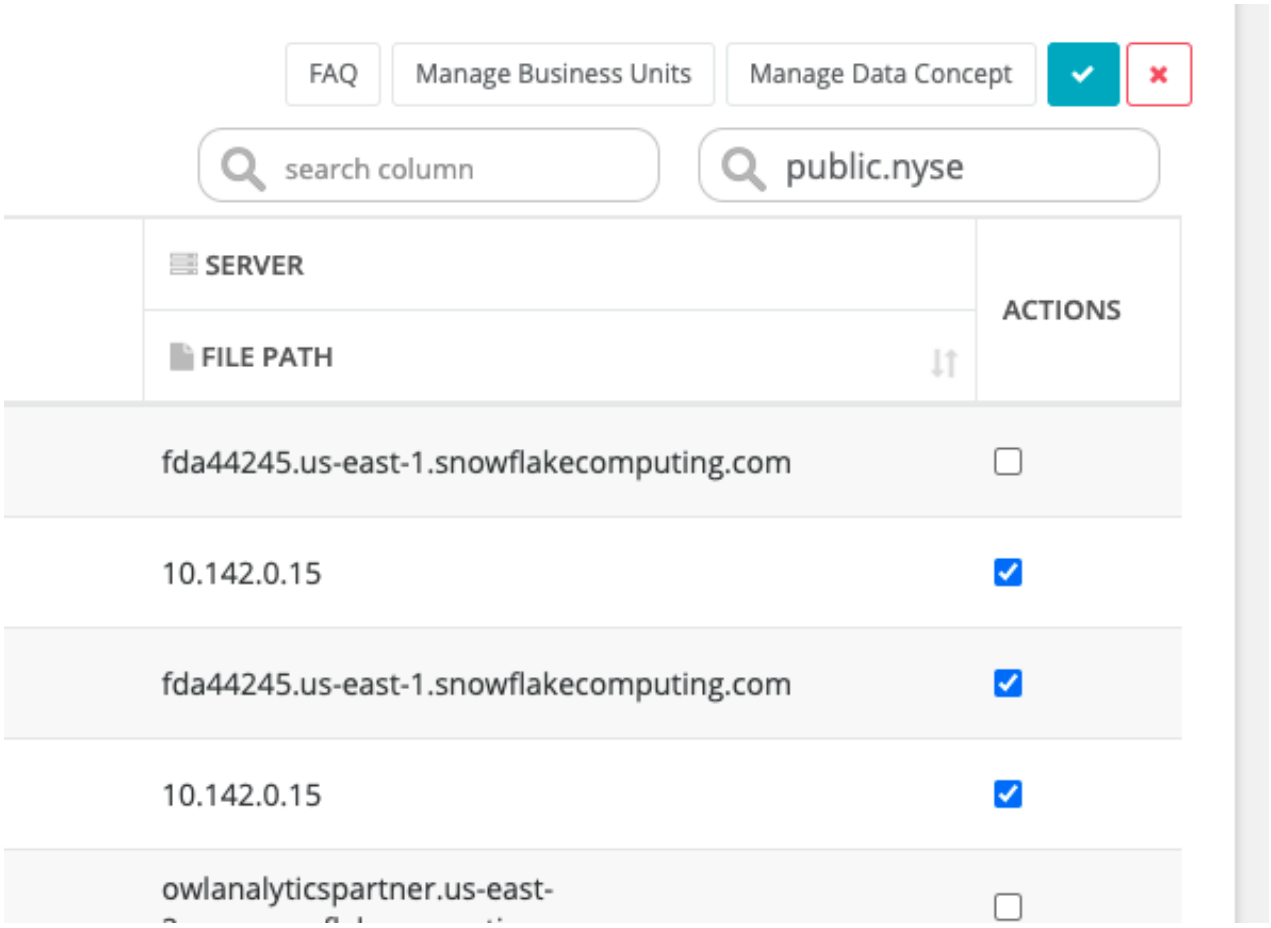


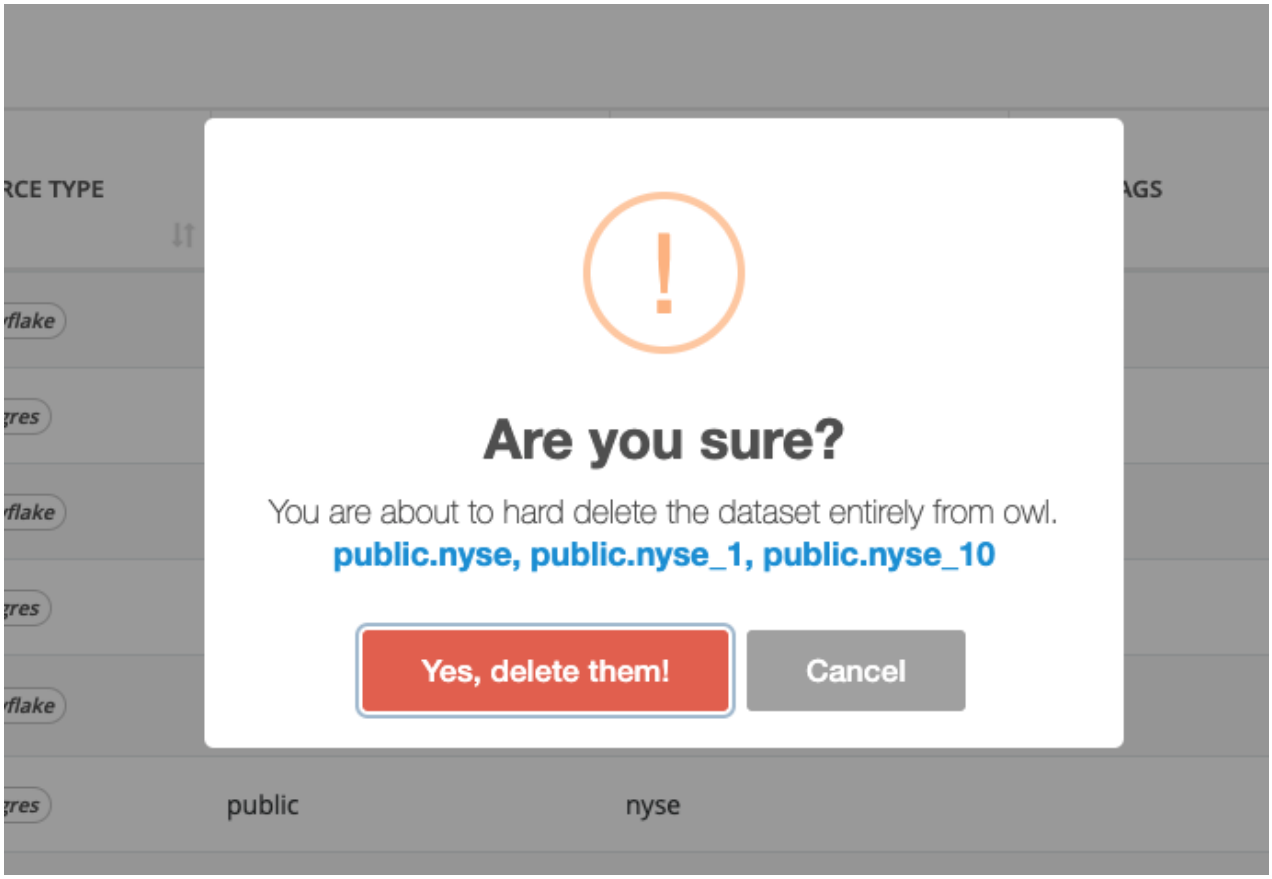
Bulk Delete

The catalog allows user to delete multiple datasets in one action through the catalog. Click the Bulk Actions button dropdown in the top right corner. Select Bulk Delete.



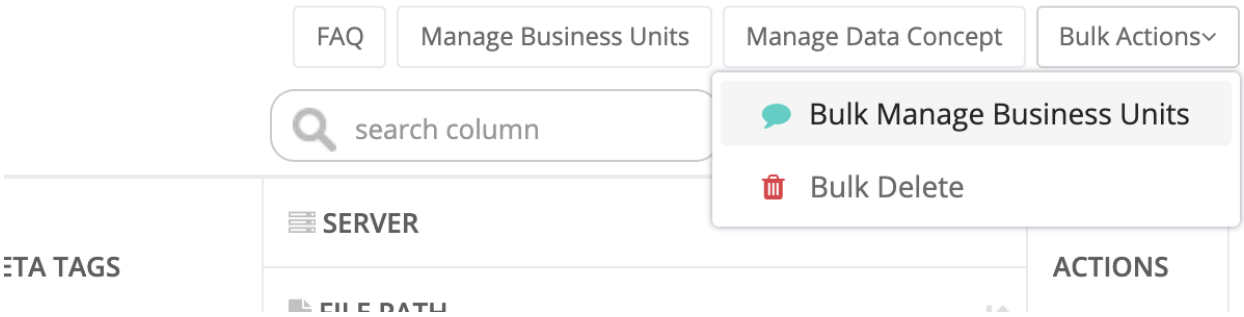
Each data set row will now have a checkbox in the Actions column. All filters can be applied now and the user can check all data sets they wish to delete. Once desired data sets are selected, the checkbox can be clicked. The user will be prompted to confirm they wish to delete data sets. The user can also click the X button to cancel.



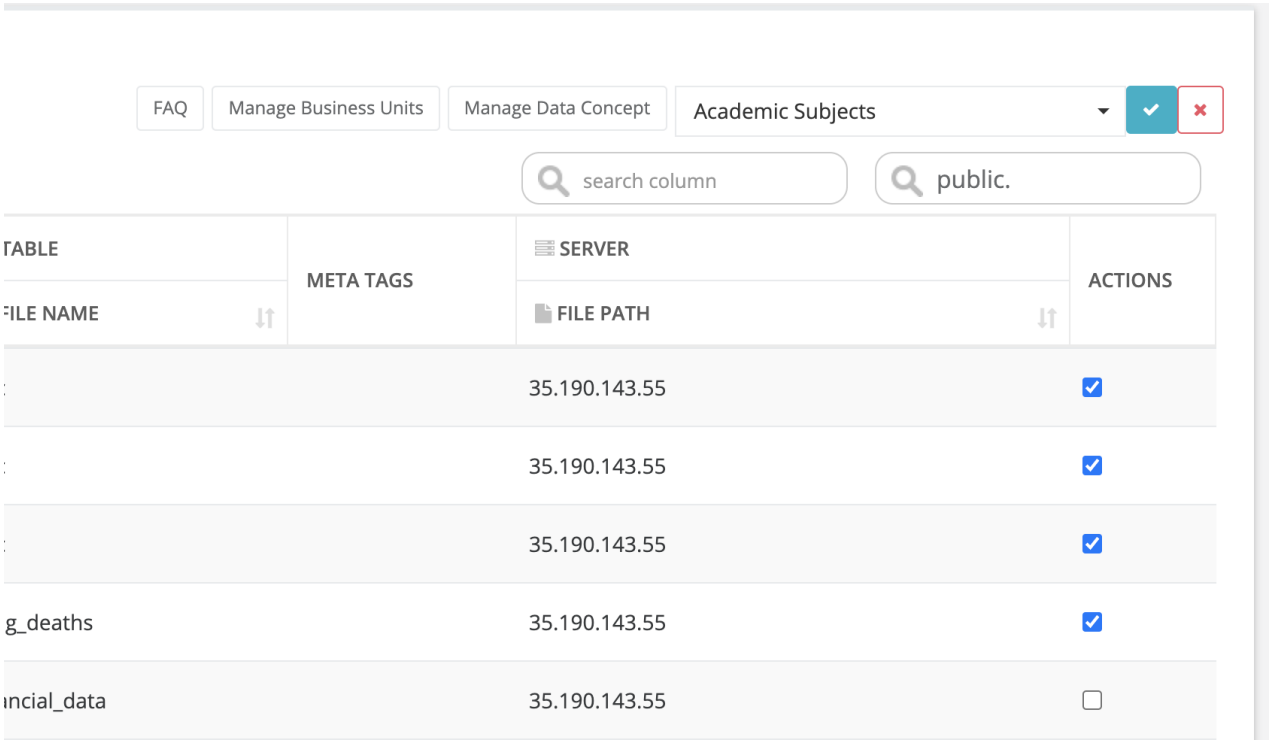


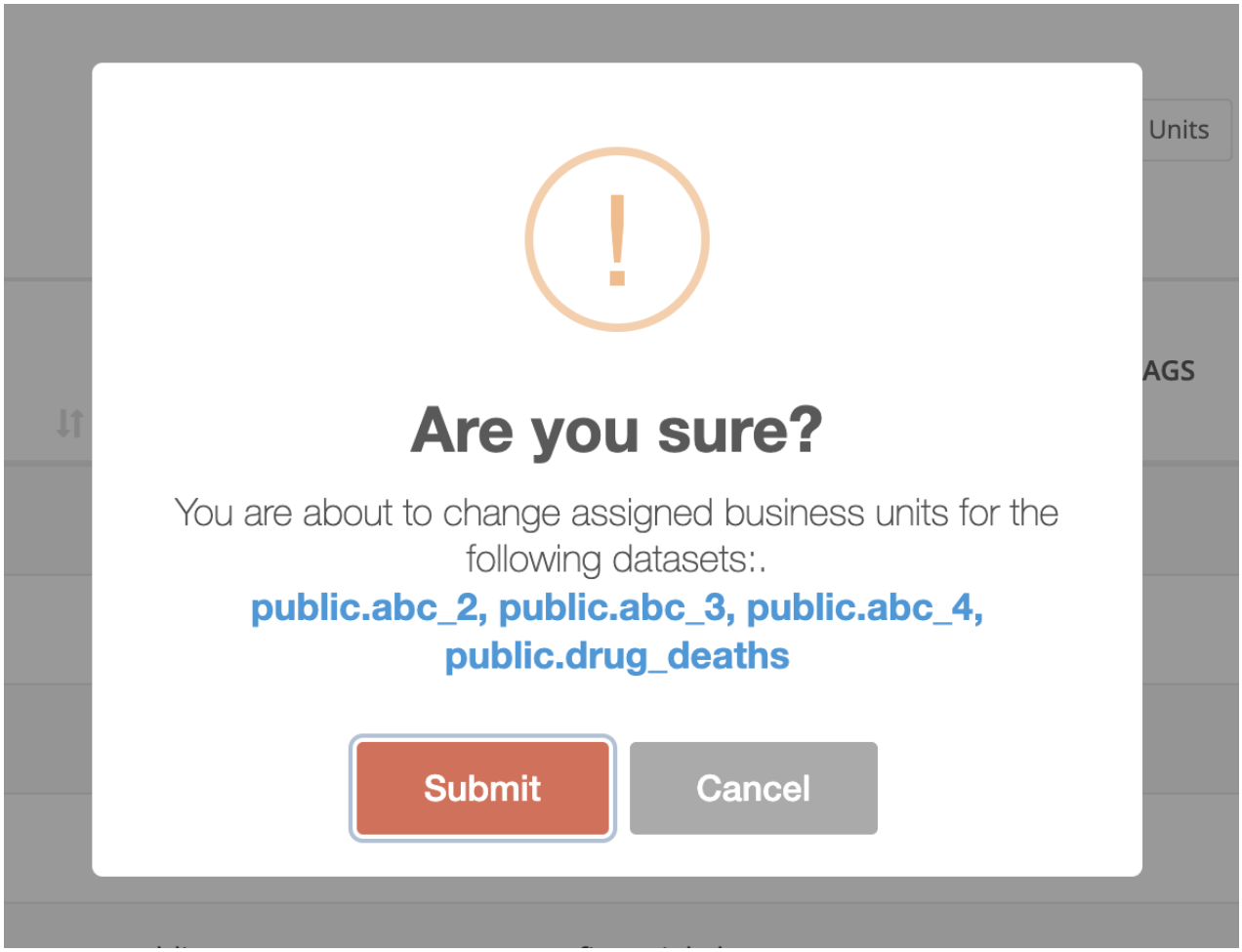
Bulk Manage Business Units

Another bulk feature allows the user to manage the assigned business unit to multiple datasets. Click the Bulk Actions button dropdown in the top right corner. Select Bulk Manage Business Units.



Each data set row will now have a checkbox in the Actions column. All filters can be applied now and the user can check all data sets they wish to delete. Once desired datasets are selected, the user can select a business unit from the business unit dropdown control in the top right corner and click the checkbox. The user will be prompted to confirm they wish to change the assigned business unit to the data sets.




Units
AGS
Are you sure?
You are about to change assigned business units for the following datasets:.
public.abc_2, public.abc_3, public.abc_4, public.drug_deaths
Submit
Cancel

Chapter 15

Collibra DQ Solutions

Use Cases

Data Projects

Builds a Better DQ Dashboard

- **DQ Dashboards.** Many DQ problems result from an improper or a too slow observation of business rules related to the data. What is not caught by handmade visual inspection or a potentially outdated man-made rule can only be flagged by AI Machine Learning. Conversely, what does get flagged should also be easily triaged and then immediately fixed with the aid of AI. The most important metric for a DQ Dashboard is the time to fix, not simply the overall DQ score.

Ensures CCPA & GDPR

Aggregation from hundreds of locations: The dashboard for what is within spec based on AI observation – not handmade rules.

Then push-down fix: The rules are created and then immediately applied via self-service. The problem is immediately identified and a fix (recommendation engine) is applied.

The value of this for both companies: The DQ problem never corrupts the whole. The longer the bad quality exists the bigger problem it can create.

Makes your Data Lake better.

Data Lakes support analytics, which will ultimately drive actions that increase revenue, support compliance, prevent churn, etc. However, whether that action is near to real-time or not, none of those can be performed without first performing a DQ check. For example, can you trigger an action before first checking the “GDPR Remove” list? A Data Quality

check must always be the first step in any action. OwlDQ with Schema Learned can perform 100+ owl checks. However beyond simply those checks, it is OwlDQ's unique Spark-based architecture listed below that enables innovation. Churn, credit check, AML, infosec checks developed in the Data Lake could be added as part of Owlcheck on the streaming data.

- **Data and Privacy in Place.** Data never has to move for a DQ OwlCheck. The latency saved from operating in place, the added hybrid flexibility, the privacy maintained serves many new use cases that were not possible before. It also removes any unnecessary consolidation for the sake of simply consolidation. DQ doesn't have to start by first moving it into a Data Lake.
- **DQ or Any Rules applied in the Stream.** The DQ rules learned by DQ can be applied back to the source on data in the stream. However, other non-DQ rules learned in the Data Lake can also be added to the OwlDQ check.
- **Self-Service and DQ push-down fix.** DQ can enable a self-service push-down fix (recommendation engine) to anything flagged at the source. The best time to fix DQ is when and where the problem started. This enables tighter integration with Data Governance tools since DQ is maintained at the source once, not downstream where corruption beyond just the data can occur.
- **Multi-cloud/On-prem/Hybrid.** OwlDQ can scan/alert/report at the source or can operate natively on the target Data Lake such as Databricks Delta in Azure or Snowflake on AWS, or Qubole on GCP. Why compromise DQ just because your data is not in one place? Why settle on a DQ strategy that only works if the data is first migrated or moved?
- **DQ Dashboards.** Many DQ problems result from an improper or a too slow observation of business rules related to the data. What is not caught by handmade visual inspection or a potentially outdated man-made rule can only be flagged by AI Machine Learning. Conversely, what does get flagged should also be easily triaged and then immediately fixed with the aid of AI. The most important metric for a DQ Dashboard is the time to fix, not simply the overall DQ score.

Speeds Migrations/Enables Replications

- **Speed Migrations/Enable Replications.** Batch collection with subsequent excel compare is very problematic. Instead, rules are generated by the data itself, and

anomalies are triggered on the fly at the source system. This type of schema learned approach will speed migration and enable replications with much less overhead.

Assists Data Aggregation

- **Greatly Reduce Support Costs.** Apply AI-generated rules to maintain consistency across all accounts, rather than apply hand-made rules per every account. You are not managing 100+ variations of rules, but a consistent set that is learned from all.
- **Capitalize on Real-time.** Batch collection with excel compare will never support real-time. Instead, rules are generated by the data itself, and anomalies are triggered on the fly at the source system.
- **Rapid Onboarding.** Universal and already tested scans are applied quickly.
- **Improve Customer Satisfaction.** Monitor the real-time speed of the DQ pushdown fix, not just the overall DQ score over time.
- **Improved SLAs.** When DQ is fixed immediately, all SLAs can be improved not just DQ SLAs.

Creating a Data Quality Pipeline

Organizations that leverage data as an asset to make decisions into their future must entrust the data, from which important business decisions are derived. While almost all businesses leverage their own collected or generated data (or plan to) for internal use, how many actually scrutinize their data? Companies that sell a product must ensure that it is run through some sort of quality assurance suite of tests/runs before it is available to a customer. So organizations that use data internally as their own asset/product should have the same or more confidence in the quality of their data. Owl-Analytics is a data quality product that observes the data to surface behaviors, patterns, outliers, schema changes, record changes, and more.

Data Scientists are trying to find insights in data to solve complex problems for their business. However, 80 percent of their time is spent discovering the data to cleanse it to make it ready for the model. Over time, models deteriorate as data underneath changes or new trends/ patterns arise. Leveraging Owl-Analytics to validate the quality of the data during the data pipeline and before the data is presented to the Data Scientist reduces

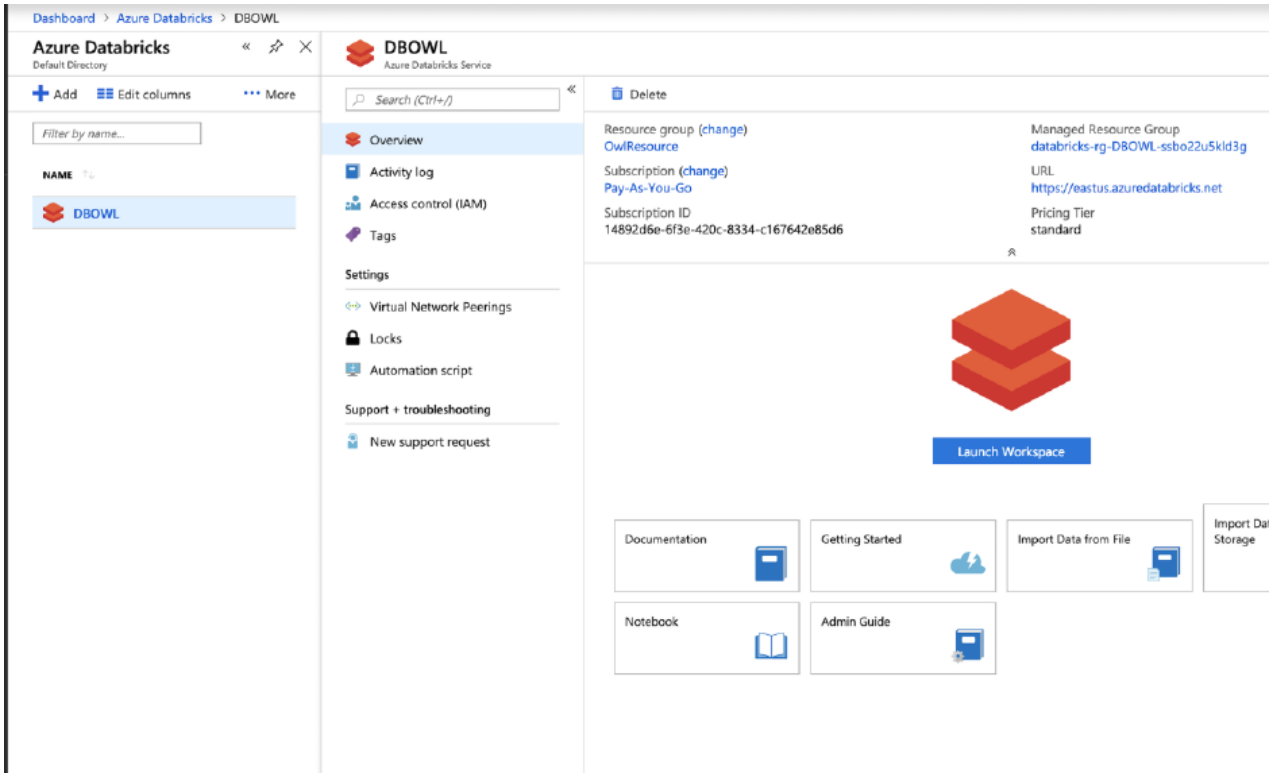
time to value for business insights as Data Scientists get time back, not cleaning / prepping the data, and helping the model maintain a longer life.

Azure Databricks allows the ability for Scala code to be written in a Jupyter Notebook against an Azure backed Databricks cluster to scale the work out to more nodes. This is to support the model and the amount of data being crunched by a business's Data Scientists. The simplistic nature of Azure and Databricks and the unification of Spark and Jupyter Notebooks, on top of a robust infrastructure (from storage to compute), allow for Owl-Analytics Data Qualified pipelines to be built and executed seamlessly. This reduces the time it usually takes to obtain valuable insights.

Here is how you can build such great DQ pipelines.

Step 1: Build a Databricks Cluster in Azure.

Within the Azure portal find Azure Databricks Service and create a cluster, after the cluster is built you should be able to launch the Workspace as shown below.

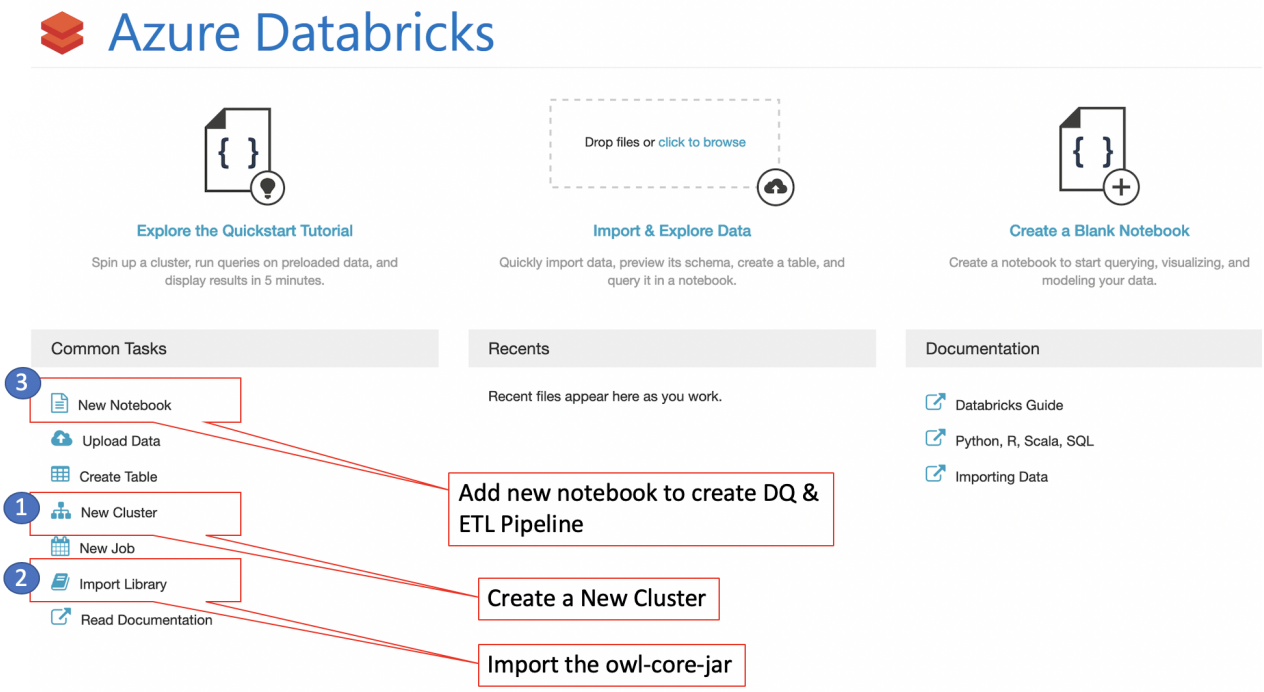


Step 2: Create a Cluster, add the Owl jar and create a Notebook.

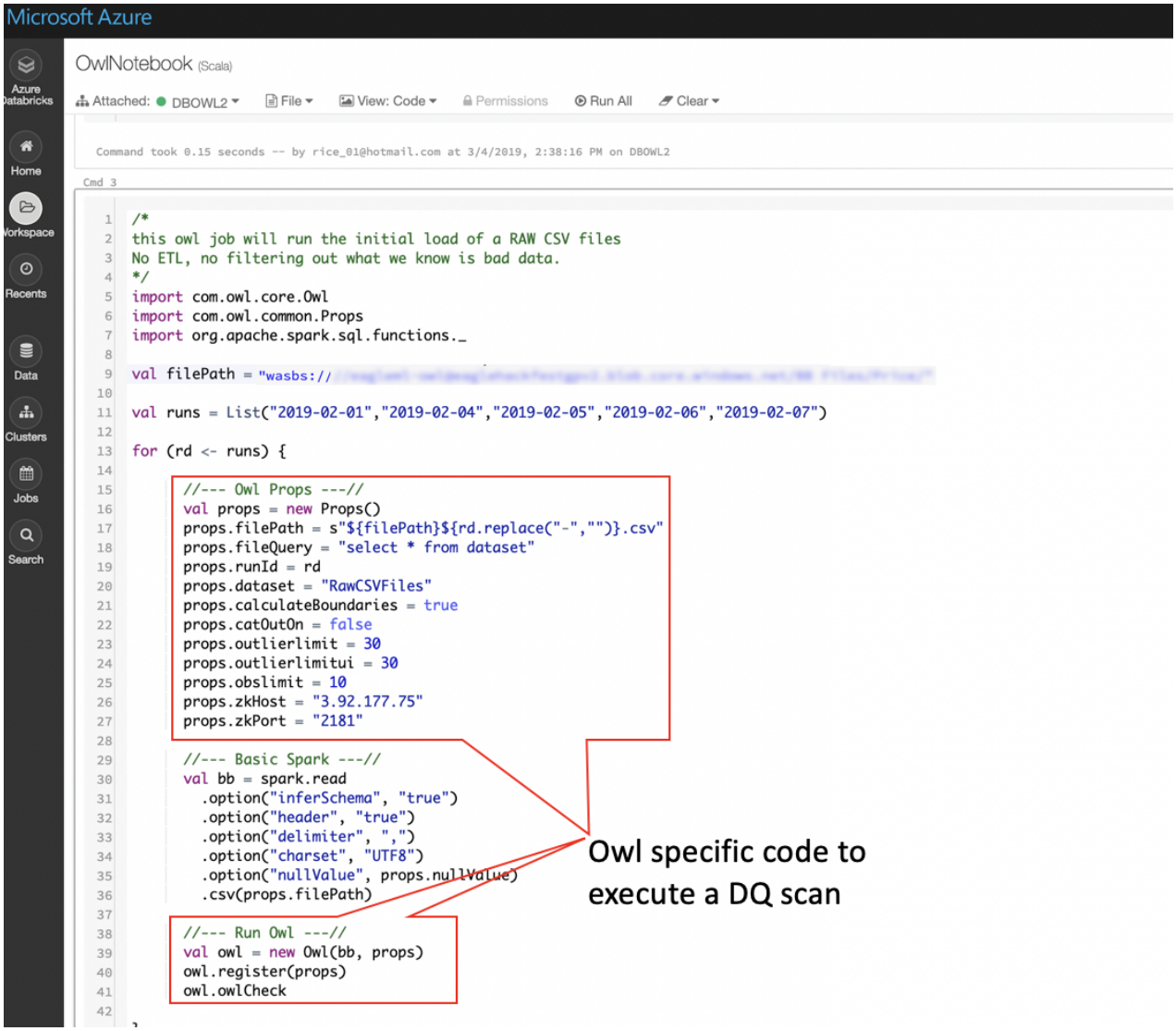
1.) Inside the Azure Databricks UI, create a cluster, provide a cluster name (in this example we will be using DBOWL2 as the cluster name) and select the Databricks Runtime Version that DQ currently supports (as of this blog post), which is Runtime: 5.2 (Scala 2.11, Spark 2.4.0).

2.) After the cluster is created, make sure to import DQ's jar file onto the cluster so that the Notebook can access the methods exposed in the jar file.

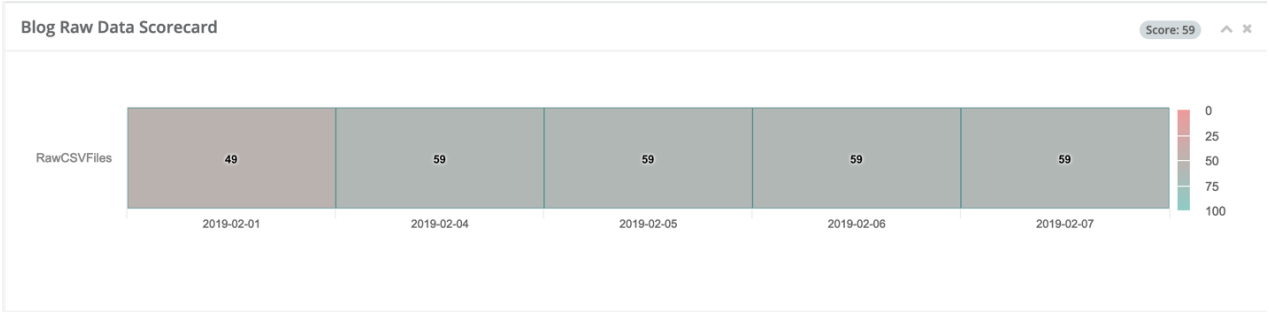
3.) Now that the Jar file has been added, create the Notebook and attach it to the cluster.



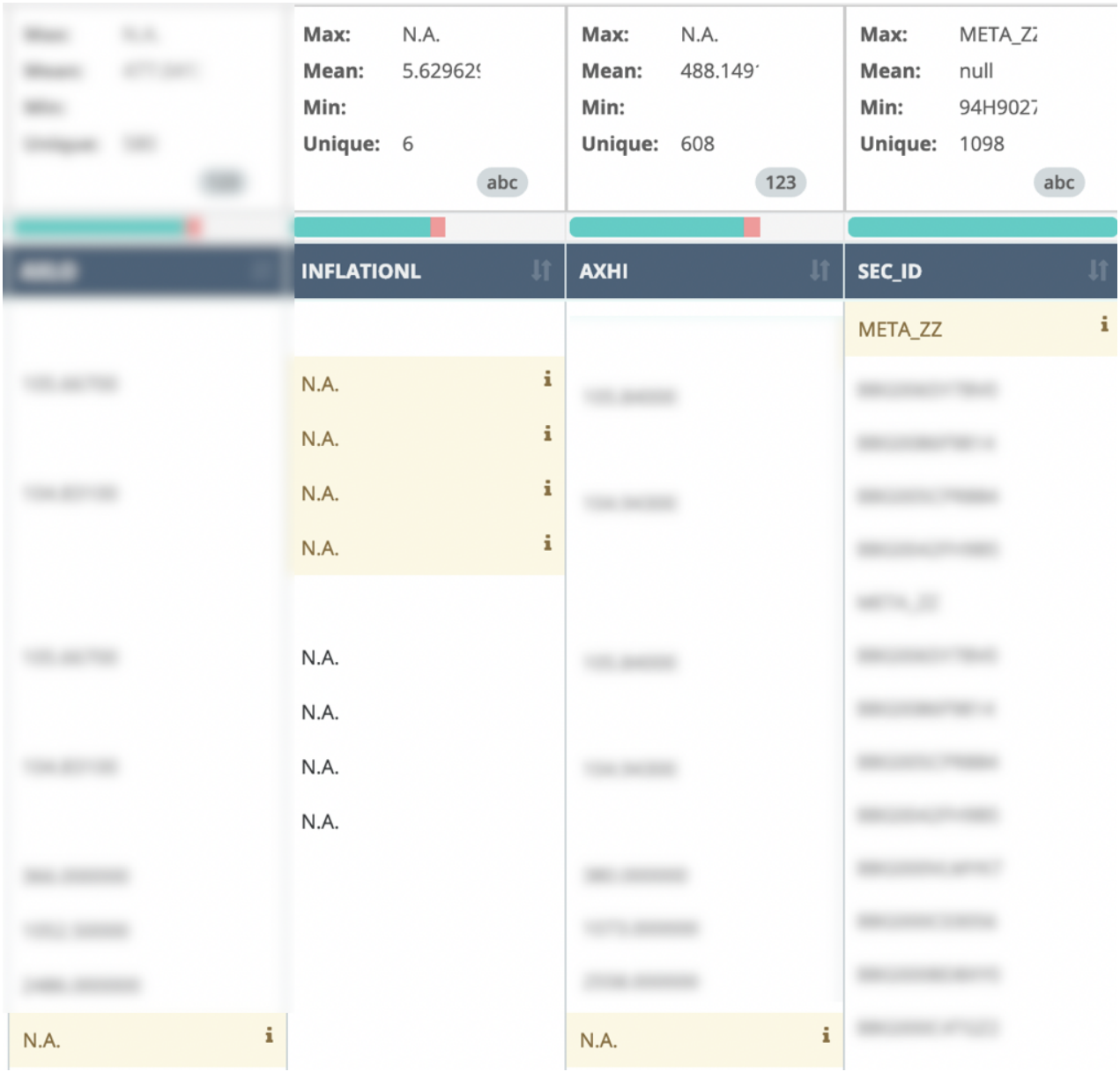
Now the cluster is running with the DQ jar loaded on the cluster. Open the Jupyter notebook attached to the cluster and begin looking at a data set as a Data Engineer would, prepping the data for use by a Data Scientist by leveraging a DQ Pipeline as shown in the below screen shot.



This Scala code imports the DQ jar and loops through the dates residing in files on Azure blob storage, pulls them into a Spark Data Frame (DF), and executes a DQ job to scan for the quality issues on the Spark DF. Once the scan is completed, the results are stored into the metadata repository under DQ's web application and visible through your browser, as shown in Figure 4 below.



The reason for a score of 49 on the raw data (as shown below in Figure 5) is due to the file having string values sprinkled in the file when something is Not Applicable (N.A.). When reading data in a column of a file that has a mix of numeric and string values the column will automatically conform to a string regardless if the majority class are integers. Also, within the files there is a single record in this file that has meta data information about the file “META_ZZ” this is also adding empty strings for all other columns. This record will also cause all columns to conform to strings.



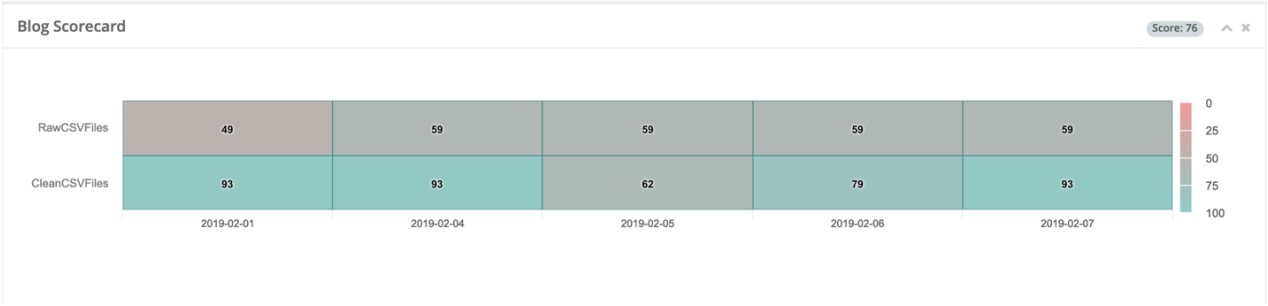
You should now have an understanding of the raw file and how you need to conform it before analysts can start to glean business value from the contents itself. First, ETL or

cleanse the data that you discovered as being in error by filtering out the erroneous record and flipping all the N.A. values to null as the next step in our ETL and DQ pipeline.

```
Command took 0.15 seconds -- by rice_01@hotmail.com at 3/4/2019, 2:38:16 PM on DBOWL2

Cmd 3

/*
this owl job will use native spark to filter out bad data replacing N.A. with nulls
and filtering out the a column with value META_ZZ...once ETL'd the data will get
stored in a cleaned directory and rerun with Owl
*/
import com.owl.core.Owl
import com.owl.common.Props
import org.apache.spark.sql.functions._

val filePath = "wasbs://

val runs = List("2019-02-01","2019-02-04","2019-02-05","2019-02-06","2019-02-07")

for (rd <- runs) {

    //--- Owl Props ---//
    val props = new Props()
    props.filePath = s"${filePath}${rd.replace("-","")}.csv"
    props.fileQuery = "select * from dataset"
    props.runId = rd
    props.dataset = "CleanCSVFiles"
    props.calculateBoundaries = true
    props.catOutOn = false
    props.nullValue = "N.A."

    //--- Basic Spark ---//
    val bb = spark.read
                  .option("inferSchema", "true")
                  .option("header", "true")
                  .option("delimiter", ",")
                  .option("charset", "UTF8")
                  .option("nullValue", props.nullValue)
                  .csv(props.filePath)

    //--- Spark File ETL ---//
    val bb1 = bb.filter(row => row.getAs("SEC_ID") != "META_ZZ")
                .withColumn("OWL_RUN_ID", lit( props.runId) )
                .withColumn("OWL_RUN_ID", to_date($"OWL_RUN_ID") )

    // file staging area
    props.filePath = s"${filePath}/cleaned/${props.runId}"
    bb1.write
       .mode("overwrite")
       .format("com.databricks.spark.csv")
       .option("header", "true")
       .save( props.filePath)

    val df = spark.read
                  .option("inferSchema", "true")
                  .option("header", "true")
                  .option("delimiter", ",")
                  .option("charset", "UTF8")
                  .option("nullValue", props.nullValue)
                  .csv( props.filePath)

    //--- Run Owl ---//
    val owl = new Owl(df, props)
    owl.register(props)
    owl.owlCheck
}
```

Spark ETL
after cleanse
write to staging then
run another Owl DQ

The DQ block of code is essentially the same, however, there is a new DQ property added to auto filter values “props.nullValue = ‘N.A’”. This finds every cell that has the value of N.A. and conforms it to a “null”. Once the file is read into a Spark DF, you use Spark to “Filter” out the erroneous record on line 36 in the code snippet above. Notice we are also adding an Owl_Run_ID date as this data set did not have a date that conforms easily. After the ETL process cleanses the data, you then have DQ’s Data Quality engine scan the newly processed Spark DF, storing the results into a data set called CleanCSVFiles (as shown in Figure 7 below).



Notice the composite scores in the boxes are substantially better for the CleanCSVFiles data set than what they are for the original RawCSVFiles. In the next article, we will look deeper at the intelligence a DQ scan garners on a data set when run over several days and how DQ surfaces different patterns, behaviors, trends and more in the data itself.

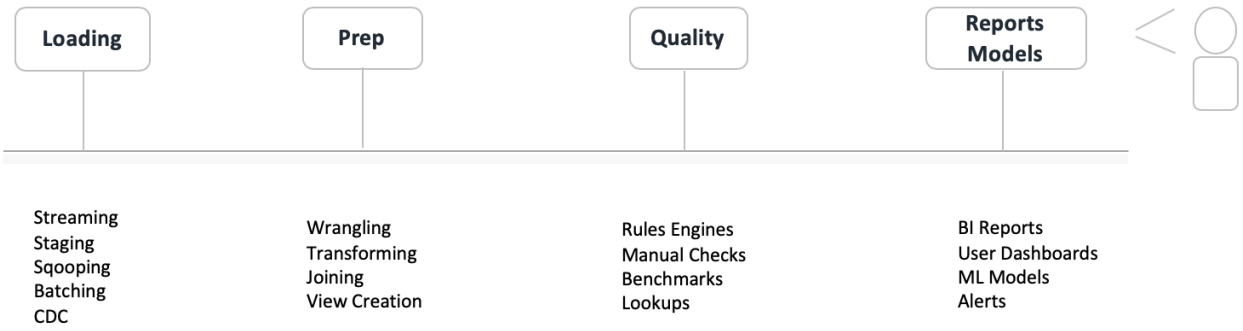
Our Approach

Because: Using raw data to drive key decisions, leads to incorrect answers and end-user distrust.

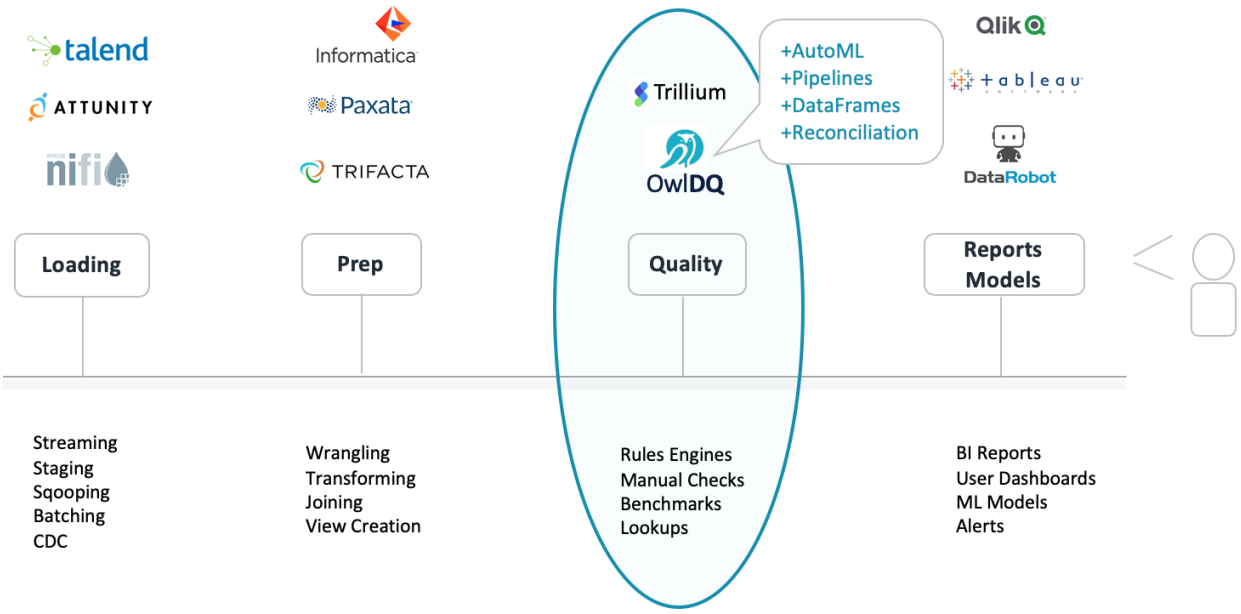
Collibra Data Quality is singularly focused on providing your end-users with the highest standards of data quality. We are purpose-built to solve the problem of data quality and to ensure end-user trust.

Whether you use a BI tool to visualize data or you are responsible for serving data to downstream subscribers, you always want to trust that your data is accurate. Showing inaccurate data in a bar chart or PDF report leads to a lack of confidence in the data provider. For example, see the data pipeline below. There are four main stages: Data

Loading, Data Preparation, Data Verification (DQ), and Data Reporting, which covers a broad category of all ways to see and deliver data.

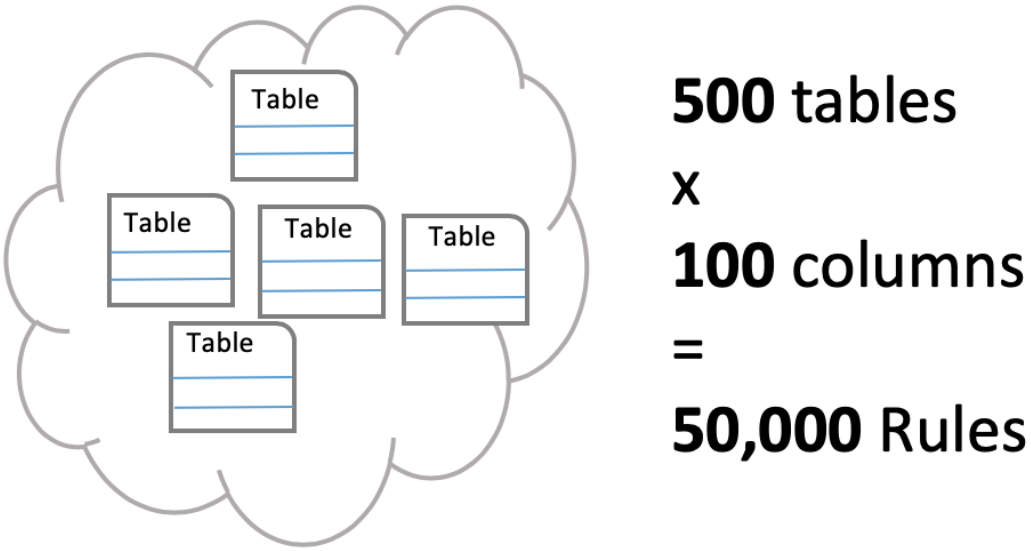


To avoid getting lost in the latest marketing jargon, a fundamental description is provided under each of the four stages. There are many ways to ingest and transform data; the descriptions are not meant to be exhaustive. Imagine a scenario where data is loaded in either a batch or stream, then joined to another dataset with some column transformations, and finally made viewable in a BI tool for consumption. But what about quality? What checks and verifications are in place to guarantee data accuracy and completeness? After all, showing someone a housing report with incorrect estimated housing values or a stock report with the wrong stock prices won't go over well. Figure 2 below shows popular company logos overlaid in each stage to bring more context to the discussion. There are easily 30+ software companies in each of the four stages, DQ chose three popular companies in each sector at random. DQ is not ranking companies. Gartner is of course an obvious choice if you are looking for companies rankings per sector.



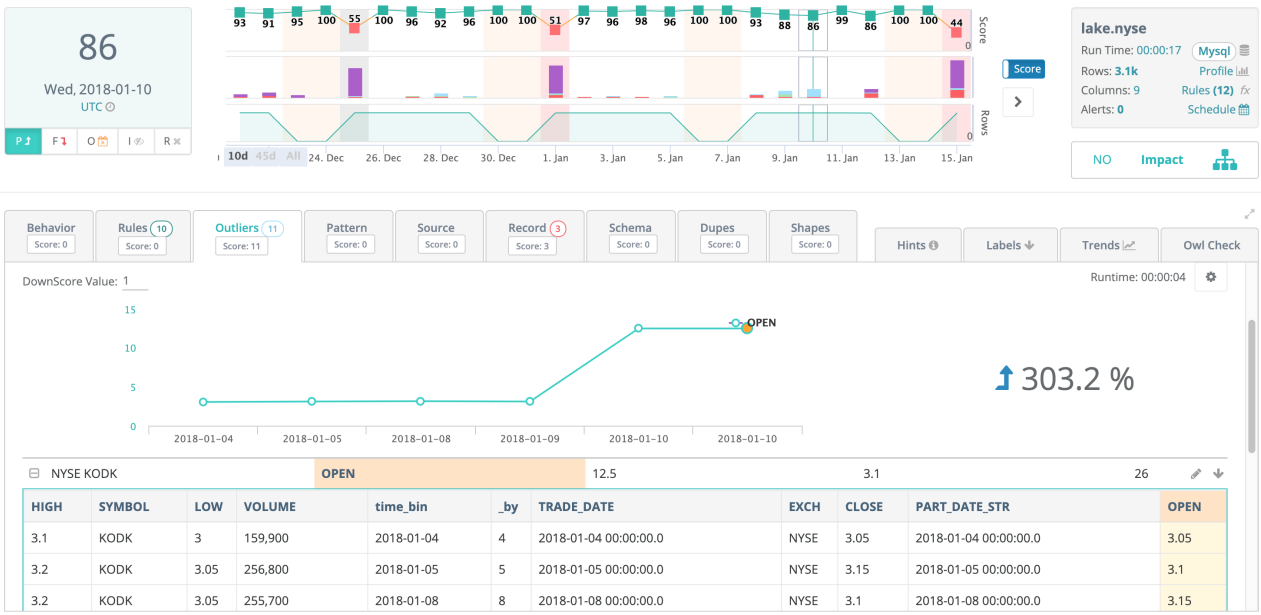
So, What's the Problem?

Detecting data issues is nuanced, manual and time consuming. The traditional solution is to write bespoke code or use a rules engine to validate specific columns in a data set. If missing data is a concern, a common remedy is to write a *nullcheck*. Another common example is a *row count check;* a piece of logic that checks if the number of rows in a data set is greater than a specified number. Of course, DQ and business rules can get much more complicated. Scale becomes a huge issue, because it is nearly impossible to write all the rules that a business truly needs to be confident in their data. Often times, the math is *f (x) = columns * dbTables*. 100 columns on average and 500 tables in a single warehouse equals 50,000 rules if you only wrote 1 rule per column. The reality is you need many rules per column, and your business has more than 500 tables and files. But there are even bigger problems with this strategy. Rules are a reactive approach to solving the problem; they are manually written and don't adapt (they are static). With a rules-only approach, you can measure your franchise risk by the number of rules you can write. This requires coders, domain experts and a tool to write and then manage the rules.



How Can Predictive DQ Help?

DQ intentionally solves the problem using a machine learning first, rules second based approach. DQ automatically puts all columns under quality control. This includes *nullchecks, emptychecks, statistical profiles, and sketches.* DQ creates snapshots and baselines to benchmark past data and discover *drift*. DQ automatically creates an ML labeling system for users to collaborate and down-train items with a click of a button. The reason for this approach is to maximize coverage while reducing the dependency of manual rule building. The greater technical benefit is that all of DQ's generated checks and rules are adaptive. DQ is constantly learning from new data and will make predictions in many cases for typos, formatting issues, outliers and relationships. This is a paradigm shift **from**, *risk being a measure of how many rules one can dream up and write*, **to** *simply click the DQ [RUN] button*.



HIGH	SYMBOL	LOW	VOLUME	time_bin	_by	TRADE_DATE	EXCH	CLOSE	PART_DATE_STR	OPEN
3.1	KODK	3	159,900	2018-01-04	4	2018-01-04 00:00:00.0	NYSE	3.05	2018-01-04 00:00:00.0	3.05
3.2	KODK	3.05	256,800	2018-01-05	5	2018-01-05 00:00:00.0	NYSE	3.15	2018-01-05 00:00:00.0	3.1
3.2	KODK	3.05	255,700	2018-01-08	8	2018-01-08 00:00:00.0	NYSE	3.1	2018-01-08 00:00:00.0	3.15

Why a Unified DQ Solution?

Aren't their other DQ companies and solutions on the market? Yes, absolutely. The challenge lies in the vast number of ways IT groups consume and process data. You need to find a product that can plug into Files the same way it plugs into DB Tables, Cloud File Systems, Data Frames and Kafka Topics, etc. You need a product that offers a consistent feature set and covers all nine dimensions of DQ. For most companies, DQ is an after thought, they will add-on a single dimension of DQ, such as *rules* or *data drift*.DQ offers a full data quality suite to cover the unique challenges of each data set. Complete coverage and consistency drives trust. A single scoring and reporting framework with nine pluggable features that can be activated in a tailorable DQ pipeline. DQ is horizontally scaleable, it can scan data from any location with infinity scale. Data quality needs to be abstracted from data ingestion for management to have a single normalized view of data health.

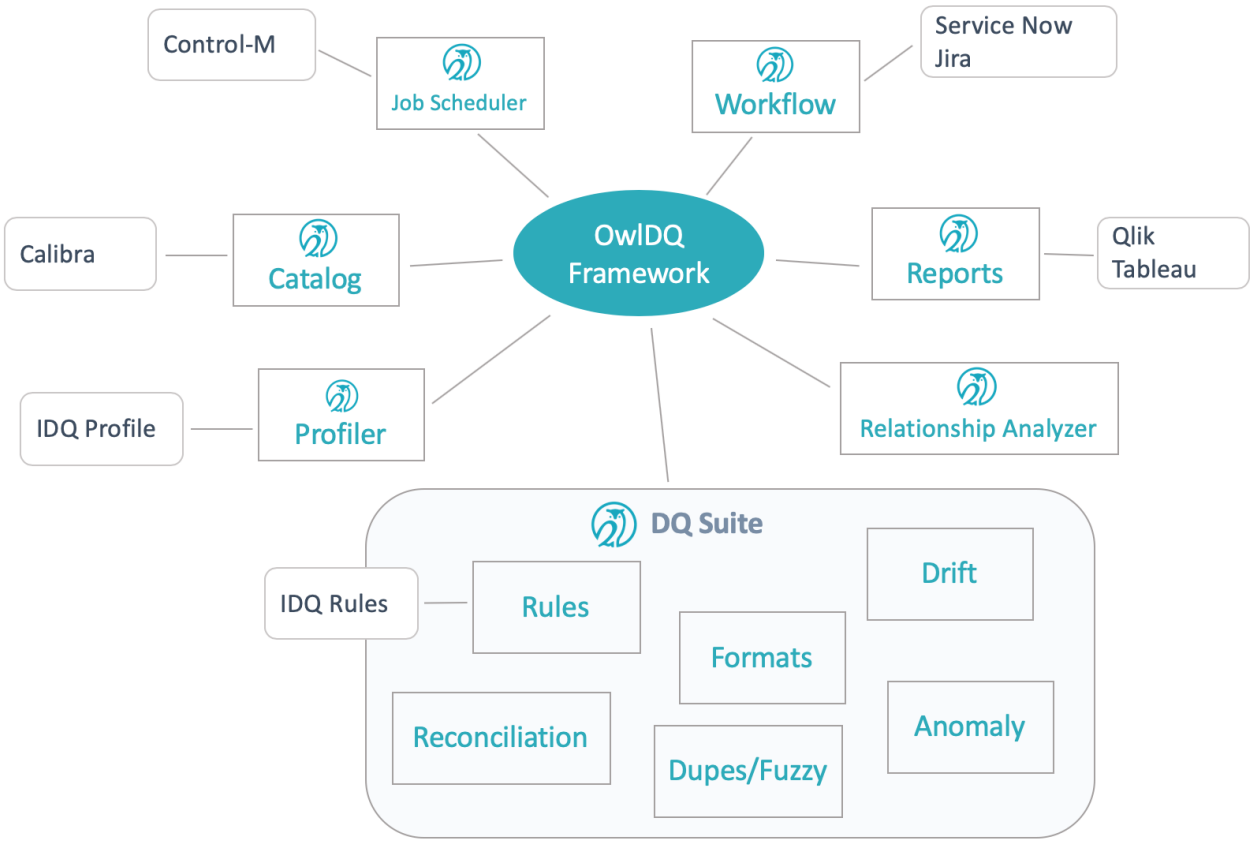
Do One Thing Extremely Well

DQ believes that data quality is such an important part of the data lifecycle that it requires a company that is solely committed to revolutionizing the way enterprises manage data quality. This is why DQ has a prescriptive approach to data quality (ML first, Rules second). The DQ software is purpose built for predicting and detecting data quality issues. Much like how Jira is used as the standard for software project management, even though it is absolutely possible to manage project line items in an excel sheet. Businesses that manage a lot of data require Score Cards, Alerts, Reports, List Views, Collaboration, Down Training, Cataloging, Scheduling and much more.

Get Started

Email us: info@collibra.com

Does your DQ Solution Have?



Unified DQ	The ability to score and manage and report on all datasets (files, tables, topics) agnostically. Providing a single pane of glass for DQ across all data sources.
Collaboration	The ability for end-users to down-train, annotate and audit each DQ item
Auto Discovery	The ability to figure out issues in your data without requiring domain experts and rule writers
Anomaly Detection	The ability to detect numeric and categorical outliers in a dataset
Correlation Analysis	The ability to measure the lift or relationship between numeric columns

Relationship Analysis	The ability to discover and make predictions on hidden relationships in your data
Alerting	The ability to send out alerts when DQ scores drop
Scheduling	The ability to schedule DQ jobs with a click of a button in the UI
Profiling	The ability to provide descriptive statistics about the current run overlaid with the past runs for trend analysis
Reconciliation	The ability to validate the source and target dataset in timeline snapshots
Duplicate Detection	The ability to find exact and similar matches in data records
Lineage Graphs	The ability to asses business impact via a business impact score by understanding how connected each dataset is
Schema Evolution	The ability to detect changes in data types, additions and removals
Rules	The ability to write custom rules for simple and complex scenarios

Our Story

Background

The Collibra DQ team comes from a variety of backgrounds. While some spent a decade building technology to detect financial crimes, others were architecting data fabrics at fortune 100 companies.

> *Regardless of the industry or experience, we all faced similar challenges as it related to data quality.*

These unique vantage points have allowed us to understand the most common data quality challenges organizations are facing.

What Did We Notice?

We tried many of the traditional tools and techniques. The biggest problem was always the amount of time it took to do everything needed to implement and maintain data quality controls.

You get left with half-baked data quality coverage and the right controls are added only after issues occur.

It turned out teams were doing the same tasks for every data set and for each department, building the exact same tools over and over again.

> Note The result was a never-ending cycle of data issues, fire drills, and a mad scramble to fix it fast. All within the context of real-time business operations.

Traditional Approach

Traditional approaches are very manual.

Start by opening a sample or spreadsheet and conduct analysis (table-by-table, column-by-column, query-by-query, and item-by-item).

Next, manually craft the rules, conditions, tolerances, and thresholds.

Then stitch together the dashboards, scoring, workflows, alerts, and reporting. And you wonder why bare-minimum coverage is common.

> Note You're only as good as the rules you thought to write.

Fast Forward

Now that the surface area of the data in an organization is so much larger, these traditional techniques don't hold up.

What Did We Need?

What we needed didn't exist. As lifelong data enthusiasts, we wanted a tool that could alert us when data changes occurred without complicated setup and lengthy analysis. We sought something that could work on big, medium, and small data and across all storage formats. Upon evaluating all the commercially available tools, and assessing costs and time of homegrown solutions, there were no great options.

DQ is the difference

Lake vs Swamp

The difference between a business-critical lake and a swamp is **data _quality_.** One organization’s data lake may be another's data swamp. The difference lies in how data is curated. A data lake describes a vast amount of data that can be stored, assessed, and analyzed. A data swamp has little data governance, DQ automation, or contextual metadata.

The accuracy and cleanliness of data is directly proportional to the quality of insights end-users will derive. Data lakes that gain broad adoption have strong governance programs. The challenge is, adding a DQ program typically takes 6-12 months but the project never really ends due to the volume, variety and velocity of incoming data. OwlDQ uses autoML so solve this problem. OwlDQ constantly monitors the lake with native integration and unlimited scale. Use OwlDQ to generate the equivalent of 10K rules, while continuously adapting to the natural variance in your data. When erroneous data enters your lake OwlDQ will alert the data steward and provide a rich visual displaying the break records and explainable AI describing the issue. OwlDQ's approach is to learn from data and become incrementally smarter each day to ensure a statistically defensible DQ program.

What is CDQ

CDQ is an intelligent data validation tool.

8 Ways to Add Value Using CDQ

1. **Crowdsourcing**

   “People that have never written SQL are now helping with data quality”

2. **Rule Coverage**

   “Did in 20 days what took 2 years with our legacy tool”

3. **Audit & Identify Gaps**

   “Audited our existing checks and could not imagine the gaps we uncovered.”

4. **Automate Repeatable Processes**

   “DQ cut 60% of our manual workloads”

5. **Technology Limitations**

   “We now scan files and Kafka, avoiding downstream issues”

6. **Getting standard**

   “No more piecemeal reports. Files, Warehouse, Lake. All metrics in one, transparent place.”

7. **Building Reports, Visuals, Workflows**

   “This takes the place of 3 tools”

8. **Prioritized Efforts**

   “Easy to see top priorities for improvement”

What Savings Does CDQ Provide?

Save Hours of Effort with Auto-generated Data Validation Checks

- **Top 10 Bank**
  Reduced 60% of their manual Data quality workload + $1.7M cost savings
- **Top 3 Healthcare Organization**
  Saved 2,000 hours during a cloud migration requirement
- **Top Insurance Organization**
  Satisfied Regulatory Second Line Controls in a 4 weeks (what originally took 2 years using their previous tool)

While Reducing System-Wide Pain Points

- Overwhelmed with tickets
- Business users find issues first
- Touchy pipelines break with minor updates
- Too busy responding to fire drills to implement new projects {% endhint %}

How Can CDQ Help?

Click a button and smile - knowing baseline validation checks are applied - instead of spending hours manually digging through data & stitching together scripts

- ☑ **Implementing Checks**
  - Autodiscovery
  - Generates SQL validations, parameters & thresholds
  - Rule suggestions

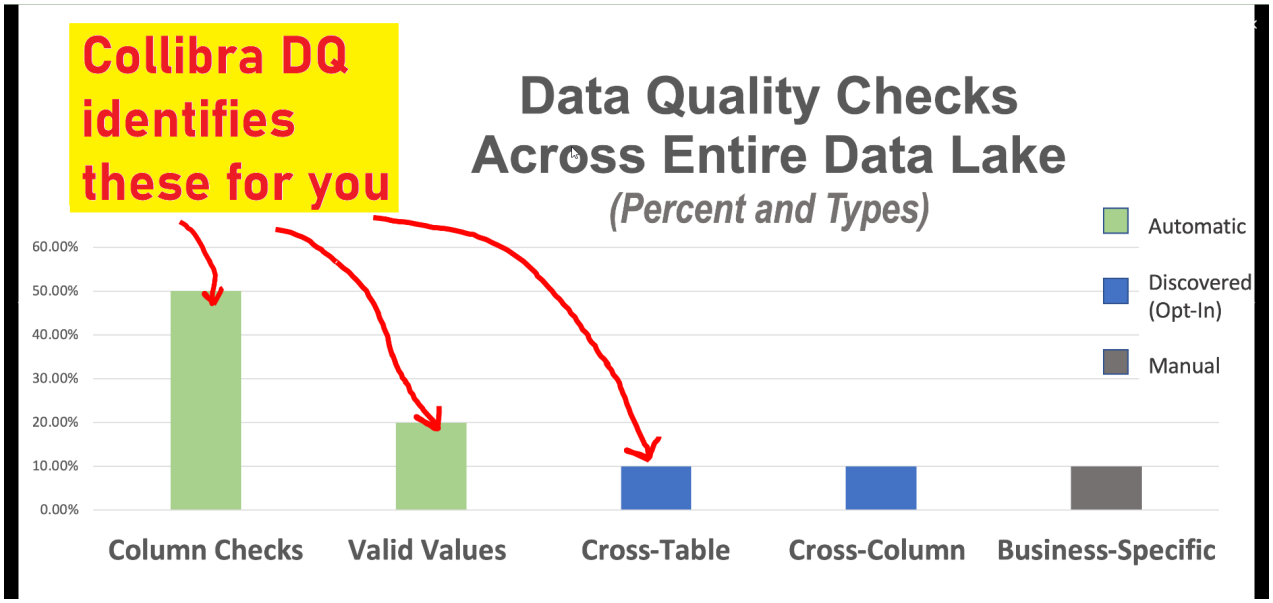
- ☑ **Taking Inventory**
  - Bulk Profiling & Metadata Collection
  - Data Mapping with Column Identification
  - Map Column Fingerprints, Cross-Table Matches & PII Checks
- ☑ **Consolidating Systems**
  - No more closed-systems or confusing scripts
  - Macro & micro views for measuring effectiveness over time
  - Global management Across Sources / Platforms / Environments
- ☑ **Enabling More Users**
  - Self-Service, Easy to use Rule Editor
  - Pre-Built Analytics and Charts
  - Extensible APIs, Open Architecture

Boost productivity. 80% faster than manual coding. Minimize development costs. Get faster, easier access to data quality metrics. Show line of business users how to self-service.

What **makes CDQ unique**?

CDQ is The Only Tool Business & Technical Users Will Love

Every feature, visual, and component within Collibra DQ is intended to make the analysis and implementation of data checks easier.



Why?

Because Humans Can’t Predict Every Which Way Data Can Go Wrong.

{% tabs %} {% tab title="Billing Issue Example" %}

{% endtab %}

{% tab title="Financial Data Example" %}

{% endtab %}

{% tab title="API Example" %}

{% endtab %}

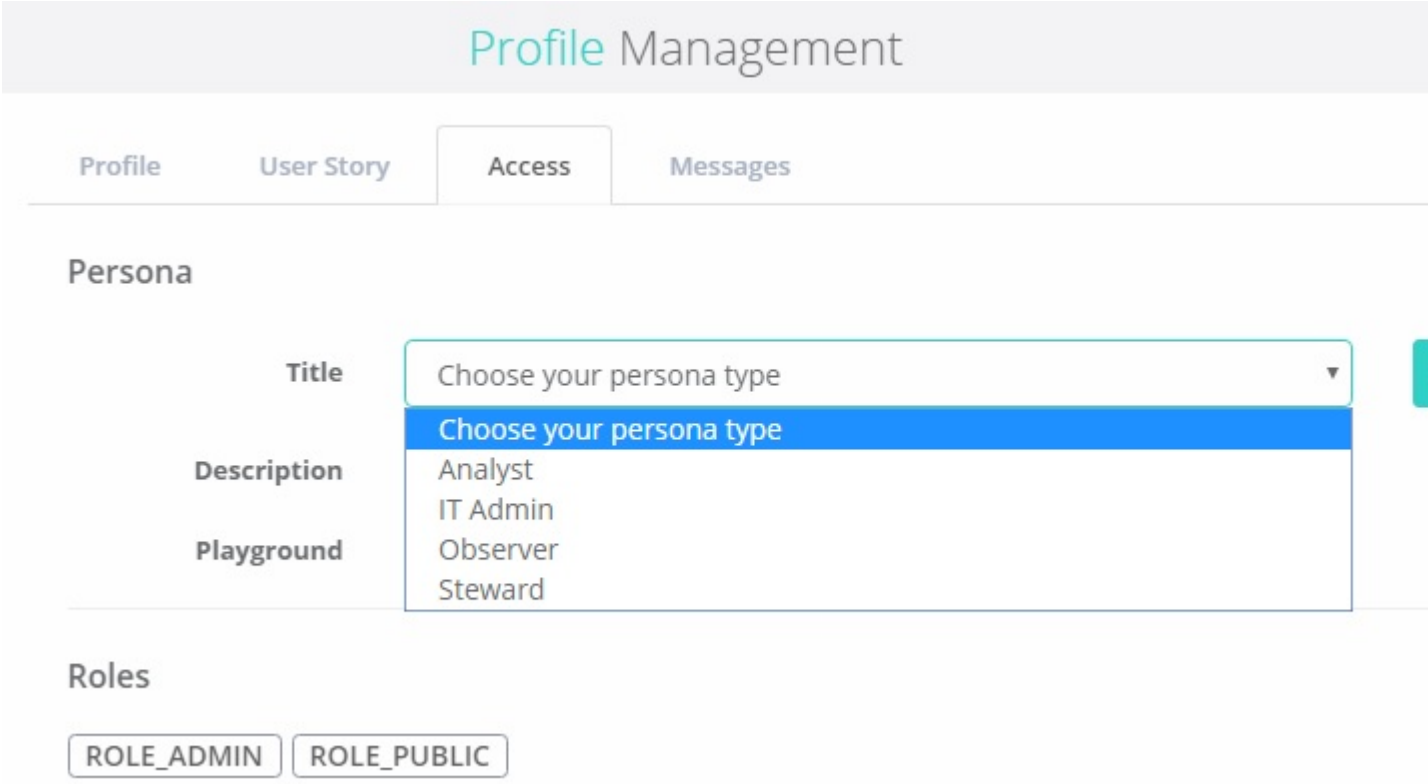
{% tab title="IoT / Meter Example" %}

{% endtab %} {% endtabs %}

****

Prescriptive Personas

Collibra DQ has four prescriptive personas to manage user permissions: Analyst, IT Admin, Observer, and Steward. Click the user icon located on the bottom left of the blue DQ Menu bar and select **User Profile**. The persona type can be assigned under the access tab in Profile Management.



Bank Loans

It is common for banks to lend money in return for monthly payments with interest. However, to do so a bank must make sure that the applications are valid and well formed to begin the underwriting and approval process. The following list comprises some basic lending concepts to Collibra DQ.

1. Credit Score Validation
2. SSN Validation
3. Loan to Value Validation
4. Interest Rate Validation
5. Duplicate Loan Applications
6. Loan Amount Validation
7. Loan Completeness Validation

member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	credit_rating	rate	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status	issue_d	loan_status
1	2500	2500	2500	36 months	13.56	84.92	FAIR	6.5	C	C1	Chef	10+ years	RENT	55000	Not Verified	18-Dec	Current
1	2500	2500	2500	36 months	13.56	84.92	FAIR	6.5	C	C1	Chef	10+ years	RENT	55000	Not Verified	18-Dec	Current
2	30000	30000	30000	60 months	0	777.23	POOR	8.5	D	D2	Unemployed	10+ years	MORTGAGE	90000	Source Verified	18-Dec	Current
2	30000	30000	30000	60 months	0	777.23	POOR	8.5	D	D2	Unemployed	10+ years	MORTGAGE	90000	Source Verified	18-Dec	Current
2	30000	30000	30000	60 months	18.94	777.23	POOR	8.5	D	D2	Postmaster	10+ years	MORTGAGE	90000	Source Verified	18-Dec	Current
2	30000	30000	30000	60 months	18.94	777.23	POOR	8.5	D	D2	Postmaster	10+ years	MORTGAGE	90000	Source Verified	18-Dec	Current
3	5000	5000	5000	36 months	0	180.69	POOR	8.5	D	D1	Administrative	6 years	MORTGAGE	59280	Source Verified	18-Dec	Current
3	5000	5000	5000	36 months	17.97	180.69	POOR	8.5	D	D1	Administrative	6 years	MORTGAGE	59280	Source Verified	18-Dec	Current
3	5000	5000	5000	36 months	17.97	180.69	POOR	8.5	D	D1	Administrative	6 years	MORTGAGE	59280	Source Verified	18-Dec	Current
4	4000	4000	4000	36 months	0	146.51	POOR	8.5	D	D2	IT Supervisor	10+ years	MORTGAGE	92000	Source Verified	18-Dec	Current

1. Credit Score

Business Check	OwlDQ Feature	Manual vs Auto
Is the credit score a whole number?	BEHAVIOR	AUTO

Business Check	OwlDQ Feature	Manual vs Auto
Is the credit score within a valid range? (between 300 - 850)	RULE	credit_ score between 300 and 850
Is the credit score NULL or Missing?	BEHAVIOR	AUTO

2. SSN Validation

Business Check	Collibra DQ Feature	Text
Is the SSN format valid?	RULE	AUTO-SSN detection
SSN is PII.	SENSITIVITY	AUTO-SSN labeled
Is the SSN NULL or Missing?	BEHAVIOR	AUTO
Does the SSN belong to the Applicant?	PATTERN	SSN -> first_name, last_name

3. Loan to Value

Business Check	Collibra DQ Feature	Text
Are loan amount and asset value (home or auto) valid numbers?	BEHAVIOR	AUTO
95% loan to value ratio to approve?	RULE	loan / asset_value < .95

4. Interest Rate

Business Check	OwlDQ Feature	Text
Interest rate between min and max allowable range for the loans credit rating.	RULE COMPLEX	loan l join rates r on l.credit_rating = r.credit_rating where l.rate between r.min_rate and r.max_rate

5. Duplicate Loan Applications

Business Check	OwlDQ Feature	Manual vs Auto
Ensure we don't issue the same loan twice.	DUPE	first_n, last_n, SSN, Address

6. Loan Amount

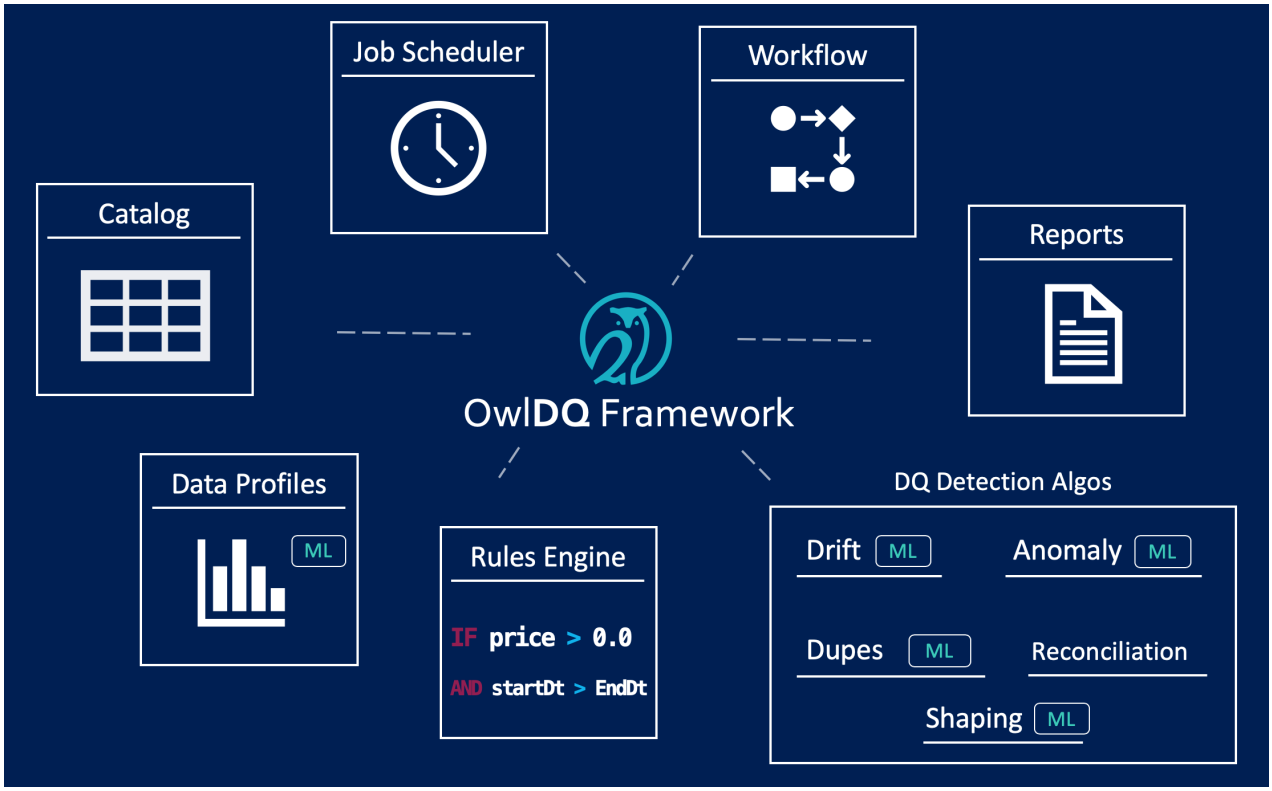
Business Check	OwlDQ Feature	Manual vs Auto
Loan amount within lendable range	OUTLIER	AUTO
Loan amount within lendable range. Only lend money between 50K and 3M.	RULE	loan_amount between 50000 and 3000000

Resulting OwlCheck

```
-lib "/home/opt/owl/drivers/postgres" \
-cxn postgres-gcp \
-q "select * from public.loan_risk_grade where last_pymnt_d =
'2019-04-01'" \
-key member_id -alias loan_risk \
-ds public.loan \
-rd "2019-04-01" \
-dl -loglevel INFO \
-h 10.142.0.29:5432/owltrunk \
-numexecutors 10 -executormemory 1g -drivermemory 4g \
-master yarn -deploymode cluster \
-sparkprinc user2@CW.COM \
-sparkkeytab /tmp/user2.keytab -tbin MONTH \
-dupe -dupeinc purpose -fpgon -fpgkey grade \
-fpginc grade,sub_grade -fpglb 365 -fpgdc last_pymnt_d \
-record member_id -dupecutoff 60 -dupepermatchupperlimit 99
```

Which components did we use?

We made use of Profiles, Duplicates, Outliers and Rules in this example. The experiments were automatically cataloged and put on a job scheduler. The next time a loan issue arises, we will be able to take remediation action using the workflow Q. Over time we can see how the bank loan program is running via the report section.



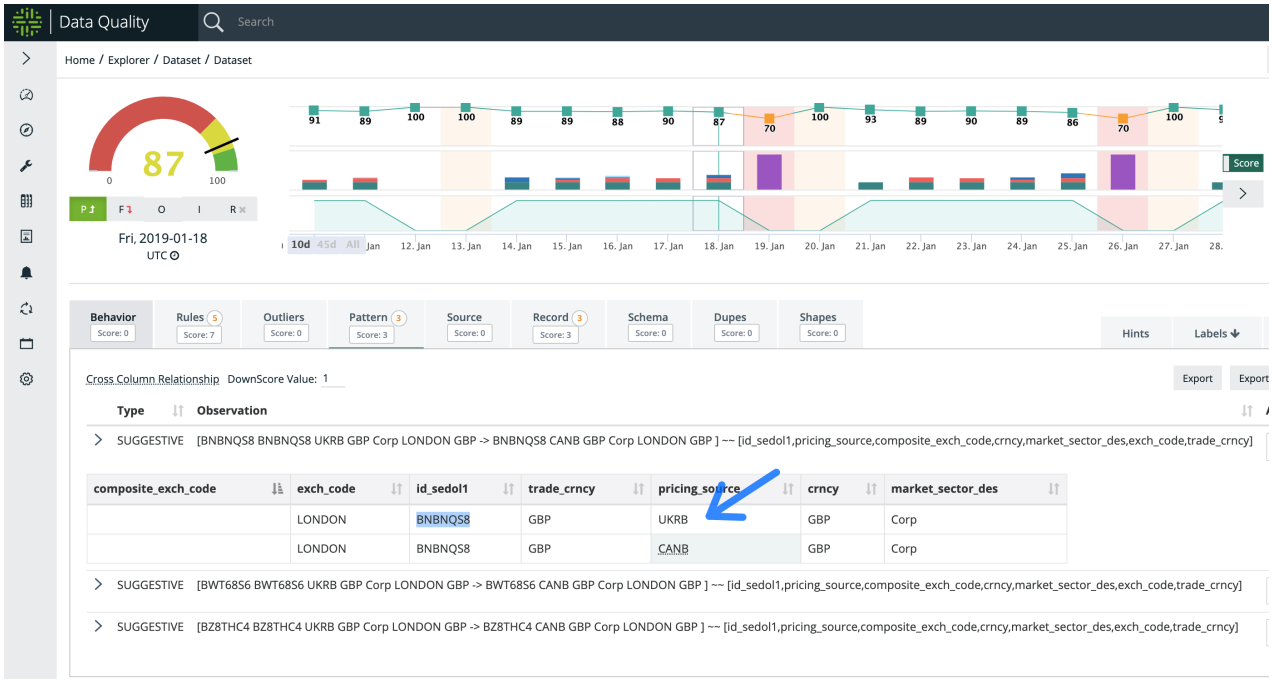
Files that can be used to replicate this example

interest_rates.csv
Binary

Owl Dataset (2).csv
Binary

Bloomberg Data

Collibra DQ finds over 50 data quality issues per day in common market data through pattern mining. Sequential pattern mining is a tool that finds statistically relevant patterns between data examples where the values are delivered in a sequence.



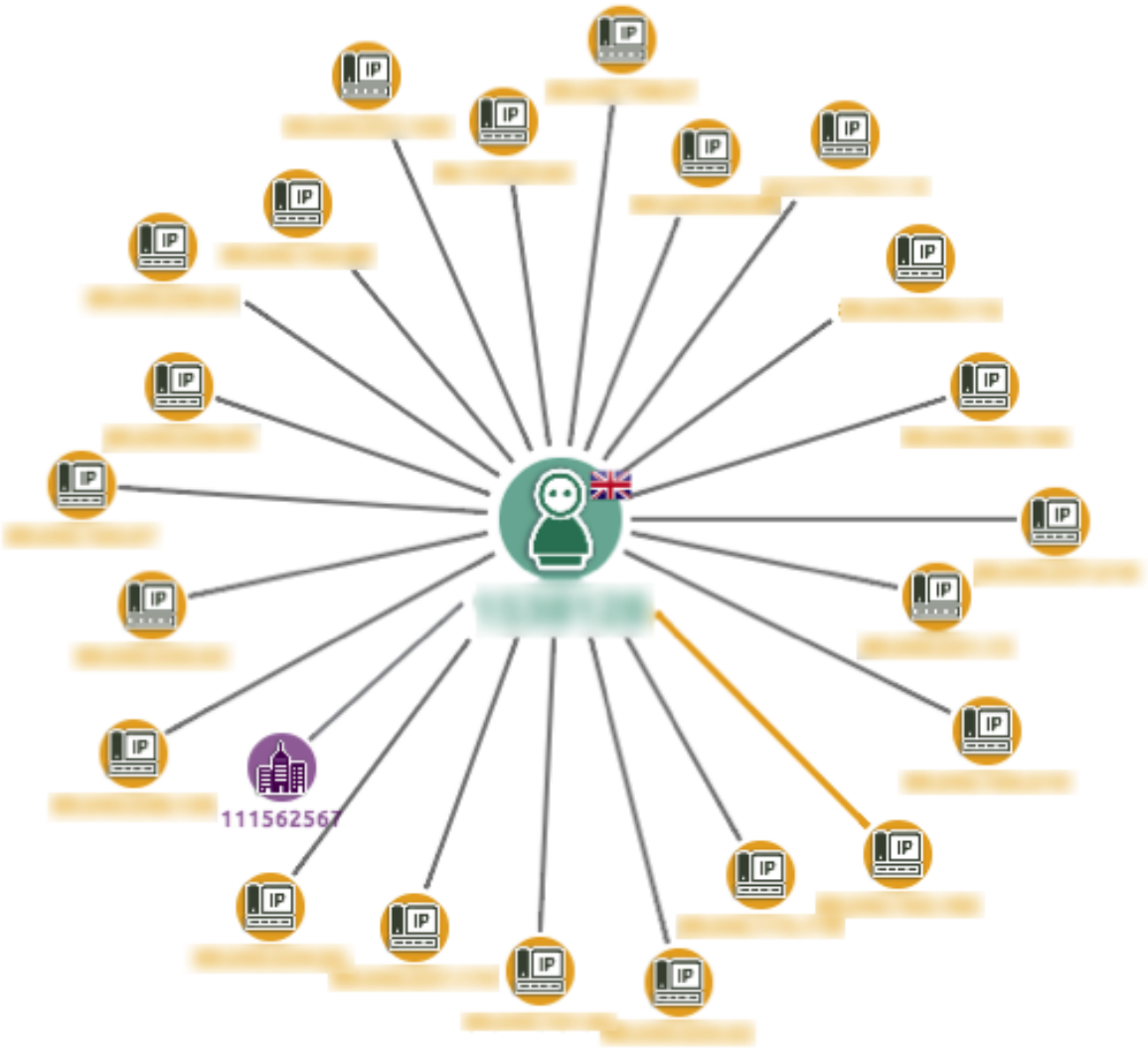
Cyber Anomalies in Real-Time

With an increasing number of cyber threats, most of the cyber security team doesn't have the capacity to manually detect, monitor, and defend against all of them. Effective cyber threat management requires leveraging automation to inform decisions.

The Collibra DQ framework, provides organizations the ability to load and process diverse security data feeds at scale to detect network data anomalies. The DQ alerts enable network admins to respond to these events in timely manner.

The following scenario demonstrates how to detect anomalies with network traffic data sets.

1. Perform IP address validation.
2. Detect the unusual network traffic patterns based on locations.
3. Identify the suspicious packets based on size.
4. Detect the malicious activity based on source and destination IP addresses.



Infosec data set preview

Data set contains Timestamp, Source Workgroup, Source IP, Source Port, Destination Workgroup, Destination IP, Destination Port, Application, Protocol and Packet size information.

Timestamp	Source Workgroup	Source IP	Source Port	Destination Workgroup	Destination IP	Destination Port	Application	Protocol	Packet Size(B)
2010-12-01 00:00:00	Atlanta	10.0.0.4	53323	SFO	10.0.0.18	443	HTTPS	TCP	122849
2010-12-01 00:00:00	Atlanta	10.0.0.4	52419	SFO	10.0.0.18	22	FTP	TCP	81055
2010-12-01 00:00:00	Atlanta	10.0.0.1	51618	SFO	10.0.0.17	22	FTP	TCP	98392
2010-12-01 00:00:00	Atlanta	10.0.0.5	58518	SFO	10.0.0.18	990	FTPS	TCP	245533
2010-12-01 00:00:00	Atlanta	10.0.0.1	58329	Manhattan	10.0.0.13	989	FTPS	TCP	250059
2010-12-01 00:00:00	Atlanta	10.0.0.5	49765	SFO	10.0.0.21	22	FTP	UDP	29012
2010-12-01 00:00:00	Atlanta	10.0.0.2	63277	SFO	10.0.0.17	22	FTP	UDP	27761
2010-12-01 00:00:01	Atlanta	10.0.0.4	63870	SFO	10.0.0.19	990	FTPS	TCP	189328
2010-12-01 00:00:01	Atlanta	10.0.0.4	63340	Manhattan	10.0.0.12	990	FTPS	TCP	180320
2010-12-01 00:00:01	Atlanta	10.0.0.4	60839	SFO	10.0.0.18	989	FTPS	TCP	230823
2010-12-01 00:00:02	Atlanta	10.0.0.1	59764	SFO	10.0.0.17	443	HTTPS	TCP	169610
2010-12-01 00:00:03	Atlanta	10.0.0.4	58744	Manhattan	10.0.0.9	443	HTTPS	TCP	141319
2010-12-01 00:00:04	Atlanta	10.0.0.4	58895	Manhattan	10.0.0.14	22	FTP	TCP	21691
2010-12-01 00:00:04	Atlanta	10.0.0.2	49241	SFO	10.0.0.21	22	SSH	TCP	17662
2010-12-01 00:00:04	Atlanta	10.0.0.4	65170	Manhattan	10.0.0.14	443	HTTPS	TCP	115949
2010-12-01 00:00:04	Atlanta	10.0.0.1	56006	Manhattan	10.0.0.12	990	FTPS	TCP	142744
2010-12-01 00:00:06	Atlanta	10.0.0.5	50528	SFO	10.0.0.21	990	FTPS	TCP	132937
2010-12-01 00:00:06	Atlanta	10.0.0.2	52919	Manhattan	10.0.0.10	22	SSH	TCP	14111
2010-12-01 00:00:06	Atlanta	10.0.0.4	58538	SFO	10.0.0.17	80	HTTP	TCP	121021
2010-12-01 00:00:06	Atlanta	10.0.0.6	56345	SFO	10.0.0.17	989	FTPS	TCP	126612
2010-12-01 00:00:06	Atlanta	10.0.0.6	59640	Manhattan	10.0.0.13	443	HTTPS	TCP	165096
2010-12-01 00:00:06	Atlanta	10.0.0.2	54127	SFO	10.0.0.21	22	SSH	TCP	19616

IP Address format validation

Business Check	Collibra DQ Feature	Text
Is IP a valid format?	RULE	AUTO-IP detection
Is the IP address NULL or Missing?	BEHAVIOR	AUTO

Source and destination workgroups

Business Check	OwlDQ Feature	Text
Is it usual network traffic based on locations?	PATTERN	Source_Workgroup -> Destination_Workgroup

Source and Destination IP Address validation

Business Check	Collibra DQ Feature	Text
Is it usual network traffic based on source and destination IP?	PATTERN	Source_IP -> Destination_IP

Packet Size

Business Check	Collibra DQ Feature	Text
Is the Packet Size NULL or Missing?	BEHAVIOR	AUTO
Packet Size within normal range?	PATTERN	Source_IP -> Packet_SizeB

Resulting OwlCheck

```
-f file:///home/danielrice/owl/bin/demos/infosec/ -d tab \
-fullfile -fq "select * from dataset" -encoding UTF-8 -ds
infosecv2 \
-rd "2020-04-04" -dl -dlinc Destination_IP,Packet_SizeB,Source_
IP \
-dlkey Source_IP -fpgon -fpginc Destination_Workgroup -fpgkey
Source_Workgroup \
-df "yyyy-MM-dd" -loglevel INFO -h 10.142.0.29:5432/owltrunk -
owluser admin \
-fpgsupport .000000001 -fpgconfidence 0.4
```

Which components did we use?

DQ addresses the issue of efficient network traffic classification by performing unsupervised anomaly detection and uses this information to create dynamic rules that classify huge amounts of Infosec data in real time.

By providing Infosec data sets, along with anomaly records DQ outlier and pattern algorithms found the anomaly in the network traffic. It mainly detects the following anomalies:

1. Traffic between Atlanta->Texas.
2. The packet size extremely low between Atlanta->Texas.
3. Atlanta source IP and Texas Destination IP.

Realtime DQ provides the alerts on network traffic anomalies, which can help network admins to do further deep analysis and takes preventative measure, which is a daunting task with huge amount of data.

Sample Data set

infosec-anomaly.csv
Binary

Financial FxRate Data

Collibra DQ Automatically Alerts to Incorrect Foreign Exchange (FX) Rate Data without a Single Rule

FX Rate data commonly looks like the below table. Often you have a TO currency and a FROM currency with the RATE being the multiplier column for conversion. For example, in March you would need to spend $1 US Dollar and 18 US cents to receive $1 Euro. "In exchange for":

TO_CURR	FROM_CURR	RATE	DATE
USD	EUR	0.82	2019-03-12
EUR	USD	1.18	2019-03-12
USD	YEN	111.0	2019-03-12

28,000 Currency Pairs

There are roughly 28,000 currency pairs and the exchange rates change throughout the day but at a minimum most banks are concerned with the daily close of the FX Rate. Now imagine trying to write a rule for each currency pair. You'd have to know the relationship and adjust a static threshold for each of the 28K pairs every couple of days to keep the rule intact. Our minds quickly jump to a conclusion that we might be able to solve this with simple math. We can get closer using averages or percent change formulas but these formulas quickly come up short when some currencies commonly fluctuate more than others. Our minds then quickly graduate from stats 101 to 201 and we could consider the individual variance of every combination. But even this only gets us so far as time is an important dimension, the length of time or window can often be tricky to calculate. The problem gets harder when you run your basic stat model and receive multiple false positive alerts. Signal-to-noise ratio is important, confidence factors are important and down-training individual foreign currencies that don't seem to fit your statical model are important. Knowing if you copied the data incorrectly, truncated nine levels of precision on the decimal, or if the source provider sent the wrong information is important. Needing the ability to flag exceptions in production on a single currency pair while *not* flagging the other 27,445 pairs. Using a feedback loop so that the data steward interactions are captured and learned from vs having to take the same corrective action over and over. What happens when there is a typo in the currency pair or a single pair goes missing? The answer is that rules don't scale and we need much more than just one off statistical metrics to have a robust and trust worthy data quality program.

Consistency

Even when it is possible to deploy a team of smart people to build a solution to handle this use case, the question then becomes "but what about all my other data, don't I want similar yet different controls on everything?" Especially since FX Rate data by itself doesn't mean that much and is often combined with a number of other data sets to produce value. What if those data sets aren't accurate either? But those data sets have very different columns, different relationships and different time windows. DQ takes an auto-learning approach whereby it interrogates and runs fitness tests against each data set individually to devise the best statistical and learning approach. The goal being to provide an automated and elegant way to have consistent controls across all your data sets.

Auto Adapting DQ Checks for FxRate Data

```
-ds fx_rate \
-rd $rd \
-dl \
-dlkey TO_CURR,FROM_CURR \
-q "select * from FX_RATE where date = '${rd}'" \
-cxn dbConnName \
-dupe -dupeinc TO_CURR,FROM_CURR -depth 0
```

What this DQ Check Does

- Automatic correlation and relationship analysis
- Histogramming and segmentation analysis
- Anomaly detection
- Currency pair tracking
- Schema evolution
- Removes 28K static rules
- Duplicate detection for redundant currency pairs

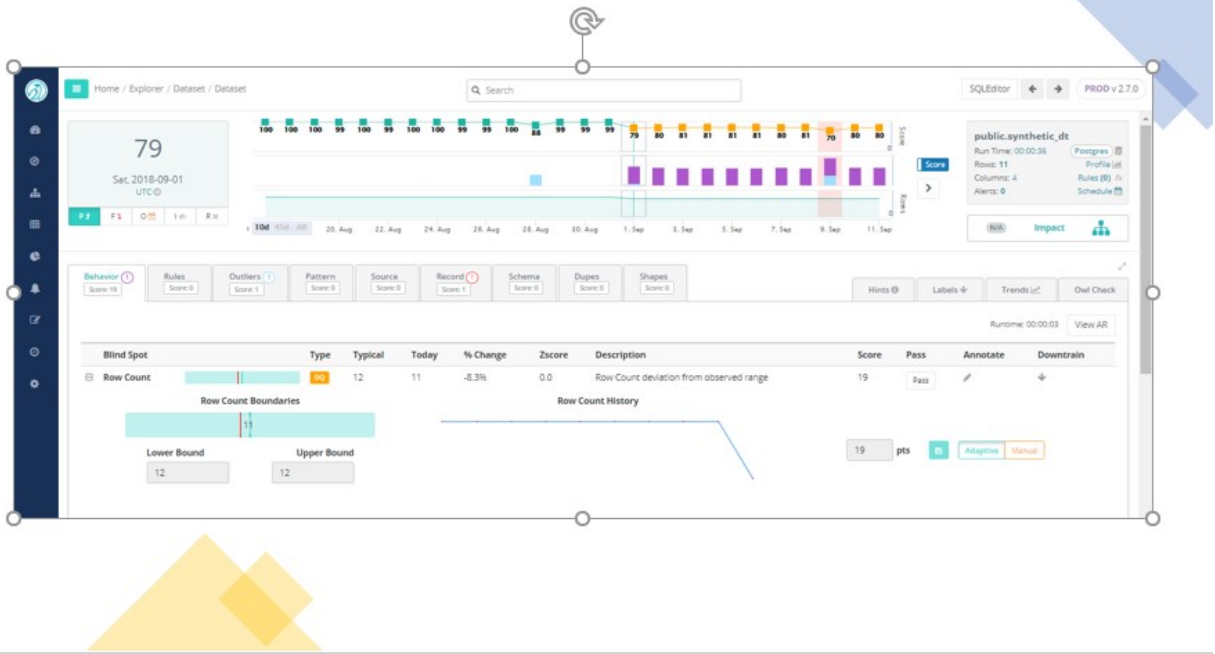
Healthcare Data Quality

Collibra DQ connects all members of the healthcare continuum with trustworthy, timely, and meaningful patient data, while reducing the time, expense, and effort required by 70 percent.

Poor data quality in healthcare is the leading problem that maligns patient outcomes. Hospitals and health information exchanges (HIEs) still struggle with patient matching issues, with most citing data quality problems and poor algorithms as top barriers to patient matching. Correctly linking patient data across organizations is a key element of value-based care, patient safety, and care coordination. Duplicate or mismatched records can result in privacy risks, claim denials, redundant medical tests or procedures, and reporting errors.

The lack of accurate and reliable data quality in healthcare leads to dire consequences that are completely preventable, as shown in DQ's troponin example below. Complete and accurate data is a vital component of our complex health system, and anything less is an

unacceptable risk. DQ provides the predictable data quality that healthcare organizations need to deliver high-quality care that we all strive to achieve.



Health Insurance Claims Data

Poor data quality is the primary cause for diagnostic providers receiving incomplete health care payments during Revenue Cycle Management (RCM).

Revenue Cycle Management is the process of identifying, collecting and managing the practice's revenue from payers based on the services provided. A complete RCM process is critical for a healthcare practice to maintain financial viability and continue to provide quality care for their patients.

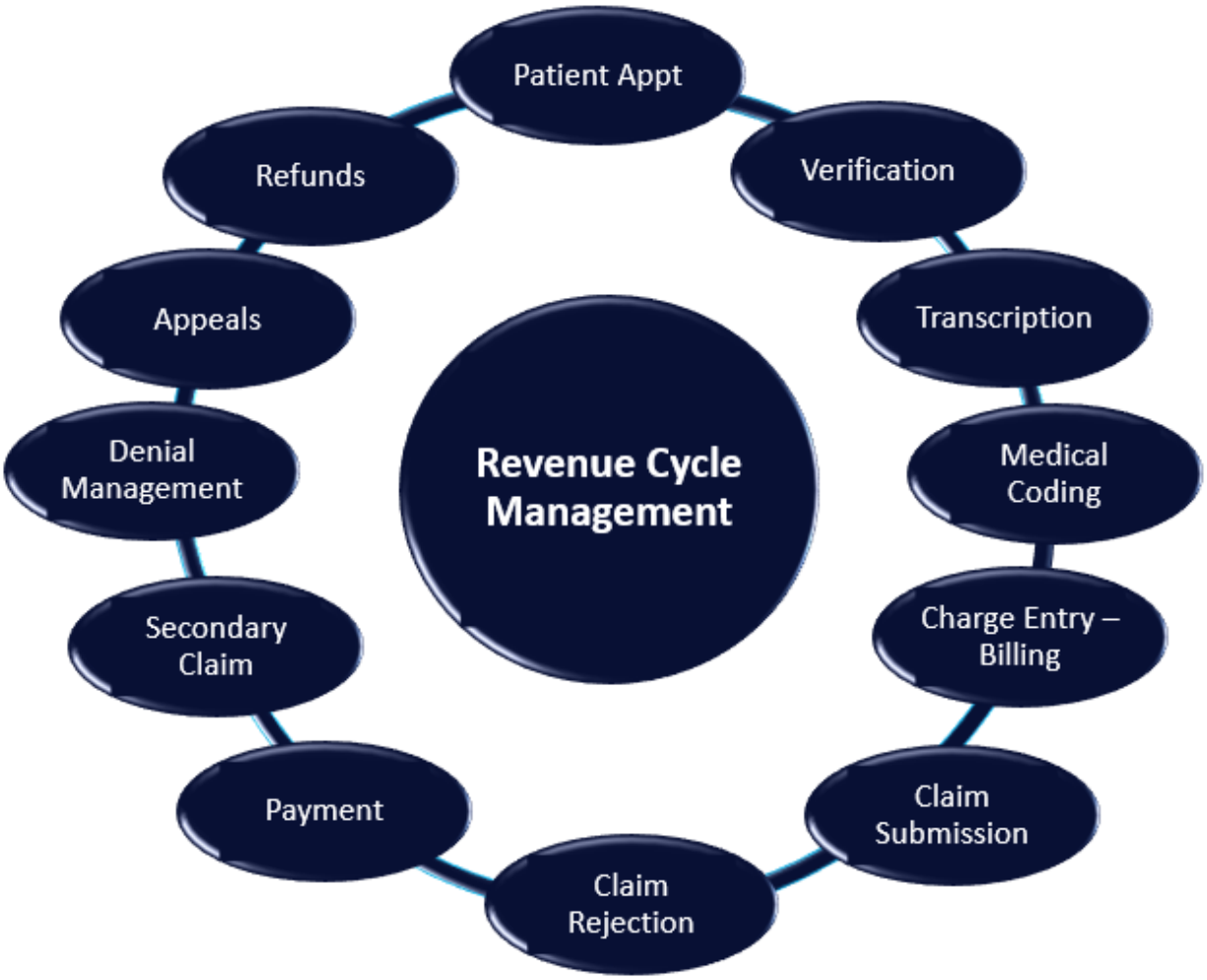
Inaccurate claims data, is the primary cause for diagnostic providers receiving incorrect payments for their services. Most providers struggle with the quality of the data that they receive, and without direct access to the patients, it can be an expensive, laborious process to correct incomplete, or missing data that is required for claim reimbursement.

Cleaning up or correcting incomplete data is not a step in the claims process that can be skipped. It must be done to assure the reimbursement process is accurate, and complete in the agreed time frame. Automating the data quality during intake is the key to the timely

completion of the reimbursement process, and saving the cost and effort of correcting the data down stream.

```
Increase revenue from insurance and patient payments
Spend less time tracking down missing patient information
Lower error processing rates
Reduce operating costs
Improve claim processing speed
```

The revenue cycle includes all the administrative and clinical functions that contribute to the capture, management and collection of patient service revenue, according to the Healthcare Financial Management Association.



```
Preregistration - Collecting preregistration information
Verification - Patient eligibility and benefit is verified
Transcription - recording the diagnoses and procedure
Medical Coding - Properly coding diagnoses and procedures.
Charge capture - Medical services into billable charges.
Claim submission - Submitting claims to insurance companies.
Claim Rejection - when necessary
Payment Posting - Determining patient balances, collection
Secondary Claim Submission
Denial Management - Applying or rejecting payments remittance
Medical Appeals - Examining the necessity of medical services.
Refund - where aplicable
```

Intraday Positions

It is common for financial organizations to receive a steady stream of files that have hourly or minutely data. The files might trail the market in a near real-time fashion. Below is an example:

```
--positions/
   |--2019/
     |--01/
       |--22/
         position_2019_01_22_09.csv
         position_2019_01_22_10.csv
         position_2019_01_22_11.csv
         position_2019_01_22_12.csv
```

File Contents @ 9am

TIME	COMPANY	TICK	SIDE	QTY
2019-01-22 09:00	T&G	xyz	LONG	300
2019-01-22 09:00	Fisher	abc	SHORT	20
2019-01-22 09:00	TradeServ	def	LONG	120

File Contents @ 10am

TIME	COMPANY	TICK	SIDE	QTY
2019-01-22 10:00	T&G	xyz	LONG	280
2019-01-22 10:00	BlackTR	ghi	SHORT	45

Notice that during the day you may or may not have a position for every company recorded. We need a way to link the "company" to its position throughout the day but not alert in cases where they simply did not trade or adjust their position. Collibra DQ offers real-time outlier detection for this scenario (see code snippet below). We also need to ensure that each company's position is only represented once per file (per hour in this case) because positions are already the aggregate view of the trades, so they should be unique. DQ offers duplicate detection (see code snippet below).

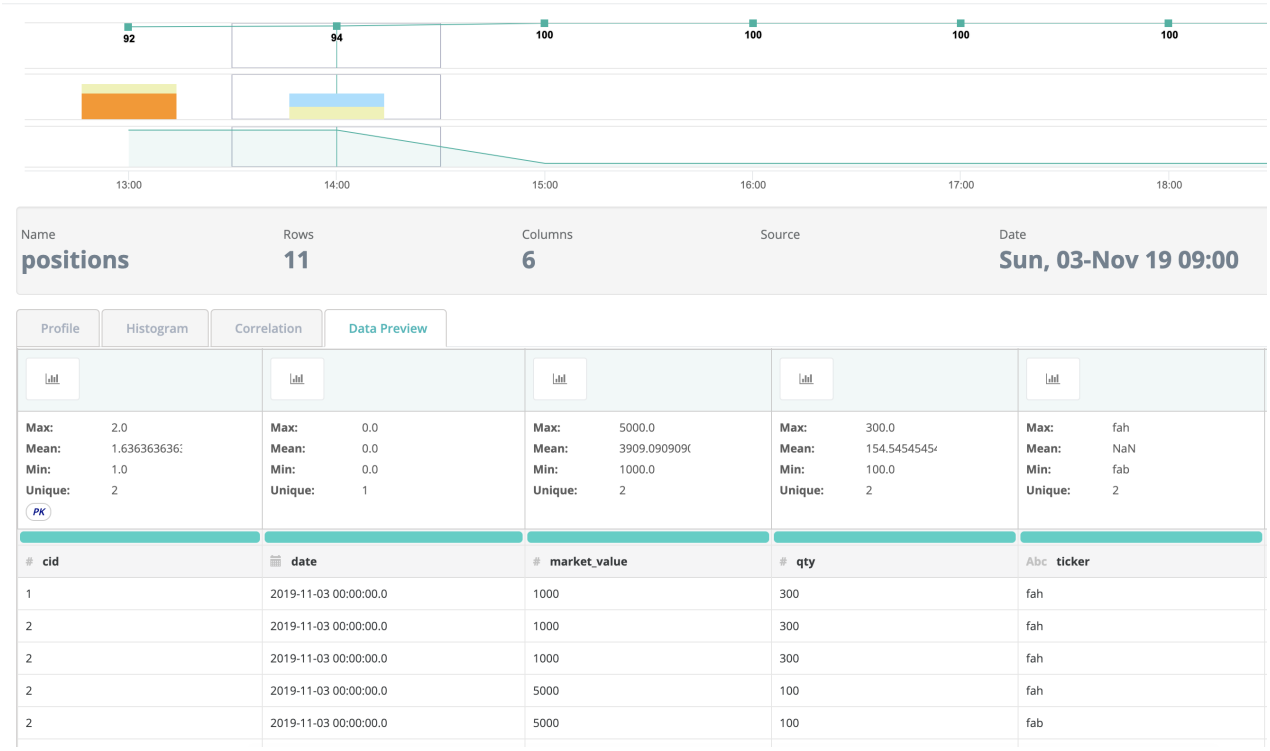
Chapter 15

Collibra DQ Pipeline

```
// Part of your pipeline includes the ingestion of files that
have the date
// and hour encoded in the file name. How do you process those
files using Owl?
//
// Format: <name>_<year>_<month>_<day>.csv

val filePath = // <set this> positions/2019/01/22/positions_
2019-01-22_09.csv

// Configure Owl.
val opt = new OwlOptions
opt.dataset = "positions"
opt.load.delimiter = ","
opt.load.fileQuery = "select * from dataset"
opt.load.filePath = file.getPath

opt.outlier.on = true
opt.outlier.key = Array("COMPANY")
opt.outlier.timeBin = TimeBin.HOUR

opt.dupe.on = true
opt.dupe.include = Array("COMPANY", "TICK")
opt.dupe.exactMatch = true

// Parse the filename to construct the run date (-rd) that will
be passed
// to Owl.
val name = file.getName.split('.').head
val parts = name.split("_")
val date = parts.slice(2, 5).mkString("-")
val hour = parts.takeRight(1).head

// Must be in format 'yyyy-MM-dd' or 'yyyy-MM-dd HH:mm'.
val rd = s"${date} ${hour}"

// Tell Owl to process data
opt.runId = rd

// Create a DataFrame from the file.
val df = OwlUtils.load(opt.load.filePath, opt.load.delimiter,
spark)

// Instantiate an OwlContext with the dataframe and our custom
configuration.
val owl = OwlUtils.OwlContext(df, spark, opt)

// Make sure Owl has catalogued the dataset.
```

```
owl.register(opt)

// Let Owl do the rest!
owl.owlCheck
```

DQ Web



DQ Coverage for Position data

- Schema evolution
- Profiling
- Correlation analysis
- Segmentation
- Outlier detection
- Duplicate detection
- Pattern mining

Security Reference Data

Pattern recognition for cross column, categorical, and conditional relationships.

Given the interconnected, automated nature of the data generated by reporting, exchanges, and source systems - hidden patterns go unnoticed.

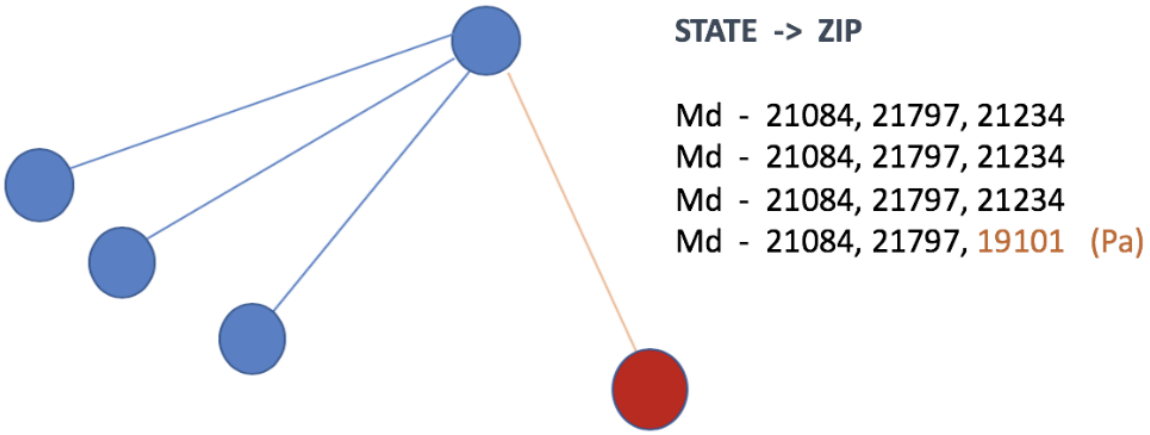
Financial firms of all shapes and sizes ingest financial data for a variety of reasons. A few vendors include Bloomberg, Thomson Reuters, ICE Data Services or SIX Financial Information.

> Note **In no uncertain terms, critical business decisions rely on the accuracy of this data.**

```
|-- RECEIVED_DATE: integer (nullable = true)
|-- B_SECURITY_ID: string (nullable = true)
|-- B_STATUS: integer (nullable = true)
|-- NUM_FIELDS: integer (nullable = true)
|-- PX_LAST: double (nullable = true)
|-- PX_FIXING: double (nullable = true)
|-- PX_BID: double (nullable = true)
|-- PX_MID: double (nullable = true)
|-- PX_ASK: double (nullable = true)
|-- ID_SEDOL1: string (nullable = true)
|-- ID_CUSIP: string (nullable = true)
|-- ID_BB_UNIQUE: string (nullable = true)
|-- LAST_UPDATE_DT: integer (nullable = true)
|-- PX_OPEN: double (nullable = true)
|-- PX_HIGH: double (nullable = true)
|-- PX_LOW: double (nullable = true)
|-- PX_VOLUME: integer (nullable = true)
|-- PRICING_SOURCE: string (nullable = true)
|-- COMPOSITE_EXCH_CODE: string (nullable = true)
|-- CRNCY: string (nullable = true)
|-- ID_MIC_PRIM_EXCH: string (nullable = true)
|-- MARKET_SECTOR_DES: string (nullable = true)
|-- EXCH_CODE: string (nullable = true)
|-- INFLATION_LINKED_INDICATOR: string (nullable = true)
|-- INFLATION_LAG: integer (nullable = true)
|-- IDX_RATIO: double (nullable = true)
|-- TRADE_CRNCY: string (nullable = true)
```

This data is not monolithic and most real-world data easily consists of over 100 columns. Maintaining the quality can be challenging, given the variety of sources feeding into just a single feed. Even the most simple quality checks can snowball into a daunting task. Everything from tickers, sedols, cusips, products, sub-products, issuers, and issuing countries can further complicate the problem. Identifying anomaly values earlier in the data ingestion process can significantly reduce downstream complexity. Furthermore, finding improbable patterns before they're used for making decisions can save costly remediation efforts.

An easy way to think about Pattern Analysis. Columns that belong 'together'.



A common application of Collibra DQ is to identify securities that are violating historical patterns. Conditional dependencies can be discovered and used as a guide to highlight misreporting. Rather than defining look-up tables, reference data sets, and predefining exact conditions - security specific patterns can be derived using probabilistic inferences.

By clicking columns that belong together, a robust framework for cross-column pattern analysis can be generated. The equivalent of dozens of multi-column, conditional rule checks can be applied with just a few clicks. In creating a confidence and probability weighted classification algorithm, this is both adaptive and granular.

Column	Key	Date	Data Type
☐ b_security_id	☑	☐	text
☐ b_status	☐	☐	int4
☑ composite_exch_code	☐	☐	text
☑ crncy	☐	☐	text
☑ exch_code	☐	☐	text

Using this technique, DQ reduces false positives, scales coverage, and quickly models a more complex series of checks than domain experts would want to develop.

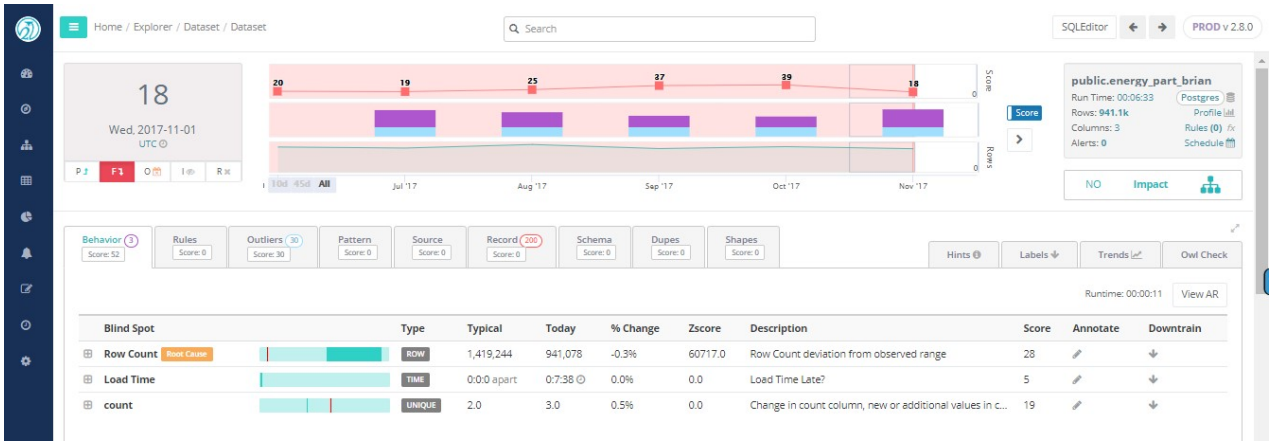
Smart Meter Data

Colliba DQ uncovered $10 million dollars in unbilled revenue for a leading U.S. energy company.

DQ's Smart Meter Data Analytics provides accurate and predictable data quality to companies often inundated with massive amounts of data and aging enterprise systems.

The U.S. smart meter data management market forecast is projected to reach $556.94 million by 2026. In 2018, U.S. electric utilities had about 86.8 million advanced (smart) metering installations. DQ provides an automated process to manage the mountain of data collected and glean critical business insights. By applying our ML algorithms during the normal data ingestion cycles, DQ uncovered $10 million dollars in unbilled revenue for a leading U.S. energy company.

In the example below, DQ detected 200 records missing from the previous run.



Consider the following opportunities that smart meter data analytics provide:

- Generate new customer insights
- Manage and prevent outages
- Improve maintenance techniques
- Build predictive models for program planning
- Develop new services and rate plans based on customer requirements

Identification of unbilled revenue: Meter events and usage information helps illustrate a picture of the customer's energy usage over time. This helps detect energy theft, meter tampering, and equipment issues that may be affecting service levels.

Outage event analysis and prevention: Today, some utilities are still unable to verify an outage unless personnel physically visit the suspected problem area to confirm. With outage event analysis, however, the utility knows the exact piece of equipment causing the problem, along with the customers directly impacted by it.

Meter quality assurance: Focusing on meter reading performance enables utilities to ensure reliability. When meter readings are expected but not delivered, the system provides an alert, and calculates overall data score from previous runs . Utilities are notified to potential data quality issues they never would have identified in the past.

It is meter data analytics that will enable utilities to tackle the problems of the future.

Validating Data Movement

Validate Data Integrity between distinct storage systems.

Record-for-Record Reconciliation

When you're copying or moving data between distinct storage systems, such as multiple HDFS clusters or between non-HDFS storage and cloud storage, it's a good idea to perform some type of validation to guarantee data integrity. This validation is essential to be sure data wasn't altered during transfer.

Detect potential data corruption caused, for example, by older versions of drivers, parsing errors, connection limits, noisy network links, memory errors on server computers and routers along the path, or software bugs (such as in a library that customers use).

Common Data Copying/Movement Scenarios

- Landing, Loading, Persisting third-party files
  - Landing daily files.
  - Loading daily files into staging location.
  - Finally, persisting data in lake or warehouse.
- Cloud Migrations
  - Between existing database storage to optimized cloud storage formats.
  - Between local file systems and cloud relational database.

- Data Lake or Data Warehouse
  - Migrating data from single storage system to distributed storage.
  - Consolidating storage systems to a single lake or warehouse.
- Same Storage, Different Environments
  - Copying same data between Dev, QA, and Prod environments.

> Warning *How do you easily validate that the same data exists in distinct locations?*

Shortcomings of Existing Validation Checks

- Low-level integrity checks like row counts and column counts may not be sufficient.
- No easy way to reconcile between across non-HDFS files and database.
- Chunk verification requires storage size, format, and metadata to be exactly equal.
- Different data types in two distinct databases (Oracle and Teradata) will not reconcile.
- Two different copies of the same files in HDFS, but with different per-file block sizes configured.
- Two different instances of HDFS with different block or chunk sizes configured.
- Copying across non-HDFS Hadoop-compatible file systems
- https://wiki.apache.org/hadoop/HCFS
- (HCFS) such as Cloud Storage.

Explicit end-to-end data integrity validation adds protection for cases that may go undetected by typical in-transit mechanisms.

Enter, Collibra DQ Integrity Validation!

To ensure and protect against target systems getting out of sync or not matching the originating source, turn on `-vs` to validate that the source matches the target. Read More

https://docs.owl-analytics.com/dq-visuals/validate-source
docs.owl-analytics.com

Complete row, column, conformity, and value checks between any two distinct storage systems can be run against high-dimension or low-dimension datasets. Works between Files and/or Database storage, On-premise, or across Cloud environments.

Get Started Today

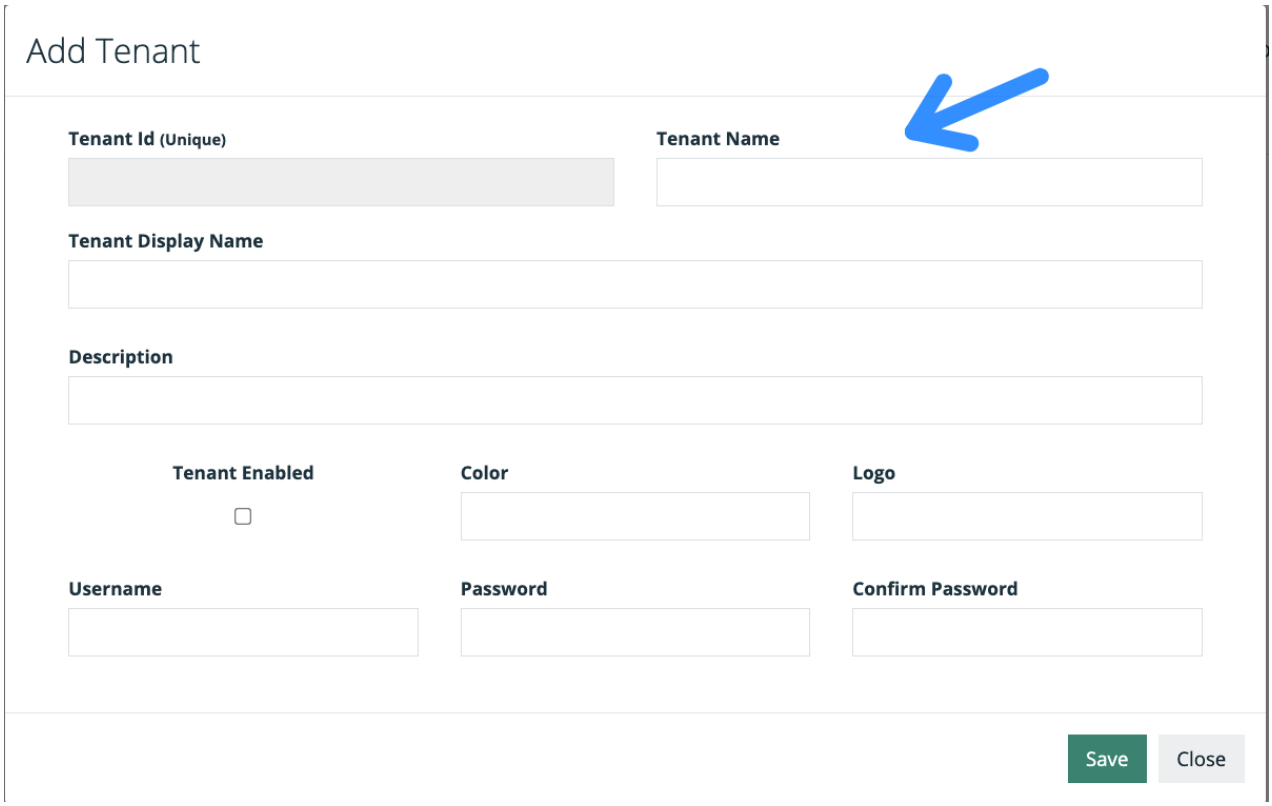
We don't want you to get stuck writing a lot of reconciliation checks we've already written. Focus on other things that actually move your project forward.

For more information, please contact **info@owl-analytics.com** or **schedule a demo at www.owldq.com** ****

Best Practices

Multi Tenant Names

> Note **Tenant names should be lower case only.**



Understanding Collibra DQ activities and what the key/date columns mean for each

- Starting with profile and expanding to rules and then other advanced capabilities.
  - https://dq-docs.collibra.com/dq-visuals/profile
- Training with DQ-team Zoom/Onsite support.
- Running with sample data.
- Introducing anomalies on sample data and running an owlcheck to see the anomalies.

Using the tool with practical scenarios

- Having Well Defined Use Cases
  - Determine a single table (dataset) that you would like to scan.
  - Have an expectation of what you would expect DQ to find in this dataset.
  - Understand which activities would capture the expected findings.
- Target internal datasets with known data issues.
- Historical Comparisons:
  - If pre-cleaned data is available with data findings that have been cleaned via legacy methods such as internal rules, run these datasets and compare the results from DQ to Internal findings.
- Work with data owners to understand findings or review expected findings.

Explorer

- The date selected with the calendar widget in the Scope (home) tab should align with the calendar widget assigned on the final (Save/Run) tab.
- If you elect to Unlock the cmd line and override the final parameters, do not re-lock or the changes will be overwritten. In general, only advanced users should override the guided settings.
- Pushdown and parallel JDBC cannot be used together. If you are using pushdown, do not select the parallel JDBC option.

Files

- File paths should not contain spaces or special characters.
- Backrun (replay) and advanced features are best suited for JDBC connections. Some features are unavailable if file and storage naming conventions do not consistently contain a date signature.

Connection Pool

If you see this message, update the agent configs in owl-env.sh or agent confg map for k8 deployments.

```
Failed to obtain JDBC Connection; nested exception is org.a-
pache.tomcat.jdbc.pool.PoolExhaustedException: [pool-29-thread-
2] Timeout: Pool empty. Unable to fetch a connection in 0
seconds, none available[size:2; busy:1; idle:0; lastwait:200].
```

Adjust these configs to modify the connection pool available.

```
export SPRING_DATASOURCE_POOL_MAX_WAIT=500
export SPRING_DATASOURCE_POOL_MAX_SIZE=10
export SPRING_DATASOURCE_POOL_INITIAL_SIZE=5
```

Freeform Agent Configs

When configuring the DQ Agent and using the Free Form Parameters at the bottom of the dialogue, you need to comma separate multiple -conf key/value pairs. I am going to write this as a forum post but use this format: "-conf some.key=x, some.other.key=y".

K8 Secrets

The following Env Vars are now managed as a Secret instead of as a Configmap:

LICENSE_KEY LIVY_SSL_KEY_PASS SERVER_SSL_KEY_PASS SPRING_AGENT_DATASOURCE_PASSWORD SPRING_AGENT_DATASOURCE_USERNAME SPRING_DATASOURCE_PASSWORD SPRING_DATASOURCE_USERNAME

Logging

When you encounter an issue in Collibra Data Quality & Observability, the log file and logs within the UI can provide information about events and help you better diagnose any problems.

Whenever there seems to be something wrong with the product, it is good practice to check the log file first.

When you report a problem to Collibra Support, it is recommended to attach the log file(s) to your support ticket.

Depending on the log and log stage, Collibra DQ logs can show you:

- Errors that have occurred on the server or agent side.
- Basic information about tasks that are performed.
- Detailed information about every action that has been performed.
- The date and time an event occurred.
- The thread that is executing an action.
- The log level of the message.

You can find the log files for Collibra DQ under **<INSTALL_PATH>/log/**.

View DQ Job log files

This section describes all of the log files available in the log directory.

> Note You can find all Collibra DQ logs under **<INSTALL_PATH>/log/**.

Log	Description
dq-web.log	This log file contains Web UI issues and error messages.

Log	Description
dq-agent.log	This file contains information about errors that occur when the Job is with the Agent. This is when Jobs are submitted to the Spark agent for processing.
dq-core.log	This file contains information about issues that occur after the Job returns to Collibra DQ after Spark submission. This log can include errors when processing drivers or any other information within the DQ Core file.

Log	Description
3rd Party Logs	These logs contain information about resource management or any 3rd party errors that pertain to DQ. These logs can include Yarn resource logs, Spark history logs, and any resource or historical logs from external data sources.
Job Logs	These logs contain the Stage 2 and Stage 3 logs and can also be accessed from the UI on Jobs page. > Note > Stage 2 and 3 logs are only applicable when Jobs submit to a cluster. If your Jobs are not submitted to a cluster, you can find your logs locally.

Collibra DQ Log Stages

This section outlines the three-stage lifecycle of a DQ Job and shows you where to find the logs for each stage.

Stage 1 logs

In Stage 1, the Agent picks up the Job from the Metastore and translates it into a valid Spark Submit request. This includes credential acquisition and injection for Cloud and Kerberos. If a job never makes it out of STAGING, the first thing to do is to check the Agent logs (**<INSTALL_HOME>/log/agent.log** or on K8s kubectl logs `<agent-pod-name> -n <namespace>`.

Stage 2 logs

Stage 2 logs contain information about particular Jobs that fail due to issues with the Agent.

In this stage, the Agent uses a Spark Submit request to pass the DQ Job to Spark. With the Job under Spark's guidance outside of Collibra DQ, the Job is not yet currently running. Spark Submit creates its own JVM to manage each submission as part of the Spark Job to the cluster or runtime. When Jobs fail at this stage, error messages such as "Failed with reason Null" appear on the Jobs page.

You can find these logs under **<INSTALL_HOME>/log/<name-of-job>.log** or from the Actions drop-down list on the Jobs page.

Stage 3 logs

Stage 3 logs contain information about particular Jobs that fail due to issues within DQ Core.

In this stage, Spark Submit instantiates the Job in the target Spark Runtime (Hadoop/K8s/Spark-Master). With the DQ Core code active, the Job is back inside Collibra DQ. Typically, when Jobs make it to this stage, they are no longer in STAGING status and any error messages appear on the Jobs page.

You can find these logs directly from the Spark Runtime, for example, Hadoop Resource Manager, K8s API via Kubectl, and so on, or from the Actions drop-down list on the Jobs page.

View Job logs in the DQ user interface

You can view the logs for individual DQ Jobs from the Jobs page. This section shows you how to access the logs from the DQ UI.

In Stage 2, Collibra DQ submits the DQ Job via Spark Submit to Spark for processing. Jobs that fail in this stage are generally due to issues with the Agent.

You can access the Stage 2 logs from the DQ UI with the following steps:

1. Go to the **Jobs** page.
2. Click **Job Actions** on the same line as your Job ID.
   » The Job Actions drop-down list expands.
3. Select **Stage 2 Logs** from the options.
   » The Stage 2 Logs open in a new window.

> Note Stage 2 logs are only applicable when Jobs submit to a cluster.

Unlike the Stage 2 logs, the DQ Core code is active and the Job is no longer under Spark's guidance, but back inside Collibra DQ. Because the Job runs in DQ Core at this point, the Stage 3 log is necessary to troubleshoot problems that happen within DQ Core.

You can access the Stage 3 logs from the DQ UI with the following steps:

1. Go to the **Jobs** page.
2. Click **Job Actions** on the same line as your Job ID.
   » The Job Actions drop-down list expands.
3. Select **Stage 3 Logs** from the options.
   » The Stage 3 Logs open in a new window.

> Note Stage 3 logs are only applicable when Jobs submit to a cluster.

> Tip
> Stage 1 logs are not accessible from the DQ UI. If a DQ Job is stuck in **Staging** status, check the Agent logs from the agent log file (**<INSTALL_HOME>/log/agent.log**) or K8s kubectl logs (`<agent-pod-name> -n <namespace>`)

Convert log outputs to JSON formatting

If you want to ingest your log outputs into a third-party log consumer, such as Elasticsearch, you can convert DQ logs to JSON formatting. This page shows you how to convert DQ logs to JSON formatting.

Steps

1. Open the **dq-web** folder and then open the **log4j2.xml** file.
2. In the **<Appenders>** section, look for the following line of code:
   ```
   <PatternLayout pattern="%d{yy/MM/dd HH:mm:ss.SSS} %X{sessionId} %X{sessionHost} %X{userName} [%t] %-5level %logger{36} - %m{nolookups}%n"/>
   ```
3. Comment out the `<PatternLayout` section with the following `<JsonLayout` to convert the output from Pattern to JSON:
   ```
   <!--<PatternLayout pattern="%d{yy/MM/dd HH:mm:ss.SSS} %X{sessionId} %X{sessionHost} %X{userName} [%t] %-5level %logger{36} - %m{nolookups}%n"/>-->
   <JsonLayout
   objectMessageAsJsonObject
   ="true" complete="false" eventEol="true" compact="true" properties="true" includeTimeMillis = "true"></JsonLayout>
   ```

   > Note You can remove the `<!-- ... -->` from the commented out line of code to revert from the JsonLayout to the PatternLayout format.

4. Set the `compact=` attribute to one of the following options:
   a. `compact="true"` for compact JSON payloads.

b. `compact="false"` for decompressed JSON payloads.

5. Optionally review the other attributes in the JSON layout.

Attribute	Data Type	Description
complete	boolean	When set to *true*, the appender includes the JSON header and footer. Records are comma-delimited. The default value is *false*.
includeTimeMillis	boolean	When set to *true*, the appender includes the timeMilis attribute in the JSON payload instead of the instant. timeMillis contains the number of milliseconds since January 1, 1970 at mid-night UTC.
objectMessageAsJsonObject	boolean	When set to *true*, ObjectMessage is serialized as JSON object to the message field of the output log. The default values is *false*.
compact	boolean	When set to *true*, the appender does not use end-of-lines or indentation. The default value is *false*.

Attribute	Data Type	Description
eventEol	boolean	When set to *true*, the appender appends an end-of-line after each record. The default value is *false*. To see one record per line, set both `eventEol="true"` and `compact="true"`.
properties	boolean	When set to *true*, the appender includes the thread context map in the generated JSON. The default value is *false*.
includeStacktrace	boolean	Optional. When set to *true*, the appender includes full stacktrace of any logged Throwable. The default value is *true*.

Tip You can optionally keep both the JSON and Pattern layouts by writing logs in a different file.

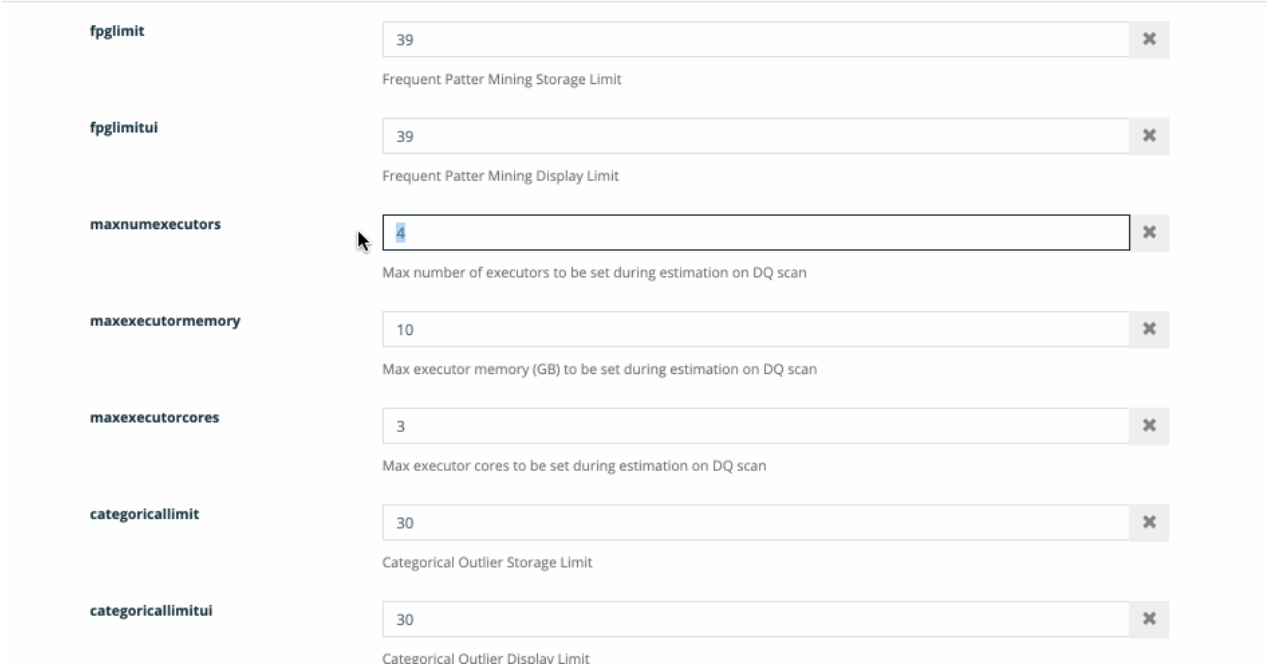
Chapter 16

Collibra DQ Benchmarks

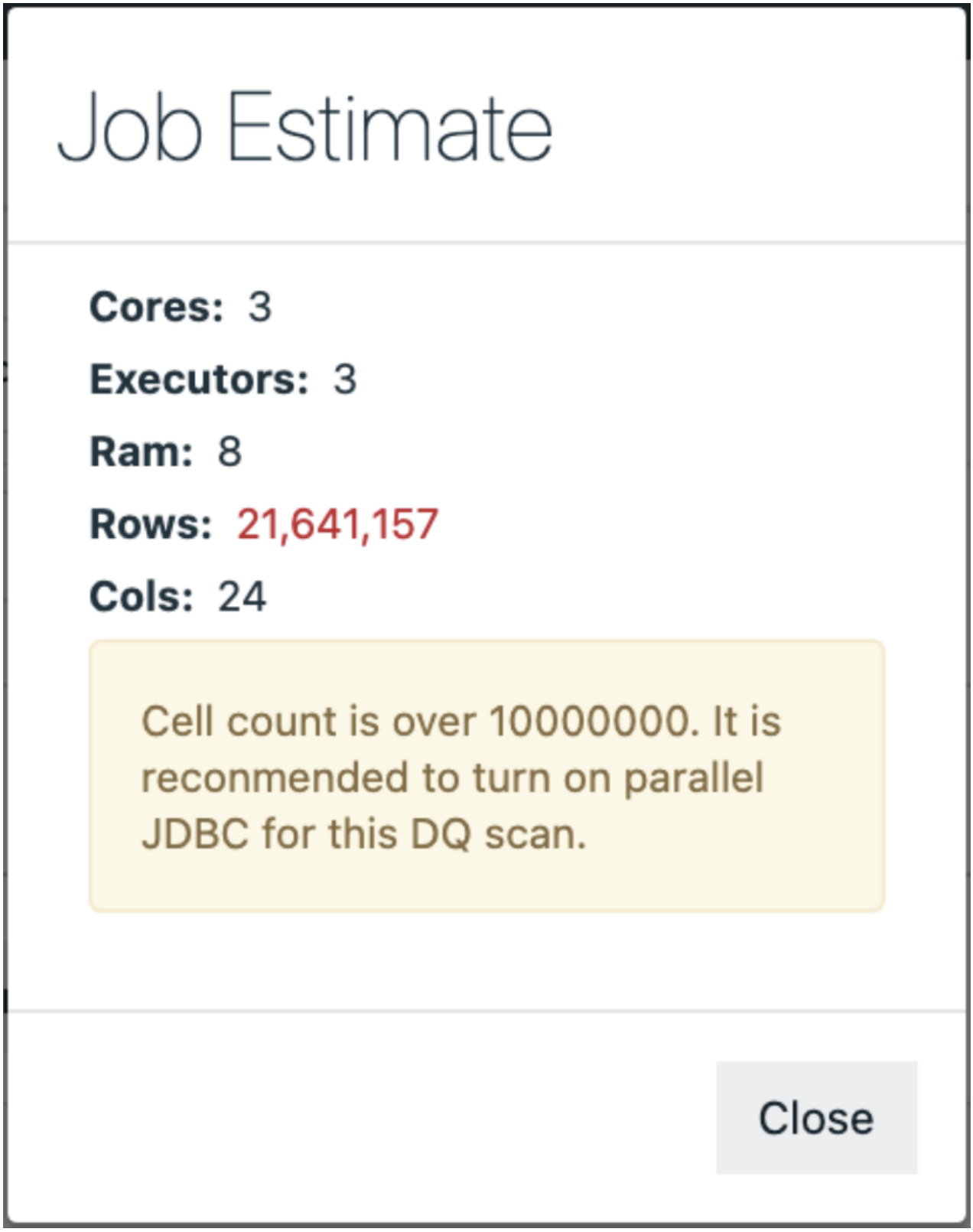
Performance Settings

Job Limits

Limits can be set to limit resources that are requested. There are options for cores, memory, executors, and cells (maxexecutorcores, maxexecutormemory, maxnumexecutors and maxcellcountforparalleljdbc).

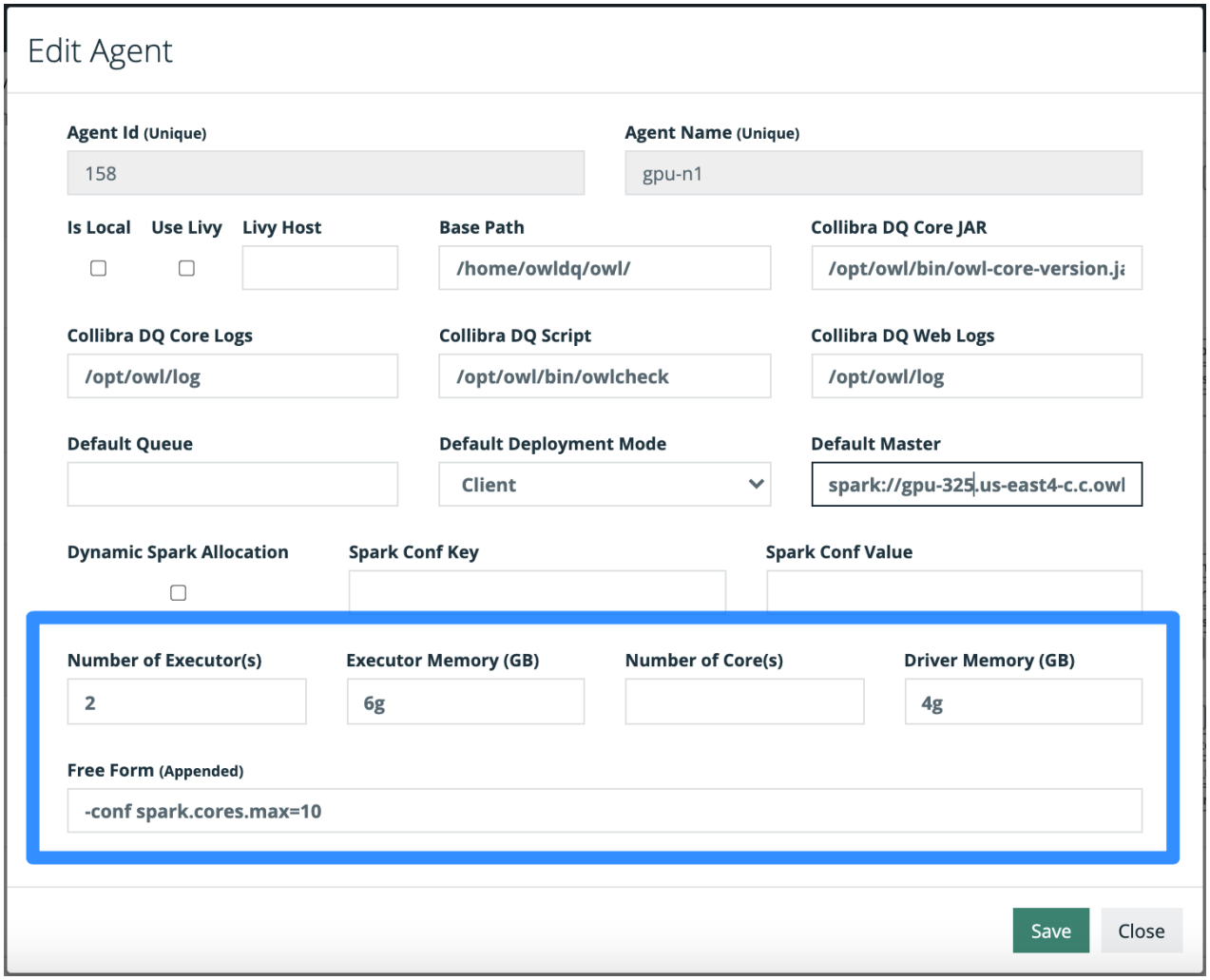


If you request more cells than the limit, you should see a warning message before hitting run.


Job Estimate
Cores: 3
Executors: 3
Ram: 8
Rows: 21,641,157
Cols: 24
Cell count is over 10000000. It is reconmended to turn on parallel JDBC for this DQ scan.
Close

Agent Defaults

Set defaults at the agent level. These should be right-sized to your environment and be used as defaults for jobs with when estimate is not available (primarily local files and remote files).



Performance Tests

Cells Per Second Performance Theory (9.5M CPS)

Rows	Cols	Cells	Seconds	Minutes	Hours	
1,000,000	1	1,000,000	0.11	0.00	0.000	
1,000,000	10	10,000,000	1.05	0.02	0.000	
1,000,000	100	100,000,000	10.53	0.18	0.003	
10,000,000	100	1,000,000,000	105.26	1.75	0.029	
100,000,000	100	10,000,000,000	1,052.63	17.54	0.292	
1,000,000,000	10	10,000,000,000	1,052.63	17.54	0.292	defining critical data elements
1,000,000,000	100	100,000,000,000	10,526.32	175.44	2.92	

Load and Profile

Dataset Name	GBs in Memory	Rows	Cols	Cells	Num Execs	Num Cores	Exec Memory	Network Time	Total Time
NYSE	0.1G	103K	9	816K	1	1	1G	00:00:15	00:00:48
AUM	14G	9M	48	432M	5	1	4G	00:01:20	00:03:50
ENERGY	5G	43M	6	258M	8	3	3G	00:00:00	00:04:35
INVEST_ DATA	20G	3.8M	158	590M	3	2	3G	00:00:40	00:03:32

NYSE

Postgres database call, no concurrent processing, simple case, small data.

```
-bhtimeoff -numexecutors 1
-lib "/opt/owl/drivers/postgres"
-executormemory 1g
-h metastore01.us-east1-b.c.owl-hadoop-
cdh.internal:5432/dev?currentSchema=public
-drivermemory 1g -master k8s:// -ds public.nyse_128 -deploymode
cluster
-q "select * from public.nyse" -bhlb 10 -rd "2020-10-26"
-driver "org.postgresql.Driver" -bhminoff
-loglevel INFO -cxn postgres-gcp -bhmaxoff
```

AUM

Postgres database call uses parallel JDBC, split on aum_id serial id.

```
-owluser kirk
-lib "/opt/owl/drivers/postgres" -datashapeoff
-numpartitions 6 -ds public.aum_dt2_50
-deploymode cluster -bhlb 10 -bhminoff
-cxn postgres-gcp -bhmaxoff -bhtimeoff
-numexecutors 6
-executormemory 4g -semanticoff
-h metastore01.us-east1-b.c.owl-hadoop-
cdh.internal:5432/dev?currentSchema=public
-columnname aum_id -corroff -drivermemory 4g -master k8s://
-q "select * from public.aum_dt2" -histoff -rd "2020-10-27"
-driver "org.postgresql.Driver" -loglevel INFO -agentjobid 7664
```

ENERGY

HDFS file with 43 million rows, converting a string date to date type, deploy mode client.

```
-f "hdfs:///demo/owl_usage_all.csv" \
-rd "2019-02-02" \
-ds energy_file \
-loglevel DEBUG -readonly \
-d "," -df dd-MMM-yy \
-master yarn \
-deploymode client  \
-numexecutors 3 \
-executormemory 10g
```

Load Profile Outliers

NYSE - 1:10 total runtime. 20 seconds for outliers

```
-bhtimeoff -owluser kirk -numexecutors 1
-lib "/opt/owl/drivers/postgres" -executormemory 1g
-dl -h metastore01.us-east1-b.c.owl-hadoop-
cdh.internal:5432/dev?currentSchema=public
-drivermemory 1g -master k8s:// -ds public.nyse_128 -deploymode
cluster
-q "select * from public.nyse" -bhlb 10
-rd "2020-10-27" -driver "org.postgresql.Driver"
-bhminoff -loglevel INFO -cxn postgres-gcp -bhmaxoff -agentjobid
7721
```

Performance Tuning

Storage Format	Num Rows	Num Columns	Bytes Disk	Num Executors	Executor Memory	Total RAM	Transfer Time	Process Time
Local File	1M	50	1G	1	3G	3G	0 mins	2 mins
HDFS File	10M	50	5G	3	8G	24G	0 mins	4 mins
Hive Table	10M	50	5G	3	8G	24G	0 mins	4 mins
JDBC Table	50M	50	25G	8	10G	80G	3 mins	8 mins
JDBC Table	10M	100	10G	3	12G	36G	3 mins	6 mins
JDBC Table	250M	9	10G	5	7G	35G	14 mins	15 mins

Storage Format	Num Rows	Num Columns	Bytes Disk	Num Executors	Executor Memory	Total RAM	Transfer Time	Process Time
JDBC Table	250M	145	70G	17	12G	204G	28 mins	30 mins

Using a 10/1 ratio of RAM to Executors is often a good rule of thumb, another and more simple option is to turn on dynamic.allocation and allow the resources to be provided as needed on demand.

Limit Columns

In most cased there are a large number of columns that go unused by the business or columns that don't require checking. One of the most efficient things you can do is limit the cols using the below cmds. As a best practice Collibra DQ strongly recommends using less than 80 columns per dataset.

```
-q "select colA, colB, colC, datCol, colD from table"
// vs
-q "select * from * from table"
```

How to limit columns when using a file

```
-fq "select colA, colB, colC from dataset"
// file query using keyword dataset
```

JDBC vs Local Data

Co-Located data (local data)

It is always a good performance practice to colocate data and processing. That doesn't mean that you tech organization chooses to do this in it's architecture and design which is why DQ accounts for both. If the data is located on the cluster that is doing the processing

use options like -hive for non JDBC and native file access. Skip tuning for JDBC as moving data to the cluster first will routinely reduce 50 percent of the performance bottleneck.

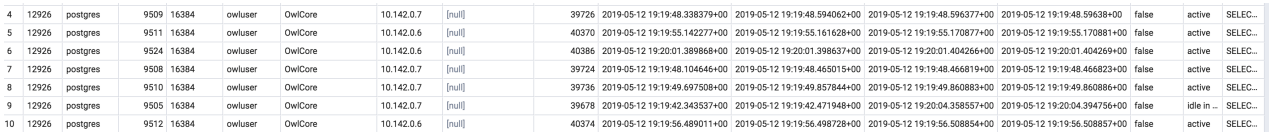
JDBC

Set fetchsize 1M rows -connectionprops fetchsize=1000 5M rows -connectionprops fetchsize=5000 10M rows -connectionprops fetchsize=10000

Set DriverMemory add more memory to the driver node as it will be responsible for the initial landing of data.

```
--driver-memory 7g
```

Add Parallel JDBC



4	12926	postgres	9509	16384	owluser	OwlCore	10.142.0.7	[null]	39726	2019-05-12 19:19:48.338379+00	2019-05-12 19:19:48.594062+00	2019-05-12 19:19:48.596377+00	2019-05-12 19:19:48.59638+00	false	active	SELEC...
5	12926	postgres	9511	16384	owluser	OwlCore	10.142.0.6	[null]	40370	2019-05-12 19:19:55.142277+00	2019-05-12 19:19:55.161628+00	2019-05-12 19:19:55.170877+00	2019-05-12 19:19:55.170881+00	false	active	SELEC...
6	12926	postgres	9524	16384	owluser	OwlCore	10.142.0.6	[null]	40386	2019-05-12 19:20:01.389868+00	2019-05-12 19:20:01.398637+00	2019-05-12 19:20:01.404266+00	2019-05-12 19:20:01.404269+00	false	active	SELEC...
7	12926	postgres	9508	16384	owluser	OwlCore	10.142.0.7	[null]	39724	2019-05-12 19:19:48.104646+00	2019-05-12 19:19:48.465015+00	2019-05-12 19:19:48.46681+00	2019-05-12 19:19:48.466823+00	false	active	SELEC...
8	12926	postgres	9510	16384	owluser	OwlCore	10.142.0.7	[null]	39736	2019-05-12 19:19:49.697508+00	2019-05-12 19:19:49.857844+00	2019-05-12 19:19:49.860883+00	2019-05-12 19:19:49.860886+00	false	active	SELEC...
9	12926	postgres	9505	16384	owluser	OwlCore	10.142.0.7	[null]	39678	2019-05-12 19:19:42.343537+00	2019-05-12 19:19:42.471948+00	2019-05-12 19:20:04.358557+00	2019-05-12 19:20:04.394756+00	false	idle in ...	SELEC...
10	12926	postgres	9512	16384	owluser	OwlCore	10.142.0.6	[null]	40374	2019-05-12 19:19:56.489011+00	2019-05-12 19:19:56.498728+00	2019-05-12 19:19:56.508854+00	2019-05-12 19:19:56.508857+00	false	active	SELEC...

Limit Features, Turn Flags Off

```
-corroff    //only losing visuals, 5% speed gain
-histoff    //only losing visuals, 4% speed gain
-hootonly   //speeds up 1% based on less logging
-readonly   //remove owl webapp read writes, 1% gain
-datashapeoff //removes Shape Detection 3% speed gain
```

Real World Scenario

Nine million rows with 46 columns on a daily basis for just 1 dataset. The data lives in Greenplum and we want to process it on a cluster platform where DQ runs. The first run results in a 12 minute runtime. While acceptable it's not ideal, here is what you should do:

1. Add Parallel JDBC for faster network.
2. Limit columns to the 18 that are of use in the downstream processing.

3. Turn off unneeded features.
4. Find out of the job is memory bound or CPU bound.

By setting the below configs this same job ran in six minutes.

```
# parallel functions
-columnname run_date -numpartitions 4 \
-lowerbound "2019-02-23 00:00:00" \
-upperbound "2019-02-24 00:00:00"
# driver optimization
-connectionprops fetchsize=6000
# analyst functions
-corroff \
-histoff
# hardware
-executormemory 4g
-numexecutors 3
```

The Full OwlCheck

```
./owlcheck  \
-u u -p pass  \
-c jdbc:postgresql://$host/postgres \                    # jdbc
url
-ds aumdt  -rd 2019-05-05  \
-q "select * from aum_dt"  \
-driver org.postgresql.Driver \                          # driver
-lib /home/owl/drivers/postgres  \                       # driver
jar
-connectionprops fetchsize=6000 \                        # driver
performance setting
-master yarn -deploymode client \
-executormemory 2G -numexecutors 2 -drivermemory 3g \    # hard-
ware sizing
-h cdh-edge.us-east1-b.c.owl-hadoop-cdh.internal:2181 \ # owl
metastore
-corroff -histoff -statsoff \                            # owl
features
-loglevel INFO \                                         # log
level
-columnname updt_ts -numpartitions 12 \                  # par-
allel jdbc
-lowerbound 1557623033193 -upperbound 1557623051585
```

```
{
  "dataset": "aumdt",
  "runId": "2019-05-05",
  "score": 100,
  "behaviorScore": 0,
  "rows": 9000000,
  "passFail": 0,
  "peak": 0,
  "avgRows": 0,
  "cols": 46,
  "runTime": "00:05:23",
  }
```


Performance Considerations

Performance Recommendations

Performance is a function of available hardware.

When running Collibra DQ Scans on a Hadoop distribution:

- Check YARN resource manager.
- Check limits on queue size.
- Contact Platform Administration team on any limitations.

When running DQ Scans on Spark standalone (single node):

- Check Spark endpoint (example: http://<IP>:<PORT>).
- Suggested maximum size on 16 core x 64GB machine: 100 million rows * 200 columns = 2 billion cells.
- If exceeding 2 billion cells, limit the width by selecting certain columns or limit depth with a WHERE clause or a FILTER condition.

When running DQ Scans on EKS:

- Check your compute pool for available pods.
- Check your worker configuration and your Spark operator configuration.
- Check minimum and maximum of allowed workers.

How-To

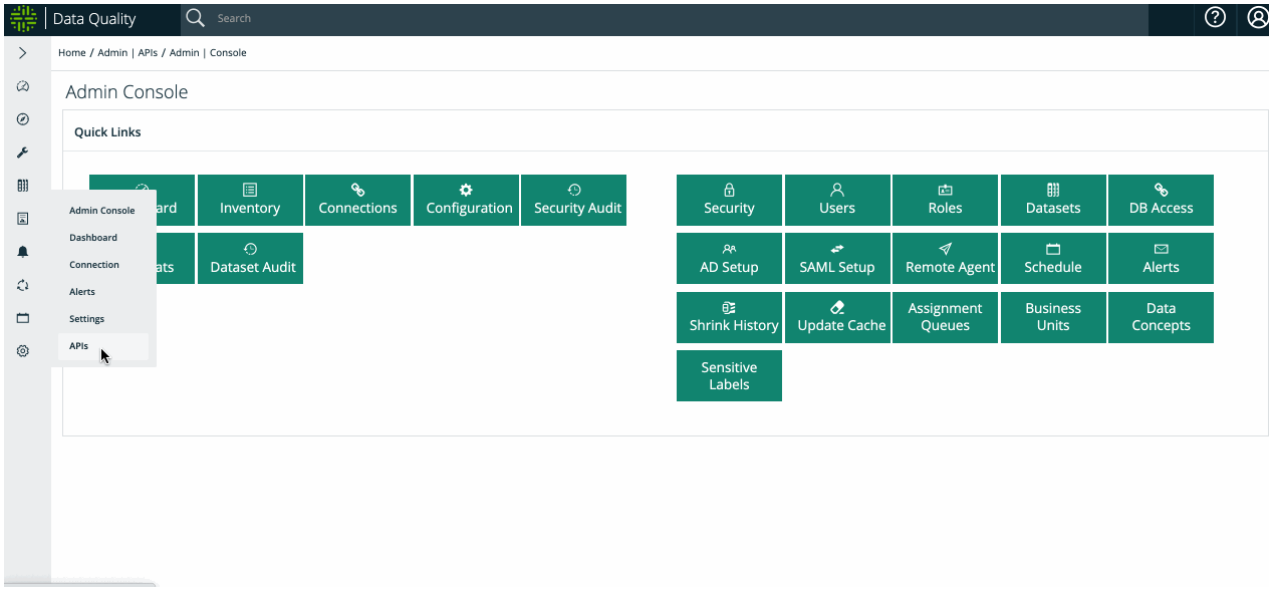
Increment DQ Scans with gradually increasing limits. Starting with a low level allows you to confirm whether database has proper indexing, skip scanning, or partitioning. Incrementing also allows you to validate security and connectivity quickly.

- Test 1: Limit 1k rows.
- Test 2: Limit 1mm rows.
- Test 3: Limit 10mm rows.

Chapter 17

Collibra DQ APIs

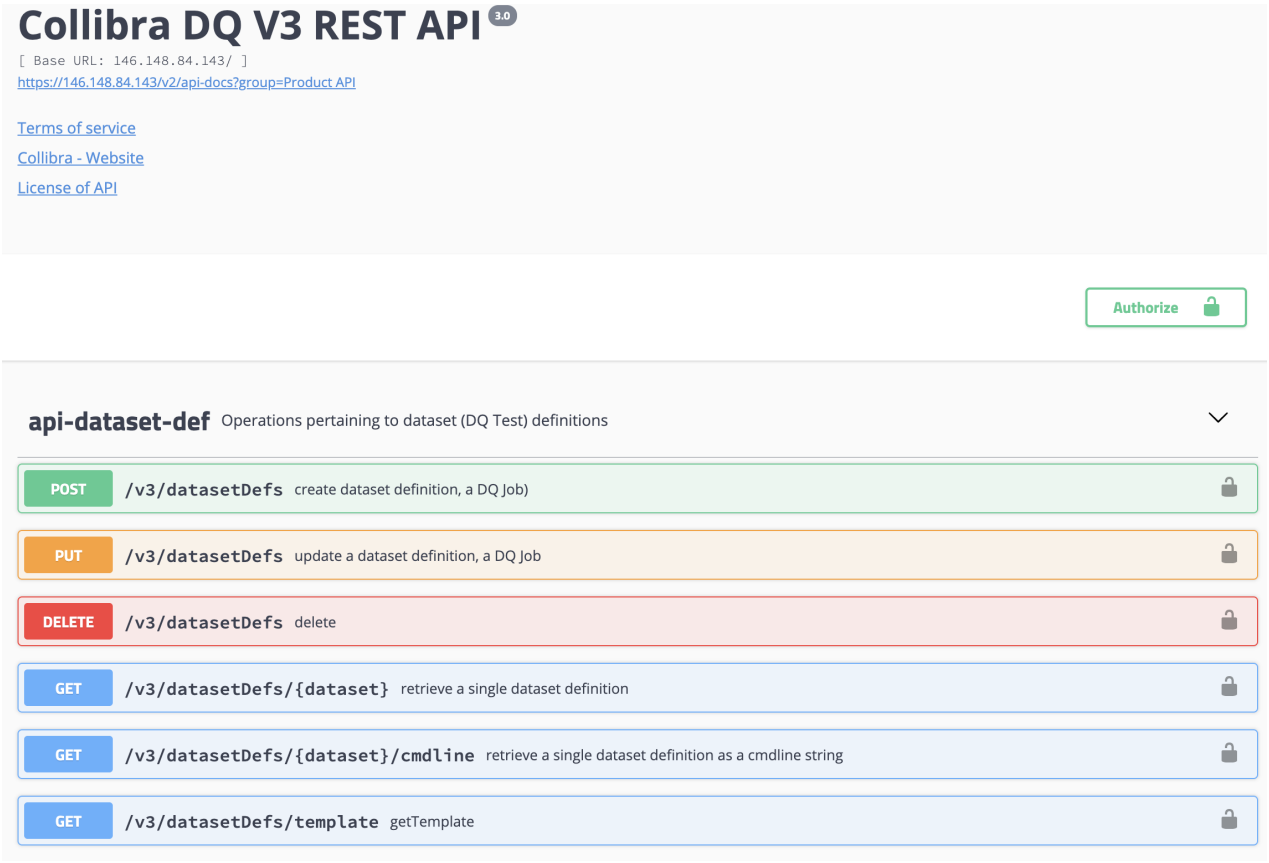
Rest



All REST APIs are available inside the application under admin section. The APIs can be used against the application in live working mode, which is preferred over documentation of APIs because it means the API works and was tested at compile time versus documentation time.

Product API

The product API is for end-users who want to interact with the official and supported API. You can also generate a client side SDK from the API with four steps below.




```
                <MadCap:concept term="Data Quality"
xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd"
/>#psuedo code example REST API

dataset = 'public.nyse'
runId = '2021-03-05'

#SAVE datasetDef
dataset = POST /v3/datasetDefs/ {json_datasetDef}

#UPDATE datasetDef
dataset = PUT /v3/datasetDefs/ {json_datasetDef}

#RUN JOB
jobId = POST /v3/jobs/run/{dataset},{runDate}

#CHECK STATUS
status = /v3/jobs{jobId}/status

#GET DQ FINDINGS
findings = /v3/jobs/{jobId}/findings
```

JWT Token For Auth

```
import requests
import json
url = "http://localhost:9000/auth/signin"
payload = json.dumps({
    "username": "<user>",
    "password": "<pass>",
    "iss": "public"
})
headers = {
    'Content-Type': 'application/json'
}
response = requests.request("POST", url, headers=headers, data-
a=payload)
print(response.text)
```

```
curl --location --request POST 'http://-
localhost:9000/auth/signin' \
--header 'Content-Type: application/json' \
--data-raw '{
    "username": "<user>",
    "password": "<pass>",
    "iss": "public"
    }'
```

Python Example

Alternatively, you can use the rest endpoints directly. This example shows how it can be done with Python.

1. Create a dataset def:
    a. Using the UI (Explorer) or
    b. Using the dataset-def-api (https://<ip>/swagger-ui.html#/dataset-def-api)
2. Confirm your Python environment has the appropriate modules and imports.
3. Fill-in the variables and customize to your preference:
    a. url, user and pass
    b. dataset, runDate, and agentName

```
import requests
import json

# Authenticate
owl = "https://<url>"
url = "https://<url>/auth/signin"
payload = json.dumps({
  "username": "<user>", # Edit Here
  "password": "<pass>", # Edit Here
  "iss": "public"       # Edit Here
})
headers = {
  'Content-Type': 'application/json'
}
response = requests.request("POST", url, headers=headers, data-
a=payload, verify=False)
owl_header = {'Authorization': 'Bearer ' + response.json()
['token']}


# Run
dataset = '<your_dataset_name>'    # Edit Here
runDate = '2021-08-08'             # Edit Here
agentName = '<your_agent_name'     # Edit Here

response = requests.post(
    url = owl +
'/v3/jobs/run?agentName='+agentName+'&dataset='+dataset+'&run-
Date='+runDate,
    headers=owl_header,
    verify=False
)

jobId = str(response.json()['jobId'])


# Status
for stat in range(100):
    time.sleep(1)

    response = requests.get(
        url = owl + '/v3/jobs/'+jobId,
        headers=owl_header,
        verify=False
    )

    job = response.json()
```

```
    if job['status'] == 'FINISHED':
        break


# Results
response = requests.get(
    url = owl + '/v3/jobs/'+jobId+'/findings',
    headers=owl_header,
    verify=False
)

print(response.json())
```

This assumes you have created a data set definition using the UI or from the template.

Command Line instead of JSON dataset def

You can run a similar job submission using the cmd line. Please note it is easiest to get the saved command line from the dataset-def-api **/v3/datasetDefs//cmdline** (with proper escaping) and passed to the **/v3/jobs/runCmdLine**.

Breaking Down The Sections

Submit the Job

Send in a data set name, date and agent to submit the job. This kicks off the engine to go do the work.

```
# Run
dataset = 'API_V3'
runDate = '2021-08-08'
agentName = 'owldq-owl-agent-owldq-dev-0'

response = requests.post(
    url = owl +
'/v3/jobs/run?agentName='+agentName+'&dataset='+dataset+'&run-
Date='+runDate,
    headers=owl_header
)

jobId = str(response.json()['jobId'])
```

Get the Status

Using the jobId returned from the job submission, you can check the status. In the example above, there is an interval to wait for the job to complete. You can create your own logic and orchestrate more precisely.

```
response = requests.get(
    url = owl + '/v3/jobs/'+jobId,
    headers=owl_header
)
```

Get the Results

Using the same jobId returned from the job submission, you can check the results. You will get a detailed json object with all the capabilities and detections in one payload. This is where you would decision, based on your organization and process.

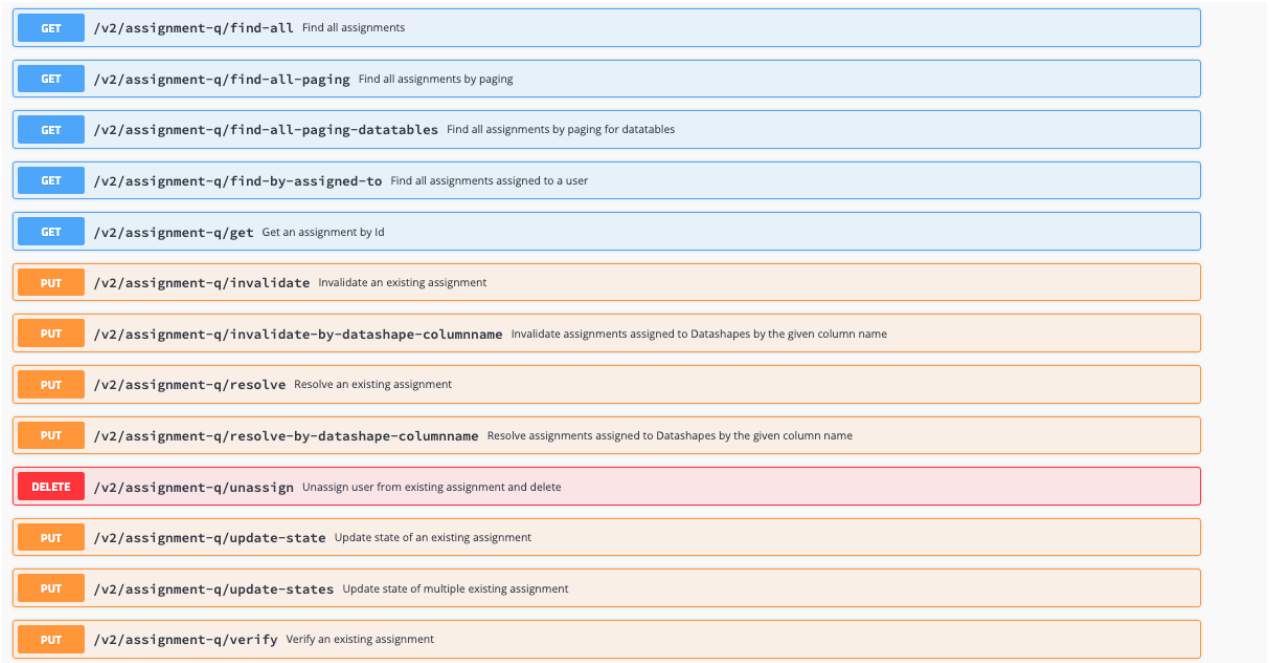
```
response = requests.get(
    url = owl + '/v3/jobs/'+jobId,
    headers=owl_header
)
```

Python Example Raw

```
import requests
import json

# Variables
owl = 'https://<ip_address>'                #Edit
user = '<user>'                             #Edit
password = '<password>'                     #Edit
tenant = 'public'                           #Edit
dataset = '<your_dataset_name>'             #Edit
runDate = '2021-08-08'                      #Edit
agentName = 'your_agent_name'               #Edit

# Authenticate
url = owl+'/auth/signin'
payload = json.dumps({"username": user, "password": password,
"iss": tenant })
headers = {'Content-Type': 'application/json'}
response = requests.request("POST", url, headers=headers, data-
a=payload, verify=False)
owl_header = {'Authorization': 'Bearer ' + response.json()
['token']}

# Run
response = requests.post(url = owl +
'/v3/jobs/run?agentName='+agentName+'&dataset='+dataset+'&run-
Date='+runDate, headers=owl_header, verify=False)
jobId = str(response.json()['jobId'])

# Status
for stat in range(100):
    time.sleep(1)

    response = requests.get(url = owl + '/v3/jobs/'+jobId, head-
ers=owl_header, verify=False)
    status = response.json()['status']

    if status == 'FINISHED':
        break

# Results
response = requests.get(url = owl + '/v3/jobs/'+jobId+'/find-
ings', headers=owl_header,  verify=False)
```

Internal API

Collibra DQ also exposes the internal API so that all potential operations are available. The caveat is that these calls may change over time or expose underlying functionality.



Data Set Definition

The JSON for the full data set definition. It can be more terse to send in the cmdline string of just the variables you use for your DQ Job.

```
-df "yyyy/MM/dd" -owluser <user> -numexecutors 1 -executormemory
1g \
-f s3a://s3-datasets/dataset.csv -h <host>:5432/dev?-
currentSchema=public \
-fq "select * from dataset" -drivermemory 1g -master k8s:// -ds
dataset_csv_1 \
-deploymode cluster -bhlb 10 -rd "2021-04-01" -fullfile -log-
level INFO -cxn s3test5 \
-sparkprinc user2@CW.COM -sparkkeytab /tmp/user2.keytab
```

```
{
    "dataset": "",
    "runId": "",
    "runIdEnd": "",
    "runState": "DRAFT",
    "passFail": 1,
    "passFailLimit": 75,
    "jobId": 0,
    "coreMaxActiveConnections": null,
    "linkId": null,
    "licenseKey": "",
    "logFile": "",
    "logLevel": "",
    "hootOnly": false,
    "prettyPrint": true,
    "useTemplate": false,
    "parallel": false,
    "plan": false,
    "dataPreviewOff": false,
    "datasetSafeOff": false,
    "obslimit": 300,
    "pgUser": "",
    "pgPassword": "",
    "host": null,
    "port": null,
    "user": "anonymous : use -owluser",
    "alertEmail": null,
    "scheduleTime": null,
    "schemaScore": 1,
    "optionAppend": "",
    "keyDelimiter": "~~",
    "agentId": null,
    "load": {
      "readonly": false,
      "passwordManager": null,
      "alias": "",
      "query": "",
      "key": "",
      "expression": "",
      "addDateColumn": false,
      "zeroFillNull": false,
      "replaceNulls": "",
      "stringMode": false,
      "operator": null,
      "dateColumn": null,
      "transform": null,
      "filter": "",
      "filterNot": "",
```

```
      "sample": 1,
      "backRun": 0,
      "backRunBin": "DAY",
      "unionLookBack": false,
      "cache": true,
      "dateFormat": "yyyy-MM-dd",
      "timeFormat": "HH:mm:ss.SSS",
      "timestamp": false,
      "filePath": "",
      "fileQuery": "",
      "fullFile": false,
      "fileHeader": null,
      "inferSchema": true,
      "fileType": null,
      "delimiter": ",",
      "fileCharSet": "UTF-8",
      "skipLines": 0,
      "avroSchema": "",
      "xmlRowTag": "",
      "flatten": false,
      "handleMaps": false,
      "handleMixedJson": false,
      "multiLine": false,
      "lib": "",
      "driverName": null,
      "connectionName": "",
      "connectionUrl": "",
      "userName": "",
      "password": "",
      "connectionProperties": {},
      "hiveNative": null,
      "hiveNativeHWC": false,
      "useSql": true,
      "columnName": null,
      "lowerBound": null,
      "upperBound": null,
      "numPartitions": 0,
      "escapeWithBackTick": false,
      "escapeWithSingleQuote": false,
      "escapeWithDoubleQuote": false,
      "escapeCharacter": "",
      "hasHeader": true
    },
    "outliers": [
      {
        "id": null,
        "on": false,
        "only": false,
```

```
        "lookback": 5,
        "key": null,
        "include": null,
        "exclude": null,
        "dateColumn": null,
        "timeColumn": null,
        "timeBin": "DAY",
        "timeBinQuery": "",
        "categorical": true,
        "by": null,
        "limit": 300,
        "minHistory": 3,
        "historyLimit": 5,
        "score": 1,
        "aggFunc": "",
        "aggQuery": "",
        "query": "",
        "q1": 0.15,
        "q3": 0.85,
        "categoricalColumnConcatenation": false,
        "limitCategorical": null,
        "measurementUnit": "",
        "multiplierUpper": 1.35,
        "multiplierLower": 1.35,
        "record": true,
        "filter": null,
        "combine": true,
        "categoricalConfidenceType": "",
        "categoricalTopN": 3,
        "categoricalBottomN": 2,
        "categoricalMaxConfidence": 0.02,
        "categoricalMaxFrequencyPercentile": 0.25,
        "categoricalMinFrequency": 1,
        "categoricalMinVariance": 0,
        "categoricalMaxCategoryN": 1,
        "categoricalParallel": true,
        "categoricalAlgorithm": "",
        "categoricalAlgorithmParameters": {}
      }
    ],
    "outlier": {
      "id": null,
      "on": false,
      "only": false,
      "lookback": 5,
      "key": null,
      "include": null,
      "exclude": null,
```

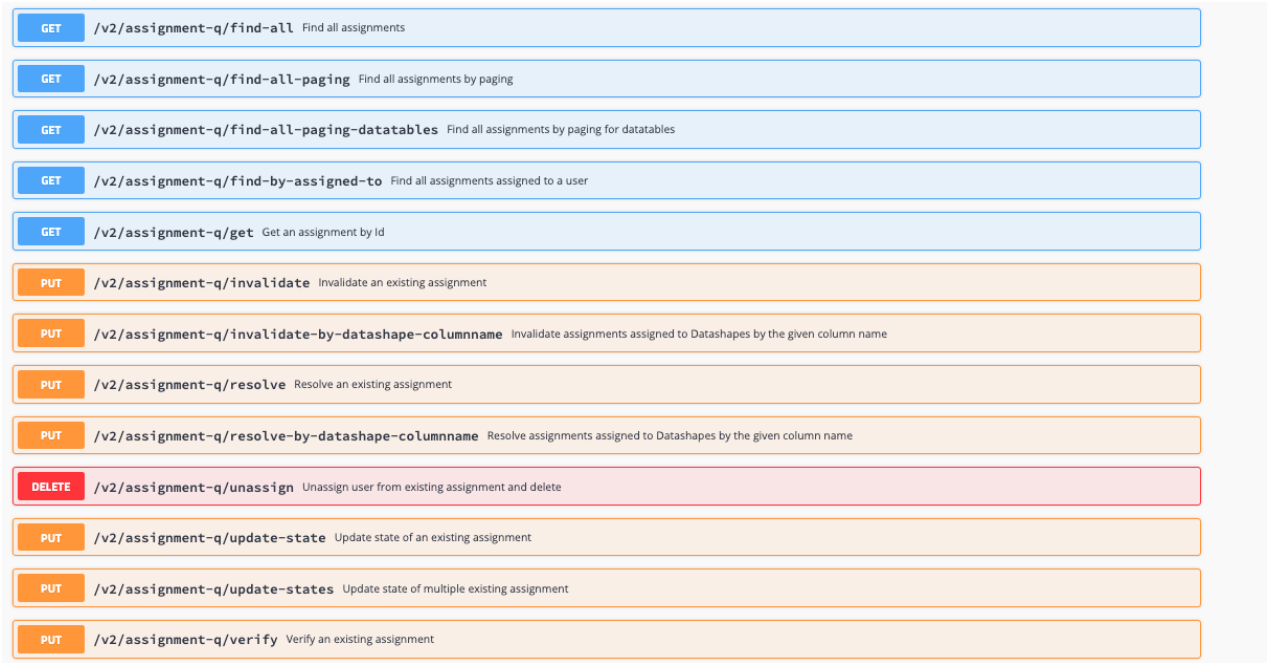
```
      "dateColumn": null,
      "timeColumn": null,
      "timeBin": "DAY",
      "timeBinQuery": "",
      "categorical": true,
      "by": null,
      "limit": 300,
      "minHistory": 3,
      "historyLimit": 5,
      "score": 1,
      "aggFunc": "",
      "aggQuery": "",
      "query": "",
      "q1": 0.15,
      "q3": 0.85,
      "categoricalColumnConcatenation": false,
      "limitCategorical": null,
      "measurementUnit": "",
      "multiplierUpper": 1.35,
      "multiplierLower": 1.35,
      "record": true,
      "filter": null,
      "combine": true,
      "categoricalConfidenceType": "",
      "categoricalTopN": 3,
      "categoricalBottomN": 2,
      "categoricalMaxConfidence": 0.02,
      "categoricalMaxFrequencyPercentile": 0.25,
      "categoricalMinFrequency": 1,
      "categoricalMinVariance": 0,
      "categoricalMaxCategoryN": 1,
      "categoricalParallel": true,
      "categoricalAlgorithm": "",
      "categoricalAlgorithmParameters": {}
    },
    "pattern": {
      "id": null,
      "only": false,
      "lookback": 5,
      "key": null,
      "dateColumn": null,
      "include": null,
      "exclude": null,
      "score": 1,
      "minSupport": 0.000033,
      "confidence": 0.6,
      "limit": 30,
      "query": "",
```

```
      "filter": null,
      "timeBin": "DAY",
      "on": false,
      "match": true,
      "lowFreq": false,
      "bucketLimit": 450000,
      "deDupe": true
    },
    "patterns": [
      {
        "id": null,
        "only": false,
        "lookback": 5,
        "key": null,
        "dateColumn": null,
        "include": null,
        "exclude": null,
        "score": 1,
        "minSupport": 0.000033,
        "confidence": 0.6,
        "limit": 30,
        "query": "",
        "filter": null,
        "timeBin": "DAY",
        "on": false,
        "match": true,
        "lowFreq": false,
        "bucketLimit": 450000,
        "deDupe": true
      }
    ],
    "dupe": {
      "on": false,
      "only": false,
      "include": null,
      "exclude": null,
      "depth": 0,
      "lowerBound": 99,
      "upperBound": 100,
      "approximate": 1,
      "limitPerDupe": 15,
      "checkHeader": true,
      "filter": null,
      "ignoreCase": true,
      "score": 1,
      "limit": 300
    },
    "profile": {
```

```
      "on": true,
      "only": false,
      "include": null,
      "exclude": null,
      "shape": true,
      "correlation": null,
      "histogram": null,
      "semantic": null,
      "limit": 300,
      "histogramLimit": 0,
      "score": 1,
      "shapeTotalScore": 0,
      "shapeSensitivity": 0,
      "shapeMaxPerCol": 0,
      "shapeMaxColSize": 0,
      "shapeGranular": null,
      "behavioralDimension": "",
      "behavioralDimensionGroup": "",
      "behavioralValueColumn": "",
      "behaviorScoreOff": false,
      "behaviorLookback": 10,
      "behaviorMinSupport": 4,
      "profilePushDown": null,
      "behaviorRowCheck": true,
      "behaviorTimeCheck": true,
      "behaviorMinValueCheck": true,
      "behaviorMaxValueCheck": true,
      "behaviorNullCheck": true,
      "behaviorEmptyCheck": true,
      "behaviorUniqueCheck": true,
      "adaptiveTier": null
    },
    "source": {
      "on": false,
      "only": false,
      "validateValues": false,
      "matches": false,
      "sourcePushDownCount": false,
      "include": null,
      "exclude": null,
      "includeSrc": null,
      "excludeSrc": null,
      "key": null,
      "map": null,
      "score": 1,
      "limit": 30,
      "dataset": "",
      "driverName": "",
```

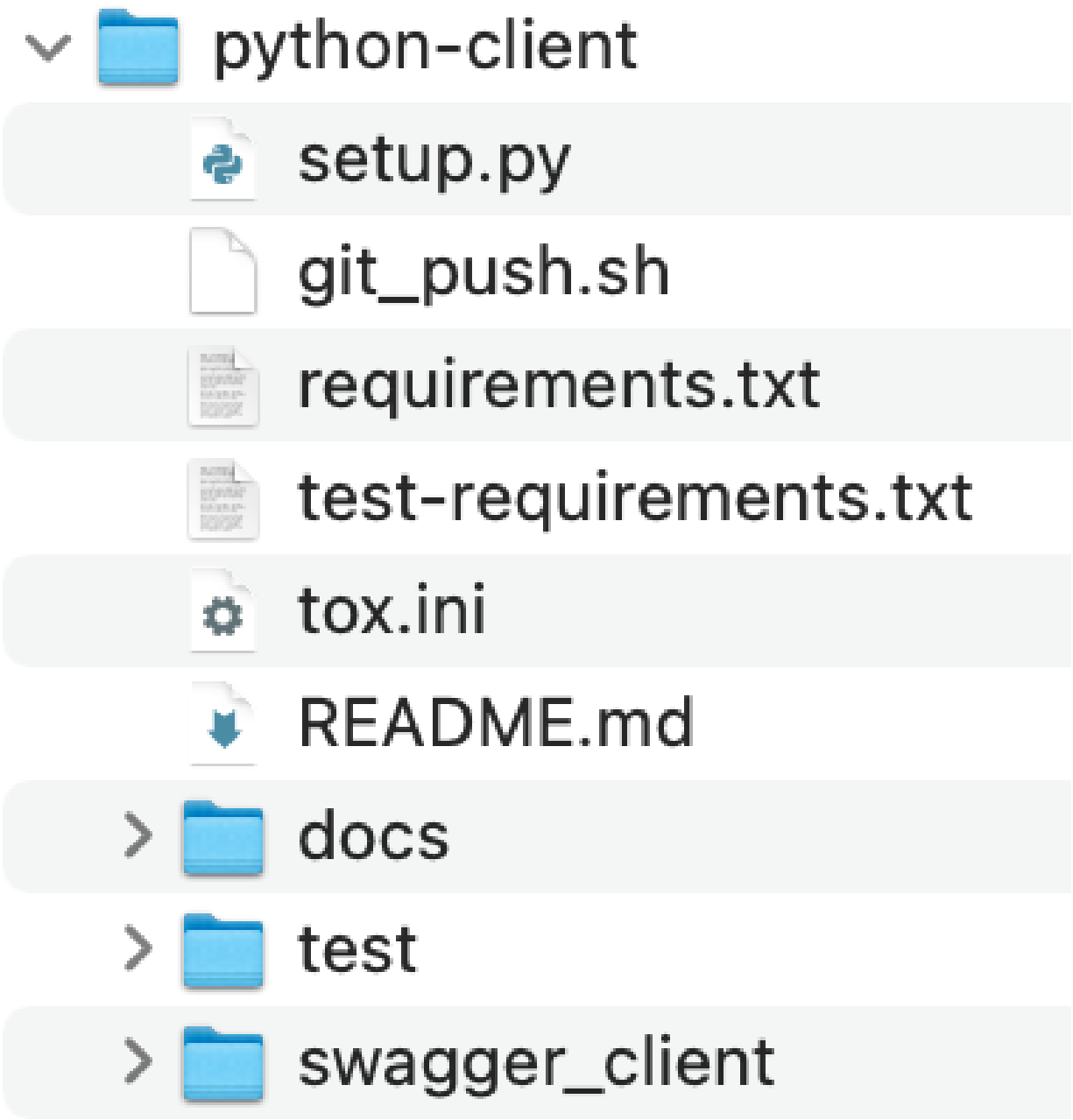
```
      "user": "",
      "password": "",
      "passwordManager": "",
      "connectionName": "",
      "connectionUrl": "",
      "query": "",
      "lib": "",
      "checkType": true,
      "checkCase": false,
      "validateValuesFilter": "",
      "validateSchemaOrder": false,
      "connectionProperties": {},
      "filePath": "",
      "fileQuery": "",
      "fullFile": false,
      "header": null,
      "skipLines": 0,
      "inferSchema": true,
      "fileType": null,
      "delimiter": ",",
      "fileCharSet": "UTF-8",
      "avroSchema": "",
      "xmlRowTag": "",
      "flatten": false,
      "handleMaps": false,
      "handleMixedJson": false,
      "multiLine": false,
      "hasHeader": true
    },
    "rule": {
      "on": true,
      "only": false,
      "lib": null,
      "name": "",
      "absoluteScoring": false,
      "ruleBreakPreviewLimit": 6
    },
    "colMatch": {
      "colMatchParallelProcesses": 3,
      "colMatchDurationMins": 20,
      "colMatchBatchSize": 2,
      "level": "exact",
      "fuzzyDistance": 1,
      "connectionList": []
    },
    "spark": {
      "numExecutors": 3,
      "driverMemory": "",
```

```
      "executorMemory": "",
      "executorCores": 1,
      "conf": "",
      "queue": "",
      "master": "local[*]",
      "principal": "",
      "keyTab": "",
      "deployMode": "",
      "jars": null,
      "packages": null,
      "files": null
    },
    "env": {
      "jdbcPrincipal": "",
      "jdbcKeyTab": ""
    },
    "record": {
      "on": false,
      "in": "",
      "notIn": "",
      "include": null,
      "percDeltaLimit": 0.1,
      "score": 1
    },
    "transforms": [],
    "pipeline": []
  }
```

Generate Client SDK

1. Go to https://editor.swagger.io/.
2. Click File Import URL.
3. Paste a URL that looks like this https://<host>/v2/api-docs?group=Product%20API.
4. Click generate client (python, java, scala, C#).



```
#Python SDK Example

#GET CMDLINE
cmdLine = get_job_cmdline(dataset)

#SUBMIT JOB
job_id = run(dataset, run_date)

#CHECK STATUS
status = get_job_status(job_id)

#GET DQ ISSUES
status = get_job_findings(dataset, run_date)
```

Swagger

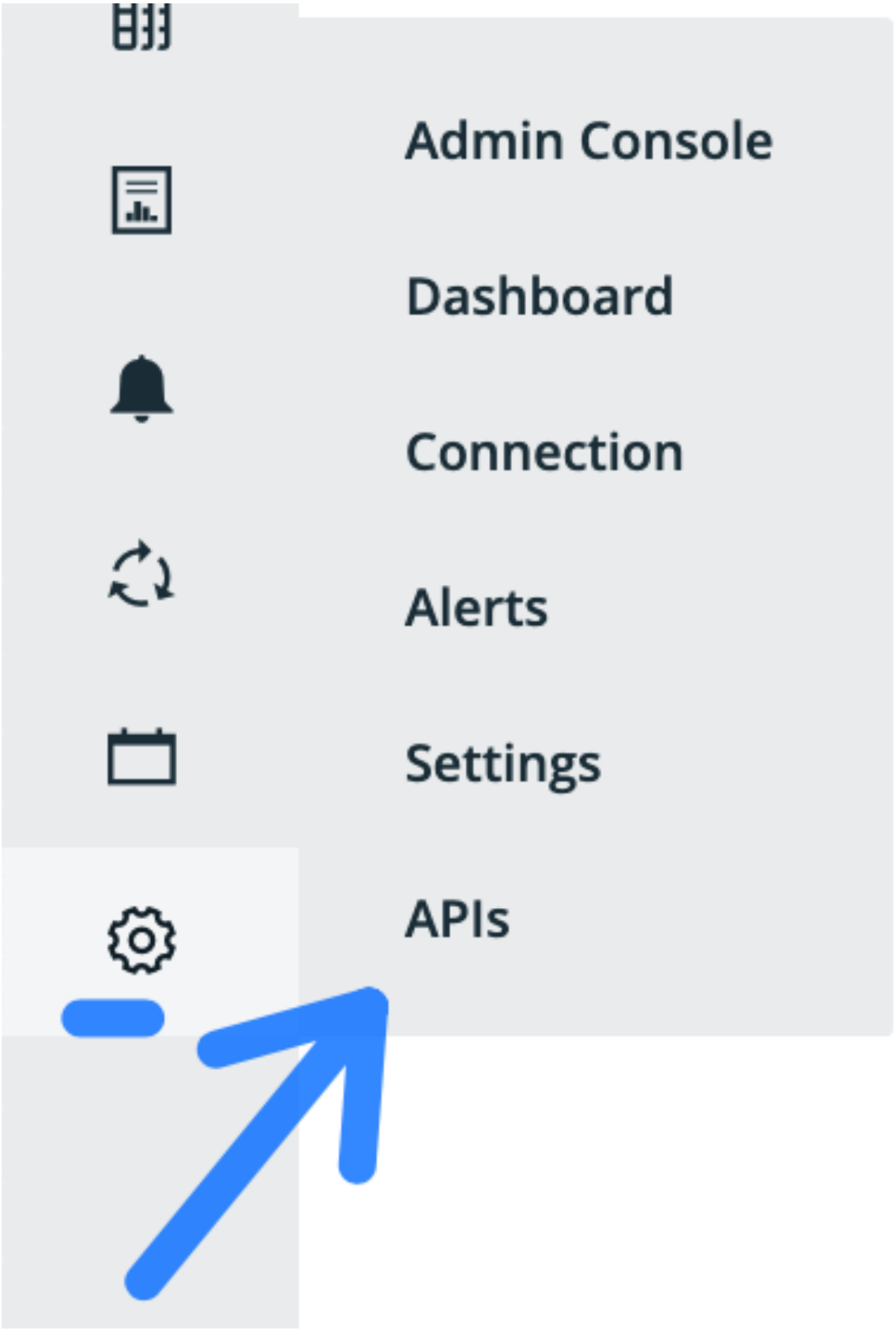
You can find Swagger in the Collibra DQ application.

Docs built into the application

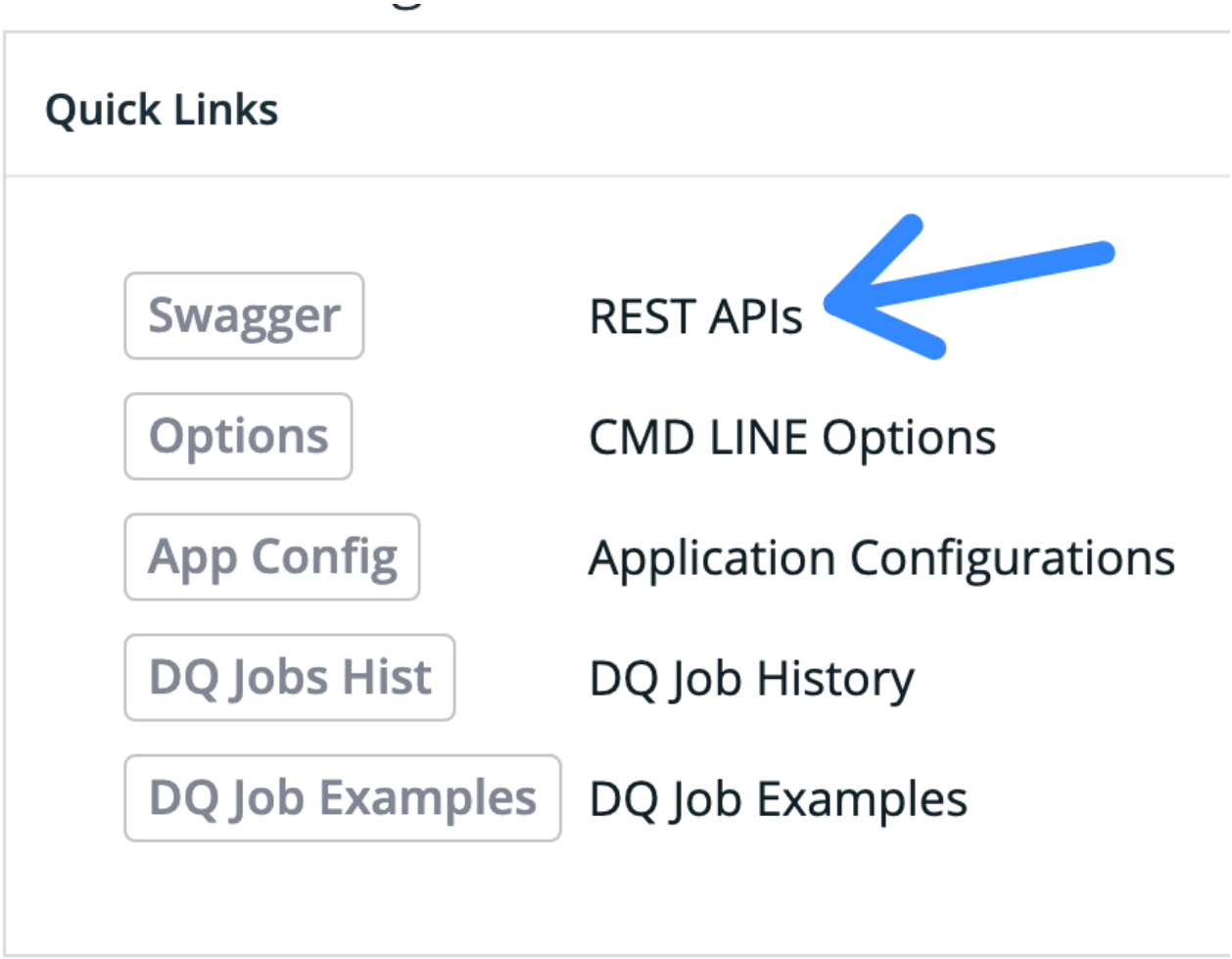
Collibra DQ comes with full swagger support out of the box.

```
http://<YOUR_IP_ADDRESS>/swagger-ui.html
```

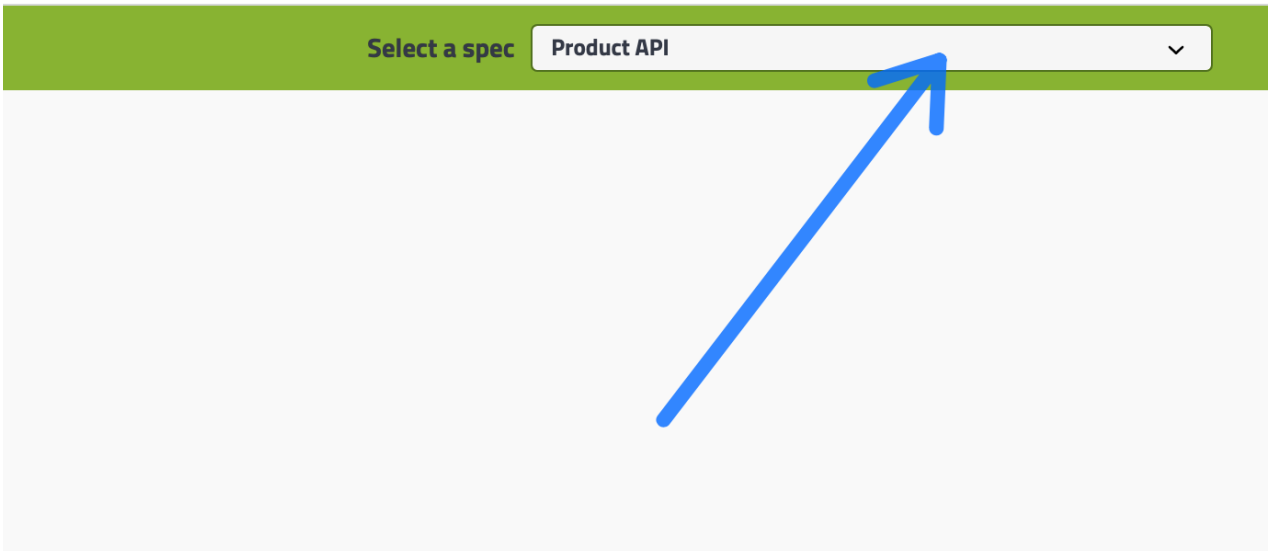
Swagger can be found in the application under the Admin section labeled APIs.



You will find a direct link to the Swagger page



Toggle between Product API and Internal API.



For example swagger API please visit - http://<YOUR_IP>:9000/v2/api-docs?group=UI Internal.

swagger

Api Documentation 1.0

[Base URL: 18.232.221.162/]
http://18.232.221.162/v2/api-docs

Api Documentation

Terms of service

Apache 2.0

basic-error-controller Basic Error Controller

controller Controller

controller-agent Controller Agent

controller-alert Controller Alert

controller-graph Controller Graph

controller-health Controller Health

controller-label Controller Label

controller-owl-check Controller Owl Check

controller-rule Controller Rule

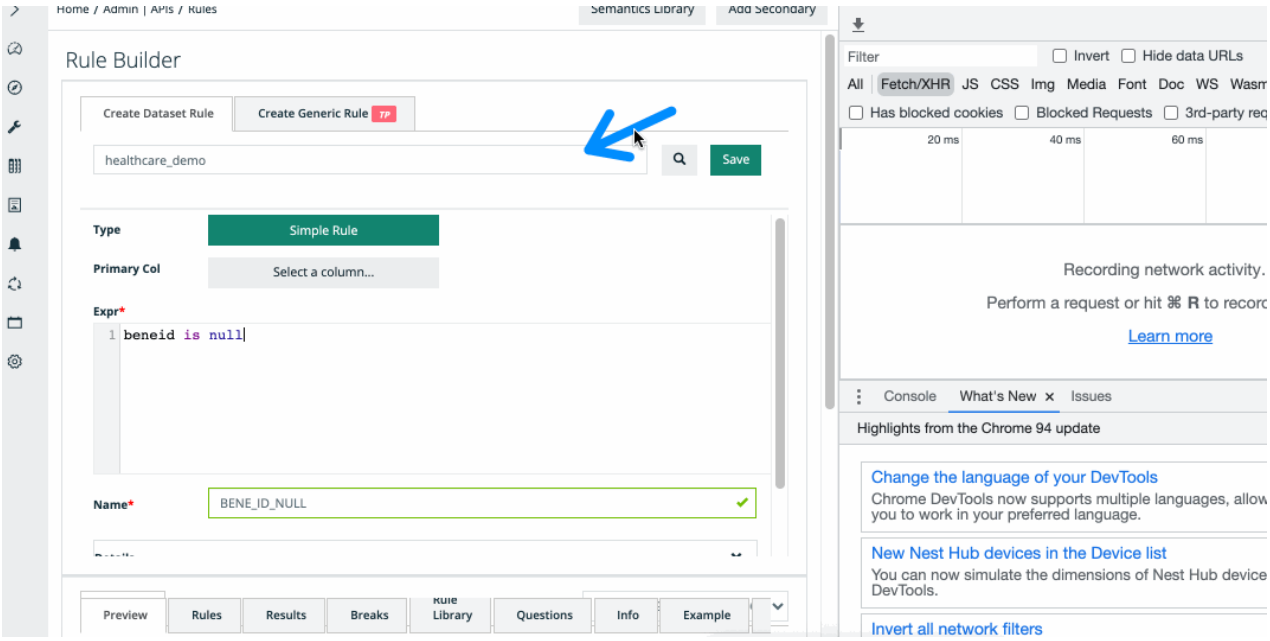
Find the endpoint

Find an API Endpoint

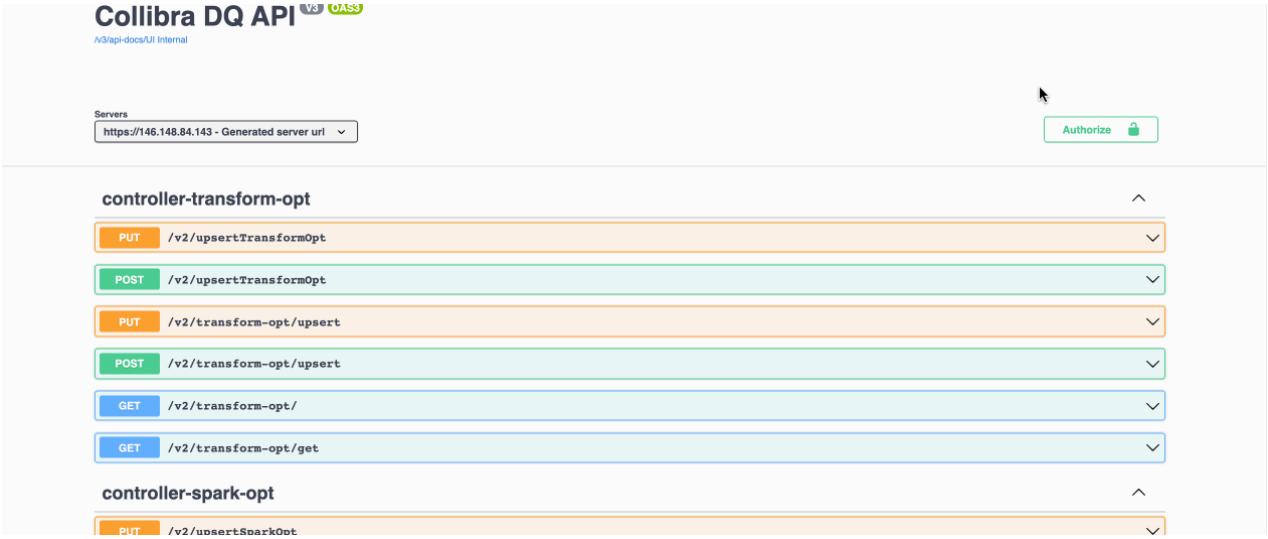
Any front-end action uses the API. You can find the corresponding endpoint using developer tools.

In this example, we will look at the api call for `/create rule`.

Locate the call in Developer Tools



Locate the API in Swagger



All UI endpoints are the API and can be located in swagger. You can script against this externally as well.

Export and Import API

Promoting and moving data sets across environments.

Pre-requirements

> Warning The database needs the stored procedure (function) defined in order to use the Export/Import API.

V2 - Stable - available from 2022.02 release

Step 1a - Export content from source schema

GET https://<collibra-dq-url>/v2/**db-export**

Exports tables from database

Parameters

Query

`data sets*`	List of strings	List of data sets to export. You need to give at least one **valid** data set name into the list.
`schema`	String	Name of the schema/tenant where you want to perform the export. \ *Default value:* **public**
`tables`	String	List of tables to export on the given schema & data set(s). \ If you leave it empty, the following tables will be exported altogether: **rule_repo, owl_catalog, owl_rule, alert_cond, owl_check_repo, job_schedule, alert_output**
Responses		
● **200: OK**		List of SQL - INSERT statements as JSON list.

```
{
    // Response
}
```

● **400: Bad Request** Any error happened with error message

```
{
    // Response
}
```

Step 1b - Import content

POST https://<collibra-dq-url>/v2/**db-import**

Import content into the target tenant

The target schema/tenant name will be part of the input SQL INSERT statements.

The import is rung on non-transactional mode, any error happens in the middle, the saved items will be left in the database as is.

Parameters

Body

*	List of SQL INSERT, which will be imported into the target Collibra DQ metastore. Format: JSON string list
Responses	
200: OK	When the import was successful.

```
{
    // Response
}
```

400: Bad Request Any error happened with error message

```
{
    // Response
}
```

We suggest using db-export, but we will not remove get-exports. We do expect to consolidate the newer logic behind the method.

Step 1c - Get-Exports

You can pass in several dataset names and several tables at once. This endpoint will create a JSON payload

> Note Exports and Imports are currently limited to the 3 tables listed below.

These are the three most common tables. These are the supported tables for re-promotion (running the export multiple times). The most common use case is to copy jobs and rules from environment A to environment B. Running the export/import

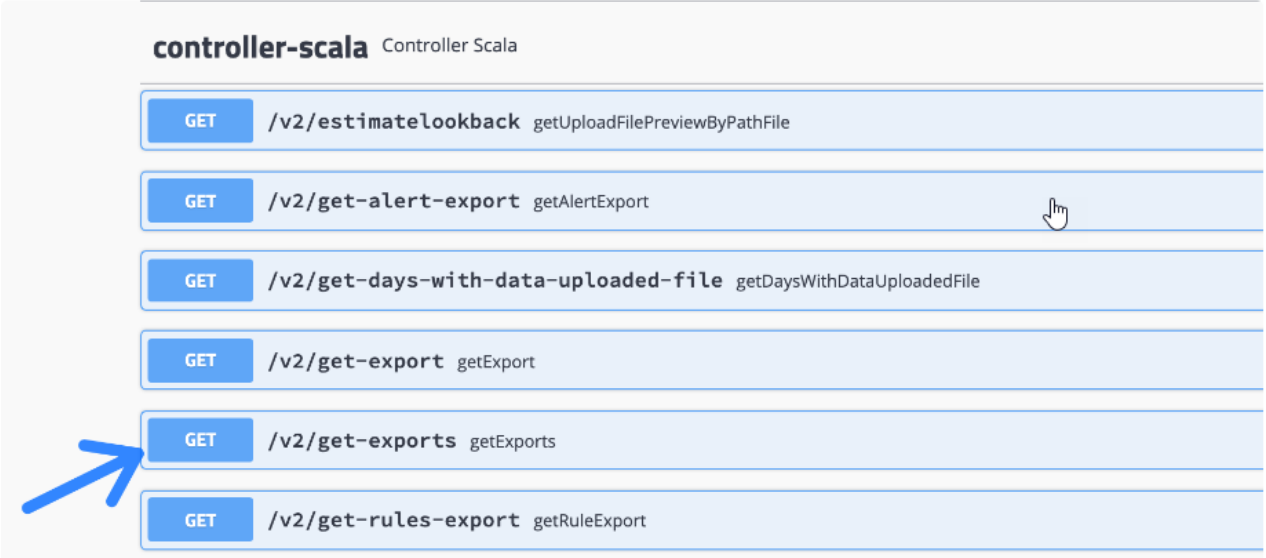
sequence on the same environment likely result in a key constraint conflict, unless in-between edits are made to the insert payload.

- owl_rule
- job_schedule
- owl_check_repo

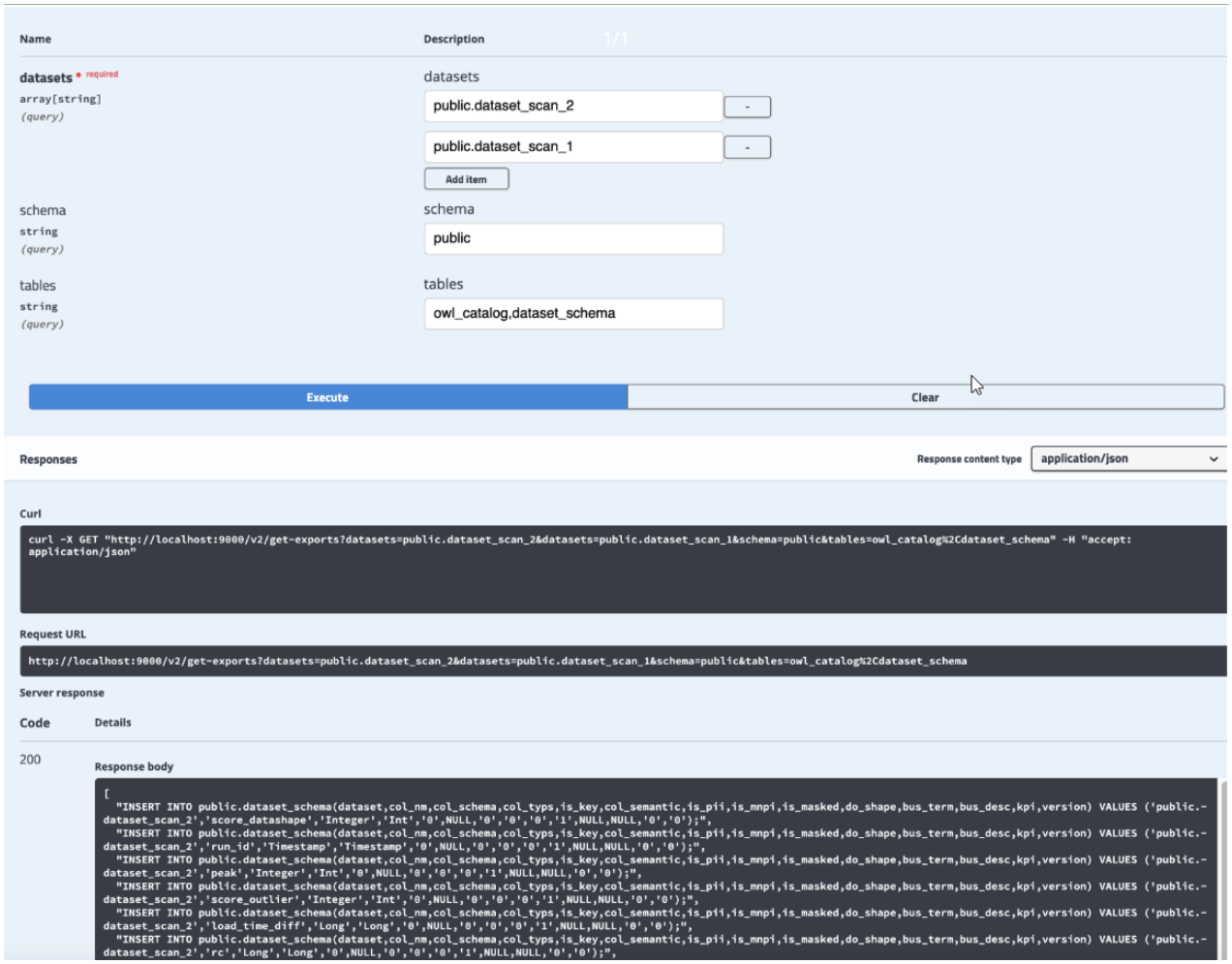
```
http://<url>/v2/get-exports?dataset=public.dataset_scan_2,pub-
lic.dataset_scan_1&schema=public&tables=owl_rule,job_sched-
ule,owl_check_repo
```

Use Swagger to build this for you

This is located under controller-scala (internal API)



Click Try it out to input the details



Step 2 - Run-Import

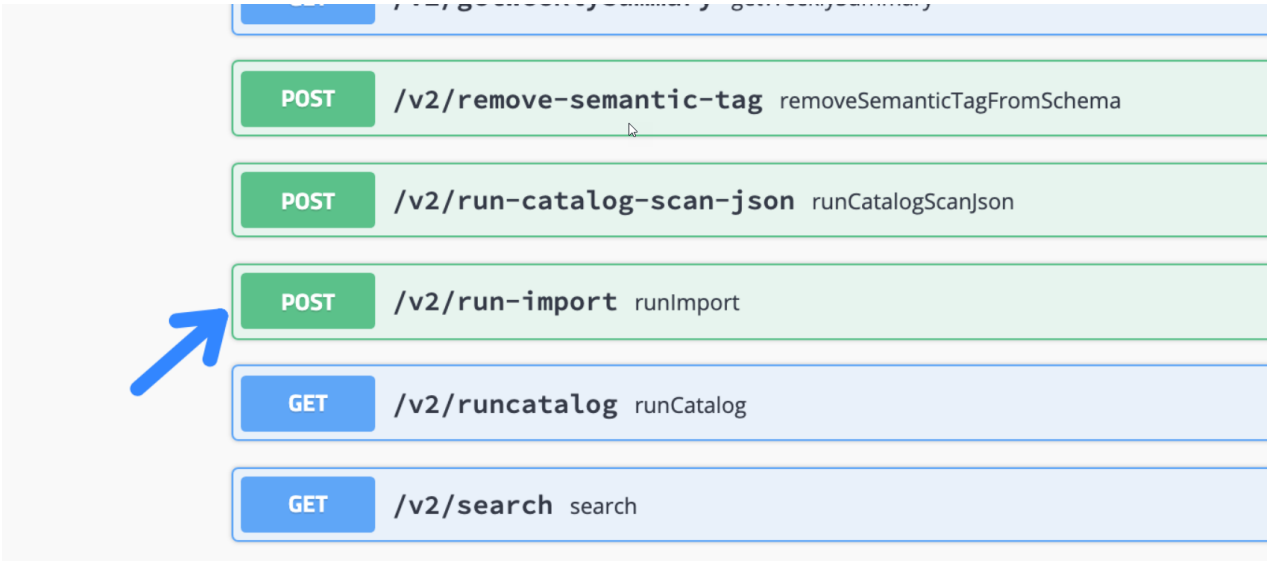
> Note You will want to perform a find/replace on the import payload to check for differences in connections, agents, spark and environment configurations. Migrating to different environments typically requires the payload to be modified.

Run import on the desired environment, passing the output of the previous statement to the body of the request.

```
http://<url>/v2/run-import
```

Use Swagger to try it out

This is under controller-catalog.



This would be the body of the POST.

```
JSON print
2   "INSERT INTO public.owl_catalog(dataset,alias,host,source,db_nm,table_nm
      ,mon,tue,wed,thu,fri,sat,sun,time_zone,t1,t2,t3,t4,semantic,runs
      ,source_name,run_mode,data_concept_id) VALUES ('public.dataset_scan_2'
      ,'public.dataset_scan_2','localhost','Postgres','public'
      ,'dataset_scan','1','1','1','1','1','1','1','UTC',NULL,NULL,NULL,NULL
      ,'1','4',NULL,'draft',NULL);",
3   "INSERT INTO public.dataset_scan(dataset,run_id,rc,pass_fail,peak,score
      ,score_rule,score_datashape,score_outlier,score_pattern,score_source
      ,score_behavior,score_dupe,score_record,score_schema,load_time_diff
      ,updt_ts,user_label,rc_src,pass_fail_rc,push_down_options) VALUES
      ('public.dataset_scan_2','2021-04-25 00:00:00-04','11','1','1','100'
      ,'0','0','0','0','0','0','0','0','0',NULL,'2021-04-25 21:03:20.024384'
      ,NULL,NULL,'1',NULL);",
4   "INSERT INTO public.owl_check_repo(dataset,cmd_line_t,cmd_line_v,rd,q,f
      ,v0,v1) VALUES ('public.dataset_scan_2','-bhtimeoff -owluser admin
      -lib \"/opt/owl/drivers/postgres/\" -h localhost:5432/postgres
      -drivermemory 2g -master local[*] -ds public.dataset_scan_2
      -deploymode client -q \"select * from public.dataset_scan where
      updt_ts >= ''${rd}'' AND updt_ts < ''${rdEnd}''\" -bhlb 10 -rd ${rd}
      -driver \"org.postgresql.Driver\" -bhminoff -loglevel INFO -cxn
      localhost_owl -bhmaxoff -rdEnd ${rdEnd} ',NULL,'2021-04-25',NULL,NULL
      ,'~~~~',NULL);"
5
```

Requirement - Stored Procedure

The following function needs to be created in the Collibra DQ metastore, before this can run.

```
CREATE OR REPLACE FUNCTION public.dump(p_schema text, p_table
text, p_where text)
 RETURNS SETOF text
 LANGUAGE plpgsql
AS $function$
 DECLARE
     dumpquery_0 text;
     dumpquery_1 text;
     selquery text;
     selvalue text;
     valrec record;
     colrec record;
 BEGIN

     -- ------ --
     -- GLOBAL --
     --   build base INSERT
```

```
    --   build SELECT array[ ... ]
    dumpquery_0 := 'INSERT INTO ' ||  quote_ident(p_schema) ||
'.' || quote_ident(p_table) || '(';
    selquery    := 'SELECT array[';

    <<label0>>
    FOR colrec IN SELECT table_schema, table_name, column_
name, data_type
                  FROM information_schema.columns
                  WHERE table_name = p_table and table_schema
= p_schema
                  ORDER BY ordinal_position
    LOOP
        dumpquery_0 := dumpquery_0 || quote_ident(colrec.-
column_name) || ',';
        selquery    := selquery    || 'CAST(' || quote_ident
(colrec.column_name) || ' AS TEXT),';
    END LOOP label0;

    dumpquery_0 := substring(dumpquery_0 ,1,length(dumpquery_
0)-1) || ')';
    dumpquery_0 := dumpquery_0 || ' VALUES (';
    selquery    := substring(selquery    ,1,length(selquery)-
1)    || '] AS MYARRAY';
    selquery    := selquery    || ' FROM ' ||quote_ident(p_
schema)||'.'||quote_ident(p_table);
    selquery    := selquery    || ' WHERE '||p_where;
    -- GLOBAL --
    -- ------ --

    -- ----------- --
    -- SELECT LOOP --
    --   execute SELECT built and loop on each row
    <<label1>>
    FOR valrec IN  EXECUTE  selquery
    LOOP
        dumpquery_1 := '';
        IF not found THEN
            EXIT ;
        END IF;

        -- ----------- --
        -- LOOP ARRAY (EACH FIELDS) --
        <<label2>>
        FOREACH selvalue in ARRAY valrec.MYARRAY
        LOOP
            IF selvalue IS NULL
            THEN selvalue := 'NULL';
            ELSE selvalue := quote_literal(selvalue);
            END IF;
```

```
            dumpquery_1 := dumpquery_1 || selvalue || ',';
        END LOOP label2;
        dumpquery_1 := substring(dumpquery_1 ,1,length
(dumpquery_1)-1) || ');';
        -- LOOP ARRAY (EACH FIELD) --
        -- ----------- --

        -- debug: RETURN NEXT dumpquery_0 || dumpquery_1 || '
--' || selquery;
        -- debug: RETURN NEXT selquery;
        RETURN NEXT dumpquery_0 || dumpquery_1;

    END LOOP label1 ;
    -- SELECT LOOP --
    -- ----------- --

 RETURN ;
 END
 $function$
;
```

This assignment needs to be added.

```
alter function dump(text, text, text) owner to <ownername>;
```

Export and Import Example

> Note Best practice is to use get-exports and the owl_rule table post 2021.09 release. Please refer to the Export and Import API page for more details.

Steps

1. Find your dataset.
2. Pass your table to the following api call - http://<url>/v2/get-rules-export?data-set=public.transit_6.
3. Run import on the desired environment, passing the output of the previous statement to the body of the request - http://<url>/v2/run-import.

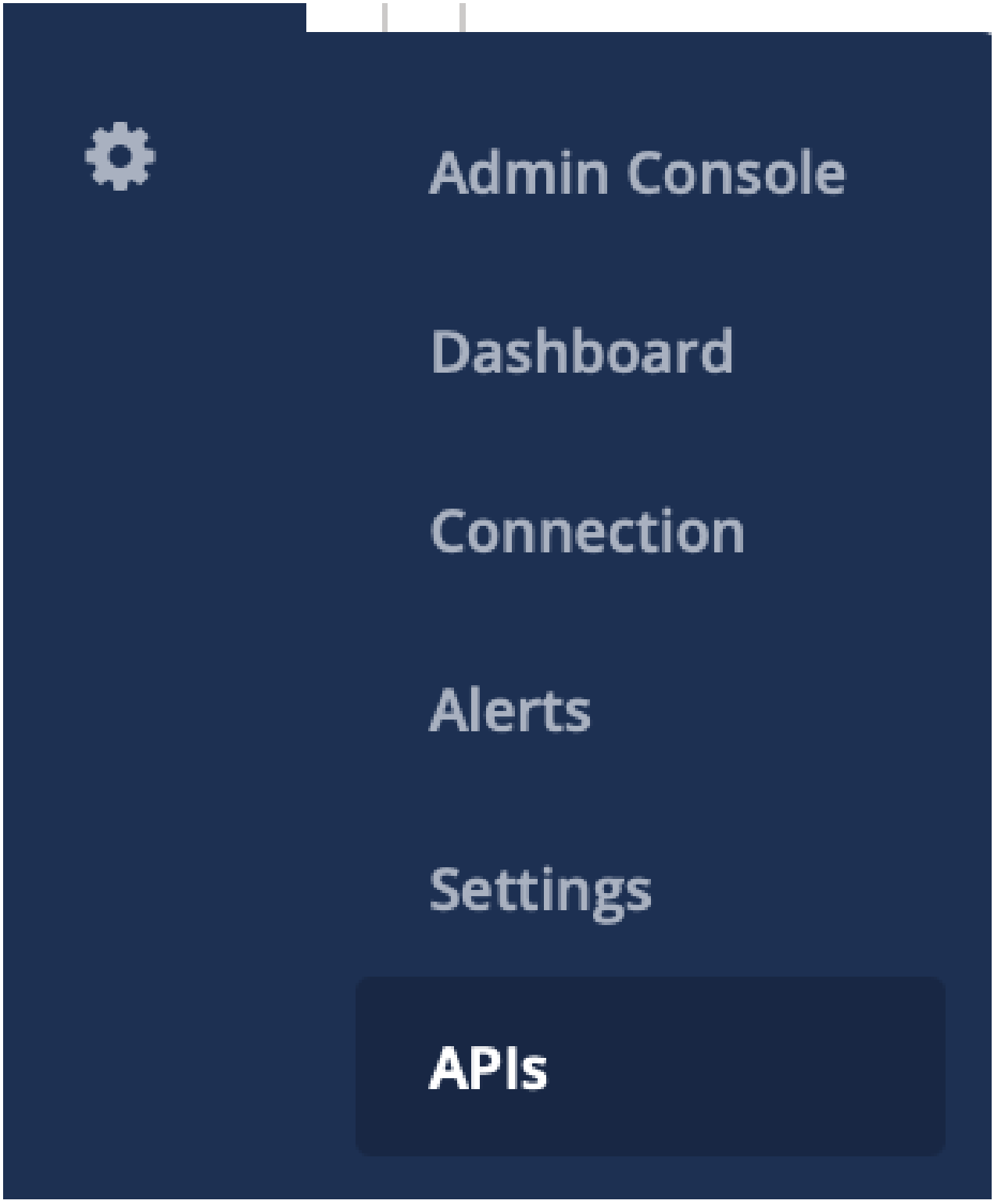
The following function needs to be declared in the postgres metastore before this can run.

```
CREATE OR REPLACE FUNCTION public.dump(p_schema text, p_table
text, p_where text)
 RETURNS SETOF text
 LANGUAGE plpgsql
AS $function$
 DECLARE
     dumpquery_0 text;
     dumpquery_1 text;
     selquery text;
     selvalue text;
     valrec record;
     colrec record;
 BEGIN

     -- ------ --
     -- GLOBAL --
     --   build base INSERT
     --   build SELECT array[ ... ]
     dumpquery_0 := 'INSERT INTO ' ||  quote_ident(p_schema) ||
'.' || quote_ident(p_table) || '(';
     selquery    := 'SELECT array[';

     <<label0>>
     FOR colrec IN SELECT table_schema, table_name, column_name,
data_type
                   FROM information_schema.columns
                   WHERE table_name = p_table and table_schema =
p_schema
                   ORDER BY ordinal_position
     LOOP
         dumpquery_0 := dumpquery_0 || quote_ident(colrec.-
column_name) || ',';
         selquery    := selquery    || 'CAST(' || quote_ident
(colrec.column_name) || ' AS TEXT),';
     END LOOP label0;

     dumpquery_0 := substring(dumpquery_0 ,1,length(dumpquery_
0)-1) || ')';
     dumpquery_0 := dumpquery_0 || ' VALUES (';
     selquery    := substring(selquery    ,1,length(selquery)-
1)    || '] AS MYARRAY';
     selquery    := selquery    || ' FROM ' ||quote_ident(p_
schema)||'.'||quote_ident(p_table);
     selquery    := selquery    || ' WHERE '||p_where;
     -- GLOBAL --
     -- ------ --

     -- ---------- --
```

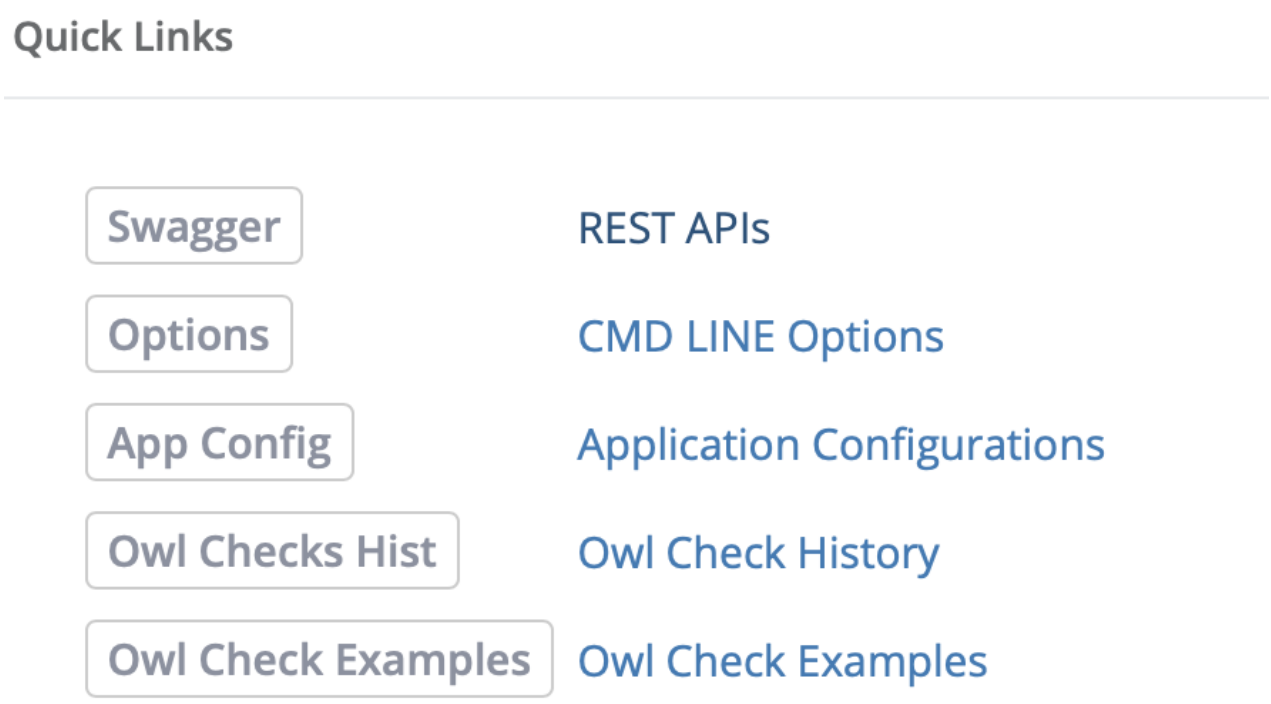
```
      -- SELECT LOOP --
      --   execute SELECT built and loop on each row
      <<label1>>
      FOR valrec IN  EXECUTE  selquery
      LOOP
          dumpquery_1 := '';
          IF not found THEN
              EXIT ;
          END IF;

          -- ----------- --
          -- LOOP ARRAY (EACH FIELDS) --
          <<label2>>
          FOREACH selvalue in ARRAY valrec.MYARRAY
          LOOP
              IF selvalue IS NULL
              THEN selvalue := 'NULL';
              ELSE selvalue := quote_literal(selvalue);
              END IF;
              dumpquery_1 := dumpquery_1 || selvalue || ',';
          END LOOP label2;
          dumpquery_1 := substring(dumpquery_1 ,1,length
(dumpquery_1)-1) || ');';
          -- LOOP ARRAY (EACH FIELD) --
          -- ----------- --

          -- debug: RETURN NEXT dumpquery_0 || dumpquery_1 || ' -
-' || selquery;
          -- debug: RETURN NEXT selquery;
          RETURN NEXT dumpquery_0 || dumpquery_1;

      END LOOP label1 ;
      -- SELECT LOOP --
      -- ----------- --

 RETURN ;
 END
 $function$
;
```

From Swagger

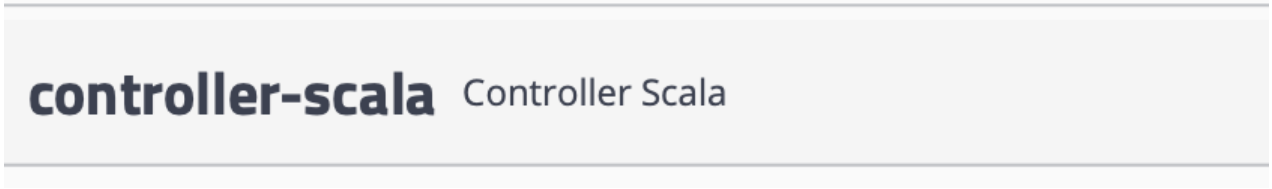
Navigate to the API page.



Find the Rest APIs link.

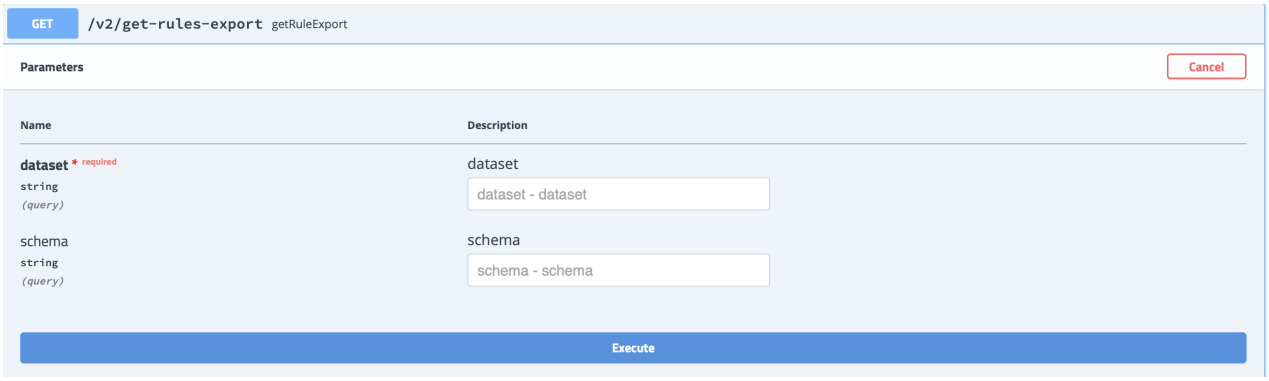


Drill-in to the controller-scala section.

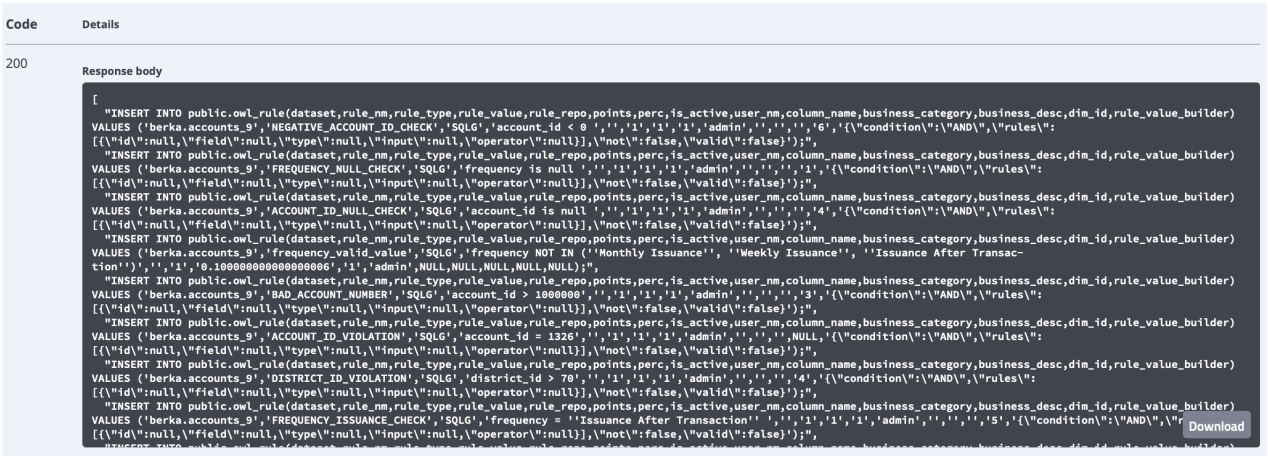


Find the get-rules-export call.

Click Try it out and enter a data set name, Execute to run the call.

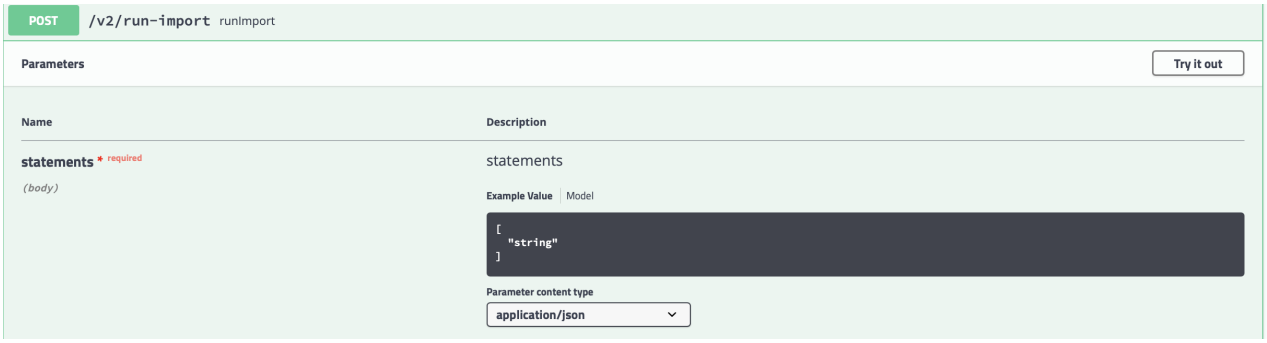


Copy the response body.

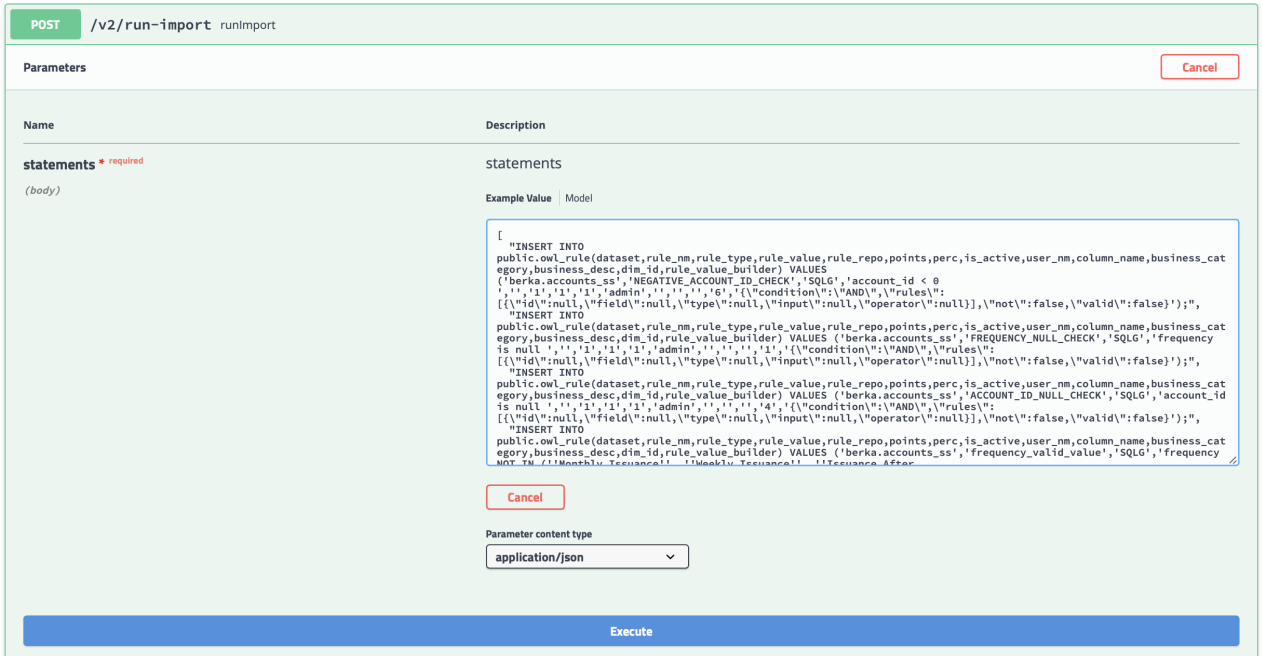


Navigate to the controller-catalog section.

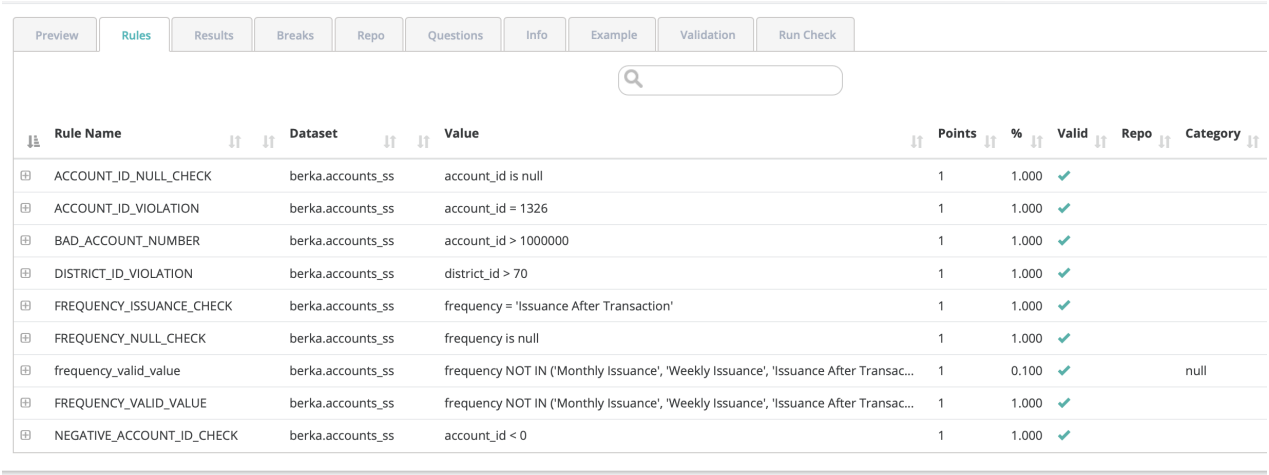
Find run-import and Try it out.



Make any edits and paste in the response body from the previous step.



Visually validate the rules were transferred to another dataset successfully.



Rule Name	Dataset	Value	Points	%	Valid	Repo	Category
ACCOUNT_ID_NULL_CHECK	berka.accounts_ss	account_id is null	1	1.000	✔		
ACCOUNT_ID_VIOLATION	berka.accounts_ss	account_id = 1326	1	1.000	✔		
BAD_ACCOUNT_NUMBER	berka.accounts_ss	account_id > 1000000	1	1.000	✔		
DISTRICT_ID_VIOLATION	berka.accounts_ss	district_id > 70	1	1.000	✔		
FREQUENCY_ISSUANCE_CHECK	berka.accounts_ss	frequency = 'Issuance After Transaction'	1	1.000	✔		
FREQUENCY_NULL_CHECK	berka.accounts_ss	frequency is null	1	1.000	✔		
frequency_valid_value	berka.accounts_ss	frequency NOT IN ('Monthly Issuance', 'Weekly Issuance', 'Issuance After Transac...	1	0.100	✔		null
FREQUENCY_VALID_VALUE	berka.accounts_ss	frequency NOT IN ('Monthly Issuance', 'Weekly Issuance', 'Issuance After Transac...	1	1.000	✔		
NEGATIVE_ACCOUNT_ID_CHECK	berka.accounts_ss	account_id < 0	1	1.000	✔		

Assignment API

Time Zone API

UTC + global DateTime standard

Background on common time issues

Controlling dates and times has always been a troublesome topic for global systems. Server clock vs server code such as new Date() which may create a date in the local timezone of the server vs the Browser or clients timezone. Moving to the cloud only makes the problem worse when you need to consider the timezone the server might be in and inherit from it's system clock.

Collibra Data Quality's Solution - Keep it Simple

If everyone worked off of a globally understood format that is not subject to misinterpretation things would be more simple. Take 03/02/2019, for example. Is this March 2nd or February 3rd? That depends on which country you live in. Collibra DQ only accepts this format: YYYY-MM-DD. Extending this to time would mean YYYY-MM-DD 00:00:00.

CmdLine Examples

./owlcheck -ds trades -rd "2019-04-01" or -rd "2019-04-01 00:00:00"

Simple Example

A user running an DQ Check in New York and a user running an DQ Check three hours later in California.

CmdLine Arg	User Location	TimeZone	Stored in Owl (UTC)
-rd 2019-04-01	New York	implied EST	2019-04-01 04:00:00
-rd 2019-04-01	California	implied PST	2019-04-01 07:00:00

These jobs run three hours apart, even though they appear to run first thing in the morning to each user. Collibra DQ stores all dates in a common **UTC** format for global consistency.

On the Jobs page, the Start Time and Update Time columns are always based on the server time zone of the DQ Web App, and appear in the YYYY-MM-DD 00:00:00 format.

Web API or URL Example

http://localhost:9000/dq/hoot?dataset=atmCustomers&runId=2019-04-01 04:00:00
http://localhost:9000/dq/hoot?dataset=atmCustomers&runId=2019-04-01 07:00:00

For Convenience, if a user prefers seeing and interacting with dates in their local time zone:

http://localhost:9000/dq/hoot?dataset=atmCustomers&runId=2019-04-01 00:00:00&tz=EST http://localhost:9000/dq/hoot?dataset=atmCustomers&runId=2019-04-01 00:00:00&tz=PST

Cookie

Use cookies file to run Collibra DQ CURL commands.

```
curl -i -X POST -d username=<username> -d password=<password>
http://localhost:9000/login -c cookies.txt

curl -i --header "Accept:application/json" -X GET -b cookies.txt
"http://localhost:9000/v2/getsecuritymap"
```

Multi-Tenant without subdomain in URL (tenant parameter required):

```
curl -i -X POST -d username=<username> -d password=<password> -d
tenant=<tenant> -d tenant=public http://localhost:9000/login -c
cookies.txt

curl -i --header "Accept:application/json" -X GET -b cookies.txt
"http://localhost:9000/v2/getsecuritymap"
```

JWT

Use JSON web tokens to run Collibra DQ CURL commands.

```
TOKEN=$(curl -s -X POST http://localhost:9000/auth/signin -H
"Content-Type:application/json" -d "{\"user-
name\":\"<username>\", \"password\":\"<password>\"}" | jq -r
'.token')

curl -i -H 'Accept: application/json' -H "Authorization: Bearer
${TOKEN}" http://localhost:9000/v2/getsecuritymap
```

Multi-Tenant without subdomain in URL (tenant parameter [iss] required):

```
TOKEN=$(curl -s -X POST http://localhost:9000/auth/signin -H
"Content-Type:application/json" -d "{\"user-
name\":\"<username>\", \"password\":\"<password>\", \"is-
s\":\"<tenant>\"}"| jq -r '.token')

curl -i -H 'Accept: application/json' -H "Authorization: Bearer
${TOKEN}" http://localhost:9000/v2/getsecuritymap
```

Without Headers and jq display:

```
curl -H 'Accept: application/json' -H "Authorization: Bearer
${TOKEN}" http://localhost:9000/v2/getsecuritymap | jq '.' | cat
```

Livy

What is Livy?

At its core, Apache Livy is an optional component that changes how the **Collibra DQ Web** app caches remote files. From **Explorer**, Livy allows for interaction with Spark clusters over REST APIs. This is especially useful with larger data or Spark clusters, because with Livy, you can drill into those bigger files.

In a future release, we plan to make Livy a standard component as part of our data visualization, but as of 2022.04 it remains optional. An orange icon shows cached remote files.

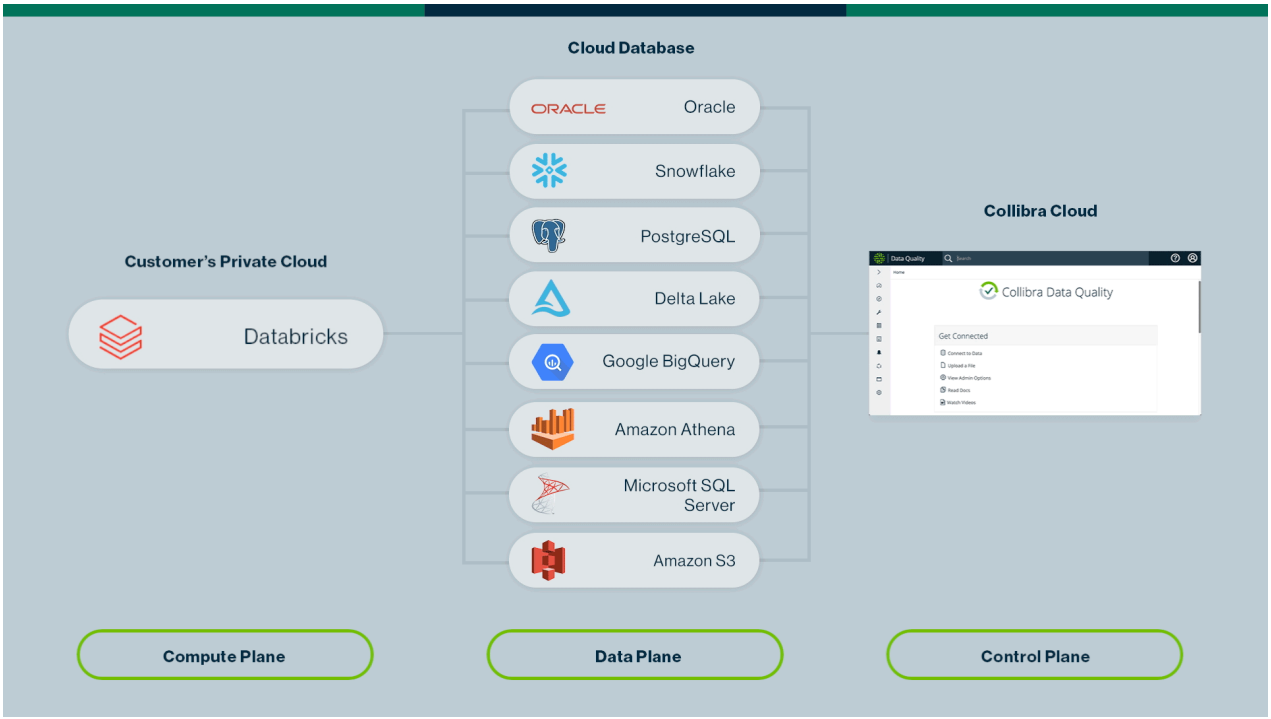
Notebook

Collibra DQ + Databricks

Introduction

This page provides guidance to help you upload Collibra DQ jars to a Databricks cluster and run a Collibra DQ job by invoking Collibra DQ APIs.

Architecture



Collibra DQ Environment Setup

This section explains the steps involved in setting up your Collibra DQ environment in Databricks. This is the first step towards invoking Collibra DQ APIs in Databricks.

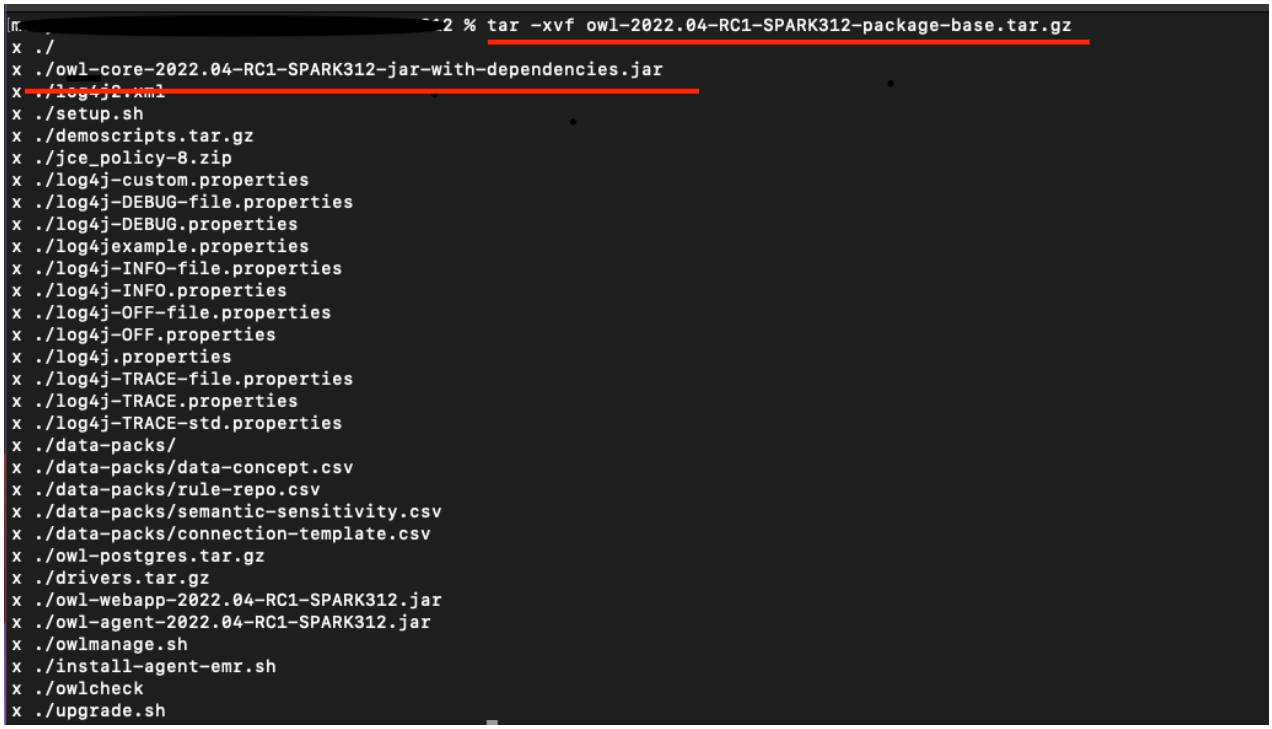
Step 1: Extract the Collibra DQ core jar from owl package zipped file.

The first step is to get the CDQ jar file. Once you have the cdq jar package file, you can get the jars by running the following commands:

`tar -xvf package.tar.gz`

For example, `tar -xvf owl-2022.04-RC1-default-package-base.tar.gz`

Running this command instructs tar to extract the files from the zipped file. From the list of the files, you need to upload the owl-core-xxxx-jar-with-dependancies.jar to our Databricks file system which will be explained in the next section.



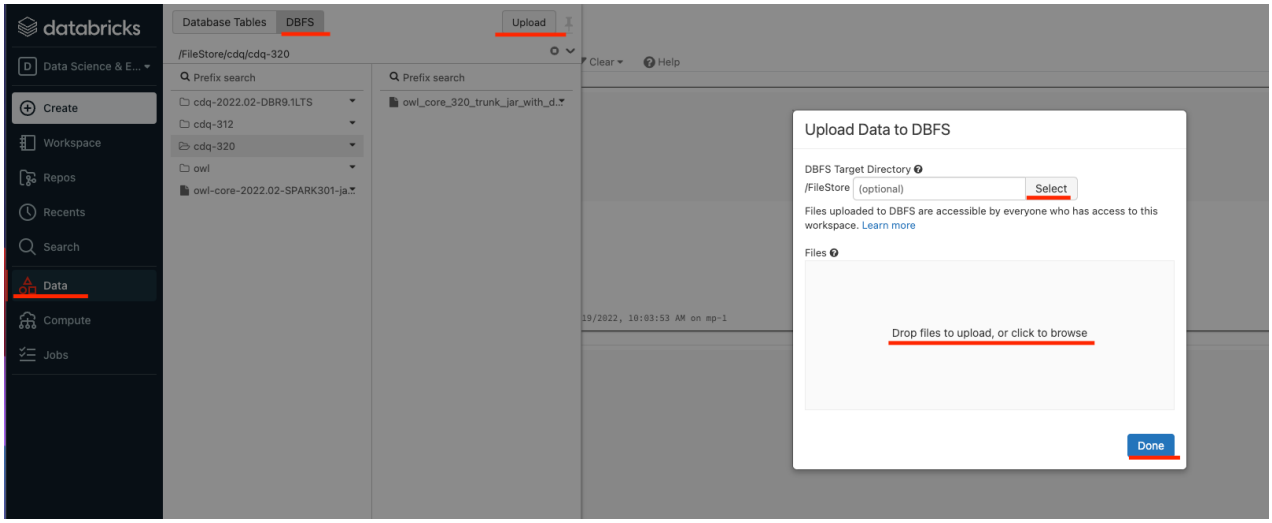
Step 2: Upload the Collibra DQ core jar file to Databricks file system using UI

The jars should be manually uploaded in Databricks file system. Below is the quick summary of the steps. For more information on uploading files in Databricks, refer to the official Databricks documentation.

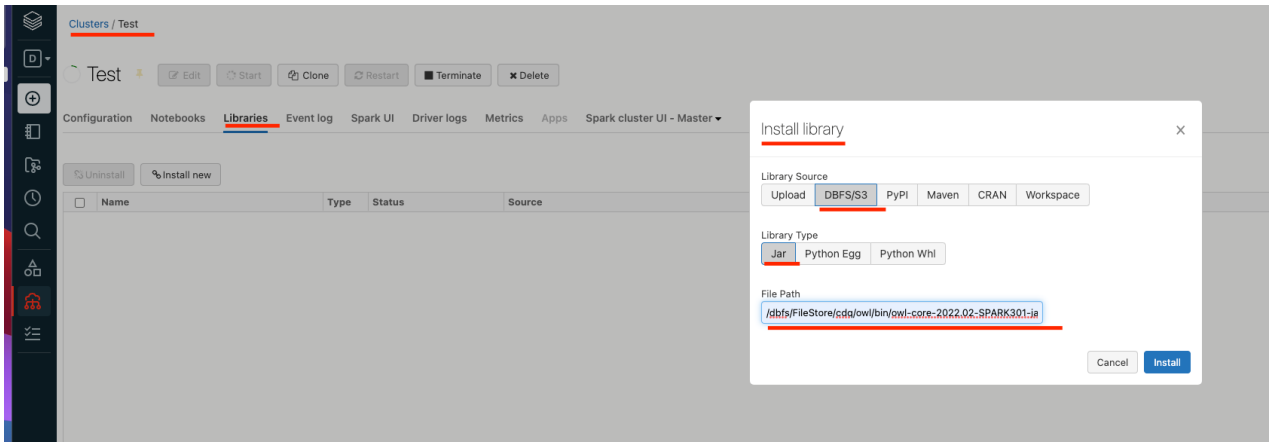
a. Login to your Databricks account.

b. Click **Data** in the sidebar.

c. Click **DBFS** at the top of the page.
d. Upload the owl-core-xxxx-jar-with-dependancies.jar to your desired path.



Step 3: Install Collibra DQ library in your Databricks cluster



Once this step is completed, you can create a workspace and start using Collibra DQ APIs.

Step 4: Update the Spark Config in your Databricks cluster

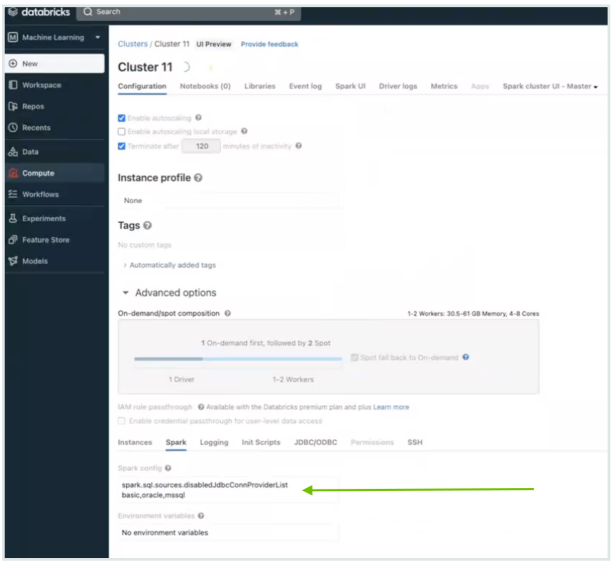
> Note This step is required if your cluster uses Spark 3.2.1 and onward.

When you bring Collibra DQ jars into Databricks, you are required as of the Collibra DQ 2023.01 release to set the property `spark.sql.sources.disabledJdbcConnProviderList='basic,oracle,mssql'` at either the Spark Cluster-level or the SparkSession-level before using Collibra DQ's set of functions for Spark profiles 3.2.1 and onwards.

Setting the property in the Spark Config of your Databricks cluster

If you have an active Databricks cluster, you can set the property in the Spark Config section of your Databricks cluster.

1. On your Databricks cluster configuration page, click the **Advanced Options** toggle and select the **Spark** tab.
2. In the **Spark Config** section, enter the following configuration property as one key-value pair per line:
   `spark.sql.sources.disabledJdbcConnProviderList basic,oracle,mssql`



Setting the property in a SparkSession

If you have your own Spark data source, you can set the property in the SparkSession. For example:
`SparkSession.getActiveSession.get.conf.set`

```
("spark.sql.sources.disabledJdbcConnProviderList",
"basic,oracle,mssql")
```

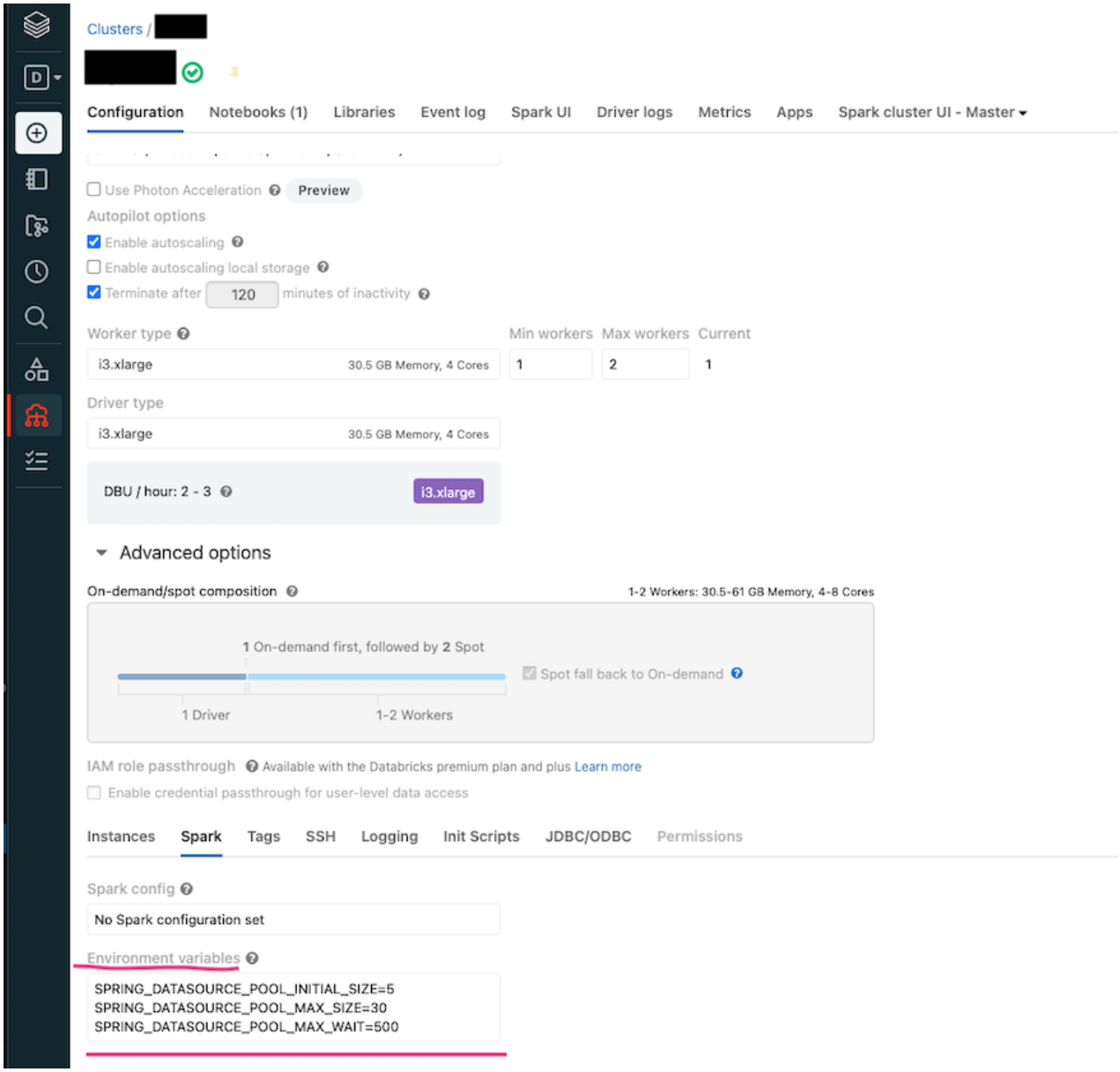
Step 5 (Optional): Update datasource pool size

> Note This step is necessary if you get *PoolExhaustedException* when you call Collibra DQ APIs.

Update the connection pool size in the Spark environment using the following environment variables:

```
SPRING_DATASOURCE_POOL_MAX_WAIT=500

SPRING_DATASOURCE_POOL_MAX_SIZE=30

SPRING_DATASOURCE_POOL_INITIAL_SIZE=5
```

For more information on setting up Databricks environment variables, refer to the official Databricks documentation.


Clusters / [redacted]
[redacted]
Configuration
Notebooks (1)
Libraries
Event log
Spark UI
Driver logs
Metrics
Apps
Spark cluster UI - Master
Use Photon Acceleration
Preview
Autopilot options
Enable autoscaling
Enable autoscaling local storage
Terminate after
120
minutes of inactivity
Worker type
Min workers
Max workers
Current
i3.xlarge
30.5 GB Memory, 4 Cores
1
2
1
Driver type
i3.xlarge
30.5 GB Memory, 4 Cores
DBU / hour: 2 - 3
i3.xlarge
Advanced options
On-demand/spot composition
1-2 Workers: 30.5-61 GB Memory, 4-8 Cores
1 On-demand first, followed by 2 Spot
Spot fall back to On-demand
1 Driver
1-2 Workers
IAM role passthrough
Available with the Databricks premium plan and plus Learn more
Enable credential passthrough for user-level data access
Instances
Spark
Tags
SSH
Logging
Init Scripts
JDBC/ODBC
Permissions
Spark config
No Spark configuration set
Environment variables
SPRING_DATASOURCE_POOL_INITIAL_SIZE=5
SPRING_DATASOURCE_POOL_MAX_SIZE=30
SPRING_DATASOURCE_POOL_MAX_WAIT=500

Collibra DQ Working Example in DataBricks

Import Collibra DQ library

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.types._
import scala.collection.JavaConverters._
import java.util.Date
import java.time.LocalDate
import java.text.SimpleDateFormat
import spark.implicits._
import java.util.{ArrayList, List, UUID}
// CDQ Imports
import com.owl.core.Owl
import com.owl.common.options._
import com.owl.common.domain2._
import com.owl.core.util.OwlUtils
spark.catalog.clearCache
```

Bringing customer data from another database

Bringing customer data from a file

```
val df = (spark.read
.format("csv").option("header", true).option("delimiter", ","
.load(dbfs:/FileStore/nyse.csv")
)
```

Bringing customer data from a database

```
val connProps = Map(
"driver" -> "org.postgresql.Driver",
"user" -> "your-username",
"password" -> "your-password",
"url" -> "jdbc:postgresql://abc:1234/postgres",
"dbtable" -> "public.example_data")
//--- Load Spark DataFrame ---//
val df = spark.read.format("jdbc").options(connProps).load dis-
play(df)
display(df) // view your data
```

Variables to set up Collibra DQ Metastore database location

```
val pgHost = "xxxx.amazonaws.com"
val pgDatabase = "postgres"
val pgSchema = "public"
val pgUser = "???????"
val pgPass = "????"
val pgPort = "0000"
```

Create a Collibra DQ Test (Rules) and detect breaks

> Note If the rules are already created and assigned to a dataset from the UI, calling owlcheck() automatically executes all the rules associated with the given dataset and there is no need to recreate the rule from notebook.

```
val dataset = "cdq_notebook_db_rules"
var date = "2018-01-11"

// Options
val opt = new OwlOptions()
opt.dataset = dataset
opt.runId = date
opt.host = pgHost
opt.port = pgPort
opt.pgUser = pgUser
opt.pgPassword = pgPass

opt.setDatasetSafeOff(false) // to enable historical overwrite
of dataset

// Create a simple rule and assign it to dataset
val simpleRule = OwlUtils.createRule(opt.dataset)
      simpleRule.setRuleNm("nyse-stocks-symbol")
      simpleRule.setRuleValue("symbol == 'BHK'")
      simpleRule.setRuleType("SQLG")
      simpleRule.setPerc(1.0)
      simpleRule.setPoints(1)
      simpleRule.setIsActive(1)
      simpleRule.setUserNm("admin")
      simpleRule.setPreviewLimit(8)

// Create a rule from generic rules that are created from UI:
val genericRule = OwlUtils.createRule(opt.dataset)
    genericRule.setRuleNm("exchangeRule") // this could be any
name
    genericRule.setRuleType("CUSTOM")
    genericRule.setPoints(1)
    genericRule.setIsActive(1)
    genericRule.setUserNm("admin")
    genericRule.setRuleRepo("exchangeCheckRule"); // Validate
the generic rule name //from UI
    genericRule.setRuleValue("EXCH") // COLUMN associate with
the rule


// Pre Routine
val cdq = com.owl.core.util.OwlUtils.OwlContext(df, opt)
cdq.removeAllRules(opt.dataset)
.register(opt)
.addRule(simpleRule)

// Scan
cdq.owlCheck()
```

```
val results = cdq.hoot() // returns object Hoot, not a DataFrame
//See Json Results(Option for downstream processing)
println("--------------Results:----------------\n")
println(results) //optional

//Post Routine, See DataFrame Results (Option for downstream pro-
cessing)
val breaks = cdq.getRuleBreakRows("nyse-stocks-symbol")
println("--------------Breaks:----------------\n")
display(breaks)

// Different Options for handling bad records
val badRecords = breaks.drop("_dataset","_run_id", "_rule_name",
"owl_id")
display(badRecords)

val goodRecords = df.except(badRecords)
display(goodRecords)
```

Write the breaks (bad records) DataFrame to a Parquet file

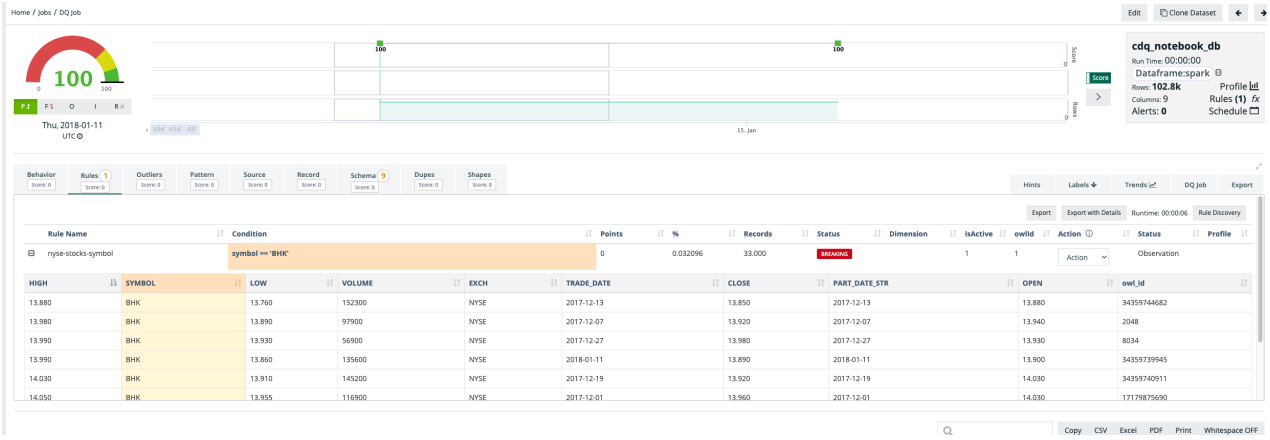
```
// Remove the file if it exists
dbutils.fs.rm("/tmp/databricks-df-example.parquet", true)
breaks.write.parquet("/tmp/databricks-df-example.parquet")
```

The following image shows the code snippet and the result in Databricks:

```
opt.pgUser = pgUser
opt.pgPassword = pgPass

// Create a template rule
val templateRule = OwlUtils.createRule(opt.dataset)
      templateRule.setRuleNm("rule1_nyse-stocks-symbol")
      templateRule.setRuleValue("symbol == 'BHK'") /* if the symbol == BHK break the rule */
      templateRule.setRuleType("SQLG")
      templateRule.setPoints(5)  // Whole int 1 and greater
      templateRule.setIsActive(1)
      templateRule.setUserNm("admin")
      templateRule.setPreviewLimit(8)

// Create a second template rule
val templateRule1 = OwlUtils.createRule(opt.dataset)
      templateRule1.setRuleNm("rule2_nyse-stocks-symbol")
      templateRule1.setRuleValue("symbol == 'MSP'") /* if the symbol == MSP break the rule: */
      templateRule1.setRuleType("SQLG")
      templateRule1.setPoints(5)  // Whole int 1 and greater
      templateRule1.setIsActive(1)
      templateRule1.setUserNm("admin")
      templateRule1.setPreviewLimit(8)

// Scan
val cdq = com.owl.core.util.OwlUtils.OwlContext(df, opt)
.register(opt)
.addRule(templateRule).addRule(templateRule1)
cdq.owlCheck()

// create another dataset and re-use the previous rules from the other dataset
val dataset2 ="nyse-dataset"
var opt1 = new OwlOptions()
opt1.dataset = dataset2
opt1.runId = date
opt1.host = pgHost
opt1.port = pgPort
opt1.pgUser = pgUser
opt1.pgPassword = pgPass
opt1.setDatasetSafeOff(false)

val cdq1 = OwlUtils.OwlContext(df, opt1)
cdq1.register(opt1)
// get the rules from "master_template_rules_dataset"
val rules = cdq1.getRulesByDataset(dataset)
if (rules.size > 0)
  cdq1.removeAllRules(dataset2).addRules(rules, dataset2)
cdq1.owlCheck()
val breaks = cdq1.getRuleBreakRows()

//display(breaks)

breaks.show()
if(breaks.count() > 0 ) {
  //val badRecords = breaks.drop("_dataset", "_runId", "_rulename", "owl_id")
  val badRecords = breaks.drop("_dataset", "_runId", "_rulename", "owl_id")
  display(badRecords)
  val goodRecords = df.except(badRecords)
  display(goodRecords)
 }
```

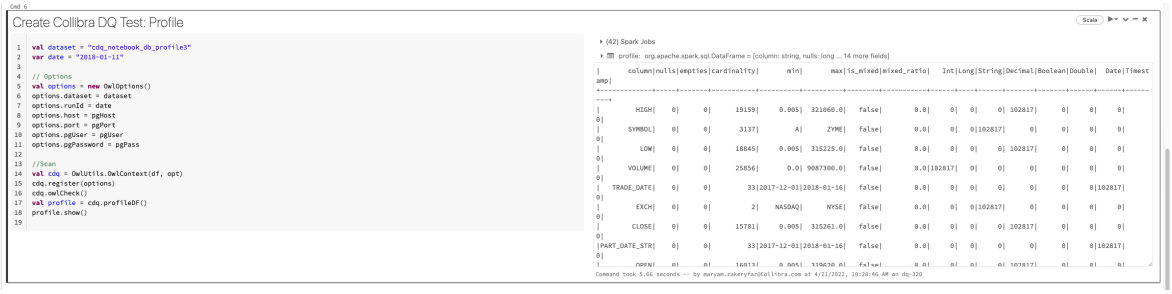
Steps to reassign the rules of one dataset to another via the API.

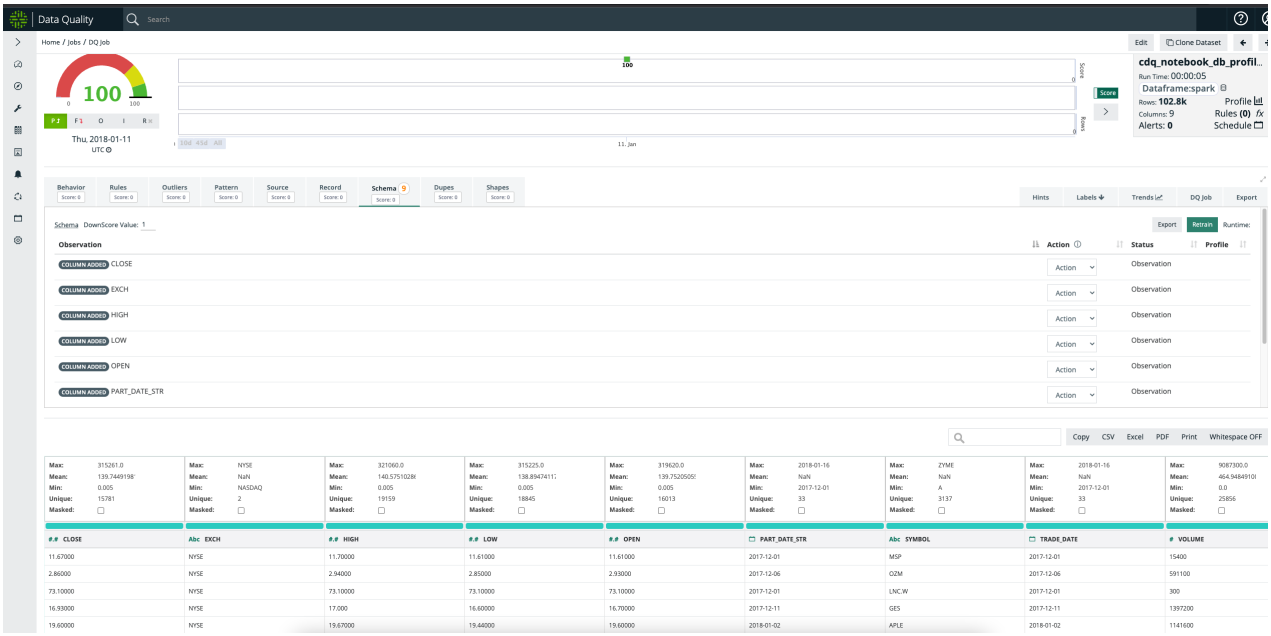
The breaks and the rules can also be viewed in Collibra DQ web.



Create a Collibra DQ Test (Profile)

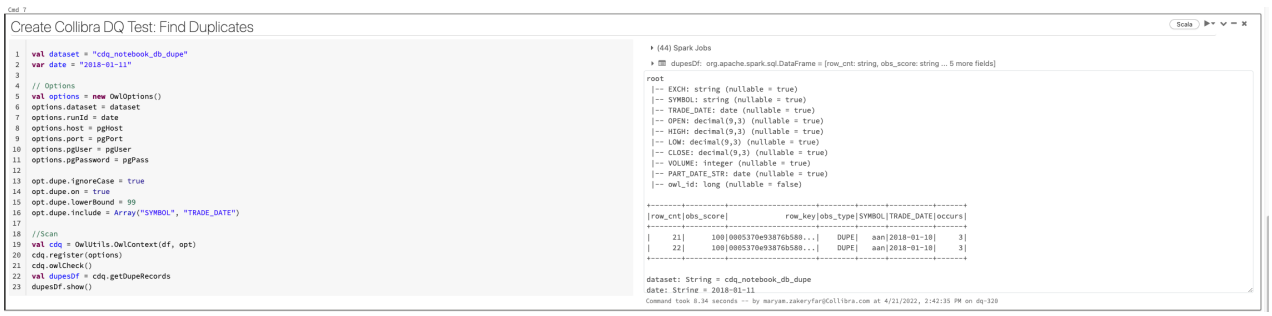
```
val dataset = "cdq_notebook_nyse_profile"

val runList = List("2018-01-01", "2018-01-02", "2018-01-03",
"2018-01-04", "2018-01-05"
for(runID <- runList {
// Options
val options = new OwlOptions()
options.dataset = dataset
options.host = pgHost
options.port = pgPort
options.pgUser = pgUser
options.pgPassword = pgPass

//Scan
val profileOpt = new ProfileOpt
profileOpt.on = true
profileOpt.setShape(true)
profileOpt.setShapeSensitivity(5.0)
profileOpt.setShapeMaxPerCol(10)
profileOpt.setShapeMaxColSize(10)
profileOpt.setShapeGranular(true)
profileOpt.behaviorEmptyCheck = true
profileOpt.behaviorMaxValueCheck = true
profileOpt.behaviorMinValueCheck = true
profileOpt.behaviorNullCheck = true
profileOpt.behaviorRowCheck = true
profileOpt.behaviorMeanValueCheck = true
profileOpt.behaviorUniqueCheck = true
profileOpt.behaviorMinSupport = 5 // default is 4
profileOpt.behaviorLookback = 5
options.profile = profileOpt

var date = runId
var df_1 = df.where($"TRADE_DATE"===s"$date")

//Scan
val cdq = OwlUtils.OwlContext(df_1, options)
cdq.register(opt)
cdq.owlCheck()
val profile = cdq.profileDF()
profile.show()
}
```

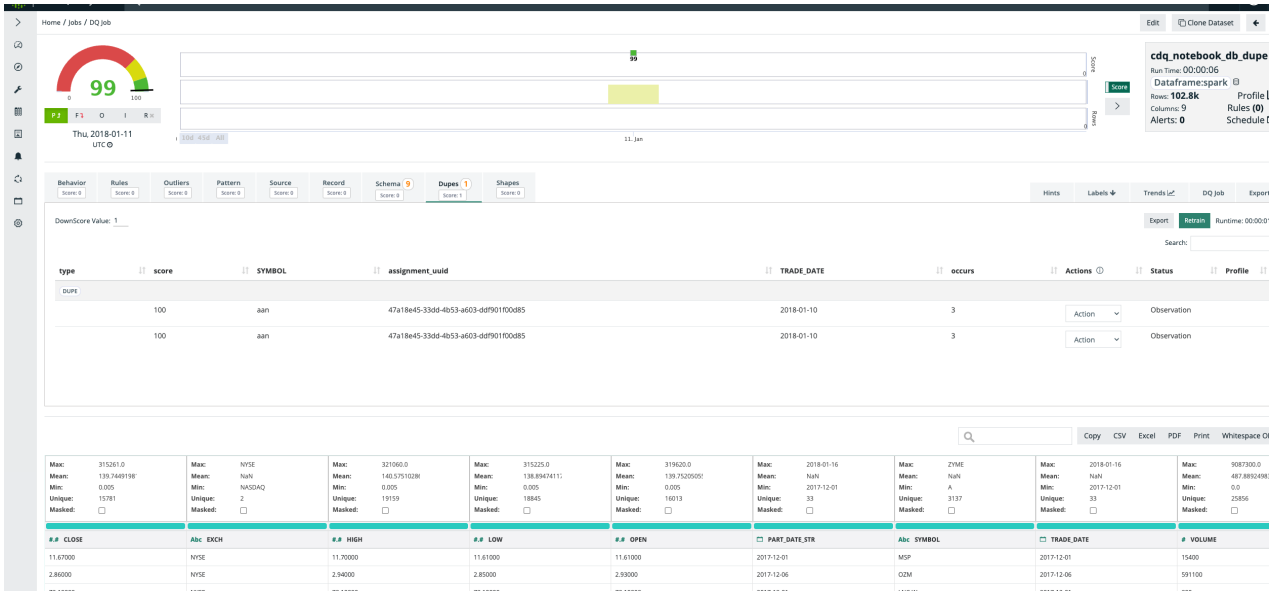

Create Collibra DQ Test: Profile
val dataset = "cdq_notebook_db_profile3"
var date = "2018-01-11"
// Options
val options = new DwlOptions()
options.dataset = dataset
options.runId = date
options.host = pgHost
options.port = pgPort
options.pgUser = pgUser
options.pgPassword = pgPass
//Scan
val cdq = DwlUtils.DwlContext(df, opt)
cdq.register(options)
cdq.dwlCheck()
val profile = cdq.profileDF()
profile.show()


Data Quality
Home / Jobs / DQ Job
100
Thu, 2018-01-11
UTC
cdq_notebook_db_profil...
Run Time: 00:00:05
Dataframe:spark
Rows: 102.8k
Columns: 9
Alerts: 0
Profile
Rules (0)
Schedule
Behavior
Rules
Outliers
Pattern
Source
Record
Schema
Dupes
Shapes
Hints
Labels
Trends
DQ Job
Export
Observation
Action
Status
Profile
COLUMN ADDED CLOSE
COLUMN ADDED EXCH
COLUMN ADDED HIGH
COLUMN ADDED LOW
COLUMN ADDED OPEN
COLUMN ADDED PART_DATE_STR
Observation
Copy
CSV
Excel
PDF
Print
Whitespace OFF
CLOSE
EXCH
HIGH
LOW
OPEN
PART_DATE_STR
SYMBOL
TRADE_DATE
VOLUME

Create a Collibra DQ Test (Dupes)

```
val dataset = "cdq_notebook_db_dupe"
var date = "2018-01-11"

// Options
val options = new OwlOptions()
options.dataset = dataset
options.runId = date
options.host = pgHost
options.port = pgPort
options.pgUser = pgUser
options.pgPassword = pgPass

opt.dupe.ignoreCase = true
opt.dupe.on = true
opt.dupe.lowerBound = 99
opt.dupe.include = Array("SYMBOL", "TRADE_DATE")

//Scan
val cdq = OwlUtils.OwlContext(df, opt)
cdq.register(options)
cdq.owlCheck()

val dupesDf = cdq.getDupeRecords
dupesDf.show()
```




Home / Jobs / DQ Job
Edit
Clone Dataset
99
Thu, 2018-01-11
UTC
cdq_notebook_db_dupe
Run Time: 00:00:06
Dataframe:spark
Rows: 102.8k
Columns: 9
Alerts: 0
Rules (0)
Behavior
Rules
Outliers
Pattern
Source
Record
Schema 9
Dupes 1
Shapes
Hints
Labels
Trends
DQ Job
DownScore Value: 1
Export
Retrain
Search:
type
score
SYMBOL
assignment_uuid
TRADE_DATE
occurs
Actions
Status
Profile
DUPE
100
aan
47a18e45-33dd-4b53-a603-ddf901f00d85
2018-01-10
3
Action
Observation
Copy
CSV
Excel
PDF
Print
CLOSE
EXCH
HIGH
LOW
OPEN
PART_DATE_STR
SYMBOL
TRADE_DATE
VOLUME
11.67000
NYSE
11.70000
11.61000
11.61000
2017-12-01
MSP
2017-12-01
15400
2.86000
NYSE
2.94000
2.85000
2.93000
2017-12-06
OZM
2017-12-06
591100

Create a Collibra DQ Test (Outlier)

```
import scala.collection.JavaConverters._
import java.util
import java.util.{ArrayList, List, UUID}

val dataset = "cdq_notebook_db_outlier"
var date = "2018-01-11"

// Options
val options = new OwlOptions()
options.dataset = dataset
options.runId = date
options.host = pgHost
options.port = pgPort
options.pgUser = pgUser
options.pgPassword = pgPass

opt.dupe.on = false

val dlMulti: util.List[OutlierOpt] = new util.ArrayList[Out-
lierOpt]
val outlierOpt = new OutlierOpt()
outlierOpt.combine = true
outlierOpt.dateColumn = "trade_date"
outlierOpt.lookback = 4
outlierOpt.key = Array("symbol")
outlierOpt.include = Array("high")
outlierOpt.historyLimit = 10
dlMulti.add(outlierOpt)

opt.setOutliers(dlMulti)

val cdq = OwlUtils.OwlContext(df, opt)
        .register(opt)

cdq.owlCheck
val outliers = cdq.getOutliers()
outliers.show
outliers.select("value")
```

Create Collibra DQ Test (ValidateSource)

```
import com.owl.common.options.SourceOpt
import java.util
import java.util.{ArrayList, List}

var opt = new OwlOptions()
val dataset = "weather-validateSrc"
opt.setDataset(dataset)
opt.runId = "2018-02-23"
clearPreviousScans(opt)
val src = Seq(
  ("abc", true, 55.5, "2018-02-23 08:30:02"),
  ("def", true, 55.5, "2018-02-23 08:30:02"),
  ("xyz", true, 55.5, "2018-02-23 08:30:02")
).toDF("name", "sunny", "feel-like-temp", "d_date")

val target = Seq(
  ("abc", 72, false, 55.5, "2018-02-23 08:30:02"), // true to
false
  ("xyz", 72, true, 65.5, "2018-02-23 09:30:02") // 08 to 09
).toDF("name", "temp", "sunny", "feel-like-temp", "d_date")

val optSource = new SourceOpt
optSource.on = true
optSource.dataset = dataset
optSource.key = Array("name")
opt.setSource(optSource)

//scan
val cdq = OwlUtils.OwlContext(src, target, opt)
  .register(opt)
cdq.owlCheck

val breakCountDf = cdq.getSourceBreaksCount()
breakCountDf.show()
```

Known API Limitations

Collibra DQ activities cannot currently be called independently. DQ Check() function should be called before calling any of the activities. For example, to get the profile DataFrame you should call the following code snippet:

```
cdq.owlCheck()
cdq.getProfileDF()
```

DQ-Databricks Submit

Introduction

In this page we will demonstrate two paths to run a spark submit job on Databricks's cluster. The first approach is to run a DQ spark submit job using Databricks UI and the second approach is by invoking Databricks rest API.

> Note These are only examples to demonstrate how to achieve DQ spark submit to Databricks's cluster. These paths are **not** supported in production and DQ team does **not** support any bug coverages or professional services or customer questions for these flows.

Limitations

There are a few limitation to spark-submit jobs in Databricks listed in this section: https://docs.databricks.com/jobs.html#create-a-job. Also, spark-submit is only on new clusters from both the UI via Jobs or calling the REST APIs. See Step 4 in: https://docs.databricks.com/jobs.html#create-a-job where it lists that spark-submit is handled by new clusters only.

Steps to create and run a spark submit job from Databricks UI:

1. Grant Collibra DQ Database access to your instance of Databricks.
2. Upload DQ jars in Databricks File System (DBFS).
3. Set up environment variables for your new cluster.
4. Prepare the DQ JSON payload.
5. Create and Run your job.
6. View the status and result of your job from the DQ Jobs page.

Database access

To begin, ensure that sure your Databricks instance has access to the DQ Database.

The entire subnet must be whitelisted to connect to the database. As specified in Databricks' documentation on subnets, Databricks must have access to at least two subnets for each database. To connect to the two Databricks subnets where the nodes will be instantiated, you must allow AWS to whitelist your IP address range.

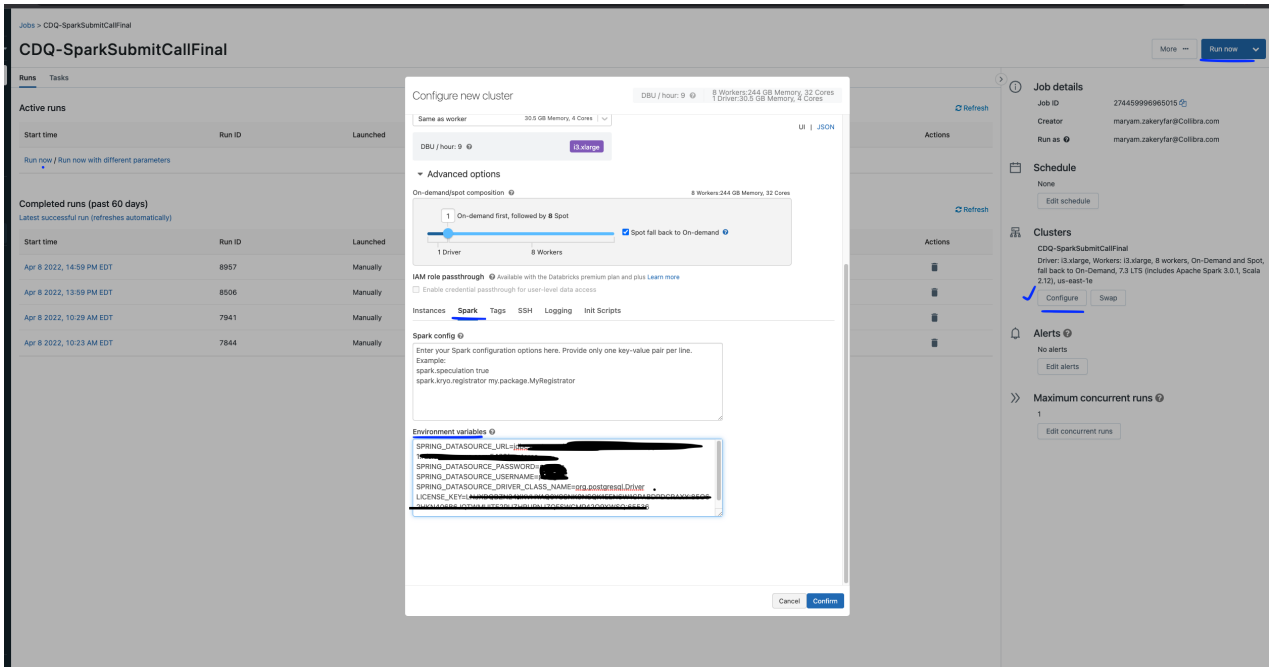
Upload DQ's jars in DBFS

The jars should be manually uploaded in Databricks file system. The steps can be found on Databricks website: https://docs.databricks.com/data/databricks-file-system.html#access-dbfs.

Environment variables for the new cluster:

Here is the documentation from Databricks about how to set up environment variables: https://docs.databricks.com/clusters/configure.html#environment-variables

These CDQ environment variables should be set on the new cluster : `SPRING_DATASOURCE_URL=xx\SPRING_DATASOURCE_USERNAME=xx\SPRING_DATASOURCE_DRIVER_CLASS_NAME=xx\LICENSE_KEY=xx // This is DQ's license key`



JSON payload

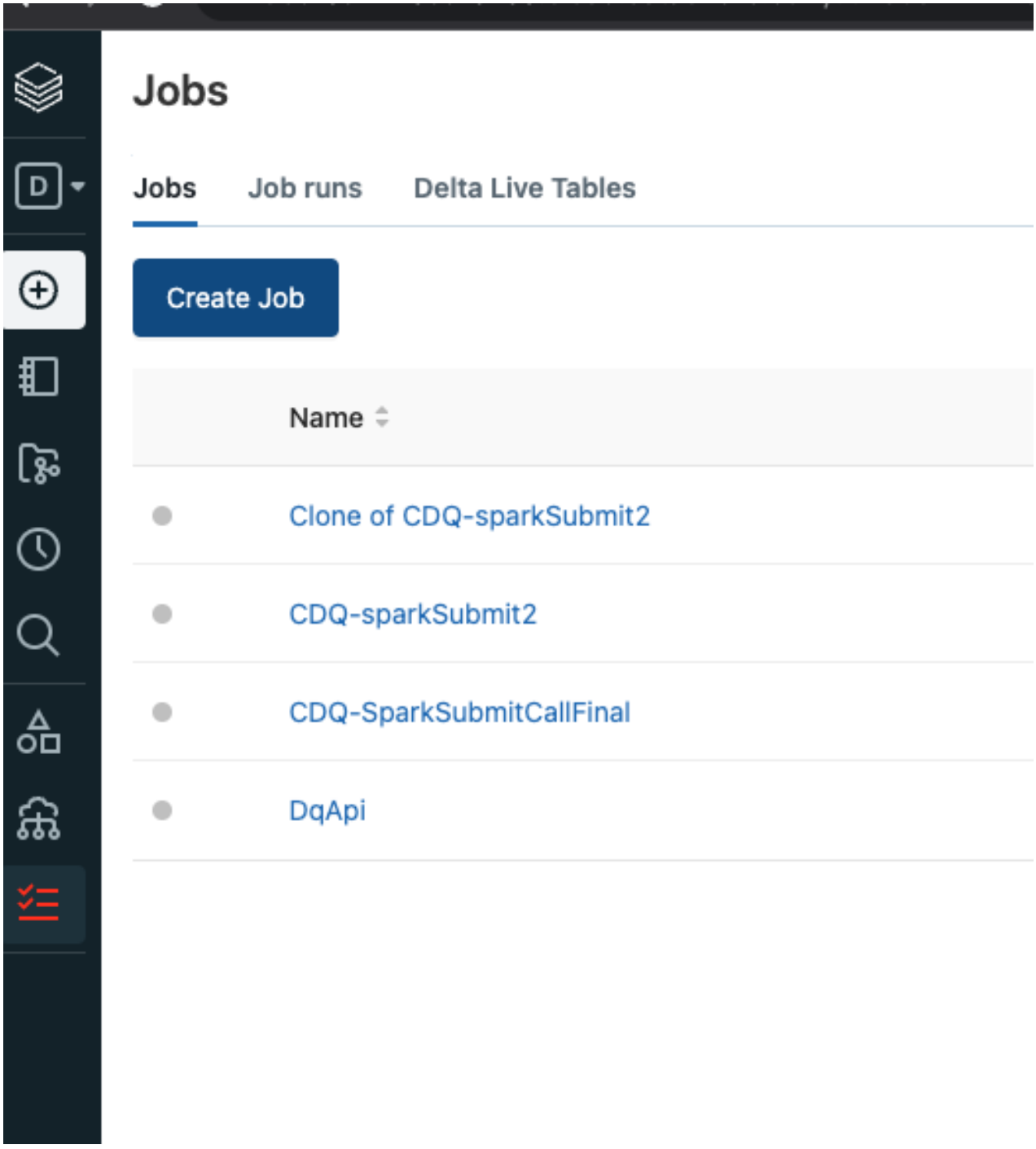
Once the above steps are completed, you can submit a spark submit job with DQ's parameters. Payload parameters can be from DQ's web Run command. You can copy and paste the parameters to prepare a JSON payload. Here is one sample:

```
                    "--class",
                   "com.owl.core.cli.OwlCheck",
                   "dbfs:/FileStore/cdq/owl-core-2022.02-
SPARK301-jar-with-dependencies.jar",
                   "-lib",
                   "dbfs:/FileStore/cdq/owl/drivers/postgres",
                   "-q",
                   "select * from xx.xxx",
                   "-bhlb",
                   "10",
                   "-rd",
                   "2022-03-16",
                   "-driver",
                   "owl.com.org.postgresql.Driver",
                   "-drivermemory",
                   "4g",
                   "-cxn",
                   "metastore",
                   "-h",
                   "xxxx.xxxxxx.amazonaws.com:xxxx/postgres",
                   "-ds",
                   "public.agent_2",
                   "-deploymode",
                   "cluster",
                   "-owluser",
                   "admin"
               ]
```

Run the job

Once you have completed the above steps, you can create a spark submit job through Databricks UI.

You can then add the environment variables to the cluster and click Run on Databricks UI.



Check the result in DQ web:

Once the job is submitted, you can login to your DQ web instance and check the job in the Jobs page.

Spark submit by invoking Databricks REST API

There are public REST APIS available for the Jobs API, including the latest version.

For this path we need to do the steps 1-4 of the the previous section and then call directly the REST API using Postman, or your preferred API testing tool. We assume that as per step 2, CDQ jars are uploaded to the DBFS path in the location dbfs:/FileStore/cdq. Also JDBC postgres driver should be uploaded to DBFS. For example: dbfs:/`FileStore/cdq/owl/drivers/postgres`

Steps:

1. Prepare the DQ JSON Payload.
2. Authenticate the Databricks REST API.

JSON payload

Sample JSON payload:

`POST /api/2.1/jobs/runs/submit HTTP/1.1Host: xxxxxx.cloud.databricks.com\ `Content-Type: application/json`\ `Authorization: Bearer ~~xxxxxxxxxxxx~~\ Cache-Control: no-cache\ Postman-Token: xxxxxxxx`

```
{
  "tasks": [
    {
      "task_key": "CDQ-SparkSubmitCallFinal",
      "spark_submit_task": {
        "parameters": [
          "--class",
          "com.owl.core.cli.OwlCheck",
          "dbfs:/FileStore/cdq/owl-core-2022.02-SPARK301-jar-
with-dependencies.jar",
          "-lib",
          "dbfs:/FileStore/cdq/owl/drivers/postgres",
          "-q",
          "select * from public.agent",
          "-bhlb",
          "10",
          "-rd",
          "2022-03-16",
          "-driver",
          "owl.com.org.postgresql.Driver",
          "-drivermemory",
          "4g",
          "-cxn",
          "metastore",
          "-h",
          "xxxs.amazonaws.com:xxx/postgres",
          "-ds",
          "public.agent_2",
          "-deploymode",
          "cluster",
          "-owluser",
          "admin"
        ]
      },
      "new_cluster": {
        "cluster_name": "",
        "spark_version": "7.3.x-scala2.12",
        "aws_attributes": {
          "zone_id": "us-east-1e",
          "first_on_demand": 1,
          "availability": "SPOT_WITH_FALLBACK",
          "spot_bid_price_percent": 100,
          "ebs_volume_count": 0
        },
        "node_type_id": "i3.xlarge",
        "spark_env_vars": {
          "SPRING_DATASOURCE_URL": "jdbc:postgresql://xxx-xx-
xxs.amazonaws.com:xx/postgres",
```

```
            "SPRING_DATASOURCE_PASSWORD": "xxx",
            "SPRING_DATASOURCE_USERNAME": "xxx",
            "SPRING_DATASOURCE_DRIVER_CLASS_NAME": "org.-
postgresql.Driver",
            "LICENSE_KEY": "xxxx"
          },
          "enable_elastic_disk": false,
          "num_workers": 8
        },
        "timeout_seconds": 0
      }
    ]
}
```

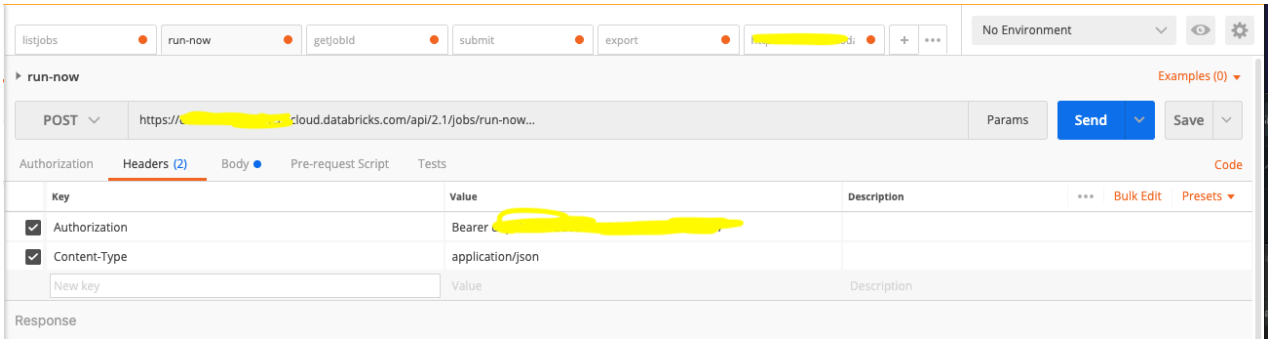
Values to be updated in above payload are:

Cluster variables:`"SPRING_DATASOURCE_URL":"SPRING_DATASOURCE_PASSWORD":"SPRING_DATASOURCE_USERNAME":"LICENSE_KEY": //CDQ License key`` ``

CDQ variables\ Users can customize the variables based on the activity they choose from CDQ Web. They can copy the variables from Run CMD option of their DQ job and paste it in their Json message. ``

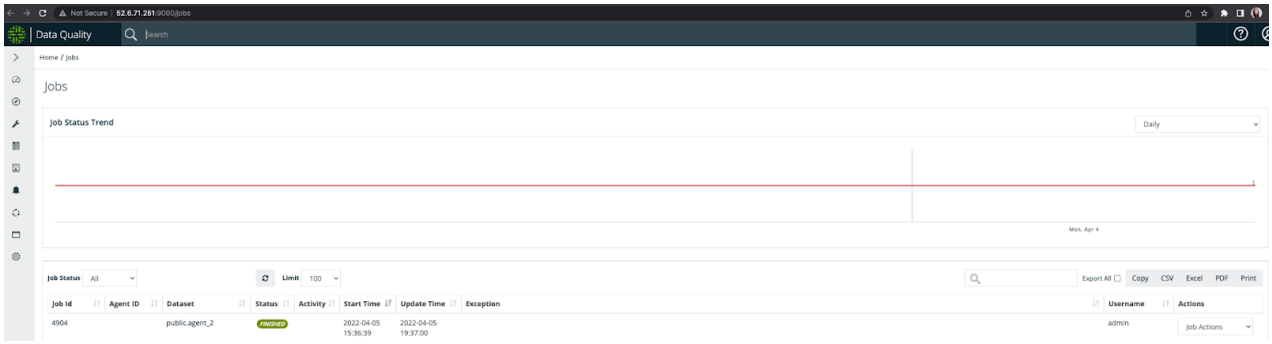
Authenticate the Databricks REST API

Here is the Databricks documentation about how to create a personal access token: https://docs.databricks.com/dev-tools/api/latest/authentication.html



View the job's result in DQ web

You can view the result of your job run by navigating to the DQ Jobs page.



Examples

Programmatic DQ

Don't like leaving your notebook? Want to build DQ into your in-house data quality pipeline? Collibra DQ can do both!

Simple

Load Table use SparkJDBC

```
//--- Configure Table From Database ---//
val connProps = Map (
  "driver"   -> "org.postgresql.Driver",
  "user"     -> s"${user}",
  "password" -> s"${pass}",
  "url"      ->
s"jdbc:postgresql://${host}:${port}/${database}",
  "dbtable"  -> "owl_test.nyse"
)

//--- Load Spark DataFrame ---//
val jdbcDF = spark.read.format("jdbc").options(connProps).load
jdbcDF.show
```

Configure Collibra DQ Options

Connect to DQ's Metadata Database and control DQ scan options. Wrap sparkDF with DQ context.

```
import com.owl.common.options._
import com.owl.core.util.OwlUtils

val opt = new OwlOptions
//--- Owl Metastore ---//
opt.host = s"$owlHost"
opt.port = s"5432/postgres?currentSchema=public"
opt.pgUser = s"$owlUser"
opt.pgPassword = s"$owlPass"
//--- Run Options ---//
opt.dataset = "owl_test.nyse"
opt.runId = "2018-01-10"
opt.datasetSafeOff = true

val owl = OwlUtils.OwlContext(jdbcDF, opt)
```

Register with Catalog and Run Profile

```
//--- Register with Owl Catalog ---//
owl.register(opt)

//--- Profile Dataset ---//
val profile = owl.profileDF
profile.show
```

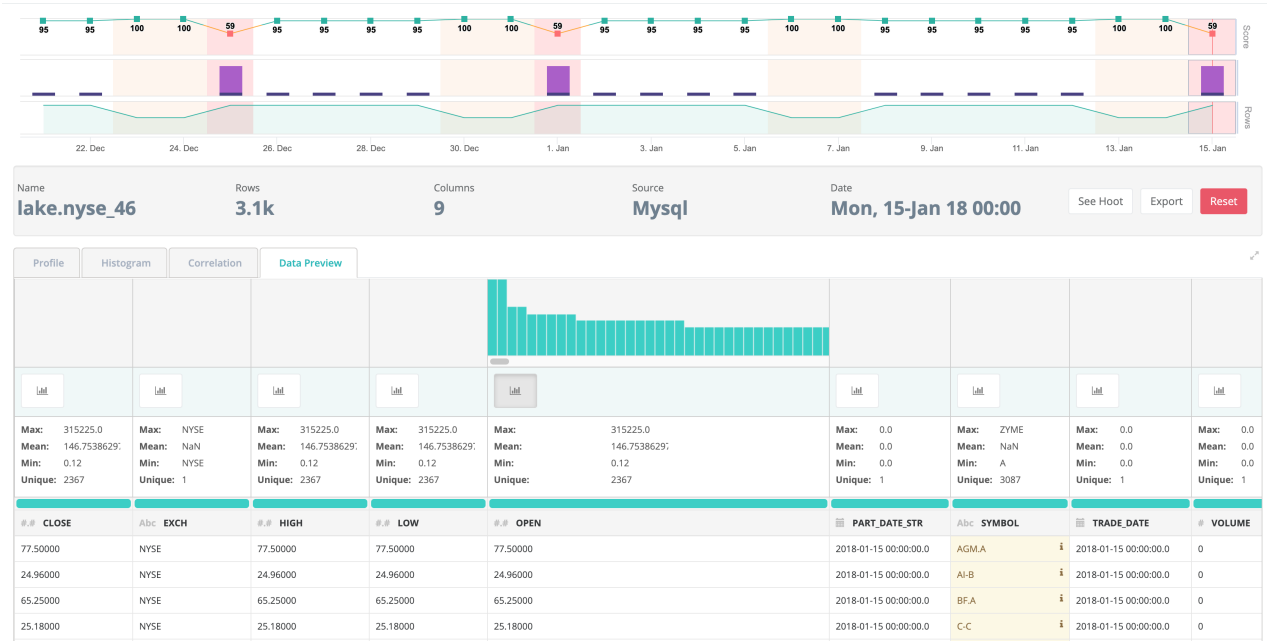
Notice that DQ returns results as Dataframes. This is a fantastic abstraction that allows you to ignore all domain objects and custom types and interact with a scaleable generic result set using common protocols like "where" or "filter" or "save" or "write" all with parallel operations.

```
+--------------+-----+-------+-----------+---+---+--------+-----
------+------+----+------+-------+-------+------+----+---------+
|        column|nulls|empties|cardinality|min|max|is_mixed|-
mixed_
ratio|   Int|Long|String|Decimal|Boolean|Double|Date|Timestamp|
+--------------+-----+-------+-----------+---+---+--------+-----
------+------+----+------+-------+-------+------+----+---------+
|     tenant_
id|
0
|
0
|
60
|
0
|
9
|
false
|
0.0|100000|   0|     0|      0|      0|     0|   0|         0|
|           a11|
0
|
0
|
1
|a11|a11|
false
|
0.0|     0|   0|100000|      0|      0|     0|   0|         0|
|           a10|
0
|
0
|
1
|a10|a10|
false
|
0.0|     0|   0|100000|      0|      0|     0|   0|         0|
|  account_type|    0|      0|          3| 02|
06
|
false
|
0.0|100000|   0|     0|      0|      0|     0|   0|         0|
```

```
|           a13|
0
|
0
|
1
|a13|a13|
false
|
0.0|     0|   0|100000|      0|      0|     0|   0|         0|
|security_
alias|
0
|
0
|
3
|
0
|
2
|
false
|
0.0|100000|   0|     0|      0|      0|     0|   0|         0|
|           a12|
0
|
0
|
1
|a12|a12|
false
|
0.0|     0|   0|100000|      0|      0|     0|   0|         0|
+--------------+-----+-------+----------+---+---+--------+-----
------+------+----+------+-------+-------+------+----+---------+
```

Profile UI

While the spark DF.show() is a nice and convenient output format, you may prefer a rich UI visual that tracks the data tests over time. The UI provides trend analysis, data drift, data relationships and more.



Duplicates

Take duplicate detection for example. A common use case where a business wants to make sure they do not have repeated or duplicate records in a table. Set the lowerBound to the percent fuzzy match you are willing to accept, commonly 87% or higher is an interesting match. You might also want to target a single day or week or month that you shouldn't have dupes within. Notice the .where function and then pass in a custom dataframe to the DQ context.

```
opt.dupe.on = true
opt.dupe.lowerBound = 99
opt.dupe.include = Array("SYMBOL", "EXCH")

val df1Day = jdbcDF.where("TRADE_DATE = '2018-01-10' ")
val owl = OwlUtils.OwlContext(df1Day, opt)

val dupes = owl.dupesDF
dupes.show

// rdd collect
dupes.rdd.collect.foreach(println)

// records linked together for remediation
owl.getDupeRecords.show
```

Outliers

Gaining and understanding of your outliers is a commonly desired DQ function. DQ has several configurations to help find the most meaningful outliers in your dataset and over time. Below compares the current day to a baseline of days in the historical dataframe.

```
opt.outlier.on = true
opt.outlier.lookback = 6
opt.outlier.dateColumn = "TRADE_DATE"
opt.outlier.timeBin = OutlierOpt.TimeBin.DAY
opt.outlier.key = Array("SYMBOL")

val df1Day = jdbcDF2.where("TRADE_DATE = '2018-01-10' ")
val owl = OwlUtils.OwlContextWithHistory(dfCurrent = df1Day,
dfHist = jdbcDF2, opt = opt)
val outliers = owl.outliersDF
outliers.show
```

Advanced

Programmatic DQ

Don't like leaving your notebook? Want to build data quality into your in-house data quality pipeline? Collibra DQ can do both!

Real World Examples

Rules

Let's assume we were provided a file named "atm_cust_file" and want to load it into a database table as well as scan it for all possible errors. We want to provide a couple levels of protection. 1) A business rule checking if *a customer joined before before the company was founded*. 2) Check if the file 100% matches to the DataFrame or db table we've created. 3) Check for all possible outliers or anomalies in the dataset. Each one of these 3 issues had a different impact to the business and causes a different flow to trigger in our pipeline.

Add Rule

Let's create a simple rule and assign points to the overall scoring system for later delegation.

```
    val rule = new domain2.Rule
    rule.setRuleNm("customer_before_company")
    rule.setRuleValue("customer_since_date < '1956-11-01'")
    rule.setPerc(1.0)
    rule.setPoints(1)
    rule.setIsActive(1)
    rule.setUserNm("Kirk")
    rule.setDataset("ATM_CUSTOMER3")
    Util.addRule(rule=rule)
```

Now let's chain together the remaining two items that were part of our original requirement. Note that DQ has six additional ML DQ features that we did not turn on in this case.

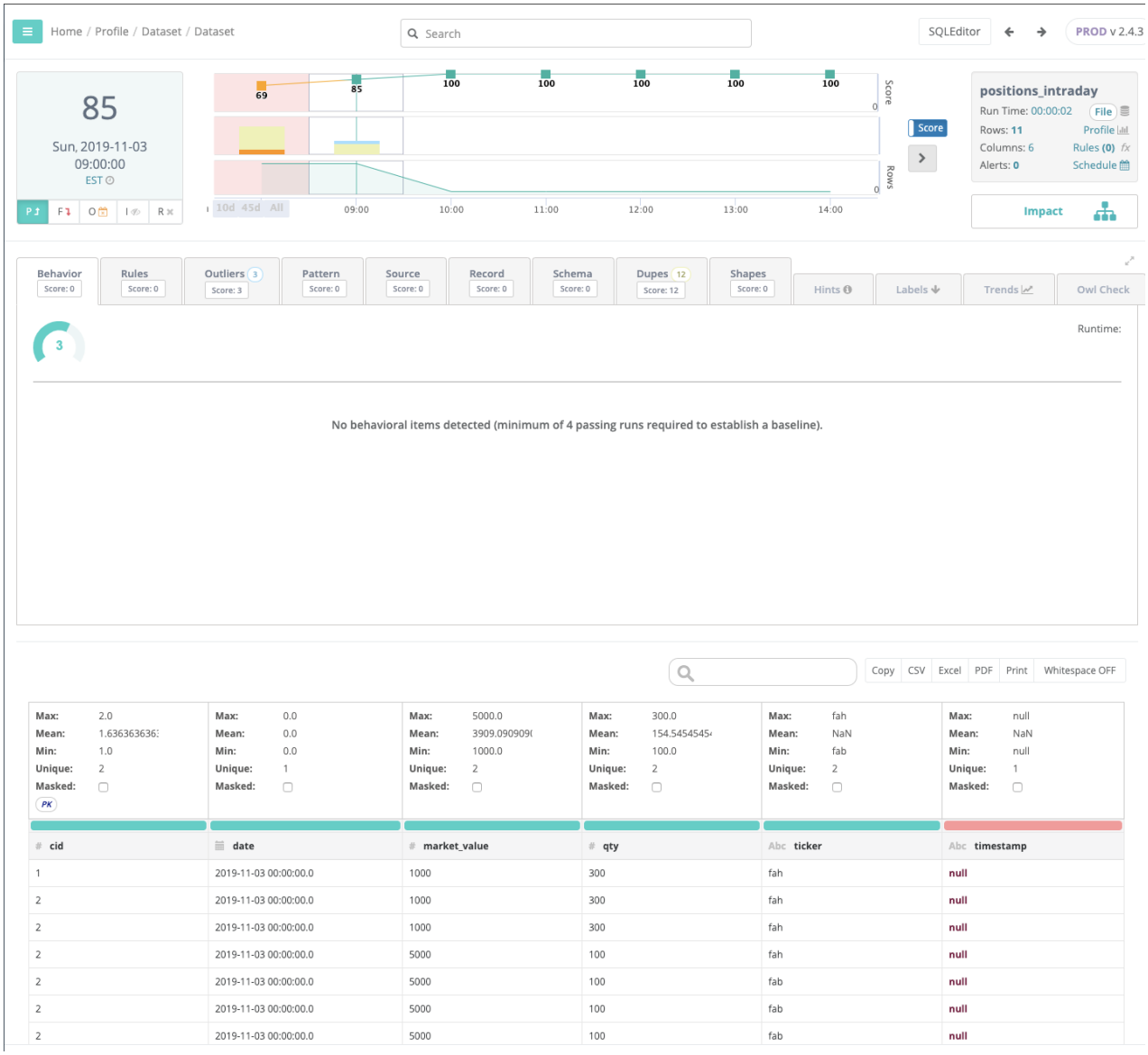
```
val owl = Util.OwlContext(df, atmCustFile, props)

// first register with catalog if not registered
owl.register(props)

// Check if dataframe matches the source file 'atm_cust_file'
val source = owl.validateSrcDF
if (source.count() > 1) {
  // create service now ticket and exit with fail based on not
matching to original file
}

owl.addAdHocRule(rule)
val ruleBreaks = owl.rulesDF
if (ruleBreaks.count() > 1) {
  if (ruleBreaks.where($"score" > 5).count > 1) {
    // create service now ticket and exit with fail based on
rules
  }
}

val outliers = owl.outliersDF
if (outliers.where($"confidence" < 10).count > 3) {
  // Owl email Alert to business group for attention
  // where 3 outliers have a confidence below 10
}
```

Ingesting Intraday Files

Here we illustrate an example of how to work with files when using DQ programmatically. This can be implemented in both a Notebook setting and in your own codebase.

```
//////////////////////////////////////////////////////////////-
///////////
    //                  USE CASE - Ingesting Intraday
Files                  //

//////////////////////////////////////////////////////////////-
///////////

    // Part of your pipeline includes the ingestion of files
that have the date
    // and hour encoded in the file name. How do you process
those files using Owl?
    //
    // Format: <name>_<year>_<month>_<day>.csv
    //
    // Build up a data structure containg the files you want to
process (here we
    // just use a simple list, but you may want to be pulling
from a pubsub
    // queue, AWS bucket, etc...). Here we just use a simple
file list of 6
    // hours of trade position data.
    val position_files = List(
      new File(getClass.getResource("/position_file_2019_11_03_
08.csv").getPath),
      new File(getClass.getResource("/position_file_2019_11_03_
09.csv").getPath),
      new File(getClass.getResource("/position_file_2019_11_03_
10.csv").getPath),
      new File(getClass.getResource("/position_file_2019_11_03_
11.csv").getPath),
      new File(getClass.getResource("/position_file_2019_11_03_
12.csv").getPath),
      new File(getClass.getResource("/position_file_2019_11_03_
13.csv").getPath),
      new File(getClass.getResource("/position_file_2019_11_03_
14.csv").getPath))

    // Create your spark session.
    val spark = SparkSession.builder
      .master("local")
      .appName("test")
      .getOrCreate()

    // Configure Owl.
    val opt = new OwlOptions
    opt.dataset = "positions"
    opt.load.delimiter = ","
```

```
    opt.spark.master = "local[1]"
    opt.dupe.on = true
    opt.dupe.include = Array("ticker", "cid")
    opt.outlier.on = true
    opt.outlier.key = Array("cid")
    opt.outlier.timeBin = TimeBin.HOUR
    // Customize this to only process a subset of the data.
    opt.load.fileQuery = "select * from dataset"

    position_files.foreach { file: File =>
      // Tell Owl where to find the file.
      opt.load.filePath = file.getPath

      // Parse the filename to construct the run date (-rd) that
will be passed
      // to Owl.
      val name = file.getName.split('.').head
      val parts = name.split("_")
      val date = parts.slice(2, 5).mkString("-")
      val hour = parts.takeRight(1).head

      // Must be in format 'yyyy-MM-dd' or 'yyyy-MM-dd HH:mm'.
      val rd = s"${date} ${hour}"

      // Tell Owl to process data
      opt.runId = rd

      // Create a DataFrame from the file.
      val df = OwlUtils.load(opt.load.filePath, opt.-
load.delimiter, spark)

      // Instantiate an OwlContext with the dataframe and our
custom configuration.
      val owl = OwlUtils.OwlContext(df, spark, opt)

      // Make sure Owl has catalogued the dataset.
      owl.register(opt)

      // Let Owl do the rest!
      owl.owlCheck()

    }
```



All Pipeline Activities in One Line

For brevity and convenience, DQ allows a DF to be loaded in the constructor and in one line run all nine dimensions of data quality "owl.owlcheck". To adjust the DQ dimensions you simply set the properties in the props object.

```
val owl = Util.OwlContext(df, atmCustFile, props)
owl.owlCheck
```

Example of some common property settings

```
val props = new Props()
props.filePath = s"${filePath}/atm_customer_${rd.replace("-","_
")}.csv"
props.runId = rd
props.dateCol = "OWL_RUN_ID"
props.dataset = "ATM_CUSTOMER3"
props.del = ","
props.datasetSafety = false
props.calculateBoundaries = true
props.fileLookBack = true
props.timeBin = "DAY"

// outlier, missing records
props.dl = true
props.dlKey = "customer_id"
props.dlLb = 4

// pattern mining
props.freqPatternMiningByKey = true
props.fpgKey = "customer_id"
props.fpgLookback = 4
props.fpgDateCol = "OWL_RUN_ID"
props.fpgCols = "card_number,first_name,last_name,checking_sav-
ings"
props.fpgLowFreq = true

// validate Src
props.validateSrc = true
props.valSrcKey = "customer_id"

// fuzzy match
props.isDupe = true
props.dupeCutOff = 88
props.depth = 3
props.dupeExcludeCols = "customer_id,card_number,customer_since_
date,OWL_RUN_ID"
```

README.md

OwlSpark DQ Pipelines

Build your own rich data quality pipelines directly on dataframes using Owl's DQ suite

Owl Predictive Data Science + Spark Scale

Auto Clean

first	last	age	ssn	phone
Kirk	brian	11	222-33-4444	410-489-8095
kirk	kirk	22	111-33-4444	410-489-8095
bryan	Kirk	11	555-33-4444	443-885-8989
Dan	dan	33	666-33-4444	(212) 333-4444
	brian	11	222-33-4444	(212) 333-4444
kirk	kirk	21	111-33-4444	(212) 333-4444
brianne	kirk	18	555-33-4444	3015859966
dan	dan	31		
kirk	brian	66		
kirk	null	age-27		
brianne	Kirt	13		
dan	DAN	35		
kirk	brian	16		

first	last	age	ssn	phone
Kirk	brian	11	222-33-4444	4104898095
kirk	kirk	22	111-33-4444	4104898095
bryan	Kirk	11	555-33-4444	4438858989
Dan	dan	33	666-33-4444	2123334444
	brian	11	222-33-4444	2123334444
kirk	kirk	21	111-33-4444	2123334444
brianne	kirk	18	555-33-4444	3015859966
dan	dan	31	666-33-4444	3015859966
kirk	brian	66	222-33-4444	3015859966
kirk	null	27	111-33-4444	3015859966
brianne	Kirt	13	555-33-4444	3015859966
dan	DAN	35	666-33-4444	3015859966
kirk	brian	16	222-33-4444	4104898095

```
val df = Seq(
  ("Kirk",  "brian",  "11",  "222-33-4444",  "410-489-8095"),
  ("kirk",  "kirk",   "22",  "111-33-4444",  "410-489-8095"),
  ("bryan", "Kirk",   "11",  "555-33-4444",  "443-885-8989"),
  ("Dan",   "dan",    "33",  "666-33-4444",  "(212) 333-4444"),
  ("",      "brian",  "11",  "222-33-4444",  "(212) 333-4444"),
  ("kirk",  "kirk",   "21",  "111-33-4444",  "(212) 333-4444"),
  ("brianne","kirk",  "18",  "555-33-4444",  "3015859966"),
  ("dan",   "dan",    "31",  "666-33-4444",  "3015859966"),
  ("kirk",  "brian",  "66",  "222-33-4444",  "3015859966"),
  ("kirk",   null,    "age-27", "111-33-4444","3015859966"),
  ("brianne","Kirt",  "13",  "555-33-4444",  "3015859966"),
  ("dan",   "DAN",    "35",  "666-33-4444",  "3015859966"),
```

Using Notebooks to build DQ Pipelines

For examples on how to do this, see our Notebook repository below.

GitHub - kirkhas/owl-notebooks: Owl Spark DQ Pipelines

Collibra Data Quality & Observability Rules

Global rules

Distributed with the application by default and you can use them from UI/CLI/Notebooks.

Rule types

- Invalid_Email_Check
- Invalid_Phone_Num_Check
- Invalid_Zip_Code_Check
- Invalid_SSN_Check
- Invalid_IP_Address_Check
- Invalid_Gender_Check
- Invalid_EIN_Check
- Invalid_State_Check
- Invalid_Credit_Card_Check
- Valid_Email_Check
- Valid_Phone_Num_Check
- Valid_Zip_Code_Check
- Valid_SSN_Check
- Valid_IP_Address_Check
- Valid_Gender_Check
- Valid_EIN_Check
- Valid_State_Check
- Valid_Credit_Card_Check
- Percent_Move_5 Percent_Move_10
- Percent_Move_20 Percent_Move_50
- Not_In_Previous_run
- Not_In_Current_run
- Having_Count_Greater_Than_One

SQL based rules

Simple rule

Simple rules would be applied to filter a condition on a single column in a single table.

Example #1

In this example you can see how to create a simple SQL rule, with name **simple_sql_rule**.

Code	Description
rule.setRuleNm("**simple_sql_rule**")	Adding the name of the given rule
rule.setRuleValue("**startDate < '2011-11-01'**")	Setting the simple SQL expression. No **JOIN** allowed between tables!
rule.setRuleType("**SQLG**")	Setting the rule type

Code

example_simple_sql_rule.scala

```
import com.owl.core.Owl
import com.owl.core.util.OwlUtils
import com.owl.common.bll.{RuleBll, RuleTemplateBll}
import com.owl.common.domain2.Rule
import com.owl.common.options.{LoadOpt, OwlOptions}

import org.junit.{Assert, Test}

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

val loadOptions = new LoadOpt {
  pghost = "localhost:5432/postgres"
  pguser = "username"
  pgpassword = "password"
}

//----- Init Spark ----- //
def sparkInit(): SparkSession = {
  val sparkSession = SparkSession.builder
    .master("local")
    .appName("test")
    .getOrCreate()
```

```
    sparkSession
  }

  @Test def simpleRule(): Unit = {

    // Arrange
    val spark = sparkInit()
    import spark.implicits._

    val headers = "firstName,lastName,startDate"
    val source = Seq(
      ("Thomas", "Martinez", "2010-11-01"),
      ("Harry", "Williams", "2012-05-01"),
      ("Ethan", "Davis", "2009-08-01")
    )
    val arr = headers.split(",")
    val df = source.toDF(arr: _*)

    val opt = new OwlOptions {
      runId = "2019-09-20"
      dataset = "simple_sql_rule_ds"
      onReadOnly = false
      load = loadOptions
    }

    val rule = new Rule {
      setDataset(opt.dataset)
      setRuleNm("simple_sql_rule")
      setRuleValue("startDate < '2011-11-01'")
      setRuleType("SQLG")
      setPerc(1.0)
      setPoints(1)
      setIsActive(1)
      setUserNm("admin")
    }

    val owl = OwlUtils.OwlContext(df, opt)
      .register(opt)

    OwlUtils.addRule(rule)

    // Act
    owl.owlCheck()

    // Assert
    import scala.collection.JavaConversions
    val hootRule = JavaConversions.asScalaBuffer
  (owl.hoot.rules).find(x => rule.getRuleNm.equals
  (x.getRuleNm)).orNull
```

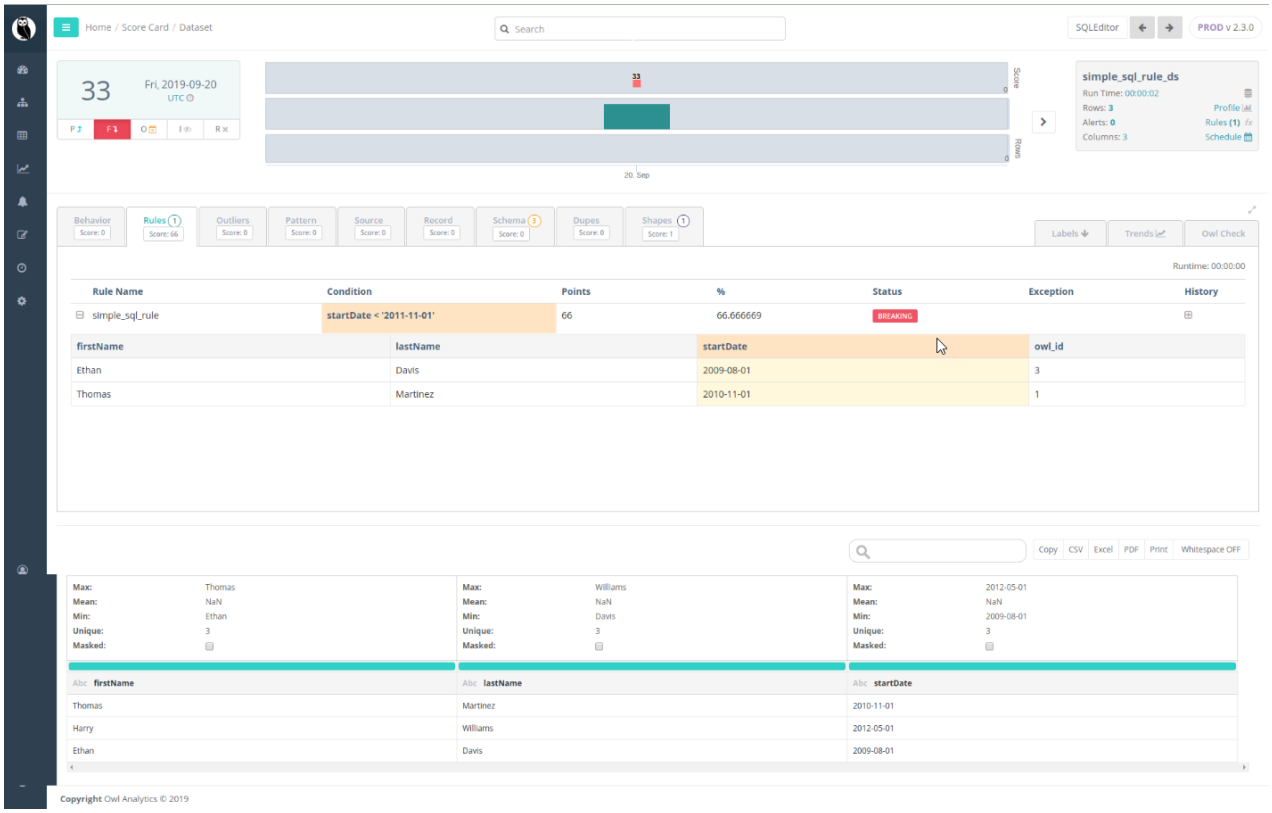
```
  Assert.assertNotNull(hootRule)
  Assert.assertEquals(66, hootRule.getScore)
}

// Execute notebook
simpleRuleNotebook()
```

Result

via Code

You can do multiple assertion on the result of the OwlCheck process.
Using **owl.hoot** parameter will provide access to the execution results, in this case for the rule.

via UI



Example #2

In this example you can see how to create a simple SQL with rule with **templates**, with name **simple_sql_rule_with_template**.

Steps

1. Create the rule template, where the template column name should be marked with **$colNm** string.\

```
val ruleTemplate = RuleTemplateBll.createRuleTemplate(
    "not_null_or_empty",
    "Column cannot contain null or empty values",
    " $colNm is null or $colNm = \'\' or $colNm  = \'null\'
"
)
```

2. Create the Rule instance, where value of **RuleValue** will be used to replace **$colNm** in the template expression.\

```
val rule = RuleBll.createRule(opt.dataset)
rule.setRuleNm("is_city_not_null_or_empty")
rule.setRuleValue("city")
rule.setRuleType("CUSTOM") // legacy type required to look
into rule repo
rule.setRuleRepo("not_null_or_empty") // custom rule name
to pull rule value from rule repo
rule.setPerc(1.0)
rule.setPoints(1)
rule.setIsActive(1)
rule.setUserNm("admin")
```

Code

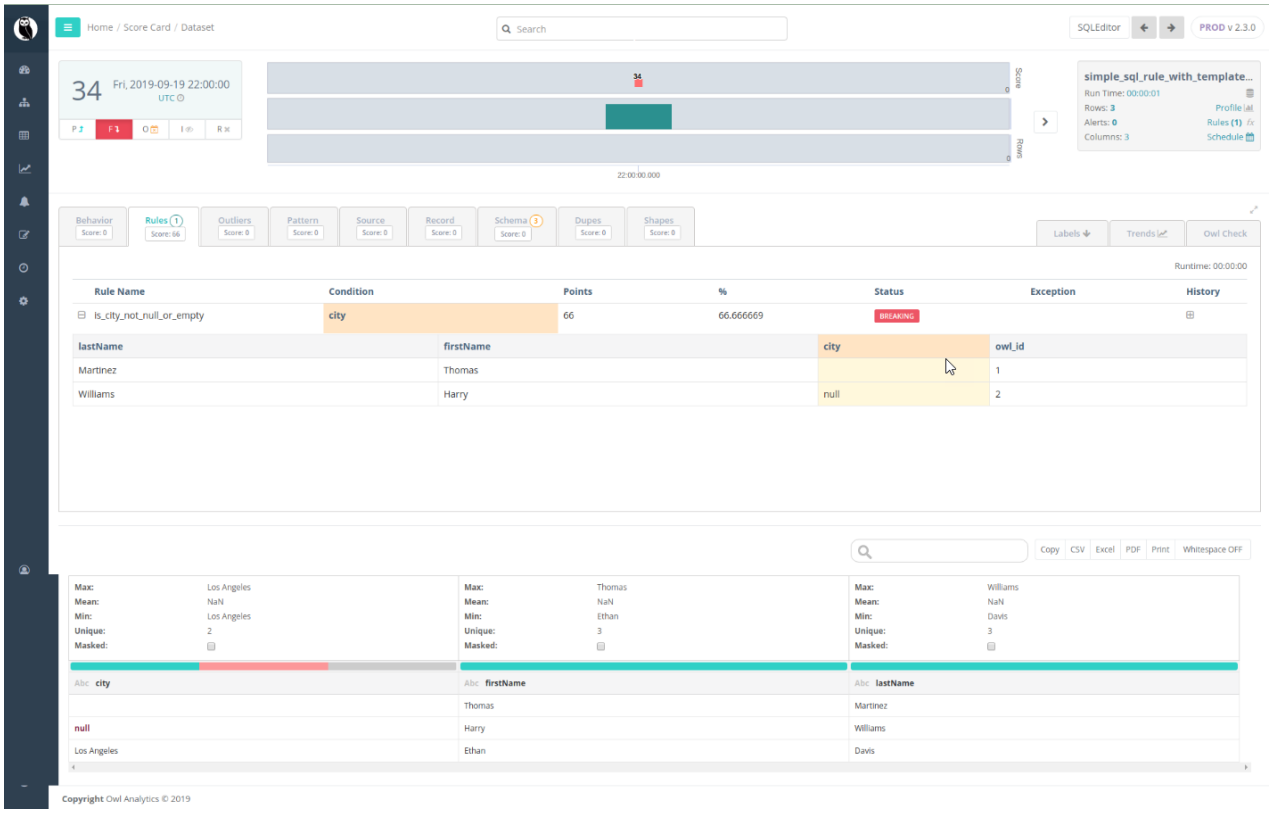
```
import com.owl.core.Owl
import com.owl.core.util.OwlUtils
import com.owl.common.bll.{RuleBll, RuleTemplateBll}
import com.owl.common.domain2.Rule
```

```
import com.owl.common.options.{LoadOpt, OwlOptions}

import org.junit.{Assert, Test}

import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

val loadOptions = new LoadOpt {
  pghost = "localhost:5432/postgres"
  pguser = "username"
  pgpassword = "password"
}

//----- Init Spark ----- //
def sparkInit(): SparkSession = {
  val sparkSession = SparkSession.builder
    .master("local")
    .appName("test")
    .getOrCreate()

  sparkSession
}

@Test def simpleRuleWithTemplate(): Unit = {

  // Arrange
  val spark = sparkInit()
  import spark.implicits._

  val headers = "firstName,lastName,city"
  val source = Seq(
    ("Thomas", "Martinez", ""),
    ("Harry", "Williams", null),
    ("Ethan", "Davis", "Los Angeles")
  )
  val arr = headers.split(",")
  val df = source.toDF(arr: _*)

  val opt = new OwlOptions {
    runId = "2019-09-20"
    dataset = "simple_sql_rule_with_template_ds"
    onReadOnly = false
    load = loadOptions
  }

  val ruleTemplate = RuleTemplateBll.createRuleTemplate("not_
null_or_empty","Column cannot contain null or empty values", "
$colNm is null or $colNm = \'\' or $colNm  = \'null\' ")

  val rule = RuleBll.createRule(opt.dataset)
```

```
  rule.setRuleNm("is_city_not_null_or_empty")
  rule.setRuleValue("city")
  rule.setRuleType("CUSTOM") // legacy type required to look
into rule repo
  rule.setRuleRepo("not_null_or_empty") // custom rule name to
pull rule value from rule repo
  rule.setPerc(1.0)
  rule.setPoints(1)
  rule.setIsActive(1)
  rule.setUserNm("admin")

  val owl = OwlUtils.OwlContext(df, opt)
    .register(opt)

  OwlUtils.addRuleTemplate(ruleTemplate)
  OwlUtils.addRule(rule)

  // Act
  owl.owlCheck()

  // Assert
  import scala.collection.JavaConversions
  val hootRule = JavaConversions.asScalaBuffer
(owl.hoot.rules).find(x => rule.getRuleNm.equals
(x.getRuleNm)).orNull
  Assert.assertNotNull(hootRule)
  Assert.assertEquals(66, hootRule.getScore)
}

// Execute notebook
simpleRuleWithTemplate()
```

Result

via UI



Freeform SQL

It would be used when applying a complex condition across multiple tables/columns and generally when more flexibility/customization is desired.

Individual statement

Syntax

```
SELECT * FROM @<dataset_name> <table_alias>
WHERE <filter_expression>
GROUP BY <group_by_expression>
HAVING <having_expression>
```

The base of the statement is given with **@<dataset_name>** style. In general, the <dataset_name> is the same, where the rule is attached to, but you can generally use any valid dataset name in the expression.

Examples

Simple rule expression

```
opt.dataset = "example_ds"

val rule = RuleBll.createRule(opt.dataset)
rule.setRuleNm("is_city_not_null_or_empty")
rule.setRuleValue("select * from @example_ds t where t.amount >
'5000'")
rule.setRuleType("SQLF")
rule.setPerc(1.0)
rule.setPoints(1)
rule.setIsActive(1)
rule.setUserNm("admin")
```

Complex rule expression

```
opt.dataset = "unique_rule_ds"

val rule = RuleBll.createRule(opt.dataset)
rule.setRuleNm("unique_rule")
rule.setRuleValue("select * from ( select count(*) as cnt, cus-
tomer_id from @unique_rule_ds group by customer_id ) having cnt
> 1")
rule.setRuleType("SQLF")
rule.setPerc(1.0)
rule.setPoints(1)
rule.setIsActive(1)
rule.setUserNm("admin")
```

RegExp expression

```
opt.dataset = "regexp_rule_ds"

val rule = RuleBll.createRule(opt.dataset)
rule.setRuleNm("LIKE_rule")
rule.setRuleValue("select * from @regexp_rule_ds.SYMBOL rlike
'^ABB+'  ")
rule.setRuleType("SQLG")
rule.setPerc(0.02)
rule.setPoints(1)
rule.setIsActive(1)
rule.setUserNm("admin")
```

Join statements

Available join types between multiple data sets

- WHERE tableA.id = tableB.id style
- INNER JOIN
- LEFT <OUTER> JOIN
- RIGHT <OUTER> JOIN

Joining other data sets

- Get the historical state of the same data set.
  Syntax: @t<n>, where the parameter *n* means, the historical run (number) of the

base data set (marked with @<data set_name>) to instruct the temporary job created at runtime to target. The following table demonstrates what each of the following @t*n* value examples mean:

@t*n* value	Description
@t1	Loads the data from the previous scan.
@t2	Loads the data from the second to last scan.
@t3	Loads the data from the third to last scan.
@t4	Loads the data from the fourth to last scan

- Get a different data set
  Syntax: @<other_data set_name>

WHERE style

Look-back dataset

```
SELECT * FROM @<dataset_name> <table_alias>, @t1 [<history_
table_alias>]
WHERE <join_expression> AND <filter_expression>
```

Example

```
opt.dataset = "example_ds"

val rule = RuleBll.createRule(opt.dataset)
rule.setRuleValue("select * from @example_ds t, @t1  where t.cus-
tomer_id = t1.customer_id  and t.card_number <> t1.card_number
")
rule.setRuleType("SQLF")
```

Different data set

```
SELECT * FROM @<dataset_name> <table_alias>, @<other_dataset_
name> [<other_alias>]
WHERE <join_expression> AND <filter_expression>
```

Example

```
opt.dataset = "example_ds"
opt2.dataset = "other_ds"

val rule = RuleBll.createRule(opt.dataset)
rule.setRuleValue("select * from @example_ds t, @other_ds ds2
where t.customer_id = ds2.customer_id  and t.card_number <> ds2.-
card_number ")
rule.setRuleType("SQLF")
```

LEFT JOIN

Example

```
opt.dataset = "example_ds"

val rule = RuleBll.createRule(opt.dataset)
rule.setRuleNm("not_back2back_days")
rule.setRuleValue(" select * from @example_ds A LEFT OUTER JOIN
@t1 B ON A.customer_id = B.customer_id where A.customer_id is
not null and B.customer_id is null  ")
rule.setRuleType("SQLF")
rule.setPerc(1.0)
rule.setPoints(1)
rule.setIsActive(1)
rule.setUserNm("admin")
```

Data type based rules

Simple check for individual columns.

Rule types

Empty check

Rule type: EMPTYCHECKDescription: Checking whether the target column has empty values or not.

Null check

Rule type: NULLCHECKDescription: Checking whether the target column has *NULL* values or not.

Date check

Rule type: DATECHECKDescription: Checking whether the target column has **only** DATE values or not.

Integer check

Rule type: INTCHECKDescription: Checking whether the target column has **only** INTEGER values or not

Double check

Rule type: DOUBLECHECKDescription: Checking whether the target column has **only** DOUBLE values or not.

String check

Rule type: STRINGCHECKDescription: Checking whether the target column has **only** STRING values or not.

Mixed datatype check

Rule type: DATATYPECHECKDescription: ---

Syntax

- **<rule_type>** - Fixed key to the rule type
- **<column_name>** - Column to apply the rule
- **<rule_name>** - Custom name of the rule

```
opt.dataset = "example_ds"

val rule = RuleBll.createRule(opt.dataset)
rule.setRuleNm("<rule_name>")
rule.setRuleValue("<column_name>")
rule.setRuleType("<rule_type>")
rule.setPerc(1.0)
rule.setPoints(1)
rule.setIsActive(1)
rule.setUserNm("admin")
```

Outliers

This real life use-case is when you have a large file or data frame with many days of data but you want the run profile to be the current day so that it trends properly overtime. Another nuance to this use-case is that the customer_id is a unique field to the user and it should not show up in the analytics i.e. an outlier. But the customer_id should be available when the user wants to query the rest api end points. The customer_id is then used to link back the users original dataset. A bloomberg_Id (BB_ID) is a common example.

CSV File

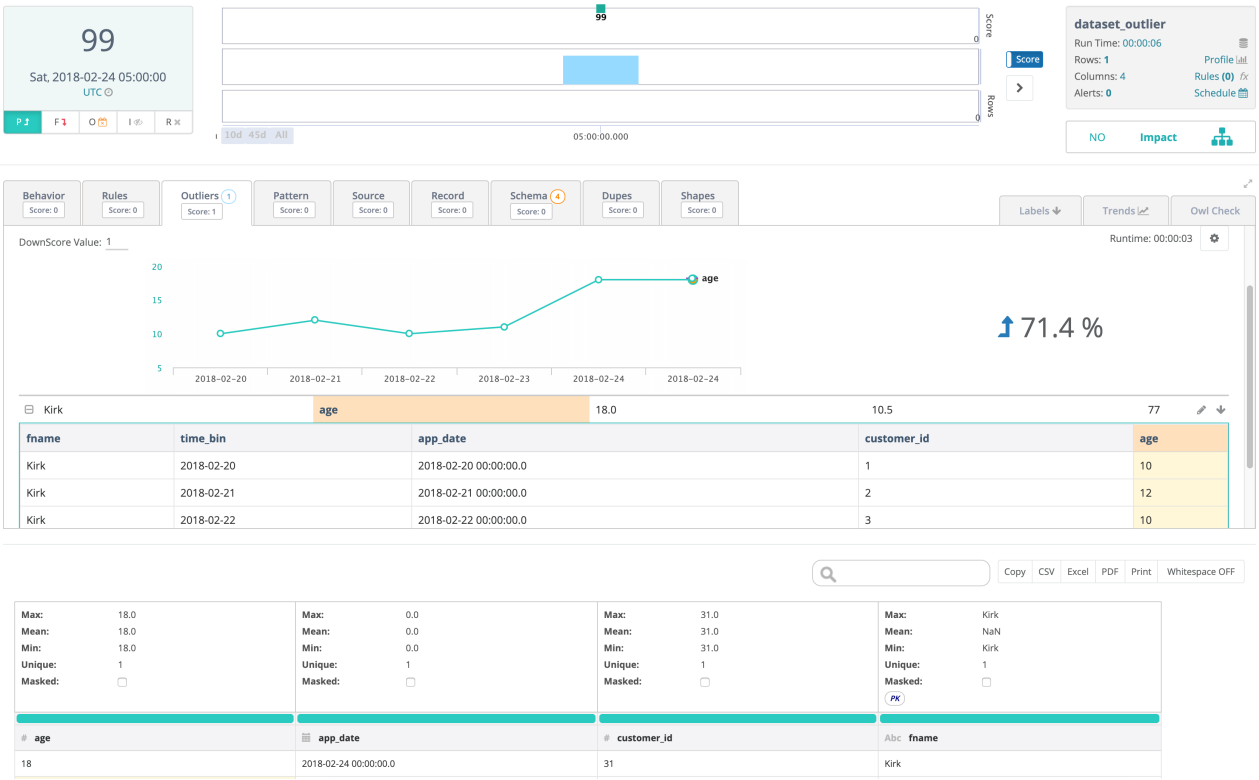
```
fname,app_date,age,customer_id
Kirk,2018-02-24,18,31
Kirk,2018-02-23,11,4
Kirk,2018-02-22,10,3
Kirk,2018-02-21,12,2
Kirk,2018-02-20,10,1
```

Notebook Code (Spark Scala)

```
val filePath = getClass.getResource("/notebooktest.csv").getPath

val spark = SparkSession.builder
  .master("local")
  .appName("test")
  .getOrCreate()

val opt = new OwlOptions()
opt.dataset = "dataset_outlier"
opt.runId = "2018-02-24"
opt.outlier.on = true
opt.outlier.key = Array("fname")
opt.outlier.dateColumn = "app_date"
opt.outlier.timeBin = OutlierOpt.TimeBin.DAY
opt.outlier.lookback = 5
opt.outlier.excludes = Array("customer_id")

val dfHist = OwlUtils.load(filePath = filePath, delim = ",",
sparkSession = spark)
val dfCurrent = dfHist.where(s"app_date = '${opt.runId}' ")

val owl = OwlUtils.OwlContextWithHistory(dfCurrent=dfCurrent,
dfHist=dfHist, opt=opt)
owl.register(opt)
owl.owlCheck()
```

Collibra DQ Web UI

Score drops from 100 to 99 based on the single outlier in the file. Row count is 1 because there is only 1 row in the current data frame. The historical data frame was provided for context and you can see those rows in the outlier drill-in. The customer_id is available in the data preview and can be used as an API hook to link back to the original data set.



After you run an DQcheck using owl.owlcheck you might want to check individual scores to see what type of issues were in the data. Collibra DQ sends back the records with issues in the format of a DataFrame using the notebook cmds or JSON from the REST api.

```
val hoot = owl.hoot

println(s"SHAPE   ${hoot.shapeScore} ")
println(s"DUPE    ${hoot.dupeScore} ")
println(s"OUTLIER ${hoot.outlierScore} ")
println(s"PATTERN ${hoot.patternScore} ")
println(s"RECORD  ${hoot.recordScore} ")
println(s"SCHEMA  ${hoot.schemaScore} ")
println(s"BEHAVIOR${hoot.behaviorScore} ")
println(s"SOURCE  ${hoot.sourceScore} ")
println(s"RULES   ${hoot.ruleScore} ")

if (hoot.shapeScore > 0) {
  owl.getShapeRecords.show
}
if (hoot.dupeScore > 0) {
  owl.getDupeRecords.show
}
```

```
+-------+---------+--------------------+--------+-----------+---
----+-------+
|row_cnt|obs_score|             row_key|obs_type|customer_
id|  fname|owl_id|
+-------+---------+--------------------+--------+-----------+---
----+-------+
|     21|       46|afa89984ce472a409...|    DUPE|         32|   -
Kirk|     1|
|     22|       46|afa89984ce472a409...|    DUPE|         31|Kir-
k's.|     2|
|     23|       60|41ea2d828b1a5fbf2...|    DUPE|         30|   -
 Dan|     3|
|     24|       60|41ea2d828b1a5fbf2...|    DUPE|         27|   -
 Dan|     6|
```

```
+--------------+--------------------+--------+---------+------
--------+--------+-------+-------+---+--------------------+-----
------+-------+------+---------+
|        dataset|              run_id|col_name|col_format|col_
format_cnt|owl_rank|row_cnt|row_key|age|            app_date|-
customer_id|  fname|owl_id|time_bin|
+--------------+--------------------+--------+---------+------
--------+--------+-------+-------+---+--------------------+-----
------+-------+------+---------+
|dataset_outlier|2018-02-24
00:00:...|   fname|   xxxx'x.|             1|       1|      2|xx-
xx'x.| 18|2018-02-24
00:00:...|         31|Kirk's.|     2|    null|
+--------------+--------------------+--------+---------+------
--------+--------+-------+-------+---+--------------------+-----
------+-------+------+---------+
```

GET http://$host/v2/getoutlier?dataset=dataset_outlier&runId=2018-02-24

GetOutlier

Parameters

Path

data set	string	name of data set
data set	string	yyyy-MM-dd format can include time and timezone
Responses		

200		


```
{
    confidence: 77
    dataset: "dataset_outlier"
    keyArr: null
    lb: 0
    outColumn: "age"
    outKey: "Kirk"
    outMedian: "10.5"
    outValue: "18.0"
    runId: "2018-02-24T05:00:00.000+0000"
    ub: 0
    }
```


GET http://$host/v2/getdatashapes?dataset=dataset_outlier&runId=2018-02-24

GetShape

Parameters

Path

data set	string	name of data set
runId	string	yyyy-MM-dd format can include time and timezone
Responses		
200		

Column Match

This example shows how one can get column level match statistics across datasources in an Owl Notebook. Supports exact and fuzzy matching.

Set ColMatch Parameters

```
    %spark
    import com.owl.common.domain._
    import com.owl.common.Props
    import com.owl.core.util.OwlUtils
    import scala.collection.JavaConverters._
    import com.owl.common.Utils
    val c1 = new Connection()
    c1.dataset = "silo.account"
    c1.user = "user"
    c1.password = "pass"
    c1.query = "select id, networth, acc_name, acc_branch from
silo.account limit 200000"
    c1.url = "jdbc:mysql://<db url>:3306"

    val c2 = new Connection()
    c2.dataset = "silo.user_account"
    c2.user = "user"
    c2.password = "pass"
    c2.query = "SELECT acc_name, acc_branch, networth FROM
silo.account limit 200000"
    c2.url = "jdbc:mysql://<db url>:3306"

    val c3 = new Connection()
    c3.dataset = "silo.user_account"
    c3.user = "user"
    c3.password = "pass"
    c3.query = "SELECT acc_name as acc_name2, acc_branch, net-
worth FROM silo.account limit 100000"
    c3.url = "jdbc:mysql://<db url>:3306"

    props.dataset = "colMatchTest1"
    props.runId = "2017-02-04"
    props.connectionList = List(c1,c2,c3).asJava
    props.colMatchBatchSize = 2
    props.colMatchDurationMins = 3
    val owl = OwlUtils.OwlContext(spark.emptyDataFrame, props)
```

Exact Match

```
%spark
props.colMatchLevel = "exact"
owl.register(props)
owl.colMatchDF().show
```

Sample Result

```
+-----------+----------------+---------+---------+---------
-----+
|   dataset_1|        dataset_2|     col_1|     col_2|matchPer-
centage|
+-----------+----------------+---------+---------+---------
-----+
|silo.account|silo.user_account|        id|  acc_
name|              0|
|silo.account|silo.user_account|        id|acc_
branch|              0|
|silo.account|silo.user_
account|        id|  networth|             0|
|silo.account|silo.user_account|        id|    owl_
id|              0|
|silo.account|silo.user_account|  networth|  acc_
name|              0|
|silo.account|silo.user_account|  networth|acc_
branch|             16|
|silo.account|silo.user_
account|  networth|  networth|           100|
|silo.account|silo.user_account|  networth|    owl_
id|              0|
|silo.account|silo.user_account|  acc_name|  acc_
name|             87|
|silo.account|silo.user_account|  acc_name|acc_
branch|              0|
|silo.account|silo.user_account|  acc_
name|  networth|             0|
|silo.account|silo.user_account|  acc_name|    owl_
id|              0|
|silo.account|silo.user_account|acc_branch|  acc_
name|              0|
|silo.account|silo.user_account|acc_branch|acc_
branch|             87|
|silo.account|silo.user_account|acc_
branch|  networth|            12|
|silo.account|silo.user_account|acc_branch|    owl_
id|              0|
|silo.account|silo.user_account|    owl_id|  acc_
name|              0|
|silo.account|silo.user_account|    owl_id|acc_
branch|              0|
|silo.account|silo.user_account|    owl_
id|  networth|             0|
|silo.account|silo.user_account|    owl_id|    owl_
id|              0|
+-----------+----------------+---------+---------+---------
-----+
only showing top 20 rows
```

Fuzzy Match

```
%spark
props.colMatchLevel = "fuzzy"
props.colMatchFuzzyDistance = 4
owl.register(props)
owl.colMatchDF().show
```

Sample Result

```
+------------+-----------------+----------+----------+----------
-----+
|   dataset_1|        dataset_2|     col_1|     col_2|matchPer-
centage|
+------------+-----------------+----------+----------+----------
-----+
|silo.account|silo.user_account|        id|  acc_
name|              5|
|silo.account|silo.user_account|        id|acc_
branch|             27|
|silo.account|silo.user_
account|        id|  networth|             22|
|silo.account|silo.user_account|        id|    owl_
id|              0|
|silo.account|silo.user_account|  networth|  acc_
name|            100|
|silo.account|silo.user_account|  networth|acc_
branch|            233|
|silo.account|silo.user_
account|  networth|  networth|            200|
|silo.account|silo.user_account|  networth|    owl_
id|              0|
|silo.account|silo.user_account|  acc_name|  acc_
name|            162|
|silo.account|silo.user_account|  acc_name|acc_
branch|            262|
|silo.account|silo.user_account|  acc_
name|  networth|             75|
|silo.account|silo.user_account|  acc_name|    owl_
id|              0|
|silo.account|silo.user_account|acc_branch|  acc_
name|            262|
|silo.account|silo.user_account|acc_branch|acc_
branch|            612|
|silo.account|silo.user_account|acc_
branch|  networth|            175|
|silo.account|silo.user_account|acc_branch|    owl_
id|              0|
|silo.account|silo.user_account|    owl_id|  acc_
name|              0|
|silo.account|silo.user_account|    owl_id|acc_
branch|               0|
|silo.account|silo.user_account|    owl_
id|  networth|              0|
|silo.account|silo.user_account|    owl_id|    owl_
id|              0|
+------------+-----------------+----------+----------+----------
-----+
only showing top 20 rows
```

AWS DataBricks

Getting started

First use vanilla spark code to setup connection properties and access a database table via spark jdbc. Entire code example available at the end for copy paste.

Load Spark JDBC DF

```
//--- GCP Postgres Connection ---//
val url = "jdbc:postgresql://owlpostgres.[redacted]us-east-1.rds.amazonaws.com:5432/postgres?currentSchema=owl_test"
var connectionProps = new java.util.Properties()
connectionProps.setProperty("driver", "org.postgresql.Driver")
connectionProps.setProperty("user", "[redacted]")
connectionProps.setProperty("password", "[redacted]")
connectionProps.setProperty("connectionUrl", url)

//--- Load DataFrame From GCP Postgres ---//
val jdbcDF2 = spark.read.jdbc(url, "owl_test.nyse", connectionProps)
jdbcDF2.printSchema
jdbcDF2.cache
val count = jdbcDF2.count
println(count)
```

Schema output, Row Count and Runtime

```
root
 |-- EXCH: string (nullable = true)
 |-- SYMBOL: string (nullable = true)
 |-- TRADE_DATE: date (nullable = true)
 |-- OPEN: decimal(9,3) (nullable = true)
 |-- HIGH: decimal(9,3) (nullable = true)
 |-- LOW: decimal(9,3) (nullable = true)
 |-- CLOSE: decimal(9,3) (nullable = true)
 |-- VOLUME: integer (nullable = true)
 |-- PART_DATE_STR: date (nullable = true)

Row Count: 102,817
Runtime: 00:00:03
```

Next Configure Owl Options and Point to Owl Metastore

This requires that you have imported the Collibra DQ libraries into your notebook or databricks env.

Configure Owl

```
import com.owl.common.options._
import com.owl.core.Owl
import com.owl.core.util.OwlUtils

val opt = new OwlOptions()
//--- Owl Metastore ---//
opt.host = "owlpostgres.[redacted]us-east-1.rds.amazonaws.com"
opt.port = s"5432/postgres?currentSchema=public"
opt.pgUser = "[redacted]"
opt.pgPassword = "[redacted]"
//--- Run Options ---//
opt.dataset = "nyse_notebook_pipeline"
opt.runId = "2018-01-10"
opt.datasetSafeOff = true
opt.profile.semantic = false
opt.profile.correlation = false
opt.profile.histogram = false
opt.dupe.include = Array("EXCH", "SYMBOL")

//OwlUtils.resetDataSource(pgHost, pgPort, pgUser, pgPass, spark)
val owl = com.owl.core.util.OwlUtils.OwlContext(jdbcDF2, opt)
owl.register(opt)
```

Next Run a Profile

Profile

```
val profile = owl.profileDF
profile.show
```

```
+-------------+-----+-------+-----------+--------+-----------+--
----+----+------+-------+-------+------+------+---------+
|       column|nulls|empties|cardinality|is_mixed|mixed_
ratio|
Int|Long|String|Decimal|Boolean|Double|  Date|Timestamp|
+-------------+-----+-------+-----------+--------+-----------+--
----+----+------+-------+-------+------+------+---------+
|
HIGH
|
0|      0|      19159|   false|        0.0|     0|   0|     0|
102817|      0|     0|     0|        0|
|
SYMBOL
|
0
|
0
|
3137
|
false
|
0.0|     0|   0|102817|      0|      0|     0|     0|         0|
|
LOW
|
0|      0|      18845|   false|        0.0|     0|   0|     0|
102817|      0|     0|     0|        0|
|
VOLUME
|
0
|
0
|
25856
|
false
|
0.0|102817|   0|     0|      0|      0|     0|     0|         0|
```

```
|   TRADE_
DATE
|
0
|
0
|
33
|
false
|
0.0|     0|   0|     0|       0|       0|     0|102817|         0|
|
EXCH
|
0
|
0
|
2
|
false
|
0.0|     0|   0|102817|       0|       0|     0|     0|         0|
|
CLOSE
|
0|      0|       15781|   false|         0.0|     0|   0|     0|
102817|       0|     0|     0|         0|
|PART_DATE_
STR
|
0
|
0
|
33
|
false
|
0.0|     0|   0|     0|       0|       0|     0|102817|         0|
|
OPEN
|
0|      0|       16013|   false|         0.0|     0|   0|     0|
102817|       0|     0|     0|         0|
+--------------+-----+-------+-----------+--------+-----------+--
----+----+------+-------+-------+------+------+---------+
```

Next Check for Duplicates

Notice there is a duplicate discovered. NYSE AAN record exists twice in the 10/1/2018. This should not happen as end of day stock data should only have 1 record per stock symbol. Great DQ finding.

Duplicates

```
opt.dupe.on = true
opt.dupe.lowerBound = 99
opt.dupe.include = Array("SYMBOL", "EXCH")

val df1Day = jdbcDF2.where("TRADE_DATE = '2018-01-10' ")
val owl = com.owl.core.util.OwlUtils.OwlContext(df1Day, opt)

val dupes = owl.dupesDF

// dataframe show
dupes.show

// rdd collect
dupes.rdd.collect.foreach(println)

// records linked together for remediation
owl.getDupeRecords.show
```

▸ (10) Spark Jobs
▸ df1Day: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [EXCH: string, SYMBOL: string ... 7 more fields]
▸ dupes: org.apache.spark.sql.DataFrame = [distance: double, records: string ... 2 more fields]

```
+--------+--------------------+--------------+-----+
|distance|             records|recordHashCode|count|
+--------+--------------------+--------------+-----+
|     1.0|[score: 100 ][AAN...|    -154885479|    0|
+--------+--------------------+--------------+-----+

[1.0,[score: 100 ][AAN,NYSE,2042] || [AAN,NYSE,2560],-154885479,0]
+-------+---------+--------------------+--------+----+------+------+
|row_cnt|obs_score|             row_key|obs_type|EXCH|SYMBOL|owl_id|
+-------+---------+--------------------+--------+----+------+------+
|     22|      100|41010a87a8668362a...|    DUPE|NYSE|   AAN|  2560|
|     21|      100|41010a87a8668362a...|    DUPE|NYSE|   AAN|  2042|
+-------+---------+--------------------+--------+----+------+------+
```

Next Scan for Outliers

Notice that KOD.w the camera company Kodak commonly trades at less than 2 pennies and jumps to $2.35. Absolutely an outlier. This was a news event named Kodak coin,

google it.

Outliers

```
opt.outlier.on = true
opt.outlier.lookback = 6
opt.outlier.dateColumn = "TRADE_DATE"
opt.outlier.timeBin = OutlierOpt.TimeBin.DAY
opt.outlier.key = Array("SYMBOL")
opt.outlier.measurementUnit = "VOLUME=100000000,HIGH=0.1,LOW=0.1,OPEN=0.1,CLOSE=0.1"

val df1Day = jdbcDF2.where("TRADE_DATE = '2018-01-10' ")
val owl = OwlUtils.OwlContextWithHistory(dfCurrent = df1Day, dfHist = jdbcDF2, opt = opt)
val outliers = owl.outliersDF
outliers.show
```

```
+-----+------+---------+----------+----------+
|  key|column|    value|prediction|confidence|
+-----+------+---------+----------+----------+
|TPG.E|VOLUME|  23400.0|       0.0|         0|
|MTB-C|VOLUME|      0.0|     100.0|         0|
|KOD.W|  OPEN|     2.35|     0.015|         1|
```

Entire Code Snippet

```
//--- GCP Postgres Connection ---//
val url = "jdb-
c:postgresql://${host}:5432/postgres?currentSchema=owl_test"
var connectionProps = new java.util.Properties()
connectionProps.setProperty("driver", "org.postgresql.Driver")
connectionProps.setProperty("user", "${user}")
connectionProps.setProperty("password", "${pass}")
connectionProps.setProperty("connectionUrl", url)

//--- Load DataFrame From GCP Postgres ---//
val jdbcDF2 = spark.read.jdbc(url, "owl_test.nyse", con-
nectionProps)
jdbcDF2.printSchema
jdbcDF2.cache
jdbcDF2.count

//--- Owl Library Imports ---//
import com.owl.common.options._
import com.owl.core.Owl
import com.owl.core.util.OwlUtils

val opt = new OwlOptions()
//--- Owl Metastore ---//
opt.host = s"${host}"
opt.port = s"5432/postgres?currentSchema=public"
opt.pgUser = s"$user"
opt.pgPassword = s"$pass"

//--- Run Options ---//
opt.dataset = "nyse_notebook_pipeline"
opt.runId = "2018-01-10"
opt.datasetSafeOff = true

opt.dupe.on = true
opt.dupe.lowerBound = 99
opt.dupe.include = Array("SYMBOL", "EXCH")

opt.outlier.on = true
opt.outlier.lookback = 6
opt.outlier.dateColumn = "TRADE_DATE"
opt.outlier.timeBin = OutlierOpt.TimeBin.DAY
opt.outlier.key = Array("SYMBOL")
opt.outlier.measurementUnit = "VOLUME-
E=100000000,HIGH=0.1,LOW=0.1,OPEN=0.1,CLOSE=0.1"

//--- Initialize Owl ---//
val currentDay = jdbcDF2.where(s"TRADE_DATE = '${opt.runId}' ")
val owl = OwlUtils.OwlContextWithHistory(dfCurrent = currentDay,
```

```
                                    dfHist = jdbcDF2, opt = opt)

//--- Pipeline Cmds ---//
owl.register(opt)
val profile = owl.profileDF
val outliers = owl.outliersDF
val dupes = owl.dupesDF
```

Required configuration

- **DataBricks Runtime**: 5.4 (includes Apache Spark 2.4.3, Scala 2.11)
- **Python Version**: 3

Required libraries

- **DQ jar file***Example: owl_core_trunk_jar_with_dependencies.jar __*
- **JDBC driver:** org.springframework:spring-jdbc:4.3.16.RELEASE\
- **Database specific JDBC connector drivers***Example#1: mysql:mysql-connector-java:8.0.17Example#2: org.postgresql:postgresql:jar:42.2.8*

Azure DataBricks

Run a Collibra DQ check on any file in Azure Blob

Read the File by setting up the azure key.

```
spark.conf.set("fs.azure.ac-
count.key.abcCompany.blob.core.windows.net"
,"GBB6Upzj4AxQld7cFv7wBYNoJzIp/WEv/5NslqszY3nAAlsalBNQ==")

val df = spark.read.parquet("wasbs://company-abc@-
abceCompany.blob.core.windows.net/FILE_NAME/20190201_FILE_NAME.-
parquet")
```

Process the file using Collibra DQ

```
// register in Owl Catalog, Optional
val owl = new Owl(df).register

// run a full DQ Check
owl.owlCheck()
```

Additional imports and input options

```
import com.owl.core._
import com.owl.common._

val props = new Props()
props.dataset = datasetName
props.runId = 2019-03-02
props..... // look at the many input options
```

Give your data a little quality time.

Options (base)

API documentation for Java Class DQOptions.

The Class properties are laid out in the following format headers to allow Gitbook indexing.

ClassName field_name

field type | *default* __Description

DQOptions data set

String | _StringUtils.Empty_ __Unique string ID for the DQCheck Data set. Cannot contain ".", "-", "#", "@" \

Example Code

Initializing

Scala

```
import com.owl.common.options.OwlOptions

val opts = new OwlOptions()
```

Java

```
import com.owl.common.options.OwlOptions

OwlOptions opts = new OwlOptions();
```

{% endtab %} {% endtabs %}

Options API

Field mappings

Field name	CLI prop	Description
dataset	*ds*	dataset name, example: userTable or users or user_file
rundId	*rd*	run date, must be in format **yyyy-MM-dd** or for incremental use Hours or Minutes **yyyy-MM-dd HH:mm**
rundIdEnd	*rdEnd*	end date for query ranges **t_date >= $ and t_date < $**, must be in format **yyyy-MM-dd** or for incremental use Hours or Minutes **yyyy-MM-dd HH:mm**
passFail	*passfaillimit*	Limit for passing or failing runs
jobId		
onReadOnly	*readonly*	Do not persist results to the DQ metastore - useful during testing.

Load Options

Field name	CLI prop	Description
fullFile	*fullfile*	use entire file for lookbacks instead of just filequery
fileQuery		
header		
headerSrc		
hasHeader		
isParallel		
isJson		
isMixedJson		
isMapsJson		
flatten		
isMultiLine		
sparkprinc		
sparkkeytab		
jdbcprinc		
jdbckeytab		
srcpwdmgr		
pwdmgr		
pguser	pguser	
pgpassword	pgpassword	

Field name	CLI prop	Description
pghost	host	
pgport	port	
executorcores		
isParquet		
isAvro		
avroSchema		
isXml		
xmlRowTag		
isOrc		
dateFmt		
timeFmt		
datasetSafety		
filePath		
delimiter		
fileLookBack		
dbLookBack		
connectionURL		
userName		
password		
sqlQuery		
connectionProps		

Field name	CLI prop	Description
zkHost	~~~~	***Deprecated***
zkPort	~~~~	***Deprecated***
zkPath	~~~~	***Deprecated***

Outlier Options

Field name	CLI prop	Description
on	dl	Deep learning.
lookback	dllb	A deep learning lookback. For example, a value 5 for a lookback of 5 days. > Note Lookback periods directly correlate to `-dllb`.
key	dlkey	A comma-delimited deep learning key. comma delim key ex: symbol,date
dateField		
bin		
includes	dlinc	A deep learning col limit. For example, open,close,high,volume.
excludes	dlexc	A deep learning col exclusion. For example, open,close,high,volume.
categorical		
by		
limit		
historyLimit		

Field name	CLI prop	Description
minhistory	dlminhist	An automatically generated flag based on the outlier lookback setting `-dllb` that defines the minimum number of days before DQ flags data as potential outliers. `-dlminhist` ensures that the number of days in the algorithm is relative to the total scope of the lookback period. It is not recommended that you override this flag from the command line. > Note Lookback periods do not directly correlate to `-dlminhist`.
score		

FPG Options

Field name	CLI prop	Description
on	fpgon	pattern mining
lookback	fpglb	lookback interval for pattern mining. Ex: **-fpglb 5**
key	fpgkey	natural key for pattern mining activity
dateField	fpgdc	date column for pattern mining. Ex: **-fpgdc date_col**
lowFreq		***Deprecated***
includes	fpginc	pattern mining is expensive use this input to limit the observed cols
excludes	fpgexc	pattern mining is expensive use this input to limit the observed cols
timeBin	fpgtbin	time bin for pattern mining. Ex: **-fpgtbin DAY**
score	fpgscore	score for pattern mining records
minSupport	fpgsupport	
confidence	fpgconfidence	

Dupe Options

Field name	CLI prop	Description
on	dupe	duplicate record detection
includes	dupeinc	duplicate record detection, column inclusion list
excludes	dupeexc	duplicate record detection, column exclusion list
depth		
lowerBound	dupelb	duplicate lower bounds on percent match default [85]
upperBound		
blocksize		
useCache		
checkHeader		
exactMatch		
ignoreCase	dupenocase	duplicate record detection, column exclusion list
score	dupescore	
limit	dupelimit	Limit for dupe rows stored

Profile Options

Field name	CLI prop	Description
on		
includes		
excludes		
dataShapeOn		

Field name	CLI prop	Description
statsOn		
correlationOn		
histogramOn		
cardinalityOn		
dataShapeColsInc		
dataShapeColsExc		

Source Options

Field name	CLI prop	Description
on		
includes		
excludes		
key		
fileQuery		
map		
score		
datasetSrc		
driverSrc		
userNameSrc		
passwordSrc		
connectionURLSrc		

Field name	CLI prop	Description
sqlQuerySrc		
connectionPropsSrc		

Rule Options

Field name	CLI prop	Description
on		
rulesOnly		
semantic		

ColMatch Options

Field name	CLI prop	Description
colMatchParallelProcesses		
colMatchDurationMins		
colMatchBatchSize		
connectionList		

Spark Options

Field name	CLI prop	Description
numExecutors		
executorMemory		
driverMemory		

Field name	CLI prop	Description
executorCores		
master		
jars		
libs		
driver		

Misc Options

Field name	CLI prop	Description
obslimit		
nullValue		

Classes

Load

```java
package com.owl.common.options;

import org.apache.commons.lang3.StringUtils;

import java.util.Properties;

/**
 * Owl Options related to data loading
 */
public class LoadOpt {
    // Options order: "unsorted",
    // "dataset scope columns", "dataset scope rows", "look
back",
    // "common options for both data sources", "file as data
source", "db as data source"

    public static final String SINGLE_QUOTE = "'";
    public static final String DOUBLE_QUOTE = "\"";
    public static final String BACK_TICK = "`";

    /**
     * If true, don't save any metadata
     * TODO confirm if this is correct
     */
    public Boolean readonly = false;

    /**
     * The Password manager.
     */
    public String passwordManager = null;

    /**
     * Catalog alias (Catalog name)
     */
    public String alias = StringUtils.EMPTY;

    // --- Dataset Scope Column specifications ------- //
    // Properties that select columns for Dataset activities or
modifies (data type or new columns)
    // prior and/or during loading into Spark DF
    /**
     * Dataset scope query. (IMPORTANT)
     * The query should contain all the columns necessary to run
the activities.
     * TODO: figure out if this gets used when using files
     */
    public String query = StringUtils.EMPTY;
```

```
    /**
     * Concatenated column names (sep = ",") for columns that
are keys
     * TODO: confirm
     */
    public String key = StringUtils.EMPTY;

    /**
     * SELECT expression to transform expressions with assign-
ment by "=" and delimited by "|".
     * e.g. colname=cast(colname as string)|colname2=colname2
(cast as date)
     */
    public String expression = StringUtils.EMPTY;

    /**
     * Add "OWL_RUN_ID" UNIX timestamp (s) column to Spark DF
usng the OwlOptions.runId.
     * Does not obey timeStampDivisor (timestamp in seconds
because Spark)
     */
    public Boolean addDateColumn = false;

    /**
     * Fill null values in Spark DF with 0 (numeric columns
only)
     */
    public Boolean zeroFillNull = false;

    /**
     * A string that indicates a null value; any value matching
this string will be set as nulls in the Spark DF
     * Default: "" -> NULL
     * Example: 'null' -> NULL
     * --
     * Note: to emptyStirngFillNull (replace String column null
-> "", use expression
     */
    public String replaceNulls = StringUtils.EMPTY;

    /**
     * All data types forced to strings for type safe pro-
cessing.
     * Not implemented in activity (yet)
     */
    public Boolean stringMode = false;

    // --- Dataset Scope Row specifications ------- //
```

```
    // Properties that filter rows for Dataset activities
    // prior and/or during loading into Spark DF
    /**
     * Convert row into string and only use rows containing this
value.
     * Strict matching only.
     */
    public String filter = StringUtils.EMPTY;

    /**
     * Convert row into string and only use rows containing this
value.
     * Strict matching only.
     */
    public String filterNot = StringUtils.EMPTY;

    // --- Look back ------- //
    // For Look back feature
    /**
     * Build up history of OwlChecks. Does not include current
OwlCheck.
     * TODO: Document the relationship with unionLookBack
     */
    public Integer backRun = null;

    /**
     * Whether to load data for looking back in history.
     * How much historical data to load is based on Out-
lierOpt.lookback and PatternOpt.lookback.
     */
    public Boolean unionLookBack = false;

    // --- Shared Data Loading Options ------- //
    // Properties that affect data loading & pre-processing for
both files and db as source
    /**
     * Whether to use cached data for activities
     */
    public Boolean cache = true;

    /**
     * The year, month, and day format of date columns in the
dataset for loading the data only.
     * Default = "yyyy-MM-dd"
     */
    public String dateFormat = "yyyy-MM-dd";

    /**
```

```
     * The hour, minute, second, and milisecond format of date
columns in the dataset for loading the data only/
     * Default = "HH:mm:ss.SSS"
     * Not used. Questionably why separate timeFormat variable
exists when dateFromat can represent hms as well.
     */
    public String timeFormat = "HH:mm:ss.SSS";

    /**
     * Whether to convert date columns (specified by activity
opts) in dataset
     * into timestamp in ms (to make it seconds, set Props.-
timeStampDivisor = "s")
     * TODO: Needs LoadOpt.timeStampDivisor and fix Utils.scala
date2Timestamp
     */
    public Boolean timestamp = false;

    /* TODO add timeStampDivisor here and map between owl props?
     public String timeStampDivisor = "ms"
     */

    // --- Using file as data source ------- //
    // Properties that control where & how static file is read
    /**
     * Full path to the file.
     * If hdfs, then "hdfs://...".
     * If s3, then "s3://...", "s3a://...", or "s3n://...".
     * If parquet, then "...parquet" or "...PARQUET"
     */
    public String filePath = StringUtils.EMPTY;

    /**
     * SQL query used on file.
     * owl_id is added if not included in select clause.
     * If empty, then defaults to full file query.
     * (Does not update LoadOpts.fullFile to true).
     */
    public String fileQuery = StringUtils.EMPTY;

    /**
     * Whether to use full file (i.e. use all columns) on data
load
     */
    public Boolean fullFile = false;

    /**
     * File column names, comma separated
```

```
     */
    public String fileHeader = null;

    /* TODO checkHeader needs to be moved here from DupeOpt
     public Boolean checkHeader = true;*/

    /**
     * Whether to have Spark infer the schema of data source
     * If props.profile2 == true, this is overwritten to false!
     * If xml file, this is ignored and schema is always
inferred by Spark on xml data load.
     * If avro file, this value is respected (but may get over-
written by props.profile2)
     * (see activity2.Load.file)
     */
    public Boolean inferSchema = true;

    /**
     * Sample without replacement from file. Valid value is a
fraction [0, 1.0].
     * Only affects when filetype is xml or unspecified (and
therefore assumed to be delimited table)
     */
    public Double sample = 1.0;

    /**
     * Filetype (avro, json, orc, parquet, xml). Unspecified
file
     */
    public FileType fileType = null;

    /**
     * Delimiter for file. If number of characters after repla-
cing "\" with "" is 2 or more character
     * (e.g. compound delimiters like \t\t), then defaults to
"\t" and attempts to read file as tsv
     * See Activity2.load.file for details
     */
    public String delimiter = ",";

    /**
     * File character encoding
     */
    public String fileCharSet = "UTF-8";

    /**
     * The Avro schema for relevant avro file. Ignored if empty
string
```

```
     */
    public String avroSchema = StringUtils.EMPTY;

    /**
     * The Xml row tag for xml file. Ignored if empty string.
     */
    public String xmlRowTag = StringUtils.EMPTY;

    /**
     * Whether to flatten arrays in nested schema
     * TODO explain better. Does this only affect JSON file?
     */
    public Boolean flatten = false;

    /**
     * Whether data contains maps in json that requires extra
handling"
     * TODO explain better. Does this only affect JSON file?
     */
    public Boolean handleMaps = false;

    /**
     * Whether to handle mixed json.
     * TODO explain better. Does this only affect JSON file?
     */
    public Boolean handleMixedJson = false;

    /**
     * Spark.read option multiline, for JSON file only
     */
    public Boolean multiLine = false;

    // --- Using database as data source ------ //
    /**
     * Path to DB Driver. (e.g. /opt/owl/driver/postgres)
     */
    public String lib = StringUtils.EMPTY;

    /**
     * DB Driver name (Java namespace, e.g. org.-
postgresql.Driver).
     * Leave as null (default) and LoadOpts.connectionURL will
resolve the driver name.
     */
    public String driverName = null;

    /**
```

```
     * Connections name in metastore DB (pub-
lic.connections.aliasname).
     * Does not refer to the "name" of the database. Refers to
"aliasname" that the user set when
     * uploading connection config to Owl.
     */
    public String connectionName = StringUtils.EMPTY;

    /**
     * The Connection url, prefixed by jdbc.
     * e.g. "jdbc:postgresql://localhost:5432"
     */
    public String connectionUrl = StringUtils.EMPTY;

    /**
     * DB username
     */
    public String userName = StringUtils.EMPTY;

    /**
     * DB password
     */
    public String password = StringUtils.EMPTY;

    /**
     * JDBC Connection properties (e.g. fetchsize)
     */
    public Properties connectionProperties = null;

    /**
     * Whether data source is Hive Native (not using JDBC)
     * TODO: Why is the default null as opposed to false?
     */
    public Boolean hiveNative = null;

    /**
     * Whether data source is Hive Hadoop Web Cluster (not using
JDBC)
     */
    public Boolean hiveNativeHWC = false;

    // --- Parallel JDBC ------- //
    /**
     * When running parallel JDBC, use LoadOpts.query and OwlOp-
tions.dataset as base table
     */
    public Boolean useSql = true;
```

```
    /**
     * When running parallel JDBC, specify column name
     * ?? Activity2.Load and web has hard-coded magic string
"OWLAUTOJDBC"
     */
    public String columnName = null;

    /**
     * When running parallel JDBC, the upper bound for partition
column.
     * (e.g. "1000000")
     */
    public String lowerBound = null;

    /**
     * When running parallel JDBC, the upper bound for partition
column.
     * (e.g. "5000000")
     */
    public String upperBound = null;

    /**
     * When running parallel JDBC, the number of partitions
used.
     * If 0, then numPartitions used is based on the number of
available Spark Executor (1/2 ~ 2/3)
     * If > 20, then overwritten to 20 (no more than 20 con-
current connections to a database on a single dataset)
     */
    public Integer numPartitions = 0;

    // --- SQL Query properties ---------- //
    // TODO: does this effect DB as source or file as source as
well?
    /**
     * Whether the escape character would be back tick (`).
     * Ignored if escapeCharacter is non-empty (if using
OwlCheck from Options).
     * Marked as true if props.escapeCharacter is a tick
     * (to preserve bijection between props and opts, and vice
versa).
     */
    public Boolean escapeWithBackTick = false;
    /**
     * Whether the escape character would be single quote (').
     * Ignored if escapeCharacter is non-empty (if using
OwlCheck from Options).
     * Marked as true if props.escapeCharacter is a tick
```

```
     * (to preserve bijection between props and opts, and vice
versa).
     */
    public Boolean escapeWithSingleQuote = false;
    /**
     * Whether the escape character would be double quote (").
     * Ignored if escapeCharacter is non-empty(if using OwlCheck
from Options).
     * Marked as true if props.escapeCharacter is a tick
     * (to preserve bijection between props and opts, and vice
versa).
     */
    public Boolean escapeWithDoubleQuote = false;

    /**
     * Specify custom escape character. This takes precedence
over all other escapeWithXYZ options.
     * i.e. if non-empty, then other escapeWithXYZ options are
ignored.
     * If empty (default), no escaping attempt is made (and SQL
query may fail if it contains reserved word)
     *
     * @deprecated Access level of this field will be changed to
private. Please use {@link #setEscapeCharacter(String)} instead.
     */
    @Deprecated
    public String escapeCharacter = StringUtils.EMPTY;


    /**
     * The enum File type.
     */
    public enum FileType {
        /**
         * Avro file type.
         */
        avro,
        /**
         * Json file type.
         */
        json,
        /**
         * Orc file type.
         */
        orc,
        /**
         * Parquet file type.
         */
        parquet,
```

```
        /**
         * Xml file type.
         */
        xml
    }
}
```

Profile

```
public class ProfileOpt {

    public Boolean on = true;          //Whether to run profile
    public Boolean only = false;       //Whether to run only pro-
file
    public String[] include;           //Which columns to include
    public String[] exclude;           //Which columns to exclude
    public Boolean shape = true;       //Disable shape detection
    public Boolean correlation = null;//On/Off Pearsons Cor-
relation, null=auto
    public Boolean histogram = null;  //On/Off Histograming,
null=auto
    public Boolean semantic = null;    //On/Off Semantic dis-
covery, null=auto

    public Integer limit = 300;
    public Integer histogramLimit = 0;
    public Double score = 1.0;          //downscore points per
Shape issue
    public Integer shapeTotalScore = 0;
    public Double shapeSensitivity = 0.00;
    public Integer shapeMaxPerCol = 0;
    public Integer shapeMaxColSize = 0;
    public String behavioralDimension = StringUtils.EMPTY;
    public String behavioralDimensionGroup = StringUtils.EMPTY;
    public String behavioralValueColumn = StringUtils.EMPTY;
    public Boolean behaviorScoreOff = false;  // disable beha-
vior scoring
```

Dupe

```
package com.owl.common.options;

/**
 * Options for Dupe Activity
 */
public class DupeOpt {

    /**
     * Whether to run Dupe Activity
     */
    public Boolean on = false;           // --dupe

    /**
     * @deprecated Unused for Activity2
     */
    public Boolean only = false;         // --dupeonly

    /**
     * Column names to include Dupe Activity
     */
    public String[] include;             // -dupeinc

    /**
     * Column names to exclude Dupe Activity
     */
    public String[] exclude;             // dupeexc

    /**
     * Indicator for complexity. See Activ-
ity2.Dupe.Scala.execute()
     * depth == 0 : exact match (sets props.dupeExactMatch =
TRUE downstream)
     */
    public Integer depth = 2;            // -depth

    /**
     * The minimum dupe scores between two duplicates. (cur-
rently calculated as "edit distance", out of upperBound)
     * Two values with dupe score less than this is lowerBound
are not duplicates (i.e. "truly" different values)
     */
    public Integer lowerBound = 80;      // -dupelb, -dupecutoff

    /**
     * The maximum possible dupe score for duplicate records
(for a given dupe detection method).
     * Currently assumed to be 100.
     */
```

```
    public Integer upperBound = 100;    // -dupeub, -dupeper-
matchupperlimit

    /**
     * Approximate dupe score used to create block index (when
DF is large)
     */
    public Integer approximate = 1;     // -dupeapprox

    /**
     * Number of observations per unique duplicate
     */
    public Integer limitPerDupe = 15;

    /**
     * Whether to process column headers when data load uses
manual column names (LoadOpts.fileHeader)
     * TODO this belongs in LoadOpts, not DupeOpts
     */
    public Boolean checkHeader = true;

    /**
     * TODO remove
     *
     * @deprecated not used;
     */
    public String filter;

    /**
     * If true, dupe activity is case insensitive. If false,
dupe activity is case sensitive.
     * Convenience feature for upper and lower set to 100
     */
    public Boolean ignoreCase = false;      //-dupenocase

    /**
     * Number of points each duplicate contributes to the total
schema score (in Hoot)
     */
    public Double score = 1.0;                //-
dupescore  points per duplicate found default 1

    /**
     * Number of unique duplicates to compute during dupe activ-
ity
     */
    public Integer limit = 300;               //-
dupelimit  default 300
```

Source

```
public class SourceOpt {

    public Boolean on = false;                //-vs
    public Boolean only = false;              //-sourceonly
    public Boolean validateValues = false;    //-validatevalues
    public Boolean matches = false;           //-matches

    public String[] include;                 //-valinc
    public String[] exclude;                 //-valexc
    public String[] key;                      //valkey
    public Map<String, String> map;           //
    public Double score = 1.0;                // points per val-
idate source found, default 1-5
    public Integer limit = 30;                //-valsrclimit
    public String delimiter = ",";            //-srcdel
    public String fileCharSet = "UTF-8";      //-srcencoding
    public String filePath = StringUtils.EMPTY; //--srcfile

    public String header = null;              //-srcheader
    public String dataset = StringUtils.EMPTY;
    public String driverName = StringUtils.EMPTY;
    public String user = StringUtils.EMPTY;
    public String password = StringUtils.EMPTY;
    public String passwordManager = StringUtils.EMPTY;
    public String connectionName = StringUtils.EMPTY;
    public String connectionUrl = StringUtils.EMPTY;
    public String query = StringUtils.EMPTY;
    public String fileQuery = StringUtils.EMPTY;
    public String lib = StringUtils.EMPTY;
    public Properties connectionProperties;
```

FAQs

How to specify OWL database connection properties

OwlOptions

```
import com.owl.common.options.OwlOptions

val opt = new OwlOptions()
opt.dataset = "<dataset_name>"
opt.runId = "<date>"  // YYYY-MM-DD
opt.load.pguser = "<db_username>"
opt.load.pgpassword = "<db_password>"
opt.load.pghost = "<ip>:<port>/<database_name>"
```

Props

```
import com.owl.common.Props

val props = new Props()
props.dataset = "<dataset_name>"
props.runId = "<date>"  // YYYY-MM-DD
props.pguser = "<db_username>"
props.pgpassword = "<db_password>"
props.host = "<ip>:<port>/<database_name>"
```

Command Line

Scale + Data Science

Where Scale meets Data Science. Scale linearly with your data by adding executors and/or memory.

```
-f "file:///Users/home/salary_data.csv" \
-d "," \
-rd "2018-01-08" \
-ds "salary_data"
-numexecutors 2 \
-executormemory 2g
```

Yarn Master

If CollibraDQ is run on an edge node on a popular hadoop distribution such as HDP, CDH, EMR it will automatically register the jobs with Yarn Resource Manager.

Spark Master

DQ also runs using spark master by using the -master input and passing in spark:url.

Spark Standalone

DQ runs in standalone most but naturally will not distribute the processing beyond the hardware it was activated on.

Options	Description
deploymode	spark deploymode option
drivermemory	driver memory example 3G for local space
executorcores	spark executor cores
executormemory	spark executor memory option example 3G
master	overrides local[*], i.e. spark://myhost:7077, yarn-client, yarn-cluster
sparkprinc	kerberos principal name ex: owl@OWL.COM

Use Spark-Submit directly bypassing DQCheck

```
spark-submit \
--driver-class-path /opt/owl/drivers/postgres42/postgresql-
42.2.4.jar \
--driver-library-path /opt/owl/drivers/postgres42/postgresql-
42.2.4.jar \
--driver-memory 3g --num-executors 2 --executor-memory 1g \
--master spark://Kirks-MBP.home:7077 \
--class com.owl.core.cli.OwlCheck /opt/owl/bin/owl-core-trunk-
jar-with-dependencies.jar \
-u user -p pass -c jdbc:postgresql://xyz.chzid9w0hpyi.us-east-
1.rds.amazonaws.com/postgres \
-ds accounts -rd 2019-05-05 -dssafeoff -q "select * from
accounts"
-driver org.postgresql.Driver -lib /opt/owl/drivers/postgres42/
```

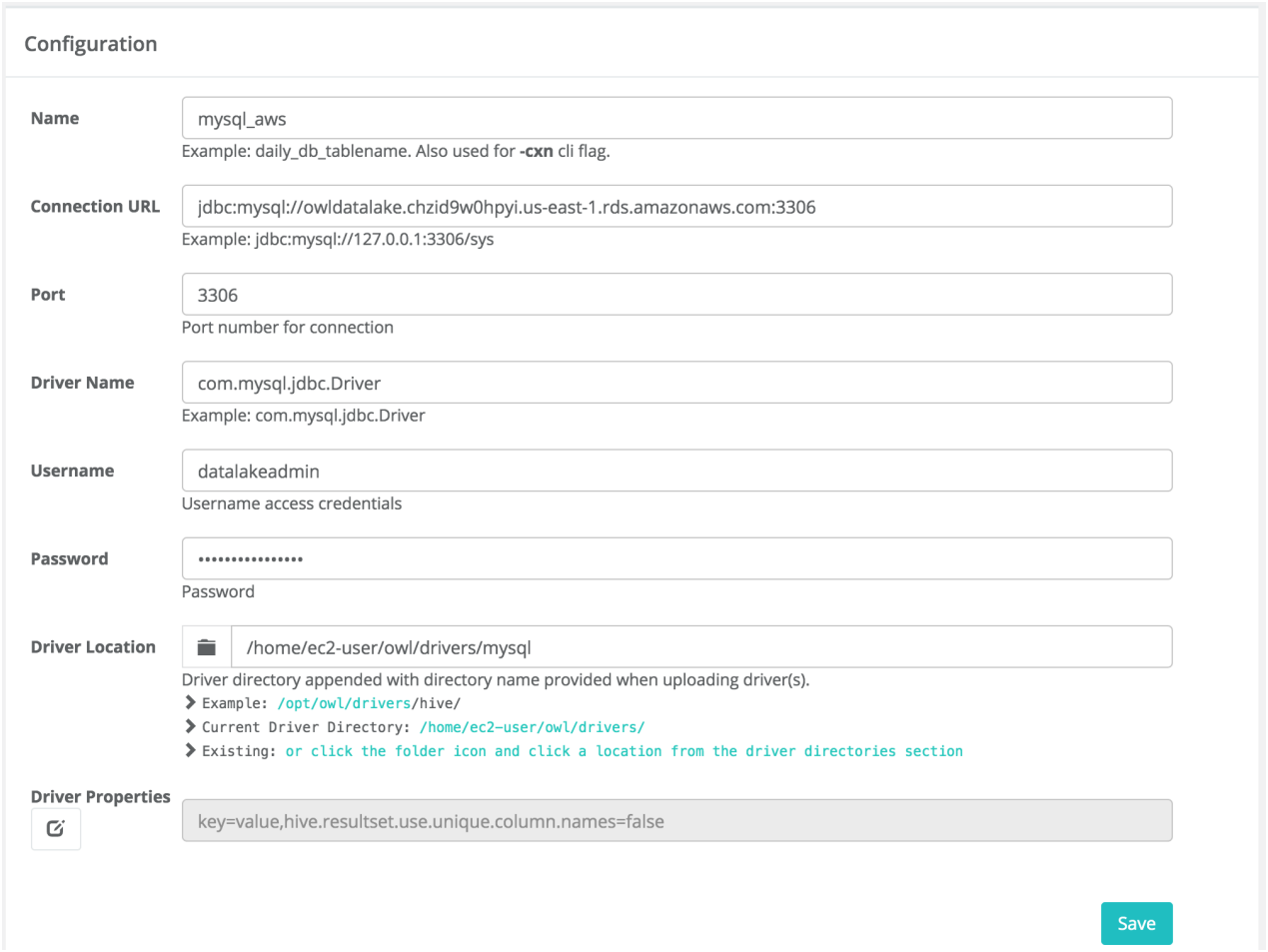
Parallel JDBC Spark-Submit

```
spark-submit \
--driver-class-path /opt/owl/drivers/postgres42/postgresql-
42.2.4.jar \
--driver-library-path /opt/owl/drivers/postgres42/postgresql-
42.2.4.jar \
--conf spark.driver.extraJavaOptions=-Dlo-
g4j.configuration=file:///opt/owl/config/log4j-TRACE.properties
\
--conf spark.executor.extraJavaOptions=-Dlo-
g4j.configuration=file:///opt/owl/config/log4j-TRACE.properties
\
--files /opt/owl/config/log4j-TRACE.properties \
--driver-memory 2g --num-executors 2 --executor-memory 1g --mas-
ter spark://Kirks-MBP.home:7077  \
--class com.owl.core.cli.OwlCheck /opt/owl/bin/owl-core-trunk-
jar-with-dependencies.jar \
-u us -p pass -c jdbc:postgresql://xyz.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com/postgres \
-ds aumdt -rd 2019-05-05 -dssafeoff -q "select * from aum_dt" \
-driver org.postgresql.Driver -lib
/opt/owl/drivers/postgres42/  \
-connectionprops fetchsize=6000 -master spark://Kirks-
MBP.home:7077 \
-corroff -histoff -statsoff \
-columnname updt_ts -numpartitions 4 -lowerbound 1557597987353 -
upperbound 1557597999947
```

DQ Job JDBC

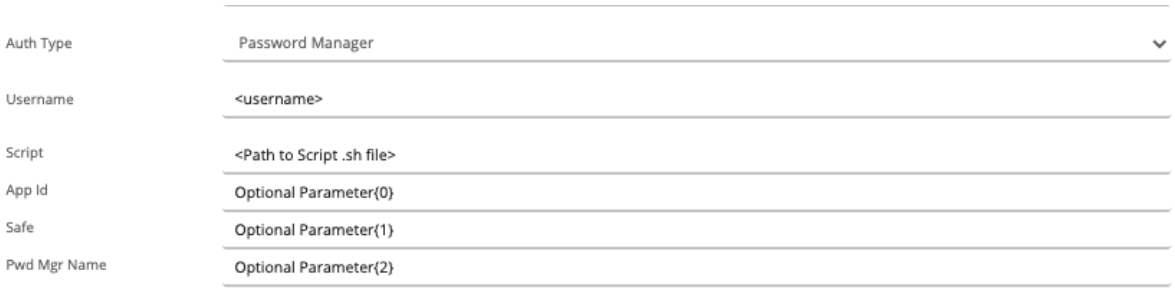
Connect to any database via JDBC.

```
-q "select * from lake.stock_eod where date = '2017-01-20' " \
-u username -p password \
-c "jdbc:mysql://instance.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com:3306" \
-rd "2017-01-20" \
-dc "date" \
-ds "stocks" \
-driver com.mysql.jdbc.Driver \
-lib "/home/ec2-user/owl/drivers/mysql/"
```



Password Manager

You can configure Collibra DQ to call a script file to retrieve a password from a password manager vault or other storage container. The customer is responsible for generating a script to pull just the password and DQ uses that value dynamically when the connection is needed for the UI or when kicking off an DQCheck.



In the connection dialog, select Password Manager from the Auth Type dropdown, and supply a user name. The script is the path to the .sh script on the machine where the web application is running, and the user account that runs DQ-web should be allowed to execute the script. You can either use the optional parameters or pass any parameters your script needs directly inline on the Script value.

Fetch Size

It is important to consider the drivers fetch size when loading greater than 1 Million rows across the network. DQ allows you to set this driver property in the WebApp but this is only for web interaction therefore "fetchsize" will not help here. DQ also allows fetchsize in the DQCheck by passing in a connection property.

CMD line

```
-connectionprops "fetchsize=3000"
```

Notebook

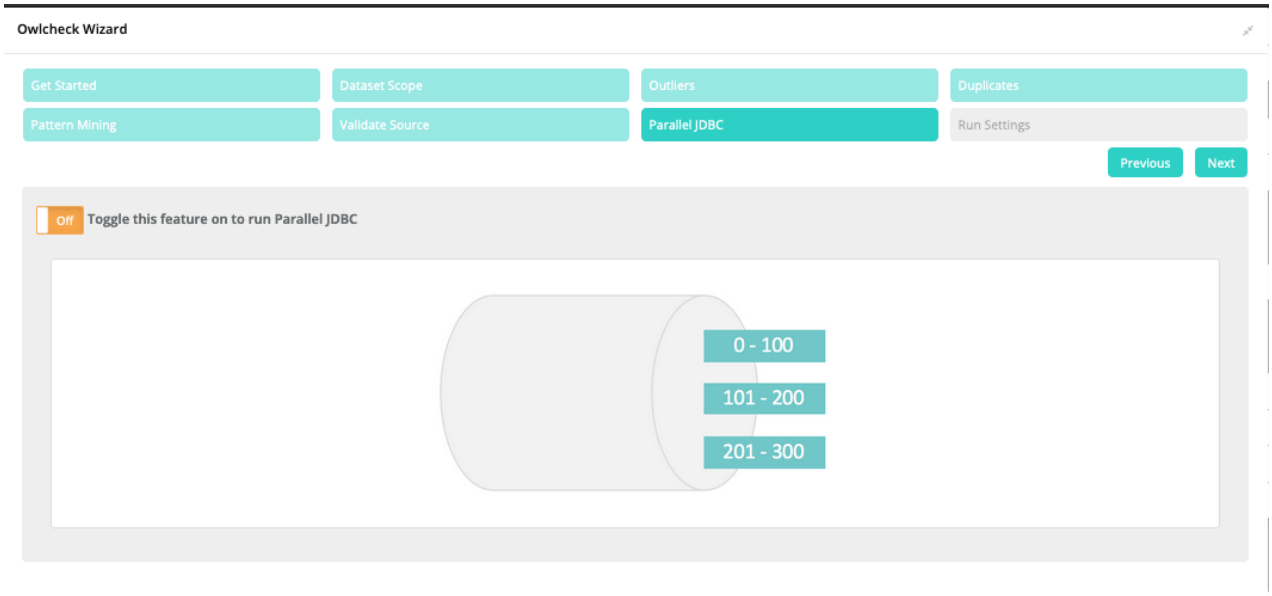
```
props.connectionProps.put("fetchsize", "3000")
```

Parallel JDBC

For greater performance or moving large datasets across a network DQ supports parallel JDBC, which can be enabled by passing `numpartitions` to DQCheck. This can be a 2-5X improvement in many cases.

```
-lib "/opt/owl/drivers/mysql8/"
-cxn mysql
-q "select * from lake.nyse where trade_date = '${rd}' "
-rd 2018-01-01
-ds nyse
-columnname volume
-numpartitions 4
-lowerbound "0"
-upperbound "5000000000"
-usesql
```

DQ also supports auto parallelization, which will configure the `numPartitions` parameter for you based on the size of your data. This is enabled in the UI when you create a dataset using the DQCheck wizard.



DQ Job BigQuery

Example CMD Line

```
-lib "/opt/owl/drivers/bigquery/bigquery/core/" \
-h <IP_ADDRESS>:5432/postgres \
-master spark://<SPARK_MASTER>:7077 \
-ds samples.loan_customer \
-deploymode client \
-q "select * from samples.loan_customer" \
-rd "2021-08-02" \
-driver "com.simba.googlebigquery.jdbc42.Driver" \
-cxn BigQuery
```

Steps for the BigQuery Connection

1. **We would use this Simba driver**: com.simba.googlebigquery.jdbc42.Driver.
2. **We would make an owl-gcp.json** (your org auth key in JSON format).
3. **We would create a JDBC connection** (for example only do not use this JDBC URL): jdbc:bigquery://https://www.googleapis.com/bigquery/v2:443;ProjectId=;OAuthType=0;OAuthServiceAcctEmail=<1234567890>-compute@developer.gserviceaccount.com;OAuthPvtKeyPath=/opt/ext/owl-gcp.json;Timeout=86400
4. **Requires a path to a JSON file** that contains the service account for authorization. That same file is provided to the Spark session to make a direct to storage connection for maximum parallelism once Core fires up."

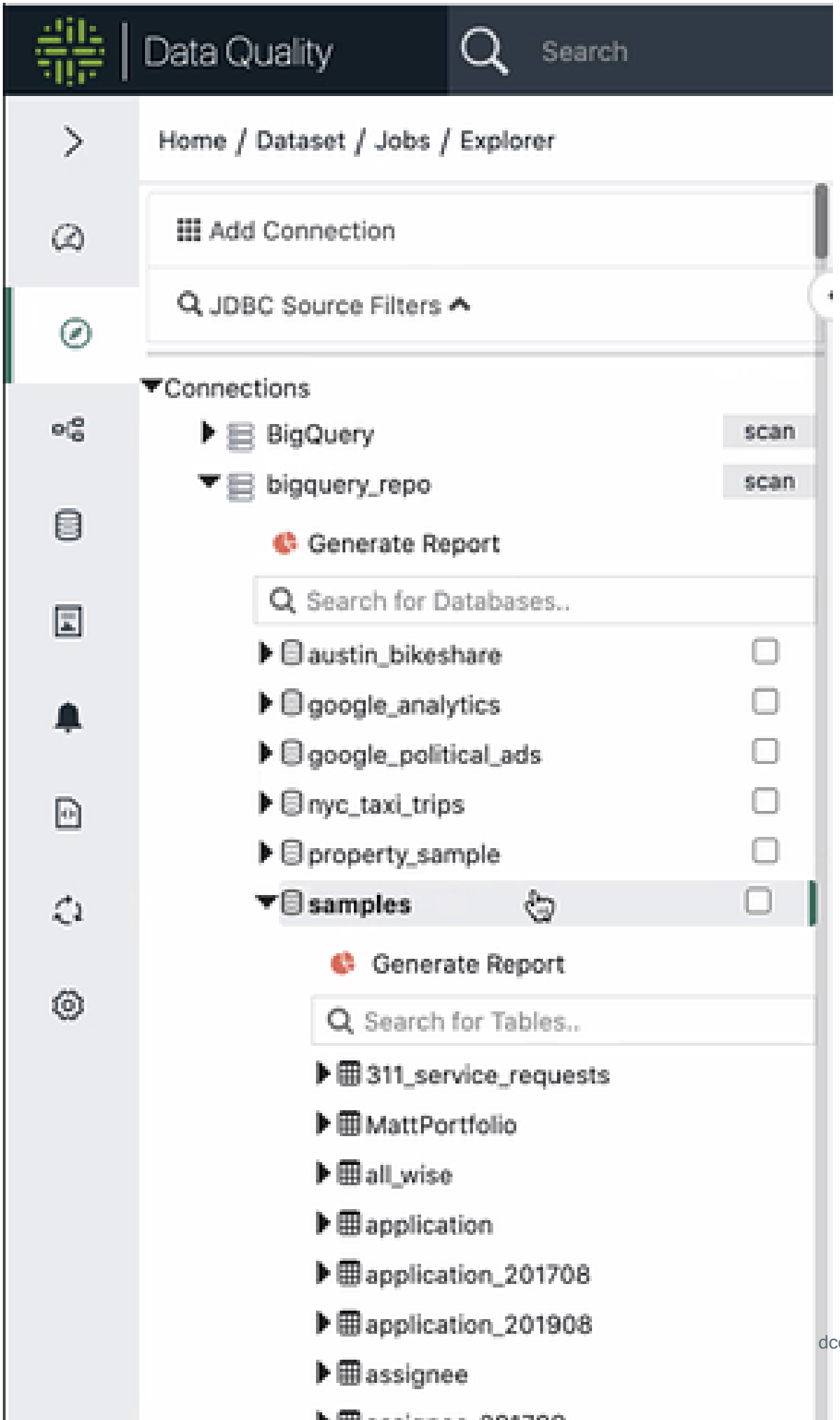
The above and explained there are actually a number of others steps which must be performed to achieve success:

1. **Password for the BigQuery Connector form in Collibra DQ must be a base64 encoded string created from the json file (see step 3. above)** and input as password. For example: `base64 your_json.json -w 0` or `cat your_json.json |`

```
base64 -w 0
```

2. **Check that this JARs exists and is on the path of the Collibra DQ Web UI server** (eg. <INSTALL_PATH>/owl/drivers/bigquery/core). Look at your driver directory location which contains this BigQuery JAR: spark-bigquery_2.12-0.18.1.jar
3. **Make sure there are all the needed JARs present in <INSTALL_PATH>/owl/drivers/bigquery/:** ****animal-sniffer-annotations-1.19.jargoogle-api-services-bigquery-v2-rev20201030-1.30.10.jargrpc-google-cloud-bigquerystorage-v1beta1-0.106.4.jarlistenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jarannotations-4.1.1.4.jargoogle-auth-library-credentials-0.22.0.jargrpc-google-cloud-bigquerystorage-v1beta2-0.106.4.jaropencensus-api-0.24.0.jarapi-common-1.10.1.jargoogle-auth-library-oauth2-http-0.22.0.jargrpc-grpclb-1.33.1.jaropencensus-contrib-http-util-0.24.0.jarauto-value-annotations-1.7.4.jarGoogleBigQueryJDBC42.jargrpc-netty-shaded-1.33.1.jarperfmark-api-0.19.0.jaravro-1.10.0.jargoogle-cloud-bigquery-1.125.0.jargrpc-protobuf-1.33.1.jarprotobuf-java-3.13.0.jarchecker-compat-qual-2.5.5.jargoogle-cloud-bigquerystorage-1.6.4.jargrpc-protobuf-lite-1.33.1.jarprotobuf-java-util-3.13.0.jarcommons-codec-1.11.jargoogle-cloud-core-1.93.10.jargrpc-stub-1.33.1.jarproto-google-cloud-bigquerystorage-v1-1.6.4.jarcommons-compress-1.20.-jargoogle-cloud-core-http-1.93.10.jargson-2.8.6.jarproto-google-cloud-bigquerystorage-v1alpha2-0.106.4.jarcommons-lang3-3.5.jargoogle-http-client-1.38.0.jarguava-23.0.jarproto-google-cloud-bigquerystorage-v1beta1-0.106.4.jarcommons-logging-1.2.jargoogle-http-client-apache-v2-1.38.0.jarhttpclient-4.5.13.jarproto-google-cloud-bigquerystorage-v1beta2-0.106.4.jarconscrypt-openjdk-uber-2.5.1.jargoogle-http-client-appengine-1.38.0.jarhttpcore-4.4.13.jarproto-google-common-protos-2.0.1.jarcoregoogle-http-client-jackson2-1.38.0.jarj2objc-annotations-1.3.jarproto-google-iam-v1-1.0.3.jarerror_prone_annotations-2.4.0.jargoogle-oauth-client-1.31.1.jarjackson-annotations-2.11.0.jargrpc-alts-1.33.1.jarjackson-core-2.11.3.jarslf4j-api-1.7.30.jarfailureaccess-1.0.1.jargrpc-api-1.33.1.jarjackson-data-bind-2.11.0.jargax-1.60.0.jargrpc-auth-1.33.1.jarjavax.annotation-api-1.3.2.jarthreetenbp-1.5.0.jargax-grpc-1.60.0.jargrpc-context-1.33.1.jarjoda-time-2.10.1.jargax-httpjson-0.77.0.jargrpc-core-1.33.1.jarjson-20200518.jargoogle-api-client-1.31.1.jargrpc-google-cloud-bigquerystorage-v1-1.6.4.jarjsr305-3.0.2.jar*
4. You may get a CLASSPATH conflict regarding the JAR files.

5. Make sure the BigQuery connector Scala version matches your Spark Scala. version.


Data Quality
Search
Home / Dataset / Jobs / Explorer
Add Connection
JDBC Source Filters
Connections
BigQuery
scan
bigquery_repo
scan
Generate Report
Search for Databases..
austin_bikeshare
google_analytics
google_political_ads
nyc_taxi_trips
property_sample
samples
Generate Report
Search for Tables..
311_service_requests
MattPortfolio
all_wise
application
application_201708
application_201908
assignee

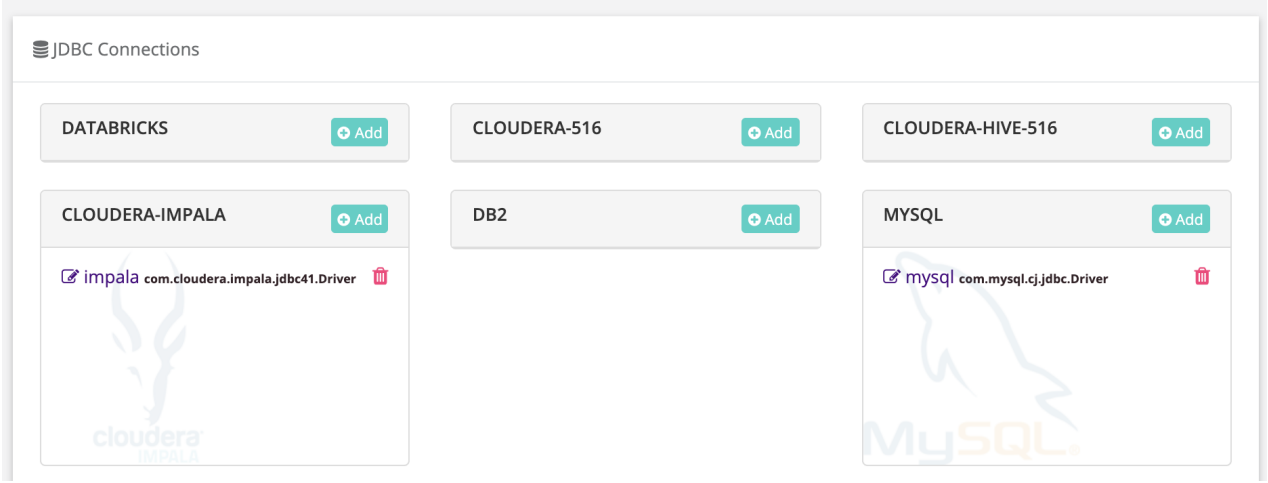
DQ Job Databricks

Lake vs Swamp

The difference between a business-critical lake and a swamp is *quality*. The accuracy and cleanliness of data is directly proportional to the quality of insights end-users will derive. Data lakes that gain broad adoption have strong governance programs. The challenge is, adding a data quality program typically takes 6-12 months but the project never really ends due to the volume, variety and velocity of incoming data. Collibra DQ uses autoML so solve this problem. DQ constantly monitors the lake with native integration and unlimited scale. Use DQ to generate the equivalent of 10K rules, while continuously adapting to the natural variance in your data. When erroneous data enters your lake DQ alerts the data steward and provide a rich visual displaying the break records and explainable AI describing the issue. DQ's approach is to learn from data and become incrementally smarter each day to ensure a statistically defensible data quality program.

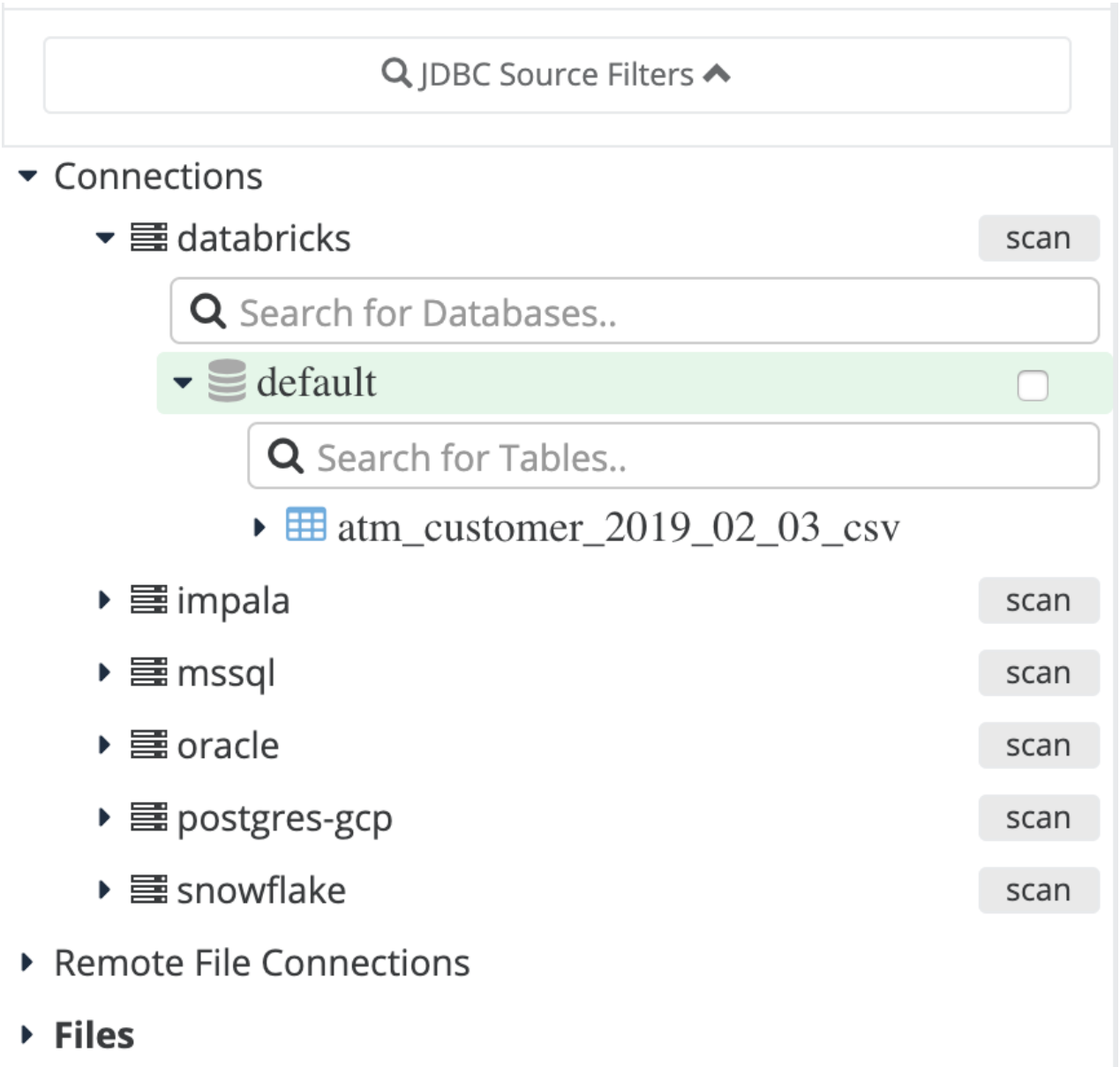
Native Integration with Delta Lake (Databricks)

Out of the box DQ comes with a connection template for Databricks. To connect, simply paste in your *username*, *password* and *connection URL*.



Explore Databricks Assets and Add DQ Checks

Quickly explore DB assets that are cataloged in Delta Lake the same way you would any database (file tree explorer). Use DQ wizard to add data quality to any Databricks table or file. Create a modern data quality program using machine learning in minutes.



9 dimensions of Data Quality

Use the wizard to apply DQ's autoML and predictive data quality features across all of your assets in Delta Lake. Click Scan button to put every table in Delta Lake under DQ management in 1 click. DQ creates a data quality program on all Delta Lake assets in a matter of hours. With traditional technologies this task used to require domain experts, rule writers and identification of critical elements.

Out of the Box DQ measures

DQ Dimension	Desc
Outliers	numeric and categorical outlier detection
Shapes	formatting and incorrect characters
Patterns	relationship probabilities
Correlations	strengths of relationships between columns
Duplicates	fuzzy and exact matching
Schema Evolution	schema drift
Rules	ability to add your own business rules
Source Matching	difference from source to target detection

DQ Job Hive

Run a data quality check on a Hive table. Use the -hive flag for a native connection via the HCat, this does not require a JDBC connection and is optimized for distributed speed and scale.

Hive Native, no JDBC Connection

Open source platforms like HDP, EMR and CDH use well known standards and because of this DQ takes advantage of things like HCat and it removes the need for JDBC

connection details as well as offers optimum data read speeds. DQ recommends and supports this with the -hive flag.

```
./owlcheck -ds hive_table -rd 2019-03-13 \
-q "select * from hive_table" -hive
```

Example output. A hoot is a valid JSON response

```
{
  "dataset": "hive_table",
  "runId": "2019-02-03",
  "score": 100,
  "behaviorScore": 0,
  "rows": 477261,
  "prettyPrint": true
}
```

Hive JDBC

1. You need to use the hive JDBC driver, commonly org.apache.hive.HiveDriver.
2. You need to locate your driver JDBC Jar with the version that came with your EMR, HDP or CDH
    a. This jar is commonly found on an edge node under /opt/hdp/libs/hive/hive-jdbc.jar etc...

```
./owlcheck -rd 2019-06-07 -ds hive_table \
-u <user> -p <pass> -q "select * from table" \
-c "jdbc:hive2://<HOST>:10000/default" \
-driver org.apache.hive.HiveDriver \
-lib /opt/owl/drivers/hive/ \
-master yarn -deploymode client
```

HDP Driver - org.apache.hive.HiveDriver

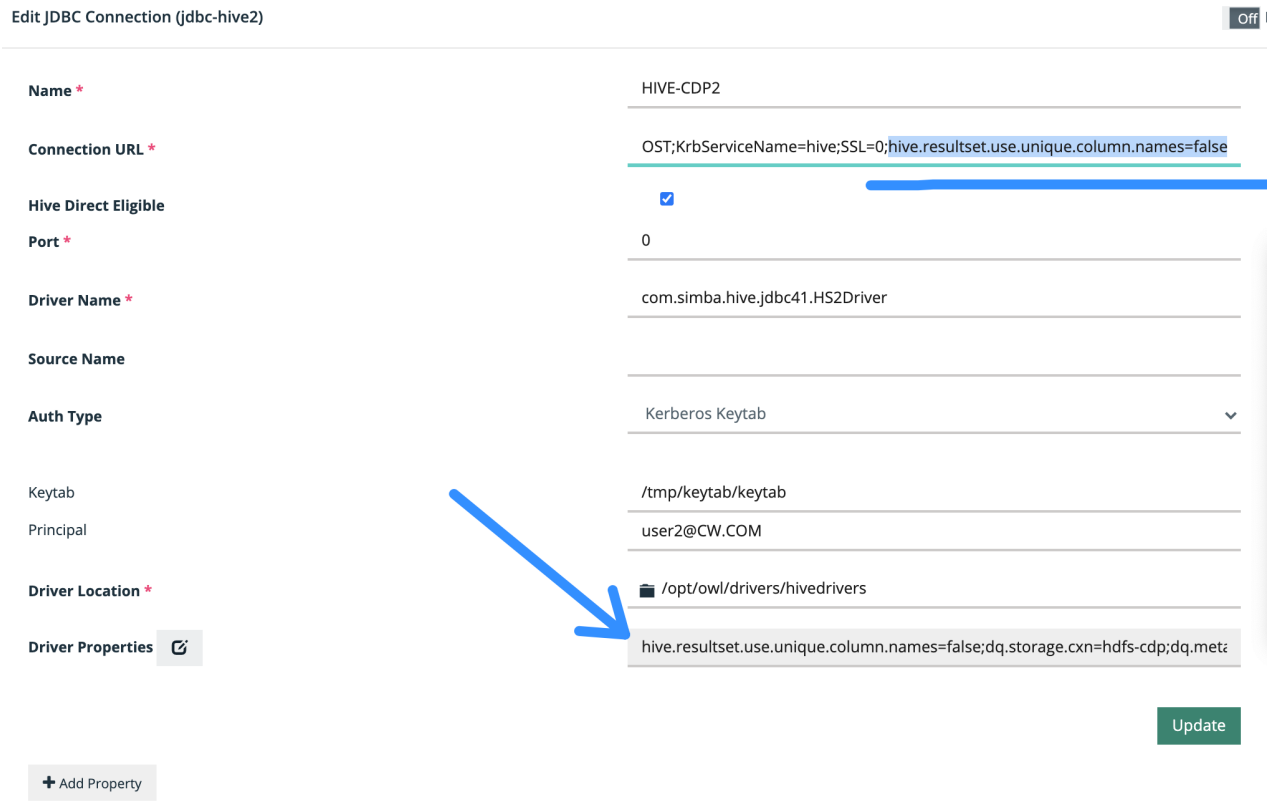
CDH Driver - com.cloudera.hive.jdbc41.Datasource

For CDH all the drivers are packaged under, HiveJDBC41_cdhversion.zip.

Troubleshooting

A common JDBC connection is hive.resultset.use.unique.column.names=false.

This can be added directly to the JDBC connection url string or to the driver properties section.



Test your hive connection via beeline to make sure it is correct before going further.

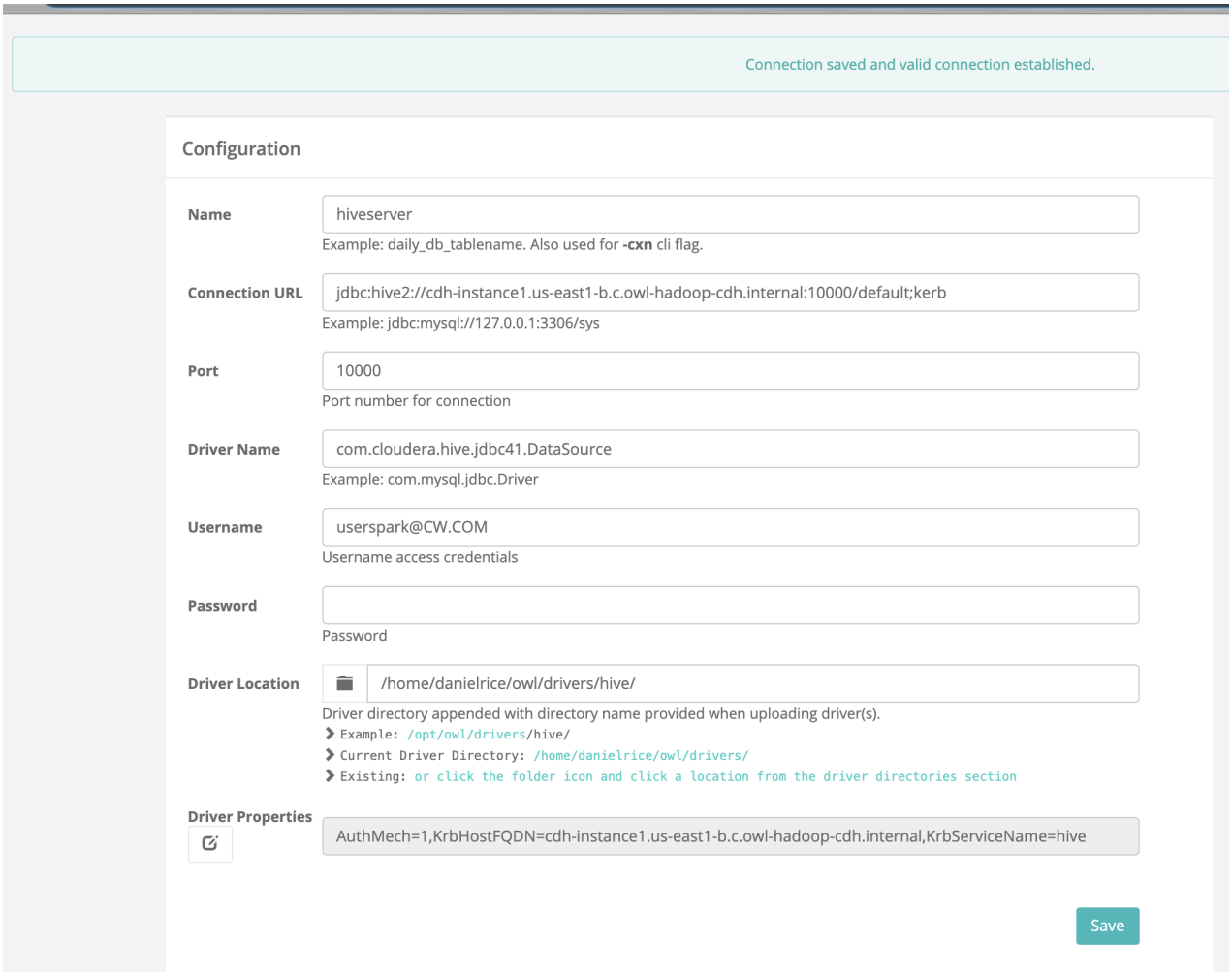
```
beeline -u 'jdbc:hive2://<HOST>:10000/default;principal=hive/cdh-instance1.us-east1-b.c.company-hadoop-cdh.internal@CW.COM;useSSL=true' -d org.apache.hive.jdbc.HiveDriver
```

Kerberos Example

```
jdbc:hive2://<HOST>:10000/default;principal=hive/cdh-instance1.us-east1-b.c.company-hadoop-cdh.internal@CW.COM;useSSL=true
```

Connecting DQ WebApp to Hive JDBC

Notice the driver properties for kerberos and principals.



In very rare cases where you can't get the jar files to connect properly one workaround is to add this to the DQ-web startup script.

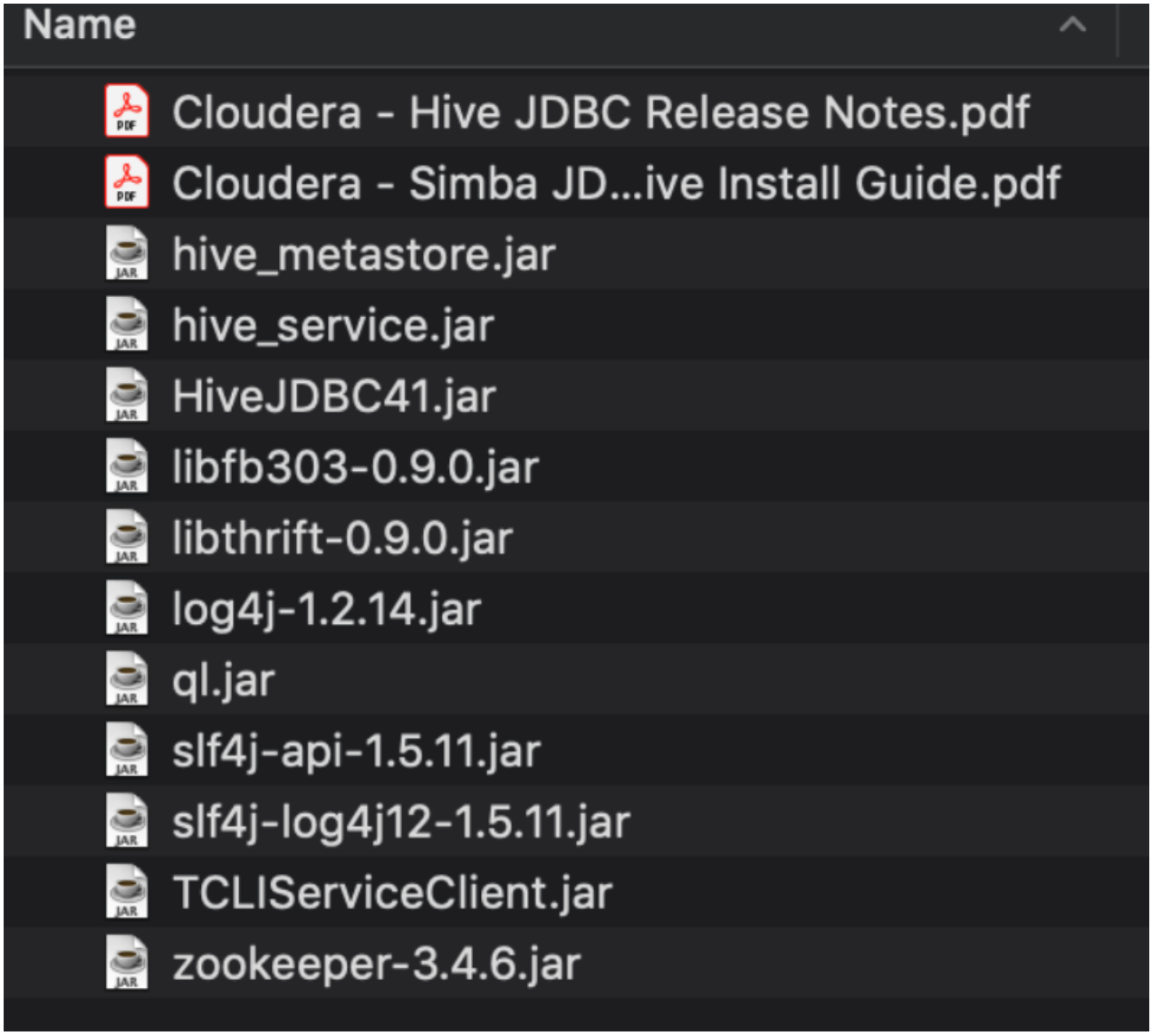
```
$JAVA_HOME/bin/java -Dload-
er.path=lib,/home/danielrice/owl/drivers/hive/ \
-DowlAppender=owlRollingFile \
-DowlLogFile=owl-web -Dlog4j.configurationFile=file://$INSTALL_
PATH/config/log4j2.xml \
$HBASE_KERBEROS -jar $owlweb $ZKHOST_KER \
--logging.level.org.springframework=INFO $TIMEOUT \
--server.session.timeout=$TIMEOUT \
--server.port=9001 > $LOG_PATH/owl-web-app.out 2>&1 & echo $!
>$INSTALL_PATH/pids/owl-web.pid
```

Class Not Found apache or client or log4j etc...

Any class not found error means that you do not have the "standalone-jar" or you do not have all the jars needed for the driver.

Hive JDBC Jars

It is common for Hive to need a lot of .jar files to complete the driver setup.



Java jar cmds

Sometimes it is helpful to look inside the jar and make sure it has all the needed files.

```
jar -tvf hive-jdbc.jar
```

DQ Job Files

For example, a large file `transaction_2021-01-01.csv` might contain the following transaction data with two transaction per day spanning all of January.

transaction_id	account_id	date	amount
1	1	2021-01-01	100
2	2	2021-01-01	120
3	1	2021-01-02	90
4	2	2021-01-02	115
...		...	...
61	1	2021-01-31	100
62	2	2021-01-31	999

and this file might be located on the directory `~/customer/transaction-2021-01-01/`.

```
~/customer
   ├── transaction-2021-01-01
   │   └── transaction_2021-01-01.csv
   ├── transaction-2021-02-01
   │   └── transaction_2021-02-01.csv
   ... # folders for 2021-03-01 to 2021-11-01 ommitted
   ├── transaction-2021-12-01
   │   └── transaction_2021-12-01.csv
```

Other folders with similar pattern may exist in your directory, such as `~/customer/transaction-2021-02-1`. Note that February data is located in a separate directory with a similar pattern for all the months of 2021. This dataset could similarly have 2 account IDs and 1 transaction per account per day (= 28 x 2 = 56 rows of data). For this example, let's assume this is the case for all the files.

To run an DQCheck on this single file containing multiple dates, you have the following choices.

DQChecks with **file**

1. Run an DQCheck on all the rows in a single file.

```
./owlcheck
    -ds DQCheck_transactions_jan21
    -rd "2021-01-01"
    -f "~/customer/transaction-2021-01-01/transaction_2021-01-
01.csv"
    -fq "select * from dataset"
    ... # other relevant options
```

Here we assume that run date (`-rd`) is "2021-01-01" because it is currently January 1, 2021. The above command would lead to an DQCheck on 62 rows of data spanning all of January 2021 from a single file located at `~/customer/transaction-2021-01-01/transaction_20210101.csv`. If you were to schedule a job to run this job monthly and next job ran on February 1st, 2021, then same DQ checks will be performed on the **same set of 62 rows** with same score as your DQCheck run from January 1, 2021. For example, the follow-up scheduled job running on February 1st, 2021 would be:

```
./owlcheck
    -ds DQCheck_transactions_jan21
    -rd "2021-02-01"
    -f "~/customer/transaction-2021-01-01/transaction_2021-01-
01.csv"
    -fq "select * from dataset"
    ... # other relevant options
```

This type of DQCheck on a single file is suitable if you are verifying a static file `~/customer/transaction-2021-01-01/transaction_20210101.csv` that does not change over time and expect the score to be the same every run. Hence, it is suggested to name the dataset that reflect this, such as `DQCheck_transaction_jan21` to reflect the idea that this dataset is checking the Data Quality of transaction table containing January 2021 data. Similar DQCheck for February 2021 data would then be a separate and independent dataset named `DQCheck_transaction_feb21` This type of DQCheck can

also be used if `~/customer/transaction-2021-01-01/transaction_20210101.csv` is changing (the rows are changing values or new rows are being added) and want to detect data quality changes. Transaction file doesn't fit with this scenario, but the idea is that the above command specifies Data Quality DQChecks on the entirety of the file. The run date is a date that you choose to assign for that DQCheck. It is conventional to have one-to-one mapping between run date and the date corresponding to the date that DQ checks are being performed. Run date does not have to match with the data underlying the file.

2. Run an DQCheck on subset of rows from a single file

The single file contains daily data for January of 2021. To run Data Quality checks on January 1st, January 2nd, ... , and January 31st, you need to run 31 DQChecks, each with subset of rows from the file. Note the `where` clause in `-fq` matching with the run date `-rd`

```
./owlcheck
    -ds DQCheck_transactions_jan21
    -rd "2021-01-01"
    -f "~/customer/transaction-2021-01-01/transaction_2021-01-
01.csv"
    -fq "select * from dataset where date = '2021-01-01'"
    ... # other relevant options

./owlcheck
    -ds DQCheck_transactions_jan21
    -rd "2021-01-02"
    -f "~/customer/transaction-2021-01-01/transaction_2021-01-
01.csv"
    -fq "select * from dataset where date = '2021-01-02'"
    ... # other relevant options

... # Owlchecks for -rd 2021-01-03 to 2021-01-30 ommitted

./owlcheck
    -ds DQCheck_transactions_jan21
    -rd "2021-01-31"
    -f "~/customer/transaction-2021-01-01/transaction_2021-01-
01.csv"
    -fq "select * from dataset where date = '2021-01-31'"
    ... # other relevant options
```

By using the same dataset name `-ds`, all 31 DQCheck will appear under one dataset `DQCheck_transaction_jan21` in the Hoot page.

A convenient way to parameterize this run date is to use `${rd}` in the query.

```
./owlcheck
    -ds DQCheck_transactions_jan21
    -rd "2021-01-01"
    -f "~/customer/transaction-2021-01-01/transaction_2021-01-
01.csv"
    -fq "select * from dataset where date = '${rd}'"
    ... # other relevant options
```

A daily scheduled job starting on January 1st, 2021 to January 31, 2021 will automatically replace the `${rd}` with "2021-01-01", "2021-01-02", ... , and "2021-01-31" for the respective run date.

3. Run an DQCheck on subset of rows from a single file with day lookback

For certain core components like **Outlier**, a set of rows corresponding to historical training data can be used to establish a baseline. For example, the row with `transaction_id` 62 has amount of 999. This looks like an outlier that we want to catch. This value of 999 seems to be an outlier because past transaction amounts for `account_id`2 are in the 100s range. We can use historical data from January 15th to January 30th and use that info to see if January 31st data contains any outliers. In this scenario, our single file `~/customer/transaction-2021-01-01/transaction_2021-01-01.csv` contains such historical data because that file contains all the data for all of January. How do we use the same file for both current data (January 31st) and historical (January 15th to January 30th) data? You do not have to split the files into two. You can simply do exactly what you would do for DQCheck on "2021-01-31" with a `-fullfile` flag. The `-fullfile` flag tells the DQCheck that "the file in `-f` contains the historical data. Construct a query and subset those rows for me".

```
./owlcheck
    -ds DQCheck_transactions_jan21
    -rd "2021-01-31"
    -f "~/customer/transaction-2021-01-01/transaction_2021-01-
01.csv"
    -fq "select * from dataset where date = '2021-01-31'"
    -fullfile
    # outlier options
    -dc "date"
    -dl
    -tbin "DAY" # look back time bin is day
    -dllb 15 # look back up to 15 days
    ... # other relevant options
```

DQChecks with *multiple files*

4. Run an DQCheck on a single file with lookback using series of file

Recall our folder structure:

```
~/customer
  ├── transaction-2021-01-01
  │   └── transaction_2021-01-01.csv
  ├── transaction-2021-02-01
  │   └── transaction_2021-02-01.csv
  ... # ommitted for space
  ├── transaction-2021-12-01
  │   └── transaction_2021-12-01.csv
```

If we want to run an DQCheck for December of 2021 and use July of 2021 to November of 2021 as our historical training dataset, how can we load *multiple files*? Just like how `-fullfile` provides a convenient way to create historical training dataset on a single file, `-fllb` (file lookback) provides a convenient way to load *series of files* with patterns while still pointing to the target file (December file) in `-f`

```
./owlcheck
    -ds DQCheck_transactions_dec21
    -rd "2021-12-01"
    -f "~/customer/transaction-2021-12-01/transaction_2021-12-
01.csv"
    -fq "select * from dataset"
    -fllb
    # outlier options
    -dc "date"
    -dl
    -tbin "MONTH"
    -dllb 5 # look back up to 5 months
    ... # other relevant options
```

One caveat to this `-fllb` method __ is that the DQCheck history must be "primed" first so that the DQ knows the file path of the past series of files. In fact, `-fllb` does not use the file path provided in `-f` and loads different files from different folders. *It relies on the DQCheck history* under the same `-ds` name. `-fllb` means lookback up to N number of past consecutive DQChecks. For each of those past DQCheck, look up the file path `-f` used in the past and follow those paths. The number N is determined by the maximum number of lookbacks from Outlier (`-dllb`) and Patterns (`-fpglb`). In the DQCheck above, because `-dllb 5` is provided along with `-fllb`, it means "Look up 5 past DQChecks and load those files as historical dataset". In summary, in order to run an DQCheck on "2021-12-01" and have that DQCheck for that date "look up" the files in `~/customer/transaction-2021-07-01/transaction_2021-07-01.csv`, `~/customer/transaction-2021-08-01/transaction_2021-08-01.csv`, `~/customer/transaction-2021-09-01/transaction_2021-09-01.csv`, `~/customer/transaction-2021-10-01/transaction_2021-10-01.csv` , and `~/customer/transaction-2021-11-01/transaction_2021-11-01.csv` , you need to have ran DQChecks for "2021-07-01", "2021-08-01", ... , and "2021-11-01" under the same dataset name. Therefore, it would be more logical, best-practice is to name the dataset `-ds DQCheck_transaction_2021` and run series of monthly owlchecks up to "2021-12-01" (but the name of the dataset is up to you)/

```
# Prime past Owlchecks so that "2021-12-01" knows the file path
of past months
./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-07-01"
    -f "~/customer/transaction-2021-07-01/transaction_2021-07-
01.csv"
    -fq "select * from dataset"

./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-08-01"
    -f "~/customer/transaction-2021-08-01/transaction_2021-08-
01.csv"
    -fq "select * from dataset"


./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-09-01"
    -f "~/customer/transaction-2021-08-01/transaction_2021-09-
01.csv"
    -fq "select * from dataset"

./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-10-01"
    -f "~/customer/transaction-2021-08-01/transaction_2021-10-
01.csv"
    -fq "select * from dataset"

./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-11-01"
    -f "~/customer/transaction-2021-11-01/transaction_2021-11-
01.csv"
    -fq "select * from dataset"

# Priming 5 past Owlchecks complete. Now run the 2021-12-01
./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-12-01"
    -f "~/customer/transaction-2021-12-01/transaction_202-11-
201.csv"
    -fq "select * from dataset"
    -fllb
    # outlier options
    -dc "date"
    -dl
```

```
    -tbin "MONTH"
    -dllb 5 # look back up to 5 months
    ... # other relevant options
```

In this scenario, since the folder paths have a pattern, we can use `-br` for priming in one command instead of writing five DQCheck commands. The flag `-br` runs DQChecks consecutively from the past and increments by monthly if `-tbin "MONTH"` (by default `-tbin DAY`so the default behavior is to increment daily). The different folder paths on each past consecutive run dates are replaced with `${rd}`. The below command is identical to the above:

```
# Prime past Owlchecks so that "2021-12-01" knows the file path
of past months
./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-12-01"
    -f "~/customer/transaction-${rd}/transaction_${rd}.csv"
    -fq "select * from dataset"
    -br 5 # run 5 runs to in the past
    -tbin "MONTH" # <-- required since we want a MONTHLY back-
run. Default "DAY"

# Priming complete. Now run the 2021-12-01
./owlcheck
    -ds DQCheck_transactions_2021
    -rd "2021-12-01"
    -f "~/customer/transaction-2021-12-01/transaction_2021-12-
01.csv"
    -fq "select * from dataset"
    -fllb
    # outlier options
    -dc "date"
    -dl
    -tbin "MONTH"
    -dllb 5 # look back up to 5 months
    ... # other relevant options
```

This pattern is designed so that a single DQCheck command can be scheduled and `${rd}` be used to replace the folder & file path. Your `~/customer` folder could contain transactions for all the years, spanning all the way back to 1992 and into the future like so:

```
~/customer

   ├── transaction-1992-01-01
   │   └── transaction_1992-01-01.csv
   ├── transaction-1992-01-01
   │   └── transaction_1992-01-01.csv
   ... # ommitted for space
   ├── transaction-2021-01-01
   │   └── transaction_2021-01-01.csv
   ├── transaction-2021-02-01
   │   └── transaction_2021-02-01.csv
   ... # ommitted for space
   ├── transaction-2021-12-01
   │   └── transaction_2021-12-01.csv
   ... # hasn't happened yet!
```

In this scenario, a monthly scheduled job would get rid of the need to "prime" the DQCheck history, since your past scheduled jobs would have already ran the past DQChecks.

DQ Job HDFS

Run data quality on a file in HDFS. Collibra DQ automatically infers the schema and create an internal training model.

```
-f "hdfs:///demo/ssn_test2.csv" \
-d "," \
-rd "2018-01-08" \
-ds "ssn_hdfs_file" \
-master yarn \
-deploymode cluster \
-numexecutors 2 \
-executormemory 2g
```

DQ Job JSON

Files

Run against a file using -json. Additionally, options are available for -flatten and -multiline. This is helpful for nested and various formats.

```
-ds json_file_example \
-f s3a://bucket_name/file.json \
-h instance.us-east4-c.c.owl-node.internal:5432/postgres \
-master spark://instance.us-east4-c.c.owl-node.internal:7077 \
-json \
-flatten \
-multiline
```

> Note Automatic flattening will infer schema and explode all structs, arrays, and map types.

Using Spark SQL

```
-ds public.json_sample \
-lib "/opt/owl/drivers/postgres/" \
-h instance.us-east4-c.c.owl-node.internal:5432/postgres \
-master spark://instance.us-east4-c.c.owl-node.internal:7077
-q "select * from public.jason"
-rd "2021-01-17"
-driver "org.postgresql.Driver"
-cxn postgres-gcp
-fq "select  \
get_json_object(col_3, '$.data._customer_name') AS  `data_cus-
tomer_name` , \
get_json_object(col_3, '$.data._active_customer') AS  `data_act-
ive_customer` , \
from dataset "
```

> Note Pass in the path to Owls' -fq parameter. This is great for mixed data types within a database. For example, if you store JSON data as a string or a blob among other data.

```
// Flatten
val colArr = new JsonReader().flattenSchema(df.schema)
colArr.foreach(x => println(x))
```

> Note This Owl utility traverses the entire schema and print the proper get JSON object spark sql strings. You can use this instead of typing each query statement into the command line -fq parameter as seen above.

Using DQ Libraries

```
import com.owl.common.options._
import com.owl.core.util.OwlUtils
import com.owl.core.activity2.JsonReader

val connProps = Map (
  "driver"   -> "org.postgresql.Driver",
  "user"     -> "user",
  "password" -> "password",
  "url"      -> "jdbc:postgresql://10.173.0.14:5432/postgres",
  "dbtable"  -> "public.data"
)

// Spark
var rdd = spark.read.format("jdbc").options
(connProps).load.select($"col_name").map(x=>x.toString()).rdd
var df = spark.read.json(rdd)

// Flatten
val colArr = new JsonReader().flattenSchema(df.schema)
val flatJson = df.select(colArr: _*)
flatJson.cache.count

// Opts
val dataset = "json_example"
val runId = s"2021-01-14"
val opt = new OwlOptions()
opt.dataset = dataset
opt.runId = runId
opt.datasetSafeOff = true

// Owlcheck
OwlUtils.resetDataSource("instance.us-east4-c.c.owl-node.in-
ternal","5432/postgres","user","pass", spark)
val owl = OwlUtils.OwlContext(flatJson, opt)
owl.register(opt)
owl.owlCheck
```

> Note
> JsonReader()
> This uses DQ's JsonReader to do the heavy lifting.

Known Limitations

- Array and nested array datatypes in JSON files are not supported.

DQ Job MySql

Video Tutorial (MySQL)

Add automatic data quality to any database in 60 seconds. This example shows a single table being selected for DQ, however Collibra DQ also provides the ability to scan all schemas and tables at once.

JDBC Connect from Kirk Haslbeck on Vimeo.

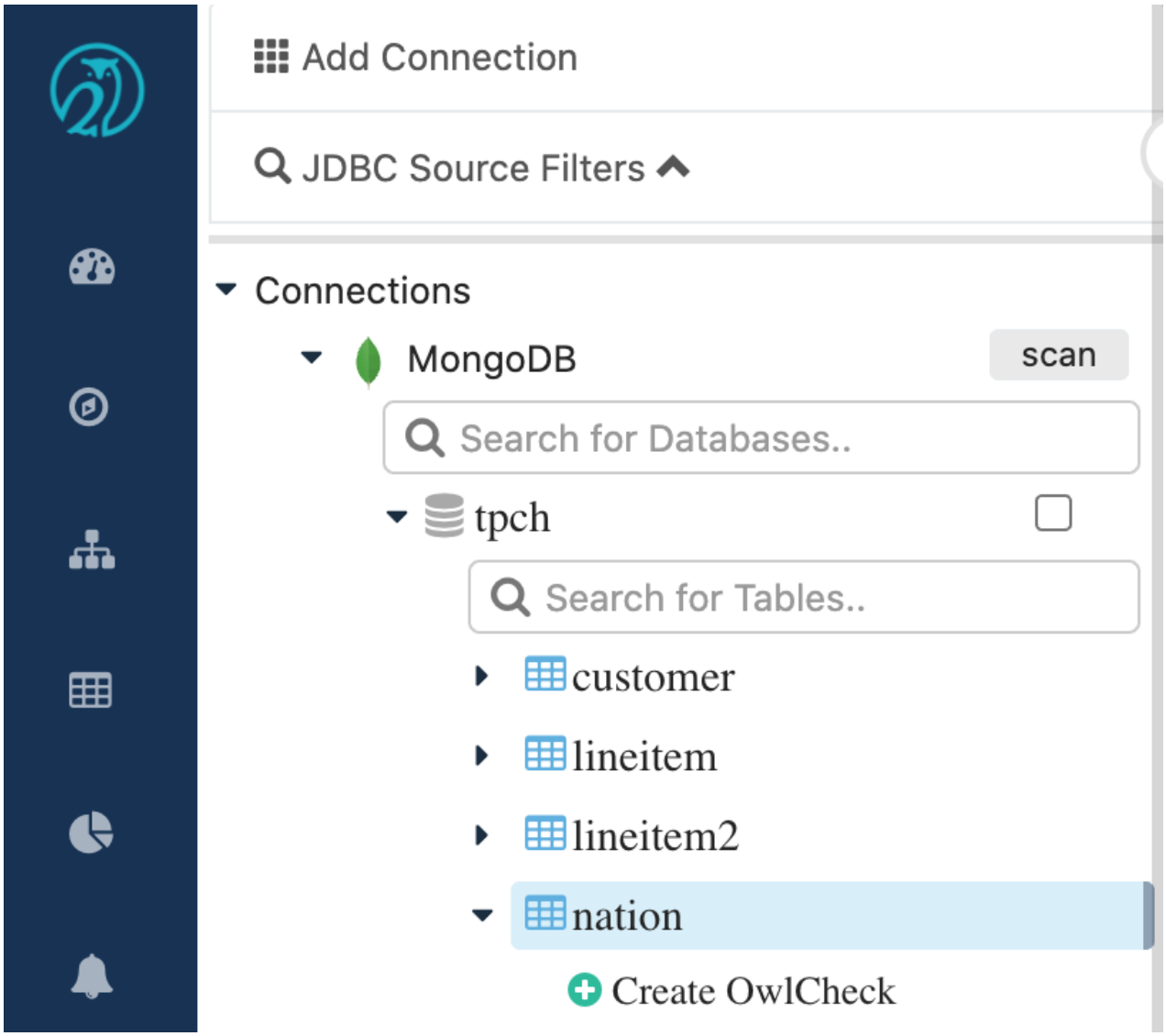
Connect to any database using JDBC. Mysql example below.

```
-q "select * from lake.stock_eod where date = '2017-01-20' " \
-u username -p password \
-c "jdbc:mysql://owldatalake.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com:3306" \
-rd "2017-01-20" \
-dc "date" \
-ds "stocks" \
-driver com.mysql.jdbc.Driver \
-lib "/home/ec2-user/owl/drivers/mysql/"
```

DQ Job MongoDB

Browse MongoDB like any other relational database

Using Collibra DQ's file tree explorer browse Mongo "collections" like "tables". Then use the wizard to create standard DQ scans.



CMD Line

Copy paste-able cmdline example for simple spark submit job.

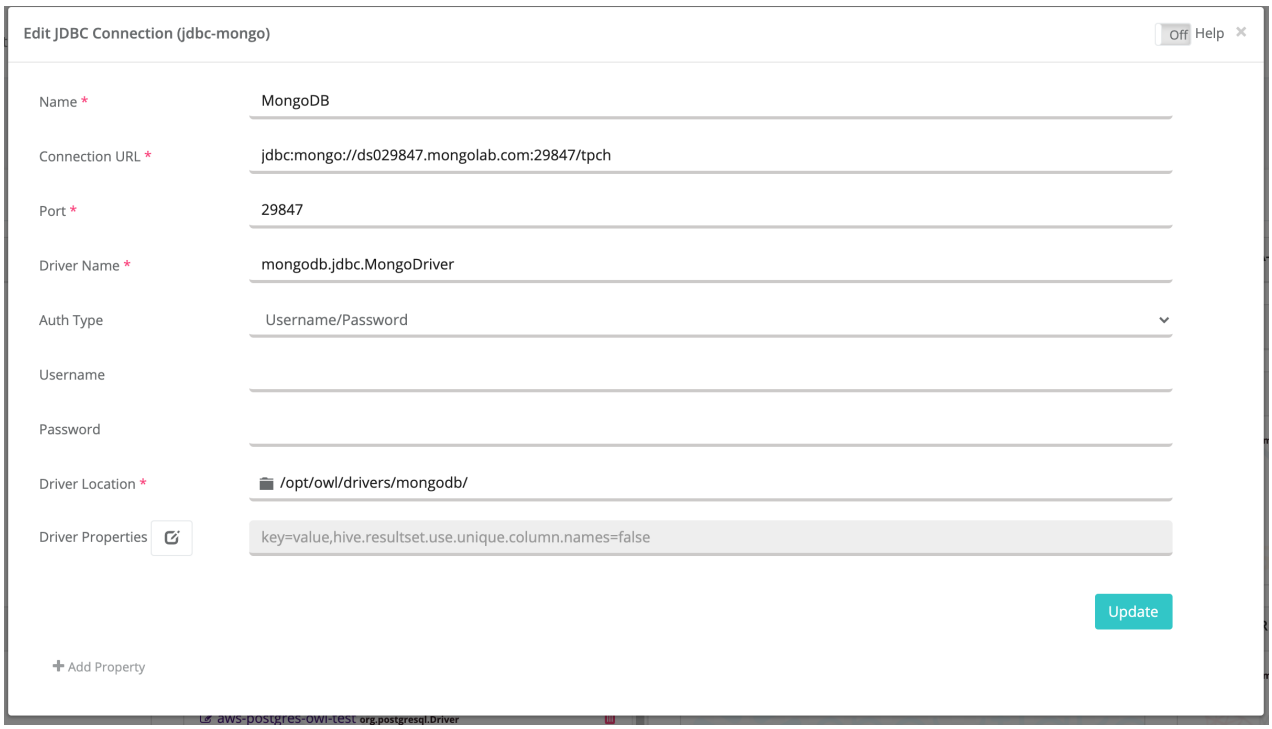
```
-lib "/opt/owl/drivers/mongodb/"
-h localhost:5432/postgres
-master local[*]
-ds tpch.lineitem_7
-br 10 -deploymode client
-q "select * from tpch.lineitem where l_shipdate between '${rd}
00:00:00.000+0000'
and '${rdEnd} 00:00:00.000+0000' "
-bhlb 10 -rd "1998-12-01"
-driver "mongodb.jdbc.MongoDriver"
-loglevel INFO -cxn MongoDB -rdEnd "1998-12-02"
```

Drivers and Config

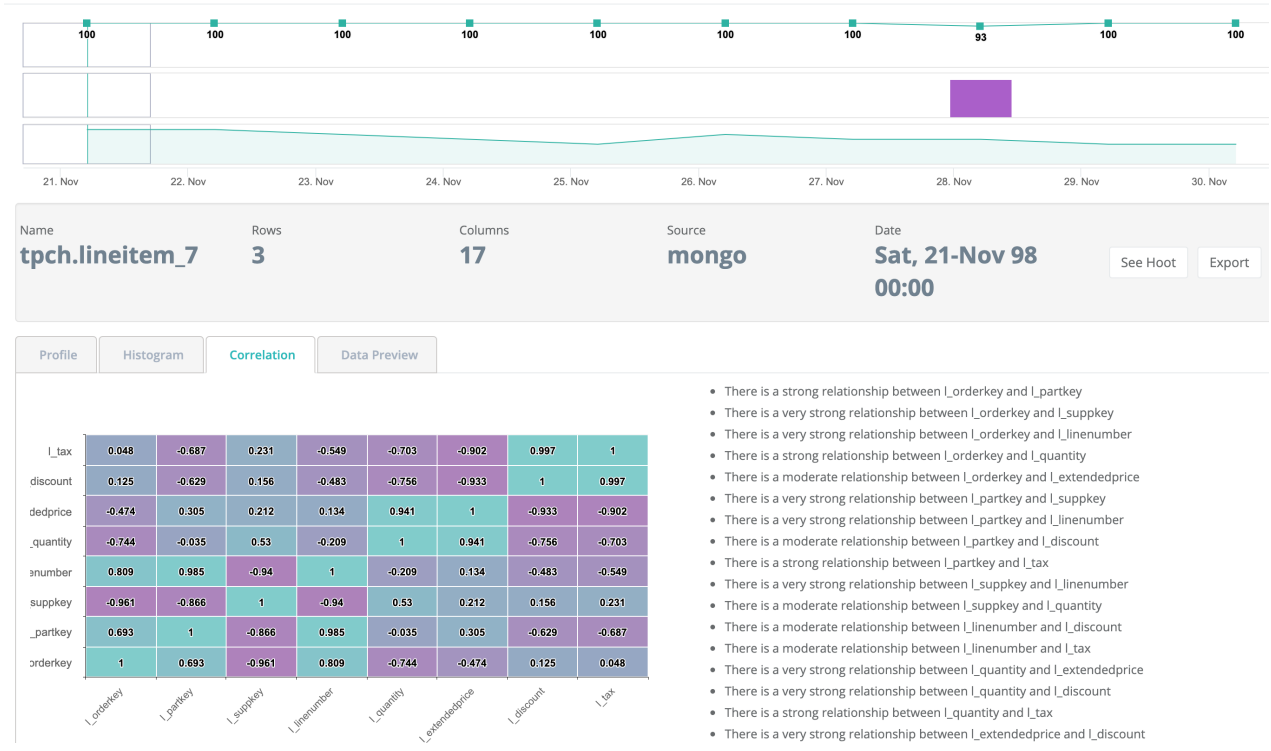
In order to make this possible Collibra DQ requires two drivers, MongoDB driver and UnityJDBC Driver. Out of the box DQ comes preconfigured with these drivers. You simply open the MongoDB connection template and paste in your JDBC URL.

```
driverClass: mongodb.jdbc.MongoDriver

path: /opt/owl/drivers/mongodb/
    +-- mongoJdbc.jar
    +-- unityJDBC.jar
```

Simply paste in JDBC Info



Discover Correlations, Relationships, DQ issues and Much More...



The following table presents the various SQL statements related to table-level actions and the corresponding MongoDB statements.https://docs.mongodb.com/manual/reference/sql-comparison/

SQL Schema Statements	MongoDB Schema Statements

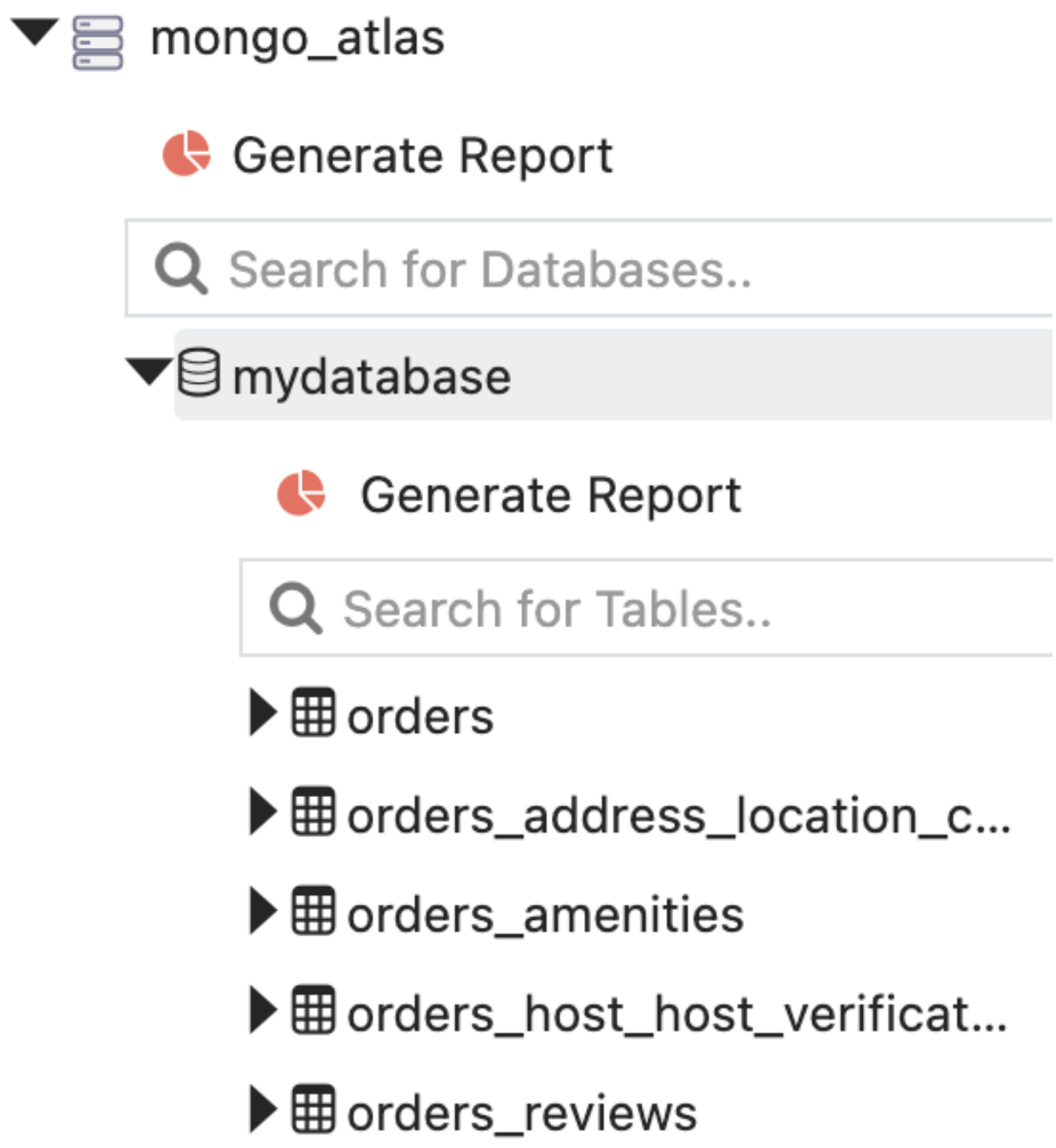
`CREATE TABLE people ( id MEDIUMINT NOT NULL AUTO_INCREMENT, user_id Varchar(30), age Number, status char(1), PRIMARY KEY (id) )`	Implicitly created on first `insertOne()` or `insertMany()` operation. The primary key `_id` is automatically added if `_id` field is not specified.
	`db.people.insertOne( { user_id: "abc123", age: 55, status: "A" } )`
	However, you can also explicitly create a collection:
	`db.createCollection("people")`

`ALTER TABLE people ADD join_date DATETIME`	Collections do not describe or enforce the structure of its documents; i.e. there is no structural alteration at the collection level. However, at the document level, `updateMany()` operations can add fields to existing documents using the `$set` operator. `db.people.updateMany( { }, { $set: { join_date: new Date() } } )`

Limiting Collections in the JDBC URL

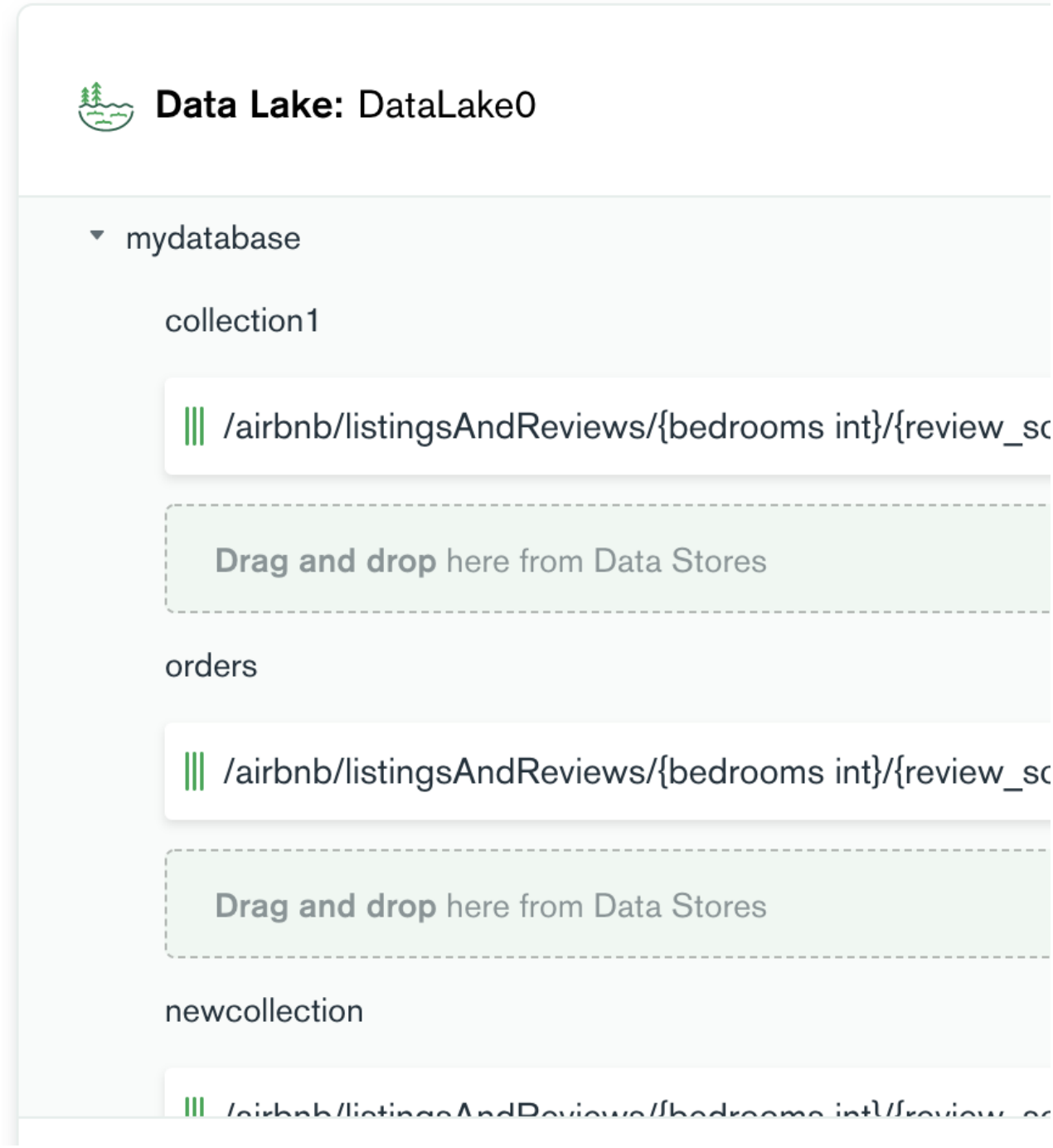
```
jdbc:mongodb://<dbuser>:<password>@datalake0-dza-
1q.a.query.-
mon-
god-
b.net/<mydata-
base>?ssl=true&authSource=admin&rebuildschema=true&tables=orders
```

There are three collections in this mongodb atlas lake. By adding &tables=orders in the URL params you can see only order collections show up in the explorer.



Three Collections in MongoDB Atlas

The total number of collections in mongodb atlas lake.



DQ Job S3

S3 permissions need to be setup appropriately.

> Note S3 connections should be defined using the root bucket. Nested S3 connections are not supported.

Example Minimum Permissions

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketMultipartUploads",
                "s3:ListBucket",
                "s3:ListMultipartUploadParts",
                "s3:GetObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:athena:*:<AWSAccountID>:workgroup/prima-
ry",
                "arn:aws:s3:::<S3 bucket name>/*",
                "arn:aws:s3:::<S3 bucket name>",
                "arn:aws:glue:*:<AWSAccountID>:catalog",
                "arn:aws:glue:*:<AWSAccountID>:database/<databas-
e name>",
                "arn:aws:glue:*:<AWSAccountID>:table/<database
name>/*"
            ]
        }
    ]
}
```

(Needs appropriate driver) http://central.maven.org/maven2/org/apache/hadoop/hadoop-aws/ Hadoop AWS Driver hadoop-aws-2.7.3.2.6.5.0-292.jar

```
-f "s3a://s3-location/testfile.csv" \
-d "," \
-rd "2018-01-08" \
-ds "salary_data_s3" \
-deploymode client \
-lib /home/ec2-user/owl/drivers/aws/
```

Databricks Utils Or Spark Conf

```
val AccessKey = "xxx"
val SecretKey = "xxxyyyzzz"
//val EncodedSecretKey = SecretKey.replace("/", "%2F")
val AwsBucketName = "s3-location"
val MountName = "kirk"

dbutils.fs.unmount(s"/mnt/$MountName")

dbutils.fs.mount(s"s3a://${AccessKey}:${SecretKey}@${AwsBuck-
etName}", s"/mnt/$MountName")
//display(dbutils.fs.ls(s"/mnt/$MountName"))

//sse-s3 example
dbutils.fs.mount(s"s3a://$AccessKey:$SecretKey@$AwsBucketName",
s"/mnt/$MountName", "sse-s3")
```

Databricks Notebooks using S3 buckets

```
val AccessKey = "ABCDED"
val SecretKey = "aaasdfwerwerasdfB"
val EncodedSecretKey = SecretKey.replace("/", "%2F")
val AwsBucketName = "s3-location"
val MountName = "abc"

// bug if you don't unmount first
dbutils.fs.unmount(s"/mnt/$MountName")

// mount the s3 bucket
dbutils.fs.mount
(s"s3a://${AccessKey}:${EncodedSecretKey}@${AwsBucketName}",
s"/mnt/$MountName")
display(dbutils.fs.ls(s"/mnt/$MountName"))

// read the dataframe
val df = spark.read.text(s"/mnt/$MountName/atm_customer/atm_cus-
tomer_2019_01_28.csv")
```

DQ Job Snowflake

Example CMD Line

```
-h <IP_ADDRESS>:5432/postgres \
-drivermemory 4g \
-master spark://<SPARK_MASTER>:7077 \
-ds PUBLIC.TRANSLATION \
-deploymode client \
-q "select * from PUBLIC.TRANSLATION" \
-rd "2021-07-24" \
-driver "net.snowflake.client.jdbc.SnowflakeDriver" \
-cxn snowflake
```

Example JDBC Connection URL

jdbc:snowflake://<IP_ADDRESS>.snowflakecomputing.com?db=DEMODB&warehouse=COMPUTE_WH&schema=PUBLIC

Drive Name

net.snowflake.client.jdbc.SnowflakeDriver

Advanced

A Collibra DQ Check is a bash script that is essentially the launch point for any DQ Job to scan a data set. A data set can be a flat file, such as textfile, json file, parquet file, etc, or a table from any number of databases, such as Oracle, Postgres, Mysql, Greenplum, DB2, SQLServer, Teradata, etc.

Example Run a data quality check on any file by setting the file path.

```
./owlcheck -ds stock_trades -rd 2019-02-23 -f /path/to/file.csv
-d ,
```

Example output below. A hoot is a valid JSON response

```
{
  "dataset": "stock_trades",
  "runId": "2019-02-03",
  "score": 100,
  "behaviorScore": 0,
  "rows": 477261,
  "passFail": 1,
  "peak": 1,
  "dayOfWeek": "Sun",
  "avgRows": 0,
  "cols": 5,
  "activeRules": 0,
  "activeAlerts": 0,
  "runTime": "00:00:23",
  "dqItems": {},
  "datashapes": [],
  "validateSrc": [],
  "alerts": [],
  "prettyPrint": true
}
```

Monthly Data

Sometimes you may want to run monthly profiles with aggregated data. In this case, the scheduling tool can supply the $ as a variable such as $runDate and the end date as $endDate. 1 line examples for bash or shell below.

```
echo "Hello World Owl"

runDate=$(date +"%Y-%m-%d")
endDate=$(date -d "$runDate +1 month" +%Y-%m-%d)

echo $runDate
echo $endDate

./owlcheck \
-q "select * from table where date >= '$runDate' and date <
'$endDate' " \
-ds example \
-rd $runDate \
-tbin MONTH
```

Monthly BackRun (Using Collibra Data Quality's built-in Monthly)

Collibra Data Quality has 2 convenient features here:

1. The use of built-in $ and $ removes the need for any shell scripting.
2. Using -br, DQ will replay 20 months of data using this template automatically.

```
./owlcheck \
-q "select * from table where date >= '${rd}' and date <
'${rdEnd}' " \
-ds example
-rd 2019-01-01
-rdEnd 2019-02-01
-tbin MONTH
-br 20
```

Daily Data

One of the most common examples is data loading or running once a day. A job control framework can pass in this value or you can pull it from shell.

```
echo "Hello World Owl"

runDate=$(date +"%Y-%m-%d")
echo $runDate

./owlcheck \
-q "select * from table where date = '$runDate' " \
-ds example \
-rd $runDate \
-tbin DAY
```

Daily Data (Using Collibra Data Quality's built-in Daily)

```
./owlcheck \
-q "select * from table where date = '${rd}' " \
-ds example \
-rd 2019-03-14
```

Daily Data with Timestamp instead of Date

```
./owlcheck \
-q "select * from table where TS >= '${rd} 00:00:00' and TS <=
'${rd} 23:59:59' " \
-ds example \
-rd 2019-03-14
```

OR Timestamp using $

```
./owlcheck \
-q "select * from table where TS >= '${rd} 00:00:00' and TS <
'${rdEnd} 00:00:00' " \
-ds example \
-rd 2019-03-14 \
-rdEnd 2019-03-15 \
-tbin DAY
```

Hourly Data

```
./owlcheck \
-q "select * from table where TS >= '${rd}' and TS < '${rdEnd}'
" \
-ds example \
-rd    "2019-03-14 09:00:00" \
-rdEnd "2019-03-14 10:00:00" \
-tbin HOUR
```

DQ Check Template with Service Hook

The best practice is to make a generic job that would be repeatable for every DQ Check. Below is an example that first hits Collibra Data Quality using a REST call and then runs the response.

```
curl -X GET "http://$host/v2/-
getowlchecktemplate?dataset=lake.loan_customer" \
-H "accept: application/json"
```

The above REST call returns the below DQ Check. It is left up to the Job Control to replace the $ with the date from the Job Control system. You can use Collibra DQ's built-in scheduler to save these steps.

```
./owlcheck \
-lib "/home/danielrice/owl/drivers/mysql/" \
-cxn mysql \
-q "select * from lake.loan_customer where load_dt = '${rd}' " \
-key post_cd_num -ds lake.loan_customer \
-rd ${rd} \
-dc load_dt -dl -dlkey usr_name,post_cd_num -dllb 5 \
-tbin DAY -by DAY -dupe -dupeinc ip_address_home,usr_name -
dupecutoff 85 \
-fpgon -fpgkey usr_name,post_cd_num -fpgdc load_dt -fpglb 5 -
fpgtbin DAY \
-loglevel INFO \
-h $host:5432/owltrunk \
-owluser {user}
```

REST API End Point

The easiest option is to use the **runtemplate** end point API call to make requests to from cmdLine or JobControl System. This endpoint gets the DQ Check saved in Collibra instead of the client needing to know the DQ Check details.

POST https://$host**/v2/runtemplate?dataset=lake.spotify**

RunTemplate

Parameters

Path

dataset	string	name of dataset. -ds OR opt.dataset

rd	string	yyyy-MM-dd format can add time or timezone. if note passed in it will use the current day
rdEnd	string	yyyy-MM-dd format can add time or timezone. if not passed it will not be used
Responses		

- **200**

```
{
  "msg": "Success, Owl Check is Running as process 13996",
  "pid": "13996",
  "runid": "2017-01-01",
  "starttime": "Thu Oct 17 13:27:01 EDT 2019",
  "cmd": "cmd": "-ds lake.spotify -rd 2019-10-17   -q \"select *
from lake.spotify\" -cxn mysql -lib /opt/owl/drivers/mysql/ -
drivermemory 2G -histoff   -owluser {user}",
  "dataset": "lake.spotify"
}
```

Curl example for the above Rest Call

```
TOKEN=$(curl -s -X POST http://$host/auth/signin -H "Content-
Type:application/json" -d "{\"username\":\"$username\", \"pass-
word\":\"$password\"}" | jq -r '.token')

curl -i -H 'Accept: application/json' \
  -H "Authorization: Bearer ${TOKEN}" \
  http://$host/v2/runtemplate?dataset=lake.spotify
```

Bash Script

A generic and repeatable DQCheck script for job schedulers, that hooks into Collibra to get the template.

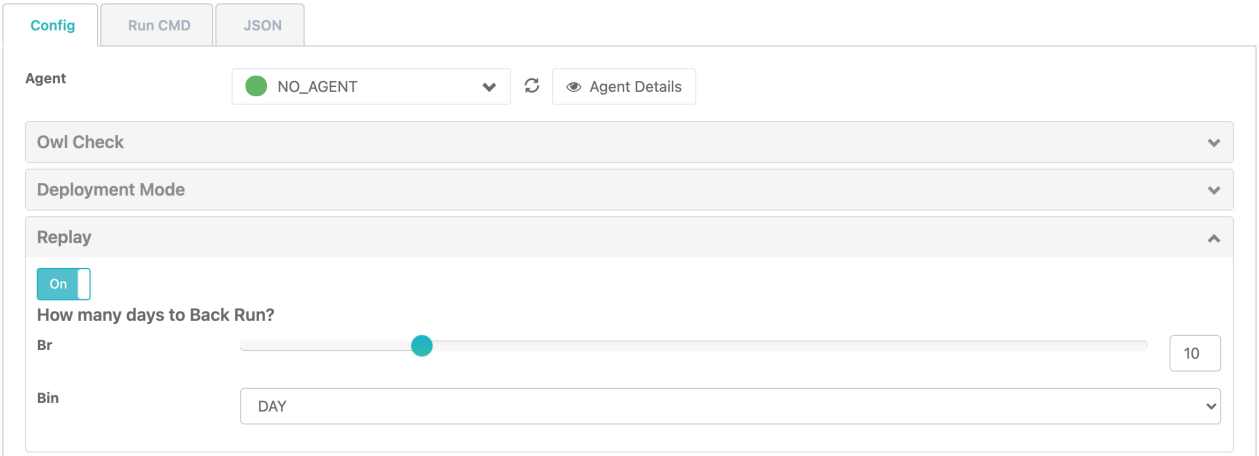
```
#1 authenticate
curl -sb -X POST -d username={user} -d password={password}
http://$OWL_HOST/login -c cookies.txt

#2 get template
owlcheck_args=$(curl -b cookies.txt -H "accept: applic-
ation/json" -X GET http://$OWL_HOST/v2/-
getowlcheckcmdlinebydataset\?dataset=insurance | sed 's/.*\[\
(.*\)\]/\1/' | sed -e "s/^\"//" -e "s/\"$//"  | sed 's/\\\"\
(.*\)\\\"/\x27\1\x27/')

#3 replace ${rd} with job_run_date
job_run_date="2019-03-14 10:00:00"
owlcheck_args=${owlcheck_args//'${rd}'/$job_run_date}

#4 run owlcheck
eval owlcheck $owlcheck_args
```

For more Information on Collibra Data Quality's Scheduler, visit the DQ Job Cron page.

DQ Job Back Run

How to Replay a Data Test

Many times you will want to see how a dataset plays out over time. This could be five days or five months. Using this slider the tool will automatically create training sets and profiles as well as run any rules or outliers you've put in place.



Quickly Replay 30 days of data, -br 30

Add -br to any DQCheck and replay in time order. Jan 1st, Jan 2nd, Jan 3rd...To do this we need to use the ${rd} variable that DQ provides as a run_date replacement for job control and templates. Also note that if you run from the cmdline you need to escape "$"s. So use \${rd}. If you are running from a Notebook or Java or Scala or the Rest API you do not need to escape the ${rd} variable.

```
./owlcheck \
-ds OWLDB2.NYSE_STOCKS3 -rd "2018-01-14" \
-lib "/opt/owl/drivers/db2/" \
-cxn db2 \
-q "select * from OWLDB2.NYSE_STOCKS where TRADE_DATE =
'\${rd}'" \
-br 4
```

Replay 4 Months of data, -br 4 -tbin MONTH

In situations where your data rolls up into Months, you may want to re-run several months of data but not a day at a time. In this case we will use -br with -tbin.

```
./owlcheck \
-ds OWLDB2.NYSE_STOCKS3 \
-rd "2018-01-01" \
-q "select * from OWLDB2.NYSE_STOCKS where TRADE_DATE =
'\${rd}'" \
-br 4 \
-tbin MONTH \
-lib "/opt/owl/drivers/db2/" \
-cxn db2
```

Monthly using a range for the entire Month

```
./owlcheck \
-ds OWLDB2.NYSE_STOCKS3 \
-rd "2018-01-01" \
-rdEnd "2018-02-01" \
-q "select * from OWLDB2.NYSE_STOCKS where TRADE_DATE >= '${rd}'
and TRADE_DATE < '${rdEnd}'" \
-br 4 \
-tbin MONTH
-lib "/opt/owl/drivers/db2/" \
-cxn db2
```

DQ Job Cron

Template for Job Control

Cron / Autosys / Control M / Oozie

It is common for organization to need to run jobs on a schedule. Below are a few shell tricks to get a date from bash and use an DQCheck with template variables.

Kinit and get run_date from shell or job control variable, pass it into Collibra DQ using $run_date.

```
%sh
run_date=$(date +%Y-%m-%d)
run_date=$(date '+%Y-%m-%d %H:%M')
echo $run_date

#kinit
echo "password" | kinit  userabc@CW.COM

~/owl/bin/owlcheck -q "select * from lake.stock_eod where date =
'$run_date' " \
-u user -p pass \
-c "jdbc:mysql://owldatalake.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com:3306" \
-rd "$run_date" \
-dc "date" \
-dl \
-dllb 7 \
-dlminhist 2 \
-tbin DAY \
-dlkey sym,exch \
-ds "lake.stock_nasdaq" \
-driver "com.mysql.jdbc.Driver" \
-lib "/home/ec2-user/owl/drivers/mysql/" \
-master yarn -deploymode client -numexecutors 1 -executormemory
1g \
-loglevel DEBUG
```

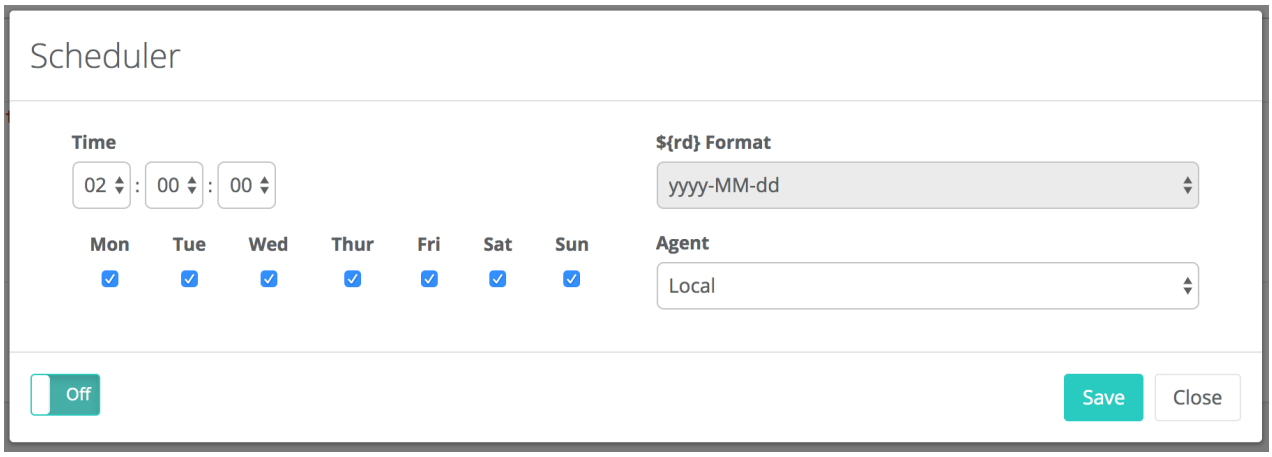
Template

You can also use -template to use DQ as a service hook and remove the need to pass in almost anything. In this case, DQ looks up the template automatically from either a previous run or if you've saved a template, and use these variables. Any variable at the cmdline will override and win/replace. This is a great way to remove connection and other information from being hard coded into the job control framework and allows edit ability from DQ Webapp.

```
%sh
~/owl/bin/owlcheck -usetemplate -ds lake.stock_nasdaq -rd $run_
date
```

Owl Scheduler - Built In

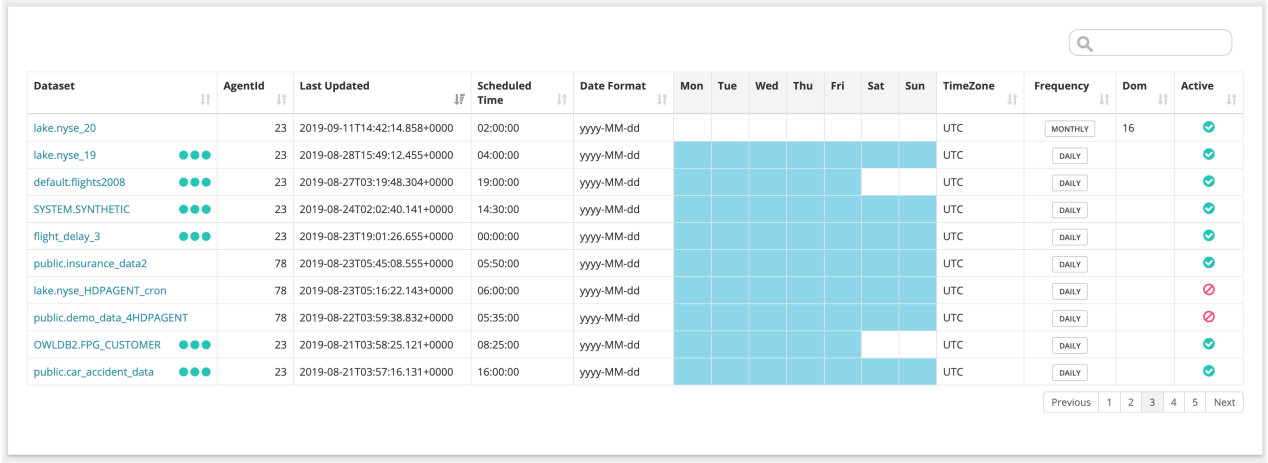
A quick option is to use DQ's built in scheduler. DQ automatically substitutes the runtime variables like $ into the job. This also gives you control to edit the DQCheck.



The schedule is based on the DQCheck Template. This way the runtime variables are replaced in each run. Notice the $ below.

All Scheduled Jobs in One Place

Under the jobs dashboard you can see an overview schedule with all running jobs and their status.



Dataset	AgentId	Last Updated	Scheduled Time	Date Format	Mon	Tue	Wed	Thu	Fri	Sat	Sun	TimeZone	Frequency	Dom	Active
lake.nyse_20	23	2019-09-11T14:42:14.858+0000	02:00:00	yyyy-MM-dd								UTC	MONTHLY	16	
lake.nyse_19	23	2019-08-28T15:49:12.455+0000	04:00:00	yyyy-MM-dd								UTC	DAILY		
default.flights2008	23	2019-08-27T03:19:48.304+0000	19:00:00	yyyy-MM-dd								UTC	DAILY		
SYSTEM.SYNTHETIC	23	2019-08-24T02:02:40.141+0000	14:30:00	yyyy-MM-dd								UTC	DAILY		
flight_delay_3	23	2019-08-23T19:01:26.655+0000	00:00:00	yyyy-MM-dd								UTC	DAILY		
public.insurance_data2	78	2019-08-23T05:45:08.555+0000	05:50:00	yyyy-MM-dd								UTC	DAILY		
lake.nyse_HDPAGENT_cron	78	2019-08-23T05:16:22.143+0000	06:00:00	yyyy-MM-dd								UTC	DAILY		
public.demo_data_4HDPAGENT	78	2019-08-22T03:59:38.832+0000	05:35:00	yyyy-MM-dd								UTC	DAILY		
OWLDB2.FPG_CUSTOMER	23	2019-08-21T03:58:25.121+0000	08:25:00	yyyy-MM-dd								UTC	DAILY		
public.car_accident_data	23	2019-08-21T03:57:16.131+0000	16:00:00	yyyy-MM-dd								UTC	DAILY		

Previous 1 2 3 4 5 Next

DQ Job Kafka

Kafka Requires Zookeeper

Apache Kafka typically requires zookeeper. This file and cmd can be run from inside /kafka/bin.

```
# Start the ZooKeeper service
# Note: Soon, ZooKeeper will no longer be required by Apache
Kafka.
$ bin/zookeeper-server-start.sh config/zookeeper.properties
```

Start a Kafka Server

Precursor step to Collibra DQ (you likely already have this step completed if you use Kafka).

```
bin/kafka-server-start.sh config/server.properties
```

Start a Kafka Topic

Precursor step to DQ (you likely already have this step completed if you use Kafka).

```
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 -
-replication-factor 1 --partitions 1 --topic test

# prefered cmd is below
bin/kafka-topics.sh --create --zookeeper localhost:2181 --rep-
lication-factor 1 --partitions 1 --topic test
```

Put a msg on "test" Topic

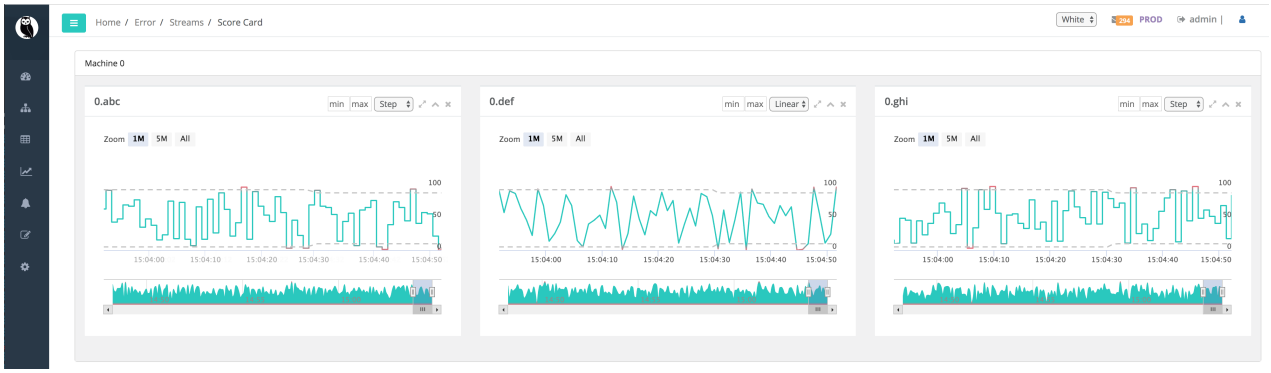
```
bin/kafka-console-producer.sh --broker-list localhost:9092 --
topic test
```

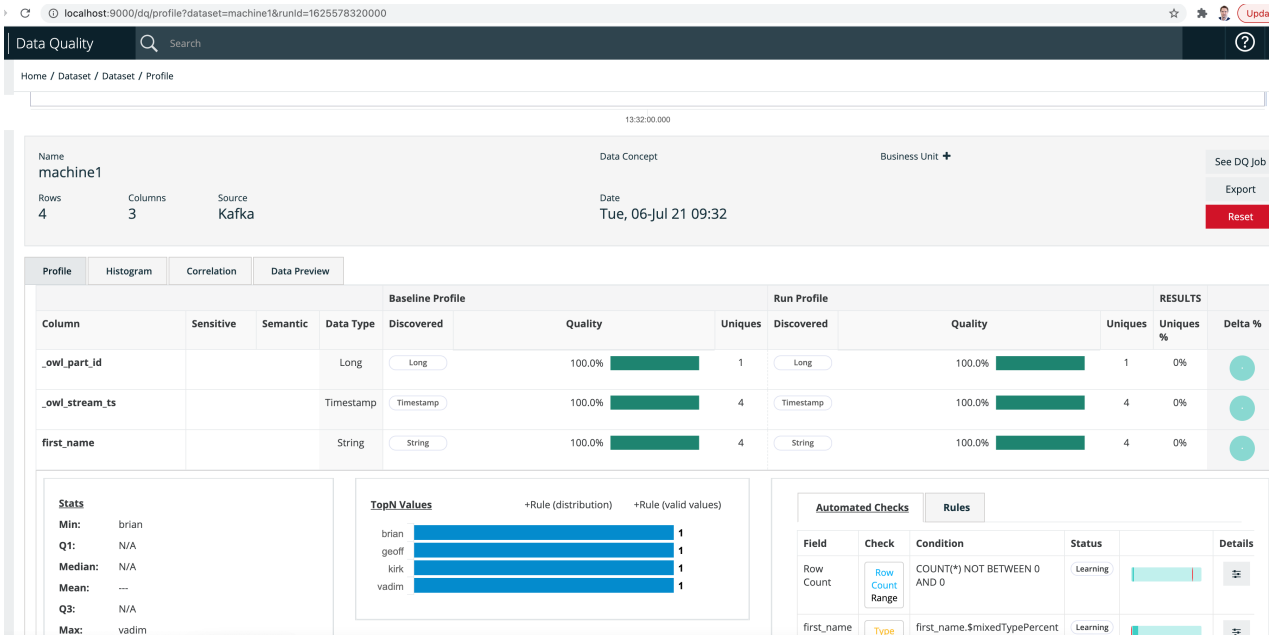
Kafka Consumer or DQ Consumer

Kafka works as a topic so you can have many consumers. Here is a basic cmdline consumer but we can add DQ as a second consumer.

```
bin/kafka-console-consumer.sh --bootstrap-server localhost:9092
--topic test --from-beginning
```

```
/opt/owl/bin/owlcheck.sh
      -kafkatopic test
      -ds machine1
      -streamformat csv
      -kafkaport  9092
      -kafkabroker localhost
      -streaminterval 60
      -stream -kafka
      -header  first_name
      -master local
```





Streams vs Sensors

Technically speaking anything moving in real-time is a stream of data but DQ classifies streams and IoT sensors as slightly different for the following reasons:

Sensors

Sensors are commonly a standard time-series. Signal, Time, Value.

Signal	Time	Value
device1-CPU	2019-02-11 13:40:55	4

Signal	Time	Value
device1-CPU	2019-02-11 14:33:20	2

Streams

Streams commonly look like messages, jsons, avro or batch data but constantly flowing. Another way to think of it is a multiple time-series.

```
[
    trade: {
        price: 23.75,
        qty: 20,
        symbol: HDP
    },
        trade: {
    }
]
```

fname	age	networth	email
Joe	45	$130,000	joe@yahoo.com
Mark	33	$125,000	mark@yahoo.com

The difference between a Sensor and a Stream in the above example is that in the case of the sensor the user is primarily concerned with the actual value of the "Value". Meaning a spike in temperature or a drop in CPUs. But in a stream of customer data there isn't a time "X" and value "Y" there are many values "Y" and you a user is interested in the overall quality of both the entire stream and the individual values. Relationship analysis and other correlative functions apply here. If you were to chart a "stream" what would you chart? The row count volume or just one of the columns or the count of something? But if you were to chart a sensor you know exactly what you would chart... the "Value" over "Time".

Fortunately DQ has already thought and worked through the many nuances required to understand, monitor and predict accurately for all of these use-case. All that is required is to subscribe the stream.

DQ Job LinkID

Link ID is an out-of-the-box feature that lets you link the findings of a DQ Job back to the source record, or key, for remediation outside the application. The link ID should be unique and is most commonly the primary key. Composite primary key is also supported.

Collibra Data Quality supports one or many primary key columns in your data sets for record linkage to your original table, file, or data frame. If your primary key column contains many columns, use a comma to delineate.

Providing the link ID

There are two ways to provide the link ID:

1. From the command line using `-linkid`.
2. In a notebook via opt.linkId.

Combining link ID and Run Discovery

To combine the features of link ID and Run Discovery, first enable link ID and then use Run Discovery. This lets you apply sensitivity labels to data classes and trigger breaks for all the records that do not match your link ID.

Link ID and the DQ Metastore

No personal data is stored in the Metastore when using link ID. The Metastore only stores:

- The rule that is applied to your DQ Job.
- The data set used for your DQ Job.
- The column of reference.
- The link ID.

For further reading on sensitive information, refer to Data Discovery in the Rule Discovery section.

Viewing break records

To view rule break records, navigate to the Breaks tab on the Rule Builder page. Rules with break records have associated link IDs that link back to the original data set. All remediation for data quality issues is performed outside the Collibra Data Quality app.

Steps

1. In Explorer, select a table and create a DQ Job. >> The DQ Job page opens.
2. In the Scope section, select the **Add Link Back to Source** checkbox. >> A new column, Link ID, appears.
3. Select the columns that represent the **primary key(s)** of your data set. >> By selecting multiple checkboxes, you can create a composite key.
4. Run your DQ Job.
    a. Click **Build Model**.
    b. Click **Save/Run**.
    c. Verify the information on the **Register** page.
    d. Click **Estimate Job** and then click **Run**. >> Your DQ Job is sent to the Jobs page.
5. Open your DQ Job.
    a. Open the **Jobs** page.
    b. Select your **DQ Job** from the list. >> Your DQ Job opens.
6. Apply a rule.
    a. In the metadata box, click **Rules**. >> The Rule Builder opens.
    b. Select a **rule type**. **Note:** All Simple and Freeform rules are eligible for Link ID.
7. Re-run your DQ Job.
    a. On the Findings page, click the **DQ Job** tab.
    b. Verify the **run command**.
    c. Click **Run DQ Job**. >> Your DQ Job is submitted to the Jobs queue.
8. Open your DQ Job.
    a. Open the **Jobs** page.
    b. Select your **DQ Job** from the list. >> Your DQ Job opens.
9. View your rule breaks.
    a. In the metadata box, click **Rules**. >> The Rule Builder opens.

b. Click the **Breaks** tab. >> A table displays a row for each record in violation of the rule you set. The Link ID column lets you identify broken records within your data set and mark them for remediation outside the DQ application.
c. Export the break records. Supported file formats include Excel and CSV.

Notebook

```
val opt = new OwlOptions()
opt.runId = "2018-02-24"
opt.dataset = "orders"
opt.linkId = Array("transaction_id", "trans_time")
```

Command Line

```
./owlcheck -ds orders \
-rd "2018-02-24" \
-linkid transaction_id,trans_time
```

> Note For rules to use linkID, the columns need to be present in the select statement (either select * or select specific column names). All Simple rules are eligible for linkID and Freeform rules need to contain the columns in the projection part of the SQL statement.

Activity Usage

Activity	Supported	Description
SHAPE	YES	One example of each shape issue will have a link back to the corrupt record for remediation.
OUTLIER	YES	Each outlier will have a link back to the detected record for remediation. If you apply a limit you will only get the limited amount. Not on categorical.

Activity	Supported	Description
DUPE	YES	Each duplicate or fuzzy match will have a link back to the record for remediation.
SOURCE	PARTIAL	Each source record that has a cell value that doesn't match to the target will have a link for remediation. SOURCE will not have links for row counts and schema as these are not record level findings.
RULE	YES	Break records for Freeform and Simple rule types will be stored (any records that did not meet the condition of the RULE will be provided with the linkID columns). These are stored as delimited strings in the rule_breaks table along with the dataset, run_id and rule name. Please note when using Freeform SQL the linkID columns should be part of the select statement. LinkID columns should be unique identifiers.
BEHAVIOR	NO	This class of data change is when a a section of your data is drifting from its normal tendency there is no 1 record to link.
SCHEMA	NO	This class of data change is at a schema/data set level there are no records to link.
RECORD	PARTIAL	In some cases when a record is added or removed it may be available for linking.
PATTERN	NO	Patterns are not always a direct link. This item is still under performance review.

Notebook API Example

```
+-----------+---------+------+------+-----+----------------
+---------------+
|     dataset|     runId|fieldNm|
format|count|          percent| transaction_id|
+-----------+---------+------+------+-----+----------------
+---------------+
|      order |2018-02-
24|  fname|xxxx'x.|    1|7.142857142857142|t-1232         |
+-----------+---------+------+------+-----+----------------
+---------------+
```

```
owl.getShapesDF
```

Rest API Example

When supplying a linkID, Collibra naturally excludes this field from most activities, meaning a unique ID or primary key column can not be duplicative or it would not be the primary key. Because of this, it is not evaluated for duplicates. The same is true for Outliers and Shapes, as a large sequence number or other variations might trigger a false positive when this column is denoted to be simply for the purpose of linking uniquely back to the source. If you also want to evaluate this column and link it, create a derived column with a different name and Collibra Data Quality will naturally handle both cases.

```
owl.getShapes
owl.getDupes
owl.getOutliers
owl.getRuleBreaks
owl.getSourceBreaks
```

getRules()

```
----Rules----
+-----------------+----------+--------------------+-------------
-----+------+
|          dataset|     runId|              ruleNm|         rule-
Value|linkId|
+-----------------+----------+--------------------+-------------
-----+------+
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|fname like
'Kirk' |  c-41|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|fname like
'Kirk' |  c-42|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|fname like
'Kirk' |  c-43|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|fname like
'Kirk' |  c-44|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|fname like
'Kirk' |  c-45|
|dataset_outlier_3|2018-02-24|if_email_is_
valid...|             email|  c-31|
|dataset_outlier_3|2018-02-24|if_email_is_
valid...|             email|  c-33|
|dataset_outlier_3|2018-02-24|if_zip_is_valid_
Z...|               zip|  c-40|
+-----------------+----------+--------------------+-------------
-----+------+
```

getDupes()

First split on ~~ then if you have a multiple part key split on ~|.

```
----Dupes----
+-----------------+----------+-----+--------------------+-------
---+
|          dataset|     runId|score|                 key|    lin-
kId|
+-----------------+----------+-----+--------------------+-------
---+
|dataset_outlier_3|2018-02-24|  100|9ec828d5194fa397b...|c-
45~~c-36|
|dataset_outlier_3|2018-02-24|  100|1f96274d1d10c9f77...|c-
45~~c-35|
|dataset_outlier_3|2018-02-24|  100|051532044be286f99...|c-
45~~c-44|
|dataset_outlier_3|2018-02-24|  100|af2e96921ae53674a...|c-
45~~c-43|
|dataset_outlier_3|2018-02-24|  100|ad6f04bf98b38117a...|c-
45~~c-42|
|dataset_outlier_3|2018-02-24|  100|1ff7d50a7a9d07d02...|c-
45~~c-41|
|dataset_outlier_3|2018-02-24|  100|6ed858ed1f4178bb0...|c-
45~~c-40|
|dataset_outlier_3|2018-02-24|  100|d2903703b348fb4cb...|c-
45~~c-39|
|dataset_outlier_3|2018-02-24|  100|24bf54412de1e720d...|c-
45~~c-38|
|dataset_outlier_3|2018-02-24|  100|7a7ce0beb41b39564...|c-
45~~c-37|
+-----------------+----------+-----+--------------------+-------
---+
```

getRuleBreaks()

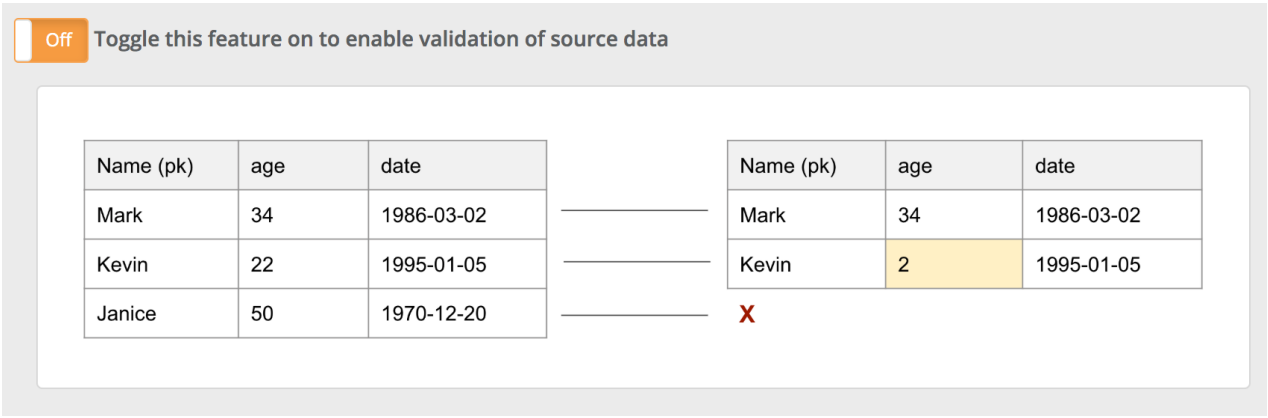
The getRuleBreaks endpoint retrieves all broken records within your data set. There is no size limit to this API.

```
----Rule-Breaks----
+-----------------+----------+--------------------+------+
|          dataset|     runId|              ruleNm|linkId|
+-----------------+----------+--------------------+------+
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|  c-41|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|  c-42|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|  c-43|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|  c-44|
|dataset_outlier_3|2018-02-24|     fname_like_Kirk|  c-45|
|dataset_outlier_3|2018-02-24|if_email_is_valid...|  c-31|
|dataset_outlier_3|2018-02-24|if_email_is_valid...|  c-33|
|dataset_outlier_3|2018-02-24|if_zip_is_valid_Z...|  c-40|
+-----------------+----------+--------------------+------+
```

DQ Job Validate Source

Reconciliation

Commonly data driven organizations have a need to ensure that two tables or a table and file match. This match might be a daily reconciliation or any snapshot in time. Collibra DQ calls this Source to Target or Left to Right matching. It covers row differences, schema differences and all cell values.



Impala/Hive -> DB2

Below is an example of comparing a table in DB2 to the same table in Impala.

```
./owlcheck \
-lib "/home/install/owl/drivers/db2" \
-cxn db2 \
-q "select * from OWLDB2.NYSE_STOCKS where TRADE_DATE = '${rd}'
" \
-ds NYSE_STOCKS_VS \
-rd "2018-01-10" \
-vs \
-valsrckey SYMBOL \
-validatevalues \
-h $host/owltrunk \
-srcq "select * from nyse where TRADE_DATE = '${rd}' " \
-srccxn impala-jdbcuser \
-libsrc /home/isntall/owl/drivers/hivedrivers \
-jdbcprinc jdbcuser@CW.COM -jdbckeytab /tmp/jdbcuser.keytab \
-owluser admin \
-executorcores 4 -numexecutors 6 -executormemory 4g -driver-
memory 4g -master yarn -deploymode cluster \
-sparkkeytab /home/install/owl/bin/user2.keytab \
-sparkprinc user2@CW.COM
```

DB2 -> Hive (Native)

Most databases only expose data through a JDBC connection but Hive offers a second path which does not require a JDBC connection. Hive has the ability to push down its processing to the local worker nodes and read directly from disk in the case when the processing is happening locally on a cluster. If your processing is not happening local to the cluster then you must use HiveJDBC. Take note of the -hive flag.

```
./owlcheck \
-hive \
-q "select * from nyse" \
-ds hiveNativeNyse \
-rd "2019-10-01" \
-vs \
-valsrckey exch,symbol,trade_date \
-validatevalues \
-srcq "select * from OWLDB2.NYSE_STOCKS" \
-srccxn db2 -libsrc /home/install/owl/drivers/db2 \
-numexecutors 2 -executormemory 5g -drivermemory 4g -master yarn
-deploymode cluster \
-sparkkeytab /home/install/owl/bin/user2.keytab -sparkprinc
user2@CW.COM
```

MySQL -> Oracle

This example compares the entire table instead of just a single day. Notice the 3 part valsrckey EXCH,SYMBOL,TRADE_DATE. Adding the date field ensures our key is unique and won't create a cartesian product. If the goal was to compare day over day with Oracle make sure to add TO_DATE('YYYY-MM-DD', '2019-10-01') to the where clause.

```
./owlcheck \
-lib /home/install/owl/drivers/mysql/ \
-cxn mysql \
-q "select * from lake.nyse" \
-ds lake.nyse \
-rd 2019-10-01 \
-vs \
-valsrckey EXCH,SYMBOL,TRADE_DATE \
-validatevalues \
-sparkkeytab /home/install/owl/bin/user2.keytab \
-sparkprinc user2@CW.COM \
-srcq "select * from SYSTEM.NYSE" \
-srccxn oracle \
-libsrc /home/danielrice/owl/drivers/oracle/
-numexecutors 2 -executormemory 5g -drivermemory 4g -master yarn
-deploymode cluster \
```

File -> MySQL Table

Taking a file and loading it into a staging table or final table is a common part of every ETL process. However it is extremely common that the file values do not match or coherence into the table properly and these silent errors are usually not caught until a business user sees the data far long down stream.

```
./owlcheck \
-ds lake.nyse \
-rd 2019-10-01 \
-cxn "mysql" \
-q "select * from lake.nyse" \
-vs \
-valsrckey EXCH,SYMBOL,TRADE_DATE \
-validatevalues \
-srcfile "hdfs:///user/source/nyse.csv" \
-srcd "," \
-lib /home/install/owl/drivers/mysql/ \
-sparkkeytab /home/install/owl/bin/user2.keytab \
-sparkprinc user2@CW.COM \
-numexecutors 2 -executormemory 5g -drivermemory 4g -master yarn
-deploymode cluster \
```

File -> File

DQ compares a File to a File. This is common in landing zones and staging areas where a file might be moved or changed and you need to know if anything changed or is incorrect.

```
./owlcheck \
-ds lake.nyse \
-rd 2019-10-01 \
-f "hdfs:///user/target/nyse.csv" \
-d "," \
-vs \
-valsrckey EXCH,SYMBOL,TRADE_DATE \
-validatevalues \
-srcfile "hdfs:///user/source/nyse.csv" \
-srcd "," \
-sparkkeytab /home/install/owl/bin/user2.keytab \
-sparkprinc user2@CW.COM \
-numexecutors 2 -executormemory 5g -drivermemory 4g -master yarn
-deploymode cluster \
```

DQ Job 43M rows

Collibra DQ commonly benchmarks on large daily datasets. In this case, a 43 million row table with 12 columns completes in under 6 mins (5:30). The best balance for this dataset was 3 executors each with 10G of ram.

```
./owlcheck \
-u user -p password \
-c jdbc:mysql://owldatalake.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com:3306 \
-q "select * from silo.account_large where acc_upd_ts > '2018-
02-01 05:0:00'" \
-rd 2019-02-02 \
-ds account_large \
-dc acc_upd_ts \
-corroff \
-histoff \
-driver com.mysql.cj.jdbc.Driver \
-lib "/home/ec2-user/owl/drivers/mysql/" \
-master yarn \
-deploymode client \
-numexecutors 3 \
-executormemory 10g \
-histoff -corroff -loglevel DEBUG -readonly
```

> Note *Not all DQ features were turned on during this run. On large datasets it is worth it to consider limiting the columns, DQ-features, or lookbacks if they are not of interest.*

Add Date Column

Example

```
./owlcheck \
-ds "datataset_date_column" \
-rd "2019-07-01" \
-f "/Users/Downloads/csv2/2019010.csv" \
-adddc
```

> Note Add date column will use the run date supplied and add a date column named **DQ_RUN_ID**.

```
-adddc
```

This is used when you are using datasets that do not contain a date column or a malformed date string.

Known Limitations

This feature is only available for files, not for database tables.

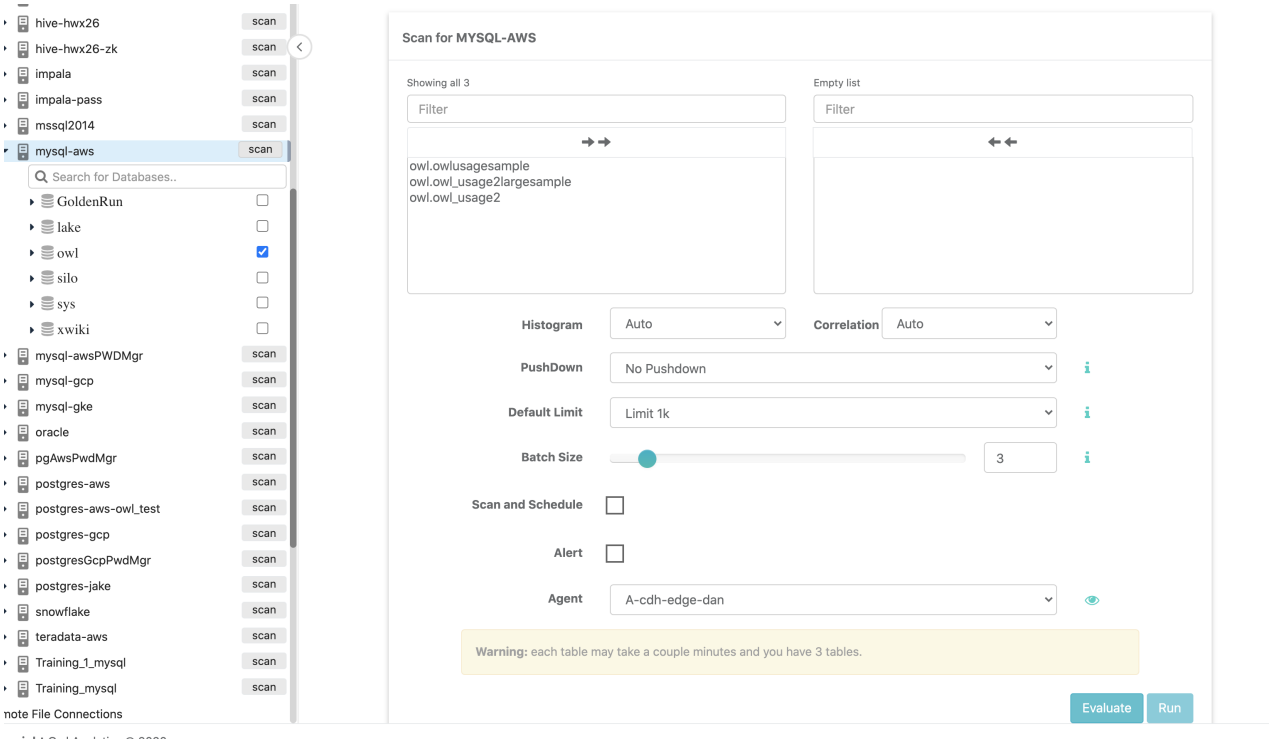
AutoProfile

AutoProfile allows you to select a set of databases and tables to quickly be cataloged. Each selected table will be profiled and added to the Collibra DQ Catalog via the selected agent. Various parameters like Alerts, Job Schedules, limits, and more can also be set.

When you expand a datasource in the **Explorer** page, you're given a list of possible databases and their associated tables. **AutoProfile** is triggered when you select the ones you want and hit scan. This will take you to a separate page that allows you to configure the various AutoProfile parameters.

A SparkSubmit will be launched for each table, so make sure the agent configuration is reasonable and the box has enough resources to handle each job.

Global Parameters



Histogram and **Correlation**: Enable or Disable

PushDown: Set metrics that will be run against the entire table, ignoring limit values.

Default Limit: The default row limit to set for each table to be scanned.

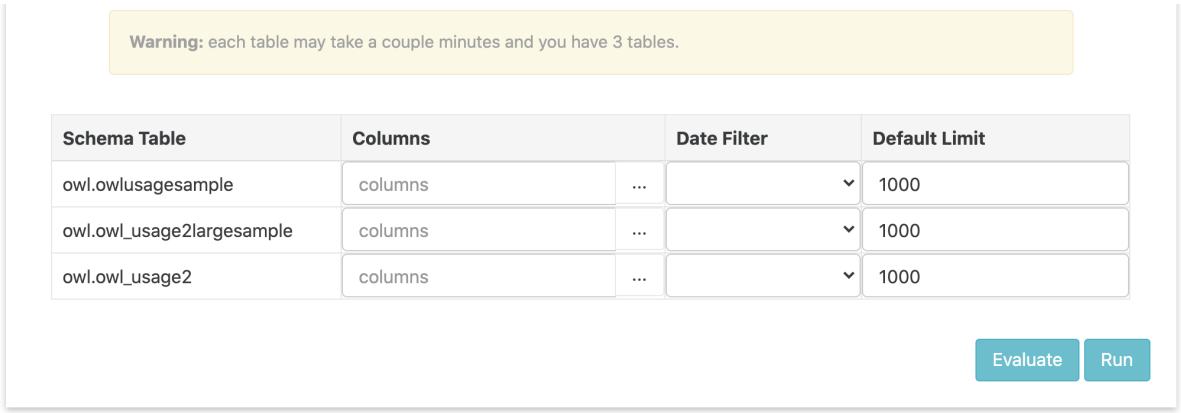
Batch Size: The number of concurrent SparkSubmit jobs the agent will be allowed to run.

Scan and Schedule : Enable scheduling.

Alert: Enable email alerts.

Agent: The agent under which jobs will run.

Per Table Parameters



Columns: The columns to select from (if none are specified, we assume all columns to be selected).

Date Filter: The date column to be used as the `runDate` parameter when the job runs.

Default Limit: Per table limits.

Cloudera CLASSPATH

What is a CLASSPATH?

A CLASSPATH is essentially a list of jars that get injected into a JVM on the start of a job execution. Like many applications, Spark can have jars injected when a job is run. Cloudera has defined a list of predefined jars (rightfully called classpath.txt):

```
/etc/spark2/conf/classpath.txt
```

That will get injected whenever Spark is called. Here is an example list of jars as defined within a cluster we have stood up @ Collibra DQ:

```
[danielrice@cdh-edge ~]$ cat /etc/spark2/conf/classpath.txt
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/activation-1.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/aopalliance-1.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/apacheds-
i18n-2.0.0-M15.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/apacheds-
kerberos-codec-2.0.0-M15.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/api-asn1-
api-1.0.0-M20.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/api-util-
1.0.0-M20.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/asm-3.2.-
jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/avro-
1.7.6-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/aws-java-
sdk-bundle-1.11.134.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/azure-
data-lake-store-sdk-2.2.9.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
beanutils-1.9.2.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
beanutils-core-1.8.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
codec-1.4.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
codec-1.9.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
configuration-1.6.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
daemon-1.0.13.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
digester-1.8.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
el-1.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
logging-1.2.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
math-2.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
math3-3.1.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/commons-
net-3.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/core-
3.1.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/curator-
client-2.7.1.jar
```

```
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/curator-
framework-2.7.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/curator-
recipes-2.7.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/disruptor-3.3.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/findbugs-
annotations-1.3.9-1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/guava-
11.0.2.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/guava-
12.0.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/guice-
3.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
annotations-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
ant-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
archive-logs-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
archives-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
auth-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
aws-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
azure-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
azure-datalake-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
common-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
datajoin-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
distcp-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
extras-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
gridmix-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
hdfs-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
hdfs-nfs-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-app-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-common-2.6.0-cdh5.16.1.jar
```

```
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-core-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-hs-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-hs-plugins-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-jobclient-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-nativetask-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-client-shuffle-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
mapreduce-examples-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
nfs-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
openstack-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
rumen-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
sls-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
streaming-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-api-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-applications-distributedshell-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-applications-unmanaged-am-launcher-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-client-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-common-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-registry-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-server-applicationhistoryservice-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-server-common-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-server-nodemanager-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-server-resourcemanager-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hadoop-
yarn-server-web-proxy-2.6.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hamcrest-
core-1.3.jar
```

```
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
annotations-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-cli-
ent-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-com-
mon-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
examples-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
external-blockcache-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
hadoop-compat-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
hadoop2-compat-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-it-
1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-pre-
fix-tree-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-pro-
cedure-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-pro-
tocol-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
resource-bundle-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
rest-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
rsgroup-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
server-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
shell-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hbase-
thrift-1.2.0-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/high-
scale-lib-1.1.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hsqldb-
1.8.0.10.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/htrace-
core-3.2.0-incubating.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/htrace-
core4-4.0.1-incubating.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/httpclient-4.2.5.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/httpcore-
4.2.5.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/hue-plu-
gins-3.9.0-cdh5.16.1.jar
```

```
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jackson-
annotations-2.2.3.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jackson-
core-2.2.3.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jackson-
core-asl-1.8.10.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jackson-
databind-2.2.3-cloudera.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jackson-
jaxrs-1.8.10.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jackson-
mapper-asl-1.8.10-cloudera.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jackson-
xc-1.8.10.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jamon-
runtime-2.4.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jasper-
compiler-5.5.23.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jasper-
runtime-5.5.23.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/java-xml-
builder-0.4.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/javax.inject-1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jaxb-api-
2.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jaxb-api-
2.2.2.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jaxb-
impl-2.2.3-1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jcodings-
1.0.8.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jets3t-
0.9.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jettison-
1.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jettison-
1.3.3.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jline-
2.11.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/joni-
2.1.2.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jruby-
cloudera-1.0.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jsch-
0.1.42.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jsp-2.1-
6.1.14.jar
```

```
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jsp-api-
2.1-6.1.14.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jsp-api-
2.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/jsr305-
3.0.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/leveldbjni-all-1.8.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/log4j-
1.2.16.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/log4j-
1.2.17.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/metrics-
core-2.2.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/metrics-
core-3.0.2.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/microsoft-windowsazure-storage-sdk-0.6.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/mockito-
all-1.8.5.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/netty-
3.10.5.Final.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/netty-
all-4.0.50.Final.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/okhttp-
2.4.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/okio-
1.4.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/paranamer-2.3.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/protobuf-
java-2.5.0.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/slf4j-
api-1.7.5.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/slf4j-
log4j12-1.7.5.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/snappy-
java-1.0.4.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/spark-
1.6.0-cdh5.16.1-yarn-shuffle.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/spymemcached-2.11.6.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/stax-api-
1.0-2.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/xercesImpl-2.9.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/xml-apis-
1.3.04.jar
```

```
/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/jars/xmlenc-
0.52.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/jars/zookeeper-3.4.5-cdh5.16.1.jar
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/lib/hadoop/LICENSE.txt
/opt/cloudera/parcels/CDH-5.16.1-1.cd-
h5.16.1.p0.3/lib/hadoop/NOTICE.txt
/usr/java/jdk1.8.0_131/lib/tools.jar
```

At each execution of any Spark job (including the use of spark-submit) this list of jars above will automatically get loaded.

What is a JAR?

A jar file is essential a compressed list of classes and methods. It is important to note that when jar files are built they will typically have an associated version number.

Someone can look at the contents of a jar file by executing:

```
jar -tvf phoenix-4.13.1-HBase-1.3-client.jar
```

Or you can wrap the above in a for loop that will look at the contents of every jar that might contain a method you are looking for.

```
for i in `ls -1 *.jar`;do jar -tvf $i | grep -i
htrace/trace;echo $i;done;
```

Common issues that can occur with CLASSPATH's

> Note **Caused by: java.lang.NoClassDefFoundError: org/apache/htrace/Trace** at org.apache.hadoop.hbase.zookeeper.RecoverableZooKeeper.exists (RecoverableZooKeeper.java:218) at org.apache.hadoop.hbase.zookeeper.ZKUtil.checkExists(ZKUtil.java:481) at org.apache.hadoop.hbase.zookeeper.ZKClusterId.readClusterIdZNode (ZKClusterId.java:65) at org.apache.hadoop.hbase.client.ZooKeeperRegistry.getClusterId (ZooKeeperRegistry.java:86) at org.apache.hadoop.hbase.client.ConnectionManager$HConnectionImplementation.retrieveClusterId(ConnectionManager.java:850) at org.apache.hadoop.hbase.client.ConnectionManager$HConnectionImplementation.<init>(ConnectionManager.java:635)

When a situation like this occurs it means that a method cannot be found in the classpath for the job that is trying to execute. This can indicate a couple things:

1. The job cannot find a jar file that contains the method flagged (in the example above the org/apache/htrace/Trace method).
2. Sometimes different versions of the same jar file gets loaded and the first jar loaded will always win. Older jars that get loaded first may not have a method defined in new jars.

At DQ, we have solved CLASSPATH / CLASSLOAD issues by automatically injecting jars defined in our ***owl/libs*** directory, and allowing users the ability to simply toggle loading them or not.

Column Matching

How much is your redundant data costing you?

Reclaim Gigabytes of Redundant Data

As data engineers, first we copy files into a landing zone, next we load the files into a staging area. After that we transform (ETL) the data into the final table. Soon that same data is copied to a lake for other groups to run analytics on. Eventually a group of analysts

will need the data in another format and a data engineer will copy the data in a newly joined or transposed fashion. Sounds familiar?

The result is the same data or similar columns of the same data being copied many times. **The answer**: *Buy more hardware*... could be OR *run a Collibra DQ health report* and gain an understanding of how much data could be removed, reclaiming disk space and instantly seeing a return on investment after clicking the button.

Tabular breakdown of percentage of fingerprint matches

dataset_1	dataset_2	col_1	col_2	perMatch
silo.account	silo.user_account	networth	acc_branch	16
silo.account	silo.user_account	networth	networth	100
silo.account	silo.user_account	acc_name	acc_name	88
silo.account	silo.user_account	acc_name	acc_branch	0
silo.account	silo.user_account	acc_name	networth	0
silo.account	silo.user_account	acc_branch	acc_name	0
silo.account	silo.user_account	acc_branch	acc_branch	100
silo.account	silo.user_account	acc_branch	networth	14
silo.user_account	silo.user_account	acc_name	acc_branch	0
silo.user_account	silo.user_account	acc_name	networth	0
silo.user_account	silo.user_account	acc_branch	networth	14

Sometimes its not as simple as comparing two tables from the same database. DQ allows a technical user to setup multiple DB connections before executing an owl health check.

```
import com.owl.common.Props
import com.owl.core.Owl

val c1 = new Connection()
c1.dataset = "silo.account"
c1.user = "user"
c1.password = "pass"
c1.query = "select id, networth, acc_name, acc_branch from
silo.account limit 200000"
c1.url = "jdbc:mysql://owldatalake.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com:3306"

val c2 = new Connection()
c2.dataset = "silo.user_account"
c2.user = "user"
c2.password = "pass"
c2.query = "SELECT acc_name, acc_branch, networth FROM silo.ac-
count limit 200000"
c2.url = "jdbc:mysql://owldatalake.chzid9w0hpyi.us-east-1.rd-
s.amazonaws.com:3306"

val props = new Props()
props.dataset = "colMatchTest1"
props.runId = "2017-02-04"
props.connectionList = List(c1,c2).asJava
props.colMatchBatchSize = 2
props.colMatchDurationMins = 3

val matchDF = new Owl(props).colMatchDF
matchDF.show

matchDF.createOrReplaceTempView("matches")
```

High level view of data overlap

Date Time Variable Options

Date Time Variables can enhance your query and file date templating and variable options. This allows easier scheduling and programmatic templating for common date variables.

Key	Function
${rd}	Replaces with -rd values in CMD. For example, 2023-01-20.
${rdEnd}	Replaces with -rdEnd values in CMD.
${yyyy}	Replaces with 4-digit year portion of -rd values in CMD. For example, 2023.
${yy}	Replaces with 2-digit year portion of -rd values in CMD. For example, 23.
${M}	Replaces with 1-digit month portion of -rd values in CMD.
${MM}	Replaces with 2-digit month portion of -rd values in CMD.
${MMM}	Replaces with 3 letter month name portion of -rd values in CMD For example, Jan, Jul, Dec.
${MMMM}	Replaces with long month name portion of -rd values in CMD For example, January, July, December.
${d}	Replaces with 1 digit day portion of -rd values in CMD.
${dd}	Replaces with 2 digit day portion of -rd values in CMD.
${HH}	Replaces with 2 digit hour portion of -rd values in CMD.
${KK}	Replaces with 1 digit hour portion of -rd values in CMD.
${mm}	Replaces with 2 digit minute portion of -rd values in CMD.
${ss}	Replaces with 2 digit second portion of -rd values in CMD.

Deploy Mode

Yarn and Cluster

Deploy Mode Client

```
--deploy-mode client
```

Deploy Mode Cluster

```
--deploy-mode cluster
```

Job Stuck in ACCEPTED State

yarn.Client: Application report for application_1557720962505_0085 (state: ACCEPTED)
yarn.Client: Application report for application_1557720962505_0085 (state: ACCEPTED).

If running in cluster mode make sure you are passing in the below.

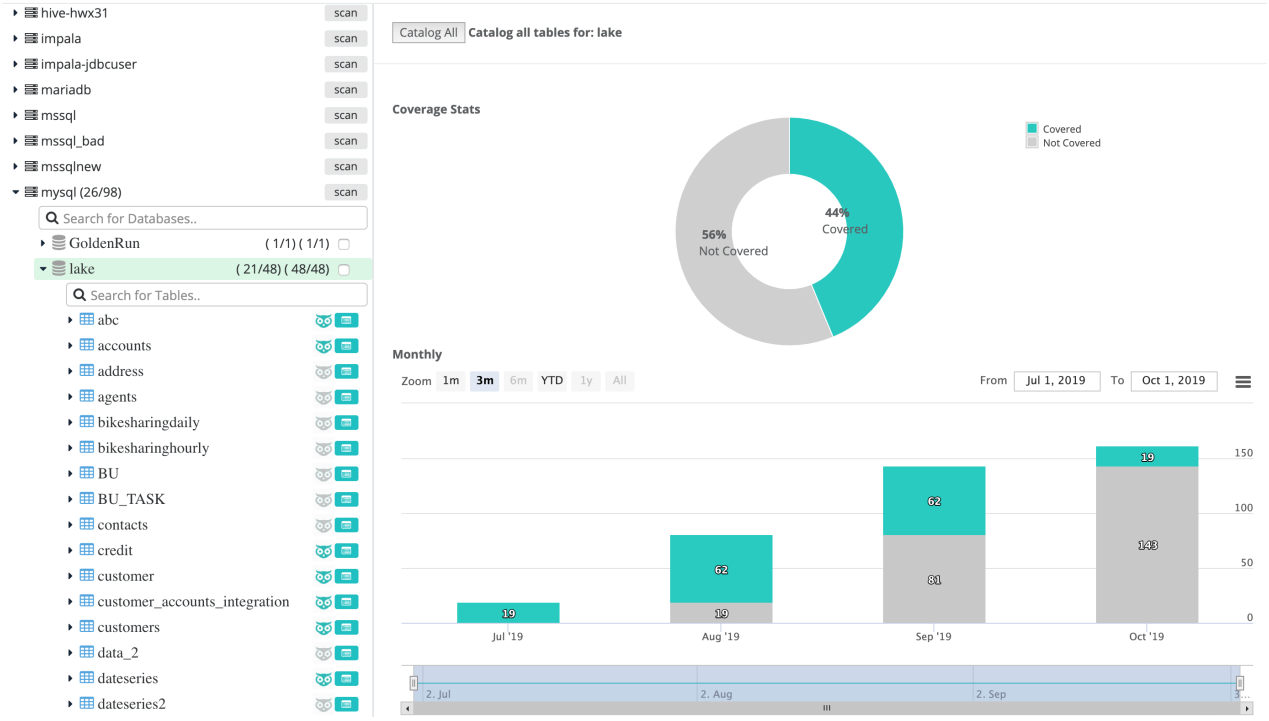
```
--deploy-mode cluster
--master yarn          # or spark master
-h 123.45.6.77:2181    # host to owl metastore
```

Explorer (advanced)

Explore Database Connections and File Systems

Use the explorer tab to quickly see which tables are cataloged with Collibra DQ (the square catalog icon) and which have DQ's quality protection (the DQ icon).

Below you will see 48/48 database tables have been cataloged with DQ but only 21/48 have an DQCheck. This means that this particular database schema is 44 percent protected from future DQ issues.



DQ coverage over time

As you add more datasets to Collibra DQ, you will see your bar chart increase over time and the donut chart fill in with more coverage.

Job Estimator

Before firing off a large ML job it can be helpful to understand the amount of cores and ram that the job requires to run efficiently. Right sizing jobs is often the best way to get the best performance out of each run. Click the [Auto Estimate] button for quick stats for both sizing and estimated runtime information.

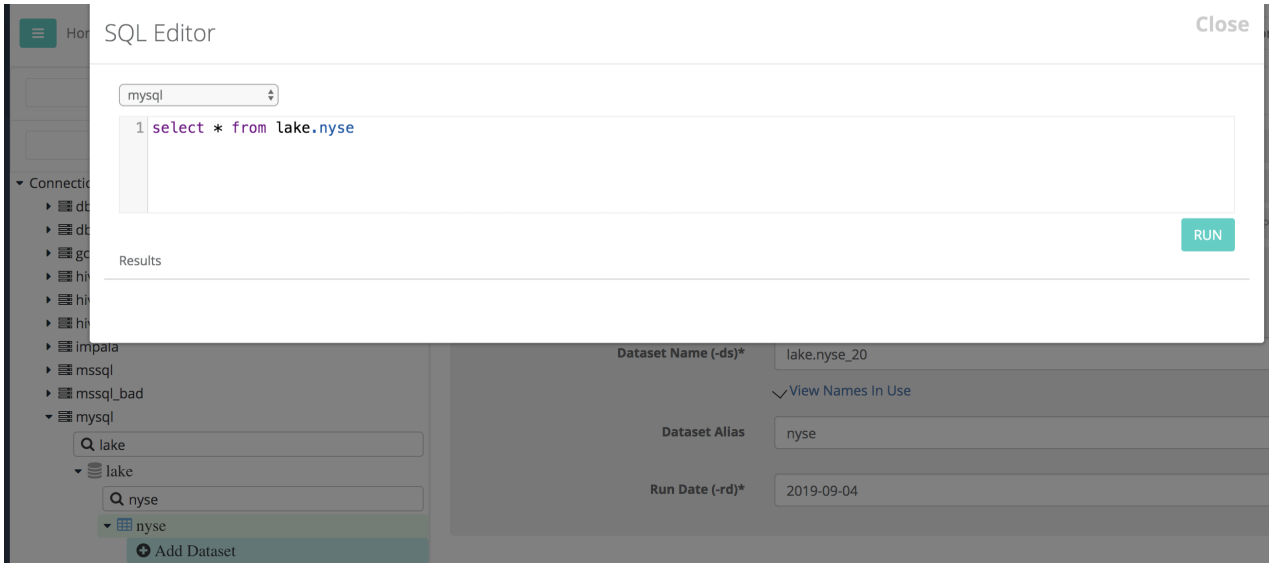
Dynamic allocation

Many clusters offer the ability to scale up and down job containers. If Dynamic Allocation is turned on you may not need or desire DQ's recommended `num-executors` or `executor-memory`. However, in our testing right sizing the job before executing is both faster and a healthy habit. Faster, because there is less orchestration and context switching while the job is doing work; we've minimized time spent running out of space and

having to reallocate and shuffle to a new container. Healthier, because it give the user real-time feedback on the cost of each feature and the ability to control the cost benefit analysis.

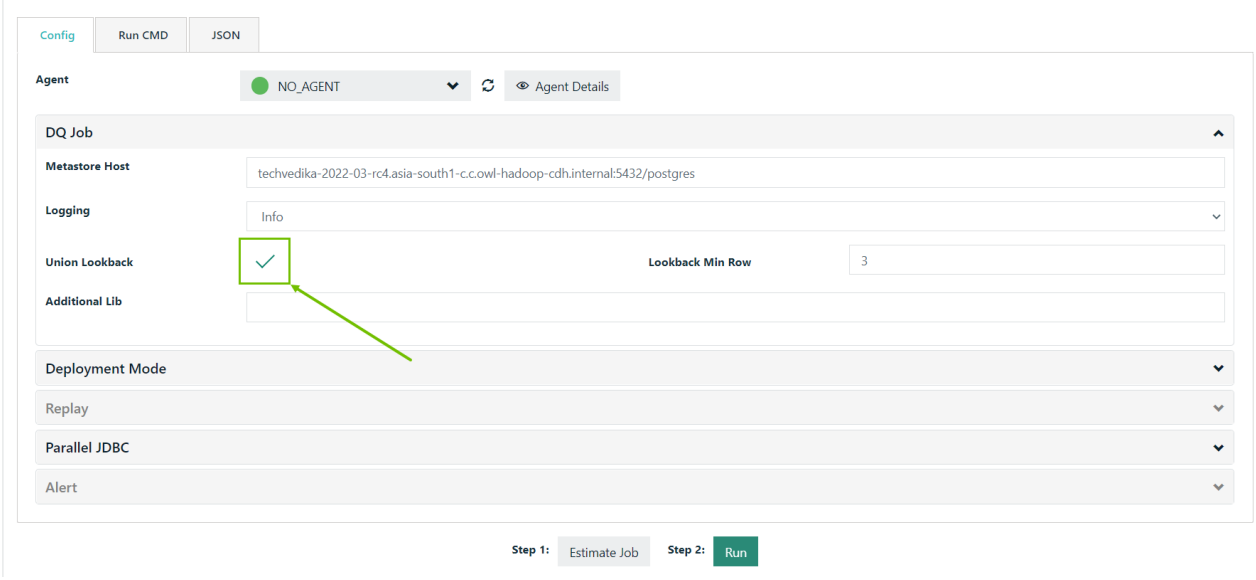
SQL Editor

Automatically tracks to the connection, database and table in the explorer and provides a quick way to ask the database simple questions like, counts, groupings and specific clauses.



File Look Back

As of 2021.11, this option is exposed in the Explorer under the Collibra DQ Job section. Users can persist (save) this option by clicking the Union Lookback checkbox.



Union Lookback (-fllb)

Union Lookback, or File Lookback (-fllb) as it is also known, is used with deep learning or pattern matching. In the example below, it is used with deep learning.

File Lookback is used to check DQ Check history for previous files.

```
-fllb
```

This is often used with files and in conjunction with -adddc in cases where a date column is not in an ideal format or you do not have a date column on the given dataset.

Despite the name, this can be used with file or database storage formats.

> Note File look back (-fllb) should only be used when a SQL layer is not available. This is considered for advanced use cases, but may not be suitable for all file types and folder structures. Best practice is to expose a date signature somewhere in the file or directory naming convention.

Example

```
-ds "demo_lookback" \
-rd "2017-07-29" \
-lib "/opt/owl/drivers/mysql" \
-cxn "mysql" \
-q "select  * from lake.dateseries where DATE_COL = '2017-07-29'
" \
-dc DATE_COL \
-dl \
-dlkey sym \
-dllb 4 \
-fllb
```

> Note This look back will load your past 4 runs as your historical training set

Fullfile Lookback (-fullfile)

Like Union Lookback, Fullfile Lookback (-fullfile) is used with deep learning and pattern matching.

Fullfile Lookback uses the entire file for lookbacks instead of just filequery.

—

Filter & Filter Not

Filter & Filter Not is similar to a grep for limiting a dataframe to rows containing a substring.

> Note This feature should only be used when -q (query) and -fq (filequery) are not applicable. This feature is primarily used with raw files for limited filtering and not advanced conditional logic.

Example

```
./owlcheck \
-ds "dataset_name" \
-rd "2018-07-23 10" \
-d "," \
-f "/Users/Documents/file.csv" \
-filter "2018-07-23"
```

> Note **-filter "2018-07-23"**
> If file.csv contained multiple strings, but you only wanted rows containing "2018-07-23".

The inverse

> Note **-filternot "2018-07-23"**
> To exclude rows containing "2018-07-23".

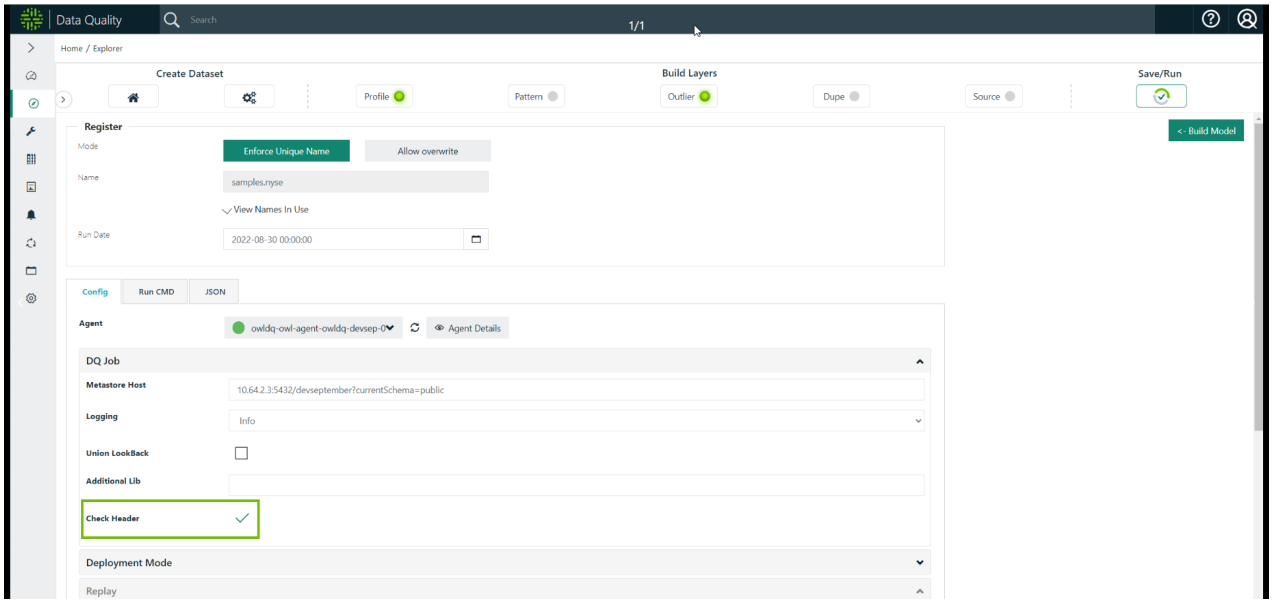
Header Check

Header Check lets you toggle column name detection on and off so column names containing special characters are not detected as schema changes. This is configurable with the **Check Header** checkbox or from the command line.

Configuring with the Check Header checkbox

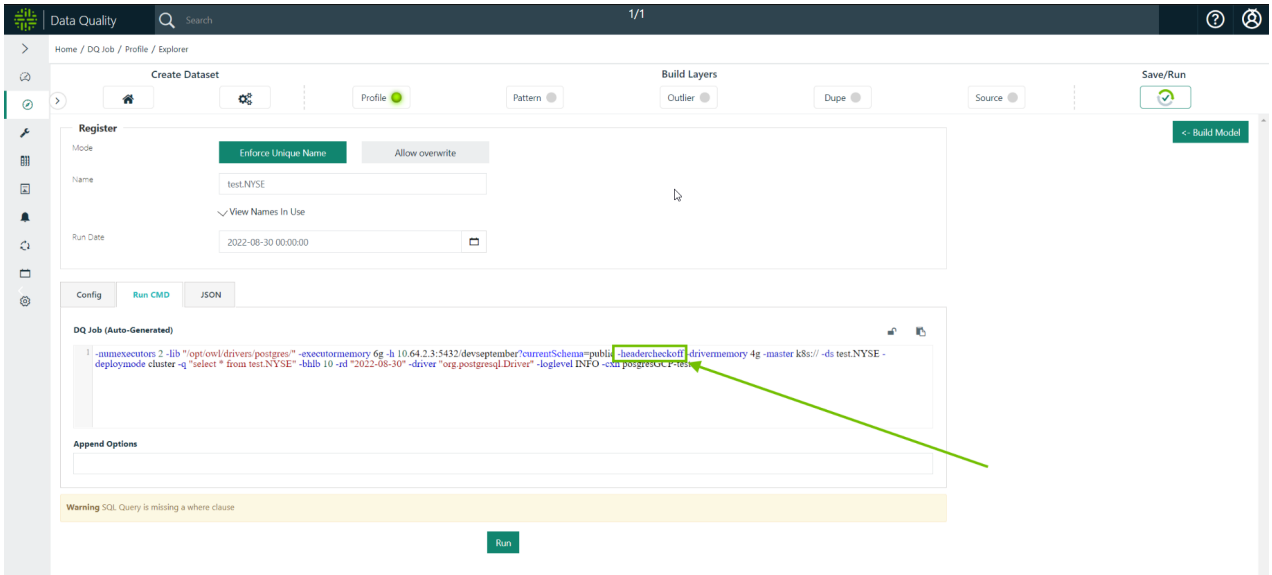
The Check Header checkbox is checked by default. When it is checked, schema findings do not display when detected.

To disable the header check, uncheck the Check Header checkbox. This allows the schema findings to display when detected.



Configuring from the command line

Header checks are enabled by default and do not appear in the command line when enabled. To disable header checks from the command line, click the lock icon to unlock the command line and use the -headercheckoff variable, as shown in the screenshot below. When you are done editing the command line, click the lock icon again to lock the command line; then, click **Run**.



Multiple Pattern Relationships

Run more than one relationship through pattern matching.

Example

```
./owlcheck \
-ds "fpg_multiple" \
-rd "2018-10-04" \

-cxn "postgres" \
-lib   "/opt/owl/drivers/postgres/" \
-q "select * from public.fpg_accounts where d_date = '2018-10-
04'" \

-fpgon \
-fpgdc "d_date" \
-fpglb "4" \
-fpgmulti "id,ssn_num=first_name,email|id,ssn_num=first_name,-
gender|id,ssn_num=last_name"
```

> Note Instead of -fpgkey and -fpgcol
> key1=cols1|keys2=cols2
> Enter multiple key=cols combinations separated by a pipe.

```
-fpgmulti "id,ssn_num=first_name,email|id,ssn_num=first_name,-
gender"
```

Nulls in Datasets

Example

```
./owlcheck \
-ds "datataset_date_column" \
-rd "2019-07-01" \
-f "/Users/Downloads/csv2/2019010.csv" \
-zfn \
-nulls "N.A."
```

Zero if null will replace null values with zero.

```
-zfn
```

To replace characters that represent a null to actual null values.

```
-nulls "N.A."
```

Spark-shell Sample

```
./bin/spark-shell --jars /opt/owl/bin/owl-core-trunk-jar-with-dependencies.jar,/opt/owl/drivers/postgres/postgresql-42.2.5.jar --deploy-mode client --master local[*]
```

Import lib's, if you get a dependency error, please import a second time.

```
import com.owl.core.util.{OwlUtils, Util}

import com.owl.common.domain2.OwlCheckQ

import com.owl.common.options._
```

Set up connection parameters to the database we want to scan if you don't already have a dataframe.

```
val`` ``url`` ``= "jdbc:postgresql://xxx.xxx.xxx.xxx:xxxx/db?currentSchema=schema"
val`` ``connProps``=``Map("driver" -> "org.postgresql.Driver","user" -> "user","password" -> "pwd","url" ->`` ``url","dbtable" -> "db.table")
```

Create a new OwlOptions object so we can assign properties.

```
val opt = new OwlOptions()
```

Set up variables for ease of re-use.

```
val dataset = "nyse_notebook_test_final"

val runId = "2017-12-18"
```

```
var date = runId

var query = s"""select * from <table> where <date_col> = '$date' """

val pgDatabase = "dev"val pgSchema = "public"
```

Set OwlOptions values to the metastore.

```
opt.dataset`` ``= datasetopt.runId`` ``= runIdopt.host`` ``=
"xxx.xxx.xxx.xxx"opt.pgUser`` ``= "xxxxx"opt.pgPassword`` ``=
"xxxxx"opt.port`` ``= s"5432/$pgDatabase?currentSchema=$pgSchema"
```

Create a connection, build the dataframe, register and run.

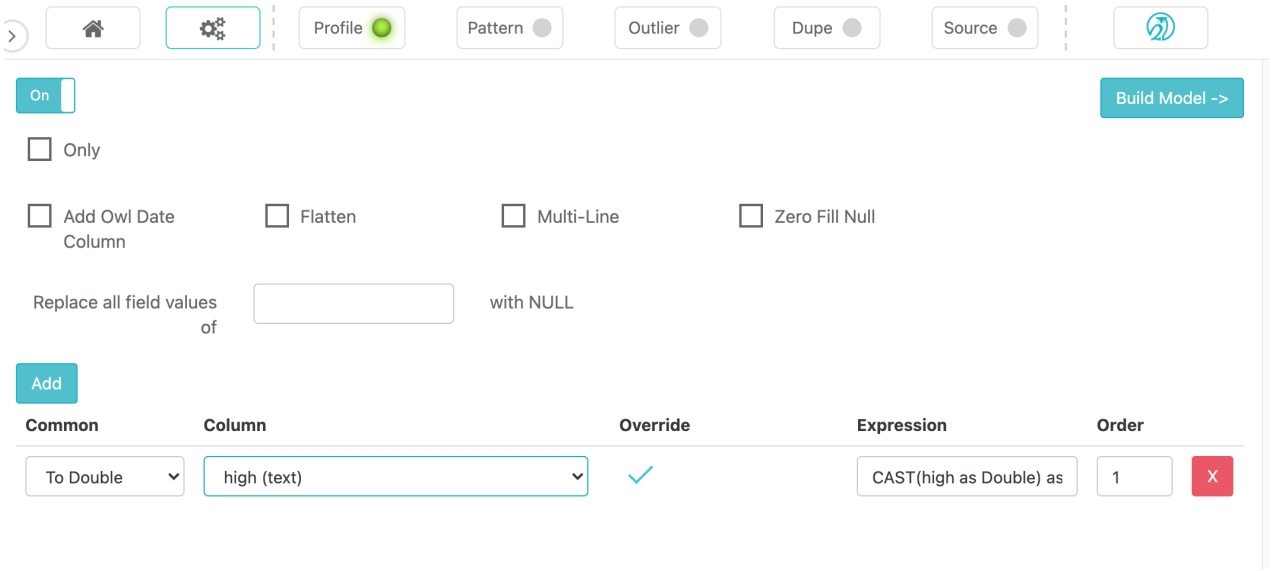
With inline processing you will already have a dataframe so you can skip down to setting the OwlContext.

```
val conn =`` ``connProps`` ``+ ("dbtable" -> s"($query)``
``$dataset")val df =`` ``spark.read.format("jdbc").options
(conn).load

val owl = OwlUtils.OwlContext(df, opt)owl.register(opt)owl.owlCheck
```

Transform

Transform Date

During Collibra DQ setup, you can transform columns such as Dates and Numbers to preferred formats. It is a common need to replace N.A. with nulls or empty white space.



Example

```
./owlcheck \
-ds  "dataset_transform" \
-rd  "2018-01-31" \
-f   "/Users/Documents/file.csv" \
-transform "purch_amt=cast(purch_amt as double)|return_amt=cast
(return_amt as double)"
```

> Note Submit an expression to transform a string to a particular type.
> In this example, transform the purch_amt column to a double.

```
-transform "purch_amt=cast(purch_amt as double)"
```

> Note Example of converting a string to a date.

```
-transform "RECEIVED_DATE=to_date(CAST(RECEIVED_DATE AS STRING),
'yyyyMMdd') as RECEIVED_DATE"
```

User Activity Report

What is a User Activity Report?

The User Activity report provides a view of the number of clicks in the Collibra DQ application by user. You can filter this report by various criteria from a Filter Type dropdown, as well as set a limit to how many users you want to view, based on the filter you choose. You can also configure the kind of chart you want to use to view the data.

Steps

1. Login to the Collibra DQ instance and click the Reports icon in the left navigation pane.
   The Reports page opens.
2. From the Reports page, select the **User Activity** tile.
3. To display results by filter type, select one of the following options from the **Filter Type** dropdown:
   - **Last x Days** - filter by last number of days.
     i. Select one of the following options from the **Last x Days** dropdown:
     - **1 day**
     - **2 days**
     - **7 days**
     - **10 days**
     - **15 days**
     - **30 days**

     ii. To limit the number of top end results that display in the chart and data table, enter a number in the **Limit** field.

> Note The chart and table update, based on your selections.

iii. Proceed to step 4.

- **Custom Range** - filter by configuring a custom range.
  i. Select a from and to date range in the Date Range fields.

  > Tip You can also select the icon, which displays a calendar, in which you can select a specific date.

  ii. To limit the number of users that the chart displays, enter a number in the **Limit** field.

  > Note The chart and table update, based on your selections.

  iii. Proceed to step 4.
- **All** - filter by username and user clicks.
  i. To limit the number of users that the chart displays, enter a number in the **Limit** field.

  > Note The chart and table update, based on your selections.

  ii. Proceed to step 4.

4. Click **Go**.
5. To configure the type of chart configuration and data you want to see in the chart, click the ellipsis icon.
6. Click **Chart Configuration**.
   The Chart Configuration window displays.

> Note As a best practice, keep the Chart Configuration as the default.

7. For the x-axis column, select one of the following options from the **X-Axis Column** dropdown:
    - **username**
    - **userClicks**
8. For the y-axis columns, select one or more of the following options from the **Y-Axis Columns** dropdown:
    - **userClicks**
    - **username**
9. For the type of chart you want to display, select one of the following options from the **Chart Type** dropdown:
    - **Line**
    - **Area**
    - **Column**
    - **Scatter**
10. Click **Save**.
    The chart displays based on the criteria you selected. The table below the chart displays the specific usernames and user clicks, based on your selections.

    > Note To close the Chart Configuration window, click **Cancel** or the **X** in the upper-right corner of the window. You can hide or show the filter options by clicking the ^ or v icons, located to the right of the filter options.

    > Tip To narrow your results in the data table, you can enter information in the Search field.

11. Navigate the pages of your report by clicking the and icons, located bottom-right of the columns.

Sample SQL query

You can use the following sample SQL query for a User Activity report.

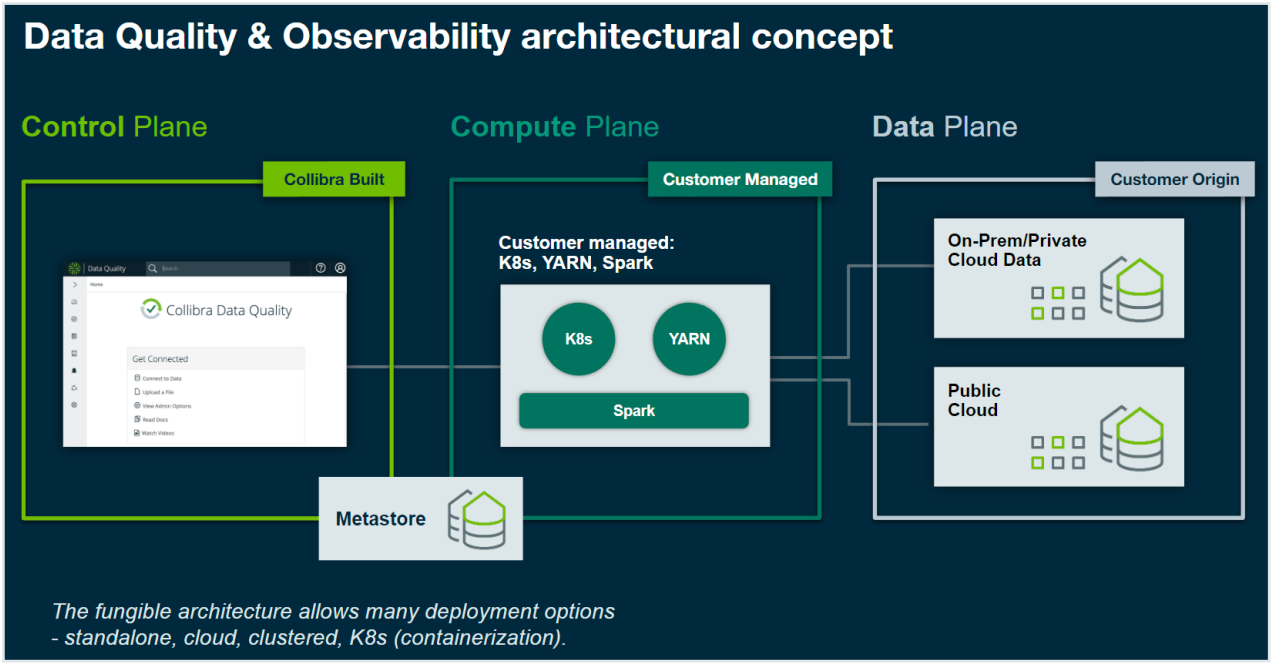
```
select username, count(*) user_clicks from activity_audit
where audit_ts > '2022-03-05'
and username <> 'admin'
group by username
order by user_clicks desc limit 22
```

Chapter 18

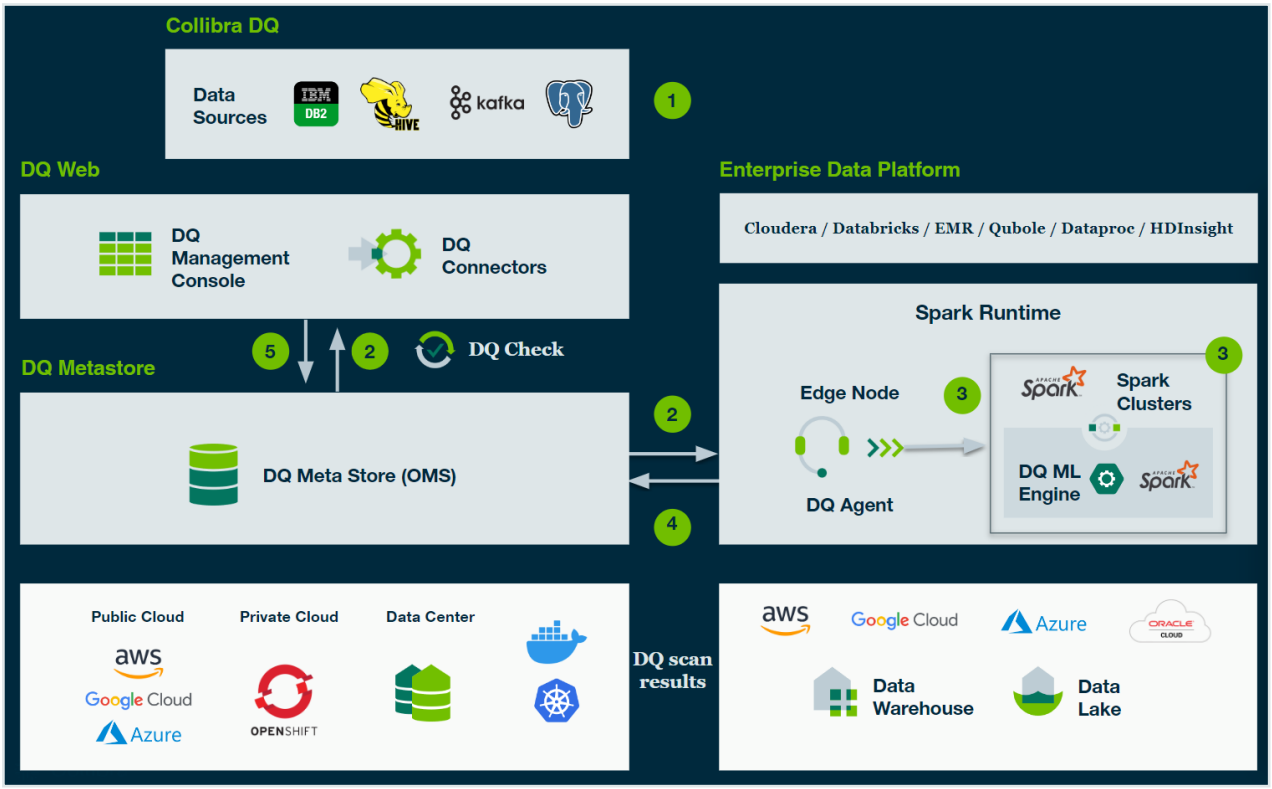
Collibra DQ Architecture

Architecture Diagram

Collibra DQ Architecture

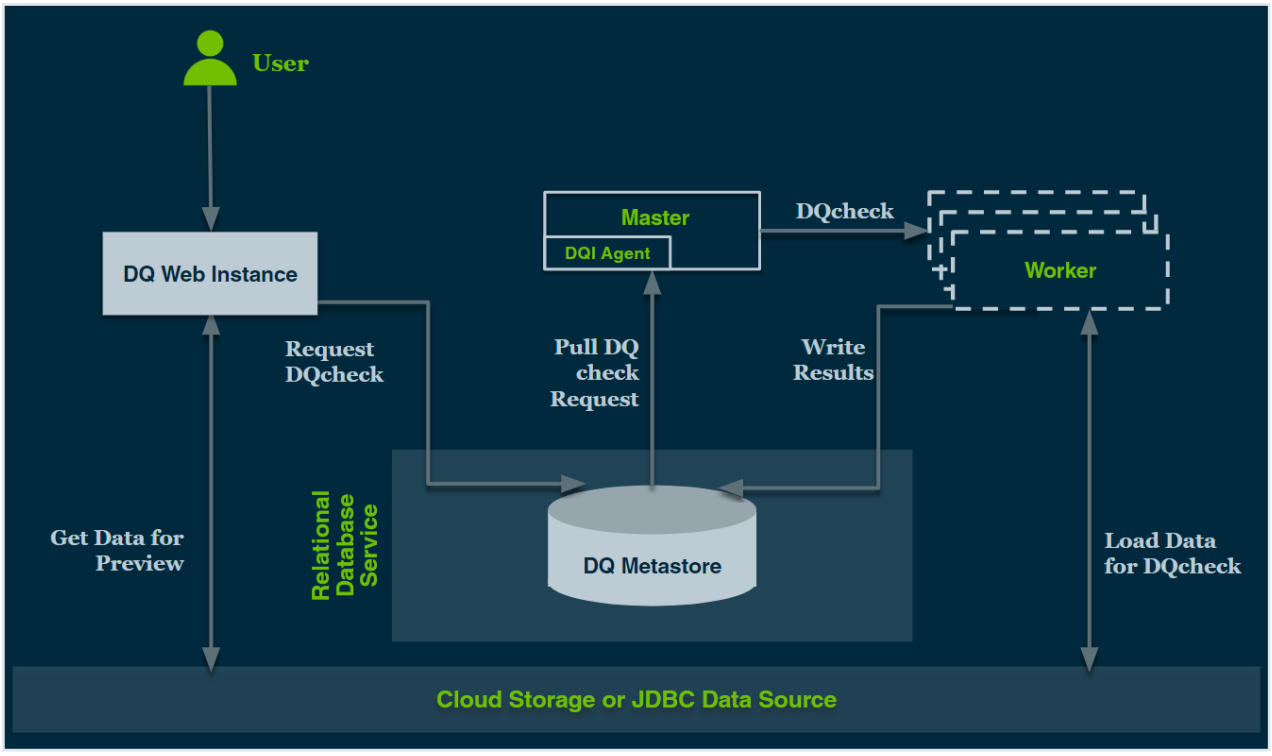


High-Level Diagram

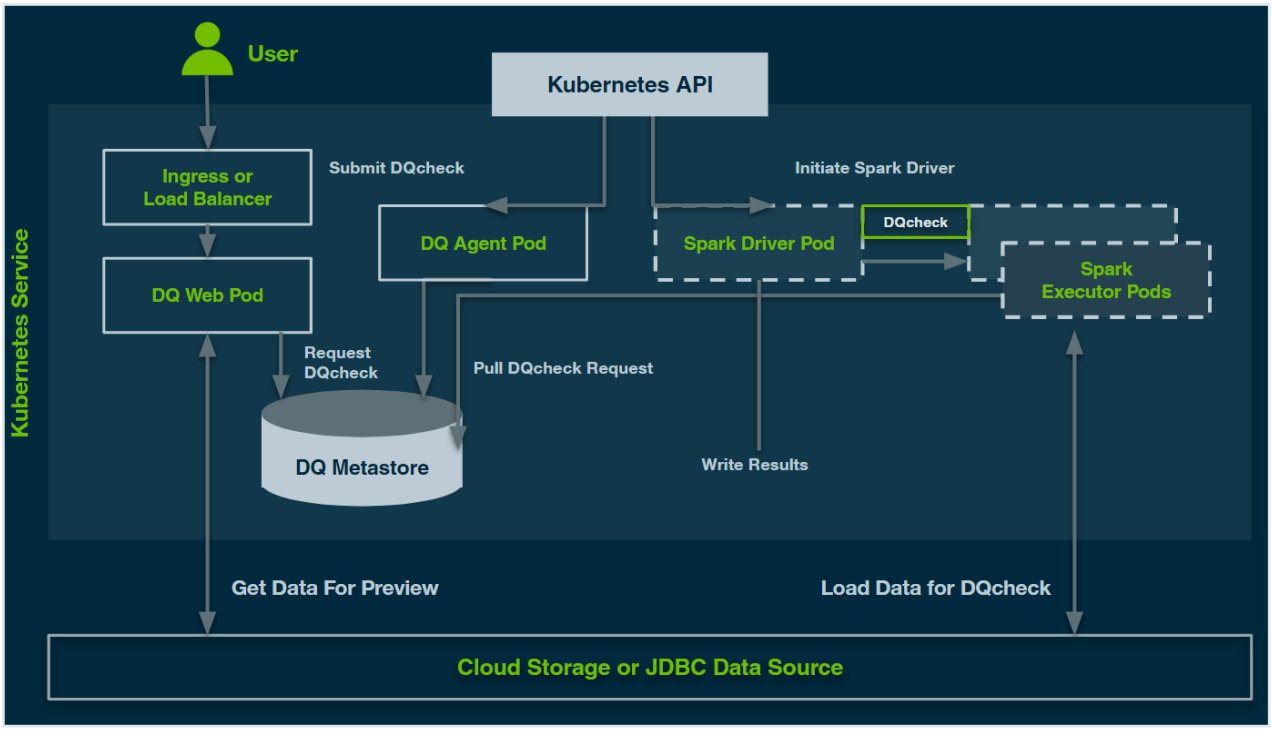


1. Connect to data sources.
2. Build the DQ scan algorithm and submit the job.
3. Execute the Spark job.
4. Write the DQ results in the Metastore.
5. Browse the results of the DQ Scan with the management console.

Collibra DQ Hadoop Deployment Diagram

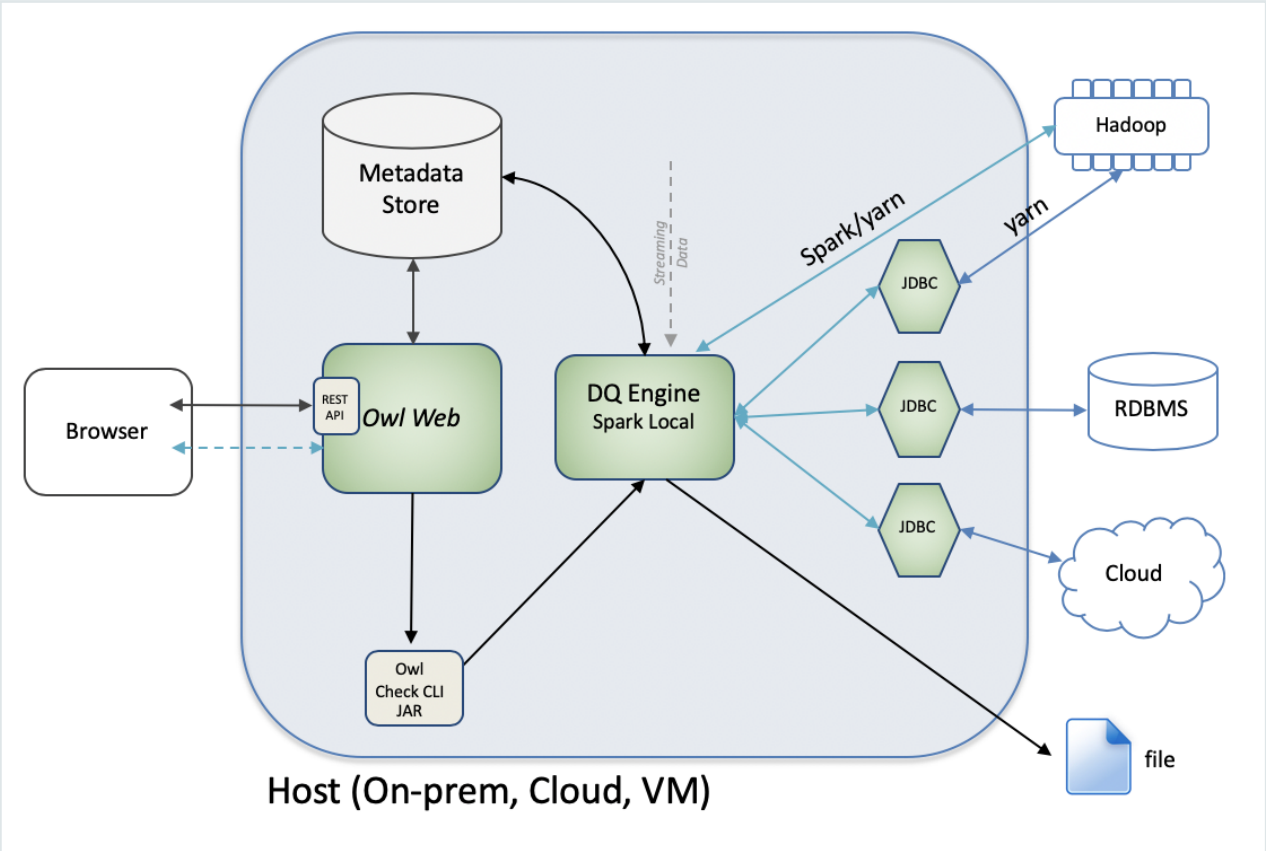


Collibra DQ Kubernetes Deployment Diagram



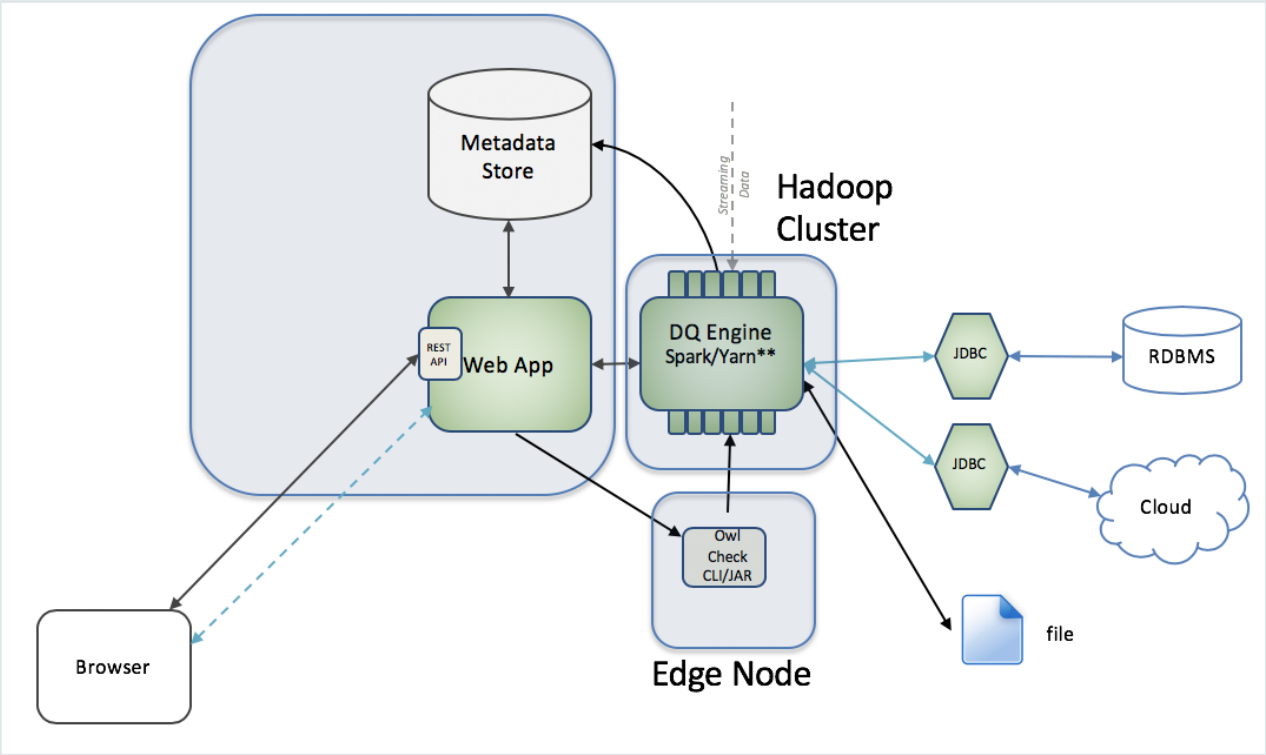
Note For Kubernetes deployments of Collibra DQ should use Auto Scaling and Spot instances to further increase efficiency and reduce cost.

Collibra DQ Standalone



The image above depicts owl-web, owl-core, Postgres and orient all deployed on the same server. This can be an edge node of a Hadoop cluster or a server that has access to run Spark-submit jobs to the Hadoop cluster. This server could also have JDBC access to other DB engines interested in being quality scanned by Collibra Data Quality & Observability. Looking at this depiction from left to right the client uses their browser to connect to Collibra DQ's Web Application running on the default port 9000. The Collibra DQ Web Application communicates with the metastore.. The Web Application can run a local DQ check, or the Data script can be launched from the CLI natively. The DQ check launches a job using Collibra DQ's built in Spark Local DQ Engine. Depending on the options supplied to the DQ check command, the Collibra DQ can scan a file or database with JDBC connectivity.

Collibra DQ Distributed



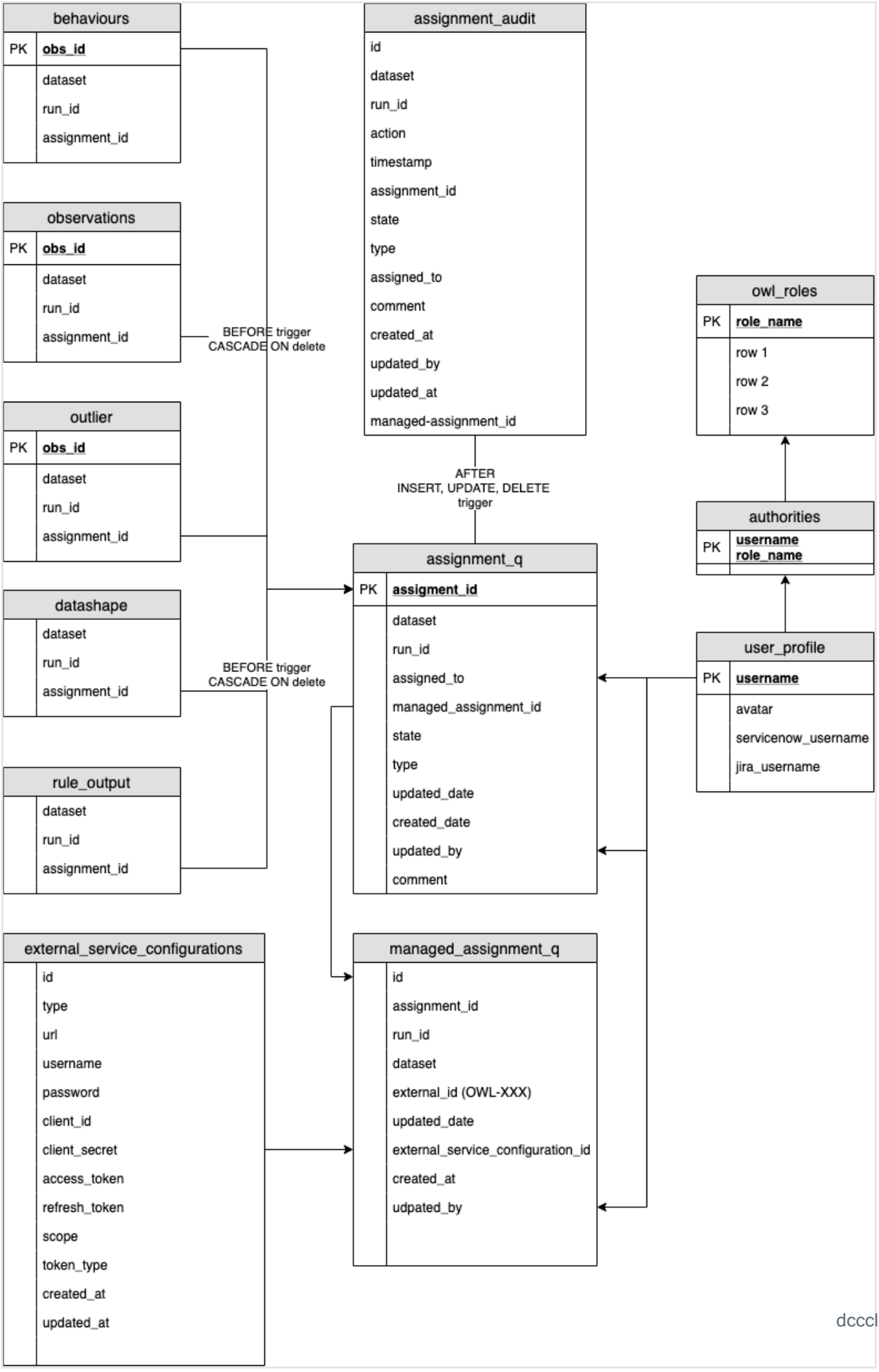
The image above depicts owl-web and owl-core deployed on different servers. In this example Owl-web is NOT deployed on the edge node. Owl-core is installed on the edge node and writes DQ check results back to the metastore that the DQ Web App points to.

> Note In this scenario, the metastore and the web-app run on the same host.

The other change is that the DQ check distributes the work on top of a Hadoop cluster to leverage Spark and use the parallel processing that comes with the Hadoop engine.

ERD

Please note there are over 110 tables in the underlying Postgres database and many are for application settings.


behaviours
PK obs_id
dataset
run_id
assignment_id
observations
PK obs_id
dataset
run_id
assignment_id
outlier
PK obs_id
dataset
run_id
assignment_id
datashape
dataset
run_id
assignment_id
rule_output
dataset
run_id
assignment_id
BEFORE trigger
CASCADE ON delete
BEFORE trigger
CASCADE ON delete
assignment_audit
id
dataset
run_id
action
timestamp
assignment_id
state
type
assigned_to
comment
created_at
updated_by
updated_at
managed-assignment_id
AFTER
INSERT, UPDATE, DELETE
trigger
assignment_q
PK assigment_id
dataset
run_id
assigned_to
managed_assignment_id
state
type
updated_date
created_date
updated_by
comment
owl_roles
PK role_name
row 1
row 2
row 3
authorities
PK username
role_name
user_profile
PK username
avatar
servicenow_username
jira_username
external_service_configurations
id
type
url
username
password
client_id
client_secret
access_token
refresh_token
scope
token_type
created_at
updated_at
managed_assignment_q
id
assignment_id
run_id
dataset
external_id (OWL-XXX)
updated_date
external_service_configuration_id
created_at
udpated_by

System Requirements

The following pages include the system requirements for installing Collibra Data Quality.

Supported Operating Systems

> Note Only 64-bit Linux operating systems are supported.

Standalone operating systems

- Red Hat Enterprise Linux 7.x
- Red Hat Enterprise Linux 8.x
- CentOS 7.x
- CentOS 8.x

Container operating system

- Red Hat Universal Base Image 8 Micro (ubi8-micro)

Hardware Sizing

Hardware Sizing (Standalone Install)

Small Tier - 16 Core, 128G RAM (r5.4xlarge / E16s v3)

Component	RAM	Cores
Web	2g	2
Postgres	2g	2
Spark	100g	10
Overhead	10g	2

Medium Tier - 32 Core, 256G RAM (r5.8xlarge / E32s v3)

Component	RAM	Cores
Web	2g	2
Postgres	2g	2
Spark	250g	26
Overhead	10g	2

Large Tier - 64 Core, 512G RAM (r5.16xlarge / E64s v3)

Component	RAM	Cores
Web	4g	3
Postgres	4g	3
Spark	486g	54
Overhead	18g	4

Estimates

Sizing should allow headroom and based on peak concurrency and peak volume requirements. If concurrency is not a requirement, you just need to size for peak volume (largest tables). Best practice to efficiently scan is to scope the job by selecting critical columns. See Performance Tuning for more information.

Bytes per Cell	Rows	Columns	Gigabytes	Gigabytes for Spark (3x)
16	1,000,000.00	25	0.4	1.2
16	10,000,000.00	25	4	12
16	100,000,000.00	25	40	120

Bytes per Cell	Rows	Columns	Gigabytes	Gigabytes for Spark (3x)
16	1,000,000.00	50	0.8	2.4
16	10,000,000.00	50	8	24
16	100,000,000.00	50	80	240
16	1,000,000.00	100	1.6	4.8
16	10,000,000.00	100	16	48
16	1,000,000,000.00	100	1600	4800
16	100,000,000.00	100	160	480
16	1,000,000.00	200	3.2	9.6
16	10,000,000.00	200	32	96
16	100,000,000.00	200	320	960
16	1,000,000,000.00	200	3200	9600

Cluster

If your program requires more horsepower or (Spark) workers than the example tiers above which is fairly common in Fortune 500 companies than you should consider the horizontal and ephemeral scale of a cluster. Common examples include Amazon EMR and Cloudera CDP. Collibra DQis built to scale up horizontally and can scale to hundreds of nodes.

Minimum System Requirements

Hardware based on role

Standalone, distributed, or fully-distributed.

- Please see Hardware Sizing for minimum resources for each component.

Java version

- Oracle JDK version 11.0.x
- Oracle JDK version 1.8.0_152.x
- Open JDK version 1.8.0.x

Encryption

- Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction

Postgres Version

- Collibra Data Quality comes prepackaged with version 11.4 of Postgres.
- Version 9.6.5 and above is supported if wanting to use an external metastore.

User Privileges

- Installation is completed via tarball.
- You must be able to create directories, launch scripts, and start processes (Java processes).
- SUDO is not required.
- ULIMIT settings of 4096 or higher
    - All owl services typically consume about 428 threads.
    - During each DQ Job, about 400 additional threads are consumed.
    - Thus 4096 threads can allow for about 9 concurrent DQ Jobs (on a standalone install). If more are needed, plan accordingly.

Planning the installation

It is always best to consult the Collibra DQ team for more information about what options make the most sense for your environment.

1. Determine how you plan to install Collibra DQ (standalone or distributed), as shown in the Architecture Diagram.
2. Plan your infrastructure, scale, and HA.

3. Validate your system prerequisites.
4. Obtain the packages from Collibra DQ.

Installation Packages/Files (BOM)

- demoscripts.tar.gz
- log4j*
- owlcheck
- owl-core-2.1.0-jar-with-dependencies.jar
- owl-webapp-2.1.0.jar
- owl-agent-2.1.0.jar
- setup.sh
- owl-postgres.tar.gz
- notebooks.tar.gz
- owlmanage.sh

Default Ports used by Collibra DQ

- 5432 – Postgres
- 9000 – Owl-web

Supported Web Browsers

Browser	Version
Google Chrome (recommended)	70.0.3538.102 or newer
Mozilla Firefox	52.8.0 or newer
Safari	12.0.1 or newer

Build Versions

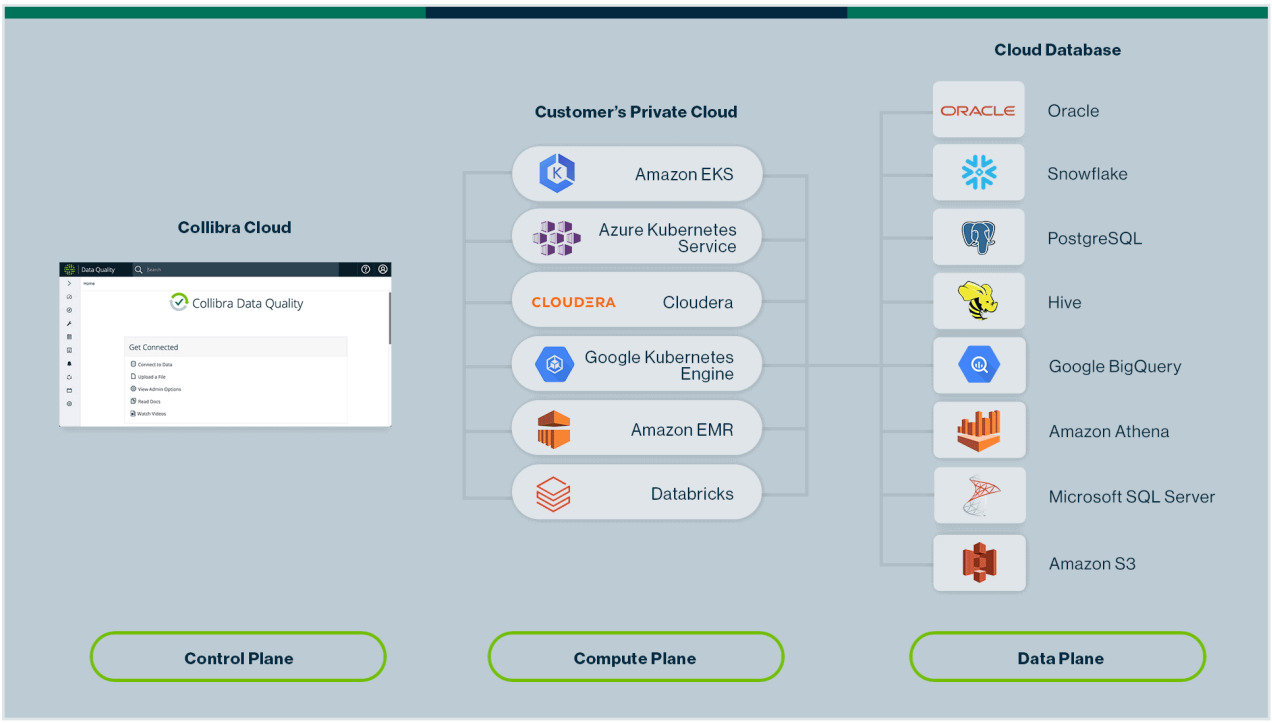
Generally Available Build Versions

- Default Build = Spark 2.3.0
- Spark 2.4.5
- Spark 3.0.1
- HDP 3
- CDH5
- CDH6-NOLOG
- K8s

Diagrams

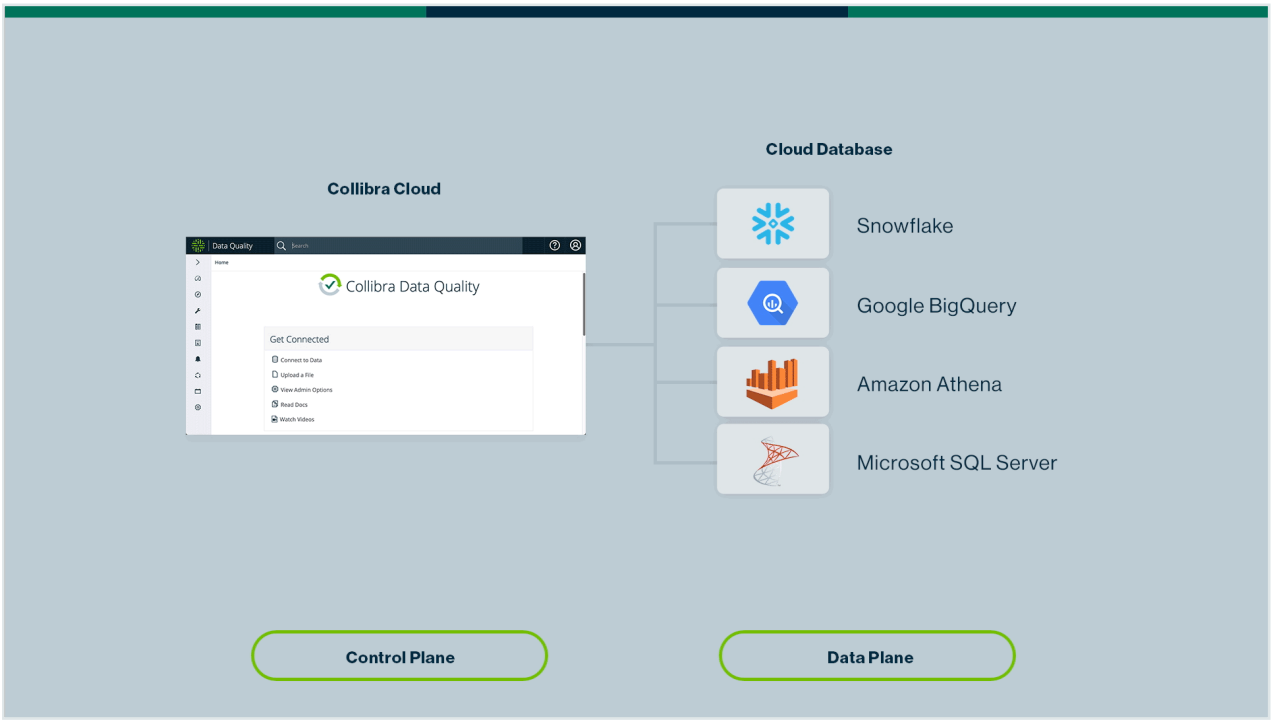
DQ Control to Compute to Data

As of March 2022, DQ Cloud is in private beta for select customers. General availability is expected in Q4 2022. Available compute plane options will start with Collibra Edge submitting jobs to Rancher K3s and Hadoop platforms (Cloudera, Dataproc, and EMR). Additional compute plane options will be added over time.



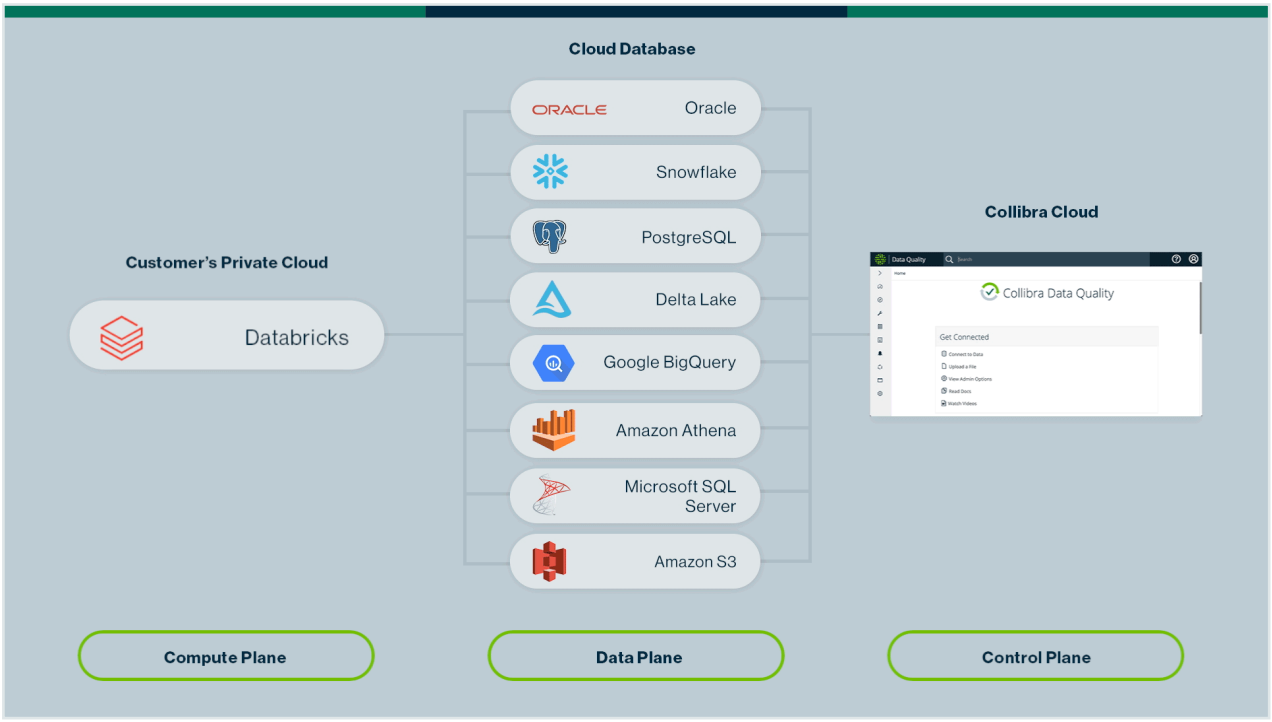
Pushdown DQ Control to Data

As of Q3 2022, Steps is in private beta for select customers. Some features may be limited.



Databricks to Data to DQ Control

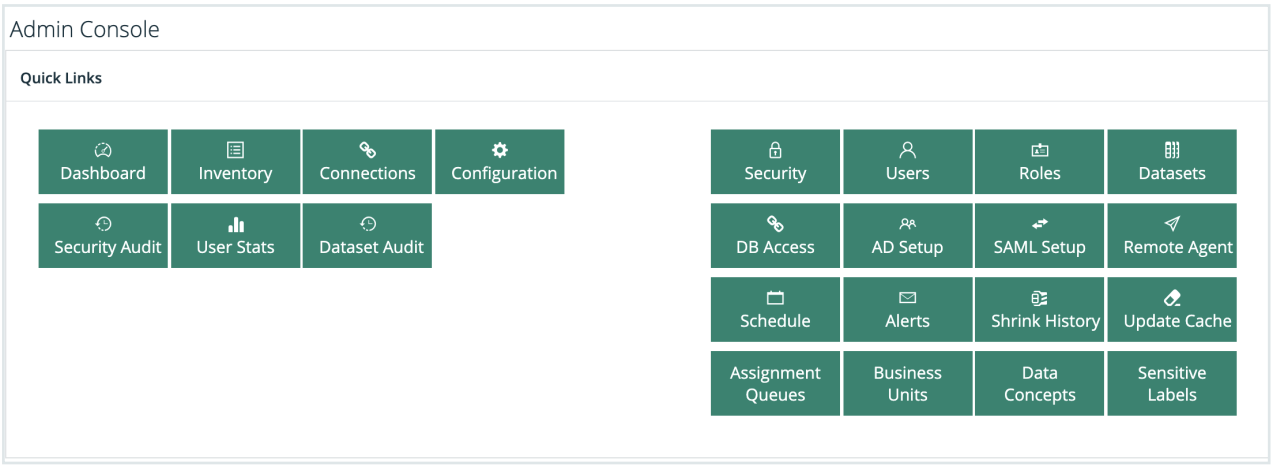
Running DQ jobs from Scala and Pyspark notebooks is generally available.



Chapter 19

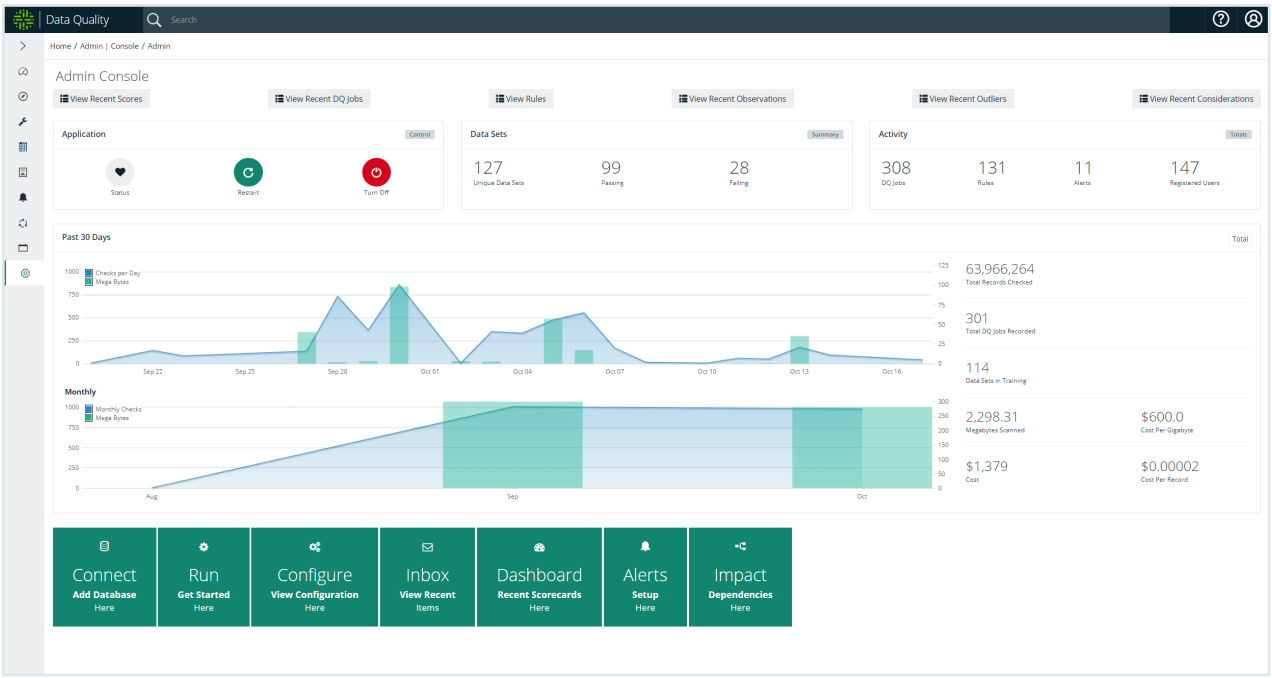
Collibra DQ Admin

Overview



Dashboard, Inventory, Connections, and Configuration are shortcut links to different operations within Collibra Data Quality & Observability.

- Dashboard
  - Shows an overall number of DQ Checks scanned the MB's, total jobs, total records scanned.
  - Total number of data sets, including passing and failed jobs (quality fails), number of alerts, and number of rules.
  - A list of messages on specific DQ Checks and what was found.



- Inventory
  - All DQ Jobs ever executed including Run Date, Data Set, Command Line, Type, Query, and Connection.
- Connections
  - Connect to a data source via the Admin Console.
- Configuration
  - Apply administrative override limits. For example, you can limit the amount of DataShapes recorded on a specific data set with the `datashapelimit` field.
  - If a duplicate record is found have a 1 point (negative) score per record. Allows administrators to increase or decrease the impact of different DQ issues found.
  - Display limits in order to improve performance of the UI.
- Audit Trail
  - List all security related changes by user, action, description and timestamp in a searchable, sortable table.

Configuration

Multi-Tenancy

Divide and conquer your data quality.

MultiTenancy allows a company the ability to instantiate different organizations within one entity. For example say your organization is called Acme and inside of Acme there are two divisions AcmeTraders and AcmeInsurance and each organization is not allowed to see one another's owlcheck results. You would simple segregate them into 2 different Tenants within the overarching Collibra DQ web application.

Prerequisite(s)

- DNS entry for owl web server IP = example below we call it hub (existing for single tenant setup)
- DNS entry for multiTenantSchemaHub = example below we call it owlhub.hub

Prerequisites For URL Based tenancy: tenant.host

A DNS entry for each tenant

- DNS entry for every tenant you want to create = example below we call it tenant1.hub

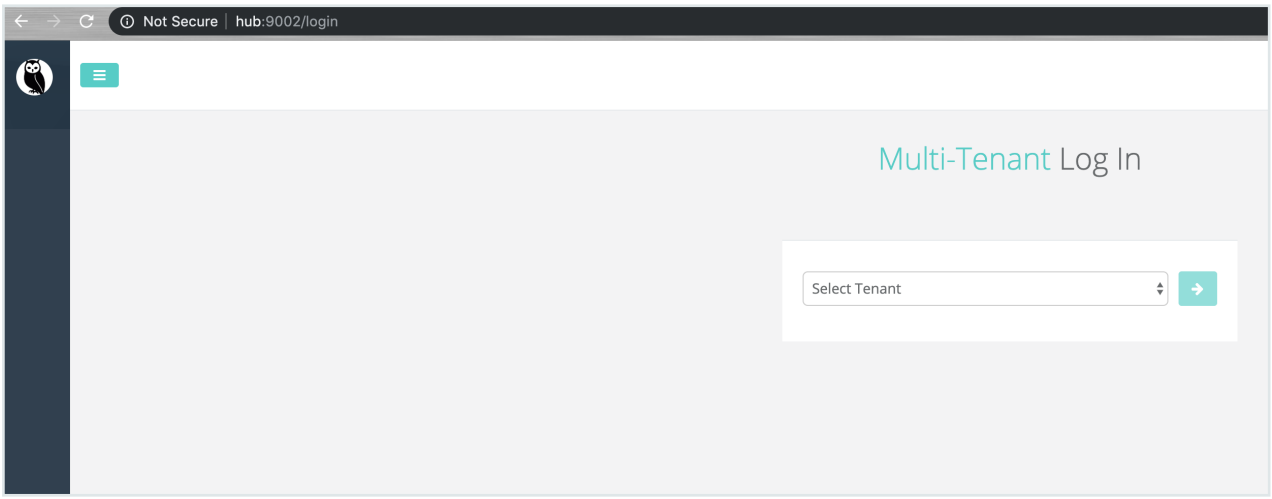
Each record above points to the same IP address.

Setup

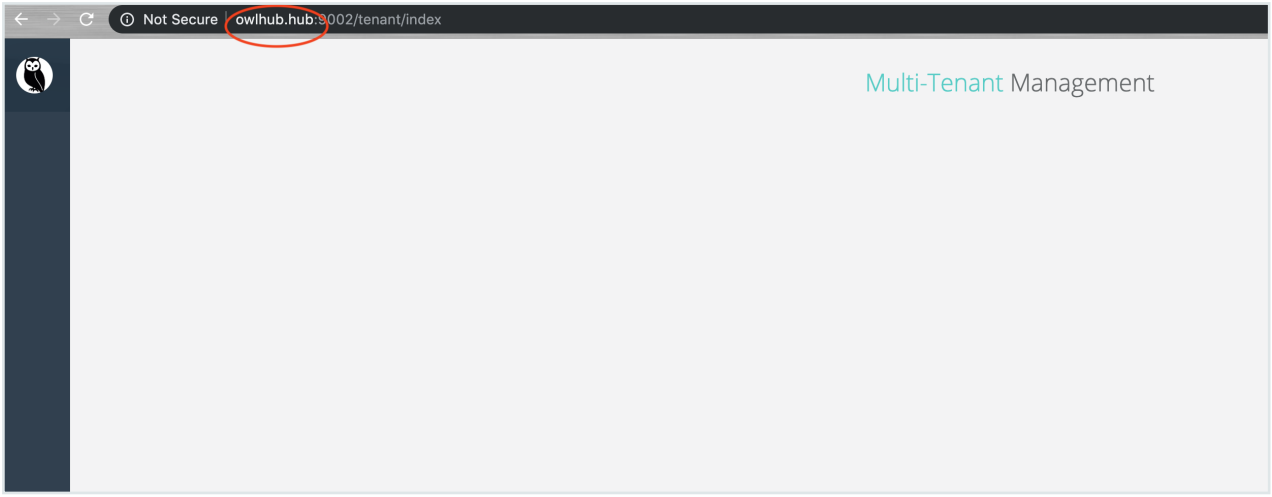
In order to setup multi-tenancy follow these steps

- If this is an upgrade please make sure to follow the steps outlined in the "Upgrading to latest Version"
- Make sure the web application has started up one time and you successfully logged into it with the default credentials.
- Then stop all the components using ./owlmanage.sh stop
- Modify the owl-env.sh file to include these to new parameters
    a. **export multiTenantSchemaHub=owlhub** (this is a new schema that will get created on owlweb start, note the name of the TenantSchemaHub can be changed to the desired name at setup time)
    b. **export MULTITENANTMODE=TRUE** (this enabled multi-tenancy to be used).
    c. **export URLBASEDMULTITENANTMODE=TRUE/FALSE**
        i. TRUE (default) means you are using the tenant name as a sub-domain (see prerequisites)
        ii. FALSE means you will let owl manage tenants via sessions/tokens
- If using agents as part of the operation of owl please be sure to modify the owl.-properties file to include the following.

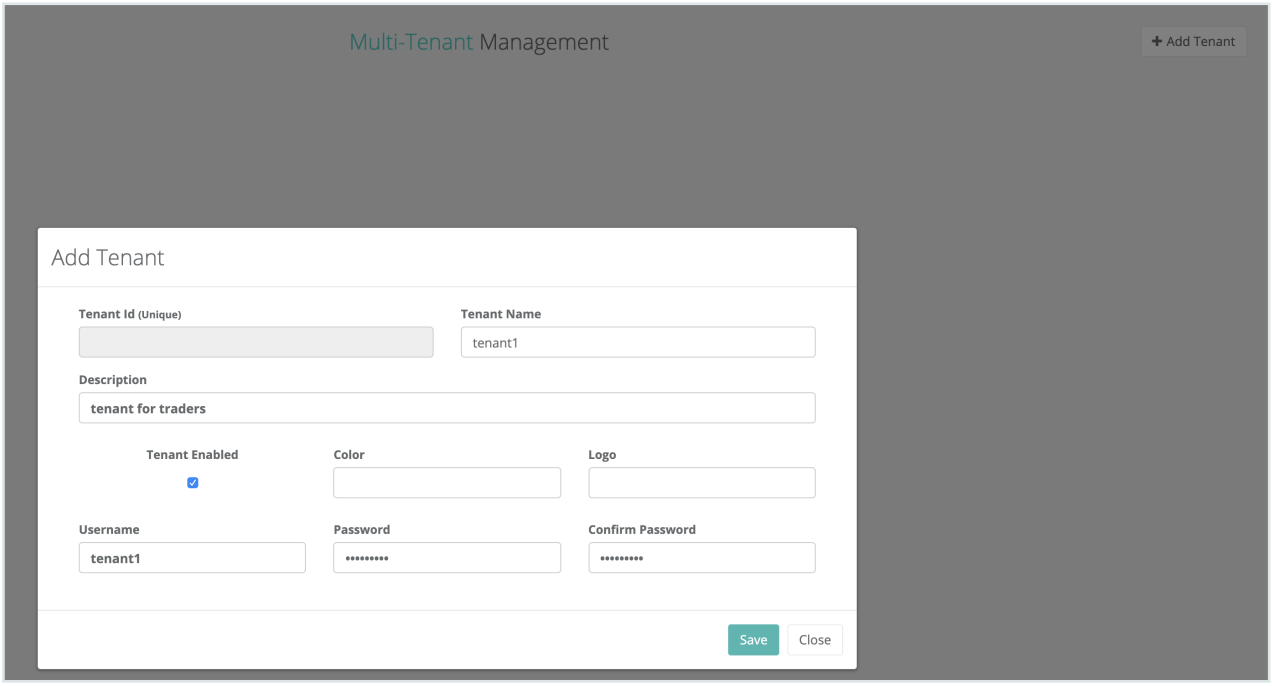
    a. spring.agent.datasource.url=jdbc:postgresql://cdh-edge-dan.us-east4-c.c.owl-hadoop-cdh.internal:5432/postgres**?currentSchema=owlhub** (matching the name of the schema set on step 3-1 above).
    b. jdbc:postgresql://cdh-edge-dan.us-east4-c.c.owl-hadoop-cdh.internal:5432/postgres**?currentSchema=owlhub** (matching the name of the schema set on step 3-1 above).
- Once the settings have been configured for multi-tenancy please start up the owlweb host first using ./owlmanage.sh start=owlweb. Once the web is up and you can hit the page please start up the agents using ./owlmanage.sh start=owlagent.
- In order to use multi-tenancy in URLBASEDMULTITENANTMODE=TRUE you'll have to make sure we have DNS entries to the tenant endpoints, otherwise click the tenant management link from the login page. Example:
    a. If I have a DNS alias named hub. I should be able to point me browser at hub:9002 (or your respective owlweb port) to get to the main Multi-Tenant login page as depicted below



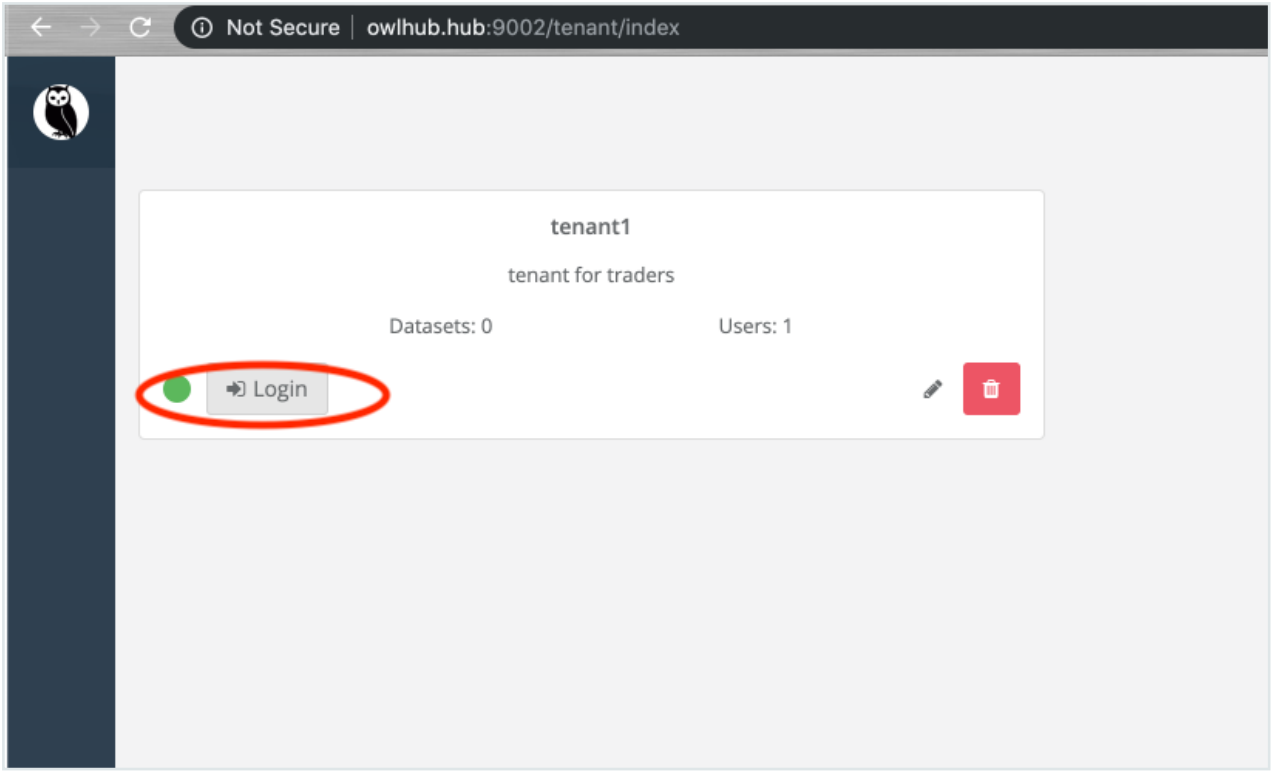
- This is where DNS alias come into place. Assuming we left the owlhub as the multiTenantSchemaHub name we hit the drop down and select owlhub and click the arrow it will place owlhub.hub into the url. This means there also has to be a DNS Alias name for your selected multiTenantSchemaHub name. NOTE: Username and password for tenant management is mtadmin / mtadmin123



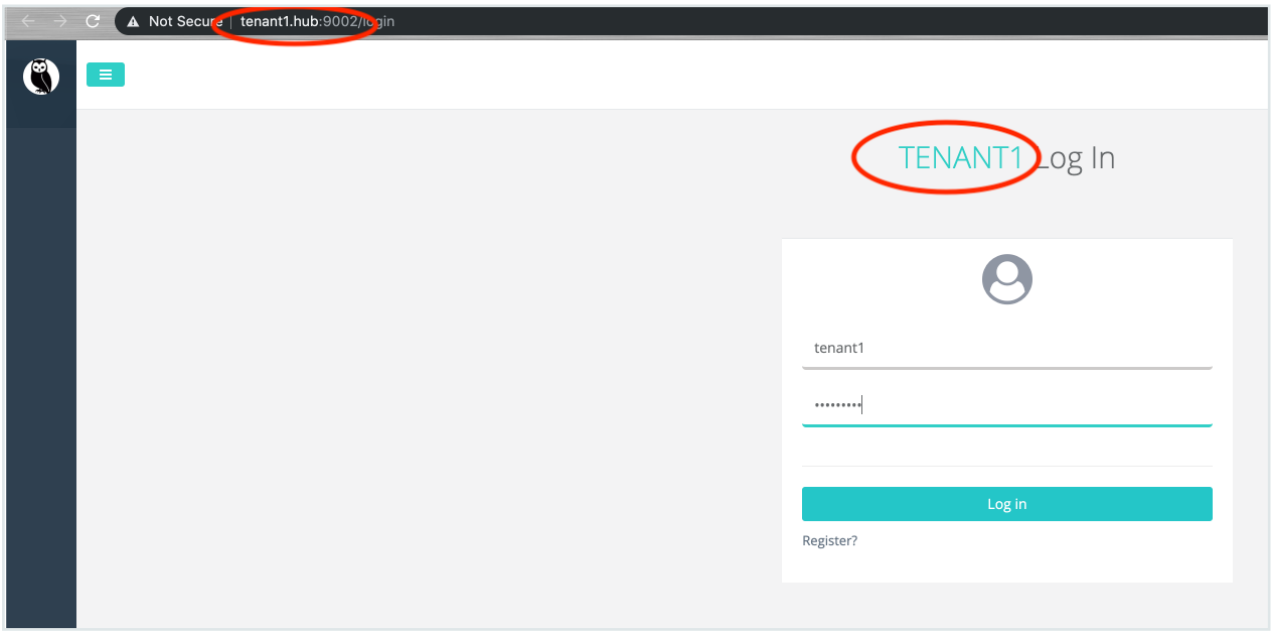
- Now that you logged into the Tenant Management screen using the hub DNS alias we can create our first tenant. In this example below I'm going to create a tenant named tenant1. First click the "+ Add Tenant" button in the top right part of the screen.



Click Save. Your tenant shows up in the list and now you can click the login button as shown below.

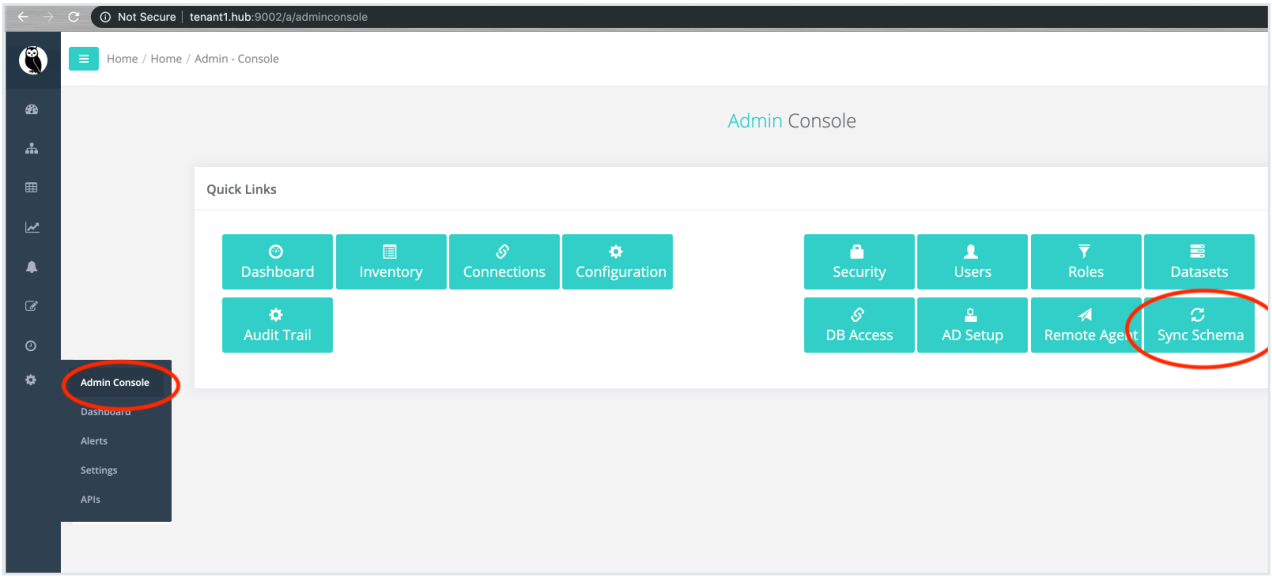


Clicking the Login button will redirect your browser to the tenant1.hub:9002 url (DNS entry needs to be in place for tenant1 as shown below).



Enter the admin username and password that you created for the tenant1 (refer to figure 3 about) and login to the tenant as the admin.

While logged in as a tenant admin the last step is to go to the Admin Console and click on "Sync Schema" this will generate the tables under the tenant called tenant1.



At this point you are ready to start administrating your tenant1 as you did with the owl web application in the past.

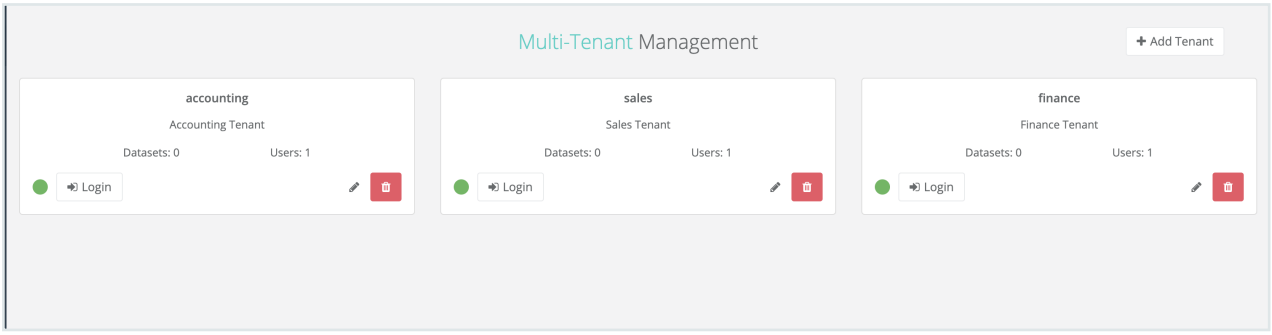
Supplemental: Adding and Editing a Tenant

In many cases it may make sense to have isolated environments to check data quality. This need could be driven by a number of factors including data access rights, organization and business models, reporting needs, and/or other security requirements.

Regardless of the need,Collibra DQ will support dynamically creating tenants via our Collibra DQ Hub Management portal as part of the Collibra DQ Web Application. That's it, there is nothing else to install, simply enable Multi-Tenant mode in the application configuration properties and you are on your way.

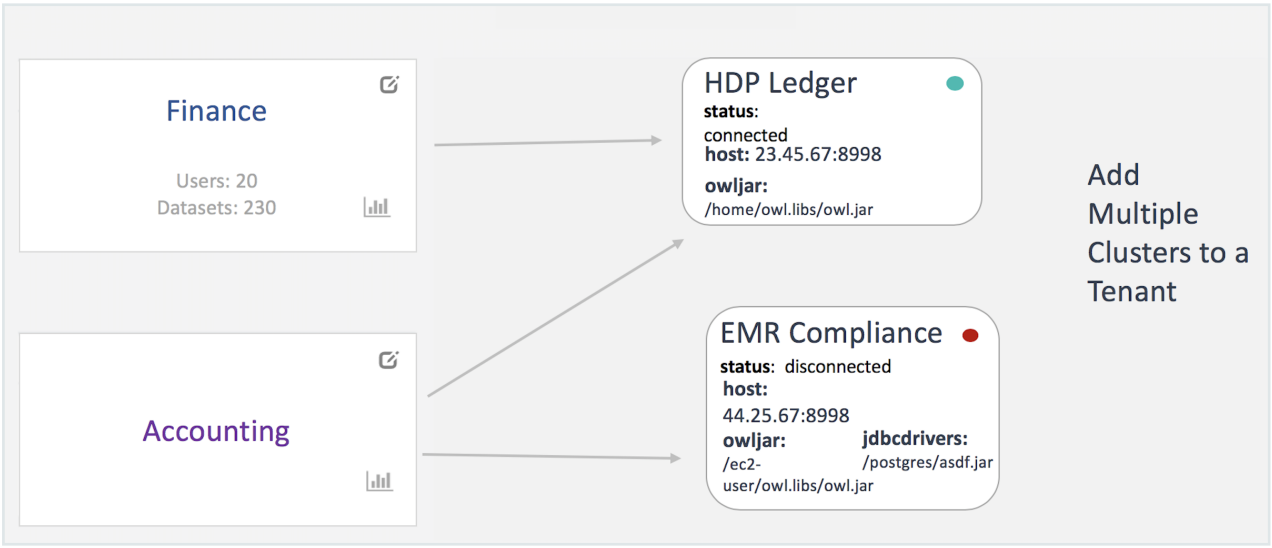
Once enabled you will have a tenant selection screen prior to login where you can chose any of your configured tenants or access the Owl Hub (with the TENANT_ADMIN role)

After selecting the owlHub tenant, you will have the ability to manage each tenant, as well as create new tenants from the management console.

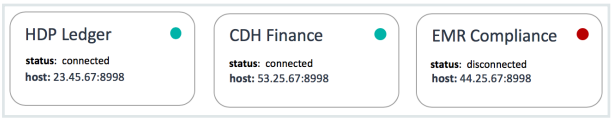


All enabled tenants will be listed in the multi-tenant drop down menu. Access to tenants are handled by the administrator(s) within each tenant individually.

Access to agents are also handled by the administrator(s) within each tenant individually.



Each agent is visible and editable as an Admin from the UI.



Time-Based Data Retention

Setting up Retention Based Data Purge

Retention based purge of data can be turned on to allow data to automatically be cleaned based on an organization's data retention policy.

Benefit

Once enabled, what type of data is removed?

- data_preview *(Drill-in records for rules, outliers, shapes, etc.)*
- dataset_field *(profiling stats)*
- rule_breaks *(Rule Exception records)*
- dataset_scan *(Job Ledger)*

Setup

In order to set up retention based data purge, three (3) environment variables need to be set up in the owl-env.sh configuration script. Note: a restart of the webapp is required for this configuration to take place.

- **cleaner_retention_enabled**
  - TRUE or FALSE on whether this feature is enabled
- **cleaner_retention_days**
  - Number of days to retain data
- **cleaner_retention_field**
  - Controls which field to use to select eligible data set runs
  - Potential values
    - updt_ts: consider the last time a data set run was updated
    - run_id: consider the run id field of the data set

Configuration

Example configuration in owl-env.sh

Organization wants to purge data where the updt_ts is more than 1 year old

In owl-env.sh, add the following lines

```
export cleaner_retention_enabled=TRUE
export cleaner_retention_days=365
export cleaner_retention_field="updt_ts"
```

Config Map

```
autoClean: "false"
cleaner_retention_days: "180"
cleaner_retention_field: updt_ts
cleaner_retention_enabled: "true"
```

Defaults for Auto Clean Process

> Note This is a separate rolling purge that is distinct **time-based retention**. This is **on by default** and uses the predefined limits below. You will see audit records for this clean-up process in Audit History of the Admin Console.

Separate from the time-based retention there is also a default auto clean mechanism that actively purges your old records. This is enabled by default and can be modified by use of the autoClean (AUTOCLEAN) boolean parameter.

```
AUTOCLEAN=false or autoClean="false"

### Depending whether this is part of owl-env.sh
### or the configMap of the web pod
```

These are the defaults. The row count threshold is the global limit when this is triggered. This is based on the records in the data_preview table. The runs threshold and the dataset per row threshold are data set-level limits that require a data set to have at least 4 scans and at least 1000 rows.

This is an example using the owl-env.sh file to control these settings.

```
export AUTOCLEAN=true
export DATASETS_PER_ROW=1000
export RUNS_THRESHOLD=4
export ROW_COUNT_THRESHOLD=200000
```

For example (using the settings above):

When data_preivew table has 200k rows
Look for data sets with 1000+ rows in data_prevew table
And have at least 4 scans
Then delete the oldest scan for those data sets

Auto clean and time-based retention run on a routine thread that triggers while the web application is running. It looks for clean-up candidates every few minutes when AUTOCLEAN=true or cleaner_retention_enabled=TRUE.

Set up SMTP

Simple Mail Transport Protocol, or SMTP, is an internet standard for email transmission. Collibra DQ allows you to configure a single SMTP server to send alerts to the attention of the data set owner in case a specific condition is met, such as:

- Data Quality Score is below a specific threshold.
- Row Count is below a specific threshold.
- If a rule is triggered.

Steps

1. In the **Admin Console**, click **Alerts**.
2. In the **Configuration** section, enter the required information:

Option	Description
SMTP Host	The name or the IP address of the SMTP server.
SMTP Port	The port used by the SMTP server.

Option	Description
SMTP User-name	The username or the account that is configured on the SMTP server for use.
SMTP Pass-word	The password of the SMTP server username or account.
(Default) To Email	The sender email address.
Reply Email	The reply-to email address.
Include Links	Links to the specific instance(s)/page(s) where the alert came from in the DQ alert email.

3. Click **Save**.

> Note If you select the **Include Links** option, links to the specific instance(s)/page(s) where the alert came from display in the DQ alert email. This allows you to quickly access the environment(s) from which the alert originated.

> Important Be aware that some email servers, such as Microsoft Exchange, require the **SMTP username** and **Reply email** fields to be the same.

For more information about DQ alerts, go to Email Alerts.

Admin Limit Settings

You can configure Admin Limit settings in the Admin Console.

Steps

1. Login to the Collibra DQ instance and click the gear icon in the left navigation pane.
2. Click **Admin Console**.

3. Click **Configuration** on the Quick Links page.

> Note You should generally modify the following Admin Limits settings to prevent too much data from being stored in the database. For example, if you set the `datashapelimit` to `50` and the `datashapelimitui` to `25`, when you run a check, only 50 shape issues get stored in the database and only the first 25 records display in the UI on the Findings page.

4. Configure the following Admin Limits settings on the Limit Settings page.

> Note Settings with "score" in the setting name represent the default score for that setting's type of findings.

> Warning When you increase the UI display limits, the performance of the web application may decrease.

Setting	Description
obslimit	Limit of observation storage findings that get stored in the database.
obslimitui	Limit of observation storage findings that display in the UI.
obsscore	Default observation score, per record.
outlierlimit	Limit of outlier findings that get stored in the database.
outlierlimitui	Limit of outlier findings that display in the UI.
outlierscore	Default outlier score, per record.
categoricallimit	Limit of categorical outlier storage findings that get stored in the database.
categoricallimitui	Limit of categorical outlier findings that display in the UI.
categoricalscore	Default categorical outlier score, per record.

Setting	Description
datashapelimit	Limit of datashape storage findings that get stored in the database.
datashapelimitui	Limit of datashape findings that display in the UI.
datashapescore	Default datashape score, per record.
dupelimit	Limit of dupe storage findings that get stored in the database.
dupelimitui	Limit of dupe findings that display in the UI.
dupescore	Default dupe score, per record.
fpglimit	Limit of frequent pattern mining storage findings that get stored in the database.
fpglimitui	Limit of frequent pattern mining findings that display in the UI.
fpgscore	Default frequent pattern mining score, per record.
valsrclimit	Limit of validate source storage findings that get stored in the database.
valsrclimitui	Limit of validate source findings that display in the UI.
valsrcscore	Default validate source score, per record.
schemascore	Default schema score, per record.
histlimit	Limit of histogram storage findings that get stored in the database.
histlimitui	Limit of histogram findings that display in the UI.
lookbacklimit	Limit of lookback intervals findings that get stored in the database.

Setting	Description
maxcolumnlimit	Limit of maximum columns display findings that get stored in the database.
passfaillimit	Limit of passing or failing threshold findings that get stored in the database.
scorecardsize	Limit of scorecard size display findings that get stored in the database.
maxnumexecutors	Maximum number of executors to be set during estimation on a DQ scan, which gets stored in the database.
maxexecutorcores	Maximum number of executors cores to be set during estimation on a DQ scan, which gets stored in the database.
maxexecutormemory	Maximum number of executor memory (GB) to be set during estimation on a DQ scan, which gets stored in the database.
maxcellcountforparalleljdbc	Maximum number of cells for a DQ scan before it is recommended to use parallel JDBC, which gets stored in the database.

5. Click **Save**.
6. To add a custom configuration, click **+Add Custom** at the top-right of the window and enter your values in the Add a Custom Configuration window.

Job Management

Delete this text and replace it with your own content.

Concurrency

Concurrency in Collibra DQ is when multiple DQ Jobs run at the same time. Collibra DQ applies a default batch size limit of 5 Jobs that can run at once. These batch size limits ensure that the DQ Agent can properly allocate resources so it does not overload in high concurrency situations. As soon as the initial 5 Jobs begin to complete, any remaining Jobs can then enter the Jobs queue and begin to process.

Concurrency sizing

Before you run DQ Jobs concurrently, it is good to understand sizing specifications for and when you might need to allocate more resources to run your Jobs concurrently.

We generally advise that you allocate around 200 megabytes of Executor Memory per Job. For example, a batch of 10 concurrent Jobs should allocate 2 gigabytes of RAM to run correctly. The following table shows general guidance for resource allocation for concurrent DQ Jobs.

Batch Size	Agent Pod Memory
5	1 GB
10	2 GB
20	4 GB
40	8 GB
80	16 GB
160	32 GB
320	64 GB

> Note The default batch size is 5. If you run more than 5 Jobs concurrently, ensure your agent has sufficient memory. See the batch size configuration page for more details.

Overriding the default batch size

Unless you set the batch size in the owl-env.sh script for Standalone deployments or the Helm chart for Cloud native deployments, the batch size defaults to 5. This means that the DQ Agent applies a limit of 5 Jobs that can run concurrently. This page shows you how to override the default batch size for both Standalone and Cloud native deployments.

1. Open a terminal session.
2. Open the owl-env.sh file with the following command:

```
vi /home/owldq/owl/config/owl-env.sh
export DEFAULT_BATCH_ID=<your-b64enc-batch-id>
```

3. Use a base64 encoder, such as the free encoder from base64encode.org, to encode your batch ID and enter the encoded value after `DEFAULT_BATCH_ID`.

> Tip Your base64 encoded string should look similar to the following example:
>
> ```
> export DEFAULT_BATCH_
> ID=QUdFbVlYWnNkRDRpWVpiZXZGTnBlbVU5Tkw9PQ==
> ```

4. Enter the following command:

```
export DEFAULT_BATCH_SIZE=<your-batch-size>
```

> Tip Enter a value in `<your-batch-size>` that meets the concurrency sizing requirements of your use case.

5. End the terminal session.

1. Open a terminal session.
2. Enter the following command to pass your new default values to your agent batch ID and agent batch size:

```
--set global.configMap.data.agent_batch_
id="default?batchSize=<your-batch-size>"
--set global.configMap.data.agent.data="<your-batch-size>"
```

> Tip Enter a value in `<your-batch-size>` that meets the concurrency sizing requirements for your use case.

   a. If your Collibra DQ app is managed by an alternative Kubernetes package manager, ensure you update the **owl-agent-configmap** with the following variables:

```
AGENT_BATCH_ID: "your-base64-encoded-value"
AGENT_BATCH_SIZE: <your-new-batch-size>
```

> Tip Your base64 encoded string should look similar to the following example:
>
> ```
> export DEFAULT_BATCH_
> ID=QUdFbVlYWnNkRDRpWVpiZXZGTnBlbVU5Tkw9PQ==
> ```

> Tip Enter a value in `<your-batch-size>` that meets the concurrency sizing requirements for your use case.

   b. Restart the DQ agent pod to complete the updates.
3. Choose one of the following options to update your DQ agent resources:
   a. Update the DQ agent from the user interface
      i. Sign in to Collibra DQ.
      ii. Hover your cursor over the ⚙ icon and click **Admin Console**.
         » The Admin Console opens.

i. Click **Remote Agent**.
   » The **Agent Management** page opens.
ii. Click the pencil icon.
   » The **Edit Agent** modal opens.
iii. Update **Number of Executors**, **Executor Memory**, **Number of Cores**, and **Driver Memory** according to the concurrency sizing requirements for your use case.
iv. Click **Save**.

b. Update the DQ agent through Helm to set your new limits
   i. Open a terminal session.
   ii. In your Helm values.yaml file, update the DQ agent's `memory` and `cpu` quotas according to the concurrency sizing requirements for your use case:

```
agent:
   enabled: true
   replicaCount: 1
   nodeSelector:
     key: ""
     value: ""
   resources:
     requests:
       memory: "<your-new-memory-value>"
       cpu: "<your-new-cpu-value>"
     limits:
       memory: "<your-new-memory-value>"
       cpu: "<your-new-cpu-value>"
```

   iii. Perform the Helm upgrade on the existing DQ release.
   iv. End the terminal session.

Advanced

Allow/Disallow Using the catalog feature from Wizard: BULK_CATALOG_ON = **TRUE**/FALSE

Preset Wizard Config for showing views: DB_VIEWS_ON = **TRUE**/FALSE

Preset Wizard Config for showing stats: DB_STATS_ON = **TRUE**/FALSE

Set File Search Path From Wizard: UPLOAD_PATH = <Path on Web Server>

Set/Unset -p/-srcp masking in UI: MASK_P_FLAG = **TRUE**/FALSE

Allow/Disallow Zeppelin Notebooks: ZEPPELIN_ENABLED = **TRUE**/FALSE

Allow/Disallow Using the orient DB: ORIENT_ENABLED = **TRUE**/FALSE

Pendo

> Warning The Pendo integration is active by default.

> Note As of the 2022.06 release, all new customers receive a new license.

Pendo is an analytics application embedded in Collibra that helps us analyze, develop, and improve our product. No sensitive information is ever collected, we only leverage high-level usage statistics to improve our offerings. If no modifications are made to the default settings, Pendo will not block or impair the intended functionality of Collibra Data Quality in any way.

Pendo in a standalone environment

If you install a standalone environment, modify the <install-dir>/config/owl-env.sh file by adding your license name

```
export DQ_INTEGRATION_PENDO_ACCOUNTID=<your-license-name>
```

> Note For more information on Collibra's subprocessors, please review Collibra's Subprocessors page.

Usage

> Note You must have admin privileges to access usage metrics.

The Usage page lets you analyze your monthly usage statistics from the Admin Console. Key monthly metrics tracked on the Usage page include:

- Total number of users.
- Total number of DQ Jobs run.
- Total number of rules applied.
- Total number of data sets.
- Total number of columns.

datasets	jobs_passing	users	distinct_columns	jobs_submitted	jobs_unknown	license_key	rules_active
293	47	250	7799	2	3	[redacted]	445
6979	0	139	165519	0	0		6457
298	588	9	4650	20	115		0
1477	527	40	58491	86	12		
233	206	24	6029	170	18	[redacted]	317
1478	556	40	58500	88	12	[redacted]	1820

Viewing the Usage page

To view the Usage page

1. Click the gear icon and then click Admin Console.
   >> The Admin Console opens.
2. Click the Usage tile.
   >> The Usage page opens.
3. Analyze your usage statistics if they are available.

License Management

This section shows you how to change your license information provisioned by Collibra.

Prerequisites

You have:

- Admin permissions.
- A license previously provisioned by Collibra.

View and edit your license

From the Admin panel, select **License** to open the License page. The table below shows the options available on this page.

> Note The **License** option only displays when React MUI is active.

Option	Description	Can edit
Key	Displays your current license key. Starting 30 days before your license expires, a counter in a banner message on the Collibra DQ home page displays how many days remain before your license expires. You can update your license key in this field to avoid any interruptions in the functionality of your DQ Jobs. Click into the field and enter your text to edit your license key. > Note To obtain a new license key, reach out to your CSM for assistance.	✓ Yes
Name	Displays your current license name. Click into the field and enter your text to edit your license name.	✓ Yes
Expiration	Displays the expiration date of your license in a yyyy-mm-dd format.	✕ No
Status	Displays the current status of your license as either Active or Expired. A green indicator means that your license is active. A red indicator means that your license is inactive or expired.	✕ No

> Warning
> Any DQ Jobs that run on accounts with expired or invalid license keys will no longer run until a valid license key is saved.

React

Switching React MUI on or off

React MUI is on by default. You can switch React MUI on or off.

> Note As of the 2023.02 version, React is still in the process of full implementation across the Collibra DQ app. As such, some pages are not yet available in the React layout.

Steps

1. Hover your cursor over the ⚙ gear icon and select **Settings**.
   » The Limit Settings opens.
2. In the upper-right corner of the Settings page, select **App Config**.
   » The App Config opens.
3. Enter the required information.

Option	Description
REACT_MUI	This flag controls whether React is switched on or off in the Collibra DQ app. You can: ◦ Enter **TRUE** to switch the React MUI on for all available React pages. ◦ Enter **FALSE** to switch the React MUI off.
UX_REACT_ON	This flag must be set to **FALSE**.

4. Click **Save**.

> Warning UX_REACT_ON must be set to **FALSE** to ensure proper functionality of Collibra DQ. If you want to view pages in the React layout, set the REACT_MUI option to **TRUE**.

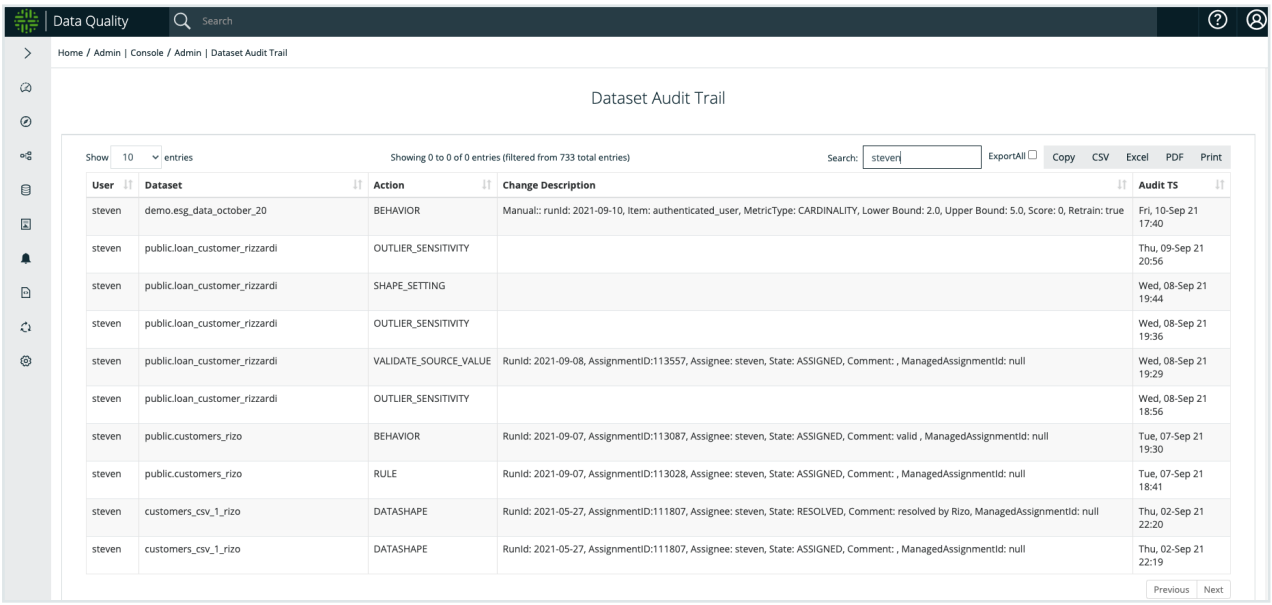
Audit

Built-in auditing allows you to track usage and modifications across data set, security, and user levels.

Dataset Audit Trail

Available Data

- User
- Data Set
- Selected Features
- runID
- Assignments
- Comments
- Timestamp

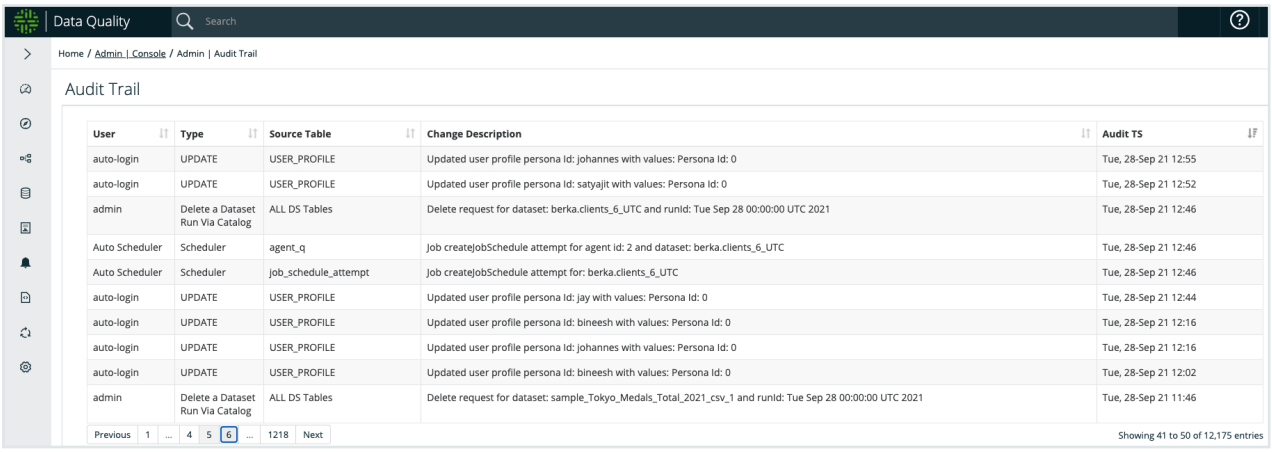


Security Audit Trail

Available Data

- User profile updates
- Role updates

- Reference table from metastore
- Data set deletion requests
- Job schedule attempts



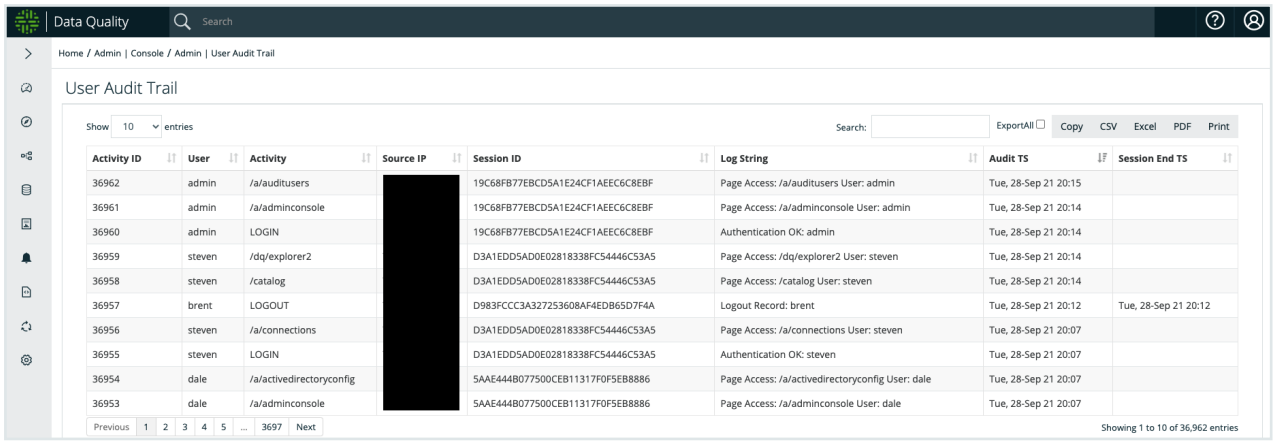
When administrators modify roles mapped to data sets or data sets mapped to roles, changes are documented automatically in the Audit Trail. The information in the entry log includes

- New and original data sets added or removed during the modification.
- New and original roles added or removed during the modification.
- A timestamp of when the modification occurs.
- Type of modification.
- Username by which the modifications are made.

User Audit Trail

Available Data

- Logins: Successful / Failed
- Privileged User Access
- User Actions / Activities
- Account Actions
- Source IP
- Timestamps


Data Quality
Search
Home / Admin | Console / Admin | User Audit Trail
User Audit Trail
Show 10 entries
Search:
ExportAll
Copy CSV Excel PDF Print
Activity ID | User | Activity | Source IP | Session ID | Log String | Audit TS | Session End TS
36962 | admin | /a/auditusers | [redacted] | 19C68FB77EBCD5A1E24CF1AEEC6C8EBF | Page Access: /a/auditusers User: admin | Tue, 28-Sep 21 20:15 |
36961 | admin | /a/adminconsole | | 19C68FB77EBCD5A1E24CF1AEEC6C8EBF | Page Access: /a/adminconsole User: admin | Tue, 28-Sep 21 20:14 |
36960 | admin | LOGIN | | 19C68FB77EBCD5A1E24CF1AEEC6C8EBF | Authentication OK: admin | Tue, 28-Sep 21 20:14 |
36959 | steven | /dq/explorer2 | | D3A1EDD5AD0E02818338FC54446C53A5 | Page Access: /dq/explorer2 User: steven | Tue, 28-Sep 21 20:14 |
36958 | steven | /catalog | | D3A1EDD5AD0E02818338FC54446C53A5 | Page Access: /catalog User: steven | Tue, 28-Sep 21 20:14 |
36957 | brent | LOGOUT | | D983FCCC3A327253608AF4EDB65D7F4A | Logout Record: brent | Tue, 28-Sep 21 20:12 | Tue, 28-Sep 21 20:12
36956 | steven | /a/connections | | D3A1EDD5AD0E02818338FC54446C53A5 | Page Access: /a/connections User: steven | Tue, 28-Sep 21 20:07 |
36955 | steven | LOGIN | | D3A1EDD5AD0E02818338FC54446C53A5 | Authentication OK: steven | Tue, 28-Sep 21 20:07 |
36954 | dale | /a/activedirectoryconfig | | 5AAE444B077500CEB11317F0F5EB8886 | Page Access: /a/activedirectoryconfig User: dale | Tue, 28-Sep 21 20:07 |
36953 | dale | /a/adminconsole | | 5AAE444B077500CEB11317F0F5EB8886 | Page Access: /a/adminconsole User: dale | Tue, 28-Sep 21 20:07 |
Previous 1 2 3 4 5 ... 3697 Next
Showing 1 to 10 of 36,962 entries

Chapter 20

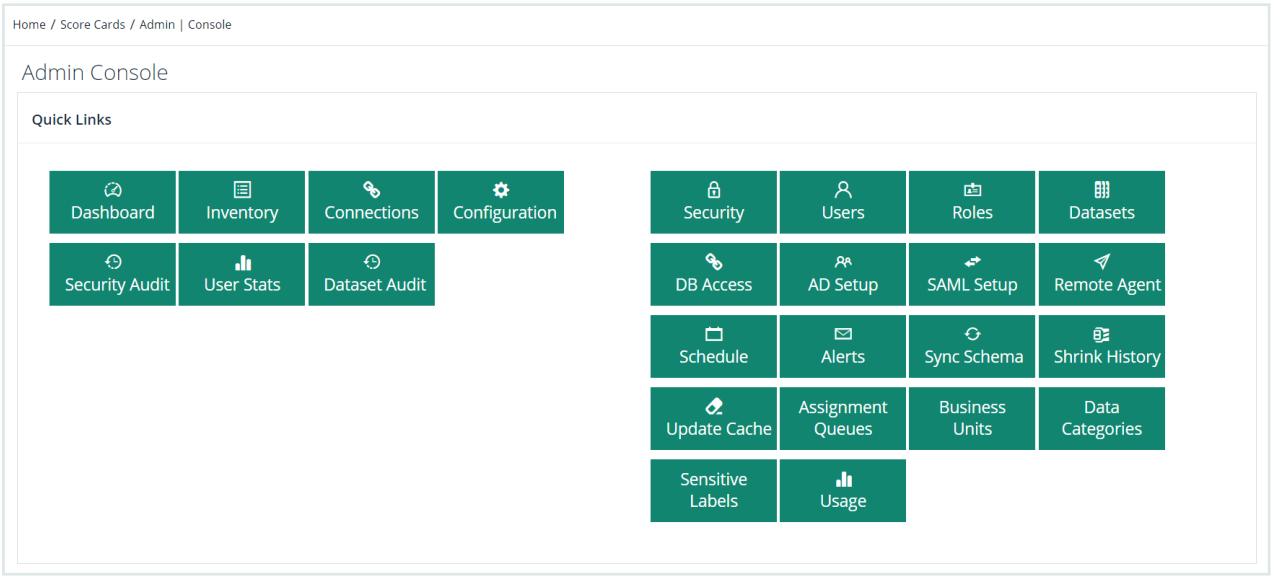
Collibra DQ Security Configuration

Overview

Collibra Data Quality offers multiple methods of user authentication, including a local user store and Active Directory or generic LDAP integration.

Security can be configured to meet your needs. Advanced options to segment groups and roles are available. Additionally, options for SAML and SSO are available.

You can control configurations at the Web (UI), Postgres, and Application layers depending on your security requirements. Encryption is available for data in-transit or at-rest.



Configuration

Configuring Active Directory (AD/LDAP)

Prerequisites

You have Admin permissions (ROLE ADMIN) assigned to your User Profile.

Configure Connectivity from Collibra Data Quality to Active Directory

Steps

1. From the Admin Console, click the **AD Setup** tile.
   >> The **Active Directory Security Settings page** opens.
2. Check the **AD Enabled** checkbox.

3. Enter the required information.

Setting	Description
AD Enabled	AD is enabled when checked.
LDAP Enabled	LDAP is enabled when checked. In most cases, LDAP should be unchecked.
Page Size	Set a value greater than 0 to control query page size. Since some LDAP providers do not support page sizing, this field can either be left blank or set to 0.
Host	The hostname or URL of your LDAP or LDAPS server, for example, ldap://12.345.678.90
Port	The port to connect to your LDAP or LDAPS server. The default ports are **389** for LDAP and **636** for LDAPS.
Base Path	The value entered is the base domain information, for example, **DC=,DC=**.
Group Search Path	The value entered is the domain object path where the groups are located, for example, **OU=OwlGroups,OU=Groups**. A Group Search Path value should not include the value of the Base Path. Note After Group Search Path is configured, it is recommended that you restart the DQ Web App.
Domain	Optional. The domain name used to signify when non-local users log in. Only used for AD configurations.
User Search Base	Optional. The base DN of where the LDAP users for Collibra DQare located, for example, **CN=Users**. This is the lowest level container (OU) of user objects. When set, it is used to narrow down user search at login. A User Search Base value should not include the value of the Base Path. Note In order to use User Search Base to properly sign in, it is required that you restart the DQ Web App.

Setting	Description
User Search Filter	Optional. When set, this LDAP filter is used to locate users at login. This filter is based on your LDAP configurations. Note In order to use User Search Filter to properly sign in, it is required that you restart the DQ Web App.
Group Search Base	Optional. The base DN where all the groups are located. Only used for LDAP configurations.
Group Search Filter	Optional. The LDAP filter used to narrow down group objects located under a base DN. Only used for LDAP configurations.
Bind User	The DN of an admin user that is used for authentication, for example, *admin@collibra.com*.
Bind Password	The password of an admin user. Enter a **Bind Password** for the bind account.

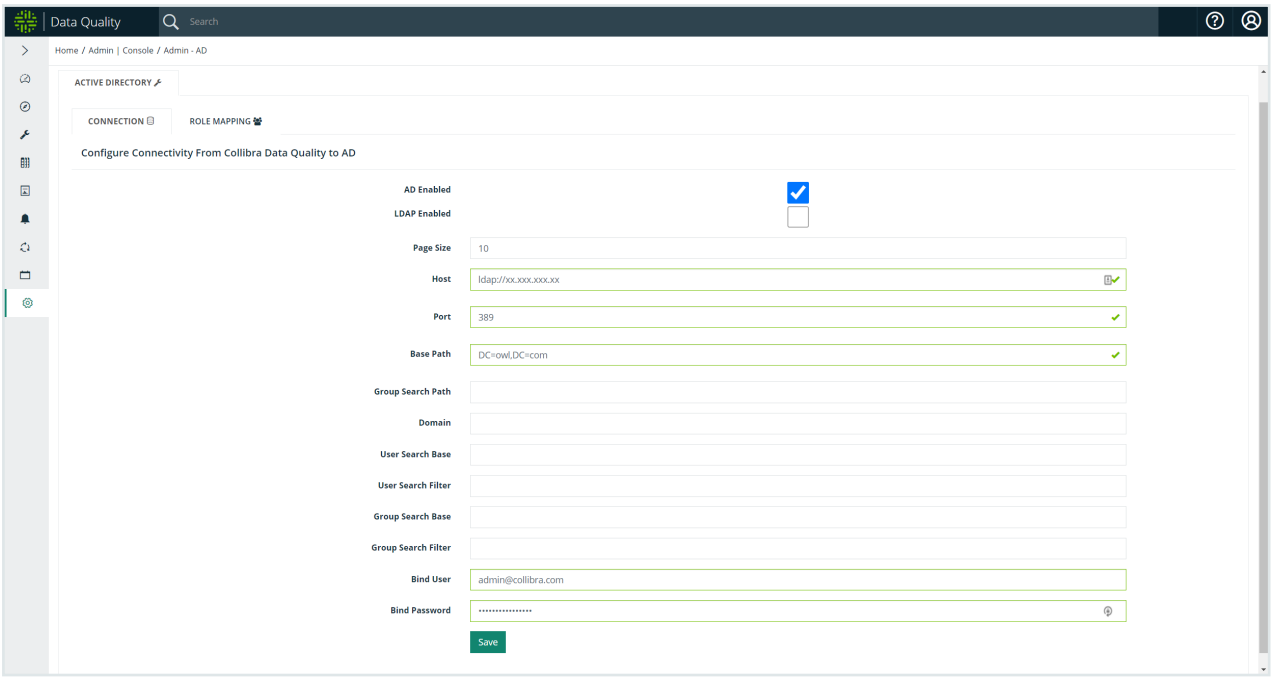
4. Click **Save**.

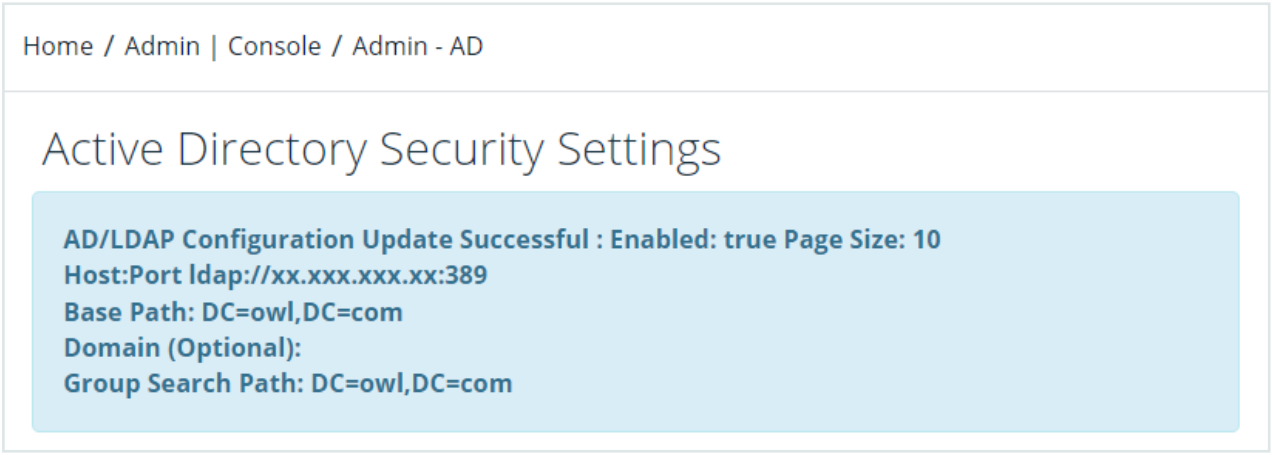
Configuration / ENV settings within owl-env.sh

Be sure to add the following script settings when configuring LDAP and Active Directory.

> Note This configuration occurs at start-up of the DQ web app. See Standalone Install for a complete list of the owl-env.sh scripts.

OWL-ENV.SH Scripts	Meaning
export LDAP_GROUP_RESULT_DN_ATTRIBUTE	The attribute to the full path of the group object, for example, **CN=OwlAppAdmin**, **OU=OwlGroups**,**OU=Groups**,**DC=owl**, **DC=com**. Default is **distinguishedname**.
export LDAP_GROUP_RESULT_NAME_ATTRIBUTE	The attribute to the simple name of the group, for example, **OwlAppAdmin**. Default is **CN**.
export LDAP_GROUP_RESULT_CONTAINER_BASE	Property used in the scenario where the LDAP_GROUP_RESULT_DN_ATTRIBUTE does not return a value. In this case, the LDAP_GROUP_RESULT_NAME_ATTRIBUTE prepends to this value, which creates a fully qualified LDAP path. For example, **OU=OwlGroups**, **OU=Groups**,**DC=owl**,**DC=com**. Default is <**null**>.





When binding to Active Directory, you do not need a special Bind User and Password. Collibra Data Quality only requires an admin user account with which to bind in order to run a read-only query on the groups. Collibra Data Quality uses AD credentials dynamically to understand what groups you want to map, but the credentials are never stored.

See AD Group to Role Mapping to learn how to map an AD Group to a Collibra Data Quality role.

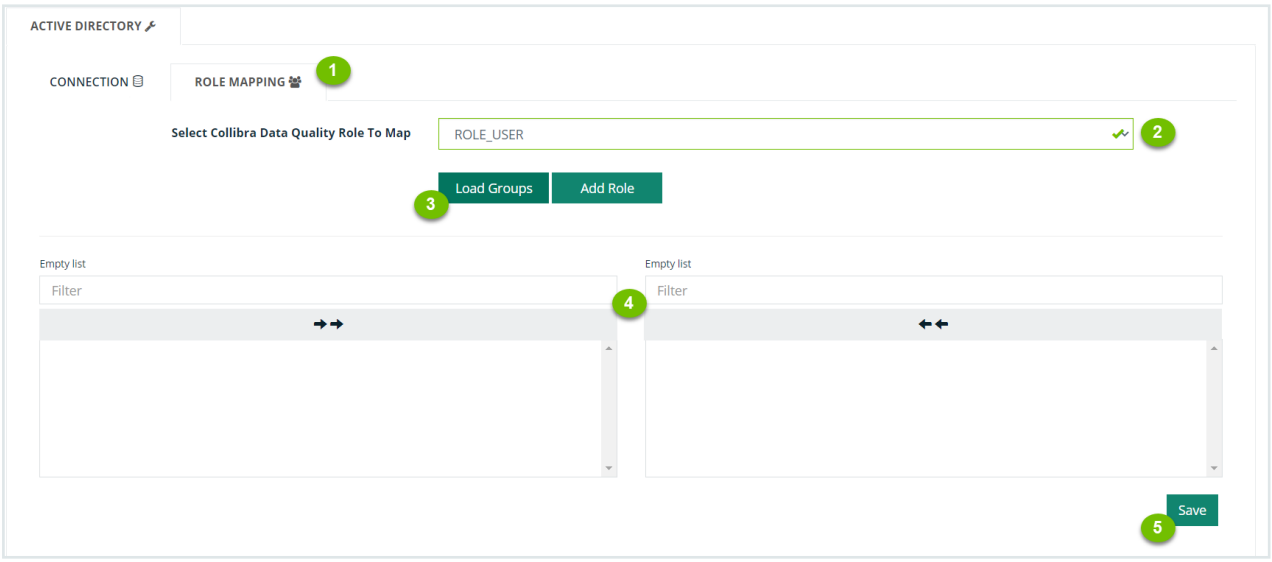
AD Group to Role Mapping

When you map an AD Group to a Collibra Data Quality role, you grant all users from the selected AD Group role-based access to the selected Collibra Data Quality role outlined in the steps below. You can find additional information on creating custom application roles on the Role-Based Access Control page.

Application properties set in owl-env.sh can be set to determine which LDAP properties correspond to LDAP query results. For group mapping, you need the **full path** (unique) and the **display name**.

For example:

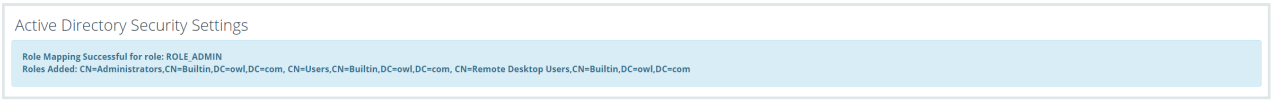
```
LDAP_GROUP_RESULT_DN_ATTRIBUTE=distinguishedname
LDAP_GROUP_RESULT_NAME_ATTRIBUTE=CN
```



1. Click the **Role Mapping** tab.
2. **Select a role** from the dropdown. Alternatively, you can add a new Collibra Data Quality Role to map the AD Group(s) you want to include by clicking the **Add Role** button.
3. Click **Load Groups**. The list box on the left will populate with roles in the group you selected.
4. **Click a role** from the list box on the left to move it to the selection box on the right.

You can use the Filter field to filter the lists in either box.

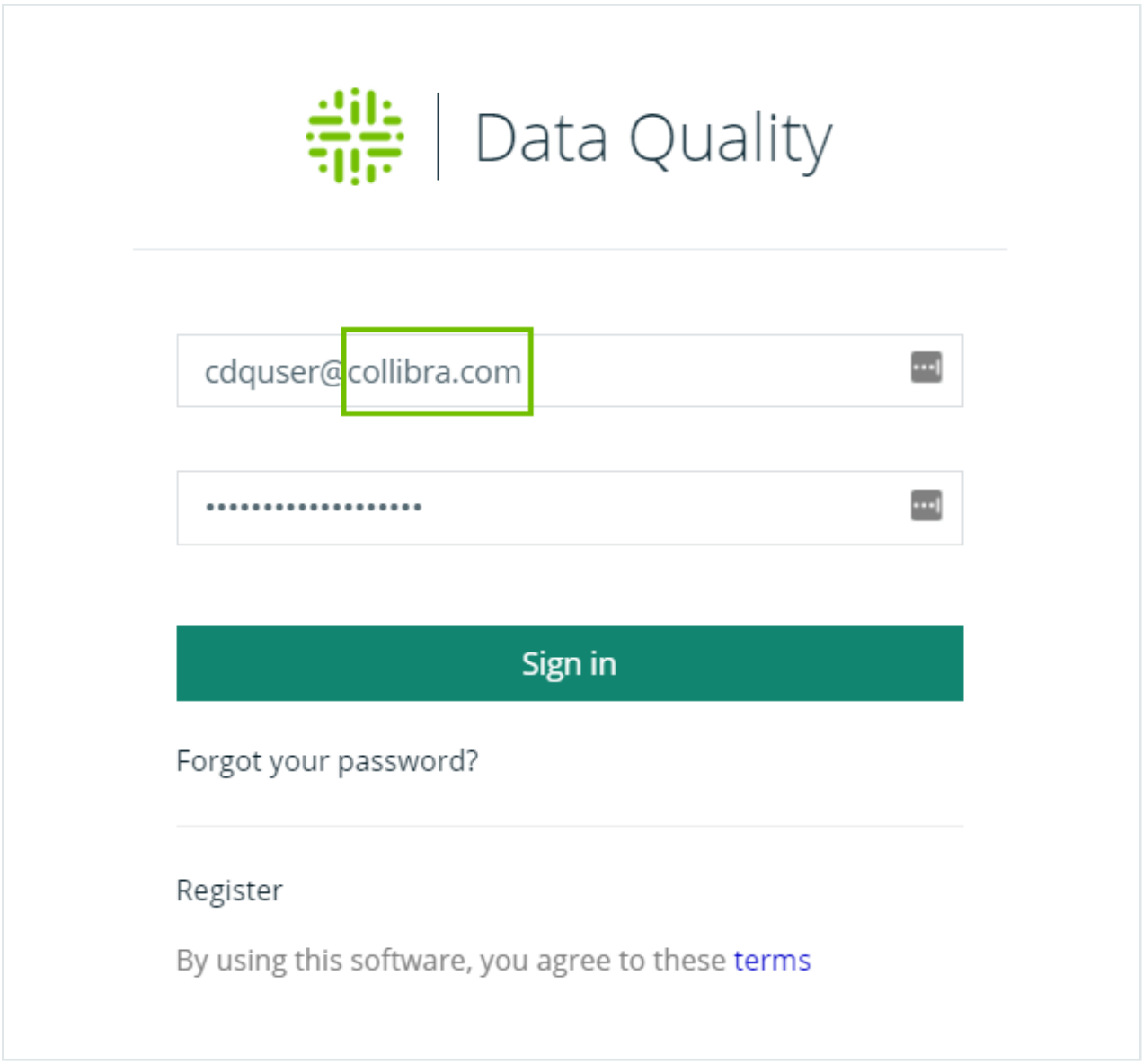
5. Click **Save**.



Once you successfully map an AD Group to an AD Role, log out of Collibra Data Quality and log in again as a domain user.

> Note You must restart Collibra Data Quality by running ./owlmanage.sh restart_owlweb when toggling AD Enabled.

When logging into the Collibra Data Quality web application, make sure to append the domain to the end of the username.



Connection Security

You can select and map any role to your connection on the Connection Manager page, so that any user with those mapped roles will see the connection in the UI.

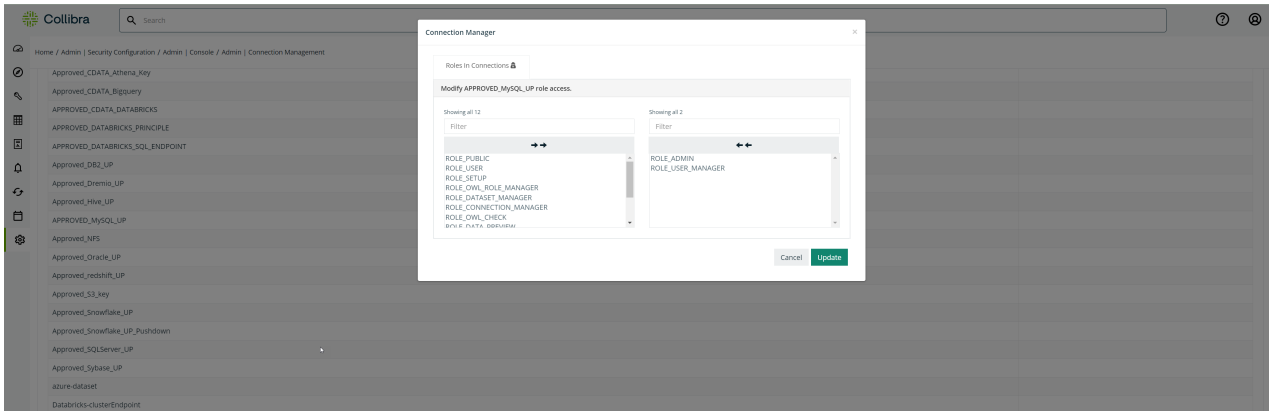
Steps

1. Log in to the Collibra DQ instance and click the gear icon in the left navigation pane.

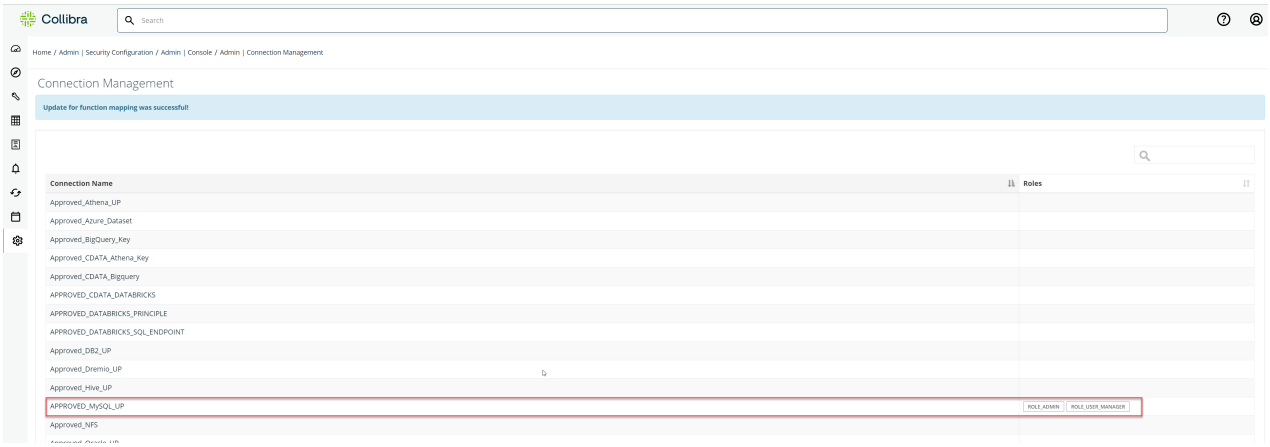
2. Click **Admin Console**.
3. In Quick Links, click the **Security** tile.
4. Click the **DB Connection Security**check box.
5. Click **Save**.
6. Click **Admin Console** again.
7. Click **DB Access**.
8. From the Connection Name column, select your connection.
   The Connection Manager displays the available roles.
9. Click the role(s) you want to map to your connection from the left column, which moves the role(s) to the right column.

> Note You can move roles from the left column to the right column and back by clicking the roles in the Connection Manager.

10. When you have completed your role selections, click **Update**.



The role(s) you selected for the connection displays in the Roles column.



Any user with those mapped roles can now see the connection in the UI.

General Collibra DQ Security Settings

You can configure general Collibra DQ security settings in the Admin console.

Steps

1. Login to the Collibra DQ instance and click the gear icon in the left navigation pane.
2. Click **Admin Console**.
3. Click **Security**.
4. Enable/disable the options in the following table.

Setting	Description
Local User Store Enabled *	When this setting is selected, you can create and administer users stored in Collibra DQ's internal database user store. For more information, go to Local User Store Authentication.
Auto Approve Users	When this setting is selected, local users are approved automatically upon login.

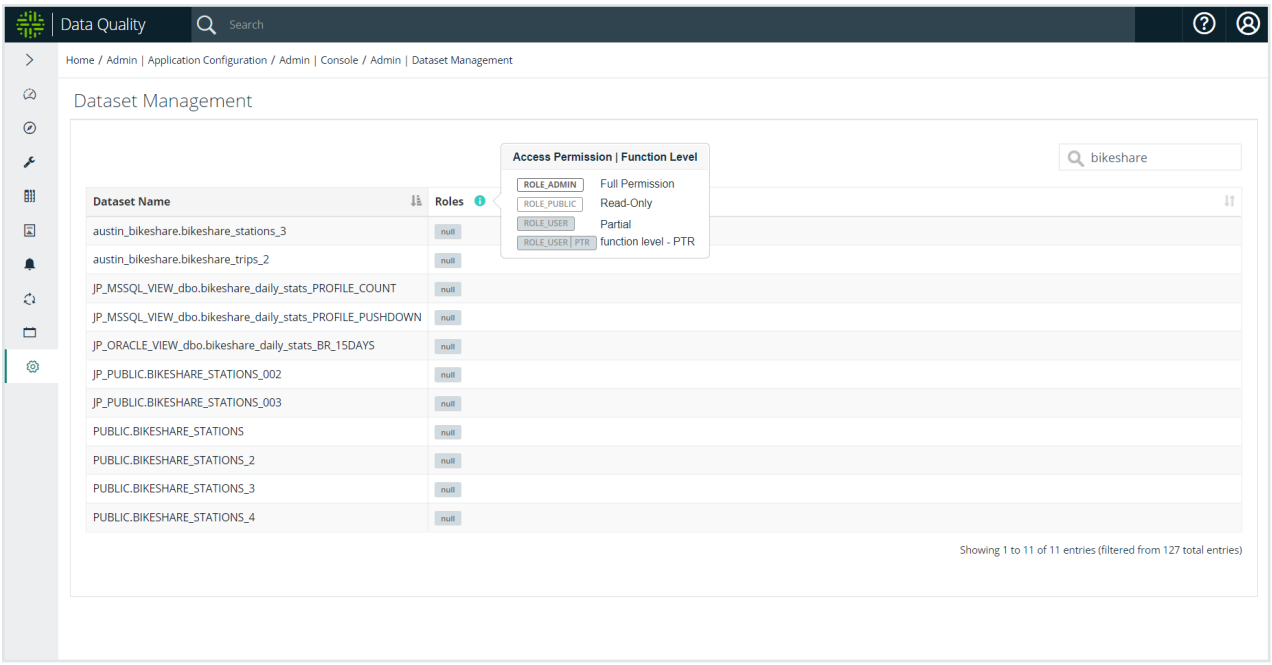
Setting	Description
DB Connection Security	When this setting is checked, you can select and map roles to your connection. > Note Any user with those mapped roles can see the connection in the UI.
DQ Job Security	When this setting is checked, the OWL CHECK role is required. > Note The OWL CHECK role is the only role that can run DQ scans when DQ Job Security is enabled.
Allow Temp File Upload for DQ Job	When this setting is checked, users can upload temp files for DQ Jobs.

Dataset Security Settings

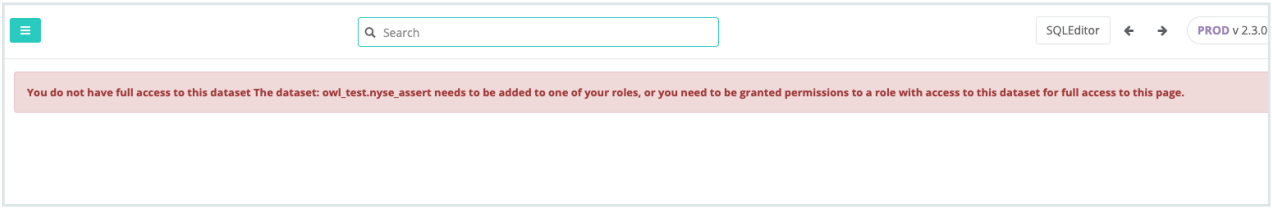
You can configure Dataset Security settings in the Admin console.

Steps

1. Login to the Collibra DQ instance and click the gear icon in the left navigation pane.
2. Click **Admin Console**.
3. Click **Security** on the Quick Links page and select **Dataset Security**.
4. Click **Save**.
5. Click the gear icon again.
6. Click **Admin Console**.
7. Click **Datasets** on the Quick Links page.



Note all the Datasets from the demo script we launched at the beginning of this document have been added to ROLE_PUBLIC ROLE defined in Collibra Data Quality except the row_count Dataset. At this point, we already have the odemo@owl.com user created and mapped to the ROLE_PUBLIC ROLE which has access to those Datasets. Lets login as odemo@owl.com and try to access row_count DatasetStats page (for an understanding on how to access the DatasetStats page see the section entitled “Understanding the DQ DatasetStats Page” in this document…..doing so will result in the error message below.



However, the other DatasetStats that are part of the PUBLIC_ROLE as odemo@owl.com is a member of that ROLE.

The last thing to notice is that, as the user odemo@owl.com cannot access the Admin pages the AD Group odemo is a part of the ADMIN_ROLE.

ACL Security

When ACL is enabled/disabled, an administrator can configure the following options in the Dataset Security pane to limit usage/permissions on datasets:

- Data_Preview role limit (role that can view source data)
- Dataset_train role limit (role that can train data sets)
- Dataset_rules role limit (role that can add / edit / delete rules)

> Note This configuration is tied to data sets and not connections or jobs.

Data Set Masking

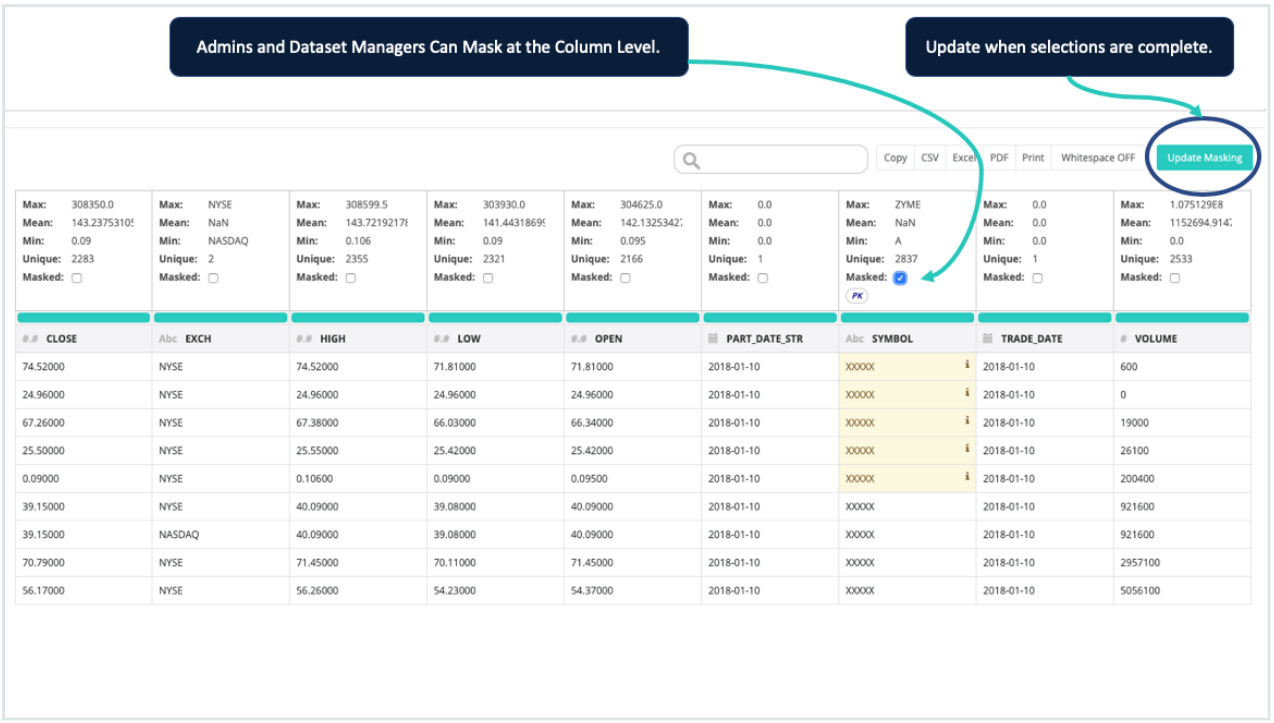
Add column level masking for sensitive data in a data set.

Masking from the UI

After an DQ check runs, you can perform column level masking from the findings page if you have ROLE_ADMIN or ROLE_DATASET_MANAGER assigned to your user.

> Note Masking updates will take effect for all existing and future runs of the selected dataset.

> Note ROLE_ADMIN and ROLE_DATASET_MANAGER also have the ability to unmask.

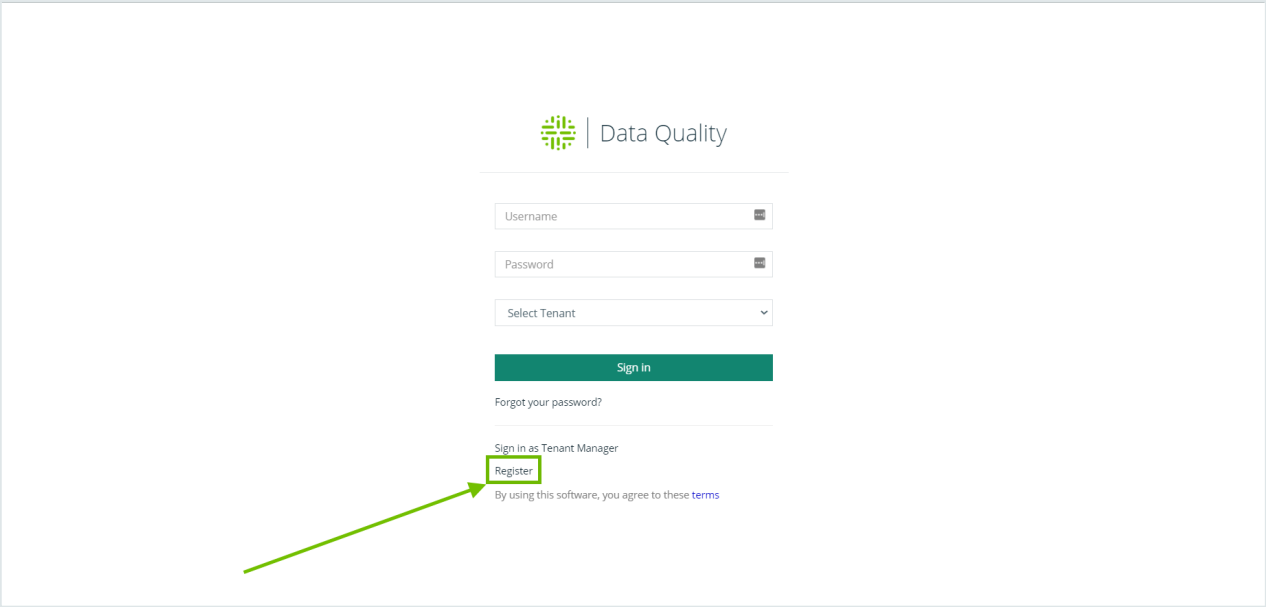


Local User Store Authentication

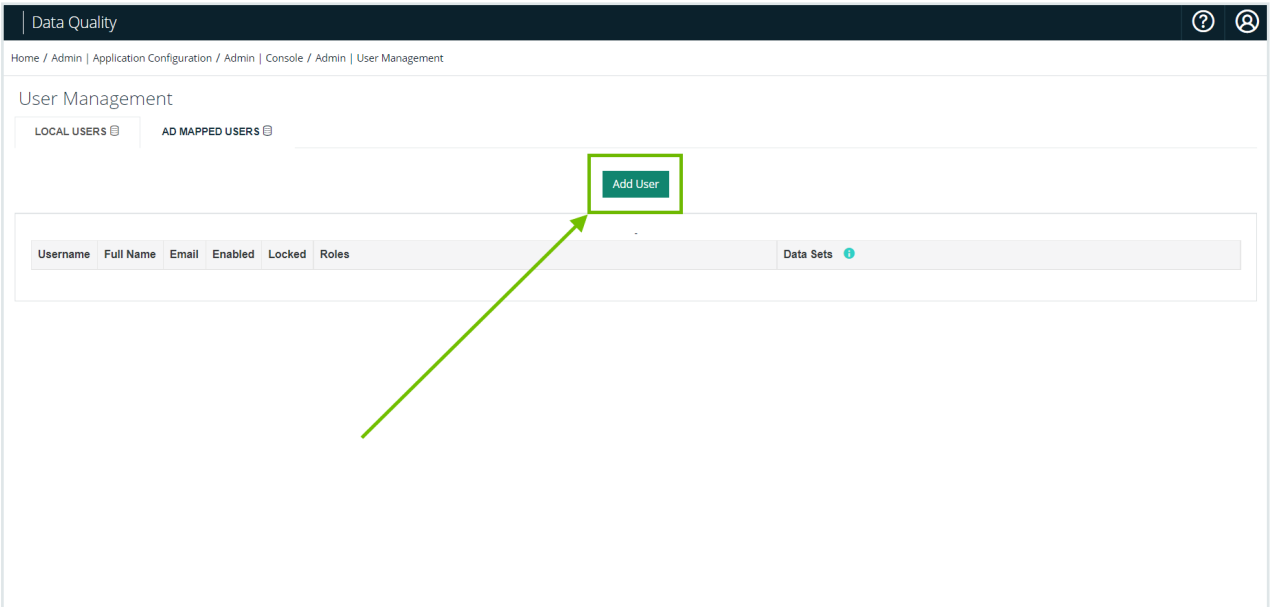
Local user store authentication is enabled by default, and the Collibra Data Quality & Observability ships with a default user with admin privileges. If you should need to enable/disable this mechanism, you can find the setting by navigating to the Admin Console and clicking on the **Security** icon. The setting labeled **Local User Store Enabled**, when selected, allows your company to create and administer users stored in Collibra Data Quality's internal database user store. Enabling/disabling this feature requires a restart of the web application.

Adding Local Users

Users can request access to the application from the login page by clicking the **Register** link beneath the Sign in button.



Application administrators can also create user accounts by navigating to the Admin Console and clicking the **Users** tile.



In either case, simply fill out the account form and a user record gets created. The user account at this stage does not have access to the application until an application administrator grants that user access to a Collibra DQ role or multiple roles.

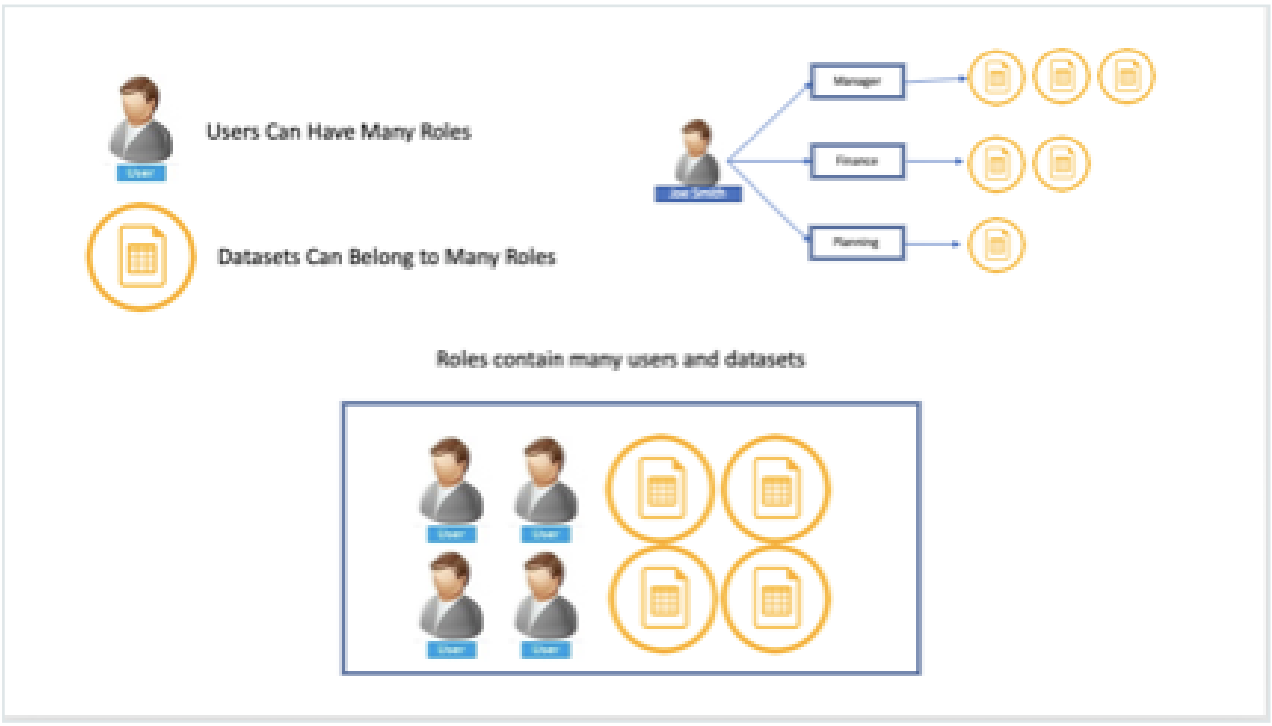
> Note If LDAP is enabled and a local user password or username is invalid and fails to authenticate, Collibra DQ attempts to authenticate against the AD/LDAP server.

Granting Local User Roles

Application administrators can manage user accounts and role access by navigating to the Admin Console and clicking the **Users** tile. On that page, locate the user you would like to modify and click their username in the table of users. This loads a pop-up window that allows you to modify the user account, including Enabled/Disabled, as well as the roles they belong to. Click the **Roles** tab to begin granting roles to the selected user. The available roles established are listed on the left. Click a role from the left-hand side to apply that role to the user. You can also add all roles by clicking the double arrow icon above the list. When you are satisfied with the role listings on the right-hand side for that user, click the **Update** button. To create custom application roles, go to Role Based Access Control (RBAC).

Role Based Access Control (RBAC)

The image below depicts Collibra Data Quality's security architecture.



Whether leveraging a Local User Store, Active Directory, or using the out of the box user accounts that come with Collibra Data Quality & Observability via LDIF, security stays the same. An admin can create many ROLEs. A user, whether local user, LDIF user, or AD user can be part of one or many roles. And a ROLE maps to a data set within Collibra Data Quality.

When dataset security is enabled and you want to access a dataset, or want to see, add, or remove an existing business unit for a dataset, you must have a role that is attached to that dataset.

For datasets, when dataset security and default dataset owner access is enabled, a user with a role attached to a dataset **or** the dataset owner can:

- **Add** - User with no dataset access (with no role attached to any existing dataset) can still create a dataset. After creating it, this user (who is the default dataset owner) can see the dataset, profile, and business units, and add and remove business units to their (owned) dataset.
- **Retrieve/See** - User can retrieve/see datasets, based on dataset access.
- **Edit** - User can edit datasets, based on dataset access.
- **Remove** - User can remove datasets, based on dataset access.

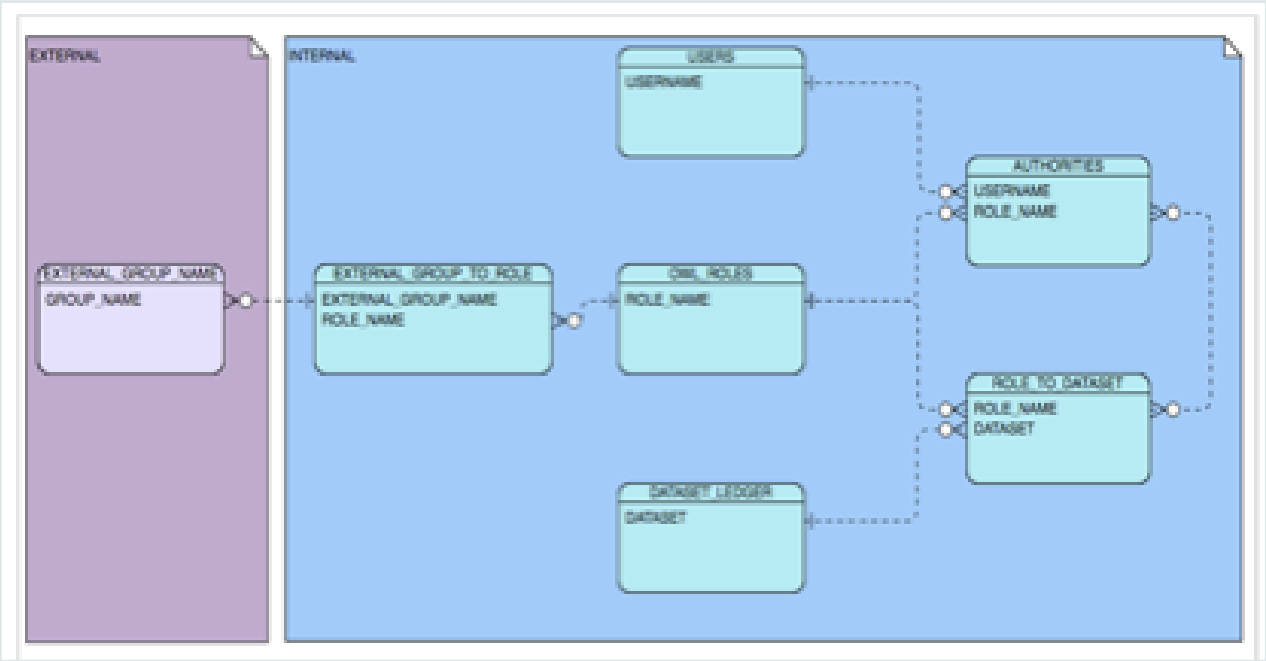
For business units, when dataset security and default dataset owner access is enabled, a user can:

- **Retrieve/See** - User can retrieve/see business units, based on dataset access.
- **Edit** - User can edit business units, based on dataset access.
- **Remove** - User can remove business units, based on dataset access.

> Note You must be an admin to create a business unit, which can then be added to a dataset.

A unique feature within Collibra DQ is the fact that we do not store information about external user accounts. This avoids the need to sync external users from an external user store such as AD to Collibra DQ. Instead, Collibra DQ will map the external group to an internal role. From here the ROLE can be mapped to the different functionality within Collibra DQ whether they are Admins / Users / and have access to different datasets and future functionality. The other benefit is that if a specific userid within the external user store is terminated, when the user is purged from the external user store such as AD they

will immediately not have access to Collibra DQ's web application. This is because when the user logs into Collibra DQ's web application that is backed by AD their login will interrogate AD to authenticate the user account. See logical flow below for how the group to role mappings work.



RBAC Usages

Collibra DQ supports RBAC configuration with both core roles and custom roles. The following table shows the core roles of Collibra DQ's RBAC configuration:

Role	Access Description
ROLE ADMIN	Allows you to modify any access, config settings, connections, and role delegation.
ROLE DATA GOVERNANCE MANAGER	Allows you to manage (create / update / delete) Business Units and Data Concepts.
ROLE USER MANAGER	Allows you to create or modify users and add users to roles.
ROLE OWL ROLE MANAGER	Allows you to create roles and edit role mappings to users, AD groups, and datasets.

Role	Access Description
ROLE CONNECTION MANAGER	Allows you to add, edit, and delete connections.
ROLE DATASET MANAGER	Allows you to create or modify datasets to roles and mask dataset columns.
ROLE OWL CHECK	This is the only role that can run DQ scans when DQ Job Security is enabled.
ROLE DATA PREVIEW	This is the only role that can view source data if Data_Preview security is enabled.
ROLE DATASET TRAIN	This is the only role that can train datasets if Dataset_Train security is enabled.
ROLE DATASET RULES	This is the only role that can add / edit / delete rules if Dataset_Rules security is enabled.
ROLE VIEW DATA	Controls which users can access the DQ SQL editor to run the SQL against the database.
ROLE PUBLIC	Public: Access to scorecards, no dataset access when dataset security is enabled.
ROLE USER	Do not use.
ROLE SETUP	Do not use.

Custom roles can be added via the Role Management page by navigating to the Admin Console and clicking on the Roles Icon. Custom roles can also be added 'on the fly' during the Active Directory Role Mapping step.

It is these custom roles that will determine the users that have access to data sets (including profile/rules/data preview/scoring), and database connections

Additional information regarding setting up Dataset and Connection security can be found in those documents respectively.

SAML Authentication

You can integrate Collibra DQ with an existing SAML solution and have your application act as a service provider. Once you set up the environment variables, you can access and configure SAML security settings as an administrator in the **SAML Setup** section of the **Admin Console**.

Set the SAML authentication properties

Before configuring SAML authentication, you must add the following required properties to your configuration.

Standalone installation

1. Add the properties as environment variables to your **owl-env.sh** file located in **<installation_directory>/owl/config/**.
2. Prefix all properties with the `export` statement.
3. Restart the web app.

Cloud native installation

1. Add the properties as environment variables to your **owl-web ConfigMap**.
2. Recycle the pod.

Required properties

Property	Description
SAML_ENABLED	Whether Collibra DQ uses SAML. If set to `false`, users sign in with a username and password. If set to `true`, SAML handles the authentication request.
SAML_ENTITY_ID	The name of the application for the identity provider, for example *Collibra DQ*. It is an immutable unique identifier of the service provider for the identity provider (IDP).

> Warning Please see CORS_ALLOWED_ORIGINS in the **Optional properties** section below if you have SAML configured in Collibra DQ, or if the app sits behind a load balancer.

Optional properties: general

You can further configure your SAML setup with the following optional properties.

Property	Description
CORS_ALLOWED_ORIGINS	Allows cross-origin requests between DQ and SAML. Replace {IDP-BASE-URL} with the value of the actual IdP URL. For example, https://ping.auth.com/ Replace with the value of the actual DQ Base URL(https://dq-env.com).
SAML_ENTITY_BASEURL	The base URL that is provided in the service provider metadata. Note Set this property when you use DNS.
SAML_LB_EXISTS	Whether the application needs to configure a load balancer. You generally need this setting only when the **Load Balancer** is set for **SSL Termination**. The default value is `false`. If set to `true`, you must also provide a value for `SAML_LB_SERVER_NAME`.
SAML_METADATA_USE_URL	Whether Collibra DQ uses a URL or a file for the identity provider metadata. The default value is `true`. If set to `false`, the file must be accessible to the owl-web and the path provided in the **Meta-Data URL** field of the **Meta Data Configurations** section under **Admin Console** --> **SAML Setup** --> **Connection**.
SAML_ROLES_PROP_NAME	The attribute name that contains role ID/name in the incoming SAML assertion. The default value is `memberOf`.

Property	Description
SAML_GRANT_ALL_PUBLIC	Whether any user authenticated by the identity provider is allowed to login the Collibra DQ application. The default value is `true`.
SAML_USER_NAME_PROP	The name of the attribute in the SAML response that contains the username of the user who is authenticating.
SAML_TENANT_PROP_NAME	If using multi-tenant mode, the variable in which the identity provider stores the tenant name of the user authenticating in the SAML response. The app will attempt to use the `RelayState` parameter to identify the tenant and then fall back on this property.
SAML_TENANT_PROP_FROM_URL	If using multi-tenant mode, this property controls where the tenant is set to use for authentication. The default value is `false`. If set to `true`, the app attempts to use the `RelayState` property to identify the tenant. If it is set to `false`, the app attempts to find the tenant in a SAML response attribute, which is defined by the `SAML_TENANT_PROP_NAME` property (default=`tenant`).
SAML_KEYSTORE_FILE	The path to the keystore for SSL validation. The store should contain the keypair of the identity provider for SSL verification.
SAML_KEYSTORE_PASS	The password for the keystore provided in `SAML_KEYSTORE_FILE`.
SAML_KEYSTORE_ALIAS	The alias of the keypair (private and public) in the keystore used for SSL verification.

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>LOCAL_
REGISTRATION_
ENABLED</td><td>An optional property, which allows you to display or hide the registration link for local users on the Collibra DQ sign-in page. Because the registration link is displayed on the sign-in page by default, this property is set to true.

To display or hide the registration link from the owl-env.sh file, select from the following properties:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>export LOCAL_
REGISTRATION_
ENABLED=true</td><td>Displays the registration link on the sign-in page for local users. This is set to <code>true</code> by default.</td></tr>
<tr><td>export LOCAL_
REGISTRATION_
ENABLED</td><td>Hides the registration link on the sign-in page for local users.</td></tr>
</table>
To display or hide the registration link from a K8s config map, select from the following properties:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>LOCAL_
REGISTRATION_
ENABLED="true"</td><td>Displays the registration link on the sign-in page for local users. This is set to <code>true</code> by default.</td></tr>
<tr><td>LOCAL_
REGISTRATION_
ENABLED="false"</td><td>Hides the registration link on the sign-in page for local users.</td></tr>
</table>

Note This property must be set globally on deployment.</td></tr>
<tr><td>SAML_MAX_
AUTH_AGE</td><td>The number of seconds that an IdP authentication is accepted by the application. If the IdP authentication occurred outside this time range, the application considers the value too old to trust and the authentication is not accepted.

The default is <code>14400 seconds</code> (4 hours).</td></tr>
<tr><td>CSRF_TOKEN_
ENABLED</td><td>Enables the use of Spring CSRF token implementation.</td></tr>
</table>

Property	Description
CSRF_TOKEN_IGNORE_URL_PATTERN	Requests URL patterns to be ignored for CSRF token validation.
CSRF_TOKEN_IGNORE_REFERER_PATTERN	Requests header referer patterns to be ignored for CSRF token validation.

> Note While CORS is still an optional configuration, it is required if you have SAML configured in DQ, or if you have DQ behind a load balancer.

> Note CORS is also enforced for multi-tenancy.

Optional Properties: Metadata

When `SAML_METADATA_USE_URL` is set to `true` (default), the following additional properties are available.

Property	Description
SAML_METADATA_TRUST_CHECK	Whether to enable Collibra DQ to do trust verification of the identity provider. The default value is `false`.
SAML_METADATA_REQUIRE_SIGNATURE	Whether Collibra DQ signs authentication requests to the identity provider. The default value is `false`.
SAML_INCLUDE_DISCOVERY_EXTENSION	Whether to enable Collibra DQ to indicate in the SAML metadata that it's able to consume responses from an IDP Discovery Service. The default value is `false`.

Optional Properties: Load Balancer

When `SAML_LB_EXISTS` is set to `true`, the following additional properties are available.

Property	Description
SAML_LB_INCLUDE_PORT_IN_REQUEST	Whether to include the port number in the request. The default value is `false`.
SAML_LB_PORT	The port number of the load balancer. The default value is `443`.
SAML_LB_SCHEME	The protocol of the load balancer. The default value is `https`.
SAML_LB_SERVER_NAME	The server or DNS name. Usually, the same as `SAML_ENTITY_BASEURL` without specifying the protocol, for example without *https://*. This property is required and has no default.
SAML_LB_CONTEXT_PATH	Any path that may be defined on the load balancer.

Example

```
#enable SAML & show the SAML SSO option on the login page
SAML_ENABLED=true

#set SSL communication properties for SAML
SAML_KEYSTORE_FILE=/keystore.p12
SAML_KEYSTORE_PASS=****
SAML_KEYSTORE_ALIAS=****

#in multi-tenant mode set the name of the IDP variable to hold
the tenat name
SAML_TENANT_PROP_NAME=tenant

#set the name of the IDP variable to hold the user roles in the
response
SAML_ROLES_PROP_NAME=memberOf

#allow login if authenticated to the IDP
SAML_GRANT_ALL_PUBLIC=true
```

```
#set the EntityId of the application to be supplied to the IDP
SAML_ENTITY_ID=OwlOneLogin

#optinally set a property that contains the username in the
response
SAML_USER_NAME_PROP=""

#optionally use a file for the IDP metadata vs a URL (default
is true)
SAML_METADATA_USE_URL=false

#optional security settings to
SAML_METADATA_TRUST_CHECK=false
SAML_METADATA_REQUIRE_SIGNATURE=false
SAML_INCLUDE_DISCOVERY_EXTENSION=false
```

Download service provider metadata for the IDP

Once you have enabled and configured SAML authentication, you can download the service provider metadata that is required by your identity provider from `https://<your_dq_environment_url>/saml/metadata`.

Enable the SAML sign in option

When you are ready with your IDP settings, add the final configuration settings in the **Admin Console**:

Steps

1. Sign in as an existing administrator with a username and password to the tenant you want to configure.
2. In the **Admin Console**, click **SAML Setup**.
3. In the **Connection** tab, select the **SAML Enabled** checkbox.
4. In the **Meta Data Configurations** section, click **+Add**.

5. Enter the required information.

Option	Description
SAML_METADATA_URL	The Identity Provider HTTP URL that returns the SAML metadata document.
SAML_METADATA_LABEL	The unique label to identify the SAML metadata configuration.
SAML_IDP_URL	The URL to be used for SSO initiated from the Collibra DQ login page.

6. Click **Save**.

> Tip Once you complete this setup, restart your application and sign in using the SAML SSO option.

> Note SAML SSO authentication via the /v3/auth/signin API is not supported.

Multi-tenancy support through SAML RelayState

As of Collibra DQ version 2021.11, in a multi-tenant environment, you can route SSO to the proper tenant with the SAML provided `RelayState` or `SAML_TENANT_PROP_FROM_URL` properties.

While in a multi-tenant environment, if you set the `SAML_TENANT_PROP_FROM_URL` property to `true`, the application attempts to use the `RelayState` parameter to identify the tenant. When you set `SAML_TENANT_PROP_FROM_URL` to `false`, the app attempts to find the tenant in a SAML response attribute, which is defined by the `SAML_TENANT_PROP_NAME` property (default=`tenant`).

You can set the `RelayState` property in the in the **SAML Setup** section of the **Admin Console**.

For more information, go to SAML Authentication.

Securing Passwords

Security is of the utmost importance for Collibra DQ and our customers. To avoid sending plain text passwords when you run DQ Jobs from the command line, you can encrypt passwords instead. To encrypt your password, execute the following command:

```
owlmanage.sh encrypt=password
```

The output password should look similar to the following example:

```
Q+Ri1S+ljpG+fDefXLY4/vXtUosspAoL
```

You can use this password in any DQ Job from the command line where you would normally use a plain text password.

The following is an example of a DQ Job with an encrypted password instead of a plain text password:

```
./owlcheck -q "SELECT id, browser->'$.name' browser FROM events"
-c "jdbc:mysql://54.212.36.218:2212/test" -u "owl" -p
"Q+Ri1S+ljpG+fDefXLY4/vXtUosspAoL" -driver "com.mysql.cj.jd-
bc.Driver" -lib "/opt/owl/drivers/mysql8" -ds jsonremotemysql -
rd "2022-07-25"
```

If you run a DQ Job from within the DQ Web UI, it automatically encrypts your password, eliminating the need to manually encrypt it. For added security, all passwords are masked in the logs and plain text passwords are never stored.

SSL Setup (HTTPS)

By Default Collibra DQ has plain HTTP enabled for testing. When you are ready to enable SSL for the web application, you can set the following environment variables in `owl-env.sh` to enable HTTPS.

> Note The URL you are using must include a fully qualified domain name.

The settings listed at the bottom of this page disable un-secure HTTP, enable secure HTTPS, and allow you to point to your certificate key store + credentials. A restart of the web application is required.

> Warning You must restart the web application from the bin directory: `./owlmanage.sh restart=owlweb`.

Before starting the SSL setup, please generate a keystore.

Generate a Keystore

You can use a Keytool JDK utility to generate a keystore, which helps to manage private and public keys and associated certificates. With this utility, you can administer your own public/private key pairs and associated certificates. The keystore allows you to protect your private keys with a password.

All certificates within the keystore are associated with a unique alias that gets used as a pointer to perform keytool operations, such as import, export, delete, and/or change certificates and keys.

> Note Implementation is dependent upon your specific architecture.

Use something similar to the following command to create the keystore.

```
keytool -genkey -alias <your alias> -keyalg RSA -keysize 2048 -
keystore <file name>
```

Generate a Certificate Signing Request (CSR) from the Keystore

You must create a CSR from the Keystore to share with the Certificate Authority (CA). This allows you to obtain a valid SSL certificate, which is signed by the CA, to secure your site.

Use a command that is similar to the following to create the CSR.

```
keytool -certreq -alias <your alias> ssl -keystore <file name>
```

> Note When prompted, enter the alias name and password.

You can now submit the CSR to a CA for signing. The CA will send you a CA-signed certificate file.

Generate a PCKS12 File Using the Public and Private Key from CA

You will receive a bundle from CA containing your public and private key, which you can then export by using a command such as the following.

```
export SERVER_SSL_KEY_STORE: <path to your key store>
```

You can call Collibra DQ's built-in 256-bit encryption for the SERVER_SSL_KEY_PASS value from the bin directory: `./owlmanage.sh encrypt=<sensitive plain text string>`. Use the response value instead of the plain text value to secure your password.

> Note Encrypting the password with `owlmanage.sh` is optional.

```
export SERVER_SSL_KEY_PASS=<secure result from owl encryption
script>
```

For Standalone Deployments

```
export SERVER_HTTP_ENABLED=false
export SERVER_HTTPS_ENABLED=true
export SERVER_REQUIRE_SSL=true

####START KEYSTORE SETTINGS####
export SERVER_SSL_KEY_TYPE=PKCS12
#SET PATH TO KEYSTORE
export SERVER_SSL_KEY_STORE=KeystorePathHere
export SERVER_SSL_KEY_PASS=*******
export SERVER_SSL_KEY_ALIAS=keystoreAliasNameHere
```

> Note The most common SSL types are JKS and PKCS12.

For Kubernetes Deployments

```
SERVER_HTTP_ENABLED: false
SERVER_HTTPS_ENABLED: true
SERVER_REQUIRE_SSL: true
SERVER_SSL_KEY_TYPE: PKCS12
SERVER_SSL_KEY_STORE: KeystorePathHere
SERVER_SSL_KEY_PASS: *******
SERVER_SSL_KEY_ALIAS: keystoreAliasNameHere
```

> Note Add standalone values to the `owl-env.sh` and Kubernetes values to the configmap.

Collibra DQ Legal

Agreements

This section contains legal agreements specific to Collibra.

Collibra Evaluation Agreement

For the full Collibra Evaluation Agreement document, follow the link below.

Collibra Click-Through Evaluation Agreement

BY ACCEPTING THESE TERMS OR OTHERWISE USING OR ACCESSING THE EVALUATION OFFERINGS, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS AND CONDITIONS IN THIS COLLIBRA EVALUATION AGREEMENT ("EVALUATION AGREEMENT"). COLLIBRA GRANTS YOU ACCESS TO AND USE OF THE EVALUATION OFFERINGS ONLY IF YOU ACCEPT THE TERMS AND CONDITIONS OF THIS EVALUATION AGREEMENT. IF YOU ARE ENTERING INTO THIS EVALUATION AGREEMENT ON BEHALF OF A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT THAT YOU HAVE THE AUTHORITY TO BIND SUCH ENTITY TO THIS EVALUATION AGREEMENT, IN WHICH CASE THE TERMS "YOU," OR "YOUR" SHALL REFER TO SUCH ENTITY. IF YOU DO NOT HAVE SUCH AUTHORITY, OR IF YOU DO NOT AGREE WITH THESE TERMS AND CONDITIONS, YOU MAY NOT ACCESS OR USE THE EVALUATION OFFERING.

This Evaluation Agreement is entered into between you (the individual and any entity for which you are acting, and your permitted successors and assigns) and Collibra, meaning Collibra Inc., if you are located in the United States, Mexico or Canada; or Collibra UK Limited, if you are located in a country other than the United States, Mexico or Canada (in each case, referred to herein as "**Collibra**").

1. Grant and Use Rights for Evaluation

1. **Evaluation Grant.** Subject to your compliance with this Evaluation Agreement, Collibra grants you, at no cost, a non-exclusive, non-transferable, non-sublicensable, temporary and limited (i) right to access and use Collibra's software-as-a-service cloud offering including any relevant accompanying software

components and documentation ("**Service**") and/or (ii) license to install, copy and use the object code form of Collibra's proprietary installed software product including any relevant accompanying documentation ("**Software**"), as applicable to your evaluation ("**Evaluation Offering**"), solely on an evaluation basis for non-production, internal, test and demonstration purposes (the "**Purpose**") during the Evaluation Term. There is no commitment to purchase the Evaluation Offering at the end of the Evaluation Term. Collibra may provision the Evaluation Offering with the assistance of its affiliates and suppliers, and during the Evaluation Term, Collibra may update the Evaluation Offering and/or limit certain features and functionality of the Evaluation Offering at its discretion.

2. **Evaluation Term.** You may use the Evaluation Offering subject to this Evaluation Agreement only for a period of twenty (20) days ("**Evaluation Term**"). Any extension of the Evaluation Term must be agreed to by both parties, in writing.

2. Restrictions

You will not (and will not permit anyone else to) do any of the following: (a) use the Evaluation Offering for anything other than the Purpose; (b) use the Evaluation Offering under this Evaluation Agreement to avoid incurring fees or exceeding any limitations agreed to in an order form entered into by and between you and Collibra; (c) provide access to, distribute or sublicense the Evaluation Offering to a third-party, (d) use the Evaluation Offering on behalf of, or to provide any product or service to, third parties, (e) use or reference the Evaluation Offering to develop a similar or competing product or service, (f) reverse engineer, decompile, disassemble, or seek to access the source code, algorithms, programming interfaces, or non-public APIs to the Evaluation Offering, except to the extent expressly permitted by applicable law (and then only with prior notice to Collibra), (g) circumvent any established usage limits, including restrictions on number of authorized users, whether through the use of APIs or other means (h) modify or create derivative works of the Evaluation Offering, or copy any element of the Service (other than authorized copies of the software components), (i) remove or obscure any product identification or proprietary notices in the Evaluation Offering, (j) publish benchmarks or performance information about the Evaluation Offering, (k) interfere with the Service's operation, circumvent its access restrictions or conduct any security or vulnerability test of the Service, or (l) transmit any viruses or other harmful materials to the Service.

3. Ownership

You will retain all right, title and interest to, and all intellectual property rights in your Content (as defined below) sent to the Evaluation Offering. Except for the limited right to use the Evaluation Offering under this Evaluation Agreement, this Evaluation Agreement does not give you any title, interest or rights, including intellectual property rights, in any component of the Evaluation Offering. The Evaluation Offering and all copyright, patent, and other proprietary rights therein are and shall remain the sole property of Collibra. This includes any information that Collibra collects and analyzes in connection with the Evaluation Offering, such as Usage Data (defined below), user feedback and other information to improve and evolve Collibra's products and services offerings.

4. Your Content

You are solely responsible for any and all applications, files, information, materials, data, software, or other content uploaded to, stored, published or displayed by you through the Service ("**Content**"). You acknowledge that the Service is provisioned for the Purpose only, and therefore you will not use any production or regulated data (including personal information) as part of your Content in the use of the Service. You are responsible for protecting the security of your Content, including any access you might provide to your Content by your employees, customers or other third parties. You will take and maintain appropriate security, protection and backup for your Content. You are responsible for complying with any laws or regulations applicable to your Content. You are solely responsible for any consequences if your Content is inadvertently exposed or lost, whether or not you have encrypted, backed up or otherwise taken steps required by the relevant laws or regulations to protect your Content. To the extent Collibra process Personal Data, as defined in the DPA, each party agrees to comply with the terms of the data processing addendum, which can be found here: www.collibra.com/data-processing-addendum ("**DPA**"), and which is hereby incorporated by reference.

5. Usage Data

Collibra collects information related to your use of the Services so that Collibra can analyze usage, support the Service, and maintain and improve its products and services, and for other lawful business purposes ("**Usage Data**"). Collibra may also use and store

this Usage Data in an aggregated, anonymized form for Collibra's own purposes. You hereby expressly agree that Collibra may collect Usage Data during the Evaluation Term.

6. Support Services

Collibra does not provide support services or service level commitments of any kind in connection with the Evaluation Offering.

7. Term and Termination; Suspension

1. **Term.** This Evaluation Agreement and your use to the Evaluation Offering will commence when you click to accept it as part of the sign-up process or, if earlier, when you use any of the Evaluation Offering (the "**Effective Date**"), and will be effective through the Evaluation Term, unless terminated earlier as permitted under this Section 7.
2. **Termination for Convenience.** This Evaluation Agreement may be terminated effective immediately by Collibra or you for any or no reason and at any time by providing notice of termination to the other party.
3. **Suspension.** Collibra may temporarily suspend your use of the Evaluation Offering if Collibra determines in its sole discretion: (i) you or your use of the Evaluation Offering is in breach of this Evaluation Agreement; (ii) Your use of the Evaluation Offering poses a security risk to the Evaluation Offering, or interferes with, disrupts, damages, or accesses in an unauthorized manner the servers, networks, or other properties or services of any other party; or (iii) suspension is required pursuant to Collibra's receipt of a subpoena, court order or other request by a law enforcement agency.
4. **Effect of Termination or Suspension.** Upon the termination or suspension of this Evaluation Agreement for any reason: (i) all rights and licenses granted to you under this Evaluation Agreement, including your ability to access any of your Content, will immediately terminate; (ii) you must promptly discontinue all access or use of the Evaluation Offering and delete or destroy any of Collibra's Confidential Information; and (iii) Collibra will have no obligation to retain your Content or configurations generated during the Evaluation Term by your use of the Evaluation Offering.

8. Disclaimer of Warranty

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, COLLIBRA PROVIDES THE EVALUATION OFFERING AS-IS WITHOUT ANY WARRANTIES OF ANY KIND, EXPRESS, IMPLIED, STATUTORY, OR IN ANY OTHER PROVISION OF THIS EVALUATION AGREEMENT OR COMMUNICATION WITH YOU, AND COLLIBRA SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, TITLE, AND ANY WARRANTIES ARISING FROM THE COURSE OF DEALING OR COURSE OF PERFORMANCE REGARDING OR RELATING TO THE EVALUATION OFFERING, THE DOCUMENTATION, OR ANY MATERIALS FURNISHED OR PROVIDED TO YOU UNDER THIS EVALUATION AGREEMENT. COLLIBRA DOES NOT WARRANT THAT THE EVALUATION OFFERING WILL OPERATE UNINTERRUPTED, OR THAT IT WILL BE FREE FROM DEFECTS OR THAT THE EVALUATION OFFERING WILL MEET (OR IS DESIGNED TO MEET) YOUR REQUIREMENTS.

9. Limitation of Liability

TO THE MAXIMUM EXTENT MANDATED BY LAW, IN NO EVENT SHALL COLLIBRA BE LIABLE FOR ANY LOST PROFITS OR BUSINESS OPPORTUNITIES, LOSS OF USE, LOSS OF REVENUE, LOSS OF GOODWILL, BUSINESS INTERRUPTION, LOSS OF DATA, INABILITY TO USE THE EVALUATION OFFERING, INCLUDING AS A RESULT OF ANY TERMINATION, DISCONTINUATION, MODIFICATION OR SUSPENSION OF THIS EVALUATION AGREEMENT OR THE EVALUATION OFFERING, OR ANY OTHER INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES UNDER ANY THEORY OF LIABILITY, WHETHER BASED IN CONTRACT, TORT, NEGLIGENCE, PRODUCT LIABILITY OR OTHERWISE. COLLIBRA'S LIABILITY UNDER THIS EVALUATION AGREEMENT SHALL NOT, IN ANY EVENT, EXCEED USD $100. THE FOREGOING LIMITATIONS SHALL APPLY REGARDLESS OF WHETHER COLLIBRA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND REGARDLESS OF WHETHER ANY REMEDY FAILS OF ITS ESSENTIAL PURPOSE. YOU MAY NOT BRING A CLAIM UNDER THIS EVALUATION AGREEMENT MORE THAN EIGHTEEN (18) MONTHS AFTER (i) THE END OF THE

EVALUATION TERM, OR (ii) THE DATE THE CLAIM FIRST ARISES, WHICHEVER IS EARLIEST.

10. Confidentiality

1. **Confidential Information**. Each party acknowledges that it or its employees may, in the course of performing its responsibilities under this Evaluation Agreement, be exposed to or acquire information which is proprietary or confidential to the other party. "**Confidential Information**" means information disclosed under this Evaluation Agreement that is designated by the disclosing party as proprietary or confidential or that should be reasonably understood to be proprietary or confidential due to its nature and the circumstances of its disclosure. Collibra's Confidential Information includes any technical or performance information about the Evaluation Offering. Confidential Information excludes information that the receiving party can document (a) is or becomes public knowledge through no fault of the receiving party, (b) it rightfully knew or possessed prior to receipt under this Evaluation Agreement, (c) it rightfully received from a third-party without breach of confidentiality obligations or (d) it independently developed without using the disclosing party's Confidential Information. As receiving party, each party will (a) hold in confidence and not disclose Confidential Information to third parties except as permitted in this Evaluation Agreement, and (b) only use Confidential Information to fulfill its obligations and exercise its rights in this Evaluation Agreement. Each party will use reasonable care to protect the Confidential Information in the same manner as it would protect its own Confidential Information of a similar nature, but in no event with less than reasonable care. The receiving party may disclose Confidential Information to its employees, agents, contractors and other representatives having a legitimate need to know, provided it remains responsible for their compliance with this Section 10 and they are bound to confidentiality obligations no less protective than this Section 10. Unauthorized use or disclosure of Confidential Information may cause substantial harm for which damages alone are an insufficient remedy. Each party may seek appropriate equitable relief, in addition to other available remedies, for breach or threatened breach of this Section 10.
2. **Required Disclosures**. Nothing in this Evaluation Agreement prohibits either party from making disclosures, including Content and other Confidential Information, if required by applicable law, subpoena or court order, provided (if permitted by

applicable law) it notifies the other party in advance and reasonably cooperates in any effort to obtain confidential treatment at disclosing party's expense.

11. General

1. **Governing Law, Jurisdiction and Venue**. This Evaluation Agreement is governed by the laws of the State of New York without regard to conflicts of laws provisions and without regard to the United Nations Convention on the International Sale of Goods, and the jurisdiction and venue for actions related to this Evaluation Agreement will be the state and United States federal courts located in New York, New York, and both parties submit to the personal jurisdiction of those courts.
2. **Compliance with Laws**. You will comply with all applicable laws and regulations in the use of the Evaluation Offering under this Evaluation Agreement.
3. **Assignment**. You may not assign this Evaluation Agreement without Collibra's prior written consent, and any such action in violation of this provision, is null and void, and a breach of this Evaluation Agreement. Collibra may assign or transfer this Evaluation Agreement to an affiliate or in connection with a merger, reorganization, acquisition or other transfer of all or substantially all its assets or voting securities. This Evaluation Agreement will bind and inure to the benefit of the party's permitted successors and assigns.
4. **Independent Contractors**. The parties are independent contractors, not agents, partners or joint venturers.
5. **Third-Party Beneficiaries**. Except as expressly provided in this Evaluation Agreement, this Evaluation Agreement does not create or establish any rights or beneficiaries for any person or entity that is not a party to this Evaluation Agreement.
6. **Export**. You acknowledge that the Evaluation Offering is subject to export control and economic sanctions restrictions imposed by the U.S. government and import restrictions by certain foreign governments (collectively "**Trade Laws**"). In using or accessing the Evaluation Offering, you will not and will not allow any third party to use the Evaluation Offering in violation if any Trade Laws or remove or export from the U.S. or allow the export or re-export of any part of the Evaluation Offering or any direct product thereof to any location, party or end-use which the U.S. government or

any agency thereof requires an export license or other governmental approval at the time of export or re-export without first obtaining such license or approval. You represent and warrant that you and any of your users: (i) are not listed on any U.S. government list of prohibited or restricted parties, including the U.S. Treasury Department list of Specially Designated Nationals and Blocked Persons, or the U.S. Commerce Department Denied Persons List or Entity List; (ii) are not an entity or person who is organized under the laws of, ordinarily resident in, or controlled by the government of, any country or region (1) that is subject to a U.S. government embargo or comprehensive sanction, (2) to which the U.S. has prohibited export transactions or (3) that has been designated by the U.S. government as a "terrorist supporting" country; (iii) will not use the Evaluation Offering for the manufacture, design or development of nuclear, chemical or biological weapons or missile technology, or for terrorist activity; and (iv) will not submit to the Evaluation Offering any information controlled under the U.S. International Traffic in Arms Regulations or listed on the Commerce Control List unless approved in writing by Collibra. You will notify Collibra promptly if you or your users become subject to any order or restriction listed in this Section.

7. **Government End-Users**. Elements of the Evaluation Offering are commercial computer software. If the user or licensee of the Evaluation Offering is an agency, department, or other entity of the United States Government, the use, duplication, reproduction, release, modification, disclosure, or transfer of the Evaluation Offering or any related documentation of any kind, including technical data and manuals, is restricted by the terms of this Evaluation Agreement in accordance with Federal Acquisition Regulation 12.212 for civilian purposes and Defense Federal Acquisition Regulation Supplement 227.7202 for military purposes. The Evaluation Offering was developed fully at private expense. All other use is prohibited.

8. **Amendments**. Except as otherwise provided herein, any amendments, modifications or supplements to this Evaluation Agreement must be in writing and signed by each party's authorized representatives or, as appropriate, agreed through electronic means provided by Collibra.

9. **Waivers and Severability**. Waivers must be signed by the waiving party's authorized representative and cannot be implied from conduct. If any provision of this Evaluation Agreement is held invalid, illegal or unenforceable, it will be limited to the

minimum extent necessary, so the rest of this Evaluation Agreement remains in effect.

10. **Entire Agreement**. This Evaluation Agreement is the parties' entire agreement regarding its subject matter and supersedes any prior or contemporaneous agreements or communications regarding its subject matter, whether written or oral. Notwithstanding the foregoing, to the extent a separate written agreement has been signed between you and Collibra in connection with the subject matter hereof, such separate written agreement will supersede this Evaluation Agreement. In this Evaluation Agreement, headings are for convenience only and "including" and similar terms are to be construed without limitation.