PIOTR CZARNAS & ANDRZEJ NAGALSKI, PhD

A STEP-BY-STEP GUIDE TO IMPROVE DATA QUALITY

HOW TO REACH A 100% DATA QUALITY SCORE

DQOps is an open-source platform for data quality that covers the entire data lifecycle, from profiling new data sources to fully automating data quality monitoring.

The approach to **managing data quality** changes throughout the data lifecycle.
The preferred interface for the data quality platform also changes: user interface, Python code, REST API, command line, editing YAML files, running locally, or configuring a shared server. DQOps supports all of these options.

Learn more about DQOps at www.dqops.com

EVALUATING NEW DATA SOURCES

Data scientists and **data analysts** want to review the data quality of new data sources or understand the data present on the data lake by profiling data.

CREATING DATA PIPELINES

The **data engineering teams** want to verify data quality checks required by the Data Contract on both source and transformed data.

TESTING DATA

An organization has a dedicated data quality team that handles quality assurance for data platforms. **Data quality engineers** want to evaluate all sorts of data quality checks.

OPERATIONS

The data platform matures and transitions to a production stage. The **data operations team** watches for schema changes and data changes that make the data unusable.

Table of Contents

Introduction

Data quality assurance refers to the process of ensuring that data meets the expectations of those who will use it. The field of data quality has adopted the word “dimension” to identify the aspects of data that can be measured and through which its quality can be quantified. While different experts have proposed different data quality dimensions, almost all of them include some version of accuracy, validity, completeness, consistency, currency, or timeliness.

Data quality can mean different things to different companies, and the number and types of measures used to assess data quality may vary depending on how a company intends to use its data. The outcomes of data quality measurements are known as data quality key performance indicators (KPIs). These scores are used to evaluate the health and quality of data sources and ensure compliance with data quality requirements, which are typically agreed upon in a data contract between a data producer and a data consumer.

Data quality KPIs can be classified into two categories: business-focused and data engineering-focused. Business-focused KPIs help to monitor data quality and ensure data trustworthiness, whereas data engineering-focused KPIs monitor data pipeline issues, allowing data engineering teams to improve data platforms.

Data quality monitoring and improvement involve different stakeholders, including the Data Owner, Data Producer, Data Engineering Team, and Data Quality Team.

The Data Owner has a deep understanding of the data’s purpose, the data model, and the business processes in their area of responsibility. Additionally, they have expertise in the line-of-business applications that utilize the monitored data. This knowledge empowers them to analyze and define business needs for data quality. They identify tables requiring data quality checks and establish acceptable metrics and thresholds for these checks. Furthermore, they collaborate with the Data Quality Team to investigate potential data quality issues and identify root causes.

The Data Producer is the owner of an external platform that is the source of data imported into the monitored data warehouse or data lake. The Data Producer may also be an external vendor involved in a data-sharing agreement.

The Data Engineering Team collects, manages, and converts raw data into usable information for data scientists and business analysts. This team also builds and maintains data pipelines and databases. Their responsibilities include designing, building, testing, and maintaining data management systems essential for high-quality data.

The Data Quality Team plays a critical role in helping organizations achieve their business goals by identifying poor data quality, which is the root cause of operational failures. They are responsible for importing metadata from monitored databases into the data quality platform, configuring data quality checks tailored to specific needs, and monitoring for data quality issues. When issues arise, the Data Quality Team facilitates the data cleaning process by engaging all stakeholders.

This eBook guides how to improve data quality by setting up a data quality monitoring process. The process involves tracking data quality KPIs to identify and prioritize data quality issues that require immediate attention. Data quality KPIs and a list of tables affected by the data quality issues should be presented on data quality dashboards. Once data quality monitoring is set up, improving data quality becomes an iterative process that involves identifying issues by the Data Quality Team, contacting the Data Owner and/or Data Engineering Team to resolve the issues, and then re-executing data quality monitoring by the Data Quality Team.

The eBook explains the data quality improvement process through a flowchart, which illustrates the sequence of activities involved in making the process work. Each step and decision in the process is represented by a shape. The rectangles represent actions and have unique abbreviations and numbers to help readers locate a detailed description in the eBook. The colors indicate which stakeholder is responsible for a particular action. The Data Owner (DO) is marked in blue, the Data Engineering Team (DE) in green, and the Data Quality Team (DQ) in orange.

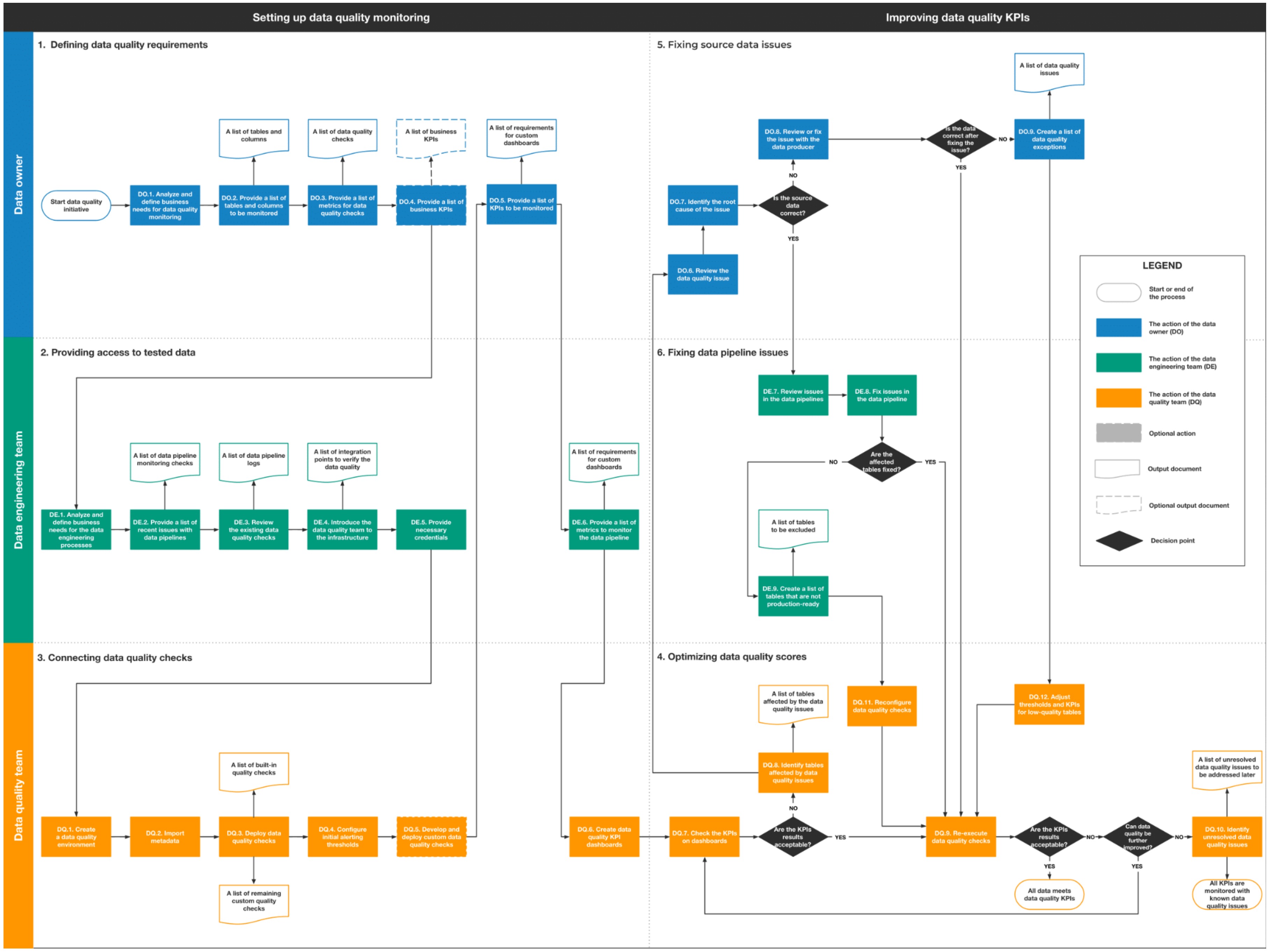
To simplify the process of monitoring data quality and identifying areas for improvement before quality issues affect the reliability of analytical processes, we have created the DQOps data quality platform. The platform provides an efficient user interface that makes it easy to add data sources, configure data quality checks, and manage issues. DQOps comes with over 150 built-in data quality checks, but users can also design custom checks to detect any business-relevant data quality issues. The platform supports incremental data quality monitoring to allow the analysis of data quality for large tables. Users can track data quality KPI scores using built-in or custom dashboards to show progress in improving data quality to business sponsors. DQOps is DevOps-friendly, allowing users to define data quality definitions in YAML files stored in Git, run data quality checks directly from their data pipelines, or automate any action with a Python Client. DQOps can be used locally or as a SaaS platform.

In DQOps, a data quality check is a combination of the data quality sensor and data quality rule. The data quality sensors capture measures such as the number of rows, the percentage of null values in a column, or the current delay between the timestamp of the latest row and the current system time. The sensors can be implemented as templated SQL queries (DQOps uses the Jinja2 templating engine) or as custom code that can call the appropriate source system's APIs. The executed data quality check has two possible statuses: passed or failed. The status of the data quality check is verified by a data quality rule that compares the sensor readout with the minimum acceptance threshold. Data quality issues are identified as failed data quality checks rejected by the data quality rule. DQOps calculates data quality KPIs as a percentage of passed data quality checks for each table, database, or connection.

Glossary

Data Engineering Team (DE)	The team that collects, manages, and converts raw data, builds and maintains data pipelines, and maintains databases. The Data Engineers are responsible for designing, building, testing, and maintaining data management systems.
Data Owner (DO)	A person who understands the purpose of the data, the data model, and the business processes in their area of responsibility; analyzes and defines business needs, indicates which tables should be checked for data quality, and sets thresholds for alerts.
Data Quality Team (DQ)	The team imports metadata from monitored databases into the data quality platform, configures data quality checks, and monitors data quality issues.
Data Producer	The owner of an external platform supplying data to be imported into the monitored data warehouse or data lake.
Data quality dashboard	A display of data quality KPIs separated by business areas, organizational units, geographical locations, suppliers, business partners, etc.
Data quality KPI	The results of data quality measurements. DQOps calculates data quality KPIs as a percentage of passed data quality checks for each table, database, or connection.
Data grouping	A data group is a group of rows loaded from a single or different source and aggregated into one table. Data groups are used to calculate separate data quality KPI scores for different groups of rows, such as individual data quality KPIs for different vendors, platforms, departments, or countries.
Rule	A set of conditions against which sensor readouts are verified, described by a list of thresholds.
Check	A test for data quality is a combination of a data quality sensor and a data quality rule. Checks are defined as YAML files.
Sensor	A template SQL query that captures metrics such as the number of rows, the percentage of null values in a column, or the current delay between the timestamp of the latest row and the current system time.
Threshold	The established metric's value, past which an alert of a given type shall be raised.

Data quality monitoring process

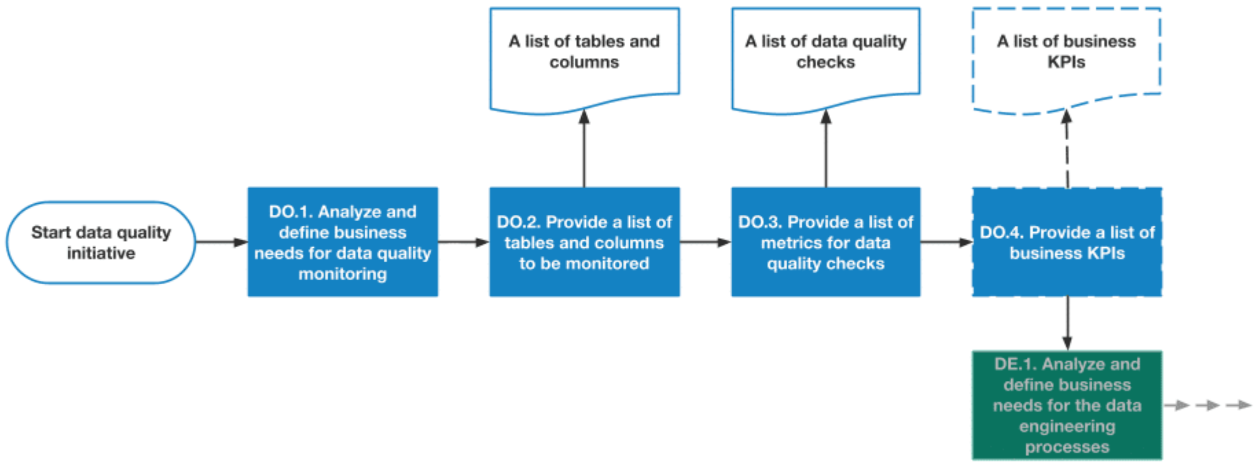


Setting up data quality monitoring
Improving data quality KPIs
Data owner
Data engineering team
Data quality team
1. Defining data quality requirements
Start data quality initiative
DO.1. Analyze and define business needs for data quality monitoring
DO.2. Provide a list of tables and columns to be monitored
A list of tables and columns
DO.3. Provide a list of metrics for data quality checks
A list of data quality checks
DO.4. Provide a list of business KPIs
A list of business KPIs
DO.5. Provide a list of KPIs to be monitored
A list of requirements for custom dashboards
2. Providing access to tested data
DE.1. Analyze and define business needs for the data engineering processes
DE.2. Provide a list of recent issues with data pipelines
A list of data pipeline monitoring checks
DE.3. Review the existing data quality checks
A list of data pipeline logs
DE.4. Introduce the data quality team to the infrastructure
A list of integration points to verify the data quality
DE.5. Provide necessary credentials
DE.6. Provide a list of metrics to monitor the data pipeline
A list of requirements for custom dashboards
3. Connecting data quality checks
DQ.1. Create a data quality environment
DQ.2. Import metadata
DQ.3. Deploy data quality checks
A list of built-in quality checks
A list of remaining custom quality checks
DQ.4. Configure initial alerting thresholds
DQ.5. Develop and deploy custom data quality checks
DQ.6. Create data quality KPI dashboards
5. Fixing source data issues
DO.6. Review the data quality issue
DO.7. Identify the root cause of the issue
Is the source data correct?
NO
YES
DO.8. Review or fix the issue with the data producer
Is the data correct after fixing the issue?
NO
YES
DO.9. Create a list of data quality exceptions
A list of data quality issues
6. Fixing data pipeline issues
DE.7. Review issues in the data pipelines
DE.8. Fix issues in the data pipeline
Are the affected tables fixed?
NO
YES
DE.9. Create a list of tables that are not production-ready
A list of tables to be excluded
4. Optimizing data quality scores
DQ.7. Check the KPIs on dashboards
Are the KPIs results acceptable?
NO
YES
DQ.8. Identify tables affected by data quality issues
A list of tables affected by the data quality issues
DQ.11. Reconfigure data quality checks
DQ.12. Adjust thresholds and KPIs for low-quality tables
DQ.9. Re-execute data quality checks
Are the KPIs results acceptable?
YES
NO
All data meets data quality KPIs
Can data quality be further improved?
YES
NO
DQ.10. Identify unresolved data quality issues
A list of unresolved data quality issues to be addressed later
All KPIs are monitored with known data quality issues
LEGEND
Start or end of the process
The action of the data owner (DO)
The action of the data engineering team (DE)
The action of the data quality team (DQ)
Optional action
Output document
Optional output document
Decision point

I. Setting up data quality monitoring

Setting up data quality monitoring can be divided into three parts. First, the Data Owner needs to define data quality requirements, translated into a list of tables and columns to be monitored, a list of data quality checks, and, optionally, a list of business-focused data quality KPIs. Next, these requirements are transferred to the Data Engineering Team, which reviews them regarding the data engineering process. The Data Engineering Team generates a list of pipeline monitoring checks, introduces the Data Quality Team to the infrastructure, and provides the necessary credentials. In the third part, based on requirements from the Data Owner and the Data Engineering Team, the Data Quality Team installs the environment, imports the metadata, and creates data quality checks. Finally, the Data Quality Team assigns the initial data quality monitoring KPIs thresholds and creates the first data quality monitoring dashboards.

1. Define data quality requirements from the business perspective



A list of tables and columns
A list of data quality checks
A list of business KPIs
Start data quality initiative
DO.1. Analyze and define business needs for data quality monitoring
DO.2. Provide a list of tables and columns to be monitored
DO.3. Provide a list of metrics for data quality checks
DO.4. Provide a list of business KPIs
DE.1. Analyze and define business needs for the data engineering processes

The Data Owner analyzes and defines business goals and needs for data quality monitoring. The next steps provide a list of tables and columns that the Data Owner wants to monitor. A list of metrics used to create data quality checks is also defined. Finally, the Data Owner, with the assistance of the Data Quality Team, defines data quality KPIs relevant from a business perspective.

DO.1. Analyze and define business needs for data quality monitoring

The Data Owner identifies current goals and scope regarding data quality monitoring. This can be a specific issue that the Data Owner is trying to fix or a strategic initiative for preventing the deterioration of data quality in the whole company.

The Data Owner also needs to identify the data elements that are critical or required for a specific business process that needs to be monitored. This data is typically referred to as critical data elements (CDEs). The Data Owner should also define the expectations of data consumers regarding the condition of the data to ensure its suitability for particular purposes.

At this stage, the Data Owner presents business needs for data quality monitoring to the Data Quality Team to help them better understand the context. This will help the Data Quality Team define monitoring methods at the later stages.

The Data Owner should take the following steps before proceeding with the data quality initiative.

- **Set business goals and scope.** To clearly understand the business owner's needs, it is important to discuss and set business goals for data quality. This will make it easier to develop metrics to measure its quality.
- **Identify CDEs for monitoring.** Prepare a list of the critical data elements and the most common problems that lead to data quality issues. This will help qualitatively assess the scope and area of the problems that need to be targeted.
- **Set data quality priorities**. Clarify what is crucial for the Data Owner to test. The Data Owner needs to collect all the necessary information about data quality measurement needs from data consumers.
- **Assess data quality dimensions.** Different companies need a different set of data quality categories. If the data must arrive on time and without delays, the company should prioritize timeliness. If it is more important that the data arrives in a specific format, the company should prioritize validity.

There is a need to review the list of previous data quality issues that the Data Owner would like to eliminate in the future. The following table shows typical data quality issues that could be detected.

Data quality dimension	Definition	Data quality issue
Accuracy	The degree of closeness of data values to real values, often measured by comparison with a known source of correct information.	Data cannot be used as a reliable source of information and impacts the organization's business intelligence, budgeting, forecasting, and other critical activities.
Completeness	The degree to which all required - records in the dataset, - data values are present with no missing information. The completeness does not measure accuracy or validity; it measures what information is missing.	Missing values/data or rows.
Consistency	The degree to which data values of two sets of attributes - within a record, - within a data file, - between data files, - within a record at different points in time comply with a rule. This dimension represents if the same information stored and used at multiple instances matches.	An inconsistent number of rows. Information stored in one place does not match relevant data stored elsewhere.
Reasonableness	The degree to which a data pattern meets expectations. Reasonableness measures the degree to which data values have a reasonable or understandable data type and size.	The values in the database are not reasonable
Timeliness	The degree to which the period between the time of creation of the real value and the time that the dataset is available is appropriate.	Data is not up-to-date

Data quality dimension	Definition	Data quality issue
Uniqueness	The degree to which records occur only once in a data file and are not duplicated.	The same data is stored in multiple locations.
Validity	The degree to which data values comply with pre-defined business rules such as the format, type, and range. E.g., zip codes. e-mails	Invalid data format.

The Data Owner does not always have full control over the data. Data that is shared with external partners and vendors should meet data-sharing KPIs. The Data Owner may define the data quality KPIs that will be monitored.

DO.2. Provide a list of tables and columns to be monitored

Now that the CDEs (critical data elements) have been identified, before the Data Owner can measure the data quality, there is a need to know how the data is structured in the data store. At this stage, the Data Owner checks the frequency of data updates, data size, format, patterns, completeness, etc. The Data Owner also carefully checks the number of tables, columns, and their names. Before implementing data quality monitoring using DQOps, the data must be uniform in these aspects.

To complete this stage, make sure the following steps have been taken.

- **Identify monitored databases, data warehouses, and data lakes.** The Data Owner should decide which data platforms will be included in the monitoring and improvement of the data quality. DQOps platform supports the most commonly used data sources.
- **Decide which stages should be monitored.** Tables have different purposes along the data lineage. Monitoring the quality of the ingestion stage measures the quality of the source data and the reliability of the data ingestion process. Monitoring can also measure the data sharing KPI for an external Data Producer (a business partner). Measuring the quality of the reporting layer and data mart ensures that the dashboards show the correct numbers. Depending on the goal of the data quality project, all or only selected stages may be monitored.
- **Identify the file formats.** If the ingestion stages, and therefore the quality of flat files, are monitored, the Data Owner should identify the expected file formats, such as CSV,

Parquet, ORC, or JSON. The DQOps platform allows data quality monitoring of CSV, Parquet, and JSON files.

- **Provide a list of tables to be monitored.** The Data Owner should gather a list of important tables affected by data quality issues.
- **Prioritize tables based on their importance.** A data quality improvement project can provide a quick return on investment when the most important tables are cleaned up first. The next steps in the project can be implemented in sprints. Groups of tables with the same priority can also be assigned a project milestone. DQOps supports assigning priorities to tables.
- **Identify large tables.** Large tables stored on Big data platforms require special attention and careful planning. Executing data quality checks on these tables can be time-consuming or can affect regular workload, reducing the responsiveness of the data platform or even slowing down regular data loading jobs.
- **Identify date-partitioned tables.** Tables that are physically partitioned by date or date and time are candidates for date-partitioned data quality checks. DQOps can monitor the data quality of each daily partition separately. Data quality readouts (such as a row count) and data quality alerts can be evaluated for each daily partition separately and can be associated with the date of the partition. Daily-partitioned tables include ingestion tables, fact tables, clickstreams tables, and transaction tables.
- **Identify append-only tables with an event timestamp column.** Date-partitioned data quality checks in DQOps are not limited to tables physically stored using date partitioning. Tables that do not change often can also be analyzed for data quality issues in a daily time gradient. DQOps treats them as date-partitioned tables and calculates separate data quality scores for each "day of data" separately.
- **Provide the list of columns to be monitored.** Specify all columns that should be monitored and identify the expected data formats. Examples of columns that should be monitored for data quality include identifiers (should not be null), measures (should be within valid ranges), columns that store a value from a dictionary (such as country codes), and all columns with known formats (such as email). When tables store all data in a text column (varchar, string), the expected data type should be specified. DQOps simplifies the identification of columns that should be monitored by providing basic statistics.
- **Add comments on known data quality issues.** The list of tables and columns should be extended with comments on the most severe and recurring data quality issues

related to these tables and columns. This information will help in selecting appropriate data quality checks.

- **Specify the frequency of updates.** The frequency of table changes and the configuration of the data pipeline scheduler are very important for the correct configuration of the data quality platform. Data quality checks should be scheduled to match the frequency of data loading operations on the monitored table.
- **Determine expected schema changes.** Configuring data quality monitoring on a table that is expected to be decommissioned or redesigned can be futile. DQOps data quality platform has several built-in data quality checks for detecting schema changes, as well as data quality checks for timeliness, which can detect that decommissioned tables are no longer fresh.

DQOps platform offers comprehensive support for various data sources (Figure 1.1.). It easily integrates with relational databases like MySQL, PostgreSQL, and Oracle, as well as cloud-based data warehouses like Amazon Redshift, Snowflake, or Google BigQuery. It also empowers you to manage data quality within popular big data processing frameworks like Spark and Databricks. Additionally, it integrates with SQL query engines like Trino and Athena, allowing you to seamlessly monitor data quality within your existing analytical workflows. DQOps extends its reach beyond traditional sources, allowing you to import and monitor flat files in various formats, including the widely used CSV, Parquet, and JSON files. This ensures that data residing outside your core databases, such as results from external APIs or data shared from partners, can be effectively evaluated for quality and incorporated into your overall data governance strategy.

The DQOps platform offers an overview of the basic statistical summaries (shown in Figure 1.2) of tables and columns. This summary includes important details such as the number of rows, data types, value ranges, and missing or unique values. It gives you a quick snapshot of your data's information, which is helpful in selecting columns for monitoring.

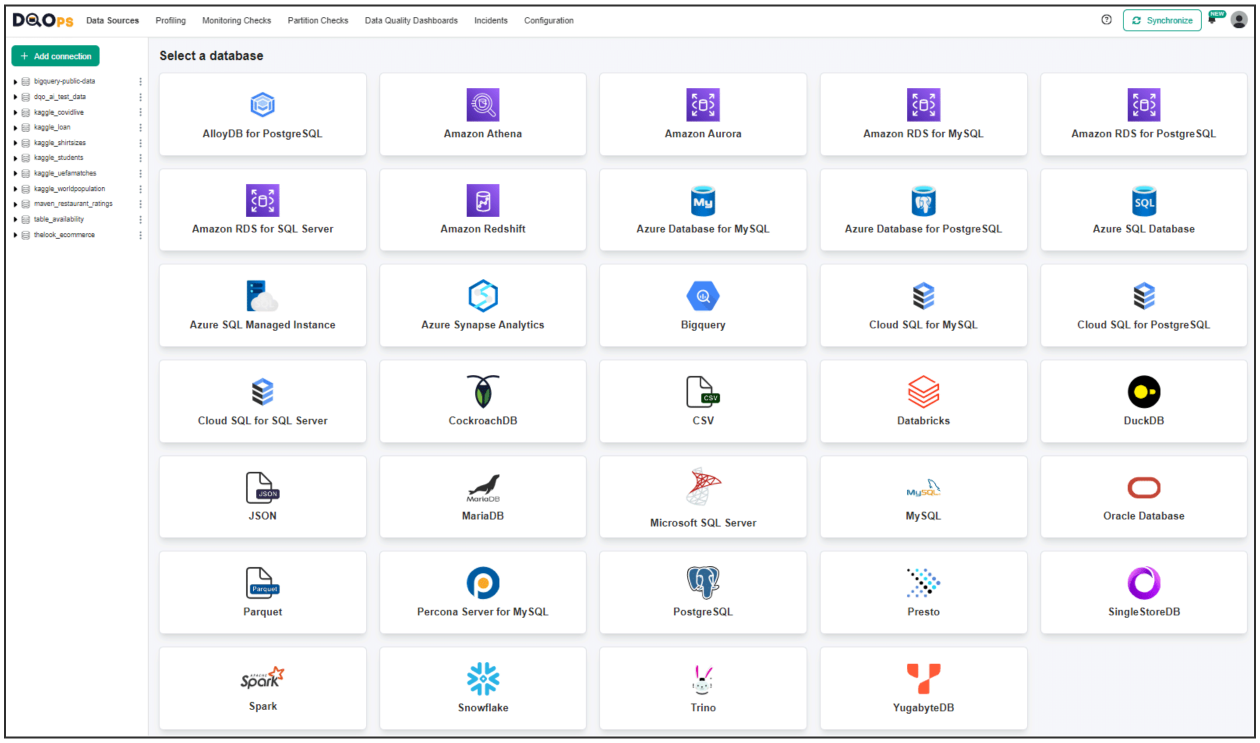
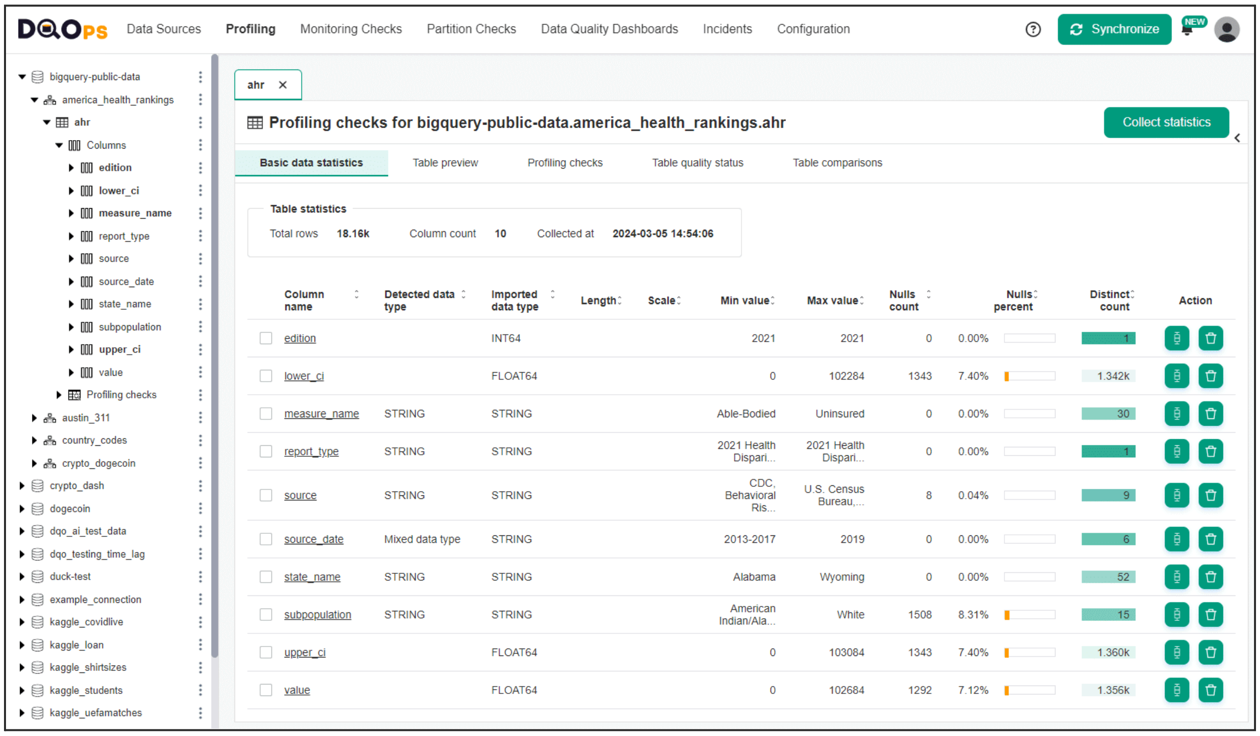


Figure 1.1. DQOps supports data quality monitoring in the most commonly used data sources.



Profiling checks for bigquery-public-data.america_health_rankings.ahr

Table statistics

Total rows 18.16k Column count 10 Collected at 2024-03-05 14:54:06

Column name	Detected data type	Imported data type	Length	Scale	Min value	Max value	Nulls count	Nulls percent	Distinct count	Action
edition		INT64			2021	2021	0	0.00%	1	
lower_ci		FLOAT64			0	102284	1343	7.40%	1.342k	
measure_name	STRING	STRING			Able-Bodied	Uninsured	0	0.00%	30	
report_type	STRING	STRING			2021 Health Dispari...	2021 Health Dispari...	0	0.00%	1	
source	STRING	STRING			CDC, Behavioral Ris...	U.S. Census Bureau,...	8	0.04%	9	
source_date	Mixed data type	STRING			2013-2017	2019	0	0.00%	6	
state_name	STRING	STRING			Alabama	Wyoming	0	0.00%	52	
subpopulation	STRING	STRING			American Indian/Ala...	White	1508	8.31%	15	
upper_ci		FLOAT64			0	103084	1343	7.40%	1.360k	
value		FLOAT64			0	102684	1292	7.12%	1.356k	

Figure 1.2. DQOps simplifies the selection of columns for monitoring by providing basic statistics.

At this stage of the process, there may be some problems worth noting.

- The absence of all tables and columns to be monitored will delay the analysis process and subsequent implementation.
- The average update time of the monitored tables must correspond to the frequency of periodic evaluation of data quality checks. Otherwise, data that changes every month could be monitored daily, raising 30x more alerts.

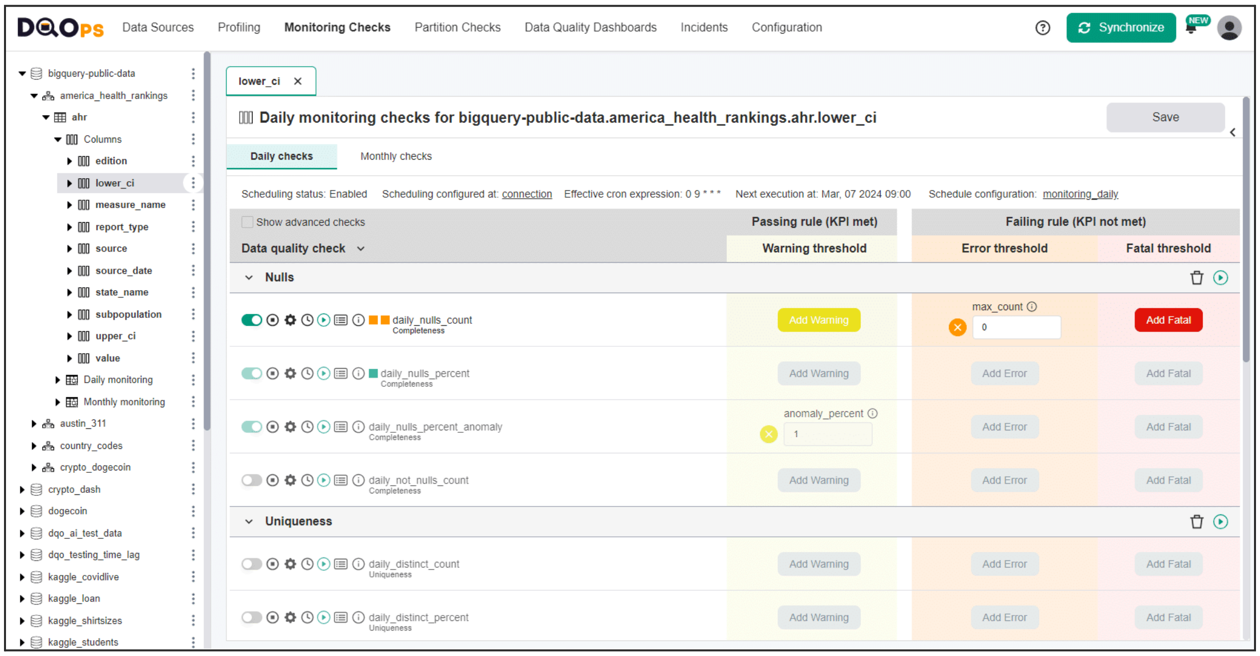
DO.3. Provide a list of metrics for data quality checks

After analyzing the data, the Data Owner, together with the Data Quality Team, defines a list of metrics that will be used to create data quality checks. The Data Owner should define specific metrics for each column selected for data quality monitoring. Each selected metric should relate to a data quality dimension, such as timeliness, completeness, validity, consistency, or uniqueness. This is an important step because a precise definition of metrics reduces the time for implementing and subsequently tweaking the data quality rules.

Make sure you complete the following steps before you move to the next stage.

- **Specify data quality expectations.** Based on the assumptions about the expected quality of the data, the Data Owner should determine how to measure the degree to which the data meets these expectations.
- **Prepare a list of metrics from the Data Owner's perspective.** This step should result in associating expectations with different measurements. After analyzing and examining the metrics structure, the Data Owner can create a list of metrics.
- **Discuss the list of metrics with the Data Quality Team.** After preparing a list of metrics, the Data Owner must discuss the list with the Data Quality Team. Metrics must be analyzed to make sure the Data Owner clearly understands their structure.
- **Finalize the list of metrics.** After a qualitative study of the metrics and the needs of the Data Owner, a final list of metrics can be created. It is important to add a detailed description of what and how the metrics should be measured. The metrics will be used to configure alerting thresholds in the data quality checks.
- **Aggregate required data quality checks.** The list of metrics to be measured must match the list of data quality checks available in the data quality platform of choice. DQOps platform has more than 150 built-in data quality checks and enables the design of custom data quality checks to meet any business-specific data quality requirements.

You can manage and run data quality checks with a user-friendly interface shown in Figure 1.3, but also Python code, REST API, and command line.



Daily monitoring checks for bigquery-public-data.america_health_rankings.ahr.lower_ci

Scheduling status: Enabled Scheduling configured at: connection Effective cron expression: 0 9 * * * Next execution at: Mar, 07 2024 09:00 Schedule configuration: monitoring_daily

	Passing rule (KPI met)	Failing rule (KPI not met)	
Data quality check	Warning threshold	Error threshold	Fatal threshold
Nulls			
daily_nulls_count (Completeness)	Add Warning	max_count 0	Add Fatal
daily_nulls_percent (Completeness)	Add Warning	Add Error	Add Fatal
daily_nulls_percent_anomaly (Completeness)	anomaly_percent 1	Add Error	Add Fatal
daily_not_nulls_count (Completeness)	Add Warning	Add Error	Add Fatal
Uniqueness			
daily_distinct_count (Uniqueness)	Add Warning	Add Error	Add Fatal
daily_distinct_percent (Uniqueness)	Add Warning	Add Error	Add Fatal

Figure 1.3. Sample screen from the DQOps platform showing check editor with activated daily_null_count check.

It is important to watch out for the following circumstances, which could hinder progress at this stage.

- Inapplicability of metrics selected for a given dataset
- Lack of understanding of the description of the metrics.
- Lack of insights about the data selected for the data quality process.
- Changes in the database schema during the requirements gathering process.

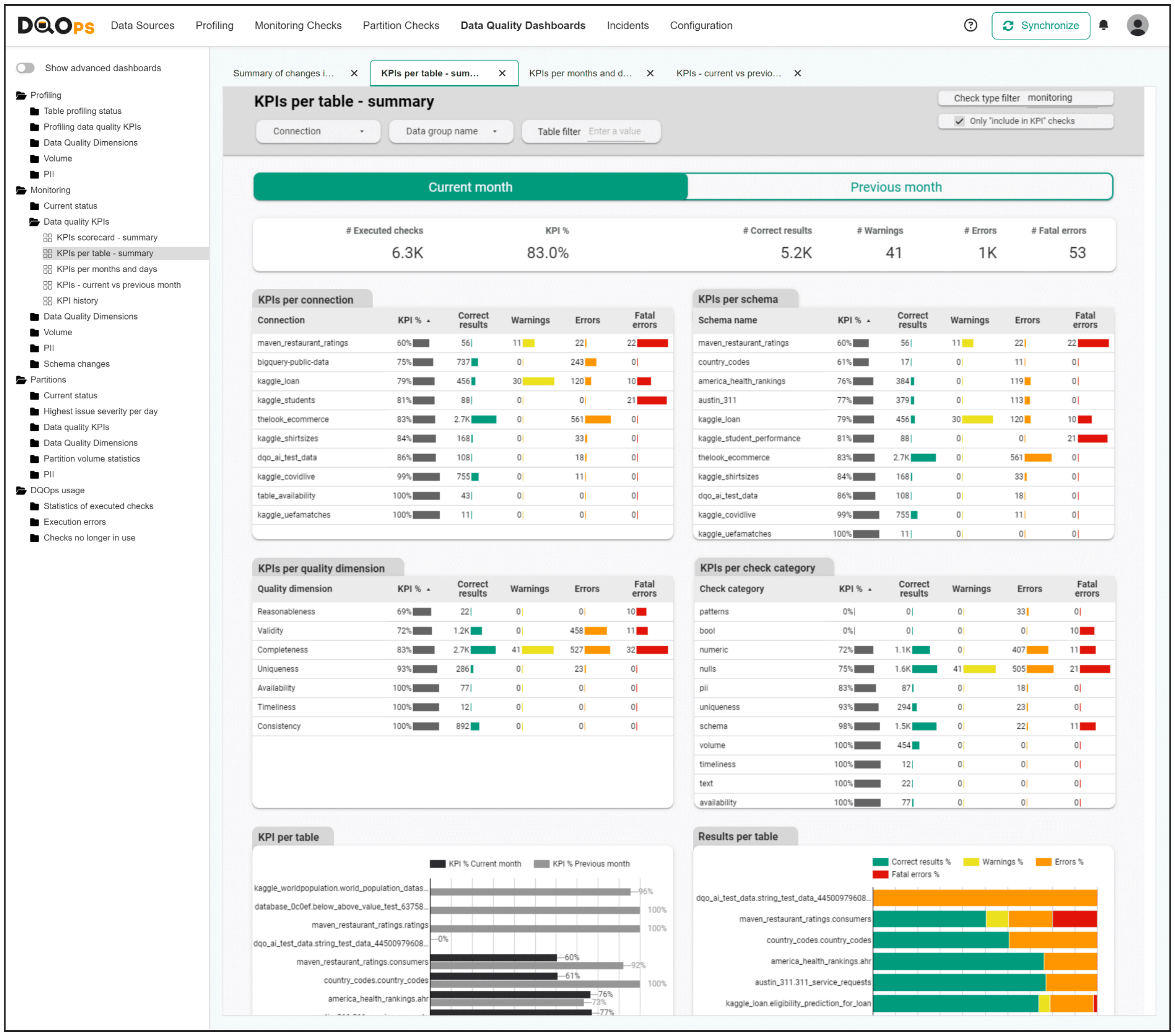
DO.4 (Optional) Provide a list of business KPIs

At this stage, the Data Owner presents the business KPIs that should be tested and the business reason behind this need. It is important to understand how the database structure aligns with the business process, as this will help identify differences and errors in the data that do not match reality. The Data Quality Team should assess the proposed business-specific KPIs to determine whether they can be monitored using a data quality tool. The data model used by the data quality tool to store the data quality check results must support proper reporting of those KPIs measured from the business perspective. It is also important to design business-specific dashboards to provide a clear view of the data quality. These dashboards should present KPIs separately for relevant categories such as business areas, organizational units, geographical locations, suppliers, and business partners. Learn more about dashboards in the DQ.6. Create data quality KPI dashboards chapter.

This stage consists of the following steps.

- **Demonstrate assumptions for KPIs.** Provide the data team with precise Data Owner's requirements for KPIs to be displayed on the data quality dashboards.
- **Review assumptions of KPIs.** The Data Quality Team analyzes data to match business KPI requirements with a list of possible data quality checks. This is a crucial step in the process as sometimes requirements cannot be implemented with available tools, or the volume of data makes it impossible to execute data quality checks in the current environment.
- **Create the final list of KPIs.** The Data Owner and the Data Quality Team work together to create an accurate list of KPIs and a detailed description of business metrics. The list should also specify the granularity of the KPIs.

The DQOps platform offers over 50 built-in dashboards, like the sample "KPIs per table - summary" dashboard shown in Figure 1.4. The platform also allows you to modify these dashboards or create entirely custom ones using Looker Studio. This flexibility ensures that your data quality insights are tailored to your needs and business context.



KPIs per table - summary

Check type filter: monitoring

✓ Only "include in KPI" checks

# Executed checks	KPI %	# Correct results	# Warnings	# Errors	# Fatal errors
6.3K	83.0%	5.2K	41	1K	53

KPIs per connection

Connection	KPI % ▲	Correct results	Warnings	Errors	Fatal errors
maven_restaurant_ratings	60%	56	11	22	22
bigquery-public-data	75%	737	0	243	0
kaggle_loan	79%	456	30	120	10
kaggle_students	81%	88	0	0	21
thelook_ecommerce	83%	2.7K	0	561	0
kaggle_shirtsizes	84%	168	0	33	0
dqo_ai_test_data	86%	108	0	18	0
kaggle_covidlive	99%	755	0	11	0
table_availability	100%	43	0	0	0
kaggle_uefamatches	100%	11	0	0	0

KPIs per schema

Schema name	KPI % ▲	Correct results	Warnings	Errors	Fatal errors
maven_restaurant_ratings	60%	56	11	22	22
country_codes	61%	17	0	11	0
america_health_rankings	76%	384	0	119	0
austin_311	77%	379	0	113	0
kaggle_loan	79%	456	30	120	10
kaggle_student_performance	81%	88	0	0	21
thelook_ecommerce	83%	2.7K	0	561	0
kaggle_shirtsizes	84%	168	0	33	0
dqo_ai_test_data	86%	108	0	18	0
kaggle_covidlive	99%	755	0	11	0
kaggle_uefamatches	100%	11	0	0	0

KPIs per quality dimension

Quality dimension	KPI % ▲	Correct results	Warnings	Errors	Fatal errors
Reasonableness	69%	22	0	0	10
Validity	72%	1.2K	0	458	11
Completeness	83%	2.7K	41	527	32
Uniqueness	93%	286	0	23	0
Availability	100%	77	0	0	0
Timeliness	100%	12	0	0	0
Consistency	100%	892	0	0	0

KPIs per check category

Check category	KPI % ▲	Correct results	Warnings	Errors	Fatal errors
patterns	0%	0	0	33	0
bool	0%	0	0	0	10
numeric	72%	1.1K	0	407	11
nulls	75%	1.6K	41	505	21
pii	83%	87	0	18	0
uniqueness	93%	294	0	23	0
schema	98%	1.5K	0	22	11
volume	100%	454	0	0	0
timeliness	100%	12	0	0	0
text	100%	22	0	0	0
availability	100%	77	0	0	0

Figure 1.4. DQOps platform offers more than 50 built-in data quality dashboards. Users can modify these dashboards or create custom ones using Looker Studio. The sample screen shows the "KPIs per table - summary" dashboard.

Problems that can appear at this stage:

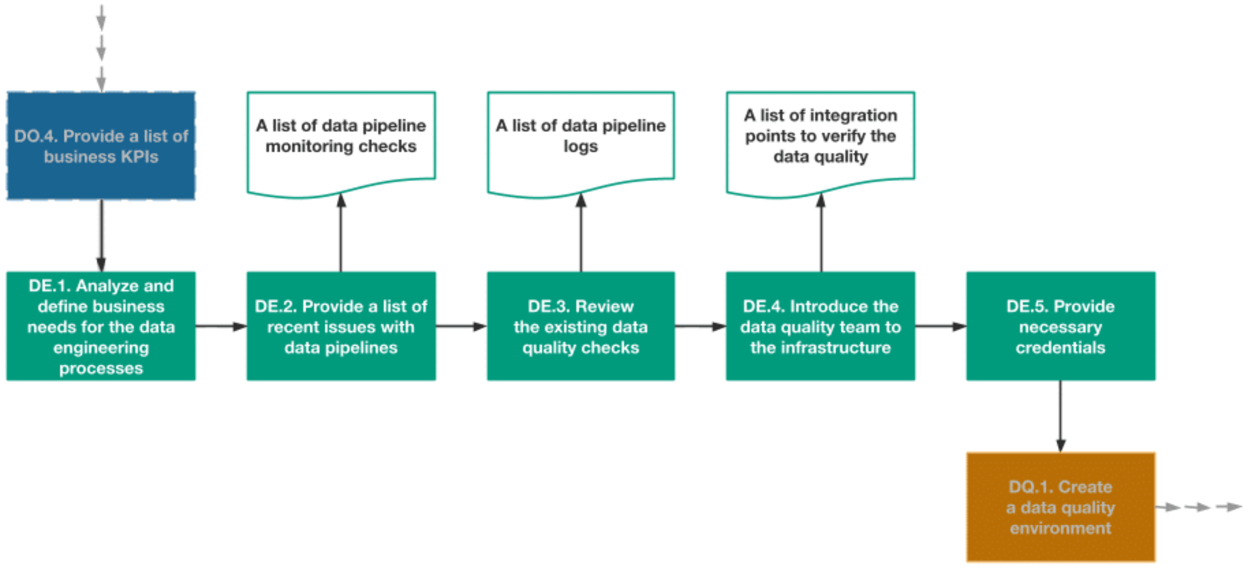
- The database model is not correctly documented.
- The Data Owner's requirements are unrealistic to accomplish with current data quality tools.

- Due to the volume of data, performing data quality checks (such as complex SQL queries) on current hardware is not economically feasible. A costly database upgrade would be required.

The result of this stage is a prepared list of KPIs with descriptions for the Data Quality Team. Sample column names are shown below.

Requirement ID	Business requirement	Related tables	Metrics to measure	Expected metric value
...	...	...	...	...

2. Define data quality requirements from the data engineering perspective


DO.4. Provide a list of business KPIs
A list of data pipeline monitoring checks
A list of data pipeline logs
A list of integration points to verify the data quality
DE.1. Analyze and define business needs for the data engineering processes
DE.2. Provide a list of recent issues with data pipelines
DE.3. Review the existing data quality checks
DE.4. Introduce the data quality team to the infrastructure
DE.5. Provide necessary credentials
DQ.1. Create a data quality environment

The Data Engineering Team manages the data warehouse or data lake. Incoming data quality issues may affect the stability of the data pipelines. These issues may accumulate, affecting tables in the downstream data warehouse or data lake layers. Many of these data quality issues may be detected in advance, allowing the Data Engineering Team to stop processing and fix the issue.

This stage describes the critical steps for gathering the data engineering requirements in the data quality area, introducing the data engineers to the data quality tool, and connecting the data quality tool to the data platform.

The data engineers must also introduce the Data Quality Team to the data platform infrastructure. One of the most critical areas is the integration points to the existing system, e.g., where to set up data quality checks at certain stages of data pipelines. Finally, the Data Engineering Team provides the Data Quality Team with all the necessary authorization credentials.

Well-designed data warehouses, ETL tools, and data pipelines report the progress of data pipelines and errors as logs. However, to fully understand the complexities of data processing, it is necessary to have a bird's eye view of the entire process. Proper implementation of data observability can provide such a view, allowing organizations to monitor, understand, and troubleshoot their data infrastructure in real-time. This includes monitoring data ingestion, transformation, and storage processes to identify anomalies, errors, or deviations from expected behavior. This empowers data teams to proactively address potential issues before they escalate, thus minimizing the risk of data downtime, inaccuracies, or disruptions to business operations.

DQOps streamlines data observability by automatically activating pre-selected data quality checks on your data sources. As shown in Figure 2.1, the user interface allows you to customize this configuration, defining which checks are activated by default. DQOps platform empowers you to detect two key anomalies: outliers, which are new minimum or maximum values deviating significantly from the norm, and distribution shifts, where typical values like mean, median, or sum change (as illustrated in Figure 2.2).

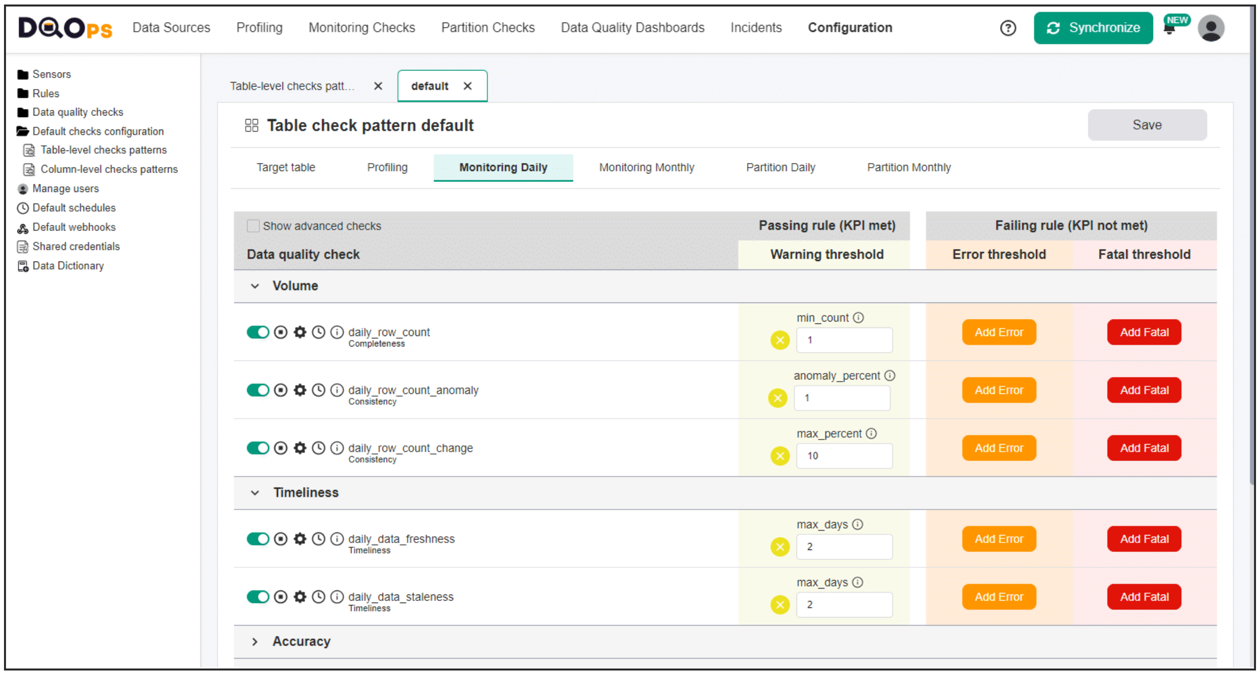


Table check pattern default

Data quality check	Passing rule (KPI met): Warning threshold	Failing rule (KPI not met): Error threshold	Fatal threshold
Volume			
daily_row_count (Completeness)	min_count: 1	Add Error	Add Fatal
daily_row_count_anomaly (Consistency)	anomaly_percent: 1	Add Error	Add Fatal
daily_row_count_change (Consistency)	max_percent: 10	Add Error	Add Fatal
Timeliness			
daily_data_freshness (Timeliness)	max_days: 2	Add Error	Add Fatal
daily_data_staleness (Timeliness)	max_days: 2	Add Error	Add Fatal
Accuracy			

Figure 2.1. A sample screen from the DQOps platform shows the default configuration of table-level volume and timeliness data quality checks. Default checks are automatically activated on monitored data sources. Learn more about how DQOps enables data observability.

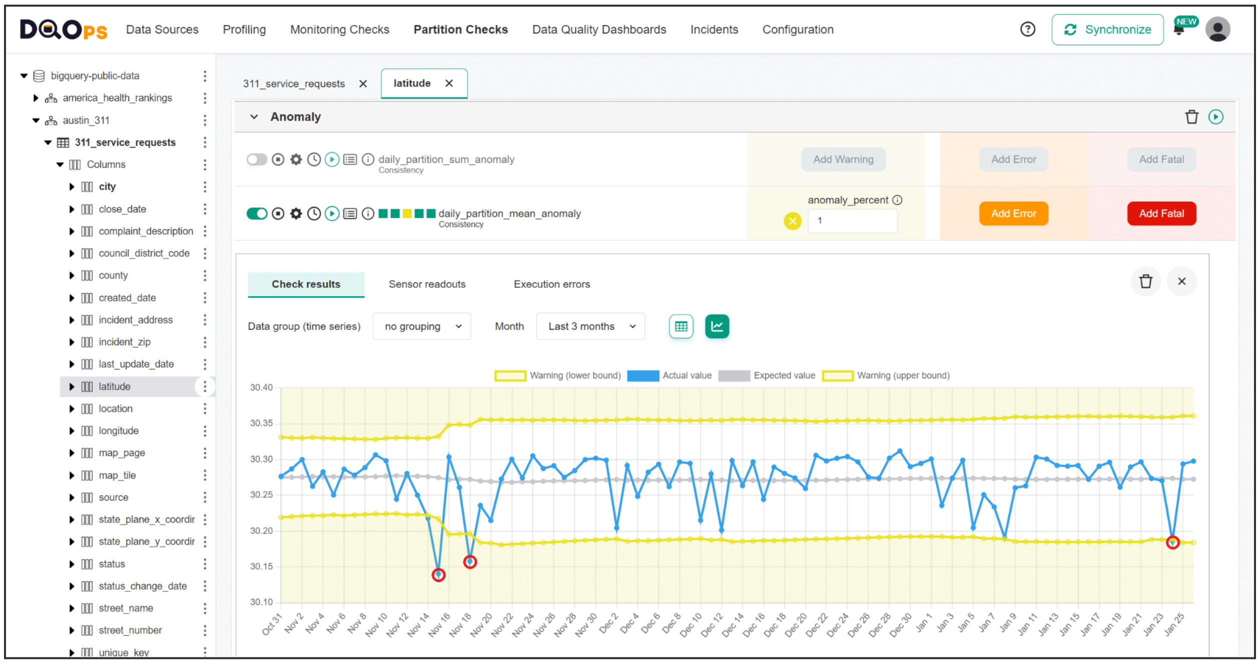


Figure 2.2. A sample screen from the DQOps platform user interface shows detected anomalies. The graph shows a time series of recent mean values measured with a daily_partition_mean_anomaly check. Learn more about anomaly detection in DQOps.

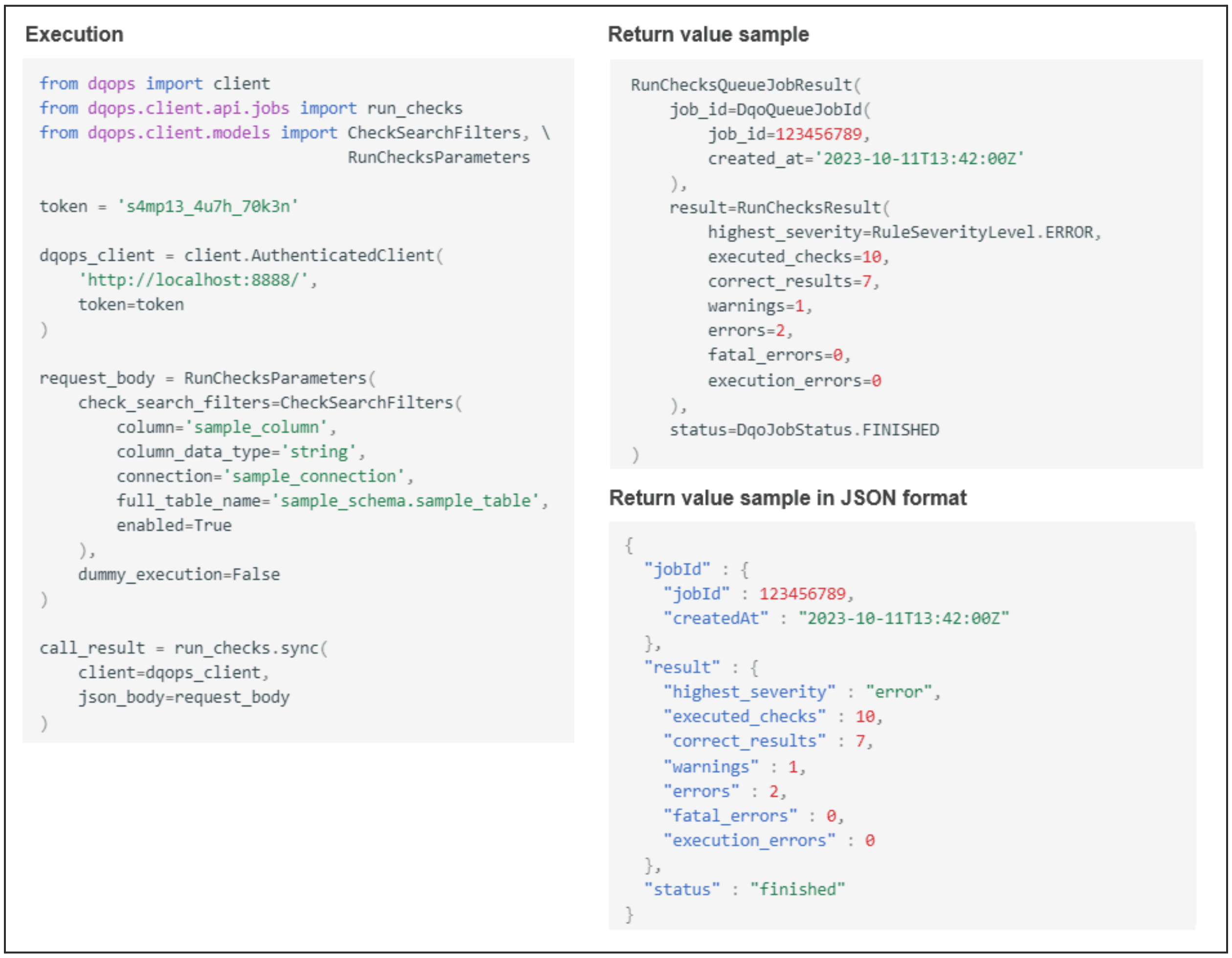
DE.1. Define requirements for the data engineering process

After analyzing the Data Owner's needs and specifying the right metrics and checks, the Data Engineering Team transfers the Data Owner's requirements to the technical team. The Data Engineering Team also specifies what they want to monitor regarding data quality.

Before moving to the next stage, make sure that the following steps have been completed.

- **Review the data quality requirements from the data engineering perspective.** Based on the data quality monitoring requirements prepared by the Data Owner, the Data Engineering Team provides feedback about the feasibility of accessing the data.
- **Define an incident resolution process.** The Data Engineering Team is responsible for resolving data quality issues caused by pipeline errors or low-quality data from the source systems. A data quality platform should be integrated into the workflow, providing early warnings for issues or helping analyze the root cause. The Data Engineering Team should provide the requirements for the data quality platform and how the platform should integrate with the existing workflow.
- **Identify notification channels for data quality issues.** The Data Engineering Team may use Slack, Microsoft Teams, or email for notifications. Jira, ServiceNow, Azure DevOps, or any other ticketing platform can also be used to track and assign work items. The data quality platform should publish notifications to these channels or directly interact with the ticketing platform, creating tickets in real time.
- **Define the need for process improvement monitoring.** If the data quality platform opens tickets directly in the system, it can lead to a large number of tickets. To manage this, the Data Engineering Team should provide a list of KPIs to track the increase or decrease in the number of issues. These KPIs can be tracked on the data quality dashboards for data engineering. One example of a KPI is the number of data quality issues detected as anomalies using an algorithm for detecting time series anomalies.
- **Integrate points with the existing DevOps and DataOps infrastructure.** The data platform may already use DevOps/DataOps practices, such as storing the data pipeline definitions in a source code repository like Git. Continuous delivery pipelines automatically deploy new code changes to the platform. Also, many modern data processing platforms use scripting to implement the data processing steps. The most popular examples are Apache Airflow for scheduling and dbt for data processing. All those platforms define the data processing steps in code, offering multiple extension points to call the data quality platform to evaluate the latest batch of data before it is accepted and loaded into the downstream systems. You can also seamlessly integrate

DQOps into custom data pipelines and ML pipelines by calling a Python client for DQOps. Figure 2.3 provides a concrete example of using Python code to call the run_checks operation and execute data quality checks within your pipelines.



Execution

```
from dqops import client
from dqops.client.api.jobs import run_checks
from dqops.client.models import CheckSearchFilters, \
                                RunChecksParameters

token = 's4mp13_4u7h_70k3n'

dqops_client = client.AuthenticatedClient(
    'http://localhost:8888/',
    token=token
)

request_body = RunChecksParameters(
    check_search_filters=CheckSearchFilters(
        column='sample_column',
        column_data_type='string',
        connection='sample_connection',
        full_table_name='sample_schema.sample_table',
        enabled=True
    ),
    dummy_execution=False
)

call_result = run_checks.sync(
    client=dqops_client,
    json_body=request_body
)
```

Return value sample

```
RunChecksQueueJobResult(
    job_id=DqoQueueJobId(
        job_id=123456789,
        created_at='2023-10-11T13:42:00Z'
    ),
    result=RunChecksResult(
        highest_severity=RuleSeverityLevel.ERROR,
        executed_checks=10,
        correct_results=7,
        warnings=1,
        errors=2,
        fatal_errors=0,
        execution_errors=0
    ),
    status=DqoJobStatus.FINISHED
)
```

Return value sample in JSON format

```
{
  "jobId" : {
    "jobId" : 123456789,
    "createdAt" : "2023-10-11T13:42:00Z"
  },
  "result" : {
    "highest_severity" : "error",
    "executed_checks" : 10,
    "correct_results" : 7,
    "warnings" : 1,
    "errors" : 2,
    "fatal_errors" : 0,
    "execution_errors" : 0
  },
  "status" : "finished"
}
```

Figure 2.3. Example of Python code calling the DQOps run_checks operation to execute data quality checks (left panel). The right panel summarizes the executed data quality checks with the highest data quality issue severity level "error."

It is important to pay attention to possible problems that may occur at this stage.

- The existing ETL platform may not offer an extension point for executing data quality checks.

- Due to the volume of data queried every day, running the data quality checks required by the Data Owner is not feasible on the existing physical infrastructure. The data quality checks should be implemented as SQL queries, but their complexity may require heavy full-table scans or complex joins across the tables. The data platform may not have enough computing power to process these queries on a daily basis.

DE.2. Provide a list of recent issues with data pipelines

Many data quality issues do not originate from the source system but are caused by bugs or instability in the data loading platform. Typical causes of issues are canceled jobs, timeouts, lack of disk space for temporary files, out-of-memory errors, network failures, or bugs in the transformation code.

The result of these issues is visible mainly as missing or incomplete data in the data warehouse or data lake. In projects that require real-time or near real-time processing, these issues should be detected as soon as possible. The data quality platform should identify most of these problems. Timeliness data quality checks (freshness) detect an increasing processing lag. Completeness checks can detect missing data when an incidental data pipeline failure occurs. Accuracy checks can compare the cleansed data with the source data system.

At this stage, the Data Engineering Team should review the list of incidents caused by some problems with the data pipelines. The Data Quality Team assesses which data quality checks and data quality dimensions can detect or predict these issues in advance.

Before starting the next stage, the following steps must be taken.

- **Prepare a list of recent issues with data pipelines.** The list of recent data quality incidents is usually tracked in a ticketing system or a task management system, such as Jira.
- **Prioritize data pipelines.** Not all data pipelines are the same. Data pipelines that load core tables should be monitored end-to-end. Pipelines that load dictionary tables should be monitored if these tables change frequently, and their referential integrity is crucial.
- **Provide the frequency of data loading.** Timeliness (data lag) data quality checks can measure the freshness of data. However, the volume of data or its format may not allow for real-time data loading. Timeliness data quality checks must consider the frequency of data loading and the expected time of the day when the data pipeline is executed. Scheduling the subsequent data quality checks should be based on the data pipeline

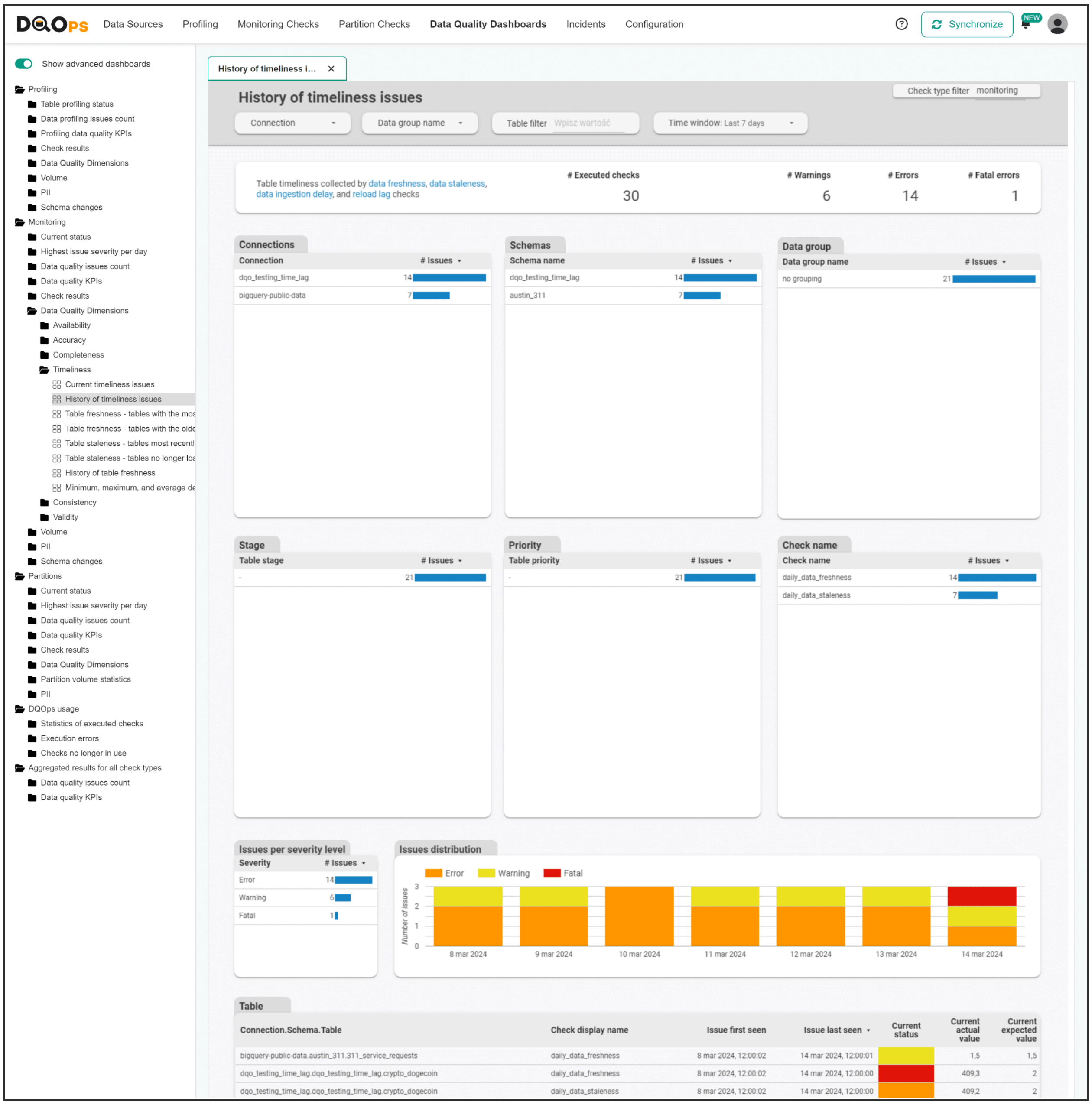
execution frequency and a relevant timeframe (time of the day for a daily refresh, day of the week for weekly, day of the month for monthly, etc.). This enables the data to be verified as soon as it is expected to appear in the database.

- **Identify parallel data streams (groupings).** You may have tables that aggregate data from different data sources, which load data in separate data pipelines. Rows from different data streams can be identified by specific columns in the table, such as country, state, vendor, data provider, or department. A data quality platform should be able to monitor each data grouping separately. In DQOps, this is simply achieved by running a "GROUP BY" clause in the data quality SQL query.

- **Define data pipeline extension points.** Some data processing and scheduling platforms, such as Airflow, support extension points. These are places in the data pipeline where a data quality tool can be called to verify a new batch of data before confirming that the data loading process can continue. DQOps easily integrates with Apache Airflow.

The DQOps platform supports different types of timeliness checks, such as data freshness, staleness, and ingestion delay. Data freshness refers to the age of the most recent row in the monitored table. It is measured by calculating the difference between the most recent event timestamp in a table and the current system time when observing the freshness. Data staleness is similar to data freshness but measures the time since the target table was most recently loaded. The ingestion delay measures the time it takes for the data pipeline or ETL process to pick up the latest data and load it into the target table.

DQOps uses data quality dashboards to present the current state of data timeliness and measure the data timeliness KPI over time. The "History of timeliness issues" dashboard, shown in Figure 2.4, allows you to review recent timeliness issues.



History of timeliness issues

Check type filter: monitoring

Connection | Data group name | Table filter | Time window: Last 7 days

Table timeliness collected by data freshness, data staleness, data ingestion delay, and reload lag checks

# Executed checks	# Warnings	# Errors	# Fatal errors
30	6	14	1

Connections

Connection	# Issues
dqo_testing_time_lag	14
bigquery-public-data	7

Schemas

Schema name	# Issues
dqo_testing_time_lag	14
austin_311	7

Data group

Data group name	# Issues
no grouping	21

Stage

Table stage	# Issues
-	21

Priority

Table priority	# Issues
-	21

Check name

Check name	# Issues
daily_data_freshness	14
daily_data_staleness	7

Issues per severity level

Severity	# Issues
Error	14
Warning	6
Fatal	1

Table

Connection.Schema.Table	Check display name	Issue first seen	Issue last seen	Current status	Current actual value	Current expected value
bigquery-public-data.austin_311.311_service_requests	daily_data_freshness	8 mar 2024, 12:00:02	14 mar 2024, 12:00:01	🟨	1,5	1,5
dqo_testing_time_lag.dqo_testing_time_lag.crypto_dogecoin	daily_data_freshness	8 mar 2024, 12:00:02	14 mar 2024, 12:00:00	🟥	409,3	2
dqo_testing_time_lag.dqo_testing_time_lag.crypto_dogecoin	daily_data_staleness	8 mar 2024, 12:00:02	14 mar 2024, 12:00:00	🟧	409,2	2

Figure 2.4. A sample "History of timeliness issues" dashboard from the DQOps helps review recent timeliness issues.

It is worth paying attention to the following problems with the data pipelines:

- Network issues during data transfer.
- Disk space issues.

- Out-of-memory errors.
- Exceeded API limits.
- Outdated credentials.
- Configuration issues.
- Suboptimal way of delivering files.
- Invalid schedules for running the data pipelines.

DE.3. Review the existing data quality checks

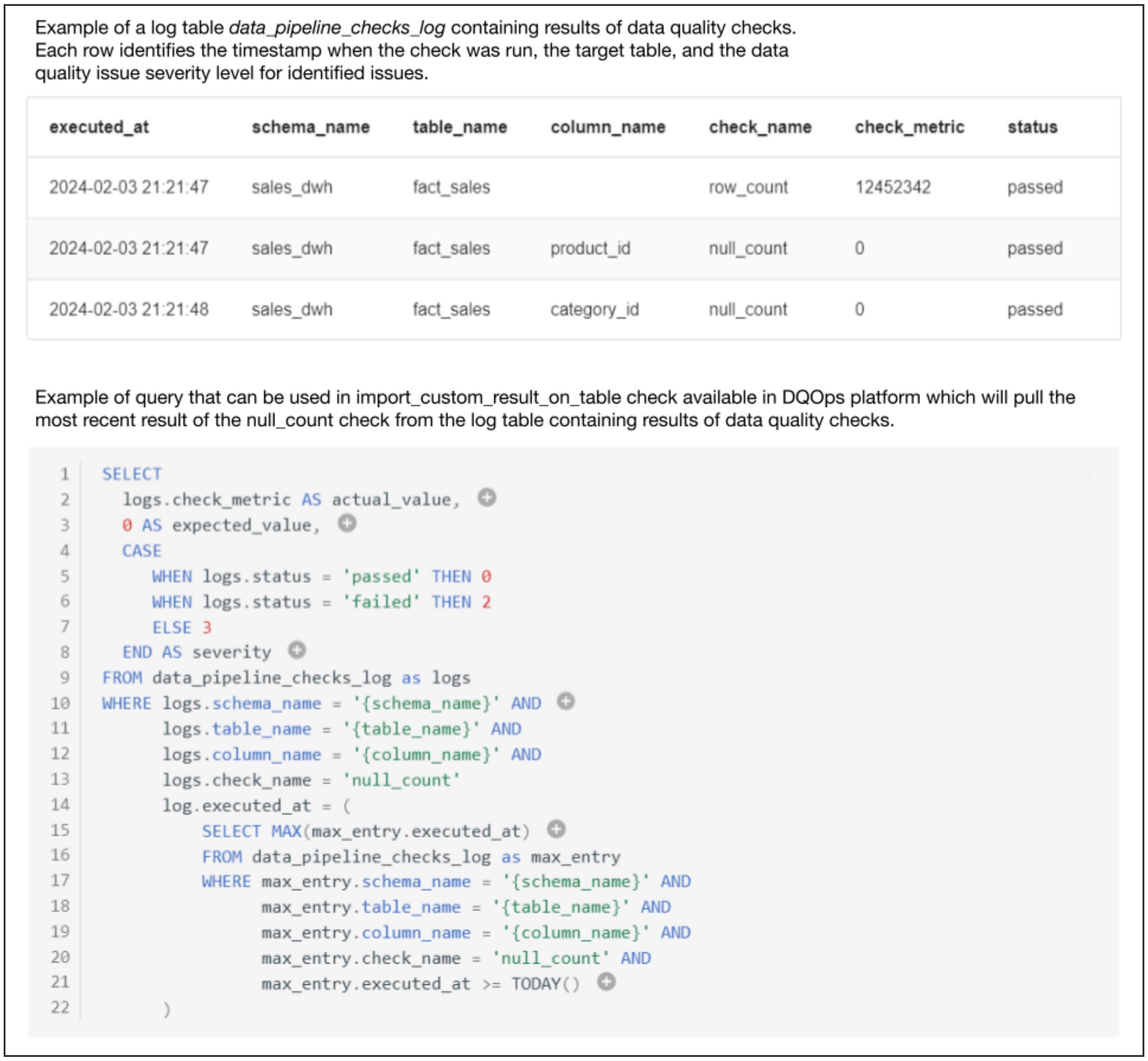
Certain data quality checks may already be implemented in the data pipeline and executed at different stages of data loading. Results of these data quality checks are typically stored in flat files, inserted into a dedicated table, or forwarded to a cloud-based logging platform such as Azure Monitor Logs, Google Cloud Logging, or AWS CloudWatch Logs. What is usually missing is a high-level picture of the most frequently arising issues. A clear understanding of recurring issues helps you find the root cause of the problem and avoid them in the future. This emphasizes the importance of identifying all of those logging integration points.

The Data Engineering Team should review the code of the data pipelines and the configuration of ETL tools, looking for all possible places where valuable log information is stored. The logs should identify the source or target tables that were referenced. The logs should also have a timestamp indicating when the error occurred. Additional important information is the severity level of the problem (e.g., distinction between warnings and errors).

The Data Engineering Team should consult with the Data Quality Team on how their existing logs can be integrated into a complete data quality result database.

In many cases, log files can be queried just like regular databases. Data quality tools, like DQOps, that support custom data quality check definitions, can query these logs or aggregate log entries. Figure 2.5 provides an example of a check available in the DQOps platform, where custom SQL expressions are used to retrieve the latest results from the log table.

Consider a simple example: the data pipeline writes an entry (inserts one row) to the logging table in the database for each file that has been processed. The log entry identifies the target table, the date and time the data is loaded, and the most important value - the number of rows loaded in that batch. A custom data quality check can aggregate all these log entries daily, add up the number of loaded rows, and verify that the total number of rows matches the threshold.


Example of a log table *data_pipeline_checks_log* containing results of data quality checks. Each row identifies the timestamp when the check was run, the target table, and the data quality issue severity level for identified issues.

executed_at	schema_name	table_name	column_name	check_name	check_metric	status
2024-02-03 21:21:47	sales_dwh	fact_sales		row_count	12452342	passed
2024-02-03 21:21:47	sales_dwh	fact_sales	product_id	null_count	0	passed
2024-02-03 21:21:48	sales_dwh	fact_sales	category_id	null_count	0	passed

Example of query that can be used in import_custom_result_on_table check available in DQOps platform which will pull the most recent result of the null_count check from the log table containing results of data quality checks.

```
SELECT
  logs.check_metric AS actual_value,
  0 AS expected_value,
  CASE
     WHEN logs.status = 'passed' THEN 0
     WHEN logs.status = 'failed' THEN 2
     ELSE 3
  END AS severity
FROM data_pipeline_checks_log as logs
WHERE logs.schema_name = '{schema_name}' AND
      logs.table_name = '{table_name}' AND
      logs.column_name = '{column_name}' AND
      logs.check_name = 'null_count'
      log.executed_at = (
          SELECT MAX(max_entry.executed_at)
          FROM data_pipeline_checks_log as max_entry
          WHERE max_entry.schema_name = '{schema_name}' AND
                max_entry.table_name = '{table_name}' AND
                max_entry.column_name = '{column_name}' AND
                max_entry.check_name = 'null_count' AND
                max_entry.executed_at >= TODAY()
      )
```

Figure 2.5. DQOps has built-in data quality checks using custom SQL expressions, which is a quick method for running custom data quality checks without defining reusable (custom) checks.

Complete the following steps before moving to the next stage.

- **Identify the logging framework.** The Data Engineering Team should find out what logging framework or custom logging method is used in the data loading process.

- **Hand over the list of existing data quality checks.** The Data Engineering Team should prepare a list of previously implemented data quality checks and hand it over to the Data Quality Team.
- **Identify existing data quality logs.** If any data quality checks are being run, the Data Engineering Team can share logs from these checks with the Data Quality Team. These existing logs can be migrated and loaded into the global data quality database.

Required fields in the log entries:

- Source table or target table.
- Date and time.
- Type of entry.
- A field to distinguish between successful and failed entries.

Optional fields in the log entries:

- A number (such as the number of rows processed).
- Severity (when errors and warnings are distinguished from each other).
- Data pipeline ID.
- Additional key/value pairs.
- A path to a file that was processed.
- Batch ID (if the data is executed in batches).

The following types of events are commonly logged and should be analyzed:

- Errors in the data pipelines.
- Exceptions raised by custom code.
- Data parsing errors.
- Start and finish notifications for a certain step in the data pipeline.
- Messages with a number of rows or files that were processed.
- Custom data quality check results, such as detection of null values.

Problems that may occur at this stage:

- The data pipeline does not have a logging layer.
- The format of logs does not allow them to be read without significant parsing or processing.
- The regular expressions needed to parse log entries are very complex.

This stage results in a list of data pipeline logs provided to the Data Quality Team. The logs should include information about the processing steps executed by the data pipeline and a record of data quality checks executed inside the data pipeline.

DE.4. Introduce the Data Quality Team to the infrastructure

Data engineers build, monitor, and maintain data pipelines and ETL processes. When introducing data quality practices into the architecture, knowledge transfer is required so that the Data Quality Team can fully understand the infrastructure and tools that are used in the data platform.

To build a rich data infrastructure, data engineers require a mix of different programming languages, data management tools, data warehouses, and entire sets of other tools for data processing, data analytics, and AI/ML. Some of the most commonly used tools in data platforms include:

- Cloud infrastructure (GCP, AWS, Azure).
- SQL databases.
- Big data SQL engines such as Apache Spark, Presto, and AWS Athena.
- ETL platforms.
- Data ingestion tools.
- Scheduling platforms (such as Airflow or Prefect).
- Business intelligence tools.

At this stage, the Data Engineering Team should introduce the Data Quality Team to the infrastructure and prepare a list of integration points to verify the data quality.

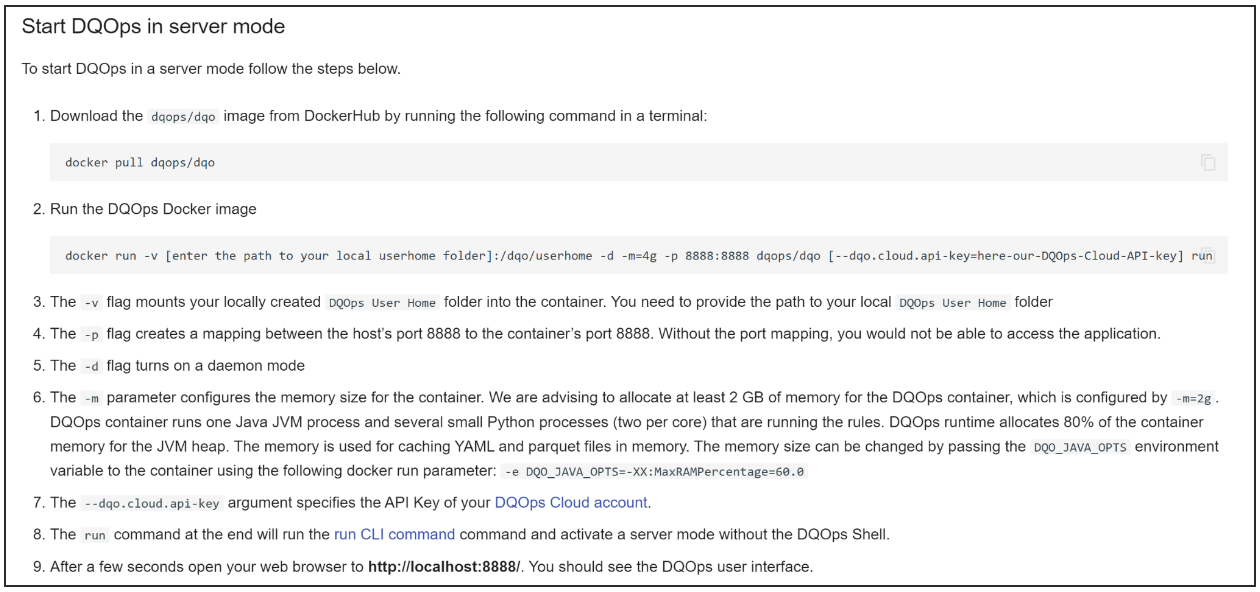
The following infrastructure components in the data platform should be identified:

- **Computing infrastructure.** Identify the type of virtual machines or SaaS platforms that are used to store and process the data.
- **Database engines.** Identify all the database engines that are used and should be monitored for data quality issues. DQOps supports integration with all the most popular databases.
- **Network topology.** Some servers may be located in secured locations, protected from external access using firewalls. A data quality platform must be whitelisted to access those servers over the network. In some circumstances, the database (such as the Kerberized Hadoop cluster) can only be queried from a dedicated node (bastion server) that is part of the cluster. Some components of the data quality platform must be deployed on a remote server using an agent architecture.
- **Personal Identifiable Information (PII) data.** Analyzed tables may store sensitive information. To avoid data leakage, data quality checks for these tables must be designed with a review and approval process. DQOps supports the detection of PII data in tables.
- **File storage.** Data lakes store the data in flat files, such as CSV or Parquet files. External tables based on flat files should also be monitored for data quality issues. The location, file format, and directory format for these file locations should be documented. Folders that meet the Apache Hive folder structure for partitioned data can be easily queried as regular tables using Apache Spark, Snowflake, Presto, or Microsoft PolyBase.
- **Big data engines.** Many big data engines, such as Apache Spark, Apache Hive, Presto, and Trino, require special drivers or additional configuration to establish a connection. DQOps supports all those engines.
- **ETL platforms.** The ETL platform used for the full extract-transform-load process can log essential information at each stage. Many ETL platforms can also call on external services, such as a data quality platform.
- **Data ingestion tools.** The data platform can use basic data ingestion tools with limited data transformation capability, such as Airbyte.
- **Scheduling tools.** Information about the configuration of a scheduling tool is essential to integrate the data quality process into the overall data processing pipeline. A scheduling tool like Apache Airflow can delay the data loading process until the data quality tool is executed. Knowledge of scheduling is also essential to configure data quality checks that should be scheduled when data is expected to be present. Still, the

data pipeline is not able to trigger the data quality checks automatically when the data load operation finishes.

- **Business intelligence tools.** Organizations use business intelligence (BI) tools to analyze data and gain insights into business operations. However, there are instances where data quality issues may arise, leading to inaccurate numbers being displayed on the dashboards generated by these tools. This is where data observability comes in. It is a process that ensures data quality issues are detected and resolved before they reach the end-users. Robotic process automation tools can be used to screen-scrape dashboards. Additional filters and transformation logic can be applied to the BI tool's dashboard or data model to ensure the accuracy of the numbers displayed. Additionally, periodic refreshes performed by the BI tool may also fail, leading to inaccuracies. Custom data quality checks can be implemented to compare aggregate numbers shown on dashboards with a data warehouse or system of record.

DQOps data quality platform can be installed using pip, Docker (Figure 2.6.), or directly compiling code.


Start DQOps in server mode

To start DQOps in a server mode follow the steps below.

1. Download the `dqops/dqo` image from DockerHub by running the following command in a terminal:

```
docker pull dqops/dqo
```

2. Run the DQOps Docker image

```
docker run -v [enter the path to your local userhome folder]:/dqo/userhome -d -m=4g -p 8888:8888 dqops/dqo [--dqo.cloud.api-key=here-our-DQOps-Cloud-API-key] run
```

3. The `-v` flag mounts your locally created `DQOps User Home` folder into the container. You need to provide the path to your local `DQOps User Home` folder
4. The `-p` flag creates a mapping between the host's port 8888 to the container's port 8888. Without the port mapping, you would not be able to access the application.
5. The `-d` flag turns on a daemon mode
6. The `-m` parameter configures the memory size for the container. We are advising to allocate at least 2 GB of memory for the DQOps container, which is configured by `-m=2g`. DQOps container runs one Java JVM process and several small Python processes (two per core) that are running the rules. DQOps runtime allocates 80% of the container memory for the JVM heap. The memory is used for caching YAML and parquet files in memory. The memory size can be changed by passing the `DQO_JAVA_OPTS` environment variable to the container using the following docker run parameter: `-e DQO_JAVA_OPTS=-XX:MaxRAMPercentage=60.0`
7. The `--dqo.cloud.api-key` argument specifies the API Key of your DQOps Cloud account.
8. The `run` command at the end will run the run CLI command command and activate a server mode without the DQOps Shell.
9. After a few seconds open your web browser to **http://localhost:8888/**. You should see the DQOps user interface.

Figure 2.6. The DQOps data quality platform can be run as a Docker container in server or command-line mode. You can also build a custom DQOps container image. The screenshot of the documentation section explains how to start DQOps in server mode.

The following problems may occur at this stage.

- Data must be queried only from a collocated server, limiting the choice of data quality tools to those that support multi-cloud remote agent architecture (such as DQOps).
- Additional firewall rules must be configured to access databases.
- Personal data must be handled with care.
- Data quality checks that access tables with sensitive data must undergo a review process. The data quality tool should support this process, for example, by requesting a pull in the source code repository. The tool must then store the definitions as easily readable flat files (such as data quality check specifications written in YAML or Python rules used in DQOps).

DE.5. Provide necessary credentials

Once the Data Engineering and Data Quality Teams have identified all the data platform components within the scope of data quality monitoring, it is time to decide on the expected level of data access authorization.

Data access rights must be defined for both the data quality engineers and data quality tools. The data quality engineers will also require access to consoles, query tools, or database management tools such as Oracle Toad, DBeaver, or Microsoft Management Studio. You may need to access cloud resources through the appropriate consoles on public clouds to review the list of available databases, schemas, datasets, and columns. The Data Quality Team engineers should be granted appropriate access rights to use AWS Management Console, Azure Portal, or GCP Console.

If landing or ingestion zones are monitored through flat file queries directly as external tables, reading and listing access rights at the file storage level are also required. Landing zones may receive files such as CSV, Parquet, or ORC. These flat files can be located on AWS S3 buckets, Azure Blob Storage, GCP Buckets, or the HDFS file system in Hadoop clusters.

The data quality tool may require installing a dedicated server or instantiating a new virtual machine. Personnel responsible for maintaining this environment must have server access rights. Alternatively, the data quality tool can be deployed to a shared environment like a Kubernetes cluster. The DQOps platform is available as SaaS or can be deployed on-premise.

Once the data quality tool is installed, it can operate on a technical account to avoid using personal accounts. Such technical accounts must be requested in advance, approved by authorized personnel, and granted respective access to data.

To avoid problems at this stage, ensure the following steps have been taken.

- **Introduction to the company's access control and/or credential management policy.** The Data Engineering Team should introduce the Data Quality Team to the company's security policies. The specific policy depends on the company but may include a password expiration policy.

- **Acquire the required computing resources for the data quality software.** The data quality tool may require a dedicated or virtual server or can be installed on an existing Kubernetes cluster. The platform must be correctly sized to meet the data storage and memory requirements for the data quality platform.

- **Ensure appropriate firewall rules.** The data quality platform can operate directly from the SaaS cloud or be installed in a separate environment that must be whitelisted to access the database or data lake.

- **Grant access to the Data Engineering Team.** The Data Engineering Team must be granted appropriate access to the databases, data lakes, and query tools.

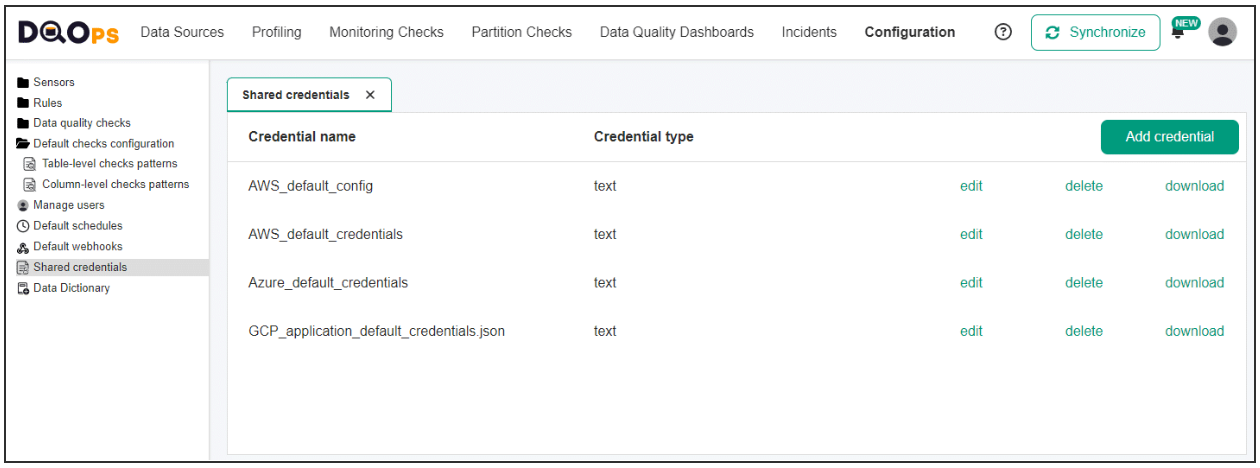
- **Create technical accounts for the data quality tool.** The data quality tool should access the monitored databases through a dedicated technical account, following the corporate password rotation policy accordingly. The workload executed by the tool on the monitored databases will be easy to identify and limit if running data quality checks affects the data platform's performance.

- **Assign access rights for the data quality tool.** Technical user accounts prepared to support the data quality tool should be granted access rights limited to listing metadata and executing SQL queries. Additional access rights that allow data manipulation, table management, or access rights should be avoided.

- **Provide access to the data pipeline platforms.** If a data quality tool is integrated directly into the data pipelines, it is helpful to provide the Data Quality Team with limited access rights. The data quality engineers will be able to monitor errors and review logs. For example, a dedicated operator called from an Apache Airflow DAG can call the data quality tool. Reviewing logs may be necessary to resolve integration issues between the data pipeline and the data quality platform.

- **Verify the scope of access rights.** The Data Engineering Team should verify the scope of access that has been granted to the Data Quality Team and the technical accounts. This helps protect against the disclosure of sensitive information and data breaches.

Credential management can be complex, especially when dealing with multiple data sources. To simplify this process, DQOps offers a centralized solution, illustrated in Figure 2.7. This figure showcases DQOps' user interface for storing and managing credentials. By securely storing your credentials within DQOps, you can streamline data access and avoid the need to manage them individually for each data source.



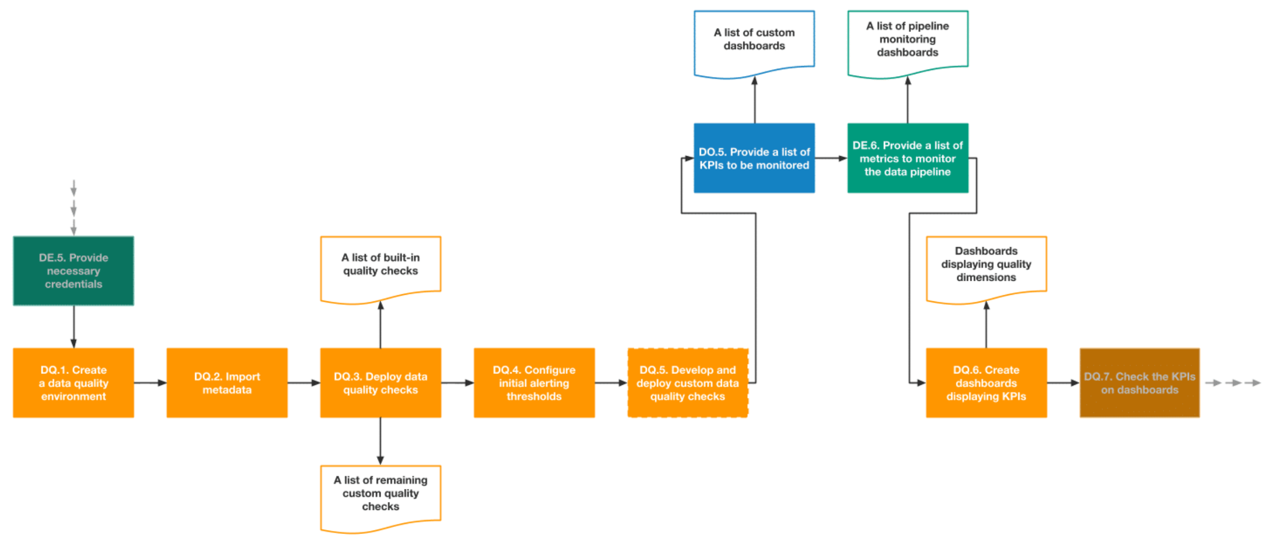
Credential name	Credential type			
AWS_default_config	text	edit	delete	download
AWS_default_credentials	text	edit	delete	download
Azure_default_credentials	text	edit	delete	download
GCP_application_default_credentials.json	text	edit	delete	download

Figure 2.7. DQOps enables users to store and manage their credentials in one place.

Note the following problems that may arise at this stage.

- Credentials are invalid.
- Access to the file storage (S3 Buckets, Azure Blobs, GCP Buckets) has not been granted.
- Firewall rules have not been reviewed.
- The credential expiration policy is not followed, so the data quality platform loses access to the monitored database.
- Granted access is not sufficient for the Data Quality Team.

3. Connecting data quality checks


A list of custom dashboards
A list of pipeline monitoring dashboards
DO.5. Provide a list of KPIs to be monitored
DE.6. Provide a list of metrics to monitor the data pipeline
DE.5. Provide necessary credentials
A list of built-in quality checks
Dashboards displaying quality dimensions
DQ.1. Create a data quality environment
DQ.2. Import metadata
DQ.3. Deploy data quality checks
DQ.4. Configure initial alerting thresholds
DQ.5. Develop and deploy custom data quality checks
DQ.6. Create dashboards displaying KPIs
DQ.7. Check the KPIs on dashboards
A list of remaining custom quality checks

At this stage, we have defined a list of monitoring tables and columns. We have also agreed upon the requirements for data quality checks and KPIs with the Data Owner and Data Engineering Teams. Any obstacles that might delay the implementation of the data quality control tool should have been identified and addressed. We have also provided the credentials to connect to monitored databases, data warehouses, and data lakes.

The Data Quality Team is now ready to activate the data quality monitoring. This chapter describes the following steps:

- Configuration of the data quality environment.
- Importing metadata about tables and columns from monitored data platforms into the data quality system.
- Activation of data quality checks based on requirements.
- Configuration of the alerting thresholds for the first time before the tuning process.
- Configuration of data quality KPI dashboards to show overall data quality scores for the entire data platform.

Once this stage is complete, the data quality platform can monitor data quality. However, the thresholds for triggering alerts are not yet properly tuned. Tuning and configuring thresholds will be covered in the next stages once the data quality KPI dashboards are operational.

DQ.1. Create a data quality environment

The Data Quality Team deploys the data quality platform on the provided environment. A SaaS-hosted environment may require configuring the firewall whitelisting rules. The supplied credentials and connection details must be verified as the first step. Both the Data Quality Team members and the data quality tool (which may use a separate technical account) must be able to query the target database and import the metadata.

The Data Quality Team can also deploy the data quality tool on provisioned on-premise servers or virtual machines. Setting up multiple clouds or accessing a secured data warehouse or data lake may require the installation of a monitoring agent on a provisioned virtual machine or Kubernetes cluster.

A fully configured data quality environment may also require a data quality metrics database and a business intelligence tool connected to that database to display customized data quality dashboards.

Finally, the data quality platform may support different integration modes. It can operate as a standalone platform triggered by a built-in scheduler that activates the data quality checks at a set time. For example, once a day at 6 AM, when the data loading processes were supposed to complete the nightly refresh process. On the other hand, a data quality platform can be integrated directly into existing data pipelines, triggered as a blocking step in existing Apache Airflow DAGs.

If the integration mode requires direct integration with an existing scheduling platform, additional steps specific to the type of scheduler (Apache Airflow, Prefect) must be followed. Some components of the data quality platform, such as the client interface, CLI interface, Airflow operators, or scripts, must also be installed on the chosen scheduler.

Pay attention to the steps below to build a reliable data quality environment.

- **Optional cloud environment configuration.** Provide and configure the cloud environment, including accounts, billings, projects, subscriptions, and instances.
- **Deploy the data quality platform.** Follow the platform's documentation, particularly to meet CPU, memory, and disk capacity requirements. An undersized environment may not be stable.
- **Deploy remote monitoring agents.** Optionally, a multi-cloud environment or a platform managed as a SaaS may require a direct connection to the monitored platform, reachable from a co-located server.

- **Configure connections to requested databases.** Use credentials for technical accounts instead of personal accounts. For highly distributed environments, appropriate firewall rules must also be configured. Otherwise, the data quality platform cannot connect to the monitored databases using database protocols. Although the user interface of web-based database platforms may use HTTPS (port 443), database connectors (such as JDBC drivers) may use separate ports for binary protocols. The default port for Microsoft SQL Server is 1433. Also, many big data platforms use the Hive Thrift server, which listens on port 10000.

- **Configure a connection to the monitored database in the data quality tool.** Connection details should be entered into the data quality platform. Sensitive information such as passwords, private keys, or API keys should be stored in a secure location like a key vault. A cloud environment can use AWS Secrets Manager, Azure Key Vault, or GCP Secret Manager.

- **Configure a data quality database (Optional).** Data quality metrics, sensor readouts, and alerts may be stored in a data warehouse for further analysis. Choose a database platform that can easily integrate with the data quality tool. DQOps simplifies this process by storing all the data quality data as Parquet files following the Hive partitioning scheme. This data can simply be replicated to a data lake or cloud bucket. Later, any SQL engine capable of querying Hive-compatible data can query the output files of the data quality tool. Data can be queried using Apache Hive, Apache Spark, DataBricks, Google BigQuery, Presto, Trino, SQL Server PolyBase, AWS Athena, and AWS Redshift Spectrum. DQOps creates a separate Data Quality Data Warehouse for each DQOps Cloud account, not sharing any tables or databases between tenants.

- **Configure in-house data quality dashboards (Optional).** Custom data quality dashboards that show business-relevant metrics can query the data quality database configured in the previous step. Ensure that all the users involved in the data quality process can access the dashboards. Business users involved in the negotiation and cooperation with external parties involved in a data-sharing agreement should be able to track the data quality KPIs related to sent or received data files. DQOps platform has over 50 customizable built-in data quality dashboards, allowing you to create data domain-specific views that will be shared with business sponsors, external partners, and other stakeholders.

- **Configure a scheduler.** Determine when the checks will be run (at what time of day and how often, e.g., daily, once a month).

Deciding how to deploy a data quality platform depends on your specific needs. Figure 3.1 compares the two primary deployment options available in DQOps: on-premise and Software as a Service (SaaS).

Differences between DQOps deployment options

On-premise

- **Control:** An on-premise solution is ideal for organizations with strict security requirements or those hesitant to store sensitive data in the cloud.
- **Customization:** The software can be customized to meet your specific needs and workflows.
- **IT expertise required:** Your IT team will be responsible for installation, maintenance, and updates.
- **Scalability:** Scaling the application to handle increased needs may require additional hardware or software investment.
- **Potential upfront costs:** On-premise solutions involve upfront licensing fees and ongoing maintenance costs.

Software as a Service

- **Ease of use:** SaaS solutions are easy to set up and use, requiring minimal IT involvement.
- **Automatic updates:** The vendor is responsible for updates and maintenance, ensuring you always have access to the latest features and bug fixes.
- **Scalability:** SaaS applications are designed to scale automatically to meet your changing needs.
- **Subscription-based cost:** You can choose from several subscription plans that match your needs with predictable costs.

Figure 3.1. The DQOps platform has two primary deployment options: on-premise and SaaS (Software as a Service). Each offers distinct advantages and considerations.

At this stage, there may be some problems that need attention:

- Insufficient computing, memory, or disk space capacity.

- Lack of prior identification of network rules and firewall settings.
- The prolonged approval process for integrating SaaS data quality solutions into the data platform architecture by the architecture or security boards.
- Issues with access to a specific integration point.
- Access to tools and resources not granted to the Data Quality Team in advance.

DQ.2. Import metadata

The data quality platform operates on monitored data sources by executing SQL queries, parsing flat files, or reading metrics from the monitored data platform. Additional data quality KPIs should be associated with the table name, such as the number of days a table has been refreshed on time. To run highly configured data quality checks and later track the quality on the table or column level, import a list of tables, columns, and data types into the data quality tool.

To avoid mistakes at this stage, make sure that the following steps have been taken.

- **Verify that the data quality platform has been granted the required access rights.** The data quality platform must be able to list schemas, tables, and columns from the monitored system.
- **Verify that the tables/columns listed in the requirements are found.** The list of tables requested for monitoring may not match their actual physical names.
- **Identify the documentation of the requested tables in the data catalog.** The organization may already have implemented a data catalog tool, such as Alation or Amundsen, which may already provide important guidelines. Reviewing the current documentation can provide crucial knowledge about the use of these data sources, their importance, and known previous data quality issues.
- **Prioritize the tables. A more extended data quality project should be divided into phases.** The most important tables, such as fact tables, should be covered first. The Data Quality Team should agree on the priority of the tables with the Data Owner and the Data Engineering Team. Importing the metadata in phases reduces the time for the first data quality insights for the most important tables. A priority field in work management, such as Jira or Azure DevOps, can be used to sort and prioritize the backlog. DQOps platform supports assigning priorities to the tables.

- **Review the table partitioning scheme.** Partitioning is essential for storing a huge volume of data while ensuring reasonable query response times. Date partitioned tables (such as click streams, event streams, and transaction logs) that have not been updated are perfectly designed for incremental data quality analysis. A data quality platform should be able to calculate separate data quality metrics for each daily partition. In addition, a data quality platform should be able to execute data quality checks incrementally, querying only the most recent time periods. For example, DQOps solves the challenge of monitoring date-partitioned tables by calculating separate metrics for each "day of data." KPIs, calculated as the number of passed data quality checks, can be counted for each daily partition separately.

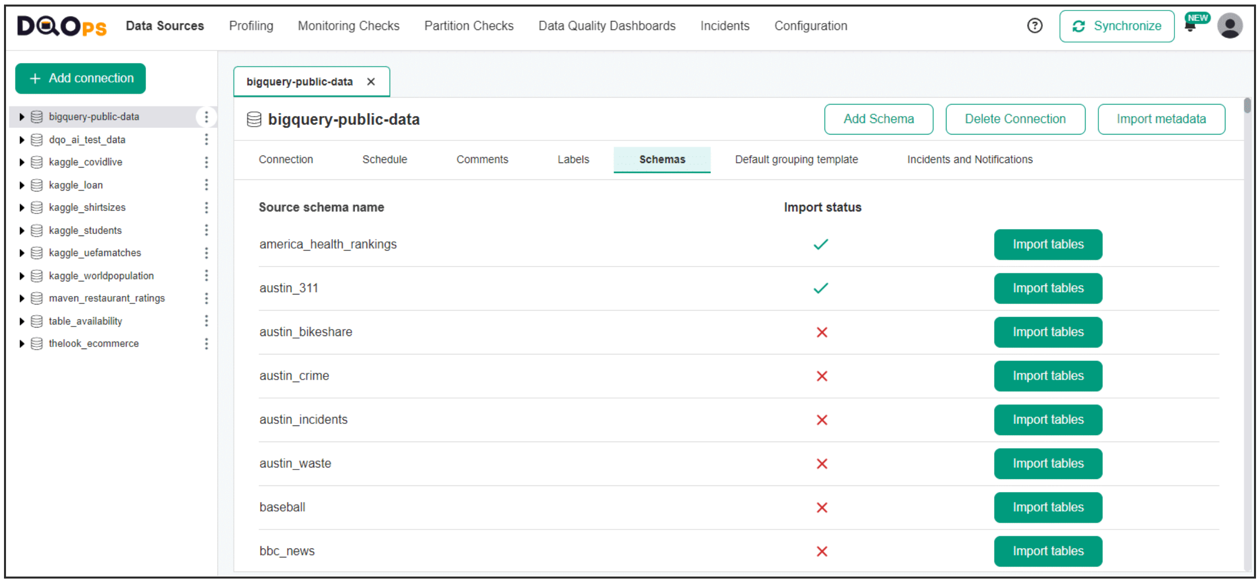
- **Identify the frequency of changes to past data.** Even daily partitioned data may be updated during the day. Especially if the data pipeline loads the data throughout the day and only the rows generated before the synchronization time exists. Synchronization on the following day will load the missing data. Still, the data quality readouts, such as the number of rows per day, will change after receiving the remaining rows for the previous day. The execution of data quality checks may be delayed to skip processing today's data or even yesterday's data to avoid calculating data quality metrics for partially loaded daily partitions. If the data quality checks are executed on partially loaded daily partitions, the data quality scores may be inadequate. The most affected are those data quality checks that compare the daily increment of the number of rows to the average daily increment.

- **Identify partition discriminators.** Previous partitions may be updated for daily partitioned data. The data may also be from a source that is generated manually, such as a Microsoft Excel file, which may be changed several times and reloaded into the database. Changes in the partition can be identified at the data level by checking the aggregated value calculated from all the rows in the partition. The simplest type of aggregated partition discriminator is the number of rows, which changes when missing rows are loaded. More complex discriminators can be calculated from the data. These can range from a simple aggregate field sum (a column measure from a fact table) to calculating the hash of all values in a column.

- **Import table metadata.** The Data Quality Team should import metadata about tables and columns based on their business priority. The most sensitive tables, such as transaction or fact tables, should be imported earlier. Importing the metadata of the entire database can be delayed and scheduled in subsequent sprints.

- **Run data profiling.** Review the tables to better understand their structure and typical content. Identify columns expected to have some percentage of null values. You should also import and identify columns that store aggregate measures, such as sales data or the number of impressions. DQOps allows you to run basic statistics on columns and tables to quickly learn about typical data values.

DQOps platform streamlines metadata import with a user-friendly interface. Figure 3.2 showcases a sample screen where you can browse and select schemas for metadata import.



Source schema name	Import status	
america_health_rankings	✓	Import tables
austin_311	✓	Import tables
austin_bikeshare	✗	Import tables
austin_crime	✗	Import tables
austin_incidents	✗	Import tables
austin_waste	✗	Import tables
baseball	✗	Import tables
bbc_news	✗	Import tables

Figure 3.2. A sample screen from the DQOps shows a list of schemas for importing metadata.

The problems that may occur at this stage:

- Incorrect paths to flat files.
- Access rights are not granted at a file system level (HDFS, S3 bucket, etc.) for data files used by external tables.
- Flat or columnar data files (Parquet, ORC) are corrupted, making some partitions unreadable.
- There is no documentation of the data model.
- There is outdated information about tables in the corporate data catalog.
- Data quality control requirements refer to missing tables or columns.
- Access rights are not granted to the Data Quality Team or technical account used by the data quality platform to query the monitored database.

- Incorrect partitioning of tables.
- Dates stored as text columns do not follow the same format.
- The tables in various databases are configured with different localization settings, resulting in format mismatch.

DQ.3. Deploy data quality checks

The activation of data quality checks should be divided into two steps. The first step, described here, involves activating data quality checks that collect data quality metrics from the monitored sources. This step should be followed by configuring the alerting thresholds, as described in the next step of the process (DQ.4. Configure initial alerting thresholds).

Data quality check definition

In DQOps, a data quality check is a test that can be run on both table or column levels. It is divided into two parts: a data quality sensor and a data quality rule. Briefly, the sensor reads the value from the data source at a given point in time, and the rule sets conditions for the sensor's readout. If the conditions are not met, the check detects an issue with the data.

Examples of sensor reads include the number of rows, the percentage of null values in a column, or the current delay between the timestamp of the latest row and the current system time. The sensors can be implemented as templated SQL queries (DQOps uses the Jinja2 templating engine) or as custom code that can call the appropriate source system's APIs. The definition of custom data quality sensors implemented as code is also supported in DQOps. The metrics captured by the data quality sensors (called sensor readouts in DQOPs) should be stored in the data quality database for further analysis. Time series analysis requires historical data to detect anomalies in the dynamics of the changes in the data set. A data quality sensor that captures the current row count of the table, which is scheduled daily, can build a complete history of the table row counts over a more extended period of time. This time series can be analyzed to detect anomalies on different levels, such as a sudden decrease or increase in the table row count. DQOps stores a copy of the sensor data locally on the monitoring agent. The data files are stored as Apache Parquet files in an Apache Hive-compatible folder tree, partitioned by the data source, monitored table name, and the month. A local copy of the sensor data enables a true multi-cloud data collection without accessing any sensitive data by an external cloud or SaaS solution.

The second part of the check, a rule, is a set of conditions against which sensor readouts are verified, described by a list of thresholds. A basic rule can score the most recent data quality result if the value is above or below a particular value or within the expected range. A standard data quality check on a table that counts the number of rows to detect empty or too-small tables uses a simple "minimum count 1" rule to instantly raise data quality alerts when the number of rows in a table is below 1. The separation of data quality rules enables more flexibility in implementing custom data quality rules that may use machine learning or time series analysis to detect not-so-obvious anomalies.

Below are the most common types of simple data quality rules.

1. **Simple rules.** These rules directly assess the current data quality sensor readout against a defined threshold.
   - **Single value comparison**
     - **Equals X.** Detects if a specific value (e.g., number of columns in a table) remains constant.
     - **Does not equal X.** This rule helps ensure a table is not empty (doesn't equal 0 rows) or identifies rows with invalid values (doesn't equal invalid value).
   - **Range comparison**
     - **Greater/less than X.** Verify whether a given value is within the expected range. For instance, it can be used to check whether a table is empty (row count >0). Similarly, it can also be used to check for null values in a column (the number of nulls in a column cannot be greater than 0).
     - **Between X and Y.** Ensures numerical values reside within a specific range (e.g., the percentage of valid rows between 95% and 100%).
2. **Relative value rules.** These rules compare the current sensor readout with a similar value from a previous time period, ignoring seasonality.
   - **The data quality sensor has not changed since the last readout.** This rule identifies minimal changes in data (e.g., the number of columns has not changed since yesterday (or the previous readout time).
   - **Change from a similar time window (not significant).** The rule compares the most recent data quality sensor readout (such as the current number of rows) to a comparable value exactly one week ago. For example, Monday's row count compared

to the previous Monday's. This type of rule avoids the effect of seasonality on data volatility.

3. **Time series rules**. These rules analyze changes in the data quality sensor readouts over a broader time window (weeks, months).

    - **Difference from average (percentage).** Compare the difference from the average value in percentage. This rule is relatively simple and easy to understand. It checks if the current value deviates more than a set percentage from the historical average (e.g., daily row count shouldn't differ more than 20% from the average).

    - **Standard deviation comparison.** This rule compares the difference from the average value in multiples of a standard deviation. The rule will automatically adjust to the variability of the data. It can detect an expected percentage of anomalies. For example, 99% of valid data quality readouts stay below 2.33 standard deviations, but outliers above 2.33 standard deviations fall into the top 1% of anomalies. The multiple of standard deviation (such as 2.33) can be easily converted to a quantile for readability. The alert is then more straightforward to understand: the increase in the number of rows was at the top 1% of the largest daily changes.

    - **Time series anomaly detection.** The rule uses statistical methods or seasonality analysis to identify anomalies and outliers. Possible algorithms in this category are ARIMA or Prophet.

Time and data grouping slicing

When configuring data quality checks, there are two more important aspects to consider. The first is the time-slicing of the table that will be monitored, and the second is the ability to calculate data quality metrics for different groups of rows stored in the same table.

Let’s discuss the first aspect. Many data quality solutions are limited to capturing data quality metrics for the whole table without considering that the old data is measured together with the most recent data. This limitation has serious implications, making many data quality results incorrect.

For instance, a simple data quality check that counts the percentage of rows with a non-negative value could produce misleading results if the data is not segmented by time. A data quality sensor that analyzes the whole table without time slicing and detects a percentage of valid rows where the value of a tested column is greater than 0 would run a SQL query similar to the following (Google BigQuery example).

```
SELECT
    CASE
        WHEN COUNT(analyzed_table.`target_column`) = 0 THEN 0.0
        ELSE 100.0 * SUM(
            CASE
                WHEN analyzed_table.`target_column` < 0 THEN 0
                ELSE 1
            END
        ) / COUNT(analyzed_table.`target_column`)
    END AS actual_value,
    CURRENT_DATETIME() AS time_period
FROM `your-google-project-id`.`<target_schema>`.`<target_table>` AS
analyzed_table
```

The above data quality sensor may return the result as follows:

time_period (metrics capture timestamp)	actual_value
2022-10-08	92.76%

This query measures the percentage of valid rows (the value in the tested column is greater than 0), but the data quality issues with the old and new rows will affect the final score equally. New issues that affected only yesterday's data may not be visible, as they are responsible for lowering the data quality score for only 1/356 of one year's data. Furthermore, reloaded daily or monthly partitioned data should be analyzed separately for each daily partition.

A data quality platform that considers time windows (time slicing) should support the calculation of data quality scores for each time period separately. DQOps solves this challenge by capturing metrics using a GROUP BY clause. For a day partitioned data, a similar query will also apply grouping by a timestamp column (an event timestamp, a transaction timestamp, or similar), truncated to the date. The following changes (marked in bold text) should be applied to the above SQL query to capture time-sliced data and calculate metrics for each day separately (Google BigQuery example).

```
SELECT
    CASE
        WHEN COUNT(analyzed_table.`target_column`) = 0 THEN 0.0
        ELSE 100.0 * SUM(
            CASE
                WHEN analyzed_table.`target_column` < 0 THEN 0
                ELSE 1
            END
        ) / COUNT(analyzed_table.`target_column`)
    END AS actual_value,
    DATE_TRUNC(CAST(CURRENT_TIMESTAMP() AS DATE), MONTH) AS time_period,
    TIMESTAMP(DATE_TRUNC(CAST(CURRENT_TIMESTAMP() AS DATE), MONTH)) AS
    time_period_utc
FROM `your-google-project-id`.`<target_schema>`.`<target_table>` AS
analyzed_table
GROUP BY time_period, time_period_utc
ORDER BY time_period, time_period_utc
```

The following time slicing is most useful for further reporting and tracking:

- hourly,
- daily,
- weekly (truncated to the beginning of the week),
- monthly,
- quarterly,
- yearly.

The results captured by the data quality sensor (a SQL query above) may look like this:

time_period	actual_value
2022-10-04	95.5%
2022-10-05	96.1%
2022-10-06	94.9%
2022-10-07	95.1%
2022-10-08	82.2%

Here, we can quickly identify a significant drop in the percentage of valid rows on 2022-10-08. This drop is below the average of around 95% valid rows per day. A score in the query that did not group the data by day and calculated an aggregate score for the table only detected a drop to 92.76%, which is not too far from the average score. It is important to note that the examples above show just five days of data, but in a real database, that drop may be below the average daily variation of the metric's value.

The second important aspect of data monitoring is the ability to calculate data quality metrics for different groups of rows stored in the same table. Data in the fact table can be loaded from other sources, countries, states, or received from various external sources. A different pipeline would load each data stream, and these pipelines may fail independently. Data streams (or data groupings) can be identified by a discriminator column, such as country or state. A data quality platform, such as DQOps, that can analyze data within separate segments adds a GROUP BY <data_grouping_discriminator_column> clause to the data quality queries. Querying data quality for each country separately without time slicing requires the following modifications (marked in bold text):

```
SELECT
    CASE
        WHEN COUNT(analyzed_table.`target_column`) = 0 THEN 0.0
        ELSE 100.0 * SUM(
            CASE
                WHEN analyzed_table.`target_column` < 0 THEN 0
                ELSE 1
            END
        ) / COUNT(analyzed_table.`target_column`)
    END AS actual_value,
    CURRENT_DATETIME() AS time_period,
    analyzed_table.`country` AS grouping_level_1
FROM `your-google-project-id`.`<target_schema>`.`<target_table>` AS
analyzed_table
GROUP BY grouping_level_1
ORDER BY grouping_level_1
```

The results pivoted for readability might look as follows:

Time_period (metrics capture timestamp)	US	UK	DE	...
2022-10-08	94.7%	95.8%	95.2%	

Data quality scores, calculated for each data source or vendor separately, can simplify the root cause analysis by linking the data quality incident to a data source, a data stream, an external data supplier, or simply a separate data pipeline that has loaded invalid data.

Finally, time slicing (capturing data quality scores separately for each time period) can be integrated with data grouping slicing. The GROUP BY clause must list columns that divide the data set by a data grouping discriminator column (country in this example). A complete SQL query that the data quality tool would execute on the data source should look like this:

```
SELECT
    CASE
        WHEN COUNT(analyzed_table.`target_column`) = 0 THEN 0.0
        ELSE 100.0 * SUM(
            CASE
                WHEN analyzed_table.`target_column` < 0 THEN 0
                ELSE 1
            END
        ) / COUNT(analyzed_table.`target_column`)
    END AS actual_value,
    analyzed_table.`country` AS grouping_level_1,
    DATE_TRUNC(CAST(CURRENT_TIMESTAMP() AS DATE), MONTH) AS time_period,
    TIMESTAMP(DATE_TRUNC(CAST(CURRENT_TIMESTAMP() AS DATE), MONTH)) AS
    time_period_utc
FROM `your-google-project-id`.`<target_schema>`.`<target_table>` AS
analyzed_table
GROUP BY grouping_level_1, time_period, time_period_utc
ORDER BY grouping_level_1, time_period, time_period_utc
```

The results of this query collect data quality scores for each day/country, allowing accurate identification of the source of the data quality issue.

Time_period	US	UK	DE	...
2022-10-04	96.4%	94.2%	95.2%	
2022-10-05	95.3%	94.7%	95.6%	
2022-10-05	93.9%	96.4%	96.2%	
2022-10-07	94.8%	94.9%	95.4%	
2022-10-08	94.7%	0%	95.2%	

Deployment of data quality checks

To complete the deployment of data quality checks, follow the steps outlined below.

- **Map built-in data quality checks to requirements.** Most data quality requirements should be easy to analyze using built-in data quality checks. These checks should be activated without any customization to their definitions.

- **Configure data profiling checks.** Profiling checks assess the initial data quality score of data sources. They should be activated on new data sources to verify that the minimum data quality requirements are met. Profiling checks are also helpful for exploring and experimenting with various types of checks to determine the most suitable ones for regular data quality monitoring. In DQOps platform, profiling checks store only one data quality profiling result for each month. If the user runs the same profiling again during the same month, the previous result is replaced. This behavior is designed for experimentation and tuning the parameters for the data quality rules.

- **Configure data monitoring checks.** Monitoring checks are used to continuously monitor the quality of data sources. The data quality results generated by monitoring checks in DQOps capture an end-of-day or an end-of-month data quality status of monitored data. If a daily data quality monitoring check is re-evaluated during the day, DQOps will overwrite previous data quality readouts and alerts. Only one most recently evaluated data quality readout and data quality alert can be stored in the data quality database. Monitoring checks are used to track and checkpoint the end-of-day (or end-of-month) data quality status for every data source, measure the improvement of the data quality score using data quality KPIs to prove the trustfulness of data sources, and present the progress of data cleansing projects to stakeholders and business sponsors of the data quality initiative. Daily monitoring checks also support anomaly detection for time series. DQOps can detect potential data quality issues when an anomaly is detected among regular daily data quality readouts.

- **Configure partition(ed) data quality checks.** Date-partitioned tables are often used in big data platforms at the data ingestion stage. Also, fact tables, clickstream tables, and transaction tables are often partitioned using a date column. DQOps supports calculating separate data quality scores for each daily partition. The data quality results generated by partition checks are stored for every date or month of data. These data quality checks run SQL queries on monitored tables, adding a GROUP BY TRUNCATE(<timestamp_column>) clause, capturing results for every daily or monthly partition. It is also worth mentioning that the table does not need to be physically partitioned by date to benefit from date-partitioned data quality checks. When the table is append-only and has a column that identifies a timestamp of the event, this column can be used to execute daily-partitioned data quality checks. Partition checks help track and check the data quality for daily partitioned data, measuring data quality KPIs at a partition level. They are also useful for analyzing append-only tables, such as fact tables in data warehouses, analyzing financial data when only the most recent data for the current or previous month is important, and older financial records are read-only (closed). Partition checks can also be used to analyze big tables incrementally, scanning only the most recent partitions and avoiding additional pressure on the data source from a data quality platform.

- **Configure data grouping hierarchy.** Tables that aggregate data from multiple data streams should be identified. The columns that identify the data stream should be added to the data grouping hierarchy configuration. DQOps supports the data grouping hierarchy up to 9 levels of nesting. Data grouping hierarchy levels are mapped to columns in the monitored tables or assigned a static value for grouping different tables populated from the same data stream (source). Examples of data grouping hierarchies include country, country/state, continent/country/state, country/department, etc.

- **Configure incremental loading.** Very big tables, often reaching terabyte or petabyte scale, are commonly partitioned by time. It is unlikely that the old data will be updated frequently. Old data in these tables is unlikely to be updated frequently, which makes it possible to reduce query execution time and cost. This can be achieved by including additional filters, such as a WHERE condition applied on the timestamp column, to limit the range of scanned data to the most recent time periods (days, hours, etc.). For date- and time-partitioned tables, it is especially important to execute data quality checks incrementally, since old data does not change and only the most recent data needs to be checked for data quality issues.

- **Schedule the execution of data quality checks.** Data quality checks that are not run at the end of the data loading pipeline should be scheduled internally by the data quality platform or by an external scheduler, such as Apache Airflow or Prefect. It is important to understand the scheduling of the data pipeline that is loading the new data. Data quality checks should be executed at the most convenient time for the new data to be present. The DQOps platform supports setting schedules for an entire connection, table, or individual check.

- **Verify access rights.** This action will help you ensure all the tests are executed correctly. A data quality tool may offer a "dry run" mode that will not store the data quality results in the data quality database.

- **Select appropriate parameters.** Some data quality checks may require providing additional parameters.

- **Make a list of checks that require customization.** Built-in data quality checks may not handle all data quality requirements. A list of data quality requirements that calculate business-relevant metrics must be identified. The DQOps platform makes it easy to create and implement custom data quality checks.

DQOps offers flexibility in defining data quality checks. Figure 3.3 showcases the user interface, where you can visually create and edit checks. For those comfortable with code, Figure 3.4 demonstrates how DQOps supports defining checks within YAML configuration files.

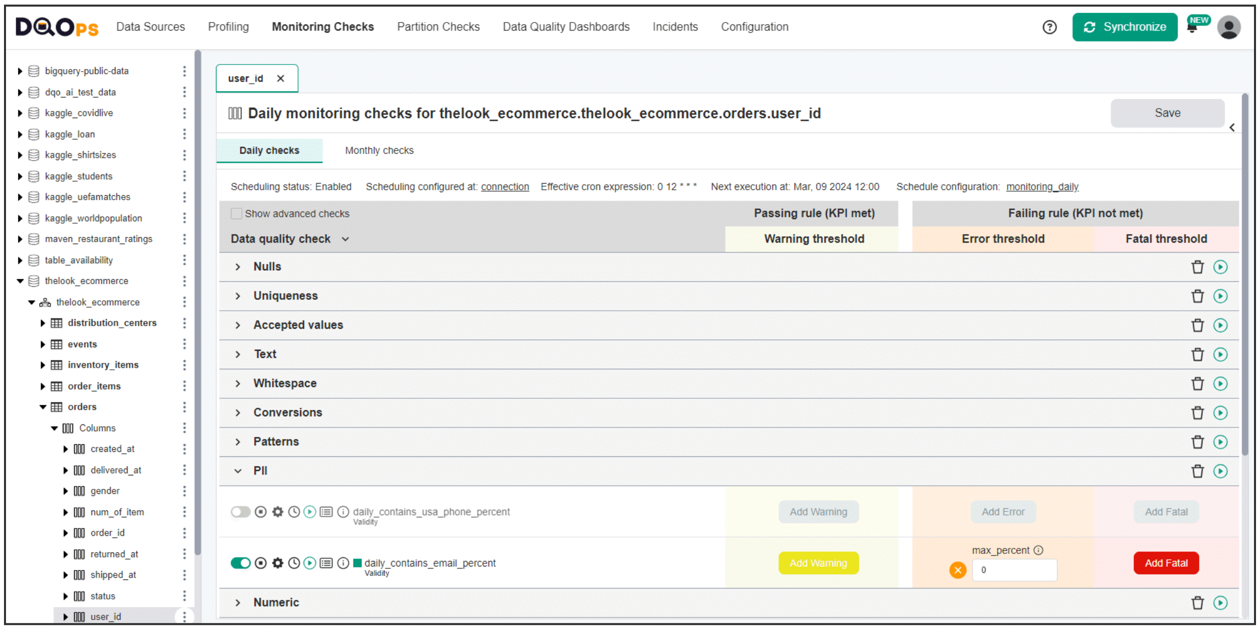
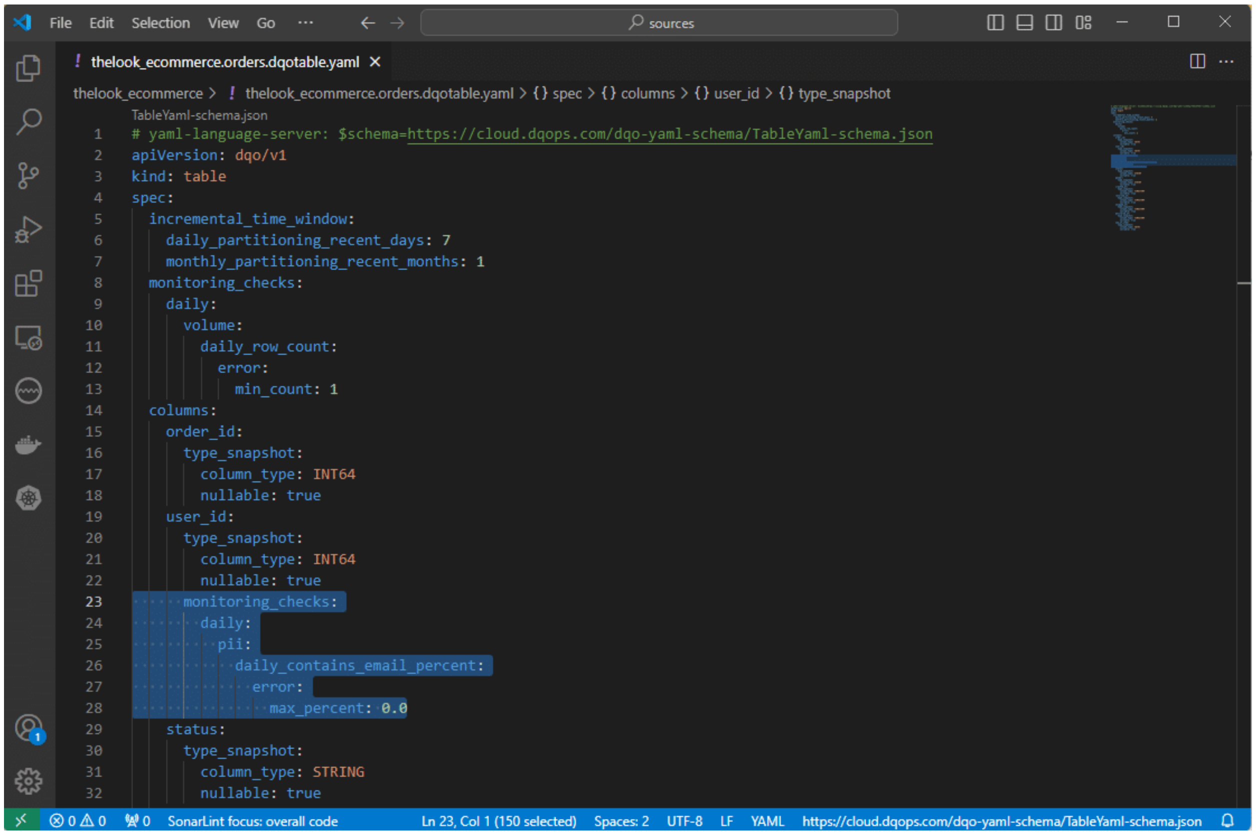


Figure 3.3. A sample screen from the DQOps user interface shows the check editor with an activated column-level daily_contains_email_percent check.


File Edit Selection View Go ···
sources
thelook_ecommerce.orders.dqotable.yaml
thelook_ecommerce > thelook_ecommerce.orders.dqotable.yaml > {} spec > {} columns > {} user_id > {} type_snapshot
TableYaml-schema.json
⊗ 0 ⚠ 0 0 SonarLint focus: overall code Ln 23, Col 1 (150 selected) Spaces: 2 UTF-8 LF YAML https://cloud.dqops.com/dqo-yaml-schema/TableYaml-schema.json

```yaml
# yaml-language-server: $schema=https://cloud.dqops.com/dqo-yaml-schema/TableYaml-schema.json
apiVersion: dqo/v1
kind: table
spec:
  incremental_time_window:
    daily_partitioning_recent_days: 7
    monthly_partitioning_recent_months: 1
  monitoring_checks:
    daily:
      volume:
        daily_row_count:
          error:
            min_count: 1
  columns:
    order_id:
      type_snapshot:
        column_type: INT64
        nullable: true
    user_id:
      type_snapshot:
        column_type: INT64
        nullable: true
      monitoring_checks:
        daily:
          pii:
            daily_contains_email_percent:
              error:
                max_percent: 0.0
    status:
      type_snapshot:
        column_type: STRING
        nullable: true
```

Figure 3.4. A sample screen showing the YAML configuration file from the DQOps with a defined column-level daily_contains_email_percent check. The YAML files in DQOps support code completion in code editors, such as Visual Studio Code. Data quality check definitions can be stored in the source code repository and versioned along with any other data pipeline or machine learning code.

It is worth paying attention to the following problems that may arise at this step.

- The data platform may not have enough computing capacity to execute certain data quality checks. For example, uniqueness checks are particularly computationally expensive because of the necessary preceding sorting step.
- The data files (Parquet, ORC, CSV) behind external tables are corrupted. Those external tables may only be analyzed when a time window and incremental loading are configured together.
- The schedule for executing data pipelines is not fixed.
- The data pipelines take longer than expected to finish, so the most recent data will not be available for a scheduled data quality check.

- Some data quality checks may require custom implementation.

The results of this step are two lists of data quality checks. The first list contains built-in data quality checks, while the second lists custom quality checks that require designing new data quality sensors and rules.

DQ.4. Configure initial alerting thresholds

Complete automation of data quality verification requires creating data quality alerts for all values outside the limits or identified as outliers. Data quality alerts should be created for all remaining data quality results that do not pass the data quality rule evaluation. The previous stage of activating data quality checks described the effect of time slicing and data slicing. Time slicing should enable measuring individual data quality results for each time period (hour, day, week, etc.) separately. Data slicing enables data quality results to be tracked for different data groupings aggregated in the same table. Data quality alerts should be directly linked to data quality readouts, inheriting the time window (the day with the data quality incident) and possible data groupings (identifying the data source).

One more concept should be mentioned here. Data quality alerting is similar to general logging, used in logging libraries. Not all alerts or anomalies are equal, so they should not be investigated and resolved at the same priority. Just as log entries in all logging libraries have a severity level, a data quality alert should also be configured with multiple severity levels.

Each data quality check in DQOps supports setting the alerting thresholds at three severity levels: warning, error, and fatal error. DQOps evaluates the sensor readout (the captured data quality metric, such as a percentage of null values) by three data quality rules, using different alerting thresholds configured as rule parameters. Suppose multiple rules at different severity levels identify a data quality issue (the rule fails). In that case, DQOps picks the severity level of the most severe rule that failed in the following order: fatal error, error, and warning.

The rule severity levels in DQOps are described below.

- **Warning.** Data quality checks with a warning level alerting threshold raise warnings for less important data quality issues, usually anomalies or expected random or seasonal data quality issues. Warnings are not treated as data quality issues. Data quality checks that have not passed the warning alerting rule, but have passed the error and fatal error alerting rules, are still counted as passed data quality checks and do not reduce the data quality KPIs score. Warnings should be used to identify potential data quality issues that should be monitored, but the data producer should not take accountability for them. For example, a percentage of data quality check monitoring null value may raise a warning when the percentage of rows with a null value exceeds 1% of all rows.

- **Error.** This is the default alerting level, comparable to logging libraries' "error" level. Data quality checks that failed to pass the rule evaluation at the "error" severity level are considered failed data quality checks for the purpose of calculating the data quality KPI score. For example, a percentage of data quality check monitoring null value may raise an error when the percentage of rows with a null value exceeds 5% of all rows.

- **Fatal error.** This is the highest alerting threshold that should only be used to identify severe data quality issues. These issues should result in stopping the data pipelines before the issue spreads throughout the system. Fatal error data quality issues are treated as failed data quality checks and reduce the data quality KPI score. The fatal error threshold should be used with caution. It is mainly useful when the data pipeline can trigger the data quality check assessment and wait for the result. If any data quality check raises a fatal error data quality issue, the data pipeline should be stopped. For example, a percentage of data quality check monitoring null value may raise a fatal error alert when the percentage of rows with a null value exceeds 30% of all rows.

Alerting threshold	Data quality check passed	Data quality KPI result is decreased	Data pipeline should be stopped
Warning	✔		
Error (default)		✔	
Fatal error		✔	✔

The process of configuring initial thresholds is described below.

- **Choose the most appropriate type of data quality rule.** The correct data quality rule for assessing the data quality sensor readout should be chosen. It should correctly map to the data quality requirements defined in the earlier steps. It is important to select a rule that makes it easy to justify a data quality incident in case any parties involved in the data ingestion process are unwilling to take responsibility for its resolution. A time-series data quality rule that detects an anomaly may not be as reliable as a simple rule. For example, the table is empty, or the number of rows is 0. In the DQOps platform, all built-in checks have preselected rules. All checks can also be customized.

- **Assign the initial thresholds.** The threshold represents the expectations and beliefs about the current data quality status. The Data Owner or the Data Engineering Team may believe there are no invalid rows, so the rule to count the number of invalid rows should be "equals 0." The correct values will be validated in later steps, and the thresholds should be adjusted to a more reasonable value later. The default alerting thresholds raise data quality issues at the "error" severity level.

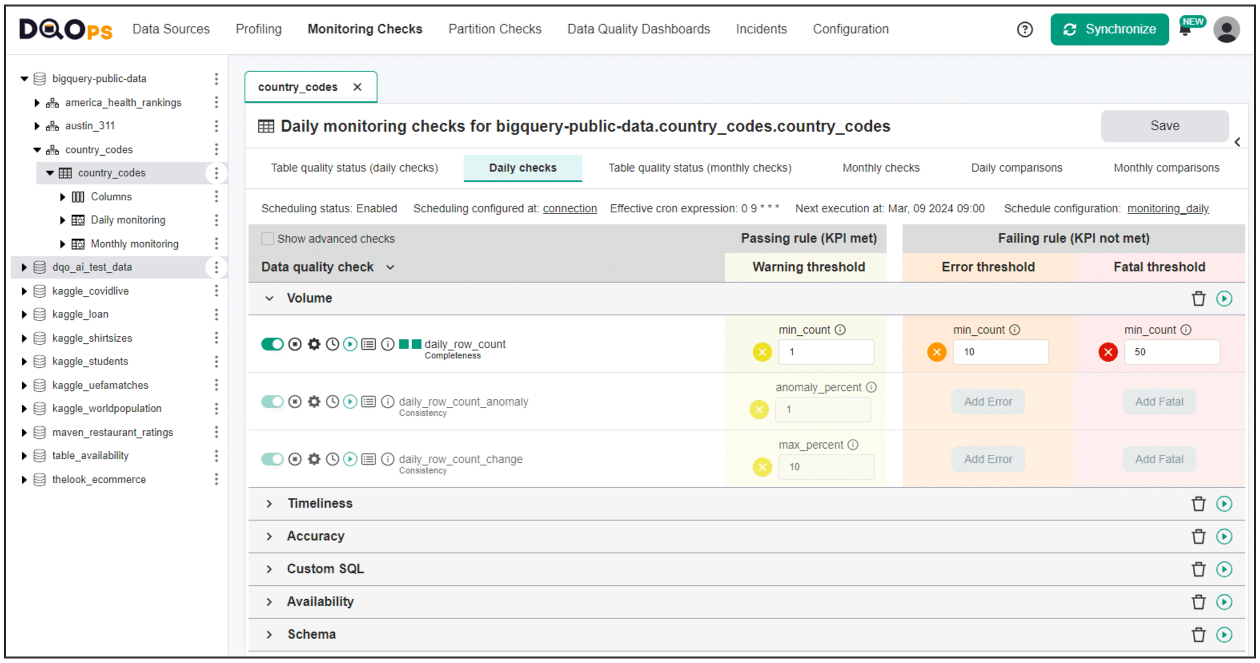
- **Execute data quality rules.** The data quality checks should be evaluated with data quality rules enabled. As a result, errors will be generated for all anomalies.

- **Review the rules with the highest percentage of errors.** In most cases, data quality rules that generate errors for the majority of time periods or date slices may be oversensitive. The variability of the row count increase may not perfectly follow a normal distribution curve, so a rule based on standard deviation will detect false positive errors.

- **Configure the alerting threshold at the warning severity level.** Optionally, an additional alerting threshold can be configured at a warning severity level for possible data quality issues that should be observed. Configuring only a warning severity threshold for a data quality check is possible when the data quality issues should not reflect on the data quality KPI. Warnings should mainly be used to detect anomalies, such as an inconsistent average number of rows loaded daily, as these issues happen occasionally and are not always data quality incidents.

- **Configure the alerting threshold at the fatal error severity level.** Alerting thresholds for the most severe data quality issues, which should result in stopping data pipelines, should be configured carefully. It is worth observing whether the master tables that are replicated to other systems are not empty.

The DQOps platform simplifies managing alert thresholds. It provides a user-friendly interface, shown in Figure 3.5, for easy configuration and modification.



Daily monitoring checks for bigquery-public-data.country_codes.country_codes

Scheduling status: Enabled Scheduling configured at: connection Effective cron expression: 0 9 * * * Next execution at: Mar, 09 2024 09:00 Schedule configuration: monitoring_daily

Data quality check	Passing rule (KPI met)	Failing rule (KPI not met)	
	Warning threshold	Error threshold	Fatal threshold
Volume			
daily_row_count (Completeness)	min_count 1	min_count 10	min_count 50
daily_row_count_anomaly (Consistency)	anomaly_percent 1	Add Error	Add Fatal
daily_row_count_change (Consistency)	max_percent 10	Add Error	Add Fatal

Figure 3.5. A sample screen from DQOps showing table-level daily_row_count check with three severity levels warning, error, and fatal error.

Note the following problems that may occur at this stage.

- Thresholds are configured too conservatively, which generates too many false positive alerts.
- The data variability does not follow a normal distribution curve, requiring tweaking the thresholds at later steps.
- Complex machine learning (ML) rules that use time series analysis to generate alerts are difficult to verify without running an ML model. The external vendor who delivered the data will not feel responsible for the lower quality of the data, since calculating the same rules in an Excel sheet would be impossible.

Data quality alerts raised by the data quality rules should be stored in the data quality database. Time-partitioned data, which uses time slicing to evaluate the data quality checks separately for each time period (e.g., day), generates individual daily alerts. While this is not a problem for a table that is not reloaded or updated, a table that receives late changes or additions will be updated even after a few months. Data quality sensors (especially the row count sensor) will detect new data and reevaluate the data quality rule. The data quality platform should support alert deduplication to avoid generating additional alerts for updated partitions. DQOps uses hashing for detecting significant changes.

Another concept for limiting the number of alerts forwarded to operations teams is to group data quality alerts into clusters of similar alerts with the same properties.

DQ.5. (Optional) Develop and deploy custom data quality checks

Specific data quality requirements influenced by business users may require more complex data quality checks. These data quality checks should be separated and thoroughly analyzed. Below are the most common data quality requirements that should be satisfied by custom data quality checks.

- **Custom data formats.** Column values must follow a complex pattern that is too complex to parse using regular expressions. These usually involve names that must follow a naming convention.

- **Multi-column checks.** These data quality checks perform arithmetic operations across different columns. A simple example is a data quality check that verifies that a net_price + tax = total_price.

- **Cross-table checks.** More complex cross-table checks may perform lookups across related tables.

- **Performance-sensitive queries.** Some data quality checks may require customizations to best use the database's existing indexes.

- **Custom filters.** If you must analyze only a subset of the data, a filter can be configured in the data quality check. Some filters may require joins. For performance reasons, they may require defining a custom data quality check that efficiently performs additional operations.

- **Old data updates.** In some rare cases, it is possible to refresh partitions that are up to a year old with updated data. Such partitions need to be retested, but they are far behind the incremental time window. To prevent executing full table scans every day, a more complex query can be used to access a logging table to detect recently modified daily partitions.

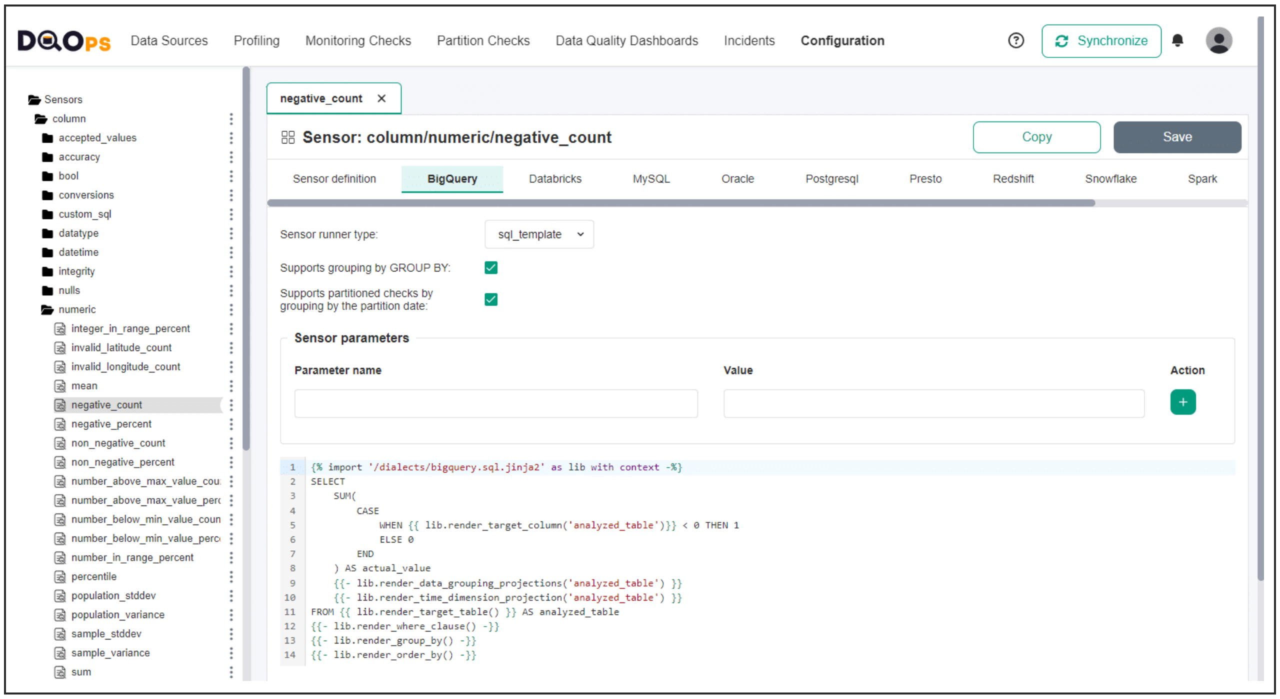
A customizable data quality platform (such as DQOps) should support the use of custom data quality check definitions, provided as custom SQL queries or implemented as custom code.

Custom data quality checks should be reusable across tables and not hard coded for individual tables. To enable the reusability of custom data quality checks, the platform should use a templating engine to define custom data quality sensors. The following example is a template of a data quality sensor that counts rows with a non-negative column value.

```
{% import '/dialects/bigquery.sql.jinja2' as lib with context -%}
SELECT
    SUM(
        CASE
            WHEN {{ lib.render_target_column('analyzed_table')}} < 0 THEN 0
            ELSE 1
        END
    ) AS actual_value
    {{- lib.render_data_grouping_projections('analyzed_table') }}
    {{- lib.render_time_dimension_projection('analyzed_table') }}
FROM {{ lib.render_target_table() }} AS analyzed_table
{{- lib.render_where_clause() -}}
{{- lib.render_group_by() -}}
{{- lib.render_order_by() -}}
```

The template has configurable parts that are dynamically populated with the target table and column names. To render the data quality sensor template into an SQL query, the Jinja2 templating engine is used. Other parts of the query template render the proper GROUP BY clause, additional result columns required to support configurable time series (daily, weekly), and additional data groping slices.

The DQOps platform uses this kind of data quality sensor template. It allows you to design custom data quality checks tailored to your specific needs using a user-friendly interface, as shown in Figure 3.6. Furthermore, DQOps seamlessly integrates your custom checks into the user interface. This makes them readily accessible and usable by everyone on your team, fostering collaboration and a data-driven culture within your organization.



Sensor runner type: sql_template

Supports grouping by GROUP BY: ✓

Supports partitioned checks by grouping by the partition date: ✓

```
{% import '/dialects/bigquery.sql.jinja2' as lib with context -%}
SELECT
    SUM(
        CASE
            WHEN {{ lib.render_target_column('analyzed_table')}} < 0 THEN 1
            ELSE 0
        END
    ) AS actual_value
    {{- lib.render_data_grouping_projections('analyzed_table') }}
    {{- lib.render_time_dimension_projection('analyzed_table') }}
FROM {{ lib.render_target_table() }} AS analyzed_table
{{- lib.render_where_clause() -}}
{{- lib.render_group_by() -}}
{{- lib.render_order_by() -}}
```

Figure 3.6. A sample screen from the DQOps platform showing the sensor definition screen for the negative_count sensor. The sensor query template, written in the Jinja2, can be modified.

In the process of defining and testing a custom data quality sensor, follow these steps:

- **Write an SQL query to pull metrics.** Implement a prototype of the custom data quality sensor by writing an SQL query that can retrieve the correct metrics. At this step, the performance implications of executing the query are easily identified.
- **Customize the SQL query to the query template.** The SQL query should be customized to the data quality sensor template by replacing the hard-coded table and column names with placeholders. Additional placeholders should also be added to support configurable time slicing and data groupings.
- **Register a definition of the custom data quality sensor.** The custom quality sensor template should be registered in the platform. In DQOps, for example, custom data quality sensors can simply be added as text files in the "sensors" folder.
- **Verify the implementation of the data quality sensor.** The newly defined data quality sensor should be attached to a tested table and executed in a dry-run mode without saving the data quality results.

- **Activate the custom data quality sensor on the target tables.** Once the sensor implementation is verified, it can be connected to the target tables.

After connecting custom data quality sensors (template SQL queries), data quality alert thresholds should be configured. A properly implemented custom data quality check should return a measure that can be evaluated by reusing data quality rules, which, in most cases, are sufficient.

More complex data quality checks may separate some parts of a complete data quality check into a custom rule function. DQOps hands over the evaluation of alerting rules to Python functions. These Python functions (data quality alerting rules) are called with three groups of arguments:

- The current value of the data quality sensor that is being evaluated.
- An array of historical data quality sensor readouts for the requested time window before the time of the period being evaluated.
- Additional data quality rule parameters to enable an additional level of configuration.

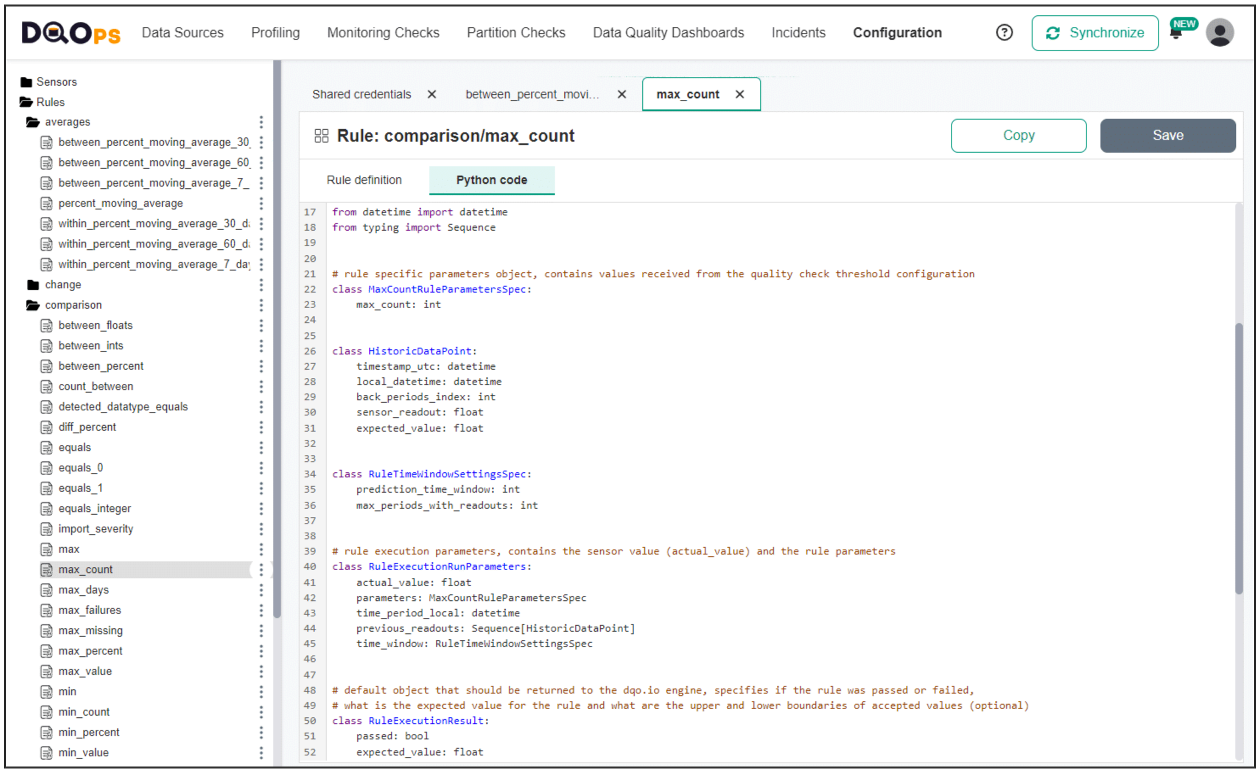
Below is an example of a simple data quality rule that compares the current sensor value with a minimum value:

```
def evaluate_rule(rule_parameters: RuleExecutionRunParameters) ->
RuleExecutionResult:
    if not hasattr(rule_parameters, 'actual_value'):
        return RuleExecutionResult()

    expected_value = rule_parameters.parameters.min_value
    lower_bound = rule_parameters.parameters.min_value
    upper_bound = None
    passed = rule_parameters.actual_value >= lower_bound

    return RuleExecutionResult(passed, expected_value, lower_bound,
    upper_bound)
```

DQOps empowers you to take complete control over your data quality rules. As illustrated in Figure 3.7, the user interface provides a user-friendly environment for both modifying existing rules and creating entirely new ones.



```python
from datetime import datetime
from typing import Sequence


# rule specific parameters object, contains values received from the quality check threshold configuration
class MaxCountRuleParametersSpec:
    max_count: int


class HistoricDataPoint:
    timestamp_utc: datetime
    local_datetime: datetime
    back_periods_index: int
    sensor_readout: float
    expected_value: float


class RuleTimeWindowSettingsSpec:
    prediction_time_window: int
    max_periods_with_readouts: int


# rule execution parameters, contains the sensor value (actual_value) and the rule parameters
class RuleExecutionRunParameters:
    actual_value: float
    parameters: MaxCountRuleParametersSpec
    time_period_local: datetime
    previous_readouts: Sequence[HistoricDataPoint]
    time_window: RuleTimeWindowSettingsSpec


# default object that should be returned to the dqo.io engine, specifies if the rule was passed or failed,
# what is the expected value for the rule and what are the upper and lower boundaries of accepted values (optional)
class RuleExecutionResult:
    passed: bool
    expected_value: float
```

Figure 3.7. A *sample screen from the DQOps platform showing a rule definition screen for the min_value sensor. The rule source code, written in Python, can be modified.*

Note the problems that can happen at this stage.

- Data quality requirements may require writing very complex SQL queries.
- Complicated tests may require combining several tables and writing complex SQL queries with multiple join statements. For partitioned data, additional data partitioning filters may be required for referenced tables.
- Performing custom tests may require extensive collaboration with the Data Engineering Team.

At this stage, we increase the number of test definitions and add custom data quality checks to the list of supported data quality checks.

DO.5. Provide a list of KPIs to be monitored

Data quality sensors capture quality-related metrics from monitored data sources. These sensor readouts should be evaluated by data quality rules to detect outliers or measures that do not meet the required thresholds. In DQOps, the combination of the data quality sensor and data quality rule is called a data quality check. An executed data quality check has two possible statuses: passed or failed. For long-term data quality monitoring, the data quality platform must measure the percentage of passed data quality checks within all executed data quality checks. This percentage of passed data quality checks is called a data quality KPI.

DQOps stores the result of executed data quality rules for both passed (no alert raised or only a warning raised) and failed (errors or fatal errors alerts raised) data quality check evaluations. Data quality checks can define the alerting threshold at three severity levels: warning, error, and fatal error. The final alert raised by the data quality check evaluation reflects the most severe level for which the threshold has been met.

Data quality KPIs can be aggregated at multiple levels, providing ways to measure the data quality for time periods (days, weeks, months, etc.), data quality dimensions, data streams (such as by country), or any combination of these grouping levels.

The expected result of calculating the data quality KPI at different grouping levels may look like the following tables:

Data quality KPIs at a day level.

Date	KPI value
2022-10-01	95.1%
2022-10-02	96.2%
2022-10-03	94.5%
2022-10-04	94.7%

Data quality KPIs at a day and data quality dimension level.

Date	Timelines	Completeness	Validity
2022-10-01	96.1%	97.4%	95.1%
2022-10-02	99.2%	94.6%	96.2%
2022-10-03	94.6%	97.0%	94.3%
2022-10-04	99.1%	93.2%	94.7%

Additionally, data quality KPIs can be calculated for different data groupings separately. Data aggregated in a single database (or a data lake) can be loaded from different data sources. To calculate a separate data quality KPI for each data source, it must be possible to identify that source at the data level. There are two ways to identify the data source in DQOps:

- **Separate tables for each data source.** This is a simple case that can be solved by tagging the table with the name of the data source, external vendor, or department. A data quality KPI can be calculated from multiple tables at once. In DQOps, such a configuration is provided as a tag value assigned to a data grouping level. Here is an example of the data grouping configuration in a YAML file:

```
apiVersion: dqo/v1
kind: table
spec:
    incremental_time_window:
       daily_partitioning_recent_days: 7
       monthly_partitioning_recent_months: 1
    default_grouping_name: by_supplier
    groupings:
      by_country:
        level_1:
          source: tag
          tag: UK
```

- **Multiple data sources aggregated into a single table.** Data from multiple sources can be aggregated in a single table. If there is a column that identifies the data source, it can be used to assign the generated alerts and sensor readouts to the correct data grouping. Here is another example of a DQOps YAML file that uses a "country" column to identify separate data groupings for separate data quality KPI calculation:

```yaml
apiVersion: dqo/v1
kind: table
spec:
    incremental_time_window:
       daily_partitioning_recent_days: 7
       monthly_partitioning_recent_months: 1
    default_grouping_name: by_supplier
    groupings:
      by_country:
        level_1:
          source: column_value
          column: country
```

Data quality KPIs can also be calculated for combinations of data sources (data groupings), time periods, and data quality dimensions. An example output of a data quality KPI calculation at a month, country-level data sources, and separate data quality dimensions would look like the following table:

Month	Data source	Timelines	Completeness	Validity
2022-10	US	96.1%	97.4%	95.1%
	UK	99.2%	94.6%	96.2%
	FR	94.6%	97.0%	94.3%
	JP	99.1%	93.2%	94.7%

The Data Quality Team and the Data Owner should agree on the selection of valid data quality KPI aggregations that will simplify further discovery of the root causes of data quality issues. The following aggregations should be discussed:

- **Time dimensions.** Tracking data quality KPIs at the monthly level allows you to compare the current month's data quality KPI with those of the previous month. Any changes to the data engineering process or improvements in the quality of the source data should be visible at this scale.

- **Data sources with separate tables.** Identify groups of related tables that are populated from the same data source. This level of aggregation allows you to track data quality at the data source level.

- **Data sources aggregated in shared tables.** Find discriminator columns that identify the data source, supplier, country, business unit, brand, or market.

- **Tables with the same purpose.** Tables can also be grouped by their purpose. All tables that are part of a single data mart can be grouped. Data quality KPIs should also be tracked separately for fact and dimension tables.

- **Data received from external vendors.** External vendors, business partners, or suppliers may share data imported into a data warehouse or data lake. The data-sharing agreement may include very specific KPIs for data sharing. These data-sharing KPIs are, in fact, data quality KPIs related to the latency of the data exchange, which is the timeliness data quality dimension. The completeness of the data received from external business partners should also be monitored to detect missing data due to outdated credentials or corrupted files. If the data sharing agreement describes the data format, the receiving party can verify the field format as validity data quality checks.

- **Data shared with external vendors.** If an organization shares (exports) data with its business partners, it should track the data quality of the exported data. This enables the organization to meet all data quality requirements in the data-sharing agreements.

- **KPIs aggregated by product or service line.** A data lake or data warehouse can aggregate all corporate data from different organizational units, business divisions, or subsidiaries. Data quality KPIs should be linked to a particular line of business.

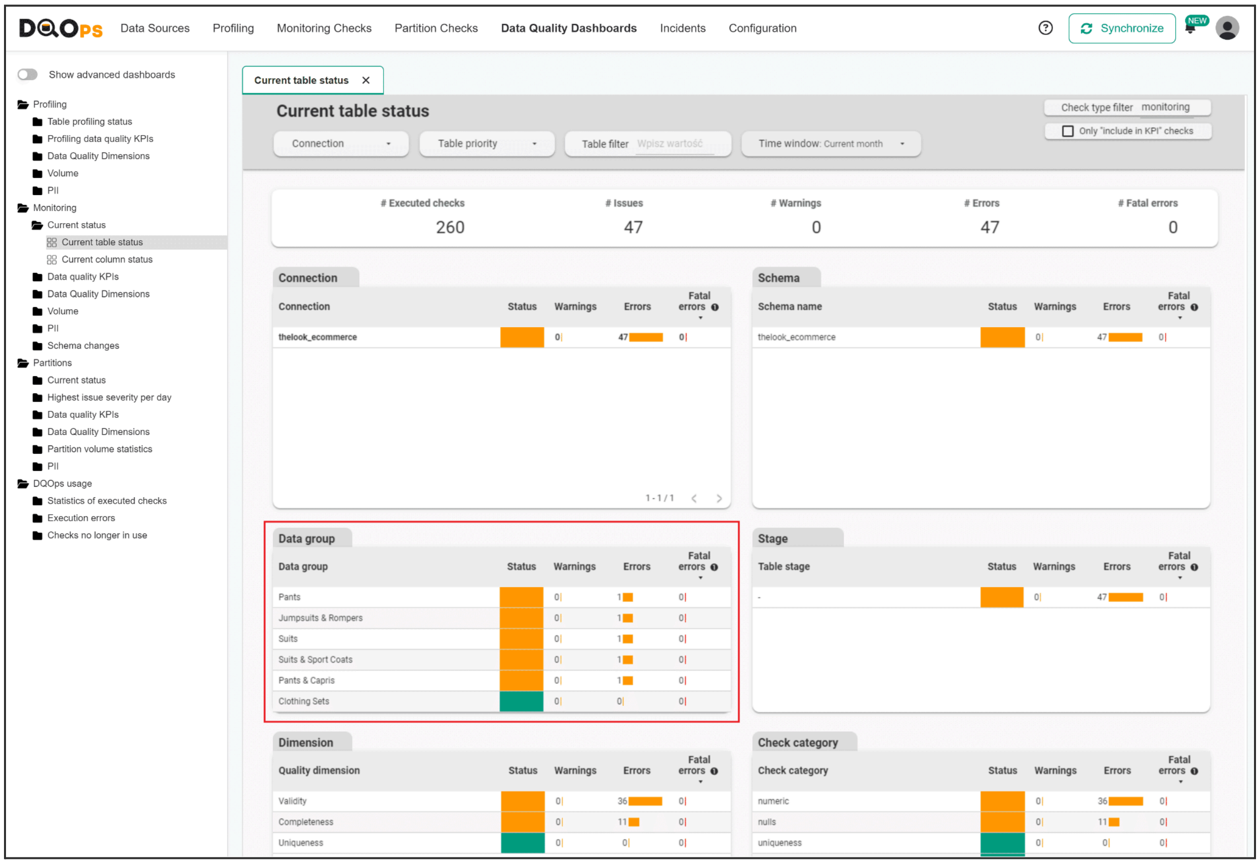
- **Internal cross-departmental KPIs.** Data quality KPIs can be calculated separately for different departments involved in the entire data lineage. From the perspective of the department responsible for the data lake, data quality KPIs for upstream data sources should be separated from the data quality KPIs of downstream data delivered to other departments.

- **Multiple copies of the same data.** The same data can be stored in different databases and data warehouses. The data quality KPIs of the original data in an OLTP database can be compared with the data quality of the copy stored in the data lake or data warehouse.

- **Data and cloud migration.** A successful data migration project should prove that the data quality of the target (migrated) database is the same as that of the source database (the old decommissioned database). Similar tables in the old and new databases should be tagged with the same data grouping label. All tables in the old database should be tagged as "old," and all new ones should be tagged as "new."

- **Data mesh.** Data may be distributed across different data lakes. It is crucial to track the data quality for each data lake. Data quality should also be measured for the data that is exchanged between the mesh nodes.

The DQOps platform provides dashboards that help analyze data quality from multiple perspectives, enabling you to effectively identify root causes. These dashboards offer numerous aggregations, including time dimensions, connections, data quality dimensions, priorities, stages, and grouping. For instance, Figure 3.8 illustrates the "Current table status" dashboard with data group aggregation by product categories.



Current table status

Connection · Table priority · Table filter Wpisz wartość · Time window: Current month

Check type filter monitoring

☐ Only "include in KPI" checks

# Executed checks	# Issues	# Warnings	# Errors	# Fatal errors
260	47	0	47	0

Connection

Connection	Status	Warnings	Errors	Fatal errors
thelook_ecommerce	🟧	0	47	0

Schema

Schema name	Status	Warnings	Errors	Fatal errors
thelook_ecommerce	🟧	0	47	0

Data group

Data group	Status	Warnings	Errors	Fatal errors
Pants	🟧	0	1	0
Jumpsuits & Rompers	🟧	0	1	0
Suits	🟧	0	1	0
Suits & Sport Coats	🟧	0	1	0
Pants & Capris	🟧	0	1	0
Clothing Sets	🟩	0	0	0

Stage

Table stage	Status	Warnings	Errors	Fatal errors
-	🟧	0	47	0

Dimension

Quality dimension	Status	Warnings	Errors	Fatal errors
Validity	🟧	0	36	0
Completeness	🟧	0	11	0
Uniqueness	🟩	0	0	0

Check category

Check category	Status	Warnings	Errors	Fatal errors
numeric	🟧	0	36	0
nulls	🟧	0	11	0
uniqueness	🟩	0	0	0

Figure 3.8. A sample screen of the "Current table status dashboard" from DQOps shows the grouping of data quality issues by different product categories.

Note the problems that may happen at this stage.

- A discriminator column that identifies different data streams has many distinct values that generate many aggregations. For example, calculating data quality KPIs at the "city" level may generate too many aggregations.
- The database model lacks documentation, so table groups do not make sense.
- The database model changes frequently, and keeping the data quality KPIs up-to-date with the current data model is difficult.

The data quality KPIs identified in this step will be used when designing data quality dashboards. The design of the data quality dashboard is described later in this guide.

DE.6. Provide a list of metrics to monitor the data pipeline

Data quality should also be measured from a data engineering perspective. Bugs or failures in data pipelines or ETL processes can cause many data quality issues. A data quality platform can identify these issues because most failures will be noticed as anomalies in time series analysis.

The most typical issues with data pipelines are easy to detect with the following data quality dimensions:

Data pipeline issue	Data quality dimension	Data quality check
The data pipeline fails to execute because the credentials are outdated.	Timeliness	The data lag (the difference between the current timestamp and the highest timestamp in the database) rises above the threshold. No new data is loaded because the data pipeline has stopped working. This issue can be detected by a data freshness check.
Data pipelines do not start at the same time of the day.	Timeliness	Some data pipelines have to wait until other pipelines have finished their work. The waiting time may vary daily, and it can get very long on some days. Tracking the ingestion delay and comparing it with the average delay detects these issues.

Data pipeline issue	Data quality dimension	Data quality check
The data pipeline failed or was canceled during execution.	Consistency	The average number of rows loaded per day drops below the average.
The data pipeline scheduled daily was canceled during execution.	Completeness	Month or week completeness checks detect days with missing data.
The incremental loading pipeline loaded the same data multiple times.	Uniqueness	The number of duplicate values, especially identifiers, indicates that some data was loaded multiple times.
The incremental loading pipeline missed some rows.	Accuracy	The number of rows per day between the source table and the target table identifies that some rows are missing. This is especially true if the data quality check is executed 1-2 days later when all late-coming data should have been already loaded.
Not all rows have been loaded from the staging table to the data warehouse.	Accuracy	Comparing the number of rows between the staging table and the target (cleansing, data vault, etc.) tables indicates a mismatch.
The column format has changed.	Validity	Validity data quality checks should be configured for each column that is required to follow a particular format. These data quality checks work best on staging tables with all columns defined as a text data type. The data quality checks can try to parse the column values or verify that columns match particular date formats.
The number of columns in the file has changed.	Consistency	Tracking changes in the number of columns identify the issue. The current number of columns (retrieved from the table's metadata) must be compared to the last known number of columns.

Data pipeline issue	**Data quality dimension**	**Data quality check**
The order of columns has changed in the file.	Consistency	Similar to detecting that the number of columns has changed, calculating the hash code from the names of all columns helps identify the issues.
The column order has changed.	Consistency	Often, the source data is retrieved using a simple SQL SELECT statement with a list of columns or just "*" to retrieve all columns. If the data pipeline is sensitive to column reordering and cannot depend on named columns, statistical analysis is required to detect issues. A data quality check that monitors the number of distinct values in a column can instantly detect that the number of unique values has changed since the last data quality check evaluation. The number of distinct values in two columns that have been reversed changes significantly from day to day.
The table schema has changed.	Consistency	The hash code of all column names and their data types can be tracked. Any changes to the hash will instantly identify a change.
Rows were rejected because column values were out of range for the target data types.	Validity	Staging tables should be tested using "value in range" data quality checks. Numeric and decimal fields with limited scale and precision cannot accept values that do not match their respective format; e.g., the DECIMAL(8, 2) column does not accept values with a format different than 123456.78. Tables in the staging area should be tested periodically.
Out-of-memory errors.	Completeness, Timeliness	Random failures of the data quality pipeline affect timeliness (data loaded on the next execution) or completeness (data never loaded), so there are some missing days.

Data pipeline issue	Data quality dimension	Data quality check
Disk space issues.	Completeness	In most cases, if the data for a particular date is too large to fit on a disk, the data for that day is dropped. Completeness data quality checks detect the gaps in the data.

The Data Quality Team and the Data Engineering Team, preparing a list of data quality KPIs, should focus on the following topics:

- **Gather a list of frequent data pipeline issues.** Obtain a list of common data pipeline issues and map it to a list of supported data quality checks that can predict these issues.
- **Identify the most critical issues that must be detected.** The list of data pipeline issues must be prioritized to avoid flooding the Data Engineering Team with alerts that notify about less critical issues.
- **Define mapping of existing data quality checks to reporting categories.** Existing data quality checks run using different data quality tools (such as dbt or Great Expectations) should be imported into the data quality database. It is important to map them to a different stage or group them into data groupings. These valuable data quality checks should be measured against expected data quality KPI levels.
- **Define KPIs for existing logs.** The frequency of warnings or errors reported to the log management platform should be measured and aggregated into a global data quality KPI score. All such logs should be identified, and the acceptable number of errors and alerts must be defined.
- **Prepare a mapping between data pipelines and tables.** To calculate aggregated data quality KPIs at a database or data pipeline level, tables referenced by a single complex data pipeline or a complex ETL job must have the same data grouping levels assigned to them. It is also helpful to assign source and target tables to different areas. In that case, a separate data quality KPI can be measured for the source tables used by a single data pipeline or an ETL job. An increase in the number of alerts on the source table would help identify affected data pipelines or ETL jobs that should be paused until the data quality issue in the source tables is resolved. DQOps uses a "stage" value at the table level to map tables to a common stage.

- **Define the expected data quality KPIs at the data pipeline level.** The data quality KPI value at the data pipeline level is a percentage of passed data quality checks for the stage, data quality dimension, and data grouping). Examples of data quality KPIs monitored from a data pipeline perspective:

  - Percentage of fresh tables on the ingestion stage, calculated by measuring the timeliness sensors for all tables assigned to the "ingestion" stage.

  - Percentage of tables loaded without any delay, measured by counting alerts raised when tables were not updated on time.

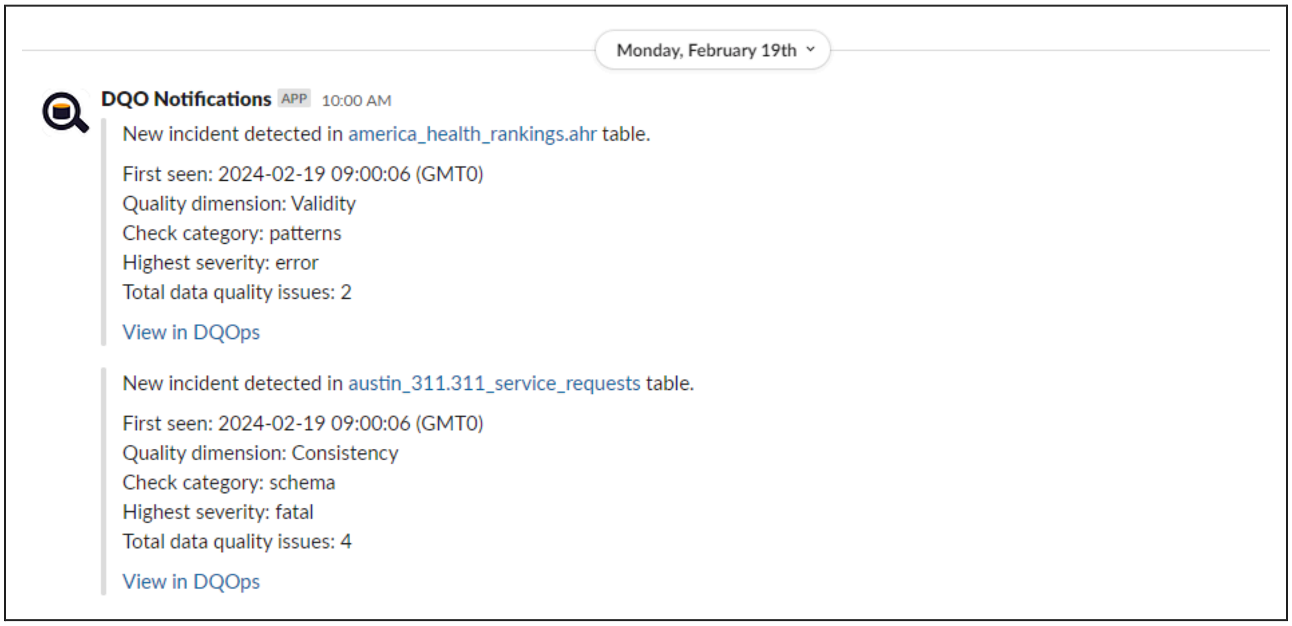
- **Agree on accepted KPI levels.** Ideally, all data quality KPIs (percentage of passed data quality checks) should be 100%, but this is not always possible. Adaptive data quality checks that use machine learning, time series analysis, anomaly detection, or simple statistical analysis (different from the standard deviation) generate false positive alerts that should be treated as warnings about a possible issue. The expected percentage of such alerts should be agreed upon with the Data Engineering Team.

- **Configure an alert notification channel.** The Data Engineering Team may require frequent or even real-time notifications about identified data quality issues. Popular notification channels include email and Slack. DQOps platform supports integration with Slack webhooks which are used to set up in-app Slack notifications, as shown in Figure 3.9.

- **Integrate with a ticketing system.** A ticketing system such as Jira or ServiceNow may already be in use by the Data Engineering Team. A ticket should be opened when a certain data quality issue is identified in a ticketing system. A list of such high-severity alerts must be identified with the Data Engineering Team.

- **Decide on the frequency of notifications.** Not all alerts should be immediately published on the notification channel or result in opening a new ticket. For some types of alerts (especially anomalies), it is advisable to delay the notification. Subsequent data quality issues might indicate a broader problem behind all of them, which can only be detected by introducing a notification delay, after which a collective notification shall be sent. On the other hand, notifications may be set to be raised only when the data quality KPI drops below a certain threshold. In that case, if one of the 20 tables in the ingestion stage is delayed, the data quality KPI for timeliness in the ingestion stage will simply drop to 95%. A drop of that KPI to or below 90% indicates that at least two tables in the ingestion stage are delayed, so the Data Engineering Team should be notified.

- **Prepare requirements for data quality KPI dashboards from the data pipeline perspective.** Data quality KPIs should be displayed on data quality dashboards. The Data Engineering Team should provide its requirements for data quality dashboards that could help them identify the root cause of issues or predict data pipeline failures. Requirements for data quality dashboards should focus on KPIs grouped by stages, schemas, tables, data pipelines, data areas, and data quality dimensions. Additional filtering on the dashboards should also be discussed. DQOps has many built-in dashboards that allow tracking and reviewing issues from the data pipeline perspective, such as schema changes, timeliness, table availability, validity, and completeness. All dashboards can be customized.


Monday, February 19th

DQO Notifications APP 10:00 AM

New incident detected in america_health_rankings.ahr table.

First seen: 2024-02-19 09:00:06 (GMT0)
Quality dimension: Validity
Check category: patterns
Highest severity: error
Total data quality issues: 2

View in DQOps

New incident detected in austin_311.311_service_requests table.

First seen: 2024-02-19 09:00:06 (GMT0)
Quality dimension: Consistency
Check category: schema
Highest severity: fatal
Total data quality issues: 4

View in DQOps

Figure 3.9. An example of DQOps notification in Slack. DQOps supports integration with Slack webhooks which are used to set up in-app Slack notifications.

The problems that may occur at this stage:

- The recent list of data pipelines is missing.
- The complexity of data pipelines or ETL processes makes it difficult to determine which tables are used by them.

- The organization lacks knowledge of the inner workings of some of the old ETL processes because the employees responsible for them have left.
- Integration with notification channels or ticketing systems requires extensive work and approval from platform owners.

DQ.6. Create data quality KPI dashboards

After gathering the lists of requested data quality KPIs requirements from the Data Owner and the Data Engineering Team, the Data Quality Team can design and build the data quality dashboards.

Depending on the audience and purpose of data quality dashboards, they can be divided into three groups: Governance, Operational, and Detailed.

Governance dashboards

Governance data quality dashboards offer senior management a clear and comprehensive picture of their organization's data health. These dashboards condense key data quality indicators (KPIs) into a high-level, macro view, quickly grasping overall data quality.

The dashboards cater to different levels of detail. Senior management can review a concise summary or analyze KPIs by individual time periods, data sources, schemas, tables, columns, data quality dimensions, check categories, data groups, and even specific days. The dashboards should also offer the ability to filter data by various criteria. This granular filtering allows for pinpointing areas needing improvement.

An example of the Governance dashboard “KPIs scorecard summary” available in the DQOps platform is shown in Figure 3.10.

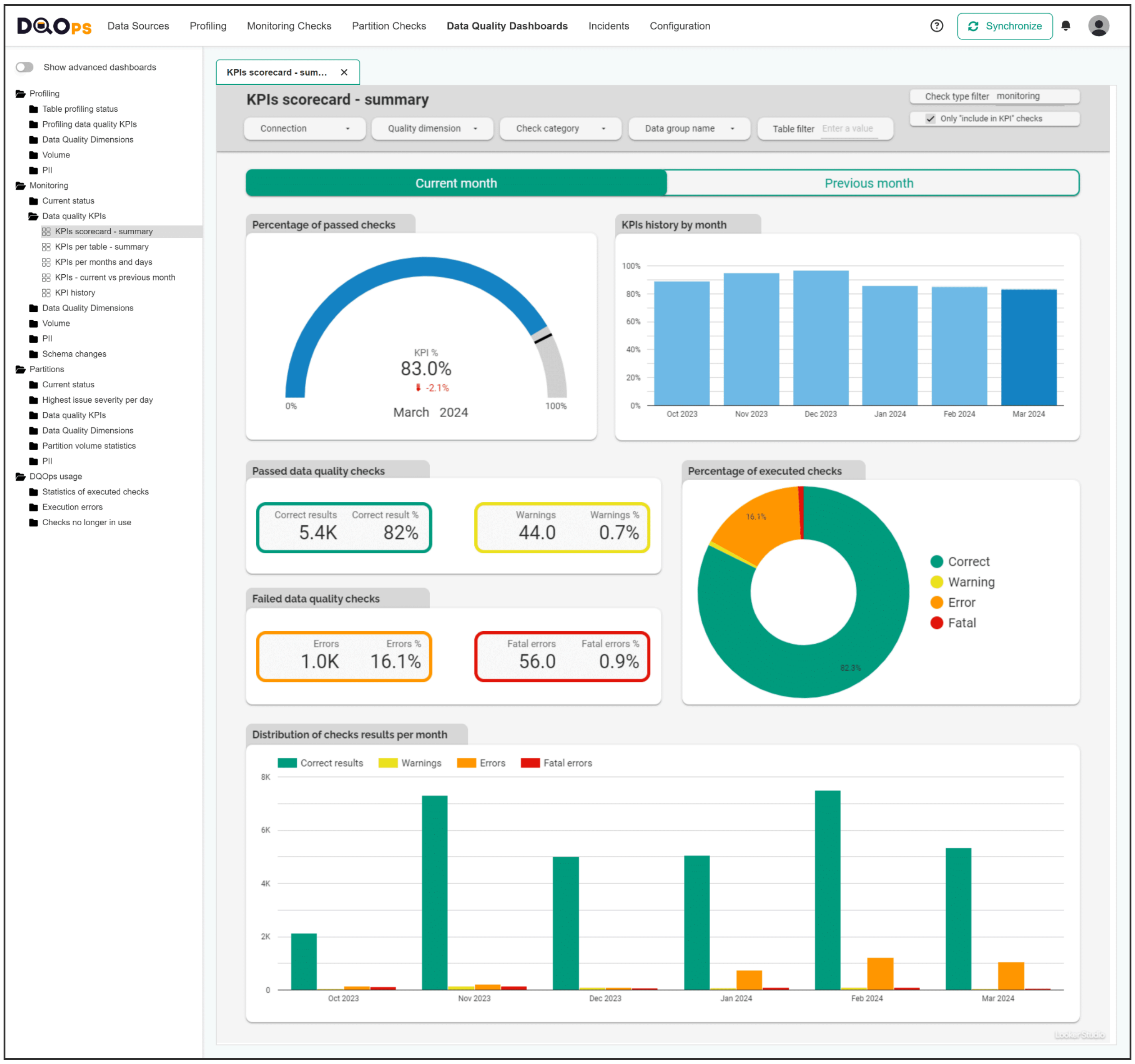


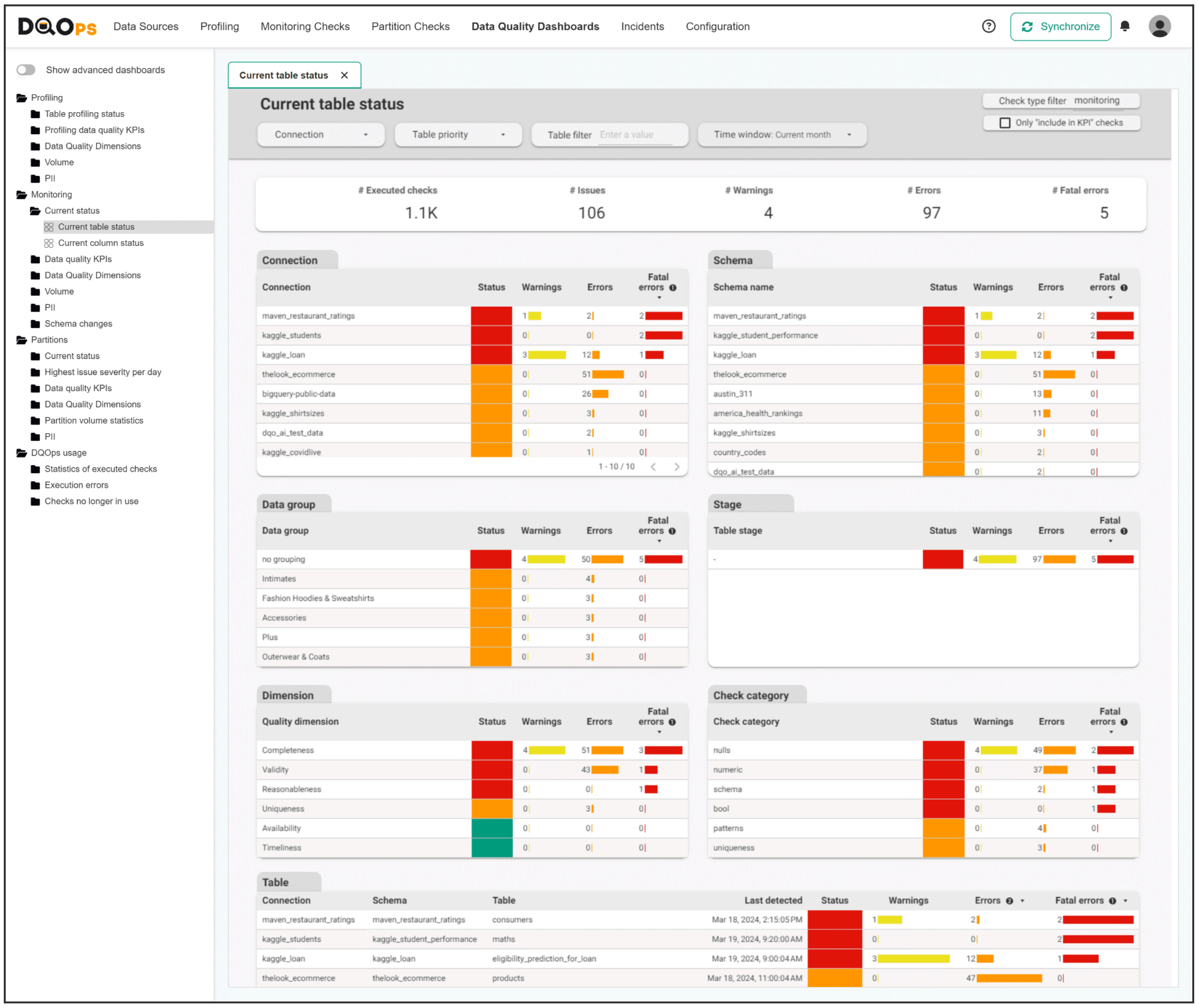
Figure 3.10. A sample screen of the governance dashboard "KPIs scorecard - summary," which is available on the DQOps Platform.

Operational dashboards

Operational data quality dashboards are designed for Data Engineers and Data Owners, providing insights into the health of data pipelines and data warehouses/lakes. These dashboards pinpoint the areas needing attention by highlighting tables or pipelines with the most frequent data quality issues. The Data Quality Team plays a vital role in this process by helping identify and eliminate false positive alerts.

Operational dashboards go beyond a high-level view. Teams can leverage granular filtering options to delve deeper into specific aspects of data quality. Filters encompass time frames, connections, schemas, data groups, stages, data quality dimensions, check categories, and individual tables and columns. This level of detail allows for pinpointing the root causes of issues and facilitates targeted solutions.

The operational dashboards available in the DQOps platform, such as the one shown in Figure 3.11, utilize a color-coded system to represent the severity level of detected data quality issues. Green indicates no issues, while yellow, orange, and red signify progressively more critical problems. This visual approach quickly identifies the most concerning tables and columns. Additionally, the dashboards display the precise number of issues per severity level, further aiding in prioritization.



Check type filter: monitoring

# Executed checks	# Issues	# Warnings	# Errors	# Fatal errors
1.1K	106	4	97	5

Connection

Connection	Status	Warnings	Errors	Fatal errors
maven_restaurant_ratings	🟥	1	2	2
kaggle_students	🟥	0	0	2
kaggle_loan	🟥	3	12	1
thelook_ecommerce	🟧	0	51	0
bigquery-public-data	🟧	0	26	0
kaggle_shirtsizes	🟧	0	3	0
dqo_ai_test_data	🟧	0	2	0
kaggle_covidlive	🟧	0	1	0

Schema

Schema name	Status	Warnings	Errors	Fatal errors
maven_restaurant_ratings	🟥	1	2	2
kaggle_student_performance	🟥	0	0	2
kaggle_loan	🟥	3	12	1
thelook_ecommerce	🟧	0	51	0
austin_311	🟧	0	13	0
america_health_rankings	🟧	0	11	0
kaggle_shirtsizes	🟧	0	3	0
country_codes	🟧	0	2	0
dqo_ai_test_data	🟧	0	2	0

Data group

Data group	Status	Warnings	Errors	Fatal errors
no grouping	🟥	4	50	5
Intimates	🟧	0	4	0
Fashion Hoodies & Sweatshirts	🟧	0	3	0
Accessories	🟧	0	3	0
Plus	🟧	0	3	0
Outerwear & Coats	🟧	0	3	0

Stage

Table stage	Status	Warnings	Errors	Fatal errors
-	🟥	4	97	5

Dimension

Quality dimension	Status	Warnings	Errors	Fatal errors
Completeness	🟥	4	51	3
Validity	🟥	0	43	1
Reasonableness	🟥	0	0	1
Uniqueness	🟧	0	3	0
Availability	🟩	0	0	0
Timeliness	🟩	0	0	0

Check category

Check category	Status	Warnings	Errors	Fatal errors
nulls	🟥	4	49	2
numeric	🟥	0	37	1
schema	🟥	0	2	1
bool	🟥	0	0	1
patterns	🟧	0	4	0
uniqueness	🟧	0	3	0

Table

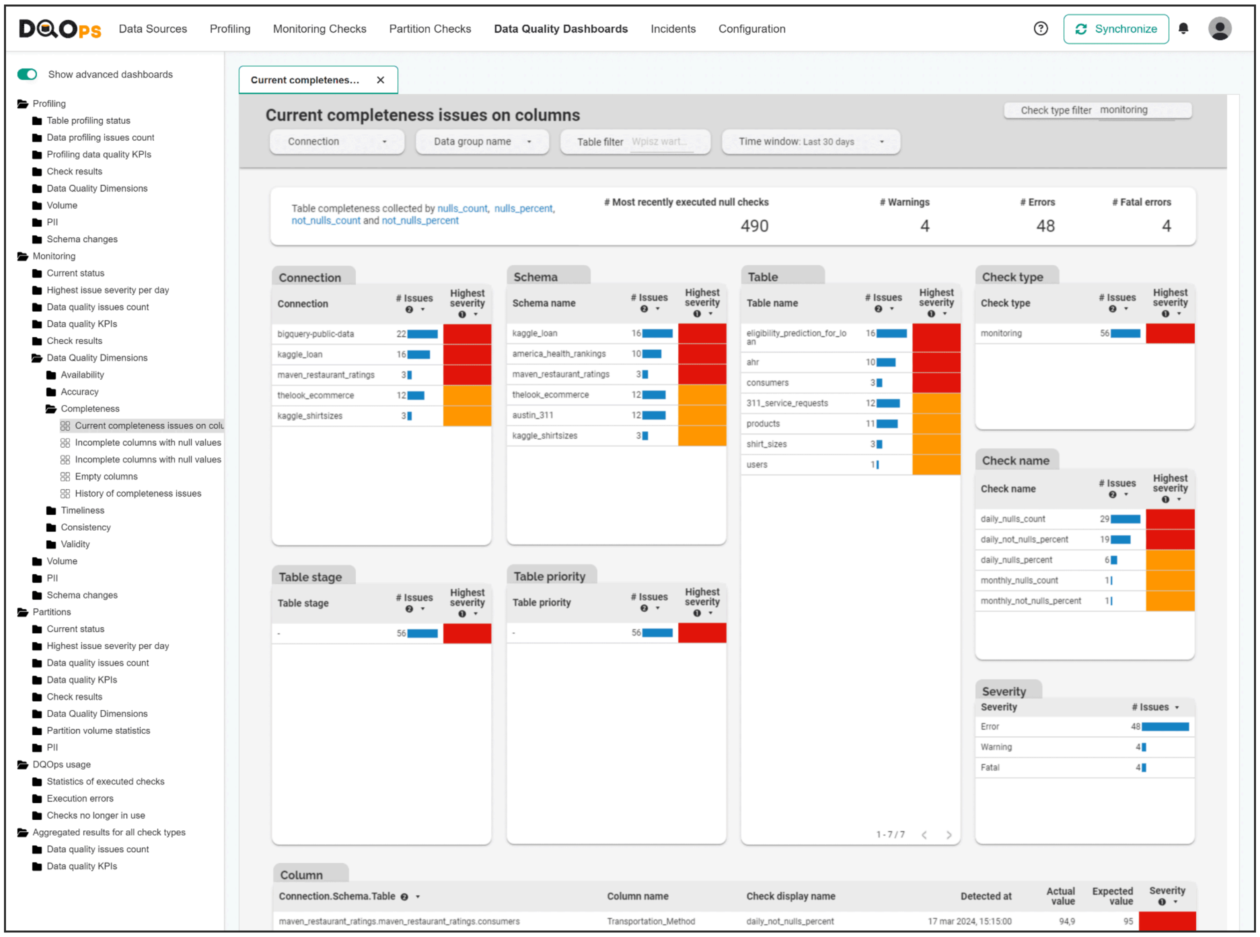
Connection	Schema	Table	Last detected	Status	Warnings	Errors	Fatal errors
maven_restaurant_ratings	maven_restaurant_ratings	consumers	Mar 18, 2024, 2:15:05 PM	🟥	1	2	2
kaggle_students	kaggle_student_performance	maths	Mar 19, 2024, 9:20:00 AM	🟥	0	0	2
kaggle_loan	kaggle_loan	eligibility_prediction_for_loan	Mar 19, 2024, 9:00:04 AM	🟥	3	12	1
thelook_ecommerce	thelook_ecommerce	products	Mar 18, 2024, 11:00:04 AM	🟧	0	47	0

Figure 3.11. A sample screen of an operational dashboard "Current table status," which is available on the DQOps Platform.

Detailed dashboards

Detailed data quality dashboards should provide a more detailed view for Data Engineers and Data Owners. This view will allow them to inspect data at the table and column level, which can be extremely helpful during the investigation phase of data quality issues. This deep-dive capability will enable them to diagnose problems effectively. Once a data quality issue has been addressed, these dashboards can be used to confirm its successful resolution.

DQOps platform offers a wide range of detailed dashboards catering to specific data quality dimensions such as availability, completeness (an example shown in Figure 3.12), timeliness, consistency, and validity. Moreover, other detailed dashboards provide detailed information on areas of interest such as schema changes, volume, and personally identifiable information. All DQOps dashboards offer a vast range of filtering options within each dashboard. You can drill down into specific timeframes, connections, schemas, data groups, tables, stages, priorities, tables, and individual columns.



Current completeness issues on columns

Check type filter: monitoring

Table completeness collected by nulls_count, nulls_percent, not_nulls_count and not_nulls_percent

# Most recently executed null checks	# Warnings	# Errors	# Fatal errors
490	4	48	4

Connection

Connection	# Issues	Highest severity
bigquery-public-data	22	🟥
kaggle_loan	16	🟥
maven_restaurant_ratings	3	🟥
thelook_ecommerce	12	🟧
kaggle_shirtsizes	3	🟧

Schema

Schema name	# Issues	Highest severity
kaggle_loan	16	🟥
america_health_rankings	10	🟥
maven_restaurant_ratings	3	🟥
thelook_ecommerce	12	🟧
austin_311	12	🟧
kaggle_shirtsizes	3	🟧

Table

Table name	# Issues	Highest severity
eligibility_prediction_for_loan	16	🟥
ahr	10	🟥
consumers	3	🟥
311_service_requests	12	🟧
products	11	🟧
shirt_sizes	3	🟧
users	1	🟧

Check type

Check type	# Issues	Highest severity
monitoring	56	🟥

Check name

Check name	# Issues	Highest severity
daily_nulls_count	29	🟥
daily_not_nulls_percent	19	🟥
daily_nulls_percent	6	🟧
monthly_nulls_count	1	🟧
monthly_not_nulls_percent	1	🟧

Table stage

Table stage	# Issues	Highest severity
-	56	🟥

Table priority

Table priority	# Issues	Highest severity
-	56	🟥

Severity

Severity	# Issues
Error	48
Warning	4
Fatal	4

Column

Connection.Schema.Table	Column name	Check display name	Detected at	Actual value	Expected value	Severity
maven_restaurant_ratings.maven_restaurant_ratings.consumers	Transportation_Method	daily_not_nulls_percent	17 mar 2024, 15:15:00	94,9	95	🟥

Figure 3.12. A sample screen of a detailed dashboard "Current completeness issues on columns," which is available on the DQOps Platform.

Design and development of data quality dashboards

Below are the main features of a good data quality dashboard:

- Clear, logical layout and ease of understanding.
- Enables data export to an Excel file when information about a data quality issue needs to be shared with additional people or external partners who do not have access to the dashboard.
- It allows a relatively easy comparison between different time periods, such as the current and previous month.
- Enables filtering for the time periods.
- Allows users to drill down through data streams, stages, data suppliers (business partners, etc.), and data pipelines.
- Enables calculation of grand totals for KPIs, i.e., the percentage of passed data quality checks for all data quality dimensions.

It is also worth mentioning that a cumulative data quality KPI may be misleading without being split into different data quality dimensions (timeliness, validity, etc.). The number of active data quality checks in each data quality dimension may differ. The cumulative data quality KPI can use additional formulas that calculate a weighted KPI across different data quality dimensions. The formula for the cumulative (weighted) data quality KPI may use different weights for each data quality dimension. An example of such a calculation is presented in the table below. This example of weighted calculation of KPIs can be applied to the customized data quality dashboards available on the DQOps platform.

Data quality dimension	Timeliness		Validity		Accuracy		Weighted KPI
KPI per dimension	95%		80%		90%		
	*		*		*		
Weight	20%		50%		30%		
	=		=		=		
Weighted KPI	19%	+	40%	+	30%	=	89%

The cost of the business intelligence (BI) platform used to present the data quality dashboards should not be neglected. The data engineers and some Data Owners may not have a license for a BI platform. Moreover, additional specialists who may be involved in the incident resolution must be granted access to data quality dashboards, which may require covering additional licensing fees.

DQOps uses Looker Studio (formerly Google Data Studio) to present data quality dashboards. All data quality results, i.e., data quality sensor readouts used on the detailed dashboards and data quality alerts aggregated on the governance and operational dashboards, are synchronized to a private data quality data warehouse in the Google Cloud. Looker Studio is used to show data quality dashboards because there is no per-user license fee, which allows granting access to all parties involved in the issue resolution process.

The following tasks must be completed during the design and development of data quality dashboards:

- **Build a data quality warehouse.** The data quality results used to create data quality dashboards must be aggregated in a database. In case a custom data quality database is used, it may be a significant effort beyond the scope of just one step in the process because data modeling and data ingestion activities must be performed. For simplicity, we assume that the data quality platform provides the data quality warehouse. With DQOps, each customer receives a private data quality lakehouse in Google Cloud (Figure 3.13).
- **Select a business intelligence environment.** You can choose from various commercially available BI technologies, such as Sisense, Tableau, Power BI, or Looker Studio. Data quality projects implemented with DQOps will receive a complementary Looker Studio instance connected to a data quality data warehouse.
- **Connect the business intelligence platform to the data quality database.** The business intelligence tool must also be connected to the data quality database. DQOps customers can ask the vendor to access a custom Looker Studio data source, which provides direct access to the data quality lakehouse.
- **Select relevant data groupings for aggregation of data quality KPIs.** The relevant data grouping hierarchy levels that identify data sources, data streams, vendors, business partners, subsidiaries, or data pipelines must be selected.
- **Design dashboards for governance, operational, and detailed dashboards.** Custom data quality dashboards should be developed according to the Agile process. For each requested data quality dashboard, the development process should involve the requirement review, mockup preparation, mockup review, development, and testing.

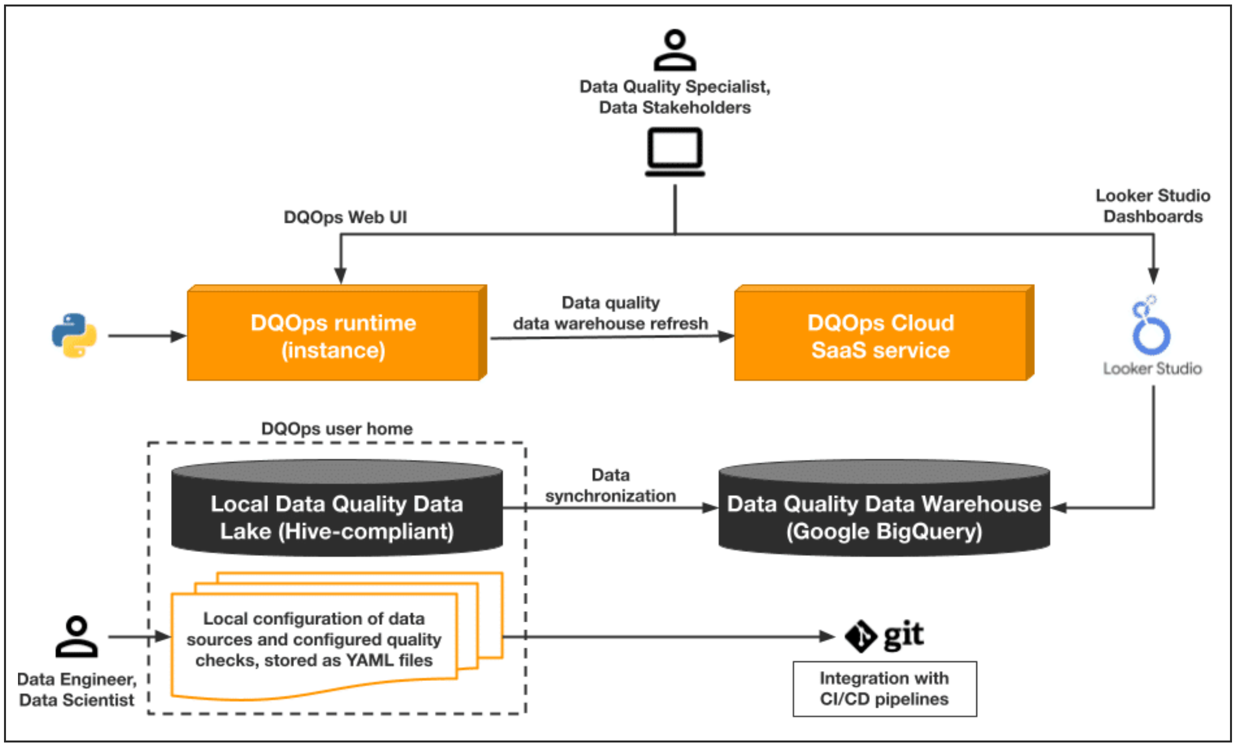


Figure 3.13. Data quality projects implemented with DQOps receive a complementary Looker Studio instance connected to a Data Quality Data Warehouse on Google Cloud. Learn more about DQOps architecture in the documentation.

Note the problems that may arise at this stage.

- The cost of a business intelligence platform can be significant.
- The data quality database must be redesigned, or additional tables must be created.
- Too much data displayed on the dashboard creates visual clutter.

II. Improving data quality KPIs

Once the first data quality dashboards are set up, the Data Quality Team can start monitoring the data quality KPIs. If the KPIs' scores are not acceptable, the team identifies the checks responsible for the alerts and creates a list of tables affected by the issue. The Data Quality Team then contacts the Data Owner, who can review the root cause of the issue. If the problem can be resolved in the source system or by an external data provider, the Data Quality Team can re-execute data quality checks and continue monitoring. An issue that the Data Owner cannot resolve because it occurred in the data pipeline or an ETL process must be verified by the Data Engineering Team.

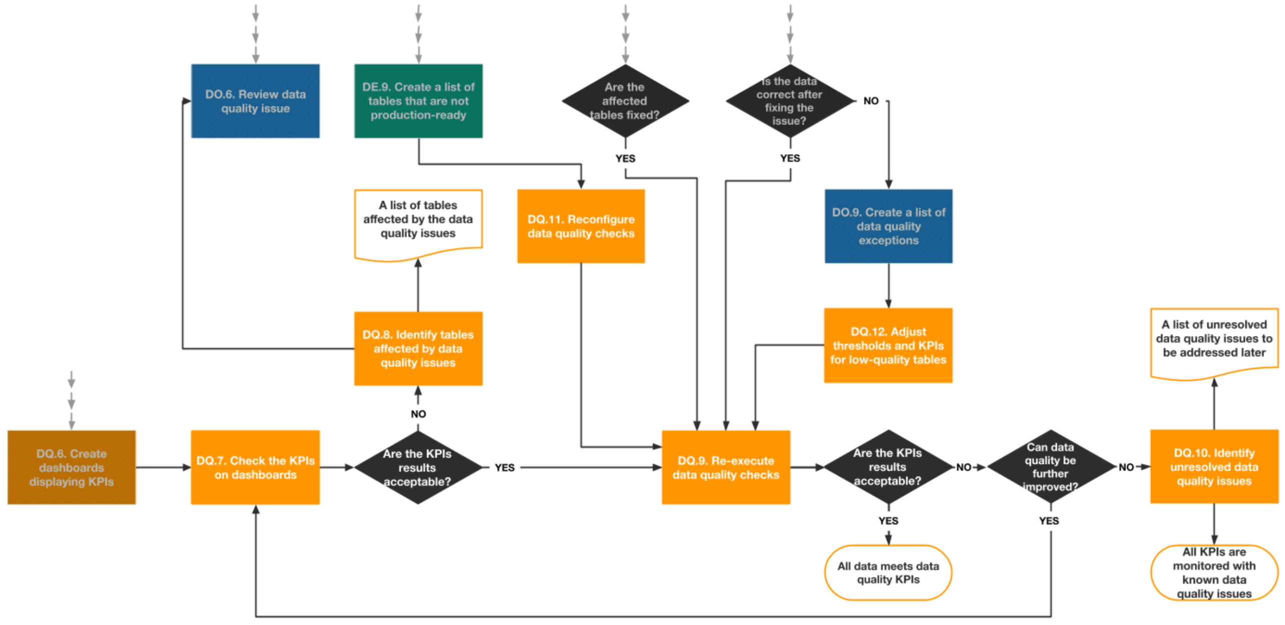
Sometimes, a data quality issue cannot be resolved promptly, or its resolution requires a manual update of invalid records. For example, a data quality requirement that the phone number is provided for every customer in the CRM cannot be easily fixed without engaging the sales team in a lengthy CRM data cleaning. In this case, the Data Owner creates a list of tolerated data quality issues. Next, the Data Quality Team adjusts the KPIs' rules and thresholds for affected tables and re-executes data quality checks. If the Data Quality Team still identifies unresolved data quality issues, they create a list of issues to be addressed at a later stage. At the end of the process, all data should meet expected data quality KPIs or a list of exceptions should be created. The remaining data quality issues should be monitored and fixed as soon as the right conditions are met, under which data quality issues can be resolved.

4. Optimizing data quality scores

The active data quality checks will raise alerts whenever the alerting thresholds are exceeded. Initially, the majority of the alerts at this stage will be false positives because the alerting thresholds are defined based on the expectations of Data Owners. At this stage, false positive alerts should be identified, and their alerting thresholds should be adjusted to the actual data quality. The remaining data quality alerts indicate actual data quality issues that may be resolved by the Data Owner, Data Producer, or Data Engineering Team.

The Data Quality Team monitors data quality KPIs on newly created data quality dashboards. Any identified data quality issues must be reviewed with the Data Owner, who should take responsibility for the next steps. The Data Producer can fix the data quality issues present at the data source level. Issues caused by a bug in the data pipeline or an ETL process should be fixed by the Data Engineering Team. Once the problem is resolved, the Data Quality Team re-executes data quality checks.

If the issue cannot be fixed immediately, the Data Quality Team may adjust the alerting thresholds or acceptable levels of data quality KPIs. This happens when there are unfixable data quality issues in the source data. The purpose of conducting the data quality process is to measure the percentage of issues (such as a percentage of null values). The Data Quality Team should investigate further if the data quality KPI deteriorates over time.


DO.6. Review data quality issue
DE.9. Create a list of tables that are not production-ready
Are the affected tables fixed?
Is the data correct after fixing the issue?
NO
YES
YES
A list of tables affected by the data quality issues
DQ.11. Reconfigure data quality checks
DO.9. Create a list of data quality exceptions
DQ.8. Identify tables affected by data quality issues
DQ.12. Adjust thresholds and KPIs for low-quality tables
A list of unresolved data quality issues to be addressed later
NO
DQ.6. Create dashboards displaying KPIs
DQ.7. Check the KPIs on dashboards
Are the KPIs results acceptable?
YES
DQ.9. Re-execute data quality checks
Are the KPIs results acceptable?
NO
Can data quality be further improved?
NO
DQ.10. Identify unresolved data quality issues
YES
YES
All data meets data quality KPIs
All KPIs are monitored with known data quality issues

DQ.7. Check the KPIs on dashboards

The data quality dashboards help to identify areas with data quality issues that require attention. As we mentioned in the DQ.6. Create data quality KPI dashboards chapter, the data quality dashboards can divided into three groups:

- **Governance dashboards.** These dashboards display global data quality KPIs categorized by the data quality dimension and the total percentage of passed data quality checks out of the total executed checks. During the data quality KPI review, only the governance dashboards are important. The issues that need to be addressed are prioritized in the operational dashboard.

- **Operational dashboards.** These dashboards provide a prioritized list of tables or columns affected by data quality issues. These are the tables that the Data Quality Team should focus on with the highest priority.

- **Detailed dashboards.** These dashboards facilitate an in-depth investigation of data quality issues by reviewing the historical data quality sensor readouts or dimension-specific data quality issues, such as availability, completeness, timeliness, consistency, or validity — for example, a sharp increase in the percentage of rows with null values.

Governance dashboards may be divided into data areas, measuring data quality KPIs associated with processing stages, vendors, external data suppliers, data marts, or even separate data streams aggregated in a single table. DQOps enables the use of up to 9 data grouping hierarchy levels to tag and segment the data quality results by tags or column values (such as a country column).

The Data Quality Team should review data quality KPIs with the Data Owner in the respective data area.

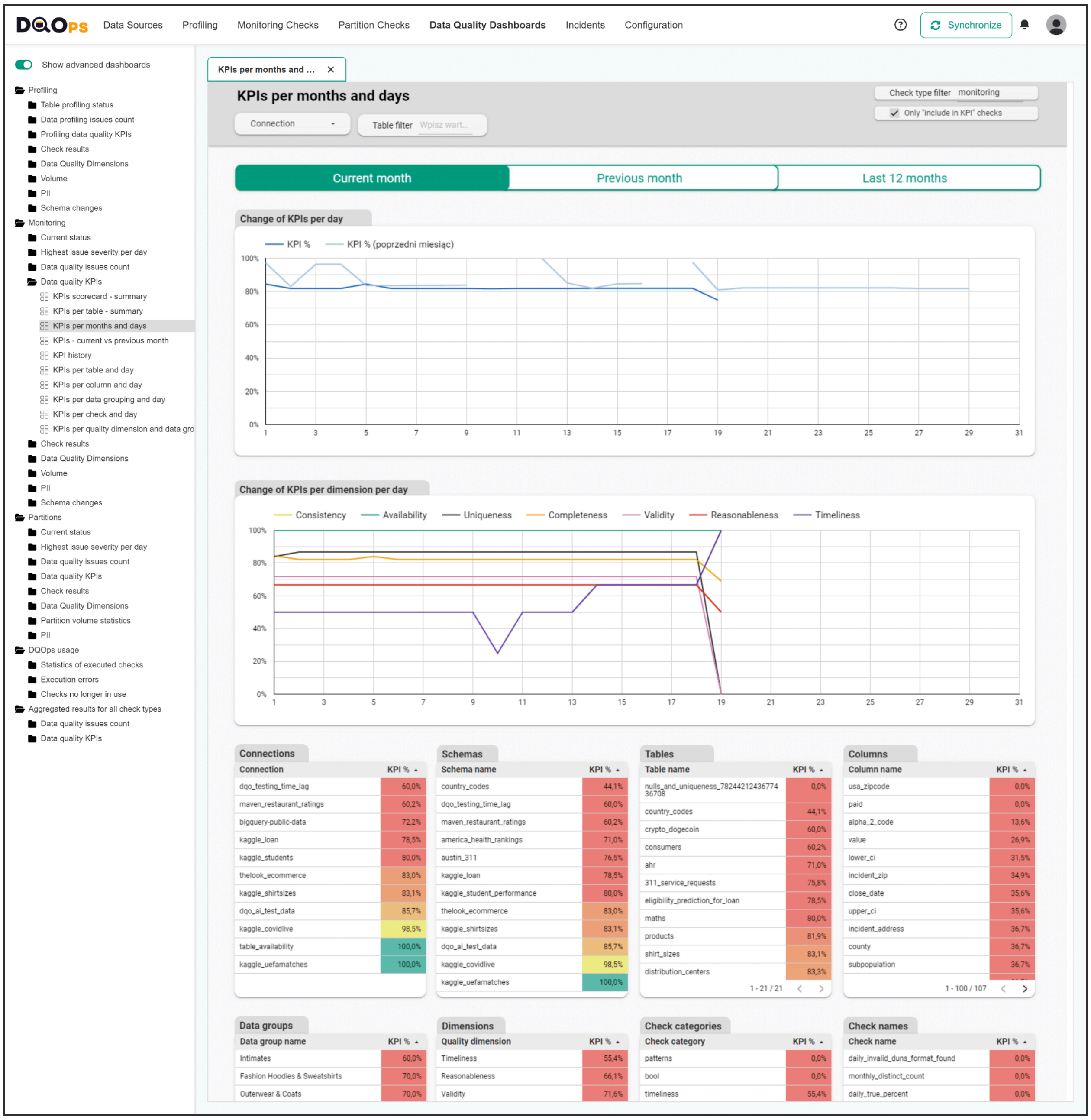
To complete this stage, pay attention to the steps outlined below.

- **Identify data quality dimensions with unmet KPIs.** Governance dashboards should show exactly which data quality dimensions have issues. Issues may be related to missing data (timeliness dimension) or invalid values (validity dimension). Identify the most important data quality dimension affected by data quality issues that must be addressed.

- **Identify data areas with unmet KPIs.** Once you have identified data quality dimensions with a high percentage of alerts, review the data quality KPIs at a deeper level. This level is a set of data quality KPI dashboards that show data quality KPIs at a data source, stage, vendor, or database level. The data quality dashboard available in the DQOps platform, shown in Figure 4.1, prioritizes data areas that do not meet data quality KPIs and should be fixed.

- **Assess acceptable KPI levels.** Acceptable KPI levels may already be agreed upon with the Data Owner for data sources whose quality is not satisfactory. The Data Owner may already know that up to 5% of tables may not be refreshed on time for some acceptable reasons. In that case, the accepted timeliness KPI would be 95%. The Data Quality Team may choose another area for an in-depth analysis if the governance dashboards show a slightly higher KPI.

- **Prioritize the affected data areas.** Identify vendors, stages, databases, or any other data area that does not meet data quality KPIs and should be fixed.



Connections

Connection	KPI %
dqo_testing_time_lag	60,0%
maven_restaurant_ratings	60,2%
bigquery-public-data	72,2%
kaggle_loan	78,5%
kaggle_students	80,0%
thelook_ecommerce	83,0%
kaggle_shirtsizes	83,1%
dqo_ai_test_data	85,7%
kaggle_covidlive	98,5%
table_availability	100,0%
kaggle_uefamatches	100,0%

Schemas

Schema name	KPI %
country_codes	44,1%
dqo_testing_time_lag	60,0%
maven_restaurant_ratings	60,2%
america_health_rankings	71,0%
austin_311	76,5%
kaggle_loan	78,5%
kaggle_student_performance	80,0%
thelook_ecommerce	83,0%
kaggle_shirtsizes	83,1%
dqo_ai_test_data	85,7%
kaggle_covidlive	98,5%
kaggle_uefamatches	100,0%

Tables

Table name	KPI %
nulls_and_uniqueness_7824421243677436708	0,0%
country_codes	44,1%
crypto_dogecoin	60,0%
consumers	60,2%
ahr	71,0%
311_service_requests	75,8%
eligibility_prediction_for_loan	78,5%
maths	80,0%
products	81,9%
shirt_sizes	83,1%
distribution_centers	83,3%

1 - 21 / 21

Columns

Column name	KPI %
usa_zipcode	0,0%
paid	0,0%
alpha_2_code	13,6%
value	26,9%
lower_ci	31,5%
incident_zip	34,9%
close_date	35,6%
upper_ci	35,6%
incident_address	36,7%
county	36,7%
subpopulation	36,7%

1 - 100 / 107

Data groups

Data group name	KPI %
Intimates	60,0%
Fashion Hoodies & Sweatshirts	70,0%
Outerwear & Coats	70,0%

Dimensions

Quality dimension	KPI %
Timeliness	55,4%
Reasonableness	66,1%
Validity	71,6%

Check categories

Check category	KPI %
patterns	0,0%
bool	0,0%
timeliness	55,4%

Check names

Check name	KPI %
daily_invalid_duns_format_found	0,0%
monthly_distinct_count	0,0%
daily_true_percent	0,0%

Figure 4.1. The sample screen of a governance dashboard "KPIs per months and days," which is available on the DQOps Platform. This dashboard helps prioritize data areas that do not meet data quality KPIs and should be fixed.

Note the following problems that may occur at this stage.

- Additional data quality dashboards must be created for selected data stream hierarchies. For example, the data quality results are measured at the country and state levels. For these hierarchies, three types of data quality KPI dashboards are possible: country/state level KPIs, country-level KPIs, and an overall organization-wide KPI.

- Incomplete data (such as missing days) affects the KPI calculation because the KPI is calculated based on too few data quality results in a given time period (month).

Once the data quality KPI review is finished, the Data Quality Team should decide if any data quality KPIs are not met. The affected tables will be identified in the next stage before the problem is reviewed with the Data Owner or the Data Engineering Team.

DQ.8. Identify tables affected by data quality issues

Once the area of focus has been identified on the governance dashboards, the Data Quality Team can use the operational and detailed dashboards to find all the tables that have generated the highest number of data quality issues. The Data Quality Team identifies the most severe data quality issues based on the priority of the affected tables and should work together with the Data Owner, Data Producer, and Data Engineering Team to resolve the issue.

The data quality KPI is a percentage of passed data quality checks. Often, alerting thresholds are configured too restrictively, resulting in low KPI scores. For example, the alerting threshold for the minimum number of new rows inserted into a table per day might be set too high. Such data quality checks can be replaced with ones that compare the daily row count increase to the average daily growth rate.

To complete this stage, pay attention to the steps outlined below.

- **Select one data quality KPI to improve.** The Data Quality Team should start by selecting a data quality KPI to improve a single data area. This could be improving the timeliness of tables at a single stage or the validity of tables from a single data source.

- **Prioritize the tables affected by the issues.** Sort the tables affected by data quality issues by the number of issues. The Data Quality Team should review three separate lists of affected tables, sorted by the number of data quality issues at the fatal error, error, and warning severity levels.

- **Review fatal error issues.** First, you need to review the tables affected by data quality issues at the fatal error severity level. These are the most significant data quality issues that affect business users or issues that can propagate down the data pipelines, spreading across the organization. A master table that stores a list of customers cannot be empty if replicated to downstream systems. When the Data Quality Team identifies any fatal error data quality incidents, the Data Owner should be informed, and the Data Engineering Team can be asked to stop the affected data pipeline to avoid spreading the data quality issue to other systems.

- **Review regular data quality errors.** The default alerting severity level is "error." Sort tables affected by regular data quality issues by the number of errors to focus on the most affected tables first.
- **Review data quality warnings.** Review alerting thresholds that trigger warnings for anomalies. A recent change to the percentage of rows with null values may suggest a potential data quality issue. If not addressed preemptively, this issue could cause more severe problems in the future.
- **Make a preliminary assessment of the data quality issue.** Review in detail the data quality alerts raised in the tables under investigation. DQOps data quality sensors capture historical data quality readouts. Reviewing historical values, such as row counts, can help find the root cause of the issue.
- **Review potential issues caused by misconfiguration of data quality checks.** When data quality checks have been activated recently, there are likely some obvious configuration issues. Most of these issues can be fixed instantly by changing the configuration and re-executing the data quality checks without involving other parties.
- **Review the data directly in the monitored table.** To understand a problem, the Data Quality Team should query the monitored table directly by running SQL queries or reviewing the files for external tables. Sometimes, a data quality issue is easy to notice on a query result screen after looking at the table's contents.
- **Prepare an issue summary for review with the Data Owner or Data Engineering Team.** The data owner, data producer, or data engineering teams should fix data quality issues that are not false positives. They should also review frequent data quality issues that happen from time to time. Occasional data quality issues are easily identified because they lower the data quality KPI. The Data Quality Team should prepare a report summarizing the investigation's results.
- **Contact the Data Owner and Data Engineering Teams.** At the end of this stage, the Data Quality Team should engage with teams that can fix the data quality issue at the source level or the data processing stage.

Data quality incident automation

Monitoring data quality in data sources using the DQOps platform will detect new data quality issues at regular intervals, so the same issues will be detected until the root cause is fixed. Additionally, DQOps supports hundreds of data quality checks, which can result in many positive and false-positive check results. Some data quality issues are expected, as data quality checks are not disabled on a decommissioned table. Planned maintenance events can cause other issues. To prevent overwhelming the support team with numerous data quality issues, DQOps groups similar issues into data quality incidents.

A data quality issue is a single check result that does not meet the data quality rule and has been assigned a severity level, such as warning, error, or fatal error. Data quality issues are stored in the check_results table and are used to measure the overall data quality by calculating data quality KPIs.

A data quality incident is a group of similar issues with the same properties. When a new issue that does not match any active incident is identified, a new data quality incident is created and associated with it. The incident is stored in the incidents table and assigned to the support and engineering teams for assessment and resolution.

Four statuses are used in the data quality incident workflow in DQOps, as shown in Figure 4.2:

- **Open** status is used for a new incident that was just detected because a new data quality issue (failed data quality check) was identified, and it did not match any other open, acknowledged, or muted incident. These issues need to be assessed and then managed by the 2nd-level support team or the data quality team.
- **Acknowledged** status is assigned when the data quality issue is confirmed and assigned to the 3rd-level support team for resolution.
- **Resolved** status is used when the 3rd-level support team solves the issue.
- **Muted** status is assigned to false-positive issues, low-impact issues, or issues that cannot be solved. DQOps will keep detecting data quality issues matching this incident. New incidents will be assigned to the muted incident for the next 60 days. The incident mute time window is configurable on a table level.

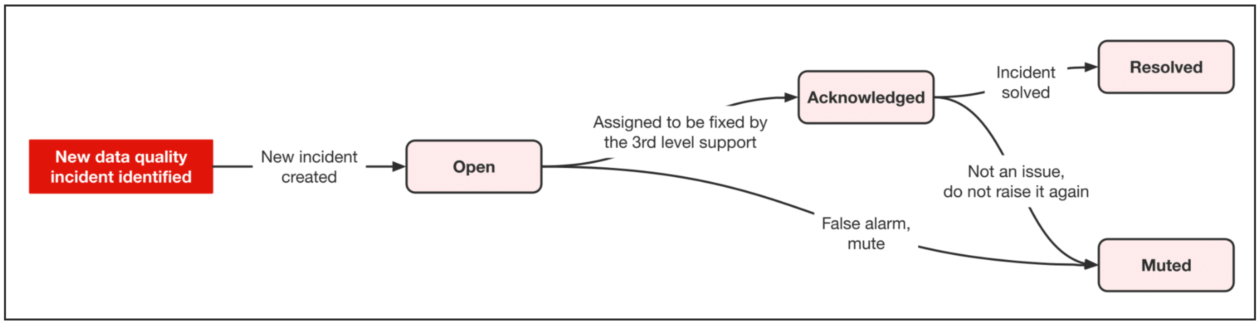

New data quality incident identified
New incident created
Open
Assigned to be fixed by the 3rd level support
Acknowledged
Incident solved
Resolved
Not an issue, do not raise it again
False alarm, mute
Muted

Figure 4.2. The diagram shows the management workflow of the data quality incident used in DQOps.

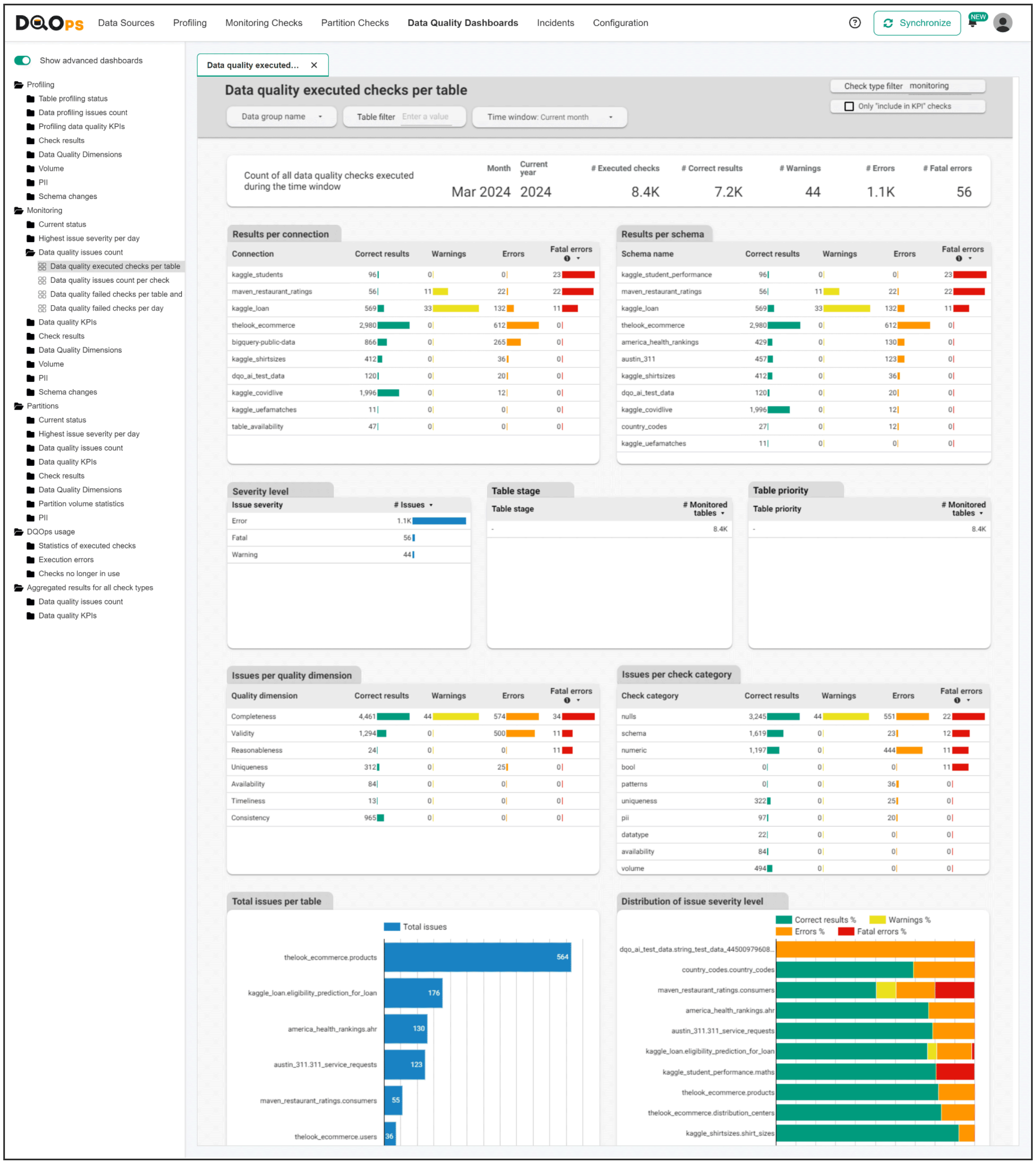
DQOps supports the following grouping levels:

- Table
- Table and data quality dimension
- **Table, data quality dimension, and check category**
- Table, data quality dimension, check category, and check type
- Table, data quality dimension, check category, and check name

The default configuration creates a data quality incident based on the grouping issues table where the issue was identified, the data quality dimension, and a data quality check category that mostly groups the check by the type of column or the way how the check is implemented.

In DQOps, It is also possible to raise data quality incidents only for error and fatal or only fatal severity issues. The default configuration assigned to each data source will create a data quality incident for all data quality issues, including warning severity issues. The warning severity issues also include many sensitive anomaly detection checks that may raise unexpected issues that will engage the support team. An alternative method of managing warning severity issues is to increase the minimum severity level for raising incidents to error, and use the current table status dashboards to review warnings.

The DQOps platform has several built-in dashboards that help identify tables with the highest number of data quality issues. Figure 4.3 shows an example of such a dashboard.



Data quality executed checks per table

Count of all data quality checks executed during the time window

Month	Current year	# Executed checks	# Correct results	# Warnings	# Errors	# Fatal errors
Mar 2024	2024	8.4K	7.2K	44	1.1K	56

Results per connection

Connection	Correct results	Warnings	Errors	Fatal errors
kaggle_students	96	0	0	23
maven_restaurant_ratings	56	11	22	22
kaggle_loan	569	33	132	11
thelook_ecommerce	2,980	0	612	0
bigquery-public-data	866	0	265	0
kaggle_shirtsizes	412	0	36	0
dqo_ai_test_data	120	0	20	0
kaggle_covidlive	1,996	0	12	0
kaggle_uefamatches	11	0	0	0
table_availability	47	0	0	0

Results per schema

Schema name	Correct results	Warnings	Errors	Fatal errors
kaggle_student_performance	96	0	0	23
maven_restaurant_ratings	56	11	22	22
kaggle_loan	569	33	132	11
thelook_ecommerce	2,980	0	612	0
america_health_rankings	429	0	130	0
austin_311	457	0	123	0
kaggle_shirtsizes	412	0	36	0
dqo_ai_test_data	120	0	20	0
kaggle_covidlive	1,996	0	12	0
country_codes	27	0	12	0
kaggle_uefamatches	11	0	0	0

Severity level

Issue severity	# Issues
Error	1.1K
Fatal	56
Warning	44

Table stage

Table stage	# Monitored tables
-	8.4K

Table priority

Table priority	# Monitored tables
-	8.4K

Issues per quality dimension

Quality dimension	Correct results	Warnings	Errors	Fatal errors
Completeness	4,461	44	574	34
Validity	1,294	0	500	11
Reasonableness	24	0	0	11
Uniqueness	312	0	25	0
Availability	84	0	0	0
Timeliness	13	0	0	0
Consistency	965	0	0	0

Issues per check category

Check category	Correct results	Warnings	Errors	Fatal errors
nulls	3,245	44	551	22
schema	1,619	0	23	12
numeric	1,197	0	444	11
bool	0	0	0	11
patterns	0	0	36	0
uniqueness	322	0	25	0
pii	97	0	20	0
datatype	22	0	0	0
availability	84	0	0	0
volume	494	0	0	0

Total issues per table

Total issues
thelook_ecommerce.products
564
kaggle_loan.eligibility_prediction_for_loan
176
america_health_rankings.ahr
130
austin_311.311_service_requests
123
maven_restaurant_ratings.consumers
55
thelook_ecommerce.users
36

Distribution of issue severity level

Correct results %
Warnings %
Errors %
Fatal errors %
dqo_ai_test_data.string_test_data_44500979608...
country_codes.country_codes
maven_restaurant_ratings.consumers
america_health_rankings.ahr
austin_311.311_service_requests
kaggle_loan.eligibility_prediction_for_loan
kaggle_student_performance.maths
thelook_ecommerce.products
thelook_ecommerce.distribution_centers
kaggle_shirtsizes.shirt_sizes

Figure 4.3. A sample screen of a detailed dashboard "Data quality executed checks per table," which is available on the DQOps Platform. This dashboard helps identify the tables with the highest number of data quality issues.

Note the following problems that may occur at this stage.

- Data quality alerting thresholds are too sensitive and unrealistic.
- Monitored tables are empty or outdated.
- Not all data quality checks were executed on time due to data platform performance issues.

Once the data quality issue investigation is completed, the Data Quality Team should be able to present the data quality issue to the Data Owner or Data Engineering Team. Data quality dashboards for the detailed dashboard should show enough information needed to identify the root cause of the problem.

DQ.9. Re-execute data quality checks

The data quality improvement process is a continuous loop of fixing data quality issues and reviewing data quality KPIs. The step in the middle is the data quality check re-execution, which is when the data quality checks are executed again to capture the most recent readouts.

When the data quality checks are re-executed after the Data Engineering Team has made major changes to the data model, many of the data quality checks already configured can become outdated. This happens often when monitored tables are deleted or replaced with new ones. In this case, the Data Quality Team should update the data quality check specifications, detach data quality checks from outdated tables, and reattach these data quality checks to the new tables.

Migration of data quality checks from outdated tables to new tables can affect the data quality KPIs because KPI is calculated for both old alerts from the outdated tables and new alerts on the new table. In some situations, the new table may be a copy of the old table with minor changes to the table schema (new columns, etc.). For time-partitioned data, when separate data quality scores are calculated for each partition, each data quality sensor readout and alert might get duplicated. The Data Quality Team should perform additional cleanup in the data quality database, removing outdated data quality results that do not reflect the state of existing tables or active data quality checks.

The stage of data quality checks re-execution can be divided into the following steps:

- **Identify outdated tables.** The Data Owner or Data Engineering Team may identify that some tables affected by data quality issues are outdated and should be excluded from the data quality monitoring process. On the other hand, serious data quality issues identified on key tables may require a major data remodeling process. The affected table can be replaced with a new table with a different schema and name. Data quality results for old tables may need to be removed from the data quality database.

- **Identify invalid data quality checks.** The Data Owner or the Data Engineering Teams may determine that some data quality checks are incorrect or that the alerting thresholds must be adjusted. These data quality checks must be re-executed with a new configuration.

- **Identify outdated data quality checks.** Alternatively, the Data Owner or the Data Engineering Team may decide that some data quality checks are not relevant and should be disabled or removed.

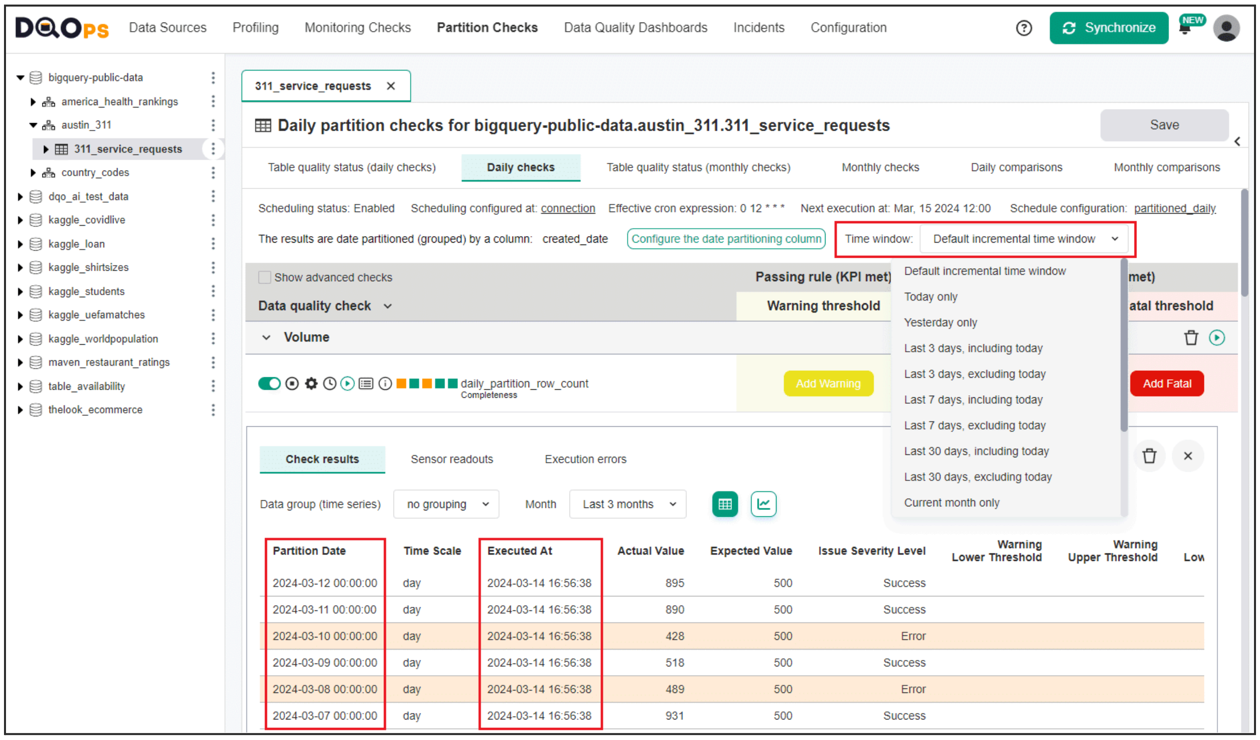
- **Identify the tables affected by the recent changes.** Identify any tables that have been fixed or the data quality checks that have been updated. These data quality checks will be re-executed on these tables.

- **Identify the range of updated partitions for date-partitioned data.** The Data Engineering Team should determine the range of affected daily partitions by partially or fully reloading large day-partition tables and re-executing the data quality check for the partitions affected by the changes.

- **Re-execute selected data quality checks.** Re-execution should be done either for the updated checks or all checks on modified tables. DQOps supports two types of time series that affect the re-execution process. The first is point-in-time results for the entire table, while the second is a data quality score for each data partition. Data quality checks that capture the data quality score calculated for the entire table generate only a single time-valid result for data quality check execution. The data quality database preserves previous data quality scores and alerts. This differs from how DQOps handles data quality results for date-partitioned data, where the data quality results (sensor readouts and alerts) for the past dates replace the existing data quality results in the data quality database. By knowing the earliest modified date, it is possible to limit the data quality check re-execution to only the time period (daily partitions) that was affected, avoiding a costly full scan of large tables. Figure 4.4 shows a screen from the DQOps user interface, which allows you to select the time window of the partition on which the check will be run.

- **Clean up outdated readouts and alerts.** The data quality database may contain outdated or duplicate data quality results for disabled or removed data quality checks or tables from the monitored database. The Data Quality Team should remove these results. If they are not removed, they will affect data quality KPIs. Some alerts may be calculated twice, making data quality KPI scores inaccurate.

- **Review data quality KPIs on dashboards.** After all the affected data quality checks have been re-executed and outdated data quality results have been removed, the Data Quality Team should review the data quality KPIs again. If new issues are identified or the KPIs still do not meet the expected values, the data improvement process may need to be repeated by going back to stage DQ.7. Check the KPIs on dashboards.



Daily partition checks for bigquery-public-data.austin_311.311_service_requests

Scheduling status: Enabled Scheduling configured at: connection Effective cron expression: 0 12 * * * Next execution at: Mar, 15 2024 12:00 Schedule configuration: partitioned_daily

The results are date partitioned (grouped) by a column: created_date

Time window: Default incremental time window

daily_partition_row_count

Data group (time series): no grouping Month: Last 3 months

Partition Date	Time Scale	Executed At	Actual Value	Expected Value	Issue Severity Level	Warning Lower Threshold	Warning Upper Threshold	Low
2024-03-12 00:00:00	day	2024-03-14 16:56:38	895	500	Success			
2024-03-11 00:00:00	day	2024-03-14 16:56:38	890	500	Success			
2024-03-10 00:00:00	day	2024-03-14 16:56:38	428	500	Error			
2024-03-09 00:00:00	day	2024-03-14 16:56:38	518	500	Success			
2024-03-08 00:00:00	day	2024-03-14 16:56:38	489	500	Error			
2024-03-07 00:00:00	day	2024-03-14 16:56:38	931	500	Success			

Figure 4.4. A sample screen from the DQOps data quality platform showing the results from running a daily_partition_row_count check using an error severity rule that raises a data quality issue when the number of rows in a daily partition is below 500 rows. DQOps allows you to select the time window of the partition on which the check will be run, allowing you to analyze only new data.

Note the problems that may arise at this stage.

- Data that was supposed to be fixed might still be corrupted or invalid.

- The data quality KPI scores on the dashboard did not change or fell below the threshold.
- The data quality check re-execution was time-consuming.
- Many different tables were updated, requiring a comprehensive data quality check review.
- Many data quality checks are outdated, making removing old data quality results from the data quality database time-consuming.

At the end of this stage, the Data Quality Team should decide whether the data quality KPIs meet acceptable levels. If the KPI scores are not acceptable and some additional data quality dimensions may be further improved, the data quality improvement process can be repeated. In this case, we repeat the steps described in stage DQ.7. Check the KPIs on dashboards. The Data Quality Team, together with the Data Owner and the Data Engineering Team, may also decide that the remaining data quality issues cannot be resolved at this moment. Data quality improvement for these issues can be undertaken at later stages of a continuous operational process.

DQ.10. Identify unresolved data quality issues

The data quality project usually has time and budget constraints. However, within these constraints, you can set up initial data quality metrics, optimize alerting thresholds, and fix data quality issues. Any data quality issues that remain unresolved at the end of the data quality project should be addressed at later stages of a continuous operational process. As a part of the project closing activities, all open issues must be documented in a report of unresolved data quality issues. Once the data quality project is closed, a data quality operations team may be formed from the Data Quality Team members to continue monitoring and fixing unresolved issues.

The unresolved data quality issues report should include the following elements:

- **Unmet data quality KPIs.** The report should include a list of high-level data quality KPIs, such as timeliness, along with a summary of actions taken during the data quality project to fix the problem.
- **Data areas with low data quality KPIs.** Collect a detailed list of unmet data quality KPIs at a data source, database, vendor, data supplier, or stage level.

- **Tables affected by open data quality issues.** Prepare a list of tables that frequently experience data quality issues or have open, unresolved issues. The data quality operations team that will be formed later will continue to improve the data quality of these tables.
- **Remaining data improvement tasks for Data Owners and Data Producers.** A list of open tickets assigned to Data Owners or Data Producers must be forwarded to the operations team. Some data quality initiatives are long-term, especially those that require data cleaning at the record level. These initiatives cannot be lost and become untracked during the transition to the operational process.
- **Remaining data pipeline improvement tasks for the Data Engineering Team.** List data pipeline or ETL tasks still under development by the Data Engineering Team. Once the data pipelines are fixed, the data quality operations team will re-execute affected data quality checks.

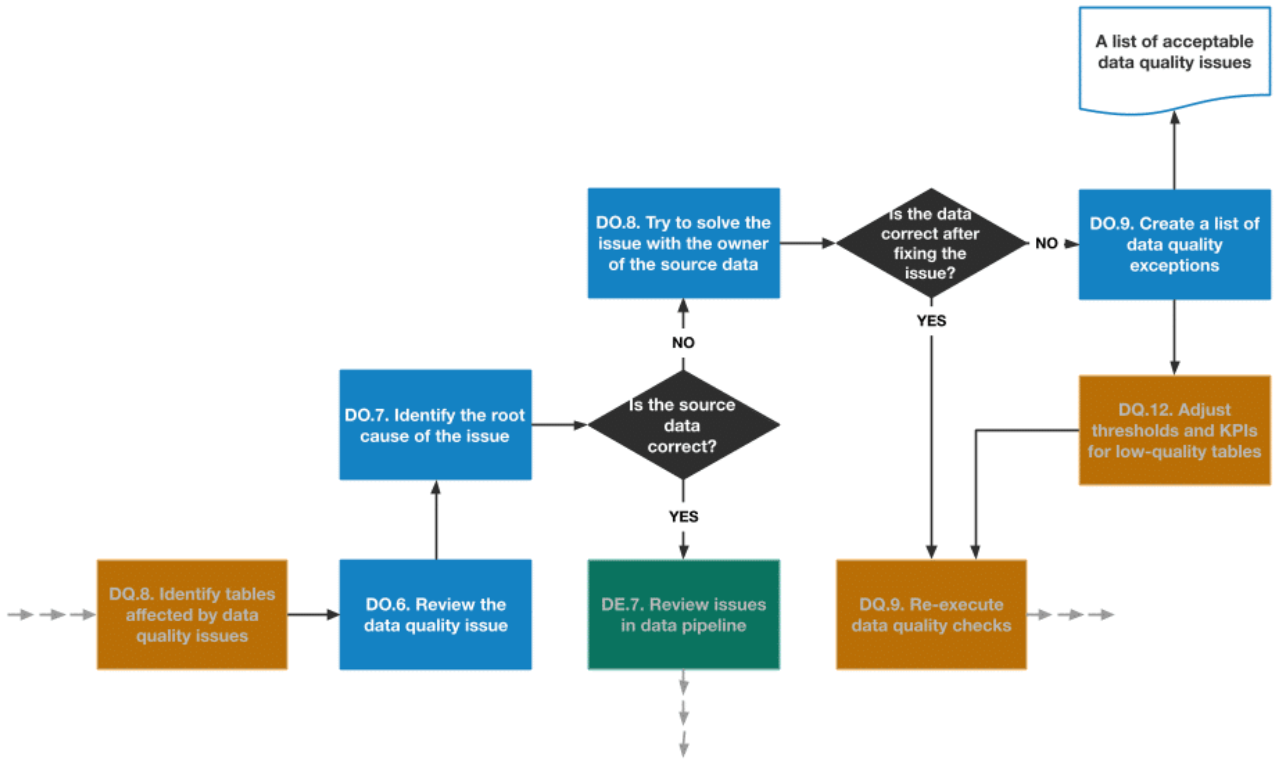
The data quality operations team's responsibilities will be similar to those described in this document but with yet another additional responsibility. The team should continuously review the data quality KPIs at the end of each reporting period and compare the weekly or monthly data quality KPIs with previous periods.

5. Fixing source data issues

Fixing source data issues requires close cooperation between the Data Quality Team, which identifies the issue, the Data Owner, who understands the data model, and possibly the Data Producers, who may correct the source data.

As a first step, the Data Owner needs to verify whether the problem is present in the source platform, such as an OLTP database, or only in the target platform (data warehouse or data lake). If the source data is correct, the Data Owner should contact the Data Engineering Team to review the data pipelines.

However, if there is a problem with the source data, the Data Owner should try to solve it with the Data Producer, even involving external data suppliers or business users. The Data Owner may create a list of acceptable data quality exceptions if the problem cannot be solved. The Data Quality Team will adjust the KPIs or alerting thresholds to meet the highest sensible data quality levels. For example, the percentage of accepted rows with null values may be increased to match a threshold that will be monitored from then on.


A list of acceptable data quality issues
DO.8. Try to solve the issue with the owner of the source data
Is the data correct after fixing the issue?
NO
DO.9. Create a list of data quality exceptions
YES
NO
DO.7. Identify the root cause of the issue
Is the source data correct?
DQ.12. Adjust thresholds and KPIs for low-quality tables
YES
DQ.8. Identify tables affected by data quality issues
DO.6. Review the data quality issue
DE.7. Review issues in data pipeline
DQ.9. Re-execute data quality checks

DO.6. Review the data quality issue

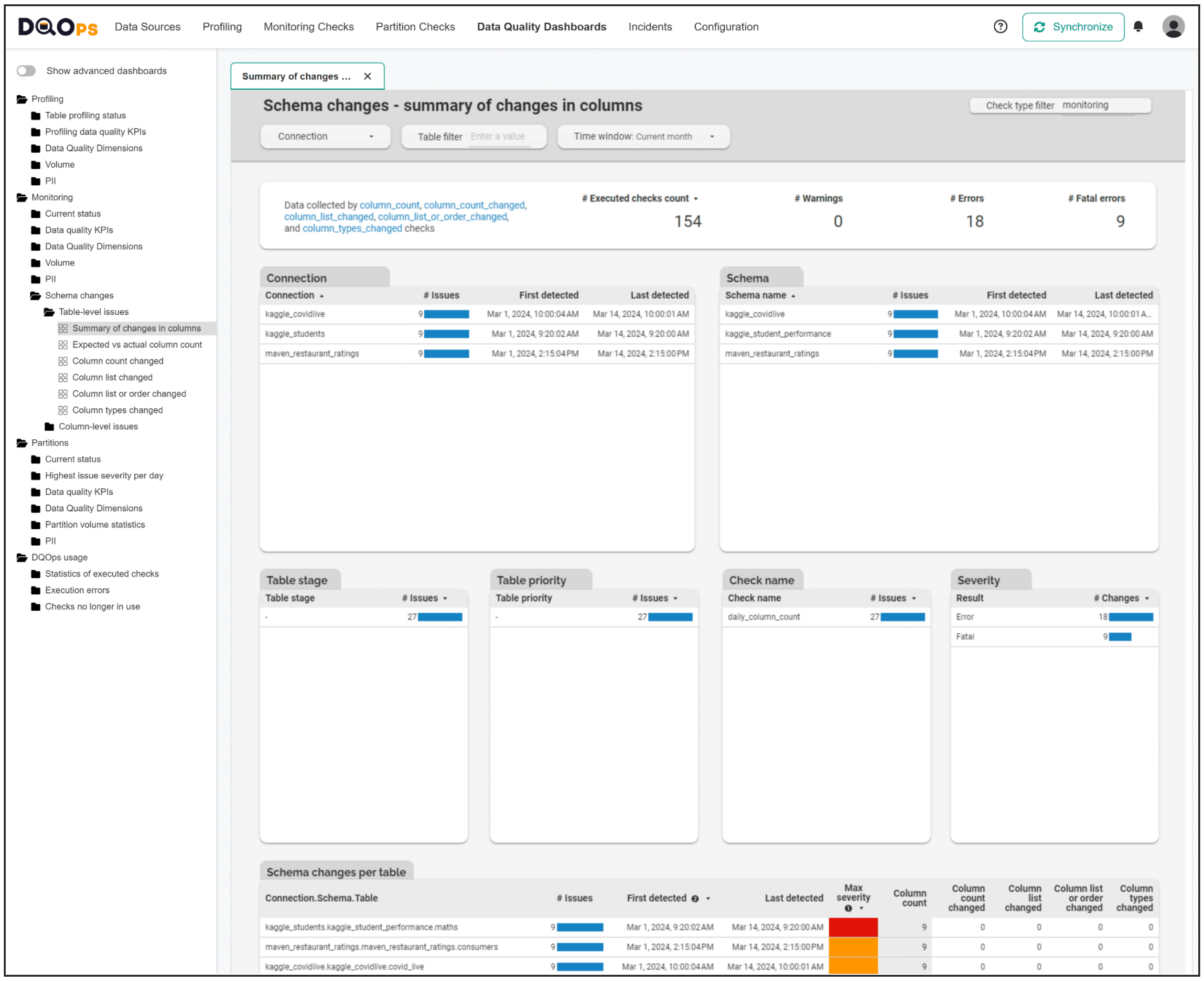
The Data Owner is a person best suited to understand the purpose of the data, the data model, and the business processes in their area of responsibility. The Data Owner should also know in-depth about all line-of-business applications synchronized with the monitored databases. The Data Owner is also the point of contact for external data suppliers, vendors, subsidiaries, or business divisions. For this reason, the Data Owner is entitled to request external parties to correct the quality of the data provided.

Communication between the Data Quality Team and the Data Owner is important at this stage. The Data Quality Team should share the result of the data quality investigation. The team should present the data quality KPIs that were not met, a list of affected tables, failed data quality checks, and historical data quality sensor readouts for reference. Based on the investigation, the Data Owner may identify an error in the data quality requirements, such as incorrect table names, so the problem can be fixed immediately without involving additional parties. The Data Owner should also be aware of the dynamics of the data, such as the data refresh calendar, which can affect the timeliness, completeness, or sometimes even the validity of the data.

The Data Owner and the Data Quality Team take the following steps at this stage.

- **Review the affected data quality KPIs.** The Data Owner should identify the drops in the affected data quality KPIs. This may be a decrease in the percentage of tables that are up-to-date when the problem is related to the timeliness dimension.
- **Review the affected tables.** The Data Quality Team and the Data Owner must understand the table schema and the purpose of the data stored in the tables. Both parties should review all relevant documentation to help them understand how these tables are created, populated, and used.
- **Review the alerting thresholds.** Data quality checks generate alerts when the alerting thresholds are exceeded. A threshold may be configured as an acceptable percentage of records with null values. Still, sometimes, due to the nature of the data or for historical reasons, a percentage or number of invalid records is acceptable. The Data Owner should confirm that the thresholds are still relevant.
- **Review the historical data quality readouts.** A data quality platform that monitors a data warehouse or data lake should store a history of data quality sensor readouts. For example, a data quality platform might have a dashboard that shows a graph of the data quality sensor readouts over a recent period of time, such as row counts for each day over the last three months.
- **Compare the issue with similar problems in the past.** The Data Owner who is an expert in their area of business may have extensive knowledge of similar problems in the past. The list of similar problems should be used in the next stage when a root cause analysis is performed.

DQOps platform offers a variety of built-in dashboards to aid in reviewing data quality issues. Figure 5.1 presents a sample screen of the dashboard named "Schema changes - summary of changes in columns," designed to summarize the results of checks that detect typical schema change issues. All built-in dashboards in the DQOps platform can be customized.



Schema changes - summary of changes in columns

Data collected by column_count, column_count_changed, column_list_changed, column_list_or_order_changed, and column_types_changed checks

# Executed checks count	# Warnings	# Errors	# Fatal errors
154	0	18	9

Connection

Connection	# Issues	First detected	Last detected
kaggle_covidlive	9	Mar 1, 2024, 10:00:04 AM	Mar 14, 2024, 10:00:01 AM
kaggle_students	9	Mar 1, 2024, 9:20:02 AM	Mar 14, 2024, 9:20:00 AM
maven_restaurant_ratings	9	Mar 1, 2024, 2:15:04 PM	Mar 14, 2024, 2:15:00 PM

Schema

Schema name	# Issues	First detected	Last detected
kaggle_covidlive	9	Mar 1, 2024, 10:00:04 AM	Mar 14, 2024, 10:00:01 A...
kaggle_student_performance	9	Mar 1, 2024, 9:20:02 AM	Mar 14, 2024, 9:20:00 AM
maven_restaurant_ratings	9	Mar 1, 2024, 2:15:04 PM	Mar 14, 2024, 2:15:00 PM

Table stage

Table stage	# Issues
-	27

Table priority

Table priority	# Issues
-	27

Check name

Check name	# Issues
daily_column_count	27

Severity

Result	# Changes
Error	18
Fatal	9

Schema changes per table

Connection.Schema.Table	# Issues	First detected	Last detected	Max severity	Column count	Column count changed	Column list changed	Column list or order changed	Column types changed
kaggle_students.kaggle_student_performance.maths	9	Mar 1, 2024, 9:20:02 AM	Mar 14, 2024, 9:20:00 AM	🟥	9	0	0	0	0
maven_restaurant_ratings.maven_restaurant_ratings.consumers	9	Mar 1, 2024, 2:15:04 PM	Mar 14, 2024, 2:15:00 PM	🟧	9	0	0	0	0
kaggle_covidlive.kaggle_covidlive.covid_live	9	Mar 1, 2024, 10:00:04 AM	Mar 14, 2024, 10:00:01 AM	🟧	9	0	0	0	0

Figure 5.1. A sample screen of a detailed dashboard "Schema changes - summary of changes in columns," which is available on the DQOps Platform. This dashboard summarizes results from checks detecting typical schema change issues.

Note the problems that may arise at this stage.

- The Data Owner is new to the organization and may not fully understand their domain of responsibility.
- The summary of the data quality issue presented by the Data Quality Team is incomplete and may require further investigation.
- The data quality issue presented to the Data Owner may result from incorrect data quality check configuration by the Data Quality Team. Therefore, the Data Quality Team

is responsible for fixing the issue during the next opportunity when the data quality checks are reconfigured and re-executed.

DO.7. Identify the root cause of the issue

Once all relevant information that describes the data quality issue has been gathered, the Data Owner can start an in-depth analysis that will require verifying the data at the record or business application levels.

In the event that the data has been prepared by an external vendor or a separate organization unit, the Data Owner should gather all relevant information about the issue that can be shared with the external party. Before escalating the issue further, the Data Owner should ensure that the problem was not caused internally during the data ingestion.

It is also possible that the problem with the data received from an external source was anticipated beforehand, in case the external vendor had clearly and in advance announced a major update in their release calendar. The announcement of a breaking change in the file format, data model, or API specification might have gone unnoticed ahead of time. Additionally, the code changes to the data pipelines were not completed on schedule before the upgrade of the source system.

The following steps can help you identify the root cause of an issue:

- **Look directly at the data in the monitored table.** The Data Owner should directly query the monitored table or review the content of source files for external tables. The problem may be obvious. For instance, the format of the date and time stored in a text column may have changed, the table is empty, or the column stores a different type of information than expected when preparing the data quality requirements.
- **Review the data lineage.** The Data Owner should check where the data came from. It might have been a different table at an earlier data processing stage, a flat file, or a table in another database, such as an OLTP database used by a line-of-business application.
- **Look directly at the data in the source table.** If the source table or source file can be identified, the Data Owner should look directly at the data in that table. If the source table contains valid data but the target (monitored) table does not, the issue must have appeared at the data transformation stage. The issue should be forwarded to the Data Engineering Team for fixing.

- **Verify the data in the line-of-business applications.** The data model of a line-of-business application can be complex. In that case, finding all the relevant information displayed on the application's user interface can be challenging when running queries on its OLTP database. The Data Owner should log into the business application, locate the relevant records, and check that all the information is present and consistent between the user interface and the database. Selected fields may not be stored in the main table but are conditionally stored in satellite tables. In this case, the data transformation process must be adapted to combine all relevant data.

- **Compare the issue with similar problems in the past.** Consult the log of previous issues in the ticketing system and knowledge base (if any is maintained). The first or second line of support may already maintain such a knowledge base somewhere.

- **Check for expected maintenance windows.** If the source system was down due to a scheduled maintenance operation, the data quality issue may have been foreseen ahead of time. In addition, a nightly job that performs a full data refresh may populate the source tables. If the data quality check is executed during a full refresh operation, it may analyze partially loaded data, raising unexpected data quality incidents.

- **Review the change log of the external platform.** The format of the data received from an external vendor or a SaaS application may have changed because the vendor announced a breaking change in the data model. The Data Owner should find any relevant announcements that may affect the quality of the data.

- **Check the error logs.** The problem may be caused by a bug in the processing logic. The length of the target column could have been too small to fit the incoming data, so the data loading job was canceled, or the ETL platform skipped some invalid rows. These types of issues will leave a trace in the log files or an external logging platform.

- **Ensure that an external party is responsible for the issue.** Before the Data Owner assigns responsibility for correcting a data quality issue to an external vendor, make sure the problem was not caused internally by the data recipient.

- **Assess the KPI levels.** The data quality KPI should be calculated as a percentage of passed data quality checks. Some data quality checks that compare the most recent data quality readout with historical readouts may identify anomalies that are nevertheless acceptable. Suppose the alerting thresholds for data quality checks that detect anomalies (such as the difference from the mean value) are too sensitive. In that case, the Data Owner must handle many false positive alerts. The Data Owner may decide to lower the KPI level (percentage of passed data quality checks), which a data quality investigation should follow.

- **Assess the alerting thresholds.** Alerting thresholds may be too sensitive and not correspond to reality. For example, the threshold for a row count increase by month may not consider the company's decreased activity during the holiday season. In that case, the threshold for the monthly row count data quality check for July should not be set to 90% of the previous month. The alerting threshold should allow a higher drop of a monthly row count increase to 80%.
- **Assess the level of relevance of the data.** The Data Owner may decide that data with lower data quality KPI scores is of low priority to the organization. Data quality improvements for the tables associated with these KPIs can be postponed or designated within the regular maintenance process in the future.

It is worth paying attention to the problems that may occur at this stage.

- The Data Owner does not have access to the monitored platform.
- The Data Owner does not have access to the source system.
- The data model of the analyzed platforms is complex and not well documented.
- The vendor does not provide detailed information about the planned changes ahead of time.
- The Data Owner cannot access line-of-business applications to verify issues at their source.
- The nature of the problem is complex, and additional parties must be involved in the investigation.

At the end of this stage, it is the responsibility of the Data Owner to determine whether the issue with data quality is present in the original source data or if it has occurred during the data transformation process. If the data quality issue is present in the data source, it should be escalated to the Data Producer. On the other hand, if the issue is internal, it can be resolved by either the Data Engineering Team (if the data was incorrectly transformed) or the Data Quality Team (if the data quality checks were misconfigured or configured with overly sensitive alerting thresholds).

DO.8. Review or fix the issue with the Data Producer

Data quality issues that arise from external vendors or line-of-business applications should be addressed with the relevant platform owner (the Data Producer). This could be an internal platform owner, such as a CRM owner, or a business process owner if the issues are caused by missing steps in the process or users not executing the process correctly, such as by entering incorrect data. If data quality issues are already present in datasets shared by an external vendor, their engineering team should be consulted and responsible for fixing them.

The following steps might help you solve the issue:

- **Collect all information about the issue.** The Data Owner must collect all the information about the incident identified by the Data Quality Team.
- **Prepare an information package.** The Data Owner should extract data samples from the data set to show exemplary invalid values. On top of that, all data quality results should be aggregated and prepared in a format that an external party can view without access to the data quality platform. This may be an Excel file that shows a list of outdated tables, the percentage of null column values in a data set, or other information that can clearly describe the data quality issue to an external party.
- **Prepare a list of suggestions.** You can prepare suggestions on how the problem can be solved, especially if the solution requires changing the database schema.
- **Decide on the business impact of the issue.** Assess the importance and the business impact of not fixing the data quality issue on the data consumer side. An external party, especially an external vendor, can calculate the cost of fixing the problem in the source platform. This must be compared with the loss generated by the data quality issue.
- **Propose a deadline for fixing the issue.** The external party must agree on the deadline and the milestones for the implementation steps.
- **Identify a point of contact on the Data Producer's side.** The Data Owner should know who to contact to discuss the data quality issue.
- **Contact an external party.** The Data Owner contacts the vendor to find the cause of the issue, correct the data, and/or understand why the data has changed.
- **Wait for a response.** Once the information package about the data quality issue has been submitted to the external party, the Data Owner can no longer resolve the issue. The Data Owner can only follow up with the external party when the relevant milestones are expected to be finalized.
- **Notify the data engineering and Data Quality Teams of schema changes.** An external solution may disrupt existing data pipelines and data quality checks. Changes to the data model must be applied to the schema of downstream tables. Schema

changes may also affect existing dashboards, ML models, data quality checks, or downstream data pipelines that export data to other data consumers.

- **Plan major changes.** The changes that affect multiple downstream teams must be planned with all respective parties.
- **Propose an alleviating solution for issues that cannot be fixed.** When a data quality issue cannot be solved, or the cost of fixing the problem does not justify the benefits, the Data Owner must decide on the next steps. Data pipelines or data models may need to be updated. Dashboards that are never expected to show valid information should also be removed.

Note the problems that may arise at this stage.

- Long response time from the vendor
- The recent change on the vendor side may require a significant modification of the business process.
- The estimated cost of fixing the problem exceeds budget constraints.
- The identified data quality issue was merely an anomaly detected by a machine learning algorithm that the vendor could not trace back.

If the Data Owner resolves the issue with the external party, the Data Quality Team is contacted. The Data Quality Team can re-execute data quality checks after the Data Engineering Team loads the corrected data. Issues that cannot be resolved must be listed as data quality exceptions.

DO.9. Create a list of data quality exceptions

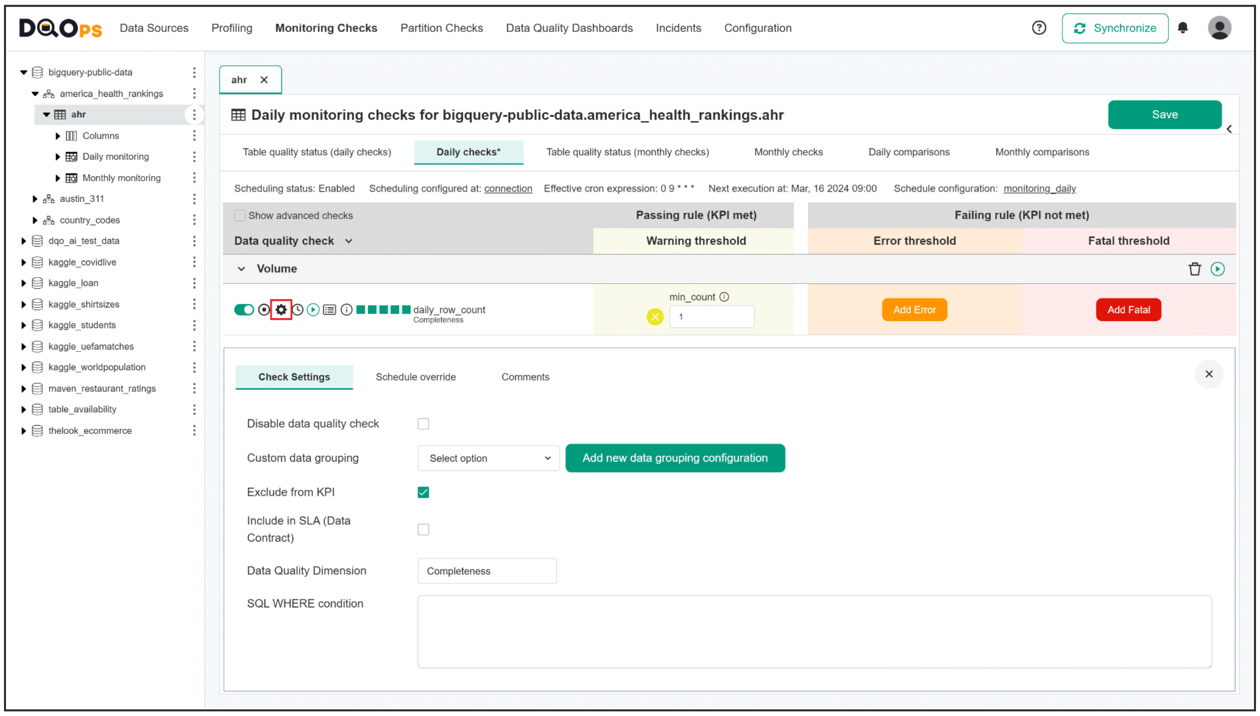
The remaining data quality issues that are tolerated or unavoidable should not affect the reported data quality KPI. They should be tracked but not precisely measured towards the total data quality KPI score due to their uncontrollable variability in the number of data quality alerts. Many of these data quality issues may be fixed in the future, but the exact time the issue is mitigated is not yet known.

The Data Owner should decide how to address the remaining data quality issues. Here are potential approaches:

- **Update the data model.** Changes to the data model may be necessary. This might involve adjusting column lengths or data types to accommodate previously rejected or truncated values. The Data Engineering Team should be involved in updating data pipelines and ETL processes accordingly.
- **Change the data processing logic.** For tables needing replacement with a different schema, decommissioning is a viable option. This might involve implementing additional pre-processing, joins, or post-processing steps.
- **Decommission tables.** Tables that must be replaced by another table with a different schema should be decommissioned. This requires informing all data consumers who use the old version of the table and carefully planning the transition process.
- **Apply temporary solutions.** If resolving an issue requires significant time, consider interim solutions. For example, business users may be advised to temporarily refrain from using affected dashboards until data reliability improves.
- **Disable data quality KPI monitoring.** If the KPI (percentage of data quality alerts) is high and changes too frequently, the monitoring can be turned off or postponed at the data quality KPI level.
- **Lower the data quality alerting thresholds.** Alerting thresholds for data quality checks can be adjusted to match the actual quality of data. For example, after discussing this with the platform owner, the Data Owner may decide that 20% of records may contain phone numbers that do not match an expected format. From this point, the Data Quality Team should simply measure whether the percentage of these invalid records is increasing, and only when this happens should they raise data quality alerts.
- **Reduce the severity of data quality alerts.** False positive alerts with no business impact should be monitored but not counted as failed data quality checks, lowering the overall data quality KPI score. DQOps supports the configuration of alerting thresholds at three severity levels: fatal errors for issues that should result in stopping data pipelines, alerts for regular data quality issues that should be fixed, and warnings for data quality issues that should only be observed. The severity level for non-critical alerts can be changed to a warning severity level and tracked from that point. These alerts will be treated as passed data quality checks and will not lower the data quality KPI.
- **Customize data quality sensor definitions.** If the current data quality checks cannot correctly detect data quality issues, it should also be possible to change the implementation of these checks. Since DQOps is an extensible data quality platform that uses the Jinja2 templating engine to define data quality sensors, it is possible to change the implementation of selected data quality checks. Another reason for

changing built-in data quality checks is related to performance issues when querying large tables. Custom data quality checks can be modified to consider the partitioning of large tables. This would enable efficient partition elimination conditions in the data quality sensor query.

- **Customize data quality alerting rules definitions.** DQOps executes Python scripts to assess the severity of data quality checks. Default rules can be customized to support higher variability in the data quality or to apply additional machine learning libraries to exclude some false alerts.
- **Plan a long-term data quality improvement project.** Consider initiating a dedicated project for critical data quality issues requiring significant resources to fix. In that case, you can temporarily disable specific data quality checks until the problem is resolved or exclude them from the overall data quality KPI calculation. The DQOps platform allows you to disable data quality checks and exclude them from calculating the data quality KPI score using the user interface, as shown in Figure 5.2.



Daily monitoring checks for bigquery-public-data.america_health_rankings.ahr

Scheduling status: Enabled Scheduling configured at: connection Effective cron expression: 0 9 * * * Next execution at: Mar, 16 2024 09:00 Schedule configuration: monitoring_daily

min_count: 1

Disable data quality check	☐
Custom data grouping	Select option
Exclude from KPI	✓
Include in SLA (Data Contract)	☐
Data Quality Dimension	Completeness
SQL WHERE condition	

Figure 5.2. The DQOps data quality platform allows additional configurations to be set at the data quality checks level. For example, a check can be excluded from calculating the data quality KPI score.

The data quality issues that cannot be solved shortly will affect downstream data consumers. The affected parties should be informed about the data quality issues, the timeline for solving the problem, and the temporary solution that can help mitigate the problem until it is fixed.

The Data Owner should contact the following parties:

- **Business users.** Business users who depend on data quality for decision-making should know how to adjust their business processes so they are not affected by invalid data.
- **Business intelligence teams.** Dashboards and reports that use incorrect data should be adjusted or even temporarily decommissioned if the numbers presented on them might lead to wrong decisions.
- **Data science teams.** Some machine learning models that depend on the data must be retrained, or an extra step of data cleaning must be applied to the training data sets.
- **Downstream data consumers.** Low-quality data sets shared with other teams will reduce trust between teams. Data consumers should be informed about the implications of data quality.
- **External parties.** The data consumer who receives low-quality data must be informed. Sharing low-quality data may even result in the termination of the data-sharing agreement between the parties.

As the data quality platform continues monitoring invalid data, the data quality issues under review by the Data Owner and the Data Engineering Teams are still at the top of the list on data quality dashboards. The Data Quality Team should receive a list of data quality exceptions to be applied to the configuration of the data quality checks. The issues will disappear from the list as soon as the Data Quality team applies all requested configuration changes, disabling false checks or adjusting the alerting thresholds.

The list of data quality exceptions should contain the following information:

- **Tables to be excluded from data quality monitoring.** The Data Quality Team will disable periodic data quality checks on these tables or unscheduled data quality check execution until the Data Producer improves data quality.
- **Data quality KPIs to be changed.** The data quality KPIs should be measured differently or even removed from the governance dashboards (top-level data quality KPI dashboards). The simplest solution is to lower the accepted KPI level for the linked data source.

- **New thresholds for data quality alerts.** The Data Owner may ask to decrease the alerting thresholds. For example, the accepted delay in receiving new data in timeliness data quality checks may be extended to account for additional delay.
- **Data quality alerts to be removed.** Some data quality checks may be removed. For example, validity checks that verify the data format of string columns may be disabled or replaced with a less sensitive data quality check that only measures the percentage of invalid records rather than detecting invalid values.
- **Data quality sensors to be customized.** Prepare a list of data quality sensors that need to be customized or custom data quality sensors that need to be modified. A custom data quality sensor should be able to detect custom data formats.
- **Data quality alerting rules to be customized.** Some alerting rules may be changed so false positive alerts are no longer raised.

DQ.12. Adjust thresholds and KPIs for low-quality tables

The Data Quality Team needs to gather all the necessary information from consultations with the Data Owner before applying any changes to the data quality platform. The list of data quality exceptions identified at the previous stage will be converted into configuration tasks for the Data Quality Team. The team will then disable any outdated data quality checks. It is important to note that data quality checks should be removed when a table is exempt from data quality monitoring.

Deactivation of data quality checks means that they will not be executed in the future, and as a result, data quality alerts will no longer be raised. However, alerts that have already been raised will still be stored in the data quality database. This may reduce the data quality KPI score calculated as a percentage of passed data quality checks. If alerts previously raised by these data quality checks are considered false positives, the Data Owner may remove them from the data quality database. In this case, the Data Quality Team should remove alerts and irrelevant data quality sensor readouts from the database.

Decommissioning entire tables from data quality monitoring is a common task. Therefore, the DQOps data model is designed to simplify this. Data quality results are stored in Parquet files, organized by connection name, table name, and month. This structure simplifies data cleaning – removing the relevant Parquet file effectively removes all associated data. For example, to remove September 2022 alerts for a specific table, you would simply delete the corresponding file:

`.data/check_results/c=conn_bq_17/t=austin_311.311_service_requests/m=2022-09-01/check_results.0.parquet`.

Removing data quality results is not just a matter of decommissioning data quality checks. Significant changes to data quality sensor configurations, alerting thresholds, or time dimensions might necessitate recalculating data quality checks for all affected daily and monthly partitions. Outdated data quality results should be removed when changes have also been applied to the configuration of the data grouping hierarchy. DQOps performs deduplication of data quality results (sensor readouts and alerts), but changes to the data dimensions (dynamic addition of GROUP BY clause) would preserve old results, which must also be removed.

Major changes to data quality KPI calculations or aggregation methods might need adjustments to data quality dashboards. The Data Quality Team must review and plan any necessary modifications to ensure the dashboards continue to provide accurate and relevant information.

The data quality checks decommissioning process has the following steps.

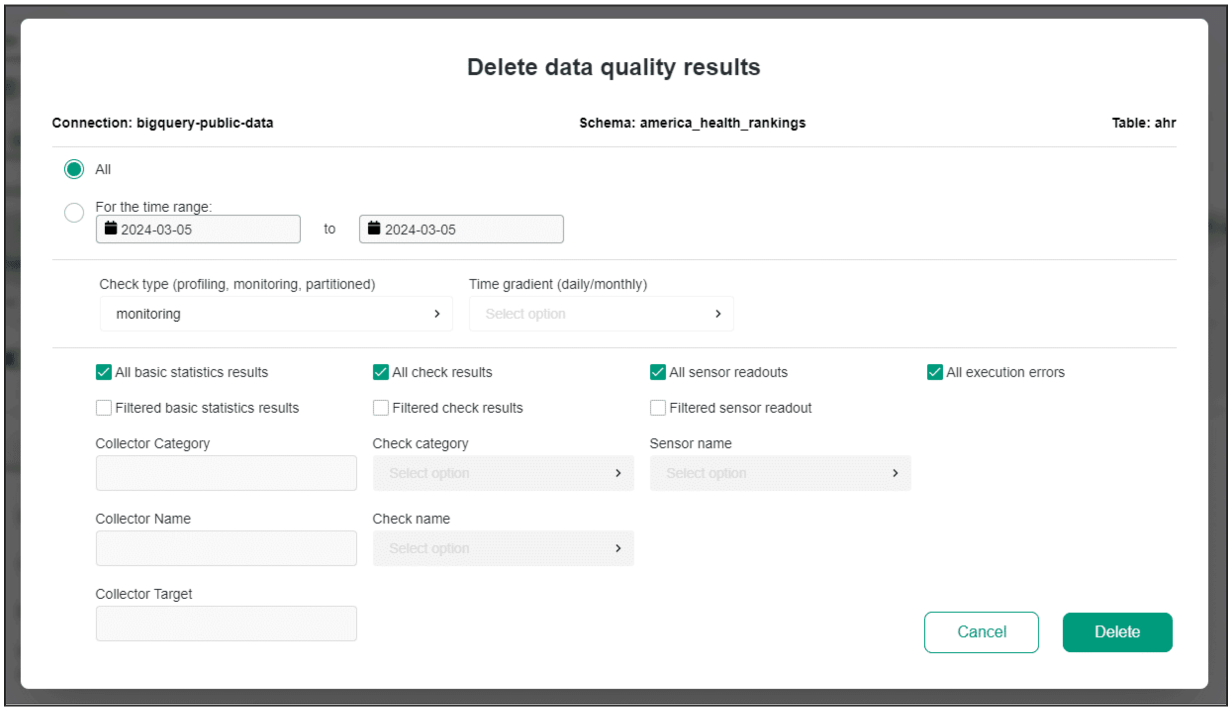
- **Deactivate data quality checks.** First, deactivate data quality checks. DQOps supports two ways of deactivating data quality checks. The checks can be removed from the data quality specification YAML files or deactivated by setting a "disable" flag at the data quality check level.

- **Exclude tables from data quality monitoring.** Tables that should no longer be monitored for data quality issues can be deactivated in two ways. The entire table-level data quality specification file can be deleted from DQOps. These files can be identified on the platform by the file extension *<schema_name>.<table_name>.dqotable.yaml*. The second option is to set a “disable” flag on the table, preserving the current configuration of data quality checks until the table is added back to data quality monitoring.

- **Remove outdated data quality results.** Outdated data quality results, including readouts and alerts, should be removed; otherwise, they will affect data quality KPI calculation. The DQOps platform offers a variety of filtering options (as illustrated in Figure 5.3) to help you remove outdated results.

- **Change the implementation of data quality checks.** Some changes may require changes to the implementation of data quality sensors or data quality alerting rules. DQOps supports the customization of built-in data quality checks by copying the implementation of these sensors and rules from the DQOps distribution into the user's home folder. The data quality sensors are stored in the "sensors" folder, and the data quality alerting rules are stored in the "rules" folder.

- **Reconfigure data quality checks and alerting thresholds.** Alerting thresholds or data quality sensor parameters should be updated in the data quality specification files.
- **Recalculate modified data quality checks.** Data quality checks that have been updated or reconfigured must be recalculated.
- **Update data quality KPI dashboards.** Optionally, modify the data quality KPI dashboards. Modification is especially advised if changes to data dimensions have been applied, resulting in new aggregations of data quality KPIs.
- **Review changes to data quality KPIs.** After any changes are made, the Data Quality Team must review the data quality KPIs. After that, they can continue to monitor data quality issues. A significant change in the level of the KPI or a significant increase in the number of data quality alerts may be due to a human error made while implementing the changes.


Delete data quality results

Connection: bigquery-public-data | Schema: america_health_rankings | Table: ahr

[✓ All]

For the time range: 2024-03-05 to 2024-03-05

Check type (profiling, monitoring, partitioned): monitoring

Time gradient (daily/monthly): Select option

[✓ All basic statistics results] [✓ All check results] [✓ All sensor readouts] [✓ All execution errors]

Filtered basic statistics results, Filtered check results, Filtered sensor readout

Collector Category:

Check category: Select option

Sensor name: Select option

Collector Name:

Check name: Select option

Collector Target:

Figure 5.3. The DQOps user interface allows you to delete data quality results at the connection, table, or column level for a specific time range, check type, or time gradient.

Note the problems that may arise at this stage.

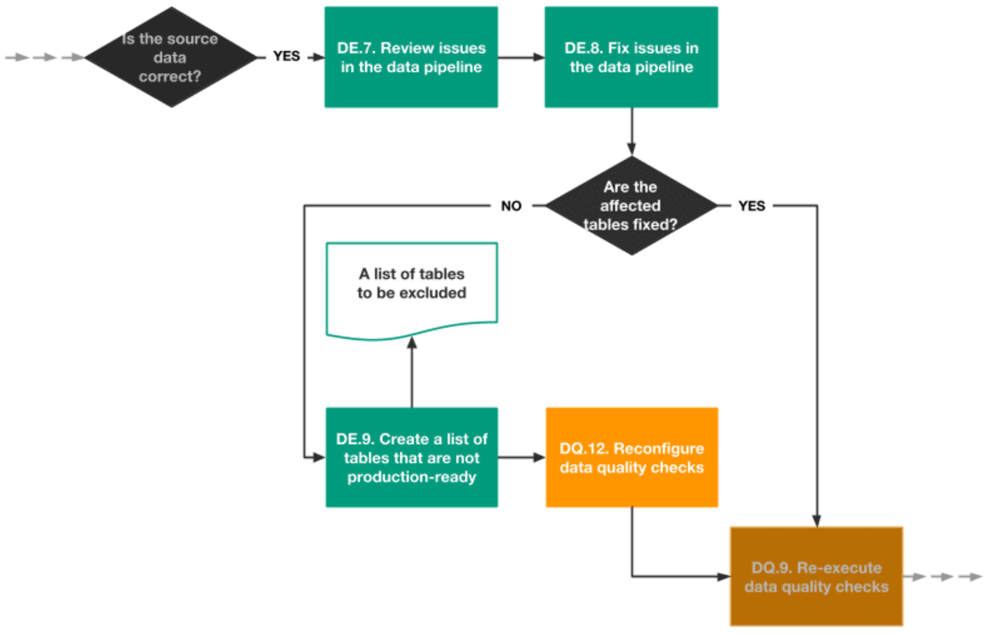
- Implemented changes do not result in improving the KPI scores.
- Removal of outdated alerts may reveal other data quality issues that went unnoticed up to this time.
- Implemented changes to KPI thresholds are too substantial, making the results unrepresentative.

6. Fixing data pipeline issues

The data quality issues that the Data Owner reviewed may have been present in the source data or introduced in the data pipeline during the data loading process. Moreover, some data quality issues present in the data source may have been fixed by the Data Producer, sometimes resulting in changes to the data model.

For all these data quality issues, the Data Engineering Team must update the data pipelines or at least perform a complete reload of updated source data. The required changes to the data pipelines may also be significant. Target tables should be categorized as not yet production-ready and excluded from the data quality KPI calculation.

Once the data pipelines have been updated and the target tables reloaded, the Data Engineering Team should re-execute data quality checks for the affected tables and review changes to the data quality KPIs.


Is the source data correct?
YES
DE.7. Review issues in the data pipeline
DE.8. Fix issues in the data pipeline
Are the affected tables fixed?
NO
YES
A list of tables to be excluded
DE.9. Create a list of tables that are not production-ready
DQ.12. Reconfigure data quality checks
DQ.9. Re-execute data quality checks

DE.7. Review issues in the data pipelines

The Data Engineering Team should review the data quality issue and determine why a data pipeline introduced the problem. Changes to the data pipeline or the data model of the source data applied by the Data Producer may also require changing the schema of target tables.

The Data Engineering Team should identify all affected data consumers along the data lineage who need to be notified. Changes to data pipelines and data models may require collaborative planning with downstream consumers or even external vendors who receive shared data.

Possible issues with data pipelines depend on the data quality dimension:

- **Timeliness**
    - The data pipeline did not start on time.
    - Too many data transformation tasks are executed, delaying the loading of the affected table.
    - The data pipeline sometimes does not finish, and the missing data is loaded during the next data pipeline execution.
- **Completeness**
    - The pipeline was not executed for particular days.
    - Several files were corrupted, or the data pipeline could not load them.
    - The data pipeline randomly fails during processing, skipping some files or loading target tables partially until it is stopped.
    - Unplanned maintenance tasks were carried out on the data transformation platform, which caused ongoing data pipeline tasks to stop.
    - The data pipeline jobs were manually stopped and never resumed.
    - Column values were truncated or converted to null values because they did not match the target column data type or were out of range for the target column's length or precision.
- **Validity**
    - Columns were truncated to fit into the target column data types.
    - Invalid values were not excluded during the data transformation phase.

- **Consistency**
  - The order of the columns has been changed, and source data has been loaded into the wrong target columns. These issues can be easily detected by monitoring the number of unique values in a column (cardinality).
  - The target table was loaded twice without truncating or partially loaded, so the number of rows has changed significantly since the last data quality check evaluation.
- **Uniqueness**
  - The table was loaded twice without truncating the target table first.
  - Some files were loaded multiple times.

Below are additional problems in data pipelines that will affect data quality checks independent of the data quality dimensions.

- Expired credentials.
- Access denials caused by lost access rights.
- New features introduced to the data platforms.
- Updates applied to the data pipeline tools.
- Changes in referenced libraries.
- Changes to the shared code base reused by multiple data pipelines (internal frameworks and libraries).
- Disk space issues.
- Network interruptions.
- Server failures or overloads.
- Unexpected system shutdown.
- Out-of-memory issues.
- Noisy neighbor problems caused by other processing jobs.
- Lack of communication between upstream and downstream teams.
- Subsequent tasks in a pipeline get stopped due to a failure of a preceding process.

The following points might help you quickly identify the issue:

- **Verify known issues.** The Data Engineering Team should check the problems that may have been identified earlier but were never fixed or might have happened in the past and were resolved for the time being. This step might help reduce the time needed to find the cause of the issue.
- **Review all logs and task boards.** The process should consist of the following steps:
    - Check the logs in tools for errors.
    - Find task boards for previous incidents and bugs that are queued for resolution.
    - Check if the issue is related to previous issues.
    - Verify if the issue is related to data pipelines that are still in development.
    - Verify if the issue is related to the latest committed improvements and changes.
    - Review maintenance notifications mentioning possible downtime of platforms used in the data pipeline.
- **Review the changes applied by the Data Producer.** The data quality issues fixed by the Data Producer may also require changing the data model. The Data Engineering Team should review all affected data pipelines and plan the required changes to the target tables and the data pipelines.
- **Examine the data.** The Data Engineering Team should make sure that they understand the data structure. The problem may have been introduced into the data pipeline because of inconsistencies between the dataset's documentation and the actual data structure.
- **Plan changes to the target table schema.** Issues that can only be fixed by updating the data model must be planned carefully. After all, target tables are used by downstream data pipelines or by data consumers such as data science or business intelligence teams.
- **Plan changes to the data pipelines.** The changes should be planned if the issue can be fixed entirely by updating the data pipeline. However, updating the data pipeline is not the only task. The target tables have to be reloaded, and downstream tables have to be updated or reloaded, resulting in further actions.
- **Identify all affected downstream data consumers.** Changes that affect other tables on the data lineage that are maintained by separate Data Engineering Teams must be

communicated to those teams. The change must be coordinated with all affected parties if it affects critical business processes.

- **Identify issues that are impossible to fix.** Finally, some data quality issues are not worth fixing, or no resources are available to make changes to the data pipelines. The Data Engineering Team may suggest retiring or temporarily disabling some pipelines or target tables.

It is worth noting the problems that may occur at this stage.

- The scope of the required changes is underestimated.
- The tools are outdated or prone to bugs.
- Many downstream data consumers will be affected by the changes.
- Not all affected parties have been identified or informed in advance.

DE.8. Fix issues in the data pipelines

Once the Data Engineering Team has identified the cause of the problem, the next stage is to fix it and notify all affected parties. The steps taken to resolve the issue should be documented and communicated to affected parties to help fix the problem faster or prevent similar problems in the future. The implementation steps required to fix an issue in the data pipeline are technology-specific and beyond the scope of this guide.

The actions described below will help you fix the issue.

- **Follow best practices and conventions.** The Data Engineering Team should use the data platform documentation and follow all internal best practices and conventions. Data pipelines implemented consistently are easier to understand and maintain in the future.

- **Reload the data.** Significant changes to the data pipeline may require a full reload of the target table. Reloading large data sets should be planned in advance.

- **Identify potential additional overlooked downstream systems.** Look for other downstream systems that may depend on the reloaded data. These systems may be overlooked during the research phase but will be susceptible to table removal or table model changes. For example, an external system may query the affected table through a database view. In case of slight changes in the table's schema, the view must be deleted and recreated.

- **Refresh data along the data lineage.** If the data has changed, tables and data marts downstream of the data lineage must also be refreshed.
- **Prevent the spread of corrupted data.** Changes should be communicated to other stakeholders who might still be using corrupted data. They should be notified of data refreshes or schema changes as soon as they occur.

It is worth paying attention to the problems that may arise at this stage.

- Loading new data does not solve the problem of corrupted data that has already been propagated.
- Lack of communication between the engineering teams.
- Fixing one problem can reveal another.
- The improvements are time-consuming.
- The Data Engineering Team assigns the lowest priority to the issue.
- Reloading data is difficult or impossible.

If the issue with the data pipeline is fixed, the Data Engineering Team informs the Data Quality Team, and the related quality checks can be re-executed. The remaining data quality issues identified as false positives will be excluded from the data quality monitoring process.

DE.9. Create a list of tables that are not production-ready

If the data pipeline issue has not been resolved, the Data Engineering Team has decided that some tables are not yet production-ready and should be excluded from the data quality KPIs measurement. Tables that fall under the following cases are not considered production-ready:

- **Under development.** Tables that are still being designed, prototyped, or frequently changed should be considered temporarily unstable. The Data Engineering Team should decide whether to monitor these tables with the data quality platform to provide live data quality findings for ongoing development. However, these tables can also be excluded from the data quality KPI calculation to avoid a negative impact on the data quality KPI reported from the governance dashboard, diminishing the measures that represent the operational state of stable parts of the system.

- **Obsolete.** Tables that have recently become obsolete and are no longer used in business operations should also be excluded from the data quality KPI calculation. These tables will negatively impact the data quality KPIs but will not be fixed.

The Data Engineering Team should prepare three lists of tables affected by recent data quality issues. These lists will guide the Data Quality Team in reconfiguring data quality checks.

- **Tables to be excluded from data quality monitoring.** The data quality platform should no longer monitor tables that are either obsolete or about to undergo significant redesigning. Any data quality issues reported on these tables are incidental and irrelevant to the business.
- **Tables to be paused from data quality monitoring.** Tables affected by ongoing changes to data pipelines can be paused until a new implementation of the data pipeline is completed. The current configuration of data quality checks on these tables may contain a lot of business knowledge, such as finely tweaked alerting thresholds for certain data quality checks. After the data is stable, the Data Quality Team can resume monitoring these tables.
- **Replaced tables.** When changes to affected tables require significant changes to the table schema, the Data Engineering Team may create a new version of the table with a different name. The Data Quality Team should switch all configured data quality checks from the old version to the new one.

Some problems may arise at this stage that are worth noting.

- The Data Engineering Team cannot estimate when the development will be completed.
- Additional parallel activities affect the availability of the Data Engineering Team to complete development.
- The Data Engineering Team cannot make an authoritative decision on whether some obsolete tables should be considered no longer necessary.

DQ.11. Reconfigure data quality checks

The Data Quality Team collects feedback from the Data Owner, the Data Producer, and the Data Engineering Team regarding necessary changes to data quality checks. Below are the most common changes that need to be implemented.

- **New data quality checks.** New data quality checks need to be activated on new tables, or additional data quality checks must be activated on already monitored tables.
- **Deleted data quality checks.** Data quality checks that generated false positive alerts need to be removed.
- **Changed alerting thresholds.** Alerting thresholds need to be adjusted according to the actual quality of data. For example, the Data Owner expected 0% of the rows with null values for the country column, but in reality, 5% of rows have no specified country. A safe threshold of up to 6% of the rows with no specified country should be configured to monitor if data quality decreases over time.
- **Renamed tables.** In case some tables have been renamed or new versions designed, the existing data quality configuration should be moved or copied to the new tables.

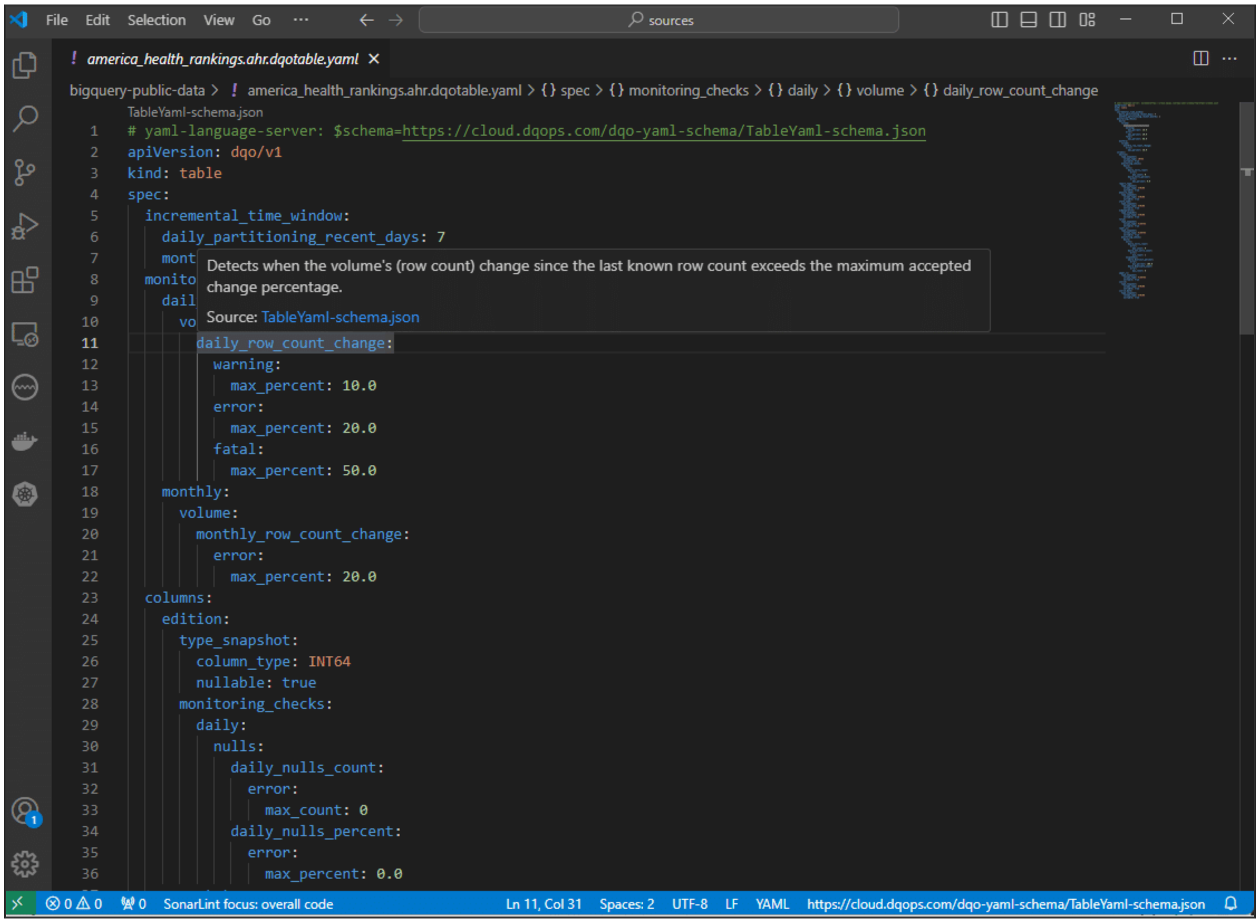
Once the changes have been applied to the data quality check configuration, the Data Quality Team should review the new data quality KPI scores. Outdated alerts related to deactivated data quality checks may affect the overall data quality score, especially if the alerts were raised by incorrectly configured data quality checks or the alerting thresholds were too sensitive.

To reconfigure data quality checks, the Data Quality Team should follow the steps given below:

- **Deactivate outdated data quality checks.** There are three ways to deactivate data quality checks supported by DQOps:
    - Deleting the configuration of data quality checks.
    - Disabling a configured data quality check that will retain the configuration but exclude the check from evaluation.
    - Removing the table metadata when a table is decommissioned.
- **Delete false positive alerts.** Alerts raised by misconfigured data quality checks affect the data quality KPI that is measured as a percentage of passed data quality checks. The Data Owner can decide to remove these alerts to track the data quality KPI for valid data quality checks only.
- **Connect new data quality checks.** Additional data quality checks can be configured when new data quality requirements are specified during a review of data quality issues with the Data Owner, Data Producer, or Data Engineering Team.
- **Re-evaluate the updated data quality checks.** All modified data quality checks should be re-executed. Updated or new data quality checks can generate new data quality alerts that affect the data quality KPI score.

- **Review data quality KPIs on data quality dashboards.** The Data Quality Team should check whether the data quality KPI has increased or decreased after applying the changes. Activating new data quality checks can have unexpected consequences. Relaxing alerting thresholds may also not increase the KPI score as expected.

- **Push a new data quality configuration to a source control repository.** DQOps stores the configuration of data quality checks as YAML files (as shown in Figure 6.1), making them easily manageable with version control systems like Git. In validated environments, all changes to the data quality configuration can be pushed to a dedicated branch and reviewed using a pull request before affecting the main branch. A complete code review of new data quality checks prevents issues with possible data leaks if the data quality check is configured to perform data lookups using a very selective filter condition, for example: WHERE SSN="SSN of a person of interest."


File Edit Selection View Go ···
sources
! america_health_rankings.ahr.dqotable.yaml ×
bigquery-public-data > ! america_health_rankings.ahr.dqotable.yaml > {} spec > {} monitoring_checks > {} daily > {} volume > {} daily_row_count_change
TableYaml-schema.json

```
# yaml-language-server: $schema=https://cloud.dqops.com/dqo-yaml-schema/TableYaml-schema.json
apiVersion: dqo/v1
kind: table
spec:
  incremental_time_window:
    daily_partitioning_recent_days: 7
    mont
  monito
    dail
      vo
        daily_row_count_change:
          warning:
            max_percent: 10.0
          error:
            max_percent: 20.0
          fatal:
            max_percent: 50.0
    monthly:
      volume:
        monthly_row_count_change:
          error:
            max_percent: 20.0
  columns:
    edition:
      type_snapshot:
        column_type: INT64
        nullable: true
      monitoring_checks:
        daily:
          nulls:
            daily_nulls_count:
              error:
                max_count: 0
            daily_nulls_percent:
              error:
                max_percent: 0.0
```

Detects when the volume's (row count) change since the last known row count exceeds the maximum accepted change percentage.

Source: TableYaml-schema.json

⊗ 0 ⚠ 0 0 SonarLint focus: overall code Ln 11, Col 31 Spaces: 2 UTF-8 LF YAML https://cloud.dqops.com/dqo-yaml-schema/TableYaml-schema.json

Figure 6.1. In the DQOps platform, data quality checks are configured on monitored tables and columns in the <schema_name>.<table_name>.dqotable.yaml YAML files. You can quickly migrate the configuration by simply changing the name of the YAML file.

It is necessary to pay attention to the following problems that may arise.

- False positive alerts raised by deactivated data quality checks lower the data quality KPI score, which may not be the desired outcome.
- Tickets associated with false positive alerts may still be active and must be rejected in the ticketing system.
- Re-evaluation of updated data quality checks may generate additional workload on the monitored database.

Summary

This eBook provides a framework for establishing and maintaining trustworthy data, allowing organizations to make confident, data-driven decisions that fuel success. It outlines a step-by-step approach that involves setting up data quality monitoring, analyzing key performance indicators (KPIs), and continuously improving data quality.

To simplify this process, the eBook introduces DQOps, an open-source data quality platform. DQOps goes beyond a typical data quality platform and empowers organizations to manage data quality throughout the entire data lifecycle. The process starts with profiling new data sources and ends with full automation of data quality monitoring in operations. This adaptability is crucial because the approach to data quality and the preferred interface for managing it change throughout this lifecycle.

Data quality is an ongoing journey, not a destination. By actively embracing continuous improvement and ensuring strong cross-team collaboration, organizations can stay ahead of the curve and maintain high-quality data.

This eBook describes the proven data monitoring process that will help you remove all data quality problems. It was created by the DQOps Team based on their experience in data cleansing and data quality monitoring.

Inside this eBook, you will find how to:

- Set data cleansing goals.
- Conduct an iterative data cleansing project.
- Measure data quality across multiple dimensions of data quality, such as accuracy, validity, completeness, consistency, currency, or timeliness.
- Detect and respond to data quality problems in the future.
- Detect problems in data pipelines.

Learn more about DQOps at www.dqops.com

DQOps