A step-by-step guide to improve data quality

How to reach a 100% data quality score

Table of Contents

Introduction

Data quality assurance is the process of ensuring that data meets the expectations of the people who will use the data. The data quality field has adopted the word “dimension” to identify these 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 means something different to different companies and the number and types of data quality measures can vary depending on how a company wants to use its data. The results of data quality measurements are defined as data quality key performance indicators (KPIs). We can divide the data quality KPIs into business-focused and data engineering-focused. The data quality KPIs for business help monitor data quality to create insights, optimize processes, and improve decision-making. The data quality KPIs for data engineering monitor the problems with the data pipelines, file skipping, pipeline failure, etc.

The data quality monitoring and improvement involve the following stakeholders: the Data Owner, Data Producer, Data Engineering Team and Data Quality Team.

The Data Owner understands the purpose of the data, the data model, and the business processes in their area of responsibility. This person should also have in-depth knowledge about all line-of-business applications synchronized into the monitored databases. The Data Owner analyzes and defines business needs, indicates which tables should be checked for data quality, and sets acceptable metrics and values for data quality checks. The Data Owner also identifies underlying causes of potential data quality issues, if necessary.

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 builds and maintains data pipelines and maintains databases. It is also responsible for designing, building, testing and maintaining data management systems.

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. The Data Quality Team imports metadata of monitored databases into the data quality platform, configures data quality checks, and monitors data quality issues. When data quality issues are detected, the Data Quality Team facilitates the data cleaning process, engaging all stakeholders.

This eBook describes how to effectively improve data quality by setting up a data quality monitoring process. The process is based on tracking data quality KPIs to prioritize and identify data quality issues that should be addressed with the highest priority. 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 is an iterative process and can be divided into the following steps: identification of issues by the Data Quality Team, contacting the Data Owner and/or Data Engineering Team to resolve the issues, re-executing data quality monitoring by the data quality team.

The data quality improvement process described in this eBook is explained as a flowchart that shows the sequence of activities to make the process work. Each step and decision in the process is indicated by a shape. The rectangles, representing actions, have unique abbreviations and numbers to help you find a detailed description in the eBook. Different 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 identification of areas for improvement, before quality issues affect the reliability of your analytical processes, we created the DQO data quality platform. Our platform integrates easily into DevOps environments and offers extensibility that allows you to define business-specific data quality checks. DQO helps you examine ETL processes, data warehouses, and databases and discover which data is incorrect by monitoring popular data platforms (e.g. GCP, Snowflake) and running predefined data quality checks.

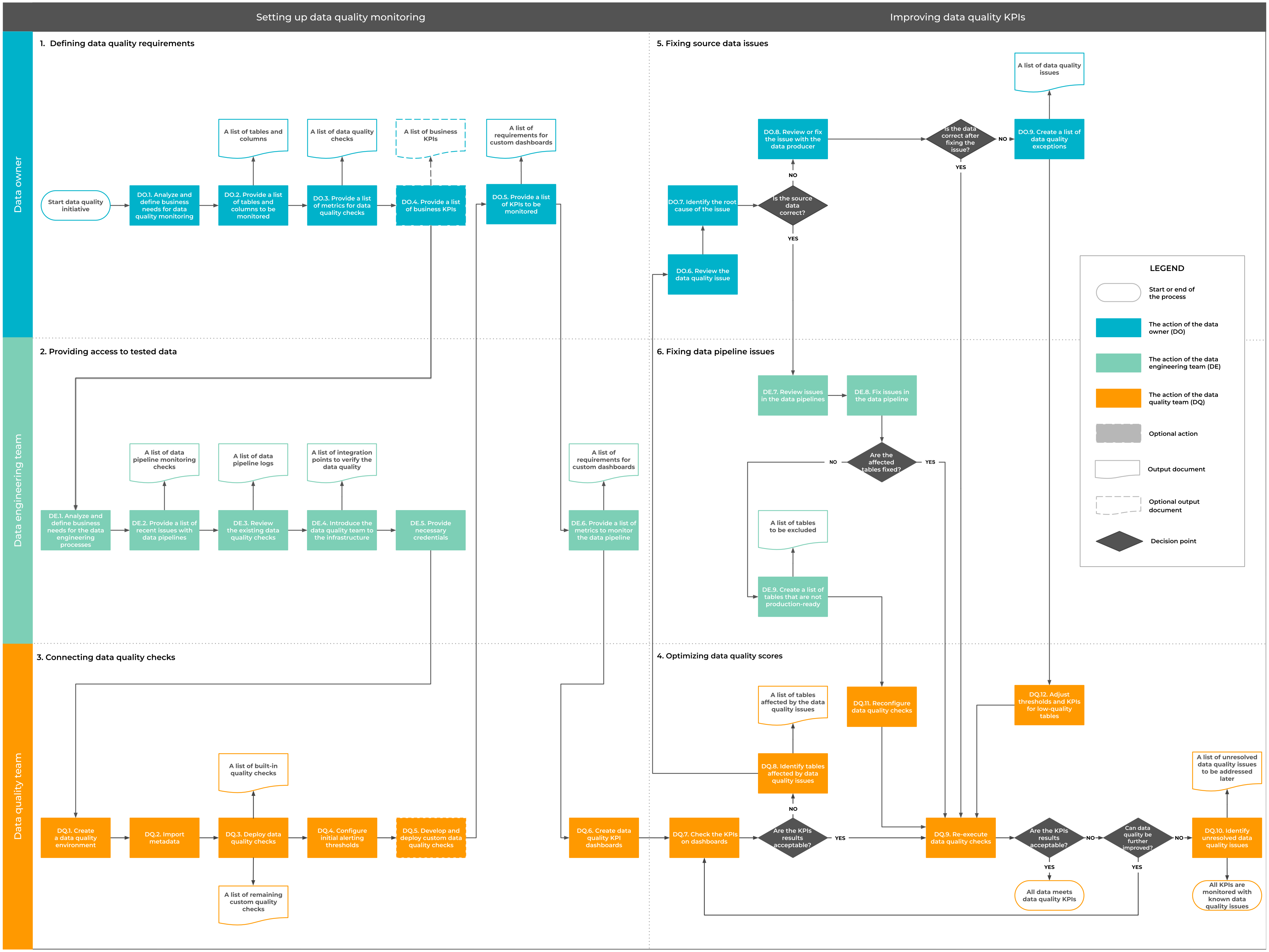
In DQO, the combination of the data quality sensor and data quality rule is called a data quality check. The data quality sensors capture metrics such as the number of rows, the percentage of null values in a column, or the current delay between the timestamp of the latest row and the current system time. The sensors can be implemented as templated SQL queries (DQO uses Jinja2 templating engine) or as a 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. DQO 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 that imports metadata of 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. DQO calculates data quality KPIs as a percentage of passed data quality checks for each table, database, or connection.
Data stream	A group of rows that were loaded from a single source into a table. A table might aggregate data coming from multiple data streams, generated by different data producers. Data streams are used to calculate separate data quality KPI scores for different groups of rows, such as an individual data quality KPI for each country.
Rule	Set of conditions against which sensor readouts are verified, described by a list of thresholds.
Check	A test for data quality which 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.

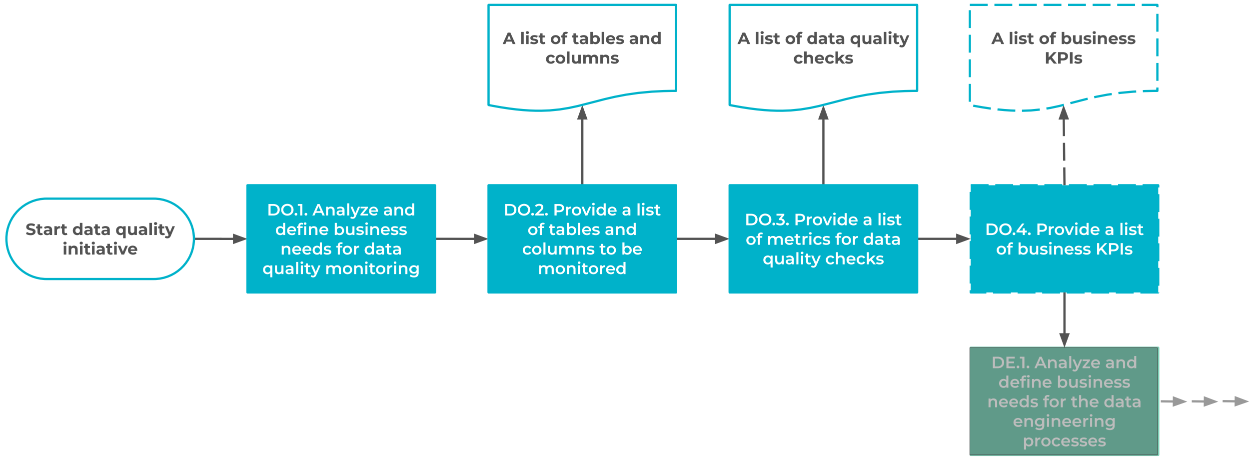
Data quality monitoring process



I. Setting up data quality monitoring

The process of setting up data quality monitoring can be divided into three parts. First, the Data Owner needs to define data quality requirements which are 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 in terms of 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



The Data Owner analyzes and defines business goals and needs for data quality monitoring. In the next steps, a list of tables and columns that the Data Owner would like to monitor is provided. 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 in terms of data quality monitoring. This can be a specific issue that the Data Owner is trying to fix or it can be 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, that 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 methods of monitoring 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 have a clear picture of business owner needs, it is important to discuss and set business goals for data quality. This will make it easier to come up with the 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 to qualitatively assess the scope and area of the problems which 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.** A different set of data quality categories is important for different companies. 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 certain 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	Data quality issue
Timeliness	The data is not up-to-date
Completeness	Missing values or rows
Validity	Invalid data format
Consistency	An inconsistent number of rows
Uniqueness	Lack of a unique identifier for the record or presence of duplicates
Reasonableness	The values in the database are not reasonable

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 data quality KPIs that would 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 the implementation of data quality monitoring using DQO, 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.

- **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.

- **Provide a list of tables to be monitored.** The Data Owner should gather a list of tables that are important and 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.

- **Identify large tables.** Very 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. DQO 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 DQO are not limited to tables physically stored using date partitioning. Tables that do not change very often can also be analyzed for data quality issues in a daily time gradient. DQO 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. The data quality platform can analyze if rows match the target type such as a number or valid date. The data quality platform can detect why some files or some rows were not loaded downstream.

- **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 a futile effort. A data quality platform can detect data quality issues on these tables as soon as they are redesigned. In particular, data quality checks for timeliness can detect that decommissioned tables are no longer fresh.

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 process of analysis 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. For each column selected for data quality monitoring, the Data Owner should define specific metrics. Each selected metric should relate to a dimension of data quality such as timeliness, completeness, validity, consistency, or uniqueness. This is an important step, where a precise definition of metrics reduces the time for implementation and subsequent tweaking of the data quality rules.

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

- **Specify data quality expectations.** From the assumptions about the expected quality of the data, the Data Owner should determine how to measure the degree to which 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. Many business-specific data quality checks can only be implemented in extensible data quality platforms that support custom data quality checks and rules such as DQO.

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

- Inapplicability of selected metrics on 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 driving the need. A good understanding of how the database structure fits the business process is crucial for understanding when the differences and errors in the data will not match reality. The Data Quality Team should review the proposed business-specific KPIs to assess the feasibility of monitoring these KPIs using a data quality tool. The data model used by the data quality tool for storing the data quality check results must support proper reporting of those KPIs measured from the business perspective. In addition, business-specific data quality dashboards should be designed, presenting KPIs separately for business areas, organizational units, geographical locations, suppliers, business partners, etc.

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 an important step in the process because 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.

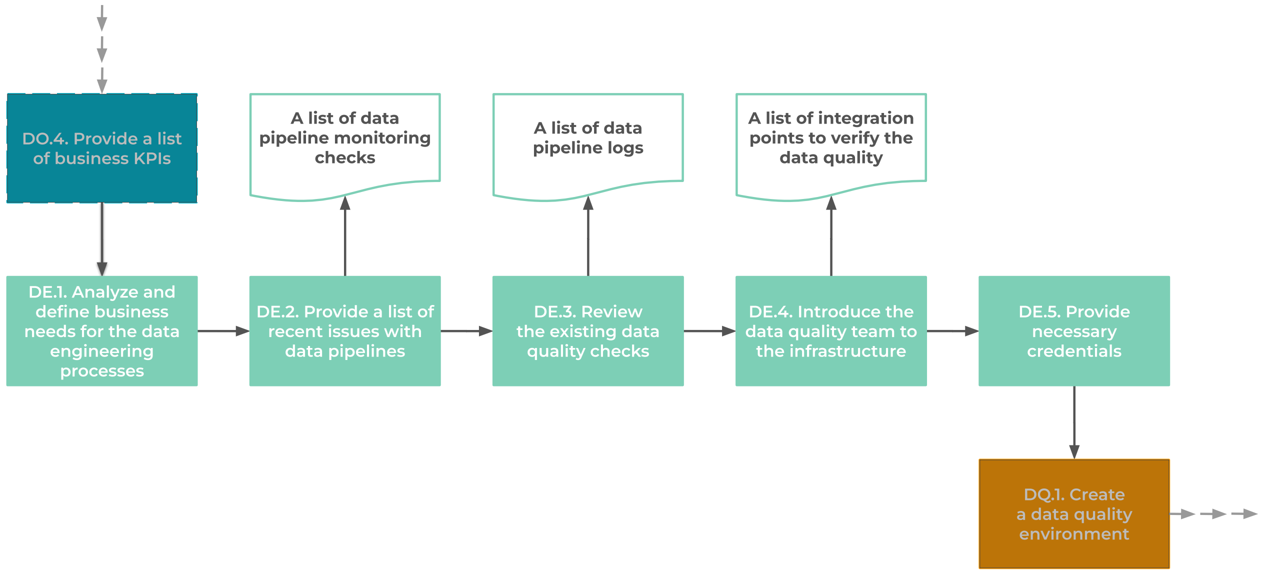
Problems that can appear at this stage:

- The database model is not documented properly.
- 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



The Data Engineering Team is responsible for managing the data warehouse or the data lake. The data quality issues of the incoming data may affect the stability of the data pipelines. The data quality issues may pile up, 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.

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 the data observability tool can provide such a view, combining business-level data quality KPIs (what the business needs) with the data processing KPIs (what errors or data quality issues affected the data processing).

This stage describes the key 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 important 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.

DE.1. Define requirements for the data engineering process

At this stage, 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 would like to monitor in terms of data quality.

Before proceeding 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 errors in the pipelines or low-quality data coming 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 data quality issue notification channels.** 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 for tracking and assigning work items. The data quality platform should publish notifications to these notification 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 can open issues directly in the ticketing system this can lead to a huge number of tickets. The Data Engineering Team should provide a list of KPIs for tracking the increase or decrease in the number of issues. These KPIs are tracked on the data quality dashboards for data engineering. A typical type of KPI is the number of data quality issues that were detected as an anomaly, using a time series anomaly detection algorithm.

- **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 such as 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.

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.
- Running data quality checks required by the Data Owner is not feasible on the existing physical infrastructure, due to the volume of data that would be queried every day. The data quality checks should be implemented as SQL queries, but the complexity of these queries may require heavy full-table scans or executing 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 mostly visible 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 take into account 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's expected to appear in the database.

- **Identify parallel data streams.** You may have tables that aggregate data from different data sources. These different data sources load data in separate data pipelines. Rows from different data streams can be identified by certain columns in the table, such as country, state, vendor, and department. Data quality checks can monitor the table for each data stream separately by simply running the "GROUP BY country" 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.

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 of 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 most cases, the logs can be queried like regular databases. A data quality tool that supports custom data quality check definitions (such as DQO) can query these logs or aggregate log entries.

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.

Complete the following steps before moving to the next stage.

- **Identify the logging framework.** The Data Engineering Team should find what logging framework or a 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.

The result of this stage is a list of data pipeline logs provided to the Data Quality Team. The logs should include both 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.
- **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. Data quality checks for these tables must be designed with a review and approval process to avoid data leakage.
- **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 establishing a connection requires additional configuration.
- **ETL platforms.** The ETL platform that is used for the full extract-transform-load process can log essential information at each stage. In addition, many ETL platforms can call on external services, such as a data quality platform.
- **Data ingestion tools.** The data platform can use basic data ingestion tools that have a 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, such as 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 at the time of day when data is expected to be present, but the data pipeline is not able to

trigger the data quality checks automatically when the data load operation finishes.

- **Business intelligence tools.** We can simply describe the value of data observability as a way to detect data quality issues before the business sees the wrong numbers. The place where end users see these wrong numbers are the dashboards generated by Business intelligence (BI) tools such as Power BI, Looker or Tableau. An end-to-end data quality process can include the use of robotic process automation tools to screen-scrape dashboards. The numbers displayed on the dashboards can be affected by additional filters and transformation logic applied to the BI tool's dashboard or data model. The BI tool may perform periodic refreshes, which may also fail. Aggregate numbers shown on dashboards can be compared with a data warehouse or system of record as part of custom data quality checks.

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 DQO).
- Additional firewall rules must be configured to access databases.
- Personal data must be handled with care.
- Data quality checks accessing tables with sensitive data must go through a review process. The data quality tool should support a review process, for example by requesting a pull in the source code repository. Data quality tools must then store the definitions as easily readable flat files (such as data quality check specifications written in YAML or Python rules used in DQO).

DE.5. Provide necessary credentials

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

Data access rights must be defined for both the data quality engineers and data quality tools. The data quality engineers will also require access using 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.

In case landing zones or ingestion zones are monitored through flat file queries directly as external tables, read and list 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 on the HDFS file system in Hadoop clusters.

The data quality tool may require installation on a dedicated server or instantiation of a new virtual machine. Server access rights are required for personnel responsible for maintaining this environment. Alternatively, the data quality tool can be deployed to a shared environment such as a Kubernetes cluster.

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, make sure that 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 it can be installed on 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. In this case, the corporate password rotation policy should be followed accordingly. The workload executed by the data quality tool on the monitored databases will be easy to identify and limit if the impact of running data quality checks affects the performance of the data platform.

- **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 would allow data manipulation, table management, or access rights should be avoided.

- **Provide access to the data pipeline platforms.** In case a data quality tool is integrated directly into the data pipelines, it is helpful to provide limited access rights to the data pipelines for the Data Quality Team. 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 any 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.

Note the following problems that may arise at this stage.

- Credentials are invalid.

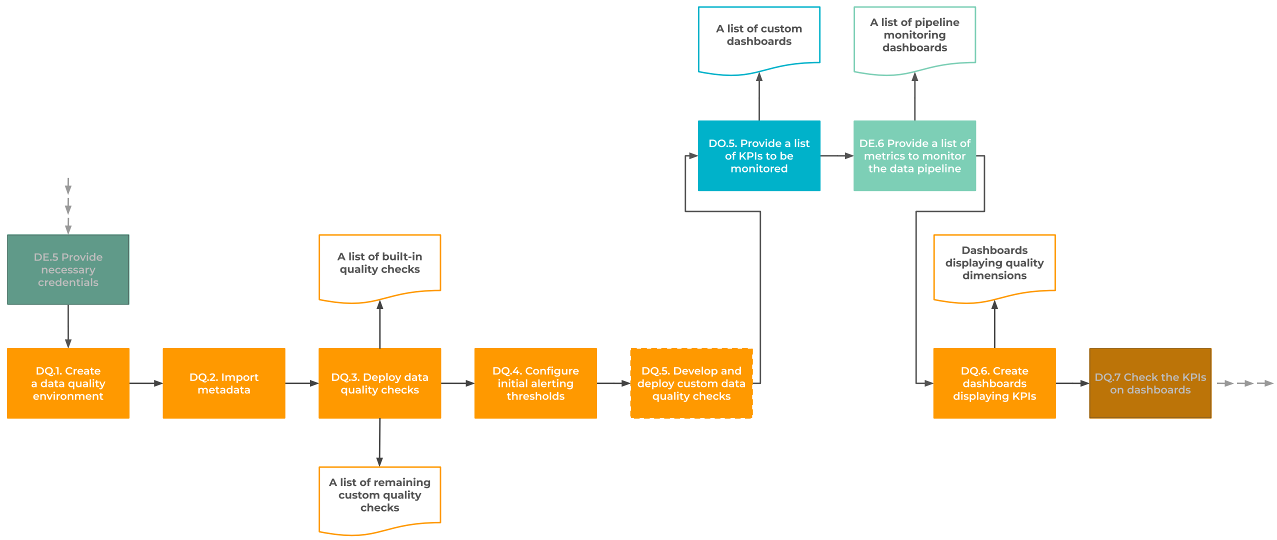
- Access to the file storage (S3 Buckets, Azure Blobs, GCP Buckets) has not been granted.

- 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



By now, a list of monitoring tables and columns is defined. Data quality checks requirements and KPIs are also agreed upon with the Data Owner and Data Engineering Teams. All obstacles that might delay the implementation of the data quality control tool should have been identified and addressed. Also provide the credentials required to connect to monitored databases, data warehouses and data lakes.

At this stage, the Data Quality Team should be able to activate the data quality monitoring. This chapter describes the following steps:

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

At the end of this stage, the data quality platform can monitor data quality, but 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 configuration of the firewall whitelisting rules. As the first step, the supplied credentials and connection details must be verified. 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 must be followed that are specific to the type of scheduler (Apache Airflow, Prefect). 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.

To successfully build a reliable data quality environment, make sure that you pay attention to the steps described below.

- **Optional cloud environment configuration.** Provide and configure the cloud environment, such as accounts, billings, projects, subscriptions and instances.
- **Deploy the data quality platform.** Follow the documentation of the data quality platform, in particular, to meet the 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 will not be able to 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 such as 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. DQO 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.

- **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 who are 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.

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

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.

- Prolonged approval process of integrating SaaS data quality solution 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, such as the number of days a table has been refreshed on time, should be associated with the table name. 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.** A data catalog tool may already be implemented in the organization. Many important guidelines may already be provided in data cataloging tools, such as Alation or Amundsen. 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 longer 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.

- **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 are not 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, DQO solves the challenge of monitoring date-partitioned tables by calculating separate metrics for each "day of data". KPIs which are 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 time of the synchronization exist. Synchronization on the next day will load the missing data, but 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. In case 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.** For daily partitioned data, previous partitions may be updated. It is also possible that the data comes from a source that is generated manually, for example, as a Microsoft Excel file. The file may be changed several times and will be 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. The number of rows changes when missing rows are loaded. More complex discriminators can be calculated from the data. These 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. Columns that are expected to have some percentage of null values should be identified. You should also import and identify columns that store aggregate measures, such as sales data or the number of impressions.

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 files or columnar data files (Parquet, ORC) are corrupted, making some partitions unreadable.
- No documentation of the data model.
- 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 the activation of data quality sensors that collect data quality metrics from the monitored sources. This should be followed by the configuration of the alerting thresholds, as described in the next step of the process (DQ.4. Configure initial alerting thresholds).

The data quality sensors capture 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. The sensors can be implemented as templated SQL queries (DQO uses Jinja2 templating engine) or as a 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 DQO.

The metrics captured by the data quality sensors should be stored in the data quality database for further analysis. Time series analysis used to detect the anomalies in the dynamics of the changes in the data set requires historical data. A data quality sensor that captures the current row count of the table, which is scheduled daily, would build a full history of the table row counts over a longer 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. DQO 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.

Two more important aspects must be considered while configuring the data quality checks. The first aspect is the time slicing of the table which will be monitored. A majority of data quality solutions are limited to capturing data quality metrics for the whole table, without taking into consideration that the old data is measured together with the most recent data. This limitation has serious implications, making many data quality results incorrect.

Let's consider a simple data quality check that counts the percentage of rows with a non-negative value. A data quality sensor that analyzes the whole table without time slicing, detecting a percentage of valid rows where the value of a tested column is greater or equal to 0, would run a SQL query similar to the following.

```
SELECT CURRENT_DATETIME() as time_window,
    100.0 * SUM(CASE WHEN tested_column >= 0 THEN 1 ELSE 0 END) /
    COUNT(*) as percentage_valid
    FROM schema.table
```

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

time_window (metrics capture timestamp)	percentage_valid
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 both 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, daily or monthly partitioned data that is reloaded should be analyzed separately, for each daily partition.

A data quality platform that takes into account time windows (time slicing) should support the calculation of data quality scores for each time period separately. DQO 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 aforementioned SQL query to capture time-sliced data to calculate metrics for each day separately (Google BigQuery example).

```
   SELECT DATETIME_TRUNC(transaction_timestamp_column, DAY) as time_window,
       100.0 * SUM(CASE WHEN tested_column >= 0 THEN 1 ELSE 0 END) /
    COUNT(*) as percentage_valid
    FROM schema.table
GROUP BY time_window
```

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_window	percentage_valid
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 easily identify a significantly large drop in the percentage of valid rows on 2022-10-08, which is below the average of ~95% of valid rows each day before the day of the data quality incident. A score in the query which did not group the data by day and calculated an aggregate score for the table only detected a drop to 92.76% which was not too far from the average score. The examples above show just 5 days of data, but in a real database, that drop will be below the average daily variation of the metric's value.

The second most 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 different sources, countries, states, or received from different external sources. Each stream of data would be loaded by a different pipeline. Data pipelines for different data streams may fail independently from each other. Data streams can be identified by a discriminator column, such as country or state. A data quality platform that can analyze data within separate segments (such as DQO) should be able to add a GROUP BY <data_stream_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 CURRENT_DATETIME() as time_window,
    100.0 * SUM(CASE WHEN tested_column >= 0 THEN 1 ELSE 0 END) /
    COUNT(*) as percentage_valid,
    country as stream_level_1
    FROM schema.table
GROUP BY stream_level_1
```

The results pivoted for readability might look as follows:

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

Data quality scores, which are 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 stream slicing. The GROUP BY clause must list columns that divide the data set by a data stream 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 DATETIME_TRUNC(transaction_timestamp_column, DAY) as time_window,
       100.0 * SUM(CASE WHEN tested_column >= 0 THEN 1 ELSE 0 END) /
    COUNT(*) as percentage_valid,
    country as stream_level_1
    FROM schema.table
GROUP BY time_window, stream_level_1
```

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_window	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%	

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

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

- **Configure ad-hoc data quality checks.** These are the basic data quality checks that can be executed at any point in time, activated on an ad-hoc basis, or scheduled for periodical evaluation. Evaluation of these data quality checks can also be triggered by a data pipeline when data has changed. The ad-hoc data quality checks in DQO do not use time series. When the data quality check is evaluated, it always verifies the entire table by running a full table scan query. The only possible type of filtering is a static filter clause, which is appended to the WHERE clause in the generated SQL SELECT statement.

- **Configure data quality checkpoints.** Checkpoints are scheduled data quality checks that are evaluated at regular intervals and measure data quality over a fixed time period. Supported time periods are hourly, daily, weekly, monthly, quarterly, and yearly. Only the most recent data quality result is stored for each time period, overriding previous data quality results for that period. The checkpoints in DQO are similar to regular data quality checks. They also perform a full scan of the monitored tables, detecting any data quality issues independently of the timestamp of the event in the affected row. If a daily data quality checkpoint is re-evaluated during the day, DQO 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. Data quality checkpoints are used to track data quality scores over time and count data quality alerts when measuring the data quality KPI score. In addition, data quality checkpoints support anomaly detection for time series. DQO can detect potential data quality issues when an anomaly is detected among regular daily or hourly data quality readouts.

- **Configure date-partitioned 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. DQO supports calculating separate data quality scores for each daily partition. A different data quality readout and alert are generated for each daily partition. The data quality check uses a GROUP BY <date_partitioning_column> condition to analyze each day (daily time period) separately. 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 clearly identifies a timestamp of the event, this column can be used to execute daily-partitioned data quality checks. DQO truncates timestamps to the beginning of the day.

- **Configure data stream 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 stream hierarchy configuration. DQO supports the data stream hierarchy up to 9 levels of nesting. Data stream 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). Example data stream hierarchies: country, country/state, continent/country/state, country/department, etc.

- **Configure incremental loading.** Very big tables (terabyte or petabyte scale) are most likely time partitioned. It is unlikely that the old data will be updated frequently. Because of that, it's possible to reduce the query execution time and the cost of executing queries on "pay-per-use" platforms such as Google BigQuery can be reduced by including additional filters. A WHERE condition applied on the timestamp column may limit the range of scanned data to a few most recent time periods (days, hours, etc.). Incremental execution of data quality checks is especially important for date- and time-partitioned tables, when old data does not change and only the most recent data should be checked for data quality issues.

- **Schedule data quality sensor execution.** Data quality sensors that are not activated at the end of the data loading pipeline should be scheduled by the data quality platform internally 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 new data to be present.

- **Verify access rights.** This action will help you make sure that 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 requiring custom data quality check implementation.** Not all data quality requirements may be handled by built-in data quality checks. A list of data quality requirements that would calculate business-relevant metrics must be identified.

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 a scheduled data quality check will not see the most recent data.
- Some data quality checks may require custom implementation.

The results of this step are two lists of data quality checks. The first is a list of built-in data quality checks to which you will need to select the appropriate data quality rules. The second list contains remaining custom quality checks that will require designing new data quality sensors and rules.

DQ.4. Configure initial alerting thresholds

The data quality sensor readouts are captured and stored in a data quality database to represent the historical measures of the monitored data source. When using just the raw data quality sensor readouts, identification of data quality issues is possible only visually on charts and dashboards.

Full automation of data quality verification requires creating data quality alerts for all values that are outside the limits or have been 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 tracking data quality results for different data streams 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 a possible data stream (identifying the data source).

In DQO, a data quality check is divided into two parts. The first part is the data quality sensor, which reads the value from the data source at a given point in time (e.g. the number of rows in a table). The second part is the data quality rule evaluation, which takes into account the history of data quality sensor readouts over a longer time period (weeks or months). A basic rule can simply score the most recent data quality result if the value is within the expected range. A standard data quality check on a table that counts the number of rows uses a simple "greater than 10" rule to instantly raise data quality alerts when the table has fewer than 10 rows. 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. **Single value** - evaluate the current data quality sensor readout.

    - **Data quality sensor value equals X.** The threshold detects that the value does not change. It is useful to detect that the number of columns in a table is constant.

    - **Data quality sensor value does not equal X.** The threshold is mostly used as "not equals 0" to detect that a table is not empty or there are no rows with invalid values.

    - **Data quality sensor value is (not) greater/less than X.** The rule identifies problems that a table is not empty (row count > 0) or there are no rows with a null value in a column (the number of nulls in a column cannot be greater than 0).

    - **Data quality sensor value is between X and Y.** The rule is useful for detecting that a numerical value is within a range. For example, the percentage of valid rows in a table is between 95% and 100%.

2. **Relative value** - compare the current readout with a similar value a day/week/month/year ago, ignoring seasonality.

    - **Data quality sensor has not changed since the last readout.** The rule detects that many columns have not changed since yesterday (or the previous time of the readout).

    - **The change from a similar time window has not changed significantly.** The rule compares the most recent data quality sensor readout (such as the current number of rows) to a similar value exactly one week ago. For example, Mondays are compared to Mondays, Tuesdays to Tuesdays. This type of rule avoids the effect of seasonality on data volatility.

3. **Time series** - compare the current readout with the historic values within a broader time window (weeks, months).

    - **Compare the difference from the average value in percentage.** This is a relatively simple rule that is easy to understand. For example, the current row count should not differ more than 20% from the average or a daily row count increase should not be bigger than 200% of the average daily increase.

    - **Compare 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 that are further than 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 simpler 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 statistics or even seasonality analysis of time series to detect anomalies and outliers. Possible algorithms in this category are ARIMA or Prophet.

One more concept should be mentioned here. The data quality alerting has some similarities to general logging, used in logging libraries. Not all alerts or anomalies are equal, thus 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.

DQO supports three alert severity levels:

- **Warning.** Data quality checks can have a warning level alerting threshold that raises warnings for less important data quality issues. Warnings are not treated as data quality issues. Data quality checks that did not pass the warning alerting rule, but did pass the error and fatal alerting rules are still counted as passed data quality checks and do not reduce the data quality KPI score. Warnings should be used to identify potential data quality issues that should be monitored. 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 the "error" level in logging libraries. Data quality checks that failed to pass the rule evaluation at the "error" severity level are considered failed data quality checks. Errors reduce the value of data quality KPIs. 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.** This is the highest alerting threshold that should only be used to identify severe data quality issues that should result in stopping the data pipelines before the issue spreads throughout the system. Fatal data quality issues are treated as failed data quality checks and reduce the data quality KPI score. The fatal 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 data quality issue, the data pipeline should be stopped. For example, a percentage of data quality check monitoring null value may raise a fatal 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	x		
Error (default)		x	
Fatal		x	x

The process of selecting appropriate data quality rules and assigning the initial thresholds is described below.

- **Choose the most appropriate type of data quality rule.** The right 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 of the 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.

- **Assign the initial thresholds.** The threshold represents just the expectations and beliefs about the current data quality status. The Data Owner or the Data Engineering Team may believe that 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. The thresholds should later be adjusted to a more reasonable value. The default alerting thresholds are raising 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 just over-sensitive. The variability of the row count increase may not perfectly follow a normal distribution curve, so a standard deviation-based rule will detect more false positive errors.

- **Configure 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. It is possible to configure only a warning severity threshold for a data quality check 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 per day, as these types of issues might happen from time to time and are not always data quality incidents.

- **Configure alerting threshold at the fatal severity level.** Alerting thresholds for the most serious 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.

- **Configure additional alerting groups.** Similar to slicing the data by data streams used to track data quality results separately for different data sources, it should be possible to group alerts on similar tables to the same category. The database may have similar sets of tables that are covering different countries or business units. For example, each country may have a similar data mart that has similar fact and dimension tables. All tables (or activated data quality checks) from the UK could be tagged as "UK", while the US tables would use a "US" tag. Later, the data quality KPIs (percentage of passed data quality checks) can be tracked separately for each group. DQO supports grouping tables by configuring data stream levels as constant values, for example, "UK" or "US".

Note the following problems that may occur at this stage.

- Thresholds are configured too conservatively, which generates too many false positive alerts.

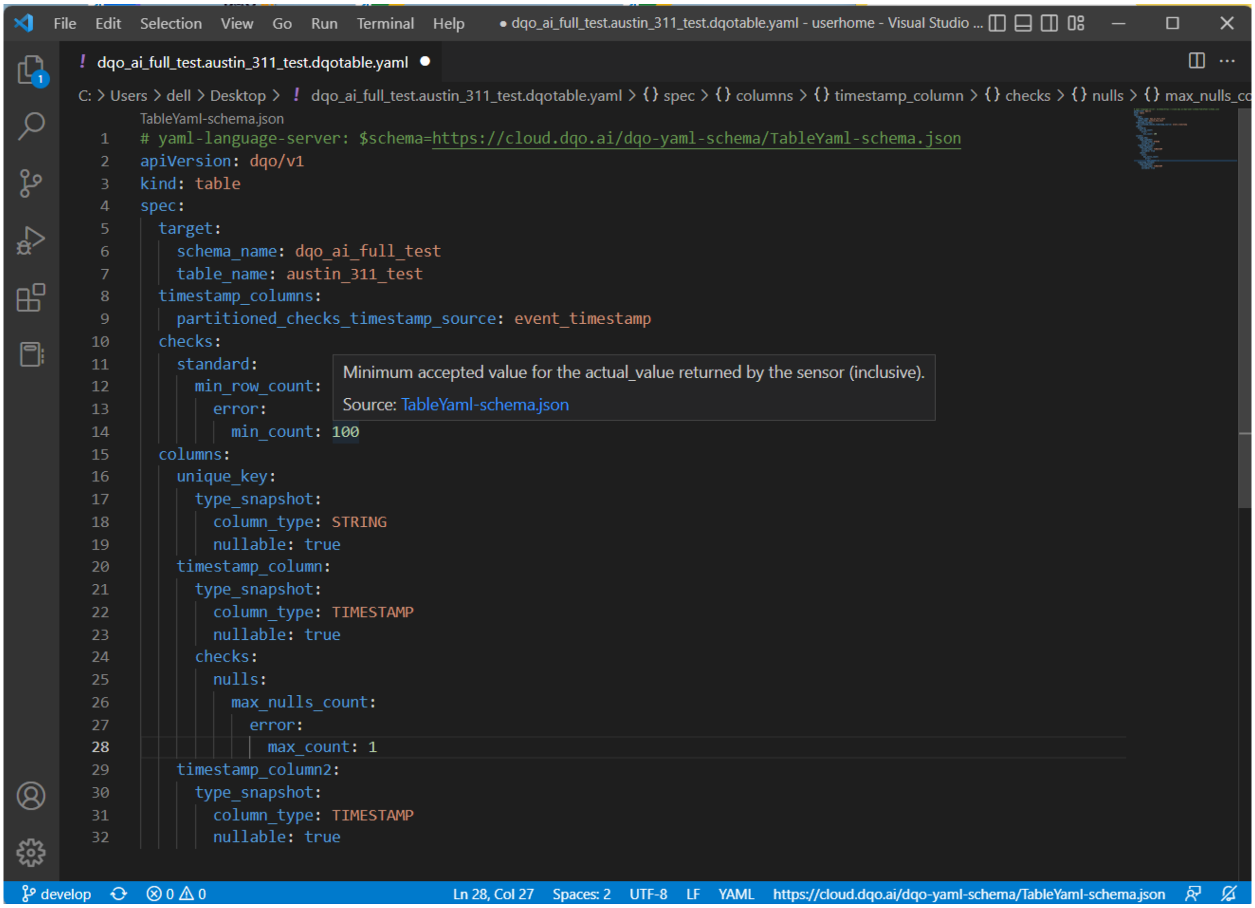
- The data variability does not follow a normal distribution curve, requiring tweaking the thresholds at later steps.

- Complex machine learning (ML) rules that are using time series analysis to generate alerts that 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 alerts for each day. 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. DQO uses hashing for detecting significant changes.

Another concept can also be used to limit the number of alerts that are forwarded to operations teams. Data quality alerts may be grouped into clusters of similar alerts. The data stream levels used for the data slicing (to identify different data streams) have proved to be the most usable way to group the same type of data quality check for all alerts related to the same data stream, regardless of the table being monitored.

Below is an example of how to configure data quality checks and rules using DQO. Data quality checks are defined as YAML files that 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.



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

Certain 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 make the best use of existing indexes in the database.

- **Custom filters.** If you need to 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, partitions that are up to a year old can be refreshed with updated data. Such partitions must be retested, but they fall far behind the incremental time window. To avoid performing full table scans every day, a more complex query can query a logging table to detect recently modified daily partitions.

A customizable data quality platform (such as DQO) should support the use of custom data quality check definitions, provided as custom SQL queries or implemented as a custom code. Custom data quality checks should be reusable across tables. Custom checks should not be 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 is counting 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_stream_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 parts of the template that are dynamically populated with the target table and column names are configurable. The data quality sensor template is rendered into an SQL query using the Jinja2 templating engine. Other parts of the query template render the proper GROUP BY clause and additional result columns required to support configurable time series (daily, weekly) and additional data stream slices.

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

- **Write a SQL query to pull metrics.** The prototype of a custom data quality sensor should first be implemented by writing a SQL query that can retrieve the correct metrics. The performance implications of executing the query are easy to identify at this step.

- **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 streams.

- **Register a custom data quality sensor definition.** The custom quality sensor template should be registered in the platform. In DQO, 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. DQO 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 historic 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:
    passed = rule_parameters.actual_value >=
rule_parameters.parameters.min_value
    expected_value = rule_parameters.parameters.min_value
    lower_bound = rule_parameters.parameters.min_value
    upper_bound = None
    return RuleExecutionResult(passed, expected_value, lower_bound,
upper_bound)
```

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 DQO, 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.

DQO stores the result of executed data quality rules for both passed (no alert raised or only a warning raised) and failed (errors or fatal alerts raised) data quality check evaluations. Data quality checks can define the alerting threshold at three severity levels: warning, error and fatal. 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 streams 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 DQO:

- **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. A data quality KPI can be calculated from multiple tables at once. In DQO, such a configuration is provided as a static value assigned to a data stream level. Here is an example of the required data stream configuration in a YAML file:

```yaml
apiVersion: dqo/v1
kind: table
spec:
  target:
    schema_name: public
    table_name: fact_sales_uk
  data_streams:
    by_country:
      level_1:
        source: static_value
        static_value: 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 stream. Here is another example of a DQO YAML file that uses a "country" column to identify separate data streams for separate data quality KPI calculation:

```yaml
apiVersion: dqo/v1
kind: table
spec:
  target:
    schema_name: public
    table_name: fact_sales
  data_streams:
    by_country:
      level_1:
        source: dynamic_from_group_by_column
        static_value: country
```

Data quality KPIs can also be calculated for combinations of data sources (data streams), 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 be also 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 that is 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 data that is missing due to outdated credentials or corrupted files. If the data sharing agreement describes the format of the data, 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 ensure that all data quality requirements in the data sharing agreements are met.

- **KPIs 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 that are 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 that is delivered to other departments.

- **Multiple copies of the same data.** The same data can be stored in different databases and data warehouses. Data quality KPIs between 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 the data quality of the source database (the old database that is decommissioned). Similar tables in the old and new databases should be tagged with the same data stream label. All tables in the old database should be tagged as "old", and all tables in the new database should be tagged as "new".

- **Data mesh.** Data may be distributed across different data lakes. It is important 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.

Note the problems that may happen at this stage.

- A discriminator column that identifies different data streams has many distinct values which 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 it is difficult to keep the data quality KPIs up-to-date with the current data model.

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. Many data quality issues can be caused by bugs or failures in data pipelines or ETL processes. A data quality platform can identify these issues because most of these 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.
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 from day to day, and on some days it can get very long. Tracking the current delay and comparing it with the average delay detects these issues.
The data pipeline failed or was canceled during execution.	Consistency	The average number of rows loaded per day drops below the average.

Data pipeline issue	Data quality dimension	Data quality check
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 that there is 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.

Data pipeline issue	Data quality dimension	Data quality check
The number of columns has changed in the file.	Integrity	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.
The order of columns has changed in the file.	Integrity	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.	Integrity	The hash code of all column names and their data types can be tracked. Any changes to the hash will instantly identify a change.

Data pipeline issue	Data quality dimension	Data quality check
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. 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.
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 important 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 important issues.

- **Define mapping of existing data quality checks to reporting categories.** Existing data quality checks that are performed 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 by mapping them to data streams. 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 that are 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 stream 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. DQO 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 stream). 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 or Slack channels.

- **Integrate with a ticketing system.** A ticketing system such as Jira or ServiceNow may already be in use by the Data Engineering Team. When a certain data quality issue is identified, a ticket should be opened 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.

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.

Data quality dashboards should be divided into multiple groups, depending on the audience and purpose of these dashboards:

- **Governance views.** These views show high-level data quality KPIs, aggregated on a macro scale. Senior management should be able to review key data metrics, such as the timeliness of data for the entire month, by organizational units, vendors, data providers, or subsidiaries. The governance views show the key data quality KPIs for the current time period at the business unit, vendor, supplier, or subsidiary level, compared with the metrics for the preceding time period. Data quality scorecards are the highest level governance views that should be shared at a corporate level.

- **Operational views.** Operational views enable the Data Engineering Team and the Data Owner to identify the areas (tables or data pipelines) in the data warehouse or data lake with the highest number of data quality issues that should be addressed. The Data Quality Team assists in this process by reviewing any false positive alerts.

- **Detailed views.** This type of data quality dashboard should show detailed information at the table or column level. An example of a detailed view is a dashboard that shows the row count history for a single table. The detailed views are used by the Data Engineering Team and the Data Owner to better understand data dynamics during the investigation phase when the data quality issue is being diagnosed and later to confirm whether it has been resolved.

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

- Clear logical layout and ease of understanding.

- Enables export of data to an Excel file when it is necessary to share information about a data quality issue with additional people or external partners, who do not have access to the dashboard.

- 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, without a split into different data quality dimensions (timeliness, validity, etc.) may be misleading. The number of active data quality checks in each data quality dimension may be different. 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 should use weights for each data quality dimension. An example of such a calculation is presented in the table below. , calculated as in the table below:

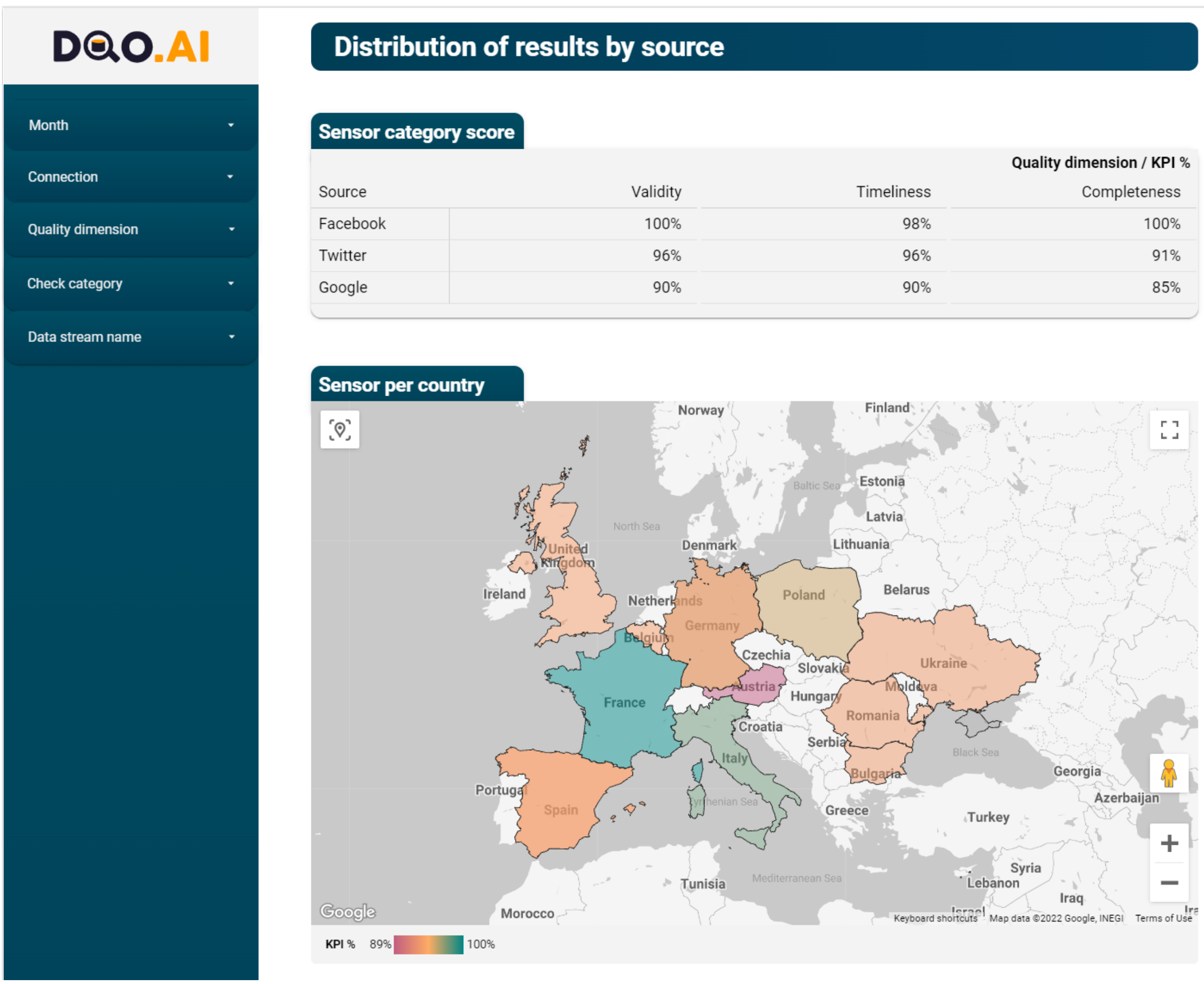
Data quality dimension	**Timeliness**		**Validity**		**Accuracy**		**Weighted KPI**
KPI per dimension	95%		80%		90%		
	*		*		*		
Weight	20%		50%		30%		
	=		=		=		
Weighted KPI	19%	+	40%	+	30%	=	89%

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.

DQO 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 views and data quality alerts aggregated on the governance and operational views, 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 example shows some basic principles of how a dashboard for a governance view should be implemented. On the dashboard below, you can see the following elements:

- Current time period, which is limited to one month, selected from the filter.
- A breakdown by the data quality dimensions, such as timeliness, validity, and completeness.
- Separate KPIs for each data source, identified by tagging activated data quality checks with the name of the data source (e.g. Twitter, Google, Facebook).
- A grand total of KPIs, calculated for the entire data source.
- Additional charts or visualizations provide additional information and allow users to apply filters on the displayed measurements. In this example, a filter on a selected country can be applied by clicking this country on the map.



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 warehouse is provided by the data quality platform. With DQO, each customer receives a private data quality lakehouse in Google Cloud.

- **Select a business intelligence environment.** Choose from a variety of commercially available BI technologies such as Sisense, Tableau, Power BI, or Looker Studio. Data quality projects implemented with DQO 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 be connected to the data quality database. DQO customers can ask the vendor to access a custom Looker Studio data source, providing direct access to the data quality lakehouse.

- **Select relevant data streams for aggregation of data quality KPIs.** The relevant data stream 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 views.** The development of custom data quality dashboards should be planned 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.

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 checks responsible for the alerts and creates a list of tables affected by the issue. Next, the data quality team 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.

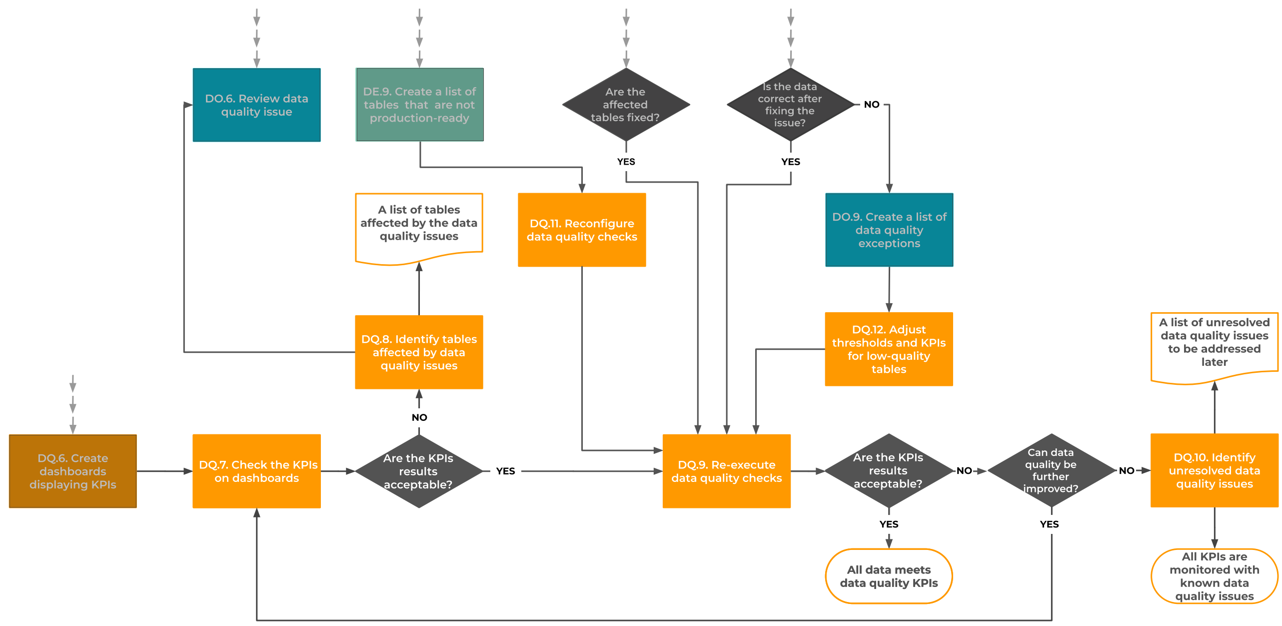
Sometimes a data quality issue cannot be resolved promptly or its resolution requires a manual update of invalid records. For example, a data quality requirement that the phone number is provided for every customer in the CRM cannot be easily fixed without engaging the sales team in a lengthy CRM data cleaning. In that 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 would still identify further 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 further 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 data quality checks that are active 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 adjusted to the actual quality of data. 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 quality issues present at the data source level can be fixed by the Data Producer. 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 perform a further investigation only if the data quality KPI deteriorates over time.



DQ.7. Check the KPIs on dashboards

The data quality KPI dashboards indicate areas with data quality issues to focus on. The data quality dashboards should be divided into three groups:

- **Governance views.** This view shows the global data quality KPIs, divided by the data quality dimension and also the total percentage of passed data quality checks out of all executed checks. During the data quality KPIs review, only the governance views are important. Prioritization of issues that should be fixed is handled in the operational view.
- **Operational views.** This view shows 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 views.** This view enables in-depth investigation of the data quality issue by reviewing the historical data quality sensor readouts or reviewing anomalies in

the data quality. This can be, for example, a sharp increase in the percentage of rows with null values.

The governance views 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 that are aggregated in a single table. DQO enables the use of up to 9 data stream hierarchy levels for tagging and segmenting 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 views 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 that is 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.

- **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 views 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.

Note the following problems that may occur at this stage.

- Additional data quality KPI dashboards must be created for selected data stream hierarchies. For example, the data quality results are measured at the country and state levels. Three kinds of data quality KPI dashboards are possible with these

- Hierarchies: KPI at a country/state level, KPI at a country level and an overall KPI at the whole organization level.

- Incomplete data (such as missing days) affects the KPI calculation because there KPI is calculated from 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 there are any data quality KPIs that 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 views, the Data Quality Team can use the operational and detailed views 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. The Data Quality Team should facilitate the resolution of the issue by working together with the Data Owner, Data Producer, and Data Engineering Team.

The data quality KPI is only a percentage of passed data quality checks, and quite often alerting thresholds are configured too restrictively. This may result 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 a data quality check that compares 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 for a single data area to be improved. This may simply 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.** Tables affected by data quality issues should be sorted by the number of data quality 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, and warning severity levels.

- **Review fatal issues.** First, you need to review the tables affected by data quality issues at the fatal severity level. These are the most significant data quality issues that affect the 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 it is replicated to downstream systems. When the Data Quality Team identifies any fatal 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". Tables affected by regular data quality issues should be sorted by the number of errors to focus on the most affected tables first.

- **Review data quality warnings.** Alerting thresholds that trigger warnings should be reviewed for anomalies. A recent change to the percentage of rows with null values may suggest a potential data quality issue, which could cause more severe problems in the future, had it not been addressed preemptively.

- **Make a preliminary assessment of the data quality issue.** Review in detail the data quality alerts raised in the tables under investigation. Data quality sensors in DQO 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, it is also likely that there are 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 very easy to notice on a query result screen after looking at the contents of the table.

- **Prepare an issue summary for review with the Data Owner or Data Engineering Team.** Data quality issues that are not false positives should be fixed by the Data Owner, Data Producer, or Data Engineering Teams. Additionally, frequent data quality issues that happen from time to time should be reviewed. Occasional data quality issues are easy to identify because they lower the data quality KPI. The Data Quality Team should prepare a report summarizing the results of the investigation.

- **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.

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 view 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 the data quality issues and reviewing the data quality KPIs. The step in the middle is the data quality check re-execution 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 most 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 that has a different schema and a new table 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.** Partial or full reload of large day-partition tables should be followed with data quality check re-execution for the partitions affected by the changes. The range of affected daily partitions should be determined by the Data Engineering Team.

- **Re-execute selected data quality checks.** Either selected data quality checks that have been updated or all data quality checks on modified tables should be re-executed. DQO supports two types of time series that affect the re-execution process. These types are point-in-time results for the entire table and 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. Previous data quality scores and alerts are preserved in the data quality database. This differs from the way DQO handles data quality results for date-partitioned data because the data quality results (sensor readouts and alerts) for the past dates will 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.

- **Clean up outdated readouts and alerts.** The data quality database may contain outdated or duplicate data quality results for data quality checks that have been disabled or tables that have been removed from the monitored database. The Data Quality Team should remove these results. If these results 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 once again review the data quality KPIs. If new issues are identified or the data quality KPIs are still not meeting the expected values, it may be necessary to repeat the data improvement process by going back to stage DQ.7. Check the KPIs on dashboards.

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 complex data quality check structure overhaul.
- Many data quality checks are outdated, making it time-consuming to remove old data quality results from the data quality database.

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. Within these constraints, you can set up initial data quality metrics, optimize alerting thresholds and fix data quality issues. The remaining data quality issues that are still open at the end of the data quality project must be addressed at the 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 that were 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 would 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 that are 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.

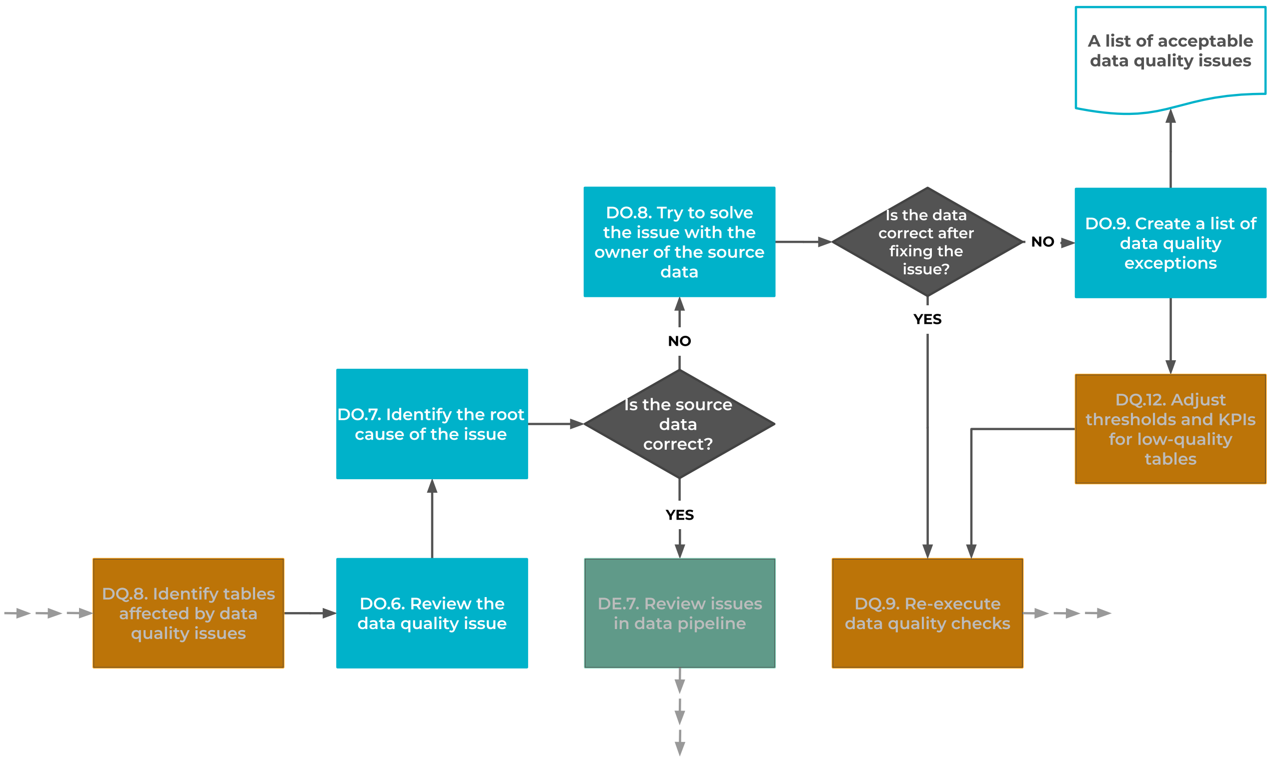
The responsibilities of the data quality operations team will be similar to the tasks described in this document, but with yet another additional responsibility. The data quality operations team should continuously review the data quality KPIs at the end of each reporting period and compare the weekly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

5. Fixing source data issues

Fixing source data issues involves close cooperation between the Data Quality Team that has identified the issue, the Data Owner who understands the data model, and possibly also the Data Producers who may correct the source data.

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

However, if there is a problem with the source data, the Data Owner tries to solve it with the Data Producer, even involving external data suppliers or business users. If the problem cannot be resolved, the Data Owner may create a list of acceptable data quality exceptions. 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 raised to match a threshold that will be monitored from that time on.



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 have in-depth knowledge 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 data quality KPIs that have been affected. 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 table schema and the purpose of the data stored in the tables must be well understood by the Data Quality Team and the Data Owner. Both parties should review all relevant documentation that can 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, but 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 of the issue is performed.

Note the problems that may arise at this stage.

- The Data Owner is new to the organization and does not have a holistic view of their domain of responsibility.
- The summary of the data quality issue presented by the Data Quality Team is incomplete.

- The data quality issue presented to the Data Owner is simply the result of incorrect data quality check configuration and it lies with the Data Quality Team. It should be fixed by the Data Quality Team at the next opportunity when the data quality checks are reconfigured and re-executed.

DO.7. Identify the root cause of the issue

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

When 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 make sure that the problem is not caused internally during the data ingestion. It is also possible that the problem with the data received from an external source may have been anticipated beforehand, in case the external vendor had announced a major update in their release calendar, doing so in a clear manner and in advance. The announcement of a breaking change in the file format, data model, or API specification might have remained unnoticed ahead of time. Also, the code changes to the data pipelines were not finished on schedule, before the upgrade of the source system.

The following steps will help you identify the root cause of the 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. The format of the date and time stored in a text column has 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 even 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.** In case the source table or source file can be identified, the Data Owner should also 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 and 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, it can be challenging to find all the relevant information displayed on the application's user interface by 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.** The log of previous issues should be consulted 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.** The data quality issue may have been foreseen ahead of time if the source system was down due to a scheduled maintenance operation. In addition, the source tables may be populated by a nightly job that performs a full data refresh. If the data quality check is executed in the middle of a full refresh operation, it may analyze partially loaded data, which will raise 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 are comparing the most recent data quality readout with historical readouts may identify anomalies that are nevertheless acceptable. If the alerting thresholds for such data quality checks that detect anomalies (such as the difference from the mean value) are too sensitive, the Data Owner will have to handle many false positive alerts. The Data Owner may decide to lower the KPI level (percentage of passed data quality checks) which should be followed by a data quality investigation.

- **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 take into account the decreased activity in the company 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 does not have access to 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, the Data Owner must decide whether the data quality issue is present in the source data or is only present in the data transformation process. Data quality issues present in the data source must be escalated to the Data Producer. All other issues are internal and can be fixed by the Data Engineering Team when the data has been incorrectly transformed or by the Data Quality Team if the data quality checks have been misconfigured or configured with too sensitive alerting thresholds.

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

The data quality issues originating at an external vendor or present in a line-of-business application must be addressed with the respective platform owner (the Data Producer). This can be an internal platform owner, such as a CRM owner. It can also be a business process owner if data quality issues are caused by missing steps in the process or users not executing the process correctly, for example, by entering incorrect data into the system.

The data quality issues already found in datasets shared by an external vendor must be consulted and fixed by the vendor's engineering team.

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 can be viewed by an external party 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 deadline for fixing the issue and the milestones for the implementation steps must be agreed on with the external party.
- **Identify a point of contact on the Data Producer's side.** The Data Owner should know who is the right person 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 is no longer in control of the issue resolution. 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.** A solution applied by an external party may have a disruptive effect on 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 cannot 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 corrected data is loaded by the Data Engineering Team. Issues that cannot be resolved, have to 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 on how to address the remaining data quality issues:

- **Update the data model.** Changes to the data model may be required to address the problem. The Data Engineering Team must be involved in the process because data pipelines and ETL processes will need to be changed. Column lengths or column data types may need to be changed to accept rejected or truncated values.

- **Change the data processing logic.** The data processing logic at the data transformation stage may require changes. Additional pre-processing, joins or post-processing steps can be added.

- **Decommission tables.** Tables that must be replaced by another table that has a different schema should be decommissioned. All data consumers who use the old version of the table must be informed about the change. The decommissioning process must be planned in advance.

- **Apply temporary solutions.** If a lot of time is needed to address a data quality issue, suggest an interim solution. The business users may be asked to stop using affected data quality dashboards until the data can be relied on.

- **Disable data quality KPI monitoring.** The data quality monitoring can be turned off or postponed at the data quality KPI level if the KPI (percentage of data quality alerts) is high and changes too frequently.

- **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 a discussion 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 that have no business impact should be monitored, but should not be counted as failed data quality checks, lowering the overall data quality KPI score. DQO supports the configuration of alerting thresholds at three severity levels: fatal 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 are not able to correctly detect data quality issues, it should also be possible to change the implementation of these checks. Since DQO is an extensible data quality platform that uses 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 take into account the partitioning of large tables. This would enable efficient partition elimination conditions in the data quality sensor query.

- **Customize data quality alerting rules definitions.** DQO 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.** Important data quality issues that require significant resources to fix should be planned as a project. Data quality checks can be temporarily disabled until the problem is solved or excluded from data quality KPI calculation.

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 be aware of how to adjust their business processes so that 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 the dashboard might cause 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 on the training data sets.

- **Downstream data consumers.** Low-quality data sets that are 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 to monitor invalid data, the data quality issues that were 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 periodical 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 views (top-level data quality KPI views). 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 must be customized or custom data quality sensors that must 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 that false positive alerts are no longer raised.

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

Based on consultations with the Data Owner, the Data Quality Team should have all the information necessary to apply changes to the data quality platform. The list of data quality exceptions prepared at the previous stage will turn into configuration tasks to be implemented by the Data Quality Team. The Data Quality Team deactivates data quality checks that are outdated. When a table is exempt from data quality monitoring, all data quality checks must be removed.

Deactivation of data quality checks will prevent them from being executed in the future, so data quality alerts would no longer be raised. However, alerts that have already been raised would still be stored in the data quality database, reducing the data quality KPI score calculated as a percentage of passed data quality checks. In the event that alerts already raised by these data quality checks are considered false positives, the Data Owner may decide to remove them from the data quality database. In this case, the Data Quality Team should remove alerts and irrelevant data quality sensor readouts directly from the database.

Since decommissioning entire tables from data quality monitoring is a common task, the DQO data model is designed to simplify this task. Data quality results are stored as Parquet files, partitioned by connection name, table name, and month. For example, alerts for September 2022 for a single table would be stored in a file .data/rule_results/c=conn_bq_17/t=austin_311.311_service_requests/m=2022-09-01/rule_results.0.parquet. Thanks to that, data cleaning is as simple as removing the file.

The removal of data quality results is not limited to decommissioning data quality checks. When data quality sensor configuration parameters or alerting thresholds are significantly modified for date partitioned data that is analyzed using a matching time dimension, data quality checks should be recalculated for all affected daily or monthly partitions. Outdated data quality results should be removed when changes have also been applied to the configuration of the data stream hierarchy. DQO 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 in the way data quality KPIs are calculated or how KPIs are aggregated across stages, connections, and data streams may also require changes to the data quality dashboards that are in use. The Data Quality Team must review all these dashboards and plan all modification tasks.

The data quality checks decommissioning process has the following steps.

- **Deactivate data quality checks.** First, deactivate data quality checks. DQO supports two ways of deactivating data quality checks. The checks can be removed from the data quality specification YAML files or they can be 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 DQO. 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, which will preserve 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 (readouts and alerts) should be removed, otherwise, they will affect data quality KPI calculation.
- **Change the implementation of data quality checks.** Some changes may require changes to the implementation of data quality sensors or data quality alerting rules. DQO supports the customization of built-in data quality checks by copying the implementation of these sensors and rules from the DQO distribution into the user's home folder. 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. It 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.

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 up to this time went unnoticed.
- Implemented changes to KPI thresholds are too substantial, making the results unrepresentative.

6. Fixing data pipeline issues

The data quality issues that were reviewed by the Data Owner may have been present in the source data or may have been introduced in the data pipeline during the data loading process. Moreover, some data quality issues that were 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 full reload of updated source data. It is also possible that the required changes to the data pipelines are 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.

DE.7. Review issues in the data pipelines

The Data Engineering Team should review the data quality issue and find the reason why the problem was introduced by a data pipeline. Changes to the data pipeline or the changes to 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 did not execute for particular days.
  - Several files were corrupted or the data pipeline was unable to load them.
  - The data pipeline randomly fails during processing, skipping some files or loading target tables partially, until the pipeline 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 the values were out of range for the length or precision of the target column.
- **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, loading source data 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 as a result of 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 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 that have been 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 with care. After all, target tables are used by downstream data pipelines or by data consumers such as data science teams or business intelligence teams.
- **Plan changes to the data pipelines.** If the issue can be fixed entirely by updating the data pipeline, the changes should be planned. However, updating the data pipeline is not the only task. The target tables have to be reloaded, also 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. If the change affects critical business processes, it must be coordinated with all affected parties.

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

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 are 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 they will be susceptible to table removal or table model changes. For example, an external system may query the affected table through a database view. The view must be deleted and recreated in case of even slight changes in the table's schema.
- **Refresh data along the data lineage.** Tables and data marts that are downstream of the data lineage must also be refreshed if the data has changed.

- **Prevent the spread of corrupted data.** It is important to communicate changes to other stakeholders who might still be using corrupted data. They should be notified of the data refreshes or schema changes as soon as they happen.

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 that have been 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 issue with the data pipeline has not been fixed, the Data Engineering Team concludes that some tables are not production-ready and should be excluded from the data quality KPIs measurement. Tables are not production-ready in the following cases.

- **Under development.** Tables that are still being designed, prototyped, or frequently changed should be considered temporarily unstable. The Data Engineering Team should decide whether these tables should be monitored by the data quality platform. This will help the Data Engineering Team in ongoing development by providing live data quality findings. On the other hand, these tables can be excluded from the data quality KPI calculation because they have a negative impact on the data quality KPI reported from the governance view, diminishing the measures that represent the operational state of stable parts of the system.

- **Obsolete.** The tables that were recently obsoleted and are no longer used in business operations should also be excluded from the data quality KPI calculation. These tables will negatively affect 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. The Data Quality Team will be guided by these lists to reconfigure data quality checks.

- **Tables to be excluded from data quality monitoring.** Tables that are obsolete or are about to be significantly redesigned should no longer be monitored by the data quality platform. Any data quality issues reported on these tables are mostly incidental and irrelevant to the business.

- **Tables to be paused from data quality monitoring.** Tables that are 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. The evaluation of data quality checks for these tables should resume once the data is stable.

- **Replaced tables.** When changes to the affected tables also require significant changes to the table schema, the Data Engineering Team may decide to create a new version of the table under a different name. The Data Quality Team will have to switch all configured data quality checks from the old version of the table to the new one.

Some problems can also arise at this stage that are worth noting.

- The Data Engineering Team is unable to 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 is not able to 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 receives combined feedback from the Data Owner, the Data Producer, and the Data Engineering Team about required 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 need to 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 have been designed, the existing data quality configuration should be moved or copied to the new tables.

After applying changes 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 perform the following steps:

- **Deactivate outdated data quality checks.** There are three ways to deactivate data quality checks supported by DQO:
  - 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 data quality checks that have been modified 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 data quality KPI score as expected.

- **Push a new data quality configuration to a source control repository.** DQO stores the configuration of data quality checks as YAML files that can be easily pushed to Git or any other source control repository. 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 full 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".

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.

Reach a 100% data quality score

This eBook describes the proven data monitoring process that will help you remove all data quality problems. It was created by the DQO 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.

DQOps