**Function: main**

**Input Parameters:**

* **env** (str): Environment name (e.g., 'development', 'production').
* **application** (str): Application name.
* **module** (str): Module name.
* **run\_type** (str): Type of job run ('Scheduled', 'Adhoc', etc.).
* **unit\_nbr** (str): Unit number.
* **job\_set\_nbr** (str): Job set number.

**Description:**

* This function is the main entry point of the program.
* It performs various tasks, including connecting to databases, extracting data, executing data quality checks, and sending email notifications.
* It logs information and errors using the provided logger.
* The function is responsible for coordinating the execution of data quality checks and updating their statuses.

**Output:**

* The function sends email notifications and updates the status of data quality checks in the database.

**Summary of Imports:**

* **import cx\_Oracle**: For connecting to Oracle databases.
* **import pandas as pd**: For data manipulation and analysis.
* **import yaml**: For working with YAML configuration files.
* **import argparse**: For parsing command-line arguments.
* **import lg**: Custom logging library.
* **import pyodbc**: For connecting to ODBC databases.
* **import numpy as np**: For numerical operations.
* **from concurrent.futures import ProcessPoolExecutor**: For concurrent execution using processes.
* **from concurrent.futures import ThreadPoolExecutor**: For concurrent execution using threads.
* **import time**: For time-related operations.
* **from datetime import date, datetime**: For working with date and time.
* **import CommonFunctions**: Custom module for common functions.
* **import re**: For regular expressions.
* **import json**: For working with JSON data.
* **import requests**: For making HTTP requests.
* **import base64**: For base64 encoding and decoding.
* **from Crypto.Cipher import AES**: For encryption and decryption using AES.
* **import os**: For interacting with the operating system.

**Function: driver**

**Input Parameters:**

* **list\_of\_rules** (str): List of data quality rules to be executed.
* **cred\_config\_path** (str): Path to the YAML credentials configuration file.
* **job\_run\_nbr** (str): Job run number.
* **app** (str): Application name.
* **unit\_nbr** (str): Unit number.
* **job\_set\_nbr** (str): Job set number.
* **env** (str): Environment name.

**Description:**

* The **driver** function executes data quality rules based on the list of rules provided.
* It logs information and errors using the provided logger.
* The function returns an HTML tag containing the results of the executed rules.

**Output:**

* An HTML tag containing the results of the executed data quality rules.

**Function: insert\_dq\_rule\_status**

**Input Parameters:**

* **cred\_config** (dict): Credentials configuration.
* **job\_run\_nbr** (str): Job run number.
* **dq\_rule\_id** (str): Data quality rule ID.
* **application** (str): Application name.
* **dq\_run\_dt** (datetime): Data quality run date and time.
* **status** (str): Status of the data quality rule.
* **app** (str): Application name.
* **logger** (logger): Logger object.
* **env** (str): Environment name.
* **db\_connection\_mobius** (Database Connection): Connection to the Mobius database.

**Description:**

* The **insert\_dq\_rule\_status** function inserts the status of a data quality rule into the **TBLMOB\_DQ\_RULE\_STATUS** table.
* It logs information and errors using the provided logger.

**Output:**

* None

**Function: update\_dq\_rule\_status**

**Input Parameters:**

* **cred\_config** (dict): Credentials configuration.
* **status** (str): Status to update.
* **job\_run\_nbr** (str): Job run number.
* **dq\_rule\_id** (str): Data quality rule ID.
* **logger** (logger): Logger object.
* **app** (str): Application name.
* **env** (str): Environment name.
* **db\_connection\_mobius** (Database Connection): Connection to the Mobius database.

**Description:**

* The **update\_dq\_rule\_status** function updates the status of a data quality rule in the **TBLMOB\_DQ\_RULE\_STATUS** table.
* It logs information and errors using the provided logger.

**Output:**

* None

**Function: get\_blacklisted\_dqs**

**Input Parameters:**

* **cred\_config** (dict): Credentials configuration.
* **list\_of\_check\_id** (list): List of data quality check IDs.
* **job\_run\_nbr** (str): Job run number.
* **logger** (logger): Logger object.
* **app** (str): Application name.
* **env** (str): Environment name.
* **db\_connection\_mobius** (Database Connection): Connection to the Mobius database.

**Description:**

* The **get\_blacklisted\_dqs** function retrieves blacklisted data quality rules based on the provided list of check IDs.
* It logs information and errors using the provided logger.

**Output:**

* A DataFrame containing blacklisted data quality rules.

**Function: get\_dq\_check\_status**

**Input Parameters:**

* **cred\_config** (dict): Credentials configuration.
* **dq\_check\_id** (str): Data quality check ID.
* **job\_run\_nbr** (str): Job run number.
* **logger** (logger): Logger object.
* **app** (str): Application name.
* **env** (str): Environment name.
* **db\_connection\_mobius** (Database Connection): Connection to the Mobius database.

**Description:**

* The **get\_dq\_check\_status** function retrieves the status of a data quality check based on the executed check ID.
* It logs information and errors using the provided logger.

**Output:**

* A DataFrame containing the status of the data quality check.

**Function: insert\_dq\_check\_status**

**Input Parameters:**

* **cred\_config** (dict): Credentials configuration.
* **job\_run\_nbr** (str): Job run number.
* **dq\_rule\_id** (str): Data quality rule ID.
* **dq\_check\_id** (str): Data quality check ID.
* **check\_nm** (str): Data quality check name.
* **check\_desc** (str): Data quality check description.
* **status** (str): Status of the data quality check.
* **app** (str): Application name.
* **env** (str): Environment name.
* **logger** (logger): Logger object.
* **db\_connection\_mobius** (Database Connection): Connection to the Mobius database.

**Description:**

* The **insert\_dq\_check\_status** function inserts the status of a data quality check into the **TBLMOB\_DQ\_CHECK\_STATUS** table.
* It logs information and errors using the provided logger.

**Output:**

* None

**Function: get\_email\_hyper\_link**

**Input Parameters:**

* **cred\_config** (dict): Credentials configuration.
* **job\_run\_nbr** (str): Job run number.
* **logger** (logger): Logger object.
* **app** (str): Application name.
* **env** (str): Environment name.
* **db\_connection\_mobius** (Database Connection): Connection to the Mobius database.

**Description:**

* The **get\_email\_hyper\_link** function retrieves an email hyperlink based on the provided job run number.
* It logs information and errors using the provided logger.

**Output:**

* An email hyperlink.

**Function: execute\_rules**

**1. Input Parameters:**

* **app**: A string representing the application name.
* **unit\_nbr**: An integer representing the unit number.
* **job\_set\_nbr**: An integer representing the job set number.
* **dq\_rule\_id**: A string representing the Data Quality (DQ) rule ID.
* **cred\_config**: A configuration object for database credentials.
* **logger**: An instance of a logger for logging information.
* **job\_run\_nbr**: An integer representing the job run number.
* **env**: A string representing the environment.
* **db\_connection\_mobius**: A database connection object.

**2. Description:**

This function is responsible for executing Data Quality (DQ) rules. It retrieves DQ rule details from the database, including SQL queries, thresholds, and other parameters. Then, it performs data comparisons based on the provided criteria, generates HTML reports, and logs the execution details.

The function performs the following steps:

* Initializes data frames for rule details, dates, left-hand side (LHS) data, right-hand side (RHS) data, and more.
* Logs the initiation of rule execution.
* Retrieves DQ rule details from the database based on the provided **app** and **dq\_rule\_id**.
* Retrieves the email display limit from the database or uses a default value if not found.
* Extracts various parameters and configurations from the retrieved DQ rule details.
* Depending on the operator (SRC\_CHECK\_1 or SRC\_CHECK\_2), it executes SQL queries and retrieves data.
* Compares the data based on the operator, threshold, and other criteria.
* Generates HTML reports for failed data points.
* Inserts DQ exception data into the database.
* Handles exceptions and logs any errors that occur during the process.
* Returns the total count of failed data points, an HTML summary of the DQ rule, and an error type if any errors occurred.

**3. Output:**

* **outcome**: An integer representing the total count of failed data points.
* **html\_email\_dq\_final**: An HTML report summarizing the DQ rule execution results.
* **error\_type**: A string indicating the type of error that occurred during execution, if any.

**4. Summary of Imports:**

* **pandas** (**pd**): Used for data manipulation and storage.
* **numpy** (**np**): Used for numerical operations.
* **datetime**: Used for timestamp generation.
* Other custom functions for database connection, query execution, and data comparison.

This document provides a detailed description of the **execute\_rules** function, including its input parameters, logic, and output. It also summarizes the relevant imports for reference.

**Function: insert\_dq\_exception\_data**

**1. Input Parameters:**

* **job\_run\_nbr**: An integer representing the job run number.
* **app**: A string representing the application name.
* **dq\_rule\_id**: A string representing the Data Quality (DQ) rule ID.
* **df\_results**: A DataFrame containing the results of failed data points.
* **cred\_config**: A configuration object for database credentials.
* **env**: A string representing the environment.
* **logger**: An instance of a logger for logging information.
* **db\_connection\_mobius**: A database connection object.

**2. Description:**

This function is responsible for inserting Data Quality (DQ) exception data into the database. It takes the results of failed data points, including their details, and inserts them into a designated DQ exception table in the database.

The function performs the following steps:

* Retrieves the exception table name from the database based on the provided **app**.
* Constructs an SQL query to insert the DQ exception data into the designated table.
* Opens a database connection using the provided credentials and environment.
* Executes the SQL query with the relevant values and commits the transaction.
* Handles exceptions and logs any errors that occur during the insertion process.

**3. Output:**

* None. This function performs database insertion without returning any specific output.

**4. Summary of Imports:**

* **pandas**: Used for data manipulation.
* Other custom functions for database connection and query execution.

This document provides a detailed description of the **insert\_dq\_exception\_data** function, including its input parameters, logic, and purpose. It also summarizes the relevant imports for reference.

Function: generate\_html\_component\_dq\_table

1. Input Parameters:

df\_results: A DataFrame containing the results of failed data points.

df\_results\_final: A DataFrame containing the final results.

verify\_cols\_list: A list of strings representing columns to verify.

key\_cols\_list: A list of strings representing key columns.

operator: A string representing the operator used for comparison.

email\_display\_limit: An integer specifying the maximum number of records to display in the email.

logger: An instance of a logger for logging information.

2. Description:

This function is responsible for generating an HTML component that displays the details of Data Quality (DQ) exceptions in a table format. It takes the results of failed data points, the final results, and other relevant information as input and produces HTML content that can be included in an email notification.

The function performs the following steps:

Initializes an HTML string template.

Iterates through the verify\_cols\_list and generates a table for each column with failed data points.

Limits the number of displayed records based on the email\_display\_limit to avoid email content overload.

Sorts and formats the data for display.

Combines the HTML tables and adds optional information about email filtering.

Returns the final HTML content for inclusion in the email notification.

3. Output:

html\_final: A string containing HTML content representing the DQ exception tables.

4. Summary of Imports:

pandas: Used for data manipulation.

This document provides a detailed description of the generate\_html\_component\_dq\_table function, including its input parameters, logic, and purpose. It also summarizes the relevant imports for reference.

**Function: send\_email**

**1. Input Parameters:**

* **logger**: An instance of the logger for logging information.
* **env**: A string specifying the environment (e.g., "dev," "prod").
* **html\_body**: A string containing the HTML content of the email body.
* **app**: A string representing the application name.
* **cred\_config**: A configuration object containing credentials.
* **job\_run\_nbr**: An identifier for the current job run.
* **config**: A configuration object containing various settings.
* **db\_connection\_mobius**: A database connection object.
* **run\_type**: A string representing the type of the job run.
* **module**: A string representing the module(s) associated with the job run.

**2. Description:**

This function is responsible for sending an email notification with HTML content to a list of subscribers. It leverages external APIs to fetch subscriber email addresses and send the email.

The function performs the following steps:

* Retrieves necessary configurations and settings from the **config** object.
* Constructs API URLs and headers for interacting with external services.
* Calls an external subscription API to fetch a list of subscriber email addresses based on application, purpose, and tracking.
* Constructs an email payload with subject, body, recipients, and other parameters.
* Sends the email using an external email engine API.
* Logs information about the email sending process, including success or failure.

**3. Output:**

* None

**4. Summary of Imports:**

* **requests**: Used for making HTTP requests to external APIs.
* **json**: Used for working with JSON data.

This document provides a detailed description of the **send\_email** function, including its input parameters, logic, and purpose. It also summarizes the relevant imports for reference.

**Function: format\_datelist**

1. Input Parameters:

* **df\_datelist**: A string containing a comma-separated list of dates.

2. Description:

This function takes a comma-separated list of dates (**df\_datelist**) and formats them by adding single quotes around each date and joining them back as a formatted string. It is commonly used for constructing SQL queries with date values.

3. Output:

* **formatted\_datelist**: A string containing formatted dates.

4. Summary of Imports:

* No external imports used in this function.

**Function: execute\_query**

1. Input Parameters:

* **logger**: An instance of the logger for logging information.
* **db\_connection**: A database connection object.
* **query**: A SQL query to be executed.
* **function\_name**: A string specifying the name of the calling function.

2. Description:

This function executes a SQL query (**query**) using the provided database connection (**db\_connection**) and returns the result as a DataFrame. It also logs information using the provided logger.

3. Output:

* **df\_result**: A DataFrame containing the result of the SQL query.

4. Summary of Imports:

* **pd**: Used for working with DataFrames.

**Function: get\_data\_for\_comparison**

1. Input Parameters:

* **query**: A SQL query to fetch data.
* **schema\_name**: A string specifying the schema name.
* **cred\_config**: A configuration object containing credentials.
* **env**: A string specifying the environment.
* **logger**: An instance of the logger for logging information.

2. Description:

This function connects to a database using the credentials provided in **cred\_config**, executes the SQL query (**query**) to retrieve data, and returns the result as a DataFrame. It logs information using the provided logger.

3. Output:

* **df\_rule\_details**: A DataFrame containing the extracted data.

4. Summary of Imports:

* **connect\_db**: An internal function for establishing a database connection.
* **pd**: Used for working with DataFrames.

**Function: get\_list\_of\_dq\_checks\_at\_dataset\_level**

1. Input Parameters:

* **module**: A string representing the module(s).
* **run\_type**: A string representing the type of the job run.
* **cred\_config**: A configuration object containing credentials.
* **env**: A string specifying the environment.
* **db\_connection\_mobius**: A database connection object.
* **logger**: An instance of the logger for logging information.
* **app**: A string representing the application name.

2. Description:

This function retrieves a list of data quality (DQ) checks at the dataset level based on the provided module, run type, and application. It connects to the database, executes a SQL query, and returns the result as a DataFrame.

3. Output:

* **df\_checks\_at\_dataset\_level**: A DataFrame containing the list of DQ checks at the dataset level.

4. Summary of Imports:

* No external imports used in this function.

**Function: get\_list\_of\_dq\_checks**

1. Input Parameters:

* **module**: A string representing the module(s).
* **run\_type**: A string representing the type of the job run.
* **cred\_config**: A configuration object containing credentials.
* **env**: A string specifying the environment.
* **db\_connection\_mobius**: A database connection object.
* **logger**: An instance of the logger for logging information.
* **app**: A string representing the application name.

2. Description:

This function retrieves a list of data quality (DQ) checks based on the provided module, run type, and application. It connects to the database, executes a SQL query, and returns the result as a DataFrame.

3. Output:

* **df\_checks**: A DataFrame containing the list of DQ checks.

4. Summary of Imports:

* No external imports used in this function.

**Function: get\_list\_of\_dq\_rules**

1. Input Parameters:

* **df\_of\_checks**: A DataFrame containing DQ checks.
* **app**: A string representing the application name.
* **cred\_config**: A configuration object containing credentials.
* **env**: A string specifying the environment.
* **db\_connection\_mobius**: A database connection object.
* **logger**: An instance of the logger for logging information.

2. Description:

This function retrieves a list of data quality (DQ) rules based on the provided DQ checks, application, and other parameters. It connects to the database, executes a SQL query, and returns the result as a DataFrame.

3. Output:

* **df\_rules**: A DataFrame containing the list of DQ rules.

4. Summary of Imports:

* No external imports used in this function.

**Function: connect\_db**

1. Input Parameters:

* **cred\_config**: A configuration object containing credentials.
* **app**: A string representing the application name.
* **env**: A string specifying the environment.

2. Description:

This function establishes a database connection using the credentials provided in **cred\_config**, along with the application and environment details.

3. Output:

* **connection**: A database connection object.

4. Summary of Imports:

* **cx\_Oracle**: Used for connecting to Oracle databases.
* **pyodbc**: Used for connecting to Microsoft SQL Server databases.

**Function: load\_config**

1. Input Parameters:

* **logger**: An instance of the logger for logging information.
* **filename**: Absolute path of the config file.

2. Description:

This function loads a configuration file specified by **filename** and returns its content. It is commonly used to load configuration settings from a file.

3. Output:

* Configuration data as a dictionary.

4. Summary of Imports:

* **yaml**: Used for reading YAML configuration files.

**Function: compare\_singleton**

1. Input Parameters:

* **operator**: A string specifying the comparison operator.
* **lhs**: Left-hand side DataFrame for comparison.
* **key\_cols**: List of key columns for joining DataFrames.
* **key\_value**: Key value for naming columns in the result.
* **threshold**: Threshold value for comparison.
* **lower\_limit**: Lower limit for comparison.
* **upper\_limit**: Upper limit for comparison.
* **email\_filter\_threshold**: Threshold for filtering email content.
* **verify\_cols\_list**: List of columns to verify.

2. Description:

This function performs comparison operations on a single DataFrame (**lhs**) based on the provided operator and criteria. It returns the result as a Boolean value, along with DataFrames containing failed rows and an HTML-formatted version of the failed rows.

3. Output:

* **result**: A Boolean value indicating the comparison result.
* **df\_final\_result**: A DataFrame containing failed rows.
* **df\_final\_result\_html**: An HTML-formatted DataFrame of failed rows.

4. Summary of Imports:

* **np**: Used for numerical operations.

**Function: compare\_df**

1. Input Parameters:

* **operator**: A string specifying the comparison operator.
* **lhs**: Left-hand side DataFrame for comparison.
* **rhs**: Right-hand side DataFrame for comparison.
* **key\_cols**: List of key columns for joining DataFrames.
* **key\_value**: Key value for naming columns in the result.
* **threshold**: Threshold value for comparison.
* **lower\_limit**: Lower limit for comparison.
* **upper\_limit**: Upper limit for comparison.
* **email\_filter\_threshold**: Threshold for filtering email content.
* **verify\_cols\_list**: List of columns to verify.

2. Description:

This function performs comparison operations between two DataFrames (**lhs** and **rhs**) based on the provided operator and criteria. It returns the result as a Boolean value, along with DataFrames containing failed rows and an HTML-formatted version of the failed rows.

3. Output:

* **result**: A Boolean value indicating the comparison result.
* **df\_final\_result**: A DataFrame containing failed rows.
* **df\_final\_result\_html**: An HTML-formatted DataFrame of failed rows.

4. Summary of Imports:

* **pd**: Used for working with DataFrames.
* **np**: Used for numerical operations.

This document provides detailed specifications for each function, including input parameters, logic, output, and import summaries for reference.

**Function: represent\_float**

Input Parameters:

* **s**: A string representing a value to check if it can be converted to a float.

Description:

This function checks if the input string **s** can be converted to a float. It returns **True** if successful, **False** otherwise.

Output:

* **result**: A Boolean indicating whether the input string can be converted to a float.

Summary of Imports:

* No external imports are used in this function.

**Function: get\_data\_date**

Input Parameters:

* **cred\_config**: A configuration object containing credentials.
* **dq\_date\_id**: A string representing the DQ date ID.
* **unit\_nbr**: A string representing the unit number.
* **job\_set\_nbr**: A string representing the job set number.
* **date\_filter**: A string representing the date filter.
* **schema\_name**: A string representing the schema name.
* **app**: A string representing the application name.
* **dq\_rule\_id**: A string representing the DQ rule ID.
* **logger**: An instance of the logger for logging information.
* **job\_run\_nbr**: A string representing the job run number.
* **env**: A string specifying the environment.
* **db\_connection\_mobius**: A database connection object.

Description:

This function is triggered to get a list of dates based on various parameters. It first determines the target application based on the schema name and logs relevant information. It then retrieves a date query configuration from a database, formats it with parameters, establishes a database connection, and executes the query. Finally, it logs the results and any errors encountered during the process.

Output:

* **df\_date**: A DataFrame containing the list of dates extracted.

Summary of Imports:

* **pd**: Used for working with DataFrames.
* **datetime**: Used to obtain the current date and time.

**Function: update\_dq\_result\_in\_datastatus**

Input Parameters:

* **logger**: An instance of the logger for logging information.
* **job\_run\_nbr**: A string representing the job run number.
* **config**: A configuration object.
* **app**: A string representing the application name.
* **dq\_check\_id**: A string representing the DQ check ID.
* **runType**: A string representing the type of job run.
* **cred\_config**: A configuration object containing credentials.
* **env**: A string specifying the environment.
* **db\_connection\_mobius**: A database connection object.
* **runTypeOrg**: A string representing the original run type.
* **module**: A string representing the module.
* **dataset**: A string representing the dataset.
* **module\_list**: A list of modules.

Description:

This function updates the data quality (DQ) check result in the Datastatus module. It constructs API URLs, extracts information from the database, and triggers API calls to update the DQ check status. It handles both scheduled and ad-hoc job runs and logs relevant information.

Output:

* None

Summary of Imports:

* **datetime**: Used to obtain the current date and time.
* **json**: Used for working with JSON data.
* **requests**: Used for making HTTP requests.

**Function: decrypt\_credentials**

Input Parameters:

* **config**: A configuration object.
* **app**: A string representing the application name.

Description:

This function decrypts encrypted credentials using AES encryption. It takes the encrypted value, key, and initialization vector (IV) from the configuration and decrypts the password, returning it as a plain text string.

Output:

* **pwd**: A string representing the decrypted password.

Summary of Imports:

* **base64**: Used for decoding base64-encoded values.
* **AES**: Used for AES encryption/decryption.

**Function: logging**

Input Parameters:

* **env**: A string specifying the environment.
* **cred\_config**: A configuration object containing credentials.
* **job\_run\_nbr**: A string representing the job run number.
* **dq\_rule\_id**: A string representing the DQ rule ID.
* **application**: A string representing the application name.
* **log\_dt**: A datetime object representing the log date and time.
* **message\_cat\_id**: A string representing the message category (INFO, ERROR, WARNING).
* **log\_source**: A string representing the log source.
* **log\_msg**: A string representing the log message.
* **db\_connection\_mobius**: A database connection object.
* **logger**: An instance of the logger for logging information.

Description:

This function logs information to both the logger and the Mobius database. It logs messages with different categories (INFO, ERROR, WARNING) and inserts log entries into the **TBLMOB\_LOG\_DQ\_JOB\_INFO** table in the Mobius database.

Output:

* None

Summary of Imports:

* No external imports are used in this function.