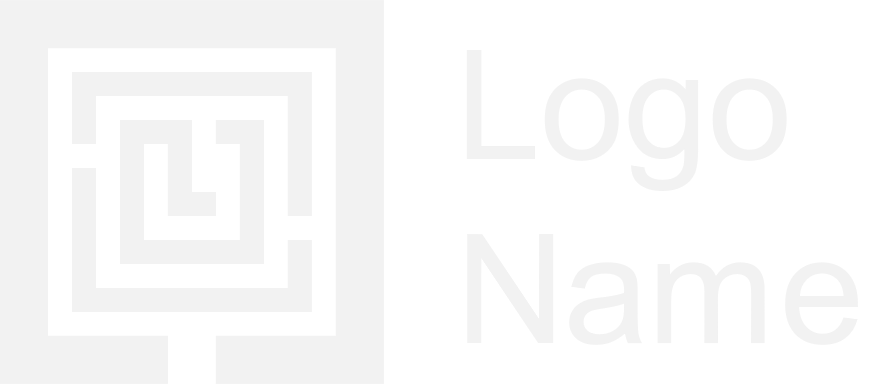


|  |
| --- |
| Data Quality 2021 |
|  |
| April 2022 |



Contents

[1. Defined Terms 3](#_Toc87022443)

[2. Data Quality Introduction 4](#_Toc87022444)

[2.1. Overall Design 4](#_Toc87022445)

[3. End-to-End Solution 4](#_Toc87022446)

[3.1. Components 4](#_Toc87022447)

[3.1.1. Data Profiling 4](#_Toc87022448)

[3.1.2. Data Quality Assessment 4](#_Toc87022449)

[3.1.3. DQ UDF 4](#_Toc87022450)

[4. System Architecture 9](#_Toc87022451)

[4.1. Technology matrix 9](#_Toc87022452)

[4.2. Connection Details 9](#_Toc87022453)

[4.2.1. Connection details between streaming and DQ UDF 9](#_Toc87022454)

# Defined Terms

Capitalized terms or abbreviations used in this design document have the meaning as set out below

|  |  |
| --- | --- |
| Term/ Acronym | Definition |
| DQ | Data Quality |
| IDF | Intelligent Data Foundation – a PySpark based processing and data quality framework from Accenture |
| ADB | Azure Databricks |
| UDF | User defined Function |
| UI | User Interface |
| ADLS | Azure Data Lake Storage |
| ADF | Azure Data Factory |

**Table 1.1 – Defined Terms**

# Data Quality Introduction

|  |
| --- |
| DQ is how-well the data is suited to serve its intended purpose. IDF has data quality for streaming data, stateful data & complete data profiling, which will solve the problem of determining quality of data for various data types in Azure cloud. Overall Design The overall design is based on a set of architectural and deployment considerations as illustrated below:   1. DQ solution is independent of other modules of IDF. 2. DQ consists sub-parts as Data profiling, DQ Assessment and Data Remediation. 3. Data profiling is profiling done on complete file and on all the columns of the file. There are few open-source libraries used in this module such as AWS pydeequ and pandas profiling. Data Profiling also include determining of CDEs (Critical Data Elements). 4. DQ Assessment is determining DQ dimension and statistics based on business rules defined by users. 5. Data Remediation is based on DQ Assessment rule. For each DQ assessment rule user can define their own remediation rule. 6. DQ is available as UDF as well which accepts streaming dataframe (continuous/foreachbatch) and returns the same dataframe with one extra column which denotes failed row in dataframe. Here in this case rule and statistics is generated in delta tables (not in SQL tables).  End-to-End SolutionComponentsData Profiling This is an UI based data profiling for the files present in Azure Storage Account. Data Quality AssessmentDQ UDF DQ UDF is based on IDF DQ Assessment. DQ function will take input as streaming data frame and table Name/ File Name of the data frame. It will deliver ingested streaming dataframe with error code (column FAILED\_RULE\_ID) to the streaming process because of the various DQ rules and criteria defined in DQ Master table. Limitation of DQ UDF is that it works on delta tables only i.e., dqMaster, dqStatistics, prcs\_exec and exception data all are in delta format. DQ UDF is egg file which needs to be installed on cluster.   * *Input Parameters:* PySpark streaming data frame, table Name & producer name as matched with entries in dqMaster. * *Intermediate puts:*   + *dqStatistics (delta):* ForEachBatch of stream data will be ingested into dqStatistics delta table, which contains failed, passed, and total row processed in each run.   + *PRCS\_Exec (delta):* ForEachBatch of stream data, audit control table is also maintained to keep track of DQ process, which process start time, end time and few more details as mentioned in PRCS\_Exec table.   + *DQ\_Exception (delta):* all the failed records, composite keys and values with error code is stored for reference. * *Output (Returns):* PySpark streaming data frame (only records which passed all DQ rules for the topic and table defined in dqMaster. * Pre-requisite:   Below are the pre-requisites for DQ UDF:   1. Create dqMaster and prcs\_exec table before executing the DQ UDF. Use below notebook to create delta tables:   [https://adb-5003007810941205.5.azuredatabricks.net/?o=5003007810941205#notebook/3978516485066720/command/3978516485066723](about:blank)     1. Define DQ rules for data in dqMaster.      1. Create constant\_stream.py file with below details like dqMaster path according to your environment details and upload it at path */FileStore/tables/constant\_stream.py* in your ADB.      * UDF code and packaging:   *Steps to create egg file:*   1. Create folder consisting of all codes to be packaged. Please refer screenshot:      1. Create setup.py file parallel to main folder. Content of setup.py is below:      1. Important point to remember all folder and subfolder should consists of \_\_init\_\_.py to be packaged as egg file.      1. All python code should be bundled in Classes and should be utilized by creating object. See below example:      1. Once above steps are done open cmd or python console. Navigate to the folder where your setup.py is:   *cd C:\Users\aparna\IDF\DQ*   1. Run below command to create egg file   *For cmd: python setup.py bdist\_egg*  *For python console: ./setup.py bdist\_egg*   1. In case any update in code, update version in setup.py file. After egg file creation below folders will be created:      1. To install egg file on ADB cluster, upload egg file from dist folder to ADB dbfs desired folder.      1. Manual step to install to ADB cluster   Compute 🡪 Click on Cluster you want to install 🡪 Libraries 🡪 Install New 🡪 Library Type 🡪 Python Egg 🡪 Install |
| 1. Auto installation code   *dbutils.fs.put("/<path>/<library-name>.sh","""*  *#!/bin/bash*  *easy\_install-3.7 /dbfs/<path-to-egg-file>/<egg-file>.egg""")*  *""")*   * Rules Types   Below are the rule types to be configured till date in DQ as UDF:   |  |  | | --- | --- | | Rules Type | Description | | Null-check | Validates if defined column in dqMaster is having null values. If null value found for any record, error code is appended to the record. Record is marked as failed record. | | Duplicate-check | Validates if a record has duplicate record in same dataset using primary or composite key. If found, all duplicate records are marked as failed record by appending error code. | | Latency-check | Validates if time difference between two date columns provided by user meets the SLA. | |
| System ArchitectureTechnology matrixConnection DetailsConnection details between streaming and DQ UDF *Import DQ UDF as library and pass below object:* |
| *from dq import DataQualityStreamProcess*  *env\_obj = {"prcs\_runid": "RunId03","gdp\_layer": "Base", "table\_name":"sales "}*  *validated\_data = DataQualityStreamProcess.data\_quality\_stream\_process\_function(spark,dbutils, input\_df, env\_obj)* |
|  |