**FUNCTIONAL REQUIREMENT DOCUMENT / Functional Specification Document**

Project Name: Sales & Inventory Data Warehouse: Sept,2024

**Revision Sheet**

|  |  |  |
| --- | --- | --- |
| **Release No.** | **Date** | **Revision Description** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Purpose

The purpose of this document is to define the functional requirements for building a **Sales & Inventory Data Warehouse** on **Azure Cloud**, aimed at integrating sales and inventory data from multiple sources (flat files, databases) into a centralized system. This warehouse will enable better decision-making through the availability of historical data, consolidated reporting, and improved data analytics for sales, inventory, and returns data.

The primary objectives include:

* Streamlining the sales and inventory data pipeline.
* Enabling **near real-time reporting** through **Azure Synapse Analytics**.
* Applying **business rules** to maintain **data quality** and consistency across various datasets.
* **Supporting business intelligence (BI)** requirements for better decision-making using **Power BI** for visualization.

# Scope

The scope of this project includes:

* **Data extraction** from source systems such as flat files and databases (e.g., SQL Server) into **Azure Data Lake Storage (ADLS)**.
* **Data transformation and validation** using **Azure Databricks** and **PySpark notebooks** to ensure data quality and business rule enforcement.
* **Data loading** into the **SalesEDW** (Enterprise Data Warehouse) layer hosted on **Azure SQL DB** for **gold-tier** data.
* **Versioning** of data using **SCD Type 2** for tracking historical changes.
* Creating **dimension** and **fact tables** in **Azure SQL DB** for accurate analysis and reporting.
* Facilitating **BI dashboards** in **Power BI** for actionable insights, focusing on sales trends, inventory levels, order returns, etc.

# Assumptions and Constraints

Assumptions:

* Time/Date is already created and available in the Gold Layer of the data lake.
* The project will use existing Azure Services like ADF, ADLS Gen2, Databricks, Azure Synapse, and Azure SQL DB.
* Incremental data processing is required for select tables, where only the data that has changed since the last run is processed.
* **Data Availability**: Source data (sales, inventory) is accessible and in the required format.
* **Business Rules**: Clear business rules for data validation are defined.
* **Data Quality**: Source systems provide accurate data with minimal errors.
* **Tool Availability**: Required Azure services (ADF, Databricks, ADLS, etc.) are pre-configured.
* **Incremental Loads**: Source systems support incremental data processing.
* **Access Control**: Only authorized users access the data via Azure AD.
* **SCD2 Support**: Historical tracking (SCD2) is supported for sales and inventory data.
* **Stakeholder Engagement**: Business stakeholders are available for input

**Constraints:**

1. **Data Volume**: High volumes of data may strain Azure resources.
2. **Real-time Reporting**: Near real-time, not true real-time reporting due to processing delays.
3. **Data Source Delays**: Pipeline depends on timely data from external sources.
4. **Service Limits**: Azure storage, compute, and Databricks capacity constraints

# DESIGN CONSIDERATIONS

## Naming Standards

**Linked Service:**

LS\_SRC/SINK\_ProjDesc\_ConStr

**Dataset:**

DS\_Format(Table or File etc)

**Pipeline:**

PL\_EXT/INT\_SRC\_TGT\_MART/ZONE\_Dela/Full\_JobFreq

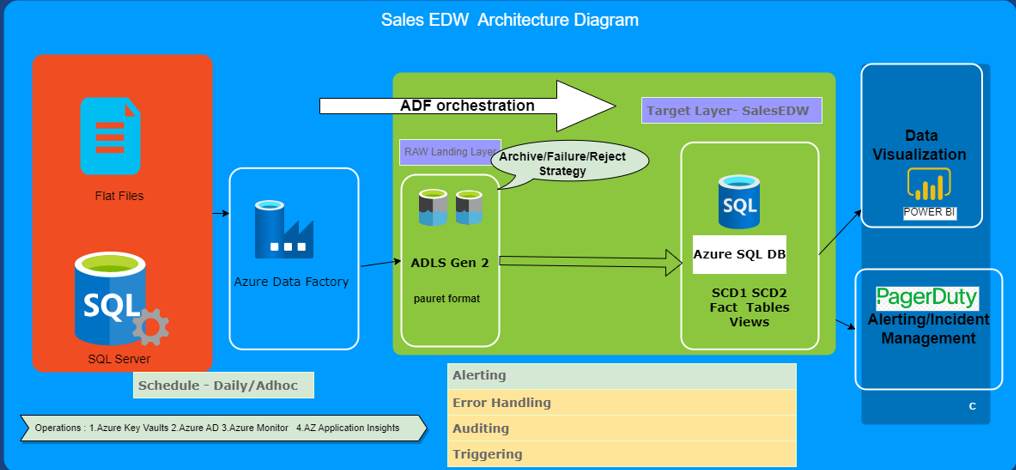
More Detail as Below:

|  |  |  |
| --- | --- | --- |
| Abbreviation | Linked Service | Dataset |
| [Azure Blob Storage](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-blob-storage) | ABLB\_ | LS\_ABLB\_ | DS\_ABLB\_ |
| [Azure Cosmos DB SQL API](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-cosmos-db) | ACSA\_ | LS\_ACSA\_ | DS\_ACSA\_ |
| [Azure Cosmos DB MongDB API](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-cosmos-db-mongodb-api) | ACMA\_ | LS\_ACMA\_ | DS\_ACMA\_ |
| [Azure Data Explorer](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-data-explorer) | ADEX\_ | LS\_ADEX\_ | DS\_ADEX\_ |
| [Azure Data Lake Storage Gen1](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-data-lake-store) | ADLS\_ | LS\_ADLS\_ | DS\_ADLS\_ |
| [Azure Data Lake Storage Gen2](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-data-lake-storage) | ADLS\_ | LS\_ADLS\_ | DS\_ADLS\_ |
| [Azure Database for MariaDB](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-database-for-mariadb) | AMDB\_ | LS\_AMDB\_ | DS\_AMDB\_ |
| [Azure Database for MySQL](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-database-for-mysql) | AMYS\_ | LS\_AMYS\_ | DS\_AMYS\_ |
| [Azure Database for PostgreSQL](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-database-for-postgresql) | APOS\_ | LS\_APOS\_ | DS\_APOS\_ |
| [Azure File Storage](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-file-storage) | AFIL\_ | LS\_AFIL\_ | DS\_AFIL\_ |
| [Azure Search](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-search) | ASER\_ | LS\_ASER\_ | DS\_ASER\_ |
| [Azure SQL Database](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-sql-database) | ASQL\_ | LS\_ASQL\_ | DS\_ASQL\_ |
| [Azure SQL Database Managed Instance](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-sql-database-managed-insance) | ASQM\_ | LS\_ASQM\_ | DS\_ASQM\_ |
| [Azure SQL Data Warehouse](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-sql-data-warehouse) | ASDW\_ | LS\_ASDW\_ | DS\_ASDW\_ |
| [Azure Table Storage](https://docs.microsoft.com/en-us/azure/data-factory/connector-azure-table-storage) | ATBL\_ | LS\_ATBL\_ | DS\_ATBL\_ |
| [SQL Server](https://docs.microsoft.com/en-us/azure/data-factory/data-factory-sqlserver-connector) | MSQL\_ | LS\_SQL\_ | DS\_SQL\_ |
| [Oracle](https://docs.microsoft.com/en-us/azure/data-factory/data-factory-onprem-oracle-connector) | ORAC\_ | LS\_ORAC\_ | DS\_ORAC\_ |
| [MySQL](https://docs.microsoft.com/en-us/azure/data-factory/data-factory-onprem-mysql-connector) | MYSQ\_ | LS\_MYSQ\_ | DS\_MYSQ\_ |
| [DB2](https://docs.microsoft.com/en-us/azure/data-factory/data-factory-onprem-db2-connector) | DB2\_ | LS\_DB2\_ | DS\_DB2\_ |
| [FTP](https://docs.microsoft.com/en-us/azure/data-factory/data-factory-ftp-connector) | FTP\_ | LS\_FTP\_ | DS\_FTP\_ |
| [SFTP](https://docs.microsoft.com/en-us/azure/data-factory/v1/data-factory-sftp-connector) | SFTP\_ | LS\_SFTP\_ | DS\_SFTP\_ |
| [File System](https://docs.microsoft.com/en-us/azure/data-factory/data-factory-onprem-file-system-connector) | FILE\_ | LS\_FILE\_ | DS\_FILE\_ |

**Activities**:

|  |  |  |  |
| --- | --- | --- | --- |
| Move & Transform | Copy Data | ACT\_MT\_Copy\_ | ACT\_MT\_Copy\_FunctionalPurpose |
|  | Data FloW | ACT\_MT\_DF | ACT\_MT\_DF\_FunctionalPurpose |
| General | Append Variable | ACT\_GEN\_APPVAR | ACT\_GEN\_APPVAR\_FunctionalPurpose |
|  | Delete | ACT\_GEN\_DEL | ACT\_GEN\_DEL\_FunctionalPurpose |
|  | Execute Pipeline | ACT\_GEN\_EPIPE | ACT\_GEN\_EPIPE\_FunctionalPurpose |
|  | Execute SSIS | ACT\_GEN\_ESSIS | ACT\_GEN\_ESSIS\_FunctionalPurpose |
|  | Get Metadata | ACT\_GEN\_META | ACT\_GEN\_META\_FunctionalPurpose |
|  | Lookup | ACT\_GEN\_LKP | ACT\_GEN\_LKP\_FunctionalPurpose |
|  | Stored Procedure | ACT\_GEN\_SPRC | ACT\_GEN\_SPRC\_FunctionalPurpose |
|  | Set Variable | ACT\_GEN\_SETVAR | ACT\_GEN\_SETVAR\_FunctionalPurpose |
|  | Validation | ACT\_GEN\_VALID | ACT\_GEN\_VALID\_FunctionalPurpose |
|  | Web | ACT\_GEN\_WEB | ACT\_GEN\_WEB\_FunctionalPurpose |
|  | WebHook | ACT\_GEN\_WHOOK | ACT\_GEN\_WHOOK\_FunctionalPurpose |
|  | Wait | ACT\_GEN\_WAIT | ACT\_GEN\_WAIT\_FunctionalPurpose |
| Iteration & conditionals | Filter | ACT\_ITRC\_FIL | ACT\_ITRC\_FIL\_FunctionalPurpose |
|  | For Each | ACT\_ITRC\_FOR | ACT\_ITRC\_FOR\_FunctionalPurpose |
|  | If Condition | ACT\_ITRC\_IF | ACT\_ITRC\_IF\_FunctionalPurpose |
|  | Switch | ACT\_ITRC\_SWITCH | ACT\_ITRC\_SWITCH\_FunctionalPurpose |
|  | Until | ACT\_ITRC\_UNTILL | ACT\_ITRC\_UNTILL\_FunctionalPurpose |
| Data Bricks | Notebook | ACT\_ADB\_NOTE | ACT\_ADB\_NOTE\_FunctionalPurpose |
|  | Jar | ACT\_ADB\_JAR | ACT\_ADB\_JAR\_FunctionalPurpose |
|  | Python | ACT\_ADB\_PY | ACT\_ADB\_PY\_FunctionalPurpose |
| Batch Service | Custom | ACT\_BAT\_CUST | ACT\_BAT\_CUST\_FunctionalPurpose |
| Data Lake Analytics | U-SQL | ACT\_ADLA\_CUST | ACT\_ADLA\_CUST\_FunctionalPurpose |
| Azure Function | Azure Function | ACT\_AF\_AF | ACT\_AF\_AF\_FunctionalPurpose |
| HDInsight | Hive | ACT\_HDI\_HIVE | ACT\_HDI\_HIVE\_FunctionalPurpose |
|  | Map Reduce | ACT\_HDI\_MAPR | ACT\_HDI\_MAPR\_FunctionalPurpose |
|  | Pig | ACT\_HDI\_PIG | ACT\_HDI\_PIG\_FunctionalPurpose |
|  | Spark | ACT\_HDI\_SPARK | ACT\_HDI\_SPARK\_FunctionalPurpose |
|  | Streaming | ACT\_HDI\_STREAM | ACT\_HDI\_STREAM\_FunctionalPurpose |
| Machine Learning | ML Batch execution | ACT\_ML\_BATCH | ACT\_ML\_BATCH\_FunctionalPurpose |
|  | ML Update Resource | ACT\_ML\_UPD | ACT\_ML\_UPD\_FunctionalPurpose |
|  | ML Execute Pipeline | ACT\_ML\_EPIPE | ACT\_ML\_EPIPE\_FunctionalPurpose |

**5.0 HIGH Level Design (project architecture DIAGRAM)**

****

**Technology components**

|  |  |
| --- | --- |
| Technology | Function |
| Azure Cloud | Application Development |
| Azure Data Factory | Data Extraction/Load |
| ADLS- Gen2 | For Storing Raw Data |
| Azure SQL DB  (Serverless) | For Storing Gold EDW Data |
| Databricks | Data Extraction and Transformation . |
| Power BI | Reporting and Data Visualization |

**Sources: (Source Folder Files)**

* **Flat Files (e.g., .dat, .txt, .csv)**: Available in the source folder, these files will be uploaded into Azure Blob Storage for further processing.
* **SQL Server Database**: A sales data dump is provided in the source folder. This data will be imported into a SQL Server database following the video guidelines provided in the source folder.

**Environments Setup:** **Development**

Create three resource groups as follows:

1. <Sales\_DataEnginnering\_Project\_YourName\_DEV >
2. <Sales\_DataEnginnering\_Project\_YourName\_QA >
3. <Sales\_DataEnginnering\_Project\_YourName\_PROD >

**Create the following setup under the 'Dev resource group Only**

**Services to Create:**

1. **Azure Data Factory**: <ADF\_SalesProject\_YourName>
   1. Orchestrate ETL processes for data pipeline.
2. **Azure Databricks**: <ADB\_SalesProject\_YourName>
   1. Run data transformation and analytics using Apache Spark.
3. **Key Vault**: <KV\_SalesProject\_YourName>
   1. Securely store secrets (e.g., database credentials).
4. **Data Lake Storage**: <adlssalesprojectyourname>
   1. Store raw and processed data (e.g., .csv, .txt).
5. **Blob Storage**: <blobstorageprojectyourname>
   1. Store incoming source flat files.
6. **Azure SQL Database**: <ASQL\_SalesProject\_YourName>
   1. Store structured sales data.

**Note**: Set up only in the **DEV** environment for now.

### 1. ADLS (Azure Data Lake Storage) Layer:

**1. ADLS Layer Requirements**

1. **Data Loading Convention**
   * **ADLS Container**: **RAW**
   * **Naming Convention**:
     + **Directory Name**: Same as the table name in the source.
     + **File Name**: TableName\_<todaystimestamp>.csv or TableName\_<todaystimestamp>.txt
     + **Example**:
       - **Target Folder Path**: RAW/SourceName/Target/ORDER\_HEADER/OrderHeader\_20231016143015.csv
   * **File Management**:
     + **Archive Folder Path**: Move successfully loaded files to the archive:
       - RAW/SourceName/Archive/ORDER\_HEADER/OrderHeader\_20231016143015.csv
     + **Failure Folder Path**: Move failed pipeline files for reprocessing:
       - RAW/SourceName/Failure/ORDER\_HEADER/OrderHeader\_20231016143015.csv
     + **Reject Folder Path**: Store records that fail validation:
       - RAW/SourceName/Reject/ORDER\_HEADER/Reject\_OrderHeader\_20231016143015.csv
2. **Tables and Files to Fetch from Source**:

**Tables to Fetch from Source SQL Server Database**:

* ORDER\_HEADER
* PRODUCT
* ORDER\_DETAILS
* RETURN\_ITEM
* INVENTORY\_LEVELS
* BRANCH

**Files to Load from Azure Blob Storage**:

* Sales/ORDER\_METHOD.csv
* Sales/RETURN\_REASON.csv
* Sales/COUNTRY.txt
* Sales/WAREHOUSE.txt
* Sales/RETAILER.csv
* Sales/PRODUCT\_NAME\_LOOKUP.csv

1. **Initial Load**:

**Table:** Perform a full data extraction from the source system into the ADLS RAW layer.

**Flat File:** The flat file will be received from the source in delta mode, meaning we will receive incremental or latest data. Therefore, there is no need to perform incremental loading for the flat file source.

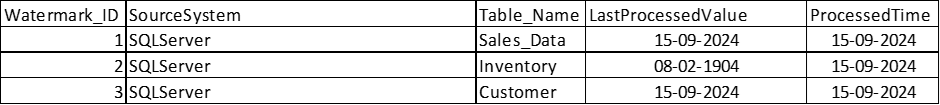
1. **Subsequent Runs (Second Load) for SQL Server Tables:**

For all subsequent runs, or from the second load onwards, **incremental data should be fetched for the following tables based on the update date columns in the source** driving or master tables. The last fetched value should be saved/updated in the watermark table after each successful data pull at the end of the pipeline (once the data is loaded into Azure SQL DB), so that it can be used for the next pull.

a. **Order\_Header (**based on the **update date columns** in the source)  
b. **Order\_Details (**based on the **update date columns** in the source)

**Azumation:** Currently, the update date column is not present in the source. You need to alter the source to include this column. However, in real-time scenarios, such columns are already present.

**Watermark Table:**



**Reference:** For more detailed information, please refer to the use case document for the watermark table.

[Use Case of Watermark Table.docx](https://docs.google.com/document/d/1gMXIdLkx28n4Qbzc-fP6MwZmEmtqaM0N/edit?usp=drive_link&ouid=114803117063647243426&rtpof=true&sd=true)

1. **Watermark Table Update Rules:**

Ensure the watermark table is updated in the following cases:

1. **Successful Data Pull:**
   * The watermark table should be updated with the last fetched value after each successful data extraction from the source to the target database.
2. **Incremental Loads:**
   * For incremental loads, the watermark should reflect the latest update date from the source tables.
3. **Error Handling:**
   * If a pipeline fails, do not update the watermark table until the pipeline has successfully completed its rerun.
4. **Initial Load:**
   * During the initial load, the watermark should be set to the maximum update date of the records fetched.
5. **Regular Audits:**
   * Conduct regular audits to ensure the watermark table accurately reflects the last fetched values, especially after multiple pipeline runs.

1. **Recovery Strategy on Pipeline Failure:**

In case any pipeline fails, please follow these steps:

* **Identify the Failed Pipeline:** Determine which pipeline(s) have failed.
* **Rerun the Pipeline:** Rerun the pipeline for the specific table(s) based on the failure or for all tables as necessary.
* **Monitor Execution:** Ensure that the rerun completes successfully and monitor for any further issues.

1. All the table details should be maintained in the config file. Config file should have at least below mentioned items for each table.
   * Source Schema
   * Source Table
   * Incremental\_Column\_Name
   * Target\_Container
   * Target\_FileName
   * Target\_FileFormat
2. File format should be csv/txt/parquet in ADLS Raw layer (but should be driven from config file)
3. Refer to “Source to ADLS Raw LLD” for low level column mapping.

[LLD\_Source\_To\_ADLS(RAW) Layer \_LOAD.xlsx](https://docs.google.com/spreadsheets/d/1-m_8Iyfrhuo1lWUZsmrRoJHzLaH9HbGE/edit?usp=drive_link&ouid=114803117063647243426&rtpof=true&sd=true)

**Pending Task: Dynamic Query for SQL Server Source:**

**Task Description:**

At the end of the project, we will explain how to create a dynamic SQL query for relational sources such as SQL Server or Oracle.

If time permits, we will be explaining this requirement cases.

### 2. Azure SQL DB Layer (Ingestion and Integration):

Once data is loaded into the ADLS Raw layer, it needs to be moved into the Azure SQL layers sequentially, starting with the Ingestion layer followed by the Integration layer.

**Ingestion Layer:**

* The ingestion layer serves as the first point of entry for data from various source systems into the Azure environment. Its main role is to efficiently extract and load data.
* **Truncate and Load:** Clears target tables before loading new data to ensure freshness.

**Integration Layer:**

* The Integration Layer is designed to transform, cleanse, and enrich the ingested data. It prepares the data for analytical purposes and ensures it aligns with business requirements.
* The Integration Layer contains data transformation processes, Slowly Changing Dimension (SCD) tables, and fact tables.

**1. Ingestion Layer Requirements:**

**Datasets to be Created in Ingestion Layer:**

1. Branch
2. Order\_Header
3. Order\_Detail
4. Order\_Method
5. Return\_Item
6. Inventory\_Levels
7. PRODUCT
8. RETURN\_REASON
9. COUNTRY
10. WAREHOUSE
11. RETAILER
12. PRODUCT\_NAME

**Validation Rules:**

* Reject records with duplicate primary keys (PK) or null PKs.

**Handling Rejections:**

* Store rejected records in Parquet format in the raw layer's reject folder.
* Load valid records into the integration layer's Azure SQL tables.
* All rejected data will be stored in Parquet format in the raw layer's reject folder.

**Implementation:**

* Refer to "ADLS to Integration layer\_Sales\_LLD" for rules.
* Use Databricks notebooks for validation and transformation.

**LLD Document for ADLS RAW to Ingestion Layer:** (Refer to this document to create and move data accordingly)

[LLD\_RAW To\_Ingestion\_Layer\_LOAD.xlsx](https://docs.google.com/spreadsheets/d/1qu9OnT8Q_ItGPQNe_kfcflPPxIpt6Ws2/edit?usp=drive_link&ouid=114803117063647243426&rtpof=true&sd=true)

1. **Integration Layer Requirements:**

**Key Points:**

* All columns highlighted in yellow are surrogate keys to be generated while loading data from Raw to Integration.
* All columns highlighted in blue are business keys for SCD.
  + - 1. ***Dimensions***

List of dimension tables to be created under the **Integration** schema/layers:

* TBL\_DIM\_ORDER (**SCD2**)
* **Load Frequency:** Daily at (9 **AM UST**)
* TBL\_DIM\_PRODUCT (**SCD1**)
* **Load Frequency:** Daily at (**9 AM UST**)
* TBL\_DIM\_ORDER\_METHOD\_LKP (**SCD1**)
  + **Load Frequency:** Ad-hoc/On Demand

* TBL\_DIM\_RETURN\_REASON\_LKP(**SCD1**)
  + - **Load Frequency:** Ad-hoc/On Demand
* TBL\_DIM\_COUNTRY\_LKP (**SCD1**)
  + **Load Frequency:** Ad-hoc/On Demand
* TBL\_DIM\_WAREHOUSE\_LKP(**SCD1**)
  + **Load Frequency:** Ad-hoc/On Demand
* TBL\_DIM\_RETAILER\_LKP (**SCD1**)
  + **Load Frequency:** Ad-hoc/On Demand
* TBL\_DIM\_PRODUCT\_NAME\_LKP**(SCD1)** 
  + **Load Frequency:** Ad-hoc/On Demand
* TBL\_DIM\_DATE\_TIME (**Role-Playing Dimension**): copy as datetime table from source
* <**Load Frequency:** Ad-hoc/On Demand >

**Refer the LLD\_Ingestion\_TO Integration Layer LOAD for Data Load :**

[LLD\_Ingestion\_TO Integration Layer LOAD.xlsx](https://docs.google.com/spreadsheets/d/1Z1SakSj3G25lUAMMHCzm05Yu9-Hk1b_Y/edit?usp=drive_link&ouid=114803117063647243426&rtpof=true&sd=true)

**Additional Points:**

* All tables must exist in Integration before developing pipelines to load data.
* Implement SCD Type 2 for all dimension datasets based on incremental data received in the integration layer:
  + **New Records**: Insert
  + **Existing Records**: Expire the existing record and create a new one (refer to Versioning Rule for SCD2 for more details).

**Transformation:**

* Leverage Databricks for transformations from ADLS to Integration:
  + Use Spark for half of the tables and Spark SQL for the other half.

### Versioning Rule for SCD2

**Fields:**

BEGIN\_EFFECTIVE\_DATE

END\_EFFECTIVE\_DATE

VERSION\_NUM

CHANGE\_INFO

* **BEGIN\_EFFECTIVE\_DATE**: The date when the record became effective or was first inserted into the system.
* **END\_EFFECTIVE\_DATE**: This is set to **31-DEC-2099** to indicate that this is the current version of the record (no end date yet).
* **VERSION\_NUM**: The version number of the record, starting with 1 for the first instance.
* **CHANGE\_INFO**: Information about what changed, or why the record was inserted (could be NULL on the first insert).

Day1 when a record P1 comes for the first time with P1 A B

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SK\_KEY** | **C1** | **C2** | **C3** | **BEGIN\_EFFECTIVE\_DATE** | **END\_EFFECTIVE\_DATE** | **Active\_flag** |  |
| 1 | P1 | A | B | *LOAD\_TIME1* | 31-DEC-2020 | Y |  |

Day2 when the same record P1 comes with P1 A C

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SK\_KEY** | **C1** | **C2** | **C3** | **BEGIN\_EFFECTIVE\_DATE** | **END\_EFFECTIVE\_DATE** | **Active\_flag** |  |
| 1 | P1 | A | B | *LOAD\_TIME1* | *LOAD\_TME2* | N |  |
| 2 | P1 | A | C | *LOAD\_TME2* | 31-DEC-2099 | Y |  |

**LLD Document for ADLS to Sales\_Integration Layer:** (Refer to this document to create and move data accordingly)

### *2. Fact Table*

List of Fact tables to be created:

* TBL\_FACT\_SALES (Transactional Fact Table)
* TBL\_FACT\_INVENTORY (Transactional Fact Table)

***3. Dimension Data Model (Layout):*** *Refer to the attached dimension data model layout.*

[Dimension data model (Layout).xlsx](https://docs.google.com/spreadsheets/d/10VJecMHeDmhsEFujkrB8UbgQ7pxM3Jg2/edit?usp=drive_link&ouid=114803117063647243426&rtpof=true&sd=true)

***4. Fact Table Data Model (Layout):***

To load the fact tables, you need to join all ingestion layer tables with the integration layer dimension tables. The fact tables will only perform insert operations; no update operations are allowed.

|  |
| --- |
| TBL\_FACT\_SALES |
| ORDER\_DATE\_KEY |
| RETAILER\_KEY |
| PRODUCT\_KEY |
| COUNTRY\_KEY |
| ORDER\_METHOD\_KEY |
| SALES\_ORDER\_KEY |
| SHIP\_DATE\_KEY |
| RETURN\_REASON\_KEY |
| CLOSE\_DAY\_KEY |
| RETURNED\_QUANTITY |
| QUANTITY |
| UNIT\_COST |
| UNIT\_PRICE |
| UNIT\_SALE\_PRICE |
| SALE\_TOTAL |
| SOURCE\_ID |
| DATA\_DATE |
| UPDATE\_DATE |
|  |
|  |

|  |
| --- |
| TBL\_FACT\_INVENTORY |
| INVENTORY\_YEAR\_KEY |
| PRODUCT\_KEY |
| WAREHOUSE\_KEY |
| OPENING\_INVENTORY |
| QUANTITY\_SALE\_SHIP |
| INV\_ADDITIONS |
| UNIT\_COST |
| CLOSING\_INVENTORY (Availability) |
| **SOURCE\_ID** |
| **DATA\_DATE** |
| **UPDATE\_DATE** |

**Compliance Control Mechanisms:** Apply security compliance control mechanisms for implementation of auditing on sensitive data

· Apply data masking on unit cost, unit sales price (cost/price related columns) and any other sensitive data.

· Apply row level security on WAREHOUSE branch code.

· Turn on Microsoft defender for SQL**.**

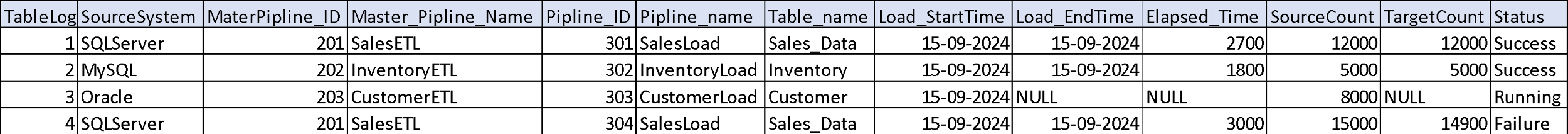
1. ***Audit Log Table:***

**Requirements**

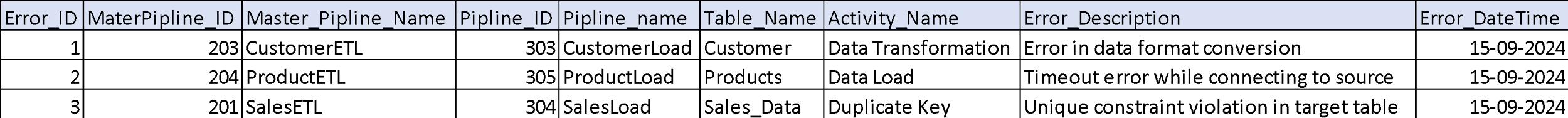
1. **Track ETL Pipeline Performance**: Monitor the performance of each ETL pipeline, including start and end times, record counts, and overall status.
2. **Capture and Investigate Errors**: Log detailed information about errors encountered during pipeline execution to facilitate troubleshooting and resolution.
3. Track the source and target count for record reconciliation

**Table Name:** TableLog table and Error\_Logging table

1. **TableLog Table**



1. **Error\_Logging Table:**



**Reference Documentation:**

* For more detailed information regarding operational aspects and logging processes, please refer to the use case documents for:
* TableLog Table
* Error\_Log Table

[UseCase TableLog and Error\_Logging.docx](https://docs.google.com/document/d/12LD3JK4rIUPuk-PB9gnc0-m1xVU3cK1M/edit?usp=drive_link&ouid=114803117063647243426&rtpof=true&sd=true)

**Orchestration For SQL Source and Flat File Source:**

**1. SQL Source Pipelines:**

1. **Create dedicated end-to-end pipelines for each SQL table to:**
   * Load data from the source system to the ADLS Raw Layer.
   * Load data from ADLS Raw Layer to Azure SQL Ingestion Layer.
   * Load data from Azure SQL Ingestion Layer to Azure SQL Integration Layer.

**2. Master Pipeline for SQL server Source**:

* Create an ADF master pipeline to:
  + Execute all components involved in Raw to Azure SQL data processing.
  + Integrate ADF pipelines, Databricks notebooks, and any other processing logic.

**3. Scheduling:**

* Schedule the master pipeline to run daily at 9 AM IST.

**4. Execution Time Monitoring:**

* Set the maximum execution duration for the entire process to 40 minutes.
* Configure an alert mechanism if the pipeline execution exceeds this time limit.

**5. Alert Mechanism:**

* Set up an alert mechanism to notify stakeholders of:
  + Any failures in pipeline execution.
  + Trigger or schedule failures.

**6. Immediate Failure Handling:**

* In case of any immediate failure, implement a retry mechanism to:
  + Rerun the pipeline up to 3 times, with a wait of 1 minute between each retry attempt.
* **Flat File Source Pipelines:**

Load data through a sequence of steps from a flat file source to the ADLS Raw Layer and then to Azure SQL layers.

* Steps:
  + **Load Data:**
    - Load data from the flat file source system to the ADLS Raw Layer.
    - Load data from the ADLS Raw Layer to the Azure SQL Ingestion Layer.
    - Load data from the Azure SQL Ingestion Layer to the Azure SQL Integration Layer.
* **Master Pipeline flat file Source:**

Create an ADF master pipeline to:

* + Execute all components involved in Raw to Azure SQL data processing.
  + Integrate ADF pipelines, Databricks notebooks, and any other processing logic.
* **Trigger Setup:** Set up a trigger for the pipeline to handle the following scenario:
  + **Wait Mechanism:**
    - Wait for 10 minutes for the flat file to be available at the source location.
    - If the file is not available after 10 minutes, fail the task.
  + **Notification Mechanism:**
    - Set up a notification system to alert stakeholders if the file has not been received, including details of the missing file's name.
* **Scheduling:**
  + - * Schedule the master pipeline to run on an **ad-hoc/on-demand basis**.
      * This allows for manual execution based on client requirements and the availability of data in the lookup table.