**Bootcamp Project 1: Data Pipeline for Customer Account Analysis**

**Project Overview**

The objective of this project is to design and implement a scalable and secure data pipeline that ingests, transforms, enriches, and loads customer-related financial data from a backend Virtual Machine into Azure Data Lake and Azure SQL Database. The pipeline ensures data quality and consistency to support analytical reporting through Power BI.

| **Tool / Service** | **Purpose** |
| --- | --- |
| **Azure Virtual Machine** | Hosting raw data files (CSV) from backend team |
| **Azure Data Factory** | Orchestration of the pipeline and transformation using Data Flows |
| **Self-hosted IR** | Connects ADF to on-premise (VM) file system |
| **Azure Data Lake Gen2** | Data storage with Bronze (Raw), Silver (Cleaned), and Gold (Enriched) layers |
| **Azure Key Vault** | Securely store secrets and credentials |
| **Azure SQL Database** | Final data destination for reporting |
| **Power BI** | Data visualization and reporting |
| **Draw.io** | Used for designing architecture diagrams |

**Architecture Diagram**

A diagram of a computer process

Description automatically generated

**Step 1: Data Ingestion (VM → ADLS Gen2 Bronze Layer)**

**Goal:**

To securely ingest raw CSV files from a Virtual Machine into Azure Data Lake Storage Gen2 under the raw (bronze) container.

**Input data :** **A screenshot of a computer

Description automatically generated**

**Activities Involved:**

1. **Self-hosted Integration Runtime (IR):**  
   Installed on the VM to bridge the gap between the VM and ADF.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA computer screen with white text

Description automatically generatedcreated a linked service using key vaultA screenshot of a computer

Description automatically generated

1. **ADF Pipeline: pl\_project1**
   * **Activity 1: Get Metadata**  
     Retrieves the list of files (childItems) from the source folder on the VM.

A screenshot of a computer

Description automatically generated

**Activity 2: ForEach Loop**

* + - Iterates over each file using dynamic expression.
    - Executes the Copy Data activity for each file

A screenshot of a computer

Description automatically generated

* + **Copy Data Activity:**  
    Transfers the files from VM → ADLS Gen2 Bronze Container in .csv format using DelimitedTextSourceand DelimitedTextSink.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedPipeline ran successfullyA screenshot of a computer

Description automatically generated **Destination folder** A screenshot of a computer

Description automatically generated This ingestion step sets the foundation for further data transformation in the silver and gold layers.

**Step 2: Data Cleaning & Transformation (Bronze → Silver Layer)**

**Goal:**

To clean the raw ingested data by removing duplicates, handling missing or irrelevant data, converting data types, and preparing it for enrichment and loading.

**Activity Involved:**

**ADF Pipeline: pipeline1**

This pipeline contains a single **Execute Data Flow** activity that calls a data flow named **dataflow1**.

**Data Flow:**

In this data flow (dataflow1), we performed data cleaning and transformation for all five input datasets:

* **Accounts**
* **Customers**
* **Loan Payments**
* **Loans**
* **Transactions**

For each file, we applied the following standard transformations:

* Aggregated records to remove duplicates based on primary keys.
* Filtered out duplicate entries using a dup\_count == 1 condition.
* Selected relevant columns for downstream processing.
* Used AlterRow transformations with upsert logic for SCD handling.
* Wrote cleaned and deduplicated data to the Silver Layer in **Delta format**.

**A screenshot of a computer program

Description automatically generated**

**Detailed Breakdown – Account Dataset Transformation**

This transformation pipeline focuses on **cleaning**, **deduplicating**, and **preparing the account data** for storage in the **Silver layer** of our Lakehouse architecture.

### Source A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **2. Aggregate**

* **Purpose**: Remove duplicate records for the same account\_id by grouping.
* **Transformation**:  
  Grouped the data by account\_id and extracted key values using aggregation functions:

| **Output Column** | **Expression Used** | **Description** |
| --- | --- | --- |
| customer\_id | first(customer\_id) | Picks the first customer\_id within each group |
| account\_type | first(account\_type) | Takes the first type if there are multiple values |
| balance | first(balance) | Keeps the first balance seen |
| dup\_count | count(account\_id) | Helps identify duplicates by counting rows in each group |

### **A screenshot of a computer Description automatically generated** **A screenshot of a computer Description automatically generated** A screenshot of a computer Description automatically generated **3. Filter**

* **Logic**: dup\_count == 1
* **Purpose**:
  + This step filters out groups where there is more than one row per account\_id, ensuring **only truly unique accounts** pass through.
  + This keeps our data clean and prevents accidental overwrites.

### **A screenshot of a computer Description automatically generated** **A screenshot of a computer Description automatically generated 4. Select**

* **Purpose**:
  + To streamline the data by passing through only the necessary fields needed downstream.
* **Why this step?**  
  Removes helper columns like dup\_count used for intermediate processing and ensures a clean output schema.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **5. AlterRow**

* **Logic Used**: insertIf(1==1)
* **Purpose**:
  + This instructs the data flow to **insert all incoming rows** into the target sink.
  + Unlike an upsert, this does **not check for existing records**. Every row is treated as new.

### A screenshot of a computer Description automatically generated **A screenshot of a computer Description automatically generated6. Sink**

* **Output Format**: Delta Lake

## **Destination**: silver\_folder/account in ADLS (Azure Data Lake Storage) A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Step 3: Data Flows using SCD Techniques**

We applied **Slowly Changing Dimension (SCD)** logic using Azure Data Factory Mapping Data Flows to handle changes in the data:

* **SCD Type 1** (Overwrite): For account, transaction, and loan\_payment datasets.
* **SCD Type 2** (History Tracking): For customer and loans datasets.

A screenshot of a computer

Description automatically generated Let’s dive deeper into one example of each.

**Account SCD Type1** **A diagram of a diagram

Description automatically generated**

### **Source –** source1

* **Description**: Reads transformed account data from the Silver layer stored in Azure Data Lake Storage (Delta format).
* **Columns**: account\_id, customer\_id, account\_type, balance
* **Purpose**: Acts as the incoming/up-to-date dataset for processing.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Rename –** Rename

* **Description**: Renames all columns from source1 by prefixing them with src\_.
* **Example**: account\_id → src\_account\_id
* **Purpose**: Prevents column name conflicts when performing joins or lookups with source2

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Derived Column –** derivedColumn1

* **Description**: Generates a src\_hashkey using a CRC32 hash function on all business attributes.
* Hash uniquely identifies the state of a record, used to detect changes.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Source –** source2

* **Description**: Reads existing dimension data from the DimAccount\_SCD1 table in the Azure SQL Database.
* **Query**: SELECT account\_id, acct\_hashkey FROM DimAccount\_SCD1
* **Purpose**: Used for comparison to detect changes based on a hash of business keys.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a data report Description automatically generated **Lookup –** lookup1

* **Description**: Performs a lookup from source2 on src\_account\_id == account\_id to bring in the existing hash key (acct\_hashkey).
* **Join Type**: Left Outer Join
* **Purpose**: Compare incoming record’s hash with the existing hash to detect insert or update.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Split –** split1

* **Description**: Splits the data into two streams based on SCD logic.
* **Conditions**:
  + isNull(account\_id) → New record (Insert)
  + src\_account\_id == account\_id && src\_hashkey != acct\_hashkey → Changed record (Update)
* **Purpose**: Identifies whether each incoming record is new or an update.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Derived Column –** derivedColumn2 **(Insert Path)**

* **Description**: Adds audit columns for newly inserted records.
* **Columns Added**:
  + src\_created\_by = "dataflow"
  + src\_created\_date = currentTimestamp()
  + src\_updated\_by = "dataflow"
  + src\_update\_date = currentTimestamp()
* **Purpose**: Maintains audit trail for insertions.

### **Derived Column –** derivedColumn3 **(Update Path)**

* **Description**: Adds audit fields for updated records.
* **Columns Added**:
  + src\_updated\_by = "dataflow-updated"
  + src\_updated\_date = currentTimestamp()
* **Purpose**: Maintains audit trail for updates.

### **Alter Row –** alterRow1

* **Description**: Flags rows for update.
* **Condition**: updateIf(1 == 1) (all records in this stream should be updated)
* **Purpose**: Tells the sink to update the records instead of inserting them.

### **Sink –** sink1 **(Insert Path)**

* **Target**: DimAccount\_SCD1 in Azure SQL DB
* **Operation**: Insert only
* **Mapped Columns**:
  + All business and audit fields (account\_id, customer\_id, account\_type, balance, acct\_hashkey, created\_by, etc.)
* **Purpose**: Inserts new records into the dimension table.

### **Sink –** sink2 **(Update Path)**

* **Target**: DimAccount\_SCD1 in Azure SQL DB
* **Operation**: Update only
* **Keys Used**: account\_id, acct\_hashkey
* **Mapped Columns**:
  + Business fields and audit columns for update (updated\_by, updated\_date)
* **Purpose**: Updates records where changes are detected.

**SCD type 2 :** **A screenshot of a computer

Description automatically generated**

### **Source –** source1

* **Source**: Delta format file from the silver\_folder/loans path in Azure Data Lake Storage.
* **Purpose**: Represents the latest incoming loan data to be compared with historical records.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Rename –** Rename

* **Purpose**: Renames all columns from source1 by adding a src\_ prefix to prevent name collisions during the transformation steps.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Derived Column –** derivedColumn1

* **Creates a Hash Key** (src\_hashkey) using CRC32 on business columns to detect changes.
* **Used For**: Lightweight change detection.
* **Purpose**: Compares this hash with existing record hash to check for any updates.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generatedA screenshot of a computer Description automatically generated **Source –** source2

* **Source**: Azure SQL Database table loans\_scd1 where is\_ACTIVE = 1.
* **Purpose**: Provides only the **latest active records** to be compared for change detection.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Lookup –** lookup1

* **Join Condition**: src\_loan\_id == loan\_id
* **Purpose**: Retrieves the hash key of the current active version of each loan for change comparison.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Split –** split1

Splits data into:

* **Insert Path**: isNull(loan\_id) → New records
* **Update Path**: src\_loan\_id == loan\_id && src\_hashkey != loan\_hashkey → Existing records with changes
* **Purpose**: Directs rows to the appropriate transformation for insert or update as per SCD Type 2.

### A screenshot of a computer Description automatically generated **A screenshot of a computer Description automatically generated Derived Column –** derivedColumn3 **(Update Path)**

* **Updates old record** with:
  + src\_updated\_by = "dataflow-updated"
  + src\_updated\_date = currentTimestamp()
  + SRC\_isActive = 0
* **Purpose**: Marks the current record as **inactive**, allowing historical tracking.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Alter Row –** alterRow1

* **Condition**: updateIf(1==1)
* **Purpose**: Flags the old record for update to deactivate it (is\_ACTIVE = 0).

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Sink –** sink2 **(Update Old Versions)**

* **Operation**: Update
* **Keys**: loan\_id, loan\_hashkey
* **Purpose**: Marks **old versions as inactive** (is\_ACTIVE = 0) to retain history

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Union –** union1

* Combines records from both split1@insert and split1@update.
* Sends them for insertion as new records.
* **Purpose**: Adds the new version of all changed or new records to the destination table.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Derived Column –** derivedColumn2 **(Union Path)**

* Adds audit and metadata fields for new version:
  + src\_created\_by, src\_created\_date
  + src\_updated\_by, src\_update\_date
  + src\_isActive = 1
* **Purpose**: Enriches new version of the record with SCD Type 2 tracking information.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated **Sink –** sink1 **(Insert New Versions)**

* **Operation**: Insert only
* **Table**: loans\_scd1
* **Purpose**: Inserts **new or updated versions** of the loan records as active.

### A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a data report Description automatically generated A screenshot of a computer Description automatically generated using materpipeline ran the flow A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated A screenshot of a computer Description automatically generated