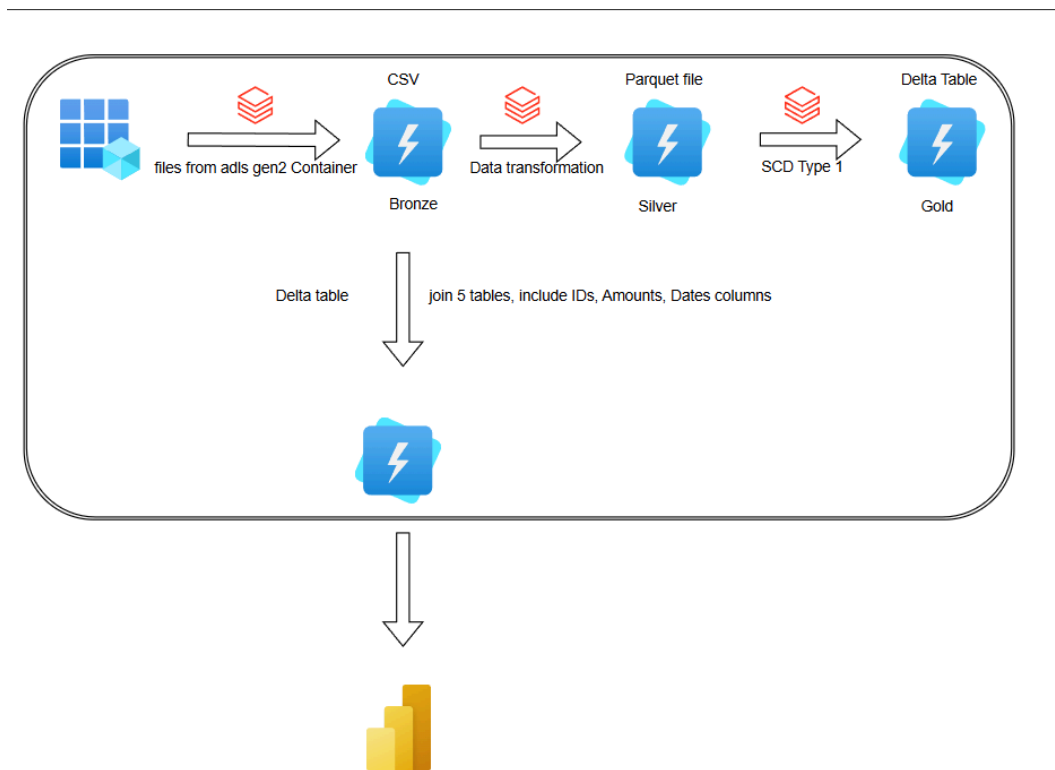


## Bootcamp Project 2 - Transactions and Loan Data for a Customer

### Introduction

This project uploads five banking CSVs to ADLS Gen2, processes them in Azure Databricks with cleaning, transformation, and SCD Type 1 deduplication. Data is stored in Delta format, secured via Key Vault and mount points, then visualized in Power BI and published to Microsoft Fabric Workspace.

### Project Architecture



Architecture of the Project

### Implementation

#### Step 1: Data Ingestion

- Source: Backend team storage account containing files:

- o accounts.csv
  - o customers.csv
  - o loan\_payments.csv
  - o loans.csv
  - o transactions.csv
- Sink: ADLS Gen2 raw (Bronze) container.

Location: [edw](#) / [Project\\_2](#) / Bronze

Search blobs by prefix (case-sensitive)

☐ Show deleted objects

Name	Modified	Access tier
<input type="checkbox"/>  accounts.csv	4/25/2025, 1:27:22 AM	Hot (Inferred)
<input type="checkbox"/>  customers.csv	4/25/2025, 12:36:28 ...	Hot (Inferred)
<input type="checkbox"/>  loan_payments.csv	4/25/2025, 12:36:28 ...	Hot (Inferred)
<input type="checkbox"/>  loans.csv	4/25/2025, 12:36:28 ...	Hot (Inferred)
<input type="checkbox"/>  transactions.csv	4/25/2025, 12:36:28 ...	Hot (Inferred)

## List of the files

### Step 2: Data Cleaning and Transformation

- Environment: Databricks Notebooks
- Mount ADLS Gen2 using scope from Azure Key Vault.

HomePage / Create Secret Scope

## Create Secret Scope

[Cancel](#) [Create](#)

A store for secrets that is identified by a name and backed by a specific store type. [Learn more](#)

Scope Name ?

Manage Principal ?

Creator ▼

### Azure Key Vault ?

DNS Name

Resource ID

Created the Scope and stored the Connecting Details

- create a mount point to Azure Data Lake gen 2 using below code

```
02:03 PM (13s) 3
configs = {"fs.azure.account.auth.type": "OAuth",
"fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
"fs.azure.account.oauth2.client.id": dbutils.secrets.get(scope="akvconnection",key="sc-appid"),
"fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="akvconnection",key="sc-appvalue1"),
"fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/1209a40a-2a3c-4de4-90aa-7ad54f3b2cca/oauth2/token"}
# Optionally, you can add <directory-name> to the source URI of your mount point.
dbutils.fs.mount(
source = "abfss://edw@sc01ad1sgen2.dfs.core.windows.net/",
mount_point = "/mnt/edw",
extra_configs = configs)

True
```

- we use pyspark code to read data from bronze layer, remove duplicates and null values and store the cleaned data into Silver layer in parquet format for all the files in Bronze layer.

▶ ✓ 02:03 PM (14s)

6

```
df_accounts=spark.read.csv("/mnt/edw/bronze/accounts.csv",header=True, inferSchema=True)
df_accounts = df_accounts.na.drop()
df_accounts=df_accounts.dropDuplicates()
df_accounts.write.mode("overwrite").format("parquet") .save("/mnt/edw/silver/Accounts")
display(df_accounts)
```

▶ (6) Spark Jobs

▶ ✓ 02:04 PM (4s)

7

```
df_customers=spark.read.csv("/mnt/edw/bronze/customers.csv",header=True, inferSchema=True)
df_customers = df_customers.na.drop()
df_customers=df_customers.dropDuplicates()
df_customers.write.mode("overwrite").format("parquet") .save("/mnt/edw/silver/Customers")
display(df_customers)
```

▶ (6) Spark Jobs

▶ df\_customers: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, first\_name: string ... 5 more fields]

▶ ✓ 02:04 PM (3s)

8

```
df_loan_payments=spark.read.csv("/mnt/edw/bronze/loan_payments.csv",header=True, inferSchema=True)
df_loan_payments = df_loan_payments.na.drop()
df_loan_payments=df_loan_payments.dropDuplicates()
df_loan_payments.write.mode("overwrite").format("parquet") .save("/mnt/edw/silver/Loan_Payment")
display(df_loan_payments)
```

▶ (6) Spark Jobs

▶ df\_loan\_payments: pyspark.sql.dataframe.DataFrame = [payment\_id: integer, loan\_id: integer ... 2 more fields]

▶ ✓ 02:04 PM (3s)

9

```
df_loans=spark.read.csv("/mnt/edw/bronze/loans.csv",header=True, inferSchema=True)
df_loans = df_loans.na.drop()
df_loans=df_loans.dropDuplicates()
df_loans.write.mode("overwrite").format("parquet") .save("/mnt/edw/silver/Loans")
display(df_loans)
```

▶ (6) Spark Jobs

▶ ✓ 02:04 PM (3s)

10

```
df_transactions=spark.read.csv("/mnt/edw/bronze/transactions.csv",header=True, inferSchema=True)
df_transactions = df_transactions.na.drop()
df_transactions=df_transactions.dropDuplicates()
df_transactions.write.mode("overwrite").format("parquet") .save("/mnt/edw/silver/Transactions")
display(df_transactions)
```

▶ (6) Spark Jobs

- Join all these files into one file and store in Silver later in parquet format.

```
▶ 02:04 PM (2s) 11

df_join1=df_accounts.join(df_customers,on="customer_id",how="left")
df_join2=df_join1.join(df_transactions,on="account_id",how="left")
df_join3=df_join2.join(df_loans,on="customer_id",how="left")
final_df=df_join3.join(df_loan_payments,on="loan_id",how="left")
display(final_df)

▶ (10) Spark Jobs
```

```
▶ 02:15 PM (<1s)

from pyspark.sql.functions import col

final_join=final_df.select(
    col("account_id").cast("int"),
    col("transaction_id").cast("int"),
    col("loan_id").cast("int"),
    col("customer_id").cast("int"),
    col("balance").cast("float"),
    col("transaction_date").cast("timestamp"),
    col("transaction_amount").cast("float"),
    col("loan_amount").cast("float"),
    col("payment_amount").cast("float"),
    col("payment_date").cast("timestamp")
)
```

```
▶ 02:15 PM (3s) 14

final_join.write.format("parquet").option("header",True).mode("overwrite").save("/mnt/edw/silver/final")

▶ (10) Spark Jobs
```

- perform SCD Type 1 on all the files data.

```
▶ ✓ 02:26 PM (10s) 15

%sql

CREATE TABLE IF NOT EXISTS delta.`/mnt/edw/gold/Accounts/` (
  account_id int,
  customer_id int,
  account_type string,
  balance float,
  hash_key bigint,
  created_by string,
  created_date timestamp,
  updated_by string,
  updtaed_date timestamp
)
```

```
▶ ✓ 02:41 PM (<1s) 16

from pyspark.sql.functions import *
dbutils.widgets.text("filedate","")
filedate=dbutils.widgets.get("filedate")

▶ ✓ 02:32 PM (2s) 17

src_path="/mnt/edw/silver/"+filedate
print(src_path)
tgt_path="/mnt/edw/gold/Accounts/"
print(tgt_path)
df_src=spark.read.format("parquet").option("header","true").option("inferSchema","true").load(src_path)
display(df_src)
df_src1=df_src.withColumn("hash_key",crc32(concat(*df_src.columns)))
```

Once SCD Type 1 is done we have to create a schedule job to run pipeline everyday or any specified time

Microsoft Azure databricks

Search data, notebooks, recents, and more... CTRL + P

sc-databricks

New

Workspace

Recents

Catalog

Workflows

Compute

Marketplace

SQL

SQL Editor

Queries

Dashboards

Genie

Alerts

Query History

SQL Warehouses

Data Engineering

Project2 Python Tabs: OFF

File Edit View Run Help Last edit was 4 hours ago

Run all Project2 Schedule Share

filedate Transactions

100 rows | 1.13s runtime

02:39 PM (3s)

```
%sql
CREATE TABLE IF NOT EXISTS
loan_id int,
customer_id int,
loan_amount float,
interest_rate float,
loan_term int,
hash_key bigint,
created_by string,
created_date timestamp,
updated_by string,
updtad_date timestamp
```

New schedule

Job name\* SC\_Project2

Simple Advanced

Schedule

Every Day at 19 : 10

Show cron syntax

Timezone (UTC-04:00) Eastern Time (US and Canada) DST

Compute\* Project2 16 GB · 4 Cores · DBR 15.4 LTS · Spark 3.5.0 · Scala 2.12