Bootcamp Project 2 - Transactions and Loan Data for a Customer

**Objective:**

The project aims to design and implement a robust data pipeline for processing customer account data. This includes copying data from ADLS GEN2 (Bronze layer) and transforming the data in the Silver layer using Data bricks Notebooks and GOLD Storage into the SCDType 1 Delta Table in ADLS GEN2. The pipeline aims to ensure efficient, accurate, and scalable data processing to support downstream analytics and reporting needs.

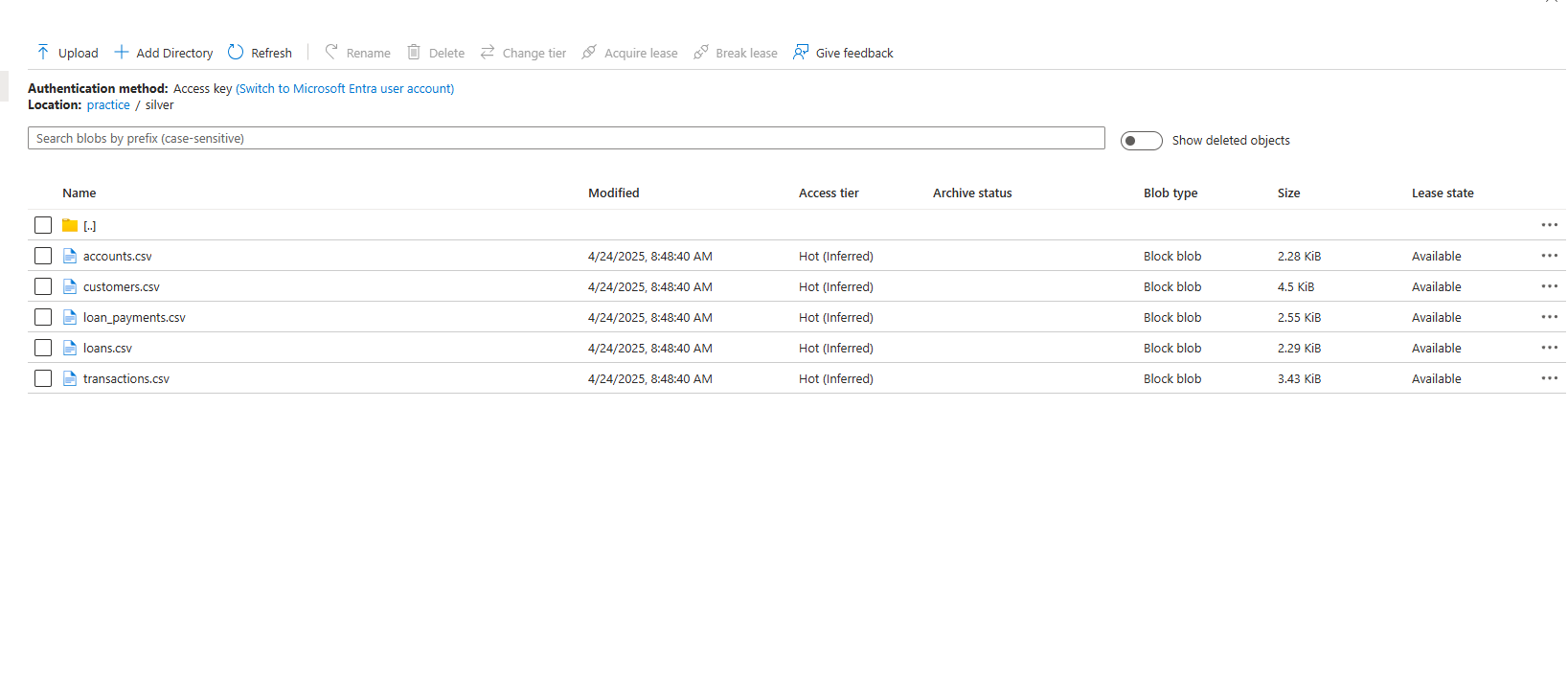
# Architecture:

# 

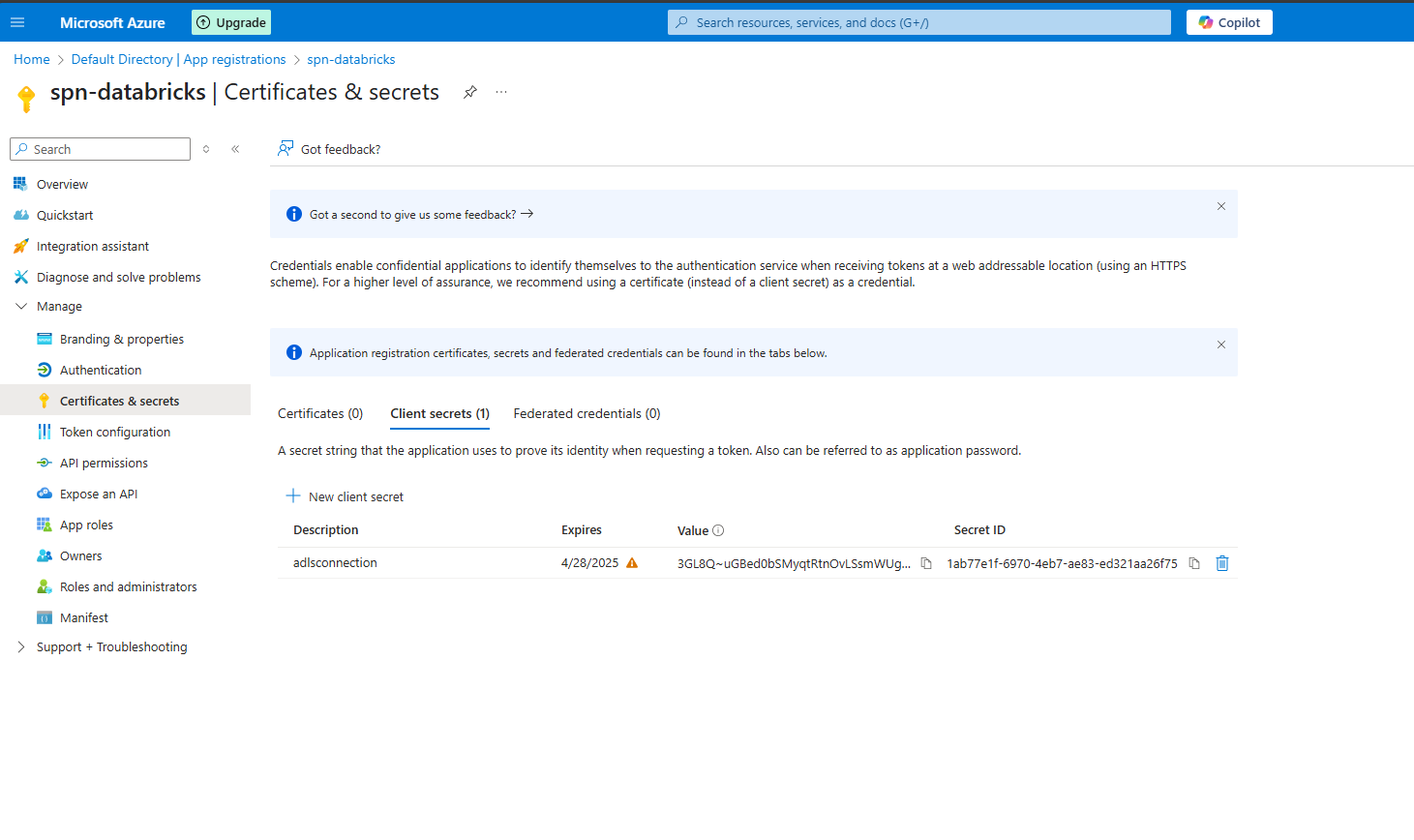
# Project Steps:

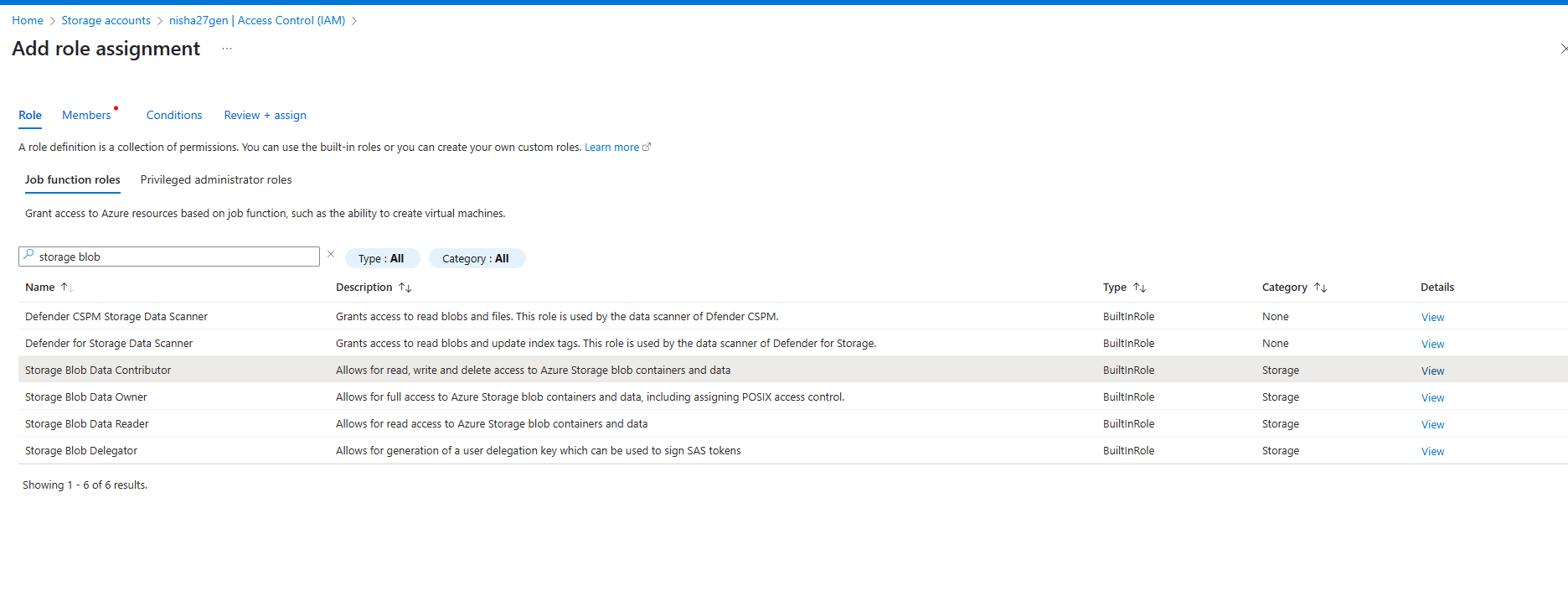
**Step 1: Data Ingestion (Raw(Bronze) Container to ADLS GEN2 (Silver)**

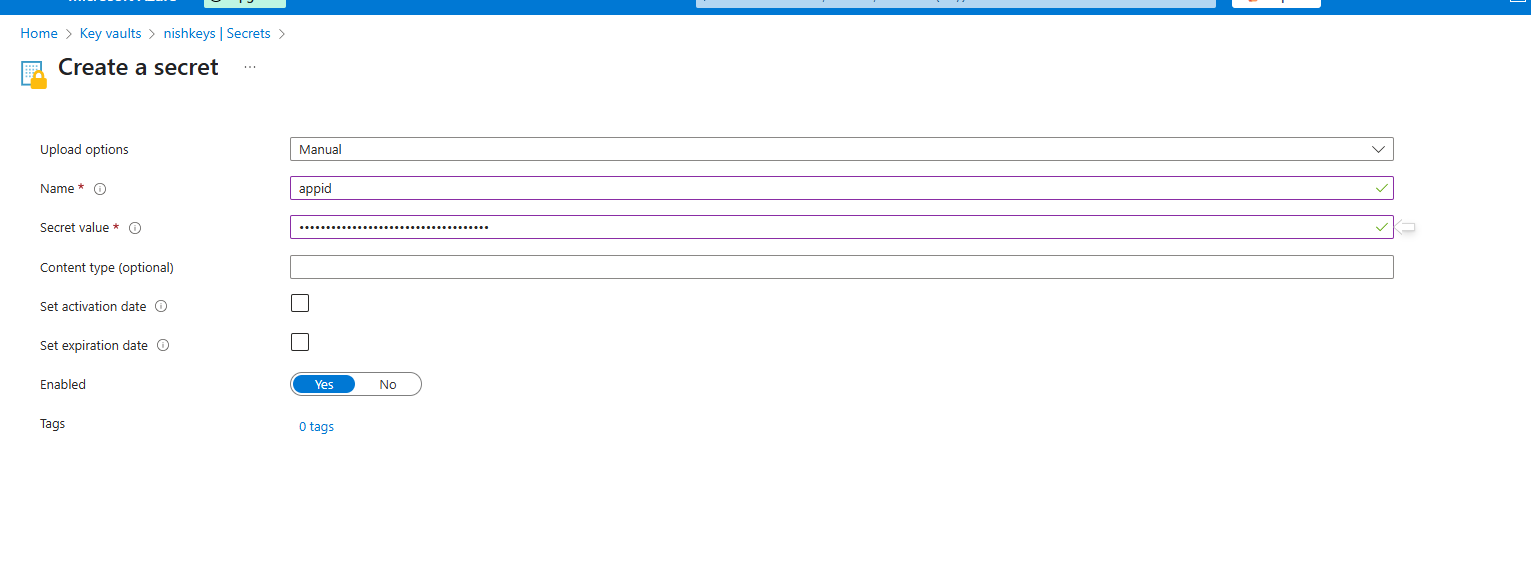
Have used upload function to ingest the data into ADLS container



Connecting SPN to databricks







# Step 2: Use Databricks Notebooks to remove the duplicates

# Mount Azure Data Lake Storage (ADLS Gen2)

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="nishadbconne",key="appid"),

          "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="nishadbconne",key="appsecret"),

          "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/760d704e-d22c-4c4e-be5c-4ff7f6b5b5eb/oauth2/token"}

# Optionally, you can add <directory-name> to the source URI of your mount point.

dbutils.fs.mount(

source = "abfss://practice@nisha27gen.dfs.core.windows.net/",

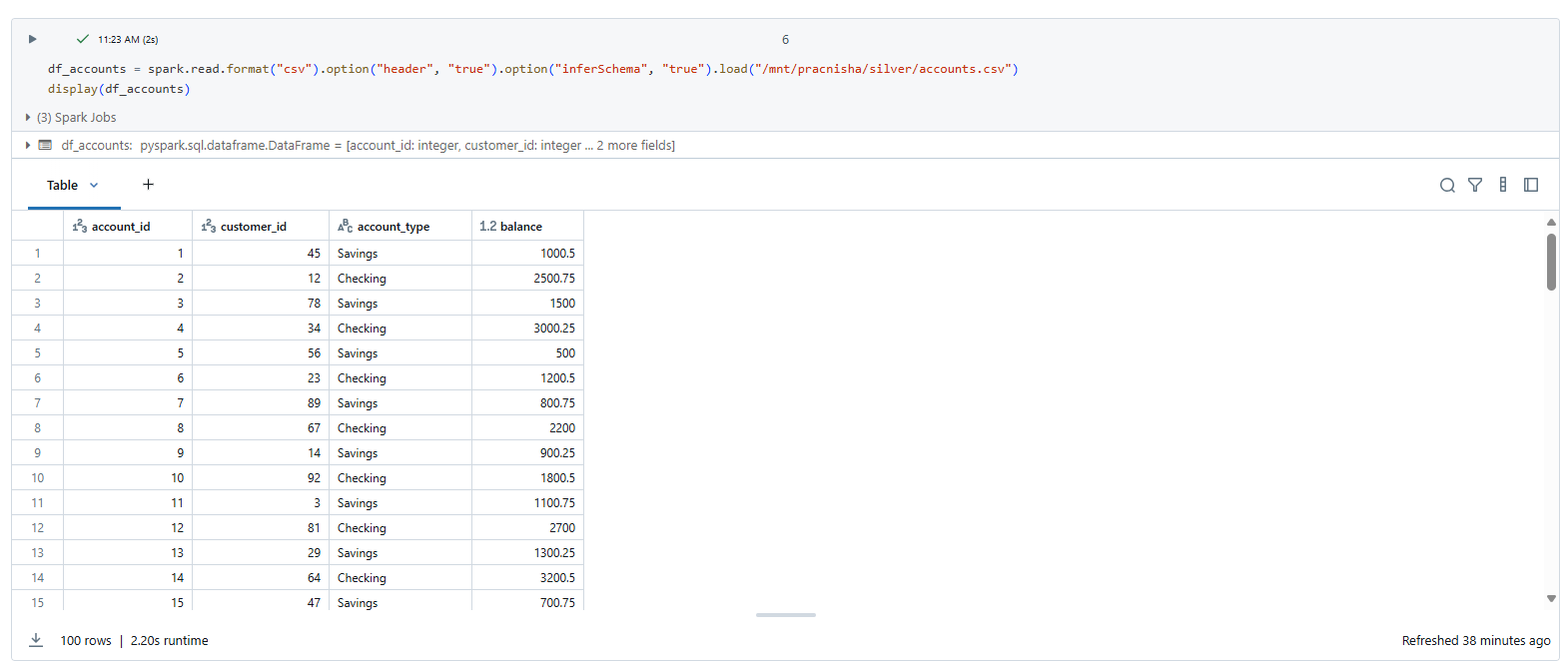
mount\_point = "/mnt/pracnisha",

extra\_configs = configs)

1. To read the file use the below code

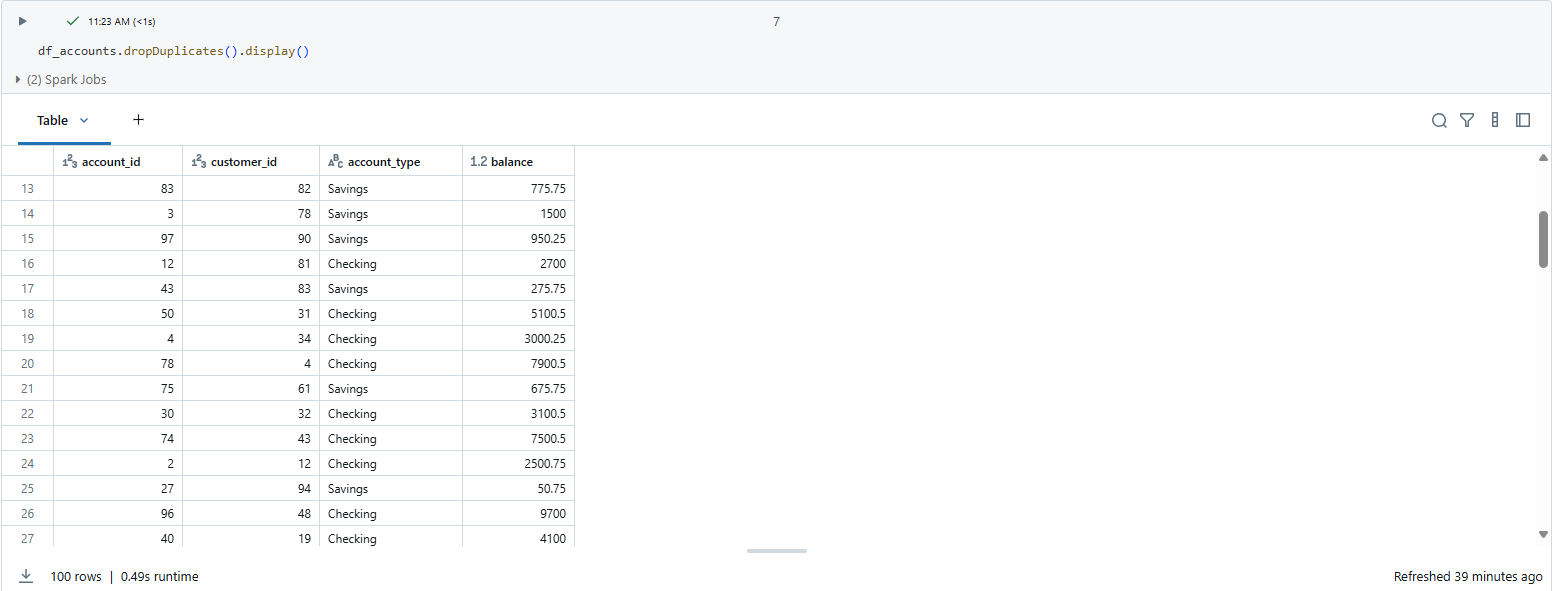
df\_accounts = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/mnt/pracnisha/silver/accounts.csv")

display(df\_accounts)



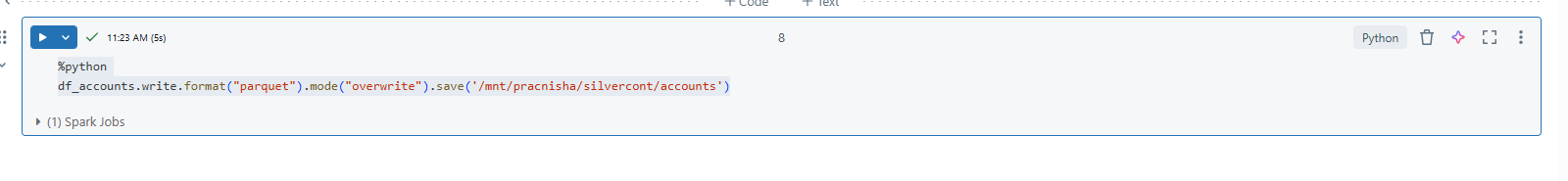
1. To remove duplicates use the below code

df\_accounts.dropDuplicates().display()



1. To store in parquet type

df\_accounts.write.format("parquet").mode("overwrite").save('/mnt/pracnisha/silvercont/accounts')



Repeat the same steps for other files as well

# Step 3: Use Notebooks using SCD Type technique (SCD 1)

# The below code is used for SCD -1 techniques

# To create delta table in Gold layer

%sql

CREATE TABLE IF NOT EXISTS account\_databricks (

  account\_id INT,

  customer\_id INT,

  account\_type STRING,

  balance INT,

  hashkey BIGINT,

  createddate TIMESTAMP,

  createdby STRING,

  updateby STRING,

  updatedate TIMESTAMP

)

USING DELTA

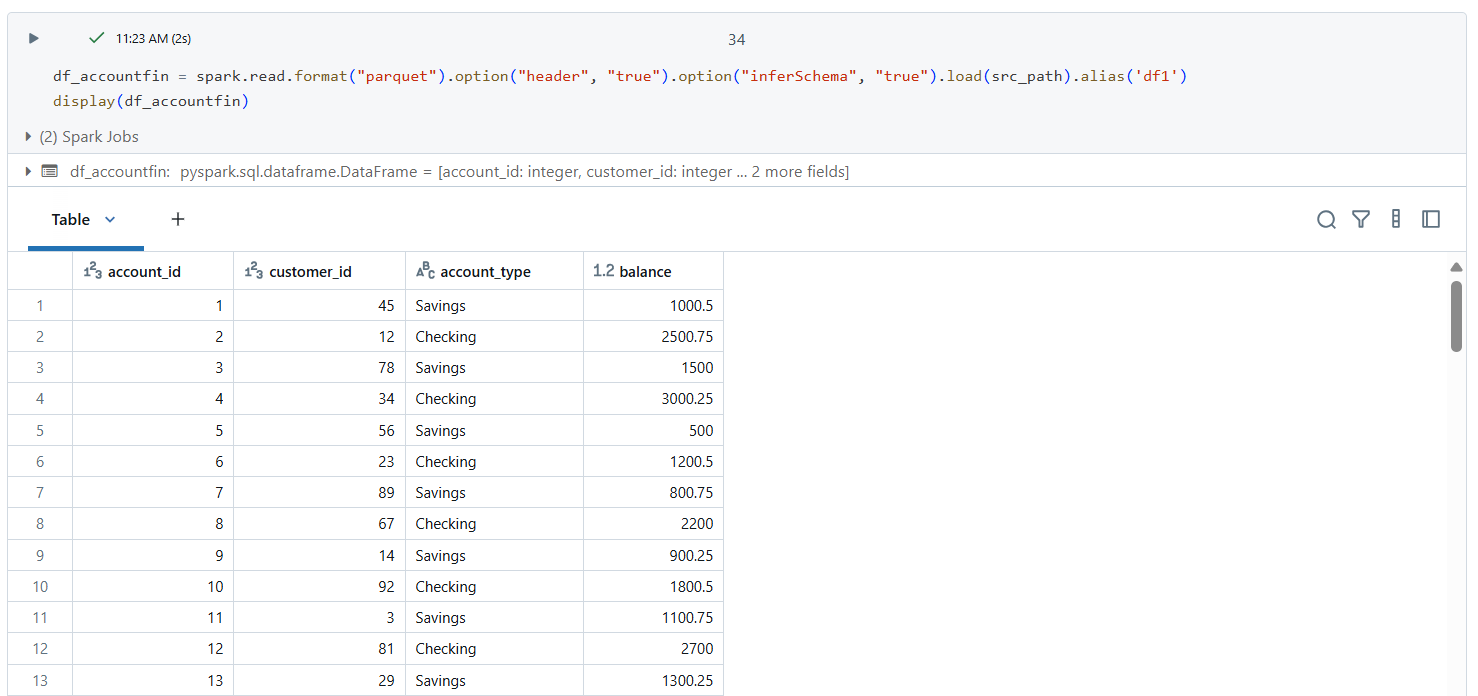
LOCATION "/mnt/pracnisha/gold/accounts";

# Read Source (Parquet) + Add HashKey

# To read parquet file

df\_accountfin = spark.read.format("parquet").option("header", "true").option("inferSchema", "true").load(src\_path).alias('df1')

display(df\_accountfin)



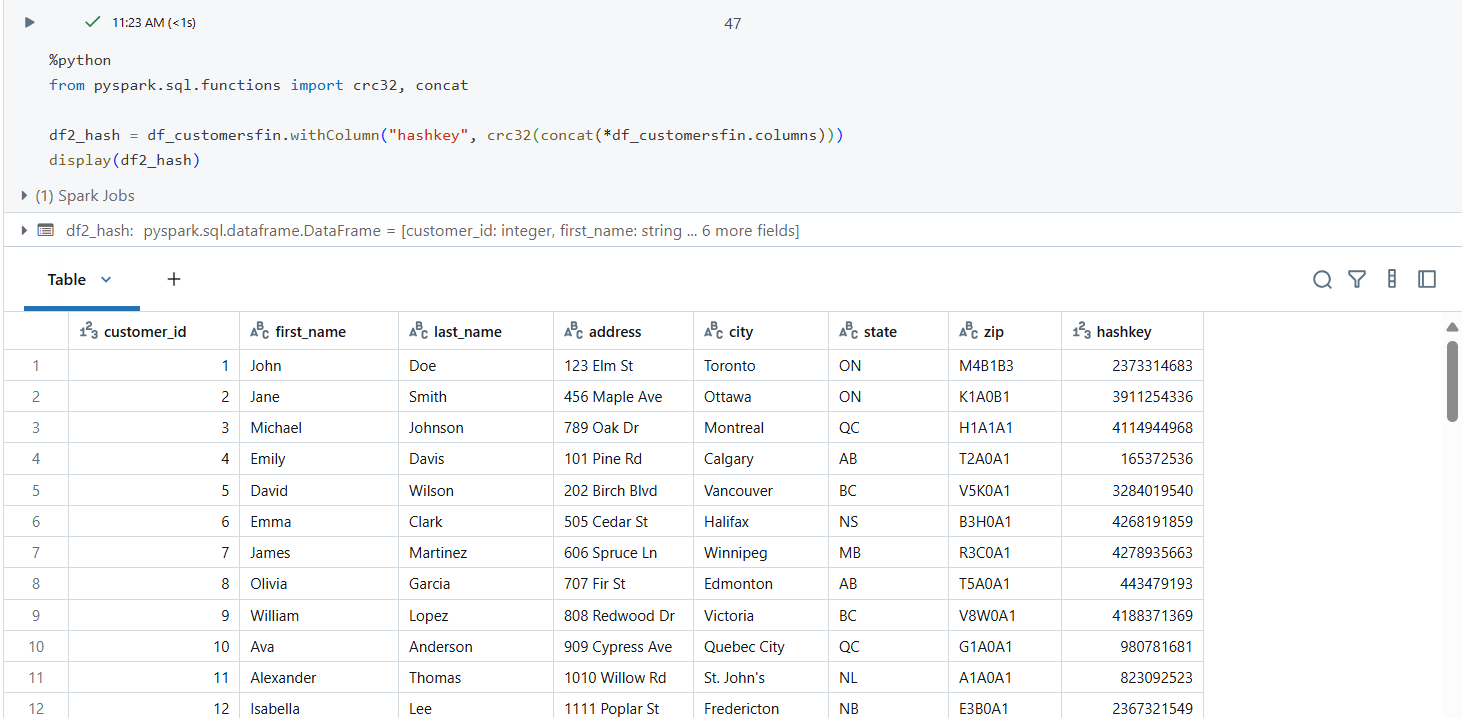
# To generate hashkey

%python

from pyspark.sql.functions import hash, concat

df\_hash = df\_accountfin.withColumn("hashkey", hash(concat(\*df\_accountfin.columns)))

display(df\_hash)



# Compare Source vs Target using HashKey

# Source path

src\_path = '/mnt/pracnisha/silvercont/accounts'

dbutils.fs.ls(src\_path)

# Target Path

tgt\_path = '/mnt/pracnisha/gold/accounts'

dbutils.fs.ls(tgt\_path)

# To compare the source and target using hashkey

%python

from pyspark.sql.functions import col

df\_source = df\_hash.alias("src").join(

    dtable.toDF().alias("tgt"),

    (col("src.account\_id") == col("tgt.account\_id")) & (col("src.hashkey") == col("tgt.hashkey")),

    "anti"

).select("src.\*")

display(df\_source)

# To perform delta merge using SCD-1 logic

%python

from pyspark.sql.functions import col, lit, current\_timestamp

dtable.alias("tgt").merge(

    df\_source.alias("src"),col("src.account\_id") == col("tgt.account\_id"))\

        .whenMatchedUpdate(set={

        "account\_id": col("src.account\_id"),

        "customer\_id": col("src.customer\_id"),

        "account\_type": col("src.account\_type"),

        "balance": col("src.balance"),

        "hashkey": col("src.hashkey"),

        "updatedate": current\_timestamp(),

        "updateby": lit("databricks-update")

    })\

        .whenNotMatchedInsert(values={

        "account\_id": col("src.account\_id"),

        "customer\_id": col("src.customer\_id"),

        "account\_type": col("src.account\_type"),

        "balance": col("src.balance"),

        "hashkey": col("src.hashkey"),

        "createddate": current\_timestamp(),

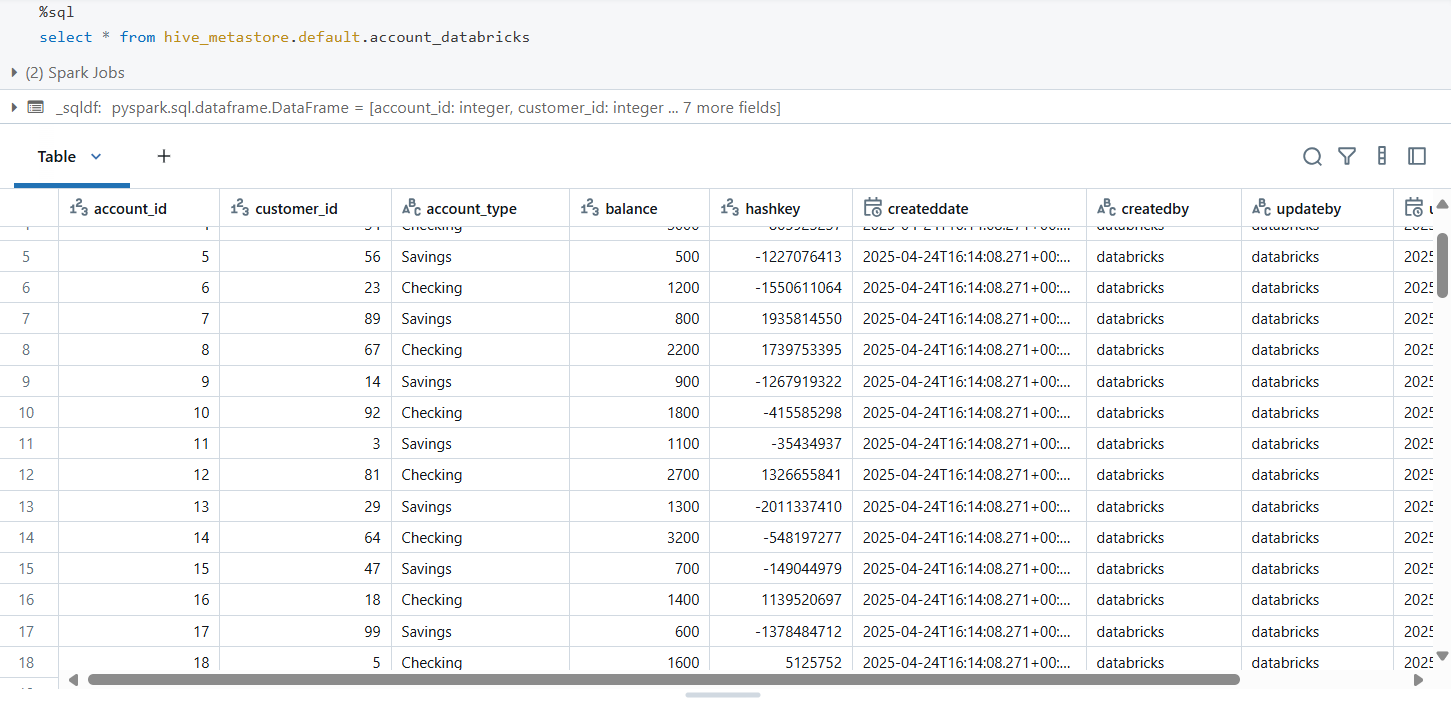
        "updatedate": current\_timestamp(),

        "createdby": lit("databricks"),

        "updateby": lit("databricks")

        }

        ).execute()



# Repeat the steps for the next four files

# Setting up the triggers and running the notebook on a specific time

# 

# Step 4: Use Power BI for Data Visualization

