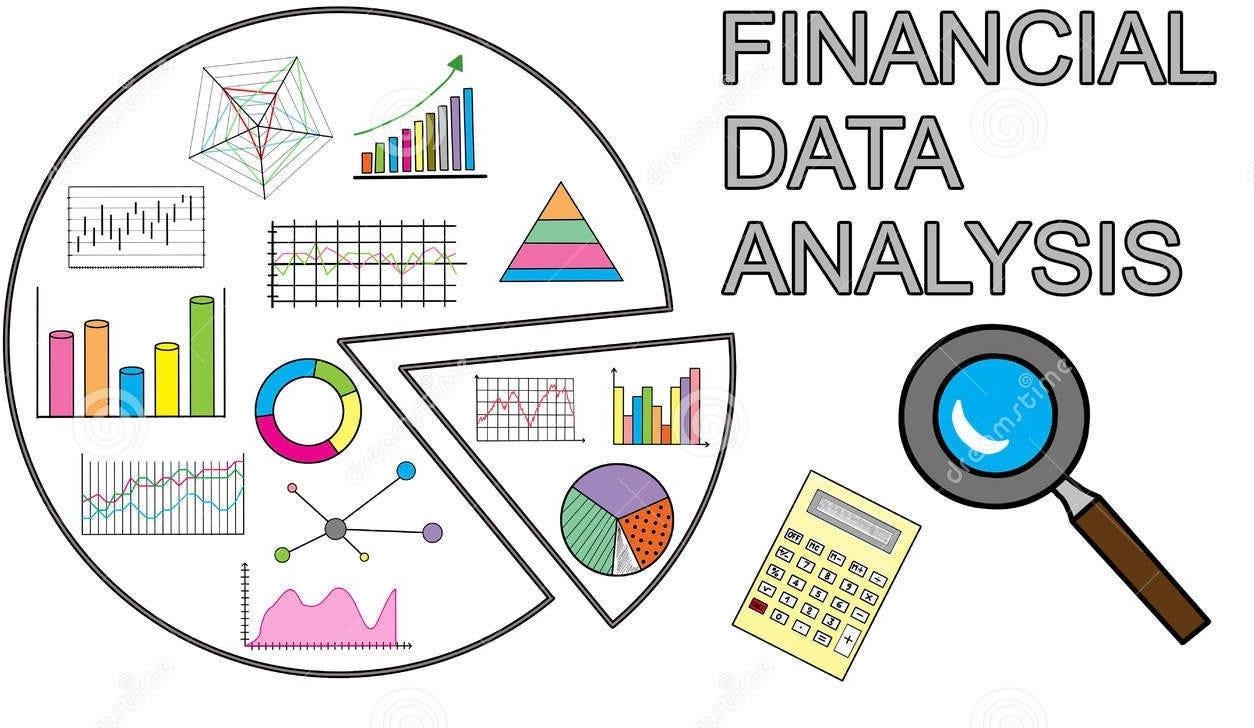
**Bootcamp Project 2 - Transactions and Loan Data for a Customer**

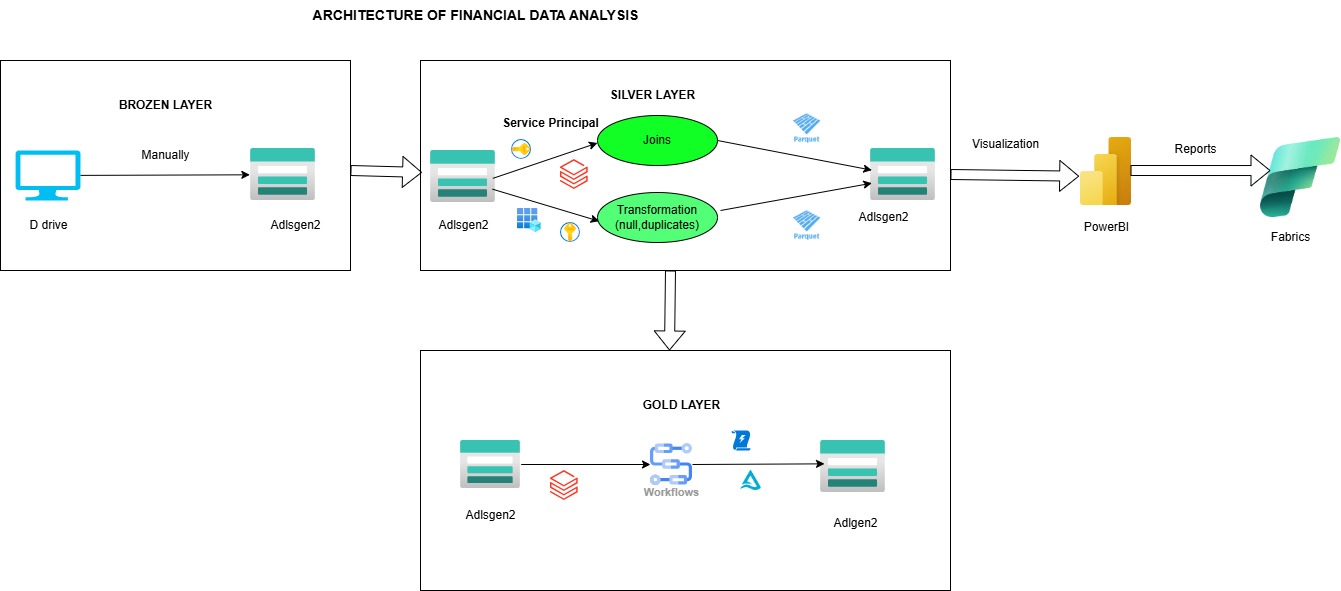


**GitHub link-**[**https://github.com/DeepthiChethi/Azure\_DE/upload/Madhumita/Project2**](https://github.com/DeepthiChethi/Azure_DE/upload/Madhumita/Project2)**.**

# **Project Overview**

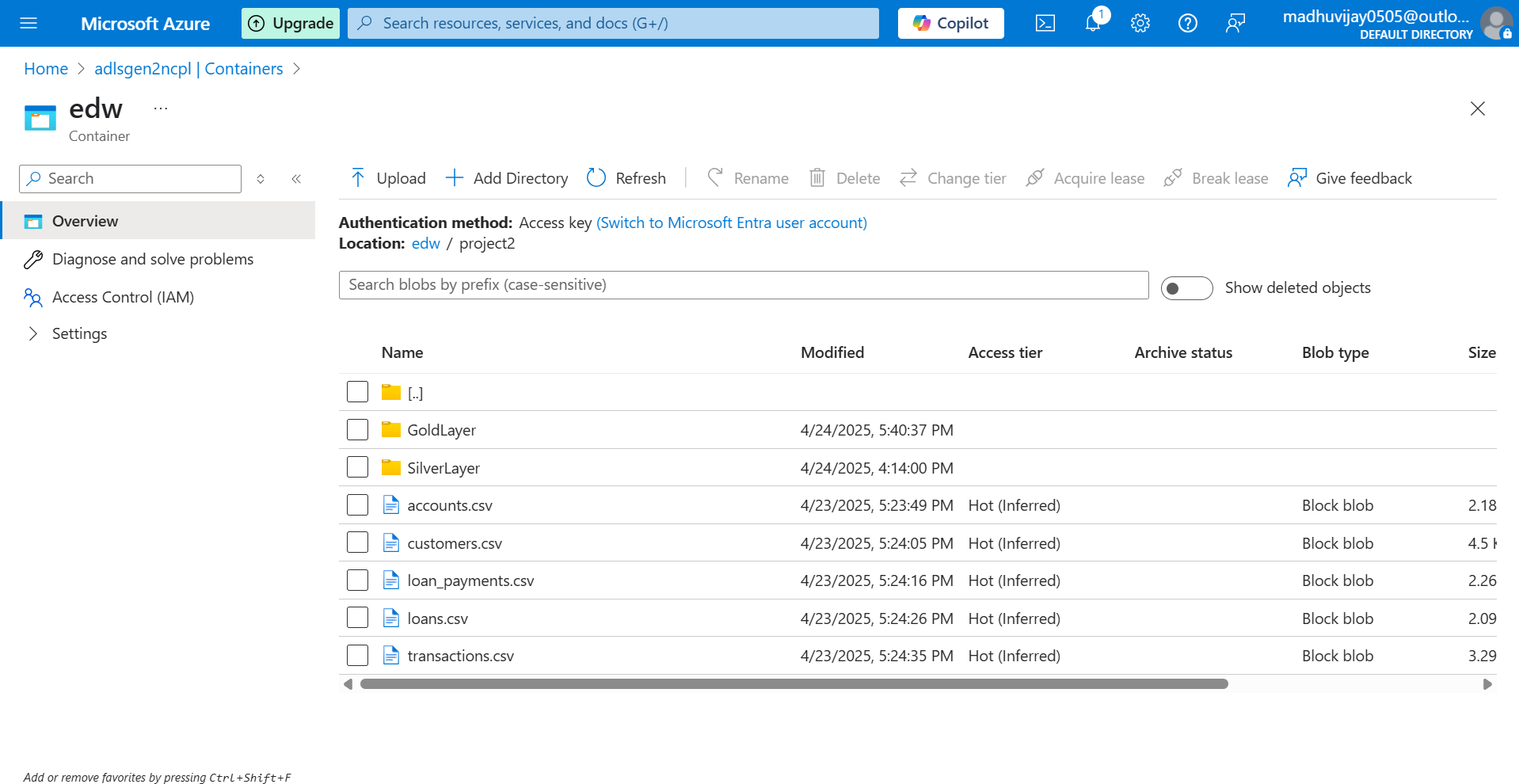
This project involves manually uploading five banking CSV files to Azure Data Lake Storage Gen2 (ADLS Gen2) and processing them using Azure Databricks Notebooks. The data is cleaned, transformed, and deduplicated in the silver layer, then written to the gold layer in Delta format using SCD Type 1 logic. Secure access is managed via Key Vault and mount points. Finally, the refined data is visualized in Power BI and published to the Microsoft Fabric Workspace.

# **Architecture of Financial Data Analysis**



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

**Upload the 5 files (accounts, customers, loan\_payments, loans, transactions) from d-drive to ADLSGen2 storage account manually.**

****

# **Step 2-: Use Databricks Notebooks to remove the duplicates**

The five files will be read from ADLS Gen2, processed through joins and transformations, and then saved back to ADLS Gen2 in Parquet format.

Connect ADLSGEN2 in databricks using service principal method

Service principal method is called as applicationId. Instead of creating individual user, create service principal with necessary permission.

Create service principal

EntraId->Manage->App registration->new reg->bootproj2->click on 2nd option->register

Copy the tenantd and AppId and note it down.

Appid-0f7a5183-5189-4671-a6ed-dcc823b8551e

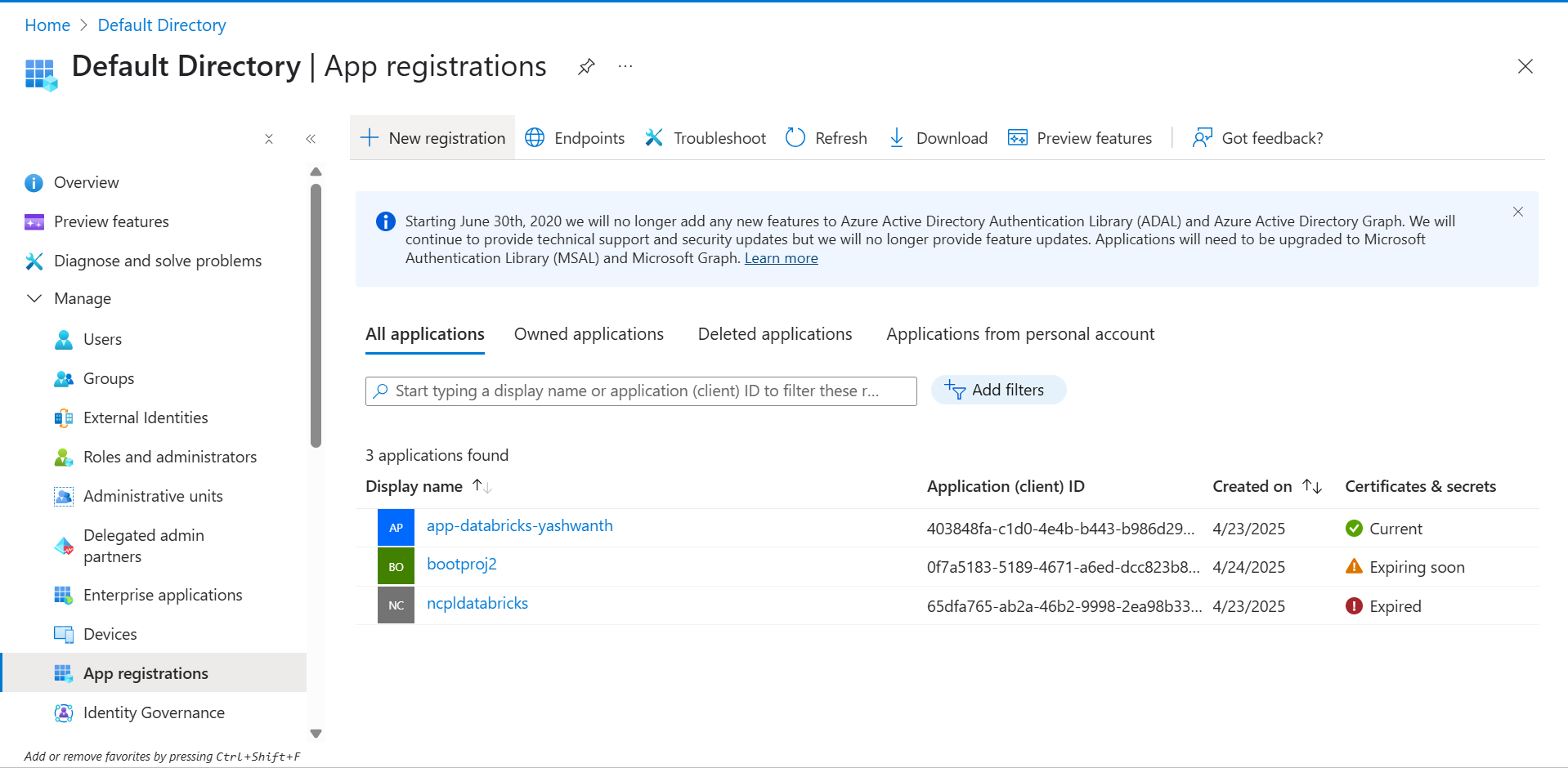
Tenanted-c34955b6-1f30-42b7-9c26-75943db8bccc

Click on client credentials->+new client secret-> description(add description)->Select the custom date.

Copy the value and secretID

Valueid-Lfo8Q~Yi\_PheSDK3\_pk2FFk4LjF1BaKgpqfCic3Y

secretId-b296c166-999d-4d2d-bf56-bd4ea8598e75



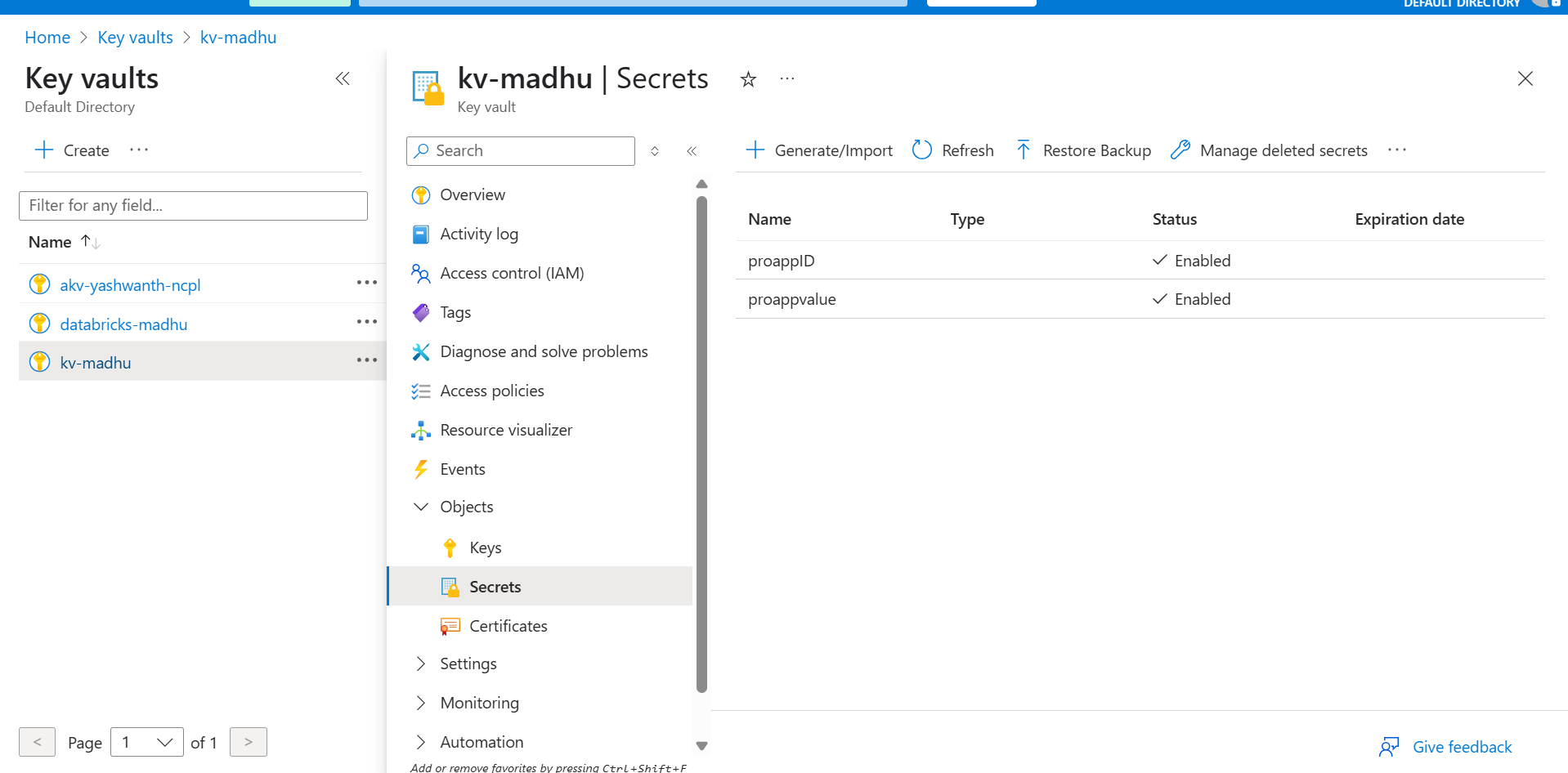
Go to key vault -> create secret keys

Name-proappID

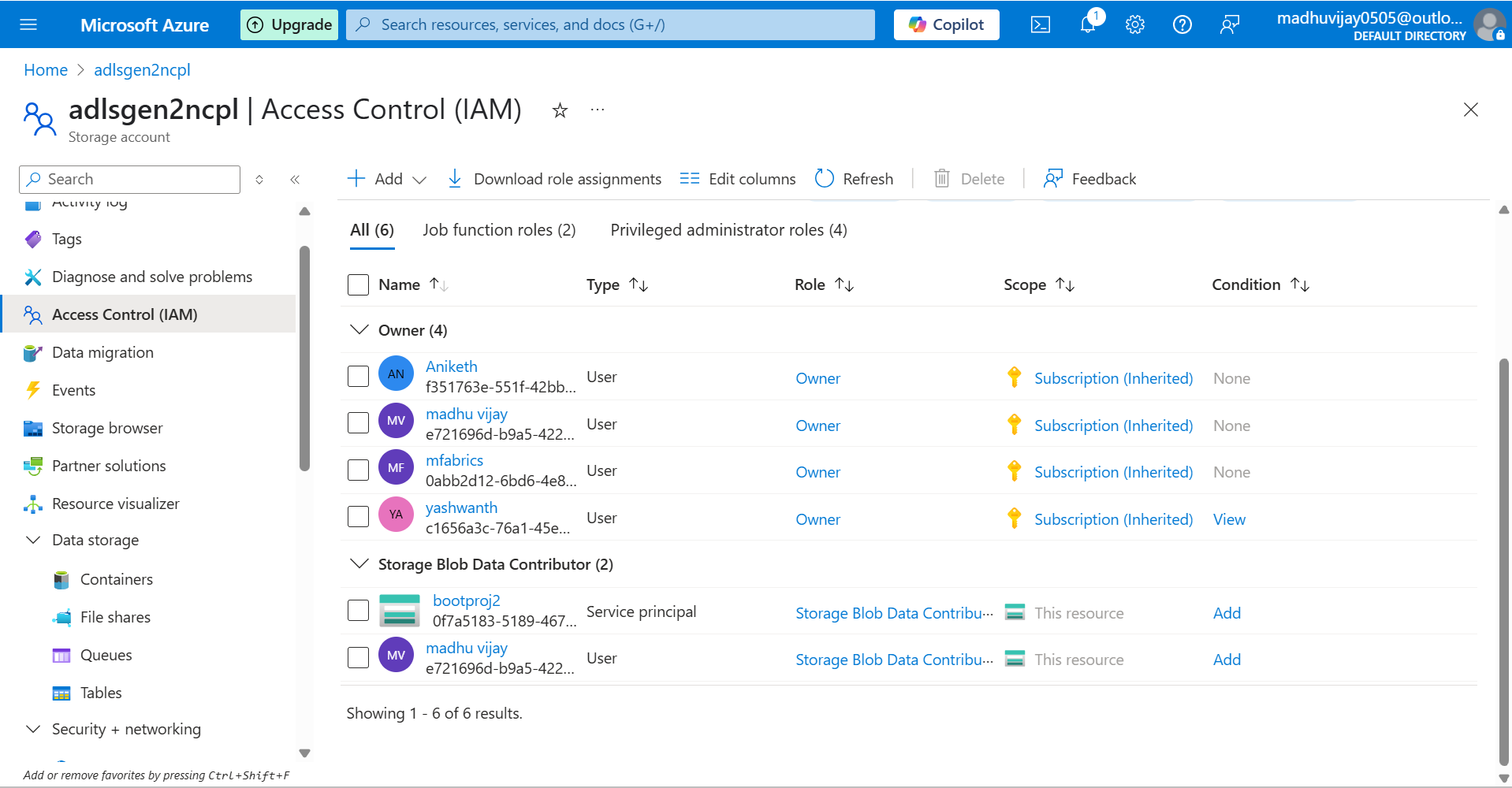
Secret- add AppiD(0f7a5183-5189-4671-a6ed-dcc823b8551e)

Name-proappvalue

Secret-add valueId(Lfo8Q~Yi\_PheSDK3\_pk2FFk4LjF1BaKgpqfCic3Y)



In ADLSGEN2, add storage blob data contributor role for service principal id and create it.



Go to databricks, and start the cluster.

Now, create scope by remove up to .net in URL. Add #secrets/createScope

In Azure key vault -> Properties-> Copy vault URI and resource id

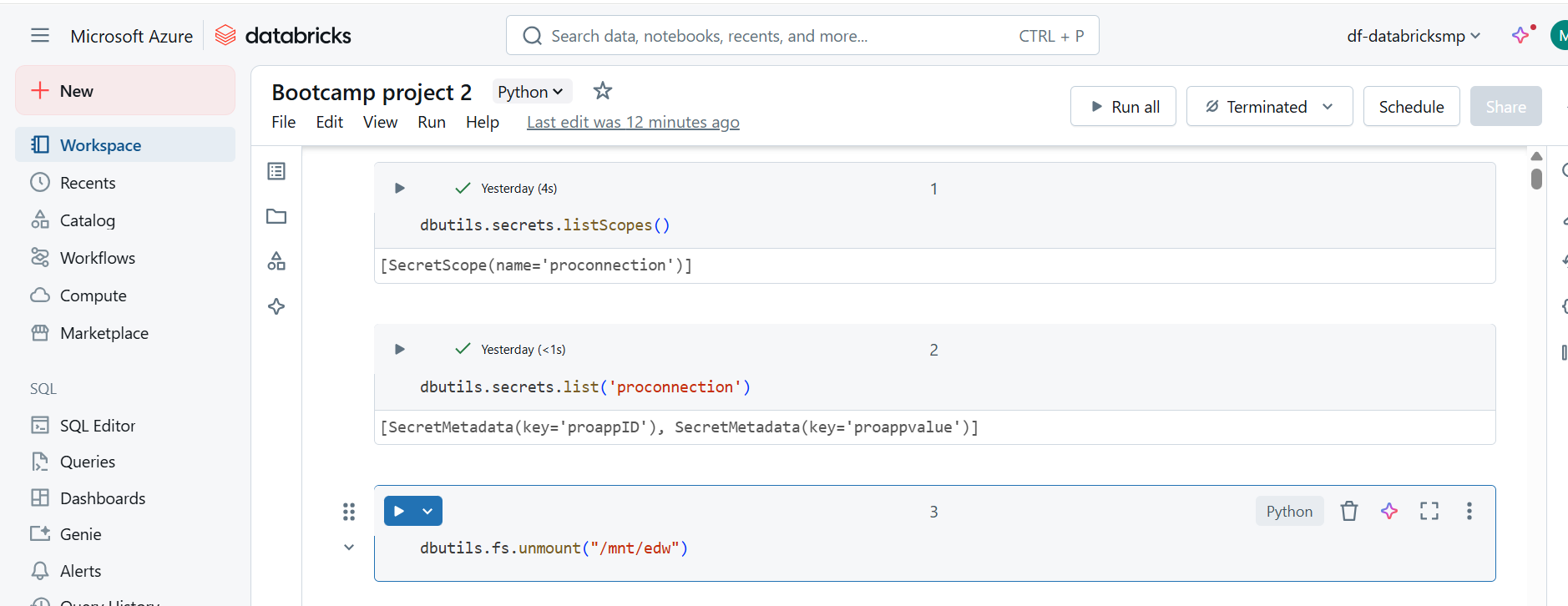
DNS name – vault URI(https://kv-madhu.vault.azure.net/)

Resource ID – ResourceID(/subscriptions/94146c24-6ae4-4b32-bd61-6674630981d5/resourceGroups/rg-madhu/providers/Microsoft.KeyVault/vaults/kv-madhu)

Check the Secretscope and SecretKeys using below commands

**dbutils.secrets.listScopes()**

**dbutils.secrets.list('proconnection')**



Mount ADLSGEN2 using the below command

**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="proconnection",key="proappID"),**

**"fs.azure.account.oauth2.client.secret":dbutils.secrets.get(scope="proconnection",key="proappvalue"),**

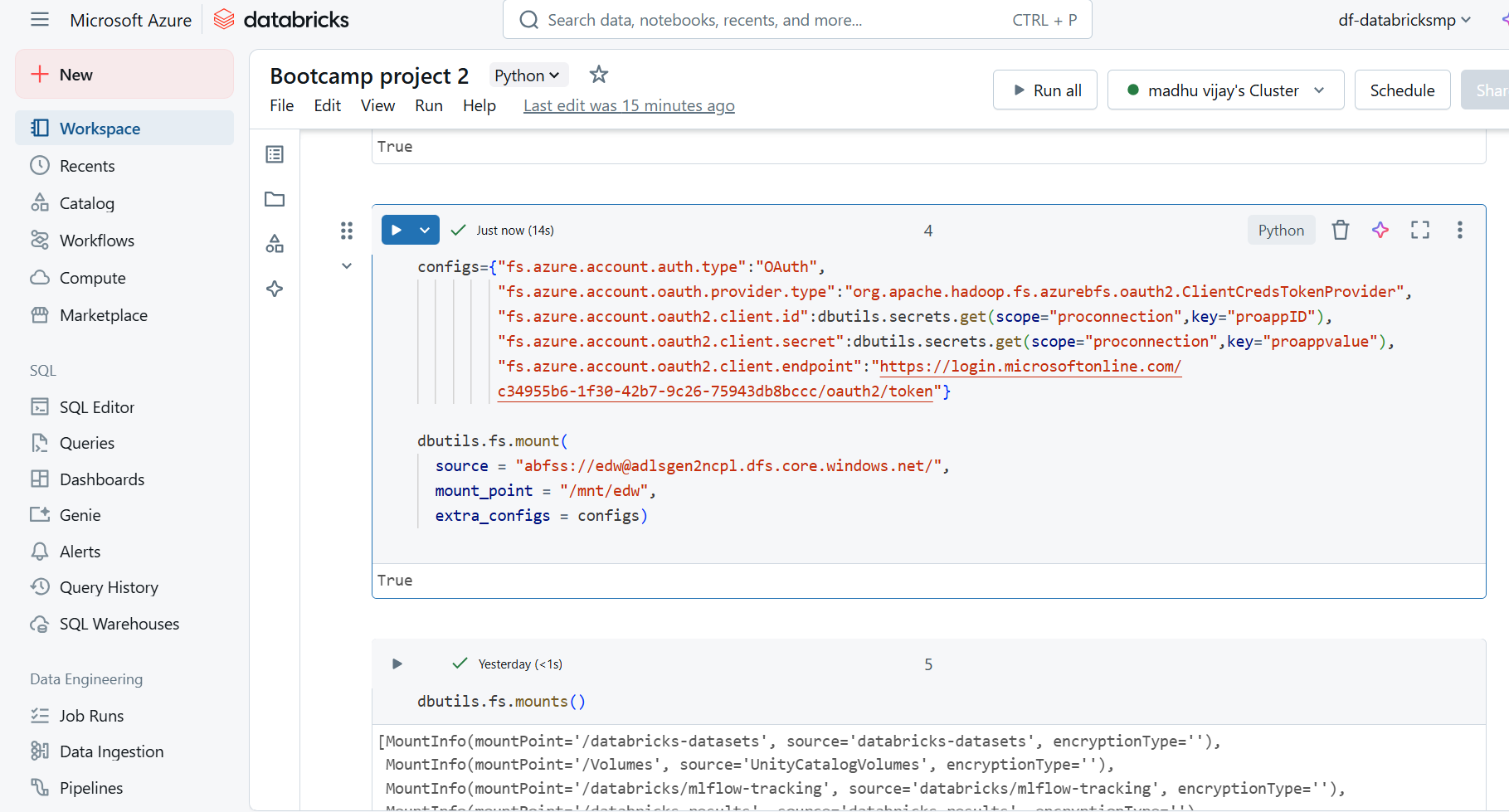
**"fs.azure.account.oauth2.client.endpoint":"https://login.microsoftonline.com/c34955b6-1f30-42b7-9c26-75943db8bccc/oauth2/token"}**

**dbutils.fs.mount(**

**source = "abfss://edw@adlsgen2ncpl.dfs.core.windows.net/",**

**mount\_point = "/mnt/edw",**

**extra\_configs = configs)**

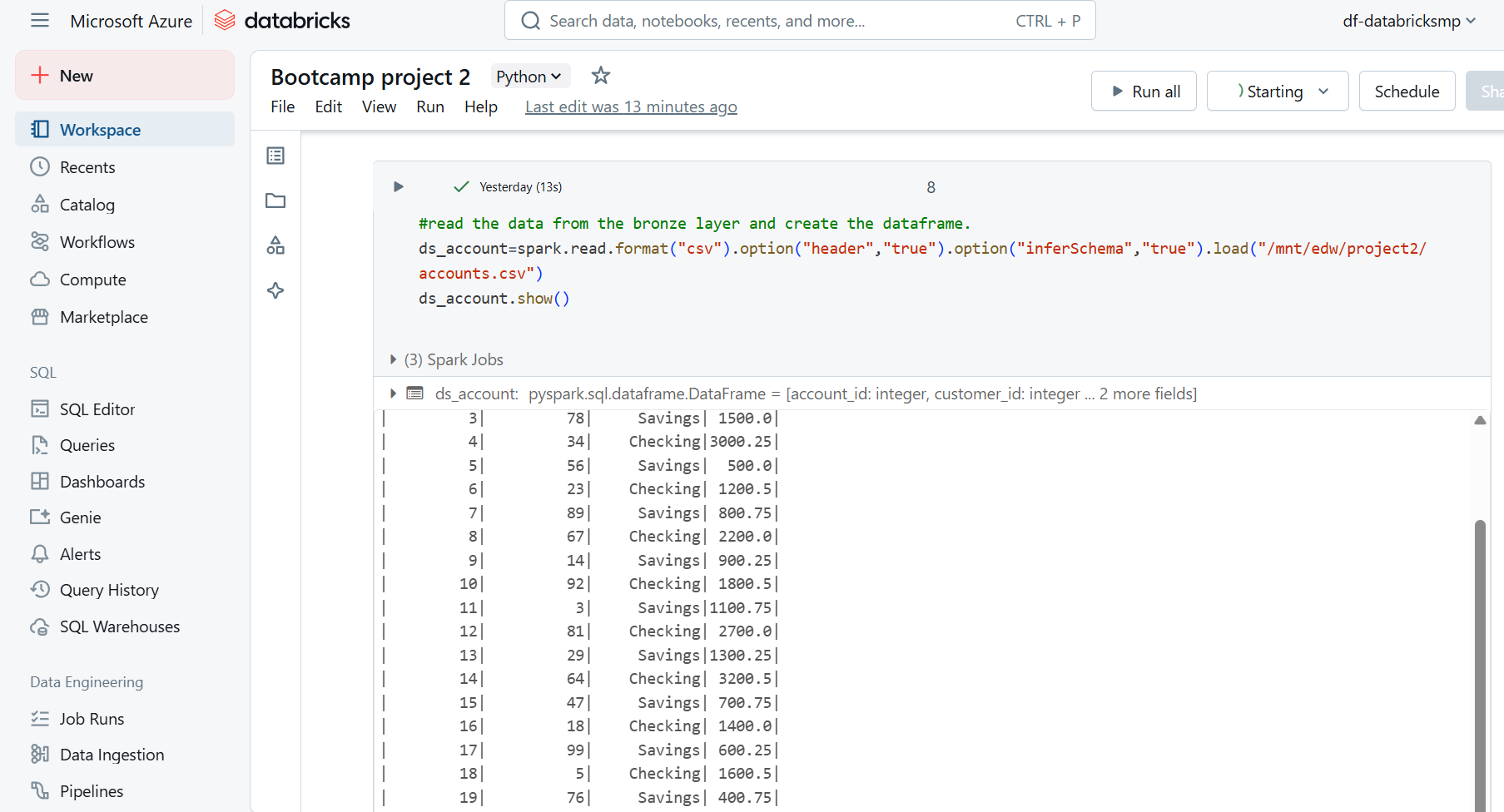


Read the data using spark.read.format.

**#read the data from the bronze layer and create the dataframe.**

**ds\_account=spark.read.format("csv").option("header","true").option("inferSchema","true").load("/mnt/edw/project2/accounts.csv")**

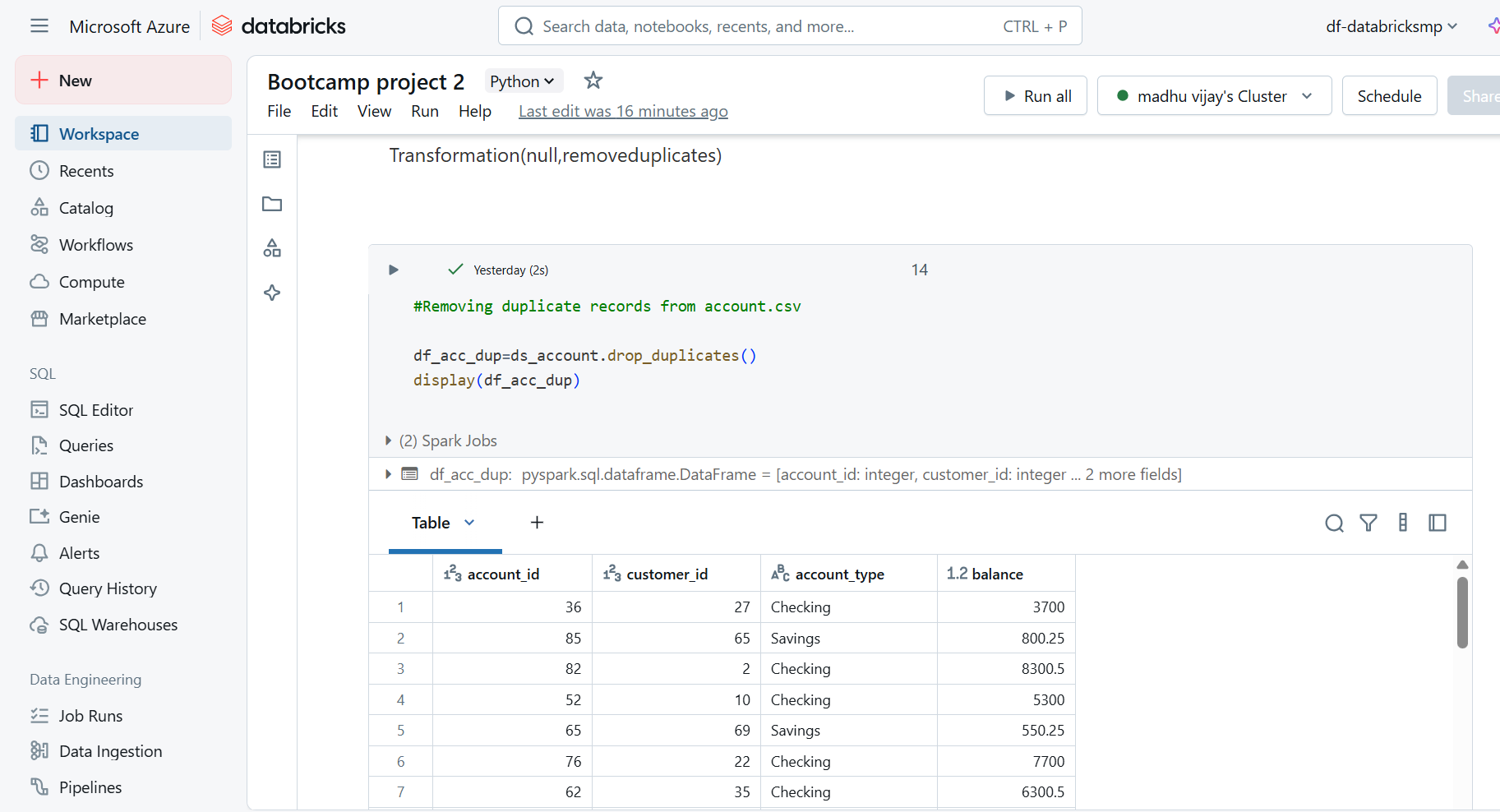
**ds\_account.show()**



Removing duplicate records from accounts.csv

**df\_acc\_dup=ds\_account.drop\_duplicates()**

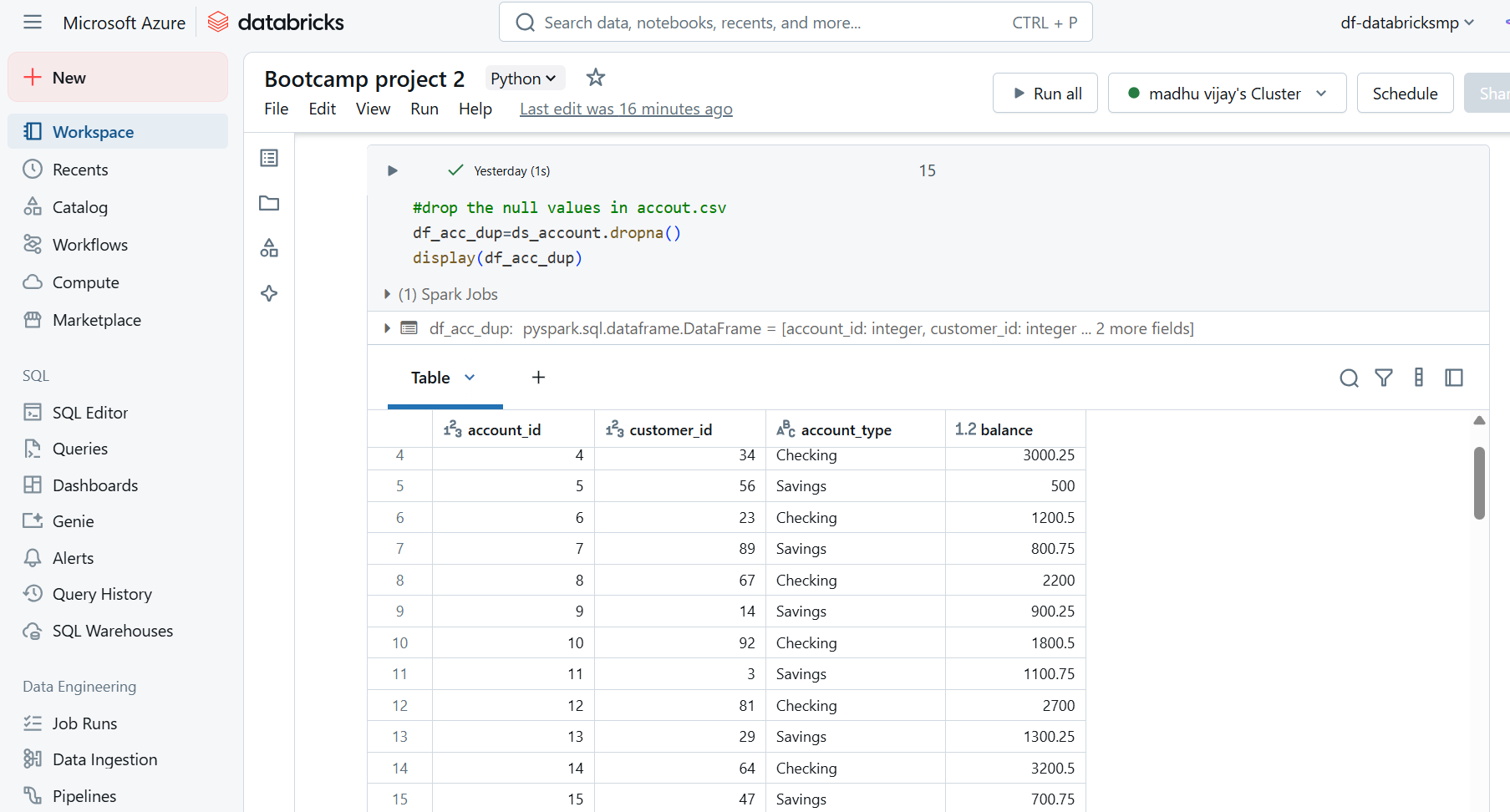
**display(df\_acc\_dup)**



Dropping null values in account.csv

**df\_acc\_dup=ds\_account.dropna()**

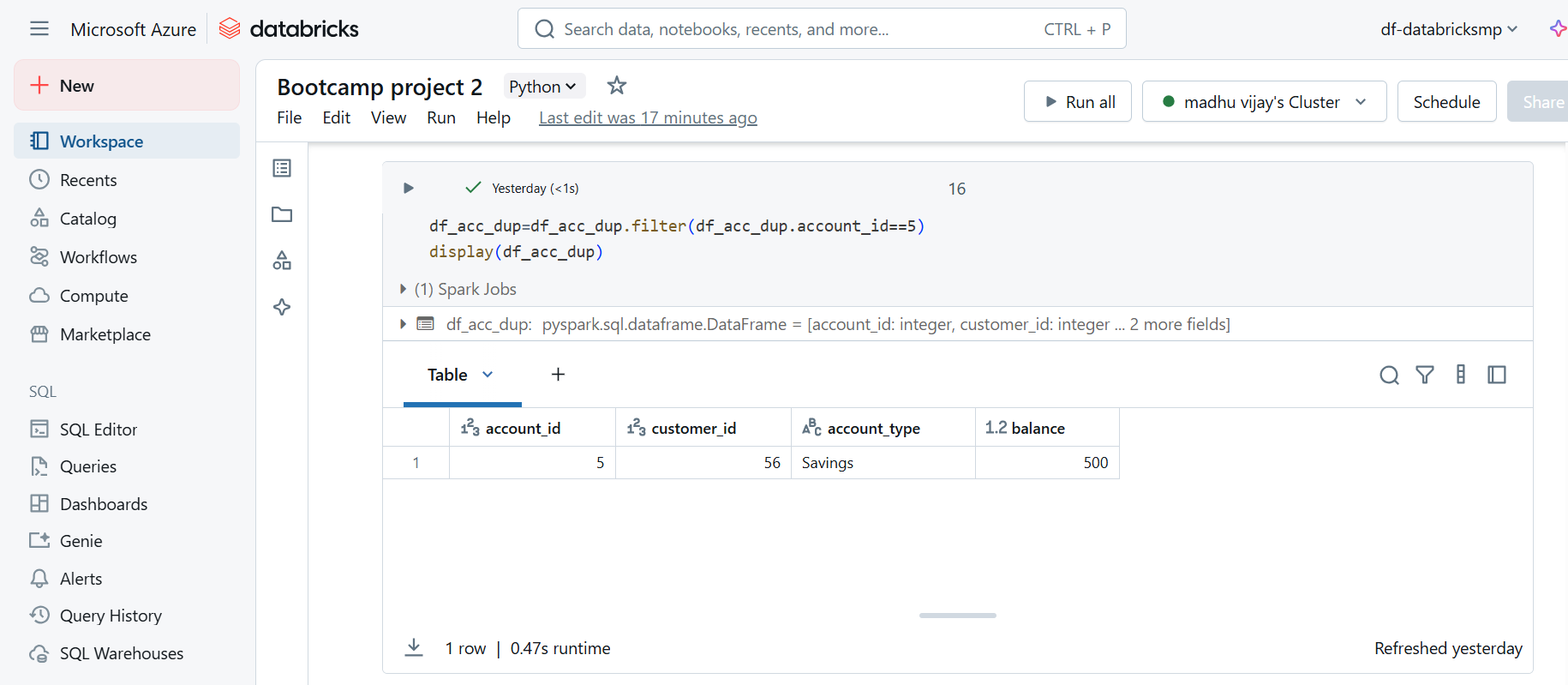
**display(df\_acc\_dup)**



Using filter condition to filter the data.

**df\_acc\_dup=df\_acc\_dup.filter(df\_acc\_dup.account\_id==5)**

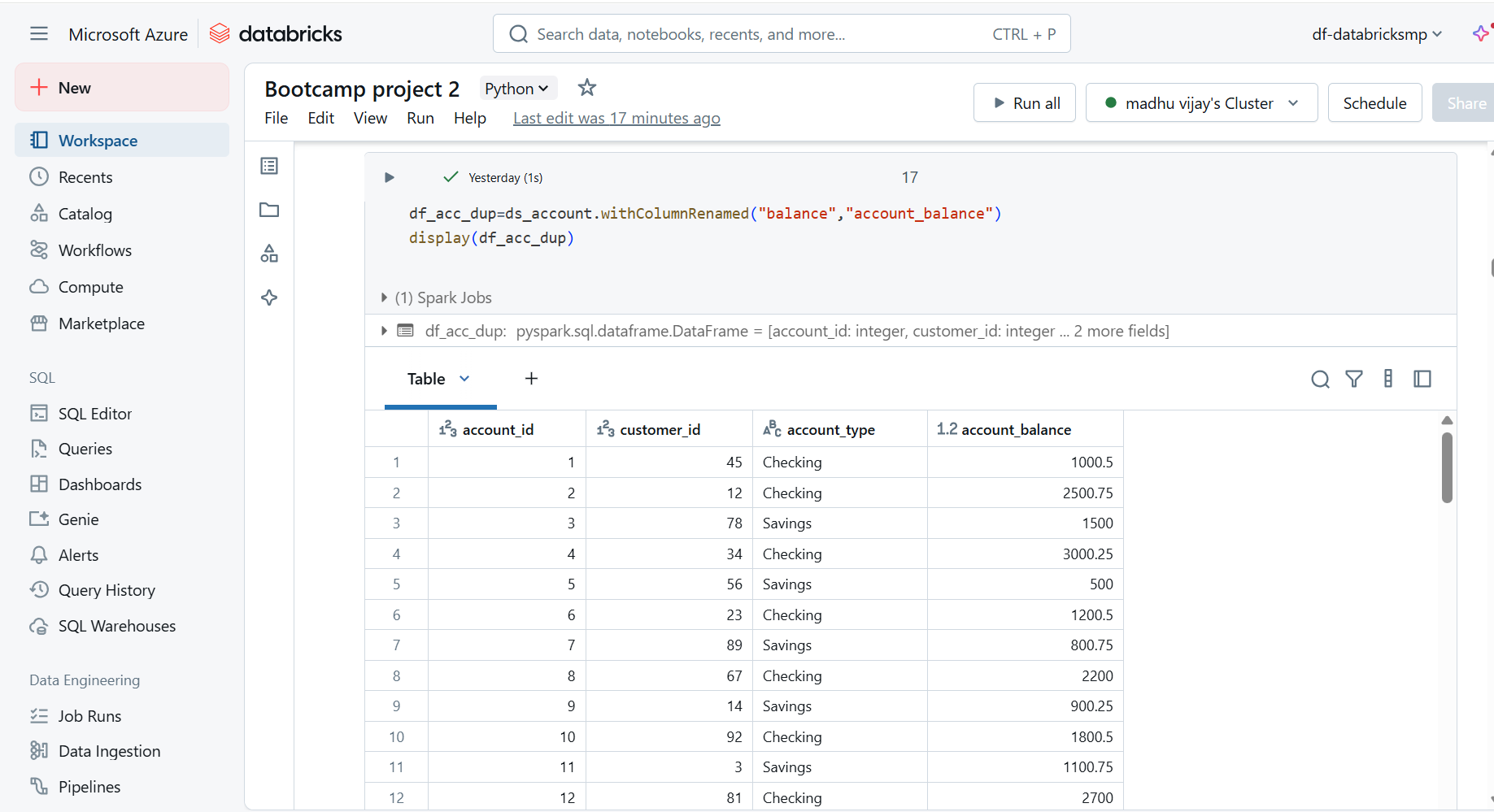
**display(df\_acc\_dup)**



Using rename condition to rename the column

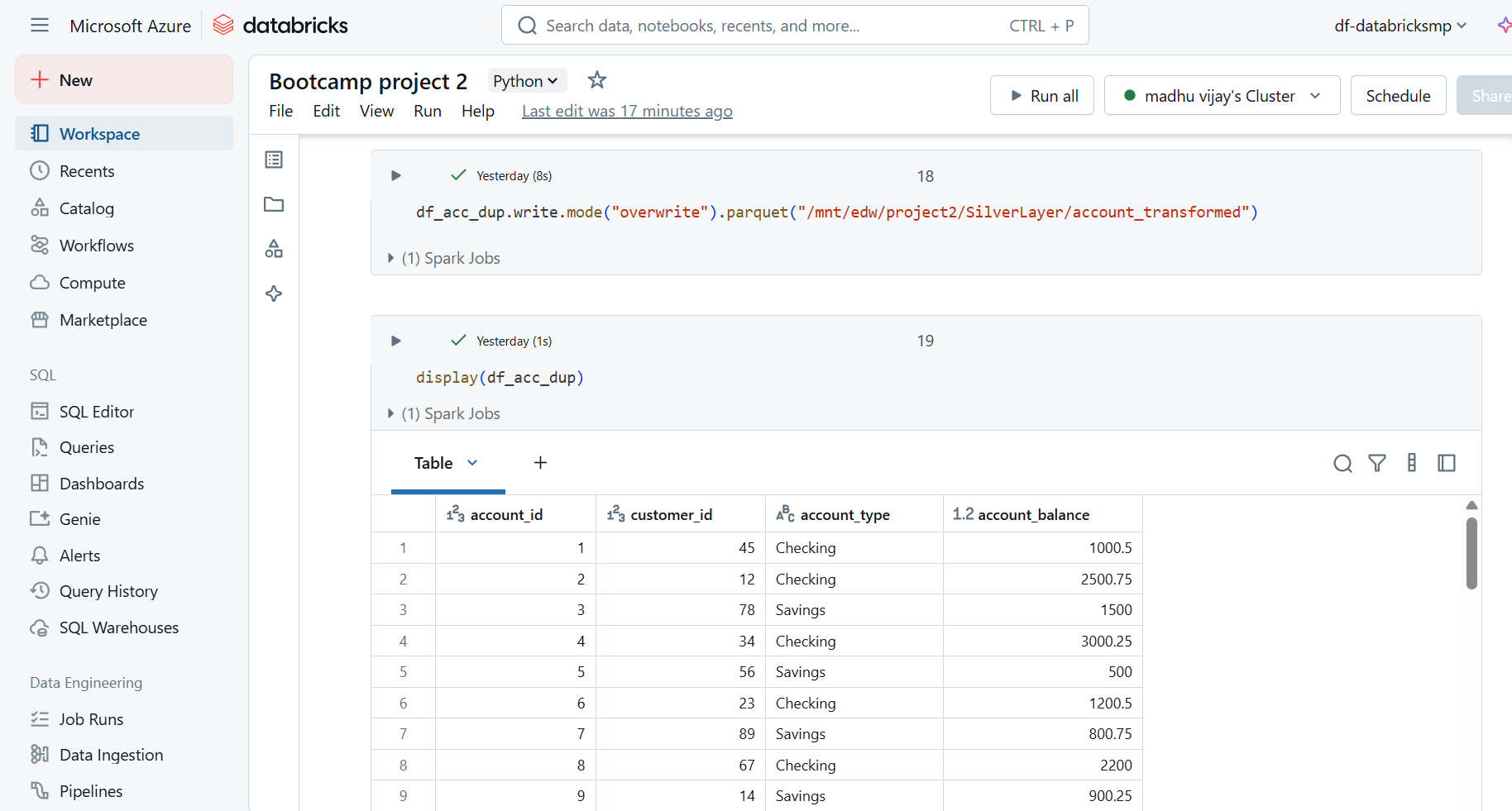
**df\_acc\_dup=ds\_account.withColumnRenamed("balance","account\_balance")**

**display(df\_acc\_dup)**



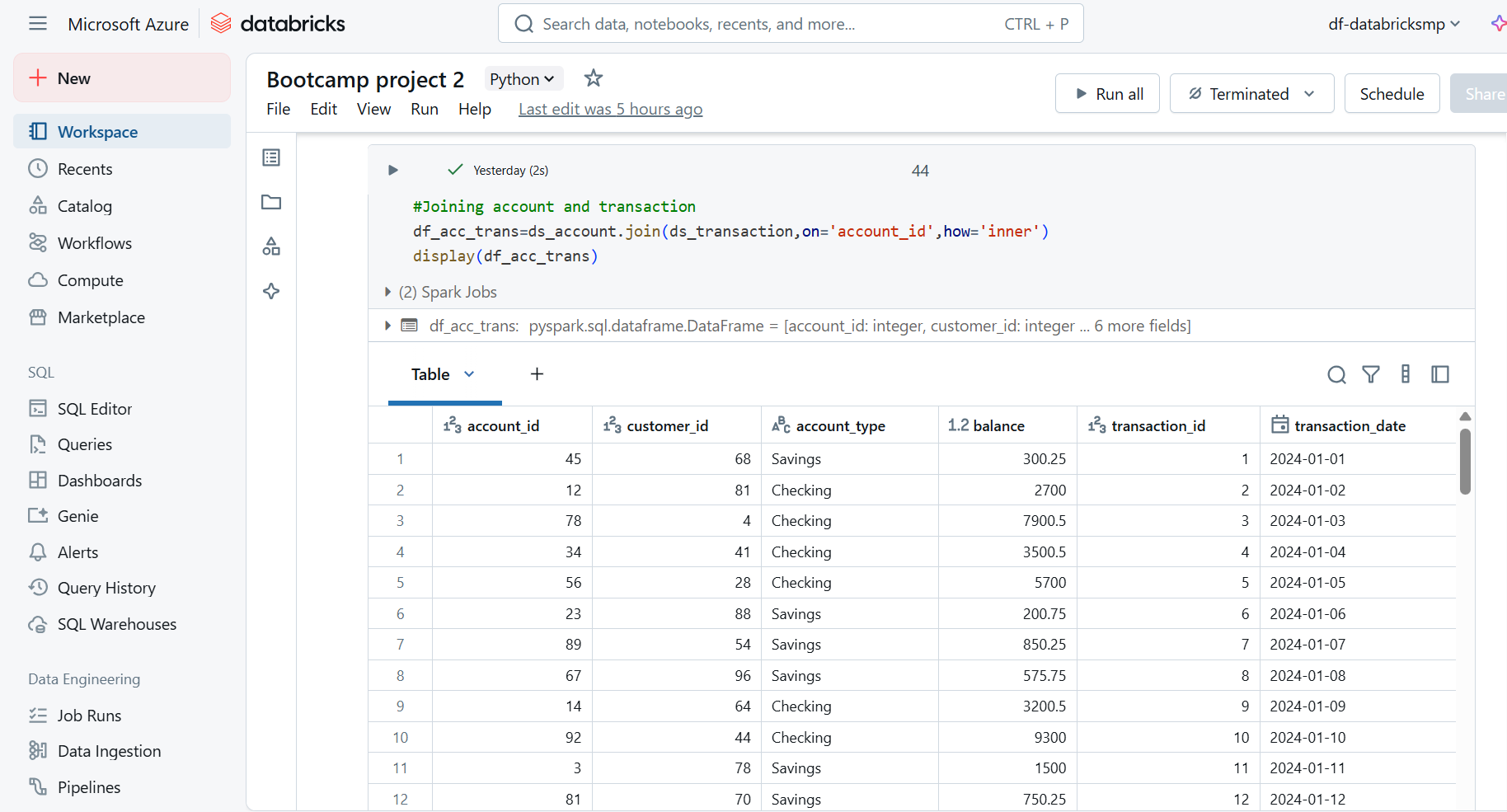
Using write mode for saving the file in parquet format

**df\_acc\_dup.write.mode("overwrite").parquet("/mnt/edw/project2/SilverLayer/account\_transformed")**

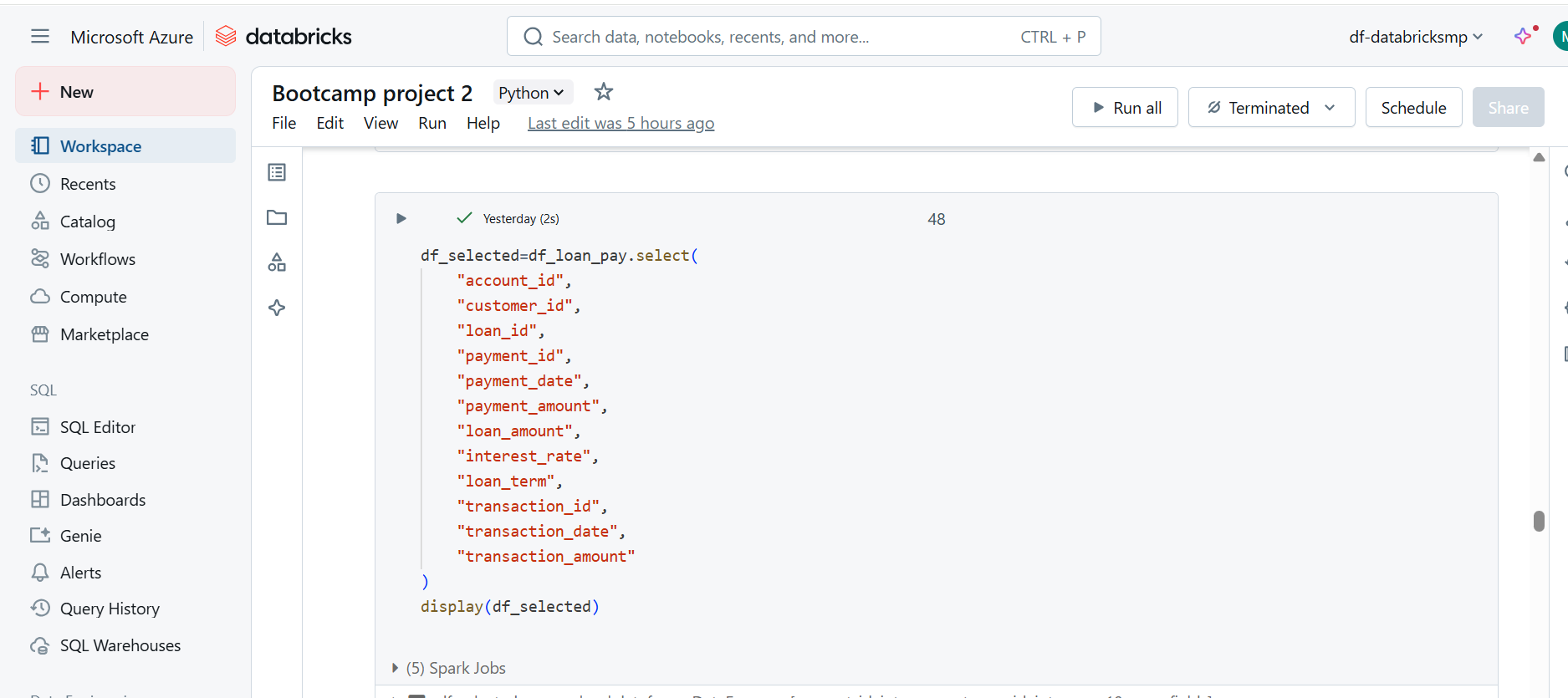


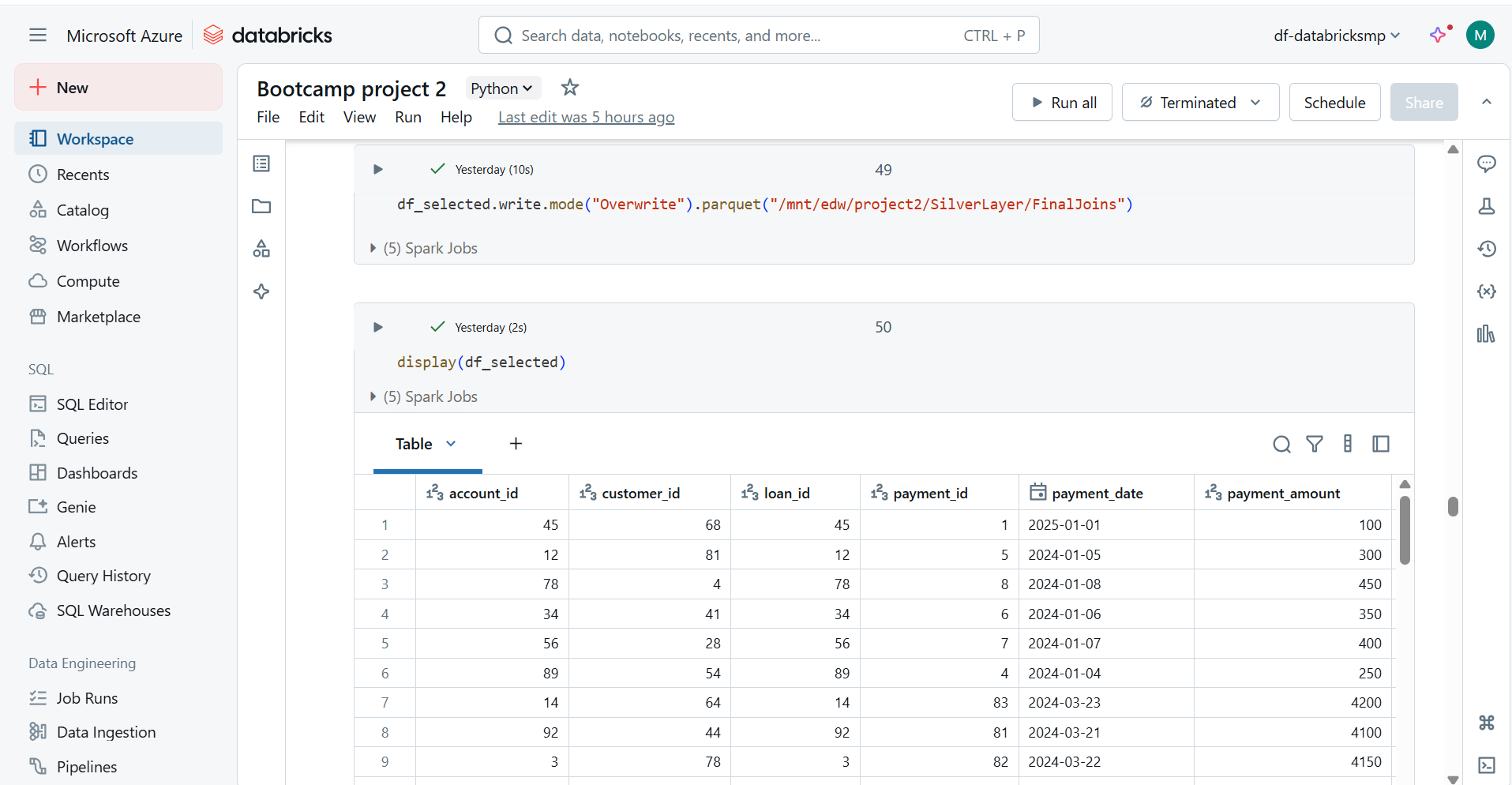
**Joins**

**Using inner joins to combine account and transaction since it has account\_id in common.**

****

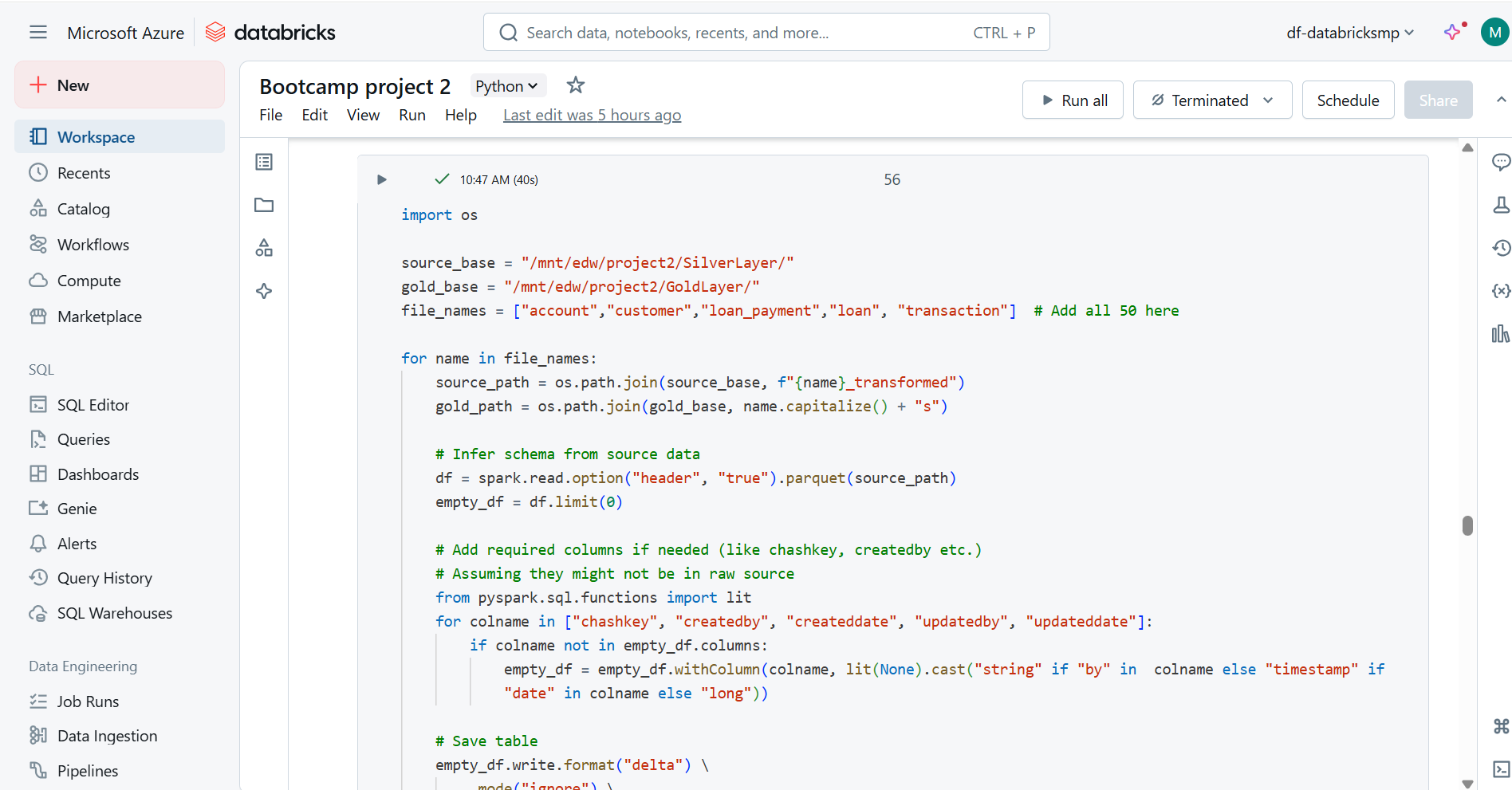
Combine the select fields and save it in parquet file

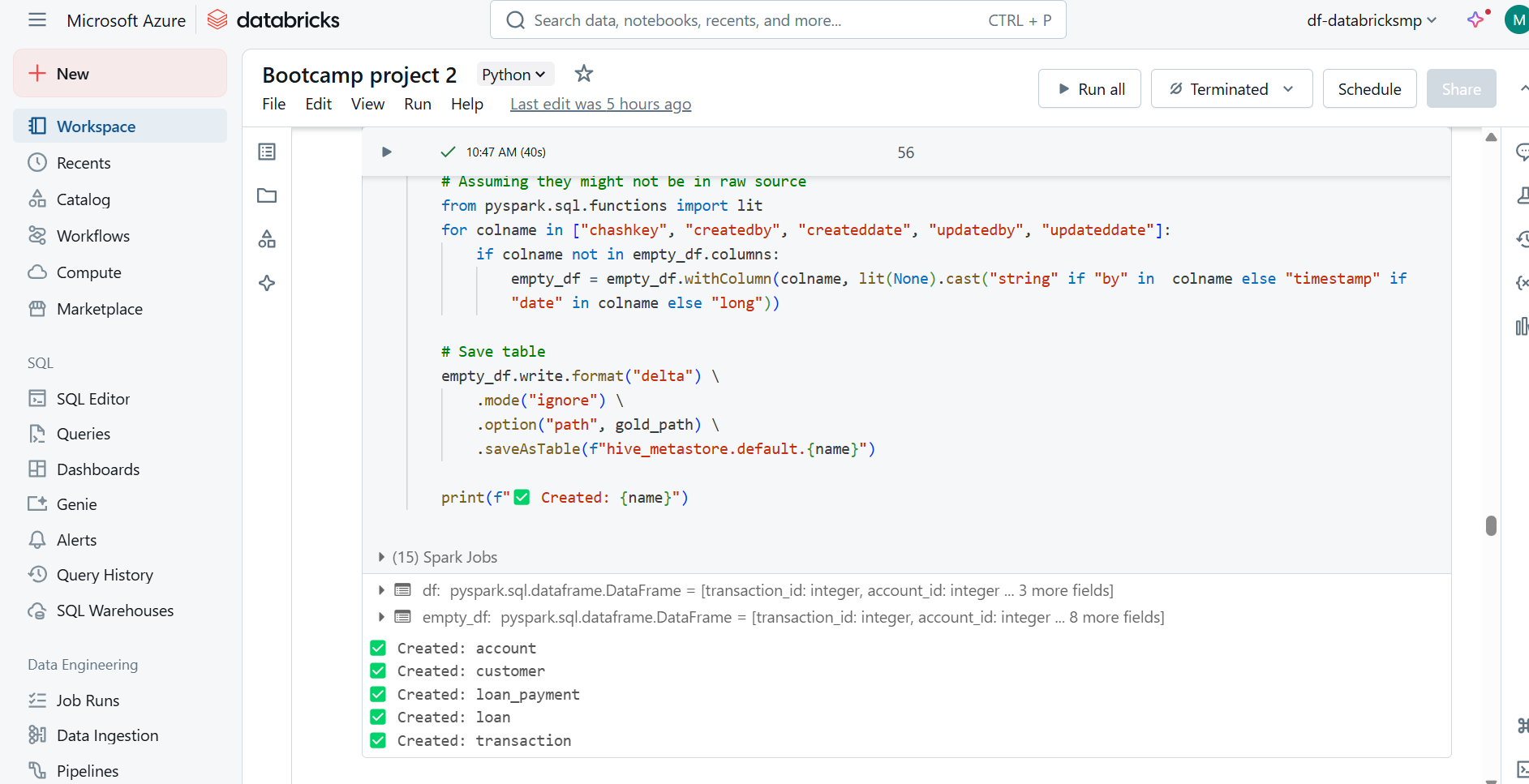




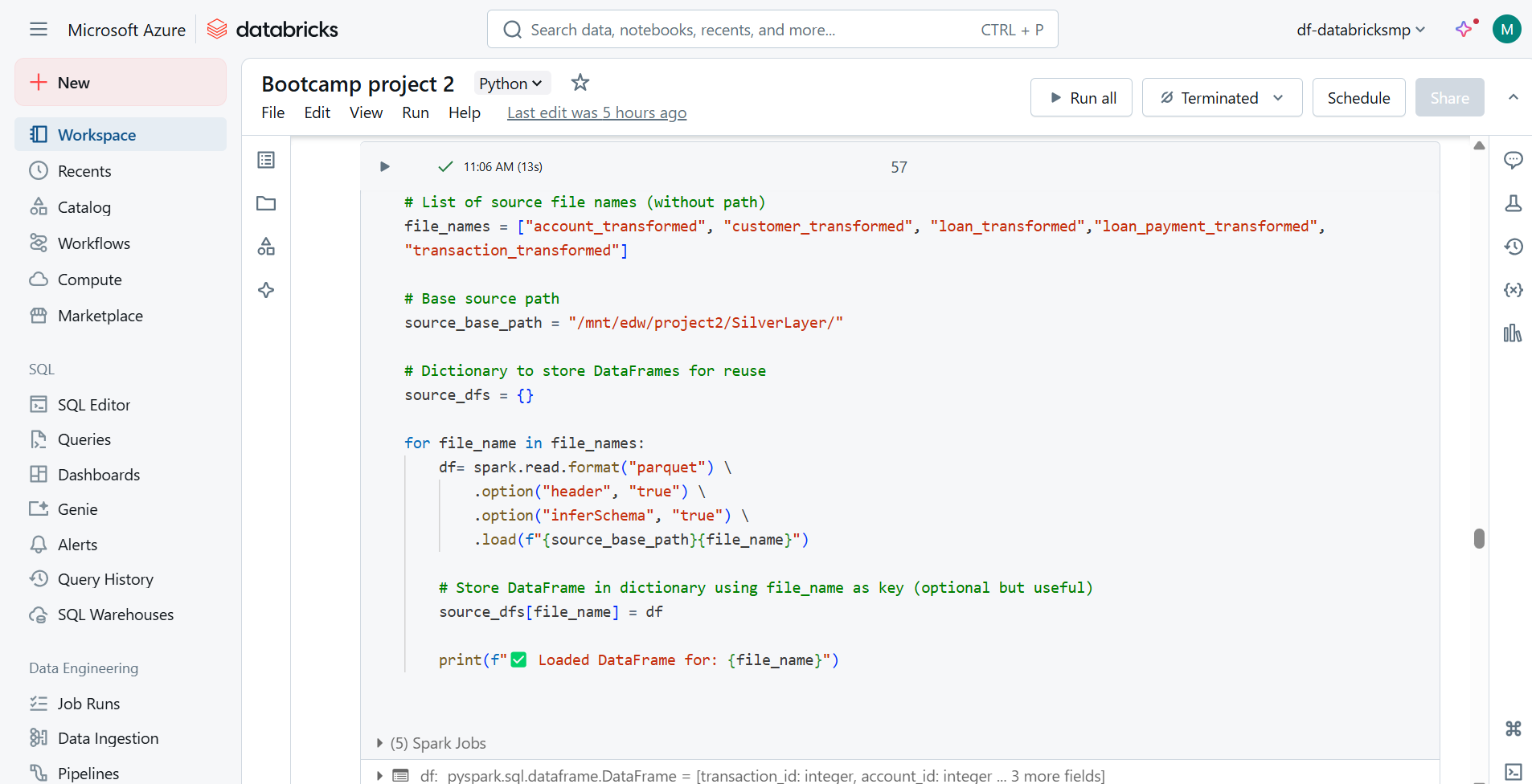
# **Step 3: Use Notebooks using SCD Type techniques (SCD 1)**

1. Create 5 target tables using pyspark and save it in hive metastore

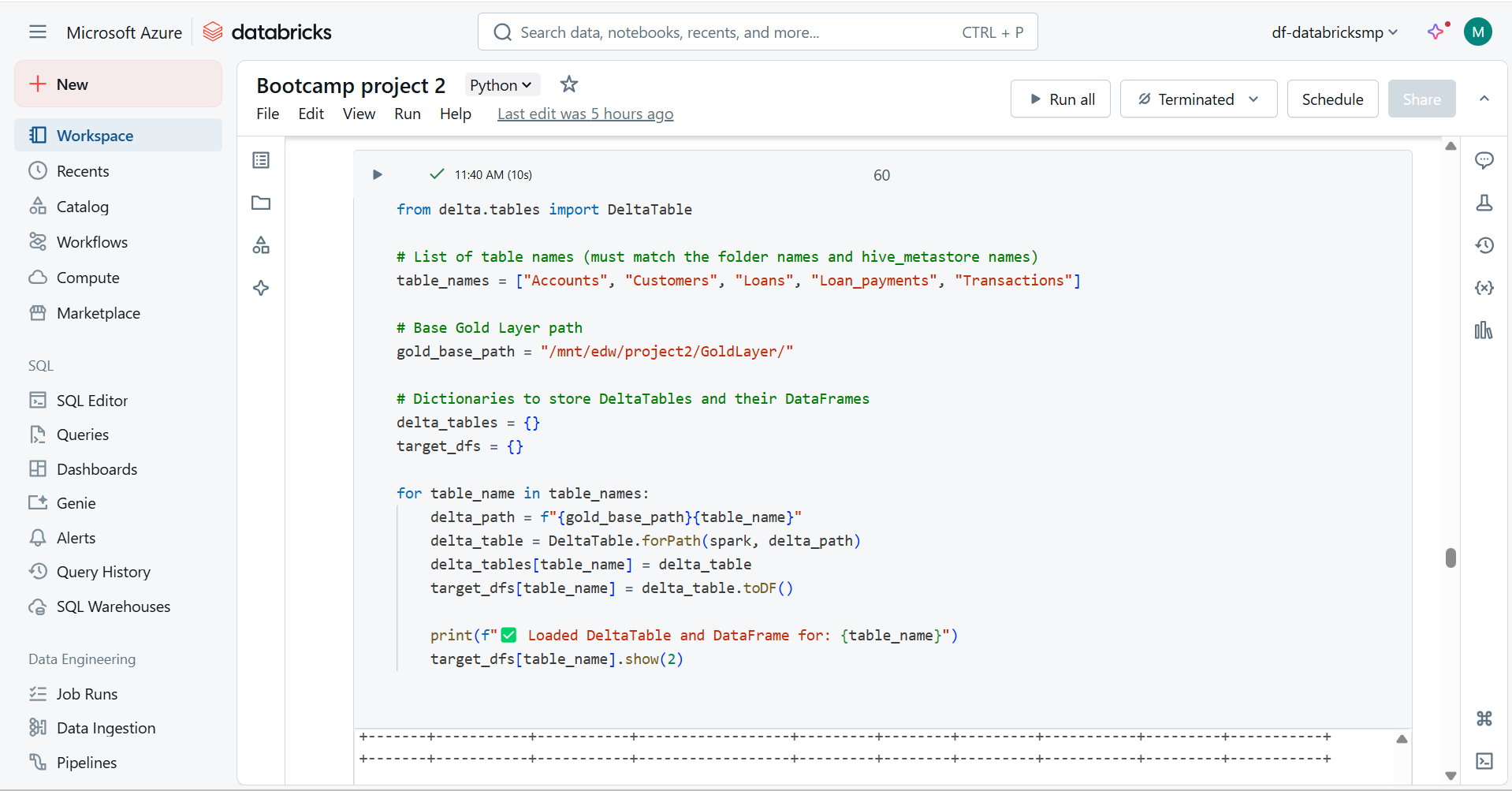


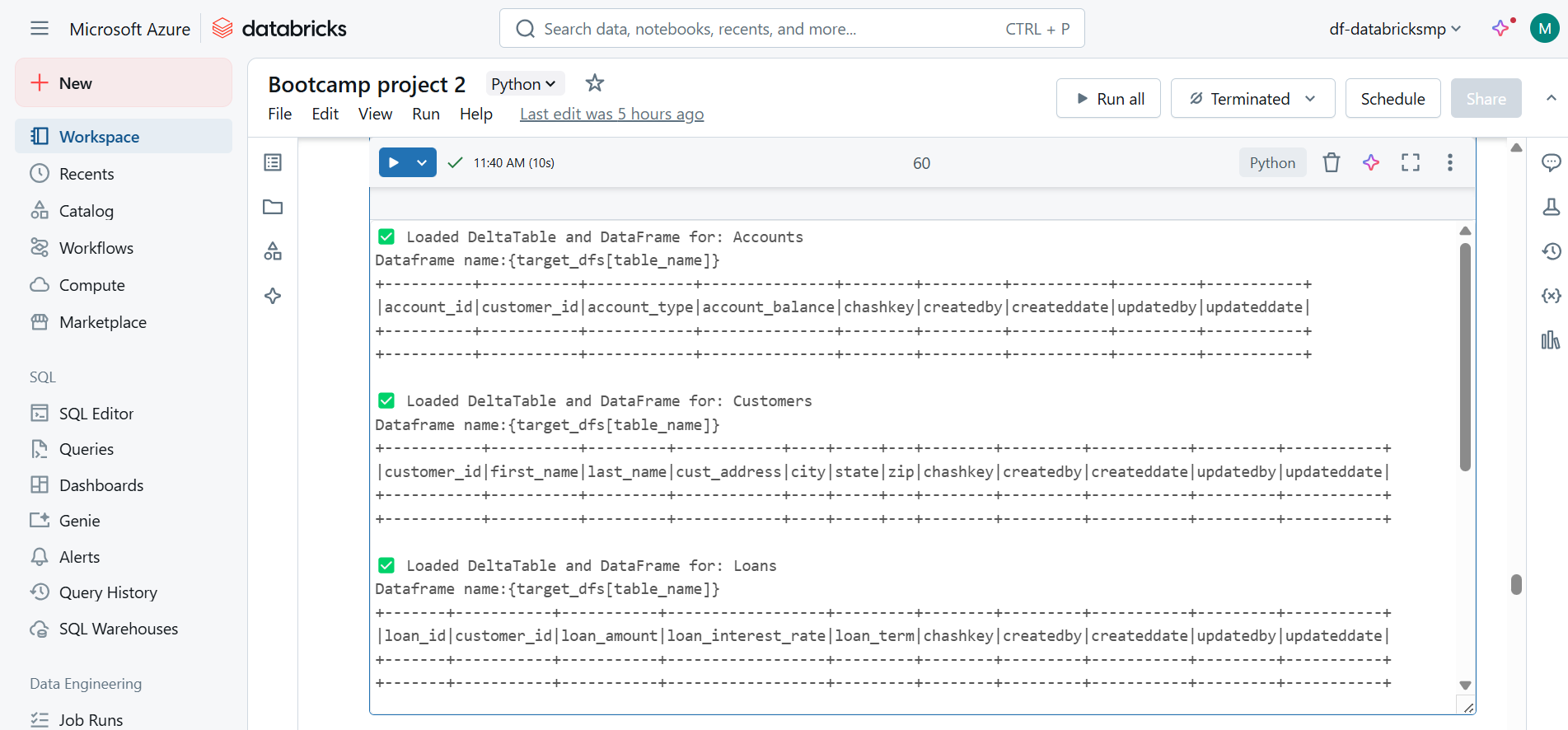


Using the below code, create dataframe using spark.read.format for 5 files and save it in source\_dfs[file\_name].

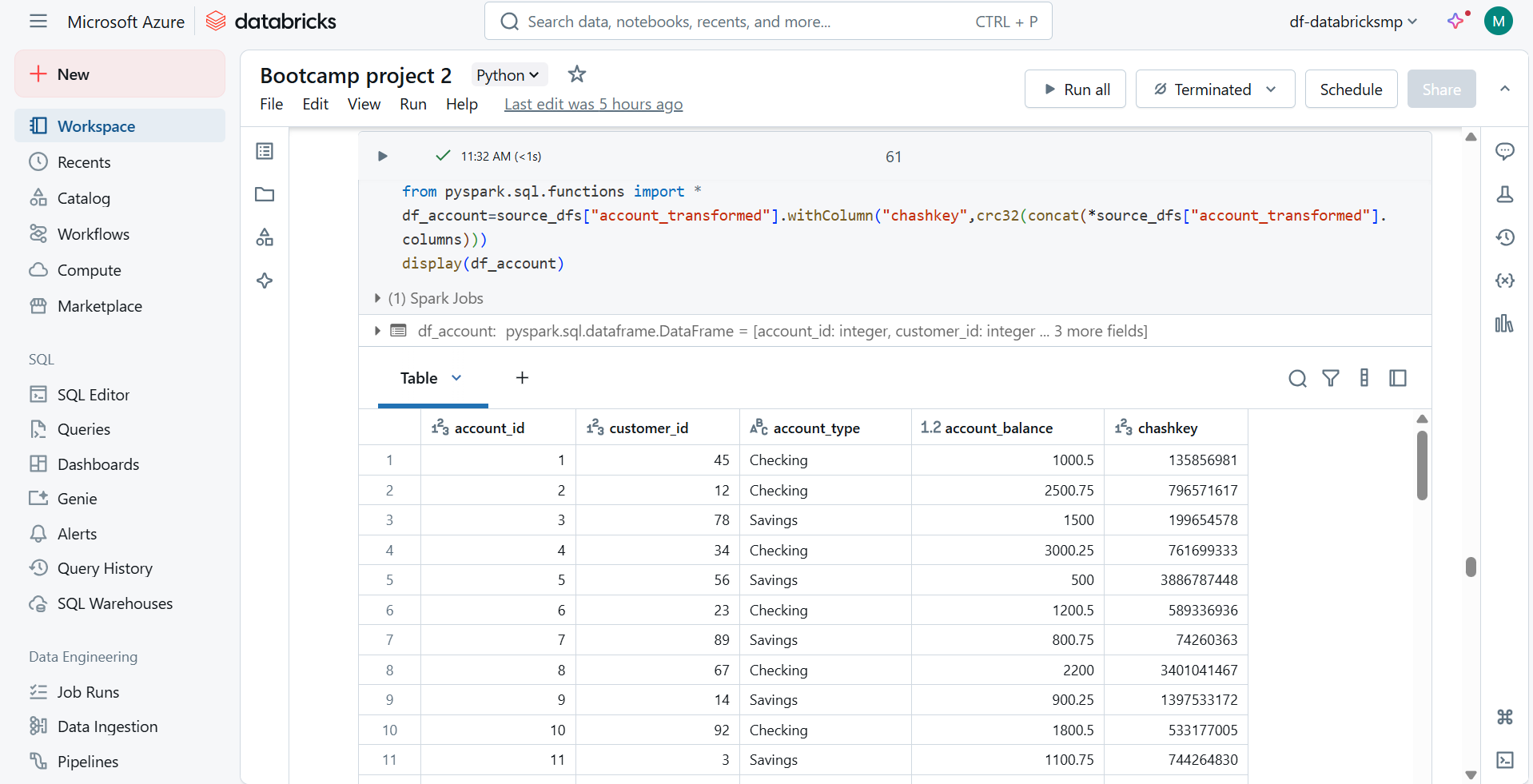


Create delta tables for 5 target tables.

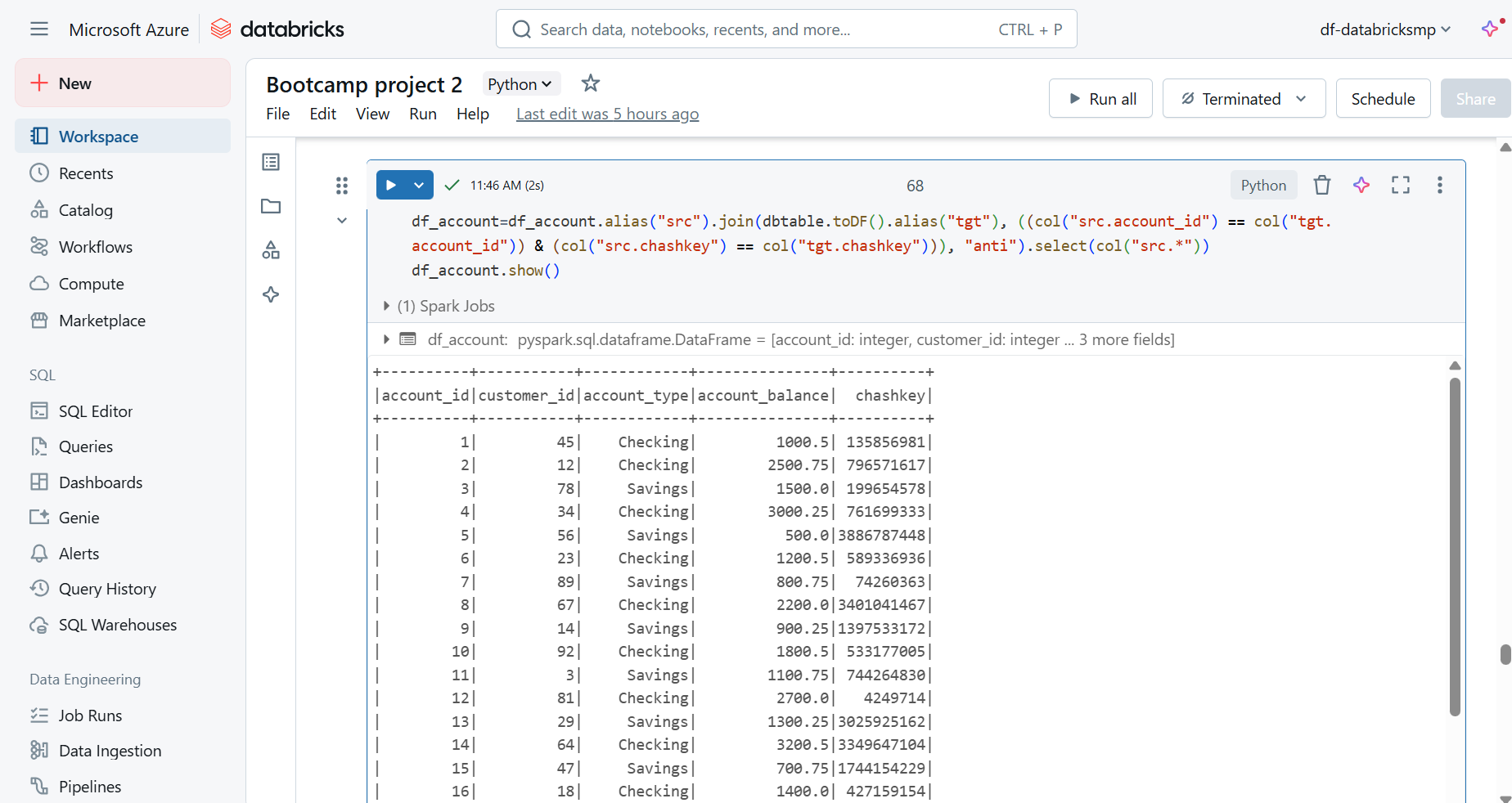




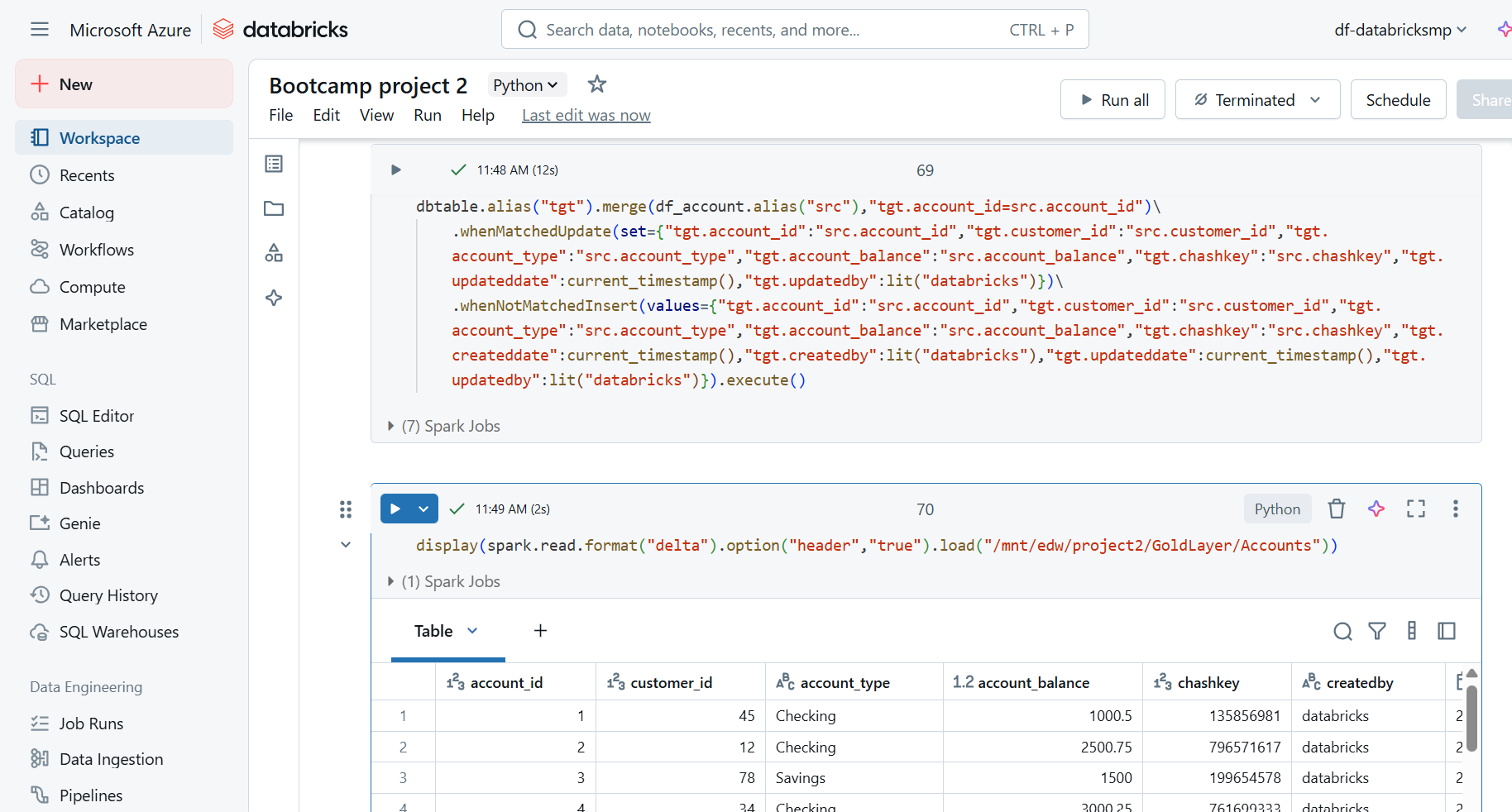
Create hashkey using crc32 function and concat() with all columns

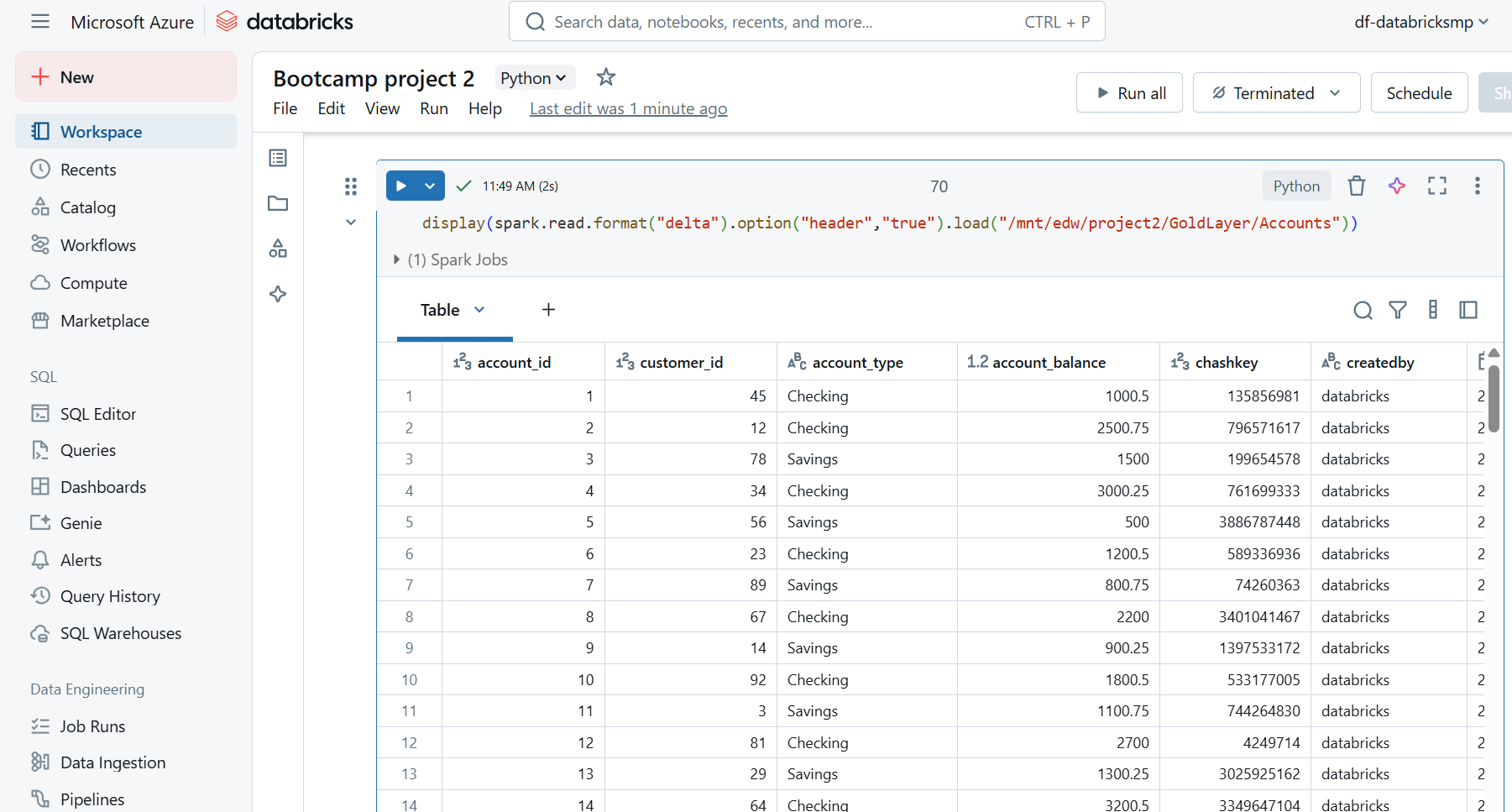


Join source with target using joins with account\_id and hashkey

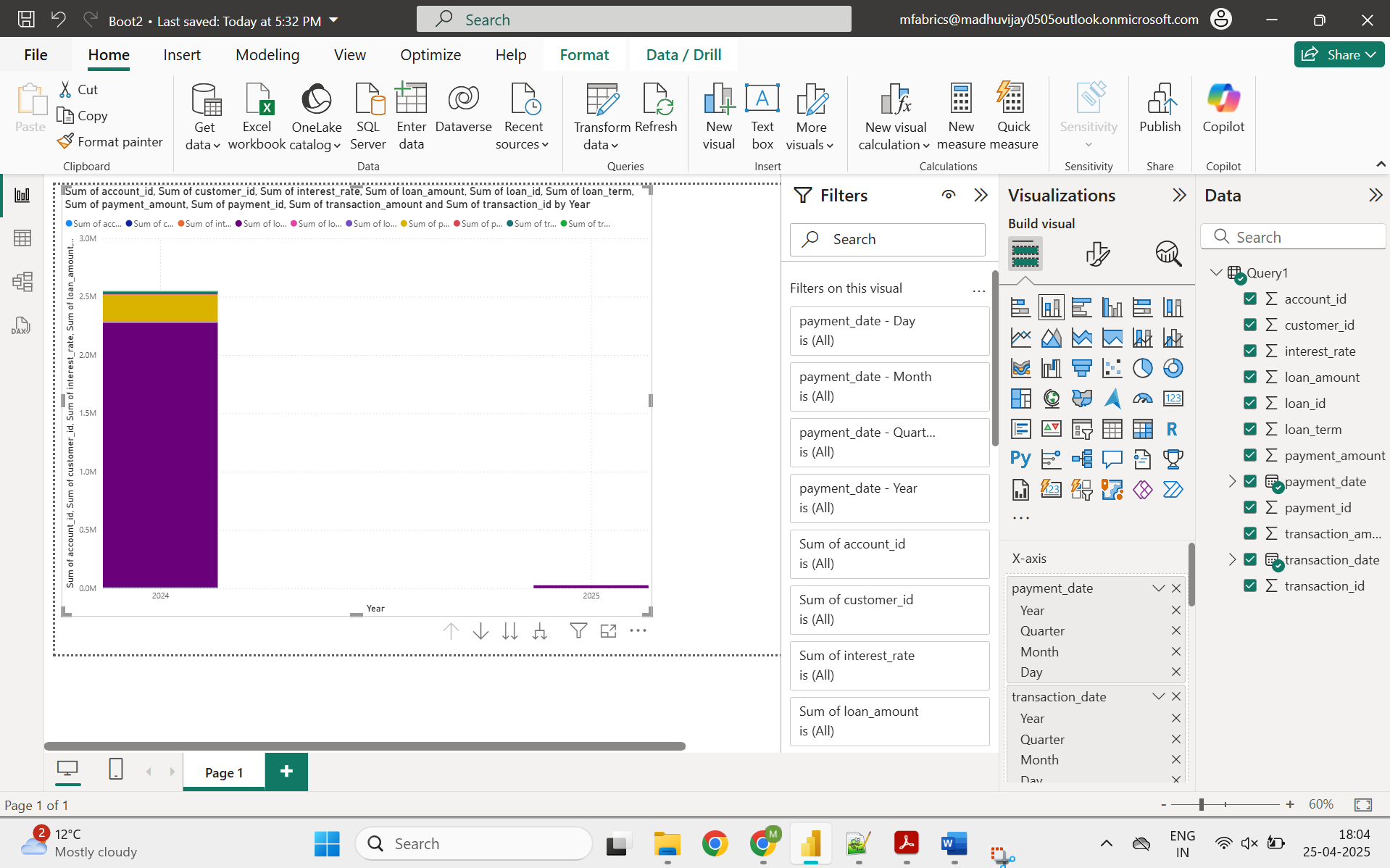


Insert or update data using the below code. Save the file in delta format.





# **Step 4-Use PowerBI for data visualization**



Publish the report to fabrics.

