

BOOTCAMP PROJECT 2

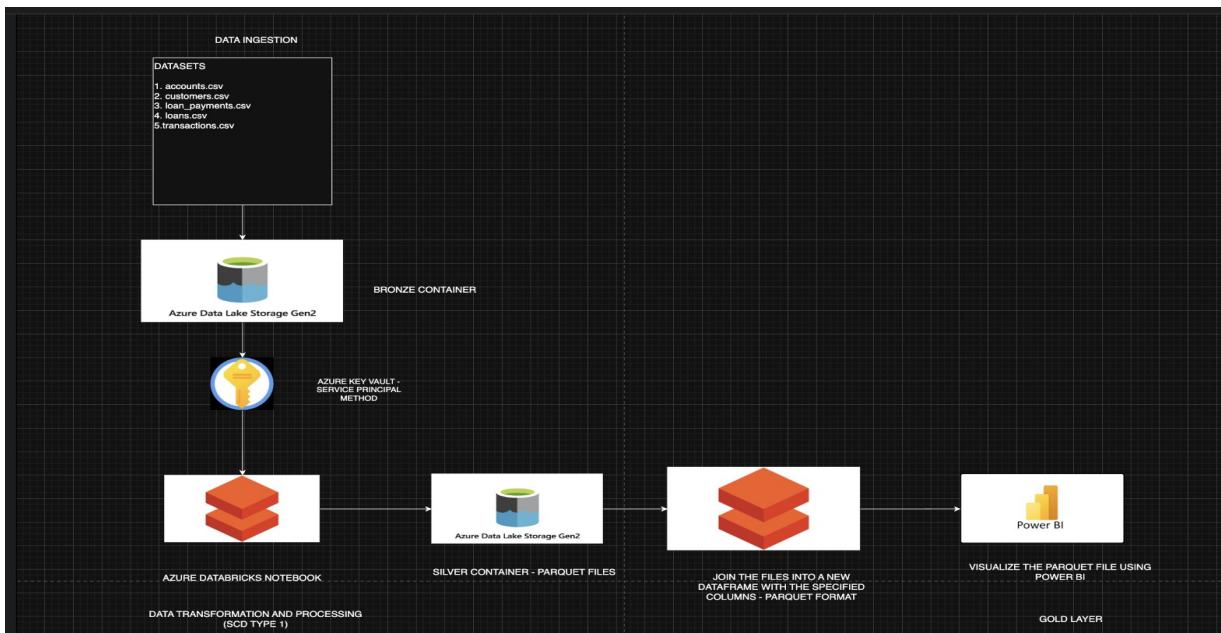
TRANSACTIONS AND LOAN DATA – DATABRICKS

SUBMITTED BY – DRUSYA SURESH

OBJECTIVE

This project aims in building a robust and efficient data processing of a financial data using Databricks notebook. This includes loading data into the ADLS storage container (Bronze layer) and transforming the data in the silver layer using Data bricks notebooks and Gold layer storage storage into SCD Type 1 Delta tables in ADLS gen2. The project also includes a visualization of the delta table using power BI and Fabric. The project aims to ensure an efficient, accurate and scalable data processing to support downstream analytics and reporting needs.

ARCHITECTURE DIAGRAM



STEPS INVOLVED:

- Step 1: Load the data into the Bronze container in the ADLS Gen2 storage account (raw data).
- Step 2: Using Data bricks notebooks connect the ADLS storage account using service principal method.
- Step 3: Read the data from the files, perform necessary transformations on the data (cleaning data), load them as Parquet files in the ADLS storage account Silver container. Also create a new Parquet file with specified columns using join condition and store it in ADLS.
- Step 4: Perform SCD Type 1 logic on the 5 Parquet files.
- Step 5: Schedule the workspace in Data bricks to run on a specified time.
- Step 6: Visualize the joined file using power BI and publish the report in Fabric.

STEP 1:

Bronze layer – Loaded the raw csv files into the ADLS storage account directly using the upload files option successfully under the folder project 2/Bronze.

Microsoft Azure

Search resources, services, and docs (G+/)

Copilot

danielvs97@outlook.com DEFAULT DIRECTORY

Home > adlsdrusya | Containers >

project2 Container

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Overview Diagnose and solve problems Access Control (IAM) Settings

Authentication method: Access key (Switch to Microsoft Entra user account)
Location: project2 / Bronze

Search blobs by prefix (case-sensitive)

| Name | Modified | Access tier | Archive status | Blob type | Size | Lease state |
|-------------------|-----------------------|----------------|----------------|------------|----------|-------------|
| [...] | | | | | | ... |
| accounts.csv | 4/23/2025, 6:14:37 PM | Hot (Inferred) | | Block blob | 2.28 KiB | Available |
| customers.csv | 4/23/2025, 6:14:51 PM | Hot (Inferred) | | Block blob | 4.5 KiB | Available |
| loan_payments.csv | 4/23/2025, 6:15:06 PM | Hot (Inferred) | | Block blob | 2.55 KiB | Available |
| loans.csv | 4/23/2025, 6:15:19 PM | Hot (Inferred) | | Block blob | 2.29 KiB | Available |
| transactions.csv | 4/23/2025, 6:15:34 PM | Hot (Inferred) | | Block blob | 3.43 KiB | Available |

Add or remove favorites by pressing Cmd+Shift+F

STEP 2:

Connecting ADLS with notebook using service principal method.

- Go to the microsoft entra ID and click on app registrations and create a new registration.

Microsoft Azure

Home > Default Directory

Default Directory | App registrations

Owned applications

This account isn't listed as an owner of any applications in this directory.

Add or remove favorites by pressing **Cmd + Shift + F**

- Overview
- Preview features
- Diagnose and solve problems
- Manage
 - Users
 - Groups
 - External Identities
 - Roles and administrators
 - Administrative units
 - Delegated admin partners
 - Enterprise applications
 - Devices
 - App registrations** (selected)
 - Identity Governance
 - Application proxy
 - Custom security attributes
 - Licenses
 - Cross-tenant synchronization

Microsoft Azure

Home > Default Directory | App registrations >

Register an application

* Name
The user-facing display name for this application (this can be changed later).

spn_databricks

Supported account types

Who can use this application or access this API?

Accounts in this organizational directory only (Default Directory only - Single tenant)
 Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
 Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
 Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform: e.g. https://example.com/auth

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

By proceeding, you agree to the Microsoft Platform Policies

Register

- Now click on the created app registration and in the following page copy the application ID and directory ID from there and paste it somewhere.

Microsoft Azure Upgrade Search resources, services, and docs (G+) Copilot Home > Default Directory | App registrations >

spn_databricks

Overview Delete Endpoints Preview features

Essentials

| | | | |
|-------------------------|--|----------------------------|---|
| Display name | : spn_databricks | Client credentials | : Add a certificate or secret |
| Application (client) ID | : 0f4b7ba6-3684-4977-9e65-adae446ceaa7 | Redirect URIs | : Add a Redirect URI |
| Object ID | : da1f8c71-4535-434b-bef2-899fad0753f2 | Application ID URI | : Add an Application ID URI |
| Directory (tenant) ID | : e9477337-8a82-46d5-adc2-b7f6fcdb858d | Managed application in ... | : spn_databricks |
| Supported account types | : My organization only | | |

Get Started Documentation

Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. [Learn more](#)

Add or remove favorites by pressing Cmd+Shift+F

Microsoft Azure Upgrade Search resources, services, and docs (G+) Copilot Home > Default Directory | App registrations >

spn_databricks

Overview Delete Endpoints Preview features

Essentials

| | | | |
|-------------------------|--|----------------------------|---|
| Display name | : spn_databricks | Client credentials | : Add a certificate or secret |
| Application (client) ID | : 0f4b7ba6-3684-4977-9e65-adae446ceaa7 | Redirect URIs | : Add a Redirect URI |
| Object ID | : da1f8c71-4535-434b-bef2-899fad0753f2 | Application ID URI | : Add an Application ID URI |
| Directory (tenant) ID | : e9477337-8a82-46d5-adc2-b7f6fcdb858d | Managed application in ... | : spn_databricks |
| Supported account types | : My organization only | | |

Get Started Documentation

Build your application with the Microsoft identity platform

The Microsoft identity platform is an authentication service, open-source libraries, and application management tools. You can create modern, standards-based authentication solutions, access and protect APIs, and add sign-in for your users and customers. [Learn more](#)

Add or remove favorites by pressing Cmd+Shift+F

- Click on the manage tab in the app registered and click on certificates and secrets tab, now create a new secret and copy paste the secret value somewhere.

The screenshot shows the Azure portal interface for managing app registrations. The left sidebar lists various management options, and the main content area shows the 'Certificates & secrets' tab selected. A modal window titled 'Add a client secret' is open, prompting for a description ('dbconn'), an expiration date ('Custom, 04/23/2025'), and an end date ('04/30/2025').

The screenshot shows the 'Certificates & secrets' section after adding a client secret. The 'Client secrets' tab is now active, showing the single entry 'dbconn' with its details: Description 'dbconn', Expires '4/30/2025', Value 'Z8Q~M6~gaPKEAMRVK1cTKAyr_KPrIHL...', and Secret ID '6f244c30-3477-4e6a-b3d6-b8712dbc9e5'. The 'Copied' status is visible next to the secret ID.

- Now go to the ADLS storage account and under the access control add a new role assignment as storage blob data contributor role to the app registration we created using the select members option.

Microsoft Azure Upgrade Search resources, services, and docs (G+)

Home > adlsdrusya | Access Control (IAM) > Add role assignment ...

Role Members* Conditions Review + assign

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. [Learn more](#)

Job function roles Privileged administrator roles

Grant access to Azure resources based on job function, such as the ability to create virtual machines.

| Name ↑↓ | Description ↑↓ | Type ↑↓ | Category ↑↓ | Details |
|------------------------------------|---|-------------|-------------|----------------------|
| Defender CSPM Storage Data Scanner | Grants access to read blobs and files. This role is used by the data scanner of Defender CSPM. | BuiltInRole | None | View |
| Defender for Storage Data Scanner | Grants access to read blobs and update index tags. This role is used by the data scanner of Defender for Storage. | BuiltInRole | None | View |
| Storage Blob Data Contributor | Allows for read, write and delete access to Azure Storage blob containers and data | BuiltInRole | Storage | View |
| Storage Blob Data Owner | Allows for full access to Azure Storage blob containers and data, including assigning POSIX access control. | BuiltInRole | Storage | View |
| Storage Blob Data Reader | Allows for read access to Azure Storage blob containers and data | BuiltInRole | Storage | View |
| Storage Blob Delegator | Allows for generation of a user delegation key which can be used to sign SAS tokens | BuiltInRole | Storage | View |

Showing 1 - 6 of 6 results.

[Review + assign](#) [Previous](#) [Next](#) [Feedback](#)

Microsoft Azure Upgrade Search resources, services, and docs (G+)

Home > adlsdrusya | Access Control (IAM) > Add role assignment ...

Role Members* Conditions Review + assign

Selected role Storage Blob Data Contributor

Assign access to User, group, or service principal Managed identity

Members [+ Select members](#)

| Name | Object ID | Type |
|---------------------|-----------|------|
| No members selected | | |

Description

[Review + assign](#) [Previous](#) [Next](#) [Feedback](#)

Microsoft Azure Upgrade Search resources, services, and docs (G+/-) Copilot Home > adlsdrusya | Access Control (IAM) > Add role assignment ...

Role **Members*** Conditions Review + assign

Selected role Storage Blob Data Contributor

Assign access to User, group, or service principal Managed identity

Members + Select members

| Name | Object ID | Type |
|---------------------|-----------|------|
| No members selected | | |

Description Optional

Review + assign Previous Next Select Close

Select members

spn

| | |
|----------------------------|---|
| spn_databricks Application | X |
|----------------------------|---|

Selected members:

| | |
|----------------------------|---|
| spn_databricks Application | X |
|----------------------------|---|

Microsoft Azure Upgrade Search resources, services, and docs (G+/-) Copilot Home > adlsdrusya | Access Control (IAM) > Add role assignment ...

Role **Members*** Conditions Review + assign

Selected role Storage Blob Data Contributor

Assign access to User, group, or service principal Managed identity

Members + Select members

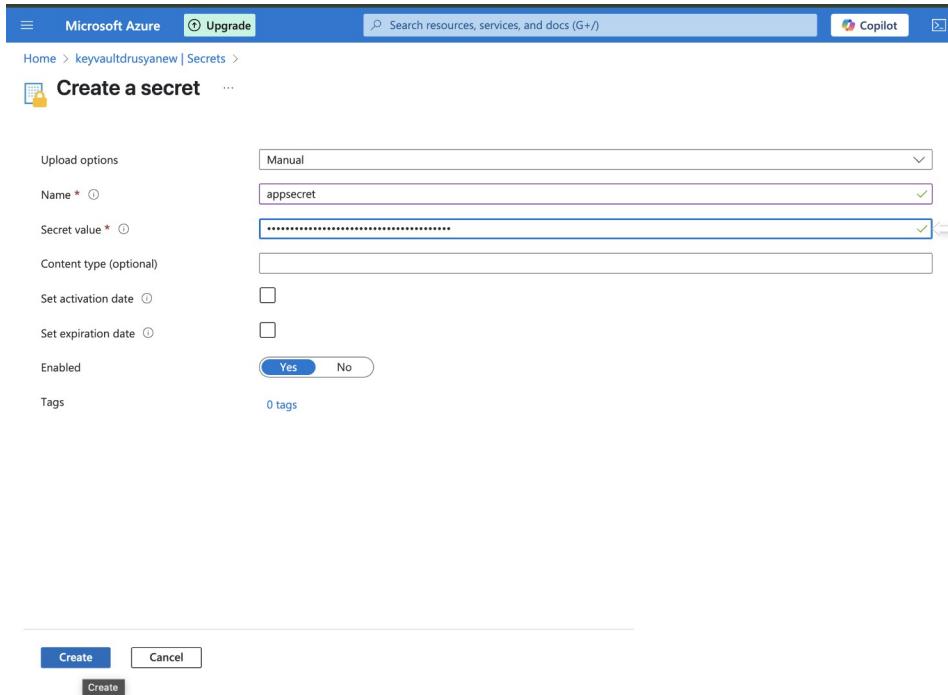
| Name | Object ID | Type |
|----------------|--------------------------------------|------|
| spn_databricks | 291fd23-e7b9-4d5d-8cb6-a657b15fa6... | App |

Description Optional

Review + assign Previous Next

Feedback <https://portal.azure.com/#>

- Now go to the key vault and add two secrets for the app id and secret value.



Microsoft Azure Upgrade Search resources, services, and docs (G+) Copilot Home > keyvaultdrusyanew | Secrets Create a secret ...

Upload options: Manual

Name * ⓘ: appsecret

Secret value * ⓘ:

Content type (optional):

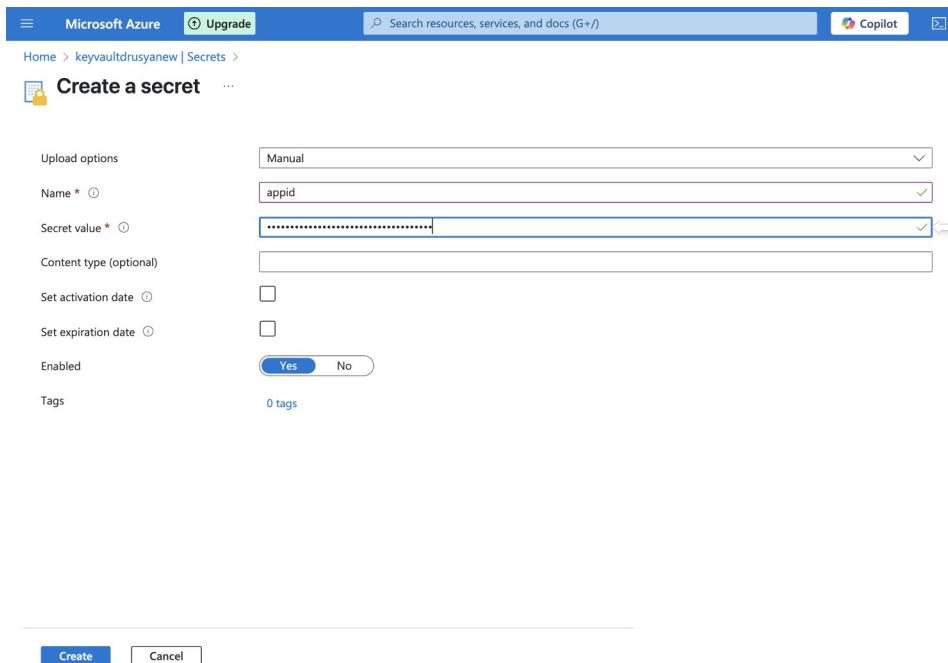
Set activation date:

Set expiration date:

Enabled: Yes No

Tags: 0 tags

Create **Cancel** **Create**



Microsoft Azure Upgrade Search resources, services, and docs (G+) Copilot Home > keyvaultdrusyanew | Secrets Create a secret ...

Upload options: Manual

Name * ⓘ: appid

Secret value * ⓘ:

Content type (optional):

Set activation date:

Set expiration date:

Enabled: Yes No

Tags: 0 tags

Create **Cancel**

The screenshot shows the Microsoft Azure Key Vault interface for the 'keyvaultdrusyanew' resource. On the left, a sidebar lists various management options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Access policies, Resource visualizer, Events, Objects, Keys, Secrets (which is selected), Certificates, Settings, Access configuration, Networking, Microsoft Defender for Cloud, Properties, and Locks. The main content area displays a table of secrets:

| Name | Type | Status | Expiration date |
|-----------------|------|-----------|-----------------|
| appid | | ✓ Enabled | |
| appsecret | | ✓ Enabled | |
| adlscredentials | | ✓ Enabled | |

A success message in the top right corner states: "Creating the secret 'appid'. The secret 'appid' has been successfully created."

- Now open Data bricks note book and create a new scope for the key vault using key vault id and resource id.

The screenshot shows the Databricks Notebook interface. The left sidebar contains a navigation menu with items such as Workspace, Recents, Catalog, Workflows, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. The main workspace is titled "Create Secret Scope" and includes fields for "Scope Name" (set to "adlconnection"), "Manage Principal" (set to "Creator"), "Azure Key Vault" (with "DNS Name" set to "https://keyvaultdrusyanew.vault.azure.net/" and "Resource ID" set to "/subscriptions/f0e0eb4f-a921-454a-8f99-913052c1f104/resourceGroups/Drusyanew"). There are "Cancel" and "Create" buttons at the top of the form.

- Now connect the ADLS with note book using the code below and paste the values of the respective ids as shown in the screenshot and run the command it will connect ADLS with the notebook.

Listing the scopes

```
dbutils.secrets.listScopes()
```

```
[SecretScope(name='adlsconnection')]
```

Connecting ADLS with notebook using service principal

```
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="adlsconnection", key="appid"),
           "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="adlsconnection", key="appsecret")}
```

Connecting ADLS with notebook using service principal

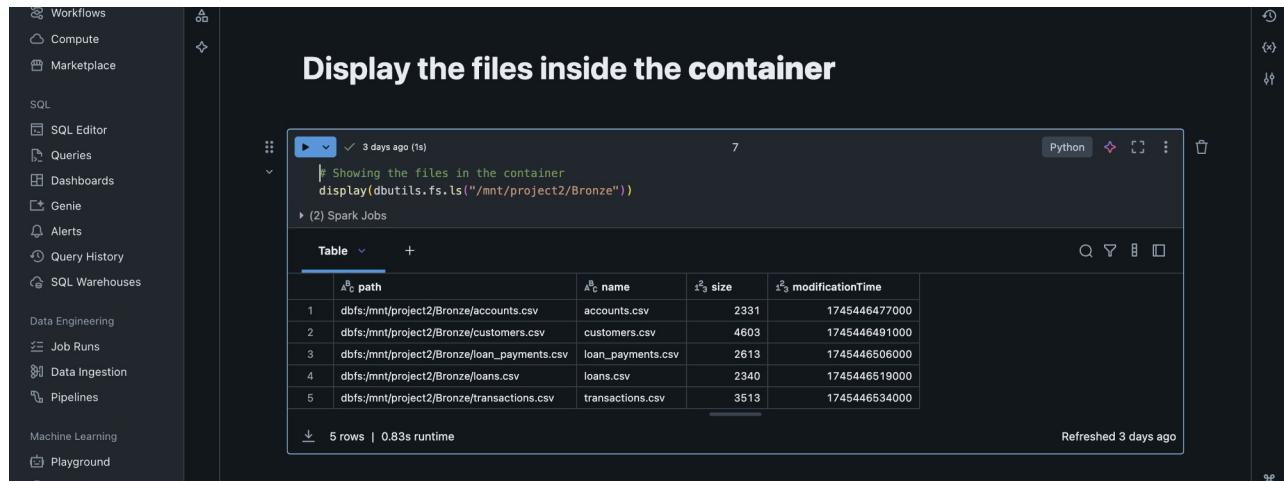
```
configs = {"fs.azure.account.auth.type": "OAuth",
           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
           "fs.azure.account.oauth2.client.id": dbutils.secrets.get(scope="adlsconnection", key="appid"),
           "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="adlsconnection", key="appsecret"),
           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/e9477337-8a82-46d5-adc2-b7f6fcdb858d/oauth2/token"}
```

```
dbutils.fs.mount(
    source = "abfss://project2@adlsdrusya.dfs.core.windows.net/",
    mount_point = "/mnt/project2",
    extra_configs = configs)
```

Display the files inside the container

STEP 3: READING THE FILES AND CLEANING

- Open the files from the notebooks and loading into a data frame using the read command.



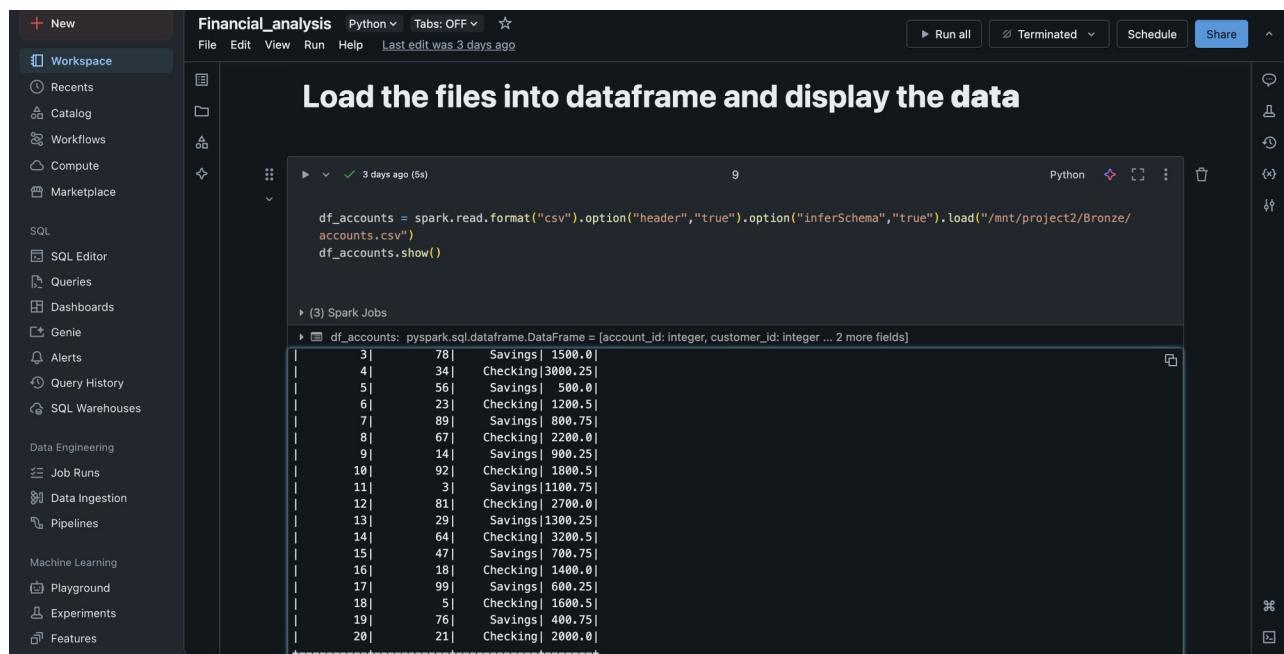
The screenshot shows a Jupyter Notebook interface with a sidebar containing various project and system navigation links. The main area displays a code cell with the following Python code:

```
# Showing the files in the container
display(dbutils.fs.ls("/mnt/project2/Bronze"))
```

Below the code cell is a table showing the results of the file listing operation:

| | path | name | size | modificationTime |
|---|---|-------------------|------|------------------|
| 1 | dbfs:/mnt/project2/Bronze/accounts.csv | accounts.csv | 2331 | 1745446477000 |
| 2 | dbfs:/mnt/project2/Bronze/customers.csv | customers.csv | 4603 | 1745446491000 |
| 3 | dbfs:/mnt/project2/Bronze/loan_payments.csv | loan_payments.csv | 2613 | 1745446506000 |
| 4 | dbfs:/mnt/project2/Bronze/loans.csv | loans.csv | 2340 | 1745446519000 |
| 5 | dbfs:/mnt/project2/Bronze/transactions.csv | transactions.csv | 3513 | 1745446534000 |

At the bottom of the table, it says "5 rows | 0.83s runtime". The status bar at the bottom right indicates "Refreshed 3 days ago".



The screenshot shows a Jupyter Notebook interface with a sidebar containing various project and system navigation links. The main area displays a code cell with the following Python code:

```
df_accounts = spark.read.format("csv").option("header","true").option("inferSchema","true").load("/mnt/project2/Bronze/accounts.csv")
df_accounts.show()
```

Below the code cell is a table showing the first few rows of the DataFrame:

| | account_id | customer_id | balance |
|----|------------|-------------|-------------------|
| 1 | 3 | 78 | Savings 1500.0 |
| 2 | 4 | 34 | Checking 3000.25 |
| 3 | 5 | 56 | Savings 500.0 |
| 4 | 6 | 23 | Checking 1200.5 |
| 5 | 7 | 89 | Savings 800.75 |
| 6 | 8 | 67 | Checking 2200.0 |
| 7 | 9 | 14 | Savings 900.25 |
| 8 | 10 | 92 | Checking 1800.5 |
| 9 | 11 | 3 | Savings 1100.75 |
| 10 | 12 | 81 | Checking 2700.0 |
| 11 | 13 | 29 | Savings 1300.25 |
| 12 | 14 | 64 | Checking 3200.5 |
| 13 | 15 | 47 | Savings 700.75 |
| 14 | 16 | 18 | Checking 1400.0 |
| 15 | 17 | 99 | Savings 600.25 |
| 16 | 18 | 5 | Checking 1600.5 |
| 17 | 19 | 76 | Savings 400.75 |
| 18 | 20 | 21 | Checking 2000.0 |

Microsoft Azure | drusya-databricks

Financial_analysis Python Tabs: OFF ⚡

File Edit View Run Help Last edit was 3 days ago

Run all Terminated Schedule Share

3 days ago (2s) 10 Python

```
df_customers = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/mnt/project2/Bronze/customers.csv")
display(df_accounts)
```

(3) Spark Jobs

df_customers: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 5 more fields]

Table +

| account_id | customer_id | account_type | balance |
|------------|-------------|--------------|---------|
| 2 | 2 | Checking | 2500.75 |
| 3 | 3 | Savings | 1500 |
| 4 | 4 | Checking | 3000.25 |
| 5 | 5 | Savings | 500 |
| 6 | 6 | Checking | 1200.5 |
| 7 | 7 | Savings | 800.75 |
| 8 | 8 | Checking | 2200 |
| 9 | 9 | Savings | 900.25 |
| 10 | 10 | Checking | 1800.5 |
| 11 | 11 | Savings | 1100.75 |
| 12 | 12 | Checking | 2700 |
| 13 | 13 | Savings | 1300.25 |
| 14 | 14 | Checking | 3200.5 |
| 15 | 15 | Savings | 700.75 |
| .. | .. | .. | .. |

100 rows | 1.56s runtime Refreshed 3 days ago

Microsoft Azure | drusya-databricks

Financial_analysis Python Tabs: OFF ⚡

File Edit View Run Help Last edit was 3 days ago

Run all Terminated Schedule Share

3 days ago (2s) 11

```
df_loan_payments = spark.read.format("csv").option("header", "true").option("inferSchema", "true").load("/mnt/project2/Bronze/loan_payments.csv")
display(df_loan_payments)
```

(3) Spark Jobs

df_loan_payments: pyspark.sql.dataframe.DataFrame = [payment_id: integer, loan_id: integer ... 2 more fields]

Table +

| payment_id | loan_id | payment_date | payment_amount | |
|------------|---------|--------------|----------------|-----|
| 1 | 1 | 45 | 2024-01-01 | 100 |
| 2 | 2 | 23 | 2024-01-02 | 150 |
| 3 | 3 | 67 | 2024-01-03 | 200 |
| 4 | 4 | 89 | 2024-01-04 | 250 |
| 5 | 5 | 12 | 2024-01-05 | 300 |
| 6 | 6 | 34 | 2024-01-06 | 350 |
| 7 | 7 | 56 | 2024-01-07 | 400 |
| 8 | 8 | 78 | 2024-01-08 | 450 |
| 9 | 9 | 90 | 2024-01-09 | 500 |
| 10 | 10 | 11 | 2024-01-10 | 550 |
| 11 | 11 | 22 | 2024-01-11 | 600 |
| 12 | 12 | 33 | 2024-01-12 | 650 |
| 13 | 13 | 44 | 2024-01-13 | 700 |
| 14 | 14 | 55 | 2024-01-14 | 750 |
| 15 | 15 | 66 | 2024-01-15 | 800 |
| .. | .. | .. | .. | .. |

100 rows | 1.67s runtime Refreshed 3 days ago

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF ⭐

File Edit View Run Help Last edit was 3 days ago

Run all Terminated Schedule Share

Table

```

3 days ago (1s) 12
df_loans = spark.read.format("csv").option("header","true").option("inferSchema","true").load("/mnt/project2/Bronze/loans.csv")
display(df_loans)

(3) Spark Jobs
df_loans: pyspark.sql.DataFrameType [loan_id: integer, customer_id: integer ... 3 more fields]

```

| loan_id | customer_id | loan_amount | interest_rate | loan_term |
|---------|-------------|-------------|---------------|-----------|
| 1 | 1 | 45 | 10000.5 | 5.5 |
| 2 | 2 | 12 | 20000.75 | 4.5 |
| 3 | 3 | 78 | 15000 | 6 |
| 4 | 4 | 34 | 30000.25 | 3.5 |
| 5 | 5 | 56 | 25000 | 5 |
| 6 | 6 | 23 | 17500.5 | 4 |
| 7 | 7 | 89 | 22500.75 | 6.5 |
| 8 | 8 | 67 | 27500 | 3 |
| 9 | 9 | 14 | 32500.25 | 5.5 |
| 10 | 10 | 92 | 37500.5 | 4.5 |
| 11 | 11 | 3 | 10000.75 | 6 |
| 12 | 12 | 81 | 20000 | 3.5 |
| 13 | 13 | 29 | 15000.25 | 5 |
| 14 | 14 | 64 | 30000.5 | 4 |
| 15 | 15 | 47 | 25000.75 | 6.5 |

100 rows | 1.27s runtime

Refreshed 3 days ago

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF ⭐

File Edit View Run Help Last edit was 3 days ago

Run all Terminated Schedule Share

Table

```

3 days ago (1s) 13
df_transactions = spark.read.format("csv").option("header","true").option("inferSchema","true").load("/mnt/project2/Bronze/transactions.csv")
display(df_transactions)

(3) Spark Jobs
df_transactions: pyspark.sql.DataFrameType [transaction_id: integer, account_id: integer ... 3 more fields]

```

| transaction_id | account_id | transaction_date | transaction_amount | transaction_type |
|----------------|------------|------------------|--------------------|-------------------|
| 1 | 1 | 45 | 2024-01-01 | 100.5 Deposit |
| 2 | 2 | 12 | 2024-01-02 | 200.75 Withdrawal |
| 3 | 3 | 78 | 2024-01-03 | 150 Deposit |
| 4 | 4 | 34 | 2024-01-04 | 300.25 Withdrawal |
| 5 | 5 | 56 | 2024-01-05 | 250 Deposit |
| 6 | 6 | 23 | 2024-01-06 | 175 Withdrawal |
| 7 | 7 | 89 | 2024-01-07 | 225.5 Deposit |
| 8 | 8 | 67 | 2024-01-08 | 275.75 Withdrawal |
| 9 | 9 | 14 | 2024-01-09 | 325 Deposit |
| 10 | 10 | 92 | 2024-01-10 | 375.25 Withdrawal |
| 11 | 11 | 3 | 2024-01-11 | 100.5 Deposit |
| 12 | 12 | 81 | 2024-01-12 | 200.75 Withdrawal |
| 13 | 13 | 29 | 2024-01-13 | 150 Deposit |
| 14 | 14 | 64 | 2024-01-14 | 300.25 Withdrawal |
| 15 | 15 | 47 | 2024-01-15 | 250 Deposit |

100 rows | 1.18s runtime

Refreshed 3 days ago

- Data cleaning.

1. Removing duplicates from the files.

```

# Removing duplicate records from the files.
df_accounts = df_accounts.dropDuplicates()
df_customers = df_customers.dropDuplicates()
df_loan_payments = df_loan_payments.dropDuplicates()
df_loans = df_loans.dropDuplicates()
df_transactions = df_transactions.dropDuplicates()

```

2. Removing null values if there are any null values.

```

#Removing null values from the files.
df_accounts = df_accounts.na.drop()
df_customers = df_customers.na.drop()
df_loan_payments = df_loan_payments.na.drop()
df_loans = df_loans.na.drop()
df_transactions = df_transactions.na.drop()

```

3. Type casting the columns with specific values.

The screenshot shows a Databricks notebook titled "Financial_analysis" in Python mode. The code block contains several lines of Python code using the pandas API to cast specific columns to integers or doubles. The code is as follows:

```

# Type cast the column values with specific values.
df_accounts = df_accounts.withColumn("account_id", df_accounts["account_id"].cast("int"))
df_accounts = df_accounts.withColumn("customer_id", df_accounts["customer_id"].cast("int"))
df_accounts = df_accounts.withColumn("balance", df_accounts["balance"].cast("double"))
df_customers = df_customers.withColumn("customer_id", df_customers["customer_id"].cast("int"))
df_loan_payments = df_loan_payments.withColumn("payment_id", df_loan_payments["payment_id"].cast("int"))
df_loan_payments = df_loan_payments.withColumn("loan_id", df_loan_payments["loan_id"].cast("int"))
df_loan_payments = df_loan_payments.withColumn("payment_amount", df_loan_payments["payment_amount"].cast("double"))
df_loans = df_loans.withColumn("loan_id", df_loans["loan_id"].cast("int"))
df_loans = df_loans.withColumn("customer_id", df_loans["customer_id"].cast("int"))
df_loans = df_loans.withColumn("loan_amount", df_loans["loan_amount"].cast("double"))
df_loans = df_loans.withColumn("interest_rate", df_loans["interest_rate"].cast("int"))
df_loans = df_loans.withColumn("loan_term", df_loans["loan_term"].cast("double"))
df_transactions = df_transactions.withColumn("transaction_id", df_transactions["transaction_id"].cast("int"))
df_transactions = df_transactions.withColumn("account_id", df_transactions["account_id"].cast("int"))
df_transactions = df_transactions.withColumn("transaction_amount", df_transactions["transaction_amount"].cast("double"))
df_transactions = df_transactions.withColumn("transaction_date", df_transactions["transaction_date"].cast("date"))

```

The sidebar on the left shows various Databricks workspace sections like Recents, Catalog, Workflows, Compute, Marketplace, SQL, and Data Engineering.

- Loading the data into ADLS in Silver container as Parquet format.

The screenshot shows the same Databricks notebook "Financial_analysis". The code block now includes a section for writing the data to the Silver container in Parquet format. The code is as follows:

```

# Load the data into Silver container as Parquet file format
df_accounts.write.mode("overwrite").format("parquet").save("/mnt/project2/Silver/accounts")
df_customers.write.mode("overwrite").format("parquet").save("/mnt/project2/Silver/customers")
df_loan_payments.write.mode("overwrite").format("parquet").save("/mnt/project2/Silver/loan_payments")
df_loans.write.mode("overwrite").format("parquet").save("/mnt/project2/Silver/loans")
df_transactions.write.mode("overwrite").format("parquet").save("/mnt/project2/Silver/transactions")

```

The sidebar on the left remains the same as the previous screenshot.

Microsoft Azure

Search resources, services, and docs (G+/)

Copilot

danielvs97@outlook.com

DEFAULT DIRECTORY

Home > adlsdrusya | Containers >

project2 ...

Container

Search

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: project2 / Silver

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name | Modified | Access tier | Archive status | Blob type | Size | Lease state |
|---------------|-----------------------|-------------|----------------|-----------|------|-------------|
| [...] | | | | | | ... |
| accounts | 4/26/2025, 4:20:51 PM | | | | - | ... |
| customers | 4/26/2025, 4:20:52 PM | | | | - | ... |
| loan_payments | 4/26/2025, 4:20:54 PM | | | | - | ... |
| loans | 4/26/2025, 4:20:55 PM | | | | - | ... |
| transactions | 4/26/2025, 4:20:56 PM | | | | - | ... |

Add or remove favorites by pressing Cmd+Shift+F

- Join the data frames together and create a new data frame with data including numeric values and dates only from the joined data frame and save it as a Parquet file in the silver container.

Microsoft Azure |  databricks

Search data, notebooks, recents, and more... drusya-databricks

Financial_analysis Python Tabs: OFF ⚡ Last edit was 1 minute ago

File Edit View Run Help

Run all PROJECT_CLUSTER Schedule Share

Create a new Parquet file with some specified columns using the join function.

```

04:42 PM (x1s) 23
df_alljoined = df_accounts \
    .join(df_customers, on="customer_id", how="left") \
    .join(df_loans, on="customer_id", how="left") \
    .join(df_loan_payments, on="loan_id", how="left") \
    .join(df_transactions, on="account_id", how="left")

```

df_alljoined: pyspark.sql.dataframe.DataFrame = [account_id: integer, loan_id: integer ... 19 more fields]

```

2 minutes ago (<1s) 24
from pyspark.sql.functions import col

df_numeric_filtered = df_alljoined.select(
    col("account_id").cast("string"),
    col("customer_id").cast("string"),
    col("loan_id").cast("string"),
    col("payment_id").cast("string"),
    col("transaction_id").cast("string"),
    col("balance").cast("double"),
    col("loan_amount").cast("double"),
    col("interest_rate").cast("double"),
    col("loan_term").cast("int"),
    col("payment_amount").cast("double"),
    cast(payment_date, "date"),
    cast(transaction_amount, "double"),
    cast(transaction_date, "date")
)

```

Microsoft Azure |  databricks

Search data, notebooks, recents, and more... drusya-databricks

Financial_analysis Python Tabs: OFF ⚡ Last edit was 2 minutes ago

File Edit View Run Help

Run all PROJECT_CLUSTER Schedule Share

df_alljoined: pyspark.sql.dataframe.DataFrame = [account_id: integer, loan_id: integer ... 19 more fields]

```

2 minutes ago (<1s) 24
from pyspark.sql.functions import col

df_numeric_filtered = df_alljoined.select(
    col("account_id").cast("string"),
    col("customer_id").cast("string"),
    col("loan_id").cast("string"),
    col("payment_id").cast("string"),
    col("transaction_id").cast("string"),
    col("balance").cast("double"),
    col("loan_amount").cast("double"),
    col("interest_rate").cast("double"),
    col("loan_term").cast("int"),
    col("payment_amount").cast("double"),
    col("payment_date").cast("date"),
    col("transaction_amount").cast("double"),
    col("transaction_date").cast("date")
)

```

df_numeric_filtered: pyspark.sql.dataframe.DataFrame = [account_id: string, customer_id: string ... 11 more fields]

```

2 minutes ago (8s) 25
df_numeric_filtered.write.mode("overwrite").parquet("/mnt/project2/Silver/numeric_filtered")

```

(5) Spark Jobs

26

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

project2

Container

Home > adisdrusya | Containers >

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: project2 / Silver

Search blobs by prefix (case-sensitive)

Show deleted objects

| Name | Modified | Access tier | Archive status | Blob type | Size | Lease state |
|------------------|-----------------------|-------------|----------------|-----------|------|-------------|
| accounts | 4/26/2025, 4:20:51 PM | | | | - | *** |
| customers | 4/26/2025, 4:20:52 PM | | | | - | *** |
| loan_payments | 4/26/2025, 4:20:54 PM | | | | - | *** |
| loans | 4/26/2025, 4:20:55 PM | | | | - | *** |
| numeric_filtered | 4/26/2025, 4:46:01 PM | | | | - | *** |
| transactions | 4/26/2025, 4:20:56 PM | | | | - | *** |

Add or remove favorites by pressing Cmd + Shift + F

STEP 4: PERFORM SCD TYPE 1 LOGIC ON 5 PARQUET FILES

- Create a delta table in hive metastore as default storage location.

The screenshot shows the Databricks interface with the 'Financial_analysis' notebook open. The code cell contains a series of CREATE TABLE IF NOT EXISTS statements for various dimensions in a Gold layer. The run was successful, indicated by a green checkmark and the message 'Just now (17s)'. The status bar at the bottom shows 'OK' and a note: 'This result is stored as _sqlpdf and can be used in other Python cells.'

```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.default.dim_accounts
(account_id INT, customer_id INT, account_type STRING, balance DOUBLE,hash_key BIGINT, createdby STRING,createddate
TIMESTAMP,updatedby STRING,updateddate TIMESTAMP)
USING DELTA
LOCATION '/mnt/project2/GOLD/dim_accounts';
CREATE TABLE IF NOT EXISTS hive_metastore.default.dim_customers
(customer_id INT, first_name STRING, last_name STRING, address STRING, city STRING, state STRING, zip STRING,hash_key
BIGINT, createdby STRING,createddate TIMESTAMP,updatedby STRING,updateddate TIMESTAMP)
USING DELTA
LOCATION '/mnt/project2/GOLD/dim_customers';
CREATE TABLE IF NOT EXISTS hive_metastore.default.dim_loans
(loan_id INT, customer_id INT, loan_amount DOUBLE, interest_rate DOUBLE, loan_term INT,hash_key BIGINT, createdby STRING,
createddate TIMESTAMP,updatedby STRING,updateddate TIMESTAMP)
USING DELTA
LOCATION '/mnt/project2/GOLD/dim_loans';
CREATE TABLE IF NOT EXISTS hive_metastore.default.dim_loanpayments
(payment_id INT, loan_id INT, payment_date DATE, payment_amount DOUBLE,hash_key BIGINT, createdby STRING,createddate
TIMESTAMP,updatedby STRING,updateddate TIMESTAMP)
USING DELTA
LOCATION '/mnt/project2/GOLD/dim_loanpayments';
CREATE TABLE IF NOT EXISTS hive_metastore.default.dim_transactions
(transaction_id INT, account_id INT, transaction_date DATE, transaction_amount DOUBLE, transaction_type STRING,hash_key
BIGINT, createdby STRING,createddate TIMESTAMP,updatedby STRING,updateddate TIMESTAMP)
USING DELTA
LOCATION '/mnt/project2/GOLD/dim_transactions';
```

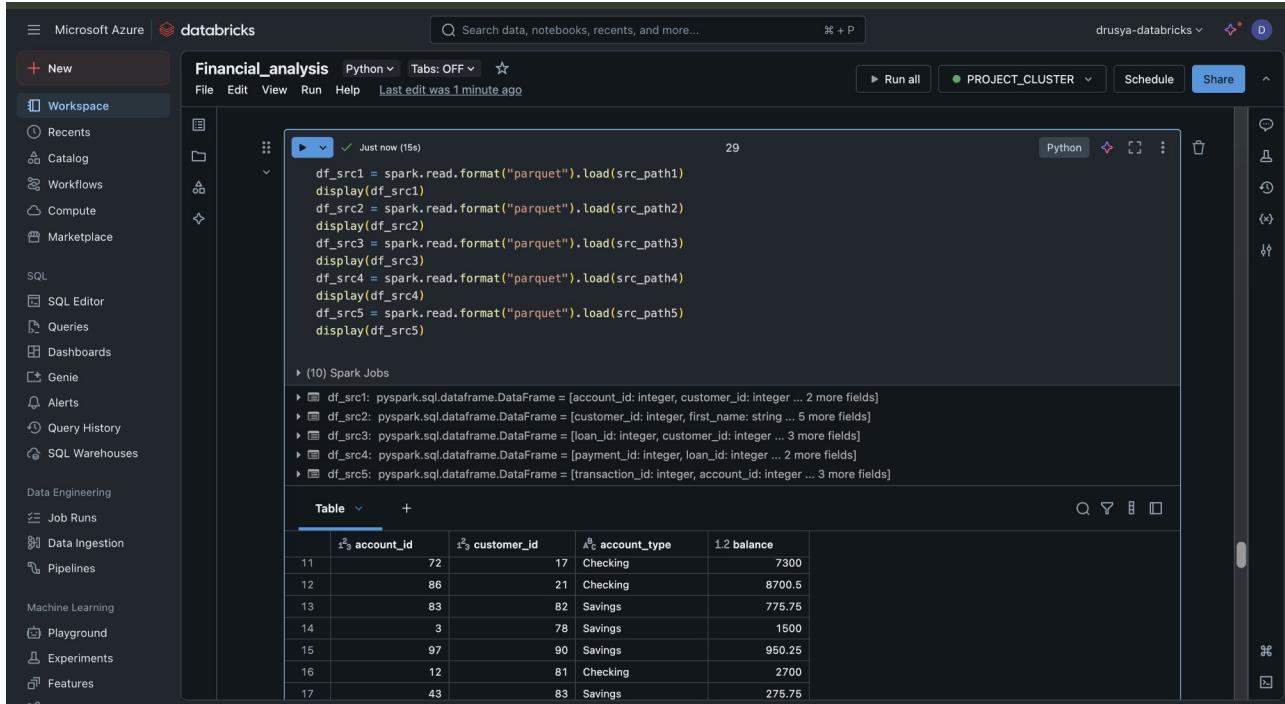
- Now enter a source path and target path

The screenshot shows the Databricks interface with the 'Financial_analysis' notebook open. The code cell contains a Python script that prints source and destination paths for five tables: accounts, customers, loans, loan_payments, and transactions. The run was successful, indicated by a green checkmark and the message 'Just now (3s)'. The output shows the paths for each table.

```
src_path1="/mnt/project2/Silver/accounts"
print(src_path1)
dest_path1="/mnt/project2/Gold/dim_accounts"
print(dest_path1)
src_path2="/mnt/project2/Silver/customers"
print(src_path2)
dest_path2="/mnt/project2/Gold/dim_customers"
print(dest_path2)
src_path3="/mnt/project2/Silver/loans"
print(src_path3)
dest_path3="/mnt/project2/Gold/dim_loans"
print(dest_path3)
src_path4="/mnt/project2/Silver/loan_payments"
print(src_path4)
dest_path4="/mnt/project2/Gold/dim_loan_payments"
print(dest_path4)
src_path5="/mnt/project2/Silver/transactions"
print(src_path5)
dest_path5="/mnt/project2/Gold/dim_transactions"
print(dest_path5)
```

```
/mnt/project2/Silver/accounts
/mnt/project2/Gold/dim_accounts
/mnt/project2/Silver/customers
/mnt/project2/Gold/dim_customers
/mnt/project2/Silver/loans
/mnt/project2/Gold/dim_loans
```

- To display my source file I use the read command.



The screenshot shows a Databricks notebook titled "Financial_analysis" in Python. The code reads five Parquet files ("src1" through "src5") and displays their contents as DataFrames. A preview of the DataFrame is shown below the code.

```

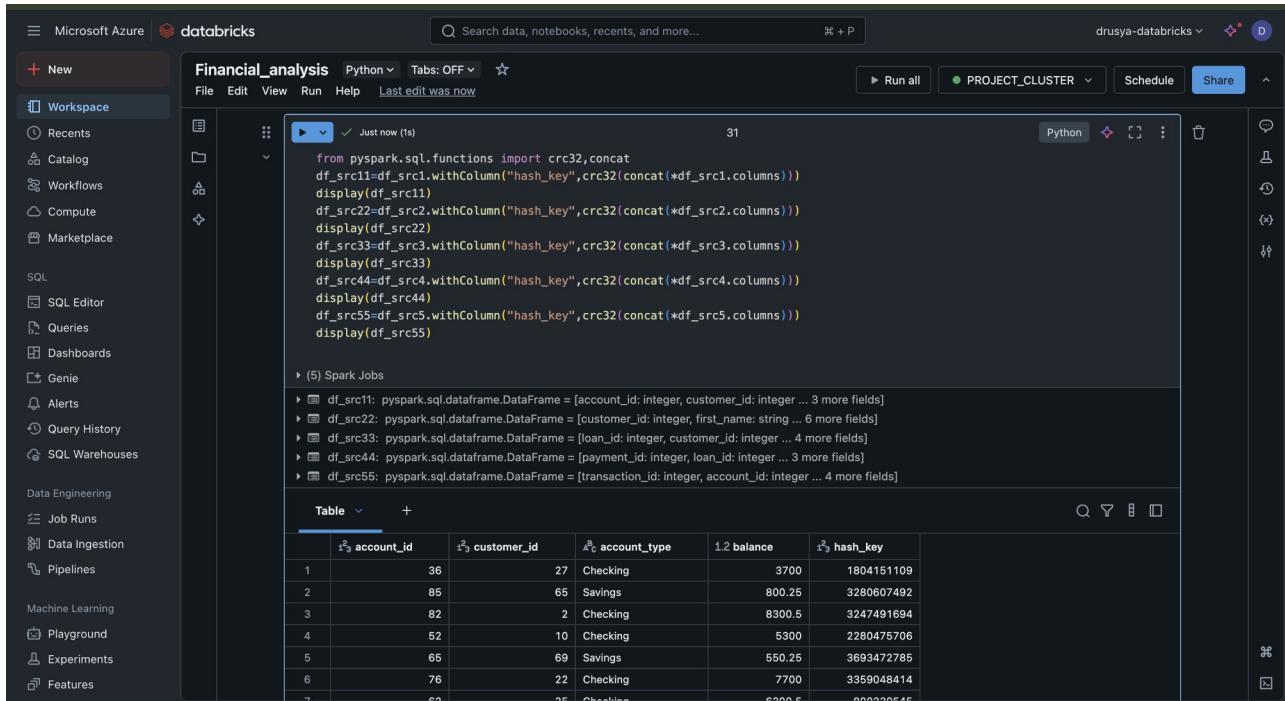
Just now (15s) 29
df_src1 = spark.read.format("parquet").load(src_path1)
display(df_src1)
df_src2 = spark.read.format("parquet").load(src_path2)
display(df_src2)
df_src3 = spark.read.format("parquet").load(src_path3)
display(df_src3)
df_src4 = spark.read.format("parquet").load(src_path4)
display(df_src4)
df_src5 = spark.read.format("parquet").load(src_path5)
display(df_src5)

(10) Spark Jobs
df_src1: pyspark.sql.dataframe.DataFrame = [account_id: integer, customer_id: integer ... 2 more fields]
df_src2: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 5 more fields]
df_src3: pyspark.sql.dataframe.DataFrame = [loan_id: integer, customer_id: integer ... 3 more fields]
df_src4: pyspark.sql.dataframe.DataFrame = [payment_id: integer, loan_id: integer ... 2 more fields]
df_src5: pyspark.sql.dataframe.DataFrame = [transaction_id: integer, account_id: integer ... 3 more fields]

Table +
```

| account_id | customer_id | account_type | balance | |
|------------|-------------|--------------|----------|--------|
| 11 | 72 | 17 | Checking | 7300 |
| 12 | 86 | 21 | Checking | 8700.5 |
| 13 | 83 | 82 | Savings | 775.75 |
| 14 | 3 | 78 | Savings | 1500 |
| 15 | 97 | 90 | Savings | 950.25 |
| 16 | 12 | 81 | Checking | 2700 |
| 17 | 43 | 83 | Savings | 275.75 |

- Now added a new column hashkey so that we can compare the data with other data files to check it is a new or updated data. For that use crc32 command.



The screenshot shows a Databricks notebook titled "Financial_analysis" in Python. The code adds a "hash_key" column to each DataFrame using the `crc32(concat(*df_src1.columns))` function. The updated DataFrames are then displayed. A preview of the DataFrame is shown below the code.

```

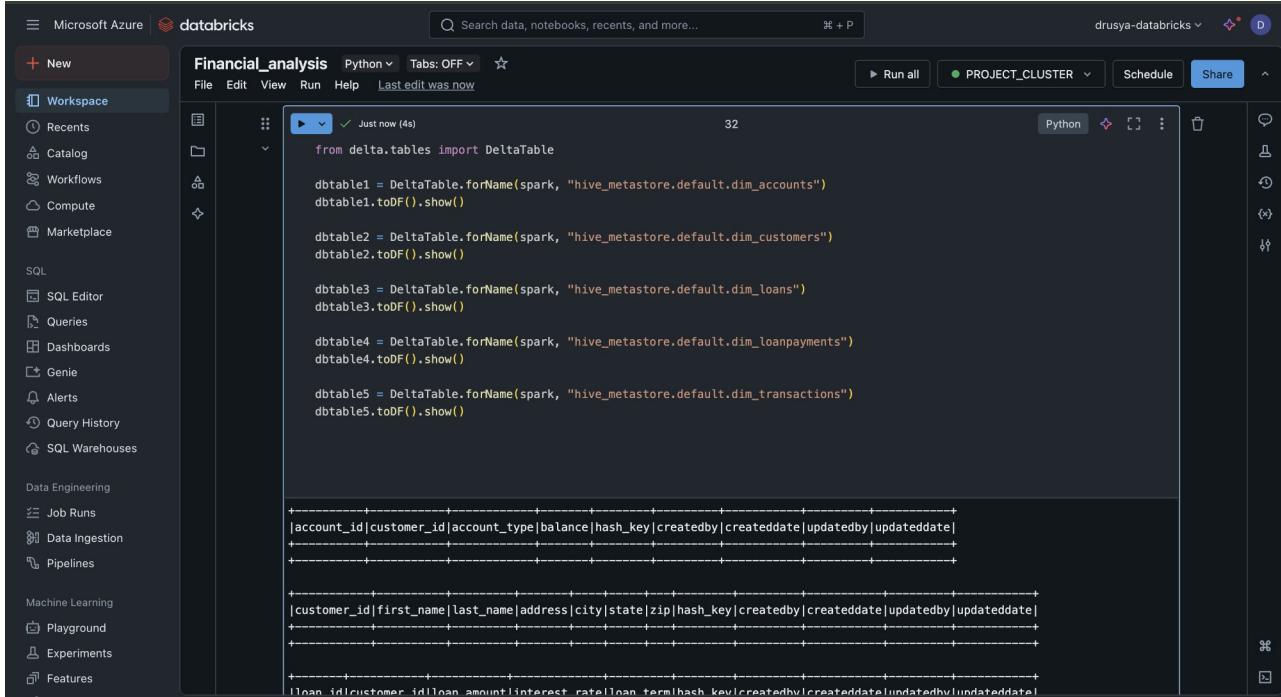
from pyspark.sql.functions import crc32,concat
df_src1=df_src1.withColumn("hash_key",crc32(concat(*df_src1.columns)))
display(df_src1)
df_src2=df_src2.withColumn("hash_key",crc32(concat(*df_src2.columns)))
display(df_src2)
df_src3=df_src3.withColumn("hash_key",crc32(concat(*df_src3.columns)))
display(df_src3)
df_src4=df_src4.withColumn("hash_key",crc32(concat(*df_src4.columns)))
display(df_src4)
df_src5=df_src5.withColumn("hash_key",crc32(concat(*df_src5.columns)))
display(df_src5)

(5) Spark Jobs
df_src1: pyspark.sql.dataframe.DataFrame = [account_id: integer, customer_id: integer ... 3 more fields]
df_src2: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 6 more fields]
df_src3: pyspark.sql.dataframe.DataFrame = [loan_id: integer, customer_id: integer ... 4 more fields]
df_src4: pyspark.sql.dataframe.DataFrame = [payment_id: integer, loan_id: integer ... 3 more fields]
df_src5: pyspark.sql.dataframe.DataFrame = [transaction_id: integer, account_id: integer ... 4 more fields]

Table +
```

| account_id | customer_id | account_type | balance | hash_key |
|------------|-------------|--------------|----------|------------|
| 1 | 36 | 27 | Checking | 1804151109 |
| 2 | 85 | 65 | Savings | 3280607492 |
| 3 | 82 | 2 | Checking | 8300.5 |
| 4 | 52 | 10 | Checking | 2280475706 |
| 5 | 65 | 69 | Savings | 550.25 |
| 6 | 76 | 22 | Checking | 3693472785 |
| 7 | 60 | 25 | Checking | 3359048414 |

- Now we are converting our delta target table into a data frame and is displaying it.



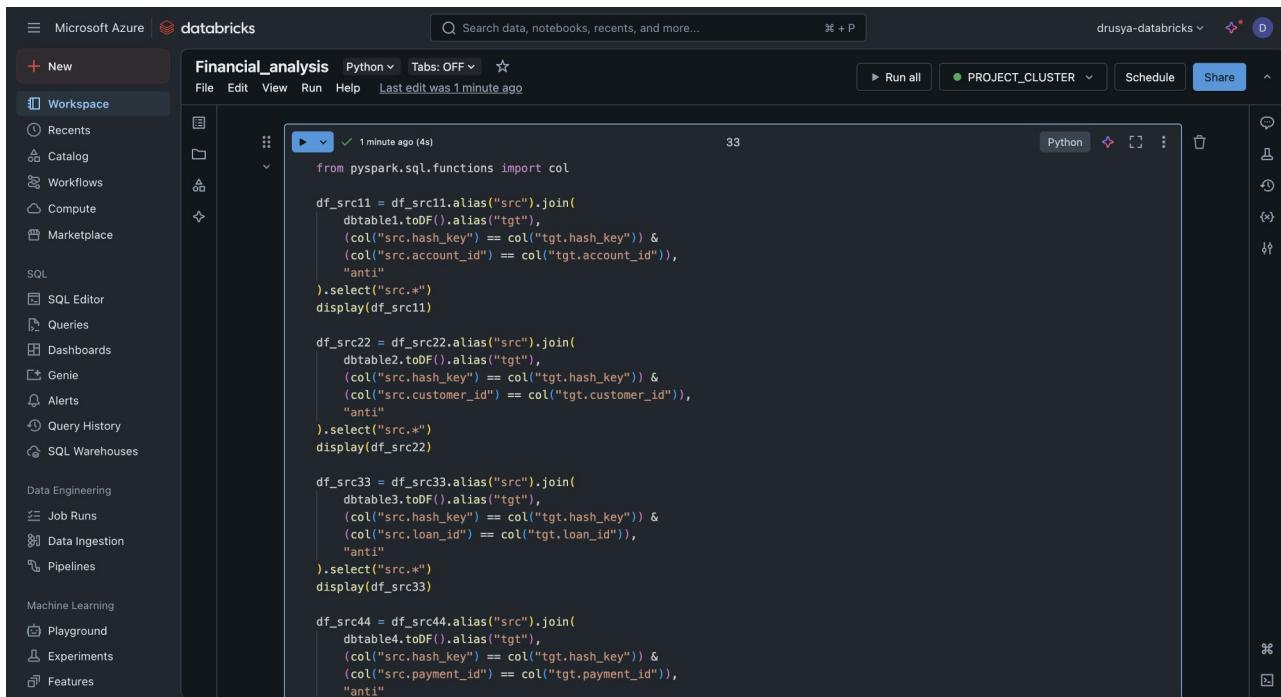
The screenshot shows a Databricks notebook titled "Financial_analysis" in Python. The code imports `DeltaTable` and creates five `DeltaTable` objects: `dbtable1` through `dbtable5`, corresponding to different dimensions in the hive metastore. Below the code, the schemas of these tables are displayed as tabular structures:

```

|account_id|customer_id|account_type|balance|hash_key|createdby|createddate|updatedby|updateddate|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|customer_id|first_name|last_name|address|city|state|zip|hash_key|createdby|createddate|updatedby|updateddate|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|loan_id|customer_id|loan_amount|interest_rate|loan_term|hash_key|createdby|createddate|updatedby|updateddate|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

- Now join the source and target based on the ID so that in the next steps we can compare hash key and ID to determine is it an insert or update.



The screenshot shows a Databricks notebook titled "Financial_analysis" in Python. The code performs four joins between source DataFrames (`df_src11` through `df_src44`) and their corresponding target tables (`dbtable1` through `dbtable4`). The joins are performed using the `join` method with `on` clauses matching columns from both sides. The resulting DataFrames are then displayed.

- Based on update or insert new records condition merge the files.

Microsoft Azure | databricks

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

drusya-databricks D

Financial_analysis Python Tabs: OFF ⚡

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

Just now (10s) 34

```
from pyspark.sql.functions import col, current_timestamp, lit

dbtable1.alias("tgt").merge(
    df_src11.alias("src"),
    "tgt.account_id = src.account_id"
).whenMatchedUpdate(
    set = {
        "tgt.account_id": col("src.account_id"),
        "tgt.customer_id": col("src.customer_id"),
        "tgt.account_type": col("src.account_type"),
        "tgt.balance": col("src.balance"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updatedate": current_timestamp(),
        "tgt.updatedby": lit("databricks")
    }
).whenNotMatchedInsert(
    values = {
        "tgt.account_id": col("src.account_id"),
        "tgt.customer_id": col("src.customer_id"),
        "tgt.account_type": col("src.account_type"),
        "tgt.balance": col("src.balance"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updatedate": current_timestamp(),
        "tgt.updatedby": lit("databricks"),
        "tgt.createddate": current_timestamp(),
        "tgt.createdby": lit("databricks") # Removed inserted_date column
    }
).execute()
```

Microsoft Azure | databricks

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

drusya-databricks D

Financial_analysis Python Tabs: OFF ⚡

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

Just now (3s) 35

```
from pyspark.sql.functions import col, current_timestamp, lit

dbtable2.alias("tgt").merge(
    df_src22.alias("src"),
    "tgt.customer_id = src.customer_id"
).whenMatchedUpdate(
    set = {
        "tgt.customer_id": col("src.customer_id"),
        "tgt.first_name": col("src.first_name"),
        "tgt.last_name": col("src.last_name"),
        "tgt.address": col("src.address"),
        "tgt.city": col("src.city"),
        "tgt.state": col("src.state"),
        "tgt.zip": col("src.zip"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updatedate": current_timestamp(),
        "tgt.updatedby": lit("databricks")
    }
).whenNotMatchedInsert(
    values = {
        "tgt.customer_id": col("src.customer_id"),
        "tgt.first_name": col("src.first_name"),
        "tgt.last_name": col("src.last_name"),
        "tgt.address": col("src.address"),
        "tgt.city": col("src.city"),
        "tgt.state": col("src.state"),
        "tgt.zip": col("src.zip"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updatedate": current_timestamp(),
        "tgt.updatedby": lit("databricks"),
        "tgt.createddate": current_timestamp(),
        "tgt.createdby": lit("databricks")
    }
).execute()
```

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF ⚡

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

New Workspace Recents Catalog Workflows Compute Marketplace SQL SQL Editor Queries Dashboards Genie Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Pipelines Machine Learning Playground Experiments Features

```
from pyspark.sql.functions import col, current_timestamp, lit

dbtable3.alias("tgt").merge(
    df_src33.alias("src"),
    "tgt.loan_id = src.loan_id"
).whenMatchedUpdate(
    set = {
        "tgt.loan_id": col("src.loan_id"),
        "tgt.customer_id": col("src.customer_id"),
        "tgt.loan_amount": col("src.loan_amount"),
        "tgt.interest_rate": col("src.interest_rate"),
        "tgt.loan_term": col("src.loan_term"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updateddate": current_timestamp(),
        "tgt.updatedby": lit("databricks")
    }
).whenNotMatchedInsert(
    values = {
        "tgt.loan_id": col("src.loan_id"),
        "tgt.customer_id": col("src.customer_id"),
        "tgt.loan_amount": col("src.loan_amount"),
        "tgt.interest_rate": col("src.interest_rate"),
        "tgt.loan_term": col("src.loan_term"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updateddate": current_timestamp(),
        "tgt.updatedby": lit("databricks"),
        "tgt.createddate": current_timestamp(),
        "tgt.createdby": lit("databricks")
    }
).execute()
```

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF ⚡

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

New Workspace Recents Catalog Workflows Compute Marketplace SQL SQL Editor Queries Dashboards Genie Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Pipelines Machine Learning Playground Experiments Features

```
from pyspark.sql.functions import col, current_timestamp, lit

dbtable4.alias("tgt").merge(
    df_src44.alias("src"),
    "tgt.payment_id = src.payment_id"
).whenMatchedUpdate(
    set = {
        "tgt.payment_id": col("src.payment_id"),
        "tgt.loan_id": col("src.loan_id"),
        "tgt.payment_date": col("src.payment_date"),
        "tgt.payment_amount": col("src.payment_amount"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updateddate": current_timestamp(),
        "tgt.updatedby": lit("databricks")
    }
).whenNotMatchedInsert(
    values = {
        "tgt.payment_id": col("src.payment_id"),
        "tgt.loan_id": col("src.loan_id"),
        "tgt.payment_date": col("src.payment_date"),
        "tgt.payment_amount": col("src.payment_amount"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updateddate": current_timestamp(),
        "tgt.updatedby": lit("databricks"),
        "tgt.createddate": current_timestamp(),
        "tgt.createdby": lit("databricks")
    }
).execute()
```

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left, there's a sidebar with various navigation options like Workspace, Recents, Catalog, Workflows, Compute, Marketplace, SQL, SQL Editor, Queries, Dashboards, Genie, Alerts, Query History, and SQL Warehouses. The main area is titled 'Financial_analysis' and contains a Python notebook cell. The code in the cell is for a PySpark DataFrame operation using the `merge` function with `whenMatchedUpdate` and `whenNotMatchedInsert` clauses. It involves joining two DataFrames, `dbtable5` and `df_src55`, on the `transaction_id` column, and then updating or inserting rows based on specific conditions involving columns like `account_id`, `transaction_date`, `transaction_amount`, `transaction_type`, and `hash_key`. The code uses `current_timestamp()` and `lit("databricks")` functions for timestamping and setting values. The cell has a status bar indicating it was run just now (3s) and is currently at row 38. The right side of the screen shows standard Databricks UI elements like 'Run all', 'PROJECT_CLUSTER', 'Schedule', and 'Share' buttons.

```
from pyspark.sql.functions import col, current_timestamp, lit

dbtable5.alias("tgt").merge(
    df_src55.alias("src"),
    "tgt.transaction_id = src.transaction_id"
).whenMatchedUpdate(
    set = {
        "tgt.transaction_id": col("src.transaction_id"),
        "tgt.account_id": col("src.account_id"),
        "tgt.transaction_date": col("src.transaction_date"),
        "tgt.transaction_amount": col("src.transaction_amount"),
        "tgt.transaction_type": col("src.transaction_type"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updatedate": current_timestamp(),
        "tgt.updatedby": lit("databricks")
    }
).whenNotMatchedInsert(
    values = {
        "tgt.transaction_id": col("src.transaction_id"),
        "tgt.account_id": col("src.account_id"),
        "tgt.transaction_date": col("src.transaction_date"),
        "tgt.transaction_amount": col("src.transaction_amount"),
        "tgt.transaction_type": col("src.transaction_type"),
        "tgt.hash_key": col("src.hash_key"),
        "tgt.updatedate": current_timestamp(),
        "tgt.updatedby": lit("databricks"),
        "tgt.createdate": current_timestamp(),
        "tgt.createdby": lit("databricks")
    }
).execute()
```

- Now read the target file.

The screenshot shows the Microsoft Azure Databricks workspace interface. The left sidebar includes sections for New, Workspace, Recents, Catalog, Workflows, Compute, Marketplace, SQL, and various engineering and machine learning categories. The main area displays a notebook titled "Financial_analysis" in Python, with the code block containing:

```
display(spark.read.format("delta").load(tgt_path1))
```

The output of the code is a table titled "Table" showing 100 rows of data. The columns are:

| | account_id | customer_id | account_type | balance | hash_key | createdby | createddate |
|----|------------|-------------|--------------|---------|------------|------------|-------------------------|
| 1 | 36 | 27 | Checking | 3700 | 1804151109 | databricks | 2025-04-27T05:31:00.426 |
| 2 | 85 | 65 | Savings | 800.25 | 3280607492 | databricks | 2025-04-27T05:31:00.426 |
| 3 | 82 | 2 | Checking | 8300.5 | 3247491694 | databricks | 2025-04-27T05:31:00.426 |
| 4 | 52 | 10 | Checking | 5300 | 2280475706 | databricks | 2025-04-27T05:31:00.426 |
| 5 | 65 | 69 | Savings | 550.25 | 3693472785 | databricks | 2025-04-27T05:31:00.426 |
| 6 | 76 | 22 | Checking | 7700 | 3359048414 | databricks | 2025-04-27T05:31:00.426 |
| 7 | 62 | 35 | Checking | 6300.5 | 800230545 | databricks | 2025-04-27T05:31:00.426 |
| 8 | 98 | 49 | Checking | 9900.5 | 3893038149 | databricks | 2025-04-27T05:31:00.426 |
| 9 | 53 | 86 | Savings | 400.25 | 1685401127 | databricks | 2025-04-27T05:31:00.426 |
| 10 | 66 | 26 | Checking | 6700.5 | 372242038 | databricks | 2025-04-27T05:31:00.426 |
| 11 | 72 | 17 | Checking | 7300 | 1425050024 | databricks | 2025-04-27T05:31:00.426 |
| 12 | 86 | 21 | Checking | 8700.5 | 2919860885 | databricks | 2025-04-27T05:31:00.426 |
| 13 | 83 | 82 | Savings | 775.75 | 293251303 | databricks | 2025-04-27T05:31:00.426 |
| 14 | 3 | 78 | Savings | 1500 | 1996545678 | databricks | 2025-04-27T05:31:00.426 |
| 15 | 97 | 90 | Savings | 950.25 | 270206465 | databricks | 2025-04-27T05:31:00.426 |

At the bottom, it says "100 rows | 0.52s runtime" and "Refreshed now".

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

New Workspace Recents Catalog Workflows Compute Marketplace SQL SQL Editor Queries Dashboards Genie Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Pipelines Machine Learning Playground Experiments Features

Just now (<1s) 40 Python

```
display(spark.read.format("delta").load(tgt_path2))
```

(1) Spark Jobs

Table +

| | first_name | last_name | address | city | state | zip | hash_key | createdby |
|---------------------------------------|----------------------|---------------------|------------------|------------------|----------------|--------------|--------------------|--------------------|
| 2 | 34 | Olivia | Reed | 3333 Birch Blvd | Orillia | ON | L3V0A1 | 2844006823 |
| 3 | 25 | Daniel | Campbell | 2424 Willow Rd | St. Catharines | ON | L2R0A1 | 3303870564 |
| 4 | 43 | Joseph | Cox | 4242 Cedar Ln | Aurora | ON | L4G0A1 | 157755620 |
| 5 | 3 | Michael | Johnson | 789 Oak Dr | Montreal | QC | H1A1A1 | 4114944968 |
| 6 | 19 | Christopher | Baker | 1818 Pine Rd | Thunder Bay | ON | P7A0A1 | 2513240418 |
| 7 | 76 | Evelyn | Wallace | 7575 Birch Blvd | Huntsville | ON | P1H0A1 | 2612860645 |
| 8 | 11 | Alexander | Thomas | 1010 Willow Rd | St. John's | NL | A1A0A1 | 823092523 |
| 9 | 35 | William | Cook | 3434 Spruce Ln | Midland | ON | L4R0A1 | 2677642904 |
| 10 | 37 | Alexander | Bell | 3636 Redwood Dr | Stratford | ON | N5A0A1 | 2227707500 |
| 11 | 36 | Ava | Morgan | 3635 Fir St | Collingwood | ON | L9Y0A1 | 3737688029 |
| 12 | 44 | Amelia | Howard | 4343 Elm St | Bradford | ON | L3Z0A1 | 447506045 |
| 13 | 29 | Michael | Collins | 2828 Cedar Ln | Thunder Bay | ON | P7B0A1 | 2312898578 |
| 14 | 41 | Matthew | Cooper | 4040 Ash Blvd | Georgetown | ON | L7G0A1 | 2809660300 |
| 15 | 52 | Abigail | Henderson | 5151 Cypress Ave | Mount Albert | ON | L0G0A1 | 569556118 |
| .. | .. | .. | .. | .. | .. | .. | .. | .. |
| 86 rows 0.37s runtime Refreshed now | | | | | | | | |

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

New Workspace Recents Catalog Workflows Compute Marketplace SQL SQL Editor Queries Dashboards Genie Alerts Query History SQL Warehouses Data Engineering Job Runs Data Ingestion Pipelines Machine Learning Playground Experiments Features

Just now (<1s) 41 Python

```
display(spark.read.format("delta").load(tgt_path3))
```

(1) Spark Jobs

Table +

| | loan_id | customer_id | loan_amount | interest_rate | loan_term | hash_key | createdby | c |
|---|-------------------|-----------------------|-----------------------|-------------------------|---------------------|--------------------|--------------------|------------|
| 1 | 5 | 56 | 25000 | 5 | 36 | 1867834730 | databricks | 2028 |
| 2 | 84 | 40 | 30000 | 3.5 | 24 | 3326743868 | databricks | 2028 |
| 3 | 87 | 93 | 22500.75 | 6.5 | 60 | 3240117383 | databricks | 2028 |
| 4 | 51 | 72 | 10000.75 | 6 | 60 | 3592350602 | databricks | 2028 |
| 5 | 68 | 8 | 27500 | 3.5 | 24 | 4092520027 | databricks | 2028 |
| 6 | 95 | 60 | 25000.75 | 6.5 | 60 | 3959425324 | databricks | 2028 |
| 7 | 57 | 97 | 22500.25 | 5.5 | 36 | 1270770825 | databricks | 2028 |
| 8 | 93 | 79 | 15000.25 | 5 | 36 | 2461134402 | databricks | 2028 |
| 9 | 23 | 88 | 15000.75 | 6.5 | 60 | 1778064991 | databricks | 2028 |
| 10 | 75 | 61 | 25000.75 | 6 | 60 | 920703527 | databricks | 2028 |
| 11 | 28 | 7 | 27500 | 3.5 | 24 | 1591891469 | databricks | 2028 |
| 12 | 34 | 41 | 30000.5 | 4.5 | 48 | 4228268298 | databricks | 2028 |
| 13 | 9 | 14 | 32500.25 | 5.5 | 36 | 3981072152 | databricks | 2028 |
| 14 | 17 | 99 | 22500.25 | 5.5 | 36 | 4079820486 | databricks | 2028 |
| 15 | 55 | 63 | 25000.75 | 6.5 | 60 | 91833510 | databricks | 2028 |
| 100 rows 0.41s runtime Refreshed 1 minute ago | | | | | | | | |

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

Table

```
display(spark.read.format("delta").load(tgt_path4))
```

(1) Spark Jobs

| # | payment_id | loan_id | payment_date | payment_amount | hash_key | createdby | createddate |
|----|------------|---------|--------------|----------------|------------|------------|--------------------|
| 1 | 13 | 44 | 2024-01-13 | 700 | 1774633368 | databricks | 2025-04-27T06:02:2 |
| 2 | 32 | 53 | 2024-02-01 | 1650 | 3300311926 | databricks | 2025-04-27T06:02:2 |
| 3 | 93 | 24 | 2024-04-02 | 4700 | 3816339412 | databricks | 2025-04-27T06:02:2 |
| 4 | 76 | 37 | 2024-03-16 | 3850 | 175174851 | databricks | 2025-04-27T06:02:2 |
| 5 | 30 | 31 | 2024-01-30 | 1550 | 3995935809 | databricks | 2025-04-27T06:02:2 |
| 6 | 67 | 38 | 2024-03-07 | 3400 | 3867437585 | databricks | 2025-04-27T06:02:2 |
| 7 | 77 | 48 | 2024-03-17 | 3900 | 1459068700 | databricks | 2025-04-27T06:02:2 |
| 8 | 50 | 51 | 2024-02-19 | 2550 | 3731656210 | databricks | 2025-04-27T06:02:2 |
| 9 | 16 | 77 | 2024-01-16 | 850 | 304986457 | databricks | 2025-04-27T06:02:2 |
| 10 | 100 | 33 | 2024-04-10 | 1000 | 3282935963 | databricks | 2025-04-27T06:02:2 |
| 11 | 19 | 10 | 2024-01-19 | 1000 | 1347506748 | databricks | 2025-04-27T06:02:2 |
| 12 | 42 | 63 | 2024-02-11 | 2150 | 929699495 | databricks | 2025-04-27T06:02:2 |
| 13 | 8 | 78 | 2024-01-08 | 450 | 2203091019 | databricks | 2025-04-27T06:02:2 |
| 14 | 29 | 20 | 2024-01-29 | 1500 | 1393641991 | databricks | 2025-04-27T06:02:2 |
| 15 | 64 | 5 | 2024-03-04 | 3250 | 3762579497 | databricks | 2025-04-27T06:02:2 |

↓ 100 rows | 0.43s runtime Refreshed now

Microsoft Azure | databricks

Financial_analysis Python Tabs: OFF

File Edit View Run Help Last edit was now

Run all PROJECT_CLUSTER Schedule Share

Table

```
display(spark.read.format("delta").load(tgt_path5))
```

(1) Spark Jobs

| # | n_id | account_id | transaction_date | transaction_amount | transaction_type | hash_key | createdby |
|----|------|------------|------------------|--------------------|------------------|------------|------------|
| 1 | 32 | 9 | 2024-02-01 | 200.75 | Withdrawal | 4150335909 | databricks |
| 2 | 11 | 3 | 2024-01-11 | 100.5 | Deposit | 826954426 | databricks |
| 3 | 25 | 66 | 2024-01-25 | 250 | Deposit | 3839839217 | databricks |
| 4 | 10 | 92 | 2024-01-10 | 375.25 | Withdrawal | 1681821004 | databricks |
| 5 | 30 | 32 | 2024-01-30 | 375.25 | Withdrawal | 401551885 | databricks |
| 6 | 61 | 52 | 2024-03-01 | 100.5 | Deposit | 2246730977 | databricks |
| 7 | 97 | 90 | 2024-04-06 | 225.5 | Deposit | 2081911181 | databricks |
| 8 | 38 | 15 | 2024-02-07 | 275.75 | Withdrawal | 1006322033 | databricks |
| 9 | 1 | 45 | 2024-01-01 | 100.5 | Deposit | 2846636750 | databricks |
| 10 | 2 | 12 | 2024-01-02 | 200.75 | Withdrawal | 463284897 | databricks |
| 11 | 18 | 5 | 2024-01-18 | 275.75 | Withdrawal | 2726697585 | databricks |
| 12 | 54 | 42 | 2024-02-23 | 300.25 | Withdrawal | 3238602855 | databricks |
| 13 | 80 | 30 | 2024-03-20 | 375.25 | Withdrawal | 2478637051 | databricks |
| 14 | 23 | 88 | 2024-01-23 | 150 | Deposit | 3611063752 | databricks |
| 15 | 39 | 74 | 2024-02-08 | 325 | Deposit | 1677328493 | databricks |

↓ 100 rows | 0.40s runtime Refreshed now

STEP 4: SCHEDULING THE NOTEBOOK USING WORKFLOWS

- Go to the workflows → Create jobs

The screenshot shows the Databricks interface with the 'Workflows' section selected. A modal window titled 'Select Notebook' is open, showing a list of notebooks under the 'Users' workspace. The notebook 'Financial_analysis' is selected. The main workspace on the right displays basic job details like ID, creator, and triggers.

The screenshot shows the 'New Job' configuration dialog. The task is named 'Loan_details', type is 'Notebook', source is 'Workspace', path is '/Workspace/Users/danidvs97@outlook.com/Financial_analysis', and compute is 'PROJECT_CLUSTER'. The 'Schedules & Triggers' section indicates 'None'. The 'Job notifications' section shows 'No notifications'.

The screenshot shows the Databricks Jobs interface. On the left, there's a sidebar with various navigation options like Workspace, Recents, Catalog, Workflows, Compute, Marketplace, SQL, Data Engineering, Machine Learning, and more. The main area shows a job named "New Job Apr 27, 2025, 03:05 AM". The "Tasks" tab is selected. A modal window titled "Schedules & Triggers" is open. Inside, the "Trigger type" dropdown is set to "None (manual)". Below it, the "Type*" dropdown has "Scheduled" selected. Other options include "File arrival" and "Continuous". The "Source*" dropdown is set to "Workspace", and the "Path*" input field contains "/Workspace/Users/danidvs97@outlook.com/Financial_analysis". The "Compute*" dropdown is set to "PROJECT_CLUSTER". The "Dependent libraries" section has a "+ Add" button. On the right side of the interface, there are sections for "Job details" (Job ID: 740012762385767, Creator: dani davis), "Schedules & Triggers" (None), and "Compute" (PROJECT_CLUSTER).

This screenshot shows the same Databricks Jobs interface as the previous one, but with different configuration settings in the "Schedules & Triggers" dialog. The "Trigger Status" section now has "Active" selected instead of "Paused". The "Trigger type" dropdown still shows "Scheduled". The "Schedule type" section includes "Simple" and "Advanced" buttons, with "Simple" currently selected. The "Periodic" section shows "Every 1 Day". The rest of the interface remains the same, including the "Job details" and "Compute" sections on the right.

- Added the git hub repository details also.

The screenshot shows the Databricks interface for creating a new job. The left sidebar is the navigation menu, and the main area is the 'Jobs' configuration screen.

Job Details:

- Job ID: 740012762385767
- Creator: dani davis
- Run as: dani davis
- Tags: Add tag
- Description: Add description
- Lineage: No lineage information for this job. Learn more

Git:

- Repository URL: github.com/Drusya-23/Data-engineering-projects
- Branch: Project2_customer-financial-analysis-using-databricks

Schedules & Triggers:

- Every day, next run at Apr 27, 2025, 04:45 AM
- Edit trigger, Pause, Delete

Task Configuration:

- Task name:** Loan_details
- Type:** Notebook
- Source:** Workspace
- Path:** /Workspace/Users/danidvs97@outlook.com/Financial_analysis
- Compute:** PROJECT_CLUSTER (16 GB - 4 Cores · DBR 15.4 LTS - Photon - Spark 3.5...)
- Note: Jobs running on all-purpose clusters are considered all-purpose compute. Learn more

Buttons: Cancel, Save task

The screenshot shows the Databricks interface for managing workflows. The left sidebar is the navigation menu, and the main area is the 'Workflows' list screen.

Workflows List:

- Jobs: New Job Apr 27, 2025, 03:05 AM (Owned by me, Accessible by me, Favorites, Tags)
- Job runs: None
- Pipelines: None

Filtering and Actions:

- Filter jobs: Filter jobs
- Owned by me, Accessible by me, Favorites, Tags
- Load tutorial, Create job

Job Details:

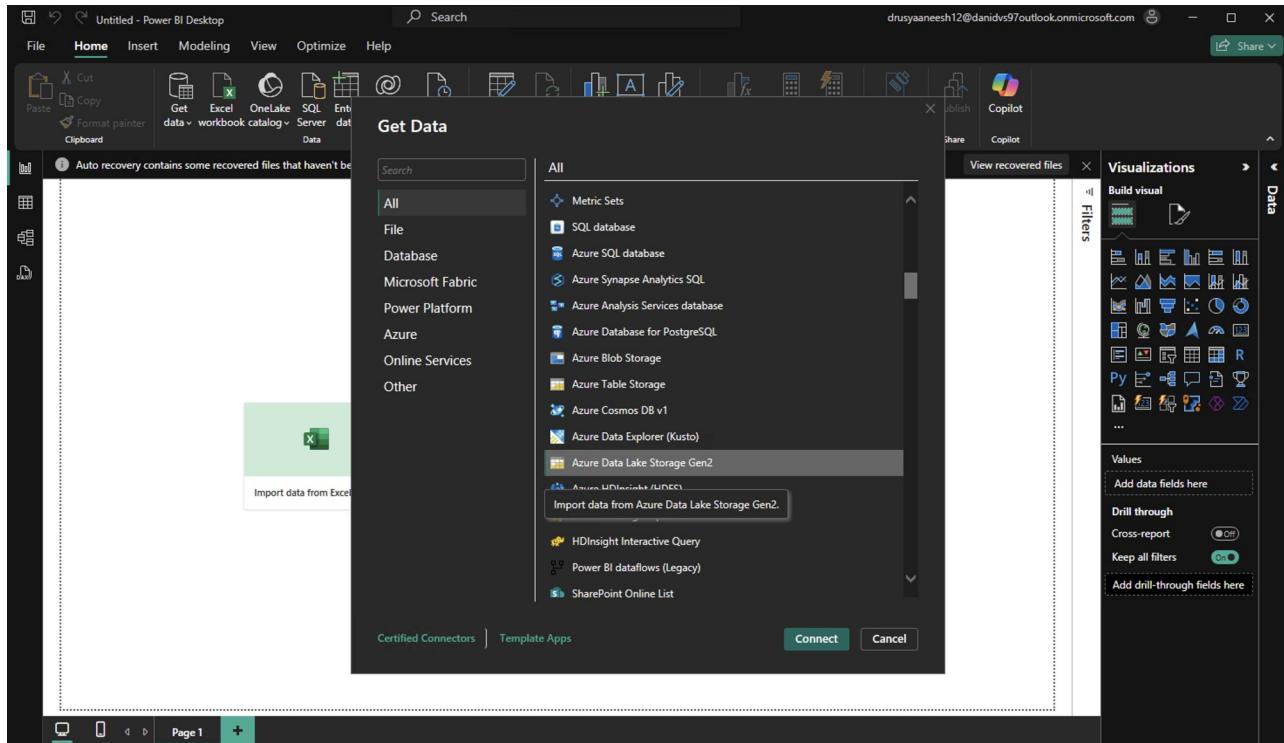
- Name: New Job Apr 27, 2025, 03:05 AM
- Created by: dani davis
- Trigger: Scheduled
- Recent runs: None

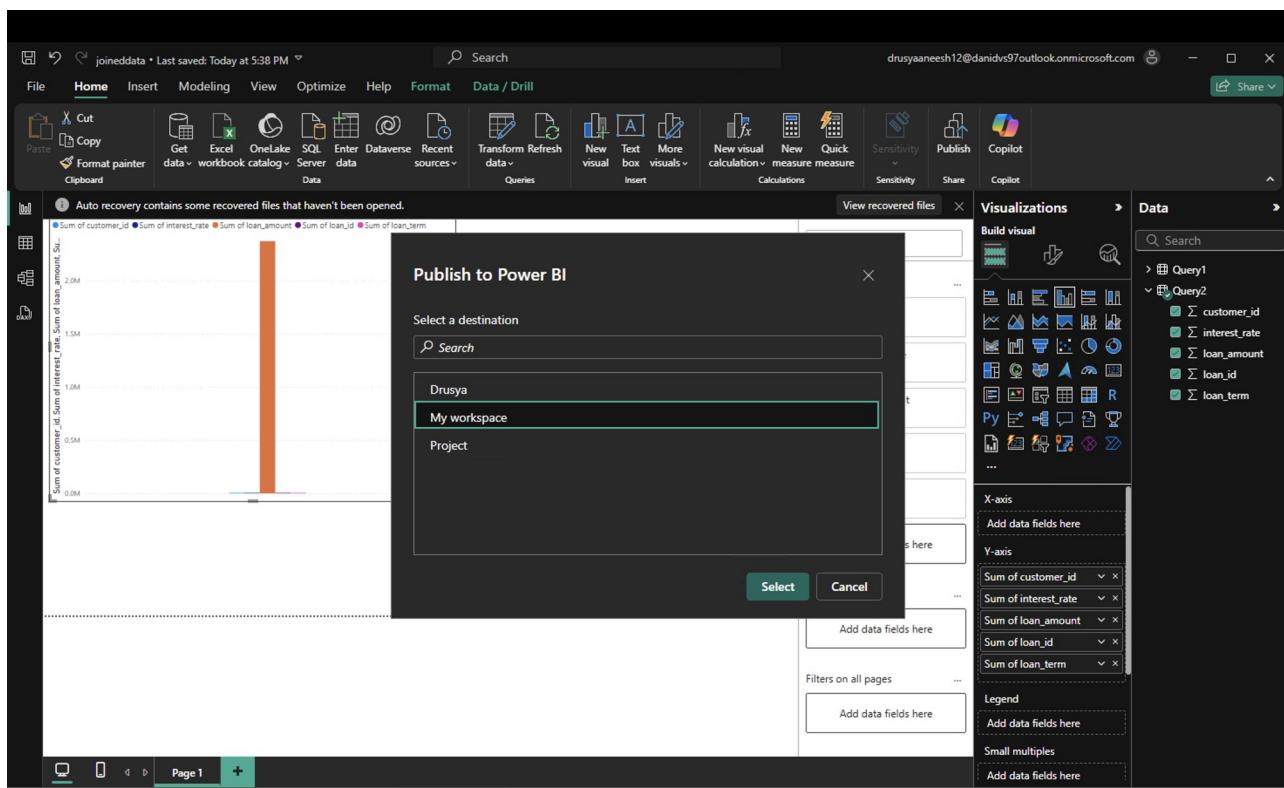
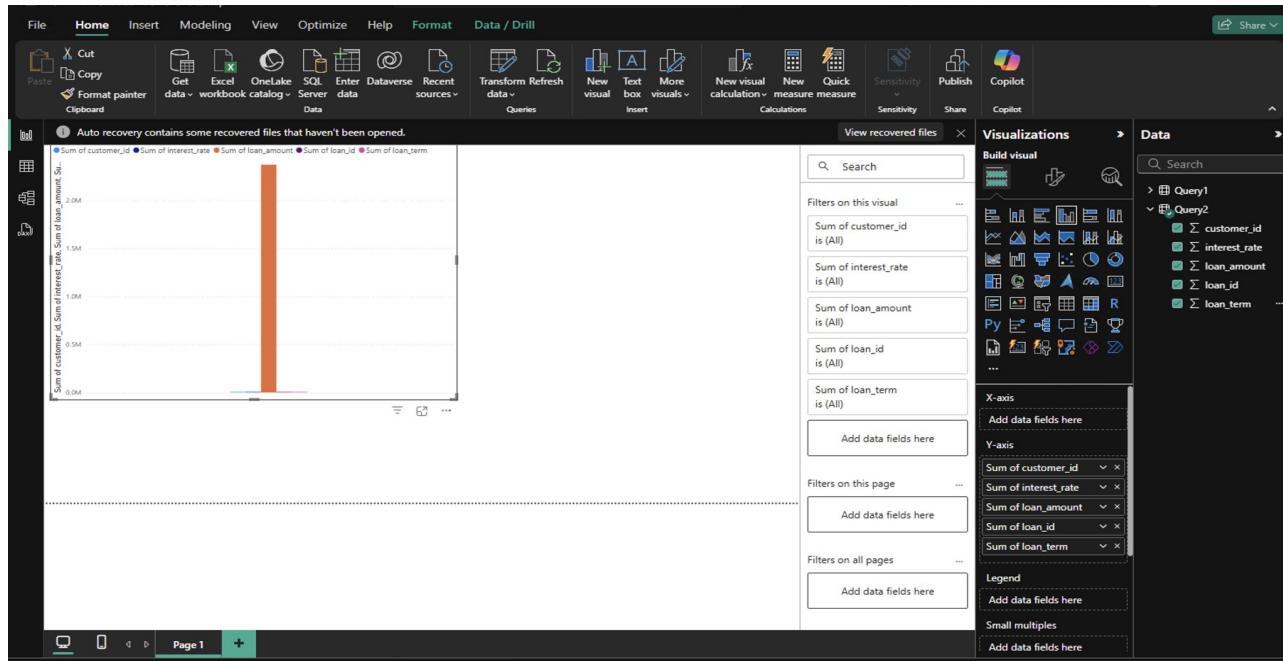
Navigation:

- < Previous, Next >

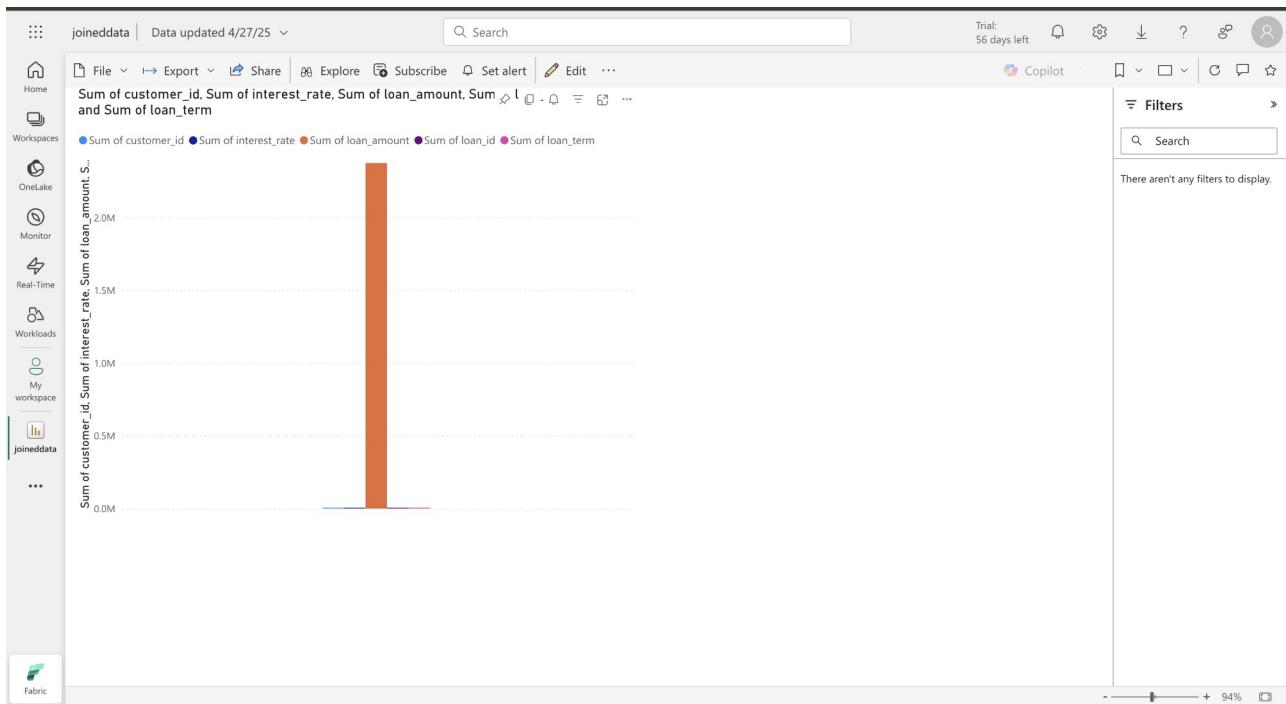
STEP 5: CREATE A REPORT OF THE JOINED FILE IN POWER BI AND PUBLISH IT IN FABRIC.

- Go to the power BI desktop and connect the storage account with power BI. Select the table we want to visualize and publish it to the fabric workspace.





- Successfully loaded the visualization in Fabric workspace.



CONCLUSION

Build an efficient and robust transformation of financial data of customers with the help of Databricks notebooks. Visualization of the files were done using Power BI and Fabric. Also done SCD Type 1 transformations on the cleaned data successfully and triggered the work to run on particular time using the workflows in Data bricks.