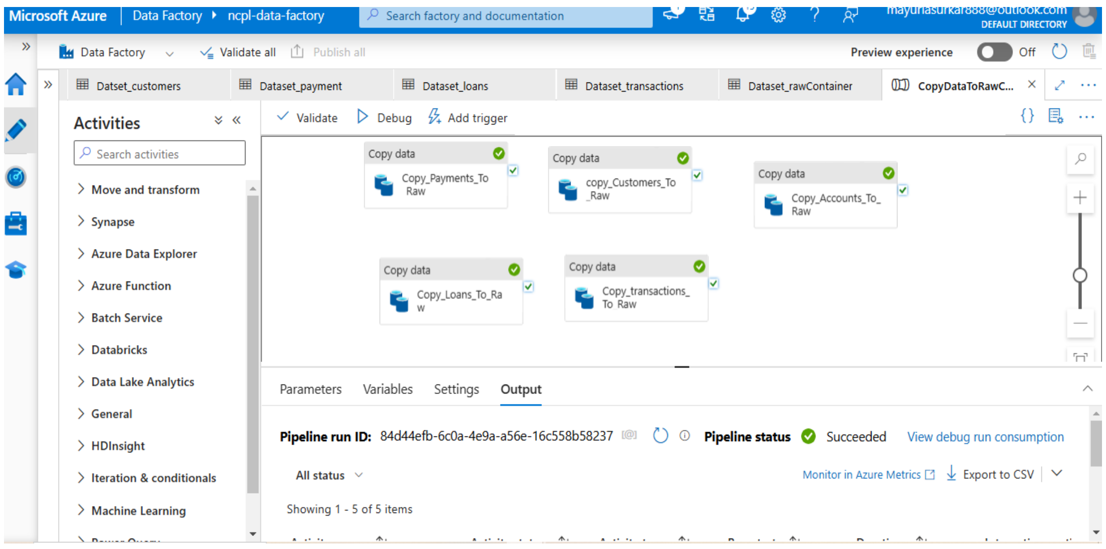
1. Data Ingestion (Backend Storage to Raw Container)

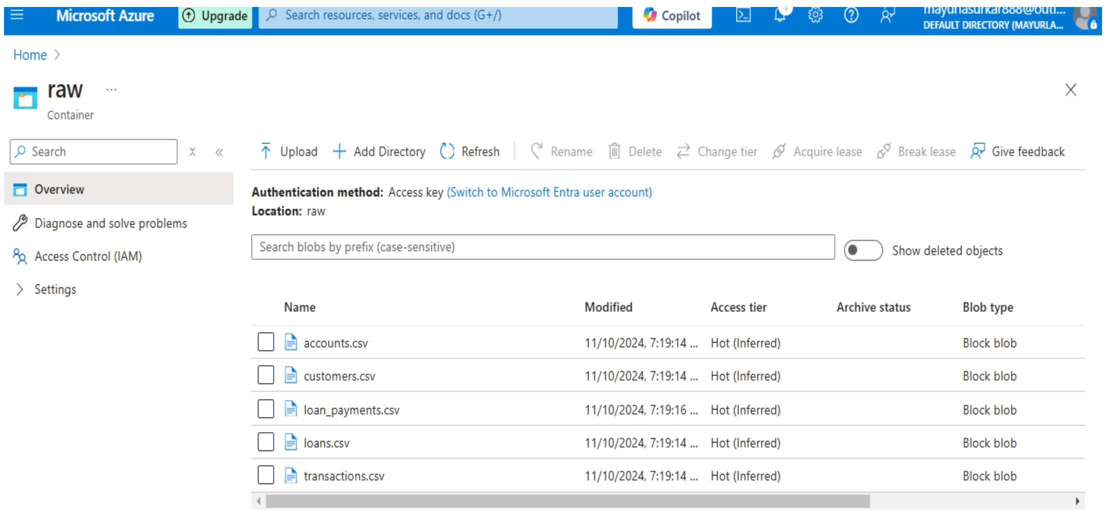
Copying data from a backend storage account to your Azure Data Lake's raw (bronze) container using Azure Data Factory (ADF).

- Set Up Azure Data Factory
- Set Up Linked Services in ADF
- Create Datasets in ADF
- Create a Copy Activity Pipeline
- Run the Pipeline



Pipeline run ID: 84d44efb-6c0a-4e9a-a56e-16c558b58237 Pipeline status Succeeded View debug run consumption

Showing 1 - 5 of 5 items

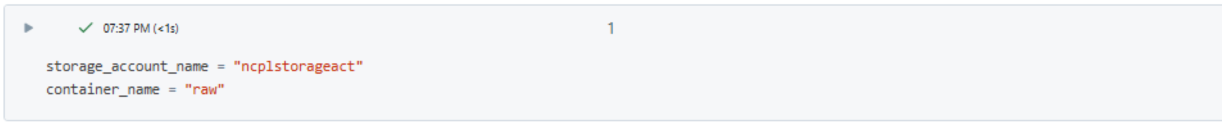


Authentication method: Access key (Switch to Microsoft Entra user account)
Location: raw

Name	Modified	Access tier	Archive status	Blob type
accounts.csv	11/10/2024, 7:19:14 ...	Hot (Inferred)		Block blob
customers.csv	11/10/2024, 7:19:14 ...	Hot (Inferred)		Block blob
loan_payments.csv	11/10/2024, 7:19:16 ...	Hot (Inferred)		Block blob
loans.csv	11/10/2024, 7:19:14 ...	Hot (Inferred)		Block blob
transactions.csv	11/10/2024, 7:19:14 ...	Hot (Inferred)		Block blob

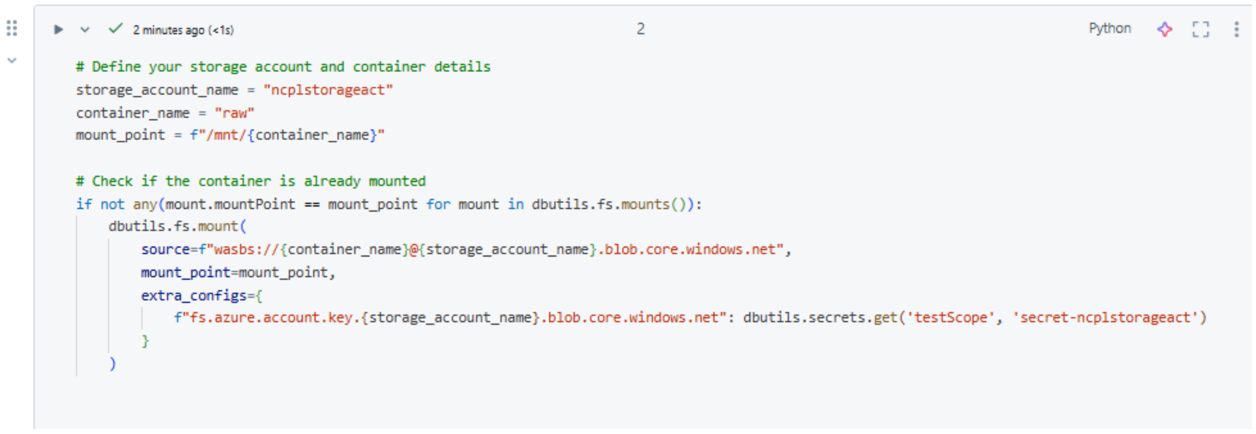
2. Step 2: Databricks Activity (Incremental/Delta Processing)

- Created a Databricks Workspace.
- Created a Databricks Cluster.
- Accessed Azure Data Lake Storage (ADLS) in Databricks.



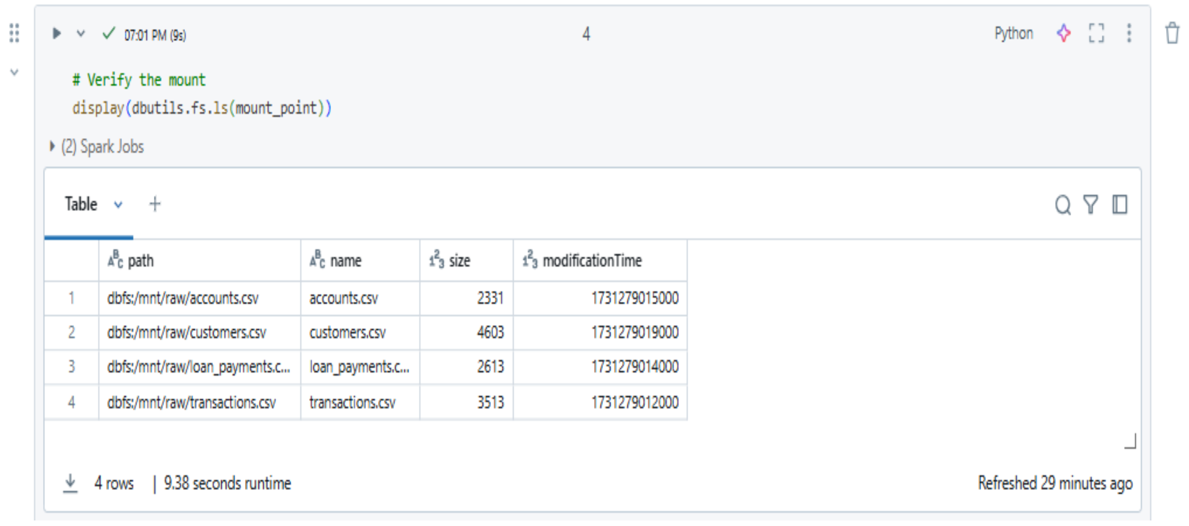
07:37 PM (<1s) 1

```python
storage_account_name = "ncplstorageact"
container_name = "raw"
```



2 minutes ago (<1s) 2 Python

```python
# Define your storage account and container details
storage_account_name = "ncplstorageact"
container_name = "raw"
mount_point = f"/mnt/{container_name}"

# Check if the container is already mounted
if not any(mount.mountPoint == mount_point for mount in dbutils.fs.mounts()):
    dbutils.fs.mount(
        source=f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",
        mount_point=mount_point,
        extra_configs={
            f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": dbutils.secrets.get('testScope', 'secret-ncplstorageact')
        }
    )
```



07:01 PM (9s) 4 Python

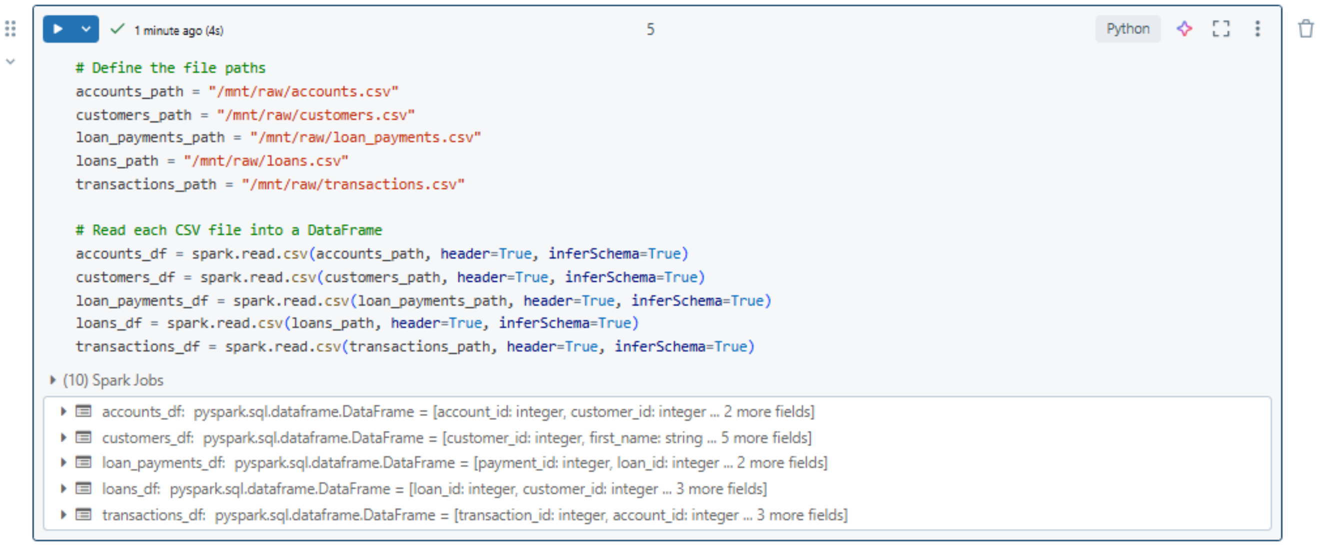
```python
# Verify the mount
display(dbutils.fs.ls(mount_point))
```

(2) Spark Jobs

Table

	path	name	size	modificationTime
1	dbfs:/mnt/raw/accounts.csv	accounts.csv	2331	1731279015000
2	dbfs:/mnt/raw/customers.csv	customers.csv	4603	1731279019000
3	dbfs:/mnt/raw/loan_payments.c...	loan_payments.c...	2613	1731279014000
4	dbfs:/mnt/raw/transactions.csv	transactions.csv	3513	1731279012000

4 rows | 9.38 seconds runtime Refreshed 29 minutes ago

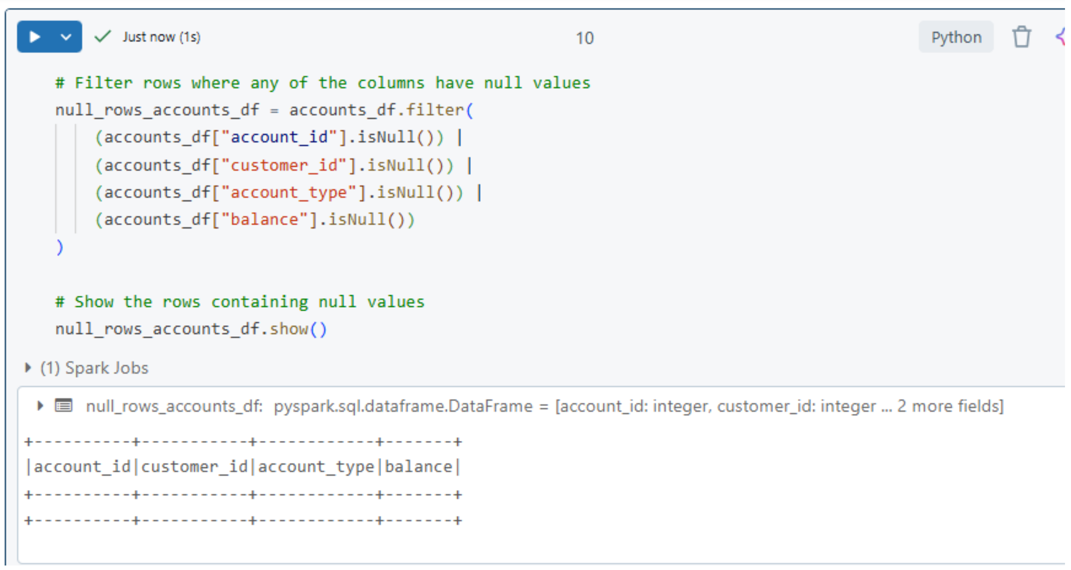


```
# Define the file paths
accounts_path = "/mnt/raw/accounts.csv"
customers_path = "/mnt/raw/customers.csv"
loan_payments_path = "/mnt/raw/loan_payments.csv"
loans_path = "/mnt/raw/loans.csv"
transactions_path = "/mnt/raw/transactions.csv"

# Read each CSV file into a DataFrame
accounts_df = spark.read.csv(accounts_path, header=True, inferSchema=True)
customers_df = spark.read.csv(customers_path, header=True, inferSchema=True)
loan_payments_df = spark.read.csv(loan_payments_path, header=True, inferSchema=True)
loans_df = spark.read.csv(loans_path, header=True, inferSchema=True)
transactions_df = spark.read.csv(transactions_path, header=True, inferSchema=True)
```

```
(10) Spark Jobs
accounts_df:  pyspark.sql.dataframe.DataFrame = [account_id: integer, customer_id: integer ... 2 more fields]
customers_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 5 more fields]
loan_payments_df:  pyspark.sql.dataframe.DataFrame = [payment_id: integer, loan_id: integer ... 2 more fields]
loans_df:  pyspark.sql.dataframe.DataFrame = [loan_id: integer, customer_id: integer ... 3 more fields]
transactions_df:  pyspark.sql.dataframe.DataFrame = [transaction_id: integer, account_id: integer ... 3 more fields]
```

Data Cleaning and transformation

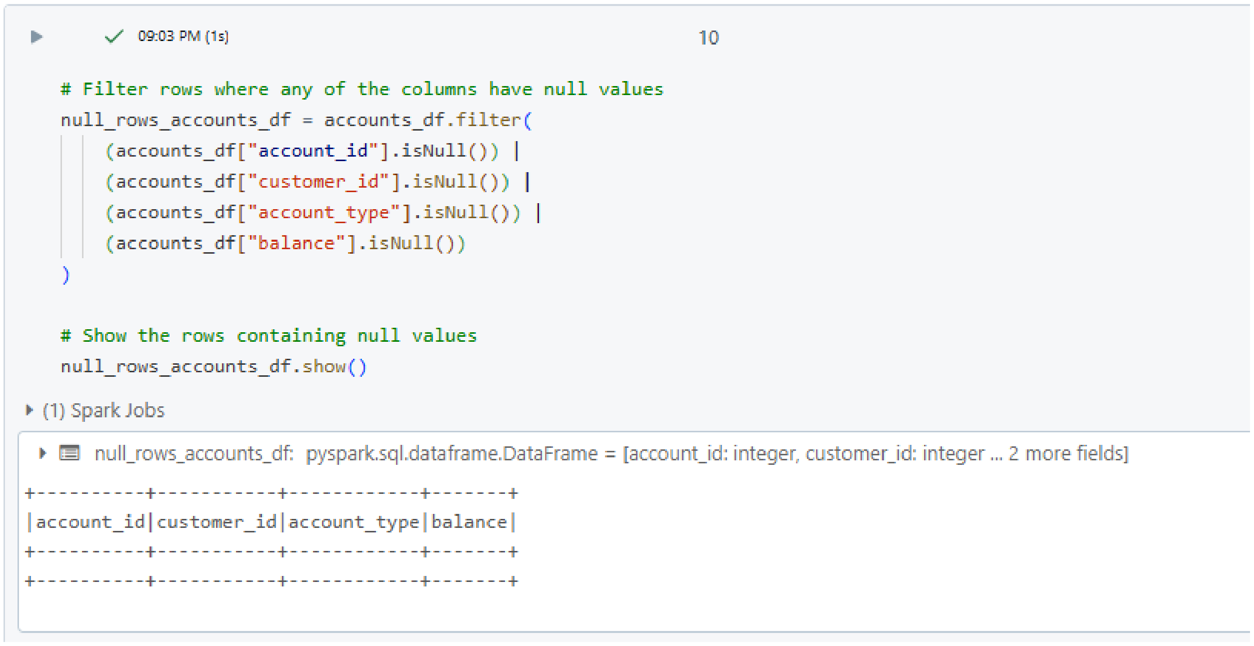
1. Checked rows containing null value in accounts_df



```
# Filter rows where any of the columns have null values
null_rows_accounts_df = accounts_df.filter(
    (accounts_df["account_id"].isNull()) |
    (accounts_df["customer_id"].isNull()) |
    (accounts_df["account_type"].isNull()) |
    (accounts_df["balance"].isNull())
)

# Show the rows containing null values
null_rows_accounts_df.show()
```

```
(1) Spark Jobs
null_rows_accounts_df:  pyspark.sql.dataframe.DataFrame = [account_id: integer, customer_id: integer ... 2 more fields]

+----------+-----------+------------+-------+
|account_id|customer_id|account_type|balance|
+----------+-----------+------------+-------+
+----------+-----------+------------+-------+
```

2. Identifying and Removing Duplicates



09:03 PM (1s) 10

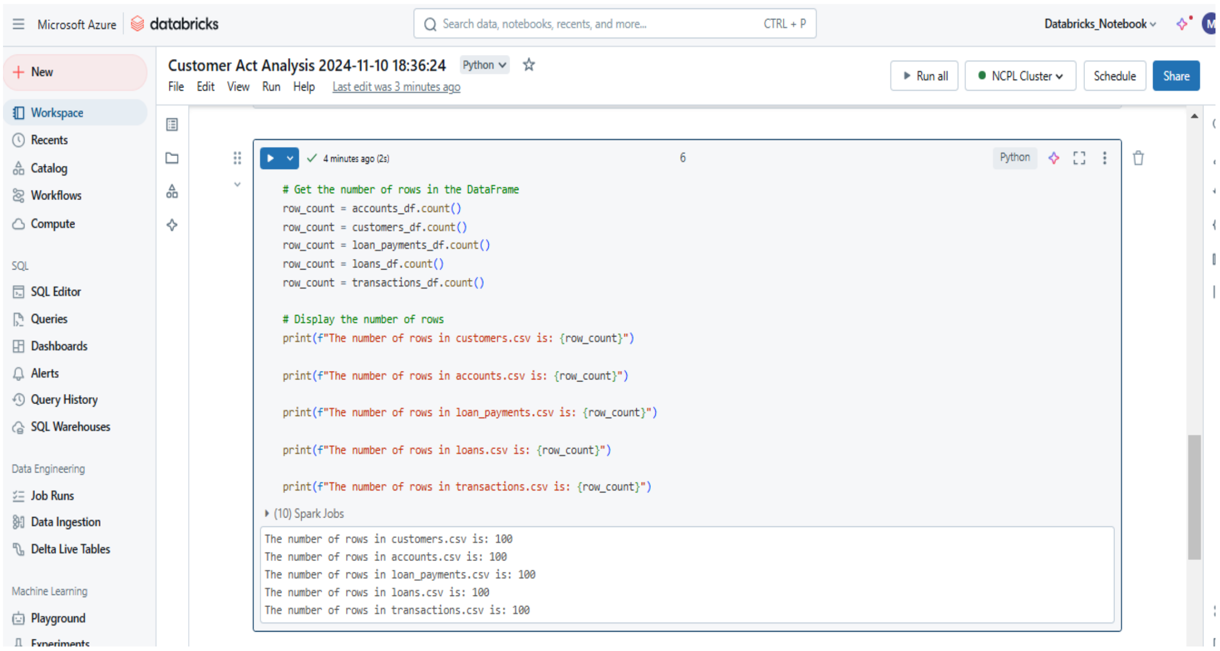
```
# Filter rows where any of the columns have null values
null_rows_accounts_df = accounts_df.filter(
    (accounts_df["account_id"].isNull()) |
    (accounts_df["customer_id"].isNull()) |
    (accounts_df["account_type"].isNull()) |
    (accounts_df["balance"].isNull())
)

# Show the rows containing null values
null_rows_accounts_df.show()
```

(1) Spark Jobs

null_rows_accounts_df: pyspark.sql.dataframe.DataFrame = [account_id: integer, customer_id: integer ... 2 more fields]

```
+----------+-----------+------------+-------+
|account_id|customer_id|account_type|balance|
+----------+-----------+------------+-------+
+----------+-----------+------------+-------+
```

3. Number of rows in each table: 100 in each table



Customer Act Analysis 2024-11-10 18:36:24 Python

Last edit was 3 minutes ago

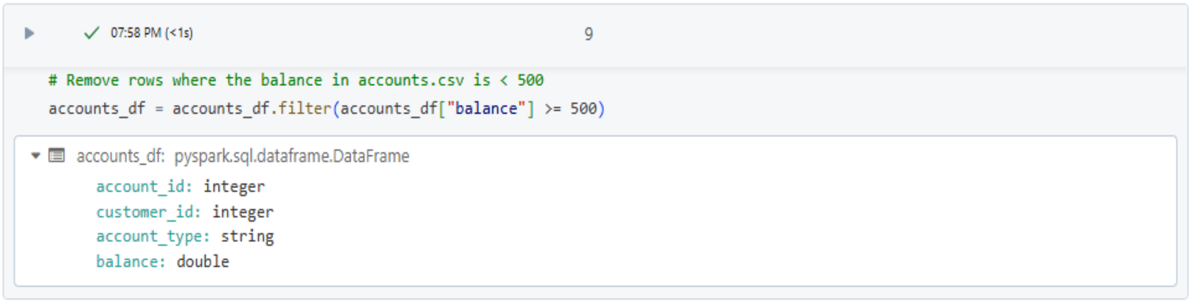
4 minutes ago (2s) 6 Python

```
# Get the number of rows in the DataFrame
row_count = accounts_df.count()
row_count = customers_df.count()
row_count = loan_payments_df.count()
row_count = loans_df.count()
row_count = transactions_df.count()

# Display the number of rows
print(f"The number of rows in customers.csv is: {row_count}")

print(f"The number of rows in accounts.csv is: {row_count}")

print(f"The number of rows in loan_payments.csv is: {row_count}")

print(f"The number of rows in loans.csv is: {row_count}")

print(f"The number of rows in transactions.csv is: {row_count}")
```

(10) Spark Jobs

```
The number of rows in customers.csv is: 100
The number of rows in accounts.csv is: 100
The number of rows in loan_payments.csv is: 100
The number of rows in loans.csv is: 100
The number of rows in transactions.csv is: 100
```

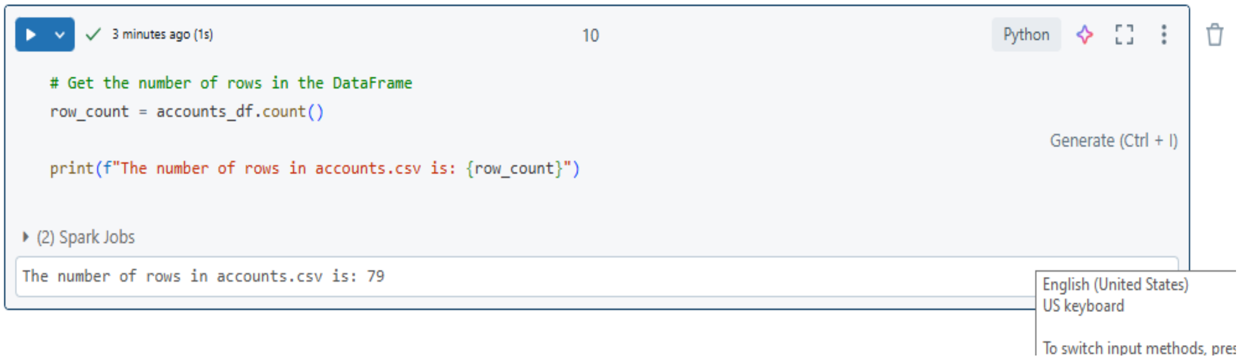
4. **filter out rows from accounts_df** where the balance column is less than 500 and checked rows and make appropriate changes in other tables



07:58 PM (<1s) 9

```
# Remove rows where the balance in accounts.csv is < 500
accounts_df = accounts_df.filter(accounts_df["balance"] >= 500)
```

```
accounts_df: pyspark.sql.dataframe.DataFrame
    account_id: integer
    customer_id: integer
    account_type: string
    balance: double
```



3 minutes ago (1s) 10 Python

```
# Get the number of rows in the DataFrame
row_count = accounts_df.count()

print(f"The number of rows in accounts.csv is: {row_count}")
```

Generate (Ctrl + I)

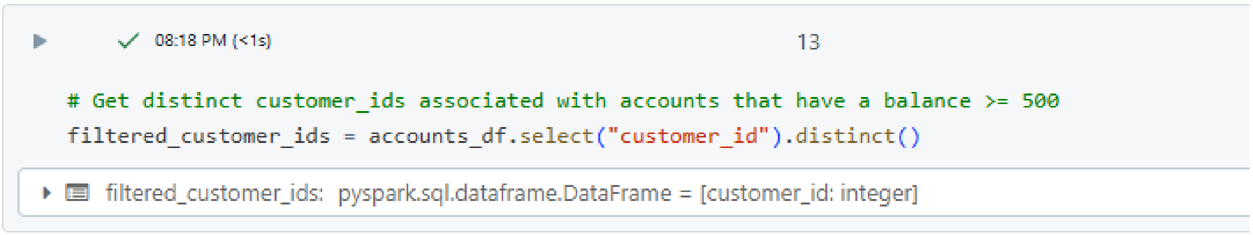
(2) Spark Jobs

```
The number of rows in accounts.csv is: 79
```

English (United States)
US keyboard

To switch input methods, pres

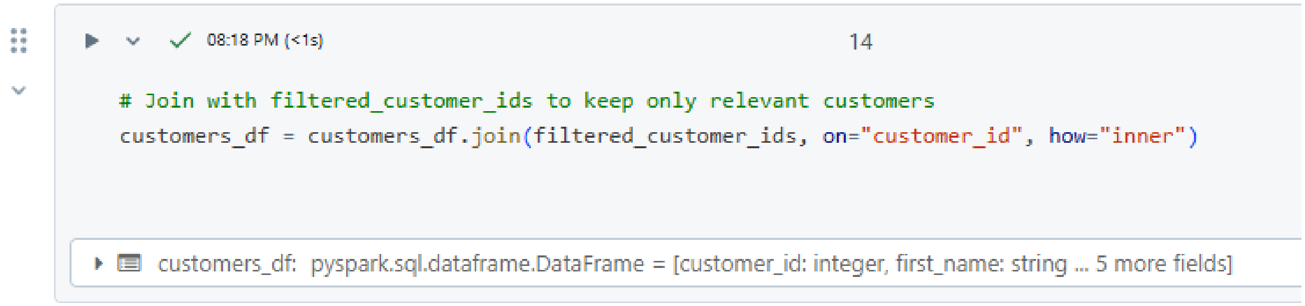
- **Filter the customers_df** to include only the customers who have an account with a balance of 500 or more:



08:18 PM (<1s) 13

```
# Get distinct customer_ids associated with accounts that have a balance >= 500
filtered_customer_ids = accounts_df.select("customer_id").distinct()
```

```
filtered_customer_ids: pyspark.sql.dataframe.DataFrame = [customer_id: integer]
```

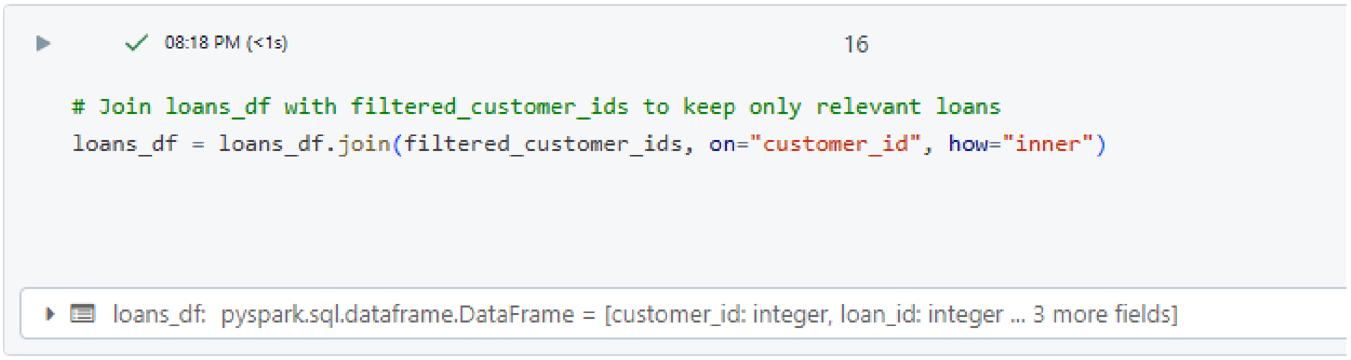


08:18 PM (<1s) 14

```
# Join with filtered_customer_ids to keep only relevant customers
customers_df = customers_df.join(filtered_customer_ids, on="customer_id", how="inner")
```

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

- **Filter the loans_df** so that only loans associated with relevant customer_ids are retained.

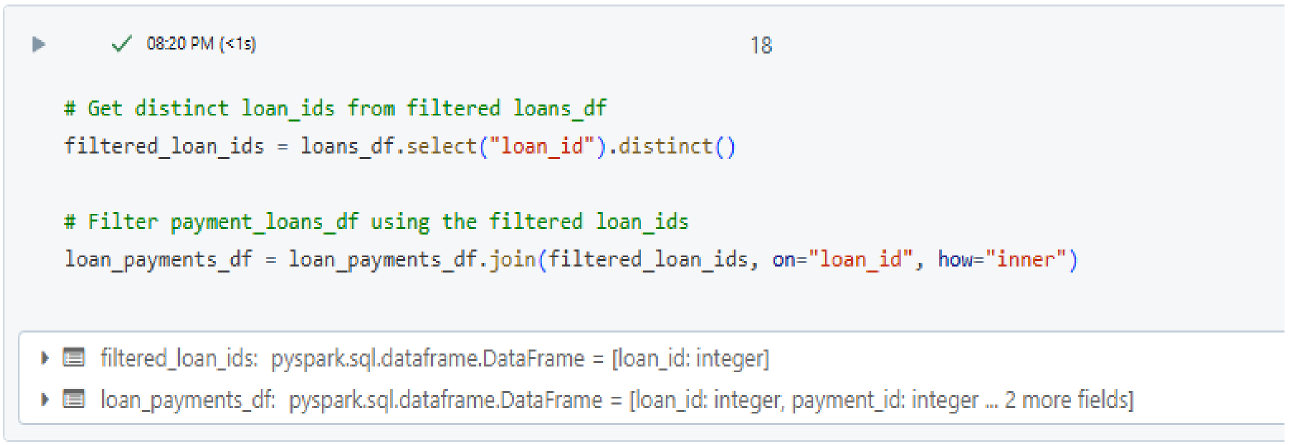


08:18 PM (<1s) 16

```
# Join loans_df with filtered_customer_ids to keep only relevant loans
loans_df = loans_df.join(filtered_customer_ids, on="customer_id", how="inner")
```

```
loans_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, loan_id: integer ... 3 more fields]
```

- **Filtered loan_payments_df**
  Used the filtered loans_df to get relevant loan_ids and filter loan_payments_df accordingly.

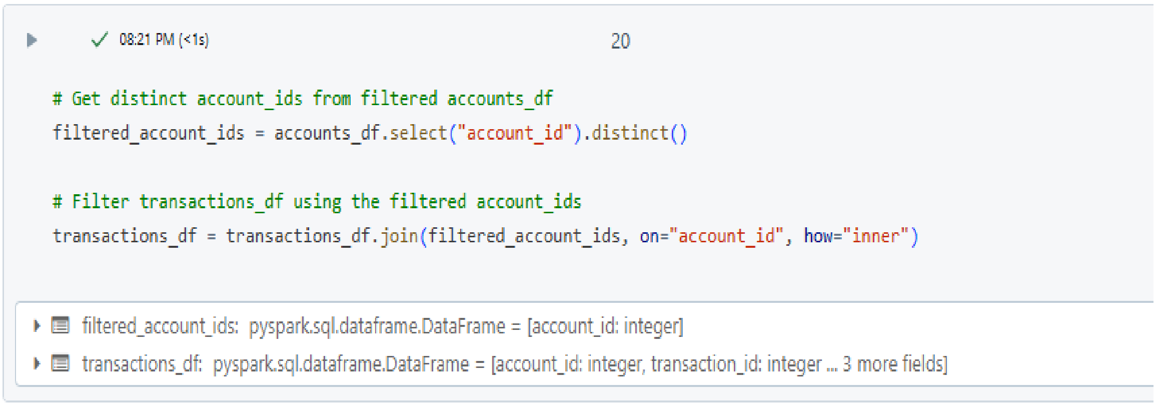


08:20 PM (<1s) 18

```
# Get distinct loan_ids from filtered loans_df
filtered_loan_ids = loans_df.select("loan_id").distinct()

# Filter payment_loans_df using the filtered loan_ids
loan_payments_df = loan_payments_df.join(filtered_loan_ids, on="loan_id", how="inner")
```

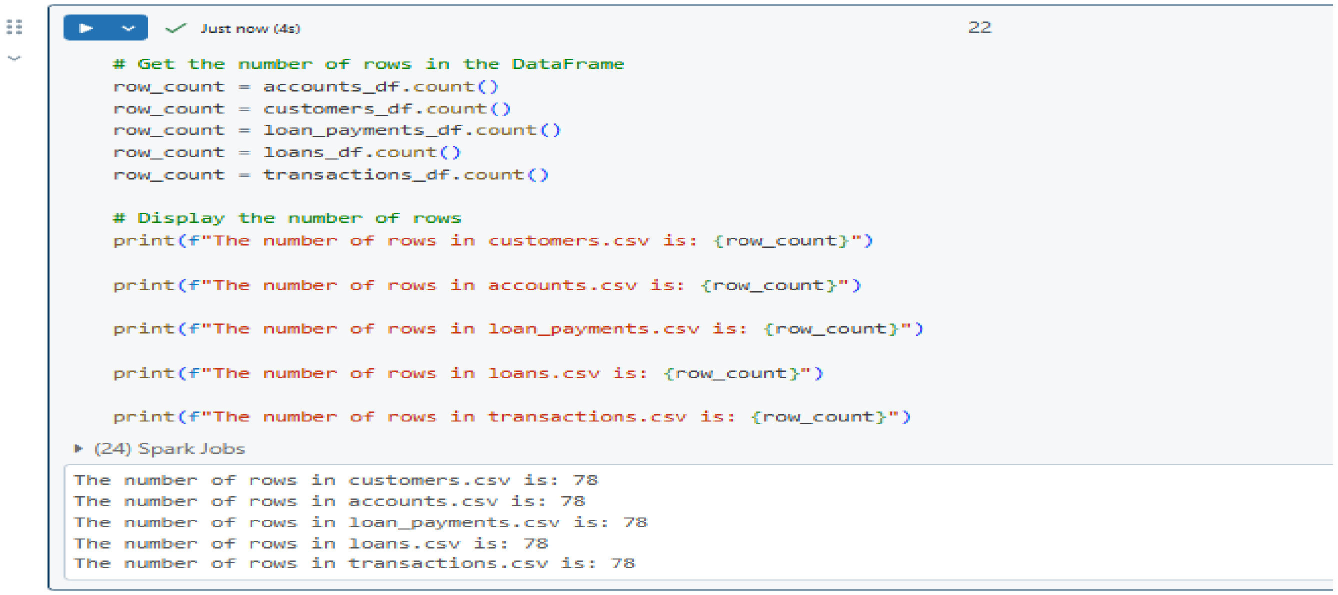
```
filtered_loan_ids:  pyspark.sql.dataframe.DataFrame = [loan_id: integer]
loan_payments_df:  pyspark.sql.dataframe.DataFrame = [loan_id: integer, payment_id: integer ... 2 more fields]
```

- **Filtered transactions_df table**
  Used the filtered account_ids from accounts_df to filter relevant transactions.



```
08:21 PM (<1s)                                   20

# Get distinct account_ids from filtered accounts_df
filtered_account_ids = accounts_df.select("account_id").distinct()

# Filter transactions_df using the filtered account_ids
transactions_df = transactions_df.join(filtered_account_ids, on="account_id", how="inner")

filtered_account_ids:  pyspark.sql.dataframe.DataFrame = [account_id: integer]
transactions_df:  pyspark.sql.dataframe.DataFrame = [account_id: integer, transaction_id: integer ... 3 more fields]
```

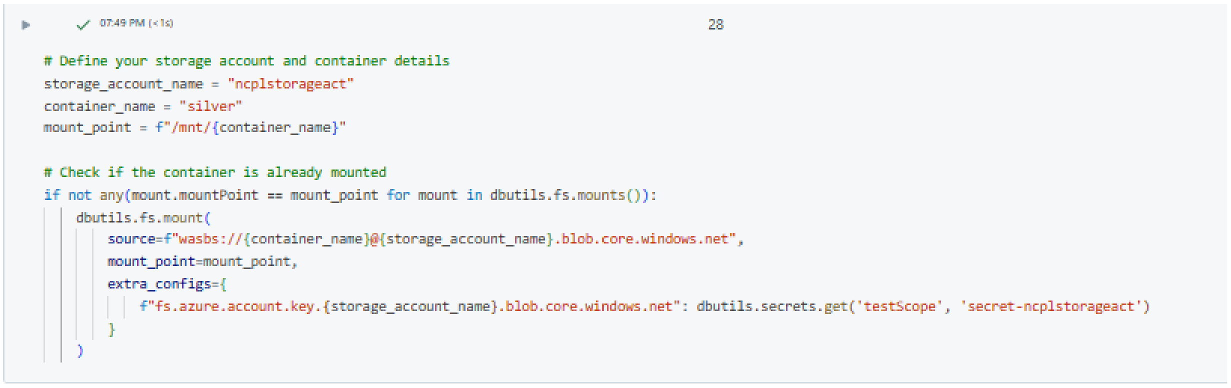
5. Number of rows after data cleaning and transformation: 78 in each table



```
Just now (4s)                                          22

    # Get the number of rows in the DataFrame
    row_count = accounts_df.count()
    row_count = customers_df.count()
    row_count = loan_payments_df.count()
    row_count = loans_df.count()
    row_count = transactions_df.count()

    # Display the number of rows
    print(f"The number of rows in customers.csv is: {row_count}")

    print(f"The number of rows in accounts.csv is: {row_count}")

    print(f"The number of rows in loan_payments.csv is: {row_count}")

    print(f"The number of rows in loans.csv is: {row_count}")

    print(f"The number of rows in transactions.csv is: {row_count}")
(24) Spark Jobs
The number of rows in customers.csv is: 78
The number of rows in accounts.csv is: 78
The number of rows in loan_payments.csv is: 78
The number of rows in loans.csv is: 78
The number of rows in transactions.csv is: 78
```

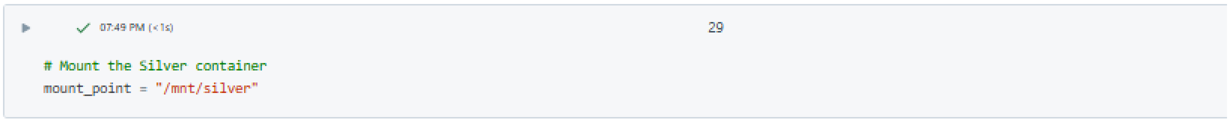
6. Save DataFrames to the Silver Container:

- Mount the Silver Container.
- Save DataFrames to the Silver Container.



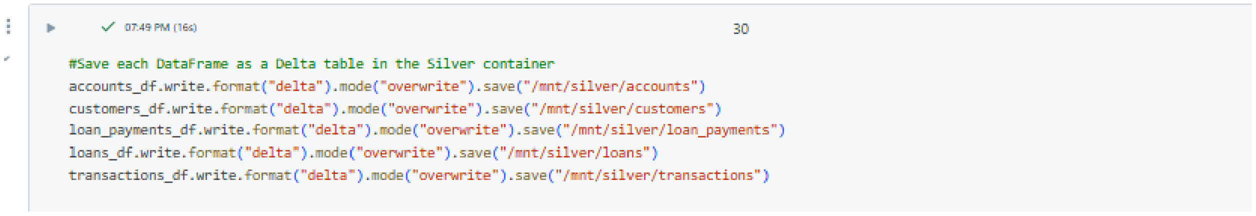
07:49 PM (<1s) 28

```
# Define your storage account and container details
storage_account_name = "ncplstorageact"
container_name = "silver"
mount_point = f"/mnt/{container_name}"

# Check if the container is already mounted
if not any(mount.mountPoint == mount_point for mount in dbutils.fs.mounts()):
    dbutils.fs.mount(
        source=f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",
        mount_point=mount_point,
        extra_configs={
            f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": dbutils.secrets.get('testScope', 'secret-ncplstorageact')
        }
    )
```



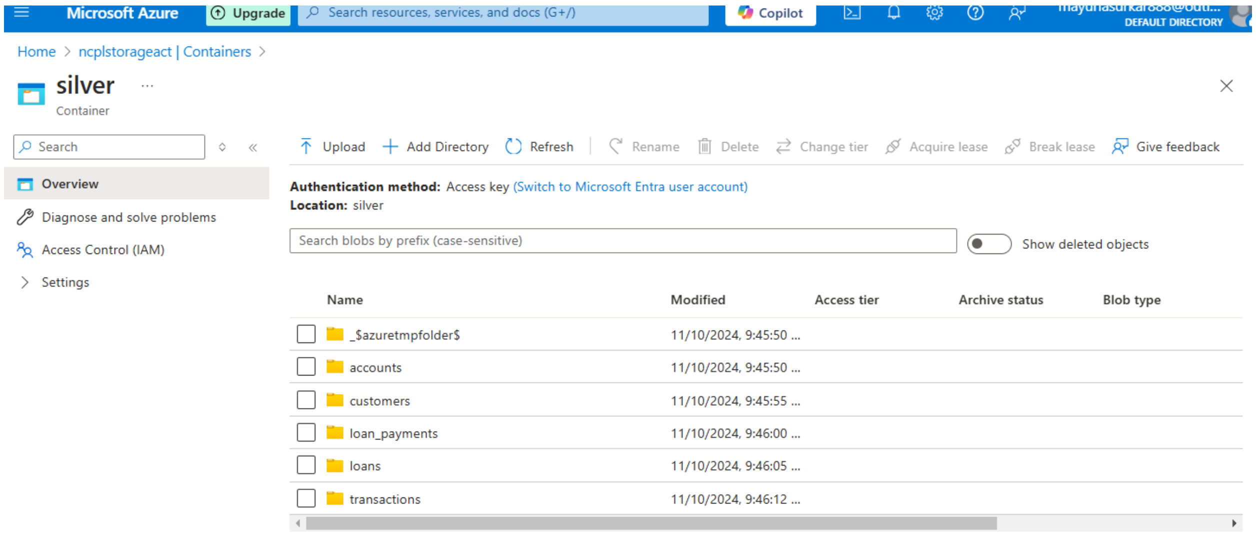
07:49 PM (<1s) 29

```
# Mount the Silver container
mount_point = "/mnt/silver"
```



07:49 PM (16s) 30

```
#Save each DataFrame as a Delta table in the Silver container
accounts_df.write.format("delta").mode("overwrite").save("/mnt/silver/accounts")
customers_df.write.format("delta").mode("overwrite").save("/mnt/silver/customers")
loan_payments_df.write.format("delta").mode("overwrite").save("/mnt/silver/loan_payments")
loans_df.write.format("delta").mode("overwrite").save("/mnt/silver/loans")
transactions_df.write.format("delta").mode("overwrite").save("/mnt/silver/transactions")
```



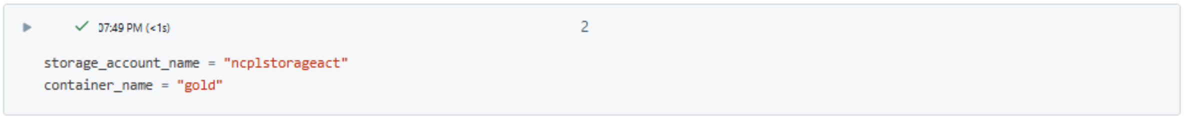
silver
Container

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

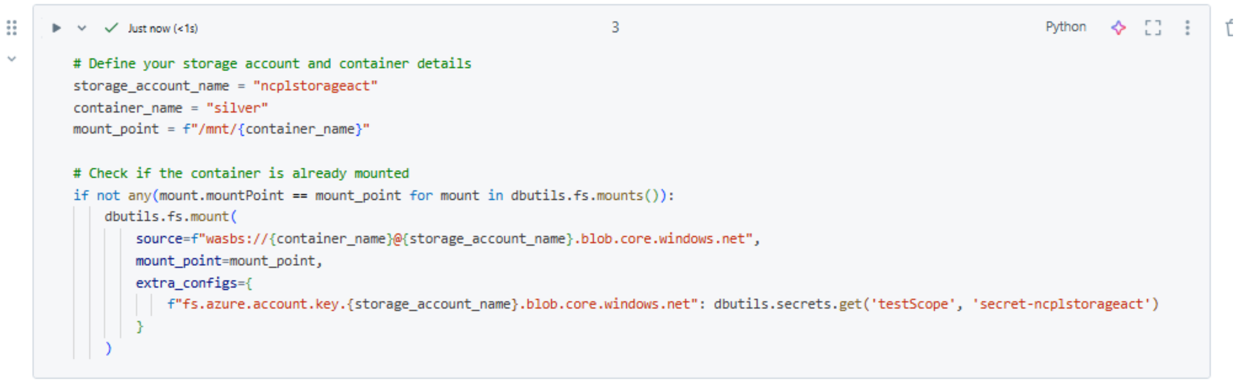
Name	Modified	Access tier	Archive status	Blob type
_$azuretmpfolder$	11/10/2024, 9:45:50 ...			
accounts	11/10/2024, 9:45:50 ...			
customers	11/10/2024, 9:45:55 ...			
loan_payments	11/10/2024, 9:46:00 ...			
loans	11/10/2024, 9:46:05 ...			
transactions	11/10/2024, 9:46:12 ...			

Databricks Activity (ETL Processing)

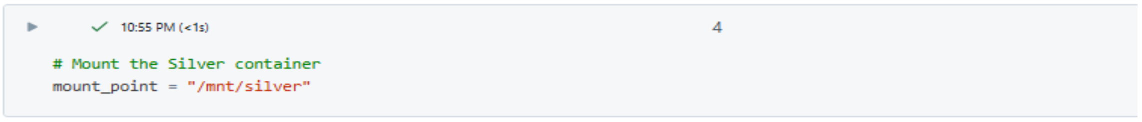
1. Set Up the New Databricks Notebook (Databricks Activity (ETL Processing))
2. Mount the Silver and Gold Containers in Databricks



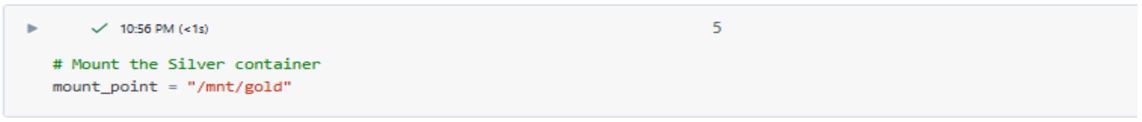
```
storage_account_name = "ncplstorageact"
container_name = "gold"
```



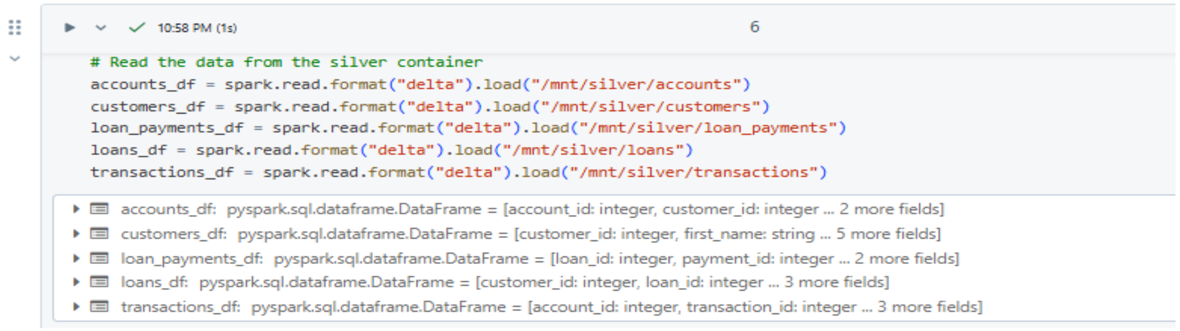
```
# Define your storage account and container details
storage_account_name = "ncplstorageact"
container_name = "silver"
mount_point = f"/mnt/{container_name}"

# Check if the container is already mounted
if not any(mount.mountPoint == mount_point for mount in dbutils.fs.mounts()):
    dbutils.fs.mount(
        source=f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net",
        mount_point=mount_point,
        extra_configs={
            f"fs.azure.account.key.{storage_account_name}.blob.core.windows.net": dbutils.secrets.get('testScope', 'secret-ncplstorageact')
        }
    )
```



```
# Mount the Silver container
mount_point = "/mnt/silver"
```



```
# Mount the Silver container
mount_point = "/mnt/gold"
```

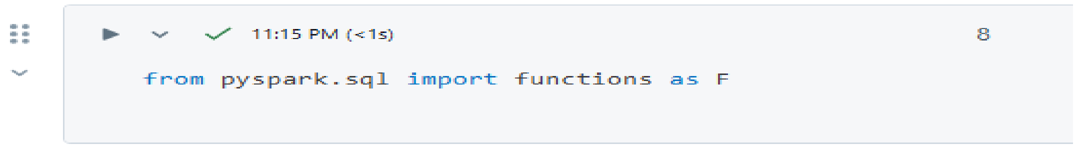


```
# Read the data from the silver container
accounts_df = spark.read.format("delta").load("/mnt/silver/accounts")
customers_df = spark.read.format("delta").load("/mnt/silver/customers")
loan_payments_df = spark.read.format("delta").load("/mnt/silver/loan_payments")
loans_df = spark.read.format("delta").load("/mnt/silver/loans")
transactions_df = spark.read.format("delta").load("/mnt/silver/transactions")
```

```
accounts_df:  pyspark.sql.dataframe.DataFrame = [account_id: integer, customer_id: integer ... 2 more fields]
customers_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 5 more fields]
loan_payments_df:  pyspark.sql.dataframe.DataFrame = [loan_id: integer, payment_id: integer ... 2 more fields]
loans_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, loan_id: integer ... 3 more fields]
transactions_df:  pyspark.sql.dataframe.DataFrame = [account_id: integer, transaction_id: integer ... 3 more fields]
```

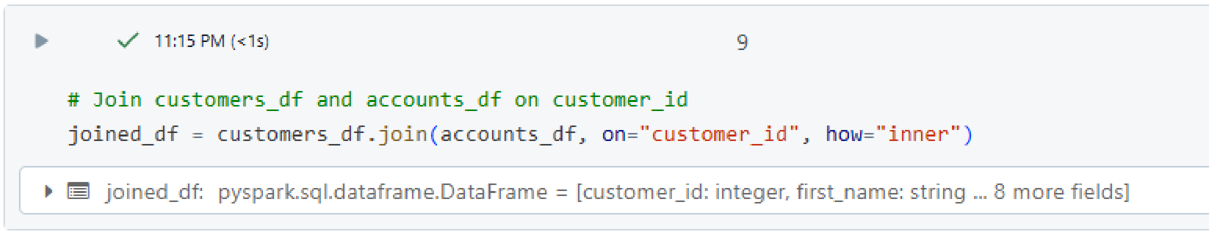
3. Data Transformation (Calculate Total Balance for Each Customer)

- Imported the necessary functions from PySpark.



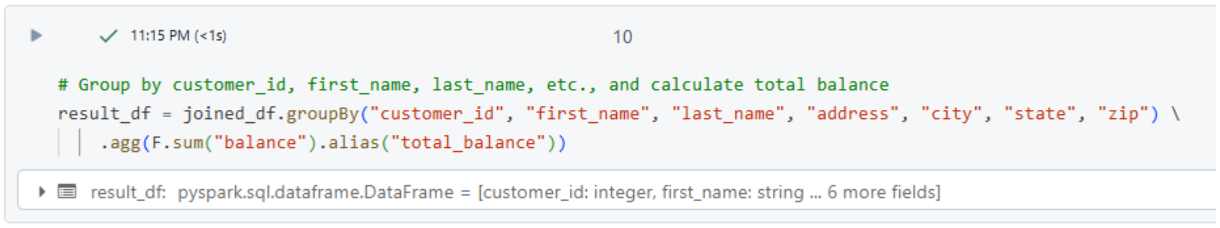
```
11:15 PM (<1s)                                   8

from pyspark.sql import functions as F
```

- Joined the DataFrames on customer_id:



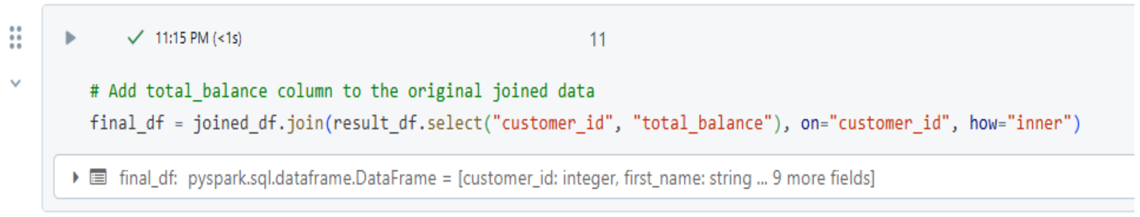
```
11:15 PM (<1s)                                   9

# Join customers_df and accounts_df on customer_id
joined_df = customers_df.join(accounts_df, on="customer_id", how="inner")

joined_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 8 more fields]
```

- Calculate the Total Balance:



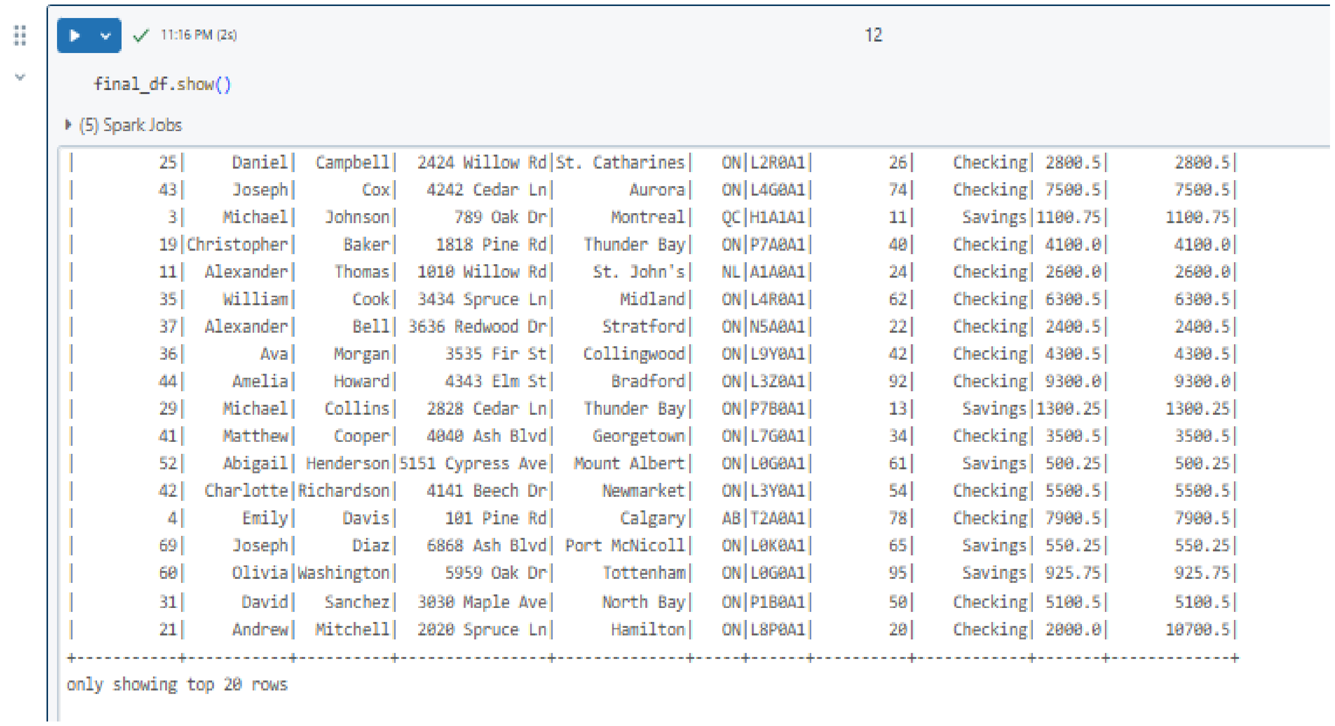
```
11:15 PM (<1s)                                   10

# Group by customer_id, first_name, last_name, etc., and calculate total balance
result_df = joined_df.groupBy("customer_id", "first_name", "last_name", "address", "city", "state", "zip") \
    .agg(F.sum("balance").alias("total_balance"))

result_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 6 more fields]
```

- Include All Columns from accounts_df:



```
11:15 PM (<1s)                                   11

# Add total_balance column to the original joined data
final_df = joined_df.join(result_df.select("customer_id", "total_balance"), on="customer_id", how="inner")

final_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 9 more fields]
```

- Display the Results:



11:16 PM (2s) 12

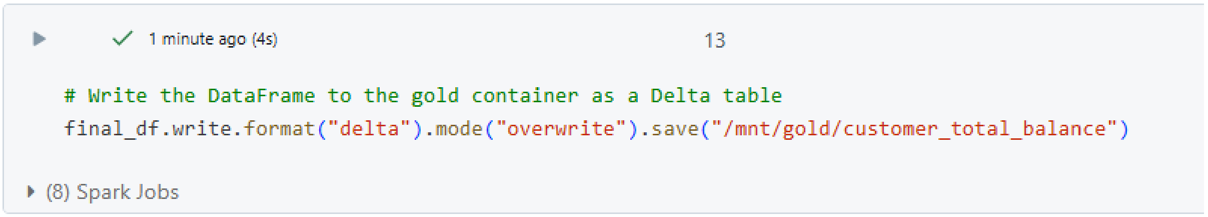
```
final_df.show()
```

(5) Spark Jobs

```
|        25|     Daniel|  Campbell|  2424 Willow Rd|St. Catharines|   ON|L2R0A1|        26|    Checking| 2800.5|       2800.5|
|        43|     Joseph|       Cox|   4242 Cedar Ln|        Aurora|   ON|L4G0A1|        74|    Checking| 7500.5|       7500.5|
|         3|    Michael|   Johnson|      789 Oak Dr|      Montreal|   QC|H1A1A1|        11|     Savings|1100.75|      1100.75|
|        19|Christopher|     Baker|    1818 Pine Rd|   Thunder Bay|   ON|P7A0A1|        40|    Checking| 4100.0|       4100.0|
|        11|  Alexander|    Thomas|  1010 Willow Rd|    St. John's|   NL|A1A0A1|        24|    Checking| 2600.0|       2600.0|
|        35|    William|      Cook|  3434 Spruce Ln|       Midland|   ON|L4R0A1|        62|    Checking| 6300.5|       6300.5|
|        37|  Alexander|      Bell| 3636 Redwood Dr|     Stratford|   ON|N5A0A1|        22|    Checking| 2400.5|       2400.5|
|        36|        Ava|    Morgan|     3535 Fir St|   Collingwood|   ON|L9Y0A1|        42|    Checking| 4300.5|       4300.5|
|        44|     Amelia|    Howard|     4343 Elm St|      Bradford|   ON|L3Z0A1|        92|    Checking| 9300.0|       9300.0|
|        29|    Michael|   Collins|   2828 Cedar Ln|   Thunder Bay|   ON|P7B0A1|        13|     Savings|1300.25|      1300.25|
|        41|    Matthew|    Cooper|   4040 Ash Blvd|    Georgetown|   ON|L7G0A1|        34|    Checking| 3500.5|       3500.5|
|        52|    Abigail| Henderson|5151 Cypress Ave|  Mount Albert|   ON|L0G0A1|        61|     Savings| 500.25|       500.25|
|        42|  Charlotte|Richardson|   4141 Beech Dr|     Newmarket|   ON|L3Y0A1|        54|    Checking| 5500.5|       5500.5|
|         4|      Emily|     Davis|     101 Pine Rd|       Calgary|   AB|T2A0A1|        78|    Checking| 7900.5|       7900.5|
|        69|     Joseph|      Diaz|   6868 Ash Blvd| Port McNicoll|   ON|L0K0A1|        65|     Savings| 550.25|       550.25|
|        60|     Olivia|Washington|     5959 Oak Dr|     Tottenham|   ON|L0G0A1|        95|     Savings| 925.75|       925.75|
|        31|      David|   Sanchez|  3030 Maple Ave|     North Bay|   ON|P1B0A1|        50|    Checking| 5100.5|       5100.5|
|        21|     Andrew|  Mitchell|  2020 Spruce Ln|      Hamilton|   ON|L8P0A1|        20|    Checking| 2000.0|      10700.5|
+----------+-----------+----------+----------------+--------------+-----+------+----------+------------+-------+-------------+
only showing top 20 rows
```

4. **Load: Saved the transformed data into the gold container.**

- Save the transformed data to the **gold** container.

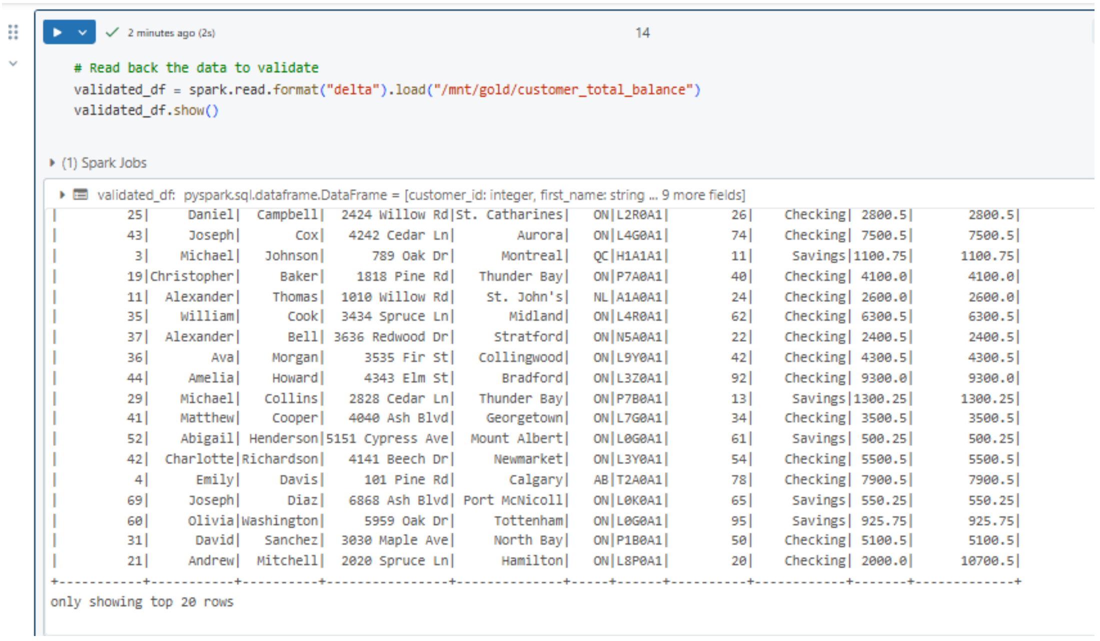


1 minute ago (4s) 13

```
# Write the DataFrame to the gold container as a Delta table
final_df.write.format("delta").mode("overwrite").save("/mnt/gold/customer_total_balance")
```

(8) Spark Jobs

- Validate the Data in the Gold Container

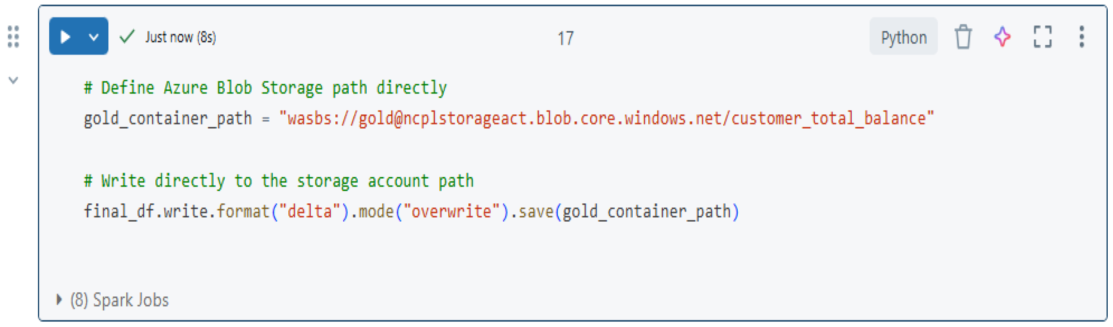


```
# Read back the data to validate
validated_df = spark.read.format("delta").load("/mnt/gold/customer_total_balance")
validated_df.show()
```

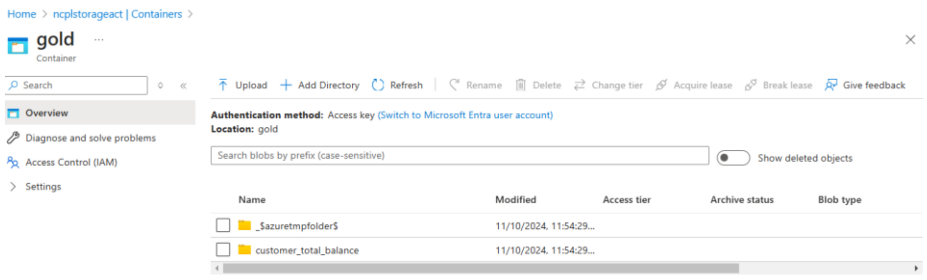
validated_df: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 9 more fields]

```
|         25|     Daniel|  Campbell|  2424 Willow Rd|St. Catharines|   ON|L2R0A1|        26|   Checking| 2800.5|      2800.5|
|         43|     Joseph|       Cox|   4242 Cedar Ln|        Aurora|   ON|L4G0A1|        74|   Checking| 7500.5|      7500.5|
|          3|    Michael|   Johnson|      789 Oak Dr|      Montreal|   QC|H1A1A1|        11|    Savings|1100.75|     1100.75|
|         19|Christopher|     Baker|    1818 Pine Rd|   Thunder Bay|   ON|P7A0A1|        40|   Checking| 4100.0|      4100.0|
|         11|  Alexander|    Thomas|  1010 Willow Rd|    St. John's|   NL|A1A0A1|        24|   Checking| 2600.0|      2600.0|
|         35|    William|      Cook|  3434 Spruce Ln|       Midland|   ON|L4R0A1|        62|   Checking| 6300.5|      6300.5|
|         37|  Alexander|      Bell| 3636 Redwood Dr|     Stratford|   ON|N5A0A1|        22|   Checking| 2400.5|      2400.5|
|         36|        Ava|    Morgan|     3535 Fir St|   Collingwood|   ON|L9Y0A1|        42|   Checking| 4300.5|      4300.5|
|         44|     Amelia|    Howard|     4343 Elm St|      Bradford|   ON|L3Z0A1|        92|   Checking| 9300.0|      9300.0|
|         29|    Michael|   Collins|   2828 Cedar Ln|   Thunder Bay|   ON|P7B0A1|        13|    Savings|1300.25|     1300.25|
|         41|    Matthew|    Cooper|   4040 Ash Blvd|    Georgetown|   ON|L7G0A1|        34|   Checking| 3500.5|      3500.5|
|         52|    Abigail| Henderson|5151 Cypress Ave|  Mount Albert|   ON|L0G0A1|        61|    Savings| 500.25|      500.25|
|         42|  Charlotte|Richardson|   4141 Beech Dr|     Newmarket|   ON|L3Y0A1|        54|   Checking| 5500.5|      5500.5|
|          4|      Emily|     Davis|     101 Pine Rd|       Calgary|   AB|T2A0A1|        78|   Checking| 7900.5|      7900.5|
|         69|     Joseph|      Diaz|   6868 Ash Blvd| Port McNicoll|   ON|L0K0A1|        65|    Savings| 550.25|      550.25|
|         60|     Olivia|Washington|     5959 Oak Dr|     Tottenham|   ON|L0G0A1|        95|    Savings| 925.75|      925.75|
|         31|      David|   Sanchez|  3030 Maple Ave|     North Bay|   ON|P1B0A1|        50|   Checking| 5100.5|      5100.5|
|         21|     Andrew|  Mitchell|  2020 Spruce Ln|      Hamilton|   ON|L8P0A1|        20|   Checking| 2000.0|     10700.5|
+-----------+-----------+----------+----------------+--------------+-----+------+----------+-----------+-------+------------+
only showing top 20 rows
```

- Write Directly to Storage Path



```
# Define Azure Blob Storage path directly
gold_container_path = "wasbs://gold@ncplstorageact.blob.core.windows.net/customer_total_balance"

# Write directly to the storage account path
final_df.write.format("delta").mode("overwrite").save(gold_container_path)
```



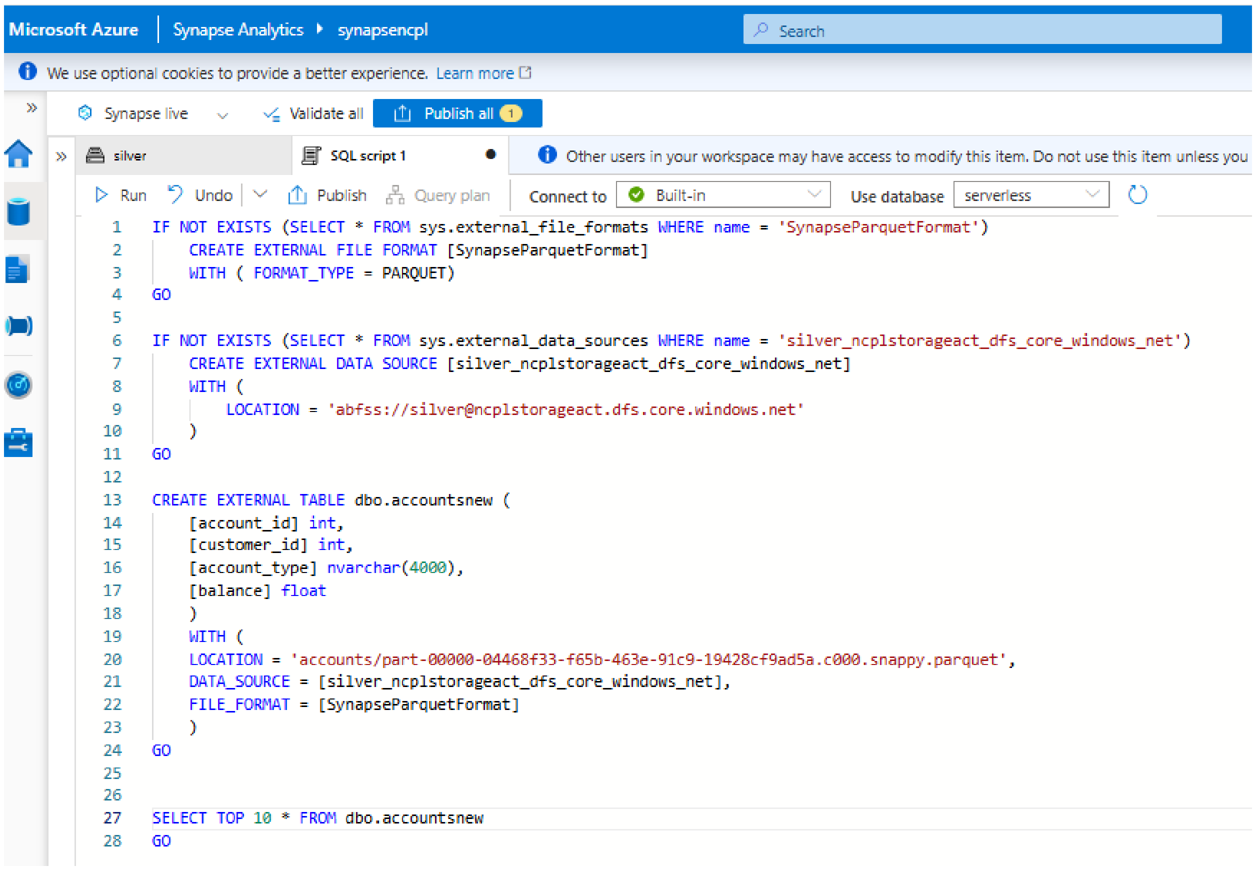
Authentication method: Access key (Switch to Microsoft Entra user account)
Location: gold

Name	Modified	Access tier	Archive status	Blob type
$azuretmpfolder$	11/10/2024, 11:54:29...			
customer_total_balance	11/10/2024, 11:54:29...			

Azure Synapse Analytics

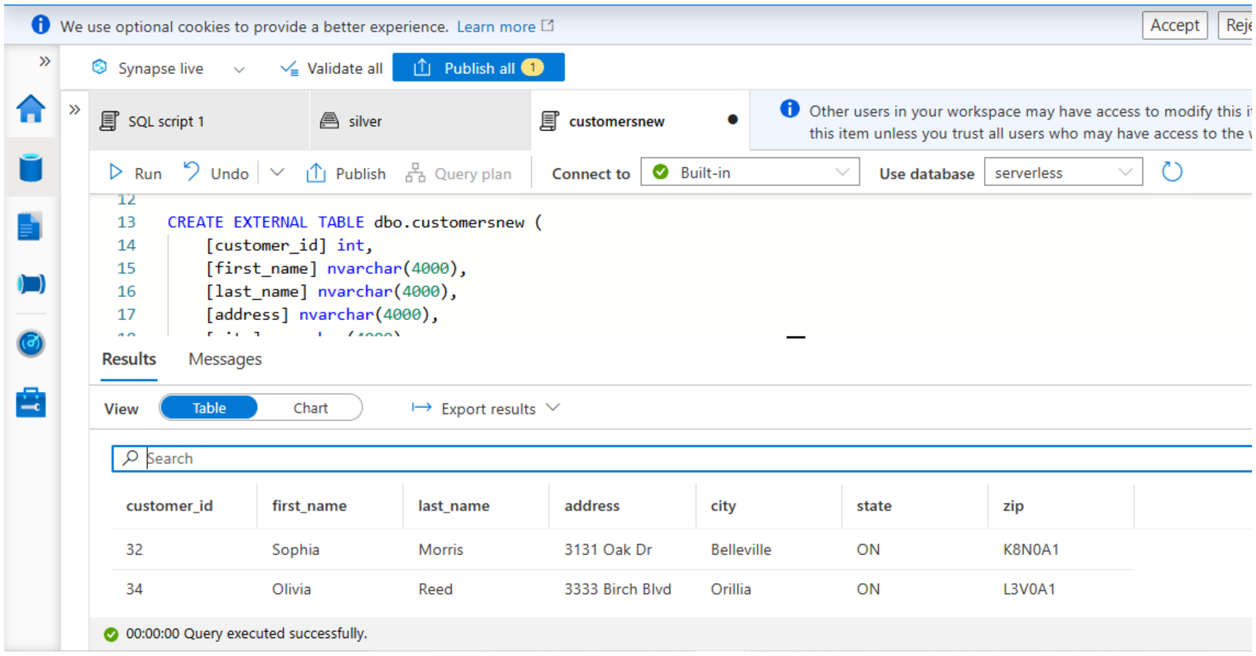
1. Create external tables to map to the data stored in the Curated(silver).

- For accounts (accountsnew):



```sql
IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormat')
    CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
    WITH ( FORMAT_TYPE = PARQUET)
GO

IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'silver_ncplstorageact_dfs_core_windows_net')
    CREATE EXTERNAL DATA SOURCE [silver_ncplstorageact_dfs_core_windows_net]
    WITH (
        LOCATION = 'abfss://silver@ncplstorageact.dfs.core.windows.net'
    )
GO

CREATE EXTERNAL TABLE dbo.accountsnew (
    [account_id] int,
    [customer_id] int,
    [account_type] nvarchar(4000),
    [balance] float
    )
    WITH (
    LOCATION = 'accounts/part-00000-04468f33-f65b-463e-91c9-19428cf9ad5a.c000.snappy.parquet',
    DATA_SOURCE = [silver_ncplstorageact_dfs_core_windows_net],
    FILE_FORMAT = [SynapseParquetFormat]
    )
GO


SELECT TOP 10 * FROM dbo.accountsnew
GO
```

- For customers (customersnew)

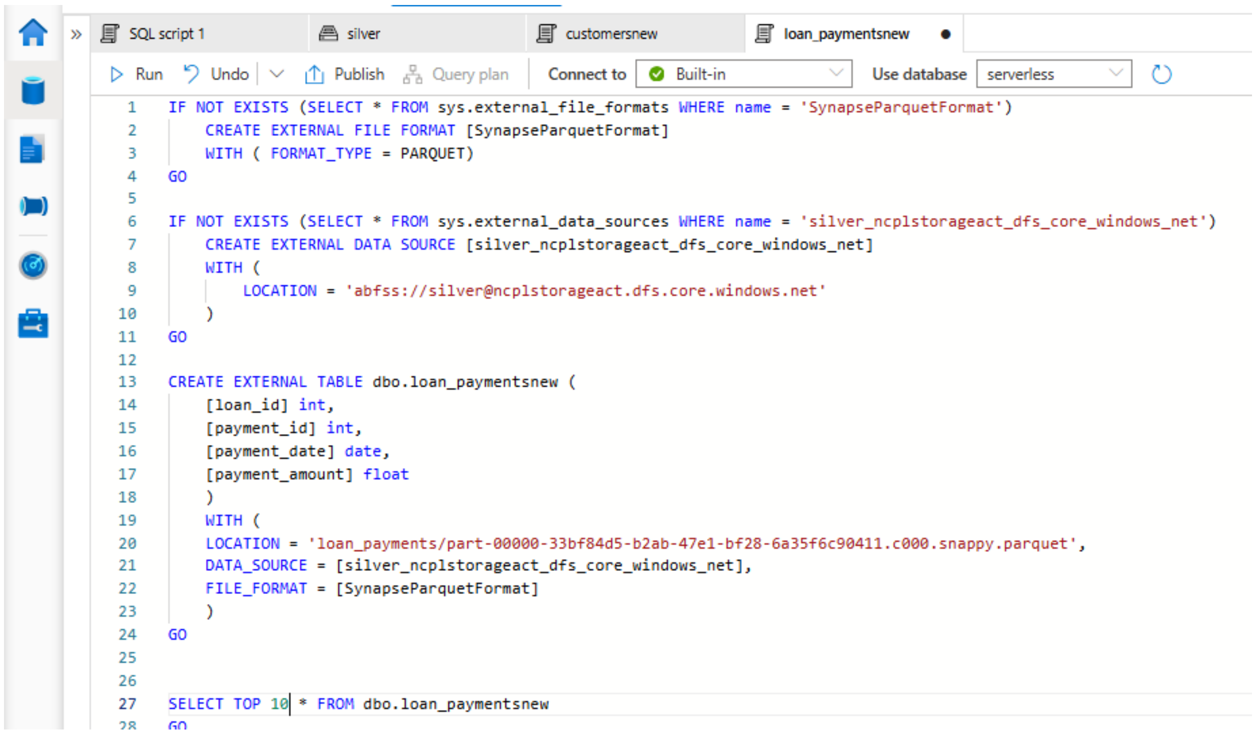

We use optional cookies to provide a better experience. Learn more
Accept
Reje
Synapse live
Validate all
Publish all 1
SQL script 1
silver
customersnew
Other users in your workspace may have access to modify this i
this item unless you trust all users who may have access to the w
Run
Undo
Publish
Query plan
Connect to
Built-in
Use database
serverless
Results
Messages
View
Table
Chart
Export results
Search

```sql
CREATE EXTERNAL TABLE dbo.customersnew (
    [customer_id] int,
    [first_name] nvarchar(4000),
    [last_name] nvarchar(4000),
    [address] nvarchar(4000),
```

customer_id	first_name	last_name	address	city	state	zip
32	Sophia	Morris	3131 Oak Dr	Belleville	ON	K8N0A1
34	Olivia	Reed	3333 Birch Blvd	Orillia	ON	L3V0A1

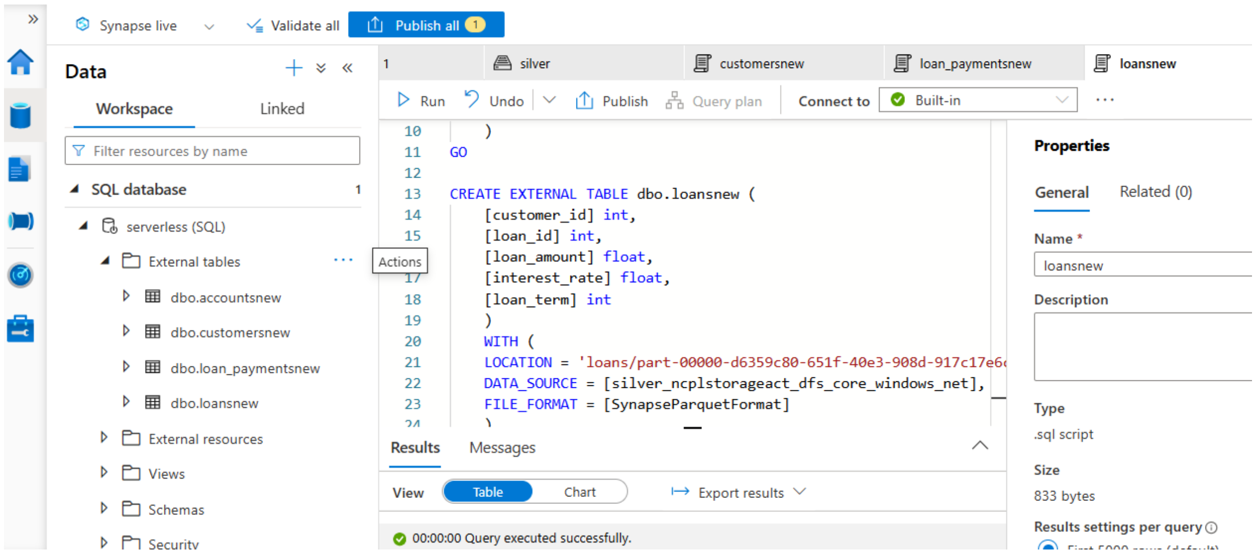
00:00:00 Query executed successfully.

- For loan_payments (loan_paymentsnew)



```sql
IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormat')
    CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
    WITH ( FORMAT_TYPE = PARQUET)
GO

IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'silver_ncplstorageact_dfs_core_windows_net')
    CREATE EXTERNAL DATA SOURCE [silver_ncplstorageact_dfs_core_windows_net]
    WITH (
        LOCATION = 'abfss://silver@ncplstorageact.dfs.core.windows.net'
    )
GO

CREATE EXTERNAL TABLE dbo.loan_paymentsnew (
    [loan_id] int,
    [payment_id] int,
    [payment_date] date,
    [payment_amount] float
    )
    WITH (
    LOCATION = 'loan_payments/part-00000-33bf84d5-b2ab-47e1-bf28-6a35f6c90411.c000.snappy.parquet',
    DATA_SOURCE = [silver_ncplstorageact_dfs_core_windows_net],
    FILE_FORMAT = [SynapseParquetFormat]
    )
GO


SELECT TOP 10 * FROM dbo.loan_paymentsnew
GO
```

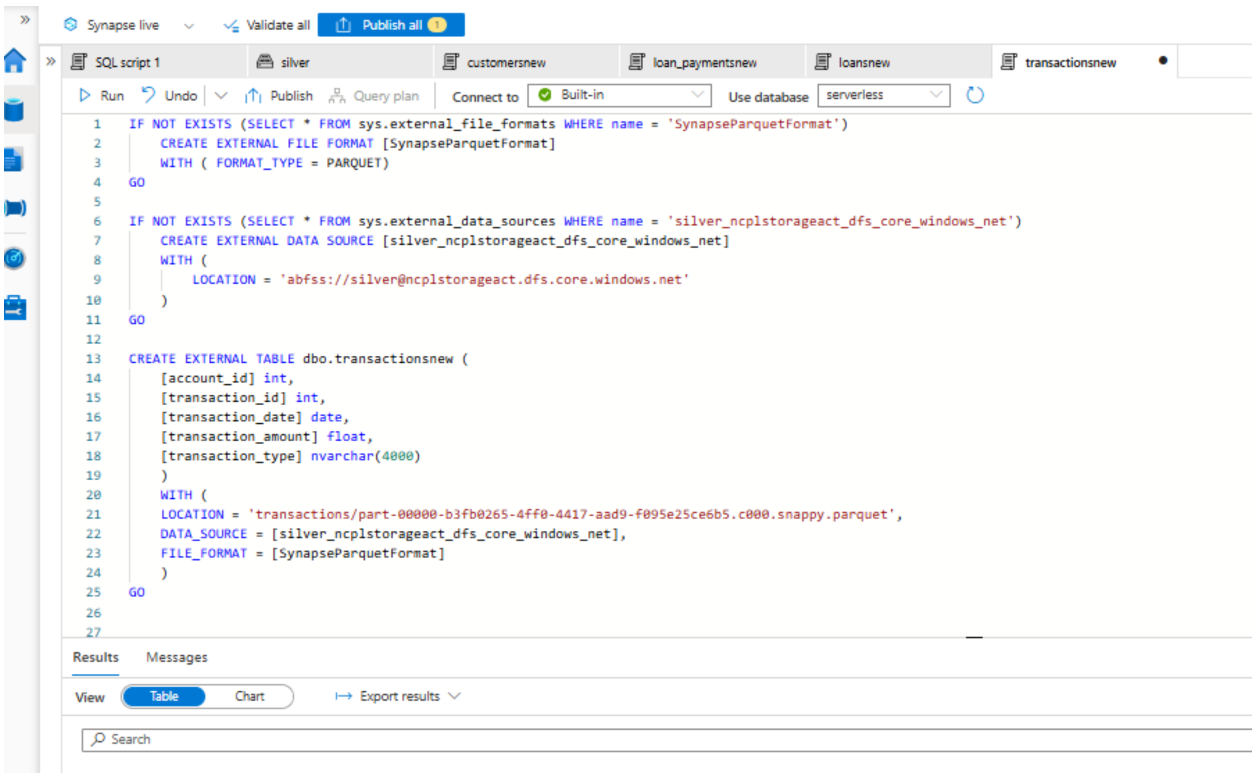
- For loans (loansnew)



```
    )
GO

CREATE EXTERNAL TABLE dbo.loansnew (
    [customer_id] int,
    [loan_id] int,
    [loan_amount] float,
    [interest_rate] float,
    [loan_term] int
    )
    WITH (
    LOCATION = 'loans/part-00000-d6359c80-651f-40e3-908d-917c17e6
    DATA_SOURCE = [silver_ncplstorageact_dfs_core_windows_net],
    FILE_FORMAT = [SynapseParquetFormat]
    )
```

00:00:00 Query executed successfully.

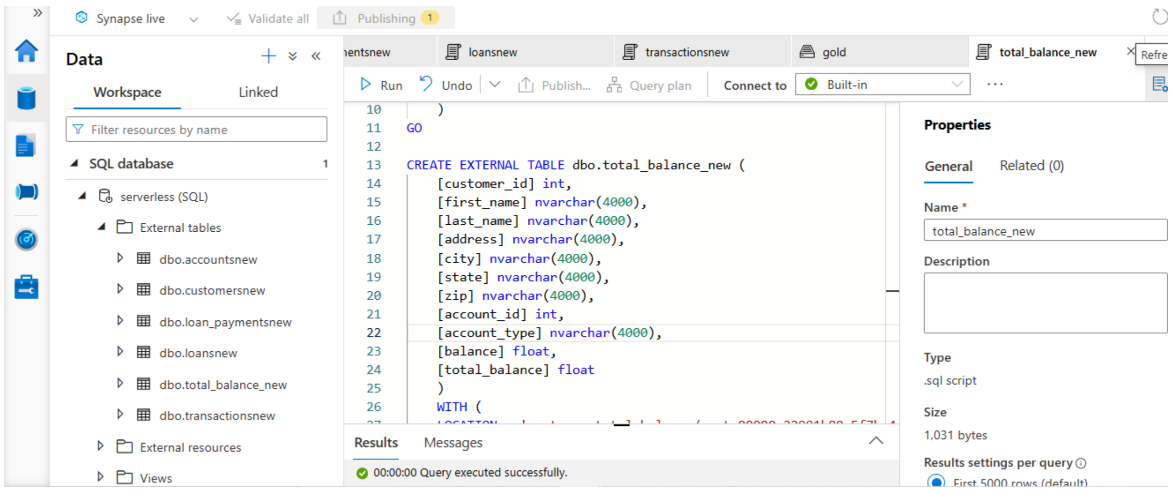
- Transactions (transactionsnew)



```
IF NOT EXISTS (SELECT * FROM sys.external_file_formats WHERE name = 'SynapseParquetFormat')
    CREATE EXTERNAL FILE FORMAT [SynapseParquetFormat]
    WITH ( FORMAT_TYPE = PARQUET)
GO

IF NOT EXISTS (SELECT * FROM sys.external_data_sources WHERE name = 'silver_ncplstorageact_dfs_core_windows_net')
    CREATE EXTERNAL DATA SOURCE [silver_ncplstorageact_dfs_core_windows_net]
    WITH (
        LOCATION = 'abfss://silver@ncplstorageact.dfs.core.windows.net'
    )
GO

CREATE EXTERNAL TABLE dbo.transactionsnew (
    [account_id] int,
    [transaction_id] int,
    [transaction_date] date,
    [transaction_amount] float,
    [transaction_type] nvarchar(4000)
    )
    WITH (
    LOCATION = 'transactions/part-00000-b3fb0265-4ff0-4417-aad9-f095e25ce6b5.c000.snappy.parquet',
    DATA_SOURCE = [silver_ncplstorageact_dfs_core_windows_net],
    FILE_FORMAT = [SynapseParquetFormat]
    )
GO
```

2. Create external tables to map to the data stored in the gold

- Customer_total_balance (Total_balance_new)



```
    )
GO

CREATE EXTERNAL TABLE dbo.total_balance_new (
    [customer_id] int,
    [first_name] nvarchar(4000),
    [last_name] nvarchar(4000),
    [address] nvarchar(4000),
    [city] nvarchar(4000),
    [state] nvarchar(4000),
    [zip] nvarchar(4000),
    [account_id] int,
    [account_type] nvarchar(4000),
    [balance] float,
    [total_balance] float
    )
    WITH (
```

00:00:00 Query executed successfully.

Properties

Name *: total_balance_new

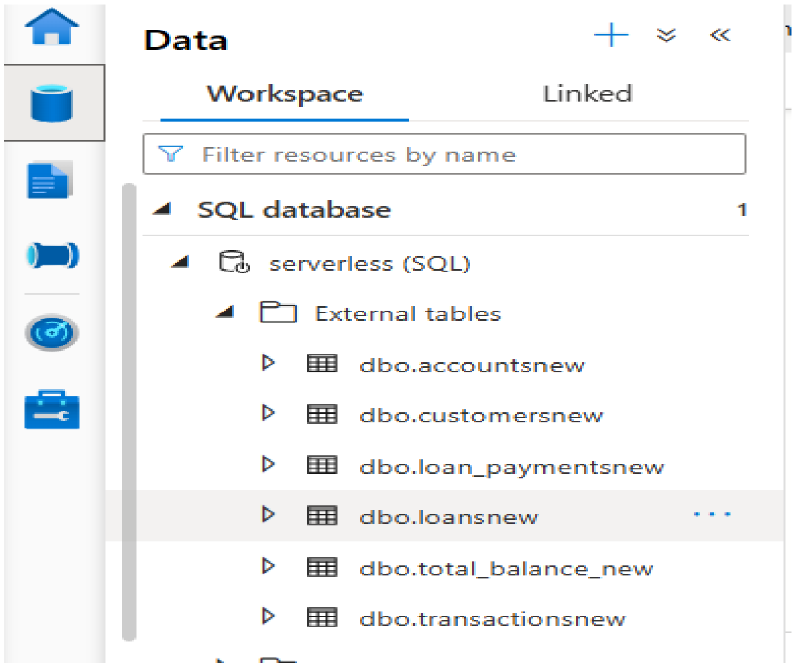
Description:

Type: .sql script

Size: 1,031 bytes

Results settings per query: First 5000 rows (default)

- List of all external tables un synapse under workspace under serverless database:
  This allows data analysts and business intelligence teams to access and query the data directly using tools like Synapse Studio or notebooks.



Data
Workspace
Linked
Filter resources by name
SQL database 1
serverless (SQL)
External tables
dbo.accountsnew
dbo.customersnew
dbo.loan_paymentsnew
dbo.loansnew
dbo.total_balance_new
dbo.transactionsnew