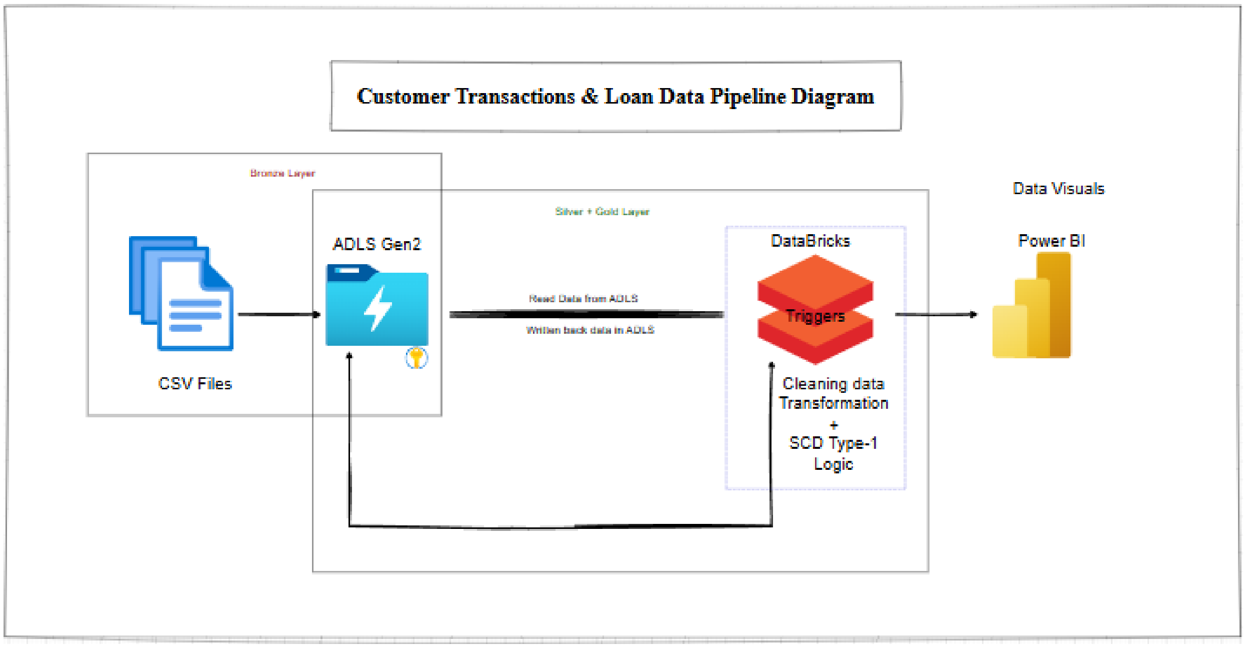
Project 2 - Transactions and Loan Data for a Customer

Objective

The goal of this project is to build an end-to-end data pipeline to process customer account data. We move raw files from ADLS Gen2 (Bronze layer), clean and transform them in Databricks (Silver layer), and finally manage the records into SCD Type-1 Gold Delta tables in ADLS Gen2. The processed data is visualized through Power BI for actionable insights.

Diagram



Data Ingestion (Bronze Layer)

- **Source:** Backend Storage (5 files)
  - accounts.csv
  - customers.csv
  - loans.csv
  - loan_payments.csv
  - transactions.csv
- **Sink:** ADLS Gen2 Bronze container (/mnt/project2/bronze/)

- **Mount:** Databricks connected to ADLS Gen2 using a Mount Point with OAuth (Key Vault for secret management).

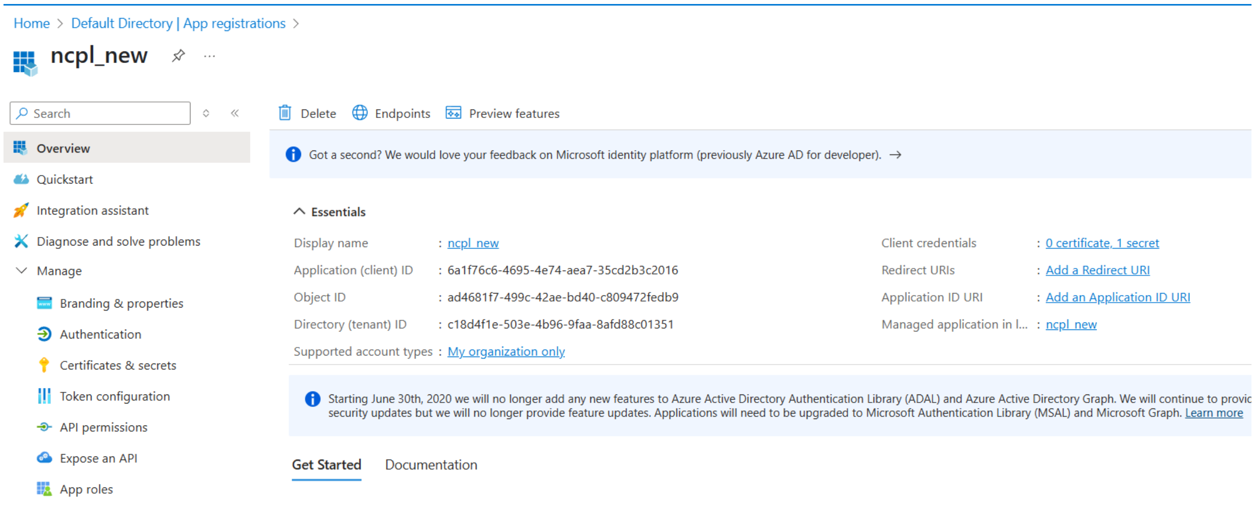
Mount ADLS Gen2 to Databricks

Connect Azure Data Lake Storage (ADLS) to Databricks using secure OAuth authentication and mount it at /mnt/project2.

For the setup

First go to Microsoft-Entra-id ------Manage----App Registration ----new app ---

We need Client Id and Tenet ID for Connection



Got a second? We would love your feedback on Microsoft identity platform (previously Azure AD for developer). →

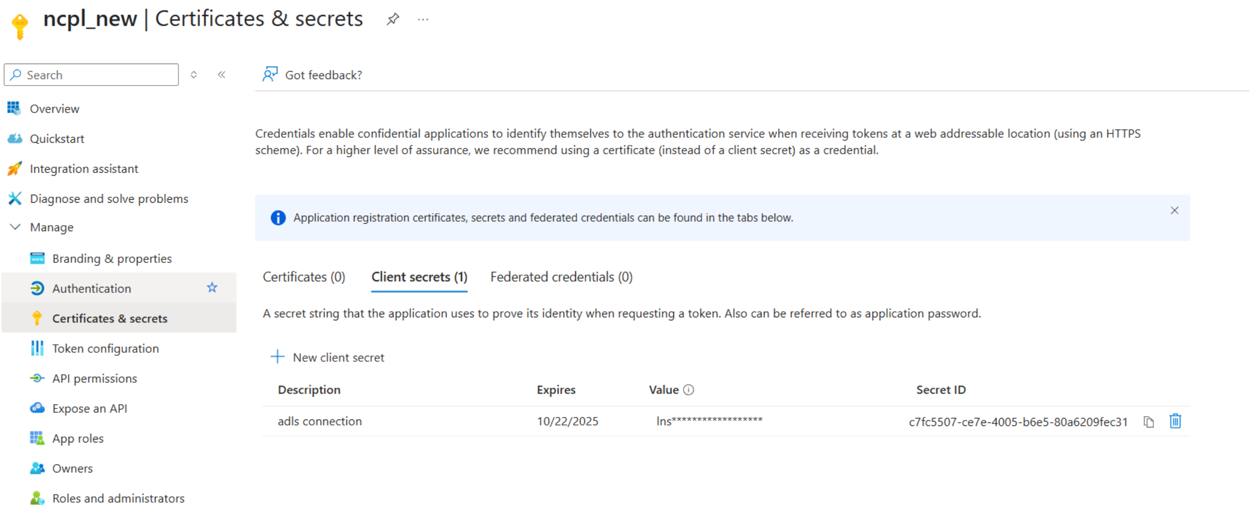
Essentials

Display name	: ncpl_new	Client credentials	: 0 certificate, 1 secret
Application (client) ID	: 6a1f76c6-4695-4e74-aea7-35cd2b3c2016	Redirect URIs	: Add a Redirect URI
Object ID	: ad4681f7-499c-42ae-bd40-c809472fedb9	Application ID URI	: Add an Application ID URI
Directory (tenant) ID	: c18d4f1e-503e-4b96-9faa-8afd88c01351	Managed application in l...	: ncpl_new
Supported account types	: My organization only		

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provic security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

In Manage ----Certificate & Secrets ---create new client Secrets---provide validation dates

Here we require value and Secret Id for connection as well



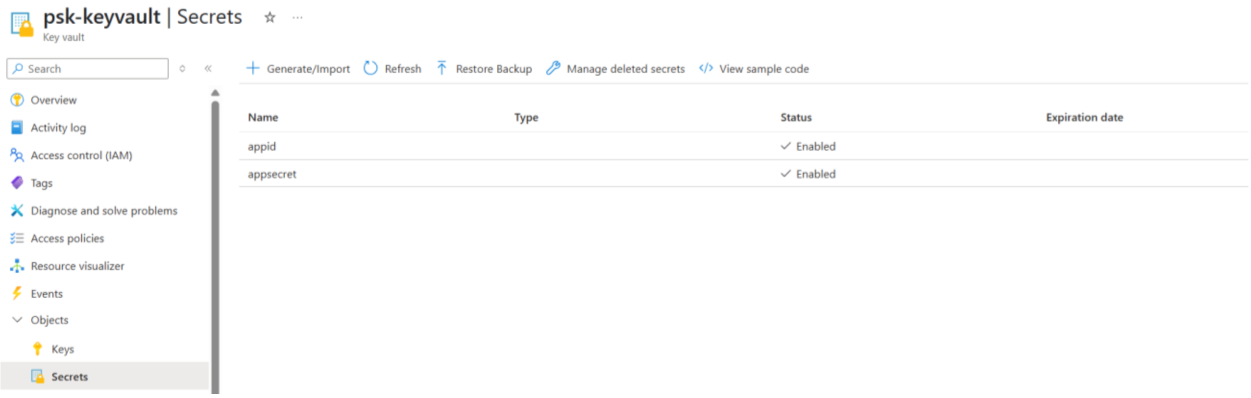
Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

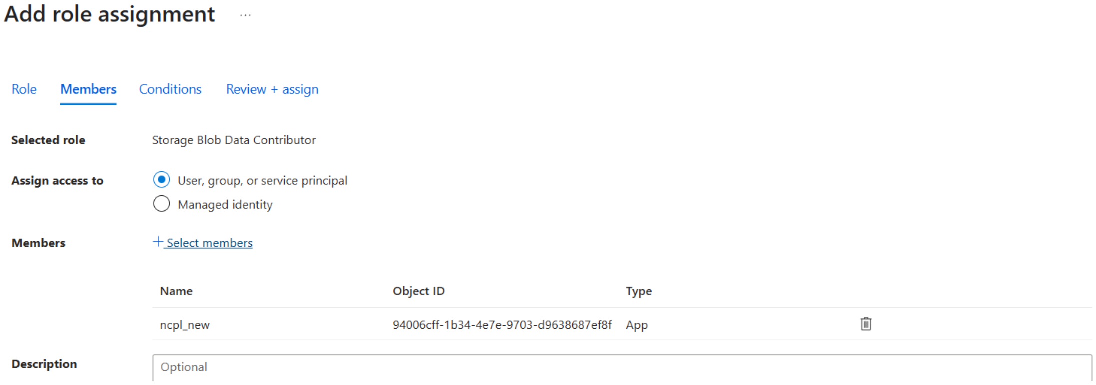
Description	Expires	Value	Secret ID
adls connection	10/22/2025	Ins*****************	c7fc5507-ce7e-4005-b6e5-80a6209fec31

Go to key vault ----objects ---Secrets---create appid & appsecrets



Name	Type	Status	Expiration date
appid		✓ Enabled	
appsecret		✓ Enabled	

Go to ADLS Gen 2 storage account ---Access Control ---add role ---select Storage Blob Data Contributor----add Registered App



Selected role: Storage Blob Data Contributor

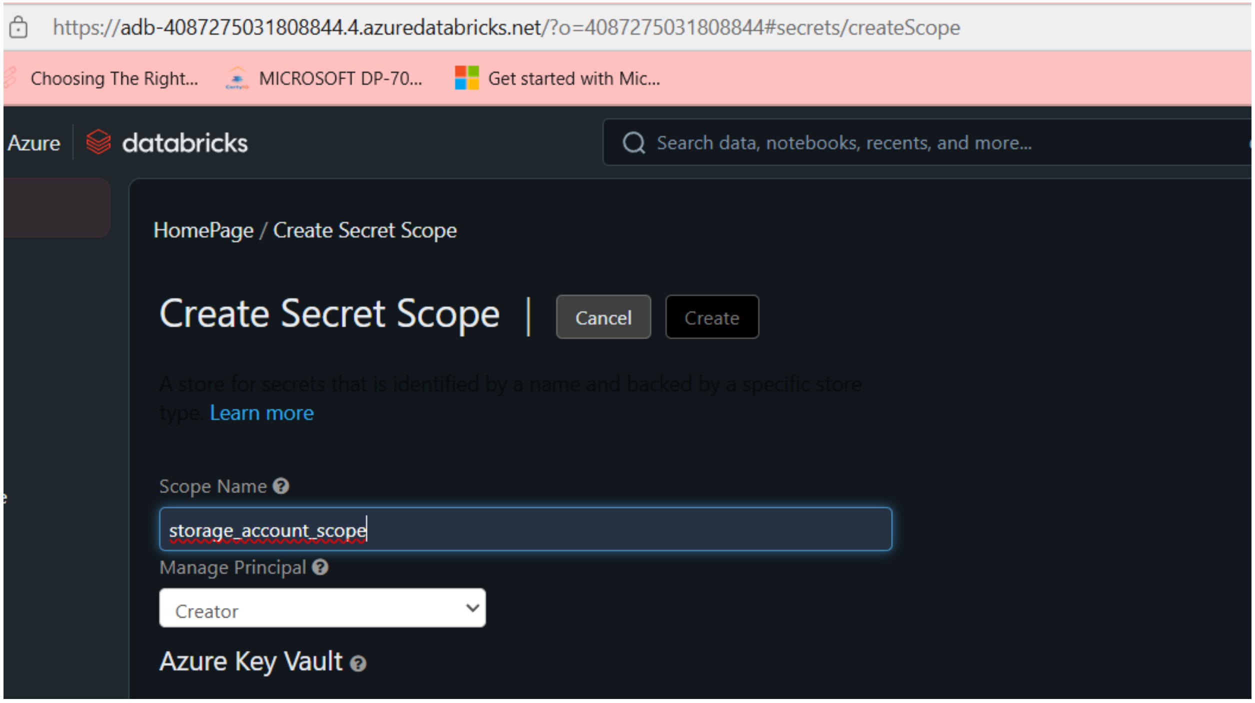
Assign access to: ● User, group, or service principal ○ Managed identity

Members

Name	Object ID	Type
ncpl_new	94006cff-1b34-4e7e-9703-d9638687ef8f	App

Description: Optional

Now we have to create Scope so after the .net/#secrets/createScope type in link bar



https://adb-4087275031808844.4.azuredatabricks.net/?o=4087275031808844#secrets/createScope

HomePage / Create Secret Scope

Create Secret Scope

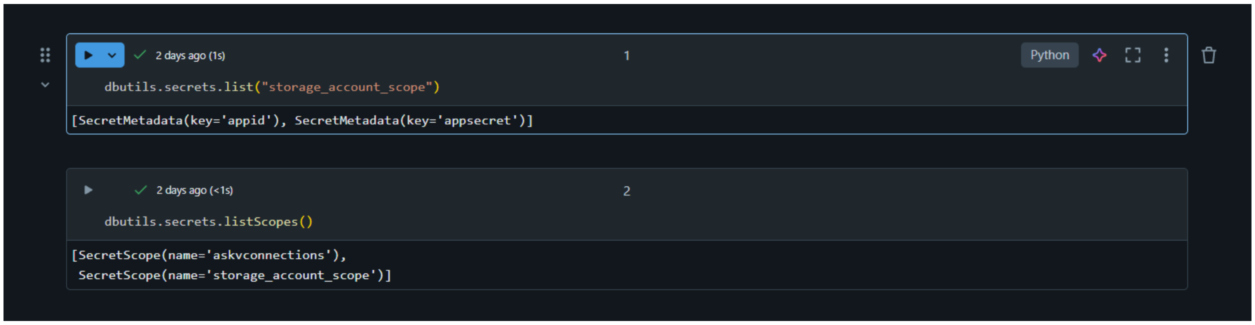
A store for secrets that is identified by a name and backed by a specific store type. Learn more

Scope Name: storage_account_scope

Manage Principal: Creator

Azure Key Vault

1. List available secrets
    - Check if the secret scope "storage_account_scope" exists.
    - Then lists all secret scopes created in Databricks.



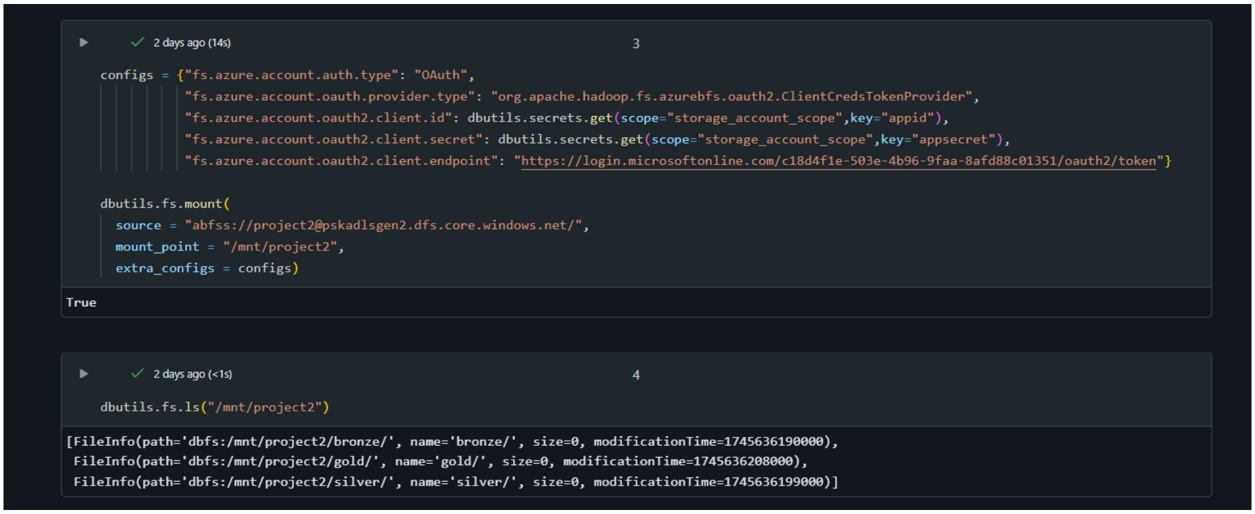
```
dbutils.secrets.list("storage_account_scope")
```

```
[SecretMetadata(key='appid'), SecretMetadata(key='appsecret')]
```

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

```
[SecretScope(name='askvconnections'),
 SecretScope(name='storage_account_scope')]
```

2. Create authentication configs for OAuth
    - We have to provide appid , appsecret and directory id , ADLS gen 2 account name and container name.
    - Mount the project2 container from storage account pskadlsgen2.



```
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="storage_account_scope",key="appid"),
           "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope="storage_account_scope",key="appsecret"),
           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/c18d4f1e-503e-4b96-9faa-8afd88c01351/oauth2/token"}

dbutils.fs.mount(
  source = "abfss://project2@pskadlsgen2.dfs.core.windows.net/",
  mount_point = "/mnt/project2",
  extra_configs = configs)
```

```
True
```

```
dbutils.fs.ls("/mnt/project2")
```

```
[FileInfo(path='dbfs:/mnt/project2/bronze/', name='bronze/', size=0, modificationTime=1745636190000),
 FileInfo(path='dbfs:/mnt/project2/gold/', name='gold/', size=0, modificationTime=1745636208000),
 FileInfo(path='dbfs:/mnt/project2/silver/', name='silver/', size=0, modificationTime=1745636199000)]
```

Mounting completed successfully.

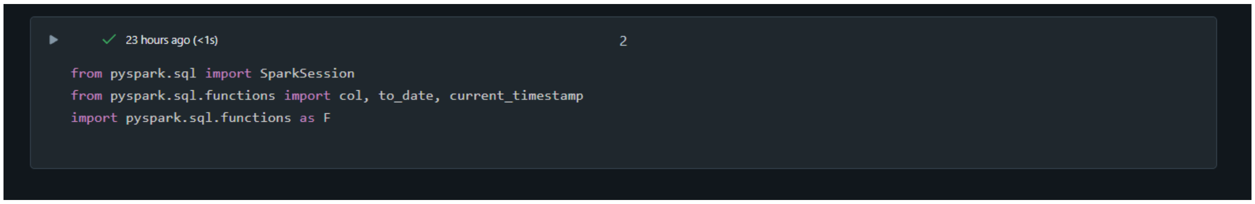
Bronze to Silver

Clean raw data from Bronze layer and save standardized data into Silver layer using Delta format.

3. Import Spark SQL Functions

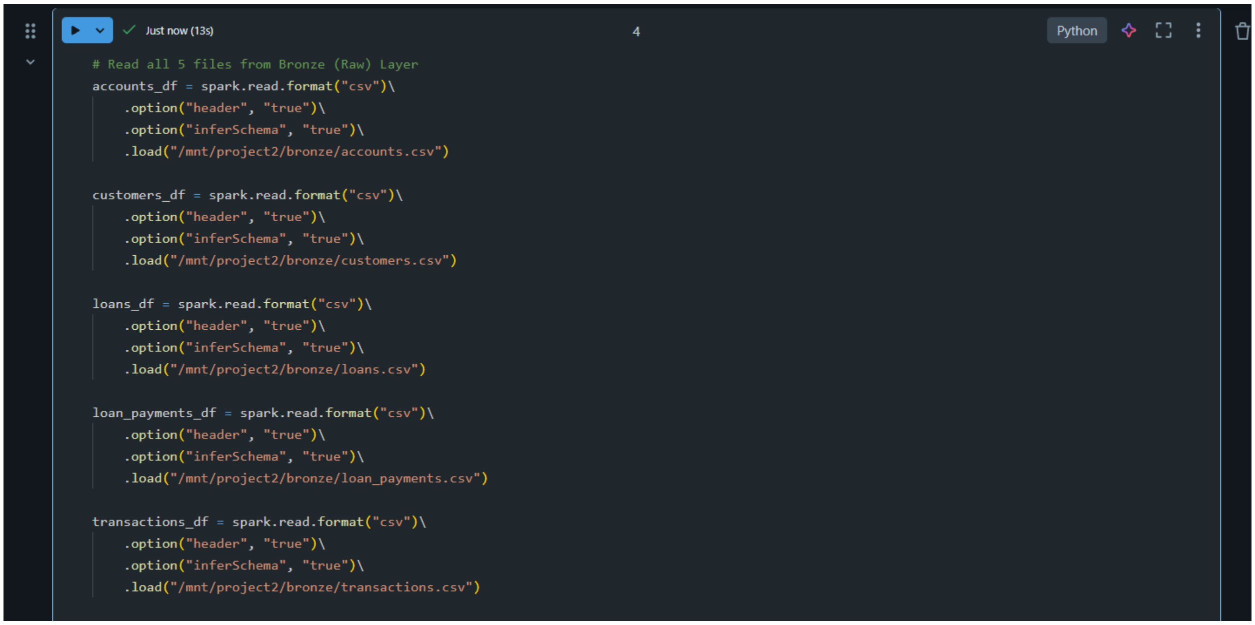
    - SparkSession: Main entry point for reading data with Spark.
    - col: Used to refer to columns in transformations easily.
    - to_date: Used to convert a string column into a date type.

- current_timestamp: Used to generate current date-time.
- import pyspark.sql.functions as F: Allows you to call Spark functions using shorthand F.functionname.



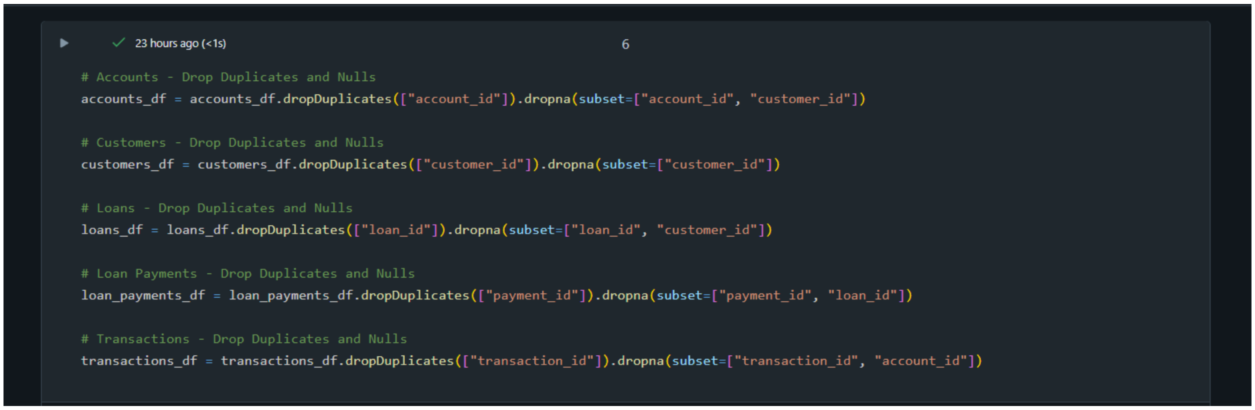
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_date, current_timestamp
import pyspark.sql.functions as F
```

4. Set paths for Bronze (input) and Silver (output)
    - Read data from bronze layer



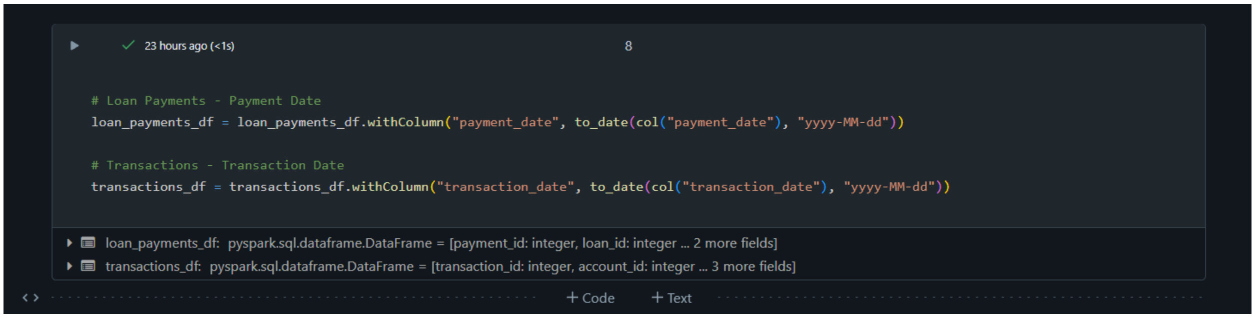
```
# Read all 5 files from Bronze (Raw) Layer
accounts_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load("/mnt/project2/bronze/accounts.csv")

customers_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load("/mnt/project2/bronze/customers.csv")

loans_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load("/mnt/project2/bronze/loans.csv")

loan_payments_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load("/mnt/project2/bronze/loan_payments.csv")

transactions_df = spark.read.format("csv")\
    .option("header", "true")\
    .option("inferSchema", "true")\
    .load("/mnt/project2/bronze/transactions.csv")
```

5. Data Cleaning and Transformation
    - Removed duplicates and null values from data



```
# Accounts - Drop Duplicates and Nulls
accounts_df = accounts_df.dropDuplicates(["account_id"]).dropna(subset=["account_id", "customer_id"])

# Customers - Drop Duplicates and Nulls
customers_df = customers_df.dropDuplicates(["customer_id"]).dropna(subset=["customer_id"])

# Loans - Drop Duplicates and Nulls
loans_df = loans_df.dropDuplicates(["loan_id"]).dropna(subset=["loan_id", "customer_id"])

# Loan Payments - Drop Duplicates and Nulls
loan_payments_df = loan_payments_df.dropDuplicates(["payment_id"]).dropna(subset=["payment_id", "loan_id"])

# Transactions - Drop Duplicates and Nulls
transactions_df = transactions_df.dropDuplicates(["transaction_id"]).dropna(subset=["transaction_id", "account_id"])
```

6. Date Formatting for Loan Payments and Transactions : String to DateType

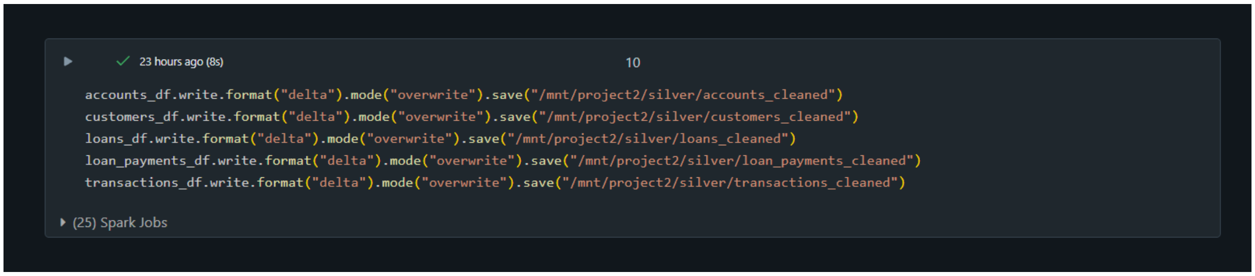


23 hours ago (<1s) 8

```
# Loan Payments - Payment Date
loan_payments_df = loan_payments_df.withColumn("payment_date", to_date(col("payment_date"), "yyyy-MM-dd"))

# Transactions - Transaction Date
transactions_df = transactions_df.withColumn("transaction_date", to_date(col("transaction_date"), "yyyy-MM-dd"))
```

```
loan_payments_df:  pyspark.sql.dataframe.DataFrame = [payment_id: integer, loan_id: integer ... 2 more fields]
transactions_df:  pyspark.sql.dataframe.DataFrame = [transaction_id: integer, account_id: integer ... 3 more fields]
```

7. Save into Silver Layer (Delta Format)
   - Used **Delta** format for ACID transactions and version control.
   - Each raw Bronze dataset now has a **cleaned and Saved in Silver Layer.**

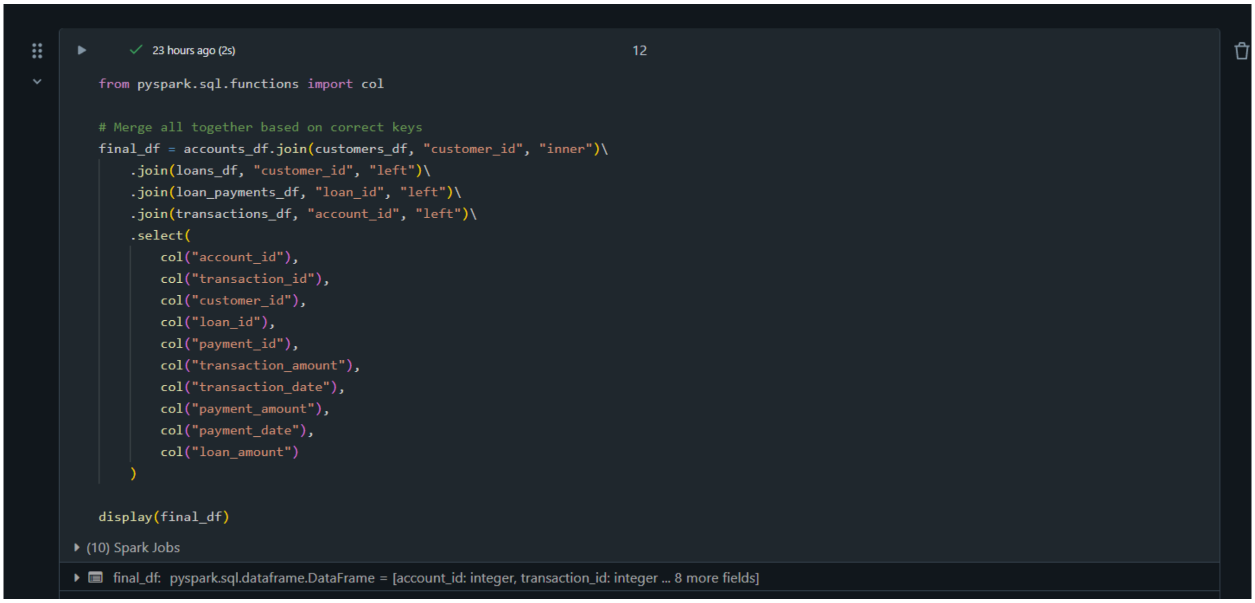


23 hours ago (8s) 10

```
accounts_df.write.format("delta").mode("overwrite").save("/mnt/project2/silver/accounts_cleaned")
customers_df.write.format("delta").mode("overwrite").save("/mnt/project2/silver/customers_cleaned")
loans_df.write.format("delta").mode("overwrite").save("/mnt/project2/silver/loans_cleaned")
loan_payments_df.write.format("delta").mode("overwrite").save("/mnt/project2/silver/loan_payments_cleaned")
transactions_df.write.format("delta").mode("overwrite").save("/mnt/project2/silver/transactions_cleaned")
```

(25) Spark Jobs

8. Merge All Data Together Based on Correct Keys
   Inner join to ensure every account has a valid customer.
   Left join and select to bring other Column

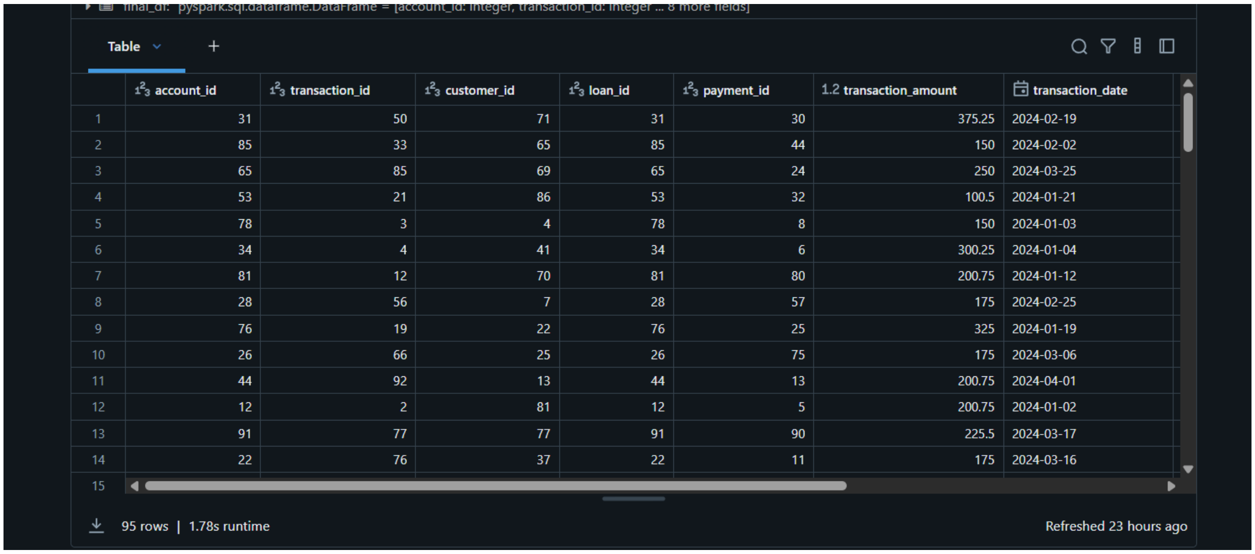


23 hours ago (2s) 12

```
from pyspark.sql.functions import col

# Merge all together based on correct keys
final_df = accounts_df.join(customers_df, "customer_id", "inner")\
    .join(loans_df, "customer_id", "left")\
    .join(loan_payments_df, "loan_id", "left")\
    .join(transactions_df, "account_id", "left")\
    .select(
        col("account_id"),
        col("transaction_id"),
        col("customer_id"),
        col("loan_id"),
        col("payment_id"),
        col("transaction_amount"),
        col("transaction_date"),
        col("payment_amount"),
        col("payment_date"),
        col("loan_amount")
    )

display(final_df)
```

(10) Spark Jobs

```
final_df:  pyspark.sql.dataframe.DataFrame = [account_id: integer, transaction_id: integer ... 8 more fields]
```



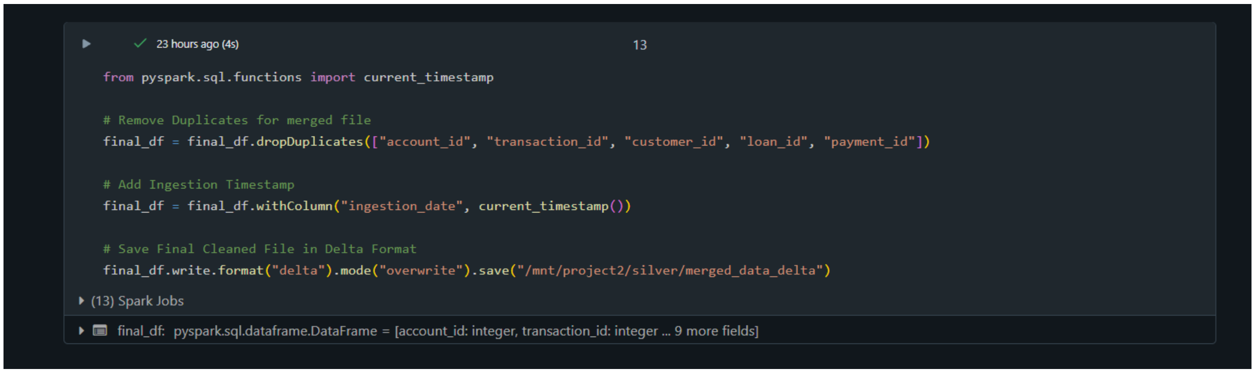
final_df: pyspark.sql.dataframe.DataFrame = [account_id: integer, transaction_id: integer ... 8 more fields]

	account_id	transaction_id	customer_id	loan_id	payment_id	transaction_amount	transaction_date
1	31	50	71	31	30	375.25	2024-02-19
2	85	33	65	85	44	150	2024-02-02
3	65	85	69	65	24	250	2024-03-25
4	53	21	86	53	32	100.5	2024-01-21
5	78	3	4	78	8	150	2024-01-03
6	34	4	41	34	6	300.25	2024-01-04
7	81	12	70	81	80	200.75	2024-01-12
8	28	56	7	28	57	175	2024-02-25
9	76	19	22	76	25	325	2024-01-19
10	26	66	25	26	75	175	2024-03-06
11	44	92	13	44	13	200.75	2024-04-01
12	12	2	81	12	5	200.75	2024-01-02
13	91	77	77	91	90	225.5	2024-03-17
14	22	76	37	22	11	175	2024-03-16
15							

95 rows | 1.78s runtime

Refreshed 23 hours ago

9. Remove Duplicates and Save Final Merged Silver Table in Delta Format



23 hours ago (4s) 13

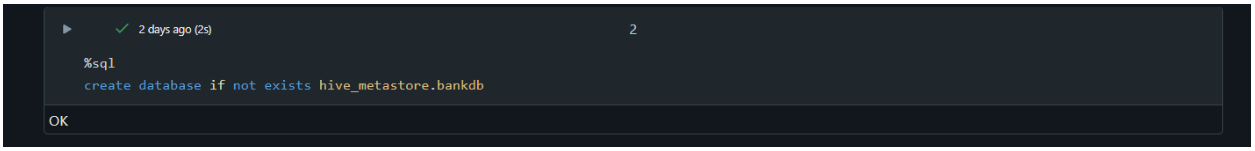
```
from pyspark.sql.functions import current_timestamp

# Remove Duplicates for merged file
final_df = final_df.dropDuplicates(["account_id", "transaction_id", "customer_id", "loan_id", "payment_id"])

# Add Ingestion Timestamp
final_df = final_df.withColumn("ingestion_date", current_timestamp())

# Save Final Cleaned File in Delta Format
final_df.write.format("delta").mode("overwrite").save("/mnt/project2/silver/merged_data_delta")
```

(13) Spark Jobs

final_df: pyspark.sql.dataframe.DataFrame = [account_id: integer, transaction_id: integer ... 9 more fields]

Silver to Gold by Using SCD Type-1

10. Create a Database in Hive Metastore
    - Create a database bankdb inside Hive Metastore if it doesn't exist to Store all Gold Delta tables in one organized database.

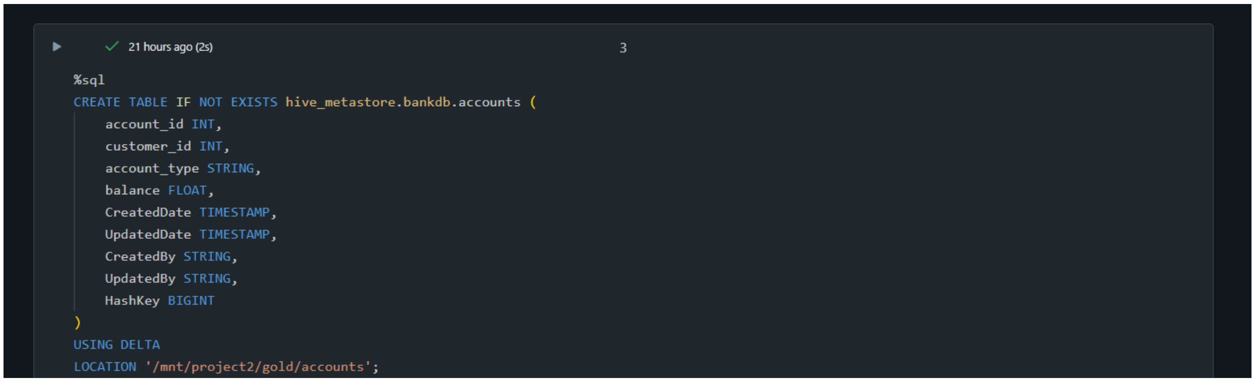


2 days ago (2s) 2

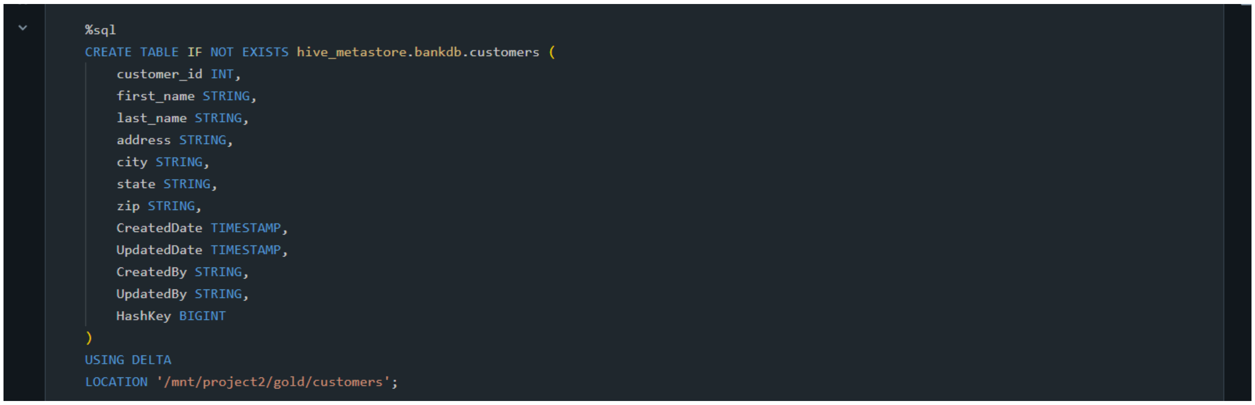
```
%sql
create database if not exists hive_metastore.bankdb
```

OK

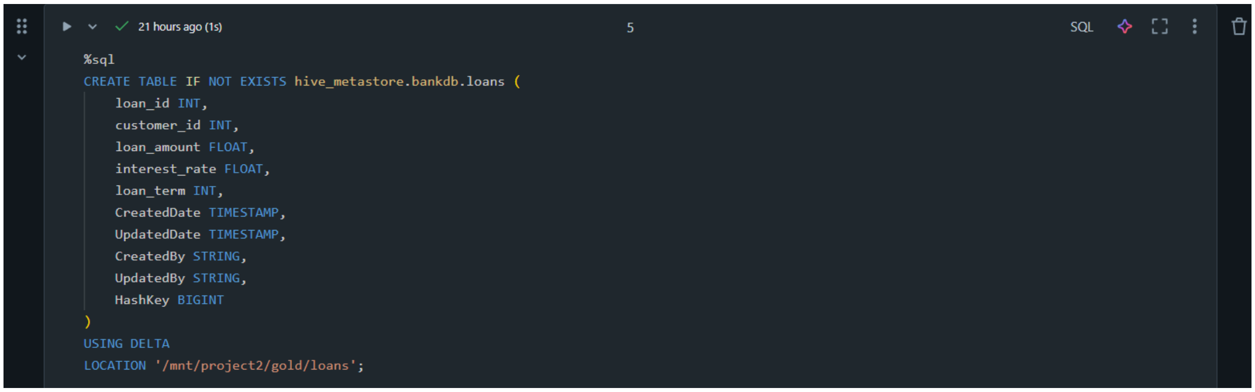
11. Create Gold Layer Tables (5 tables)
    For each table (Accounts, Customers, Loans, Loan Payments, Transactions):



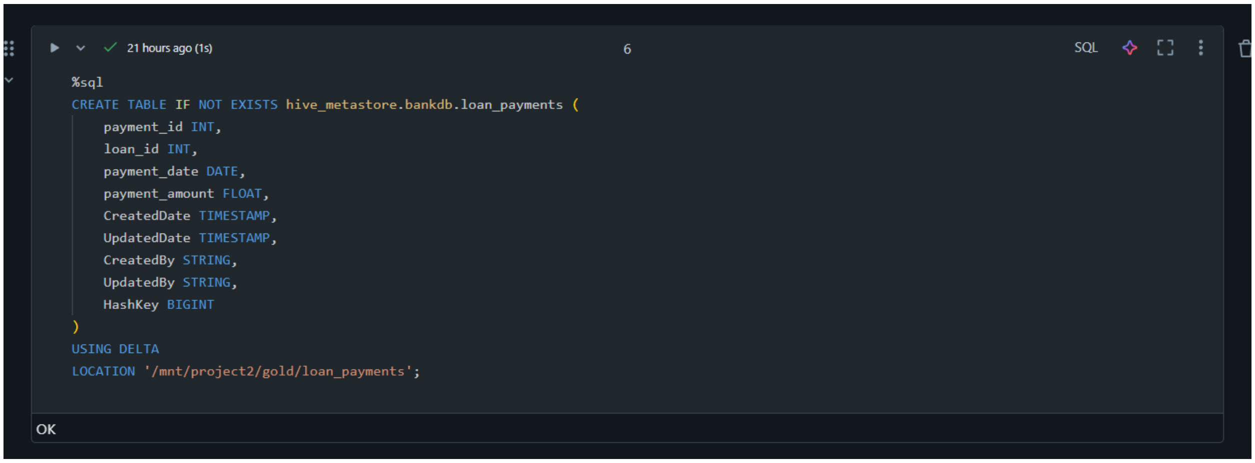
```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.bankdb.accounts (
    account_id INT,
    customer_id INT,
    account_type STRING,
    balance FLOAT,
    CreatedDate TIMESTAMP,
    UpdatedDate TIMESTAMP,
    CreatedBy STRING,
    UpdatedBy STRING,
    HashKey BIGINT
)
USING DELTA
LOCATION '/mnt/project2/gold/accounts';
```



```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.bankdb.customers (
    customer_id INT,
    first_name STRING,
    last_name STRING,
    address STRING,
    city STRING,
    state STRING,
    zip STRING,
    CreatedDate TIMESTAMP,
    UpdatedDate TIMESTAMP,
    CreatedBy STRING,
    UpdatedBy STRING,
    HashKey BIGINT
)
USING DELTA
LOCATION '/mnt/project2/gold/customers';
```

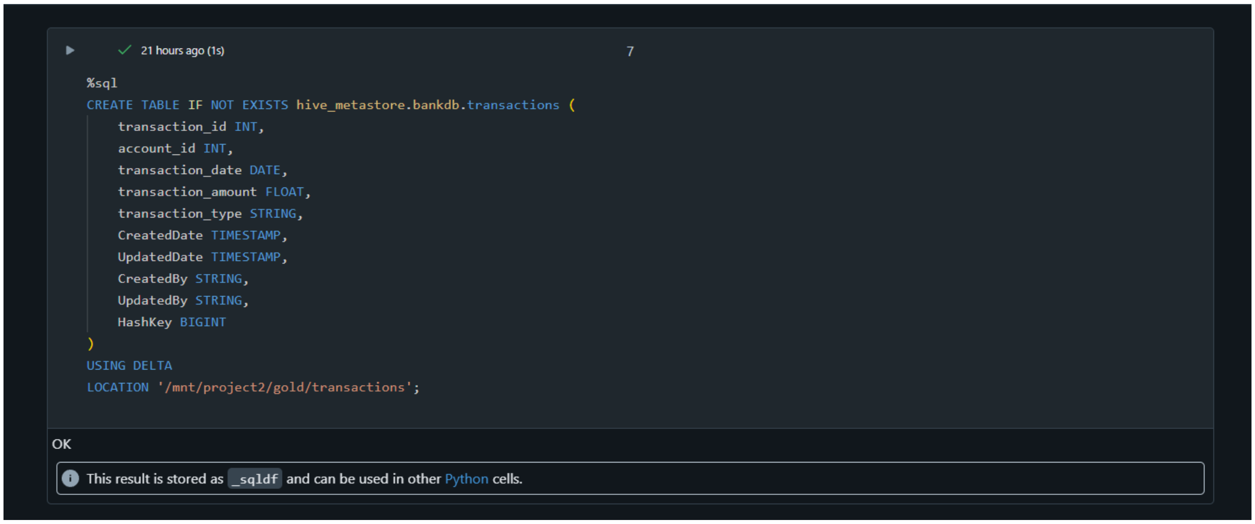


```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.bankdb.loans (
    loan_id INT,
    customer_id INT,
    loan_amount FLOAT,
    interest_rate FLOAT,
    loan_term INT,
    CreatedDate TIMESTAMP,
    UpdatedDate TIMESTAMP,
    CreatedBy STRING,
    UpdatedBy STRING,
    HashKey BIGINT
)
USING DELTA
LOCATION '/mnt/project2/gold/loans';
```



```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.bankdb.loan_payments (
    payment_id INT,
    loan_id INT,
    payment_date DATE,
    payment_amount FLOAT,
    CreatedDate TIMESTAMP,
    UpdatedDate TIMESTAMP,
    CreatedBy STRING,
    UpdatedBy STRING,
    HashKey BIGINT
)
USING DELTA
LOCATION '/mnt/project2/gold/loan_payments';
```

```
OK
```

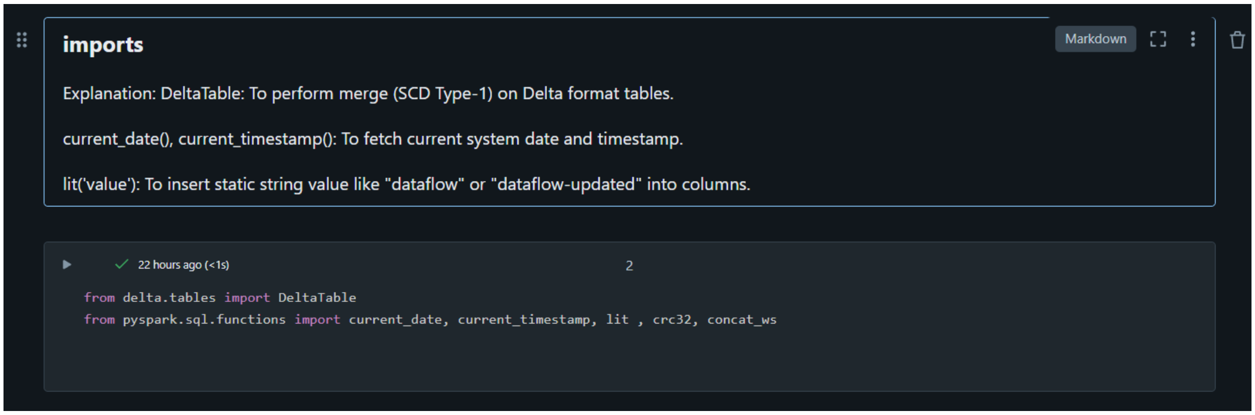


```
%sql
CREATE TABLE IF NOT EXISTS hive_metastore.bankdb.transactions (
    transaction_id INT,
    account_id INT,
    transaction_date DATE,
    transaction_amount FLOAT,
    transaction_type STRING,
    CreatedDate TIMESTAMP,
    UpdatedDate TIMESTAMP,
    CreatedBy STRING,
    UpdatedBy STRING,
    HashKey BIGINT
)
USING DELTA
LOCATION '/mnt/project2/gold/transactions';
```

```
OK
```

This result is stored as `_sqldf` and can be used in other Python cells.

12. Import Required Libraries
    - **DeltaTable**: To manage Delta format operations (like Merge).
    - **current_date, current_timestamp**: Fetch current system date or time.
    - **lit**: Insert constant values into DataFrame.
    - **col**: Reference columns easily.
    - **crc32, concat_ws**: For creating **HashKey** (row-level comparison).



imports

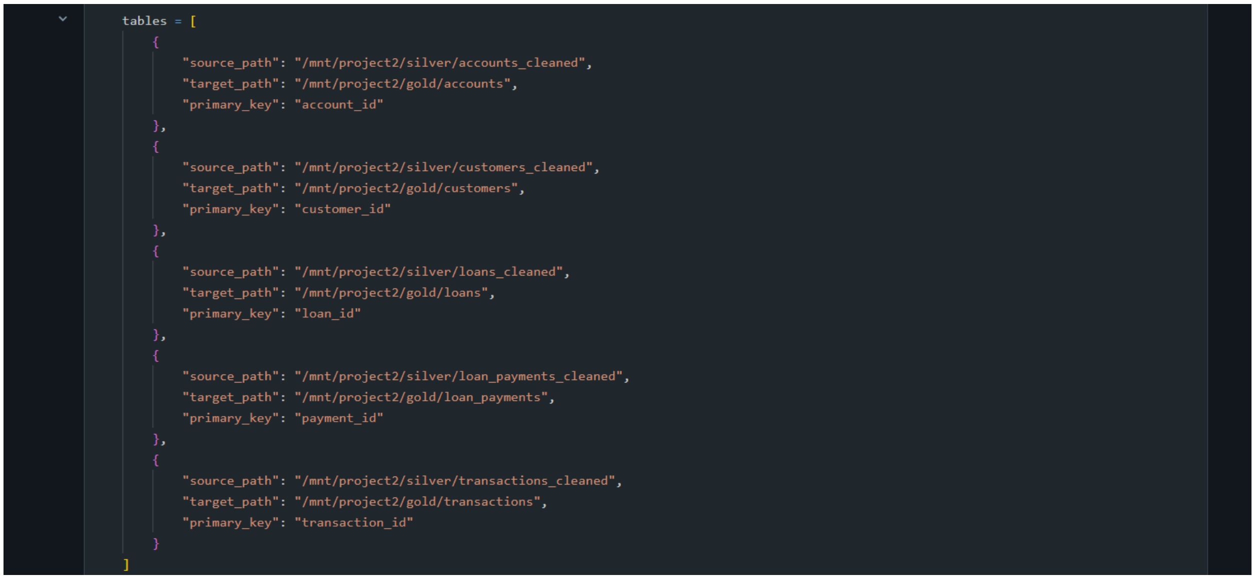
Explanation: DeltaTable: To perform merge (SCD Type-1) on Delta format tables.

current_date(), current_timestamp(): To fetch current system date and timestamp.

lit('value'): To insert static string value like "dataflow" or "dataflow-updated" into columns.

```
from delta.tables import DeltaTable
from pyspark.sql.functions import current_date, current_timestamp, lit , crc32, concat_ws
```

13. Define Table Metadata (Sources and Targets)
    - **source_path**: Clean Silver Delta file path.
    - **target_path**: Gold Delta table path.
    - **primary_key**: Column used to join and compare rows.



```
tables = [
    {
        "source_path": "/mnt/project2/silver/accounts_cleaned",
        "target_path": "/mnt/project2/gold/accounts",
        "primary_key": "account_id"
    },
    {
        "source_path": "/mnt/project2/silver/customers_cleaned",
        "target_path": "/mnt/project2/gold/customers",
        "primary_key": "customer_id"
    },
    {
        "source_path": "/mnt/project2/silver/loans_cleaned",
        "target_path": "/mnt/project2/gold/loans",
        "primary_key": "loan_id"
    },
    {
        "source_path": "/mnt/project2/silver/loan_payments_cleaned",
        "target_path": "/mnt/project2/gold/loan_payments",
        "primary_key": "payment_id"
    },
    {
        "source_path": "/mnt/project2/silver/transactions_cleaned",
        "target_path": "/mnt/project2/gold/transactions",
        "primary_key": "transaction_id"
    }
]
```

SCD Type-1 Logic

Explanation

1. Import Libraries: Needed functions like current_timestamp(), lit() etc.

2. Define Tables: List all table info - source, target, primary key.

3. Read Source: Load the silver layer Delta files.

4. Drop Null Primary Keys: Ensures primary key integrity.

5. Add Metadata: CreatedDate, UpdatedDate, CreatedBy, UpdatedBy.

6. Add HashKey: For change detection using crc32 checksum.

7. Read Target: Load existing Gold tables.

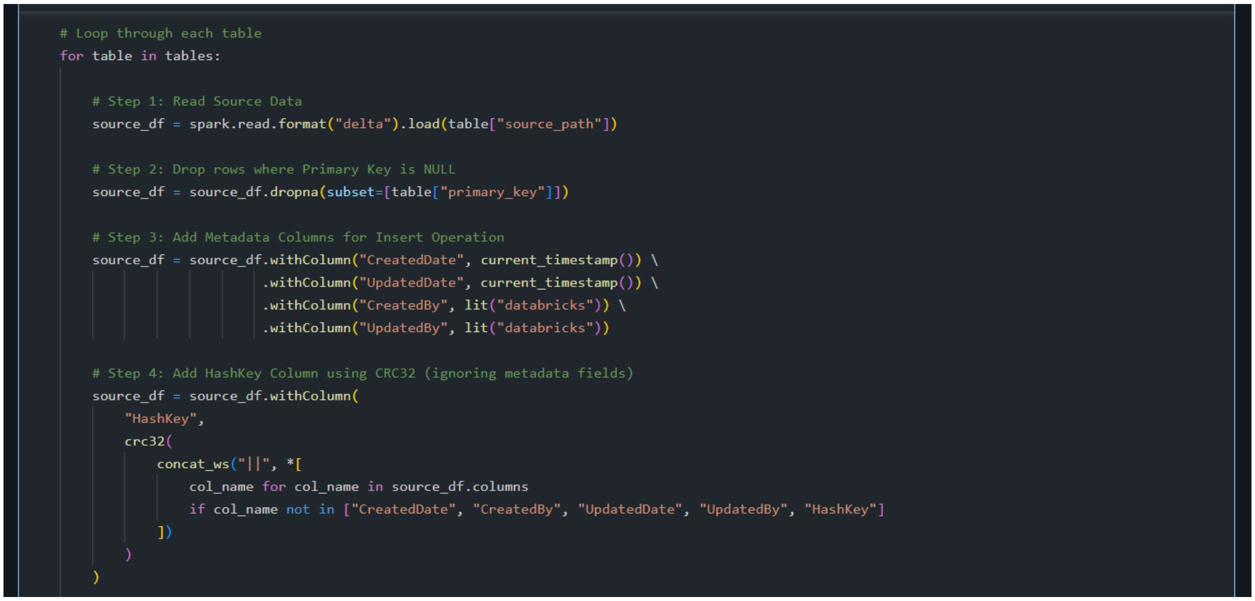
8. Update Mapping: Define fields to update if records match.

9. Insert Mapping: Define fields to insert if record does not exist.

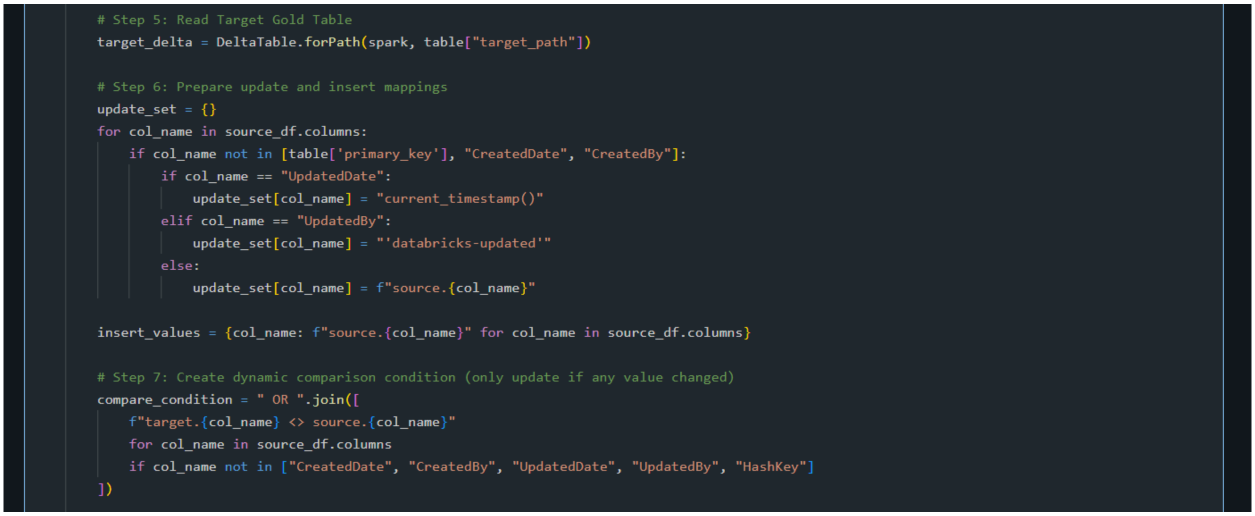
10. Compare Condition: Only update when something actually changes.

11. Merge Logic: Merge records into Gold layer - updates or inserts.

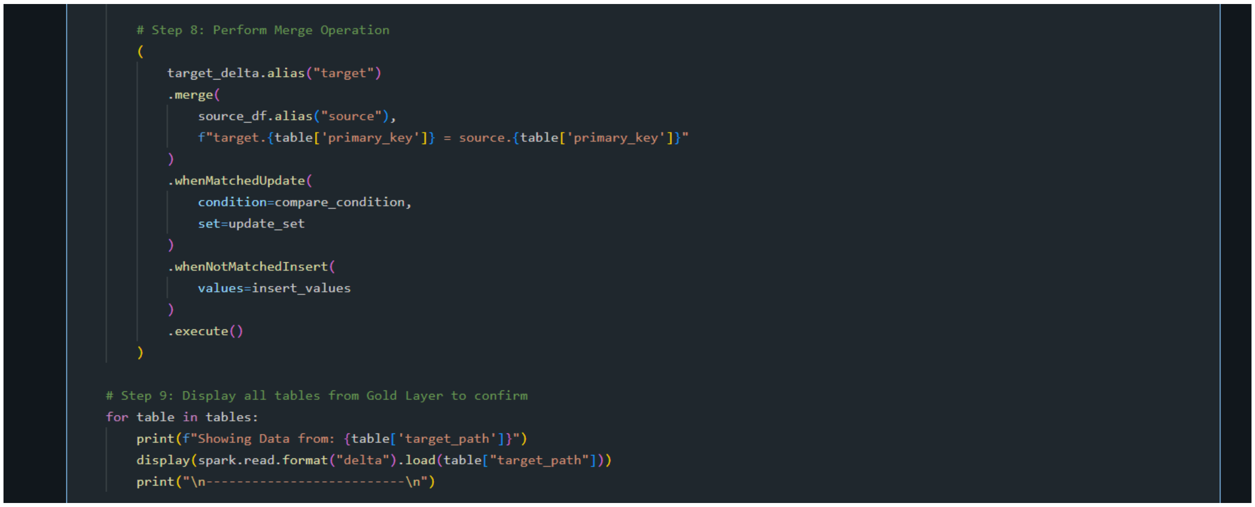
12. Display: Show final data from each Gold table.



```python
# Loop through each table
for table in tables:

    # Step 1: Read Source Data
    source_df = spark.read.format("delta").load(table["source_path"])

    # Step 2: Drop rows where Primary Key is NULL
    source_df = source_df.dropna(subset=[table["primary_key"]])

    # Step 3: Add Metadata Columns for Insert Operation
    source_df = source_df.withColumn("CreatedDate", current_timestamp()) \
                         .withColumn("UpdatedDate", current_timestamp()) \
                         .withColumn("CreatedBy", lit("databricks")) \
                         .withColumn("UpdatedBy", lit("databricks"))

    # Step 4: Add HashKey Column using CRC32 (ignoring metadata fields)
    source_df = source_df.withColumn(
        "HashKey",
        crc32(
            concat_ws("||", *[
                col_name for col_name in source_df.columns
                if col_name not in ["CreatedDate", "CreatedBy", "UpdatedDate", "UpdatedBy", "HashKey"]
            ])
        )
    )
```

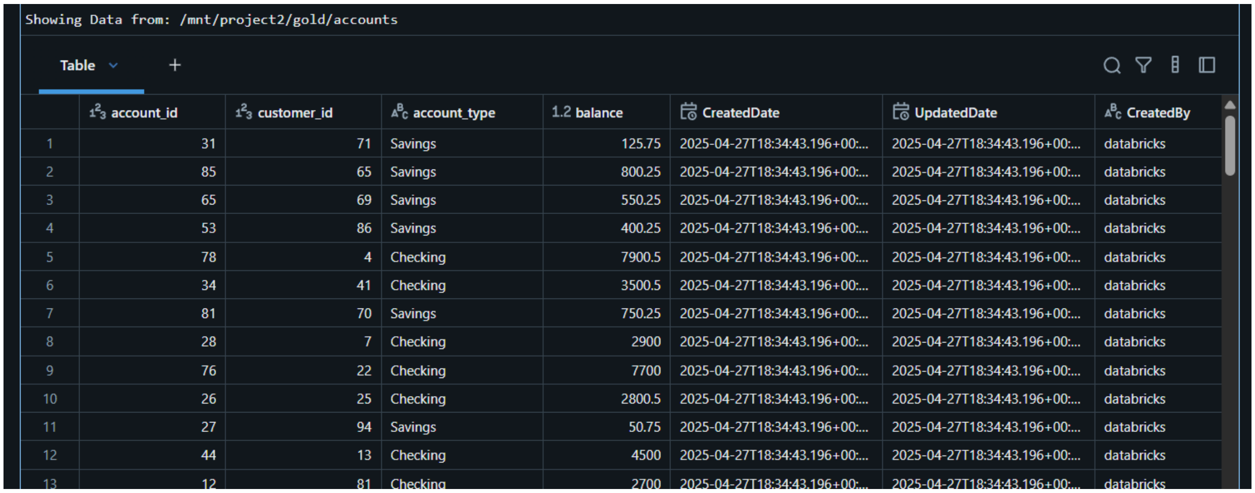


```python
    # Step 5: Read Target Gold Table
    target_delta = DeltaTable.forPath(spark, table["target_path"])

    # Step 6: Prepare update and insert mappings
    update_set = {}
    for col_name in source_df.columns:
        if col_name not in [table['primary_key'], "CreatedDate", "CreatedBy"]:
            if col_name == "UpdatedDate":
                update_set[col_name] = "current_timestamp()"
            elif col_name == "UpdatedBy":
                update_set[col_name] = "'databricks-updated'"
            else:
                update_set[col_name] = f"source.{col_name}"

    insert_values = {col_name: f"source.{col_name}" for col_name in source_df.columns}

    # Step 7: Create dynamic comparison condition (only update if any value changed)
    compare_condition = " OR ".join([
        f"target.{col_name} <> source.{col_name}"
        for col_name in source_df.columns
        if col_name not in ["CreatedDate", "CreatedBy", "UpdatedDate", "UpdatedBy", "HashKey"]
    ])
```



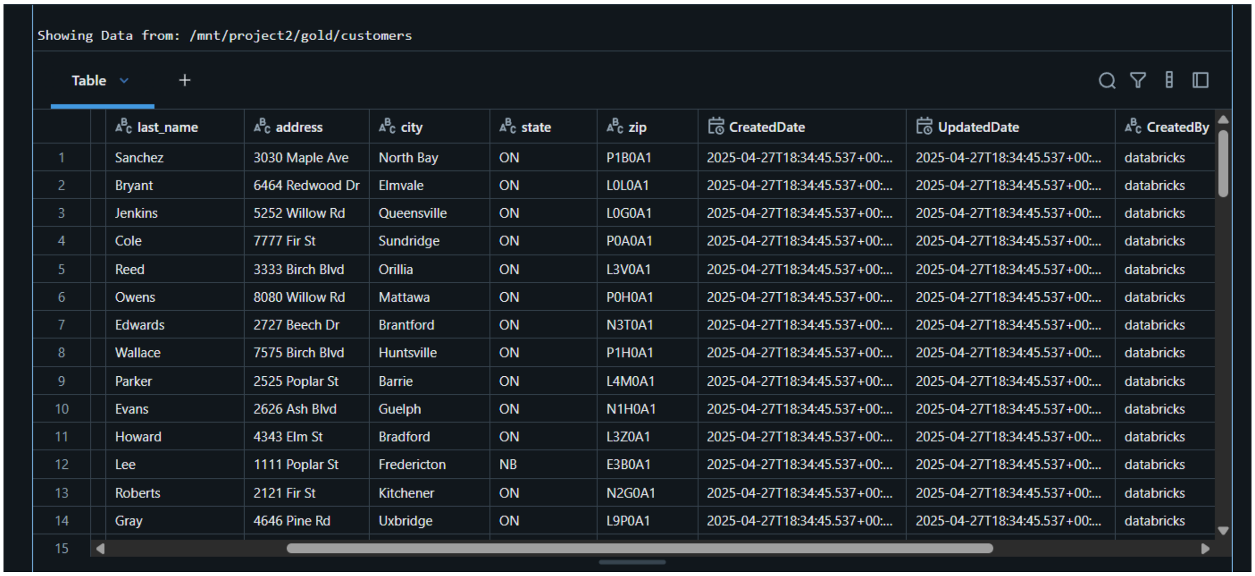
```python
    # Step 8: Perform Merge Operation
    (
        target_delta.alias("target")
        .merge(
            source_df.alias("source"),
            f"target.{table['primary_key']} = source.{table['primary_key']}"
        )
        .whenMatchedUpdate(
            condition=compare_condition,
            set=update_set
        )
        .whenNotMatchedInsert(
            values=insert_values
        )
        .execute()
    )

# Step 9: Display all tables from Gold Layer to confirm
for table in tables:
    print(f"Showing Data from: {table['target_path']}")
    display(spark.read.format("delta").load(table["target_path"]))
    print("\n-------------------------\n")
```

Insert data tables



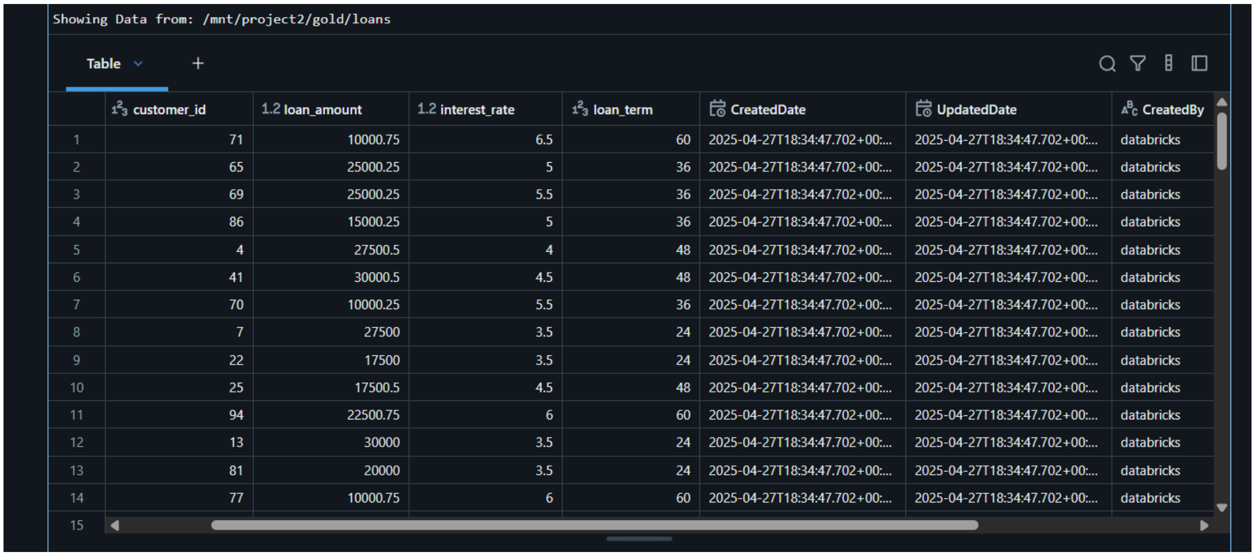
```
Showing Data from: /mnt/project2/gold/accounts
```

	account_id	customer_id	account_type	balance	CreatedDate	UpdatedDate	CreatedBy
1	31	71	Savings	125.75	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
2	85	65	Savings	800.25	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
3	65	69	Savings	550.25	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
4	53	86	Savings	400.25	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
5	78	4	Checking	7900.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
6	34	41	Checking	3500.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
7	81	70	Savings	750.25	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
8	28	7	Checking	2900	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
9	76	22	Checking	7700	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
10	26	25	Checking	2800.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
11	27	94	Savings	50.75	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
12	44	13	Checking	4500	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks
13	12	81	Checking	2700	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks



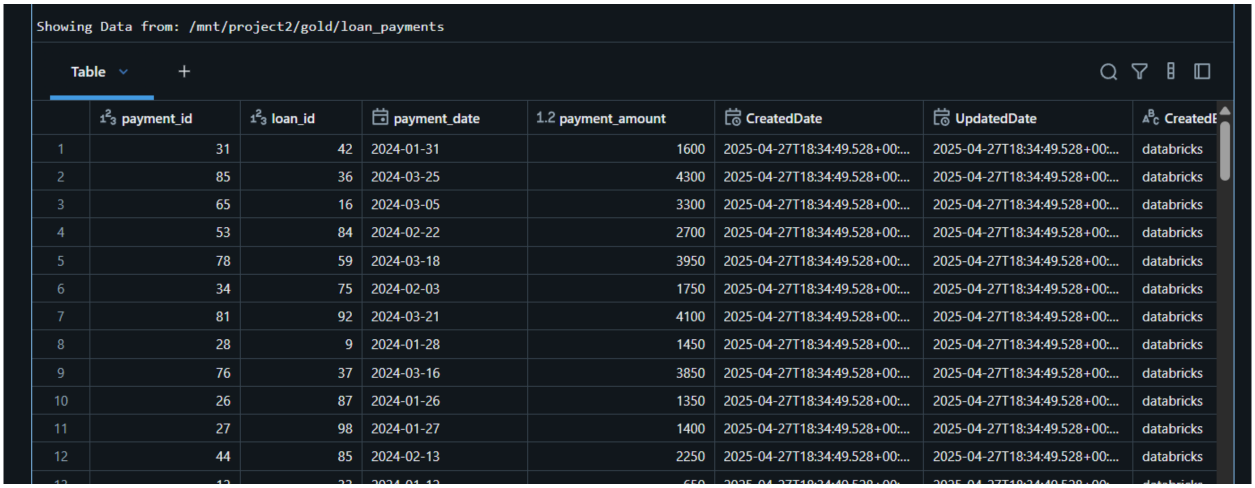
```
Showing Data from: /mnt/project2/gold/customers
```

	last_name	address	city	state	zip	CreatedDate	UpdatedDate	CreatedBy
1	Sanchez	3030 Maple Ave	North Bay	ON	P1B0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
2	Bryant	6464 Redwood Dr	Elmvale	ON	L0L0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
3	Jenkins	5252 Willow Rd	Queensville	ON	L0G0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
4	Cole	7777 Fir St	Sundridge	ON	P0A0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
5	Reed	3333 Birch Blvd	Orillia	ON	L3V0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
6	Owens	8080 Willow Rd	Mattawa	ON	P0H0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
7	Edwards	2727 Beech Dr	Brantford	ON	N3T0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
8	Wallace	7575 Birch Blvd	Huntsville	ON	P1H0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
9	Parker	2525 Poplar St	Barrie	ON	L4M0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
10	Evans	2626 Ash Blvd	Guelph	ON	N1H0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
11	Howard	4343 Elm St	Bradford	ON	L3Z0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
12	Lee	1111 Poplar St	Fredericton	NB	E3B0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
13	Roberts	2121 Fir St	Kitchener	ON	N2G0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
14	Gray	4646 Pine Rd	Uxbridge	ON	L9P0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks
15								



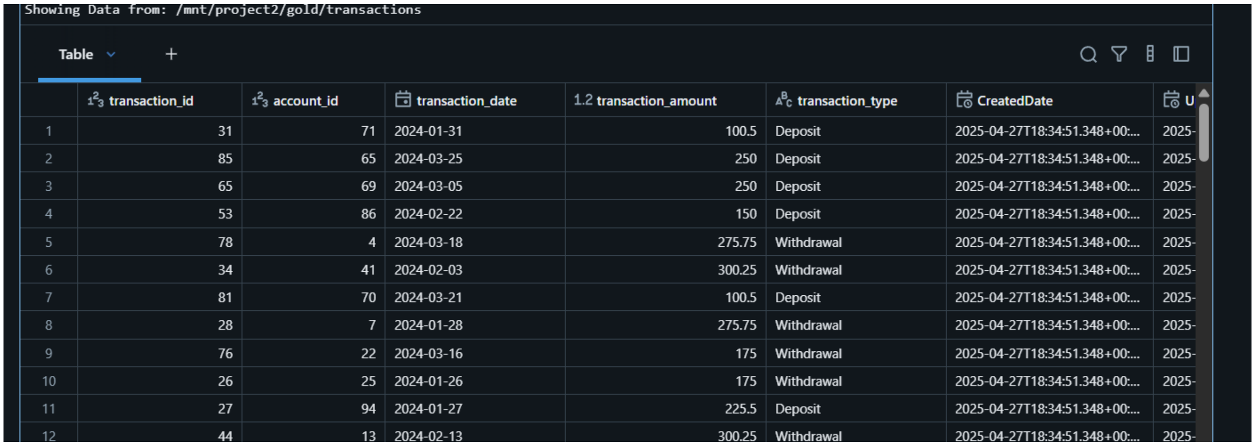
```
Showing Data from: /mnt/project2/gold/loans
```

	customer_id	loan_amount	interest_rate	loan_term	CreatedDate	UpdatedDate	CreatedBy
1	71	10000.75	6.5	60	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
2	65	25000.25	5	36	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
3	69	25000.25	5.5	36	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
4	86	15000.25	5	36	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
5	4	27500.5	4	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
6	41	30000.5	4.5	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
7	70	10000.25	5.5	36	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
8	7	27500	3.5	24	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
9	22	17500	3.5	24	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
10	25	17500.5	4.5	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
11	94	22500.75	6	60	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
12	13	30000	3.5	24	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
13	81	20000	3.5	24	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
14	77	10000.75	6	60	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks
15							



Showing Data from: /mnt/project2/gold/loan_payments

	payment_id	loan_id	payment_date	payment_amount	CreatedDate	UpdatedDate	CreatedE
1	31	42	2024-01-31	1600	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
2	85	36	2024-03-25	4300	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
3	65	16	2024-03-05	3300	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
4	53	84	2024-02-22	2700	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
5	78	59	2024-03-18	3950	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
6	34	75	2024-02-03	1750	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
7	81	92	2024-03-21	4100	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
8	28	9	2024-01-28	1450	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
9	76	37	2024-03-16	3850	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
10	26	87	2024-01-26	1350	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
11	27	98	2024-01-27	1400	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks
12	44	85	2024-02-13	2250	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks

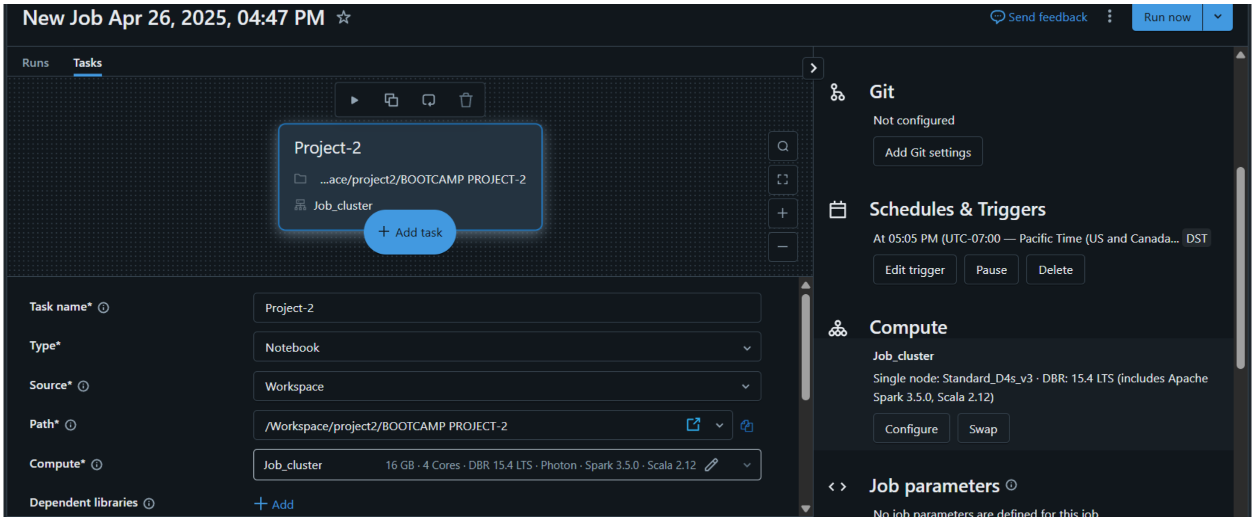


Showing Data from: /mnt/project2/gold/transactions

	transaction_id	account_id	transaction_date	transaction_amount	transaction_type	CreatedDate	U
1	31	71	2024-01-31	100.5	Deposit	2025-04-27T18:34:51.348+00:...	2025-
2	85	65	2024-03-25	250	Deposit	2025-04-27T18:34:51.348+00:...	2025-
3	65	69	2024-03-05	250	Deposit	2025-04-27T18:34:51.348+00:...	2025-
4	53	86	2024-02-22	150	Deposit	2025-04-27T18:34:51.348+00:...	2025-
5	78	4	2024-03-18	275.75	Withdrawal	2025-04-27T18:34:51.348+00:...	2025-
6	34	41	2024-02-03	300.25	Withdrawal	2025-04-27T18:34:51.348+00:...	2025-
7	81	70	2024-03-21	100.5	Deposit	2025-04-27T18:34:51.348+00:...	2025-
8	28	7	2024-01-28	275.75	Withdrawal	2025-04-27T18:34:51.348+00:...	2025-
9	76	22	2024-03-16	175	Withdrawal	2025-04-27T18:34:51.348+00:...	2025-
10	26	25	2024-01-26	175	Withdrawal	2025-04-27T18:34:51.348+00:...	2025-
11	27	94	2024-01-27	225.5	Deposit	2025-04-27T18:34:51.348+00:...	2025-
12	44	13	2024-02-13	300.25	Withdrawal	2025-04-27T18:34:51.348+00:...	2025-

Schedule a Tigger

I overwrite new updated data in bronze file before this notebook run as per schedule so we can see updated data in delta table.



New Job Apr 26, 2025, 04:47 PM ☆

Task name* Project-2

Type* Notebook

Source* Workspace

Path* /Workspace/project2/BOOTCAMP PROJECT-2

Compute* Job_cluster 16 GB · 4 Cores · DBR 15.4 LTS · Photon · Spark 3.5.0 · Scala 2.12

Dependent libraries + Add

Git

Not configured

Schedules & Triggers

At 05:05 PM (UTC-07:00 — Pacific Time (US and Canada... DST

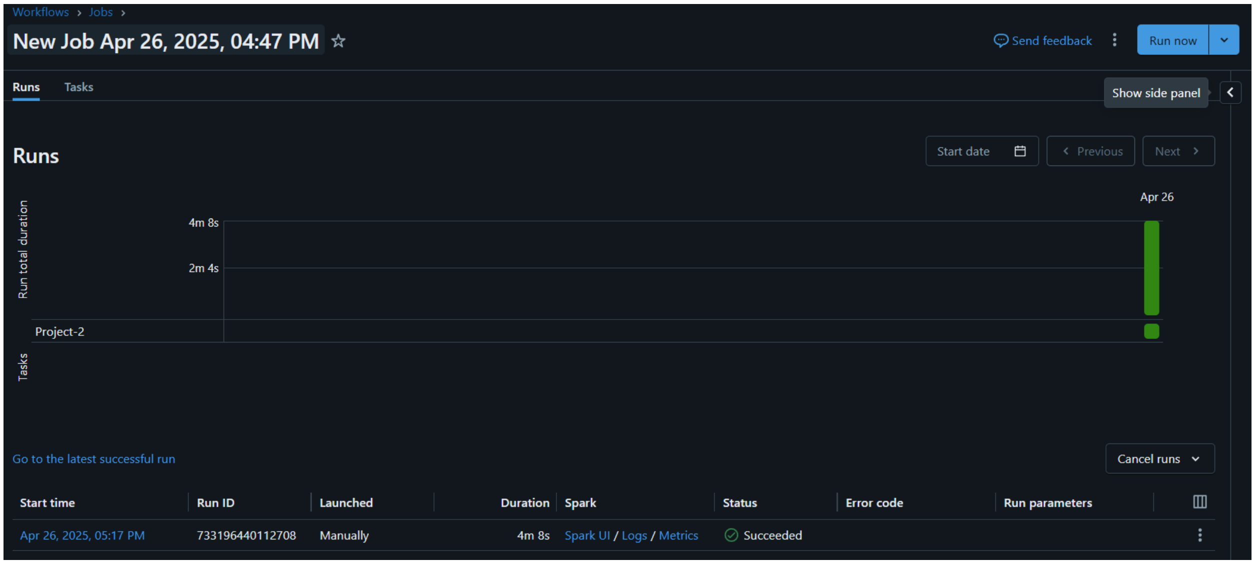
Compute

Job_cluster

Single node: Standard_D4s_v3 · DBR: 15.4 LTS (includes Apache Spark 3.5.0, Scala 2.12)

Job parameters

No job parameters are defined for this job

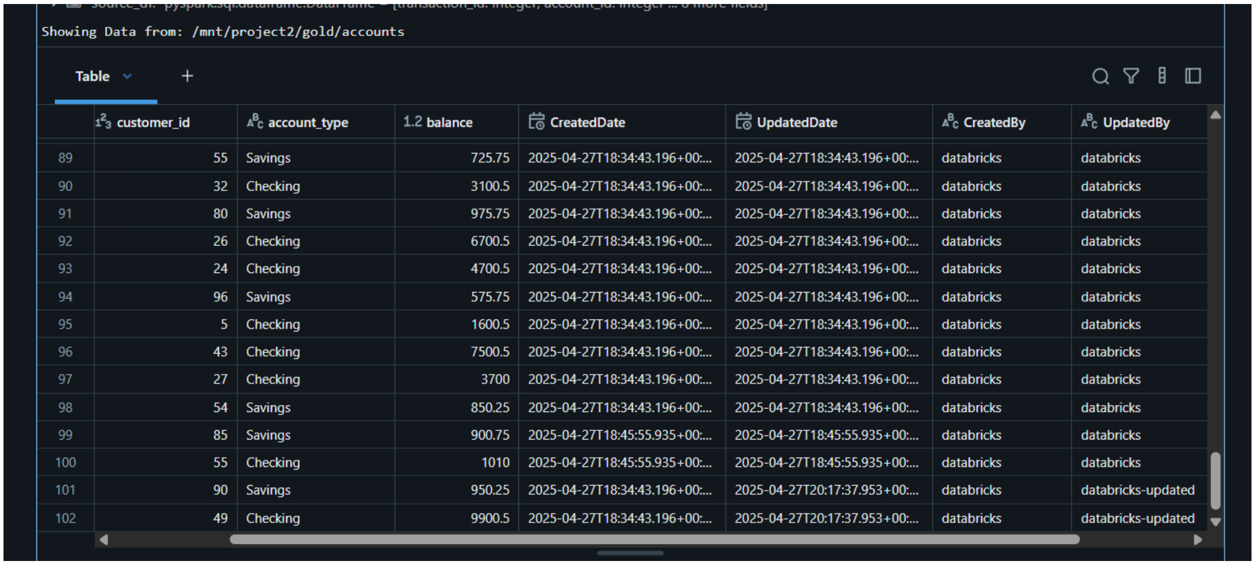


New Job Apr 26, 2025, 04:47 PM

Runs

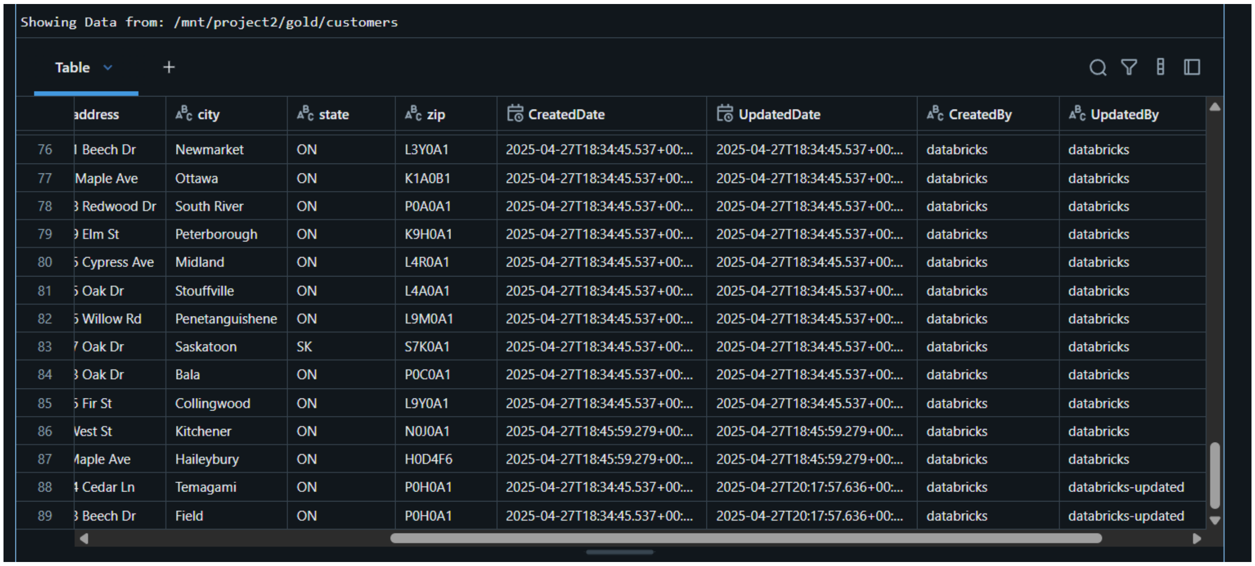
Start time	Run ID	Launched	Duration	Spark	Status	Error code	Run parameters
Apr 26, 2025, 05:17 PM	733196440112708	Manually	4m 8s	Spark UI / Logs / Metrics	Succeeded		

Notebook run successfully and we can observed the new data inserted and old data undated in below tables.



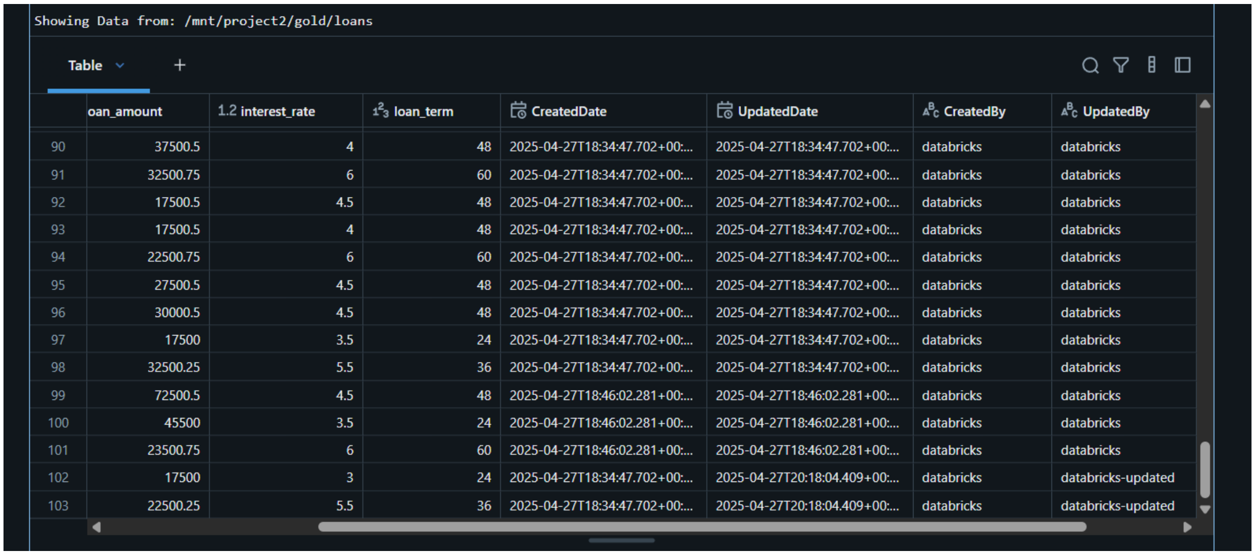
```
source_df: pyspark.sql.dataframe.DataFrame = [transaction_id: integer, account_id: integer ... 6 more fields]
Showing Data from: /mnt/project2/gold/accounts
```

	customer_id	account_type	balance	CreatedDate	UpdatedDate	CreatedBy	UpdatedBy
89	55	Savings	725.75	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
90	32	Checking	3100.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
91	80	Savings	975.75	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
92	26	Checking	6700.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
93	24	Checking	4700.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
94	96	Savings	575.75	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
95	5	Checking	1600.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
96	43	Checking	7500.5	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
97	27	Checking	3700	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
98	54	Savings	850.25	2025-04-27T18:34:43.196+00:...	2025-04-27T18:34:43.196+00:...	databricks	databricks
99	85	Savings	900.75	2025-04-27T18:45:55.935+00:...	2025-04-27T18:45:55.935+00:...	databricks	databricks
100	55	Checking	1010	2025-04-27T18:45:55.935+00:...	2025-04-27T18:45:55.935+00:...	databricks	databricks
101	90	Savings	950.25	2025-04-27T18:34:43.196+00:...	2025-04-27T20:17:37.953+00:...	databricks	databricks-updated
102	49	Checking	9900.5	2025-04-27T18:34:43.196+00:...	2025-04-27T20:17:37.953+00:...	databricks	databricks-updated



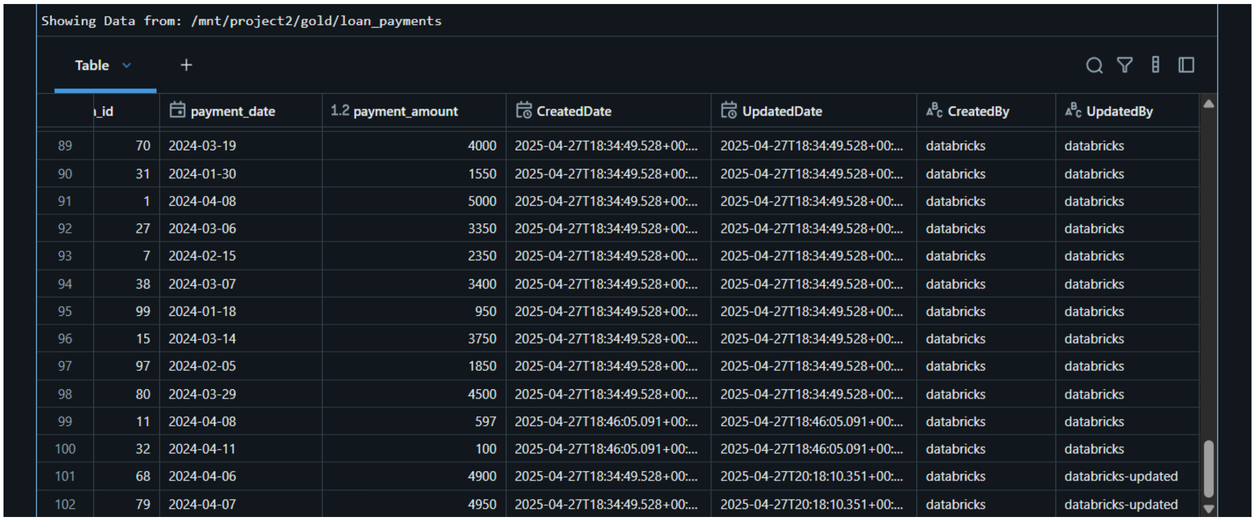
```
Showing Data from: /mnt/project2/gold/customers
```

	address	city	state	zip	CreatedDate	UpdatedDate	CreatedBy	UpdatedBy
76	l Beech Dr	Newmarket	ON	L3Y0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
77	Maple Ave	Ottawa	ON	K1A0B1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
78	3 Redwood Dr	South River	ON	P0A0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
79	9 Elm St	Peterborough	ON	K9H0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
80	5 Cypress Ave	Midland	ON	L4R0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
81	5 Oak Dr	Stouffville	ON	L4A0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
82	5 Willow Rd	Penetanguishene	ON	L9M0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
83	7 Oak Dr	Saskatoon	SK	S7K0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
84	3 Oak Dr	Bala	ON	P0C0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
85	5 Fir St	Collingwood	ON	L9Y0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T18:34:45.537+00:...	databricks	databricks
86	Vest St	Kitchener	ON	N0J0A1	2025-04-27T18:45:59.279+00:...	2025-04-27T18:45:59.279+00:...	databricks	databricks
87	laple Ave	Haileybury	ON	H0D4F6	2025-04-27T18:45:59.279+00:...	2025-04-27T18:45:59.279+00:...	databricks	databricks
88	4 Cedar Ln	Temagami	ON	P0H0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T20:17:57.636+00:...	databricks	databricks-updated
89	3 Beech Dr	Field	ON	P0H0A1	2025-04-27T18:34:45.537+00:...	2025-04-27T20:17:57.636+00:...	databricks	databricks-updated



```
Showing Data from: /mnt/project2/gold/loans
```

	oan_amount	interest_rate	loan_term	CreatedDate	UpdatedDate	CreatedBy	UpdatedBy
90	37500.5	4	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
91	32500.75	6	60	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
92	17500.5	4.5	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
93	17500.5	4	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
94	22500.75	6	60	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
95	27500.5	4.5	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
96	30000.5	4.5	48	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
97	17500	3.5	24	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
98	32500.25	5.5	36	2025-04-27T18:34:47.702+00:...	2025-04-27T18:34:47.702+00:...	databricks	databricks
99	72500.5	4.5	48	2025-04-27T18:46:02.281+00:...	2025-04-27T18:46:02.281+00:...	databricks	databricks
100	45500	3.5	24	2025-04-27T18:46:02.281+00:...	2025-04-27T18:46:02.281+00:...	databricks	databricks
101	23500.75	6	60	2025-04-27T18:46:02.281+00:...	2025-04-27T18:46:02.281+00:...	databricks	databricks
102	17500	3	24	2025-04-27T18:34:47.702+00:...	2025-04-27T20:18:04.409+00:...	databricks	databricks-updated
103	22500.25	5.5	36	2025-04-27T18:34:47.702+00:...	2025-04-27T20:18:04.409+00:...	databricks	databricks-updated

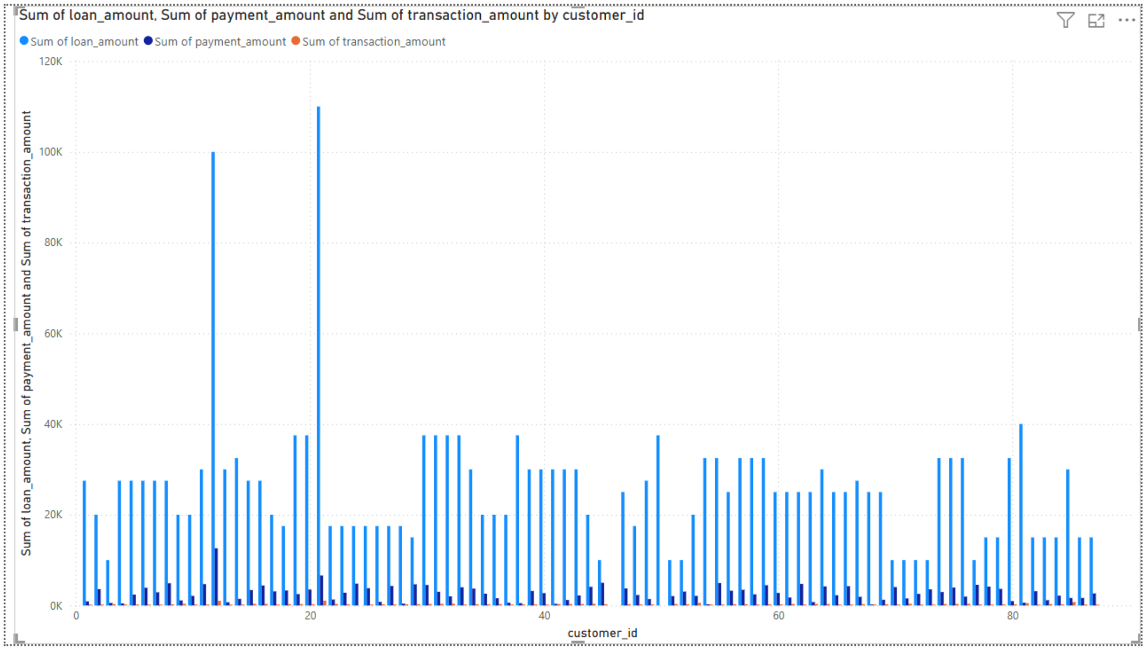


```
Showing Data from: /mnt/project2/gold/loan_payments
```

	_id	payment_date	payment_amount	CreatedDate	UpdatedDate	CreatedBy	UpdatedBy
89	70	2024-03-19	4000	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
90	31	2024-01-30	1550	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
91	1	2024-04-08	5000	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
92	27	2024-03-06	3350	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
93	7	2024-02-15	2350	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
94	38	2024-03-07	3400	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
95	99	2024-01-18	950	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
96	15	2024-03-14	3750	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
97	97	2024-02-05	1850	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
98	80	2024-03-29	4500	2025-04-27T18:34:49.528+00:...	2025-04-27T18:34:49.528+00:...	databricks	databricks
99	11	2024-04-08	597	2025-04-27T18:46:05.091+00:...	2025-04-27T18:46:05.091+00:...	databricks	databricks
100	32	2024-04-11	100	2025-04-27T18:46:05.091+00:...	2025-04-27T18:46:05.091+00:...	databricks	databricks
101	68	2024-04-06	4900	2025-04-27T18:34:49.528+00:...	2025-04-27T20:18:10.351+00:...	databricks	databricks-updated
102	79	2024-04-07	4950	2025-04-27T18:34:49.528+00:...	2025-04-27T20:18:10.351+00:...	databricks	databricks-updated

Power BI bar chart

Clustered Bar Chart showing Top Customers (Transaction, Loan, Payment Amounts).



"This graph compares how much each customer borrowed, paid back, and transacted. It shows that loan balances are high, but repayments and daily transactions are relatively low."