Data Pipeline for Customer Account Analysis

Bootcamp Project - 2

Project Objective

The objective of this project is to design and implement a robust, end-to-end data pipeline for processing and analyzing customer account data using Azure cloud services. The pipeline includes ingesting data from a backend storage system, cleaning and transforming the data using Azure Databricks, and writing the data to Azure SQL Database.

Architecture Overview

Technologies Used:

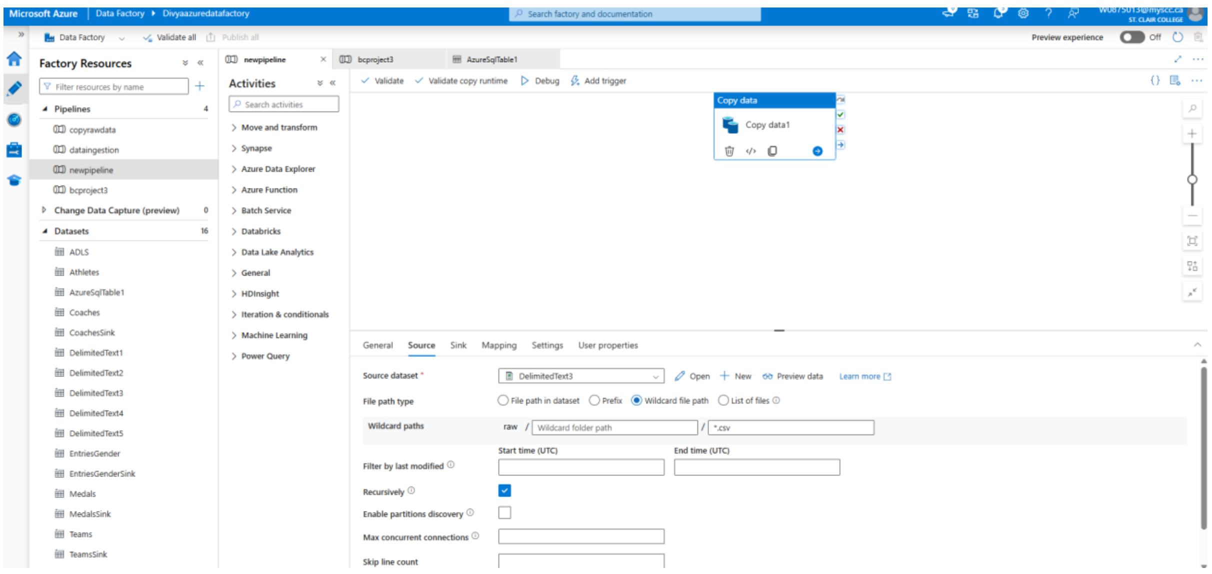
- **Azure Data Factory (ADF)**
- **Azure Data Lake Storage (ADLS Gen2)** – Bronze, Silver, and Gold Containers
- **Azure Databricks**
- **Apache Spark (PySpark)**
- **Azure SQL Database**

Step-by-Step Implementation

Step 1: Data Ingestion (Backend Storage → Bronze Container)

Objective: Transfer raw data from the backend team's storage account to the raw (bronze) container in ADLS.

- **Tool:** Azure Data Factory (Copy Activity)
- **Source:** Backend storage account
    - accounts.csv
    - customers.csv
    - loan_payments.csv
    - loans.csv
    - transactions.csv
- **Sink:** Bronze container in user's ADLS Gen2 storage
- **Reference Dataset:** Kaggle AI Bank Dataset
  Kaggle Dataset Link



Source dataset *: DelimitedText3
File path type: ( ) File path in dataset ( ) Prefix [✓ Wildcard file path] ( ) List of files
Wildcard paths: raw / Wildcard folder path / *.csv
Filter by last modified: Start time (UTC) End time (UTC)
Recursively: ✓
Enable partitions discovery: ☐
Max concurrent connections:
Skip line count:

Configuration Notes:

- Linked services were configured for both source and destination.
- File format settings: CSV with headers and proper delimiters.

Step 2: Databricks Activity (Delta Processing on Raw Data)

Objective: Clean and transform raw data for structured storage.

- **Tool:** Azure Databricks Notebook (Notebook 1)
- **Language:** PySpark

Operations:

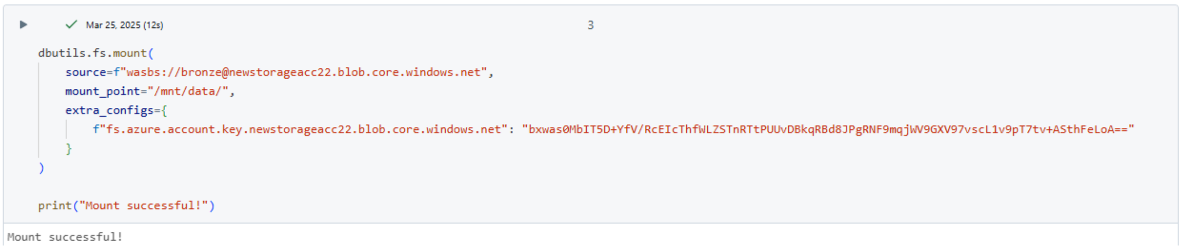
- **Read Data:** Read all 5 CSVs from the Bronze container.

Data Cleaning:

- **Null Handling:** Removes rows where either the zip or address fields are null.
- **Data Standardization:** Converts address endings like “St” → “Street” and “Ave” → “Avenue” for consistency.

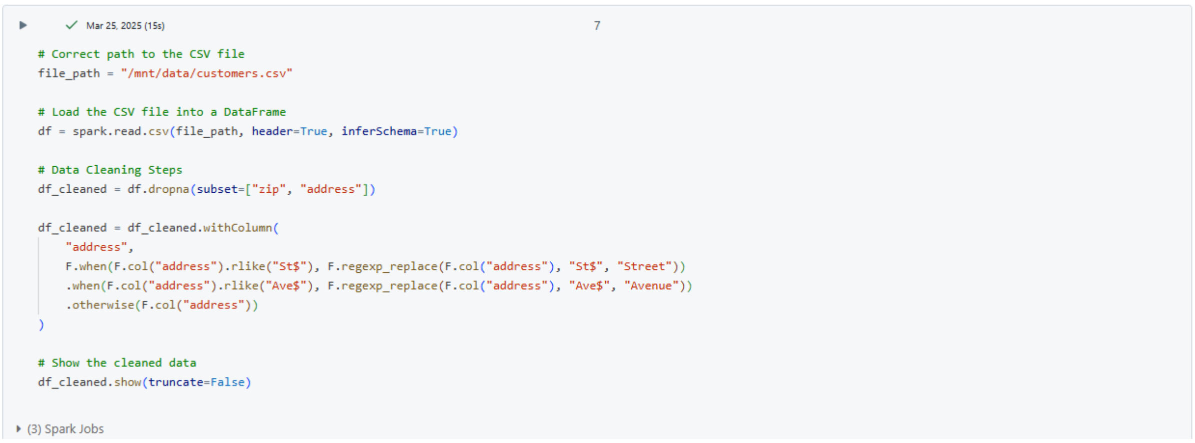
Data Transformation:

- Handle missing values using defined imputation strategies.
- Write outputs as Parquet/Delta files to the curated (silver) container.


Mar 25, 2025 (12s) 3

```
dbutils.fs.mount(
    source=f"wasbs://bronze@newstorageacc22.blob.core.windows.net",
    mount_point="/mnt/data/",
    extra_configs={
        f"fs.azure.account.key.newstorageacc22.blob.core.windows.net": "bxwas0MbIT5D+YfV/RcEIcThfWLZSTnRTtPUUvDBkqRBd8JPgRNF9mqjWV9GXV97vscL1v9pT7tv+ASthFeLoA=="
    }
)

print("Mount successful!")
```

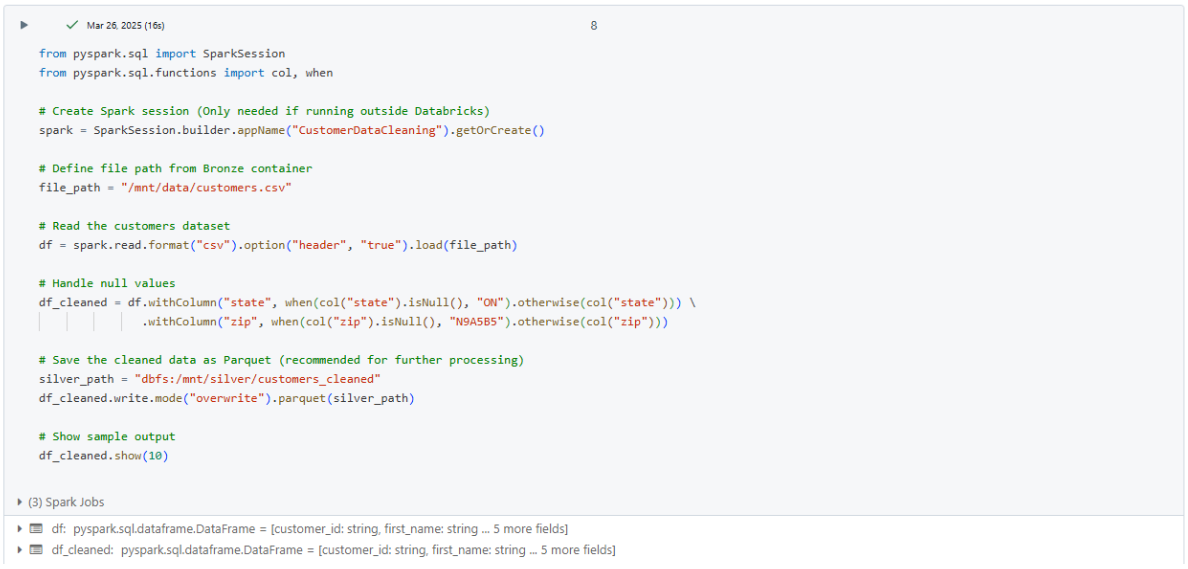
```
Mount successful!
```



Mar 25, 2025 (15s) 7

```python
# Correct path to the CSV file
file_path = "/mnt/data/customers.csv"

# Load the CSV file into a DataFrame
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Data Cleaning Steps
df_cleaned = df.dropna(subset=["zip", "address"])

df_cleaned = df_cleaned.withColumn(
    "address",
    F.when(F.col("address").rlike("St$"), F.regexp_replace(F.col("address"), "St$", "Street"))
    .when(F.col("address").rlike("Ave$"), F.regexp_replace(F.col("address"), "Ave$", "Avenue"))
    .otherwise(F.col("address"))
)

# Show the cleaned data
df_cleaned.show(truncate=False)
```

(3) Spark Jobs



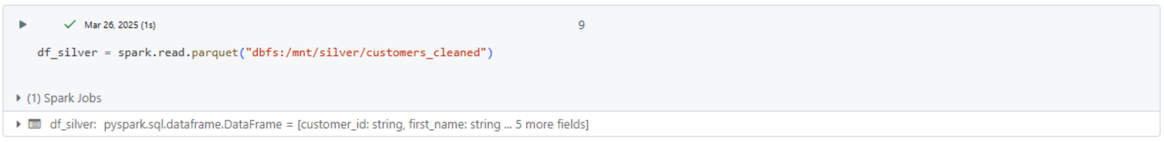
Mar 26, 2025 (16s) 8

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when

# Create Spark session (Only needed if running outside Databricks)
spark = SparkSession.builder.appName("CustomerDataCleaning").getOrCreate()

# Define file path from Bronze container
file_path = "/mnt/data/customers.csv"

# Read the customers dataset
df = spark.read.format("csv").option("header", "true").load(file_path)

# Handle null values
df_cleaned = df.withColumn("state", when(col("state").isNull(), "ON").otherwise(col("state"))) \
                .withColumn("zip", when(col("zip").isNull(), "N9A5B5").otherwise(col("zip")))

# Save the cleaned data as Parquet (recommended for further processing)
silver_path = "dbfs:/mnt/silver/customers_cleaned"
df_cleaned.write.mode("overwrite").parquet(silver_path)

# Show sample output
df_cleaned.show(10)
```

(3) Spark Jobs

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

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

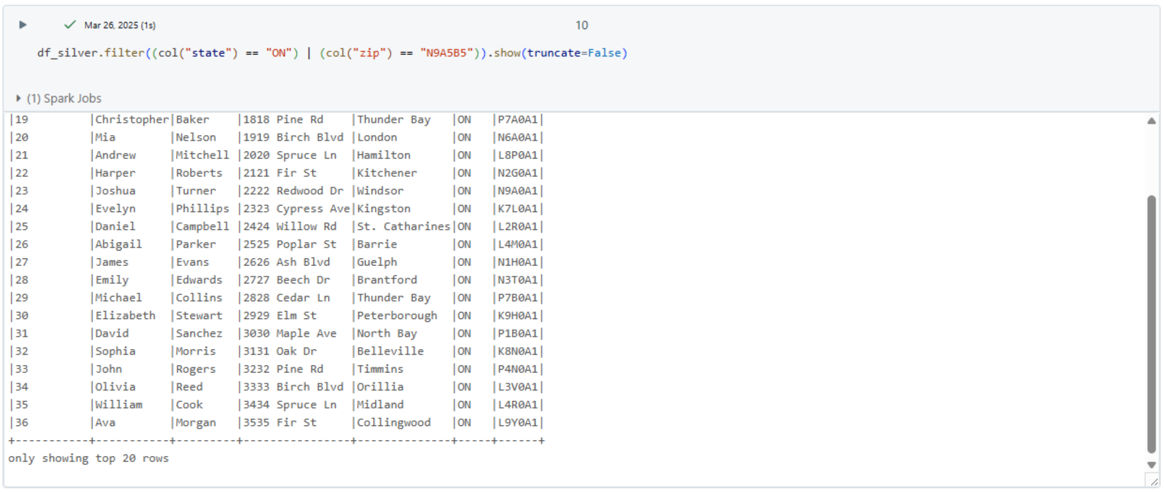


Mar 26, 2025 (1s) 9

```python
df_silver = spark.read.parquet("dbfs:/mnt/silver/customers_cleaned")
```

(1) Spark Jobs

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

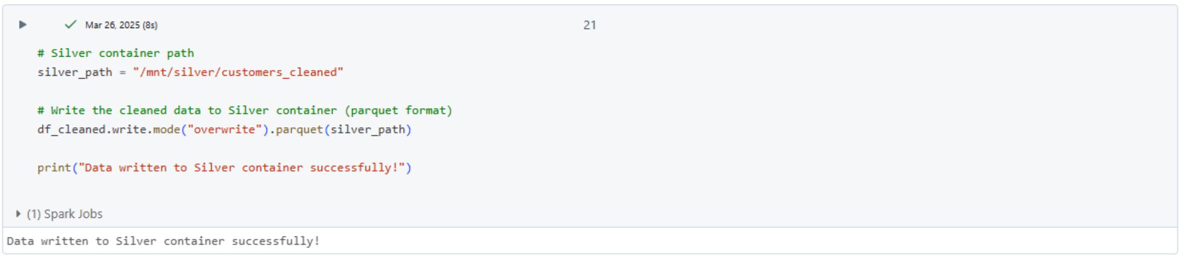


Mar 26, 2025 (1s) 10

```python
df_silver.filter((col("state") == "ON") | (col("zip") == "N9A5B5")).show(truncate=False)
```

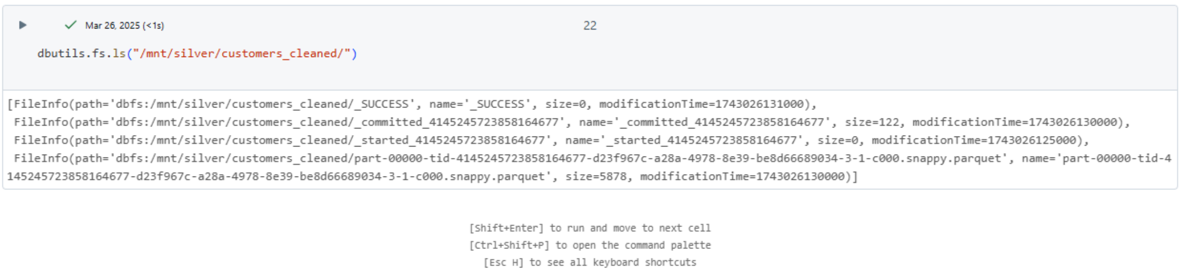
(1) Spark Jobs

```
|19         |Christopher|Baker    |1818 Pine Rd    |Thunder Bay    |ON   |P7A0A1|
|20         |Mia        |Nelson   |1919 Birch Blvd |London         |ON   |N6A0A1|
|21         |Andrew     |Mitchell |2020 Spruce Ln  |Hamilton       |ON   |L8P0A1|
|22         |Harper     |Roberts  |2121 Fir St     |Kitchener      |ON   |N2G0A1|
|23         |Joshua     |Turner   |2222 Redwood Dr |Windsor        |ON   |N9A0A1|
|24         |Evelyn     |Phillips |2323 Cypress Ave|Kingston       |ON   |K7L0A1|
|25         |Daniel     |Campbell |2424 Willow Rd  |St. Catharines|ON   |L2R0A1|
|26         |Abigail    |Parker   |2525 Poplar St  |Barrie         |ON   |L4M0A1|
|27         |James      |Evans    |2626 Ash Blvd   |Guelph         |ON   |N1H0A1|
|28         |Emily      |Edwards  |2727 Beech Dr   |Brantford      |ON   |N3T0A1|
|29         |Michael    |Collins  |2828 Cedar Ln   |Thunder Bay    |ON   |P7B0A1|
|30         |Elizabeth  |Stewart  |2929 Elm St     |Peterborough   |ON   |K9H0A1|
|31         |David      |Sanchez  |3030 Maple Ave  |North Bay      |ON   |P1B0A1|
|32         |Sophia     |Morris   |3131 Oak Dr     |Belleville     |ON   |K8N0A1|
|33         |John       |Rogers   |3232 Pine Rd    |Timmins        |ON   |P4N0A1|
|34         |Olivia     |Reed     |3333 Birch Blvd |Orillia        |ON   |L3V0A1|
|35         |William    |Cook     |3434 Spruce Ln  |Midland        |ON   |L4R0A1|
|36         |Ava        |Morgan   |3535 Fir St     |Collingwood    |ON   |L9Y0A1|
+-----------+-----------+---------+----------------+--------------+-----+------+
only showing top 20 rows
```



```python
# Silver container path
silver_path = "/mnt/silver/customers_cleaned"

# Write the cleaned data to Silver container (parquet format)
df_cleaned.write.mode("overwrite").parquet(silver_path)

print("Data written to Silver container successfully!")
```

```
Data written to Silver container successfully!
```



```python
dbutils.fs.ls("/mnt/silver/customers_cleaned/")
```

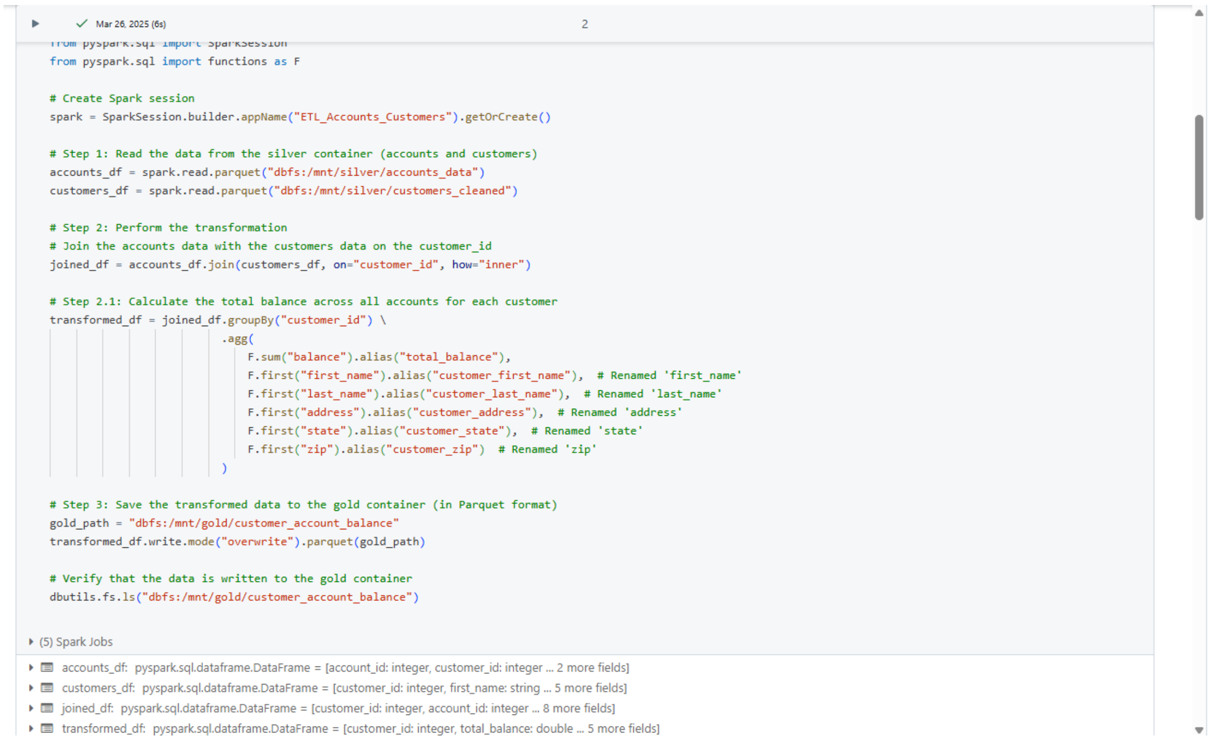
```
[FileInfo(path='dbfs:/mnt/silver/customers_cleaned/_SUCCESS', name='_SUCCESS', size=0, modificationTime=1743026131000),
 FileInfo(path='dbfs:/mnt/silver/customers_cleaned/_committed_4145245723858164677', name='_committed_4145245723858164677', size=122, modificationTime=1743026130000),
 FileInfo(path='dbfs:/mnt/silver/customers_cleaned/_started_4145245723858164677', name='_started_4145245723858164677', size=0, modificationTime=1743026125000),
 FileInfo(path='dbfs:/mnt/silver/customers_cleaned/part-00000-tid-4145245723858164677-d23f967c-a28a-4978-8e39-be8d66689034-3-1-c000.snappy.parquet', name='part-00000-tid-4
145245723858164677-d23f967c-a28a-4978-8e39-be8d66689034-3-1-c000.snappy.parquet', size=5878, modificationTime=1743026130000)]
```

[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts

Step 3: Databricks Activity (ETL from Silver to Gold)

Objective: Apply business logic to generate a refined dataset of customer balances.

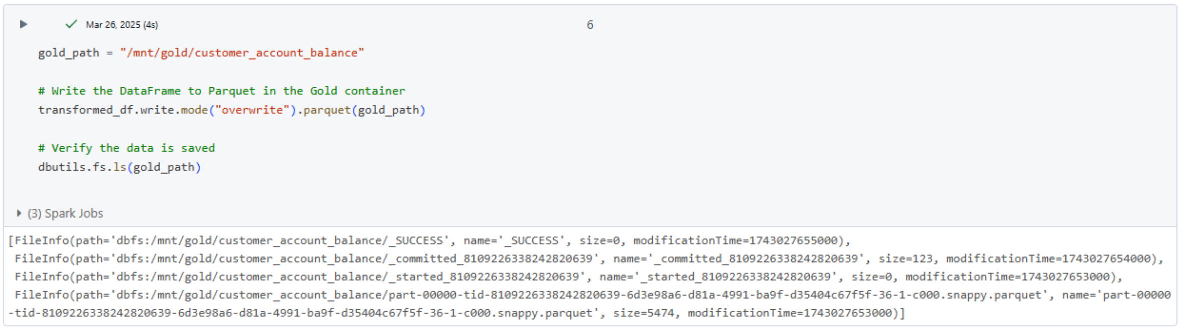
- **Tool:** Azure Databricks Notebook (Notebook 2)
- **Language:** PySpark
- **Operations:**
    - **Data Source:** Read accounts and customers datasets from the Silver container.
    - **Business Logic:**
        - Join accounts with customers on customer_id.
        - Calculate total balance across all accounts for each customer.
        - Retain all original columns from both datasets.
    - **Data Loading:** Write the final dataset into the refined (gold) container as Parquet/Delta files.



Mar 26, 2025 (6s) 2

```
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

# Create Spark session
spark = SparkSession.builder.appName("ETL_Accounts_Customers").getOrCreate()

# Step 1: Read the data from the silver container (accounts and customers)
accounts_df = spark.read.parquet("dbfs:/mnt/silver/accounts_data")
customers_df = spark.read.parquet("dbfs:/mnt/silver/customers_cleaned")

# Step 2: Perform the transformation
# Join the accounts data with the customers data on the customer_id
joined_df = accounts_df.join(customers_df, on="customer_id", how="inner")

# Step 2.1: Calculate the total balance across all accounts for each customer
transformed_df = joined_df.groupBy("customer_id") \
                           .agg(
                               F.sum("balance").alias("total_balance"),
                               F.first("first_name").alias("customer_first_name"),  # Renamed 'first_name'
                               F.first("last_name").alias("customer_last_name"),  # Renamed 'last_name'
                               F.first("address").alias("customer_address"),  # Renamed 'address'
                               F.first("state").alias("customer_state"),  # Renamed 'state'
                               F.first("zip").alias("customer_zip")  # Renamed 'zip'
                           )

# Step 3: Save the transformed data to the gold container (in Parquet format)
gold_path = "dbfs:/mnt/gold/customer_account_balance"
transformed_df.write.mode("overwrite").parquet(gold_path)

# Verify that the data is written to the gold container
dbutils.fs.ls("dbfs:/mnt/gold/customer_account_balance")
```

```
(5) 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]
joined_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, account_id: integer ... 8 more fields]
transformed_df:  pyspark.sql.dataframe.DataFrame = [customer_id: integer, total_balance: double ... 5 more fields]
```



Mar 26, 2025 (4s) 6

```
gold_path = "/mnt/gold/customer_account_balance"

# Write the DataFrame to Parquet in the Gold container
transformed_df.write.mode("overwrite").parquet(gold_path)

# Verify the data is saved
dbutils.fs.ls(gold_path)
```

```
(3) Spark Jobs
[FileInfo(path='dbfs:/mnt/gold/customer_account_balance/_SUCCESS', name='_SUCCESS', size=0, modificationTime=1743027655000),
 FileInfo(path='dbfs:/mnt/gold/customer_account_balance/_committed_8109226338242820639', name='_committed_8109226338242820639', size=123, modificationTime=1743027654000),
 FileInfo(path='dbfs:/mnt/gold/customer_account_balance/_started_8109226338242820639', name='_started_8109226338242820639', size=0, modificationTime=1743027653000),
 FileInfo(path='dbfs:/mnt/gold/customer_account_balance/part-00000-tid-8109226338242820639-6d3e98a6-d81a-4991-ba9f-d35404c67f5f-36-1-c000.snappy.parquet', name='part-00000
-tid-8109226338242820639-6d3e98a6-d81a-4991-ba9f-d35404c67f5f-36-1-c000.snappy.parquet', size=5474, modificationTime=1743027653000)]
```

Step 4: Write Transformed Data to Azure SQL Database

Objective:

Persist the final transformed customer-account data (stored in the Gold container) into an Azure SQL Database table for further analytics, reporting, or dashboard integration.

Steps:

1. **Read the Transformed Data from Gold Container:**

gold_path = "/mnt/gold/customer_account_balance"

df_gold = spark.read.parquet(gold_path)

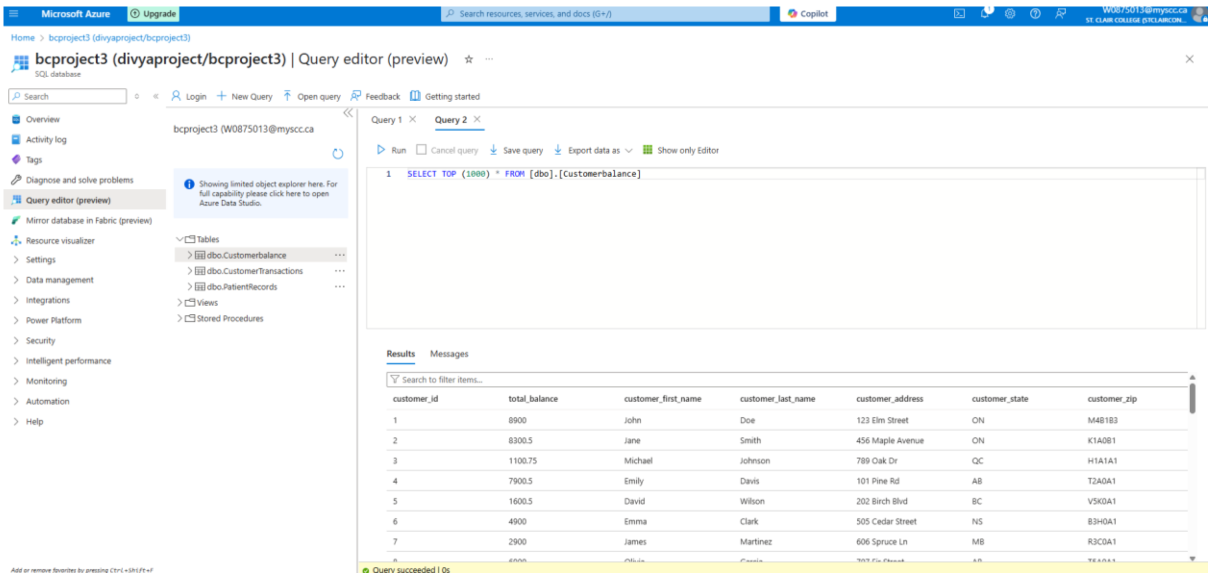
2. **Configure Azure SQL Database Connection:**

jdbc_url = "jdbc:sqlserver://divyaproject.database.windows.net:1433;database=YourDatabaseName"

```
connection_properties = {
    "user": "YourSQLUsername",
    "password": "YourSQLPassword",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}
```

3. **Write the DataFrame to Azure SQL Database:**

```
df_gold.write.jdbc(
    url=jdbc_url,
    table="CustomerAccountBalance",  # Replace with your target table name
    mode="overwrite",               # Use "append" for incremental loads
    properties=connection_properties
)
```



Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

```
SELECT TOP (1000) * FROM [dbo].[Customerbalance]
```

customer_id	total_balance	customer_first_name	customer_last_name	customer_address	customer_state	customer_zip
1	8900	John	Doe	123 Elm Street	ON	M4B1B3
2	8300.5	Jane	Smith	456 Maple Avenue	ON	K1A0B1
3	1100.75	Michael	Johnson	789 Oak Dr	QC	H1A1A1
4	7900.5	Emily	Davis	101 Pine Rd	AB	T2A0A1
5	1600.5	David	Wilson	202 Birch Blvd	BC	V5K0A1
6	4900	Emma	Clark	505 Cedar Street	NS	B3H0A1
7	2900	James	Martinez	606 Spruce Ln	MB	R3C0A1

Query succeeded | 0s

Add or remove favorites by pressing Ctrl+Shift+F