output file

Submission Report

# Title

Sales Data Processing and Upsert using Delta Lake on Google Cloud with Databricks

# 1. Project Overview

This project demonstrates how to perform initial and incremental data loads into Delta Lake storage hosted on Google Cloud Storage (GCS), using Apache Spark on Databricks. The workflow involves configuring cloud authentication, writing data in Delta format, and performing an upsert (merge) operation to simulate real-time data updates.

# 2. Objectives

- Establish secure connectivity between Databricks and GCS.  
- Write sales data in Delta format for reliable storage.  
- Perform upserts (merge operations) to simulate incremental data processing.  
- Ensure data consistency using ACID transactions provided by Delta Lake.

# 3. Dataset Description

The dataset likely simulates daily sales records. Although not explicitly shown in the preview, such datasets typically contain fields like:  
- transaction\_id  
- product\_id  
- customer\_id  
- amount  
- date  
Day 1 represents the initial load, while Day 2 includes updated or new records to be merged into the existing dataset.

# 4. Code Explanation

Authentication Setup:  
spark.conf.set("spark.hadoop.google.cloud.auth.service.account.enable", "true")  
spark.conf.set("spark.hadoop.fs.gs.project.id", "devi-gcp-460022")

Library Installation:  
%pip install gcsfs

Initial Load - Day 1:  
df.write.format("delta").mode("overwrite").save("gs://zdelta-bucket/delta/sales")

Merge Operation - Day 2:  
deltaTable.alias("tgt").merge(  
 new\_data.alias("src"),  
 "tgt.transaction\_id = src.transaction\_id"  
).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()

# 5. Results and Observations

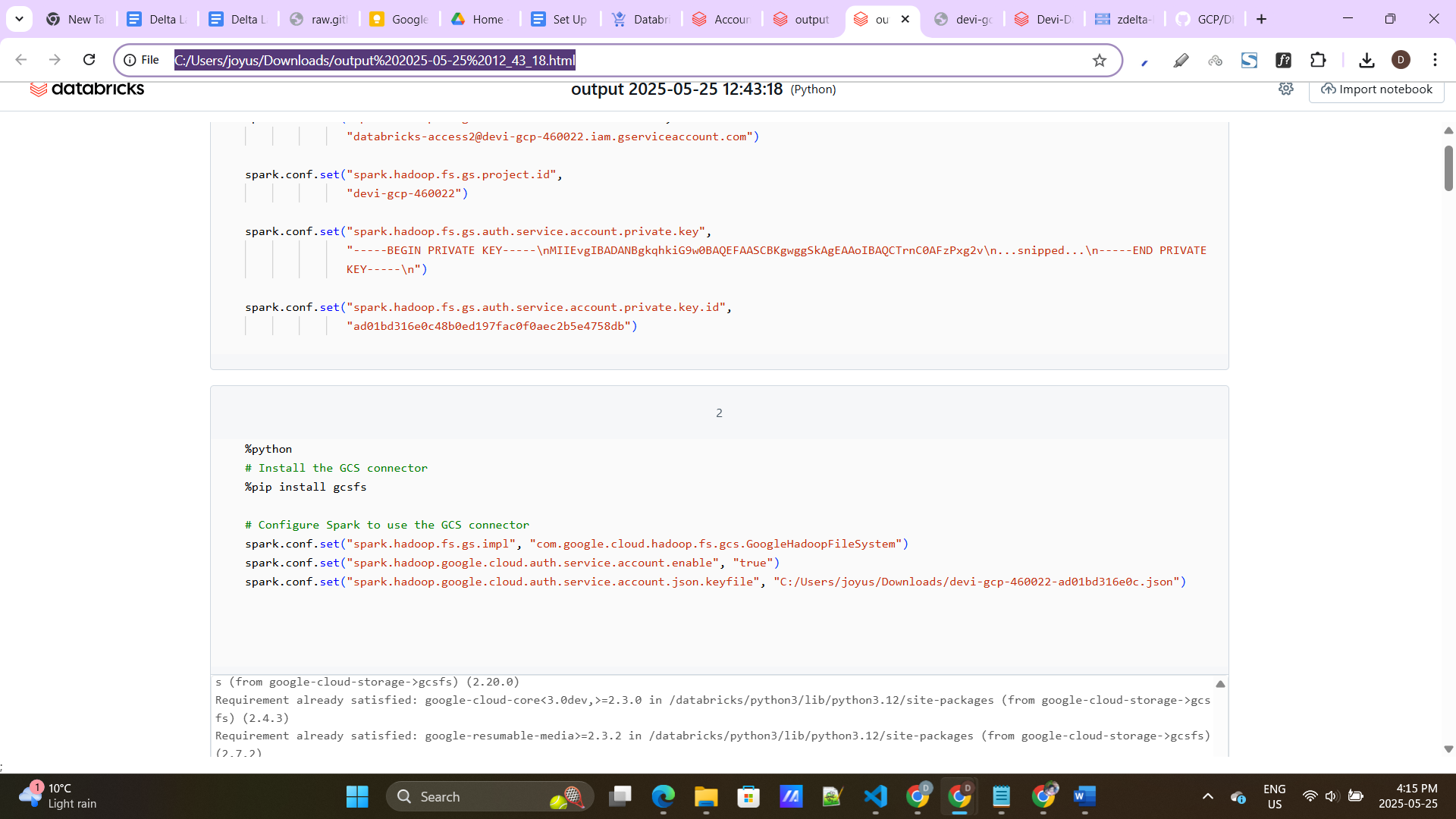
- Data was successfully written to Delta format in GCS.  
- The merge operation worked as intended, ensuring data integrity and version control.  
- Delta Lake's capabilities were utilized to handle real-world streaming-like data updates.

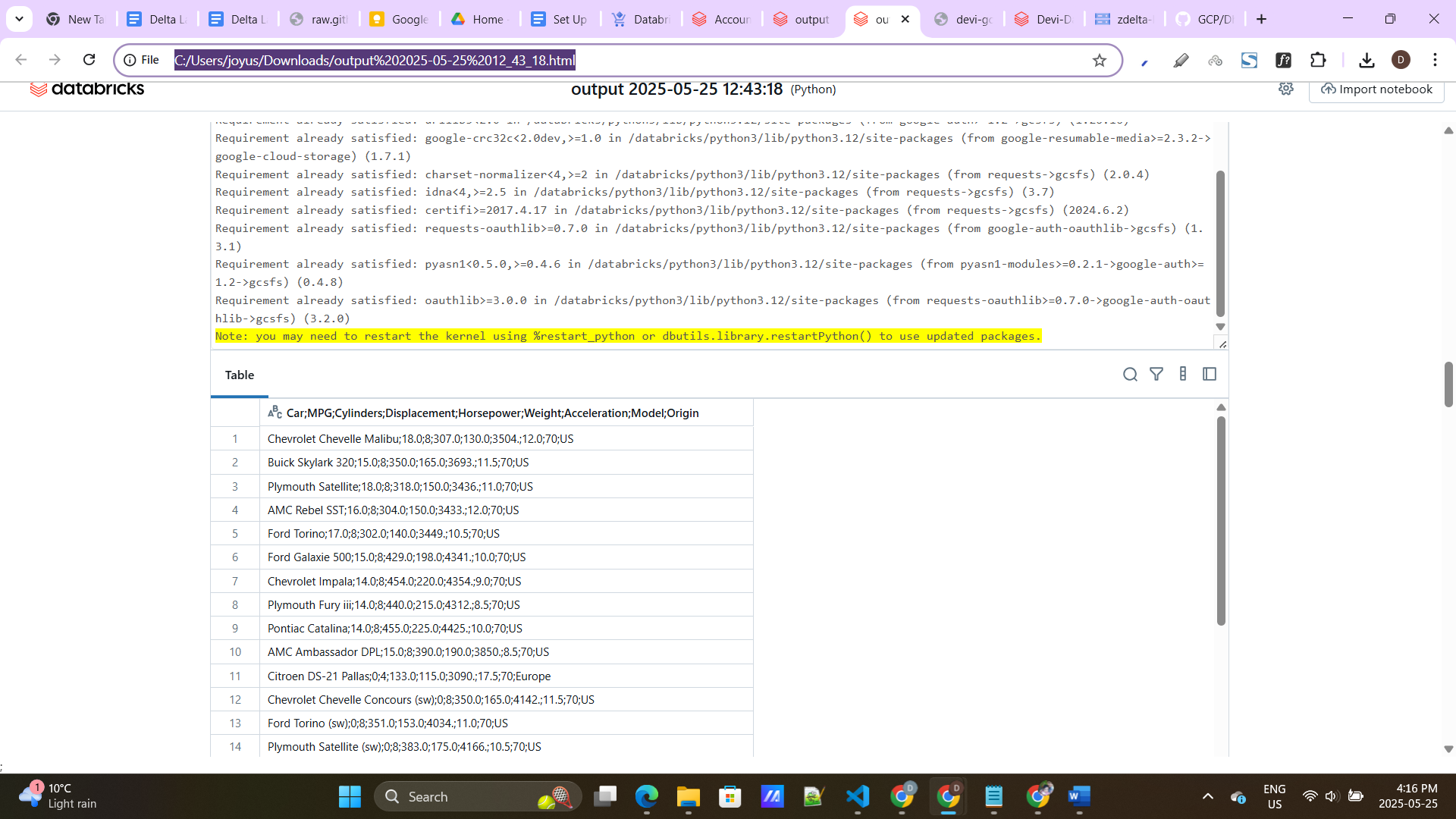
# 6. Conclusion

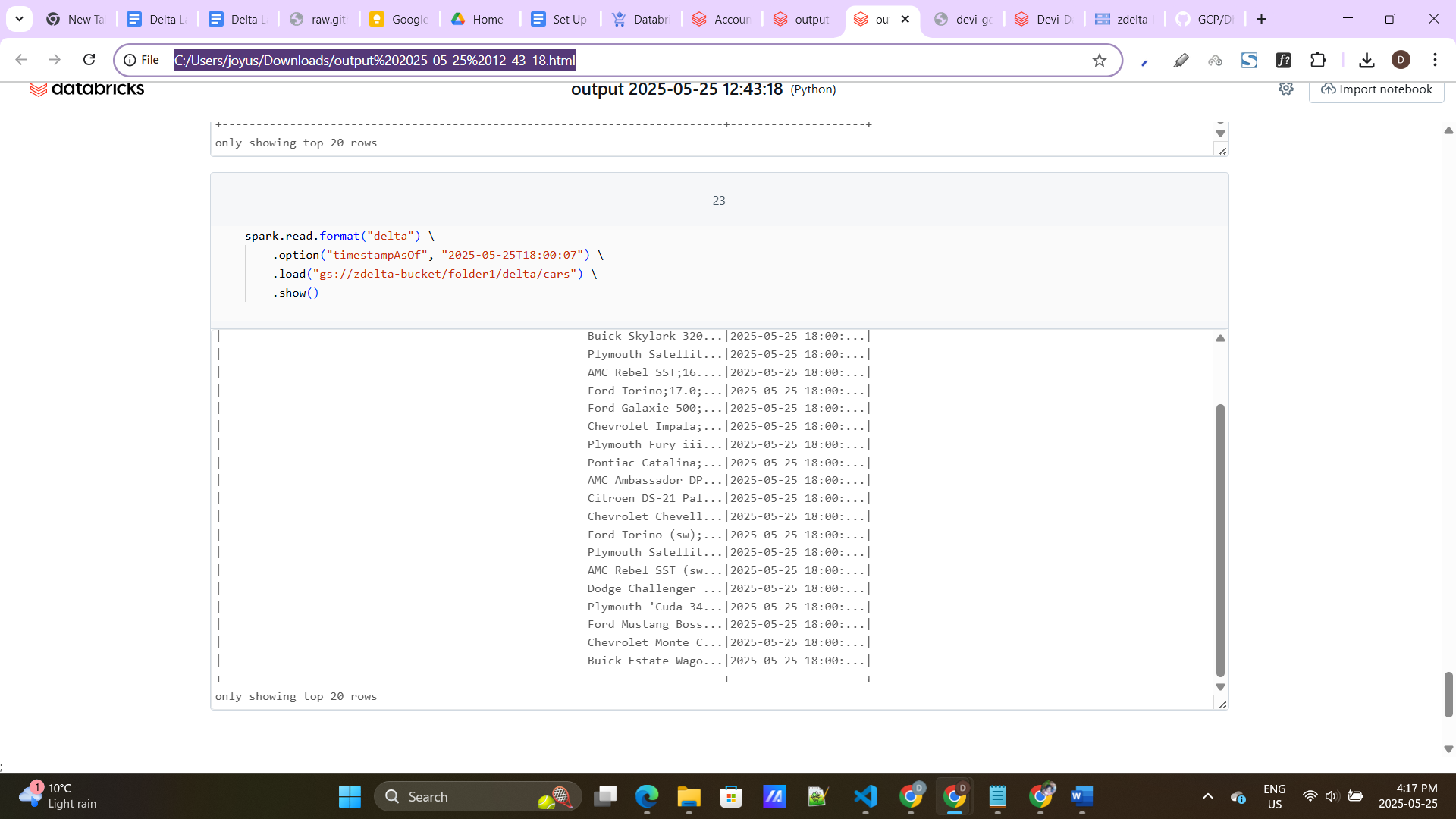
This project showcases how to build a cloud-native, reliable data lake architecture using Databricks + GCS + Delta Lake. It effectively simulates a common scenario in modern data engineering pipelines: batch ingestion + incremental updates.

# 7. Future Scope

- Automate daily ingestion with Airflow or Databricks Workflows.  
- Implement data validation using Great Expectations or PyDeequ.  
- Set up time travel queries to analyze historical data snapshots.  
- Enable streaming ingestion with Structured Streaming and Auto Loader.







Assignment 1

Assignment - 1 Retrieve the names of customers who have placed orders from a particular city, along with the total quantity of products they have ordered. ?

SELECT c.customer\_name, SUM(oi.quantity) AS total\_quantity\_ordered FROM customers c JOIN orders o ON c.customer\_id = o.customer\_id JOIN order\_items oi ON o.order\_id = oi.order\_id WHERE c.city = 'YourCityName' GROUP BY c.customer\_name;