ASSIGNMENT - 4 : DATA WAREHOUSING LAB

Submitted by : Shivie Saksenaa

1. Objective

The objective of this lab was to understand how **Google Cloud Storage (GCS)** and **BigQuery** integrate to provide a complete cloud-based data storage and analytics solution.
The goal was to:

- Store data in **GCS**,
- Query it directly using an **external table** in **BigQuery**, and
- Compare performance with a **managed (partitioned) BigQuery table**.

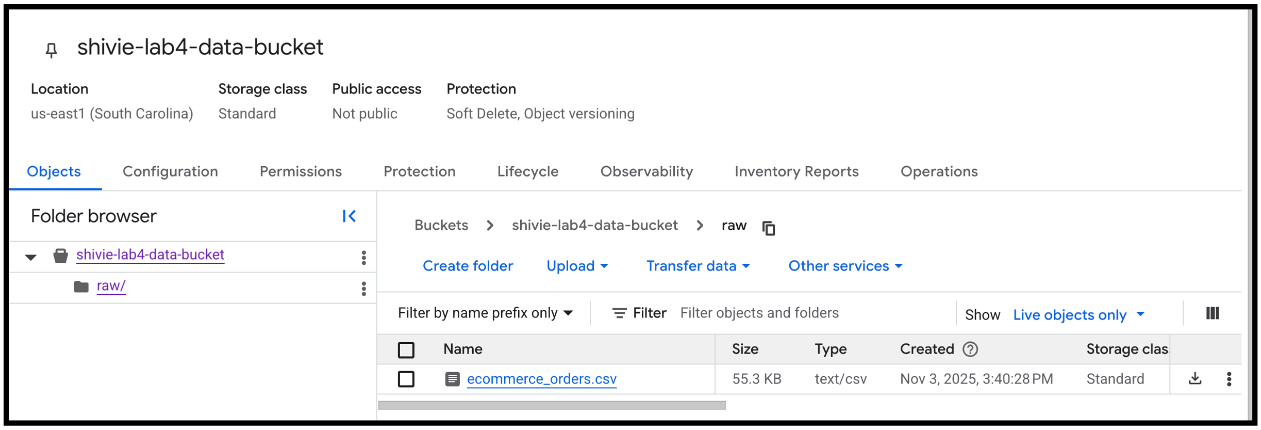
2. Tools and Services Used

Tool	Purpose
Google Cloud Storage (GCS)	Store raw CSV data securely
BigQuery	Data warehousing and SQL-based analytics
GCP Project ID	clean-equinox-472523-c4
Dataset	ecommerce_dw
Region	us-east1
Dataset used	Ecommerce_Orders.csv

3. Steps Performed

Step 1: Upload data to Google Cloud Storage

1. Created bucket : shivie-lab4-data-bucket
2. Added folder : raw/
3. Uploaded file : ecommerce_orders.csv
4. Final GCS path:
   gs://shivie-lab4-data-bucket/raw/ecommerce_orders.csv



shivie-lab4-data-bucket

Location	Storage class	Public access	Protection
us-east1 (South Carolina)	Standard	Not public	Soft Delete, Object versioning

shivie-lab4-data-bucket
raw/

Buckets > shivie-lab4-data-bucket > raw

	Name	Size	Type	Created	Storage clas
□	ecommerce_orders.csv	55.3 KB	text/csv	Nov 3, 2025, 3:40:28 PM	Standard

Step 2: Created External Table in BigQuery

1. In BigQuery Console , I created the Table **orders.ext**

   - **Source:** Google Cloud Storage
   - **URI:** gs://shivie-lab4-data-bucket/raw/ecommerce_orders.csv
   - **File Format:** CSV
   - **Auto Detect Schema:** Enabled
   - **Destination Dataset:** ecommerce_dw
   - **Table Type:** External Table

Result : Table was created : clean-equinox-472523-c4.ecommerce_dw.orders_ext

Step 3: Created Managed (Partitioned) Table

1. In BigQuery, I created the Table **orders_partitioned**

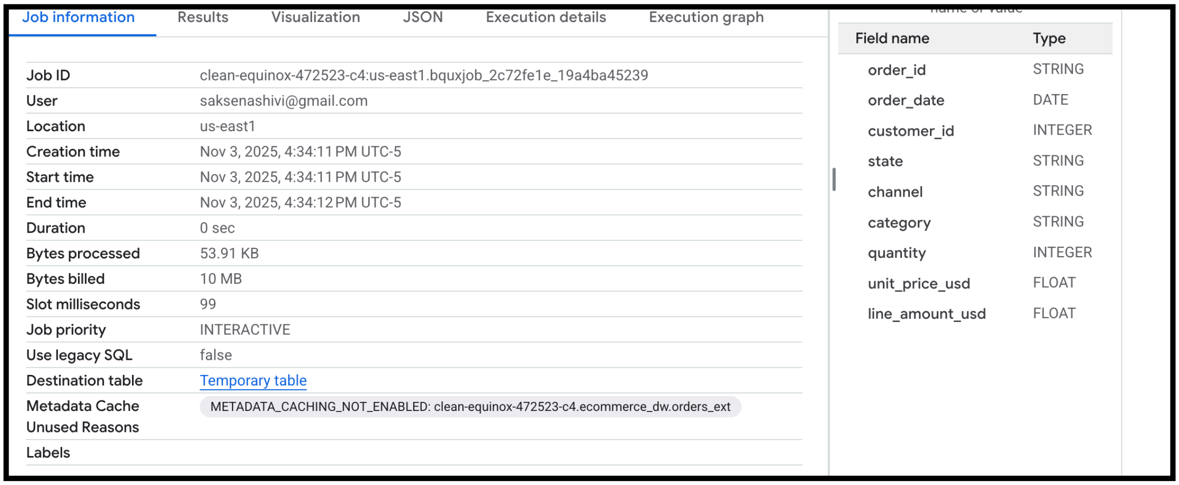
   - **Source:** Same GCS file
   - **Destination Table:** ecommerce_dw.orders
   - **Table Type:** Native (Managed)
   - **Partition By:** order_date
   - **Cluster By:** state, category

Result: Table Created: clean-equinox-472523-c4.ecommerce_dw.orders

AIM: Understand the difference in performance of External Table and Native table.
Query Tested: on both the tables - External and the Native One

```
SELECT
  DATE_TRUNC(order_date, MONTH) AS month,
  ROUND(SUM(line_amount_usd), 2) AS total_revenue
FROM `clean-equinox-472523-c4.ecommerce_dw.orders_ext`
GROUP BY month
ORDER BY month;
```

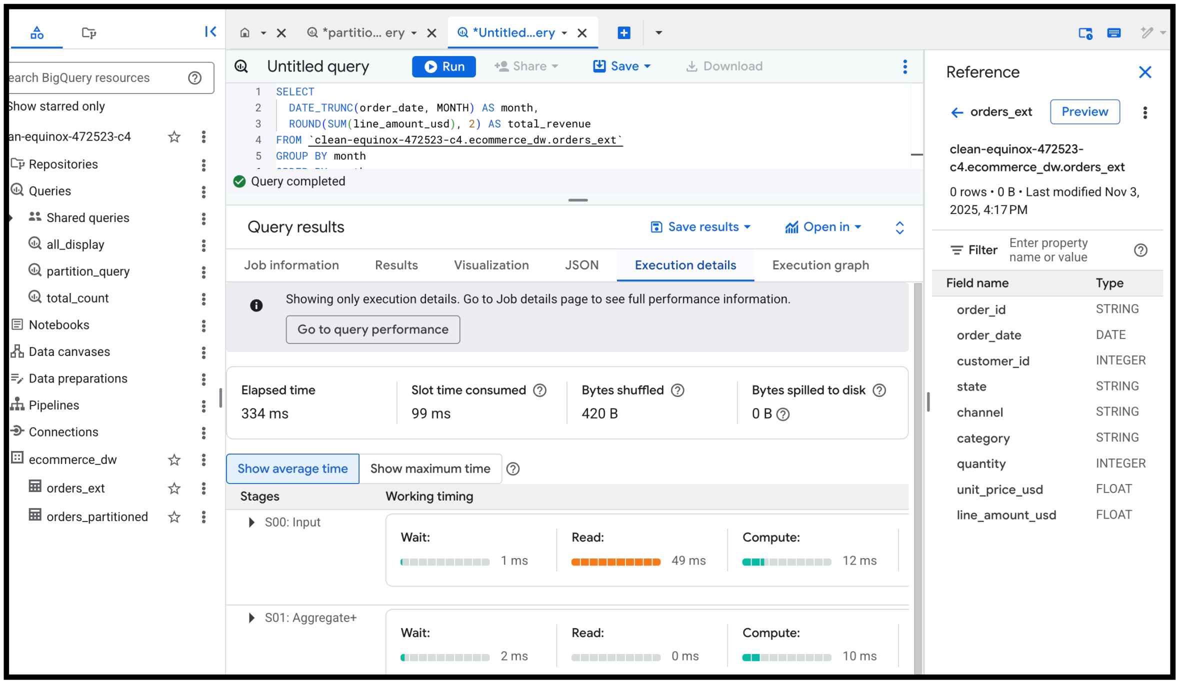
4. Result



Job ID	clean-equinox-472523-c4:us-east1.bquxjob_2c72fe1e_19a4ba45239
User	saksenashivi@gmail.com
Location	us-east1
Creation time	Nov 3, 2025, 4:34:11 PM UTC-5
Start time	Nov 3, 2025, 4:34:11 PM UTC-5
End time	Nov 3, 2025, 4:34:12 PM UTC-5
Duration	0 sec
Bytes processed	53.91 KB
Bytes billed	10 MB
Slot milliseconds	99
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table
Metadata Cache Unused Reasons	METADATA_CACHING_NOT_ENABLED: clean-equinox-472523-c4.ecommerce_dw.orders_ext
Labels	

name or value

Field name	Type
order_id	STRING
order_date	DATE
customer_id	INTEGER
state	STRING
channel	STRING
category	STRING
quantity	INTEGER
unit_price_usd	FLOAT
line_amount_usd	FLOAT



```
SELECT
  DATE_TRUNC(order_date, MONTH) AS month,
  ROUND(SUM(line_amount_usd), 2) AS total_revenue
FROM `clean-equinox-472523-c4.ecommerce_dw.orders_ext`
GROUP BY month
```

Query completed

Showing only execution details. Go to Job details page to see full performance information.

Elapsed time	Slot time consumed	Bytes shuffled	Bytes spilled to disk
334 ms	99 ms	420 B	0 B

Stages	Wait:	Read:	Compute:
S00: Input	1 ms	49 ms	12 ms
S01: Aggregate+	2 ms	0 ms	10 ms

clean-equinox-472523-c4.ecommerce_dw.orders_ext

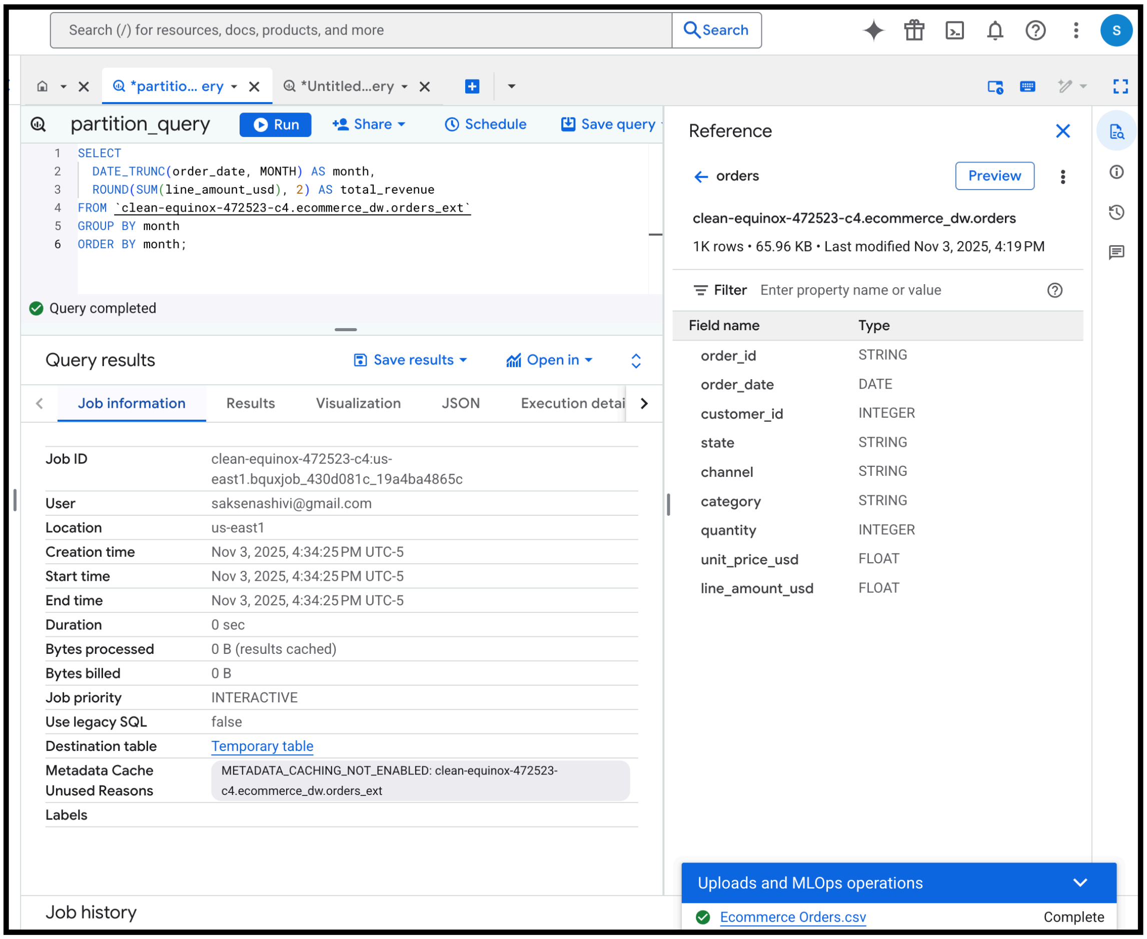
0 rows • 0 B • Last modified Nov 3, 2025, 4:17 PM

Field name	Type
order_id	STRING
order_date	DATE
customer_id	INTEGER
state	STRING
channel	STRING
category	STRING
quantity	INTEGER
unit_price_usd	FLOAT
line_amount_usd	FLOAT

Observation and Inference – External Table (orders_ext)

- When I ran the query on the external table, BigQuery read the data directly from my GCS file (gs://shivie-lab4-data-bucket/raw/ecommerce_orders.csv).
  The job processed 53.91 KB and took around 334 ms. Since the table only references the CSV, BigQuery had to scan the entire file every time I ran the query.
  Caching, partitioning, and clustering weren't available, and most of the time was spent reading data from GCS.
- From this, I inferred that external tables are great for quick, ad-hoc analysis of raw files but not ideal for repeated or large analytical queries. They provide flexibility without needing to load data, but performance is slower compared to a managed table

Managed Table



```
SELECT
  DATE_TRUNC(order_date, MONTH) AS month,
  ROUND(SUM(line_amount_usd), 2) AS total_revenue
FROM `clean-equinox-472523-c4.ecommerce_dw.orders_ext`
GROUP BY month
ORDER BY month;
```

Query completed

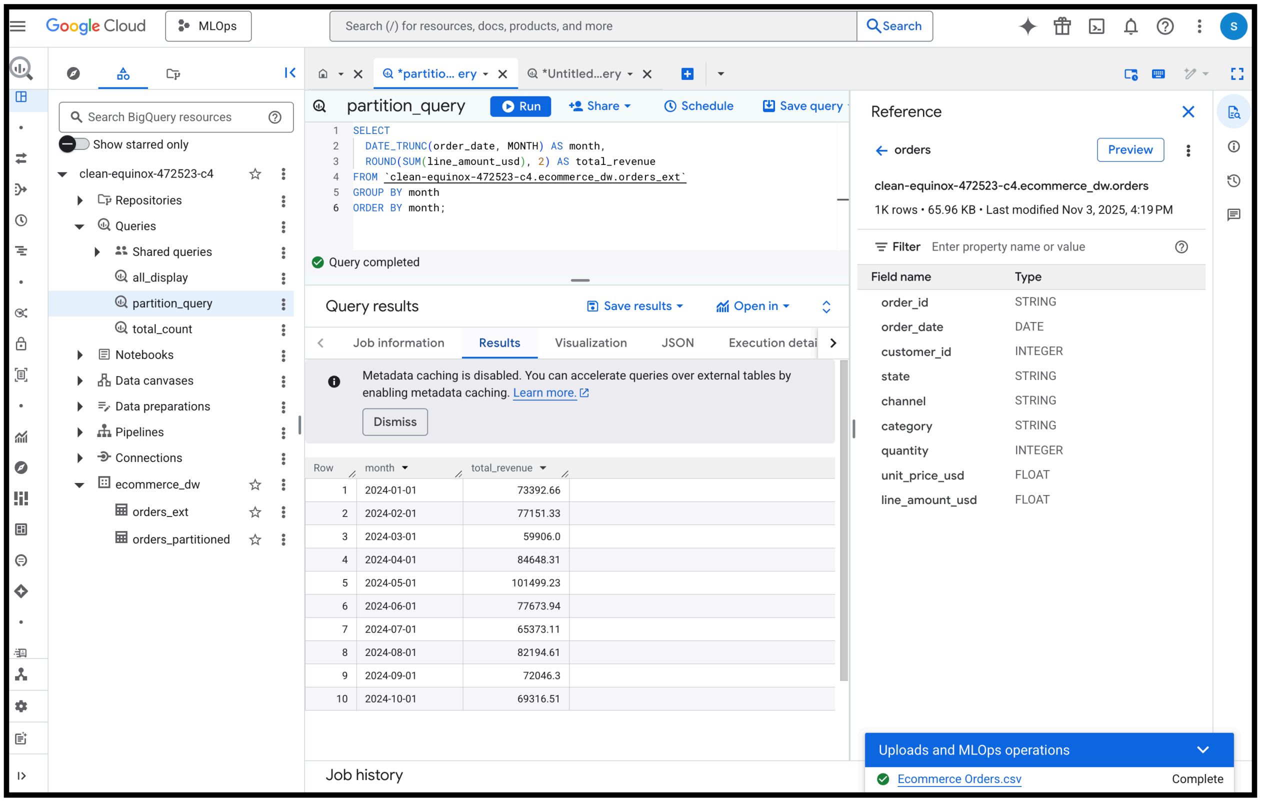
Job ID	clean-equinox-472523-c4:us-east1.bquxjob_430d081c_19a4ba4865c
User	saksenashivi@gmail.com
Location	us-east1
Creation time	Nov 3, 2025, 4:34:25 PM UTC-5
Start time	Nov 3, 2025, 4:34:25 PM UTC-5
End time	Nov 3, 2025, 4:34:25 PM UTC-5
Duration	0 sec
Bytes processed	0 B (results cached)
Bytes billed	0 B
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table
Metadata Cache Unused Reasons	METADATA_CACHING_NOT_ENABLED: clean-equinox-472523-c4.ecommerce_dw.orders_ext
Labels	

clean-equinox-472523-c4.ecommerce_dw.orders

1K rows • 65.96 KB • Last modified Nov 3, 2025, 4:19 PM

Field name	Type
order_id	STRING
order_date	DATE
customer_id	INTEGER
state	STRING
channel	STRING
category	STRING
quantity	INTEGER
unit_price_usd	FLOAT
line_amount_usd	FLOAT

Ecommerce Orders.csv Complete



```
SELECT
  DATE_TRUNC(order_date, MONTH) AS month,
  ROUND(SUM(line_amount_usd), 2) AS total_revenue
FROM `clean-equinox-472523-c4.ecommerce_dw.orders_ext`
GROUP BY month
ORDER BY month;
```

Query completed

Metadata caching is disabled. You can accelerate queries over external tables by enabling metadata caching. Learn more.

Row	month	total_revenue
1	2024-01-01	73392.66
2	2024-02-01	77151.33
3	2024-03-01	59906.0
4	2024-04-01	84648.31
5	2024-05-01	101499.23
6	2024-06-01	77673.94
7	2024-07-01	65373.11
8	2024-08-01	82194.61
9	2024-09-01	72046.3
10	2024-10-01	69316.51

clean-equinox-472523-c4.ecommerce_dw.orders

1K rows • 65.96 KB • Last modified Nov 3, 2025, 4:19 PM

Field name	Type
order_id	STRING
order_date	DATE
customer_id	INTEGER
state	STRING
channel	STRING
category	STRING
quantity	INTEGER
unit_price_usd	FLOAT
line_amount_usd	FLOAT

Ecommerce Orders.csv Complete

Observation and Inference – Managed Table (orders)

- When I ran the same query on the managed, partitioned table, BigQuery used its internal storage instead of reading from GCS.
  The job showed 0 bytes processed and finished instantly because the data was already stored in BigQuery and cached.
  Since this table is partitioned by order_date and clustered by state and category, BigQuery only scans the required partitions, making queries faster and more efficient.

- From this, I inferred that managed tables are much better for repeated and large-scale analytics. They support partitioning, clustering, and caching, which reduce scan cost and improve performance compared to external tables.