

# ASSIGNMENT - 4 : DATA WAREHOUSING LAB

Submitted by : Shvie 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
<b>Google Cloud Storage (GCS)</b>	Store raw CSV data securely
<b>BigQuery</b>	Data warehousing and SQL-based analytics
<b>GCP Project ID</b>	clean-equinox-472523-c4
<b>Dataset</b>	ecommerce_dw
<b>Region</b>	us-east1
<b>Dataset used</b>	Ecommerce_Orders.csv

## 3. Steps Performed

### Step 1: Upload data to Google Cloud Storage

1. Created bucket : shvie-lab4-data-bucket
2. Added folder : raw/
3. Uploaded file : ecommerce\_orders.csv
4. Final GCS path:  
[gs://shvie-lab4-data-bucket/raw/ecommerce\\_orders.csv](gs://shvie-lab4-data-bucket/raw/ecommerce_orders.csv)

The screenshot shows the Google Cloud Storage interface. At the top, it displays the bucket details: Location us-east1 (South Carolina), Storage class Standard, Public access Not public, and Protection Soft Delete, Object versioning. Below this is a navigation bar with tabs: Objects (which is selected), Configuration, Permissions, Protection, Lifecycle, Observability, Inventory Reports, and Operations. The main area is titled 'Folder browser' and shows a tree view with 'shivie-lab4-data-bucket' expanded, revealing a 'raw/' folder. To the right of the tree view is a breadcrumb trail: Buckets > shivie-lab4-data-bucket > raw. Below the breadcrumb are buttons for Create folder, Upload, Transfer data, and Other services. There are also filters for name prefix, type, and status (Live objects only). A table lists the contents of the 'raw/' folder, showing one item: 'ecommerce\_orders.csv'.

## 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 information		Results	Visualization	JSON	Execution details	Execution graph	Name or value
Field name	Type						
Job ID	clean-equinox-472523-c4:us-east1.bqjob_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	METADATA_CACHING_NOT_ENABLED: clean-equinox-472523-c4.ecommerce_dw.orders_ext						
Unused Reasons							
Labels							

The screenshot shows the BigQuery interface with a completed query named "Untitled query". The query results in 0 rows. The execution details show an elapsed time of 334 ms, with most time spent in Compute (12 ms). The schema on the right lists fields: order\_id (STRING), order\_date (DATE), customer\_id (INTEGER), state (STRING), channel (STRING), category (STRING), quantity (INTEGER), unit\_price\_usd (FLOAT), and 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

The screenshot shows a data exploration interface with the following details:

- Search Bar:** Search (/) for resources, docs, products, and more.
- Query Editor:** A query named "partition\_query" is displayed:

```
1 SELECT
2   DATE_TRUNC(order_date, MONTH) AS month,
3   ROUND(SUM(line_amount_usd), 2) AS total_revenue
4 FROM `clean-equinox-472523-c4.ecommerce_dw.orders_ext`
5 GROUP BY month
6 ORDER BY month;
```
- Status:** Query completed.
- Job Information:** Details about the job execution, including Job ID, User, Location, Creation time, Start time, End time, Duration, Bytes processed, Bytes billed, Job priority, Use legacy SQL, Destination table (Temporary table), Metadata Cache, Unused Reasons, and Labels.
- Reference:** Reference to the "orders" table from the "clean-equinox-472523-c4.ecommerce\_dw.orders" dataset.
- Field Definition:** A table showing field names and types:

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
- Uploads and MLOps operations:** Shows an upload for "Ecommerce Orders.csv" which is complete.

The screenshot shows the Google Cloud BigQuery interface with the following details:

- Search Bar:** Search (/) for resources, docs, products, and more.
- Left Sidebar:** Navigation pane showing "Google Cloud" and "MLOps". The "Queries" section is expanded, showing "partition\_query" which is selected.
- Query Editor:** A query named "partition\_query" is displayed:

```
1 SELECT
2   DATE_TRUNC(order_date, MONTH) AS month,
3   ROUND(SUM(line_amount_usd), 2) AS total_revenue
4 FROM `clean-equinox-472523-c4.ecommerce_dw.orders_ext`
5 GROUP BY month
6 ORDER BY month;
```
- Status:** Query completed.
- Job Information:** A message indicates that metadata caching is disabled and provides a link to learn more.
- Results:** A table showing the results of the query:

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
- Reference:** Reference to the "orders" table from the "clean-equinox-472523-c4.ecommerce\_dw.orders" dataset.
- Field Definition:** A table showing field names and types:

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
- Uploads and MLOps operations:** Shows an upload for "Ecommerce Orders.csv" which is 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.