GOOGLE CLOUD PLATFORM FIRESTORE FINAL PROJECT

Google Cloud

Building a Scalable NoSQL Database with Firestore

Nandan Prabhu
CSDAIA24GP003
2320667

Introduction

GCP, or Google Cloud Platform, is a suite of cloud computing services offered by Google. It provides a variety of tools and resources that allow businesses and organizations to run their applications and data centres on Google's infrastructure.

Here's a quick breakdown of GCP:

- **Services:** It offers a range of services including compute, storage, data analytics, and machine learning. Think of storage for housing your data, compute for running applications, analytics for crunching numbers, and machine learning for building intelligent models.
- **Scalability:** GCP is known for its scalability, meaning you can easily adjust your resource usage up or down based on your needs.
- **Security:** Security is a major focus of GCP, with Google using the same infrastructure for GCP that it uses for its own products, like Gmail and Search.
- **Benefits:** Businesses use GCP for a variety of reasons, including cost savings, increased agility, and access to advanced technologies like AI and machine learning.

Core GCP Services:

- **Compute Engine:** This service provides virtual machines (VMs) that you can use to run your workloads on Google's infrastructure. VMs offer a familiar computing environment for developers.
- **Cloud Functions:** This serverless offering lets you run code without having to manage servers yourself. It's ideal for event-driven applications or small tasks.

- **Kubernetes Engine:** This managed Kubernetes service allows you to deploy and manage containerized applications in Google Cloud. Kubernetes is a popular container orchestration platform.
- **Cloud Storage:** This offers scalable and durable storage for a variety of data types, from archives to frequently accessed objects.
- **BigQuery:** A data warehouse service for large datasets. BigQuery allows you to analyse massive datasets quickly and efficiently.

Advantages of GCP:

- **Cost-Effectiveness:** GCP offers a pay-as-you-go pricing model, so you only pay for the resources you use.
- **Scalability:** Easily scale resources up or down to meet your needs.
- **Security:** GCP offers robust security features to protect your data and applications.
- **Reliability:** Google's global infrastructure ensures high availability and reliability for your applications.
- **Open Source Friendly:** GCP embraces open-source technologies and offers integration with many popular open-source tools.

FIRESTORE

Firestore, a key component of Firebase and Google Cloud Platform (GCP), is a flexible and scalable NoSQL database designed for mobile, web, and server development. Here's a comprehensive breakdown of Firestore:

What it is:

- **NoSQL Document Database:** Firestore stores data in JSON-like documents, offering a flexible schema compared to traditional relational databases.
- **Cloud-Hosted:** You don't need to manage or maintain servers yourself. Google takes care of the infrastructure.
- **Offline Support:** Firestore caches data locally on devices, enabling your app to function even without an internet connection.

Key Features:

- **Rich Querying:** Firestore allows complex queries to retrieve specific data efficiently. You can filter, sort, and perform aggregations on your data.
- **ACID Transactions:** Firestore ensures Atomicity, Consistency, Isolation, and Durability (ACID) for data operations, guaranteeing data integrity.
- **Offline Persistence:** As mentioned earlier, Firestore caches data locally, allowing users to interact with data even when offline. Changes are synced back to the cloud when the device regains connectivity.
- **Real-time Updates:** Firestore keeps data synchronized across connected devices in real-time using listeners. This is ideal for collaborative apps or applications where data updates need to be reflected immediately for all users.
- **Scalability:** Firestore scales seamlessly to accommodate growing data volumes and user bases. You won't need to worry about managing server capacity.
- **Security:** Firestore integrates with Google Cloud's security tools, allowing you to implement granular access controls to protect your data.

- **Integration with other GCP Services:** Firestore integrates well with other GCP services like Cloud Functions for building serverless triggers based on database events.

Use Cases:

- **Mobile and Web Apps:** Firestore's offline capabilities and real-time updates make it a popular choice for building dynamic mobile and web applications.
- **Social Applications:** Features like real-time data updates are perfect for social apps where users need to see changes instantaneously.
- **E-commerce Applications:** Firestore can efficiently manage product catalogs, user data, and shopping carts.
- **Real-time Collaboration:** Firestore is well-suited for collaborative applications where multiple users need to work with the same data simultaneously.

Benefits of using Firestore:

- **Flexibility:** The document-based schema allows you to store data in a way that aligns with your application's needs.
- **Scalability:** Easily handle growing data volumes without worrying about infrastructure limitations.
- **Offline Support:** Provides a seamless user experience even in situations with unreliable internet connectivity.
- **Real-time Updates:** Keeps all connected devices in sync, offering an up-to-date view of the data.
- **Security:** Leverage Google Cloud's robust security features to protect your data.

TASKS:

1.Create a firestore database

2.Create one collection

3.Add few documents

4.Create python script which will read data from a file to firestore.

OBJECTIVE:

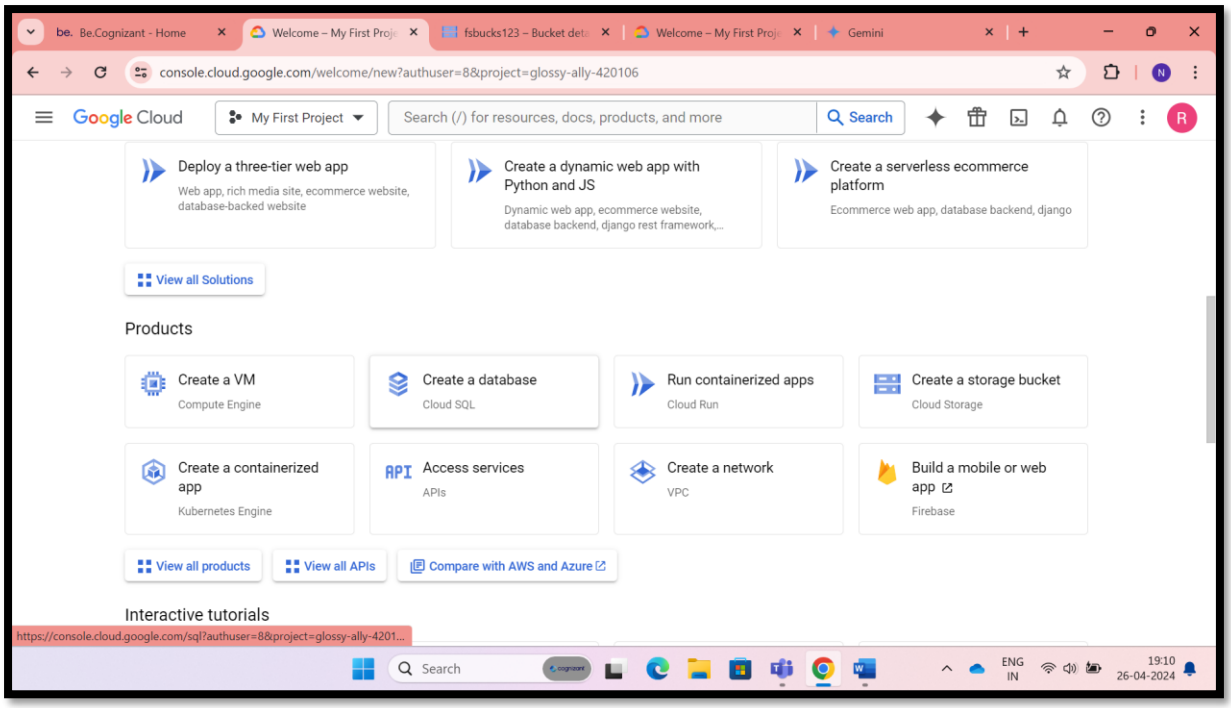
To set up a Firestore database, create a collection, add documents to the collection, and develop a Python script to import data from a file into Firestore.

WAYS TO INTERACT WITH GCP:

Google Cloud gives you three basic ways to interact with the services and resources.

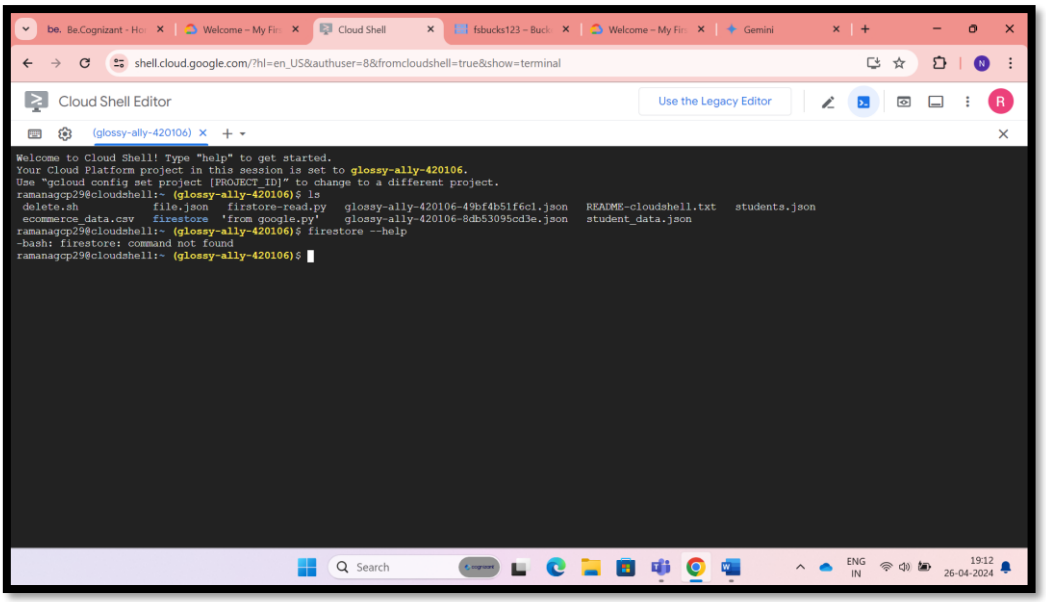
1.Google cloud console (GUI):

- The **Google Cloud Console** is a web-based interface that allows you to manage GCP resources visually.
- You can create, configure, and monitor services, virtual machines, databases, and more.
- It’s beginner-friendly and provides an intuitive way to interact with GCP.
- When you use the Google Cloud console, you either create a new project or choose an existing project, and then use the resources that you create in the context of that project.



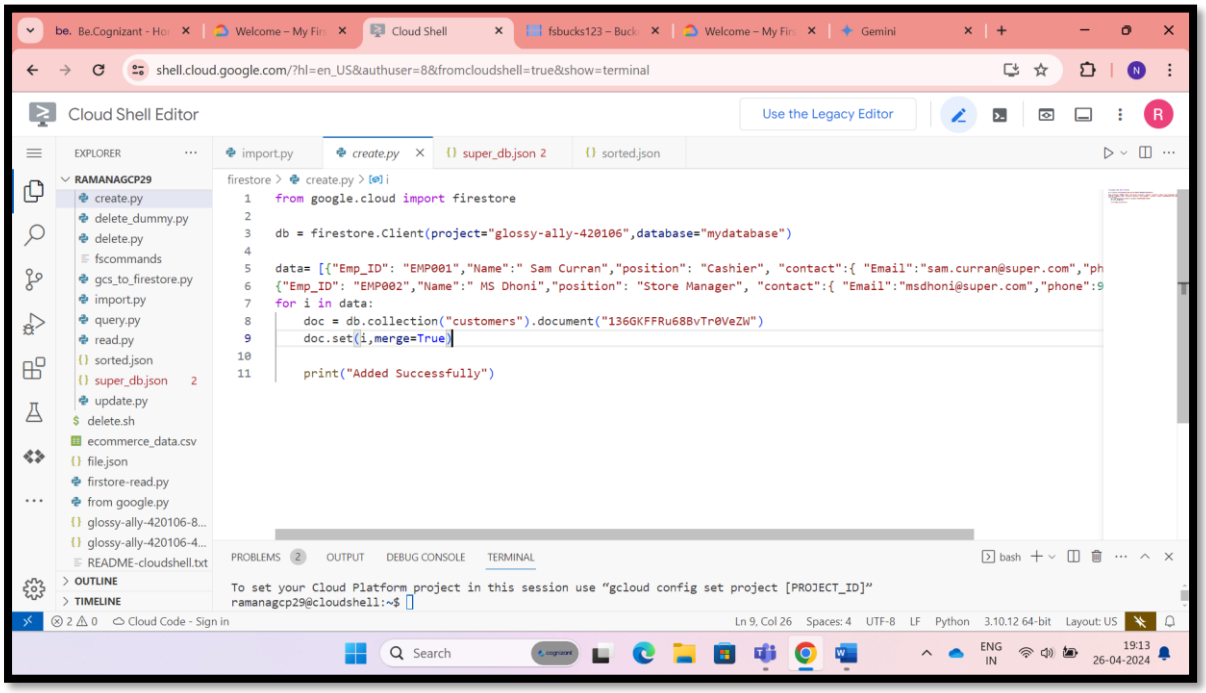
2. Command-Line Interface (CLI):

- If you prefer to work at the command line, you can perform most Google Cloud tasks by using the google cloud cli. The gcloud CLI lets you manage development workflow and Google Cloud resources in a terminal window.
- The **gcloud CLI tool**, part of the **Google Cloud SDK**, lets you interact with GCP services via the command line.
- Use it for tasks like creating instances, managing storage, configuring networking, and deploying applications.
- The CLI is powerful for scripting and automation.



3. Python Client Libraries:

- GCP offers **Python client libraries** for various services.
- You can use these libraries in your Python code to interact with GCP programmatically.
- For example, you can create virtual machines, manage storage buckets, and access APIs directly from your Python scripts.



IMPLEMENTATION

1. **Basic Firestore Creation and Operations**
2. **Firestore Integration with Other GCP Services**

IMPLEMENTATION :1

- **Basic Firestore Creation and Operations**

Task 1: Create a Firestore Database

There are two modes in firestore while creating database. They are:

- Native mode
- Datastore mode

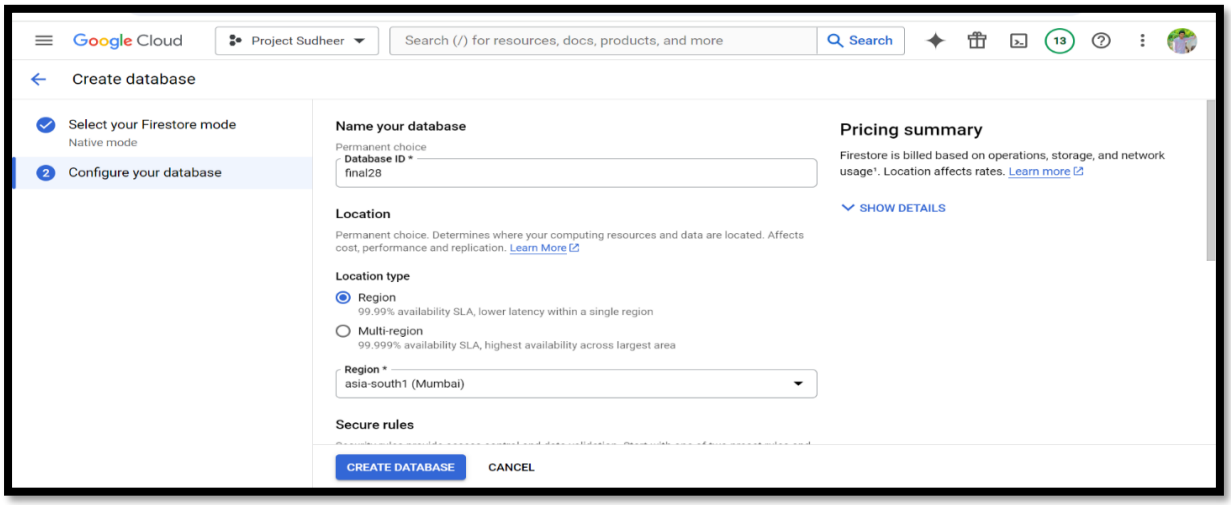
The main difference is:

1. **Firestore in Native mode**:
    - Next major version of Datastore.
    - Combines Datastore and Firebase Realtime Database features.
    - Offers real-time updates, a new data model, and mobile/web client libraries.
2. **Firestore in Datastore mode**:
    - Uses Datastore system behaviour but accesses Firestore's storage layer.
    - Removes Datastore limitations (e.g., eventual consistency, transaction limits).
    - Disables Firestore features not compatible with Datastore.
3. **Use Cases**:
    - **Native mode**: Ideal for real-time apps, scalability, and strong consistency.

- **Datastore mode**: Suitable for flexible data modelling and hierarchical data.

GUI Interactive way:

- Select Native mode and create database

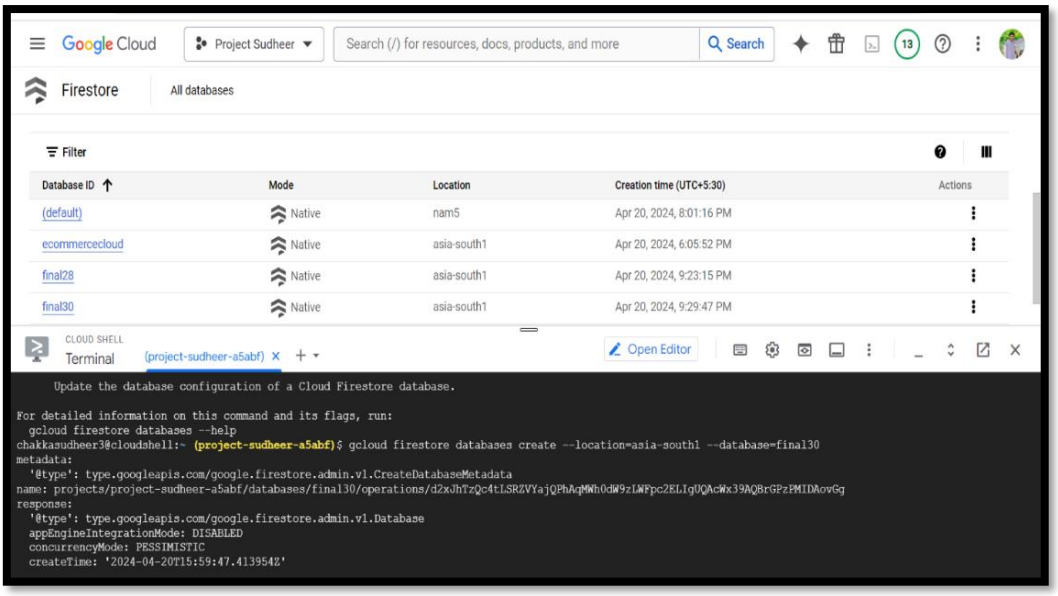


CLI way:

- In command line interface way, it is limited to do certain database operations in firestore.

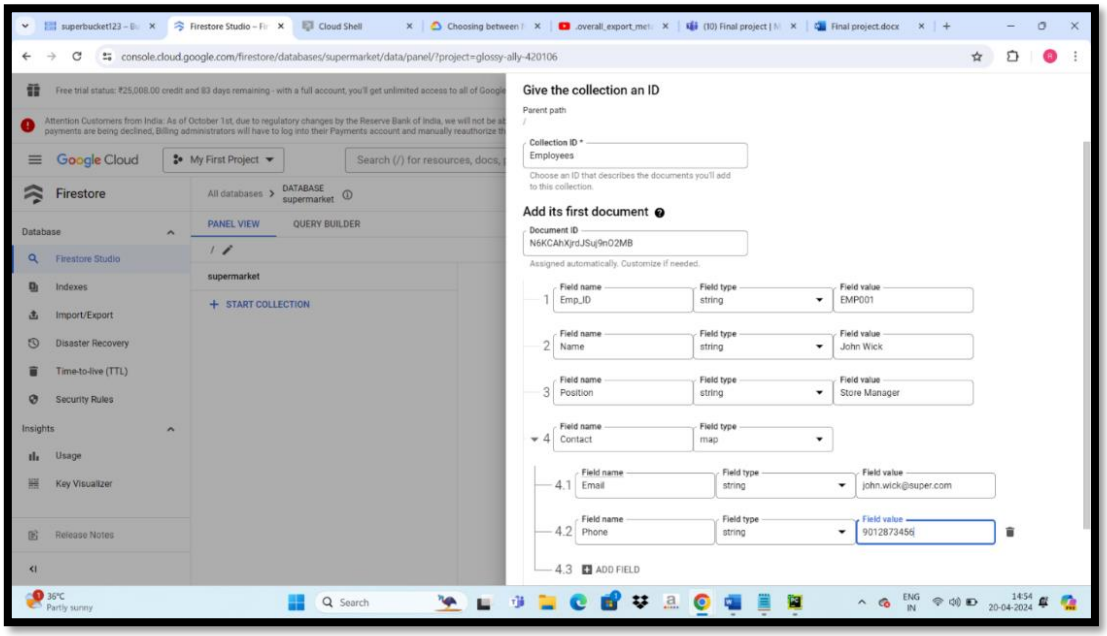
Command to create database in firestore is:

gcloud firestore databases create –database=DATABASE_ID –location=LOCATION –type=DATABASE_TYPE \
[--delete-protection]



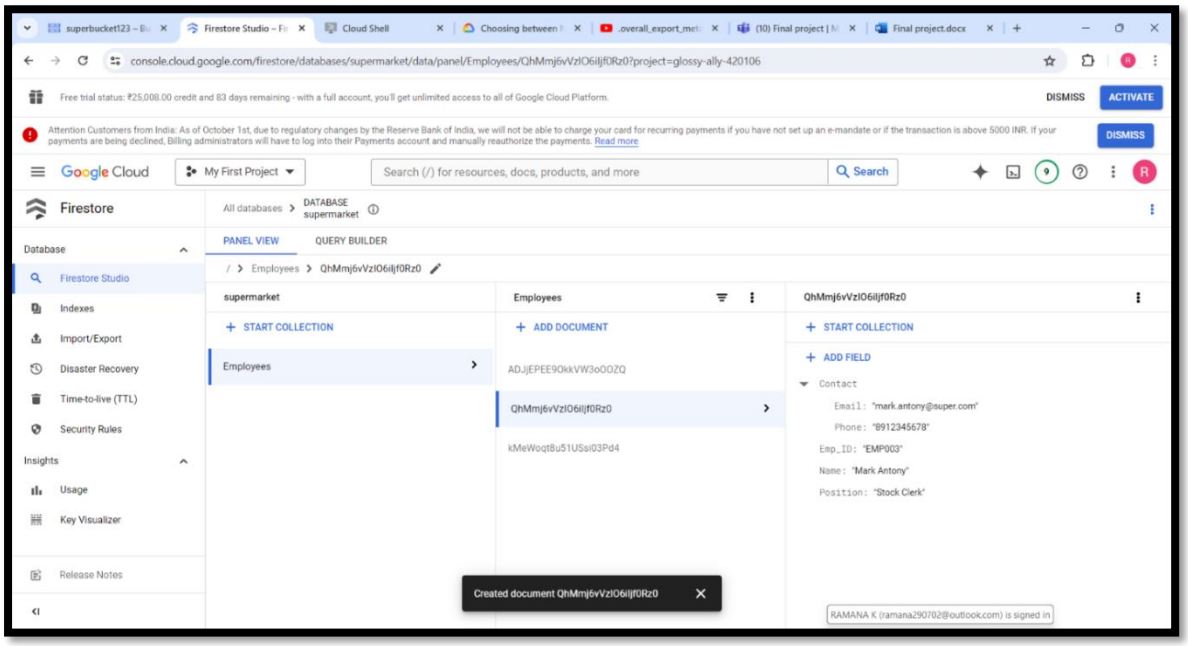
Task 2: Creation of collection

- Collection creation is possible only with GUI and Python way, but we can't create collections and documents using CLI.



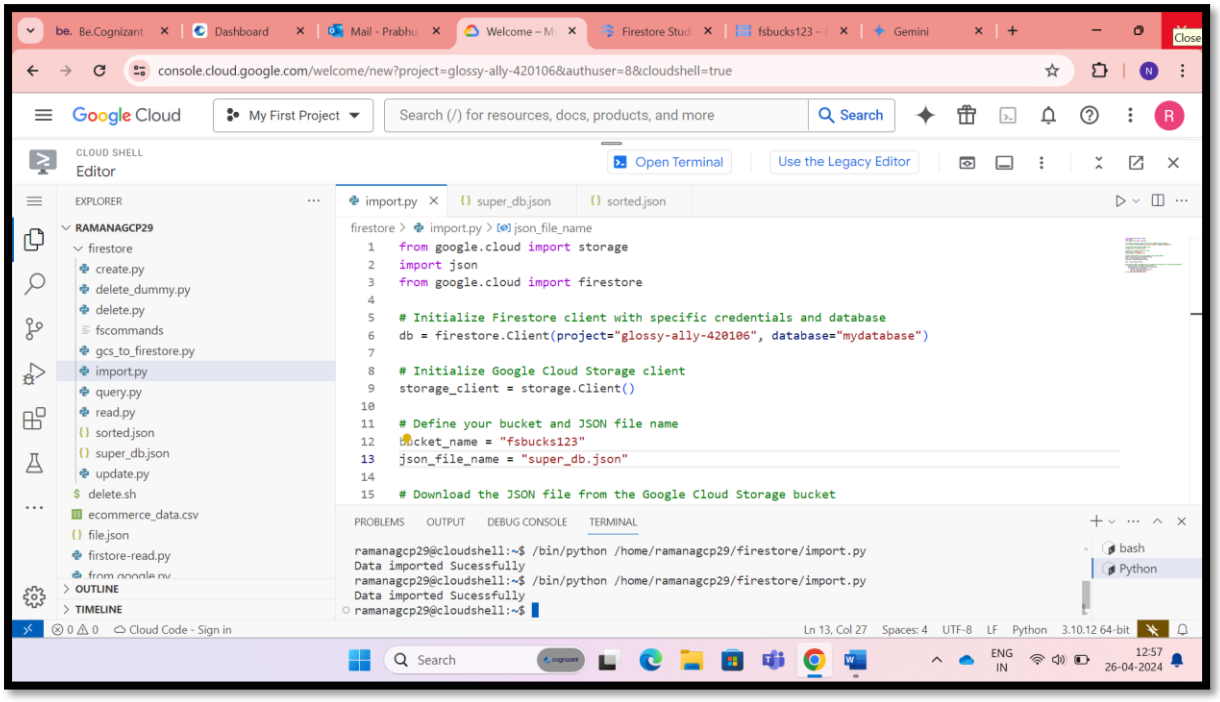
Task 3: Add few documents in the collection.

- Add few documents to existing collection and documents has its own unique id.

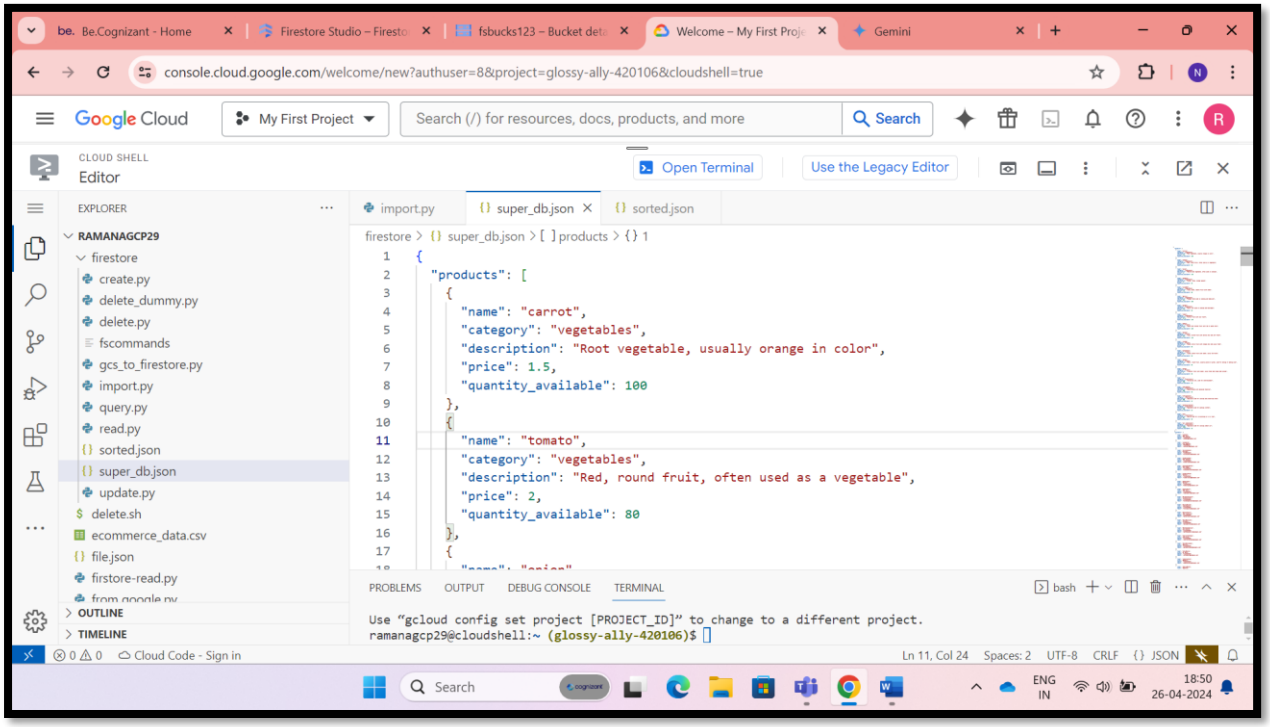


Task 4: Create python script which will read data from a file to firestore.

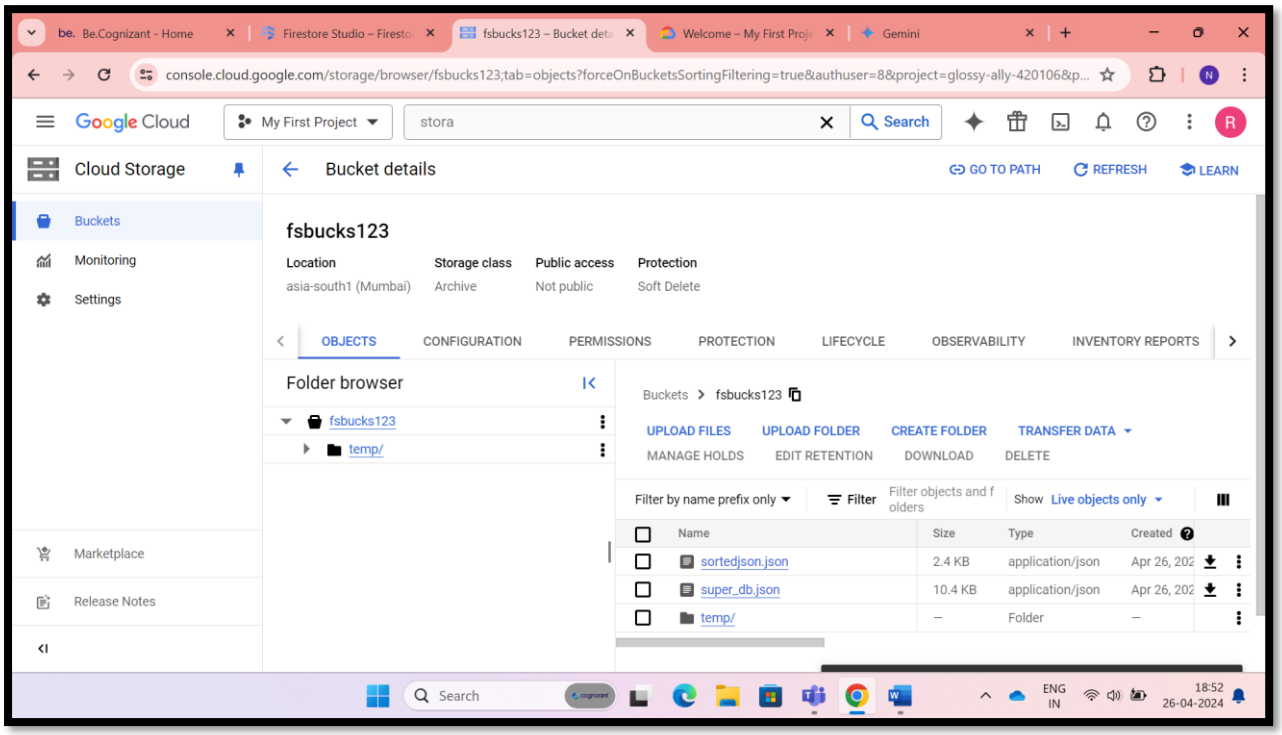
- Python script is executed on the editor provided by GCP with requires authorization.



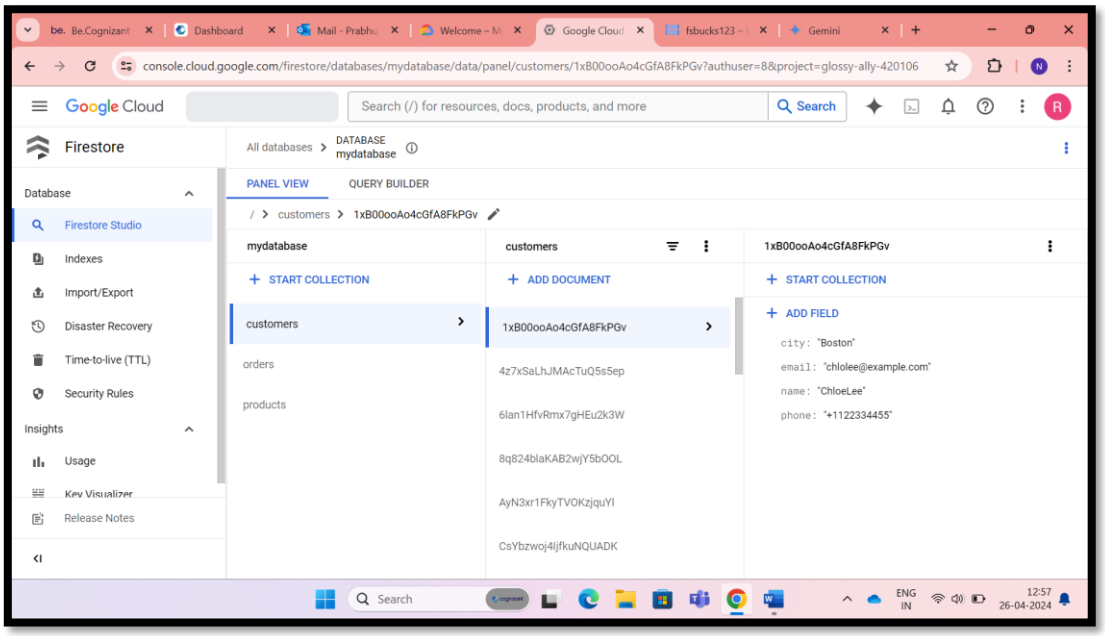
➢ This is the JSON file used to write data into FIRESTORE.



➢ I have used Google cloud storage as pipeline for writing data to FIRESTORE.

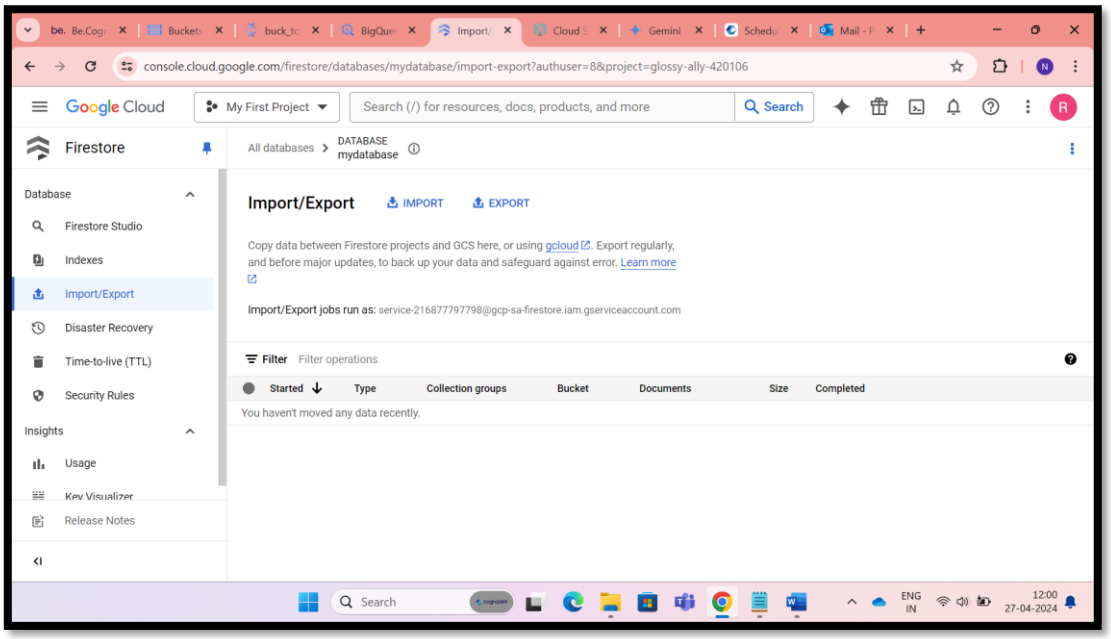


- After execution the Collections have been created with Documents with all data inside it.

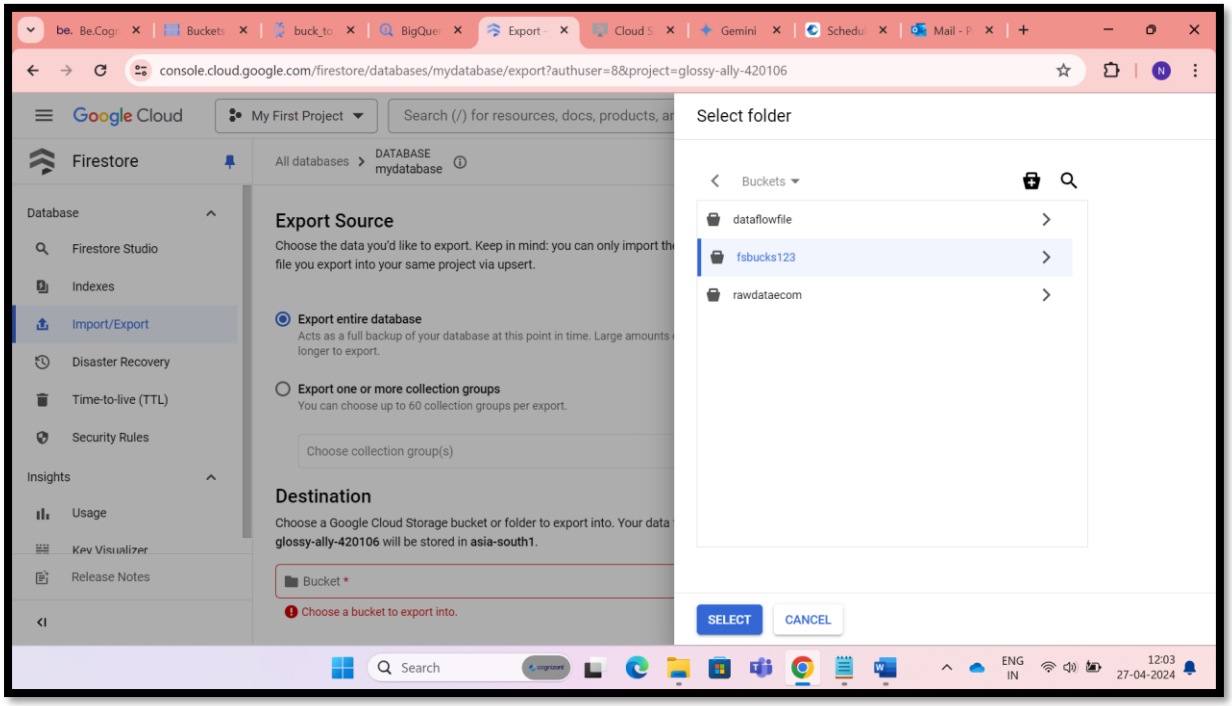


Additional Tasks:

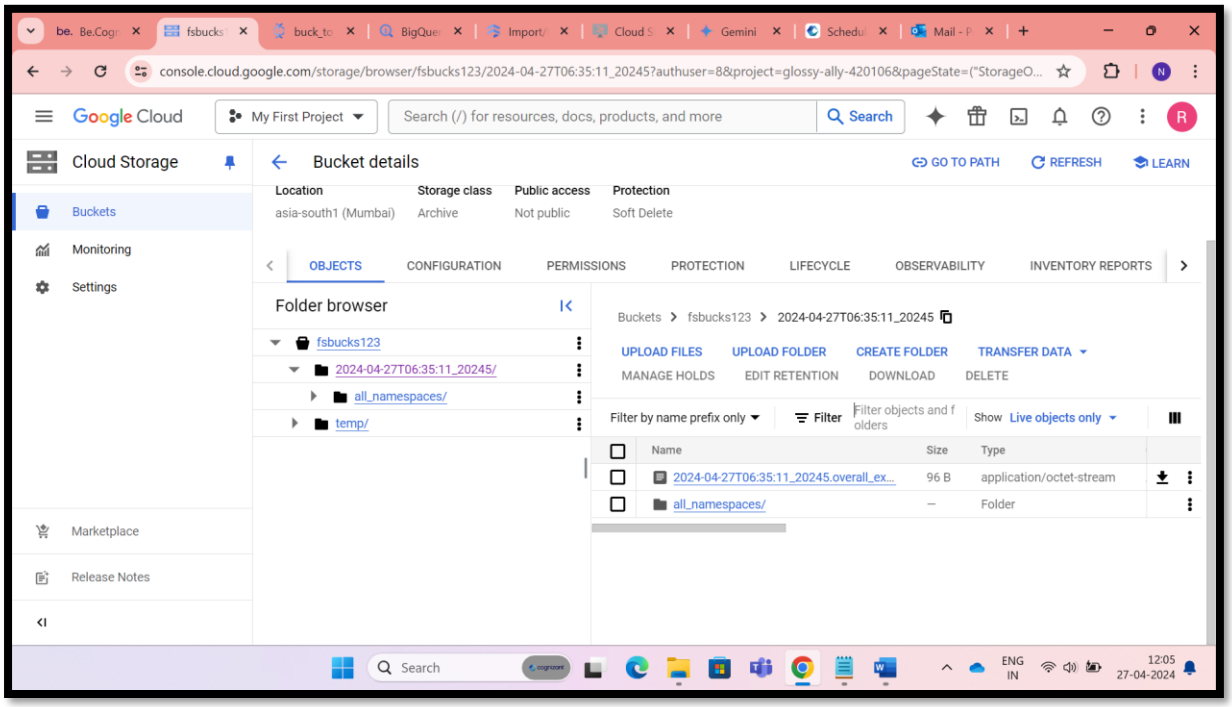
- **Import/Export of data from firestore.**



➢ Exporting data from firestore to Google cloud storage.

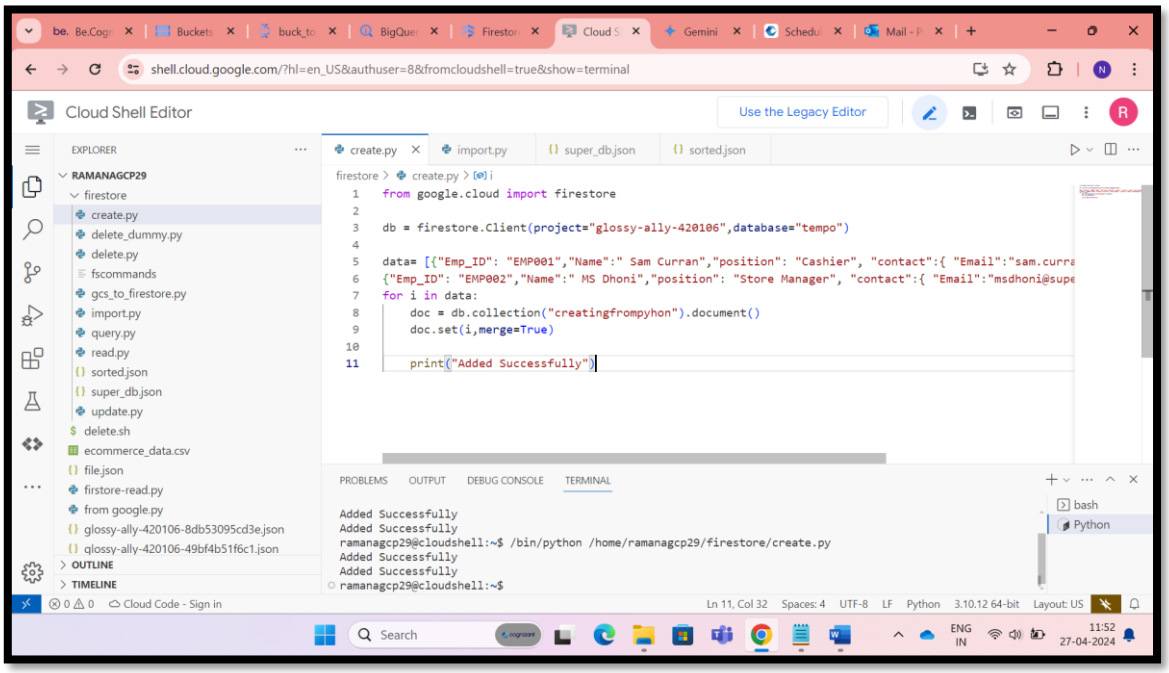


➢ Successfully data is exported.

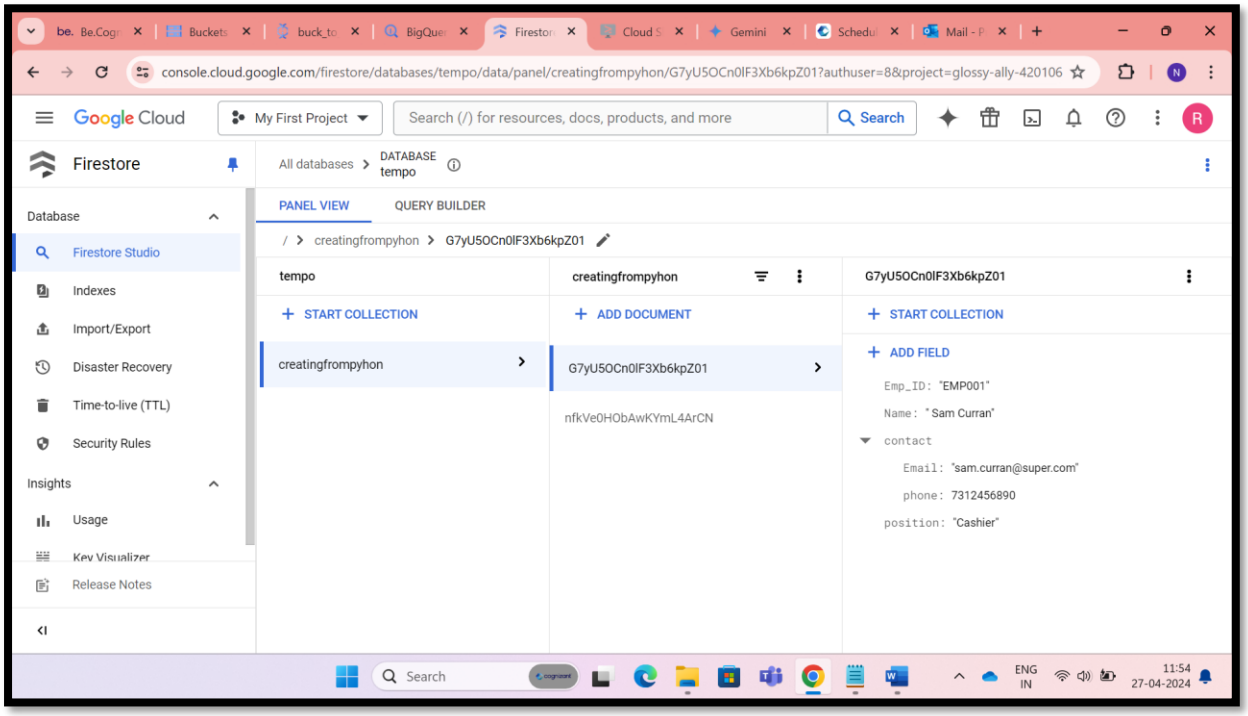


➢ Some CRUD operations using python.

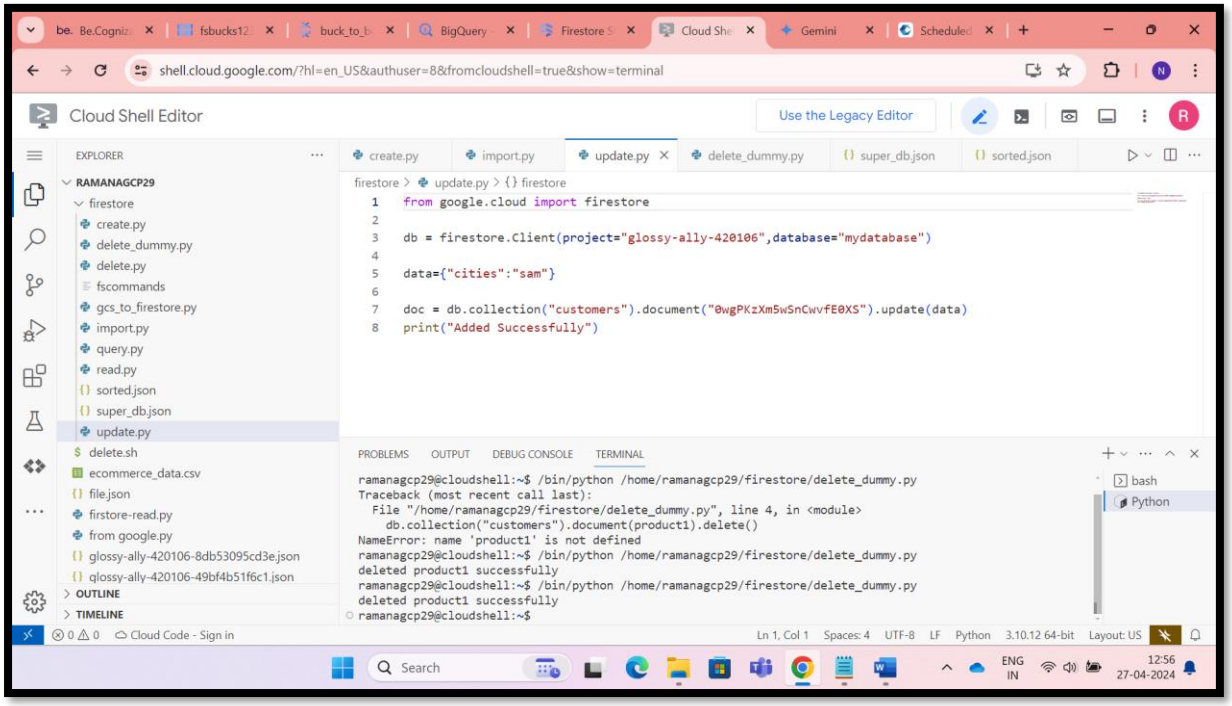
➢ Creating collection and adding documents using python.



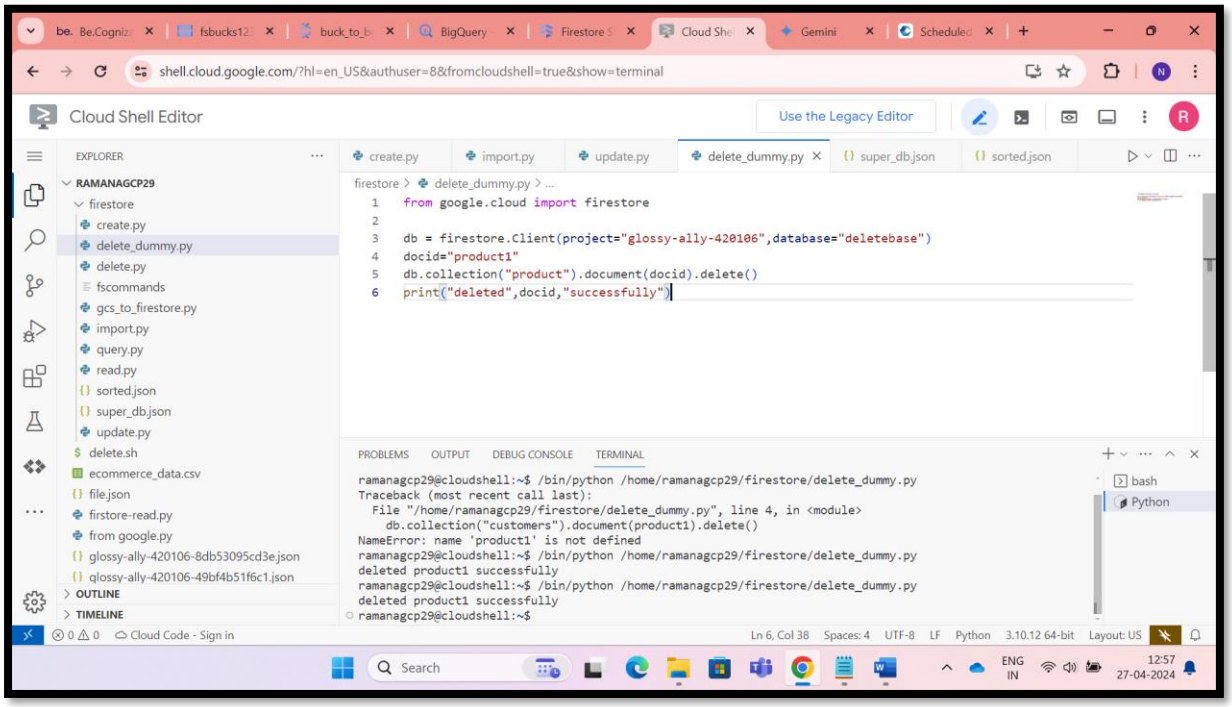
➢ The data is added successfully.



➢ Updating a field in a document.

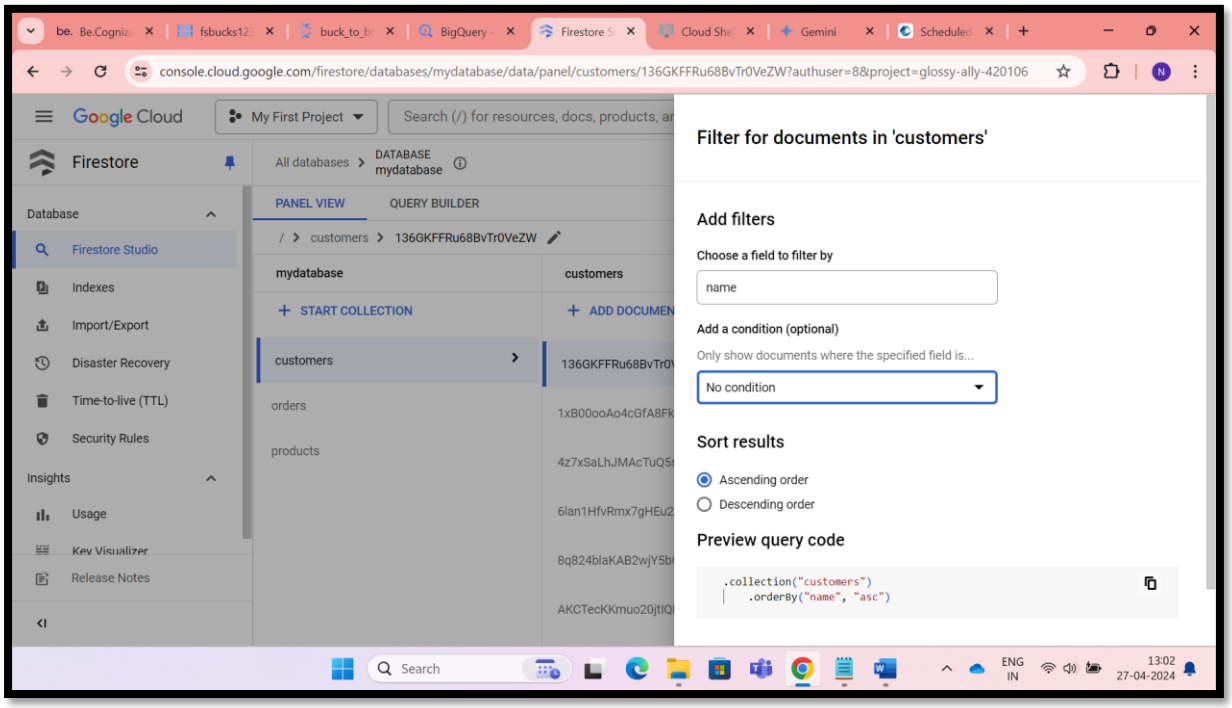


➢ Deleting a document inside a collection.

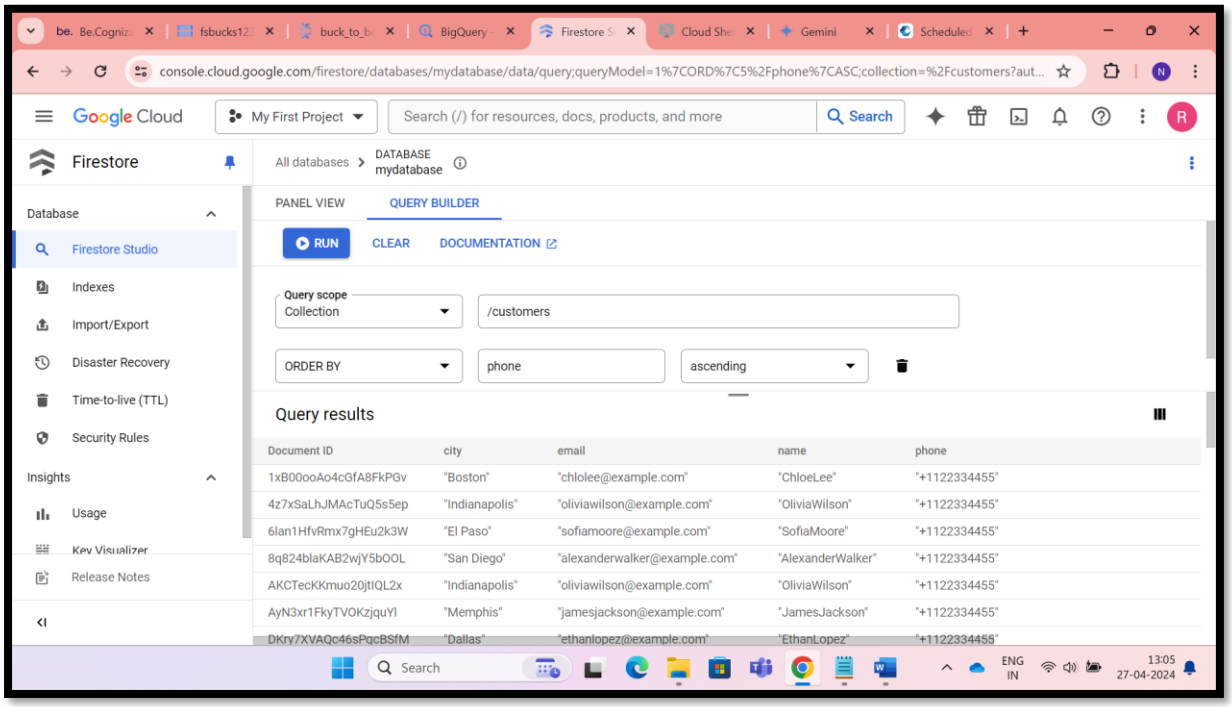


Filtering and querying in Firestore

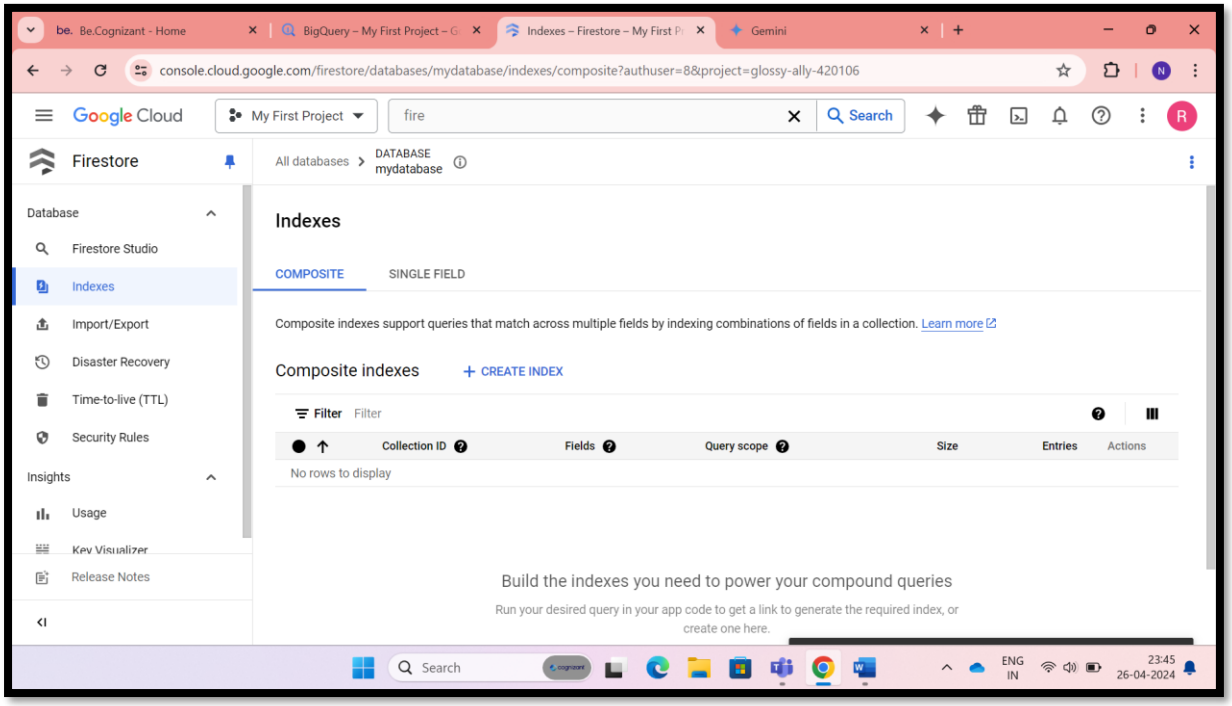
- Filtering



- Querying



➢ Indexing

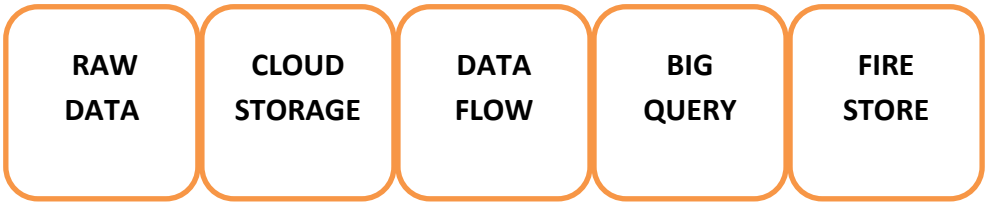

Be.Cognizant - Home
BigQuery – My First Project – G
Indexes – Firestore – My First Pr
Gemini
console.cloud.google.com/firestore/databases/mydatabase/indexes/composite?authuser=8&project=glossy-ally-420106
Google Cloud
My First Project
fire
Search
Firestore
All databases
DATABASE
mydatabase
Database
Firestore Studio
Indexes
Import/Export
Disaster Recovery
Time-to-live (TTL)
Security Rules
Insights
Usage
Key Visualizer
Release Notes
Indexes
COMPOSITE
SINGLE FIELD
Composite indexes support queries that match across multiple fields by indexing combinations of fields in a collection. Learn more
Composite indexes
+ CREATE INDEX
Filter
Filter
Collection ID
Fields
Query scope
Size
Entries
Actions
No rows to display
Build the indexes you need to power your compound queries
Run your desired query in your app code to get a link to generate the required index, or create one here.
Search
ENG
IN
23:45
26-04-2024

IMPLEMENTATION :2

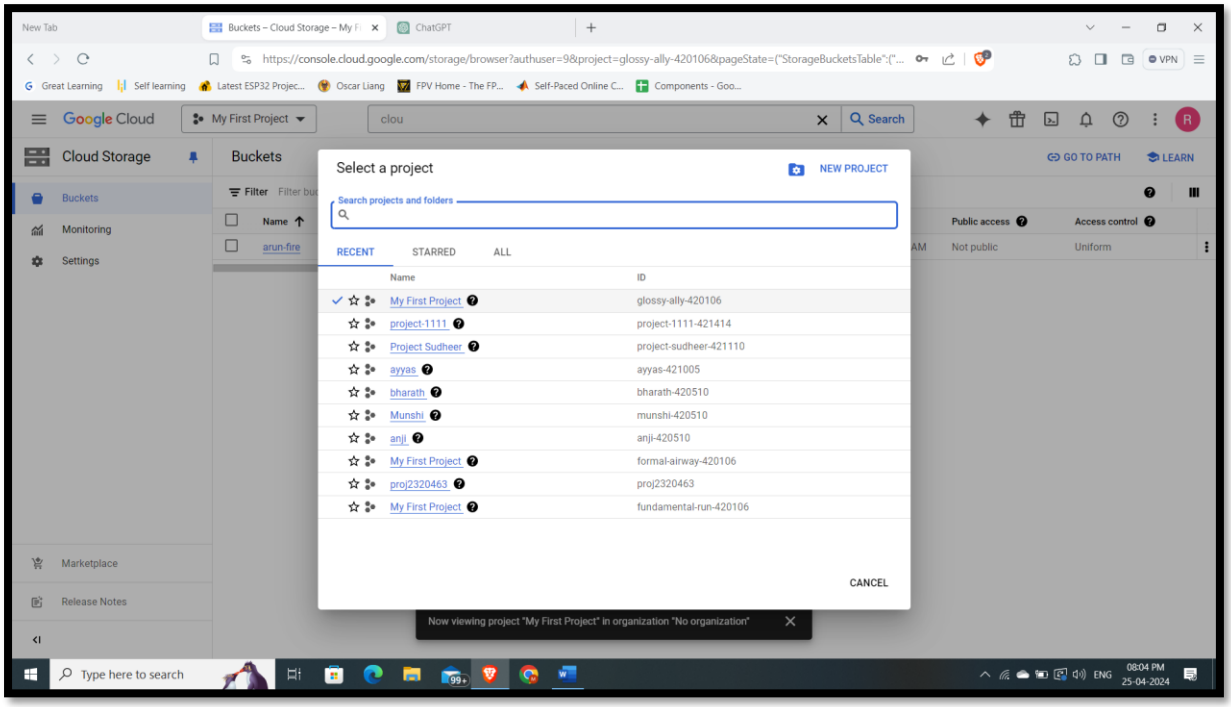
- **Firestore Integration with Other GCP Services**

Here in the implementation part 2, I have integrated some of the GCP services and used them to do some kind of data analysis.

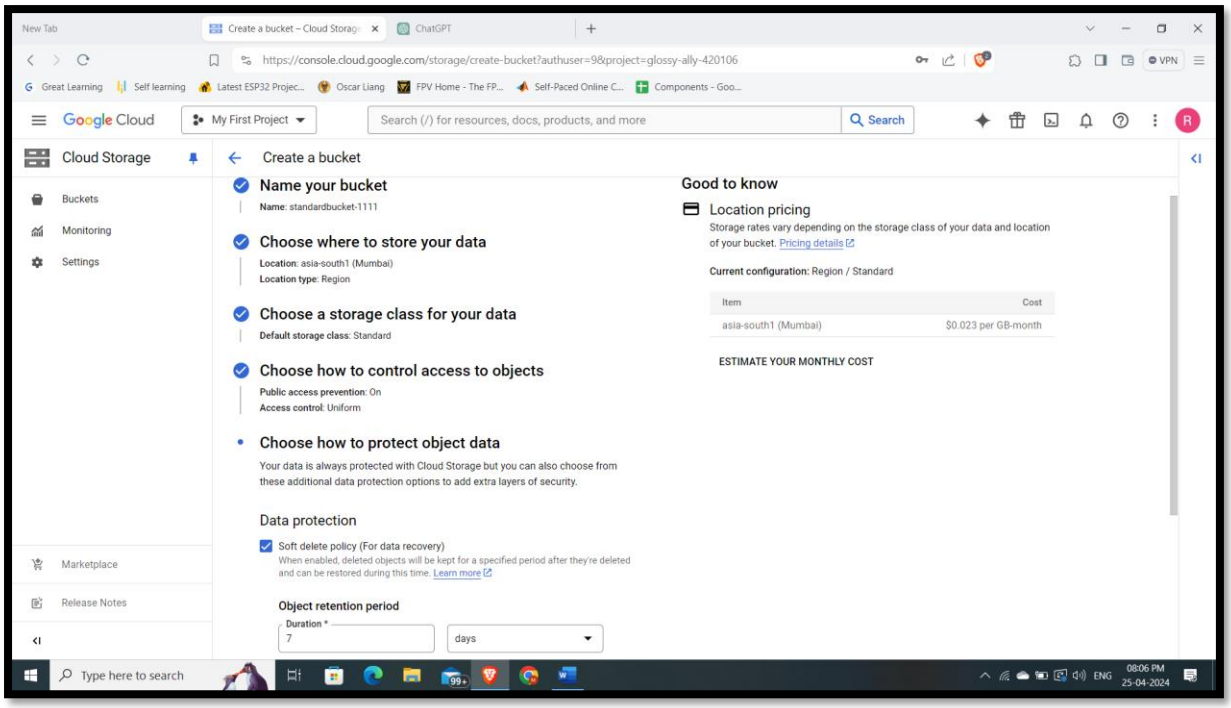
I have used the ecommerce data for our data analysis.



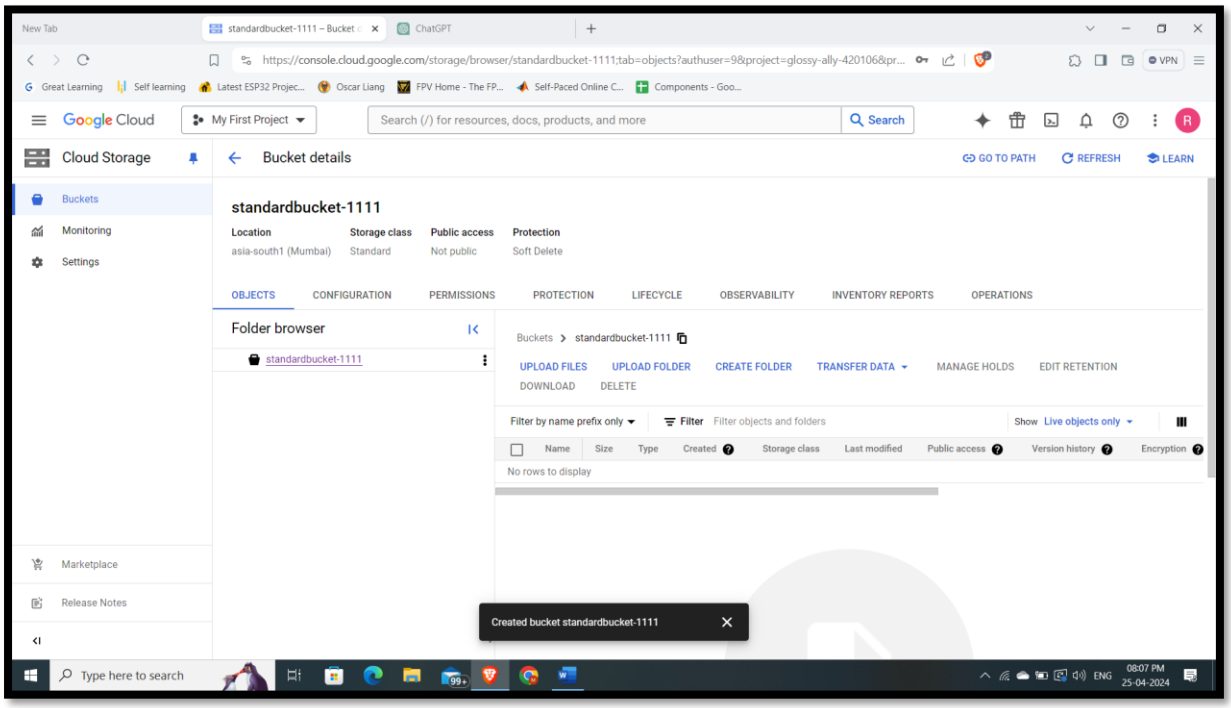
- Create a Project.



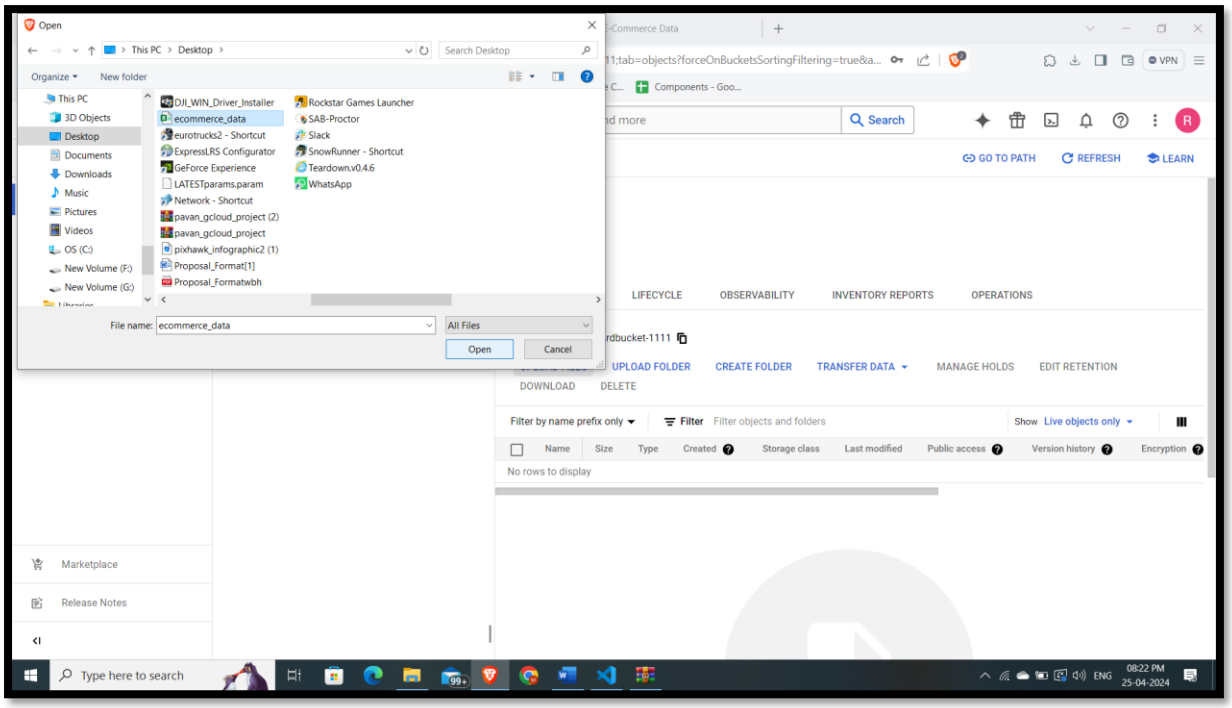
- Create Google cloud storage bucket.



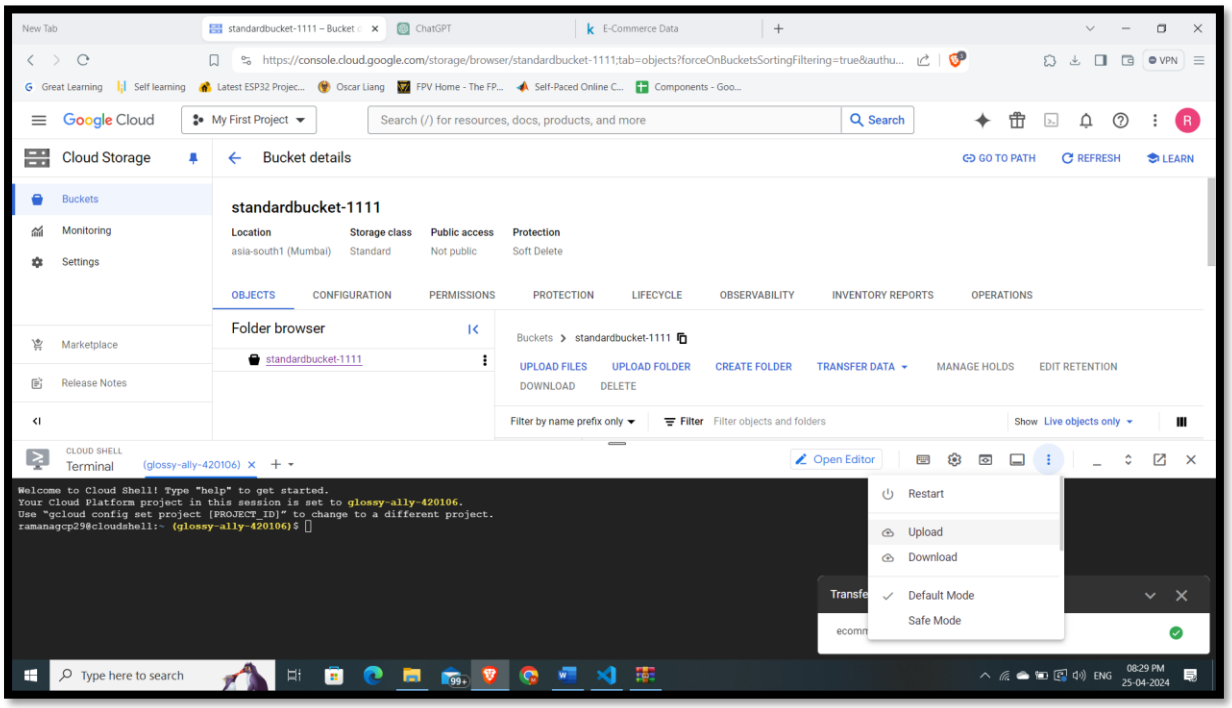
- I have created a standard bucket where we can upload our raw data.



- Upload the raw data file from local machine to bucket.



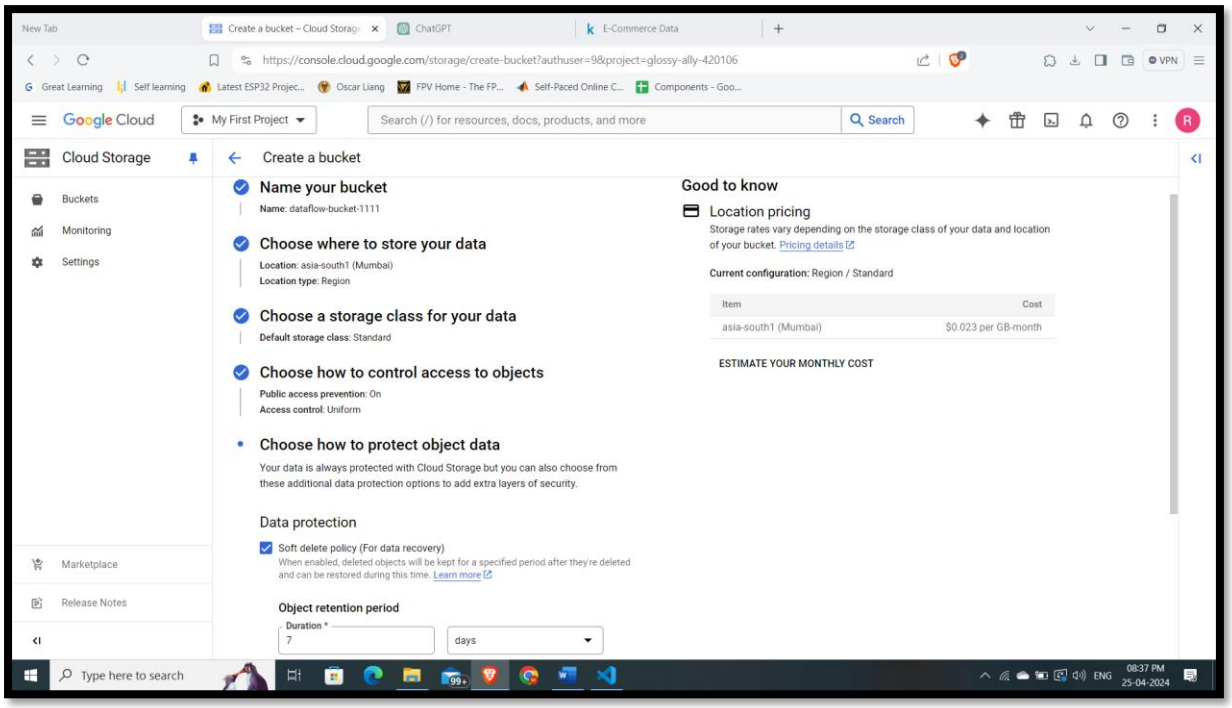
- We can also upload file using GSUTIL commands(optional).



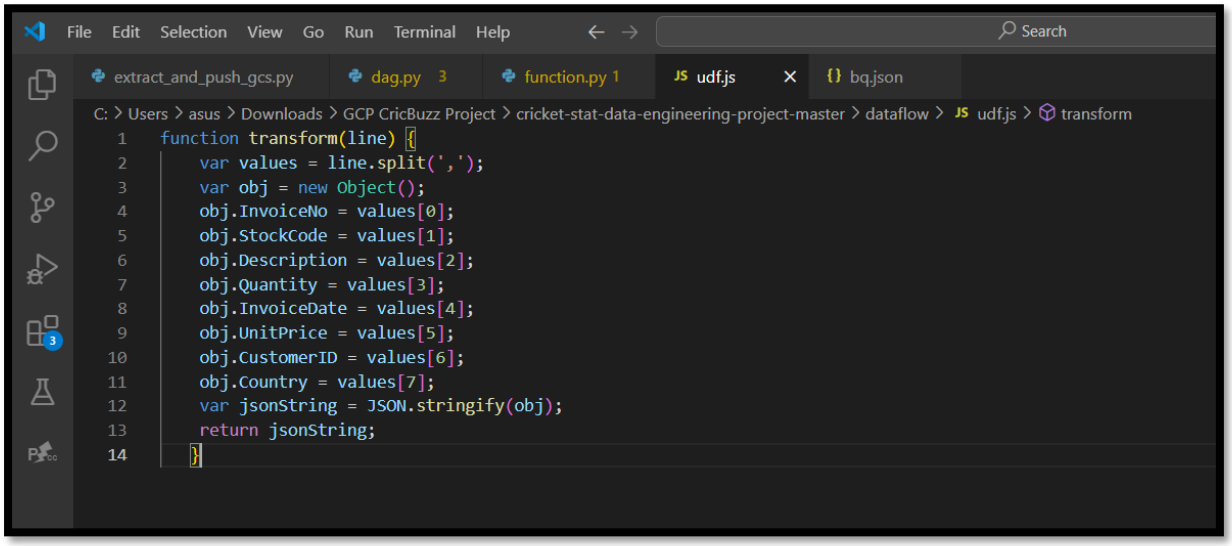
Command to copy file from CLI to GCS bucket.

gsutil cp /home/ramanagcp29/ecommerce_data.csv gs://standardbucket-1111/

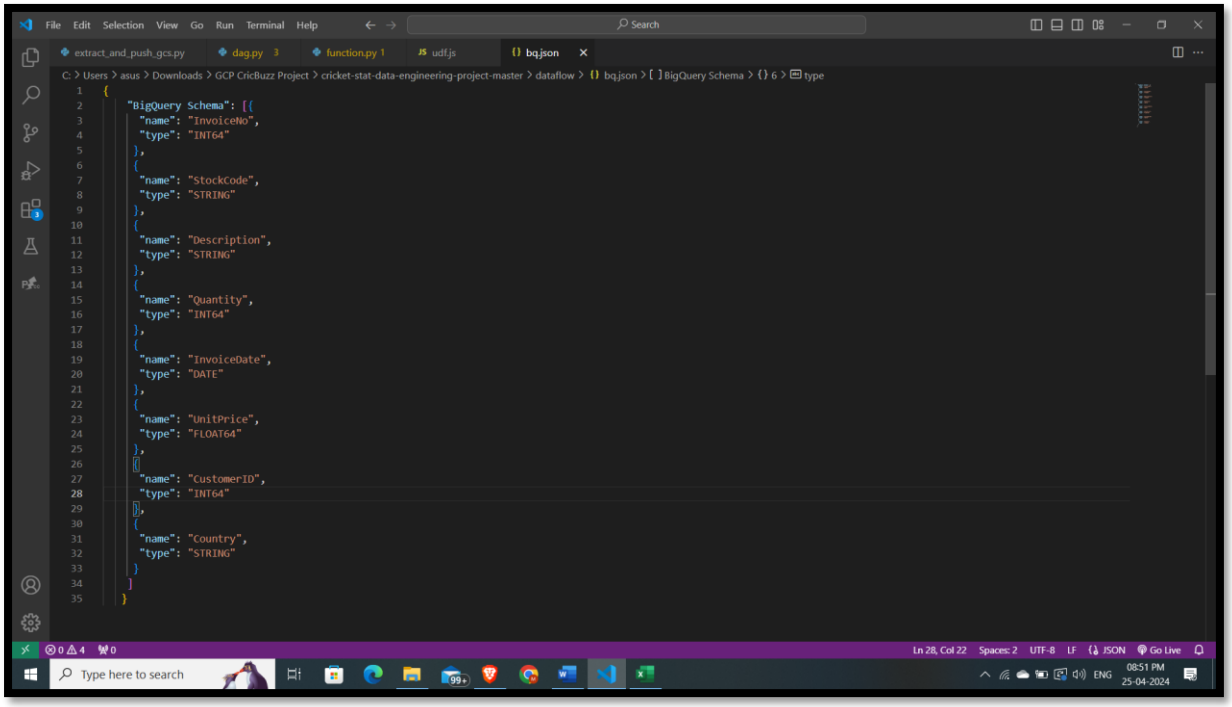
- Let's create another bucket where we will store files related to DATAFLOW.



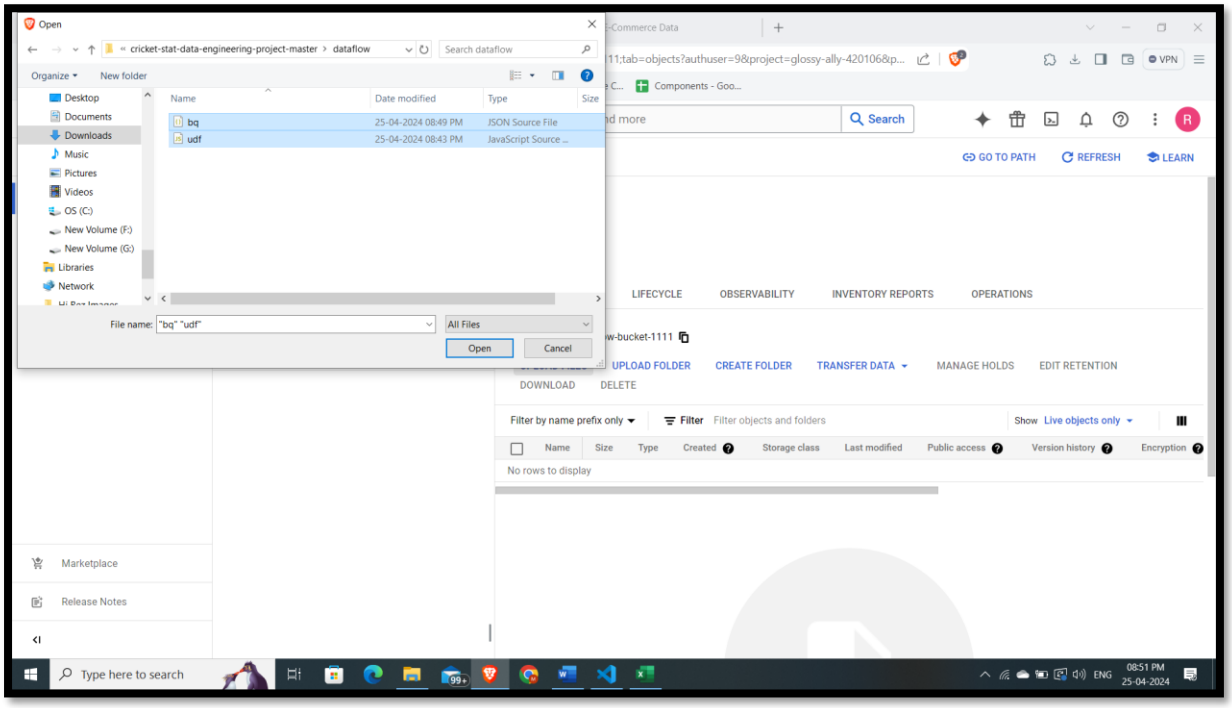
- Creating a UDF function which defines the fields in our raw data which we will be using for analytical purpose.



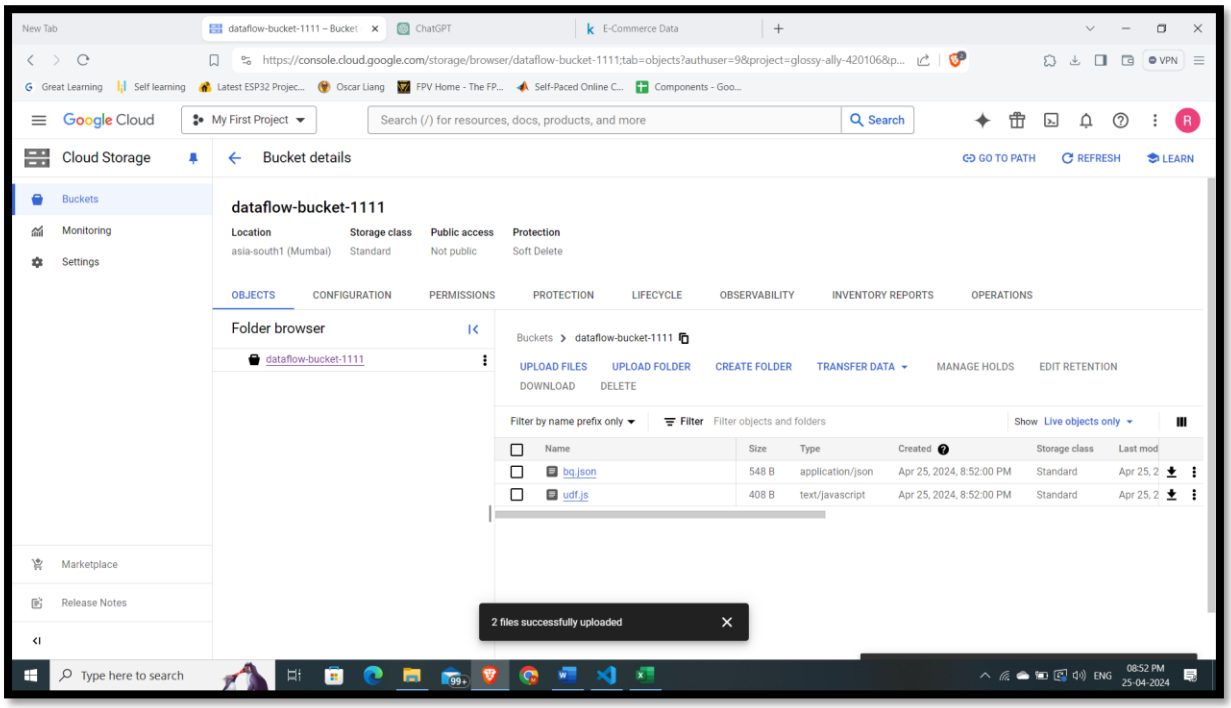
- Defining the datatypes of the raw data, which will be the schema for the BIGQUERY in JSON file format.



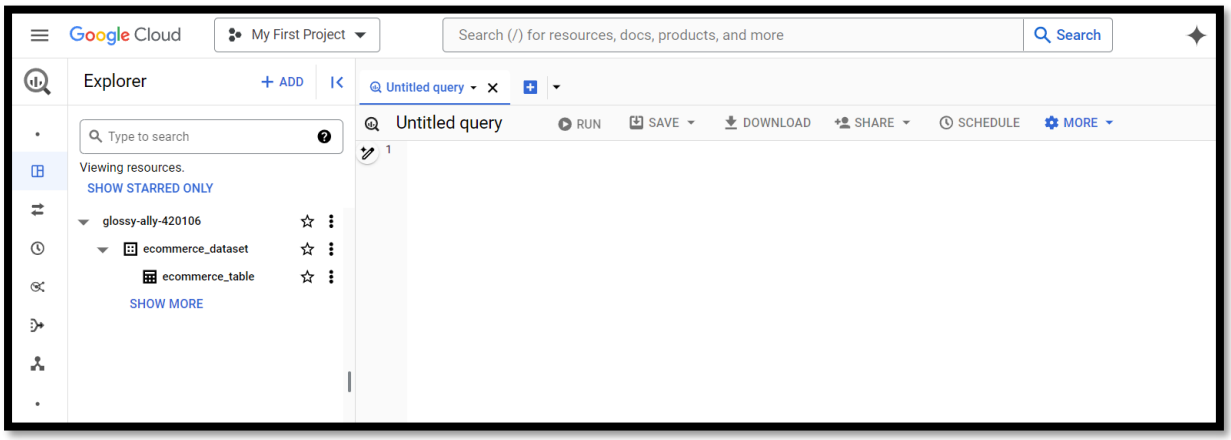
- Uploading data to the dataflow bucket.



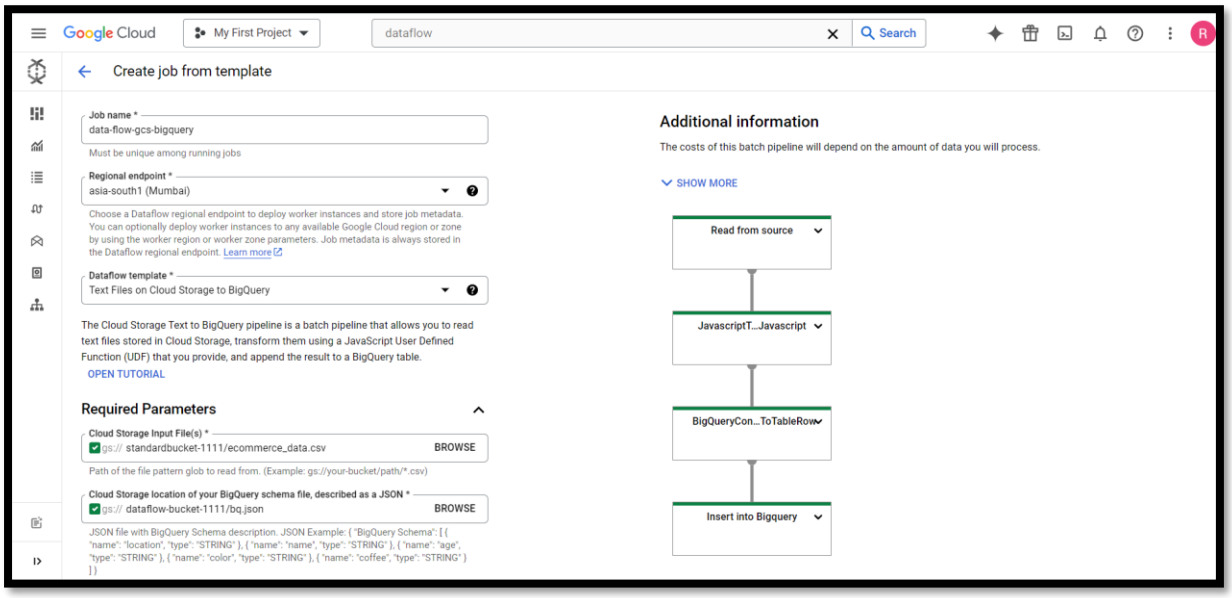
➢ Uploading required files successfully.



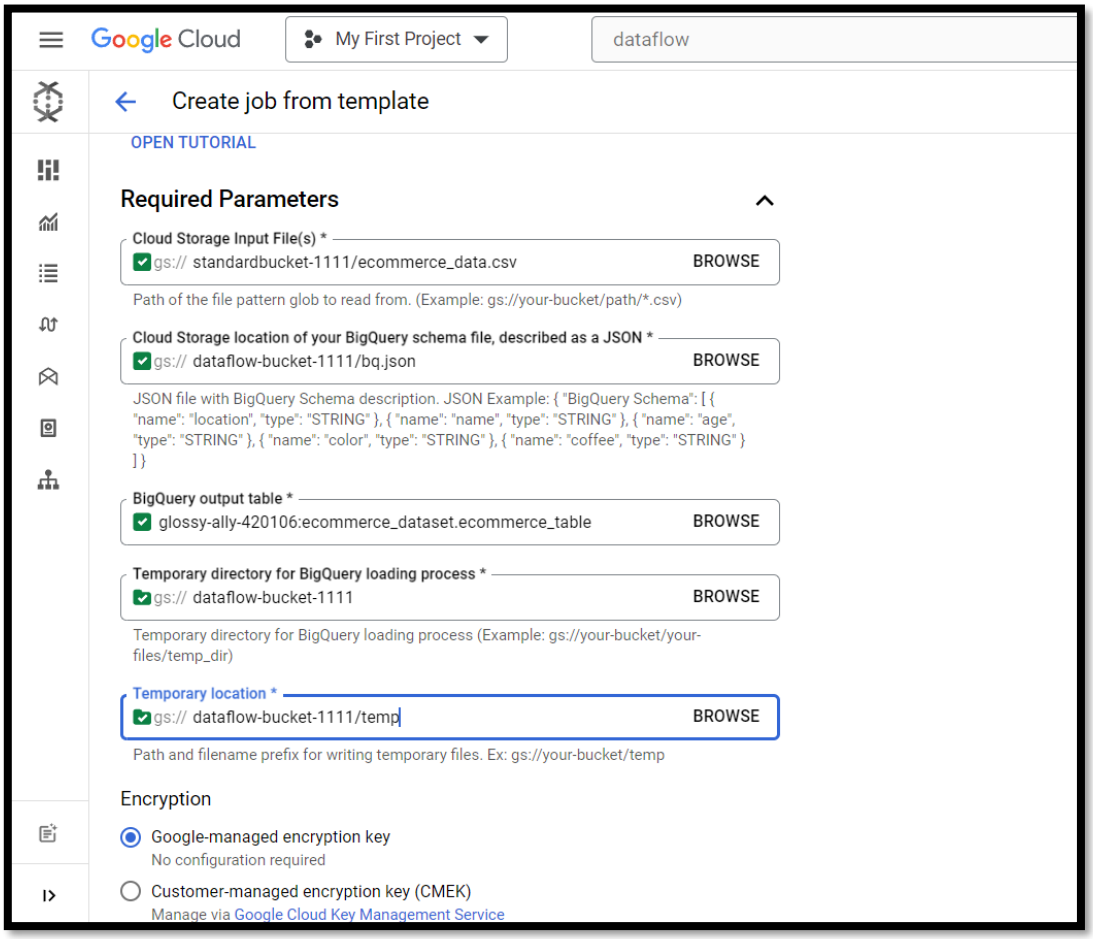
➢ Create a database in BIGQUERY and an empty table.



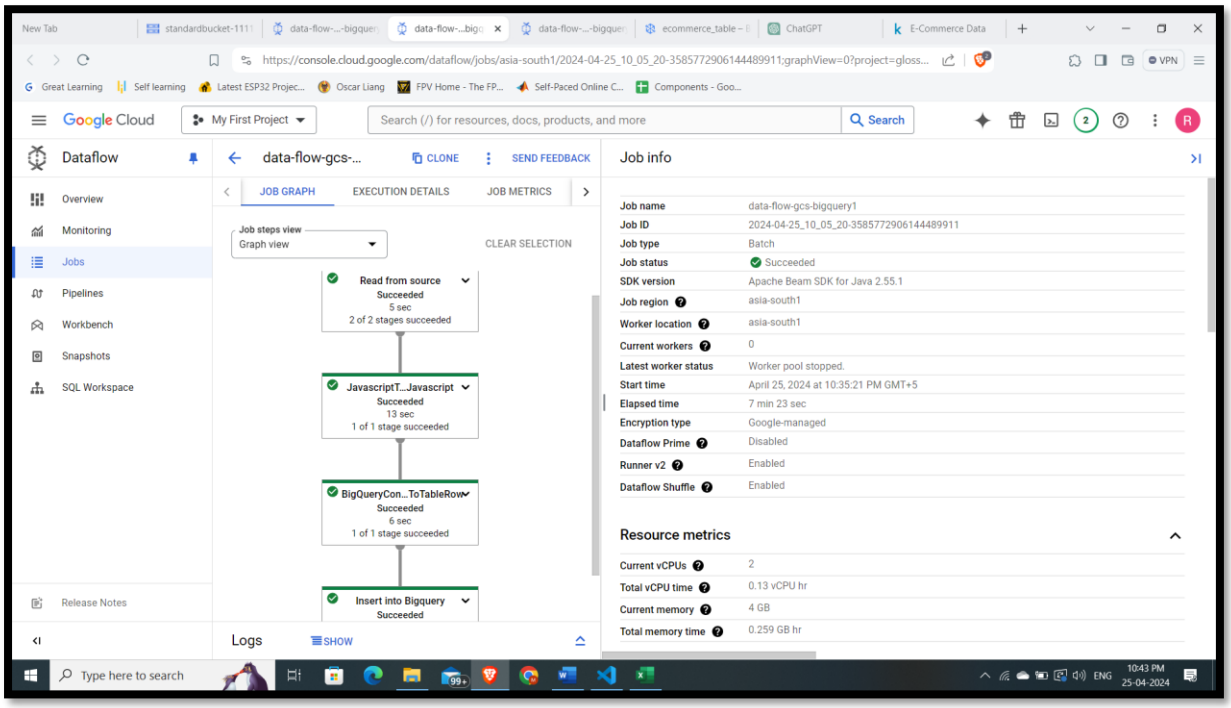
➢ Create job from template in DATAFLOW.



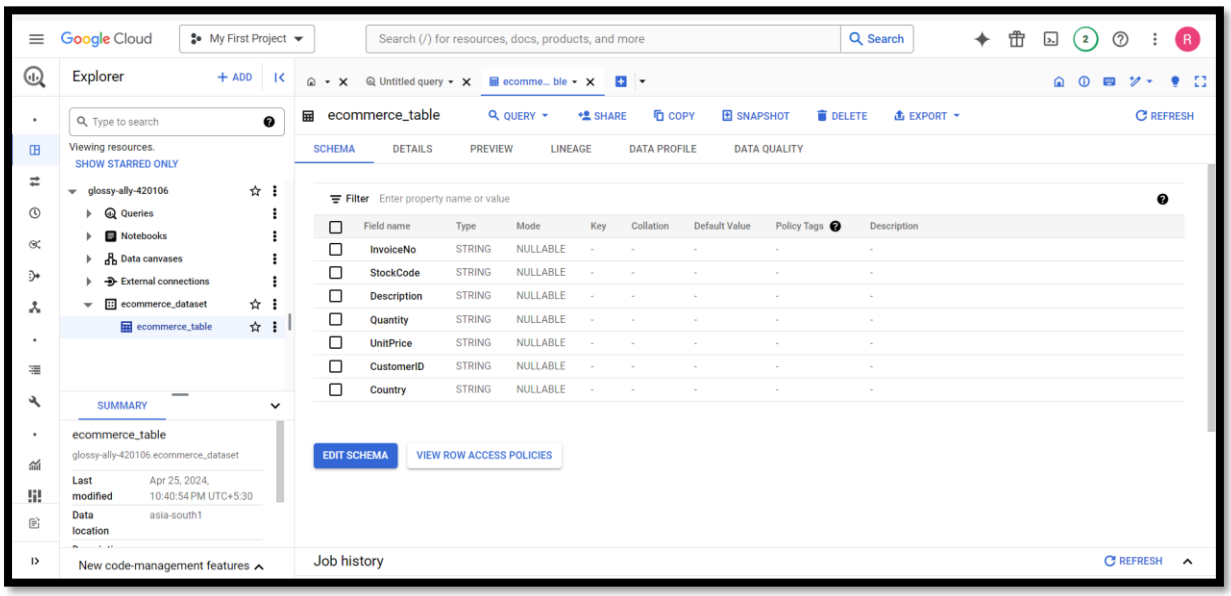
➢ Give all the required input output file and folder paths.



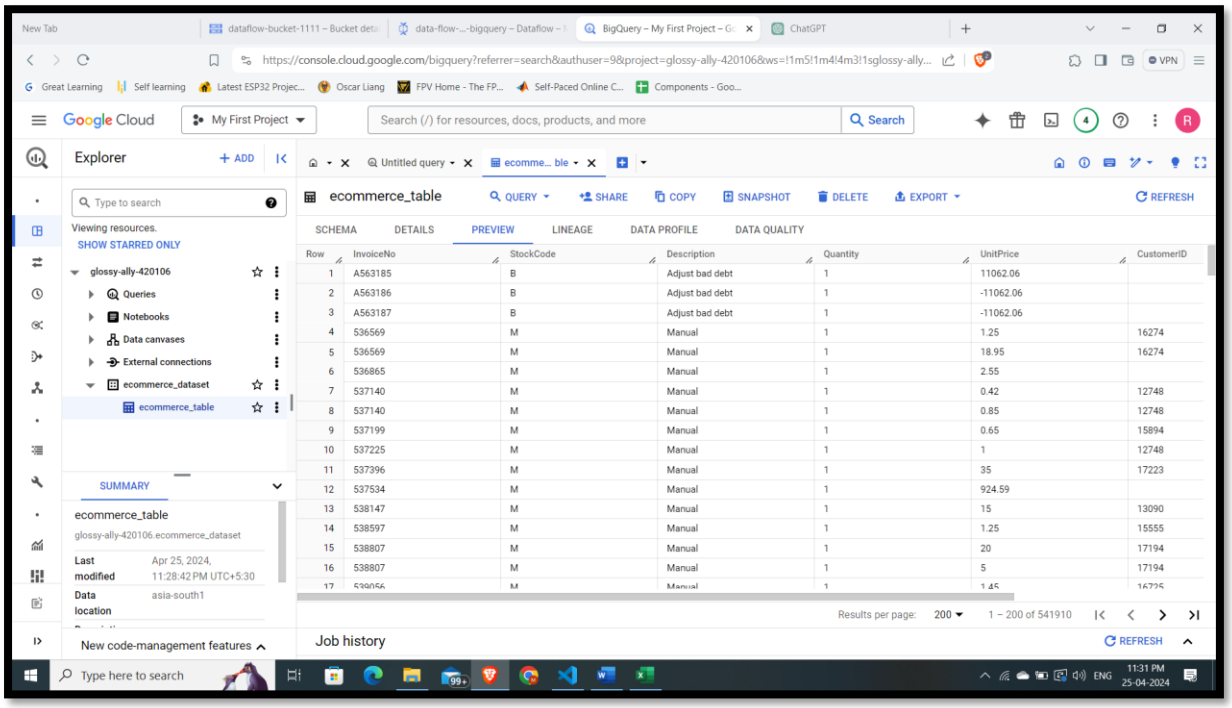
➢ Run the job and check for successful completion.



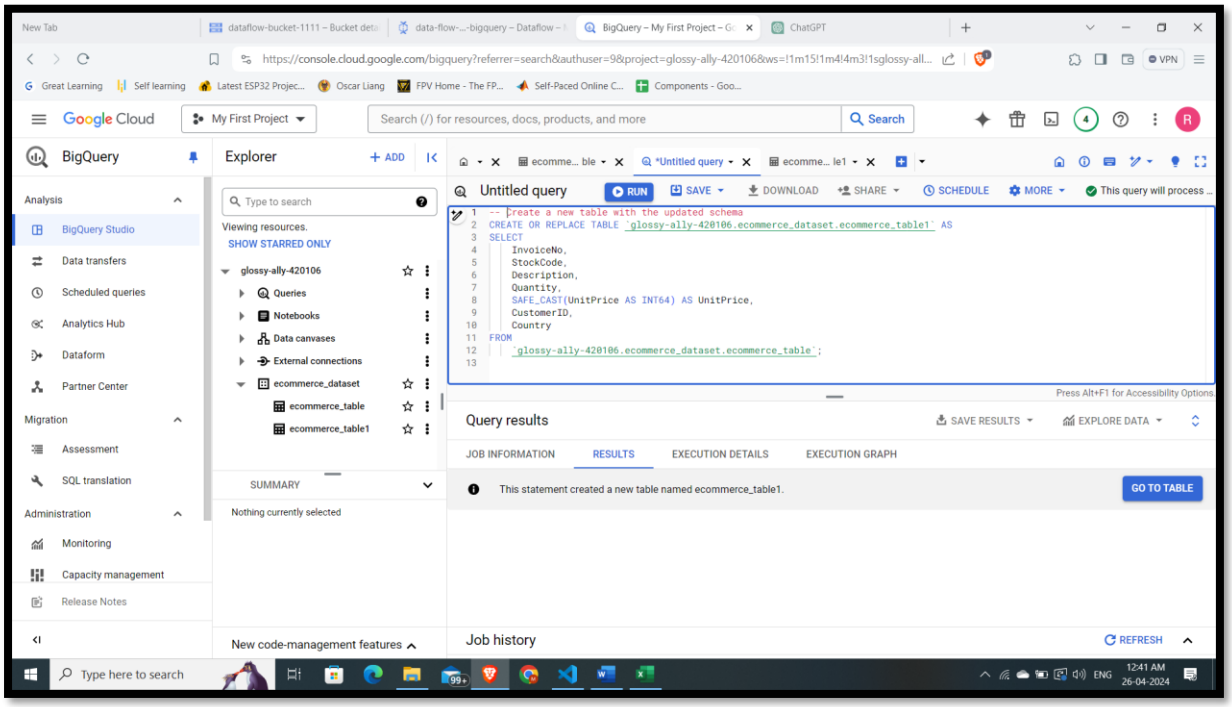
➢ The data is successfully loaded into the bigquery table with the desired schema.



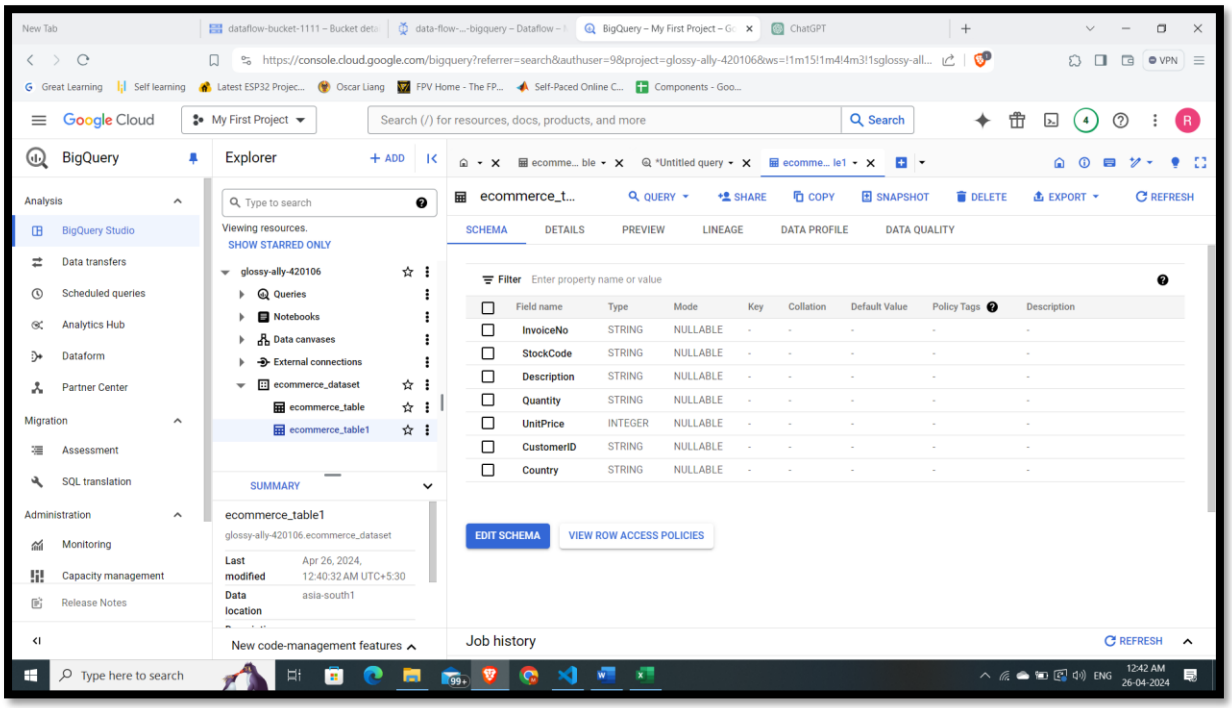
➢ This is the preview of the data in table with all the columns and rows.



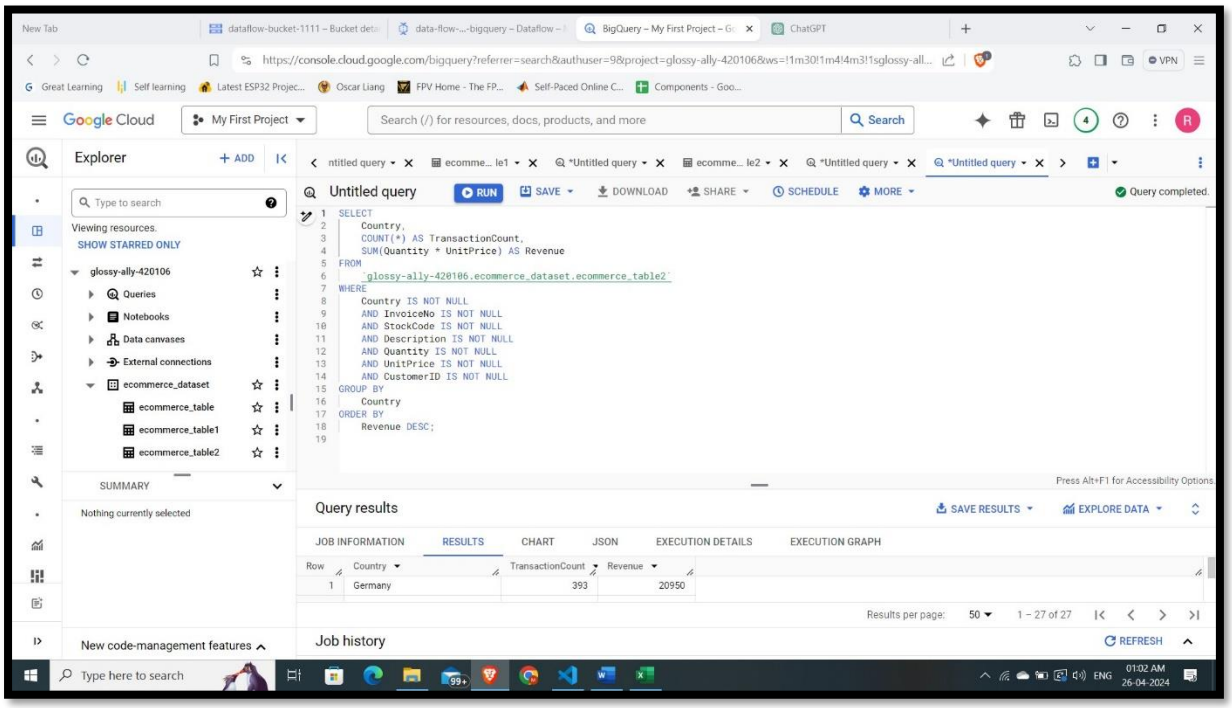
➢ Query to cast the datatypes, string to integer and created new table.



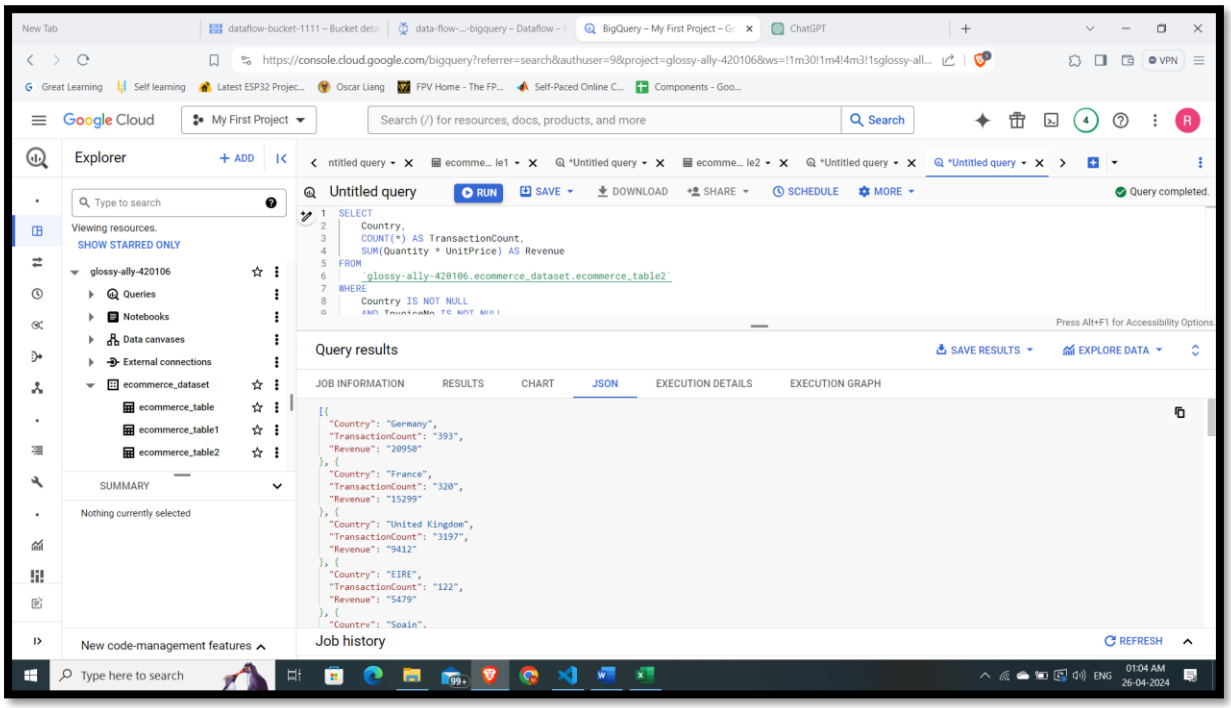
- Updated schema of UnitPrice field with integer datatype.



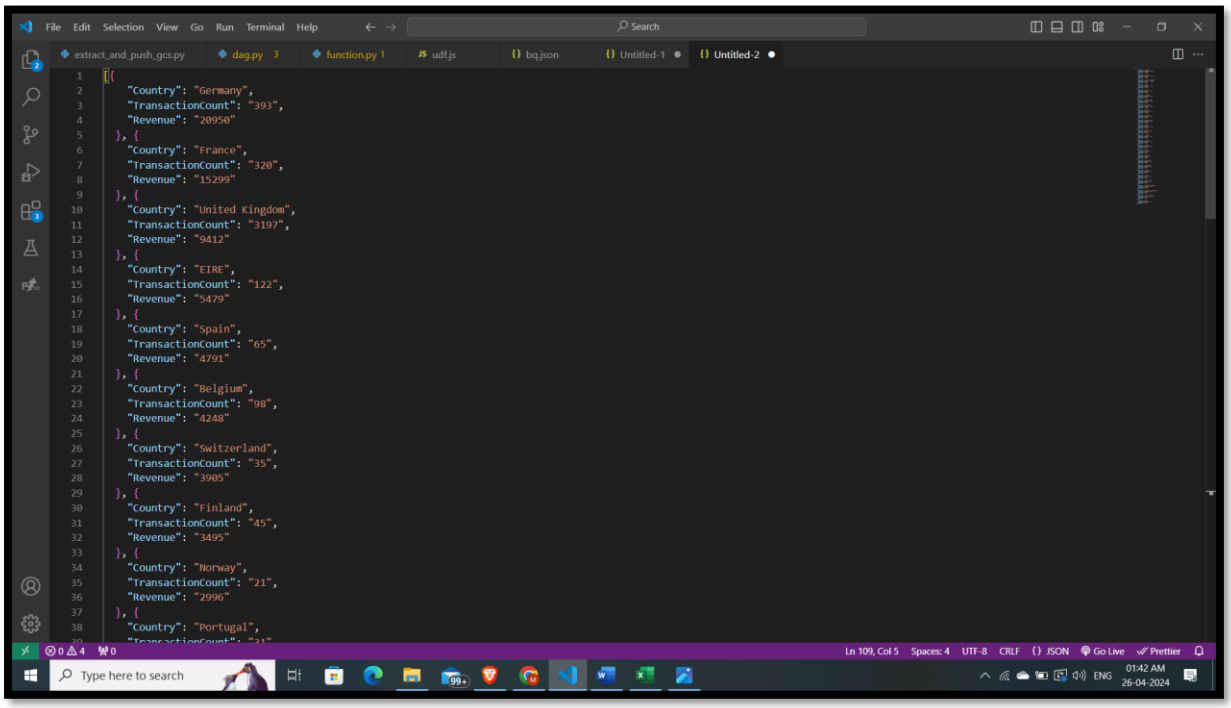
- Data cleansing by removing null values, and count, sum operation on the above table.



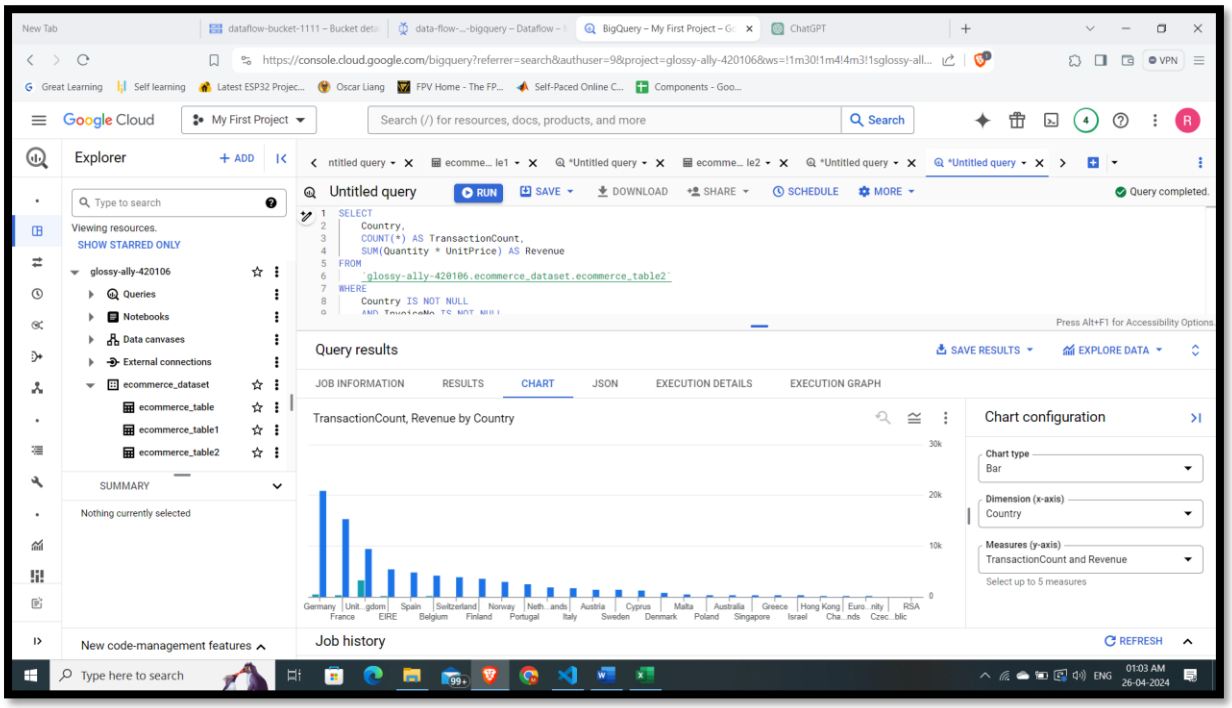
➢ The tables in bigquery also gives the data in JSON format.



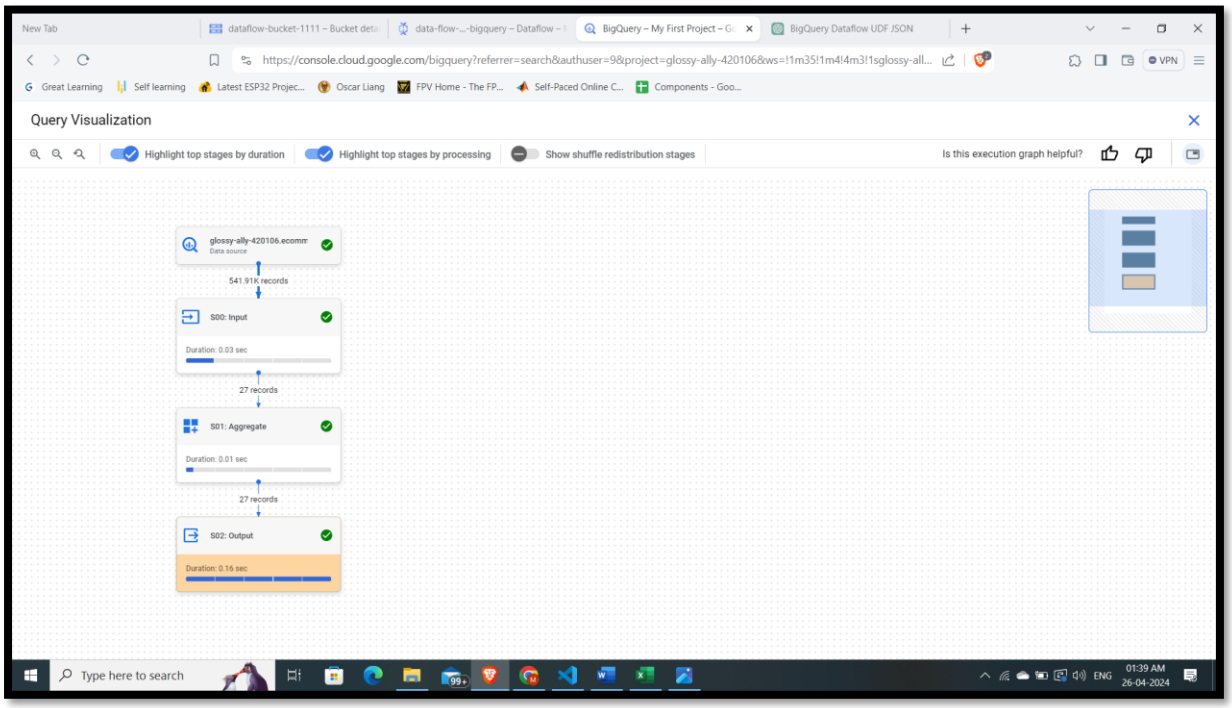
➢ JSON data copied to local.



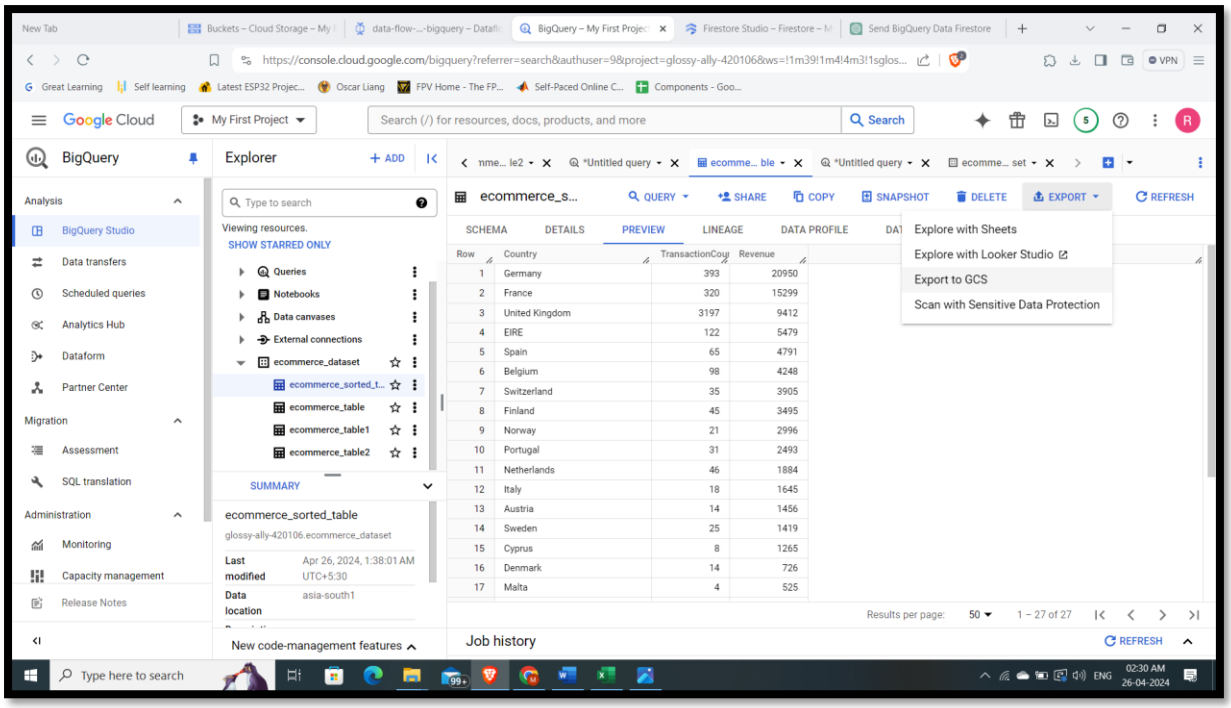
➢ The Data also can be viewed in chart or graphical format.



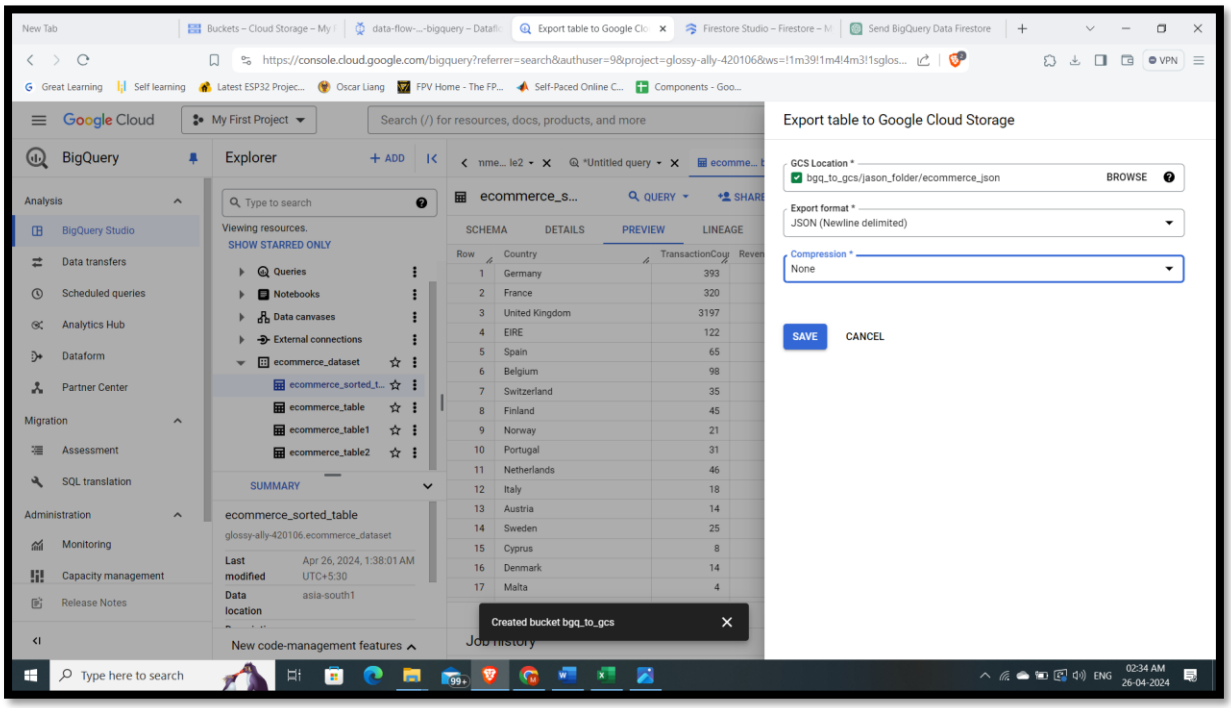
➢ Query Visualization.



➢ Exporting table back to Google cloud storage.



➢ Choosing export format and location.



- New table data from Bigquery is saved here.