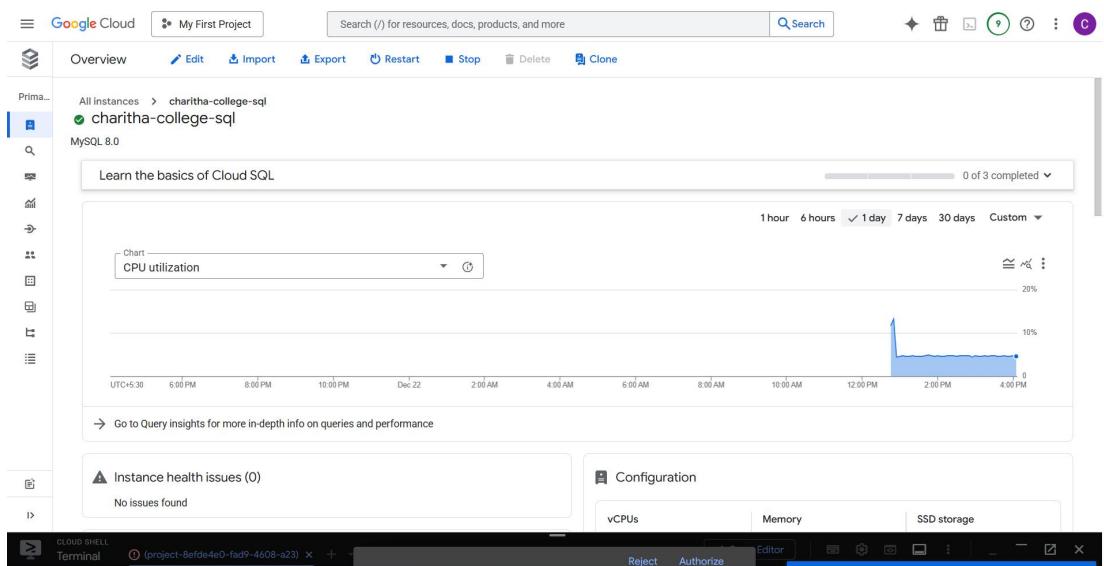


# Hands-on GCP Tasks

## A. You are building a student management system.

Requirements:

1. Create a Cloud SQL (MySQL or PostgreSQL) instance.



2. Create a database named college\_db.
3. Create a table students with columns:
  - student\_id (Primary Key)
  - name
  - department
  - marks
4. Insert at least 5 records.
5. Write SQL queries to:
  - Fetch students with marks > 75
  - Count students per department

# CollegeDB is created and inserted the data with specified columns

```
Welcome to Cloud Shell! Type "help" to get started, or type "gemini" to try prompting with Gemini CLI.
Your Cloud Platform project in this session is set to project-8efde4e0-fad9-4608-a23.
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
charithacherry07208@cloudshell:~(project-8efde4e0-fad9-4608-a23)$ gcloud sql connect charitha-college-sql --user=charitha-college-user
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [charitha-college-user].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \q.
Your MySQL connection id is 127
Server version: 8.0.41-google (Google)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database college-db
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '-db' at line 1
mysql> create database collegedb
-> ;
Query OK, 1 row affected (0.09 sec)

mysql> use collegedb;
Database changed
mysql> CREATE TABLE students (
->   student_id INT PRIMARY KEY,
->   name VARCHAR(100),
->   department VARCHAR(50),
->   marks INT
-> );
Query OK, 0 rows affected (0.10 sec)

mysql> INSERT INTO students (student_id, name, department, marks) VALUES
-> (1, 'sai', 'Information Technology', 85),
-> (2, 'charitha', 'Data Science', 72),
-> (3, 'varun', 'Computer Science', 90),
-> (4, 'sneha', 'Finance', 68);
```

## Students data is Fetched with marks > 75

```
Welcome to Cloud Shell! Type "help" to get started, or type "gemini" to try prompting with Gemini CLI.
Your Cloud Platform project in this session is set to project-8efde4e0-fad9-4608-a23.
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
charithacherry07208@cloudshell:~(project-8efde4e0-fad9-4608-a23)$ gcloud sql connect charitha-college-sql --user=charitha-college-user
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [charitha-college-user].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \q.
Your MySQL connection id is 127
Server version: 8.0.41-google (Google)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database college-db
-> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '-db' at line 1
mysql> create database collegedb
-> ;
Query OK, 1 row affected (0.09 sec)

mysql> use collegedb;
Database changed
mysql> CREATE TABLE students (
->   student_id INT PRIMARY KEY,
->   name VARCHAR(100),
->   department VARCHAR(50),
->   marks INT
-> );
Query OK, 0 rows affected (0.10 sec)

mysql> INSERT INTO students (student_id, name, department, marks) VALUES
-> (1, 'sai', 'Information Technology', 85),
-> (2, 'charitha', 'Data Science', 72),
-> (3, 'varun', 'Computer Science', 90),
-> (4, 'sneha', 'Finance', 68);
```

## 6. Secure the database by:

- Creating a read-only user
- Allowing access only from a specific IP

Google Cloud My First Project cloud sql Search

Cloud SQL Connections

Primary Instance MySQL 8.0

Overview Cloud SQL Studio System insights Query insights Connections Users Databases Backups Replicas Operations Release Notes

Networking

| Setting                 | Value   |
|-------------------------|---|
| Connection name         | project-8efde4e0-fad9-4608-a23:asia-south2:charitha-college-sql |
| Private IP connectivity | Disabled  |
| Public IP connectivity  | Enabled   |
| Public IP address       | 34.131.52.192   |

Security

| Setting                             | Value  |
|-------------------------------------|--|
| Authorized networks                 | 1 network<br>Charitha-network : 103.197.112.37 |
| Google Cloud services authorization | Disabled                                       |
| App Engine authorization            | Enabled  |
| SSL / TLS encryption                | Enabled  |
| Allow only SSL connections          | Enabled  |
| Require trusted client certificates | Disabled                                       |
| Server certificate authority mode   | Google managed internal certificate authority  |
| Server certificate expiration time  | Dec 20, 2035, 12:42:02 PM                      |

Need help connecting? Tools, guided workflows and helpful links to help you connect to your Cloud SQL instance.

Test Connection from GCE Docs: Connection Overview

## Expected Skills Tested

- Cloud SQL setup
- Basic SQL + permissions
- Networking & security basics

## B. You are creating a real-time chat application backend.

Requirements:

1. Enable Firestore (Native mode).  
-- Enabled the firestore to the native mode during the creation firestoreDB
2. Create a collection chats.  
-- Created the chats collection to store the messages between the users
3. Each document should store:
  - sender
  - receiver
  - message
  - timestamp

The screenshot shows the Google Cloud Firestore interface. On the left, the sidebar includes 'Google Cloud' and 'My First Project' at the top, followed by sections for 'Database' (selected), 'Firestore Studio' (selected), 'Security', 'Indexes', 'Import/Export', 'Disaster Recovery', 'Time-to-live (TTL)', 'Insights' (Usage, Query insights, Monitoring, Key Visualizer), and 'Release Notes'. The main area shows a database structure under 'college-db': 'chats' > 'Document3' > 'Document1' through 'Document10'. A detailed view of 'Document3' is shown on the right, containing fields: 'message' (value: 'Doing great'), 'receiver' (value: 'Charitha'), 'sender' (value: 'Anjani'), and 'timestamp' (value: 'December 22, 2025 at 1:23:31.075PM...').

## 4. Perform the following:

- Insert at least 10 chat messages

The screenshot shows the Google Cloud Firestore Studio interface. On the left, there's a sidebar with 'Firestore' selected under 'Database'. The main area has 'Query builder' selected. A table titled 'Query results' lists 10 documents from the '/chats' collection. Each document includes fields: Document ID, message, receiver, sender, and timestamp. The messages are as follows:

| Document ID | message                    | receiver   | sender     | timestamp                                |
|-------------|----------------------------|------------|------------|--|
| Document3   | "Doing great"              | "Charitha" | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document6   | "Heyyy"                    | "Varun"    | "Charitha" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document7   | "What's up"                | "Charitha" | "Varun"    | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document1   | "Hi! How are you?"         | "Charitha" | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document10  | "Okay, will catch up soon" | "Charitha" | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document2   | "I am Good. wt about u ?"  | "Anjani"   | "Charitha" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document4   | "Hello there"              | "varun"    | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document5   | "Hii!"                     | "Anjani"   | "Varun"    | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document8   | "Meeting at 5 "            | "Charitha" | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document9   | "Sure, will see u "        | "Anjani "  | "Charitha" | December 22, 2025 at 1:23:31 PM UTC+5:30 |

- Query all messages between two users

The screenshot shows the Google Cloud Firestore Studio interface with a query builder. The 'Query scope' is set to 'Collection' with the path '/chats'. The 'Limit' is set to 100. In the 'Selection' section, a 'WHERE' clause is defined with the field 'sender' and operator '==' followed by the value 'Anjani'. The resulting table shows 5 documents where the sender is 'Anjani'. The messages are:

| Document ID | message                    | receiver   | sender   | timestamp                                |
|-------------|----------------------------|------------|----------|--|
| Document3   | "Doing great"              | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document1   | "Hi! How are you?"         | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document10  | "Okay, will catch up soon" | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document4   | "Hello there"              | "varun"    | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document8   | "Meeting at 5 "            | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |

At the bottom, the URL is https://console.cloud.google.com/firestore/databases/college-db/data?project=project.

The screenshot shows the Google Cloud Firestore Query builder interface. The left sidebar includes sections for Database, Security, Indexes, Import/Export, Disaster Recovery, Time-to-live (TTL), Insights (Usage, Query insights, Monitoring, Key Visualizer), and Release Notes. The main area displays a query builder with the following configuration:

- Query scope: Collection /chats
- Limit: 100
- Selection: WHERE receiver = "Charitha"

The results table shows five documents:

| Document ID | message                    | receiver   | sender   | timestamp                                |
|-------------|----------------------------|------------|----------|--|
| Document3   | "Doing great"              | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document7   | "What's up"                | "Charitha" | "Varun"  | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document1   | "Hi! How are you?"         | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document10  | "Okay, will catch up soon" | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document8   | "Meeting at 5"             | "Charitha" | "Anjani" | December 22, 2025 at 1:23:31 PM UTC+5:30 |

At the bottom, there are pagination controls: Rows per page: 50, 1 – 5 of 5.

- Sort messages by timestamp

The screenshot shows the Google Cloud Firestore Query builder interface, identical to the previous one but with a different sorting condition. The selection dropdown now contains "timestamp".

The results table shows the same five documents, but they are listed in chronological order from oldest to newest based on the timestamp field.

| Document ID | message                    | receiver   | sender     | timestamp                                |
|-------------|----------------------------|------------|------------|--|
| Document3   | "Doing great"              | "Charitha" | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document6   | "Heyyy"                    | "Varun"    | "Charitha" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document7   | "What's up"                | "Charitha" | "Varun"    | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document1   | "Hi! How are you?"         | "Charitha" | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document10  | "Okay, will catch up soon" | "Charitha" | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document2   | "I am Good. wt about u ?"  | "Anjani"   | "Charitha" | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document4   | "Hello there"              | "varun"    | "Anjani"   | December 22, 2025 at 1:23:31 PM UTC+5:30 |
| Document5   | "Hii!"                     | "Anjani"   | "Varun"    | December 22, 2025 at 1:23:31 PM UTC+5:30 |

## 5. Create a Firestore security rule:

- Only authenticated users can read/write

The screenshot shows the Google Cloud Platform interface for a project named 'My First Project'. Under the 'Database' section, 'Firestore' is selected. In the left sidebar under 'Database', 'Security' is highlighted. The main content area displays the 'Security Rules' page. It includes a note about security rules protecting the database from unauthorized access. Below this is a code editor containing the default security rules for Firestore:

```
1 rules_version = '2';
2 service cloud.firestore {
3     match /databases/{database}/documents {
4         match /{document=**} {
5             allow read, write: if false;
6         }
7     }
8 }
```

At the bottom of the code editor, there is a note: "You can write or edit rules using the Firebase console or command-line interface once you enable Firebase." A blue button labeled "Enable Firebase" is visible.

- Users can read only their own chats

## Expected Skills Tested

- Firestore data modeling
- NoSQL querying
- Security rules

# C. You are designing an e-commerce platform.

## Requirements:

1. Use Cloud SQL to store:

- Orders
- Payments

=> Created the Sql instance and then created the ecommerce database as ecommerce-db

=> To authenticate it, created the user with password so that other unauthenticated cant read/write the data

The screenshot shows the Google Cloud Platform Cloud SQL interface. On the left, there's a sidebar with options like Overview, Cloud SQL Studio, System insights, Query insights, Connections, Users, Databases (which is selected), Backups, Replicas, and Operations. Below the sidebar, there's a Release Notes section. The main area shows a primary instance named 'ecommerce-sql' with a single database named 'ecommerce-db'. The table below lists the database details:

| Name               | Collation          | Character set | Type   |
|--------------------|--------------------|---------------|--------|
| ecommerce_db       | utf8mb4_0900_ai_ci | utf8mb4       | User   |
| information_schema | utf8mb3_general_ci | utf8mb3       | System |
| mysql              | utf8mb3_general_ci | utf8mb3       | System |
| performance_schema | utf8mb4_0900_ai_ci | utf8mb4       | System |
| sys                | utf8mb4_0900_ai_ci | utf8mb4       | System |

At the bottom right, there's a 'Logs' tab and a sidebar titled 'Uploads and My First Project operations' showing two recent log entries: 'Created ecommerce-sql.' at 2:35:35 PM GMT+5 and 'Created charitha-college-sql.' at 12:46:21 PM GMT+5.

Google Cloud My First Project Search (/) for resources, docs, products, and more

CLOUD SHELL Terminal (project-8efde4e0-fad9-4608-a23) + Open Editor

Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more

```
Welcome to Cloud Shell! Type "help" to get started, or type "gemini" to try prompting with Gemini CLI.
Your Cloud Platform project in this session is set to project-8efde4e0-fad9-4608-a23.
Use `gcloud config set project [PROJECT_ID]` to change to a different project.
charithacherry@206elabshell:~$ gcloud sql connect ecommerce-sql --user=ecommerce-user
Allowing up to 5 minutes for connection...done.
Connecting to database with SQL user [ecommerce-user]. Enter password:
Welcome to the MySQL monitor. Commands end with ; or \q.
Your MySQL connection id is 856
Server version: 8.0.41-google (Google)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use ecommerce-db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> desc Orders;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| order_id | int | NO | PRI | NULL | auto_increment |
| user_id | varchar(50) | YES | NULL | NULL | |
| order_amount | decimal(10,2) | YES | NULL | NULL | |
| order_date | timestamp | YES | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----+-----+-----+-----+
5 rows in set (0.09 sec)

mysql> INSERT INTO Orders (user_id, order_amount) VALUES
-> ('Sal', 2500),
-> ('Charitha', 1800),
-> ('Sujan', 4000),
-> ('Shreya', 1200),
-> ('Varun', 3000);
```

CLOUD SHELL Terminal (project-8efde4e0-fad9-4608-a23) + Open Editor

Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more

```
Query OK, 5 rows affected (0.09 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from Orders;
+-----+-----+-----+-----+
| order_id | user_id | order_amount | order_date |
+-----+-----+-----+-----+
| 1 | Sal | 2500.00 | 2025-12-22 09:41:24 |
| 2 | Charitha | 1800.00 | 2025-12-22 09:41:24 |
| 3 | Sujan | 4000.00 | 2025-12-22 09:41:24 |
| 4 | Shreya | 1200.00 | 2025-12-22 09:41:24 |
| 5 | Varun | 3000.00 | 2025-12-22 09:41:24 |
+-----+-----+-----+-----+
5 rows in set (0.08 sec)

mysql> CREATE TABLE Payments (
-> payment_id INT PRIMARY KEY AUTO_INCREMENT,
-> order_id INT,
-> payment_status VARCHAR(20)
-> );
Query OK, 0 rows affected (0.11 sec)

mysql> desc Payments;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| payment_id | int | NO | PRI | NULL | auto_increment |
| order_id | int | YES | NULL | NULL | |
| payment_status | varchar(20) | YES | NULL | NULL | |
+-----+-----+-----+-----+
3 rows in set (0.08 sec)

mysql> INSERT INTO payments (order_id, payment_status) VALUES
-> (1, 'SUCCESS'),
-> (2, 'FAILED'),
-> (3, 'SUCCESS'),
-> (4, 'SUCCESS'),
-> (5, 'SUCCESS');
Query OK, 5 rows affected (0.09 sec)
```

CLOUD SHELL Terminal (project-8efde4e0-fad9-4608-a23) + Open Editor

Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more

```
mysql> CREATE TABLE Payments (
-> payment_id INT PRIMARY KEY AUTO_INCREMENT,
-> order_id INT,
-> payment_status VARCHAR(20)
-> );
Query OK, 0 rows affected (0.11 sec)

mysql> desc Payments;
+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| payment_id | int | NO | PRI | NULL | auto_increment |
| order_id | int | YES | NULL | NULL | |
| payment_status | varchar(20) | YES | NULL | NULL | |
+-----+-----+-----+-----+
3 rows in set (0.08 sec)

mysql> INSERT INTO payments (order_id, payment_status) VALUES
-> (1, 'SUCCESS'),
-> (2, 'FAILED'),
-> (3, 'SUCCESS'),
-> (4, 'SUCCESS'),
-> (5, 'SUCCESS');
Query OK, 5 rows affected (0.09 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * from payments;
+-----+-----+-----+-----+
| payment_id | order_id | payment_status | payment_date |
+-----+-----+-----+-----+
| 21 | 1 | SUCCESS | 2025-12-22 09:44:17 |
| 22 | 2 | FAILED | 2025-12-22 09:44:17 |
| 23 | 3 | SUCCESS | 2025-12-22 09:44:17 |
| 24 | 4 | SUCCESS | 2025-12-22 09:44:17 |
| 25 | 5 | SUCCESS | 2025-12-22 09:44:17 |
+-----+-----+-----+-----+
5 rows in set (0.09 sec)

mysql> 
```

## 2. Use Firestore or Bigtable to store:

- User activity logs

The screenshot shows the Google Cloud Firestore Studio interface. On the left, the sidebar includes sections for Database (Security, Indexes, Import/Export, Disaster Recovery, Time-to-live (TTL)), Insights (Usage, Query insights, Monitoring, Key Visualizer), and Release Notes. The main area displays a collection named 'user\_activity' under the database 'ecommerce-db'. The collection contains several documents, each representing a user activity log. One document is expanded to show its fields: activity (e.g., 'product-click'), category (e.g., 'electronics'), device (e.g., 'tablet'), page (e.g., '/product/adidas\_shoes'), product (e.g., 'adidas\_shoes'), timestamp (e.g., 'December 21, 2025 at 3:53:26.060 PM...'), and user-id (e.g., 'U1002').

- Clickstream data

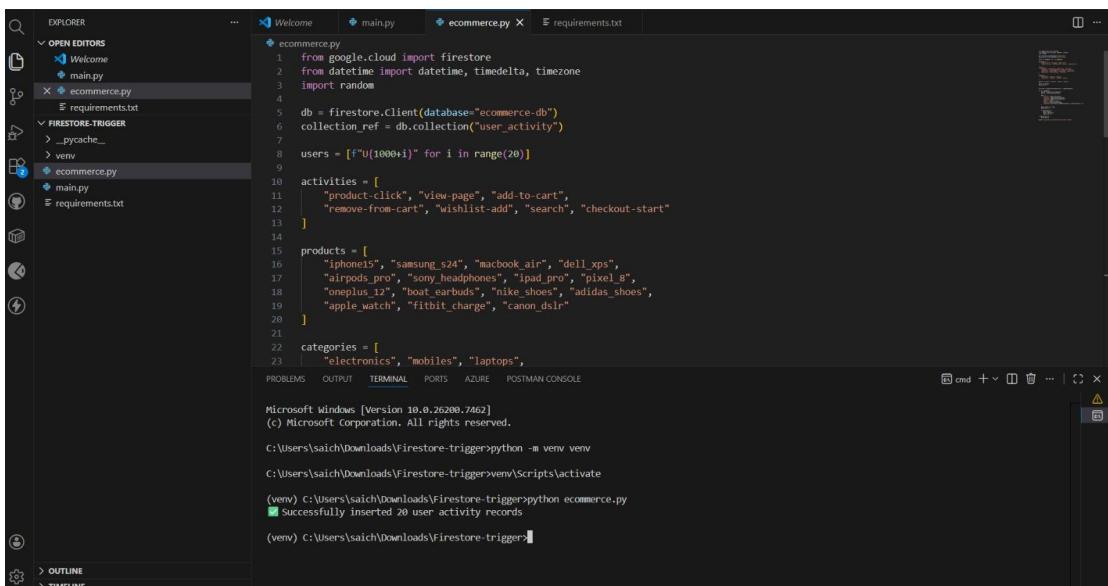
The screenshot shows the Google Cloud Firestore Studio interface with the 'Query builder' tab selected. It allows users to define a query scope (Collection: '/user\_activity', Limit: 100) and add filters. Below the query builder, the 'Results' tab is active, displaying a table of query results. The columns include Document ID, activity, category, device, page, product, timestamp, and user-id. The results list various user interactions, such as product-clicks on electronics devices like tablets and mobile phones, and view-pages on fashion and accessories pages.

| Document ID           | activity         | category      | device    | page                     | product         | timestamp              | user-id |
|-----------------------|------------------|---------------|-----------|--------------------------|-----------------|------------------------|---------|
| 1FCgBa4F0jAomdPiaE0J  | 'product-click'  | 'electronics' | 'tablet'  | '/product/adidas_shoes'  | 'adidas_shoes'  | December 21, 2025 a... | "U1002" |
| 5Z7A3rz6Czq7PqhuCusE  | 'view-page'      | 'fashion'     | 'mobile'  | '/product/airpods_pro'   | 'airpods_pro'   | December 21, 2025 a... | "U1011" |
| SeuoUj5SO8ZnXYtMzXnH  | 'search'         | 'fashion'     | 'tablet'  | '/product/dell_xps'      | 'dell_xps'      | December 21, 2025 a... | "U1009" |
| 5j8DriJhj55ndaL4hSs   | 'product-click'  | 'accessories' | 'laptop'  | '/product/adidas_shoes'  | 'adidas_shoes'  | December 21, 2025 a... | "U1011" |
| 7RFxsbe8lxzal8xGtfVg  | 'view-page'      | 'fashion'     | 'desktop' | '/product/canon_dslr'    | 'canon_dslr'    | December 21, 2025 a... | "U1013" |
| BrCbzrOvef9xmxCYALP   | 'remove-from...' | 'electronics' | 'desktop' | '/product/canon_dslr'    | 'canon_dslr'    | December 21, 2025 a... | "U1001" |
| HckbIXF0S325r5wgTK    | 'view-page'      | 'accessories' | 'tablet'  | '/product/fitbit_charge' | 'fitbit_charge' | December 21, 2025 a... | "U1006" |
| JzvbuUgOWIW6Vlz2W75K  | 'product-click'  | 'accessories' | 'tablet'  | '/product/canon_dslr'    | 'canon_dslr'    | December 21, 2025 a... | "U1003" |
| VZDQHJU4-A-1PAUd1d4D1 | 'product-click'  | 'fashion'     | 'mobile'  | '/product/fitbit_charge' | 'fitbit_charge' | December 21, 2025 a... | "U1011" |

### 3. Implement:

- One SQL query to fetch total orders per user
- One NoSQL query to fetch last 50 user activities

activities



```
ecommerce.py
1 from google.cloud import firestore
2 from datetime import datetime, timedelta, timezone
3 import random
4
5 db = firestore.Client(database="ecommerce-db")
6 collection_ref = db.collection("user_activity")
7
8 users = [f"U{1000+i}" for i in range(20)]
9
10 activities = [
11     "product-click", "view-page", "add-to-cart",
12     "remove-from-cart", "wishlist-add", "search", "checkout-start"
13 ]
14
15 products = [
16     "iphones15", "samsung_s24", "macbook_air", "dell_xps",
17     "airpods_pro", "sony_headphones", "ipad_pro", "pixel_8",
18     "oneplus_12", "boat_earbuds", "nike_shoes", "adidas_shoes",
19     "apple_watch", "fitbit_charge", "canon_dslr"
20 ]
21
22 categories = [
23     "electronics", "mobiles", "laptops",
24 ]
```

Microsoft Windows [Version 10.0.26280.7462]  
(c) Microsoft Corporation. All rights reserved.  
C:\Users\saich\Downloads\Firestore-trigger>python -m venv venv  
C:\Users\saich\Downloads\Firestore-trigger>venv\Scripts\activate  
(venv) C:\Users\saich\Downloads\Firestore-trigger>python ecommerce.py  
Successfully inserted 20 user activity records  
(venv) C:\Users\saich\Downloads\Firestore-trigger>

### 4. Explain (practically):

#### I) Why SQL is Chosen for Transactions (Orders & Payments)

SQL databases (like Cloud SQL – MySQL/PostgreSQL) are used for transactions because they guarantee data correctness and reliability, which is critical in e-commerce systems.

Reasons beyond:

##### 1) ACID Properties

Atomicity: Either the whole transaction succeeds or fails

Consistency: Data always remains valid.

Isolation: Multiple users can place orders at the same time without conflicts.

Durability: Once payment is confirmed, data is never lost.

## 2) Strong Consistency

When a payment is marked SUCCESS, the order status must update immediately.

SQL ensures all users see the same correct data instantly.

## 3) Relational Integrity

Orders and payments are related.

Payment cannot exist without an order

Invalid data is blocked automatically

## 4) Rollback Support

If payment fails:

Order creation can be rolled back

No partial or corrupted data is stored

Example:

A user places an order → payment fails → order must not be confirmed

## II) Why NoSQL is Chosen for Logs & Clickstream Data

NoSQL databases (like Firestore / Bigtable) are used for logs and user activity because they are built for scale, speed, and flexibility.

Reason Beyond:

### 1) High Write Volume

User clicks, page views, and activities happen every second

NoSQL can handle millions of writes per second

## 2) Flexible Schema

Log structure can change anytime

New fields can be added without schema changes

## 3) Horizontal Scalability

Data automatically spreads across multiple servers

No performance bottleneck as traffic grows

## 4) Time-Based Queries

Logs are usually read as:

“Last 50 activities”

“Recent user actions”

### Example

An e-commerce site tracks millions of clicks per day.

### Expected Skills Tested

- GCP service selection
- Data modeling
- Real-world architecture decisions