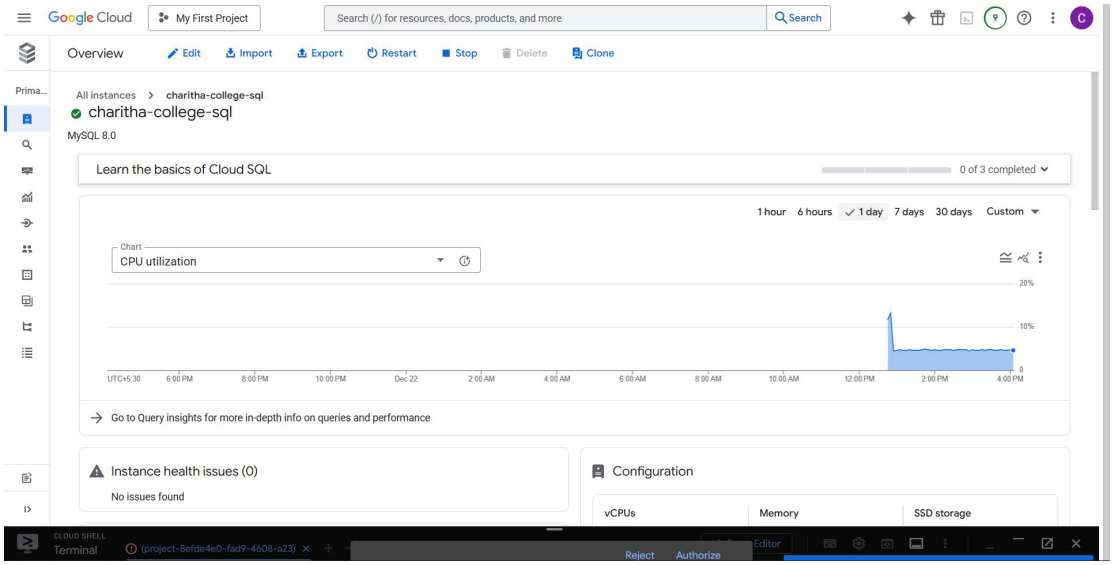
Hands-on GCP Tasks

A. You are building a student management system.

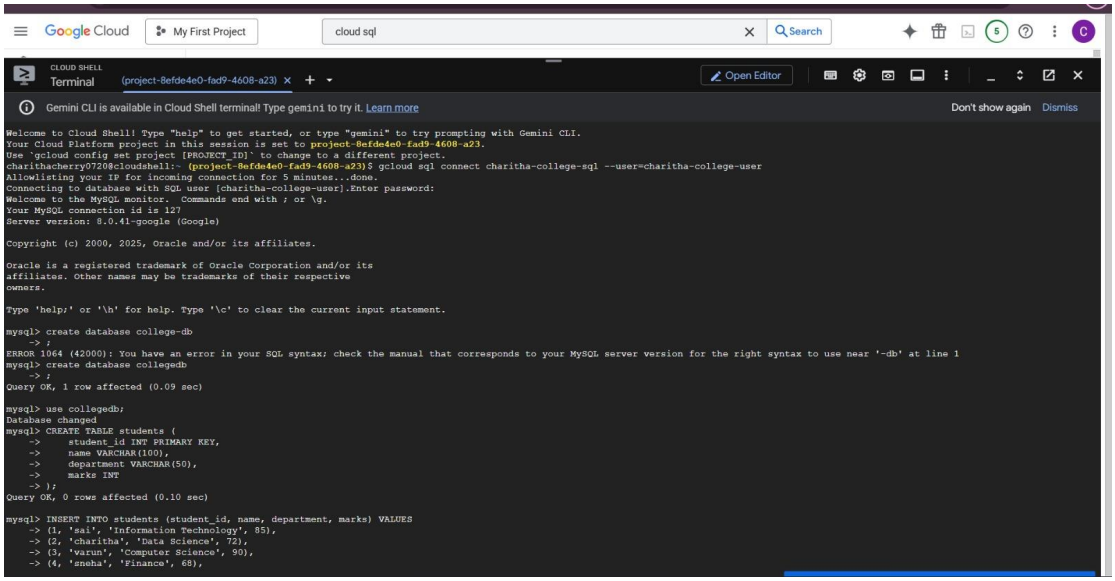
Requirements:

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

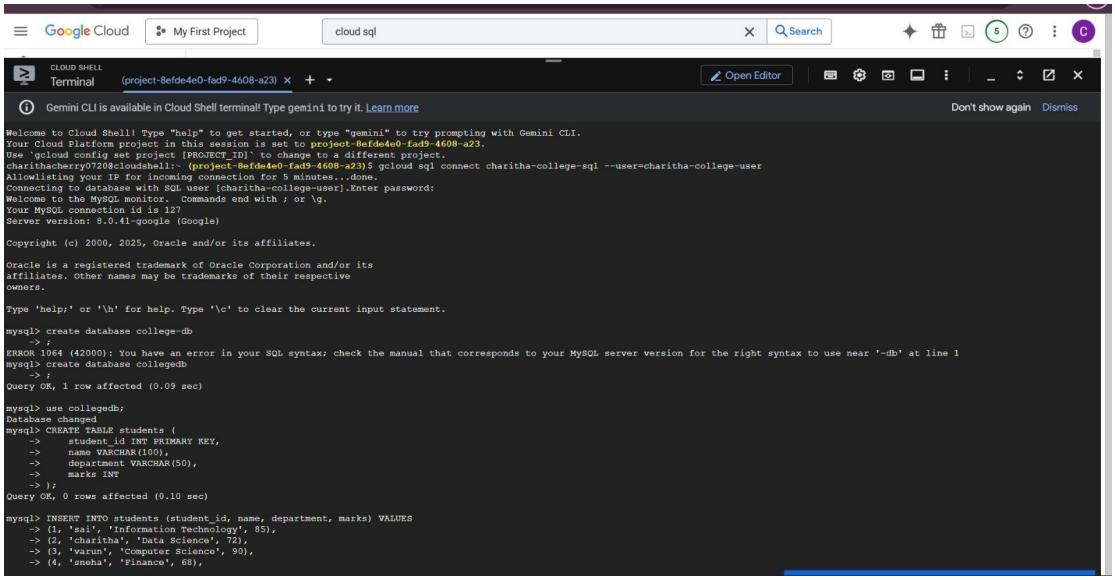


2. Create a database named college_db.
3. Create a table students with columns:
   - student_id (Primary Key)
   - name
   - department
   - marks
4. Insert at least 5 records.
5. Write SQL queries to:
   - Fetch students with marks > 75
   - Count students per department

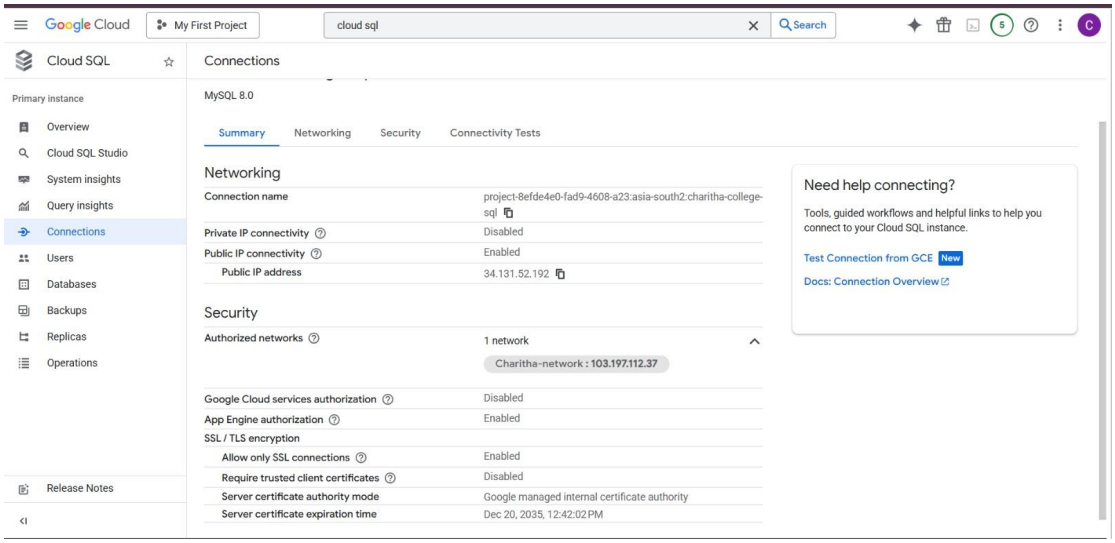
CollegeDB is created and inserted the data with specified columns



Students data is Fetched with marks > 75



6. Secure the database by:
    - Creating a read-only user
    - Allowing access only from a specific IP



Expected Skills Tested

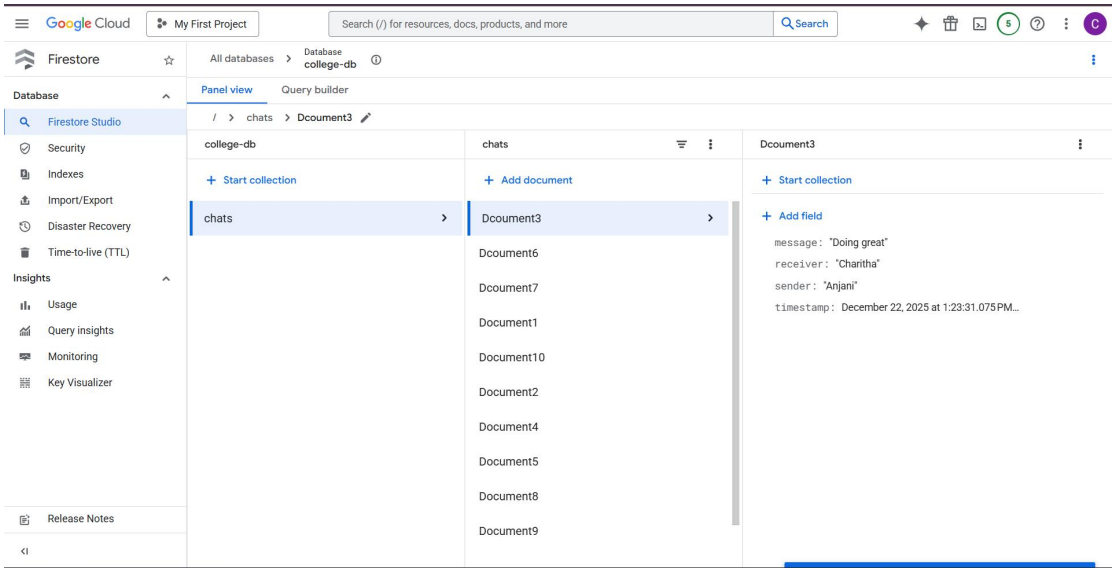
- Cloud SQL setup
- Basic SQL + permissions
- Networking & security basics

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

Requirements:

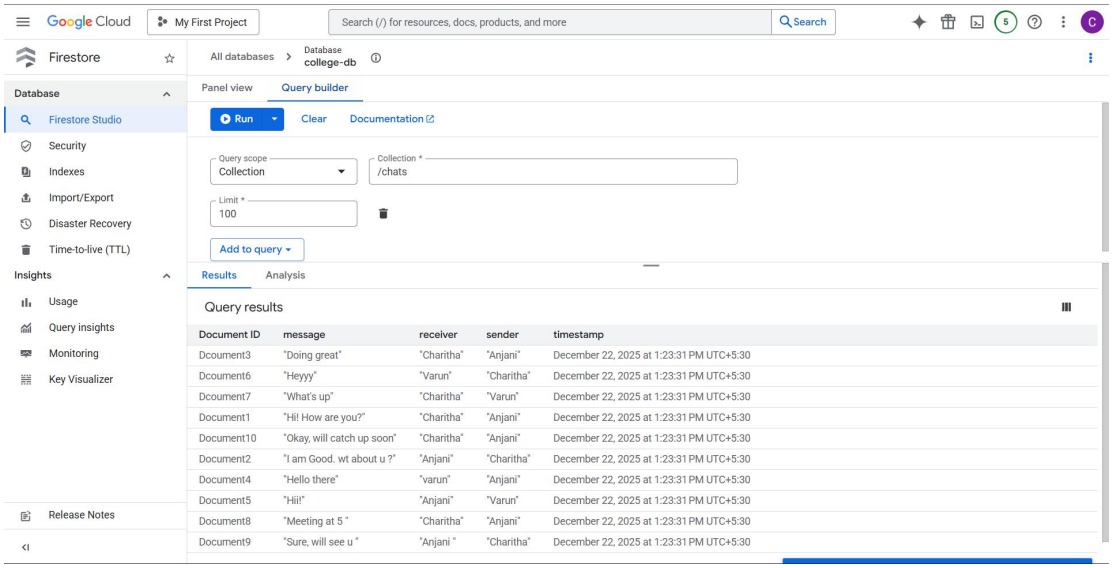
1. Enable Firestore (Native mode).
   -- Enabled the firestore to the native mode during the creation firestoreDB
2. Create a collection chats.
   -- Created the chats collection to store the messages between the users

3. Each document should store:
   - sender
   - receiver
   - message
   - timestamp

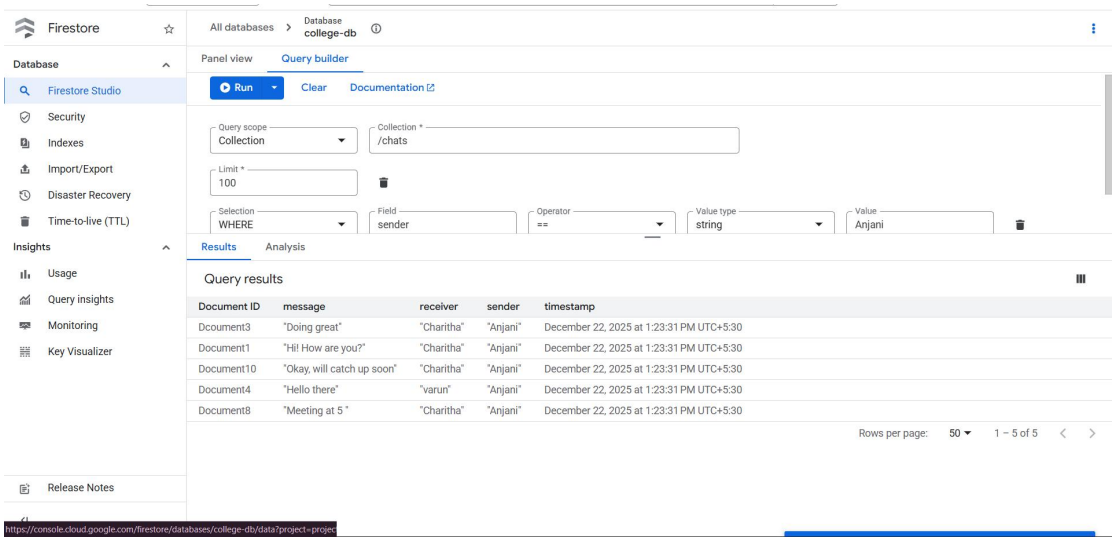


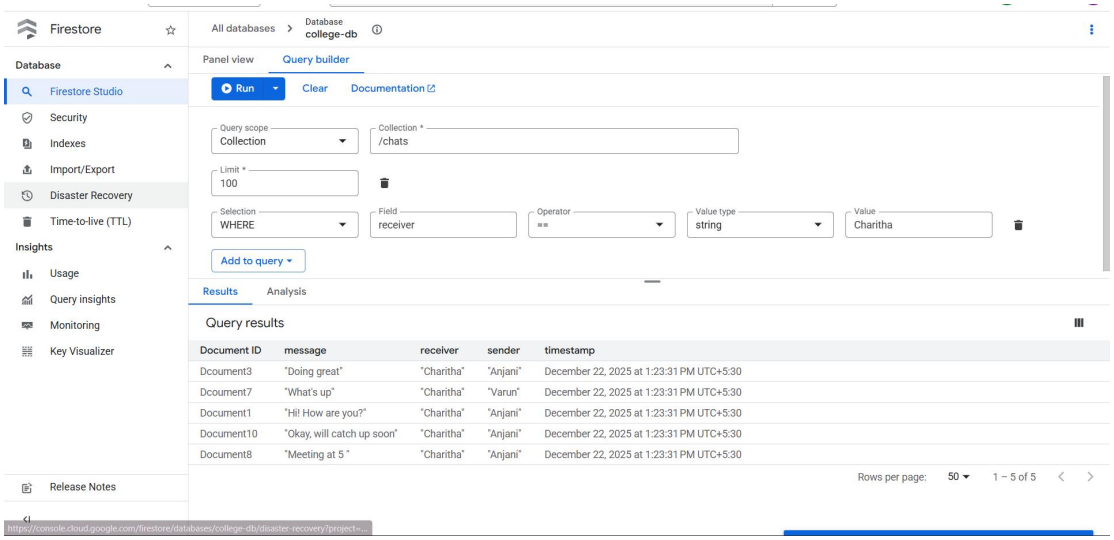
4. Perform the following:

- Insert at least 10 chat messages

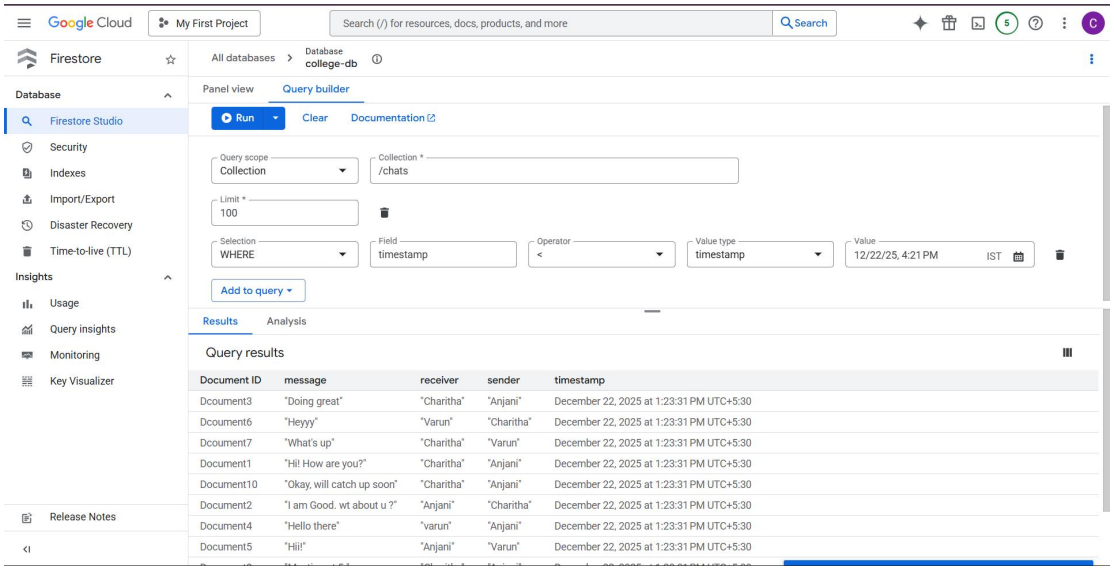


- Query all messages between two users



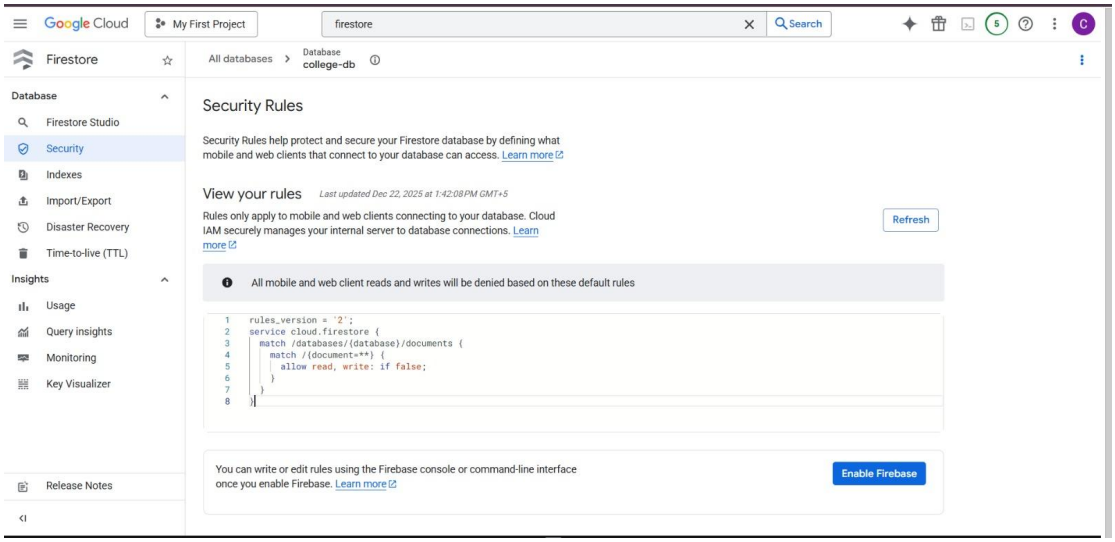


- Sort messages by timestamp



5. Create a Firestore security rule:

- Only authenticated users can read/write



- Users can read only their own chats

Expected Skills Tested

- Firestore data modeling
- NoSQL querying
- Security rules

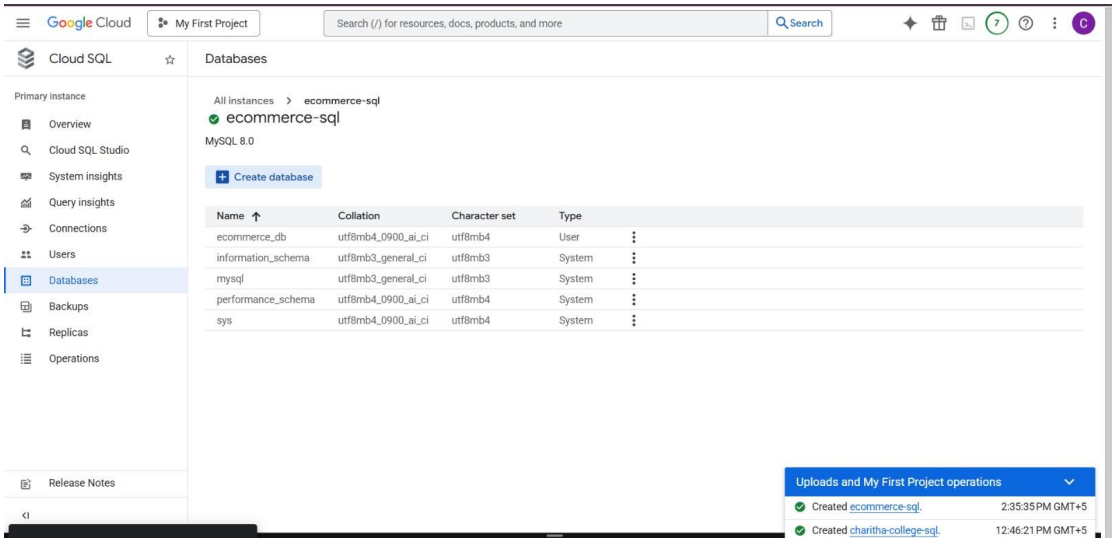
C. You are designing an e-commerce platform.

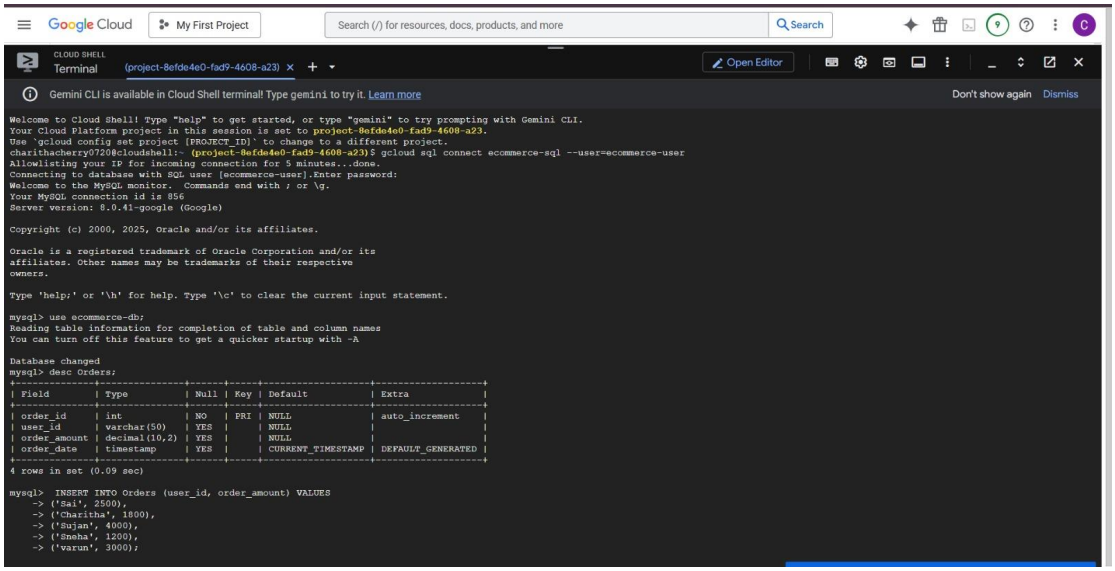
Requirements:

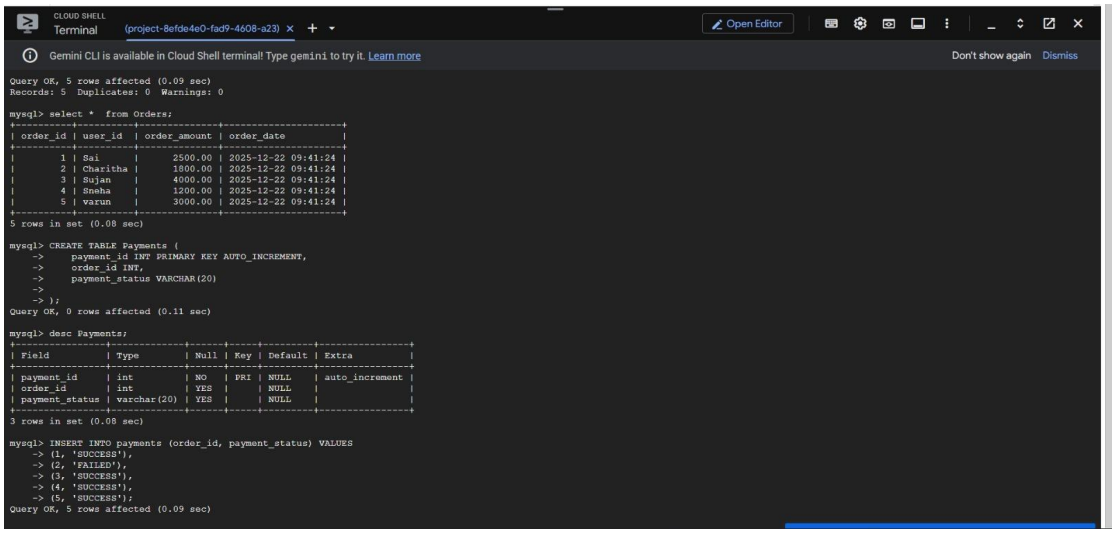
1. Use Cloud SQL to store:
- Orders
- Payments

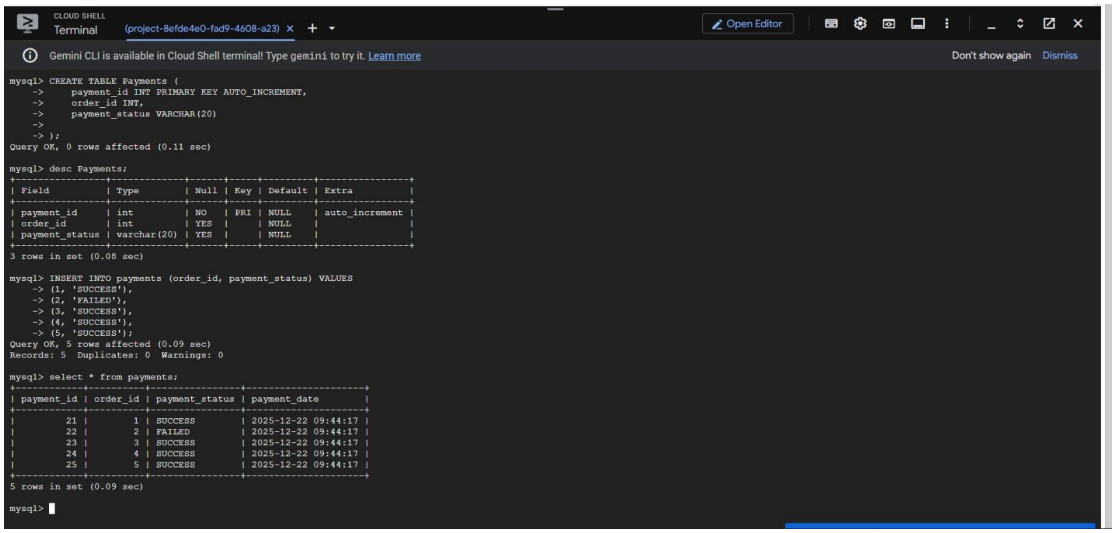
=> Created the Sql instance and then created the ecommerce database as ecommerce-db

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



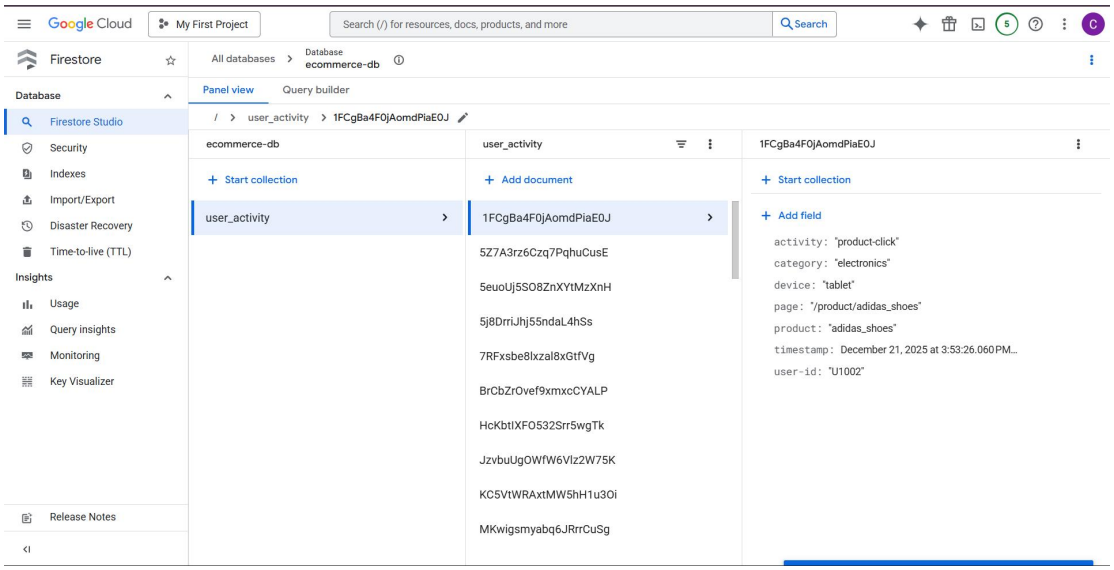

Google Cloud
My First Project
Search (/) for resources, docs, products, and more
Search
CLOUD SHELL
Terminal
(project-8efde4e0-fad9-4608-a23)
Open Editor
Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more
Don't show again
Dismiss
Welcome to Cloud Shell! Type "help" to 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.
charithacherry0720@cloudshell:~ (project-8efde4e0-fad9-4608-a23)$ gcloud sql connect ecommerce-sql --user=ecommerce-user
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [ecommerce-user].Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
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	
order_amount	decimal(10,2)	YES		NULL	
order_date	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
+--------------+---------------+------+-----+-------------------+-------------------+
4 rows in set (0.09 sec)
mysql> INSERT INTO Orders (user_id, order_amount) VALUES
-> ('Sai', 2500),
-> ('Charitha', 1800),
-> ('Sujan', 4000),
-> ('Sneha', 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
Don't show again
Dismiss
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	Sai	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	Sneha	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	
payment_status	varchar(20)	YES		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)

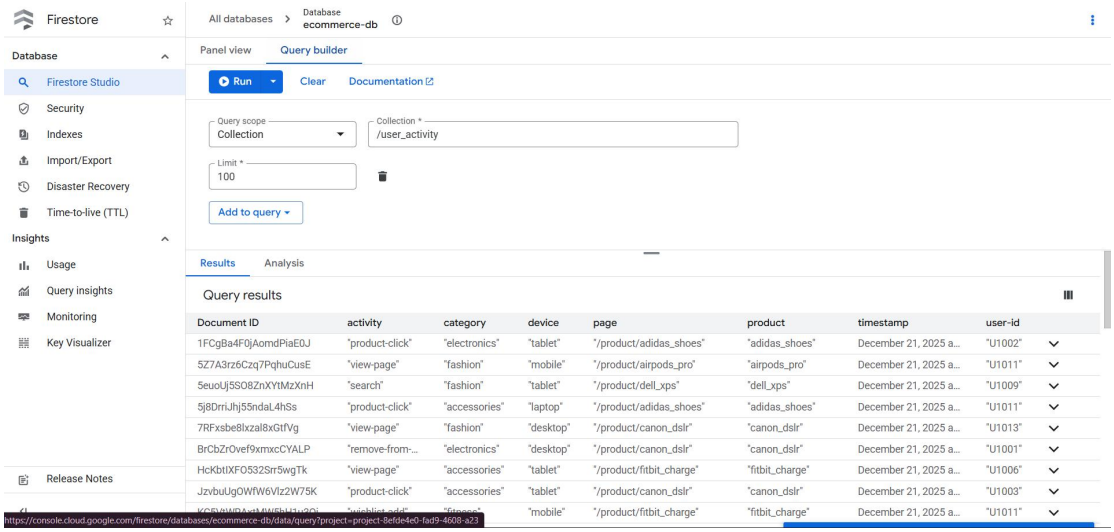

CLOUD SHELL
Terminal
(project-8efde4e0-fad9-4608-a23)
Open Editor
Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more
Don't show again
Dismiss
mysql> CREATE TABLE Payments (
-> payment_id INT PRIMARY KEY AUTO_INCREMENT,
-> order_id INT,
-> payment_status VARCHAR(20)
->
->);
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	
payment_status	varchar(20)	YES		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:

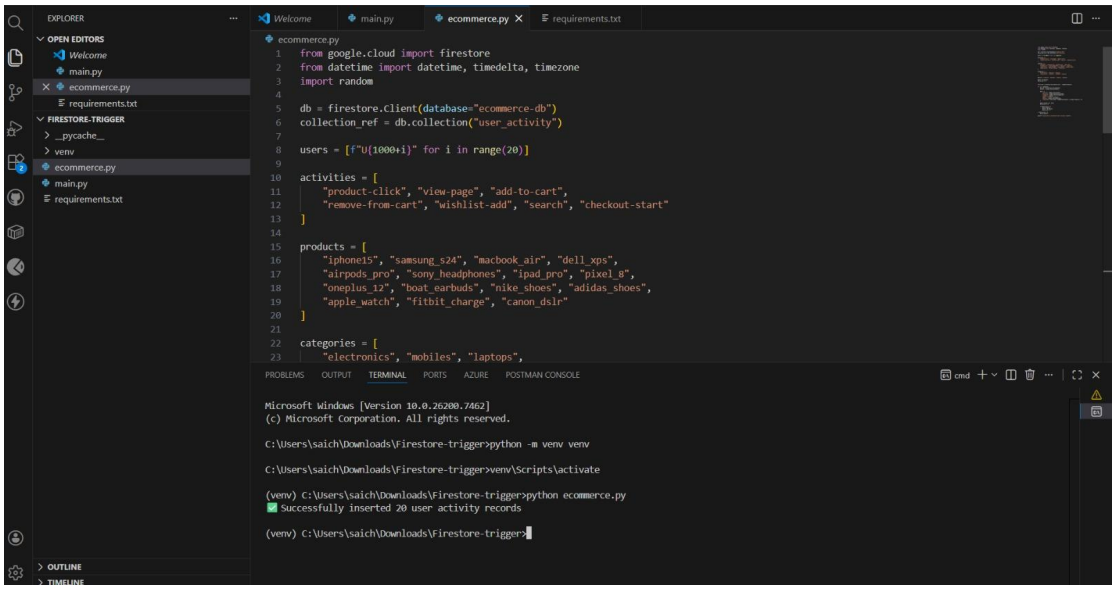
- User activity logs



- Clickstream data



3. Implement:
    - One SQL query to fetch total orders per user
    - One NoSQL query to fetch last 50 user activities



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