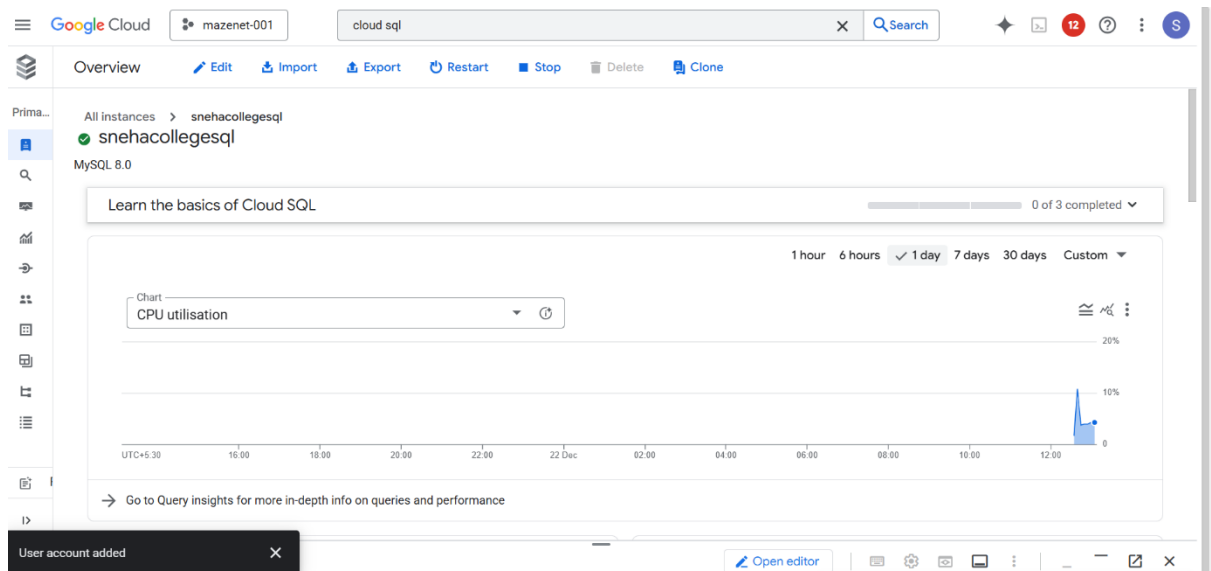


Task1: Student management system

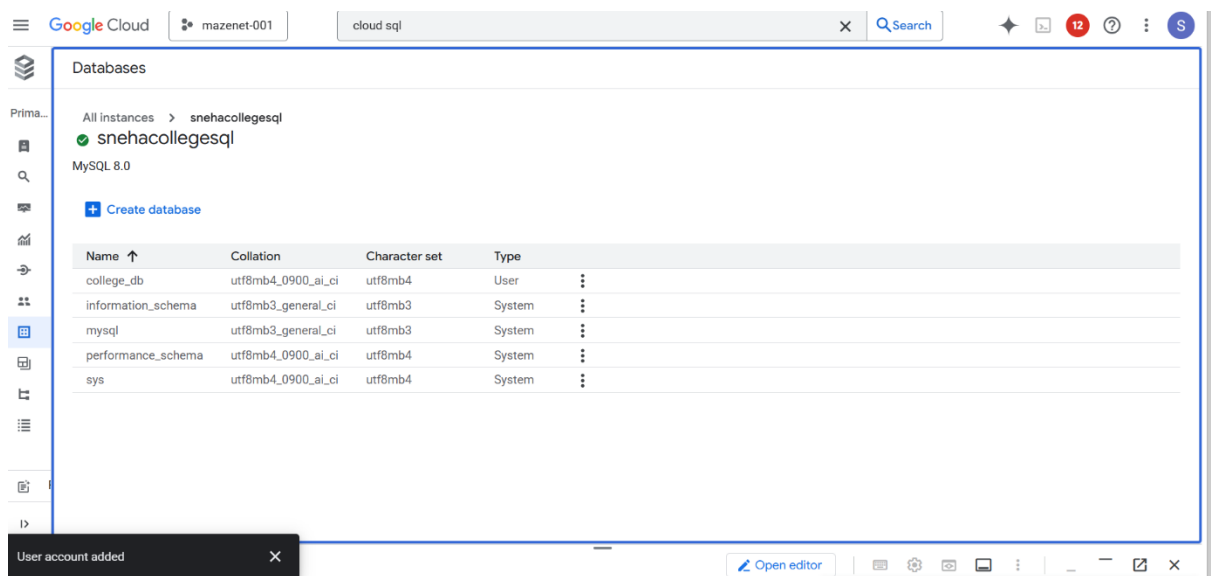
1. Create Cloud SQL Instance

- Go to Google Cloud Console → SQL → Create Instance.
- Choose MySQL
- Click Create and wait for the instance to be ready.



2. Create Database

- Once instance is ready, go to Databases tab.
- Click Create database → Name it college_db



3.create student table

```
mysql> USE college_db;
Database changed
mysql> SHOW TABLES;
Empty set (0.08 sec)

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> DESCRIBE students;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| student_id | int | NO | PRI | NULL | |
| name | varchar(100) | YES | | NULL | |
| department | varchar(50) | YES | | NULL | |
| marks | int | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.07 sec)

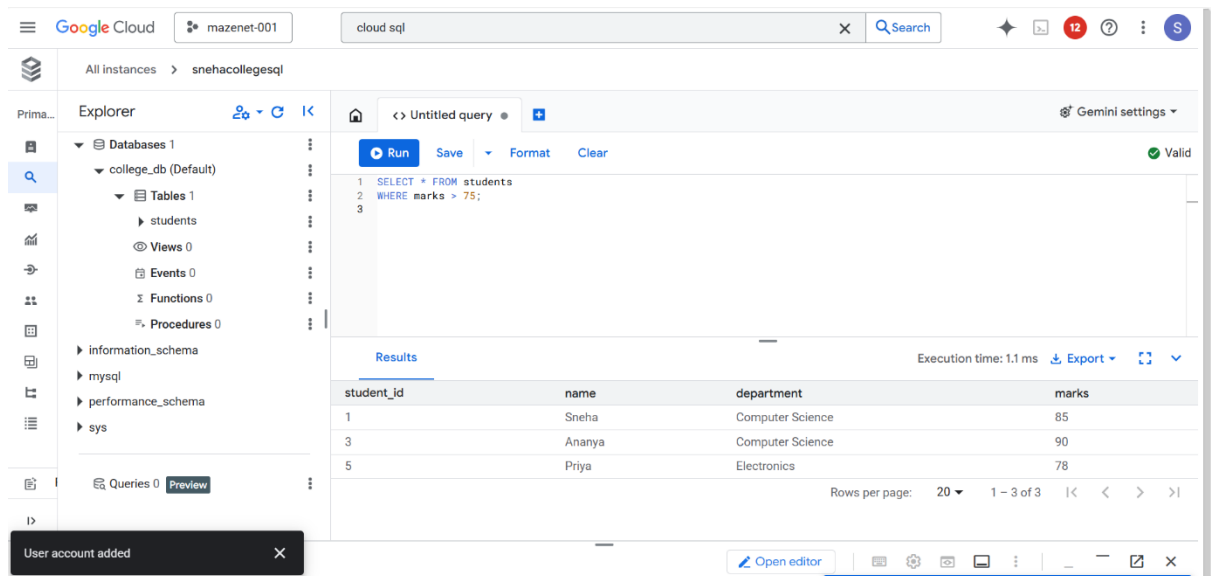
mysql> INSERT INTO students VALUES
  -> (1, 'Sneha', 'Computer Science', 85),
  -> (2, 'Rahul', 'Electronics', 72),
  -> (3, 'Ananya', 'Computer Science', 90),
  -> (4, 'Vikram', 'Mechanical', 68),
  -> (5, 'Priya', 'Electronics', 78);
Query OK, 5 rows affected (0.08 sec)

mysql> SELECT * FROM students;
+-----+-----+-----+-----+
| student_id | name | department | marks |
+-----+-----+-----+-----+
| 1 | Sneha | Computer Science | 85 |
| 2 | Rahul | Electronics | 72 |
| 3 | Ananya | Computer Science | 90 |
| 4 | Vikram | Mechanical | 68 |
| 5 | Priya | Electronics | 78 |
+-----+-----+-----+-----+
5 rows in set (0.08 sec)
```

6. Fetch Students with Marks > 75

Query:

- SELECT * FROM students WHERE marks > 75;



The screenshot shows the Google Cloud SQL console interface. On the left, the 'Explorer' pane shows the database structure for 'college_db (Default)', including a table named 'students'. The main area displays an 'Untitled query' with the SQL statement: `SELECT * FROM students WHERE marks > 75;`. The 'Run' button is highlighted. Below the query, the 'Results' section shows the execution time as 1.1 ms and a table with 3 rows of data:

student_id	name	department	marks
1	Sneha	Computer Science	85
3	Ananya	Computer Science	90
5	Priya	Electronics	78

At the bottom, a 'User account added' notification is visible.

7. Count Students per Department

Query:

```
SELECT department, COUNT(*) AS total_students FROM students GROUP BY department;
```

The screenshot shows the Google Cloud SQL console interface. On the left, the Explorer pane displays the database structure for 'college_db (Default)', including tables like 'students', 'information_schema', 'mysql', 'performance_schema', and 'sys'. The main editor shows a query titled '<> Untitled query' with the following SQL code:

```
1 SELECT department, COUNT(*) AS student_count
2 FROM students
3 GROUP BY department;
4
5
```

The query has been executed successfully, as indicated by the 'Valid' status and the 'Results' pane. The results show the following data:

department	student_count
Computer Science	2
Electronics	2
Mechanical	1

The execution time is 0.8 ms. A 'User account added' notification is visible at the bottom left.

8. Create Read-Only User

- SQL:
- `CREATE USER 'readonly'@'%' IDENTIFIED BY 'password';`
- `GRANT SELECT ON college_db.* TO 'readonly'@'%';`
- `FLUSH PRIVILEGES;`

The screenshot shows the Google Cloud SQL console interface. The main editor displays the following SQL code:

```
1 CREATE USER 'readonly_user'@'%' IDENTIFIED BY 'ReadOnly#123';
2
3
4
```

The query has been executed, but it resulted in a syntax error, as indicated by the 'Syntax error at or near "USER"' message. The execution time is 51.2 ms. A 'User account added' notification is visible at the bottom left.

Google Cloud mazenet-001 cloud sql

All Instances > snehacollegesql

Explorer

- Databases 1
 - college_db (Default)
 - Tables 1
 - students
 - Views 0
 - Events 0
 - Functions 0
 - Procedures 0
 - information_schema
 - mysql
 - performance_schema
 - sys

Queries 0 Preview

<> Untitled query

Run Save Format Clear

Syntax error at or near 'GRANT'

```
1 GRANT SELECT ON college_db.* TO 'readonly_user'@'%';
2 FLUSH PRIVILEGES;
3
4
5
6
```

Results

Execution time: 52.6 ms

Statement executed successfully

User account added

Open editor

Google Cloud mazenet-001 cloud sql

All Instances > snehacollegesql

Explorer

- Databases 1
 - college_db (Default)
 - Tables 1
 - students
 - Views 0
 - Events 0
 - Functions 0
 - Procedures 0
 - information_schema
 - mysql
 - performance_schema
 - sys

Queries 0 Preview

<> Untitled query

Run Save Format Clear

Syntax error at or near 'GRANTS'

```
1 SHOW GRANTS FOR 'readonly_user'@'%';
2
3
4
5
```

Results

Execution time: 0.6 ms

Grants for readonly_user@%

GRANT USAGE ON *.* TO 'readonly_user'@'%'
GRANT SELECT ON 'college_db'.* TO 'readonly_user'@'%'

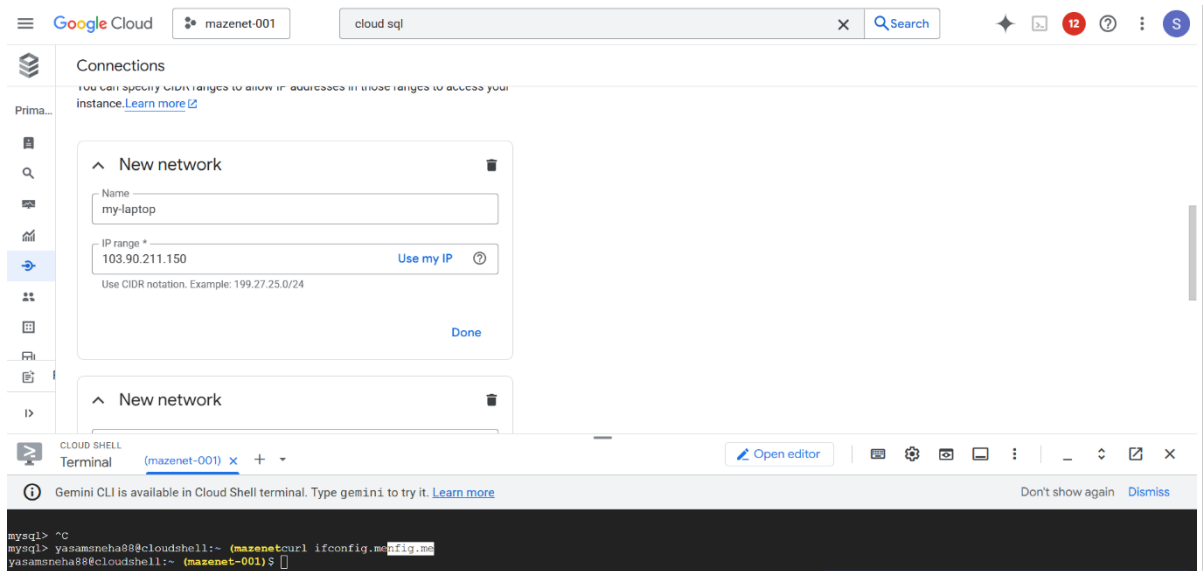
Rows per page: 20 1 - 2 of 2

User account added

Open editor

9. Restrict Access by IP

- In Cloud SQL → Connections → Authorized Networks, add your IP.
- This ensures only specific IPs can connect.



10. Test Security



Task2: real-time chat application backend.

1. Enable Firestore (Native Mode)

Using Google Cloud Console

Go to Google Cloud Console → Firestore

Click Create

Create Collection: chats

Firestore Data Model

chats (collection)

chatId (document)

sender: "sneha"

receiver: "rahul"

message: "Hi Rahul"

timestamp: <Firestore Timestamp>

The screenshot shows the Google Cloud Firestore console for a database named 'sneha'. The left sidebar contains navigation options: Database (Firestore studio, Security, Indexes, Import/Export, Disaster recovery, Time to live (TTL)), Insights (Usage, Query Insights, Monitoring, Key Visualizer, Release notes), and a terminal at the bottom. The main panel is in 'Panel view' and shows the path '/ > chats > 7fBDJIU6d8sq8KIVQ4Dd'. The 'chats' collection is selected, and a document with ID '7fBDJIU6d8sq8KIVQ4Dd' is highlighted. The document's content is displayed on the right: message: 'i am fine', receiver: 'sneha', sender: 'rahul', and timestamp: '22 December 2025 at 14:36...'. The URL bar shows the console path with a cloudshell parameter.

This screenshot shows the same Firestore console interface, but the path is '/ > chats > 2lwxyIPmV4FcMAddpvbt'. The document with ID '2lwxyIPmV4FcMAddpvbt' is selected, and its content is shown on the right: message: 'hi', receiver: 'ananya', sender: 'rahul', and timestamp: '22 December 2025 at 14:36...'. A black toast notification at the bottom center reads 'Created document 2lwxyIPmV4FcMAddpvbt'. The terminal at the bottom shows the '(mazenet-001)' session.

3. Insert At Least 10 Chat Messages

The screenshot shows the Google Cloud Firestore console for a database named 'sneha'. The 'Query builder' tab is active, showing a query for the 'chats' collection. The 'Query results' table displays 10 chat messages. A notification at the bottom indicates that a document was created.

Document ID	message	receiver	sender	timestamp
2lwxyIPmV4FcMAddpvbt	"hi"	"ananya"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
2V09qHkZGg3CWI8BRfaW	"how r u?"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
7fBDJlU6d8sq8KIVQ4Dd	"i am fine"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
8uvAxbNSXeV6nB5chfZ	"hi "	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
CXGBbI9V4gtGYvmtpgC	"hey"	"sneha"	"ananya"	22 December 2025 at 14:36:38 UTC+5:30
lBwzNQ1GuEcj4eIGgyiC	"hello"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30

4. Query Messages Between Two Users

Get all messages between sneha and rahul

The screenshot shows the Google Cloud Firestore console for a database named 'sneha'. The 'Query builder' tab is active, showing a query for the 'chats' collection with a filter for messages between 'sneha' and 'rahul'. The 'Query results' table displays 4 chat messages. A notification at the bottom indicates that a document was created.

Document ID	message	receiver	sender	timestamp
2V09qHkZGg3CWI8BRfaW	"how r u?"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
lp7QQLc17my74ywuek6r	"hi"	"ananya"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
rtiMHma36QJQb2NE4Lu1	"great"	"sneha"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
z0uwZ1gKBFWwEhwPUYMz	"hi rahul"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30

Firestore

Database

Firestore studio

Security

Indexes

Import/Export

Disaster recovery

Time to live (TTL)

Insights

Usage

Query Insights

Monitoring

Key Visualizer

Release notes

All databases > Database sneha

Panel view Query builder

Query scope

Collection

Collection *

/chats

Limit *

100

Selection

WHERE

Field

receiver

Operator

==

Value type

string

Value

rahul

Results Analysis

Query results

Document ID	message	receiver	sender	timestamp
2V09qHkZGg3CWl8BRfaW	"how r u?"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
ebHLhjK4bbMeZMdK7UYC	"hello"	"rahul"	"ananya"	22 December 2025 at 14:36:38 UTC+5:30
z0uwZ1gKBFWwEhwPUYMz	"hi rahul"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30

Rows per page: 50 1 - 3 of 3

CLOUD SHELL

Terminal

(mazenet-001)

Open editor

5. Sort Messages by Timestamp

Firestore

Database

Firestore studio

Security

Indexes

Import/Export

Disaster recovery

Time to live (TTL)

Insights

Usage

Query Insights

Monitoring

Key Visualizer

Release notes

All databases > Database sneha

Panel view Query builder

Query scope

Collection

Collection *

/chats

Limit *

100

Selection

ORDER BY

Field

timestamp

Order

ascending

Add to query

Results Analysis

Query results

Document ID	message	receiver	sender	timestamp
2lwxyIPmV4FcMAAddpvt	"hi"	"ananya"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
2V09qHkZGg3CWl8BRfaW	"how r u?"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
7fBDJlU6d8sq8KlVQ4Dd	"i am fine"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
8uvAxbNSXeV6nB5chfZ	"hi"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
CXGBbI9V4gtGYvmtpgC	"hey"	"sneha"	"ananya"	22 December 2025 at 14:36:38 UTC+5:30
lBwzNQ1GuEcj4elGgyiC	"hello"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30

CLOUD SHELL

Terminal

(mazenet-001)

Open editor

6.Firestore Security Rules

Firestore

Project Overview

Project shortcuts

Functions

Firestore Database

Product categories

Build

Run

Analytics

AI

Related development tools

Firestore Studio

Blaze

Pay as you go

Modify

mazenet-001

Cloud Firestore

Overview

Reads

349 total

Writes

122 total

Deletes

20 total

Graph

Dec 16

Dec 17

Dec 18

Dec 19

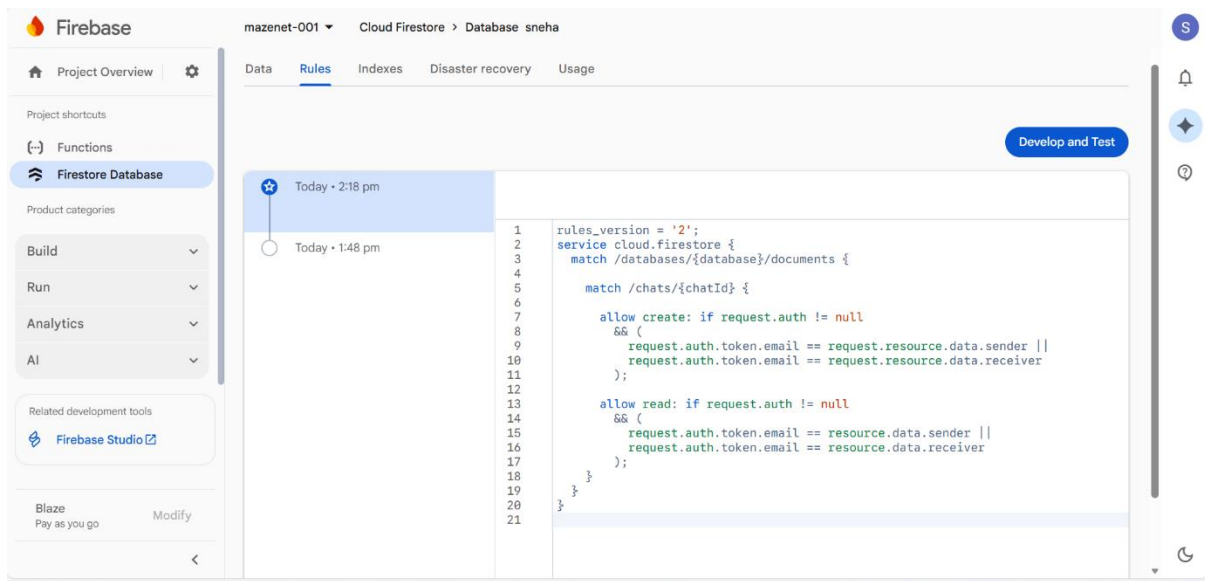
Dec 20

Dec 21

Databases

Name	Edition	Location	Mode	Scheduled backups
(default)	Standard	nam5	Native	

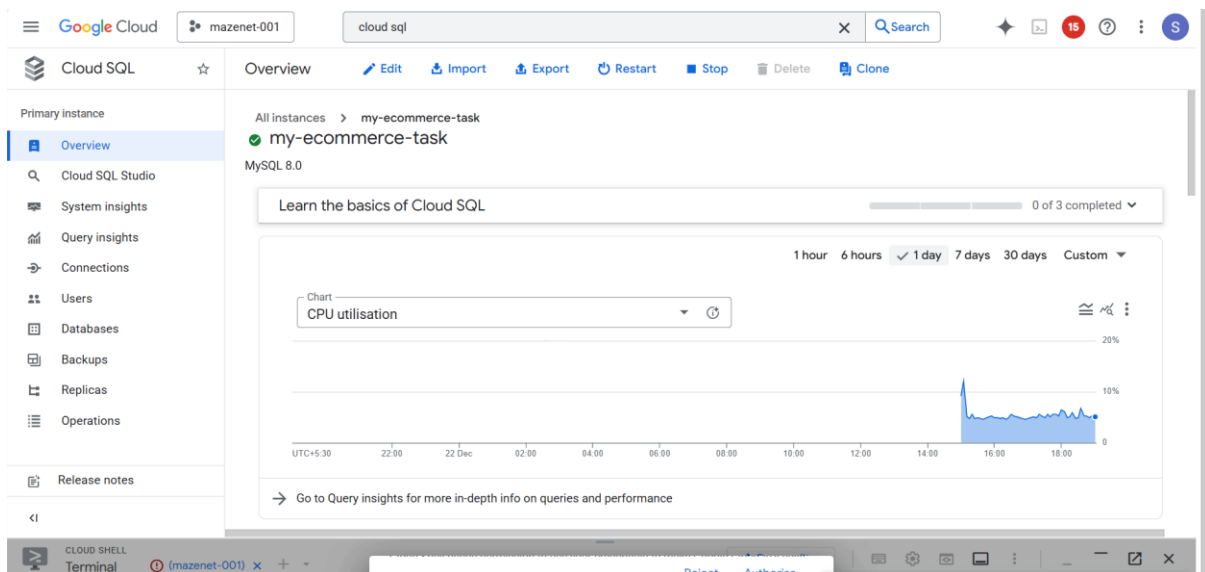
Then Publish



Task 3: E-commerce Platform Data Architecture

1. Cloud SQL for Orders and Payments

- **Why Cloud SQL:**
 - Relational database (MySQL/PostgreSQL) suitable for structured transactional data.
 - Supports ACID properties



Tables Creation:

Orders Table

The screenshot shows the Google Cloud SQL console interface. The left sidebar displays the 'Explorer' view with a tree structure of databases and schemas. The main panel shows a query editor with a 'Run' button. The query executed is:

```
1 CREATE TABLE Orders (  
2   order_id SERIAL PRIMARY KEY,  
3   user_id INT NOT NULL,  
4   product_id INT NOT NULL,  
5   quantity INT NOT NULL,  
6   order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
7 );  
8
```

The results section shows a green checkmark and the message 'Statement executed successfully'. The execution time is 32.9 ms. A notification at the bottom left states 'User account added'.

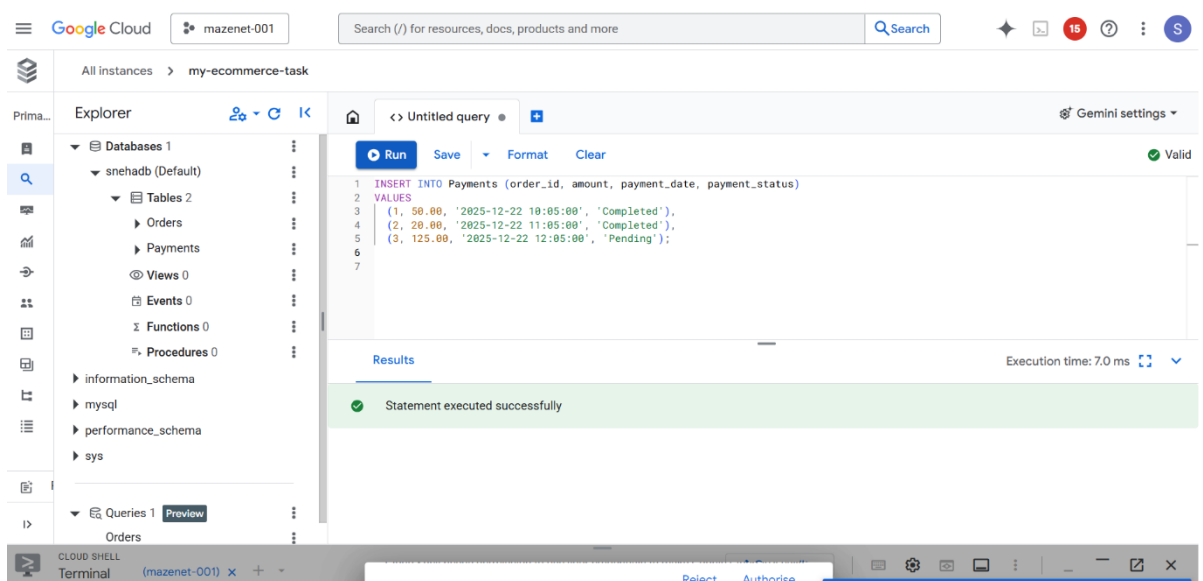
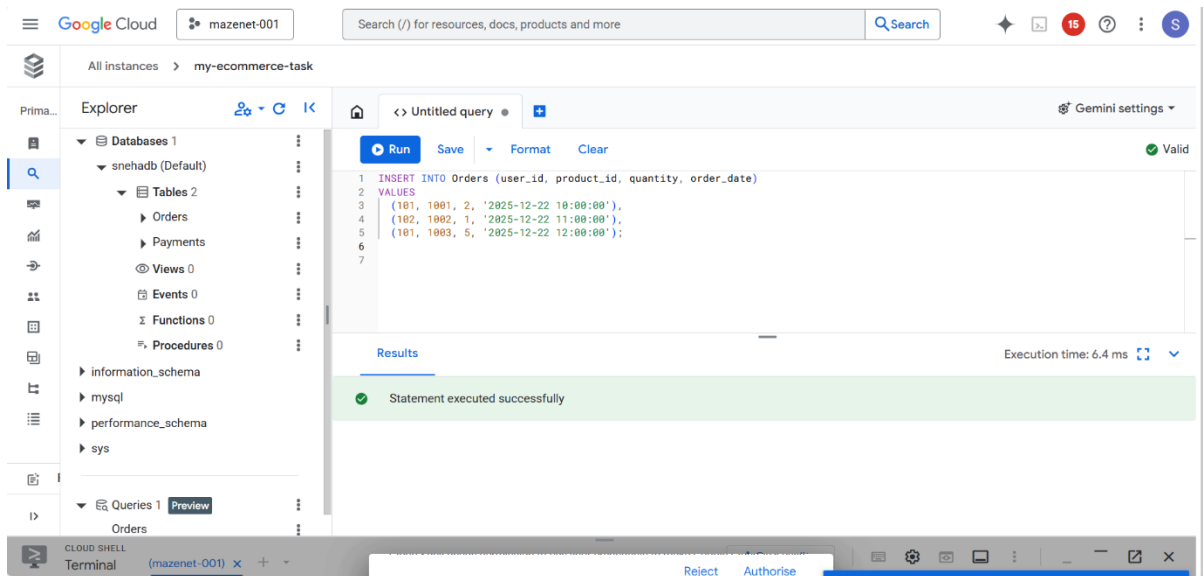
Payments Table

The screenshot shows the Google Cloud SQL console interface. The left sidebar displays the 'Explorer' view with a tree structure of databases and schemas. The main panel shows a query editor with a 'Run' button. The query executed is:

```
1 CREATE TABLE Payments (  
2   payment_id SERIAL PRIMARY KEY,  
3   order_id INT REFERENCES Orders(order_id),  
4   amount DECIMAL(10,2) NOT NULL,  
5   payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
6   payment_status VARCHAR(20)  
7 );  
8
```

The results section shows a green checkmark and the message 'Statement executed successfully'. The execution time is 32.6 ms. A notification at the bottom left states 'Query Orders saved'.

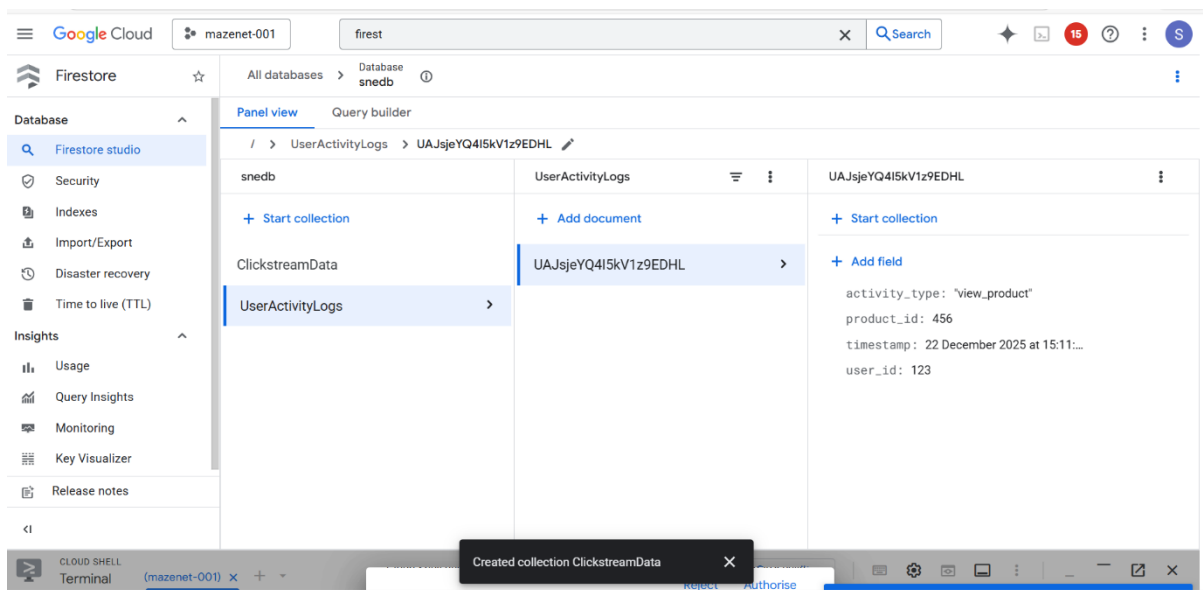
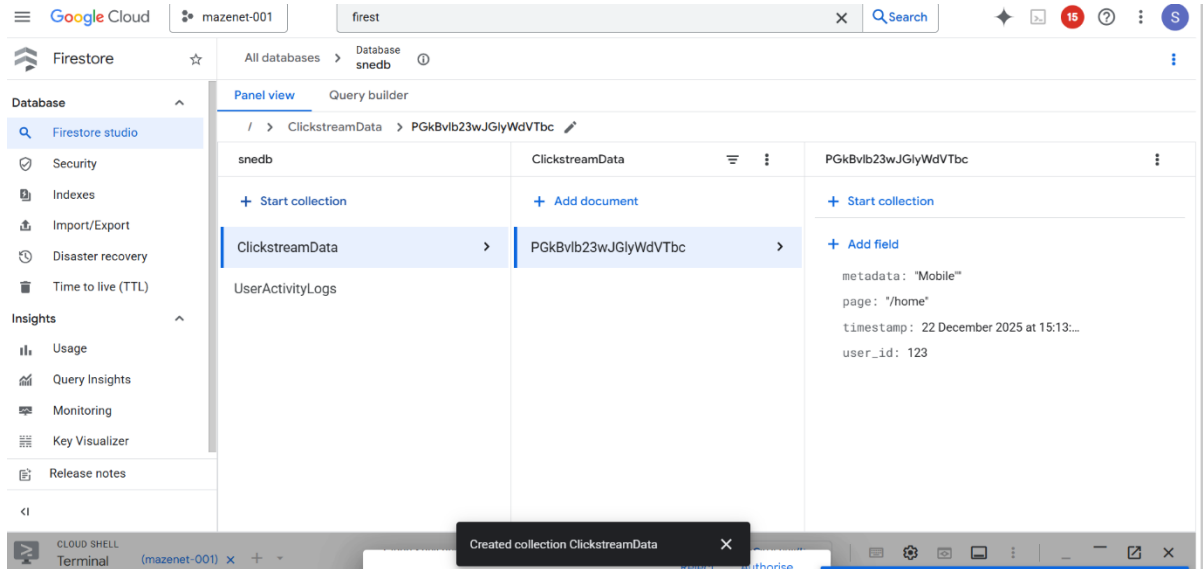
Insert the content



2. Firestore

- **Why NoSQL**
 - Handles high volume, semi-structured data.
 - Provides scalability and fast read/write.
 - Useful for analytics, not transactional integrity.
- Example Firestore Collection: user_activities
 - Document fields:

- user_id → string
- activity_type → string (e.g., click, page_view)
- timestamp → timestamp



3. Queries

SQL Query: Fetch total orders per user

```
SELECT user_id, COUNT(order_id) AS total_orders, SUM(total_price) AS total_spent
FROM Orders
GROUP BY user_id;
```

Google Cloud console showing a SQL query in Cloud Shell. The query is executed against the 'my-e-commerce-task' database. The query is:

```
1 SELECT user_id, COUNT(*) AS total_orders
2 FROM Orders
3 GROUP BY user_id
4 ORDER BY total_orders DESC;
```

The results show two rows:

user_id	total_orders
101	2
102	1

NoSQL Query: Fetch last 50 user activities

Google Cloud console showing a Firestore query in Cloud Shell. The query is configured to fetch the last 50 documents from the ClickstreamData collection, sorted by timestamp in descending order. The results show a single document with metadata, page, timestamp, and user_id.

Document ID	metadata	page	timestamp	user_id
PGKBv1b23wJGlyWdVTbc	'Mobile'	'/home'	22 December 2025 at 15:13:24 UTC+5:30	123

- User activity logs and clickstream data are stored in Firestore collections. Each document represents one log entry.
- Logs are checked by opening Firestore and in Data tab, selecting the ClickstreamData collection, viewing documents sorted by timestamp.
- The latest 50 logs are fetched using orderBy(timestamp) and limit(50)

