

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.

The screenshot shows the Google Cloud SQL Instances page. The instance 'snehacollegesql' is listed under MySQL 8.0. A chart titled 'CPU utilisation' shows usage over the last 24 hours, with a peak around 10% at approximately 11:00 UTC+5:30. A success message 'User account added' is displayed at the bottom.

2. Create Database

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

The screenshot shows the Google Cloud SQL Databases page. It lists several databases: college_db, information_schema, mysql, performance_schema, and sys. The 'college_db' database was created successfully, as indicated by the message 'User account added' at the bottom.

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 interface. On the left, the sidebar displays the database structure: Databases (college_db), Tables (students), Views, Events, Functions, Procedures, information_schema, mysql, performance_schema, and sys. A modal window at the bottom left says "User account added". In the main area, there's an "Untitled query" editor with the following SQL code:

```
1 SELECT * FROM students
2 WHERE marks > 75;
```

Below the editor, the "Results" section shows the output of the query:

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

Execution time: 1.1 ms. There are also "Export" and "Copy" buttons.

7. Count Students per Department

Query:

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

The screenshot shows the Google Cloud SQL interface. In the top navigation bar, it says "Google Cloud" and "mazenet-001". The main title is "cloud sql". On the left, there's a sidebar with icons for databases, tables, views, events, functions, procedures, information_schema, mysql, performance_schema, and sys. Below that is a search bar and a "Queries 0" section with a "Preview" button. The main area has tabs for "Explorer" and "Cloud SQL". Under "Explorer", there's a "Databases 1" section with "college_db (Default)" expanded, showing "Tables 1" with "students", "Views 0", "Events 0", "Functions 0", and "Procedures 0". There are also sections for "information_schema", "mysql", "performance_schema", and "sys". A modal window at the bottom left says "User account added". The central query editor has a "Run" button and a code block:

```

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

```

The results table shows:

department	student_count
Computer Science	2
Electronics	2
Mechanical	1

Execution time: 0.8 ms. There are "Export" and "Copy" buttons.

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 interface. The top navigation bar and sidebar are identical to the previous screenshot. The main area shows a query editor with a "Run" button and a code block:

```

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

```

A syntax error message "Syntax error at or near "USER'" is displayed next to the code. The results table shows:

Statement	Result
CREATE USER 'readonly_user'@'%' IDENTIFIED BY 'ReadOnly@123';	Statement executed successfully

Execution time: 51.2 ms. There are "Copy" and "Open editor" buttons.

Google Cloud mazenet-001 cloud sql

All instances > snehacollegesql

Explorer Run Save Format Clear

GRANT SELECT ON college_db.* TO 'readonly_user'@'%';
FLUSH PRIVILEGES;

Results Statement executed successfully

User account added

Execution time: 52.6 ms

This screenshot shows the Google Cloud SQL interface. In the left sidebar, the database 'college_db' is selected. The main area contains a query editor with the following SQL code:

```
GRANT SELECT ON college_db.* TO 'readonly_user'@'%';
FLUSH PRIVILEGES;
```

The results pane below the query editor shows a green checkmark and the message "Statement executed successfully". The status bar at the bottom right indicates an execution time of 52.6 ms.

Google Cloud mazenet-001 cloud sql

All instances > snehacollegesql

Explorer Run Save Format Clear

SHOW GRANTS FOR 'readonly_user'@'%';

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

User account added

Execution time: 0.6 ms

This screenshot shows the Google Cloud SQL interface. In the left sidebar, the database 'college_db' is selected. The main area contains a query editor with the following SQL code:

```
SHOW GRANTS FOR 'readonly_user'@'%';
```

The results pane below the query editor shows the grants for the user 'readonly_user'. The status bar at the bottom right indicates an execution time of 0.6 ms.

9. Restrict Access by IP

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

The screenshot shows the Google Cloud Platform interface with the search bar set to "cloud sql". On the left, there's a sidebar with various icons. The main area is titled "Connections" and contains a sub-section for "New network". It has fields for "Name" (set to "my-laptop") and "IP range" (set to "103.90.211.150"). A note says "Use CIDR notation. Example: 199.27.25.0/24". At the bottom of this section is a "Done" button. Below it, another "New network" section is partially visible. At the bottom of the page, there's a "CLOUD SHELL Terminal" tab with the identifier "(mazenet-001)". The terminal window shows a MySQL prompt: "mysql> ^C" followed by "mysql> yasamsneha88@cloudshell:~ (mazenetcurl ifconfig.me|ifconfig.me" and "yasamsneha88@cloudshell:~ (mazenet-001)\$". There are also some status messages and a "Gemini CLI is available in Cloud Shell terminal. Type gemini to try it." note.

10. Test Security

```
mysql> yasamsneha88@cloudshell:~ (mazenetcurl ifconfig.me|ifconfig.me
34.198.54.141yasamsneha88@cloudshell:~ (mazenet-001)$
```

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 the database 'sneha'. The left sidebar has sections for Database (Security, Indexes, Import/Export, Disaster recovery, Time to live (TTL)), Insights (Usage, Query Insights, Monitoring, Key Visualizer, Release notes), and Cloud Shell (Terminal). The main area shows a 'Panel view' of the 'chats' collection under the 'sneha' database. A specific document, '7fBDJlU6d8sq8KIVQ4Dd', is selected and expanded. Its fields are listed as:

- message: "i am fine"
- receiver: "sneha"
- sender: "rahul"
- timestamp: 22 December 2025 at 14:36:...

The screenshot shows the Google Cloud Firestore console for the database 'sneha'. The left sidebar is identical to the first screenshot. In the main area, a new document is being created in the 'chats' collection under '2lwxyIPmV4FcMAddpvbt'. The document's fields are listed as:

- message: "hi"
- receiver: "ananya"
- sender: "rahul"
- timestamp: 22 December 2025 at 14:36:...

A black toast notification at the bottom center says 'Created document 2lwxyIPmV4FcMAddpvbt'.

3. Insert At Least 10 Chat Messages

The screenshot shows the Google Cloud Firestore Query builder interface. The left sidebar has 'Firestore studio' selected under 'Database'. The main area shows a 'Query builder' tab with a 'Collection' dropdown set to '/chats' and a 'Limit' of 100. Below this is a 'Results' tab displaying a table of 10 documents. Each document has fields: Document ID, message, receiver, sender, and timestamp. The messages are: "hi", "how r u?", "i am fine", "hi", "hey", "hello", "ananya", "rahul", "sneha", and "rahul". The timestamp for all is 22 December 2025 at 14:36:38 UTC+5:30. A tooltip 'Created document 2lwxyIPmV4FcMAddpvbt' is visible over the first document.

Document ID	message	receiver	sender	timestamp
2lwxyIPmV4FcMAddpvbt	"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
7IBDJU6e8sq8KIVQ4Dd	"i am fine"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
8uvAxbsNSxeV6nB5chfZ	"hi"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
CXGBbI9V4gtIGYvmtpgC	"hey"	"sneha"	"ananya"	22 December 2025 at 14:36:38 UTC+5:30
IBwzNQ1GuEqj4elGgyIC	"hello"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
2lwxyIPmV4FcMAddpvbt	"ananya"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
2lwxyIPmV4FcMAddpvbt	"rahul"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
2lwxyIPmV4FcMAddpvbt	"sneha"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
2lwxyIPmV4FcMAddpvbt	"rahul"	"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 Query builder interface. The left sidebar has 'Firestore studio' selected under 'Database'. The main area shows a 'Query builder' tab with a 'Collection' dropdown set to '/chats'. Below this is a 'Selection' section with 'WHERE' and 'Field: sender' set to 'sneha'. The results table shows 4 documents where the sender is 'sneha' and the receiver is either 'rahul' or 'ananya'. The timestamp for all is 22 December 2025 at 14:36:38 UTC+5:30.

Document ID	message	receiver	sender	timestamp
2V09qHkZGg3CWl8BRfaW	"how r u?"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30
lp7QLc17m72ywuwekf6r	"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
z0uwZ1gKBFwwElhwPUYMz	"hi rahul"	"rahul"	"sneha"	22 December 2025 at 14:36:38 UTC+5:30

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 has tabs for Panel view and Query builder. In the Query builder tab, the Query scope is set to Collection, the Collection path is '/chats', the Limit is 100, the Selection is WHERE, the Field is receiver, the Operator is ==, the Value type is string, and the Value is rahul. Below this, the Results tab is selected, showing the query results:

Document ID	message	receiver	sender	timestamp
2V09qHkZGg3CWi8BRfaW	"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

At the bottom, there are buttons for Open editor, a toolbar, and navigation controls.

5. Sort Messages by Timestamp

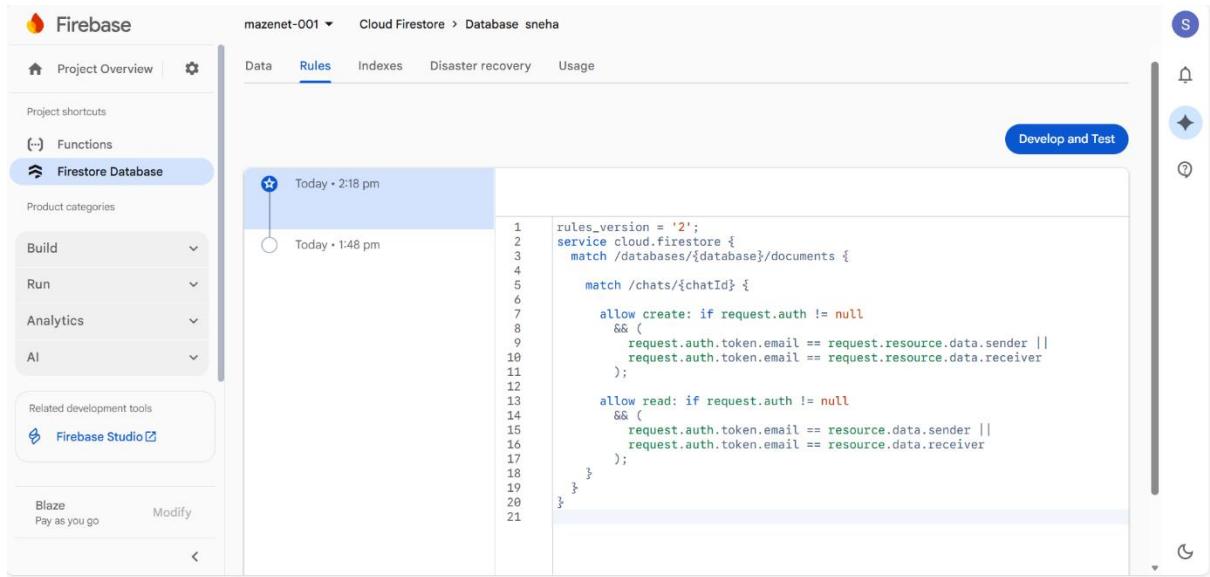
This screenshot shows the Google Cloud Firestore Query builder interface with a different configuration. The Selection dropdown now contains ORDER BY, the Field is timestamp, and the Order is ascending. The results table shows the same three messages as before, but they are listed in chronological order from oldest to newest based on timestamp.

Document ID	message	receiver	sender	timestamp
2lwxypmV4FcMAddpvbt	"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
7BDJUJU6d8sq8KVQ4Dd	"i am fine"	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
8uVxbnNSxv6nBScfZ	"hi "	"sneha"	"rahul"	22 December 2025 at 14:36:38 UTC+5:30
CXGBbl9V4gtlGYvmtpgC	"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

6. Firestore Security Rules

This screenshot shows the Google Cloud Firestore Overview page for the project 'mazenet-001'. The left sidebar includes Project Overview, Functions, Firestore Database (selected), Analytics, AI, and Blaze (Pay as you go). The main area displays the Cloud Firestore section with an Overview chart showing Reads (349 total), Writes (122 total), and Deletes (20 total) over the last 7 days. Below the chart, the Databases section lists the default database with details like Edition (Standard), Location (nam5), Mode (Native), and Scheduled backups (disabled).

Then Publish



The screenshot shows the Firebase console interface for managing Cloud Firestore rules. The left sidebar has 'Project Overview' selected under 'Build'. The main area is titled 'Cloud Firestore > Database sneha'. The 'Rules' tab is active. A code editor displays the following security rule:

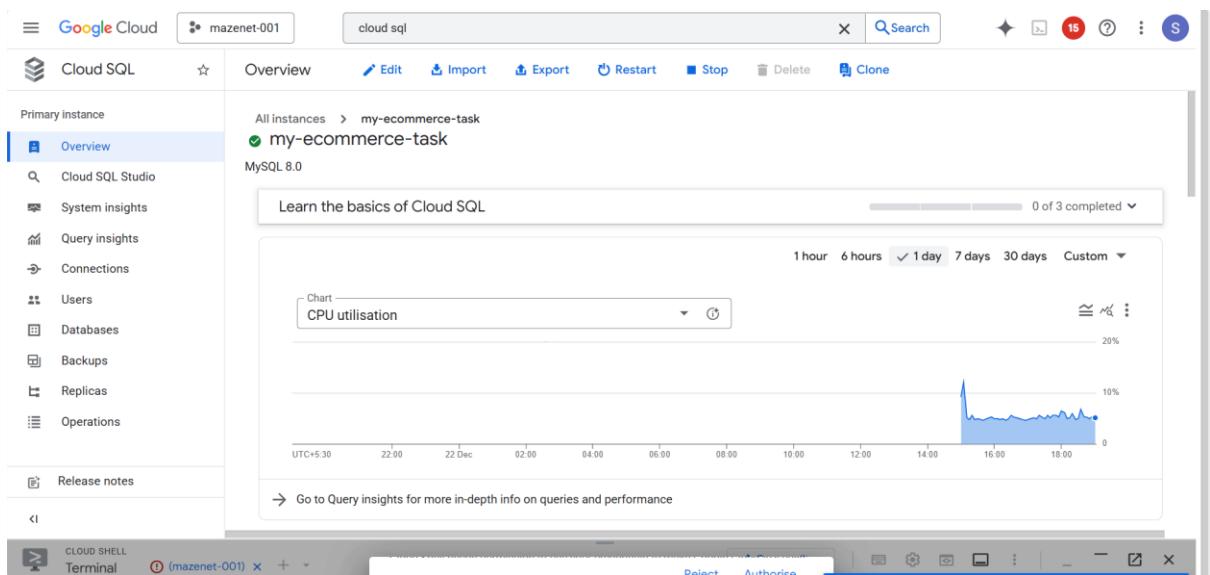
```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /chats/{chatId} {
      allow create: if request.auth != null
        && (
          request.auth.token.email == request.resource.data.sender ||
          request.auth.token.email == request.resource.data.receiver
        );
      allow read: if request.auth != null
        && (
          request.auth.token.email == resource.data.sender ||
          request.auth.token.email == resource.data.receiver
        );
    }
  }
}
```

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



The screenshot shows the Google Cloud Platform Cloud SQL interface. The left sidebar lists 'Overview', 'Cloud SQL Studio', 'System insights', 'Query insights', 'Connections', 'Users', 'Databases', 'Backups', 'Replicas', and 'Operations'. The main area shows an 'Overview' for the 'my-e-commerce-task' instance, which is a MySQL 8.0 instance. It displays a progress bar for learning Cloud SQL basics (0 of 3 completed), a chart of CPU utilisation over the last 24 hours, and a link to 'Go to Query insights for more in-depth info on queries and performance'. At the bottom, there is a 'CLOUD SHELL Terminal' tab with '(mazenet-001)' and a toolbar with 'Reject' and 'Authorise' buttons.

Tables Creation:

Orders Table

The screenshot shows the Google Cloud SQL interface for a database named 'my-e-commerce-task'. In the left sidebar, under 'Databases 1', the 'snehadb (Default)' database is selected, showing 'Tables 1'. A query editor window is open with the following SQL code:

```
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 );
```

The 'Run' button is highlighted. Below the code, the results pane shows a green success message: 'Statement executed successfully'. The execution time is listed as 32.9 ms.

Payments Table

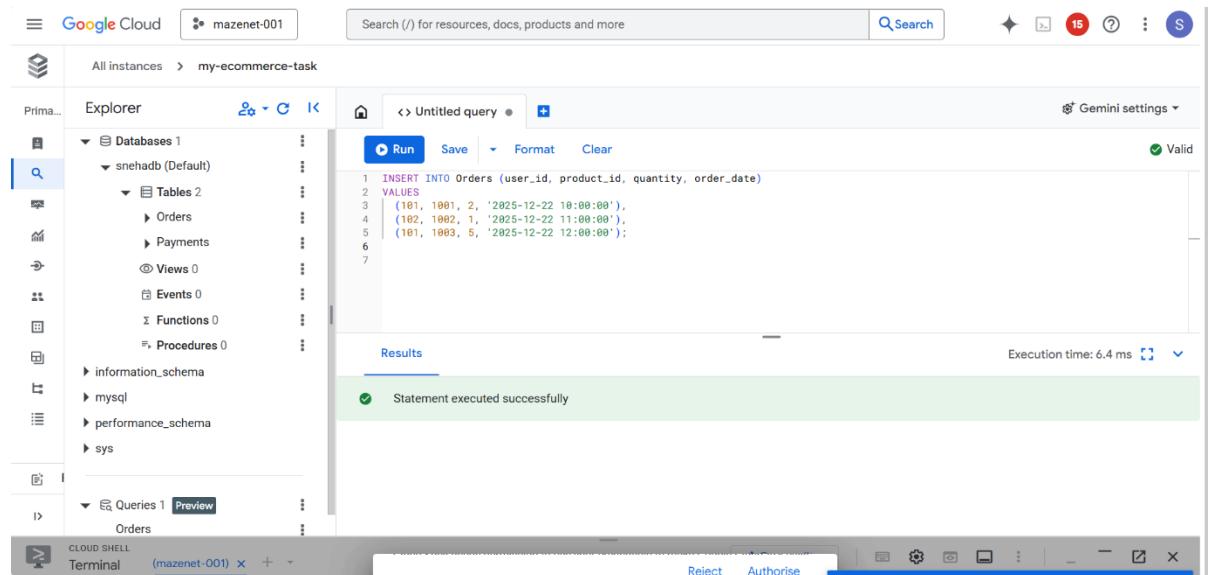
The screenshot shows the Google Cloud SQL interface for the same database. In the left sidebar, under 'Databases 1', the 'snehadb (Default)' database is selected, showing 'Tables 2'. A query editor window is open with the following SQL code:

```
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 );
```

The 'Run' button is highlighted. Below the code, the results pane shows a green success message: 'Statement executed successfully'. The execution time is listed as 32.6 ms.

A terminal window at the bottom shows a message: 'Query Orders saved'.

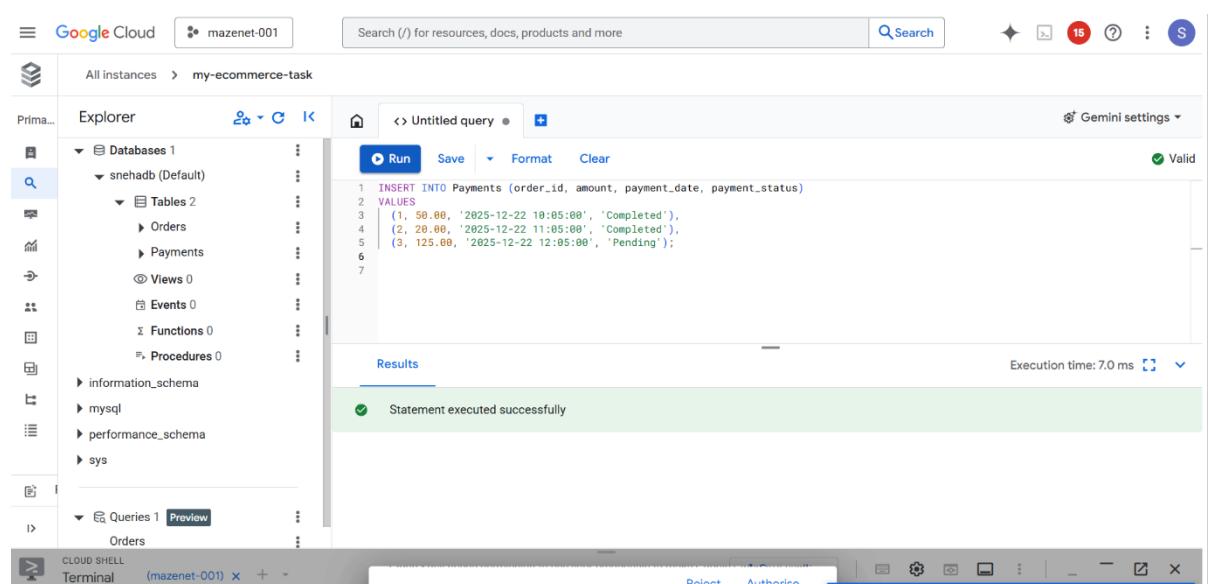
Insert the content



The screenshot shows the Google Cloud SQL Editor interface. The left sidebar displays the database structure under 'snehadb (Default)'. The main area shows an 'Untitled query' window with the following SQL code:

```
1 INSERT INTO Orders (user_id, product_id, quantity, order_date)
2 VALUES
3   (101, 1001, 2, '2025-12-22 10:00:00'),
4   (102, 1002, 1, '2025-12-22 11:00:00'),
5   (101, 1003, 5, '2025-12-22 12:00:00');
```

The results pane below shows a green success message: "Statement executed successfully". The execution time is listed as 6.4 ms.



The screenshot shows the Google Cloud SQL Editor interface. The left sidebar displays the database structure under 'snehadb (Default)'. The main area shows an 'Untitled query' window with the following SQL code:

```
1 INSERT INTO Payments (order_id, amount, payment_date, payment_status)
2 VALUES
3   (1, 50.00, '2025-12-22 10:05:00', 'Completed'),
4   (2, 20.00, '2025-12-22 11:05:00', 'Completed'),
5   (3, 125.00, '2025-12-22 12:05:00', 'Pending');
```

The results pane below shows a green success message: "Statement executed successfully". The execution time is listed as 7.0 ms.

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

Google Cloud Platform interface showing the creation of a collection named ClickstreamData in the snedb database. The collection contains a single document with the ID PGkBvlb23wJGlyWdVTbc. The document data is:

```

{
  "metadata": "Mobile",
  "page": "/home",
  "timestamp": "22 December 2025 at 15:13:...",
  "user_id": 123
}

```

Google Cloud Platform interface showing the creation of a collection named UserActivityLogs in the snedb database. The collection contains a single document with the ID UAJsjeYQ4i5kV1z9EDHL. The document data is:

```

{
  "activity_type": "view_product",
  "product_id": 456,
  "timestamp": "22 December 2025 at 15:11:...",
  "user_id": 123
}

```

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;

```

The screenshot shows the Google Cloud SQL Explorer interface. On the left, the sidebar lists databases, tables, and queries. In the main area, a query editor window titled 'Untitled query' contains the following SQL code:

```

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

```

Below the query editor, the results table shows two rows of data:

user_id	total_orders
101	2
102	1

The results table includes a footer with 'Execution time: 1.3 ms', 'Export' options, and a 'Rows per page' dropdown set to 20.

NoSQL Query: Fetch last 50 user activities

The screenshot shows the Google Cloud Firestore Query builder interface. The left sidebar has sections for Database, Security, Indexes, Import/Export, Disaster recovery, and Time to live (TTL). The main area shows the 'Query builder' tab with the following configuration:

- Collection:** /ClickstreamData
- Limit:** 50
- Selection:** WHERE user_id = 123

Below the builder, the 'Results' tab displays one document from the ClickstreamData collection:

Document ID	metadata	page	timestamp	user_id
PGkBvlb23wJGlyWdVTbc	"Mobile"	"/home"	22 December 2025 at 15:13:24 UTC+5:30	123

The results table includes a footer with 'Rows per page' dropdown set to 50.

- 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)

