

MANISHA GCP HANDSON TASKS

TASK 1 : SYNOPSIS

Student Management System using Google Cloud SQ

1.Introduction

A Student Management System is used to store, manage, and analyze student academic data in an organized and secure manner.

In this project, Google Cloud SQL is used to create a relational database that stores student details such as student ID, name, department, and marks.

Cloud SQL provides a fully managed database service with built-in security, scalability, and reliability.

2.Objective of the Project

The objectives of this project are:

- To create a Cloud SQL (MySQL/PostgreSQL) instance
- To create a database named college_db
- To design and create a students table
- To insert student records into the database
- To perform SQL queries for data analysis
- To secure the database using user permissions and network restrictions

3.Tools and Technologies Used

- Google Cloud Platform (GCP)
- Cloud SQL (MySQL)
- SQL
- GCP Networking and Security

4.Cloud SQL Instance Creation

A **Cloud SQL instance** is created using MySQL (or PostgreSQL) through the Google Cloud Console.

The instance is configured with appropriate region and credentials.

Once created, the instance acts as the backend database server for the application.

Google Cloud mazenet-001 Search (/) for resources, docs, products, and more

Cloud SQL Overview Edit Import Export Restart Stop Delete Clone

Primary instance All instances > manisha-college-mysql manisha-college-mysql MySQL 8.0

Learn the basics of Cloud SQL 0 of 3 completed

Chart CPU utilization 1 hour 6 hours 1 day 7 days 30 days Custom

20% 10% 0%

UTC+5:30 8:00 PM 8:00 PM 10:00 PM 10:00 PM Dec 22 2:00 AM 4:00 AM 8:00 AM 10:00 AM 12:00 PM 2:00 PM

Go to Query insights for more in-depth info on queries and performance

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

5.Database Creation

Inside the Cloud SQL instance, a database named **college_db** is created. This database is used to store all student-related data.

Google Cloud mazenet-001 Search (/) for resources, docs, products, and more

Cloud SQL Databases

Primary instance All instances > manisha-college-mysql manisha-college-mysql MySQL 8.0

+ Create database

Name ↑	Collation	Character set	Type	⋮
college_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	⋮

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

6.Table Creation – Students Table

A table named **students** is created inside the `college_db` database.

Table Structure:

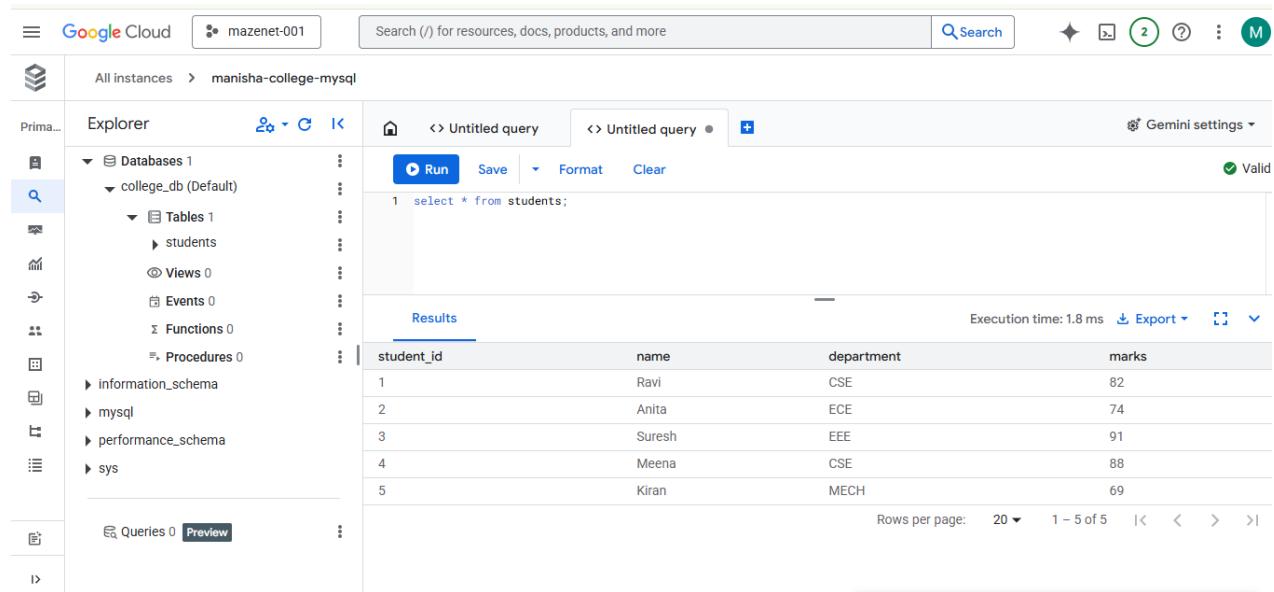
- **student_id** – Primary Key used to uniquely identify each student
- **name** – Stores the student's name
- **department** – Stores the department name
- **marks** – Stores the student's marks

This table structure ensures **data integrity and uniqueness**.

7.Data Insertion

At least **five student records** are inserted into the `students` table to simulate real academic data.

The inserted records include students from different departments with different marks.



The screenshot shows the Google Cloud MySQL interface. On the left, the sidebar lists databases, tables, and other schema objects. The main area shows an 'Untitled query' window with the following SQL code:

```
1 select * from students;
```

The 'Results' section displays the data from the 'students' table:

student_id	name	department	marks
1	Ravi	CSE	82
2	Anita	ECE	74
3	Suresh	EEE	91
4	Meena	CSE	88
5	Kiran	MECH	69

Execution time: 1.8 ms

8.SQL Queries Performed

Fetch Students with Marks Greater Than 75

This operation is used to identify students with high academic performance.

The screenshot shows the Google Cloud MySQL Explorer interface. On the left, the sidebar lists databases, tables, and other schema components. The main area displays an SQL query in the editor:

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

The results table shows three rows of student data:

student_id	name	department	marks
1	Ravi	CSE	82
3	Suresh	EEE	91
4	Meena	CSE	88

Execution time: 1.1 ms

Count Students per Department

This operation is used to analyze the number of students in each department.

The screenshot shows the Google Cloud MySQL Explorer interface. The sidebar lists databases, tables, and other schema components. The main area displays an SQL query in the editor:

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

The results table shows the count of students per department:

department	total_students
CSE	2
ECE	1
EEE	1
MECH	1

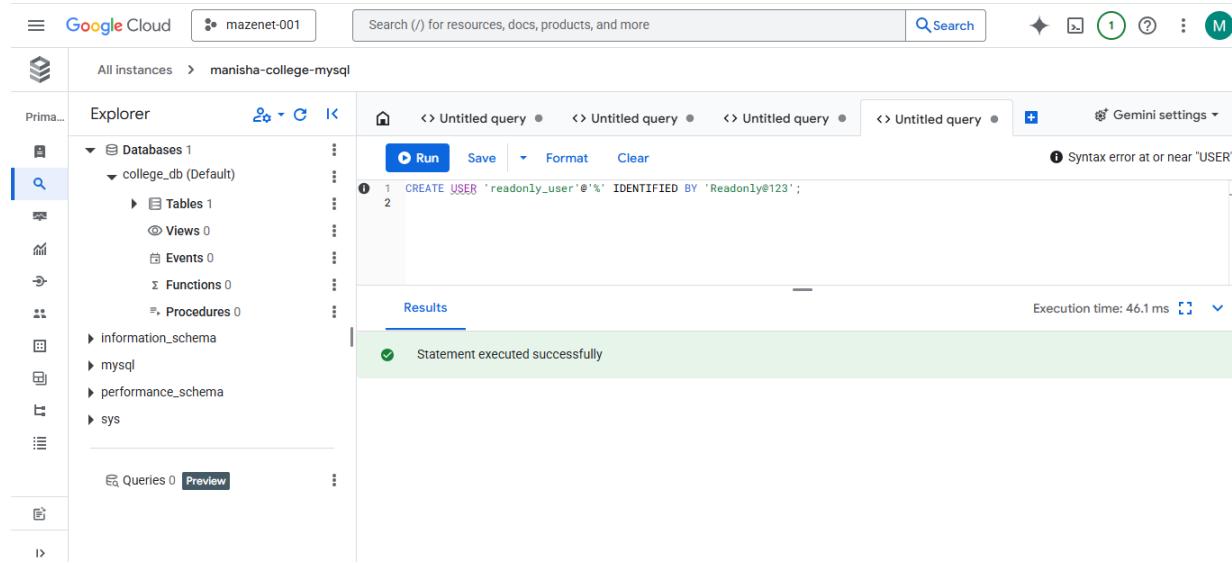
Execution time: 1.5 ms

9. Database Security Implementation

Security is an important part of database management.
The following security measures are implemented:

Read-Only User Creation

A separate database user is created with **read-only permissions**. This user can view student data but cannot modify or delete records, ensuring data safety.



The screenshot shows the Google Cloud SQL Explorer interface. The left sidebar lists databases, tables, views, events, functions, and procedures. The main area shows a query editor with the following SQL code:

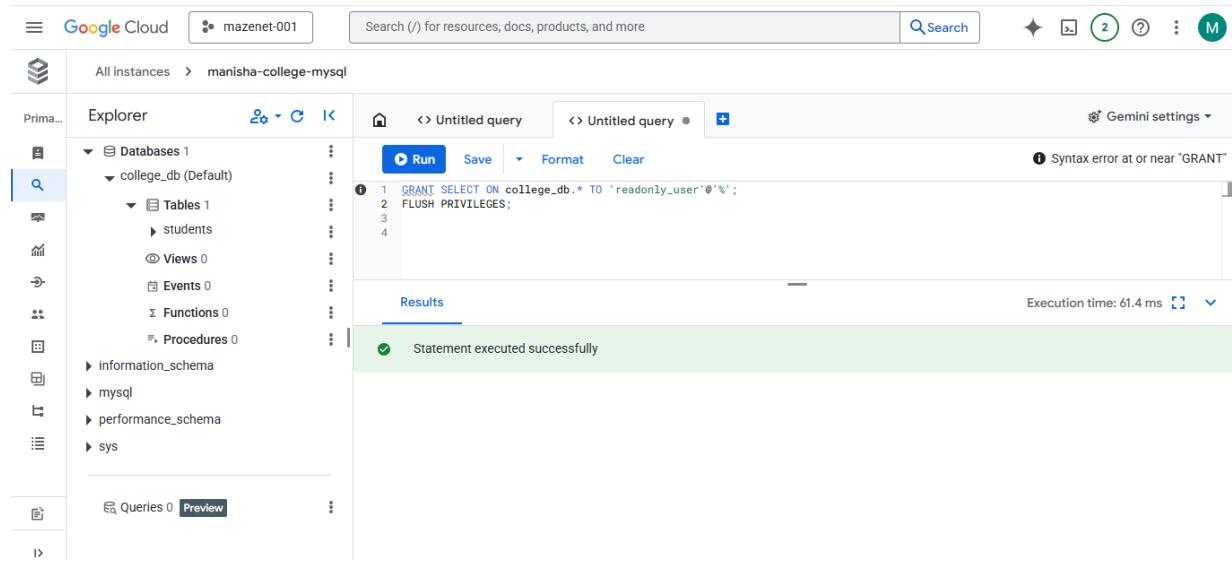
```
CREATE USER 'readonly_user'@'%' IDENTIFIED BY 'Readonly@123';
```

The results pane shows a green success message: "Statement executed successfully".

Network Access Restriction

Database access is restricted to a **specific authorized IP address** using Cloud SQL networking settings.

Only systems from the allowed IP can connect to the database.



The screenshot shows the Google Cloud SQL Explorer interface. The left sidebar lists databases, tables, views, events, functions, and procedures. The main area shows a query editor with the following SQL code:

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

The results pane shows a green success message: "Statement executed successfully".

All instances > manisha-college-mysql
manisha-college-mysql

MySQL 8.0

Summary Networking Security Connectivity Tests

Networking

Connection name	mazenet-001:asia-south2:manisha-college-mysql
Private IP connectivity	Disabled
Public IP connectivity	Enabled
Public IP address	34.131.188.55

Security

Authorized networks	1 network
my-laptop : 123.201.165.89	
Google Cloud services authorization	Disabled
App Engine authorization	Enabled

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

[Test Connection from GCE](#) New
[Docs: Connection Overview](#)

Results

The Student Management System successfully stores and manages student records using Cloud SQL.

SQL queries return accurate results, and security mechanisms prevent unauthorized access to the database.

11 .Conclusion

This project demonstrates the effective use of **Google Cloud SQL** for building a Student Management System.

It covers database creation, table design, data insertion, query execution, and database security.

The system is reliable, secure, and suitable for real-world academic data manage

TASK2: SYNOPSIS

Real-Time Chat Application Backend using Google Cloud Firestore

1.Introduction

A **Real-Time Chat Application** allows users to exchange messages instantly. In this project, **Google Cloud Firestore** is used as the backend database to store and manage chat messages in real time. Firestore is a NoSQL, document-based

database that provides real-time data synchronization, scalability, and low latency, making it suitable for chat applications.

2.Objective of the Project

The objectives of this project are:

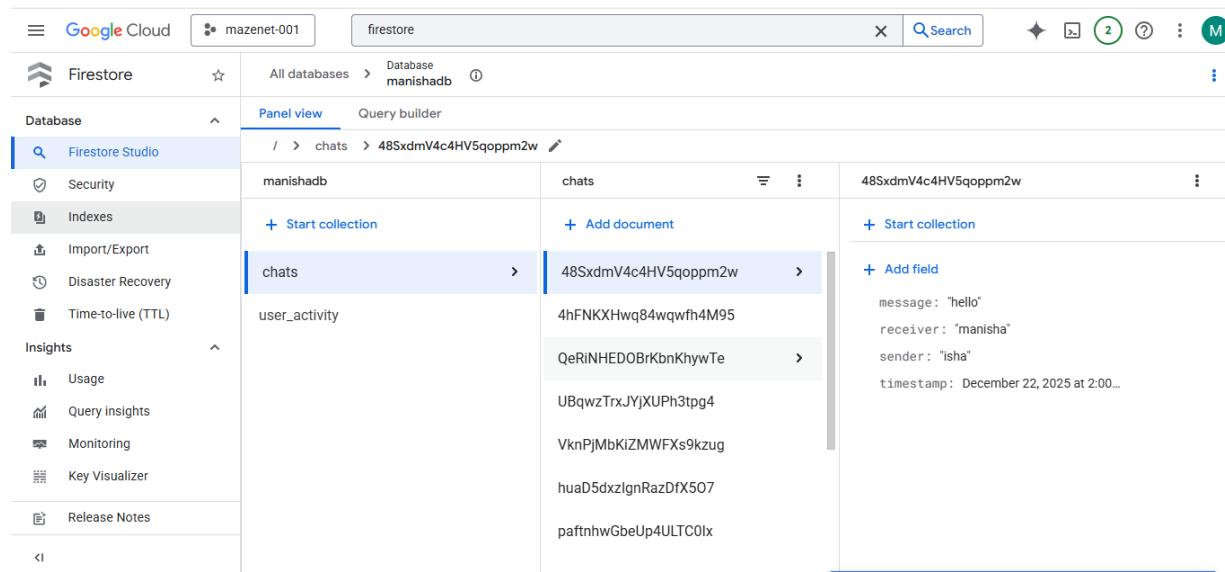
- To enable and configure **Firestore in Native mode**
- To design a data model for storing chat messages
- To insert multiple chat messages into Firestore
- To retrieve and sort chat messages between users
- To implement **security rules** to protect user data

3.Tools and Technologies Used

- **Google Cloud Platform (GCP)**
- **Cloud Firestore (Native mode)**
- **NoSQL Data Model**
- **Firebase Security Rules**

4.Firestore Setup

Firestore is enabled in **Native mode** through the Google Cloud Console. Native mode supports real-time updates and is suitable for mobile and web applications such as chat systems.



The screenshot shows the Google Cloud Firestore interface. The left sidebar includes sections for Database, Firestore Studio, 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 database named 'manishadb' under 'All databases'. A 'chats' collection is selected, showing documents with IDs like '48SxdmV4c4HV5qoppm2w', '4hFNKXHwq84wqwfh4M95', 'QeRINHEDOBkbnKhywTe', 'UBqwzTrxJYjXUPh3tpg4', 'VknPjMbKizMWFXs9kzug', 'huaD5dxzIggnRazDfx507', and 'paftnhwGbeUp4ULTC0lx'. Each document contains fields: 'message' (values: "hello", "hi"), 'receiver' (values: "manisha", "isha"), 'sender' (values: "isha", "manisha"), and 'timestamp' (value: December 22, 2025 at 2:00...).

5.Data Modeling – Chats Collection

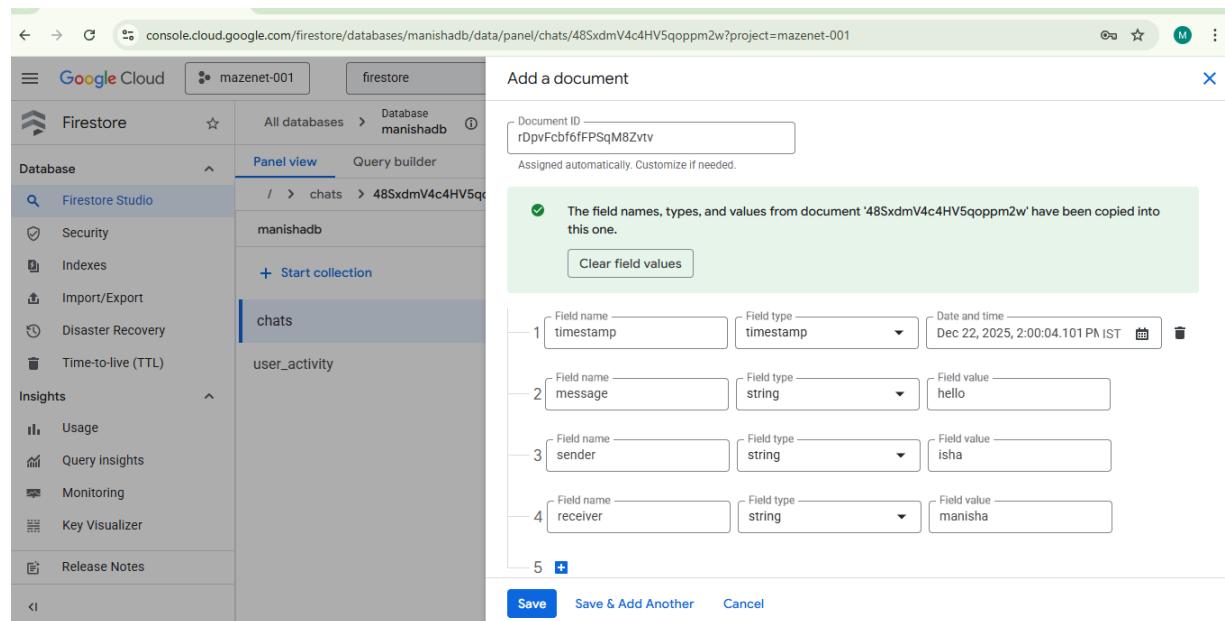
A collection named **chats** is created in Firestore.

Each document in the collection represents **one chat message**.

Each chat document stores the following fields:

- **sender** – User who sends the message
- **receiver** – User who receives the message
- **message** – Text content of the chat
- **timestamp** – Time when the message was sent

This design allows efficient storage and retrieval of chat data.



The screenshot shows the Google Cloud Firestore console interface. On the left, the sidebar includes 'Google Cloud' and 'mazenet-001' project dropdowns, followed by 'firestore'. Under 'Database', 'Firestore Studio' is selected. The main area shows a tree view with 'Database' expanded, showing 'manishadb' and 'chats'. A sub-collection 'user_activity' is also visible. On the right, a modal window titled 'Add a document' is open. It shows a 'Document ID' field with the value 'rDpvFcblfFPSSqM8Zvtv'. Below it, a note says 'Assigned automatically. Customize if needed.' A green box contains the message 'The field names, types, and values from document '48SxdmV4c4HV5qoppm2w' have been copied into this one.' A 'Clear field values' button is present. Five fields are defined: 1. Field name: timestamp, Field type: timestamp, Date and time: Dec 22, 2025, 2:00:04.101 PM IST. 2. Field name: message, Field type: string, Field value: hello. 3. Field name: sender, Field type: string, Field value: isha. 4. Field name: receiver, Field type: string, Field value: manisha. A 'Save' button at the bottom is highlighted in blue.

6.Data Insertion

At least **10 chat message documents** are inserted into the `chats` collection. Messages are stored between two users with different timestamps to simulate a real conversation.

The screenshot shows the Google Cloud Firestore interface. On the left, the sidebar includes sections for Database (Security, Indexes, Import/Export, Disaster Recovery, TTL), Insights (Usage, Query insights, Monitoring, Key Visualizer), and Release Notes. The main area shows a database named "manishadb" with a "chats" collection. A specific document in the "chats" collection is selected, showing fields: message ("hello"), receiver ("manisha"), sender ("isha"), and timestamp (December 22, 2025 at 2:00...). Other documents in the "chats" collection include 4hFNKXHwq84wqwfh4M95, QeRiNHEDOBkbnKhywTe, UBqwzTrxJYjXUPh3tpg4, VknPjMbKiZMWFXs9kzug, huaD5dxzIgnRazDfX50l, and paftnhwGbeUp4ULTC0lx.

7.Querying Chat Messages

Firestore queries are used to:

- Retrieve all chat messages exchanged between two users
- Sort messages based on the **timestamp** field

This enables the chat messages to be displayed in the correct conversation order.

Google Cloud mazenet-001 firestore

Firestore Database manishadb

Panel view Query builder

Selection WHERE Field sender Operator == Value type string Value isha

Selection WHERE Field receiver Operator == Value type string Value manisha

Add to query

Results Analysis

Query results

Document ID	message	receiver	sender	timestamp
48SxdmV4c4HV5qoppm2w	"hello"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30
4hFNKXHwq84wqwfh4M95	"what are you doing?"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30
UBqwzTrxJYjXUPh3tpg4	"okay good"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30
qTvDrt0z6X1u1dWAvJRS	"I'm fine and you ?"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30

Rows per page: 50 1 – 4 of 4 < >

Google Cloud mazenet-001 firestore

Firestore Database manishadb

Panel view Query builder

Selection WHERE Field sender Operator == Value type string Value isha

Selection WHERE Field receiver Operator == Value type string Value manisha

Add to query

Results Analysis

Query results

Document ID	message	receiver	sender	timestamp
48SxdmV4c4HV5qoppm2w	"hello"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30
4hFNKXHwq84wqwfh4M95	"what are you doing?"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30
UBqwzTrxJYjXUPh3tpg4	"okay good"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30
qTvDrt0z6X1u1dWAvJRS	"I'm fine and you ?"	"manisha"	"isha"	December 22, 2025 at 2:00:04 PM UTC+5:30

Rows per page: 50 1 – 4 of 4 < >

Google Cloud Platform - mazenet-001 | firestore

Firestore Database manishadb

Panel view Query builder

Limit: 100

Selection: ORDER BY timestamp

Order: ascending

Results Analysis

Query results

Document ID	message	receiver	sender	timestamp
48SxdmV4c4HV5qoppm2w	"hello"	"manisha"	"ishaa"	December 22, 2025 at 2:00:04 PM UTC+5:30
4hFNOKHwq84wqwfh4M95	"what are you doing?"	"manisha"	"ishaa"	December 22, 2025 at 2:00:04 PM UTC+5:30
QeRINHEDOBrKbnKhywTe	"hi"	"ishaa"	"manisha"	December 22, 2025 at 2:00:04 PM UTC+5:30
UBBqwzTrxJYjXUPh3tpg4	"okay good"	"manisha"	"ishaa"	December 22, 2025 at 2:00:04 PM UTC+5:30
VknPjmBkIZMWFxs9kzug	"working on tasks"	"ishaa"	"manisha"	December 22, 2025 at 2:00:04 PM UTC+5:30
huaD5dxzIgRazDfx507	"I'm fine "	"ishaa"	"manisha"	December 22, 2025 at 2:00:04 PM UTC+5:30

8. Security Rules Implementation

To protect chat data, **Firestore security rules** are implemented with the following conditions:

- Only **authenticated users** are allowed to read and write data
- Users can **read only their own chat messages** (where they are sender or receiver)

These rules prevent unauthorized access and ensure data privacy.

console.firebaseio.google.com/u/0/project/mazenet-001/firestore/databases/manishadb/security/rules

Firebase Project: mazenet-001 | Cloud Firestore > Database manishadb

Published changes can take up to a minute to propagate

Rules

```

rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /chats/{chatId} {
      // Only authenticated users can access their chats
      allow read, write: if request.auth != null
        && (
          request.auth.uid == resource.data.sender ||
          request.auth.uid == resource.data.receiver
        );
    }
  }
}

```

9.Results

The real-time chat backend successfully stores chat messages, retrieves conversations between users, and displays messages in chronological order. Security rules ensure that only authorized users can access their chat data.

Conclusion

This project demonstrates the use of **Google Cloud Firestore** to build a real-time chat application backend.

It covers Firestore setup, NoSQL data modeling, querying, and security rule implementation.

The system is scalable, secure, and suitable for real-time communication applications

TASK 3: SYNOPSIS

E-Commerce Platform Design using Cloud SQL and Firestore

1.Introduction

An **E-Commerce Platform** handles different types of data such as customer orders, payments, and user activity logs.

In this project, a **hybrid database architecture** is designed using **Google Cloud SQL** and **Firestore (NoSQL)** to efficiently manage both transactional and analytical data.

The design follows real-world industry practices by selecting the **right database for the right type of data**.

2.Objective of the Project

The objectives of this project are:

- To design an e-commerce backend using appropriate GCP services
- To store **orders and payments** using a relational database
- To store **user activity logs and clickstream data** using a NoSQL database
- To perform SQL and NoSQL queries for data analysis
- To justify database choices based on real-world requirements

3. Tools and Technologies Used

- Google Cloud Platform (GCP)
- Cloud SQL (MySQL / PostgreSQL)
- Cloud Firestore (NoSQL)
- SQL and NoSQL querying concepts

4. Cloud SQL for Orders and Payments

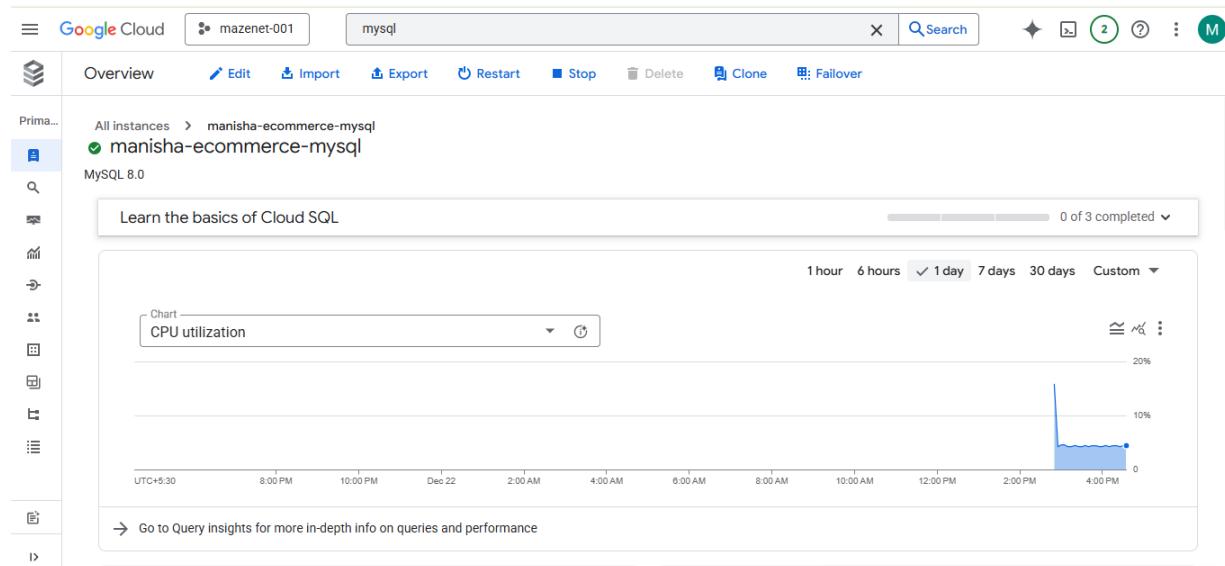
Purpose

Cloud SQL is used to store **orders and payments**, which are critical transactional data.

Data Stored

- Order details such as order ID, user ID, order amount, and order date
- Payment details such as payment ID, order ID, payment status, and payment date

Cloud SQL provides **ACID compliance**, ensuring data consistency and accuracy during financial transactions.



Google Cloud mazenet-001 mysql

Databases

All instances > manisha-ecommerce-mysql

manisha-ecommerce-mysql

MySQL 8.0

Create database

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

Google Cloud mazenet-001 mysql

All instances > manisha-ecommerce-mysql

Explorer

Databases 1

ecommerce_db (Default)

Tables 3

- orders
- payments
- users

Views 0

Events 0

Functions 0

Procedures 0

information_schema

mysql

performance_schema

sys

Run Save Format Clear

1 CREATE TABLE users (
2 | user_id INT PRIMARY KEY,
3 | name VARCHAR(100),
4 | email VARCHAR(100)
5);
6
7 INSERT INTO users (user_id, name, email) VALUES
8 (1, 'Manisha', 'manisha@gmail.com'),
9 (2, 'Rahul', 'rahul@gmail.com'),
10 (3, 'Isha', 'isha@gmail.com').

Results

user_id	name	email
1	Manisha	manisha@gmail.com
2	Rahul	rahul@gmail.com
3	Isha	isha@gmail.com

Execution time: 2.3 ms Export

Google Cloud mazenet-001 mysql Search

All instances > manisha-ecommerce-mysql

Explorer Databases 1 ecommerce_db (Default) Tables 3

```
Run Save Format Clear
```

```
1 select * from orders;
```

Results Execution time: 1.6 ms

order_id	user_id	order_amount	order_date
101	1	1200.00	2025-12-22 09:32:21
102	1	850.00	2025-12-22 09:32:21
103	2	499.00	2025-12-22 09:32:21
104	3	1599.00	2025-12-22 09:32:21
105	2	999.00	2025-12-22 09:32:21

Gemini settings Valid

Google Cloud mazenet-001 mysql Search

All instances > manisha-ecommerce-mysql

Explorer Databases 1 ecommerce_db (Default) Tables 3

```
Run Save Format Clear
```

```
1
2
3 select * from payments;
4
```

Results Execution time: 1.3 ms

payment_id	order_id	payment_status	payment_date
201	101	SUCCESS	2025-12-22 09:32:42
202	102	SUCCESS	2025-12-22 09:32:42
203	103	FAILED	2025-12-22 09:32:42
204	104	SUCCESS	2025-12-22 09:32:42
205	105	PENDING	2025-12-22 09:32:42

Rows per page: 20 1 – 5 of 5 < > >>

Google Cloud mazenet-001 mysql Search

All instances > manisha-ecommerce-mysql

Explorer Databases 1 ecommerce_db (Default) Tables 3

```
Run Save Format Clear
```

```
1 SELECT
2   user_id,
3   COUNT(order_id) AS total_orders
4 FROM orders
5 GROUP BY user_id;
6
```

Results Execution time: 1.2 ms

user_id	total_orders
1	2
2	2
3	1

Rows per page: 20 1 – 3 of 3 < > >>

5. Firestore for User Activity Logs and Clickstream Data

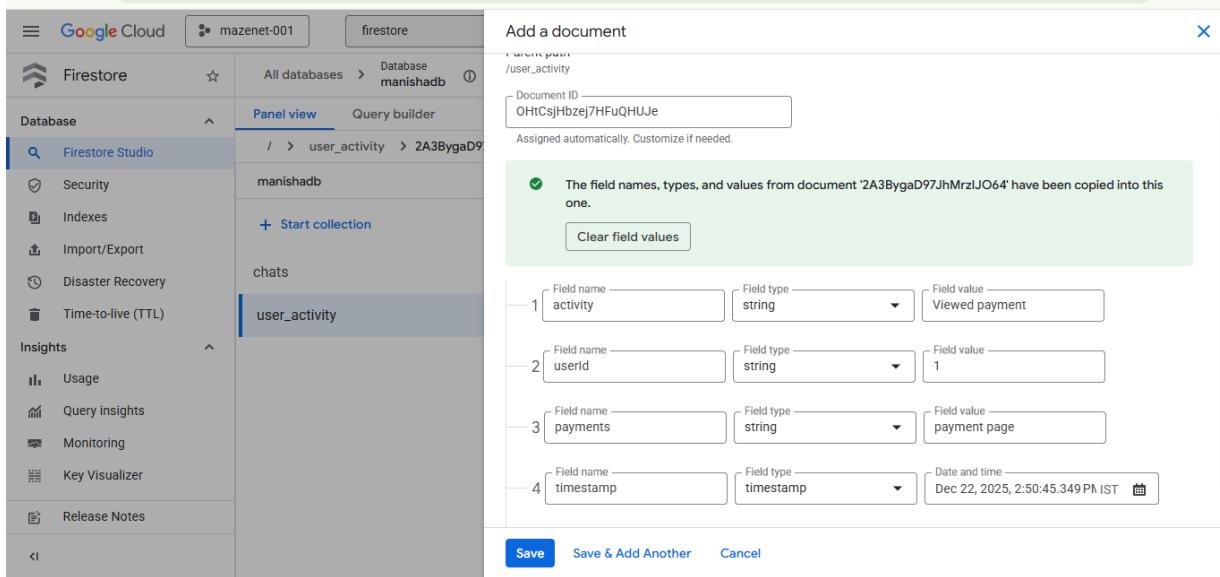
Purpose

Firestore is used to store **user activity logs** and **clickstream data**, which are generated in large volumes.

Data Stored

- User ID
- Activity (view product, add to cart, click, etc.)
- Payments
- Timestamp

Each document represents **one user action**, allowing efficient time-based queries.



The screenshot shows the Google Cloud Firestore interface. On the left, the sidebar includes sections for Firestore, Database (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: All databases > Database manishadb > Collection user_activity > Document 2A3BygaD97JhMrzIJO64. A modal window titled "Add a document" is open, prompting for a "Document ID" (set to OHTCsjHbzej7HFuQHUJe). It displays a success message: "The field names, types, and values from document '2A3BygaD97JhMrzIJO64' have been copied into this one." Below this, four fields are being defined: 1. Field name: activity, Field type: string, Field value: Viewed payment. 2. Field name: userId, Field type: string, Field value: 1. 3. Field name: payments, Field type: string, Field value: payment page. 4. Field name: timestamp, Field type: timestamp, Field value: Dec 22, 2025, 2:50:45.349 PM IST. At the bottom of the modal are "Save", "Save & Add Another", and "Cancel" buttons.

The screenshot shows the Google Cloud Firestore console. The left sidebar has sections for Firestore Studio (selected), Security, Indexes, Import/Export, Disaster Recovery, and Time-to-live (TTL). Below that is the Insights section with Usage, Query insights, Monitoring, Key Visualizer, and Release Notes. The main area shows a database named 'manishadb' under 'All databases'. Under 'manishadb', there is a 'user_activity' collection. A specific document in this collection is selected, with the ID '2A3BygaD97JhMrzIJO64'. The document contains the following fields:

activity	: "Viewed payment"
payments	: "payment page"
timestamp	: December 22, 2025 at 2:50...
userId	: "1"

6.SQL Query Implementation

A SQL query is implemented to **fetch the total number of orders placed by each user**.

This helps analyze customer behavior and purchase frequency using relational data.

The screenshot shows the Google Cloud MySQL Workbench interface. The left sidebar lists databases (ecommerce_db), tables (orders, payments, users), views, events, functions, procedures, and information_schema. The main area shows an SQL editor with the following query:

```
1 SELECT
2   | user_id,
3   | COUNT(order_id) AS total_orders
4 FROM orders
5 GROUP BY user_id;
```

The results pane shows the output of the query:

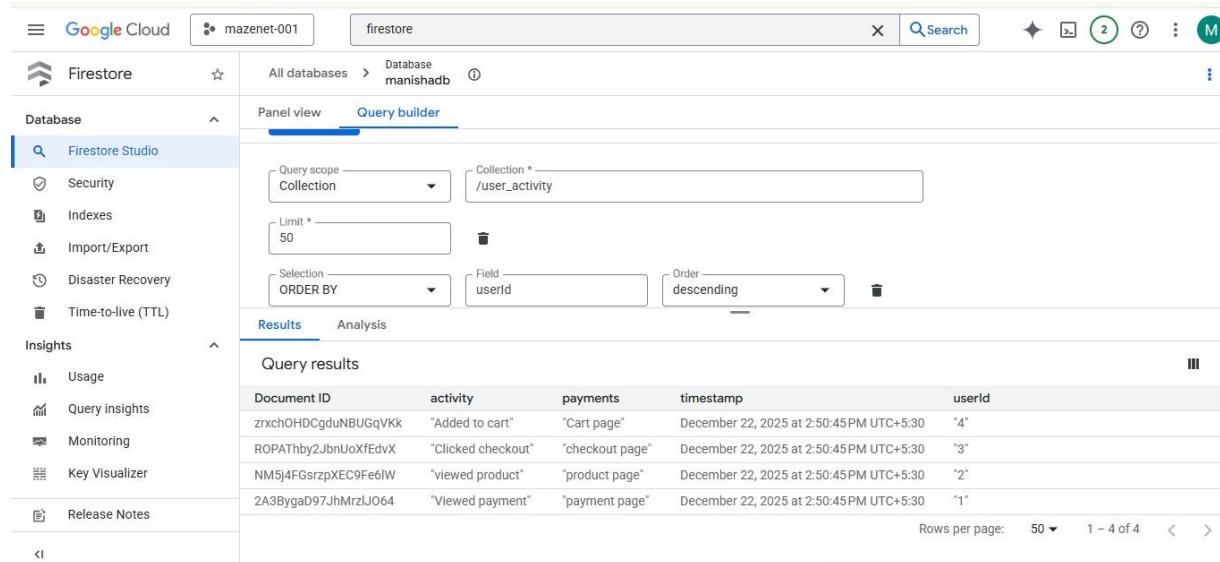
user_id	total_orders
1	2
2	2
3	1

Execution time: 1.2 ms

7.NoSQL Query Implementation

A Firestore query is implemented to **fetch the last 50 user activities** based on timestamp.

This query supports analytics such as user engagement tracking and behavior analysis.



The screenshot shows the Google Cloud Platform interface for Firestore. The left sidebar has sections for Database (selected), Security, Indexes, Import/Export, Disaster Recovery, Time-to-live (TTL), Insights (Usage, Query insights, Monitoring, Key Visualizer), and Release Notes. The main area is titled "Query builder" under "Database". It shows a query for the collection "/user_activity" with a limit of 50. The selection is set to ORDER BY "userId" in descending order. Below this, the "Results" tab is selected, showing a table with four columns: Document ID, activity, payments, timestamp, and userId. The table contains four rows of data. At the bottom right, there are pagination controls for "Rows per page: 50" and "1 - 4 of 4".

Document ID	activity	payments	timestamp	userId
zrchiOHDcgdubNUUqVKK	"Added to cart"	"Cart page"	December 22, 2025 at 2:50:45 PM UTC+5:30	"4"
ROPATHby2JbnUoXfEdvX	"Clicked checkout"	"checkout page"	December 22, 2025 at 2:50:45 PM UTC+5:30	"3"
NM5j4FGsrzpXEC9F6lw	"viewed product"	"product page"	December 22, 2025 at 2:50:45 PM UTC+5:30	"2"
2A3BygaD97JhMrzLJ064	"Viewed payment"	"payment page"	December 22, 2025 at 2:50:45 PM UTC+5:30	"1"

8.Practical Explanation of Database Selection

Why SQL is Used for Transactions

- Orders and payments involve financial data
- Requires strong consistency
- Supports commit and rollback
- Maintains relationships between tables

Example: If a payment fails, the order must not be confirmed.

Why NoSQL is Used for Logs

- User logs are high-volume and write-heavy
- Schema can change frequently
- No transactions are required
- Fast time-based queries

Example: Missing one click log does not affect business operations.

Conclusion

This project demonstrates the design of an e-commerce platform using a hybrid database approach on Google Cloud Platform.

Cloud SQL is used for transactional data to ensure accuracy, while Firestore is used for log data to handle scalability and performance.

The design follows real-world best practices used in modern cloud-based applications.