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:

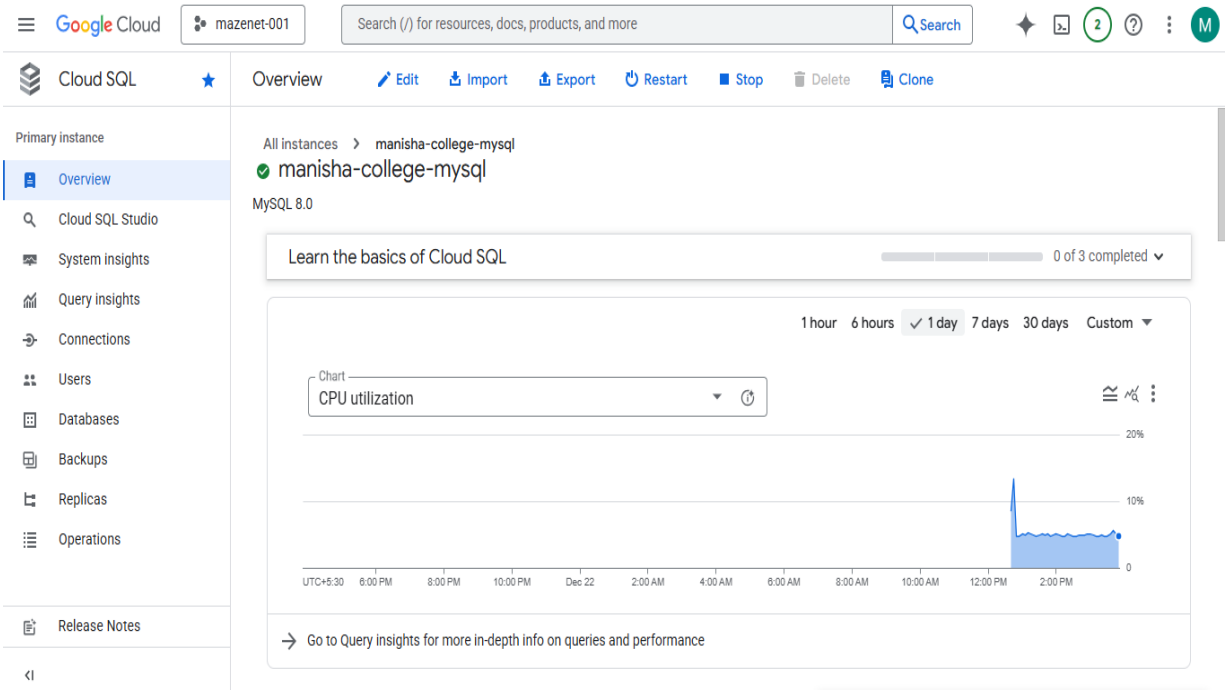
- To create a Cloud SQL (MySQL/PostgreSQL) instance
- To create a database named college_db
- To design and create a students table
- To insert student records into the database
- To perform SQL queries for data analysis
- To secure the database using user permissions and network restrictions

3.Tools and Technologies Used

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

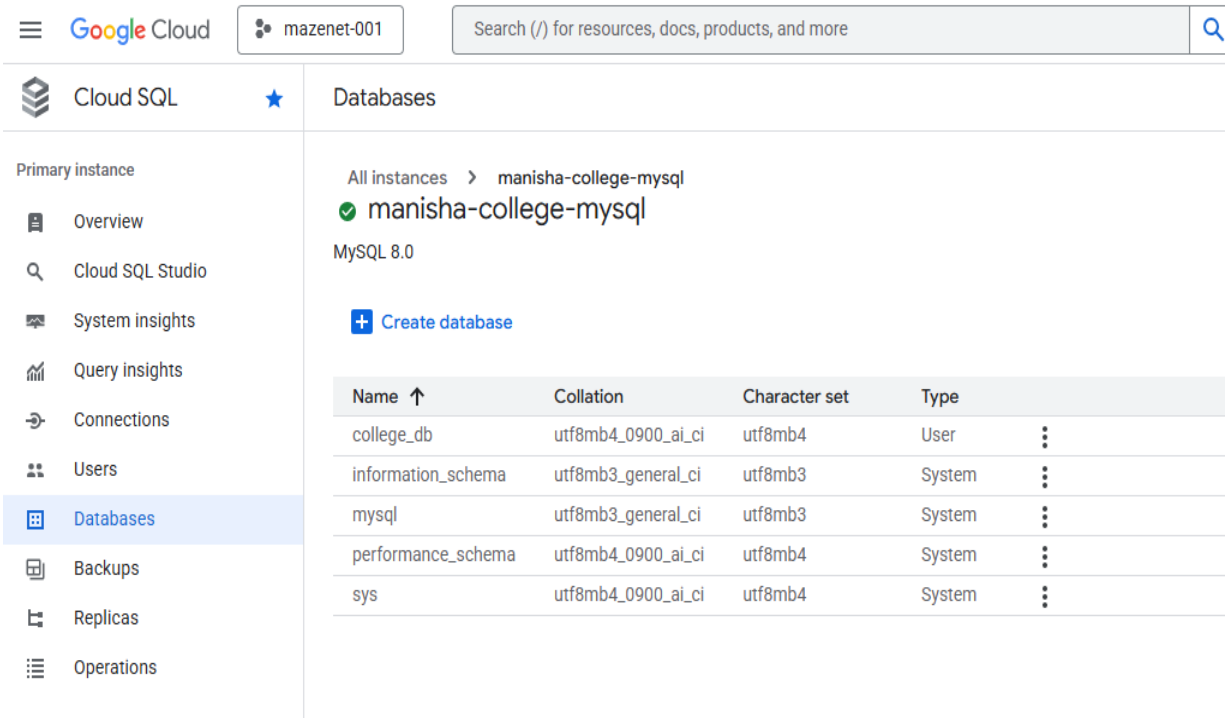
4.Cloud SQL Instance Creation

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.



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.



6.Table Creation – Students Table

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

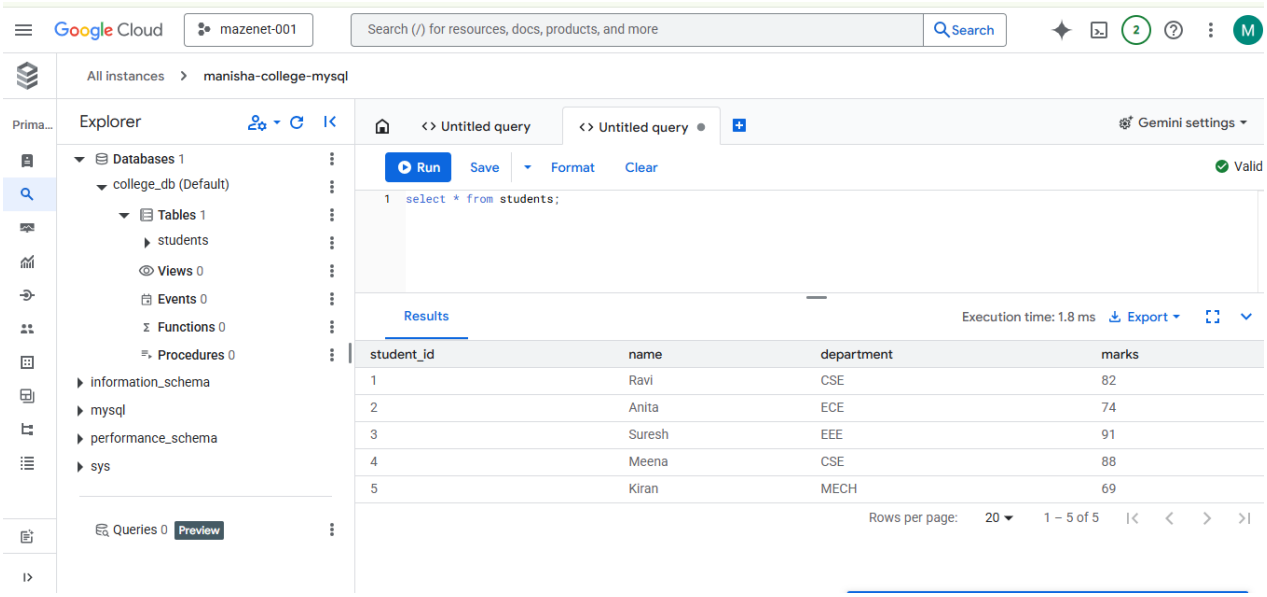
Table Structure:

- **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

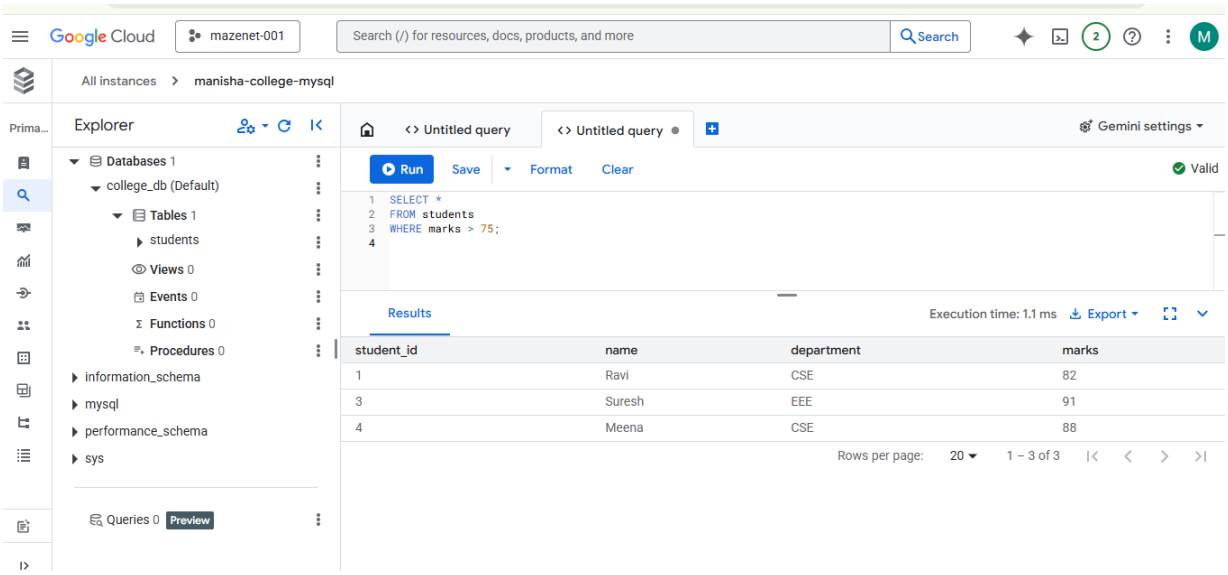
At least **five student records** are inserted into the `students` table to simulate real academic data.
The inserted records include students from different departments with different marks.



8.SQL Queries Performed

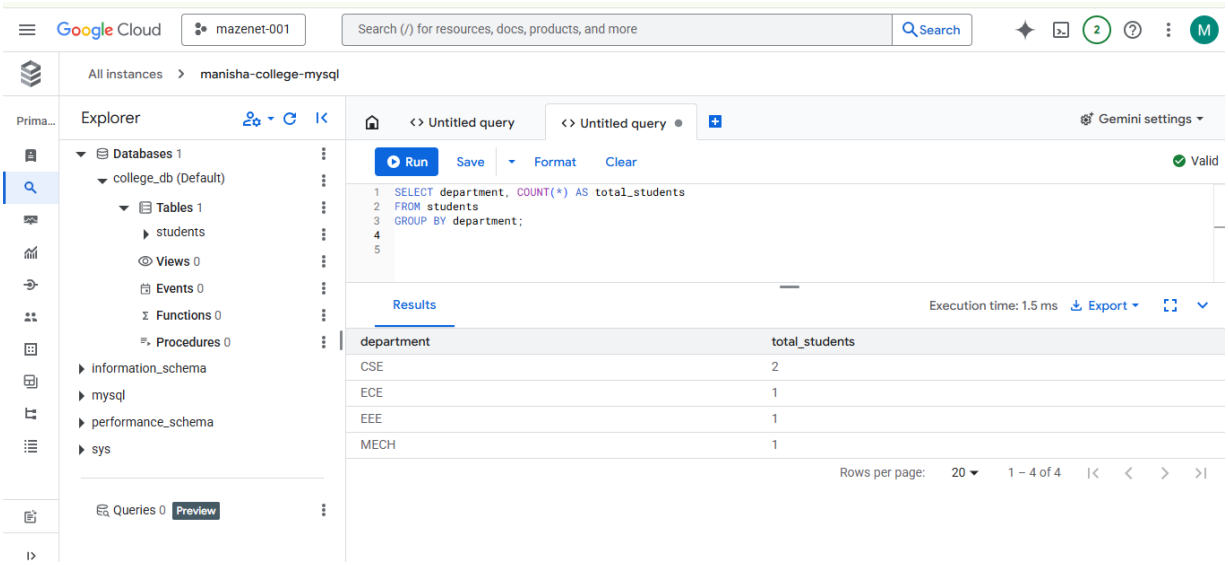
Fetch Students with Marks Greater Than 75

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



Count Students per Department

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

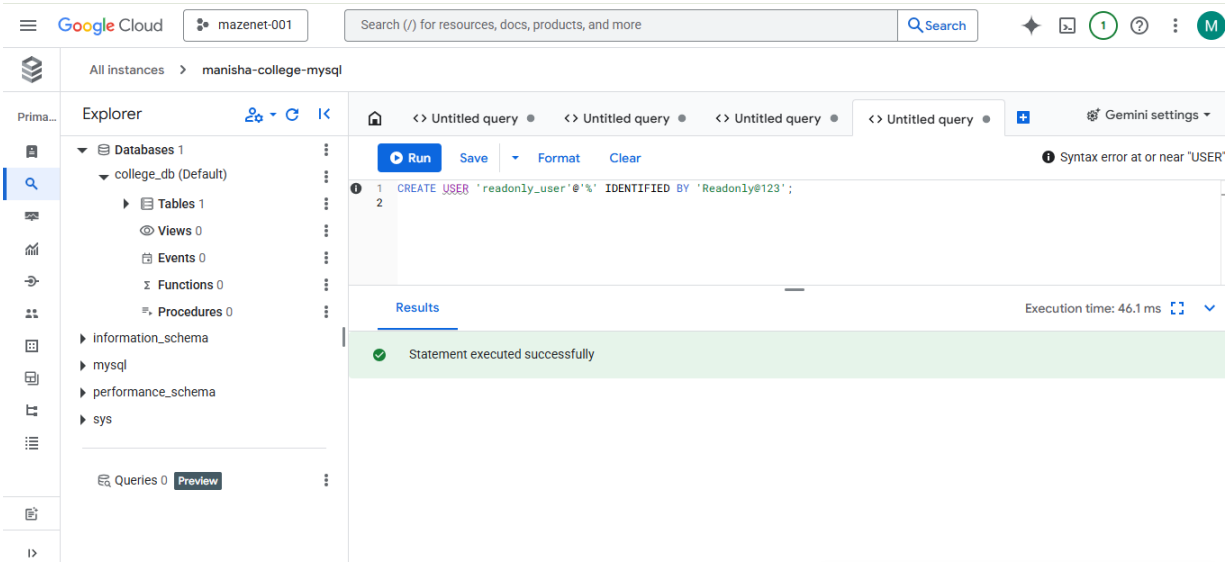


9. Database Security Implementation

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

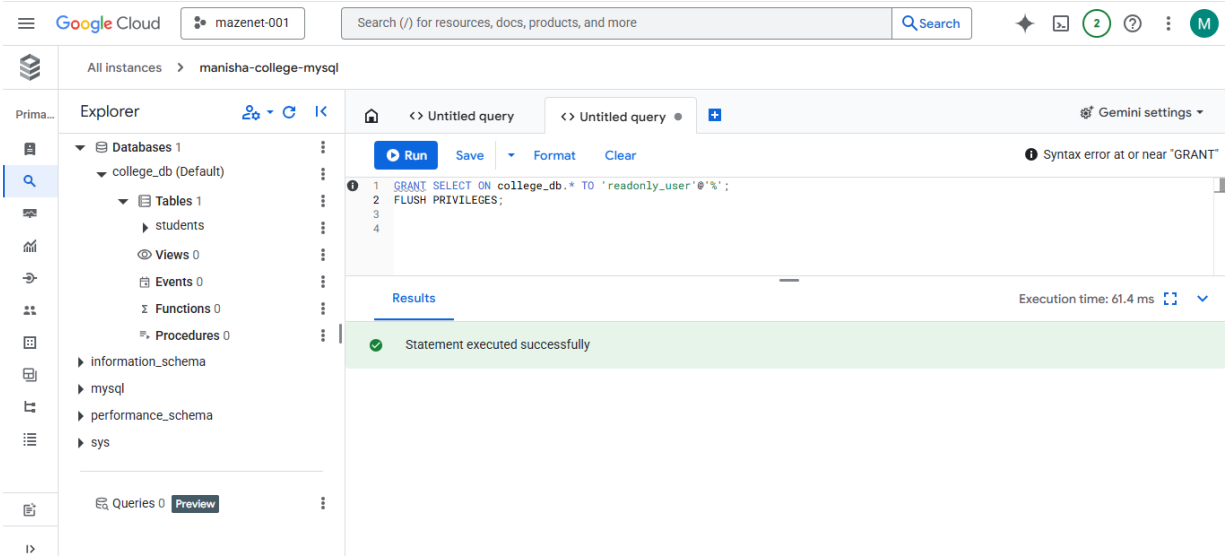
Read-Only User Creation

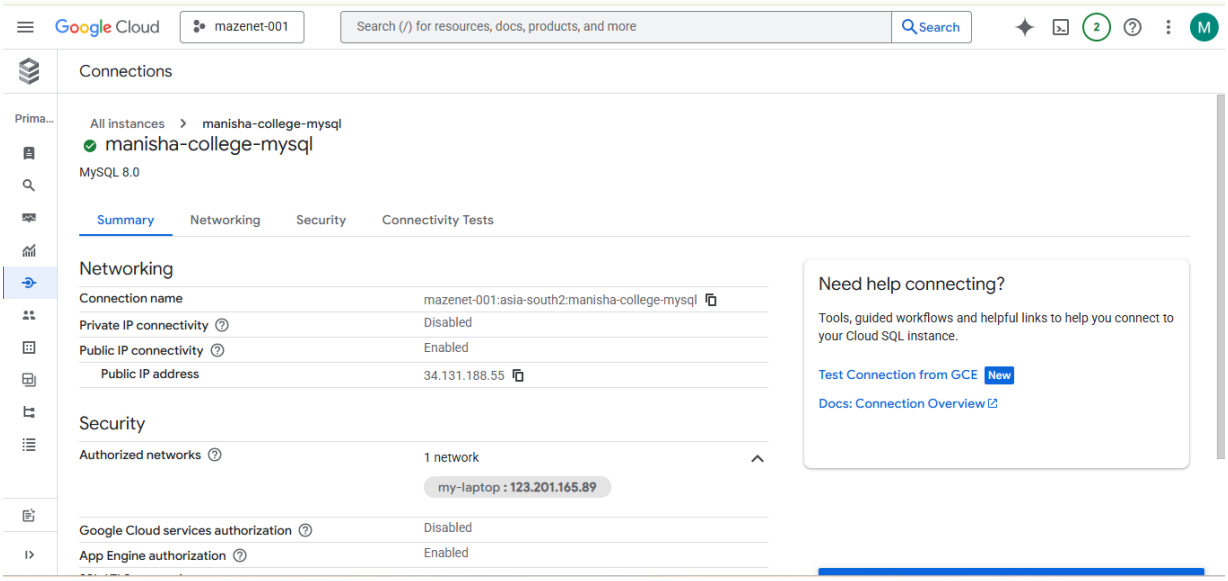
A separate database user is created with **read-only permissions**.
This user can view student data but cannot modify or delete records, ensuring data safety.



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.





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:

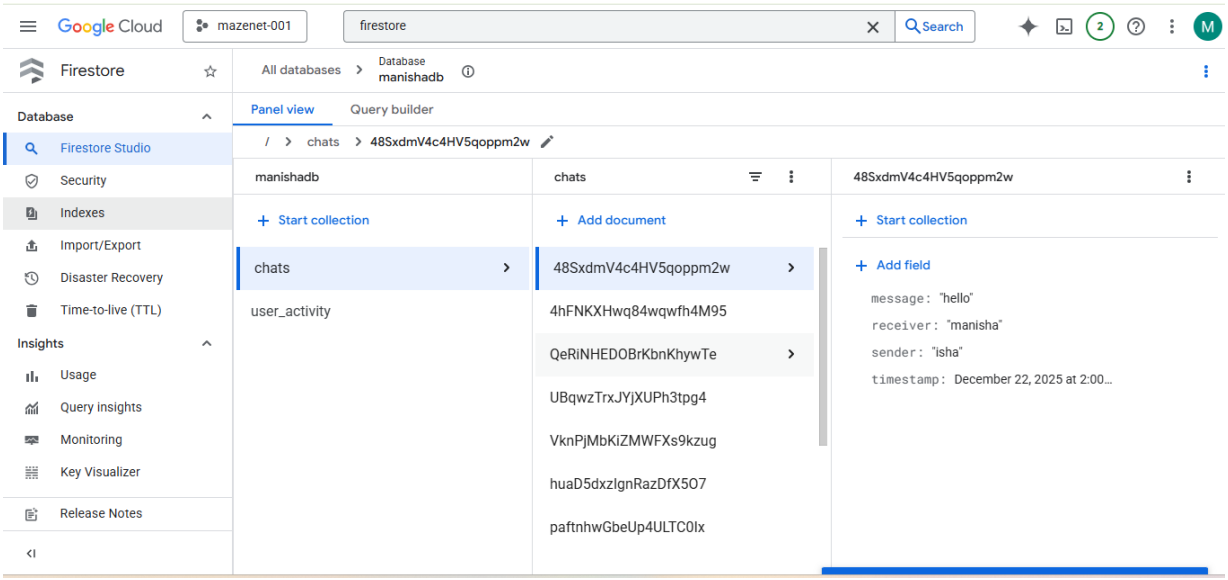
- To enable and configure **Firestore in Native mode**
- To design a data model for storing chat messages
- To insert multiple chat messages into Firestore
- To retrieve and sort chat messages between users
- To implement **security rules** to protect user data

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.



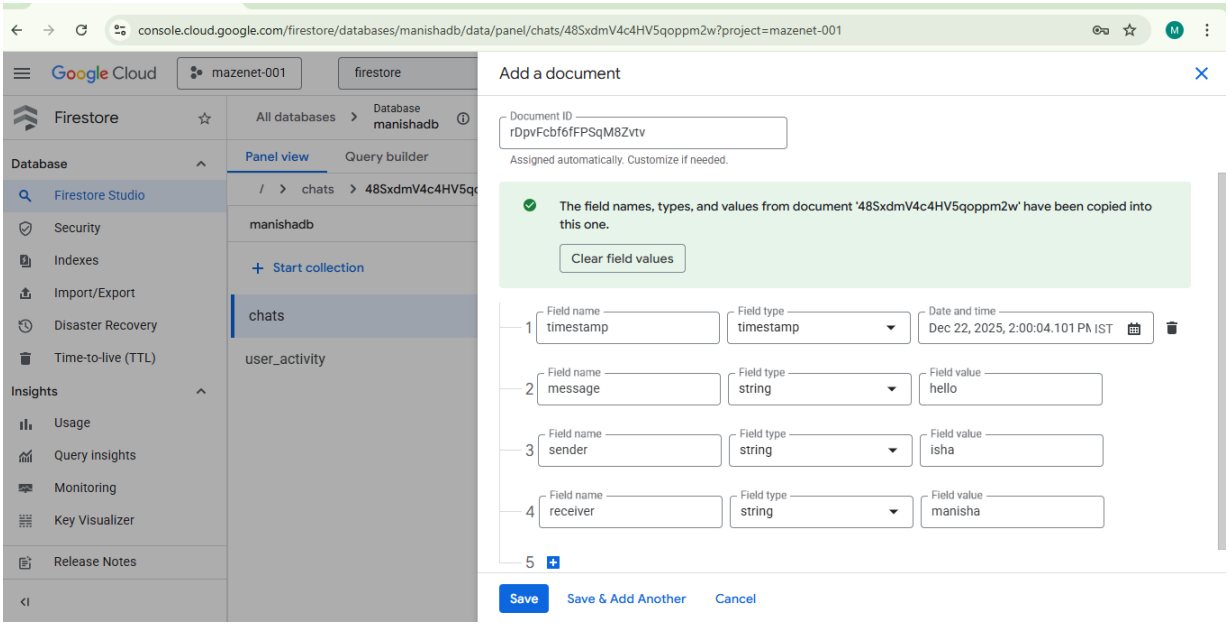
5.Data Modeling – Chats Collection

A collection named **`chats`** is created in Firestore.
Each document in the collection represents **one chat message**.

Each chat document stores the following fields:

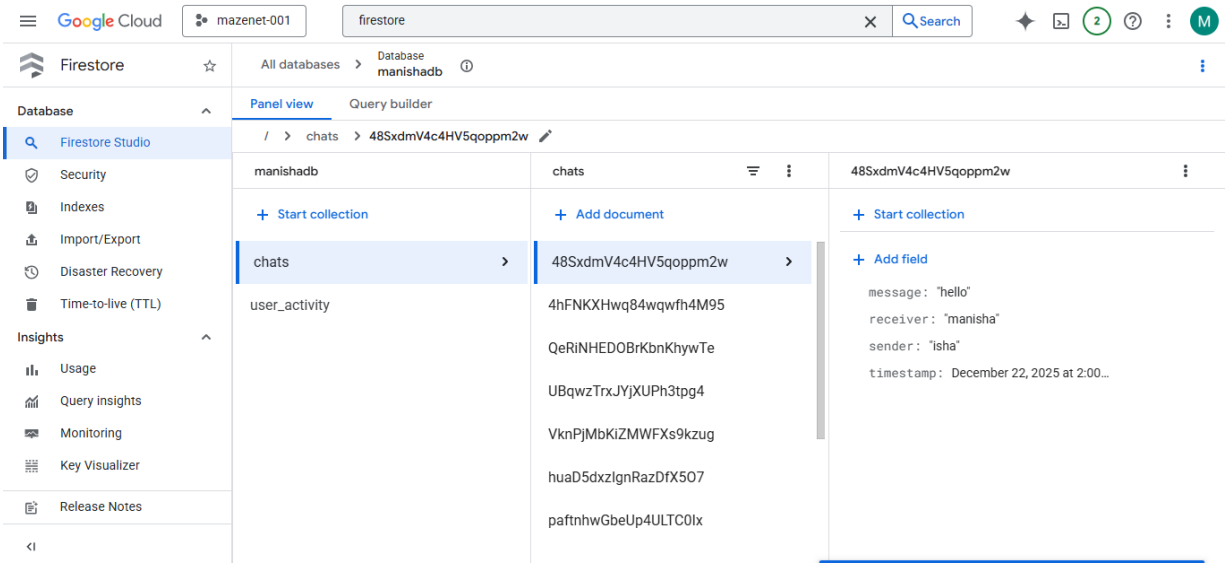
- **sender** – User who sends the message
- **receiver** – User who receives the message
- **message** – Text content of the chat
- **timestamp** – Time when the message was sent

This design allows efficient storage and retrieval of chat data.



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.

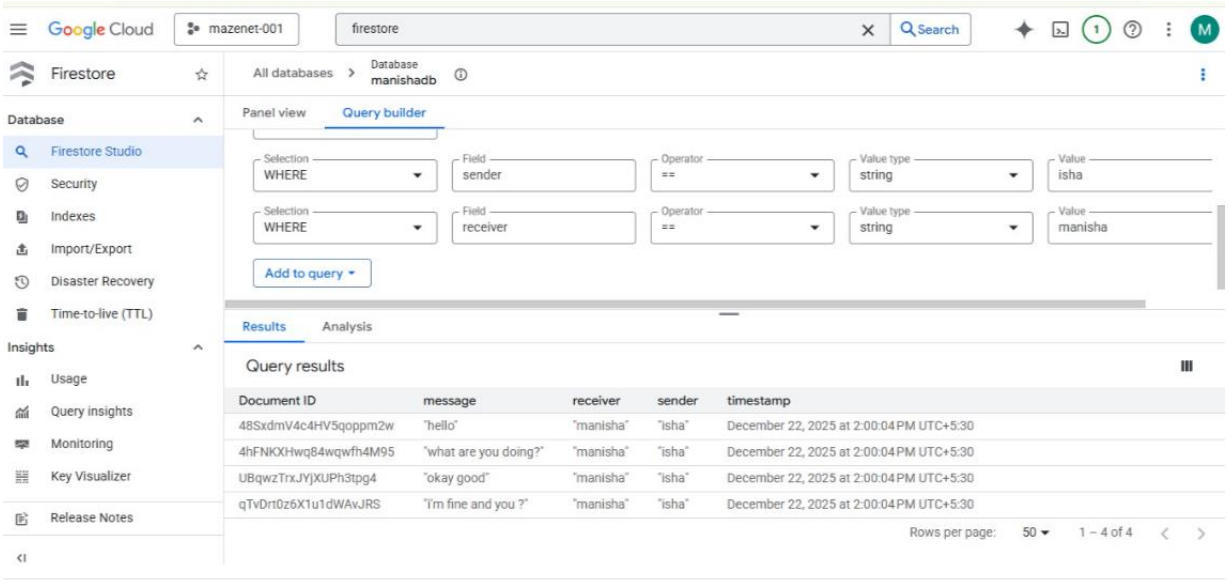


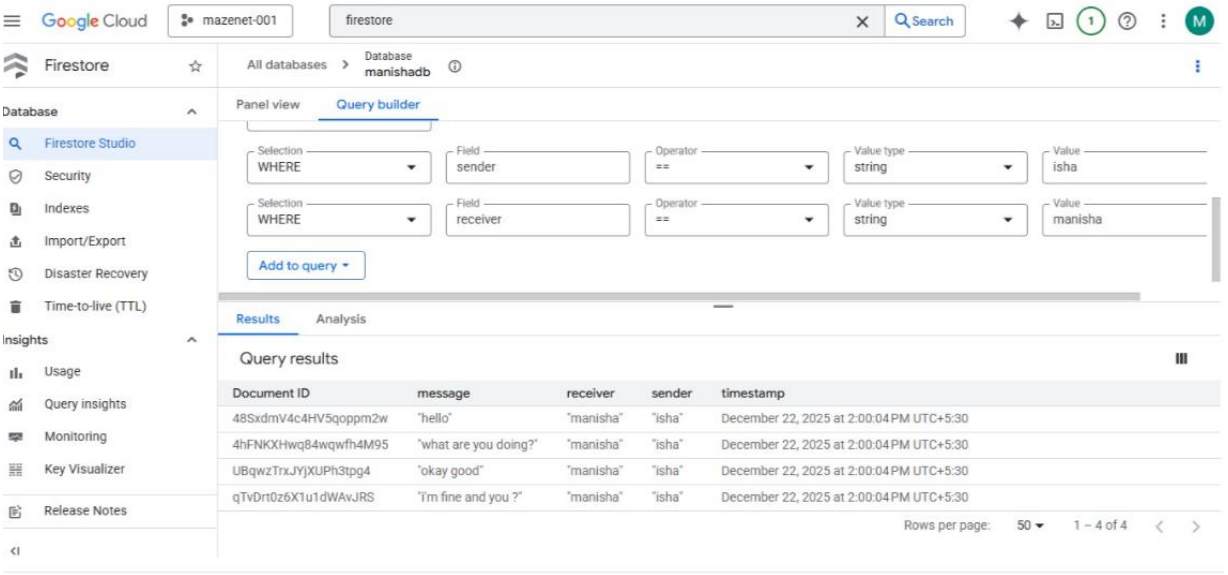
7.Querying Chat Messages

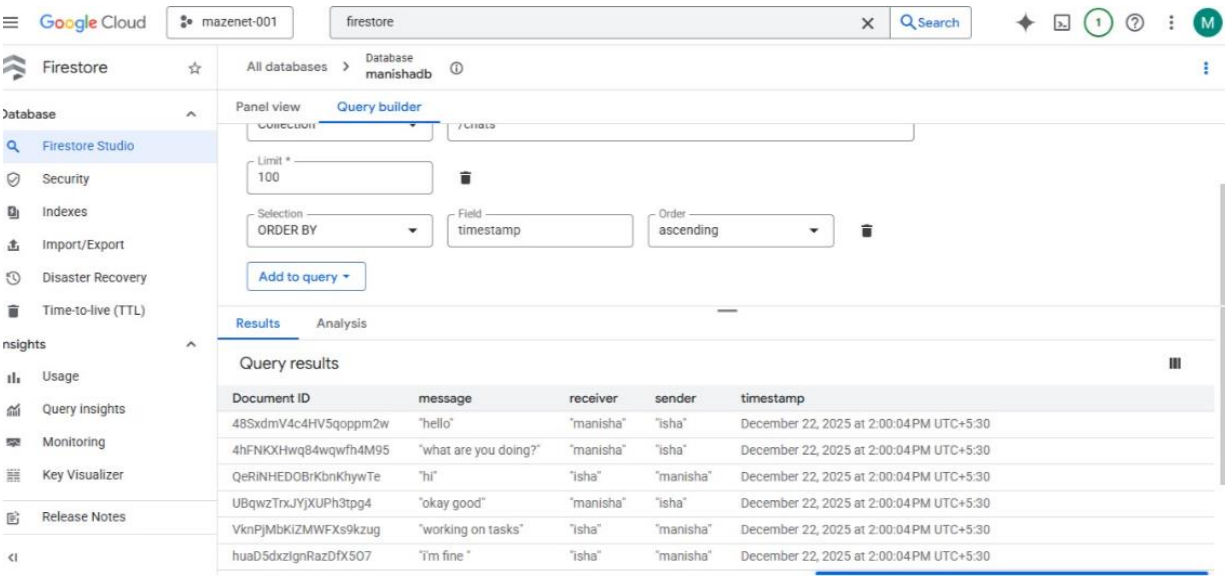
Firestore queries are used to:

- 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
Search
Firestore
All databases
Database
manishadb
Panel view
Query builder
Database
Firestore Studio
Security
Indexes
Import/Export
Disaster Recovery
Time-to-live (TTL)
Insights
Usage
Query insights
Monitoring
Key Visualizer
Release Notes
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
Search
Firestore
All databases
Database
manishadb
Panel view
Query builder
Database
Firestore Studio
Security
Indexes
Import/Export
Disaster Recovery
Time-to-live (TTL)
Insights
Usage
Query insights
Monitoring
Key Visualizer
Release Notes
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

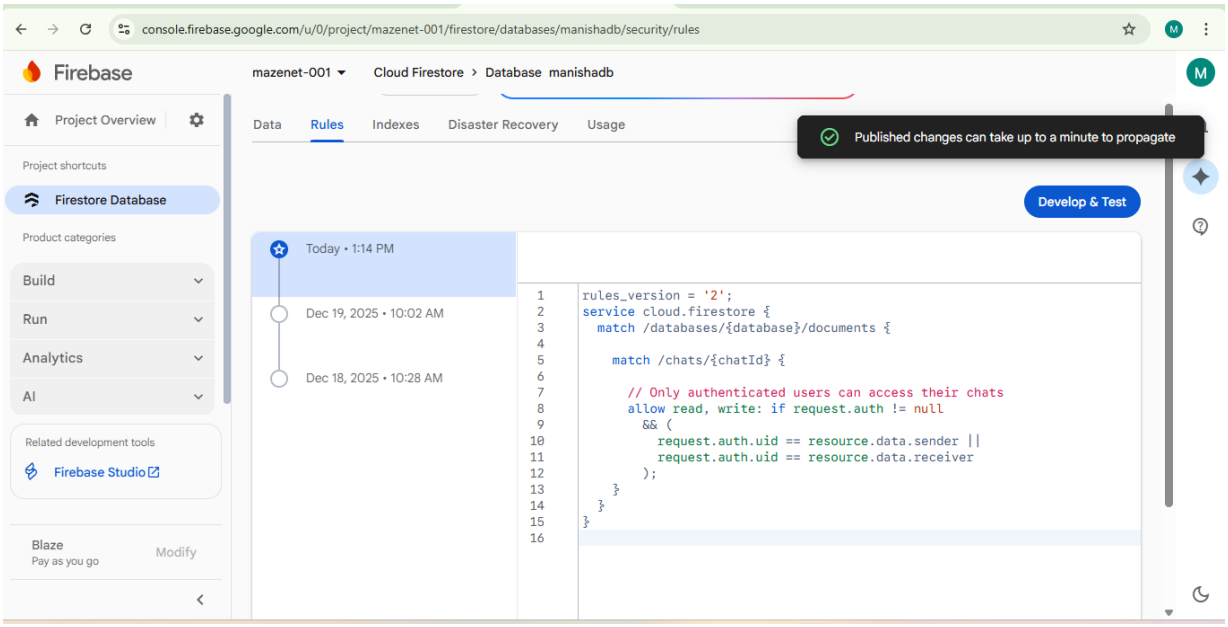


8.Security Rules Implementation

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.



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

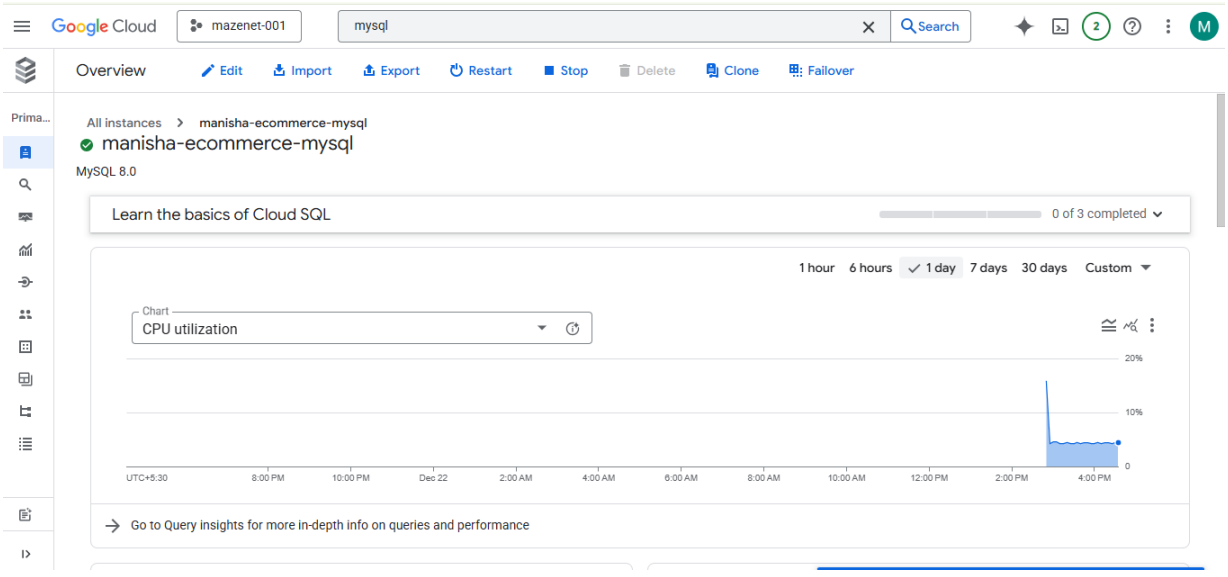
Purpose

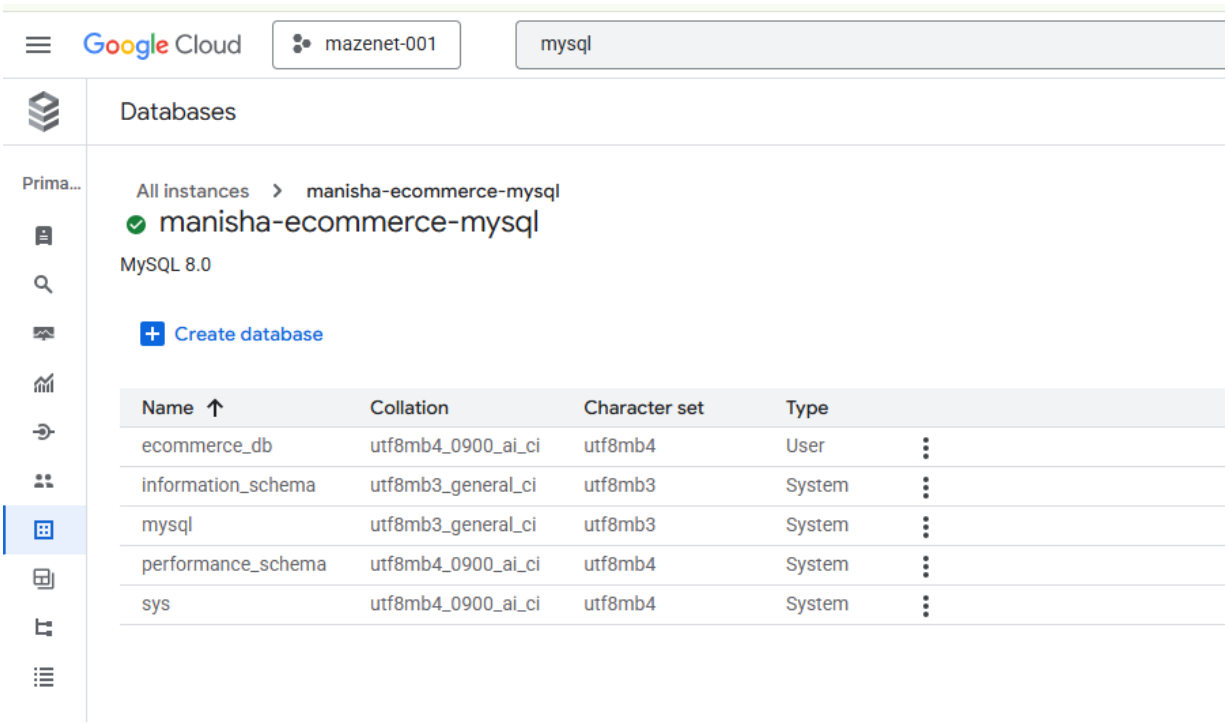
Cloud SQL is used to store **orders and payments**, which are critical transactional data.

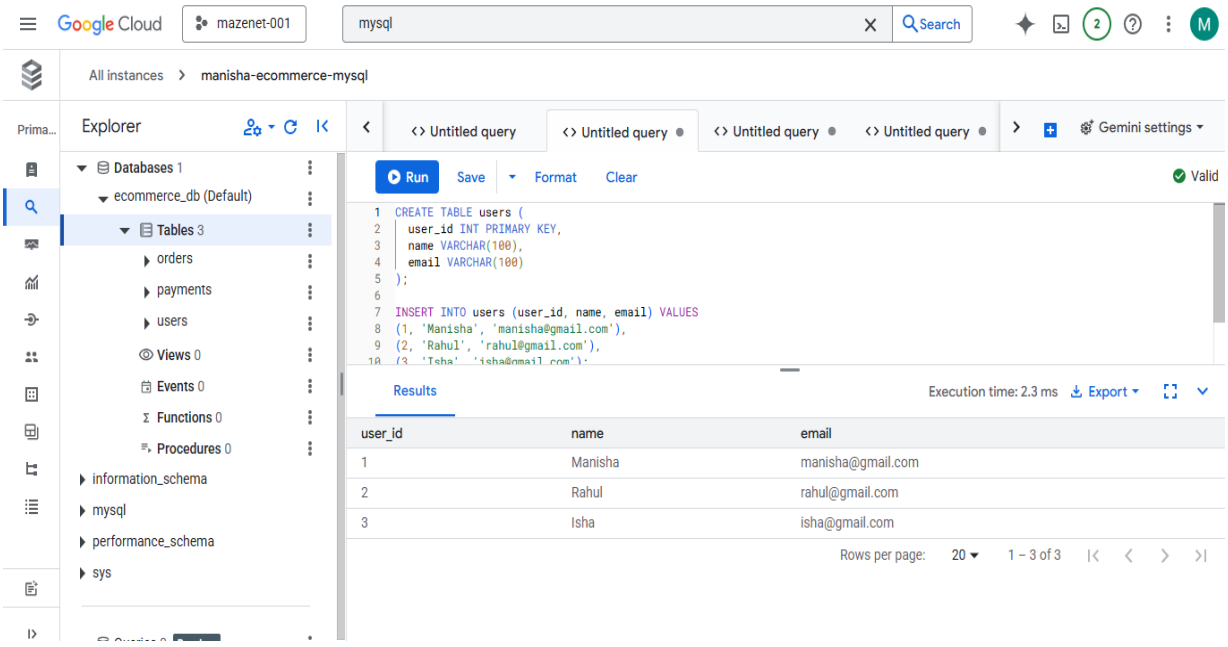
Data Stored

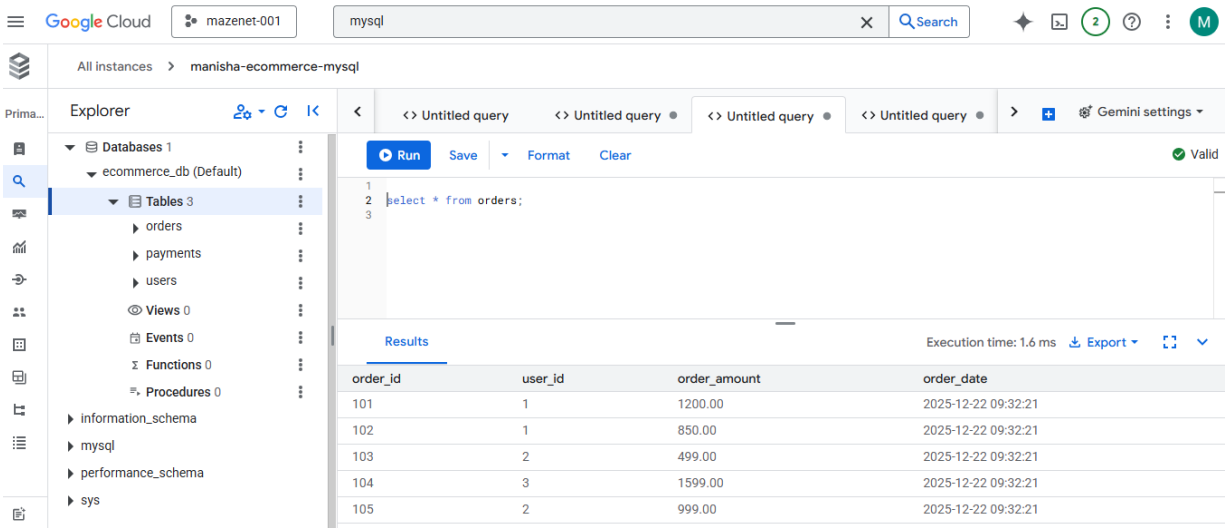
- Order details such as order ID, user ID, order amount, and order date
- Payment details such as payment ID, order ID, payment status, and payment date

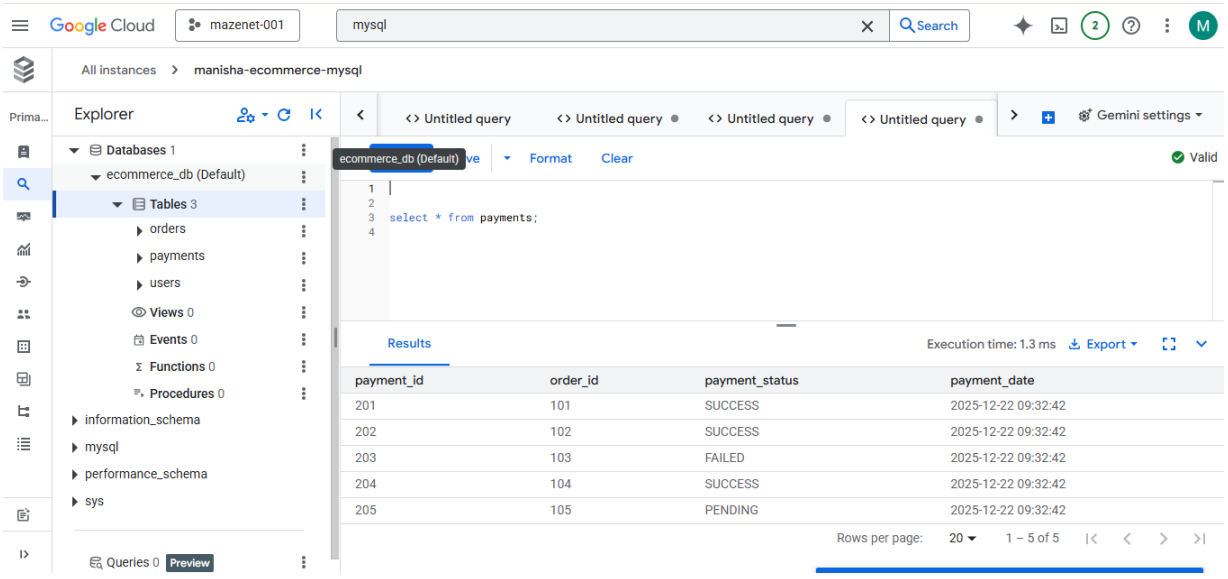
Cloud SQL provides **ACID compliance**, ensuring data consistency and accuracy during financial transactions.

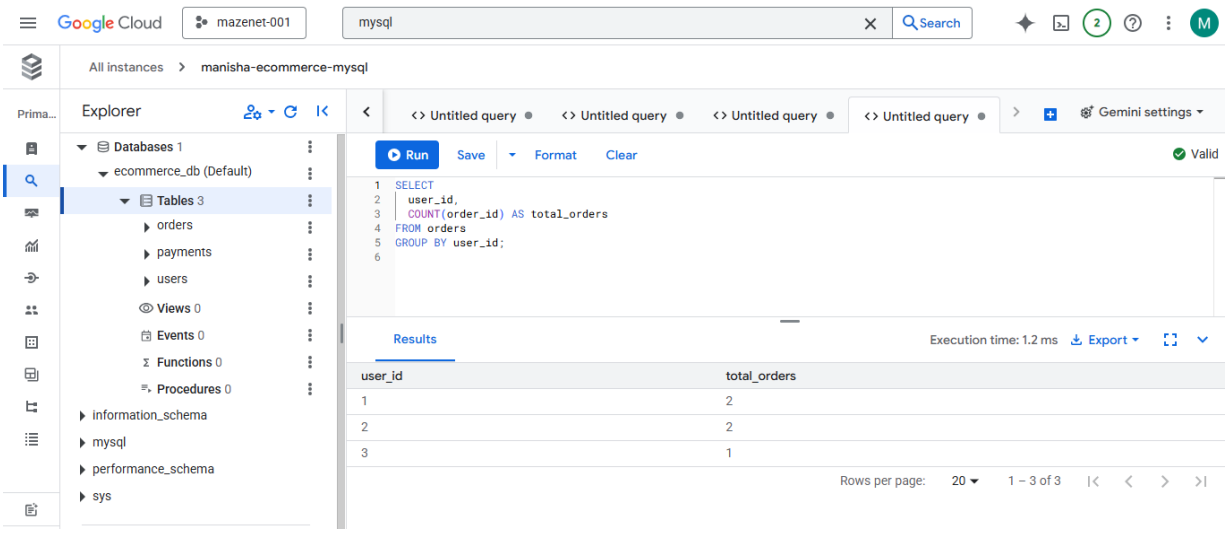



Google Cloud
mazenet-001
mysql
Databases
All instances
manisha-ecommerce-mysql
manisha-ecommerce-mysql
MySQL 8.0
Create database
Name
Collation
Character set
Type
ecommerce_db
utf8mb4_0900_ai_ci
utf8mb4
User
information_schema
utf8mb3_general_ci
utf8mb3
System
mysql
utf8mb3_general_ci
utf8mb3
System
performance_schema
utf8mb4_0900_ai_ci
utf8mb4
System
sys
utf8mb4_0900_ai_ci
utf8mb4
System


Google Cloud
mazenet-001
mysql
Search
All instances
manisha-ecommerce-mysql
Explorer
Untitled query
Untitled query
Untitled query
Untitled query
Gemini settings
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
Valid
CREATE TABLE users (
user_id INT PRIMARY KEY,
name VARCHAR(100),
email VARCHAR(100)
);
INSERT INTO users (user_id, name, email) VALUES
(1, 'Manisha', 'manisha@gmail.com'),
(2, 'Rahul', 'rahul@gmail.com'),
Results
Execution time: 2.3 ms
Export
user_id
name
email
1
Manisha
manisha@gmail.com
2
Rahul
rahul@gmail.com
3
Isha
isha@gmail.com
Rows per page:
20
1 – 3 of 3


Google Cloud
mazenet-001
mysql
Search
All instances
manisha-ecommerce-mysql
Explorer
Untitled query
Untitled query
Untitled query
Untitled query
Gemini settings
Run
Save
Format
Clear
Valid
select * from orders;
Databases 1
ecommerce_db (Default)
Tables 3
orders
payments
users
Views 0
Events 0
Functions 0
Procedures 0
information_schema
mysql
performance_schema
sys
Results
Execution time: 1.6 ms
Export
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


Google Cloud
mazenet-001
mysql
Search
All instances
manisha-ecommerce-mysql
Explorer
Untitled query
Untitled query
Untitled query
Untitled query
Gemini settings
ecommerce_db (Default)
Format
Clear
Valid
select * from payments;
Databases 1
ecommerce_db (Default)
Tables 3
orders
payments
users
Views 0
Events 0
Functions 0
Procedures 0
information_schema
mysql
performance_schema
sys
Queries 0
Preview
Results
Execution time: 1.3 ms
Export
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
Untitled query
Untitled query
Untitled query
Untitled query
Gemini settings
Run
Save
Format
Clear
Valid
SELECT
user_id,
COUNT(order_id) AS total_orders
FROM orders
GROUP BY user_id;
Databases 1
ecommerce_db (Default)
Tables 3
orders
payments
users
Views 0
Events 0
Functions 0
Procedures 0
information_schema
mysql
performance_schema
sys
Results
Execution time: 1.2 ms
Export
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

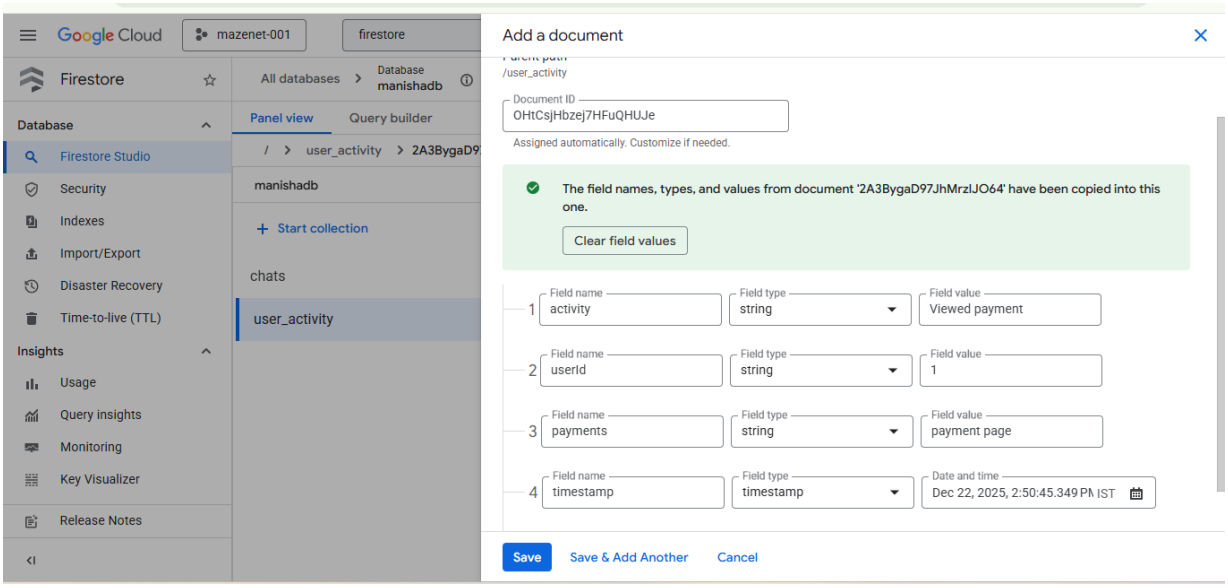
Purpose

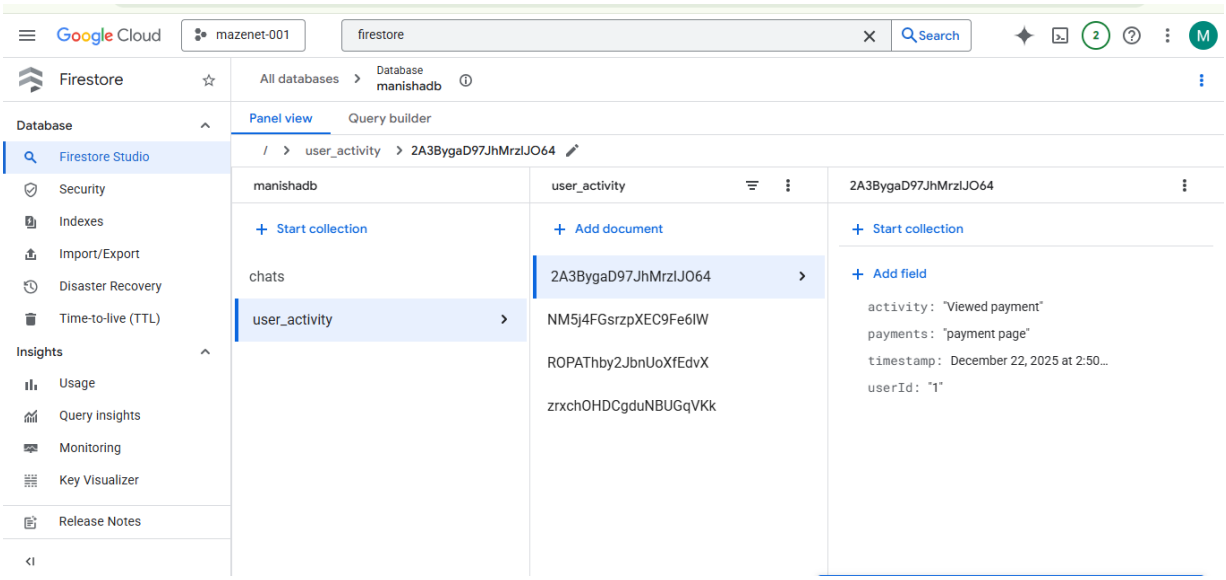
Firestore is used to store **user activity logs** and **clickstream data**, which are generated in large volumes.

Data Stored

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

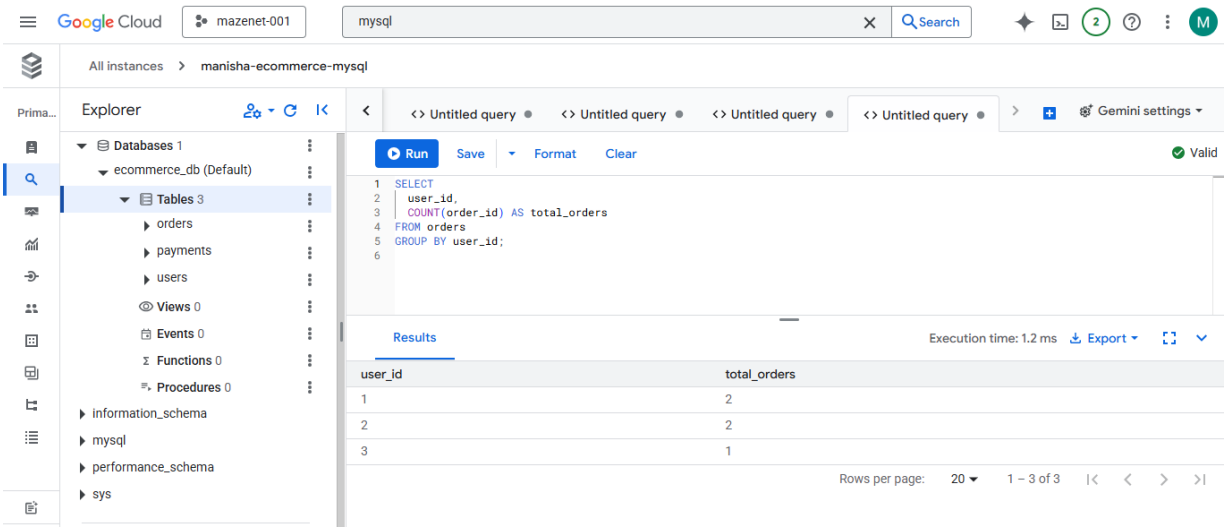
Each document represents **one user action**, allowing efficient time-based queries.





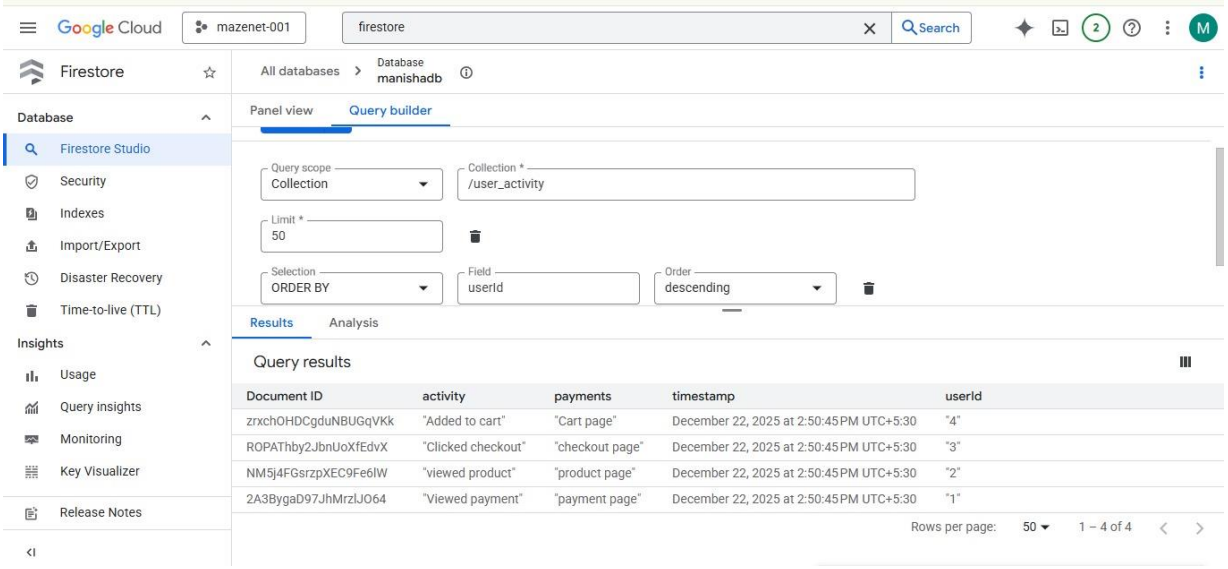
6.SQL Query Implementation

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.



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.



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.