MYSQL Project

on

AI Chatbot System

Submitted

At

I.T Vedant 2024

Submitted To:
Mr. Samir Warsolkar sir.
Branch:Thane(IT Vedant)

Submiited By
Prerana Rokade.
Course Name: Data Science and Data Analytics with Artificial Intelligence

Branch :Thane

Timing: 10AM TO 12PM

Rokadeprerana55@gmail.com

AI Chatbot System Description:

The SQL Part of the AI chatbot system is designed to facilitate natural language interactions between users and an intelligent virtual assistant. The Entity relationship diagram for such a system is explained in detail hereafter.

1.User Management:

Users can register accounts with the chatbot system.Authentication is performed using username/password credentials.

2.Conversation Handling:

Each user can initiate multiple conversations with the chatbot. Conversations are timestamped to track their duration.

3.Message Processing:

Users can send messages containing queries or requests to the chatbot.Messages are associated with the corresponding conversation and sender.

4.Intent Detection:

The chatbot analyzes user messages to detect the underlying intent.Detected intents are associated with interactions between users and the chatbot.

5.Interaction Logging:

Each user interaction with the chatbot is logged, including the detected intent and confidence level.

The ER diagram for the AI chatbot system contains a total of 5 entities (tables) and their associated attributes (columns). These entities include

1.user

2.Conversation

3.Message

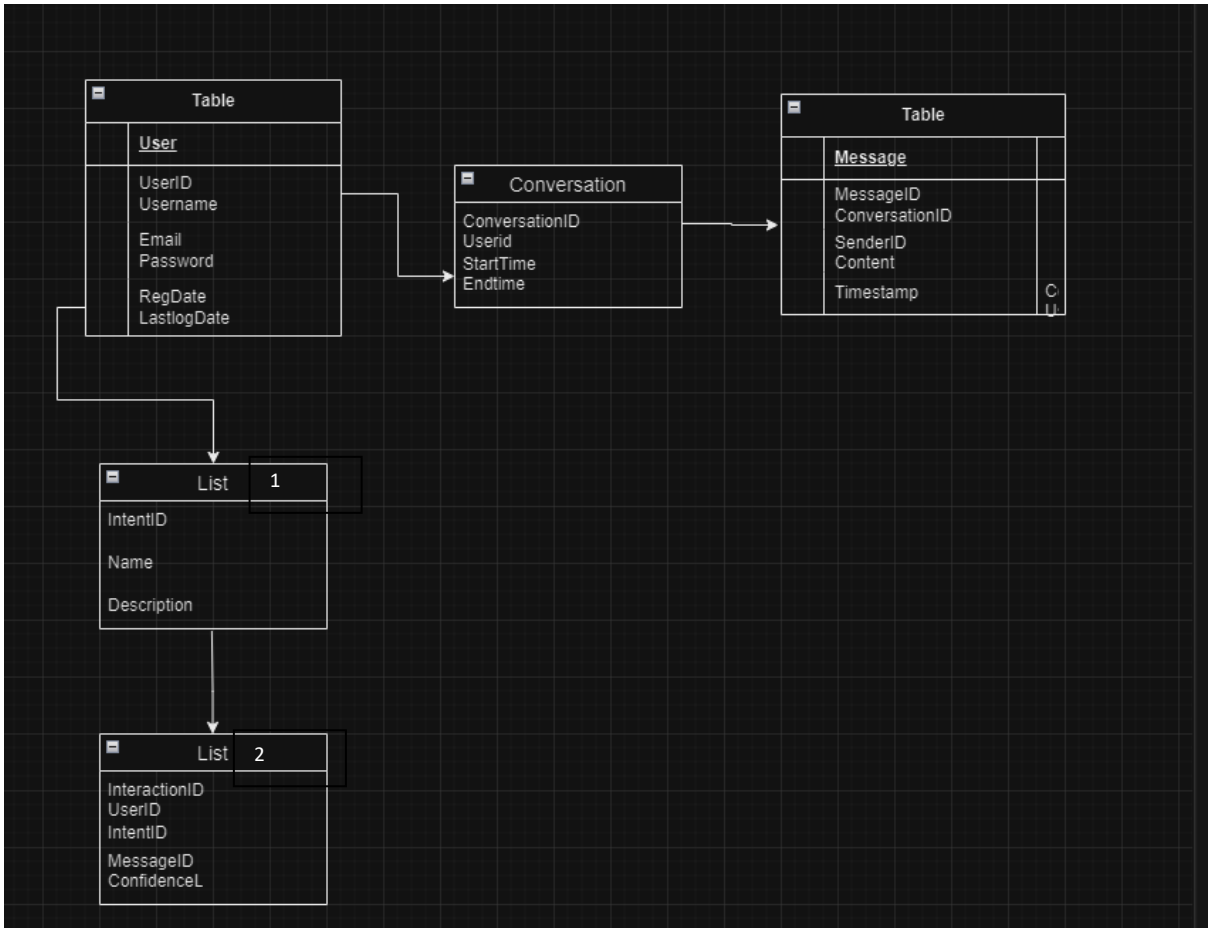
4.Intent

5.Interaction.

The database schema encompasses the structure needed to store user information, conversations between users and the chatbot, individual messages exchanged within conversations, detected intents behind user messages, and the interactions between users and the chatbot, including the confidence level of detected intents.

While the exact size of the database would depend on factors such as the number of users, volume of conversations, frequency of interactions, and the complexity of intents, the ER diagram provides a comprehensive framework for organizing and managing the data required for the AI chatbot system.

ER-Diagram (Entity Relation – Diagram) for Artificial Intelligence Chatbot Using Sql



. Table Descriptions:

1.User

Field	Type	Null	Key	Default	Extra
UserID	int(11)	NO	PRI	NULL	
Username	varchar(50)	YES		NULL	
Email	varchar(100)	YES		NULL	
Password	varchar(100)	YES		NULL	
RegistrationDate	datetime	YES		NULL	
LastLoginDate	datetime	YES		NULL	

2.Conversation

Field	Type	Null	Key	Default	Extra
ConversationID	int(11)	NO	PRI	NULL	
UserID	int(11)	YES	MUL	NULL	
StartTime	datetime	YES		NULL	
EndTime	datetime	YES		NULL	

3.Message

Field	Type	Null	Key	Default	Extra
MessageID	int(11)	NO	PRI	NULL	
ConversationID	int(11)	YES	MUL	NULL	
SenderID	int(11)	YES	MUL	NULL	
Content	text	YES		NULL	
Timestamp	datetime	YES		NULL	

4. Intent

Field	Type	Null	Key	Default	Extra
IntentID	int(11)	NO	PRI	NULL	
Name	varchar(50)	YES		NULL	
Description	text	YES		NULL	

5.Interaction

Field	Type	Null	Key	Default	Extra
InteractionID	int(11)	NO	PRI	NULL	
UserID	int(11)	YES	MUL	NULL	
IntentID	int(11)	YES	MUL	NULL	
MessageID	int(11)	YES	MUL	NULL	
ConfidenceLevel	decimal(5,4)	YES		NULL	

```
USE AI_Chatbot;

-- Drop tables if they exist
DROP TABLE IF EXISTS Interaction;
DROP TABLE IF EXISTS Intent;
DROP TABLE IF EXISTS Message;
DROP TABLE IF EXISTS Conversation;
DROP TABLE IF EXISTS User;

-- Create User table
CREATE TABLE  User (
    UserID INT AUTO_INCREMENT PRIMARY KEY,
    Username VARCHAR(255),
    Email VARCHAR(255),
    Password VARCHAR(255),
    RegistrationDate DATETIME,
    LastLoginDate DATETIME
);

-- Create Conversation table
CREATE TABLE Conversation (
```

```
    ConversationID INT AUTO_INCREMENT PRIMARY KEY,
    UserID INT,
    StartTime DATETIME,
    EndTime DATETIME,
    FOREIGN KEY (UserID) REFERENCES User(UserID)
);

-- Create Message table
CREATE TABLE IF NOT EXISTS Message (
    MessageID INT AUTO_INCREMENT PRIMARY KEY,
    ConversationID INT,
    SenderID INT,
    Content TEXT,
    FOREIGN KEY (ConversationID) REFERENCES Conversation(ConversationID),
    FOREIGN KEY (SenderID) REFERENCES User(UserID)
);

-- Create Intent table
CREATE TABLE IF NOT EXISTS Intent (
    IntentID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255),
    Description TEXT
);

-- Create Interaction table
CREATE TABLE IF NOT EXISTS Interaction (
    InteractionID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    UserID INT,
    IntentID INT,
    MessageID INT,
    ConfidenceLevel FLOAT,
    FOREIGN KEY (UserID) REFERENCES User(UserID),
    FOREIGN KEY (IntentID) REFERENCES Intent(IntentID),
    FOREIGN KEY (MessageID) REFERENCES Message(MessageID)
);


-- Insert sample data into User table with real historical timestamps
INSERT INTO User (Username, Email, Password, RegistrationDate, LastLoginDate)
VALUES
    ('John Doe', 'john.doe@example.com', 'password123', '2023-12-01 08:00:00', '2024-04-20 10:30:00'),
    ('Jane Smith', 'jane.smith@example.com', 'securepass', '2023-12-05 10:00:00', '2024-04-22 14:45:00'),
    ('Alice Johnson', 'alice.johnson@example.com', 'pass1234', '2023-12-10 12:00:00', '2024-04-23 09:15:00'),
    ('Bob White', 'bob.white@example.com', 'password', '2023-12-15 14:00:00', '2024-04-24 11:20:00'),
    ('Emma Davis', 'emma.davis@example.com', 'password1234', '2023-12-20 16:00:00', '2024-04-25 08:45:00'),
    ('James Wilson', 'james.wilson@example.com', '123456', '2023-12-25 18:00:00', '2024-04-26 10:00:00'),
    ('Sarah Brown', 'sarah.brown@example.com', 'password321', '2023-12-30 20:00:00', '2024-04-27 12:00:00'),
    ('Michael Taylor', 'michael.taylor@example.com', 'mypassword', '2024-01-04 22:00:00', '2024-04-28 14:00:00'),
    ('Laura Clark', 'laura.clark@example.com', 'password123', '2024-01-09 08:00:00', '2024-04-29 16:00:00'),
```

('David Martinez', 'david.martinez@example.com', 'password', '2024-01-14 10:00:00', '2024-04-30 18:00:00'),

('Jennifer Hall', 'jennifer.hall@example.com', 'securepass123', '2024-01-19 12:00:00', '2024-05-01 20:00:00'),

('Daniel Thompson', 'daniel.thompson@example.com', 'pass123', '2024-01-24 14:00:00', '2024-05-02 22:00:00'),

('Jessica Lee', 'jessica.lee@example.com', 'password123', '2024-01-29 16:00:00', '2024-05-03 08:00:00'),

('Kevin Rodriguez', 'kevin.rodriguez@example.com', 'securepassword', '2024-02-03 18:00:00', '2024-05-04 10:00:00'),

('Amanda Garcia', 'amanda.garcia@example.com', 'password321', '2024-02-08 20:00:00', '2024-05-05 12:00:00'),

('Ryan Martinez', 'ryan.martinez@example.com', '123456', '2024-02-13 22:00:00', '2024-05-06 14:00:00'),

('Nicole Hernandez', 'nicole.hernandez@example.com', 'mypassword', '2024-02-18 08:00:00', '2024-05-07 16:00:00'),

('Justin Smith', 'justin.smith@example.com', 'password1234', '2024-02-23 10:00:00', '2024-05-08 18:00:00'),

('Samantha Johnson', 'samantha.johnson@example.com', 'password', '2024-02-28 12:00:00', '2024-05-09 20:00:00'),

('Brandon Davis', 'brandon.davis@example.com', 'password123', '2024-03-04 14:00:00', '2024-05-10 22:00:00'),

('Rachel Wilson', 'rachel.wilson@example.com', 'securepass', '2024-03-09 16:00:00', '2024-05-11 08:00:00'),

('Tyler Miller', 'tyler.miller@example.com', 'pass1234', '2024-03-14 18:00:00', '2024-05-12 10:00:00'),

('Lauren Anderson', 'lauren.anderson@example.com', 'password', '2024-03-19 20:00:00', '2024-05-13 12:00:00'),

('Andrew Thompson', 'andrew.thompson@example.com', 'password1234', '2024-03-24 22:00:00', '2024-05-14 14:00:00'),

('Megan Moore', 'megan.moore@example.com', '123456', '2024-03-29 08:00:00', '2024-05-15 16:00:00');

```
select * from  User;
-- Insert sample data into Message table
INSERT INTO Message (ConversationID, SenderID, Content)
VALUES
    (1, 1, 'Hello, how can I help you?'),
    (1, 2, 'Hi there, I have a question about...'),
    (2, 1, 'I need assistance with...');

-- Insert sample data into Intent table
INSERT INTO Intent (Name, Description)
VALUES
    ('Greeting', 'Intent for greeting messages'),
    ('Inquiry', 'Intent for inquiry messages'),
    ('Request', 'Intent for request messages');

-- Insert sample data into Interaction table
INSERT INTO Interaction (UserID, IntentID, MessageID, ConfidenceLevel)
VALUES
    (1, 1, 1, 0.9),
    (2, 2, 2, 0.8),
    (3, 3, 3, 0.7);

-- Insert sample data into Conversation table with specified timestamps
INSERT INTO Conversation (UserID, StartTime, EndTime)
VALUES
    (1, '2024-01-01 08:00:00', '2024-01-01 08:30:00'),
    (2, '2024-01-05 10:00:00', '2024-01-05 11:00:00'),
```

```
    (3, '2024-01-10 12:00:00', '2024-01-10 12:30:00'),
    (4, '2024-01-15 14:00:00', '2024-01-15 14:30:00'),
    (5, '2024-01-20 16:00:00', '2024-01-20 16:30:00'),
    (6, '2024-01-25 18:00:00', '2024-01-25 18:30:00'),
    (7, '2024-01-30 20:00:00', '2024-01-30 20:30:00'),
    (8, '2024-02-04 22:00:00', '2024-02-04 22:30:00'),
    (9, '2024-02-09 08:00:00', '2024-02-09 08:30:00'),
    (10, '2024-02-14 10:00:00', '2024-02-14 10:30:00'),
    (11, '2024-02-19 12:00:00', '2024-02-19 12:30:00'),
    (12, '2024-02-24 14:00:00', '2024-02-24 14:30:00'),
    (13, '2024-02-29 16:00:00', '2024-02-29 16:30:00'),
    (14, '2024-03-05 18:00:00', '2024-03-05 18:30:00'),
    (15, '2024-03-10 20:00:00', '2024-03-10 20:30:00'),
    (16, '2024-03-15 22:00:00', '2024-03-15 22:30:00'),
    (17, '2024-03-20 08:00:00', '2024-03-20 08:30:00'),
    (18, '2024-03-25 10:00:00', '2024-03-25 10:30:00'),
    (19, '2024-03-30 12:00:00', '2024-03-30 12:30:00'),
    (20, '2024-04-04 14:00:00', '2024-04-04 14:30:00'),
    (21, '2024-04-09 16:00:00', '2024-04-09 16:30:00'),
    (22, '2024-04-14 18:00:00', '2024-04-14 18:30:00'),
    (23, '2024-04-19 20:00:00', '2024-04-19 20:30:00'),
    (24, '2024-04-24 22:00:00', '2024-04-24 22:30:00'),
    (25, '2024-04-29 08:00:00', '2024-04-29 08:30:00');
    select * from Conversation;

-- Insert sample data into Intent table,
INSERT INTO Intent (Name, Description)
```

```
VALUES
    ('Farewell', 'Intent for farewell messages'),
    ('Complaint', 'Intent for complaint messages'),
    ('Praise', 'Intent for praise messages'),
    ('Feedback', 'Intent for feedback messages'),
    ('Question', 'Intent for question messages'),
    ('Confirmation', 'Intent for confirmation messages'),
    ('Apology', 'Intent for apology messages'),
    ('Appreciation', 'Intent for appreciation messages'),
    ('Suggestion', 'Intent for suggestion messages'),
    ('Offer', 'Intent for offer messages'),
    ('Warning', 'Intent for warning messages'),
    ('Acknowledgment', 'Intent for acknowledgment messages'),
    ('Encouragement', 'Intent for encouragement messages'),
    ('Request', 'Intent for request messages'),
    ('Explanation', 'Intent for explanation messages'),
    ('Reminder', 'Intent for reminder messages'),
    ('Assurance', 'Intent for assurance messages'),
    ('Agreement', 'Intent for agreement messages'),
    ('Disagreement', 'Intent for disagreement messages'),
    ('Confusion', 'Intent for confusion messages'),
    ('Clarification', 'Intent for clarification messages'),
    ('Approval', 'Intent for approval messages'),
    ('Rejection', 'Intent for rejection messages'),
    ('Support', 'Intent for support messages');

-- Insert sample data into Interaction table
```

```
INSERT INTO Interaction (UserID, IntentID, MessageID, ConfidenceLevel)
VALUES
    (1, 1, 1, 0.9),
    (2, 2, 2, 0.8),
    (3, 3, 3, 0.7),
    (4, 4, 4, 0.6),
    (5, 5, 5, 0.5),
    (6, 6, 6, 0.4),
    (7, 7, 7, 0.3),
    (8, 8, 8, 0.2),
    (9, 9, 9, 0.1),
    (10, 10, 10, 0.9),
    (11, 11, 11, 0.8),
    (12, 12, 12, 0.7),
    (13, 13, 13, 0.6),
    (14, 14, 14, 0.5),
    (15, 15, 15, 0.4),
    (16, 16, 16, 0.3),
    (17, 17, 17, 0.2),
    (18, 18, 18, 0.1),
    (19, 19, 19, 0.9),
    (20, 20, 20, 0.8),
    (21, 21, 21, 0.7),
    (22, 22, 22, 0.6),
    (23, 23, 23, 0.5),
    (24, 24, 24, 0.4),
    (25, 25, 25, 0.3);
```

```
select * from Interaction;

-- Total number of messages sent by each user Joins

SELECT u.UserID, u.Username, COUNT(m.MessageID) AS TotalMessagesSent

FROM User u

LEFT JOIN Conversation c ON u.UserID = c.UserID

LEFT JOIN Message m ON c.ConversationID = m.ConversationID

GROUP BY u.UserID, u.Username;


-- Total number of conversations started by each user:

SELECT u.UserID, u.Username, COUNT(c.ConversationID) AS TotalConversationsStarted

FROM User u

LEFT JOIN Conversation c ON u.UserID = c.UserID

GROUP BY u.UserID, u.Username;


-- Average confidence level of interactions for each user:

SELECT u.UserID, u.Username, AVG(i.ConfidenceLevel) AS AvgConfidenceLevel

FROM User u

LEFT JOIN Interaction i ON u.UserID = i.UserID

GROUP BY u.UserID, u.Username;


-- Total number of interactions of each type (intent):

SELECT i.Name, COUNT(*) AS TotalInteractions

FROM Intent i

LEFT JOIN Interaction inter ON i.IntentID = inter.IntentID

GROUP BY i.Name;
```

```
-- Find the most active users (by total interactions) who joined within the last month:
SELECT u.UserID, u.Username, COUNT(i.InteractionID) AS TotalInteractions
FROM User u
LEFT JOIN Interaction i ON u.UserID = i.UserID
WHERE u.RegistrationDate >= DATE_SUB(NOW(), INTERVAL 1 MONTH)
GROUP BY u.UserID, u.Username
ORDER BY TotalInteractions DESC;


-- Find users who have sent the most messages:
SELECT Username, Email
FROM User
WHERE UserID = (
   SELECT UserID
   FROM (
      SELECT UserID, COUNT(*) AS TotalMessages
      FROM Conversation
      JOIN Message ON Conversation.ConversationID = Message.ConversationID
      GROUP BY UserID
      ORDER BY TotalMessages DESC
      LIMIT 1
   ) AS SubQuery
);

-- List conversations started by users who have registered within the last month:
SELECT ConversationID, StartTime
```

```
FROM Conversation
WHERE UserID IN (
    SELECT UserID
    FROM User
    WHERE RegistrationDate >= DATE_SUB(NOW(), INTERVAL 1 MONTH)
);

-- Find intents with more than 100 interactions:
SELECT Name
FROM Intent
WHERE IntentID IN (
    SELECT IntentID
    FROM Interaction
    GROUP BY IntentID
    HAVING COUNT(*) > 20
);

-- List users who have never started a conversation:
SELECT Username
FROM User
WHERE UserID NOT IN (
    SELECT DISTINCT UserID
    FROM Conversation
);

-- Find the conversation with the most messages:
SELECT ConversationID
```

```
FROM Conversation
WHERE ConversationID = (
    SELECT ConversationID
    FROM (
        SELECT ConversationID, COUNT(*) AS TotalMessages
        FROM Message
        GROUP BY ConversationID
        ORDER BY TotalMessages DESC
        LIMIT 1
    ) AS SubQuery
);
-- Find conversations started by users who have sent the most messages
SELECT c.ConversationID, c.StartTime
FROM Conversation c
INNER JOIN (
    SELECT UserID, COUNT(*) AS TotalMessages
    FROM Conversation
    JOIN Message ON Conversation.ConversationID = Message.ConversationID
    GROUP BY UserID
    ORDER BY TotalMessages DESC
    LIMIT 1
) AS SubQuery ON c.UserID = SubQuery.UserID;
-- Find users who have not interacted with any intents
SELECT u.UserID, u.Username
FROM User u
LEFT JOIN Interaction i ON u.UserID = i.UserID
WHERE i.UserID IS NULL;
```

-- Find the unique set of users who have either started a conversation or interacted with an intent:

```
SELECT UserID FROM Conversation
UNION
SELECT UserID FROM Interaction;
-- Join with the Conversation table using the earliest login time
SELECT
    UserID,
    MIN(LastLoginDate) AS EarliestLoginTime
FROM User
GROUP BY UserID;
```

Commands:

Joins

1.Total number of messages sent by each user Joins?

Query: SELECT u.UserID, u.Username, COUNT(m.MessageID) AS TotalMessagesSent

```
FROM User u
LEFT JOIN Conversation c ON u.UserID = c.UserID
LEFT JOIN Message m ON c.ConversationID = m.ConversationID
GROUP BY u.UserID, u.Username;
```

Result:

UserID	Username	TotalMessagesSent
1	John Doe	2
2	Jane Smith	1
3	Alice Johnson	0
4	Bob White	0
5	Emma Davis	0
6	James Wilson	0
7	Sarah Brown	0

2.Total number of conversations started by each user:

Query: SELECT u.UserID, u.Username, COUNT(c.ConversationID) AS TotalConversationsStarted

FROM User u

LEFT JOIN Conversation c ON u.UserID = c.UserID

GROUP BY u.UserID, u.Username;

Result:

UserID	Username	TotalConversationsStarted
1	John Doe	2
2	Jane Smith	2
3	Alice Johnson	2
4	Bob White	1
5	Emma Davis	1
6	James Wilson	1
7	Sarah Brown	1

3.Average confidence level of interactions for each user:

Query: SELECT u.UserID, u.Username, AVG(i.ConfidenceLevel) AS AvgConfidenceLevel

FROM User u

LEFT JOIN Interaction i ON u.UserID = i.UserID

GROUP BY u.UserID, u.Username;

Result:

UserID	Username	AvgConfidenceLevel
1	John Doe	NULL
2	Jane Smith	NULL
3	Alice Johnson	NULL
4	Bob White	NULL
5	Emma Davis	NULL
6	James Wilson	NULL
7	Sarah Brown	NULL

4. Total number of interactions of each type (intent):

Query :SELECT i.Name, COUNT(*) AS TotalInteractions

FROM Intent i

LEFT JOIN Interaction inter ON i.IntentID = inter.IntentID

GROUP BY i.Name;

Result:

Name	TotalInteractions
Acknowledgment	1
Agreement	1
Apology	1
Appreciation	1
Approval	1
Assurance	1
Clarification	1

5. Total number of interactions by each user for each intent:

Query : SELECT u.UserID, u.Username, i.Name AS IntentName, COUNT(*) AS TotalInteractions

FROM User u

LEFT JOIN Interaction inter ON u.UserID = inter.UserID

LEFT JOIN Intent i ON inter.IntentID = i.IntentID

GROUP BY u.UserID, u.Username, i.Name;

Result:

UserID	Username	IntentName	TotalInteractions
1	John Doe	NULL	1
2	Jane Smith	NULL	1
3	Alice Johnson	NULL	1
4	Bob White	NULL	1
5	Emma Davis	NULL	1
6	James Wilson	NULL	1
7	Sarah Brown	NULL	1

6.Join with the Conversation table using the earliest login time

Quary;

```
SELECT
    UserID,
    MIN(LastLoginDate) AS EarliestLoginTime
FROM User
GROUP BY UserID;
```

Results:

Result Grid | Filter Rows:

UserID	EarliestLoginTime
1	2024-04-20 10:30:00
2	2024-04-22 14:45:00
3	2024-04-23 09:15:00
4	2024-04-24 11:20:00
5	2024-04-25 08:45:00
6	2024-04-26 10:00:00
7	2024-04-27 12:00:00

7.Join with the Conversation table using the earliest login time

```
SELECT
    c.ConversationID,
    c.UserID,
    u.Username,
    c.StartTime,
    c.EndTime
```

```
FROM Conversation c
JOIN (
    -- Subquery to get the earliest login time for each user
    SELECT
        UserID,
        MIN(LastLoginDate) AS EarliestLoginTime
    FROM User
    GROUP BY UserID
) AS EarliestLogin ON c.UserID = EarliestLogin.UserID
JOIN User u ON c.UserID = u.UserID;
```

Results;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

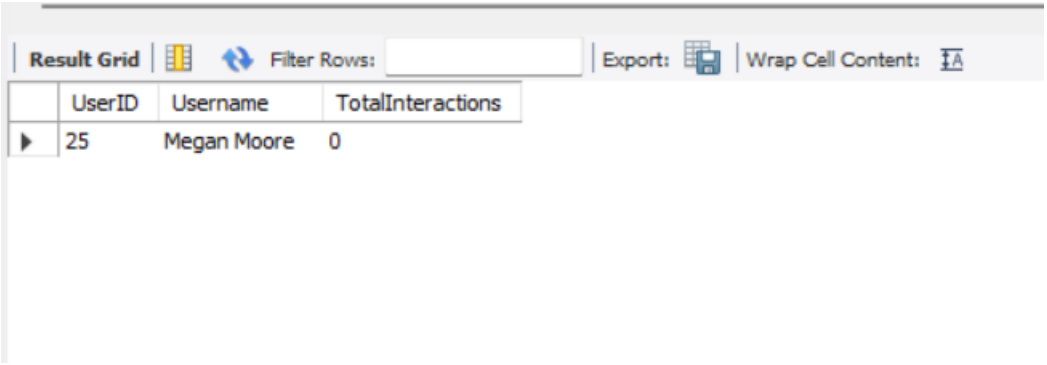
ConversationID	UserID	Username	StartTime	EndTime
1	1	John Doe	2024-04-25 14:39:14	2024-04-25 14:39:14
4	1	John Doe	2024-01-01 08:00:00	2024-01-01 08:30:00
29	1	John Doe	2024-04-25 14:39:17	2024-04-25 14:48:12
60	1	John Doe	2024-04-25 14:39:17	2024-04-25 14:48:29
2	2	Jane Smith	2024-04-25 14:39:14	2024-04-25 14:39:14
5	2	Jane Smith	2024-01-05 10:00:00	2024-01-05 11:00:00
30	2	Jane Smith	2024-04-25 14:39:17	2024-04-25 14:48:12

8..Find the most active users (by total interactions) who joined within the last month:

Quary;

```
SELECT u.UserID, u.Username, COUNT(i.InteractionID) AS TotalInteractions
FROM User u
LEFT JOIN Interaction i ON u.UserID = i.UserID
WHERE u.RegistrationDate >= DATE_SUB(NOW(), INTERVAL 1 MONTH)
GROUP BY u.UserID, u.Username
ORDER BY TotalInteractions DESC;
```

Result:



UserID	Username	TotalInteractions
25	Megan Moore	0

9. Find users who have not interacted with any intents

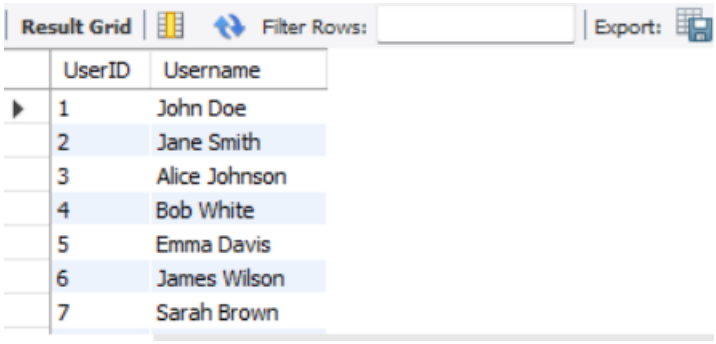
Quary:SELECT u.UserID, u.Username

FROM User u

LEFT JOIN Interaction i ON u.UserID = i.UserID

WHERE i.UserID IS NULL;

Result:



UserID	Username
1	John Doe
2	Jane Smith
3	Alice Johnson
4	Bob White
5	Emma Davis
6	James Wilson
7	Sarah Brown

10.Find users who have not interacted with any intents

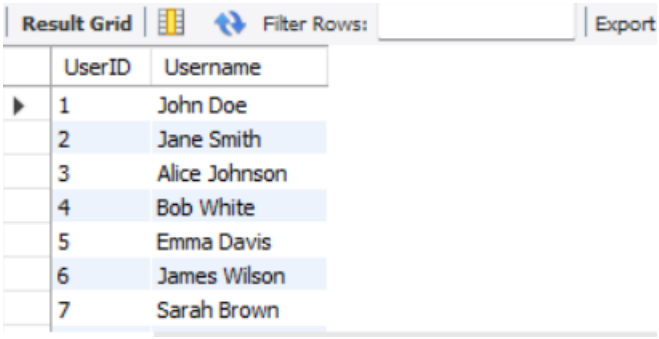
Quary:SELECT u.UserID, u.Username

FROM User u

LEFT JOIN Interaction i ON u.UserID = i.UserID

WHERE i.UserID IS NULL;

Results:



UserID	Username
1	John Doe
2	Jane Smith
3	Alice Johnson
4	Bob White
5	Emma Davis
6	James Wilson
7	Sarah Brown

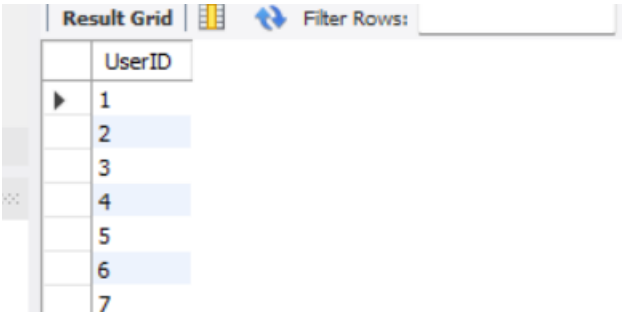
11. Find the unique set of users who have either started a conversation or interacted with an intent:

Quary:SELECT UserID FROM Conversation

UNION

SELECT UserID FROM Interaction;

Results:



UserID
1
2
3
4
5
6
7

Sub-query

1.Find users who have sent the most messages:

Query : SELECT Username, Email

FROM User

WHERE UserID = (

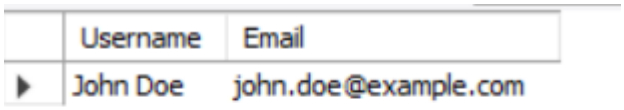
SELECT UserID

FROM (

SELECT UserID, COUNT(*) AS TotalMessages

```
        FROM Conversation
        JOIN Message ON Conversation.ConversationID = Message.ConversationID
        GROUP BY UserID
        ORDER BY TotalMessages DESC
        LIMIT 1
    ) AS SubQuery
);
```

Result:



	Username	Email
▶	John Doe	john.doe@example.com

2 . List conversations started by users who have registered within the last month:

Query :

```
SELECT ConversationID, StartTime
FROM Conversation
WHERE UserID IN (
    SELECT UserID
    FROM User
    WHERE RegistrationDate >= DATE_SUB(NOW(), INTERVAL 1 MONTH)
);
```

Results:

	ConversationID	StartTime
▶	1	2024-04-24 20:17:41
	4	2024-04-24 20:17:59
	2	2024-04-24 20:17:41
	5	2024-04-24 20:17:59
	3	2024-04-24 20:17:41
	6	2024-04-24 20:17:59
	7	2024-04-24 20:17:59

3. Find intents with more than 100 interactions:

Query :SELECT Name

```
FROM Intent
WHERE IntentID IN (
    SELECT IntentID
    FROM Interaction
    GROUP BY IntentID
    HAVING COUNT(*) > 20
);
```

Result:

	Name

4. List users who have never started a conversation:

Query :SELECT Username

```
FROM User
WHERE UserID NOT IN (
    SELECT DISTINCT UserID
```

```
    FROM Conversation
);
```

Results:

	Username
▶	Christopher Taylor
	Stephanie White
	Jonathan Martinez
	Taylor Lee

5 .Find the conversation with the most messages:

Query :

```
SELECT ConversationID
FROM Conversation
WHERE ConversationID = (
    SELECT ConversationID
    FROM (
        SELECT ConversationID, COUNT(*) AS TotalMessages
        FROM Message
        GROUP BY ConversationID
        ORDER BY TotalMessages DESC
        LIMIT 1
    ) AS SubQuery
);
```

Result:

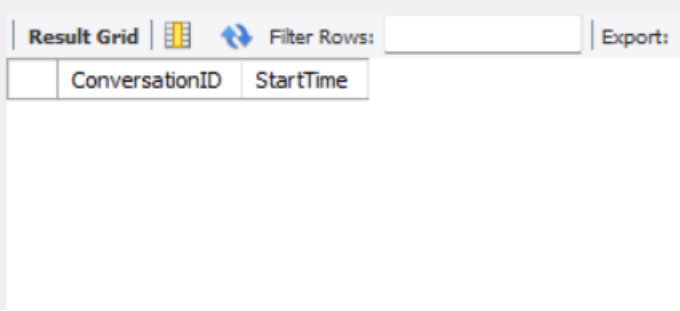
	ConversationID
▶	1
*	NULL

6.Find conversations started by users who have sent the most messages

Quary;SELECT c.ConversationID, c.StartTime

```
FROM Conversation c
INNER JOIN (
    SELECT UserID, COUNT(*) AS TotalMessages
    FROM Conversation
    JOIN Message ON Conversation.ConversationID = Message.ConversationID
    GROUP BY UserID
    ORDER BY TotalMessages DESC
    LIMIT 1
) AS SubQuery ON c.UserID = SubQuery.UserID;
```

Result;



Conclusion

AI chatbot system leverages SQL for database management, providing functionalities for user registration and authentication, conversation handling, message processing, intent detection, and interaction logging. Through the use of SQL commands, the system can store user information, track conversations, analyze messages, detect intents, and log interactions with associated confidence levels. With these capabilities, the system facilitates natural language interactions between users and the virtual assistant, enabling efficient communication and support. Overall, the project demonstrates the integration of SQL with AI technology to create an effective chatbot system for various applications.

Thank You