**CSA0593**

**Database management system**

K.SWETHA

192311404

ASSIGNMENT - 2

Develop a database for managing users, posts, comments, and messages.

- Model tables for users, posts, comments, and messages.

- Write stored procedures for posting, commenting, and messaging.

- Implement triggers to update user activity and post popularity.

- Write SQL queries to analyse user engagement and popular posts.

1). Here's a database structure and corresponding SQL implementation for managing users, posts, comments, and messages. This includes tables, stored procedures, triggers, and analytical queries to track user engagement and post popularity.

**Database Design**

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

last\_activity TIMESTAMP

);

CREATE TABLE Posts (

post\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

content TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

popularity\_score INT DEFAULT 0,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE

);

CREATE TABLE Comments (

comment\_id INT PRIMARY KEY AUTO\_INCREMENT,

post\_id INT NOT NULL,

user\_id INT NOT NULL,

content TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (post\_id) REFERENCES Posts(post\_id) ON DELETE CASCADE,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE

);

CREATE TABLE Messages (

message\_id INT PRIMARY KEY AUTO\_INCREMENT,

sender\_id INT NOT NULL,

receiver\_id INT NOT NULL,

content TEXT NOT NULL,

sent\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (sender\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (receiver\_id) REFERENCES Users(user\_id) ON DELETE CASCADE

);

**Stored Procedures:**

1. **Stored Procedure for Posting**

DELIMITER $$

CREATE PROCEDURE CreatePost (

IN p\_user\_id INT,

IN p\_content TEXT

)

BEGIN

INSERT INTO Posts (user\_id, content)

VALUES (p\_user\_id, p\_content);

-- Update user last activity

UPDATE Users

SET last\_activity = NOW()

WHERE user\_id = p\_user\_id;

END $$

DELIMITER ;

1. **Stored Procedure for Commenting:**

DELIMITER $$

CREATE PROCEDURE AddComment (

IN p\_post\_id INT,

IN p\_user\_id INT,

IN p\_content TEXT

)

BEGIN

INSERT INTO Comments (post\_id, user\_id, content)

VALUES (p\_post\_id, p\_user\_id, p\_content);

-- Update user last activity

UPDATE Users

SET last\_activity = NOW()

WHERE user\_id = p\_user\_id;

-- Increment post popularity score

UPDATE Posts

SET popularity\_score = popularity\_score + 1

WHERE post\_id = p\_post\_id;

END $$

DELIMITER ;

1. **Stored Procedure for Messaging:**

DELIMITER $$

CREATE PROCEDURE SendMessage (

IN p\_sender\_id INT,

IN p\_receiver\_id INT,

IN p\_content TEXT

)

BEGIN

INSERT INTO Messages (sender\_id, receiver\_id, content)

VALUES (p\_sender\_id, p\_receiver\_id, p\_content);

-- Update sender last activity

UPDATE Users

SET last\_activity = NOW()

WHERE user\_id = p\_sender\_id;

END $$

DELIMITER ;

**Triggers:-**

**1. Trigger to Update User Last Activity on Posting, Commenting, or Messaging**

This trigger automatically updates the user’s last activity timestamp whenever they create a new post, comment, or send a message.

DELIMITER $$

CREATE TRIGGER UpdateUserLastActivity

AFTER INSERT ON Posts

FOR EACH ROW

BEGIN

UPDATE Users

SET last\_activity = NOW()

WHERE user\_id = NEW.user\_id;

END $$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER UpdateUserLastActivityOnComment

AFTER INSERT ON Comments

FOR EACH ROW

BEGIN

UPDATE Users

SET last\_activity = NOW()

WHERE user\_id = NEW.user\_id;

END $$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER UpdateUserLastActivityOnMessage

AFTER INSERT ON Messages

FOR EACH ROW

BEGIN

UPDATE Users

SET last\_activity = NOW()

WHERE user\_id = NEW.sender\_id;

END $$

DELIMITER ;

2. **Trigger to Update Post Popularity on Comments**

DELIMITER $$

CREATE TRIGGER UpdatePostPopularity

AFTER INSERT ON Comments

FOR EACH ROW

BEGIN

UPDATE Posts

SET popularity\_score = popularity\_score + 1

WHERE post\_id = NEW.post\_id;

END $$

DELIMITER ;

**SQL Queries**

1).

SELECT post\_id, user\_id, content, popularity\_score

FROM Posts

ORDER BY popularity\_score DESC

LIMIT 5;

2).

SELECT user\_id, username, last\_activity

FROM Users

ORDER BY last\_activity DESC

LIMIT 10;

3).

SELECT p.post\_id, p.content, COUNT(c.comment\_id) AS comment\_count

FROM Posts p

LEFT JOIN Comments c ON p.post\_id = c.post\_id

GROUP BY p.post\_id

ORDER BY comment\_count DESC;

4).

(SELECT COUNT(\*) FROM Posts WHERE user\_id = u.user\_id) AS total\_posts,

(SELECT COUNT(\*) FROM Comments WHERE user\_id = u.user\_id) AS total\_comments,

(SELECT COUNT(\*) FROM Messages WHERE sender\_id = u.user\_id) AS total\_messages\_sent

FROM Users u

ORDER BY total\_posts + total\_comments + total\_messages\_sent DESC;

5).

SELECT DATE(created\_at) AS activity\_date,

(SELECT COUNT(\*) FROM Posts WHERE DATE(created\_at) = activity\_date) AS daily\_posts,

(SELECT COUNT(\*) FROM Comments WHERE DATE(created\_at) = activity\_date) AS daily\_comments,

(SELECT COUNT(\*) FROM Messages WHERE DATE(sent\_at) = activity\_date) AS daily\_messages

FROM Posts

GROUP BY activity\_date

ORDER BY activity\_date DESC;

**Conclusion:-**

In conclusion, this database design and implementation offers a comprehensive solution for managing a social media or community platform that handles users, posts, comments, and messages. The structured tables establish clear relationships among users, posts, comments, and messages, ensuring data consistency and integrity. Stored procedures streamline core actions such as posting, commenting, and messaging, while triggers automate real-time updates to user activity and post popularity, promoting timely and accurate data management.

The analytical queries enable in-depth insights into user engagement and popular content, supporting data-driven decisions to enhance user experience. By leveraging this design, platform administrators can easily monitor user activity, identify trending posts, and gauge overall engagement, all of which contribute to a more engaging and responsive platform for users.