# Social Media Database - Project Report

## 1. Introduction

Social Media platforms play a vital role in communication and content sharing in the digital era. Managing users, posts, likes, comments, and followers requires efficient data storage and retrieval. A relational database is the best way to implement such a system, ensuring data consistency, referential integrity, and support for advanced SQL features like constraints, triggers, and stored procedures.

## 2. Objectives

- To maintain user details such as username, email, password, and profile bio.  
- To store posts created by users along with timestamps.  
- To manage likes, comments, and followers relationships efficiently.  
- To implement CRUD operations (Create, Read, Update, Delete).  
- To apply advanced SQL features like aggregate functions, group by, having, subqueries, stored procedures, and triggers.  
- To ensure referential integrity using foreign keys and constraints.

## 3. System Requirements

Software Requirements

- Database: MySQL  
- Operating System: Windows/Linux  
- Tools Used: MySQL Workbench or phpMyAdmin

Hardware Requirements

- Processor: Intel i3 or higher  
- RAM: 4GB minimum  
- Storage: 10GB free space

## 4. Database Design

The system contains the following tables:  
  
1. Users – stores details of registered users.  
2. Posts – stores user posts.  
3. Likes – maintains information about likes on posts.  
4. Comments – stores comments on posts.  
5. Followers – maintains follower-following relationships.  
6. Post Logs – logs activities performed on posts (via triggers).

## 5. SQL Implementation

### 5.1 Create Database and Tables

CREATE DATABASE project;  
USE project;  
  
-- Creating Users Table --  
CREATE TABLE users (  
 user\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 username VARCHAR(50) UNIQUE NOT NULL,  
 email VARCHAR(100) UNIQUE NOT NULL,  
 password VARCHAR(100) NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
ALTER TABLE users ADD bio VARCHAR(255);  
  
-- Creating Posts Table --  
CREATE TABLE posts (  
 post\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 user\_id INT,  
 content TEXT NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE  
);  
  
-- Creating Likes Table --  
CREATE TABLE likes (  
 like\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 user\_id INT,  
 post\_id INT,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE,  
 FOREIGN KEY (post\_id) REFERENCES posts(post\_id) ON DELETE CASCADE  
);  
  
-- Creating Comments Table --  
CREATE TABLE comments (  
 comment\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 post\_id INT,  
 user\_id INT,  
 comment\_text 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  
);  
  
-- Creating Followers Table --  
CREATE TABLE followers (  
 follower\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 user\_id INT,  
 follower\_user\_id INT,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (user\_id) REFERENCES users(user\_id) ON DELETE CASCADE,  
 FOREIGN KEY (follower\_user\_id) REFERENCES users(user\_id) ON DELETE CASCADE  
);

### 5.2 Insert Sample Records

-- Insert Users --  
INSERT INTO users (username, email, password) VALUES  
('harshi\_1070', 'harshi@gmail.com', 'pass123'),  
('kishore\_1510', 'kishore@gmail.com', 'pass456'),  
('sumanth07', 'sumanth@gmail.com', 'pass789');  
  
-- Insert Posts --  
INSERT INTO posts (user\_id, content) VALUES  
(1, 'Hello world! This is my first post.'),  
(2, 'Good morning everyone!'),  
(3, 'Enjoying a great day!');  
  
-- Update Post --  
UPDATE posts SET content = 'Hello Everyone!!' WHERE post\_id = 1;  
  
-- Insert Likes --  
INSERT INTO likes (user\_id, post\_id) VALUES (2, 1), (3, 1), (1, 2);  
  
-- Insert Comments --  
INSERT INTO comments (post\_id, user\_id, comment\_text) VALUES  
(1, 2, 'Nice post!'),  
(1, 3, 'Welcome Alice!'),  
(2, 1, 'Good morning Bob!');  
  
-- Delete Comment --  
DELETE FROM comments WHERE comment\_id = 3;  
  
-- Insert Followers --  
INSERT INTO followers (user\_id, follower\_user\_id) VALUES  
(1, 2),  
(1, 3),  
(2, 1);

### 5.3 Sample Queries

-- Get Posts by User --  
SELECT \* FROM posts WHERE user\_id = 1;  
  
-- Aggregate Functions --  
SELECT COUNT(\*) AS total\_posts FROM posts;  
  
-- Number of Likes per Post --  
SELECT post\_id, COUNT(\*) AS total\_likes FROM likes GROUP BY post\_id;  
  
-- Having Clause Example --  
SELECT user\_id, COUNT(\*) AS total\_followers FROM followers GROUP BY user\_id HAVING COUNT(\*) > 1;  
  
-- Username Starting with 'har' --  
SELECT \* FROM users WHERE username LIKE 'har%';  
  
-- Users Who Liked Post 1 --  
SELECT username FROM users WHERE user\_id IN (SELECT user\_id FROM likes WHERE post\_id = 1);

### 5.4 Stored Procedure

DELIMITER //  
CREATE PROCEDURE GetUserPosts(IN uid INT)  
BEGIN  
 SELECT \* FROM posts WHERE user\_id = uid;  
END //  
DELIMITER ;  
  
CALL GetUserPosts(1);

### 5.5 Trigger

CREATE TABLE post\_logs (  
 log\_id INT AUTO\_INCREMENT PRIMARY KEY,  
 post\_id INT,  
 action VARCHAR(50),  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
  
DELIMITER //  
CREATE TRIGGER after\_post\_insert  
AFTER INSERT ON posts  
FOR EACH ROW  
BEGIN  
 INSERT INTO post\_logs (post\_id, action) VALUES (NEW.post\_id, 'New Post Created');  
END //  
DELIMITER ;  
  
INSERT INTO posts (user\_id, content) VALUES (1, 'Trigger test post!');

## 6. Results

- Successfully created and managed a Social Media Database.  
- Demonstrated CRUD operations for users, posts, likes, comments, and followers.  
- Applied aggregate functions and advanced SQL features.  
- Implemented stored procedures for modular queries.  
- Created triggers to automatically log post insertions.  
- Ensured referential integrity using foreign keys and constraints.

## 7. Conclusion

The Social Media Database project provides an efficient way to handle users, posts, likes, comments, and followers. It demonstrates the use of relational integrity, normalization, and advanced SQL features like stored procedures and triggers. This project can be extended to include features like direct messaging, story sharing, or integration with a web/mobile application.