Q.3) **Social Media Platform Database System**

This database is designed for managing user profiles, posts, likes, comments, and follower relationships on a social media platform. It will help manage interactions like posting, commenting, liking, and following other users, while also tracking engagement metrics, notifications, and activity feeds

### 1. Database Tables Design

**Table: Users Table**

**This table stores user profile information, ensuring unique usernames and managing personal details.**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| user\_id | INT (PK) | Unique identifier for each user |
| username | VARCHAR(50) | User's unique username (unique constraint) |
| email | VARCHAR(100) | User's email address |
| first\_name | VARCHAR(50) | User's first name |
| last\_name | VARCHAR(50) | User's last name |
| profile\_picture | VARCHAR(255) | URL of the user’s profile picture |
| bio | VARCHAR(255) | Short bio or description provided by the user |
| date\_joined | DATE | Date the user joined the platform |

**Table: Posts Table**

**This table stores posts created by users, including the content and timestamps.**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| post\_id | INT (PK) | Unique identifier for each post |
| user\_id | INT (FK) | References Users(user\_id) |
| content | TEXT | Content of the post (can include text, media links) |
| created\_at | TIMESTAMP | Timestamp of when the post was created |

**Table: Comments Table**

**This table manages comments on posts, allowing users to interact with posts through text-based responses.**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| comment\_id | INT (PK) | Unique identifier for each comment |
| post\_id | INT (FK) | References Posts(post\_id) |
| user\_id | INT (FK) | References Users(user\_id) |
| comment\_text | TEXT | The text content of the comment |
| created\_at | TIMESTAMP | Timestamp of when the comment was created |

**Table: Likes Table**

**This table tracks likes on posts and comments, associating users with the content they’ve liked.**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| like\_id | INT (PK) | Unique identifier for each like |
| post\_id | INT (FK) | References Posts(post\_id) |
| user\_id | INT (FK) | References Users(user\_id) |
| created\_at | TIMESTAMP | Timestamp when the like was created |

**Table: Followers Table**

**This table manages the following relationships between users, ensuring that users can follow others but not themselves.**

| **Column** | **Data Type** | **Description** |
| --- | --- | --- |
| follower\_id | INT (FK) | References Users(user\_id) (follower) |
| following\_id | INT (FK) | References Users(user\_id) (followed user) |
| created\_at | TIMESTAMP | Timestamp when the follow relationship was created |

### 2. Constraints for Referential Integrity

### **Foreign Keys:**

### **user\_id in Posts, Comments, Likes references Users(user\_id).**

### **post\_id in Comments and Likes references Posts(post\_id).**

### **follower\_id and following\_id in Followers reference Users(user\_id).**

### **Primary Keys:**

### **user\_id in Users table.**

### **post\_id in Posts table.**

### **comment\_id in Comments table.**

### **like\_id in Likes table.**

### **Composite primary key on follower\_id and following\_id in Followers table.**

### **Check Constraints:**

### **Ensure follower\_id ≠ following\_id to prevent self-following in the Followers table.**

### 3. Stored Procedures

#### a. Post Content

**This procedure allows a user to create a post on the platform.**

#### CREATE PROCEDURE CreatePost(IN userId INT, IN postContent TEXT)

#### BEGIN

#### INSERT INTO Posts (user\_id, content, created\_at)

#### VALUES (userId, postContent, NOW());

#### END;

#### b. Like a Post

**This procedure allows a user to like a post.**

#### CREATE PROCEDURE LikePost(IN userId INT, IN postId INT)

#### BEGIN

#### INSERT INTO Likes (user\_id, post\_id, created\_at)

#### VALUES (userId, postId, NOW());

#### END;

#### c. Comment on Post

**This procedure allows a user to comment on a post.**

### CREATE PROCEDURE CommentOnPost(IN userId INT, IN postId INT, IN commentText TEXT)

### BEGIN

### INSERT INTO Comments (user\_id, post\_id, comment\_text, created\_at)

### VALUES (userId, postId, commentText, NOW());

### END;

d. Follow User

This procedure allows a user to follow another user.

CREATE PROCEDURE FollowUser(IN followerId INT, IN followingId INT)

BEGIN

IF followerId != followingId THEN

INSERT INTO Followers (follower\_id, following\_id, created\_at)

VALUES (followerId, followingId, NOW());

ELSE

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Cannot follow yourself.';

END IF;

END;

### 4. Triggers

#### a. Trigger for Post Likes (Update Like Count)

**This trigger ensures that every time a post is liked, the like count for the post is updated.**

#### CREATE TRIGGER AfterPostLikeInsert

#### AFTER INSERT ON Likes

#### FOR EACH ROW

#### BEGIN

#### DECLARE newLikeCount INT;

#### SELECT COUNT(\*) INTO newLikeCount FROM Likes WHERE post\_id = NEW.post\_id;

#### UPDATE Posts SET like\_count = newLikeCount WHERE post\_id = NEW.post\_id;

#### END;

#### b. Trigger for Commenting (Update Comment Count)

**This trigger ensures that every time a comment is added to a post, the comment count for the post is updated.**

CREATE TRIGGER AfterCommentInsert

AFTER INSERT ON Comments

FOR EACH ROW

BEGIN

DECLARE newCommentCount INT;

SELECT COUNT(\*) INTO newCommentCount FROM Comments WHERE post\_id = NEW.post\_id;

UPDATE Posts SET comment\_count = newCommentCount WHERE post\_id = NEW.post\_id;

END;

#### c. Trigger for Following (Update Followers Count)

#### CREATE TRIGGER AfterFollowInsert

#### AFTER INSERT ON Followers

#### FOR EACH ROW

#### BEGIN

#### DECLARE newFollowerCount INT;

#### SELECT COUNT(\*) INTO newFollowerCount FROM Followers WHERE following\_id = NEW.following\_id;

#### UPDATE Users SET follower\_count = newFollowerCount WHERE user\_id = NEW.following\_id;

#### END;

### 5. SQL Queries for Reports

#### a. Report on User Engagement

#### SELECT u.username,

#### COUNT(p.post\_id) AS total\_posts,

#### COUNT(l.like\_id) AS total\_likes,

#### COUNT(c.comment\_id) AS total\_comments

#### FROM Users u

#### LEFT JOIN Posts p ON u.user\_id = p.user\_id

#### LEFT JOIN Likes l ON p.post\_id = l.post\_id

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

#### GROUP BY u.user\_id

#### ORDER BY total\_likes DESC;

#### b. Report on Post Popularity

#### SELECT p.post\_id, p.content, COUNT(l.like\_id) AS like\_count

#### FROM Posts p

#### LEFT JOIN Likes l ON p.post\_id = l.post\_id GROUP BY p.post\_id

#### C. Report on Most-Followed Users

SELECT u.username, COUNT(f.follower\_id) AS followers\_count

FROM Users u

LEFT JOIN Followers f ON u.user\_id = f.following\_id

GROUP BY u.user\_id

ORDER BY followers\_count DESC;

**Conclusion**

**This Social Media Platform Database System allows the management of users, posts, comments, likes, and followers. It incorporates essential features like:**

* **User profile management**
* **Interactions like posts, comments**