Social Media Platform Database Documentation – Adrian Flores

1. Database Schema

Users

Field	Data Type	Description
user_id	INT (PK)	Unique user identifier
username	VARCHAR(50)	Username
email	VARCHAR(100)	Email address
password	VARCHAR(255)	Password (hashed)
date_of_birth	DATE	Date of birth
profile_picture	VARCHAR(255)	Profile picture URL
post_count	INT	Number of posts (via trigger)
follower_count	INT	Number of followers (via trigger)

Posts

Field	Data Type	Description
post_id	INT (PK)	Unique post identifier
user_id	INT (FK)	User who created the post
post_text	TEXT	Post content
post_date	DATETIME	Date and time of posting
media_url	VARCHAR(255)	Optional media attachment

Comments

Field	Data Type	Description
comment_id	INT (PK)	Unique comment ID
post_id	INT (FK)	Post being commented on
user_id	INT (FK)	User who made the comment
comment_text	TEXT	Text of the comment
comment_date	DATETIME	Date and time of the comment

Likes

Field	Data Type	Description
like_id	INT (PK)	Unique like ID
post_id	INT (FK)	Liked post ID
user_id	INT (FK)	User who liked the post
like_date	DATETIME	Date and time of the like

Follows

Field	Data Type	Description
follower_id	INT (FK)	User who follows
following_id	INT (FK)	User being followed
follow_date	DATETIME	Date of the follow

Messages

Field	Data Type	Description
message_id	INT (PK)	Message ID
sender_id	INT (FK)	Sender user ID
receiver_id	INT (FK)	Receiver user ID
message_text	TEXT	Message content
message_date	DATETIME	Date and time of the message
is_read	BOOLEAN	Read/unread status

Notifications

Field	Data Type	Description
notification_id	**INT (PK)**	**Notification ID**
user_id	**INT (FK)**	**User receiving the notification**
notification_text	**TEXT**	**Content of the notification**
notification_date	**DATETIME**	**Date and time of the notification**
is_read	**BOOLEAN**	**Read/unread status**

2. Sample Data

Table	Code
Users	**INSERT INTO Users (username, email, password, date_of_birth, profile_picture) VALUES ('alice', 'alice@example.com', 'passAlice123', '1990-06-01', 'alice.jpg');**

Table	Code
Posts	INSERT INTO Posts (user_id, post_text, post_date, media_url) VALUES (4, 'Just had a great lunch.', '2024-12-14 17:21:44', 'media1.jpg');

Table	Code
Comments	INSERT INTO Comments (post_id, user_id, comment_text, comment_date) VALUES (7, 1, 'Nice post!', '2023-11-29 02:26:00');

Table	Code
Likes	INSERT INTO Likes (post_id, user_id, like_date) VALUES (8, 1, '2023-03-11 14:42:24');

Table	Code
Follows	INSERT INTO Follows (follower_id, following_id, follow_date) VALUES (4, 3, '2023-03-07 03:31:21');

Table	Code
Messages	INSERT INTO Messages (sender_id, receiver_id, message_text, message_date, is_read) VALUES (3, 2, 'Hello!', '2024-02-11 16:55:01', 0);

Table	Code
Notifications	INSERT INTO Notifications (user_id, notification_text, notification_date, is_read) VALUES (3, 'You have a new follower.', '2023-12-09 15:54:19', 1);

3. SQL Query Explanations

Query	Code	Explanation
User Timeline	SELECT p.post_id, u.username, p.post_text, p.post_date, p.media_url FROM Posts p JOIN Users u ON p.user_id = u.user_id WHERE p.user_id = 1 OR p.user_id IN (SELECT following_id FROM Follows WHERE follower_id = 1) ORDER BY p.post_date DESC;	**Retrieve the posts and activities of a user's timeline**
Post Engagement	SELECT c.comment_id, u.username, c.comment_text, c.comment_date FROM Comments c JOIN Users u ON c.user_id = u.user_id WHERE c.post_id = 9 ORDER BY c.comment_date ASC; SELECT l.like_id, u.username, l.like_date FROM Likes l JOIN Users u ON l.user_id = u.user_id WHERE l.post_id = 2;	**Retrieve the comments and likes for a specific post**
Follower List	SELECT f.follower_id, u.username, f.follow_date FROM Follows f JOIN Users u ON f.follower_id = u.user_id WHERE f.following_id = 1;	**List of all users following a specific user.**
	SELECT m.message_id, u.username AS sender, m.message_text,	

Unread Messages	m.message_date FROM Messages m JOIN Users u ON m.sender_id = u.user_id WHERE m.receiver_id = 1 AND m.is_read = FALSE ORDER BY m.message_date DESC;	**Fetch all unread messages for a given user.**
Top Posts	SELECT p.post_id, u.username, p.post_text, COUNT(l.like_id) AS like_count FROM Posts p JOIN Users u ON p.user_id = u.user_id LEFT JOIN Likes l ON p.post_id = l.post_id GROUP BY p.post_id, u.username, p.post_text ORDER BY like_count DESC LIMIT 10;	**Posts with the highest like count.**
Recent Notifications:	SELECT notification_id, notification_text, notification_date, is_read FROM Notifications WHERE user_id = 1 ORDER BY notification_date DESC LIMIT 10;	**Latest notifications per user, ordered by date.**

4. Triggers and Stored Procedures

Triggers and Stored Procedures	Code	Explanation

after_new_message	DELIMITER $$
	CREATE TRIGGER after_new_message
	AFTER INSERT ON Messages
	FOR EACH ROW
	BEGIN
	  INSERT INTO Notifications (user_id, notification_text, notification_date, is_read)
	  VALUES (
	  NEW.receiver_id,
	  CONCAT('You received a new message from user ID ', NEW.sender_id),
	  NOW(),
	  FALSE
	  );
	END$$
	DELIMITER ;
		Creates a notification when a message is sent.

after_new_follow	DELIMITER $$
	CREATE TRIGGER after_new_follow
	AFTER INSERT ON Follows
	FOR EACH ROW
	BEGIN
	  UPDATE Users
	  SET follower_count = follower_count + 1
	  WHERE user_id = NEW.following_id;
	END$$
	DELIMITER ;
		Automatically updates follower_count .

after_new_post	DELIMITER $$
	CREATE TRIGGER after_new_post
	AFTER INSERT ON Posts
	FOR EACH ROW
	BEGIN
	  UPDATE Users
	  SET post_count = post_count + 1
	  WHERE user_id = NEW.user_id;
		Automatically updates post_count in the Users table.

GetFollowRecommendations (userId)	DELIMITER $$
	CREATE PROCEDURE GetFollowRecommendations(IN current_user_id INST)
	BEGIN
	  SELECT u.user_id, u.username, u.follower_count
	  FROM Users u
	  WHERE u.user_id != current_user_id
	  AND u.user_id NOT IN (
	  SELECT following_id
	  FROM Follows
	  WHERE follower_id = current_user_id
	  )
	  ORDER BY u.follower_count DESC
	  LIMIT 5;
	END$$
	DELIMITER ;
		Suggests popular users the current user is not following yet.