Metrics and Dashboard

Informatics for Social Media

Mini Project Report -Database Lab (DSE 2241)

Department of Data Science & Computer Applications

B. Tech Data Science

4th Semester – Batch 3 - Group: 4

Shravani	
Bhavyaa Goyal	
Avadh Gandhi	
Ganesh Chaudhari	
Mitwa Saraf	

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Date: 16 April 2024

CERTIFICATE

This is to certify that Shravani, Bhavyaa Goyal, Avadh Gandhi, Ganesh Chaudhari and Mitwa Saraf, have successfully executed a mini project titled “Metrics and Dashboard Informatics for Social Media” rightly bringing for the competencies and skill sets they have gained during the course- Database Lab (DSE 2241), thereby resulting in the culmination of this project.

ABSTRACT

Social media platforms have become integral parts of modern society, facilitating communication, information sharing, and collaboration on an unprecedented scale. Managing the vast amount of user-generated content and interactions on these platforms presents significant challenges. This project addresses these challenges by developing a comprehensive social media management system. The methodology involves designing and implementing a relational database schema using SQL to efficiently store and manage various types of social media data. A backend server application is created to handle HTTP requests from the frontend, enabling seamless interaction with the database. JavaScript is utilized on the frontend to facilitate dynamic user interface updates through asynchronous requests to the backend.

The implemented system enables users to register, create posts, interact through comments, and likes, and engage with others effectively. It provides a user-friendly interface for content management while offering insights into user interactions for informed decision-making. This system streamlines social media management efforts, enhances user engagement, and supports data-driven strategies for organizations and individuals.

In conclusion, the developed social media management system offers an effective solution for navigating the complexities of social media content management. By leveraging relational databases and modern web technologies, it empowers users to harness the full potential of social media platforms for communication and collaboration.

Content

Chapter 1

Introduction

Social media is undoubtably an indispensable part of the modern society. Most young adults stay in touch with peers directly or indirectly via social media. As out DBS project, we plan to replicate the functions of this network of digital societies in our own accustomed yet straightforward rendition of a social media database as a dashboard.

The social media database has the potential of carrying billions of accounts on the servers thus the database must be robust. Handling queries as fast as possible in order to retrieve data should be of utmost importance.

This database also needs to be personalized yet generalized, thus being versatile is very vital to this concept. Our Project involves Tables containing profile info (i.e. the data necessary in order to register a person on the network), Account info (i.e. information about the account holder that may/may not be open to the world but shares personal side of the user), messaging system data, notification control, etc.

Chapter 2

Synopsis

2.1 Proposed System

This project aims to provide a structural insight of what goes into making the backend and database for a primarily functionating social media. We have a table that can be used to keep track of users using unique usernames made using alphanumeric combination of first and last names and a number. Filing the posts, followers, followings, comments, likes and mapping all attributes from user A to B is an essential task of our system.

2.2 Objectives

Main objectives of our project are:

- To store essential user details required for registration and verification of user on the network.
- To record account details and type of account.
- To map relations between users based on likes, comments, followers and followings.
- To keep track of posts by a user.
- To tabulate the exchange of messages between two users using unique usernames.
- To have backup of user credentials on the social media platform.

Chapter 3

Functional Requirements

3.1 Likes and Comments counter

We have used PL/SQL triggers that are responsible to increment the like and comment count every time a post is liked by a new user or a new comment is made on the post.

Table 3.1 Count triggers

Action	**Process**	**Result**
Post liked	Set variable like_count to like_count +1	Like_count incremented
Comment added	Set variable comment_count to comment_count +1	Comment_count incremented

3.2 Backup trigger

In order to save user data, we backup the username and password on the database. We use triggers to identify when a new username password pair is updated, and back it up.

Table 3.2 Backup

Action	**Process**	**Result**
New username created	Save the username to backup table	Username backed up
Password for username created	Save the password to the backup table	Respective password backed up

3.3 Total Interactions

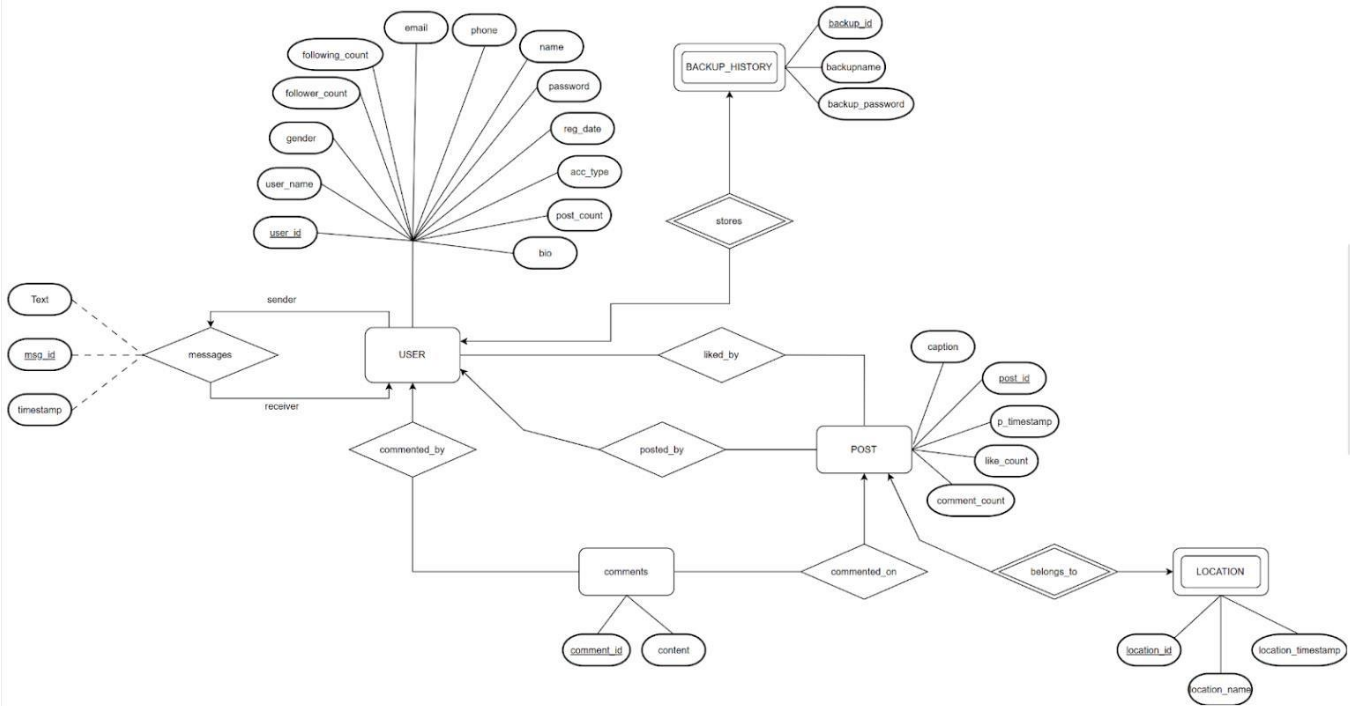
A function to return the total interactions that a post gets using functions in PL/SQL. Post interaction is the total number of likes+ total number of comments on a post.

Table 3.3 Total interactions

Input	**Output**	**Function**
Post_id	Total interaction	Reports total likes and comments on a post

Chapter 4: Detailed Design

4.1 ER Diagram



4.2 Schema diagram

User (user_id, username, name, password, bio, email, phone, gender, reg_date, post_count, followers_count, following_count, acc_type)

Posts (post_id, like_count, comment_count, caption, image_url, p_timestamp, user_id)

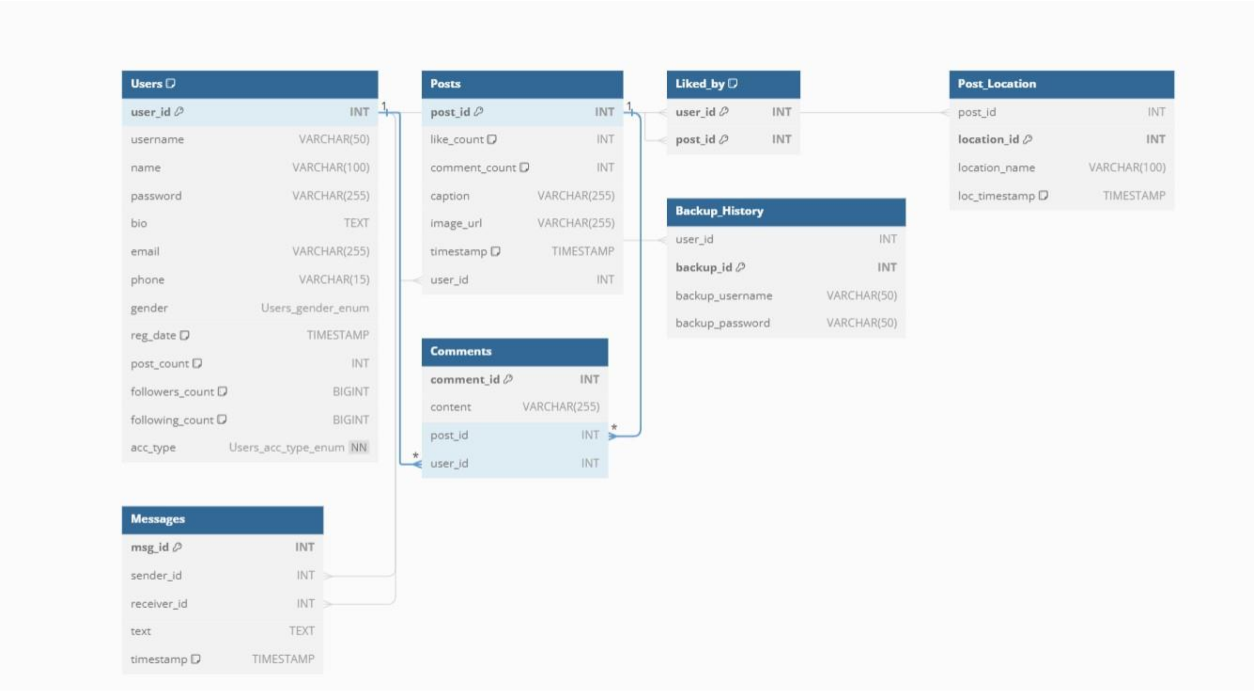
Liked_by (user_id, post_id)

Post_location (location_id, location_name, loc_timestamp, post_id)

Comments (comment_id, content, post_id, user_id)

Backup_history (backup_id, backup_username, backup_password, user_id)

Messages (msg_id , sender_id, reciever_id, text, timestamp)



4.3 Data Dictionary

Users

Column	Data type	Constraint	Constraint name
User_id	Int	Auto increment, Priamary key	
username	Varchar(50)	Unique	
Name	Varchar(100)		
Password	Varchar(50)	Not null	
bio	Text		
Email	Varchar(100)	Unique, Valid only if contains ‘%@gmail.com'	unique_email
Phone	Varchar(10)	Length should be 10 and REGEXP '^[0-9]+$'	valid_phone
Gender	Enum	Valid values- Male, Female, Other	
Reg_date	Timestamp	Default- current timestamp	
Post_count	Int	Default 0	
Followers_count	Bigint	Default 0	
Following_count	Bigint	Default 0	
Acc_type	Enum	Valid values- personal, business, Not Null	

Posts

Column	Data type	Constraint	Constraint name
Post_id	Int	Auto increment	
Like_count	Int	Default 0	
Comment_count	Int	Default 0	
Caption	Varchar(255)		
Image_url	Varchar(255)		
Timestamp	Timestamp	Default current_timestamp	
User_id	Int	Foreign key	fk_posts_user_id

Liked_by

Column	Data type	Constraint	Constraint name
User_id	Int	Primary key, foreign key	fk_liked_by_user_id
Post_id	Int	Primary key, foreign key	fk_liked_by_post_id

Post_Location

Column	Data type	Constraint	Constraint name
Post_id	Int	foreign key	fk_post_location_post_id
location_id	Int	Auto increment, primary key	
location_name	Varchar(100)		
loc_timestamp	Timestamp	Default current timestamp	

Messages

Column	Data type	Constraint	Constraint name
Msg_id	Int	Auto increment, uto increment	
Sender_id	Int	Foreign key	
Receiver_id	Int	Foreign key	
Text	Text		
Timestamp	Timestamp	Default current timestamp	

Comments

Column	Data type	Constraint	Constraint name
Comment_id	Int	Auto increment, Primary key	
Content	Varchar(255)		

Post_id	Int	Foreign key	fk_comments_post_id
User_id	Int	Foreign key	fk_comments_user_id

Backup_history

Column	Data type	Constraint	Constraint name
user_id	Int	Foreign key	
Backup_id	Int	Auto increment, Primary key	
Backup_username	Varchar(50)		
Backup_password	Varchar(50)		

4.4 Relational Model Implementation

```
CREATE TABLE Users (
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(50) UNIQUE,
   name VARCHAR(100),
   password VARCHAR(255),
   bio TEXT,
   email VARCHAR(255) UNIQUE CONSTRAINT unique_email CHECK (email LIKE '%@gmail.com'),
   phone VARCHAR(15) CONSTRAINT valid_phone CHECK (LENGTH(phone) = 10),
   gender ENUM('Male', 'Female', 'Other'),
   reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   post_count INT UNSIGNED DEFAULT 0,
   followers_count BIGINT UNSIGNED DEFAULT 0,
   following_count BIGINT UNSIGNED DEFAULT 0,
   acc_type ENUM('personal', 'business') NOT NULL,
   CONSTRAINT unique_email UNIQUE (email),
   CONSTRAINT valid_phone UNIQUE (phone)
);
CREATE TABLE Posts (
   post_id INT AUTO_INCREMENT,
   like_count INT DEFAULT 0,
   comment_count INT DEFAULT 0,
   caption VARCHAR(255),
   image_url VARCHAR(255),
   timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   user_id INT,
   PRIMARY KEY (post_id),
   CONSTRAINT fk_posts_user_id FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
CREATE TABLE Liked_by (
   user_id INT,
    post_id INT,
    PRIMARY KEY (user_id, post_id),
    CONSTRAINT fk_liked_by_user_id FOREIGN KEY (user_id) REFERENCES
Users(user_id),
    CONSTRAINT fk_liked_by_post_id FOREIGN KEY (post_id) REFERENCES
Posts(post_id)
);
CREATE TABLE Post_Location (
    post_id INT,
    location_id INT AUTO_INCREMENT,
    location_name VARCHAR(100),
    loc_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (location_id),
    CONSTRAINT fk_post_location_post_id FOREIGN KEY (post_id) REFERENCES
Posts(post_id)
);
CREATE TABLE Messages (
    msg_id INT AUTO_INCREMENT PRIMARY KEY,
    sender_id INT,
    receiver_id INT,
    text TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (sender_id) REFERENCES Users(user_id),
    FOREIGN KEY (receiver_id) REFERENCES Users(user_id)
);
CREATE TABLE Comments (
    comment_id INT AUTO_INCREMENT,
    content VARCHAR(255),
    post_id INT,
    user_id INT,
    PRIMARY KEY (comment_id),
    CONSTRAINT fk_comments_post_id FOREIGN KEY (post_id) REFERENCES
Posts(post_id),
    CONSTRAINT fk_comments_user_id FOREIGN KEY (user_id) REFERENCES
Users(user_id)
);
CREATE TABLE Backup_History (
    user_id INT,
    backup_id INT AUTO_INCREMENT,
    backup_username VARCHAR(50),
    backup_password VARCHAR(50),
    PRIMARY KEY (backup_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);
```

Chapter 5: Implementation

5.1 Queries

5.1.1 Query to get the total number of likes for each post

```
SELECT p.post_id, p.caption, COUNT(l.post_id) AS total_likes
FROM Posts p
LEFT JOIN Liked_by l ON p.post_id = l.post_id
GROUP BY p.post_id, p.caption;
```

5.1.2 Query to get the average number of comments per user

```
SELECT u.user_id, u.username, COUNT(DISTINCT p.post_id) AS total_posts,
SUM(p.like_count) AS total_likes
FROM Users u
LEFT JOIN Posts p ON u.user_id = p.user_id
GROUP BY u.user_id, u.username;
```

5.1.3 Query to find the total number of posts made by each user along with their total number of likes:

```
SELECT u.user_id, u.username, COUNT(DISTINCT p.post_id) AS total_posts,
SUM(p.like_count) AS total_likes
FROM Users u
LEFT JOIN Posts p ON u.user_id = p.user_id
GROUP BY u.user_id, u.username;
```

5.1.4 Query to find the average number of comments per post

```
SELECT p.post_id, AVG(c.comment_count) AS avg_comments_per_post
FROM Posts p
LEFT JOIN (
    SELECT post_id, COUNT(*) AS comment_count
    FROM Comments
    GROUP BY post_id
) c ON p.post_id = c.post_id
GROUP BY p.post_id;
```

5.1.5 Query to find the user with the highest number of followers

```
SELECT user_id, username, followers_count

FROM Users

ORDER BY followers_count DESC

LIMIT 1;
```

5.1.6 Query to find the total number of likes received by each user

```
SELECT u.user_id, u.username, SUM(p.like_count) AS total_likes_received

FROM Users u

LEFT JOIN Posts p ON u.user_id = p.user_id

GROUP BY u.user_id, u.username;
```

5.1.7 Query to find users who have not made any posts

```
SELECT u.user_id, u.username

FROM Users u

LEFT JOIN Posts p ON u.user_id = p.user_id

WHERE p.post_id IS NULL;
```

5.2 Triggers

5.2.1 Trigger to increase the comment_count whenever someone inserts a value in the comment table

```
DELIMITER $$

CREATE TRIGGER increase_comment_count

AFTER INSERT ON Comments

FOR EACH ROW

BEGIN

   UPDATE Posts

   SET comment_count = comment_count + 1

   WHERE post_id = NEW.post_id;

END $$

DELIMITER;
```

5.2.2 Trigger to increase the like_count whenever someone inserts a value in the liked_by table

```
DELIMITER $$
CREATE TRIGGER increase_like_count
AFTER INSERT ON Liked_by
FOR EACH ROW
BEGIN
    UPDATE Posts
    SET like_count = like_count + 1
    WHERE post_id = NEW.post_id;
END $$
DELIMITER ;
```

5.2.3 Trigger to call the Procedure Backup_user_credentials

```
CREATE TRIGGER backup_details
BEFORE INSERT ON Users
FOR EACH ROW
BEGIN
    -- Check if the username and password fields are being assigned values
    IF NEW.username IS NOT NULL AND NEW.password IS NOT NULL THEN
        -- Call the procedure to backup username and password
        CALL Backup_User_Credentials(NEW.user_id, NEW.username, NEW.password);
    END IF;
END $$
```

5.2.4 Trigger to update post timestamp on comment insertion:

```
DELIMITER $$
CREATE TRIGGER update_post_timestamp
AFTER INSERT ON Comments
FOR EACH ROW
BEGIN
```

```
    UPDATE Posts
    SET timestamp = CURRENT_TIMESTAMP
    WHERE post_id = NEW.post_id;
END $$
DELIMITER ;
```

5.3 Stored Procedures

5.3.1 Procedure to store the username and password in the backup_history table automatically:

```
DELIMITER $$
CREATE PROCEDURE Backup_User_Credentials (IN new_user_id INT, IN new_username VARCHAR(50), IN new_password VARCHAR(50))
BEGIN
    -- Insert username and password into the Backup_History table
    INSERT INTO Backup_History (user_id, backup_username, backup_password)
    VALUES (new_user_id, new_username, new_password);
END $$
DELIMITER ;
```

5.3.2 Procedure to delete user and associated data:

```
DELIMITER $$
CREATE PROCEDURE Delete_User_And_Data (IN user_id_to_delete INT)
BEGIN
    -- Delete user's comments
    DELETE FROM Comments WHERE user_id = user_id_to_delete;
    -- Delete user's likes
    DELETE FROM Liked_by WHERE user_id = user_id_to_delete;
    -- Delete user's posts
    DELETE FROM Posts WHERE user_id = user_id_to_delete;
    -- Delete user's backup_history
    DELETE FROM backup_history WHERE user_id = user_id_to_delete;
    -- Delete user
    DELETE FROM Users WHERE user_id = user_id_to_delete;
END $$
DELIMITER ;
```

5.3.3 Procedure to fetch all posts by a user:

```
DELIMITER $$
CREATE PROCEDURE Fetch_All_User_Posts (IN user_id_to_fetch INT)
BEGIN
    SELECT *
    FROM Posts
    WHERE user_id = user_id_to_fetch;
END $$
DELIMITER ;
```

5.4 Stored Function

5.4.1 Function to get the total likes on a post

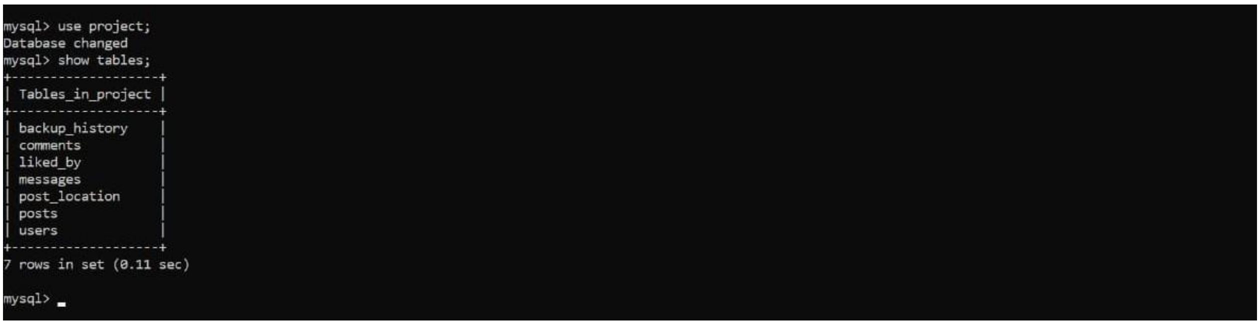
```
DELIMITER $$

CREATE FUNCTION GetTotalLikes(p_id INT) RETURNS INT
READS SQL DATA
BEGIN
    DECLARE total_likes INT;

    SELECT like_count INTO total_likes
    FROM Posts
    WHERE post_id = p_id;

    IF total_likes IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Post ID does not exist';
    END IF;

    RETURN total_likes;
END$$

DELIMITER ;
```

5.4.2 Function to calculate average likes per post:

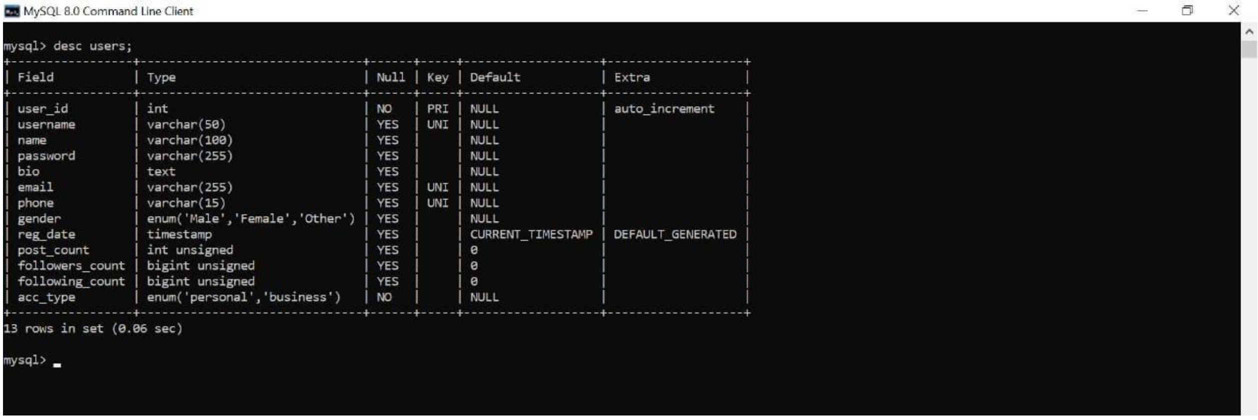
```
DELIMITER $$
CREATE FUNCTION Calculate_Avg_Likes_Per_Post () RETURNS DECIMAL(10,2)
READS SQL DATA
BEGIN
    DECLARE avg_likes DECIMAL(10,2);

    SELECT AVG(like_count) INTO avg_likes
    FROM Posts;

    RETURN avg_likes;
END $$
DELIMITER ;
```

Chapter 6: Result

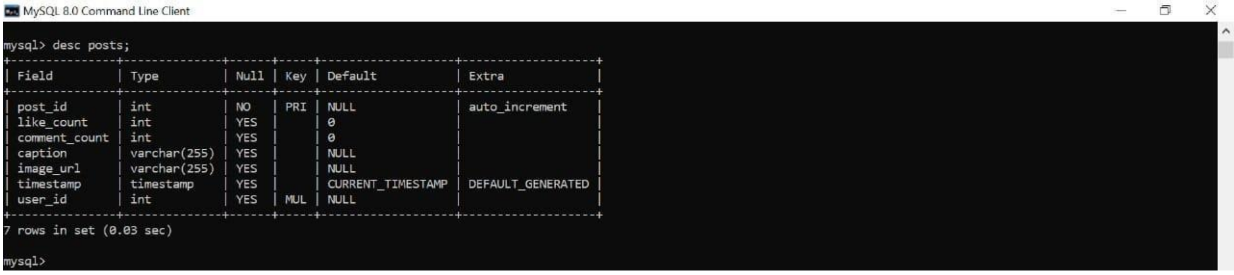
Tables:



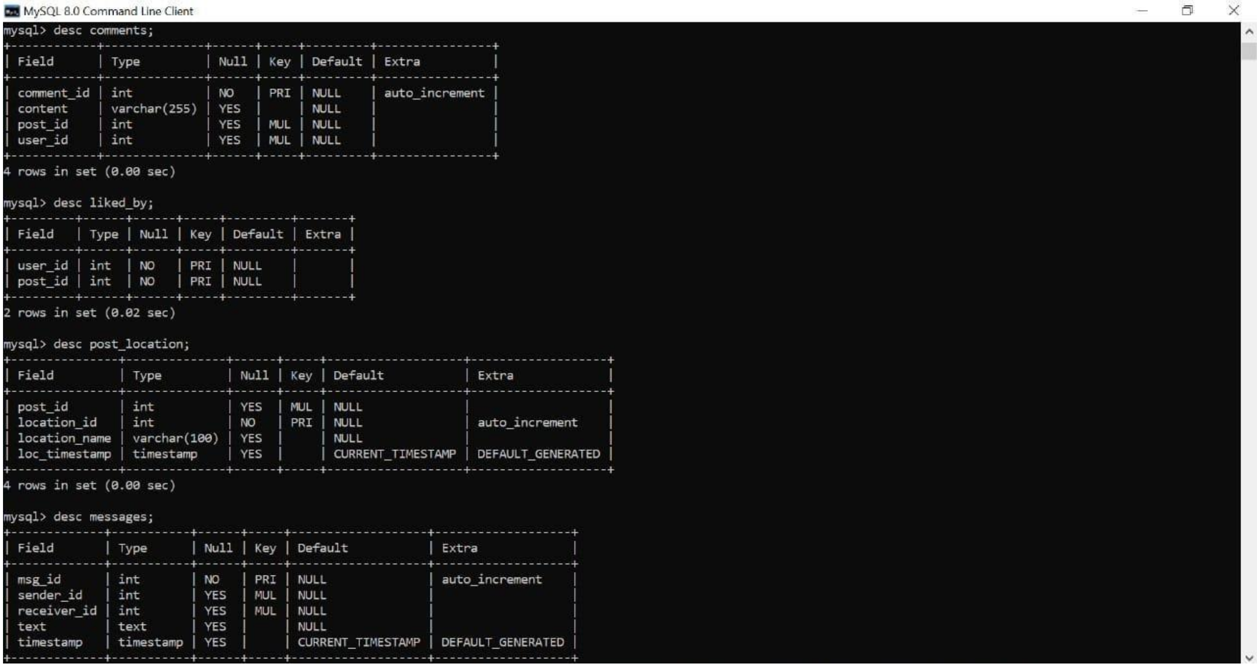
```
mysql> use project;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_project |
+-------------------+
| backup_history    |
| comments          |
| liked_by          |
| messages          |
| post_location     |
| posts             |
| users             |
+-------------------+
7 rows in set (0.11 sec)

mysql> _
```



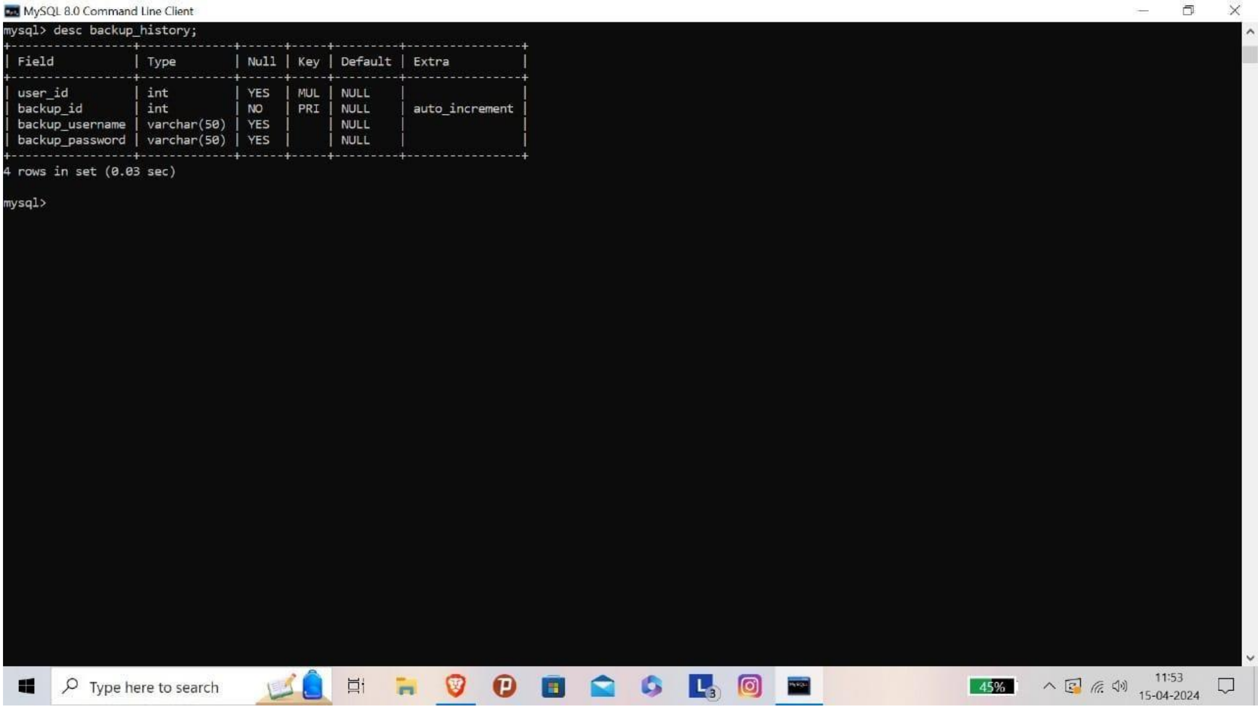
```
mysql> desc users;
+-----------------+------------------------------+------+-----+-------------------+-------------------+
| Field           | Type                         | Null | Key | Default           | Extra             |
+-----------------+------------------------------+------+-----+-------------------+-------------------+
| user_id         | int                          | NO   | PRI | NULL              | auto_increment    |
| username        | varchar(50)                  | YES  | UNI | NULL              |                   |
| name            | varchar(100)                 | YES  |     | NULL              |                   |
| password        | varchar(255)                 | YES  |     | NULL              |                   |
| bio             | text                         | YES  |     | NULL              |                   |
| email           | varchar(255)                 | YES  | UNI | NULL              |                   |
| phone           | varchar(15)                  | YES  | UNI | NULL              |                   |
| gender          | enum('Male','Female','Other') | YES  |     | NULL              |                   |
| reg_date        | timestamp                    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| post_count      | int unsigned                 | YES  |     | 0                 |                   |
| followers_count | bigint unsigned              | YES  |     | 0                 |                   |
| following_count | bigint unsigned              | YES  |     | 0                 |                   |
| acc_type        | enum('personal','business')  | NO   |     | NULL              |                   |
+-----------------+------------------------------+------+-----+-------------------+-------------------+
13 rows in set (0.06 sec)

mysql> _
```



```
mysql> desc posts;
+---------------+--------------+------+-----+-------------------+-------------------+
| Field         | Type         | Null | Key | Default           | Extra             |
+---------------+--------------+------+-----+-------------------+-------------------+
| post_id       | int          | NO   | PRI | NULL              | auto_increment    |
| like_count    | int          | YES  |     | 0                 |                   |
| comment_count | int          | YES  |     | 0                 |                   |
| caption       | varchar(255) | YES  |     | NULL              |                   |
| image_url     | varchar(255) | YES  |     | NULL              |                   |
| timestamp     | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
| user_id       | int          | YES  | MUL | NULL              |                   |
+---------------+--------------+------+-----+-------------------+-------------------+
7 rows in set (0.03 sec)

mysql>
```

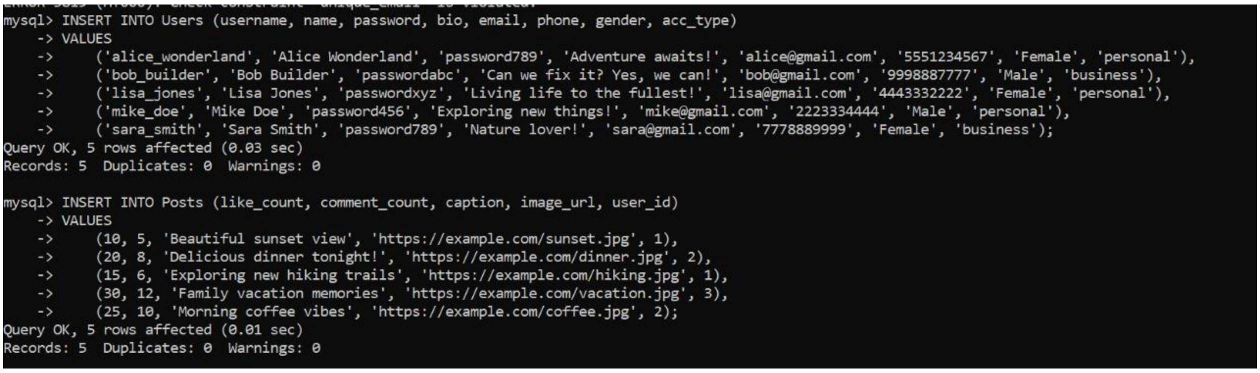


```
mysql> desc comments;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| comment_id | int          | NO   | PRI | NULL    | auto_increment |
| content    | varchar(255) | YES  |     | NULL    |                |
| post_id    | int          | YES  | MUL | NULL    |                |
| user_id    | int          | YES  | MUL | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> desc liked_by;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| user_id | int  | NO   | PRI | NULL    |       |
| post_id | int  | NO   | PRI | NULL    |       |
+---------+------+------+-----+---------+-------+
2 rows in set (0.02 sec)

mysql> desc post_location;
+---------------+--------------+------+-----+-------------------+-------------------+
| Field         | Type         | Null | Key | Default           | Extra             |
+---------------+--------------+------+-----+-------------------+-------------------+
| post_id       | int          | YES  | MUL | NULL              |                   |
| location_id   | int          | NO   | PRI | NULL              | auto_increment    |
| location_name | varchar(100) | YES  |     | NULL              |                   |
| loc_timestamp | timestamp    | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+---------------+--------------+------+-----+-------------------+-------------------+
4 rows in set (0.00 sec)

mysql> desc messages;
+-------------+-----------+------+-----+-------------------+-------------------+
| Field       | Type      | Null | Key | Default           | Extra             |
+-------------+-----------+------+-----+-------------------+-------------------+
| msg_id      | int       | NO   | PRI | NULL              | auto_increment    |
| sender_id   | int       | YES  | MUL | NULL              |                   |
| receiver_id | int       | YES  | MUL | NULL              |                   |
| text        | text      | YES  |     | NULL              |                   |
| timestamp   | timestamp | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-------------+-----------+------+-----+-------------------+-------------------+
```

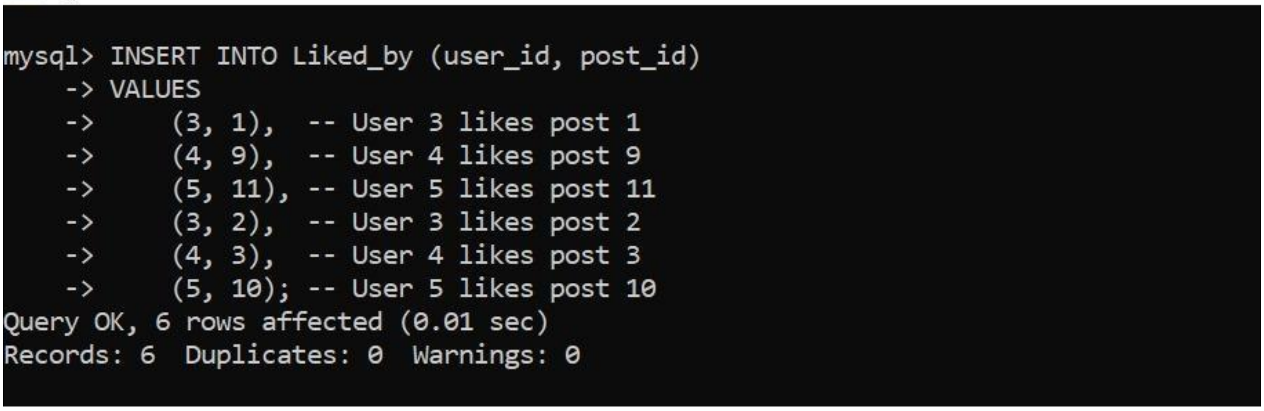


```
mysql> desc backup_history;
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| user_id         | int         | YES  | MUL | NULL    |                |
| backup_id       | int         | NO   | PRI | NULL    | auto_increment |
| backup_username | varchar(50) | YES  |     | NULL    |                |
| backup_password | varchar(50) | YES  |     | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
4 rows in set (0.03 sec)

mysql>
```

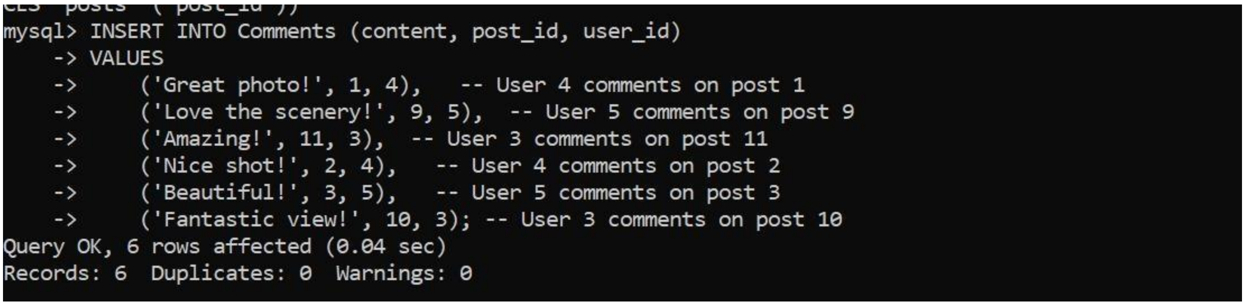
Insertion:



```
mysql> INSERT INTO Users (username, name, password, bio, email, phone, gender, acc_type)
    -> VALUES
    ->     ('alice_wonderland', 'Alice Wonderland', 'password789', 'Adventure awaits!', 'alice@gmail.com', '5551234567', 'Female', 'personal'),
    ->     ('bob_builder', 'Bob Builder', 'passwordabc', 'Can we fix it? Yes, we can!', 'bob@gmail.com', '9998887777', 'Male', 'business'),
    ->     ('lisa_jones', 'Lisa Jones', 'passwordxyz', 'Living life to the fullest!', 'lisa@gmail.com', '4443332222', 'Female', 'personal'),
    ->     ('mike_doe', 'Mike Doe', 'password456', 'Exploring new things!', 'mike@gmail.com', '2223334444', 'Male', 'personal'),
    ->     ('sara_smith', 'Sara Smith', 'password789', 'Nature lover!', 'sara@gmail.com', '7778889999', 'Female', 'business');
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Posts (like_count, comment_count, caption, image_url, user_id)
    -> VALUES
    ->     (10, 5, 'Beautiful sunset view', 'https://example.com/sunset.jpg', 1),
    ->     (20, 8, 'Delicious dinner tonight!', 'https://example.com/dinner.jpg', 2),
    ->     (15, 6, 'Exploring new hiking trails', 'https://example.com/hiking.jpg', 1),
    ->     (30, 12, 'Family vacation memories', 'https://example.com/vacation.jpg', 3),
    ->     (25, 10, 'Morning coffee vibes', 'https://example.com/coffee.jpg', 2);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

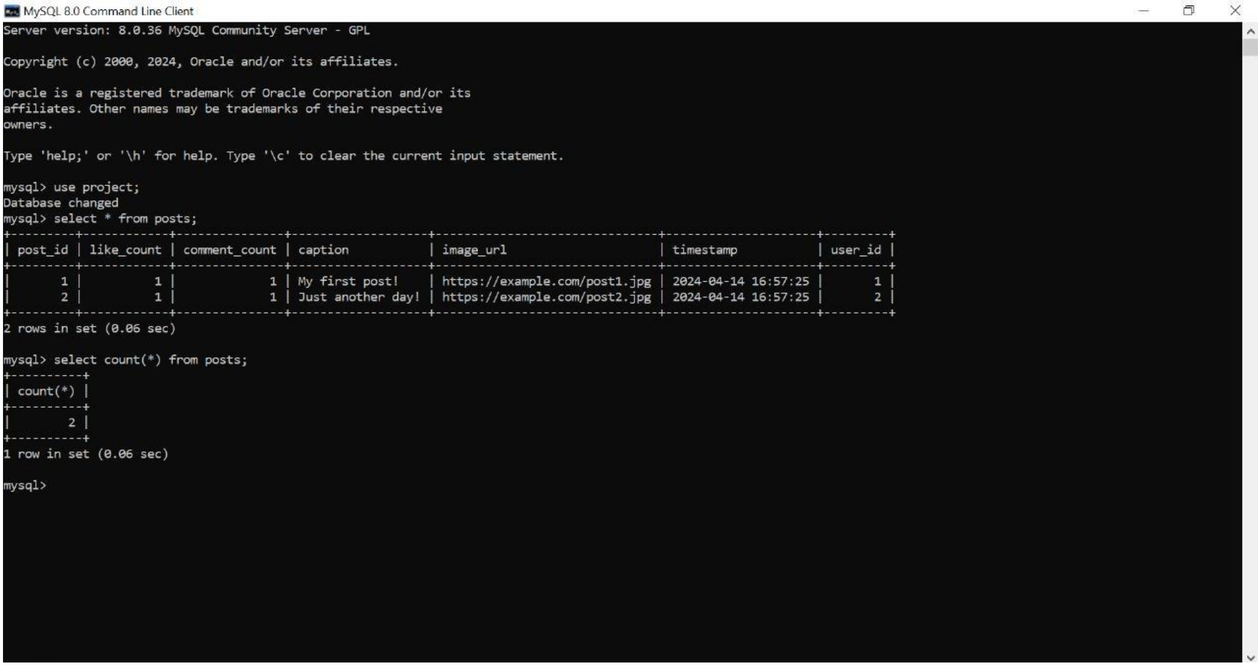


```
mysql> INSERT INTO Liked_by (user_id, post_id)
    -> VALUES
    ->     (3, 1),  -- User 3 likes post 1
    ->     (4, 9),  -- User 4 likes post 9
    ->     (5, 11), -- User 5 likes post 11
    ->     (3, 2),  -- User 3 likes post 2
    ->     (4, 3),  -- User 4 likes post 3
    ->     (5, 10); -- User 5 likes post 10
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

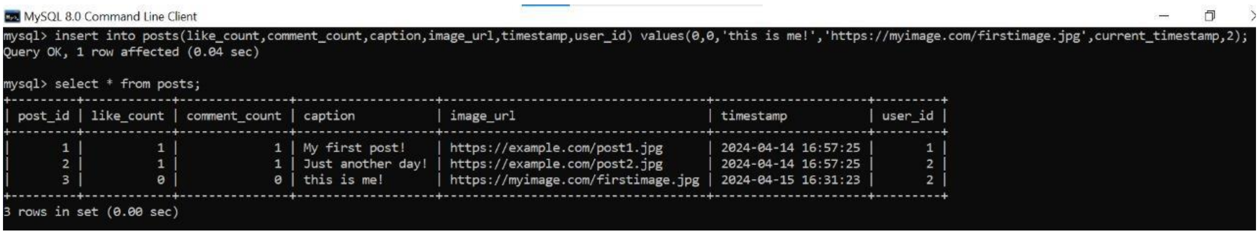


```
mysql> INSERT INTO Comments (content, post_id, user_id)
    -> VALUES
    ->     ('Great photo!', 1, 4),   -- User 4 comments on post 1
    ->     ('Love the scenery!', 9, 5),  -- User 5 comments on post 9
    ->     ('Amazing!', 11, 3),  -- User 3 comments on post 11
    ->     ('Nice shot!', 2, 4),   -- User 4 comments on post 2
    ->     ('Beautiful!', 3, 5),   -- User 5 comments on post 3
    ->     ('Fantastic view!', 10, 3); -- User 3 comments on post 10
Query OK, 6 rows affected (0.04 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

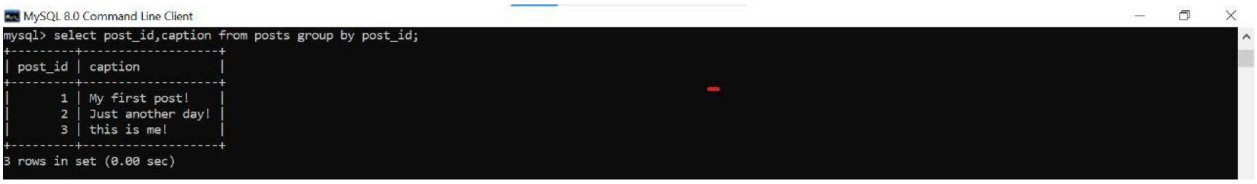
Implementation



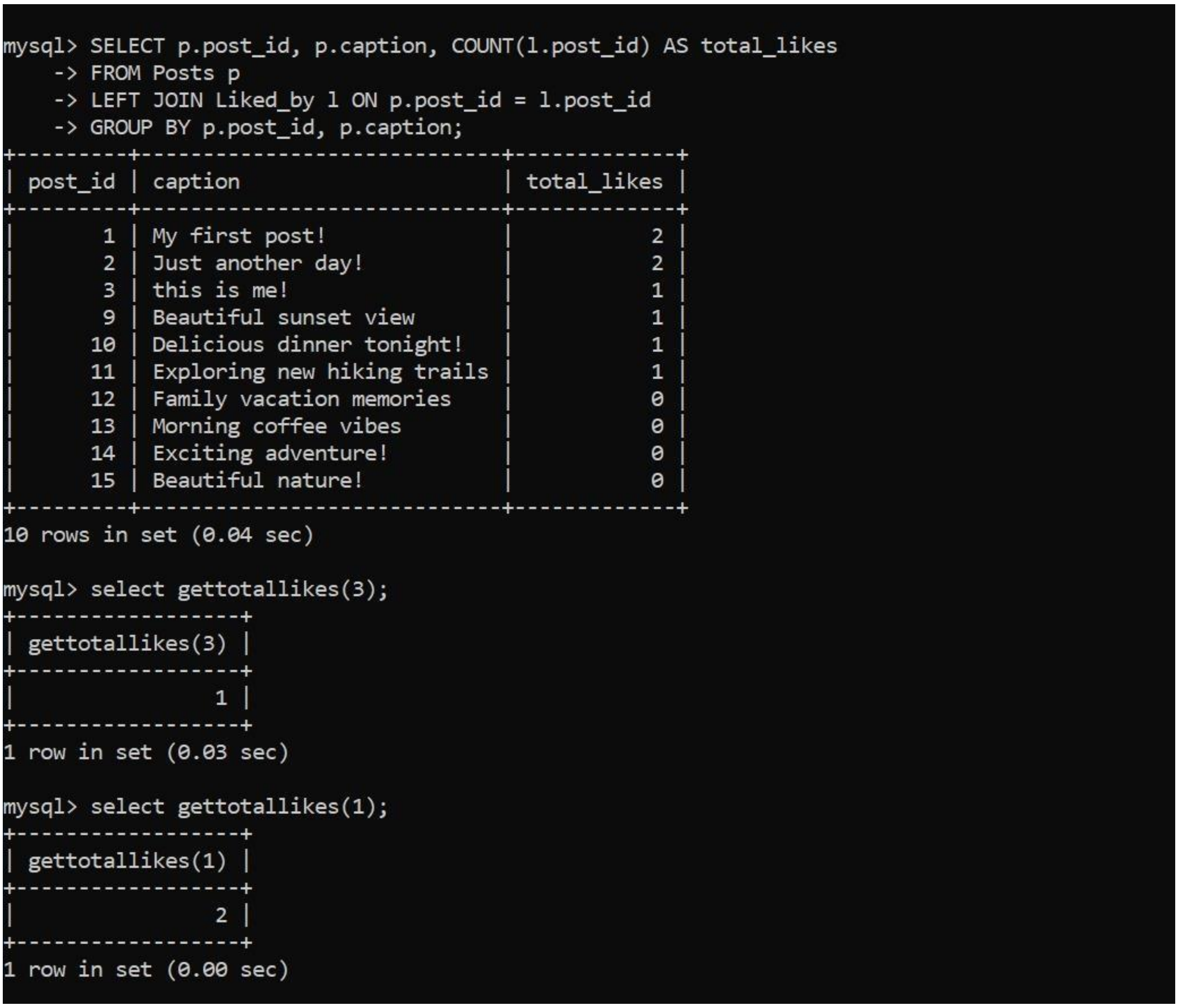
```
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use project;
Database changed
mysql> select * from posts;
+---------+------------+---------------+------------------+------------------------------+---------------------+---------+
| post_id | like_count | comment_count | caption          | image_url                    | timestamp           | user_id |
+---------+------------+---------------+------------------+------------------------------+---------------------+---------+
|       1 |          1 |             1 | My first post!   | https://example.com/post1.jpg | 2024-04-14 16:57:25 |       1 |
|       2 |          1 |             1 | Just another day! | https://example.com/post2.jpg | 2024-04-14 16:57:25 |       2 |
+---------+------------+---------------+------------------+------------------------------+---------------------+---------+
2 rows in set (0.06 sec)

mysql> select count(*) from posts;
+----------+
| count(*) |
+----------+
|        2 |
+----------+
1 row in set (0.06 sec)

mysql>
```



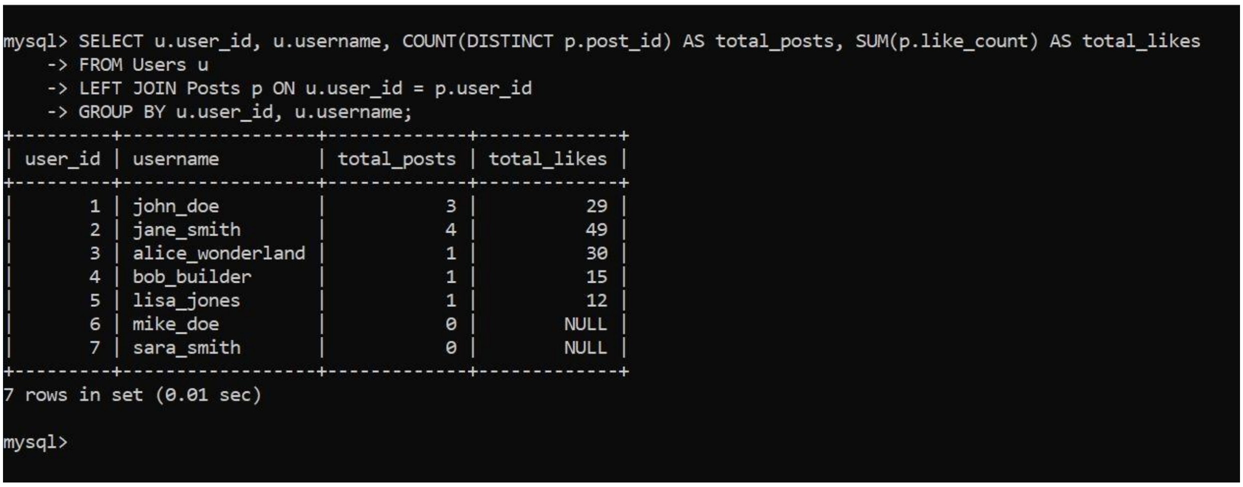
```
mysql> insert into posts(like_count,comment_count,caption,image_url,timestamp,user_id) values(0,0,'this is me!','https://myimage.com/firstimage.jpg',current_timestamp,2);
Query OK, 1 row affected (0.04 sec)

mysql> select * from posts;
+---------+------------+---------------+-------------------+------------------------------------+---------------------+---------+
| post_id | like_count | comment_count | caption           | image_url                          | timestamp           | user_id |
+---------+------------+---------------+-------------------+------------------------------------+---------------------+---------+
|       1 |          1 |             1 | My first post!    | https://example.com/post1.jpg      | 2024-04-14 16:57:25 |       1 |
|       2 |          1 |             1 | Just another day! | https://example.com/post2.jpg      | 2024-04-14 16:57:25 |       2 |
|       3 |          0 |             0 | this is me!       | https://myimage.com/firstimage.jpg | 2024-04-15 16:31:23 |       2 |
+---------+------------+---------------+-------------------+------------------------------------+---------------------+---------+
3 rows in set (0.00 sec)
```



```
mysql> select post_id,caption from posts group by post_id;
+---------+-------------------+
| post_id | caption           |
+---------+-------------------+
|       1 | My first post!    |
|       2 | Just another day! |
|       3 | this is me!       |
+---------+-------------------+
3 rows in set (0.00 sec)
```



```
mysql> SELECT p.post_id, p.caption, COUNT(l.post_id) AS total_likes
    -> FROM Posts p
    -> LEFT JOIN Liked_by l ON p.post_id = l.post_id
    -> GROUP BY p.post_id, p.caption;
+---------+-----------------------------+-------------+
| post_id | caption                     | total_likes |
+---------+-----------------------------+-------------+
|       1 | My first post!              |           2 |
|       2 | Just another day!           |           2 |
|       3 | this is me!                 |           1 |
|       9 | Beautiful sunset view       |           1 |
|      10 | Delicious dinner tonight!   |           1 |
|      11 | Exploring new hiking trails |           1 |
|      12 | Family vacation memories    |           0 |
|      13 | Morning coffee vibes        |           0 |
|      14 | Exciting adventure!         |           0 |
|      15 | Beautiful nature!           |           0 |
+---------+-----------------------------+-------------+
10 rows in set (0.04 sec)

mysql> select gettotallikes(3);
+------------------+
| gettotallikes(3) |
+------------------+
|                1 |
+------------------+
1 row in set (0.03 sec)

mysql> select gettotallikes(1);
+------------------+
| gettotallikes(1) |
+------------------+
|                2 |
+------------------+
1 row in set (0.00 sec)
```



```
mysql> SELECT u.user_id, u.username, COUNT(DISTINCT p.post_id) AS total_posts, SUM(p.like_count) AS total_likes
    -> FROM Users u
    -> LEFT JOIN Posts p ON u.user_id = p.user_id
    -> GROUP BY u.user_id, u.username;
+---------+-----------------+-------------+-------------+
| user_id | username        | total_posts | total_likes |
+---------+-----------------+-------------+-------------+
|       1 | john_doe        |           3 |          29 |
|       2 | jane_smith      |           4 |          49 |
|       3 | alice_wonderland |          1 |          30 |
|       4 | bob_builder     |           1 |          15 |
|       5 | lisa_jones      |           1 |          12 |
|       6 | mike_doe        |           0 |        NULL |
|       7 | sara_smith      |           0 |        NULL |
+---------+-----------------+-------------+-------------+
7 rows in set (0.01 sec)

mysql>
```

Chapter 7: Conclusion and Future Work

The social media dashboard project has successfully implemented core functionalities such as user management, post creation, likes, comments, and backups. It provides a foundation for users to interact with each other and share content within a structured environment. The project ensures data integrity and scalability while aiming to deliver a seamless user experience.

Future scope:

- **Advanced Analytics**: Implement analytics features to analyze user behavior, post-performance, and trends.
- **Security Enhancements**: Strengthen security measures to protect user data and prevent unauthorized access.
- **Mobile Application Development**: Create a mobile app version for on-the-go access and convenience.
- **Social Networking Features**: Expand features to include messaging, groups, events, and content sharing to enhance social interactions.
- **Monetization:** Explore options for monetization, such as advertising, sponsored content, or premium subscriptions.
- **Community Building**: Foster a sense of community by facilitating user interactions, discussions, and collaborations.
- **Accessibility and Inclusivity**: Ensure the platform is accessible to users with disabilities and inclusive of diverse communities.

By focusing on these areas, the social media dashboard can evolve into a comprehensive platform that meets the needs and expectations of its users while staying competitive in the dynamic social media landscape.

Each Team Member contribution:

Shravani	Implementation, Result
Bhavya Goyal	Implementation, Queries, Function, Triggers, Procedures, Result
Avadh Gandhi	Abstract, Introduction, Synopsis, Functional requirements, ER diagram, Implementation
Ganesh Chaudhari	Abstract, Introduction, Synopsis, ER diagram, Schema diagram, Triggers, Procedures
Mitwa Saraf	Documentation, ER diagram, Schema diagram, Data dictionary, Queries, Functions