DBMS - Mini Project
Freeter- A Clone of Twitter

Submitted By:

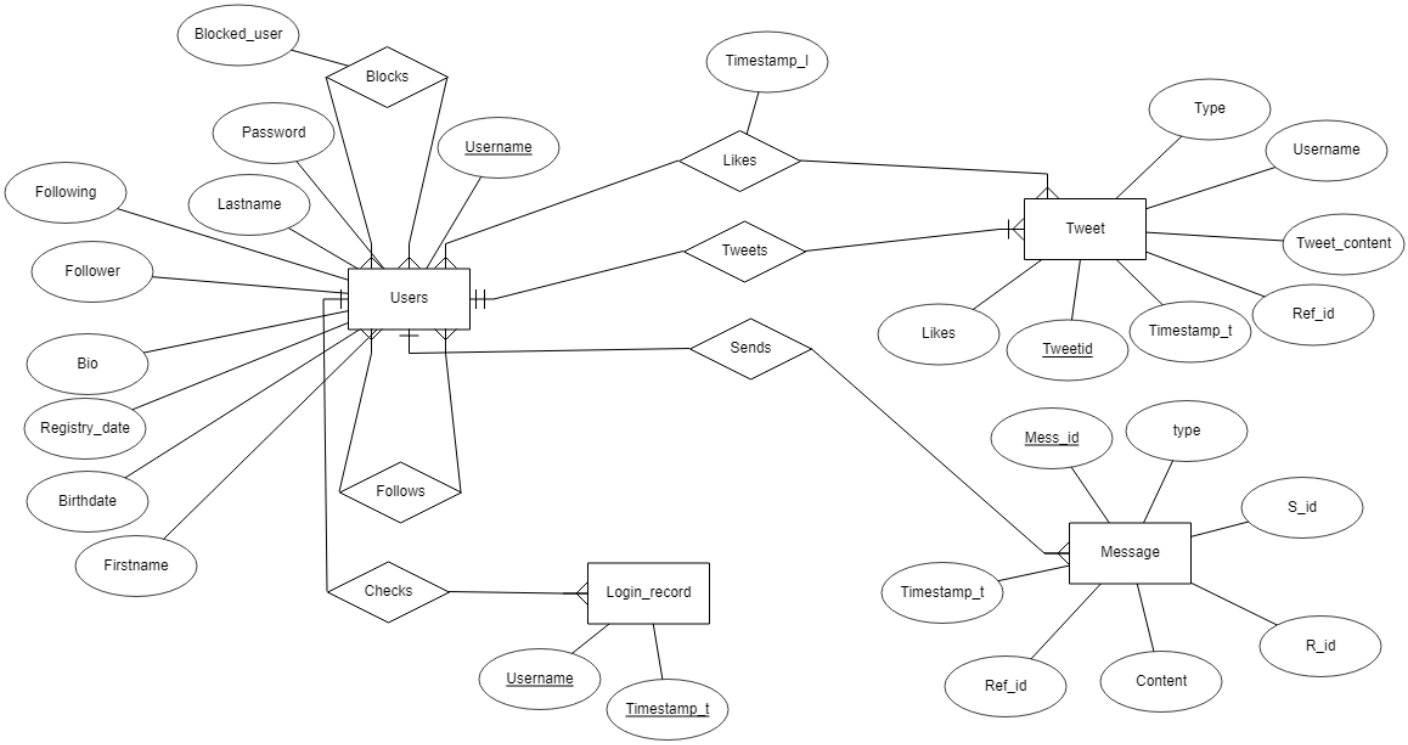
V Himadhith
PES1UG20CS478
V Semester Section H

Short Description and Scope of the Project

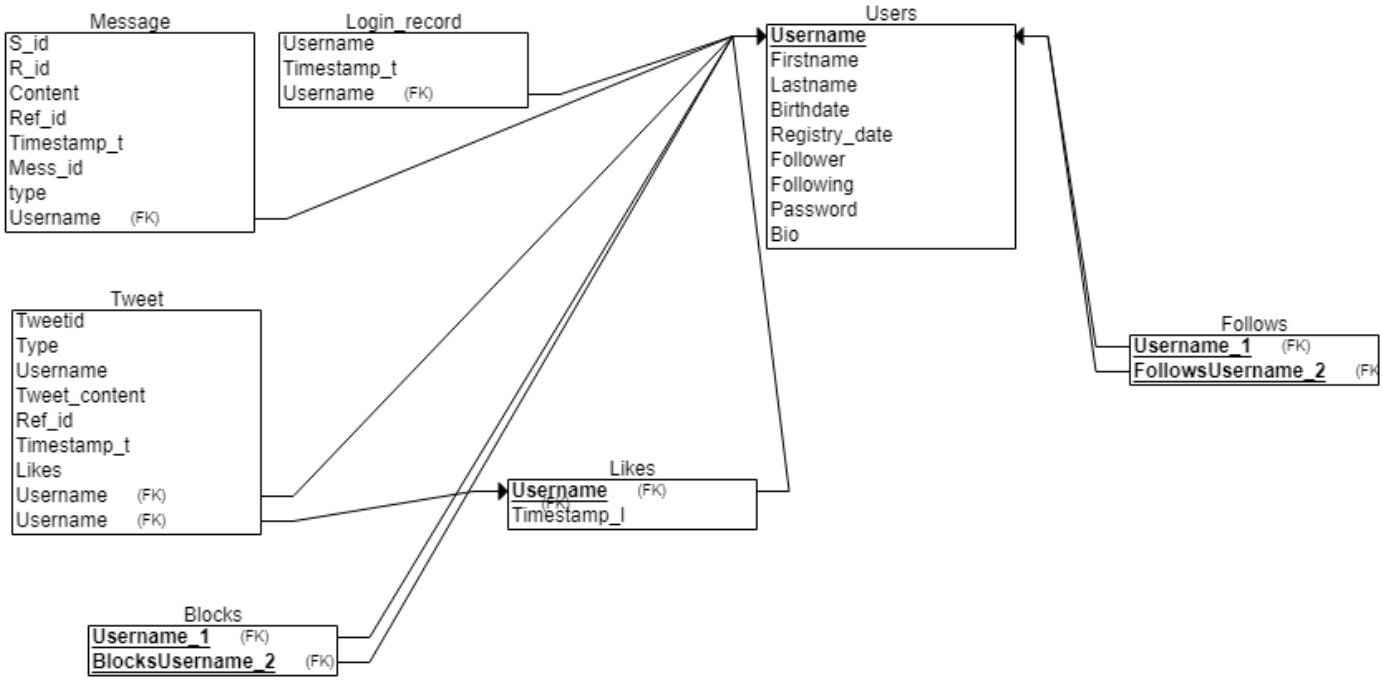
The project is a free social media platform where the users can interact with their friends and also be able to look at the vast majority of the content even if they are not followers of the person. Freeter creates a safe and secure environment for users to work with along with being able to handle the queries that the user makes with an inbuilt window which can take the user query and fetch the desired results.

Since there are rumors that twitter is about to be shut down or on the verge of being a pay to use application, we aim to provide the user with services similar to the competition but for absolutely free.

ER Diagram


Blocked_user
Blocks
Password
Username
Following
Lastname
Follower
Users
Bio
Registry_date
Birthdate
Firstname
Follows
Checks
Login_record
Username
Timestamp_t
Timestamp_l
Likes
Tweets
Sends
Tweet
Type
Username
Tweet_content
Ref_id
Timestamp_t
Tweetid
Likes
Mess_id
type
S_id
Message
Timestamp_t
R_id
Ref_id
Content

Relational Schema



DDL statements - Building the database

```
CREATE TABLE users (
username    VARCHAR(20) NOT NULL ,
firstName    VARCHAR(20)NOT NULL,
lastName    VARCHAR(20)NOT NULL,
birthDate    DATE NOT NULL,
registery_date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
bio        Varchar(64),
followers   INT  NOT NULL DEFAULT 0,
following   INT  NOT NULL DEFAULT 0,
password VARCHAR(128) NOT NULL ,

PRIMARY KEY (username)
);


CREATE TABLE tweet(
tweetid          INT AUTO_INCREMENT,
type            CHAR(1) NOT NULL CHECK ( type in ('T', 'C')) ,
username         VARCHAR(20) NOT NULL,
tweet_content     VARCHAR(256) NOT NULL,
ref_id          INT,
timestamp_t     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
likes           INT NOT NULL DEFAULT 0,

PRIMARY KEY (tweetid),
FOREIGN KEY (username) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (ref_id) REFERENCES tweet(tweetid)
ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE message(
 mess_id           INT AUTO_INCREMENT,
 type           CHAR(1) NOT NULL CHECK ( type in ('M', 'T')) ,
 s_id           VARCHAR(20) NOT NULL ,
 r_id           VARCHAR(20) NOT NULL ,
 content        VARCHAR(256),
 ref_id         INT ,
 timestamp_t     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

PRIMARY KEY (mess_id),
FOREIGN KEY (s_id) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (r_id) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (ref_id) REFERENCES tweet(tweetid)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE login_record(
  username      VARCHAR(20) NOT NULL ,
  timestamp_t     TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

PRIMARY KEY (username, timestamp_t),
FOREIGN KEY (username) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE follow (
follower    VARCHAR(20) NOT NULL ,
following    VARCHAR(20) NOT NULL ,

PRIMARY KEY (follower, following),
FOREIGN KEY (follower) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE ,
FOREIGN KEY (following) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE likes (
username        VARCHAR(20),
tweetid          INT,
timeStamp_l   TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,

PRIMARY KEY     (tweetid, username),
FOREIGN KEY     (tweetid) REFERENCES tweet(tweetid)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY     (username) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE block(
username        VARCHAR(20),
blocked_user     VARCHAR(20),

PRIMARY KEY (username, blocked_user ),
FOREIGN KEY (username) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE ,
FOREIGN KEY (blocked_user) REFERENCES users(username)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

Populating the Database

```
LOAD DATA INFILE 'C:/Users/Himadhith/Desktop/DBMS_project/Backend/data/users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:/Users/Himadhith/Desktop/DBMS_project/Backend/data/tweet.csv'
INTO TABLE tweet
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

insert into follow (follower, following) values ('afallawe9', 'jhearsey2');
insert into follow (follower, following) values ('jphizackarley6', 'tglisane5');
insert into follow (follower, following) values ('cgallaher3', 'eretchless7');
insert into follow (follower, following) values ('ahymas1', 'cnother4');
insert into follow (follower, following) values ('piamittii8', 'jphizackarley6');
insert into follow (follower, following) values ('eretchless7', 'piamittii8');
insert into follow (follower, following) values ('jhearsey2', 'mjoynes0');
insert into follow (follower, following) values ('mjoynes0', 'afallawe9');
insert into follow (follower, following) values ('cnother4', 'ahymas1');
insert into follow (follower, following) values ('tglisane5', 'cgallaher3');
insert into likes (username, tweetid, timeStamp_l) values ('jphizackarley6', '20', '5/31/2022');
insert into likes (username, tweetid, timeStamp_l) values ('ahymas1', '13', '11/13/2022');
insert into likes (username, tweetid, timeStamp_l) values ('mjoynes0', '9', '9/23/2022');
insert into likes (username, tweetid, timeStamp_l) values ('afallawe9', '2', '12/18/2021');
insert into likes (username, tweetid, timeStamp_l) values ('piamittii8', '15', '12/12/2021');


insert into login_record (username, timestamp_t) values ('mjoynes0', '11/16/2021');
insert into login_record (username, timestamp_t) values ('tglisane5', '5/13/2022');
insert into login_record (username, timestamp_t) values ('eretchless7', '5/22/2022');
insert into login_record (username, timestamp_t) values ('jphizackarley6', '1/19/2022');
insert into login_record (username, timestamp_t) values ('afallawe9', '6/23/2022');
insert into login_record (username, timestamp_t) values ('cgallaher3', '6/18/2022');
insert into login_record (username, timestamp_t) values ('piamittii8', '8/27/2022');
insert into login_record (username, timestamp_t) values ('jhearsey2', '4/6/2022');
insert into login_record (username, timestamp_t) values ('ahymas1', '4/11/2022');
insert into login_record (username, timestamp_t) values ('cnother4', '7/12/2022');

insert into likes (username, tweetid, timeStamp_l) values ('jphizackarley6', '20', '5/31/2022');
insert into likes (username, tweetid, timeStamp_l) values ('ahymas1', '13', '11/13/2022');
insert into likes (username, tweetid, timeStamp_l) values ('mjoynes0', '9', '9/23/2022');
insert into likes (username, tweetid, timeStamp_l) values ('afallawe9', '2', '12/18/2021');
insert into likes (username, tweetid, timeStamp_l) values ('piamittii8', '15', '12/12/2021');
insert into likes (username, tweetid, timeStamp_l) values ('cgallaher3', '10', '1/1/2022');
insert into likes (username, tweetid, timeStamp_l) values ('tglisane5', '17', '10/9/2022');
insert into likes (username, tweetid, timeStamp_l) values ('jhearsey2', '7', '9/27/2022');
insert into likes (username, tweetid, timeStamp_l) values ('eretchless7', '1', '11/4/2022');
insert into likes (username, tweetid, timeStamp_l) values ('cnother4', '6', '12/28/2021');
```

Join Queries

Showcase at least 4 join queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

Find the activities of the users that another user is following

```
SELECT y.type, y.username, y.tweet_content , y.cc AS ref_content, y.us AS ref_username, y.timestamp_t
FROM follow, (  SELECT tweet.tweetid, tweet.type, tweet.username, tweet.tweet_content, tweet.ref_id, tweet.timestamp_t, tweet.likes, t.tweet_content AS cc, t.username AS us
                FROM tweet LEFT JOIN tweet AS t
                ON tweet.ref_id = t.tweetid) as y
WHERE follow.following = y.username AND follow.follower = person AND y.username NOT IN
                                                                (
                                                                  SELECT block.username
                                                                  FROM block
                                                                  WHERE blocked_user = person
                                                                  )
ORDER BY y.timestamp_t DESC ;
```

Find the message sent by a specific user

```
SELECT message.type, message.content, tweet.tweet_content
FROM message LEFT JOIN tweet ON message.ref_id = tweet.tweetid
WHERE r_id = person AND s_id = p_username AND (NOT message.type = 'T' OR
  tweet.username NOT IN (
  SELECT block.username
  FROM block
  WHERE blocked_user = person
  ))
ORDER BY message.timestamp_t DESC ;
```

Find all the messages the user has received

```
SELECT message.type, message.content, tweet.tweet_content
FROM message LEFT JOIN tweet ON message.ref_id = tweet.tweetid
WHERE r_id = person AND s_id = p_username AND (NOT message.type = 'T' OR
  tweet.username NOT IN (
```

```
        SELECT block.username
        FROM block
        WHERE blocked_user = person
        ))
    ORDER BY message.timestamp_t DESC ;
```

List out all the current users tweets and their replies

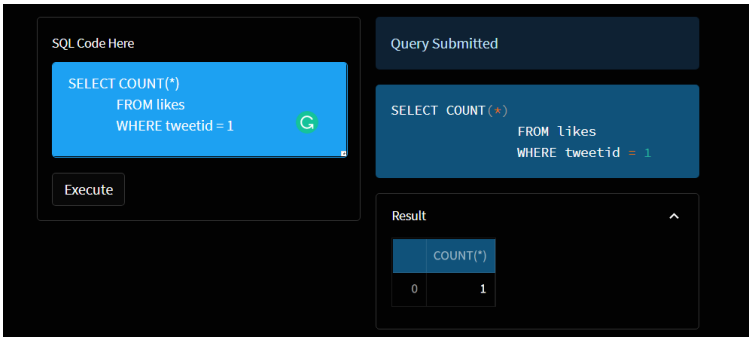
```
SELECT tweet.type ,tweet.tweet_content, t.tweet_content AS
refrence_content,t.username AS refrence_username, tweet.timestamp_t
FROM tweet LEFT JOIN tweet as t
ON tweet.ref_id = t.tweetid
WHERE  tweet.username = person
ORDER BY tweet.timestamp_t DESC ;
```

Aggregate Functions

Showcase at least 4 Aggregate function queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

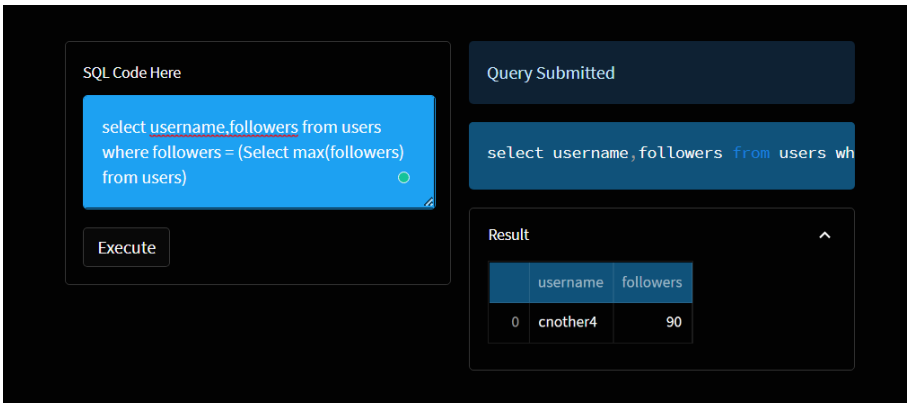
Show the number of likes on a tweet

```
SELECT COUNT(*)
FROM likes
WHERE tweetid = p_tweetid
```



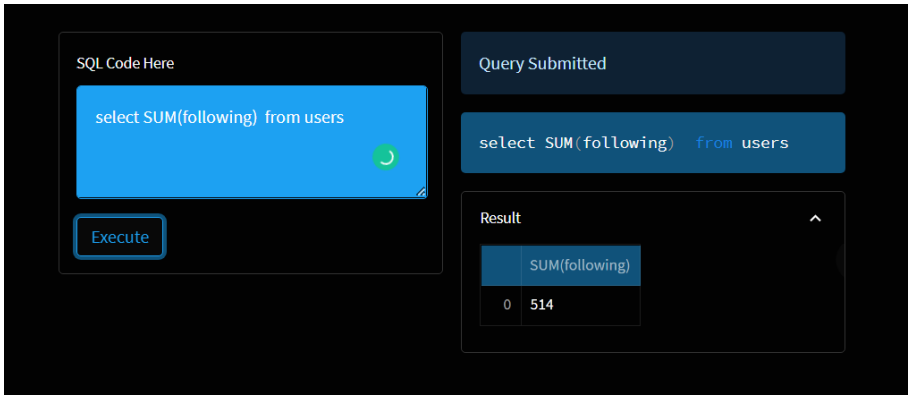
User with the most number of followers

```
select username,followers
from users
where followers = (Select max(followers) from users)
```



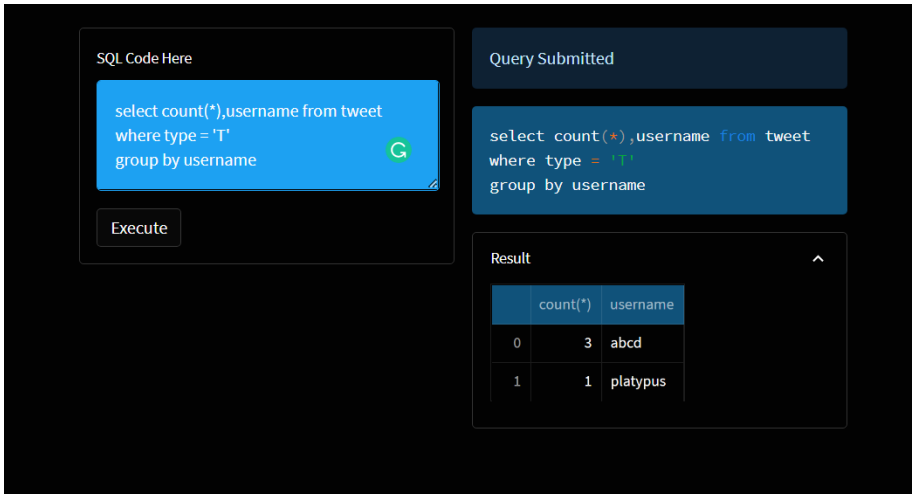
Count of all followers in the app

select SUM(following) from users



Count of all tweets grouped by usernames

select count(*),username from tweet
where type = 'T'
group by username

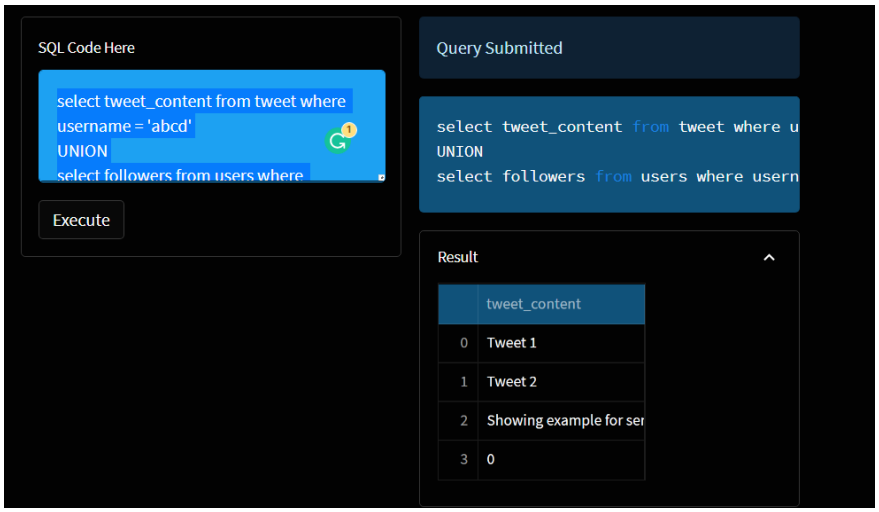


Set Operations

Showcase at least 4 Set Operations queries
Write the query in English Language, Show the equivalent SQL statement and also a screenshot of the query and the results

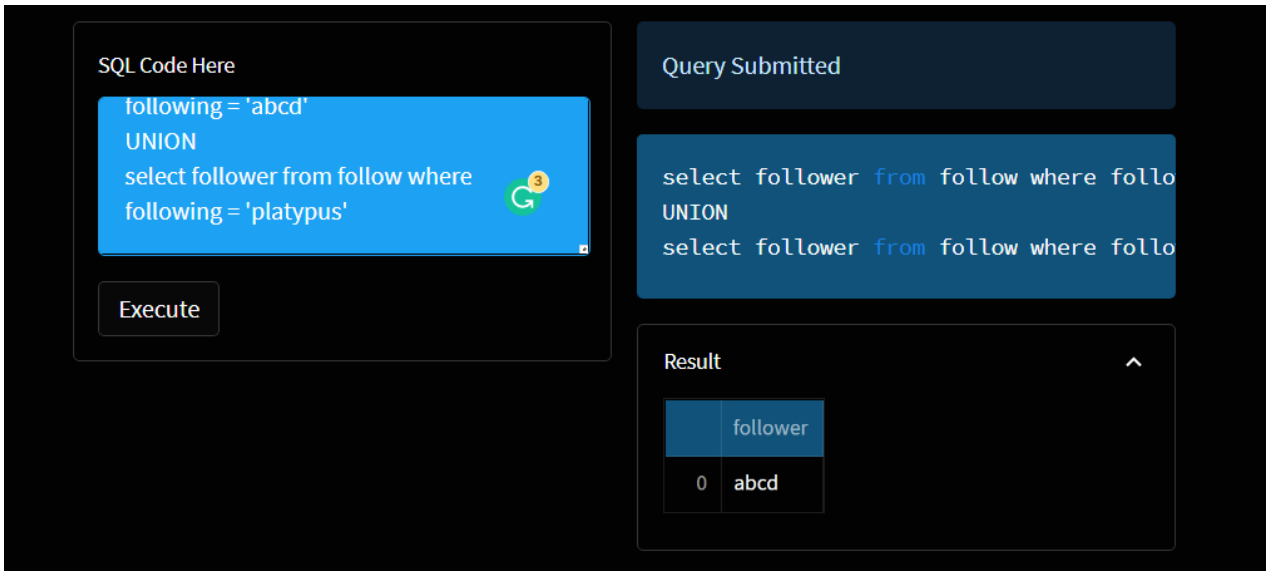
Find the number of followers along with the tweets for a particular user

select tweet_content from tweet where username = 'abcd'
UNION
select followers from users where username = 'abcd'



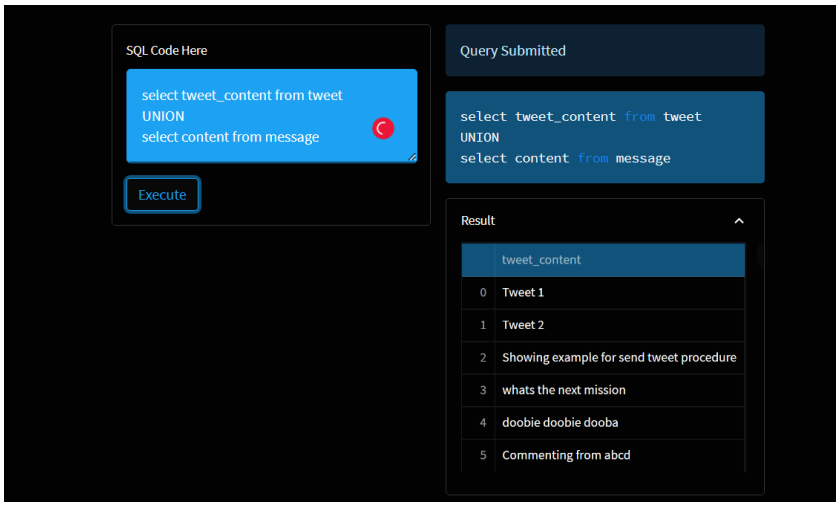
Finding which all users have same followers

select follower from follow where following = 'abcd'
UNION
select follower from follow where following = 'platypus'



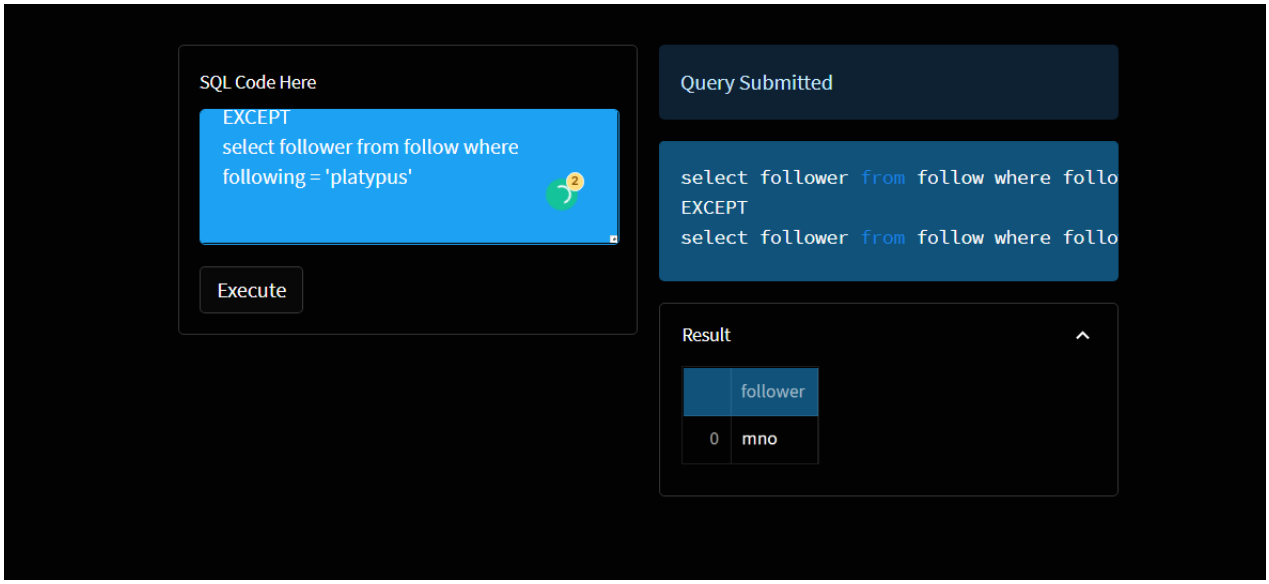
All the messages and the tweets in the app

select tweet_content from tweet
UNION
select content from message



Users which are not common among the followers

select follower from follow where following = 'abcd'
EXCEPT
select follower from follow where following = 'platypus'

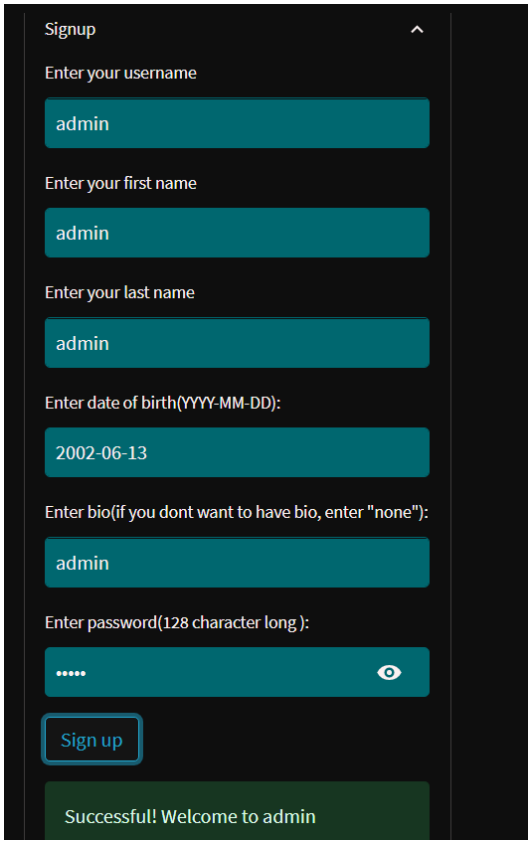


Functions and Procedures

Create a Function and Procedure. State the objective of the function / Procedure. Run and display the results.

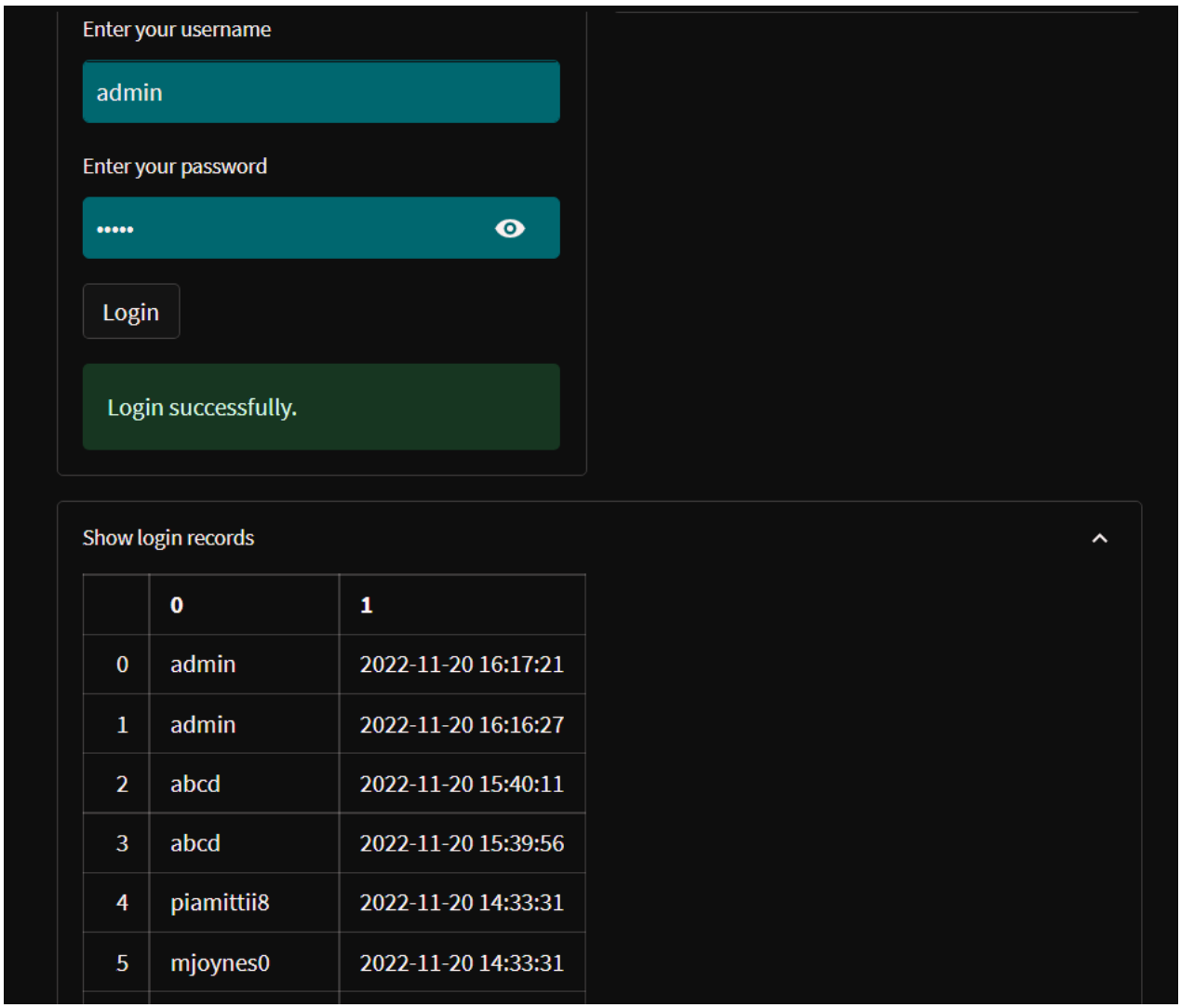
Create account function which helps the user to create an account the first time they use the app.

```
DELIMITER //
CREATE PROCEDURE create_account(
    IN p_username VARCHAR(20),
    IN p_firstname VARCHAR(20),
    IN p_lastname VARCHAR(20),
    IN p_birthdate DATE,
    IN p_bio VARCHAR(64),
    IN p_password VARCHAR(128)
)
BEGIN
    DECLARE EXIT HANDLER FOR 1062
    BEGIN
        SELECT 'Sorry, this username is already taken.' AS message;
    END;

    insert into users(username, firstName, lastName, birthDate, bio,password)
    values (p_username, p_firstname, p_lastname, p_birthdate, p_bio,SHA2(p_password, 512));
    SELECT CONCAT('Successful! Welcome to       ',p_username,'');
    commit;
end //
```



Login record keeps track of all the users who login and displays them when the an admin account requires them.

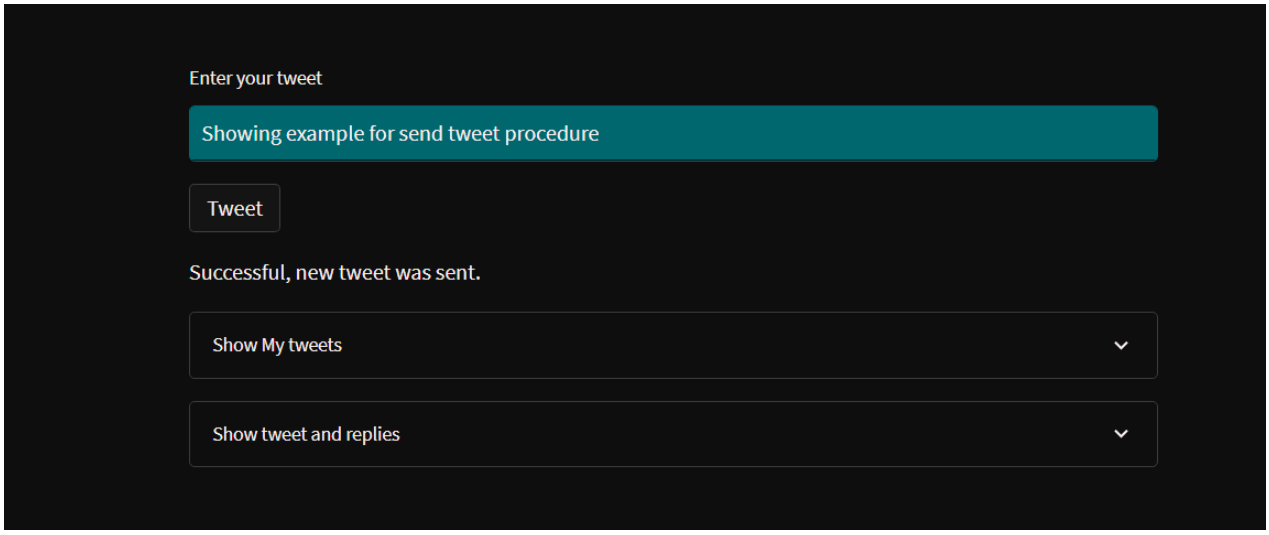
```
CREATE PROCEDURE user_logins()
BEGIN
   SELECT *
   FROM login_record
   ORDER BY timestamp_t DESC;
end //
```



	0	1
0	admin	2022-11-20 16:17:21
1	admin	2022-11-20 16:16:27
2	abcd	2022-11-20 15:40:11
3	abcd	2022-11-20 15:39:56
4	piamittii8	2022-11-20 14:33:31
5	mjoynes0	2022-11-20 14:33:31

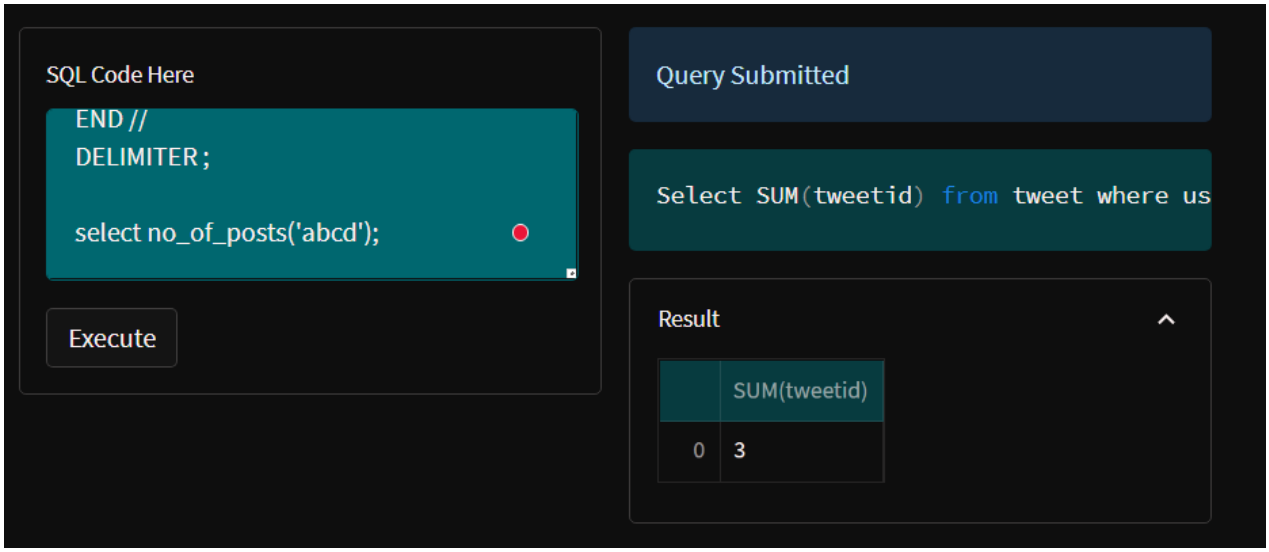
Send tweet sends a tweet which has been written by the users and is made available to all the users of the app

```
CREATE PROCEDURE send_tweet(
   IN p_content VARCHAR(256)
)
BEGIN
   DECLARE person VARCHAR(20);
   CALL  find_subject(person);
   INSERT INTO tweet(type, username, tweet_content)
   VALUES ('T', person, p_content);
   SELECT 'Successful, new tweet was sent.' AS mess;
end //
```



Find the number of tweets a particular user has tweeted

```
DELIMITER //
CREATE FUNCTION no_of_posts(uname char) RETURNS INT DETERMINISTIC
BEGIN
        DECLARE tweets INT;
   Select SUM(tweetid) INTO tweets from tweet where username = uname ;
   return tweets;
END //
DELIMITER ;

select no_of_posts('abcd');
```

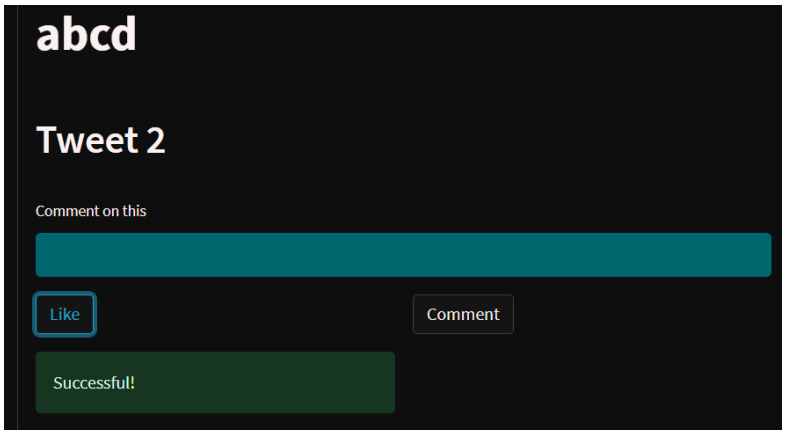


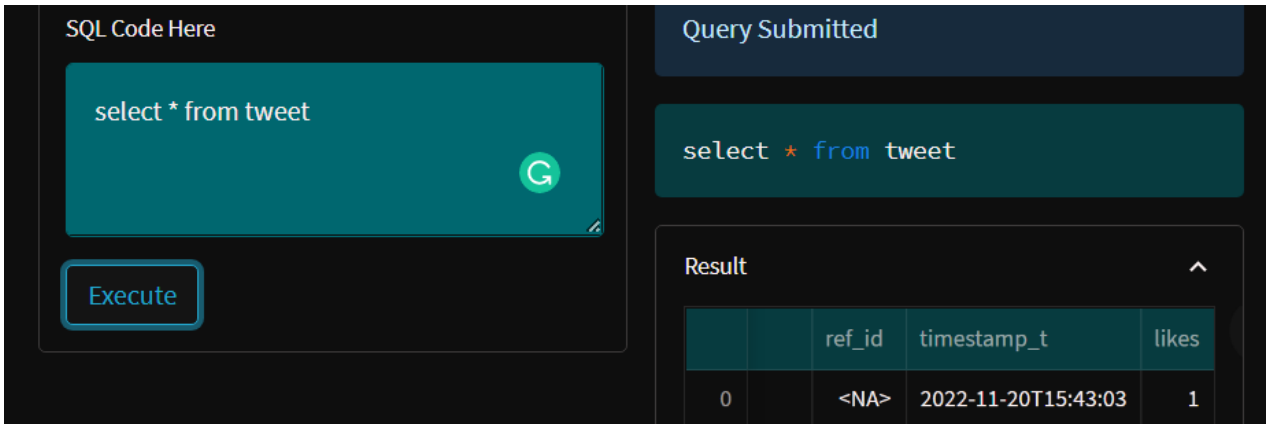
Triggers and Cursors

Create a Trigger and a Cursor. State the objective. Run and display the results

The following trigger auto_like updates the table tweet which contain all the tweets and increments the number of likes

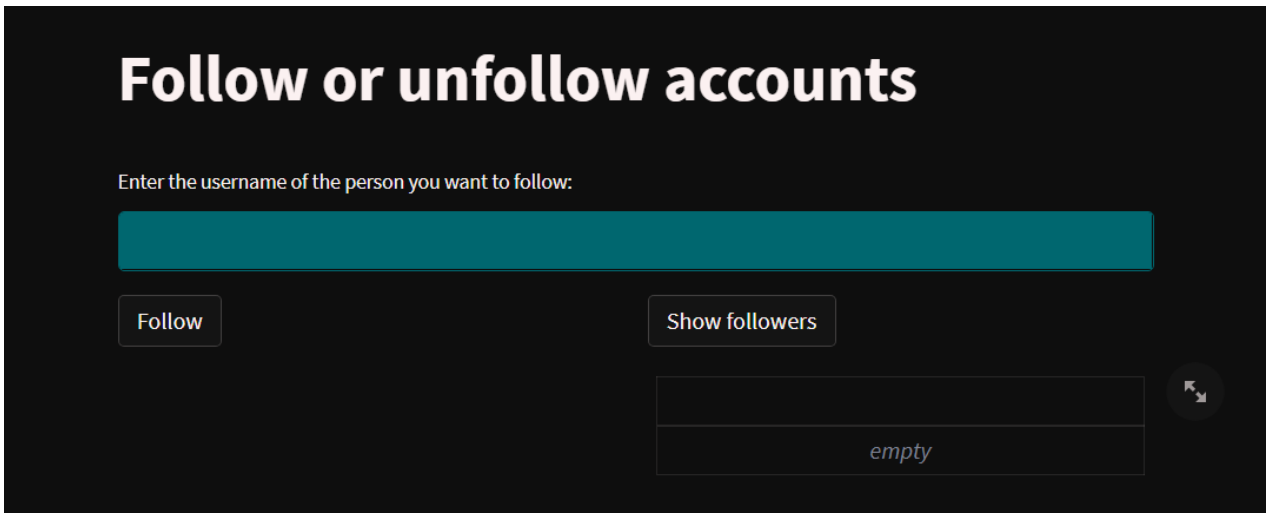
```
DELIMITER //

CREATE TRIGGER auto_like
AFTER INSERT
ON likes FOR EACH ROW
   BEGIN
      DECLARE id INT;
      SET id = NEW.tweetid;
      UPDATE tweet SET likes = likes + 1 WHERE tweetid = id;
   END //
```

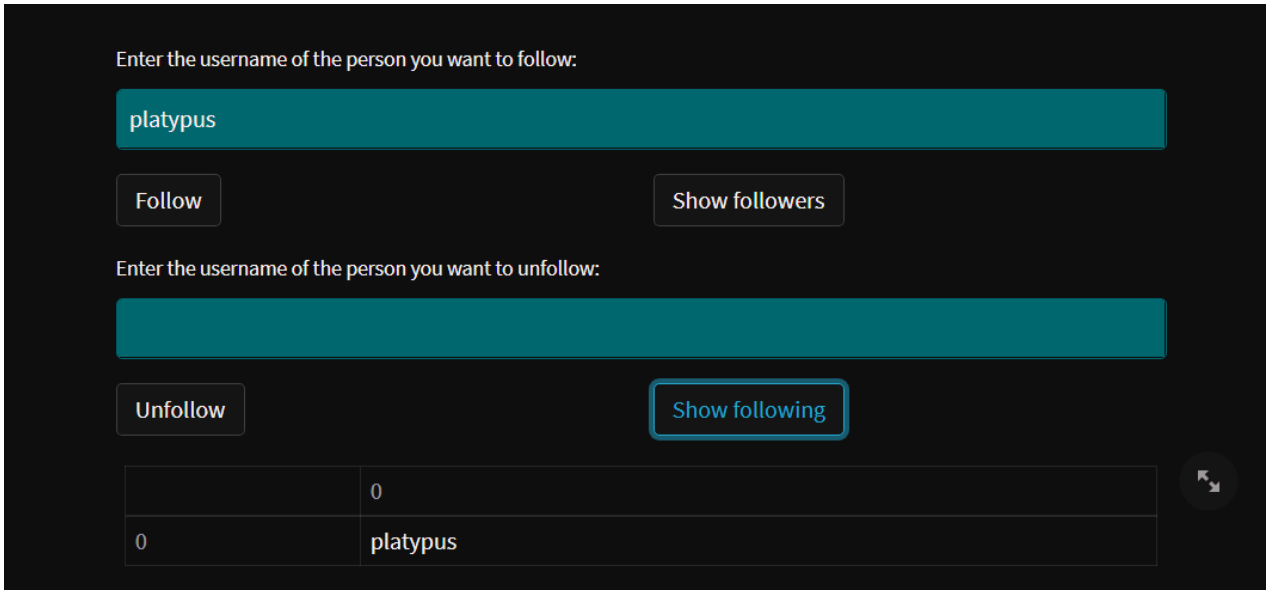




The following trigger auto_follow updates the table users incrementing the following and followers attributes .

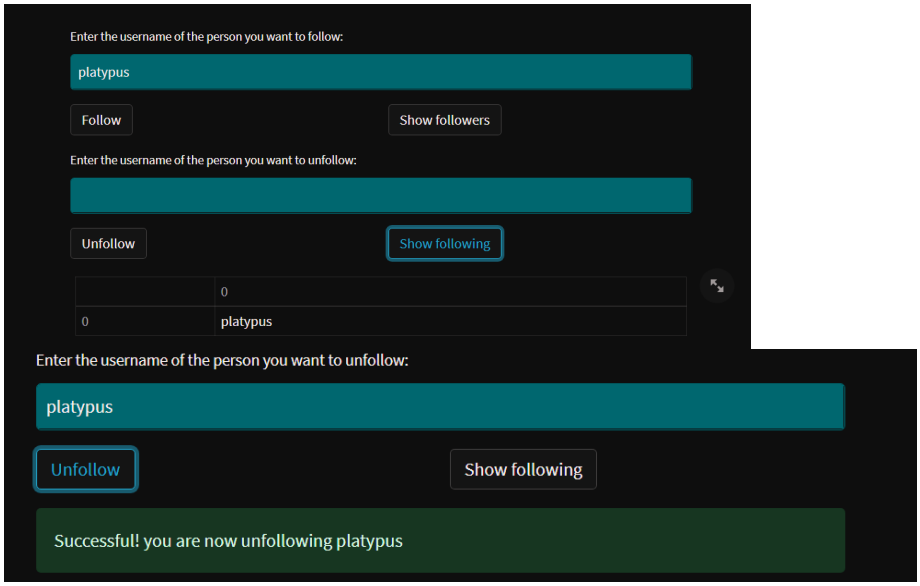
```
CREATE TRIGGER auto_follow
BEFORE INSERT
ON follow FOR EACH ROW
   BEGIN
      DECLARE follower_temp VARCHAR(20);
      DECLARE following_temp VARCHAR(20);
      SET follower_temp = NEW.follower;
      SET following_temp = NEW.following;
      UPDATE users SET following = users.following + 1 WHERE username = follower_temp;
      UPDATE users SET followers = followers + 1 WHERE username = following_temp;
   end //
```

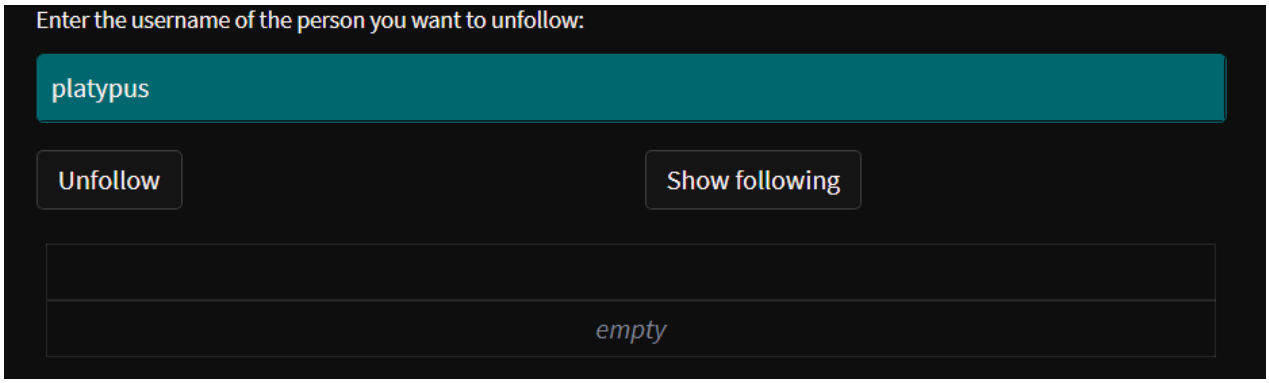




The following trigger auto_stop_follow reduces the followers and following count from the table users accordingly

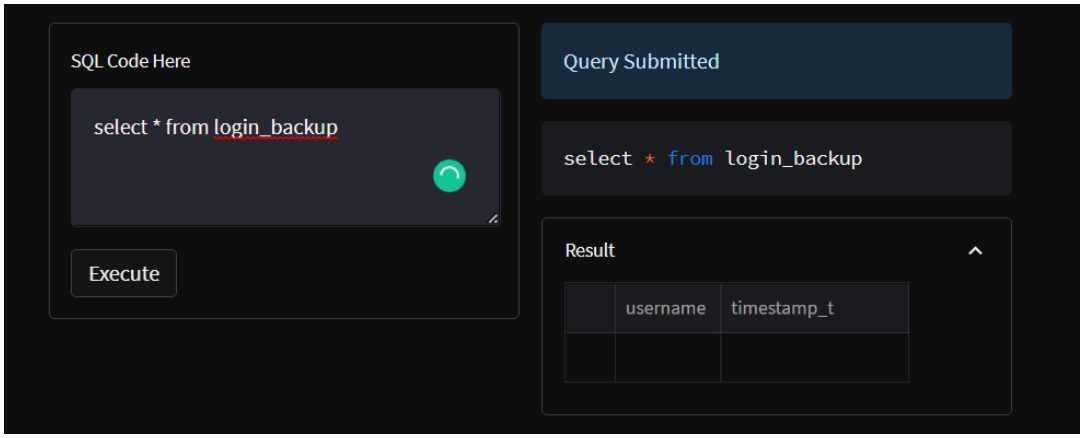
```
CREATE TRIGGER auto_stop_follow
BEFORE DELETE
ON follow FOR EACH ROW
   BEGIN
      DECLARE follower_temp VARCHAR(20);
      DECLARE following_temp VARCHAR(20);
      SET follower_temp = OLD.follower;
      SET following_temp = OLD.following;
      UPDATE users SET following = users.following - 1 WHERE username = follower_temp;
      UPDATE users SET followers = followers - 1 WHERE username = following_temp;
   end //
```



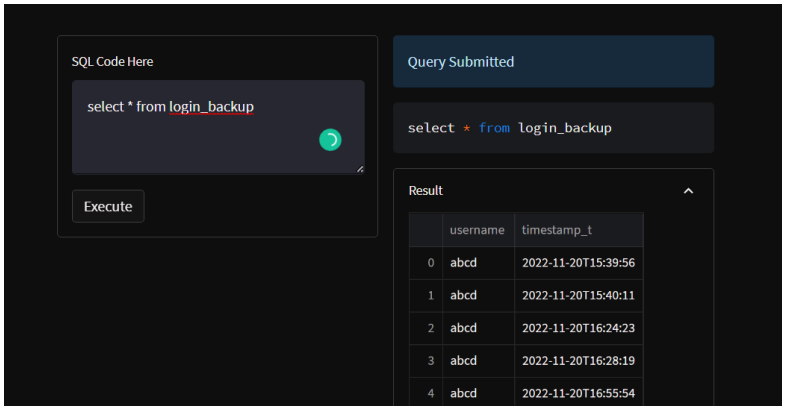


The following cursor creates a backup for all the login records and stores them in a new table called login_backup

```
DELIMITER //
CREATE procedure log_back()
BEGIN
        DECLARE done INT default 0;
   DECLARE uname varchar(20);
   DECLARE tim_stm TIMESTAMP;
   DECLARE cur CURSOR FOR SELECT * FROM login_record;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
   OPEN cur;
        label: LOOP
   FETCH cur INTO uname,tim_stm;
   INSERT INTO login_backup VALUES(uname,tim_stm);
   IF done = 1 THEN LEAVE label;
        END IF;
        END LOOP;
        CLOSE cur;
  END//
DELIMITER ;

CALL log_back;
```



After CALL log_back;

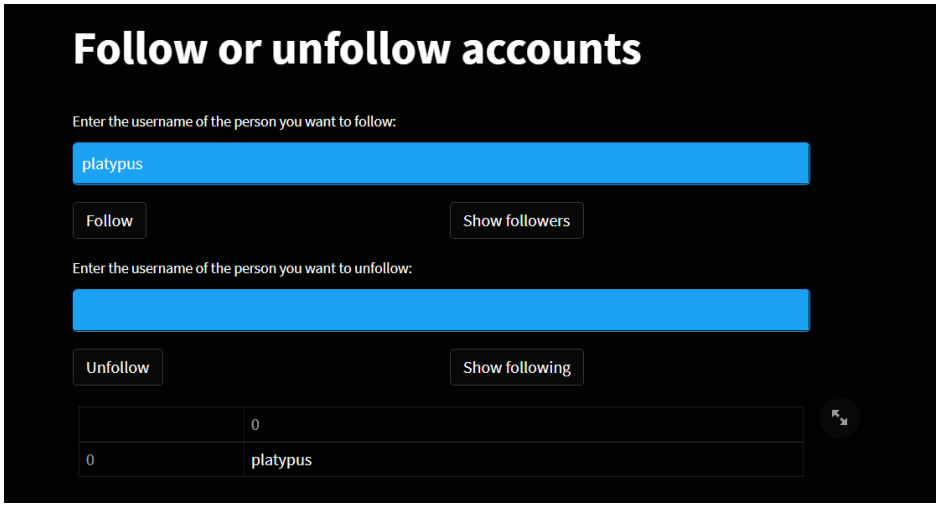


Developing a Frontend

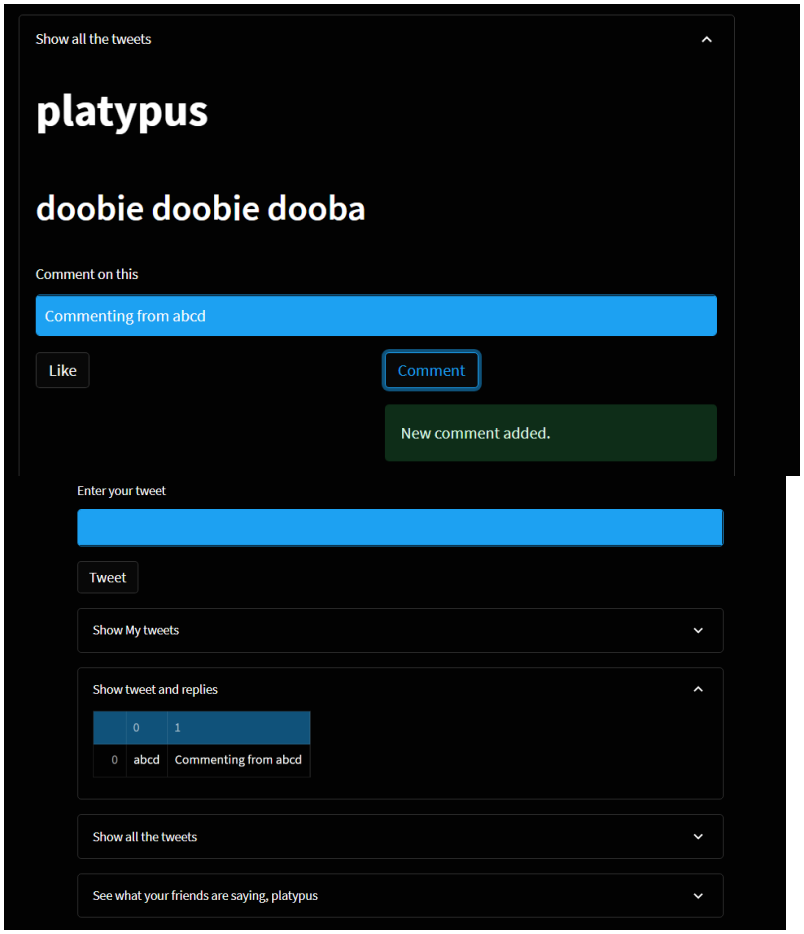
The frontend should support

1. Addition, Modification and Deletion of records from any chosen table
2. There should be an window to accept and run any SQL statement and display the result

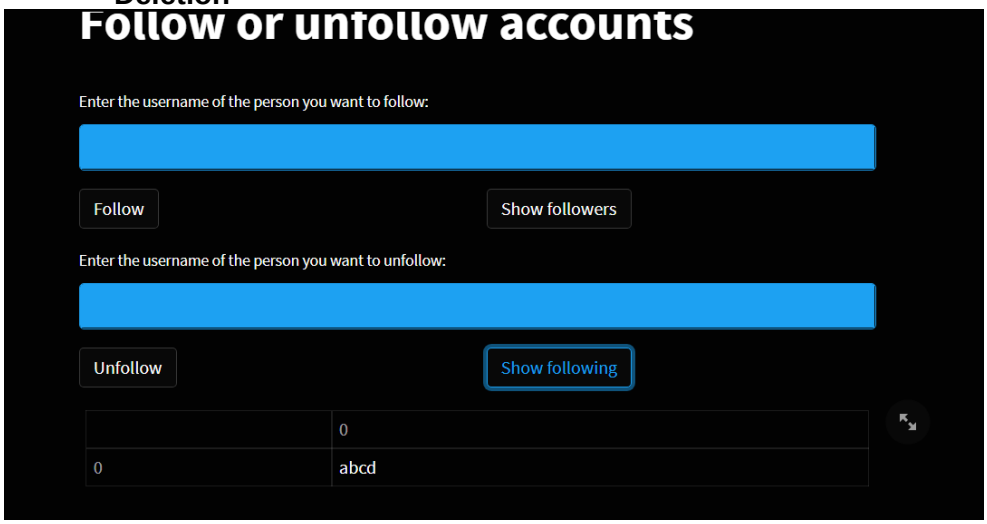
1. **Addition**

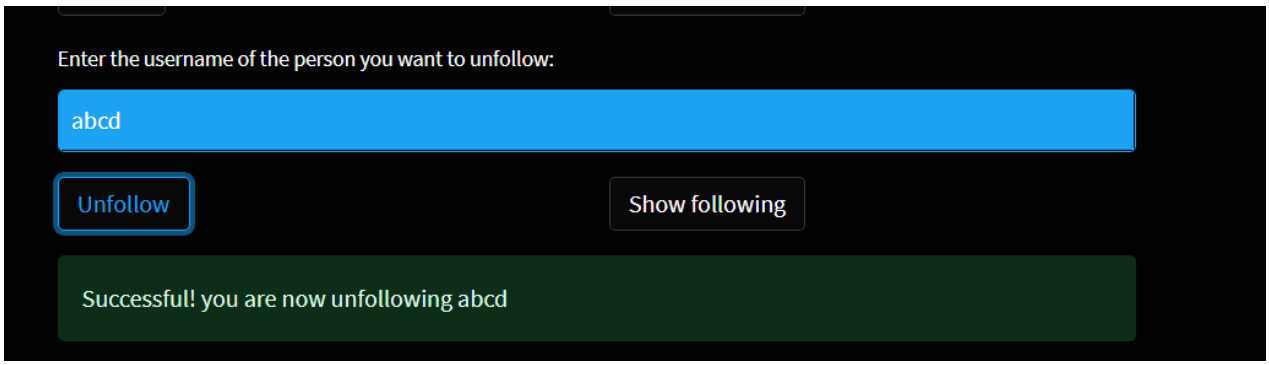


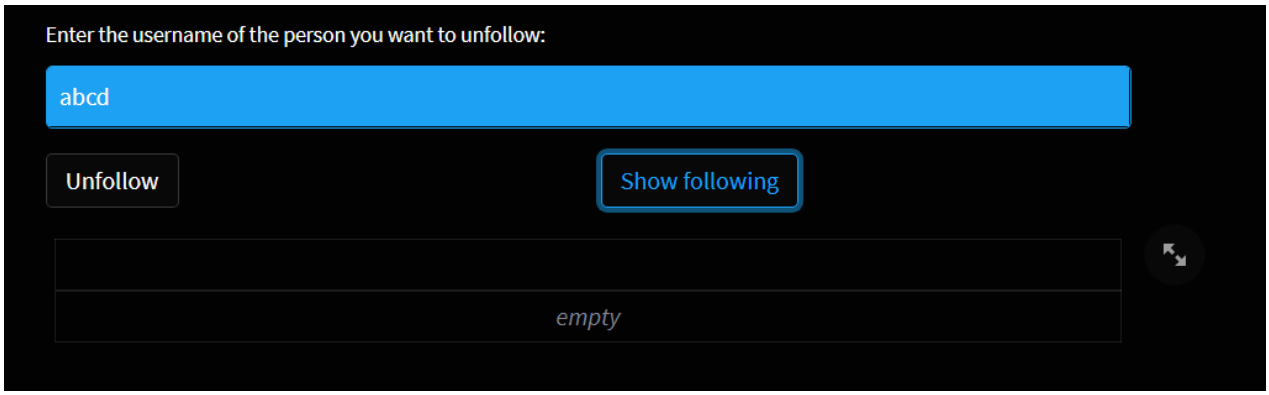
Modification



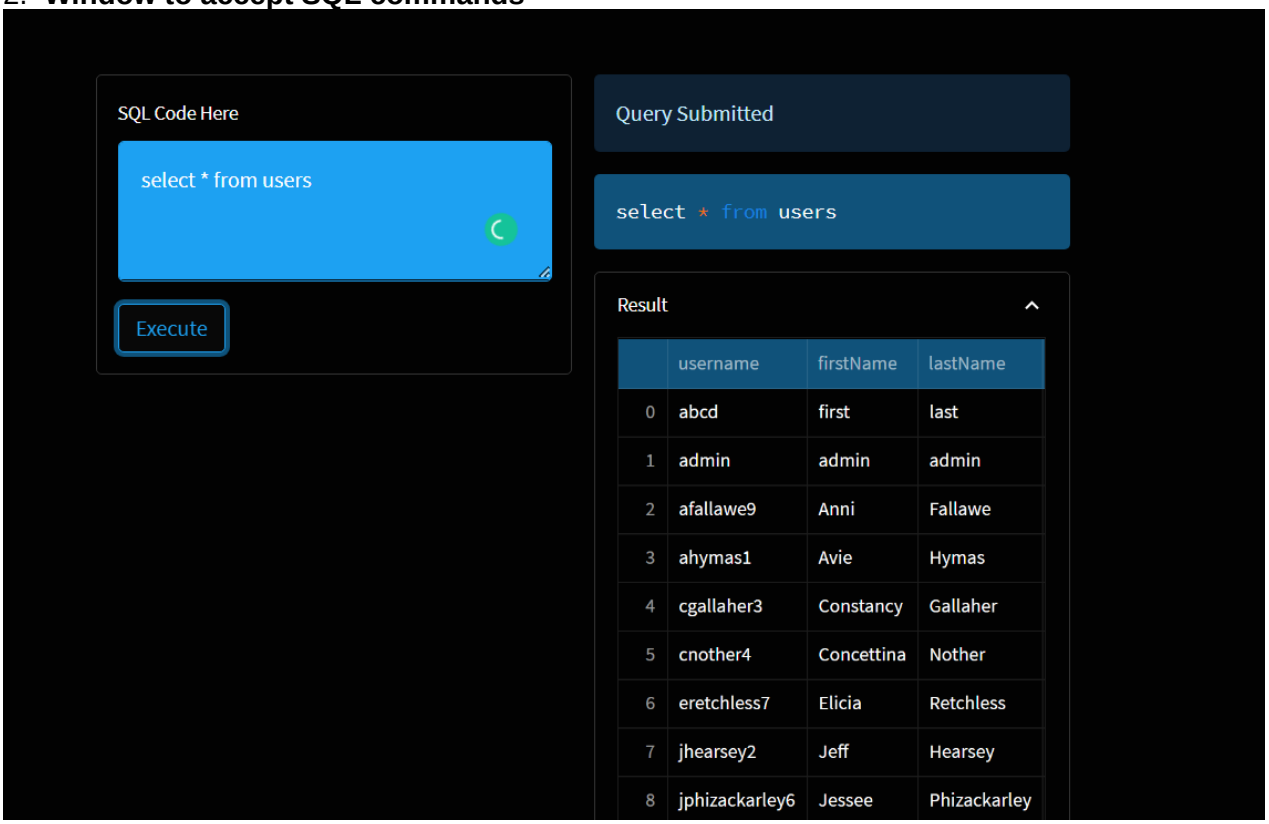
Deletion


Follow or unfollow accounts
Enter the username of the person you want to follow:
Follow
Show followers
Enter the username of the person you want to unfollow:
Unfollow
Show following
0
0
abcd

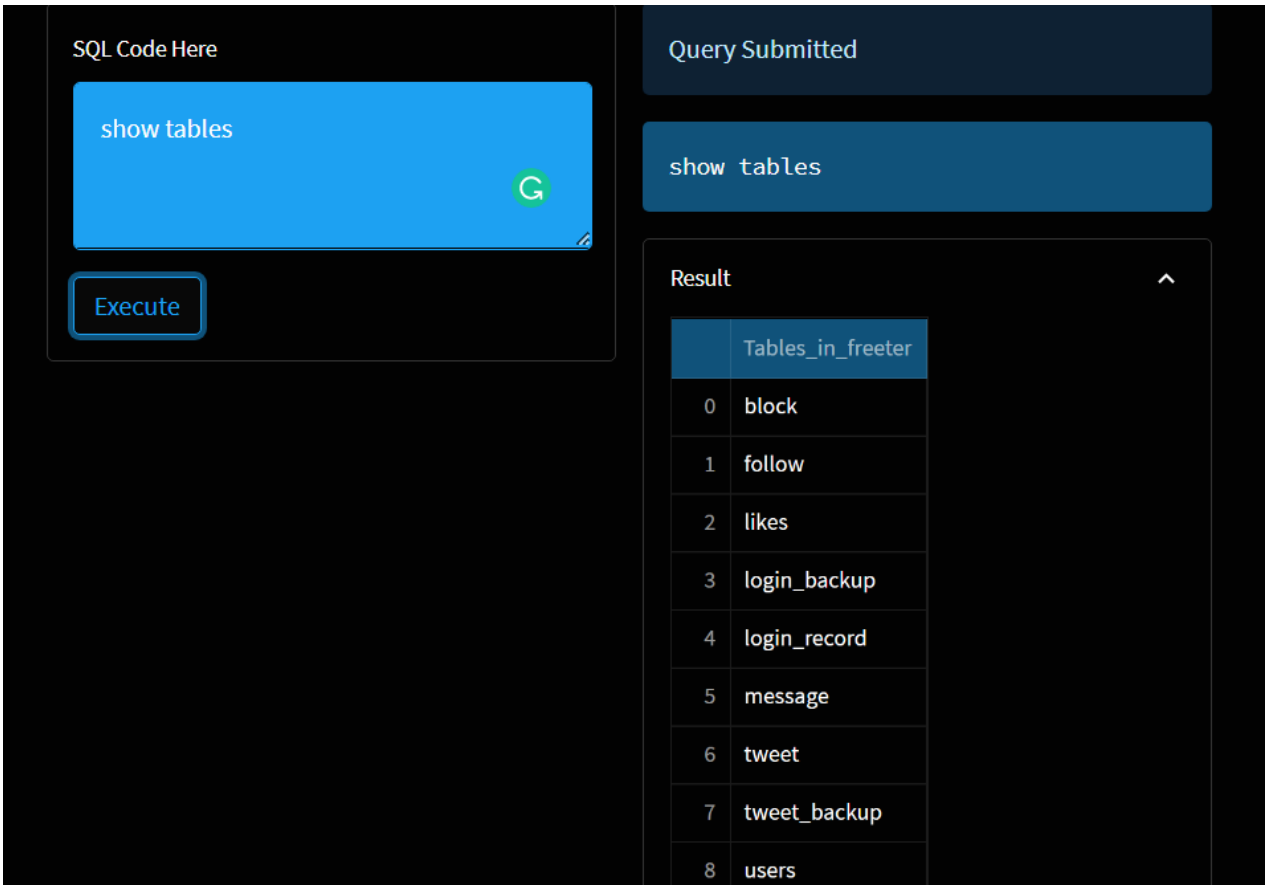

Enter the username of the person you want to unfollow:
abcd
Unfollow
Show following
Successful! you are now unfollowing abcd


Enter the username of the person you want to unfollow:
abcd
Unfollow
Show following
empty

2. **Window to accept SQL commands**



	username	firstName	lastName
0	abcd	first	last
1	admin	admin	admin
2	afallawe9	Anni	Fallawe
3	ahymas1	Avie	Hymas
4	cgallaher3	Constancy	Gallaher
5	cnother4	Concettina	Nother
6	eretchless7	Elicia	Retchless
7	jhearsey2	Jeff	Hearsey
8	jphizackarley6	Jessee	Phizackarley



	Tables_in_freeter
0	block
1	follow
2	likes
3	login_backup
4	login_record
5	message
6	tweet
7	tweet_backup
8	users