# DBMS - Mini Project

Freds - A Clone of Thread

**Submitted by:**

**YANGALA NIKITHA - PES1UG21CS728**
**YASHAS B - PES2UG21CS621**

**V Semester Section L**

## Table of contents:

## Short Description and Scope of the Project

The project is a free social media platform which serves as a community group also where the users can interact with their friends and alsobe able to look at the vast majority of the announcements/events even if they are not followers of the person.
Freds 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.

The features include:

1. Login/Sign up: The home page consists of a login/signup form which creates a new account if the user is new or allows them to login if they are already existing user.
2. Direct message: Direct message or dm in short is the main mode of communication which is widely used nowadays, hence this feature allows the user to interact with other users by sending them private messages.
3. Followers: The users are given an option which allows them to follow other users or block users whom they feel are pestering them.
4. SQL reader: An inbuilt sql terminal which can be used to communicate with the backend and fetch results and display them in a pretty table.
5. Thread: The main feature of this platform where the users can speak their mind out, butonly within the guidelines to ensure that the platform is safe for everyone to use and interact with.
6. Community/Group: The second main feature where people can be aware of their community happenings and be updated.
7. There is also a facility for an admin account which can keep track of all the accounts login information.

In terms of future additions, I'd be implementing a mechanism for the admins to remove a user who has been blocked by lots of other users indicating that person might have been toxic.

## SOFTWARE AND TOOLS

**PROGRAMMING LANGUAGE - PYTHON**

**DATABASE - MYSQL**

**TOOLS   - MYSQL WORKBENCH AND COMMAND PROMPT, STREAMLIT**

## E R Diagram

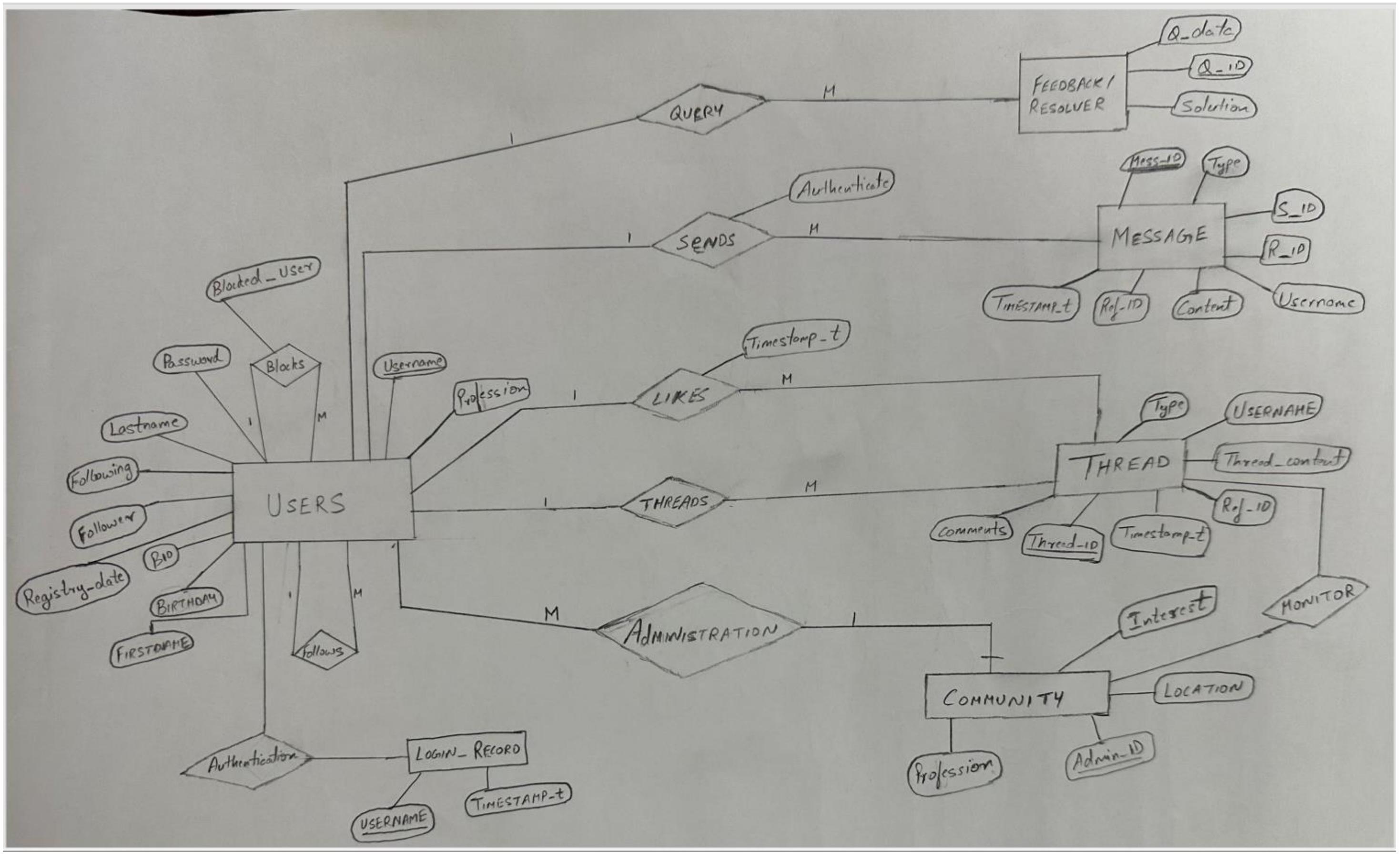


*Figure 1 ER diagram*

## Relational Schema

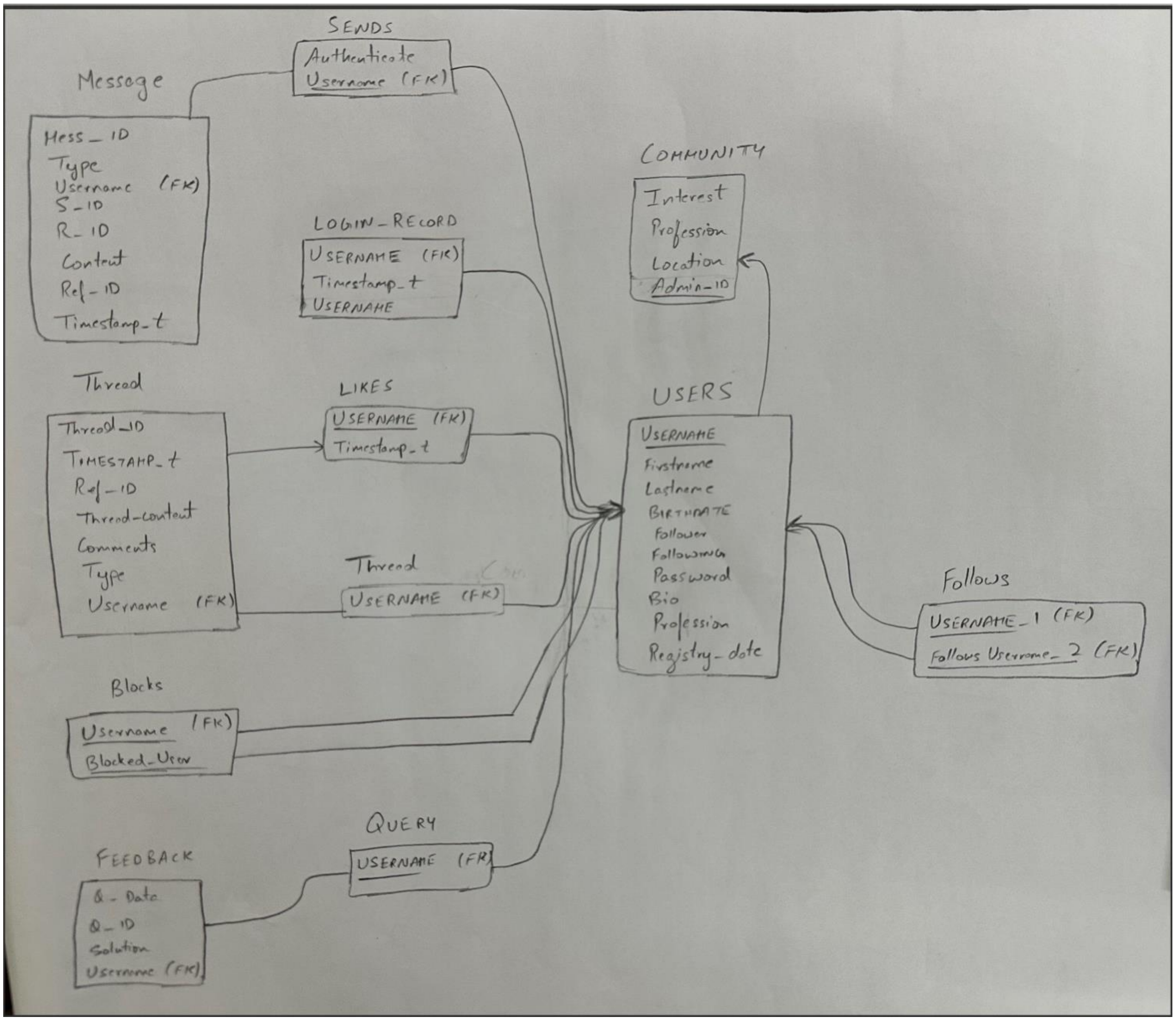
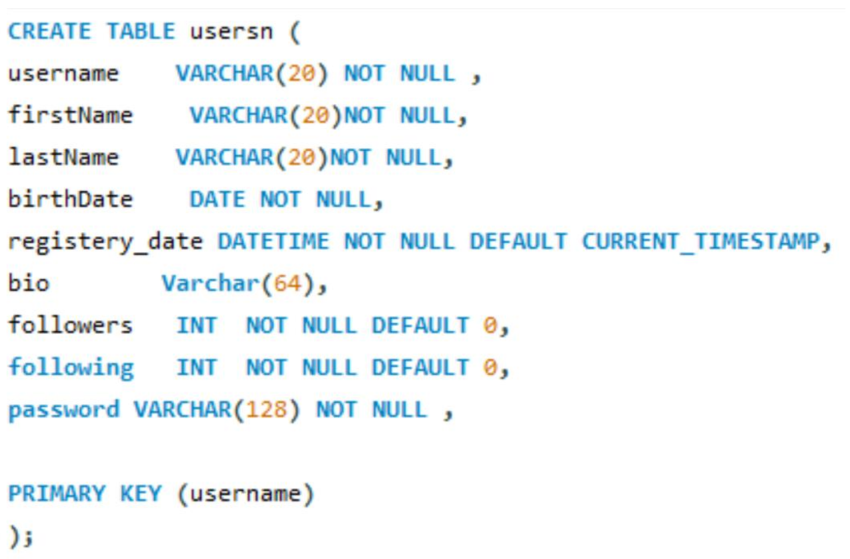


*Figure 2 Relational Schema Diagram*

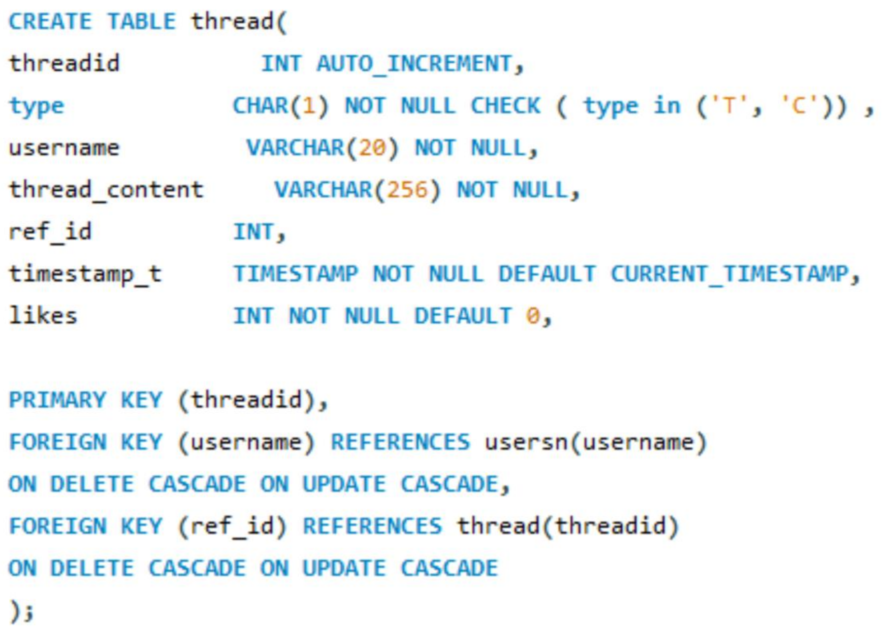
## DDL statements - Building the database



```
CREATE TABLE usersn (
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)
);
```

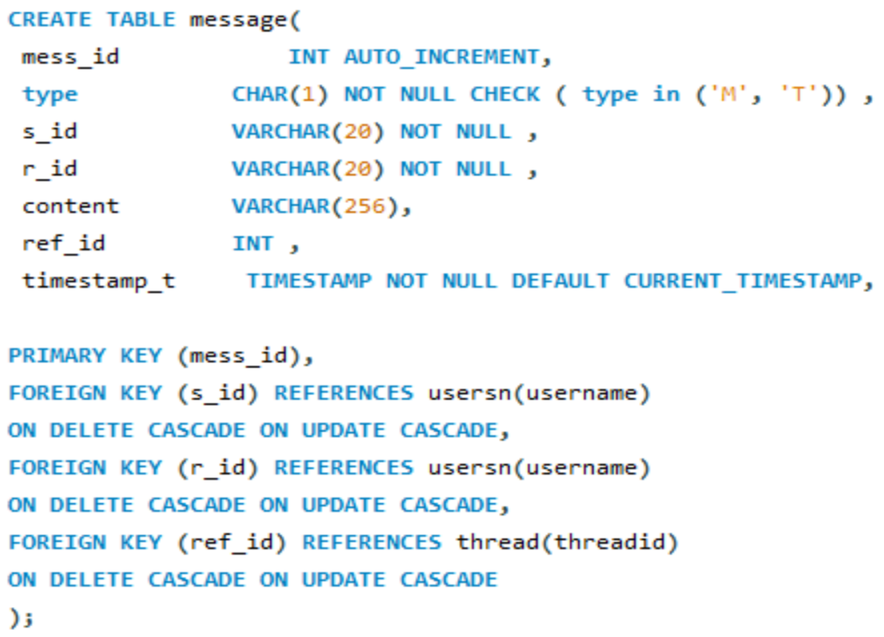
*Figure 3 Creation of table users*



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

PRIMARY KEY (threadid),
FOREIGN KEY (username) REFERENCES usersn(username)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (ref_id) REFERENCES thread(threadid)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

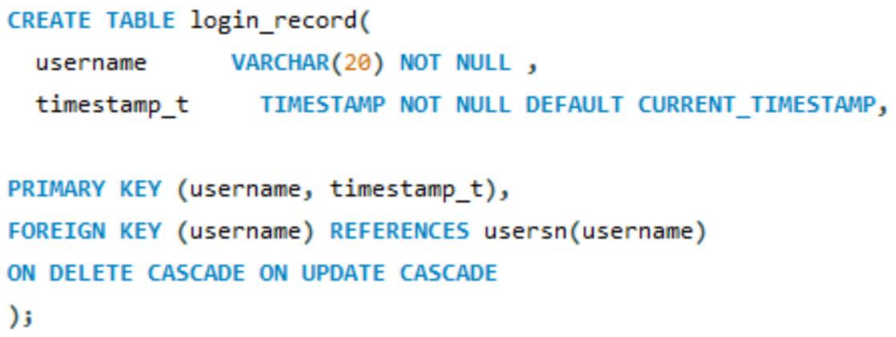
*Figure 4 Creation of table thread*



```
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 usersn(username)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (r_id) REFERENCES usersn(username)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (ref_id) REFERENCES thread(threadid)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

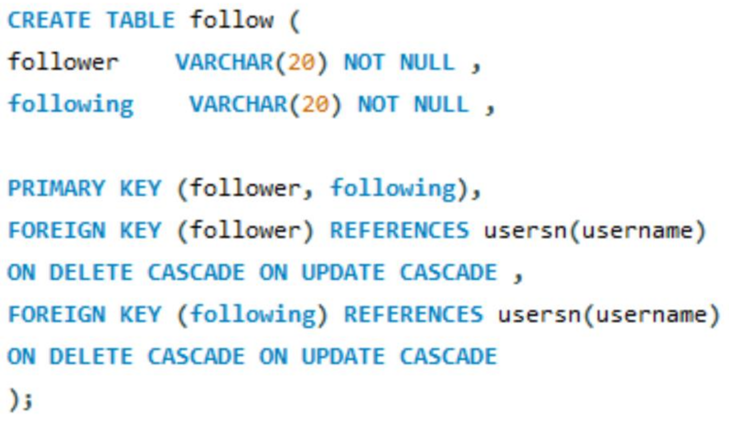
*Figure 5 Creation of table message*



```
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 usersn(username)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

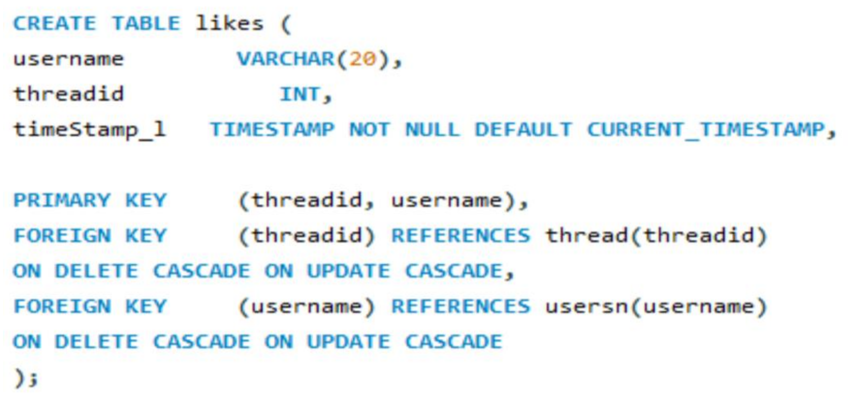
*Figure 6 Creation of table login_record*



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

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

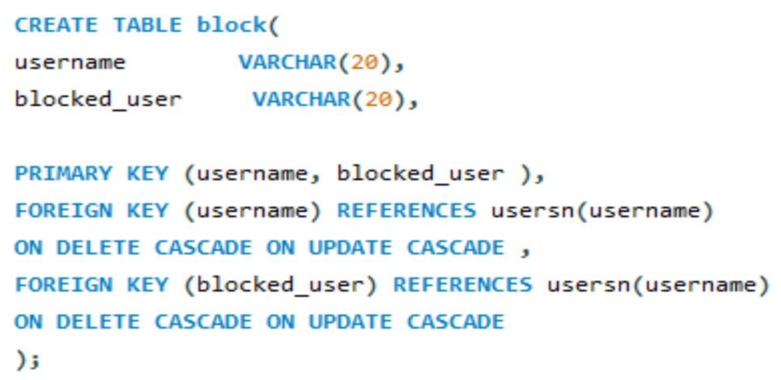
*Figure 7 Creation of table follow*



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

PRIMARY KEY      (threadid, username),
FOREIGN KEY      (threadid) REFERENCES thread(threadid)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY      (username) REFERENCES usersn(username)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

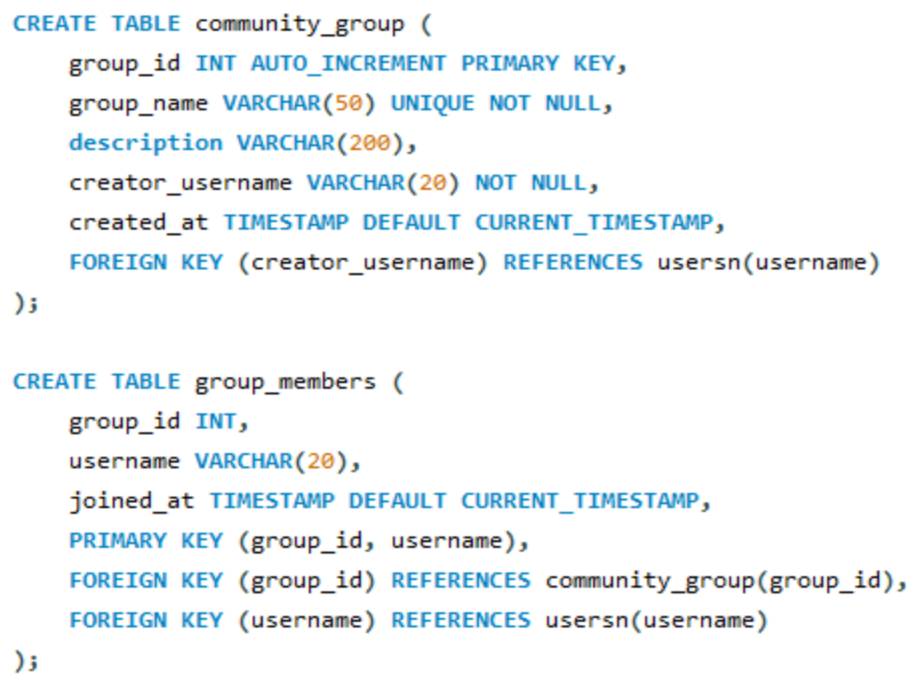
*Figure 8 Creation of table likes*



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

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

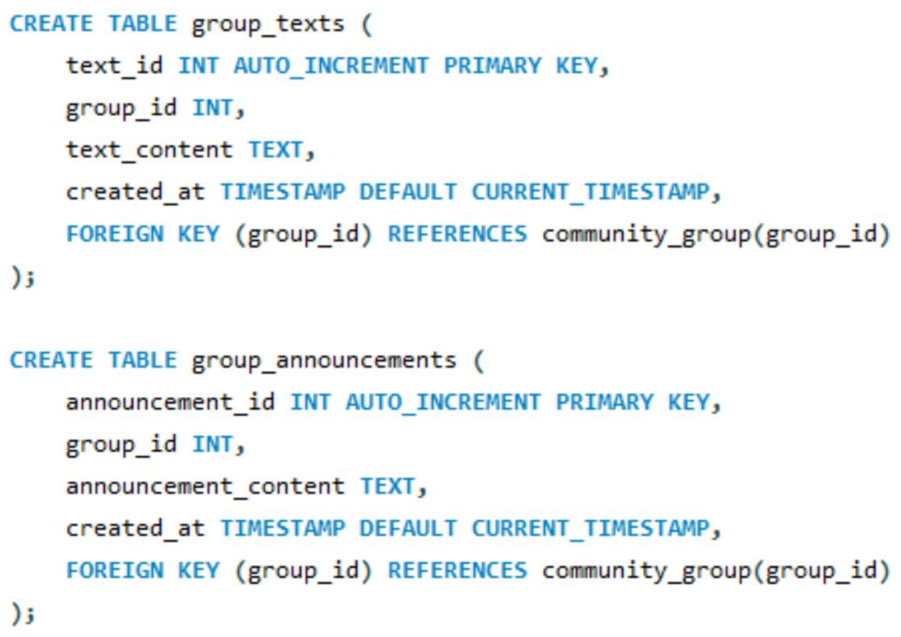
*Figure 9 Creation of table block*



```
CREATE TABLE community_group (
    group_id INT AUTO_INCREMENT PRIMARY KEY,
    group_name VARCHAR(50) UNIQUE NOT NULL,
    description VARCHAR(200),
    creator_username VARCHAR(20) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (creator_username) REFERENCES usersn(username)
);

CREATE TABLE group_members (
    group_id INT,
    username VARCHAR(20),
    joined_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (group_id, username),
    FOREIGN KEY (group_id) REFERENCES community_group(group_id),
    FOREIGN KEY (username) REFERENCES usersn(username)
);
```

*Figure 10 Creation of table for community group and group_members*



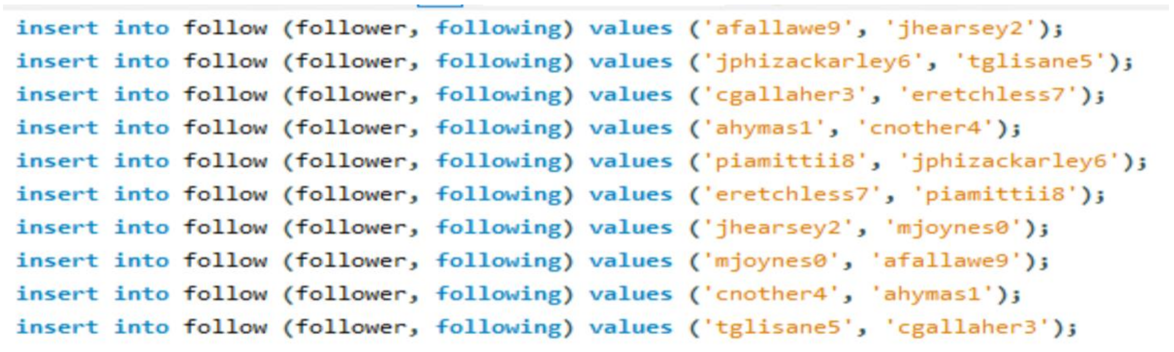
```sql
CREATE TABLE group_texts (
    text_id INT AUTO_INCREMENT PRIMARY KEY,
    group_id INT,
    text_content TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (group_id) REFERENCES community_group(group_id)
);

CREATE TABLE group_announcements (
    announcement_id INT AUTO_INCREMENT PRIMARY KEY,
    group_id INT,
    announcement_content TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (group_id) REFERENCES community_group(group_id)
);
```

*Figure 11 Creation of table for group_texts and group_announcements*

Populating the Database - **CRUD OPERATIONS**

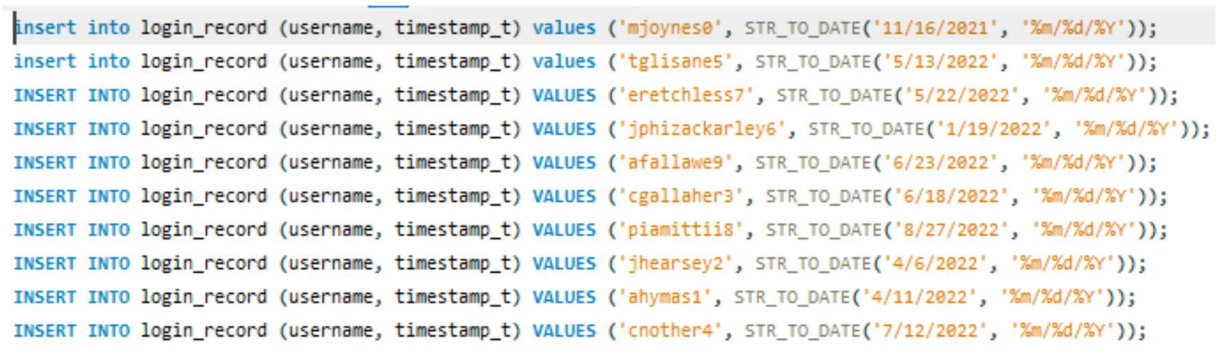
insert into follow (follower, following) values ('afallawe9', 'jhearsey2');
insert into follow (follower, following) values ('jphizackarley6', 'tglisane5');
insert into follow (follower, following) values ('cgallaher3', 'eretchless7');
insert into follow (follower, following) values ('ahymas1', 'cnother4');
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 ('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');
```

*Figure 12 Adding values into the table follow*

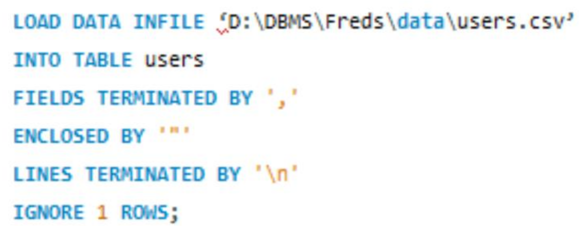
insert into login_record (username, timestamp_t) values ('mjoynes0', '2021-11-16');
insert into login_record (username, timestamp_t) values ('tglisane5', '2022-5-13');
insert into login_record (username, timestamp_t) values ('eretchless7', '2022-5-22');
insert into login_record (username, timestamp_t) values ('jphizackarley6', '2022-1-19');
insert into login_record (username, timestamp_t) values ('afallawe9', '2022-6-23');
insert into login_record (username, timestamp_t) values ('cgallaher3', '2022-6-18');



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

*Figure 13 Adding values into the table login_record*

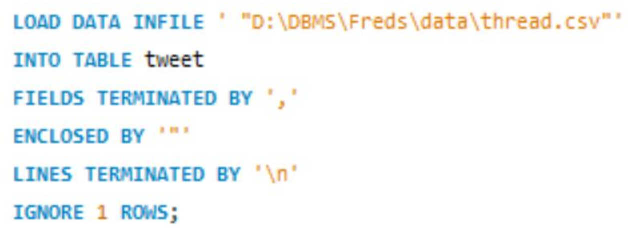
```
LOAD DATA INFILE ‘D:\DBMS\Freds\data\users.csv’
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```



```
LOAD DATA INFILE ‘D:\DBMS\Freds\data\users.csv’
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

*Figure 14 Adding Users*

```
LOAD DATA INFILE ' "D:\DBMS\Freds\data\thread.csv"'
INTO TABLE thread
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```



```
LOAD DATA INFILE ' "D:\DBMS\Freds\data\thread.csv"'
INTO TABLE tweet
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

*Figure 15 Adding threads data*

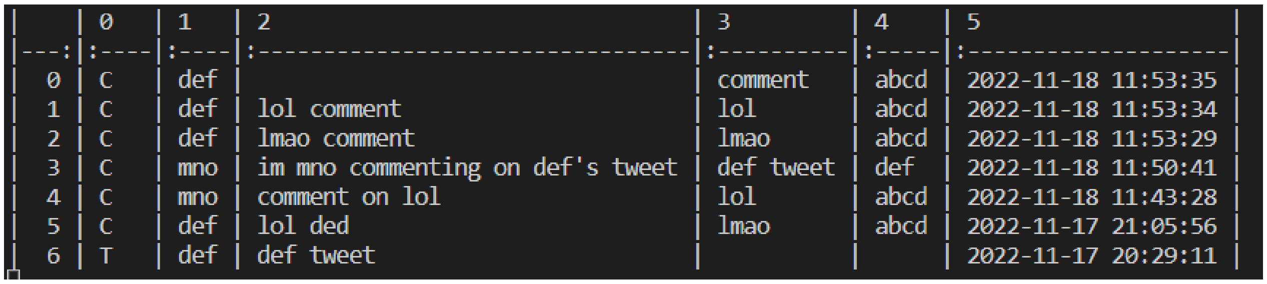
## Join Queries

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

**1.Find the activities of the users that another user is following**

```
   SELECT y.type, y.username, y.thread_content , y.cc AS ref_content, y.us
ASref_username, y.timestamp_t
   FROM follow, ( SELECT thread.threadid, thread.type, thread.username,
thread.thread_content, thread.ref_id, thread.timestamp_t, thread.likes, t.thread_content
AS cc,t.username AS us
               FROM thread LEFT JOIN thread
               AS tON thread.ref_id = t.threadid)
               as y
   WHERE follow.following = y.username AND follow.follower = person AND
y.username NOT IN(
SELECT block.usernameFROM block
WHERE blocked_user = person)
   ORDER BY y.timestamp_t DESC ;
```

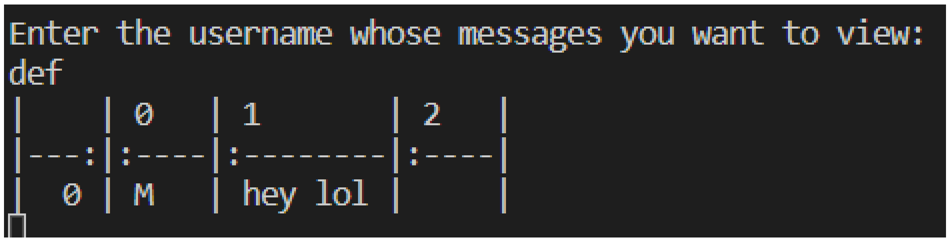


```
|    | 0   | 1   | 2                               | 3          | 4     | 5                   |
|---:|:----|:----|:--------------------------------|:-----------|:------|:--------------------|
|  0 | C   | def |                                 | comment    | abcd  | 2022-11-18 11:53:35 |
|  1 | C   | def | lol comment                     | lol        | abcd  | 2022-11-18 11:53:34 |
|  2 | C   | def | lmao comment                    | lmao       | abcd  | 2022-11-18 11:53:29 |
|  3 | C   | mno | im mno commenting on def's tweet | def tweet  | def   | 2022-11-18 11:50:41 |
|  4 | C   | mno | comment on lol                  | lol        | abcd  | 2022-11-18 11:43:28 |
|  5 | C   | def | lol ded                         | lmao       | abcd  | 2022-11-17 21:05:56 |
|  6 | T   | def | def tweet                       |            |       | 2022-11-17 20:29:11 |
```

*Figure 16 Find following activity*

**2.Find the message sent by a specific user**

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

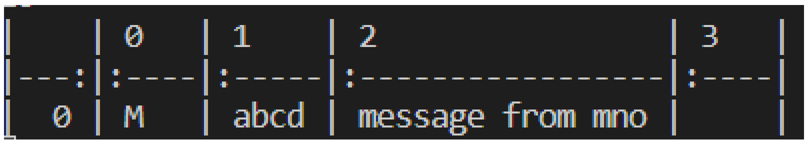


```
Enter the username whose messages you want to view:
def
|    | 0    | 1        | 2    |
|---:|:-----|:---------|:-----|
|  0 | M    | hey lol  |      |
```

Figure 17 Find specific user message

**3.Find all the messages the user has received**

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



```
|    | 0    | 1     | 2                  | 3    |
|---:|:-----|:------|:-------------------|:-----|
|  0 | M    | abcd  | message from mno   |      |
```

Figure 18 Find all the direct messages

**4.List out all the current users threads and their replies**

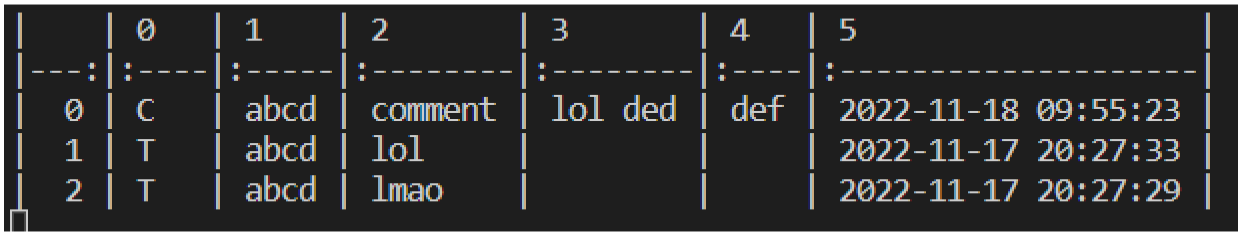
SELECT thread.type ,thread.thread_content, t.thread_content AS refrence_content,t.username AS refrence_username, thread.timestamp_t

FROM thread LEFT JOIN thread

as tON thread.ref_id = t.threadid

WHERE  thread.username = person

ORDER BY thread.timestamp_t DESC ;



```
|    | 0    | 1     | 2        | 3        | 4    | 5                   |
|---:|:-----|:------|:---------|:---------|:-----|:--------------------|
|  0 | C    | abcd  | comment  | lol ded  | def  | 2022-11-18 09:55:23 |
|  1 | T    | abcd  | lol      |          |      | 2022-11-17 20:27:33 |
|  2 | T    | abcd  | lmao     |          |      | 2022-11-17 20:27:29 |
```

Figure 19 Find all threads and replies

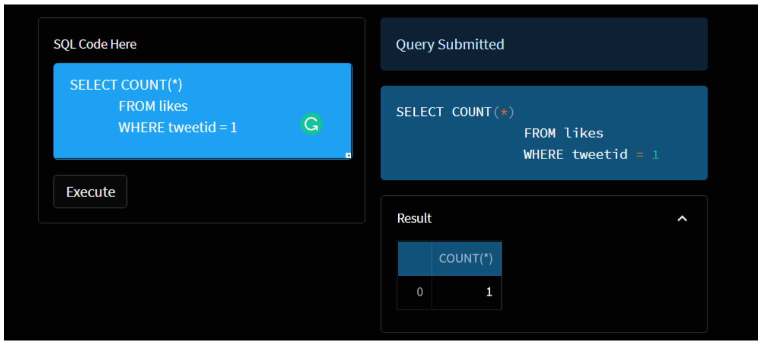
## Aggregate Functions

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

### 1.Show the number of likes on a thread

```
SELECT COUNT(*)
FROM likes
WHERE threadid = p_threadid
```



SQL Code Here

```
SELECT COUNT(*)
        FROM likes
        WHERE tweetid = 1
```

Execute

Query Submitted

```
SELECT COUNT(*)
        FROM likes
        WHERE tweetid = 1
```

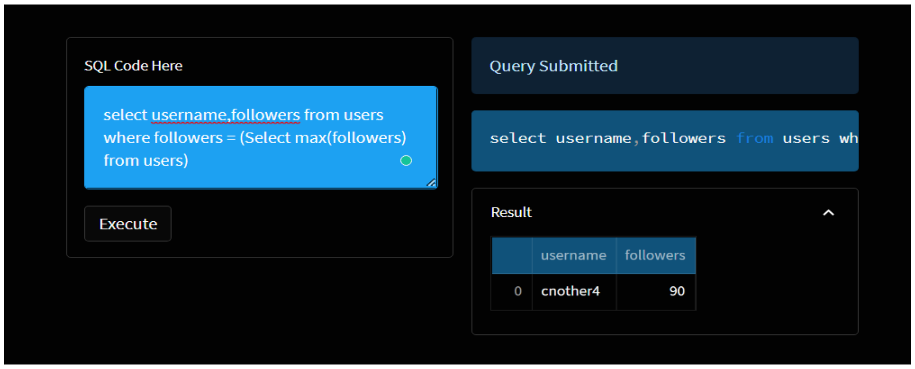
Result

|   | COUNT(*) |
|---|---|
| 0 | 1 |

*Figure 20 Number of likes table*

### 2. User with the most number of followers

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



SQL Code Here

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

Execute

Query Submitted

```
select username,followers from users wh
```

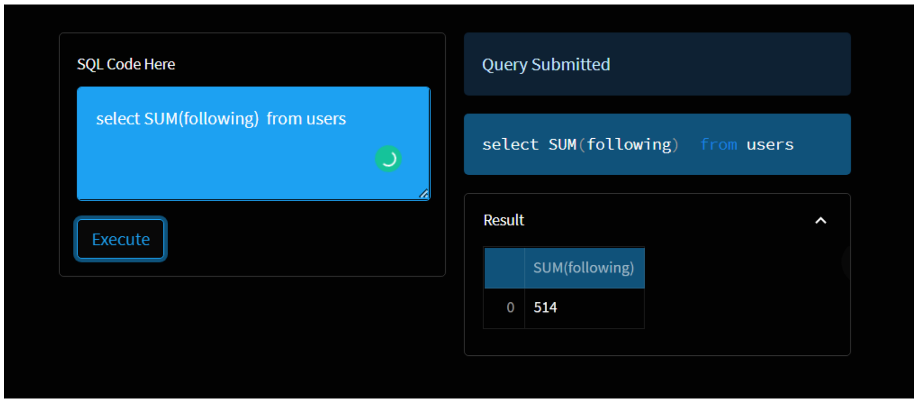
Result

|   | username | followers |
|---|---|---|
| 0 | cnother4 | 90 |

*Figure 21 Most number of followers table*

### 3.Count of all followers in the app

select SUM(following)  from users



```
select SUM(following)  from users
```

|   | SUM(following) |
|---|---|
| 0 | 514 |

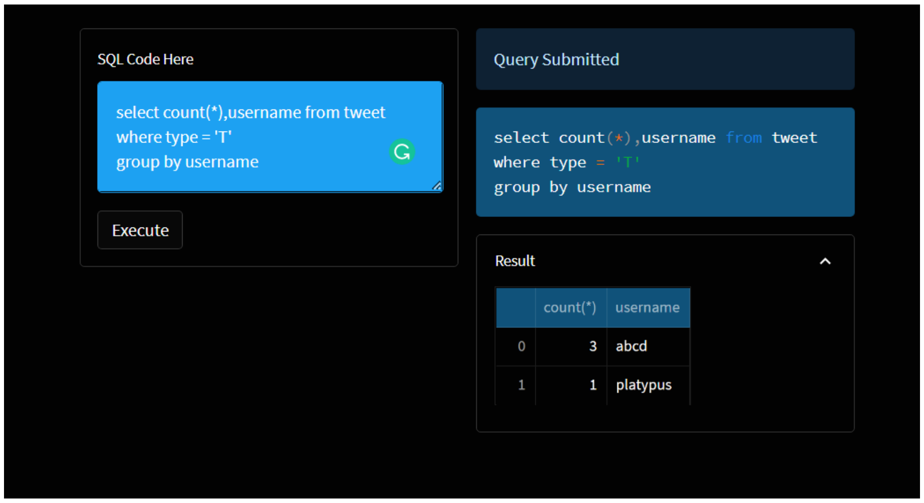
*Figure 22 Sum of all users*

### 4.Count of all threads grouped by usernames

select count(*),username from

threadwhere type = 'T'

group by username



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

|   | count(*) | username |
|---|---|---|
| 0 | 3 | abcd |
| 1 | 1 | platypus |

*Figure 23 All the threads table*

## Set Operations

Showcase at least 4 Set Operations queries

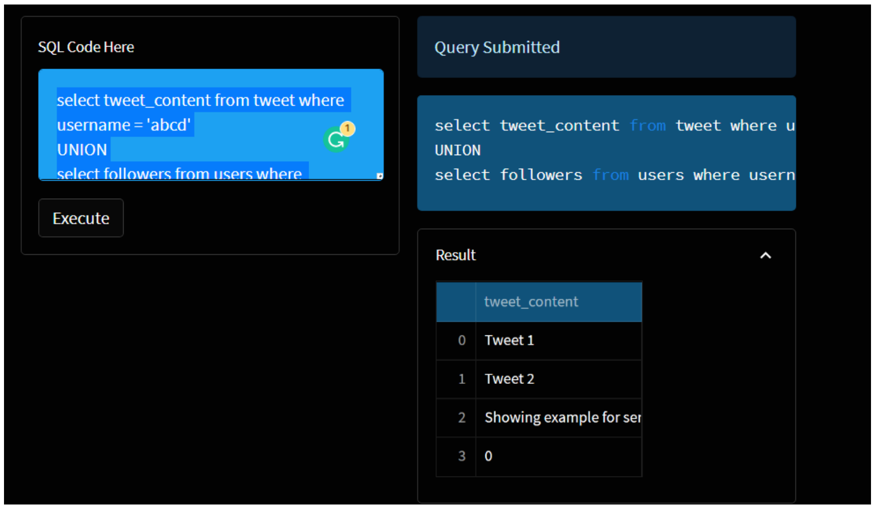
Write the query in English Language, Show the equivalent SQL statement and also ascreenshot of the query and the results

### 1.Find the number of followers along with the threads for a particular user

select thread_content from thread where username =

'abcd'UNION

select followers from users where username = 'abcd'



SQL Code Here

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

Execute

Query Submitted

```
select tweet_content from tweet where u
UNION
select followers from users where usern
```

Result

```
   tweet_content
0  Tweet 1
1  Tweet 2
2  Showing example for ser
3  0
```

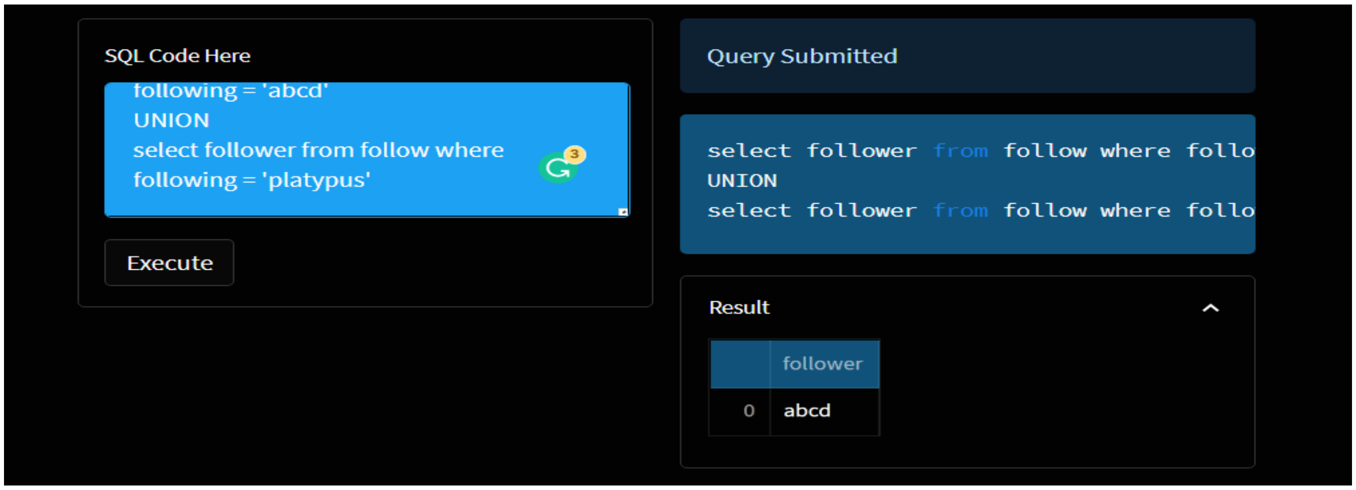
Figure 24 Union of total foillowers and threads table

### 2.Finding which all users have same followers

select follower from follow where following = 'abcd'

UNION

select follower from follow where following = 'platypus'



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

```
select follower from follow where follo
UNION
select follower from follow where follo
```

|   | follower |
|---|---|
| 0 | abcd |

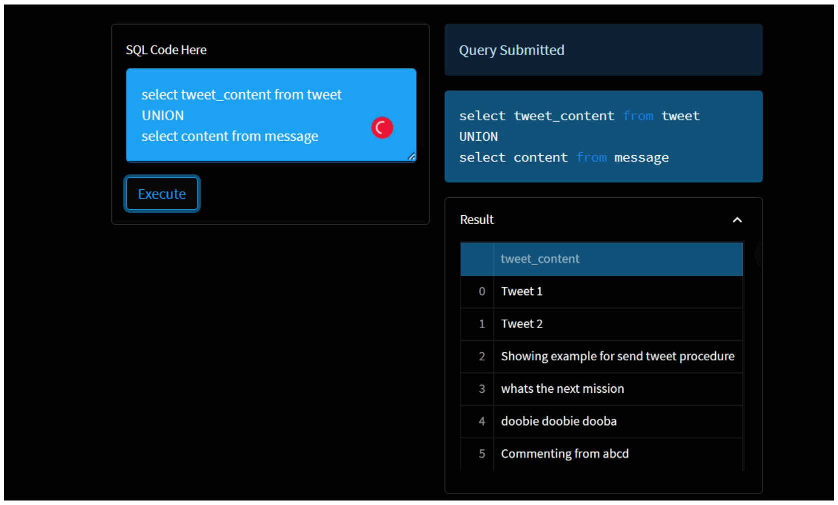
Figure 25 Union of all common followers

### 3. All the messages and the threads in the app

select thread_content from

threadUNION

select content from message



```
select tweet_content from tweet
UNION
select content from message
```

|   | tweet_content |
|---|---|
| 0 | Tweet 1 |
| 1 | Tweet 2 |
| 2 | Showing example for send tweet procedure |
| 3 | whats the next mission |
| 4 | doobie doobie dooba |
| 5 | Commenting from abcd |

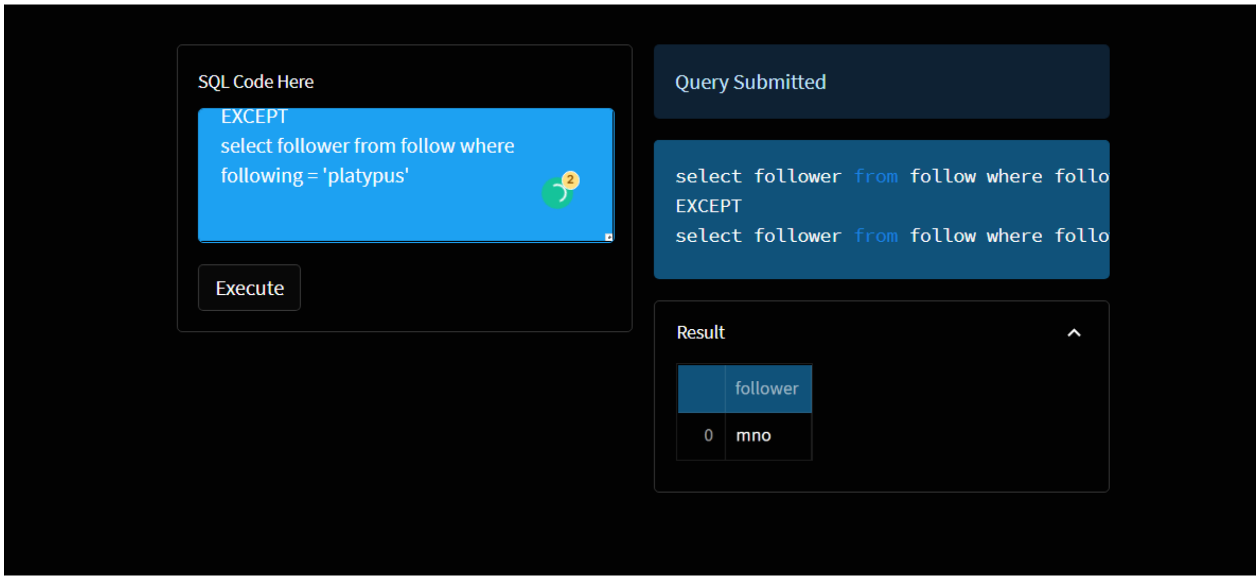
Figure 26 Union of all the threads and messages

### 4. Users which are not common among the followers

select follower from follow where following = 'abcd'

EXCEPT

select follower from follow where following = 'platypus'



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

```
select follower from follow where follo
EXCEPT
select follower from follow where follo
```

|   | follower |
|---|---|
| 0 | mno |

*Figure 27 Uncommon followers table*

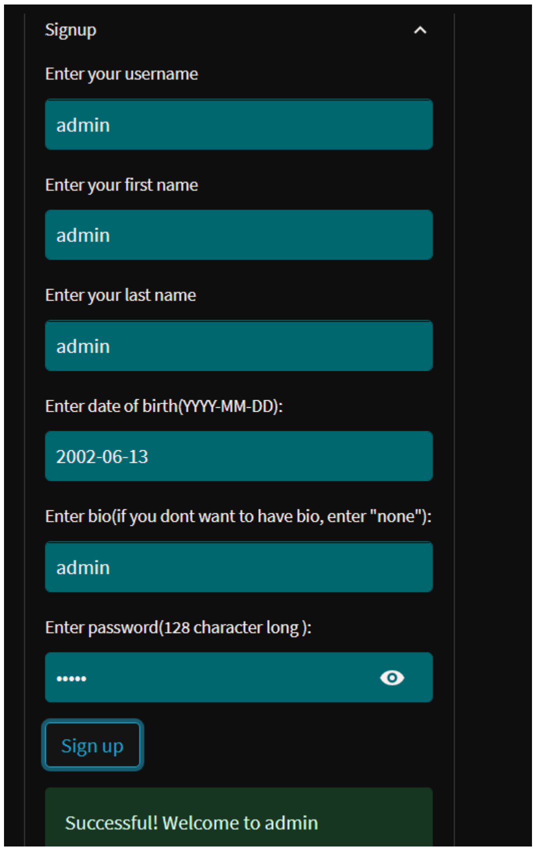
## Functions and Procedures

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

**1.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 //
```


Signup

Enter your username

admin

Enter your first name

admin

Enter your last name

admin

Enter date of birth(YYYY-MM-DD):

2002-06-13

Enter bio(if you dont want to have bio, enter "none"):

admin

Enter password(128 character long ):

•••••

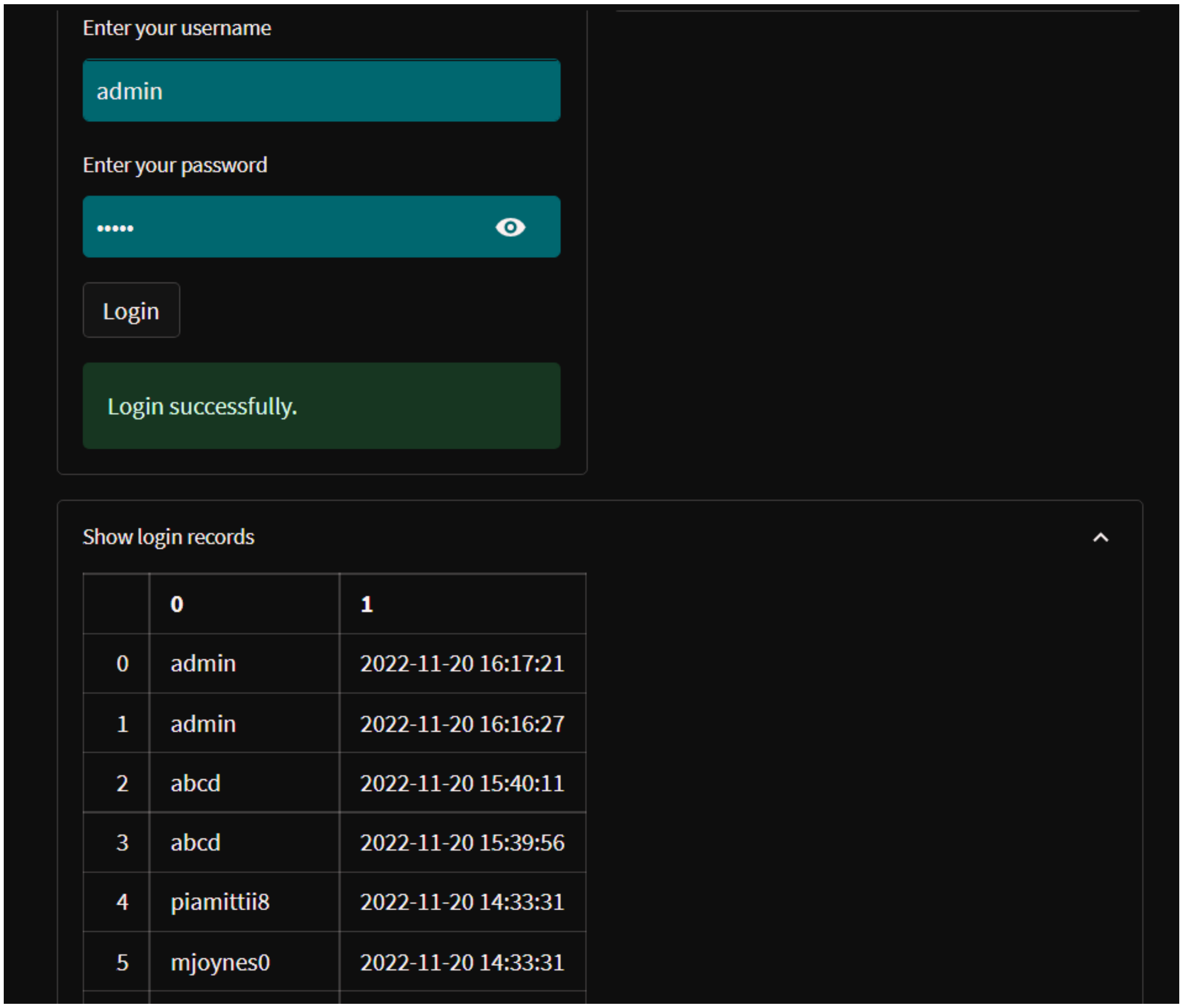
Sign up

Successful! Welcome to admin

*Figure 28 Sign up page which call a create_account procedure*

**2.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 //
```



Enter your username

admin

Enter your password

•••••

Login

Login successfully.

Show login records

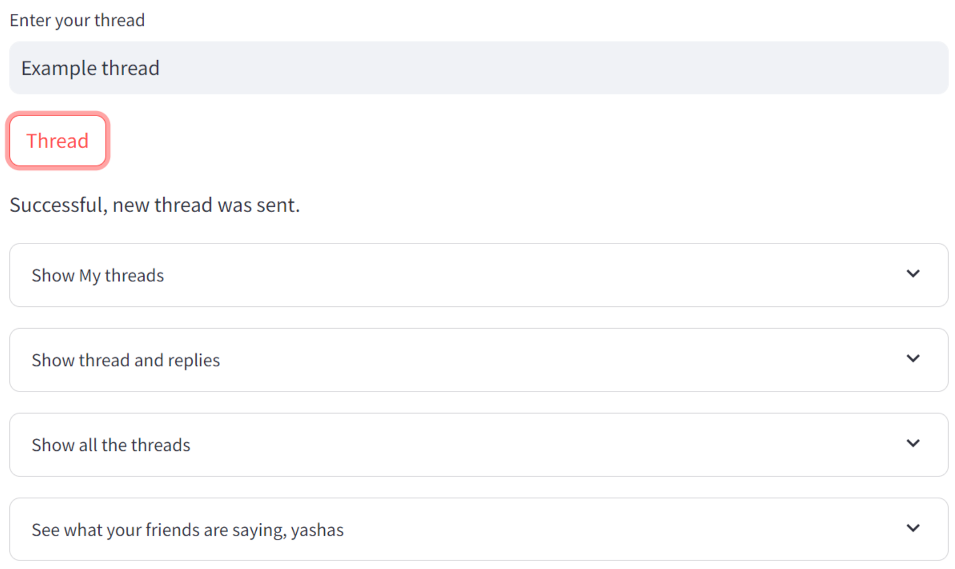
| | 0 | 1 |
|---|---|---|
| 0 | admin | 2022-11-20 16:17:21 |
| 1 | admin | 2022-11-20 16:16:27 |
| 2 | abcd | 2022-11-20 15:40:11 |
| 3 | abcd | 2022-11-20 15:39:56 |
| 4 | piamittii8 | 2022-11-20 14:33:31 |
| 5 | mjoynes0 | 2022-11-20 14:33:31 |

*Figure 29 Admin can see login records*

**3.Send thread sends a thread which has been written by the users and is made available to allthe users of the app**

```
CREATE PROCEDURE
   send_thread(IN p_content
   VARCHAR(256)
)
BEGIN
   DECLARE person VARCHAR(20);
   CALL  find_subject(person);
```

```
    INSERT INTO thread(type, username,
    thread_content)VALUES ('T', person, p_content);
    SELECT 'Successful, new thread was sent.' AS
mess;end //
```


Enter your thread

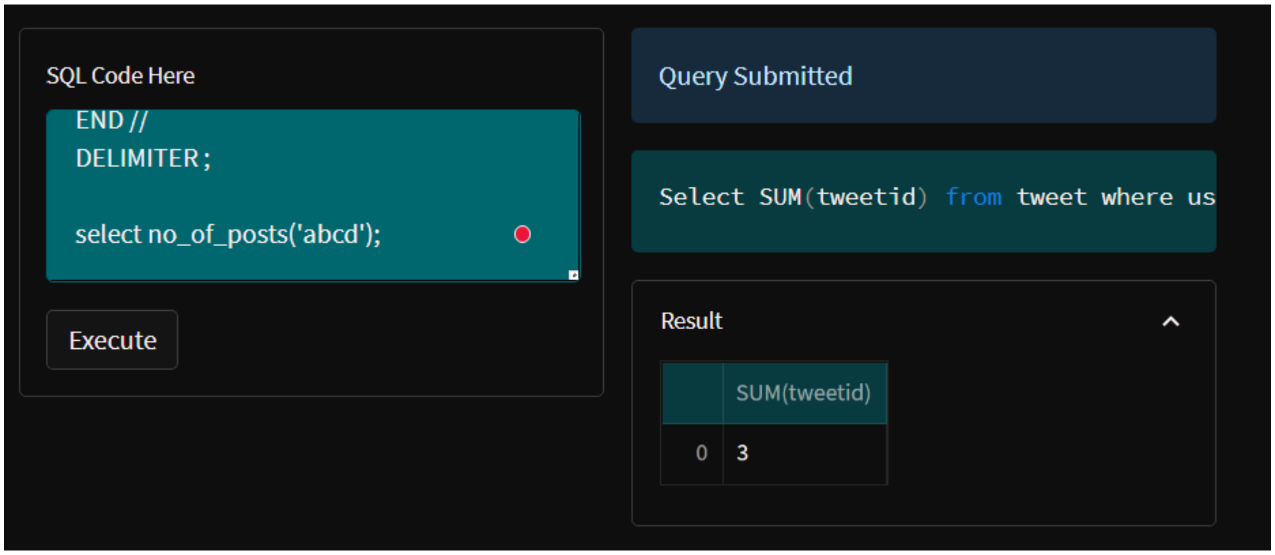
Example thread

Successful, new thread was sent.

*Figure 30 Sending a new thread*

**4.Find the number of threads a particular user has threaded**

```
DELIMITER //
CREATE FUNCTION no_of_posts(uname char) RETURNS INT DETERMINISTIC
BEGIN
          DECLARE threads INT;
    Select SUM(threadid) INTO threads from thread where username =
    uname ;return threads;
END //
DELIMITER ;
select no_of_posts('abcd');
```



```
END //
DELIMITER ;

select no_of_posts('abcd');
```

```
Select SUM(tweetid) from tweet where us
```

|   | SUM(tweetid) |
|---|---|
| 0 | 3 |

*Figure 31 Function which calculates the number of posts*

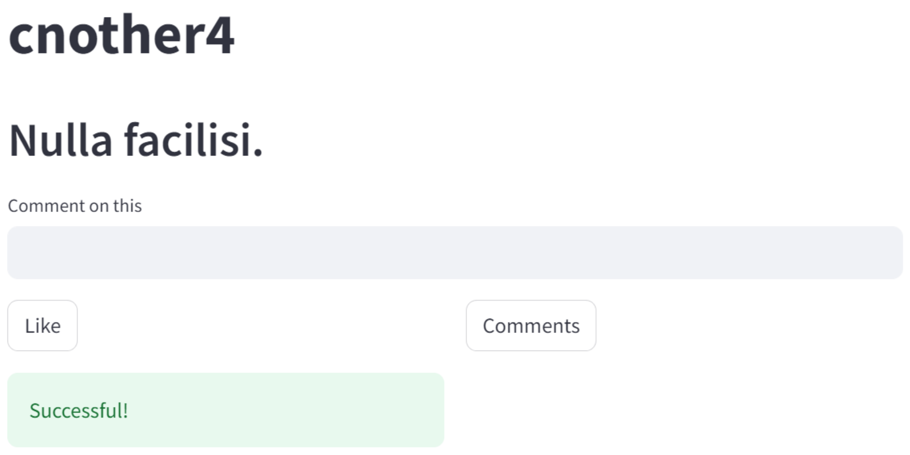
## Triggers and Cursors

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

**1.The following trigger auto_like updates the table thread which contain all the threads andincrements the number of likes**

```
DELIMITER //


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


cnother4

Nulla facilisi.

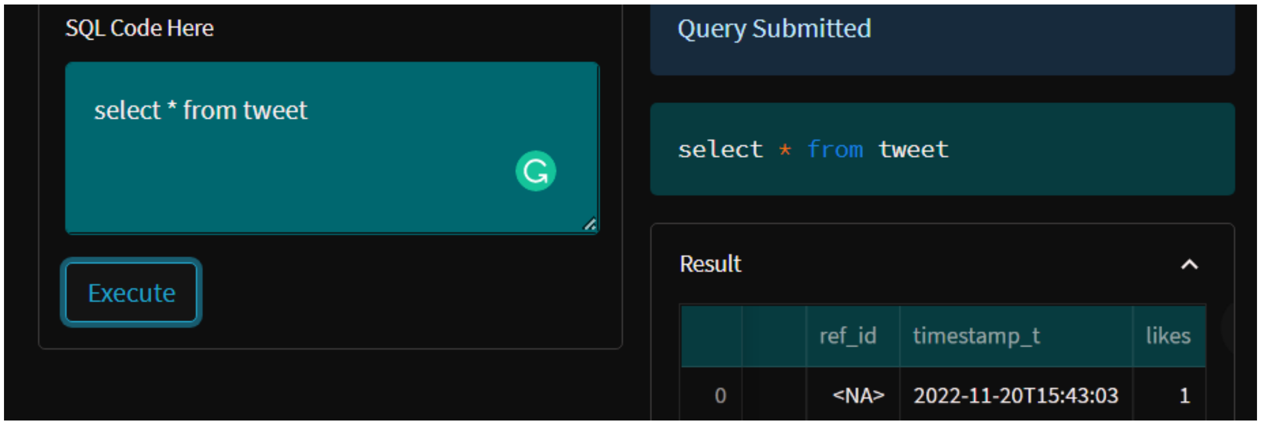
Comment on this

Like

Comments

Successful!

*Figure 32 Text input box for the user to enter the thread*



```
select * from tweet
```

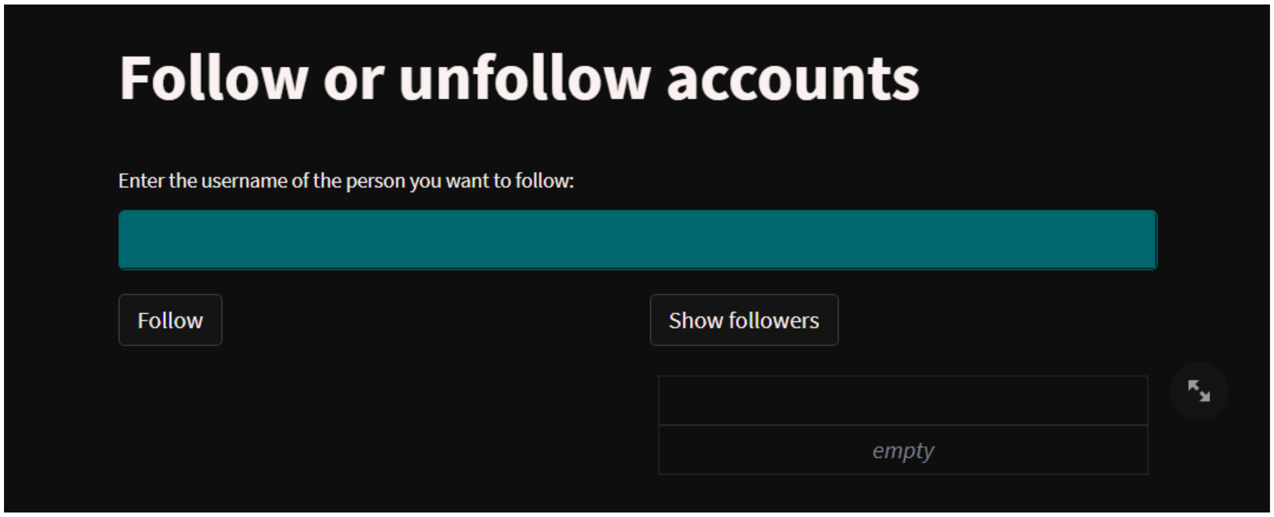
```
select * from tweet
```

|   |   | ref_id | timestamp_t | likes |
|---|---|---|---|---|
| 0 |   | <NA> | 2022-11-20T15:43:03 | 1 |

*Figure 33 SQL terminal embedded inside front end to retrieve data*

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

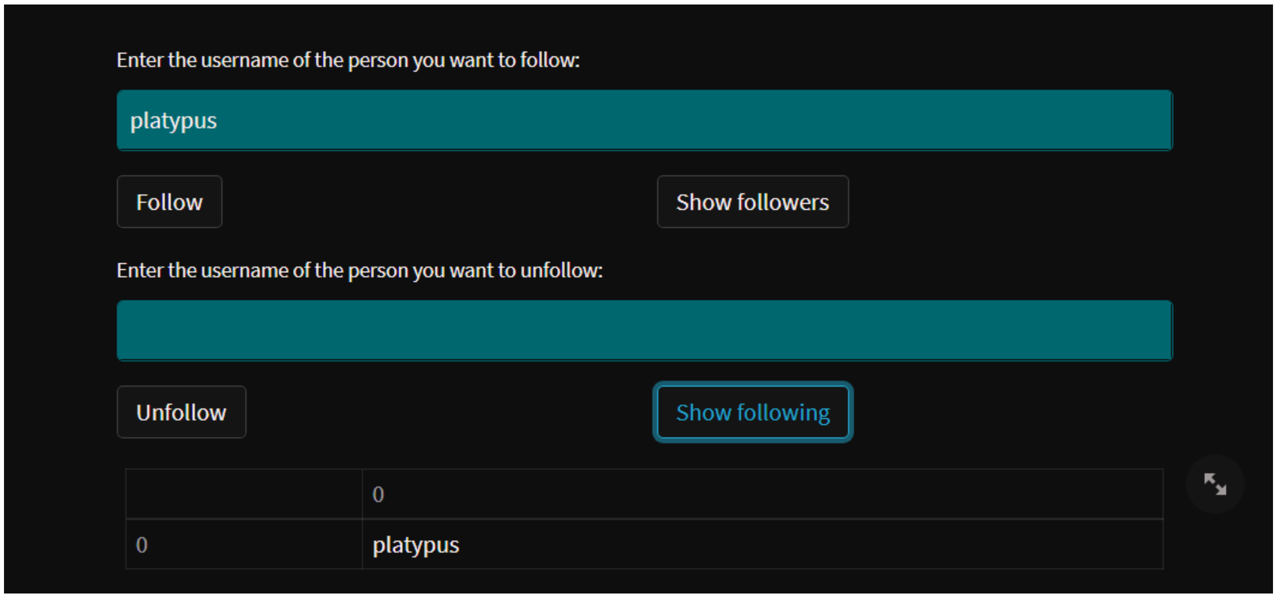
```
CREATE TRIGGER auto_follow
BEFORE INSERT
ON follow FOR EACH ROW
   BEGIN
      DECLARE follower_temp VARCHAR(20);
      DECLARE following_temp VARCHAR(20);
      SET follower_temp = NEW.follower;
      SET following_temp = NEW.following;
      UPDATE users SET following = users.following + 1 WHERE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



Follow or unfollow accounts

Enter the username of the person you want to follow:

Figure 34 Input text area for the user to enter another users name to follow



Enter the username of the person you want to follow:

platypus

Enter the username of the person you want to unfollow:

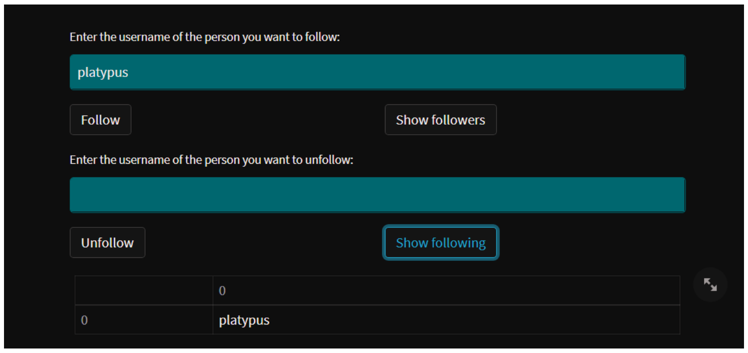
| | 0 |
|---|---|
| 0 | platypus |

Figure 35 Table containing list of following users

**3. 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 //
```

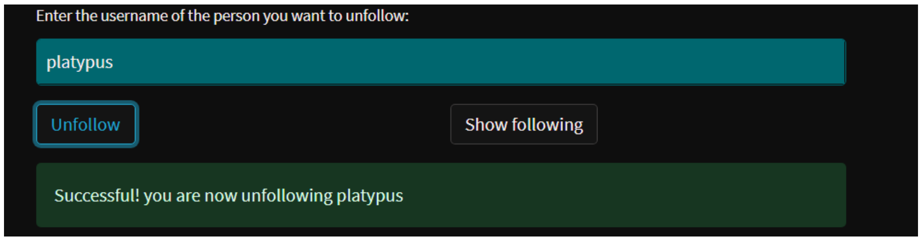

Enter the username of the person you want to follow:

platypus

Enter the username of the person you want to unfollow:

| | 0 |
|---|---|
| 0 | platypus |

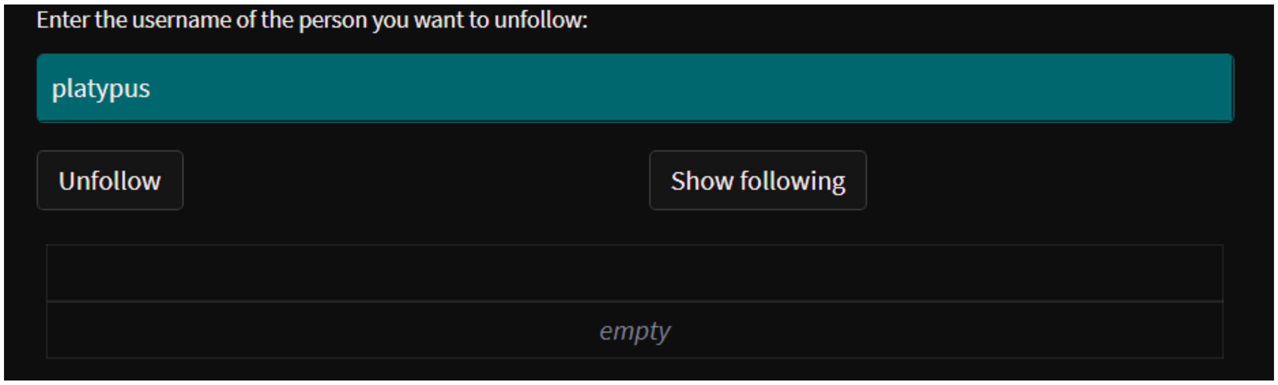
Figure 36 Unfollow user


Enter the username of the person you want to unfollow:

platypus

Successful! you are now unfollowing platypus

Figure 37 Username entered to unfollow


Enter the username of the person you want to unfollow:

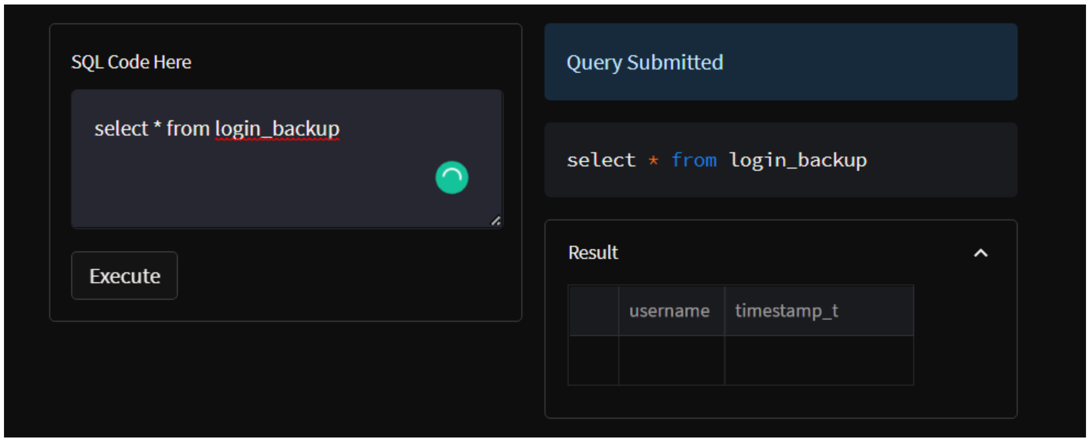
platypus

*Figure 38 Table after the user has unfollowed*

**4.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;

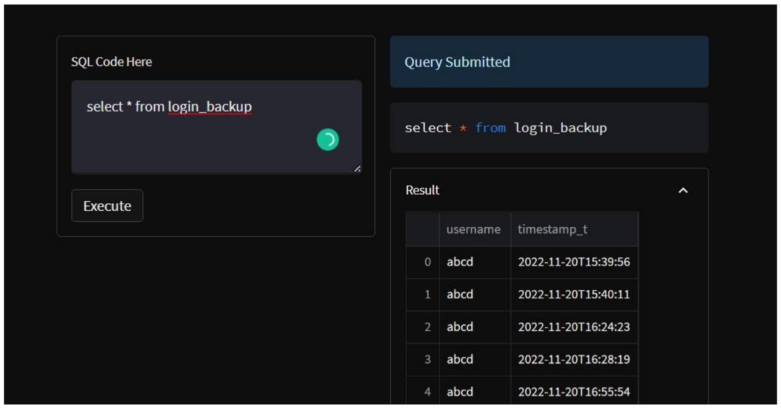


```
select * from login_backup
```

|  | username | timestamp_t |
|---|---|---|
|  |  |  |

*Figure 39 The inbuilt terminal showing the empty table*

After CALL log_back;



```
select * from login_backup
```

|  | username | timestamp_t |
|---|---|---|
| 0 | abcd | 2022-11-20T15:39:56 |
| 1 | abcd | 2022-11-20T15:40:11 |
| 2 | abcd | 2022-11-20T16:24:23 |
| 3 | abcd | 2022-11-20T16:28:19 |
| 4 | abcd | 2022-11-20T16:55:54 |

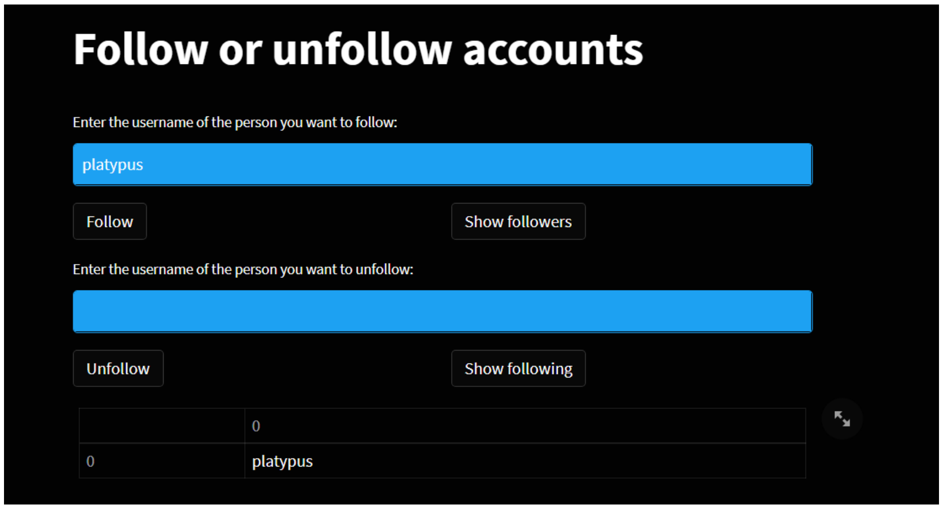
*Figure 40 The table filled after calling procedure which uses callback*

## 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



Follow or unfollow accounts

Enter the username of the person you want to follow:

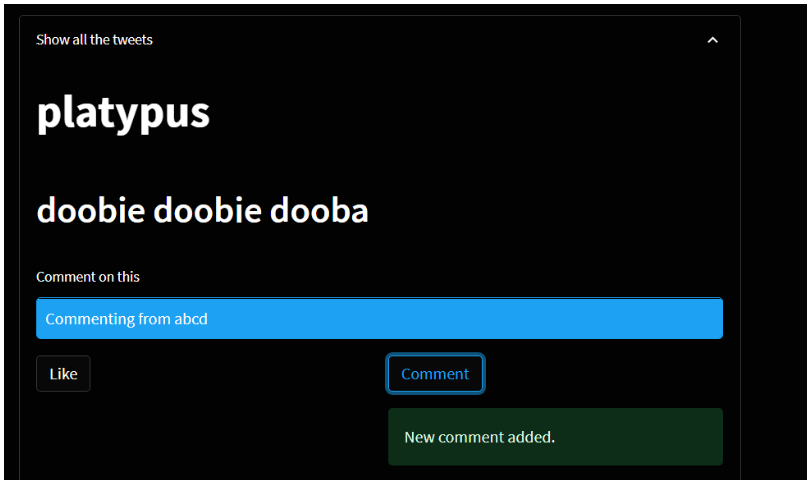
platypus

Enter the username of the person you want to unfollow:

| | 0 |
|---|---|
| 0 | platypus |

*Figure 41 Addition of followers*

### Modification



Show all the tweets

platypus

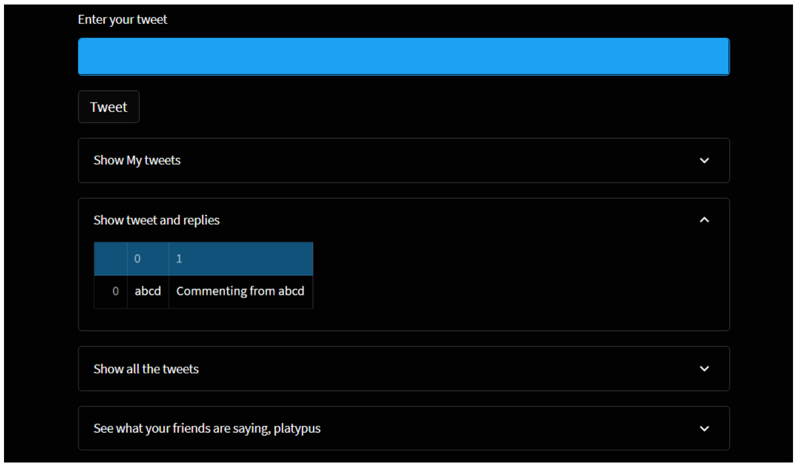
doobie doobie dooba

Comment on this

Commenting from abcd

New comment added.

*Figure 42 Modifications of comment*

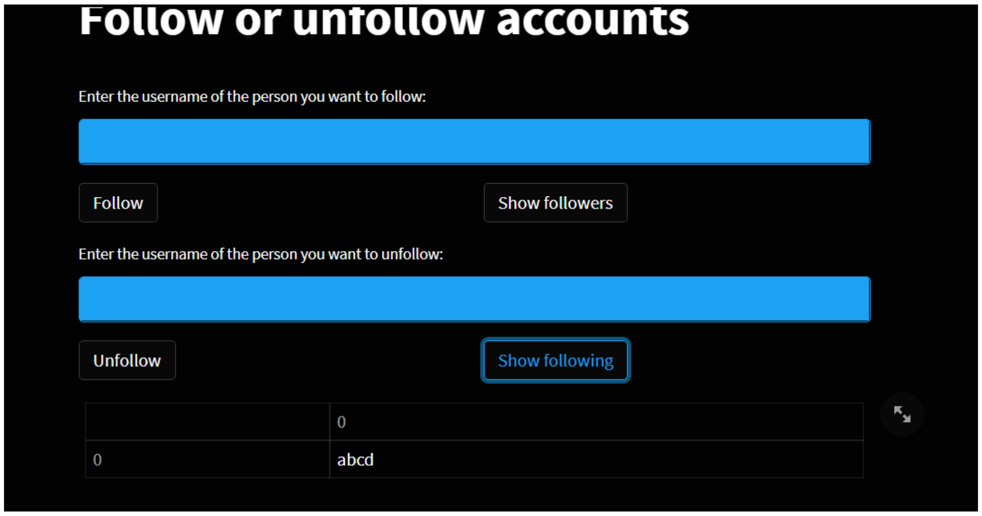


Enter your tweet

| | 0 | 1 |
|---|---|---|
| 0 | abcd | Commenting from abcd |

*Figure 43 Table showing the modification*

## Deletion



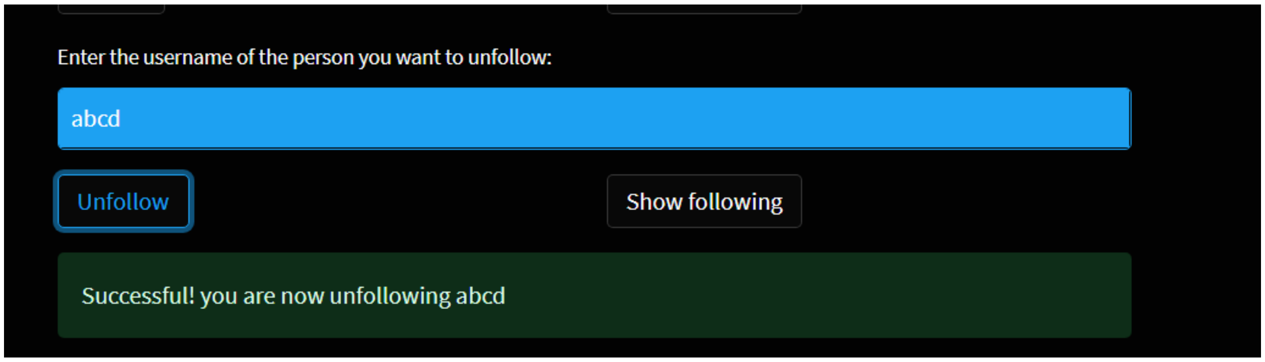
Follow or unfollow accounts

Enter the username of the person you want to follow:

Enter the username of the person you want to unfollow:

| | 0 |
|---|---|
| 0 | abcd |

*Figure 44 Deletion of following user*

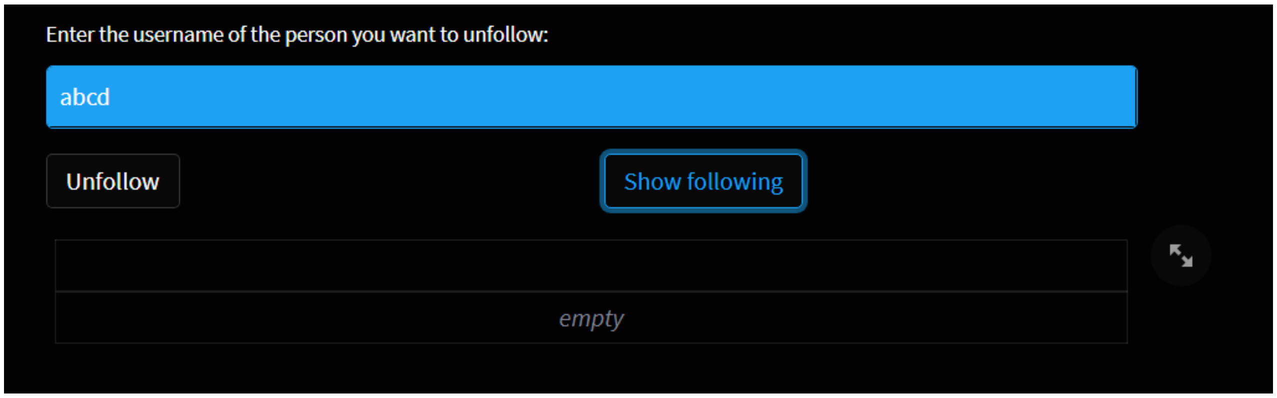


Enter the username of the person you want to unfollow:

abcd

Successful! you are now unfollowing abcd

*Figure 45 Image showing that the procedure has worked successfully*



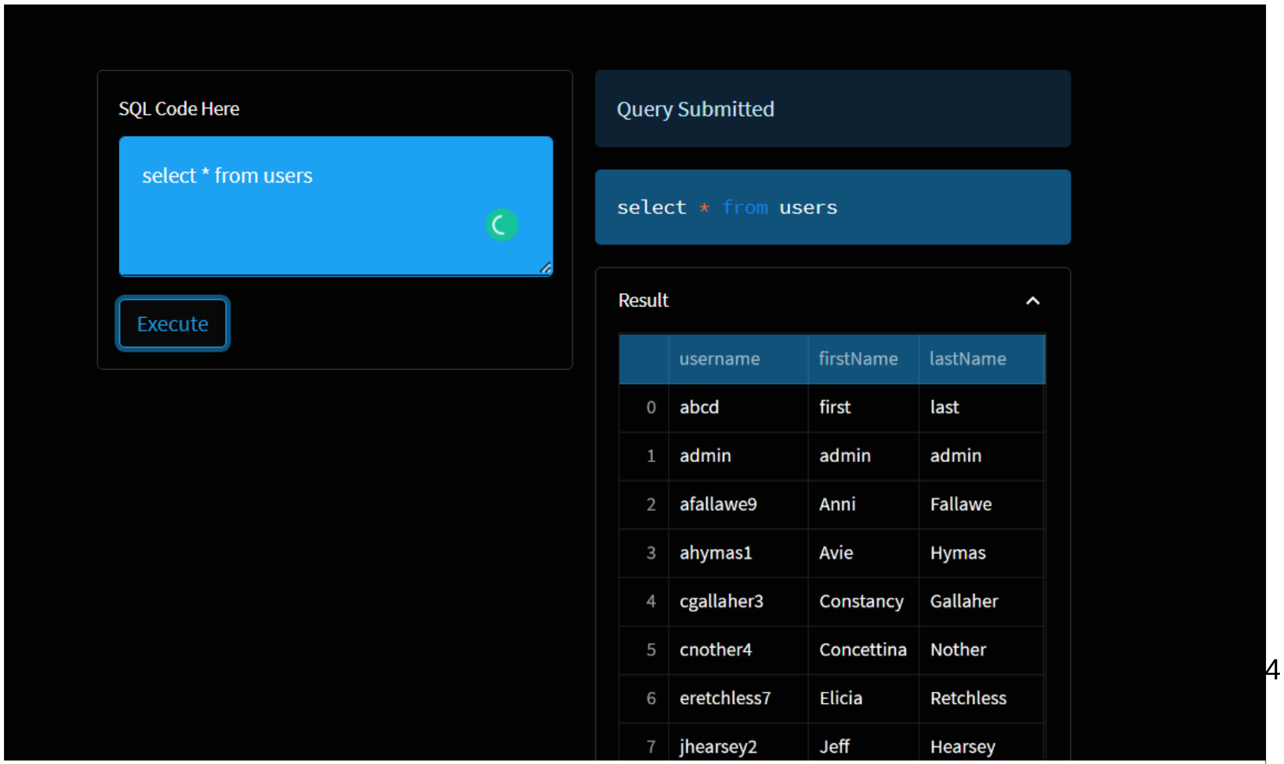
Enter the username of the person you want to unfollow:

abcd

empty

*Figure 46 Table after deletion of record*

## 2. Window to accept SQL commands



SQL Code Here

```
select * from users
```

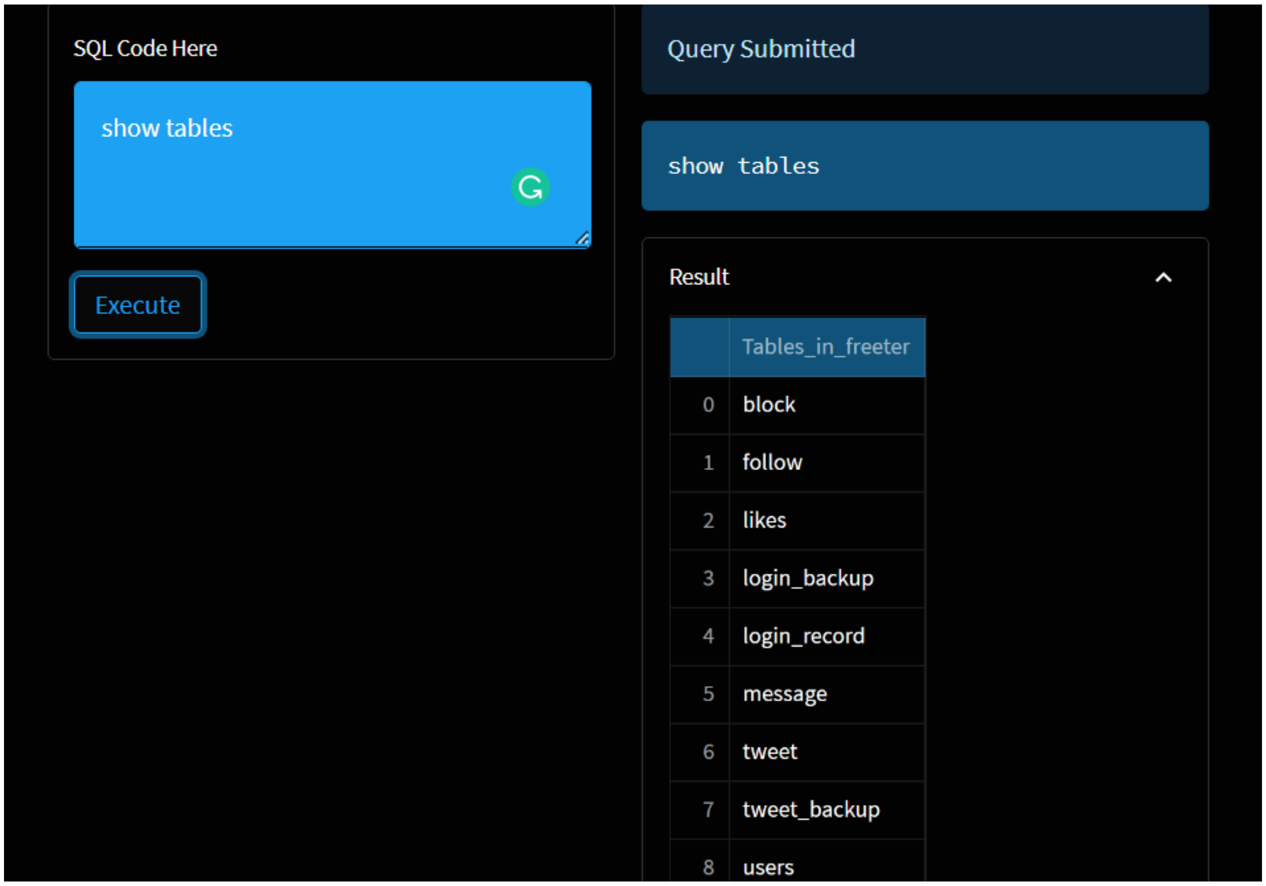
Query Submitted

```
select * from users
```

Result

| | 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 |

*Figure 47 Inbuilt terminal which shows all the users*



```
show tables
```

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

*Figure 45 Showing all the tables*