INSTRUCTIONS

1) We have used xampp server which comes with mysql handling interface(phpmyadmin) so to import the mysql file, run xampp server on a local host. Go to link- localhost/phpmyadmin, create a user with username- root and no password and directly import the Mini-Facebook.sql
2) Or use the command- "mysql -u your_user -p your_database < dump_file.sql" to import the sql script on your default sql server.

QUERIES

SELECT * FROM `User` WHERE `user_id` = '$userID'
This query selects all the users from the database with user_id equal to the input user id.

INSERT INTO `User` (`user_id`,`password`) VALUES ('$userID','$pass')
Query to create a new user for the application. This query INSERTs a new user_id and corresponding login password into the database

INSERT INTO `Profile` (`user_id`,`first_name`,last_name,email_id,dob,gender,image) VALUES ('$userID','$first_name','$last_name','$email','$date','$gender','' . $img .'')
This query is called up whenever a new user signs up for the application and inputs his profile data to generate a new account.

UPDATE `User` SET `online` = 1 WHERE `user_id` = '$userID'
Query to make a user online whenever he logs into his account and make him visible to all his friends for chat.

SELECT * FROM `Request` WHERE `receiver_id` = '$inp_ID' order by `date_time` desc limit 0,6
Query to SELECT the first 6 latest friend request notifications that the user should be notified of.

SELECT DISTINCT `user_id`,`Post`.`date_time`,`seen` FROM `Post_notification`, `Posts`,`Post` WHERE `Post`.`post_id` = `Posts`.`posts_post_id` AND `receiver_id` = '$inp_ID' AND `Posts`.`posts_post_id` = `Post_notification`.`post_id` ORDER BY `date_time` DESC limit 0,6
Query to SELECT the first 6 latest new post notifications that the user should be notified of. These notifications include notification for all the post that recently posted by his friends.

Select Comment`.`sender_id`,`Comment`.`date_time`,`seen` FROM `Comment_notification` , `Comment` WHERE `Comment_notification`.`post_id` = `Comment`.`post_id` AND `Comment_notification`.`comment_id` = `Comment`.`comment_id` AND `Comment`.`sender_id` IN (SELECT `user_id2` AS user FROM `Friends_with` WHERE `user_id1`='$inp_ID' UNION SELECT

`user_id1` AS user FROM `Friends_with` WHERE `user_id2`='$inp_ID') ORDER BY `date_time` DESC limit 0,6
This query extracts all the comment notifications that the user should be notified of. These notifications include the activity of any of the friend of the user on any of the post that he himself posted or one of his friends posted.

SELECT DISTINCT `Event`.`sender_id`,`Event`.`date_time`,`seen` FROM `Event_Notification` , `Event` WHERE `Event_Notification`.`event_id` = `Event`.`event_id` AND `Event_Notification`.`event_id` = `Event`.`event_id` AND `sender_id` IN (SELECT `user_id2` AS user FROM `Friends_with` WHERE `user_id1`='$inp_ID' UNION SELECT `user_id1` AS user FROM `Friends_with` WHERE `user_id2`='$inp_ID') ORDER BY `date_time` DESC limit 0,6"
This query extracts all the event notifications that the user should be notified of. These notifications include the notifications for the events that one of the friends of the user created.

SELECT COUNT(*) AS `count` fROM `Likes` WHERE `user_id` = '$userID' AND `likes_post_id` = '$postID'
Query to count the number of likes a particular user has made on a post. This value can either be 0 or 1.

INSERT INTO `Likes` VALUES('$userID','$postID')
Query to INSERT the like by a user on a post into the database.

UPDATE `Post` SET `likes` = `likes` + 1 WHERE `post_id` = '$postID'
Update the number of likes for the post into the database whenever a user likes a post. The number of likes for that post is incremented by 1.

UPDATE `Post` SET `likes` = `likes` - 1 WHERE `post_id` = '$postID'
Update the number of likes for the post into the database whenever a user dislikes a post. The number of likes for that post is decremented by 1.

DELETE FROM `Likes` WHERE `user_id` = '$userID' AND `likes_post_id` = '$postID'
Query to delete the like by a user on a post provided he had liked it in the past, which is checked in another query.

SELECT * FROM `User` WHERE `user_id` = '$inp_ID' AND `password` = '$inp_pass'
Query to check if a user with given user id and password exists in the database. This query is used at the time of login of existing user.

UPDATE `User` SET `online` = 0 WHERE `user_id` = '$userID'
Query to update user status to offline whenever he logs out of the system.

```
SELECT * from Event,Event_notification where Event.event_id = Event_notification.event_id and
receiver_id = \"".$_SESSION['userId']."\";
```
Query to find out all the events that the user should be able to see. These events are either created by the user or by one of his friends.

```
SELECT user_id1 as id from Friends_with where user_id2 = \"".$_SESSION['userId']."\" UNION SELECT
user_id2 as id from Friends_with where user_id1 = \"".$_SESSION['userId']."\";
```
Query to find out all the friends of a user.

```
SELECT first_name,last_name from Profile,Event where Profile.user_id = Event.sender_id and
sender_id = \"".$_SESSION['event_info']['sender_id']."\";
```
Query to find out the name of the user who originally created the event.

```
DELETE * FROM `EVENT` WHERE `event_id` = '$eventID' AND `sender_id` = '$userID';
```
Query to delete a particular event from the database. Referenced tuples from all the relations will automatically be deleted as per the cascading rule. A user can only delete the events he created.

```
DELETE * FROM `Post` WHERE `post_id` = '$postID' AND '$userID' = (SELECT `user_id` FROM `Posts`
WHERE `posts_post_id` = '$postID');
```
Query to delete a particular post from the database. Referenced tuples from all the relations will automatically be deleted as per the cascading rule. A user can only delete a post he created.

```
DELETE * FROM `Comment` WHERE `post_id` = '$postID' AND `comment_id` = '$commendID'
'$userID' = `sender_id`
```
Query to delete a particular comment from the database. Referenced tuples from all the relations will automatically be deleted as per the cascading rule. A user can only delete a comment he created.

```
"SELECT * from (SELECT post_id,max(date_time) as date_time from (SELECT post_id,date_time FROM
`Post`,`Posts` WHERE Post.post_id not in (SELECT post_id from Comment) and `post_id` =
`posts_post_id` and user_id= \"".$_SESSION['userId']."\"
 UNION
SELECT Post.post_id,date_time FROM `Post`,`Post_notification` where Post.post_id not in
(SELECT post_id from Comment) and `Post`.`post_id` = `Post_notification`.`post_id` AND receiver_id=
\"".$_SESSION['userId']."\"
 UNION
SELECT post_id,date_time from Comment where post_id in (SELECT post_id FROM `Post`,`Posts`
WHERE `post_id` = `posts_post_id` and user_id= \"".$_SESSION['userId']."\"
UNION
SELECT Post.post_id FROM `Post`,`Post_notification` where  `Post`.`post_id` =
`Post_notification`.`post_id` AND receiver_id= \"".$_SESSION['userId']."\" ))t1 group by
t1.post_id)t2 order by t2.date_time desc ;";
```

This query returns all the post_id that should be shown to a user(he should be either the sender or a friend of him). The ones with the most recent changes will be shown first(be it a new post /comment).
This is used to retreive all the post_id to be shown to the user

SELECT first_name,last_name from Profile where user_id = \"".$_SESSION['userId']."\"
This query returns the name of the Session user.
It is required when we give the comment option at the end of each post.

SELECT post_id,date_time,likes,data from Post where post_id = \"".$_SESSION['post']['post_id']."\"
It returns all the data of the particular post_id. It is used in displaying post data.

SELECT first_name,last_name,image from Profile,Posts where Profile.user_id = Posts.user_id and posts_post_id = \"".$_SESSION['post_info']['post_id']."\";
It returns name of post sender.

SELECT * FROM Comment WHERE Comment.post_id = \"".$_SESSION['post_info']['post_id']."\" ORDER BY date_time asc
This is to retrieve comments regarding a particular post.It is required to display comments of a post.

SELECT first_name, last_name,image from Profile,Comment where Profile.user_id = Comment.sender_id and Comment.post_id = \"".$_SESSION['comment_info']['post_id']."\" and Comment.comment_id = \"".$_SESSION['comment_info']['comment_id']."\";";
This gives the name of sender of comment.

SELECT user_id1 as id from Friends_with where user_id2 = \"".$_SESSION['userId']."\" UNION SELECT user_id2 as id from Friends_with where user_id1 = \"".$_SESSION['userId']."\"
This gives the id of all friends of the user(session user). It is used to display the friend list.
It can be used to find out who all are online.

SELECT first_name,last_name from Profile,User where Profile.user_id = User.user_id and User.user_id = \"".$_SESSION['friends']['id']."\" and User.online = 1
It finds out names of all those friends who are online at that time.

SELECT * from Follow where followed_user_id = \"".$_SESSION['userId']."\"
This gives the ids of all those who are following the user.

SELECT first_name,last_name from Profile where user_id = \"".$_SESSION['friends']['followedby_user_id']. "\"
This gives the names of all the followers of session user.

```
INSERT INTO `Post` (`likes`, `data`) VALUES (0,'$postData')
```
This INSERTs a new post into the database.
Note-post_id is not INSERTed as it is set on auto-increment.

```
SELECT max(`post_id`) AS `post_id` FROM `Post` WHERE `likes`=0 AND `data`='$postData'
```
This is used to find out current post_id (as two posts can have the same data with no likes)

```
INSERT INTO `Posts` VALUES ('".$_SESSION['userId']."',$postID)
```
This stores the information about the user who has posted the post.

```
(SELECT `user_id2` AS user FROM `Friends_with` WHERE `user_id1`= '".$_SESSION['userId']."') UNION
(SELECT `user_id1` AS user FROM `Friends_with` WHERE `user_id2`= '".$_SESSION['userId']."')
```
This finds out all friends of the sender(session user in our case) of the post.
It is used while sending them notification.

```
INSERT INTO `Post_notification` VALUES ($postID,'".$row['user']."',0 )
```
All these friends are INSERTed alongwith the post_id in post_notification.
It is helpful while sending them notification.

```
INSERT INTO `Comment` (`post_id`,`comment_id`,`sender_id`,`data`) VALUES
($post[0],$post[1],'$sender_id','$commentData')
```
This INSERTs a new comment for that post.

```
INSERT INTO `Comment_notification` VALUES ('$post[0]','$post[1]','".$row['user']."',0 )
```
INSERT all friends of the comment sender into it, so that the notification goes to them.

```
SELECT * FROM `user` WHERE `user_id` = '$userID'
```
For checking existing users in the 'user' table

```
INSERT INTO `user` (`user_id`,`password`) VALUES ('$userID','$pass')
```
For INSERTing an entry in the 'user' table

```
INSERT INTO `Profile` (`user_id`,`first_name`,last_name,email_id,dob,gender,image) VALUES
('$userID','$first_name','$last_name','$email','$date','$gender','$img')
```
For INSERTing an entry in the 'profile' table using the details provided in the login form.

```
UPDATE Profile SET age='$age',dob='$dob',relationship_status='$relationStat',gender='$gender'
WHERE user_id='$_SESSION['userId']'
```

Adding basic information to 'profile'

UPDATE Profile SET graduation_school='$grad_school', high_school='$high_school', primary_school='$prim_school' WHERE user_id='$_SESSION['userId']'
Adding Education details to 'profile'

UPDATE Profile SET house_no='$house_no', street='$street',pin='$pin_code',city='$city', state='$state',phone_no='$phone_no',email_id='$email_id',quote='$quote' WHERE user_id='$_SESSION['userId']'
Adding Contact details to 'profile'

INSERT INTO `request` (`receiver_id`,`sender_id`,seen) VALUES ('$friend_add','" . $_SESSION['userId'] . "',0);
Adding friend request to the 'request' table

DELETE FROM `friends_with` WHERE `user_id1`='$friend_add' AND `user_id2`= '" . $_SESSION['userId'] . "'
DELETE FROM `friends_with` WHERE `user_id2`='$friend_add' AND `user_id1`= '" . $_SESSION['userId'] . "'
Used for unfriending. Deletes a---friends_with---b and b---friends_with---a (both entries will not be present, iust a precaution taken here.)

DELETE FROM `request` WHERE `receiver_id` = '" . $_SESSION['userId'] . "' AND `sender_id`='$friend_add' INSERT INTO `friends_with` (`user_id1`,`user_id2`) VALUES ('$friend_add','" . $_SESSION['userId'] . "')
Used for accepting friend request, delete the request from the 'request' table, and add an entry in the 'friends_with' table.

DELETE FROM `request` WHERE `sender_id` = '" . $_SESSION['userId'] . "' AND `receiver_id`='$friend_add'
Used for deleting a friend request.

INSERT INTO `follow` (`followed_user_id`,`followedby_user_id`) VALUES ('$Follower_add','" . $_SESSION['userId'] . "')
Add an entry in the 'follow' table so the the user currently logged-in follows the person.

DELETE FROM `follow` WHERE `followed_user_id`='$Follower_add' AND `followedby_user_id`= '" . $_SESSION['userId'] . "'
Delete an entry from 'follow' table so that the user currently logged in ceases to follow the person.

```
SELECT * FROM `Profile` WHERE `user_id` = \"".$_SESSION['userId']."\"
```
Get details of the logged in user.

```
SELECT * FROM `Profile` WHERE `user_id` = \"".$user_id_page."\"
```
Get details of the profile corresponding to the person requested by the current logged in user.

```
SELECT user_id1 as id from friends_with where user_id2 = \"".$_SESSION['userId']."\" and user_id1 =
\"".$user_id_page."\"  UNION SELECT user_id2 as id from friends_with where user_id1 =
\"".$_SESSION['userId']."\" and user_id2 = \"".$user_id_page."\";
```
To check if the person whose profile is currently being viewed is a friend.

```
SELECT `sender_id` from `request` where sender_id='" . $_SESSION['userId'] . "' and
receiver_id='$user_id_page';
```
To check if the person whose profile is currently being viewed has been already sent a friend request

```
SELECT `sender_id` from `request` where sender_id='$user_id_page' and receiver_id='" .
$_SESSION['userId'] . "'
```
To check if the person whose profile is currently being viewed has sent a friend request to the user who is currently logged in.

```
SELECT followed_user_id as id from follow where followedby_user_id = \"".$_SESSION['userId']."\"
and followed_user_id = \"".$user_id_page."\"
```
To check if the person whose profile is currently being viewed is being followed by the user who is currently logged in.

```
INSERT INTO event (sender_id, event_name, event_date_time, description, house_no, street, pin,
city, state, country)  VALUES ('$_SESSION['userId']', '$event_name', '$event_time', '$description',
'$house_no','$street','$pin_code','$city','$state', '$country')
```
To create an event.

```
SELECT distinct id from ((SELECT date_time,receiver_id as id,data from `Message`   where sender_id =
\"".$_SESSION['userId']."\") UNION  (SELECT date_time,sender_id as id,data from `Message`  where
receiver_id = \"".$_SESSION['userId']."\")  order by date_time DESC ) AS `table`
```
SELECT id's of users the logged in person has chatted with in descending order of time

```
SELECT `user_id1` AS `id` from `Friends_with` where `user_id2` = \"".$_SESSION['userId']."\" UNION
SELECT `user_id2` AS `id`from `Friends_with` where `user_id1` = \"".$_SESSION['userId']."\"
```
SELECT id's of users who are friends with the logged in person

```
SELECT `first_name`,`last_name` from `Profile` where user_id = \"".$row['id']."\"
```

Get first & last name corresponding to a userid

```
SELECT * from `Message` where `sender_id` = \"".$_SESSION['userId']."\" AND `receiver_id` =
\"".$other_id."\" OR  `sender_id` = \"".$other_id."\" AND `receiver_id` = \"".$_SESSION['userId']."\"
order by date_time ASC
```

Get all messages between two given users in ascendig order of time

```
INSERT into `Message` (sender_id,receiver_id,date_time,data) values ('$sender', '$receiver',
'$mysql_date_now', '$message')
```

INSERT new message into database

```
SELECT MAX(`event_id`) AS `event_id` FROM `Event` WHERE `description`='$description'
```

Select the maximum ID of the event created in the event table.

```
INSERT INTO `Event_Notification` VALUES ($eventID,'".$row['user']."',0 )
```

Insert notifications in the event_notification table of the users that are friends with the created of the event.

Triggers-

```
CREATE OR REPLACE TRIGGER SELFFRIEND (
BEFORE INSERT ON friends_with
REFERENCING NEW AS new OLD AS Old
FOR EACH ROW
DECLARE
BEGIN
if (new.user_id1 = new.user_id2) then
RAISE trial_error;
);
```

```
CREATE OR REPLACE TRIGGER WRONGEVENT (
BEFORE INSERT ON event
REFERENCING NEW AS new OLD AS Old
FOR EACH ROW
DECLARE
BEGIN
if (new.date_time < CURRENT_TIMESTAMP) then
RAISE trial_error;
);
```