Homework 2 Report

Q1.
Query: SELECT USER_ID, F_NAME, L_NAME FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE INVITER_ID=5 AND STATUS ='Y');

Explanation: First we'll select all the friends list from FREINDSHIP relation of INVITER_ID 5 and then we can list down all the details of those friends from USERS table. In following picture INVITER_ID is 101 instead of 5.

```
MySQL [hw2]> SELECT USER_ID, F_NAME, L_NAME FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE INVITER_ID=101 AND STATUS ='Y');
+---------+--------+--------+
| USER_ID | F_NAME | L_NAME |
+---------+--------+--------+
|     106 | Pitbul | Osuna  |
|     105 | Bad    | Bunny  |
+---------+--------+--------+
2 rows in set (0.03 sec)
```

Q2.
Query: SELECT USER_ID, F_NAME, L_NAME, DOB, GENDER FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE INVITER_ID=1 AND STATUS ='N');

Explanation: We can pullup the records from FREINDSHIP table with status as 'N' or not accepted. The inner query will return all pending request to user id 1.i.e user ids who sent request but user 1 has not accepted them. The outer query will display the name, gender DOB of those invitees. In following picture INVITER_ID is 101 instead of 1.

```
MySQL [hw2]> SELECT USER_ID, F_NAME, L_NAME, DOB, GENDER FROM USERS WHERE USER_ID IN (select INVITER_ID from FREINDSHIP WHERE INVITEE_ID=101 AND STATUS ='N');
+---------+--------+---------+------------+--------+
| USER_ID | F_NAME | L_NAME  | DOB        | GENDER |
+---------+--------+---------+------------+--------+
|     104 | Bekey  | Shakira | 1994-05-03 | F      |
+---------+--------+---------+------------+--------+
1 row in set (0.05 sec)
```

Q3.
Query: SELECT USER_ID, F_NAME, L_NAME, DOB, GENDER FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE INVITER_ID = 1 AND STATUS ='Y') HAVING USERS.GENDER='F'
INTERSECT
SELECT USER_ID, F_NAME, L_NAME, DOB, GENDER FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE INVITER_ID= 2 AND STATUS ='Y') HAVING USERS.GENDER='F';

Explanation: In order to retrieve the common or mutual friends of user 1 and 2, we'll use INTERSECT set Operation on the results of user 1's friends and user 2's friends. The intersection of subquery one and two will result in desired output.

Since INTERSECT doesn't work in MySQL, we can replace the above-mentioned query in following format:
SELECT DISTINCT * FROM
(SELECT f1, f2, f3... FROM table1)
INNER JOIN
(SELECT f1, f2, f3... FROM table2)
USING (primary key)

Q4.

Query: SELECT USER_ID, FROM USERS WHERE USER_ID IN (SELECT INVITER_ID, COUNT(INVITEE_ID) AS FREINDS_COUNT FROM FREINDSHIP WHERE INVITER_ID IN
(SELECT USER_ID FROM (select * FROM USERS WHERE USER_ID IN (select C.USER_ID from COMMENT C JOIN POST P on P.POST_ID=C.POST_ID where P.USER_ID=10) HAVING GENDER='F' AND DOB>'1983-05-03') AS FrndCommnted)
AND STATUS ='Y' GROUP BY INVITER_ID);

Explanation: First select users who commented on user id 10's post by following query:
select * FROM USERS WHERE USER_ID IN (select C.USER_ID from COMMENT C JOIN POST P on P.POST_ID=C.POST_ID where P.USER_ID=10

Now, Filter the results of above mentioned query with Gender as 'F' and DOB > '1990-12-20'. Furthermore, we have to calculate the friends count of those users which is done by following query:

SELECT USER_ID, FROM USERS WHERE USER_ID IN (SELECT INVITER_ID, COUNT(INVITEE_ID) AS FREINDS_COUNT FROM FREINDSHIP WHERE INVITER_ID IN (*output from above queries*) AND STATUS ='Y' GROUP BY INVITER_ID);

```
MySQL [hw2]>
MySQL [hw2]> SELECT INVITER_ID as USER_ID,  COUNT(INVITEE_ID) AS FREINDS_COUNT FROM FREINDSHIP WHERE INVITER_ID IN (SELECT USER_ID FROM (select USER_ID, GENDER, DOB FROM USERS WHERE
 USER_ID IN(select C.USER_ID from COMMENT C JOIN POST P on P.POST_ID=C.POST_ID where P.USER_ID=100) HAVING GENDER='F' AND DOB>'1983-05-03') AS CUSTOMERIDS ) AND STATUS ='Y'  GROUP BY
 INVITER_ID ;
+---------+--------------+
| USER_ID | FREINDS_COUNT |
+---------+--------------+
|     104 |            3 |
+---------+--------------+
1 row in set (0.03 sec)
```

Q5.

Query: SELECT * from ((select USERS.USER_ID from USERS where USER_ID NOT IN (select U.USER_ID from USERS U JOIN FREINDSHIP F on F.INVITER_ID=U.USER_ID and F.STATUS='Y')) as A
CROSS JOIN
(select USERS.USER_ID from USERS where USER_ID NOT IN (select U.USER_ID from USERS U JOIN FREINDSHIP F on F.INVITER_ID=U.USER_ID and F.STATUS='Y')) As B) where A.USER_ID NOT IN
(SELECT USER_ID FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE STATUS ='Y') **INTERSECT** (SELECT USER_ID FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE STATUS ='Y')));

Explanation: First select the users who are friend of each other's by CROSS JOIN and then by using INTERSECT select users who are not friends but have some mutual friends.
Then we search for pairs which are **not** in result set of above two operations.

```
MySQL [hw2]> SELECT * from ((select USERS.USER_ID from USERS where USER_ID NOT IN(select U.USER_ID from USERS U JOIN FREINDSHIP F on F.INVITER_ID=U.USER_ID and F.STATUS='Y') ) as A
    -> CROSS JOIN
    -> (select USERS.USER_ID from USERS where USER_ID NOT IN(select U.USER_ID from USERS U JOIN FREINDSHIP F on F.INVITER_ID=U.USER_ID and F.STATUS='Y')) As B )  and A.USER_ID NOT IN
  (SELECT USER_ID FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE STATUS ='Y') and  USER_ID IN(SELECT USER_ID FROM USERS WHERE USER_ID IN (select INVITEE_ID from
FREINDSHIP WHERE STATUS ='Y')));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'and A.USER_ID NOT IN (SELEC
T USER_ID FROM USERS WHERE USER_ID IN (select INVITEE' at line 3
MySQL [hw2]> SELECT * from ((select USERS.USER_ID from USERS where USER_ID NOT IN(select U.USER_ID from USERS U JOIN FREINDSHIP F on F.INVITER_ID=U.USER_ID and F.STATUS='Y') ) as A
    -> CROSS JOIN
    -> (select USERS.USER_ID from USERS where USER_ID NOT IN(select U.USER_ID from USERS U JOIN FREINDSHIP F on F.INVITER_ID=U.USER_ID and F.STATUS='Y')) As B ) where  A.USER_ID NOT
IN (SELECT USER_ID FROM USERS WHERE USER_ID IN (select INVITEE_ID from FREINDSHIP WHERE STATUS ='Y') and  USER_ID IN(SELECT USER_ID FROM USERS WHERE USER_ID IN (select INVITEE_ID fro
m FREINDSHIP WHERE STATUS ='Y')));
+---------+---------+
| USER_ID | USER_ID |
+---------+---------+
|     107 |     102 |
|     107 |     103 |
|     107 |     105 |
|     107 |     107 |
+---------+---------+
4 rows in set (0.04 sec)
```

Q6.
Query: SELECT A.FemaleUser, B.MaleUser FROM
((SELECT * FROM
(SELECT U.USER_ID as FemaleUser, COUNT(C.USER_ID) AS CNT FROM USERS U JOIN COMMENT C ON U.USER_ID=C.USER_ID AND U.GENDER='F' GROUP BY C.USER_ID) AS CNTA WHERE CNTA.CNT >=5) A
CROSS JOIN
(SELECT * FROM (SELECT U.USER_ID as MaleUser, COUNT(C.USER_ID) AS CNT FROM USERS U JOIN COMMENT C ON U.USER_ID=C.USER_ID AND U.GENDER='M' GROUP BY C.USER_ID) AS CNTB WHERE CNTB.CNT >=5) B);

Explanation: First, let's select **Female** users who commented 5 or more than five (at least 5) on a user's post. The following query will result the **female** users who commented at least 5 times on someone else's post:
SELECT U.USER_ID as FemaleUser, COUNT(C.USER_ID) AS CNT FROM USERS U JOIN COMMENT C ON U.USER_ID=C.USER_ID AND U.GENDER='F' GROUP BY C.USER_ID) AS CNTA WHERE CNTA.CNT >=5) A

Similarly we can calculate **male** users who commented at least 5 times on someone else's post with changing the U.GENDER='M' and then we can take cross join to make the pair of male and female users.

```
MySQL [hw2]>
MySQL [hw2]> SELECT A.FemaleUser, B.MaleUser FROM
    -> ((SELECT * FROM
    -> (SELECT U.USER_ID as FemaleUser, COUNT(C.USER_ID) AS CNT FROM USERS U JOIN COMMENT C ON U.USER_ID=C.USER_ID AND U.GENDER='F' GROUP BY C.USER_ID) AS CNTA WHERE CNTA.CNT >0) A
    ->
    -> CROSS JOIN
    -> (SELECT * FROM (SELECT U.USER_ID as MaleUser, COUNT(C.USER_ID) AS CNT FROM USERS U JOIN COMMENT C ON U.USER_ID=C.USER_ID AND U.GENDER='M' GROUP BY C.USER_ID) AS CNTB WHERE
    -> CNTB.CNT >0) B) ;
+------------+----------+
| FemaleUser | MaleUser |
+------------+----------+
|        102 |      100 |
|        104 |      100 |
|        102 |      101 |
|        104 |      101 |
+------------+----------+
4 rows in set (0.04 sec)
```