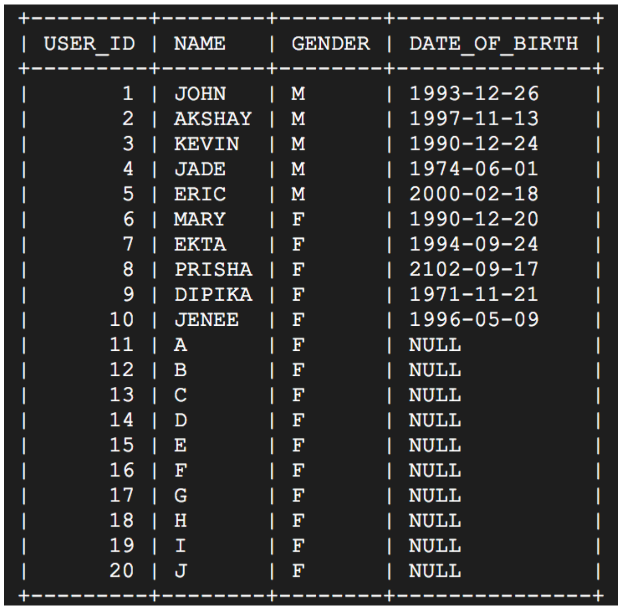
# Home Work-2

**Note: Nicely formatted query is in Ekta_Rita_HW2.docx file**

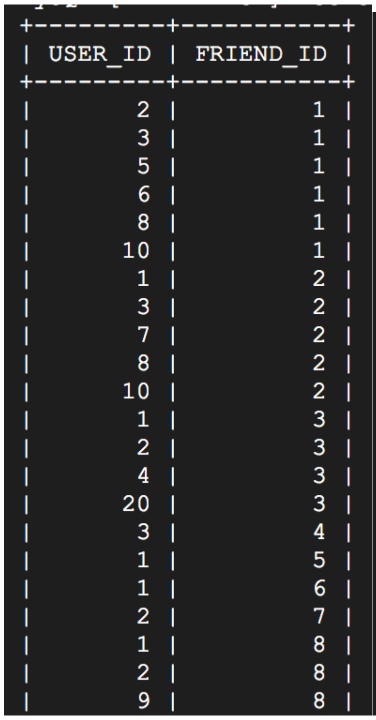
The tables used for the queries are as follows:

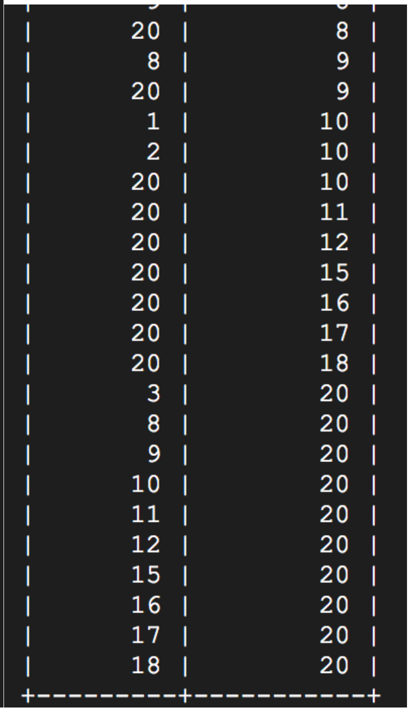
USERS TABLE



```
+---------+--------+--------+---------------+
| USER_ID | NAME   | GENDER | DATE_OF_BIRTH |
+---------+--------+--------+---------------+
|       1 | JOHN   | M      | 1993-12-26    |
|       2 | AKSHAY | M      | 1997-11-13    |
|       3 | KEVIN  | M      | 1990-12-24    |
|       4 | JADE   | M      | 1974-06-01    |
|       5 | ERIC   | M      | 2000-02-18    |
|       6 | MARY   | F      | 1990-12-20    |
|       7 | EKTA   | F      | 1994-09-24    |
|       8 | PRISHA | F      | 2102-09-17    |
|       9 | DIPIKA | F      | 1971-11-21    |
|      10 | JENEE  | F      | 1996-05-09    |
|      11 | A      | F      | NULL          |
|      12 | B      | F      | NULL          |
|      13 | C      | F      | NULL          |
|      14 | D      | F      | NULL          |
|      15 | E      | F      | NULL          |
|      16 | F      | F      | NULL          |
|      17 | G      | F      | NULL          |
|      18 | H      | F      | NULL          |
|      19 | I      | F      | NULL          |
|      20 | J      | F      | NULL          |
+---------+--------+--------+---------------+
```

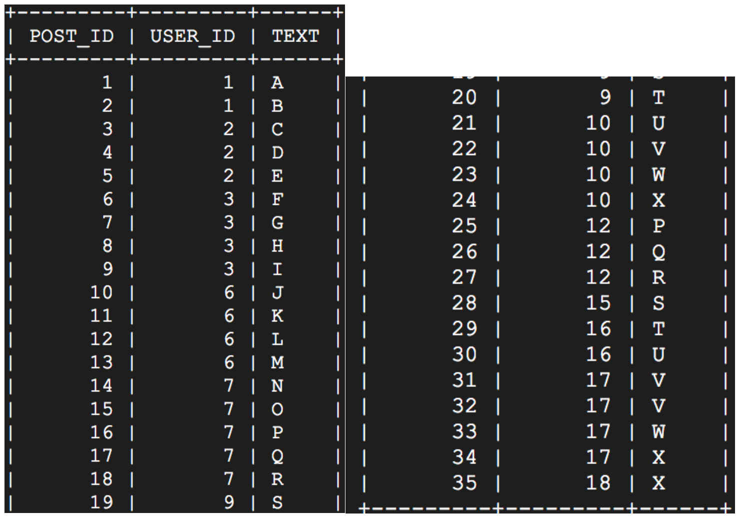
FRIENDSHIPS TABLE





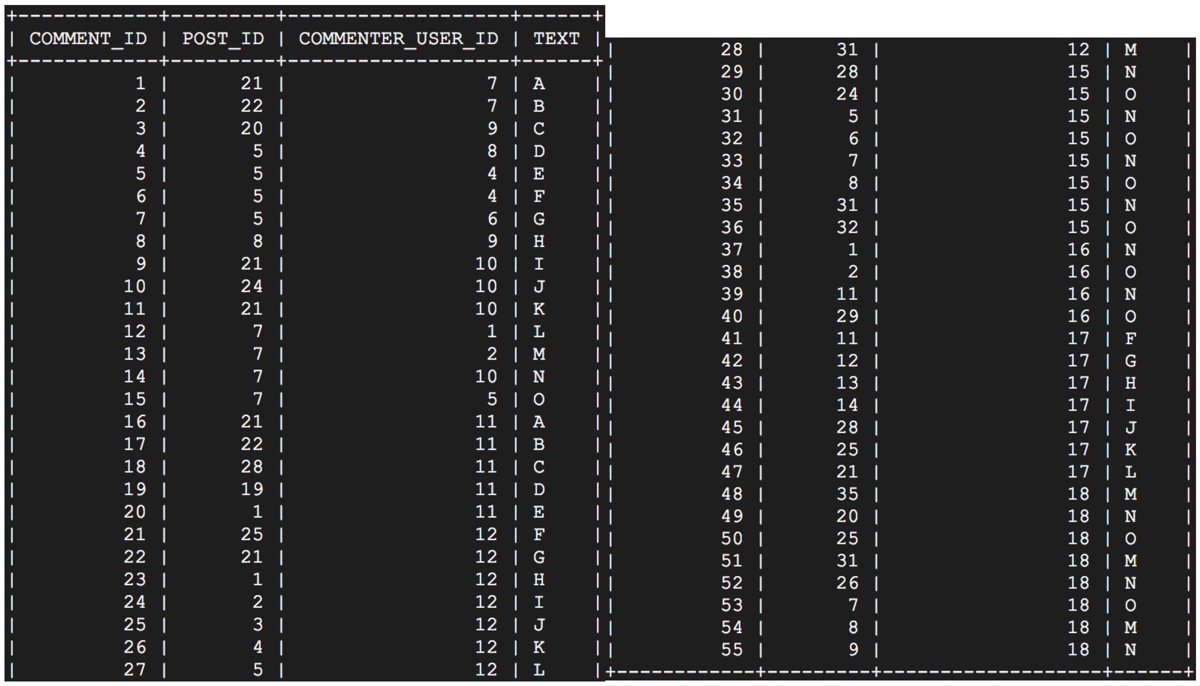
```
+---------+-----------+
| USER_ID | FRIEND_ID |
+---------+-----------+
|       2 |         1 |
|       3 |         1 |
|       5 |         1 |
|       6 |         1 |
|       8 |         1 |
|      10 |         1 |
|       1 |         2 |
|       3 |         2 |
|       7 |         2 |
|       8 |         2 |
|      10 |         2 |
|       1 |         3 |
|       2 |         3 |
|       4 |         3 |
|      20 |         3 |
|       3 |         4 |
|       1 |         5 |
|       1 |         6 |
|       2 |         7 |
|       1 |         8 |
|       2 |         8 |
|       9 |         8 |
|      20 |         8 |
|       8 |         9 |
|      20 |         9 |
|       1 |        10 |
|       2 |        10 |
|      20 |        10 |
|      20 |        11 |
|      20 |        12 |
|      20 |        15 |
|      20 |        16 |
|      20 |        17 |
|      20 |        18 |
|       3 |        20 |
|       8 |        20 |
|       9 |        20 |
|      10 |        20 |
|      11 |        20 |
|      12 |        20 |
|      15 |        20 |
|      16 |        20 |
|      17 |        20 |
|      18 |        20 |
+---------+-----------+
```

POSTS TABLE



| POST_ID | USER_ID | TEXT |
|---|---|---|
| 1 | 1 | A |
| 2 | 1 | B |
| 3 | 2 | C |
| 4 | 2 | D |
| 5 | 2 | E |
| 6 | 3 | F |
| 7 | 3 | G |
| 8 | 3 | H |
| 9 | 3 | I |
| 10 | 6 | J |
| 11 | 6 | K |
| 12 | 6 | L |
| 13 | 6 | M |
| 14 | 7 | N |
| 15 | 7 | O |
| 16 | 7 | P |
| 17 | 7 | Q |
| 18 | 7 | R |
| 19 | 9 | S |
| 20 | 9 | T |
| 21 | 10 | U |
| 22 | 10 | V |
| 23 | 10 | W |
| 24 | 10 | X |
| 25 | 12 | P |
| 26 | 12 | Q |
| 27 | 12 | R |
| 28 | 15 | S |
| 29 | 16 | T |
| 30 | 16 | U |
| 31 | 17 | V |
| 32 | 17 | V |
| 33 | 17 | W |
| 34 | 17 | X |
| 35 | 18 | X |

COMMENTS TABLE

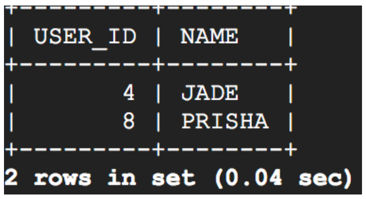


| COMMENT_ID | POST_ID | COMMENTER_USER_ID | TEXT |
|---|---|---|---|
| 1 | 21 | 7 | A |
| 2 | 22 | 7 | B |
| 3 | 20 | 9 | C |
| 4 | 5 | 8 | D |
| 5 | 5 | 4 | E |
| 6 | 5 | 4 | F |
| 7 | 5 | 6 | G |
| 8 | 8 | 9 | H |
| 9 | 21 | 10 | I |
| 10 | 24 | 10 | J |
| 11 | 21 | 10 | K |
| 12 | 7 | 1 | L |
| 13 | 7 | 2 | M |
| 14 | 7 | 10 | N |
| 15 | 7 | 5 | O |
| 16 | 21 | 11 | A |
| 17 | 22 | 11 | B |
| 18 | 28 | 11 | C |
| 19 | 19 | 11 | D |
| 20 | 1 | 11 | E |
| 21 | 25 | 12 | F |
| 22 | 21 | 12 | G |
| 23 | 1 | 12 | H |
| 24 | 2 | 12 | I |
| 25 | 3 | 12 | J |
| 26 | 4 | 12 | K |
| 27 | 5 | 12 | L |
| 28 | 31 | 12 | M |
| 29 | 28 | 15 | N |
| 30 | 24 | 15 | O |
| 31 | 5 | 15 | N |
| 32 | 6 | 15 | O |
| 33 | 7 | 15 | N |
| 34 | 8 | 15 | O |
| 35 | 31 | 15 | N |
| 36 | 32 | 15 | O |
| 37 | 1 | 16 | N |
| 38 | 2 | 16 | O |
| 39 | 11 | 16 | N |
| 40 | 29 | 16 | O |
| 41 | 11 | 17 | F |
| 42 | 12 | 17 | G |
| 43 | 13 | 17 | H |
| 44 | 14 | 17 | I |
| 45 | 28 | 17 | J |
| 46 | 25 | 17 | K |
| 47 | 21 | 17 | L |
| 48 | 35 | 18 | M |
| 49 | 20 | 18 | N |
| 50 | 25 | 18 | O |
| 51 | 31 | 18 | M |
| 52 | 26 | 18 | N |
| 53 | 7 | 18 | O |
| 54 | 8 | 18 | M |
| 55 | 9 | 18 | N |

**1. List the ids and names of users who have no posts and have one or more comments on POST_ID=5.**

```
SELECT DISTINCT U.USER_ID, U.NAME
FROM COMMENTS C LEFT JOIN USERS U
ON C.COMMENTER_USER_ID = U.USER_ID
WHERE C.POST_ID = 5
AND U.USER_ID NOT IN
(
    SELECT P.USER_ID FROM POSTS P
)
ORDER BY U.USER_ID;
```

**Explanation:**
The query selects the USER_ID and NAME of those users who have commented on POST_ID = 5 and which are not in the result returned by the inner query which finds the users who have at least 1 post. There is a left join in table COMMENTS and USERS so that it maps the COMMENTER_USER_ID to USER_ID. The result obtained is as follows:

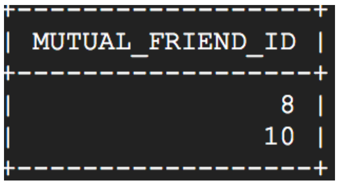


```
+---------+--------+
| USER_ID | NAME   |
+---------+--------+
|       4 | JADE   |
|       8 | PRISHA |
+---------+--------+
2 rows in set (0.04 sec)
```

**2. List the USER_ID of female mutual friends between users 1 and 2.**

```
SELECT F1.FRIEND_ID MUTUAL_FRIEND_ID
FROM FRIENDSHIPS F1, USERS U1
WHERE F1.USER_ID = 1
AND F1.FRIEND_ID = U1.USER_ID
AND U1.GENDER = 'F'
AND F1.FRIEND_ID IN
(
    SELECT F2.FRIEND_ID
    FROM FRIENDSHIPS F2, USERS U2
    WHERE F2.USER_ID = 2
    AND F2.FRIEND_ID = U2.USER_ID
    AND U2.GENDER = 'F'
);
```

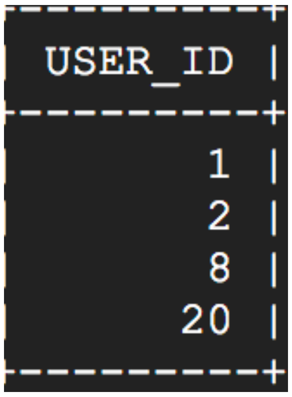
**Explanation:**
Finding all the female friends of USER_ID = 1 and all the female friends of USER_ID = 2. After this only those female friends of USER_ID = 1 is selected who are also there in the female friend list of USER_ID = 2. The result obtained is as follows:


```
+------------------+
| MUTUAL_FRIEND_ID |
+------------------+
|                8 |
|               10 |
+------------------+
```

**3. List the USER_ID of users who have more than 2 friends whom have at least one post.**

```
SELECT F.USER_ID
FROM FRIENDSHIPS F RIGHT JOIN
(
    SELECT DISTINCT P1.USER_ID FROM POSTS P1
) P2
ON F.FRIEND_ID = P2.USER_ID
GROUP BY F.USER_ID
HAVING COUNT(F.FRIEND_ID) > 2
ORDER BY F.USER_ID;
```

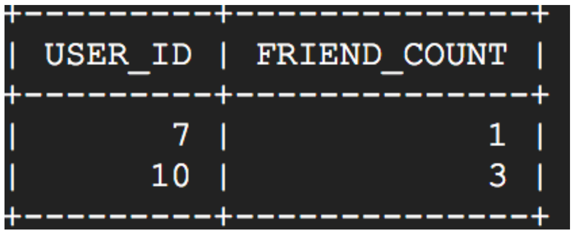
**Explanation:**
Getting only those distinct friends who have posted at least 1 post from the POSTS table and counting these friends of each user by using a right join which maps the FRIEND_ID to USER_ID. Now selecting only those users whose friend count from above condition is greater than 2 by using HAVING clause. Assuming more than 2 means greater than 2 and not greater than equal to 2. The result obtained is as follows:


```
---------+
 USER_ID |
---------+
       1 |
       2 |
       8 |
      20 |
---------+
```

**4. List unique USER_ID of female users who were born after ‘1990-12-20’ and commented on posts of USER_ID=10. Show their friends count in a separate column.**

```
SELECT USR.USER_ID, COUNT(F.FRIEND_ID) FRIEND_COUNT
FROM
(
    SELECT DISTINCT U.USER_ID FROM USERS U, COMMENTS C
    WHERE U.GENDER = 'F' AND U.DATE_OF_BIRTH > '1990-12-20'
    AND C.COMMENTER_USER_ID = U.USER_ID
    AND C.POST_ID IN
    (
        SELECT P.POST_ID
        FROM POSTS P
        WHERE P.USER_ID = 10
    )
) USR LEFT JOIN FRIENDSHIPS F
ON USR.USER_ID = F.USER_ID
GROUP BY USR.USER_ID;
```

**Explanation:**
Select the USER_ID of female users who satisfy the given condition and then left join it with the FRIENDSHIPS table to get the count of the friends of each user. The inner query selects distinct users because there can be duplicate users who have commented on multiple posts.
Assuming born after ‘1990-12-20’ does not include that date. The result obtained is as follows:

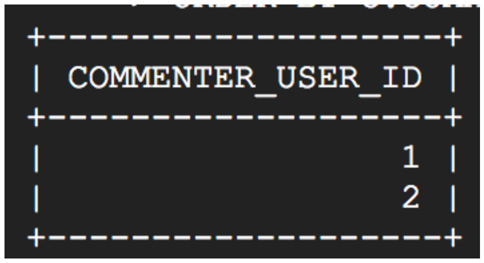


```
+---------+--------------+
| USER_ID | FRIEND_COUNT |
+---------+--------------+
|       7 |            1 |
|      10 |            3 |
+---------+--------------+
```

**5. List the USER_ID of users who commented on POST_ID=7 and are friends with the post creator.**

```
SELECT DISTINCT C.COMMENTER_USER_ID
FROM
(
    SELECT F.FRIEND_ID
    FROM FRIENDSHIPS F
    WHERE F.USER_ID =
    (
        SELECT P.USER_ID
        FROM POSTS P
        WHERE P.POST_ID = 7
    )
) PF
LEFT JOIN
COMMENTS C
ON C.COMMENTER_USER_ID = PF.FRIEND_ID
WHERE C.POST_ID = 7
ORDER BY C.COMMENTER_USER_ID;
```

**Explanation:**
Get the list of friends of the user who has posted with POST_ID = 7 and then join these friends with the COMMENTS table on their FRIEND_ID and COMMENTER_USER_ID and the post on which they comment is POST_ID = 7.
I have used DISTINCT C.COMMENTER_USER_ID because there is a possibility that the user must have commented more than once on POST_ID = 7. The result obtained is as follows:



```
+-------------------+
| COMMENTER_USER_ID |
+-------------------+
|                 1 |
|                 2 |
+-------------------+
```

**6. List the USER_ID and NAME of the 3 most female commenters, who are friends with USER_ID=20, with at least 3 comments on all the posts combined, excluding the comments under ones posted by USER_ID=10 and themselves. Show their augmented count of comments in a separate column. Also, show their total number of comments in another separate column.**

```
SELECT F.USER_ID, F.NAME, AUG.COUNT AUGMENTED_COMMENTS_COUNT,
F.NON_AUGMENTED_TOTAL TOTAL_COUNT
FROM
(    SELECT FU.USER_ID, FU.NAME,
     COUNT(COMMENTS.COMMENT_ID) NON_AUGMENTED_TOTAL
     FROM COMMENTS
     RIGHT JOIN
    (    SELECT U.USER_ID, U.NAME
         FROM USERS U, FRIENDSHIPS F
         WHERE U.USER_ID = F.FRIEND_ID
         AND F.USER_ID = 20
         AND U.GENDER = 'F'
    ) FU
    ON COMMENTS.COMMENTER_USER_ID = FU.USER_ID
    GROUP BY FU.USER_ID, FU.NAME
) F
RIGHT JOIN
(    SELECT A.USER_ID1 USER_ID, A.COUNT1 COUNT
     FROM
     (    SELECT AUG2.USER_ID1, AUG2.COUNT1, AUG2.USER_ID2, AUG2.COUNT2,
          AUG2.USER_ID3, COUNT(C.COMMENT_ID) COUNT3,
          AUG2.COUNT1 + AUG2.COUNT2 + COUNT(C.COMMENT_ID) TOTAL
          FROM
          (    SELECT AUG1.USER_ID1, AUG1.COUNT1, AUG1.USER_ID2,
               COUNT(C.COMMENT_ID) COUNT2, AUG1.USER_ID3
               FROM
               (    SELECT UC.USER_ID1, COUNT(C.COMMENT_ID) COUNT1,
                    UC.USER_ID2, UC.USER_ID3
                    FROM
                    (    SELECT FU1.USER_ID USER_ID1, FU2.USER_ID USER_ID2,
                         FU3.USER_ID USER_ID3
                         FROM
                         (    SELECT U.USER_ID
                              FROM USERS U, FRIENDSHIPS F
                              WHERE U.USER_ID = F.FRIEND_ID
```

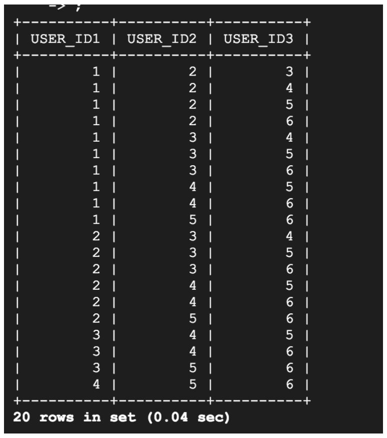
```
                AND F.USER_ID = 20 AND U.GENDER = 'F'
            ) FU1,
            (    SELECT U.USER_ID
                FROM USERS U, FRIENDSHIPS F
                WHERE U.USER_ID = F.FRIEND_ID
                AND F.USER_ID = 20 AND U.GENDER = 'F'
            ) FU2,
            (    SELECT U.USER_ID
                FROM USERS U, FRIENDSHIPS F
                WHERE U.USER_ID = F.FRIEND_ID
                AND F.USER_ID = 20 AND U.GENDER = 'F'
            ) FU3
            WHERE FU1.USER_ID < FU2.USER_ID
            AND FU2.USER_ID < FU3.USER_ID
            ORDER BY FU1.USER_ID, FU2.USER_ID, FU3.USER_ID
        ) UC, COMMENTS C
        WHERE C.COMMENTER_USER_ID = UC.USER_ID1
        AND C.POST_ID NOT IN
        (    SELECT P.POST_ID
            FROM POSTS P
            WHERE P.USER_ID = 10
            OR P.USER_ID = UC.USER_ID1
            OR P.USER_ID = UC.USER_ID2
            OR P.USER_ID = UC.USER_ID3
        )
        GROUP BY UC.USER_ID1, UC.USER_ID2, UC.USER_ID3
        HAVING COUNT(C.COMMENT_ID) >= 3
    ) AUG1, COMMENTS C
    WHERE C.COMMENTER_USER_ID = AUG1.USER_ID2
    AND C.POST_ID NOT IN
    (    SELECT P.POST_ID
        FROM POSTS P
        WHERE P.USER_ID = 10
        OR P.USER_ID = AUG1.USER_ID1
        OR P.USER_ID = AUG1.USER_ID2
        OR P.USER_ID = AUG1.USER_ID3
    )
    GROUP BY AUG1.USER_ID1, AUG1.USER_ID2, AUG1.USER_ID3
    HAVING COUNT(C.COMMENT_ID) >= 3
) AUG2, COMMENTS C
```

```
        WHERE C.COMMENTER_USER_ID = AUG2.USER_ID3
        AND C.POST_ID NOT IN
       (    SELECT P.POST_ID
            FROM POSTS P
            WHERE P.USER_ID = 10
            OR P.USER_ID = AUG2.USER_ID1
            OR P.USER_ID = AUG2.USER_ID2
            OR P.USER_ID = AUG2.USER_ID3
       )
       GROUP BY AUG2.USER_ID1, AUG2.USER_ID2, AUG2.USER_ID3
       HAVING COUNT(C.COMMENT_ID) >= 3
       ORDER BY
       (AUG2.COUNT1 + AUG2.COUNT2 + COUNT(C.COMMENT_ID)) DESC,
       AUG2.COUNT1 DESC, AUG2.COUNT2 DESC, COUNT(C.COMMENT_ID) DESC
       LIMIT 1
) A
UNION
SELECT A.USER_ID2, A.COUNT2
FROM
(    SELECT AUG2.USER_ID1, AUG2.COUNT1, AUG2.USER_ID2, AUG2.COUNT2,
     AUG2.USER_ID3, COUNT(C.COMMENT_ID) COUNT3,
     AUG2.COUNT1 + AUG2.COUNT2 + COUNT(C.COMMENT_ID) TOTAL
     FROM
     (    SELECT AUG1.USER_ID1, AUG1.COUNT1, AUG1.USER_ID2,
          COUNT(C.COMMENT_ID) COUNT2, AUG1.USER_ID3
          FROM
          (    SELECT UC.USER_ID1, COUNT(C.COMMENT_ID) COUNT1,
               UC.USER_ID2, UC.USER_ID3
               FROM
               (    SELECT FU1.USER_ID USER_ID1, FU2.USER_ID USER_ID2,
                    FU3.USER_ID USER_ID3
                    FROM
                    (    SELECT U.USER_ID
                         FROM USERS U, FRIENDSHIPS F
                         WHERE U.USER_ID = F.FRIEND_ID
                         AND F.USER_ID = 20 AND U.GENDER = 'F'
                    ) FU1,
                    (    SELECT U.USER_ID
                         FROM USERS U, FRIENDSHIPS F
                         WHERE U.USER_ID = F.FRIEND_ID
                    AND F.USER_ID = 20 AND U.GENDER = 'F'
                ) FU2,
                (    SELECT U.USER_ID
                    FROM USERS U, FRIENDSHIPS F
                    WHERE U.USER_ID = F.FRIEND_ID
                    AND F.USER_ID = 20 AND U.GENDER = 'F'
                ) FU3
                WHERE FU1.USER_ID < FU2.USER_ID
                AND FU2.USER_ID < FU3.USER_ID
                ORDER BY FU1.USER_ID, FU2.USER_ID, FU3.USER_ID
            ) UC, COMMENTS C
            WHERE C.COMMENTER_USER_ID = UC.USER_ID1
            AND C.POST_ID NOT IN
            (    SELECT P.POST_ID
                FROM POSTS P
                WHERE P.USER_ID = 10
                OR P.USER_ID = UC.USER_ID1
                OR P.USER_ID = UC.USER_ID2
                OR P.USER_ID = UC.USER_ID3
            )
            GROUP BY UC.USER_ID1, UC.USER_ID2, UC.USER_ID3
            HAVING COUNT(C.COMMENT_ID) >= 3
        ) AUG1, COMMENTS C
        WHERE C.COMMENTER_USER_ID = AUG1.USER_ID2
        AND C.POST_ID NOT IN
        (    SELECT P.POST_ID
            FROM POSTS P
            WHERE P.USER_ID = 10
            OR P.USER_ID = AUG1.USER_ID1
            OR P.USER_ID = AUG1.USER_ID2
            OR P.USER_ID = AUG1.USER_ID3
        )
        GROUP BY AUG1.USER_ID1, AUG1.USER_ID2, AUG1.USER_ID3
        HAVING COUNT(C.COMMENT_ID) >= 3
    ) AUG2, COMMENTS C
    WHERE C.COMMENTER_USER_ID = AUG2.USER_ID3
    AND C.POST_ID NOT IN
    (    SELECT P.POST_ID
        FROM POSTS P
        WHERE P.USER_ID = 10
        OR P.USER_ID = AUG2.USER_ID1
        OR P.USER_ID = AUG2.USER_ID2
        OR P.USER_ID = AUG2.USER_ID3
    )
    GROUP BY AUG2.USER_ID1, AUG2.USER_ID2, AUG2.USER_ID3
    HAVING COUNT(C.COMMENT_ID) >= 3
    ORDER BY
    (AUG2.COUNT1 + AUG2.COUNT2 + COUNT(C.COMMENT_ID)) DESC,
    AUG2.COUNT1 DESC, AUG2.COUNT2 DESC, COUNT(C.COMMENT_ID) DESC
    LIMIT 1
) A
UNION
SELECT A.USER_ID3, A.COUNT3
FROM
(   SELECT AUG2.USER_ID1, AUG2.COUNT1, AUG2.USER_ID2, AUG2.COUNT2,
    AUG2.USER_ID3, COUNT(C.COMMENT_ID) COUNT3,
    AUG2.COUNT1 + AUG2.COUNT2 + COUNT(C.COMMENT_ID) TOTAL
    FROM
    (   SELECT AUG1.USER_ID1, AUG1.COUNT1, AUG1.USER_ID2,
        COUNT(C.COMMENT_ID) COUNT2, AUG1.USER_ID3
        FROM
        (   SELECT UC.USER_ID1, COUNT(C.COMMENT_ID) COUNT1,
            UC.USER_ID2, UC.USER_ID3
            FROM
            (   SELECT FU1.USER_ID USER_ID1, FU2.USER_ID USER_ID2,
                FU3.USER_ID USER_ID3
                FROM
                (   SELECT U.USER_ID
                    FROM USERS U, FRIENDSHIPS F
                    WHERE U.USER_ID = F.FRIEND_ID
                    AND F.USER_ID = 20 AND U.GENDER = 'F'
                ) FU1,
                (   SELECT U.USER_ID
                    FROM USERS U, FRIENDSHIPS F
                    WHERE U.USER_ID = F.FRIEND_ID
                    AND F.USER_ID = 20 AND U.GENDER = 'F'
                ) FU2,
                (   SELECT U.USER_ID
                    FROM USERS U, FRIENDSHIPS F
                    WHERE U.USER_ID = F.FRIEND_ID
                AND F.USER_ID = 20 AND U.GENDER = 'F'
            ) FU3
            WHERE FU1.USER_ID < FU2.USER_ID
            AND FU2.USER_ID < FU3.USER_ID
            ORDER BY FU1.USER_ID, FU2.USER_ID, FU3.USER_ID
        ) UC, COMMENTS C
        WHERE C.COMMENTER_USER_ID = UC.USER_ID1
        AND C.POST_ID NOT IN
        (   SELECT P.POST_ID
            FROM POSTS P
            WHERE P.USER_ID = 10
            OR P.USER_ID = UC.USER_ID1
            OR P.USER_ID = UC.USER_ID2
            OR P.USER_ID = UC.USER_ID3
        )
        GROUP BY UC.USER_ID1, UC.USER_ID2, UC.USER_ID3
        HAVING COUNT(C.COMMENT_ID) >= 3
    ) AUG1, COMMENTS C
    WHERE C.COMMENTER_USER_ID = AUG1.USER_ID2
    AND C.POST_ID NOT IN
    (   SELECT P.POST_ID
        FROM POSTS P
        WHERE P.USER_ID = 10
        OR P.USER_ID = AUG1.USER_ID1
        OR P.USER_ID = AUG1.USER_ID2
        OR P.USER_ID = AUG1.USER_ID3
    )
    GROUP BY AUG1.USER_ID1, AUG1.USER_ID2, AUG1.USER_ID3
    HAVING COUNT(C.COMMENT_ID) >= 3
) AUG2, COMMENTS C
WHERE C.COMMENTER_USER_ID = AUG2.USER_ID3
AND C.POST_ID NOT IN
(   SELECT P.POST_ID
    FROM POSTS P
    WHERE P.USER_ID = 10
    OR P.USER_ID = AUG2.USER_ID1
    OR P.USER_ID = AUG2.USER_ID2
    OR P.USER_ID = AUG2.USER_ID3
)
GROUP BY AUG2.USER_ID1, AUG2.USER_ID2, AUG2.USER_ID3
```

```
        HAVING COUNT(C.COMMENT_ID) >= 3
        ORDER BY
        (AUG2.COUNT1 + AUG2.COUNT2 + COUNT(C.COMMENT_ID)) DESC,
        AUG2.COUNT1 DESC, AUG2.COUNT2 DESC, COUNT(C.COMMENT_ID) DESC
        LIMIT 1
    ) A
) AUG
ON F.USER_ID = AUG.USER_ID
ORDER BY AUG.COUNT DESC, F.NAME;
```

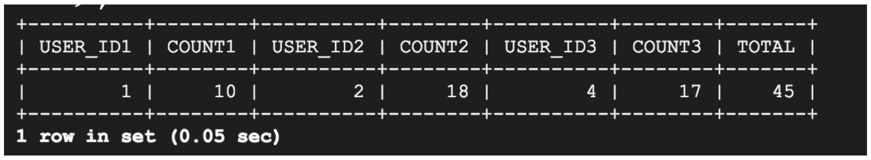
**NOTE**: For this query I have used the data provided in the example. So the results obtained are for that data and not the above data.

**Explanation:**
Firstly, find all the female users who are friends with USER_ID = 20. After we get the list of all these female users (FU) find various combinations by joining this list twice where FU1.USER_ID < FU2.USER_ID and FU2.USER_ID < FU3.USER_ID. This will give the various triplets/user combinations (UC) that are possible in the following format:



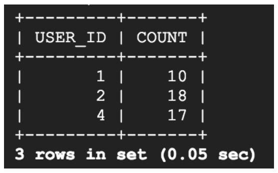
```
    -> ;
+----------+----------+----------+
| USER_ID1 | USER_ID2 | USER_ID3 |
+----------+----------+----------+
|        1 |        2 |        3 |
|        1 |        2 |        4 |
|        1 |        2 |        5 |
|        1 |        2 |        6 |
|        1 |        3 |        4 |
|        1 |        3 |        5 |
|        1 |        3 |        6 |
|        1 |        4 |        5 |
|        1 |        4 |        6 |
|        1 |        5 |        6 |
|        2 |        3 |        4 |
|        2 |        3 |        5 |
|        2 |        3 |        6 |
|        2 |        4 |        5 |
|        2 |        4 |        6 |
|        2 |        5 |        6 |
|        3 |        4 |        5 |
|        3 |        4 |        6 |
|        3 |        5 |        6 |
|        4 |        5 |        6 |
+----------+----------+----------+
20 rows in set (0.04 sec)
```

Now for each of these combinations find their augmented count of comments such that the count is >=3 starting with USER_ID1, then the result obtained (AUG1) is used to find the augmented count of comments for USER_ID2 and eventually this result (AUG2) is used for counting augmented comments for USER_ID3. After getting the individual counts we find the sum of these counts for each combination and store it in the separate column.
We sort the result in descending order of the sum of augmented count of comments. If there is a tie, we sort it in the descending order of augmented count 1, then augmented count 2 and then augmented count 3.
The top most record is selected as a result which is given an alias name ‘A’. This intermediate table has the following structure:



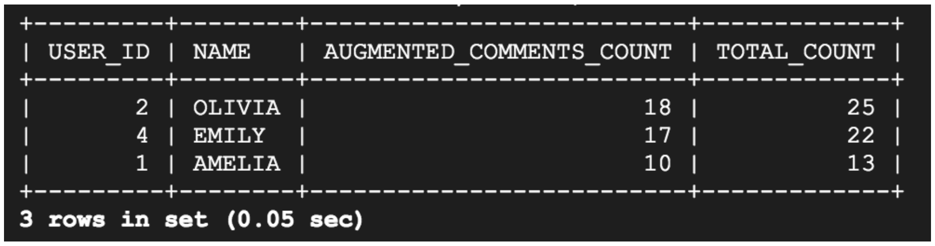
```
+----------+--------+----------+--------+----------+--------+-------+
| USER_ID1 | COUNT1 | USER_ID2 | COUNT2 | USER_ID3 | COUNT3 | TOTAL |
+----------+--------+----------+--------+----------+--------+-------+
|        1 |     10 |        2 |     18 |        4 |     17 |    45 |
+----------+--------+----------+--------+----------+--------+-------+
1 row in set (0.05 sec)
```

Now we perform a union on this intermediate result by selecting USER_ID1, COUNT1 first union with USER_ID2, COUNT2 union with USER_ID3, COUNT3. We get an intermediate result as follows which we give an alias name 'AUG':



```
+---------+-------+
| USER_ID | COUNT |
+---------+-------+
|       1 |    10 |
|       2 |    18 |
|       4 |    17 |
+---------+-------+
3 rows in set (0.05 sec)
```

Next, we find all the female users who are friends with USER_ID = 20 along with their total count of comments on all the posts. This result is given an alias name 'F'.
Finally, we perform a right join on F and AUG on F.USER_ID = AUG.USER_ID and sort them in the descending order of AUGMENTED_COMMENTS_COUNT and if there is a tie, we sort it in alphabetical order of NAME. The final result is as follows:



```
+---------+--------+--------------------------+-------------+
| USER_ID | NAME   | AUGMENTED_COMMENTS_COUNT | TOTAL_COUNT |
+---------+--------+--------------------------+-------------+
|       2 | OLIVIA |                       18 |          25 |
|       4 | EMILY  |                       17 |          22 |
|       1 | AMELIA |                       10 |          13 |
+---------+--------+--------------------------+-------------+
3 rows in set (0.05 sec)
```