1) LIST THE IDS AND NAMES OF THE USERS WHO HAVE NO POSTS AND HAVE ONE OR MORE COMMENTS ON POST_ID=5.

```
SELECT DISTINCT(U.USER_ID), U.NAME
FROM USERS AS U, POSTS,COMMENTS AS C
WHERE U.USER_ID NOT IN (SELECT USER_ID
                                        FROM POSTS)
AND
U.USER_ID IN(SELECT COMMENTER_USER_ID
                    FROM COMMENTS
                    WHERE POST_ID=5
                    GROUP BY COMMENTER_USER_ID
                     HAVING COUNT(COMMENT_ID)>0);
```

EXPLANATION:

IN THE ABOVE QUERY THERE ARE 2 SUBQUERIES, ONE RETURNING THE USER_ID OF ALL USERS WHO HAVE POSTED AND OTHER SELECTING THE COMMENTER_USER_ID FROM COMMENTS TABLE HAVING COMMENT COUNT AT LEAST 1 OR MORE. THE USER_ID OF THE USERS WHO ARE NOT IN THE RESULT OF THE FIRST SUBQUERY ARE SELECTED WHICH ACCOUNTS TO USERS WHO HAVE NOT POSTED, AND USER_ID OF THE COMMENTER WHO HAVE POSTED ON POST_ID=5 IS EXTRACTED FROM SECOND SUBQUERY. FINALLY, BOTH THE SUBQUERIES ARE AND'ED WITH THEIR RESPECTIVE ATTRIBUTE FROM THE MAIN QUERY TO GET THE USER_ID OF USERS WHO HAVE NOT POSTED AND COMMENTED ON POST_ID =5. USER_ID IS DISTINCT BECAUSE NOT IN CHECKS FOR EACH AND EVERY USER_ID WHICH MAY RESULT IN DUPLICATES.

2) LIST THE USER_ID OF FEMALE MUTUAL FRIEND BETWEEN USERS 1 AND 2.

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

EXPLANATION:

THE ABOVE QUERY RETURN RESULT OF USERS WHO ARE MUTUAL FRIEND TO USER 1 AND USER 2. THE FROM CLAUSE OF THE MAIN QUERY HAS A DERIVED TABLE WHICH RETURNS THE RESULT OF DISTINCT FRIEND_ID OF USER=1 AND ALSO FEMALE. SIMILARLY, WHERE CLAUSE OF THE MAIN QUERY CHECKS IF THE FRIEND_ID OF USER=1 IS IN THE RESLUT OF SUBQUERY GIVING US THE SET OF USERS WHO ARE FEMALES

AND FRIEND WITH USER_ID=2. THIS GIVES US THE SET OF USERS WHO ARE MUTUAL FRIEND TO USER_ID=1 AND USER_ID=2, THE COLUMN NAME OF THE RESULTANT QUERY IS ALIASED TO USER_IDENTIFICATION_NUMBER FOR SIMPLICITY.

3) LIST THE USERS_ID OF USERS WHO HAVE MORE THAN 2 FRIENDS WHOM HAVE AT LEAST ONE POST

```
SELECT USER_ID
FROM FRIENDSHIPS
WHERE FRIEND_ID IN (SELECT USER_ID
                     FROM POSTS
                    GROUP BY USER_ID
                    HAVING COUNT(POST_ID)>0)
GROUP BY USER_ID
HAVING COUNT(FRIEND_ID)>2;
```

EXPLANATION:

THE ABOVE QUERY SELECTS THE SET OF USERS WHO HAVE FRIENDS WHO MAKE AT LEAST ONE POST DENOTED BY COUNT OF POST_ID > 0 IN HAVING CALUSE. THIS SUBQUERY IS THEN USED AS A CONDITION TO CHECK IF THE FRIEND ID IN FRIENDSHIP THAT MATCHES WITH THE SUBQUERY RESULT. IF MATCHES, WHILE GROUPING BY USER_ID OF FRIENDSHIP TABLE FIND THE COUNT OF FRIEND_ID FOR THE GROUPED USERS AND CHECK IF IT IS GREATER THAN 2. IF ALL THE CONDITIONS SATISFIES THAN OUTPUT THE USERS WHO HAVE MORE THAN 2 FRIENDS HAVING AT LEAST ONE POST.

4) LIST UNIQUE USER_ID OF FEMALE USERS WHO WERE BORN AFTER '1990-12-20' AND COMMENTED ON POSTS OF USER_ID. SHOW THEIR FRIENDS COUNT IN SEPARATE COLUMN.

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

EXPLANATION:

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

THE TEXT WHICH IS COLORED RED IN THE ABOVE QUERY IS A SUBQUERY WHICH SELECTS THE COMMENTER _USER_ID FROM COMMENTS TABLE WHO HAVE COMMENTED ON THE POST_ID BY THE USER 10.

THE OUTPUT OF THIS IS FED AS INPUT TO THE BLUE COLORED TEXT WHEREIN USER_ID IS SELECTED SUCH THAT THE USER IS A FEMALE WHO IS BORN AFTER ‘1990-12-20’ AND IS IN THE OUTPUT OF RED TEXT. THIS WILL GIVE YOU USER_ID OF THE USER WHO COMMENTED ON POST POSTED BY USER_ID 10 AND FEMALE AND BORN AFTER ‘1990-12-20’.

LEFT JOIN IS PERFORMED ON THE ABOVE RESULT WITH FRIENDSHIPS TABLE ON THE CONDITION THAT USER_ID OF THE BOTH THE TABLE MATCHES. THIS WILL RESULT IN ALL THE POSSIBLE FRIENDS TO USER_ID IN THE RIGHT COLUMN WITH USER_ID BY THEMSELVES IN THE LEFT COLUMN.

THE ABOVE RESULT IS PASSED AS A TABLE WITH ALIASING AS N TO THE OUTER MOST QUERY IN FROM CLAUSE, THIS IS GROUPED BY USER_ID TO GET THE COUNT OF FRIEND_ID FOR THAT PARTICULAR USER. (PINK TEXT)

5) LIST THE USER_ID OF THE USER WHO COMMENTED ON POST_ID=7 AND ARE FRIENDS WITH POST CREATOR.

```
SELECT USER_ID
FROM FRIENDSHIPS
WHERE USER_ID IN (SELECT COMMENTER_USER_ID
                  FROM (SELECT COMMENTER_USER_ID,POST_ID
                        FROM COMMENTS
                        WHERE POST_ID=7)AS C )

AND FRIEND_ID =(SELECT USER_ID FROM POSTS WHERE POST_ID=7);
```

EXPLANATION:

IN THE OUTERMOST WHERE CLAUSE OF THE QUERY I USED 2 CONDITION ONE ON USER_ID PRESENT IN THE LIST OF USER ID'S FROM THE SUBQUERY ON SELECTING COMMENTER_USER_ID FOR THE POST_ID =7.
THE SECOND CONDITION IS TO OBTAIN THE USER_ID FOR POST_ID=7, WHICH RETURN A SINGLE OUTPUT.
FINALLY, WITH ALL THESE CONDITION THE OUTER QUERY WILL FIND ALL THE USERS WHO HAVE COMMENTED ON THE POST_ID 7 AND FRIENDS TO USER_ID OF THE POST CREATOR.

6)

```
SELECT COMM AS USER_ID,NAME,CNTT AS ACC,TOTAL


FROM (

SELECT COMM,CNTT,TOTAL

FROM(

SELECT S.COMMENTER_USER_ID AS COMM,CNTT, S.U1NEW,S.U2NEW,S.U3NEW



FROM
 (
SELECT U1 AS U1NEW,U2 AS U2NEW,U3 AS U3NEW,COUNT(COMMENT_ID) AS
CNTT,COMMENTER_USER_ID

FROM

(SELECT U1,U2,U3,
POST_THAT_CAN_BE_COMMENTED,COMMENT_ID,COMMENTER_USER_ID

FROM (

SELECT DISTINCT U1,U2,U3, POST_ID AS POST_THAT_CAN_BE_COMMENTED
FROM
(SELECT U1 , U2, U3
FROM (((SELECT DISTINCT(COMMENTER_USER_ID)  AS U1
FROM COMMENTS, FRIENDSHIPS AS F, USERS AS U
 WHERE F.USER_ID=20 AND U.GENDER='F' AND
U.USER_ID=COMMENTER_USER_ID AND F.FRIEND_ID=COMMENTER_USER_ID
AND POST_ID
NOT IN
(SELECT POST_ID
FROM POSTS  WHERE USER_ID=10)) AS C1)

CROSS JOIN

((SELECT DISTINCT(COMMENTER_USER_ID)  AS U2
FROM COMMENTS, FRIENDSHIPS AS F ,USERS AS U
WHERE F.USER_ID=20  AND U.GENDER='F'  AND
U.USER_ID=COMMENTER_USER_ID AND F.FRIEND_ID=COMMENTER_USER_ID
AND POST_ID
  NOT IN
 (SELECT POST_ID
FROM POSTS
WHERE USER_ID=10)) AS C2)

CROSS JOIN

 ((SELECT DISTINCT(COMMENTER_USER_ID) AS U3
FROM COMMENTS, FRIENDSHIPS AS F, USERS AS U
WHERE F.USER_ID=20 AND U.GENDER='F' AND
U.USER_ID=COMMENTER_USER_ID AND F.FRIEND_ID=COMMENTER_USER_ID
AND POST_ID  NOT IN  (SELECT POST_ID
FROM POSTS
WHERE USER_ID=10)) AS
 C3))
WHERE U1!=U2 AND U2!=U3 AND U3!=U1) AS COMB
LEFT JOIN


(SELECT POST_ID,USER_ID
FROM POSTS ) AS P1


ON COMB.U1!=P1.USER_ID AND COMB.U2!=P1.USER_ID AND
COMB.U3!=P1.USER_ID) AS NEWTB


LEFT JOIN

(SELECT POST_ID, COMMENT_ID,COMMENTER_USER_ID
FROM COMMENTS

) AS CMT

ON


(COMMENTER_USER_ID=NEWTB.U2 OR  COMMENTER_USER_ID=NEWTB.U3 OR
COMMENTER_USER_ID=NEWTB.U1 )
AND
NEWTB.POST_THAT_CAN_BE_COMMENTED=CMT.POST_ID) AS AA
GROUP BY U1NEW,U2NEW,U3NEW,COMMENTER_USER_ID
HAVING COUNT(COMMENT_ID)>3
) AS S







JOIN







(SELECT U1NEW,U2NEW,U3NEW, SUM(CNT) AS SUMMATION
FROM
(
SELECT U1 AS U1NEW,U2 AS U2NEW,U3 AS U3NEW,COUNT(COMMENT_ID) AS
CNT ,COMMENTER_USER_ID

FROM

(SELECT U1,U2,U3,
POST_THAT_CAN_BE_COMMENTED,COMMENT_ID,COMMENTER_USER_ID

FROM (

SELECT DISTINCT U1,U2,U3, POST_ID AS POST_THAT_CAN_BE_COMMENTED
FROM
(SELECT U1 , U2, U3
FROM (((SELECT DISTINCT(COMMENTER_USER_ID)  AS U1
FROM COMMENTS, FRIENDSHIPS AS F, USERS AS U
 WHERE F.USER_ID=20 AND U.GENDER='F' AND
U.USER_ID=COMMENTER_USER_ID AND F.FRIEND_ID=COMMENTER_USER_ID
AND POST_ID
NOT IN
(SELECT POST_ID
FROM POSTS  WHERE USER_ID=10)) AS C1)
CROSS JOIN
((SELECT DISTINCT(COMMENTER_USER_ID)  AS U2
FROM COMMENTS, FRIENDSHIPS AS F ,USERS AS U
WHERE F.USER_ID=20  AND U.GENDER='F'  AND
U.USER_ID=COMMENTER_USER_ID AND F.FRIEND_ID=COMMENTER_USER_ID
AND POST_ID
  NOT IN
 (SELECT POST_ID
FROM POSTS
WHERE USER_ID=10)) AS C2)
 CROSS JOIN
 ((SELECT DISTINCT(COMMENTER_USER_ID) AS U3
FROM COMMENTS, FRIENDSHIPS AS F , USERS AS U
WHERE F.USER_ID=20  AND U.GENDER='F'  AND
U.USER_ID=COMMENTER_USER_ID AND F.FRIEND_ID=COMMENTER_USER_ID
AND POST_ID  NOT IN  (SELECT POST_ID
FROM POSTS
WHERE USER_ID=10)) AS
 C3))
WHERE U1!=U2 AND U2!=U3 AND U3!=U1) AS COMB

LEFT JOIN


(SELECT POST_ID,USER_ID
FROM POSTS ) AS P1


ON COMB.U1!=P1.USER_ID AND COMB.U2!=P1.USER_ID AND
COMB.U3!=P1.USER_ID) AS NEWTB


LEFT JOIN

(SELECT POST_ID, COMMENT_ID,COMMENTER_USER_ID
FROM COMMENTS

) AS CMT

ON


(COMMENTER_USER_ID=NEWTB.U2 OR  COMMENTER_USER_ID=NEWTB.U3 OR
COMMENTER_USER_ID=NEWTB.U1 )
AND
NEWTB.POST_THAT_CAN_BE_COMMENTED=CMT.POST_ID) AS AA
GROUP BY U1NEW,U2NEW,U3NEW,COMMENTER_USER_ID
HAVING COUNT(COMMENT_ID)>3) AS B
GROUP BY U1NEW,U2NEW,U3NEW
HAVING SUM(CNT)
ORDER BY SUM(CNT) DESC) AS Y



USING
(U1NEW, U2NEW, U3NEW)
LIMIT 3) AS W


JOIN



(SELECT COUNT(C5.COMMENTER_USER_ID) AS TOTAL,C5.COMMENTER_USER_ID
AS COMMENTEE
FROM COMMENTS AS C5
GROUP BY C5.COMMENTER_USER_ID
HAVING COUNT(C5.COMMENTER_USER_ID) )AS E


ON COMM=COMMENTEE
```

```
ORDER BY CNTT DESC


)
AS FIN
, USERS
WHERE USER_ID=COMM
ORDER BY (CNTT) DESC;
```

EXPLANATION :

STEP 1: I AM SELECTING THE COMMENTER_USER_ID OF USERS WHO ARE FEMALE AND FRIEND WITH USER_ID =20 AND WHO COMMENTED POST ID IS NOT THE ONE POSTED BY THE USER_ID =10

STEP 2: PERFORM STEP 1, 2 MORE TIME SO THAT WE GET THE POSSIBLE COMBINATION OF USER WHO ARE FREIND WITH USER_ID 20 AND COMMENTED ON POST_ID NOT CREATED BY USER_ID=10. I HAVE USED CROSS JOIN ON ALL THREE SUBQUERIES TO GET ALL THE POSSIBLE COMBINATION OF USERS.

STEP 3: THIS CROSS JOINED TABLE IS USED AS A TABLE TO DISPLAY COMMENTED_USER U1, U2, U3. THE ABOVE CROSS JOIN MAY INCLUDE DUPLICATES USERS IN THE SAME ROW SAY (1, 2, 2) USERS 2 AND 2 ARE THE SAME TO AVOIDE THIS I HAVE USED A CONDITION IN THE WHERE CLAUSE SPECIFYING TO DISPLAY THE USERS (COMBINATION) WHERE ALL THE USERS IN THE COMBINATION ARE UNIQUE (U1!=U2 AND U2!=U3 AND U3!=U1)

STEP 4: THE RESULT FROM THE STEP 3 IS LEFT JOINED WITH THE POST TABLE ON THE CONDITION POSTER IS NOT AMONGST U1, U2, U3. THIS RESULTS IN THE POST_ID THAT CAN BE COMMENTED BY THE COMBINATION OF USERS.

STEP 5: STEP 4 IS USED AS SUBQUERY TO JOIN WITH COMMENTS TABLE TO GET COMMENTER_USER_ID ALONG WITH THE COMBINATION OF USERS.

STEP 6: IN THIS STEP, THE RESULT FROM STEP 5 A TEMPORARY TABLE ON WHICH GROUP BY CLAUSE ID USED TO GROUP BY U1,U2,U3 AND COMMENTER_USER_ID AND DETERMINE THE COUNT OF COMMENT BY THEM.

STEP 7: THEN SAME PROCEDURE IS REPEATED FROM STEP 1 TO STEP 6, WITH FURTHER ENHANCEMENT IN CALCULATING THE SUM OF AUGMENTED COMMENT FOR ALL THE GROUP OF USERS. THIS AUGMENTED SUM IS COMPARED AND THE HIGHEST IS TAKEN. THIS RESULT IN ANOTHER TABLE.

STEP 8: COMBINE THE RESULT FROM STEP 6 WITH STEP 7 USING JOIN OPERATOR USING CONDITION (U1NEW,U2NEW,U3NEW)

STEP 9: FINALLY, THE TABLE FROM STEP 8 IS JOINED WITH COMMENTER TABLE. IN THE COMMENTER TABLE TOTAL COUNT OF COMMENTER IS FOUND BY GROUPING THEM ON COMMENTER_USER_ID AND COUNTING ON COMMENTER_USER_ID.

STEP 10: AT LAST THE NAME OF THE USER IS RETRIEVED BASED ON COMMENTER_ID FROM STEP 9 ’s TABLE. FINALLY, ORDER THE ENTIRE TABLE ON AUGMENTED COUNT IN DESCENDING ORDER.

IF THERE IS A TIE BETWEEN THE SUM OF AUGMENTED SET OF COMMENTERS THAN OUTPUT THE TOP MOST COMMENTER’S GROUP FROM THE TABLE.