QUERIES AND EXPLANATIONS

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

```
SELECT USER_ID , NAME FROM USERS
WHERE USER_ID NOT IN ( SELECT DISTINCT USER_ID FROM POSTS ) AND
USER_ID IN ( SELECT DISTINCT COMMENTER_USER_ID FROM COMMENTS
WHERE POST_ID = 5 );
```

We select USER_ID column which contains an ID for a user and NAME column which contains the name of the user from USERS table.
In the WHERE clause we have 2 subqueries:

- The first selects USER_ID of all users who have a post in the POSTS table. We use NOT IN operator to specify that the outer query must not have any users who have a post in the POSTS table.
- The second subquery selects COMMENTER_USER_ID of all users who have comments on POST_ID 5 in the COMMENTS table. We use the IN operator to specify that the outer query must have users which satisfy the second subquery.

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

```
SELECT USER_ID FROM USERS WHERE GENDER ="F" AND
USER_ID IN (SELECT FRIEND_ID FROM FRIENDSHIPS WHERE USER_ID=1) AND
USER_ID IN (SELECT FRIEND_ID FROM FRIENDSHIPS WHERE USER_ID =2);
```

The idea used is that if a user shows up in friend lists of both users 1 and 2 then that person is a mutual friend. We use two subqueries to make sure that a USER_ID selected is present in friends lists of USER_ID 1 and 2.
The outer query filters out the users so that they are all female.
Hence, we get all female mutual friends between 1 and 2.

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

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

We select USER_IDs of users from USERS table.
To filter we use a subquery which selects all users from FRIENDSHIPS table who have more than 2 friends.
This subquery has another subquery within it which ensures that only those users are selected whose friends have a post in the POSTS table. We do this by picking up FRIEND_IDs only if they appear under all distinct USER_IDs from POSTS table.

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 DISTINCT USER_ID, COUNT(FRIEND_ID) AS FRIENDS_COUNT FROM FRIENDSHIPS
WHERE USER_ID IN ( 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 ) ) )
GROUP BY USER_ID;
```

We choose USER_ID and the count of their FRIEND_IDs from FRIENDSHIPS table.
We filter our users using a subquery which selects users from USERS table who are female and were born after 1990-12-20.
An additional subquery within this subquery is used to pick up a user from the COMMENTS table who commented on a post was written by USER_ID 10.
We pick up the POST_ID of the post written by user 10 from the POSTS table through another subquery within the previous.

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

```
SELECT USER_ID FROM FRIENDSHIPS WHERE
USER_ID IN (SELECT COMMENTER_USER_ID FROM COMMENTS WHERE POST_ID = 7) AND
FRIEND_ID IN(SELECT USER_ID FROM POSTS WHERE POST_ID = 7);
```

W choose a user from the FRIENDSHIPS table.
We filter the user by adding a subquery which chooses only the COMMENTER_USER_ID on post 7 from COMMENTS table.
We also use another filter to ensure that the USER_ID which created POST_ID 7 in the POSTS table shows up in the friends list of the user.

6.

```
SELECT U1.USER_ID,  U1.NAME, U2.USER_ID , U2.NAME, U3.USER_ID, U3.NAME
FROM USERS AS U1
JOIN USERS AS U2 ON U1.USER_ID < U2.USER_ID
JOIN USERS AS U3 ON U2.USER_ID < U3.USER_ID
WHERE U1.GENDER="F" AND U2.GENDER="F" AND U3.GENDER="F"
AND U1.USER_ID IN (SELECT USER_ID FROM FRIENDSHIPS
WHERE FRIEND_ID = 20 AND USER_ID IN (SELECT COMMENTER_USER_ID
FROM COMMENTS
GROUP BY COMMENTER_USER_ID HAVING COUNT(*) > 2))
AND U2.USER_ID IN (SELECT USER_ID FROM FRIENDSHIPS
WHERE FRIEND_ID = 20 AND USER_ID IN (SELECT COMMENTER_USER_ID
FROM COMMENTS
GROUP BY COMMENTER_USER_ID
HAVING COUNT(*) > 2))
AND U3.USER_ID IN(SELECT USER_ID FROM FRIENDSHIPS
WHERE FRIEND_ID = 20 AND USER_ID IN (SELECT COMMENTER_USER_ID
FROM COMMENTS
GROUP BY COMMENTER_USER_ID
HAVING COUNT(*) > 2))
ORDER BY U1.USER_ID, U2.USER_ID, U3.USER_ID;
```

The following query gives us all valid combinations of triplets that have user 20 as friend and are female. It does not show ACC or total comments.