HW2

Q1- List the ids and names of users who have no posts and have one or more comments on POST_ID=5

Ans:

```
SELECT USERS.USER_ID, USERS.NAME FROM USERS

WHERE NOT EXISTS(SELECT POSTS.USER_ID FROM POSTS)

AND EXISTS(SELECT COMMENTS.COMMENTER_USER_ID FROM COMMENTS

WHERE (COMMENTS.POST_ID=5)

GROUP BY(COMMENTS.COMMENTER_USER_ID) HAVING COUNT(COMMENTS.COMMENT_ID)>=1

AND USERS.USER_ID=COMMENTS.COMMENTER_USER_ID);
```

[Explain]

Step1: "users who have one or more comments on POST_ID=5" =>

using GROUP BY(COMMENTS.COMMENTER_USER_ID) to calculate how many comments posted by each user and using HAVING COUNT(COMMENTS.COMMENT_ID) to limit the number of comments.

Step 2: combine Step1 and info from table USERS based on the description "List the ids and names of users who have no posts" => that is,

SELECT USERS.USER_ID, USERS.NAME FROM USERS WHERE NOT EXISTS(SELECT POSTS.USER_ID FROM POSTS) INTERSECT Step1

However, MySQL doesn't have intersection operator, we can use EXISTS to simulate it.

For example,

```
SELECT contact_id, last_name, first_name FROM contacts WHERE contact_id < 100
INTERSECT SELECT customer_id, last_name, first_name FROM customers WHERE last_name <> 'Johnson';
```

It should be change into this form

```
SELECT contacts.contact_id, contacts.last_name, contacts.first_name FROM contacts
WHERE contacts.contact_id < 100
AND EXISTS (SELECT * FROM customers WHERE customers.last_name <> 'Johnson'
        AND customers.customer_id = contacts.contact_id
        AND customers.last_name = contacts.last_name
        AND customers.first_name = contacts.first_name);
```

Q2- List the USER_ID of female mutual friends between users 1 and 2.

Ans:

```
SELECT t.USER_ID FROM USERS AS t WHERE t.GENDER="F" AND t.USER_ID IN

(SELECT t1.USER_ID FROM FRIENDSHIPS AS t1 WHERE t1.FRIEND_ID=1

AND t1.USER_ID IN (SELECT t2.USER_ID FROM FRIENDSHIPS AS t2 WHERE t2.FRIEND_ID=2));
```

[Explain]

Step1: combine info from table FRIENDSHIPS based on the description: "mutual friends between users 1 and 2" => that is,

```
SELECT USER_ID FROM FRIENDSHIPS WHERE FRIEND_ID=1 INTERSECT SELECT USER_ID FROM FRIENDSHIPS WHERE FRIEND_ID=2;
```

However, MySQL doesn't have intersection operator, we can use IN to simulate it.

For example,

```
SELECT category_id FROM products INTERSECT SELECT category_id FROM inventory;
```

It should be change into this form

```
SELECT products.category_id FROM products WHERE products.category_id IN (SELECT inventory.category_id FROM inventory);
```

Therefore, we have the part

```
(SELECT t1.USER_ID FROM FRIENDSHIPS AS t1 WHERE t1.FRIEND_ID=1

AND t1.USER_ID IN (SELECT t2.USER_ID FROM FRIENDSHIPS AS t2 WHERE t2.FRIEND_ID=2);
```

Step 2: combine info from table USERS and Step1 according to the description: "the USER_ID of female …" => that is,

```
SELECT USER_ID FROM USERS WHERE GENDER="F" INTERSECT result from Step1
```

Q3- List the USER_ID of users who have more than 2 friends whom have at least one post.

Ans:

```
SELECT DISTINCT USER_ID FROM FRIENDSHIPS

WHERE EXISTS (SELECT DISTINCT USER_ID FROM POSTS)

GROUP BY (USER_ID) HAVING COUNT(FRIEND_ID)>=2;
```

[Explain]

Step1: “people who have at least one post” (USER_ID and FRIEND_ID is a pair in FRIENDSHIPS)

```
SELECT DISTINCT USER_ID FROM FRIENDSHIPS WHERE EXISTS (SELECT DISTINCT USER_ID FROM POSTS)
```

Step2: “List the USER_ID of users who have more than 2 friends”

```
SELECT USER_ID FROM FRIENDSHIPS GROUP BY (USER_ID) HAVING COUNT(FRIEND_ID)>=2
```

Step3: combine Step1 & Step2

Q4- 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.

Ans:

```
SELECT FRIENDSHIPS.USER_ID, COUNT(FRIENDSHIPS.FRIEND_ID) FROM FRIENDSHIPS

WHERE FRIENDSHIPS.USER_ID IN (

SELECT USERS.USER_ID FROM USERS

WHERE (USERS.GENDER="F" AND USERS.DATE_OF_BIRTH > '1990-12-20')

AND EXISTS(

SELECT DISTINCT COMMENTS.COMMENTER_USER_ID FROM COMMENTS

WHERE COMMENTS.POST_ID IN (SELECT POSTS.POST_ID FROM POSTS

WHERE (POSTS.USER_ID=10))

AND USERS.USER_ID = COMMENTS.COMMENTER_USER_ID))

GROUP BY (FRIENDSHIPS.USER_ID);
```

[Explain]

Step1: “posts of USER_ID=10” => that is,

```
Result = SELECT POST_ID FROM POSTS WHERE (USER_ID=10)
```

Step2: “users commented on posts found in Step1”

ReVal = SELECT DISTINCT COMMENTER_USER_ID FROM COMMENTS WHERE (POST_ID IN Result)

Step3: “USER_ID of female users who were born after 1990-12-20”

Temp = SELECT USER_ID FROM USERS WHERE GENDER="F" AND DATE_OF_BIRTH > '1990-12-20'
AND EXISTS (ReVal AND USER_ID = COMMENTER_USER_ID)

Step4: “Show their friends count in a separate column”

SELECT USER_ID, COUNT(FRIEND_ID) FROM FRIENDSHIPS WHERE (USER_ID IN Temp)

Q5- List the USER_ID of users who commented on POST_ID=7 and are friends with the post creator.

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

[Explain]

Step1: “the USER_ID of users who commented on POST_ID=7” => that is,

```
SELECT DISTINCT COMMENTS.COMMENTER_USER_ID FROM COMMENTS WHERE (COMMENTS.POST_ID=1007)
```

Step2: “List the USER_ID of users that are friends with the post (POST_ID=7) creator.”

```
SELECT DISTINCT FRIENDSHIPS.FRIEND_ID FROM FRIENDSHIPS WHERE (
FRIENDSHIPS.USER_ID IN (SELECT DISTINCT POSTS.USER_ID FROM POSTS WHERE(POSTS.POST_ID=7)))
```

Step3: combine Step1 & Step2

Q6