CREATE COMMANDS

CREATE TABLE USERS(

USER\_ID int Primary Key,

FNAME varchar(50) NOT NULL,

LNAME varchar(50) NOT NULL,

GENDER ENUM('M','F') NOT NULL,

DATE\_OF\_BIRTH DATE NOT NULL);

USE enum because we want to restrict it to two values for gender, mysql by default store value in YYYY-MM-DD by using DATE variable, FNAME and LNAME has varchar data type because it has written that size is 50 at max.

CREATE TABLE FRIENDSHIPS(

INVITER\_ID int NOT NULL,

INVITEE\_ID int NOT NULL,

STATUS ENUM('0','1') DEFAULT '0' NOT NULL,

PRIMARY KEY(INVITER\_ID, INVITEE\_ID),

FOREIGN KEY(INVITER\_ID) REFERENCES USERS(USER\_ID),

FOREIGN KEY(INVITEE\_ID) REFERENCES USERS(USER\_ID)

);

USER ENUM to restrict STATUS value to ‘0’,’1’ and INVITER\_ID, INVITEE\_ID are composite primary key so part table constraint. I need to put status to ‘0’ as default because it explicitly states that all value must be NOT NULL so I assume as soon as inviter invited the invitee its status is ‘0’

CREATE TABLE POSTS(

POST\_ID int Primary Key,

USER\_ID int NOT NULL,

TEXT TEXT NOT NULL,

FOREIGN KEY(USER\_ID) REFERENCES USERS(USER\_ID)

);

CREATE TABLE COMMENTS(

COMMENT\_ID int Primary Key,

POST\_ID int NOT NULL,

USER\_ID int NOT NULL,

TEXT TEXT NOT NULL,

FOREIGN KEY(USER\_ID) REFERENCES USERS(USER\_ID),

FOREIGN KEY(POST\_ID) REFERENCES POSTS(POST\_ID));

Q1

SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITEE\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITER\_ID = 5 AND f.INVITEE\_ID != f.INVITER\_ID

UNION

SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITER\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITEE\_ID = 5 AND f.INVITEE\_ID != f.INVITER\_ID;

Friends can be exist in two ways one he send request and other way when he receives request so I used union for that. Also a person can't be friend of himself but given schema allow this so to remove this I added condition that inviter and invitee can't be same. Also if there are two entries that is one inviter accept invitee request but invitee doesnot accept inviter request then I take or that is friendship is valid.

Q2

SELECT USER\_ID, FNAME, LNAME, GENDER, DAY(DATE\_OF\_BIRTH) as DAY\_OF\_BIRTH

FROM FRIENDSHIPS JOIN USERS ON INVITER\_ID = USER\_ID

WHERE INVITEE\_ID = 1 AND STATUS = '0' AND INVITER\_ID != INVITEE\_ID

AND NOT EXISTS(

SELECT m.STATUS FROM FRIENDSHIPS m WHERE m.INVITER\_ID = 1 AND

m. INVITEE\_ID = USER\_ID AND m.STATUS = '1');

Here, I mean from day of month is the actual day from the format yyyy-mm-dd that's why I used DAY() function. And second will only be showing in the pending list if from invitee in this case USER\_ID 1 doesnot accept their request but due to database limitation person cannot be pending friend of himself so we need to eliminate that. Also suppose both person send the request and only one of them accepted the request in that case it again not shown in the pending list so that's why I used exists to eliminate the case.

Q3

SELECT t.USER\_ID as USER\_ID, t.FNAME as FNAME, t.LNAME as LNAME

FROM(

SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITEE\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITER\_ID = 1 AND f.INVITEE\_ID != f.INVITER\_ID AND u.GENDER = 'F'

UNION

SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITER\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITEE\_ID = 1 AND f.INVITEE\_ID != f.INVITER\_ID AND u.GENDER = 'F')

t,(

SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITEE\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITER\_ID = 2 AND f.INVITEE\_ID != f.INVITER\_ID AND u.GENDER = 'F'

UNION

SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITER\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITEE\_ID = 2 AND f.INVITEE\_ID != f.INVITER\_ID AND u.GENDER = 'F') q

WHERE t.USER\_ID = q.USER\_ID;

I am assuming that the users mention here are USER\_ID 1 and 2.

Join the table listing female friends of user\_id 2 using above query and table listing friends of user\_id 1 and join them on user\_id gives us intersection that is the mutual friend lists.

Q4

SELECT COUNT(\*) as NUMBER\_OF\_FRIENDS, m.MAIN as USER\_ID

FROM(SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME, f.INVITER\_ID as MAIN

FROM FRIENDSHIPS f JOIN USERS u

ON f.INVITEE\_ID = u.USER\_ID

WHERE f.STATUS = '1'

AND f.INVITER\_ID IN(

SELECT u.USER\_ID FROM USERS u

JOIN ( SELECT DISTINCT(COMMENTS.USER\_ID)

FROM POSTS

JOIN COMMENTS ON POSTS.POST\_ID = COMMENTS.POST\_ID

WHERE POSTS.USER\_ID=10 )m

ON u.USER\_ID = m.USER\_ID WHERE u.GENDER = 'F' AND u.DATE\_OF\_BIRTH > '1990-12-20')

AND f.INVITEE\_ID != f.INVITER\_ID

UNION

SELECT u.USER\_ID as USER\_ID, u.FNAME as FNAME, u.LNAME as LNAME, f.INVITEE\_ID as MAIN

FROM FRIENDSHIPS f JOIN USERS u ON f.INVITER\_ID = u.USER\_ID

WHERE f.STATUS = '1'

AND f.INVITEE\_ID IN(SELECT u.USER\_ID FROM USERS u

JOIN (

SELECT DISTINCT(COMMENTS.USER\_ID) FROM POSTS

JOIN COMMENTS ON POSTS.POST\_ID = COMMENTS.POST\_ID

WHERE POSTS.USER\_ID=10 )m

ON u.USER\_ID = m.USER\_ID

WHERE u.GENDER = 'F' AND u.DATE\_OF\_BIRTH > '1990-12-20')

AND f.INVITEE\_ID != f.INVITER\_ID) m GROUP BY m.MAIN;

I make assumption as time is not mention that born after ‘1990-12-20’ means next day not that day itself.

Here, I first found the number of girls commented on the post of USER\_ID 10 and born after ‘1990-12-20’ then find out their friends in the table so that all friends of the girls can be kept in one table so that we can apply groupby to retrieve resultant result. HERE I used NUMBER\_OF\_FRIENDS to represent their friend count as it is not mention in the question.

Q5

SELECT temp.FIR as FIRST, temp.SECOND as SECOND

FROM

(SELECT fir.USER\_ID as FIR, second.USER\_ID as SECOND

FROM USERS fir, USERS second

WHERE fir.USER\_ID!=second.USER\_ID AND

NOT EXISTS

(

SELECT \* FROM FRIENDSHIPS

WHERE ((INVITER\_ID = fir.USER\_ID AND INVITEE\_ID=second.USER\_ID AND STATUS = '1')

OR (INVITEE\_ID = fir.USER\_ID AND INVITER\_ID=second.USER\_ID AND STATUS = '1'))

))temp,

(

SELECT u.USER\_ID as USER\_ID, f.INVITER\_ID as Main\_first

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITEE\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITER\_ID IN( SELECT USER\_ID FROM USERS)

AND f.INVITEE\_ID != f.INVITER\_ID

UNION

SELECT u.USER\_ID as USER\_ID, f.INVITEE\_ID as Main\_first

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITER\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITEE\_ID IN( SELECT USER\_ID FROM USERS)

AND f.INVITEE\_ID != f.INVITER\_ID

)temp\_first,

(

SELECT u.USER\_ID as USER\_ID, f.INVITER\_ID as Main\_Second

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITEE\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITER\_ID IN( SELECT USER\_ID FROM USERS)

AND f.INVITEE\_ID != f.INVITER\_ID

UNION

SELECT u.USER\_ID as USER\_ID, f.INVITEE\_ID as Main\_first

FROM FRIENDSHIPS f JOIN USERS u ON

f.INVITER\_ID = u.USER\_ID

WHERE f.STATUS = '1' AND f.INVITEE\_ID IN( SELECT USER\_ID FROM USERS)

AND f.INVITEE\_ID != f.INVITER\_ID

)temp\_second

WHERE temp.FIR = temp\_first.Main\_first

AND temp.SECOND = temp\_second.Main\_Second

AND temp\_second.USER\_ID = temp\_first.USER\_ID

AND temp.FIR < temp.SECOND

GROUP BY temp.FIR,temp.SECOND

LIMIT 10;

HERE, I first find the pair of users using join such that they must not be direct friends

After that I find all the friends of each user in pair then join them to get intersection of friend list of each user to see if they have common friend if they have then that pair is the answer to avoid duplication I added condition that First userId is less second userId. And as we need to find only 10 such pair I used **Limit function** to do that. Here I assume we just need to show USER\_ID of pair that’s why I used FIRST, SECOND for this purpose they represent USER\_ID of users in a pair.

Q6

SELECT temp\_result1.Maker as FIRST, temp\_result1.Commentor as SECOND

FROM

(SELECT result.Maker, result.Commentor, SUM(result.Num\_Comments) as Number\_Comments

FROM

(SELECT post\_temp.Maker,post\_temp.POST\_ID, comment\_temp.Commentor, COUNT(comment\_temp.COMMENT\_ID) as Num\_Comments FROM

(SELECT fir.USER\_ID as FIR, second.USER\_ID as SECOND

FROM USERS fir, USERS second

WHERE fir.USER\_ID != second.USER\_ID)temp

JOIN

(SELECT POST\_ID, USER\_ID as Maker

FROM POSTS) post\_temp

ON temp.FIR = post\_temp.Maker

JOIN

(SELECT COMMENT\_ID, POST\_ID, USER\_ID as Commentor

FROM COMMENTS)comment\_temp

ON post\_temp.POST\_ID = comment\_temp.POST\_ID

WHERE comment\_temp.Commentor = temp.SECOND

GROUP BY post\_temp.Maker,post\_temp.POST\_ID, comment\_temp.Commentor

ORDER BY post\_temp.Maker, Commentor)result

JOIN USERS creater

ON creater.USER\_ID = result.Maker

JOIN USERS comm

ON result.Commentor = comm.USER\_ID

WHERE creater.GENDER != comm.GENDER

GROUP BY result.Maker,result.Commentor

HAVING SUM(result.Num\_Comments)>=5)temp\_result1,

(SELECT result.Maker, result.Commentor, SUM(result.Num\_Comments) as Number\_Comments

FROM

(SELECT post\_temp.Maker,post\_temp.POST\_ID, comment\_temp.Commentor, COUNT(comment\_temp.COMMENT\_ID) as Num\_Comments FROM

(SELECT fir.USER\_ID as FIR, second.USER\_ID as SECOND

FROM USERS fir, USERS second

WHERE fir.USER\_ID != second.USER\_ID)temp

JOIN

(SELECT POST\_ID, USER\_ID as Maker

FROM POSTS) post\_temp

ON temp.FIR = post\_temp.Maker

JOIN

(SELECT COMMENT\_ID, POST\_ID, USER\_ID as Commentor

FROM COMMENTS)comment\_temp

ON post\_temp.POST\_ID = comment\_temp.POST\_ID

WHERE comment\_temp.Commentor = temp.SECOND

GROUP BY post\_temp.Maker,post\_temp.POST\_ID, comment\_temp.Commentor

ORDER BY post\_temp.Maker, Commentor)result

JOIN USERS creater

ON creater.USER\_ID = result.Maker

JOIN USERS comm

ON result.Commentor = comm.USER\_ID

WHERE creater.GENDER != comm.GENDER

GROUP BY result.Maker,result.Commentor

HAVING SUM(result.Num\_Comments)>=5)temp\_result2

WHERE

((temp\_result1.Maker = temp\_result2.Commentor) AND (temp\_result1.Commentor=temp\_result2.Maker))

AND temp\_result1.Commentor > temp\_result1.Maker

LIMIT 10;

Here, I divided problem into separate table using subquery then operate on the them using joins

First I used cross join on user to get list of pairs and put where condition so that no two elements in pair are same. In this First one I used as the person who make Post and other person who make comments on this person notes using joins I get that data and by group by on post\_maker, postId, commentor\_id I get number of comments that person posted on that comments then the resultant table I join with the user table again to eliminate those cases where maker and commenter belong to same gender after that GROUP\_BY on post\_maker and commenter to get total number of comments and filter the result using having then the resultant table I use self join to those result where poster\_maker and commenter post on eachother at least 5 times and used one extra condition which is poster\_maker id < commentor\_id to remove the duplicate entry that is like this

1,4

4,1

AND use limit to get at max 10 result.