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.