Project 1: Using SQL to create, populate and query a database

Introduction to Database Systems (CS 4163/6523)

Points: 100

You will use a MySQL DBMS, installed on our university server, to create, populate and query a database corresponding to a social network miniworld. To connect to the server, first download and install the current version of MySQL WorkBench. You should then create a connection to the TU MySQL server: use Hostname 129.244.40.38 and username/password as received from your course instructor. *If you are connecting from outside the university network, you have to first connect to our VPN (see* ***https://utulsa.edu/information-technology/support/vpn/*** *for instructions).* To create and populate tables, create a file with `create table` and `insert values` commands in a file called *socialNetworkData.sql* and load the file by selecting the `‘‘Open SQL Script’’` option from the `‘‘File’’` dropdown menu. You can execute the entire file to create and populate the database. You can use `‘‘New Query Tab’’` option from the from the `‘‘File’’` dropdown menu to create a new window for devloping the SQL queries in an interactive mode. You can also store all the queries into a *socialNetworkQueries.sql* file that you can load and execute, either for all the queries or for a select subset, at any time.

Creating and populating tables (20 points)

You will be working on a relational database for a prototype social network. Create the following tables with given attributes, data types, and constraints:

User: Id (5 digit character string), Name (variable length character string), Gender (Character);

Friends: Id1 (5 digit character string), Id2 (5 digit character string), Startdate (Date);

Comments: CommentId (Number), Poster (5 digit character string), Recipient (5 digit character string), Text (variable length character string), PostDate (Date);

The primary keys for User, Friends, and Comments tables are (Id), (Id1, Id2), and (CommentId) respectively. Id1, Id2, Poster, and Recipient are all foreign keys that refer to Id. Additionally, the following attributes cannot be assigned NULL values: Name, Text. Select set of character values that can be assigned to Gender and use the CHECK clause for enforcement.

Note that a friendship between two users will be entered twice in the Friends table, with each user listed as Id1 in one row and Id2 in the other. Create at least 5 rows for the User table and at least 10 rows for the Friends and Comments tables. Choose the data for the tables such that each of the queries mentioned later returns at least one row.

Queries (80 points)

Answer the following queries using single SQL statements. Replace variables in the queries with constants depending on the data you have entered such that each query selects at least one row.

1. List all attributes of comments containing the 'word' *lol*, ordered by decreasing Poster and increasing Recipient. **(4 points)**
2. Display, without duplicates, the Posters of one gender who posted to Recipients of a different gender (*display Poster and Recipint Ids and Genders*). **(4 points)**
3. List all attributes of comments posted this year either by Poster with name X or to Recipient with name Y. **(6 points)**
4. For each user, find the number of friends of each gender (*output should have 3 columns: Id, Gender, Count*). **(6 points)**
5. List all attributes of Users who do not have any friends of the same gender. **(9 points)**
6. List all attributes of Users who have started at least two friendships on the same date. **(9 points)**
7. List all User attributes and the number of friends of users who have commented to at least two different users. **(12 points)**
8. List all attributes of Users who have posted comments to all users of a specific gender (gender chosen from your defined set; no aggregate operations allowed) **(15 points)**
9. List all attributes of User(s) who are friends with the most number of Users. **(15 points)**

For finding Users, it is sufficient to list their Ids. The DATE data can be entered as '1861-05-07' for 7th May, 1861 and TIMESTAMP data as '2000-01-01 00:01:02.03'.

Submit, via Harvey, your *socialNetworkData.sql* and *socialNetworkQueries.sql* files. In addition, submit a text file, *socialNeworkQueryResults*, that contains clearly formatted listings of each query (first the query as stated in the assignment, followed by the SQL query) and the corresponding output.