Database Systems for Software Engineers

SOEN 363 - Winter 2022

Assignment 2

Out: Feb 02, 2022

Due: Feb 15, 2022

1 SQL Queries [100 Points]

Consider a simplified version of a known social application, *Twitter*! At a high-level, *Twitter* works as follows:

- A `User` may post a `Tweet` that is a short piece of text. a `Tweet` may be `retweeted` by any user, including the original poster.
- A `User` may `tag` their tweets with zero or more hashtags of their own choices. A hashtag must begin with the '#' sign . For example, a user tweeting about the Database Applications course may decide to tag the tweet with `#DBApps #socool`.
- Users may `follow` zero or more other users. Whenever a user logs in, they are assumed to read ALL tweets by the users they follow.

Given the above *Twitter*'s description, we define the following relation schemas:

Users(userId, userName, city, lastLogin)
Tweets(tweetId, userId, tweetTimestamp, tweetText)
Retweets(userId, tweetId, retweetTimestamp)
TweetTags(tweetId, hashtag)
Follows(followerId, followeeId)

- `userName` is in the format "<`first_name`> <`last_name`>".
- Original tweets are added to `Tweets` with the poster's `userId`. `Retweets` contains retweets by any user, including the original poster.

Now, we would like to extract some useful information from the database and we leave this job to our database expert (you!). For each of the following problems, write an SQL query. State the reason clearly if an expression and/or query cannot be expressed. You may want to populate your database with mock data to test your queries. For your ease, you are given the SQL table creation queries.

```
CREATE TABLE Users(
    userId SERIAL PRIMARY KEY,
    userName VARCHAR,
    city VARCHAR,
    lastLogin TIMESTAMP);

CREATE TABLE Tweets(
    tweetId SERIAL PRIMARY KEY,
    userId INT NOT NULL REFERENCES Users(userId),
    tweetTimestamp TIMESTAMP NOT NULL,
    tweetText VARCHAR);

CREATE TABLE Retweets(
    userId INT REFERENCES Users(userId),
    tweetId INT REFERENCES Tweets(tweetId),
    retweetTimestamp TIMESTAMP,
    PRIMARY KEY (userId, tweetId, retweetTimestamp));

CREATE TABLE TweetTags (
    tweetId INT,
    hashtag VARCHAR CHECK (hashtag LIKE '#%'),
    PRIMARY KEY (tweetId, hashtag));

CREATE TABLE Follows(
    followerId INT REFERENCES Users(userId),
    followeeId INT REFERENCES Users(userId),
    PRIMARY KEY (followerId, followeeId));
```

5pts (a) Find all users who retweeted a tweet that was tagged with "#MachineLearning" before 2008. Report the user names of the tweeter and retweeter as well as the tweet text and date.

10pts (b) Find all users who used the highest number of distinct hashags in their tweets, report the user name and number of distinct hashtags.

10pts (c) Find the most retweeted hashtag since the beginning of 2021, report the hashtag and number of distinct retweets. In case there are multiple such hashtags, return any of them.

10pts (d) Find all users (`userName` and `city`) who follow users who follow user "Yoshua Bengio". You must use nested select statements to write this query.

5pts (e) Write the previous query using join operations.

10pts (f) A **news feed** for a user consists of both tweets and retweets by people they directly follow and have been posted after the user's last login. Get the news feed for the user "Geoffrey Hinton". You need to report the tweet text only.

10pts (g) Find the top 5 users who have the most number of tweets in their **news feed**. Do not consider retweets in this query.

15pts (h) Find users whose tweets have never been retweeted.

10pts (i) Find top 10 fans of "Andrew Ng", i.e. people who retweeted his tweets the most.

15pts (j) Find all users (`uname` and `city`) who follow everyone that "Ian Goodfellow" follows.

2 Submission

- The assignment is due at 11:59PM on February 15, 2022.
- Answers for the previous queries must be in the SQL dialect of **Postgres**. You might want to install PostgresSQL and pgAdmin on your machine or use an online DBMS such as `https://sqliteonline.com`
- You are not allowed to modify the SQL creation queries provided. Modifying them might delay grading your assignment and you will incur a 20% penalty.
- The submission for this assignment consists of one Zip file containing:
  - One directory `queries` containing a `.txt` file per problem with the following naming format `a.txt, b.txt, ...`
  - One directory `results` containing the return results for each problem. Use the same naming format; you can have the results as screenshots or textual output from your DBMS.
  - A README file mentioning your name and ID and any comments regarding your submission.
  - OPTIONAL: `.txt` or `.sql` file containing insert statements to populate the tables.
- Your queries and statements **must** be digitally typed (photos of handwritten answers are not allowed).
- If you have any questions regarding the submission, contact your respective TA at least two hours before the submission deadline:

- Monday lab SF: Philippe (p_arrie@live.concordia.ca)
- Monday lab SA: Ahmed (ahmed.aly.20211@mail.concordia.ca)
- Tuesday lab SB: Yasaman (y_sabbag@live.concordia.ca)
- Tuesday lab SE: Hussein Abdallah (hussein.abdallah@mail.concordia.ca)
- Wednesday lab SC: Mossad Helali (mossad.helali@mail.concordia.ca)
- Wednesday lab SD: Reham Omar (reham.omar@mail.concordia.ca)

3 Late Policy

- 0-24 hours late = 25% penalty.
- 24-48 hours late = 50% penalty.
- More than 48 hours late = you lose all the points for this assignment.
- Submissions of corrupted files, blank files, or the assignment sheet are not accepted and will be considered late submissions.