Middle East Technical University

Department of Computer Engineering

CENG 352
Database Management Systems
Spring 2020–2021
Project 1

1 Introduction

In this project you are supposed to write many SQL queries on a relational database which will be created by using a dataset. The dataset is taken from Yelp Open Dataset which is available for access to users for personal and non-commercial use. You can reach it from here. Also, you can find the documentation on the same link if you are interested in the dataset. Yelp Dataset contains too much information for this task, so we have done some cleaning to reduce the size of the dataset and converted json files to csv files. You can reach the data which will be used on this project from **here.** This project has three parts:

- Create the database using the given 'csv' files.
- Write proper SQL queries for certain problems.
- Create triggers and views.

Note that all tasks should be completed using **PostgreSQL**.

2 Database Schema

Business(business_id, business_name, address, state, is_open, stars)
Users(user_id, user_name, review_count, yelping_since, useful, funny, cool, fans, average_stars)
Friend(user_id1, user_id2)
Review(review_id, user_id, business_id, stars, date, useful, funny, cool)
Tip(tip_id,business_id, user_id, date, tip_text, compliment_count)

Yelp is a system in which you can find any kind of service such as restaurants, dentists or vet. You can see businesses in your area, their stars (measure for appreciation given by other users) and also you can vote on such businesses that you've visited. The main purpose of Yelp is create an environment for users to share information among themselves.

2.1 Foreign Key Constraints

Friend's user_id1 and user_id2 references Users' user_id.
Review's user_id references Users' user_id.
Review's business_id references Business's business_id.
Tip's business_id references Business's business_id.
Tip's user_id references Users' user_id.

2.2 Explanation of Attributes

Business

- **business_id**: Primary key of the business table that stores id's of each business.
- **business_name**: Name of that business.
- **address**: Address of the business.
- **state**: State code that the business is located at.
- **is_open**: Boolean value, that indicates whether the business still working or closed completely.
- **stars**: Float value, average stars given to that business by users.

Users

- **user_id**: Primary key of the Users table that stores id's of each user.
- **user_name**: Name of the user.
- **review_count**: Indicates how many reviews have been made by that user.
- **yelping_since**: The date user started using Yelp system.
- **useful**: Count of this useful votes that sent by this user.
- **funny**: Count of this funny votes that sent by this user.
- **cool**: Count of this cool votes that sent by this user.
- **fans**: Number of fans of that user.
- **average_stars**: Float value, average stars given by that user.

Friend

- **user_id1**: First user's id.
- **user_id2**: Friend of the first user.

Review

- **review_id**: Primary key of the Review table that stores id's of each review.
- **user_id**: User that gave the review.
- **business_id**: Business that is reviewed by the user.
- **stars**: Given stars to the business by the user.
- **date**: Review date.
- **useful**: Count of useful tags given by other users to this review.
- **funny**: Count of funny tags given by other users to this review.
- **cool**: Count of cool tags given by other users to this review.

Tip

- **tip_id**: The id of the tip. **It is NOT** provided in the .csv file, it must be an auto incremented value.
- **business_id**: Business that is given a tip.

- **user_id**: User that gave the tip about the business.
- **date**: Date of the given Tip.
- **compliment_count**: Compliments that the tip is received.
- **tip_text**: String, text of the tip.

2.3 General Information About Relations

- Reviews are supposed to be detailed reports created by a user for a business. However, since Review texts are too large in terms of bytes we've ignored them.
- Tips are shorter reviews like 'this restaurant's pizza is really good.'. They are similar to the reviews but less detailed.
- Friend relation is storing a user and his/her friends. Note that if user1-user2 pair is given in the 'csv' file, user2-user1 will also appear.

3 Tasks

3.1 Task 1 - Creating the Database - 15 pts

Using the given 'csv' files and considering the database schema above create a database using PostgreSQL. For this task you should create a file named 'task1.sql' that contains SQL statements that you've used to create the database.

3.2 Task 2 - Advanced SQL Queries - 70 pts

For this task you should be able to write SQL queries for given problems. Please order your queries (from query1 to query14) and create a file named 'task2.sql'. Each question is 5 pts.

1. Find the users whose review_count is higher than its fans and reviewed at least a business which has more than 3.5 stars. List user_id, user_name, difference between review_count and fans (Ordered by difference between review_count and fans, user_id DESC) (1504630 rows)
2. Find the users who have tipped a currently open business located in 'TX' and get compliments more than 2. List all the user-business pairs (this means if there is two businesses that a user get more than 2 compliments list both tips for that user) with user_name, business_name, tip_date, compliment_count (Ordered by compliment_count, tip_date DESC) (38 rows)
3. Find the top 20 users by their friend_count (more friends are better). List user_names and count of friends (Ordered by friend_count, user_name DESC) (20 rows)
4. Find the users who have given lower stars to a business than its current stars. List distinct users' user_name (not distinc user_name distinct user), average_stars and yelping_since (Ordered by average_stars, yelping_since DESC) (1139971 rows)
5. List open "good" businesses who received the highest number of "good" tips in 2020. "Good" business is the one which has word 'good' in its tip text. List the business_name, state and the stars (Ordered by stars, business_name DESC). (4 rows)

6. Find the users who have lower average_stars than **all** of his/her friends' average_stars(Consider the average_stars of Users table.). List user_name, yelping_since and average_stars of such users.(Ordered by average_stars, yelping_since DESC) (105952 rows)

7. The average stars of businesses in a state gives us Average of the State. Find top 10 state by highest average stars. List state code and average stars (Ordered by average stars DESC) (10 rows).

8. A tip is a GOOD tip if its compliment_count is higher than 0 (at least 1 compliment needed). For each year calculate the percentages of GOOD tips if the percentage is higher than 1 percent, this year is called a TIP YEAR. Find all the TIP YEARs. List date of the year and average compliment_count (Ordered by year ASC) (5 rows).

9. List the names of the user's who **only** reviewed businesses who have stars more than 3.5. (Ordered by user_name ASC) (832155 rows)

10. Popular businesses are those with more than 1000 reviews. For each such business find the business_name and average stars for each year. List only those with average stars greater than 3 (in ascending order of years, business_name) (5847 rows).

11. Find the users who have got more useful votes than cool votes (Note that Users table contains votes sent by the users, now we need to find votes s/he got by reviews). List user_name, useful, cool and difference of useful and cool. (Order by difference, user_name DESC) (1003635 rows)

12. List pairs of friends and business names if both friends reviewed the same business with the same stars. List the business id, friends as user1 user2 and stars.(Ordered by business_id, stars DESC). (348516 rows)

13. **Cross tabulations** Write a single SQL query that computes the statistics present in a cross tabulation over stars and state on Business table. The aggregate reported in the crosstab should simply count businesses, but restrict your attention to the open businesses. (147 rows)

14. **Window functions** List the top 3 users and their rank (i.e. 1, 2, or 3) when ranked by review_count for each grouping formed by number_of_fans. Restrict your attention to those with a number_of_fans between 50 and 60, inclusive. (33 rows)

3.2.1 Task 2 Specifications

Your submission file format (for task2) should be as follows:

```
/* Question 1 */
SELECT...
/* Question 2 */
SELECT...
...
/* Question 14 */
SELECT...
```

You don't have to write your queries on single lines. You are allowed (and encouraged) to write them in multiple lines for better readability. You should not write anything for the unsolved questions. You should have lines for only the solved questions.

3.3 Task 3 - Triggers and Views - 15 pts

Triggers. It is often useful to have the DBMS perform some actions automatically in response to operations on the database. Write necessary triggers to achieve the following functionalities:

1. We want to keep review count for each user consistent with the number of reviews they have in Users table. Every time a new user review is inserted into Review table, the review_count in Users table must be incremented. Write a trigger to keep review_count of the users up−to−date in this manner.

2. Our system does not want Users who write reviews with 0 stars. If there is a review with 0 stars that review should not be inserted into the Review table. Further, delete all reviews and tips of that user.

Views. It is inefficient to calculate the number of reviews (review_count) of a business for each related query. Create a view called 'BusinessCount' that contains the columns business_id, business_name, and review_count. The review_count should be the total number of reviews made for that business.

4 Submission

Send a 'tar.gz' file with your 7 digit student id and starting with 'e' like 'e1234567.tar.gz' that contains all 3 '.sql' file completed in previous tasks. You should submit 'tar.gz' file to Odtuclass before the deadline.