MLDS-413 Introduction to Databases and Information Retrieval

Homework 7: Regular expressions; Common Table Expressions; Recursive networks

Name 1: _____

NetID 1: _____

Name 2: _____

NetID 2: _____

Instructions

You should submit this homework assignment via Canvas. Acceptable formats are word files, text files, and pdf files. Paper submissions are not allowed and they will receive an automatic zero.

As explained during lecture and in the syllabus, assignments are done in groups. The groups have been created and assigned. Each group needs to submit only one assignment (i.e., there is no need for both partners to submit individually the same homework assignment).

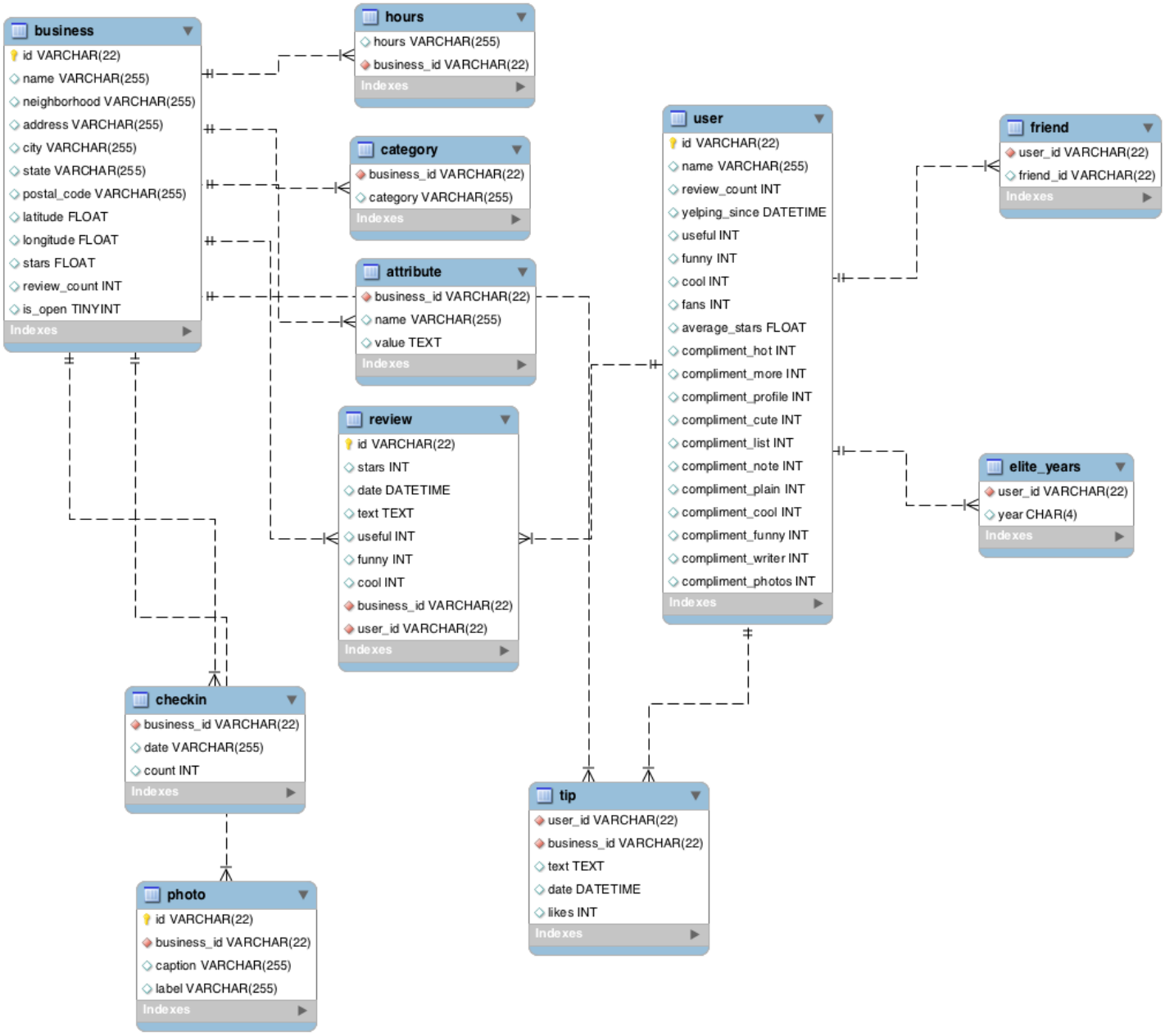
Each group can submit solutions multiple times (for example, you may discover an error in your earlier submission and choose to submit a new solution set). We will grade only the last submission and ignore earlier ones.

Make sure you submit your solutions before the deadline. The policies governing academic integrity, tardiness and penalties are detailed in the syllabus.

Yelp Database (yelp)

The database "yelp" has data from the Yelp business review app (http://yelp.com/). Please follow the instrutions from Homework 6 to connect to the yelp database on MLDS' Postgres server.

The database schema is provided below:



Note that the position of the linking lines does not directly indicate which columns are linked; there is no such requirement or standard for ER diagrams. You will need to infer which columns are the ones linking the tables.

You will use this database to answer the following questions. Unless otherwise noted, for each question please provide:

- The query you constructed
- The output of that query
- Any other information requested by the question (e.g., timing results)

1) **(10 points)** Find the name of the businesses for which there is a review that contains the case-**insensitive** text string "wing" at least 25 times in the same review. *Hint 1:* You do not have to search for complete words but only for **text strings** that are case-insensitive, i.e., "sunwing", "wing", "winging", "Wings", "WiNg" are all hits. *Hint 2*: The regular expressions format in PostgreSQL is different than the MySQL variant we discussed in class. PostgreSQL does pattern matching with regular expressions using the `SIMILAR TO` operator, instead of the `REGEXP` operator. In the `SIMILAR TO` operator "_" matches any character and "`%`" matches any sequence of zero or more characters. The remaining rules are similar to the ones we learned in class, e.g., parentheses "`( )`" are used to group items together into a single logical item, square brackets "`[ ]`" are used to denote a class of characters, angled brackets "`{ }`" are used to denote repetition, etc. The regular expressions syntax rules for PostgreSQL 10 can be found at Section 9.7.2 at https://www.postgresql.org/docs/10/functions-matching.html.

2) **(10 points)** What is the name, address (including city, state, postal code), and **average** rating of the highest-rated **restaurant** with "McDonald" in its name? *Hint 1*: You must use the category named "`Restaurants`", otherwise you'll get results for other types of businesses with "`McDonald`" in the name. *Hint 2*: We are asking for the restaurant with the highest ***average*** rating. Many such restaurants have at least one 5-star rating, but only one location has a star rating **average** close to 5. *Hint 3*: You do not need to concatenate the address into a single string. It is OK for the address, city, state and postal code to occupy a separate column each in your result table.

3) **(10 points)** What are the names of the businesses for which there are at least 5 reviews where each one of these reviews contains the text "`barf`"? *Hint:* Similarly to question 2, you do not have to match individual words, but only sub-strings. For example, "barf", "barfing" and "barfday" should all be considered hits.

4) **(10 points)** With execution timing on, find the name of the user with id `'CxDOIDnH8gp9KXzpBHJYXw'`. Include the time it took to execute the query in your answer. *Note 1:* you may want to run this ~10 times and get the average timing across all runs to get a more reliable measurement.

5) **(10 points)** With execution timing on, find the name of the user with 3336 compliment_plain compliments. Include the time it took to execute the query in your answer. *Note:* you may want to run this ~10 times and get the average timing across all runs to get a more reliable measurement.

6) **(10 points)** Which query is faster, query 5 or query 6, and by how much, and why is it faster? *Note*: this question does not ask you to write a query or provide a query's output. Simply provide your answers below.

7) **(10 points)** Find the absolute number and percentage of businesses that have photos in the database, and businesses without any photo. *Hint*: to obtain a floating-point result in SQL arithmetic operations, at least one of the arithmetic operands must be a floating point number.

8) **(10 points)** Some businesses are open fewer days of the week than others. Use a common table expression to find airports that are open only once a week and report their business id, name, and hours of operation.

9) **(20 points)** You are tasked with doing some city planning, which requires that you find clusters of businesses that are physically located very close to each other. Your first task is to find the IDs, names and GPS coordinates (latitude, longitude) of businesses that are clustered around McDonald's at address Av. Maip 2779. A business is considered part of the cluster if it is within 0.005 degrees away from any other business in the cluster. *Hint 1:* When you need to include an apostrophe as part of a text string in PostgreSQL, you need to escape it with another apostrophe, e.g., to find all "McDonald's" you need a query like `SELECT * FROM business WHERE name='McDonald''s';` Note the use of two apostrophes between letters d and s. *Hint 2:* You can use the Pythagorean theorem to find businesses within the requested range like in question 3. *Hint 3:* You need recursion!