# MSiA-413 Introduction to Databases and Information Retrieval

Homework 7: PostgreSQL on large real-world databases

Name 1: Sam Swain

NetID 1: sms5736

Name 2: Yaasir Ahmed

NetID 2: yap80676

# 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).

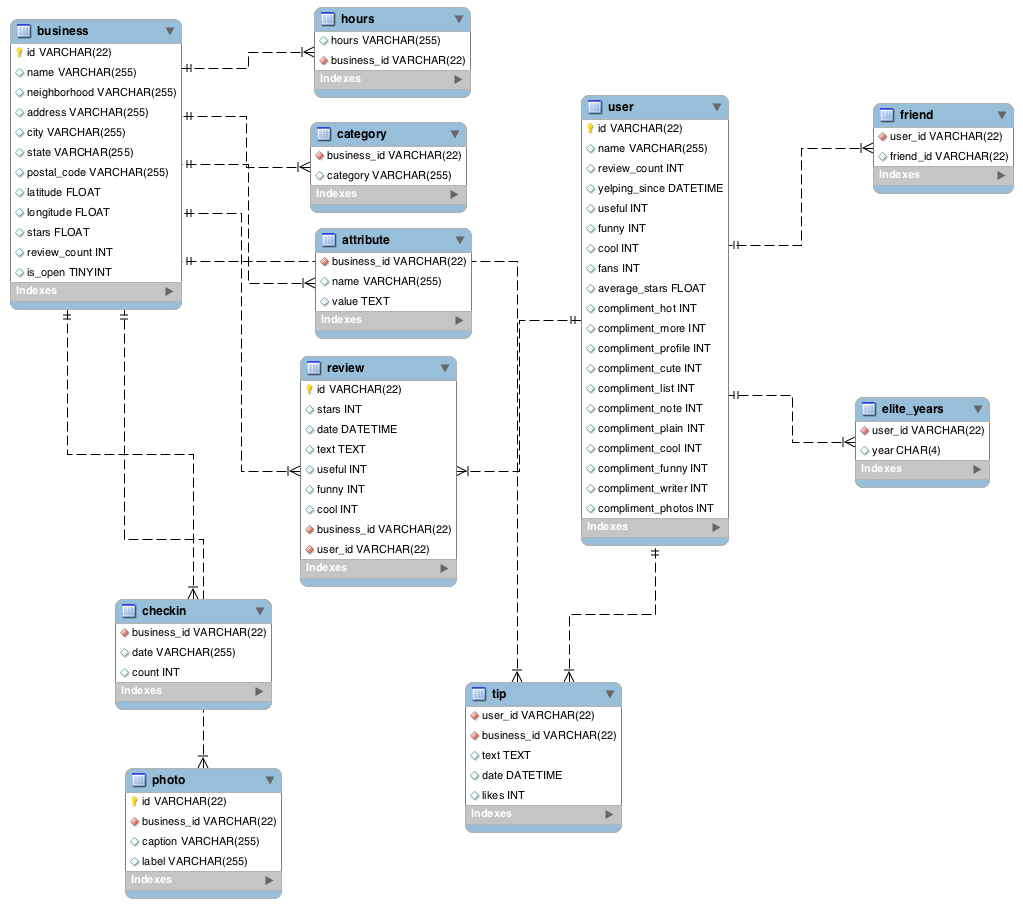
Each group can submit solutions multiple times (for example, you may discover an error in your earlier submission and choose to submit a new solution set). We will grade only the last submission and ignore earlier ones.

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 MSiA’s 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>.

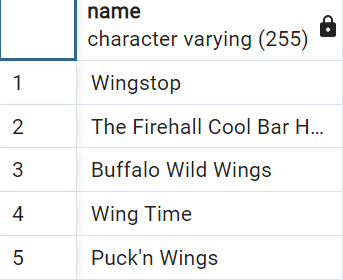
**select b.name**

**from review r**

**join business b on r.business\_id=b.id**

**where lower(text) similar to '(%wing%){25,}';**

**Output:**



1. **(10 points)** Find the top 3 users that have provided the largest number of reviews of businesses within a range of 0.1 degrees from latitude 36.0 and longitude -115.0. For each one of these users, provide in your answer the name and the number of reviews that user provided for businesses within that range. *Hint:* You can use the Pythagorean theorem to find businesses within the requested range. For example, locations within d degrees from latitude X and longitude Y satisfy the formula **sqrt(power(longitude-Y, 2.0) + power(latitude-X, 2.0)) ≤ d**.

**with**

**top\_rev\_ids as**

**(**

**select**

**r.user\_id,**

**count(r.user\_id) num\_revs**

**from review r**

**join business b on**

**r.business\_id = b.id**

**where**

**sqrt(power(longitude + 115.0, 2.0) + power(latitude - 36.0, 2.0)) <= 0.1**

**group by r.user\_id**

**order by num\_revs desc**

**limit 3**

**)**

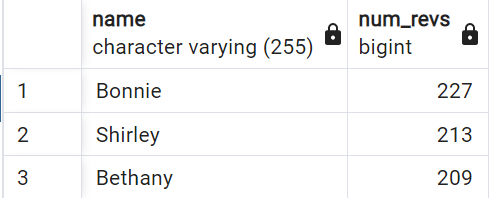
**select name, num\_revs**

**from top\_rev\_ids t**

**join public.user u on**

**t.user\_id=u.id**

**Output:**



1. **(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.

**with**

**business\_star\_id as**

**(**

**select c.business\_id, round(avg(r.stars), 2) avg\_stars**

**from category c**

**join review r on c.business\_id = r.business\_id**

**join business b on c.business\_id = b.id**

**where**

**c.category = 'Restaurants'**

**and b.name similar to '(%McDonald%)'**

**group by c.business\_id**

**order by avg\_stars DESC**

**limit 1**

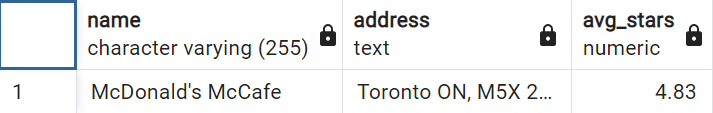
**)**

**select name, city ||' '|| state ||', '|| postal\_code address, avg\_stars**

**from business\_star\_id bs**

**join business b on bs.business\_id=b.id**

**Output:**



1. **(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.

**with**

**gt\_5\_barfs as**

**(**

**select business\_id**

**from review r**

**where text similar to '(%barf%)'**

**group by business\_id**

**having count(text) >= 5**

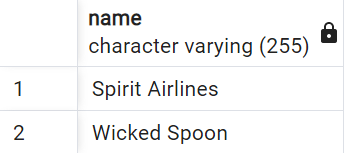
**)**

**select name**

**from business b**

**join gt\_5\_barfs g on g.business\_id = b.id**

**Output:**



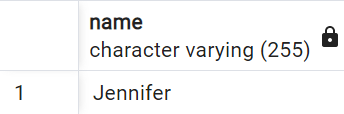
1. **(10 points)** With execution timing on, find the name of the user with id 'CxDOIDnH8gp9KXzpBHJYXw'. Include the time it took to execute the query in your answer. *Note 1:* you may want to run this ~10 times and get the average timing across all runs to get a more reliable measurement.

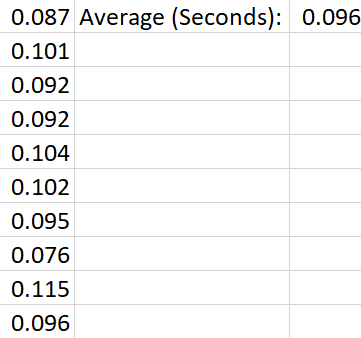
**select name**

**from public.user**

**where id = 'CxDOIDnH8gp9KXzpBHJYXw'**

**Output:**





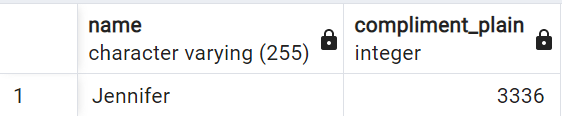
1. **(10 points)** With execution timing on, find the name of the user with 3336 compliment\_plain compliments. Include the time it took to execute the query in your answer. *Note:* you may want to run this ~10 times and get the average timing across all runs to get a more reliable measurement.

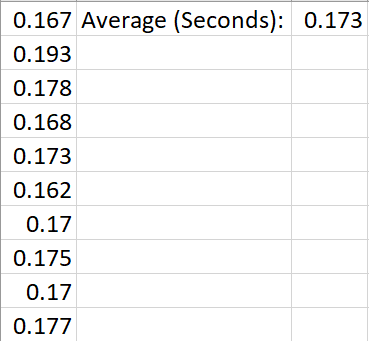
**select name, compliment\_plain**

**from public.user**

**where compliment\_plain = 3336**

**Output:**





1. **(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.

**Query 5 is faster. The primary key of the table is what we were looking for in query 5. The table is indexed by the ID column and that’s why it is so much faster.**

1. **(10 points)** Find the name of the user that has given the largest number of useful reviews to closed businesses. Print both the user name and the number of such reviews the user has given.

**select u.name, count(r.useful) num\_useful**

**from public.user u**

**join review r on u.id = r.user\_id**

**join business b on r.business\_id = b.id**

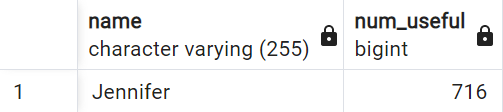
**where b.is\_open = 0 and r.useful > 0**

**group by u.id**

**order by count(r.useful) desc**

**limit 1;**

**Output:**



1. **(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!

**with recursive business\_close(b\_id,**

**b\_name,**

**b\_latitude,**

**b\_longitude) as**

**(**

**select**

**b.id,**

**b.name,**

**b.latitude,**

**b.longitude**

**from**

**business b**

**where**

**b.address = 'Av. Maip 2779'**

**union**

**select**

**b\_temp.id,**

**b\_temp.name,**

**b\_temp.latitude,**

**b\_temp.longitude**

**from**

**business\_close bc, business b\_temp**

**where**

**sqrt(power(bc.b\_longitude - b\_temp.longitude, 2.0) + power(bc.b\_latitude - b\_temp.latitude, 2.0)) <= 0.005**

**)**

**select \***

**from business\_close**

**Output:**

