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).

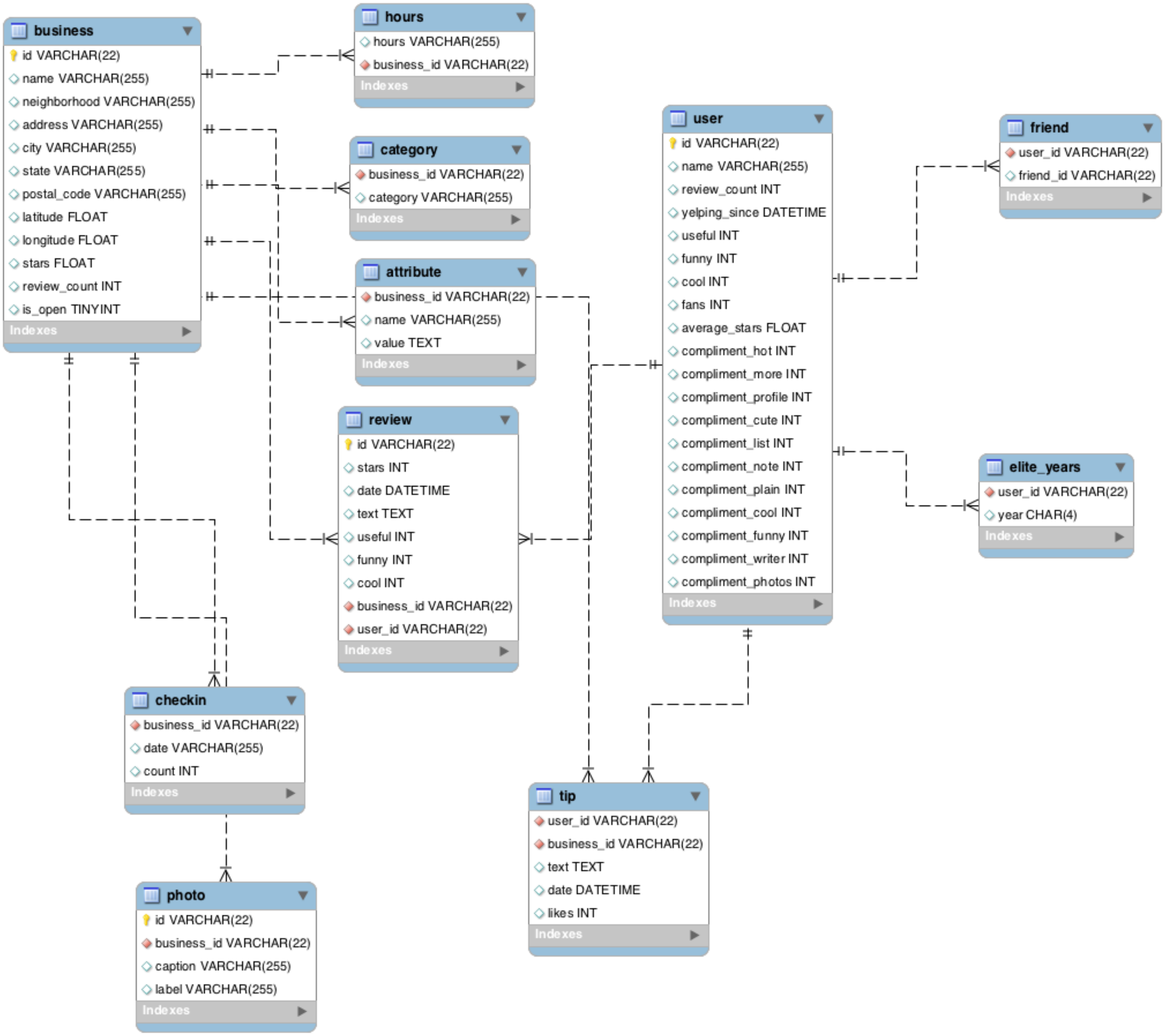
Each group can submit solutions multiple times (for example, you may discover an error in your earlier submission and choose to submit a new solution set). We will grade only the last submission and ignore earlier ones.

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.

```
SELECT business.name
FROM    business
   JOIN review
     ON business.id = review.business_id
WHERE review.text SIMILAR TO '(%[wW][iI][nN][gG]){25,}%';
```

Output:

```
Puck'n Wings
Buffalo Wild Wings
The Firehall Cool Bar Hot Grill
Wing Time
Wingstop
```

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.

Solution 1:
Using the stars rating from the business table:

```
SELECT name, address, city, state, postal_code, stars
FROM    business
   JOIN category
     ON category.business_id = business.id
WHERE category = 'Restaurants'
      AND name LIKE '%McDonald%'
ORDER BY stars DESC LIMIT 1;
```

Output of solution 1:

```
McDonald's McCafe, 100 King Street W, Exchange Tower, Toronto, ON, M5X 2A2, 5
```

Solution 2:
Calculating the true average star rating from the reviews directly:

```
SELECT name, address, city, state, postal_code, AVG(review.stars) AS AvgStars
FROM    business AS B
   JOIN review
     ON B.id = review.business_id
   JOIN category
     ON review.business_id = category.business_id
WHERE category = 'Restaurants'
      AND name LIKE '%McDonald%'
GROUP BY review.business_id, B.name, B.address, B.city, B.state, B.postal_code
ORDER BY AvgStars DESC LIMIT 1;
```

Output of solution 2:

```
McDonald's McCafe, 100 King Street W, Exchange Tower, Toronto, ON, M5X 2A2, 4.8333
```

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.

Full credit solution 1:
Grouping by individual businesses (e.g., individual McDonald's, not the entire chain)

```
SELECT business.name
FROM    business
   JOIN review ON business.id = review.business_id
WHERE review.text LIKE '%barf%'
GROUP BY business.id, business.name
HAVING COUNT(distinct review.id) >= 5;
```

Output of solution 1:

```
Spirit Airlines
Wicked Spoon
```

Full credit solution 2:
Grouping by business name (i.e., aggregating all the stores of a chain together). This grouping for this dataset returns the same answer as the query above, but it may have not been the case – it just so happens that these businesses are not franchised.

```
SELECT business.name
FROM    business
   JOIN review ON business.id = review.business_id
WHERE review.text LIKE '%barf%'
GROUP BY business.name
HAVING COUNT(distinct review.id) >= 5;
```

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.

```
SELECT U.name FROM public.user AS U WHERE U.id='CxDOIDnH8gp9KXzpBHJYXw';
```

Output:

```
Jennifer
```

```
Time: 0.457 ms
```

Note that I use the command line interface. If you use the graphical user interface (pgAdmin) the query may appear ~100x slower than it actually is due to the latency of the graphical interface itself.

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.

```
SELECT U.name FROM public.user AS U WHERE U.compliment_plain=3336;
```

Output:

```
Jennifer
```

```
Time: 85.381 ms
```

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.

Query 5 is 187x faster than query 6.

Query 5 is a point query on id, i.e., it searches for a unique user with a particular id. The table is indexed on the id (as the command "\d public.user" indicates), and thus the search is fast. On the other hand, query 6 is a point query on compliment_plain which has no index, and hence it is much slower.

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.

```
WITH photo_stats(type, num_businesses) AS
   (SELECT 'Businesses with photos', COUNT(DISTINCT business_id)
   FROM photo
   UNION
   SELECT 'Businesses without photos', count(DISTINCT business.id)
   FROM business LEFT JOIN photo ON business.id = photo.business_id
   WHERE photo.business_id IS NULL)
SELECT *, num_businesses * 100.0 / (SELECT COUNT(*) FROM business)
FROM photo_stats;
```

Output:

```
"Businesses without photos"     "128789"    "82.220264429675878932"
"Businesses with photos"  "27850"     "17.779735570324121068"
```

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.

```
WITH lazy_airport(id, name, days_count) AS
   (SELECT id, name, COUNT(*) as days_count
   FROM hours
         JOIN business ON hours.business_id = business.id
         JOIN category ON category.business_id = business.id
   WHERE category LIKE '%Airport%'
   GROUP BY id, name
   HAVING COUNT(*) <= 1)
SELECT id, name, hours
FROM hours JOIN lazy_airport ON hours.business_id = lazy_airport.id;
```

Output:

```
"6VaeaNoma3zRLsIDrF1Cjg"  "Howard   Johnson   Phoenix   Airport/   Downtown   Area"
"Monday|6:00-6:00"
```

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!

```
WITH RECURSIVE business_cluster(id, name, latitude, longitude) AS
(
  SELECT id, name, latitude, longitude
  FROM business
  WHERE name='McDonald''s'
```

```
        AND address='Av. Maip 2779'
  UNION
  SELECT business.id, business.name, business.latitude, business.longitude
  FROM business, business_cluster
  WHERE sqrt(  power(business.longitude - business_cluster.longitude, 2.0)
             + power(business.latitude  - business_cluster.latitude,  2.0))
            <= 0.005
)
SELECT * from business_cluster;
```

Output:

```
           id           |          name              | latitude | longitude
------------------------+----------------------------+----------+-----------
 softZjpREG65wpAns2FaWA | McDonald's                 |   -34.51 |  -58.4911
 bGxzQDGOTpab_6hdqsqv9g | Burger King                | -34.5089 |  -58.4919
 i1e8KsIy1ELvI7G6mvvZkw | Havanna                    | -34.5133 |  -58.4894
 m-SUr48X9gMHtwvraM-KmA | Compaa del Sol             | -34.5134 |  -58.4896
 WNsimvxr-0NimM57I5gj4A | Arnaldo                    | -34.5137 |  -58.4888
 yadScsa2pShYsQAVXbNivw | La Farola de Olivos        | -34.5108 |  -58.4908
 YBaWP2r64BPJazkmyf1fig | Almacn de Pizzas           | -34.5089 |  -58.4916
 zMAiU0s8ScUYHwAESCB8Qg | Prosciutto                 | -34.5122 |  -58.4898
 4-xLjGavuWFqEfNuznxL3A | D' Lucky                   |  -34.516 |  -58.4884
 AwpX8mheEmMhaIuIqEhMkA | Estacin Mitre - Lnea Mitre |  -34.515 |  -58.4897
 Ss6J7HFhMCxoq7M8wXqc8A | Salve Bruna                | -34.5159 |   -58.488
```