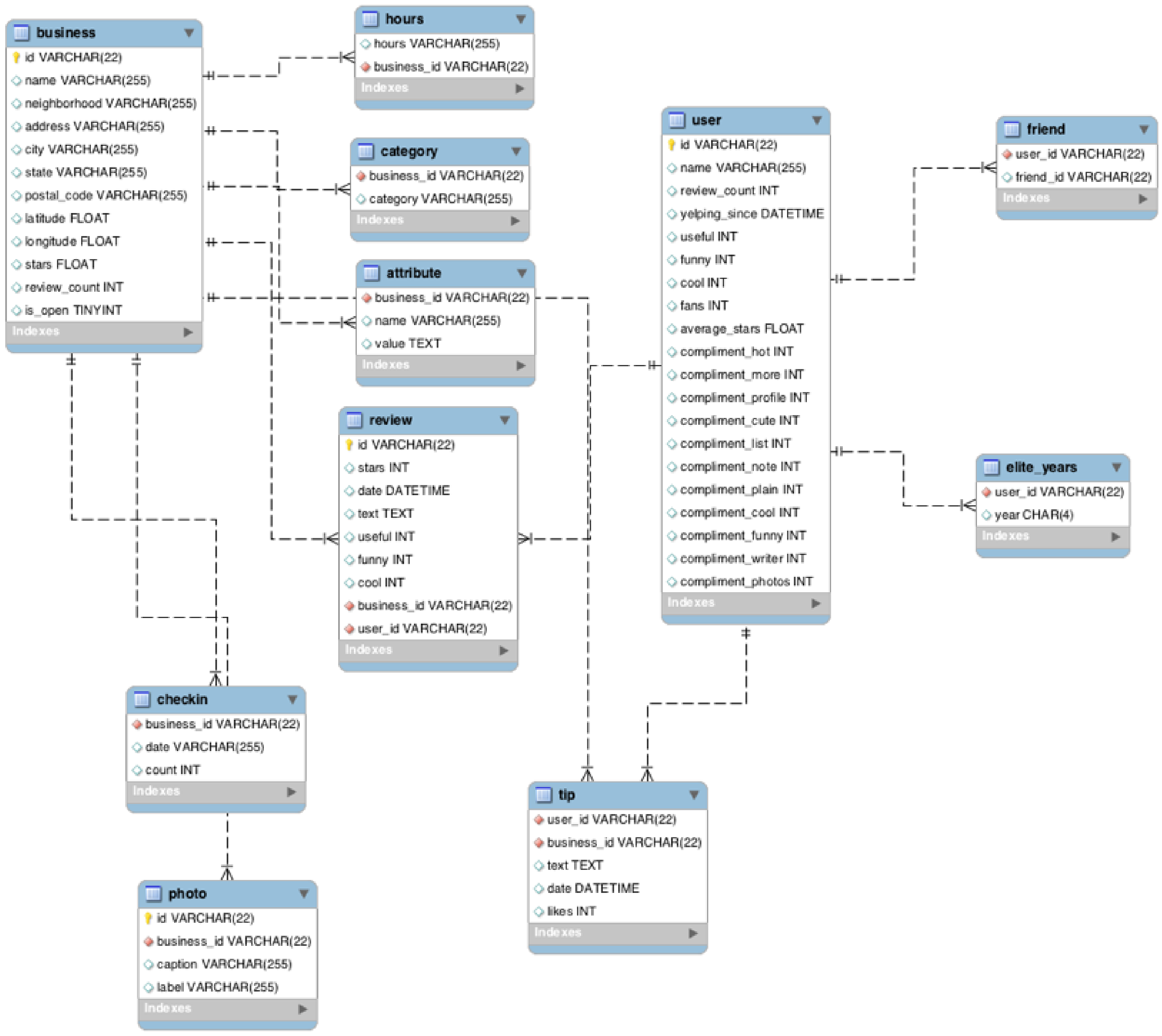
Data Scientist Role Play: Profiling and Analyzing the Yelp Dataset Coursera Worksheet

The YELP Dataset



Part 1: Yelp Dataset Profiling and Understanding

Task 1.1: Profile the data by finding the total number of records for each of the tables below:

i. Attribute table = 10000

```
SELECT COUNT (*)
FROM attribute

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

ii. Business table = 10000

```
SELECT COUNT (*)
FROM business

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

iii. Category table = 10000

```
SELECT COUNT (*)
FROM category

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

iv. Checkin table = 10000

```
SELECT COUNT (*)
FROM checkin

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

v. elite_years table = 10000

```
SELECT COUNT (*)
FROM elite_years

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

vi. friend table = 10000

```
SELECT COUNT (*)
FROM friend

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

vii. hours table = 10000

```
SELECT COUNT (*)
FROM hours

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

viii. photo table = 10000

```
SELECT COUNT (*)
FROM photo

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

ix. review table = 10000

```
SELECT COUNT (*)
FROM review

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

x. tip table = 10000

```
SELECT COUNT (*)
FROM tip

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

xi. user table = 10000

```
SELECT COUNT (*)
FROM user

/* output:
+-----------+
| COUNT (*) |
+-----------+
|     10000 |
+-----------+*/
```

Task 1.2: Find the total distinct records by either the foreign key or primary key for each table. If two foreign keys are listed in the table, please specify which foreign key.

i. Business = 10000 (primary key = id)

```
SELECT COUNT (DISTINCT id)
FROM business

/* output:
+---------------------+
| COUNT (DISTINCT id) |
+---------------------+
|               10000 |
+---------------------+*/
```

ii. Hours = 1562 (foreign key = business_id)

```
SELECT COUNT (DISTINCT business_id)
FROM hours
/* output:
+------------------------------+
| COUNT (DISTINCT business_id) |
+------------------------------+
|                         1562 |
+------------------------------+ */
```

iii. Category = 2643 (foreign key = business_id)

```
SELECT COUNT (DISTINCT business_id)
FROM category
/* output:
+-------------------------------+
| COUNT (DISTINCT business_id) |
+-------------------------------+
|                          2643 |
+-------------------------------+ */
```

iv. Attribute = 1115 (foreign key = business_id)

```
SELECT COUNT (DISTINCT business_id)
FROM attribute
/* output:
+-------------------------------+
| COUNT (DISTINCT business_id) |
+-------------------------------+
|                          1115 |
+-------------------------------+ */
```

v. Review = 10000 (primary key = id)

8090 (foreign key = business_id)

9581 (foreign key = user_id)

```
SELECT COUNT (DISTINCT id)
FROM review
/* output:
+----------------------+
| COUNT (DISTINCT id) |
+----------------------+
|               10000 |
+----------------------+ */

SELECT COUNT (DISTINCT business_id)
FROM review
/* output:
+-------------------------------+
| COUNT (DISTINCT business_id) |
+-------------------------------+
|                         8090 |
+-------------------------------+ */

SELECT COUNT (DISTINCT user_id)
FROM review
/* output:
+---------------------------+
| COUNT (DISTINCT user_id) |
+---------------------------+
|                     9581 |
+---------------------------+ */
```

vi. Checkin = 493 (foreign key = business_id)

```
SELECT COUNT (DISTINCT business_id)
FROM checkin
/* output:
+-------------------------------+
| COUNT (DISTINCT business_id) |
+-------------------------------+
|                          493 |
+-------------------------------+ */
```

vii. Photo = 10000 (primary key = id)

6493 (foreign key = business_id)

```
SELECT COUNT (DISTINCT id)
FROM photo
/* output:
+---------------------+
| COUNT (DISTINCT id) |
+---------------------+
|               10000 |
+---------------------+ */

SELECT COUNT (DISTINCT business_id)
FROM photo

/* output:
+------------------------------+
| COUNT (DISTINCT business_id) |
+------------------------------+
|                         6493 |
+------------------------------+ */
```

viii. Tip = 537 (foreign key = user_id)
3979 (foreign key = business_id)

```
SELECT COUNT (DISTINCT user_id)
FROM tip
/* output:
+--------------------------+
| COUNT (DISTINCT user_id) |
+--------------------------+
|                      537 |
+--------------------------+ */

SELECT COUNT (DISTINCT business_id)
FROM tip
/* output:
+------------------------------+
| COUNT (DISTINCT business_id) |
+------------------------------+
|                         3979 |
+------------------------------+ */
```

ix. User = 10000 (primary key = id)

```
SELECT COUNT (DISTINCT id)
FROM user
/* output:
+---------------------+
| COUNT (DISTINCT id) |
+---------------------+
|               10000 |
+---------------------+ */
```

x. Friend = 11 (foreign key = user_id)

```
SELECT COUNT (DISTINCT user_id)
FROM friend
/* output:
+--------------------------+
| COUNT (DISTINCT user_id) |
+--------------------------+
|                       11 |
+--------------------------+ */
```

xi. Elite_years = 2780 (foreign key = user_id)

```
SELECT COUNT (DISTINCT user_id)
FROM elite_years
/* output:
+--------------------------+
| COUNT (DISTINCT user_id) |
+--------------------------+
|                     2780 |
+--------------------------+ */
```

Note: Primary Keys are denoted in the ER-Diagram with a yellow key icon.

Task 1.3: Are there any columns with null values in the Users table? Indicate "yes," or "no."

Answer: No

SQL code used to arrive at answer:

```
SELECT COUNT (*)
FROM user
WHERE
(name ISNULL)
OR (review_count ISNULL)
OR (yelping_since ISNULL)
OR (useful ISNULL)
OR (funny ISNULL)
OR (cool ISNULL)
OR (fans ISNULL)
OR (average_stars ISNULL)
OR (compliment_hot ISNULL)
OR (compliment_more ISNULL)
OR (compliment_profile ISNULL)
OR (compliment_cute ISNULL)
OR (compliment_list ISNULL)
OR (compliment_note ISNULL)
OR (compliment_plain ISNULL)
OR (compliment_cool ISNULL)
OR (compliment_funny ISNULL)
OR (compliment_writer ISNULL)
OR (compliment_photos ISNULL)

/* output:
+-----------+
| COUNT (*) |
+-----------+
|         0 |
+-----------+ */
```

Task 1.4: For each table and column listed below, display the smallest (minimum), largest (maximum), and average (mean) value for the following fields:

i. Table: Review, Column: Stars

```
        min: 1.0                max: 5.0                avg: 3.7082
```

```
SELECT
MIN(stars),
MAX(stars),
AVG(stars)
FROM review
```

ii. Table: Business, Column: Stars

```
        min: 1.0                max: 5.0                avg: 3.6549
```

```
SELECT
MIN(stars),
MAX(stars),
AVG(stars)
FROM business
```

iii. Table: Tip, Column: Likes

```
        min: 0          max: 2          avg: 0.0144
```

```
SELECT
MIN(likes),
MAX(likes),
AVG(likes)
FROM tip
```

iv. Table: Checkin, Column: Count

```
        min: 1          max: 53         avg: 1.9414
```

```
SELECT
MIN(count),
MAX(count),
AVG(count)
FROM checkin
```

v. Table: User, Column: Review_count

```
        min: 0          max: 2000               avg: 24.2995
```

```
SELECT
MIN(review_count),
MAX(review_count),
AVG(review_count)
FROM user
```

Task 1.5: List the cities with the most reviews in descending order:

SQL code used to arrive at answer:

```
SELECT COUNT(r.id) AS total_reviews, b.city
FROM review r INNER JOIN business b
ON r.business_id = b.id
GROUP BY b.city
ORDER BY total_reviews DESC
```

Copy and Paste the Result Below:

```
/*
+---------------+-----------------+
| total_reviews | city            |
+---------------+-----------------+
|           193 | Las Vegas       |
|            65 | Phoenix         |
|            51 | Toronto         |
|            37 | Scottsdale      |
|            30 | Henderson       |
|            28 | Tempe           |
|            23 | Pittsburgh      |
|            22 | Chandler        |
|            21 | Charlotte       |
|            18 | Montréal        |
|            16 | Madison         |
|            13 | Gilbert         |
|            13 | Mesa            |
|            12 | Cleveland       |
|             6 | North Las Vegas |
|             5 | Edinburgh       |
|             5 | Glendale        |
|             5 | Lakewood        |
|             4 | Cave Creek      |
|             4 | Champaign       |
|             4 | Markham         |
|             4 | North York      |
|             3 | Mississauga     |
|             3 | Surprise        |
|             2 | Avondale        |
+---------------+-----------------+ */
```

Task 1.6: Find the distribution of star ratings to the business in the following cities:

i. Avon

SQL code used to arrive at answer:

```
SELECT
r.stars,
COUNT(r.id) AS total_reviews
FROM review r INNER JOIN business b
ON r.business_id = b.id
WHERE b.city = 'Avon'
GROUP BY r.stars
```

Copy and Paste the Resulting Table Below (2 columns â€“ star rating and count):

```
/*
+-------+---------------+
| stars | total_reviews |
+-------+---------------+
+-------+---------------+
(Zero rows) */
```

ii. Beachwood

SQL code used to arrive at answer:

```
SELECT
r.stars,
COUNT(r.id) AS total_reviews
FROM review r INNER JOIN business b
ON r.business_id = b.id
WHERE b.city = 'Beachwood'
GROUP BY r.stars
```

Copy and Paste the Resulting Table Below (2 columns â€“ star rating and count):

```
/*
+-------+---------------+
| stars | total_reviews |
+-------+---------------+
|     3 |             1 |
+-------+---------------+ */
```

Task 1.7: Find the top 3 users based on their total number of reviews:

SQL code used to arrive at answer:

```
SELECT name, review_count
FROM user
ORDER BY review_count DESC
LIMIT 3
```

Copy and Paste the Result Below:

```
/*
+--------+--------------+
| name   | review_count |
+--------+--------------+
| Gerald |         2000 |
| Sara   |         1629 |
| Yuri   |         1339 |
+--------+--------------+ */
```

Task 1.8: Does posing more reviews correlate with more fans?:

No, because when we query for users, posts and fans, ordering by number of fans in descending order, we can observe that users with the higher review count are not necessarily the ones with the most fans. The top 3 users with most reviews, that were queried on the last task are not on the list of users with most friends.

```
SELECT name, review_count, fans
FROM user
ORDER BY fans DESC
LIMIT 10

/* output:
+-----------+--------------+------+
| name      | review_count | fans |
+-----------+--------------+------+
| Amy       |          609 |  503 |
| Mimi      |          968 |  497 |
| Harald    |         1153 |  311 |
| Gerald    |         2000 |  253 |
| Christine |          930 |  173 |
| Lisa      |          813 |  159 |
| Cat       |          377 |  133 |
| William   |         1215 |  126 |
| Fran      |          862 |  124 |
| Lissa     |          834 |  120 |
+-----------+--------------+------+ */
```

Task 1.9: Are there more reviews with the word "love" or with the word "hate" in them?:

Answer: More reviews with the word 'love' (1780) than with the word 'hate' ()

SQL code used to arrive at answer:

```
SELECT COUNT(text) as love_count
FROM review
WHERE text LIKE '%love%'

/* output:
+------------+
| love_count |
+------------+
|       1780 |
+------------+ */

SELECT COUNT(text) as hate_count
FROM review
WHERE text LIKE '%hate%'

/* output:
+------------+
| hate_count |
+------------+
|        232 |
+------------+ */
```

Task 1.10: Find the top 10 users with the most fans:

SQL code used to arrive at answer:

```
SELECT name, fans
FROM user
ORDER BY fans DESC
LIMIT 10
```

Copy and Paste the Result Below:

```
/*
+-----------+------+
| name      | fans |
+-----------+------+
| Amy       |  503 |
| Mimi      |  497 |
| Harald    |  311 |
| Gerald    |  253 |
| Christine |  173 |
| Lisa      |  159 |
| Cat       |  133 |
| William   |  126 |
| Fran      |  124 |
| Lissa     |  120 |
+-----------+------+ */
```

Part 2: Inferences and Analysis

Task 2.1 Pick one city and category of your choice and group the businesses in that city or category by their overall star rating. Compare the businesses with 2-3 stars to the businesses with 4-5 stars and answer the following questions. Include your code.

I chose the city of Toronto and the category 'Restaurants'

i. Do the two groups you chose to analyze have a different distribution of hours?

Yes, it seems like the lower-star group is composed by restaurants that are open for more hours than the restaurants in the higher-star group.

```
-- First query for group 1: 2-3 stars
SELECT b.name, h.hours, b.stars
FROM ((business b INNER JOIN hours h ON b.id = h.business_id)
INNER JOIN category c ON b.id = c.business_id)
WHERE (c.category = 'Restaurants' AND b.city = 'Toronto'
AND b.stars<4)
ORDER BY b.stars

/* output:
+------------------+----------------------+-------+
| name             | hours                | stars |
+------------------+----------------------+-------+
| 99 Cent Sushi    | Monday|11:00-23:00   |   2.0 |
| 99 Cent Sushi    | Tuesday|11:00-23:00  |   2.0 |
| 99 Cent Sushi    | Friday|11:00-23:00   |   2.0 |
| 99 Cent Sushi    | Wednesday|11:00-23:00 |   2.0 |
| 99 Cent Sushi    | Thursday|11:00-23:00 |   2.0 |
| 99 Cent Sushi    | Sunday|11:00-23:00   |   2.0 |
| 99 Cent Sushi    | Saturday|11:00-23:00 |   2.0 |
| Big Smoke Burger | Monday|10:30-21:00   |   3.0 |
| Big Smoke Burger | Tuesday|10:30-21:00  |   3.0 |
| Big Smoke Burger | Friday|10:30-21:00   |   3.0 |
| Big Smoke Burger | Wednesday|10:30-21:00 |   3.0 |
| Big Smoke Burger | Thursday|10:30-21:00 |   3.0 |
| Big Smoke Burger | Sunday|11:00-19:00   |   3.0 |
| Big Smoke Burger | Saturday|10:30-21:00 |   3.0 |
| Pizzaiolo        | Monday|9:00-23:00    |   3.0 |
| Pizzaiolo        | Tuesday|9:00-23:00   |   3.0 |
| Pizzaiolo        | Friday|9:00-4:00     |   3.0 |
| Pizzaiolo        | Wednesday|9:00-23:00 |   3.0 |
| Pizzaiolo        | Thursday|9:00-23:00  |   3.0 |
| Pizzaiolo        | Sunday|10:00-23:00   |   3.0 |
| Pizzaiolo        | Saturday|10:00-4:00  |   3.0 |
+------------------+----------------------+-------+ */

-- Second query for group 1: 4-5 stars
SELECT b.name, h.hours, b.stars
FROM ((business b INNER JOIN hours h ON b.id = h.business_id)
INNER JOIN category c ON b.id = c.business_id)
WHERE (c.category = 'Restaurants' AND b.city = 'Toronto'
AND b.stars>=4)
ORDER BY b.stars

/* output:
+-------------+----------------------+-------+
| name        | hours                | stars |
+-------------+----------------------+-------+
| Edulis      | Sunday|12:00-16:00   |   4.0 |
| Edulis      | Friday|18:00-23:00   |   4.0 |
| Edulis      | Wednesday|18:00-23:00 |   4.0 |
| Edulis      | Thursday|18:00-23:00  |   4.0 |
| Edulis      | Saturday|18:00-23:00  |   4.0 |
| Cabin Fever | Monday|16:00-2:00     |   4.5 |
| Cabin Fever | Tuesday|18:00-2:00    |   4.5 |
| Cabin Fever | Friday|18:00-2:00     |   4.5 |
| Cabin Fever | Wednesday|18:00-2:00  |   4.5 |
| Cabin Fever | Thursday|18:00-2:00   |   4.5 |
| Cabin Fever | Sunday|16:00-2:00     |   4.5 |
| Cabin Fever | Saturday|16:00-2:00   |   4.5 |
| Sushi Osaka | Monday|11:00-23:00    |   4.5 |
| Sushi Osaka | Tuesday|11:00-23:00   |   4.5 |
| Sushi Osaka | Friday|11:00-23:00    |   4.5 |
| Sushi Osaka | Wednesday|11:00-23:00 |   4.5 |
| Sushi Osaka | Thursday|11:00-23:00  |   4.5 |
| Sushi Osaka | Sunday|14:00-23:00    |   4.5 |
| Sushi Osaka | Saturday|11:00-23:00  |   4.5 |
+-------------+-----------------------+-------+
*/
```

ii. Do the two groups you chose to analyze have a different number of reviews?

Yes. The average number of reviews for restaurants in the high-star group (~41 reviews/restaurant) is more than 2 times the average number of reviews for low-star restaurants (~19 reviews/restaurant)

```
-- First query for group 1: 2-3 stars
SELECT AVG(b.review_count) AS avg_review_count
FROM business b INNER JOIN category c ON b.id = c.business_id
WHERE (b.city = 'Toronto' AND c.category = 'Restaurants' AND b.stars <4)

/* output:
+------------------+
| avg_review_count |
+------------------+
|             18.6 |
+------------------+ */

-- Second query for group 1: 4-5 stars
SELECT AVG(b.review_count) AS avg_review_count
FROM business b INNER JOIN category c ON b.id = c.business_id
WHERE (b.city = 'Toronto' AND c.category = 'Restaurants' AND b.stars >=4)

/* output:
+------------------+
| avg_review_count |
+------------------+
|             41.2 |
+------------------+ */
```

iii. Are you able to infer anything from the location data provided between these two groups? Explain.

An analysis of neighborhood location for restaurants on both groups was performed, but no consistent results were observed.

```
-- First query for group 1: 2-3 stars
SELECT name, neighborhood
FROM business b INNER JOIN category c ON b.id = c.business_id
WHERE (b.city = 'Toronto' AND c.category = 'Restaurants' AND b.stars <4)

/* output:
+--------------------+------------------------+
| name               | neighborhood           |
+--------------------+------------------------+
| Royal Dumpling     | Willowdale             |
| Big Smoke Burger   | Downtown Core          |
| 99 Cent Sushi      | Downtown Core          |
| Pizzaiolo          | Entertainment District |
| The Kosher Gourmet |                        |
+--------------------+------------------------+ */

-- Second query for group 1: 4-5 stars
SELECT name, neighborhood
FROM business b INNER JOIN category c ON b.id = c.business_id
WHERE (b.city = 'Toronto' AND c.category = 'Restaurants' AND b.stars >=4)

/* output:
+-------------+--------------+
| name        | neighborhood |
+-------------+--------------+
| Mama Mia    |              |
| Cabin Fever | High Park    |
| Sushi Osaka | Etobicoke    |
| Naniwa-Taro | Willowdale   |
| Edulis      | Niagara      |
+-------------+--------------+ */
```

Task 2.2: Group business based on the ones that are open and the ones that are closed. What differences can you find between the ones that are still open and the ones that are closed? List at least two differences and the SQL code you used to arrive at your answer.

i. Difference 1:

Although there is not much difference on the average rating count for businesses that are open and the ones that are closed (rating for open business is slightly higher), there is a significant difference between the average review_count of those groups of businesses. Open businesses have on average more than 35% reviews than closed businesses.

```
SELECT is_open,
AVG(stars) AS average_rating,
AVG(review_count) AS average_review_count
FROM business
WHERE is_open = 0
UNION
SELECT is_open,
AVG(stars) AS average_rating,
AVG(review_count) AS average_review_count
FROM business
WHERE is_open = 1

/* output:
+---------+----------------+----------------------+
| is_open | average_rating | average_review_count |
+---------+----------------+----------------------+
|       0 |  3.52039473684 |        23.1980263158 |
|       1 |  3.67900943396 |        31.7570754717 |
+---------+----------------+----------------------+ */
```

ii. Difference 2:
Although for both open and closed businesses the most frequent category is 'Restaurants', business of categories 'Nightlife' and 'Bars' are among the most frequent types of closed busineses, while 'Food' and 'Medical Health' are among the most frequent types of open businesses.

```
SELECT b.is_open, c.category, COUNT(*) AS total_businesses
FROM business b INNER JOIN category c ON b.id = c.business_id
WHERE b.is_open = 0
GROUP BY c.category
ORDER BY total_businesses DESC
LIMIT 5

/* output:
+---------+----------------+------------------+
| is_open | category       | total_businesses |
+---------+----------------+------------------+
|       0 | Restaurants    |               18 |
|       0 | Nightlife      |                8 |
|       0 | Bars           |                6 |
|       0 | Shopping       |                5 |
|       0 | American (New) |                3 |
+---------+----------------+------------------+ */

SELECT b.is_open, c.category, COUNT(*) AS total_businesses
FROM business b INNER JOIN category c ON b.id = c.business_id
WHERE b.is_open = 1
GROUP BY c.category
ORDER BY total_businesses DESC
LIMIT 5

/* output:
+---------+------------------+------------------+
| is_open | category         | total_businesses |
+---------+------------------+------------------+
|       1 | Restaurants      |               53 |
|       1 | Shopping         |               25 |
|       1 | Food             |               20 |
|       1 | Health & Medical |               16 |
|       1 | Home Services    |               15 |
+---------+------------------+------------------+ */
```

Task 2.3: For this last part of your analysis, you are going to choose the type of analysis you want to conduct on the Yelp dataset and are going to prepare the data for analysis.

Ideas for analysis include: Parsing out keywords and business attributes for sentiment analysis, clustering businesses to find commonalities or anomalies between them, predicting the overall star rating for a business, predicting the number of fans a user will have, and so on. These are just a few examples to

get you started, so feel free to be creative and come up with your own problem you want to solve. Provide answers, in-line, to all of the following:

i. Indicate the type of analysis you chose to do:
Predict the overall number star rating for a business.

ii. Write 1-2 brief paragraphs on the type of data you will need for your analysis and why you chose that data:
For predicting the target features (overall number of stars of a business), I have selected general aspects of the business, such as location data, and also the category type of the business.

In order to create this dataset, I had to perform a INNER JOIN between tables 'business' and 'category', selecting the features that would help me on the prediction. Using this dataset, one may use machine learning techniques such as decision trees or neural networks to perform the prediction.

iii. Output of your finished dataset:

```
/* output:
+-------+--------------------+-------+------------------------+
| stars | city               | state | category               |
+-------+--------------------+-------+------------------------+
|   3.0 | Markham            | ON    | Asian Fusion           |
|   3.0 | Markham            | ON    | Restaurants            |
|   2.0 | Stuttgart-Vaihingen | BW    | Transportation         |
|   2.0 | Stuttgart-Vaihingen | BW    | Public Transportation  |
|   2.0 | Stuttgart-Vaihingen | BW    | Hotels & Travel        |
|   2.0 | Stuttgart-Vaihingen | BW    | Train Stations         |
|   2.0 | Stuttgart-Vaihingen | BW    | Metro Stations         |
|   3.0 | Edinburgh          | EDH   | Restaurants            |
|   3.0 | Edinburgh          | EDH   | Delis                  |
|   3.5 | Charlotte          | NC    | Electronics            |
|   3.5 | Charlotte          | NC    | Shopping               |
|   3.5 | Charlotte          | NC    | Automotive             |
|   3.5 | Charlotte          | NC    | Car Stereo Installation |
|   2.0 | Champaign          | IL    | Restaurants            |
|   2.0 | Champaign          | IL    | Mexican                |
|   4.0 | Mesa               | AZ    | Restaurants            |
|   4.0 | Mesa               | AZ    | Bars                   |
|   4.0 | Mesa               | AZ    | Italian                |
|   4.0 | Mesa               | AZ    | Nightlife              |
|   4.0 | Mesa               | AZ    | Pizza                  |
|   4.0 | Mesa               | AZ    | Salad                  |
|   4.0 | Mesa               | AZ    | Gluten-Free            |
|   3.5 | Sheffield Village  | OH    | American (Traditional) |
|   3.5 | Sheffield Village  | OH    | Restaurants            |
|   3.5 | Sheffield Village  | OH    | Southern               |
+-------+--------------------+-------+------------------------+
*/
```

iv. Provide the SQL code you used to create your final dataset:

```sql
SELECT b.stars, b.city, b.state, c.category
FROM business b INNER JOIN category c ON b.id = c.business_id
```