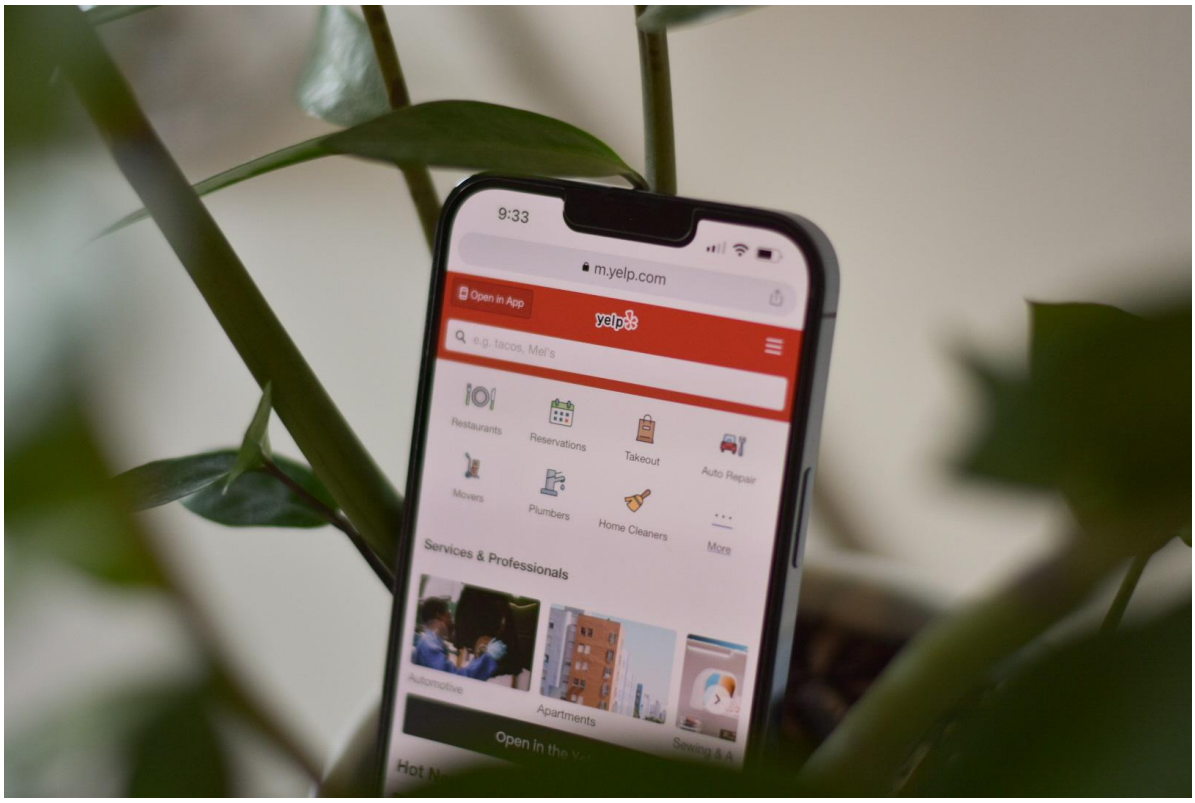
PROFILE AND ANALYZE THE YELP DATASET



Julia Ohorodnyk

12/05/2022

SQL for Data Science University of California, Davis

Part 1. Profiling and Analyzing the Yelp Dataset

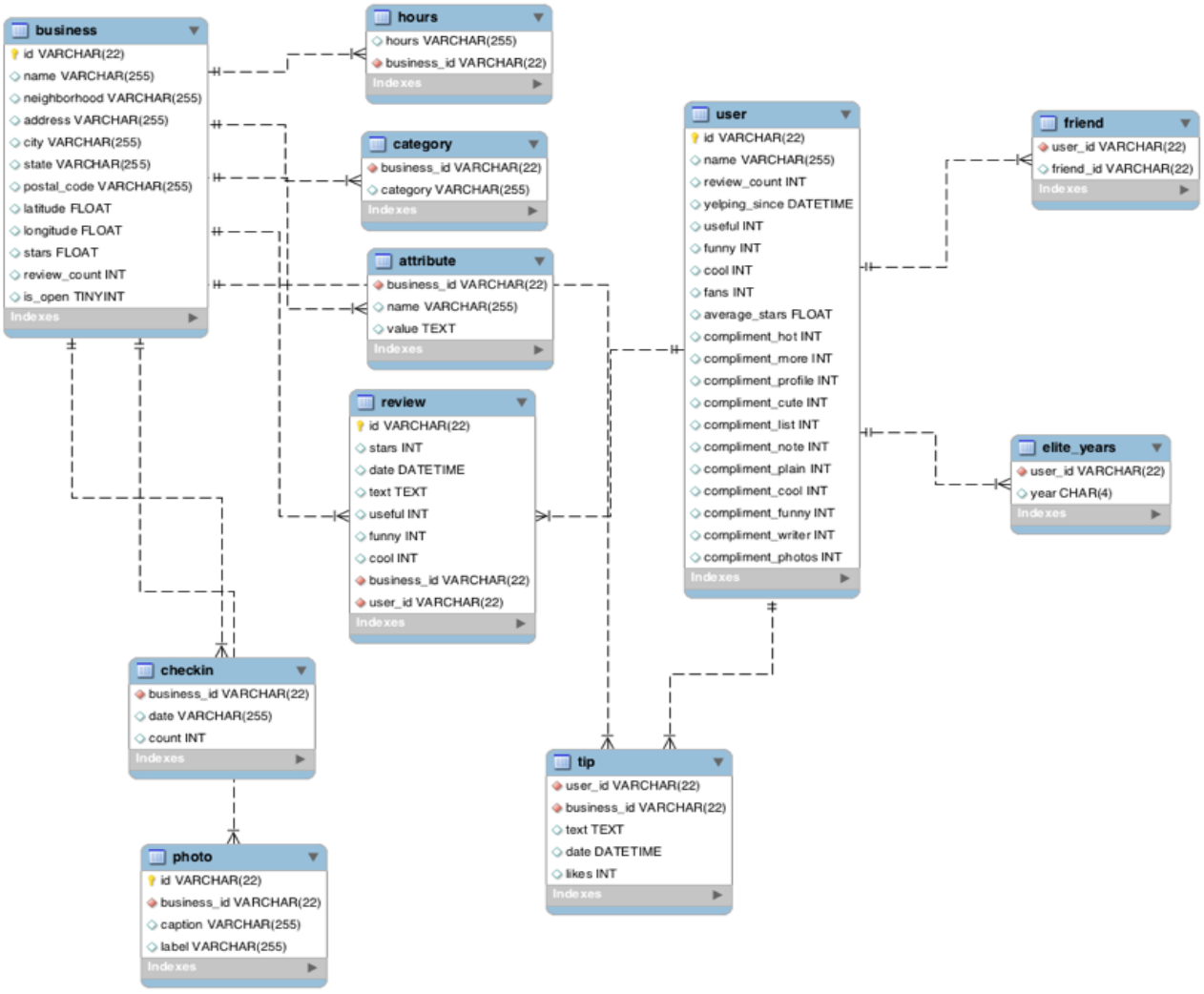
This project is a part of the **SQL for Data Science** course on Coursera from the University of California, Davis. All-access to the data is done through Coursera UI.

Yelp is a platform for users to share reviews and rate their interactions with various organizations – businesses, restaurants, health clubs, hospitals, local governmental offices, charitable organizations, etc. For the analysis, I will work with the Yelp Dataset, provided by the US-based organization Yelp.

First, define primary and secondary keys for each table by observing the ER Diagram of the Yelp Dataset. Then, find the total distinct records by either the foreign or primary keys for each table.

```
SELECT COUNT(DISTINCT (business_id)) AS Total_id
FROM attribute
```

Table	Primary key(PK)	Foreign key(FK)	Total_number
attribute		business_id	1115
business	id		10000
category		business_id	2643
checkin		business_id	493
elite_years		user_id	2780
friend		user_id	11
hours		business_id	1562
photo	id	business_id	PK(id) = 10000, FK(business_id) = 6493
review	id	user_id, business_id	PK(id) = 10000, FK(user_id) = 9681, FK(business_id) = 8090
tip		user_id, business_id	FK(user_id) = 537, FK(busines_id) = 3979
user	id		10000



Entity Relationship Diagram the Yelp Dataset

Primary Keys are denoted with a yellow key icon, and foreign keys with a red diamond.

Profile the data by finding the total number of records for each of the tables

```
SELECT COUNT(*) as total_amount
FROM attribute.
```

After running the same base query for other tables, I found that each has 10000 records.

Table	Total amount per table
attribute	10000
business	
category	
checkin	
elite_years	
friend	
hours	
photo	
review	
tip	
user	

Search the Null values for the User table

```
SELECT COUNT(*)
FROM user
WHERE id IS NULL OR
      name IS NULL OR
      review_count IS NULL OR
      yelping_since IS NULL OR
      useful IS NULL OR
      funny IS NULL OR
      cool IS NULL OR
      fans IS NULL OR
      average_stars IS NULL OR
      compliment_hot IS NULL OR
      compliment_more IS NULL OR
      compliment_profile IS NULL OR
      compliment_cute IS NULL OR
      compliment_list IS NULL OR
      compliment_note IS NULL OR
      compliment_plain IS NULL OR
```

```
        compliment_cool IS NULL OR
        compliment_funny IS NULL OR
        compliment_writer IS NULL OR
        compliment_photos IS NULL
```

There are no columns in the User table with Null values.

Find the smallest, largest, and average values by applying MIN(), MAX(), and AVG() functions.

```
SELECT MIN(stars) as min_stars,
       MAX(stars) as max_stars,
       AVG(stars) as avg_stars
FROM review
```

Table	Column	Min_stars	Max_stars	Avg_stars
review	stars	1	5	3.7082
business	stars	1.0	5.0	3.6549
tip	likes	0	2	0.0144
checkin	count	1	54	1.9414
user	review_count	0	2000	24.2995

List the cities with the most reviews in descending order:

```
SELECT city,
       SUM(review_count) as total_review
FROM business
GROUP By city
ORDER BY total_reviews DESC
```

city	total_review
Las Vegas	82854
Phoenix	34503
Toronto	24113
Scottsdale	20614
Charlotte	12523
Henderson	10871
Tempe	10504
Pittsburgh	9798
Montréal	9448
Chandler	8112
Mesa	6875
Gilbert	6380
Cleveland	5593
Madison	5265
Glendale	4406
Mississauga	3814
Edinburgh	2792
Peoria	2624
North Las Vegas	2438
Markham	2352
Champaign	2029
Stuttgart	1849
Surprise	1520
Lakewood	1465
Goodyear	1155

Find the distribution of star ratings to the business for the Avon city.

```
SELECT stars,
      SUM(review_count) AS total_reviews
FROM business
WHERE city is 'Avon'
GROUP BY stars
```

stars	total_reviews
1.5	10
2.5	6
3.5	88
4.0	21
4.5	31
5.0	3

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

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

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

Look for possible correlation between posting more reviews and having more fans.

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

```
+-----------+--------------+------+
| name      | review_count | fans |
+-----------+--------------+------+
| Gerald    |         2000 |  253 |
| Sara      |         1629 |   50 |
| Yuri      |         1339 |   76 |
| .Hon      |         1246 |  101 |
| William   |         1215 |  126 |
| Harald    |         1153 |  311 |
| eric      |         1116 |   16 |
| Roanna    |         1039 |  104 |
| Mimi      |          968 |  497 |
| Christine |          930 |  173 |
| Ed        |          904 |   38 |
| Nicole    |          864 |   43 |
| Fran      |          862 |  124 |
| Mark      |          861 |  115 |
| Christina |          842 |   85 |
| Dominic   |          836 |   37 |
| Lissa     |          834 |  120 |
| Lisa      |          813 |  159 |
| Alison    |          775 |   61 |
| Sui       |          754 |   78 |
| Tim       |          702 |   35 |
| L         |          696 |   10 |
| Angela    |          694 |  101 |
| Crissy    |          676 |   25 |
| Lyn       |          675 |   45 |
+-----------+--------------+------+
```

Based on the temporary table above, there is no correlation between the amount of posted reviews and amount of fans for each user. For example, the user with the name “Fran” has posted fewer reviews than the user “Sara,” but at the same time, the user “Fran” has more fans.

Search the reviews for the word "love" or the word "hate" in them.

```
SELECT SUM(text LIKE "%love%") as love,
       SUM(text LIKE "%hate%") as hate
FROM review
```

love	hate
1780	232

Display the top 10 users with the most number of fans

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

name	fans
Amy	503
Mimi	497
Harald	311
Gerald	253
Christine	173
Lisa	159
Cat	133
William	126
Fran	124
Lissa	120

The user with the name Amy has the largest number of fans.

Part 2. Inferences and Analysis

For my analysis, in the second part, I picked the city “Toronto” and the category “Restaurants” and the stars to compare businesses with low and high ratings.

Do the two groups in my analysis have a different number of reviews?

```
SELECT b.name,
   b.review_count as reviews,
   CASE
       WHEN stars < 4.0 THEN '0-3'
       WHEN stars >= 4.0 THEN '4-5'
   END AS stars_range
FROM business as b
   INNER JOIN category as c ON b.id = c.business_id
WHERE city = 'Toronto' and category = 'Restaurants'
GROUP BY b.name
ORDER BY reviews
```

```
+--------------------+---------+-------------+
| name               | reviews | stars_range |
+--------------------+---------+-------------+
| The Kosher Gourmet |       3 | 0-3         |
| Royal Dumpling     |       4 | 0-3         |
| 99 Cent Sushi      |       5 | 0-3         |
| Mama Mia           |       8 | 4-5         |
| Sushi Osaka        |       8 | 4-5         |
| Cabin Fever        |      26 | 4-5         |
| Pizzaiolo          |      34 | 0-3         |
| Big Smoke Burger   |      47 | 0-3         |
| Naniwa-Taro        |      75 | 4-5         |
| Edulis             |      89 | 4-5         |
+--------------------+---------+-------------+
```

Total review numbers for restaurants in Toronto

I compared working hours and the number of reviews for restaurants in Toronto with low ratings (stars <4) and high rating(stars >= 4). The total number of reviews for each restaurant has different amounts of reviews. There is no correlation between the number of reviews and star groups. For example, the “Cabin Fever” restaurant has 26 reviews and is assigned to a high rating star group, while “Big Smoke Burger” has a lower star rating but a much higher number of reviews = 47.

Do the two groups in my analysis have a different distribution of hours?

```
SELECT b.name, h.hours, b.stars, b.review_count as reviews,
CASE
   WHEN stars < 4.0 THEN '0-3'
   WHEN stars >= 4.0 THEN '4-5'
END AS stars_range,
CASE
      WHEN hours LIKE "%monday%" THEN 1
      WHEN hours LIKE "%tuesday%" THEN 2
      WHEN hours LIKE "%wednesday%" THEN 3
      WHEN hours LIKE "%thursday%" THEN 4
      WHEN hours LIKE "%friday%" THEN 5
      WHEN hours LIKE "%saturday%" THEN 6
      WHEN hours LIKE "%sunday%" THEN 7
      END AS week_day
FROM business as b
INNER JOIN category as c ON b.id = c.business_id
INNER JOIN hours as h ON b.id = h.business_id
WHERE city = 'Toronto' and category = 'Restaurants'
GROUP BY hours
ORDER BY week_day, stars_range
```

name	hours	stars	reviews	stars_range	week_day
Big Smoke Burger	Monday|10:30-21:00	3.0	47	0-3	1
99 Cent Sushi	Monday|11:00-23:00	2.0	5	0-3	1
Pizzaiolo	Monday|9:00-23:00	3.0	34	0-3	1
Cabin Fever	Monday|16:00-2:00	4.5	26	4-5	1
Big Smoke Burger	Tuesday|10:30-21:00	3.0	47	0-3	2
99 Cent Sushi	Tuesday|11:00-23:00	2.0	5	0-3	2
Pizzaiolo	Tuesday|9:00-23:00	3.0	34	0-3	2
Cabin Fever	Tuesday|18:00-2:00	4.5	26	4-5	2
Big Smoke Burger	Wednesday|10:30-21:00	3.0	47	0-3	3
99 Cent Sushi	Wednesday|11:00-23:00	2.0	5	0-3	3
Pizzaiolo	Wednesday|9:00-23:00	3.0	34	0-3	3
Edulis	Wednesday|18:00-23:00	4.0	89	4-5	3
Cabin Fever	Wednesday|18:00-2:00	4.5	26	4-5	3
Big Smoke Burger	Thursday|10:30-21:00	3.0	47	0-3	4
99 Cent Sushi	Thursday|11:00-23:00	2.0	5	0-3	4
Pizzaiolo	Thursday|9:00-23:00	3.0	34	0-3	4
Edulis	Thursday|18:00-23:00	4.0	89	4-5	4
Cabin Fever	Thursday|18:00-2:00	4.5	26	4-5	4
Big Smoke Burger	Friday|10:30-21:00	3.0	47	0-3	5
99 Cent Sushi	Friday|11:00-23:00	2.0	5	0-3	5
Pizzaiolo	Friday|9:00-4:00	3.0	34	0-3	5
Edulis	Friday|18:00-23:00	4.0	89	4-5	5
Cabin Fever	Friday|18:00-2:00	4.5	26	4-5	5
Pizzaiolo	Saturday|10:00-4:00	3.0	34	0-3	6
Big Smoke Burger	Saturday|10:30-21:00	3.0	47	0-3	6

Hours distribution for low and high star range groups

Restaurants with high ratings mostly open and close later than restaurants with low ratings.

Search for the differences between two business groups based on the ones that are open and the ones that are closed.

```
SELECT is_open,
     AVG(stars) as avg_stars,
     SUM(review_count) as total_review
FROM business as b
INNER JOIN category as c
ON b.id = c.business_id
WHERE city = 'Toronto' and category = 'Restaurants'
GROUP BY is_open
```

```
+---------+-----------+--------------+
| is_open | avg_stars | total_review |
+---------+-----------+--------------+
|       0 |       3.0 |           13 |
|       1 |       3.5 |          286 |
+---------+-----------+--------------+
```

The query above returns the two groups of restaurants based in Toronto. Group “1” defines the open restaurants, and group “0” for the closed.

Differences:

- Average stars rating for open restaurants is higher - 3.5 than the average stars rating for closed - 3.0.
- The total number of reviews for open businesses is 22 times the total review number of closed restaurants.

Part 3. Prepare a subset of the Yelp dataset to make my own data observation and analysis

In the last part of this project, I want to find the business categories where users' friends left more reviews. I started my research by joining tables: friend, user, review, business, and category to count friends that left a review by category.

```
SELECT COUNT(*) total
FROM friend as f
INNER JOIN user as u ON f.user_id = u.id
INNER JOIN review as r ON u.id = r.user_id
INNER JOIN business as b ON r.business_id = b.id
INNER JOIN category as c ON b.id = c.business_id
```

The code above returned an unexpected result with value = 0. To understand if it is an issue with the running code or the dataset itself, I started the investigation by checking the number of unique categories after joining category and business tables.

```
SELECT COUNT(DISTINCT c.category),
       COUNT(DISTINCT b.id)
FROM category as c
INNER JOIN business b ON b.id = c.business_id
```

The result showed the total number of unique categories equal to 257 and business 184. That means not all businesses are assigned to the categories.

The next step is to count reviews for the joined business and categories table only for the business table to determine the difference.

```
SELECT COUNT(*) total
FROM review as r
INNER JOIN business as b ON r.business_id = b.id
INNER JOIN category as c ON b.id = c.business_id
```

—

```
SELECT COUNT(*) total
FROM review as r
INNER JOIN business as b ON r.business_id = b.id
```

After querying the dataset, I found the total amount of reviews for

- businesses = 636
- businesses and categories = 73

Having such different results, I can see that not all businesses with reviews have categories.

The next step of the analysis is to add a user table to look for the intersection with business. There are four records found.

```
SELECT COUNT(*) as total
FROM review as r
INNER JOIN user as u ON u.id = r.user_id
INNER JOIN business as b ON r.business_id = b.id
```

Following the previous code, I added the category table, and there are no intersections.

```
SELECT COUNT(*) as total
FROM review as r
INNER JOIN user as u ON u.id = r.user_id
INNER JOIN business as b ON r.business_id = b.id
INNER JOIN category as c ON b.id = c.business_id
```

Conclusion:

By investigating the Yelp data I found that the dataset contains inconsistent data and does not let me proceed with the chosen type of analysis.