Profiling and Analyzing the Yelp Dataset

authored by Olha A.

"Attention! All queries are designed to be executed in Yelp Dataset SQL Lookup to retrieve the necessary data for each task. Since there was no option to create temporary tables in places where it would simplify the query (using existing queries), some parts of the code are duplicated with certain modifications."

Part 1: Yelp Dataset Profiling and Understanding

1. Profile the data by finding the total number of records for each of the tables below:

Answer:

#	table_name	number of records
i.	attribute	10000
ii.	business	10000
iii.	category	10000
iv.	checkin	10000
v.	elite_years	10000
vi.	friend	10000
vii.	hours	10000
viii.	photo	10000
ix.	review	10000
x.	tip	10000
x.	user	10000

SQL code used to arrive at answer:

```
-- This SQL query counts the total number of records in the 'table_name' table.

-- Start by selecting the 'COUNT(*)' aggregate function to count all records.
SELECT COUNT(*)
-- Specify the table from which we want to count records (replace 'table_name' with
the actual table name).
FROM table_name;
```

or alternatively we can create temporary table

```
-- Create a temporary table 'total_number_records' to store counts of records in
different tables.

-- Define the columns for the 'total_number_records' table.
CREATE TEMP TABLE total_number_records (
                    column_name VARCHAR(255), -- Name of the table
                    N_records INT -- Number of records in the table
                    );

-- Insert data into the 'total_number_records' table.
-- Replace 'attribute', 'business', 'category', etc., with actual table names.
```

```
-- Use subqueries to count records in each table.
INSERT INTO total_number_records (column_name, N_records)
VALUES
	('attribute', (SELECT COUNT(*) FROM attribute)),
	('business', (SELECT COUNT(*) FROM business)),
	('category', (SELECT COUNT(*) FROM category)),
	('checkin', (SELECT COUNT(*) FROM checkin)),
	('elite_years', (SELECT COUNT(*) FROM elite_years)),
	('friend', (SELECT COUNT(*) FROM friend)),
	('hours', (SELECT COUNT(*) FROM hours)),
	('photo', (SELECT COUNT(*) FROM photo)),
	('review', (SELECT COUNT(*) FROM review)),
	('tip', (SELECT COUNT(*) FROM tip)),
	('user', (SELECT COUNT(*) FROM user));
```

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.

Answer:

#	table_name.key_name		count of distinct values
i.	business		10000
ii.	hours		1562
iii.	gategory		2643
iv.	attribute		1115
v.	review	.id	10000
		.business_id	8090
		.user_id	9581
vi.	checkin		493
vii.	photo	.business_id	6493
		.id	10000
viii.	tip	.user_id	537
		.business_i	3979
ix.	user		10000
x.	friend		11
xi.	elite_years		2780

Note: Primary Keys are denoted in the ER-Diagram with a yellow key icon and foreign key with a red icon and the same logic is used in the Answer table.

SQL code used to arrive at answer:

```
-- Calculate the count of distinct values in a primary key (foreign key) column.

-- Replace 'table_name' with the actual table name.
```

```
SELECT COUNT(DISTINCT(key))
FROM table_name;
```

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

Answer: no

number_with_null_records
0

SQL code used to arrive at answer:

```
-- Count the number of records in the 'user' table where any of the specified
columns contain NULL values.

SELECT Count(*) number_with_null_records
FROM user
WHERE NULL IN (id, name, review_count, yelping_since,
                    useful, funny, cool, fans, average_stars,
                    compliment_hot, compliment_more, compliment_profile,
                    compliment_cute, compliment_list, compliment_note,
                    compliment_plain, compliment_cool, compliment_funny,
                    compliment_writer, compliment_photos);
```

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

SQL code used to arrive at answer:

```
-- Calculate the minimum, maximum, and average values of a specific column in the
'table_name' table.

-- Replace 'column_name' with the actual column name you want to analyze.
-- Replace 'table_name' with the actual table name.
SELECT
       MIN(column_name) AS min,
       MAX(column_name) AS max,
       AVG(column_name) AS avg
FROM table_name;
```

Answer:

#	table	column	min	max	avg
i.	review	stars	1	5	3.7082
ii.	business	stars	1.0	5.0	3.6549
iii.	tip	likes	0	2	0.0144
iv.	checkin	count	1	53	1.9414
v.	user	review_count	0	2000	24.2995

5. List the cities with the most reviews in descending order:

SQL code used to arrive at answer:

```
-- Calculate the total review count for businesses in each city and order the
results by review count in descending order.
SELECT city,
       SUM(review_count) AS numb_reviews
FROM business
GROUP BY city
ORDER BY numb_reviews DESC;
```

Result Below:

city	numb_reviews
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
North Las Vegas	2438
Markham	2352
Champaign	2029
Stuttgart	1849
Surprise	1520
Lakewood	1465
Goodyear	1155
(Output limit exceeded, 25 of 362 total rows shown)	

6. Find the distribution of star ratings to the business in the following cities:

SQL code used to arrive at answer:

```
-- Calculate the count of businesses with each star rating in the city name and order the results by star rating in descending order.

SELECT city,
       stars,
       COUNT(stars) AS numb_stars
FROM business
WHERE city = "city name"
GROUP BY stars
ORDER BY stars DESC;
```

Resulting Table

#	city	Resulting Table	
		stars	numb_stars
i.	Avon	5.0	1
		4.5	1
		4.0	2
		3.5	3
		2.5	2
		1.5	1
ii.	Beachwood	5.0	5
		4.5	2
		4.0	1
		3.5	2
		3.0	2
		2.5	1
		2.0	1

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

SQL code used to arrive at answer:

```
-- This query retrieves the top 3 users based on their total number of reviews, along with their user IDs and names.

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

Result:

id	name	review_count
-G7Zkl1wIWBBmD0KRy_sCw	Gerald	2000
-3s52C4zL_DHRK0ULG6qtg	Sara	1629

-8lbUNlXVSoXqaRRiHiSNg	Yuri	1339

8. Does posing more reviews correlate with more fans?

```
-- Calculate the R-squared value for the correlation between reviews and fans

SELECT ROUND(
            -- Calculate the numerator of the R-squared formula
            (
            (SUM((review_count - avg_review)*(fans - avg_fans))) *
            (SUM((review_count - avg_review)*(fans - avg_fans)))
            ) /
            -- Calculate the denominator of the R-squared formula
            (
            (SUM((review_count - avg_review)*(review_count - avg_review))) *
            (SUM((fans - avg_fans)*(fans - avg_fans)))
            ), 4) AS r2_reviews_fans

FROM user,
      -- Subquery to calculate the average review count (avg_review) and average
      number of fans (avg_fans) across all users
            (
            SELECT AVG(review_count) as avg_review,
            AVG(fans) AS avg_fans
            FROM user
            )
```

Result:

r2_reviews_fans
0.4371

Interpretation of the results:

To answer the question of whether posting more reviews correlates with having more fans, we're calculating the squared correlation coefficient (R-squared) between the number of reviews (review_count) and the number of fans (fans) for users in your dataset.

Pearson's correlation coefficient (r) of 0.661 indicates a moderate positive linear correlation between the number of reviews and the number of fans. This means that as the number of reviews increases, there is a tendency for the number of fans to also increase, and vice versa. The positive sign of the correlation coefficient indicates that when one variable (reviews) goes up, the other variable (fans) tends to go up as well. However, the strength of this correlation is considered moderate, meaning that it's not extremely strong but still indicates a noticeable relationship between the two variables.

9. Are there more reviews with the word "love" or with the word "hate" in them?

Answer: There are 1548 more reviews with the word “love”, then reviews with word "hate".

SQL code used to arrive at answer:

```
-- Calculate the count of reviews containing the word "hate" as n_hate
-- Calculate the count of reviews containing the word "love" as n_love
SELECT
	(SELECT COUNT(id) FROM review WHERE text LIKE '%hate%') AS n_hate,
	(SELECT COUNT(id) FROM review WHERE text LIKE '%love%') AS n_love;
```

Result:

n_hate	n_love
232	1780

10. Find the top 10 users with the most fans:

SQL code used to arrive at answer:

```
-- Retrieve the top 10 users with the most fans
SELECT id, name, fans
FROM user
ORDER BY fans DESC
Limit 10
```

Result:

id	name	fans
-9I98YbNQnLdAmcYfb324Q	Amy	503
-8EnCioUmDygAbsYZmTeRQ	Mimi	497
--2vR0DIsmQ6WfcSzKWigw	Harald	311
-G7Zkl1wIWBBmD0KRy_sCw	Gerald	253
-0IiMAZI2SsQ7VmyzJjokQ	Christine	173
-g3XIcCb2b-BD0QBCcq2Sw	Lisa	159
-9bbDysuiWeo2VShFJJtcw	Cat	133
-FZBTkAZEXoP7CYvRV2ZwQ	William	126
-9da1xk7zgnnfO1uTVYGkA	Fran	124
lh59ko3dxChBSZ9U7LfUw	Lissa	120

Part 2: Inferences and Analysis

1. Pick one city and category of our 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.

City = Toronto , Category = Restaurants , 2-3 stars and 4-5 stars groups

SQL code used to get data:

```
-- Retrieve businesses in Toronto categorized as "Restaurants" with star ratings
between 2-3 or 4-5

SELECT *
FROM business b
	LEFT JOIN category c ON b.id=c.business_id
WHERE (c.category = 'Restaurants' AND b.city = 'Toronto' ) AND
	(b.stars BETWEEN 2 AND 3 OR b.stars BETWEEN 4 AND 5)
```

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

SQL code used to arrive at answer:

```
-- This SQL query retrieves data related to restaurants in Toronto and categorizes
them based on their star ratings and operating hours.

SELECT *,
	COUNT(*) AS numb_restaurants,

	-- Categorize restaurants' opening time
	CASE WHEN CAST(start_hour AS TIME) BETWEEN
			CAST('00:00:00' AS TIME) AND
			CAST('12:00:00' AS TIME) THEN 'Before Noon'
		ELSE 'After Noon'
	END AS open_time_period,

	-- Categorize restaurants' closing time
	CASE WHEN CAST(end_hour AS TIME) BETWEEN
		CAST('12:00:00' AS TIME) AND
		CAST('23:59:00' AS TIME) THEN 'Before Midnight'
		ELSE 'After Midnight'
	END AS close_time_period

FROM (
```

```
        -- Subquery to process the initial data
        SELECT
                -- Categorize restaurants by star ratings
                CASE WHEN b.stars BETWEEN 2 AND 3 THEN '2-3'
                     WHEN b.stars BETWEEN 4 AND 5 THEN '4-5'
                END AS stars_group,

                -- Extract the day from hours
                SUBSTR(TRIM(h.hours), 1, INSTR(h.hours, '|') - 1) AS day,

                -- Extract the starting hour
                SUBSTR(TRIM(h.hours), INSTR(h.hours, '|') + 1, INSTR(h.hours, '-') -
                        INSTR(h.hours, '|') - 1) AS start_hour,

                -- Extract the ending hour
                SUBSTR(TRIM(h.hours), INSTR(h.hours, '-') + 1) AS end_hour

        FROM business b
                LEFT JOIN category c ON b.id = c.business_id
                LEFT JOIN hours h ON h.business_id = b.id
        WHERE c.category = 'Restaurants' AND
                b.city = 'Toronto' AND
                stars_group <> 'None'
        ) AS TimePeriods

-- Filter out 'None' values in day, start_hour, and end_hour
WHERE day <> 'None' OR start_hour <> 'None' OR end_hour <> 'None'

-- Group the results by day and star rating
GROUP BY day, stars_group, start_hour, end_hour

-- Sort the results by day and star rating
ORDER BY day, stars_group;
```

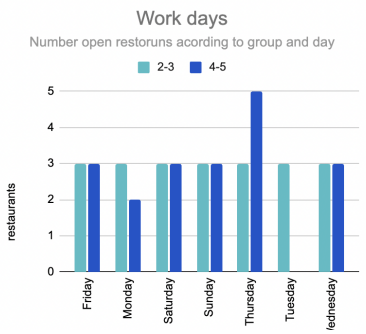
Result:

stars_group	day	start_hour	end_hour	numb_ restaurants	start_time_period	end_time_period
4-5	Friday	11:00:00	23:00:00	1	Before Noon	Before Midnight
4-5	Friday	18:00:00	23:00:00	1	After Noon	Before Midnight
4-5	Friday	18:00:00	02:00:00	1	After Noon	After Midnigh
4-5	Monday	11:00:00	23:00:00	1	Before Noon	Before Midnight
4-5	Monday	16:00:00	02:00:00	1	After Noon	After Midnigh
4-5	Saturday	11:00:00	23:00:00	1	Before Noon	Before Midnight
4-5	Saturday	16:00:00	02:00:00	1	After Noon	After Midnigh

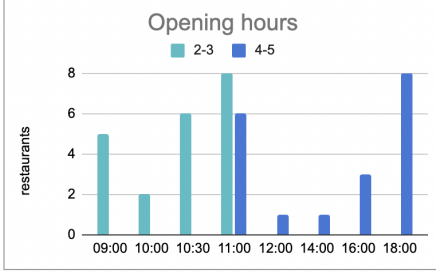
4-5	Saturday	18:00:00	23:00:00	1	After Noon	Before Midnight
4-5	Sunday	12:00:00	16:00:00	1	After Noon	Before Midnight
4-5	Sunday	14:00:00	23:00:00	1	After Noon	Before Midnight
4-5	Sunday	16:00:00	02:00:00	1	After Noon	After Midnigh
4-5	Thursday	11:00:00	23:00:00	1	Before Noon	Before Midnight
4-5	Thursday	18:00:00	23:00:00	1	After Noon	Before Midnight
4-5	Thursday	18:00:00	02:00:00	1	After Noon	After Midnigh
4-5	Thursday	11:00:00	23:00:00	1	Before Noon	Before Midnight
4-5	Thursday	18:00:00	02:00:00	1	After Noon	After Midnigh
4-5	Wednesday	11:00:00	23:00:00	1	Before Noon	Before Midnight
4-5	Wednesday	18:00:00	23:00:00	1	After Noon	Before Midnight
4-5	Wednesday	18:00:00	02:00:00	1	After Noon	After Midnigh
2-3	Friday	10:30	21:00	1	Before Noon	Before Midnight
2-3	Friday	11:00	23:00	1	Before Noon	Before Midnight
2-3	Friday	09:00	04:00	1	Before Noon	After Midnigh
2-3	Monday	10:30:00	21:00:00	1	Before Noon	Before Midnight
2-3	Monday	11:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Monday	09:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Saturday	10:00:00	04:00:00	1	Before Noon	After Midnigh
2-3	Saturday	10:30:00	21:00:00	1	Before Noon	Before Midnight
2-3	Saturday	11:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Sunday	10:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Sunday	11:00:00	19:00:00	1	Before Noon	Before Midnight
2-3	Sunday	11:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Thursday	10:30:00	21:00:00	1	Before Noon	Before Midnight
2-3	Thursday	11:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Thursday	09:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Tuesday	10:30:00	21:00:00	1	Before Noon	Before Midnight
2-3	Tuesday	11:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Tuesday	09:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Wednesday	10:30:00	21:00:00	1	Before Noon	Before Midnight
2-3	Wednesday	11:00:00	23:00:00	1	Before Noon	Before Midnight
2-3	Wednesday	09:00:00	23:00:00	1		lidnight

A little bit of

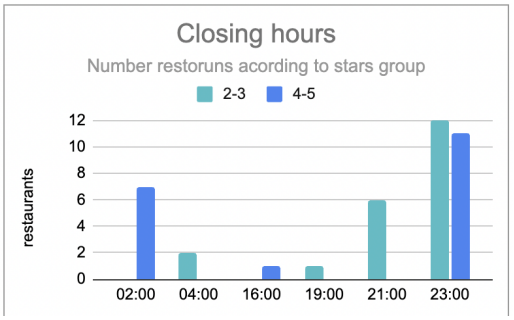
SUM of numb_restaurants	*stars_group*		
day	2-3	4-5	Grand Total
Friday	3	3	6
Monday	3	2	5
Saturday	3	3	6
Sunday	3	3	6
Thursday	3	5	8
Tuesday	3		3
Wednesday	3	3	6
Grand Total	**21**	**19**	**40**



COUNTA of star	*stars_group*		
start_hour	2-3	4-5	Grand Total
09:00	5		5
10:00	2		2
10:30	6		6
11:00	8	6	14
12:00		1	1
14:00		1	1
16:00		3	3
18:00		8	8
Grand Total	**21**	**19**	**40**



COUNTA of star stars_group		
end_hour	2-3	4-5
02:00		7
04:00	2	
16:00		1
19:00	1	
21:00	6	
23:00	12	11
Grand Total	**21**	**19**



SUM of numb_restaurants	start_time_period	end_time_period		
	After Noon		Before Noon	
stars_group	After Midnigh	Before Midnight	After Midnigh	Before Midnight
2-3			2	19
4-5	7	6		6
Grand Total	**7**	**6**	**2**	**25**

Answering the question:

Yes, star-groups we chose to analyze have a different distribution of working time. Restaurants with 2-3 stars most of all start working before noon and end before midnight, while Restaurants with 4-5 stars are open most likely after noon and closing before midnight. 2-3 star restaurants work usually all week, where 4-5 stars are closed on Tuesdays and the biggest amount are open on Thursday. Each of the selected groups has a few different opening and closing time hours.Where 11:00 is common for opening time and 23:00 for closing time.

Warning: We did NOT TAKE INTO ACCOUNT 'is_open' **characteristic of businesses!!!**

ii. Do the two groups we chose to analyze have a different number of reviews? Select all information that we need:

SQL code used to arrive at answer:

```
-- This SQL query retrieves data for restaurants in Toronto that fall into two star rating groups (2-3 stars and 4-5 stars).

SELECT
	-- Create a new column 'stars_group' for star rating categories
	CASE
		WHEN b.stars BETWEEN 2 AND 3 THEN '2-3'
		WHEN b.stars BETWEEN 4 AND 5 THEN '4-5'
	END AS stars_group,
	is_open,
	name,
	address,
	stars,
```

```
        review_count
FROM business b
        LEFT JOIN category c ON b.id = c.business_id


WHERE
        -- Filter for restaurants in Toronto
        (c.category = 'Restaurants' AND b.city = 'Toronto') AND
        -- Filter for star ratings 2-3 and 4-5
        (b.stars BETWEEN 2 AND 3 OR b.stars BETWEEN 4 AND 5)
-- Sort the results by star rating and review count
ORDER BY stars, review_count;
```

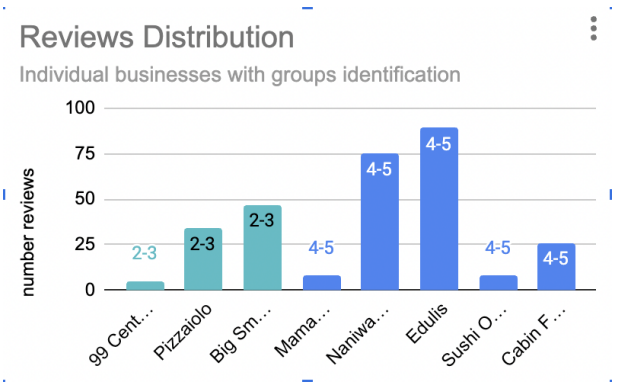
Result:

stars_group	is_open	name	address	stars	review_count
2-3	0	99 Cent Sushi	389 Church Street	2.0	5
2-3	1	Pizzaiolo	270 Adelaide Street W	3.0	34
2-3	1	Big Smoke Burger	260 Yonge Street	3.0	47
4-5	0	Mama Mia	816 Saint Clair Avenue W	4.0	8
4-5	1	Naniwa-Taro	7 Byng Avenue	4.0	75
4-5	1	Edulis	169 Niagara Street	4.0	89
4-5	1	Sushi Osaka	5084 Dundas Street W	4.5	8
4-5	1	Cabin Fever	1669 Bloor Street W	4.5	26

Than for each group min, max and average and total amount of reviews:

stars_group	min	max	avg	total
2-3	5	47	28.67	86
4-5	8	89	41.2	206

A little bit of magic from google sheets:



Restaurants in Toronto with 4-5 stars ratings have more reviews than those with 2-3 stars. Individually, 4-5 star restaurants have a total of 206 reviews, while 2-3 star restaurants have 86 reviews.

On average, each 4-5 star restaurant has approximately 89 reviews, whereas each 2-3 star restaurant has around 47 reviews.

These findings are also visually represented in the accompanying graphics.

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

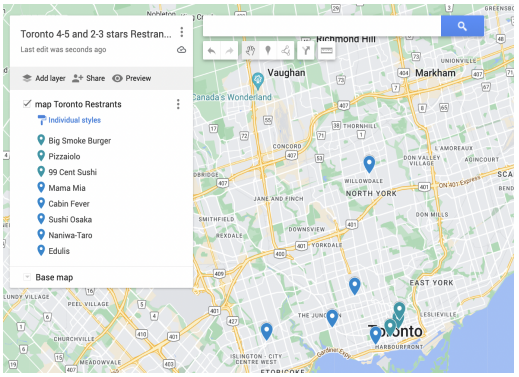
SQL code used for analysis:

```
-- Retrieve open status, name, state, city, postal code, address, latitude, and
longitude for restaurants in Toronto with star ratings between 2-3 and 4-5.
SELECT
	CASE
		WHEN b.stars BETWEEN 2 AND 3 THEN '2-3'
		WHEN b.stars BETWEEN 4 AND 5 THEN '4-5'
	END AS stars_group,
	is_open, name,
	state, city,
	postal_code, address,
	latitude, longitude
FROM business b LEFT JOIN category c ON b.id=c.business_id
WHERE (c.category = 'Restaurants' AND
	b.city = 'Toronto' )AND
	(b.stars BETWEEN 2 AND 3 OR b.stars BETWEEN 4 AND 5)
ORDER BY stars_group
```

Result:

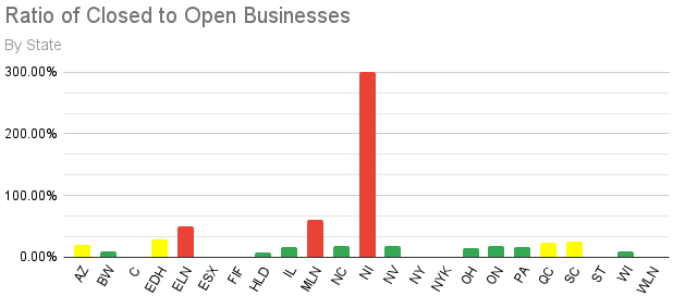
stars_group	is_open	name	state	city	postal_code	address	latitude	longitude
2-3	1	Big Smoke Burger	ON	Toronto	M4B 2L9	260 Yonge Street	43.6546	-79.3805
2-3	1	Pizzaiolo	ON	Toronto	M5H 1X6	270 Adelaide Street W	43.6479	-79.3901
2-3	0	99 Cent Sushi	ON	Toronto	M5B 2E5	389 Church Street	43.6614	-79.379
4-5	0	Mama Mia	ON	Toronto	M6C 1B6	816 Saint Clair Avenue W	43.6809	-79.4302
4-5	1	Cabin Fever	ON	Toronto	M6P 1A6	1669 Bloor Street W	43.6553	-79.4567
4-5	1	Sushi Osaka	ON	Toronto	M9A 1C2	5084 Dundas Street W	43.6452	-79.5324
4-5	1	Naniwa-Taro	ON	Toronto	M2N 5R6	7 Byng Avenue	43.7766	-79.4142
4-5	1	Edulis	ON	Toronto	M5V	169 Niagara Street	43.6419	-79.4066

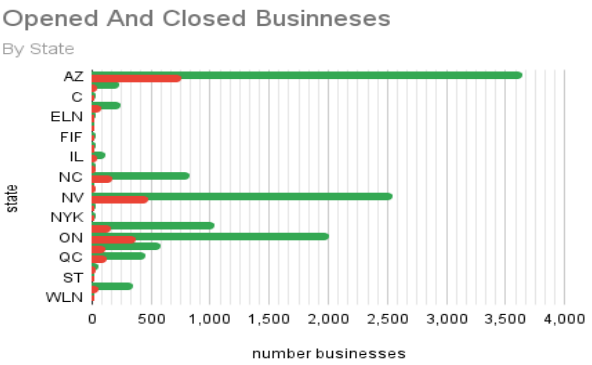
With using this dataset and Google maps we can create map
(points from different groups have different color)
2-3 stars restaurants are pleased more close to each other in the Old/Downtown Toronto area while 4-5 stars rate restaurants are located in different parts of Toronto such Fashion District, West Bend, slington, umewood, Yonge-Doris (in HighPark, Etobicoke, Willowdale, Niagara neighborhoods)
[link to the map]

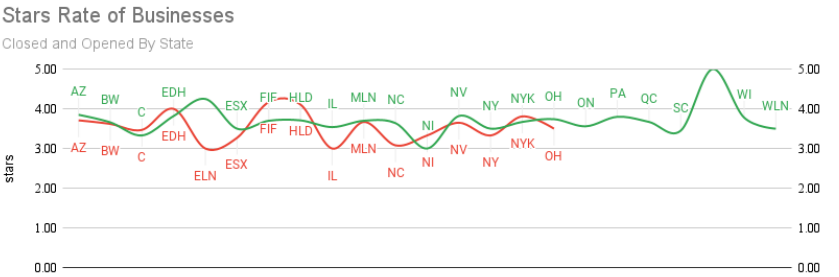


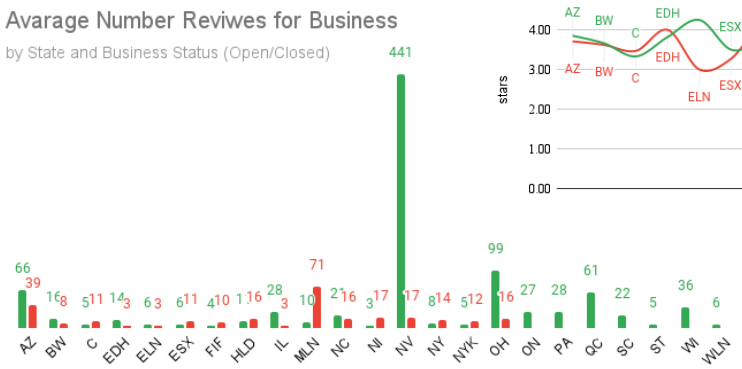
2. Group business based on the ones that are open and the ones that are closed. What differences can you find between the ones that are still open and the ones that are closed? List at least two differences and the SQL code we used to arrive at our answer.

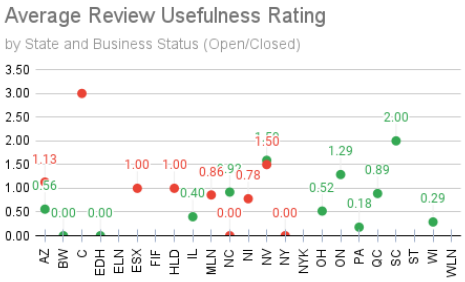
A little bit of magic from Google sheets:
(Closed Business = RED, Opened Business = GREEN)
!!! Warning!!! We've also included businesses with no reviews or working hours.

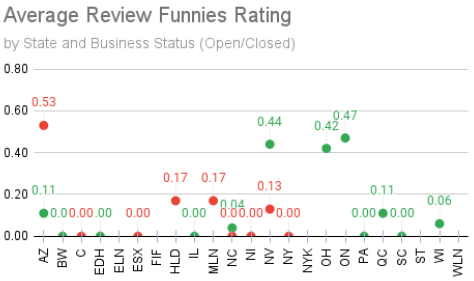


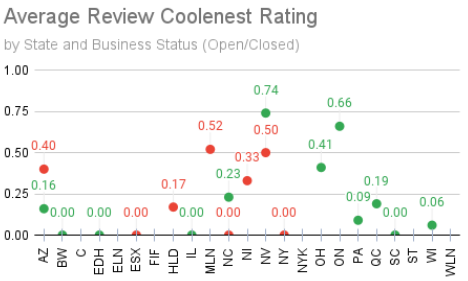


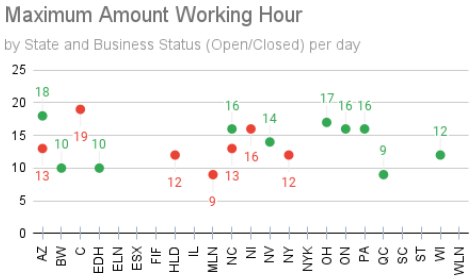


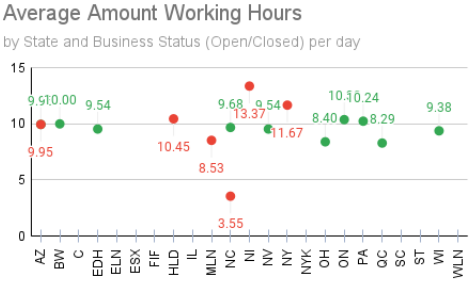


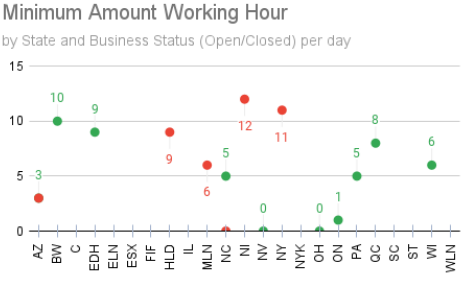












i. Difference 1:

Usually average stars rating of the closed ones are lower than the opened business for State Level of deepneth.

ii. Difference 2:

Usually the average review for business is valued as useful more often for closed ones than for opened.

iii. Difference 3:

There are some states where the ratio of closed to opened businesses are very high (NI, MLN,ELN).

iiii. Difference 4: Usually the closed business has less reviews then those which are still open.

SQL code used for analysis:

WE can take a look at opened and closed businesses on different levels of depth like on state or state/city or state/city/category or even more deeply.

So each level can be reached by adding addition level of grouping accordingly adding city, category to the outer SELECT statement into the next query:

State level:

```
-- This SQL code calculates various statistics and aggregations for
businesses, focusing on open and closed businesses, grouped by state.

SELECT is_open, state,
       ROUND(AVG(stars),2) avg_stars,
       SUM(open) num_open,
       SUM(closed) num_closed,
       COUNT(DISTINCT category) num_categories,
       ROUND(AVG(review_count)) avg_numb_reviews,
       SUM(review_count) total_reviews,
       ROUND(AVG(useful),2) avg_review_useful,
       ROUND(AVG(funny),2) avg_review_funny,
       ROUND(AVG(cool),2) avg_review_cool,
       ROUND(AVG(ABS(CAST(end_hour AS TIME) - CAST(start_hour AS TIME))),2)
       avg_open_hours,
       ROUND(MIN(ABS(CAST(end_hour AS TIME) - CAST(start_hour AS TIME))),2)
       min_open_hours,
       ROUND(MAX(ABS(CAST(end_hour AS TIME) - CAST(start_hour AS TIME))),2)
       max_open_hours
FROM (
       -- Subquery to combine data from various tables and calculate
       intermediate values
```

```
        SELECT
                b.id, b.state, b.city, b.stars, b.review_count, b.is_open,
                C.category,
                r.useful, r.funny, r.cool,
                -- Determine if photos exist for the business
                CASE WHEN p.id <> 'None' THEN 1 ELSE 0 END photos,
                -- Identify closed businesses
                CASE WHEN b.is_open = 0 THEN 1 ELSE 0 END closed,
                -- Identify open businesses
                CASE WHEN b.is_open = 1 THEN 1 ELSE 0 END open,
                -- Identify user reviews
                CASE WHEN r.user_id = 'None' THEN 0 ELSE 1 END user_review,
                -- Extract the day from business hours
                SUBSTR(TRIM(h.hours), 1, INSTR(h.hours, '|') - 1) AS day,
                SUBSTR(TRIM(h.hours), INSTR(h.hours, '|') + 1, INSTR(h.hours, '-') -
                        INSTR(h.hours, '|') - 1) AS start_hour,
                SUBSTR(TRIM(h.hours), INSTR(h.hours, '-') + 1) AS end_hour
        FROM business b
                LEFT JOIN category c ON b.id = c.business_id
                LEFT JOIN review r ON b.id = r.business_id
                LEFT JOIN hours h ON b.id = h.business_id
                LEFT JOIN photo p ON b.id = p.business_id

        )
GROUP BY is_open, state
ORDER BY num_open DESC, state, num_closed DESC;
```

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

i. Indicate the type of analysis we chose to do:

To gain insights into customer satisfaction and detect trends, we will conduct sentiment analysis on user reviews with business information on different levels of deepness.

ii. About the type of data we will need for our analysis and why we chose that data:

For the sentiment analysis and trend detection project, the chosen data encompasses a rich variety of information, which is essential for gaining comprehensive insights into customer satisfaction and business performance. The dataset includes both review-specific and business-specific data, making it a well-rounded source for analysis.

On the review side, details such as the text of the reviews, star ratings, and user interactions (usefulness, coolness, and funniness) are crucial. The text of the reviews allows for sentiment analysis, helping to determine whether customers are expressing positive, negative, or neutral sentiments about their experiences. The star ratings provide a

quantifiable measure of customer satisfaction. Additionally, user interactions offer insights into the perceived helpfulness and engagement levels of reviews, which can be indicative of their influence on other customers.

On the business side, information about each business's name, location (state and city), star rating, review count, and categories it belongs to is essential. Business names and locations are essential for identifying and distinguishing businesses within the dataset. Star ratings and review counts offer an overview of the businesses' overall performance and popularity. Finally, categories provide valuable context by categorizing businesses into different industries or types.

By combining these two sets of data and structuring them in the final dataset, the analysis can delve into various aspects of customer satisfaction, such as how sentiments vary across different types of businesses, how reviews impact business ratings, and whether trends emerge in specific categories or locations. This well-curated dataset is an ideal choice for uncovering actionable insights that businesses can use to enhance their services and customer experiences.

iii. Finished dataset:

We possess a table comprising 14 rows and 14 columns. The table below provides details about the column names, their respective contents, and the associated data types.

#	Column name	Example of content	Data type
1	review_id	-5f6HtLwJbG84021Gqi9Zw	VARCHAR(22)
2	text	Mr. Esser...Banning Buddy Guy from future Blues events in the city of Pittsburgh is the most assinine thing i've ever heard anyone do. It's 2015. It's a Blues Fest in Pittsburgh, not Dade Cunty Florida. Have you ever been to a Steeler game? You cheat all Buddy fans and Blues fans alike. Screw ever coming into your understaffed club again. Im sure Buddy can work around you and entertain Blues fans before this Legend is gone for good. You've lost the plot for sure.	TEXT
3	review_rating	2	INT
4	review_is_useful	0	INT
5	review_is_cool	0	INT
6	review_is_funny	0	INT
7	business_id	-9lOQ0Lfm8wiu8eSdUXS8A	VARCHAR(22)
8	business_name	Moondogs Pub	VARCHAR(255)
9	business_rating	3.5	FLOAT
10	number_reviews	723	INT
11	business_is_open	1	TINYINT
12	state	PA	VARCHAR(255)
13	city	Pittsburgh	VARCHAR(255)
14	category	Music Venues, Arts & Entertainment, Nightlife	

iv. The SQL code used to create final dataset:

```
-- This SQL code retrieves data from the 'review' and 'business' tables, combining
information about reviews and their associated businesses.

SELECT r.id review_id, r.text,
       r.stars AS review_rating,
       r.useful AS review_is_useful, r.cool AS review_is_cool,
       r.funny AS review_is_funny,
       b.id AS business_id, b.name AS business_name,
       b.stars AS business_rating,
       b.review_count AS number_reviews,
       b.is_open AS business_is_open,
       b.state, b.city, categories
FROM review r
       INNER JOIN business b ON r.business_id = b.id
       -- Subquery to concatenate categories associated with each
business
       INNER JOIN (
              SELECT business_id,
                     -- Concatenate categories for each business
                     GROUP_CONCAT(category, ', ') AS categories
              FROM category
              GROUP BY business_id
              ) c ON b.id = c.business_id;
```