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

Part 1: Yelp Dataset Profiling and Understanding

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

For each table, I used the following SQL query to find the total number of records in the table:

```
SELECT
    COUNT(*)
FROM
    table;
```

i. attribute table = 10,000
ii. business table = 10,000
iii. category table = 10,000
iv. checkin table = 10,000
v. elite_years table = 10,000
vi. friend table = 10,000
vii. hours table = 10,000
viii. photo table = 10,000
ix. review table = 10,000
x. tip table = 10,000
xi. user table = 10,000

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.

For each table with primary or foreign key X, I used the following SQL query to find the total number of

distinct records in the table:

```
SELECT
    COUNT(DISTINCT X)
FROM
    table;
```

i. business = 10,000
ii. hours = 1562
iii. category = 2643
iv. attribute = 1115
v. review = 8090 (using business_id)
vi. checkin = 493
vii. photo = 10,000
viii. tip = 537 (using user_id)
ix. user = 10,000
x. friend = 11
xi. elite_years = 2780

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

Answer: No.

SQL code used to arrive at answer: For each column X in the user table, I ran the following query, which counts the number of nulls in the column X. For each column, 0 was returned, indicating that no column has null values.

```
SELECT
    COUNT(*) - COUNT(X)
FROM
    user;
```

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

For each table and column listed below, I used the following SQL query:

```
SELECT
    min(column),
    max(column),
    avg(column)
FROM
    table;
```

i. Table: Review, Column: Stars

min: 1 max: 5 avg: 3.7082

ii. Table: Business, Column: Stars

min: 1 max: 5 avg: 3.6549

iii. Table: Tip, Column: Likes

min: 0 max: 2 avg: 0.0144

iv. Table: Checkin, Column: Count

min: 1 max: 53 avg: 1.9414

v. Table: User, Column: Review_count

min: 0 max: 2000 avg: 24.2995

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

SQL code used to arrive at answer:

```
SELECT
    city,
    sum(review_count)
FROM
    business
GROUP BY
    city
ORDER BY
    sum(review_count) DESC;
```

Copy and Paste the Result Below:

city	sum(review_count)
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 |
+-----------------+---------------------+
(Output limit exceeded, 25 of 362 total rows shown)
```

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

i. Avon

SQL code used to arrive at answer:

```sql
SELECT
    stars AS star_rating,
    COUNT(*) AS count
FROM
    business
WHERE
    city = 'Avon'
GROUP BY
    stars
ORDER BY
    stars DESC;
```

Copy and Paste the Resulting Table Below:

```
+-------------+-------+
| star_rating | count |
+-------------+-------+
|         5.0 |     1 |
|         4.5 |     1 |
|         4.0 |     2 |
|         3.5 |     3 |
```

```
|         2.5 |     2 |
|         1.5 |     1 |
+-------------+-------+
```

ii. Beachwood

SQL code used to arrive at answer:

```
SELECT
    stars AS star_rating,
    COUNT(*) AS count
FROM
    business
WHERE
    city = 'Beachwood'
GROUP BY
    stars
ORDER BY
    stars DESC;
```

Copy and Paste the Resulting Table Below:

```
+-------------+-------+
| star_rating | count |
+-------------+-------+
|         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:

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

Copy and Paste the Result Below:

```
+------------------------+--------+--------------+
| id                     | name   | review_count |
+------------------------+--------+--------------+
| -G7Zkl1wIWBBmD0KRy_sCw | Gerald |         2000 |
| -3s52C4zL_DHRK0ULG6qtg | Sara   |         1629 |
| -8lbUNlXVSoXqaRRiHiSNg | Yuri   |         1339 |
+------------------------+--------+--------------+
```

8. Does posting more reviews correlate with more fans? Please explain your findings and interpretation of the results.

I ran the following query to compare review counts and fans:

```
SELECT
   name,
   review_count,
   fans
FROM
    user
```

```
ORDER BY
    review_count DESC,
    fans DESC;
```

The output was as follows:

```
+-----------+--------------+------+
| 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 |
```

```
+----------+-------------+------+
(Output limit exceeded, 25 of 10000 total rows shown)
```

Examining the table from bottom-to-top, we can see whether fans tend to increase as review_count increases. However, there does not seem to be any strong correlation between review_count and fans, since as the review_count increases, the number of fans sometimes decreases (sharply) and sometimes increases (sharply). Thus, there does not appear to be a (strong) correlation between posting more reviews and having more fans.

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

Answer:

There are more reviews with the word “love” (1780) compared to reviews with the word “hate” (232).

SQL code used to arrive at answer:

```
SELECT
    sum(CASE WHEN text LIKE '%love%' THEN 1 ELSE 0 END) AS
love_reviews, -- counts the number of reviews with the word
'love' in them
    sum(CASE WHEN text LIKE '%hate%' THEN 1 ELSE 0 END) AS
hate_reviews -- counts the number of reviews with the word
'hate' in them
FROM
    review;
```

```
+-------------+-------------+
| love_reviews | hate_reviews |
+-------------+-------------+
|         1780 |          232 |
```

```
+--------------+--------------+
```

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

SQL code used to arrive at answer:

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

Copy and Paste the Result Below:

```
+------------------------+-----------+------+
| 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 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 decided to use Toronto as the city and Restaurants as the category. I ran the following query to take an initial look at the Toronto restaurants, organized into two groups by rating:

```
SELECT
    b.city,
    c.category,
    b.name,
    -- The next column categorizes a restaurant based on its
number of stars
    CASE
        WHEN b.stars BETWEEN 2 AND 3 THEN 'Low Rating'
        WHEN b.stars BETWEEN 4 AND 5 THEN 'High Rating'
    END AS Rating
FROM
    business b
JOIN
    category c
ON
    b.id = c.business_id
WHERE
    Rating IS NOT NULL AND b.city = 'Toronto' AND c.category =
'Restaurants';
```

```
+---------+-------------+------------------
+-------------+
| city    | category    | name             | Rating
|
+---------+-------------+------------------
+-------------+
```

```
| Toronto | Restaurants | Sushi Osaka      | High
Rating |
| Toronto | Restaurants | Big Smoke Burger | Low Rating
|
| Toronto | Restaurants | Edulis           | High
Rating |
| Toronto | Restaurants | Pizzaiolo        | Low Rating
|
| Toronto | Restaurants | 99 Cent Sushi    | Low Rating
|
| Toronto | Restaurants | Cabin Fever      | High
Rating |
| Toronto | Restaurants | Mama Mia         | High
Rating |
| Toronto | Restaurants | Naniwa-Taro      | High
Rating |
+---------+-------------+------------------
+-------------+
```

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

In this first query (output below), I first look at the hours for Toronto restaurants with 4-5 stars.

```sql
SELECT
    b.name AS Restaurant,
    h.hours AS Hours,
    -- The next column categorizes a restaurant based on its
number of stars
    CASE
        WHEN b.stars BETWEEN 2 AND 3 THEN 'Low Rating'
        WHEN b.stars BETWEEN 4 AND 5 THEN 'High Rating'
    END AS Rating
FROM
```

```
    business b
JOIN
    category c
ON
    b.id = c.business_id
JOIN
    hours h
ON
    b.id = h.business_id
WHERE
    Rating IS NOT NULL AND c.category = 'Restaurants' AND
    b.city = 'Toronto' AND Rating = 'High Rating';
```

```
+-------------+-----------------------+-------------+
| Restaurant  | Hours                 | Rating      |
+-------------+-----------------------+-------------+
| Sushi Osaka | Monday|11:00-23:00    | High Rating |
| Sushi Osaka | Tuesday|11:00-23:00   | High Rating |
| Sushi Osaka | Friday|11:00-23:00    | High Rating |
| Sushi Osaka | Wednesday|11:00-23:00 | High Rating |
| Sushi Osaka | Thursday|11:00-23:00  | High Rating |
| Sushi Osaka | Sunday|14:00-23:00    | High Rating |
| Sushi Osaka | Saturday|11:00-23:00  | High Rating |
| Edulis      | Sunday|12:00-16:00    | High Rating |
| Edulis      | Friday|18:00-23:00    | High Rating |
| Edulis      | Wednesday|18:00-23:00 | High Rating |
| Edulis      | Thursday|18:00-23:00  | High Rating |
| Edulis      | Saturday|18:00-23:00  | High Rating |
| Cabin Fever | Monday|16:00-2:00     | High Rating |
| Cabin Fever | Tuesday|18:00-2:00    | High Rating |
| Cabin Fever | Friday|18:00-2:00     | High Rating |
| Cabin Fever | Wednesday|18:00-2:00  | High Rating |
| Cabin Fever | Thursday|18:00-2:00   | High Rating |
| Cabin Fever | Sunday|16:00-2:00     | High Rating |
| Cabin Fever | Saturday|16:00-2:00   | High Rating |
```

```
+------------+----------------------+------------+
```

Sushi Osaka is open every day, usually from 11am-11pm. Edulis is open Wednesday-Sunday, usually from 6pm-11pm. Cabin Fever is open every day, usually from 4pm or 6pm until 2am.

In this second query (output below), I then look at the hours for Toronto restaurants with 2-3 stars.

```
SELECT
    b.name AS Restaurant,
    h.hours AS Hours,
    -- The next column categorizes a restaurant based on its
number of stars
    CASE
        WHEN b.stars BETWEEN 2 AND 3 THEN 'Low Rating'
        WHEN b.stars BETWEEN 4 AND 5 THEN 'High Rating'
    END AS Rating
FROM
    business b
JOIN
    category c
ON
    b.id = c.business_id
JOIN
    hours h
ON
    b.id = h.business_id
WHERE
    Rating IS NOT NULL AND c.category = 'Restaurants' AND
    b.city = 'Toronto' AND Rating = 'Low Rating';

+------------------+----------------------
+------------+
| Restaurant       | Hours                | Rating
```

|
+------------------+----------------------
+------------+
| Big Smoke Burger | Monday|10:30-21:00 | Low Rating
|
| Big Smoke Burger | Tuesday|10:30-21:00 | Low Rating
|
| Big Smoke Burger | Friday|10:30-21:00 | Low Rating
|
| Big Smoke Burger | Wednesday|10:30-21:00 | Low Rating
|
| Big Smoke Burger | Thursday|10:30-21:00 | Low Rating
|
| Big Smoke Burger | Sunday|11:00-19:00 | Low Rating
|
| Big Smoke Burger | Saturday|10:30-21:00 | Low Rating
|
| Pizzaiolo | Monday|9:00-23:00 | Low Rating
|
| Pizzaiolo | Tuesday|9:00-23:00 | Low Rating
|
| Pizzaiolo | Friday|9:00-4:00 | Low Rating
|
| Pizzaiolo | Wednesday|9:00-23:00 | Low Rating
|
| Pizzaiolo | Thursday|9:00-23:00 | Low Rating
|
| Pizzaiolo | Sunday|10:00-23:00 | Low Rating
|
| Pizzaiolo | Saturday|10:00-4:00 | Low Rating
|
| 99 Cent Sushi | Monday|11:00-23:00 | Low Rating
|
| 99 Cent Sushi | Tuesday|11:00-23:00 | Low Rating

```
|
| 99 Cent Sushi    | Friday|11:00-23:00    | Low Rating
|
| 99 Cent Sushi    | Wednesday|11:00-23:00 | Low Rating
|
| 99 Cent Sushi    | Thursday|11:00-23:00  | Low Rating
|
| 99 Cent Sushi    | Sunday|11:00-23:00    | Low Rating
|
| 99 Cent Sushi    | Saturday|11:00-23:00  | Low Rating
|
+------------------+-----------------------
+------------+
```

Big Smoke Burger is open everyday, usually from 10:30am to 9pm. Pizzaiolo is open everyday, usually from 9am until 4pm or 11pm. 99 Cent Sushi is open everyday, usually from 11am to 11pm.

The higher-rated restaurants generally open later and may also stay open later.

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

Here is the SQL code (output below) for the number of reviews for each Toronto restaurant in both rating groups:

```sql
SELECT
    b.name AS Restaurant,
    review_count AS Number_of_Reviews,
    -- The next column categorizes a restaurant based on its number of stars
    CASE
        WHEN b.stars BETWEEN 2 AND 3 THEN 'Low Rating'
```

```
        WHEN b.stars BETWEEN 4 AND 5 THEN 'High Rating'
    END AS Rating
FROM
    business b
JOIN
    category c
ON
    b.id = c.business_id
WHERE
    Rating IS NOT NULL AND c.category = 'Restaurants' AND b.city
= 'Toronto'
ORDER BY
    Rating;
```

```
+------------------+--------------------+-------------+
| Restaurant       | Number_of_Reviews  | Rating      |
+------------------+--------------------+-------------+
| Sushi Osaka      |                  8 | High Rating |
| Edulis           |                 89 | High Rating |
| Cabin Fever      |                 26 | High Rating |
| Mama Mia         |                  8 | High Rating |
| Naniwa-Taro      |                 75 | High Rating |
| Big Smoke Burger |                 47 | Low Rating  |
| Pizzaiolo        |                 34 | Low Rating  |
| 99 Cent Sushi    |                  5 | Low Rating  |
+------------------+--------------------+-------------+
```

The higher-rated restaurants do generally seem to have more reviews.

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

The following query (output below) looks at the Toronto neighborhood for each restaurant in each rating group.

It seems that the lower-rated restaurants are in/around the downtown core, whereas the higher-rated restaurants may not be downtown.

```
SELECT
    b.name AS Restaurant,
    neighborhood AS Neighborhood,
    -- The next column categorizes a restaurant based on its
number of stars
    CASE
        WHEN b.stars BETWEEN 2 AND 3 THEN 'Low Rating'
        WHEN b.stars BETWEEN 4 AND 5 THEN 'High Rating'
    END AS Rating
FROM
    business b
JOIN
    category c
ON
    b.id = c.business_id
WHERE
    Rating IS NOT NULL AND c.category = 'Restaurants' AND b.city
= 'Toronto'
ORDER BY
    Rating;
```

```
+------------------+------------------------
+-------------+
| Restaurant       | Neighborhood           | Rating
|
+------------------+------------------------
+-------------+
| Sushi Osaka      | Etobicoke              | High
Rating |
| Edulis           | Niagara                | High
Rating |
| Cabin Fever      | High Park              | High
```

```
Rating |
| Mama Mia          |                           | High
Rating |
| Naniwa-Taro       | Willowdale                | High
Rating |
| Big Smoke Burger  | Downtown Core             | Low
Rating  |
| Pizzaiolo         | Entertainment District    | Low
Rating  |
| 99 Cent Sushi     | Downtown Core             | Low
Rating  |
+-----------------+-------------------------
+-------------+
```

2. Group businesses 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:

The open restaurants have a higher average rating than the closed restaurants: 3.68 versus 3.52.

ii. Difference 2:

The open restaurants have more reviews, on average, than the closed restaurants: the open restaurants have an average of 31.76 reviews each, while the closed restaurants have an average of 23.2 reviews each.

SQL code used for analysis:

```
SELECT
    is_open,
    round(avg(stars), 2) AS Average_Rating,
```

```
    round(avg(review_count), 2) AS Average_Review_Count
FROM
    business
GROUP BY
    is_open;
```

```
+---------+----------------+----------------------+
| is_open | Average_Rating | Average_Review_Count |
+---------+----------------+----------------------+
|       0 |           3.52 |                 23.2 |
|       1 |           3.68 |                31.76 |
+---------+----------------+----------------------+
```

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:

I chose to analyze the different characteristics of reviews and their reviewers for the five different star ratings of reviews.

ii. Write 1-2 brief paragraphs on the type of data you will need for your analysis and why you chose that data:

I decided to inner join the review and user tables, and then group by the “stars” column of the review table, which has the distinct values 1, 2, 3, 4, 5. The inner join of these tables consists of each Yelp review and all of the user information for the user who posted that review. I then used the average aggregate function to find, for each star rating, the average usefulness, funniness, and coolness of reviews with that star rating, as well as, for each star rating, the average usefulness, funniness, and coolness of users who left reviews with that star rating.

It seems that the most useful, funny, and cool reviews were the 3-star reviews, while the least useful, funny, and cool reviews were the 5-star reviews. The 3-star reviews were posted by the reviewers who were the most useful, funny, and cool, while the 1-star reviews were posted by the reviewers who were the least useful, funny, and cool.

iii. Output of your finished dataset:

Review_Rating	Review_Usefulness	Review_Funniness	Review_Coolness	User_Usefulness	User_Funniness	User_Coolness
1	1.5	0.75	0.38	18.38	6.0	7.0
2	1.0	0.8	0.4	312.4	240.8	

```
260.6 |
|             3 |               2.5 |             2.25
|            1.75 |           680.5 |          249.25 |
750.5 |
|             4 |              0.78 |             0.61
|            0.52 |           69.78 |           39.48 |
47.65 |
|             5 |              0.66 |             0.38
|            0.53 |          264.94 |          194.06 |
151.69 |
+---------------+-------------------+------------------
+-----------------+-----------------+----------------
+---------------+
```

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

```sql
SELECT
    r.stars AS Review_Rating,
    round(avg(r.useful), 2) AS Review_Usefulness,
    round(avg(r.funny), 2) AS Review_Funniness,
    round(avg(r.cool), 2) AS Review_Coolness,
    round(avg(u.useful), 2) AS User_Usefulness,
    round(avg(u.funny), 2) AS User_Funniness,
    round(avg(u.cool), 2) AS User_Coolness
FROM
    user u
JOIN
    review r
ON
    u.id = r.user_id
GROUP BY
    r.stars;
```