Ivan Radonjic

3/8/2024

SQL for Data Science

Profiling and Analyzing the Yelp Dataset

This project simulates real-world data analysis tasks using SQL. We'll be working with a dataset provided by Yelp, a US-based organization offering a platform for users to review and rate various businesses and organizations.

Objective: The objective of this project is to analyze the Yelp dataset to answer specific questions, make inferences, and gain insights into the behavior of users and organizations.

Dataset: Yelp has generously made a portion of their data available for personal, educational, and academic purposes. This dataset contains reviews and ratings provided by users across a wide range of organizations including businesses, restaurants, health clubs, hospitals, local governmental offices, and charitable organizations.

Assignment Overview: Question Profiling: We'll start by asking a series of questions about the dataset to better understand its characteristics.

Data Analysis: Using SQL queries, we'll analyze the dataset to answer the questions posed and uncover insights.

Part 1: Yelp Dataset Profiling and Understanding

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

i. *Attribute Table*

```
SELECT count(*) FROM attribute
```

Answer: 1000 Records

ii. *Business Table*

```
SELECT count(*) FROM business
```

Answer: 1000 Records

iii. *Category Table*

```
SELECT count(*) FROM category
```

Answer: 1000 Records

iv. *Checkin Table*

```
SELECT count(*) FROM checkin
```

Answer: 1000 Records

v. *Elite_years Table*

```
SELECT count(*) FROM elite_years
```

Answer: 1000 Records

vi. *Friend Table*

```
SELECT count(*) FROM friend
```

Answer: 1000 Records

vii. *Hours Table*

```
SELECT count(*) FROM hours
```

Answer: 1000 Records

viii. *Photo Table*

```
SELECT count(*) FROM photo
```

Answer: 1000 Records

ix. *Review Table*

```
SELECT count(*) FROM review
```

Answer: 1000 Records

x. *Tip Table*

```
SELECT count(*) FROM tip
```

Answer: 1000 Records

xi. *User Table*

```
SELECT count(*) FROM user
```

Answer: 1000 Records

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 Table*

```
SELECT count(distinct id) FROM business
```

Answer: 10000 Records
Key: id

ii. *Hours Table*

```
SELECT count(distinct business_id) FROM hours
```

Answer: 1562 Records
Key: business_id

iii. *Category Table*

```
SELECT count(distinct business_id) FROM category
```

Answer: 2643 Records
Key: business_id

iv. *Attribute Table*

```
SELECT count(distinct business_id) FROM attribute
```

Answer: 1115 Records
Key: business_id

v. *Review Table*

```
SELECT count(distinct id), count(distinct business_id),
count(distinct user_id)
FROM review
```

Answer: 10000 Records
Key: id

vi. *Checkin Table*

```
SELECT count(distinct business_id) FROM checkin
```

Answer: 493 Records
Key: business_id

vii. *Photo Table*

```
SELECT count(distinct id), count(distinct business_id)
FROM photo
```

Answer: 6493 Records
Key: business_id

viii. *Tip Table*

```
SELECT count(distinct business_id) FROM tip
```

Answer: 537 Records
Key: business_id

ix. *User Table*

```
SELECT count(distinct id) FROM user
```

Answer: 10000 Records
Key: id

x. *Friend Table*

```
SELECT count(distinct user_id) FROM friend
```

Answer: 11 Records
Key: user_id

xi. *Elite_years Table*

```
SELECT count(distinct user_id) FROM elite_years
```

Answer: 2780
Key: user_id

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

   Answer: No

```
SELECT *Column to check* as column_interest
FROM user
WHERE columns_interest is NULL;
```

Interest column names FROM 'user' table to check each column

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

```
SELECT min(stars), max(stars), avg(stars) FROM review
+------------+------------+------------+
| min(stars) | max(stars) | avg(stars) |
+------------+------------+------------+
|          1 |          5 |     3.7082 |
+------------+------------+------------+
```

Output = Min: 1, Max: 5, Avg: 3.7082

ii. Table: Business, Column: Stars

```
SELECT min(stars), max(stars), avg(stars) FROM business
+------------+------------+------------+
| min(stars) | max(stars) | avg(stars) |
+------------+------------+------------+
|        1.0 |        5.0 |     3.6549 |
+------------+------------+------------+
```

Output= Min: 1.0, Max: 5.0, Avg: 3.6549

iii. Table: Tip, Column: Likes

```
SELECT min(likes), max(likes), avg(likes) FROM tip
+------------+------------+------------+
| min(likes) | max(likes) | avg(likes) |
+------------+------------+------------+
|          0 |          2 |     0.0144 |
+------------+------------+------------+
```

Output = Min: 0, Max: 2, Avg: 0.0144

iv. Table: Checkin, Column: Count

```
SELECT min(count), max(count), avg(count) FROM checkin
+------------+------------+------------+
| min(count) | max(count) | avg(count) |
+------------+------------+------------+
|          1 |         53 |     1.9414 |
+------------+------------+------------+
```

Output= Min: 1, Max: 53, Avg: 1.9414

v. Table: User, Column: Review_count

```
SELECT min(review_count), max(review_count), avg(review_count)
FROM user
+-------------------+-------------------+-------------------+
| min(review_count) | max(review_count) | avg(review_count) |
+-------------------+-------------------+-------------------+
|                 0 |              2000 |           24.2995 |
+-------------------+-------------------+-------------------+
```

Output = 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 sum(review_count) AS total_reviews, city
FROM business
GROUP BY city
ORDER BY total_reviews DESC
```

   Copy and Paste the Result Below:

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

```
SELECT stars as star_rating_dist, count(stars) as count
FROM business
WHERE city='Avon'
GROUP BY stars
```

Copy and Paste the Resulting Table Below (2 columns – star rating and count):

```
+------------------+-------+
| star_rating_dist | count |
+------------------+-------+
|              1.5 |     1 |
|              2.5 |     2 |
|              3.5 |     3 |
|              4.0 |     2 |
|              4.5 |     1 |
|              5.0 |     1 |
+------------------+-------+
```

   ii. Beachwood

SQL code used to arrive at answer:

```
SELECT stars as star_rating_dist, count(stars) as count
FROM business
WHERE city='Beachwood'
GROUP BY stars
```

Copy and Paste the Resulting Table Below (2 columns – star rating and count):

```
+------------------+-------+
| star_rating_dist | count |
+------------------+-------+
|              2.0 |     1 |
|              2.5 |     1 |
|              3.0 |     2 |
|              3.5 |     2 |
|              4.0 |     1 |
|              4.5 |     2 |
|              5.0 |     5 |
+------------------+-------+
```

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

   SQL code used to arrive at answer:

```
SELECT id, name, sum(review_count) as total_count
FROM user
GROUP BY id
ORDER BY total_count desc
LIMIT 3
```

   Copy and Paste the Result Below:

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

8. Does posing more reviews correlate with more fans?

   Please explain your findings and interpretation of the results:

   - Certainly! The data suggests a clear correlation between the number of reviews posted and the number of fans, indicating that users with higher review counts tend to have more fans.

   Code:

```
SELECT range AS range_fans,
       COUNT(*) AS num_user,
       AVG(review_count) AS avg_num_review,
       AVG(fans) AS avg_num_fans
FROM (SELECT CASE WHEN fans BETWEEN 0 AND 9 THEN '0 - 9'
                  WHEN fans BETWEEN 10 AND 99 THEN '10 - 99'
                  ELSE '100-1000' END AS range,
             review_count,
             fans
      FROM user) AS subtable
GROUP BY subtable.range
```

```
+------------+----------+--------------------+--------------------+
| range_fans | num_user |     avg_num_review |       avg_num_fans |
+------------+----------+--------------------+--------------------+
| 0 - 9      |     9690 | 15.008565531475748 |  0.447265221878225 |
| 10 - 99    |      294 |  283.3265306122449 | 25.598639455782312 |
| 100-1000   |       16 |              891.5 |             189.75 |
+------------+----------+--------------------+--------------------+
```

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

   Answer:

   - The analysis indicates that there are significantly more reviews containing the word "love" (1780 occurrences) compared to those containing "hate" (232 occurrences), suggesting a prevalent positive sentiment among users' reviews.

   SQL code used to arrive at answer:

   Check for love:

```
SELECT COUNT(text)
FROM review
WHERE text LIKE '%love%';
+-------------+
| COUNT(text) |
+-------------+
|        1780 |
+-------------+
```

   Check for hate:

```
SELECT COUNT(text)

FROM review
WHERE text LIKE '%hate%';
+-------------+

| COUNT(text) |
+-------------+
|         232 |
+-------------+
```

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

    SQL code used to arrive at answer:

```
SELECT name, fans as num_fans

FROM user
ORDER BY fans desc
LIMIT 10
```

    Copy and Paste the Result Below:

```
+-----------+----------+
| name      | num_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

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

    - I also chose Las Vegas and Restaurant; They have reviews with 2-3 stars, 3-4 stars and 4-5 stars. Businesses with 3-4 stars are open more of the time.

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

    - The Restaurants in Las Vegas have a total of 1,062 reviews (3-4 stars have 1,059 reviews, 4-5 stars have 3 reviews)

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

    - We can infer that the restaurant scene in Las Vegas, particularly within the 'Restaurants' category, is quite competitive and generally maintains high standards in terms of star ratings and review counts. The relatively high average star rating and review count suggest that these restaurants are likely popular among customers and may offer a satisfactory dining experience.

SQL code used for analysis:

1i:

```
SELECT CASE WHEN stars > 4.0 THEN '4-5 stars'
            WHEN stars > 3.0 THEN '3-4 stars'
            WHEN stars > 2.0 THEN '2-3 stars'
            ELSE 'below 2' END AS 'STAR',
       COUNT(DISTINCT b.id) AS count,
       COUNT(hours) AS open_days_total,
       COUNT(hours) / COUNT(DISTINCT b.id)  AS open_days_avg
  FROM business b
  INNER JOIN hours h
  ON b.id = h.business_id
  WHERE city = 'Las Vegas'
 GROUP BY STAR
```

```
+-----------+-------+-----------------+---------------+
| STAR      | count | open_days_total | open_days_avg |
+-----------+-------+-----------------+---------------+
| 2-3 stars |     2 |              14 |             7 |
| 3-4 stars |     8 |              50 |             6 |
| 4-5 stars |     3 |              18 |             6 |
+-----------+-------+-----------------+---------------+
```

1ii:

```
SELECT business.city,
       category.category,
       business.name,
       business.stars,
       business.review_count
FROM business
INNER JOIN category ON business.id = category.business_id
WHERE business.city = 'Las Vegas' AND category.category ='Restaurants'
ORDER BY business.stars;
```

```
+-----------+-------------+----------------------+-------+--------------+
| city      | category    | name                 | stars | review_count |
+-----------+-------------+----------------------+-------+--------------+
| Las Vegas | Restaurants | Wingstop             |   3.0 |          123 |
| Las Vegas | Restaurants | Big Wong Restaurant  |   4.0 |          768 |
| Las Vegas | Restaurants | Jacques Cafe         |   4.0 |          168 |
| Las Vegas | Restaurants | Hibachi-San          |   4.5 |            3 |
+-----------+-------------+----------------------+-------+--------------+
```

1iii.

```
SELECT b.city,
       c.category,
       COUNT(*) AS 'Total Businesses',
       AVG(b.stars) AS 'Average Star Rating',
       MIN(b.stars) AS 'Minimum Star Rating',
       MAX(b.stars) AS 'Maximum Star Rating',
       AVG(b.review_count) AS 'Average Review Count'
FROM business b
JOIN category c ON b.id = c.business_id
WHERE b.city = 'Las Vegas' AND c.category = 'Restaurants'
GROUP BY b.city, c.category;
```

```
+-----------+-------------+-----------------+---------------------+---------------------+---------------------+---------------------+
| city      | category    | Total Businesses | Average Star Rating | Minimum Star Rating | Maximum Star Rating | Average Review Count |
+-----------+-------------+-----------------+---------------------+---------------------+---------------------+---------------------+
| Las Vegas | Restaurants |               4 |               3.875 |                 3.0 |                 4.5 |                 265.5 |
+-----------+-------------+-----------------+---------------------+---------------------+---------------------+---------------------+
```

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:
The businesses that are open outnumber the businesses that are closed.

ii. Difference 2:
It is very interesting how the mean rating (average number of stars) is lower for the open businesses as opposed to the closed.

SQL code used for analysis:

i & ii -

```
SELECT is_open,
       count(distinct b.id) count_business,
       count(distinct r.id) count_review,
      avg(r.stars) AS mean_rating
FROM business b
JOIN review r ON b.id =  r.business_id
WHERE city = 'Las Vegas'
GROUP BY is_open
```

```
+---------+----------------+--------------+--------------------+
| is_open | count_business | count_review |        mean_rating |
+---------+----------------+--------------+--------------------+
|       0 |             20 |           28 | 3.9642857142857144 |
|       1 |            112 |          165 | 3.8545454545454545 |
+---------+----------------+--------------+--------------------+
```