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

Orkun T. Aran

Part 1: Yelp Dataset Profiling and Understanding

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

Query: SELECT COUNT(*) FROM table_name

I didn't want to write every query to provide a cleaner report. You'll see answers below for the question 1.

i. Attribute table = 10000
ii. Business table = 10000
iii. Category table = 10000
iv. Checkin table = 10000
v. elite_years table = 10000
vi. friend table = 10000
vii. hours table = 10000
viii. photo table = 10000
ix. review table = 10000
x. tip table = 10000
xi. user table = 10000

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

```
-- Finds the distinct count of primary key
SELECT COUNT(DISTINCT(id)) FROM business
```

ii. Hours = 1562

```
-- Finds the distinct count of primary key
SELECT COUNT(DISTINCT(business_id)) FROM hours
```

iii. Category = 2643

```
-- Finds the distinct count of primary key
SELECT COUNT(DISTINCT(business_id)) FROM category
```

iv. Attribute = 1115

```
-- Finds the distinct count of primary key
SELECT COUNT(DISTINCT(business_id)) FROM attribute
```

v. Review = 9581

```
-- Finds the distinct count of foreign key- user_id
SELECT COUNT(DISTINCT(user_id)) FROM review
```

vi. Checkin = 493

```
-- Finds the distinct count of foreign key – business_id
SELECT COUNT(DISTINCT(business_id)) FROM Checkin
```

vii. Photo = 6493

```
-- Finds the distinct count of foreign key – business_id
SELECT COUNT(DISTINCT(business_id)) FROM Photo
```

viii. Tip = 3979

```
-- Finds the distinct count of foreign key – business_id
SELECT COUNT(DISTINCT(business_id)) FROM tip
```

ix. User = 10000

```
-- Finds the distinct count of primary key
SELECT COUNT(DISTINCT(id)) FROM user
```

x. Friend = 11

```
-- Finds the distinct count of foreign key – user_id
SELECT COUNT(DISTINCT(user_id)) FROM friend
```

xi. Elite_years = 2780

```
-- Finds the distinct count of foreign key – elite_years
SELECT COUNT(DISTINCT(user_id)) FROM elite_years
```

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:

```
/* This query checks each column in USER table wheter it has a NULL or not.
If a NULL found returns 1, else returns 0
*/
```

```
SELECT
   SUM(case when id is null then 1 else 0 end) as id,
   SUM(case when name is null then 1 else 0 end) as name,
   SUM(case when review_count is null then 1 else 0 end) as review_count,
   SUM(case when yelping_since is null then 1 else 0 end) as yelping_since,
   SUM(case when useful is null then 1 else 0 end) as useful,
   SUM(case when cool is null then 1 else 0 end) as cool,
   SUM(case when fans is null then 1 else 0 end) as fans,
   SUM(case when average_Stars is null then 1 else 0 end) as average_Stars,
   SUM(case when compliment_hot is null then 1 else 0 end) as compliment_hot,
   SUM(case when compliment_more is null then 1 else 0 end) as compliment_more,
   SUM(case when compliment_profile is null then 1 else 0 end) as compliment_profile,
   SUM(case when compliment_cute is null then 1 else 0 end) as compliment_cute,
   SUM(case when compliment_list is null then 1 else 0 end) as compliment_list,
   SUM(case when compliment_note is null then 1 else 0 end) as compliment_note,
   SUM(case when compliment_plain is null then 1 else 0 end) as compliment_plain,
   SUM(case when compliment_cool is null then 1 else 0 end) as compliment_cool,
   SUM(case when compliment_funny is null then 1 else 0 end) as compliment_funny,
   SUM(case when compliment_writer is null then 1 else 0 end) as compliment_writer,
   SUM(case when compliment_photos is null then 1 else 0 end) as compliment_photos
FROM user
```

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

```
-- A query to return min,max and average of given column of table
SELECT MIN(column),
          MAX(column),
          AVG(column)
FROM table
```

i. Table: Review, Column: Stars

min: 1 max: 5 avg: 3.70

ii. Table: Business, Column: Stars

min: 1 max: 5 avg: 3.65

iii. Table: Tip, Column: Likes

min: 0 max: 2 avg: 0.014

iv. Table: Checkin, Column: Count

min: 1 max: 53 avg: 1.94

v. Table: User, Column: Review_count

min: 0 max: 2000 avg: 24.29

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

SQL code used to arrive at answer:

```
-- City and review count in descending order
SELECT city, SUM(review_count) AS total_reviews
FROM business
GROUP BY city
ORDER BY total_reviews DESC
```

Copy and Paste the Result Below:

```
+-----------------+---------------+
| city            | total_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
Markham	2352
Champaign	2029
Stuttgart	1849
Surprise	1520
Lakewood	1465
Goodyear	1155

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

i. Avon

SQL code used to arrive at answer:

```
-- Star ratings for AVON
SELECT stars,
   SUM(review_count) AS count
FROM business
WHERE city = 'Avon'
GROUP BY stars
```

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

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

+-------+-------+

ii. Beachwood

SQL code used to arrive at answer:

```
-- Star ratings for Beachwood
SELECT stars,
    SUM(review_count) AS count
FROM business
WHERE city = 'Beachwood'
GROUP BY stars
```

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

stars	count
2.0	8
2.5	3
3.0	11
3.5	6
4.0	69
4.5	17
5.0	23

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

SQL code used to arrive at answer:

```
-- Top 3 users based on review count
SELECT id, name, SUM(review_count) AS total_reviews FROM user
GROUP BY id
ORDER BY total_reviews DESC
```

```
LIMIT 3
```

Copy and Paste the Result Below:

```
+------------------------+--------+---------------+
| id                     | name   | total_reviews |
+------------------------+--------+---------------+
| -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:

Posting more reviews doesn't seem to have a relation with more fans. The highest fan size is 503 and its total review count is 609, which no way near maximum review count 2000. A query which gives review count and fans, ordered by fans shows us the review count received by highest fan base. This table is represented below. We can see from the table review count is primarily not affected by fan base.

```
+------+--------------+
| fans | review_count |
+------+--------------+
|  503 |          609 |
|  497 |          968 |
|  311 |         1153 |
|  253 |         2000 |
|  173 |          930 |
|  159 |          813 |
|  133 |          377 |
|  126 |         1215 |
|  124 |          862 |
|  120 |          834 |
|  115 |          861 |
|  111 |          408 |
|  105 |          255 |
|  104 |         1039 |
|  101 |          694 |
|  101 |         1246 |
|   96 |          307 |
|   89 |          584 |
|   85 |          842 |
|   84 |          220 |
|   81 |          408 |
|   80 |          178 |
```

```
|  78 |        754 |
|  76 |       1339 |
|  73 |        161 |
+------+--------------+
```

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

Answer:

More reviews with ‘Love’ .

SQL code used to arrive at answer:

```
/* Select with-in select statement to compare
'love' and 'hate' words in reviews
*/
SELECT
((SELECT COUNT(text) FROM review
WHERE text LIKE '%love%')) AS love,
(SELECT COUNT(text) FROM review
WHERE text LIKE '%hate%')  AS hate
FROM review
LIMIT 1
```

```
+------+------+
| love | hate |
+------+------+
| 1780 |  232 |
+------+------+
```

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

SQL code used to arrive at answer:

```
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 would like to divide my intuition into steps for explanation. First of all, after examining category and business tables, I chose 'Bars' category. Following code is to examine categories

```
/*
See the total number of categories
*/

SELECT c.category, COUNT((c.category)) FROM business b
JOIN category c ON c.business_id = b.id
GROUP BY c.category
ORDER BY COUNT((c.category)) DESC

----------
/*
Now examine Bars category. The question asks us to pick one of the category or city; I chose category = bars. And investigate hours and stars of bars
```

```
*/

select  h.hours  from category c
JOIN business b ON c.business_id = b.id
JOIN hours h ON c.business_id = h.business_id
WHERE c.category = 'Bars'
ORDER BY stars
```

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

Yes. The higher the stars, the shorter the bars work.

```
/*
Now examine  hours category
*/
select  DISTINCT h.hours  from category c
JOIN business b ON c.business_id = b.id
JOIN hours h ON c.business_id = h.business_id
WHERE c.category = 'Bars' AND stars BETWEEN 4 AND 5
ORDER BY hours DESC;



select  DISTINCT h.hours  from category c
JOIN business b ON c.business_id = b.id
JOIN hours h ON c.business_id = h.business_id
WHERE c.category = 'Bars' AND stars BETWEEN 2 AND 3
ORDER BY hours DESC;
```

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

Yes. Bars with higher stars have higher reviews.

```
/*
Review count comparison between >4 and
*/
SELECT
(SELECT SUM(review_count) FROM category c
JOIN business b ON c.business_id = b.id
```

```
WHERE c.category = 'Bars' AND stars >=4.0) AS high_stars,
(SELECT SUM(review_count) FROM category c
JOIN business b ON c.business_id = b.id
WHERE c.category = 'Bars' AND stars BETWEEN 2 AND 3) AS low_stars
FROM category c
JOIN business b ON c.business_id = b.id
WHERE category = 'Bars'
oRDER BY stars DESC
LIMIT 1;
+------------+-----------+
| high_stars | low_stars |
+------------+-----------+
|        917 |       290 |
+------------+-----------+
```

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

There is no information could be gathered by locations of the bars.

SQL code used for analysis:

```
/*
Interpretation by location
*/
SELECT *
FROM category c
JOIN business b ON c.business_id = b.id
WHERE category = 'Bars'
ORDER BY stars;
```

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 open bars have higher reviews in total.

ii. Difference 2:

The open bars have slightly higher overall – average stars compared to closed ones.

SQL code used for analysis:

```
SELECT COUNT(DISTINCT(id)),
        AVG(review_count),
        SUM(review_count),
        AVG(stars),
        is_open
    FROM business
    GROUP BY is_open;
```

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:

Clustering users to find type of users to determine their behavior types.

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

Types of users are really important in terms of reviewing businesses. Their kind or toxic behaviors may change the potential customers for each business. Also, on the other side it may be important to determine what kind of services each use like or dislike, and for future Yelp may suggest businesses to users by their likings.

To interfere this analysis, I could use any kind of data that comes from the users. I will merge user and review tables.

iii. Output of your finished dataset:

NOTE: To fit the tables, I split the data into multiple parts while copy-pasting. Also limited the total output to 5. The database has 97 records, and also the 'text' takes too much space here, and also it is limited to 25 characters with SUBSTR(text,0,25).

```
+----------------------+----------+-------------+--------------------+
| id                   | name     | review_count | yelping_since      |
+----------------------+----------+-------------+--------------------+
| --i0PK1aTXScdV2UkNDkSQ | A'Starra  |          9 | 2014-11-09 00:00:00 |
| --Qh8yKWAvIP4V4K8ZPfHA | Dixie    |        503 | 2011-01-19 00:00:00 |
| -0DgO-WJ7yBjAihY_PoUpw | Tonia    |          1 | 2016-10-24 00:00:00 |
| -0oUqPRPpbi2MyiK39cCTg | soragamii |         26 | 2009-07-28 00:00:00 |
| -0udWcFQEt2M8kM3xcIofw | Kaitlan  |        235 | 2015-05-01 00:00:00 |
+----------------------+----------+-------------+--------------------+

+--------+-------+------+------+--------------+
| useful | funny | cool | fans | average_stars |
+--------+-------+------+------+--------------+
|      1 |     0 |    0 |    0 |         3.22 |
|     21 |    32 |   23 |   41 |         3.19 |
|      0 |     0 |    0 |    0 |          1.0 |
|      1 |     0 |    0 |    1 |          3.5 |
|     30 |     4 |   10 |    3 |         3.92 |
+--------+-------+------+------+--------------+

+---------------+----------------+------------------+----------------+----------------+
| compliment_hot | compliment_more | compliment_profile | compliment_cute | compliment_list |
+---------------+----------------+------------------+----------------+----------------+
|             0 |              0 |                0 |              0 |              0 |
|            32 |              3 |                2 |              4 |              1 |
|             0 |              0 |                0 |              0 |              0 |
|             2 |              0 |                0 |              0 |              0 |
|             1 |              0 |                0 |              1 |              0 |
+---------------+----------------+------------------+----------------+----------------+



+----------------+-----------------+----------------+-----------------+
| compliment_note | compliment_plain | compliment_cool | compliment_funny |
+----------------+-----------------+----------------+-----------------+
|              0 |               0 |              0 |               0 |
|             49 |              92 |             67 |              67 |
|              0 |               0 |              0 |               0 |
|              0 |               3 |              2 |               2 |
|              4 |               3 |              2 |               2 |
+----------------+-----------------+----------------+-----------------+
+------------------+------------------+----------------------+-------+--------------------+
| compliment_writer | compliment_photos | review_id            | stars | review_date        |
```

```
+------------------+------------------+----------------------+-------+---------------------+
|              0 |              1 | -1QoNm-2j_YVxUfcSxKbyg |     4 | 2015-11-12 00:00:00 |
|             34 |              6 | -5mrWdN6NoBQN0ifLdzKBg |     4 | 2017-01-09 00:00:00 |
|              0 |              0 | -6YDCouxgOc9lLK43PUupA |     1 | 2017-04-10 00:00:00 |
|              0 |              0 | -4bZNsu9p_Ra2P0tTo-VZQ |     3 | 2011-08-21 00:00:00 |
|              2 |              0 | -1ihUXLbluxAhdWK1C3OUg |     3 | 2016-08-30 00:00:00 |
+------------------+------------------+----------------------+-------+---------------------+

+---------------------+--------------+-------------+-------------+
| SUBSTR(r.text, 0,20) | useful_review | funny_review | cool_review |
+---------------------+--------------+-------------+-------------+
| Came here on a trip  |           0 |          0 |          0 |
| This is a very beau  |           6 |          3 |          6 |
| Hired based on revi  |           1 |          0 |          0 |
| The DJ was awesome   |           0 |          0 |          0 |
| My pedicure was 4 s  |           1 |          0 |          0 |
+---------------------+--------------+-------------+-------------+
```

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

```
SELECT u.id
    , u.name
    , u.review_count
    , u.yelping_since
    , u.useful
    , u.funny
    , u.cool
    , u.fans
    , u. average_stars
    , u.compliment_hot
    , u.compliment_more
    , u.compliment_profile
    , u.compliment_cute
    , compliment_list
    , u.compliment_note
    , u.compliment_plain
    , u.compliment_cool
    , u.compliment_funny
    , u.compliment_writer
    , u.compliment_photos
    , r.id AS review_id
```

```
, r.stars
, r.date AS review_date
, r.text
, r.useful AS useful_review
, r.funny AS funny_review
, r. cool  AS cool_review
FROM user u
JOIN review r ON r.user_id = u.id
```