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 = 10,000
ii. Business table = 10,000
iii. Category table = 10,000
iv. Checkin table = 10,000
v. elite_years = 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

CODE:

```
SELECT COUNT (*) AS Total
FROM attribute
SELECT COUNT (*) AS Total
FROM business
SELECT COUNT (*) AS Total
FROM category
SELECT COUNT (*) AS Total
FROM checkin
SELECT COUNT (*) AS Total
FROM elite_years
SELECT COUNT (*) AS Total
FROM friend
SELECT COUNT (*) AS Total
FROM hours
SELECT COUNT (*) AS Total
FROM photo
SELECT COUNT (*) AS Total
FROM review
SELECT COUNT (*) AS Total
FROM tip
SELECT COUNT (*) AS Total
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.

i. Attribute table = 1,115
ii. Business table = 10,000
iii. Category table = 2,643
iv. Checkin table = 493

v. elite_years = 2,780
vi. friend table = 11
vii. hours table = 1,562
viii. photo table = 10,000
ix. review table = 10,000
x. tip table = 537 (user_id), 3,979 (business_id)
xi. user table = 10,000

CODE:

```
SELECT COUNT(DISTINCT id) AS Total
FROM business
SELECT COUNT(DISTINCT business_id) AS Total
FROM hours
SELECT COUNT(DISTINCT business_id) AS Total
FROM category
SELECT COUNT(DISTINCT business_id) AS Total
FROM attribute
SELECT COUNT(DISTINCT id) AS Total
FROM review
SELECT COUNT(DISTINCT business_id) AS Total
FROM checkin
SELECT COUNT(DISTINCT id) AS Total
FROM photo
SELECT COUNT(DISTINCT user_id) AS Total
FROM tip
SELECT COUNT(DISTINCT business_id) AS Total
FROM tip
SELECT COUNT(DISTINCT id) AS Total
FROM user
SELECT COUNT(DISTINCT user_id) AS Total
FROM friend
SELECT COUNT(DISTINCT user_id) AS Total
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:

```
SELECT *
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;
```

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

min_stars	max_stars	avg_stars
1	5	3.7082

ii. Table: Business, Column: Stars

min_stars	max_stars	avg_stars
1.0	5.0	3.6549

iii. Table: Tip, Column: Likes

min_likes	max_likes	avg_likes
0	2	0.0144

iv. Table: Checkin, Column: Count

min_count	max_count	avg_count
1	53	1.9414

v. Table: User, Column: review_count

min_review_count	max_review_count	avg_review_count
0	2000	24.2995

CODE:

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

```
SELECT MIN(stars) AS min_stars, MAX(stars) AS max_stars, AVG(stars) AS avg_stars
FROM business
SELECT MIN(likes) AS min_likes, MAX(likes) AS max_likes, AVG(likes) AS avg_likes
FROM tip
SELECT MIN(count) AS min_count, MAX(count) AS max_count, AVG(count) AS avg_count
FROM checkin
SELECT MIN(review_count) AS min_review_count,
MAX(review_count) AS max_review_count,
AVG(review_count) AS avg_review_count
FROM user
```

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

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

(Output limit exceeded, 25 of 362 total rows shown)

Code:

```
SELECT city, SUM(review_count) AS total_reviews
FROM business
GROUP BY city
ORDER BY total_reviews DESC
```

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

result:

city	stars	count

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

code:

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

ii. Beachwood

result:

```
+-----------+-------+-------+
| city      | stars | count |
+-----------+-------+-------+
| Beachwood |   5.0 |     5 |
| Beachwood |   3.0 |     2 |
| Beachwood |   3.5 |     2 |
| Beachwood |   4.5 |     2 |
| Beachwood |   2.0 |     1 |
| Beachwood |   2.5 |     1 |
| Beachwood |   4.0 |     1 |
+-----------+-------+-------+
```

code:

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

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

Result:

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

Code:

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

```
ORDER BY review_count DESC
LIMIT 3;
```

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

Result:

```
+------+-----------+
| name | MAX(fans) |
+------+-----------+
| Amy  |       503 |
+------+-----------+
```

```
+--------+-------------------+
| name   | MAX(review_count) |
+--------+-------------------+
| Gerald |              2000 |
+--------+-------------------+
```

Users with most number of reviews:

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

Users with the most number of fans:

```
+-----------+--------------+------+
| name      | review_count | fans |
+-----------+--------------+------+
| Amy       |          609 |  503 |
| Mimi      |          968 |  497 |
| Harald    |         1153 |  311 |
| Gerald    |         2000 |  253 |
| Christine |          930 |  173 |
| Lisa      |          813 |  159 |
| Cat       |          377 |  133 |
| William   |         1215 |  126 |
| Fran      |          862 |  124 |
| Lissa     |          834 |  120 |
+-----------+--------------+------+
```

```
+-----------------+
|    correlation |
+-----------------+
| 0.437136492915 |
+-----------------+
```

Interpretation: The correlation between the two variable is moderate with a correlation of 0.4

Code:

```
SELECT name, MAX(fans)
FROM user
SELECT name, MAX(review_count)
SELECT name, review_count, fans
FROM user
ORDER BY review_count DESC
LIMIT 10;

select avg( (review_count - avg_x) * (fans - avg_y) )*avg( (review_count - avg_x) * (fans -
avg_y) )/(var_x*var_y) as correlation
from user, (select
      avg_x,
      avg_y,
      avg((review_count - avg_x)*(review_count - avg_x)) as var_x,
      avg((fans - avg_y)*(fans - avg_y)) as var_y
      from user, (select
          avg(review_count) as avg_x,
          avg(fans) as avg_y
          from user)
  );
```

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

result:

```
+----------------+
| mentioned_love |
+----------------+
|           1780 |
+----------------+
+----------------+
| mentioned_hate |
+----------------+
|            232 |
+----------------+
```

There are more reviews that mentioned love than hate

Code:

```
SELECT COUNT(*) AS mentioned_love
FROM review
WHERE text LIKE '%love%';
SELECT COUNT(*) AS mentioned_hate
FROM review
WHERE text LIKE '%hate%';
```

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

Result:

```
+-----------+------+
| name      | fans |
+-----------+------+
| Amy       |  503 |
| Mimi      |  497 |
```

Harald	311
Gerald	253
Christine	173
Lisa	159
Cat	133
William	126
Fran	124
Lissa	120

Code:

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

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 picked Las Vegas

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

They have similar distribution of hours but not exactly the same.

ii. Do the two groups you chose to analyze have a different number of reviews?
Yes, there are more 4-5 stars reviews compared to 2-3 stars reviews

iii. Are you able to infer anything from the location data provided between these two groups? Explain.
Based on the table shown below, most of the 4-5 stars reviews are located in the northwestern neighborhood of Las Vegas, while 2-3 star reviews tend to be located in the Southern or mid Las Vegas area. Higher reviews may be given to businesses due to the location of the business.

2-3 stars by neighborhood

neighborhood	num_of_neighborhood	stars_category
Eastside	49	2-3
Southwest	28	2-3
	21	2-3
Southeast	12	2-3

4-5 stars by neighborhood

neighborhood	num_of_neighborhood	stars_category
	131	4-5
Summerlin	61	4-5
Chinatown	49	4-5

```
| Southeast     |                  30 | 4-5            |
| Centennial    |                  21 | 4-5            |
| Spring Valley |                  18 | 4-5            |
+---------------+---------------------+----------------+
```

Code:

2-3 stars hours of operation

```
SELECT b.name, b.city, b.stars,
c.category, h.hours, COUNT (h.hours) AS hours_freq,
CASE WHEN b.stars BETWEEN 2.0 AND 3.9 THEN '2-3'
    WHEN b.stars BETWEEN 4.0 AND 5.0 THEN '4-5'
    ELSE 'Other'
    END stars_category
FROM business b
LEFT JOIN category c ON b.id=c.business_id
LEFT JOIN hours h ON c.business_id=h.business_id
WHERE city = 'Las Vegas' AND stars_category = '2-3' AND hours <> 'None'
GROUP BY hours
ORDER BY hours_freq DESC;
```

4-5 stars hours of operation

```
SELECT b.name, b.city, b.stars,
c.category, h.hours, COUNT (h.hours) AS hours_freq,
CASE WHEN b.stars BETWEEN 2.0 AND 3.9 THEN '2-3'
    WHEN b.stars BETWEEN 4.0 AND 5.0 THEN '4-5'
    ELSE 'Other'
    END stars_category
FROM business b
LEFT JOIN category c ON b.id=c.business_id
LEFT JOIN hours h ON c.business_id=h.business_id
WHERE city = 'Las Vegas' AND stars_category = '4-5' AND hours <> 'None'
GROUP BY hours
ORDER BY hours_freq DESC;
```

2-3 stars neighborhood FULL TABLE

```
SELECT b.name, b.city, b.neighborhood,
COUNT(b.neighborhood) AS num_of_neighborhood,
c.category, h.hours, COUNT (h.hours) AS hours_freq,
CASE WHEN b.stars BETWEEN 2.0 AND 3.9 THEN '2-3'
    WHEN b.stars BETWEEN 4.0 AND 5.0 THEN '4-5'
    ELSE 'Other'
    END stars_category
FROM business b
LEFT JOIN category c ON b.id=c.business_id
LEFT JOIN hours h ON c.business_id=h.business_id
WHERE city = 'Las Vegas' AND stars_category = '2-3' AND hours <> 'None'
GROUP BY neighborhood
ORDER BY num_of_neighborhood DESC;
```

2-3 stars by neighborhood code

```
SELECT b.neighborhood,
COUNT(b.neighborhood) AS num_of_neighborhood,
CASE WHEN b.stars BETWEEN 2.0 AND 3.9 THEN '2-3'
    WHEN b.stars BETWEEN 4.0 AND 5.0 THEN '4-5'
    ELSE 'Other'
    END stars_category
FROM business b
LEFT JOIN category c ON b.id=c.business_id
LEFT JOIN hours h ON c.business_id=h.business_id
WHERE city = 'Las Vegas' AND stars_category = '2-3' AND hours <> 'None'
GROUP BY neighborhood
ORDER BY num_of_neighborhood DESC;
```

4-5 stars by neighborhood code

```
SELECT b.neighborhood,
COUNT(b.neighborhood) AS num_of_neighborhood,
CASE WHEN b.stars BETWEEN 2.0 AND 3.9 THEN '2-3'
    WHEN b.stars BETWEEN 4.0 AND 5.0 THEN '4-5'
    ELSE 'Other'
    END stars_category
FROM business b
LEFT JOIN category c ON b.id=c.business_id
LEFT JOIN hours h ON c.business_id=h.business_id
WHERE city = 'Las Vegas' AND stars_category = '4-5' AND hours <> 'None'
GROUP BY neighborhood
ORDER BY num_of_neighborhood DESC;
```

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 use to arrive at your answer.

i. Difference 1: businesses that are open have more number of reviews than the ones that are closed, this may be due to open businesses are still receiving reviews, while closed businesses are no longer receiving reviews.

ii. Difference 2: businesses that are open have a higher percentage of 4-5 star reviews compared to their 2-3 stars counterpart, this may signify that businesses that receive higher reviews receive more regular customers, or potential customers may have seen the review and choose to patron businesses with better reviews.

Businesses that are open

status	stars_category	Total
open	2-3	467
open	4-5	724
open	Other	46

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

Businesses that are closed

```
+--------+----------------+-------+
| status | stars_category | Total |
+--------+----------------+-------+
| closed | 2-3            |   141 |
| closed | 4-5            |    95 |
| closed | Other          |    10 |
+--------+----------------+-------+
```

Code used:

```
SELECT status, stars_category,
COUNT (Name) Total
FROM
(SELECT b.name, b.city, b.neighborhood,
c.category, h.hours,
CASE WHEN b.is_open == 1 THEN 'open'
    WHEN b.is_open == 0 THEN 'closed'
    ELSE 'na'
    END status,
CASE WHEN b.stars BETWEEN 2.0 AND 3.9 THEN '2-3'
    WHEN b.stars BETWEEN 4.0 AND 5.0 THEN '4-5'
    ELSE 'Other'
    END stars_category
FROM business b
LEFT JOIN category c ON b.id=c.business_id
LEFT JOIN hours h ON c.business_id=h.business_id
WHERE city = 'Las Vegas' AND status = 'closed'
GROUP BY b.name)
GROUP BY stars_category;
```

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 will be looking for the best state for a shopping destination by looking at the frequencies of shopping places and the number of 4-5 star reviews

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

Data types I will need include string characters of shopping areas, the state, city it is located, and integers of the reviews. I chose this as this will make it possible to calculate the number of frequencies a shopping place shows up in the data.

iii. Output of your finished dataset:
based on the output of this dataset, shoppers should look for AZ as their next shopping destination with most 4-5 star reviews and the most number of shopping centers, and avoid ON as they have many shopping centers but more undesirable reviews.

city	state	total_shop	category	stars_category
Scottsdale	AZ	11	Shopping	4-5
Las Vegas	NV	4	Shopping	4-5
Strongsville	OH	2	Shopping	4-5
Pittsburgh	PA	1	Shopping	4-5
Middleton	WI	1	Shopping	4-5
Stuttgart	BW	1	Shopping	2-3
Edinburgh	EDH	1	Shopping	2-3
Charlotte	NC	3	Shopping	2-3
Mississauga	ON	6	Shopping	2-3

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

```sql
SELECT b.city, b.state, COUNT (state) AS total_shop,
c.category,
CASE WHEN b.stars BETWEEN 2.0 AND 3.9 THEN '2-3'
    WHEN b.stars BETWEEN 4.0 AND 5.0 THEN '4-5'
    ELSE 'Other'
    END stars_category
FROM business b
LEFT JOIN category c ON b.id=c.business_id
LEFT JOIN review r ON b.id=r.id
WHERE category = 'Shopping'
GROUP BY state
ORDER BY stars_category DESC;
```