Final Project

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

Submission: Ronobir Das

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 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.

- i. Business = id(PK), 10000
- ii. Hours = business_id(FK), 1562
- iii. Category = business_id(FK), 2643
- iv. Attribute = business_id(FK), 1115
- v. Review =

Attribute Name	Key Type	Distinct Records
id	PK	10,000
business_id	FK	8090
user_id	FK	9581

- vi. Checkin = business_id(PK/FK), 493
- vii. Photo =

Attribute Name	Key Type	Distinct Records
id	PK	10,000
business_id	FK	6493

- viii. Tip =

Attribute Name	Key Type	Distinct Records
business_id	FK	3979
user_id	FK	537

- ix. User = id(PK), 10,000
- x. Friend = user_id(fk), 11
- xi. Elite_years = user_id(FK), 2780

Note: Primary Keys are denoted in the ER-Diagram with a yellow key icon.

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

Answer: No there are no null values.

SQL code used to arrive at answer:

```
SELECT
  Count(*) as NullCount
From
  user
WHERE
  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: 1  max: 5  avg: 3.7802
```

ii. Table: Business, Column: Stars

```
    min: 1.0 max: 5.0 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 as City, sum(review_count) as "Number of Reviews"
FROM business
GROUP BY city
ORDER BY sum(review_count) desc
```

Copy and Paste the Result Below:

City	Number of Reviews
Las Vegas	82854
Phoenix	34503
Toronto	24113
Scottsdale	20614
Charlotte	12523
Henderson	10871

City	Number of Reviews
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:

```
SELECT stars, count(id) as "Number of Businesses"
FROM business
WHERE city LIKE 'Avon'
GROUP BY stars
ORDER BY stars DESC
```

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

stars	Number of Businesses
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, count(id) as "Number of Businesses"
FROM business
WHERE city LIKE 'Beachwood'
GROUP BY stars
ORDER BY stars DESC
```

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

stars	Number of Businesses
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?

Users with the Most Reviews

SQL Query:

```
    SELECT name, review_count, fans
    FROM user
    ORDER BY review_count desc
```

Results:

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

name	review_count	fans
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

Users with the Most Fans SQL Query:

```
    SELECT name, review_count, fans
    FROM user
    ORDER BY fans desc
```

Results:

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
Mark	861	115
Tiffany	408	111
bernice	255	105
Roanna	1039	104
Angela	694	101
.Hon	1246	101
Ben	307	96
Linda	584	89
Christina	842	85
Jessica	220	84
Greg	408	81
Nieves	178	80
Sui	754	78
Yuri	1339	76
Nicole	161	73

Please explain your findings and interpretation of the results:

Possibly there is a weak correlation that the number of reviews a user posts relates to the number of fans they have in a positive correlation. Without using a correlation calculation and just inspecting the data we can see many cases where having a large number of reviews does not result in a ton of fans. For example eric with 1116 reviews but only 16 fans, or L with 696 reviews and only 10 fans or Crissy with 676 reviews and 25 fans. To interpret the results it makes that with more reviews there will be more users who will see your posts and as a result with more eyes there more chances of getting fans. But there's plenty of cases as highlighted above where just having many reviews does not mean you have that many fans.

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

Answer: There's more reviews with the word Love than Hate. There's 1780 reviews with the word love contained in the review text and 232 that contained the word hate.

SQL code used to arrive at answer:

```
SELECT
    (
        SELECT
            count(id)
        from
            review
        WHERE
            text LIKE '%love%'
    ) as LoveCount,
    (
        SELECT
            count(id)
        from
            review
        WHERE
            text LIKE '%hate%'
    ) as HateCount
```

LoveCount	HateCount
1780	232

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

SQL code used to arrive at answer:

```
SELECT name, fans
FROM user
ORDER BY fans desc
LIMIT 10
```

Copy and Paste the Result Below:

name	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.

City: Toronto **Category:** Restaurants

I chose this city and category combination after analyzing which pair had the most businesses to analyze.

This is the table used for the analysis:

id	name	neighborhood	postal_code	stars	ratingcategory	review_count	Number of Days Open	Number of Hours operational per Day
-n27mJ_jQWGCulukTvg9Mg	Cabin Fever	High Park	M6P 1A6	4.5	high	26	7.0	8.9
0B3W6KxkD3o4W4l6cq735w	Big Smoke Burger	Downtown Core	M4B 2L9	3.0	low	47	7.0	10.1
0e-j5VcEn54EZT-FKCUZdw	Sushi Osaka	Etobicoke	M9A 1C2	4.5	high	8	7.0	11.6
1NyHpXJqSLHnvDCOW0nJDg	Pizzaiolo	Entertainment District	M5H 1X6	3.0	low	34	7.0	15.1
1nTMWMa6v-eBKkPYA3gxkQ	99 Cent Sushi	Downtown Core	M5B 2E5	2.0	low	5	7.0	12.0
37kk0IW6jL7ZlxZF6k2QBg	Edulis	Niagara	M5V	4.0	high	89	5.0	4.8

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

ratingcategory	Number of Days Open per Category	Average Number of Operational Hours per Day per Category	Number of Businesses Per Category
high	6.0	8.4	3
low	7.0	12.4	3

The High Rating Category (4 - 5 Star Businesses) are on average open less than the Low Rating Category (2 - 3 Star Businesses). They are open for less days (High: 6 days, Low: 7 days), and for a shorter number of operating hours per day (High: 8.4 hours vs Low: 12.4 hours)

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

ratingcategory	Number of Businesses Per Category	Average number of reviews per business per category	Total number of reviews per category
high	3	41.0	123
low	3	29.0	86

The High Rating Category has more reviews on average than the Low Rating Category.

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

ratingcategory	neighborhood	postal_code
high	High Park	M6P 1A6
low	Downtown Core	M4B 2L9
high	Etobicoke	M9A 1C2
low	Entertainment District	M5H 1X6
low	Downtown Core	M5B 2E5
high	Niagara	M5V

I didn't feel the need to aggregate in this case because there's too small a dataset and location data can be complicated depending on what parameters are used. But looking at this table, we can see 2 of the Low rated restaurants are located in Downtown Core. Otherwise there's nothing else to be seen from this location data alone. Maybe seeing it on a map would make things clearer but Text based analysis does not make things clear.

SQL code used for analysis:

Base Table used for aggregate analysis:

```sql
SELECT id, name, neighborhood, postal_code, stars,
  CASE
    WHEN stars BETWEEN 2.0 AND 3.0 THEN 'low'
    WHEN stars BETWEEN 4.0 AND 5.0 THEN 'high'
  END AS ratingcategory,
review_count, dayofweek, StartTime, EndTime, workhours
FROM business, category, (SELECT business_id, dayofweek, fStartHour as StartTime, fEndhour as EndTime,
             CASE
                 WHEN (julianday(fEndhour) - julianday(fStartHour)) < 0 THEN (24 - (-1*round(((julianday(fEndhour) - julianday(fStartHour))*24),1)))
                 ELSE round(((julianday(fEndhour) - julianday(fStartHour))*24),1)
             END AS workhours
      FROM
        (SELECT *,
                CASE
                    WHEN cast(substr(starthour, 1, instr(starthour, ':')) AS int) >= 10 THEN starthour
                    ELSE ('0' || starthour)
                END AS fStartHour ,
                CASE
                    WHEN cast(substr(endhour, 1, instr(endhour, ':')) AS int) > 10 THEN endhour
                    ELSE ('0' || endhour)
                END AS fEndhour
         FROM
           (SELECT *,
                   substr(timeperday, instr(timeperday, '-')+1) AS endhour,
                   substr(timeperday, 1, instr(timeperday, '-')-1) AS starthour
            FROM
              (SELECT business_id,
                      substr(hours, instr(hours, '|')+1) AS timeperday,
                      substr(hours, 1, instr(hours, '|')-1) AS dayofweek
               FROM hours
               )))) as schedule
WHERE business.id = category.business_id
AND business.id = schedule.business_id
AND category = 'Restaurants'
AND city = 'Toronto'
```

Base Table Explanation: I chose the city of Toronto and Category of Restaurants so I didn't need them in my analysis as that's already presumed. The base table joins form 3 tables, the business table, the category table and hours table. Because we are focusing on one city category pair, I can remove those columns from the base table. The questions for this section asks for hours, number of reviews and location details. As a result I pick the columns id (the id of the business), name (the name of the business), neighborhood (location information), postal_code (location information), dayoftheweek (Day of the Week extracted from the Hours table), StartTime (when does the business open extracted from the Hours table), EndTime (when does the business close), Workhours (how many hours per scheduled day is the business open extracted from the Hours Table). Finally, we have the Stars column and a Bin column that's formatted using a CASE statement to bin them based on what the question asked, where if a business has a rating between 2 to 3 stars is a bin known as LOW and a business between 4 to 5 is a bin known as HIGH. This computed column is known as rating category.

Base Table Schema:

id	name	neighborhood	postal_code	stars	ratingcategory	review_count	dayofweek	StartTime	EndTime	workhours
0e-j5VcEn54EZT-FKCUZdw	Sushi Osaka	Etobicoke	M9A 1C2	4.5	high	8	Monday	11:00	23:00	12.0

Aggregate Analysis Table: This is an additional aggregation done on the base table, to aggregate the Hours table data.

```
SELECT
id, name, neighborhood, postal_code, stars, ratingcategory,
review_count, count(dayofweek) as "Number of Days Open",
sum(workhours)/count(dayofweek) as "Number of Hours operational per Day"
FROM
(
SELECT id, name, neighborhood, postal_code, stars,
  CASE
    WHEN stars BETWEEN 2.0 AND 3.0 THEN 'low'
    WHEN stars BETWEEN 4.0 AND 5.0 THEN 'high'
  END AS ratingcategory,
review_count, dayofweek, StartTime, EndTime, workhours
FROM business, category, (SELECT business_id, dayofweek, fStartHour as StartTime, fEndhour as EndTime,
             CASE
                 WHEN (julianday(fEndhour) - julianday(fStartHour)) < 0 THEN (24 - (-1*round(((julianday(fEndhour) -
julianday(fStartHour))*24),1)))
                 ELSE round(((julianday(fEndhour) - julianday(fStartHour))*24),1)
             END AS workhours
      FROM
        (SELECT *,
                CASE
                    WHEN cast(substr(starthour, 1, instr(starthour, ':')) AS int) >= 10 THEN starthour
                    ELSE ('0' || starthour)
                END AS fStartHour ,
                CASE
                    WHEN cast(substr(endhour, 1, instr(endhour, ':')) AS int) > 10 THEN endhour
                    ELSE ('0' || endhour)
                END AS fEndhour
         FROM
           (SELECT *,
                   substr(timeperday, instr(timeperday, '-')+1) AS endhour,
                   substr(timeperday, 1, instr(timeperday, '-')-1) AS starthour
            FROM
```

```
              (SELECT business_id,
                      substr(hours, instr(hours, '|')+1) AS timeperday,
                      substr(hours, 1, instr(hours, '|')-1) AS dayofweek
               FROM hours
               )))) as schedule
WHERE business.id = category.business_id
AND business.id = schedule.business_id
AND category = 'Restaurants'
AND city = 'Toronto'
)
GROUP BY id, name, neighborhood, postal_code, stars, ratingcategory,
review_count
```

Aggregate Analysis Table Result:

id	name	neighborhood	postal_code	stars	ratingcategory	review_count	Number of Days Open	Number of Hours operational per Day
-n27mJ_jQWGCulukTvg9Mg	Cabin Fever	High Park	M6P 1A6	4.5	high	26	7	8.857142857142858
0B3W6KxkD3o4W4l6cq735w	Big Smoke Burger	Downtown Core	M4B 2L9	3.0	low	47	7	10.142857142857142
0e-j5VcEn54EZT-FKCUZdw	Sushi Osaka	Etobicoke	M9A 1C2	4.5	high	8	7	11.571428571428571
1NyHpXJqSLHnvDCOW0nJDg	Pizzaiolo	Entertainment District	M5H 1X6	3.0	low	34	7	15.142857142857142
1nTMWMa6v-eBKkPYA3gxkQ	99 Cent Sushi	Downtown Core	M5B 2E5	2.0	low	5	7	12.0
37kk0IW6jL7ZlxZF6k2QBg	Edulis	Niagara	M5V	4.0	high	89	5	4.8

SQL Query for Part 1:

```
SELECT ratingcategory, avg(numberofdays) as "Number of Days Open per Category", avg(numberofhours) as "Average Number of
Operational Hours per Day per Category", count(id) as "Number of Businesses Per Category"
FROM
(SELECT
id, name, neighborhood, postal_code, stars, ratingcategory,
review_count, count(dayofweek) as numberofdays,
sum(workhours)/count(dayofweek) as numberofhours
FROM
(
SELECT id, name, neighborhood, postal_code, stars,
  CASE
    WHEN stars BETWEEN 2.0 AND 3.0 THEN 'low'
    WHEN stars BETWEEN 4.0 AND 5.0 THEN 'high'
  END AS ratingcategory,
review_count, dayofweek, StartTime, EndTime, workhours
FROM business, category, (SELECT business_id, dayofweek, fStartHour as StartTime, fEndhour as EndTime,
             CASE
                 WHEN (julianday(fEndhour) - julianday(fStartHour)) < 0 THEN (24 - (-1*round(((julianday(fEndhour) -
julianday(fStartHour))*24),1)))
                 ELSE round(((julianday(fEndhour) - julianday(fStartHour))*24),1)
             END AS workhours
      FROM
        (SELECT *,
                CASE
                    WHEN cast(substr(starthour, 1, instr(starthour, ':')) AS int) >= 10 THEN starthour
                    ELSE ('0' || starthour)
                END AS fStartHour ,
                CASE
                    WHEN cast(substr(endhour, 1, instr(endhour, ':')) AS int) > 10 THEN endhour
                    ELSE ('0' || endhour)
                END AS fEndhour
         FROM
           (SELECT *,
                   substr(timeperday, instr(timeperday, '-')+1) AS endhour,
                   substr(timeperday, 1, instr(timeperday, '-')-1) AS starthour
            FROM
              (SELECT business_id,
                      substr(hours, instr(hours, '|')+1) AS timeperday,
                      substr(hours, 1, instr(hours, '|')-1) AS dayofweek
```

```
               FROM hours
               )))) as schedule
WHERE business.id = category.business_id
AND business.id = schedule.business_id
AND category = 'Restaurants'
AND city = 'Toronto'
)
GROUP BY id, name, neighborhood, postal_code, stars, ratingcategory,
review_count
)
GROUP BY ratingcategory
```

SQL Query for Part 2:

```
SELECT ratingcategory, count(id) as "Number of Businesses Per Category",
round(avg(review_count)) as "Average number of reviews per business per category"
, sum(review_count) "Total number of reviews per category"
FROM
(SELECT
id, name, neighborhood, postal_code, stars, ratingcategory,
review_count, count(dayofweek) as numberofdays,
sum(workhours)/count(dayofweek) as numberofhours
FROM
(
SELECT id, name, neighborhood, postal_code, stars,
  CASE
    WHEN stars BETWEEN 2.0 AND 3.0 THEN 'low'
    WHEN stars BETWEEN 4.0 AND 5.0 THEN 'high'
  END AS ratingcategory,
review_count, dayofweek, StartTime, EndTime, workhours
FROM business, category, (SELECT business_id, dayofweek, fStartHour as StartTime, fEndhour as EndTime,
             CASE
                 WHEN (julianday(fEndhour) - julianday(fStartHour)) < 0 THEN (24 - (-1*round(((julianday(fEndhour) -
julianday(fStartHour))*24),1)))
                 ELSE round(((julianday(fEndhour) - julianday(fStartHour))*24),1)
             END AS workhours
      FROM
        (SELECT *,
                CASE
                    WHEN cast(substr(starthour, 1, instr(starthour, ':')) AS int) >= 10 THEN starthour
                    ELSE ('0' || starthour)
                END AS fStartHour ,
                CASE
                    WHEN cast(substr(endhour, 1, instr(endhour, ':')) AS int) > 10 THEN endhour
                    ELSE ('0' || endhour)
                END AS fEndhour
         FROM
           (SELECT *,
                   substr(timeperday, instr(timeperday, '-')+1) AS endhour,
                   substr(timeperday, 1, instr(timeperday, '-')-1) AS starthour
            FROM
              (SELECT business_id,
                      substr(hours, instr(hours, '|')+1) AS timeperday,
                      substr(hours, 1, instr(hours, '|')-1) AS dayofweek
               FROM hours
               )))) as schedule
WHERE business.id = category.business_id
AND business.id = schedule.business_id
AND category = 'Restaurants'
AND city = 'Toronto'
)
GROUP BY id, name, neighborhood, postal_code, stars, ratingcategory,
review_count
)
GROUP BY ratingcategory
```

SQL Query for Part 3:

```
SELECT ratingcategory, neighborhood, postal_code
FROM check aggregate table above
```

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.

Classifying the Two Types (Open vs Closed)

What's the average star rating between them:

is_open	avg(stars)
0	3.520394736842105
1	3.679009433962264

1 is open, 0 is Closed

```
SELECT
is_open, avg(stars)
FROM business
GROUP by is_open
```

Review Count per business (Open Vs Closed):

is_open	avg(review_count)
0	23.198026315789473
1	31.757075471698112

```
SELECT
is_open, avg(review_count)
FROM business
GROUP by is_open
```

Find the top category between open and closed businesses:

Type	category	countcategory
Closed	Restaurants	18
Open	Restaurants	53

```
SELECT * FROM
(SELECT 'Closed' as Type, category.category, count(category.business_id) as countcategory
FROM business, category
WHERE business.id = category.business_id
AND is_open = 0
GROUP BY category.category
ORDER BY countcategory DESC
LIMIT 1)

UNION
SELECT * FROM (
SELECT 'Open' as Type, category.category, count(category.business_id) as countcategory
FROM business, category
WHERE business.id = category.business_id
AND is_open = 1
GROUP BY category.category
ORDER BY countcategory DESC
LIMIT 1)
```

Differences between the number of useful, funny, cool reviews between open and closed businesses:

is_open	useful	funny	cool
0	0.97	0.21	0.42
1	0.86	0.27	0.39

Between Open vs Closed restaurants, find the ratio (target word/total number of reviews per open and closed) of a target word:

is_open	like	hate	bad	rude	love	comeback
0	0.38	0.04	0.1	0.01	0.17	0.04

is_open	like	hate	bad	rude	love	comeback
1	0.27	0.02	0.07	0.03	0.2	0.02

```
SELECT is_open
,round(cast(count(CASE WHEN review.text LIKE '%like%' THEN review.id END) as real)/(cast(count(review.id) as real)),2) as
"like"
,round(cast(count(CASE WHEN review.text LIKE '%hate%' THEN review.id END) as real)/(cast(count(review.id) as real)),2) as
"hate"
,round(cast(count(CASE WHEN review.text LIKE '%bad%' THEN review.id END) as real)/(cast(count(review.id) as real)),2) as "bad"
,round(cast(count(CASE WHEN review.text LIKE '%rude%' THEN review.id END) as real)/(cast(count(review.id) as real)),2) as
"rude"
,round(cast(count(CASE WHEN review.text LIKE '%love%' THEN review.id END) as real)/(cast(count(review.id) as real)),2) as
"love"
,round(cast(count(CASE WHEN review.text LIKE '%come back%' THEN review.id END) as real)/(cast(count(review.id) as real)),2) as
"comeback"

FROM business, review
WHERE business.id = review.business_id
GROUP BY is_open
```

Yelp has various business attributes. One of them being a Price Category which goes from $ (least expensive) to $$$$ most expensive. Find the ratio of each category to the total number of businesses in each category:

is_open	$	$$	$$$	$$$$
0	0.25	0.67	0.08	0.0
1	0.3	0.57	0.09	0.05

```
SELECT is_open
,round(cast(count(CASE WHEN attribute.value = 1 THEN business.id END) as real)/count(business.id),2) as "$"
,round(cast(count(CASE WHEN attribute.value = 2 THEN business.id END) as real)/count(business.id),2) as "$$"
,round(cast(count(CASE WHEN attribute.value = 3 THEN business.id END) as real)/count(business.id),2) as "$$$"
,round(cast(count(CASE WHEN attribute.value = 4 THEN business.id END) as real)/count(business.id),2) as "$$$$"
FROM business, attribute
where business.id = attribute.business_id
AND attribute.name LIKE '%price%'
GROUP BY is_open
```

i. Difference 1:

We see that a closed business has less reviews (review_count) than open reviews. One reason could be because an Open business is open to collect more reviews

ii. Difference 2:

We can see that for closed businesses, there are more $$ price category closed than open restaurants. So Mid-Tier restaurants by price are more likely to closed based on this dataset.

iii. Further Difference:

We can see doing the review analysis that surprisingly for closed restaurants, more reviews are tagged as Useful and Cool, and more reviews contained the word Liked for closed restaurants than Open ones!

SQL code used for analysis: Shown above

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:

```
Geospatial Analysis including Regionalization and Clustering.
```

Comparing Various Regions and their business attributes of the Yelp Dataset. I want to find Region Attributes for a what I designate to be related geographic regions to partition the dataset and then find attributes of that given region. I hypothesize each geographic region will have a similar number of restaurants in the same restaurant category. I also think a given region will have similar amounts of reviews for a region due to population. If we normalize this we will be able to do cross region comparisons using the Yelp Dataset, comparing one region to another. For example, comparing the Southwest of Region of the US (LA, Las Vegas) to the Northeast region of the US (New York, DC). I retain the Business ID information to verify each record being distinct. I don't need to retain other traits like what the business name is, or if it is open or not.

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

My dataset will be composed of the Business, Review, and Category table. The primary piece of information will be the latitude and longitude information which will be used for defining geographic regions using something like Geohasing. All other tables will give the business attribute information that can be used for the analysis. I can also include the Attribute and Hours table, as well as the Checkin and Photo tables. This will give us various attributes per region which we can use for the region comparison analysis.

iii. Output of your finished dataset:

id	city	state	postal_code	latitude	longitude	stars	review_count	category	name	value
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Nightlife	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Bars	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Food	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Restaurants	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Smokehouse	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	American (Traditional)	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Barbeque	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Nightlife	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Bars	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Food	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Restaurants	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Smokehouse	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	American (Traditional)	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Barbeque	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Nightlife	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Bars	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Food	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Restaurants	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Smokehouse	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	American (Traditional)	RestaurantsPriceRange2	2
2skQeu3C36VCiB653MIfrw	Phoenix	AZ	85028	33.5818	-112.008	4.0	431	Barbeque	RestaurantsPriceRange2	2

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

```
SELECT business.id, business.city, business.state, business.postal_code, business.latitude, business.longitude,
business.stars, business.review_count
,category.category
,attribute.name, attribute.value
FROM business, category, attribute, review
WHERE business.id = category.business_id
AND business.id = attribute.business_id
AND business.id = review.business_id
AND attribute.name LIKE '%price%'
```