

CIS9760\_Project2\_Yaheng Wu(Phoebe)

# Analysis of Yelp Business Intelligence Data

Big Idea:

In this project, I analyzed a subset of the Yelp's business, reviews and user data to answer the following three questions:

- Do Yelp Reviews Skew Negative?
- Should the Elite be Trusted?
- What is the Most Recommended Restaurant?

Data Source:

The three datasets originally come from [Kaggle](#) and they have been uploaded into an S3 bucket for the use of this project.

- s3://yelppreviewdataset/yelp\_academic\_dataset\_business.json
- s3://yelppreviewdataset/yelp\_academic\_dataset\_review.json
- s3://yelppreviewdataset/yelp\_academic\_dataset\_user.json

## Part I: Installation and Initial Setup

### 1. Install Packages

```
In [1]: from pyspark.sql import SparkSession
my_spark = SparkSession.builder.getOrCreate()
sc.install_pypi_package("pandas==1.0.3")
sc.install_pypi_package("matplotlib==3.2.1")
sc.install_pypi_package("seaborn==0.10.0")
sc.install_pypi_package("wordcloud==1.8.1")
sc.list_packages()
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1606507498582_0001	pyspark	idle	<a href="#">Link</a>	<a href="#">Link</a>	✓

SparkSession available as 'spark'.

Collecting pandas==1.0.3

Downloading [https://files.pythonhosted.org/packages/4a/6a/94b219b8ea0f2d580169e85ed1edc0163743f55aaeca8a44c2e8fc1e344e/pandas-1.0.3-cp37-cp37m-manylinux1\\_x86\\_64.whl](https://files.pythonhosted.org/packages/4a/6a/94b219b8ea0f2d580169e85ed1edc0163743f55aaeca8a44c2e8fc1e344e/pandas-1.0.3-cp37-cp37m-manylinux1_x86_64.whl) (10.0MB)  
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (from pandas==1.0.3)

Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib64/python3.7/site-packages (from pandas==1.0.3)

Collecting python-dateutil>=2.6.1 (from pandas==1.0.3)

Downloading [https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207ae8907090de0b306af2bce5d134d78615cb/python\\_dateutil-2.8.1-py2.py3-none-any.whl](https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207ae8907090de0b306af2bce5d134d78615cb/python_dateutil-2.8.1-py2.py3-none-any.whl) (227kB)  
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas==1.0.3)

Installing collected packages: python-dateutil, pandas

Successfully installed pandas-1.0.3 python-dateutil-2.8.1

Collecting matplotlib==3.2.1

Downloading [https://files.pythonhosted.org/packages/b2/c2/71fcf957710f3ba1f09088b35776a799ba7dd95f7c2b195ec800933b276b/matplotlib-3.2.1-cp37-cp37m-manylinux1\\_x86\\_64.whl](https://files.pythonhosted.org/packages/b2/c2/71fcf957710f3ba1f09088b35776a799ba7dd95f7c2b195ec800933b276b/matplotlib-3.2.1-cp37-cp37m-manylinux1_x86_64.whl) (12.4 MB)

Requirement already satisfied: python-dateutil>=2.1 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib==3.2.1)

Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib==3.2.1)

Downloading <https://files.pythonhosted.org/packages/8a/bb/488841f56197b13700afd5658fc279a2025a39e22449b7cf29864669b15d/pyparsing-2.4.7-py2.py3-none-any.whl> (67kB)

Collecting cycler>=0.10 (from matplotlib==3.2.1)

Downloading <https://files.pythonhosted.org/packages/f7/d2/e07d3ebb2bd7af696440ce7e754c59dd546ffe1bbe732c8ab68b9c834e61/cycler-0.10.0-py2.py3-none-any.whl>

Requirement already satisfied: numpy>=1.11 in /usr/local/lib64/python3.7/site-packages (from matplotlib==3.2.1)

Collecting kiwisolver>=1.0.1 (from matplotlib==3.2.1)

Downloading [https://files.pythonhosted.org/packages/d2/46/231de802ade4225b76b96cffe419cf3ce52bbe92e3b092cf12db7d11c207/kiwisolver-1.3.1-cp37-cp37m-manylinux1\\_x86\\_64.whl](https://files.pythonhosted.org/packages/d2/46/231de802ade4225b76b96cffe419cf3ce52bbe92e3b092cf12db7d11c207/kiwisolver-1.3.1-cp37-cp37m-manylinux1_x86_64.whl) (1.1 MB)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.1->matplotlib==3.2.1)

Installing collected packages: pyparsing, cycler, kiwisolver, matplotlib

Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.2.1 pyparsing-2.4.7

Collecting seaborn==0.10.0

Downloading <https://files.pythonhosted.org/packages/70/bd/5e6bf595fe6ee0f257ae49336dd180768c1ed3d7c7155b2fdf894c1c808a/seaborn-0.10.0-py3-none-any.whl> (215kB)

Requirement already satisfied: pandas>=0.22.0 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from seaborn==0.10.0)

Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib64/python3.7/site-packages (from seaborn==0.10.0)

Collecting scipy>=1.0.1 (from seaborn==0.10.0)

Downloading [https://files.pythonhosted.org/packages/dc/7e/8f6a79b102ca1ea928bae8998b05bf5dc24a90571db13cd119f275ba6252/scipy-1.5.4-cp37-cp37m-manylinux1\\_x86\\_64.whl](https://files.pythonhosted.org/packages/dc/7e/8f6a79b102ca1ea928bae8998b05bf5dc24a90571db13cd119f275ba6252/scipy-1.5.4-cp37-cp37m-manylinux1_x86_64.whl) (25.9MB)

Requirement already satisfied: matplotlib>=2.1.2 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from seaborn==0.10.0)

Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (from pandas>=0.22.0->seaborn==0.10.0)

Requirement already satisfied: python-dateutil>=2.6.1 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from pandas>=0.22.0->seaborn==0.10.0)

Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib>=2.1.2->seaborn==0.10.0)

Requirement already satisfied: cycler>=0.10 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib>=2.1.2->seaborn==0.10.0)

Requirement already satisfied: kiwisolver>=1.0.1 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib>=2.1.2->seaborn==0.10.0)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas>=0.22.0->seaborn==0.10.0)

Installing collected packages: scipy, seaborn

Successfully installed scipy-1.5.4 seaborn-0.10.0

Collecting wordcloud==1.8.1

Downloading [https://files.pythonhosted.org/packages/1b/06/0516bdba2ebdc0d5bd476aa66f94666dd0ad6b9abda723fdf28e451db919/wordcloud-1.8.1-cp37-cp37m-manylinux1\\_x86\\_64.whl](https://files.pythonhosted.org/packages/1b/06/0516bdba2ebdc0d5bd476aa66f94666dd0ad6b9abda723fdf28e451db919/wordcloud-1.8.1-cp37-cp37m-manylinux1_x86_64.whl) (366kB)

Collecting pillow (from wordcloud==1.8.1)

Downloading [https://files.pythonhosted.org/packages/af/fa/c1302a26d5e1a17fa8e10e43417b6cf038b0648c4b79fcf2302a4a0c5d30/Pillow-8.0.1-cp37-cp37m-manylinux1\\_x86\\_64.whl](https://files.pythonhosted.org/packages/af/fa/c1302a26d5e1a17fa8e10e43417b6cf038b0648c4b79fcf2302a4a0c5d30/Pillow-8.0.1-cp37-cp37m-manylinux1_x86_64.whl) (2.2MB)

Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib64/python3.7/site-packages (from wordcloud==1.8.1)

Requirement already satisfied: matplotlib in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from wordcloud==1.8.1)

Requirement already satisfied: python-dateutil>=2.1 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib->wordcloud==1.8.1)

Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib->wordcloud==1.8.1)

Requirement already satisfied: cyclar>=0.10 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib->wordcloud==1.8.1)  
 Requirement already satisfied: kiwisolver>=1.0.1 in /mnt/tmp/1606508098465-0/lib/python3.7/site-packages (from matplotlib->wordcloud==1.8.1)  
 Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.1->matplotlib->wordcloud==1.8.1)  
 Installing collected packages: pillow, wordcloud  
 Successfully installed pillow-8.0.1 wordcloud-1.8.1

Package	Version
-----	-----
beautifulsoup4	4.9.1
boto	2.49.0
click	7.1.2
cyclar	0.10.0
jmespath	0.10.0
joblib	0.16.0
kiwisolver	1.3.1
lxml	4.5.2
matplotlib	3.2.1
mysqlclient	1.4.2
nlTK	3.5
nose	1.3.4
numpy	1.16.5
pandas	1.0.3
Pillow	8.0.1
pip	9.0.1
py-dateutil	2.2
pyarsing	2.4.7
python-dateutil	2.8.1
python37-sagemaker-pyspark	1.4.0
pytz	2020.1
PyYAML	5.3.1
regex	2020.7.14
scipy	1.5.4
seaborn	0.10.0
setuptools	28.8.0
six	1.13.0
soupsieve	1.9.5
tqdm	4.48.2
wheel	0.29.0
windmill	1.6
wordcloud	1.8.1

## 2. Importing

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from os import path
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import pyspark.sql.functions as F
from pyspark.sql.functions import explode, split, desc, col, avg, udf, when
from pyspark.sql.types import IntegerType, StringType, DoubleType
```

## 3. Loading Data

```
In [3]: business_df = spark.read.json('s3://yelpreviewdataset/yelp_academic_dataset_business.js
```

## 4. Overview of Data

```
In [4]: print(f'Columns: {len(business_df.columns)} | Rows: {business_df.count().:,}')

```

Columns: 14 | Rows: 209,393

```
In [5]: business_df.printSchema()

```

```
root
|-- address: string (nullable = true)
|-- attributes: struct (nullable = true)
|   |-- AcceptsInsurance: string (nullable = true)
|   |-- AgesAllowed: string (nullable = true)
|   |-- Alcohol: string (nullable = true)
|   |-- Ambience: string (nullable = true)
|   |-- BYOB: string (nullable = true)
|   |-- BYOBCorkage: string (nullable = true)
|   |-- BestNights: string (nullable = true)
|   |-- BikeParking: string (nullable = true)
|   |-- BusinessAcceptsBitcoin: string (nullable = true)
|   |-- BusinessAcceptsCreditCards: string (nullable = true)
|   |-- BusinessParking: string (nullable = true)
|   |-- ByAppointmentOnly: string (nullable = true)
|   |-- Caters: string (nullable = true)
|   |-- CoatCheck: string (nullable = true)
|   |-- Corkage: string (nullable = true)
|   |-- DietaryRestrictions: string (nullable = true)
|   |-- DogsAllowed: string (nullable = true)
|   |-- DriveThru: string (nullable = true)
|   |-- GoodForDancing: string (nullable = true)
|   |-- GoodForKids: string (nullable = true)
|   |-- GoodForMeal: string (nullable = true)
|   |-- HairSpecializesIn: string (nullable = true)
|   |-- HappyHour: string (nullable = true)
|   |-- HasTV: string (nullable = true)
|   |-- Music: string (nullable = true)
|   |-- NoiseLevel: string (nullable = true)
|   |-- Open24Hours: string (nullable = true)
|   |-- OutdoorSeating: string (nullable = true)
|   |-- RestaurantsAttire: string (nullable = true)
|   |-- RestaurantsCounterService: string (nullable = true)
|   |-- RestaurantsDelivery: string (nullable = true)
|   |-- RestaurantsGoodForGroups: string (nullable = true)
|   |-- RestaurantsPriceRange2: string (nullable = true)
|   |-- RestaurantsReservations: string (nullable = true)
|   |-- RestaurantsTableService: string (nullable = true)
|   |-- RestaurantsTakeOut: string (nullable = true)
|   |-- Smoking: string (nullable = true)
|   |-- WheelchairAccessible: string (nullable = true)
|   |-- WiFi: string (nullable = true)
|-- business_id: string (nullable = true)
|-- categories: string (nullable = true)
|-- city: string (nullable = true)
|-- hours: struct (nullable = true)
|   |-- Friday: string (nullable = true)
|   |-- Monday: string (nullable = true)
|   |-- Saturday: string (nullable = true)
|   |-- Sunday: string (nullable = true)

```

```

|         |-- Thursday: string (nullable = true)
|         |-- Tuesday: string (nullable = true)
|         |-- Wednesday: string (nullable = true)
|-- is_open: long (nullable = true)
|-- latitude: double (nullable = true)
|-- longitude: double (nullable = true)
|-- name: string (nullable = true)
|-- postal_code: string (nullable = true)
|-- review_count: long (nullable = true)
|-- stars: double (nullable = true)
|-- state: string (nullable = true)

```

Display the first 5 rows with the following columns:

- business\_id
- name
- city
- state
- categories

```
In [6]: busi_df = business_df.select('business_id', 'name', 'city', 'state', 'stars', 'categories')
        busi_df.show(5)
```

```

+-----+-----+-----+-----+-----+-----+
---+
|      business_id|      name|      city|state|stars|      categories|
+-----+-----+-----+-----+-----+-----+
---+
|f9NumwFMBDn751xgF...|The Range At Lake...|    Cornelius|    NC|  3.5|Active Life, Gu
n/...|
|YzvJg0SayhoZgCljU...|  Carlos Santo, NMD|    Scottsdale|    AZ|  5.0|Health & Medica
l,...|
|XNoUzKckATkOD1hP6...|          Felinus|    Montreal|    QC|  5.0|Pets, Pet Servic
e...|
|60AZjbxqM5o129BuH...|Nevada House of Hose|North Las Vegas|    NV|  2.5|Hardware Stores,
...|
|51M2Kk903DFYI6gnB...|USE MY GUY SERVIC...|        Mesa|    AZ|  4.5|Home Services, P
l...|
+-----+-----+-----+-----+-----+-----+
---+
only showing top 5 rows

```

## Part II: Analyzing Categories

Let's now answer: **How many unique categories are represented in this dataset?**

Essentially, we have the categories per business as a list - this is useful to quickly see what each business might be represented as but it is difficult to easily answer the following questions such as:

- How many businesses are categorized as Active Life?
- What are the top 20 most popular categories available?

### 1. Association Table

We need to "break out" these categories from the business ids? One common approach to take is to build an association table mapping a single business id multiple times to each distinct category.

For instance, given the following:

business_id	categories
abcd123	a,b,c

We would like to derive something like:

business_id	category
abcd123	a
abcd123	b
abcd123	c

What this does is allow us to then perform a myriad of rollups and other analysis on this association table which can aid us in answering the questions asked above.

Display the first 5 rows of the association table below

```
In [7]: associ_table_one = business_df.select('business_id', explode(split(business_df.categories)))
associ_table_one.show(5)
```

```
+-----+-----+
| business_id | category |
+-----+-----+
| f9NumwFMBDn751xgF... | Active Life |
| f9NumwFMBDn751xgF... | Gun/Rifle Ranges |
| f9NumwFMBDn751xgF... | Guns & Ammo |
| f9NumwFMBDn751xgF... | Shopping |
| YzvJg0SayhoZgCljU... | Health & Medical |
+-----+-----+
only showing top 5 rows
```

## 2. Total Unique Categories

Finally, we are ready to answer the question: **what is the total number of unique categories available?**

```
In [8]: associ_table_one.select('category').distinct().count()
```

1336

## 3. Top Categories By Business

Now let's find the top categories in this dataset by rolling up categories.

### Counts of Businesses / Category

```
In [9]: category_count = associ_table_one.select('category').groupby(associ_table_one.category)
category_count.show()
```

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

category	count
Dermatologists	341
Paddleboarding	36
Aerial Tours	28
Hobby Shops	828
Bubble Tea	720
Embassy	13
Handyman	682
Tanning	938
Aerial Fitness	29
Tempura	1
Falafel	159
Outlet Stores	399
Summer Camps	318
Clothing Rental	55
Sporting Goods	2311
Cooking Schools	118
College Counseling	15
Lactation Services	50
Ski & Snowboard S...	50
Museums	359

only showing top 20 rows

## Bar Chart of Top Categories

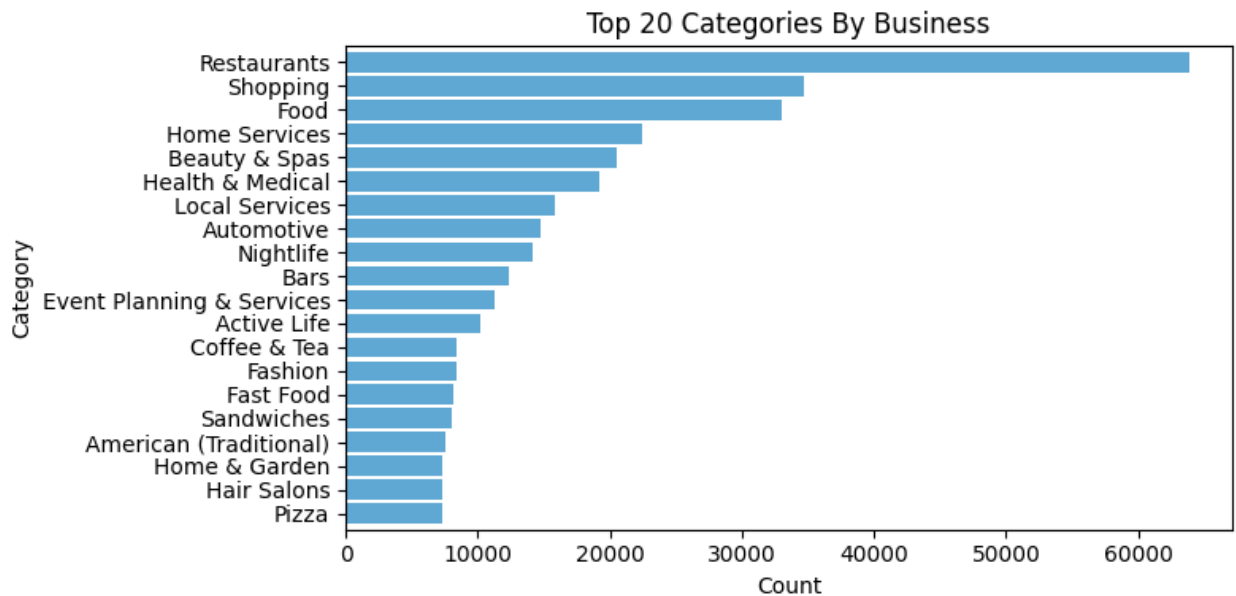
With this data available, let us now build a barchart of the top 20 categories

```
In [10]: top_20 = category_count.sort(desc('count')).limit(20).toPandas()

ax = top_20.plot(kind='barh', x='category', y='count',
                 figsize=(8, 4), color='#5fa8d3', zorder=2, width=0.85)

ax.invert_yaxis()
ax.set_xlabel("Count")
ax.set_ylabel("Category")
ax.set_title("Top 20 Categories By Business")
ax.get_legend().remove()

plt.tight_layout()
%matplotlib plt
```



Clears the entire current figure with all its axes

```
In [11]: plt.clf()
plt.cla()
plt.close()
```

## Part III. Do Yelp Reviews Skew Negative?

Oftentimes, it is said that the only people who write a written review are those who are extremely dissatisfied or extremely satisfied with the service received.

How true is this really? Let's try and answer this question.

### 1. Loading Review Data

Begin by loading the review data set from S3 and printing schema to determine what data is available

```
In [12]: review_df = spark.read.json('s3://yelpreviewdataset/yelp_academic_dataset_review.json')
review_df.printSchema()
```

```
root
|-- business_id: string (nullable = true)
|-- cool: long (nullable = true)
|-- date: string (nullable = true)
|-- funny: long (nullable = true)
|-- review_id: string (nullable = true)
|-- stars: double (nullable = true)
|-- text: string (nullable = true)
|-- useful: long (nullable = true)
|-- user_id: string (nullable = true)
```

Let's begin by listing the business\_id and stars columns together for the user reviews data

```
business_stars = review_df.select('business_id', 'stars')
```



```
In [13]: business_stars.show(5)
```

```
+-----+-----+
|      business_id|stars|
+-----+-----+
|-MhfebM0QIsKt87iD...| 2.0|
|lbrU8StCq3yDfr-QM...| 1.0|
|HQ128KMwrEKHqhFrr...| 5.0|
|5JxlZaqCnk1MnbgRi...| 1.0|
|IS4cv902ykd8wj1TR...| 4.0|
+-----+-----+
only showing top 5 rows
```

Now, let's aggregate along the stars column to get a resultant dataframe that displays average stars per business as accumulated by users who **took the time to submit a written review**

```
In [14]: written_review = review_df.where(col("text").isNotNull()).groupby(review_df.business_id)
written_review.show(5)
```

```
+-----+-----+
|      business_id|      avg(stars)|
+-----+-----+
|VHsNB3pdGVcRgs6C3...| 3.411764705882353|
|RMjCnixEY5i12Ciqn...| 3.5316455696202533|
|ipFreSFhjClfNETuM...| 2.6|
|dLDMU8b0LnkDTmPUr...| 4.942857142857143|
|Qm2datcYBPXrPATVG...| 4.352941176470588|
+-----+-----+
only showing top 5 rows
```

Now the fun part - let's join our two dataframes (reviews and business data) by business\_id

```
In [15]: user_review = review_df.groupby(review_df.business_id).agg(avg(col("stars")))
# inner join
joined_written_review = business_df.join(written_review, on=['business_id'])
joined_user_review = business_df.join(user_review,on=['business_id'])
```

Let's see a few of these:

```
In [16]: joined_written_review.select("""avg(stars)""", "stars", "name", "city", "state").sort(desc(
```

```
+-----+-----+-----+-----+-----+
|avg(stars)|stars|      name|      city|state|
+-----+-----+-----+-----+-----+
|      5.0|  5.0|Clark Bodywork Th...| Cave Creek|  AZ|
|      5.0|  5.0|Compass Wealth Pl...| Scottsdale|  AZ|
|      5.0|  5.0|    Colossus Tattoo|    Tempe|  AZ|
|      5.0|  5.0|    Dream Lawns|    Las Vegas|  NV|
|      5.0|  5.0|    Dogs Day Out|Indian Trail|  NC|
+-----+-----+-----+-----+-----+
only showing top 5 rows
```

Compute a new dataframe that calculates what we will call the skew (for lack of a better word) between the avg stars accumulated from written reviews and the actual star rating of a business (ie: the average of stars given by reviewers who wrote an actual review and reviewers who just provided a star rating).

The formula you can use is something like:

$(\text{row}[\text{'avg(stars)'}] - \text{row}[\text{'stars'}]) / \text{row}[\text{'stars'}]$

If the **skew** is negative, we can interpret that to be: reviewers who left a written response were more dissatisfied than normal. If **skew** is positive, we can interpret that to be: reviewers who left a written response were more satisfied than normal.

```
In [17]: fv_joined_written_review = joined_written_review.select("avg(stars)", "stars", "name", "city", "state")
        .sort("avg(stars)", ascending=False)

fv_joined_user_review = joined_user_review.select("avg(stars)", "stars", "name", "city", "state")
        .sort("avg(stars)", ascending=False)

fv_df = fv_joined_written_review.withColumn("skew",
        ((fv_joined_written_review["avg(stars)"] - fv_joined_written_review["stars"]) / fv_joined_written_review["stars"]))

fv_df.sort("skew", ascending=False).show()
```

avg(stars)	stars	name	city	state	skew
2.333333333333335	1.0	Mikado Sushi Robata	Toronto	ON	1.333333333333335
3.333333333333335	1.5	Black Brook Golf ...	Mentor	OH	1.222222222222223
2.0	1.0	Convenient Food M...	Elyria	OH	1.0
2.0	1.0	Water Dr	Calgary	AB	1.0
2.0	1.0	DollarPlus Discou...	Las Vegas	NV	1.0
2.0	1.0	Torrey Pines Reha...	Las Vegas	NV	1.0
2.0	1.0	Foothills Primary...	Chandler	AZ	1.0
2.0	1.0	H&R Block	Calgary	AB	1.0
2.0	1.0	StorageOne	Las Vegas	NV	1.0
2.0	1.0	Children's Campus...	Phoenix	AZ	1.0
2.0	1.0	Affordable Decks ...	Bethel Park	PA	1.0
2.0	1.0	Golden West Pool ...	Las Vegas	NV	1.0
2.8	1.5	RideNow Powerspor...	Phoenix	AZ	0.8666666666666666
1.833333333333333	1.0	Tri-County Snow P...	Medina	OH	0.833333333333333
1.8	1.0	Mr. Transmission/...	Matthews	NC	0.8
1.8	1.0	Colangelo's no Fr...	Oakville	ON	0.8
1.8	1.0	The Continental A...	Phoenix	AZ	0.8
1.8	1.0	Mathis Towing and...	Charlotte	NC	0.8
1.8	1.0	Pulte Homes	Las Vegas	NV	0.8
1.8	1.0	1-2-3 Automotive	Henderson	NV	0.8

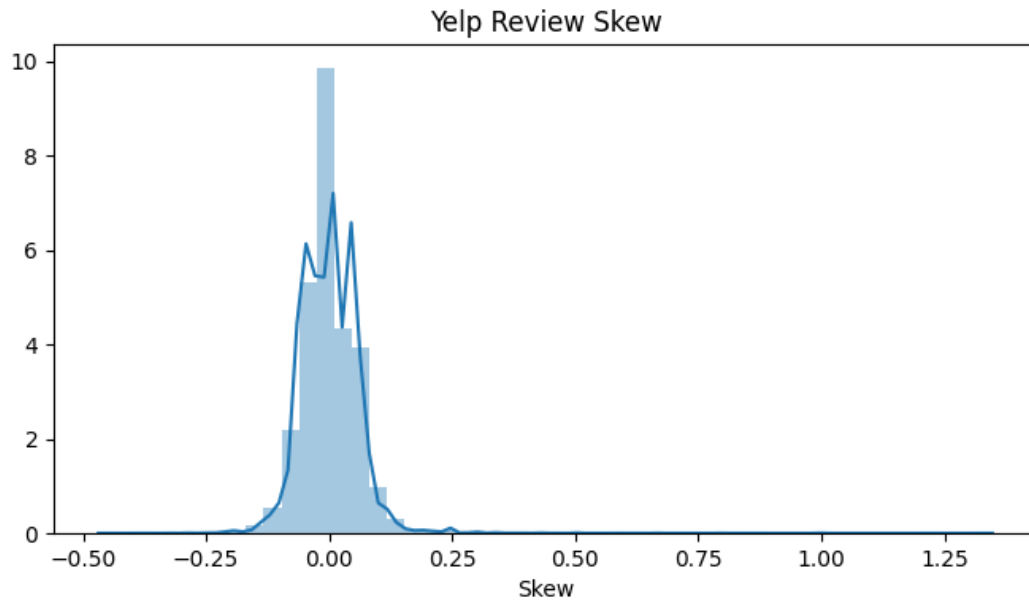
only showing top 20 rows

```
In [18]: fv_df = fv_df.toPandas()
```

And finally, graph it!

```
In [19]: plt.figure(figsize=(8,4))
        ax = sns.distplot(fv_df["skew"])
        ax.set_xlabel('Skew')
        plt.title("Yelp Review Skew")

        %matplotlib plt
```



Clears the entire current figure with all its axes

```
In [20]: plt.clf()  
plt.cla()  
plt.close()
```

So, do Yelp (written) Reviews skew negative? Does this analysis actually prove anything? Expound on implications / interpretations of this graph.

The distribution of skew appears to be normal, but skewed a little bit to the right. The implications of the above graph are that the satisfaction level of reviewers who left positively skewed reviews is greater than the dissatisfaction level of reviewers who left negatively skewed reviews. In other words, reviewers who left a written response were slightly more satisfied than normal.

## Part IV. Should the Elite be Trusted?

How accurate or close are the ratings of an "elite" user (check Users table schema) vs the actual business rating

It takes a special Yelper to become an Elite. Frequent, quality reviews and photos are important in the application of the elite status on Yelp. Elite candidates need to meet the criteria below for the consideration.

To become Elite, Yelpers agree that they

- Are using their real name on Yelp.
- Have a clear photo of themselves on their profile page.
- Are of legal drinking age where they live.

They also agree that they are NOT

- A business owner.
- Closely affiliated with a business owner.
- Managing a Yelp Business Account.
- Working for one of Yelp's competitors.

It's important to know that accepting compensation or freebies in exchange for reviews or leveraging the Elite Squad for personal or commercial gain will result in Elite status being revoked or account closure.

## 1. Loading User Data

```
In [21]: user_df = spark.read.json('s3://yelpreviewdataset/yelp_academic_dataset_user.json')
```

## 2. Overview of Data

```
In [22]: user_df.printSchema()
print(f'User Dataset Columns: {len(user_df.columns)} | Rows: {user_df.count():,}')
review_df.printSchema()
print(f'Review Dataset Columns: {len(review_df.columns)} | Rows: {review_df.count():,}')
```

```
root
|-- average_stars: double (nullable = true)
|-- compliment_cool: long (nullable = true)
|-- compliment_cute: long (nullable = true)
|-- compliment_funny: long (nullable = true)
|-- compliment_hot: long (nullable = true)
|-- compliment_list: long (nullable = true)
|-- compliment_more: long (nullable = true)
|-- compliment_note: long (nullable = true)
|-- compliment_photos: long (nullable = true)
|-- compliment_plain: long (nullable = true)
|-- compliment_profile: long (nullable = true)
|-- compliment_writer: long (nullable = true)
|-- cool: long (nullable = true)
|-- elite: string (nullable = true)
|-- fans: long (nullable = true)
|-- friends: string (nullable = true)
|-- funny: long (nullable = true)
|-- name: string (nullable = true)
|-- review_count: long (nullable = true)
|-- useful: long (nullable = true)
|-- user_id: string (nullable = true)
|-- yelping_since: string (nullable = true)
```

User Dataset Columns: 22 | Rows: 1,968,703

```
root
|-- business_id: string (nullable = true)
|-- cool: long (nullable = true)
|-- date: string (nullable = true)
|-- funny: long (nullable = true)
|-- review_id: string (nullable = true)
|-- stars: double (nullable = true)
|-- text: string (nullable = true)
|-- useful: long (nullable = true)
|-- user_id: string (nullable = true)
```

Review Dataset Columns: 9 | Rows: 8,021,122

In [23]: `user_df.select('user_id','elite').show(5)`

```

+-----+-----+
|          user_id|          elite|
+-----+-----+
|ntlvfPzc8eglqv92...|          |
|FOBRP1BHa3WPHFB5q...|2008,2009,2010,20...|
|zZUnPeh2hEp0WydbA...|          2010|
|QaELAmRcDc5TfJEy1...|          2009|
|xvu8G900tezTzbbfq...|2009,2010,2011,20...|
+-----+-----+
only showing top 5 rows

```

### 3. Split Elite column

```

In [24]: user_elite_split = user_df.select('user_id', explode(split(user_df.elite, ',')).alias('
user_elite_split = user_elite_split.withColumn('elite', user_elite_split.elite.cast(Int
user_elite_split.show(5)
print(f'User Elite Split Dataset Columns: {len(user_elite_split.columns)} | Rows: {user

```

```

+-----+-----+
|          user_id|elite|
+-----+-----+
|ntlvfPzc8eglqv92...| null|
|FOBRP1BHa3WPHFB5q...| 2008|
|FOBRP1BHa3WPHFB5q...| 2009|
|FOBRP1BHa3WPHFB5q...| 2010|
|FOBRP1BHa3WPHFB5q...| 2011|
+-----+-----+
only showing top 5 rows

```

User Elite Split Dataset Columns: 2 | Rows: 2,125,315

In [25]: `user_elite_split.select("elite").distinct().sort('elite', ascending=False).show()`

```

+-----+
|elite|
+-----+
| 2018|
| 2017|
| 2016|
| 2015|
| 2014|
| 2013|
| 2012|
| 2011|
| 2010|
| 2009|
| 2008|
| 2007|
| 2006|
| null|
+-----+

```

```

In [26]: Elite_or_Not = user_elite_split.select('user_id',
          when(user_elite_split.elite.isNull(), "Not Elite").otherwise("Elite").alias(

```

```
Elite_or_Not.show()
```

```
+-----+-----+
|          user_id|Elite or Not|
+-----+-----+
|ntlvfPzc8eglqv92...|    Not Elite|
|FOBRP1BHa3WPHFB5q...|    Elite|
|FOBRP1BHa3WPHFB5q...|    Elite|
|FOBRP1BHa3WPHFB5q...|    Elite|
|FOBRP1BHa3WPHFB5q...|    Elite|
|FOBRP1BHa3WPHFB5q...|    Elite|
|FOBRP1BHa3WPHFB5q...|    Elite|
|FOBRP1BHa3WPHFB5q...|    Elite|
|zZUnPeh2hEp0WydbA...|    Elite|
|QaELAmRcDc5TfJEy1...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|xvu8G900tezTzbbfq...|    Elite|
|z5_82komKV3mI4ASG...|    Elite|
|ttumcu6hWshk_EJW...|    Not Elite|
+-----+-----+
only showing top 20 rows
```

```
In [27]: unique_user_df = Elite_or_Not.dropDuplicates(['user_id'])
         unique_user_df.show()
```

```
+-----+-----+
|          user_id|Elite or Not|
+-----+-----+
|---RfKzBwQ8t3wu-L...|    Not Elite|
|--1UpCuUDJQbqiuFX...|    Not Elite|
|--AGAPpP1pgp1afbq...|    Not Elite|
|--C-42rr7hPSsUROJ...|    Not Elite|
|--ChzqcPs4YFWlw1j...|    Not Elite|
|--ET3paBtrThD95dk...|    Not Elite|
|--GLTFzU93A40YB56...|    Not Elite|
|--I4wRDhmM2J2VLzK...|    Not Elite|
|--RquisWmBzcezXZr...|    Not Elite|
|--UizzbnQLZg7bEv2...|    Not Elite|
|--cd_gA-9Q8gM9P2c...|    Not Elite|
|--dhSVo0FDBiMCCwD...|    Not Elite|
|--fpTdHQOGWGbAjk9...|    Not Elite|
|--ju6XpRd0dY1Swmf...|    Not Elite|
|--oVdTxD7QVr8Y0U...|    Not Elite|
|--pWqE-KOWDwo5ADG...|    Not Elite|
|--t6W1JHbStaCp5RO...|    Not Elite|
|--tmwndDOZJwfrvvt...|    Not Elite|
|--yrdC1dIR6VYsW6k...|    Not Elite|
|-06viLTmt1RTHxxDg...|    Not Elite|
+-----+-----+
only showing top 20 rows
```

## 4. Join "Unique User" Dataset with Review Dataset

```
In [28]: user_join_review = review_df.join(unique_user_df, on = "user_id", how='left')
         print(f'User Join Review Dataset Columns: {len(user_join_review.columns)} | Rows: {user
```

User Join Review Dataset Columns: 10 | Rows: 8,021,122

## 5. Clean Data

Combined Datasets which includes elite and non-elite

```
In [29]: combine_df = user_join_review.select('review_id', 'business_id', 'stars', 'user_id', 'Elite
```

Combined (Elite and Non-Elite) Average Ratings Grouped by Business ID

```
In [30]: combine_stars_df = combine_df.groupBy("business_id").agg(F.mean('stars').alias('Stars'))
```

Elite Only Dataset

```
In [31]: elite_df = combine_df.filter(col("Elite or Not") == "Elite")
```

Elite Average Rating Grouped by Business ID

```
In [32]: elite_stars_df = elite_df.groupBy("business_id").agg(F.mean('stars').alias('Stars rated
```

Non-Elite Dataset

```
In [33]: non_elite_df = combine_df.filter(col("Elite or Not") == "Not Elite")
```

Non-Elite Average Rating Grouped by Business ID

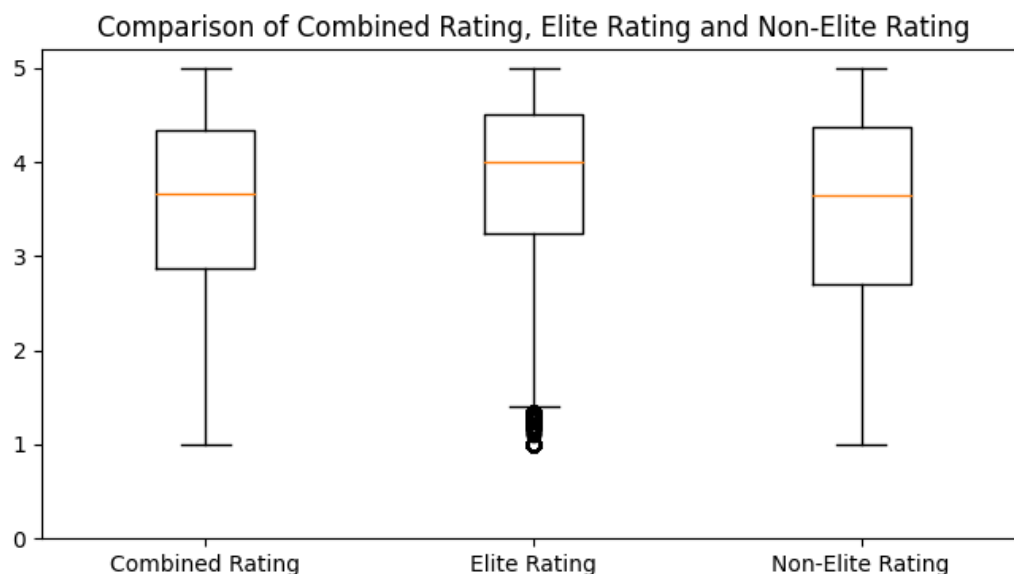
```
In [34]: non_elite_stars_df = non_elite_df.groupBy("business_id").agg(F.mean('stars').alias('Sta
```

Prepare data for plotting

```
In [35]: combined_data = combine_stars_df.toPandas()["Stars"].values.tolist()
elite_data = elite_stars_df.toPandas()["Stars rated by elite"].values.tolist()
non_elite_data = non_elite_stars_df.toPandas()["Stars rated by non elite"].values.tolis
data = [combined_data, elite_data, non_elite_data]
```

And finally, graph it!

```
In [36]: fig = plt.figure(figsize=(8, 4))
plt.boxplot(data)
plt.xticks([1, 2, 3], ['Combined Rating', 'Elite Rating', 'Non-Elite Rating'])
plt.title("Comparison of Combined Rating, Elite Rating and Non-Elite Rating")
y_ticks = np.arange(0, 6, 1)
plt.yticks(y_ticks)
%matplotlib plt
```



Clears the entire current figure with all its axes

```
In [37]: plt.clf()
plt.cla()
plt.close()
```

As we can see from the above boxplot, elite data has more outliers. Additionally, the first, third quantiles and the median of the elite ratings are also higher than the non-elites' ratings. From my point of view, I would say elite should not be trusted because they tend to give higher ratings compared with the ratings given by non-elite.

## Part V. What is the Most Recommended Restaurant?

### 1. Filter business data to collect 5 star rated restaurants

```
In [38]: five_strs_resta_df = business_df.select('business_id', 'name', 'city', 'stars', 'review'
                                                .where(col('categories').like("%Restaurants%"))
                                                .filter(col("stars") == 5))
```

### 2. Check which city has the largest number of 5 star rated restaurants

```
In [39]: city_count = five_strs_resta_df.select('city').groupby(five_strs_resta_df.city).count()
city_count.show()
```

```
+-----+-----+
|      city|count|
+-----+-----+
|  Las Vegas|   225|
|  Montréal|   171|
```



Toronto	165
Phoenix	132
Pittsburgh	82
Calgary	79
Cleveland	61
Charlotte	55
Scottsdale	43
Mesa	36
Madison	34
Mississauga	27
Henderson	22
Tempe	19
Chandler	18
Gilbert	18
Glendale	16
Laval	15
Brampton	13
Matthews	9

+-----+-----+  
only showing top 20 rows

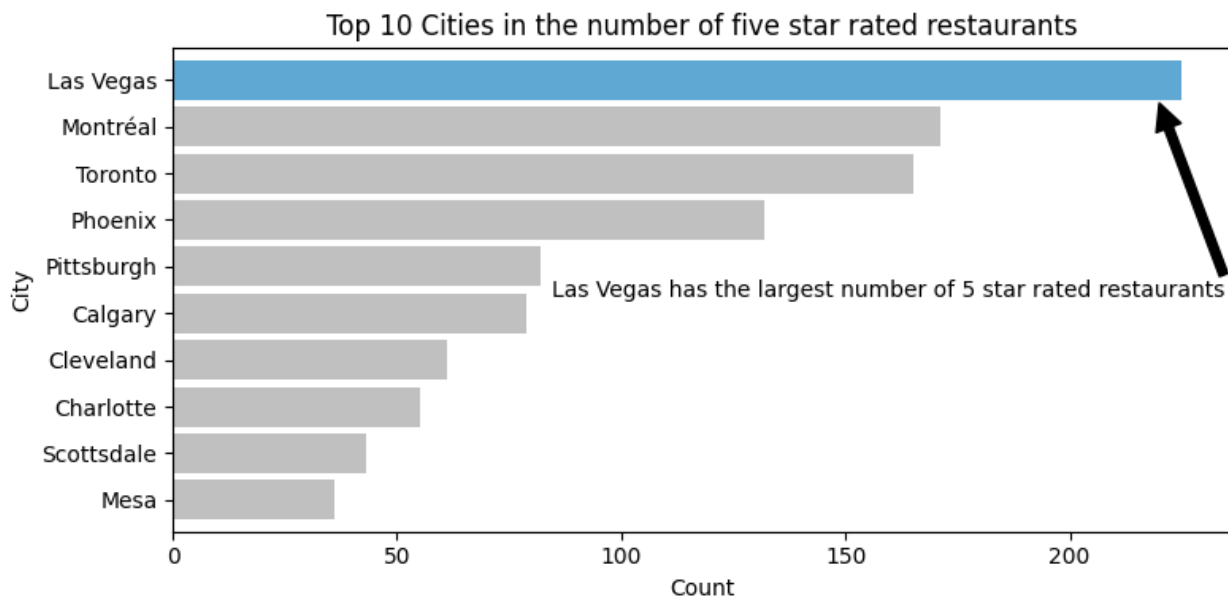
### 3. Plot the top 10 cities that have the largest number of 5 star rated restaurants

```
In [40]: top_10 = city_count.sort(desc('count')).limit(10).toPandas()

ax = top_10.plot(kind='barh', x='city', y='count',
                  figsize=(8, 4), zorder=2, width=0.85, \
                  color=['#5fa8d3', 'silver', 'silver', 'silver', 'silver', 'silver', 'silver', 'silver', 'silver', 'silver'])

ax.invert_yaxis()
ax.set_xlabel("Count")
ax.set_ylabel("City")
ax.set_title("Top 10 Cities in the number of five star rated restaurants")
ax.annotate('Las Vegas has the largest number of 5 star rated restaurants',
            xy=(220, 0.5), xycoords='data',
            xytext=(30, -90), textcoords='offset points',
            arrowprops=dict(facecolor='black'),
            horizontalalignment='right', verticalalignment='bottom')
ax.get_legend().remove()

plt.tight_layout()
%matplotlib plt
```



Clears the entire current figure with all its axes

```
In [41]: plt.clf()
plt.cla()
plt.close()
```

#### 4. Deep dive into Las Vegas and check which restaurants has the most reviews

```
In [42]: Las_Vegas_five_strs_resta = five_strs_resta_df.filter(col("city") == "Las Vegas")
Las_Vegas_top_10_most_reviews_resta = Las_Vegas_five_strs_resta.select('business_id','review_count')
Las_Vegas_top_10_most_reviews_resta.sort("review_count")
Las_Vegas_top_10_most_reviews_resta.show()
```

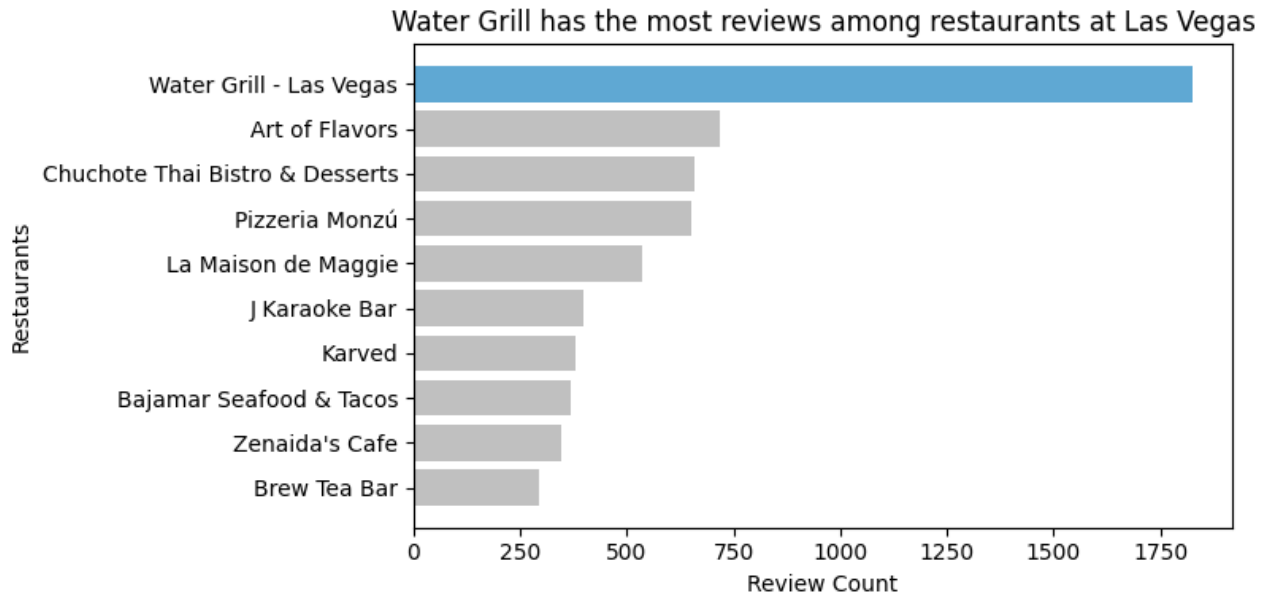
business_id	name	categories	review_count
IhNASEZ3XnBHmuuVn...	Brew Tea Bar	Restaurants, Food...	1827
8fFTJBh0RB2EKG53i...	Zenaida's Cafe	Cafes, Breakfast ...	717
G4hjhtA_wQ-tSOGpg...	Bajamar Seafood &...	Fast Food, Dive B...	658
cePE3rCuUOVSCCAHS...	Karved	Restaurants, Sand...	651
3pSUr_cdrphur06m1...	J Karaoke Bar	Nightlife, Bars, ...	534
2B46bRpDh49eDyjXG...	La Maison de Maggie	Cafes, Creperies,...	397
kafUXRNfQm3e_U14S...	Pizzeria Monzú	Italian, Pizza, B...	381
4Pl-ziyL2oerGyIP1...	Chuchote Thai Bis...	Comfort Food, Res...	370
07UMzd3i-Zk8dMeyY...	Art of Flavors	American (New), I...	345
sVjL9DQ8hYW_pS-jp...	Water Grill - Las...	Restaurants, Seafood	294

#### 5. Plot the top 10 five-star rated restaurants at Las Vegas in the number of reviews

```
In [58]: fig = plt.figure(figsize =(8, 4))
restaurants_name = Las_Vegas_top_10_most_reviews_resta.toPandas()["name"].values.tolist()
review_count = Las_Vegas_top_10_most_reviews_resta.sort('review_count').toPandas()["review_count"].values.tolist()
plt.barh(restaurants_name, review_count, color=['silver','silver','silver','silver','si
```

```
plt.xlabel("Review Count")
plt.ylabel("Restaurants")
plt.title("Water Grill has the most reviews among restaurants at Las Vegas")
plt.tight_layout()
plt.show()

%matplotlib plt
```



Clears the entire current figure with all its axes

```
In [44]: plt.clf()
plt.cla()
plt.close()
```

## 6. A glance of what type of restaurants are those top 10 restaurants in the number of reviews at Las Vegas

```
In [45]: Las_Vegas_top_10_most_reviews_resta.select('name', 'categories').sort("review_count")
```

```
+-----+-----+
|name|categories|
+-----+-----+
|Water Grill - Las Vegas|Restaurants, Seafood|
|Art of Flavors|American (New), Ice Cream & Frozen Yogurt, Restaurants, Gelato, Desserts, Food|
|Chuchote Thai Bistro & Desserts|Comfort Food, Restaurants, Thai, Beer, Wine & Spirits, Desserts, Food|
|Pizzeria Monzú|Italian, Pizza, Breakfast & Brunch, Restaurants|
|La Maison de Maggie|Cafes, Creperies, Restaurants, French, Gluten-Free|
|J Karaoke Bar|Nightlife, Bars, Restaurants, Asian Fusion, Cocktail Ba
```



and was brought out a bib (necessary) and a bowl for shells (cleaned ever 5 minutes). It was second to my favorite place for cioppino in San Francisco, but delicious nonetheless. My husband got the swordfish and really enjoyed as well. It felt very intimate, despite the size of the restaurant and all service, from the hostess (who gave me a black napkin since I was wearing black), to David, to the young man who brought over an 8 pound live crab just so we could look at it. We appreciate you!

| Was here for work this week in Las Vegas and came up on the Water Grill by accident. Myself and the rest of the staff ended up eating at this restaurant almost every night of our visit. The service is impeccable, the bartenders knew our names, knew our orders and made everything right. Food is so fresh and raw bar is incredible! The few little glitches that we did have with food were made more than right by the management and by the staff without expectation and without hesitation. I would highly recommend going here if you haven't been. For this place to be in the middle of the desert, the seafood tasted incredibly fresh. Standout dishes for me were the raw Peruvian scallops, Trojan roll and the garbanzo mussels. The bread is delicious and always comes out fresh.

| The Water Grill in the Forum Shops at Caesars Palace was an amazing experience! We were welcomed guests and treated to the very best of service Las Vegas has to offer. We had the whole Dover sole escabeche style, the sea bass which the chef was so great to accommodate and swap the cauliflower curry and purée for the gnocchi it was originally paired with (low carb choice!), and we had an amazing selection of oysters. The view is fun, people watching in Vegas out of the hot sun! The wait staff is fabulous with so much experience it's a wonder they were able to get this dream team together! Kit from Jersey City is also one of the top waiters at none other than Joel Robuchon (the mansion) in MGM Grand. He wanted to expand his horizons and is an aficionado of what the executive chef and general manager are "bringing to the table!" We had east coast and west coast oysters along with some amazing homemade gelato and sorbet. These cheese bread smelled like a small pillow of luxury (but no carbs for me... boooo) so I did enjoy and over the top selection of cheeses for my dessert. It is fair to say we will be back soon. There is so much to explore on this fresh, adventurous and aspirational seafood menu! Thanks for a wonderful time! Oh and do yourself a favor and try the Gruener!

| Omg! Locals run to this fish place! Fresh fish, amazing and impeccable service from Lewis; and the sommelier Karla, knows her wine! Antwynet, the manager, was phenomenal and this place caters to tourists and locals. Great pricing for Las Vegas and just a good experience! Trading on old Las Vegas style concierge! 5 stars and we are addicted!

| Hands down the best mussels/chorizo dish I've ever had. Must try, it's spicy, delicious, the broth is amazing. I couldn't finish, it's a huge appetizer, and a great new restaurant

| My fiancé and I made reservations on the 28th for dinner.

We were seated immediately and our waiter, Lewis W., came over and asked if we wanted drinks. Gave him our order and he made a suggestion for a slight variation on the drink I ordered; well worth the suggestion. From there, throughout dinner and dessert, we couldn't have been happier to have been taken care of by Lewis. He was awesome and we were so glad to have met him. He has to be one of the most gracious and caring waiters I have ever met. We will for sure be back and will definitely remember to ask for him. Quite a gem to have at this location.

Once again, many thanks to Lewis for making our date night wonderful.

| Our group from work ate here for dinner on a Friday evening. The food, the service, presentation. AMAZING! Nikki, Leslie, and Shannon were kind, friendly, and helpful. Started off with some clam chowder and went with the filet mignon as my entree. Others went with a mixed organic green salad and king salmon.

| We are with dear friends who picked this place after much research.. I am completely in awe. The atmosphere was great. Wonderful bar. We had Shannon as a waitress and she was extremely lovely and patient.. my friend didn't like the first glass of wine.. they didn't have any issues and Shannon went beyond to make sure there was a wine my friend liked as she listened to her :).. this is just the beginning of the experience! We ordered the 1lb king crab legs (two of them for each couple) and it was spectacular. I grew up eating crab and king crab and their king crab leg entree was by far the best I have ever tasted! We also had their oysters, etc.. still not finished!! They oysters were wonderful, the polenta was so good (and I don't like polenta at all), wedge salad (wonderful) and then we topped it off with their bread pudding which puts all the other restaurants to

shame and I have had bread pudding at many restaurants including New Orleans. Do yourself a favor and stop here and eat our menu.. you Will not be disappointed :) thank you Shannon and staff!

|Place had amazingly fresh seafood, really chill music, and an awesome vibe! As far as I know, this is the only place in LV that has fresh raw seafood, and they do it well. Phas on and Cade were awesome servers that gave us great advice on things to do in LV and what wines to get.

|We had the best seafood meal in a long time. We live up the Pac NW so i wasn't expecting to have such a great seafood meal in Vegas. Cade our bartender was exceptional. Would absolutely go back.

|Huge restaurant with lots of casual seating. No wait. We were seated immediately. Our waiter was very attentive and recommended several appetizers to share. If you love seafood then you need to check-out this place. There's an extensive menu with something for everyone. Take a break from gambling or shopping at Caesars and check-out Water Grill for a relaxing evening. We went to a show after dinner and our server made sure we were out at the right time without having to rush. Great service!

|Lewis, what great waiter who spend his times to make sure everyone understand how the menu works. Very outspoken, smile all the times, love what he does and he make that we all leave here with great foods and great feeling for his customers. I recommend him a GEM gentleman and waiter again. The Water Grill in Las Vegas, the foods is real GEM!!! Also you can bet on it.

Pat and Marlene Orlando from AZ.

|Excellent restaurant. There was so much on the menu wasn't even sure what to order. Wanted almost everything on the menu! Went with oysters and the wild Alaskan red king crab nuggets for an appetizer and then shared the wild Nootka king salmon because we had eaten earlier. The salmon was amazing. The crab and oysters were delicious too. Wanted to try the bread pudding for dessert but just was too full! The sour dough bread was also really good. Wasn't really a fan of the cheese bread but I'm sure some people would like it. Will definitely be back. Service was great. Want to try more items off this menu!

|Very nice Sea Food and excellent service. This is the place to be when you are up for some seafood in the desert.

|Natalie was our server - awesome service! She made sure to check on us often - ask how our food was, if we needed anything else, etc. she did a wonderful job of "walking" us through every part of the menu, and answering any questions we had about specific items. We would definitely recommend eating here when you visit Las Vegas! Food was excellent!

|Five star for both the food and service. The seafood is fresh and in high quality. We ordered an oyster sampler, and the oysters are fatty and tasty. The seafood platter was also very good. What is even better is that the food is not overpriced! Highly recommended!

|This years choice for our anniversary dinner. LOVE LOVE LOVE the decor and atmosphere. The clam chowder was the BEST I've had, aside from San Francisco. My husband enjoyed his tuna poké. The ribetlye was delightful and he really liked his Chilean sea bass...he doesn't care for butternut squash, but ate all of the gnocchi and puree.

|Beyond amazing! From customer service to the presentation of the food Water Grill delivers the best tasting food. Perfectly cooked and great seasoning on everything. The cucumber margarita is a must!!

|Amazing food, great service, fresh seafood! Had the spiny lobster and scallops everything was perfect!

|LOVED sitting at the raw bar!!! Great atmosphere, amazing raw bar! This staff takes its food and artistry seriously. The platters and towers are beautiful! We ate here before the Madonna debacle and, since we walked out of her show, this became the highlight of our evening! Ask questions about the oyster choices! I had a sampling and they were all

amazing! Sushi delicious, too! AND THE HAMACHI NACHOS! Yum!

1

+

—

—

—

—

—

—

—

—

—

—

—

—

—

—

O

only showing top 20 rows

## 9. WordCloud for the written reviews of the Water Grill

```
In [60]: text = WaterGrill_review.toPandas().values
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'white',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (8, 4),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.title("WordCloud for the written reviews of the Water Grill")
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()

%matplotlib plt
```

