STA9760 Project 2

Yu Qiao Chen

Analysis of Yelp Business Intelligence Data

Installation and Initial Setup

Begin by installing the necessary libraries that you may need to conduct your analysis. At the very least, you must install pandas and matplotlib

In [1]:
```
sc.list_packages()
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1606251756147_0001	pyspark	idle	Link	Link	✔

```
SparkSession available as 'spark'.

Package                    Version
-------------------------- ---------
beautifulsoup4             4.9.1
boto                       2.49.0
click                      7.1.2
jmespath                   0.10.0
joblib                     0.16.0
lxml                       4.5.2
mysqlclient                1.4.2
nltk                       3.5
nose                       1.3.4
numpy                      1.16.5
pip                        9.0.1
py-dateutil                2.2
python37-sagemaker-pyspark 1.4.0
pytz                       2020.1
PyYAML                     5.3.1
regex                      2020.7.14
setuptools                 28.8.0
```

```
six                        1.13.0
soupsieve                  1.9.5
tqdm                       4.48.2
wheel                      0.29.0
windmill                   1.6
```

In [2]:

```
sc.install_pypi_package("pandas==1.1.4")
sc.install_pypi_package("matplotlib==3.3.3")
sc.install_pypi_package("seaborn==0.11.0")
sc.install_pypi_package("pyparsing==2.4.7")
sc.install_pypi_package("cycler==0.10.0")
sc.install_pypi_package("kiwisolver==1.3.1")
sc.install_pypi_package("python-dateutil==2.8.1")
sc.install_pypi_package("scipy==1.5.4")
```

```
Collecting pandas==1.1.4
  Downloading https://files.pythonhosted.org/packages/bf/4c/cb7da76f3a5e077e545f9cf8575b8f488a4e8ad60490838f89c5cdd5bb57/
pandas-1.1.4-cp37-cp37m-manylinux1_x86_64.whl (9.5MB)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib64/python3.7/site-packages (from pandas==1.1.4)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (from pandas==1.1.4)
Collecting python-dateutil>=2.7.3 (from pandas==1.1.4)
  Downloading 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>=2.7.3->pandas==
1.1.4)
Installing collected packages: python-dateutil, pandas
Successfully installed pandas-1.1.4 python-dateutil-2.8.1

Collecting matplotlib==3.3.3
  Downloading https://files.pythonhosted.org/packages/30/f2/10c822cb0ca5ebec58bd1892187bc3e3db64a867ac26531c6204663fc218/
matplotlib-3.3.3-cp37-cp37m-manylinux1_x86_64.whl (11.6MB)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib64/python3.7/site-packages (from matplotlib==3.3.3)
Requirement already satisfied: python-dateutil>=2.1 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages (from matplot
lib==3.3.3)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 (from matplotlib==3.3.3)
  Downloading https://files.pythonhosted.org/packages/8a/bb/488841f56197b13700afd5658fc279a2025a39e22449b7cf29864669b15d/
pyparsing-2.4.7-py2.py3-none-any.whl (67kB)
Collecting pillow>=6.2.0 (from matplotlib==3.3.3)
  Downloading https://files.pythonhosted.org/packages/af/fa/c1302a26d5e1a17fa8e10e43417b6cf038b0648c4b79fcf2302a4a0c5d30/
Pillow-8.0.1-cp37-cp37m-manylinux1_x86_64.whl (2.2MB)
Collecting cycler>=0.10 (from matplotlib==3.3.3)
  Downloading https://files.pythonhosted.org/packages/f7/d2/e07d3ebb2bd7af696440ce7e754c59dd546ffe1bbe732c8ab68b9c834e61/
cycler-0.10.0-py2.py3-none-any.whl
Collecting kiwisolver>=1.0.1 (from matplotlib==3.3.3)
  Downloading https://files.pythonhosted.org/packages/d2/46/231de802ade4225b76b96cffe419cf3ce52bbe92e3b092cf12db7d11c207/
kiwisolver-1.3.1-cp37-cp37m-manylinux1_x86_64.whl (1.1MB)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.1->matplotlib=
```

```
=3.3.3)
Installing collected packages: pyparsing, pillow, cycler, kiwisolver, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.3.3 pillow-8.0.1 pyparsing-2.4.7

Collecting seaborn==0.11.0
  Downloading https://files.pythonhosted.org/packages/bc/45/5118a05b0d61173e6eb12bc5804f0fbb6f196adb0a20e0b16efc2b8e98be/
seaborn-0.11.0-py3-none-any.whl (283kB)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib64/python3.7/site-packages (from seaborn==0.11.0)
Collecting scipy>=1.0 (from seaborn==0.11.0)
  Downloading https://files.pythonhosted.org/packages/dc/7e/8f6a79b102ca1ea928bae8998b05bf5dc24a90571db13cd119f275ba6252/
scipy-1.5.4-cp37-cp37m-manylinux1_x86_64.whl (25.9MB)
Requirement already satisfied: matplotlib>=2.2 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages (from seaborn==0.1
1.0)
Requirement already satisfied: pandas>=0.23 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages (from seaborn==0.11.
0)
Requirement already satisfied: python-dateutil>=2.1 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages (from matplot
lib>=2.2->seaborn==0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /mnt/tmp/1606252837674-0/lib/python3.7/site-pa
ckages (from matplotlib>=2.2->seaborn==0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages (from matplotlib>=2.
2->seaborn==0.11.0)
Requirement already satisfied: cycler>=0.10 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages (from matplotlib>=2.2
->seaborn==0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages (from matplotlib
>=2.2->seaborn==0.11.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (from pandas>=0.23->seaborn==0.11.
0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.1->matplotlib>
=2.2->seaborn==0.11.0)
Installing collected packages: scipy, seaborn
Successfully installed scipy-1.5.4 seaborn-0.11.0

Requirement already satisfied: pyparsing==2.4.7 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages

Requirement already satisfied: cycler==0.10.0 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages
Requirement already satisfied: six in /usr/local/lib/python3.7/site-packages (from cycler==0.10.0)

Requirement already satisfied: kiwisolver==1.3.1 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages

Requirement already satisfied: python-dateutil==2.8.1 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil==2.8.1)

Requirement already satisfied: scipy==1.5.4 in /mnt/tmp/1606252837674-0/lib/python3.7/site-packages
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib64/python3.7/site-packages (from scipy==1.5.4)
```

Checking the packages after installation

In [3]:
```
sc.list_packages()
```

```
Package                    Version
-------------------------- ---------
beautifulsoup4             4.9.1
boto                       2.49.0
click                      7.1.2
cycler                     0.10.0
jmespath                   0.10.0
joblib                     0.16.0
kiwisolver                 1.3.1
lxml                       4.5.2
matplotlib                 3.3.3
mysqlclient                1.4.2
nltk                       3.5
nose                       1.3.4
numpy                      1.16.5
pandas                     1.1.4
Pillow                     8.0.1
pip                        9.0.1
py-dateutil                2.2
pyparsing                  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                    0.11.0
setuptools                 28.8.0
six                        1.13.0
soupsieve                  1.9.5
tqdm                       4.48.2
wheel                      0.29.0
windmill                   1.6
```

Importing

Now, import the installed packages from the previous block below.

In [4]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pyparsing as pp
import cycler
import kiwisolver as kiwi
import dateutil
```

```python
import scipy as sci

from pyspark.sql.functions import explode, split
import pyspark.sql.functions as F
```

Loading Data

We are finally ready to data. Using spark load the data from S3 into a dataframe object that we can manipulate further down in our analysis.

In [5]:
```python
business = spark.read.json('s3://sta9760yelpproject02/yelp_academic_dataset_business.json')
```

Overview of Data

Business

Display the number of rows and columns in our dataset.

In [6]:
```python
print(f"Rows: {business.count()}, Columns: {len(business.columns)}")
```

```
Rows: 209393, Columns: 14
```

Display the DataFrame Schema below.

In [7]:
```python
business.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 [8]:
```
business.select("business_id", "name", "city", "state", "categories").show(5)
```

```
+--------------------+--------------------+---------------+-----+--------------------+
|         business_id|                name|           city|state|          categories|
+--------------------+--------------------+---------------+-----+--------------------+
|f9NumwFMBDn751xgF...|The Range At Lake...|      Cornelius|   NC|Active Life, Gun/...|
|Yzvjg0SayhoZgCljU...|   Carlos Santo, NMD|     Scottsdale|   AZ|Health & Medical,...|
|XNoUzKckATkOD1hP6...|             Felinus|       Montreal|   QC|Pets, Pet Service...|
|6OAZjbxqM5ol29BuH...|Nevada House of Hose|North Las Vegas|   NV|Hardware Stores, ...|
|51M2Kk903DFYI6gnB...|USE MY GUY SERVIC...|           Mesa|   AZ|Home Services, Pl...|
+--------------------+--------------------+---------------+-----+--------------------+
only showing top 5 rows
```

Analyzing Categories

Let's now answer this question: 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 questions such as:

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

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.

Implement the code necessary to derive the table described from your original yelp dataframe.

Extracting only business ID and categories

In [9]:
```
business_id_cat = business.select("business_id", "categories")
```

Using Explode to separate the category list

In [10]:
```
business_cat_exploded = business_id_cat.withColumn('categories', explode(split('categories', ", ")))
```

Display the first 5 rows of your association table below

In [11]:
```
business_cat_exploded.show(5)
```

```
+--------------------+----------------+
|         business_id|      categories|
```

```
+--------------------+----------------+
|f9NumwFMBDn751xgF...|     Active Life|
|f9NumwFMBDn751xgF...|Gun/Rifle Ranges|
|f9NumwFMBDn751xgF...|     Guns & Ammo|
|f9NumwFMBDn751xgF...|        Shopping|
|Yzvjg0SayhoZgCljU...|Health & Medical|
+--------------------+----------------+
only showing top 5 rows
```

Total Unique Categories

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

Below, implement the code necessary to calculate this figure.

In [12]:
```
business_cat_exploded.select('categories').distinct().count()
```

```
1336
```

Top Categories By Business

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

Counts of Businesses / Category

So now, let's unroll our distinct count a bit and display the per count value of businesses per category.

The expected output should be:

category	count
a	15
b	2
c	45

Or something to that effect.

In [13]:
```
business_cat_exploded.groupby('categories').count().show()
```

```
+--------------------+-----+
|          categories|count|
+--------------------+-----+
|      Paddleboarding|   36|
|      Dermatologists|  341|
|        Aerial Tours|   28|
|         Hobby Shops|  828|
|          Bubble Tea|  720|
|             Embassy|   13|
|             Tanning|  938|
|            Handyman|  682|
|      Aerial Fitness|   29|
|             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|
|              Doulas|   45|
+--------------------+-----+
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.

HINT: don't forget about the matplotlib magic!

Load spark df onto a pandas df

In [14]:

```
business_df = business_cat_exploded.toPandas()
business_df
```

	business_id	categories
0	f9NumwFMBDn751xgFiRbNA	Active Life
1	f9NumwFMBDn751xgFiRbNA	Gun/Rifle Ranges
2	f9NumwFMBDn751xgFiRbNA	Guns & Ammo
3	f9NumwFMBDn751xgFiRbNA	Shopping

```
4       Yzvjg0SayhoZgCljUJRF9Q       Health & Medical
...                        ...                    ...
872789  RSSIsgO000OuWQTRoITacpA                  Pets
872790  tOcYmewXFhQeZh3V42ymwg           Tax Services
872791  tOcYmewXFhQeZh3V42ymwg  Professional Services
872792  tOcYmewXFhQeZh3V42ymwg            Accountants
872793  tOcYmewXFhQeZh3V42ymwg     Financial Services

[872794 rows x 2 columns]
```

Grouping the DF by category and extracting only the top 20 results

In [15]:

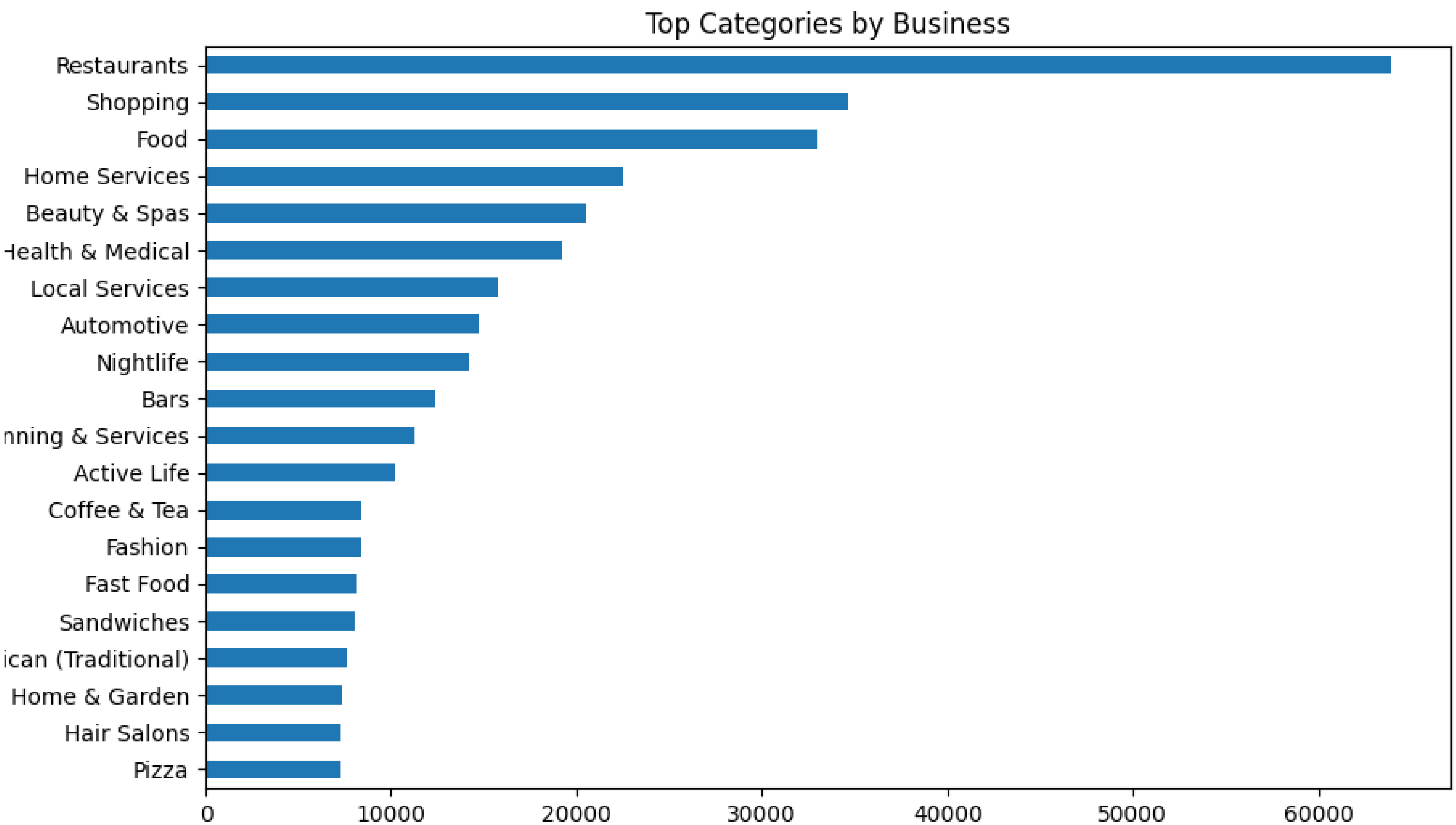
```
top20_business = business_df.groupby('categories').count().sort_values('business_id', ascending = False)[0:20]
top20_business
```

```
                           business_id
categories
Restaurants                      63944
Shopping                         34644
Food                             32991
Home Services                    22487
Beauty & Spas                    20520
Health & Medical                 19227
Local Services                   15783
Automotive                       14720
Nightlife                        14211
Bars                             12400
Event Planning & Services        11263
Active Life                      10225
Coffee & Tea                      8415
Fashion                           8374
Fast Food                         8106
Sandwiches                        8064
American (Traditional)            7596
Home & Garden                     7331
Hair Salons                       7303
Pizza                             7302
```

In [16]:

```
plt.figure(figsize = (10,6))
top20_business.sort_values('business_id', ascending = True).plot(kind = 'barh',
                                                                 figsize = (10,6),
                                                                 legend = False,
                                                                 title = 'Top Categories by Business',
                                                                 xlabel = 'Count', ylabel = 'Category')

%matplot plt
```



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.

Loading Review Data

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

In [17]:
```
review = spark.read.json('s3://sta9760yelpproject02/yelp_academic_dataset_review.json')
review.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.

In [18]:
```
review_star = review.select('business_id', 'stars')
review_star.show(5)
```

```
+--------------------+-----+
|         business_id|stars|
+--------------------+-----+
|-MhfebM0QIsKt87iD...|  2.0|
|lbrU8StCq3yDfr-QM...|  1.0|
|HQl28KMwrEKHqhFrr...|  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 [19]:
```
review_star_mean = review_star.groupby('business_id').mean()
review_star_mean.show(5)
```

```
+--------------------+------------------+
|         business_id|        avg(stars)|
```

```
+--------------------+------------------+
|VHsNB3pdGVcRgs6C3...| 3.411764705882353|
|RMjCnixEY5i12Ciqn...|3.5316455696202533|
|ipFreSFhjClfNETuM...|               2.6|
|dLDMU8bOLnkDTmPUr...| 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 [20]:
```python
business_review = business.join(review_star_mean,
                                business.business_id == review_star_mean.business_id).drop(review_star_mean.business_id)
business_review.show(5)
```

```
+--------------------+--------------------+--------------------+--------------------+---------+--------------------+-----
--+-------------+-------------+--------------------+-----------+------------+-----+-----+----------------+
|             address|          attributes|         business_id|          categories|     city|               hours|is_op
en|     latitude|    longitude|                name|postal_code|review_count|stars|state|      avg(stars)|
+--------------------+--------------------+--------------------+--------------------+---------+--------------------+-----
--+-------------+-------------+--------------------+-----------+------------+-----+-----+----------------+
|3355 Las Vegas Bl...|[,, 'full_bar', {...|--9e1ONYQuAa-CB_R...|Seafood, Cajun/Cr...|Las Vegas|[17:0-22:30, 17:0...|
1|    36.123183|   -115.16919|Delmonico Steakhouse|      89109|        1759|  4.0|   NV|4.11784140969163|
|      1040 E Main St|[,,, {'touristy':...|--FnvijzY20d1nk9H...|Restaurants, Mexican|     Mesa|[9:0-21:0, 9:0-21...|
1|33.4153931367|-111.808122173|Mr. Pancho Mexica...|      85203|           8|  4.5|   AZ|             4.5|
| 1645 E Roosevelt St|[,,,,,,,,,, True,,...|--phjqoPSPa8sLmUV...|Medical Centers, ...|  Phoenix|[8:0-17:0, 0:0-0:...|
1|   33.4580955|  -112.0460477|Maricopa County D...|      85006|          12|  4.0|   AZ|            3.75|
|9495 Las Vegas Bl...|[,, 'full_bar', {...|--q7kSBRb0vWC8lSk...|Pizza, Restaurant...|Las Vegas|                null|
0|   36.0166929|   -115.173115|Double Play Sport...|      89123|           7|  4.0|   NV|             4.0|
|5970 S Cooper Rd,...|[True,,,,,,,,, Tr...|--ttCFj_csKJhxnaM...|Cosmetic Dentists...| Chandler|[7:0-13:0, 7:0-15...|
1|    33.219528|    -111.80753| Impressions Dental|      85249|          45|  2.5|   AZ|          2.6875|
+--------------------+--------------------+--------------------+--------------------+---------+--------------------+-----
--+-------------+-------------+--------------------+-----------+------------+-----+-----+----------------+
only showing top 5 rows
```

In [21]:
```python
business_review_skew = business_review.select('avg(stars)', 'stars', 'name', 'city', 'state', 'business_id')
business_review_skew
```

```
DataFrame[avg(stars): double, stars: double, name: string, city: string, state: string, business_id: string]
```

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:

(row['avg(stars)'] - row['stars']) / row['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 [22]:
```
business_review_skew = business_review_skew.withColumn('skew', (F.col('avg(stars)') - F.col('stars')) / F.col('stars'))
business_review_skew.show(5)
```

```
+----------------+-----+--------------------+---------+-----+--------------------+--------------------+
|      avg(stars)|stars|                name|     city|state|         business_id|                skew|
+----------------+-----+--------------------+---------+-----+--------------------+--------------------+
|4.11784140969163|  4.0|Delmonico Steakhouse|Las Vegas|   NV|--9e1ONYQuAa-CB_R...|0.029460352422907565|
|             4.5|  4.5|Mr. Pancho Mexica...|     Mesa|   AZ|--FnvijzY20d1nk9H...|                 0.0|
|            3.75|  4.0|Maricopa County D...|  Phoenix|   AZ|--phjqoPSPa8sLmUV...|             -0.0625|
|             4.0|  4.0|Double Play Sport...|Las Vegas|   NV|--q7kSBRb0vWC8lSk...|                 0.0|
|          2.6875|  2.5|  Impressions Dental| Chandler|   AZ|--ttCFj_csKJhxnaM...|               0.075|
+----------------+-----+--------------------+---------+-----+--------------------+--------------------+
only showing top 5 rows
```

And finally, graph it!

Putting business_review_skew from last step into a Pandas DF

In [23]:
```
business_review_skew_df = business_review_skew.toPandas()
```

Get skew column into a new Pandas DF

In [24]:
```
skew = business_review_skew_df["skew"]
skew
```

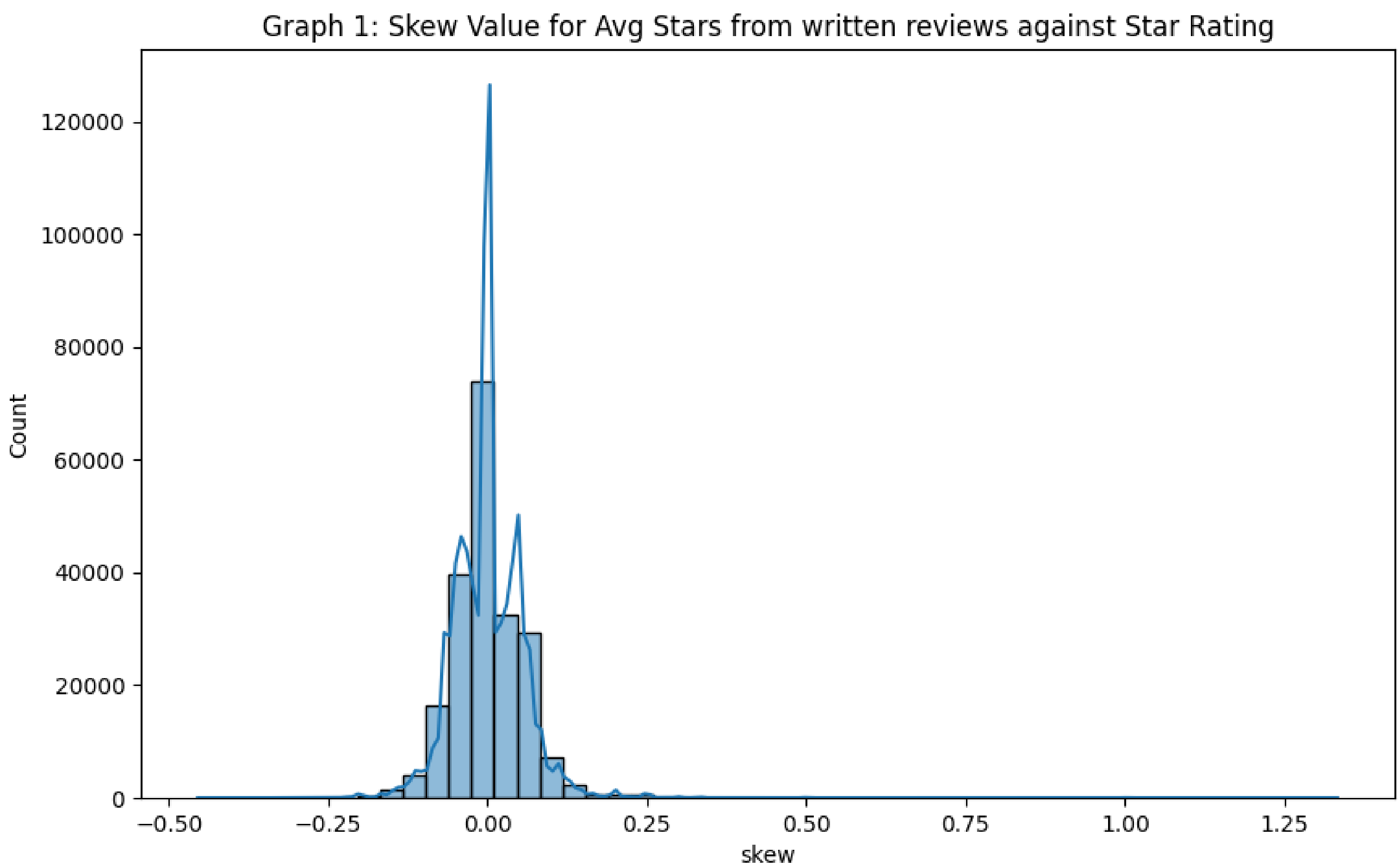
```
0         0.029460
1         0.000000
2        -0.062500
3         0.000000
4         0.075000
            ...
209388    0.000000
209389   -0.066667
209390   -0.066667
209391    0.037037
209392   -0.047619
Name: skew, Length: 209393, dtype: float64
```

Graphing the result

In [25]:
```
plt.figure(figsize = (10,6))

sns.histplot(data = skew,
             bins = 50,
             kde = True).set_title("Graph 1: Skew Value for Avg Stars from written reviews against Star Rating")

plt.show()
%matplot plt
```

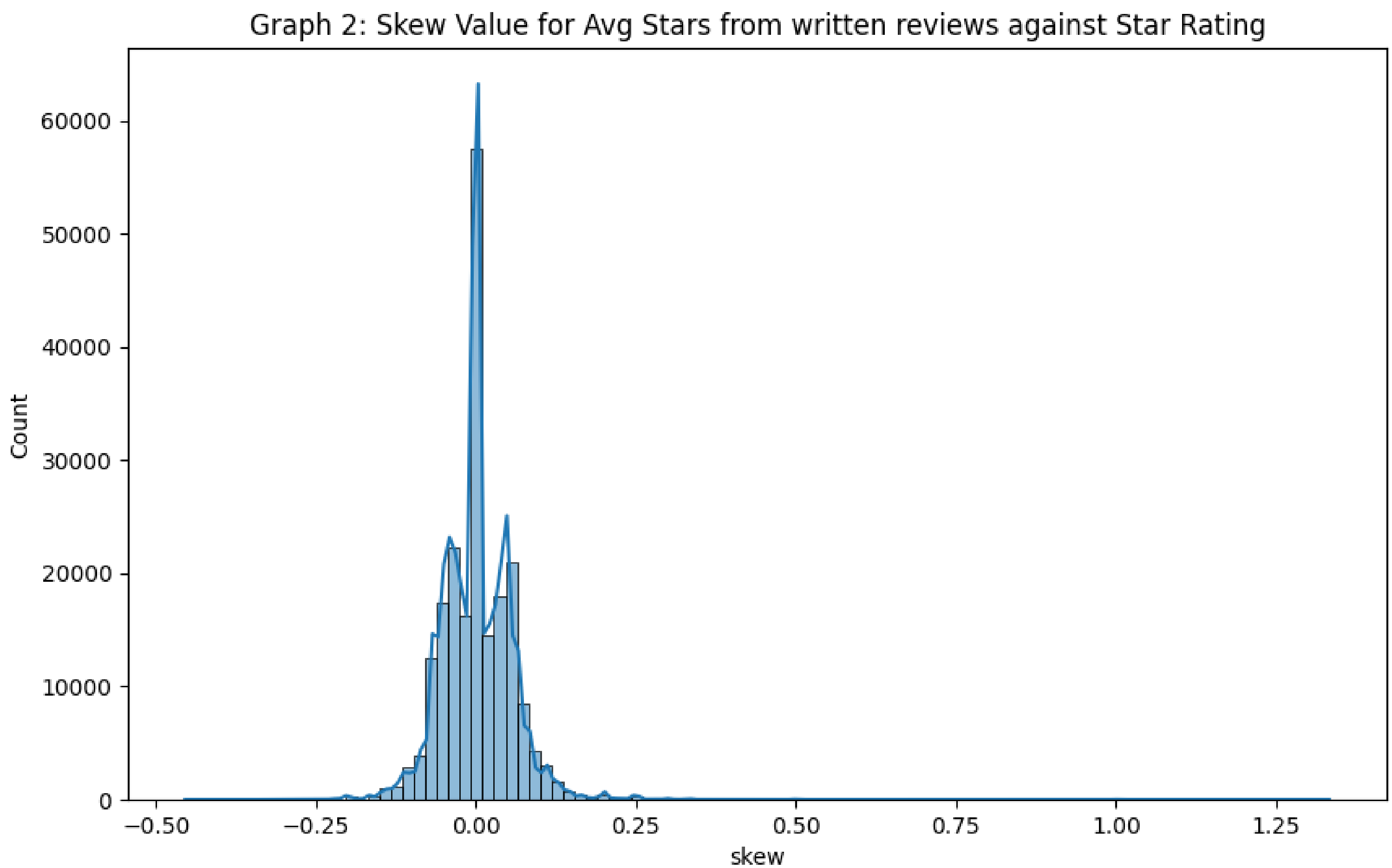


In [26]:

```
plt.figure(figsize = (10,6))

sns.histplot(data = skew,
             bins = 100,
             kde = True).set_title("Graph 2: Skew Value for Avg Stars from written reviews against Star Rating")

plt.show()
%matplot plt
```



Counting the number of positives, negatives, and neutral

In [27]:
```
positive = 0
negative = 0
neutral = 0

for i in skew.index:
    if skew[i] == 0:
        neutral += 1
    if skew[i] > 0:
        positive += 1
    if skew[i] < 0:
        negative += 1

print(f"Postive: {positive}, Negative: {negative}, Neutral: {neutral}")
```

```
Postive: 77821, Negative: 80982, Neutral: 50590
```

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

If you compare only positive and negative, then the Yelp Review is skewed slightly more towards **negative**, meaning more people with written records were more disatisfied.

Even though the distribution of the graph appears normally distributed, the count of negative is 3000 records more so it is slightly actually skewed to the right(negative). However, if you consider combining positive and neutral into one category, then that can be translated into more restaurants actually live up to their Star Rating because a skew value of "0" means the customer agree with the restaurants star rating. It is a neutral rating from written records.

Should the Elite be Trusted? (Or, some other analysis of your choice)

For the final portion - you have a choice:

- Try and analyze some interesting dimension to this data. The ONLY requirement is that you must use the Users dataset and join on either the business* or reviews** dataset
- Or, you may try and answer the question posed: how accurate or close are the ratings of an "elite" user (check Users table schema) vs the actual business rating.

Feel free to use any and all methodologies at your disposal - only requirement is you must render one visualization in your analysis

I will be answering the question:

Are elite user ratings skewed more towards positive or negative compared to non-elite users?

I will use the similar method to measure the skewness value from the last review analysis. Conceptual Workflow:

1. Separate the User dataset into elite user subset and non-elite users subset.
2. Join each dataset to the reviews dataset.
3. Calculate the average rating by business_id from the reviews dataset.
4. Join both dataset to business
5. Calcualte skew value for both elite user and non-elite user.
6. Calculate the difference from step 5.
7. Graph and analyze.

I will calculate and compare the restaurant's average rating from all written elite user review to its Star rating

Load User Data Set

In [28]:
```
user = spark.read.json('s3://sta9760yelpproject02/yelp_academic_dataset_user.json')
```

Check Schema and the number of rows

In [29]:
```
user.printSchema()
user.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)
```

```
1968703
```

Filter out the dataset to two: any users who were never elite users versus elite users

Note: I am considering all users as elite if they were at least an elite user once

In [30]:
```
elite_user = user.filter(F.col('elite').contains('2'))
print(f"There are {elite_user.count()} elite users")
```

```
There are 75961 elite users
```

Making a list of elite user user_id to filter out non-elites

In [31]:
```
elite_user_id = list(elite_user.select('user_id').toPandas()['user_id'])
```

Filter Out the non-elites

In [32]:
```
non_elite = user.filter(F.col('user_id').isin(elite_user_id) == False)
print(f"There are {non_elite.count()} non-elite users")
```

```
There are 1892742 non-elite users
```

In [33]:
```
non_elite.select('user_id', 'elite').show(5)
```

```
+--------------------+-----+
|             user_id|elite|
+--------------------+-----+
|ntlvfPzc8eglqvk92...|     |
|ttumcu6hWshk_EJVW...|     |
```

```
|UYACF30806j2mfbB5...|     |
|f6YuZP6iennHFVlnF...|     |
|HwPGLzF_uXB3MF8bc...|     |
+--------------------+-----+
only showing top 5 rows
```

In [34]:

```
elite_user.select('elite').show(5)
```

```
+--------------------+
|               elite|
+--------------------+
|2008,2009,2010,20...|
|                2010|
|                2009|
|2009,2010,2011,20...|
|                2007|
+--------------------+
only showing top 5 rows
```

Joining user and review dataset together for both dataset

In [35]:

```
#elite

elite_user_review = elite_user.join(review, elite_user.user_id == review.user_id).drop(review.user_id)
elite_user_review.select('stars', 'user_id', 'business_id').show(20)
```

```
+-----+--------------------+--------------------+
|stars|             user_id|         business_id|
+-----+--------------------+--------------------+
|  4.0|1Dul59QEe-Q-7OQHT...|-8F04F54iDT6VgWPC...|
|  5.0|3pMczoCBOSKBcqMhV...|p2OOk46G_AOO0nCWl...|
|  1.0|jO44Apni7iJZVVK4H...|jyFoxS8MofdpkAAK6...|
|  4.0|RO78oDy7vbEcOJU8a...|ewty6EB70nwPJsUkA...|
|  5.0|TFxeEvpjMNQ3AWL49...|0M3KCmdY-_xlIu5vE...|
|  5.0|Fl1oTs6usaCfyjLnY...|-h0o-BilkKaCa7HX9...|
|  5.0|RO78oDy7vbEcOJU8a...|DEtOIjhV0MWZ8fD8-...|
|  2.0|LEr8vS6PRymCg-SJH...|Jt28TYWanzKrJYYr0...|
|  5.0|M7vDDzoPNQDN2FdTc...|NFm869_w6cvVaWaNp...|
|  4.0|Ania9MCwET-TBzVjV...|Da6eZFThE9xanUAGN...|
|  5.0|iQ0TzrwN4BgflE8Ao...|jAIeziQkY_JScpBT1...|
|  5.0|BwfJx0BwTT34qhA0U...|DIUK7_PjGCOMcpS2f...|
|  2.0|fm2npkf_1BNUPRZQb...|u_vPjx925UPEG9DFO...|
|  1.0|jO44Apni7iJZVVK4H...|4s-eHLQcoMqAgfzbi...|
|  4.0|TFxeEvpjMNQ3AWL49...|ZxbUza8_Y17R18WcW...|
|  4.0|jO44Apni7iJZVVK4H...|MjAuGkgPjNLyMrN74...|
|  5.0|TFxeEvpjMNQ3AWL49...|CfxVkwEJk1NAqgqMS...|
```

```
|  4.0|ibuFkg04Hc8MkRy6J...|AFgPDQm_pczo5uV3y...|
|  4.0|ibuFkg04Hc8MkRy6J...|AFgPDQm_pczo5uV3y...|
|  3.0|TFxeEvpjMNQ3AWL49...|44a_evCaWZ63cPA1k...|
+-----+--------------------+--------------------+
only showing top 20 rows
```

In [36]:
```
#non-elite

non_elite_review = non_elite.join(review, non_elite.user_id == review.user_id).drop(review.user_id)
non_elite_review.select('stars', 'user_id', 'business_id').show(20)
```

```
+-----+--------------------+--------------------+
|stars|             user_id|         business_id|
+-----+--------------------+--------------------+
|  5.0|---RfKzBwQ8t3wu-L...|Z3ZSar8IVAR2qIupq...|
|  1.0|--1UpCuUDJQbqiuFX...|kJhQq1BFz7lOYLve7...|
|  5.0|--1UpCuUDJQbqiuFX...|EpPOZAG0u7qHP-jv5...|
|  5.0|--AGAPpP1pgp1afbq...|OLmcIJ7VBCxaYhZSN...|
|  5.0|--AGAPpP1pgp1afbq...|WoiOpMEcbAfOqNYXq...|
|  4.0|--C-42rr7hPSsUROJ...|L-_-9JNAb6UDyq7wa...|
|  2.0|--ChzqcPs4YFWlw1j...|6pG7n8Rx_7ZXeQQk6...|
|  4.0|--ChzqcPs4YFWlw1j...|4KmrrhtfnngTVFa2d...|
|  4.0|--ChzqcPs4YFWlw1j...|AMTNJbYbu0OMMAkx4...|
|  3.0|--ChzqcPs4YFWlw1j...|KVsv8wRGnLX8QWoNZ...|
|  4.0|--ChzqcPs4YFWlw1j...|F9CcIFltPDXiOkCCF...|
|  5.0|--ET3paBtrThD95dk...|QZV9hW3WP9o9SmmV2...|
|  5.0|--GLTFzU93A40YB56...|pT6baSMzC6rZfwhp_...|
|  2.0|--I4wRDhmM2J2VLzK...|JmI9nslLD7KZqRr__...|
|  5.0|--RquisWmBzcezXZr...|HW7JPZBImm3tyEpDg...|
|  4.0|--RquisWmBzcezXZr...|XNFA-aJFX8IQjol8D...|
|  5.0|--RquisWmBzcezXZr...|W2Vis19kUa7kP6GkS...|
|  2.0|--UizzbnQlZg7bEv2...|hDD6-yk1yuuRIvfdt...|
|  4.0|--cd_gA-9Q8gM9P2c...|9Eghhu_LzEJgDKNgi...|
|  3.0|--cd_gA-9Q8gM9P2c...|fQwB9Z98YEhkJit7c...|
+-----+--------------------+--------------------+
only showing top 20 rows
```

Before joining the elite_user_review and non_elite_review dataset with the business dataset, we must group the ratings by business id and calculate its mean average written review rating

We need to also change the column name to avoid duplicate column name

In [37]:
```
#elite

elite_user_review_grouped = elite_user_review.select('business_id', 'stars').groupby('business_id').mean()
```

```
elite_user_review_grouped = elite_user_review_grouped.withColumnRenamed('avg(stars)', 'elite_avg(stars)')
elite_user_review_grouped.show(5)
```

```
+--------------------+------------------+
|         business_id|  elite_avg(stars)|
+--------------------+------------------+
|VHsNB3pdGVcRgs6C3...|               4.0|
|-I06hkMFrX0KBqu61...|               5.0|
|RMjCnixEY5i12Ciqn...|3.3461538461538463|
|ipFreSFhjClfNETuM...|3.0588235294117645|
|Qm2datcYBPXrPATVG...| 4.666666666666667|
+--------------------+------------------+
only showing top 5 rows
```

In [38]:

```
#non-elite

non_elite_review_grouped = non_elite_review.select('business_id', 'stars').groupby('business_id').mean()
non_elite_review_grouped = non_elite_review_grouped.withColumnRenamed('avg(stars)', 'non_elite_avg(stars)')
non_elite_review_grouped.show(5)
```

```
+--------------------+--------------------+
|         business_id|non_elite_avg(stars)|
+--------------------+--------------------+
|oFsufzhFo0QUlgkXd...|                 3.0|
|uC3qwaxsOkdJzpOc0...|  3.2488372093023257|
|VmSrPPO2WXmOKjUW7...|   3.201058201058201|
|--9e1ONYQuAa-CB_R...|    4.08596214511041|
|eKznX8VTfcQrjCqXp...|   4.406976744186046|
+--------------------+--------------------+
only showing top 5 rows
```

Joining elite_user_review_grouped with business_review_skew from previous analysis first, then with non_elites

In [39]:

```
#first join
business_review_skew_elite = business_review_skew.join(elite_user_review_grouped,
                                                       business_review_skew.business_id == elite_user_review_grouped.busi

#second join
business_review_skew_elite = business_review_skew_elite.join(non_elite_review_grouped,
                                                             business_review_skew_elite.business_id == non_elite_review_g
business_review_skew_elite.show(5)
```

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

```
|       avg(stars)|stars|                name|     city|state|         business_id|                skew|  elite_avg(star
s)|non_elite_avg(stars)|
+-----------------+-----+--------------------+---------+-----+--------------------+--------------------+---------------
--+--------------------+
|  4.11784140969163|  4.0|Delmonico Steakhouse|Las Vegas|   NV|--9e1ONYQuAa-CB_R...|0.029460352422907565|4.19160583941605
85|    4.08596214511041|
|             3.75|  4.0|Maricopa County D...|  Phoenix|   AZ|--phjqoPSPa8sLmUV...|             -0.0625|
4.0|               3.625|
|              4.0|  4.0|Double Play Sport...|Las Vegas|   NV|--q7kSBRb0vWC8lSk...|                 0.0|
4.0|                 4.0|
| 4.976744186046512|  5.0|   Kidz Cuts By Lori|Henderson|   NV|-0ZO00Vm2ADchytlE...|-0.00465116279069...|
5.0|  4.9743589743589745|
|3.8107142857142855|  4.0|Río Mirage Café y...|El Mirage|   AZ|-1VaIJza42Hjev6uk...|-0.04732142857142...| 3.7931034482758
62|   3.812749003984064|
+-----------------+-----+--------------------+---------+-----+--------------------+--------------------+---------------
--+--------------------+
only showing top 5 rows
```

Calculating a skew value

In [40]:

```python
business_review_skew_elite = business_review_skew_elite.withColumn('skew_elite',
                                                                   (F.col('elite_avg(stars)') - F.col('stars')) / F.col('

business_review_skew_elite.show(5)
```

```
+-----------------+-----+--------------------+---------+-----+--------------------+--------------------+---------------
--+--------------------+--------------------+
|       avg(stars)|stars|                name|     city|state|         business_id|                skew|  elite_avg(star
s)|non_elite_avg(stars)|          skew_elite|
+-----------------+-----+--------------------+---------+-----+--------------------+--------------------+---------------
--+--------------------+--------------------+
|  4.11784140969163|  4.0|Delmonico Steakhouse|Las Vegas|   NV|--9e1ONYQuAa-CB_R...|0.029460352422907565|4.19160583941605
85|    4.08596214511041|0.047901459854014616|
|             3.75|  4.0|Maricopa County D...|  Phoenix|   AZ|--phjqoPSPa8sLmUV...|             -0.0625|
4.0|               3.625|                 0.0|
|              4.0|  4.0|Double Play Sport...|Las Vegas|   NV|--q7kSBRb0vWC8lSk...|                 0.0|
4.0|                 4.0|                 0.0|
| 4.976744186046512|  5.0|   Kidz Cuts By Lori|Henderson|   NV|-0ZO00Vm2ADchytlE...|-0.00465116279069...|
5.0|  4.9743589743589745|                 0.0|
|3.8107142857142855|  4.0|Río Mirage Café y...|El Mirage|   AZ|-1VaIJza42Hjev6uk...|-0.04732142857142...| 3.7931034482758
62|   3.812749003984064|-0.05172413793103...|
+-----------------+-----+--------------------+---------+-----+--------------------+--------------------+---------------
--+--------------------+--------------------+
only showing top 5 rows
```

In [41]:

```python
business_review_skew_elite = business_review_skew_elite.withColumn('skew_non_elite',
                                                                   (F.col('non_elite_avg(stars)') - F.col('stars')) / F.c

business_review_skew_elite.show(5)
```

```
+------------------+-----+--------------------+---------+-----+--------------------+--------------------+----------------
--+--------------------+--------------------+--------------------+
|        avg(stars)|stars|                name|     city|state|         business_id|                skew|  elite_avg(star
s)|non_elite_avg(stars)|          skew_elite|      skew_non_elite|
+------------------+-----+--------------------+---------+-----+--------------------+--------------------+----------------
--+--------------------+--------------------+--------------------+
|  4.11784140969163|  4.0|Delmonico Steakhouse|Las Vegas|   NV|--9e1ONYQuAa-CB_R...|0.029460352422907565|4.19160583941605
85|    4.08596214511041|0.047901459854014616|0.021490536277602557|
|              3.75|  4.0|Maricopa County D...|  Phoenix|   AZ|--phjqoPSPa8sLmUV...|             -0.0625|
4.0|               3.625|                 0.0|            -0.09375|
|               4.0|  4.0|Double Play Sport...|Las Vegas|   NV|--q7kSBRb0vWC8lSk...|                 0.0|
4.0|                 4.0|                 0.0|                 0.0|
| 4.976744186046512|  5.0|   Kidz Cuts By Lori|Henderson|   NV|-0ZO00Vm2ADchytlE...|-0.00465116279069...|
5.0|  4.9743589743589745|                 0.0|-0.00512820512820511|
|3.8107142857142855|  4.0|Río Mirage Café y...|El Mirage|   AZ|-1VaIJza42Hjev6uk...|-0.04732142857142...| 3.7931034482758
62|   3.812749003984064|-0.05172413793103...|-0.04681274900398402|
+------------------+-----+--------------------+---------+-----+--------------------+--------------------+----------------
--+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Calculating the difference between skew_elite and skew_non_elite to compare how much the opinions differ between the two groups, by [skew_elite] - [skew_non_elite]. If the result is positive, then it means elites are more satisifed than regular users do for a restaurant, otherwise if negative, that means elites are less satisfied than regular users do.

And the further away the value is from 0 means the elites have a bigger difference in rating/opinion compared to regular users

In [42]:

```
business_review_skew_elite = business_review_skew_elite.withColumn('skew_diff', (F.col('skew_elite') - F.col('skew_non_el
business_review_skew_elite.select('skew_elite', 'skew_non_elite', 'skew_diff').show(5)
```

```
+--------------------+--------------------+--------------------+
|          skew_elite|      skew_non_elite|           skew_diff|
+--------------------+--------------------+--------------------+
|0.047901459854014616|0.021490536277602557| 0.02641092357641206|
|                 0.0|            -0.09375|             0.09375|
|                 0.0|                 0.0|                 0.0|
|                 0.0|-0.00512820512820511| 0.00512820512820511|
|-0.05172413793103...|-0.04681274900398402|-0.00491138892705...|
+--------------------+--------------------+--------------------+
only showing top 5 rows
```

Graph to analyze

In [43]:

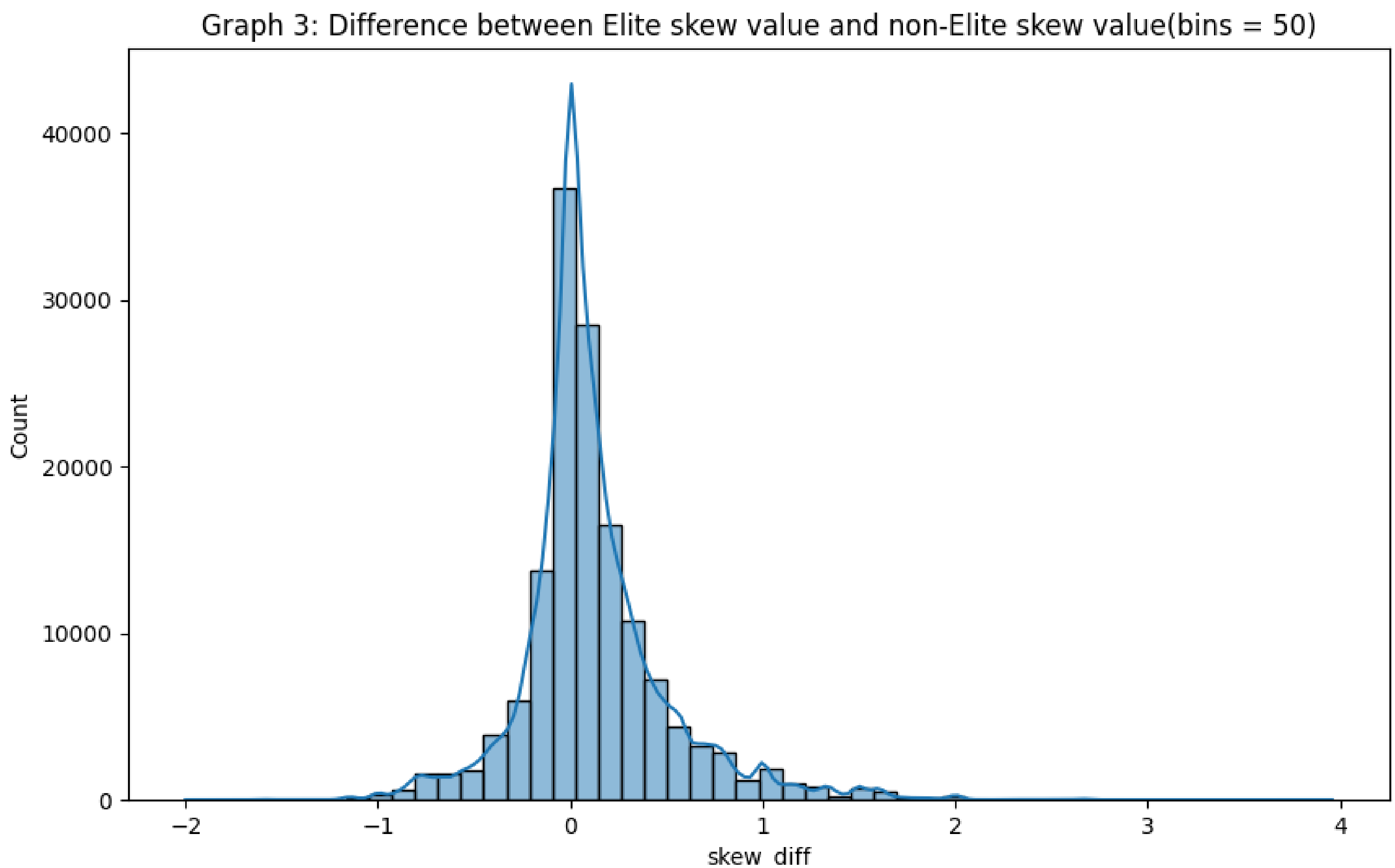
```
business_review_skew_elite_df = business_review_skew_elite.toPandas()
```

In [44]:

```
plt.figure(figsize = (10,6))

sns.histplot(data = business_review_skew_elite_df["skew_diff"],
             bins = 50,
             kde = True).set_title('Graph 3: Difference between Elite skew value and non-Elite skew value(bins = 50)')

plt.show()
%matplot plt
```

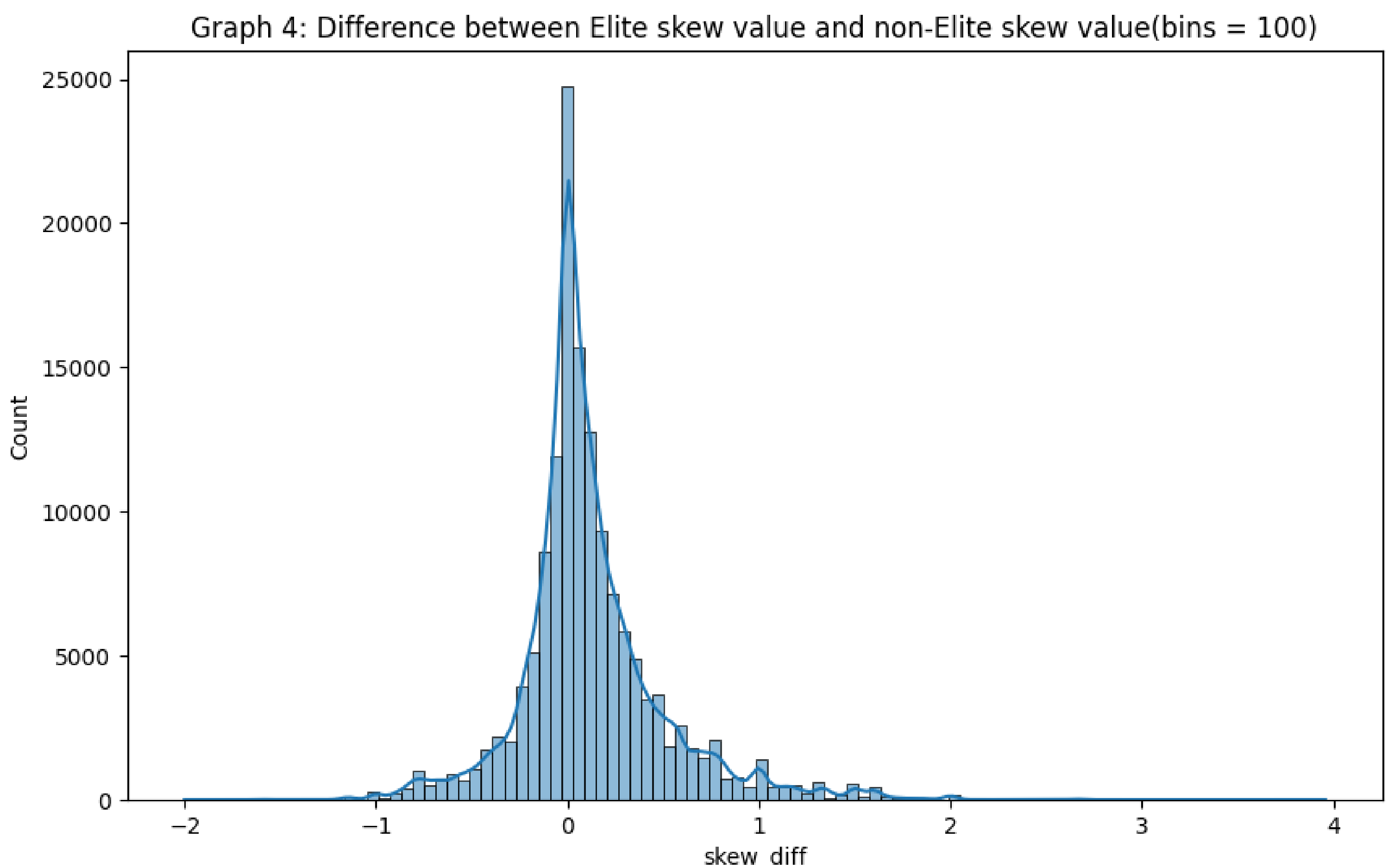


In [45]:

```
plt.figure(figsize = (10,6))

sns.histplot(data = business_review_skew_elite_df["skew_diff"],
             bins = 100,
             kde = True).set_title('Graph 4: Difference between Elite skew value and non-Elite skew value(bins = 100)')

plt.show()
%matplot plt
```



Count the positive, negative, and neutral reviews

In [46]:
```
positive_ = 0
negative_ = 0
neutral_ = 0

for i in business_review_skew_elite_df.index:
    if business_review_skew_elite_df["skew_diff"][i] == 0:
        neutral_ += 1
    if business_review_skew_elite_df["skew_diff"][i] > 0:
        positive_ += 1
    if business_review_skew_elite_df["skew_diff"][i] < 0:
        negative_ += 1

print(f"Postive: {positive_}, Negative: {negative_}, Neutral: {neutral_}")
```

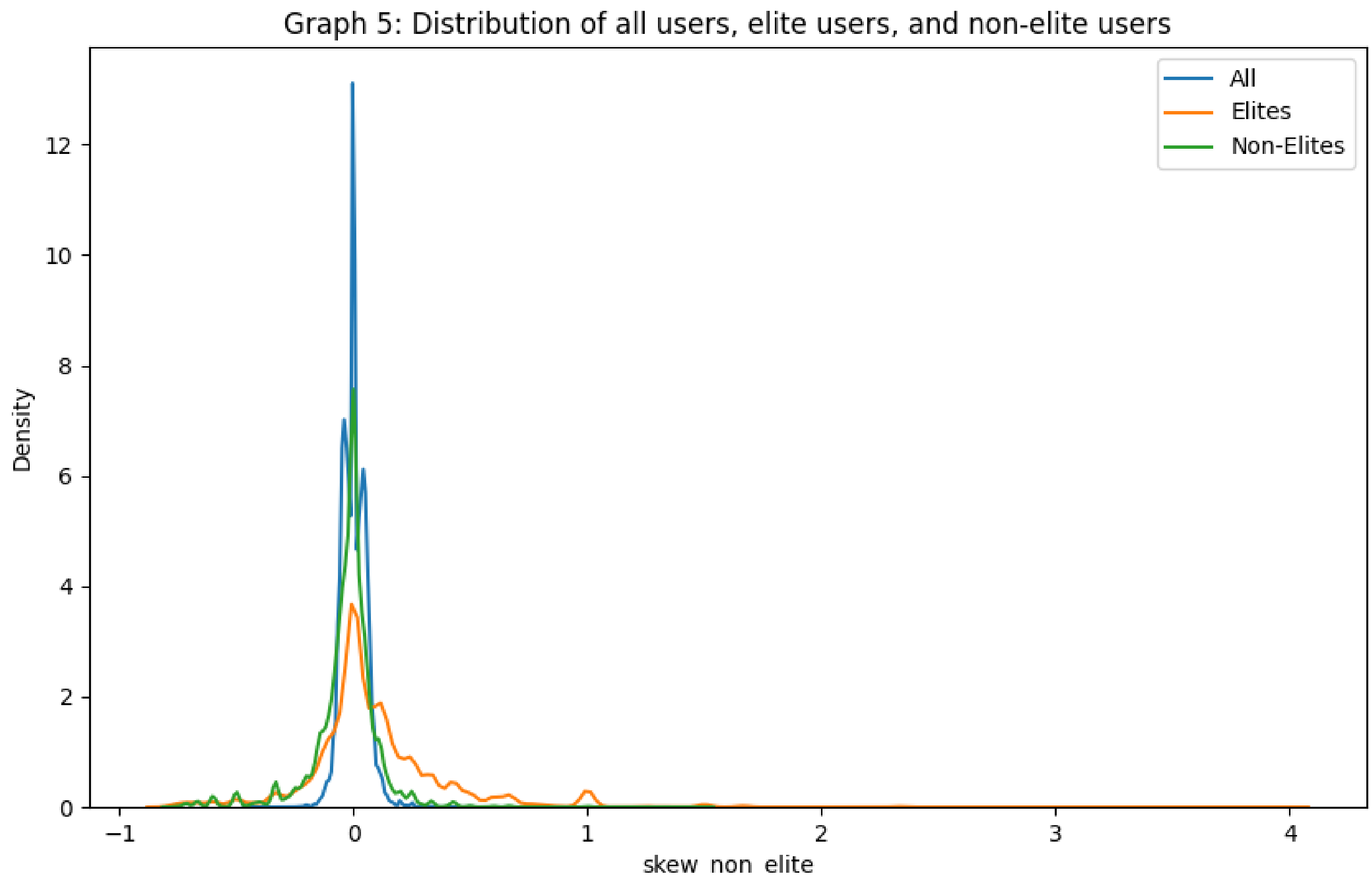
```
Postive: 86339, Negative: 49029, Neutral: 11111
```

Graphing the distribution of all skew value

In [47]:
```
plt.figure(figsize = (10,6))

sns.distplot(business_review_skew_elite_df["skew"], hist = False, label = 'All')
sns.distplot(business_review_skew_elite_df["skew_elite"], hist = False, label = 'Elites')
sns.distplot(business_review_skew_elite_df["skew_non_elite"],
             hist = False,
             label = 'Non-Elites').set_title('Graph 5: Distribution of all users, elite users, and non-elite users')
plt.legend()

plt.show()
%matplot plt
```



What does this mean?

According to the difference graphs (graph 3 and 4), more elite users usually tend to be more satisfied than a normal user would do. In other words, more elite users tend to give out higher written review ratings compared to the star rating of a restaurant becuase the graphs are slightly skewed to the left. However, this does not take into consideration of how high or low the rating is. But generally speaking, elite user's opinion are not too drastically different from non-elite users.

Furthermore, more counts of difference in skewness are positive. I kind of interpreted this elite users hyping up restuarants' ratings. Also in graph 5, the distribution of all skew value, the distribution of elite skew value is more spread out instead of being concentrated like the other two. The distribution of non-elite users and the start ratings are pretty similar. Therefore, elite users tends to be more opinionated than non-elite users.

Elite users ratings are skewed more towards **positive** compared to non-elites. In order words, I probably will not take the words of an elite user for granted especially when they provide a positive rating because it may set my expectation for a restaurant high.