Analysis of Yelp Business Intelligence Data by Sky Song

In this notebook, We will analyze a subset of Yelp's business, reviews and user data. \ This dataset comes from Kaggle \ I took steps to pull this data into s3 bucket: ('s3://sta9760yelpdatasetsky/yelp/')\ Total of 11GB 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]:

```
%%info
```

Current session configs: `{'conf': {'spark.pyspark.python': 'python3', 'spark.pyspark.virtualenv.enabled': 'true', 'spark.pyspark.virtualenv.type': 'native', 'spark.pyspark.virtualenv.bin.path': '/usr/bin/virtualenv'}, 'kind': 'pyspark'}`

No active sessions.

In [2]:

```
sc.install_pypi_package("matplotlib==3.2.1")
sc.install_pypi_package("pandas==1.0.3")
```

```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1638207336197_0001	pyspark	idle	Link	Link	✔

```
SparkSession available as 'spark'.

Collecting matplotlib==3.2.1
  Downloading https://files.pythonhosted.org/packages/b2/c2/71fcf957710f3ba1f09088b35776a799ba7dd95f7c2b195ec800933b276b/
matplotlib-3.2.1-cp37-cp37m-manylinux1_x86_64.whl (12.4MB)
Collecting python-dateutil>=2.1 (from matplotlib==3.2.1)
  Downloading https://files.pythonhosted.org/packages/36/7a/87837f39d0296e723bb9b62bbb257d0355c7f6128853c78955f57342a56d/
python_dateutil-2.8.2-py2.py3-none-any.whl (247kB)
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/a0/34/895006117f6fce0b4de045c87e154ee4a20c68ec0a4c9a36d900888fb6bc/
pyparsing-3.0.6-py3-none-any.whl (97kB)
```

```
Collecting cycler>=0.10 (from matplotlib==3.2.1)
  Downloading https://files.pythonhosted.org/packages/5c/f9/695d6bedebd747e5eb0fe8fad57b72fdf25411273a39791cde838d5a8f51/
cycler-0.11.0-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/09/6b/6e567cb2e86d4e5939a9233f8734e26021b6a9c1bc4b1edccba236a84cc2/
kiwisolver-1.3.2-cp37-cp37m-manylinux_2_5_x86_64.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=
=3.2.1)
Installing collected packages: python-dateutil, pyparsing, cycler, kiwisolver, matplotlib
Successfully installed cycler-0.11.0 kiwisolver-1.3.2 matplotlib-3.2.1 pyparsing-3.0.6 python-dateutil-2.8.2

Collecting pandas==1.0.3
  Downloading 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)
Requirement already satisfied: python-dateutil>=2.6.1 in /mnt/tmp/1638208065004-0/lib/python3.7/site-packages (from panda
s==1.0.3)
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: pandas
Successfully installed pandas-1.0.3
```

In [3]:

```
# "you can run the latest version of seaborn but before that you only need to run another package which is scipy."
```

In [4]:

```
sc.install_pypi_package("scipy==1.7.0")
```

```
Collecting scipy==1.7.0
  Downloading https://files.pythonhosted.org/packages/b2/85/b00f13b52d079b5625e1a12330fc6453c947a482ff667a907c7bc60ed220/
scipy-1.7.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (28.5MB)
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in /usr/local/lib64/python3.7/site-packages (from scipy==1.7.0)
Installing collected packages: scipy
Successfully installed scipy-1.7.0
```

In [5]:

```
sc.install_pypi_package("seaborn==0.11.2")
```

```
Collecting seaborn==0.11.2
```

```
  Downloading https://files.pythonhosted.org/packages/10/5b/0479d7d845b5ba410ca702ffcd7f2cd95a14a4dfff1fde2637802b258b9b/
seaborn-0.11.2-py3-none-any.whl (292kB)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib64/python3.7/site-packages (from seaborn==0.11.2)
Requirement already satisfied: scipy>=1.0 in /mnt/tmp/1638208065004-0/lib/python3.7/site-packages (from seaborn==0.11.2)
Requirement already satisfied: matplotlib>=2.2 in /mnt/tmp/1638208065004-0/lib/python3.7/site-packages (from seaborn==0.1
1.2)
Requirement already satisfied: pandas>=0.23 in /mnt/tmp/1638208065004-0/lib/python3.7/site-packages (from seaborn==0.11.
2)
Requirement already satisfied: python-dateutil>=2.1 in /mnt/tmp/1638208065004-0/lib/python3.7/site-packages (from matplot
lib>=2.2->seaborn==0.11.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /mnt/tmp/1638208065004-0/lib/python3.7/site-pa
ckages (from matplotlib>=2.2->seaborn==0.11.2)
Requirement already satisfied: cycler>=0.10 in /mnt/tmp/1638208065004-0/lib/python3.7/site-packages (from matplotlib>=2.2
->seaborn==0.11.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /mnt/tmp/1638208065004-0/lib/python3.7/site-packages (from matplotlib
>=2.2->seaborn==0.11.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (from pandas>=0.23->seaborn==0.11.
2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.1->matplotlib>
=2.2->seaborn==0.11.2)
Installing collected packages: seaborn
Successfully installed seaborn-0.11.2
```

Importing

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

In [90]:

```
import matplotlib
import pandas
import scipy
import seaborn
import numpy as np
```

Loading Data

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

In [68]:

```
df1 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_business.json')
```

Overview of Data

Display the number of rows and columns in our dataset.

In [69]:

```
print(f'Columns: {len(df1.dtypes)} | Rows: {df1.count():,}')
```

```
Columns: 14 | Rows: 160,585
```

In [70]:

```
df1.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`

To really see no null value in name, city, state and categories:

In [71]:

```
df1.createOrReplaceTempView("business")
output = spark.sql('select business_id,name,city,state,stars,categories from business WHERE business_id != "null" and nam
output.show()
```

```
+--------------------+--------------------+-----------+-----+-----+--------------------+
|         business_id|                name|       city|state|stars|          categories|
+--------------------+--------------------+-----------+-----+-----+--------------------+
|6iYb2HFDywm3zjuRg...| Oskar Blues Taproom|    Boulder|   CO|  4.0|Gastropubs, Food,...|
|tCbdrRPZA0oiIYSmH...|Flying Elephants ...|   Portland|   OR|  4.0|Salad, Soup, Sand...|
|bvN78flM8NLprQ1a1...|      The Reclaimory|   Portland|   OR|  4.5|Antiques, Fashion...|
|oaepsyvc0J17qwi8c...|         Great Clips|Orange City|   FL|  3.0|Beauty & Spas, Ha...|
|PE9uqAjdw0E4-8mjG...|   Crossfit Terminus|    Atlanta|   GA|  4.0|Gyms, Active Life...|
+--------------------+--------------------+-----------+-----+-----+--------------------+
```

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

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.

In [72]:
```
associationTable = spark.sql("select business_id,explode(split(categories,', ')) as category from business")
associationTable.createOrReplaceTempView("categories")
output = spark.sql("select * from categories limit 5")
output.show()
```

```
+--------------------+------------+
|         business_id|    category|
+--------------------+------------+
|6iYb2HFDywm3zjuRg...|  Gastropubs|
|6iYb2HFDywm3zjuRg...|        Food|
|6iYb2HFDywm3zjuRg...|Beer Gardens|
|6iYb2HFDywm3zjuRg...| Restaurants|
|6iYb2HFDywm3zjuRg...|        Bars|
+--------------------+------------+
```

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 [73]:
```
total = spark.sql("select distinct category from categories")
print(total.count())
```

```
1330
```

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 [74]:

```
output = spark.sql("select category, count(*) as count from categories group by category")
output.show()
```

```
+--------------------+-----+
|            category|count|
+--------------------+-----+
|       Dermatologists|  351|
|       Paddleboarding|   67|
|         Aerial Tours|    8|
|          Hobby Shops|  610|
|           Bubble Tea|  779|
|              Embassy|    9|
|              Tanning|  701|
|             Handyman|  507|
|       Aerial Fitness|   13|
|              Falafel|  141|
|         Summer Camps|  308|
|        Outlet Stores|  184|
|      Clothing Rental|   37|
|       Sporting Goods| 1864|
|      Cooking Schools|  114|
```

```
|  College Counseling|   20|
|  Lactation Services|   47|
|Ski & Snowboard S...|   55|
|             Museums|  336|
|              Doulas|   52|
+--------------------+-----+
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!

```
%matplot plt
```

In [75]:

```
from matplotlib import pyplot as plt
```

In [76]:

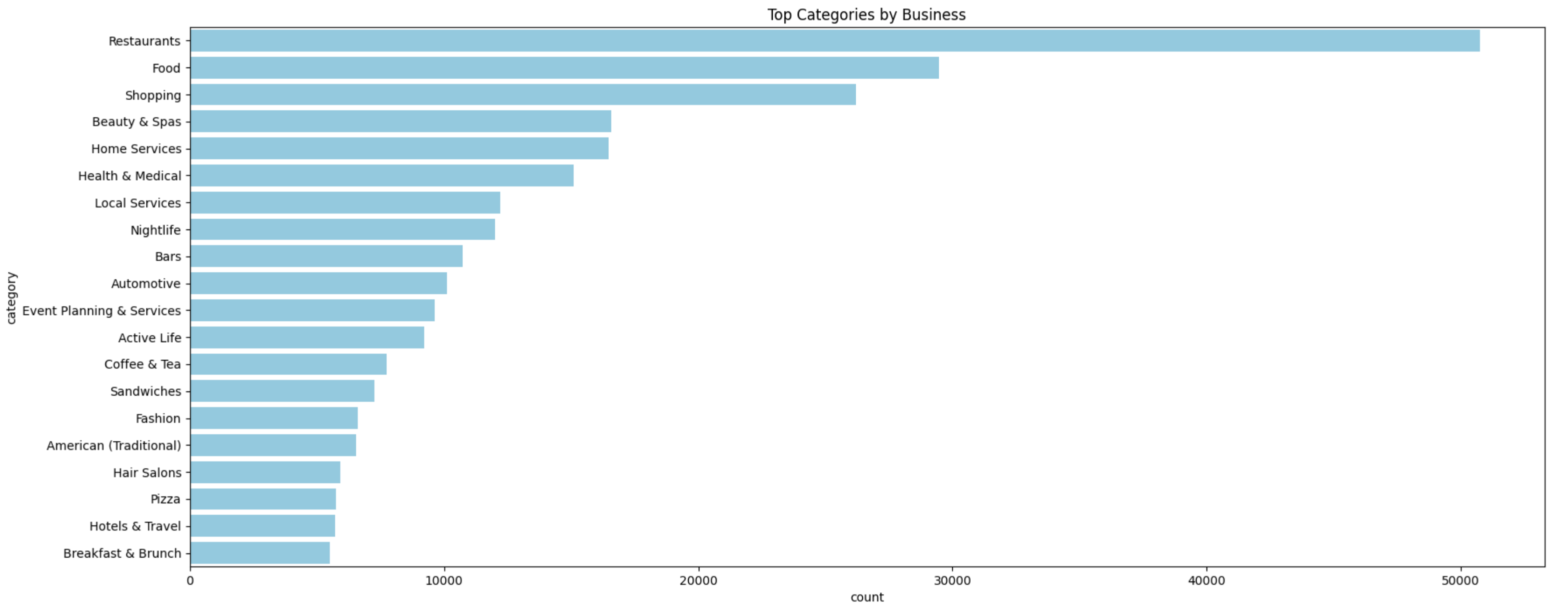
```
top_df = spark.sql("select category, count(*) as count from categories group by category order by count(*) desc limit 20"
```

In [77]:

```
top_df2 = top_df.toPandas()
```

In [78]:

```
fig, ax = plt.subplots(figsize = (20,8))
top_cate_plot = seaborn.barplot(x = 'count', y = 'category', data = top_df2, ax = ax, color = 'skyblue')
ax.set_title('Top Categories by Business')
%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 User Data

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

In [19]:

```
# User data not quie like same as professor's notebook
# It is the review one
```

In [20]:

```
# Choose review and look like same as professor guidline
```

```
df2 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_review.json')
df2.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)
```

In [22]:

```
df2.createOrReplaceTempView("review")
output = spark.sql('select business_id, stars from review limit 5')
output.show()
```

```
+--------------------+-----+
|         business_id|stars|
+--------------------+-----+
|DiRIdYhGyuTNZurKy...|  3.0|
|bPmWDBkjBhV11Yk4B...|  5.0|
|xHdKDNcJrvkYJkAGs...|  4.0|
|Irp5sgl7XASH5ZTw2...|  5.0|
|R8fLQ6TLz06MQR69K...|  3.0|
+--------------------+-----+
```

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 [23]:

```
output = spark.sql('select business_id, avg(stars) as avgStars from review group by business_id')
output.createOrReplaceTempView("averageReview")
output1 = spark.sql('select * from averageReview limit 5')
output1.show()
```

```
+--------------------+-----------------+
|         business_id|         avgStars|
+--------------------+-----------------+
```

```
|yWG3JLNsqEkU1Y8wj...|3.423076923076923|
|4jQ1y1_ItTCj3C9Xl...|             3.28|
|ZmRWz7YKDbc_ONBS1...|              4.0|
|DT-WVQB-R_iiShvCo...|              1.8|
|-2ysHxktRcDom1m9A...|              5.0|
+--------------------+-----------------+
```

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

In [24]:

```
output = spark.sql('''select rev.*, bus.stars, bus.name, bus.city, bus.state
                        from business as bus
                        left outer join averageReview as rev
                        on bus.business_id = rev.business_id''')
output.createOrReplaceTempView("joinedOutput")
```

In [25]:

```
output = spark.sql('select avgStars, stars, name, city, state from joinedOutput WHERE name != "null" and city != "null" a
output.show()
```

```
+------------------+-----+--------------------+----------+-----+
|          avgStars|stars|                name|      city|state|
+------------------+-----+--------------------+----------+-----+
|               5.0|  5.0|    CheraBella Salon|   Peabody|   MA|
|             3.875|  4.0|Mezcal Cantina & ...|  Columbus|   OH|
|3.8666666666666667|  4.0|    Red Table Coffee|    Austin|   TX|
|               5.0|  5.0|          WonderWell|    Austin|   TX|
|             3.375|  3.5|         Avalon Oaks|Wilmington|   MA|
+------------------+-----+--------------------+----------+-----+
```

Let's see a few of these:

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']
```

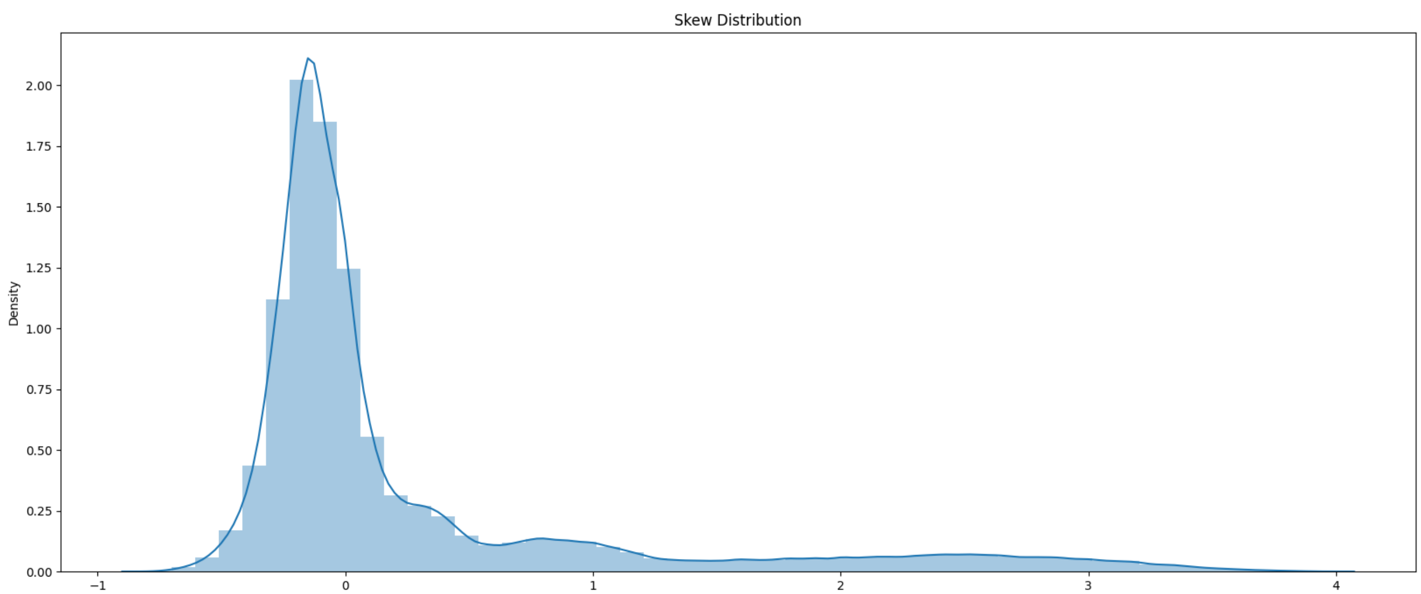
If the **skew** is negative, we can interpret that to be: reviewers who left a written response were more dissatisfied than normal. If **skew** is positive, we can interpret that to be: reviewers who left a written response were more satisfied than normal.

In [26]:
```
skew_df = spark.sql("select (avgStars-stars)/stars from joinedOutput")
```

And finally, graph it!

In [27]:
```
skew_df_2 = skew_df.toPandas()

fig, ax = plt.subplots(figsize = (20,8))
skew_plot = seaborn.distplot(skew_df_2)
ax.set_title('Skew Distribution')
%matplot plt
```



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

No, the skew visualization shows a ***positive*** skew which tails are in positive direction. \ The mean of positively skewed data will be greater than the median. \ It can be understanded as reviewers who left a written response were more satisfied than normal.

Should the Elite be Trusted?

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

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

In [145…

```
df1 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_business.json')
df2 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_review.json')
```

In [146…

```
df3 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_user.json')
df3.printSchema()
```

```
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)
```

In [54]:

```
# User Look
# df3.createOrReplaceTempView("user")
# output = spark.sql('select user_id, name, elite, average_stars, cool,fans, useful,review_count from user where elite !=
# output.show()
```

```
+--------------------+---------+--------------------+-------------+-----+----+------+------------+
|             user_id|     name|               elite|average_stars| cool|fans|useful|review_count|
+--------------------+---------+--------------------+-------------+-----+----+------+------------+
|q_QQ5kBBwlCcbL1s4...|     Jane|2006,2007,2008,20...|         3.85|11291|1357| 15038|        1220|
|dIIKEfOgo0KqUfGQv...|     Gabi|2007,2008,2009,20...|         4.09|18046|1025| 21272|        2136|
|D6ErcUnFALnCQN4b1...|    Jason|           2010,2011|         3.76|  130|  16|   188|         119|
|JnPIjvC0cmooNDfsa...|      Kat|2009,2010,2011,20...|         3.77| 4035| 420|  7234|         987|
|37Hc8hr3cw0iHLoPz...|Christine|      2009,2010,2011|         3.72| 1124|  47|  1577|         495|
+--------------------+---------+--------------------+-------------+-----+----+------+------------+
```

In [ ]:

```
# To join the review star to elite
```

In [147…

```
df_business = df1.withColumnRenamed('stars',"business_stars")
df_review = df2.withColumnRenamed('stars',"review_stars")
df_business_join_review = df_business.join(df_review, on=['business_id'], how='inner')
df_business_user_review = df3.join(df_business_join_review, on=['user_id'], how='inner')
```

In [148…

```
df_business_user_review.select('business_id','business_stars','review_stars','user_id','elite').sort('business_id','user_
```

```
+--------------------+--------------+------------+--------------------+--------------------+
|         business_id|business_stars|review_stars|             user_id|               elite|
```

```
+--------------------+--------------+-----------+--------------------+--------------------+
|--0zrn43LEaB4jUWT...|           1.0|        1.0|Du8CplP209Es9T3FY...|                2008|
|--164t1nclzzmca7e...|           4.0|        3.0|1P9BpFZ_d3PGCdytD...|      2010,2011,2012|
|--164t1nclzzmca7e...|           4.0|        5.0|3d4fac-e3Plyib8QU...|2017,2018,2019,20,20|
|--164t1nclzzmca7e...|           4.0|        4.0|4ZfHbIbmyTuCX0BXN...| 2012,2013,2014,2015|
|--164t1nclzzmca7e...|           4.0|        5.0|5GHfNK-pcCYJon1cS...|                2010|
|--164t1nclzzmca7e...|           4.0|        5.0|5GHfNK-pcCYJon1cS...|                2010|
|--164t1nclzzmca7e...|           4.0|        1.0|8P8dgzKDQg7OSlEiA...|     2018,2019,20,20|
|--164t1nclzzmca7e...|           4.0|        4.0|8X1B-J73QOFV91Y0e...|2009,2010,2011,20...|
|--164t1nclzzmca7e...|           4.0|        2.0|A9-iDWYBSM4MtolTz...| 2014,2015,2016,2017|
|--164t1nclzzmca7e...|           4.0|        3.0|BdLon9gg9reglwmdD...|2010,2011,2012,20...|
+--------------------+--------------+-----------+--------------------+--------------------+
only showing top 10 rows
```

In [149…

```python
df_business_user_review = df_business_user_review.withColumn('difference',(((df_business_user_review['review_stars']-df_b
df_business_user_review = df_business_user_review.select('business_id','business_stars','review_stars','difference','user
df_business_user_review.show(10)
```

```
+--------------------+--------------+-----------+----------+--------------------+--------------------+
|         business_id|business_stars|review_stars|difference|             user_id|               elite|
+--------------------+--------------+-----------+----------+--------------------+--------------------+
|--0zrn43LEaB4jUWT...|           1.0|        1.0|      -0.0|Du8CplP209Es9T3FY...|                2008|
|--164t1nclzzmca7e...|           4.0|        5.0|     -25.0|llksdcDyLTNkiibAQ...|2009,2010,2011,20...|
|--164t1nclzzmca7e...|           4.0|        1.0|      75.0|Jgxz4UF56FK0taE4i...|           2012,2013|
|--164t1nclzzmca7e...|           4.0|        5.0|     -25.0|WJDYWvNrnMx2PWgfK...|                2012|
|--164t1nclzzmca7e...|           4.0|        4.0|      -0.0|DA90NhtNTNpXxdrXI...|2010,2011,2012,20...|
|--164t1nclzzmca7e...|           4.0|        2.0|      50.0|A9-iDWYBSM4MtolTz...| 2014,2015,2016,2017|
|--164t1nclzzmca7e...|           4.0|        3.0|      25.0|LhnoqfSZobV3bch7o...|2010,2011,2012,20...|
|--164t1nclzzmca7e...|           4.0|        5.0|     -25.0|kTY5w80WqY4Ak-jac...|           2012,2013|
|--164t1nclzzmca7e...|           4.0|        4.0|      -0.0|4ZfHbIbmyTuCX0BXN...| 2012,2013,2014,2015|
|--164t1nclzzmca7e...|           4.0|        3.0|      25.0|1P9BpFZ_d3PGCdytD...|      2010,2011,2012|
+--------------------+--------------+-----------+----------+--------------------+--------------------+
only showing top 10 rows
```

In [150…

```python
df_business_user_review.describe(['difference']).show()
```

```
+-------+-----------------+
|summary|       difference|
+-------+-----------------+
|  count|          2169088|
|   mean|-4.350569024167467|
```

```
| stddev| 30.29839006240463|
|    min|            -400.0|
|    max|              80.0|
+-------+------------------+
```

In [151…

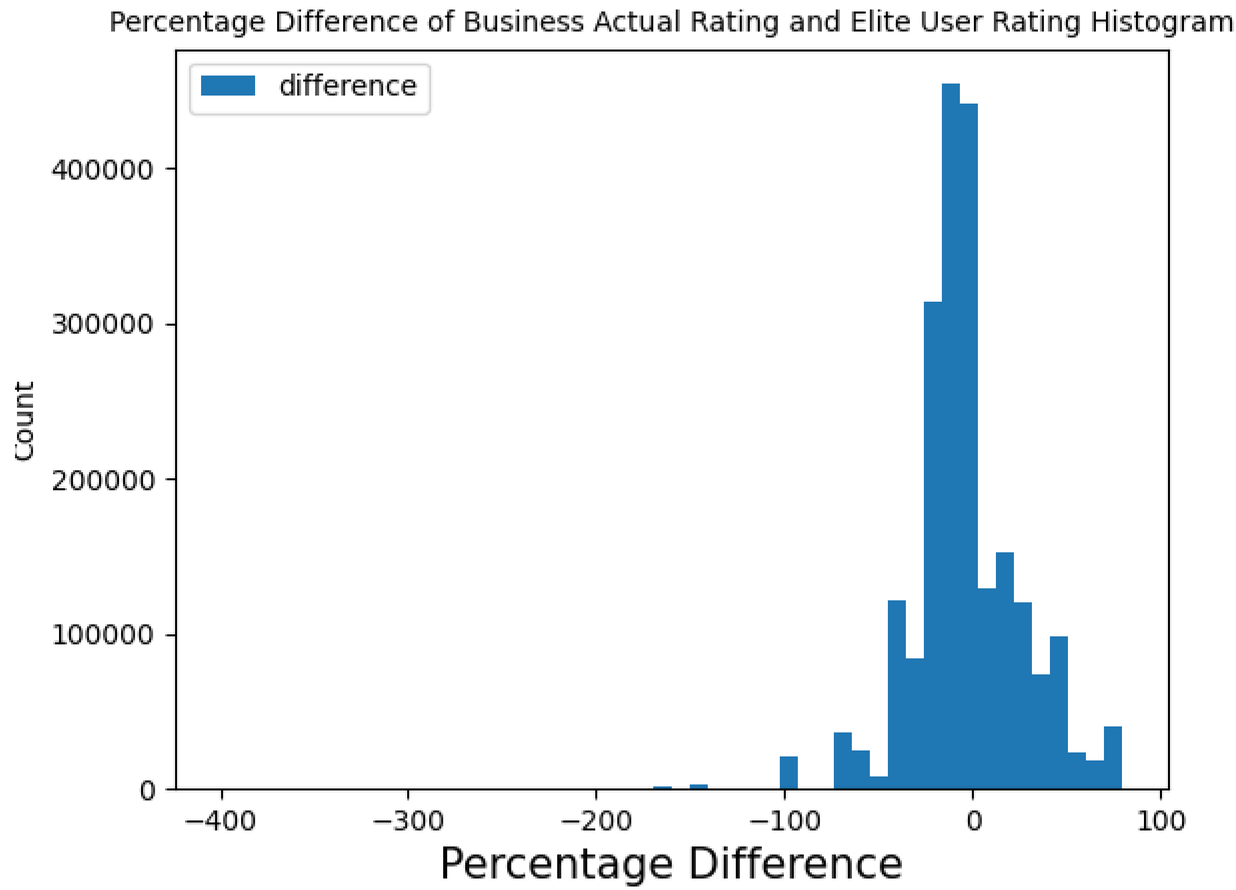
```
df_business_user_review_plot = df_business_user_review.select('difference').toPandas()
df_business_user_review_plot.plot.hist(bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot object at 0x7efc9d721a50>
```

In [152…

```
plt.xlabel('Percentage Difference',fontsize=15)
plt.ylabel('Count',fontsize=10)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title('Percentage Difference of Business Actual Rating and Elite User Rating Histogram',fontsize=10)

%matplot plt
```



In [ ]:

```
# Compare to nonelite....
```

In [122…

```
df1 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_business.json')
df2 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_review.json')
df3 = spark.read.json('s3://sta9760yelpdatasetsky/yelp/yelp_academic_dataset_user.json')
```

In [134…

```
df_business = df1.withColumnRenamed('stars',"business_stars")
df_review = df2.withColumnRenamed('stars',"review_stars")
```

```
df_business_join_review = df_business.join(df_review, on=['business_id'], how='inner')
df_business_user_review = df3.join(df_business_join_review, on=['user_id'], how='inner')
```

In [136…

```
df_business_user_review.select('business_id','business_stars','review_stars','user_id','elite').sort('business_id','user_
```

```
+--------------------+--------------+------------+--------------------+-----+
|         business_id|business_stars|review_stars|             user_id|elite|
+--------------------+--------------+------------+--------------------+-----+
|--0DF12EMHYI8XIgo...|           4.5|         5.0|4NXvQ0bjbcpU0LW1n...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|YFAFZsD_-x8O_7tdl...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|bj5IdqQ8M09xJo1B3...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|dF09N5TYcKOwTlhxz...|     |
|--0DF12EMHYI8XIgo...|           4.5|         1.0|hliUuVm4vGVdODfRQ...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|vYza0QbNkPUAd0ydK...|     |
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|0OsSL00kvQvMoOxf9...|     |
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|1ixRBXu2YUaFoKXFT...|     |
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|HzbScQ9XNEUrr8B_c...|     |
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|RB_J91Ur9DlbbHZlV...|     |
+--------------------+--------------+------------+--------------------+-----+
only showing top 10 rows
```

In [137…

```
# Not Elite info
df_business_user_review = df_business_user_review.withColumn('difference',(((df_business_user_review['review_stars']-df_b
df_business_user_review = df_business_user_review.select('business_id','business_stars','review_stars','difference','user
df_business_user_review.show(10)
```

```
+--------------------+--------------+------------+------------------+--------------------+-----+
|         business_id|business_stars|review_stars|        difference|             user_id|elite|
+--------------------+--------------+------------+------------------+--------------------+-----+
|--0DF12EMHYI8XIgo...|           4.5|         5.0|-11.11111111111111|4NXvQ0bjbcpU0LW1n...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|-11.11111111111111|bj5IdqQ8M09xJo1B3...|     |
|--0DF12EMHYI8XIgo...|           4.5|         1.0| 77.77777777777779|hliUuVm4vGVdODfRQ...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|-11.11111111111111|vYza0QbNkPUAd0ydK...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|-11.11111111111111|dF09N5TYcKOwTlhxz...|     |
|--0DF12EMHYI8XIgo...|           4.5|         5.0|-11.11111111111111|YFAFZsD_-x8O_7tdl...|     |
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|              -0.0|1ixRBXu2YUaFoKXFT...|     |
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|              -0.0|RB_J91Ur9DlbbHZlV...|     |
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|              -0.0|HzbScQ9XNEUrr8B_c...|     |
```

```
|--0r8K_AQ4FZfLsX3...|           5.0|         5.0|               -0.0|0OsSL00kvQvMoOxf9...|     |
+--------------------+-------------+------------+------------------+--------------------+-----+
only showing top 10 rows
```

In [143…

```
#Not Elite summary
df_business_user_review.describe(['difference']).show()
```

```
+-------+--------------------+
|summary|          difference|
+-------+--------------------+
|  count|             8635403|
|   mean|-0.08967093748134816|
| stddev|   38.874466446914425|
|    min|              -400.0|
|    max|                80.0|
+-------+--------------------+
```

In [139…

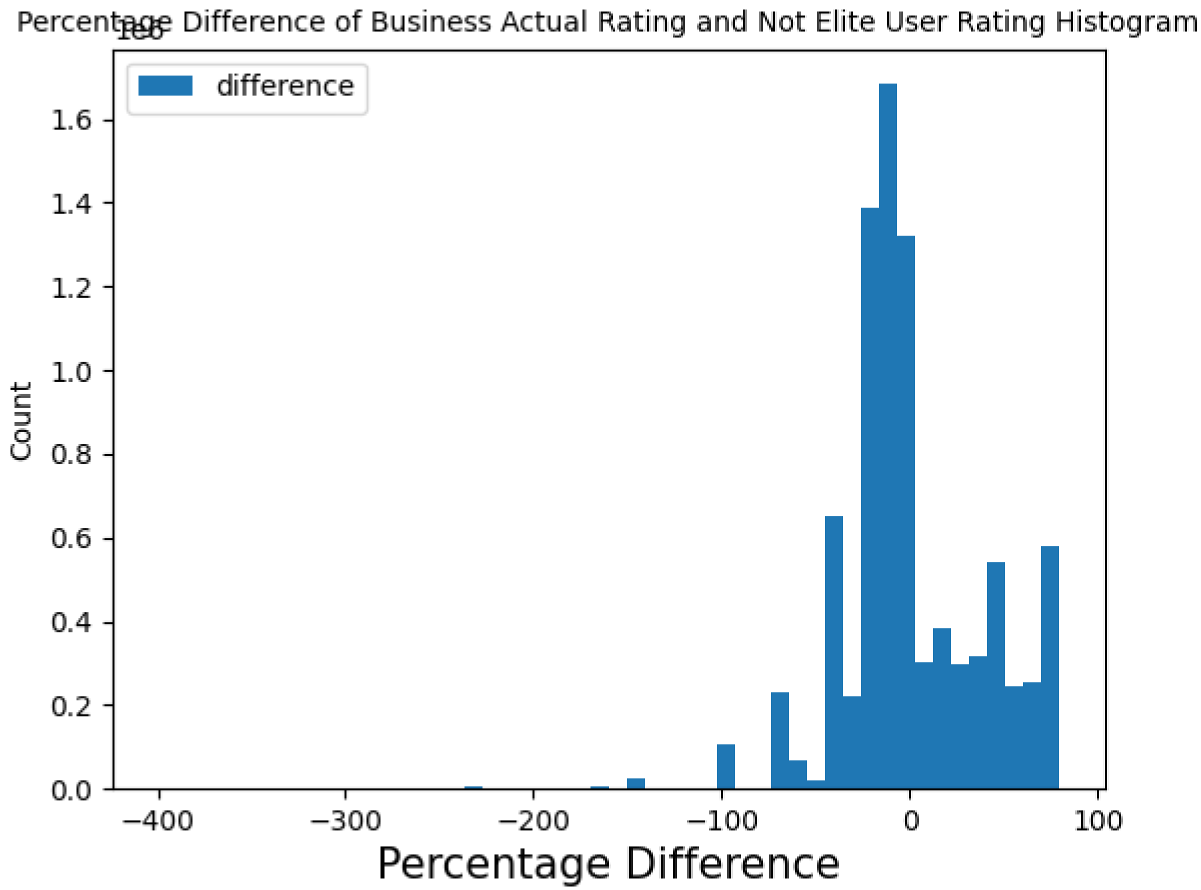
```
df_business_user_review_plot = df_business_user_review.select('difference').toPandas()
df_business_user_review_plot.plot.hist(bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot object at 0x7efc16f83490>
```

In [142…

```
plt.xlabel('Percentage Difference',fontsize=15)
plt.ylabel('Count',fontsize=10)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.title('Percentage Difference of Business Actual Rating and Not Elite User Rating Histogram',fontsize=10)

%matplot plt
```



Conclusion

By the summary and graphs, they show that the elite provides stars that have less standard deviation than non elite. Although there are 4 times more non elite data than elite, the standard deviation of percentage difference for non elite is still higher than elite people.

Based on their mean, they are pretty much even on giving extreme positive or extreme negative stars. The elite tends to leave a slight negative impact like -4.35% than actual business_star.

Based on graph, the non-elite people tends to give more negative review_stars than actual business_stars.

All in all, we shall always trust the elite! Or trust elite more than non elite.

In [ ]: