Analysis of Yelp Business Intelligence Data

We will analyze a subset of Yelp's business, reviews and user data. This dataset comes to us from Kaggle although we have taken steps to pull this data into a publis s3 bucket: `s3://sta9760-yelpdataset/yelp-light/*business.json`

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.install_pypi_package("matplotlib==3.2.1")
sc.install_pypi_package("pandas==1.0.3")
sc.install_pypi_package("seaborn==0.11.0")
sc.install_pypi_package("scipy==1.5.4")
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
5	application_1606229628628_0006	pyspark	idle	Link	Link	✔

```
SparkSession available as 'spark'.

Collecting matplotlib==3.2.1
  Using cached https://files.pythonhosted.org/packages/b2/c2/71fcf957710f3ba1f09088b3577
6a799ba7dd95f7c2b195ec800933b276b/matplotlib-3.2.1-cp37-cp37m-manylinux1_x86_64.whl
Collecting python-dateutil>=2.1 (from matplotlib==3.2.1)
  Using cached https://files.pythonhosted.org/packages/d4/70/d60450c3dd48ef87586924207ae
8907090de0b306af2bce5d134d78615cb/python_dateutil-2.8.1-py2.py3-none-any.whl
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib==3.2.1)
  Using cached https://files.pythonhosted.org/packages/8a/bb/488841f56197b13700afd5658fc
279a2025a39e22449b7cf29864669b15d/pyparsing-2.4.7-py2.py3-none-any.whl
Collecting cycler>=0.10 (from matplotlib==3.2.1)
  Using cached https://files.pythonhosted.org/packages/f7/d2/e07d3ebb2bd7af696440ce7e754
c59dd546ffe1bbe732c8ab68b9c834e61/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)
  Using cached https://files.pythonhosted.org/packages/d2/46/231de802ade4225b76b96cffe41
9cf3ce52bbe92e3b092cf12db7d11c207/kiwisolver-1.3.1-cp37-cp37m-manylinux1_x86_64.whl
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, matplotli
b
Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.2.1 pyparsing-2.4.7 p
ython-dateutil-2.8.1

Collecting pandas==1.0.3
  Using cached https://files.pythonhosted.org/packages/4a/6a/94b219b8ea0f2d580169e85ed1e
dc0163743f55aaeca8a44c2e8fc1e344e/pandas-1.0.3-cp37-cp37m-manylinux1_x86_64.whl
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (f
rom 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/1606243180516-0/lib/py
thon3.7/site-packages (from pandas==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

Collecting seaborn==0.11.0
  Using cached https://files.pythonhosted.org/packages/bc/45/5118a05b0d61173e6eb12bc5804
f0fbb6f196adb0a20e0b16efc2b8e98be/seaborn-0.11.0-py3-none-any.whl
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)
  Using cached https://files.pythonhosted.org/packages/dc/7e/8f6a79b102ca1ea928bae8998b0
5bf5dc24a90571db13cd119f275ba6252/scipy-1.5.4-cp37-cp37m-manylinux1_x86_64.whl
Requirement already satisfied: matplotlib>=2.2 in /mnt/tmp/1606243180516-0/lib/python3.
7/site-packages (from seaborn==0.11.0)
Requirement already satisfied: pandas>=0.23 in /mnt/tmp/1606243180516-0/lib/python3.7/si
te-packages (from seaborn==0.11.0)
Requirement already satisfied: python-dateutil>=2.1 in /mnt/tmp/1606243180516-0/lib/pyth
on3.7/site-packages (from matplotlib>=2.2->seaborn==0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /mnt/tmp/1606
243180516-0/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn==0.11.0)
Requirement already satisfied: cycler>=0.10 in /mnt/tmp/1606243180516-0/lib/python3.7/si
te-packages (from matplotlib>=2.2->seaborn==0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /mnt/tmp/1606243180516-0/lib/python
3.7/site-packages (from matplotlib>=2.2->seaborn==0.11.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (f
rom 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: scipy==1.5.4 in /mnt/tmp/1606243180516-0/lib/python3.7/si
te-packages
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib64/python3.7/site-packages
(from scipy==1.5.4)
```

Importing

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

In [50]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

Loading Data

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

In [3]:

```
business = spark.read.json('s3://sta9760-project2-yelpdataset/yelp_academic_dataset_bus
```

Overview of Data

Display the number of rows and columns in our dataset.

In [4]:
```
num_rows = business.count()
num_cols = len(business.columns)
print('Columns:', num_cols, "| Rows:", num_rows)
```

```
Columns: 14 | Rows: 209393
```

Display the DataFrame schema below.

In [5]:
```
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 [6]:
```
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.

In [7]:
```
from pyspark.sql.functions import explode, split
```

In [8]:
```
business_categories = business.select("business_id", "categories")
business_categories_exploded = business_categories.withColumn('categories', explode(spl
```

Display the first 5 rows of your association table below.

In [9]:
```
business_categories_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 [10]:
```
business_categories_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 [11]:
```
business_categories_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!

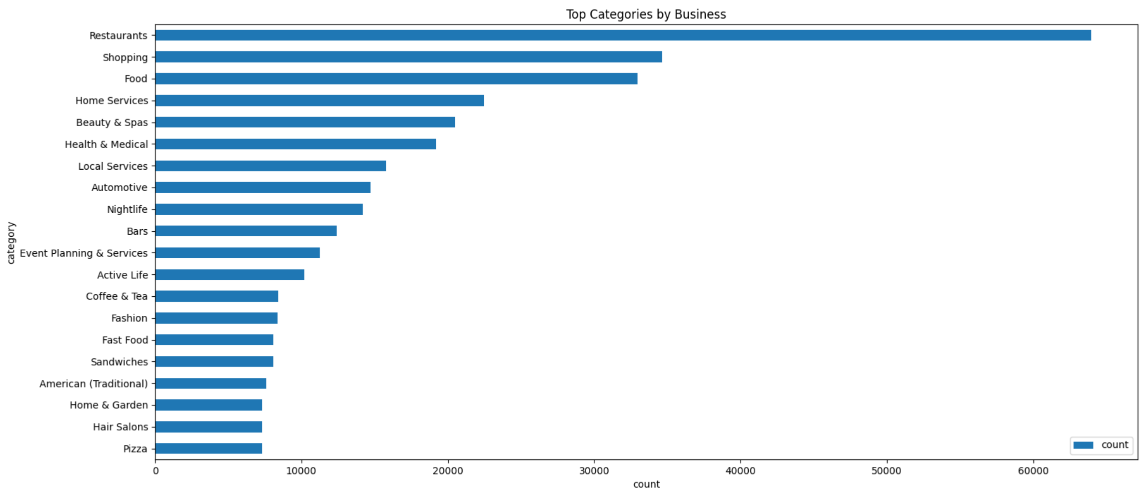
```
%matplot plt
```

In [12]:
```
barchart_business = business_categories_exploded.groupby('categories').count().orderBy(
pdf = barchart_business.limit(20).toPandas()
```

In [13]:
```
pdf = pdf.sort_values('count', ascending=True)
pdf.plot(kind='barh', x='categories', y='count', figsize=(18, 8))
```

```
plt.title('Top Categories by Business')
plt.xlabel('count')
plt.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 User Data

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

In [14]:
```
review = spark.read.json('s3://sta9760-project2-yelpdataset/yelp_academic_dataset_revie
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 [15]:
```
review.select('business_id', 'stars').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 [16]:
```python
from pyspark.sql.functions import length
from pyspark.sql.functions import col

written_reviews = review.select('business_id', 'user_id', 'stars', 'text').where(length
avg_stars = written_reviews.groupby("business_id").avg("stars")
avg_stars.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 [17]:
```python
business = business.join(avg_stars, 'business_id', 'inner')
```

Let's see a few of these:

In [18]:
```python
business.select('avg(stars)', 'stars', 'name', 'city', 'state').show(5)
```

```
+----------------+-----+--------------------+---------+-----+
|      avg(stars)|stars|                name|     city|state|
+----------------+-----+--------------------+---------+-----+
|4.11784140969163|  4.0|Delmonico Steakhouse|Las Vegas|   NV|
|             4.5|  4.5|Mr. Pancho Mexica...|     Mesa|   AZ|
|            3.75|  4.0|Maricopa County D...|  Phoenix|   AZ|
|             4.0|  4.0|Double Play Sport...|Las Vegas|   NV|
|          2.6875|  2.5|  Impressions Dental| Chandler|   AZ|
+----------------+-----+--------------------+---------+-----+
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:

```
(row['avg(stars)'] - row['stars']) / row['stars']
```

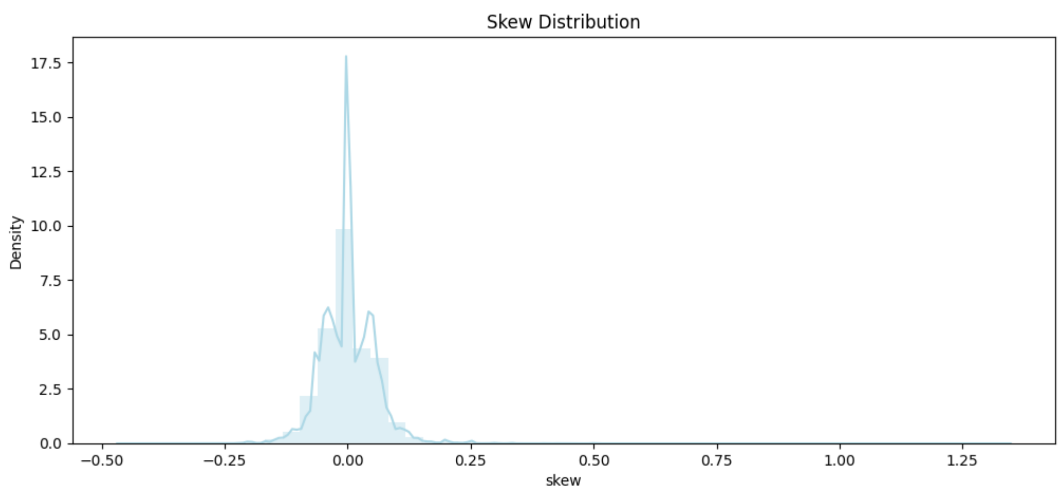
If the **skew** is negative, we can interpret that to be: reviewers who left a written response were more dissatisfied than normal. If **skew** is positive, we can interpret that to be: reviewers who left a written response were more satisfied than normal.

In [19]:
```
skew_df = business.select('avg(stars)', 'stars', 'name', 'city', 'state').withColumn('s
skew_df.show(10)
```

```
+------------------+-----+--------------------+----------+-----+--------------------+
|        avg(stars)|stars|                name|      city|state|                skew|
+------------------+-----+--------------------+----------+-----+--------------------+
|  4.11784140969163|  4.0|Delmonico Steakhouse| Las Vegas|   NV|0.029460352422907565|
|               4.5|  4.5|Mr. Pancho Mexica...|      Mesa|   AZ|                 0.0|
|              3.75|  4.0|Maricopa County D...|   Phoenix|   AZ|             -0.0625|
|               4.0|  4.0|Double Play Sport...| Las Vegas|   NV|                 0.0|
|            2.6875|  2.5|  Impressions Dental|  Chandler|   AZ|               0.075|
| 4.976744186046512|  5.0|    Kidz Cuts By Lori| Henderson|   NV|-0.00465116279069...|
|3.8107142857142855|  4.0|Río Mirage Café y...| El Mirage|   AZ|-0.04732142857142...|
|3.7941176470588234|  4.0|    Steep & Brew West|   Madison|   WI|-0.05147058823529416|
|1.4762931034482758|  1.5|       Showtime Tours| Las Vegas|   NV|-0.01580459770114...|
|               2.0|  2.0|August Moon Chine...|Woodbridge|   ON|                 0.0|
+------------------+-----+--------------------+----------+-----+--------------------+
only showing top 10 rows
```

And finally, graph it!

In [20]:
```
pd_skew = skew_df.toPandas()
fig = plt.figure(figsize=(12,5))
ax = sns.distplot(pd_skew['skew'], hist=True, kde=True, color = 'lightblue').set_title(
plt.show()
%matplot plt
```



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

From this distribution, we can see that the Yelp reviews are skewed to the left. If the skew number is 0, it means that the reviews are neutral and not skewed. However, there are many more reviews with negative skew numbers. Thus, reviewers who left a written response were more dissatisfied than normal

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

In [21]:
```
user = spark.read.json('s3://sta9760-project2-yelpdataset/yelp_academic_dataset_user.js
user.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 [22]:
```
user.select('user_id', 'average_stars', 'elite').show(5, truncate=False)
```

```
+----------------------+-------------+--------------------------------------------+
|user_id               |average_stars|elite                                       |
+----------------------+-------------+--------------------------------------------+
|ntlvfPzc8eglqvk92iDIAw|3.57         |                                            |
|FOBRPlBHa3WPHFB5qYDlVg|3.84         |2008,2009,2010,2011,2012,2013               |
|zZUnPeh2hEp0WydbAZEOOg|3.44         |2010                                        |
|QaELAmRcDc5TfJEylaaP8g|3.08         |2009                                        |
|xvu8G900tezTzbbfqmTKvA|4.37         |2009,2010,2011,2012,2014,2015,2016,2017,2018|
+----------------------+-------------+--------------------------------------------+
only showing top 5 rows
```

Create the elite user subset from the total user dataset

In [23]:
```
import pyspark.sql.functions as F
df = user.select('user_id', 'average_stars', 'elite').withColumn("length_of_elite", F.l
df.show(truncate=False)
df.count()
```

```
+----------------------+-------------+--------------------------------------------+-----
----------+
|user_id               |average_stars|elite                                       |lengt
h_of_elite|
+----------------------+-------------+--------------------------------------------+-----
----------+
|ntlvfPzc8eglqvk92iDIAw|3.57         |                                            |0
|
|FOBRPlBHa3WPHFB5qYDlVg|3.84         |2008,2009,2010,2011,2012,2013               |29
|
|zZUnPeh2hEp0WydbAZEOOg|3.44         |2010                                        |4
|
|QaELAmRcDc5TfJEylaaP8g|3.08         |2009                                        |4
|
|xvu8G900tezTzbbfqmTKvA|4.37         |2009,2010,2011,2012,2014,2015,2016,2017,2018|44
|
|z5_82komKV3mI4ASGe2-FQ|2.88         |2007                                        |4
|
|ttumcu6hWshk_EJVWrduDg|4.0          |                                            |0
|
|f4_MRNHvN-yRn7EA8YWRxg|3.63         |2011,2012,2013,2014,2015,2016,2017,2018     |39
|
|UYACF30806j2mfbB5vdmJA|3.75         |                                            |0
|
|QG13XBbgHWydzThRBGJtyw|4.1          |2008,2009                                   |9
|
|f6YuZP6iennHFVlnFJOXLQ|3.8          |                                            |0
|
|I_6wY8_RsewziNnKhGZg4g|3.63         |2010,2011,2012,2013,2014,2015,2016,2017     |39
|
|q-v8elVPvKz0KvK69QSj1Q|3.37         |2011,2012,2013,2014,2015,2016,2017,2018     |39
|
|HwPGLzF_uXB3MF8bc5u5dg|4.5          |                                            |0
|
|y4UuVowA9i3zj2hHyRMfHw|4.17         |                                            |0
|
|1WBxJ2r3A2QYfRSEzgcmkQ|3.82         |2010,2011,2012,2013,2014,2015,2016          |34
|
|-TT5e-YQU9xLb1JAGCGkQw|3.91         |2010,2011,2012,2013                         |19
|
|6bbHSJ0PrgSxh7e5nigKMw|2.21         |                                            |0
|
|4VmuXuSRhv5UxYUy3tMpiQ|3.88         |2012,2013                                   |9
|
|pVU2DdtBFppBAX5G5t3rcw|3.79         |                                            |0
|
+----------------------+-------------+--------------------------------------------+-----
----------+
only showing top 20 rows

1968703
```

In total, there are 1968703 users in this dataset.

In [24]:
```
elite = user.select('user_id', 'elite').where(length(col("elite")) > 0)
```

```
elite.count()
```

```
75961
```

There is a total of 75961 elite users throughout the years in this dataset.

Join elite user dataset with review dataset on user_id to find out the reviews they have submitted:

In [25]:
```
elite_reviews = elite.join(review, 'user_id', 'inner')
elite_reviews.select('business_id', 'user_id', 'review_id', 'stars').show(10)
```

```
+--------------------+--------------------+--------------------+-----+
|         business_id|             user_id|           review_id|stars|
+--------------------+--------------------+--------------------+-----+
|-8F04F54iDT6VgWPC...|1Dul59QEe-Q-7OQHT...|Kg5ncegiJ3utWRxDt...|  4.0|
|p2OOk46G_AOO0nCWl...|3pMczoCBOSKBcqMhV...|nBNDv9j_tiPPo5MMe...|  5.0|
|jyFoxS8MofdpkAAK6...|jO44Apni7iJZVVK4H...|EI6L-L0Dcj6HAUaBO...|  1.0|
|ewty6EB70nwPJsUkA...|RO78oDy7vbEcOJU8a...|Ryohf9HJcpk2C49vf...|  4.0|
|0M3KCmdY-_xlIu5vE...|TFxeEvpjMNQ3AWL49...|Lc0Tj-Me2Jwu_V9au...|  5.0|
|-h0o-BilkKaCa7HX9...|Fl1oTs6usaCfyjLnY...|GORTMUfkTtGViv4ap...|  5.0|
|DEtOIjhV0MWZ8fD8-...|RO78oDy7vbEcOJU8a...|aULkXMFrsMvctmJ5Q...|  5.0|
|Jt28TYWanzKrJYYr0...|LEr8vS6PRymCg-SJH...|JfNrW6b2mgcynJ3w2...|  2.0|
|NFm869_w6cvVaWaNp...|M7vDDzoPNQDN2FdTc...|OcaQAzflKbxKLS2rT...|  5.0|
|Da6eZFThE9xanUAGN...|Ania9MCwET-TBzVjV...|wuNIHeqK_pjpE4Hrp...|  4.0|
+--------------------+--------------------+--------------------+-----+
only showing top 10 rows
```

Calculate sum of elite user ratings each business received:

In [26]:
```
ereview_count_by_business = elite_reviews.select('business_id', 'user_id', 'review_id',
ereview_count_by_business = ereview_count_by_business.withColumnRenamed("count","erevie
ereview_count_by_business.show(5)
```

```
+--------------------+-------------+
|         business_id|ereview_count|
+--------------------+-------------+
|VHsNB3pdGVcRgs6C3...|           25|
|-I06hkMFrX0KBqu61...|            1|
|RMjCnixEY5i12Ciqn...|           26|
|ipFreSFhjClfNETuM...|           17|
|Qm2datcYBPXrPATVG...|            3|
+--------------------+-------------+
only showing top 5 rows
```

Calculate the average elite rating each business received:

In [27]:
```
avg_reviews_by_elite = elite_reviews.select('business_id', 'user_id', 'review_id', 'sta
avg_reviews_by_elite = avg_reviews_by_elite.withColumnRenamed("avg(stars)","avg_e_stars
avg_reviews_by_elite = avg_reviews_by_elite.join(ereview_count_by_business, 'business_i
avg_reviews_by_elite.show(5)
```

```
+--------------------+------------------+-------------+
|         business_id|       avg_e_stars|ereview_count|
+--------------------+------------------+-------------+
|--9e1ONYQuAa-CB_R...|4.1916058394160585|          548|
```

```
|--phjqoPSPa8sLmUV...|               4.0|           4|
|--q7kSBRb0vWC8lSk...|               4.0|           1|
|-0ZO00Vm2ADchytlE...|               5.0|           8|
|-1VaIJza42Hjev6uk...| 3.793103448275862|          29|
+--------------------+------------------+------------+
only showing top 5 rows
```

Join the average elite stars dataset with the business dataset:

In [28]:
```
avg_reviews_by_elite = avg_reviews_by_elite.join(business, 'business_id', 'inner')
avg_reviews = avg_reviews_by_elite.select('business_id', 'stars', 'avg_e_stars', 'erevi
avg_reviews.show(5)
```

```
+--------------------+-----+------------------+-------------+
|         business_id|stars|       avg_e_stars|ereview_count|
+--------------------+-----+------------------+-------------+
|--9e1ONYQuAa-CB_R...|  4.0|4.1916058394160585|          548|
|--phjqoPSPa8sLmUV...|  4.0|               4.0|            4|
|--q7kSBRb0vWC8lSk...|  4.0|               4.0|            1|
|-0ZO00Vm2ADchytlE...|  5.0|               5.0|            8|
|-1VaIJza42Hjev6uk...|  4.0| 3.793103448275862|           29|
+--------------------+-----+------------------+-------------+
only showing top 5 rows
```

Calculate sum of all user ratings each business received:

In [29]:
```
allreview_count_by_business = review.groupby('business_id').count()
allreview_count_by_business = allreview_count_by_business.withColumnRenamed("count","al
allreview_count_by_business.show(5)
```

```
+--------------------+---------------+
|         business_id|allreview_count|
+--------------------+---------------+
|VHsNB3pdGVcRgs6C3...|            136|
|RMjCnixEY5i12Ciqn...|             79|
|ipFreSFhjClfNETuM...|             80|
|dLDMU8bOLnkDTmPUr...|             35|
|Qm2datcYBPXrPATVG...|             17|
+--------------------+---------------+
only showing top 5 rows
```

Joining all rating data together:

In [30]:
```
avg_reviews = avg_reviews.join(allreview_count_by_business, 'business_id', 'inner')
avg_reviews.show(5)
```

```
+--------------------+-----+------------------+-------------+---------------+
|         business_id|stars|       avg_e_stars|ereview_count|allreview_count|
+--------------------+-----+------------------+-------------+---------------+
|--9e1ONYQuAa-CB_R...|  4.0|4.1916058394160585|          548|           1816|
|--phjqoPSPa8sLmUV...|  4.0|               4.0|            4|             12|
|--q7kSBRb0vWC8lSk...|  4.0|               4.0|            1|              7|
|-0ZO00Vm2ADchytlE...|  5.0|               5.0|            8|             86|
|-1VaIJza42Hjev6uk...|  4.0| 3.793103448275862|           29|            280|
+--------------------+-----+------------------+-------------+---------------+
only showing top 5 rows
```

Calculate the percentage of elite ratings each business received:

In [31]:
```
avg_reviews = avg_reviews.withColumn('elite_percentage', (col('ereview_count') / col('a
avg_reviews = avg_reviews.withColumnRenamed("stars","avg_stars")
avg_reviews.show(5)
```

```
+--------------------+---------+------------------+-------------+---------------+-------
-----------+
|         business_id|avg_stars|       avg_e_stars|ereview_count|allreview_count|  elite
_percentage|
+--------------------+---------+------------------+-------------+---------------+-------
-----------+
|--9e1ONYQuAa-CB_R...|      4.0|4.1916058394160585|          548|           1816|30.1762
11453744493|
|--phjqoPSPa8sLmUV...|      4.0|               4.0|            4|             12| 33.333
33333333333|
|--q7kSBRb0vWC8lSk...|      4.0|               4.0|            1|              7|14.2857
14285714285|
|-0ZO00Vm2ADchytlE...|      5.0|               5.0|            8|             86|  9.302
32558139535|
|-1VaIJza42Hjev6uk...|      4.0| 3.793103448275862|           29|            280|10.3571
42857142858|
+--------------------+---------+------------------+-------------+---------------+-------
-----------+
only showing top 5 rows
```

Let's take a look at the top 25 businesses with the highest number of reviews

In [32]:
```
avg_reviews.orderBy('allreview_count', ascending=False).show(25)
```

```
+--------------------+---------+------------------+-------------+---------------+-------
-----------+
|         business_id|avg_stars|       avg_e_stars|ereview_count|allreview_count|  elite
_percentage|
+--------------------+---------+------------------+-------------+---------------+-------
-----------+
|RESDUcs7fIiihp38-...|      4.0| 4.060636515912898|         2985|          10417|28.6550
83037342806|
|4JNXUYY8wbaaDmk3B...|      4.0|  4.13179992698065|         2739|           9536|28.7227
34899328863|
|K7lWdNUhCbcnEvI0N...|      3.5|3.8233124308373294|         2711|           7594| 35.699
23623913616|
|f4x1YBxkLrZg652xt...|      4.0| 3.901702361339923|         1821|           6859| 26.549
05962968363|
|cYwJA2A6I12KNkm2r...|      4.0| 3.938221317040054|         1473|           5586| 26.369
49516648765|
|DkYS3arLOhA8si5uU...|      4.5|  4.27808988764045|         2492|           5370| 46.405
95903165735|
|faPVqws-x-5k2CQKD...|      4.5| 4.403796376186367|         1159|           4979|23.2777
66619803174|
|5LNZ67Yw9RD6nf4_U...|      4.0| 4.263126131563066|         1657|           4973| 33.319
92760908908|
|2weQS-RnoOBhb1KsH...|      3.5| 3.819277108433735|         1826|           4953| 36.866
54552796285|
|iCQpiavjjPzJ5_3gP...|      4.0| 4.105722599418041|         2062|           4882| 42.236
78820155674|
|AV6weBrZFFBfRGCbc...|      2.5| 2.957685664939551|         1158|           4819|24.0298
81718198798|
|vHz2RLtfUMVRPFmd7...|      4.5| 4.394230769230769|          728|           4801|15.1635
07602582795|
```

```
|SMPbvZLSMMb7KU76Y...|       3.5| 3.845360824742268|         1455|              4749| 30.638
02905874921|
|ujHiaprwCQ5ewziu0...|       3.5| 3.616777308388654|         1657|              4731| 35.024
30775734517|
|El4FC8jcawUVgw_0E...|       3.0|3.4377713458755426|         1382|              4589|30.1154
93571584224|
|QXV3L_QFGj8r6nWX2...|       4.5| 4.193423597678917|          517|              4357|11.8659
62818453065|
|rcaPajgKOJC2vo_l3...|       4.0| 4.120710059171597|         1690|              4305|39.2566
78281068524|
|JDZ6_yycNQFTpUZzL...|       4.5| 4.154639175257732|          388|              4225| 9.1834
31952662723|
|OETh78qcgDltvHULo...|       4.0| 4.213844252163164|          809|              4217|19.1842
54209153426|
|3kdSl5mo9dWC4clrQ...|       4.5| 4.297560975609756|          615|              4125|14.9090
90909090908|
|YJ8ljUhLsz6CtT_2O...|       3.5|  3.62015503875969|          774|              4119|18.7909
68681718866|
|KskYqH1Bi7Z_61pH6...|       4.0| 4.215588723051409|         1206|              4119|   29.2
78951201748|
|RwMLuOkImBIqqYj4S...|       4.0| 4.294117647058823|         1020|              4088|24.9510
76320939332|
|FaHADZARwnY4yvlvp...|       3.5| 3.411930673115679|         2481|              4064| 61.048
22834645669|
|u_vPjx925UPEG9DFO...|       2.5| 3.091928251121076|          892|              4024|22.1669
98011928428|
+--------------------+---------+------------------+-------------+-----------------+-------
-----------+
only showing top 25 rows
```

In [33]:
```
avg_reviews.summary().show()
```

```
+-------+--------------------+------------------+------------------+-----------------+---
---------------+------------------+
|summary|         business_id|         avg_stars|       avg_e_stars|    ereview_count|
allreview_count|  elite_percentage|
+-------+--------------------+------------------+------------------+-----------------+---
---------------+------------------+
|  count|              148225|            148225|            148225|           148225|
148225|            148225|
|   mean|                null|3.571276775172879| 3.781229949617148|11.84906054983977|51.
138910440209145|29.071736936940418|
| stddev|                null|0.904201241413331|0.9734324285178444|41.04359275261798| 14
9.5476463476886| 19.78786142447515|
|    min|--1UhMGODdWsrMast...|               1.0|               1.0|                1|
3|0.5208333333333333|
|    25%|                null|               3.0| 3.232142857142857|                1|
6|14.285714285714285|
|    50%|                null|               3.5|               4.0|                3|
15|              25.0|
|    75%|                null|               4.0|               4.5|                9|
43| 38.83495145631068|
|    max|zzzaIBwimxVej4tY6...|               5.0|               5.0|             2985|
10417|             100.0|
+-------+--------------------+------------------+------------------+-----------------+---
---------------+------------------+
```

- There are many businesses with very few reviews. We are going to filter them out and keep only businesses that have at least 25 total reviews.
- We also want to filter out businesses that have less than 5 elite reviews and also businesses where elite reviews account for less than 5% of total reviews.

- This way, we have quality data points to calculate the businesses' average reviews and average elite reviews.

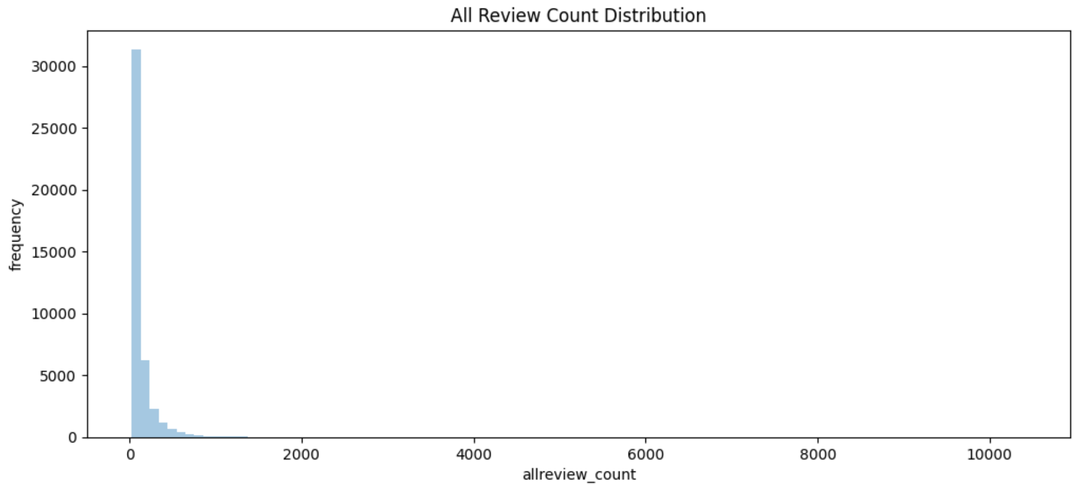
In [34]:
```
avg_reviews = avg_reviews.where(col("allreview_count") >= 25)
avg_reviews = avg_reviews.where(col("ereview_count") >= 5)
avg_reviews = avg_reviews.where(col("elite_percentage") >= 5)
```

In [35]:
```
avg_reviews.summary().show()
```

```
+-------+--------------------+------------------+------------------+------------------+--
----------------+------------------+
|summary|         business_id|         avg_stars|       avg_e_stars|     ereview_count|
allreview_count|  elite_percentage|
+-------+--------------------+------------------+------------------+------------------+--
----------------+------------------+
|  count|               43291|             43291|             43291|             43291|
43291|             43291|
|   mean|                null|3.6322561271395903| 3.781319959083377|33.84530271880991|13
9.58448638285094| 26.40005955217848|
| stddev|                null|0.7020745673904851|0.5875925928795263| 71.2158803696474|25
3.57361456349582|13.860105184781096|
|    min|--1UhMGODdWsrMast...|               1.0|               1.0|                 5|
25|               5.0|
|    25%|                null|               3.0|3.4166666666666665|                 9|
42|            15.625|
|    50%|                null|               3.5|3.8260869565217392|                17|
72| 23.88059701492537|
|    75%|                null|               4.0| 4.186440677966102|                33|
140| 35.13513513513514|
|    max|zzzaIBwimxVej4tY6...|               5.0|               5.0|              2985|
10417|            96.875|
+-------+--------------------+------------------+------------------+------------------+--
----------------+------------------+
```

Our dataset looks a lot better after some filtering and cleaning.

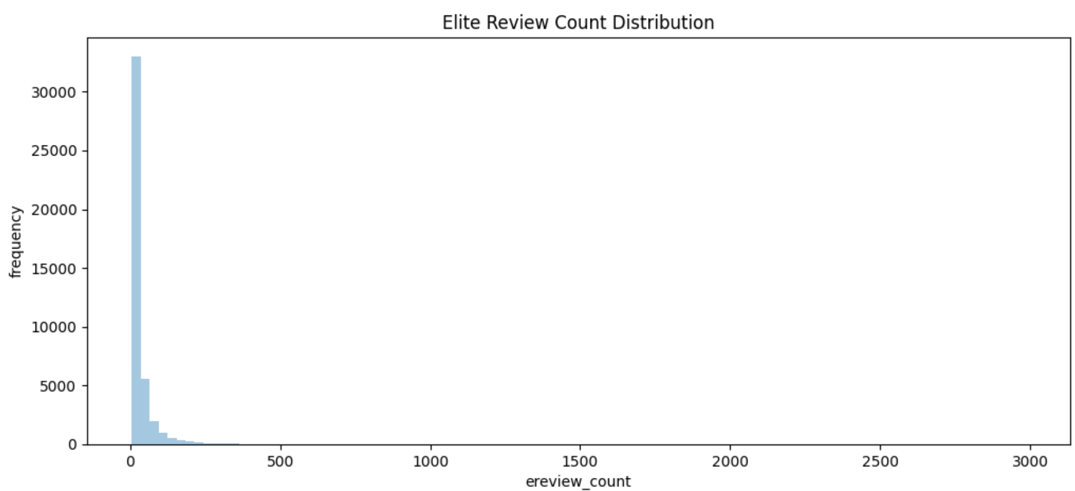
In [36]:
```
pd_avg_reviews = avg_reviews.toPandas()
fig = plt.figure(figsize=(12,5))
ax = sns.distplot(pd_avg_reviews['allreview_count'], hist=True, bins=100, kde=False).se
plt.xlabel("allreview_count")
plt.ylabel("frequency")
plt.show()
%matplot plt
```



In [37]:
```
pd_avg_reviews['allreview_count'].mode()
```

```
0    25
dtype: int64
```

In [38]:
```
fig = plt.figure(figsize=(12,5))
ax = sns.distplot(pd_avg_reviews['ereview_count'], hist=True, bins=100, kde=False).set_
plt.xlabel("ereview_count")
plt.ylabel("frequency")
plt.show()
%matplot plt
```



Let's take a look at the difference between average elite rating and average rating:

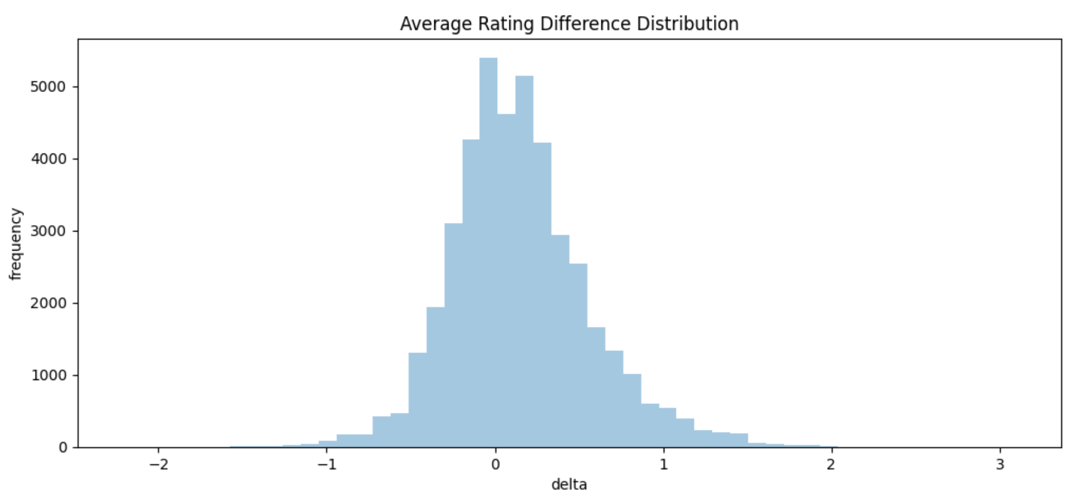
In [39]:
```
avg_reviews = avg_reviews.withColumn('delta', (col('avg_e_stars') - col('avg_stars')))
avg_reviews.show(10)
```

```
+--------------------+---------+------------------+-------------+--------------+-------
-----------+-------------------+
|         business_id|avg_stars|       avg_e_stars|ereview_count|allreview_count|   elite
_percentage|              delta|
+--------------------+---------+------------------+-------------+--------------+-------
```

```
-----------+-------------------+
|--9e1ONYQuAa-CB_R...|       4.0|4.1916058394160585|           548|             1816|30.1762
11453744493|0.19160583941605847|
|-0ZO00Vm2ADchytlE...|       5.0|               5.0|             8|               86|  9.302
32558139535|                0.0|
|-1VaIJza42Hjev6uk...|       4.0| 3.793103448275862|            29|              280|10.3571
42857142858|-0.2068965517241379|
|-2Arz8twKJmxHMS3S...|       4.0|               3.7|            10|               34|29.4117
64705882355|-0.2999999999999998|
|-2ToCaDFpTNmmg3QF...|       1.5|          2.109375|            64|              464|13.7931
03448275861|           0.609375|
|-2wh_ZsD2n5xFYgzp...|       4.0|3.7058823529411766|            17|               34|
50.0|-0.2941176470588234|
|-5NXoZeGBdx3Bdk70...|       4.0|               3.4|            10|               76|13.1578
94736842104|-0.6000000000000001|
|-5__awbuGMHAk6cZg...|       3.0|2.4285714285714284|             7|               78| 8.9743
58974358974|-0.5714285714285716|
|-ArzVOksIBWmtM1ey...|       4.5| 4.666666666666667|            12|               57|21.0526
31578947366|0.16666666666666696|
|-BbnAc9YEO6pjvJGE...|       4.0|4.2342342342342345|           111|              270| 41.111
11111111111| 0.2342342342342345|
+--------------------+----------+------------------+--------------+-----------------+-------
-----------+-------------------+
only showing top 10 rows
```

Now, let's look at the distribution of this delta:

In [40]:
```python
pd_avg_reviews = avg_reviews.toPandas()
fig = plt.figure(figsize=(12,5))
ax = sns.distplot(pd_avg_reviews['delta'], hist=True, kde=False).set_title('Average Rat
plt.xlabel("delta")
plt.ylabel("frequency")
plt.show()
%matplot plt
```



We see that the difference between average elite reviews and average total reviews is normally distributed. There might be some skewness to the right. We will first look at the summary of the distribution:

In [41]:
```python
avg_reviews.select('delta').summary().show()
```

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

```
|summary|               delta|
+-------+--------------------+
|  count|               43291|
|   mean|   0.149063831943786|
| stddev| 0.42068188793625066|
|    min| -2.2142857142857144|
|    25%|-0.11538461538461542|
|    50%| 0.11224489795918347|
|    75%|               0.375|
|    max|  3.0999999999999996|
+-------+--------------------+
```

From an initial glance, we see that the mean is around 0.15 which suggests that elite users give a rating of 0.15 stars higher than the average of total users' ratings. The standard deviation is 0.42 which suggests that we should perform a hypothesis test in order to statistically reject the null hypothesis (average rating of elite users equals the average rating of all users).

Hypothesis Testing

1. Determine a null and alternate hypothesis:

- Null Hypothesis: Average rating of elite users and all users are the same
- Alternate Hypothesis: Average rating of elite users and all users are different

2. Sample size

From the summary above of the dataset, the sample size for our test is 43291. (N = 43291)

In [42]:
```
N = 43291
```

3. Determine a confidence interval and degrees of freedom

We select α = 0.05 - there is 95% confidence that the conclusion of this test will be valid.
Degree of freedom:
df = sample size (elite users) + sample size (all users) -2 = 43291 + 43291 - 2

In [43]:
```
degree_freedom = 2*avg_reviews.count() - 2
```

In [44]:
```
print(degree_freedom)
```

```
86580
```

4. Calculate standard deviation

Calculate the variance

In [45]:
```
from pyspark.sql.functions import var_samp
var_elite = avg_reviews.select(var_samp("avg_e_stars")).head()[0]
var_all = avg_reviews.select(var_samp("avg_stars")).head()[0]
```

In [46]:
```
print(var_elite)
print(var_all)
```

```
0.3452650552068847
0.4929086981765366
```

Calculate the standard deviation

In [47]:
```
s = np.sqrt((var_elite + var_all)/2)
print(s)
```

```
0.6473691965885546
```

5. Calculate the t-statistics

In [48]:
```
# From our statistic summary of dataset
mean_elite = 3.781319959083376
mean_all = 3.6322561271395903
t = (mean_elite - mean_all)/(s*np.sqrt(2/N))
print(t)
```

```
33.876931165656266
```

6. Compare with critical value

p-value after comparison with t

In [49]:
```
p = 1 - stats.t.cdf(t,df=degree_freedom)
print("t = " + str(t))
print("p = " + str(2*p))
```

```
t = 33.876931165656266
p = 0.0
```

After comparing the t statistic with the critical t value (using the stats module), we obtained a p value of 0.0. A p-value less than 0.05 (typically ≤ 0.05) is statistically significant. It indicates strong evidence against the null hypothesis, as there is less than a 5% probability the null is correct (and the results are random).

Therefore, we reject the null hypothesis, and accept the alternative hypothesis. Thus, it proves that the average rating of elite users and all users are different and that the elite reviewers should not be trusted.