Capstone Project - The Battle of Neighborhoods (Week 1-2)

Business Problem section: Analysis of Restaraunts in 31 Major European Cities using data from TripAdvisor

Before I visit a restaraunt, I like to look it up and check out the menu, reviews, and ratings. So I thought this would be a fun dataset to work with to find where to eat in Europe. I also did some Sentiment Analysis to the reviews available in this data set to see what reviewers mostly say.

You could be on a budget, could be looking for a top rated restaraunt in town, finding a healthy food restaraunt, or looking to see where to have the best cup of coffee. So let's see what I can find!

Data Section

This dataset has been obtained by scraping TA (the famous tourism website) for information about restaurants for a given city. The scraper goes through the restaurants listing pages and fulfills a raw dataset. The raw datasets for the main cities in Europe have been then curated for futher analysis purposes, and aggregated to obtain this dataset.

IMPORTANT: the restaurants list contains the restaurants that are registrered in the TA database only. All the restaurants of a city may not be resgistered in this database.

[https://www.kaggle.com/damienbeneschi/krakow-ta-restaurans-data-raw (https://www.kaggle.com/damienbeneschi/krakow-ta-restaurans-data-raw)](https://www.kaggle.com/damienbeneschi/krakow-ta-restaurans-data-raw)

The dataset contain restaurants information for 31 cities in Europe: Amsterdam (NL), Athens (GR) , Barcelona (ES) , Berlin (DE), Bratislava (SK), Bruxelles (BE), Budapest (HU), Copenhagen (DK), Dublin (IE), Edinburgh (UK), Geneva (CH), Helsinki (FI), Hamburg (DE), Krakow (PL), Lisbon (PT), Ljubljana (SI), London (UK), Luxembourg (LU), Madrid (ES), Lyon (FR), Milan (IT), Munich (DE), Oporto (PT), Oslo (NO), Paris (FR), Prague (CZ), Rome (IT), Stockholm (SE), Vienna (AT), Warsaw (PL), Zurich (CH).

The data is a .csv file comma-separated that contains 125 433 entries (restaurants). It is structured as follow: Name: name of the restaurant City: city location of the restaurant Cuisine Style: cuisine style(s) of the restaurant, in a Python list object (94 046 non-null) Ranking: rank of the restaurant among the total number of restaurants in the city as a float object (115 645 non-null) Rating: rate of the restaurant on a scale from 1 to 5, as a float object (115 658 non-null) Price Range: price range of the restaurant among 3 categories , as a categorical type (77 555 non-null) Number of Reviews: number of reviews that customers have let to the restaurant, as a float object (108 020 non-null) Reviews: 2 reviews that are displayed on the restaurants scrolling page of the city, as a list of list object where the first list contains the 2 reviews, and the second le dates when these reviews were written (115 673 non-null) URL_TA: part of the URL of the detailed restaurant page that comes after 'www.tripadvisor.com' as a string object (124 995 non-null) ID_TA: identification of the restaurant in the TA database constructed a one letter and a number (124 995 non-null)

Missing information for restaurants (for example unrated or unreviewed restaurants) are in the dataset as NaN (numpy.nan).

Methodology section

The Methodology section will describe the main components of our analysis and predication system. The Methodology section comprises four stages:

1. Collect Inspection Data
2. Explore and Understand Data
3. Data preparation and preprocessing
4. Modeling

1. Collect Inspection Data

After importing the necessary libraries, we download the data from the HM Land Registry website as follows:

In [1]:
```
import os # Operating System
import numpy as np
import pandas as pd
import datetime as dt # Datetime
import json # library to handle JSON files

!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # convert an address into latitude and
longitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas
dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

!conda install -c conda-forge folium=0.5.0 --yes
import folium #import folium # map rendering library

print('Libraries imported.')
```

```
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/hmd/anaconda3

  added / updated specs:
    - geopy


The following NEW packages will be INSTALLED:

  python_abi         conda-forge/linux-64::python_abi-3.6-1_cp36m

The following packages will be UPDATED:

  ca-certificates     pkgs/main::ca-certificates-2020.1.1-0 --> conda-forge::
ca-certificates-2020.4.5.1-hecc5488_0
  conda                       pkgs/main::conda-4.8.3-py36_0 --> conda-forge::
conda-4.8.3-py36h9f0ad1d_1

The following packages will be SUPERSEDED by a higher-priority channel:

  certifi              pkgs/main::certifi-2020.4.5.1-py36_0 --> conda-forge::
certifi-2020.4.5.1-py36h9f0ad1d_0
  openssl              pkgs/main::openssl-1.1.1g-h7b6447c_0 --> conda-forge::
openssl-1.1.1g-h516909a_0


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.

Libraries imported.
```

In [2]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import ast
import operator
from matplotlib import cm
from itertools import cycle, islice
%matplotlib inline

import os
print(os.listdir("/home/hmd/Documents"))
```

```
['TA_restaurants_curated.csv.zip', 'TA_restaurants_curated1.csv']
```

In [3]:
```
#Read the data for examination (Source: http://landregistry.data.gov.uk/)
df_ppd = pd.read_csv("/home/hmd/Documents/TA_restaurants_curated1.csv", engi
ne ='python')
```

Before using data, we will have to explore and understand it.

2. Explore and Understand Data

In [4]:
```
df_ppd.head(5)
```

Out[4]:

	Unnamed: 0	Name	City	Cuisine Style	Ranking	Rating	Price Range	Number of Reviews	Reviews	
0	0	Martine of Martine's Table	Amsterdam	['French', 'Dutch', 'European']	1.0	5.0	_ $	136.0	[['Just like home', 'A Warm Welcome to Wintry ...	/F
1	1	De Silveren Spiegel	Amsterdam	['Dutch', 'European', 'Vegetarian Friendly', '...	2.0	4.5		812.0	[['Great food and staff', 'just perfect'], ['0...	/F
2	2	La Rive	Amsterdam	['Mediterranean', 'French', 'International', '...	3.0	4.5		567.0	[['Satisfaction', 'Delicious old school restau...	/F
3	3	Vinkeles	Amsterdam	['French', 'European', 'International', 'Conte...	4.0	5.0		564.0	[['True five star dinner', 'A superb evening o...	/F g
4	4	Librije's Zusje Amsterdam	Amsterdam	['Dutch', 'European', 'International', 'Vegeta...	5.0	4.5		316.0	[['Best meal.... EVER', 'super food experience...	/F g

In [5]:
```
df_ppd.shape
```

Out[5]:
```
(125527, 11)
```

In [6]:
```
df_ppd.describe()
```

Out[6]:

	Unnamed: 0	Ranking	Rating	Number of Reviews
count	125527.000000	115876.000000	115897.000000	108183.000000
mean	3974.686131	3657.463979	3.987441	125.184983
std	4057.687698	3706.255301	0.678814	310.833311
min	0.000000	1.000000	-1.000000	2.000000
25%	1042.000000	965.000000	3.500000	9.000000
50%	2445.000000	2256.000000	4.000000	32.000000
75%	5626.000000	5237.000000	4.500000	114.000000
max	18211.000000	16444.000000	5.000000	16478.000000

In [7]:
```
df_ppd.columns
```

Out[7]:
```
Index(['Unnamed: 0', 'Name', 'City', 'Cuisine Style', 'Ranking', 'Rating',
       'Price Range', 'Number of Reviews', 'Reviews', 'URL_TA', 'ID_TA'],
      dtype='object')
```

3. Data preparation and preprocessing

In [8]:
```
#Lets see if we can remove certain columns
df_ppd.drop(df_ppd.columns[[0, 4]], axis = 1, inplace=True,  errors='raise')
df_ppd.head()
```

Out[8]:

	Name	City	Cuisine Style	Rating	Price Range	Number of Reviews	Reviews	URL_TA	
0	Martine of Martine's Table	Amsterdam	['French', 'Dutch', 'European']	5.0	— $	136.0	[['Just like home', 'A Warm Welcome to Wintry ...	/Restaurant_Review-g188590-d11752080-Reviews-M...	d1
1	De Silveren Spiegel	Amsterdam	['Dutch', 'European', 'Vegetarian Friendly', '...	4.5		812.0	[['Great food and staff', 'just perfect'], ['0...	/Restaurant_Review-g188590-d693419-Reviews-De_...	
2	La Rive	Amsterdam	['Mediterranean', 'French', 'International', '...	4.5		567.0	[['Satisfaction', 'Delicious old school restau...	/Restaurant_Review-g188590-d696959-Reviews-La_...	
3	Vinkeles	Amsterdam	['French', 'European', 'International', 'Conte...	5.0		564.0	[['True five star dinner', 'A superb evening o...	/Restaurant_Review-g188590-d1239229-Reviews-Vi...	d
4	Librije's Zusje Amsterdam	Amsterdam	['Dutch', 'European', 'International', 'Vegeta...	4.5		316.0	[['Best meal.... EVER', 'super food experience...	/Restaurant_Review-g188590-d6864170-Reviews-Li...	d

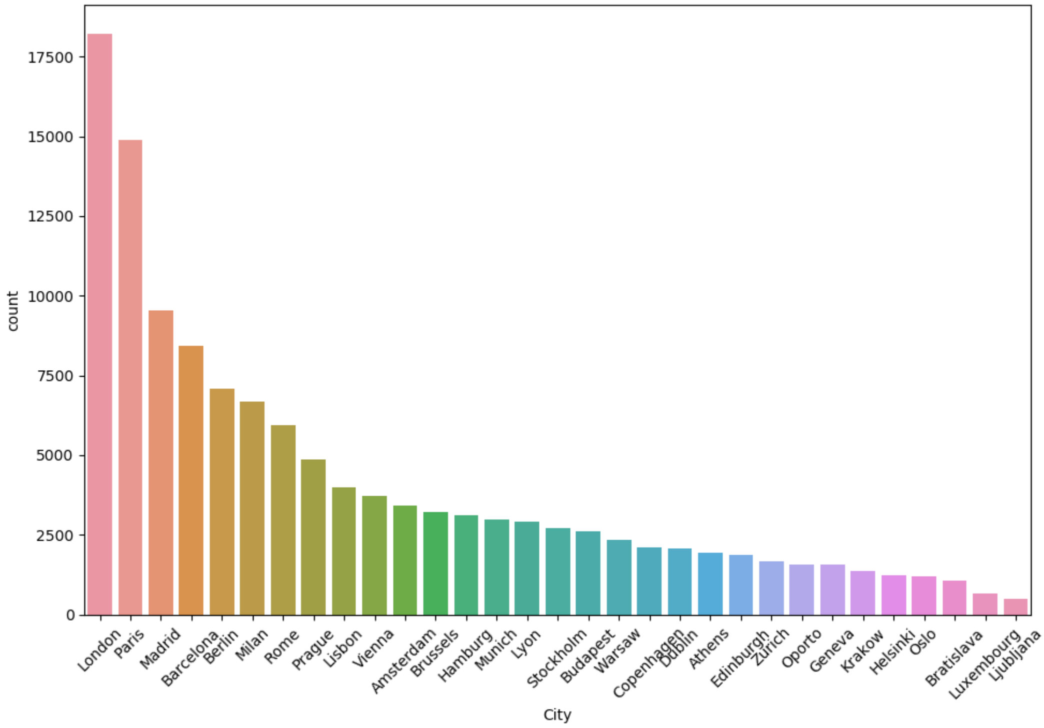
In [9]:
```
df_ppd.count()
```

Out[9]:
```
Name                  125527
City                  125527
Cuisine Style          94176
Rating                115897
Price Range            77672
Number of Reviews     108183
Reviews               115911
URL_TA                125527
ID_TA                 125527
dtype: int64
```

In [10]:
```
df_ppd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125527 entries, 0 to 125526
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Name               125527 non-null  object
 1   City               125527 non-null  object
 2   Cuisine Style      94176 non-null   object
 3   Rating             115897 non-null  float64
 4   Price Range        77672 non-null   object
 5   Number of Reviews  108183 non-null  float64
 6   Reviews            115911 non-null  object
 7   URL_TA             125527 non-null  object
 8   ID_TA              125527 non-null  object
dtypes: float64(2), object(7)
memory usage: 8.6+ MB
```

In [11]:
```
plt.figure(figsize=(10,7), dpi =100)
plot = sns.countplot(df_ppd['City'], order=df_ppd['City'].value_counts().ind
ex)
plot.set_xticklabels(plot.get_xticklabels(), rotation = 45)
plt.tight_layout()
```



we can see that London and Paris have the highest number of reviews & Ljubljana, Luxenbourg have the least number of reviews

Lets try to groupby the city

In [12]:
```
byCity = df_ppd.groupby('City')
byCity['Rating'].mean()
```

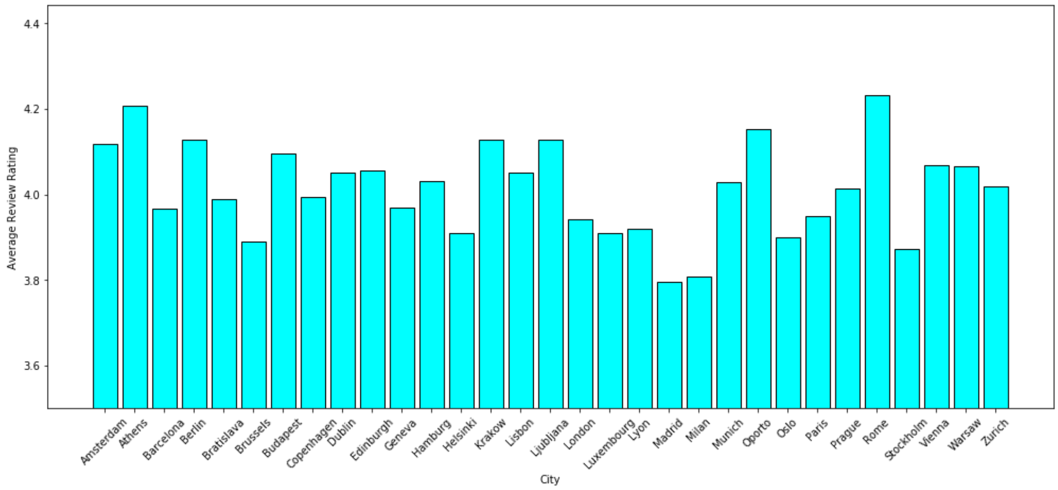
Out[12]:
```
City
Amsterdam     4.118381
Athens        4.207774
Barcelona     3.966829
Berlin        4.127020
Bratislava    3.989314
Brussels      3.890106
Budapest      4.095854
Copenhagen    3.994670
Dublin        4.051151
Edinburgh     4.056818
Geneva        3.969482
Hamburg       4.030508
Helsinki      3.908813
Krakow        4.128812
Lisbon        4.052128
Ljubljana     4.128205
London        3.942896
Luxembourg    3.909310
Lyon          3.920382
Madrid        3.796698
Milan         3.808955
Munich        4.027525
Oporto        4.152145
Oslo          3.899385
Paris         3.948714
Prague        4.013423
Rome          4.232140
Stockholm     3.873528
Vienna        4.067984
Warsaw        4.067102
Zurich        4.018495
Name: Rating, dtype: float64
```

Takeaway: Almost all of the cities have a good avg restaurtant rating.

In [13]:
```
#Average Review Rating per City

x = list()
y = list()
for city in list(df_ppd['City'].unique()):
    x.append(city)
    y.append(df_ppd[df_ppd['City'] == city]['Rating'].mean())
fig, ax = plt.subplots(1,1,figsize=(17,7))
ax.bar(x,y,color = 'cyan',edgecolor = 'black')
ax.set_ylim(bottom=3.5)
ax.set_xticklabels(labels = x, rotation = 45)
ax.set_xlabel('City')
ax.set_ylabel('Average Review Rating')
```

Out[13]:
```
Text(0, 0.5, 'Average Review Rating')
```

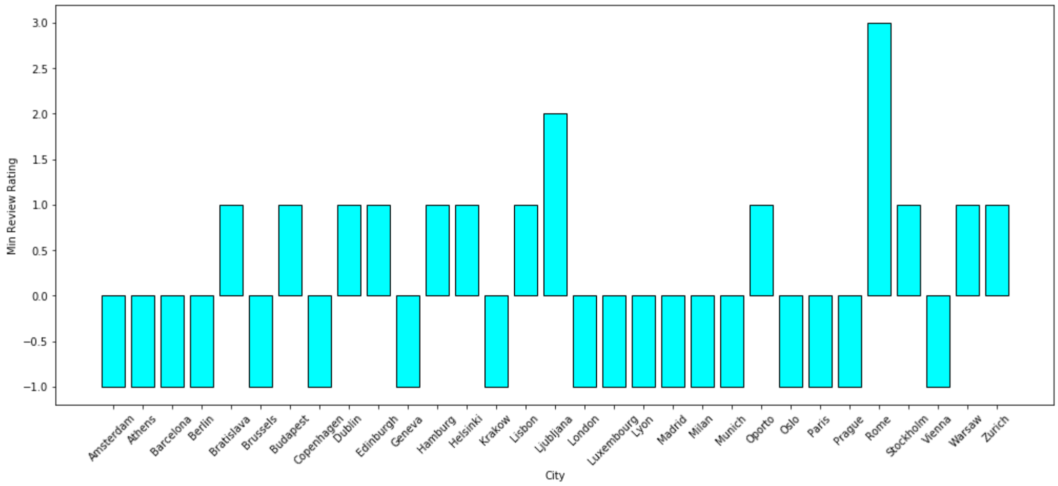


Madrid & Milan seem to have the lowest average rating, Rome & Athens seem to have the highest average rating

In [14]:
```
### Min Review rating per city


x = list()
y = list()
for city in list(df_ppd['City'].unique()):
    x.append(city)
    y.append(df_ppd[df_ppd['City'] == city]['Rating'].min())
fig, ax = plt.subplots(1,1,figsize=(17,7))
ax.bar(x,y,color = 'cyan',edgecolor = 'black')
ax.set_xticklabels(labels = x, rotation = 45)
ax.set_xlabel('City')
ax.set_ylabel('Min Review Rating')
```

Out[14]: Text(0, 0.5, 'Min Review Rating')



few cities have "Negative (-1) Rating". Rome seems to have best ratings amongst all the cities.

In [15]:
```
#Lets look at the Negative Ratings
print('Total negative ratings count : ', len(df_ppd[df_ppd['Rating'] < 0]))
df_ppd[df_ppd['Rating'] < 0]
```

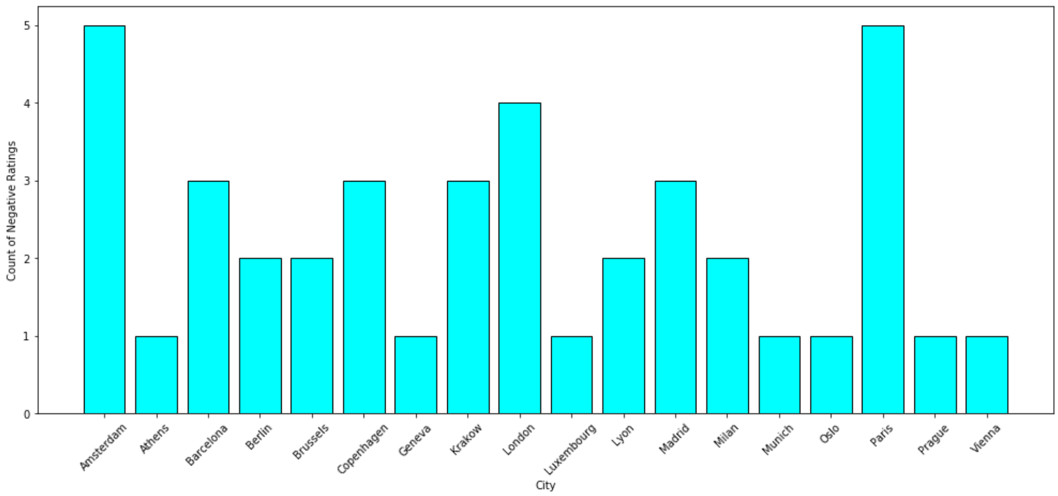
```
Total negative ratings count :  41
```

Out[15]:

	Name	City	Cuisine Style	Rating	Price Range	Number of Reviews	Reviews	URL
3239	Reggae Rita's	Amsterdam	['Caribbean', 'Jamaican']	-1.0	— $	NaN	[['A TRUE BLESSING FOR YOUR STOMACH'], ['01/10...	/Restaurant_Re g188 d12291 Reviews
3240	Lokaal Spaanders	Amsterdam	['French', 'European', 'Fusion', 'Street Food'...	-1.0	— $	NaN	[[], []]	/Restaurant_Re g188 d12333 Review
3241	Bistro Berlage	Amsterdam	['Dutch', 'European']	-1.0	— $	NaN	[['Bistro is changing!'], ['01/09 /2018']]	/Restaurant_Re g188 d13276 Reviews
3242	Fondue oost	Amsterdam	['French', 'German', 'Belgian', 'Dutch', 'Euro...	-1.0	NaN	NaN	[['Might look good on socialMedia, sucks in r....	/Restaurant_Re g188 d13331 Review
3247	Pigs&Punch	Amsterdam	['Bar', 'Barbecue', 'Grill', 'Pub']	-1.0	NaN	NaN	[[], []]	/Restaurant_Re g188 d13356 Review
5221	Sah	Athens	NaN	-1.0	NaN	NaN	[[], []]	/Restaurant_Re g189 d13355 Reviews
13155	La Petite Champagnerie	Barcelona	['International', 'Spanish']	-1.0	$	NaN	[[], []]	/Restaurant_Re g187 d13173 Review
13156	Les Dues Sicilies Rec Comtal	Barcelona	['Italian']	-1.0	— $	NaN	[[], []]	/Restaurant_Re g187 d13199 Review
13158	Vapiano	Barcelona	['Pizza', 'Mediterranean', 'Italian']	-1.0	— $	2.0	[[], []]	/Restaurant_Re g187 d13329 Review
20159	Cafe MokannTi	Berlin	['German', 'African']	-1.0	$	NaN	[[], []]	/Restaurant_Re g187 d12249 Reviews
20167	Spazio - Italian Bistrot	Berlin	['Italian', 'Bar', 'Pub']	-1.0	$	7.0	[['Top products ,italian soul', 'The best plac...	/Restaurant_Re g187 d13344 Reviews
24951	Belga & Co	Brussels	['European']	-1.0	— $	NaN	[[], []]	/Restaurant_Re g188 d13531 Reviews
24953	Bistrot Bocaux	Brussels	['French', 'Belgian', 'European', 'Soups']	-1.0	— $	2.0	[[], []]	/Restaurant_Re g1136 d13540 Review
29715	Kujaku	Copenhagen	['Sushi']	-1.0	NaN	NaN	[[], []]	/Restaurant_Re g189 d12707 Reviews
29716	Karma Sushi Kodbyen	Copenhagen	['Japanese', 'Sushi']	-1.0		NaN	[['Nice atmosphere and friendly staff'], ['01/...	/Restaurant_Re g189 d12908 Reviews

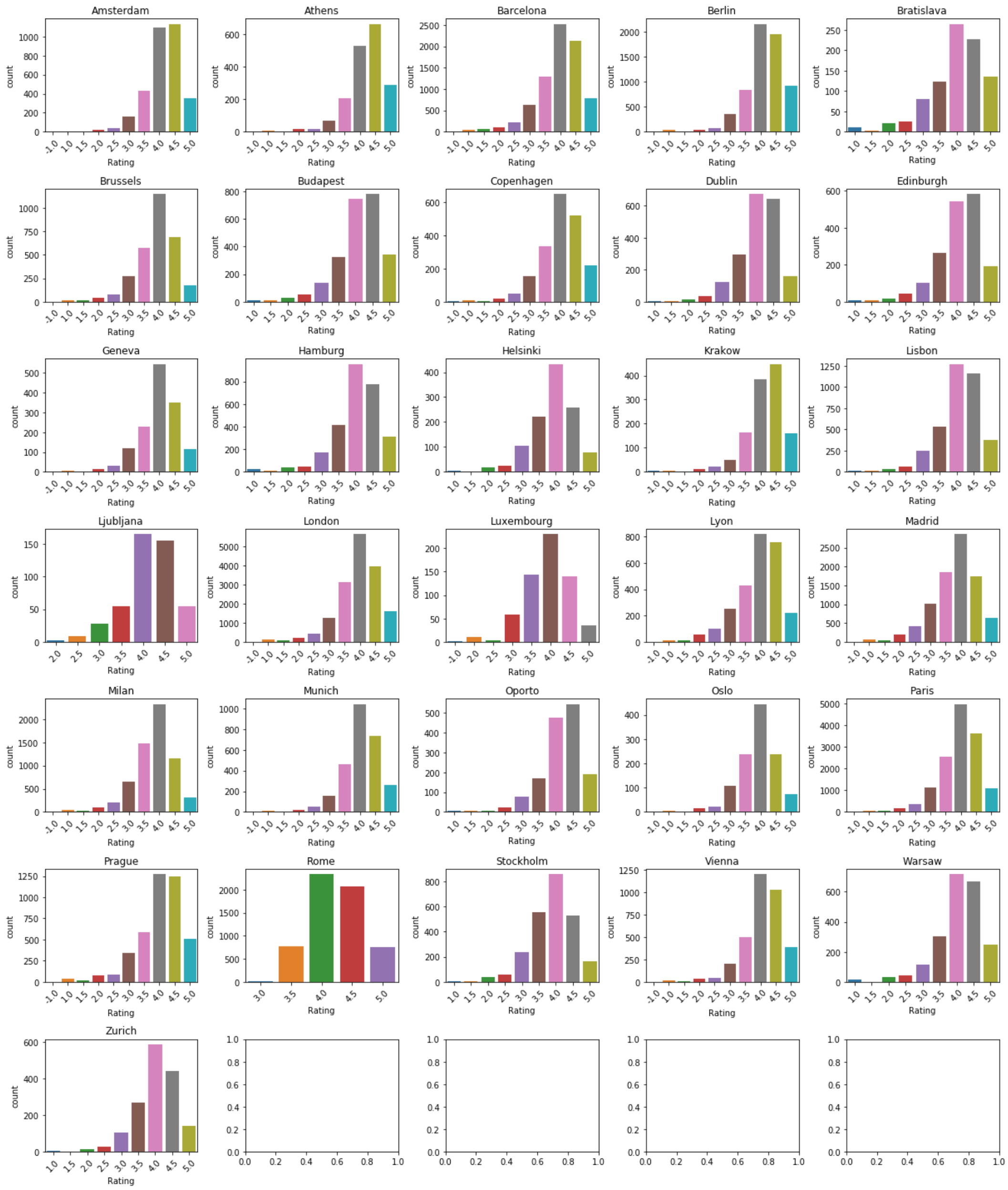
In [16]:
```python
x = list()
y = list()
count = 0
for city in list(df_ppd['City'].unique()):
    count = len(df_ppd[(df_ppd['City'] == city) & (df_ppd['Rating'] < 0)])
    if count > 0:
        y.append(count)
        x.append(city)
    count = 0
fig, ax = plt.subplots(1,1,figsize=(17,7))
ax.bar(x,y,color = 'cyan',edgecolor = 'black')
ax.set_xticklabels(labels = x, rotation = 45)
ax.set_xlabel('City')
ax.set_ylabel('Count of Negative Ratings')
```

Out[16]:
```
Text(0, 0.5, 'Count of Negative Ratings')
```



In [17]:
```
#Ratings count per city


city = list(df_ppd['City'].unique())
fig, axes = plt.subplots(nrows=7,ncols=5,figsize=(17,20))
i = 0
ratings = list(df_ppd['Rating'].unique())
ratingsCount = list()
for c in city:
    reviewsCity = df_ppd[df_ppd['City'] == c]
    plot = sns.countplot(x='Rating', data = reviewsCity, ax=axes.flatten
()[i])
    plot.set_title(c)
    plot.set_xticklabels(plot.get_xticklabels(), rotation = 45)
    plt.tight_layout()
    i = i + 1
```



All of the cities have majority of their restaurant ratings as "Good" (>4.0 stars)!!!

4. Modeling

After exploring the dataset and gaining insights into it, we are ready to use the clustering methodology to analyze real estates. We will use the k-means clustering technique as it is fast and efficient in terms of computational cost, is highly flexible to account for mutations in real estate market in London and is accurate.

In [18]:
```
CLIENT_ID = 'GAVPNY0E40J3IURC5XEDJSAL20LALKUW0BP23E42RYNSVONE' # your Foursq
uare ID
CLIENT_SECRET = '3BVTHOBVRMNMH0X5KAHG5RW0342S5FL0KGPFVCU3AD25G0B1' # your Fo
ursquare Secret
VERSION = '20200421'
```

Lets plot some of the features on the MAP

In [19]:
```
conda install basemap
```

```
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/hmd/anaconda3

  added / updated specs:
    - basemap


The following packages will be SUPERSEDED by a higher-priority channel:

  ca-certificates    conda-forge::ca-certificates-2020.4.5~ --> pkgs/main::ca
-certificates-2020.1.1-0
  certifi            conda-forge::certifi-2020.4.5.1-py36h~ --> pkgs/main::ce
rtifi-2020.4.5.1-py36_0
  conda              conda-forge::conda-4.8.3-py36h9f0ad1d~ --> pkgs/main::co
nda-4.8.3-py36_0
  openssl            conda-forge::openssl-1.1.1g-h516909a_0 --> pkgs/main::op
enssl-1.1.1g-h7b6447c_0


Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [20]:
```
conda install basemap -c conda-forge
```

```
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/hmd/anaconda3

  added / updated specs:
    - basemap


The following packages will be UPDATED:

  ca-certificates     pkgs/main::ca-certificates-2020.1.1-0 --> conda-forge::
ca-certificates-2020.4.5.1-hecc5488_0
  conda                       pkgs/main::conda-4.8.3-py36_0 --> conda-forge::
conda-4.8.3-py36h9f0ad1d_1

The following packages will be SUPERSEDED by a higher-priority channel:

  certifi              pkgs/main::certifi-2020.4.5.1-py36_0 --> conda-forge::
certifi-2020.4.5.1-py36h9f0ad1d_0
  openssl              pkgs/main::openssl-1.1.1g-h7b6447c_0 --> conda-forge::
openssl-1.1.1g-h516909a_0


Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [21]:
```
import os
import conda

conda_file_dir = conda.__file__
conda_dir = conda_file_dir.split('lib')[0]
proj_lib = os.path.join(os.path.join(conda_dir, 'share'), 'proj')
os.environ["PROJ_LIB"] = proj_lib

from mpl_toolkits.basemap import Basemap
```

In [22]:
```
from mpl_toolkits.basemap import Basemap
from matplotlib.patches import Polygon
from matplotlib.collections import PatchCollection
from matplotlib.colors import Normalize
import matplotlib.cm
from numpy import meshgrid
```

In [ ]:
```
data = {'City': ['Amsterdam', 'Athens', 'Barcelona', 'Berlin', 'Bratislava',
'Brussels', 'Budapest', 'Copenhagen', 'Dublin', 'Edinburgh', 'Geneva', 'Hamb
urg', 'Helsinki', 'Krakow', 'Lisbon', 'Ljubljana', 'London', 'Luxembourg', '
Lyon', 'Madrid', 'Milan', 'Munich', 'Oporto', 'Oslo', 'Paris', 'Prague', 'Ro
me', 'Stockholm', 'Vienna', 'Warsaw', 'Zurich'],
        'Lat':  [52.38, 37.98, 42.38, 52.52, 48.14, 50.85, 47.49, 55.67, 53.
34, 55.95, 46.20, 55.55, 60.16, 50.06, 38.72, 46.05, 51.50, 49.81, 45.76, 4
0.41, 45.46, 48.13, 41.15, 59.91, 48.85, 50.07, 41.90, 59.32, 48.20, 52.22,
47.37],
        'Long': [4.9,   23.72, 2.17,  13.40, 17.10, 4.35,  19.04, 12.56, -6.
26, -3.18,  6.14,  9.99, 24.93, 19.94, -9.13, 15.50, 0.12,  6.12,  4.83,  -
3.70,  9.19, 11.58, -8.62, 10.75,  2.35, 14.43, 12.49, 18.06, 16.37, 21.01,
8.54]}

dfr =pd.df_ppd(data, columns = ['City', 'Lat', 'Long'])

#print(place)
print(data['City'])
```

In [ ]:
```
newDf = pd.merge(df_ppd, dfr, how='left', on='City')
newDf.head()
```

In [ ]:
```
for i in range(0, df_ppd['City'].count()):
    if df_ppd.loc[i,'Price'] == 0:
        df_ppd.loc[i,'Price'] = 2

print(df_ppd['Price'].nunique())
print(rdf_ppd['Price'].unique())
sns.countplot(x='Price', data=df_ppd)
```

In [ ]: