Hotels always wondering about the flow of business and what makes people want to stay with us versus somewhere else or vice versa especially since covid has affected soo much that is going on in our lives with traveling being at the top of that list. Most people before they go to new restuarants, bars, hotels, etc, they look at the reviews that the business might have on the internet. Those reviews influence others and if they are likely to want to visit that place or not.

A company could find this review model beneficial when looking at how customers view how they are doing as a company and what they can work on without having to read each view one by one and filtering out certain things. During this project I will train a model that classify review ratings from different hotels using it's text and using different models and see how it performs. I hope to use the other source of data from Datafiniti to make sure the perfomance of my model works because that then means it could be use across any data set with reviews.

Data Selection & EDA

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
#importing and reading file
df = pd.read_csv('Hotel_Reviews.csv')
df.head()
```

Out[2]:

	id	dateAdded	dateUpdated	address	categories	primaryCategories	city	country	
0	AVwc252WIN2L1WUfpqLP	2016-10-30T21:42:42Z	2018-09-10T21:06:27Z	5921 Valencia Cir	Hotels,Hotels and motels,Hotel and motel reser...	Accommodation & Food Services	Rancho Santa Fe	US	us/ca/ranchosantafe/5921vale
1	AVwc252WIN2L1WUfpqLP	2016-10-30T21:42:42Z	2018-09-10T21:06:27Z	5921 Valencia Cir	Hotels,Hotels and motels,Hotel and motel reser...	Accommodation & Food Services	Rancho Santa Fe	US	us/ca/ranchosantafe/5921vale
2	AVwc252WIN2L1WUfpqLP	2016-10-30T21:42:42Z	2018-09-10T21:06:27Z	5921 Valencia Cir	Hotels,Hotels and motels,Hotel and motel reser...	Accommodation & Food Services	Rancho Santa Fe	US	us/ca/ranchosantafe/5921vale
3	AVwdOclqIN2L1WUfti38	2015-11-28T19:19:35Z	2018-09-10T21:06:16Z	7520 Teague Rd	Hotels,Hotels and motels,Travel agencies and b...	Accommodation & Food Services	Hanover	US	us/md/hanover/7520teagu
4	AVwdOclqIN2L1WUfti38	2015-11-28T19:19:35Z	2018-09-10T21:06:16Z	7520 Teague Rd	Hotels,Hotels and motels,Travel agencies and b...	Accommodation & Food Services	Hanover	US	us/md/hanover/7520teagu

5 rows × 25 columns

In [3]:
```
#dimensions of dataframe
df.shape
```

Out[3]: (10000, 25)

In [4]:
```
#column types
df.dtypes
```

Out[4]:
```
id                      object
dateAdded               object
dateUpdated             object
address                 object
categories              object
primaryCategories       object
city                    object
country                 object
keys                    object
latitude               float64
longitude              float64
name                    object
postalCode              object
province                object
reviews.date            object
reviews.dateSeen        object
reviews.rating         float64
reviews.sourceURLs      object
reviews.text            object
reviews.title           object
reviews.userCity        object
reviews.userProvince    object
reviews.username        object
sourceURLs              object
websites                object
dtype: object
```

In [5]:
```
#Finding missing info
print(df.isna().sum())
```

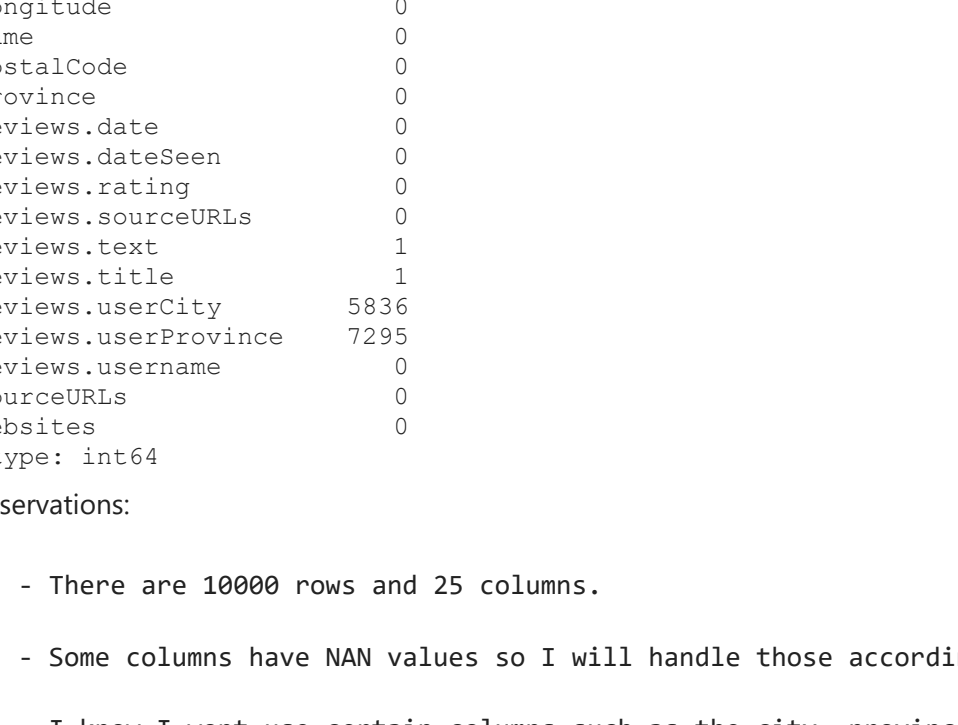
```
id                         0
dateAdded                  0
dateUpdated                0
address                    0
categories                 0
primaryCategories          0
city                       0
country                    0
keys                       0
latitude                   0
longitude                  0
name                       0
postalCode                 0
province                   0
reviews.date               0
reviews.dateSeen           0
reviews.rating             0
reviews.sourceURLs         0
reviews.text               1
reviews.title              1
reviews.userCity        5836
reviews.userProvince    7295
reviews.username           0
sourceURLs                 0
websites                   0
dtype: int64
```

Observations:

```
- There are 10000 rows and 25 columns.

- Some columns have NAN values so I will handle those according .

- I know I wont use certain columns such as the city, province, or username of the people who wrote the
review so I will be dropping them soon .
```

In [6]:
```
#Graph 1 : Histogram of Reviews by Occurance
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize= (8,5))
plt.hist (df['reviews.rating'], bins=4 , color = 'blue', edgecolor='black')
plt.xlabel('Review Rating')
plt.ylabel('Review Occurance')
```

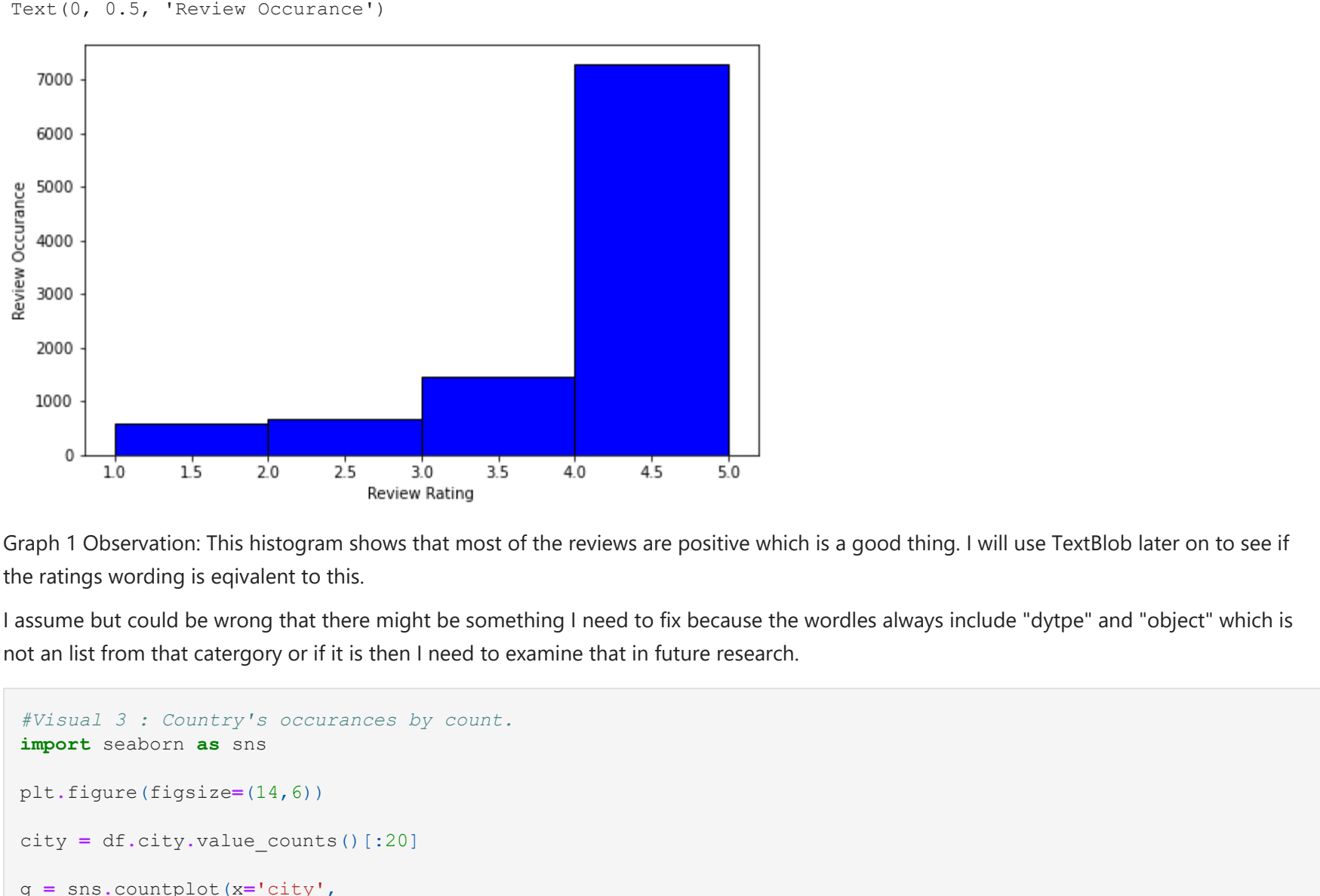
Out[6]: Text(0, 0.5, 'Review Occurance')



Graph 1 Observation: This histogram shows that most of the reviews are positive which is a good thing. I will use TextBlob later on to see if the ratings wording is eqivalent to this.

I assume but could be wrong that there might be something I need to fix because the wordles always include "dytpe" and "object" which is not an list from that catergory or if it is then I need to examine that in future research.

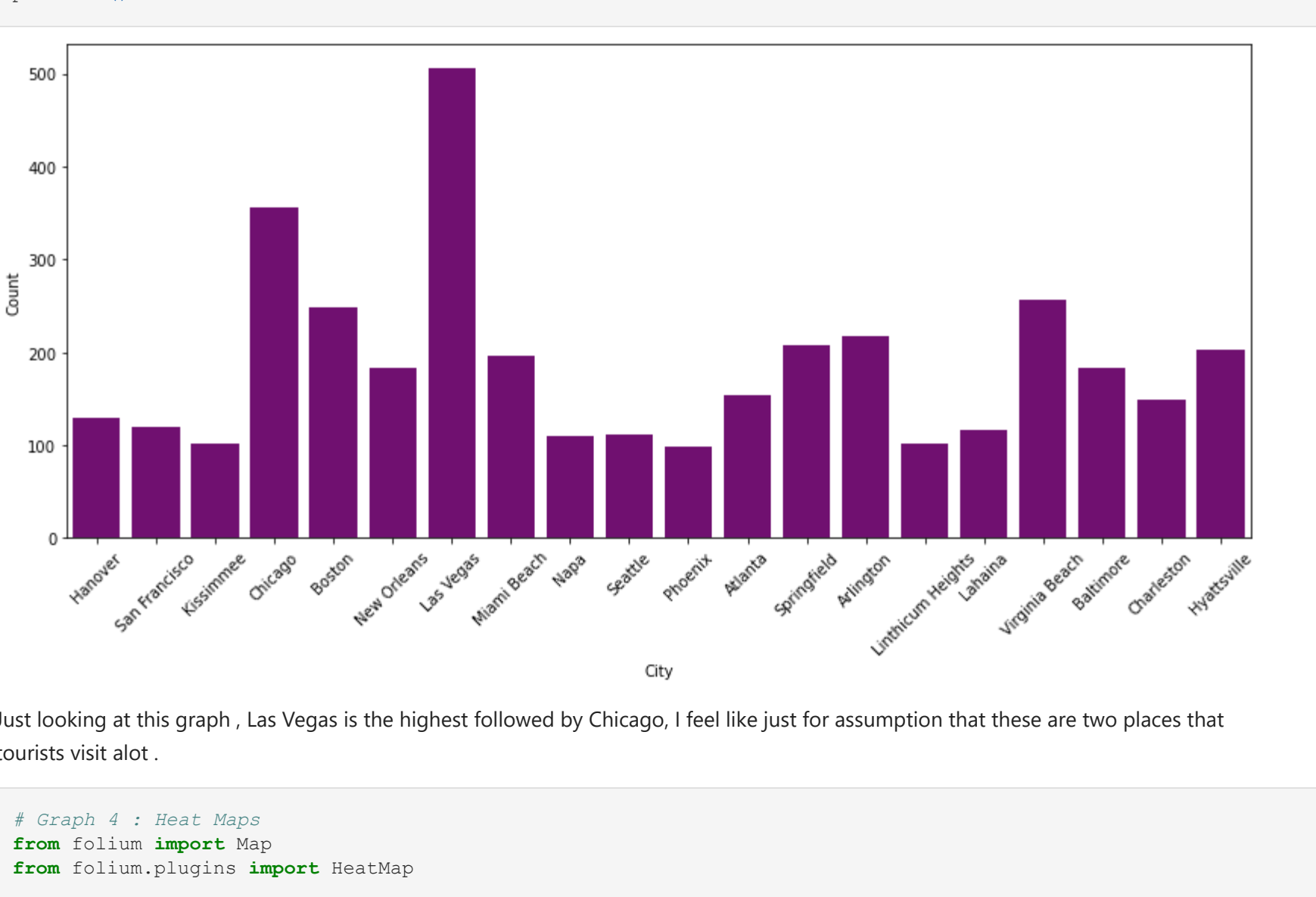
In [7]:
```
#Visual 3 : Country's occurances by count.
import seaborn as sns

plt.figure(figsize=(14,6))

city = df.city.value_counts()[:20]

g = sns.countplot(x='city',
                  data=df[df.city.isin(city.index.values)],
                  color='purple')

g.set_xlabel("City")
g.set_ylabel("Count")
g.set_xticklabels(g.get_xticklabels(),rotation=45)

plt.show()
```



Just looking at this graph , Las Vegas is the highest followed by Chicago, I feel like just for assumption that these are two places that tourists visit alot .

In [8]:
```
# Graph 4 : Heat Maps
from folium import Map
from folium.plugins import HeatMap

location_map = Map(location=[39.155929, -76.716341], zoom_start=4)
hm_wide = HeatMap(list(zip(df['latitude'].values, df['longitude'].values)), min_opacity=0.2, radius=18, blur=15
location_map.add_child(hm_wide)
```

Out[8]:



MILESTONE 2:

In [9]:
```
#dropping unnecessary columns because these are not essential in looking at the reviews of the data.
df = df.drop(['id', 'reviews.dateSeen', 'dateAdded', 'dateUpdated', 'reviews.sourceURLs', 'reviews.userCity', '
```

In [10]:
```
#Dropping the two missing pieces of info from above.
df = df.dropna()
```

In [11]:
```
#iterate through each row to make sure everyone row is a string
for i in range(0,len(df)-1):
    if type(df.iloc[i]['reviews.text']) !=str:
        df.iloc[i]['reviews.text']=str(df.iloc[i])
['reviews.text']
```

Out[11]: ['reviews.text']

In [12]:
```
#Change ratings column to a int .
df['reviews.rating'] = df['reviews.rating'].astype(int)

#get rid of rating of 3
df = df[df['reviews.rating']!=3]

#Seperate positive (4 or 5) and negative (1,2) and no middle
def sentiment (n):
    return 1 if n >=4 else 0

df['sentiment'] = df['reviews.rating'].apply(sentiment)
df.head()
```

Out[12]:

	city	country	latitude	longitude	name	postalCode	province	reviews.date	reviews.rating	reviews.text	reviews.title	sentimen
0	Rancho Santa Fe	US	32.990959	-117.186136	Rancho Valencia Resort Spa	92067	CA	2013-11-14T00:00:00Z	5	Our experience at Rancho Valencia was absolute...	Best romantic vacation ever!!!!	
1	Rancho Santa Fe	US	32.990959	-117.186136	Rancho Valencia Resort Spa	92067	CA	2014-07-06T00:00:00Z	5	Amazing place. Everyone was extremely warm and...	Sweet sweet serenity	
2	Rancho Santa Fe	US	32.990959	-117.186136	Rancho Valencia Resort Spa	92067	CA	2015-01-02T00:00:00Z	5	We booked a 3 night stay at Rancho Valencia to...	Amazing Property and Experience	
3	Hanover	US	39.155929	-76.716341	Aloft Arundel Mills	21076	MD	2016-05-15T00:00:00Z	2	Currently in bed writing this for the past hr ...	Never again...beware, if you want sleep.	
4	Hanover	US	39.155929	-76.716341	Aloft Arundel Mills	21076	MD	2016-07-09T00:00:00Z	5	I live in Md and the Aloft is my Home away fro...	ALWAYS GREAT STAY...	

In [13]:
```
def combined_features(row):
    return row['reviews.title'] , ' ' , row['reviews.text']

df['reviews'] = df.apply(combined_features,axis=1)
df.head()
```

Out[13]:

	city	country	latitude	longitude	name	postalCode	province	reviews.date	reviews.rating	reviews.text	reviews.title	sentimen
0	Rancho Santa Fe	US	32.990959	-117.186136	Rancho Valencia Resort Spa	92067	CA	2013-11-14T00:00:00Z	5	Our experience at Rancho Valencia was absolute...	Best romantic vacation ever!!!!	
1	Rancho Santa Fe	US	32.990959	-117.186136	Rancho Valencia Resort Spa	92067	CA	2014-07-06T00:00:00Z	5	Amazing place. Everyone was extremely warm and...	Sweet sweet serenity	
2	Rancho Santa Fe	US	32.990959	-117.186136	Rancho Valencia Resort Spa	92067	CA	2015-01-02T00:00:00Z	5	We booked a 3 night stay at Rancho Valencia to...	Amazing Property and Experience	
3	Hanover	US	39.155929	-76.716341	Aloft Arundel Mills	21076	MD	2016-05-15T00:00:00Z	2	Currently in bed writing this for the past hr ...	Never again...beware, if you want sleep.	
4	Hanover	US	39.155929	-76.716341	Aloft Arundel Mills	21076	MD	2016-07-09T00:00:00Z	5	I live in Md and the Aloft is my Home away fro...	ALWAYS GREAT STAY...	

In [14]:
```
# import libraries
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
import re
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\nekei\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [15]:
```
#converting all letters to lower case
df['reviews'] = df['reviews'].apply(lambda x : " ".join(x.lower()for x in str(x).split()))

#remove non alpha characters
df['reviews'] = df['reviews'].apply(lambda x : " ".join([re.sub('[^A-Za-z]+','',x)for x in nltk.word_tokenize(x

#remove extra spaces
df['reviews'] = df['reviews'].apply(lambda x : re.sub(' +', ' ',x))

#lemmatization (takes away suffixes to bring out root word)
lemmatizer = WordNetLemmatizer()
df['reviews'] = df['reviews'].apply(lambda x: " ".join([lemmatizer.lemmatize(w) for w in nltk.word_tokenize(x)]
```

In [16]:
```
#remove stop words
from nltk.corpus import stopwords
stop = stopwords.words('english')
df['reviews'] = df['reviews'].apply(lambda x: " ".join([x for x in x.split() if x not in stop]))
```

In [17]:
```
stop
```

Out[17]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
 'do',
 'does',
 'did',
 'doing',
 'a',
 'an',
 'the',
 'and',
 'but',
 'if',
 'or',
 'because',
 'as',
 'until',
 'while',
 'of',
 'at',
 'by',
 'for',
 'with',
 'about',
 'against',
 'between',
 'into',
 'through',
 'during',
 'before',
 'after',
 'above',
 'below',
 'to',
 'from',
 'up',
 'down',
 'in',
 'out',
 'on',
 'off',
 'over',
 'under',
 'again',
 'further',
 'then',
 'once',
 'here',
 'there',
 'when',
 'where',
 'why',
 'how',
 'all',
 'any',
 'both',
 'each',
 'few',
 'more',
 'most',
 'other',
 'some',
 'such',
 'no',
 'nor',
 'not',
 'only',
 'own',
 'same',
 'so',
 'than',
 'too',
 'very',
 's',
 't',
 'can',
 'will',
 'just',
 'don',
 "don't",
 'should',
 "should've",
 'now',
 'd',
 'll',
 'm',
 'o',
 're',
 've',
 'y',
 'ain',
 'aren',
 "aren't",
 'couldn',
 "couldn't",
 'didn',
 "didn't",
 'doesn',
 "doesn't",
 'hadn',
 "hadn't",
 'hasn',
 "hasn't",
 'haven',
 "haven't",
 'isn',
 "isn't",
 'ma',
 'mightn',
 "mightn't",
 'mustn',
 "mustn't",
 'needn',
 "needn't",
 'shan',
 "shan't",
 'shouldn',
 "shouldn't",
 'wasn',
 "wasn't",
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```

In [18]:
```
df['reviews']
```

Out[18]:
```
0       best romantic vacation ever experience rancho ...
1       sweet sweet serenity amazing place everyone wa...
2       amazing property experience booked night stay ...
3       never beware want sleep currently bed writing ...
4       always great stay live md aloft home away home...
                              ...
9994    gorgeous outlook beachwatersun small town stat...
9996    clean comfortable quiet live close needed stay...
9997    passing rolled laid head woke continental brea...
9998    terrible absolutely terrible wa told wa given ...
9999    polde filthy outdated noisy neighbour wa worst...
Name: reviews, Length: 8535, dtype: object
```

In [19]:
```
#import libraries
from nltk.stem.porter import PorterStemmer

#Stemming words
def stem_words(text):
    porter = PorterStemmer()
    stemmed = ' '.join([porter.stem(word) for word in text.split()])
    return stemmed

#Using porter stemmer to stem words
df['reviews'] = df['reviews'].apply(lambda x : stem_words(x))

#Tokenize
def tokenization (text) :
    tokens = re.split('W+',text)
    return tokens

df['reviews'] = df['reviews'].apply(lambda x : tokenization(x))

df['reviews'] = df['reviews'].astype(str)
```

In [20]:
```
df['reviews']
```

Out[20]:
```
0       ['best romant vacat ever experi rancho valenci...
1       ['sweet sweet seren amaz place everyon wa extr...
2       ['amaz properti experi book night stay rancho ...
3       ['never bewar want sleep current bed write pas...
4       ['alway great stay live md aloft home away hom...
                              ...
9994    ['gorgeou outlook beachwatersun small town sta...
9996    ['clean comfort quiet live close need stay som...
9997    ['pass roll laid head woke continent breakfast...
9998    ['terribl absolut terribl wa told wa given non...
9999    ['pold filthi outdat noisi neighbour wa worst ...
Name: reviews, Length: 8535, dtype: object
```

I would like to add another part in here to focus more so on the reviews and graphs of the reviews since I have cleaned the text now because that is my main focus at the point for my project.

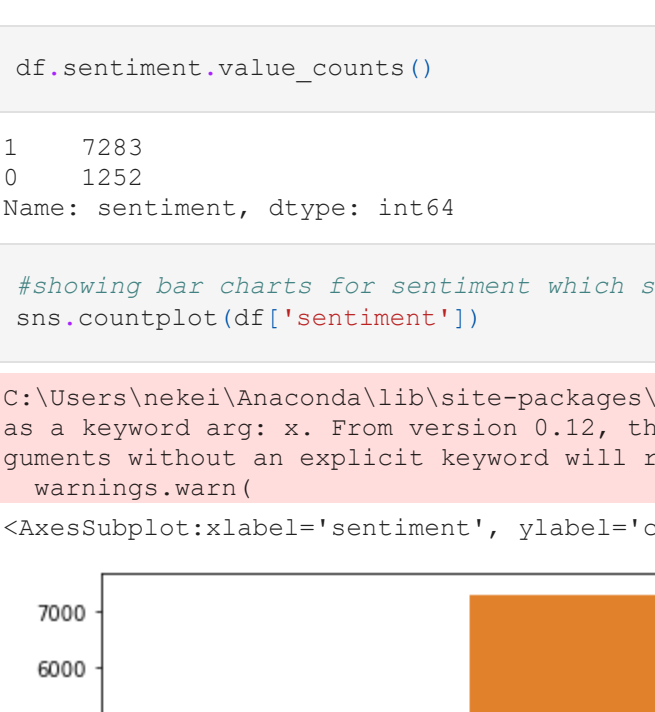
In [21]:
```
df['reviews'][10]
```

Out[21]:
```
"['love stay staff friendli help room larg nice furnish feel new first stay actual opul requir free breakfast w
a good suffici choic includ fresh fruit salad egg bacon bagel pastri wa great locat purpos']"
```

In [22]:
```
#creating bar charts for the number of each obersevation including three since that is neutral
sns.countplot(df['reviews.rating'])
```

```
C:\Users\nekei\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[22]: <AxesSubplot:xlabel='reviews.rating', ylabel='count'>



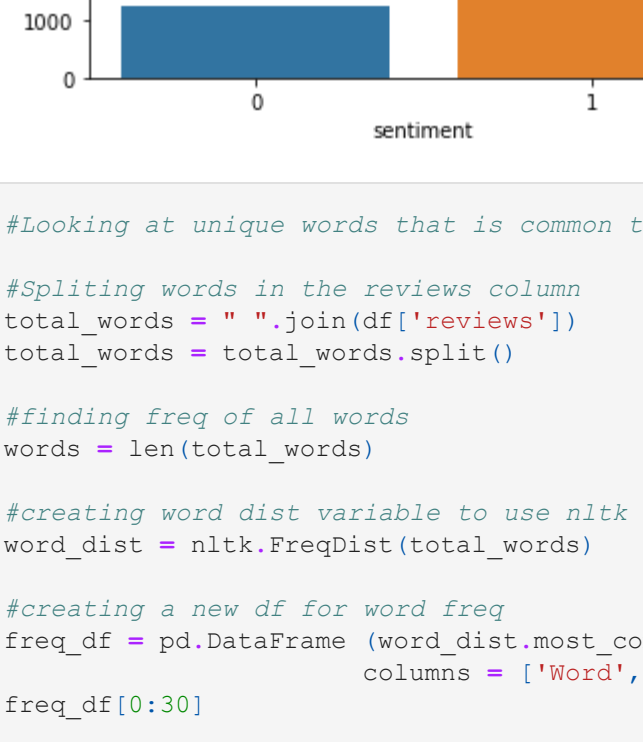
In [23]:
```
df.sentiment.value_counts()
```

Out[23]:
```
1    7283
0    1252
Name: sentiment, dtype: int64
```

In [24]:
```
#showing bar charts for sentiment which seperated graph into good (1) or bad (2)
sns.countplot(df['sentiment'])
```

```
C:\Users\nekei\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[24]: <AxesSubplot:xlabel='sentiment', ylabel='count'>



In [25]:
```
#Looking at unique words that is common throughout and its freq

#Spliting words in the reviews column
total_words = " ".join(df['reviews'])
total_words = total_words.split()

#finding freq of all words
words = len(total_words)

#creating word dist variable to use nltk to show freq dist
word_dist = nltk.FreqDist(total_words)

#creating a new df for word freq
freq_df = pd.DataFrame (word_dist.most_common(words),
                        columns = ['Word','Freq'])
freq_df[0:30]
```

Out[25]:

	Word	Freq
0	wa	13677
1	room	7552
2	hotel	7181
3	stay	6054
4	staff	3887
5	great	3532
6	clean	3256
7	good	2938
8	locat	2661
9	nice	2577
10	breakfast	2514
11	friendli	2005
12	place	1896
13	nt	1836
14	would	1799
15	comfort	1768
16	bed	1650
17	night	1575
18	time	1558
19	servic	1507
20	u	1491
21	help	1465
22	one	1308
23	['great	1293
24	area	1193
25	get	1089
26	like	1087
27	well	1069
28	desk	1041
29	pool	1030

In [26]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation

# Feature Engineering: Extracting Topics using Latent Dirichlet Allocation (LDA)
vectorizer = CountVectorizer(max_features=1000, max_df=0.95)
X = vectorizer.fit_transform(df['reviews'])

lda = LatentDirichletAllocation(n_components=5, random_state=42)
topics = lda.fit_transform(X)

df['topic'] = topics.argmax(axis=1)
```

In [64]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Sentiment Analysis using VADER
nltk.download('vader_lexicon')
analyzer = nltk.sentiment.SentimentIntensityAnalyzer()

def get_sentiment_score(text):
    return analyzer.polarity_scores(text)['compound']

df['sentiment_scores'] = df['reviews'].apply(get_sentiment_score)

df['rounded_sentiment'] = df['sentiment_scores'].apply(round)

df.loc[df['rounded_sentiment'] == -1, 'rounded_sentiment'] = 0
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\nekei\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [65]:
```
hotel_sentiment = df.groupby('name')['rounded_sentiment'].mean()
hotel_topics = df.groupby(['name', 'topic'])['topic'].count().unstack(fill_value=0)

hotel_sentiment
```

Out[65]:
```
name
1906 Lodge At Coronado Beach             1.000000
250 Main Hotel                           0.500000
AC Hotel Chicago Downtown                1.000000
AC Hotel Miami Beach                     1.000000
AC Hotel by Marriott Boston Downtown     0.000000
                                           ...
XV Beacon                                1.000000
Yakutat Lodge                            1.000000
dana hotel and spa                       0.657895
hampton inn Springfield southeast        0.500000
hotel le bleu                            0.666667
Name: rounded_sentiment, Length: 1576, dtype: float64
```

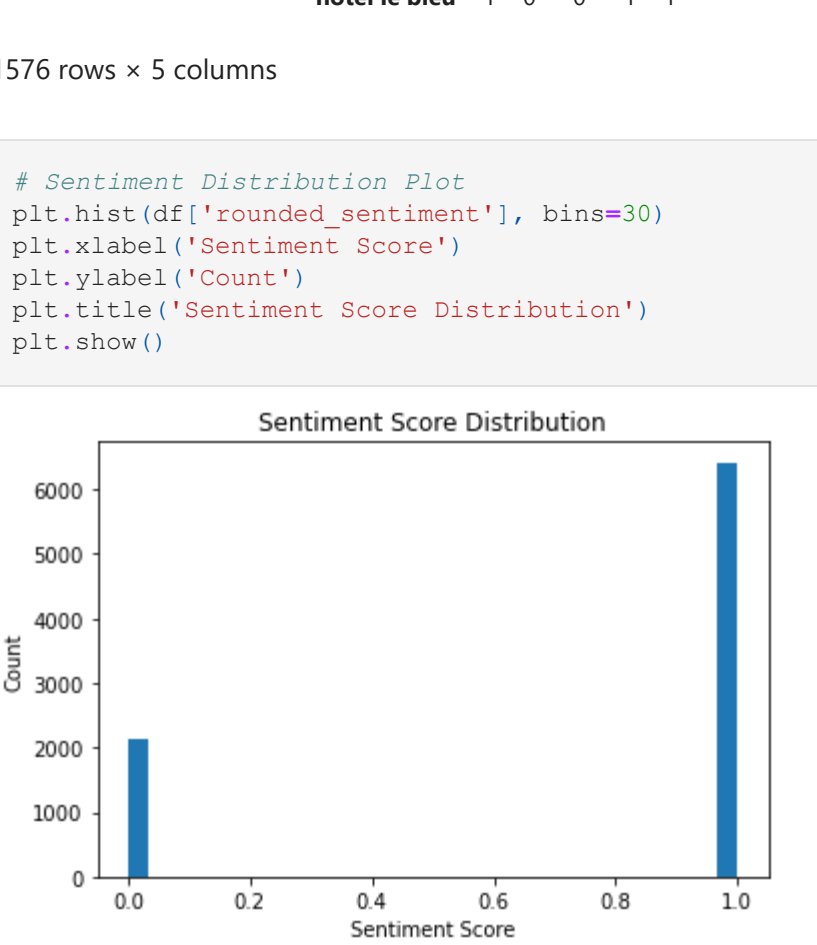
In [66]:
```
hotel_topics
```

Out[66]:

topic	0	1	2	3	4
name					
1906 Lodge At Coronado Beach	0	0	1	0	0
250 Main Hotel	0	2	0	0	0
AC Hotel Chicago Downtown	0	3	0	4	1
AC Hotel Miami Beach	0	0	1	0	0
AC Hotel by Marriott Boston Downtown	0	0	1	0	0
...	...	...	...	...	...
XV Beacon	0	0	1	0	0
Yakutat Lodge	0	0	1	0	0
dana hotel and spa	4	2	12	14	6
hampton inn Springfield southeast	0	0	1	1	0
hotel le bleu	1	0	0	1	1

1576 rows × 5 columns

In [67]:
```
# Sentiment Distribution Plot
plt.hist(df['rounded_sentiment'], bins=30)
plt.xlabel('Sentiment Score')
plt.ylabel('Count')
plt.title('Sentiment Score Distribution')
plt.show()
```



Milestone 3:

In [31]:
```
#split data into training and testing sets
from sklearn.model_selection import train_test_split

x = df['reviews']
y = df['sentiment']

x_train,x_test,y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 30)

print("Train: ",x_train.shape, y_train.shape,"Test: ", (x_test.shape,y_test.shape))
```

```
Train:  (6401,) (6401,) Test:  ((2134,), (2134,))
```

In [32]:
```python
#import TfIdfVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
tv = TfidfVectorizer()

#fit training and test data
tf_x_train = tv.fit_transform(x_train)
tf_x_test = tv.transform(x_test)
```

In [33]:
```python
#Using SVM for classification
from sklearn.svm import LinearSVC
svc = LinearSVC(random_state=0)

#fit training data into model
svc.fit(tf_x_train,y_train)

#Predicting test data
y_test_pred = svc.predict(tf_x_test)
```

In [34]:
```python
#analyze the results
from sklearn.metrics import classification_report

report=classification_report(y_test, y_test_pred,output_dict=True)
report
```

Out[34]:
```
{'0': {'precision': 0.8415492957746479,
  'recall': 0.7836065573770492,
  'f1-score': 0.8115449915110357,
  'support': 305},
 '1': {'precision': 0.9643243243243244,
  'recall': 0.9753963914707491,
  'f1-score': 0.9698287578146236,
  'support': 1829},
 'accuracy': 0.9479850046860356,
 'macro avg': {'precision': 0.9029368100494861,
  'recall': 0.8795014744238991,
  'f1-score': 0.8906868746628296,
  'support': 2134},
 'weighted avg': {'precision': 0.9467768155578523,
  'recall': 0.9479850046860356,
  'f1-score': 0.9472061951517398,
  'support': 2134}}
```

I got a 95.% accuracy with SVM. My model did a pretty fair job at predicting both instances , we see that there are more positive reviews overall and that's probably why my precision was higher than the precision of the negative reviews because it had more information to test. I am now going to try Logistic Regression modeling to see if it improves.

In [35]:
```python
#LogReg
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression(max_iter=1000,solver='saga')
```

In [36]:
```python
#Fitting logistic regression model
log_reg.fit(tf_x_train,y_train)
```

Out[36]:
```
▾ LogisticRegression
LogisticRegression(max_iter=1000, solver='saga')
```

In [37]:
```python
#predicting the output of data
y_test_pred=log_reg.predict(tf_x_test)
```

In [38]:
```python
#anaylzing with classification report
from sklearn.metrics import classification_report
report=classification_report(y_test, y_test_pred,output_dict=True)
report
```

Out[38]:
```
{'0': {'precision': 0.9074074074074074,
  'recall': 0.6426229508196721,
  'f1-score': 0.7523992322456814,
  'support': 305},
 '1': {'precision': 0.943169968717414,
  'recall': 0.9890650628758885,
  'f1-score': 0.9655724579663731,
  'support': 1829},
 'accuracy': 0.9395501405810684,
 'macro avg': {'precision': 0.9252886880624107,
  'recall': 0.8158440068477804,
  'f1-score': 0.8589858451060273,
  'support': 2134},
 'weighted avg': {'precision': 0.9380586373211854,
  'recall': 0.9395501405810684,
  'f1-score': 0.9351048694730221,
  'support': 2134}}
```

I got a 93.6% accuracy with LogReg which is better than the SVM. Also it did a better job with looking at the negative class instances and identifying the precision of it. My model's recall was at 60% for the first instances which shows me that the number of actual positive instances that were identified which is lower than I would have like.

Now I am going to repeat those steps with another dataframe that is similar to the orginal just at a later date to see how well my model is. The dataset is "hotel_reviews_june19th.csv" , which is 1,400 hotels and about 10,000 reviews.

In [39]:
```python
df2 = pd.read_csv('hotel_reviews_June19th.csv')
```

In [40]:
```python
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 26 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10000 non-null  object 
 1   dateAdded             10000 non-null  object 
 2   dateUpdated           10000 non-null  object 
 3   address               10000 non-null  object 
 4   categories            10000 non-null  object 
 5   primaryCategories     10000 non-null  object 
 6   city                  10000 non-null  object 
 7   country               10000 non-null  object 
 8   keys                  10000 non-null  object 
 9   latitude              10000 non-null  float64
 10  longitude             10000 non-null  float64
 11  name                  10000 non-null  object 
 12  postalCode            10000 non-null  object 
 13  province              10000 non-null  object 
 14  reviews.date          10000 non-null  object 
 15  reviews.dateAdded     0 non-null      float64
 16  reviews.dateSeen      10000 non-null  object 
 17  reviews.rating        10000 non-null  int64  
 18  reviews.sourceURLs    10000 non-null  object 
 19  reviews.text          10000 non-null  object 
 20  reviews.title         9999 non-null   object 
 21  reviews.userCity      10000 non-null  object 
 22  reviews.userProvince  9998 non-null   object 
 23  reviews.username      10000 non-null  object 
 24  sourceURLs            10000 non-null  object 
 25  websites              10000 non-null  object 
dtypes: float64(3), int64(1), object(22)
memory usage: 2.0+ MB
```

In [41]:
```python
#dropping unnecessary columns because these are not essential in looking at the reviews of the data.
df2 = df2.drop(['id', 'reviews.dateSeen','reviews.dateAdded', 'dateAdded', 'dateUpdated', 'reviews.sourceURLs',
```

In [42]:
```python
for i in range(0,len(df2)-1):
    if type(df2.iloc[i]['reviews.text']) !=str:
        df2.iloc[i]['reviews.text']=str(df.iloc[i])
['reviews.text']
```

Out[42]: ['reviews.text']

In [43]:
```python
#Change ratings column to a int .
df2['reviews.rating'] = df2['reviews.rating'].astype(int)

#get rid of rating of 3
df2 = df2[df2['reviews.rating']!=3]

#Seperate positive (4 or 5) and negative (1,2) and no middle
def sentiment (n):
    return 1 if n >=4 else 0

df2['sentiment'] = df2['reviews.rating'].apply(sentiment)
df2.head()
```

Out[43]:

	city	country	latitude	longitude	name	postalCode	province	reviews.date	reviews.rating	reviews.text	reviews.title	sentime
1	Carmel by the Sea	US	36.557220	-121.921940	Best Western Carmel's Town House Lodge	93921	CA	2016-04-02T00:00:00Z	4	We stayed in the king suite with the separatio...	Clean rooms at solid rates in the heart of Carmel	
3	Carmel by the Sea	US	36.557220	-121.921940	Best Western Carmel's Town House Lodge	93921	CA	2016-08-22T00:00:00Z	5	Not cheap but excellent location. Price is som...	Very good	
4	Carmel by the Sea	US	36.557220	-121.921940	Best Western Carmel's Town House Lodge	93921	CA	2016-03-21T00:00:00Z	2	If you get the room that they advertised on th...	Low chance to come back here	
5	Lexington	US	38.047014	-84.497742	21c Museum Hotel Lexington	40507	KY	2016-03-15T00:00:00Z	4	This is such a fun, lovely hotel. The attentio...	Loved staying here	
6	Lexington	US	38.047014	-84.497742	21c Museum Hotel Lexington	40507	KY	2016-04-18T00:00:00Z	1	We recently stayed at this hotel on a trip to ...	Does not live up to its reputation	

In [44]:
```python
df2['reviews'] = df2.apply(combined_features,axis =1 )
df2.head()
```

Out[44]:

	city	country	latitude	longitude	name	postalCode	province	reviews.date	reviews.rating	reviews.text	reviews.title	sentime
1	Carmel by the Sea	US	36.557220	-121.921940	Best Western Carmel's Town House Lodge	93921	CA	2016-04-02T00:00:00Z	4	We stayed in the king suite with the separatio...	Clean rooms at solid rates in the heart of Carmel	
3	Carmel by the Sea	US	36.557220	-121.921940	Best Western Carmel's Town House Lodge	93921	CA	2016-08-22T00:00:00Z	5	Not cheap but excellent location. Price is som...	Very good	
4	Carmel by the Sea	US	36.557220	-121.921940	Best Western Carmel's Town House Lodge	93921	CA	2016-03-21T00:00:00Z	2	If you get the room that they advertised on th...	Low chance to come back here	
5	Lexington	US	38.047014	-84.497742	21c Museum Hotel Lexington	40507	KY	2016-03-15T00:00:00Z	4	This is such a fun, lovely hotel. The attentio...	Loved staying here	
6	Lexington	US	38.047014	-84.497742	21c Museum Hotel Lexington	40507	KY	2016-04-18T00:00:00Z	1	We recently stayed at this hotel on a trip to ...	Does not live up to its reputation	

In [45]:
```python
#converting all letters to lower case
df2['reviews'] = df2['reviews'].apply(lambda x : " ".join(x.lower()for x in str(x).split()))

#remove non alpha characters
df2['reviews'] = df2['reviews'].apply(lambda x : " ".join([re.sub('[^A-Za-z]+','',x)for x in nltk.word_tokenize

#remove extra spaces
df2['reviews'] = df2['reviews'].apply(lambda x : re.sub(' +', ' ',x))

#lemmatization (takes away suffixes to bring out root word)
df2['reviews'] = df2['reviews'].apply(lambda x: " ".join([lemmatizer.lemmatize(w) for w in nltk.word_tokenize(
```

In [46]:
```python
#remove stop words
from nltk.corpus import stopwords
stop = stopwords.words('english')
df2['reviews'] = df2['reviews'].apply(lambda x: " ".join([x for x in x.split() if x not in stop]))
```

In [47]:
```python
#Using porter stemmer to stem words
df2['reviews'] = df2['reviews'].apply(lambda x : stem_words(x))

#Tokenize
df2['reviews'] = df2['reviews'].apply(lambda x : tokenization(x))

df2['reviews'] = df2['reviews'].astype(str)
```

In [48]:
```python
df2['reviews']
```

Out[48]:
```
1       ['clean room solid rate heart carmel stay king...
3       ['good cheap excel locat price somewhat standa...
4       ['low chanc come back get room advertis websit...
5       ['love stay fun love hotel attent detail impre...
6       ['doe live reput recent stay hotel trip lexing...
                              ...                        
9995    ['accommod friendli staff friend took trip ham...
9996    ['comfort friendli clean profession check depa...
9997    ['great locat hampton locat quiet street acros...
9998    ['great atmospher awesom wing favorit wa garli...
9999    ['health care appoint clean facil freeway staf...
Name: reviews, Length: 8618, dtype: object
```

In [49]:
```python
x2 = df2['reviews']
y2 = df2['sentiment']

x2_train,x2_test,y2_train, y2_test = train_test_split(x2, y2, test_size = 0.25, random_state = 30)

print("Train: ",x2_train.shape, y2_train.shape,"Test: ", (x2_test.shape,y2_test.shape))
```

```
Train:  (6463,) (6463,) Test:  ((2155,), (2155,))
```

In [50]:
```python
#fit training and test data
tf_x2_train = tv.fit_transform(x2_train)
tf_x2_test = tv.transform(x2_test)
```

In [51]:
```python
#fit training data into model
svc.fit(tf_x2_train,y2_train)

#Predicting test data
y2_test_pred = svc.predict(tf_x2_test)
```

In [52]:
```python
report2=classification_report(y2_test, y2_test_pred,output_dict=True)
report2
```

Out[52]:
```
{'0': {'precision': 0.9095022624434389,
  'recall': 0.7556390977443609,
  'f1-score': 0.8254620123203285,
  'support': 266},
 '1': {'precision': 0.9663908996897621,
  'recall': 0.9894123875066173,
  'f1-score': 0.9777661522364635,
  'support': 1889},
 'accuracy': 0.9605568445475638,
 'macro avg': {'precision': 0.9379465810666006,
  'recall': 0.8725257426254891,
  'f1-score': 0.901614082278396,
  'support': 2155},
 'weighted avg': {'precision': 0.9593689147674781,
  'recall': 0.9605568445475638,
  'f1-score': 0.9589666621122445,
  'support': 2155}}
```

My accuracy was 96.15% which is better than what I did on my first data frame but also I see that my recall increased signfically here from the previous 60%.

In [53]:
```python
df2['reviews']
```

Out[53]:
```
1       ['clean room solid rate heart carmel stay king...
3       ['good cheap excel locat price somewhat standa...
4       ['low chanc come back get room advertis websit...
5       ['love stay fun love hotel attent detail impre...
6       ['doe live reput recent stay hotel trip lexing...
                              ...                        
9995    ['accommod friendli staff friend took trip ham...
9996    ['comfort friendli clean profession check depa...
9997    ['great locat hampton locat quiet street acros...
9998    ['great atmospher awesom wing favorit wa garli...
9999    ['health care appoint clean facil freeway staf...
Name: reviews, Length: 8618, dtype: object
```

In [54]:
```python
#Example Review 1

print("Example Review:" , df2['reviews'][1])

sentiment = "Positive" if y2_test.iloc[1] == 1 else "Negative"
predicted_sentiment = "Positive" if y2_test_pred[1] == 1 else "Negative"

# Print the actual and predicted sentiments
print("Actual Sentiment:", sentiment)
print("Predicted Sentiment:", predicted_sentiment)
```

```
Example Review: ['clean room solid rate heart carmel stay king suit separ bedroom live space sofa bed wa nt goo
d back discomfort day left three night stay room clean king bed comfort hotel locat within walk distanc place w
ant']
Actual Sentiment: Positive
Predicted Sentiment: Positive
```

In [55]:
```python
#Example Review 2

print("Example Review:" , df2['reviews'][50])

sentiment = "Positive" if y2_test.iloc[50] == 1 else "Negative"
predicted_sentiment = "Positive" if y2_test_pred[50] == 1 else "Negative"

# Print the actual and predicted sentiments
print("Actual Sentiment:", sentiment)
print("Predicted Sentiment:", predicted_sentiment)
```

```
Example Review: ['random fee stay night hotel wa beauti staff pleasant inquir extra charg card one knew email t
wice call intern wa honeymoon respons frustrat know much wa charg without reason enteract staff sorri hear ha h
appen stay u want make sure guest best possibl stay see drop ball came ask reach briandaviswhitneyhotelcom call
directli get resolv right away thank hope hear soon']
Actual Sentiment: Positive
Predicted Sentiment: Positive
```

In [56]:
```python
#Example Review 3

print("Example Review:" , df2['reviews'][107])

sentiment = "Positive" if y2_test.iloc[107] == 1 else "Negative"
predicted_sentiment = "Positive" if y2_test_pred[107] == 1 else "Negative"

# Print the actual and predicted sentiments
print("Actual Sentiment:", sentiment)
print("Predicted Sentiment:", predicted_sentiment)
```

```
Example Review: ['disappoint area wa busi freeway describ stripper neighborhood need earli checkin arrang month
ago nt get abl check notic wet section carpet thought mayb clean someth next morn room satur went front desk sa
id probabl ac problem said problem fix left hour later return problem fix water travel went front desk said som
eth like ca nt fix day expect odd expect fix least suck water ca nt hmmmmm get new room said fix time limit way
soon said send someon look twenti minut bar nice bar staff went room mainten fix minut place towel floor stay l
eft thank feedback apolog unabl fix air condit issu immedi sever interact earli day offer room move sever time
could get resourc fix leak air condit sorri staff spoke empathet offer altern accommod tri honor earli check re
quest depend departur pattern quickli clean room new arriv hope doe prohibit join u']
Actual Sentiment: Positive
Predicted Sentiment: Positive
```

In [57]:
```python
# I made this code below because I was thinkoing my model and prediction model had only positive values.
# this is proof that there is negative and positive values.

import numpy as np


# Check if any predicted sentiment is negative (0)
if any(sentiment == 0 for sentiment in y2_test_pred):
    print("At least one predicted sentiment is negative (0).")

    # Filter negative values using list comprehension
    negative_values = [sentiment for sentiment in y2_test_pred if sentiment == 0]

    # Calculate the sum of the negative values using NumPy
    sum_of_negatives = np.count_nonzero(y2_test_pred == 0)

    # Print the sum of the negative values
    print("Sum of negative values:", sum_of_negatives)
else:
    print("All predicted sentiments are positive (1).")
```

```
At least one predicted sentiment is negative (0).
Sum of negative values: 221
```