In [2]:

```
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\ukolv\anaconda3\lib\s
ite-packages (1.8.2.2)
Requirement already satisfied: numpy>=1.6.1 in c:\users\ukolv\anaconda3\li
b\site-packages (from wordcloud) (1.21.5)
Requirement already satisfied: matplotlib in c:\users\ukolv\anaconda3\lib
\site-packages (from wordcloud) (3.5.2)
Requirement already satisfied: pillow in c:\users\ukolv\anaconda3\lib\site
-packages (from wordcloud) (9.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\ukolv\anaconda3\li
b\site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ukolv\anacond
a3\lib\site-packages (from matplotlib->wordcloud) (1.4.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\ukolv\anac
onda3\lib\site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\ukolv\anaconda
3\lib\site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\ukolv\anacond
a3\lib\site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\ukolv\anaconda3
\lib\site-packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: six>=1.5 in c:\users\ukolv\anaconda3\lib\si
te-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
```

In [2]:

```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\ukolv\anaconda3\lib\s
ite-packages (6.4.4)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\ukolv\anacon
da3\lib\site-packages (from nbconvert) (0.4)
Requirement already satisfied: nbformat>=4.4 in c:\users\ukolv\anaconda3\l
ib\site-packages (from nbconvert) (5.5.0)
Requirement already satisfied: beautifulsoup4 in c:\users\ukolv\anaconda3
\lib\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\ukolv\anacond
a3\lib\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: traitlets>=5.0 in c:\users\ukolv\anaconda3
\lib\site-packages (from nbconvert) (5.1.1)
Requirement already satisfied: jupyter-core in c:\users\ukolv\anaconda3\li
b\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: jupyterlab-pygments in c:\users\ukolv\anaco
nda3\lib\site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: defusedxml in c:\users\ukolv\anaconda3\lib
\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=2.4 in c:\users\ukolv\anaconda3\lib
\site-packages (from nbconvert) (2.11.3)
Requirement already satisfied: bleach in c:\users\ukolv\anaconda3\lib\site
-packages (from nbconvert) (4.1.0)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\ukolv\an
aconda3\lib\site-packages (from nbconvert) (0.5.13)
Requirement already satisfied: pygments>=2.4.1 in c:\users\ukolv\anaconda3
\lib\site-packages (from nbconvert) (2.11.2)
Requirement already satisfied: testpath in c:\users\ukolv\anaconda3\lib\si
te-packages (from nbconvert) (0.6.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\ukolv\anac
onda3\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\ukolv\anaconda
3\lib\site-packages (from jinja2>=2.4->nbconvert) (2.0.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\ukolv\ana
conda3\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (7.3.4)
Requirement already satisfied: nest-asyncio in c:\users\ukolv\anaconda3\li
b\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.5.5)
Requirement already satisfied: jsonschema>=2.6 in c:\users\ukolv\anaconda3
\lib\site-packages (from nbformat>=4.4->nbconvert) (4.16.0)
Requirement already satisfied: fastjsonschema in c:\users\ukolv\anaconda3
\lib\site-packages (from nbformat>=4.4->nbconvert) (2.16.2)
Requirement already satisfied: soupsieve>1.2 in c:\users\ukolv\anaconda3\l
ib\site-packages (from beautifulsoup4->nbconvert) (2.3.1)
Requirement already satisfied: webencodings in c:\users\ukolv\anaconda3\li
b\site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: packaging in c:\users\ukolv\anaconda3\lib\s
ite-packages (from bleach->nbconvert) (21.3)
Requirement already satisfied: six>=1.9.0 in c:\users\ukolv\anaconda3\lib
\site-packages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: pywin32>=1.0 in c:\users\ukolv\anaconda3\li
b\site-packages (from jupyter-core->nbconvert) (302)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.1
4.0 in c:\users\ukolv\anaconda3\lib\site-packages (from jsonschema>=2.6->n
bformat>=4.4->nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\ukolv\anaconda3\l
ib\site-packages (from jsonschema>=2.6->nbformat>=4.4->nbconvert) (21.4.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\ukolv\an
aconda3\lib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.
5.0->nbconvert) (2.8.2)
Requirement already satisfied: tornado>=6.0 in c:\users\ukolv\anaconda3\li
b\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbcon
vert) (6.1)
```

```
Requirement already satisfied: pyzmq>=23.0 in c:\users\ukolv\anaconda3\lib
\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconv
ert) (23.2.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\ukolv
\anaconda3\lib\site-packages (from packaging->bleach->nbconvert) (3.0.9)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as mticker
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [4]:

```
originalData = pd.read_csv('OneDrive\Desktop\AB_NYC_2019.csv')
data = pd.read_csv('OneDrive\Desktop\AB_NYC_2019.csv').dropna(axis = 0, how = 'any')
display(data)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem
5	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hill
...	...	...	...	...	...	...
48782	36425863	Lovely Privet Bedroom with Privet Restroom	83554966	Rusaa	Manhattan	Upper East Side
48790	36427429	No.2 with queen size bed	257683179	H Ai	Queens	Flushing
48799	36438336	Seas The Moment	211644523	Ben	Staten Island	Great Kills
48805	36442252	1B-1B apartment near by Metro	273841667	Blaine	Bronx	Mott Haven
48852	36455809	Cozy Private Room in Bushwick, Brooklyn	74162901	Christine	Brooklyn	Bushwick

38821 rows × 16 columns

In [5]:

```
data['price'].mean()
```

Out[5]:

```
142.33252621004095
```

In [6]:

```
data['price'].std()
```

Out[6]:

196.99475591833985

In [7]:

```
data['latitude'].min()
```

Out[7]:

40.50641

In [8]:

```
data['latitude'].max()
```

Out[8]:

40.91306

In [9]:

```
data['longitude'].min()
```

Out[9]:

-74.24442

In [10]:

```
data['longitude'].max()
```

Out[10]:

-73.71299

In [11]:

```
data['price'].min()
```

Out[11]:

0

In [12]:

```
data['price'].max()
```

Out[12]:

10000

In [13]:

```
data['minimum_nights'].min()
```

Out[13]:

```
1
```

In [14]:

```
data['minimum_nights'].max()
```

Out[14]:

```
1250
```

In [15]:

```
data['number_of_reviews'].min()
```

Out[15]:

```
1
```

In [16]:

```
data['number_of_reviews'].max()
```

Out[16]:

```
629
```

In [17]:

```
data['last_review'].min()
```

Out[17]:

```
'2011-03-28'
```

In [18]:

```
data['last_review'].max()
```

Out[18]:

```
'2019-07-08'
```

In [19]:

```
data['reviews_per_month'].min()
```

Out[19]:

```
0.01
```

In [20]:

```
data['reviews_per_month'].max()
```

Out[20]:

```
58.5
```

In [21]:

```
data['calculated_host_listings_count'].min()
```

Out[21]:

```
1
```

In [22]:

```
data['calculated_host_listings_count'].max()
```

Out[22]:

```
327
```

In [23]:

```
data['availability_365'].min()
```

Out[23]:

```
0
```

In [24]:

```
data['availability_365'].max()
```

Out[24]:

```
365
```

1) First I obtained the uncleaned data set and printed its contents to have a priliminary understanding of the data. To begin cleaning, I dropped every row with at least one missing column using the pandas dropna method. Afterwards, I printed the minumum and maximum values of each numeric (non word) feature of the data set to examine whether the numbers were reasonable without further inspection. For instance, I inspected the latitude and longitutde to ensure that they were within valid range of NYC. There were a number of rows that had relatively unlikely values, but I think that having a couple strange values could prove to be useful and interesting. It would be a diservice to myself to remove artifacts thinking they were errors.

In [ ]:

```

```

In [25]:

```
all_neighbourhoods = data.groupby('neighbourhood').filter(lambda neighbourhood: 5 < len(r
```

In [26]:

```
neighbourhoods = all_neighbourhoods.groupby('neighbourhood').agg('mean')
```

In [27]:

```
sortedByPrice = neighbourhoods.sort_values(by='price')['price']
```

Bottom 5 Based on Price

In [28]:

```
sortedByPrice.head()
```

Out[28]:

```
neighbourhood
Tremont         49.900000
Hunts Point     51.812500
Bronxdale       51.875000
Soundview       52.846154
Corona          56.932203
Name: price, dtype: float64
```

Top 5 Based on Price

In [29]:

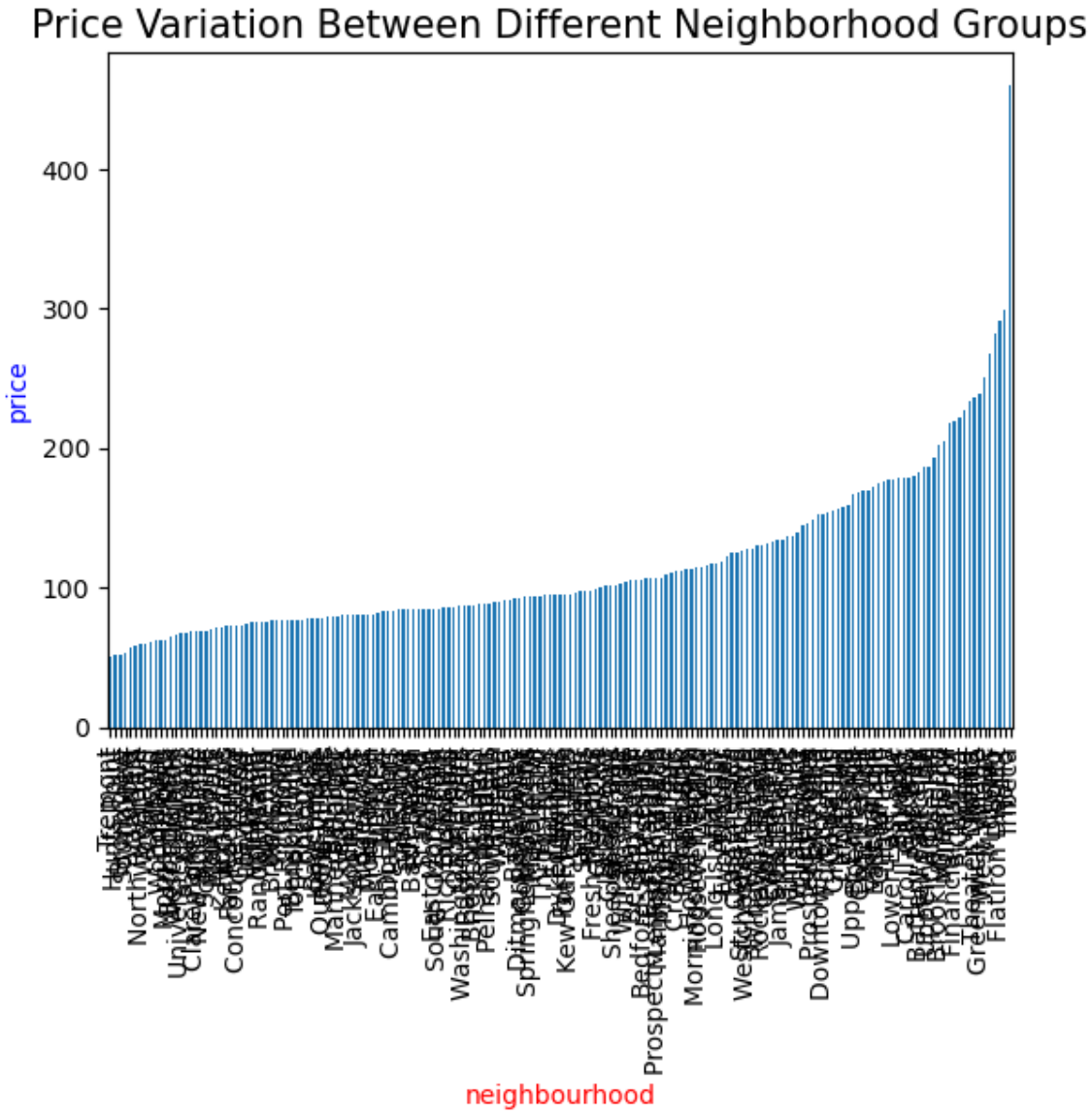
```
sortedByPrice.tail()
```

Out[29]:

```
neighbourhood
Midtown              267.583164
SoHo                 281.303136
Flatiron District    291.482759
NoHo                 298.451613
Tribeca              460.300000
Name: price, dtype: float64
```

In [30]:

```
plt.title('Price Variation Between Different Neighborhood Groups', fontsize = 15)
plt.xlabel('neighbourhoods', fontsize=10, color='red')
plt.ylabel('price', fontsize=10, color='blue')

data['price'] = data['price'].astype('float')
sortedByPrice.plot(kind = 'bar')
plt.show()
```



In [ ]:

```
```

In [31]:

```
reviewsVsAvailability = data['reviews_per_month'].corr(data['availability_365'])
print(reviewsVsAvailability)
```

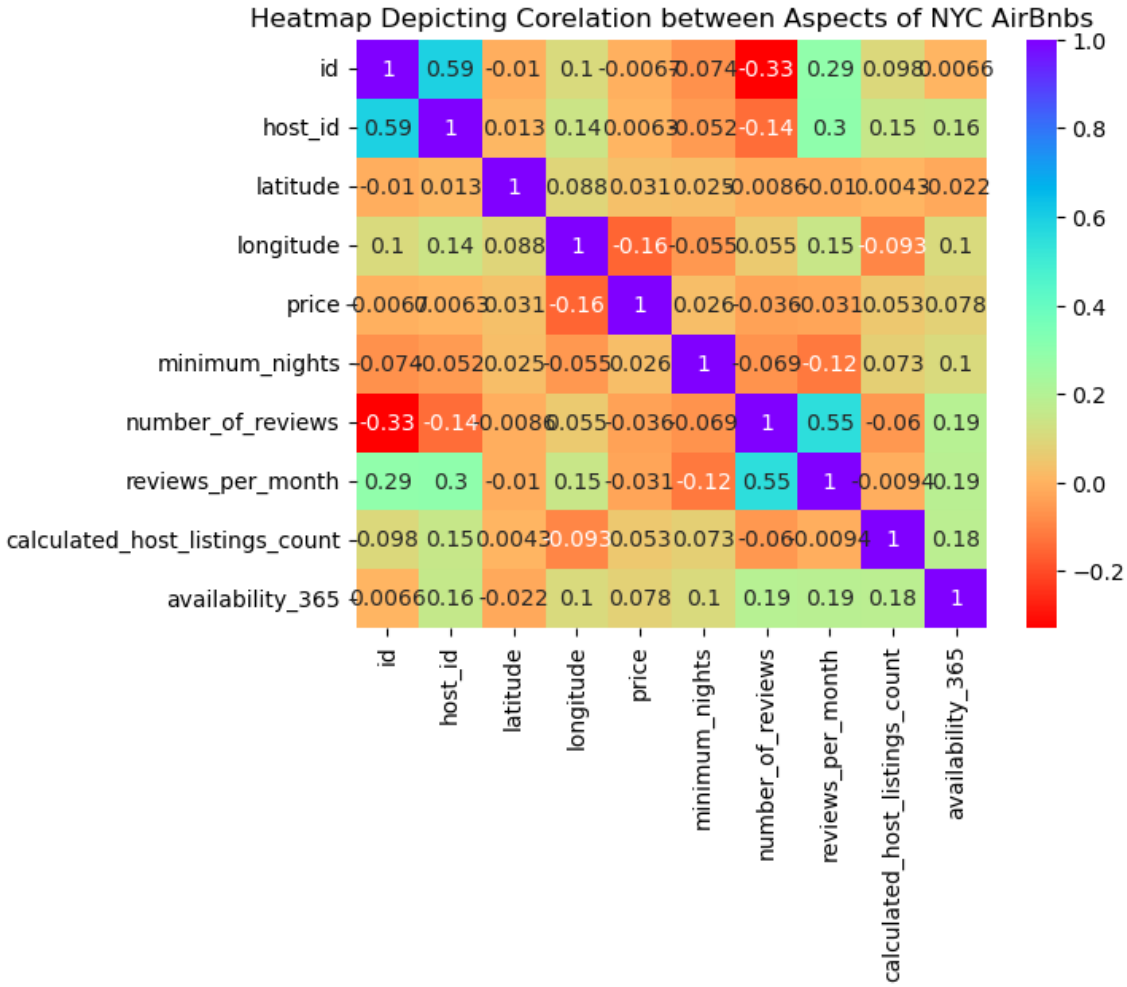
```
0.18589567903408583
```

In [83]:

```
dataplot = sns.heatmap(data.corr(method='pearson'), cmap="rainbow_r", annot=True)
plt.title('Heatmap Depicting Corelation between Aspects of NYC AirBnbs')
```

Out[83]:

```
Text(0.5, 1.0, 'Heatmap Depicting Corelation between Aspects of NYC AirBnb
s')
```



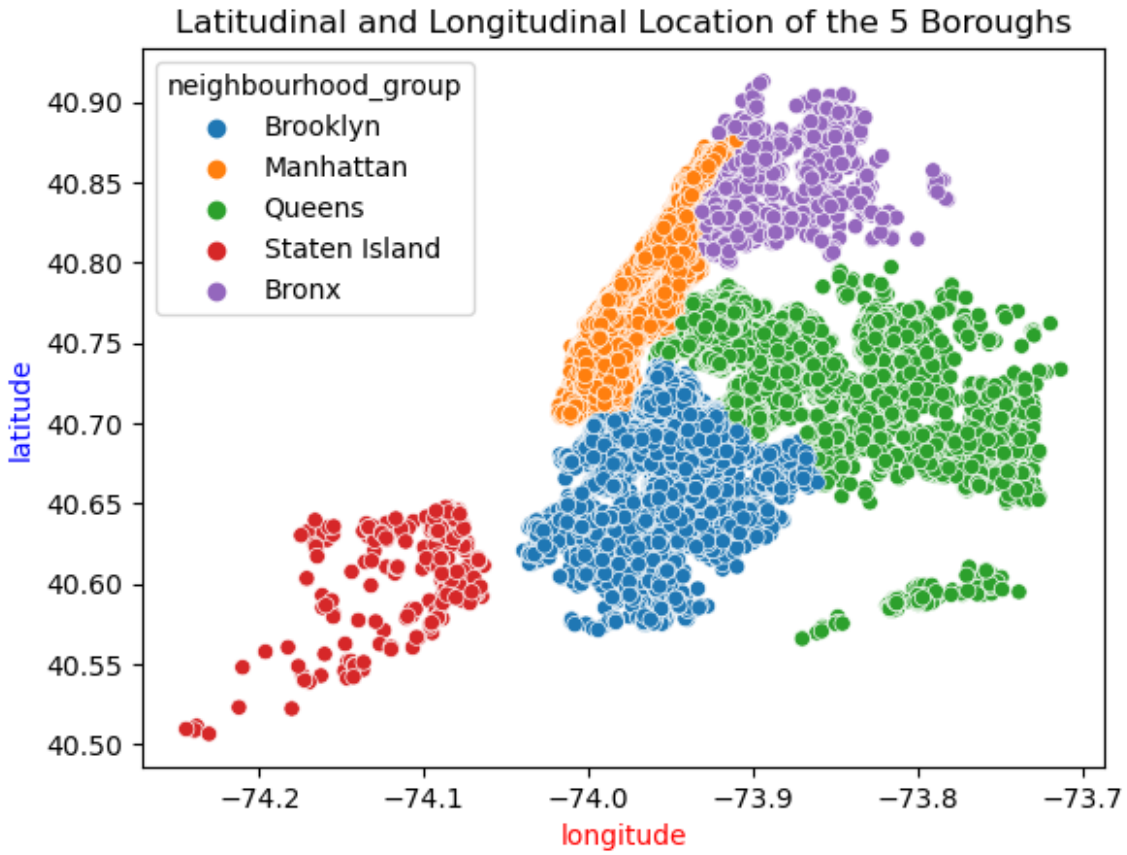
In [ ]:

In [138]:

```
sns.scatterplot(data=data, x='longitude', y='latitude', hue='neighbourhood_group')
plt.title('Latitudinal and Longitudinal Location of the 5 Boroughs')
plt.xlabel('longitude', fontsize=10, color='red')
plt.ylabel('latitude', fontsize=10, color='blue')
```

Out[138]:

```
Text(0, 0.5, 'latitude')
```

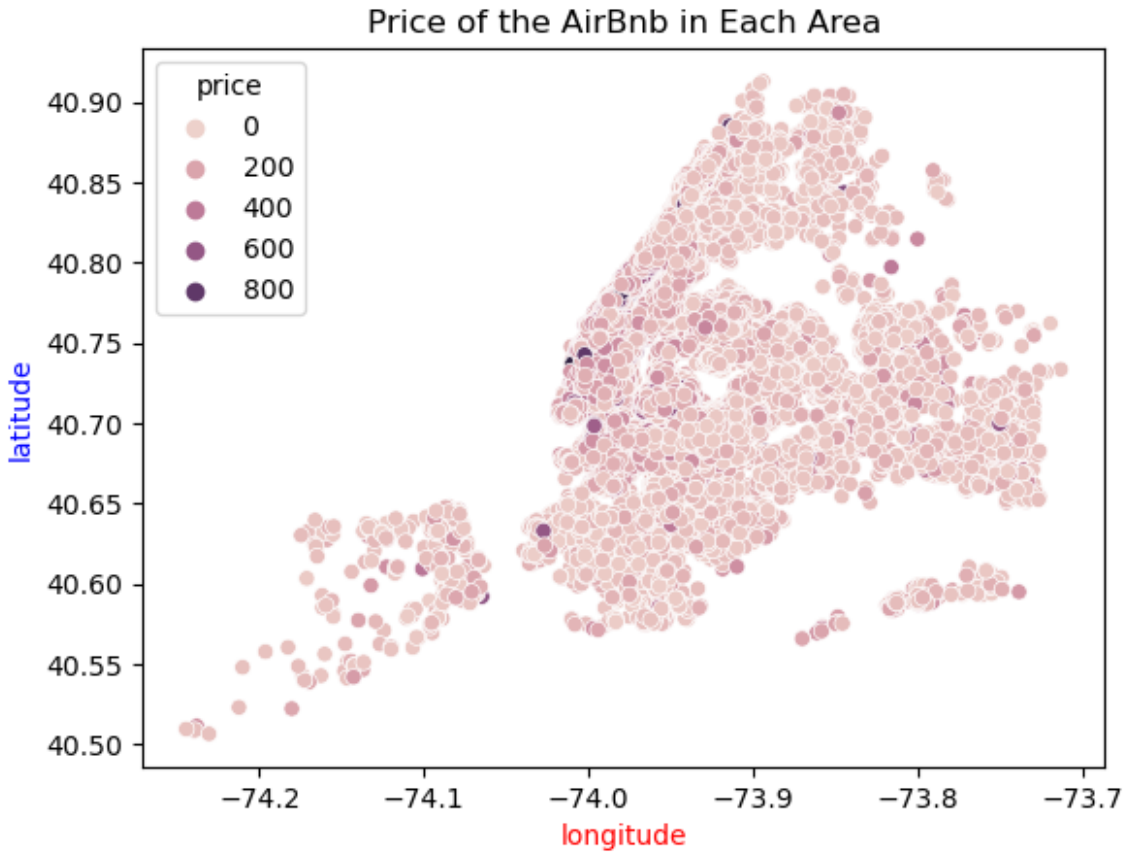


In [149]:

```
lessThan1000 = data[(data['price'] < 1000)]
sns.scatterplot(data=lessThan1000, x='longitude', y='latitude', hue='price')
plt.title('Price of the AirBnb in Each Area')
plt.xlabel('longitude', fontsize=10, color='red')
plt.ylabel('latitude', fontsize=10, color='blue')
```

Out[149]:

```
Text(0, 0.5, 'latitude')
```

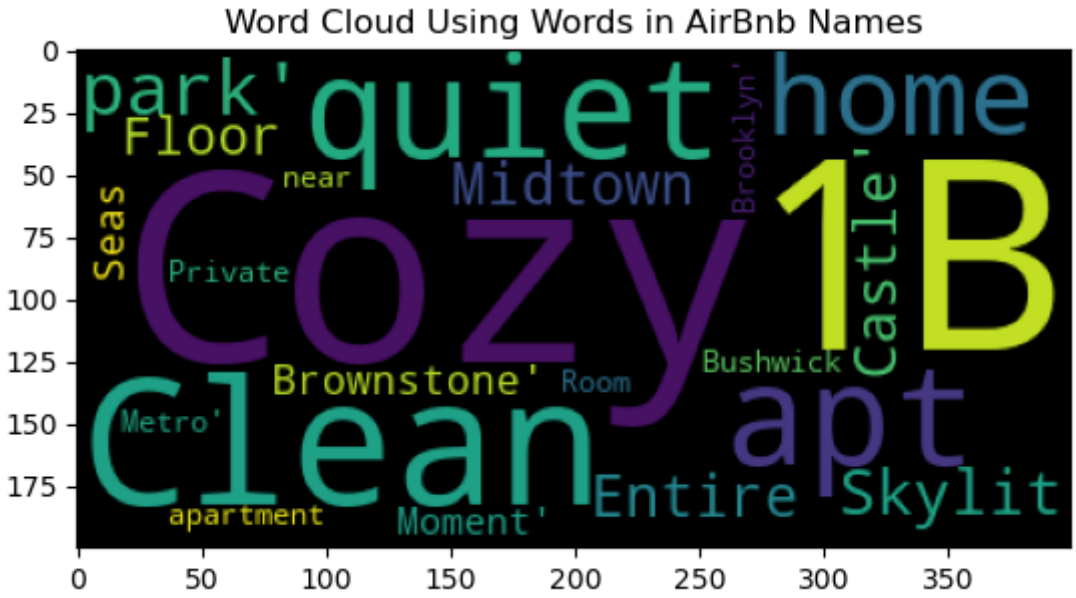


By looking at the two scatterplots side by side, it is evident that the majority of the most expensive Airbnbs are in the upperbrooklyn to lower manhattan area, as is demonstrated by the dark purple points. However, it is interesting to consider that there are both very expensive and very inexpensive Airbnbs spread across all five boroughs of NYC.

In [ ]:

In [101]:

```
words = data['name'].values
wordCloud = WordCloud().generate(str(words))
plt.title('Word Cloud Using Words in AirBnb Names')
plt.imshow(wordCloud)
plt.show()
```



Type *Markdown* and LaTeX: α^2

In [152]:

```
data.groupby('neighbourhood').agg('mean').sort_values(by='calculated_host_listings_count'
```

Out[152]:

```
neighbourhood
Holliswood                1.000000
Howland Hook              1.000000
Rossville                 1.000000
Mount Eden                1.000000
West Farms                1.000000
                           ...
Woodside                 12.100000
Eastchester              13.000000
Theater District         20.655405
Murray Hill              45.469453
Financial District      118.245580
Name: calculated_host_listings_count, Length: 218, dtype: float64
```

In [153]:

```
data.groupby('host_id').agg('count').sort_values(by='calculated_host_listings_count')['ca
```

Out[153]:

```
host_id
2438           1
47952362       1
47933513       1
47932600       1
47931146       1
             ...
7503643       49
137358866     51
16098958      61
61391963      79
219517861    207
Name: calculated_host_listings_count, Length: 30232, dtype: int64
```

In [140]:

```
busiest_host_ids = data.groupby('host_id').agg('mean').sort_values(by='calculated_host_li
```

In [141]:

```
busiest_host_listings = data.loc[data['host_id'].isin(busiest_host_ids)]
```

In [142]:

```
busiest_host_listings.groupby('neighbourhood').agg('count')
```

Out[142]:

neighbourhood	id	name	host_id	host_name	neighbourhood_group	latitude	longitude	rc
Astoria	2	2	2	2	2	2	2	
Battery Park City	1	1	1	1	1	1	1	
Bushwick	2	2	2	2	2	2	2	
Chelsea	15	15	15	15	15	15	15	
East Harlem	4	4	4	4	4	4	4	
East Village	1	1	1	1	1	1	1	
Elmhurst	7	7	7	7	7	7	7	
Financial District	197	197	197	197	197	197	197	
Harlem	9	9	9	9	9	9	9	
Hell's Kitchen	24	24	24	24	24	24	24	
Jackson Heights	2	2	2	2	2	2	2	
Kips Bay	1	1	1	1	1	1	1	
Maspeth	2	2	2	2	2	2	2	
Midtown	9	9	9	9	9	9	9	
Murray Hill	36	36	36	36	36	36	36	
Sunnyside	9	9	9	9	9	9	9	
Theater District	8	8	8	8	8	8	8	
Tribeca	3	3	3	3	3	3	3	
Upper East Side	9	9	9	9	9	9	9	
Upper West Side	2	2	2	2	2	2	2	
West Village	1	1	1	1	1	1	1	
Woodside	14	14	14	14	14	14	14	

In [143]:

```
busiest_host_listings.groupby('host_id').agg('mean').sort_values('price')[['price']]
```

Out[143]:

host_id	price
137358866	43.823529
12243051	198.034483
30283594	243.395349
219517861	270.144928
107434423	285.392857

In [144]:

```
data['price'].mean()
```

Out[144]:

```
142.33252621004095
```

In [145]:

```
busiest_host_listings.groupby('host_id').agg('mean').sort_values('availability_365')[['av
```

Out[145]:

host_id	availability_365
137358866	205.921569
107434423	271.428571
12243051	287.241379
219517861	288.362319
30283594	318.813953

In [146]:

```
data['availability_365'].mean()
```

Out[146]:

```
114.88629865279101
```

In [147]:

```
busiest_host_listings.groupby('host_id').agg('mean').sort_values('reviews_per_month')[['r
```

Out[147]:

host_id	reviews_per_month
30283594	0.091628
107434423	0.215714
12243051	0.298276
137358866	0.444706
219517861	1.920580

In [148]:

```
data['reviews_per_month'].mean()
```

Out[148]:

```
1.373229180082972
```

First, I grouped the data by neighbourhood and sorted by calculated_host_listings_count to determine which neighbourhoods had the largest number of listings. After that, I did grouped the data by host_id and sorted it by calculated_host_listings_count similarly to the previous step. However, I also used the index and the values in the index to create a seperate table that used the host_id as a basis. From this, I was able to display a table depicting host_id vs various other factors. Finally, I printed the average values of each aspect of the original data set. From this, I now had the values of each element of the busiest regions and hosts, and that of the entire data set. I was able to use this to make predictions on why these hosts are the busiest. For instance, the average availablity of the averge AirBnb is 114.88629865279101. However, for the busiest hosts, the availablity was about 200-300. Furthermore, this explains why the price of the busiest host is approximately 100$ more than that of the dataset as a whole.

In [ ]:

```

```

In [ ]:

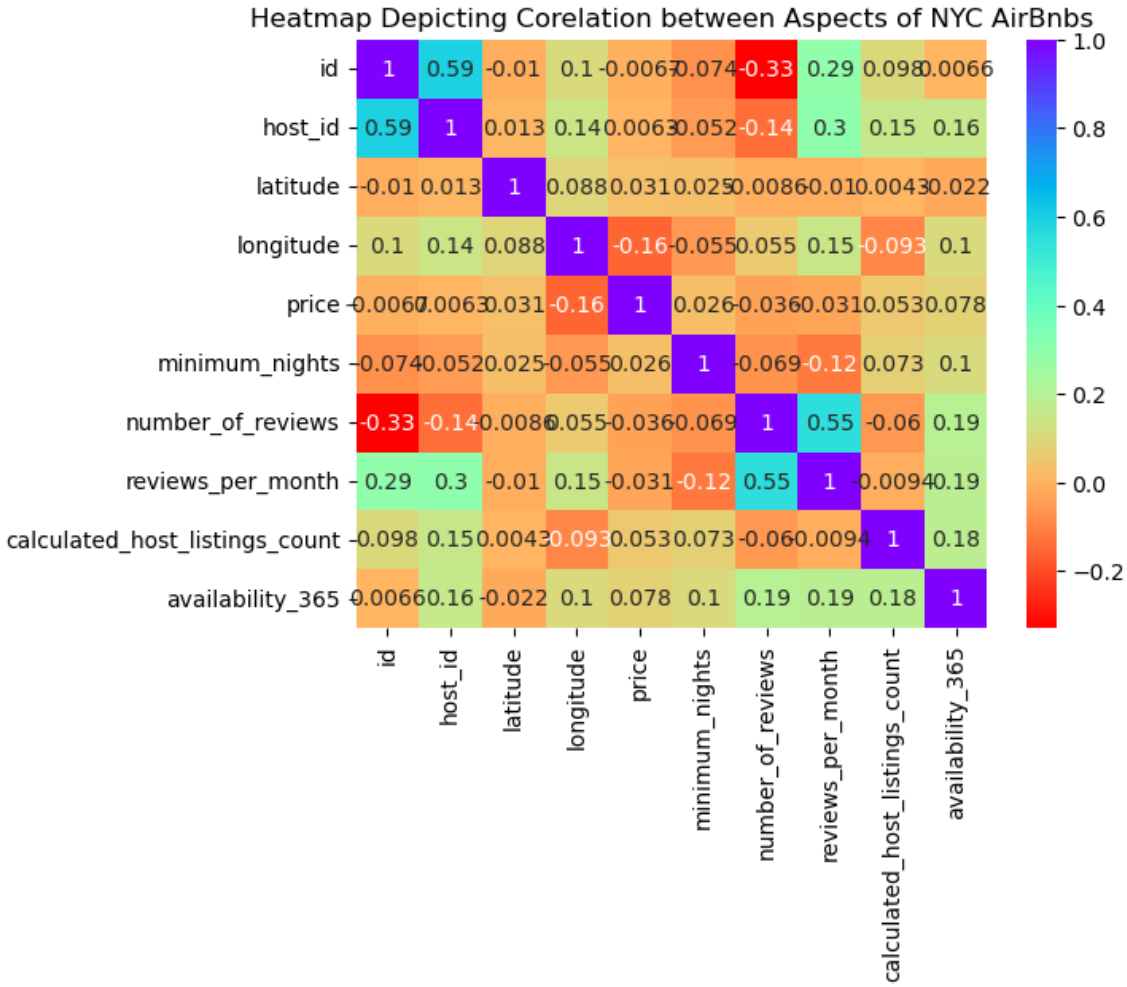
```

```

In [114]:

```
dataplot = sns.heatmap(data.corr(method='pearson'), cmap="rainbow_r", annot=True)
plt.title('Heatmap Depicting Corelation between Aspects of NYC AirBnbs')
```

Out[114]:

```
Text(0.5, 1.0, 'Heatmap Depicting Corelation between Aspects of NYC AirBnb
s')
```



In [132]:

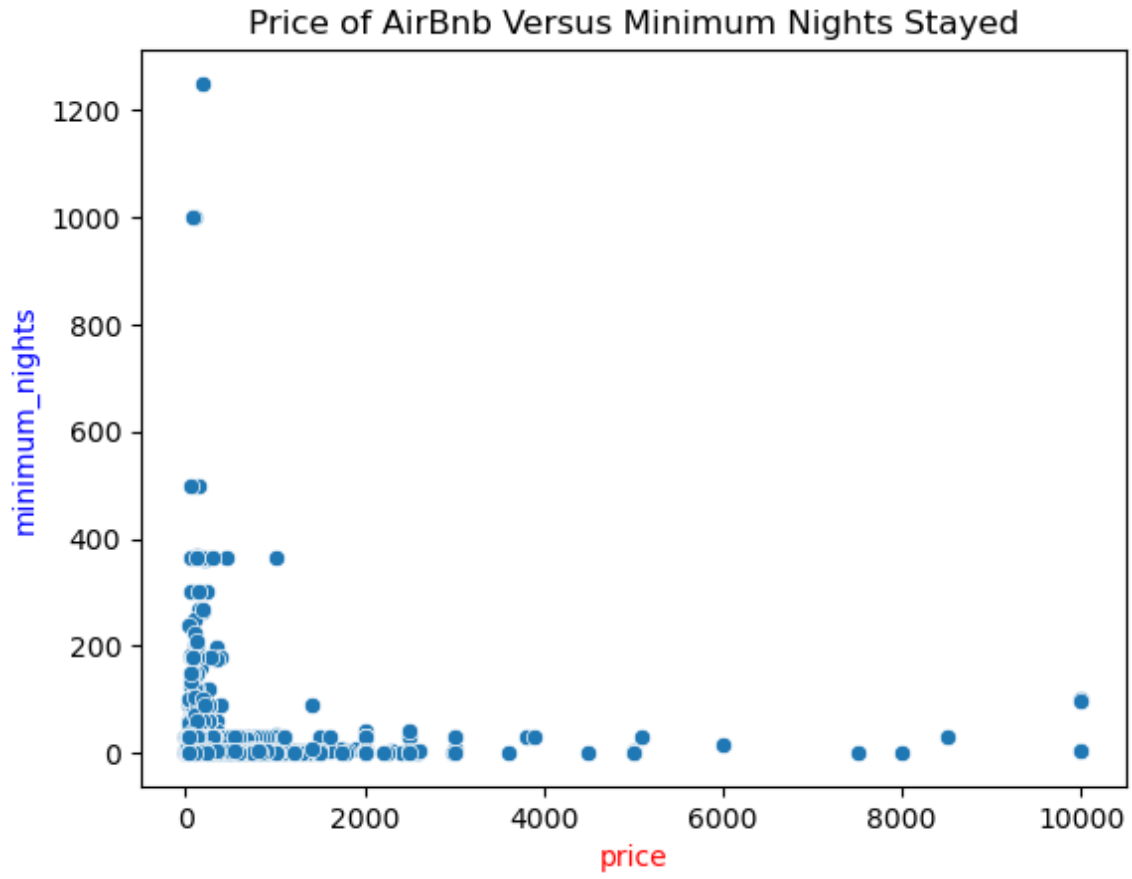
```
minNightsPerPrice = all_neighbourhoods.groupby('minimum_nights').agg('mean').sort_values(
```

In [154]:

```
sns.scatterplot(data=data, x='price', y='minimum_nights')
plt.title('Price of AirBnb Versus Minimum Nights Stayed')
plt.xlabel('price', fontsize=10, color='red')
plt.ylabel('minimum_nights', fontsize=10, color='blue')
```

Out[154]:

```
Text(0, 0.5, 'minimum_nights')
```



The first graph is a heatmap that demonstrates the corelation between every element of the dataset. I think that this is the most interesting model of the entire project. Before starting the assignment, I expected for there to be significant correlations between price and neighbourhood or price and availability. However, I was shocked to discover that the correlation between the elements was mostly approximately 0, meaning that there was no correlation between the elements. The maximum correlation was -0.33 between number of reviews and id. I expected a much larger correlation between a number of the elements. Of course, it is possible that I did something wrong, but if not then its a very interesting observation. The second graph represents how price may affect the minimum nights stayed at an AirBnb. It is evident that the vast majority of the points are in the 0-2000$ range, especially leaning towards the 0. It is logical to conclude that the cheaper an AirBnb is the liklier it is for people to stay there. The interesting thing about this graph is the outliers. Of course, most of the time outliers are a bad thing. However, I think its interesting to consider the outliers in this graph. There are some people who payed 10,000$ for 100-200 nights and others who payed nearly nothing but stayed for 1000+ nights. As such, this is a perfect demosntration of how a single outlier in either axis can skew the data.