In [284…
```
import seaborn as sb #importing necessary packages for working with the data
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
```

In [285…
```
pd.set_option("display.max_rows",100)
```

Step 1: Importing Data

In [286…
```
data = pd.read_csv('AB_NYC_2019.csv') # import data from csv to python using
```

Step 2: Cleaning Data (Question 1)

Missing names for listing and name of host.

Replaced with listing id and host id respectively.

Missing values for reviews per month because 0 is not available. Thus replaced with 0.

In [287…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48879 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48874 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

In [288…
```
ndata = data.isnull()
```

In [289…
```
for i,j in enumerate(ndata['name']): #replacing null name with id
    if(j==True):
        data.loc[i,'name'] = str(data.loc[i,'id'])

for i,j in enumerate(ndata['host_name']): #replacing null host name with hos
    if(j==True):
        data.loc[i,'host_name'] = str(data.loc[i,'host_id'])
for i,j in enumerate(ndata['reviews_per_month']): # replacing null reviews p
    if(j==True):
        data.loc[i,'reviews_per_month'] = 0
```

In [290…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   name                            48895 non-null  object 
 2   host_id                         48895 non-null  int64  
 3   host_name                       48895 non-null  object 
 4   neighbourhood_group             48895 non-null  object 
 5   neighbourhood                   48895 non-null  object 
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object 
 9   price                           48895 non-null  int64  
 10  minimum_nights                  48895 non-null  int64  
 11  number_of_reviews               48895 non-null  int64  
 12  last_review                     38843 non-null  object 
 13  reviews_per_month               48895 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

Step 3: Neighborhood Pricing Trends (Question 2)

In [292…
```
ntc = dict() #total price of neighbourhood and number of listings in that ne
for i,j in enumerate(data['neighbourhood']):
    if not(j in  ntc):
        ntc[j] = [0,0,0,0]
    ntc[j] = [ntc[j][0]+data.loc[i,'price'],ntc[j][1]+1,0,data.loc[i,'neighb
```

In [293…
```
for i in ntc:
    ntc[i] = [ntc[i][0],ntc[i][1],float(ntc[i][0])/float(ntc[i][1]),ntc[i][3
```

In [294…
```
neighbourhooddata=pd.DataFrame(ntc.values(), index=ntc.keys())
neighbourhooddata.columns = ["Total Price", "# of Listings", "Mean Price", "
```

In [312…

```
sortedbymean=neighbourhooddata[neighbourhooddata['# of Listings'] > 5].sort_
sortedbymean
#sort by mean price neighbourhoods with more than 5 listings
```

Out[312…

	Total Price	# of Listings	Mean Price	Neighbourhood Group
Bull's Head	284	6	47.333333	Staten Island
Hunts Point	909	18	50.500000	Bronx
Tremont	567	11	51.545455	Bronx
Soundview	802	15	53.466667	Bronx
Bronxdale	1085	19	57.105263	Bronx
...	...	...	...	...
Flatiron District	27354	80	341.925000	Manhattan
Battery Park City	25729	70	367.557143	Manhattan
Riverdale	4863	11	442.090909	Bronx
Sea Gate	3415	7	487.857143	Brooklyn
Tribeca	86843	177	490.638418	Manhattan

190 rows × 4 columns

The top 5 most expensive neighborhoods are Tribeca, Sea Gate, Riverdale, Battery Park City, and Flatiron District

The top 5 least expensive neighborhoods are Bronxdale, Soundview, Tremont, Hunts Point, and Bull's Head

In [296…

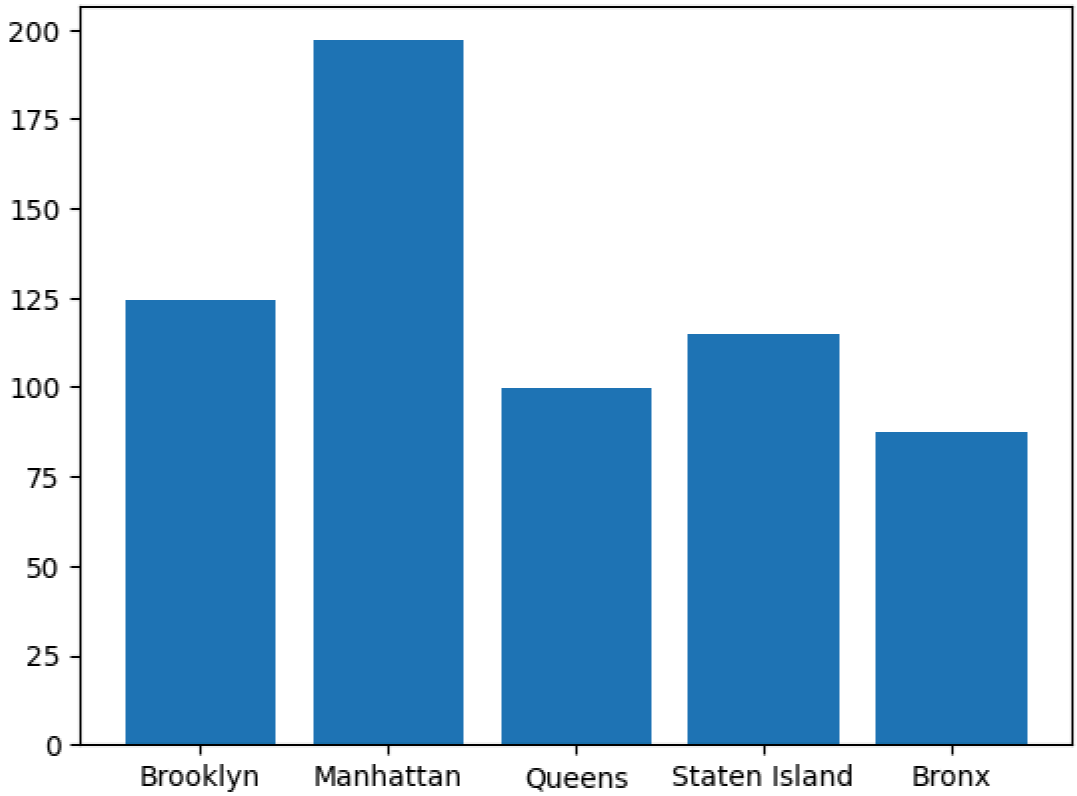
```
ngm = dict() # mean pricing per neighbourhood_group
for i,j in enumerate(data['neighbourhood_group']):
    if not(j in  ngm):
        ngm[j] = [0,0,0]
    ngm[j] = [ngm[j][0]+data.loc[i,'price'],ngm[j][1]+1,0]
```

In [297…

```
for i in ngm:
    ngm[i] = [ngm[i][0],ngm[i][1],float(ngm[i][0])/ngm[i][1]]
plt.bar(ngm.keys(),[i[2] for i in ngm.values()])
```

Out[297…

```
<BarContainer object of 5 artists>
```



Highest Mean Price of Listing is in Manhattan, while lowest mean price of listing is in the Bronx

Step 4: Pearson Analysis (Question 3)

Selecting Features: Latitude, Longitude, Price, Reviews Per Month, Availability_365

In [298…

```
def pearson(a,b): # function for calculating pearson
    mean_a = sum(a)/float(len(a))
    mean_b = sum(b)/float(len(b))
    cov = sum([(i-mean_a)*(j-mean_b)for i,j in zip(a,b)])
    stda = sum([(i-mean_a)**2 for i in a])**0.5
    stdb = sum([(j-mean_b)**2 for j in b])**0.5
    return cov/(stda*stdb)
```

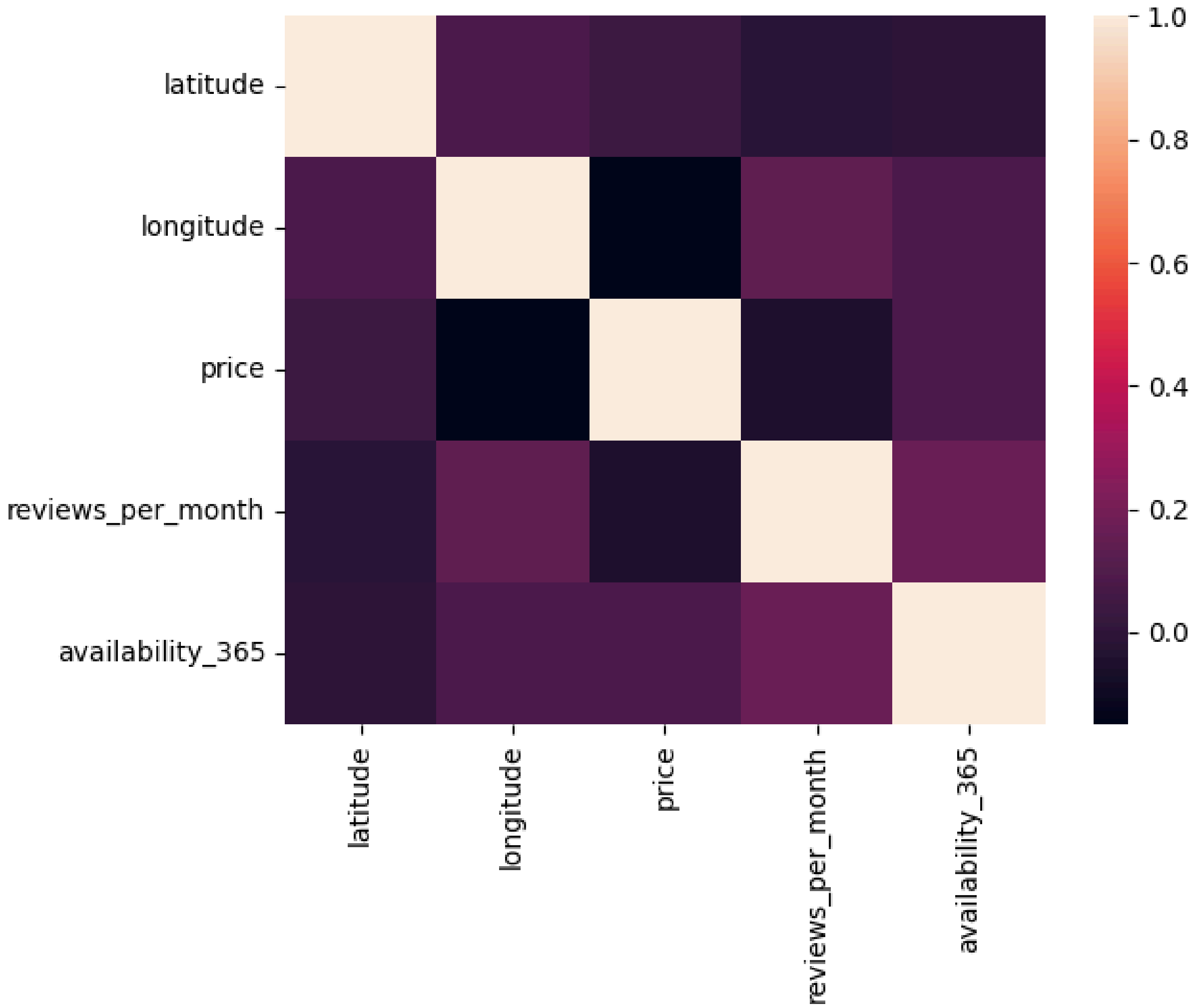
In [299…

```
features = ["latitude","longitude","price","reviews_per_month","availability
pearsonmat = []
max=[-10,"",""]
min=[10,"",""]
for i in features: # finding pearson correlations for each pair of features
    pearsonrow = []
    for j in features:
        p =pearson(data[i],data[j])
        pearsonrow.append(p)
        if i!=j:
            max = [p,i,j] if p> max[0] else max # maximum and minimum correl
            min = [p,i,j] if p< min[0] else min
```

```
    pearsonmat.append(pearsonrow)
print(pearsonmat)
print(max)
print(min)
sb.heatmap(pearsonmat,xticklabels=features,yticklabels=features) # mapping c
```

```
[[1.0, 0.08478836838914387, 0.033938668232625535, -0.01875772712399139, -0.0
10983458290207526], [0.08478836838914387, 1.0000000000000002, -0.15001926996
895382, 0.13851616595337943, 0.08273074786279411], [0.033938668232625535, -
0.15001926996895382, 0.9999999999999998, -0.05056409232837583, 0.08182882742
169546], [-0.01875772712399139, 0.13851616595337943, -0.05056409232837583,
1.0, 0.16373167028258948], [-0.010983458290207526, 0.08273074786279411, 0.08
182882742169546, 0.16373167028258948, 1.0000000000000002]]
[0.16373167028258948, 'reviews_per_month', 'availability_365']
[-0.15001926996895382, 'longitude', 'price']
```

Out[299… <Axes: >



Largest correlation is between reviews per month and availability 365, at 0.1637

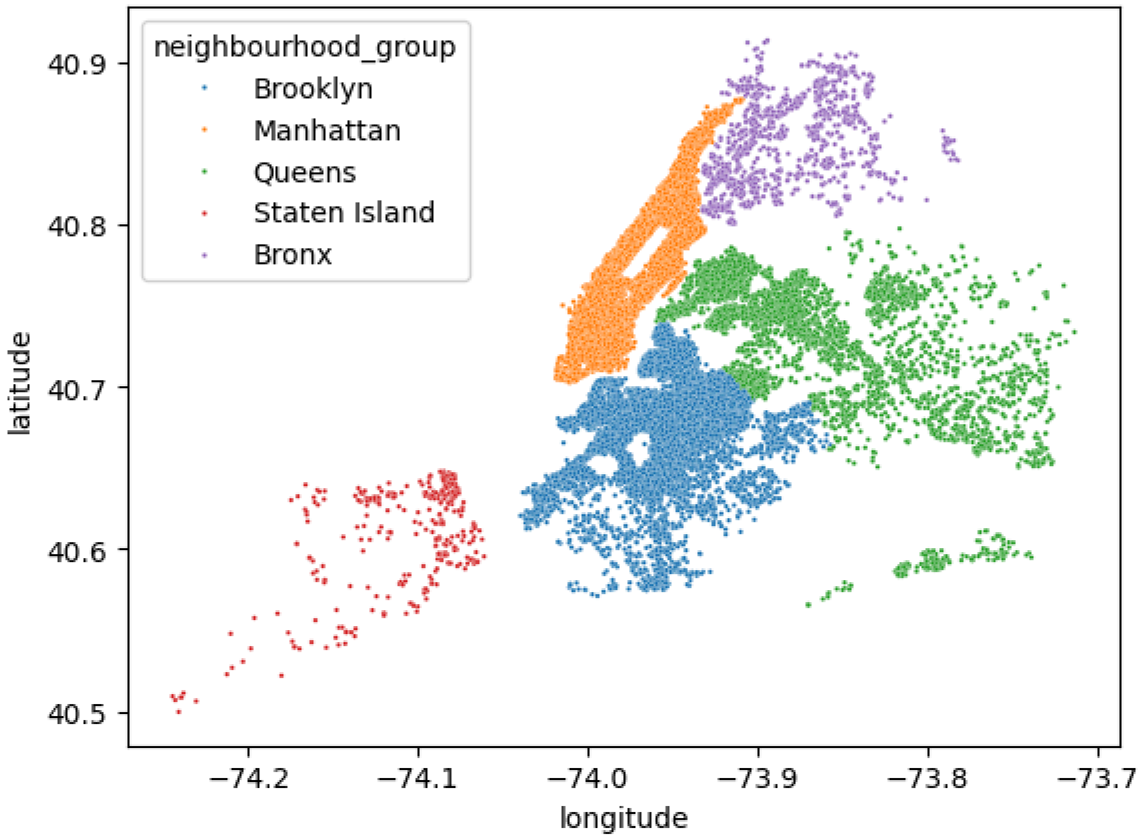
Largest negative correlation is between longitude and price at -0.15

Step 5 Coordinate Trends (Question 4)

Mapping neighbourhood group of listings and pricing of listings using color

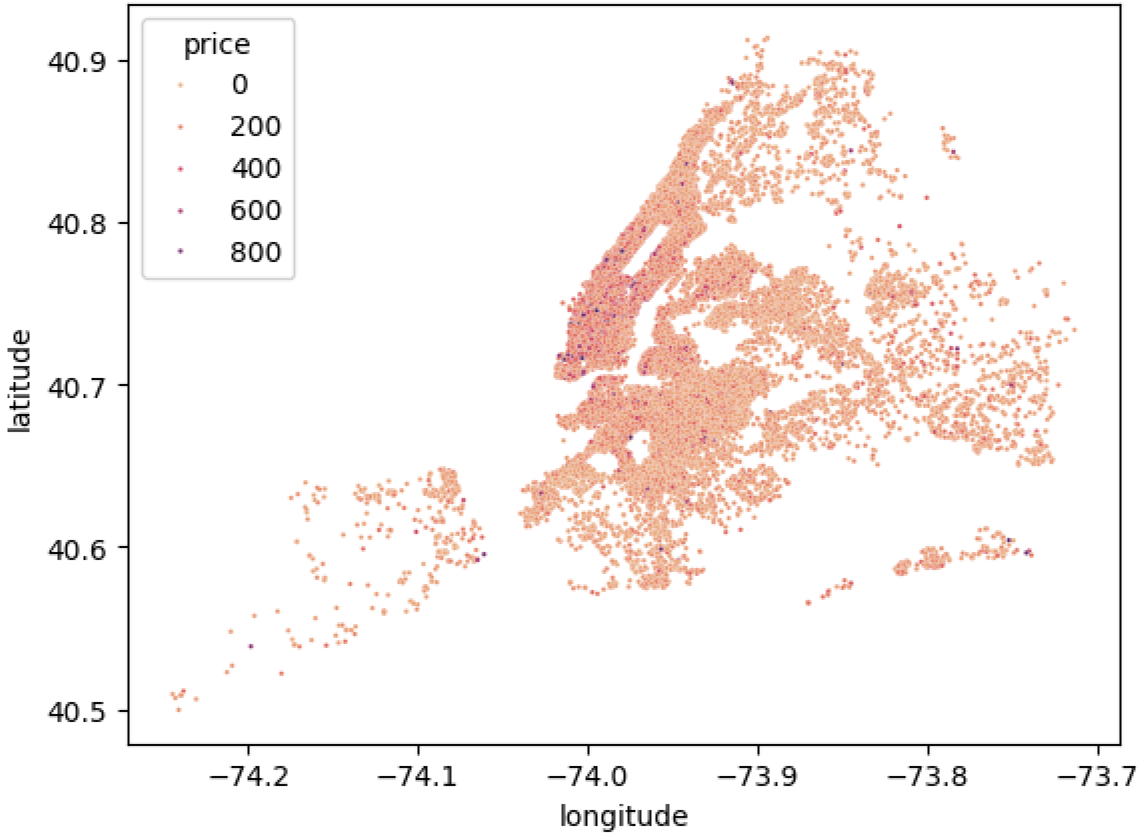
In [300…
```
sb.scatterplot(data = data,x="longitude",y="latitude",hue='neighbourhood_gro
#color map of the listings in each neghbourhood_group
```

Out[300… <Axes: xlabel='longitude', ylabel='latitude'>



In [301…
```
under1k=data[data['price'] < 1000]
sb.scatterplot(data = under1k,x="longitude",y="latitude",hue='price',palette
```

Out[301… <Axes: xlabel='longitude', ylabel='latitude'>



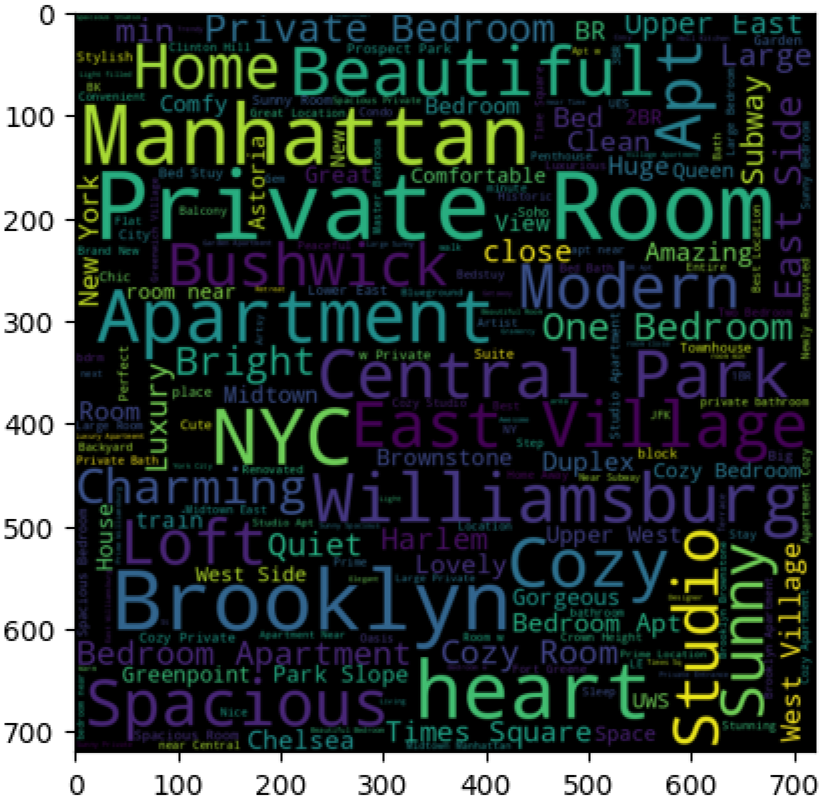
We see that the bottom left side of manhattan is the most expensive, and there is a gradient outwards from there

Step 6: Word Cloud (Question 5)

Word cloud of words used in Airbnb names

In [302…
```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
text = ""
for name in data["name"]:
    text = text+" " + name
wordcloud = WordCloud(width=720, height=720).generate(text)
plt.imshow(wordcloud, interpolation='bilinear')
```

Out[302…
```
<matplotlib.image.AxesImage at 0x750574061d20>
```



Step 7: Busiest Areas (Question 6)

Finding the areas with the busiest hosts which is defined as having a high number of listings.

In [303…

```
nhl = dict() #total price of neighbourhood and number of listings in that ne
for i,j in enumerate(data['neighbourhood']):
    if not(j in  nhl):
        nhl[j] = [0,0,0,0]
    nhl[j] = [nhl[j][0]+data.loc[i,'calculated_host_listings_count'],nhl[j][
```

In [304…

```
for i in nhl: # Finding mean number of listings of each host in a neighbourh
    nhl[i] = [nhl[i][0],ntc[i][1],float(nhl[i][0])/float(nhl[i][1]),nhl[i][3

nhldataframe=pd.DataFrame(nhl.values(), index=nhl.keys())
nhldataframe.columns = ["Total Number of Listings", "# of Hosts", "Mean List
```

In [305…

```
nhldataframe[nhldataframe["# of Hosts"] > 5].sort_values(by="Mean Listings")
```

Out[305…

	Total Number of Listings	# of Hosts	Mean Listings	Neighbourhood Group
Rosebank	7	7	1.000000	Staten Island
Shore Acres	7	7	1.000000	Staten Island
Mariners Harbor	8	8	1.000000	Staten Island
Melrose	10	10	1.000000	Bronx
Grymes Hill	7	7	1.000000	Staten Island
...	...	...	...	...
Woodside	4554	235	19.378723	Queens
Tribeca	7606	177	42.971751	Manhattan
Murray Hill	26125	485	53.865979	Manhattan
Theater District	18704	288	64.944444	Manhattan
Financial District	85454	744	114.857527	Manhattan

190 rows × 4 columns

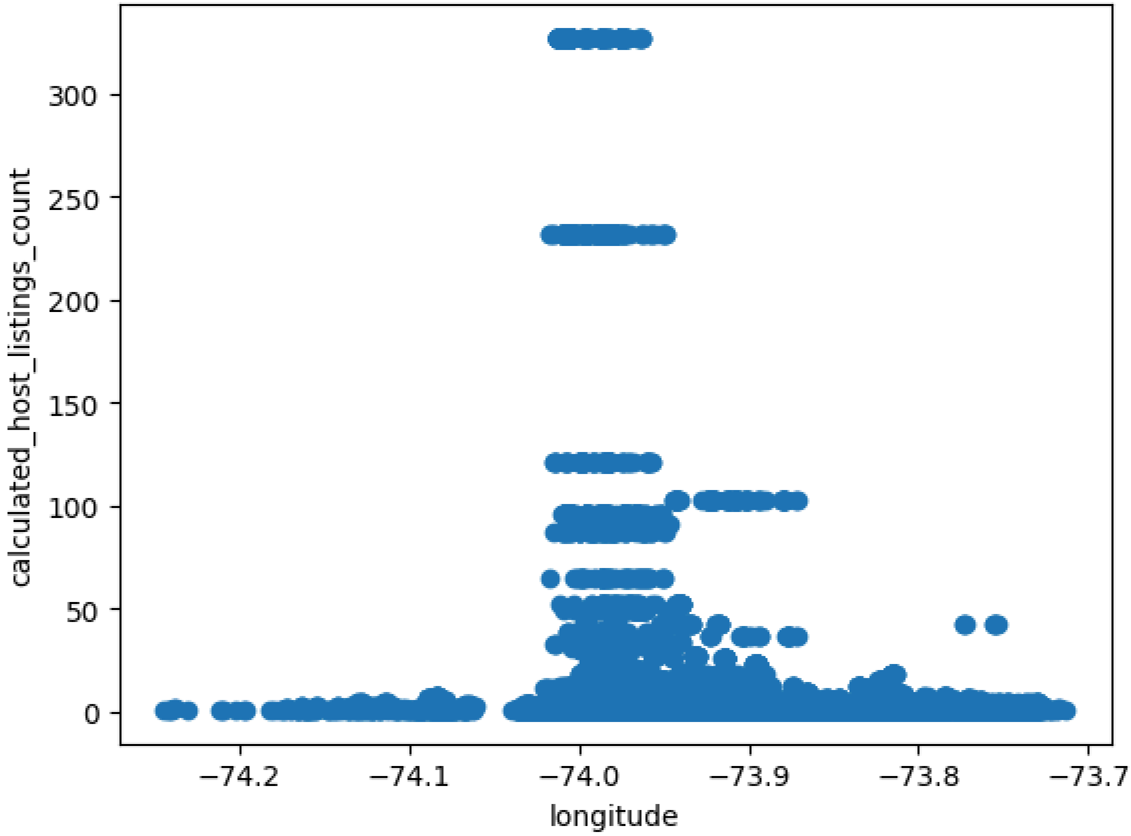
Considering only neighbourhoods with at least 5 listings, We Have Tribeca having the highest mean number of listings at 490.64 per host.

In [306…

```
plt.scatter(data['longitude'],data['calculated_host_listings_count'])
plt.xlabel('longitude')
plt.ylabel('calculated_host_listings_count')
```

Out[306…

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



We see above that near the center in longitude is where hosts are the busiest.

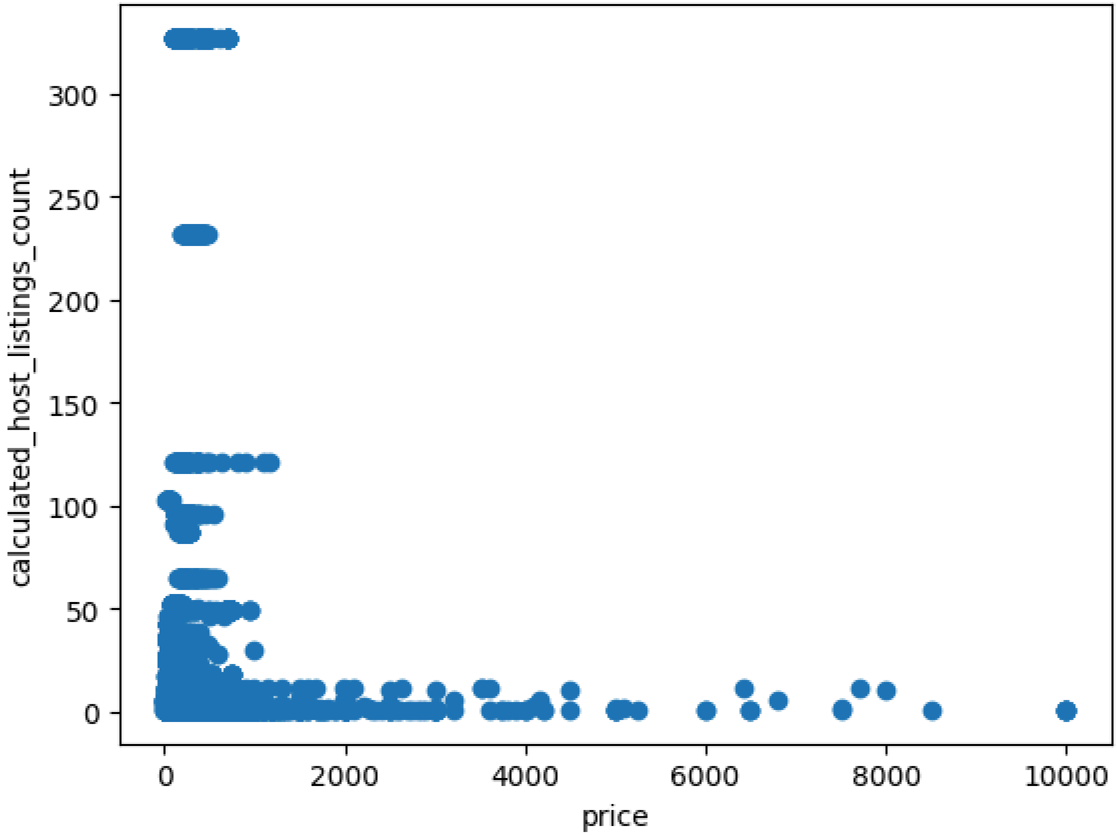
In [307…

```
plt.scatter(data['price'],data['calculated_host_listings_count'])
print("pearson correlation:" + str(pearson(data['price'],data['calculated_ho
plt.xlabel('price')
plt.ylabel('calculated_host_listings_count')
```

```
pearson correlation:0.057471688368065814
```

Out[307…

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



We see above that hosts with a lot of listings are more likely to be at the bottom part of the price range.

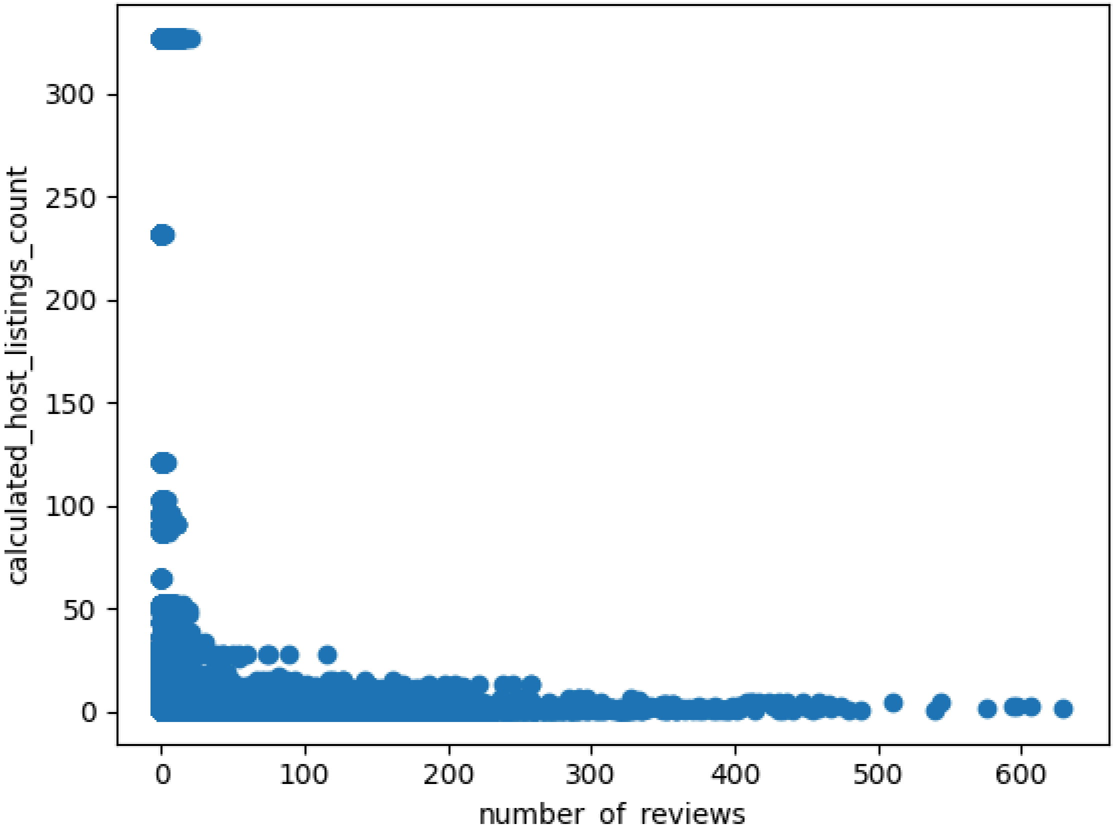
In [308…

```
plt.scatter(data['number_of_reviews'],data['calculated_host_listings_count']
print("pearson correlation: " + str(pearson(data['number_of_reviews'],data['
plt.xlabel('number_of_reviews')
plt.ylabel('calculated_host_listings_count')
```

```
pearson correlation: -0.07237606054175609
```

Out[308…

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

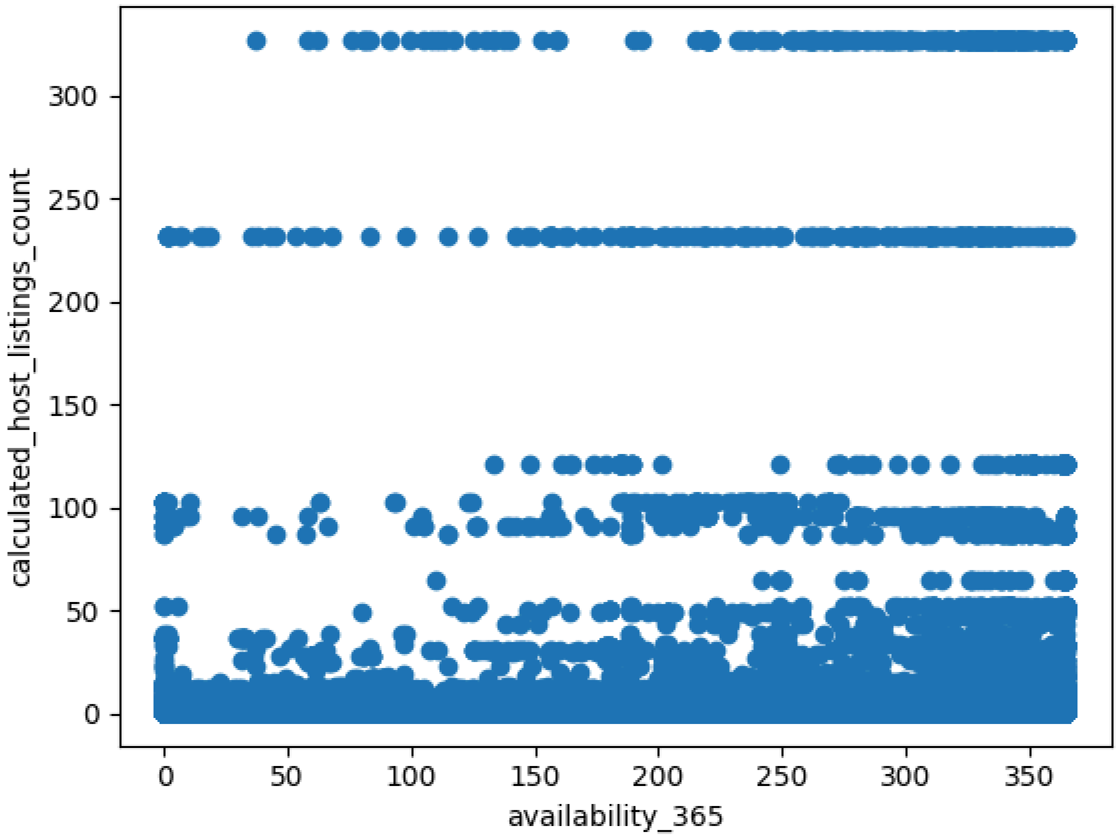


We see above that hosts with a lot of listings don't have that many reviews for each of their listings

In [309…
```
plt.scatter(data['availability_365'],data['calculated_host_listings_count'])
print("pearson correlation: " + str(pearson(data['availability_365'],data['c
plt.xlabel('availability_365')
plt.ylabel('calculated_host_listings_count')
```

```
pearson correlation: 0.2257013721911263
```

Out[309… Text(0, 0.5, 'calculated_host_listings_count')



Through the pearson correlation of 0.226, we may conclude that the longer the availability, the more listings the host has.

From the above graphs we can conclude that the hosts with a lot of listings are at the bottom of the price range. We can also conclude that hosts with a lot of listings have a longitude that is at the center of manhattan. The financial district has a longitude of -74.01, which is near the center of manhattan and also has a mean price per listing of 225.50 which is near the bottom of the price range. This is why the financial district has the busiest hosts.

Step 8: Interesting Plots (Question 7)

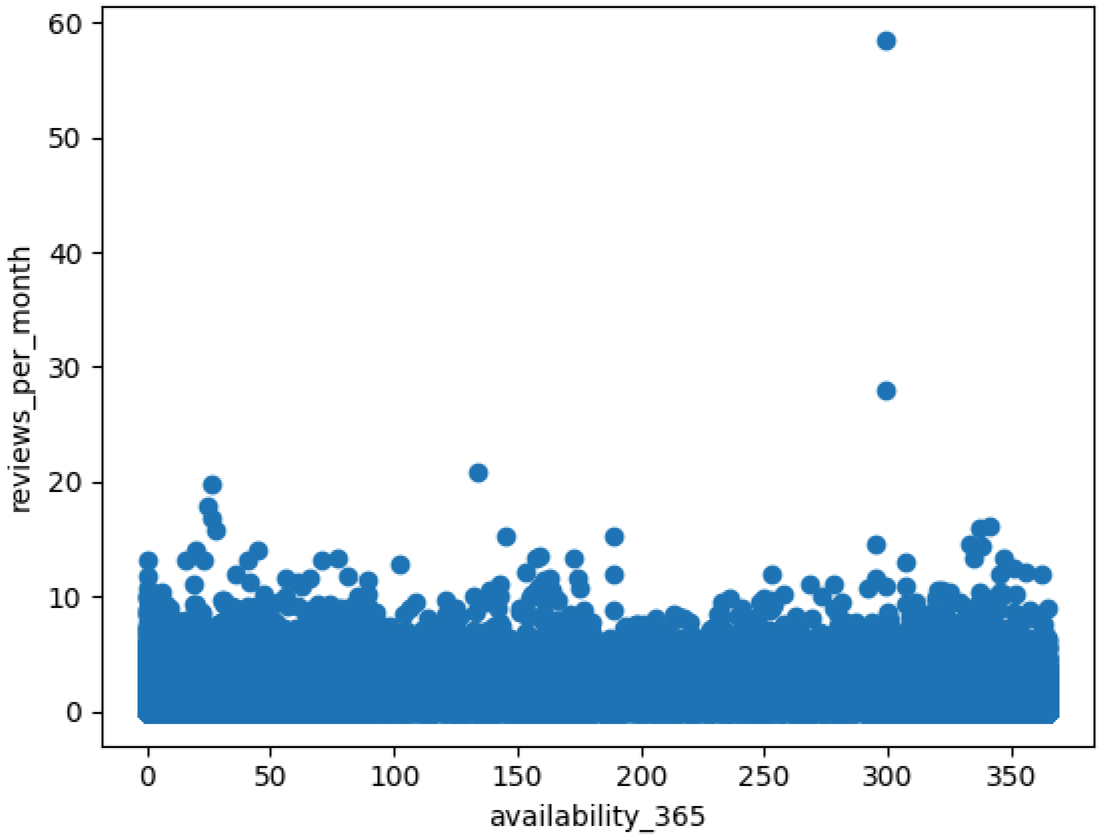
In [310…

```
print("pearson correlation: " + str(pearson(data['availability_365'],data['r
plt.scatter(data['availability_365'],data['reviews_per_month'])
plt.xlabel('availability_365')
plt.ylabel('reviews_per_month')
```

```
pearson correlation: 0.16373167028258948
```

Out[310…

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



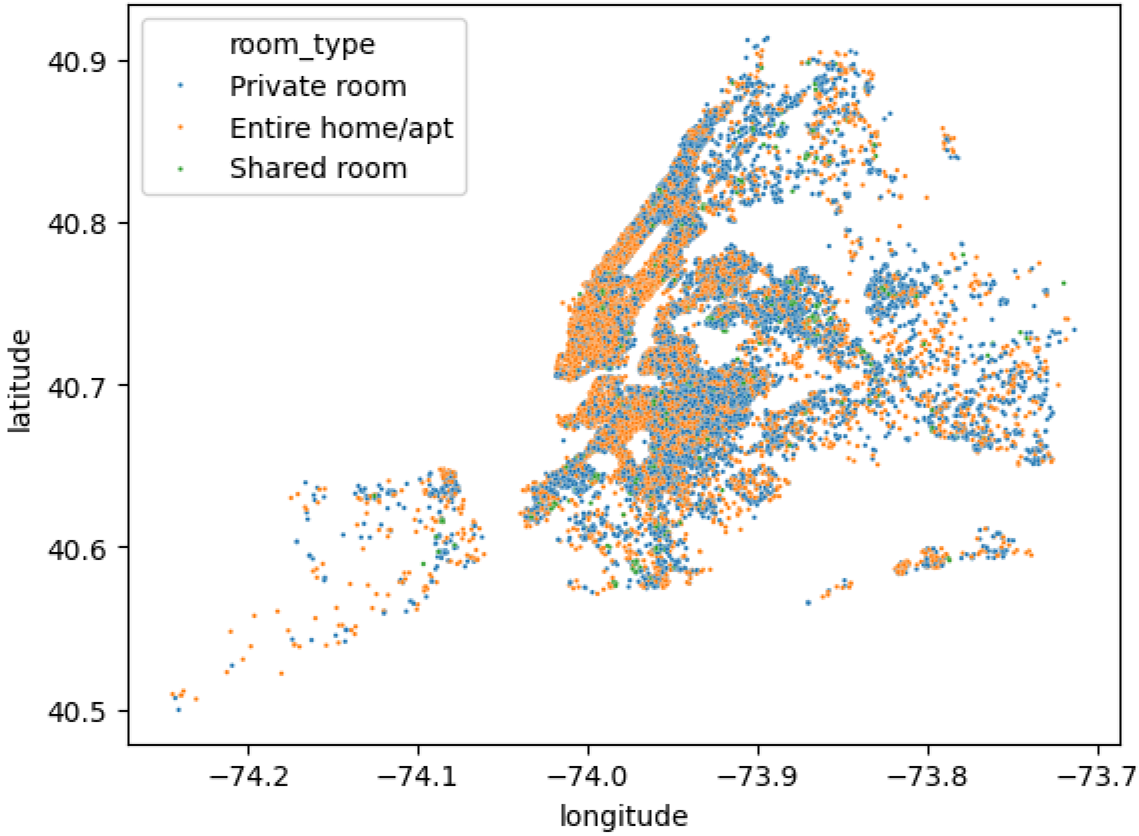
The above plot is that of reviews per month vs availability. There is a pearson correlation of 0.163, however it is interesting that this correlation isn't stronger. One would think that the longer the availability, the more reviews it should have due to more bookings.

In [311…

```
sb.scatterplot(data = data,x="longitude",y="latitude",hue='room_type',s=3)
#color map of the room type
```

Out[311…

```
<Axes: xlabel='longitude', ylabel='latitude'>
```



This is a color map of room type for each listing. We see that in Manhattan, most rooms are Entire home/apt, whereas as you move further out from manhattan, private rooms begin taking over.