Methodology Document:

Problem background

- Suppose that you are working as a data analyst at Airbnb. For the past few months, Airbnb has seen a major decline in revenue. Now that the restrictions have started lifting and people have started to travel more, Airbnb wants to make sure that it is fully prepared for this change.

End Objective

- To prepare for the next best steps that Airbnb needs to take as a business, you have been asked to analyse a dataset consisting of various Airbnb listings in New York. Based on this analysis, you need to give two presentations to the following groups.

Presentation - I

- Data Analysis Managers: These people manage the data analysts directly for processes and their technical expertise is basic.
- Lead Data Analyst: The lead data analyst looks after the entire team of data and business analysts and is technically sound.

Presentation - II

- Head of Acquisitions and Operations, NYC: This head looks after all the property and host acquisitions and operations. Acquisition of the best properties, price negotiation, and negotiating the services the properties offer falls under the purview of this role.
- Head of User Experience, NYC: The head of user experience looks after the customer preferences and also handles the properties listed on the website and the Airbnb app. basically, the head of user experience tries to optimise the order of property listing in certain neighbourhoods and cities in order to get every property the optimal amount of traction.

Methodology

1. Loading the Dataset

Imported necessary libraries and loaded the dataset and observed the head and tail.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
data = pd.read_csv('AB_NYC_2019.csv')
data.head(2)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45

```
data.tail(2)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_o
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	

2. Inspecting the Dataset

Observed the shape of the Dataset and datatypes of different columns and changed the datatypes of the columns which are not appropriate. Observed the statistical information of the Numerical columns.

```
data.shape
```

```
(48895, 16)
```

```
data.info()
```

```
# Changing the datatypes of id and Host_id columns
data["id"]=data["id"].astype(object)
data["host_id"]=data["host_id"].astype(object)
```

```
data.last_review = pd.to_datetime(data.last_review)
```

```
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  object
 1   name                            48879 non-null  object
 2   host_id                         48895 non-null  object
 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  datetime64[ns]
 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: datetime64[ns](1), float64(3), int64(5), object(7)
memory usage: 6.0+ MB
```

```
data.describe()
```

	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	48895.000000
mean	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.143982	112.781327
std	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.952519	131.622289
min	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.000000	0.000000
25%	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.000000	0.000000
50%	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.000000	45.000000
75%	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.000000	227.000000
max	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	365.000000

3. Data Cleaning

Done missing value Treatment, Removed the unnecessary columns and identified the outliers.

3.1 Missing Values

```
data.isnull().sum()
```

```
8]: id                                    0
    name                                 16
    host_id                               0
    host_name                            21
    neighbourhood_group                   0
    neighbourhood                         0
    latitude                              0
    longitude                             0
    room_type                             0
    price                                 0
    minimum_nights                        0
    number_of_reviews                     0
    last_review                       10052
    reviews_per_month                 10052
    calculated_host_listings_count        0
    availability_365                      0
    dtype: int64
```

```
round((data.isnull().sum()/len(data))*100,2)
```

```
id                                 0.00
name                               0.03
host_id                            0.00
host_name                          0.04
neighbourhood_group                0.00
neighbourhood                      0.00
latitude                           0.00
longitude                          0.00
room_type                          0.00
price                              0.00
minimum_nights                     0.00
number_of_reviews                  0.00
last_review                       20.56
reviews_per_month                 20.56
calculated_host_listings_count     0.00
availability_365                   0.00
dtype: float64
```

- We have 20.56% of missing values in the columns last_review and reviews_per_month and 0.03% of missing values in column name and 0.04% of missing values in host_name.
- It can be observed that both the columns last_review and reviews_per_month has same number of missing values because of a reason.

```
#Analysing number_of_reviews column
data[data['number_of_reviews']==0]
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3
19	7750	Huge 2 BR Upper East Cental Park	17985	Sing	Manhattan	East Harlem	40.79685	-73.94872	Entire home/apt	190	7

```
len(data[data['number_of_reviews']==0])
```

```
10052
```

- So, when number_of_reviews is 0 then the value in the columns last_review and reviews_per_month is NaT and NaN which means if reviews are not given atleast once how can they calculate last_review and reviews_per_month.

```
data['reviews_per_month'] = data['reviews_per_month'].fillna(value=0)
data['name'] = data['name'].fillna(value='None')
data['host_name'] = data['host_name'].fillna(value='None')
```

```
data.isnull().sum()
```

```
id                                  0
name                                0
host_id                             0
host_name                           0
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     10052
reviews_per_month                   0
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

Removing Unnecessary Columns ¶

can remove last_review column as it will be of no use

```
data.drop('last_review',axis=1,inplace=True)
```

```
data.isnull().sum()
```

```
id                                0
name                              0
host_id                           0
host_name                         0
neighbourhood_group               0
neighbourhood                     0
latitude                          0
longitude                         0
room_type                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

```
data.info()
```

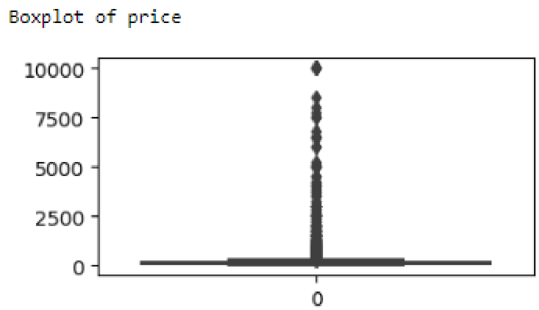
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  object 
 1   name                            48895 non-null  object 
 2   host_id                         48895 non-null  object 
 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  reviews_per_month               48895 non-null  float64
 13  calculated_host_listings_count  48895 non-null  int64  
 14  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(5), object(7)
memory usage: 5.6+ MB
```

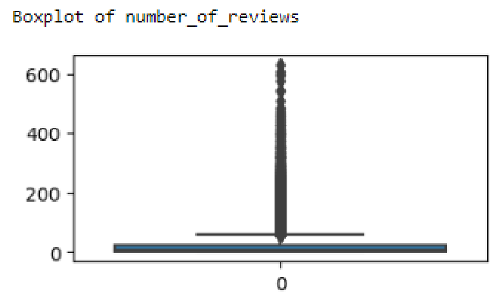
3.3 Identifying Outliers

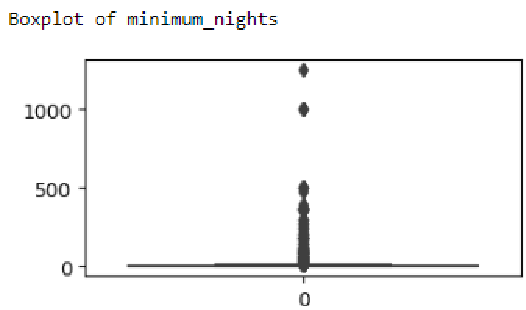
Given

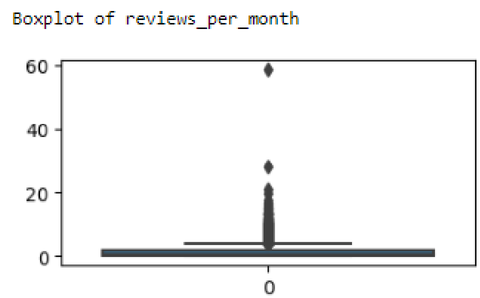
- room_type, neighbourhood_group, neighbourhood are categorical variables
- price, minimum_nights,number_of_reviews, reviews_per_month, calculated_host_listings_count, availability_365 are continuous variables(numerical)

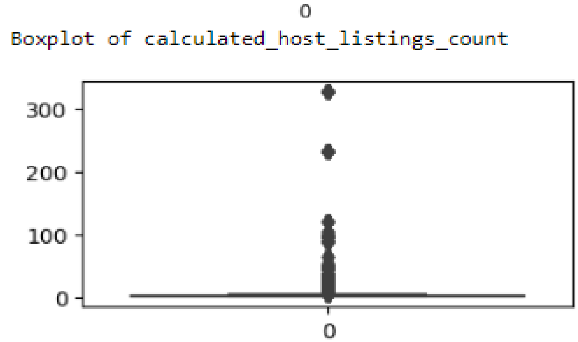
```
for i in con_cols:
    plt.figure(figsize=(4,2))
    print("Boxplot of",i)
    sns.boxplot(data[i])
    plt.show()
```

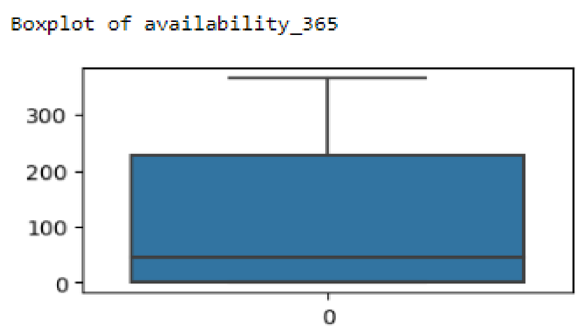












- Price of some listings can be high because of their popularity and number_of_reviews.

- Reviews of some listings can be high because of their ambiance and hospitality.

- If reviews are higher then there is a way that reviews_per_month of that listing can be high

- A single person can have many listings

- So, we cannot considers above columns has outliers as they are in a natural way

4. Identifying and Binning Continuous Variables

```
con_cols = data[['price', 'minimum_nights','number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'avail
```

Continuous variables can be binned into groups

```
con_cols.describe(percentiles = [0.25,0.5,0.75,0.9,0.95,1])
```

2]:

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000
mean	152.720687	7.029962	23.274466	1.090910	7.143982	112.781327
std	240.154170	20.510550	44.550582	1.597283	32.952519	131.622289
min	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000
25%	69.000000	1.000000	1.000000	0.040000	1.000000	0.000000
50%	106.000000	3.000000	5.000000	0.370000	1.000000	45.000000
75%	175.000000	5.000000	24.000000	1.580000	2.000000	227.000000
90%	269.000000	28.000000	70.000000	3.250000	5.000000	337.000000
95%	355.000000	30.000000	114.000000	4.310000	15.000000	359.000000
100%	10000.000000	1250.000000	629.000000	58.500000	327.000000	365.000000
max	10000.000000	1250.000000	629.000000	58.500000	327.000000	365.000000

- Price is ranging from 0 to 10000 dollars
- Highest number of reviews are 629
- A single person is holding 327 listings

Bucketed some continuous columns

1. Bucketing price column

```
con_vars['price'].describe(percentiles = [0.25,0.5,0.75,0.9])
```

]:

```
count    48895.000000
mean       152.720687
std        240.154170
min          0.000000
25%         69.000000
50%        106.000000
75%        175.000000
90%        269.000000
max      10000.000000
Name: price, dtype: float64
```

```
def price_categories_function(row):
    if 0<=row<=69:
        return '0-69'
    elif 70<=row<=106:
        return '70-106'
    elif 107<=row<=175:
        return '107-175'
    elif 176<=row<=269:
        return '176-269'
    else:
        return '269-10000'
```

```
data['price_categories'] = data.price.map(price_categories_function)
data['price_categories']
```

2. Categorizing number_of_reviews column

```
con_vars['number_of_reviews'].describe(percentiles = [0.25,0.5,0.75,0.9,0.99,1])
```

```
count    48895.000000
mean        23.274466
std         44.550582
min          0.000000
25%          1.000000
50%          5.000000
75%         24.000000
90%         70.000000
99%        214.000000
100%       629.000000
max        629.000000
Name: number_of_reviews, dtype: float64
```

```
def number_of_reviews_categories_function(row):
    if 0<=row<=10:
        return '0-10'
    elif 11<=row<=25:
        return '11-25'
    elif 26<=row<=70 :
        return '26-70'
    elif 70<=row<=214:
        return '70-214'
    else:
        return '215-629'
```

```
data['number_of_reviews_categories'] = data.number_of_reviews.map(number_of_reviews_categories_function)
data['number_of_reviews_categories']
```

3. Categorizing calculated_host_listings_count column

```
con_vars['calculated_host_listings_count'].describe(percentiles = [0.25,0.5,0.75,0.9,0.95,0.96,0.97,0.98,0.99])
```

```
count    48895.000000
mean         7.143982
std         32.952519
min          1.000000
25%          1.000000
50%          1.000000
75%          2.000000
90%          5.000000
95%         15.000000
96%         28.240000
97%         49.000000
98%         91.000000
99%        232.000000
max        327.000000
Name: calculated_host_listings_count, dtype: float64
```

```
def calculated_host_listings_count_categories_function(row):
    if 1<=row<=30:
        return '1-30'
    elif 31<=row<=50:
        return '31-50'
    elif 51<=row<=100:
        return '51-100'
    elif 101<=row<=235:
        return '101-235'
    else:
        return '236-327'
```

```
data['calculated_host_listings_count_categories'] = data.calculated_host_listings_count.map(calculated_host_listings_count_ca
data['calculated_host_listings_count_categories']
```

4. Categorizing availability_365 column

```
con_vars['availability_365'].describe(percentiles = [0.25,0.5,0.75,0.9,0.95,1])
```

```
count    48895.000000
mean       112.781327
std        131.622289
min          0.000000
25%          0.000000
50%         45.000000
75%        227.000000
90%        337.000000
95%        359.000000
100%       365.000000
max        365.000000
Name: availability_365, dtype: float64
```

```
def availability_365_categories_function(row):
    if 0<=row<=50:
        return '0-50'
    elif 51<=row<=150:
        return '51-150'
    elif 151<=row<=230 :
        return '151-230'
    elif 231<=row<=300:
        return '231-300'
    else:
        return '301-365'
```

```
data['availability_365_categories'] = data.availability_365.map(availability_365_categories_function)
data['availability_365_categories']
```

Finally converted the Cleaned file into a CSV file and performed some analysis in Jupyter Notebook as well as in visualization tool Tableau.

```
data.to_csv('airbnb_data.csv',index=None)
```

```
data.head(2)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45

```
data.shape
```

```
(48895, 19)
```

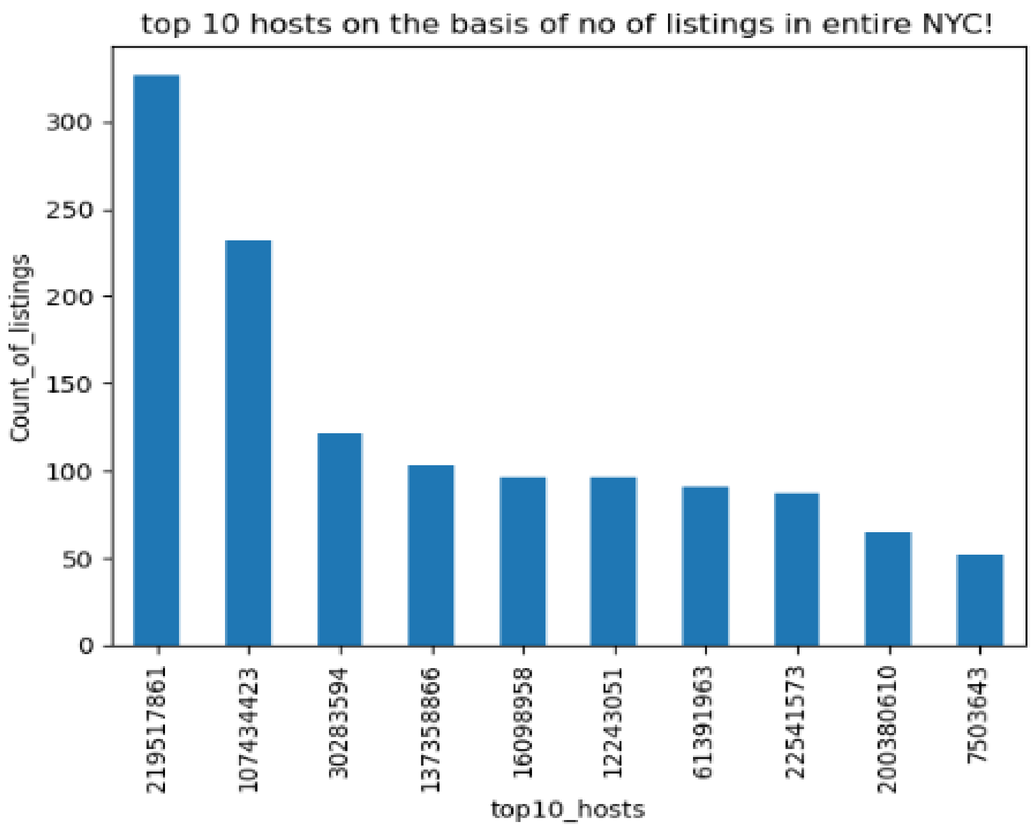
So finally we have 48895 rows and 19 columns.

5. EDA (Exploratory Data Analysis)

Top 10 Hosts

```
top_10_hosts = data['host_id'].value_counts()[:10]
```

```
top_10_hosts.plot(kind='bar')
plt.xlabel('top10_hosts')
plt.ylabel('Count_of_listings')
plt.title('top 10 hosts on the basis of no of listings in entire NYC!')
plt.show()
```

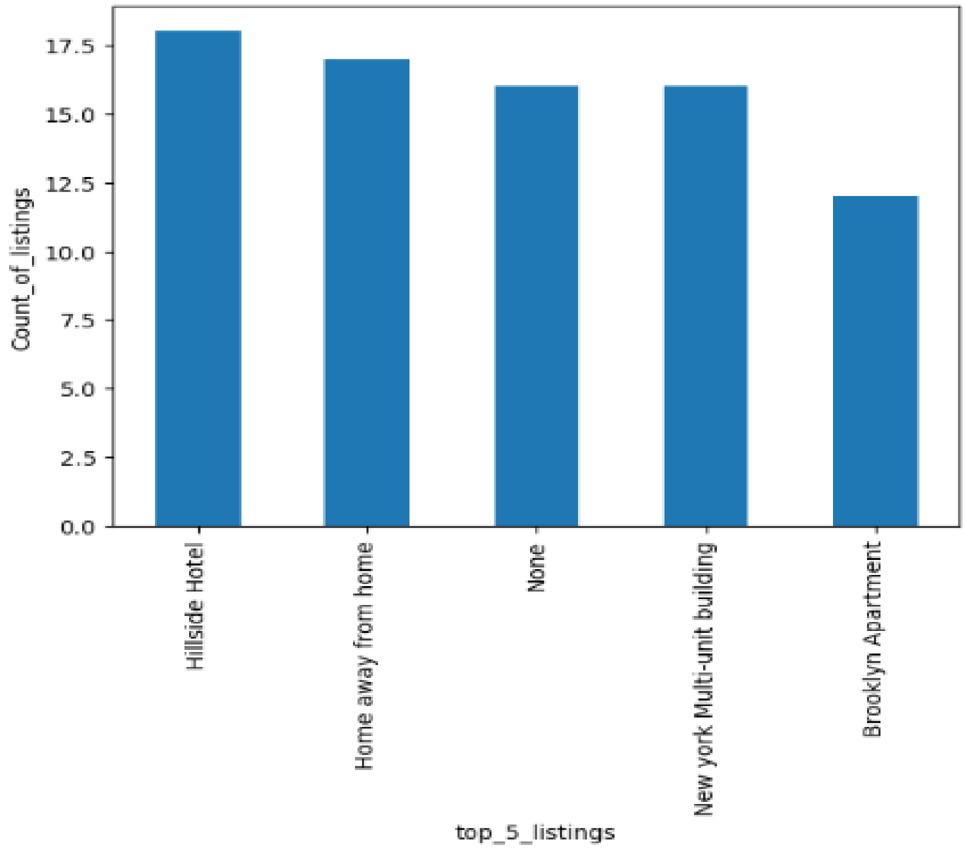


Hosts with above host_id numbers are the top_10_hosts with a max listings of 327

Highest Listing Names

```
top_5_listing_names = data['name'].value_counts()[:5]
```

```
top_5_listing_names.plot(kind='bar')
plt.xlabel('top_5_listings')
plt.ylabel('Count_of_listings')
plt.show()
```



- **Hillside Hotel is found to have listed more listings in entire NYC, followed by Home away from Home.**

Neighbourhood_group needed to targeted?

```
data['neighbourhood_group'].value_counts()
```

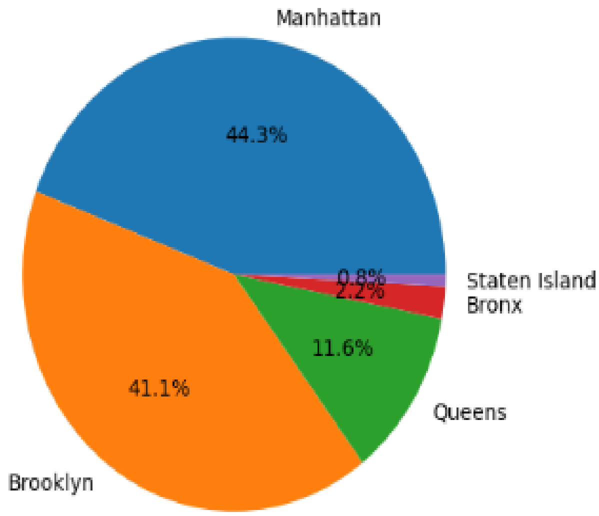
```
;4]: Manhattan        21661
     Brooklyn         20104
     Queens            5666
     Bronx             1091
     Staten Island      373
     Name: neighbourhood_group, dtype: int64
```

```
round(data.neighbourhood_group.value_counts(normalize= True) * 100,1)
```

```
;5]: Manhattan        44.3
     Brooklyn         41.1
     Queens           11.6
     Bronx             2.2
     Staten Island     0.8
     Name: neighbourhood_group, dtype: float64
```

```
dt = [44.3,41.1,11.6,2.2,0.8]
keys = ['Manhattan','Brooklyn','Queens','Bronx','Staten Island']
```

```
plt.pie(dt,labels=keys, autopct='%.1f%%')
plt.show()
```

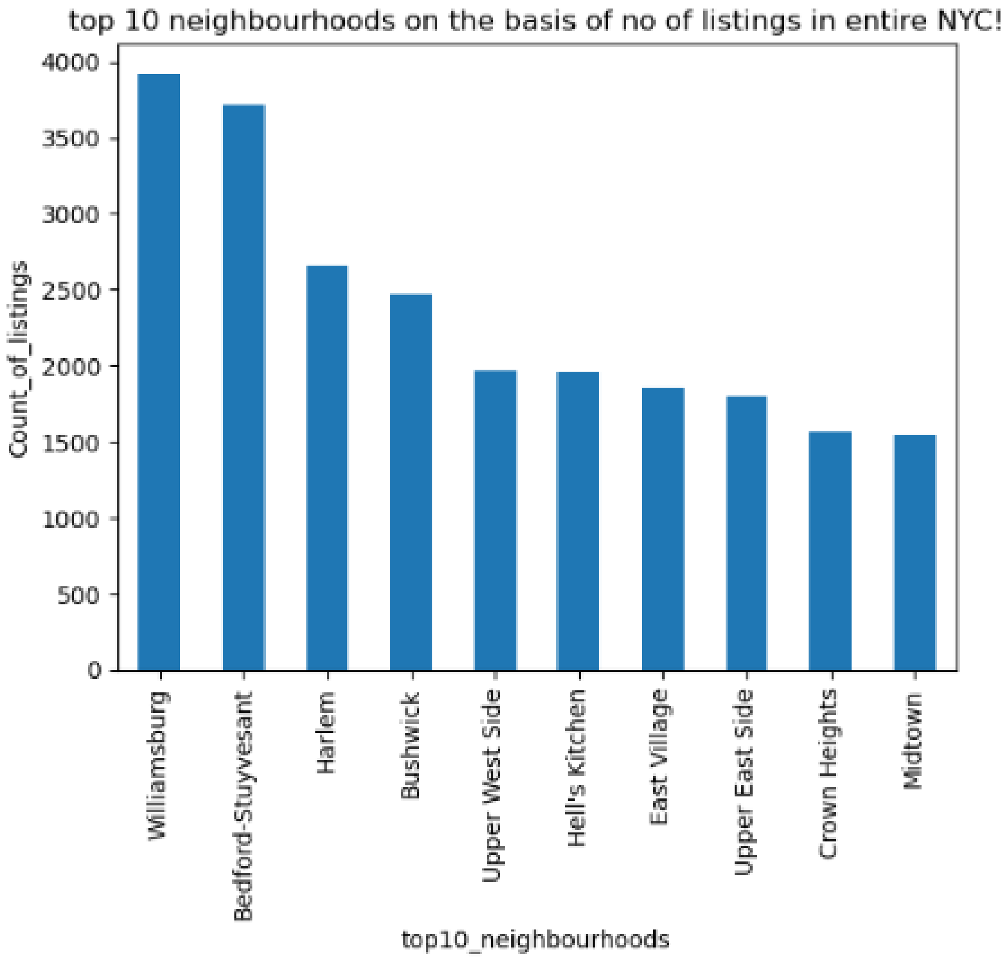


- The properties listed at Manhattan is 44.3% and Brooklyn is 41.1% which contributes about approx 85.4% of Newyork properties
- While other three neighbourhood_groups Staten Island,Queens,Bronx contributes only 14.6% of Newyork properties
- We must target Staten Island,Queens,Bronx for acquisition of more properties.

Neighbourhoods to be Targeted

```
top_10_neighbourhoods = data['neighbourhood'].value_counts()[:10]
```

```
top_10_neighbourhoods.plot(kind='bar')
plt.xlabel('top10_neighbourhoods')
plt.ylabel('Count_of_listings')
plt.title('top 10 neighbourhoods on the basis of no of listings in entire NYC!')
plt.show()
```



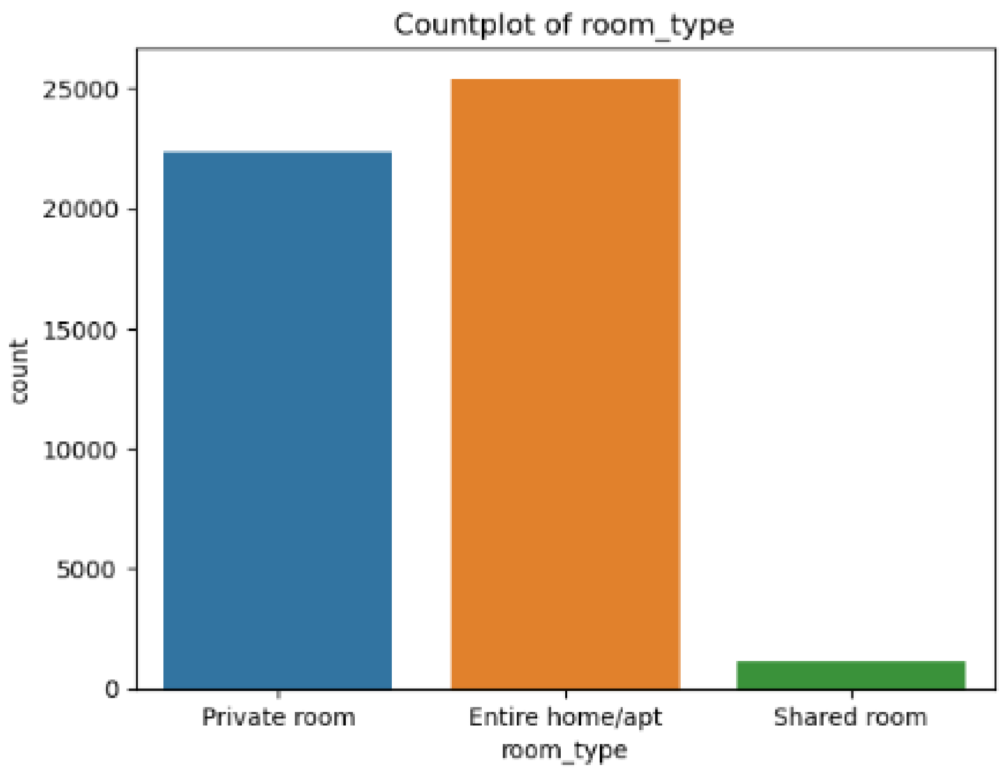
- Willamsburg ranks the top area to have more listings followed by BEdford-Stuyvesant
- So they can target on areas from where lower listings are found

Room Type

```
data['room_type'].value_counts()
```

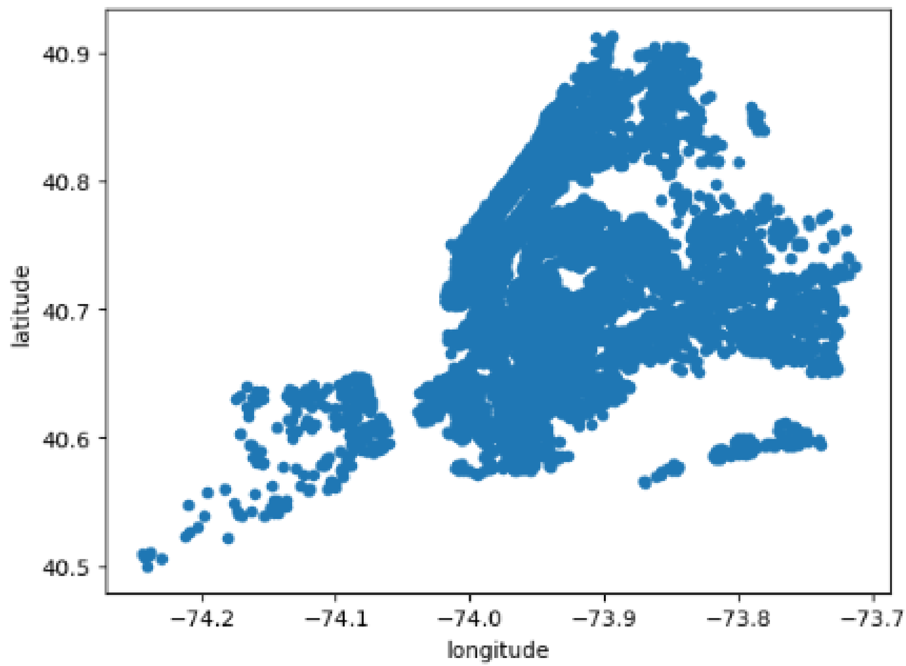
```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

```
plt.title('Countplot of room_type')
sns.countplot(x = 'room_type', data = data)
plt.show()
```

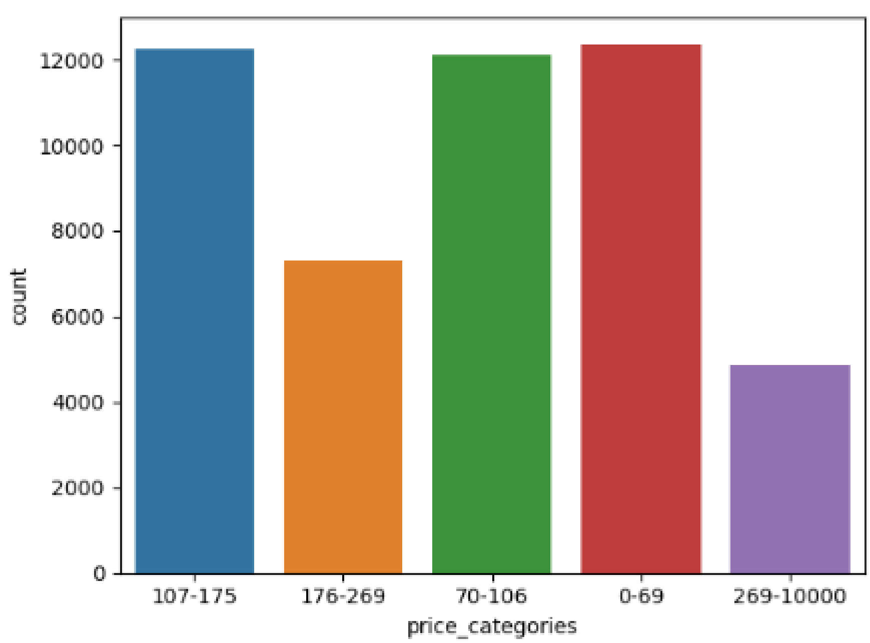


Most of the people are interested in staying in Entire home/apt than in shared and private room

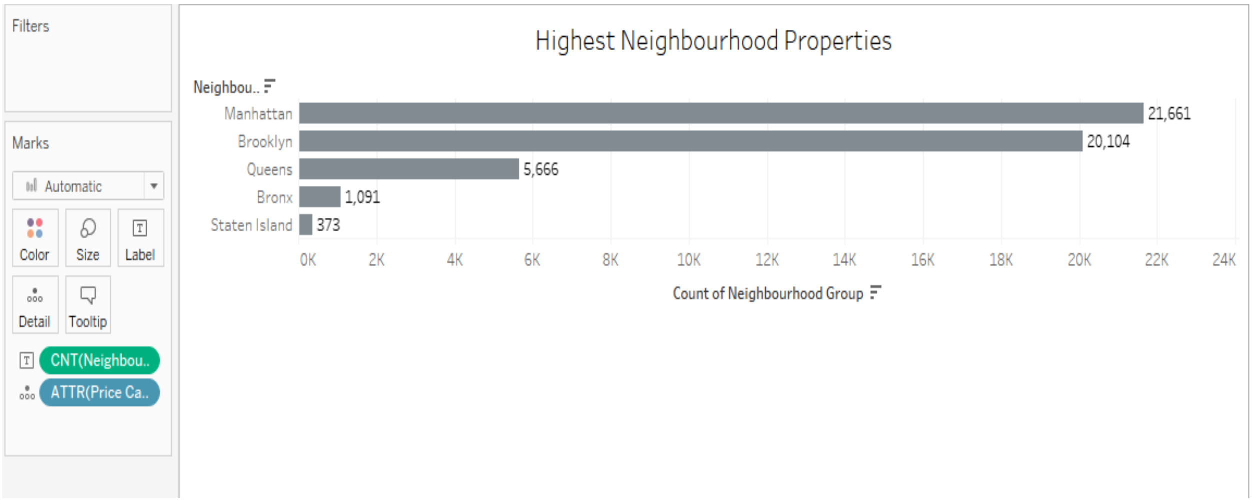
```
data.plot(kind='scatter', x='longitude', y='latitude')
plt.show()
```

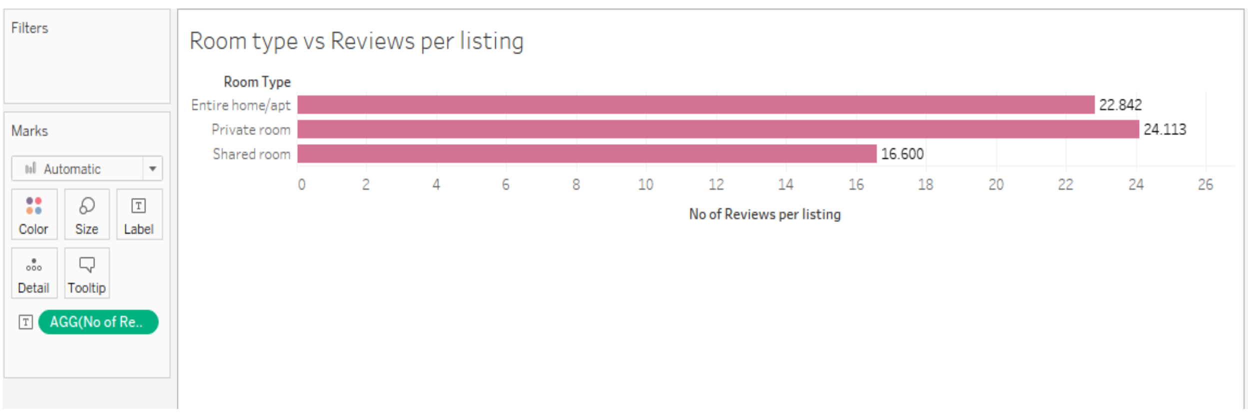


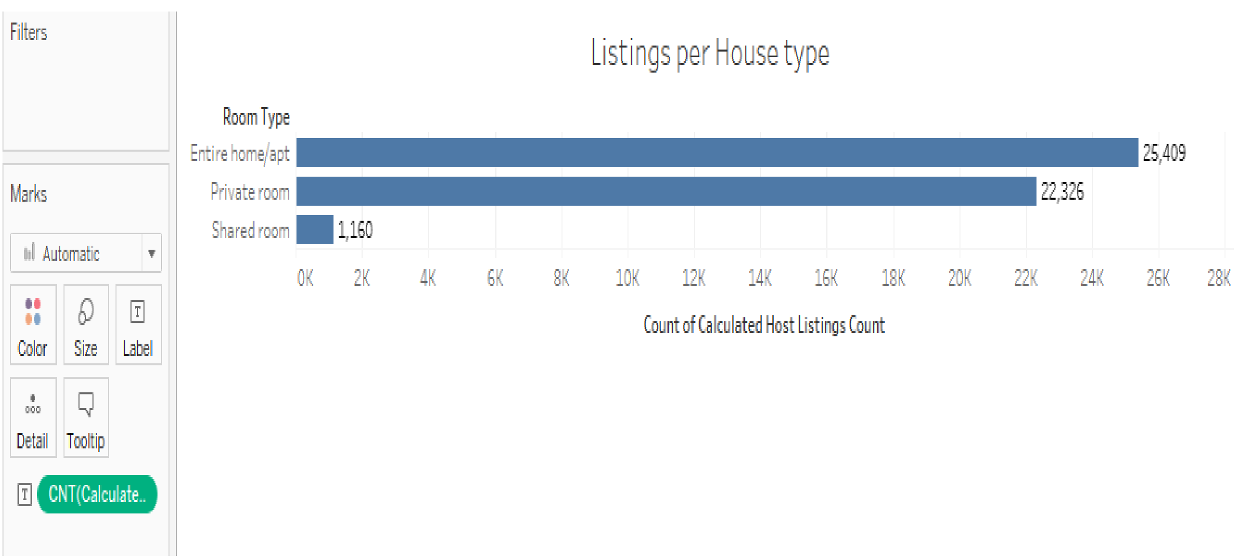

```
sns.countplot(x = 'price_categories', data = data)
plt.show()
```

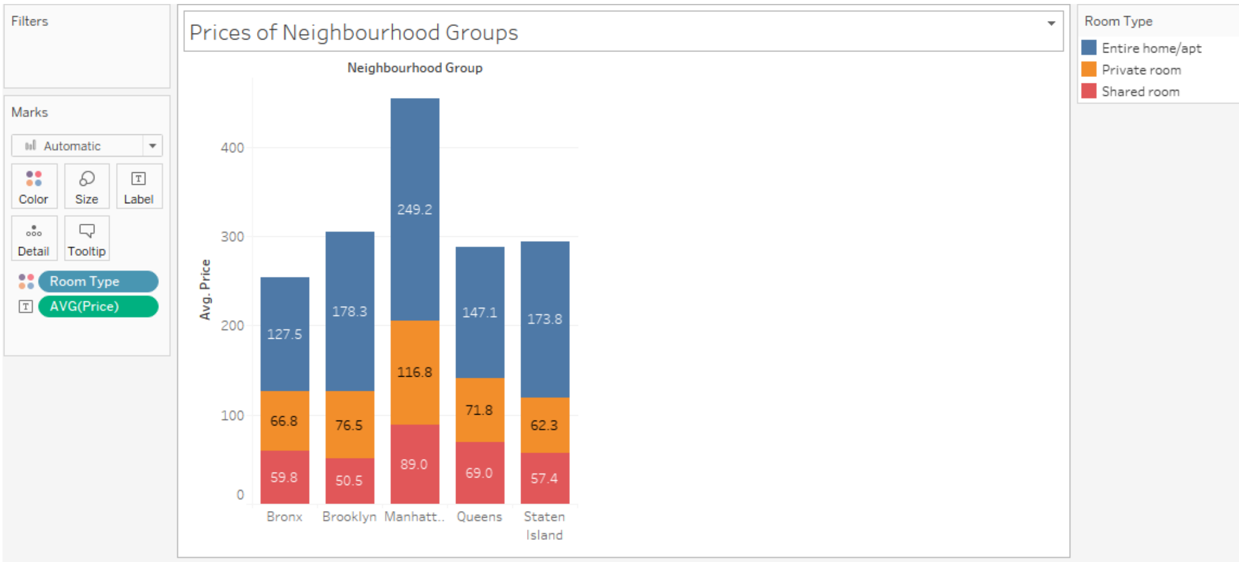


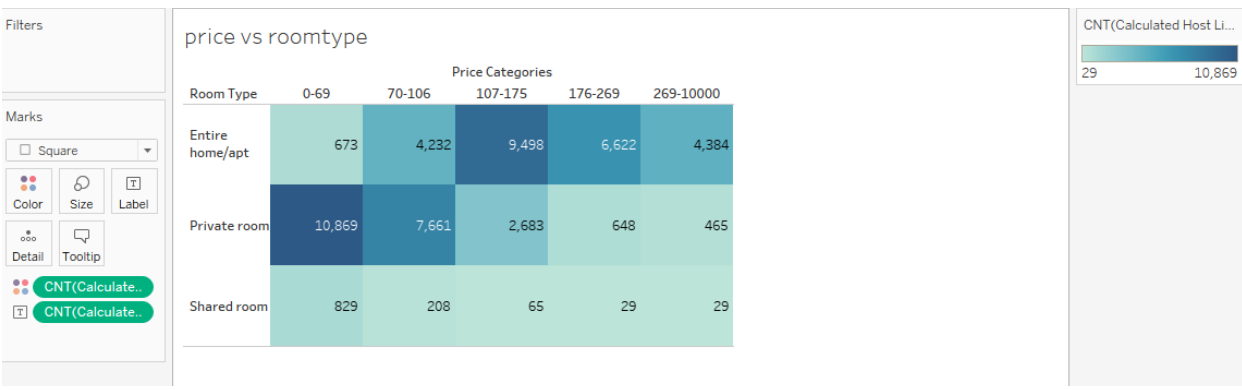
Observed some visualizations in Tableau



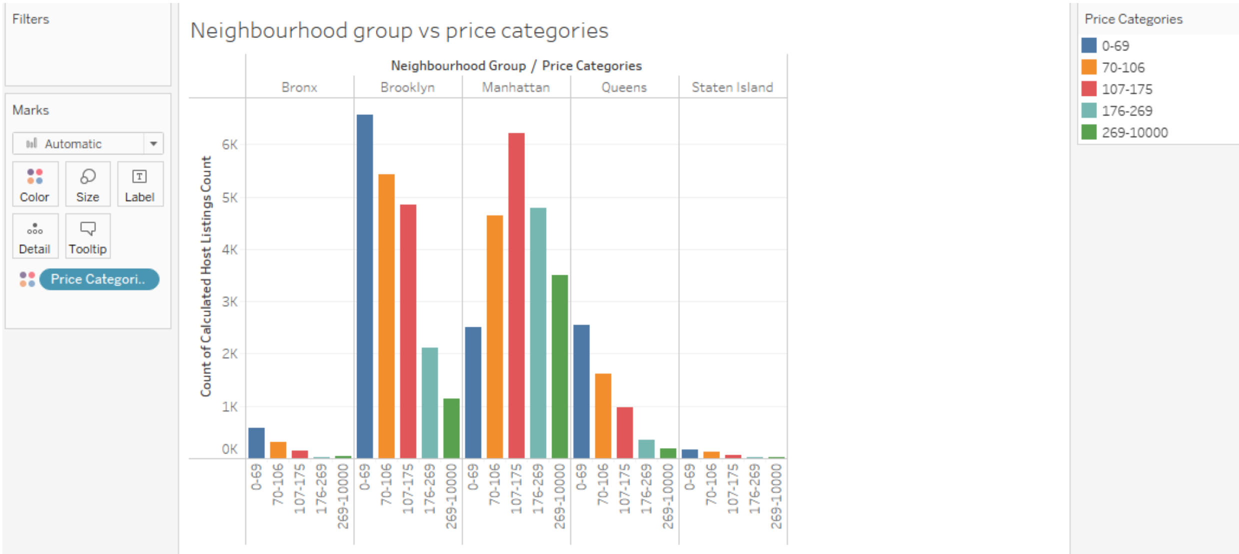




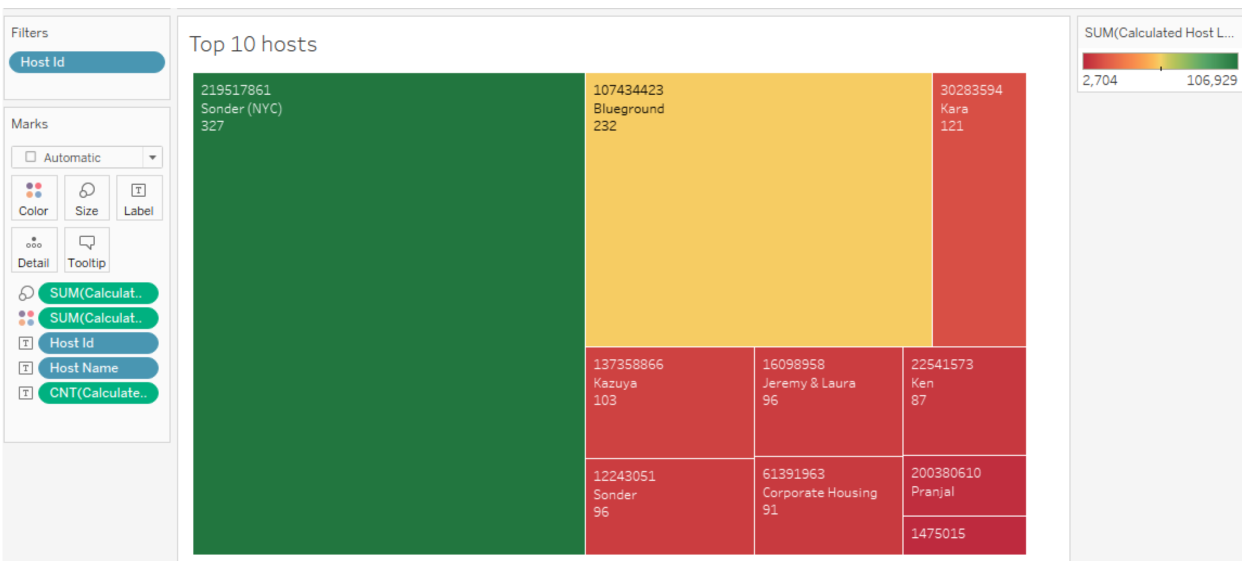

Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Room Type
AVG(Price)
Prices of Neighbourhood Groups
Neighbourhood Group
Avg. Price
400
300
200
100
0
127.5
66.8
59.8
178.3
76.5
50.5
249.2
116.8
89.0
147.1
71.8
69.0
173.8
62.3
57.4
Bronx
Brooklyn
Manhatt..
Queens
Staten Island
Room Type
Entire home/apt
Private room
Shared room


Filters
Marks
Square
Color
Size
Label
Detail
Tooltip
CNT(Calculate..
CNT(Calculate..
price vs roomtype
Price Categories
Room Type
0-69
70-106
107-175
176-269
269-10000
Entire home/apt
673
4,232
9,498
6,622
4,384
Private room
10,869
7,661
2,683
648
465
Shared room
829
208
65
29
29
CNT(Calculated Host Li...
29
10,869

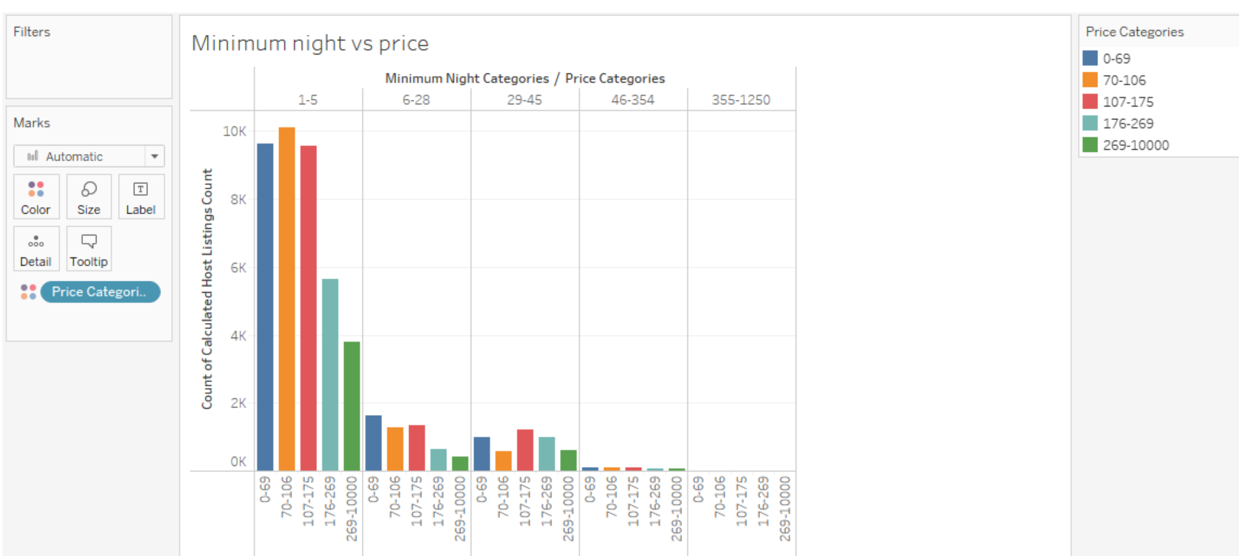

Filters
Marks
Circle
Color
Size
Label
Detail
Tooltip
CNT(Calculate..
Room Type
Neighbourhoo..
AGG(No of Re..
CNT(Calculate..
AGG(No of Re..
neighbourhood wise business
Private room
Brooklyn
21.09
10,132
Private room
Manhattan
26.20
7,982
Private room
Queens
Entire home/apt
Entire home/apt
Brooklyn
27.95
9,559
Entire home/apt
Manhattan
17.82
13,199
AGG(No of Reviews per...
1.56
33.28



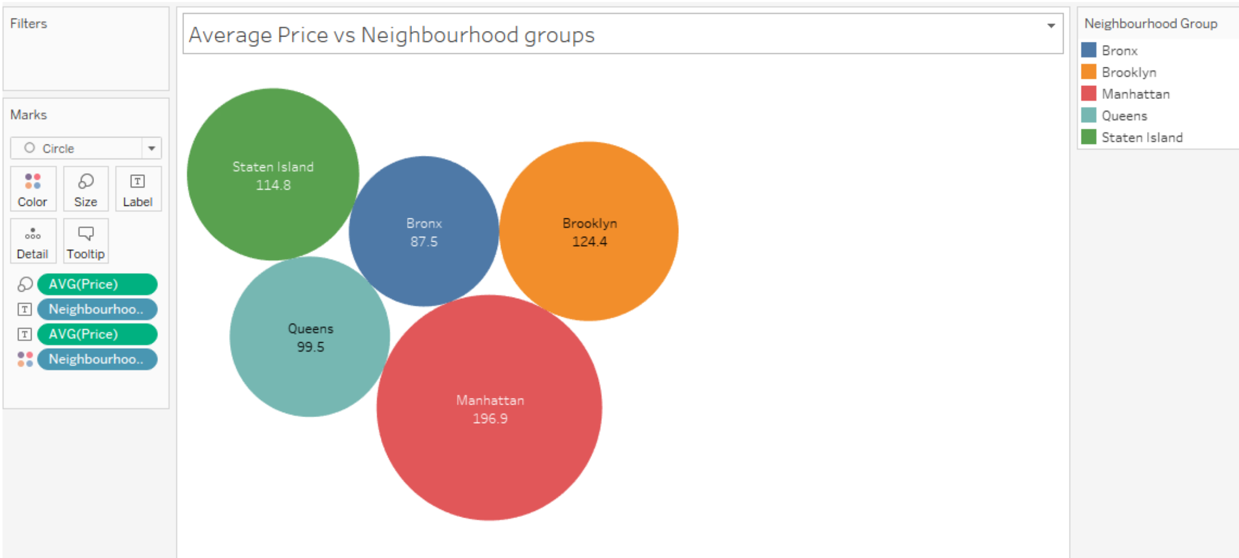
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Price Categori..
Neighbourhood group vs price categories
Neighbourhood Group / Price Categories
Bronx
Brooklyn
Manhattan
Queens
Staten Island
Count of Calculated Host Listings Count
6K
5K
4K
3K
2K
1K
0K
0-69
70-106
107-175
176-269
269-10000
Price Categories
0-69
70-106
107-175
176-269
269-10000



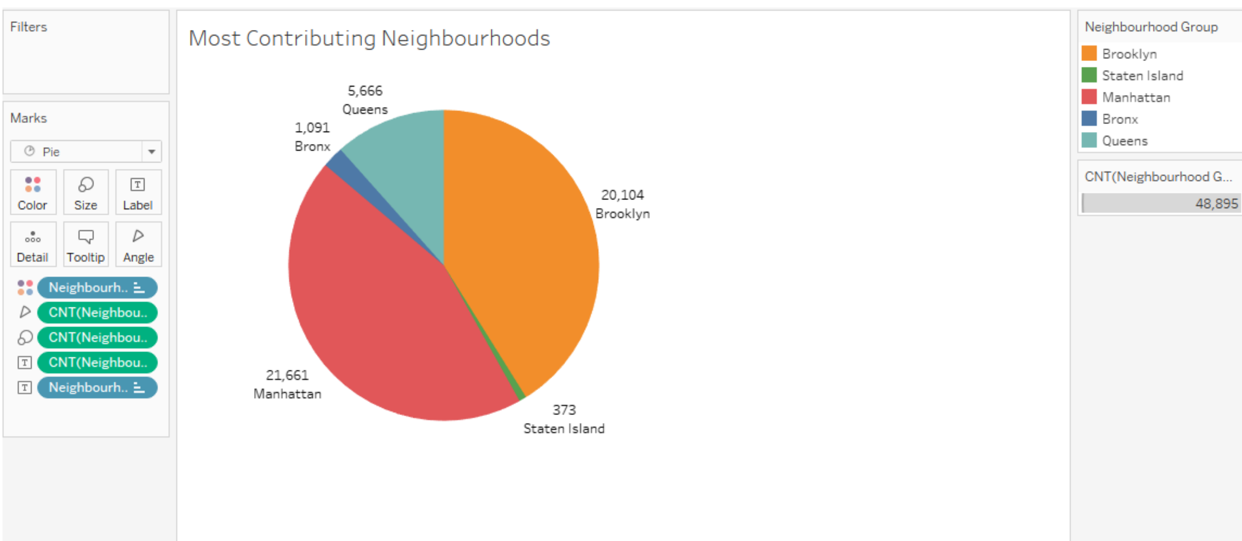
Filters
Host Id
Marks
Automatic
Color
Size
Label
Detail
Tooltip
SUM(Calculat..
SUM(Calculat..
Host Id
Host Name
CNT(Calculate..
Top 10 hosts
219517861
Sonder (NYC)
327
107434423
Blueground
232
30283594
Kara
121
137358866
Kazuya
103
16098958
Jeremy & Laura
96
22541573
Ken
87
12243051
Sonder
96
61391963
Corporate Housing
91
200380610
Pranjal
1475015
SUM(Calculated Host L...
2,704
106,929



Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Price Categori..
Minimum night vs price
Minimum Night Categories / Price Categories
1-5
6-28
29-45
46-354
355-1250
Count of Calculated Host Listings Count
10K
8K
6K
4K
2K
0K
0-69
70-106
107-175
176-269
269-10000
Price Categories
0-69
70-106
107-175
176-269
269-10000



Filters
Marks
Circle
Color
Size
Label
Detail
Tooltip
AVG(Price)
Neighbourhoo..
AVG(Price)
Neighbourhoo..
Average Price vs Neighbourhood groups
Staten Island
114.8
Bronx
87.5
Brooklyn
124.4
Queens
99.5
Manhattan
196.9
Neighbourhood Group
Bronx
Brooklyn
Manhattan
Queens
Staten Island



Filters
Marks
Pie
Color
Size
Label
Detail
Tooltip
Angle
Neighbourh..
CNT(Neighbou..
CNT(Neighbou..
CNT(Neighbou..
Neighbourh..
Most Contributing Neighbourhoods
5,666
Queens
1,091
Bronx
20,104
Brooklyn
21,661
Manhattan
373
Staten Island
Neighbourhood Group
Brooklyn
Staten Island
Manhattan
Bronx
Queens
CNT(Neighbourhood G...
48,895