EDA on Airbnb NYC 2019

Airbnb is an online marketplace that connects people who want to rent out their homes with people looking for accommodations in that locale. NYC is the most populous city in the United States, and one of the most popular tourism and business places globally. Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present a more unique, personalized way of experiencing the world. Nowadays, Airbnb became one of a kind service that is used by the whole world. Data analysts become a crucial factor for the company that provided millions of listings through Airbnb. These listings generate a lot of data that can be analyzed and used for security, business decisions, understanding of customers' and providers' behavior on the platform, implementing innovative additional services, guiding marketing initiatives, and much more.

Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present a more unique, personalized way of experiencing the world. Today, Airbnb became one of a kind service that is used and recognized by the whole world. Data analysis on millions of listings provided through Airbnb is a crucial factor for the company. These millions of listings generate a lot of data - data that can be analyzed and used for security, business decisions, understanding of customers' and providers' (hosts) behavior and performance on the platform, guiding marketing initiatives, implementation of innovative additional services and much more. This dataset has around 49,000 observations in it with 16 columns and it is a mix between categorical and numeric values

Importing important packages

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:

```
data = pd.read_csv('https://raw.githubusercontent.com/rahulinchal/Airbnb-NYC-2019/main/A
data.head()
```

Out[2]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latit
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79

Data Description

Id : Unique for each Propety Listing.
name : Name of the each Propety Listing.
host_id : Unique ID for host who have listed the property on Airbnb.
host_name : Name of host
neighbourhood_group : Name of Each boroughs of NYC, Manhattan, Brooklyn,Queens,Bronx, State Island.
neighbourhood : Area in each borough of NYC
latitude, longitude : Co-ordinates of each listed property

room_type : Differnt types of room available for listing , Private room,Entire home/apt,Shared room.
price : Price of listing.
minimum_nigths : Mandatory number of nights to be booked for available for each type of property.
number_of_review : Number of reviews for each Listed property
last_review : Date on whcih last time the listing was reviewed
review_per_month : Number of reviews per month
calculated_host_listings_count : Number of listing each host owns
availablity_365 : Number of days the given listing is available for booking

In [3]:

```
# Copying the data
df = data.copy()
```

In [4]:

```
# Getting the info
df.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
```

What is Describe ?

The describe() method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these information for each column: count - The number of not-empty values. mean - The average (mean) value. std - The standard deviation.

FInding the duplicated values

In [5]:

```
df.describe().T.style.background_gradient()
```

Out[5]:

	count	mean	std	min
id	48895.000000	19017143.236180	10983108.385610	2539.000000
host_id	48895.000000	67620010.646610	78610967.032667	2438.000000
latitude	48895.000000	40.728949	0.054530	40.499790
longitude	48895.000000	-73.952170	0.046157	-74.244420
price	48895.000000	152.720687	240.154170	0.000000
minimum_nights	48895.000000	7.029962	20.510550	1.000000
number_of_reviews	48895.000000	23.274466	44.550582	0.000000
reviews_per_month	38843.000000	1.373221	1.680442	0.010000
calculated_host_listings_count	48895.000000	7.143982	32.952519	1.000000
availability_365	48895.000000	112.781327	131.622289	0.000000

In [6]:

```
df.duplicated().sum()
```

Out[6]:

0

There are no duplicated values

In [7]:

```
df.shape
```

Out[7]:

(48895, 16)

In [8]:

```
df.head()
```

Out[8]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latit
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
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
```

Replacing the null values with appropriate values

In [10]:

```
df['name'].replace(np.nan, 'Other Hotel', inplace =True)
df['host_name'].replace(np.nan, 'other', inplace = True)
df['last_review'].replace(np.nan, 'Not Reviewed', inplace = True)
df['reviews_per_month'].replace(np.nan, '0', inplace = True)
```

In [11]:

```
df.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                     48895 non-null  object 
 13  reviews_per_month               48895 non-null  object 
 14  calculated_host_listings_count  48895 non-null  int64  
 15  availability_365                48895 non-null  int64  
dtypes: float64(2), int64(7), object(7)
memory usage: 6.0+ MB
```

We dont need id and last_reviewed as it is completely unique and has no significance for visualization

In [12]:

```
df.drop(['id', 'last_review'], axis = 1, inplace = True)
```

Also dropping the name column as it signifies nothing.

In [13]:

```
df.drop(['name'], axis = 1, inplace = True)
```

In [14]:

```
df.isnull().sum()
```

Out[14]:

```
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
```

Hence the data is free from duplicates and null values and is ready for visualization.

Data Visulaization.

1. Top 10 Host name

In [15]:

```
df.head()
```

Out[15]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_t
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv rc
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home,
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv rc
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home,
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home,

In [16]:

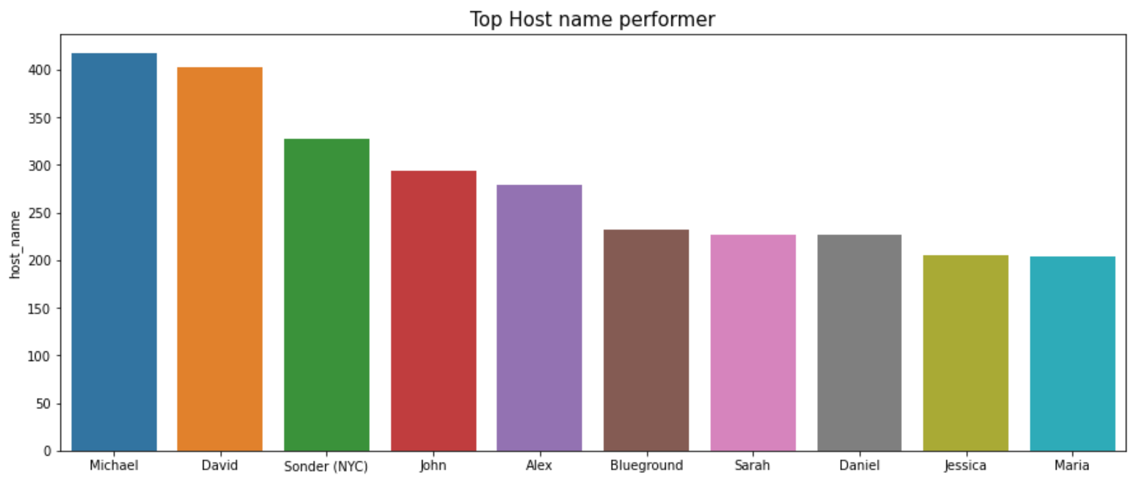
```
# Getting the value counts
df['host_name'].value_counts().iloc[:10]
```

Out[16]:

```
Michael         417
David           403
Sonder (NYC)    327
John            294
Alex            279
Blueground      232
Sarah           227
Daniel          226
Jessica         205
Maria           204
Name: host_name, dtype: int64
```

In [17]:

```
# Visualizing using bar plot
plt.figure(figsize = (15,6))
sns.barplot(data = df, x = df['host_name'].value_counts().iloc[:10].keys(),
           y = df['host_name'].value_counts().iloc[:10])
plt.title("Top Host name performer", fontsize = 15)
plt.show()
```



Observation

1. Host name is the name of the host who listed the hotel in the airbnb.
2. It looks like the persom Michael has the lasrgest booking under his name with 417 bookings
3. David is the host name with 403 bookings.

2. Neighbourhood_group.

Neighbourhood_group - Name of Each boroughs of NYC, Manhattan, Brooklyn,Queens,Bronx, State Island

In [18]:

```
df.head()
```

Out[18]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_ty
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv ro
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	En home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv ro
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	En home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	En home/

In [19]:

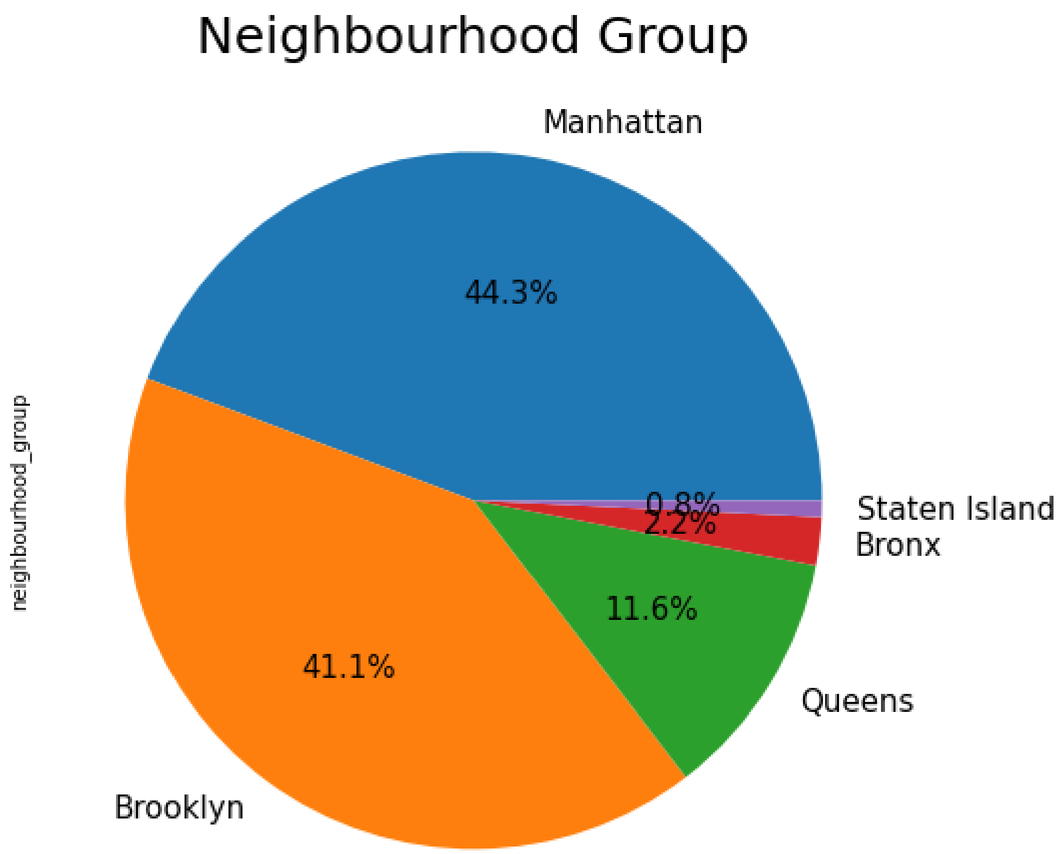
```
# Getting the vlaue count
df['neighbourhood_group'].value_counts()
```

Out[19]:

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

In [20]:

```
# Visualizing using pie chart
df['neighbourhood_group'].value_counts().plot(kind = 'pie', figsize = (8,8), autopct = '
plt.title("Neighbourhood Group", fontsize = 25)
plt.show()
```



Observation

1. **neighbourhood_group** : Name of Each boroughs of NYC, Manhattan, Brooklyn,Queens,Bronx, State Island.
2. It looks like Manhatten group has the largest bookings
3. Followed by brooklyn with 41.1% share.

4. Finding the top 10 host_id

Host_id - Unique ID for host who have listed the property on Airbnb.

In [21]:

```
df.head()
```

Out[21]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_t
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv rc
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv rc
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home/

In [22]:

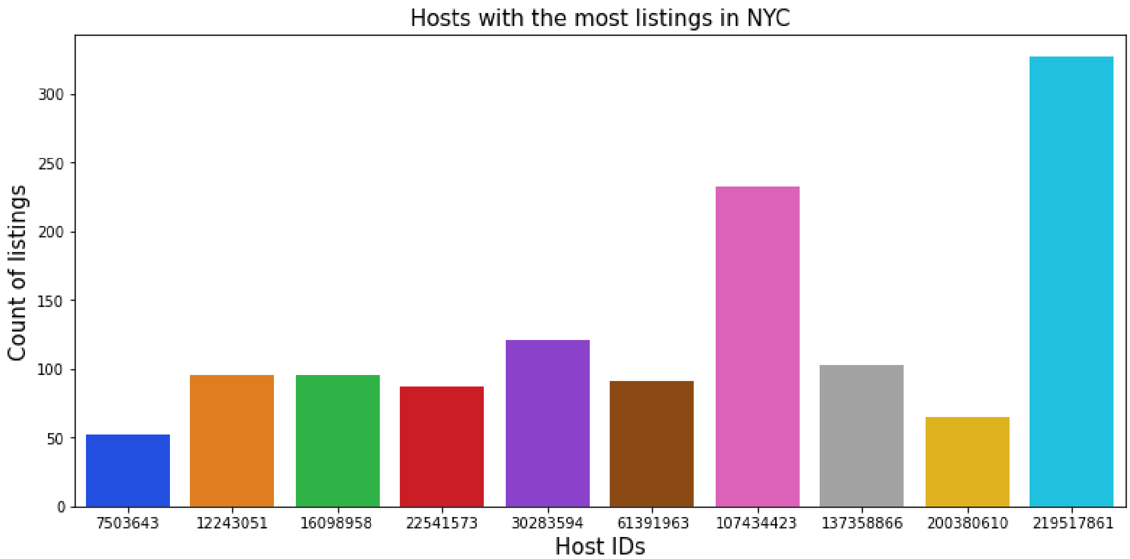
```
# Finding the value count
df['host_id'].value_counts().reset_index().iloc[:10]
```

Out[22]:

	index	host_id
0	219517861	327
1	107434423	232
2	30283594	121
3	137358866	103
4	16098958	96
5	12243051	96
6	61391963	91
7	22541573	87
8	200380610	65
9	7503643	52

In [23]:

```
# Plotting the bar graph for that
plt.figure(figsize = (13,6))
sns.barplot(x=df['host_id'].value_counts().iloc[:10].keys(), y=df['host_id'].value_count
                 palette='bright')
plt.title("Hosts with the most listings in NYC", fontsize = 15)
plt.xlabel("Host IDs", fontsize = 15)
plt.ylabel("Count of listings", fontsize = 15)
plt.show()
```



Observation

1. We can see that there is a good distribution between top 10 hosts with the most listings.
2. First host has more than 300+ listings.

5. Neighbourhood_group according to price

In [24]:

```
df.head()
```

Out[24]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_ty
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv ro
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv ro
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home/

In [25]:

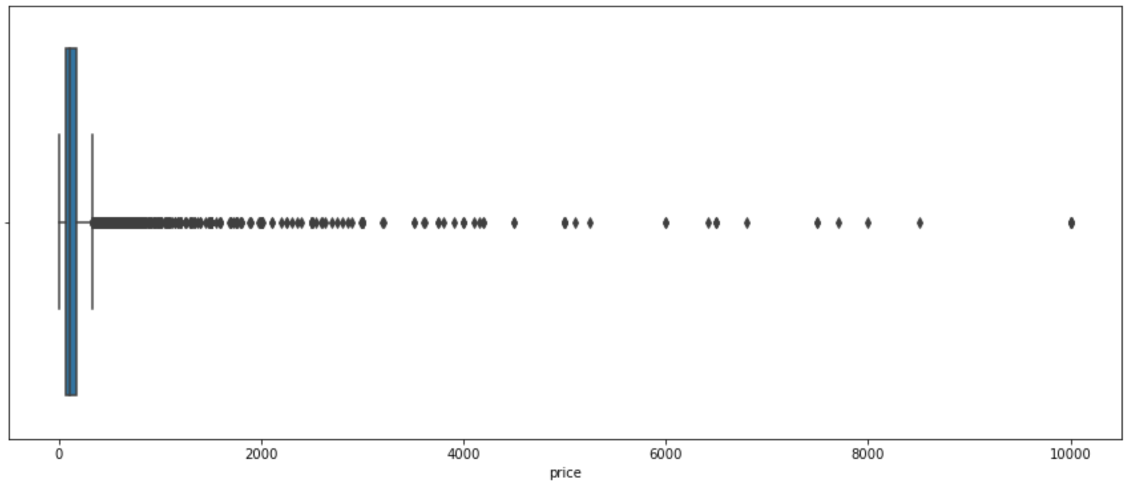
```
df.shape
```

Out[25]:

```
(48895, 13)
```

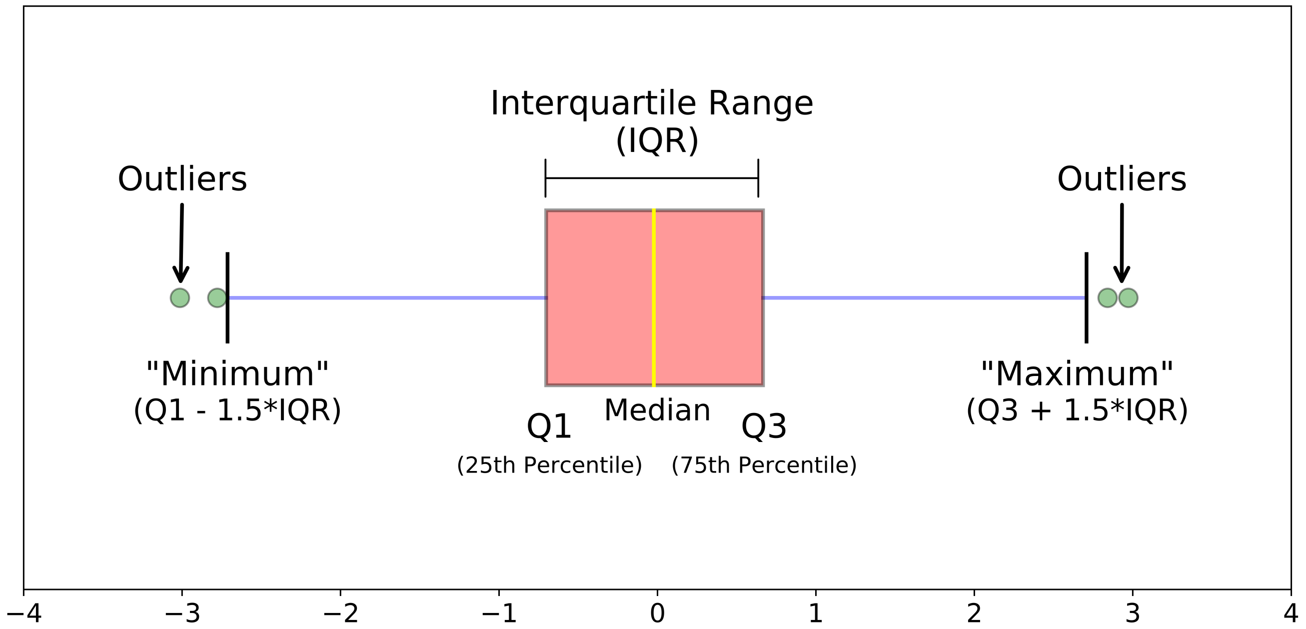
In [26]:

```
plt.figure(figsize = (15,6))
sns.boxplot(x = df['price'])
plt.show()
```



Boxplot -

A box plot is a chart that shows data from a five-number summary including one of the measures of central tendency. It does not show the distribution in particular as much as a stem and leaf plot or histogram does. But it is primarily used to indicate a distribution is skewed or not and if there are potential unusual observations (also called outliers) present in the data set. Boxplots are also very beneficial when large numbers of data sets are involved or compared.



In [27]:

```
# Getting the mathematical answers for the price column
df['price'].describe()
```

Out[27]:

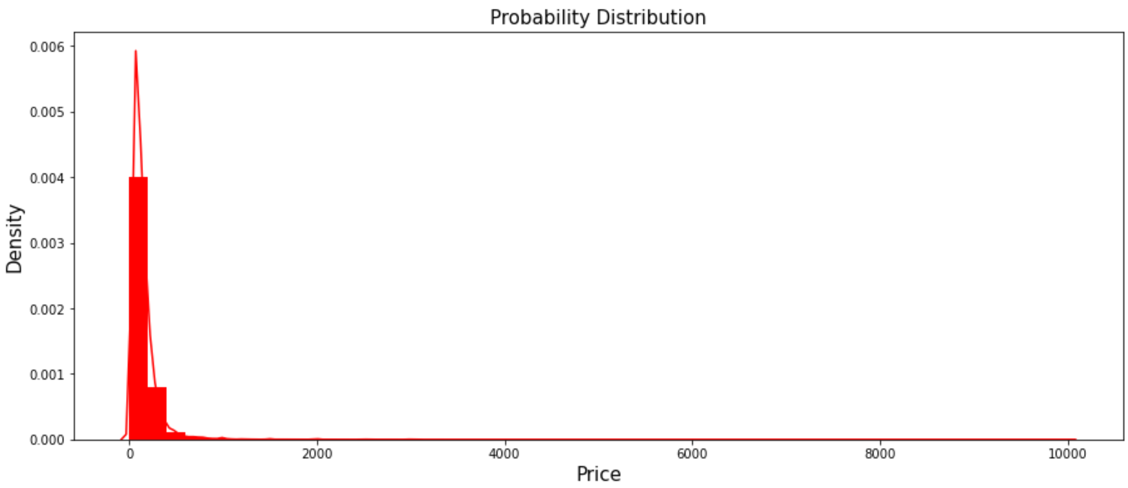
```
count    48895.000000
mean       152.720687
std        240.154170
min          0.000000
25%         69.000000
50%        106.000000
75%        175.000000
max      10000.000000
Name: price, dtype: float64
```

Probability density Function graph

A function that defines the relationship between a random variable and its probability, such that you can find the probability of the variable using the function, is called a Probability Density Function (PDF) in statistics.

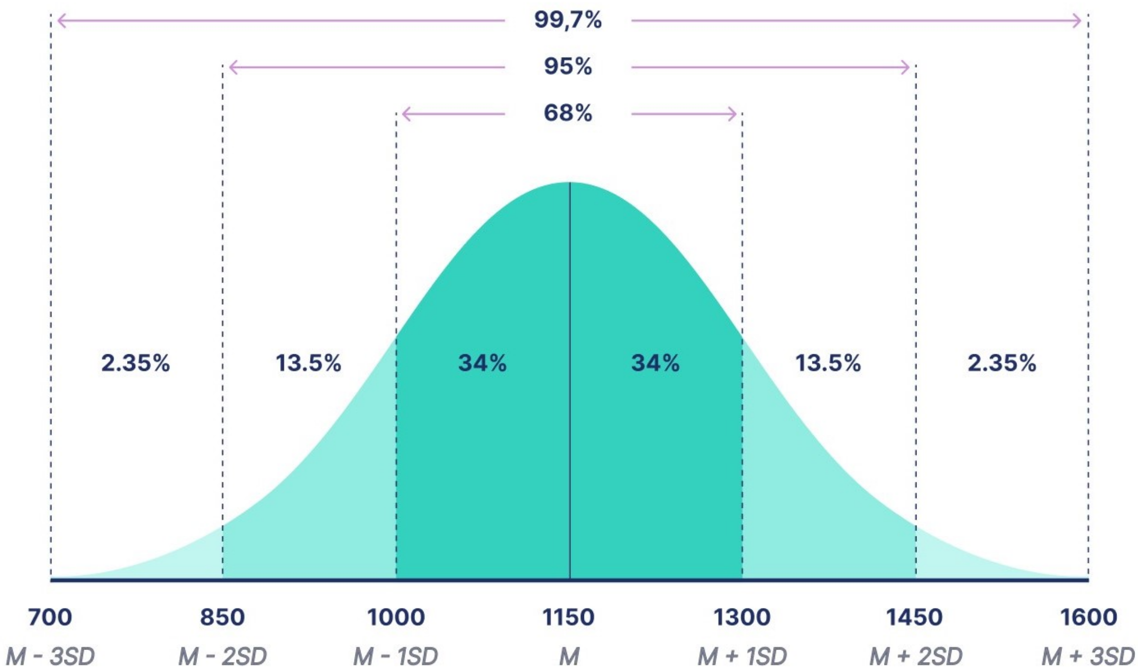
In [28]:

```
plt.figure(figsize = (15,6))
sns.distplot(df['price'], color = 'red', hist_kws={"linewidth": 15,'alpha':1})
plt.title("Probability Distribution", fontsize = 15)
plt.xlabel('Price', fontsize = 15)
plt.ylabel('Density', fontsize = 15)
plt.show()
```



Normal Distribution

A normal distribution is a type of continuous probability distribution in which most data points cluster toward the middle of the range, while the rest taper off symmetrically toward either extreme. The middle of the range is also known as the mean of the distribution.



Calculating the interquartile ranges

In [29]:

```
Q1 = np.percentile(data['price'], 25, interpolation = 'midpoint')

# Third quartile (Q3)
Q3 = np.percentile(data['price'], 75, interpolation = 'midpoint')

# Interquaritle range (IQR)
IQR = Q3 - Q1

print('The IQR is',IQR)
print('The Minimum value is', (Q3 - (1.5* (IQR))))
print('The maximum value is', (Q3 + (1.5* (IQR))))
```

```
The IQR is 106.0
The Minimum value is 16.0
The maximum value is 334.0
```

As we can see that 99% of the data lies withing 334 dollar with mean being 153 and median 106.

In [30]:

```
df_new = df[df['price'] < 334 ]
df_new.head()
```

Out[30]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_t
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv rc
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv rc
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home/

In [31]:

```
df.groupby(['neighbourhood_group'])['price'].describe().T.reset_index()
```

Out[31]:

neighbourhood_group	index	Bronx	Brooklyn	Manhattan	Queens	
0	count	1091.000000	20104.000000	21661.000000	5666.000000	373.0
1	mean	87.496792	124.383207	196.875814	99.517649	114.8
2	std	106.709349	186.873538	291.383183	167.102155	277.6
3	min	0.000000	0.000000	0.000000	10.000000	13.0
4	25%	45.000000	60.000000	95.000000	50.000000	50.0
5	50%	65.000000	90.000000	150.000000	75.000000	75.0
6	75%	99.000000	150.000000	220.000000	110.000000	110.0
7	max	2500.000000	10000.000000	10000.000000	10000.000000	5000.0

In [32]:

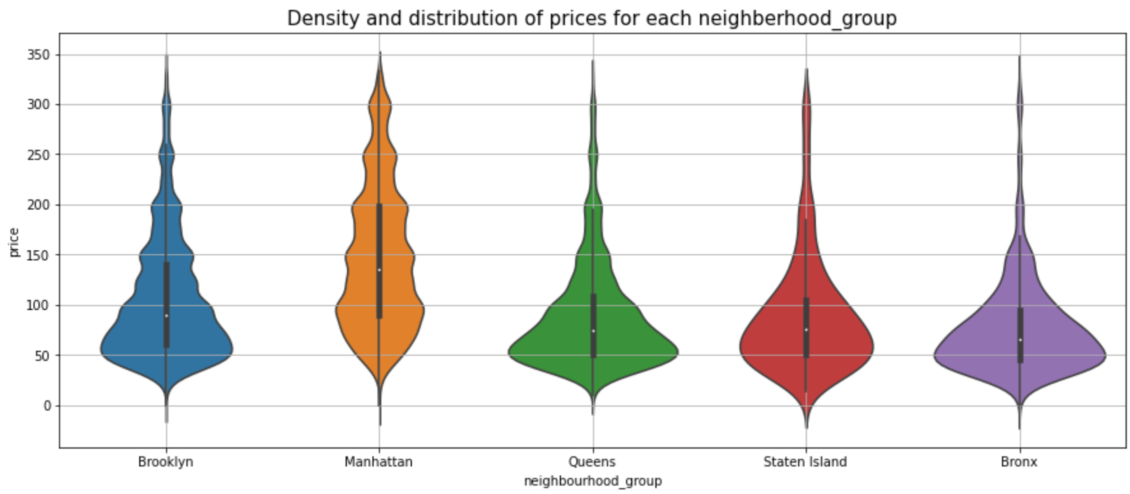
```
df_new.groupby(['neighbourhood_group'])['price'].describe().T.reset_index()
```

Out[32]:

neighbourhood_group	index	Bronx	Brooklyn	Manhattan	Queens	Sta Isl
0	count	1070.000000	19415.000000	19501.000000	5567.000000	365.000
1	mean	77.365421	105.699614	145.904620	88.904437	89.235
2	std	47.110940	60.937808	70.417743	53.536041	57.700
3	min	0.000000	0.000000	0.000000	10.000000	13.000
4	25%	45.000000	60.000000	90.000000	50.000000	50.000
5	50%	65.000000	90.000000	135.000000	74.000000	75.000
6	75%	95.000000	140.000000	199.000000	108.000000	105.000
7	max	325.000000	333.000000	333.000000	325.000000	300.000

In [33]:

```
plt.figure(figsize = (15,6))
sns.violinplot(data = df_new, x = df_new['neighbourhood_group'], y = df_new['price'])
plt.title('Density and distribution of prices for each neighberhood_group', fontsize = 1
plt.grid()
```



Violin Plot

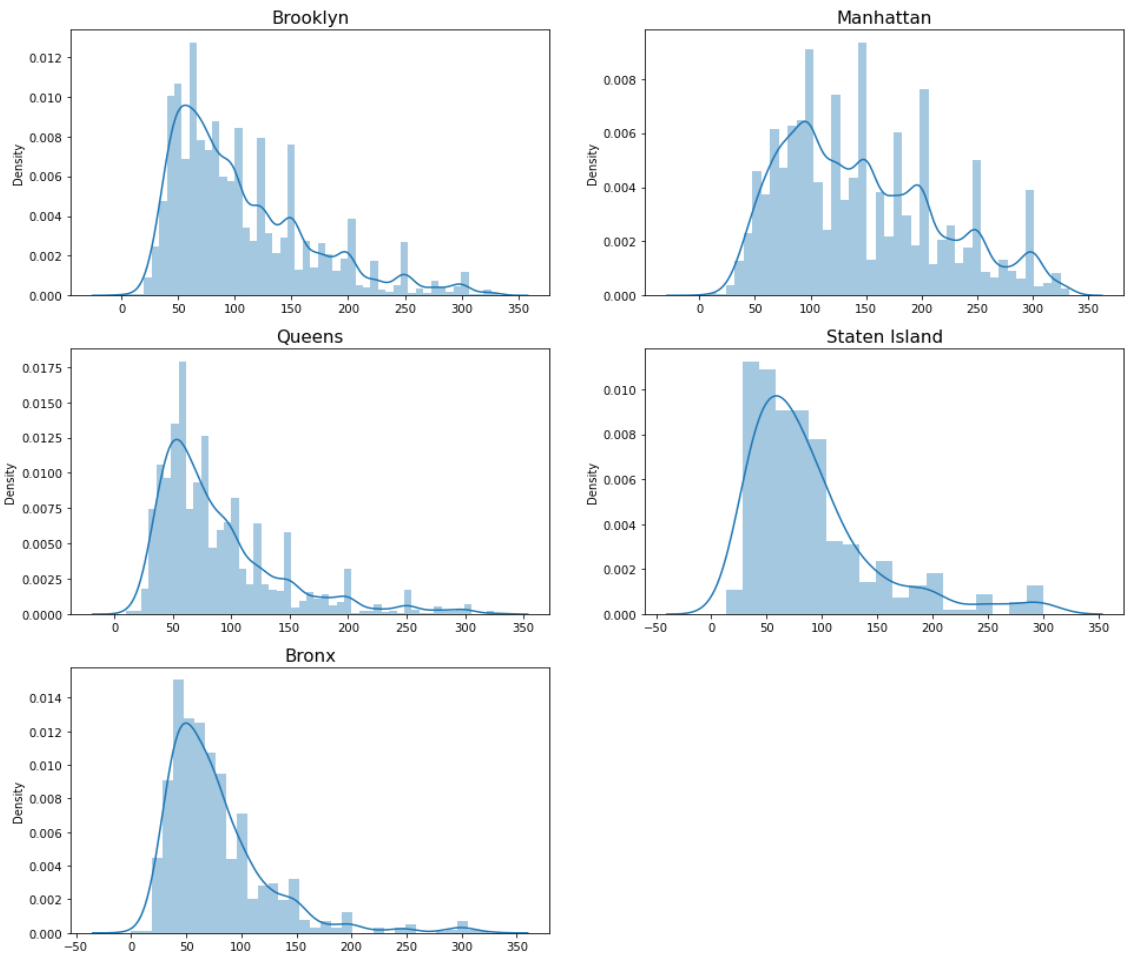
A violin plot is a hybrid of a box plot and a kernel density plot, which shows peaks in the data. It is used to visualize the distribution of numerical data. Unlike a box plot that can only show summary statistics, violin plots depict summary statistics and the density of each variable.

In [34]:

```
plt.figure(figsize = (16,15))

plt.subplot(3,2,1)
n1 = df_new[df_new['neighbourhood_group'] == 'Brooklyn']
sns.distplot(x = n1['price'])
plt.title("Brooklyn", fontsize = 15)

plt.subplot(3,2,2)
n2 = df_new[df_new['neighbourhood_group'] == 'Manhattan']
sns.distplot(x = n2['price'])
plt.title("Manhattan", fontsize = 15)

plt.subplot(3,2,3)
n3 = df_new[df_new['neighbourhood_group'] == 'Queens']
sns.distplot(x = n3['price'])
plt.title("Queens", fontsize = 15)

plt.subplot(3,2,4)
n4 = df_new[df_new['neighbourhood_group'] == 'Staten Island']
sns.distplot(x = n4['price'])
plt.title("Staten Island", fontsize = 15)

plt.subplot(3,2,5)
n5 = df_new[df_new['neighbourhood_group'] == 'Bronx']
sns.distplot(x = n5['price'])
plt.title("Bronx", fontsize = 15)
```

Out[34]:

```
Text(0.5, 1.0, 'Bronx')
```



Histplot

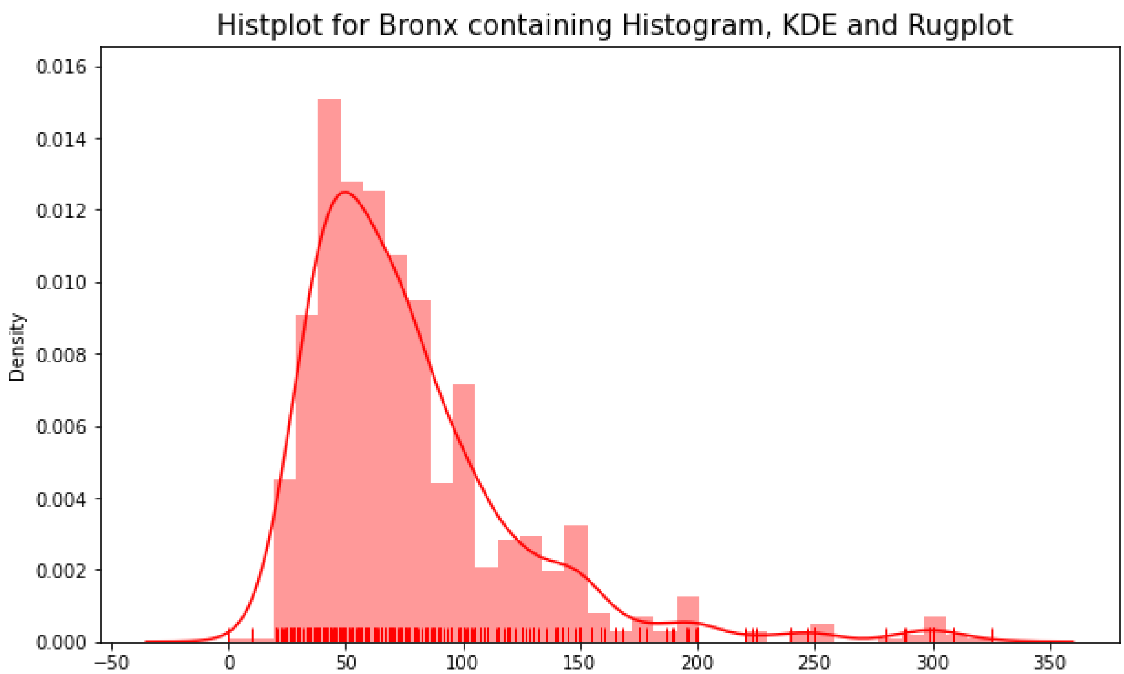
Histplot is a combination of 3 plots.
1. Histogram - A histogram is a bar graph-like representation of data that buckets a range of classes into columns along the horizontal x-axis. The vertical y-axis represents the number count or percentage of occurrences in the data for each column. Columns can be used to visualize patterns of data distributions.

2. KDE - Kernal Density Estimation - Kdeplot is a Kernel Distribution Estimation Plot which depicts the probability density function of the continuous or non-parametric data variables i.e. we can plot for the univariate or multiple variables altogether.

3. Rugplot - A rug plot is a plot of data for a single quantitative variable, displayed as marks along an axis. It is used to visualise the distribution of the data. As such it is analogous to a histogram with zero-width bins, or a one-dimensional scatter plot.

In [35]:

```
plt.figure(figsize = (10,6))
sns.distplot(x = n5['price'],  rug = True, color ='r')
plt.title(" Histplot for Bronx containing Histogram, KDE and Rugplot", fontsize = 15)
plt.show()
```



Observation

1. we can observe that state that Manhattan has the highest range of prices for the listings with 150 price as median observation, followed by Brooklyn with 90 per night.
2. Queens and Staten Island appear to have very similar distributions, Bronx is the cheapest of them all.

6. Room type

In [36]:

```
df.head()
```

Out[36]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_t
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv rc
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv rc
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home/

In [37]:

```
# Getting the value counts
df['room_type'].value_counts()
```

Out[37]:

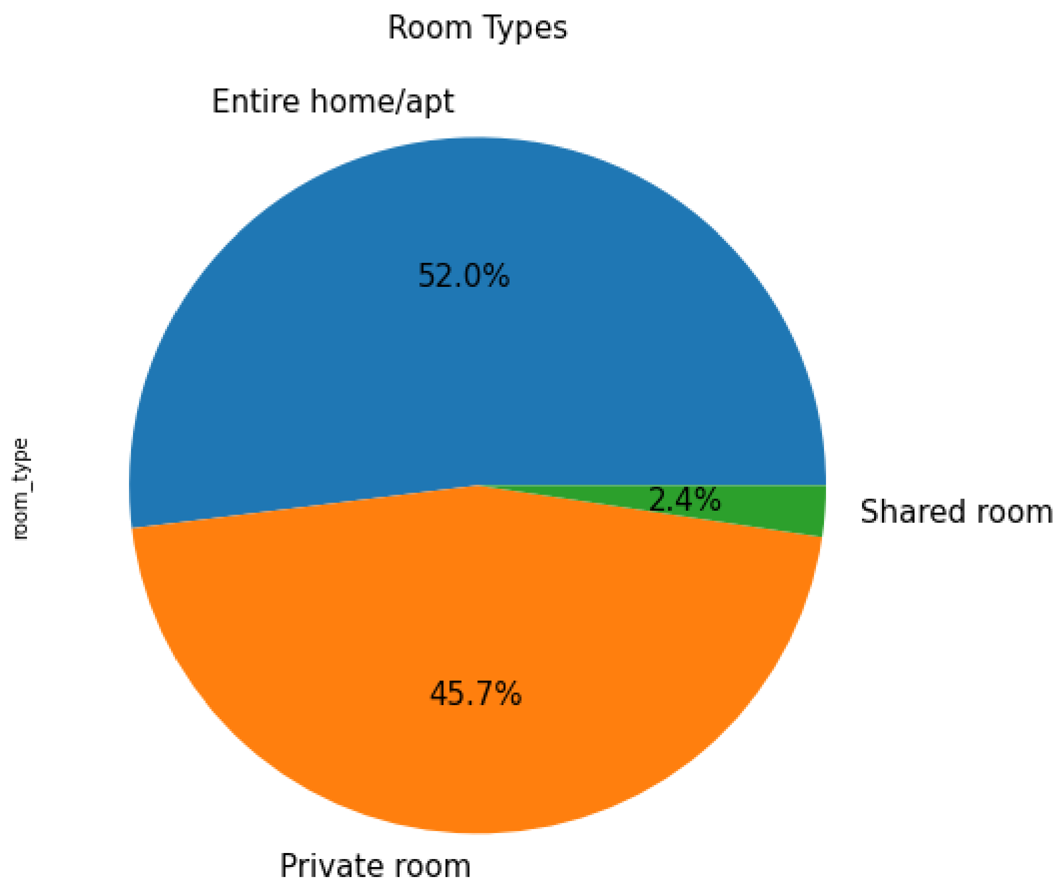
```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

In [38]:

```
# Visualizing using pie chart
df['room_type'].value_counts().plot(kind = 'pie', figsize = (8,8), fontsize = 15, autopc
plt.title("Room Types", fontsize = 15)
```

Out[38]:

```
Text(0.5, 1.0, 'Room Types')
```



Observation

1. Most of the people happen to rent the entire home or apartment which constitutes to 52% according to the chart.
2. Followed by 45.7% people consider having private room, and shared is the least considered room type.

7. Prices for different room type

In [39]:

```
df.head()
```

Out[39]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_t
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv rc
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv rc
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home/

In [40]:

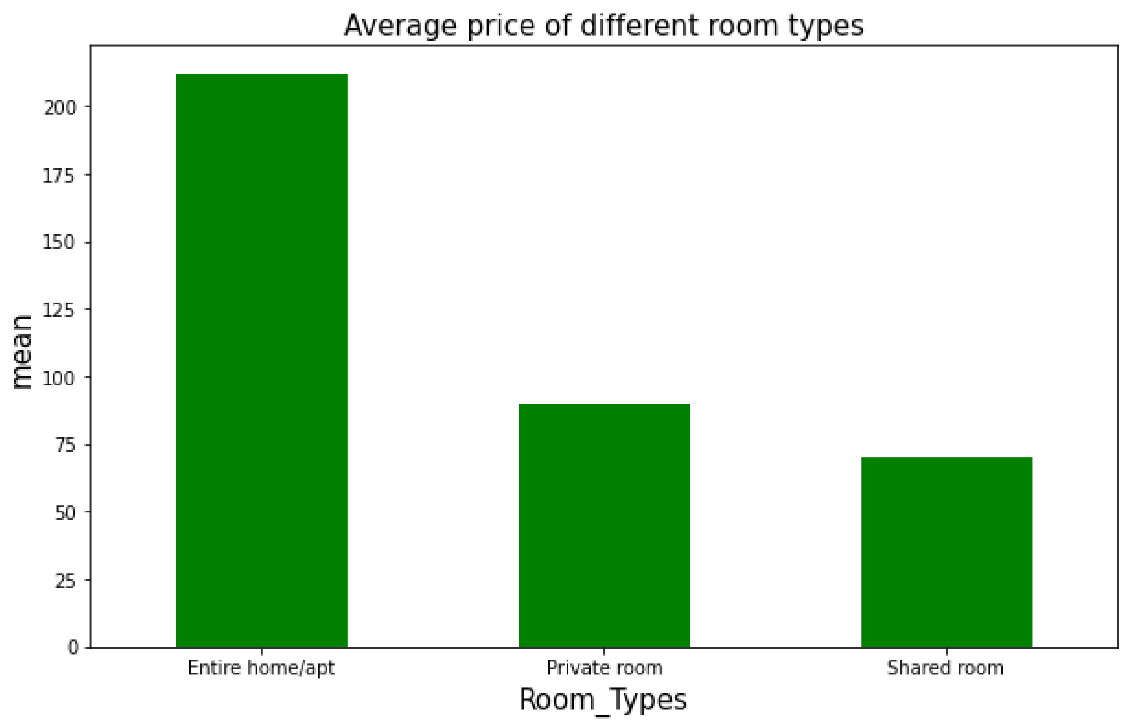
```
df.groupby(['room_type'])['price'].mean().reset_index()
```

Out[40]:

	room_type	price
0	Entire home/apt	211.794246
1	Private room	89.780973
2	Shared room	70.127586

In [41]:

```
df.groupby(['room_type'])['price'].mean().plot(kind = 'bar', figsize = (10,6), color = '
plt.xticks( rotation = 360)
plt.title("Average price of different room types", fontsize = 15)
plt.xlabel('Room_Types', fontsize = 15)
plt.ylabel('mean', fontsize = 15)
plt.show()
```



In [42]:

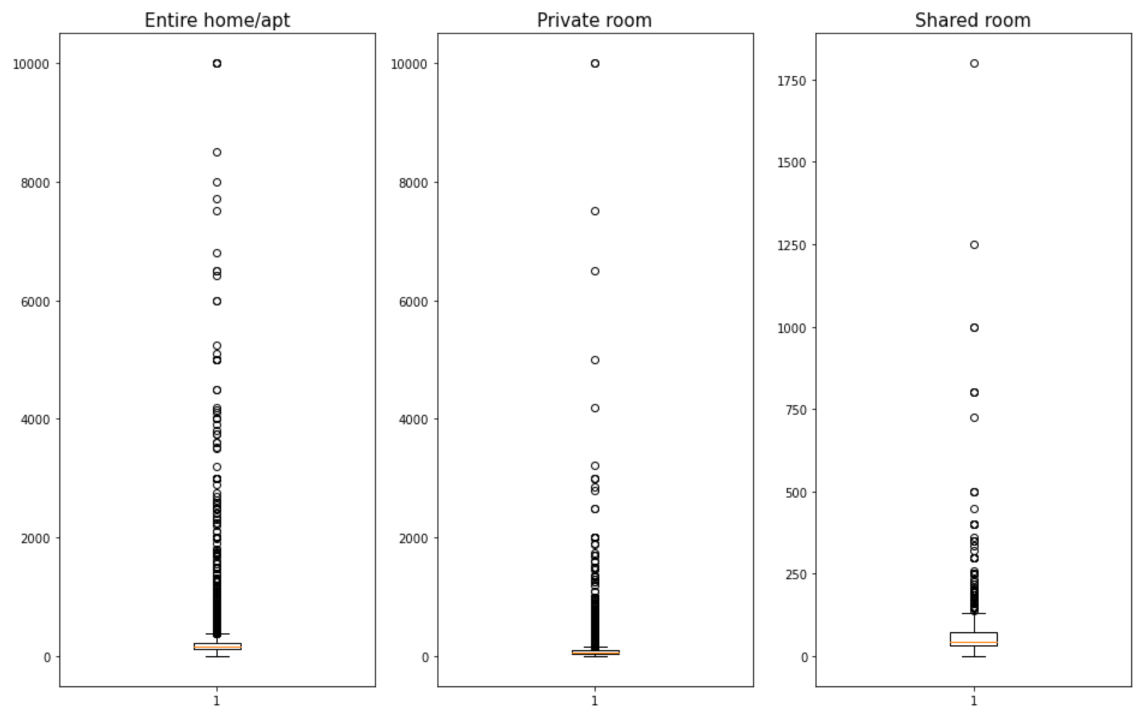
```
df.groupby(['room_type'])['price'].describe()
```

Out[42]:

room_type	count	mean	std	min	25%	50%	75%	max
Entire home/apt	25409.0	211.794246	284.041611	0.0	120.0	160.0	229.0	10000.0
Private room	22326.0	89.780973	160.205262	0.0	50.0	70.0	95.0	10000.0
Shared room	1160.0	70.127586	101.725252	0.0	33.0	45.0	75.0	1800.0

In [43]:

```
plt.figure(figsize = (16,10))

plt.subplot(1,3,1)
entire = df[df['room_type'] == 'Entire home/apt']
plt.boxplot(x  = entire['price'])
plt.title("Entire home/apt", fontsize = 15)

plt.subplot(1,3,2)
private = df[df['room_type'] == 'Private room']
plt.boxplot(x  = private['price'])
plt.title("Private room", fontsize = 15)

plt.subplot(1,3,3)
shared = df[df['room_type'] == 'Shared room']
plt.boxplot(x  = shared['price'])
plt.title("Shared room", fontsize = 15)

plt.show()
```



Observation

1. As we can see from the boxplot, the room type Entire home/apt has highest price going upto 10000 dollars, also it has a lot of outliers which means that the average price would be higher compared to the other two.
2. On the other hand, Private room has less outliers compared to the entire home/apt but the price also goes upto 10000 dollars. But, it has the avergae price of 90 dollar approximately.
3. Shared room is the least preffered room type and it also reflect to the the mean price and outliers. The maximum price of the shared room is only 1000 dollars and has the average price revolving around 70 dollar.

8. Minimum nights for different room types

In [44]:

```
df.head()
```

Out[44]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_ty
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv ro
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv ro
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home

In [45]:

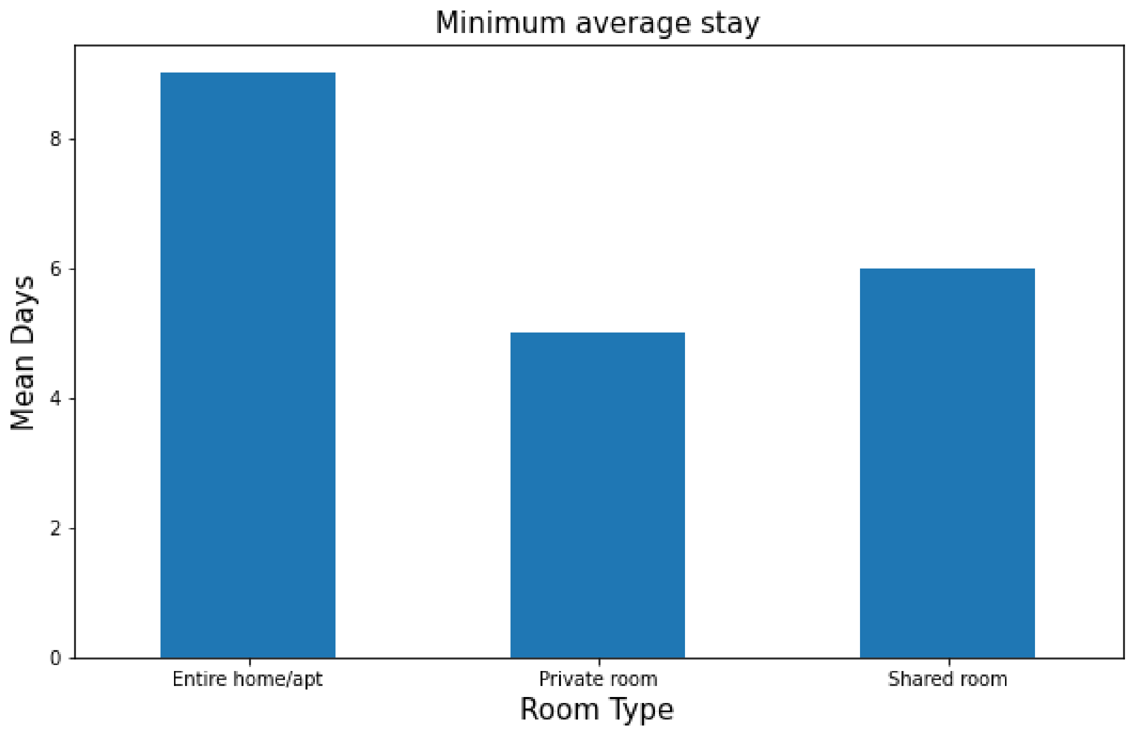
```
# getting the average of the minimum night and rounding it off to 0.
round(df.groupby(['room_type'])['minimum_nights'].mean().reset_index(), 0)
```

Out[45]:

	room_type	minimum_nights
0	Entire home/apt	9.0
1	Private room	5.0
2	Shared room	6.0

In [46]:

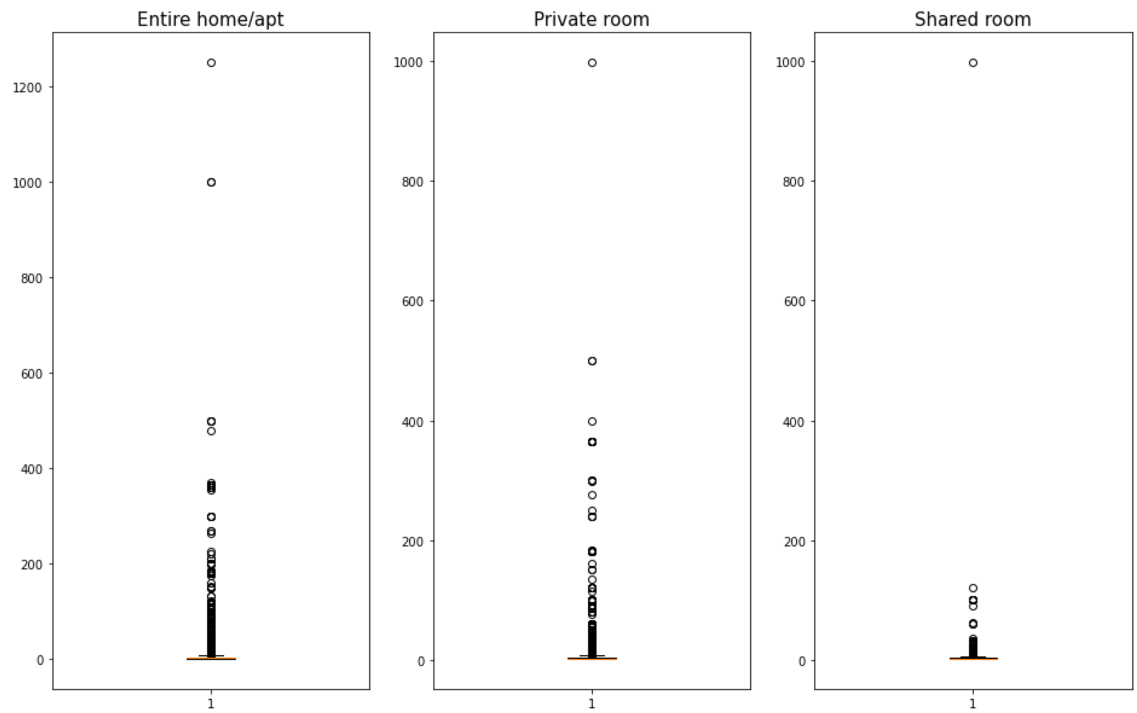
```
# Visualizing wrt bar graph
round(df.groupby(['room_type'])['minimum_nights'].mean(), 0).plot(kind = 'bar', figsize
plt.xticks(rotation = 360)
plt.title("Minimum average stay", fontsize = 15)
plt.xlabel("Room Type", fontsize = 15)
plt.ylabel("Mean Days", fontsize = 15)
plt.show()
```



Drawing the Boxplot so that we can has an idea about extreme values

In [47]:

```
plt.figure(figsize = (16,10))

plt.subplot(1,3,1)
entire1 = df[df['room_type'] == 'Entire home/apt']
plt.boxplot(x  = entire1['minimum_nights'])
plt.title("Entire home/apt", fontsize = 15)

plt.subplot(1,3,2)
private1 = df[df['room_type'] == 'Private room']
plt.boxplot(x  = private1['minimum_nights'])
plt.title("Private room", fontsize = 15)

plt.subplot(1,3,3)
shared1 = df[df['room_type'] == 'Shared room']
plt.boxplot(x  = shared1['minimum_nights'])
plt.title("Shared room", fontsize = 15)

plt.show()
```



Observation

If outliers are taken into consideration, then

1. According to the data, he mimimum days to stays in entire home/apt is 9 days, also it has maximum price.
2. The minimum days to stay for private room 5 days.
3. The minimum days to stay for shared room is 6 days.

9. Avaibility 365

In [48]:

```
df.head()
```

Out[48]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_ty
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv ro
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	En home
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv ro
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	En home
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	En home

In [49]:

```
df['availability_365'].value_counts().iloc[:10].sort_index()
```

Out[49]:

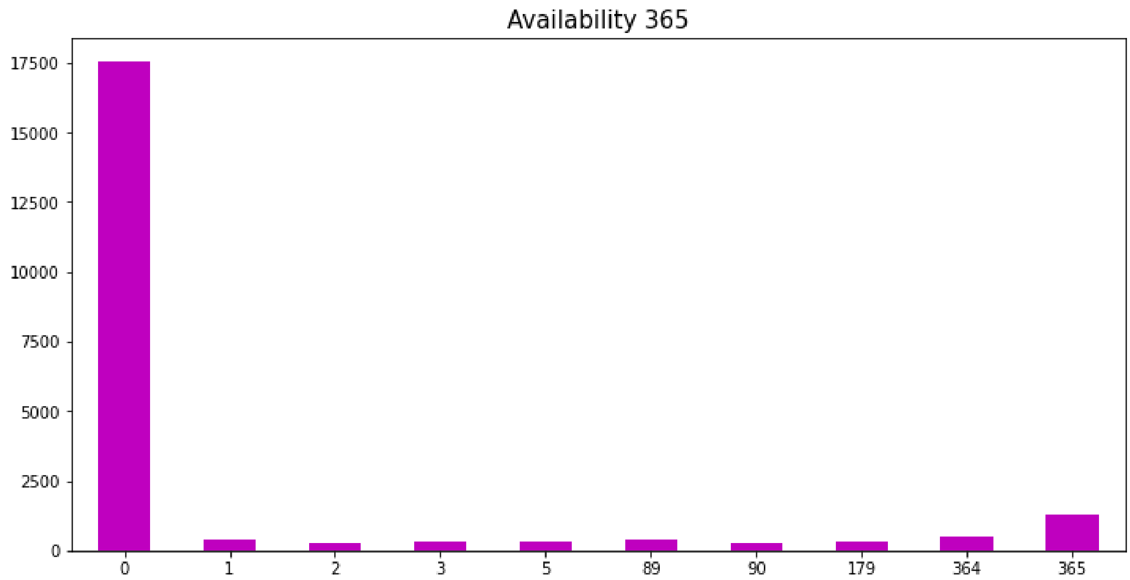
```
0      17533
1        408
2        270
3        306
5        340
89       361
90       290
179      301
364      491
365     1295
Name: availability_365, dtype: int64
```

In [50]:

```
df['availability_365'].value_counts().iloc[:10].sort_index().plot(kind = 'bar', figsize
plt.xticks(rotation = 360)
plt.title('Availability 365', fontsize = 15)
```

Out[50]:

```
Text(0.5, 1.0, 'Availability 365')
```



10. Neighbouthood group with respect to room type

In [51]:

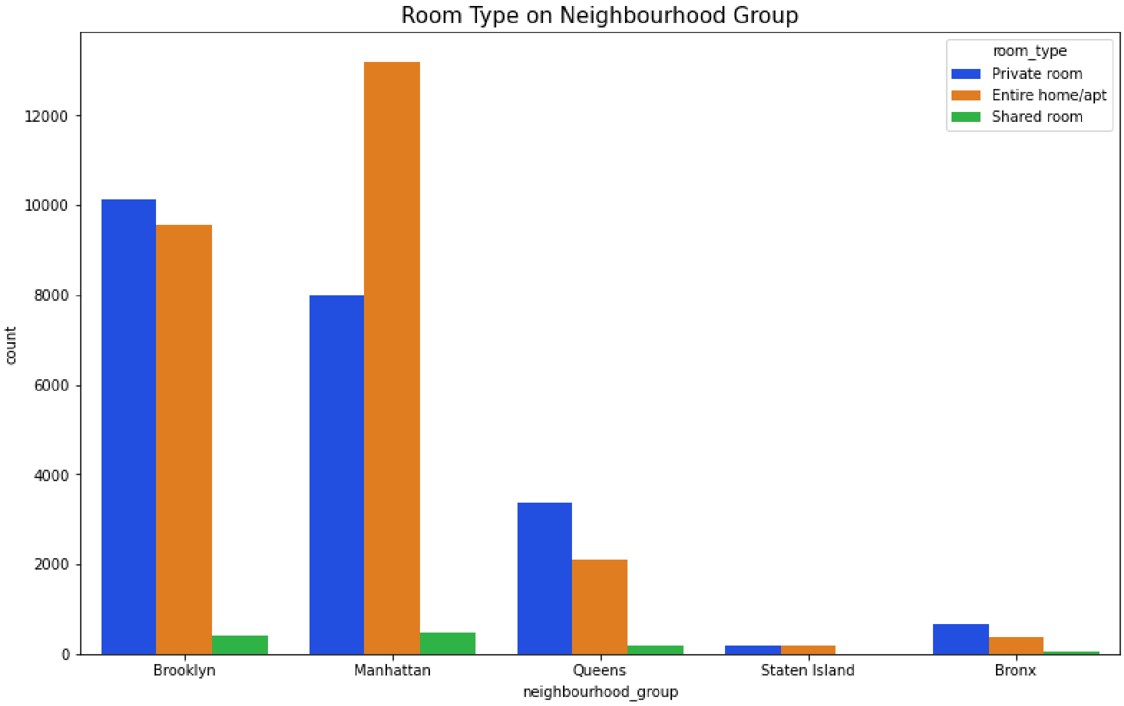
```
df.head()
```

Out[51]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_ty
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv ro
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	En home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv ro
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	En home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	En home/

In [52]:

```
plt.figure(figsize = (13,8))
sns.countplot(df.neighbourhood_group, hue=df.room_type, palette="bright")
plt.title("Room Type on Neighbourhood Group", fontsize = 15)
plt.show()
```



Observation

1. It looks like, the neighbourhood group Manhatten has the highest Entire home amongst all othergroups.
2. but Brooklyn has the most number iof private rooms.
3. Manhatten and Brooklyn has almost same number of Shared room.

11. finding the top 10 and bottom 10 of the neighbourhood

In [53]:

```
df.head()
```

Out[53]:

	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_t
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Priv rc
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Er home/
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Priv rc
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Er home/
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Er home/

In [54]:

```
print(df['neighbourhood'].value_counts().iloc[:10], '\n')
print(df['neighbourhood'].value_counts().tail(10))
```

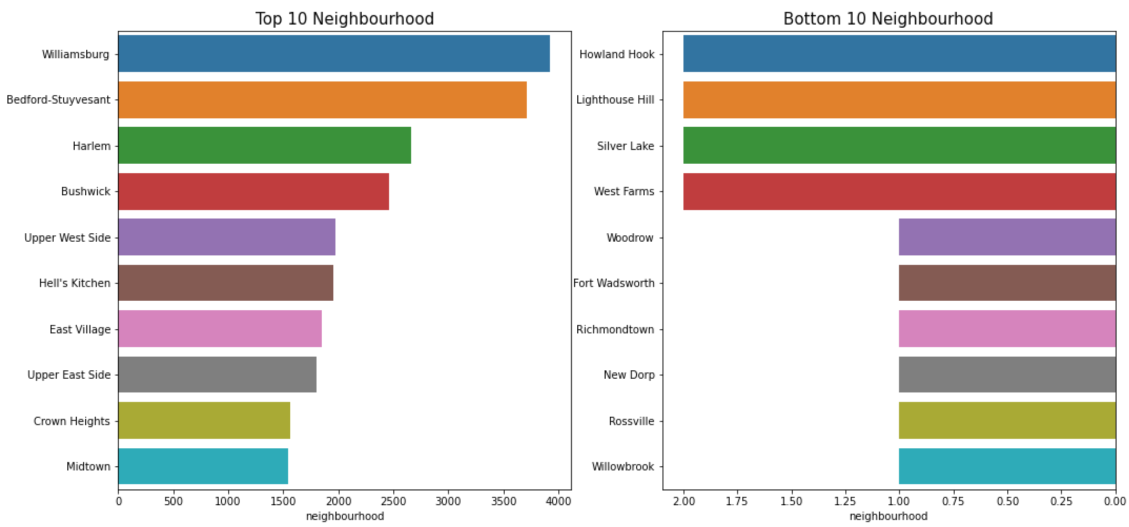
```
Williamsburg          3920
Bedford-Stuyvesant    3714
Harlem                2658
Bushwick              2465
Upper West Side       1971
Hell's Kitchen        1958
East Village          1853
Upper East Side       1798
Crown Heights         1564
Midtown               1545
Name: neighbourhood, dtype: int64

Howland Hook       2
Lighthouse Hill    2
Silver Lake        2
West Farms         2
Woodrow            1
Fort Wadsworth     1
Richmondtown       1
New Dorp           1
Rossville          1
Willowbrook        1
Name: neighbourhood, dtype: int64
```

In [55]:

```
plt.figure(figsize = (17,8))

plt.subplot(1,2,1)
sns.barplot(data = df, y = df['neighbourhood'].value_counts().iloc[:10].keys(),
            x = df['neighbourhood'].value_counts().iloc[:10])
plt.title("Top 10 Neighbourhood", fontsize = 15)

plt.subplot(1,2,2)
sns.barplot(data = df, y = df['neighbourhood'].value_counts().tail(10).keys(),
            x = df['neighbourhood'].value_counts().tail(10)).invert_xaxis()
plt.title("Bottom 10 Neighbourhood", fontsize = 15)
```

Out[55]:

```
Text(0.5, 1.0, 'Bottom 10 Neighbourhood')
```



Observation

1. Williamsburg and Bedford - Stuyvesant are the two highest Neighbourhood
2. willowbrook, Rossville, New Dorp, Richmondtown and woodrow are the least of the neighbourhood.

Conclusion

This Airbnb ('AB_NYC_2019') dataset for the 2019 year appeared to be a very rich dataset with a variety of columns that allowed us to dive deep into each significant column presented.

To begin, firstly, we identified the data of top ten host_id and we figured out that top host ID has 327 listings.

Secondly, we take "Neighbourhood_Group", and we found that Airbnb listings in New York City are concentrated in five neighborhoods: "Brooklyn," "Manhattan," "Queens," "Staten Island," and "Bronx". Moreover, we also learned from this chart that "Manhattan" and "Brooklyn" have the most hotel properties. Then, we found that Manhattan is the most expensive as the rental charges are more evenly distributed across all the price ranges, median price in Manhattan is approx $150 thats around double the median price of Bronx and the distributions in Queens and Staten Island appear to be very similar, while the Bronx appears to be the cheapest of the three.

Thirdly, we take the data of "room_type" and figured out that it is devided into three subcategaries and we can observe that the Entire Home/Apartment has the highest share, followed by the Private Room, and the least preferred is Shared Room. Futhermore, entire Home/Apartment is listed most near Manhattan, while Private Rooms and Apartments Near Brooklyn are Nearly equal.

Fourthly, we put our latitude and longitude columns to good use by creating a geographical map of Newyork city which represents the location of all the areas with their latitude and longtitude. In other map is Color-coded for listing price of room as per the location.

In addition, we returned to the first column "name" and found out the words from the hotel names, as well as the count for the most frequently used words by hosts. Hosts prefer to use Private rooms,brooklyn,central park,modern,nyc and Beautiful these words in their listing to seek customer attention.

Finally, we looked for the listings with the "most reviews". Count the rating of top ten reviewed hotels, and found out The top 10 most reviewed listings on Airbnb for NYC have an average price of $65 per night, with the majority of them under 50, and 9/10 of them are "Private Room" types, with the top reviewed listing having 629 reviews.