Airbnb Case Study by Priya & Shubham

1.Importing Libraries

2.Loading the dataset

3.Data Cleaning:

1. **Introduction and Library Imports**:
    - Introduction and Importing Required Libraries
2. **Data Loading**:
    - Loading the Dataset
3. **Data Preprocessing**:
    - Data Cleaning and Preprocessing
    - Handling Duplicates, Cleaning Columns, and Dealing with Null Values
    - Data Transformation
4. **Exploratory Data Analysis (EDA)**:
    - Exploratory Data Analysis and Visualization
    - Analyzing Relationships Between Features through Visualizations
5. **Average Preferred Price by Location**:
    - Determining the Average Preferred Price by Location
6. **Number of Active Hosts by Location**:
    - Identifying Locations with the Highest Number of Active Hosts
7. **Rent by Location**:
    - Analyzing Rent Variations by Location
8. **Most Popular Airbnb Host in New York**:
    - Finding the Most Popular/Demanded Airbnb Host in New York
9. **Total Count of Each Room Type**:
    - Counting the Total Number of Each Room Type
10. **Room Types and Availability by Neighborhood Groups**:
    - Exploring the Relationship Between Room Types and Availability in Different Neighborhood Groups
11. **Top 25 Most Used Words in Listing Names**:
    - Discovering the Top 25 Most Frequent Words in Listing Names
12. **Top 10 Hosts with Most Listings**:
    - Finding the Top 10 Hosts with the Most Listings
13. **Top Three Hosts by Turnover**:
    - Identifying the Top Three Hosts Based on Their Turnover
14. **Total Nights Spent by Location**:
    - Calculating the Total Number of Nights Spent per Location
15. **Total Nights Spent by Room Types**:

15. **Total Nights Spent by Room Types**:
    - Analyzing the Total Number of Nights Spent by Room Types
16. **Top 10 Highest Listing Neighborhoods**:
    - Identifying the Top 10 Neighborhoods with the Highest Number of Listings

Importing Libraries

In [2]:

```
#Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [3]:

```
file_path = 'AB_NYC_2019.csv'
```

In [4]:

```
df=pd.read_csv(file_path)
df.head()
```

Out[4]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.7

In [5]:

```
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
```

In [6]:

```
df.shape
```

Out[6]:

```
(48895, 16)
```

Cleaning Data:

In [7]:

```
#removing dupes
df.duplicated().sum()
df.drop_duplicates(inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 6.3+ MB
```

In [8]:

```
airbnb_non_null_df = df.dropna() #droping null values
airbnb_non_null_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38821 entries, 0 to 48852
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              38821 non-null  int64  
 1   name                            38821 non-null  object 
 2   host_id                         38821 non-null  int64  
 3   host_name                       38821 non-null  object 
 4   neighbourhood_group             38821 non-null  object 
 5   neighbourhood                   38821 non-null  object 
 6   latitude                        38821 non-null  float64
 7   longitude                       38821 non-null  float64
 8   room_type                       38821 non-null  object 
 9   price                           38821 non-null  int64  
 10  minimum_nights                  38821 non-null  int64  
 11  number_of_reviews               38821 non-null  int64  
 12  last_review                     38821 non-null  object 
 13  reviews_per_month               38821 non-null  float64
 14  calculated_host_listings_count  38821 non-null  int64  
 15  availability_365                38821 non-null  int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 5.0+ MB
```

Summing up the count for each category of room:

In [9]:

```
# Finding unique values from column 'room_type'
airbnb_room_type = df.room_type.unique()
airbnb_room_type
```

Out[9]:

```
array(['Private room', 'Entire home/apt', 'Shared room'], dtype=object)
```

In [10]:

```
#Identifying the room type with the highest listing frequency
airbnb_roomtype_frequency = dict(df.room_type.value_counts())
airbnb_roomtype_frequency
```

Out[10]:

```
{'Entire home/apt': 25409, 'Private room': 22326, 'Shared room': 1160}
```

1.What is the average preferred price by customers according to the location?

In [11]:

```
df.head()
```

Out[11]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.7

In [12]:

```
avg_preffered_price_df = df.groupby(['neighbourhood_group','room_type'], as_index=False
avg_preffered_price_df
```

Out[12]:

	Location	room_type	Average Price
0	Bronx	Entire home/apt	127.506596
1	Bronx	Private room	66.788344
2	Bronx	Shared room	59.800000
3	Brooklyn	Entire home/apt	178.327545
4	Brooklyn	Private room	76.500099
5	Brooklyn	Shared room	50.527845
6	Manhattan	Entire home/apt	249.239109
7	Manhattan	Private room	116.776622
8	Manhattan	Shared room	88.977083
9	Queens	Entire home/apt	147.050573
10	Queens	Private room	71.762456
11	Queens	Shared room	69.020202
12	Staten Island	Entire home/apt	173.846591
13	Staten Island	Private room	62.292553
14	Staten Island	Shared room	57.444444

In [13]:

```
avg_preffered_price_df = df.groupby(['neighbourhood_group','room_type'])['price'].mean(
colors = ['red', 'green', 'blue', 'orange']
avg_preffered_price_df
```

Out[13]:

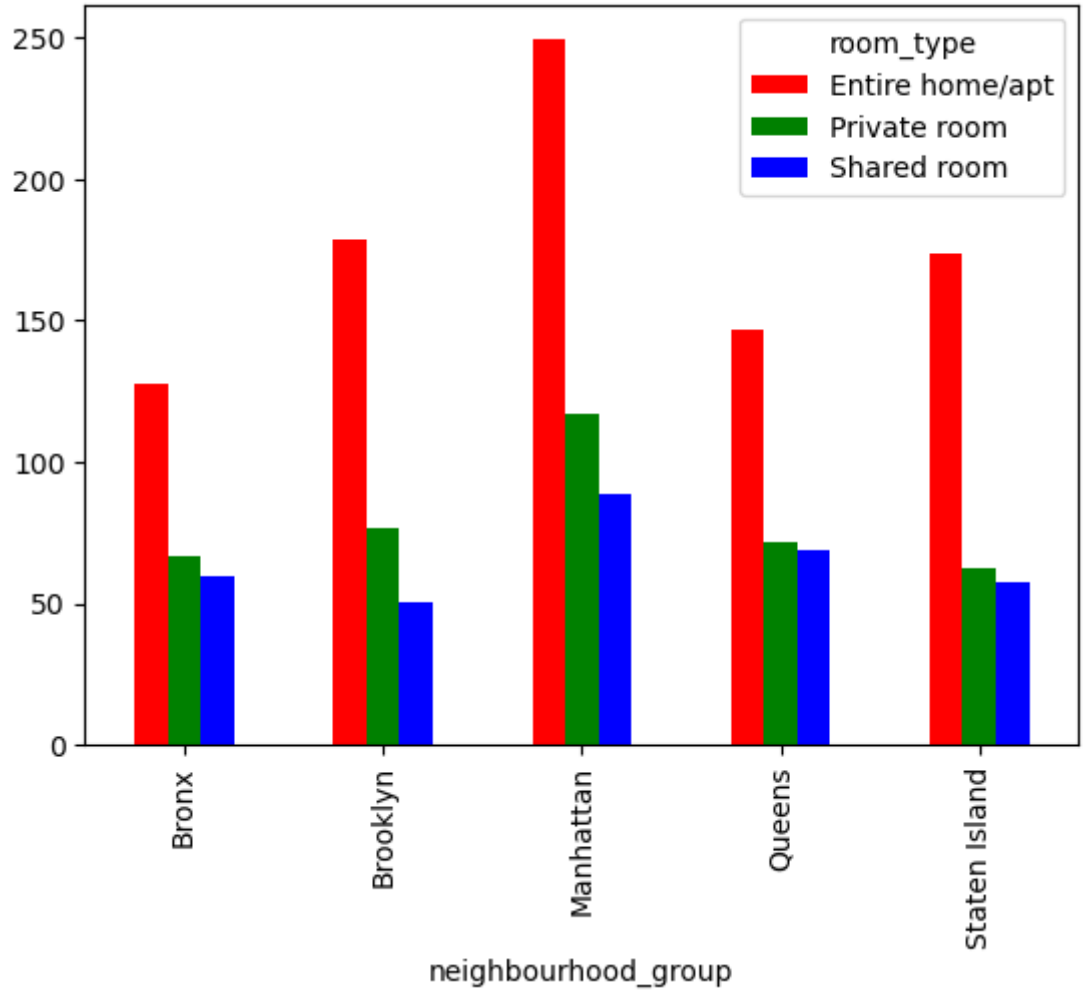
room_type	Entire home/apt	Private room	Shared room
neighbourhood_group			
Bronx	127.506596	66.788344	59.800000
Brooklyn	178.327545	76.500099	50.527845
Manhattan	249.239109	116.776622	88.977083
Queens	147.050573	71.762456	69.020202
Staten Island	173.846591	62.292553	57.444444

In [14]:

```
avg_preffered_price_df.plot.bar(color=colors)
```

Out[14]:

```
<AxesSubplot:xlabel='neighbourhood_group'>
```



2. The number of active hosts in each location (Where most hosts prioritize property ownership?

In [15]:

```
#Apply group by operation on neighbourhood_group for find the number of host according
no_of_host_per_location = df.groupby('neighbourhood_group',as_index=False)['host_id'].c
no_of_host_per_location
```

Out[15]:

	Location	Host
2	Manhattan	21661
1	Brooklyn	20104
3	Queens	5666
0	Bronx	1091
4	Staten Island	373

Answer: "Manhattan stands out as the primary hub where hosts predominantly conduct their business."

In [16]:

```
#Take a simple format of above question without index for plot the line chart
no_of_host_per_location = df.groupby('neighbourhood_group')['host_id'].count()
no_of_host_per_location
```

Out[16]:

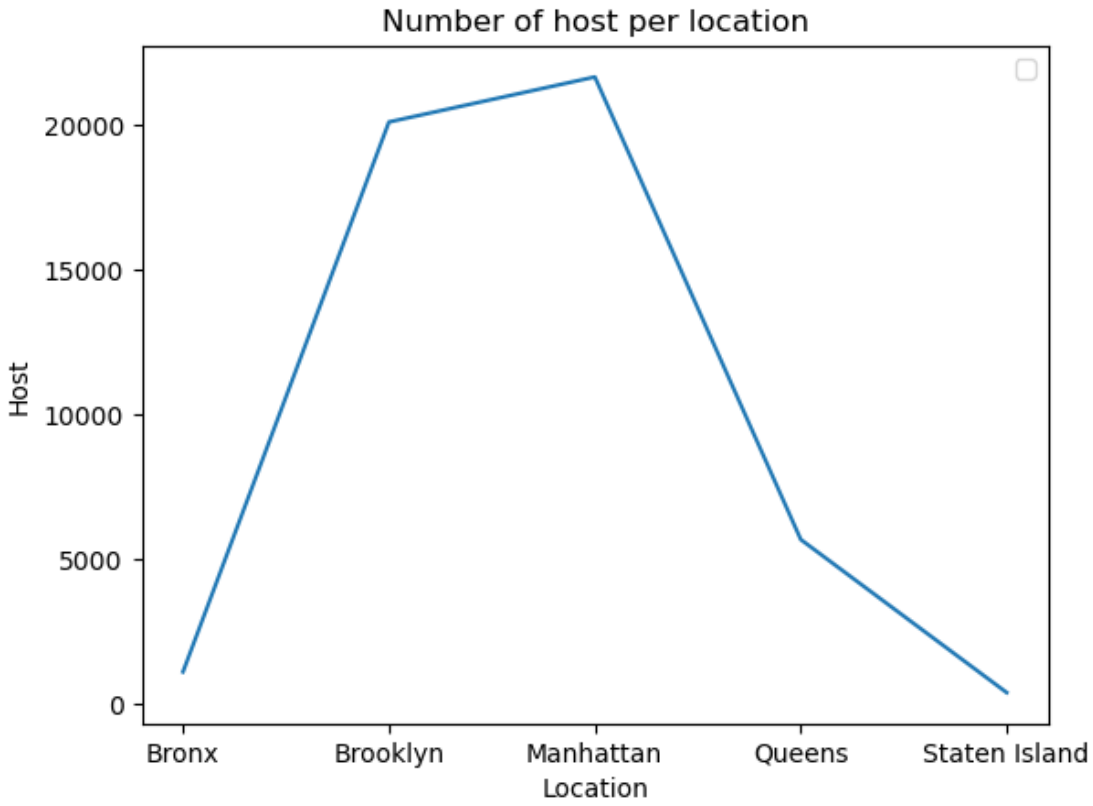
```
neighbourhood_group
Bronx             1091
Brooklyn         20104
Manhattan        21661
Queens            5666
Staten Island      373
Name: host_id, dtype: int64
```

In [17]:

```
#Graph
plt.legend(labels = ['No. of Host'])
plt.plot(no_of_host_per_location)
plt.title('Number of host per location')
plt.ylabel('Host')
plt.xlabel('Location')
```

Out[17]:

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



In [18]:

```
df.describe()
```

Out[18]:

	id	host_id	latitude	longitude	price	minimum_ni
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.00
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.02
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.51
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.00
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.00
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.00
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.00
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.00

Noted point:

Here we can see that there is minimum price 0$.We have to fix it anyway because Airbnb is not provide free stay in any hotel. Solution: For that we check where the price is 0. We create one funtion in which as per minimum nights the price will be set as per formula [Refer below cells for more detail]

There are some entries in dataset in which minimum nights = 1 but it's respective price is 0 Solution: We replace the price value with the mean value of price which amount is less than 100$. [Refer below cells for more detail]

In [19]:

```
#identify the observations where minimum nights is 1 and price is minimum
df[df['minimum_nights'] ==1].sort_values(['price', 'minimum_nights'], ascending=[True,
```

Out[19]:

	id	name	host_id	host_name	neighbourhood_group	neighbourh
25794	20639628	Spacious comfortable master bedroom with nice ...	86327101	Adeyemi	Brooklyn	Bedf Stuyve
25795	20639792	Contemporary bedroom in brownstone with nice view	86327101	Adeyemi	Brooklyn	Bedf Stuyve
25796	20639914	Cozy yet spacious private brownstone bedroom	86327101	Adeyemi	Brooklyn	Bedf Stuyve
21700	17437106	Couch in Harlem Harvey Refugees only	33511962	Morgan	Manhattan	Ha
22835	18490141	IT'S SIMPLY CONVENIENT!	97001292	Maria	Queens	Jam
...	...	...	...	...	...	
44034	33998396	3000 sq ft daylight photo studio	3750764	Kevin	Manhattan	Che
42523	33007610	70' Luxury MotorYacht on the Hudson	7407743	Jack	Manhattan	Battery I
45666	34895693	Gem of east Flatbush	262534951	Sandra	Brooklyn	East Flatt
29662	22779726	East 72nd Townhouse by (Hidden by Airbnb)	156158778	Sally	Manhattan	Upper East
4377	2953058	Film Location	1177497	Jessica	Brooklyn	Clintor

12720 rows × 16 columns

In [20]:

```
df.loc[df['price'] < 100,'price'].count()
```

Out[20]:

21877

In [21]:

```
airbnb_price_df = df.copy()
round(df.loc[df['price'] < 100,'price'].mean())
```

Out[21]:

```
65
```

In [22]:

```
#Change the value of price where minimum night = 1 and price of their property is less
airbnb_price_df['price'] = np.where(
   (airbnb_price_df['price'] <= 50) & (airbnb_price_df['minimum_nights'] == 1) ,round(
   )
airbnb_price_df[airbnb_price_df['price']==0].count()
```

Out[22]:

```
id                                8
name                              8
host_id                           8
host_name                         8
neighbourhood_group               8
neighbourhood                     8
latitude                          8
longitude                         8
room_type                         8
price                             8
minimum_nights                    8
number_of_reviews                 8
last_review                       7
reviews_per_month                 7
calculated_host_listings_count    8
availability_365                  8
dtype: int64
```

In [23]:

```
airbnb_price_df[airbnb_price_df['price']==0]
```

Out[23]:

	id	name	host_id	host_name	neighbourhood_group	neighbourh
23161	18750597	Huge Brooklyn Brownstone Living, Close to it all.	8993084	Kimberly	Brooklyn	Bedf Stuyve
25433	20333471	★Hostel Style Room | Ideal Traveling Buddies★	131697576	Anisha	Bronx	East Morris
25634	20523843	MARTIAL LOFT 3: REDEMPTION (upstairs, 2nd room)	15787004	Martial Loft	Brooklyn	Bush
25753	20608117	Sunny, Quiet Room in Greenpoint	1641537	Lauren	Brooklyn	Greenp
25778	20624541	Modern apartment in the heart of Williamsburg	10132166	Aymeric	Brooklyn	Williamst
26259	20933849	the best you can find	13709292	Qiuchi	Manhattan	Murray
26841	21291569	Coliving in Brooklyn! Modern design / Shared room	101970559	Sergii	Brooklyn	Bush
26866	21304320	Best Coliving space ever! Shared room.	101970559	Sergii	Brooklyn	Bush

In [24]:

```
#Function that gives a new price according to minimum night
def price_calculator(min_night):
  '''Get the price based on the minimum night you are given'''
  mean_price = round(df.loc[df['price'] < 100,'price'].mean())
  new_price = mean_price * min_night

  return new_price
```

In [25]:

```
airbnb_price_df['price'] = np.where(
   (airbnb_price_df['price'] == 0) ,price_calculator(airbnb_price_df['minimum_nights'])
```

In [26]:

```
airbnb_price_df.describe()
```

Out[26]:

	id	host_id	latitude	longitude	price	minimum_ni
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.00
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	153.829614	7.02
std	1.098311e+07	7.861097e+07	0.054530	0.046157	239.999410	20.51
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	10.000000	1.00
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.00
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.00
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.00
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.00

3. Which locations have the highest and lowest rent payments by customers?

In [27]:

```
#Get the highest rent according to location
max_price_df = airbnb_price_df.groupby('neighbourhood_group',as_index=False)['price'].r
max_price_df
```

Out[27]:

	Location	Maximum price
1	Brooklyn	10000
2	Manhattan	10000
3	Queens	10000
4	Staten Island	5000
0	Bronx	2500

In [28]:

```
min_price_df = airbnb_price_df.groupby('neighbourhood_group',as_index=False)['price'].r
min_price_df
```

Out[28]:

	Location	Minimum price
1	Brooklyn	10
2	Manhattan	10
3	Queens	10
0	Bronx	20
4	Staten Island	20

In [29]:

```
#Get the combine dataframe of minimum and maximum price according to location
merge_price_df = pd.merge(max_price_df, min_price_df, on='Location')
merge_price_df
```

Out[29]:

	Location	Maximum price	Minimum price
0	Brooklyn	10000	10
1	Manhattan	10000	10
2	Queens	10000	10
3	Staten Island	5000	20
4	Bronx	2500	20

In [30]:

```
#Create a copy of merge price dataframe
merge_price_df_copy = merge_price_df.copy()
```

In [31]:

```
#Create a function for log transformation of maximum price and minimum price
def log_values(values):
  '''This function takes a price value as a input and give it's log values'''
  new_max_price = np.log2(values)

  return new_max_price
```

In [32]:

```
merge_price_df_copy['Maximum price'] = merge_price_df_copy.apply(lambda x: log_values(x
merge_price_df_copy['Minimum price'] = merge_price_df_copy.apply(lambda x: log_values(x
```

In [33]:

```
merge_price_df_copy
```

Out[33]:

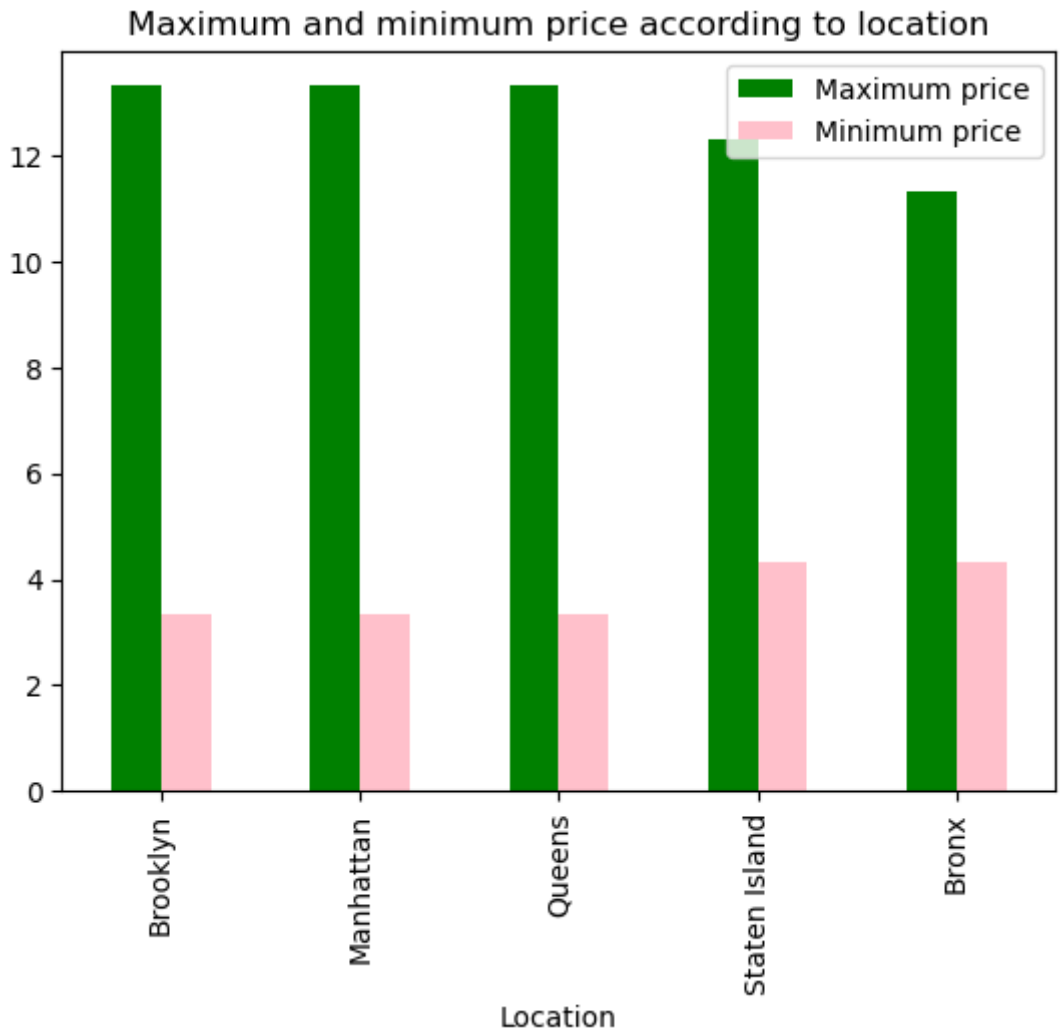
	Location	Maximum price	Minimum price
0	Brooklyn	13.287712	3.321928
1	Manhattan	13.287712	3.321928
2	Queens	13.287712	3.321928
3	Staten Island	12.287712	4.321928
4	Bronx	11.287712	4.321928

In [34]:

```
merge_price_df_copy.plot(x="Location", y=['Maximum price','Minimum price'], kind="bar",
plt.legend(labels = ['Maximum price','Minimum price'])
plt.title('Maximum and minimum price according to location')
```

Out[34]:

```
Text(0.5, 1.0, 'Maximum and minimum price according to location')
```



4. Finding out which Airbnb hosts are the most popular in New York.

In [35]:

```
#Get the host based on number of reviews
host_based_on_review_df = airbnb_price_df.groupby(['host_id','host_name'],as_index=Fals
host_based_on_review_df
```

Out[35]:

	host_id	host_name	number_of_reviews
21304	37312959	Maya	2273
1052	344035	Brooklyn& Breakfast -Len-	2205
18626	26432133	Danielle	2017
20872	35524316	Yasu & Akiko	1971
21921	40176101	Brady	1818
...	...	...	...
21806	39695769	Avra	0
21809	39706334	Erin	0
21812	39724060	Jaime	0
21816	39731713	Polina	0
37438	274321313	Kat	0

37439 rows × 3 columns

In [36]:

```
#Get the host based on availability in a year
host_based_on_availability_df = airbnb_price_df.groupby(['host_id','host_name'],as_inde
host_based_on_availability_df
```

Out[36]:

	host_id	host_name	availability_365
0	2438	Tasos	0.0
11587	10263977	A	0.0
11588	10264372	Tyrell	0.0
11589	10264377	Anthony	0.0
22251	41757762	Sara	0.0
...	...	...	...
22355	42237225	Orlando	365.0
36648	262287464	Daniel	365.0
32610	165448425	Shana	365.0
21179	36881439	Etkin	365.0
12734	12112004	Kylie	365.0

37439 rows × 3 columns

In [37]:

```
#Merge two dataframe based on number of reviews and availability in a year
popular_host_df = pd.merge(host_based_on_review_df,host_based_on_availability_df,on = |

popular_host_df
```

Out[37]:

	host_id	host_name	Number of reviews	Availability in a year
12	22959695	Gurpreet Singh	1157	0.0
41	99392252	Michael	732	0.0
47	121391142	Deloris	693	0.0
122	792159	Wanda	480	0.0
125	37818581	Sofia	479	0.0
...	...	...	...	...
37151	37424221	Trevor	0	365.0
37239	35741633	Chen	0	365.0
37255	38363932	Marie	0	365.0
37257	40733012	Victor	0	365.0
37263	40834217	Jay	0	365.0

37439 rows × 4 columns

In [38]:

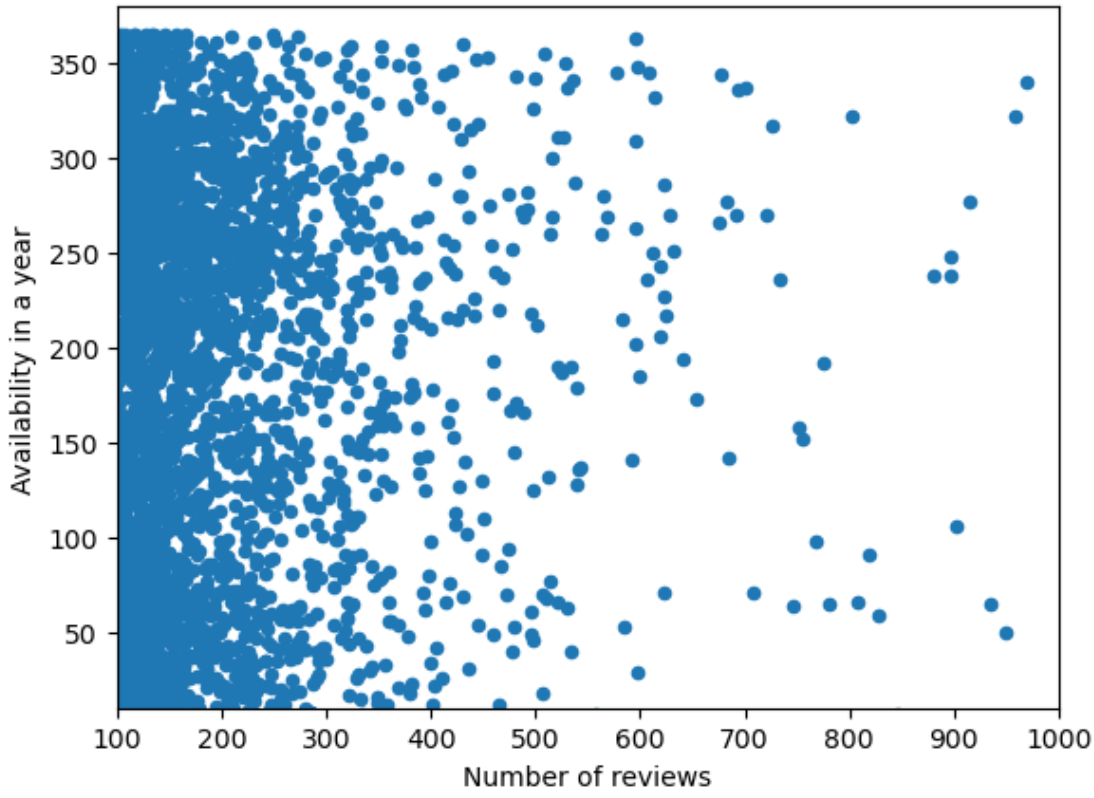
```
#Most popular top 5 host in new york
popular_host_df.head(5)
```

Out[38]:

	host_id	host_name	Number of reviews	Availability in a year
12	22959695	Gurpreet Singh	1157	0.0
41	99392252	Michael	732	0.0
47	121391142	Deloris	693	0.0
122	792159	Wanda	480	0.0
125	37818581	Sofia	479	0.0

In [39]:

```
#Create scatter plot based on number of reviews and availability in a year
popular_host_df.plot.scatter(x='Number of reviews', y='Availability in a year', xlim=(1
```



Answering below Questions:

Find the total count of each room type

Which are the top 25 most used words in listing names?

Find top 10 hosts with most listings

In [40]:

```
#Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [41]:

```
df.head()
```

Out[41]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.7

In [42]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 6.3+ MB
```

In [43]:

```
df.shape
```

Out[43]:

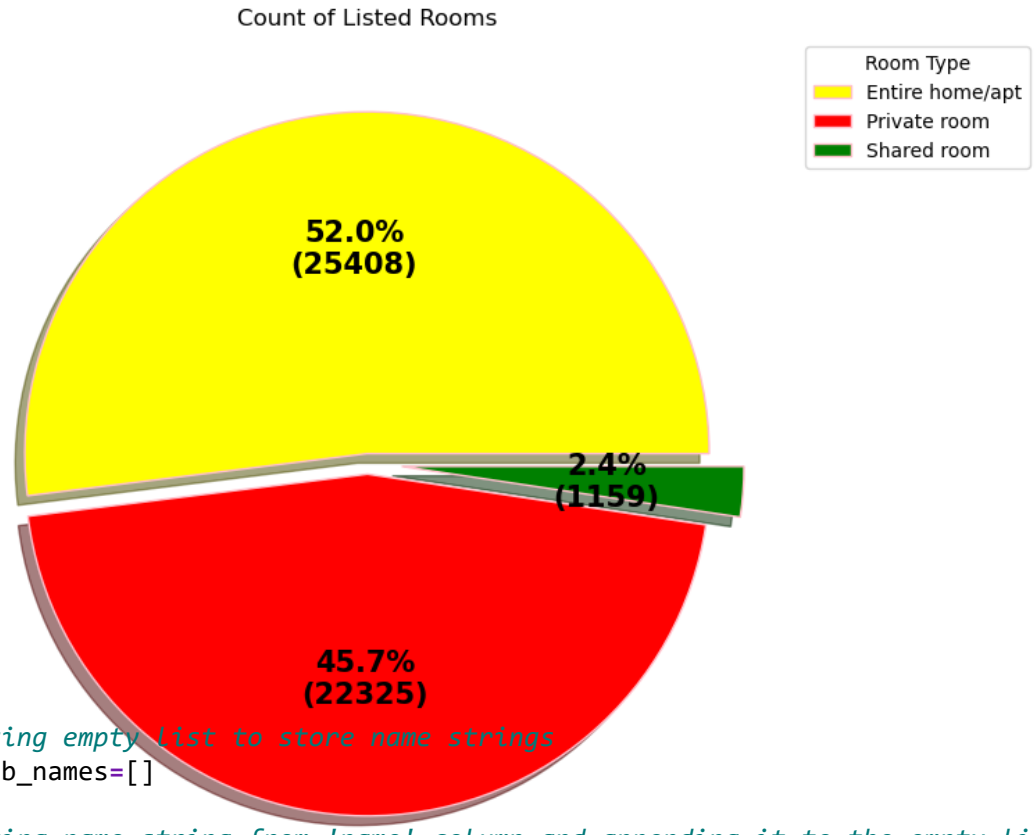
```
(48895, 16)
```

In [44]:

```
room_type = list(airbnb_roomtype_frequency.keys())
data = list(airbnb_roomtype_frequency.values())

# Creating color parameters

colors = ( "yellow", "red", "green",)

# Creating explode data

explode = (0.03, 0.03, 0.1)

# Wedge properties
wp = { 'linewidth' : 1, 'edgecolor' : "pink" }

# Creating autocpt arguments
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    return "{:.1f}%\n({:d})".format(pct, absolute)

# Creating Pie Chart

# Creating plot
fig, airbnb_pie_chart = plt.subplots(figsize =(12, 8))
wedges, texts, autotexts = airbnb_pie_chart.pie(data, autopct = lambda pct: func(pct, d
                                                explode = explode,
                                                shadow = True,
                                                colors = colors,
                                                startangle = 0,
                                                wedgeprops = wp,
                                                textprops = dict(color ="black"))

# Adding legend
airbnb_pie_chart.legend(wedges, room_type,
                        title ="Room Type",
                        loc ="upper left",
                        bbox_to_anchor=(1, 0., 0.,1))

plt.setp(autotexts, size = 15, weight = "bold")
airbnb_pie_chart.set_title("Count of Listed Rooms")
```

Out[44]:

```
Text(0.5, 1.0, 'Count of Listed Rooms')
```



mes?

```
#Creting empty list to store name strings
airbnb_names=[]

#Getting name string from 'name' column and appending it to the empty list
for name in df.name:
    airbnb_names.append(name)

#Setting a function to split name strings into seperate words
def split_name(name):
    ns = str(name).split()
    return ns

#Creating empty list to store the count of words
names_count = []

#Getting name string to append it to the names_count list
for n in airbnb_names:
    for word in split_name(n):
        word = word.lower()
        names_count.append(word)
```

In [46]:

```
# Importing 'counter' library to count and generate raw data and count top 25 most used
from collections import Counter

# Counting most common words
count_words = Counter(names_count).most_common()
count_words[:25]
```

Out[46]:

```
[('in', 16725),
 ('room', 9379),
 ('bedroom', 7231),
 ('private', 6978),
 ('apartment', 6112),
 ('cozy', 4627),
 ('the', 3869),
 ('to', 3827),
 ('studio', 3772),
 ('brooklyn', 3629),
 ('apt', 3571),
 ('spacious', 3387),
 ('1', 3357),
 ('with', 3092),
 ('2', 3079),
 ('of', 2993),
 ('east', 2967),
 ('and', 2869),
 ('manhattan', 2853),
 ('&', 2820),
 ('park', 2632),
 ('sunny', 2536),
 ('beautiful', 2320),
 ('near', 2295),
 ('williamsburg', 2293)]
```

In [47]:

```
items_to_remove = {('in', 16733), ('the', 3869), ('to', 3827), ('of', 2993), ('-', 2272
top_25_cleaned = [e for e in count_words if e not in items_to_remove]
top_25 = top_25_cleaned[:25]
top_25
```

Out[47]:

```
[('in', 16725),
 ('room', 9379),
 ('bedroom', 7231),
 ('private', 6978),
 ('apartment', 6112),
 ('cozy', 4627),
 ('studio', 3772),
 ('brooklyn', 3629),
 ('apt', 3571),
 ('spacious', 3387),
 ('1', 3357),
 ('with', 3092),
 ('2', 3079),
 ('east', 2967),
 ('and', 2869),
 ('manhattan', 2853),
 ('&', 2820),
 ('park', 2632),
 ('sunny', 2536),
 ('beautiful', 2320),
 ('near', 2295),
 ('williamsburg', 2293),
 ('-', 2255),
 ('village', 2055),
 ('heart', 2044)]
```

In [48]:

```
word_count_df = pd.DataFrame(top_25)
word_count_df.rename(columns={0:'Words',1:'Counts'},inplace=True)
word_count_df
```

Out[48]:

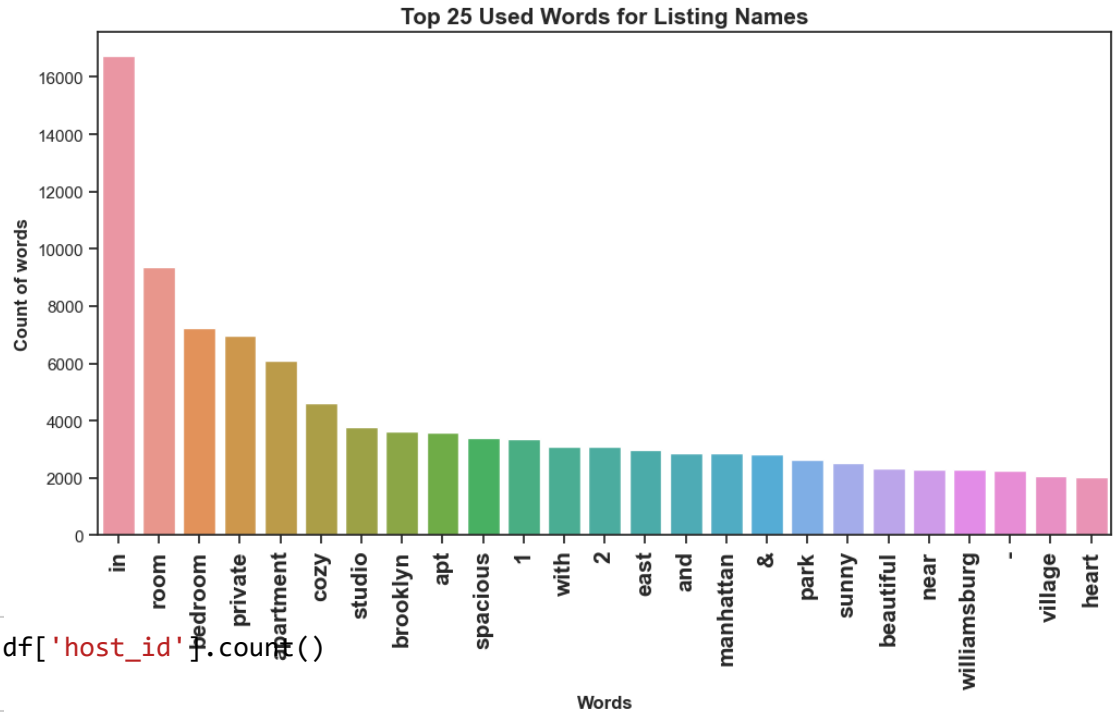
	Words	Counts
0	in	16725
1	room	9379
2	bedroom	7231
3	private	6978
4	apartment	6112
5	cozy	4627
6	studio	3772
7	brooklyn	3629
8	apt	3571
9	spacious	3387
10	1	3357
11	with	3092
12	2	3079
13	east	2967
14	and	2869
15	manhattan	2853
16	&	2820
17	park	2632
18	sunny	2536
19	beautiful	2320
20	near	2295
21	williamsburg	2293
22	-	2255
23	village	2055
24	heart	2044

In [49]:

```
#Setting the figure size
sns.set(rc={'figure.figsize':(12,6)})

#Setting background colour of chart as white
sns.set_style('ticks')

#Plotting the Chart
count_viz = sns.barplot(x='Words',y='Counts', data = word_count_df)

# Naming the Chart
count_viz.set_title('Top 25 Used Words for Listing Names', weight = 'bold').set_fontsiz


# Naming X & Y axis
count_viz.set_ylabel('Count of words', weight = 'bold')
count_viz.set_xlabel('Words', weight = 'bold')

#Adjusting Bar labels
count_viz.set_xticklabels(count_viz.get_xticklabels(),rotation = 90, weight = 'bold', s
```

Out[49]:

```
[Text(0, 0, 'in'),
 Text(1, 0, 'room'),
 Text(2, 0, 'bedroom'),
 Text(3, 0, 'private'),
 Text(4, 0, 'apartment'),
 Text(5, 0, 'cozy'),
 Text(6, 0, 'studio'),
 Text(7, 0, 'brooklyn'),
 Text(8, 0, 'apt'),
 Text(9, 0, 'spacious'),
 Text(10, 0, '1'),
 Text(11, 0, 'with'),
 Text(12, 0, '2'),
 Text(13, 0, 'east'),
 Text(14, 0, 'and'),
 Text(15, 0, 'manhattan'),
 Text(16, 0, '&'),
 Text(17, 0, 'park'),
 Text(18, 0, 'sunny'),
 Text(19, 0, 'beautiful'),
 Text(20, 0, 'near'),
 Text(21, 0, 'williamsburg'),
 Text(22, 0, '-'),
 Text(23, 0, 'village'),
 Text(24, 0, 'heart')]
```




```
df['host_id'].count()
```

Out[50]:

48895

In [51]:

```
#Creating DataFrame
count_host_id = list(df['host_id'].value_counts())
host_id = list(df.host_id)
listing_count= list(zip(host_id,count_host_id))
count_host_id_df= pd.DataFrame(listing_count)
count_host_id_df.rename(columns={0:'Host_Id',1:'Counts'},inplace=True)

#Storing top 10 hosts with most listings
top_host_id = count_host_id_df.head(10)
top_host_id
```

Out[51]:

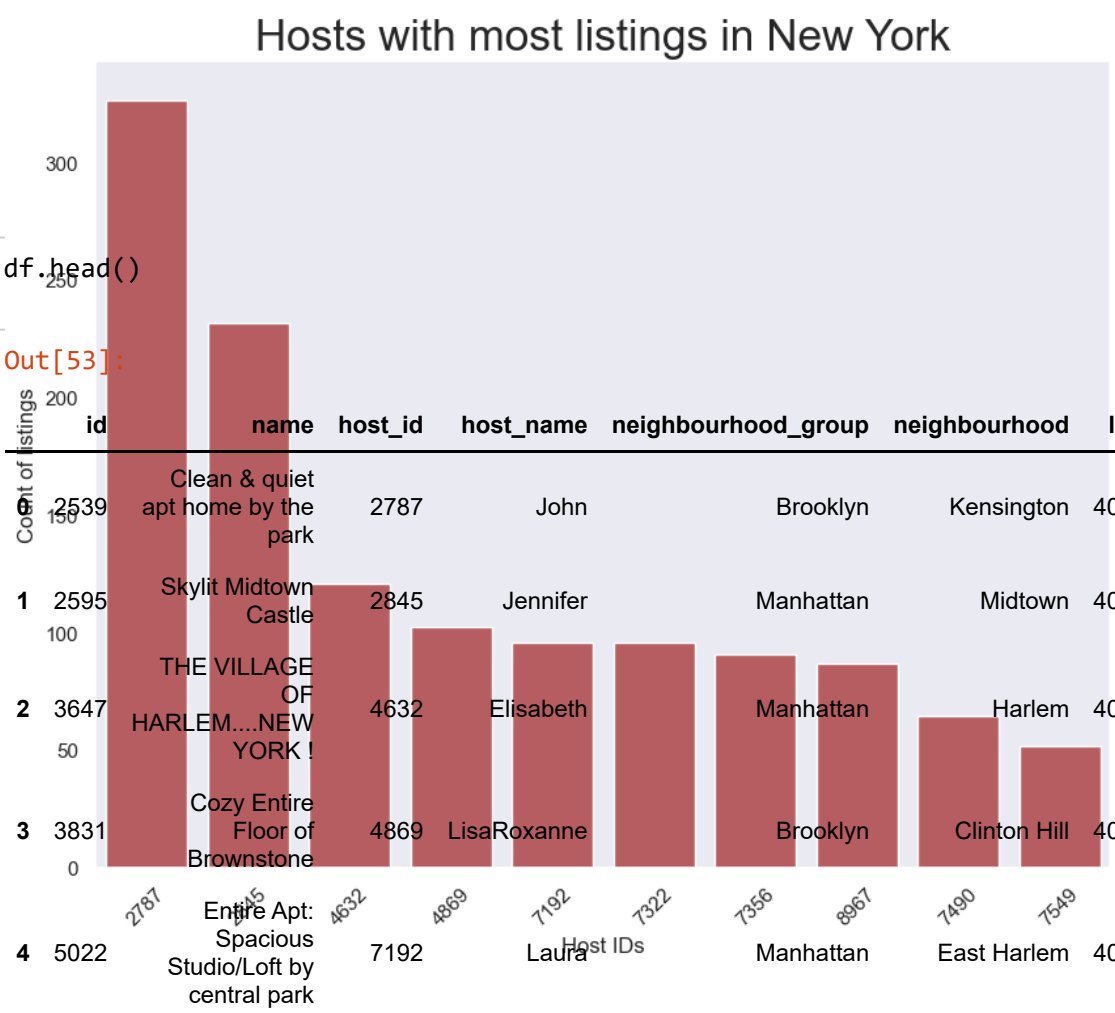
	Host_Id	Counts
0	2787	327
1	2845	232
2	4632	121
3	4869	103
4	7192	96
5	7322	96
6	7356	91
7	8967	87
8	7490	65
9	7549	52

In [52]:

```
sns.set(rc={'figure.figsize':(10,8)})

#Setting background colour of chart as white
sns.set_style('dark')

#Plotting the Chart
viz_bar = sns.barplot(x= 'Host_Id', y= 'Counts', color='r', data=top_host_id,
                      order=top_host_id.sort_values('Counts',ascending = False).Host_Id

#Setting font size for title
sns.set(font_scale = 2)

# Naming the Chart
viz_bar.set_title('Hosts with most listings in New York')

# Naming X & Y axis
viz_bar.set_xlabel('Host IDs')
viz_bar.set_ylabel('Count of listings')

#Adjusting Bar labels
viz_bar.set_xticklabels(viz_bar.get_xticklabels(), rotation=45)
```

Out[52]:

```
[Text(0, 0, '2787'),
 Text(1, 0, '2845'),
 Text(2, 0, '4632'),
 Text(3, 0, '4869'),
 Text(4, 0, '7192'),
 Text(5, 0, '7322'),
 Text(6, 0, '7356'),
 Text(7, 0, '8967'),
 Text(8, 0, '7490'),
 Text(9, 0, '7549')]
```




```
df.head()
```

Out[53]:

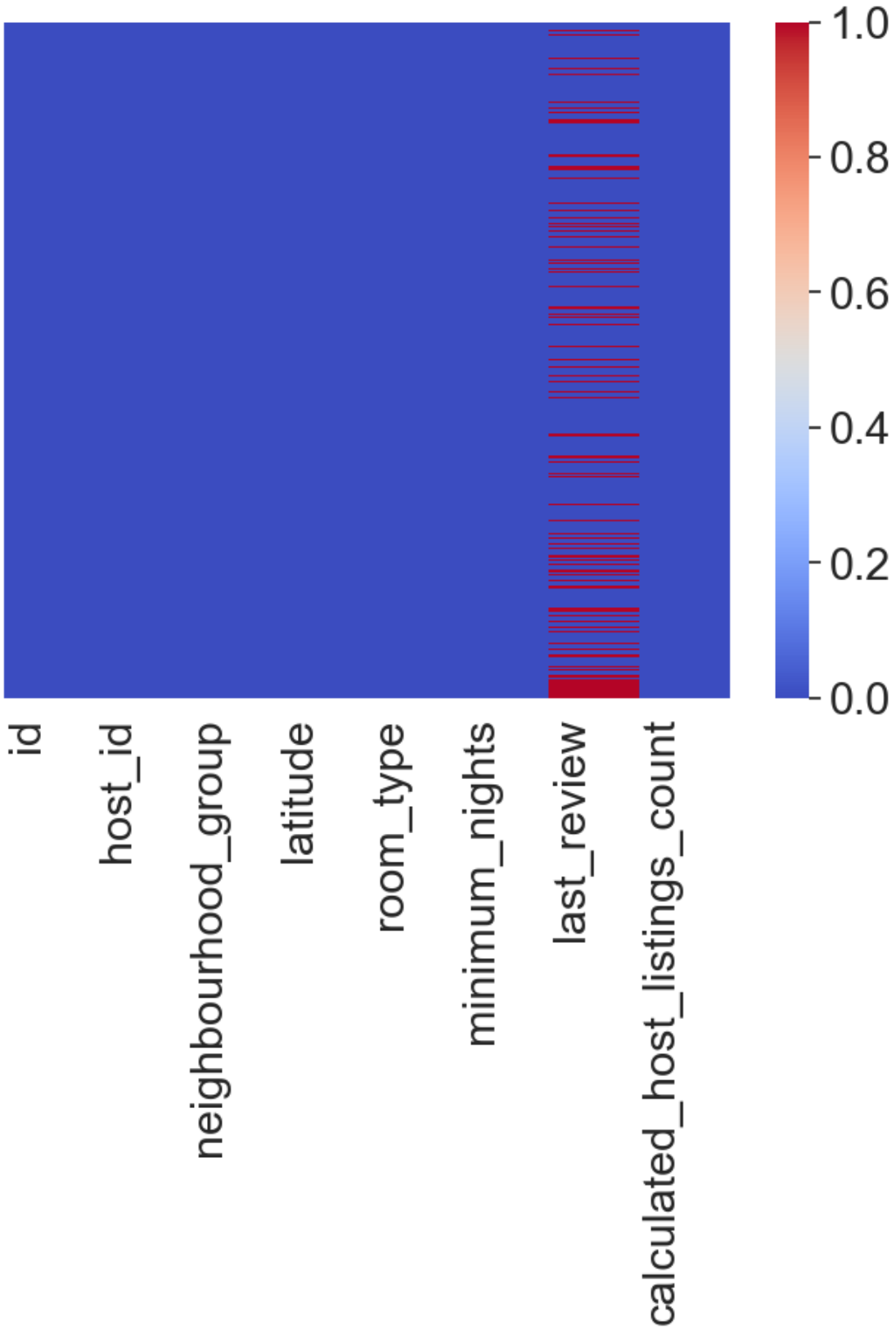
	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.7

In [60]:

```
custom_cmap = sns.color_palette("coolwarm", as_cmap=True)
plt.figure(figsize=(8,6))
# datavisualization of null value
df.isnull()
sns.heatmap(df.isnull(),cmap=custom_cmap, yticklabels=False)
```

Out[60]:

<AxesSubplot:>



In [61]:

```
# after removing last_review , reviews_per_month columns
df.drop(['last_review', 'reviews_per_month'], axis = 1,inplace=True)
```

In [62]:

```
#find any dupicated value
df.duplicated().sum()
```

Out[62]:

```
0
```

In [63]:

```
df.isnull()
```

Out[63]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	l
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
48890	False	False	False	False	False	False	False	
48891	False	False	False	False	False	False	False	
48892	False	False	False	False	False	False	False	
48893	False	False	False	False	False	False	False	
48894	False	False	False	False	False	False	False	

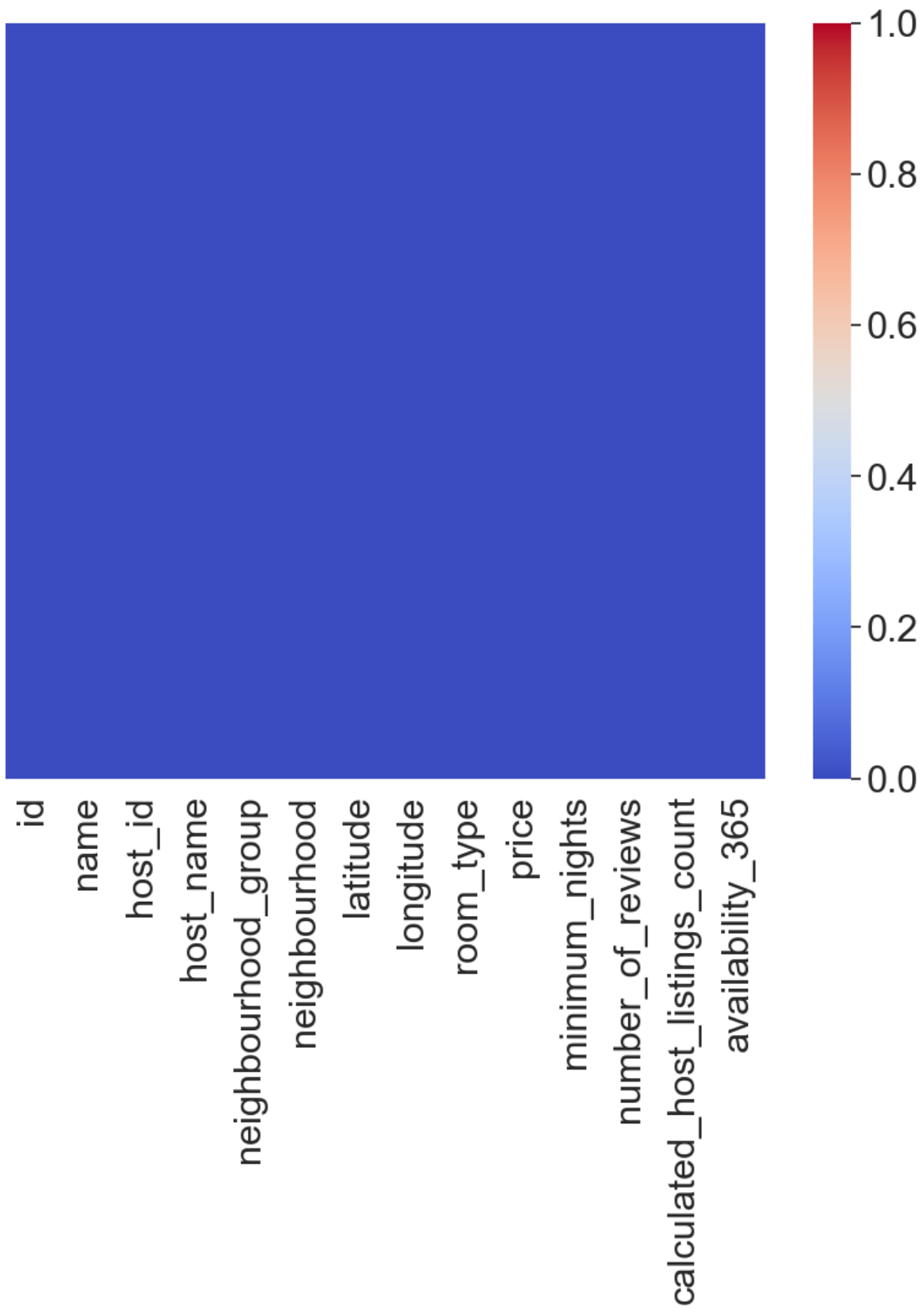
48895 rows × 14 columns

In [64]:

```
# datavisualization of null value after removing last_review , reviews_per_month columr
sns.heatmap(df.isnull(),cmap=custom_cmap, yticklabels=False)
```

Out[64]:

<AxesSubplot:>



In [65]:

```
non_null_file = df.dropna()
non_null_file.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 48858 entries, 0 to 48894
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48858 non-null  int64  
 1   name                            48858 non-null  object 
 2   host_id                         48858 non-null  int64  
 3   host_name                       48858 non-null  object 
 4   neighbourhood_group             48858 non-null  object 
 5   neighbourhood                   48858 non-null  object 
 6   latitude                        48858 non-null  float64
 7   longitude                       48858 non-null  float64
 8   room_type                       48858 non-null  object 
 9   price                           48858 non-null  int64  
 10  minimum_nights                  48858 non-null  int64  
 11  number_of_reviews               48858 non-null  int64  
 12  calculated_host_listings_count  48858 non-null  int64  
 13  availability_365                48858 non-null  int64  
dtypes: float64(2), int64(7), object(5)
memory usage: 5.6+ MB
```

6. What are the top three hosts ranked by their revenue?

In [66]:

```
new=non_null_file.copy()
new.head()
```

Out[66]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.7

In [67]:

```
new['price'].describe()
```

Out[67]:

```
count    48858.000000
mean       152.740309
std        240.232386
min          0.000000
25%         69.000000
50%        106.000000
75%        175.000000
max      10000.000000
Name: price, dtype: float64
```

In [68]:

```
# replace whose price less than 50
new.loc[new['price']<50,'price'] = new.loc[(new['price']>50) & (new['price']<80),'price
new['price'].describe()
```

Out[68]:

```
count    48858.000000
mean       155.369008
std        239.104348
min         50.000000
25%         69.000000
50%        106.000000
75%        175.000000
max      10000.000000
Name: price, dtype: float64
```

In [69]:

```
# after removing price less than 100
new.head()
```

Out[69]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.6
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.7
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.8
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.6
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.7

In [70]:

```
#  find the maximum price across different host name
top_host=new.groupby(['host_name','host_id'])['price'].sum().reset_index()
top_host.rename(columns={'price':'total_price'},inplace=True)
top_host.head()
```

Out[70]:

	host_name	host_id	total_price
0	#NAME?	128580688	150
1	'Cil	45354224	120
2	(Ari) HENRY LEE	40605120	140
3	(Email hidden by Airbnb)	5610823	261
4	(Email hidden by Airbnb)	7580102	389

In [71]:

```
# find top three host best on their turnover
top_3=top_host.sort_values('total_price',ascending=False).iloc[:3,:3]
top_3
```

Out[71]:

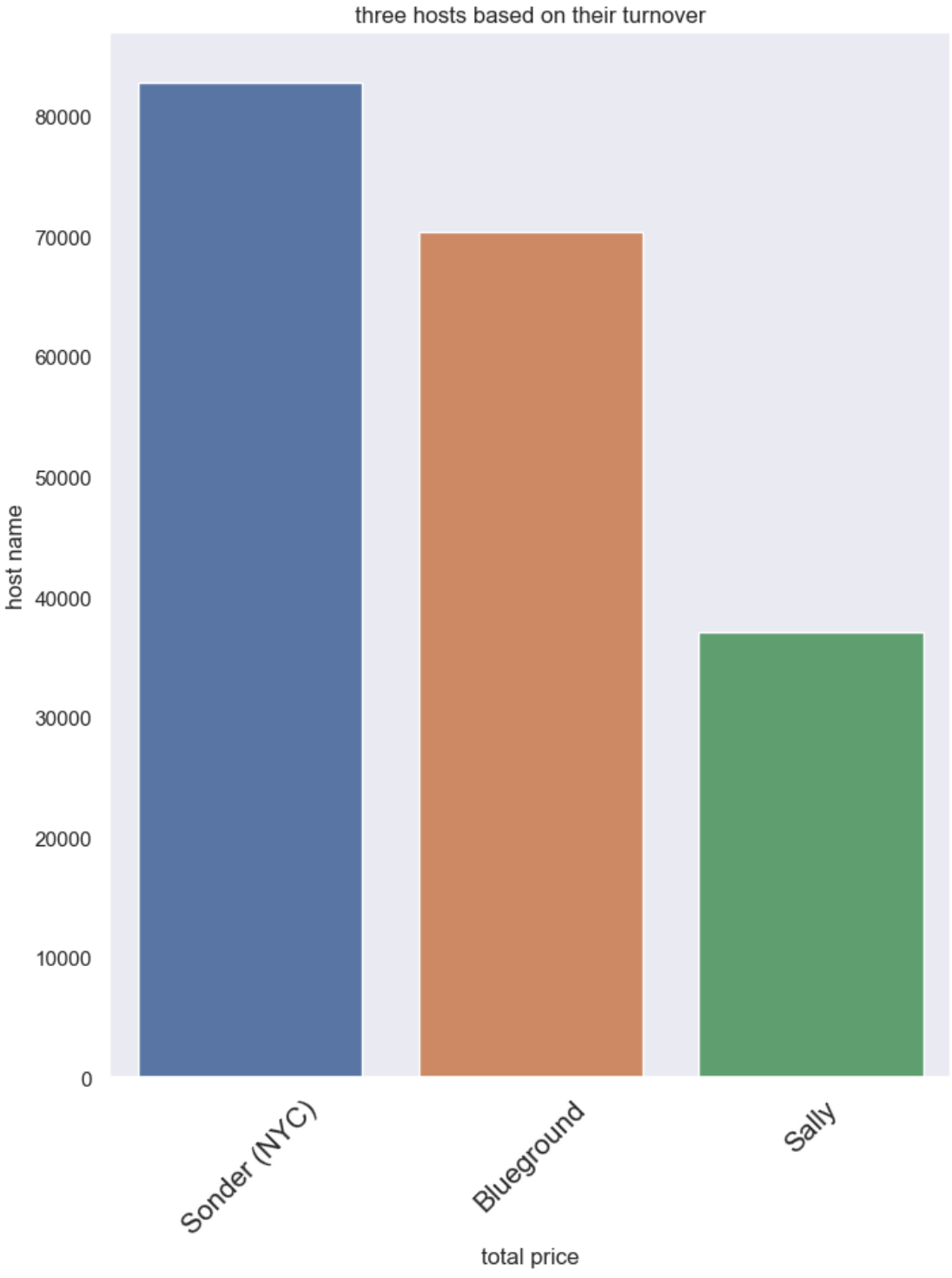
	host_name	host_id	total_price
33209	Sonder (NYC)	219517861	82795
4856	Blueground	107434423	70331
31216	Sally	156158778	37097

In [77]:

```
sns.set(rc={'figure.figsize':(8,10)})
sns.set_style('dark')
abc= sns.barplot(x='host_name',y='total_price',data = top_3)
abc.set_title('three hosts based on their turnover')
abc.set_ylabel('host name')
abc.set_xlabel('total price')

abc.set_xticklabels(abc.get_xticklabels(),rotation = 45,size='15')
```

Out[77]:

```
[Text(0, 0, 'Sonder (NYC)'), Text(1, 0, 'Blueground'), Text(2, 0, 'Sall
y')]
```



7. Which neighborhoods have the ten highest listing counts?

In [78]:

```
# find listing value of all neighourhood in dataset
neighour=new['neighbourhood'].value_counts().sort_values(ascending=False).reset_index()
#top ten neighourhood with listing value
top_10=neighour[:10]
top_10
```

Out[78]:

	index	neighbourhood
0	Williamsburg	3917
1	Bedford-Stuyvesant	3713
2	Harlem	2655
3	Bushwick	2462
4	Upper West Side	1969
5	Hell's Kitchen	1954
6	East Village	1852
7	Upper East Side	1797
8	Crown Heights	1563
9	Midtown	1545

In [80]:

```
final=top_10.copy()
#rename that dataframe
final.rename(columns={'index':'neighourhood','neighbourhood':'listing_value' },inplace=

final
```

Out[80]:

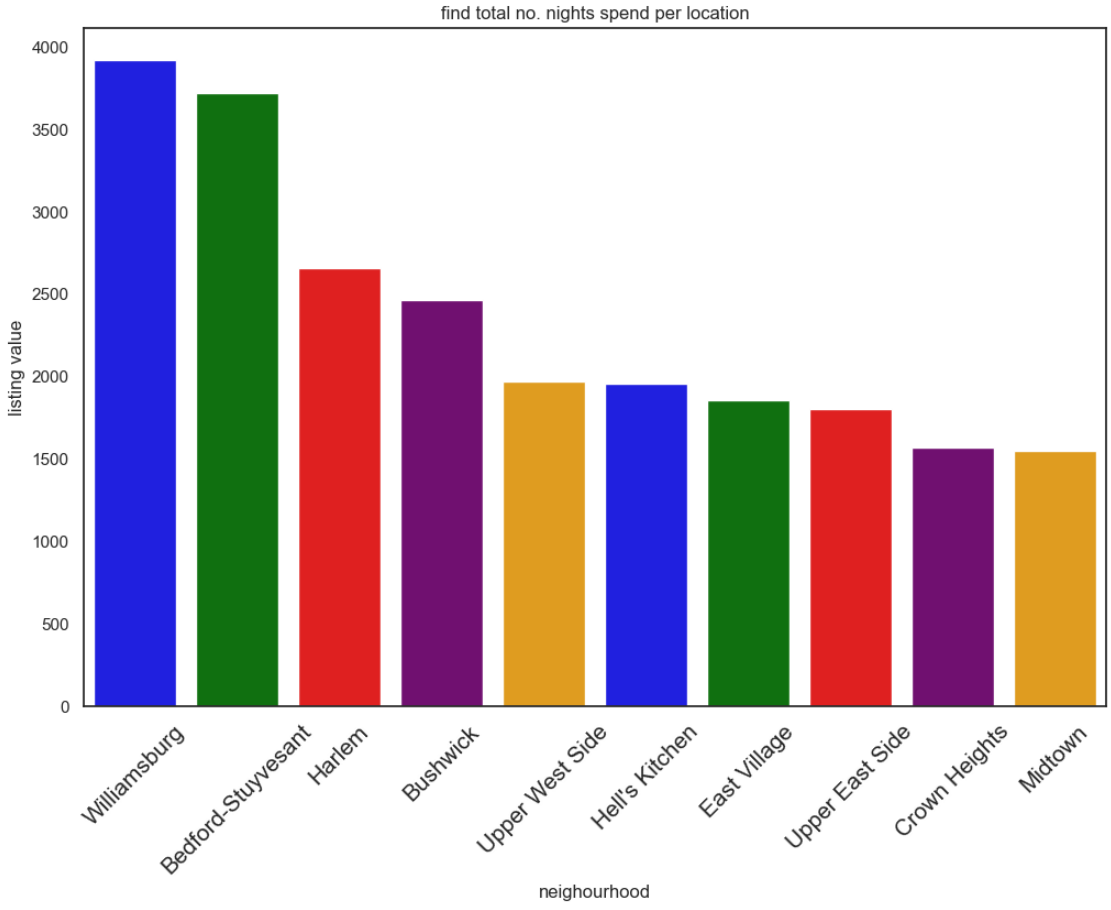
	neighourhood	listing_value
0	Williamsburg	3917
1	Bedford-Stuyvesant	3713
2	Harlem	2655
3	Bushwick	2462
4	Upper West Side	1969
5	Hell's Kitchen	1954
6	East Village	1852
7	Upper East Side	1797
8	Crown Heights	1563
9	Midtown	1545

In [83]:

```
bar_colors = ['blue', 'green', 'red', 'purple', 'orange']
# data visualizing with barplot
sns.set(rc={'figure.figsize':(12,8)})
sns.set_style('white')
#Plotting the Chart
abc= sns.barplot(x='neighourhood', y='listing_value',data = final, palette=bar_colors)
abc.set_title('find total no. nights spend per location')
# Naming X & Y axis
abc.set_ylabel('listing value')
abc.set_xlabel('neighourhood')
#Adjusting Bar labels

abc.set_xticklabels(abc.get_xticklabels(),rotation = 45,size='15')
```

Out[83]:

```
[Text(0, 0, 'Williamsburg'),
 Text(1, 0, 'Bedford-Stuyvesant'),
 Text(2, 0, 'Harlem'),
 Text(3, 0, 'Bushwick'),
 Text(4, 0, 'Upper West Side'),
 Text(5, 0, "Hell's Kitchen"),
 Text(6, 0, 'East Village'),
 Text(7, 0, 'Upper East Side'),
 Text(8, 0, 'Crown Heights'),
 Text(9, 0, 'Midtown')]
```



Conclusion:

In conclusion, Manhattan emerges as the focal point for hosts conducting their business in New York. It serves as the epicenter of hosting activities, likely due to its central and attractive location.

Brooklyn, Queens, and Manhattan stand out as the regions where customers are willing to pay the highest nightly rates, reaching up to $10,000, while there are listings available for as low as $10.

When considering room types, there are clear pricing distinctions. Entire homes and apartments command the highest average price at approximately $157, followed by private rooms at around $75, and shared rooms at roughly $60.

Notably, 'Entire home/apt' listings dominate the market with a substantial 52% share, while 'Shared Room' listings constitute a mere 2.4% of the total.

Private rooms in Brooklyn and Manhattan attract guests who prefer extended stays, indicating a preference for these areas when seeking longer-term accommodations.

Analyzing listing descriptions, we find that words like 'bedroom,' 'cozy,' 'private,' 'apartment,' and 'spacious' are recurrent, reflecting what guests value in their accommodations. In contrast, terms like 'park,' 'near,' 'village,' and 'heart' are comparatively less emphasized.

Furthermore, the dataset reveals that the top 10 hosts collectively account for almost 2.5% of all listings, totaling 1,270 properties. Among these hosts, Sonder (nyc), Red Awning, and Henry have the highest turnovers, with Sonder (nyc) standing out as the best-performing host.

Interestingly, Manhattan proves to be the preferred choice for the majority of customers, with 63.2% of them opting for stays in this vibrant borough. Only a small fraction, 1.6%, choose to spend their nights in shared rooms, indicating a strong preference for private and entire home accommodations.