Exploratory Data Analysis on NYC Airbnb 2019 dataset

Introduction

The data from this analysis is from Kaggle New York City Airhub Open Data. The data describes the listing activity and metrics in NYC , Ny for 2012includes information such as the location of the listing properties , the neighborhood of the properties , room type, price, minimum rights required review and availability of the listing/

The Purpose of this analysis is to perform, exploratory data analysis as well as data visualization to understand how different fators influence the listing properties on Airbnb and ultimately to make predicstions on the availability of the listing properties.

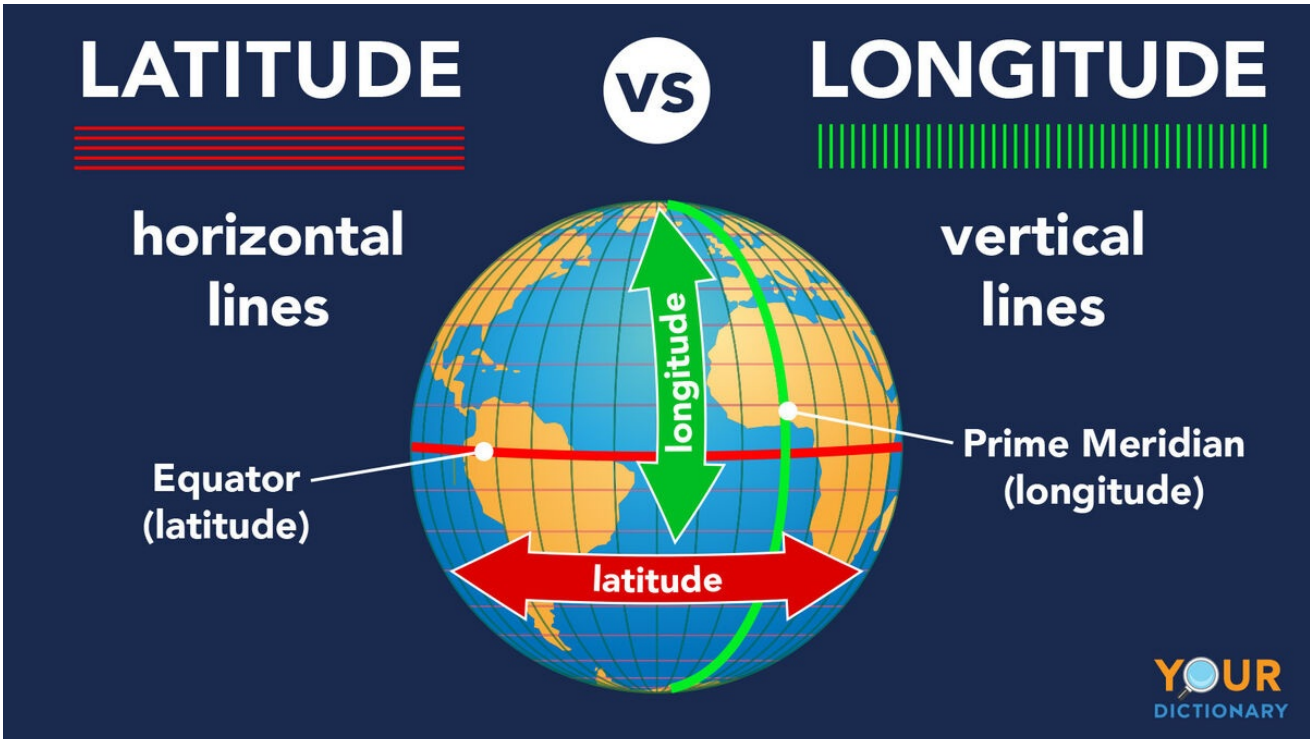
The following questioms will be answered on the course of this analysis.

- Where are the most of the properties listed smf where is the busiest areas?
- what type of rooms are most popular?
- How different area/neighborhood affect the listing property price and demands?
- What are the most important factors when customer choose an airbnb property
  - Price
  - Location
  - Room Type
  - Customer Review

  ## Data loading and Processing

  ### We start the analysys by importing necessary libraries and loading the data . The libraries used in this analysis are

  #### - Pandas #### - Numpy #### - Matplotlib #### - Seaborn #### - Sklearn #### - statsmodels



In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
%matplotlib inline
```

In [3]:
```
df=pd.read_csv(r"C:\Users\HP\Downloads\AB_NYC_2019.csv")
```

In [4]:
```
df
```

Out[4]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mini
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	
...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	
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	

48895 rows × 16 columns

In [5]:

```
# Display a concise summary of the DataFrame, including the data types and non-null counts for each column
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]:

```
# Check the number of missing values in each column of the DataFrame
df.isnull().sum()
```

Out[6]:

```
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
```

In [7]:

```
# Retrieve the unique values present in the 'name' column of the DataFrame
df.name.unique()
```

Out[7]:

```
array(['Clean & quiet apt home by the park', 'Skylit Midtown Castle',
       'THE VILLAGE OF HARLEM....NEW YORK !', ...,
       'Sunny Studio at Historical Neighborhood',
       '43rd St. Time Square-cozy single bed',
       "Trendy duplex in the very heart of Hell's Kitchen"], dtype=object)
```

In [11]:

```
# Group the DataFrame by the 'price' column and count the number of occurrences of 'latitude' for each price va
df.groupby('price').latitude.count()
```

Out[11]:

```
price
0        11
10       17
11        3
12        4
13        1
         ..
7703      1
8000      1
8500      1
9999      3
10000     3
Name: latitude, Length: 674, dtype: int64
```

In [12]:

```
# Set 'reviews_per_month' and 'last_review' to 0 for rows where 'number_of_reviews' is 0
df.loc[df.number_of_reviews==0, 'reviews_per_month'] = 0
df.loc[df.number_of_reviews==0, 'last_review'] = 0
```

In [13]:

```
df
```

Out[13]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mini
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	
...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	
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	

48895 rows × 16 columns

In [14]:

```
# Filter the DataFrame to keep only rows where 'host_id' and 'host_name' are not null
df = df[pd.notnull(df['host_id'])]
df = df[pd.notnull(df['host_name'])]
```

In [15]:

```
df
```

Out[15]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mini
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	
...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	
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	

48874 rows × 16 columns

In [16]:

```
# Sort the DataFrame based on the values in the 'latitude' column in ascending order
df.sort_values(by=['latitude'])
```

Out[16]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_
14119	10830083	Beautiful well kept private home!	56078939	Tony	Staten Island	Tottenville	40.49979	-74.24084	Private room	110	
46919	35489384	Cozy Apartment	236186921	Iveth	Staten Island	Tottenville	40.50641	-74.23059	Entire home/apt	75	
15278	12230928	Villa DiGioia visit NYC via SI	65806798	Michael J	Staten Island	Tottenville	40.50708	-74.24285	Private room	100	
1424	639199	Beautiful 4BR/4BA Home, Staten Island, NY City.	1483081	Marina	Staten Island	Tottenville	40.50868	-74.23986	Entire home/apt	299	
23460	18997371	Cozy Getaway	90104417	Sueann	Staten Island	Tottenville	40.50873	-74.23914	Entire home/apt	85	
...	...	...	...	...	...	...	...	...	...	...	
3349	2008227	Private Studio in Private Home	9539641	Dianne	Bronx	North Riverdale	40.90804	-73.90005	Private room	53	
48033	36041232	Nice house room 2 near van cortlandt park	230720704	Pp	Bronx	North Riverdale	40.91167	-73.89566	Private room	40	
23011	18635370	Fantastic Sunny peaceful room in Riverdale	91385196	Vicdania	Bronx	North Riverdale	40.91169	-73.90564	Private room	50	
47790	35916310	Nice house private room	230720704	Pp	Bronx	North Riverdale	40.91234	-73.89417	Private room	40	
48029	36040561	Nice room to rent 1	230720704	Pp	Bronx	North Riverdale	40.91306	-73.89389	Private room	40	

48874 rows × 16 columns

In [17]:

```
# Sort the DataFrame based on the values in the 'longitude' column in ascending order
df.sort_values(by=['longitude'])
```

Out[17]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimu
45652	34888503	Charming town of Tottenville right outside NYC	962249	Dora	Staten Island	Tottenville	40.50943	-74.24442	Entire home/apt	70	
15278	12230928	Villa DiGioia visit NYC via SI	65806798	Michael J	Staten Island	Tottenville	40.50708	-74.24285	Private room	100	
14119	10830083	Beautiful well kept private home!	56078939	Tony	Staten Island	Tottenville	40.49979	-74.24084	Private room	110	
1424	639199	Beautiful 4BR/4BA Home, Staten Island, NY City.	1483081	Marina	Staten Island	Tottenville	40.50868	-74.23986	Entire home/apt	299	
23460	18997371	Cozy Getaway	90104417	Sueann	Staten Island	Tottenville	40.50873	-74.23914	Entire home/apt	85	
...	...	...	...	...	...	...	...	...	...	...	
38562	30325639	Cozy shared studio in a safe neighborhood	21495656	Ramy	Queens	Little Neck	40.76212	-73.71928	Shared room	32	
45592	34844239	❀ Bright and cozy townhouse | Ideal for famili...	154268909	Malik	Queens	Bellerose	40.74027	-73.71829	Entire home/apt	180	
11610	9031216	upstairs apartment private, spacious	47140247	Hilary	Queens	Bellerose	40.72756	-73.71795	Entire home/apt	42	
47208	35638944	☀Bright & sunny townhouse | Perfect for famili...	154268909	Malik	Queens	Bellerose	40.74006	-73.71690	Entire home/apt	240	
10920	8423666	"Bloom of Floral Park" 1 BR Basement Suite	44361695	Mordeana	Queens	Bellerose	40.73351	-73.71299	Private room	65	

48874 rows × 16 columns

In [18]:

```python
# Sort the DataFrame based on the values in the 'price' column in ascending order
df.sort_values(by=['price'])
```

Out[18]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimun
23161	18750597	Huge Brooklyn Brownstone Living, Close to it all.	8993084	Kimberly	Brooklyn	Bedford-Stuyvesant	40.69023	-73.95428	Private room	0	
25794	20639628	Spacious comfortable master bedroom with nice ...	86327101	Adeyemi	Brooklyn	Bedford-Stuyvesant	40.68173	-73.91342	Private room	0	
26259	20933849	the best you can find	13709292	Qiuchi	Manhattan	Murray Hill	40.75091	-73.97597	Entire home/apt	0	
26866	21304320	Best Coliving space ever! Shared room.	101970559	Sergii	Brooklyn	Bushwick	40.69166	-73.90928	Shared room	0	
26841	21291569	Coliving in Brooklyn! Modern design / Shared room	101970559	Sergii	Brooklyn	Bushwick	40.69211	-73.90670	Shared room	0	
...	...	...	...	...	...	...	...	...	...	...	
6530	4737930	Spanish Harlem Apt	1235070	Olson	Manhattan	East Harlem	40.79264	-73.93898	Entire home/apt	9999	
12342	9528920	Quiet, Clean, Lit @ LES & Chinatown	3906464	Amy	Manhattan	Lower East Side	40.71355	-73.98507	Private room	9999	
17692	13894339	Luxury 1 bedroom apt. - stunning Manhattan views	5143901	Erin	Brooklyn	Greenpoint	40.73260	-73.95739	Entire home/apt	10000	
29238	22436899	1-BR Lincoln Center	72390391	Jelena	Manhattan	Upper West Side	40.77213	-73.98665	Entire home/apt	10000	
9151	7003697	Furnished room in Astoria apartment	20582832	Kathrine	Queens	Astoria	40.76810	-73.91651	Private room	10000	

48874 rows × 16 columns

In [19]:

```
# Calculate the mean value of the 'price' column in the DataFrame
np.mean(df.price)
```

Out[19]:

```
152.7386340385481
```

In [21]:

```
# Install the matplotlib library for data visualization
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\hp\anaconda3\lib\site-packages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (1
.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (0.11.
0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (
4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (
1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (1.24.3
)
Requirement already satisfied: packaging>=20.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (23
.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (9.4.
0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotli
b) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib
) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-packages (from python-dateutil>=2.7->
matplotlib) (1.16.0)
```

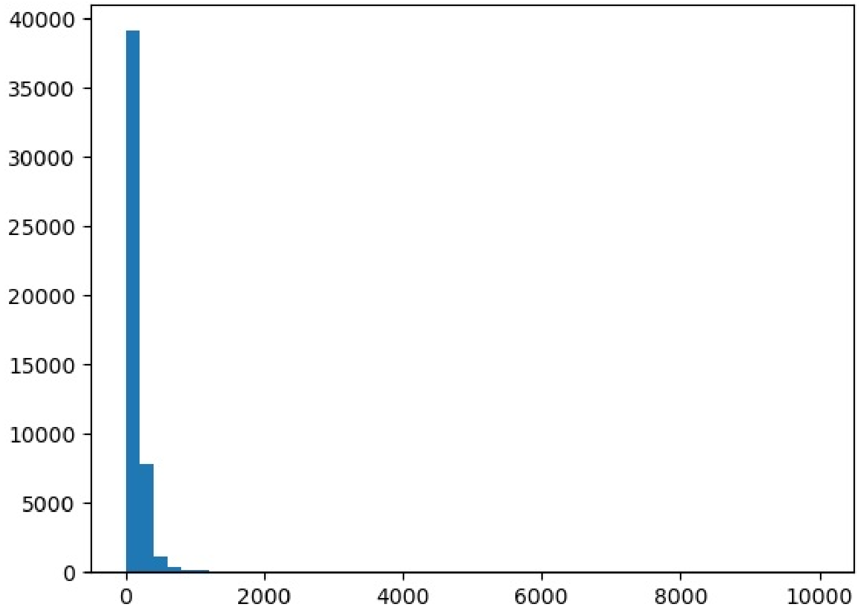
In [22]:

```
import matplotlib.pyplot as plt
# Create a histogram of the 'price' column with 50 bins
```

```
plt.hist(df.price,bins=50)
```

Out[22]:

```
(array([3.9091e+04, 7.7950e+03, 1.1290e+03, 3.8500e+02, 1.7600e+02,
        9.1000e+01, 3.4000e+01, 3.7000e+01, 1.7000e+01, 9.0000e+00,
        2.7000e+01, 5.0000e+00, 1.4000e+01, 4.0000e+00, 7.0000e+00,
        8.0000e+00, 2.0000e+00, 2.0000e+00, 4.0000e+00, 2.0000e+00,
        5.0000e+00, 2.0000e+00, 2.0000e+00, 0.0000e+00, 0.0000e+00,
        7.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        2.0000e+00, 0.0000e+00, 4.0000e+00, 0.0000e+00, 1.0000e+00,
        0.0000e+00, 0.0000e+00, 2.0000e+00, 1.0000e+00, 0.0000e+00,
        1.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 6.0000e+00]),
 array([    0.,   200.,   400.,   600.,   800.,  1000.,  1200.,  1400.,
         1600.,  1800.,  2000.,  2200.,  2400.,  2600.,  2800.,  3000.,
         3200.,  3400.,  3600.,  3800.,  4000.,  4200.,  4400.,  4600.,
         4800.,  5000.,  5200.,  5400.,  5600.,  5800.,  6000.,  6200.,
         6400.,  6600.,  6800.,  7000.,  7200.,  7400.,  7600.,  7800.,
         8000.,  8200.,  8400.,  8600.,  8800.,  9000.,  9200.,  9400.,
         9600.,  9800., 10000.]),
 <BarContainer object of 50 artists>)
```



In [18]:

```
# Count the number of rows where the 'price' column is greater than 2000
len(df[df.price > 2000])
```

Out[18]:

```
86
```

In [23]:

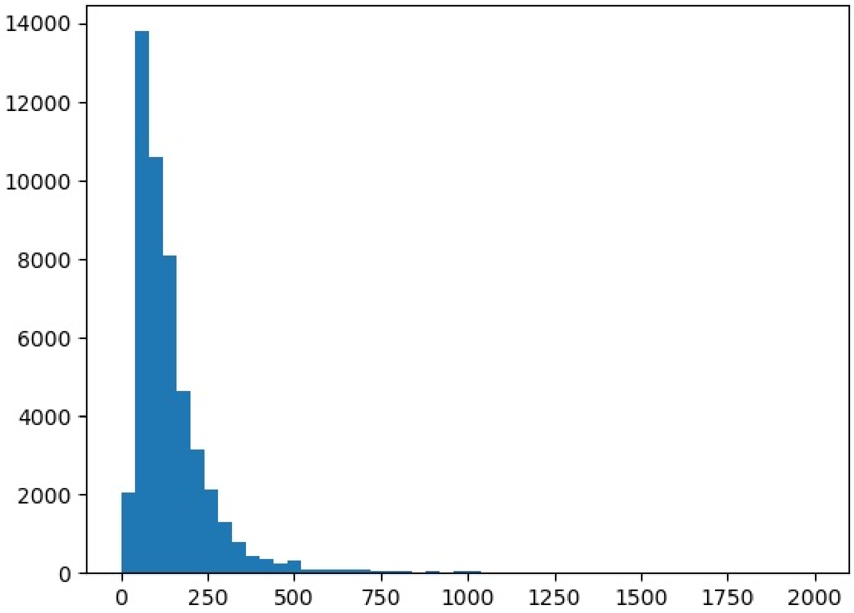
```
# Filter the DataFrame to include only rows where the 'price' column is less than 2000
df=df[df.price < 2000]
```

In [24]:

```
# Create a histogram of the 'price' column with 50 bins after filtering
plt.hist(df['price'], bins=50, edgecolor='black')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.title('Histogram of Prices (Filtered)')
plt.show()
plt.hist(df.price,bins=50)
```

Out[24]:

```
(array([2.0400e+03, 1.3778e+04, 1.0593e+04, 8.0680e+03, 4.6120e+03,
        3.1470e+03, 2.1220e+03, 1.3110e+03, 7.8400e+02, 4.3100e+02,
        3.6500e+02, 2.5800e+02, 3.3400e+02, 9.8000e+01, 7.4000e+01,
        9.3000e+01, 8.5000e+01, 1.0500e+02, 6.8000e+01, 3.4000e+01,
        6.1000e+01, 2.6000e+01, 3.5000e+01, 2.2000e+01, 3.2000e+01,
        6.1000e+01, 7.0000e+00, 1.6000e+01, 3.0000e+00, 4.0000e+00,
        1.5000e+01, 6.0000e+00, 8.0000e+00, 3.0000e+00, 2.0000e+00,
        2.0000e+00, 2.0000e+00, 3.0000e+01, 1.0000e+00, 2.0000e+00,
        4.0000e+00, 0.0000e+00, 5.0000e+00, 5.0000e+00, 3.0000e+00,
        2.0000e+00, 0.0000e+00, 3.0000e+00, 0.0000e+00, 4.0000e+00]),
 array([   0.  ,   39.98,   79.96,  119.94,  159.92,  199.9 ,  239.88,
         279.86,  319.84,  359.82,  399.8 ,  439.78,  479.76,  519.74,
         559.72,  599.7 ,  639.68,  679.66,  719.64,  759.62,  799.6 ,
         839.58,  879.56,  919.54,  959.52,  999.5 , 1039.48, 1079.46,
        1119.44, 1159.42, 1199.4 , 1239.38, 1279.36, 1319.34, 1359.32,
        1399.3 , 1439.28, 1479.26, 1519.24, 1559.22, 1599.2 , 1639.18,
        1679.16, 1719.14, 1759.12, 1799.1 , 1839.08, 1879.06, 1919.04,
        1959.02, 1999.  ]),
 <BarContainer object of 50 artists>)
```



In [21]:

```
# Count the number of rows where the 'price' column is greater than 1000
len(df[df.price > 1000])
```

Out[21]:

```
129
```

In [25]:

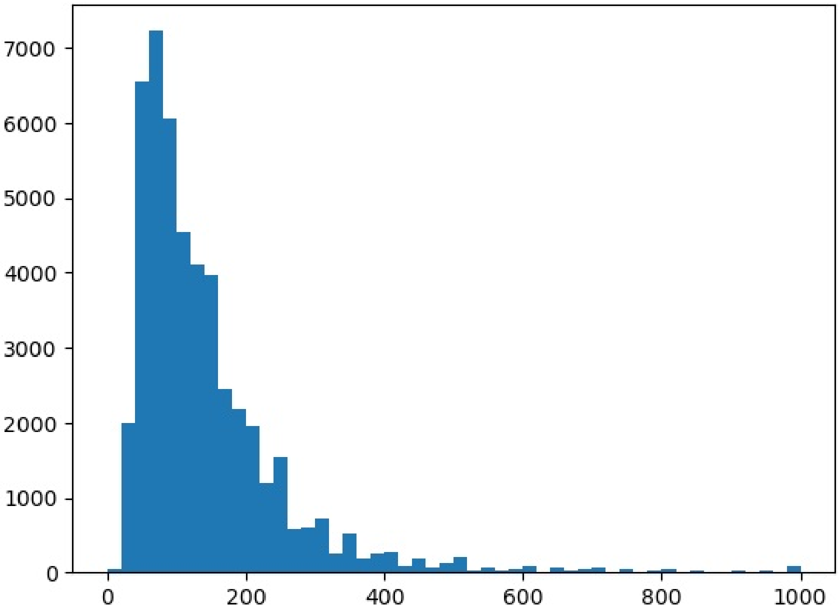
```
# Filter the DataFrame to include only rows where the 'price' column is less than or equal to 1000
df = df[df.price <=1000]
```

In [26]:

```
# Create a histogram of the 'price' column with 50 bins after further filtering
plt.hist(df['price'], bins=50, edgecolor='black')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.title('Histogram of Prices (Filtered to <= 1000)')
plt.show()
plt.hist(df.price, bins=50)
```

Out[26]:

```
(array([5.400e+01, 1.986e+03, 6.558e+03, 7.220e+03, 6.049e+03, 4.544e+03,
        4.106e+03, 3.962e+03, 2.437e+03, 2.175e+03, 1.947e+03, 1.200e+03,
        1.541e+03, 5.810e+02, 5.970e+02, 7.140e+02, 2.590e+02, 5.250e+02,
        1.830e+02, 2.480e+02, 2.700e+02, 9.500e+01, 1.970e+02, 6.100e+01,
        1.300e+02, 2.040e+02, 3.200e+01, 6.600e+01, 2.200e+01, 5.200e+01,
        8.600e+01, 7.000e+00, 6.800e+01, 1.700e+01, 4.500e+01, 6.000e+01,
        1.200e+01, 5.600e+01, 3.000e+00, 3.100e+01, 5.600e+01, 5.000e+00,
        2.300e+01, 3.000e+00, 1.400e+01, 2.100e+01, 3.000e+00, 1.900e+01,
        5.000e+00, 8.600e+01]),
 array([   0.,   20.,   40.,   60.,   80.,  100.,  120.,  140.,  160.,
         180.,  200.,  220.,  240.,  260.,  280.,  300.,  320.,  340.,
         360.,  380.,  400.,  420.,  440.,  460.,  480.,  500.,  520.,
         540.,  560.,  580.,  600.,  620.,  640.,  660.,  680.,  700.,
         720.,  740.,  760.,  780.,  800.,  820.,  840.,  860.,  880.,
         900.,  920.,  940.,  960.,  980., 1000.]),
 <BarContainer object of 50 artists>)
```



In [27]:

```
# Sort the DataFrame based on the values in the 'minimum_nights' column in ascending order
df.sort_values(by=['minimum_nights'])
```

Out[27]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
37538	29781403	Modern Apartment in Brooklyn with deck and yard	22464812	Maruf	Brooklyn	Kensington	40.64261	-73.98350	Entire home/apt	200	
14332	11164047	Comfy Brownstone Room in Brooklyn	58070616	Brandon	Brooklyn	Bedford-Stuyvesant	40.69032	-73.93699	Private room	40	
14333	11164599	Top Floor Apartment with Roof Access.	5162894	Catherine	Manhattan	Midtown	40.74389	-73.98515	Private room	120	
37536	29780863	Private Studio Chelsea 23 x 8th Ave 30sec to t...	6458347	Esther	Manhattan	Chelsea	40.74499	-73.99845	Private room	180	
...	...	...	...	...	...	...	...	...	...	...	
26341	20990053	Beautiful place in Brooklyn! #2	151084261	Angie	Brooklyn	Williamsburg	40.71772	-73.95059	Private room	79	
13404	10053943	Historic Designer 2 Bed. Apartment	2697686	Glenn H.	Manhattan	Harlem	40.82915	-73.94034	Entire home/apt	99	
38664	30378211	Shared Studio (females only)	200401254	Meg	Manhattan	Greenwich Village	40.73094	-73.99900	Shared room	110	
2854	1615764	NaN	6676776	Peter	Manhattan	Battery Park City	40.71239	-74.01620	Entire home/apt	400	
5767	4204302	Prime W. Village location 1 bdrm	17550546	Genevieve	Manhattan	Greenwich Village	40.73293	-73.99782	Entire home/apt	180	

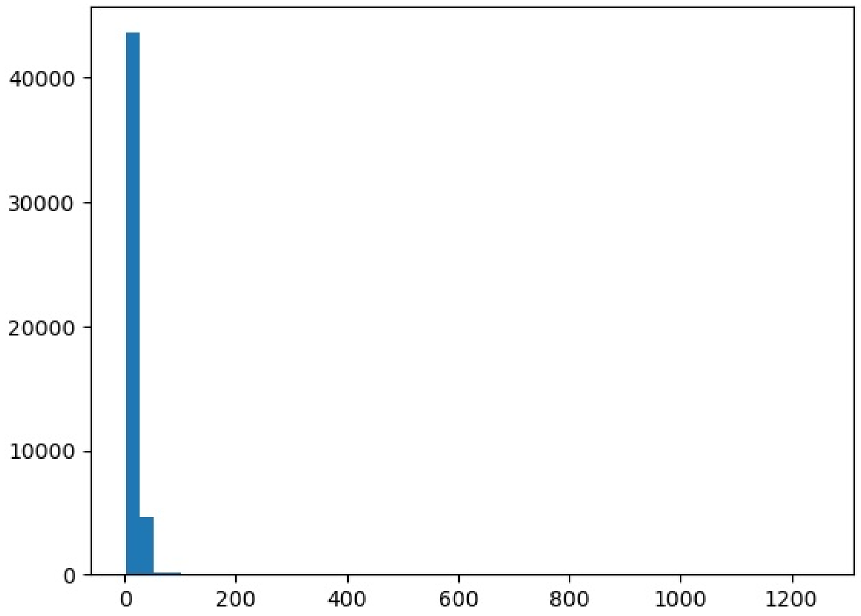
48635 rows × 16 columns

In [28]:

```python
import matplotlib.pyplot as plt

# Create a histogram of the 'minimum_nights' column with 50 bins
plt.hist(df['minimum_nights'], bins=50, edgecolor='black')
plt.xlabel('Minimum Nights')
plt.ylabel('Frequency')
plt.title('Histogram of Minimum Nights')
plt.show()
plt.hist(df.minimum_nights,bins=50)
```

Out[28]:

```
(array([4.3607e+04, 4.5960e+03, 1.3400e+02, 1.3300e+02, 3.3000e+01,
        9.0000e+00, 4.0000e+00, 5.3000e+01, 4.0000e+00, 4.0000e+00,
        4.0000e+00, 7.0000e+00, 0.0000e+00, 0.0000e+00, 3.5000e+01,
        1.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 6.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 4.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 1.0000e+00]),
 array([1.00000e+00, 2.59800e+01, 5.09600e+01, 7.59400e+01, 1.00920e+02,
        1.25900e+02, 1.50880e+02, 1.75860e+02, 2.00840e+02, 2.25820e+02,
        2.50800e+02, 2.75780e+02, 3.00760e+02, 3.25740e+02, 3.50720e+02,
        3.75700e+02, 4.00680e+02, 4.25660e+02, 4.50640e+02, 4.75620e+02,
        5.00600e+02, 5.25580e+02, 5.50560e+02, 5.75540e+02, 6.00520e+02,
        6.25500e+02, 6.50480e+02, 6.75460e+02, 7.00440e+02, 7.25420e+02,
        7.50400e+02, 7.75380e+02, 8.00360e+02, 8.25340e+02, 8.50320e+02,
        8.75300e+02, 9.00280e+02, 9.25260e+02, 9.50240e+02, 9.75220e+02,
        1.00020e+03, 1.02518e+03, 1.05016e+03, 1.07514e+03, 1.10012e+03,
        1.12510e+03, 1.15008e+03, 1.17506e+03, 1.20004e+03, 1.22502e+03,
        1.25000e+03]),
 <BarContainer object of 50 artists>)
```



In [29]:

```python
# Count the number of rows where the 'minimum_nights' column is greater than 200
len(df[df.minimum_nights > 200])
```

Out[29]:

```
66
```

In [30]:

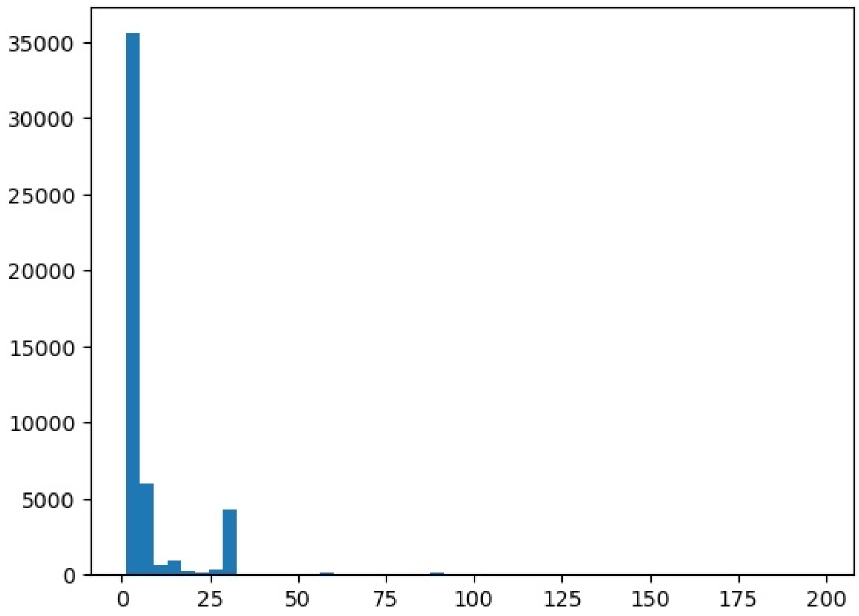
```python
# Filter the DataFrame to include only rows where the 'minimum_nights' column is less than 200
df = df[df.minimum_nights < 200]
```

In [31]:

```python
import matplotlib.pyplot as plt

# Create a histogram of the 'minimum_nights' column with 50 bins after filtering
plt.hist(df['minimum_nights'], bins=50, edgecolor='black')
plt.xlabel('Minimum Nights')
plt.ylabel('Frequency')
plt.title('Histogram of Minimum Nights (Filtered)')
plt.show()
plt.hist(df.minimum_nights, bins=50)
```

Out[31]:

```
(array([3.5546e+04, 5.9500e+03, 6.8300e+02, 9.1100e+02, 2.7200e+02,
        1.6300e+02, 3.1800e+02, 4.2780e+03, 1.8000e+01, 1.3000e+01,
        4.0000e+00, 3.4000e+01, 1.3000e+01, 8.0000e+00, 1.0900e+02,
        3.0000e+00, 1.0000e+00, 8.0000e+00, 5.0000e+00, 0.0000e+00,
        8.0000e+00, 2.0000e+00, 1.0600e+02, 2.0000e+00, 2.0000e+00,
        1.3000e+01, 2.0000e+00, 3.0000e+00, 2.0000e+00, 0.0000e+00,
        2.6000e+01, 0.0000e+00, 0.0000e+00, 3.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 6.0000e+00, 1.0000e+00, 0.0000e+00,
        2.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 1.0000e+00,
        4.4000e+01, 4.0000e+00, 0.0000e+00, 0.0000e+00, 1.0000e+00]),
 array([  1.  ,   4.94,   8.88,  12.82,  16.76,  20.7 ,  24.64,  28.58,
         32.52,  36.46,  40.4 ,  44.34,  48.28,  52.22,  56.16,  60.1 ,
         64.04,  67.98,  71.92,  75.86,  79.8 ,  83.74,  87.68,  91.62,
         95.56,  99.5 , 103.44, 107.38, 111.32, 115.26, 119.2 , 123.14,
        127.08, 131.02, 134.96, 138.9 , 142.84, 146.78, 150.72, 154.66,
        158.6 , 162.54, 166.48, 170.42, 174.36, 178.3 , 182.24, 186.18,
        190.12, 194.06, 198.  ]),
 <BarContainer object of 50 artists>)
```



In [32]:

```
# Count the number of rows where the 'minimum_nights' column is greater than 100
len(df[df.minimum_nights > 100])
```

Out[32]:

```
95
```

In [33]:

```
# Sort the DataFrame based on the values in the 'number_of_reviews' column in ascending order
df.sort_values(by=['number_of_reviews'])
```

Out[33]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_
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	
40761	31650146	Gorgeous Spacious 1BR in Prime Lower East Side	1306854	Ani	Manhattan	Lower East Side	40.71991	-73.98505	Entire home/apt	110	
40760	31649210	one bedroom	111586798	Percival	Bronx	Soundview	40.82121	-73.87764	Private room	65	
40758	31647962	Spacious and stylish Harlem apartment	237280886	Nicola	Manhattan	Harlem	40.82636	-73.94985	Entire home/apt	95	
13619	10192898	Heart of West Village, over NYE!	7108710	Katie	Manhattan	West Village	40.73440	-74.00262	Private room	105	
...	...	...	...	...	...	...	...	...	...	...	
13495	10101135	Room Near JFK Twin Beds	47621202	Dona	Queens	Jamaica	40.66939	-73.76975	Private room	47	
2015	891117	Private Bedroom in Manhattan	4734398	Jj	Manhattan	Harlem	40.82264	-73.94041	Private room	49	
2030	903947	Beautiful Bedroom in Manhattan	4734398	Jj	Manhattan	Harlem	40.82124	-73.93838	Private room	49	
2031	903972	Great Bedroom in Manhattan	4734398	Jj	Manhattan	Harlem	40.82085	-73.94025	Private room	49	
11759	9145202	Room near JFK Queen Bed	47621202	Dona	Queens	Jamaica	40.66730	-73.76831	Private room	47	

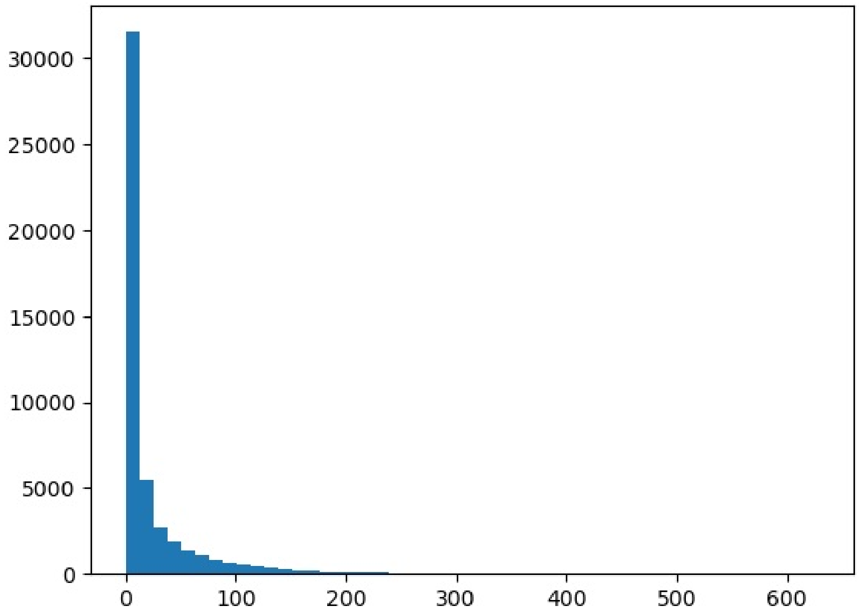
48565 rows × 16 columns

In [34]:

```
import matplotlib.pyplot as plt

# Create a histogram of the 'number_of_reviews' column with 50 bins
plt.hist(df['number_of_reviews'], bins=50, edgecolor='black')
plt.xlabel('Number of Reviews')
plt.ylabel('Frequency')
plt.title('Histogram of Number of Reviews')
plt.show()
plt.hist(df.number_of_reviews, bins=50)
```

Out[34]:

```
(array([3.1513e+04, 5.4690e+03, 2.7130e+03, 1.9280e+03, 1.3600e+03,
        1.1180e+03, 8.4900e+02, 6.2700e+02, 5.3000e+02, 4.3600e+02,
        3.9500e+02, 2.8600e+02, 2.3800e+02, 1.9100e+02, 1.5400e+02,
        1.3400e+02, 1.3300e+02, 9.5000e+01, 8.5000e+01, 5.2000e+01,
        4.4000e+01, 3.7000e+01, 2.7000e+01, 2.0000e+01, 1.3000e+01,
        2.6000e+01, 1.3000e+01, 1.0000e+01, 1.1000e+01, 7.0000e+00,
        6.0000e+00, 8.0000e+00, 8.0000e+00, 4.0000e+00, 5.0000e+00,
        4.0000e+00, 3.0000e+00, 3.0000e+00, 2.0000e+00, 0.0000e+00,
        1.0000e+00, 0.0000e+00, 1.0000e+00, 1.0000e+00, 0.0000e+00,
        1.0000e+00, 0.0000e+00, 2.0000e+00, 1.0000e+00, 1.0000e+00]),
 array([  0.  ,  12.58,  25.16,  37.74,  50.32,  62.9 ,  75.48,  88.06,
        100.64, 113.22, 125.8 , 138.38, 150.96, 163.54, 176.12, 188.7 ,
        201.28, 213.86, 226.44, 239.02, 251.6 , 264.18, 276.76, 289.34,
        301.92, 314.5 , 327.08, 339.66, 352.24, 364.82, 377.4 , 389.98,
        402.56, 415.14, 427.72, 440.3 , 452.88, 465.46, 478.04, 490.62,
        503.2 , 515.78, 528.36, 540.94, 553.52, 566.1 , 578.68, 591.26,
        603.84, 616.42, 629.  ]),
 <BarContainer object of 50 artists>)
```



In [35]:
```
# Count the number of listings that have received more than 300 reviews
len(df[df.number_of_reviews > 300])
```

Out[35]:
```
131
```

In [36]:
```
# Count the number of listings that have received more than 400 reviews
len(df[df.number_of_reviews > 400])
```

Out[36]:
```
39
```

In [37]:
```
# Filter the DataFrame to include only listings with 400 or fewer reviews
df=df[df.number_of_reviews <=400]
```

In [38]:
```
# Sort the DataFrame by the number of listings each host has, in ascending order
df.sort_values(by=['calculated_host_listings_count'])
```

Out[38]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minin
24480	19699218	Clean, Big, Sunny Room in Little Italy/Chinatown!	138784297	Andre	Manhattan	Chinatown	40.71353	-73.99632	Private room	115	
28006	21885667	Private Bedroom in Spacious Queens Home	11911154	Flor De Liz	Queens	Ditmars Steinway	40.77129	-73.91712	Private room	35	
28007	21885677	Cozy apt near Bloomingdales and Central Park.	70055156	Yngridd	Manhattan	Midtown	40.75893	-73.96360	Entire home/apt	200	
28008	21885860	HUGE Bedroom in Brooklyn Off Lorimer J/M/Z & L	3105557	Joshua	Brooklyn	Williamsburg	40.70383	-73.94431	Private room	43	
28009	21885914	Gorgeous 1 BR in heart of Prospect Heights!	7683267	Nicholas	Brooklyn	Prospect Heights	40.67933	-73.96912	Entire home/apt	100	
...	...	...	...	...	...	...	...	...	...	...	
44180	34087750	Sonder | 116 John | Ideal 1BR + Gym	219517861	Sonder (NYC)	Manhattan	Financial District	40.70722	-74.00499	Entire home/apt	164	
44552	34289331	Sonder | 11th Ave | Sunny 1BR + Gym	219517861	Sonder (NYC)	Manhattan	Hell's Kitchen	40.76070	-73.99610	Entire home/apt	189	
44426	34214603	Sonder | 11th Ave | Vibrant 1BR + Gym	219517861	Sonder (NYC)	Manhattan	Hell's Kitchen	40.76198	-73.99644	Entire home/apt	184	
44178	34087090	Sonder | Stock Exchange | Gorgeous 1BR + Kitchen	219517861	Sonder (NYC)	Manhattan	Financial District	40.70588	-74.01214	Entire home/apt	230	
39774	30937597	Sonder | The Nash | Pristine Studio + Gym	219517861	Sonder (NYC)	Manhattan	Murray Hill	40.74884	-73.97589	Entire home/apt	252	

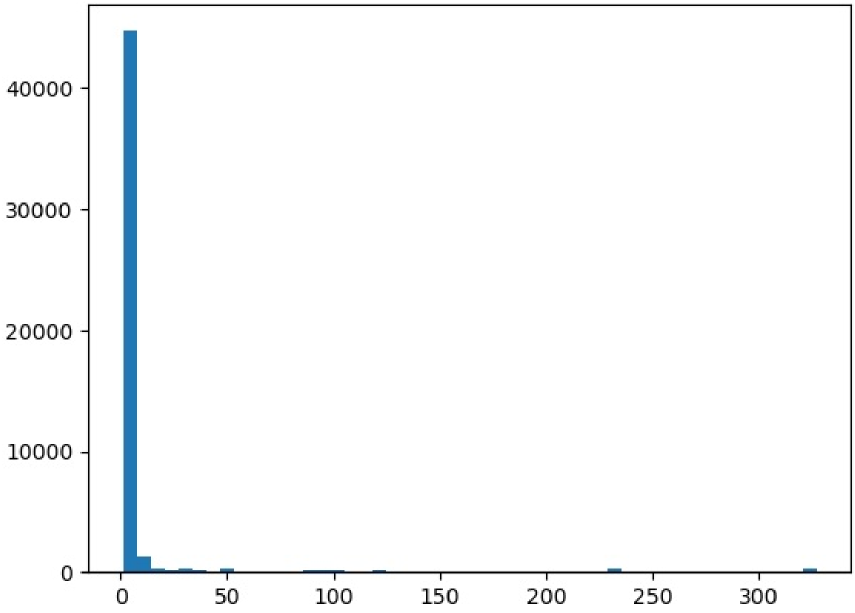
48526 rows × 16 columns

In [39]:

```
# Create a histogram to visualize the distribution of the number of listings per host
plt.hist(df.calculated_host_listings_count, bins=50)
```

Out[39]:

```
(array([4.4724e+04, 1.3280e+03, 2.7100e+02, 1.9300e+02, 3.0800e+02,
        1.4400e+02, 4.3000e+01, 2.9900e+02, 0.0000e+00, 6.5000e+01,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 1.7800e+02, 1.9200e+02,
        1.0300e+02, 0.0000e+00, 0.0000e+00, 1.1900e+02, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        2.3200e+02, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 3.2700e+02]),
 array([  1.  ,   7.52,  14.04,  20.56,  27.08,  33.6 ,  40.12,  46.64,
         53.16,  59.68,  66.2 ,  72.72,  79.24,  85.76,  92.28,  98.8 ,
        105.32, 111.84, 118.36, 124.88, 131.4 , 137.92, 144.44, 150.96,
        157.48, 164.  , 170.52, 177.04, 183.56, 190.08, 196.6 , 203.12,
        209.64, 216.16, 222.68, 229.2 , 235.72, 242.24, 248.76, 255.28,
        261.8 , 268.32, 274.84, 281.36, 287.88, 294.4 , 300.92, 307.44,
        313.96, 320.48, 327.  ]),
 <BarContainer object of 50 artists>)
```



In [40]:

```
# Sort the DataFrame by the availability of listings for the entire year, in ascending order
df.sort_values(by=['availability_365'])
```

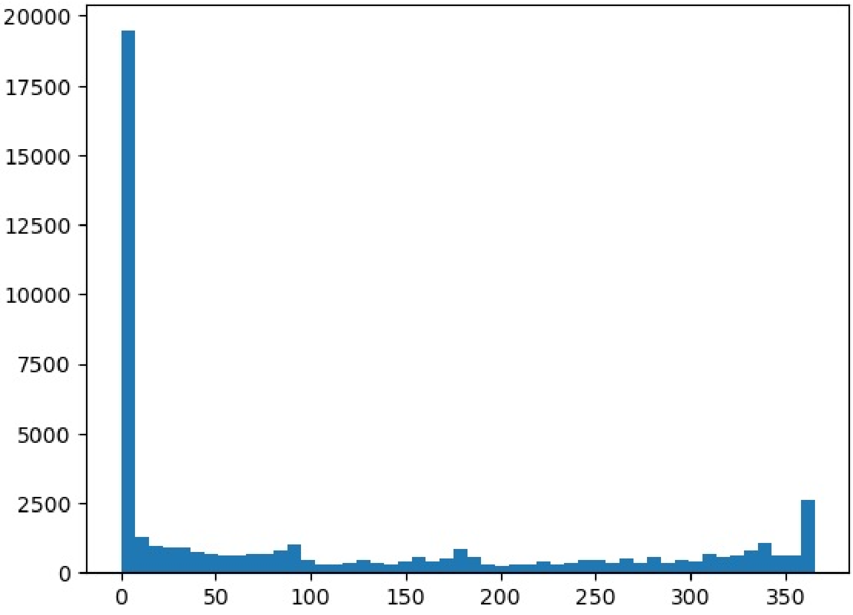
Out[40]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minin
24480	19699218	Clean, Big, Sunny Room in Little Italy/Chinatown!	138784297	Andre	Manhattan	Chinatown	40.71353	-73.99632	Private room	115	
28799	22238046	Bronx 167th Grand ConCourse	162457374	Daniel	Bronx	Concourse Village	40.83314	-73.91708	Private room	38	
14159	10886628	Large Bedroom Available in 5BR Apt	56412357	Scott	Brooklyn	Greenpoint	40.72527	-73.94803	Private room	34	
14158	10886532	2000sf Williamsburg Apt. w/ Theater	17646340	Donald	Brooklyn	Williamsburg	40.70094	-73.94350	Entire home/apt	120	
14157	10886372	BK Bedroom in a Comfortable Apartment by the P...	56410306	Cole	Brooklyn	Prospect-Lefferts Gardens	40.66070	-73.96168	Private room	60	
...	...	...	...	...	...	...	...	...	...	...	
1894	840594	Huge beautiful one bed West Village	4389865	Fiona	Manhattan	West Village	40.73126	-74.00502	Entire home/apt	400	
42648	33110021	Big private room	211906172	Sercan	Queens	Rego Park	40.72407	-73.86585	Private room	70	
15659	12648471	Spacious 3 Bedroom in Park Slope	52577563	Rosa	Brooklyn	Sunset Park	40.66455	-73.99205	Entire home/apt	135	
1707	773497	Great spot in Brooklyn	4081688	Santiago	Brooklyn	Bedford-Stuyvesant	40.69407	-73.94551	Shared room	200	
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	

48526 rows × 16 columns

In [41]:
```
# Create a histogram to visualize the distribution of listings' availability throughout the year
plt.hist(df.availability_365, bins=50)
```

Out[41]:
```
(array([19457.,  1269.,   959.,   906.,   894.,   714.,   696.,   615.,
          612.,   657.,   685.,   804.,  1018.,   430.,   282.,   285.,
          331.,   438.,   328.,   317.,   386.,   565.,   400.,   528.,
          822.,   552.,   303.,   257.,   268.,   290.,   386.,   288.,
          327.,   432.,   475.,   352.,   490.,   350.,   549.,   329.,
          447.,   413.,   703.,   584.,   629.,   776.,  1042.,   625.,
          637.,  2624.]),
 array([  0. ,   7.3,  14.6,  21.9,  29.2,  36.5,  43.8,  51.1,  58.4,
         65.7,  73. ,  80.3,  87.6,  94.9, 102.2, 109.5, 116.8, 124.1,
        131.4, 138.7, 146. , 153.3, 160.6, 167.9, 175.2, 182.5, 189.8,
        197.1, 204.4, 211.7, 219. , 226.3, 233.6, 240.9, 248.2, 255.5,
        262.8, 270.1, 277.4, 284.7, 292. , 299.3, 306.6, 313.9, 321.2,
        328.5, 335.8, 343.1, 350.4, 357.7, 365. ]),
 <BarContainer object of 50 artists>)
```



In [42]:
```
len(df)
```

Out[42]:

```
48526
```

1.which neighbourhood_group is the biggest one?

In [43]:

```
a=df.groupby(by=['neighbourhood_group']).neighbourhood_group.count()
a=a.sort_values(ascending=False)
print(a)
```

```
neighbourhood_group
Manhattan        21427
Brooklyn         20011
Queens            5630
Bronx             1088
Staten Island      370
Name: neighbourhood_group, dtype: int64
```

2.which neighbourhood_group is the most expensive?

In [44]:

```
a=df.groupby(by=['neighbourhood_group']).price.mean()
a=a.sort_values(ascending=False)
print(a)
```

```
neighbourhood_group
Manhattan        179.038036
Brooklyn         117.773625
Staten Island     98.581081
Queens            95.141208
Bronx             85.325368
Name: price, dtype: float64
```

3.which neighbourhood_group has the most possibility to available in year?

In [45]:

```
a=df.groupby(by=['neighbourhood_group']).availability_365.sum()
a=a.sort_values(ascending=False)
print(a)
```

```
neighbourhood_group
Manhattan        2382233
Brooklyn         1998566
Queens            810714
Bronx             180275
Staten Island      73771
Name: availability_365, dtype: int64
```

3.which neighbourhood_group has the most possibility to available in year?

In [46]:

```
a=df.groupby(by=['neighbourhood']).availability_365.sum()
a=a.sort_values(ascending=False)
print(a)
```

```
neighbourhood
Bedford-Stuyvesant            430899
Williamsburg                  290582
Harlem                        279836
Hell's Kitchen                269681
Midtown                       237567
                               ...
Sea Gate                         199
Rossville                         59
Bay Terrace, Staten Island         0
New Dorp                           0
Woodrow                            0
Name: availability_365, Length: 221, dtype: int64
```

4.which neughbourhood_group has the best hosts to stay for a few nights

In [47]:

```
a=df.groupby(by=['neighbourhood_group']).minimum_nights.mean()
a=a.sort_values(ascending=False)
print(a)
```

```
neighbourhood_group
Manhattan        7.796938
Brooklyn         5.575883
Queens           4.846714
Bronx            4.232537
Staten Island    3.843243
Name: minimum_nights, dtype: float64
```

5.which host_name is the most popular hosts between customers?

In [48]:

```
a=df.groupby(by=['host_name']).calculated_host_listings_count.max()
a=a.sort_values(ascending=False)
print(a)
```

```
host_name
Sonder (NYC)      327
Blueground        232
Kara              121
Kazuya            103
Sonder             96
                 ...
Islandgetawayz      1
Iso                 1
Isobel              1
Isoke               1
진                   1
Name: calculated_host_listings_count, Length: 11396, dtype: int64
```

data = pd.read_csv(r"C:\Users\HP\Downloads\AB_NYC_2019.csv")

data

In [51]:
```
data.head(10)
```

Out[51]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_night
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	1
5	5099	Large Cozy 1 BR Apartment In Midtown East	7322	Chris	Manhattan	Murray Hill	40.74767	-73.97500	Entire home/apt	200	
6	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40.68688	-73.95596	Private room	60	4
7	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Hell's Kitchen	40.76489	-73.98493	Private room	79	
8	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Upper West Side	40.80178	-73.96723	Private room	79	
9	5238	Cute & Cozy Lower East Side 1 bdrm	7549	Ben	Manhattan	Chinatown	40.71344	-73.99037	Entire home/apt	150	

In [52]:
```
data.tail()
```

Out[52]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimur
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	
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	

In [53]:

```
data.info()
```

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

In [54]:

```
data.describe()
```

Out[54]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calc
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	

In [55]:

```
data.isna().sum()
```

Out[55]:

```
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
```

what are the top 10 host iDS with the highest number of bookings?

In [56]:

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

Out[56]:

```
host_id
219517861    327
107434423    232
30283594     119
137358866    103
16098958      96
12243051      96
61391963      91
22541573      87
200380610     65
7503643       52
Name: count, dtype: int64
```

q1.what are the top 10 host iDs with the highest number of bookings?

In [61]:

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

Out[61]:

```
host_id
219517861    327
107434423    232
30283594     119
137358866    103
16098958      96
12243051      96
61391963      91
22541573      87
200380610     65
7503643       52
Name: count, dtype: int64
```

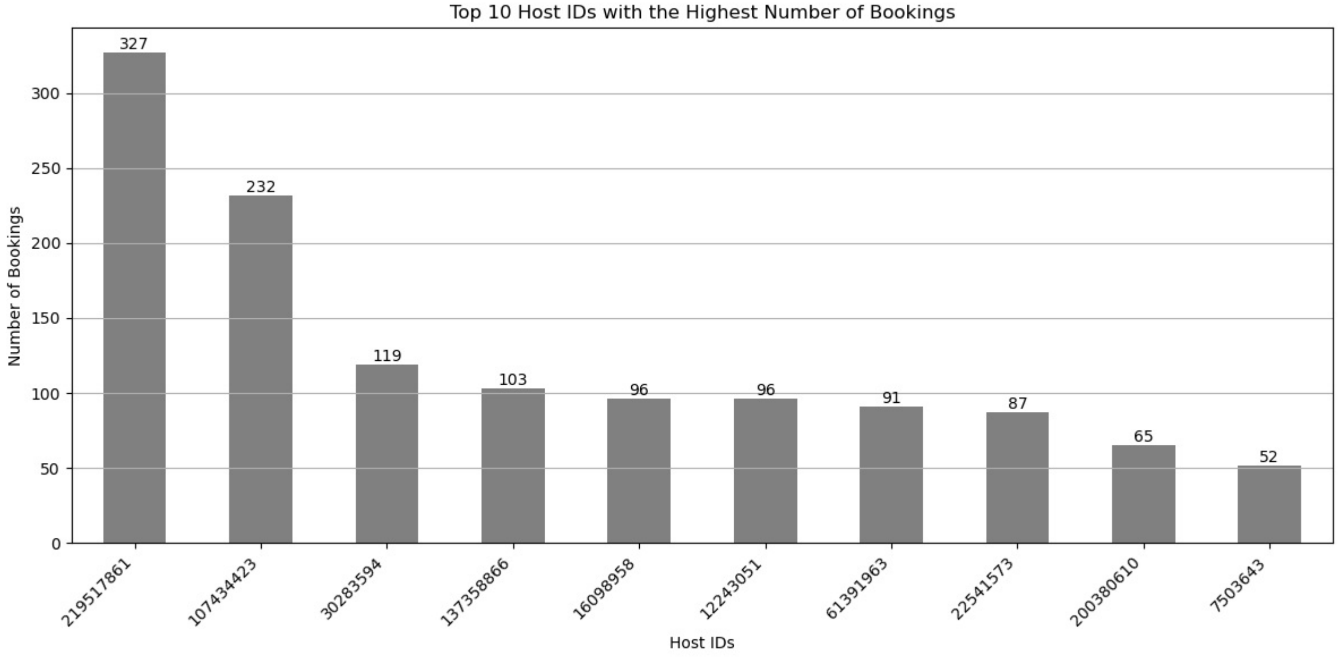
In [59]:

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\hp\anaconda3\lib\site-packages (3.7.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (1
.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (0.11.
0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (
4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (
1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (1.24.3
)
Requirement already satisfied: packaging>=20.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (23
.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib) (9.4.
0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\hp\anaconda3\lib\site-packages (from matplotli
b) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hp\anaconda3\lib\site-packages (from matplotlib
) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-packages (from python-dateutil>=2.7->
matplotlib) (1.16.0)
```

In [60]:

```
 # Visualizing top 10 host IDs with the highest number of bookings
top_10_host_IDs = df['host_id'].value_counts().iloc[:10]
# Plotting
plt.figure(figsize=(12, 6))
ax = top_10_host_IDs .plot(kind='bar', color='grey')
for bars in ax.containers:
    ax.bar_label(bars)
plt.title('Top 10 Host IDs with the Highest Number of Bookings')
plt.xlabel('Host IDs')
plt.ylabel('Number of Bookings')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [62]:

```
# Percentage of bookings for Top 10 Host ID's
hostidPer = (df['host_id'].value_counts().iloc[:10].sort_values(ascending=False)/len(df))*100
hostidPer
```

Out[62]:

```
host_id
219517861    0.673866
107434423    0.478094
30283594     0.245229
137358866    0.212257
16098958     0.197832
12243051     0.197832
61391963     0.187528
22541573     0.179285
200380610    0.133949
7503643      0.107159
Name: count, dtype: float64
```

Observation¶ The host named Michael has 417 bookings attributed to him, accounting for 85% of the total bookings. The person with the Name David stands at the second position with the total bookings of 403.

In [63]:

```
df.head()
```

Out[63]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_night
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	1

Question 5: Which Neighbourhood group has the highest number of bookings?

In [64]:

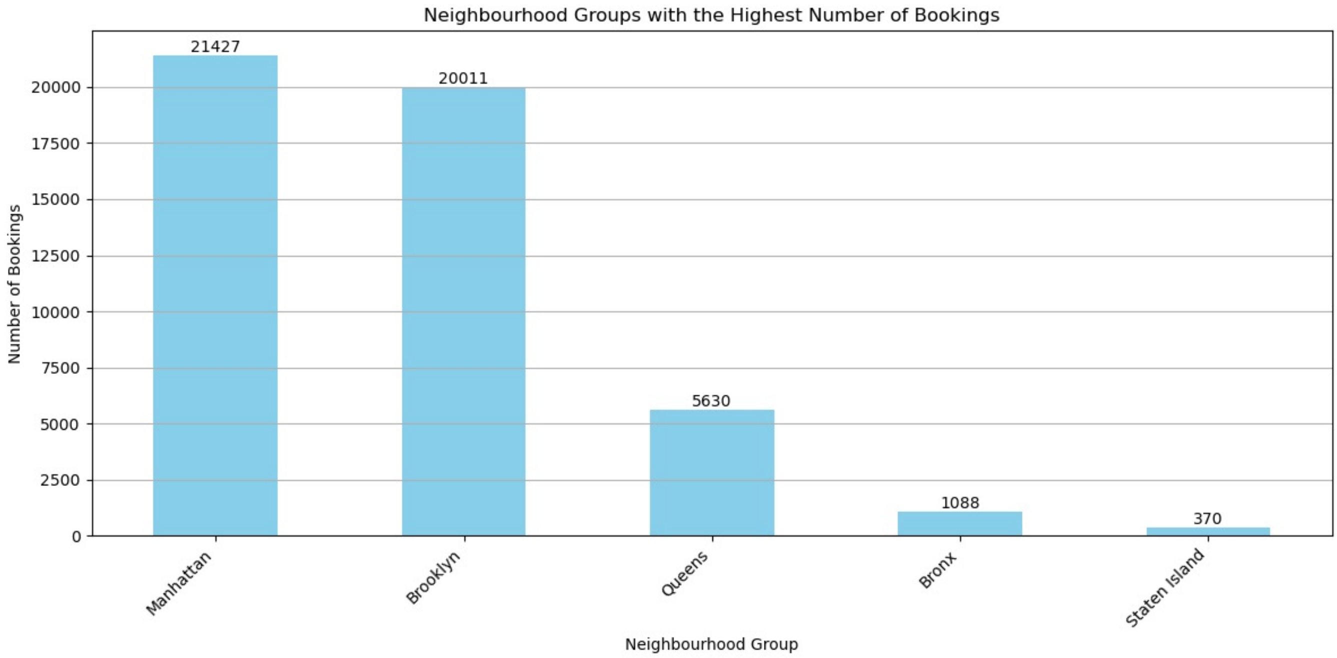
```
# Getting value counts
df['neighbourhood_group'].value_counts()
```

Out[64]:

```
neighbourhood_group
Manhattan        21427
Brooklyn         20011
Queens            5630
Bronx             1088
Staten Island      370
Name: count, dtype: int64
```

In [65]:

```
# Visualizing neighbourhood groups with the highest number of bookings
neightop = df['neighbourhood_group'].value_counts()
# Plotting
plt.figure(figsize=(12, 6))
ax = neightop.plot(kind='bar', color='skyblue')
for bars in ax.containers:
    ax.bar_label(bars)
plt.title('Neighbourhood Groups with the Highest Number of Bookings')
plt.xlabel('Neighbourhood Group')
plt.ylabel('Number of Bookings')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [66]:

```
# Percentage of bookings for Neighbourhood groups
neighbourhood_grpPer = (df['neighbourhood_group'].value_counts().sort_values(ascending=False)/len(df))*100
neighbourhood_grpPer
```

Out[66]:

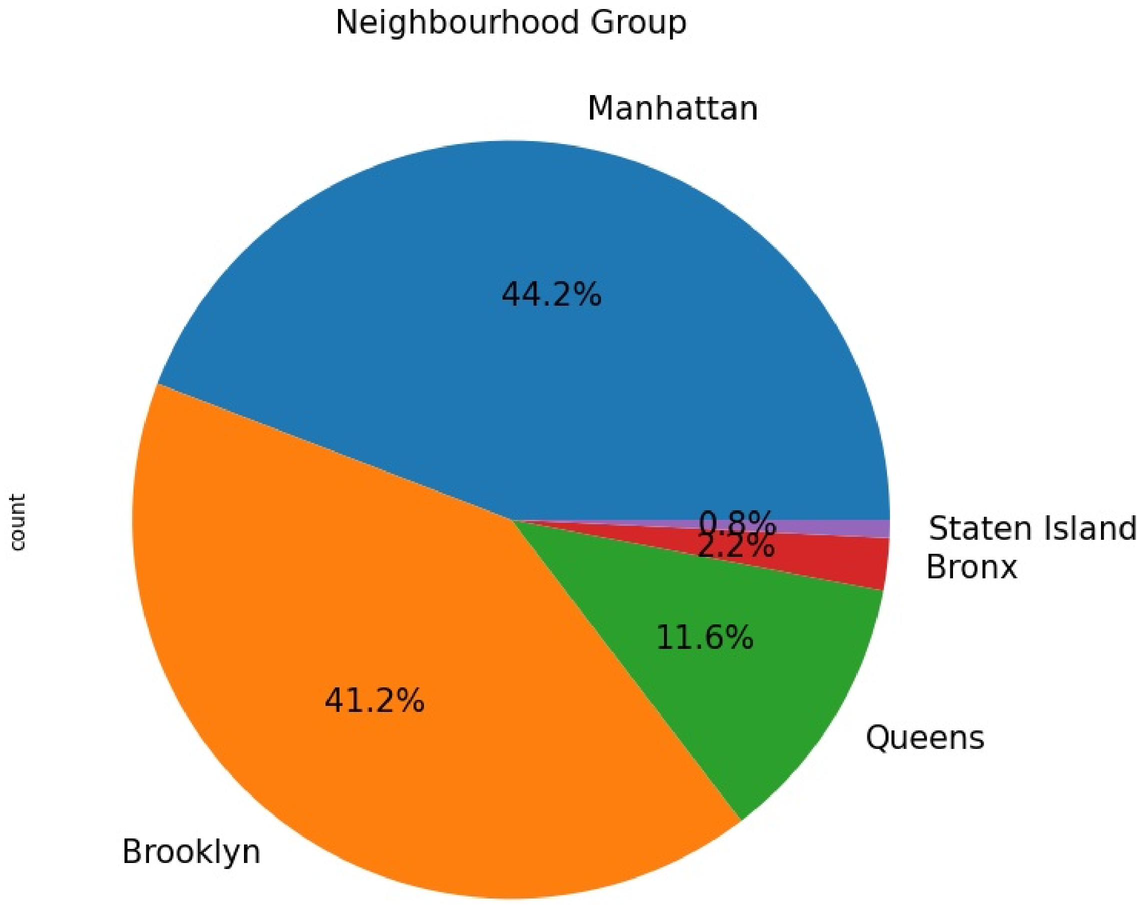
```
neighbourhood_group
Manhattan        44.155710
Brooklyn         41.237687
Queens           11.602028
Bronx             2.242097
Staten Island     0.762478
Name: count, dtype: float64
```

In [67]:

```
# Visualizing using pie chart
df['neighbourhood_group'].value_counts().plot(kind = 'pie', figsize = (8,8), fontsize = 15, autopct = '%1.1f%%'
plt.title("Neighbourhood Group", fontsize = 15)
```

Out[67]:

```
Text(0.5, 1.0, 'Neighbourhood Group')
```



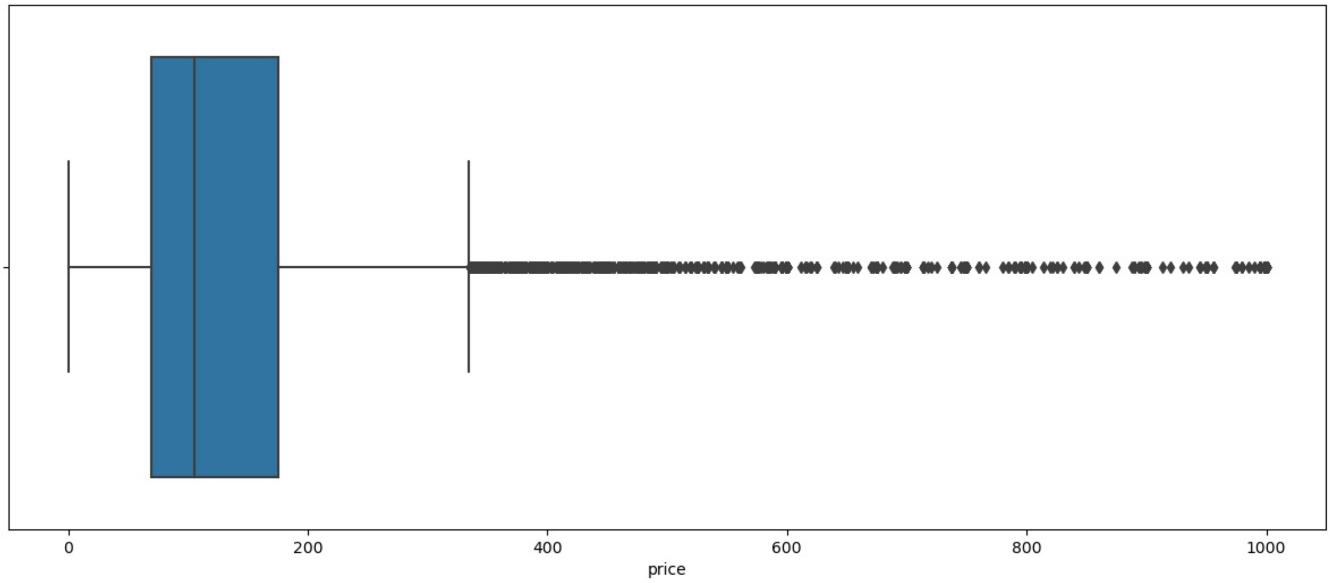
observations

- An observation reveals that among all the neighborhood groups, the Manhattan group has the highest number of bookings, totaling 21,661, which constitutes 44.3% of all bookings across all groups.
- Brooklyn ranks as the second-highest neighborhood group with a total of 20,104 bookings, covering 41% of all bookings.
- Staten Island is the neighbourhood group with the least number of bookings which constitutes only 0.76% of all the bookings

Question 6: Which Neighbourhood Group has the maximum price range for rooms?

In [68]:

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



In [69]:

```
# Generate descriptive statistics for the 'price' column, including count, mean, standard deviation, min, max,
df['price'].describe()
```

Out[69]:

```
count    48526.000000
mean       141.325681
std        116.791978
min          0.000000
25%         69.000000
50%        105.000000
75%        175.000000
max       1000.000000
Name: price, dtype: float64
```

In [70]:

```
# Filter the DataFrame to include only listings with a price less than 334 and store the result in a new DataFr
df_new = df[df['price'] < 334 ]
df_new.head()
```

Out[70]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_night
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	1

In [71]:

```
# Group the DataFrame by 'neighbourhood_group' and generate descriptive statistics for the 'price' column
# Transpose the result, reset the index, and store it in a new DataFrame
df.groupby(['neighbourhood_group'])['price'].describe().T.reset_index()
```

Out[71]:

neighbourhood_group	index	Bronx	Brooklyn	Manhattan	Queens	Staten Island
0	count	1088.000000	20011.000000	21427.000000	5630.000000	370.000000
1	mean	85.325368	117.773625	179.038036	95.141208	98.581081
2	std	77.831942	94.411744	133.962626	74.630484	96.268905
3	min	0.000000	0.000000	0.000000	10.000000	13.000000
4	25%	45.000000	60.000000	95.000000	50.000000	50.000000
5	50%	65.000000	90.000000	149.000000	75.000000	75.000000
6	75%	99.000000	150.000000	220.000000	110.000000	109.000000
7	max	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000

Observation

The price range for Bronx Neighbourhood group is in the range 0 and 2500

The price range for Brooklyn Neighbourhood group is in the range 0 and 10000

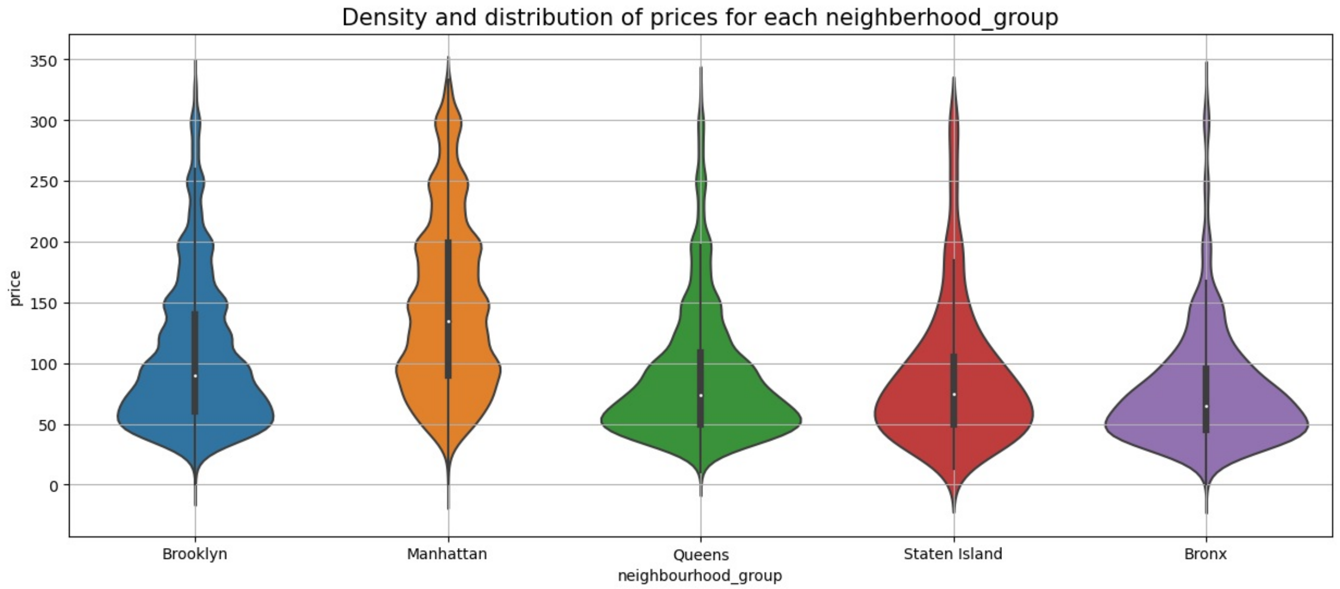
The price range for Manhattan Neighbourhood group is in the range 0 and 10000

The price range for Queens Neighbourhood group is in the range 10 and 10000

The price range for Staten Island Neighbourhood group is in the range 13 and 5000

In [72]:

```
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 = 15)
plt.grid()
```

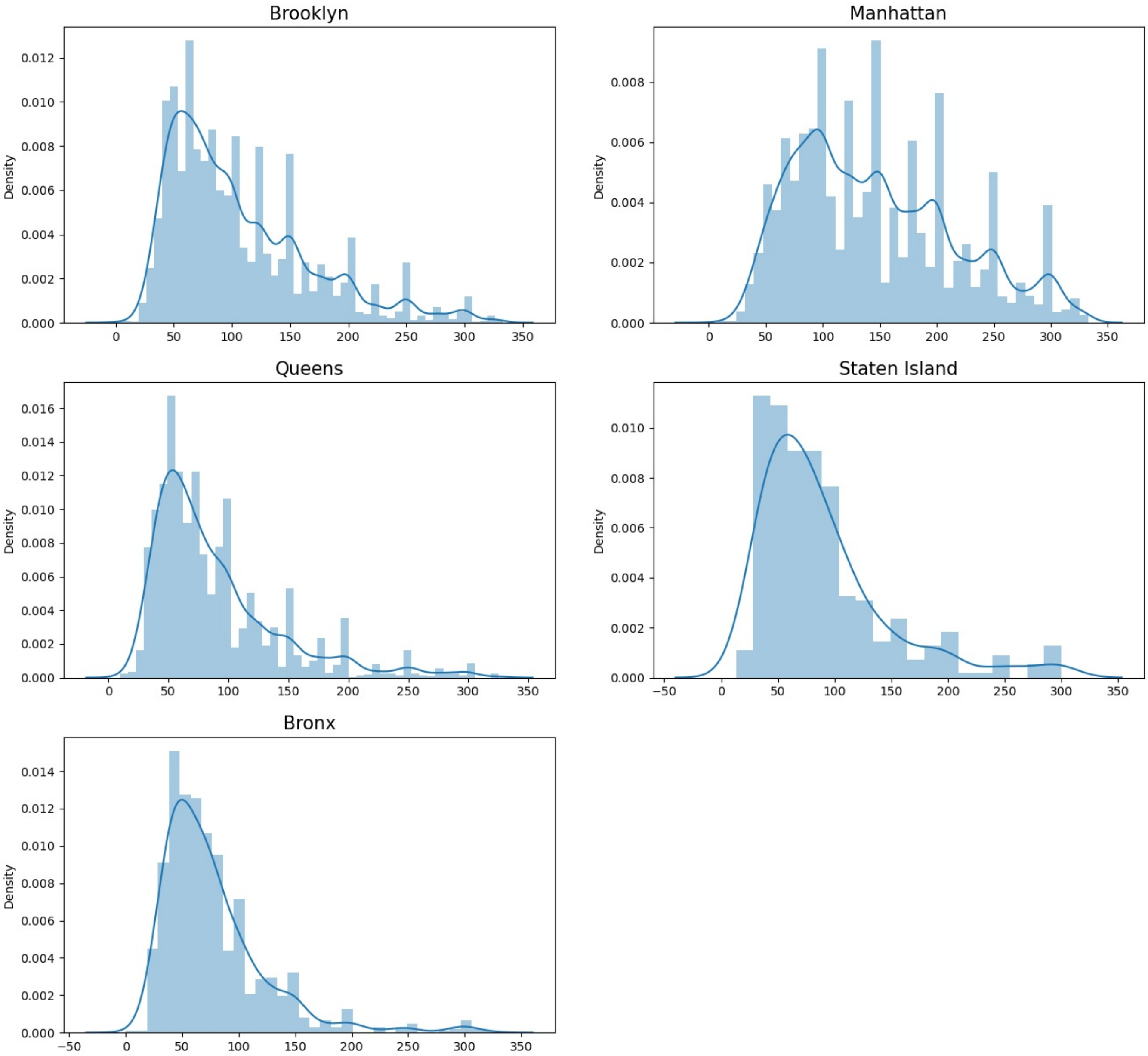


In [73]:

```
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)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_26408\1285570170.py:5: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x = n1['price'])
C:\Users\HP\AppData\Local\Temp\ipykernel_26408\1285570170.py:10: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x = n2['price'])
C:\Users\HP\AppData\Local\Temp\ipykernel_26408\1285570170.py:15: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x = n3['price'])
C:\Users\HP\AppData\Local\Temp\ipykernel_26408\1285570170.py:20: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x = n4['price'])
C:\Users\HP\AppData\Local\Temp\ipykernel_26408\1285570170.py:25: UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x = n5['price'])
```

```
Out[73]: Text(0.5, 1.0, 'Bronx')
```



Observation

we can observe that Manhattan has the highest range of prices for the listings with 150 price as median observation, followed by Brooklyn with 90 per night.

Queens and Staten Island appear to have very similar distributions, Bronx is the cheapest of them all.

Question 7: What are the Top 10 Neighbourhoods having highest number of bookings?¶

In [74]:

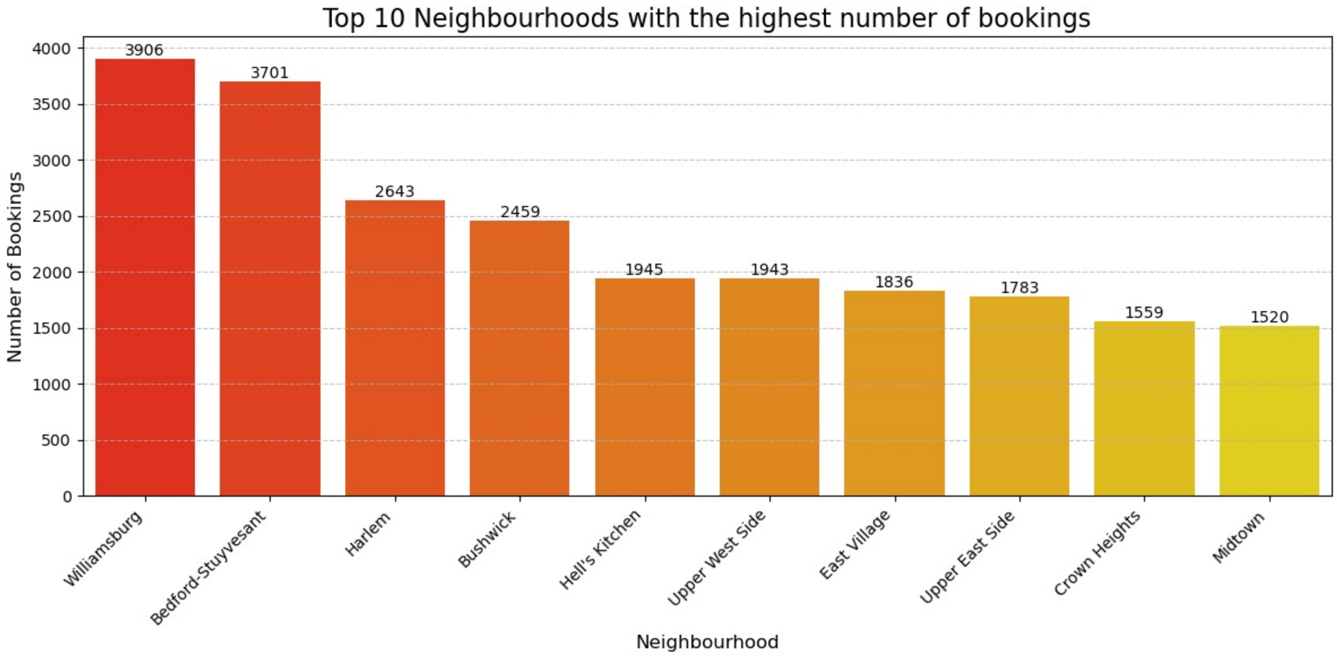
```
df['neighbourhood'].value_counts().iloc[:10]
```

Out[74]:

```
neighbourhood
Williamsburg          3906
Bedford-Stuyvesant    3701
Harlem                2643
Bushwick              2459
Hell's Kitchen        1945
Upper West Side       1943
East Village          1836
Upper East Side       1783
Crown Heights         1559
Midtown               1520
Name: count, dtype: int64
```

In [75]:

```
# Visualizing the Top 10 Neighbourhoods with the highest number of bookings
plt.figure(figsize=(12, 6))
ax = sns.barplot(x=df['neighbourhood'].value_counts().iloc[:10].keys(), y=df['neighbourhood'].value_counts().il
for bars in ax.containers:
    ax.bar_label(bars)
plt.title("Top 10 Neighbourhoods with the highest number of bookings", fontsize=16)
plt.xlabel("Neighbourhood", fontsize=12)
plt.ylabel("Number of Bookings", fontsize=12)
plt.xticks(rotation=45, ha="right", fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Question 8: Which room type has highest number of bookings?¶

In [76]:

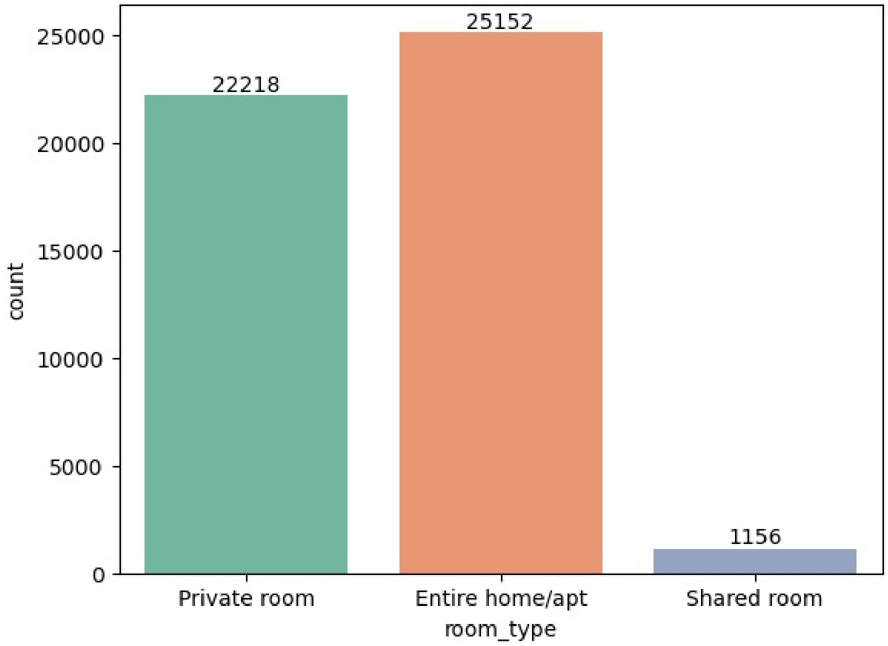
```
# Getting the value counts
df['room_type'].value_counts()
```

Out[76]:

```
room_type
Entire home/apt    25152
Private room       22218
Shared room         1156
Name: count, dtype: int64
```

In [77]:

```
# Visualizing using Count Plot
ax = sns.countplot(x = 'room_type',data = df, palette="Set2")

for bars in ax.containers:
    ax.bar_label(bars)
```



conclusion: Throught this analysis,we have a better idea on the key factors that influences the demand of an airbnb listing property.Tourists/customers prefer location close to downtown, lower price and entire room which offers them more privacy when toring the city.These can all be taken into consideration for airbnb hosts when posting their properties online.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js