AirBnb NYC EDA - Exploratory Data Analysis

PROJECT SUMMARY

Purpose of the Project :- The purpose of this project is to conduct **comprehensive data analysis** on Airbnb with the aim of extracting meaningful insights and identifying trends or patterns.By performing end-to-end analysis, including some advanced analytics techniques such as IQR techniques, exploratory data analysis (EDA).Wtih the help of this project, we will solve various business problem statements and enhance business strategies & drive overall business growth in the Airbnb ecosystem.

Tools & Libraries :- Pandas,Numpy,matplotlib & Seaborn.

Content -

Overview of AirBnb - An introduction to the Airbnb platform, and its business model.

Description of Dataset - A detailed explanation of the variables in a dataset.

QnA - Some question and Answer related to Airbnb to getting comfortable with data.

Business Problem Statement - Some problem statement related to Airbnb to find data driven insights.

Steps of EDA - Perform some steps of Exploratory data analysis such as data cleaning, data mining.

Explanation of QnA - Detailed explanation of QnA section.

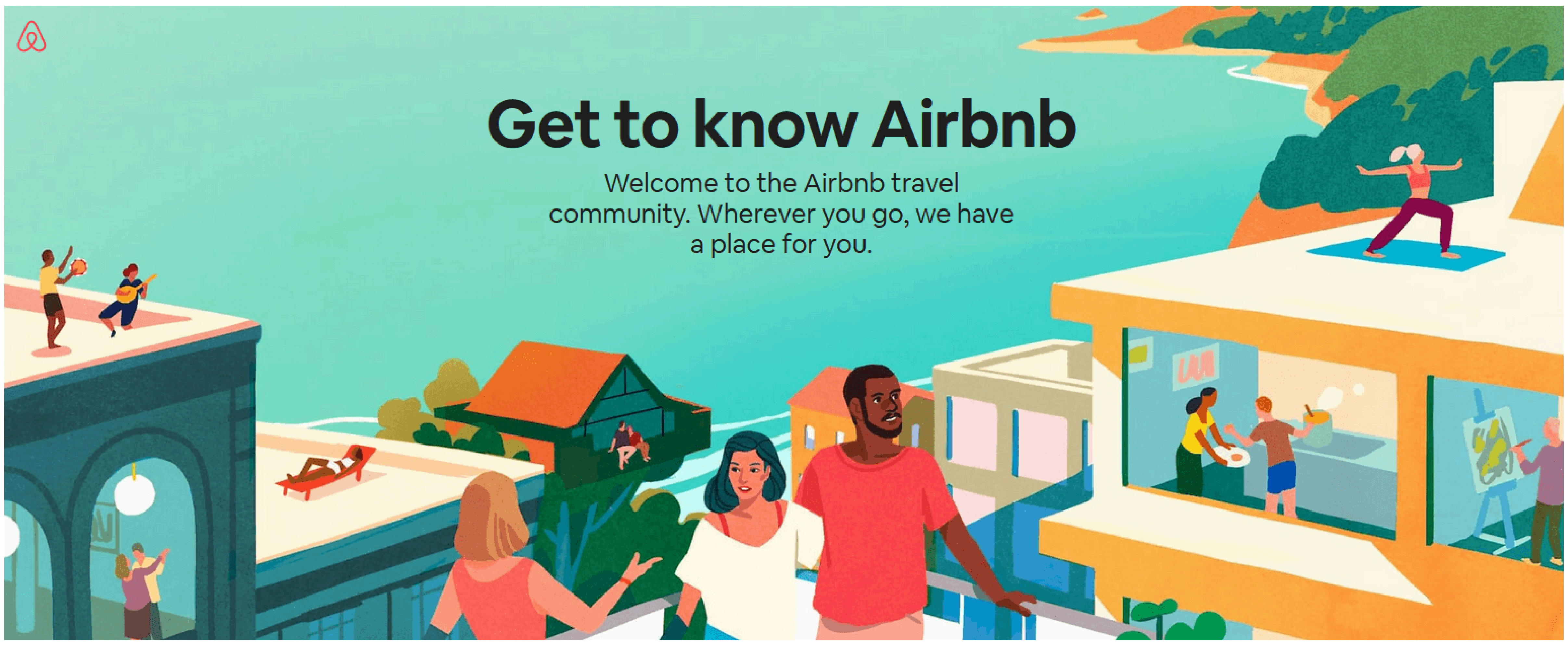
Analysis Problem Statement - Analyze the business problem statement to identify trend or pattern.

Business Conclusion - Discussion about the insights or patterns which help in making future - decisions.

OVERVIEW OF AirBnb

In [5]:
```
from IPython.display import Image
Image(filename='D:\download.png', width=1000)
```

Out[5]:

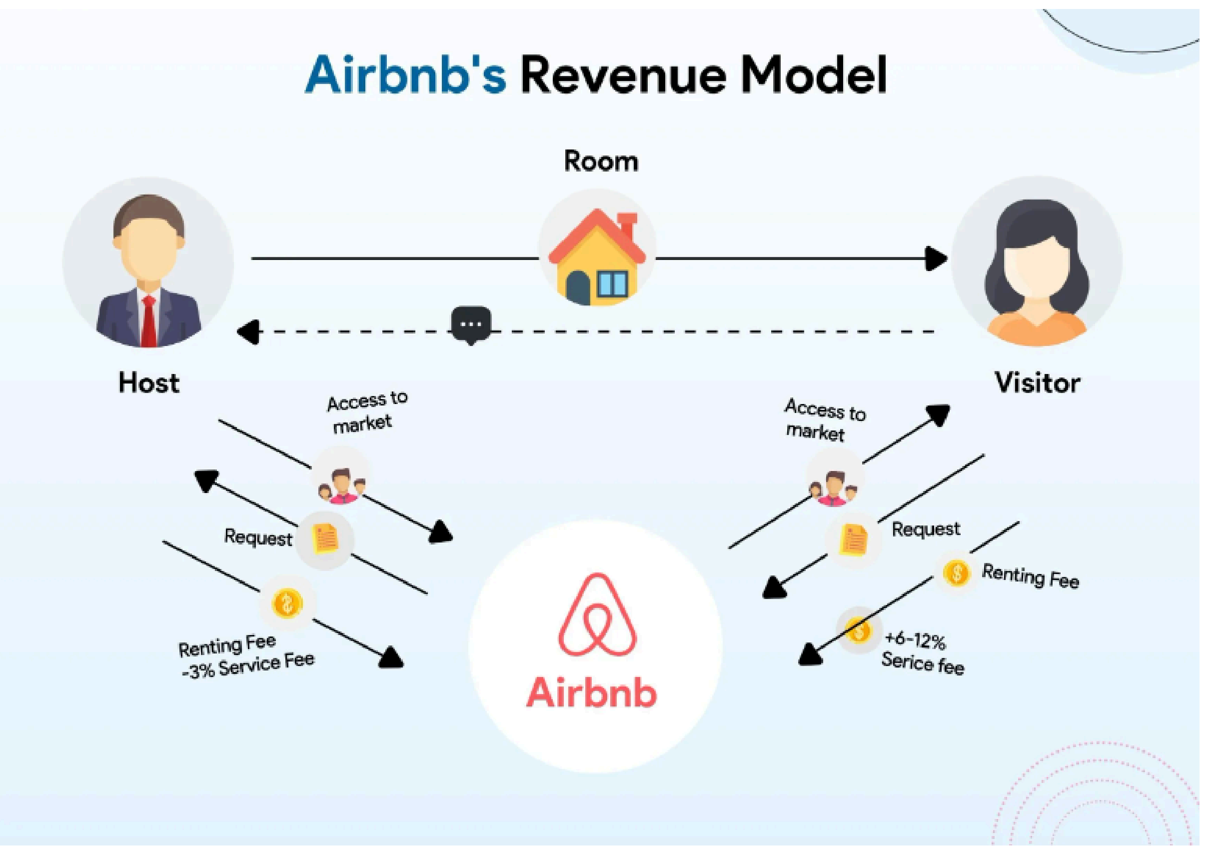


Airbnb, Inc. is an American company operating an online marketplace for short- and long-term homestays and experiences. The company acts as a broker and charges a commission from each booking. **The company was founded in 2008 by Brian Chesky, Nathan Blecharczyk, and Joe Gebbia.** The idea originated when Chesky and Gebbia, struggling to pay rent, rented out air mattresses in their San Francisco apartment, which evolved into **"Air Bed & Breakfast."** Airbnb has built a vast global community, operating in over 220 countries with millions of listings. The company went public in December 2020, marking a significant milestone. Today, under CEO Brian Chesky's leadership, Airbnb continues to innovate, adapting to changing travel trends and fostering cultural exchange through personalized travel experiences.

Let's understand the Business Model

In [24]:
```
from IPython.display import Image
Image(filename="D:\Business Model Picture.png", width=500)
```

Out[24]:



Airbnb's revenue model primarily revolves a **two-sided marketplace** around charging service fees to both hosts and guests for facilitating bookings through its platform. Airbnb generates revenue through service fees charged to both hosts and guests for each booking made on its platform. For guests, the service fee typically ranges from **6% to 12%** of the booking subtotal, while hosts are charged around **3%** of the subtotal. This fee structure allows Airbnb to earn a commission on every transaction. In addition to these fees, Airbnb offers various value-added services to hosts, such as professional photography and property management services, which also contribute to its revenue.

DESCRIPTION OF VARIABLES IN DATASET

Column Name	Description
Id	Unique identifier for each listing in the dataset.
Name	Name or title of the listing, as it appears on the Airbnb website.
Host_id	Unique identifier for each host in the dataset.
Host_name	Name of the host as it appears on the Airbnb website.
Neighbourhood_group	Grouping of neighborhoods in New York City, such as Manhattan or Brooklyn.
Neighbourhood	Specific neighborhood in which the listing is located.
Latitude	Geographic latitude of the listing.
Longitude	Geographic longitude of the listing.
Room_type	Type of room or property being offered.
Price	Nightly price for the listing, in US dollars.
Minimum_nights	Minimum number of nights that a guest must stay at the listing.
Total_reviews	Total number of reviews that the listing has received.
Reviews_per_month	Average number of reviews that the listing receives per month.
Host_listings_count	Total number of listings that the host has on Airbnb.
Availability_365	The number of days that the listing is available for booking.

QnA

1.Which City has the Highest No. of Listing Property ?
2.Which Area have the Highest Reviews (Across all cities) ?
3.Whose host have the Highest no. of Listing Property ?
4.How many Host are in Manhattan ?
5.Which City have Lowest Avg. Price ?
6.How many Private rooms in NYC ?
7.Which Room Type have the most reviewed(%) in NYC ?
8.How many Areas in Queens have price between 100 USD to 150 USD ?

➤ Manhattan.
➤ Bedford-Stuyvesant.
➤ Michael.
➤ 15,080 Hosts.
➤ Bronx City.
➤ 21,996 Rooms.
➤ Private Rooms (49.88%)
➤ 11 Areas.

BUSINESS PROBLEM STATEMENT

1. Analyze the distribution of prices across the dataset to identify common price ranges. **Click here**

2. Find out Top 10 Neighbourhood based on listing properties to identify the neighborhoods with the highest number of listed properties to understand the popularity and demand in different areas. **Click here**

3. Find out Top 10 host based on listing properties to determine the hosts who have the highest number of listed properties to recognize key contributors to the platform. **Click here**

4. Find the best location for travelers to identify the ideal locations for travelers based on factors such as average price and user reviews. **Click here**

5. Analyze the price trends of different room types (e.g., entire home/apt, private room, shared room) across various cities. **Click here**

6. Identify any unique characteristics or preferences in each city that influence the distribution of room types. **Click here**

7. Determine the city with the highest average price and investigate the reason behind to its pricing. **Click here**

8. Examine the relationship between availability of listings and different cities. **Click here**

9. What is the relationship of diversity of price in each city. **Click here**

10. Investigate the relationship between user reviews and room types across different cities. **Click here**

4 STEPS OF EDA

In [89]:
```
#import necessary libaries

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

STEP 1: Data Loading

In [90]:
```
#Loading the dataset

df = pd.read_csv("D:\Airbnb (Pandas).csv")
df
```

Out[90]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	r
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-10-2018	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-05-2019	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-07-2019	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-11-2018	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	NaN	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	NaN	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	NaN	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	NaN	
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	NaN	

48895 rows × 16 columns

STEP 2: Data Cleaning

a) Identify Duplicates Row !!

In [91]:
```
#find how many rows have duplicated

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

Out[91]:
```
0
```

b) Handle Missing Values !!

In [92]:
```
# find how many columns are null.

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

Out[92]:
```
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 [93]:
```
#find the mode of the host_name column

mode_result = df['host_name'].mode()
mode_result
```

Out[93]:
```
0    Michael
Name: host_name, dtype: object
```

- Replace NaN in host_name Column with Mode of that column.

In [94]:
```
#fill null value with Michael in host_name column

df['host_name'].fillna(value='Michael',inplace=True)
df
```

Out[94]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	r
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-10-2018	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-05-2019	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-07-2019	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-11-2018	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	NaN	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	NaN	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	NaN	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	NaN	
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	NaN	

48895 rows × 16 columns

In [95]:

```python
#delete the unnessary columns

df.drop(columns='name',inplace=True)
df
```

Out[95]:

	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_mont
0	2539	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-10-2018	0.2
1	2595	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-05-2019	0.3
2	3647	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	Nal
3	3831	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-07-2019	4.6
4	5022	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-11-2018	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	
48890	36484665	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	NaN	Nal
48891	36485057	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	NaN	Nal
48892	36485431	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	NaN	Nal
48893	36485609	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	NaN	Nal
48894	36487245	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	NaN	Nal

48895 rows × 15 columns

STEP 3: Manipulating Data

In [96]:

```python
#fill the last_review column with forward fill

df['last_review'].fillna(method='ffill',inplace=True)
df
```

Out[96]:

	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_mont
0	2539	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-10-2018	0.2
1	2595	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-05-2019	0.3
2	3647	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	21-05-2019	Nal
3	3831	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-07-2019	4.6
4	5022	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-11-2018	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	.
48890	36484665	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	08-07-2019	Nal
48891	36485057	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	08-07-2019	Nal
48892	36485431	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	08-07-2019	Nal
48893	36485609	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	08-07-2019	Nal
48894	36487245	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	08-07-2019	Nal

48895 rows × 15 columns

In [97]:

```
#fill null values of reviews_per_month column with 0

df['reviews_per_month'].fillna(value=0,inplace=True)
df
```

Out[97]:

	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_mont
0	2539	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-10-2018	0.2
1	2595	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-05-2019	0.3
2	3647	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	21-05-2019	0.0
3	3831	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-07-2019	4.6
4	5022	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-11-2018	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	.
48890	36484665	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	08-07-2019	0.0
48891	36485057	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	08-07-2019	0.0
48892	36485431	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	08-07-2019	0.0
48893	36485609	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	08-07-2019	0.0
48894	36487245	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	08-07-2019	0.0

48895 rows × 15 columns

In [98]:

```
#Re-check null value in columns

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

Out[98]:

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

In [99]:

```
#find average price

avg_price = df['price'].mean()
round(avg_price,2)
```

Out[99]:

```
152.72
```

- Replace '0' in Price Column with Avg Price.

In [100…

```
#replace 0 with average price for more accuracy
```

```
df['price'].replace(0,round(avg_price),inplace=True)
df.sample(5)
```

Out[100]:

	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_mont
39344	30694973	137358866	Kazuya	Queens	Woodside	40.74188	-73.90146	Private room	61	30	2	31-05-2019	0.7
817	290457	207124	Mikki & Bazi	Manhattan	Chinatown	40.71283	-73.99703	Entire home/apt	139	30	37	16-02-2019	0.4
18323	14372031	5662183	Brian	Brooklyn	Williamsburg	40.70786	-73.95030	Private room	80	2	166	22-06-2019	4.6
39621	30826777	204006071	呈刚	Queens	Long Island City	40.74751	-73.93744	Private room	45	3	4	07-06-2019	0.6
11644	9059397	14614459	Dario	Manhattan	Upper East Side	40.78370	-73.94877	Entire home/apt	180	1	1	05-01-2016	0.0

STEP 4: Understanding Data

In [101…

```
#understand the structure of dataset and summary statistics.

df.describe()
```

Out[101]:

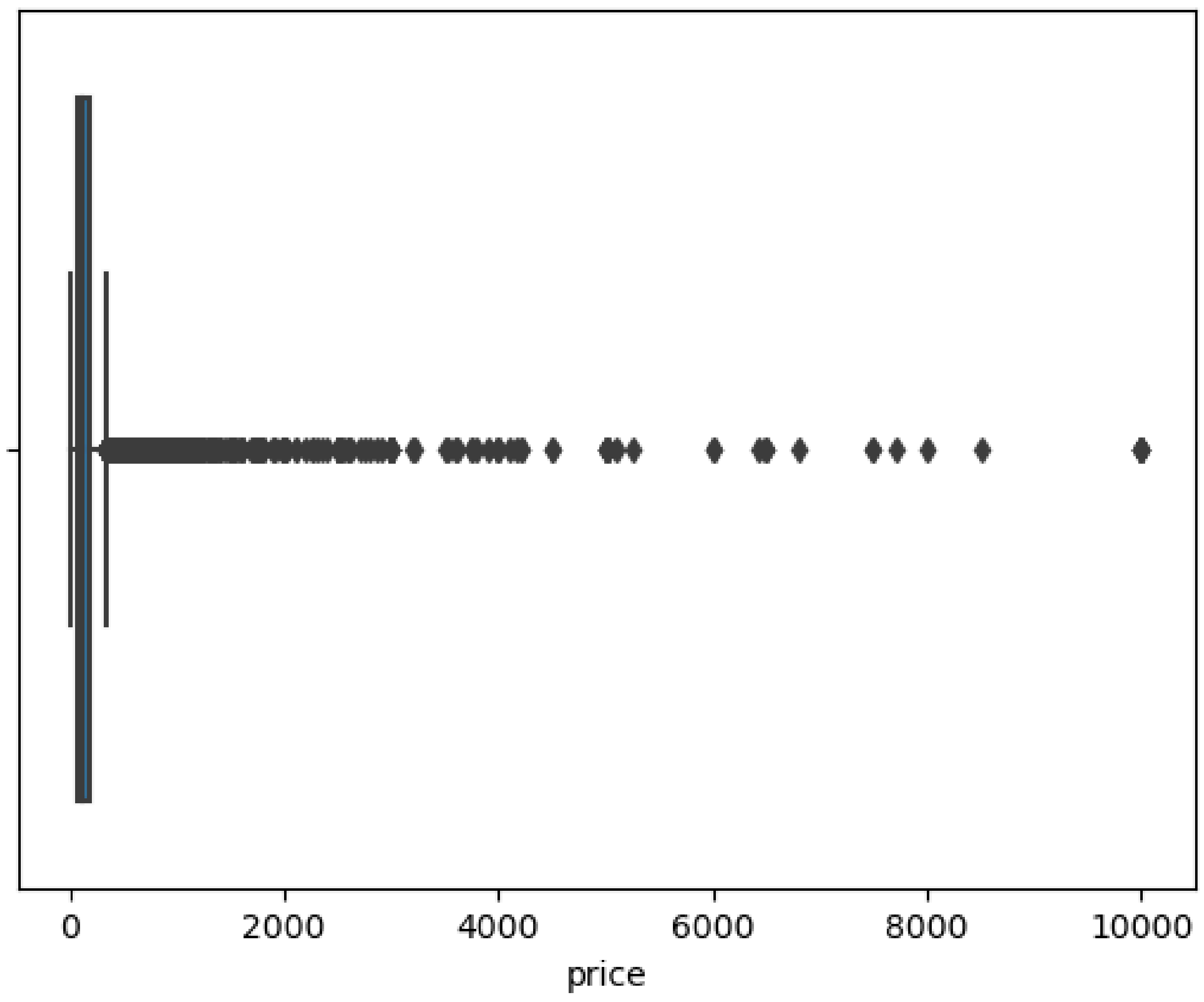
	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.755108	7.029962	23.274466	1.090910	7.143982	112.781327
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.143242	20.510550	44.550582	1.597283	32.952519	131.622289
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	10.000000	1.000000	0.000000	0.000000	1.000000	0.000000
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.040000	1.000000	0.000000
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.370000	1.000000	45.000000
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	1.580000	2.000000	227.000000
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	365.000000

- The Avg. Price of Airbnb listing properties is 152.75.
- The Minimum Price of Properties is Approx 10USD & Maximum 10k USD.
- Total 48,895 Reviews are mentioned across different cities.
- Maximum Reviews is 629 in one of the areas across different Cities.

Let's First check the outliers in dataset !!

In [102…

```
# we check outlier in price columns because we see that price is very important column in this dataset.

sns.boxplot(x = df['price'])
plt.show()
```



- As we see, our dataset contains outliers in Price column.

Remove Outliers (IQR Technique)

In [103…

```
#find 25 & 75 percentile for further use

Q1 = np.percentile(df['price'], 25)
print(Q1)
Q3 = np.percentile(df['price'], 75)
print(Q3)
```

```
69.0
175.0
```

In [104…

```
#find median(50 percentile) by difference of q3 -q1

IQR = Q3 - Q1
IQR
```

Out[104]:

```
106.0
```

In [105…

```
#create a lower & upper limit of price

lower_limit = Q1 - (1.5*IQR)
print(lower_limit)
upper_limit = Q3 + (1.5*IQR)
print(upper_limit)
```

```
-90.0
334.0
```

In [106…

```
#create a new dataframe with lower & upper limit of price

df1 = df[(df['price']>lower_limit) & (df['price']<upper_limit)]
df1.sample(5)
```

Out[106]:

	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month
19479	15543169	11845677	Kim	Brooklyn	Clinton Hill	40.68229	-73.96154	Entire home/apt	160	2	97	05-07-2019	3.0(
10257	7853584	41385305	Khalid	Manhattan	Harlem	40.81899	-73.94694	Entire home/apt	80	2	1	09-03-2016	0.02
39969	31054795	28142165	Souha	Manhattan	Harlem	40.82219	-73.95381	Private room	89	2	10	18-06-2019	2.01
12293	9505047	49255756	Sol	Brooklyn	Williamsburg	40.71848	-73.95817	Private room	80	7	11	04-05-2019	0.25
33703	26710580	22800762	Mary	Manhattan	Hell's Kitchen	40.76484	-73.98969	Entire home/apt	220	4	0	07-07-2019	0.0(

In [107…

```
#check new dataframe shape and price column

print(df1.shape)
print(df1['price'].max())
print(df1['price'].min())
```

```
(45918, 15)
333
10
```

In [170…

```
#reset the index of new dataframe

df1.reset_index(inplace=True)
df1
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_28336\2839659791.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1.drop(columns=['level_0','index'],inplace=True)
```

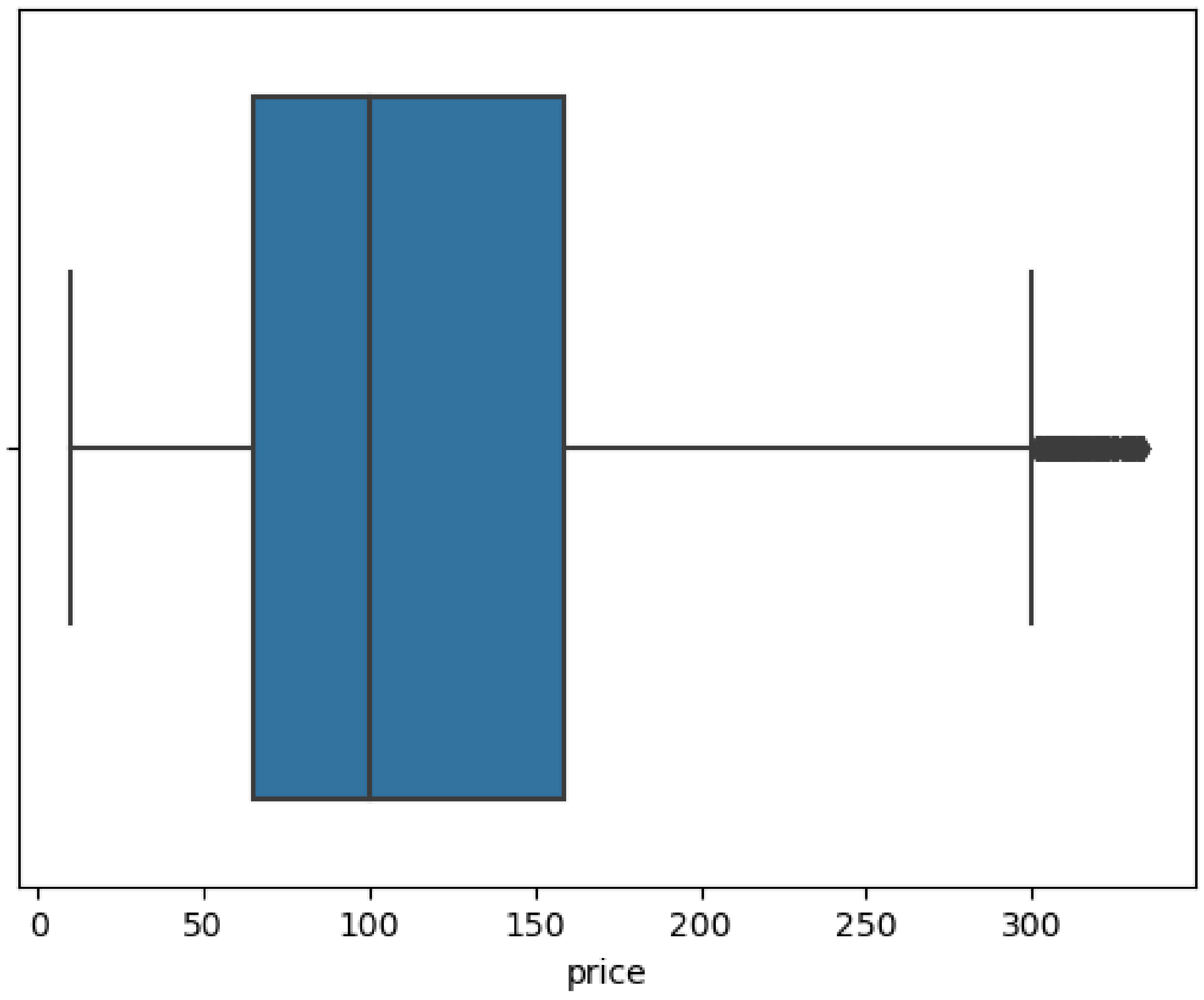
Out[170]:

	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_mont
0	2539	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-10-2018	0.2
1	2595	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-05-2019	0.3
2	3647	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	21-05-2019	0.0
3	3831	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-07-2019	4.6
4	5022	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-11-2018	0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	
45913	36484665	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	08-07-2019	0.0
45914	36485057	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	08-07-2019	0.0
45915	36485431	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	08-07-2019	0.0
45916	36485609	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	08-07-2019	0.0
45917	36487245	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	08-07-2019	0.0

45918 rows × 16 columns

In [109…

```
#check new dataframe have no outliers for use of futher analysis

sns.boxplot(x = df1['price'])
plt.show()
```



- As we see, now no outliers in price columns. We are good to go for analysis.

EXPLAINATION OF Q/A ANSWERS

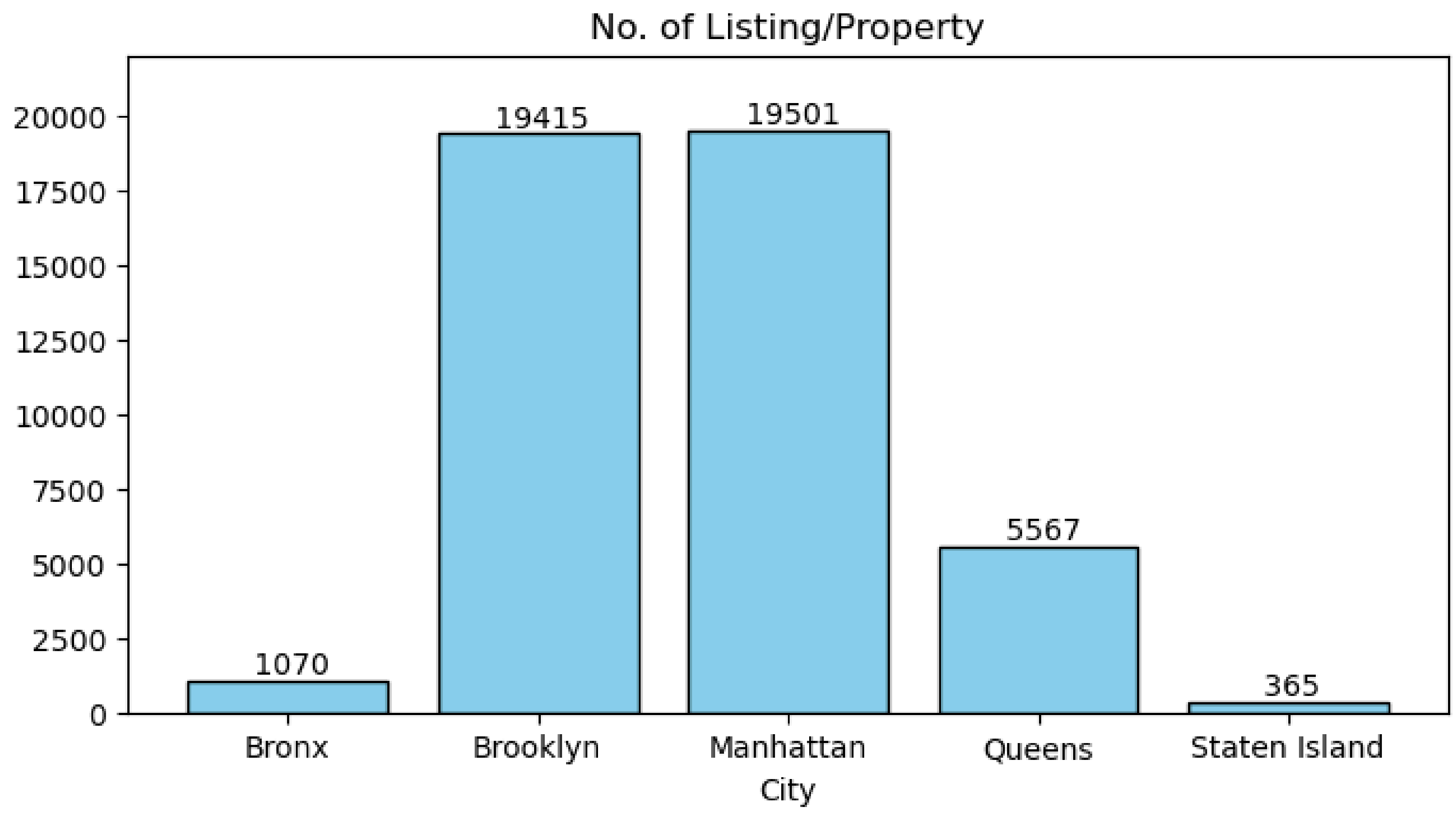
In [110…
```
#find cities wise no.of properties with the help of pivot table

Cities_wise_prop = df1.pivot_table(index='neighbourhood_group',aggfunc='count')
Cities_wise_prop['host_id']
```

Out[110]:
```
neighbourhood_group
Bronx             1070
Brooklyn         19415
Manhattan        19501
Queens            5567
Staten Island      365
Name: host_id, dtype: int64
```

In [111…
```
#store data in variable x and y

x = Cities_wise_prop['host_id'].index
y = Cities_wise_prop['host_id'].values
x
```

Out[111]:
```
Index(['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island'], dtype='object', name='neighbourhood_group')
```

In [112…
```
#adjust the size of graph
plt.figure(figsize=(8,4))

#plot the graph and add data labels
bars = plt.bar(x,y,color='skyblue',edgecolor='black')
plt.bar_label(bars,label_type='edge',fmt=' '+'%.0f')
plt.ylim(0,22000)

#label the title & x -axis of graph
plt.title('No. of Listing/Property')
plt.xlabel('City')

plt.show()
```



Manhattan City has the Highest No. of lising Property.

In [113…
```
#find highest reviewed neighbourhood by sum of reviews of neighbourhood

high_views = df1.pivot_table(index=['neighbourhood'],aggfunc='sum').sort_values(by='number_of_reviews',ascending=False)
pd.DataFrame(high_views['number_of_reviews'])
```

Out[113]:

	number_of_reviews
neighbourhood	
Bedford-Stuyvesant	108773
Williamsburg	82399
Harlem	74770
Bushwick	52112
Hell's Kitchen	47489
...	...
West Farms	7
Breezy Point	5
Sea Gate	4
Bay Terrace, Staten Island	3
New Dorp	0

219 rows × 1 columns

Bedford-Stuyvesant area has the Highest Reviews (Across all areas).

In [114…

```
#find the host which has highest no. of listing properties

highest_host = df1.groupby('host_name')['neighbourhood'].count()
highest_host.sort_values(ascending=False)
```

Out[114]:

```
host_name
Michael            404
David              368
John               276
Sonder (NYC)       272
Alex               253
                  ...
Jennifer & Inam      1
Jennie And Dan       1
Jenni & Eric         1
Jenn And Mike        1
진                    1
Name: neighbourhood, Length: 11008, dtype: int64
```

Michael have the Highest no. of Listing Property.

In [115…

```
#find city wise hosts with the help of groupby func.

no_of_host = df1.groupby(['neighbourhood_group','host_id'])['host_id'].count()
no_of_host
```

Out[115]:

```
neighbourhood_group  host_id
Bronx                12221        2
                     42761        1
                     119445       1
                     120623       1
                     153817       1
                                 ..
Staten Island        258635350    1
                     268430876    1
                     269592097    1
                     271528362    1
                     272557707    1
Name: host_id, Length: 35498, dtype: int64
```

In [116…

```
#find no. of hosts in manhattan by filter

no_of_h_manh = no_of_host.filter(like='Manhattan')
pd.DataFrame(no_of_h_manh)
```

Out[116]:

		host_id
neighbourhood_group	**host_id**	
Manhattan	**2845**	2
	3867	2
	4396	2
	4632	1
	7192	1
	...	...
	274103383	1
	274188386	1
	274273284	1
	274311461	1
	274321313	1

15080 rows × 1 columns

In [117…

```
#count no. of columns with len func.

len(no_of_h_manh.index)
```

Out[117]:

```
15080
```

15,080 Hosts are in Manhattan City.

In [118…

```
#find city wise avg price
```

```
avg_grp_neighhood = df1.groupby('neighbourhood_group')['price'].mean()
round(avg_grp_neighhood,2)
```

Out[118]:

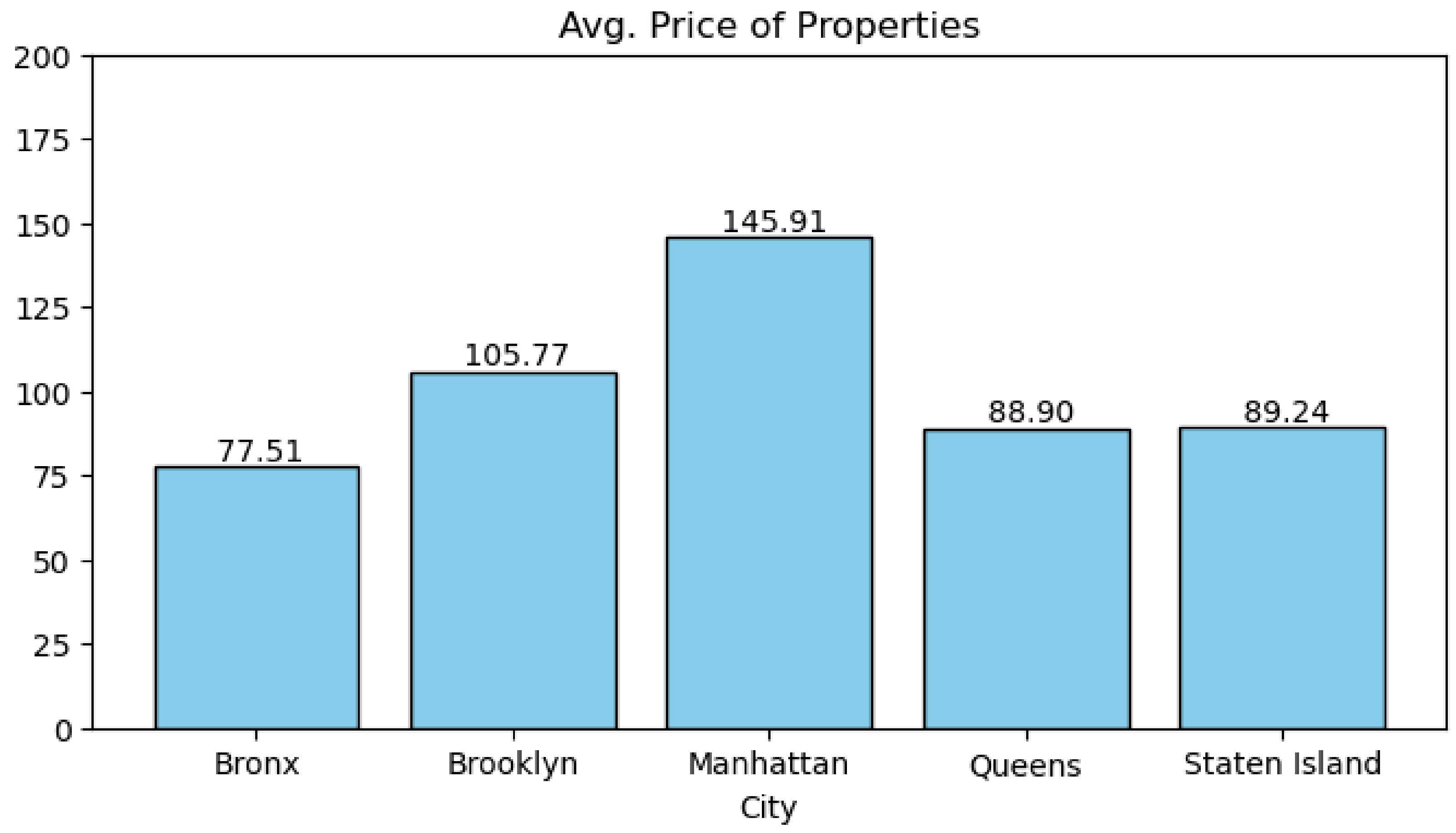
```
neighbourhood_group
Bronx             77.51
Brooklyn         105.77
Manhattan        145.91
Queens            88.90
Staten Island     89.24
Name: price, dtype: float64
```

In [119…

```
#store data in x and y variable for plot

x5 = avg_grp_neighhood.index
y5 = avg_grp_neighhood.values
```

In [120…

```
#adjust the size of graph
plt.figure(figsize=(8,4))

#plot the graph and add data labels
bars = plt.bar(x5,y5,color='skyblue',edgecolor='black')
plt.bar_label(bars,label_type='edge',fmt=' '+'%.2f')
plt.ylim(0,200)

#label the title & x - axis of graph
plt.title('Avg. Price of Properties')
plt.xlabel('City')

plt.show()
```



Bronx City has the lowest Avg.Price.

In [121…

```
#find room_type wise no. of rooms

count_of_romty = df1.pivot_table(index='room_type',aggfunc='count')
count_of_romty['id']
```

Out[121]:

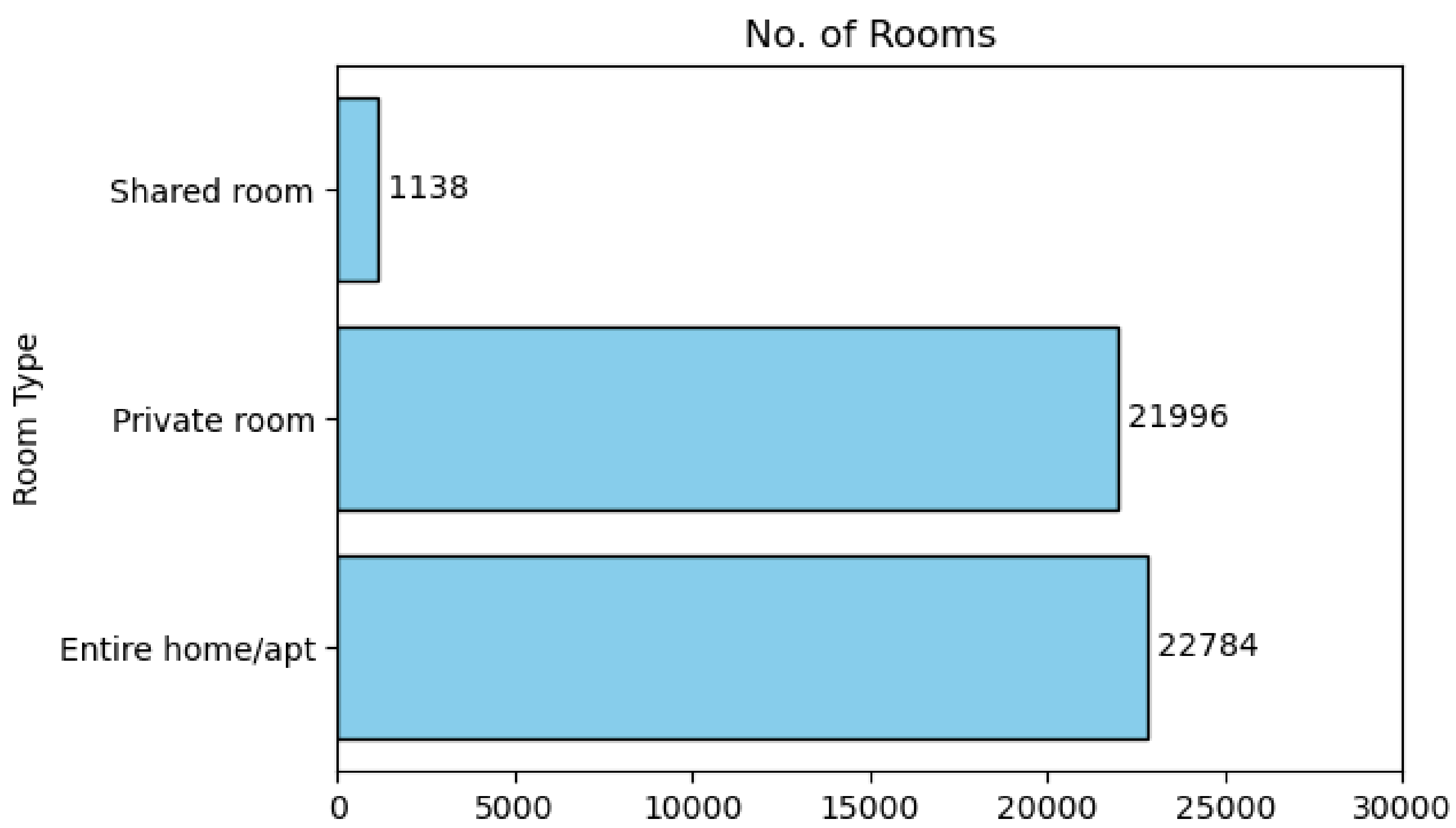
```
room_type
Entire home/apt    22784
Private room       21996
Shared room         1138
Name: id, dtype: int64
```

In [122…

```
#store data in x and y variable for plot

x1 = count_of_romty['id'].index
y1 = count_of_romty['id'].values
```

In [123…

```
#adjust the size of graph
plt.figure(figsize=(6,4))

#plot the graph and add data labels
bars = plt.barh(x1,y1,color='skyblue',edgecolor='black')
plt.bar_label(bars,label_type='edge',fmt=' '+'%.0f')
plt.xlim(0,30000)

#label the title & y - axis of graph
plt.title('No. of Rooms')
plt.ylabel('Room Type')

plt.show()
```



21,996 private rooms are in entire New York.

In [124…

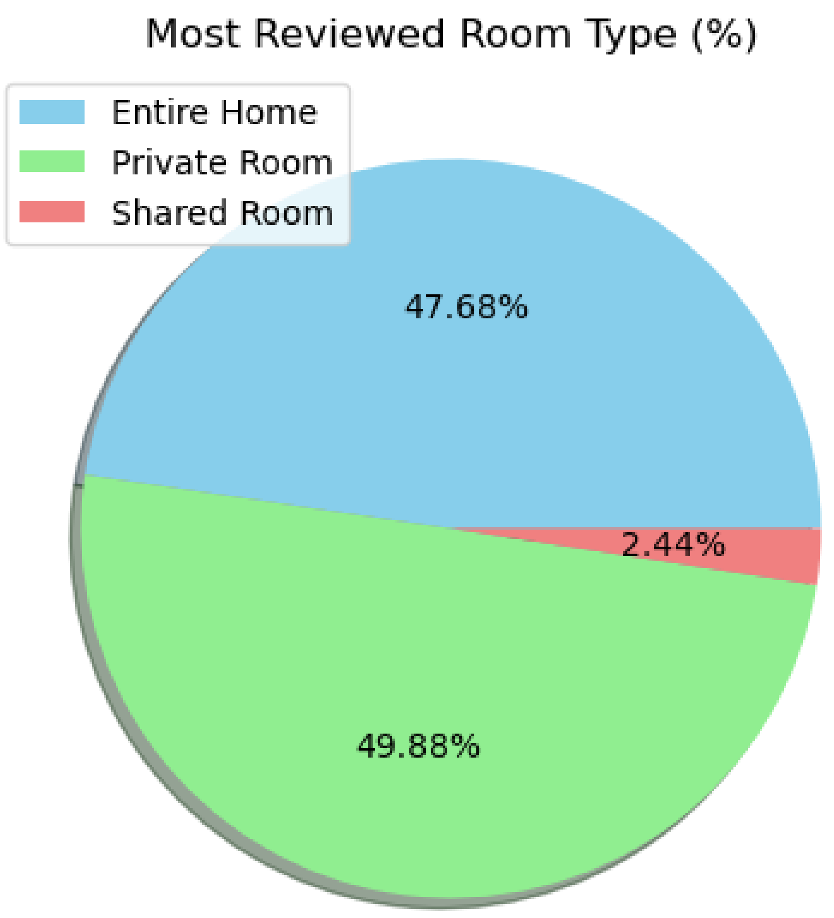
```
#find room_type wise reviews per month

review_roomty = df1.pivot_table(index='room_type',aggfunc='sum')
review_roomty['reviews_per_month']
```

Out[124]:

```
room_type
Entire home/apt    24255.09
Private room       25372.46
Shared room         1242.55
Name: reviews_per_month, dtype: float64
```

In [125…

```
#adjust the size of graph
plt.figure(figsize=(20,5))

#plot the graph and add data labels
plt.pie(review_roomty['reviews_per_month'],autopct='%.2f%%',shadow=True,colors=['skyblue','lightgreen','lightcoral'])

#label the title & legend of graph
plt.title('Most Reviewed Room Type (%)')
plt.legend(['Entire Home','Private Room','Shared Room'],loc=2)

plt.show()
```



Private rooms has the most reviewed room_type in NYC.

In [126…

```
#find neighbourhood_group & neighbourhood wise avg price

no_of_area = df1.groupby(['neighbourhood_group','neighbourhood'])['price'].mean()
no_of_area
```

Out[126]:

```
neighbourhood_group  neighbourhood
Bronx                Allerton           78.756098
                     Baychester         75.428571
                     Belmont            77.125000
                     Bronxdale          57.105263
                     Castle Hill        63.000000
                                          ...
Staten Island        Tompkinsville      76.190476
                     Tottenville       144.857143
                     West Brighton      80.555556
                     Westerleigh        71.500000
                     Willowbrook       249.000000
Name: price, Length: 219, dtype: float64
```

In [127…

```
#filter queens city in neighbourhood_group
no_of_p_area = no_of_area.filter(like='Queens')

#store the filtered data in new dataframe
no_of_p_area = pd.DataFrame(no_of_p_area)

#filter the price by greater than 100 & less than 150
greater_price = no_of_p_area[(no_of_p_area['price'] >= 100) & (no_of_p_area['price'] <= 150)]
greater_price
```

Out[127]:

		price
neighbourhood_group	**neighbourhood**	
Queens	**Arverne**	135.097222
	Bay Terrace	142.000000
	Belle Harbor	146.000000
	Holliswood	135.750000
	Howard Beach	115.400000
	Jamaica Estates	136.941176
	Jamaica Hills	132.125000
	Kew Gardens Hills	100.840000
	Long Island City	111.236434
	Middle Village	109.580645
	Rockaway Beach	124.672727

In [128…

```
#find the no. of areas by len func.

len(greater_price)
```

Out[128]: 11

11 Areas where have price between 100 USD to 150 USD in Queens City .

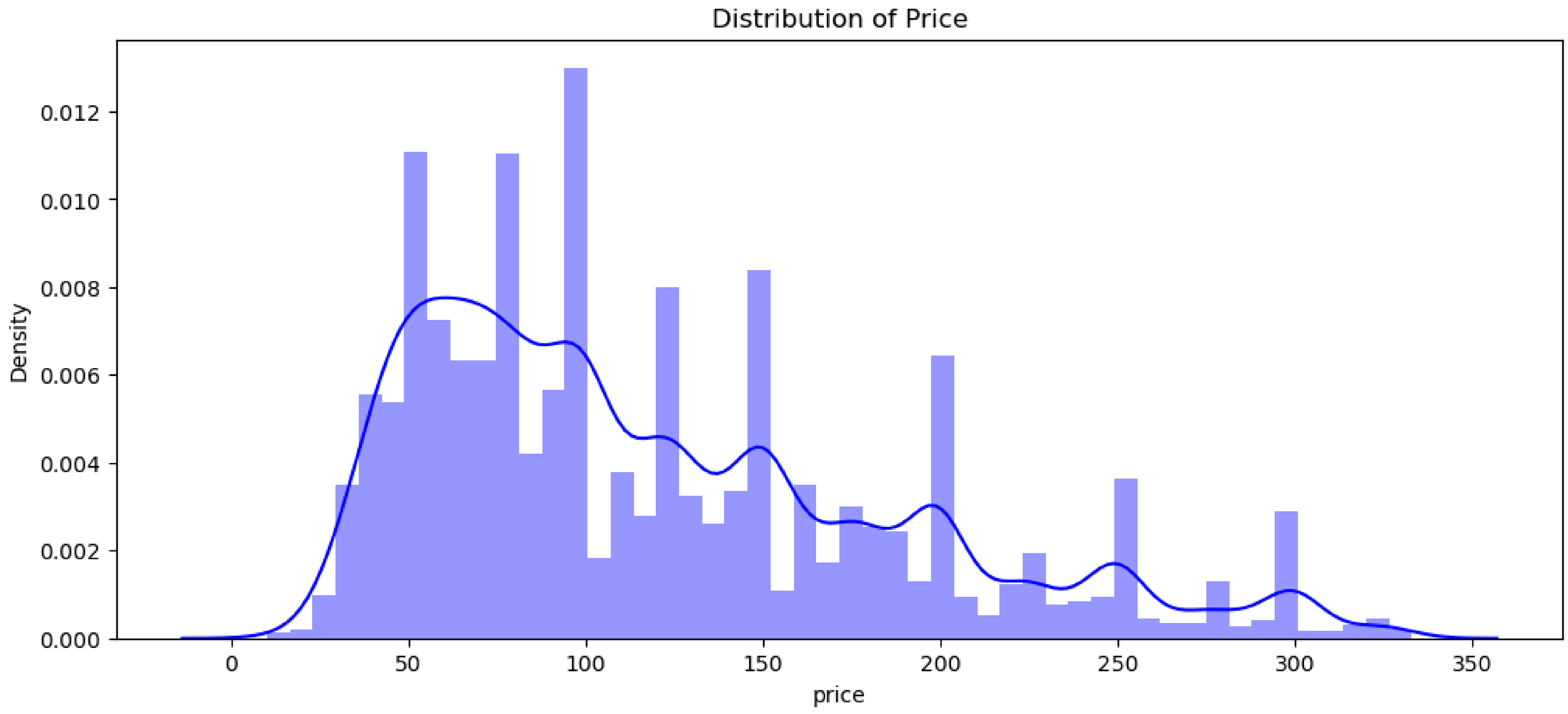
Analysis the Problem Statement

Let's go on 1st Probem Statement -

In [129…
```
#adjust the size of graph
plt.figure(figsize=(12,5))

#plot the graph
sns.distplot(df1['price'],kde=True,color=('b'))

#label the title
plt.title('Distribution of Price')
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_28336\544408645.py:4: 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(df1['price'],kde=True,color=('b'))
```

Out[129]: Text(0.5, 1.0, 'Distribution of Price')



- As per the above fig, We observe that the price charged on airbnb appears to be from **20 to 330 USD (Approx)**, with the majority of listing properties are falling in the price range of **50 to 150 USD**.
- With the close observation of graph, We also find a pattern that with the increasing in price , the density of listing properties getting relatively lower.
- We can clearly observe that only fewer listing properties are available at price **above 250 USD**.

Let's go on 2nd Probem Statement -

In [130…
```
#find neighbourhood wise no. of listing property
top_neigh = df1.groupby('neighbourhood')['neighbourhood'].count()

#find top 10 neighhood with the help of sort_values func.
top_10_neigh = top_neigh.sort_values(ascending=False).head(10)

#convert into Dataframe
top_10_neigh = pd.DataFrame(top_10_neigh)
top_10_neigh
```

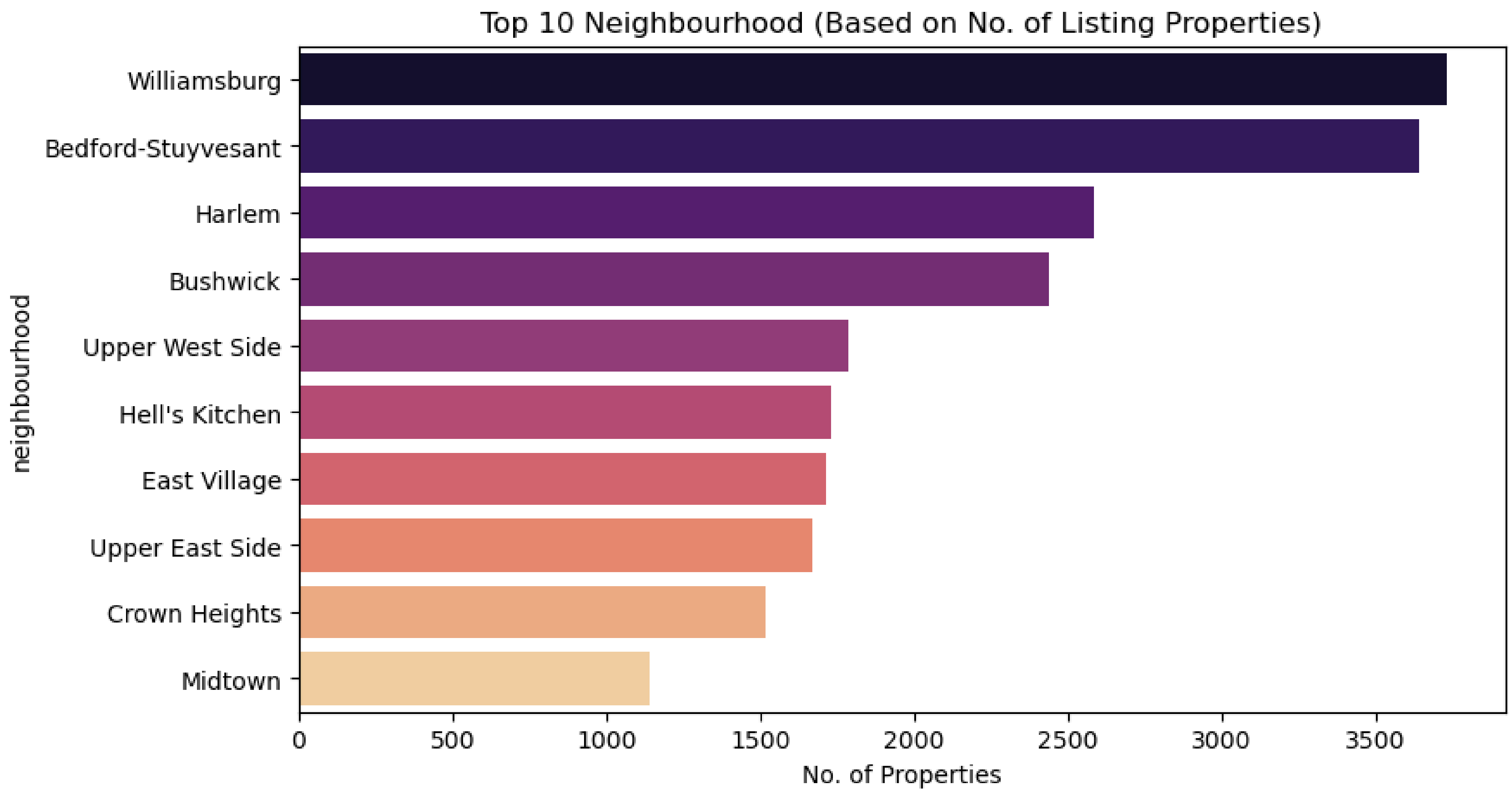
Out[130]:

neighbourhood	neighbourhood
Williamsburg	3732
Bedford-Stuyvesant	3638
Harlem	2585
Bushwick	2438
Upper West Side	1788
Hell's Kitchen	1731
East Village	1714
Upper East Side	1670
Crown Heights	1519
Midtown	1143

In [131…
```
#store data in x and y variable for plot

x1  = top_10_neigh.index
y1 = top_10_neigh['neighbourhood'].values
x1
```

Out[131]:
```
Index(['Williamsburg', 'Bedford-Stuyvesant', 'Harlem', 'Bushwick',
       'Upper West Side', 'Hell's Kitchen', 'East Village', 'Upper East Side',
       'Crown Heights', 'Midtown'],
      dtype='object', name='neighbourhood')
```

In [133…
```
#adjust the size of graph
plt.figure(figsize = (9,5))

#plot the graph and add data labels
sns.barplot(x = y1 ,y = x1,orient='h',palette='magma')

#label the title & x- axis of graph
plt.title('Top 10 Neighbourhood (Based on No. of Listing Properties)')
plt.xlabel('No. of Properties')

plt.show()
```



Let's go on 3rd Probem Statement -

In [134…
```
#find neighbourhood wise no. of listing property
top_hosts = df1.groupby('host_name')['host_name'].count()

#find top 10 neighbood with the help of sort_values func.
top_10_hosts = top_hosts.sort_values(ascending=False).head(10)

#convert into Dataframe
top_10_hosts = pd.DataFrame(top_10_hosts)
top_10_hosts
```

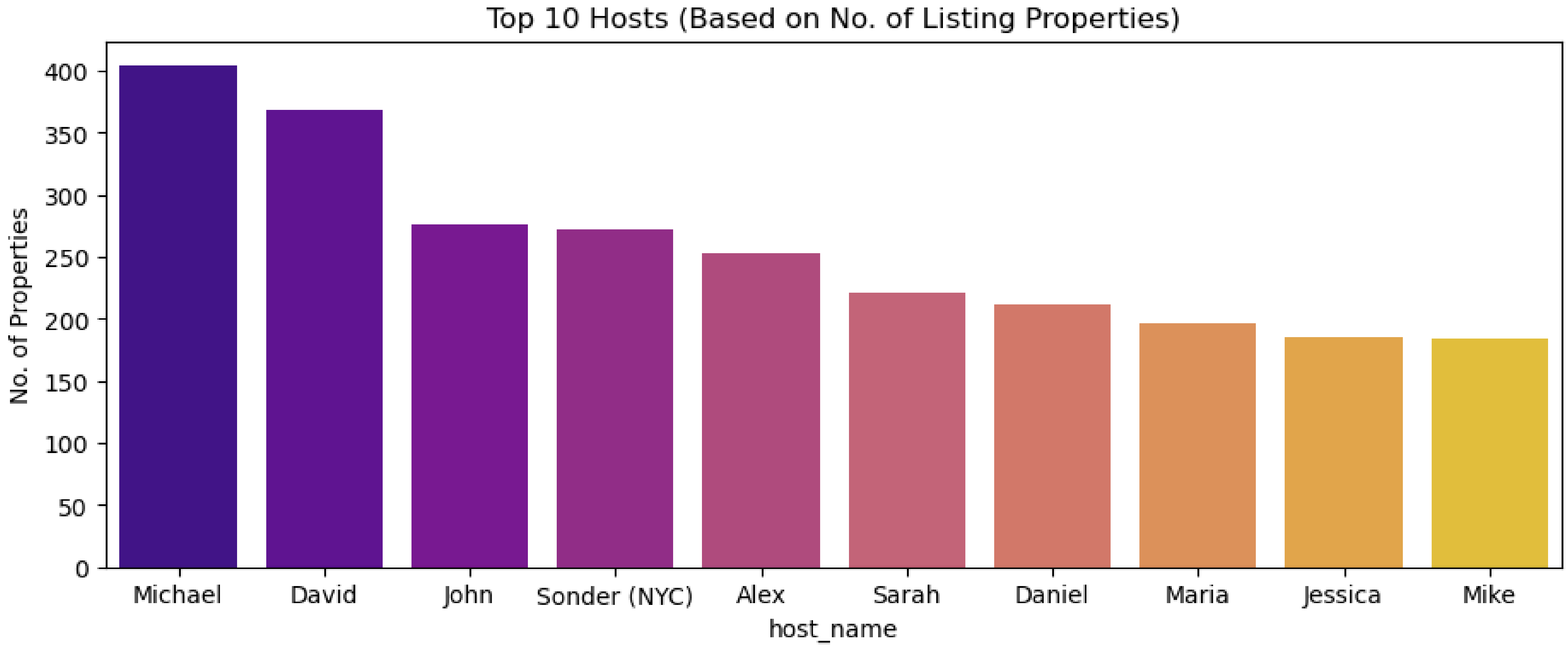
Out[134]:

	host_name
host_name	
Michael	404
David	368
John	276
Sonder (NYC)	272
Alex	253
Sarah	221
Daniel	212
Maria	197
Jessica	185
Mike	184

In [135…
```
#store data in x and y variable for plot

x2 = top_10_hosts.index
y2 = top_10_hosts['host_name'].values
x2
```

Out[135]:
```
Index(['Michael', 'David', 'John', 'Sonder (NYC)', 'Alex', 'Sarah', 'Daniel',
       'Maria', 'Jessica', 'Mike'],
      dtype='object', name='host_name')
```

In [137…
```
#adjust the size of graph
plt.figure(figsize = (11,4))

#plot the graph and add data labels
sns.barplot(x = x2, y = y2, palette='plasma')

#label the title & y- axis of graph
plt.title('Top 10 Hosts (Based on No. of Listing Properties)')
plt.ylabel('No. of Properties')

plt.show()
```



Let's go on 4th Problem Statement -

In [138…

```
#create dataframe of columns neighbourhood, price, number_of_reviews

best_locat = df1[['neighbourhood','price','number_of_reviews']]
best_locat
```

Out[138]:

	neighbourhood	price	number_of_reviews
0	Kensington	149	9
1	Midtown	225	45
2	Harlem	150	0
3	Clinton Hill	89	270
4	East Harlem	80	9
...	...	...	...
45913	Bedford-Stuyvesant	70	0
45914	Bushwick	40	0
45915	Harlem	115	0
45916	Hell's Kitchen	55	0
45917	Hell's Kitchen	90	0

45918 rows × 3 columns

In [139…

```
#firstly, find neighbourhood wise no. of reviews then, sort no. of reviews in descending order
best_locat_r = best_locat.pivot_table(index='neighbourhood',aggfunc='mean').sort_values(by='number_of_reviews',ascending=False)

#display starting 5 neighbourhood
five_best_locat_r = best_locat_r.head(5)
five_best_locat_r
```

Out[139]:

neighbourhood	number_of_reviews	price
Silver Lake	118.500000	70.000000
East Elmhurst	82.097826	77.820652
Richmondtown	79.000000	78.000000
Eltingville	76.000000	141.666667
Mount Eden	70.000000	58.500000

In [140…

```
#store data in x and y variable for plot

x2 = five_best_locat_r.index
y2 = five_best_locat_r['number_of_reviews'].values
```

In [141…

```
#firstly, find neighbourhood wise avg price then, sort price in ascendng order
best_locat_p = best_locat.pivot_table(index='neighbourhood',aggfunc='mean').sort_values(by='price')

##display starting 5 neighbourhood
five_best_locat_p = best_locat_p.head(5)
five_best_locat_p
```

Out[141]:

neighbourhood	number_of_reviews	price
Bull's Head	15.333333	47.333333
Hunts Point	9.777778	50.500000
Tremont	20.636364	51.545455
Soundview	29.400000	53.466667
Corona	28.507937	54.412698

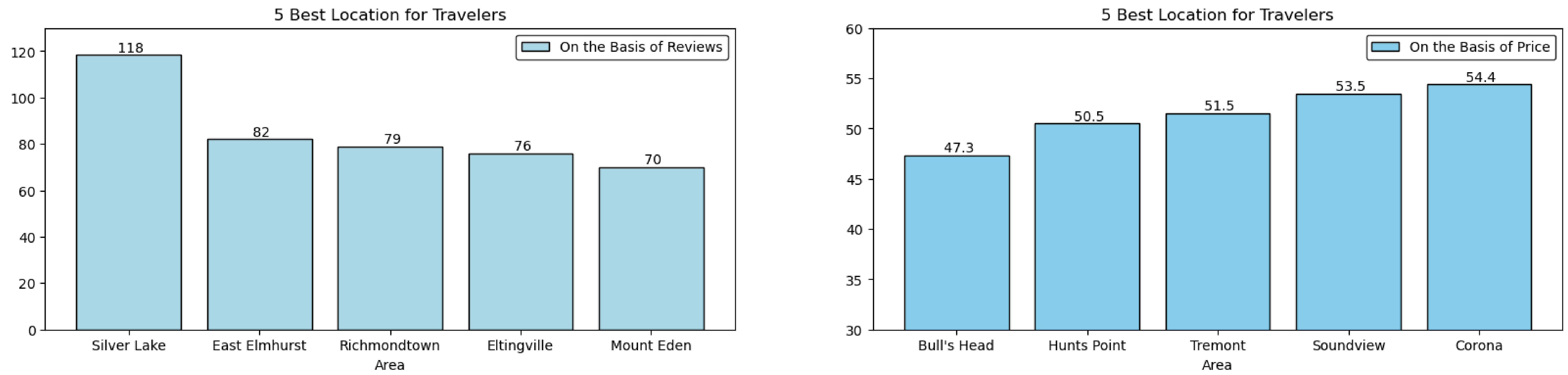
In [142…

```
#store data in x and y variable for plot

x3 = five_best_locat_p.index
y3 = five_best_locat_p['price'].values
```

In [143…

```
#adjust the size of graph
plt.figure(figsize=(20,4))
```

```
#plot bar in 1st column of figure
plt.subplot(121)

bars1 = plt.bar(x2,y2,label='On the Basis of Reviews',color='lightblue',edgecolor='black')
plt.xlabel('Area')
plt.ylim(0,130)
plt.bar_label(bars1,label_type='edge',fmt=' '+'%.f')
plt.title('5 Best Location for Travelers')
plt.legend(edgecolor='black')


#plot bar in 2nd column of figure
plt.subplot(122)

bars = plt.bar(x3,y3,label='On the Basis of Price',color='skyblue',edgecolor='black')
plt.ylim(30,60)
plt.bar_label(bars,label_type='edge',fmt=' '+'%.1f')
plt.title('5 Best Location for Travelers')
plt.xlabel('Area')
plt.legend(edgecolor='black')


plt.show()
```



* Those Travelers whose pritority is Reviews ,then **Silver Lake** is the Best Location.
* Those Travelers whose pritority is Price ,then **Bull's Head** is the Best Location.

Let's go on 5th Probem Statement -

In [144…

```
#find neighbourhood_group , room_type wise avg price
avg_price = df1.groupby(['neighbourhood_group','room_type'])['price'].mean()

#round off avg price upto 2 decimal
avg_price = round(avg_price,2)
avg_price
```

Out[144]:

```
neighbourhood_group  room_type
Bronx                Entire home/apt    112.20
                     Private room        60.83
                     Shared room         47.25
Brooklyn             Entire home/apt    148.22
                     Private room        70.37
                     Shared room         48.78
Manhattan            Entire home/apt    181.63
                     Private room        98.06
                     Shared room         75.94
Queens               Entire home/apt    131.33
                     Private room        65.73
                     Shared room         46.99
Staten Island        Entire home/apt    121.09
                     Private room        62.29
                     Shared room         57.44
Name: price, dtype: float64
```

In [145…

```
#convert room_type row to column with the help of unstack

avg_price_plot = avg_price.unstack()
avg_price_plot
```

Out[145]:

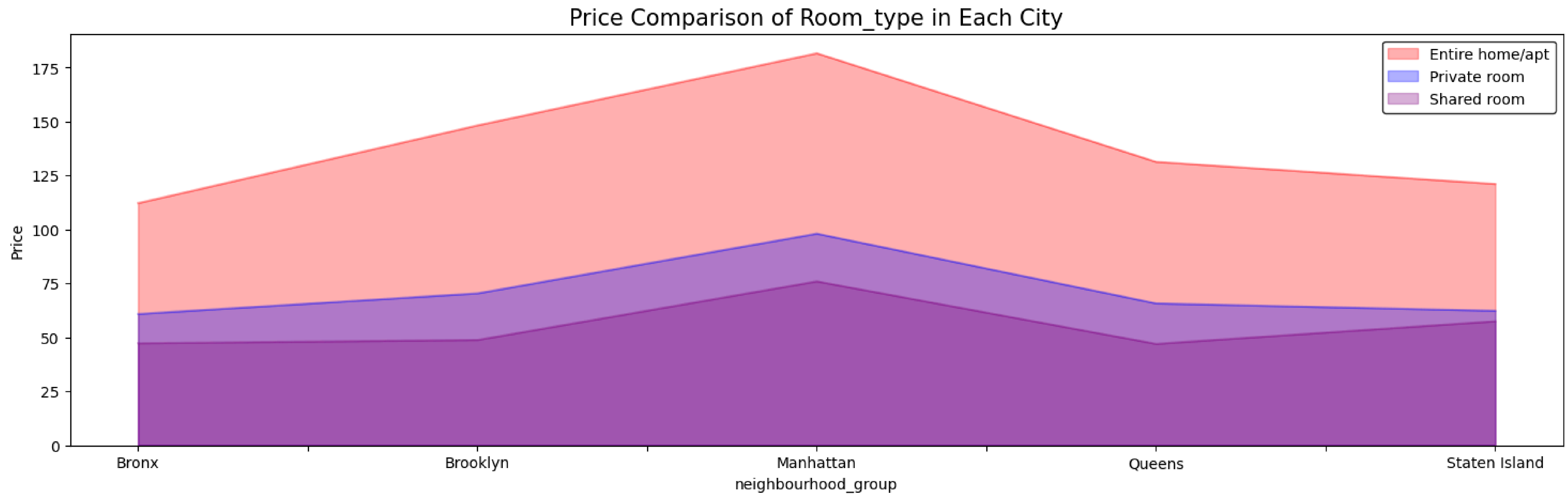
room_type	Entire home/apt	Private room	Shared room
neighbourhood_group			
Bronx	112.20	60.83	47.25
Brooklyn	148.22	70.37	48.78
Manhattan	181.63	98.06	75.94
Queens	131.33	65.73	46.99
Staten Island	121.09	62.29	57.44

In [146…

```
#plot a area chart & adjust the size of a chart
avg_price_plot.plot(kind='area',figsize = (18,5),stacked = False,color = ['r','blue','purple'],alpha = 0.3)

#label the title , y- axis & legend of graph
plt.title('Price Comparison of Room_type in Each City',fontsize= 15)
plt.legend(edgecolor='black')
plt.ylabel('Price')

plt.show()
```

Out[146]:

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



- Upon Observation of this area graph, We observed a consistent pattern across all cities suggests a common trend that **the avg price of entire home/apt is way much higher than the avg price of private & shared room_type.**
- With the help of deep observation, We clearly see that only Brooklyn & Manhattan have huge gap or difference between avg price of private & shared room compared to other cities.
- After observe the chart carefully, We find that Only Bronx & Staten island have least gap or difference between avg price of private & shared room compared to other cities which means **in these 2 cities visitor have the flexibility to choose their preferred room type without significant concern for price discrepancies.**

Let's go on 6th Problem Statement -

In [147…

```
#find neighbourhood_group , room_type wise no. of listing property

city_wise_room = df1.groupby(['neighbourhood_group','room_type'])['room_type'].count()
pd.DataFrame(city_wise_room)
```

Out[147]:

neighbourhood_group	room_type	room_type
Bronx	Entire home/apt	363
	Private room	648
	Shared room	59
Brooklyn	Entire home/apt	8942
	Private room	10062
	Shared room	411
Manhattan	Entire home/apt	11289
	Private room	7747
	Shared room	465
Queens	Entire home/apt	2022
	Private room	3351
	Shared room	194
Staten Island	Entire home/apt	168
	Private room	188
	Shared room	9

In [148…

```
#store data in labels variable for plot

labels = city_wise_room.index.get_level_values(1).unique()
labels
```

Out[148]:

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

In [149…

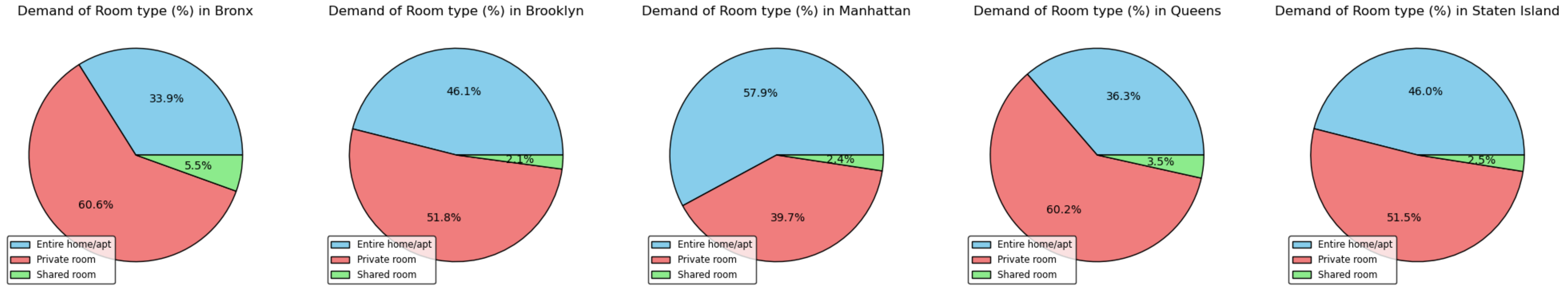
```
#store data in city_name variable for plot

city_name = city_wise_room.index.get_level_values(0).unique()
city_name = list(city_name)
city_name
```

Out[149]:

```
['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island']
```

In [150…

```
# Create 5 pie charts of 5 cities

fig, axes = plt.subplots(1,5,figsize=(25,6))
l = ['skyblue','lightcoral','lightgreen']

# Create a loop for data filter one by one in each pie.
# I created already a list of city name which help me to use one by one.

for i in range(0,5):
    axes[i].pie(city_wise_room.filter(like=city_name[i]),autopct='%.1f%%',colors=l,wedgeprops={'edgecolor':'black'})
    axes[i].legend(labels,loc=3,fontsize='small',edgecolor='black')
    axes[i].set_title('Demand of Room type (%) in ' + city_name[i])

plt.show()
```



Here are some insights or patterns -

- Upon observation of the aforementioned figure, it becomes apparent that **the demand for Shared rooms is relatively low across all cities**. This suggests a preference among residents and visitors for more private accommodations over shared living spaces.
  - We observe that **only Bronx City exhibits the highest demand percentage for Shared rooms** in comparison to all other cities. This indicates a unique preference for shared living spaces among residents and visitors specifically within the Bronx area.
- Upon closer examination, it becomes evident that **only Manhattan exhibits a higher demand percentage for Entire Home/apt compared to Private Room**. This observation underscores the desirability of having an entire home or apartment for lodging purposes in Manhattan, possibly due to the city's bustling urban environment and diverse attractions.
- It is evident from the data that **the demand percentage for Private rooms is notably high across all cities, with the exception of one (Manhattan)**. This trend suggests a widespread preference among travelers for the privacy and comfort offered by private accommodations, highlighting the importance of providing such options in the hospitality industry.

Let's go on 7th Problem Statement -

Firstly, We have to find which city have the highest avg. price ?

In [151…
```
#find city wise avg price

df1.groupby('neighbourhood_group')['price'].mean()
```

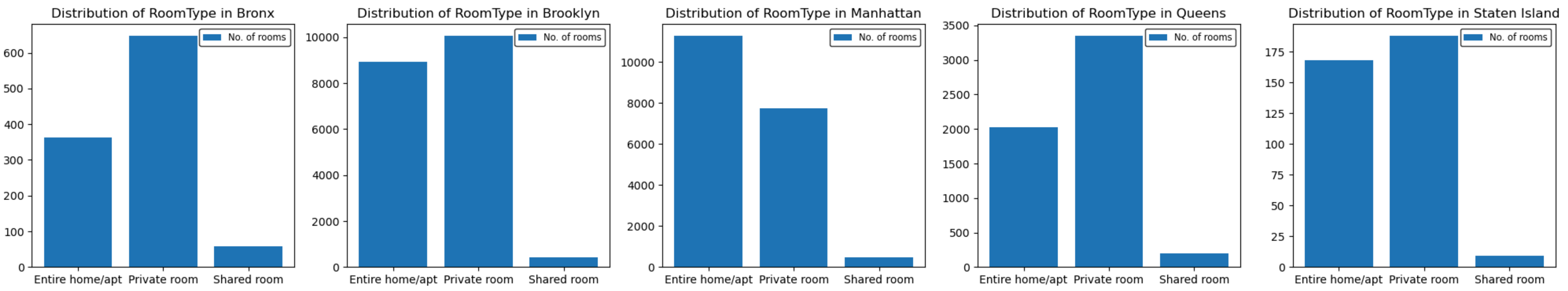
Out[151]:
```
neighbourhood_group
Bronx             77.508411
Brooklyn         105.770538
Manhattan        145.912466
Queens            88.904437
Staten Island     89.235616
Name: price, dtype: float64
```

Manhattan have the highest avg. price,then we have to find **why ?***

Let's Find the Reason behind it !!

We have to plot bar chart of room_type of each city for better visualization of distribution of room_type.

In [152…
```
# Create 5 bar charts in fig. of 1 row & 5 columns

fig, axes = plt.subplots(1,5,figsize=(25,4))

# Create a loop for data filter one by one in each bar
# i created already a list of city name which help me to use one by one

for i in range(0,5):
    axes[i].bar(labels,city_wise_room.filter(like=city_name[i]).values,label='No. of rooms')
    axes[i].legend(loc=1,fontsize='small',edgecolor='black')
    axes[i].set_title('Distribution of RoomType in ' + city_name[i])

plt.show()
```



df1.groupby('room_type')['price'].mean()

room_type
Entire home/apt - 162.50
Private room - 79.06
Shared room - 59.56

Firstly, Observe the Charts Carefully ! To find what thing makes Manhattan special or different from other cities.

*Ohh Yes!! I get it , i find that Manhattan has the highest no. of entire_room in comparison of all other cities.So, let's find the avg. price of room_type of overall cities, To understand the relation of each room_type. We see that entirehome/apt has way more pricing than other two room_type.Now, We have said that **the reason behind why manhattan high price, it is because manhattan have higehst no. of that roomtype which is overall the highest avg.price(162.50USD).**

Let's go on 8th Problem Statement -

In [153…
```
#create a duplicate column of availability_365

df1['availability_Category']  = df1['availability_365'].values
df1.sample(5)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_28336\3964176059.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['availability_Category']  = df1['availability_365'].values
```

Out[153]:

	index	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_pe
16920	17875	14006823	51501835	Jeniffer	Manhattan	Hell's Kitchen	40.76469	-73.99394	Entire home/apt	107	30	8	04-05-2019	
17864	18864	14968436	16686968	Ricardo	Manhattan	Harlem	40.81041	-73.94337	Private room	55	2	114	27-05-2019	
41971	44463	34247102	11522108	Cecilia	Brooklyn	Park Slope	40.67428	-73.97559	Entire home/apt	150	3	4	25-06-2019	
27746	29258	22450373	15535829	Jay	Staten Island	West Brighton	40.63229	-74.11351	Entire home/apt	99	2	3	18-05-2019	
7834	8332	6402807	6354467	Robert	Manhattan	Chelsea	40.74476	-73.99862	Entire home/apt	100	4	4	03-01-2017	

In [154…

```python
#fill availability_Category column with value 'normal'
df1['availability_Category'] = 'Normal'

#fill availability_Category column with a condition
df1.loc[df1['availability_365'] == 365, 'availability_Category'] = 'Everyday Available'
df1.loc[df1['availability_365'] == 0, 'availability_Category'] = 'Busy Entire Year'

#display random rows
df1.sample(5)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_28336\4137814515.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['availability_Category'] = 'Normal'
```

Out[154]:

	index	id	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_pe
22389	23632	19117051	12775618	Jared	Brooklyn	Greenpoint	40.73736	-73.95668	Entire home/apt	185	1	4	13-11-2017	
3468	3719	2243548	9644281	Michelle	Manhattan	Lower East Side	40.72082	-73.99028	Entire home/apt	300	1	2	13-03-2016	
4461	4805	3404873	17171419	Mordechai	Manhattan	Washington Heights	40.85083	-73.92870	Private room	39	4	48	12-06-2019	
20682	21850	17554981	5162192	Amy	Manhattan	Upper West Side	40.79790	-73.96024	Entire home/apt	130	30	2	17-08-2017	
4281	4617	3231460	15384170	Jonathan	Brooklyn	Fort Greene	40.68727	-73.97200	Entire home/apt	175	3	12	31-07-2016	

In [155…

```python
#find neighbourhood_group, availability_Category wise no. of listing property

available_category1  = df1.groupby(['neighbourhood_group','availability_Category'])['availability_Category'].count()
pd.DataFrame(available_category1)
```

Out[155]:

		availability_Category
neighbourhood_group	**availability_Category**	
Bronx	**Busy Entire Year**	175
	Everyday Available	54
	Normal	841
Brooklyn	**Busy Entire Year**	7691
	Everyday Available	411
	Normal	11313
Manhattan	**Busy Entire Year**	7587
	Everyday Available	437
	Normal	11477
Queens	**Busy Entire Year**	1354
	Everyday Available	188
	Normal	4025
Staten Island	**Busy Entire Year**	40
	Everyday Available	11
	Normal	314

In [156…

```python
#filter only Busy Entire Year, Everyday Available in column availability_Category

available_category2 = available_category1.filter(axis=0,regex='E')
pd.DataFrame(available_category2)
```

Out[156]:

neighbourhood_group	availability_Category	availability_Category
Bronx	Busy Entire Year	175
	Everyday Available	54
Brooklyn	Busy Entire Year	7691
	Everyday Available	411
Manhattan	Busy Entire Year	7587
	Everyday Available	437
Queens	Busy Entire Year	1354
	Everyday Available	188
Staten Island	Busy Entire Year	40
	Everyday Available	11

In [157…

```
#store data in x & y variable for plot

x1 = available_category2.index.get_level_values(0)
y1 =  available_category2.values
x1
```

Out[157]:

```
Index(['Bronx', 'Bronx', 'Brooklyn', 'Brooklyn', 'Manhattan', 'Manhattan',
       'Queens', 'Queens', 'Staten Island', 'Staten Island'],
      dtype='object', name='neighbourhood_group')
```

In [159…

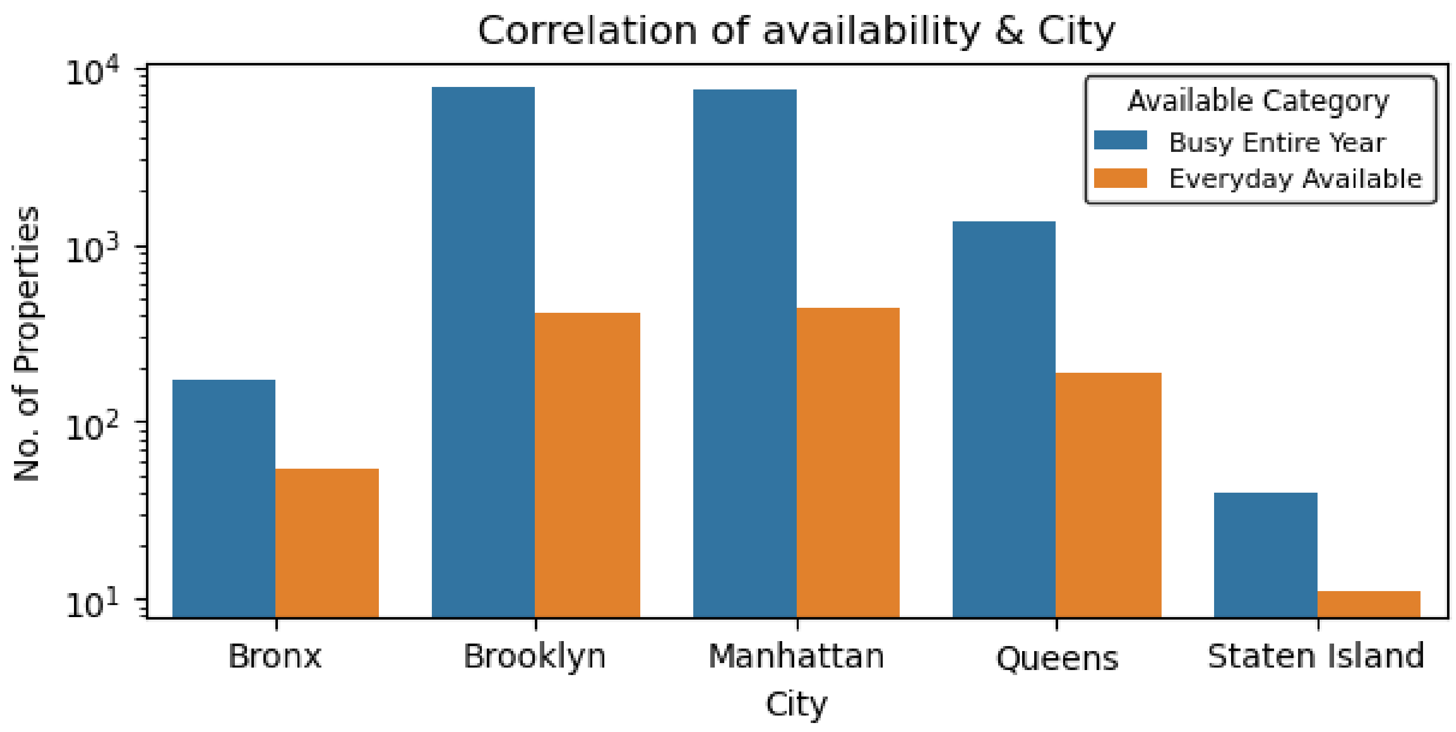
```
#store data which group in plot

grop = available_category2.index.get_level_values(1)
grop
```

Out[159]:

```
Index(['Busy Entire Year', 'Everyday Available', 'Busy Entire Year',
       'Everyday Available', 'Busy Entire Year', 'Everyday Available',
       'Busy Entire Year', 'Everyday Available', 'Busy Entire Year',
       'Everyday Available'],
      dtype='object', name='availability_Category')
```

In [160…

```
#adjust the size of graph
plt.figure(figsize=(7,3))

#plot the graph and add data labels
sns.barplot(x=x1,y=y1,hue=grop)

#label the title ,axis & legend of graph
plt.title('Correlation of availability & City')
plt.xlabel('City')
plt.ylabel('No. of Properties')
plt.legend(title='Available Category',title_fontsize='small',fontsize=8,edgecolor='black')

#convert y scale to log
plt.yscale('log')

plt.show()
```



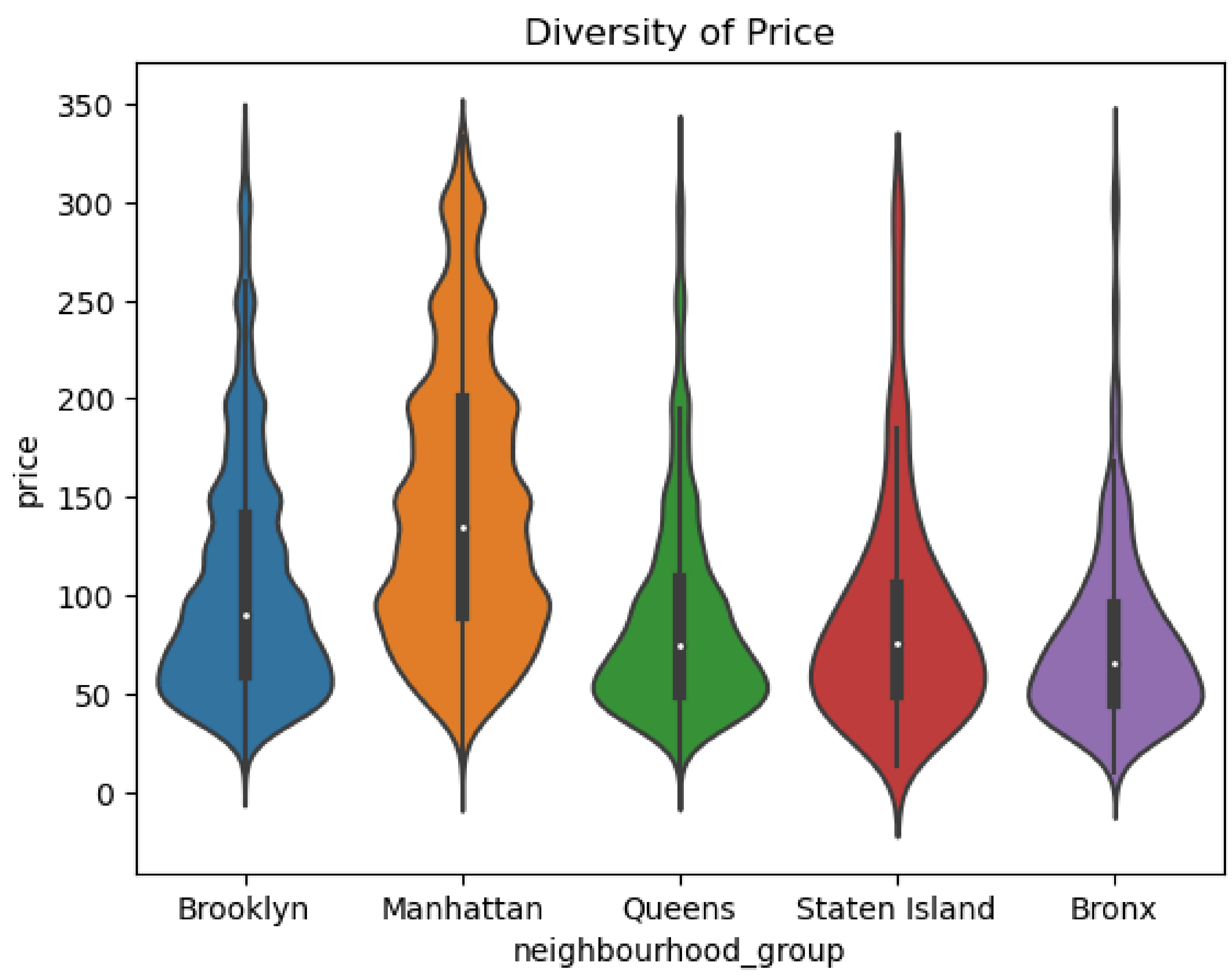
As per the above fig, We observed a consistent pattern across all cities suggests a common trend of availability that no. of properties which busy entire year is more than no. of properties which everyday available.

Let's go on 9th Problem Statement -

In [161…

```
#plot the graph and add data labels
sns.violinplot(x='neighbourhood_group',y='price',scale='width',data= df1)

#label the title
plt.title('Diversity of Price')

plt.show()
```

Out[161]: Text(0.5, 1.0, 'Diversity of Price')



- As per the above fig, We observe that Manhattan has the highest diversity of price compared to all cities means all type of price range of properties are available in Manhattan.Makes Manhattan a versatile choice for potential visitors seeking lodging options across different price points.
- With the help of above fig, We observe that Queens & Bronx both have same distribution of price between **100 to 150 USD** but queens has more diversifiy price than Bronx.

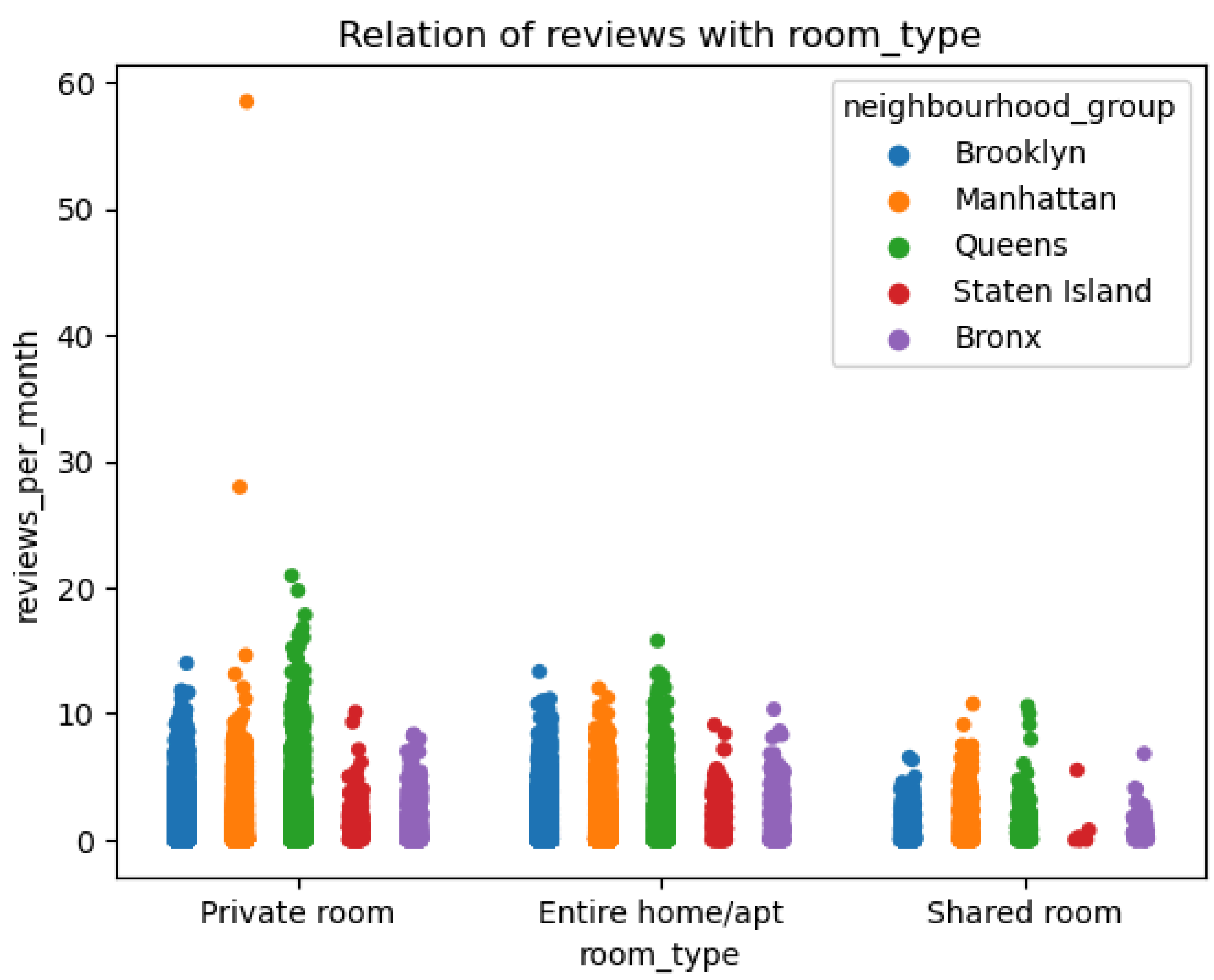
Let's go on 10th Problem Statement -

In [162…

```
#plot the graph and add data labels
sns.stripplot(data=df1 ,x = 'room_type',y = 'reviews_per_month',hue = 'neighbourhood_group',dodge=True)

#label the title
plt.title('Relation of reviews with room_type')

plt.show()
```

Out[162]:

```
Text(0.5, 1.0, 'Relation of reviews with room_type')
```



- As per the above the fig, We observe that shared room has the lowest reviews of between **0 to 10 reviews/month** compared to other two room_type which means shared room are less popular than other two room_type.
- We observe that Queens city has more reviews **more than 10 reviews/month** in Private room_type compared to entire & shared room_type which means in queens city mostly visitors preferred private room_type.
- With help of deep observation, we find that in entire home/apt room_type (Brooklyn,Manhattan,Queens) has approx equally no. of reviews/month which means entire home room_type concept is equally popular in these cities.

Let's plot some advance graph to showcase the overall correlaion & other trend or patterns.

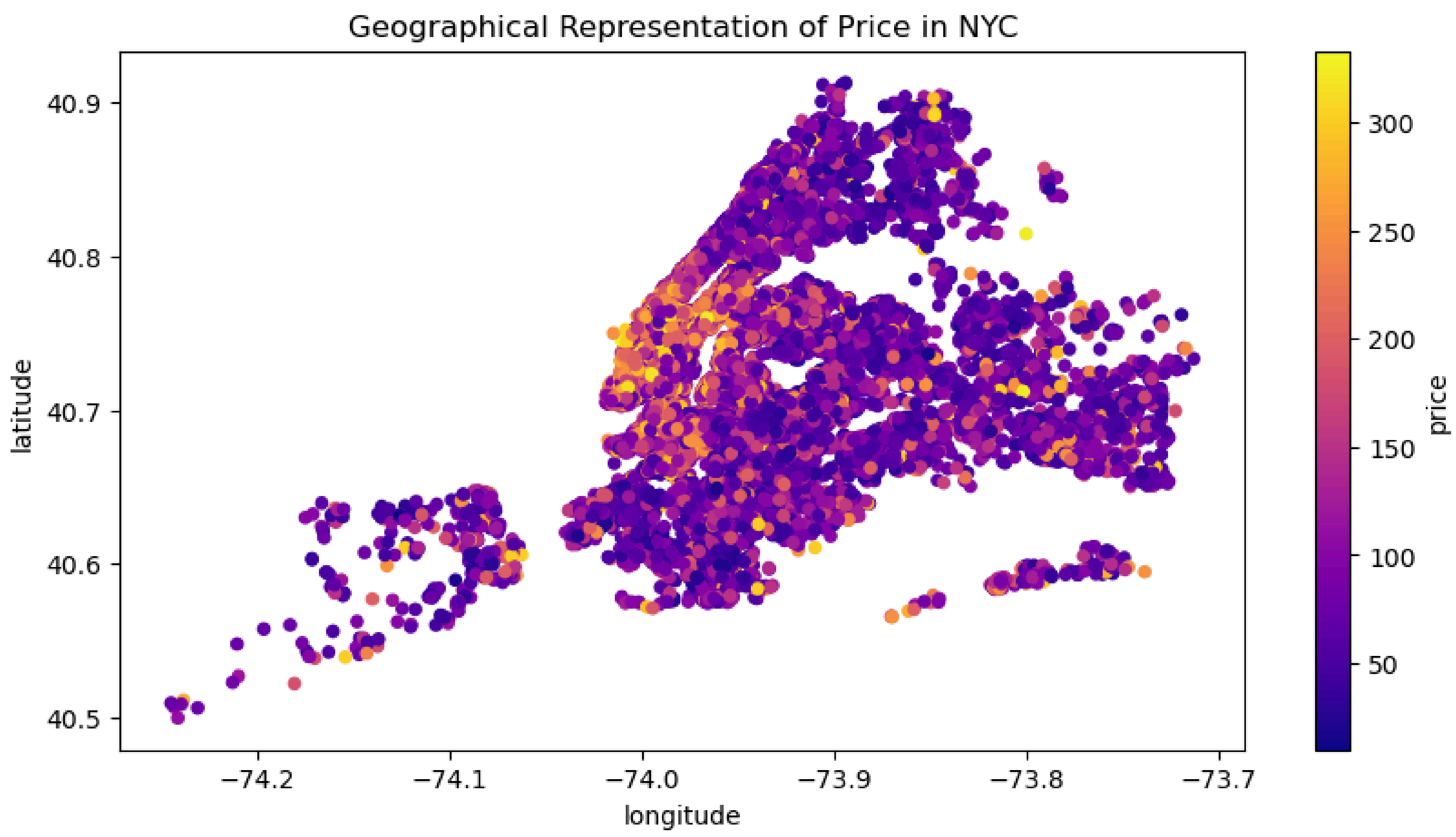
Let's showcase the 1st graph - How price is distributed on geographical basis.

In [163…

```
#plot the graph and add data labels
df1.plot.scatter(x = 'longitude',figsize=(10,5),y = 'latitude',c = 'price',cmap = 'plasma')

#label the title
plt.title('Geographical Representation of Price in NYC')

plt.show()
```

Out[163]:

```
Text(0.5, 1.0, 'Geographical Representation of Price in NYC')
```



- In this scattermap, Each point on the plot is **color-coded based on the price value**, as indicated by the color scale on the right side of the graph.
- With the help of this graph , We visualize that yellowish shades indicates areas with relatively higher prices, while the bluish color represent more affordable regions.

Let's showcase the 2nd graph - How different variables correlated with each other.

In [172…

```
#create new dataframe except some columns which have non-numerial
data_corr2 = df1.drop(columns=['host_name','neighbourhood_group','neighbourhood','last_review','room_type','availability_Category'])

#rename some columns
data_corr2.rename(columns={'calculated_host_listings_count':'host_listing_count'},inplace=True)

#display overview of new dataframe
data_corr2.head(2)
```

Out[172]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	host_listing_count	availability_365
0	2539	2787	40.64749	-73.97237	149	1	9	0.21	6	365
1	2595	2845	40.75362	-73.98377	225	1	45	0.38	2	355

In [173…

```
#create a numeric relation between columns

corr1 = data_corr2.corr()
corr1
```

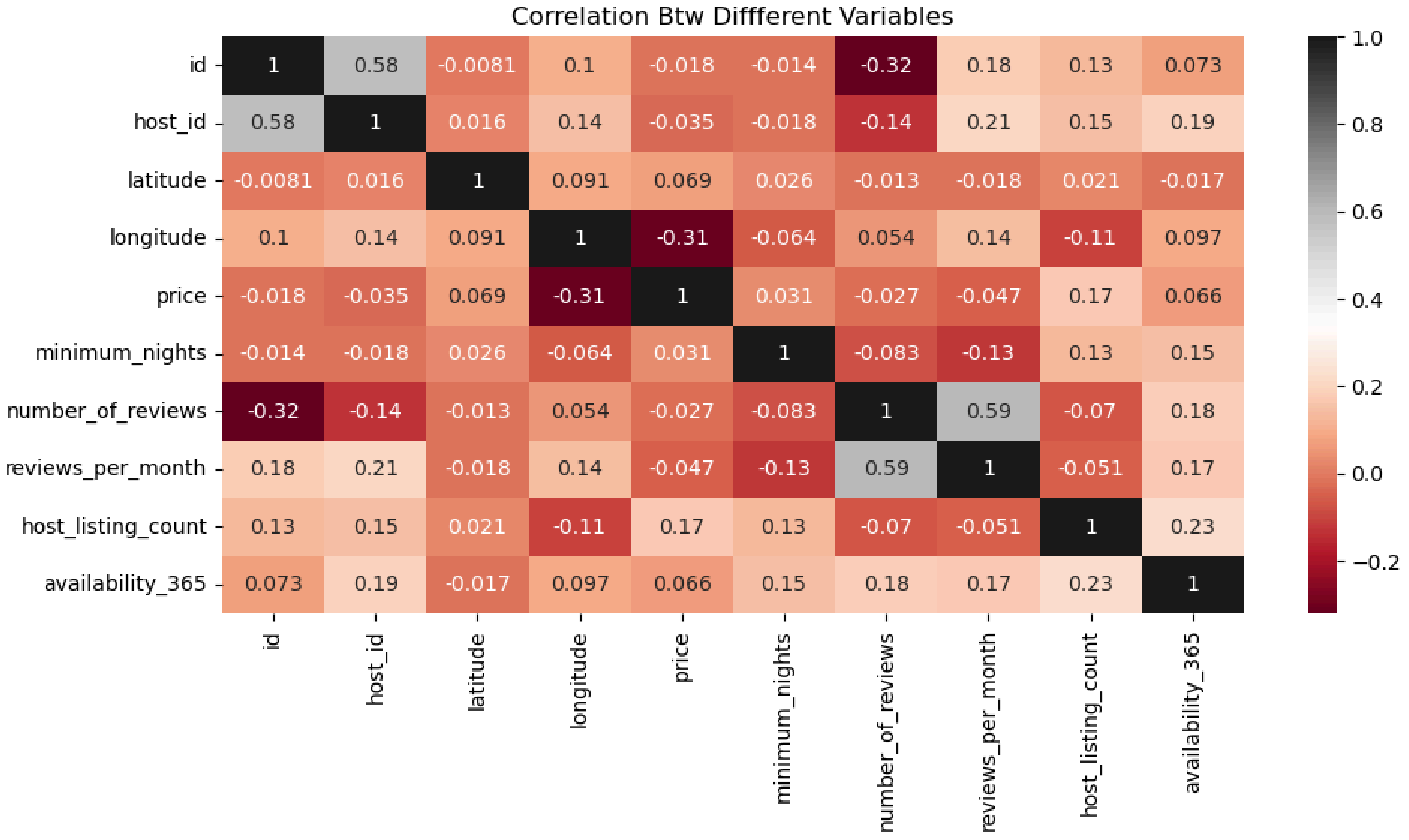
Out[173]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	host_listing_count	availability_365
id	1.000000	0.581439	-0.008072	0.101403	-0.018104	-0.013841	-0.320428	0.178978	0.125179	0.073188
host_id	0.581439	1.000000	0.015965	0.144330	-0.034878	-0.017972	-0.136529	0.208308	0.147276	0.193673
latitude	-0.008072	0.015965	1.000000	0.091354	0.068653	0.025853	-0.012515	-0.017978	0.021285	-0.017492
longitude	0.101403	0.144330	0.091354	1.000000	-0.306737	-0.064128	0.053831	0.140512	-0.107333	0.097181
price	-0.018104	-0.034878	0.068653	-0.306737	1.000000	0.031163	-0.027433	-0.047066	0.172910	0.066249
minimum_nights	-0.013841	-0.017972	0.025853	-0.064128	0.031163	1.000000	-0.082851	-0.127749	0.133237	0.146329
number_of_reviews	-0.320428	-0.136529	-0.012515	0.053831	-0.027433	-0.082851	1.000000	0.593832	-0.070357	0.183707
reviews_per_month	0.178978	0.208308	-0.017978	0.140512	-0.047066	-0.127749	0.593832	1.000000	-0.050757	0.171570
host_listing_count	0.125179	0.147276	0.021285	-0.107333	0.172910	0.133237	-0.070357	-0.050757	1.000000	0.225251
availability_365	0.073188	0.193673	-0.017492	0.097181	0.066249	0.146329	0.183707	0.171570	0.225251	1.000000

In [174…

```
#adjust the size of graph
plt.figure(figsize=(11,5))

#plot the graph
sns.heatmap(corr1,annot=True,cmap='RdGy')

#label the title
plt.title('Correlation Btw Diffferent Variables')
```

Out[174]: Text(0.5, 1.0, 'Correlation Btw Diffferent Variables')



- In this heatmap, **the color intensity in each cell represents strong or weak correlation** between variables with the help of the right side scale (-0.2 to 1).
- With the help of heatmap, We find that if cells indicates dark red intensity which means it has weak correlation **(close to 0)** and if cells indicates dark black intensity which means it has strong correlation **(close to 1)**.

BUSINESS CONCLUSION

1. With the help of price analysis, it can be highly beneficial for Airbnb's business. Airbnb understanding common price ranges **to guide hosts in setting up their listing properties price to stand out in competitive market.**

2. Through this analysis, Airbnb examine where the no. of listing properties are high which help in **decisions about where to invest in new features or services based on the popularity of neighborhoods.**

3. By this analysis, Airbnb recognizing top hosts by providing **incentives, or loyalty programs to maintain and enhance their engagement** with the platform.

4. This analysis helps Airbnb to identify the best location for travelers based on reviews and price which helps Airbnb in **marketing campaigns to attract travelers who prioritize staying in well-reviewed places & also marketing promotions for budget-conscious travelers**, showcasing locations where they can find the best value for money.

5. Through this analysis, Airbnb understands the average price differences between room types which helps Airbnb business to optimize pricing strategies like - cities Brooklyn and Manhattan where there's a significant price gap, **Airbnb could encourage hosts to offer competitive pricing for private and shared rooms to attract budget-conscious travelers.**

6. With the help of analysis of market demand, Airbnb understands the trend or pattern of demand (%) of room_type which helps business to make demand & supply strategies like - **Offer incentives to hosts in the Bronx for listing shared rooms and to Manhattan hosts for listing entire homes/apartments.**

7. This analysis helps airbnb business understanding the diversity of price range which helps in Targeted Marketing like **Manhattan offers a wide range of pricing options ,this insights help to build marketing strategies to attract visitors with varying budgets.**

8. This analysis help airbnb business to understand the popularlity of each room_type which helps **business to make decisions regarding financial investment like airbnb support host for listing private room_type in queens city financially.**

---xxx--