First of all I imported the libraries that are essential for our data analysis process. We will do make our data clean in order to perform analysis.

In [6]:
```
import pandas as pd
import numpy as nm
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.stats.multicomp as mc

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

I installed opendatastes to import data from the website of Kaggle.

In [7]:
```
!pip install opendatasets
```

```
Requirement already satisfied: opendatasets in c:\users\47455\anaconda3\lib\site-packages (0.1.22)
Requirement already satisfied: kaggle in c:\users\47455\anaconda3\lib\site-packages (from opendatasets) (1.5.13)
Requirement already satisfied: tqdm in c:\users\47455\anaconda3\lib\site-packages (from opendatasets) (4.64.0)
Requirement already satisfied: click in c:\users\47455\anaconda3\lib\site-packages (from opendatasets) (8.0.4)
Requirement already satisfied: colorama in c:\users\47455\anaconda3\lib\site-packages (from click->opendatasets) (0.4.4)
Requirement already satisfied: six>=1.10 in c:\users\47455\anaconda3\lib\site-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: python-slugify in c:\users\47455\anaconda3\lib\site-packages (from kaggle->opendatasets) (5.0.2)
Requirement already satisfied: requests in c:\users\47455\anaconda3\lib\site-packages (from kaggle->opendatasets) (2.27.1)
Requirement already satisfied: urllib3 in c:\users\47455\anaconda3\lib\site-packages (from kaggle->opendatasets) (1.26.9)
Requirement already satisfied: certifi in c:\users\47455\anaconda3\lib\site-packages (from kaggle->opendatasets) (2021.10.8)
Requirement already satisfied: python-dateutil in c:\users\47455\anaconda3\lib\site-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: text-unidecode>=1.3 in c:\users\47455\anaconda3\lib\site-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\47455\anaconda3\lib\site-packages (from requests->kaggle->opendatasets) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\47455\anaconda3\lib\site-packages (from requests->kaggle->opendatasets) (3.3)
```

In [8]:
```
import opendatasets as od
```

In [9]:
```
dataset = 'https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata'
```

In [10]:
```
od.download(dataset)
```

```
Skipping, found downloaded files in ".\airbnbopendata" (use force=True to force download)
```

In [11]:
```
import os
```

In [12]:
```
data_dir = './airbnbopendata'
```

In [13]:
```
os.listdir(data_dir)
```

Out[13]:
```
['Airbnb_Open_Data.csv']
```

In [14]:
```
frame = pd.read_csv('Airbnb_Open_Data.csv',low_memory=False)
frame
```

Out[14]:

	id	NAME	host id	host_identity_verified	host name	neighbourhood group	neighbourhood	lat	long	country	...	service fee	minimum nights	number of reviews	last review	reviews per month	review rate number	calculated host listings count	availability 365
0	1001254	Clean & quiet apt home by the park	80014485718	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	-73.97237	United States	...	$193	10.0	9.0	10/19/2021	0.21	4.0	6.0	286.0
1	1002102	Skylit Midtown Castle	52335172823	verified	Jenna	Manhattan	Midtown	40.75362	-73.98377	United States	...	$28	30.0	45.0	5/21/2022	0.38	4.0	2.0	228.0
2	1002403	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	NaN	Elise	Manhattan	Harlem	40.80902	-73.94190	United States	...	$124	3.0	0.0	NaN	NaN	5.0	1.0	352.0
3	1002755	NaN	85098326012	unconfirmed	Garry	Brooklyn	Clinton Hill	40.68514	-73.95976	United States	...	$74	30.0	270.0	7/5/2019	4.64	4.0	1.0	322.0
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Lyndon	Manhattan	East Harlem	40.79851	-73.94399	United States	...	$41	10.0	9.0	11/19/2018	0.10	3.0	1.0	289.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
102594	6092437	Spare room in Williamsburg	12312296767	verified	Krik	Brooklyn	Williamsburg	40.70862	-73.94651	United States	...	$169	1.0	0.0	NaN	NaN	3.0	1.0	227.0
102595	6092990	Best Location near Columbia U	77864383453	unconfirmed	Mifan	Manhattan	Morningside Heights	40.80460	-73.96545	United States	...	$167	1.0	1.0	7/6/2015	0.02	2.0	2.0	395.0
102596	6093542	Comfy, bright room in Brooklyn	69050334417	unconfirmed	Megan	Brooklyn	Park Slope	40.67505	-73.98045	United States	...	$198	3.0	0.0	NaN	NaN	5.0	1.0	342.0
102597	6094094	Big Studio-One Stop from Midtown	11160591270	unconfirmed	Christopher	Queens	Long Island City	40.74989	-73.93777	United States	...	$109	2.0	5.0	10/11/2015	0.10	3.0	1.0	386.0
102598	6094647	585 sf Luxury Studio	68170633372	unconfirmed	Rebecca	Manhattan	Upper West Side	40.76807	-73.98342	United States	...	$206	1.0	0.0	NaN	NaN	3.0	1.0	69.0

102599 rows × 26 columns

Removing the duplicates

In [15]:
```
# Count the number of rows in the original DataFrame
print('Number of rows before removing duplicates:', len(frame))

# Drop duplicate rows based on all columns
frame = frame.drop_duplicates()

# Count the number of rows in the updated DataFrame
print('Number of rows after removing duplicates:', len(frame))
```

```
Number of rows before removing duplicates: 102599
Number of rows after removing duplicates: 102058
```

checking that do we have null values in our data.

In [128…
```
print(frame.isnull().sum())
```

```
id                                     0
NAME                                 250
host id                                0
host_identity_verified               289
host name                            404
neighbourhood group                   29
neighbourhood                         16
lat                                    8
long                                   8
country                              532
country code                         131
instant_bookable                     105
cancellation_policy                   76
room type                              0
Construction year                    214
price                                247
service fee                          273
minimum nights                       400
number of reviews                    183
last review                        15832
reviews per month                  15818
review rate number                   319
calculated host listings count       319
availability 365                     448
house_rules                        51842
license                           102056
dtype: int64
```

I wanted to see the column names so, I performed the below code

In [17]:
```
print(frame.columns)
```

```
Index(['id', 'NAME', 'host id', 'host_identity_verified', 'host name',
       'neighbourhood group', 'neighbourhood', 'lat', 'long', 'country',
       'country code', 'instant_bookable', 'cancellation_policy', 'room type',
       'Construction year', 'price', 'service fee', 'minimum nights',
       'number of reviews', 'last review', 'reviews per month',
       'review rate number', 'calculated host listings count',
       'availability 365', 'house_rules', 'license'],
      dtype='object')
```

In [18]:
```
print(frame.head())
```

```
        id                                              NAME      host id  \
0  1001254                Clean & quiet apt home by the park  80014485718
1  1002102                             Skylit Midtown Castle  52335172823
2  1002403               THE VILLAGE OF HARLEM....NEW YORK !  78829239556
3  1002755                                               NaN  85098326012
4  1003689  Entire Apt: Spacious Studio/Loft by central park  92037596077

  host_identity_verified host name neighbourhood group neighbourhood  \
0            unconfirmed  Madaline            Brooklyn    Kensington
1               verified     Jenna           Manhattan       Midtown
2                    NaN     Elise           Manhattan        Harlem
3            unconfirmed     Garry            Brooklyn  Clinton Hill
4               verified    Lyndon           Manhattan   East Harlem

        lat      long        country  ... service fee minimum nights  \
0  40.64749 -73.97237  United States  ...       $193            10.0
1  40.75362 -73.98377  United States  ...        $28            30.0
2  40.80902 -73.94190  United States  ...       $124             3.0
3  40.68514 -73.95976  United States  ...        $74            30.0
4  40.79851 -73.94399  United States  ...        $41            10.0

  number of reviews last review  reviews per month review rate number  \
0               9.0  10/19/2021               0.21                4.0
1              45.0   5/21/2022               0.38                4.0
2               0.0         NaN                NaN                5.0
3             270.0    7/5/2019               4.64                4.0
4               9.0  11/19/2018               0.10                3.0

  calculated host listings count  availability 365  \
0                            6.0             286.0
1                            2.0             228.0
2                            1.0             352.0
3                            1.0             322.0
4                            1.0             289.0

                                         house_rules license
0  Clean up and treat the home the way you'd like...     NaN
1  Pet friendly but please confirm with me if the...     NaN
2  I encourage you to use my kitchen, cooking and...     NaN
3                                                NaN     NaN
4  Please no smoking in the house, porch or on th...     NaN

[5 rows x 26 columns]
```

Checking the Shape of the data.

In [20]:
```
print(frame.shape)
```

```
(102058, 26)
```

Used the mode function to fill the null values of each data.

In [21]:
```
frame_copy=frame.copy()
frame_copy['long'].fillna(frame_copy['long'].mode()[0], inplace=True)
frame_copy['NAME'].fillna(frame_copy['NAME'].mode()[0], inplace=True)
frame_copy['neighbourhood group'].fillna(frame_copy['neighbourhood group'].mode()[0],inplace = True)
frame_copy['neighbourhood'].fillna(frame_copy['neighbourhood'].mode()[0],inplace = True)
frame_copy['host name'].fillna(frame_copy['host name'].mode()[0],inplace=True)
#replacing null values using mode in name column
frame_copy['lat'].fillna(frame_copy['lat'].mode()[0],inplace=True)
#replacing null values using mode in name column
frame_copy['long'].fillna(frame_copy['long'].mode()[0], inplace=True)
#replacing null values using mode in name column
frame_copy['country'].fillna(frame_copy['country'].mode()[0], inplace=True)
#replacing null values using mode in name column
frame_copy['country code'].fillna(frame_copy['country code'].mode()[0], inplace=True)
#replacing null values using mode in name column
frame_copy['instant_bookable'].fillna(frame_copy['instant_bookable'].mode()[0], inplace=True)
#replacing null values using mode in name column
frame_copy['cancellation_policy'].fillna(frame_copy['cancellation_policy'].mode()[0], inplace=True)
frame_copy['host_identity_verified'].fillna(frame_copy['host_identity_verified'].mode()[0], inplace=True)

frame_copy['Construction year'].fillna(frame_copy['Construction year'].mode()[0], inplace=True)
frame_copy['service fee'].fillna(frame_copy['service fee'].mode()[0], inplace=True)
frame_copy['minimum nights'].fillna(frame_copy['minimum nights'].mode()[0], inplace=True)
frame_copy['number of reviews'].fillna(frame_copy['number of reviews'].mode()[0], inplace=True)
frame_copy['last review'].fillna(frame_copy['last review'].mode()[0], inplace=True)
frame_copy['reviews per month'].fillna(frame_copy['reviews per month'].mode()[0], inplace=True)
frame_copy['review rate number'].fillna(frame_copy['review rate number'].mode()[0], inplace=True)
frame_copy['calculated host listings count'].fillna(frame_copy['calculated host listings count'].mode()[0], inplace=True)
frame_copy['availability 365'].fillna(frame_copy['availability 365'].mode()[0], inplace=True)
frame_copy['house_rules'].fillna(frame_copy['house_rules'].mode()[0], inplace=True)
frame_copy['price'].fillna(frame_copy['price'].mode()[0], inplace=True)
```

In [22]:
```
frame = frame_copy.copy()
```

In [133…
```
print(frame.isnull().sum())
```

```
id                                     0
NAME                                   0
host id                                0
host_identity_verified                 0
host name                              0
neighbourhood group                    0
neighbourhood                          0
lat                                    0
long                                   0
country                                0
country code                           0
instant_bookable                       0
cancellation_policy                    0
room type                              0
Construction year                      0
price                                  0
service fee                            0
minimum nights                         0
number of reviews                      0
last review                            0
reviews per month                      0
review rate number                     0
calculated host listings count         0
availability 365                       0
house_rules                            0
license                           102056
dtype: int64
```

to remove the license column I used the drop.

In [134…
```
frame = frame.drop("license", axis=1)
```

In [135…
```
print(frame.isnull().sum())
```

```
id                                0
NAME                              0
host id                           0
host_identity_verified            0
host name                         0
neighbourhood group               0
neighbourhood                     0
lat                               0
long                              0
country                           0
country code                      0
instant_bookable                  0
cancellation_policy               0
room type                         0
Construction year                 0
price                             0
service fee                       0
minimum nights                    0
number of reviews                 0
last review                       0
reviews per month                 0
review rate number                0
calculated host listings count    0
availability 365                  0
house_rules                       0
dtype: int64
```

The 'types_of_rooms' variable contains the percentage of each type of room in the dataset, rounded to one decimal point.

In [136…
```
types_of_rooms = frame['room type'].value_counts(normalize=True).mul(100).round(1)
types_of_rooms
```

Out[136]:
```
Entire home/apt    52.4
Private room       45.4
Shared room         2.2
Hotel room          0.1
Name: room type, dtype: float64
```

The explanation of the code below,

The first four lines of code convert the price column from object to float data type, by removing commas and dollar signs, and then parsing it as numeric data using pandas. The next two lines of code create two groups based on the room type (Entire home/apt and Private room) and extract the price values for each group. The stats.ttest_ind() function from the scipy library is then used to perform an independent two-sample t-test between the two groups, assuming unequal variances.

Finally, the t-statistic and p-value for the t-test are printed using f-strings.

The purpose of this code is to perform a statistical analysis to test whether there is a significant difference in mean price between the two groups of listings (Entire home/apt and Private room). The t-test is a common method for comparing means of two groups and determining whether the difference between them is statistically significant or not.

In [23]:
```
from scipy import stats
frame['price'] = frame['price'].astype(str)
frame['price'] = frame['price'].str.replace(',', '').str.replace('$', '', regex=False)
frame['price'] = frame['price'].astype(float)
frame['price'] = pd.to_numeric(frame['price'])
# Split data into two groups based on room type
group1 = frame[frame['room type'] == 'Entire home/apt']['price']
group2 = frame[frame['room type'] == 'Private room']['price']

# Perform t-test
t_stat, p_value = stats.ttest_ind(group1, group2, equal_var=False)

# Print results
print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")
```

```
T-statistic: 0.10435722164673732
P-value: 0.9168860835285177
```

In the code below, I performed an ANOVA (Analysis of Variance) test to determine if there are significant differences in mean price between three groups of listings, based on their neighbourhood_group feature.

In [24]:
```
# create three groups based on neighbourhood_group
group1 = frame[frame['neighbourhood group'] == 'Brooklyn']['price']
group2 = frame[frame['neighbourhood group'] == 'Manhattan']['price']
group3 = frame[frame['neighbourhood group'] == 'Queens']['price']

# perform ANOVA test
f_statistic, p_value = stats.f_oneway(group1, group2, group3)

# print results
print("F-statistic:", f_statistic)
print("P-value:", p_value)
```

```
F-statistic: 3.1372044124800285
P-value: 0.043408308935618714
```

In [25]:
```
# perform one-way ANOVA
f_stat, p_val = stats.f_oneway(frame[frame['neighbourhood group'] == 'Brooklyn']['price'],
                               frame[frame['neighbourhood group'] == 'Manhattan']['price'],
                               frame[frame['neighbourhood group'] == 'Queens']['price'],
                               frame[frame['neighbourhood group'] == 'Staten Island']['price'],
                               frame[frame['neighbourhood group'] == 'Bronx']['price'])

# perform Tukey's HSD post-hoc test
m_comp = mc.MultiComparison(frame['price'], frame['neighbourhood group'])
tukey_res = m_comp.tukeyhsd()

print(tukey_res)
```

```
          Multiple Comparison of Means - Tukey HSD, FWER=0.05
=======================================================================
    group1        group2     meandiff p-adj    lower      upper   reject
-----------------------------------------------------------------------
        Bronx      Brooklyn   -1.1063    1.0   -20.5605   18.3478  False
        Bronx     Manhattan   -5.0561  0.988   -24.4837   14.3715  False
        Bronx        Queens    2.4874 0.9998   -18.2014   23.1761  False
        Bronx Staten Island   -3.9998 0.9999   -40.9394   32.9399  False
        Bronx       brookln  -46.6689    1.0 -1025.4278    932.09  False
        Bronx      manhatan -166.6689 0.9988 -1145.4278    812.09  False
     Brooklyn     Manhattan   -3.9498 0.5913    -10.656    2.7565  False
     Brooklyn        Queens    3.5937  0.933    -6.1821   13.3695  False
     Brooklyn Staten Island   -2.8934    1.0   -35.0194   29.2325  False
     Brooklyn       brookln  -45.5626    1.0 -1024.1516  933.0264  False
     Brooklyn      manhatan -165.5626 0.9989 -1144.1516  813.0264  False
    Manhattan        Queens    7.5435 0.2499    -2.1794   17.2663  False
    Manhattan Staten Island    1.0563    1.0   -31.0536   33.1663  False
    Manhattan       brookln  -41.6128    1.0 -1020.2013  936.9757  False
    Manhattan      manhatan -161.6128  0.999 -1140.2013  816.9757  False
       Queens Staten Island   -6.4871 0.9973   -39.3754   26.4012  False
       Queens       brookln  -49.1562    1.0 -1027.7706  929.4581  False
       Queens      manhatan -169.1562 0.9987 -1147.7706  809.4581  False
Staten Island       brookln  -42.6691    1.0 -1021.7618  936.4236  False
Staten Island      manhatan -162.6691  0.999 -1141.7618  816.4236  False
      brookln      manhatan    -120.0    1.0 -1503.9172 1263.9172  False
-----------------------------------------------------------------------
```

In [178…
```
print(frame['neighbourhood group'].unique())
```

```
['Brooklyn' 'Manhattan' 'brookln' 'manhatan' 'Queens' 'Staten Island'
 'Bronx']
```

In [180…
```
frame['neighbourhood group'] = frame['neighbourhood group'].replace({'brookln': 'Brooklyn', 'manhatan': 'Manhattan'})
```

In [194…
```
print(frame['neighbourhood group'].unique())
```

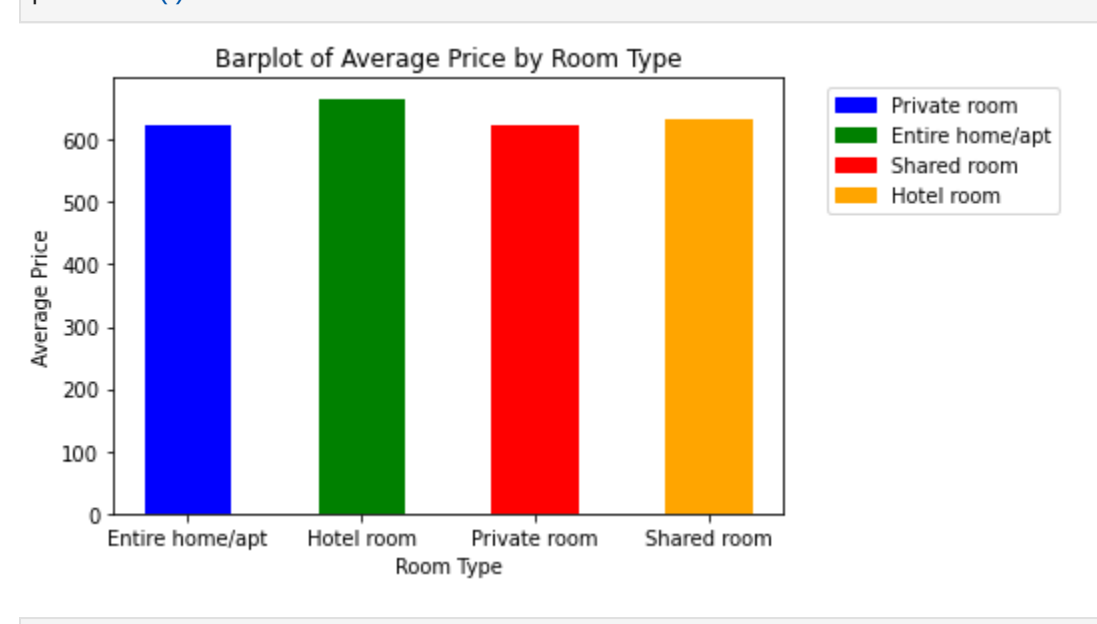
```
['Brooklyn', 'Manhattan', 'Queens', 'Staten Island', 'Bronx']
Categories (5, object): ['Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island']
```

In [198…
```
print(frame['room type'].unique())
```

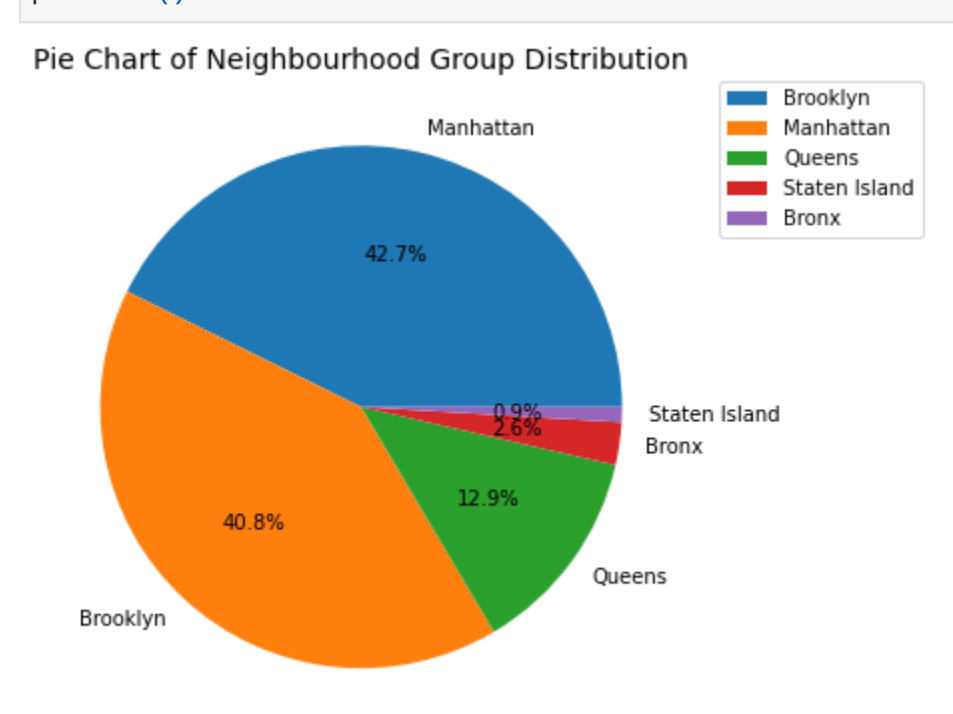
```
['Private room' 'Entire home/apt' 'Shared room' 'Hotel room']
```

Now, In the below code data analysis have been performed by using different types of graphs like barpchart, piechart, boxplot, histogram, scatterplot and violinplot.

In [200…
```
# group data by room type and calculate average price
avg_price = frame.groupby('room type')['price'].mean()

# create barplot
fig, ax = plt.subplots()
ax.bar(avg_price.index, avg_price.values, color=['blue', 'green', 'red', 'orange'], width=0.5)

# add labels and title
ax.set_xlabel('Room Type')
ax.set_ylabel('Average Price')
ax.set_title('Barplot of Average Price by Room Type')

# create legend
colors = {'Private room': 'blue', 'Entire home/apt': 'green', 'Shared room': 'red', 'Hotel room': 'orange'}
labels = list(colors.keys())
handles = [plt.Rectangle((0,0),1,1, color=colors[label]) for label in labels]
ax.legend(handles, labels, bbox_to_anchor=(1.05, 1), loc='upper left')

# display plot
plt.show()
```

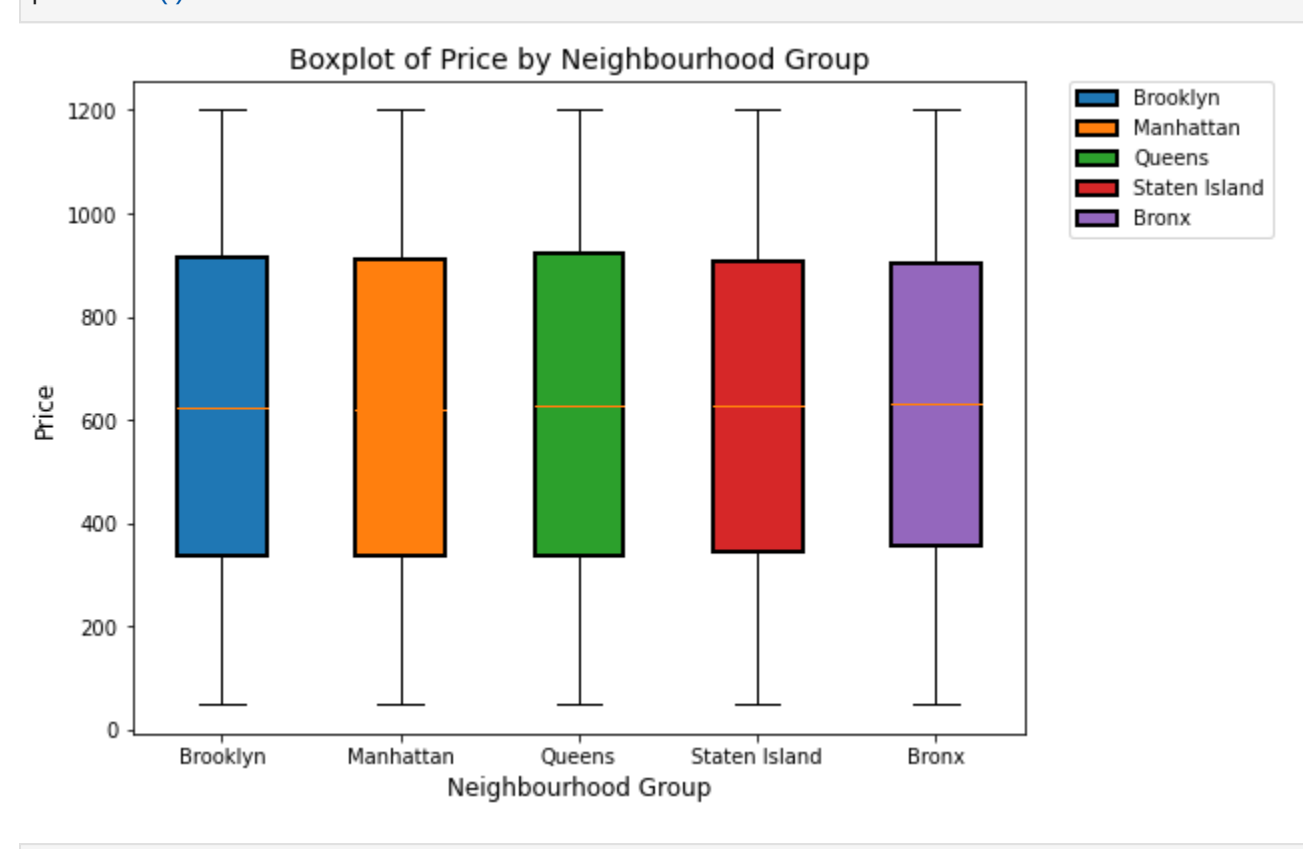


In [207…
```
import matplotlib.pyplot as plt

# create a list of labels for each neighbourhood group
labels = ['Brooklyn', 'Manhattan', 'Queens', 'Staten Island', 'Bronx']

# create a list of colors for each neighbourhood group
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']

# filter the DataFrame to include only the neighbourhood group column
ng_counts = frame['neighbourhood group'].value_counts()

# create a pie chart of neighbourhood group distribution
plt.figure(figsize=(8,6))
plt.pie(ng_counts, labels=ng_counts.index, colors=colors, autopct='%1.1f%%')

# add legend outside the chart
plt.legend(labels=labels, bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)

# set title
plt.title('Pie Chart of Neighbourhood Group Distribution', fontsize=14)

# show the plot
plt.show()
```

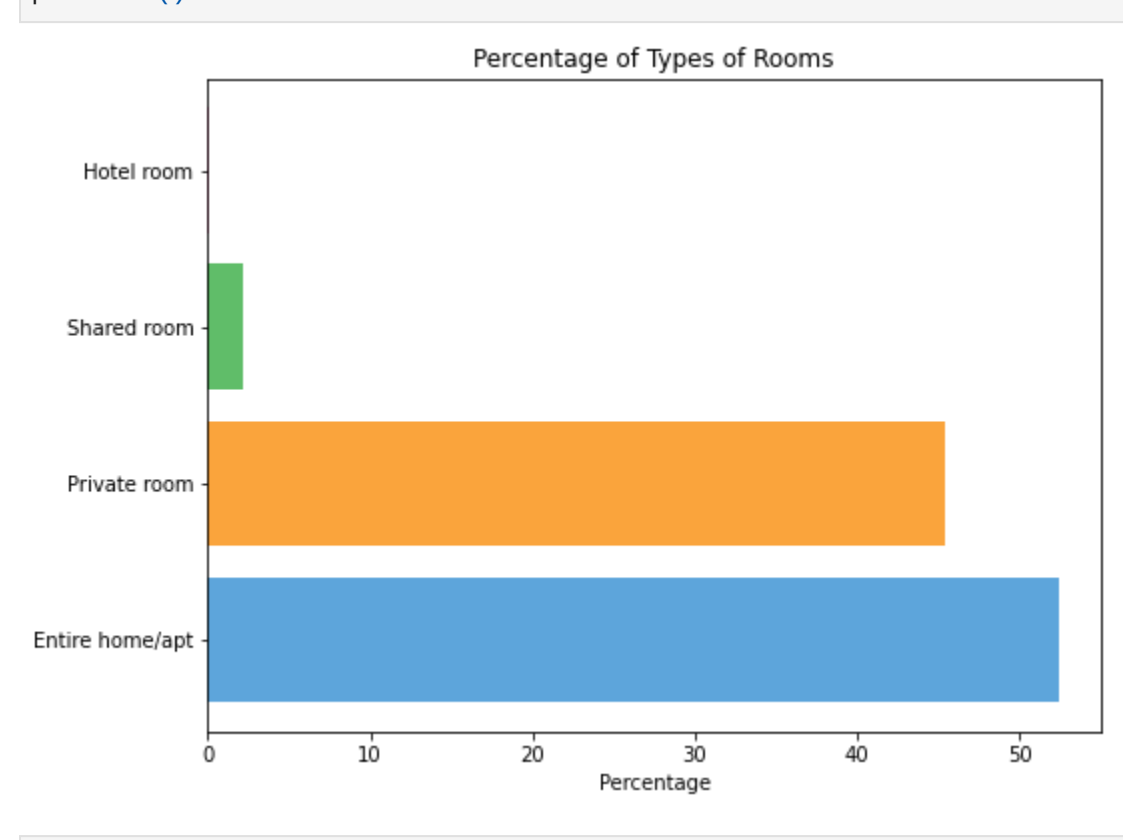


In [ ]:

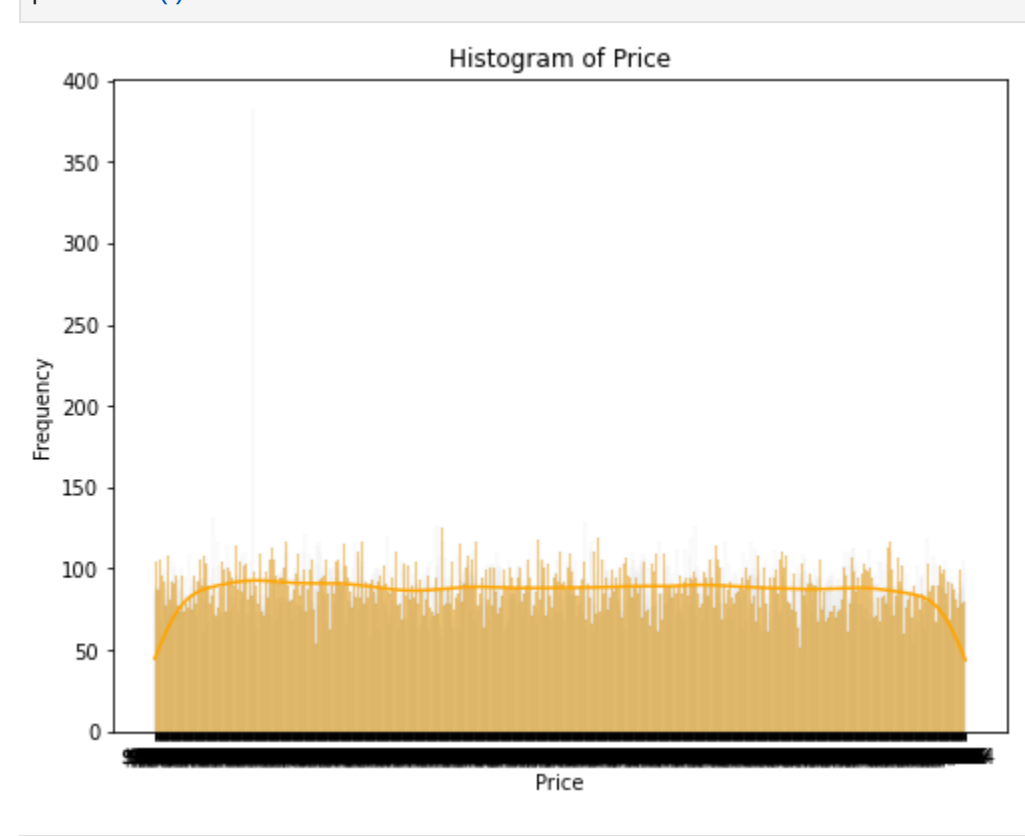
In [195…
```
# create a list of labels for each neighbourhood group
labels = ['Brooklyn', 'Manhattan', 'Queens', 'Staten Island', 'Bronx']

# define colors for each box
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']

# define box properties
boxprops = dict(linestyle='-', linewidth=2, color='black')

# create a boxplot of price by neighbourhood group
plt.figure(figsize=(8,6))
boxplots = plt.boxplot([frame[frame['neighbourhood group']=='Brooklyn']['price'],
            frame[frame['neighbourhood group']=='Manhattan']['price'],
            frame[frame['neighbourhood group']=='Queens']['price'],
            frame[frame['neighbourhood group']=='Staten Island']['price'],
            frame[frame['neighbourhood group']=='Bronx']['price']],
             labels=labels, boxprops=boxprops, patch_artist=True)

# set colors for each box
for patch, color in zip(boxplots['boxes'], colors):
    patch.set_facecolor(color)
    patch.set_label('')

# set axis labels and title
plt.xlabel('Neighbourhood Group', fontsize=12)
plt.ylabel('Price', fontsize=12)
plt.title('Boxplot of Price by Neighbourhood Group', fontsize=14)

# add legend
plt.legend(handles=boxplots['boxes'], labels=labels, bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)

# show the plot
plt.show()
```



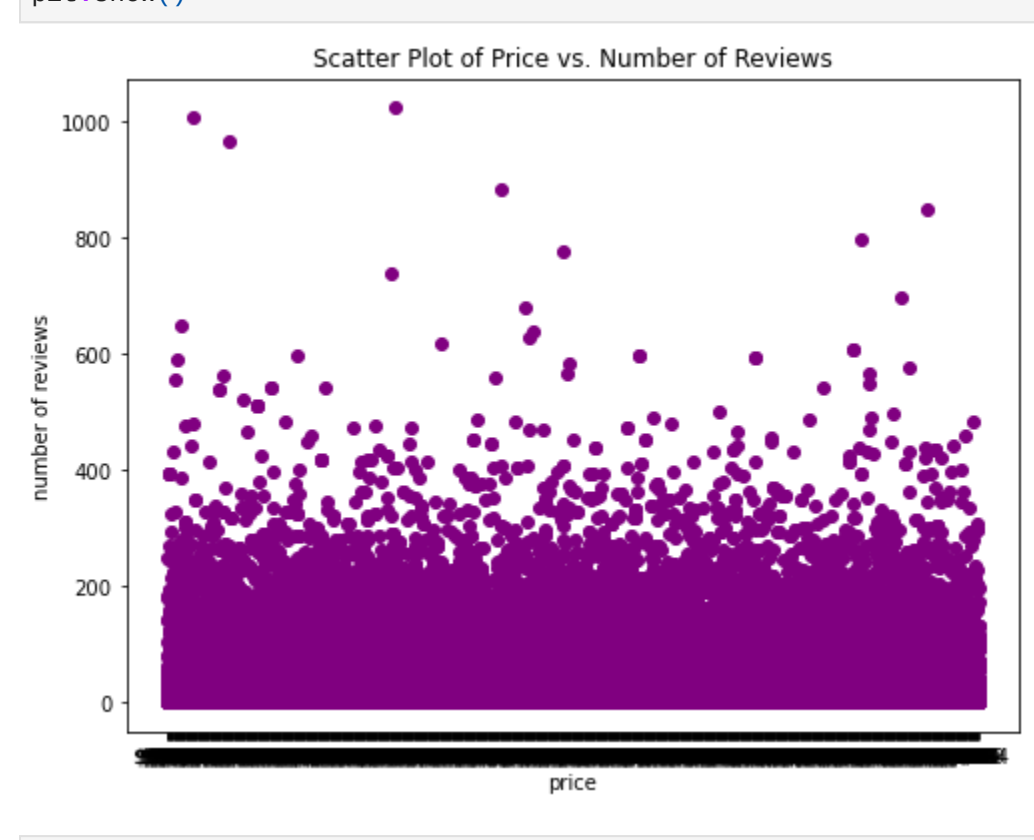
In [137…
```
# Create a horizontal bar chart
colors = ['#5DA5DB', '#FAA43C', '#60BD69', '#F17CB1']
plt.figure(figsize=(8, 6))
ax = plt.gca()
ax.barh(types_of_rooms.index, types_of_rooms.values, color=colors)
plt.title('Percentage of Types of Rooms')
plt.xlabel('Percentage')
plt.show()
```



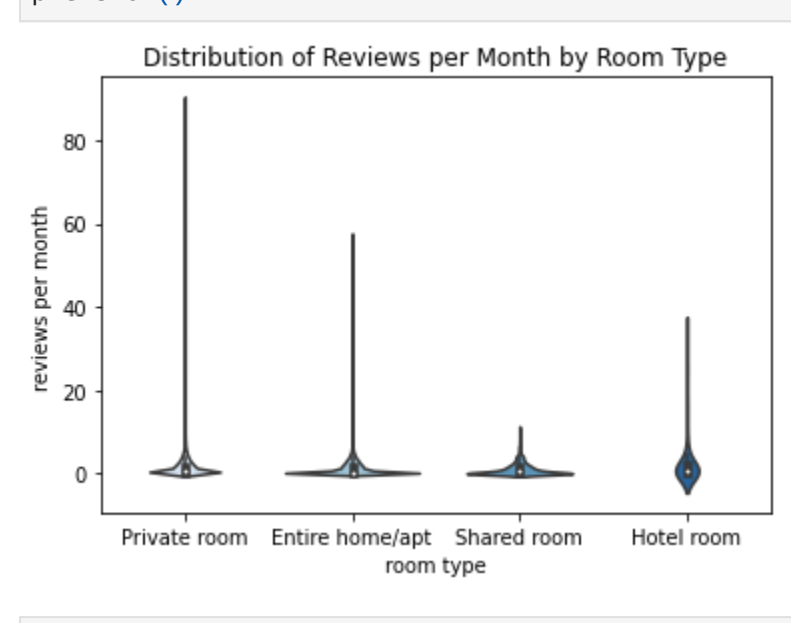
In [138…
```
plt.figure(figsize=(8, 6))
sns.histplot(data=frame, x='price', kde=True,color='orange')
plt.title('Histogram of Price')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



In [139…
```
import matplotlib.pyplot as plt
color = ['purple']
plt.figure(figsize=(8, 6))
plt.scatter(frame['price'], frame['number of reviews'],color=color)
plt.title('Scatter Plot of Price vs. Number of Reviews')
plt.xlabel('price')
plt.ylabel('number of reviews')
plt.show()
```



In [215…
```
sns.violinplot(x='room type', y='reviews per month', data=frame, palette='Blues')
plt.title('Distribution of Reviews per Month by Room Type')
plt.show()
```



In [ ]:

In [ ]:

In [ ]: