

Project Name - EDA of Airbnb Hotel Booking

Project Type - EDA

Contribution - Kashif Khan

✓ GitHub Link -

<https://github.com/KashifIqbal/EDA-of-Airbnb-Hotel-Booking>

✓ Introduction

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate.

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

This dataset has around 49,000 observations in it with 16 columns and it is a mix between categorical and numeric values.

Through EDA, Airbnb can understand pricing trends, identify popular neighborhoods, optimize search ranking, improve the guest experience, and identify opportunities for growth.

There are many factors which influence the price of a listing. Which is why we aim to find the most important factors that affect the price and more importantly the features that is common among the most expensive listings. This will allow an aspiring Airbnb host to ensure that his listing is equipped with those important features such that he will be able to charge a higher price without losing customers. Moreover, a traveler will also know the factors to look into to get the lowest price possible while having certain features he prefers.

Some of the specific findings that can be derived from EDA of Airbnb data include identifying the most popular neighborhoods for guests, understanding the factors that influence pricing, such as seasonality and demand, identifying areas where hosts need to improve, and providing targeted training and support to hosts to improve the overall guest experience.

Overall, EDA of Airbnb data is an important tool for making data-driven business decisions and improving the overall quality of the Airbnb platform. By leveraging the insights derived from EDA, Airbnb can improve the guest experience, attract more hosts and guests to its platform, and grow its business.

✓ **Dataset Description**

Id - Unique Id

Name - Name of Listing

host_id - Unique host id

host_name -Name of the host

neighbourhood_group - Location

neighbourhood - area

latitude - Latitude range

longitude - Longitude range

room_type - Type of Listing

price - Price of listing

minimum_nights - Minimum nights to be paid for

Number_of reviews - Number of reviews

last_review - Content of the last reviews

review_per month - Number of check per month

calculated_host_listing_count - Total count

availability_365 - Availability around the year

✓ ***Let's Begin !***

✓ ***1. Know Your Data***

✓ **Import Libraries**

```
1 # Importing Libraries
2
3 import pandas as pd
4 import numpy as np
5 import matplotlib.pyplot as plt
6 %matplotlib inline
7 import seaborn as sns
```

Dataset Loading

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

↗ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
1 # Load Dataset
2 file_path='/content/drive/MyDrive/Colab Notebooks/data set/capstone project/all project zip file/Airbnb NYC.csv'
3 df=pd.read_csv(file_path)
```

Dataset First View

```
1 # Dataset First Look
2 df.head(5)
```

↗

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	lat
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40
1	2595	Skyliit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40

Next steps: [Generate code with df](#) [View recommended plots](#)

```
1 #Dataset last five rows
2 df.tail()
```

↗

	id	name	host_id	host_name	neighbourhood_group	neighbourhood
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford Stuyvesant
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Ayse	Manhattan	Harlem
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen

Dataset Rows & Columns count


```
1 # Dataset Rows & Columns count
2 shape=df.shape
```

```
1 print(f'number of rows in data is {shape[0]} and number of colmns is {shape[1]}')
```

 number of rows in data is 48895 and number of colmns is 16

Dataset Information

```
1 # Dataset Info
2 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


Duplicate Values

```
1 # Dataset Duplicate Value Count
2 no_duplicate_rows=df[df.duplicated()].shape[0]
3 print(f'we have total {no_duplicate_rows} rows duplicate')
```

 we have total 0 rows duplicate

Missing Values/Null Values

```
1 # Missing Values/Null Values Count
2 df.isnull().sum().sort_values(ascending=False)[:10].reset_index().rename(columns={"index" : 'columns' , 0 : "null values"})
```

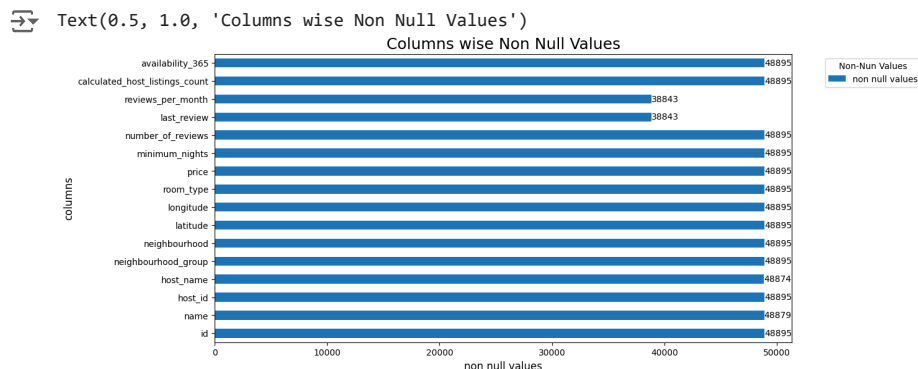


	columns	null values
0	last_review	10052
1	reviews_per_month	10052
2	host_name	21
3	name	16
4	id	0
5	host_id	0
6	neighbourhood_group	0
7	neighbourhood	0
8	latitude	0
9	longitude	0

```

1 # Visualizing the missing values
2
3 non_null_df=df.notnull().sum().reset_index().rename(columns={'index' : 'columns' ,0 : 'non null values'}) #only taking non null values
4
5 na_vis=non_null_df.plot(kind='barh', x='columns' ,figsize=(12,6)) # using matplotlib for making horizontal bar chart
6
7 na_vis.bar_label(na_vis.containers[0])
8
9 plt.xlabel('non null values' ,{'fontsize':12 , 'fontweight':15})
10 plt.ylabel('columns' ,{'fontsize':12 , 'fontweight': 15})
11 plt.legend(title='Non-Nun Values' ,bbox_to_anchor=(1.05,1),loc=2) #specifies where this anchor point is located relative to the axes
12 plt.title('Columns wise Non Null Values' , {'fontsize': 18 , 'fontweight':20})

```



We have null values in last_review, reviews_per_month, host_name and name columns. So we will have to do data wrangling to get rid of these null values so that the dataset becomes much clear to work with.

What did you know about your dataset?

We have null values in reviews_per_month ,last_review ,host_name and name column. So we have to do Data Wrangling to get rid of null values so that dataset becomes more clear

2. Understanding Your Variables

Checking do we have Duplicate Values in Variable

```

1 # checking if coulumn has duplicates or unque values using loop
2 for col in df.columns:
3     if df[col].is_unique ==True:
4         print(f'{col} is ----- unique')
5
6     if df[col].is_unique ==False:
7         print(f'{col} is ----- duplicates')

```

id is ----- unique
name is ----- duplicates
host_id is ----- duplicates
host_name is ----- duplicates
neighbourhood_group is ----- duplicates
neighbourhood is ----- duplicates
latitude is ----- duplicates
longitude is ----- duplicates
room_type is ----- duplicates
price is ----- duplicates

```

minimum_nights is ----- duplicates
number_of_reviews is ----- duplicates
last_review is ----- duplicates
reviews_per_month is ----- duplicates
calculated_host_listings_count is ----- duplicates
availability_365 is ----- duplicates

```

✓ Checking Unique Values for each variable.

```

1 # Checking Unique Values for each variable.
2 def unique_values(df):
3     unique_val = []
4
5     for col in df.columns:
6         unique_val.append(df[col].unique())
7
8     return unique_val

```

```
1 unique_values(df)
```

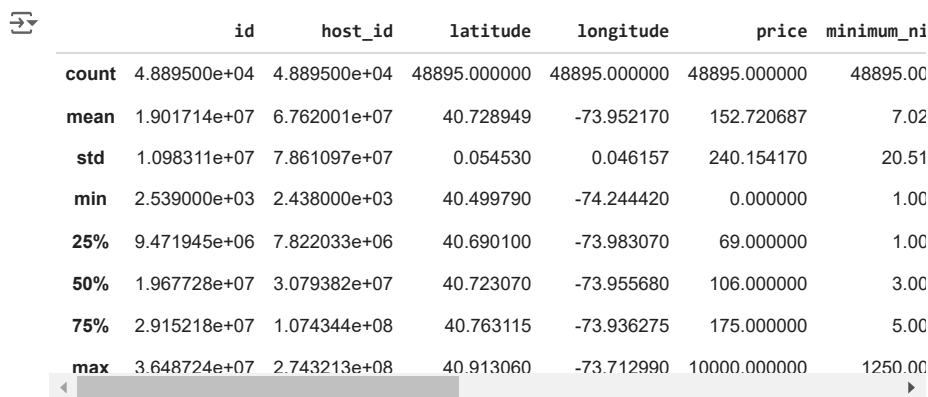
```

7.260e+00, 7.320e+00, 1.168e+01, 5.640e+00, 8.690e+00, 7.010e+00,
8.000e+00, 6.280e+00, 7.920e+00, 9.170e+00, 8.640e+00, 6.200e+00,
6.480e+00, 1.205e+01, 7.340e+00, 6.990e+00, 7.700e+00, 6.040e+00,
1.077e+01, 7.610e+00, 1.012e+01, 8.840e+00, 8.290e+00, 9.340e+00,
7.560e+00, 6.500e+00, 7.210e+00, 6.840e+00, 9.360e+00, 7.580e+00,
8.310e+00, 8.020e+00, 8.720e+00, 8.400e+00, 7.420e+00, 1.578e+01,
7.990e+00, 9.580e+00, 8.970e+00, 6.860e+00, 8.620e+00, 6.410e+00,
7.410e+00, 6.720e+00, 1.311e+01, 6.140e+00, 7.250e+00, 6.430e+00,
8.350e+00, 1.103e+01, 7.830e+00, 1.029e+01, 9.090e+00, 6.320e+00,
6.670e+00, 8.990e+00, 9.620e+00, 6.750e+00, 1.067e+01, 7.600e+00,
8.440e+00, 9.120e+00, 8.510e+00, 7.910e+00, 2.094e+01, 6.880e+00,
6.830e+00, 9.150e+00, 7.110e+00, 9.500e+00, 9.290e+00, 9.640e+00,
1.005e+01, 9.030e+00, 1.330e+01, 1.523e+01, 9.080e+00, 1.138e+01,
7.360e+00, 8.110e+00, 9.330e+00, 1.532e+01, 7.890e+00, 8.470e+00,
8.030e+00, 9.560e+00, 8.500e+00, 1.036e+01, 9.930e+00, 9.100e+00,
9.230e+00, 6.390e+00, 8.450e+00, 8.480e+00, 9.680e+00, 6.490e+00,
7.170e+00, 1.313e+01, 9.850e+00, 5.850e+01, 2.795e+01, 1.462e+01,
6.690e+00, 8.370e+00, 6.630e+00, 7.880e+00, 7.850e+00, 7.300e+00,
7.020e+00, 7.220e+00, 7.290e+00, 1.037e+01, 9.210e+00, 6.470e+00,
1.212e+01, 6.530e+00, 1.121e+01, 6.520e+00, 9.070e+00, 8.940e+00,
1.200e+01, 6.820e+00, 8.730e+00, 9.400e+00, 1.017e+01, 1.125e+01,
7.160e+00, 1.333e+01, 1.056e+01, 7.060e+00, 9.000e+00, 1.031e+01,
1.400e+01, 7.630e+00, 1.211e+01, 1.184e+01, 1.114e+01, 1.116e+01,
9.780e+00, 1.342e+01, 1.023e+01, 7.860e+00, 1.171e+01, 9.730e+00,
1.091e+01, 1.054e+01]],
array([ 6,  2,  1,  4,  3,  5,  7, 13, 28, 11,  8,  9, 52,
       18, 15, 19, 10, 39, 26, 29, 12, 21, 96, 14, 34, 43,
      121, 37, 49, 31, 91, 16, 87, 33, 23, 50, 20, 25, 232,
       17, 47, 103, 65, 30, 27, 327, 32]),
array([365, 355, 194,  0, 129, 220, 188,  6, 39, 314, 333, 46, 321,
       12, 21, 249, 347, 364, 304, 233, 85, 75, 311, 67, 255, 284,
      359, 269, 340, 22, 96, 345, 273, 309, 95, 215, 265, 192, 251,
      302, 140, 234, 257, 30, 301, 294, 320, 154, 263, 180, 231, 297,
      292, 191, 72, 362, 336, 116, 88, 224, 322, 324, 132, 295, 238,
      209, 328, 38, 7, 272, 26, 288, 317, 207, 185, 158, 9, 198,
      219, 342, 312, 243, 152, 137, 222, 346, 208, 279, 250, 164, 298,
      260, 107, 199, 299, 20, 318, 216, 245, 189, 307, 310, 213, 278,
       16, 178, 275, 163, 34, 280, 1, 170, 214, 248, 262, 339, 10,
      290, 230, 53, 126, 3, 37, 353, 177, 246, 225, 18, 343, 326,
      162, 240, 363, 247, 323, 125, 91, 286, 60, 58, 351, 201, 232,
      258, 341, 244, 329, 253, 348, 2, 56, 68, 360, 76, 15, 226,
      349, 11, 316, 281, 287, 14, 86, 261, 331, 51, 254, 103, 42,
      325, 35, 203, 5, 276, 102, 71, 78, 8, 182, 79, 49, 156,
      200, 106, 135, 81, 142, 179, 52, 237, 204, 181, 296, 335, 282,
      274, 98, 157, 174, 223, 361, 283, 315, 36, 271, 139, 193, 136,
      277, 221, 264, 236, 89, 23, 218, 235, 119, 350, 161, 259, 27,
      167, 358, 59, 337, 43, 25, 127, 303, 115, 268, 44, 65, 252,
       64, 111, 90, 338, 31, 241, 285, 183, 84, 166, 28, 83, 305,
      356, 308, 229, 210, 153, 332, 120, 313, 69, 293, 4, 300, 40,
      117, 206, 144, 354, 41, 270, 306, 33, 50, 80, 97, 118, 134,
       17, 289, 121, 205, 74, 62, 29, 109, 168, 146, 242, 352, 155,
      291, 266, 101, 190, 327, 217, 171, 110, 87, 202, 70, 147, 169,
      212, 122, 330, 54, 196, 57, 73, 149, 239, 63, 195, 47, 319,
       19, 112, 344, 77, 160, 141, 13, 24, 150, 128, 176, 357, 211,
      172, 256, 165, 32, 105, 267, 148, 93, 45, 175, 159, 48, 100,
      184, 114, 133, 186, 334, 94, 151, 228, 113, 55, 66, 173, 104,
      197, 99, 131, 143, 124, 130, 187, 145, 108, 123, 92, 61, 138,
      227, 82]])

```

✓ Understanding Dataset Variables

```
1 # Dataset Describe
2 df.describe()
```



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

```
1 # Dataset Columns
2 df.columns
```

```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365'],
      dtype='object')
```

By basic inspection, a particular property name will have one particular host name hosted by that same individual but a particular host name can have multiple properties in an area. So, host_name is a categorical variable here. Also neighbourhood_group (comprising of Manhattan, Brooklyn, Queens, Bronx, Staten Island), neighbourhood and room_type (private, shared, Entire home/apt) fall into this category.

While id, latitude, longitude, price, minimum_nights, number_of_reviews, last_review, reviews_per_month, calculated_host_listings_count, availability_365; all of these columns fall under numerical variables

✓ Data Wrangling

✓ We are creating a separate copy of our original dataset.

```
1 # making a copy of dataframe.
2 airbnb = df.copy()
```

✓ We are replacing all NaN values in "review_per_month_ with 0 and host_name with Not Mentioned.

```
1 # filling na values
2 airbnb.fillna({'reviews_per_month':0},inplace=True)
3 airbnb.fillna({'host_name':'Not Mentioned'}, inplace = True)
```

✓ We have 3 columns that are not significant for our future data exploration. Now, we drop unnecessary columns such as id, name, last_review.

```
1 #dropping columns that are not significant for our future data exploration
2 airbnb.drop(['id','last_review','name'],axis=1,inplace=True)
```


✓ We are renaming some columns to ease the understanding of the dataset.

```
1 # renaming some column name for better understanding
2 airbnb.rename(columns= {'neighbourhood_group': 'Location', 'neighbourhood': 'Area'},inplace=True)
```

✓ Now let us check the updated dataset.

```
1 #check the dataset now
```

```
2  airbnb
```



	host_id	host_name	Location	Area	latitude	longitude	room_type	pr
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	
...
48890	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	
48891	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	
48892	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	
48893	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	
48894	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	

48895 rows × 13 columns


Next steps:

[Generate code with airbnb](#)




[View recommended plots](#)

```
1  airbnb.describe()
```



	host_id	latitude	longitude	price	minimum_nights	number_of_reviews
count	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48
mean	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	
std	7.861097e+07	0.054530	0.046157	240.154170	20.510550	
min	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	
25%	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	
50%	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	
75%	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	
max	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	

```
1  #checking our all new data frame info
2  airbnb.info()
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   host_id                               48895 non-null  int64
1   host_name                             48895 non-null  object
2   Location                               48895 non-null  object
3   Area                                   48895 non-null  object
4   latitude                               48895 non-null  float64
5   longitude                              48895 non-null  float64
6   room_type                             48895 non-null  object
7   price                                  48895 non-null  int64
8   minimum_nights                        48895 non-null  int64
9   number_of_reviews                     48895 non-null  int64
10  reviews_per_month                     48895 non-null  float64
11  calculated_host_listings_count        48895 non-null  int64
12  availability_365                       48895 non-null  int64
dtypes: float64(3), int64(6), object(4)
memory usage: 4.8+ MB
```

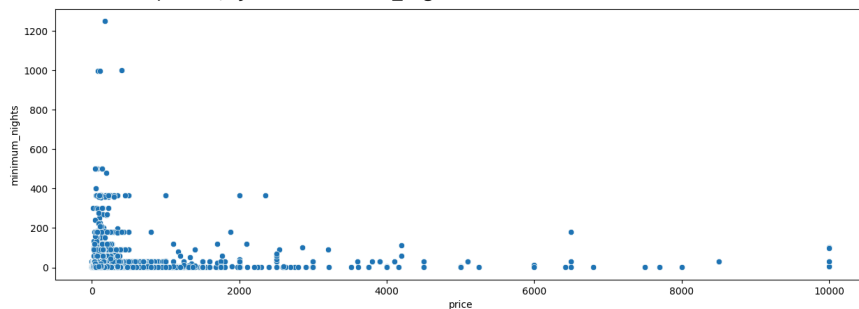
Now let's check the relationship between numerical variables


```

1 #set up figure size
2 plt.figure(figsize=(15,5))
3 # plot scatterplot
4 sns.scatterplot( y='minimum_nights', x='price', data=airbnb)

```

<Axes: xlabel='price', ylabel='minimum_nights'>



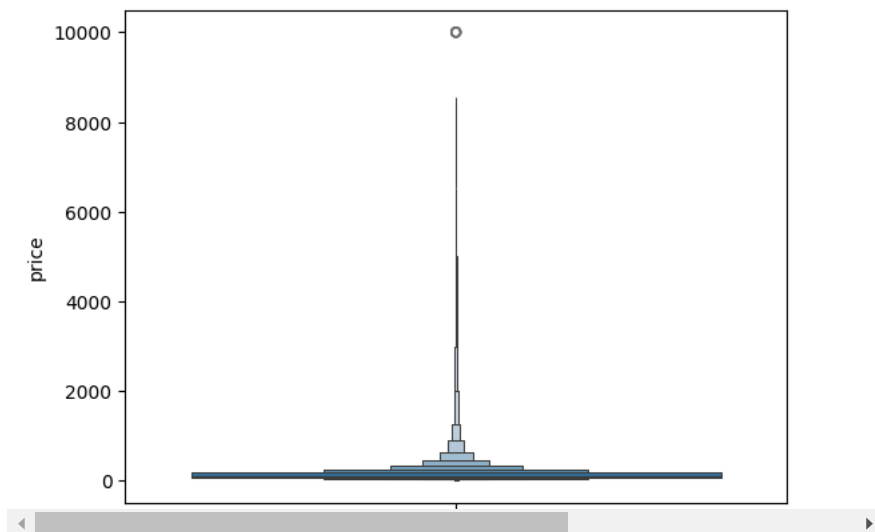
There's an interesting lookout from the above scatter plot, what do you see? Many data points are clustered on 0 price range, few have min nights for stay but price is 0, which looks like anomaly in price. Let's see the boxplot of this price column to have a feel of the presence of outliers. Don't worry, I'll be handling these outlier values!

```

1 sns.boxenplot(airbnb['price'],orient='horizontal')
2
3 plt.show()

```

/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:1601: UserWarning: Horizontal warnings.warn(single_var_warning.format("Horizontal", "y"))



```
1 airbnb['price'].describe()
```

```

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

```

✓ Removing outliers from price column

```

1 # Select the column where you want to remove outliers
2 column_name = 'price'
3
4 # Calculate the IQR and define the threshold
5 q1 = airbnb[column_name].quantile(0.25)
6 q3 = airbnb[column_name].quantile(0.75)
7 iqr = q3 - q1
8 threshold = 1.5
9
10 # Remove the outliers
11 filtered_airbnb = airbnb[(airbnb[column_name] >= q1 - threshold*iqr) & (airbnb[column_name] <= q3 + threshold*iqr)]
12

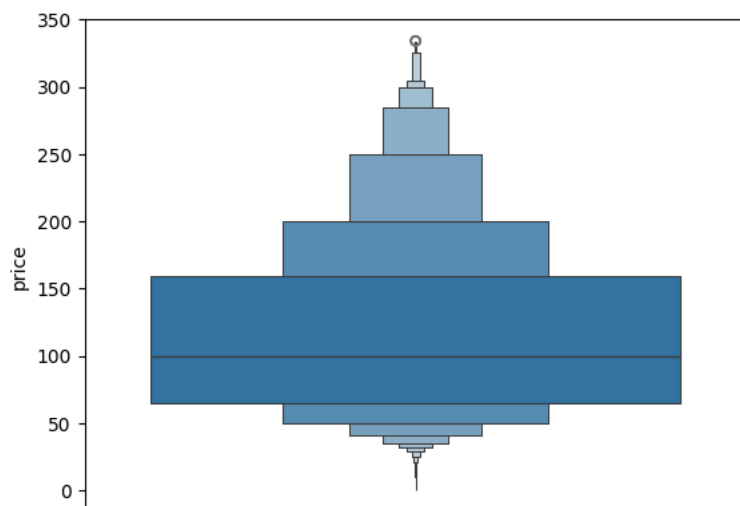
```

```

1 #Now, let's check the boxen plot once again as by now I have removed the outliers.
2 sns.boxenplot(filtered_airbnb['price'],orient='horizontal')

```

⚠ /usr/local/lib/python3.10/dist-packages/seaborn/_base.py:1601: UserWarning: Horizontal warnings.warn(single_var_warning.format("Horizontal", "y"))
<Axes: ylabel='price'>



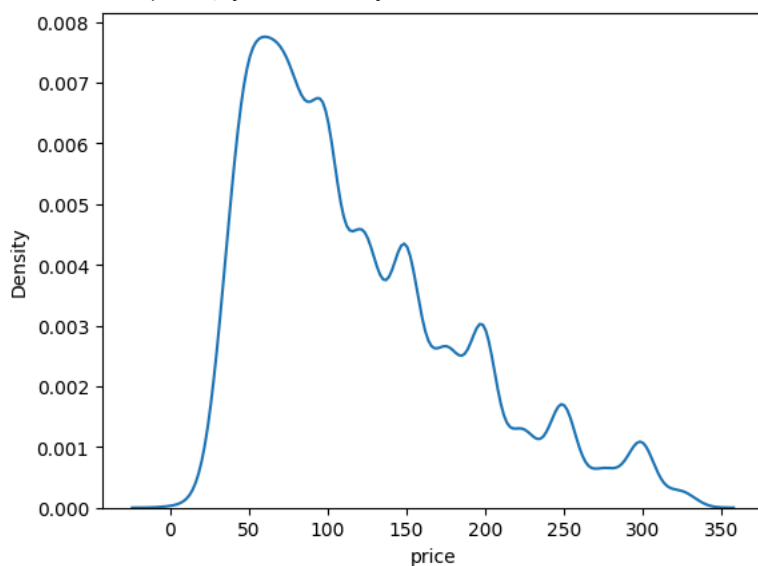
✓ **Let's check the distribution of price after removing of outliers.**

```

1 # plot kdeplot
2 sns.kdeplot(filtered_airbnb['price'])

```

⚠ <Axes: xlabel='price', ylabel='Density'>



✓ **Now let's add one more column that will define the price range from price column of the dataset.**

```

1 # Lets define a new column named Price range
2
3 def get_price_range(pr):
4     if pr >= 0 and pr < 50:
5         price_range = "0-50"
6     elif pr >= 50 and pr < 100:
7         price_range = "50-100"
8     elif pr >= 100 and pr < 150:
9         price_range = "100-150"
10    elif pr >= 150 and pr < 200:
11        price_range = "150-200"
12    elif pr >= 200 and pr < 250:
13        price_range = "200-250"
14    elif pr >= 250 and pr < 300:
15        price_range = "250-300"
16    else:
17        price_range = ">300"
18    return price_range

```

```

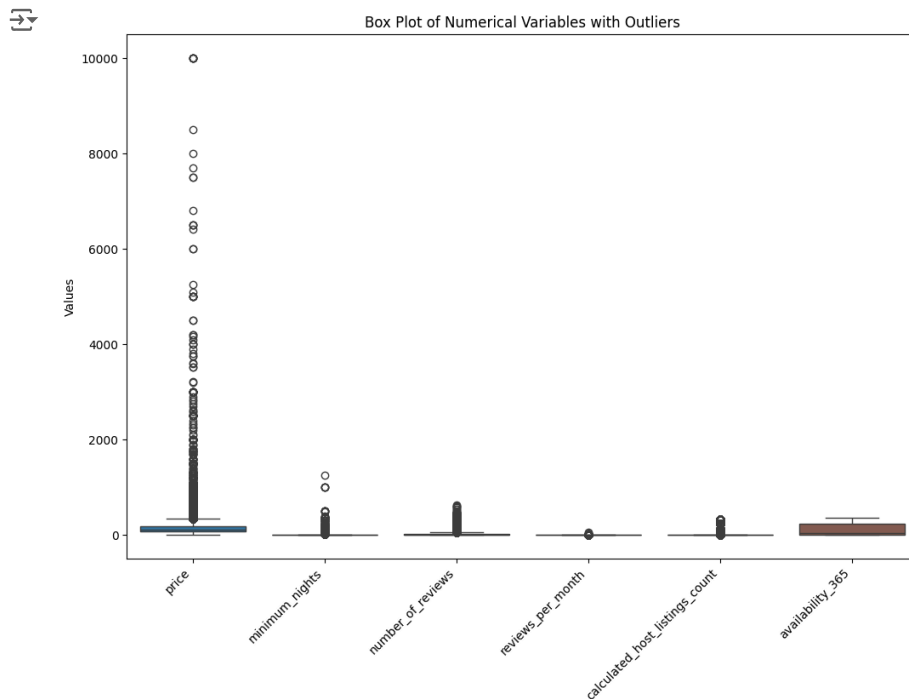
1 # Lets add a new column named Price range
2 airbnb['price_range'] = airbnb.apply(lambda x: get_price_range(x['price']), axis=1)

```

```

1 # Selecting the numerical columns for the box plot
2 numerical_columns = ['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availability_365']
3
4 # Creating the box plot with outliers shown
5 plt.figure(figsize=(12, 8))
6 sns.boxplot(data=df[numerical_columns], showfliers=True) # Set showfliers=True to show outliers
7 plt.title('Box Plot of Numerical Variables with Outliers')
8 plt.ylabel('Values')
9 plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better visibility
10 plt.show()

```



```

1 #check the new column
2 airbnb.head(5)

```



	host_id	host_name	Location	Area	latitude	longitude	room_type	price
0	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149
1	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225
2	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150
3	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89
4	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80

Next steps:

[Generate code with airbnb](#)[View recommended plots](#)

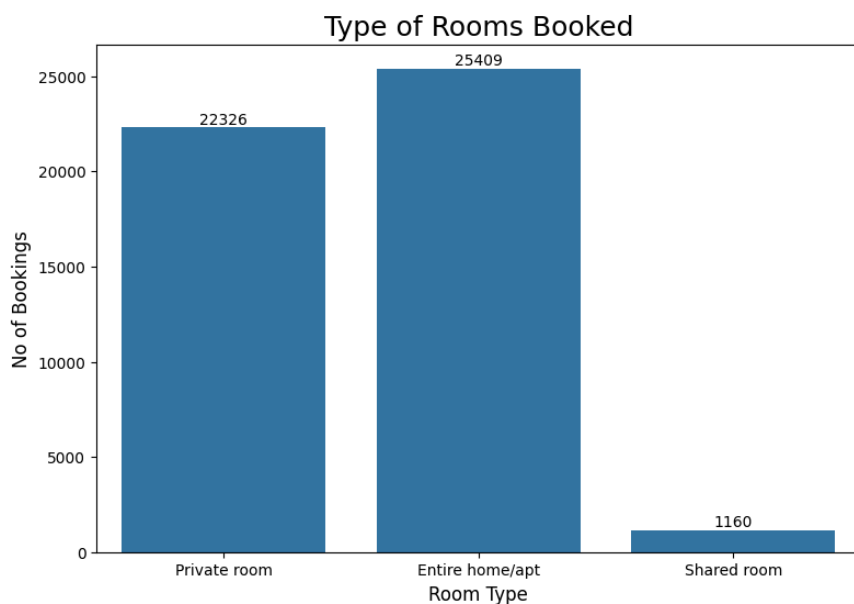
▼ Data Analysis on Airbnb Dataset

▼ 1.Top room category in the Airbnb ?

```

1 # Chart-1: Plot of distribution of types of rooms booked.
2
3 # set figure size
4 plt.figure(figsize=(9,6))
5
6 # set up plot title
7 plt.title('Type of Rooms Booked', {'fontsize': 18, 'fontweight': 20})
8
9 # plot count plot
10 dtr = sns.countplot(x='room_type',data=airbnb)
11
12 # labels for each bar
13 for container in dtr.containers:
14     dtr.bar_label(container)
15
16 # set x-axis and y-axis labels
17 plt.xlabel("Room Type", {'fontsize': 12, 'fontweight': 15})
18 plt.ylabel("No of Bookings", {'fontsize': 12, 'fontweight': 15})
19
20 # display figure
21 plt.show()

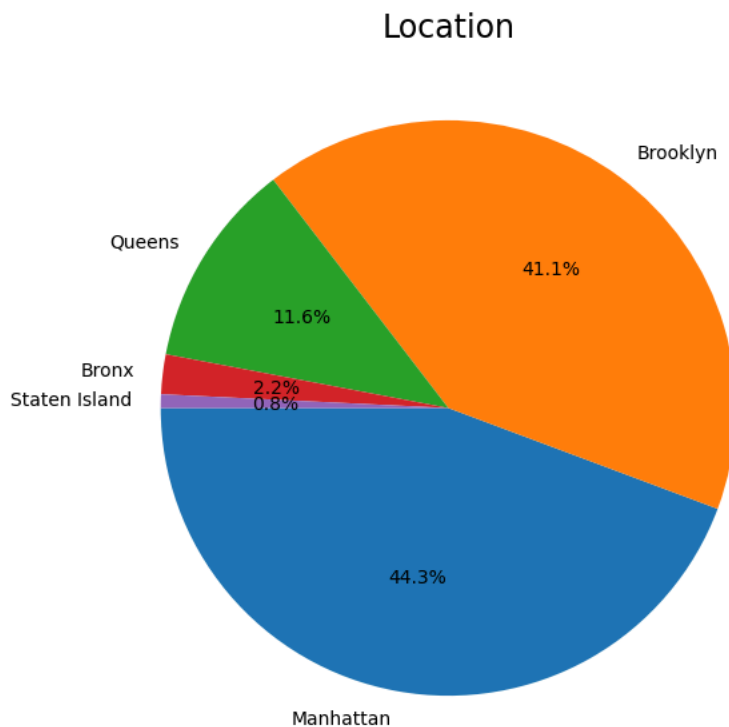
```



From the above Bar Chart's output we can say that most of the room booked by the guest on airbnb are either **Entire home/apartments** or **private rooms** and **Shared rooms** are least preferable.

✓ 2 . Which location has highest share of bookings?

```
1 # Chart - 2: Let us take a look at total property according to Location.
2
3 # set figure size
4 plt.figure(figsize=(13,7))
5
6 # set the pie chart title
7 plt.title("Location",{'fontsize': 17, 'fontweight': 25})
8
9 # plot pie chart
10 plt.pie(airbnb["Location"].value_counts(), labels=airbnb.Location.value_counts().index, autopct='%1.1f%%', startangle=180)
11
12 # display figure
13 plt.show()
```



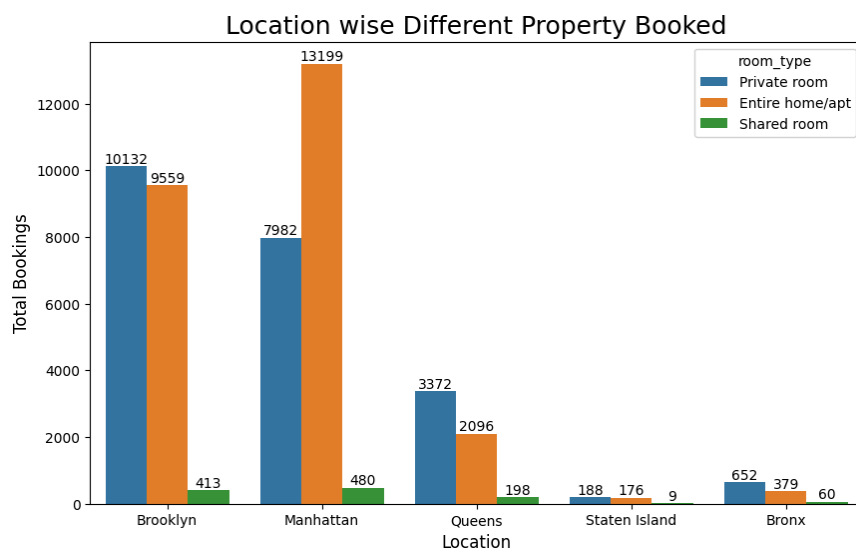
The pie chart above shows that Airbnb Listings are near **Manhattan**, and **Brooklyn** has the highest share of booking. Around **85%** bookings are between **Manhattan and Brooklyn** and rests **15%** shares between the rest of the left Locations. It means **Manhattan and Brooklyn** are the most popular location between guests.

✓ 3. Now let's look at the different types of properties that are booked and show their Locationwise distribution.

```

1 # Chart - 3: Locationwise different property Booked.
2
3 # set figure size
4 plt.figure(figsize=(10,6))
5
6 # set plot title
7 plt.title('Location wise Different Property Booked',{'fontsize': 18, 'fontweight': 20})
8
9 # plot count plot
10 lwb = sns.countplot(x='Location',hue= 'room_type', data=airbnb)
11
12 # labels for each bar
13 for container in lwb.containers:
14     lwb.bar_label(container)
15
16 # set x-axis and y-axis labels
17 plt.ylabel("Total Bookings", {'fontsize': 12, 'fontweight': 15})
18 plt.xlabel('Location', {'fontsize': 12, 'fontweight': 15})
19
20 # display figure
21 plt.show()

```



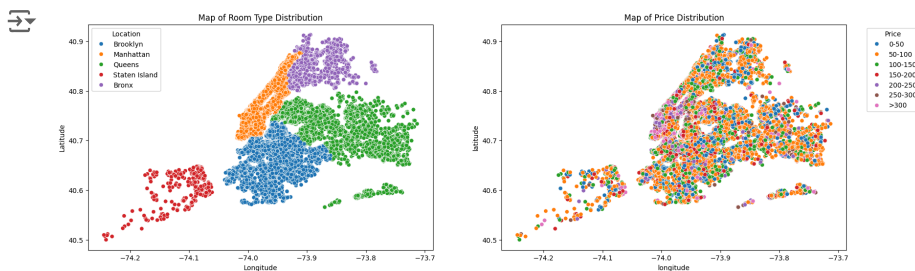
The graph shows that the **Entire Home/Apartment** is listed most near **Manhattan**, while Private Rooms and Apartments in **Brooklyn** are nearly equal.

✓ 4 . Which location has the most expensive AIRBNB bookings?

```

1 # Chart - 4: Plot between Map of Property Type Distribution vs Map of Price Distribution
2
3 # create a figure with two subplots
4 fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))
5
6 # plot first subplot
7 sns.scatterplot(ax=axs[0], x='longitude', y='latitude', hue='Location', data=airbnb)
8
9 # set the title of the first subplot
10 axs[0].set_title("Map of Room Type Distribution", {'fontsize': 12, 'fontweight': 15})
11
12 # set x-axis and y-axis labels of the first subplot
13 axs[0].set_xlabel('Longitude', {'fontsize': 10, 'fontweight': 12})
14 axs[0].set_ylabel('Latitude', {'fontsize': 10, 'fontweight': 12})
15
16
17 # set the order
18 desired_order = ['0-50', '50-100', '100-150', '150-200', '200-250', '250-300', '>300']
19
20 # plot second subplot
21 sns.scatterplot(ax=axs[1], data=airbnb, x="longitude", y="latitude", hue="price_range", hue_order = desired_order)
22
23 # set x-axis and y-axis labels of the second subplot
24 axs[1].legend(title="Price", bbox_to_anchor=(1.05, 1), loc=2)
25 axs[1].set_title("Map of Price Distribution", {'fontsize': 12, 'fontweight': 15})
26
27 # set title
28 axs[0].set_xlabel('Longitude', {'fontsize': 10, 'fontweight': 12})
29 axs[0].set_ylabel('Latitude', {'fontsize': 10, 'fontweight': 12})
30
31 # display the figure
32 plt.show()

```



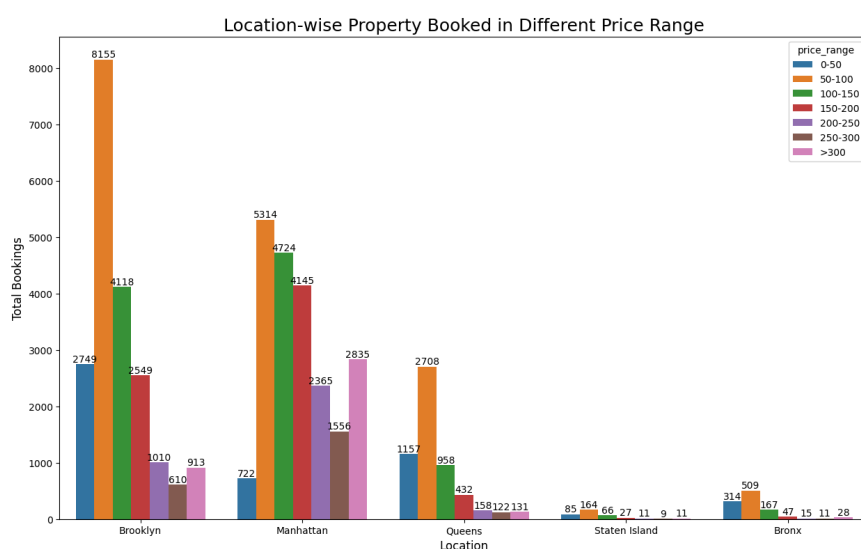
The region that stands out as the most expensive is **Manhattan**. The high cost of accommodations in this region significantly impacts the booking trends. Despite its expensiveness, **Manhattan** remains a popular choice for bookings, and also from the previous visualization we find that the **Entire Home/Apartment** is listed mostly near **Manhattan** means most guest preferred to stay in independent property.

✓ **5. The next challenge is to find out locationwise total bookings in different price range.**

```

1 # Chart - 5: Location-wise Distribution of Property in Different Price Range.
2
3 # set figure size
4 plt.figure(figsize=(15,9))
5
6 # set plot title
7 plt.title('Location-wise Property Booked in Different Price Range',{'fontsize': 18, 'fontweight': 20})
8
9 # set the order
10 desired_order = ['0-50', '50-100', '100-150', '150-200', '200-250', '250-300', '>300']
11
12 # plot countplot
13 ldr = sns.countplot(x='Location', hue= 'price_range',width = 0.8, data=airbnb, hue_order = desired_order)
14
15 # labelling of each bars in the plot
16 for container in ldr.containers:
17     ldr.bar_label(container)
18
19 # set x-axis and y-axis labels
20 plt.ylabel("Total Bookings", {'fontsize': 12, 'fontweight': 15})
21 plt.xlabel('Location', {'fontsize': 12, 'fontweight': 15})
22
23 # display figure
24 plt.show()

```



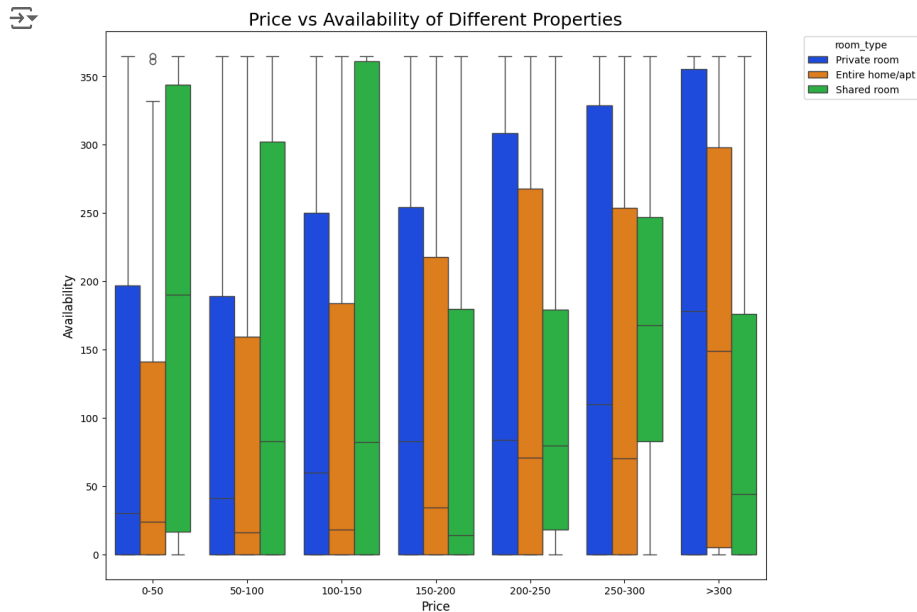
We can see that in all the locations, the properties within the price range of 50-100 dollars are the most booked properties followed by the properties in 0-50 dollars price range. But in case of **Manhattan** we can see that properties in higher price range are almost as popular as the 50-100 dollar range. So, this further concludes our previous scatterplot statement of Manhattan being the most expensive location.

✓ 6. Pricewise Room Availability Plot of Different Properties.


```

1 # Chart - 6: Pricewise Room Availability Plot of Different Properties
2
3 # set figure size
4 plt.figure(figsize=(12,10))
5
6 # set plot title
7 plt.title('Price vs Availability of Different Properties', {'fontsize': 18, 'fontweight': 20})
8
9 # set the order
10 desired_order = ['0-50', '50-100', '100-150', '150-200', '200-250', '250-300', '>300']
11
12 # plot the boxplot
13 sns.boxplot(x='price_range', y='availability_365', hue='room_type', palette='bright', data=airbnb, order = desired_order)
14
15 # set x-axis and y-axis labels
16 plt.xlabel('Price', {'fontsize': 12, 'fontweight': 15})
17 plt.ylabel('Availability', {'fontsize': 12, 'fontweight': 15})
18 plt.legend(title="room_type", bbox_to_anchor=(1.05, 1), loc=2)
19 # display figure
20 plt.show()

```



From the boxplot we can see that in the lower price point mostly the shared rooms are available throughout the 365 days while the Private Rooms and Entire Home/Apt are mostly available for about 200 days.

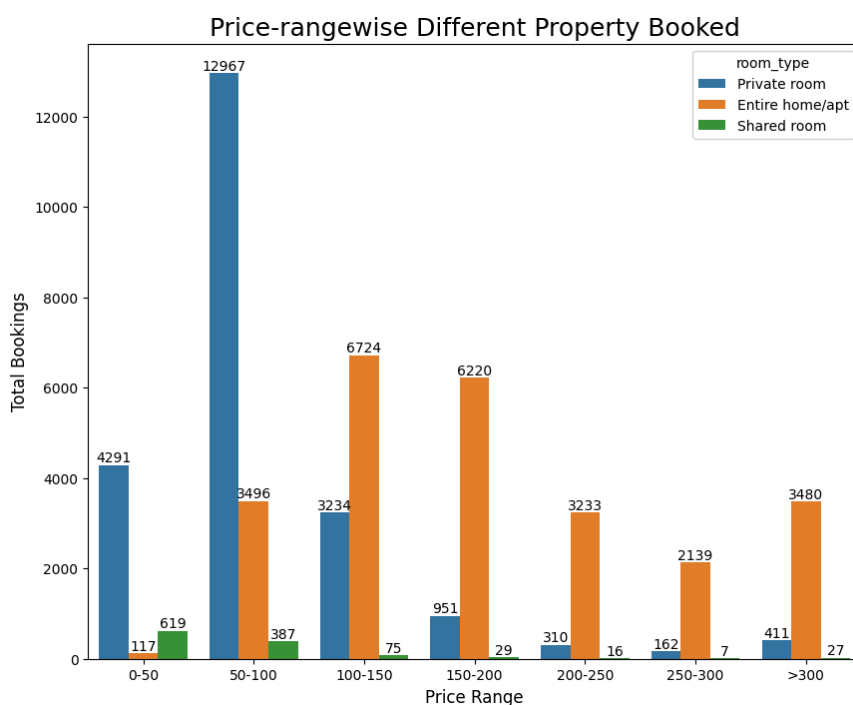
As the price range goes up the availability of shared room type decreases while the private rooms and entire room/apartment availability increases.

✓ 7. Now let's look at the distribution of properties in different price range.

```

1 # Chart - 7: Total Properties Booked in Different Price Range.
2
3 # set the figure size
4 plt.figure(figsize=(10,8))
5
6 # set plot title
7 plt.title('Price-rangewise Different Property Booked',{'fontsize': 18, 'fontweight': 20})
8
9 # set the order
10 desired_order = ['0-50', '50-100', '100-150', '150-200', '200-250', '250-300', '>300']
11
12 # plot count plot
13 tb = sns.countplot(x='price_range', hue= 'room_type', data=airbnb, order =desired_order)
14
15 # set x-axis and y-axis labels
16 plt.ylabel("Total Bookings", {'fontsize': 12, 'fontweight': 15})
17 plt.xlabel('Price Range', {'fontsize': 12, 'fontweight': 15})
18
19 # labelling of each bars in the plot
20 for container in tb.containers:
21     tb.bar_label(container)
22
23 # display figure
24 plt.show()

```



For private rooms, within the 50-100 dollars price range we observe that they are booked the most and the number of bookings is highest (12967) among all the categories. The number of bookings in 0-50 dollars (4291) and 100-150 dollars (3496) price range are also having a high number of bookings but in the higher price range (more than 150 dollars range) the number of bookings for private room type decreases significantly.

We can conclude that private rooms are relatively affordable mode of accommodation and also, they also provide privacy so we see most of the booking in 0-150 dollars price range.

For the Entire home/apt type of property, we see relatively lesser number of bookings in the lowest price range of 0-50 dollars (117) range while the number of bookings in the rest of the price range is almost uniform and the no of bookings maximizes in 100-150 and 150-200 dollars range (both has over 6000 bookings).

From this distribution data it is evident that the entire home/apartments are the most luxurious type of accommodation so their bookings stand in comparatively higher price point as compared to others.

From the property distribution chart according to price range we can see that out of all the different room types, the shared rooms are mostly abundant in lower price range. The number of shared rooms within 0-50 price range is highest (619) and it is almost half (387) in 50-100 price range and the total number of properties booked in more than 100 dollars price range is less than 150.

So, we can conclude that guests usually don't like to spend much money in a shared type of accommodation as it is viewed as most affordable option of staying.

✓ 8. Let's find out the average price of Rooms depending on room type.

```
1 # Chart - 8: Room Type-wise Average Price of Rooms Booked in Each Neighbourhood
2
3 # create a new dataset
4 mean_price = airbnb.groupby(['Location', 'room_type'])['price'].mean().unstack().round(1)
5
6 # plot the dataset
7
8 mp= mean_price.plot(kind = 'bar', figsize= (10,6))
9
10 # set plot title
11 plt.title('Average price of Room Booked in Each Location', {'fontsize': 18, 'fontweight': 20})
12
13 # labels for each bar
14 for container in mp.containers:
15     mp.bar_label(container)
16
17 # set x-axis and y-axis labels
18 plt.xlabel('Location', {'fontsize': 12, 'fontweight': 15})
19 plt.ylabel('avg_Price', {'fontsize': 12, 'fontweight': 15})
20
21 # display figure
22 plt.show()
```



As predicted in the previous queries we can see that the average price of the entire home/ apartments is the highest and it is significantly higher than the other two options.

While the average price of private rooms is higher than share room types, the difference between the two is very low. So, we see a large number of bookings in the private room types as compared to the shared room types as in private rooms, the guests have the added benefit of privacy in almost similar prices.

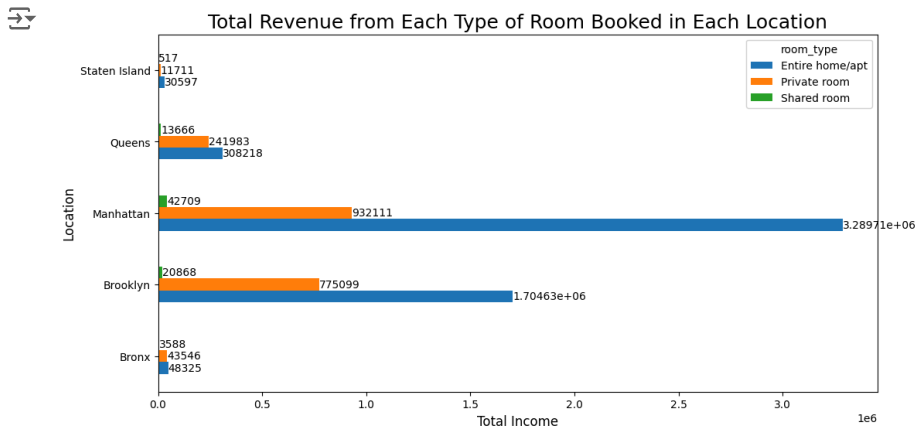
From this bar plot we can also see that in all room types the **Manhattan** area is much more expensive as compared to the other locations, while in the rest of the locations the average price of rooms is almost similar. This further backs up our previous statement of **Manhattan** being the most expensive location.

✓ 9. What are the total revenues from each type of room booked in different neighbourhood area??

```

1 # Chart - 9: Room-type-wise Total price for each Location.
2
3 # define a new dataset for total revenue
4 total_revenue = airbnb.groupby(['Location', 'room_type'])['price'].sum().unstack()
5
6 # plot the dataset
7 tr = total_revenue.plot(kind = 'barh', figsize= (12,6))
8
9 # set plot title
10 plt.title('Total Revenue from Each Type of Room Booked in Each Location', {'fontsize': 18, 'fontweight': 20})
11
12 # set x-axis and y-axis labels
13 plt.xlabel('Total Income', {'fontsize': 12, 'fontweight': 15})
14 plt.ylabel('Location', {'fontsize': 12, 'fontweight': 15})
15
16 # labels for each bar
17 for container in tr.containers:
18     tr.bar_label(container)
19
20 # display figure
21 plt.show()

```



In terms of revenue generated from AIRBNB, we can see that the **Manhattan** location generated the most amount of money as already we have seen in problem number 3 and problem number 4, that **Manhattan** sees the most number of bookings and the price range of bookings is higher at this location.

Brooklyn is the second most revenue generating location and **Queens** is third in the list and as the trend of number of bookings indicates, the revenue generated from **Bronx** and **Staten Island** locations are the lowest.

✓ 10. Let's find out the most popular properties.

```

1 # Chart - 10: Properties receiving highest reviews.
2 highest_reviews = df.sort_values(by='number_of_reviews', ascending=False)
3 highest_reviews.head()

```



	id	name	host_id	host_name	neighbourhood_group	neighbourhood
11759	9145202	Room near JFK Queen Bed	47621202	Dona	Queens	Jamaica
2031	903972	Great Bedroom in Manhattan	4734398	Jj	Manhattan	Harlem
2030	903947	Beautiful Bedroom in Manhattan	4734398	Jj	Manhattan	Harlem
2015	891117	Private Bedroom in Manhattan	4734398	Jj	Manhattan	Harlem
13495	10101135	Room Near JFK Twin Beds	47621202	Dona	Queens	Jamaica

Next steps:

[Generate code with highest_reviews](#)[View recommended plots](#)

The above table shows the top 5 properties which have received the highest number of reviews. Out of the 5, three properties are from the Manhattan neighborhood group in Harlem.

The top property which has received the most reviews is from Queens in the neighborhood of Jamaica. The property having 5th highest review is also from Queens.

The price is also almost similar of all the properties, approximately 50 Dollars.

They all offer minimum 1 night's stay which most of the people prefer as it is very flexible.

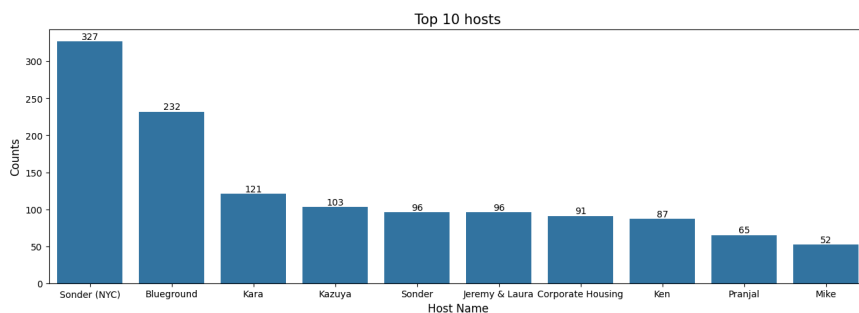
The availability of the rooms is also high with the top 4 having an availability rate of approximately 300 days or more.

11 . Finding the host with highest no of property listed.

```

1 # Chart - 11: Host with the highest number of listed properties booked.
2
3 # let's make a new dataset to see who has the most listings and in which neighbourhood.
4 top_hosts = airbnb.groupby(['host_name'])['calculated_host_listings_count'].max().reset_index().sort_values('calculated_host_listings_count', ascending=False)
5
6 # rename column name calculated_host_listing_count to counts
7 top_hosts.rename(columns={'calculated_host_listings_count': 'counts'}, inplace=True)
8
9 # set figure size
10 plt.figure(figsize=(16,5))
11
12 # set plot title
13 plt.title('Top 10 hosts',{'fontsize': 15, 'fontweight': 30})
14
15 # plot the bar chart
16 th = sns.barplot(x='host_name',y='counts',data=top_hosts)
17
18 # labels for each bar
19 for container in th.containers:
20     th.bar_label(container)
21
22 # set x-axis and y-axis labels
23 plt.ylabel("Counts", {'fontsize': 12, 'fontweight': 15})
24 plt.xlabel('Host Name', {'fontsize': 12, 'fontweight': 15})
25
26 #display figure
27 plt.show()

```



We can see that Sonder (NYC) has the highest number of properties that are listed but his property was not in the top 5 highest reviews table we saw earlier.

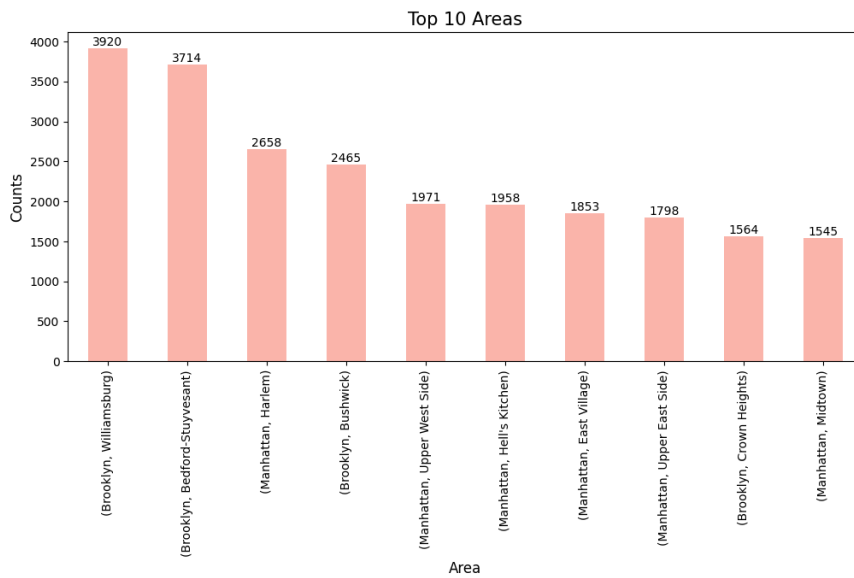
Having maximum properties listed on the Airbnb does not mean that you will have will maximum customers also.

✓ 12. Let's find top 10 areas with most numbers of visitors.

```

1 # Chart - 12: Top 5 areas with most number of bookings.
2
3 # let's create a new dataset which will define the top 10 areas with most number of bookings.
4 top_10_areas = airbnb.groupby(['Location', 'Area'])['Area'].count().sort_values(ascending= False).head(10)
5
6 # set figure size
7 plt.figure(figsize=(12,5))
8
9 # set plot title
10 plt.title('Top 10 Areas',{'fontsize': 15, 'fontweight': 30})
11
12 # plot the bar chat
13 ta = top_10_areas.plot(kind = 'bar', cmap = 'Pastel1')
14
15 # labels for each bar
16 for container in ta.containers:
17     ta.bar_label(container)
18
19 # set x-axis and y-axis labels
20 plt.ylabel("Counts", {'fontsize': 12, 'fontweight': 15})
21 plt.xlabel('Area', {'fontsize': 12, 'fontweight': 15})
22
23 #display figure
24 plt.show()

```



Though we see the most numbers of booking are in **Manhattan** location but the top two of most booked neighbourhoods area seems to be from **Brooklyn** location, though six of the remaining eight areas are from **Manhattan**.

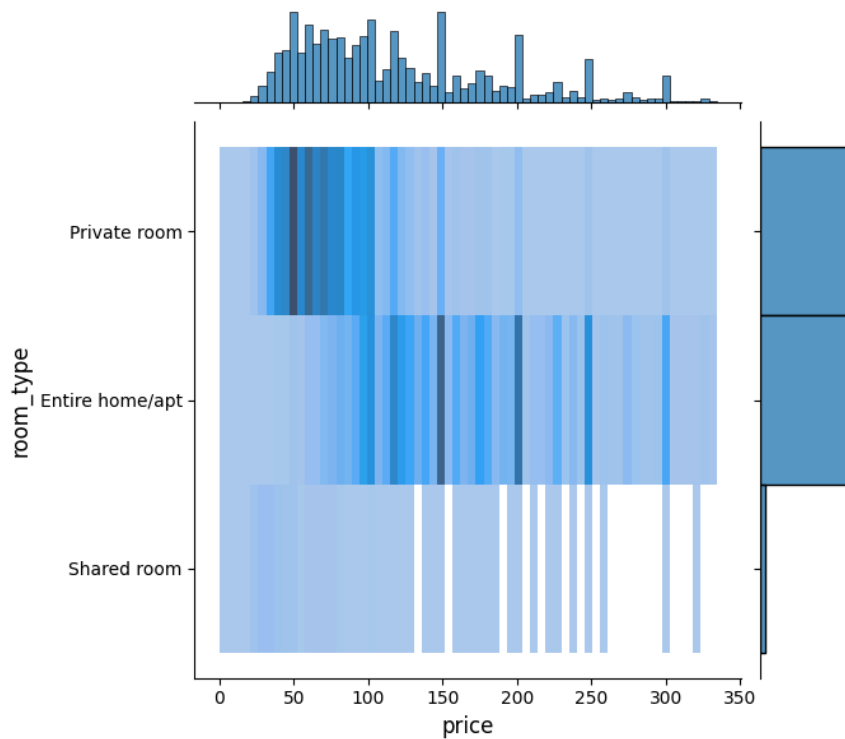
✓ 13 . Create a graph for analyzing the sale of a particular room type according to the price range fluctuation.

```

1 # Chart - 13: Chart for Sale of a particular room type according to the price range.
2 # set figure size
3 plt.figure(figsize=(10,4))
4
5 # plot jointplot
6 sns.jointplot(x='price', y='room_type',kind="hist", data=filtered_airbnb)
7
8 # set x-axis and y-axis labels
9 plt.xlabel('price',{'fontsize': 12, 'fontweight': 15})
10 plt.ylabel('room_type',{'fontsize': 12, 'fontweight': 15})
11
12 #display figure
13 plt.show()
14

```

<Figure size 1000x400 with 0 Axes>



The above plot demonstrates the booking pattern of different room types with respect to their price range. Private rooms are booked mostly in the price range of 25 to 80 dollars. The entire home/apt type is highly booked in the price range of 120 to 175 dollars.

✓ 14. Correlation HEATMAP.

```
1 # Chart - 14: Correlation Heatmap.
2
3 # create a separate dataset for heatmap.
4 corr_airbnb = airbnb[['price', 'minimum_nights', 'calculated_host_listings_count', 'availability_365']]
5
6 #plot heatmap
7 sns.heatmap(corr_airbnb.corr(), cmap='Paired', annot=True)
8
9 #show figure
10 plt.show()
```