Project Name Airbnb Bookings Analysis

```
# This is formatted as code
```

Project Type - EDA

Contribution - Team

Team Member 1 - Sandesh Salunke

Team Member 2 - Irfan Momin

Team Member 3 - Sushil Ghodvinde

Team Member 4 - Rushikesh Pingle

Project Summary -

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.

GitHub Link -

https://github.com/Sandy1386/Sandesh-Salunke.git

Problem Statement

**

1) Which hosts are having heighest number of appartments ?

2) Which are the top 10 neighbourhood which are having maximum number of appartments for airbnb in the respective neighbourhood ?

3) What are the neighbourhood in each group which are having maximum prices in thier rspective neighbourhood_group ?

4) How neighborhood is realted with reviews ?

5) What can we learn from predictions? (ex: locations, prices, reviews, etc)

6) What is the distribution of the room type and its distribution over the location ?

7) How does the Room_type is distributed over Neighbourhood_Group are the ratios of respective room_types more or less same over each neighbourhood_group ?

8) How the price column is distributed over room_type and are there any Surprising items in price column ?

9) Which are the top 5 hosts that have obatained heighest no. of reviews ?

10) What is the average preferred price by customers according to the neighbourhood_group for each category of Room_type?

Define Your Business Objective?

We believe that Airbnb can be more than a marketplace that merely connects guests to Hosts. There goal are to provide the ultimate service for guests, anticipating their needs and going above and beyond—just like a good Host.

General Guidelines : -

1. Well-structured, formatted, and commented code is required.
2. Exception Handling, Production Grade Code & Deployment Ready Code will be a plus. Those students will be awarded some additional credits.

   The additional credits will have advantages over other students during Star Student selection.

   ```
   [ Note: - Deployment Ready Code is defined as, the whole .ipynb notebook should be executable in one go
             without a single error logged. ]
   ```

3. Each and every logic should have proper comments.
4. You may add as many number of charts you want. Make Sure for each and every chart the following format should be answered.

```
# Chart visualization code
```

- Why did you pick the specific chart?
- What is/are the insight(s) found from the chart?
- Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

5. You have to create at least 20 logical & meaningful charts having important insights.

[Hints : - Do the Vizualization in a structured way while following "UBM" Rule.

U - Univariate Analysis,

B - Bivariate Analysis (Numerical - Categorical, Numerical - Numerical, Categorical - Categorical)

M - Multivariate Analysis]

Let's Begin !

Dataset Loading

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

# Import Libraries
import pandas as pd # import pandas opration
import numpy as np  # import numpy
from numpy  import mean



import seaborn as sns
from skimage.io import imread

import matplotlib.pyplot as plt # import matplotlib
sns.set()
%matplotlib inline

import statistics
from collections import Counter
from wordcloud import WordCloud, ImageColorGenerator
sns.set_theme(style="ticks", color_codes=True)
```

```
     Mounted at /content/drive
```

```
# Load Dataset

file_path ="/content/Airbnb NYC 2019.csv"
df_air = pd.read_csv(file_path)
```

Dataset First View

```
# Dataset First Look
df_air = pd.read_csv(file_path)
df_air.head(10)
```

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

```
df_air.isna().sum()
```

```
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
```

Dataset Rows & Columns count

```
# Dataset Rows & Columns count
# we have to check the data roe and colums
df_air.shape
```

```
(48895, 16)
```

Dataset Information

```
# Dataset Info
### Below are the information of data file####
```

```
df_air.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

```
# Dataset Duplicate Value Count
# Below entry is use to check duplicate entries.

df_air.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...  
48890    False
48891    False
48892    False
48893    False
48894    False
Length: 48895, dtype: bool
```

```
###  We can use another method for duplicated count##
df_air.duplicated().sum()
```

```
0
```

Missing Values/Null Values

```
# Missing Values/Null Values Count
df_air.isnull().sum()
```

```
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
```

```
# To check the missing value in name
df_air[df_air['name'].isnull()]
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nigh
2854	1615764	NaN	6676776	Peter	Manhattan	Battery Park City	40.71239	-74.01620	Entire home/apt	400	10
3703	2232600	NaN	11395220	Anna	Manhattan	East Village	40.73215	-73.98821	Entire home/apt	200	
5775	4209595	NaN	20700823	Jesse	Manhattan	Greenwich Village	40.73473	-73.99244	Entire home/apt	225	
5975	4370230	NaN	22686810	Michaël	Manhattan	Nolita	40.72046	-73.99550	Entire home/apt	215	
6269	4581788	NaN	21600904	Lucie	Brooklyn	Williamsburg	40.71370	-73.94378	Private room	150	
6567	4756856	NaN	1832442	Carolina	Brooklyn	Bushwick	40.70046	-73.92825	Private room	70	
6605	4774658	NaN	24625694	Josh	Manhattan	Washington Heights	40.85198	-73.93108	Private room	40	
8841	6782407	NaN	31147528	Huei-Yin	Brooklyn	Williamsburg	40.71354	-73.93882	Private room	45	
11963	9325951	NaN	33377685	Jonathan	Manhattan	Hell's Kitchen	40.76436	-73.98573	Entire home/apt	190	
12824	9787590	NaN	50448556	Miguel	Manhattan	Harlem	40.80316	-73.95189	Entire home/apt	300	
13059	9885866	NaN	37306329	Juliette	Manhattan	Chinatown	40.71632	-73.99328	Private room	67	
13401	10052289	NaN	49522403	Vanessa	Brooklyn	Brownsville	40.66409	-73.92314	Private room	50	
15819	12797684	NaN	69715276	Yan	Manhattan	Upper West Side	40.79843	-73.96404	Private room	100	
16071	12988898	NaN	71552588	Andrea	Bronx	Fordham	40.86032	-73.88493	Shared room	130	
18047	14135050	NaN	85288337	Jeff	Brooklyn	Bedford-Stuyvesant	40.69421	-73.93234	Private room	70	
									Entire		

```
# To check the missing host_id in name
df_air[df_air['host_name'].isnull()]
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	mini
360	100184	Bienvenue	526653	NaN	Queens	Queens Village	40.72413	-73.76133	Private room	50	
2700	1449546	Cozy Studio in Flatbush	7779204	NaN	Brooklyn	Flatbush	40.64965	-73.96154	Entire home/apt	100	
5745	4183989	SPRING in the City!! Zen-Style Tranquil Bedroom	919218	NaN	Manhattan	Harlem	40.80606	-73.95061	Private room	86	
6075	4446862	Charming Room in Prospect Heights!	23077718	NaN	Brooklyn	Crown Heights	40.67512	-73.96146	Private room	50	
6582	4763327	Luxurious, best location, spa inc'l	24576978	NaN	Brooklyn	Greenpoint	40.72035	-73.95355	Entire home/apt	195	
8163	6292866	Modern Quiet Gem Near All	32722063	NaN	Brooklyn	East Flatbush	40.65263	-73.93215	Entire home/apt	85	
8257	6360224	Sunny, Private room in Bushwick	33134899	NaN	Brooklyn	Bushwick	40.70146	-73.92792	Private room	37	
8852	6786181	R&S Modern Spacious Hideaway	32722063	NaN	Brooklyn	East Flatbush	40.64345	-73.93643	Entire home/apt	100	
9138	6992973	1 Bedroom in Prime Williamsburg	5162530	NaN	Brooklyn	Williamsburg	40.71838	-73.95630	Entire home/apt	145	
9817	7556587	Sunny Room in Harlem	39608626	NaN	Manhattan	Harlem	40.82929	-73.94182	Private room	28	
14040	10709846	Sunny, spacious room in Greenpoint	7822683	NaN	Brooklyn	Greenpoint	40.73539	-73.95838	Private room	55	
14631	11553543	Cozy Room Astoria	26138712	NaN	Queens	Ditmars Steinway	40.77587	-73.91775	Private room	45	
15174	12113879	Sunny, Large West Village 1 BR Near Everything	5300585	NaN	Manhattan	Chelsea	40.73949	-73.99801	Entire home/apt	220	
19565	15648096	Spacious 2 bedroom close to	100971588	NaN	Bronx	Highbridge	40.83844	-73.92489	Entire home/apt	75	

```
# Visualizing the missing values
missing = df_air.isnull().sum()
plt.figure(figsize=(15,8))
missing.plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd53318820>
```

What did you know about your dataset?

The above data is having 48895 rows and 16 coloumn. The data is having a lot of null an showing 0 duplicate values.

2. Understanding Your Variables

```
# Dataset Columns
# here we had check all columns
df_air.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')
```

```
# Dataset Describe
# here we had check all Describe
df_air.describe()
```

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

Variables Description

Answer Here

```
df_air[df_air['host_name']=='John']
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minim
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	
429	148201	NYC - Sunny Greenwich Village 1br	715807	John	Manhattan	Greenwich Village	40.72831	-74.00177	Entire home/apt	175	
620	234870	Private Room With GREAT Location	1229984	John	Queens	Long Island City	40.74581	-73.95295	Private room	75	
991	400039	Big Beautiful Railroad in Brooklyn	1488809	John	Brooklyn	Bushwick	40.70339	-73.92945	Entire home/apt	130	
1141	484297	Large home in most desirable Brooklyn hood!	2397411	John	Brooklyn	Clinton Hill	40.68545	-73.96534	Entire home/apt	350	
...	...	...	...	...	...	...	...	...	...	...	
47624	35836317	Gorgeous Duplex 2BED/1.5BA Modern	269242923	John	Manhattan	Kips Bay	40.74490	-73.97888	Entire home/apt	288	
47689	35871036	Huge 1 bedroom w/ a backyard near the heart of...	226414996	John	Queens	Ditmars Steinway	40.77170	-73.90799	Entire home/apt	90	
47915	35984474	Perfect Weekend Stay	229739739	John	Brooklyn	Flatbush	40.64726	-73.95455	Private room	85	
48212	36140542	vacation	229739739	John	Brooklyn	Flatbush	40.64600	-73.95455	Private	85	
48705	36391615		70653354	John	Manhattan		40.72013	-73.98769		235	

Check Unique Values for each variable.

```
# Check Unique Values for each variable.
##
df_air['host_id'].nunique()
```

```
37457
```

```
df_air['host_name'].nunique()
```

```
11452
```

3. ***Data Wrangling***

Data Wrangling Code

```
# Write your code to make your dataset analysis ready.
# let analysis the data by using the chart
df_air.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	ca
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	

What all manipulations have you done and insights you found?

std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	

```
percentage_of_data_having_availbility_0 = round(len(df_air[df_air['availability_365']==0] ['availability_365'])/len(df_air['availability
print(f'The Percentage of data having availability as 0 is {percentage_of_data_having_availbility_0}')
```

```
The Percentage of data having availability as 0 is 35.86
```

So we can clearly notice that availability column is having minimum valuse as well as 25th percentile is 0. So that seem awkward beacuse having availability days 0 for 36% of data is bit shocking, If you have a business of providing sheltersfor AirBnb your availablity is 0 days that is extreme case and extreme case is obviously shocking when it come 36% of data is having extreme case. But its not practicaly to excactly detect which apartment are having realy availability 0 days, so we will not aalter this column as if we try to alter we can end up manipulating appartment which are really mostly busu(i.e 0 no. days availability)

we can also clearly see that minimum price is 0, which is surprising as price 0 does'nt make sense to do business

Let check out the last review wise count of where of where availability 365 is 0

```
df_air[df_air['availability_365']==0].groupby(['last_review']).size().sort_values(ascending=False).head(15)
```

```
last_review
2019-01-01    194
2018-01-01    142
2019-01-02    129
2019-06-23     90
2018-01-02     86
2017-01-01     85
2019-05-27     75
2017-01-02     73
2016-01-02     67
2019-07-01     63
2016-01-03     61
2018-12-30     61
2019-01-03     60
2019-06-24     59
2018-12-31     57
dtype: int64
```

Lets fill these data with appropriate price value(By filling the price with median price for each room type)

```
df_air.loc[ (df_air.room_type=='Entire home/apt') & (df_air.price==0),'price']=df_air.loc[(df_air.room_type=='Entire home/apt') & (df_ai
df_air.loc[ (df_air.room_type=='Private room') & (df_air.price==0),'price']=df_air.loc[(df_air.room_type=='Private room') & (df_air.pric
df_air.loc[ (df_air.room_type=='Shared room') & (df_air.price==0),'price']=df_air.loc[(df_air.room_type=='Shared room') & (df_air.price!
```

```
df_air.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	ca
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.739094	7.029962	23.274466	1.373221	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.146276	20.510550	44.550582	1.680442	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	10.000000	1.000000	0.000000	0.010000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	

NOTE: We can notic that we have succesfully updated the price column where we have values as 0,we succesfully updated the value with respective price value

```
df_air.fillna({'reviews_per_month':0},inplace=True)
```

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

```
df_air.head()
```

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

Observations: • Total 16 columns are present in the dataset. • total observations are 48895.

```
df_air.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               48895 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
```

```
#name column
df_air.name
```

```
0                    Clean & quiet apt home by the park
1                                 Skylit Midtown Castle
2                   THE VILLAGE OF HARLEM....NEW YORK !
3                       Cozy Entire Floor of Brownstone
4      Entire Apt: Spacious Studio/Loft by central park
                              ...
48890     Charming one bedroom - newly renovated rowhouse
48891       Affordable room in Bushwick/East Williamsburg
48892              Sunny Studio at Historical Neighborhood
48893                 43rd St. Time Square-cozy single bed
```

```
48894    Trendy duplex in the very heart of Hell's Kitchen
Name: name, Length: 48895, dtype: object
```

```
df_air[df_air['name'].isnull()].head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights
2854	1615764	NaN	6676776	Peter	Manhattan	Battery Park City	40.71239	-74.01620	Entire home/apt	400	1000
3703	2232600	NaN	11395220	Anna	Manhattan	East Village	40.73215	-73.98821	Entire home/apt	200	1
5775	4209595	NaN	20700823	Jesse	Manhattan	Greenwich Village	40.73473	-73.99244	Entire home/apt	225	1
5975	4370230	NaN	22686810	Michaël	Manhattan	Nolita	40.72046	-73.99550	Entire home/apt	215	7
6269	4581788	NaN	21600904	Lucie	Brooklyn	Williamsburg	40.71370	-73.94378	Private room	150	1

Observation:

1. This columns is having names describing the property which is host is trying to give on rent,so the nature of this names is short and consise and this is required as this can draw an attention of customer.
2. The question arises that how to fill the missing values in this columns.we will explore further dataset and try to find out better optiions to fill the missing values.
3. This Feature can be important in model building like Recommender systems.
4. Of course there is no point in removing these cells although they are limited in numbers.

```
df_air.room_type
```

```
0           Private room
1        Entire home/apt
2           Private room
3        Entire home/apt
4        Entire home/apt
              ...
48890       Private room
48891       Private room
48892    Entire home/apt
48893        Shared room
48894       Private room
Name: room_type, Length: 48895, dtype: object
```

Observations

1. room_type column is not having any null values,and also we can try to use this values in place of NAN values in name column this can solve our purpose and we can atlest put front what is the type of room!
2. yes, we will replace the nan values with the values which are in room_type column.

```
#This code snippit will replce the nan values.
#fillna() method will do the job...
df_air.name.fillna(df_air.room_type, inplace=True)
#check the changes.
df_air.isnull().sum()
```

```
id                                    0
name                                  0
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                     0
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

Observations:

1. We have replaced the nan values with corresponding room_type values.
2. This should solve our purpose.

```
#list of words.
name_list = list(df_air.name.values)
words = []
for i in name_list:
  words+=i.split()
#let's see top 50 used words .
_top_50_words=Counter(words).most_common()
_top_50_words=_top_50_words[0:50]
#dataFrame for top 50 words.
top_50_words = pd.DataFrame(_top_50_words,columns = ['words','frequency'])
# visualization
plt.figure(figsize=(20,5))
ax_1= sns.barplot(x='words',y='frequency',data = top_50_words)
ax_1.set_title('top 50 words')
ax_1.set_ylabel('frequency of words')
ax_1.set_xlabel('Words')
ax_1.set_xticklabels(ax_1.get_xticklabels(), rotation=60)
```

```
[Text(0, 0, 'in'),
 Text(0, 0, 'Private'),
 Text(0, 0, 'Room'),
 Text(0, 0, 'room'),
 Text(0, 0, 'Bedroom'),
 Text(0, 0, 'Cozy'),
 Text(0, 0, 'Apartment'),
 Text(0, 0, 'to'),
 Text(0, 0, 'Brooklyn'),
 Text(0, 0, '1'),
 Text(0, 0, '2'),
 Text(0, 0, 'the'),
 Text(0, 0, 'bedroom'),
 Text(0, 0, 'of'),
```

Observations:

1) there are top 50 words use in data fram

```
 Text(0, 0, 'Studio'),
```

```
neigh_unique_values = df_air['neighbourhood'].value_counts()
neigh_unique_values
```

```
Williamsburg          3920
Bedford-Stuyvesant    3714
Harlem                2658
Bushwick              2465
Upper West Side       1971
                      ...
Fort Wadsworth           1
Richmondtown             1
New Dorp                 1
Rossville                1
Willowbrook              1
Name: neighbourhood, Length: 221, dtype: int64
 Text(0, 0, 'West'),
```

```
#.most_common() Return a list of the n most common elements and their counts from the most common to the least.
top_50_=Counter( df_air['neighbourhood']).most_common()
top_50_=top_50_[0:50]
top_50_[:20]
```

```
[('Williamsburg', 3920),
 ('Bedford-Stuyvesant', 3714),
 ('Harlem', 2658),
 ('Bushwick', 2465),
 ('Upper West Side', 1971),
 ("Hell's Kitchen", 1958),
 ('East Village', 1853),
 ('Upper East Side', 1798),
 ('Crown Heights', 1564),
 ('Midtown', 1545),
 ('East Harlem', 1117),
 ('Greenpoint', 1115),
 ('Chelsea', 1113),
 ('Lower East Side', 911),
 ('Astoria', 900),
 ('Washington Heights', 899),
 ('West Village', 768),
 ('Financial District', 744),
 ('Flatbush', 621),
 ('Clinton Hill', 572)]
```

```
#count_plot
plt.figure(figsize=(20,5))
ax_4 = sns.barplot(x='neighbourhood',y='count',data = pd.DataFrame(top_50_,columns=['neighbourhood','count'][:20]))
ax_4.set_title('Count of Neighbourhood')
ax_4.set_ylabel('Frequency')
ax_4.set_xlabel('Neighbourhood')
ax_4.set_xticklabels(ax_4.get_xticklabels(), rotation=80);
plt.show()
```

Observations:

1. The above plot shows us some of the top Neighbours towns we can say.
2. People like to stay at these towns more offent.

```
df_air['host_name'].nunique()
```

```
11452
```

Observations:

1. host_name this column is defining the name of host(owner).
2. There are 11452 unique hosts/owners we can use this featue directly in model building just by encoding it.response encoding will be usefull for this feature.
3. we can take Nan value as one data point for model building.

```
df_air['neighbourhood_group']
```

```
0         Brooklyn
1        Manhattan
2        Manhattan
3         Brooklyn
4        Manhattan
           ...
48890     Brooklyn
48891     Brooklyn
48892    Manhattan
48893    Manhattan
48894    Manhattan
Name: neighbourhood_group, Length: 48895, dtype: object
```

```
df_air['neighbourhood_group'].unique()
```

```
#count_plot
plt.figure(figsize=(15,8))
ax_3 = sns.countplot(x='neighbourhood_group',data = df_air)
ax_3.set_title('Count of neighbourhood_group')
ax_3.set_ylabel('frequency')
ax_3.set_xlabel('neighbourhood_group')
plt.show()
```

Observations:

1. Brooklyn and Manhattan have the highest hotel/room bookings.

```
neigh_unique_values = df_air['neighbourhood'].value_counts()
neigh_unique_values
```

```
Williamsburg          3920
Bedford-Stuyvesant    3714
Harlem                2658
Bushwick              2465
Upper West Side       1971
                      ...
Fort Wadsworth           1
Richmondtown             1
New Dorp                 1
Rossville                1
Willowbrook              1
Name: neighbourhood, Length: 221, dtype: int64
```

Observations:

1. There are 221 unique neighbor

```
df_air[['latitude','longitude']]
```

	latitude	longitude
0	40.64749	-73.97237
1	40.75362	-73.98377
2	40.80902	-73.94190
3	40.68514	-73.95976
4	40.79851	-73.94399
...	...	...
48890	40.67853	-73.94995
48891	40.70184	-73.93317
48892	40.81475	-73.94867
48893	40.75751	-73.99112
48894	40.76404	-73.98933

48895 rows × 2 columns

Observation:

1. we can see the exact locations from this columns.

1) Which hosts are having heighest number of appartments ?

```
## Which Host are have heighest numer of appartment
### In this i will demonstrate that why we need to go   with host_id rather then host_name

df_air['host_name'].value_counts()
```

```
Michael              417
David                403
Sonder (NYC)         327
John                 294
Alex                 279
                    ...
Rhonycs                1
Brandy-Courtney        1
Shanthony              1
Aurore And Jamila      1
Ilgar & Aysel          1
Name: host_name, Length: 11452, dtype: int64
```

Observation:

From this we can see that host name michael its appearing 417 time in the host_name column, so this might imply that michael is having heighest number of room,but from the host_id column its showing heighest appearance of any host_id is 327, so this clearly implies that there can be multiple may have same name thats why we are we are getting diffrent heighest apperance in host_name as campared to host_id.

Lets check which host_name is actually having heighest number of apperments

```
df_air[['host_name','host_id']].value_counts()
```

```
host_name       host_id
Sonder (NYC)    219517861    327
Blueground      107434423    232
Kara            30283594     121
Kazuya          137358866    103
Jeremy & Laura  16098958      96
                            ...
Graham          22550881       1
                10103520       1
                8952737        1
                6407741        1
현선             18497228       1
Length: 37439, dtype: int64
```

```
df_air['host_id'].value_counts()
```

```
219517861    327
107434423    232
30283594     121
137358866    103
16098958      96
            ...
23727216       1
89211125       1
19928013       1
1017772        1
68119814       1
Name: host_id, Length: 37457, dtype: int64
```

```
df_air[df_air['host_id']==219517861]['host_name'].unique()
```

```
array(['Sonder (NYC)'], dtype=object)
```

```
## So sonder (NYC) is having maximum number of rooms for the guest, for Airbnb he mignt be very important person.

df_sonder= df_air[df_air['host_name']=='Sonder (NYC)']
df_sonder[['host_name','neighbourhood_group','neighbourhood','latitude','longitude']].head(6)
```

	host_name	neighbourhood_group	neighbourhood	latitude	longitude
38293	Sonder (NYC)	Manhattan	Financial District	40.70637	-74.00645
38294	Sonder (NYC)	Manhattan	Financial District	40.70771	-74.00641
38588	Sonder (NYC)	Manhattan	Financial District	40.70743	-74.00443
39769	Sonder (NYC)	Manhattan	Murray Hill	40.74792	-73.97614
39770	Sonder (NYC)	Manhattan	Murray Hill	40.74771	-73.97528
39771	Sonder (NYC)	Manhattan	Murray Hill	40.74845	-73.97446

Sonder(NYC) is having multiple appermen in same building in diffrent diffrent neighbourdhood

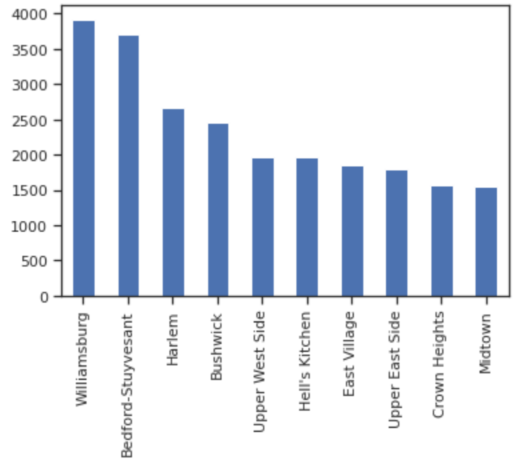
2)Which are the top 10 neighbourdhood which are having maximum number of appertment for Airbnb?

```
df_air['neighbourhood'].value_counts().head(10)
```

```
Williamsburg          3920
Bedford-Stuyvesant    3714
Harlem                2658
Bushwick              2465
Upper West Side       1971
Hell's Kitchen        1958
East Village          1853
Upper East Side       1798
Crown Heights         1564
Midtown               1545
Name: neighbourhood, dtype: int64
```

```
# plotting top neighbourhood which are having maximum number of appartments for airbnb in the respective neighbourhood.
pd.value_counts(df_air['neighbourhood'])[:10].plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fcd502fb6a0>



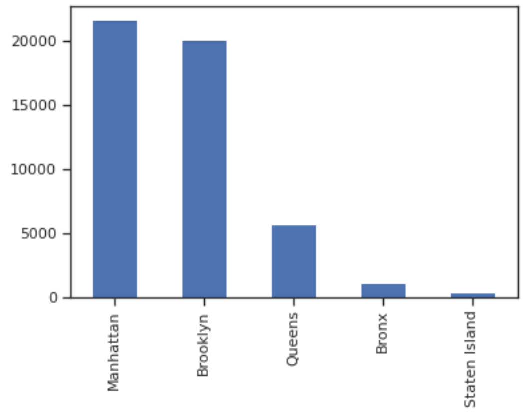
3) What are the neighbourhood in each group which are having maximum price in thier respective neighbourhood_group

```
df_air['neighbourhood_group'].value_counts()
```

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

```
pd.value_counts(df_air['neighbourhood_group']).plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fcd5034d730>




```
#Top 3 neighbourhood in thier respective neighbourhood group which are having maximum prices.

df_Manhattan=df_air[df_air['neighbourhood_group']=='Manhattan']
df_Brooklyn=df_air[df_air['neighbourhood_group']=='Brooklyn']
df_Queens=df_air[df_air['neighbourhood_group']=='Queens']
df_Bronx=df_air[df_air['neighbourhood_group']=='Bronx']
df_Staten=df_air[df_air['neighbourhood_group']=='Staten Island']

# top 3  neighbourhood in Manhattan which are having maximum prices
print('Top 3 neighbourhood in manhattan which are having maximum price')
df_Manhattan.groupby(['neighbourhood'])['price'].max().sort_values(ascending=False).reset_index().head(3)
```

Top 3 neighbourhood in manhattan which are having maximum price

	neighbourhood	price
0	Upper West Side	10000
1	East Harlem	9999
2	Lower East Side	9999

```
# top 3  neighbourhood in Manhattan which are having maximum prices
print('Top 3 neighbourhood in manhattan which are having maximum price')
```

```
df_Brooklyn.groupby(['neighbourhood'])['price'].max().sort_values(ascending=False).reset_index().head(3)
```

Top 3 neighbourhood in manhattan which are having maximum price

	neighbourhood	price
0	Greenpoint	10000
1	Clinton Hill	8000
2	East Flatbush	7500

```
# top 3  neighbourhood in Manhattan which are having maximum prices
print('Top 3 neighbourhood in manhattan which are having maximum price')
df_Queens.groupby(['neighbourhood'])['price'].max().sort_values(ascending=False).reset_index().head(3)
```

Top 3 neighbourhood in manhattan which are having maximum price

	neighbourhood	price
0	Astoria	10000
1	Bayside	2600
2	Forest Hills	2350

```
# top 3  neighbourhood in Manhattan which are having maximum prices
print('Top 3 neighbourhood in manhattan which are having maximum price')
df_Bronx.groupby(['neighbourhood'])['price'].max().sort_values(ascending=False).reset_index().head(3)
```

Top 3 neighbourhood in manhattan which are having maximum price

	neighbourhood	price
0	Riverdale	2500
1	City Island	1000
2	Longwood	680

```
# top 3  neighbourhood in Manhattan which are having maximum prices
print('Top 3 neighbourhood in manhattan which are having maximum price')
df_Staten.groupby(['neighbourhood'])['price'].max().sort_values(ascending=False).reset_index().head(3)
```

Top 3 neighbourhood in manhattan which are having maximum price

	neighbourhood	price
0	Randall Manor	5000
1	Prince's Bay	1250
2	St. George	1000

4) How neidhbourhood is realed with reviews?

```
##Top 5 Neighbourhood is having heighest review per month
df_air.groupby(['neighbourhood'])['reviews_per_month'].max().sort_values(ascending=False).reset_index().head(5)
```

	neighbourhood	reviews_per_month
0	Theater District	58.50
1	Rosedale	20.94
2	Springfield Gardens	19.75
3	East Elmhurst	16.22
4	Jamaica	15.32

Top 5 Neighbourhood is having heighest number of views

```
df_air.groupby(['neighbourhood'])['number_of_reviews'].sum().sort_values(ascending=False).reset_index().head(5)
```

neighbourhood number of reviews

```
df_air[df_air['name'].isnull()].head()
```

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of

Hell's Kitchen 50227

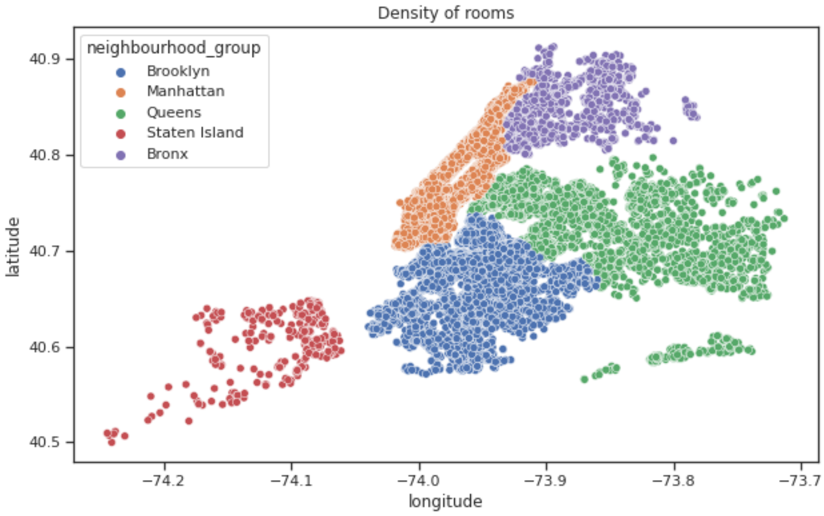
Observation:

1. This columns is having names describing the property which is host is trying to give on rent,so the nature of this names is short and consise and this is required as this can draw an attention of customer.
2. The question arises that how to fill the missing values in this columns.we will explore further dataset and try to find out better optiions to fill the missing values.
3. This Feature can be important in model building like Recommender systems.
4. Of course there is no point in removing these cells although they are limited in numbers.

5) What can we learn from predictions?

```
#simple scatterplot:

plt.figure(figsize=(10,6))
ax_5 = sns.scatterplot(df_air.longitude,df_air.latitude,hue=df_air.neighbourhood_group)
ax_5.set_title('Density of rooms')
ax_5.set_ylabel('latitude')
ax_5.set_xlabel('longitude')
plt.show()
```

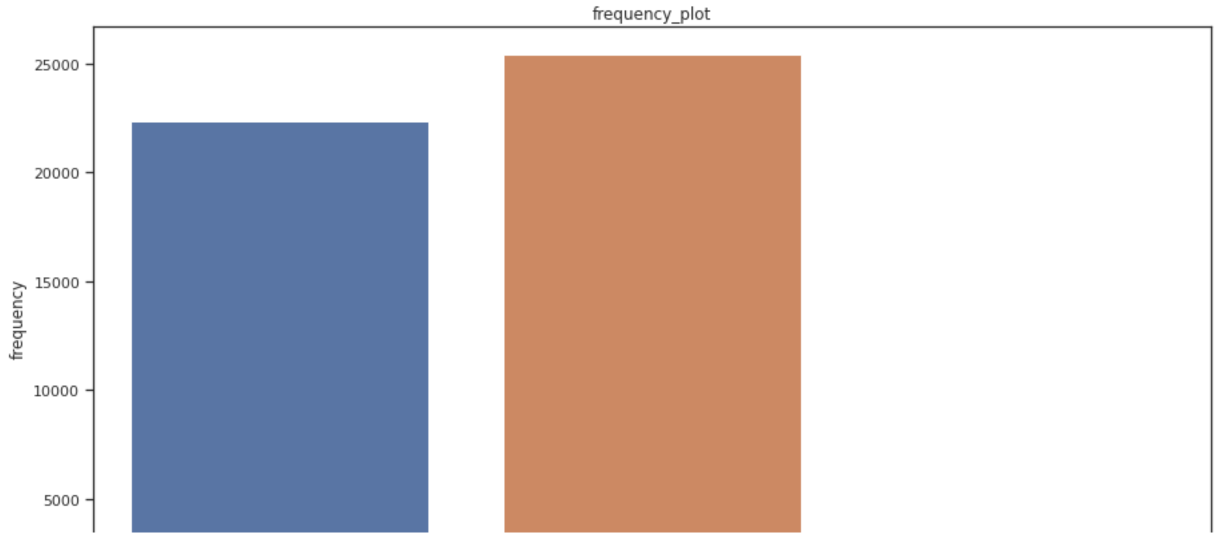
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y
  warnings.warn(
```



Observations:

As we see neighbourhood group location

```
# Column_no_6 room_type:
df_air[['room_type']].nunique()

#count_plot'
plt.figure(figsize=(15,8))
sns_6 = sns.countplot(x='room_type',data = df_air)
sns_6.set_title('frequency_plot')
sns_6.set_ylabel('frequency')
sns_6.set_xlabel('room_type')

plt.show()
```



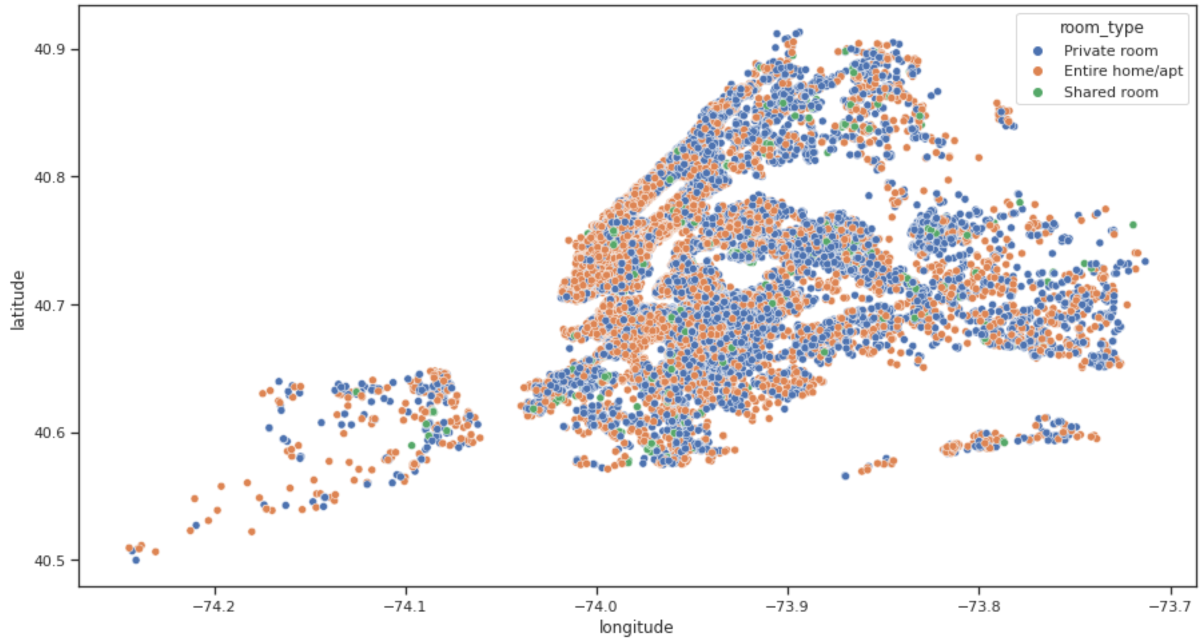
Observations There are three types of rooms

1. Private room
2. Entire home/apt room_type
3. Shared room.

People mostly prefered to take whole apartment on rent followed by Private room.very few people prefered to have shared rooms.

```
plt.figure(figsize=(15,8))
sns.scatterplot(x=df_air['longitude'],y=df_air['latitude'], hue=df_air['room_type'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd5024dcd0>
```



6) What is the distribution of the room type and its distribution over the location ?

```
plt.figure(figsize=(8,5))
df_air['room_type'].value_counts().plot(kind='bar',color=['r','b','y'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd51e9d400>
```

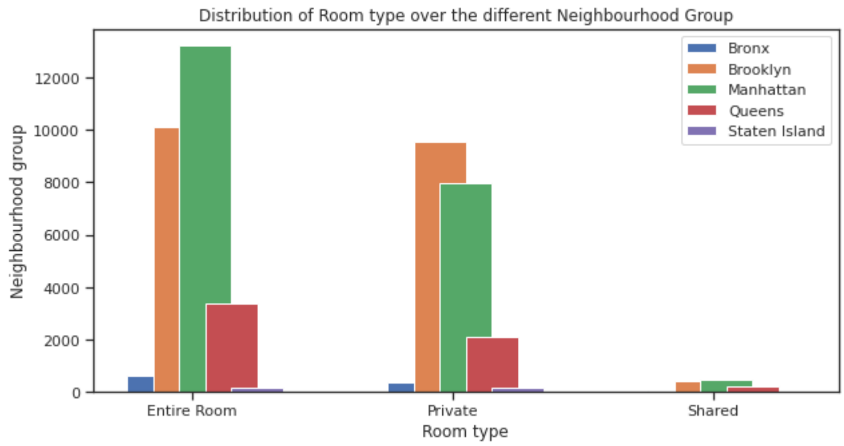
Observation

So we can notice the following

1. That maximum number of room are enter home/apartment and private room there are only few shared rooms.
2. So mostly host prefer to give entire home/apartment and private room rather than share rooms

7) How dose the room type is distributed over neighbourhood group are the ratios of respective room type more or less same over each neighbourhood group?

```
  plt.figure(figsize=(10,5))
  N=5 #number of bars in each category
  ind= np.arange(3)
  width=0.3

  # storing the values of all values count by the room type for specific neighbourhood group

bronx_values=df_Bronx['room_type'].value_counts().values
brooklyn_values=df_Brooklyn['room_type'].value_counts().values
manhattan_values=df_Manhattan['room_type'].value_counts().values
queen_values=df_Queens['room_type'].value_counts().values
staten_values=df_Staten['room_type'].value_counts().values
# plotting the values
plt.bar(ind,bronx_values,0.2,label='Bronx')
plt.bar(ind+0.1,brooklyn_values,0.2,label='Brooklyn')
plt.bar(ind+0.2,manhattan_values,0.2,label='Manhattan')
plt.bar(ind+0.3,queen_values,0.2,label='Queens')
plt.bar(ind+0.4,staten_values,0.2,label='Staten Island')
plt.xlabel('Room type')
plt.ylabel('Neighbourhood group')
plt.title('Distribution of Room type over the different Neighbourhood Group')
plt.xticks(ind + width / 2, ('Entire Room', 'Private', 'Shared'))
plt.legend(loc='best')
plt.show()
```

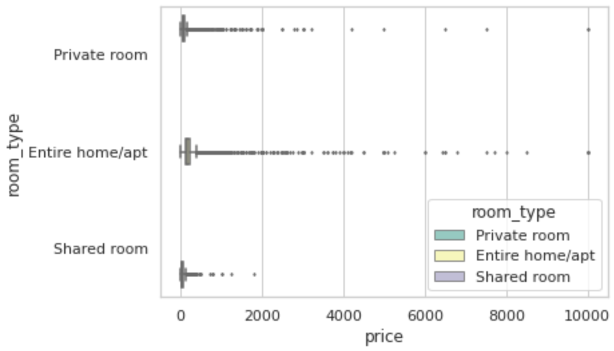



```
#It seem more or less same ratio in every neighbouurhood
```

8) How the price column is distributed over room_type and are there any surprising iteam in price column?

```
# From the previous expoloration we get to know that price coloumn is having many value as 0 as it doesn't make the sense.
#   So will try to get rid of those instances for analysis of price column.
df_price = df_air[df_air['price']!=0].copy()

sns.set_theme(style='whitegrid')
sns.boxplot(y='room_type',x='price',hue='room_type',palette='Set3',linewidth=1.5,fliersize=1.5,data=df_price)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd4bd68400>
```



Observations

We can notice that there are many outliers for price in each of the room type category, so lets just why there is so high price or what else we can conclude for host have highest price for the rooms

```
# let check our the who is having highest price of all.
# and we will check its   rating, minimum night,availability_365 and last reviews in order jude.

df_air[df_air['price']==df_air['price'].max()][['host_name','reviews_per_month','last_review','availability_365','price','neighbourhood_
```

	host_name	reviews_per_month	last_review	availability_365	price	neighbourhood_group
9151	Kathrine	0.04	2016-02-13	0	10000	Queens
17692	Erin	0.16	2017-07-27	0	10000	Brooklyn
29238	Jelena	0.00	NaN	83	10000	Manhattan

Observations

clearly if i would have working in Airbnb i would have suggested the following

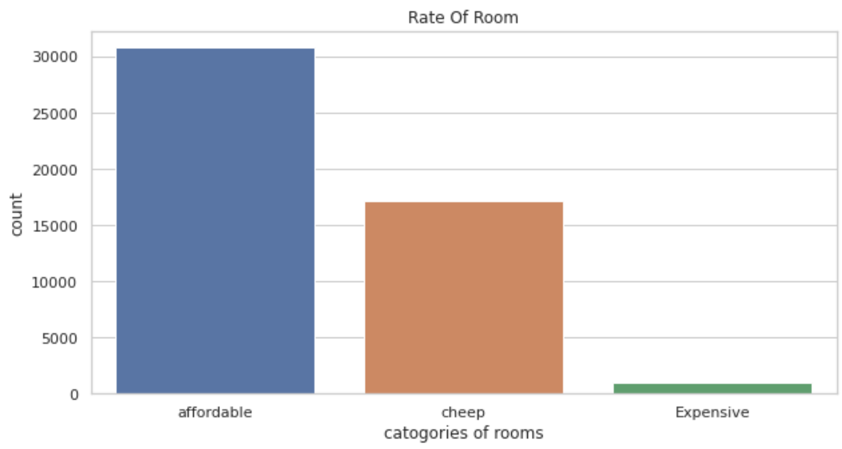
1. Kathrine and Erin have price so high and having no availability then what is the benifit of keeping too high price.
2. The last review is also 2-3 years back.Which is also bad.
3. The review may be low as there may be very few people who is staying in kathrine, Erin and jelena apartment so might have less reviews per month
4. I would have suggested to keep moderate(average) price so that more people would visit and stay in her appartment , it would also increase her reviews per month

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

	price
count	48895.000000
mean	152.739094
std	240.146276
min	10.000000
25%	69.000000
50%	106.000000
75%	175.000000
max	10000.000000

```
# Function to catogory the type of rooms.
def price_catagory(price):
  if price<=80:
          return "cheep"
  elif price>=80 and price<=500:
          return 'affordable'
  else:
          return 'Expensive'

plt.figure(figsize=(10,5))
ax_7 = sns.countplot(x=df_air['price'].apply(price_catagory))
ax_7.set_title('Rate Of Room')
ax_7.set_xlabel('catogories of rooms')
ax_7.set_ylabel('count')
```

```
plt.show()
```



Observations

1. We have considered to devide the whole price range into three catogories
2. cheep(price range below or equal to 80) ***B.Affordable(forprice range 80 to 500)***
3. Expensive(for price more then 500$)so its look like people having more intrest in having "affordable" rooms/appartments rathre then having cheep and expensive rooms .

9) Which are the top 5 host that have obatained heighest no.of review?

```
host_highest_df = df_air.groupby(['host_id','host_name'],as_index=False)['number_of_reviews'].sum().sort_values(['number_of_reviews'],as
host_highest_df
```

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

37439 rows × 3 columns

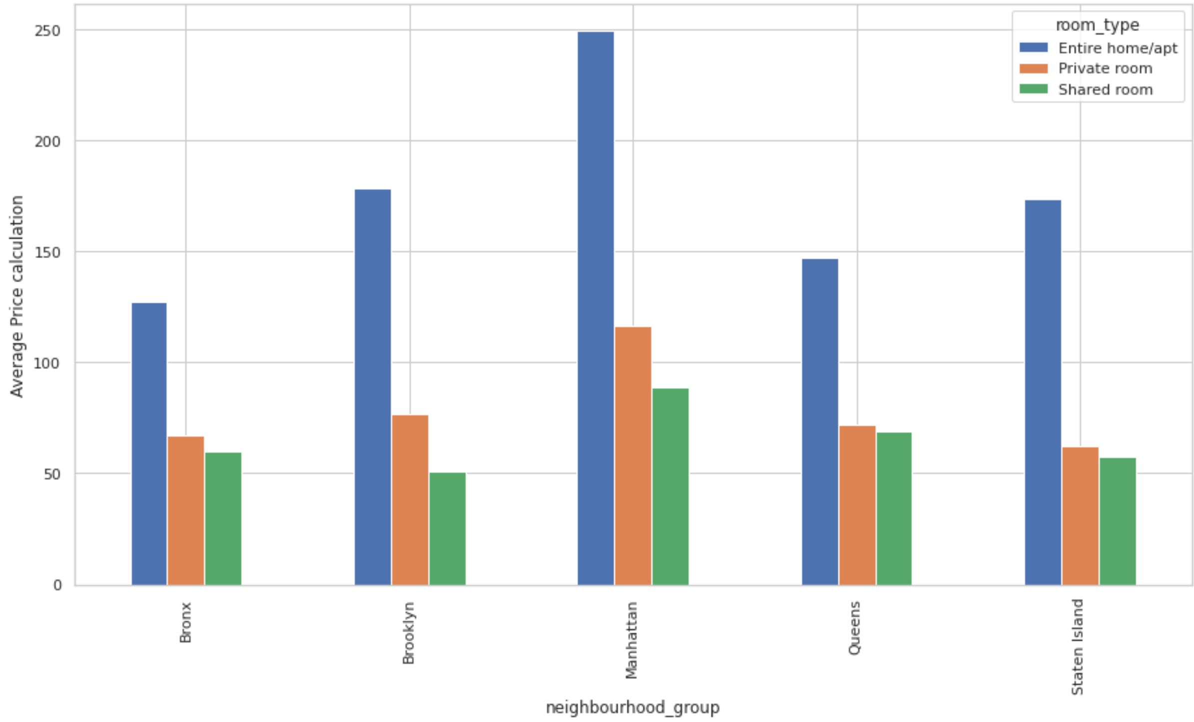
10) What is the average preferred price by customers according to the neighbourhood group for each category of room type ?

```
#applying groupby over neighbourhood group and room type
# That applying mean of price and unstacking for clear visualization

avg_price_df = df_air.groupby(['neighbourhood_group','room_type'])['price'].mean().unstack()
avg_price_df
```

room_type	Entire home/apt	Private room	Shared room
neighbourhood_group			
Bronx	127.506596	66.895706	59.800000
Brooklyn	178.344283	76.541552	50.745763
Manhattan	249.251231	116.776622	88.977083
Queens	147.050573	71.762456	69.020202
Staten Island	173.846591	62.292553	57.444444

```
avg_price_df.plot.bar(figsize=(15,8),ylabel='Average Price calculation')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd5008d2e0>
```



Observations

As we can see that manhattan is most costly and bronx is cheap for each room type

But I think we can make it more useful for buissness implimentation if we do some analysis on successfull hosts according to the heighest no of reviews so that we can suggest this price to our host for good buisness.

11) We had deeply analysis the data here

```
df_air[['minimum_nights']].value_counts()
```

```
minimum_nights
1                 12720
2                 11696
3                  7999
30                 3760
4                  3303
                  ...
182                   1
183                   1
184                   1
185                   1
1250                  1
Length: 109, dtype: int64
```

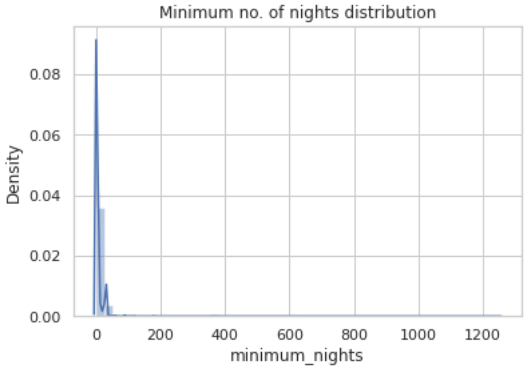
```
df_air[['minimum_nights']].describe()
```

	minimum_nights
count	48895.000000
mean	7.029962
std	20.510550
min	1.000000
25%	1.000000
50%	3.000000
75%	5.000000
max	1250.000000

```
ax = sns.distplot(df_air.minimum_nights)
plt.title('Minimum no. of nights distribution')
```

```
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will t
  warnings.warn(msg, FutureWarning)
```



Observations

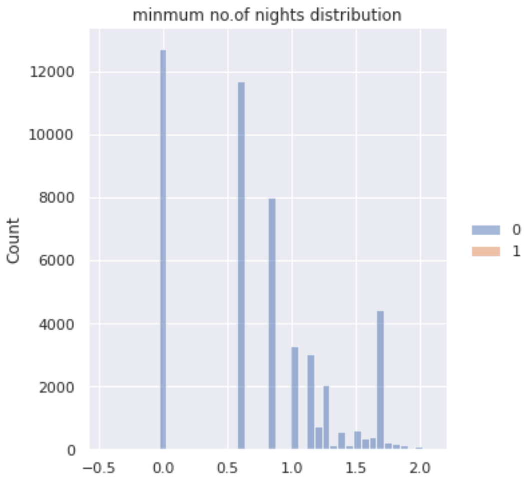
1. Average booking is around 7 nights.
2. minimum booking is for 1 night.
3. max booking is for more then a year or we can say for few years.

```
from scipy.stats import boxcox
# power transform
data_box_cox_transform = boxcox(df_air.minimum_nights)
#lambda=0 it means log transform by defination of box-cox transform.

data_box_cox_transform
```

```
(array([0.        , 0.        , 0.86162699, ..., 1.41767289, 0.        ,
        1.28379566]), -0.46182559978389276)
```

```
from time import thread_time
from matplotlib import text
from pyparsing.helpers import string
sns.set_theme();  np.random.seed(0)
plt.figure(figsize=(10,5))
ax = sns.displot(data_box_cox_transform)
plt.title ('minmum no.of nights distribution')

text={0.5,1.0,'minimum no.of night distribution'}
```

```
<Figure size 720x360 with 0 Axes>
```



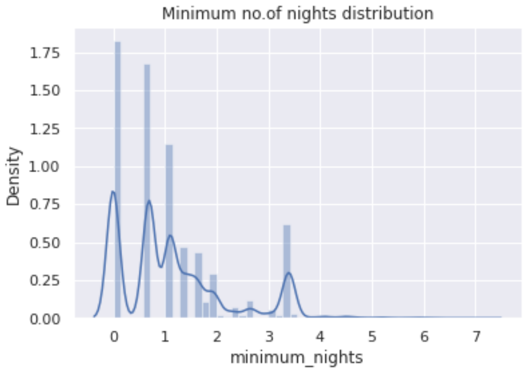
Observations:

1. it's very clear that the data is right skwed.

```
from nltk.draw.util import Text
log_transfrom = np.log(df_air['minimum_nights'])
```

```
ax = sns.distplot(log_transfrom)
plt.title('Minimum no.of nights distribution')
Text={0.5, 1.0, 'Minimum no.of nights distribution'}
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will t
  warnings.warn(msg, FutureWarning)
```



Observations

1. This plots shows that majority of room booking is one for 1 to 4 days.
2. Box-Cox transformed plot strictly shows that the majority of booking lies between 0 to 3 days.we have set the lambda parameter not equal to zero so it by defination of bax-cox transform selected the best value of lambda.

```
df_air[['number_of_reviews']].value_counts()
```

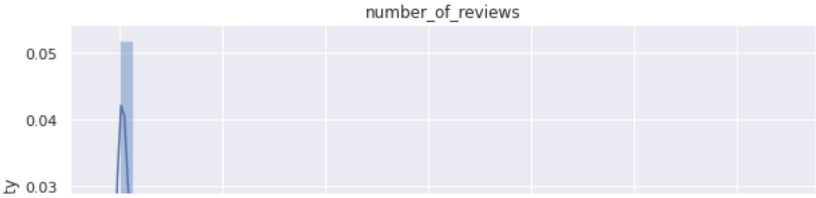
```
number_of_reviews
0                    10052
1                     5244
2                     3465
3                     2520
4                     1994
                     ...
352                      1
351                      1
341                      1
340                      1
629                      1
Length: 394, dtype: int64
```

```
df_air[['number_of_reviews']].describe()
```

	number_of_reviews
count	48895.000000
mean	23.274466
std	44.550582
min	0.000000
25%	1.000000
50%	5.000000
75%	24.000000
max	629.000000

```
plt.figure(figsize=(10,5))
ax = sns.distplot(x=df_air["number_of_reviews"])
plt.title('number_of_reviews')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will b
  warnings.warn(msg, FutureWarning)
Text(0.5, 1.0, 'number_of_reviews')
```




```
#the distribution tells it has positive skew
# also the distribution doesn't deviate much form normal distribution.
# skewness and kurtosis

print('Skewness: %f' % df_air['price'].skew())
print('kurtosis: %f' % df_air['price'].kurt())
```

```
Skewness: 19.120643
kurtosis: 585.744382
```

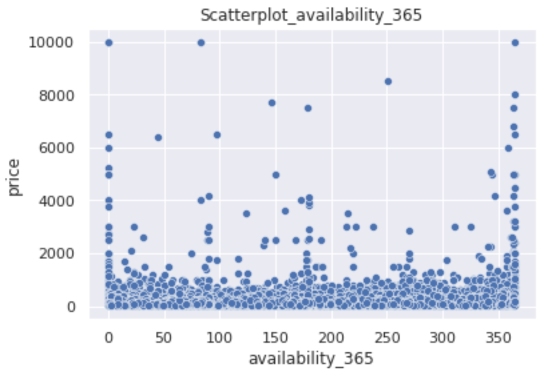
Observations

1. look the skew and kurtosis come out very large.Since the skewness has value >1 it is highly skewed.
2. Also kurtosis look high as well which indicates presence of good amount of outliers,We will look later into that when we handle outliers!!

```
df_air[['availability_365']].value_counts()
```

```
availability_365
0                   17533
365                  1295
364                   491
1                     408
89                    361
                     ...
195                    26
196                    24
183                    24
181                    23
202                    20
Length: 366, dtype: int64
```

```
ax= sns.scatterplot(data=df_air,x='availability_365',y='price')
plt.title("Scatterplot_availability_365")

text={0.5, 1.0,'Scatterplot_availability_365'}
```



Observation

1. From above plot we can see that most of the available rooms are in the prose range of 0 to 2000
2. Very few are available for price above 2000$, this is quite oblivious that are very few peoples who prefer to have expensive rooms.

```
#price vs minimum_nights

var='minimum_nights'
data = pd.concat([df_air['price'],df_air[var]],axis=1)
data.plot.scatter(x=var,y='price',ylim=(0,12000))
plt.title("Price Vs Minimum_Nights")
```

```
WARNING:matplotlib.axes._axes:*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mappi
Text(0.5, 1.0, 'Price Vs Minimum_Nights')
```

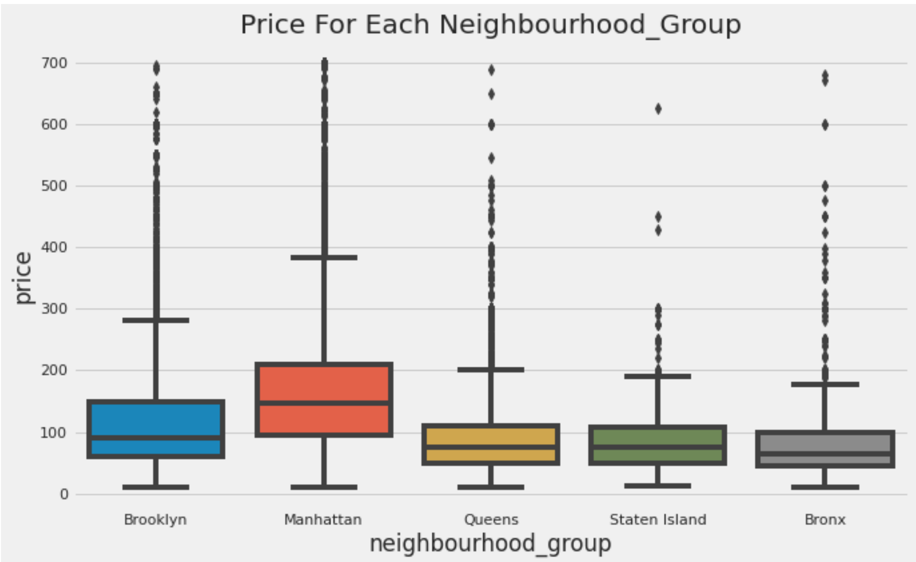
Observations

Look many data point are clustured on 0 price range, few have min night for stay but price is 0. look like anomaly in price.

```
price_df = pd.DataFrame(df_air['price'].apply(price_catagory))
price_df.head()
```

	price
0	affordable
1	affordable
2	affordable
3	affordable
4	cheep

```
plt.style.use('fivethirtyeight')

price_500 = df_air[df_air.price<700]
plt.figure(figsize=(10,6))
plt.title("Price For Each Neighbourhood_Group")
sns.boxplot(y ='price',x ='neighbourhood_group',data=price_500)
plt.show()
```



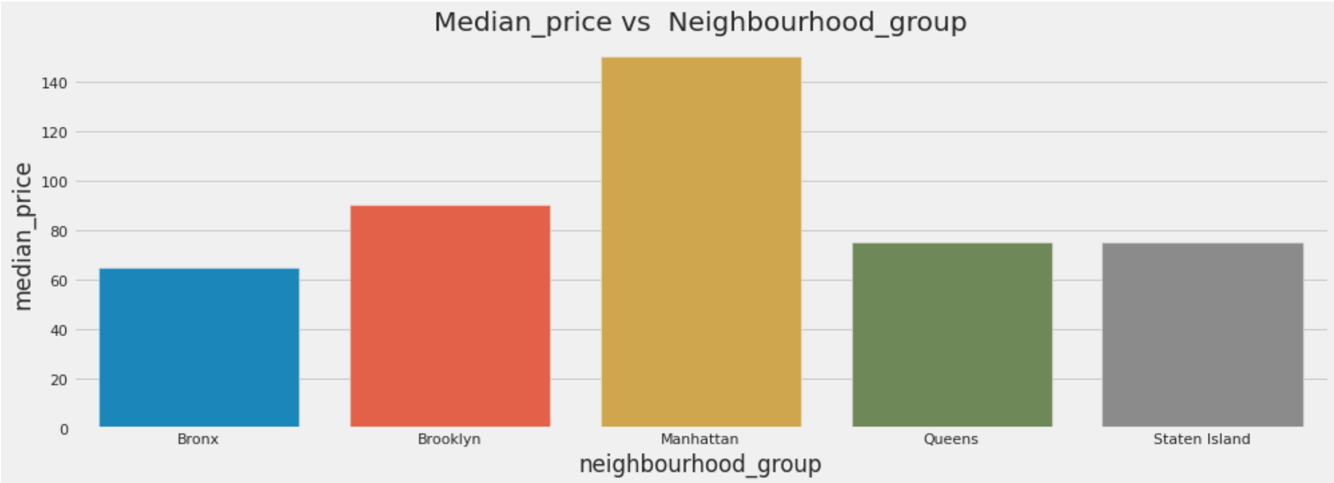
observations:

1. We can see that manhattan is the most expensive destination immedialty followed by brooklyn.
2. Queens, staten island and bronx, are having price rang less as compaired to other two.

```
# grouping median price with neighbourhood_group
neigh_group_price_group = df_air.groupby(['neighbourhood_group']).agg({'price':'median'}).reset_index()
neigh_group_price_group
```

	neighbourhood_group	price
0	Bronx	65.0
1	Brooklyn	90.0
2	Manhattan	150.0
3	Queens	75.0

```
plt.figure(figsize=(15,5))
ax_12 =sns.barplot(x= 'neighbourhood_group', y = 'price', data = neigh_group_price_group)
ax_12.set_title('Median_price vs  Neighbourhood_group')
ax_12.set_xlabel('neighbourhood_group')
ax_12.set_ylabel('median_price')
plt.show()
```



Observations:

In this chart, we have done median price Vs neighbourhood group

```
df_air.boxplot(column=['price'])
plt.show()
```

Observations

With help of this chart we can see the presence of many outliers in price. definitely we'll remove those patience!

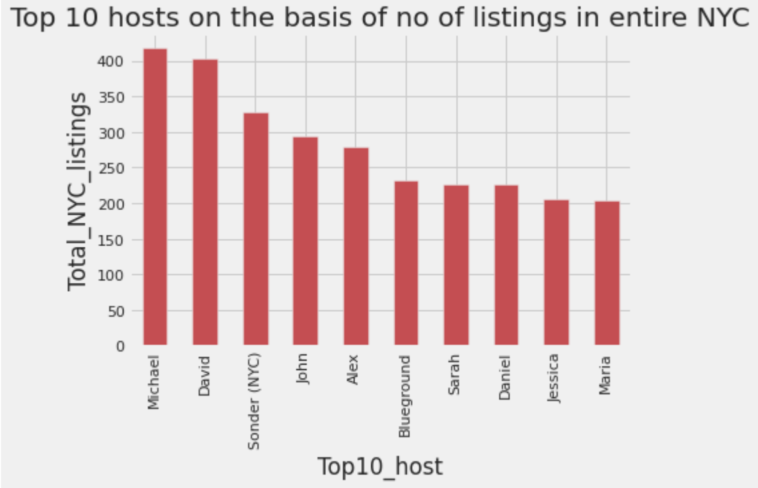
Double-click (or enter) to edit

```
# Chart - 5 visualization code.
top_10_hosts=df_air['host_name'].value_counts()[:10]
top_10_hosts #top 10 hosts on the basis of no of listings in entire NYC
```

```
Michael         417
David           403
Sonder (NYC)    327
John            294
Alex            279
Blueground      232
Sarah           227
Daniel          226
Jessica         205
Maria           204
Name: host_name, dtype: int64
```

```
top_10_hosts.plot(kind='bar',color="r")
plt.xlabel('Top10_host')
plt.ylabel('Total_NYC_listings')
plt.title('Top 10 hosts on the basis of no of listings in entire NYC')
```

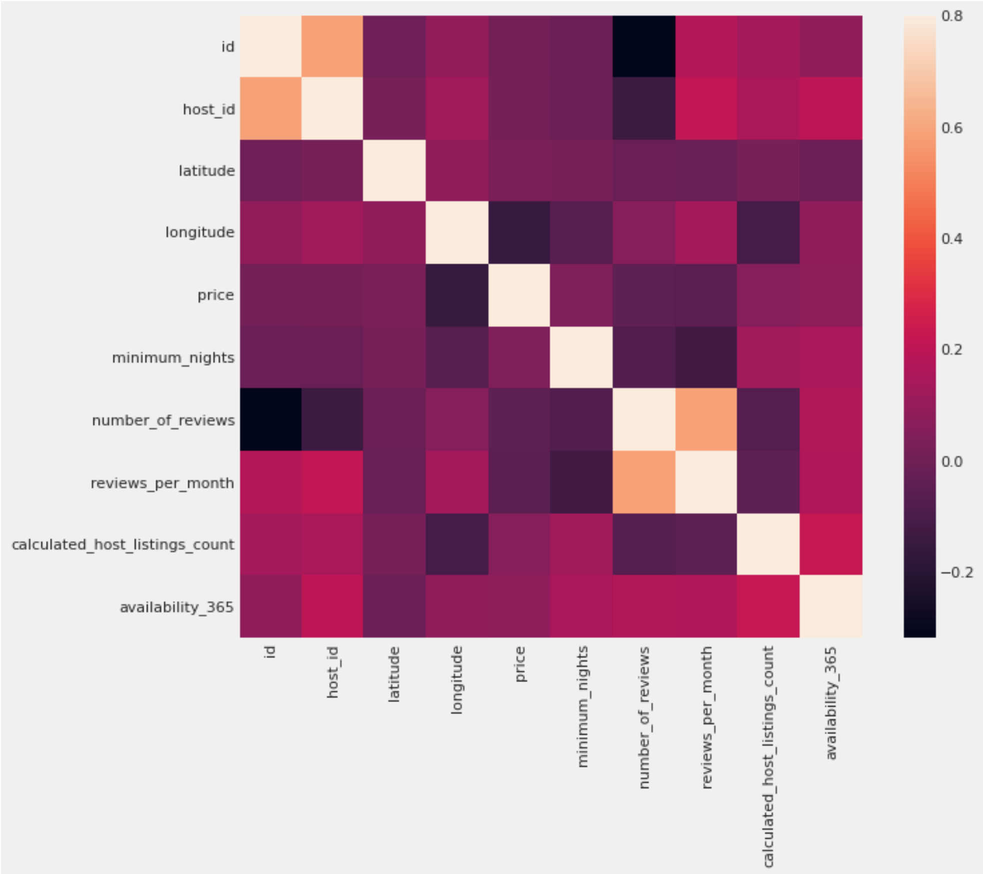
```
Text(0.5, 1.0, 'Top 10 hosts on the basis of no of listings in entire NYC')
```



Observations

1.This chart is good to show the top 10 hosts on the basis of no of listings in entire NYC

```
#Chart Visualization Code.

corrmat = df_air.corr()
f, ax = plt.subplots(figsize=(10,8))
sns.heatmap(corrmat, vmax= .8, square=True);
```

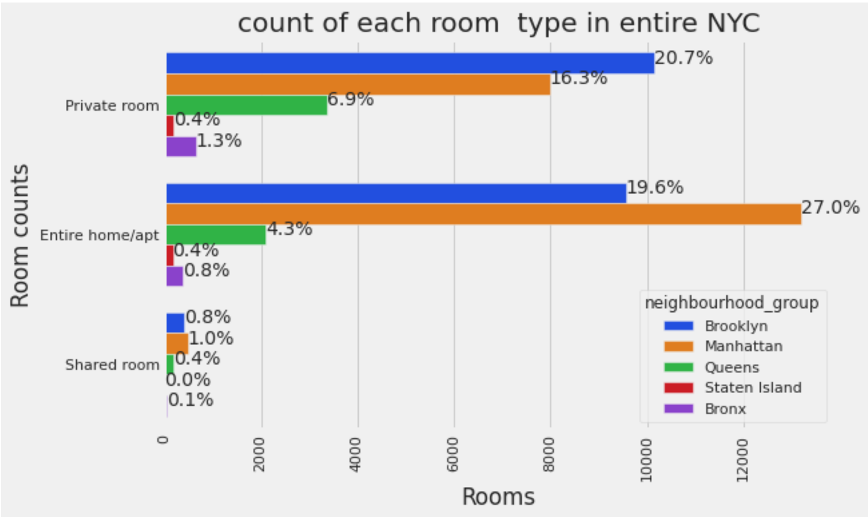


Heatmap shows the correlation between diffrent feature

We an see that the corelation amoung the host_id to review per month and avilability 360 and also there are correlation in the min_night to no.off listing count and avilability 360. the price also show the sum correlation with avilability 360 and host_listing_count.

We get the overall correlaion between the different features that can effect the listing. so we know that correlation that will help to analysis the future.

```
plt.rcParams['figure.figsize']=(8,5)
ax =sns.countplot(y='room_type',hue='neighbourhood_group',data=df_air,palette='bright')

total = len(df_air['room_type'])
for p in ax.patches:
    percentage = '{:.1f}%' .format(100 * p.get_width()/total)
    x = p.get_x() + p.get_width() + 0.02
    y = p.get_y() + p.get_height() /2
    ax.annotate(percentage,(x,y))

plt.title('count of each room  type in entire NYC')
plt.xlabel('Rooms')
plt.xticks(rotation=90)
plt.ylabel('Room counts')
plt.show()
```



Observations

Manhattan has more listed properties with Entire home/apt around 27% of total listed properties followed by Brooklyn with around 19.6%. Private rooms are more in Brooklyn as in 20.7% of the total listed properties followed by Manhattan with 16.3% of them. While 6.9% of private rooms are from Queens. Very few of the total listed have shared rooms listed on Airbnb where there's negligible or almost very rare shared rooms in Staten Island and Bronx. We can infer that Brooklyn,Queens,Bronx has more private room types while Manhattan which has the highest no of listings in entire NYC has more Entire home/apt room types.

3. Will the gained insights help creating a positive business impact? Are there any insights that lead to negative growth? Justify with specific reason.

Airbnb started with a simple idea: create an opportunity for travelers to experience life as a local in new and exciting destinations, without the hassle of hotels. Simple ideas often breed complex businesses, and Airbnb has now grown into a worldwide phenomenon. They facilitate rentals of millions of homes across the globe each year. This business success came about for a myriad of reasons — strong ideation, a growing customer base, and increased demand for travel. But Airbnb would not be successful without a powerful marketing strategy underpinning their operations. In this post we'll dive into Airbnb's marketing mix, the strategies that made them a successful brand, and how marketers can use these strategies to improve their own brand promotion operations

5. Solution to Business Objective

What do you suggest the client to achieve Business Objective ? Explain Briefly.

Airbnb's product offerings differ between easy, affordable vacation rentals for its customers and an earning opportunity for its hosts. Renters are where Airbnb makes most of its income. As such, they maintain a high-quality mobile app and internet presence where these customers can easily make rental reservations.

Hosts are Airbnb's life-blood. Without this market they would be unable to rent out locations. The company offers hosts impressive incentives, making Airbnb hosting an easy opportunity to add a consistent stream of income.