

THE SECRETS OF AIRBNB

DATA METHODOLOGY

1. Importing libraries and reading the data

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

```
inp0 = pd.read_csv('AB_NYC_2019.csv')
```

```
inp0 = pd.read_csv('AB_NYC_2019.csv')
inp0.head(10)
```

	id	name	host id	host name	neighbourhood group	neighbourhood	latitude	longitude	room type	price	minimum nights	number of reviews	last review	rev
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	19-10-2018	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	21-05-2019	
2	3647	THE VILLAGE OF HARLEM...NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	05-07-2019	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	19-11-2018	

2. Creating features

2.1 categorizing the "availability_365" column into 5 categories

```
def availability_365_categories_function(row):  
    """  
    Categorizes the "minimum_nights" column into 5 categories  
    """  
    if row <= 1:  
        return 'very Low'  
    elif row <= 100:  
        return 'Low'  
    elif row <= 200 :  
        return 'Medium'  
    elif (row <= 300):  
        return 'High'  
    else:  
        return 'very High'
```

2.2 categorizing the "minimum_nights" column into 5 categories

```
def minimum_night_categories_function(row):  
    """  
    Categorizes the "minimum_nights" column into 5 categories  
    """  
    if row <= 1:  
        return 'very Low'  
    elif row <= 3:  
        return 'Low'  
    elif row <= 5 :  
        return 'Medium'  
    elif (row <= 7):  
        return 'High'  
    else:  
        return 'very High'
```

2.3 categorizing the "number_of_reviews" column into 5 categories

```
def number_of_reviews_categories_function(row):  
    """  
    Categorizes the "number_of_reviews" column into 5 categories  
    """  
    if row <= 1:  
        return 'very Low'  
    elif row <= 5:  
        return 'Low'  
    elif row <= 10 :  
        return 'Medium'  
    elif (row <= 30):  
        return 'High'  
    else:  
        return 'very High'
```

2.4 categorizing the "price" column into 5 categories

```
inp0.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
```

3. Fixing columns

```
# To see Non-Null counts and data types
inp0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 48895 entries, 0 to 48894
```

```
Data columns (total 20 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
16	availability_365_categories	48895 non-null	object
17	minimum_night_categories	48895 non-null	object
18	number_of_reviews_categories	48895 non-null	object
19	price_categories	48895 non-null	object

```
dtypes: float64(3), int64(7), object(10)
```

```
memory usage: 7.5+ MB
```

Fix: reviews_per_month is of object Dtype. datetime64 is a better Dtype for this column.

```
inp0.last_review = pd.to_datetime(inp0.last_review)
inp0.last_review
```

```
0      2018-10-19
1      2019-05-21
2           NaT
3      2019-05-07
4      2018-11-19
...
48890          NaT
48891          NaT
48892          NaT
48893          NaT
48894          NaT
Name: last_review, Length: 48895, dtype: datetime64[ns]
```


4. Data types

4.1 Categorical

```
inp0.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', 'availability_365_categories',  
      'minimum_night_categories', 'number_of_reviews_categories',  
      'price_categories'],  
      dtype='object')
```

```
# Categorical nominal  
categorical_columns = inp0.columns[[0,1,3,4,5,8,16,17,18,19]]  
categorical_columns
```

```
Index(['id', 'name', 'host_name', 'neighbourhood_group', 'neighbourhood',  
      'room_type', 'availability_365_categories', 'minimum_night_categories',  
      'number_of_reviews_categories', 'price_categories'],  
      dtype='object')
```

4.2 Numerical

```
numerical_columns = inp0.columns[[9,10,11,13,14,15]]
numerical_columns
```

```
Index(['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month',  
      'calculated_host_listings_count', 'availability_365'],  
      dtype='object')
```

```
inp0[numerical_columns].describe()
```

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
count	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	48895.000000
mean	152.720687	7.029962	23.274466	1.373221	7.143982	112.781327
std	240.154170	20.510550	44.550582	1.680442	32.952519	131.622289
min	0.000000	1.000000	0.000000	0.010000	1.000000	0.000000
25%	69.000000	1.000000	1.000000	0.190000	1.000000	0.000000
50%	106.000000	3.000000	5.000000	0.720000	1.000000	45.000000
75%	175.000000	5.000000	24.000000	2.020000	2.000000	227.000000
max	10000.000000	1250.000000	629.000000	58.500000	327.000000	365.000000

4.3 Coordinates and date

```
coordinates = inp0.columns[[5,6,12]]
inp0[coordinates]
```

	neighbourhood	latitude	last_review
0	Kensington	40.64749	2018-10-19
1	Midtown	40.75362	2019-05-21
2	Harlem	40.80902	NaT
3	Clinton Hill	40.68514	2019-05-07
4	East Harlem	40.79851	2018-11-19
...
48890	Bedford-Stuyvesant	40.67853	NaT
48891	Bushwick	40.70184	NaT
48892	Harlem	40.81475	NaT
48893	Hell's Kitchen	40.75751	NaT
48894	Hell's Kitchen	40.76404	NaT

48895 rows × 3 columns

5. Missing values

```
# To see the number of missing values  
inp0.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  
availability_365_categories     0  
minimum_night_categories        0  
number_of_reviews_categories    0  
price_categories                0  
dtype: int64
```

- Two columns (last_review , reviews_per_month) has around 20.56% missing values.
name and host_name has 0.3% and 0.4 % missing values

- We need to see if the values are, MCAR: It stands for Missing completely at random.
The reason behind the missing value is not dependent on any other features or if it is
MNAR: It stands for Missing not at random. There is a specific reason behind the missing value.

- There is no dropping or imputation of columns as we are just analyzing the dataset and not making a model. Also most of the features are important for our analysis.

5.1 Missing values Analysis

```
# Selecting the data with missing values for 'last_review' feature
inp1 = inp0.loc[inp0.last_review.isnull(),:]
```

5.2 Missing values Analysis ('neighbourhood_group' feature)

```
# Count of 'neighbourhood_group' with missing values
inp1.groupby('neighbourhood_group').neighbourhood_group.count()
```

```
neighbourhood_group
Bronx                215
Brooklyn             3657
Manhattan            5029
Queens               1092
Staten Island         59
Name: neighbourhood_group, dtype: int64
```

```
# Count of 'neighbourhood_group'
inp0.groupby('neighbourhood_group').neighbourhood_group.count()
```

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

```
(inp1.groupby('neighbourhood_group').neighbourhood_group.count()/inp0.groupby('neighbourhood_group').neighbourhood_group.count())*100
```

```
neighbourhood_group  
Bronx              19.706691  
Brooklyn           18.190410  
Manhattan          23.216841  
Queens             19.272856  
Staten Island      15.817694  
Name: neighbourhood_group, dtype: float64
```

```
((inp1.groupby('neighbourhood_group').neighbourhood_group.count()/inp0.groupby('neighbourhood_group').neighbourhood_group.count())*100).mean()
```

```
19.240898461107257
```

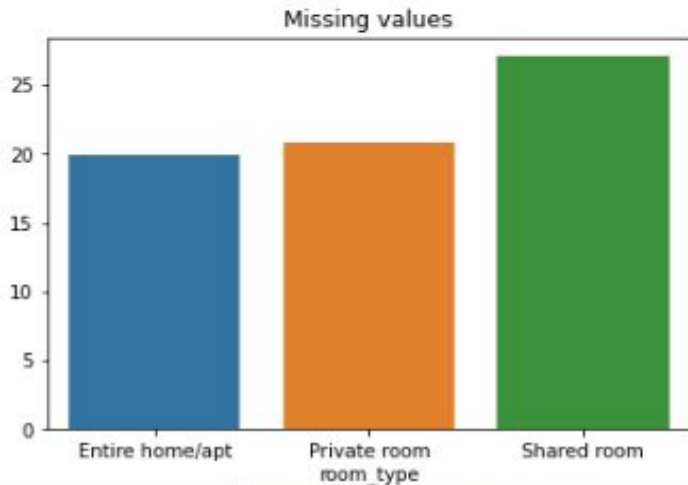
- Each neighbourhood_group has about 19 % missing values in 'last_review' feature.

5.3 Missing values Analysis ('room_type' feature)

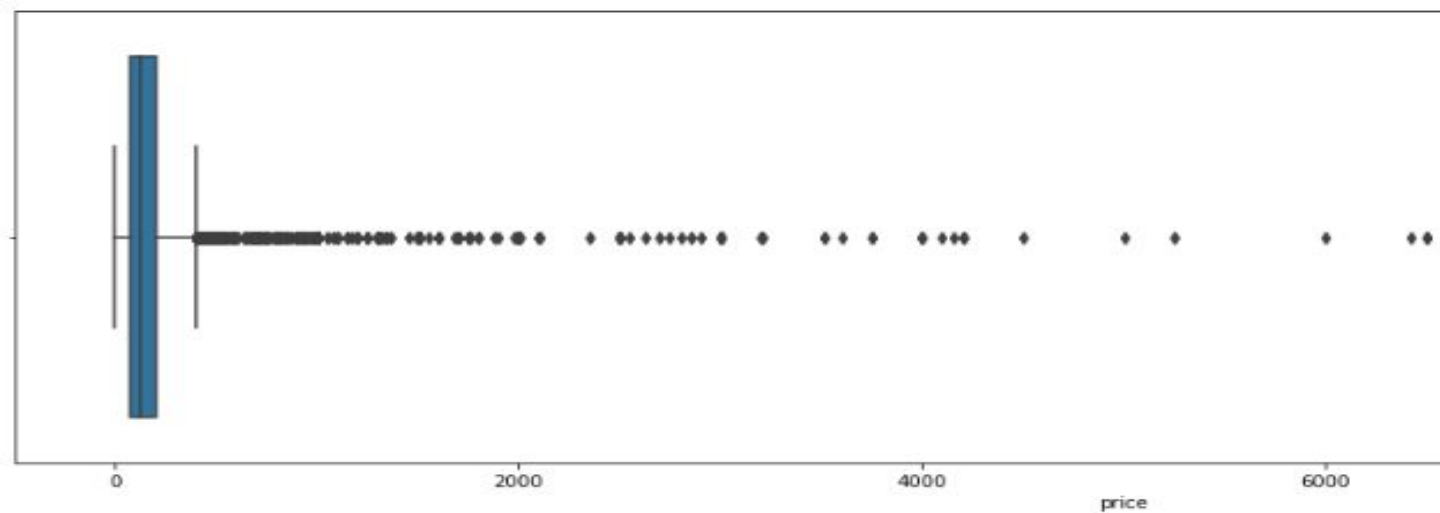
```
# Count of 'room_type' with missing values
inp3 = (inp1.groupby('room_type').room_type.count()/inp0.groupby('room_type').room_type.count())*100
inp3
```

```
room_type
Entire home/apt    19.981109
Private room       20.877004
Shared room        27.068966
Name: room_type, dtype: float64
```

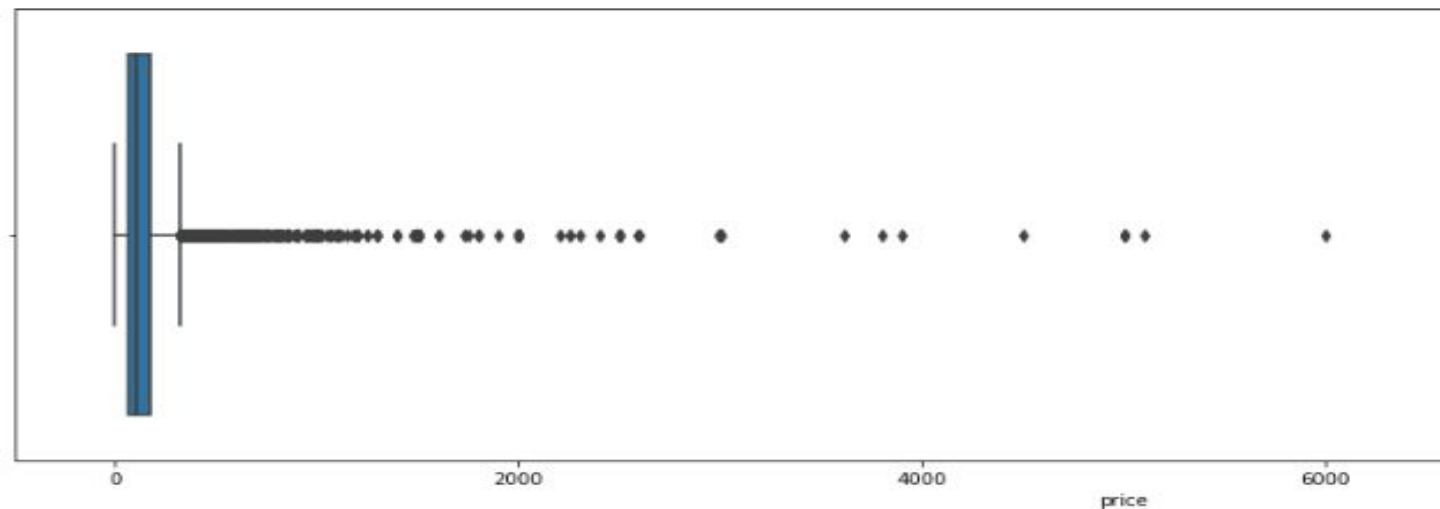
```
plt.title('Missing values')
sns.barplot(x = inp3.index, y = inp3.values)
plt.show()
```



'Shared room' has the highest missing value percentage (27 %) for 'last_review' feature while to other room types has only about 20 %.



- The pricing is higher when 'last_review' feature is missing .



- reviews are less likely to be given for shared rooms

- When the prices are high reviews are less likely to be given

- The above analysis seems to show that the missing values here are not MCAR (missing completely at random)

6. Univariate Analysis

6.1 name

```
inp0.name.value_counts()
```

```
Hillside Hotel          18
Home away from home     17
New york Multi-unit building 16
Brooklyn Apartment      12
Loft Suite @ The Box House Hotel 11
..
Brownstone garden 2 bedroom duplex, Central Park 1
Bright Cozy Private Room near Columbia Univ 1
1 bdrm/large studio in a great location 1
Cozy Private Room #2 Two Beds Near JFK and J Train 1
Trendy duplex in the very heart of Hell's Kitchen 1
Name: name, Length: 47896, dtype: int64
```

6.2 host_id

```
inp0.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
```

6.3 host_name

```
inp0.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

```
inp0.host_name.value_counts().index[:10]
```

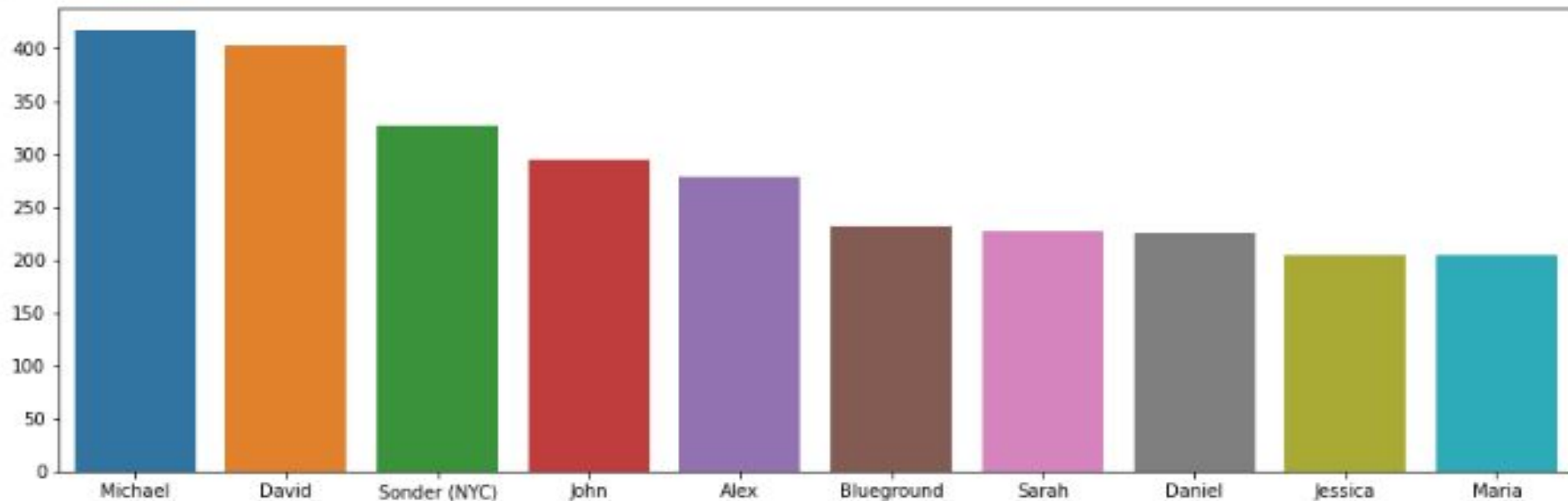
```
Index(['Michael', 'David', 'Sonder (NYC)', 'John', 'Alex', 'Blueground',  
      'Sarah', 'Daniel', 'Jessica', 'Maria'],  
      dtype='object')
```

```
# Top 10 host's
```

```
plt.figure(figsize=(15,5))
```

```
sns.barplot(x = inp0.host_name.value_counts().index[:10] , y = inp0.host_name.value_counts().values[:10])
```

```
plt.show()
```

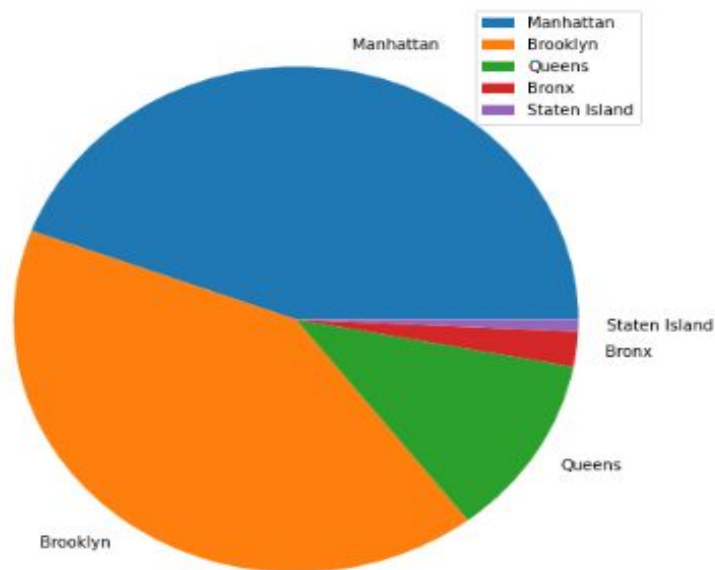


6.4 neighbourhood_group

```
inp0.neighbourhood_group.value_counts()
```

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

```
plt.figure(figsize=(8,8))  
plt.pie(x = inp0.neighbourhood_group.value_counts(normalize= True) * 100, labels = inp0.neighbourhood_group.value_counts(normalize= True).index)  
plt.legend()  
plt.show()
```



What are the neighbourhoods they need to target?

81 % of the listing are Manhattan and Brooklyn neighbourhood_group

6.5 neighbourhood

```
inp0.neighbourhood.value_counts()
```

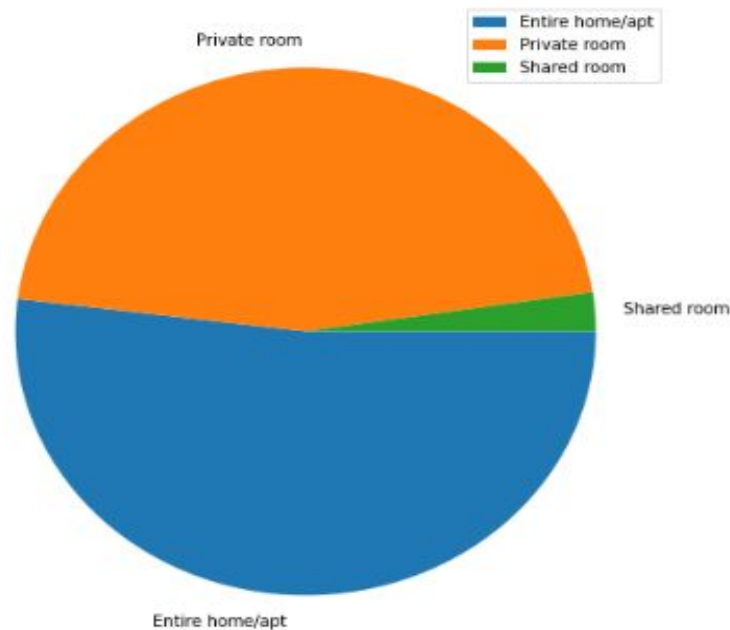
```
Williamsburg      3920
Bedford-Stuyvesant 3714
Harlem            2658
Bushwick          2465
Upper West Side   1971
...
Fort Wadsworth    1
Richmondtown      1
New Dorp          1
Rossville         1
Willowbrook       1
Name: neighbourhood, Length: 221, dtype: int64
```

6.6 room_type

```
inp0.room_type.value_counts()
```

```
Entire home/apt    25409  
Private room       22326  
Shared room        1160  
Name: room_type, dtype: int64
```

```
plt.figure(figsize=(8,8))  
plt.pie(x = inp0.room_type.value_counts(normalize= True) * 100,labels = inp0.room_type.value_counts(normalize= True).index,counter-clock=False)  
plt.legend()  
plt.show()
```



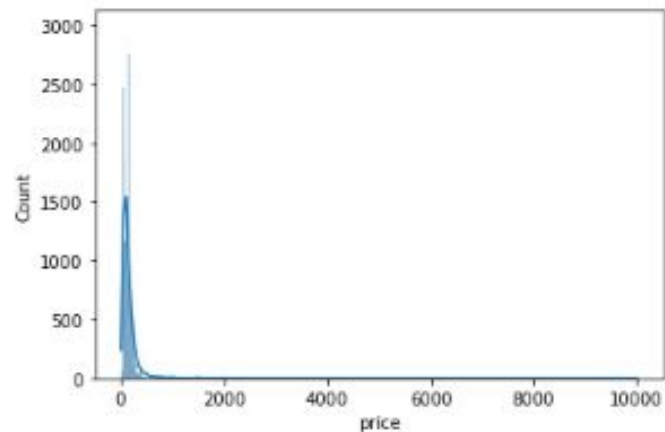
6.7 price

```
inp0.price.value_counts()
```

```
100    2051
150    2047
50     1534
60     1458
200    1401
...
780      1
386      1
888      1
483      1
338      1
Name: price, Length: 674, dtype: int64
```

```
sns.histplot(data = inp0.price,kde = True)
```

```
<AxesSubplot:xlabel='price', ylabel='Count'>
```



6.8 minimum_nights

```
inp0.minimum_nights.value_counts()
```

```
1      12720
2      11696
3       7999
30      3760
4       3303
```

```
...
```

```
186      1
366      1
68       1
87       1
36       1
```

```
Name: minimum_nights, Length: 109, dtype: int64
```

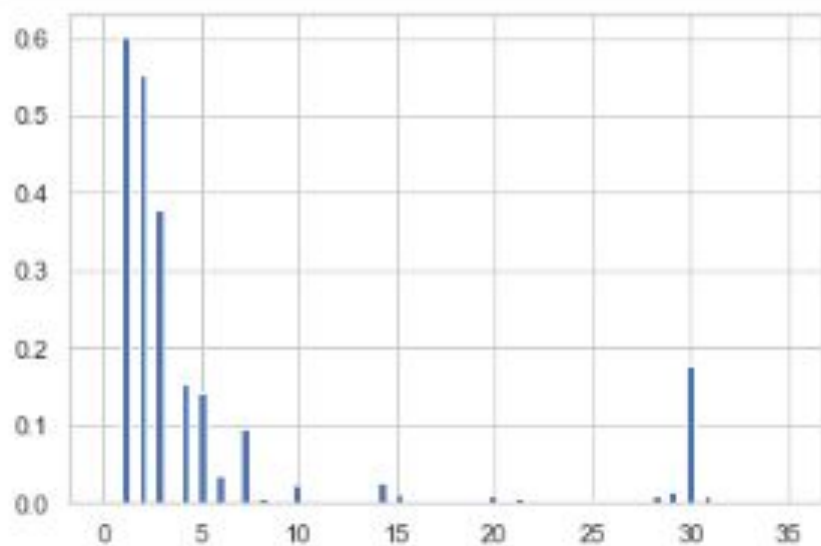
```
inp0.minimum_nights.describe()
```

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

```
Name: minimum_nights, dtype: float64
```



```
plt.hist(data = inp0, x = 'minimum_nights',bins=80,range=(0,35),density=True)  
plt.show()
```

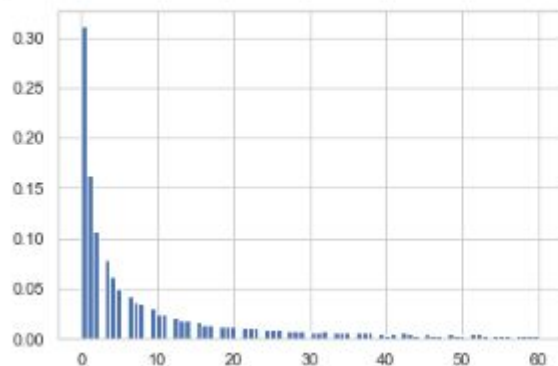


6.9 number_of_reviews

```
inp0.number_of_reviews.describe()
```

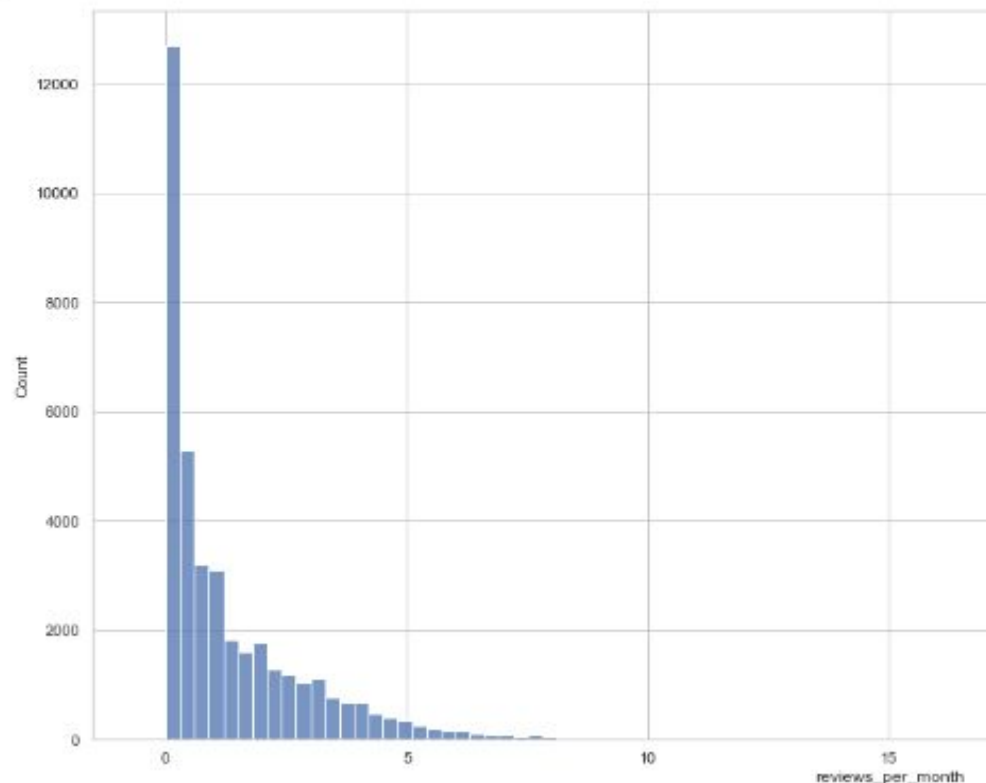
```
count      48895.000000
mean         23.274466
std          44.550582
min           0.000000
25%           1.000000
50%           5.000000
75%          24.000000
max          629.000000
Name: number_of_reviews, dtype: float64
```

```
plt.hist(data = inp0, x = 'number_of_reviews', bins=80, range=(0,60), density=True)
plt.show()
```



6.10 reviews_per_month

```
plt.figure(figsize = (20,10))  
sns.histplot(data = inp0, x = 'reviews_per_month',bins=100,binrange=(0,30))  
plt.show()
```



```
inp0.reviews_per_month.describe()
```

```
count    38843.000000  
mean      1.373221  
std       1.680442  
min       0.010000  
25%       0.190000  
50%       0.720000  
75%       2.020000  
max       58.500000  
Name: reviews_per_month, dtype: float64
```

6.11 calculated_host_listings_count

```
inp0.calculated_host_listings_count.describe()
```

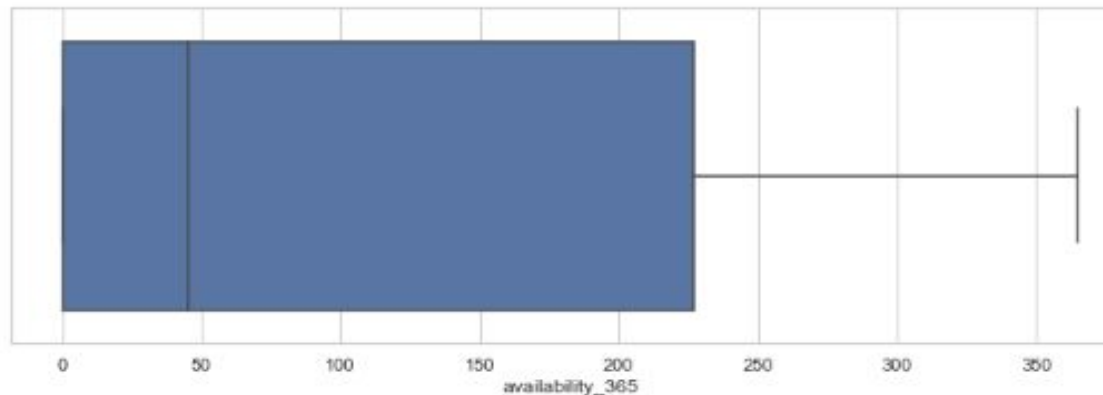
```
count    48895.000000
mean         7.143982
std        32.952519
min         1.000000
25%         1.000000
50%         1.000000
75%         2.000000
max        327.000000
Name: calculated_host_listings_count, dtype: float64
```

6.12 availability_365

```
inp0.availability_365.describe()
```

```
count    48895.000000
mean      112.781327
std       131.622289
min        0.000000
25%        0.000000
50%       45.000000
75%      227.000000
max      365.000000
Name: availability_365, dtype: float64
```

```
plt.figure(figsize = (12,4))
sns.boxplot(data = inp0 , x = 'availability_365')
plt.show()
```



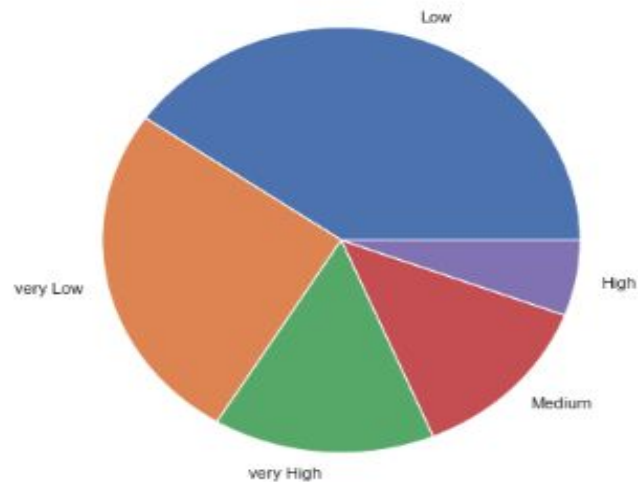
6.13 minimum_night_categories

```
inp0.minimum_night_categories.value_counts(normalize=True)*100
```

```
Low          40.280192
very Low     26.014930
very High    14.997444
Medium       12.960425
High         5.747009
Name: minimum_night_categories, dtype: float64
```

```
plt.figure(figsize=(12,7))
plt.title('Minimum night categories', fontdict={'fontsize': 20})
plt.pie(x = inp0.minimum_night_categories.value_counts(),labels=inp0.minimum_night_categories.value_counts().index)
plt.show()
```

Minimum night categories

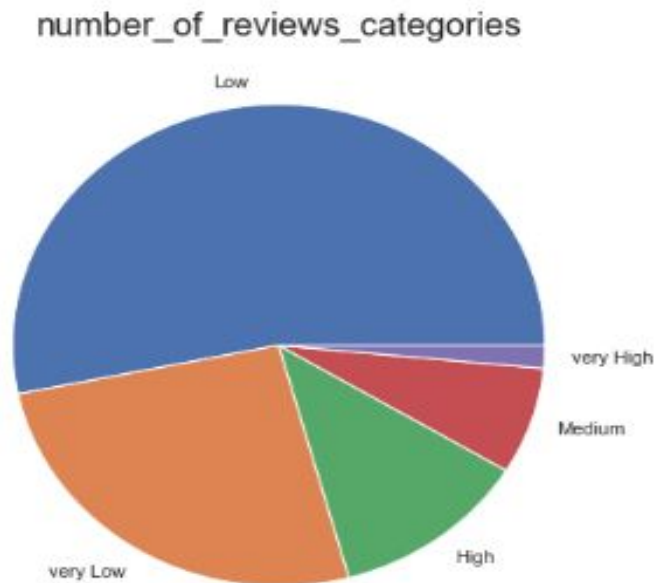


6.14 number_of_reviews_categories

```
inp0.number_of_reviews_categories.value_counts(normalize=True)*100
```

```
Low          53.240618
very Low     26.014930
High         12.052357
Medium        7.164332
very High     1.527764
Name: number_of_reviews_categories, dtype: float64
```

```
plt.figure(figsize=(12,7))
plt.title('number_of_reviews_categories', fontdict={'fontsize': 20})
plt.pie(x = inp0.number_of_reviews_categories.value_counts(),labels=inp0.number_of_reviews_categories.value_counts().index)
plt.show()
```

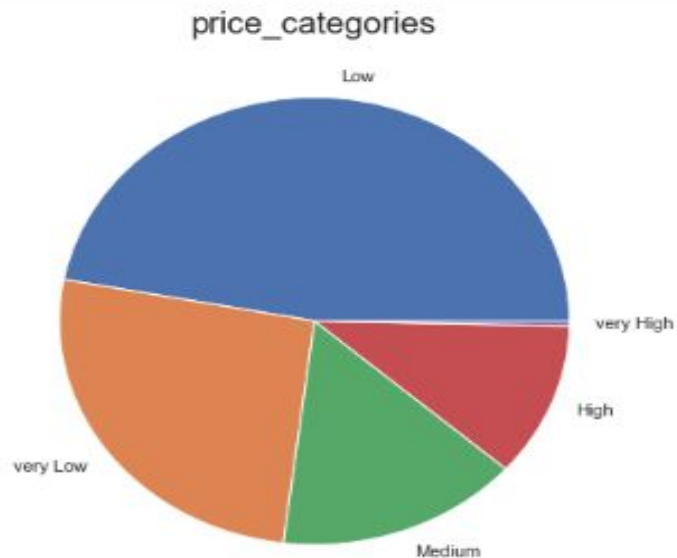


6.15 price_categories

```
inp0['price_categories'].value_counts()
```

```
Low          22998
very Low     12720
Medium       7556
High         5447
very High     174
Name: price_categories, dtype: int64
```

```
plt.figure(figsize=(12,7))
plt.title('price_categories', fontdict={'fontsize': 20})
plt.pie(x = inp0.price_categories.value_counts(), labels=inp0.price_categories.value_counts().index,)
plt.show()
```



What is the pricing ranges preferred by customers?

'Low' price ranges are preferred by customers followed by very 'Low' price ranges.

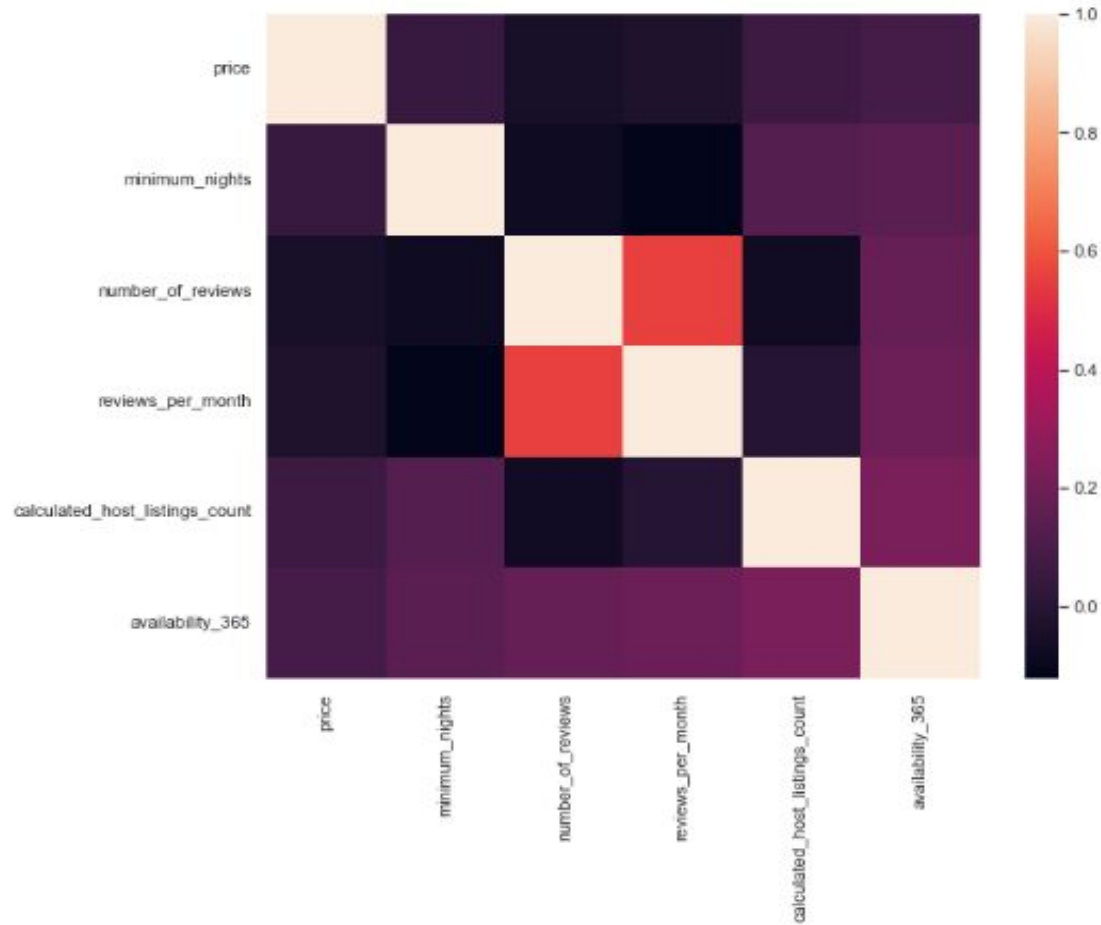
7. Bivariate and Multivariate Analysis

7.1 Finding the correlations

```
inp0[numerical_columns].corr()
```

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
price	1.000000	0.042799	-0.047954	-0.030608	0.057472	0.081829
minimum_nights	0.042799	1.000000	-0.080116	-0.121702	0.127960	0.144303
number_of_reviews	-0.047954	-0.080116	1.000000	0.549868	-0.072376	0.172028
reviews_per_month	-0.030608	-0.121702	0.549868	1.000000	-0.009421	0.185791
calculated_host_listings_count	0.057472	0.127960	-0.072376	-0.009421	1.000000	0.225701
availability_365	0.081829	0.144303	0.172028	0.185791	0.225701	1.000000

```
plt.figure(figsize=(10,8))
sns.heatmap(data = inp0[numerical_columns].corr())
plt.show()
```



7.2 Finding Top correlations

```
corr_matrix = inp0[numerical_columns].corr().abs()

#the matrix is symmetric so we need to extract upper triangle matrix without diagonal (k = 1)

sol = (corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
        .stack()
        .sort_values(ascending=False))
```

corr_matrix

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
price	1.000000	0.042799	0.047954	0.030608	0.057472	0.081829
minimum_nights	0.042799	1.000000	0.080116	0.121702	0.127960	0.144303
number_of_reviews	0.047954	0.080116	1.000000	0.549868	0.072376	0.172028
reviews_per_month	0.030608	0.121702	0.549868	1.000000	0.009421	0.185791
calculated_host_listings_count	0.057472	0.127960	0.072376	0.009421	1.000000	0.225701
availability_365	0.081829	0.144303	0.172028	0.185791	0.225701	1.000000

```
# Top meaningful correlations  
sol[1:8]
```

calculated_host_listings_count	availability_365	0.225701
reviews_per_month	availability_365	0.185791
number_of_reviews	availability_365	0.172028
minimum_nights	availability_365	0.144303
	calculated_host_listings_count	0.127960
	reviews_per_month	0.121702
price	availability_365	0.081829

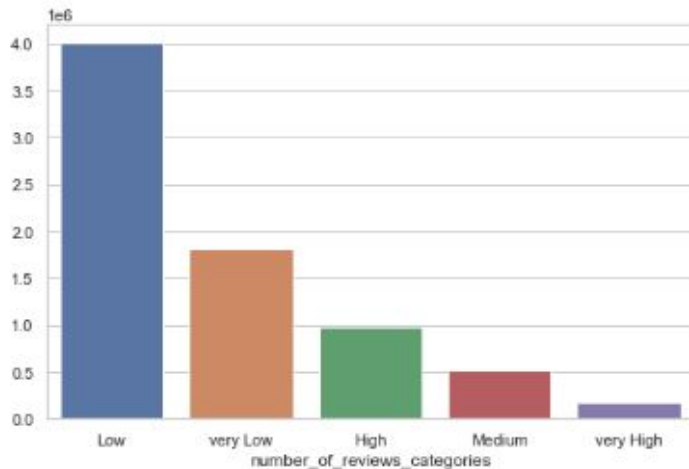
dtype: float64

7.3 number_of_reviews_categories and prices

```
# prices for each of reviews_categories
x1 = inp0.groupby('number_of_reviews_categories').price.sum().sort_values(ascending = False)
x1
```

```
number_of_reviews_categories
Low          4002323
very Low     1806531
High         971346
Medium       508647
very High    178431
Name: price, dtype: int64
```

```
plt.figure(figsize=(8,5))
sns.barplot(x = x1.index,y = x1.values)
plt.show()
```



What is the pricing ranges preferred by customers?

The total price for 'Low' or 'very Low' number_of_reviews_categories are high.

.4 ('room_type' and 'number_of_reviews_categories')

```
inp0.room_type.value_counts()
```

```
Entire home/apt    25409  
Private room       22326  
Shared room        1160  
Name: room_type, dtype: int64
```

```
pd.crosstab(inp0['room_type'], inp0['number_of_reviews_categories'])
```

number_of_reviews_categories	High	Low	Medium	very High	very Low
room_type					
Entire home/apt	3809	14909	1960	504	4227
Private room	1950	10769	1494	226	7887
Shared room	134	354	49	17	606

The various kinds of properties that exist w.r.t. customer preferences.?

Entire home/apt have more reviews than Shared rooms

'Shared room' are less likely to give reviews. only 16 %

7.5 'room_type' and 'price_categories'

```
pd.crosstab(inp0['room_type'], inp0['price_categories'])
```

price_categories	High	Low	Medium	very High	very Low
room_type					
Entire home/apt	3714	13086	4262	120	4227
Private room	1620	9597	3170	52	7887
Shared room	113	315	124	2	606

7.6 'room_type' and 'reviews_per_month'

```
inp0.room_type.value_counts()
```

```
Entire home/apt    25409
Private room       22326
Shared room        1160
Name: room_type, dtype: int64
```

```
inp0.groupby('room_type').reviews_per_month.mean()
```

```
room_type
Entire home/apt    1.306578
Private room       1.445209
Shared room        1.471726
Name: reviews_per_month, dtype: float64
```

```
inp0.groupby('room_type').reviews_per_month.median()
```

```
room_type
Entire home/apt    0.66
Private room       0.77
Shared room        0.98
Name: reviews_per_month, dtype: float64
```

For each 'room_type' there are ~1.4 reviews per month on average.

7.7 minimum_night_categories and reviews_per_month

```
inp0.groupby('minimum_night_categories').reviews_per_month.sum().sort_values()
```

```
minimum_night_categories
High          1227.57
very High     2235.19
Medium        4689.73
very Low     20395.49
Low          24792.06
Name: reviews_per_month, dtype: float64
```

Customers are more likely to leave reviews for low number of minimum nights

Adjustments in the existing properties to make it more customer-oriented. ?

minimum_nights should be on the lower side to make properties more customer-oriented

7.8 'availability_365_categories', 'price_categories' and 'reviews_per_month'

```
inp0.availability_365_categories.value_counts()
```

```
very Low    17941
Low         11829
very High   8108
Medium      5792
High        5225
Name: availability_365_categories, dtype: int64
```

If the combination of availability and price is very high, reviews_per_month will be low on average.

Very high availability and very low price are likely to get more reviews.

		reviews per month
availability_365_categories	price_categories	
High	High	0.598431
	Low	2.200373
	Medium	1.056111
	very High	0.342308
	very Low	3.289381
Low	High	0.638307
	Low	1.783956
	Medium	0.883844
	very High	0.803750
	very Low	2.896114
Medium	High	0.591070
	Low	1.993565
	Medium	1.157492
	very High	0.517500
	very Low	2.893918
very High	High	0.428464
	Low	1.490562
	Medium	0.694283
	very High	0.276571
	very Low	2.206077
very Low	High	0.337780
	Low	0.506051
	Medium	0.276970
	very High	0.480588
	very Low	0.673759