UNVEILING THE PROBLEMS OF AIRBNB IN NYC

Importing libraries and reading the data

```
# Importing Necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
# Read and understand the dataset and check the first five rows
Airbnb_data = pd.read_csv('AB_NYC_2019.csv')
Airbnb_data.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_revie
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	2
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	

1. Creating features

By categorizing, understanding the relationships and the connections between things improves and findings can be communicated in the better way.

1.1 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'
```

1.2 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'
```

1.3 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'
```

1.4 categorizing the "price" column into 5 categories

```
Airbnb_data.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
```

2.Fixing columns

```
# Check the datatypes of all the columns of the dataframe after categorizing the columns in data
Airbnb_data.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
```

- reviews_per_month column is of object Dtype. datetime64 is a better Data type for this column.

```
Airbnb_data.last_review = pd.to_datetime(Airbnb_data.last_review)
Airbnb_data.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]
```

- There are no more Data types to be fixed and data does not contain inconsistencies such as shifted columns, which is need to align correctly. The columns necessery for the futher analysis are also derived.

3. Data types

```
# printing all the columns in the dataset
print(Airbnb_data.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')
```

3.1 Categorical

```
# Categorical nominal
categorical_columns = Airbnb_data.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')
```

3.2 Numerical

```
numerical_columns = Airbnb_data.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')
```

```
Airbnb_data[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

3.3 Coordinates and date

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

4. Missing value Treatment

- In Data cleaning the first step is to check the missing values
- Check the number of null (missing) values in the columns
- Missing value means that values is not present in the data

```
# To see the sum of missing values for each column
Airbnb_data.isnull().mean()*100
```

```
id                                0.000000
name                              0.032723
host_id                           0.000000
host_name                         0.042949
neighbourhood_group               0.000000
neighbourhood                     0.000000
latitude                          0.000000
longitude                         0.000000
room_type                         0.000000
price                             0.000000
minimum_nights                    0.000000
number_of_reviews                 0.000000
last_review                      20.558339
reviews_per_month                20.558339
calculated_host_listings_count    0.000000
availability_365                  0.000000
availability_365_categories       0.000000
minimum_night_categories          0.000000
number_of_reviews_categories      0.000000
price_categories                  0.000000
dtype: float64
```

- Two columns (last_review , reviews_per_month) has around 20.56% missing values. name and host_name has 0.03% and 0.04% 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.

4.1 Missing values Analysis

```
# Selecting the data with no missing values for 'last_review' feature
Airbnb_data1 = Airbnb_data[~Airbnb_data.last_review.isnull()]
Airbnb_data1.head()
```

```
# Count of 'neighbourhood_group' with missing values
Airbnb_data.neighbourhood_group.value_counts(dropna=False)
```

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

```
# Count of 'neighbourhood_group'
Airbnb_data1.neighbourhood_group.value_counts(dropna=False)
```

```
Manhattan        16621
Brooklyn         16439
Queens            4572
Bronx              875
Staten Island      314
Name: neighbourhood_group, dtype: int64
```

```python
#Checking missing values percentage of each neighbourhood
1-Airbnb_data1.neighbourhood_group.value_counts(dropna=False)/Airbnb_data.neighbourhood_group.value_counts(dropna=False)
```

```
Manhattan        0.232676
Brooklyn         0.182302
Queens           0.193082
Bronx            0.197984
Staten Island    0.158177
Name: neighbourhood_group, dtype: float64
```

```python
(1-Airbnb_data1.neighbourhood_group.value_counts(dropna=False)/Airbnb_data.neighbourhood_group.value_counts(dropna=False)).mean()
```

```
0.19284405038537897
```

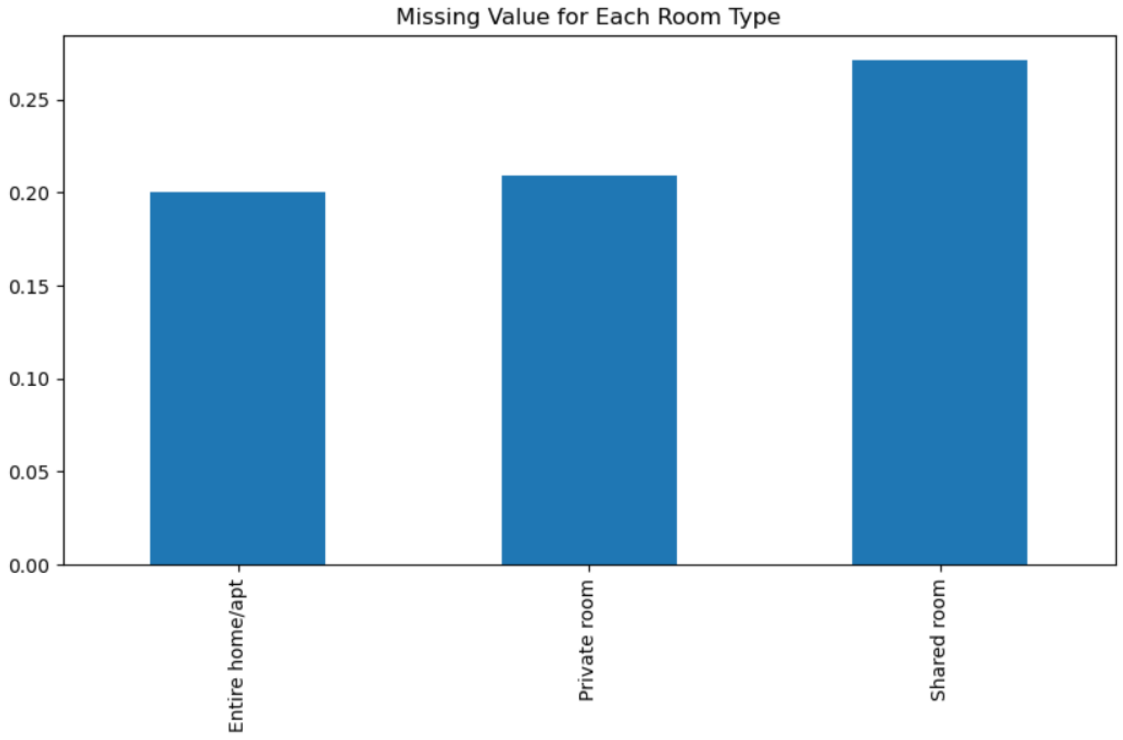
Insights:

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

```
# Count of 'room_type' with missing values
(1-Airbnb_data1.room_type.value_counts(dropna=False)/Airbnb_data.room_type.value_counts(dropna=False))*100
```

```
Entire home/apt    20.024401
Private room       20.926274
Shared room        27.068966
Name: room_type, dtype: float64
```

```
plt.figure(figsize=(10,5))
plt.title("Missing Value for Each Room Type")
(1-Airbnb_data1.room_type.value_counts(dropna=False)/Airbnb_data.room_type.value_counts(dropna=False)).plot.bar()
plt.show()
```



Insights:

- The Each neighbourhood_group has about 22 % missing values in 'last_review' feature.

```python
print('Mean and Median of prices with last_review feature missing')
print('Mean   = ', Airbnb_data[Airbnb_data['last_review'].isnull()].price.mean())
print('Median = ', Airbnb_data[Airbnb_data['last_review'].isnull()].price.median())

print('\nMean and Median of prices with last_review feature not missing')
print('Mean   = ', Airbnb_data[Airbnb_data['last_review'].notnull()].price.mean())
print('Median = ', Airbnb_data[Airbnb_data['last_review'].notnull()].price.median())
```

```
Mean and Median of prices with last_review feature missing
Mean   =  192.9190210903303
Median =  120.0

Mean and Median of prices with last_review feature not missing
Mean   =  142.317946605566
Median =  101.0
```

Insights:

- 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)

5.Univariate Analysis

5.1 host_id

```
Airbnb_data1.host_id.value_counts()
```

```
219517861    207
61391963      79
16098958      61
137358866     51
7503643       49
            ...
6389984        1
68684053       1
60077920       1
9997184        1
74162901       1
Name: host_id, Length: 30232, dtype: int64
```

5.2 name

```
Airbnb_data1.name.value_counts()
```

```
Home away from home                         12
Loft Suite @ The Box House Hotel            11
Private Room                                10
#NAME?                                      10
Brooklyn Apartment                           9
                                            ..
Sunny light filled comfy bedroom             1
Astoria living, like a true NY'er            1
Sunny light filled bedroom                   1
Magical 1 BR in Crown Heights                1
Cozy Private Room in Bushwick, Brooklyn      1
Name: name, Length: 38244, dtype: int64
```

5.3 host_name

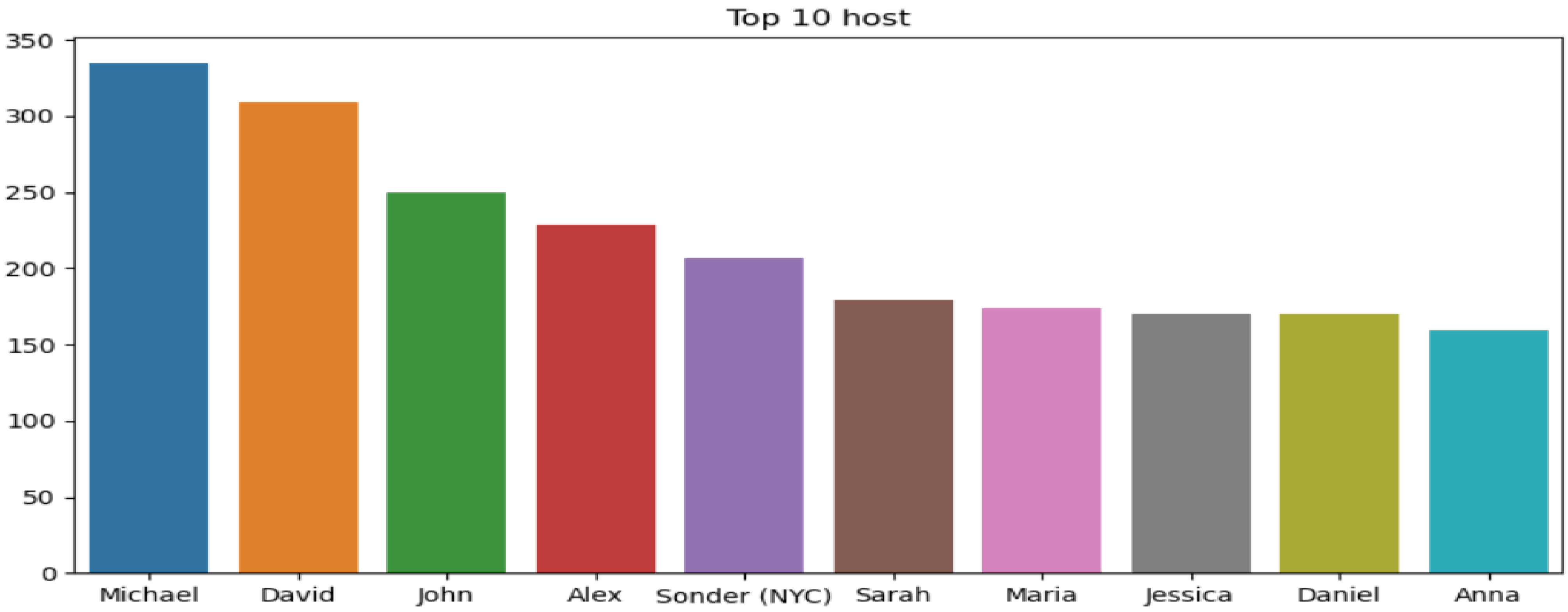
```
Airbnb_data1.host_name.value_counts()
```

```
Michael            335
David              309
John               250
Alex               229
Sonder (NYC)       207
                  ...
Krisztián            1
Kila                 1
Maisha               1
Martin & Hande       1
Rusaa                1
Name: host_name, Length: 9885, dtype: int64
```

```
Airbnb_data1.host_name.value_counts().index[:10]
```

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

```
# Top 10 host's
plt.figure(figsize=(10,5))
plt.title("Top 10 host")
sns.barplot(x = Airbnb_data1.host_name.value_counts().index[:10] , y = Airbnb_data1.host_name.value_counts().values[:10])
plt.show()
```

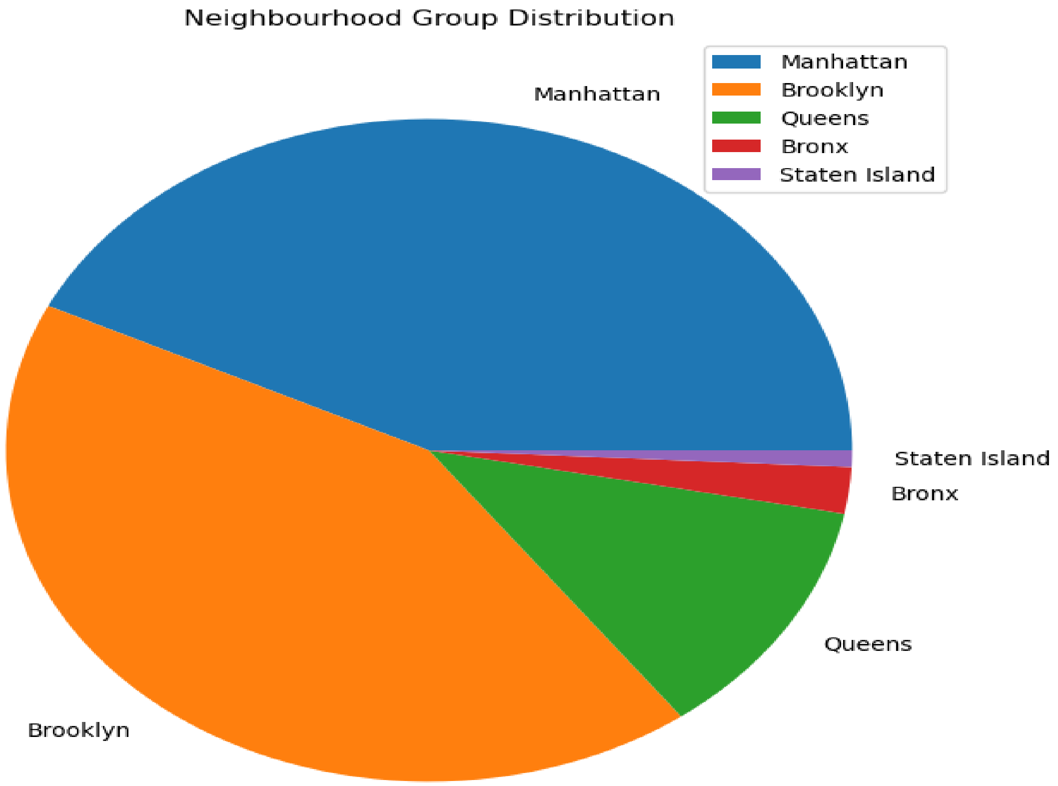


5.4 neighbourhood_group

```
Airbnb_data1.neighbourhood_group.value_counts(normalize=True)*100
```

```
Manhattan        42.814456
Brooklyn         42.345638
Queens           11.777131
Bronx             2.253935
Staten Island     0.808841
Name: neighbourhood_group, dtype: float64
```

```
plt.figure(figsize=(8,8))
plt.title("Neighbourhood Group Distribution")
plt.pie(x = Airbnb_data1.neighbourhood_group.value_counts(normalize= True) * 100, labels = Airbnb_data1.neighbourhood_group.value
plt.legend()
plt.show()
```



Insights:

- What are the neighbourhoods they need to target?
- 81 % of the listing are Manhattan and Brooklyn neighbourhood_group

5.5 neighbourhood

```
Airbnb_data1.neighbourhood.value_counts()
```

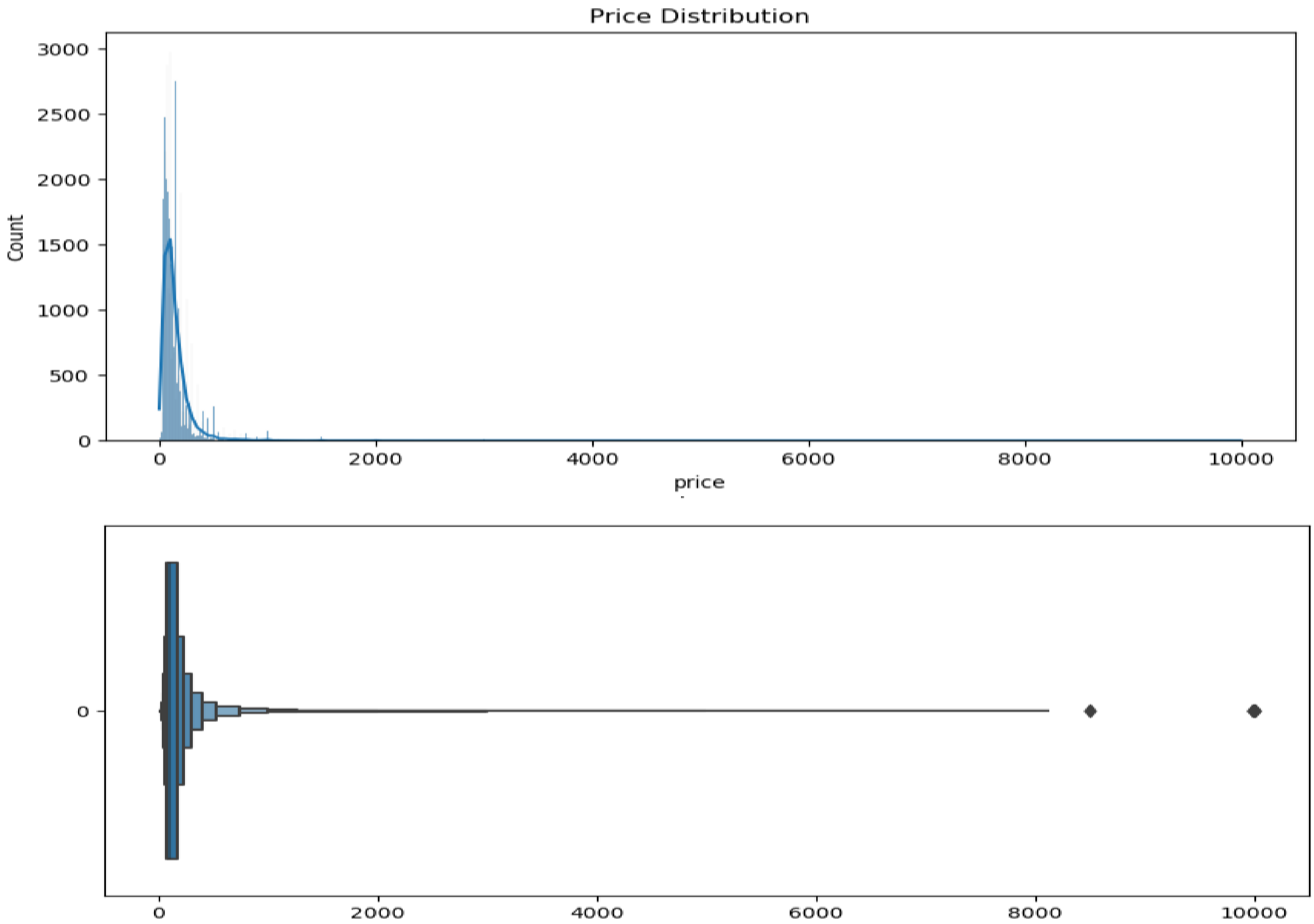
```
Williamsburg          3163
Bedford-Stuyvesant    3141
Harlem                2204
Bushwick              1942
Hell's Kitchen        1528
                      ... 
Holliswood               2
New Dorp Beach           2
Richmondtown             1
Rossville                1
Willowbrook              1
Name: neighbourhood, Length: 218, dtype: int64
```

5.6 Price

```
Airbnb_data1.price.value_counts()
```

```
150     1596
100     1517
50      1188
60      1155
75      1095
        ...
578        1
789        1
1795       1
1095       1
323        1
Name: price, Length: 581, dtype: int64
```

```
plt.figure(figsize=(10,10))
plt.subplot(2, 1, 1)
plt.title("Price Distribution")
sns.histplot(data = Airbnb_data.price,kde = True)
plt.subplot(2, 1, 2)
sns.boxenplot(data = Airbnb_data1.price,
    orient ="h")
plt.show()
```

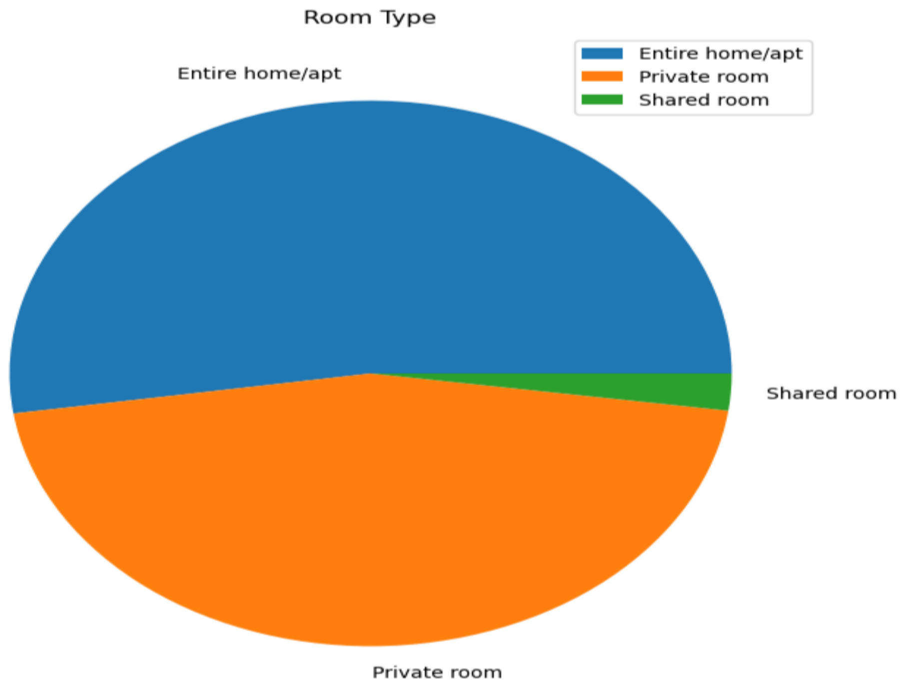

Price Distribution
3000
2500
2000
1500
1000
500
0
Count
0
2000
4000
6000
8000
10000
price
0
0
2000
4000
6000
8000
10000

5.7 room_type

```
Airbnb_data1.room_type.value_counts(normalize=True)
```

```
Entire home/apt    0.523454
Private room       0.454754
Shared room        0.021792
Name: room_type, dtype: float64
```

```
plt.figure(figsize=(8,8))
plt.title("Room Type")
plt.pie(x = Airbnb_data1.room_type.value_counts(normalize=True),
labels = Airbnb_data.room_type.value_counts(normalize= True).index)
plt.legend()
plt.show()
```



5.8 minimum_nights

```
Airbnb_data1.minimum_nights.value_counts()
```

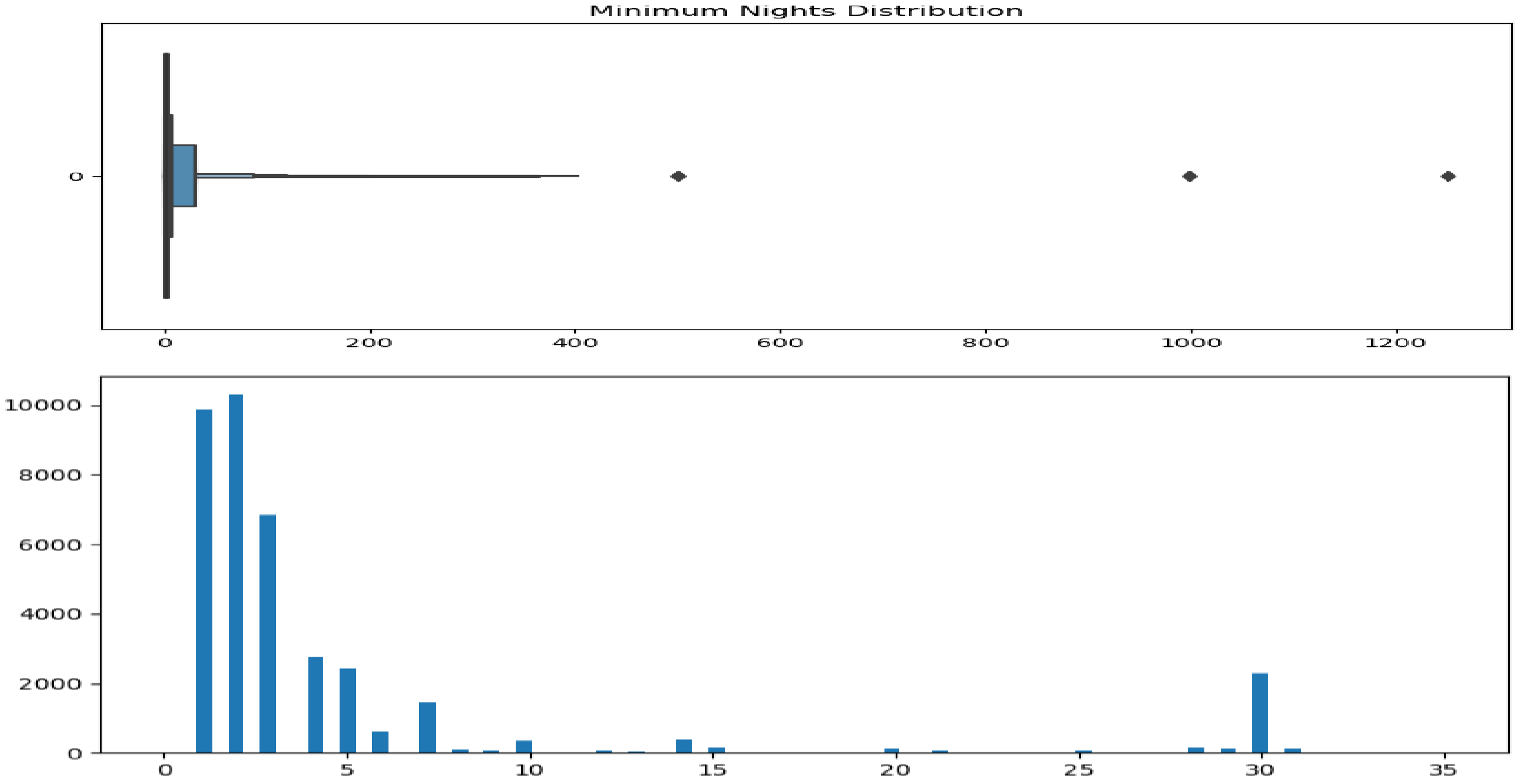
```
2      10300
1       9878
3       6840
4       2749
5       2425
       ...
181        1
122        1
65         1
44         1
210        1
Name: minimum_nights, Length: 89, dtype: int64
```

```
Airbnb_data1.minimum_nights.describe()
```

```
count    38821.000000
mean         5.869220
std         17.389026
min          1.000000
25%          1.000000
50%          2.000000
75%          4.000000
max       1250.000000
Name: minimum_nights, dtype: float64
```

```
plt.figure(figsize=(10,10))

plt.subplot(2,1,1)
plt.title("Minimum Nights Distribution")
sns.boxenplot(data = Airbnb_data1.minimum_nights, orient="h")
plt.subplot(2,1,2)
plt.hist(data = Airbnb_data1, x = 'minimum_nights', bins = 80,range=(0,35) )
plt.show()
```

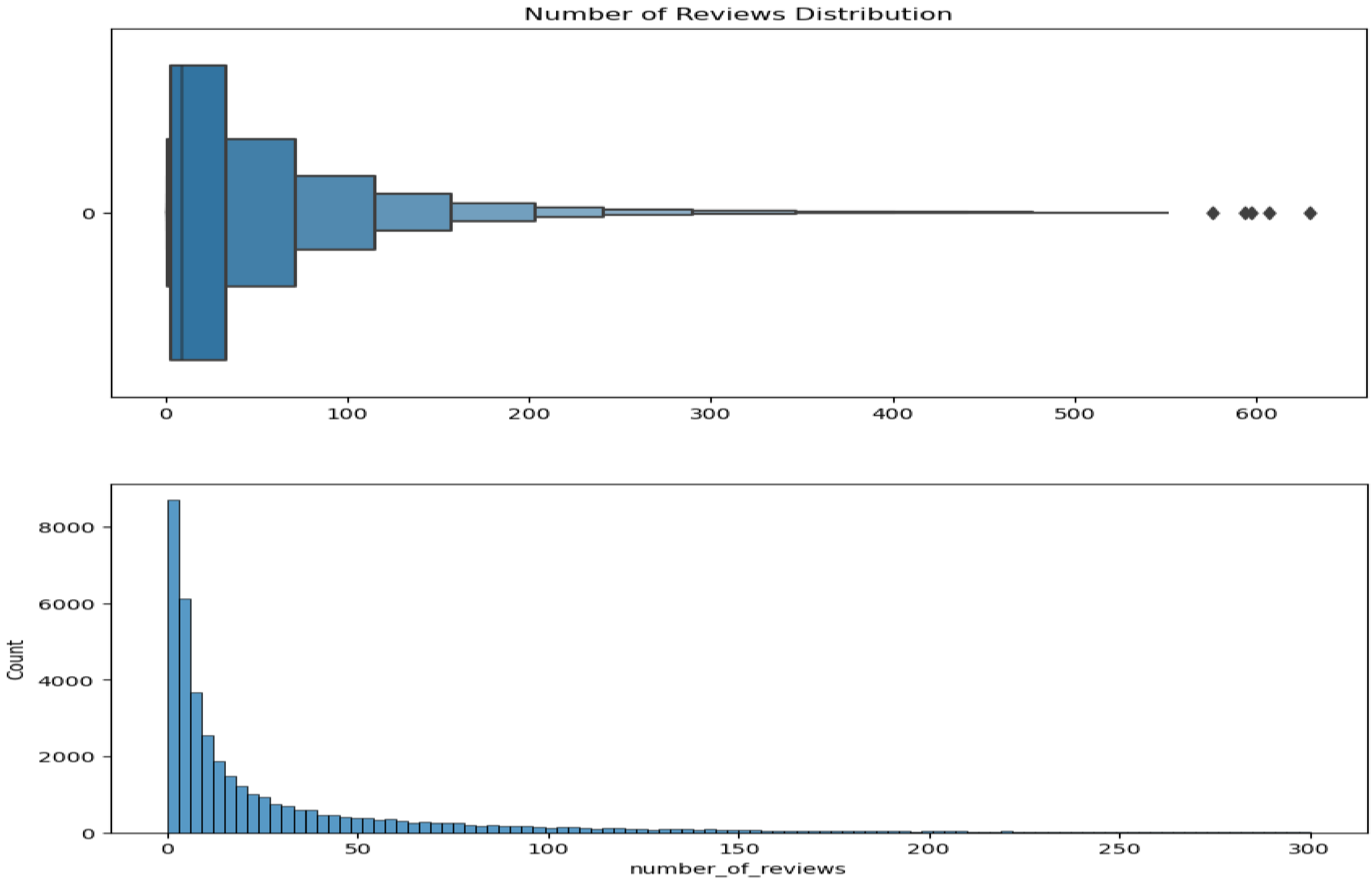


5.9 number_of_reviews

```
Airbnb_data1.number_of_reviews.describe()
```

```
count    38821.000000
mean        29.290255
std         48.182900
min          1.000000
25%          3.000000
50%          9.000000
75%         33.000000
max        629.000000
Name: number_of_reviews, dtype: float64
```

```
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.title("Number of Reviews Distribution")
sns.boxenplot(data = Airbnb_data1.number_of_reviews,orient="h")
plt.subplot(2,1,2)
sns.histplot(data = Airbnb_data1, x = 'number_of_reviews',bins=100,binrange=(0,300))
plt.show()
```

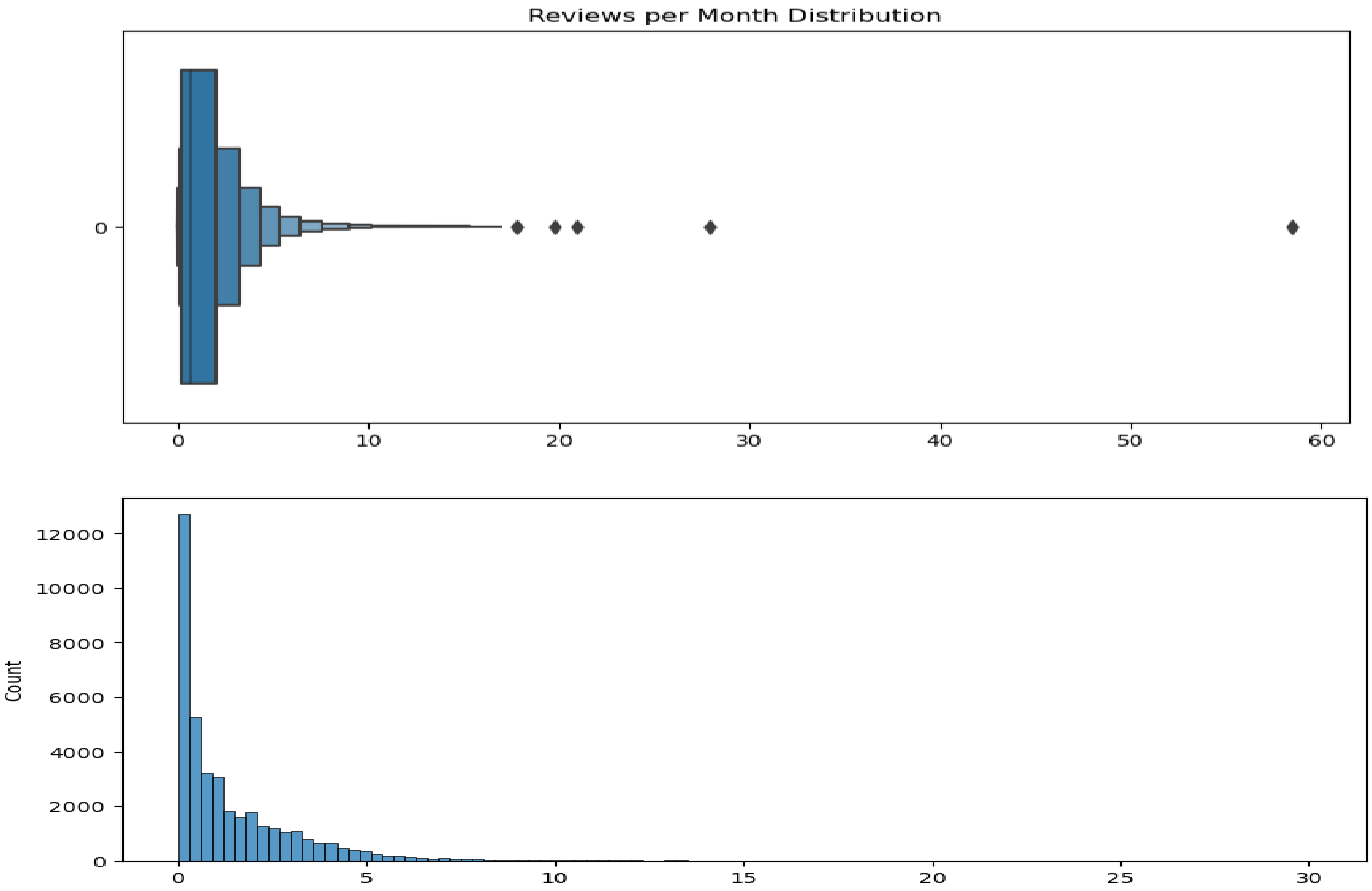

Number of Reviews Distribution
0
0
100
200
300
400
500
600
8000
6000
4000
2000
0
Count
0
50
100
150
200
250
300
number_of_reviews

5.10 reviews_per_month

```
Airbnb_data1.reviews_per_month.describe()
```

```
count    38821.000000
mean         1.373229
std          1.680328
min          0.010000
25%          0.190000
50%          0.720000
75%          2.020000
max         58.500000
Name: reviews_per_month, dtype: float64
```

```
plt.figure(figsize=(10,10))
plt.subplot(2,1,1)
plt.title("Reviews per Month Distribution")
sns.boxenplot(data = Airbnb_data1.reviews_per_month,orient="h")
plt.subplot(2,1,2)
sns.histplot(data = Airbnb_data1, x = 'reviews_per_month',bins=100,binrange=(0,30))
plt.show()
```

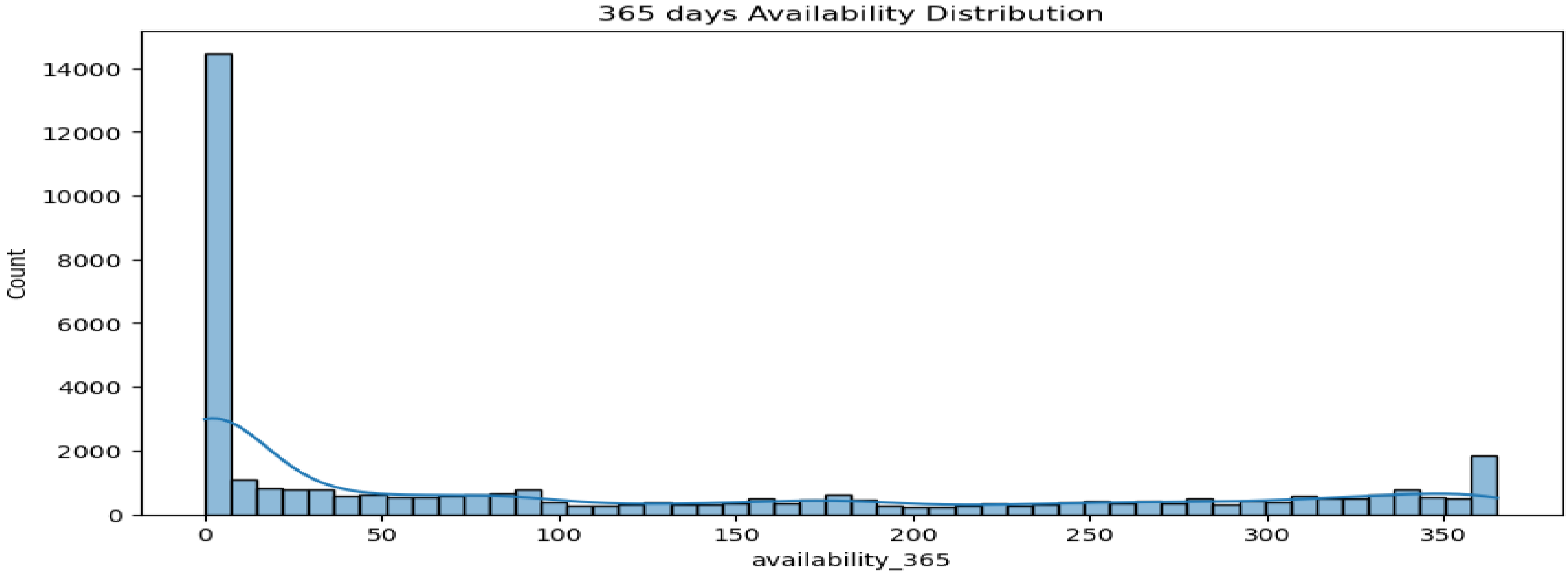

Reviews per Month Distribution
Count

5.11 availability_365

```
Airbnb_data1.availability_365.describe()
```

```
count    38821.000000
mean       114.886299
std        129.529950
min          0.000000
25%          0.000000
50%         55.000000
75%        229.000000
max        365.000000
Name: availability_365, dtype: float64
```

```
plt.figure(figsize = (10,5))
plt.title("365 days Availability Distribution")
sns.histplot(data = Airbnb_data1, x = 'availability_365',bins=50,binrange=(0,365), kde=True)
plt.show()
```

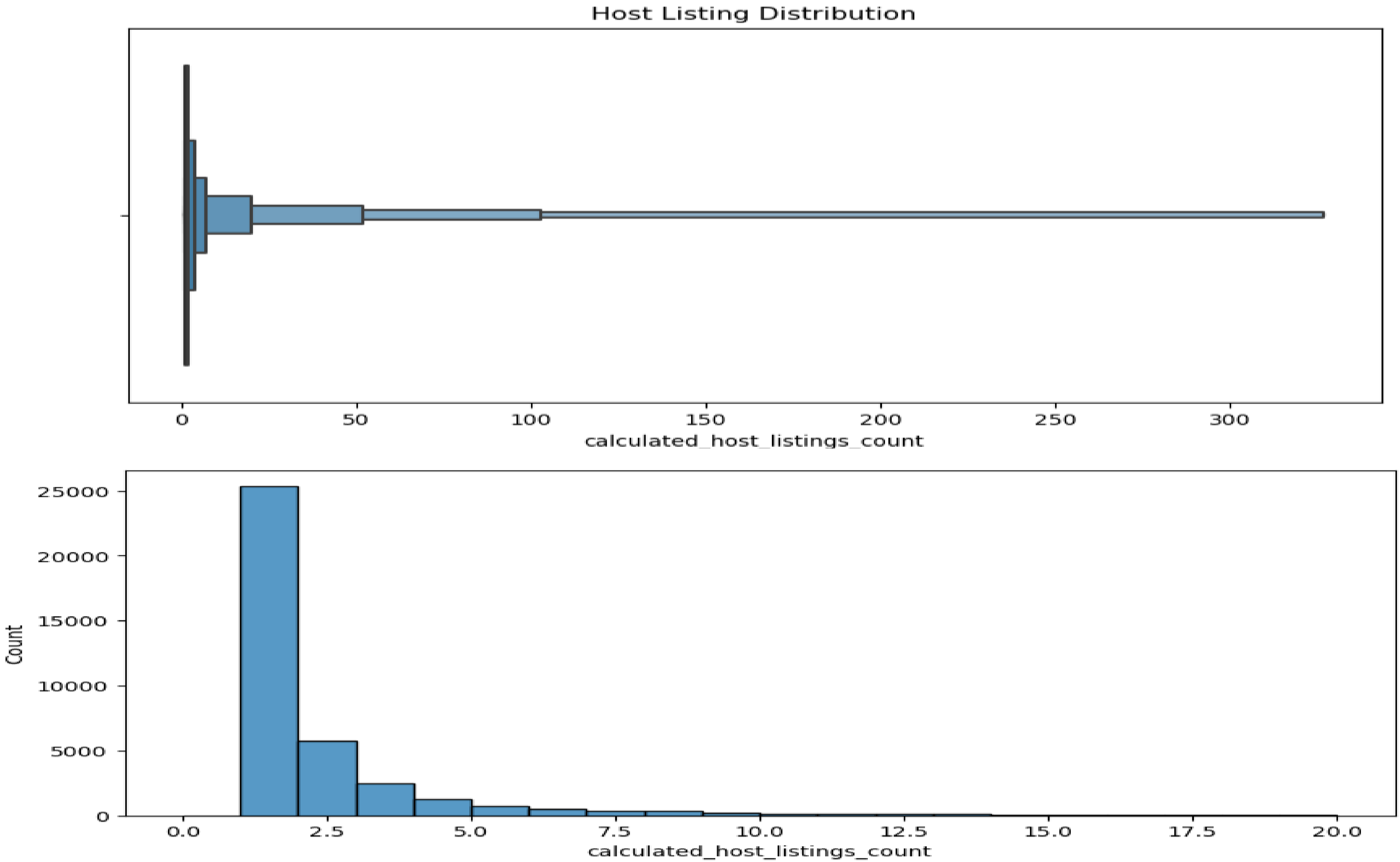


5.12 calculated_host_listings_count

```
Airbnb_data1.calculated_host_listings_count.describe()
```

```
count    38821.000000
mean         5.166611
std         26.302954
min          1.000000
25%          1.000000
50%          1.000000
75%          2.000000
max        327.000000
Name: calculated_host_listings_count, dtype: float64
```

```
plt.figure(figsize = (10,10))
plt.subplot(2,1,1)
plt.title("Host Listing Distribution")
sns.boxenplot(data = Airbnb_data1 , x = 'calculated_host_listings_count')
plt.subplot(2,1,2)
sns.histplot(data = Airbnb_data1, x = 'calculated_host_listings_count',bins=20,binrange=(0,20))
plt.show()
```

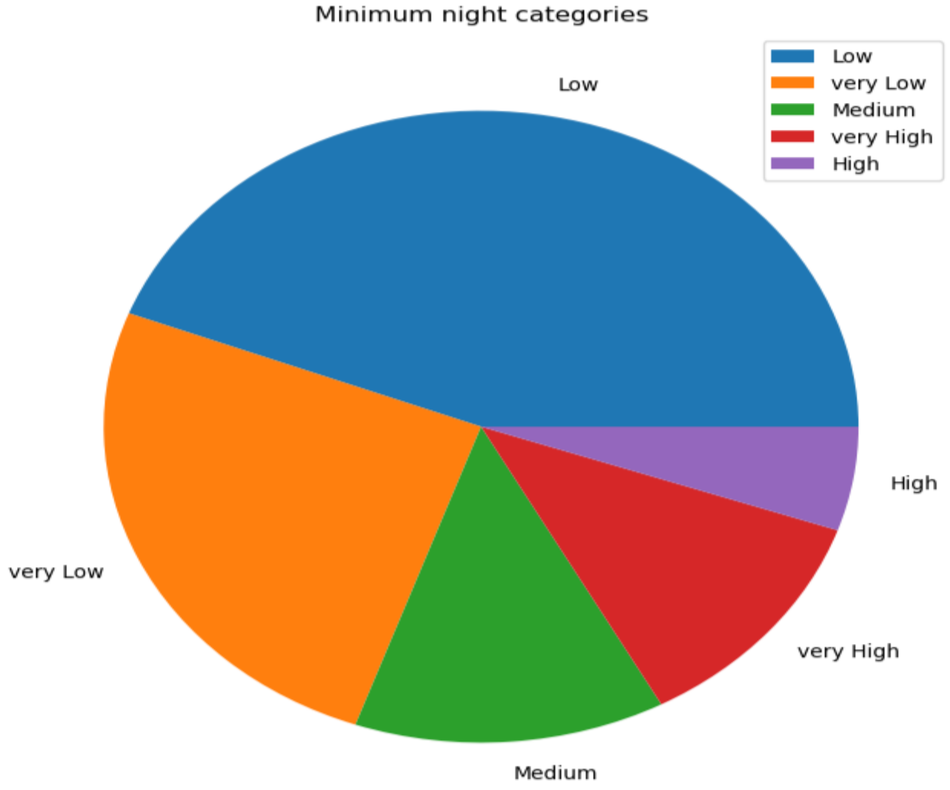

Host Listing Distribution
0
50
100
150
200
250
300
calculated_host_listings_count
Count
25000
20000
15000
10000
5000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
calculated_host_listings_count

5.13 minimum_night_categories

```
Airbnb_data1.minimum_night_categories.value_counts(normalize= True)*100
```

```
Low           44.151361
very Low      25.444991
Medium        13.327838
very High     11.756524
High           5.319286
Name: minimum_night_categories, dtype: float64
```

```
plt.figure(figsize=(8,8))
plt.title('Minimum night categories')
plt.pie(x = Airbnb_data1.minimum_night_categories.value_counts(), labels=Airbnb_data1.minimum_night_categories.value_counts().inc
plt.legend()
plt.show()
```

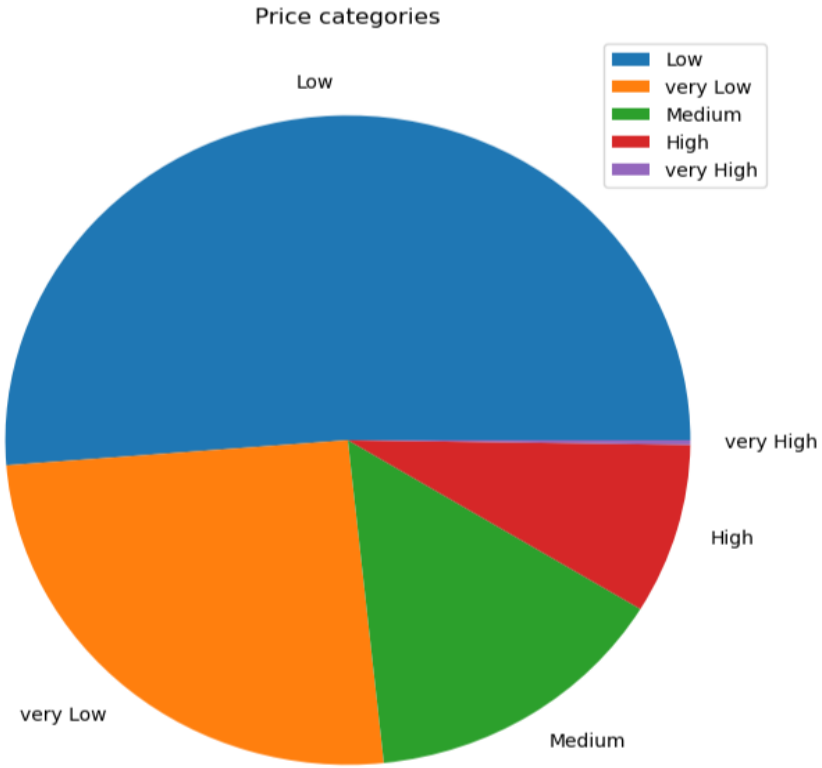


5.14 price_categories

```
Airbnb_data1['price_categories'].value_counts(normalize=True)*100
```

```
Low           51.232580
very Low      25.444991
Medium        14.615801
High           8.477370
very High      0.229257
Name: price_categories, dtype: float64
```

```
plt.figure(figsize=(8,8))
plt.title('Price categories')
plt.pie(x = Airbnb_data1.price_categories.value_counts(),labels=Airbnb_data1.price_categories.value_counts().index)
plt.legend()
plt.show()
```



Insights:

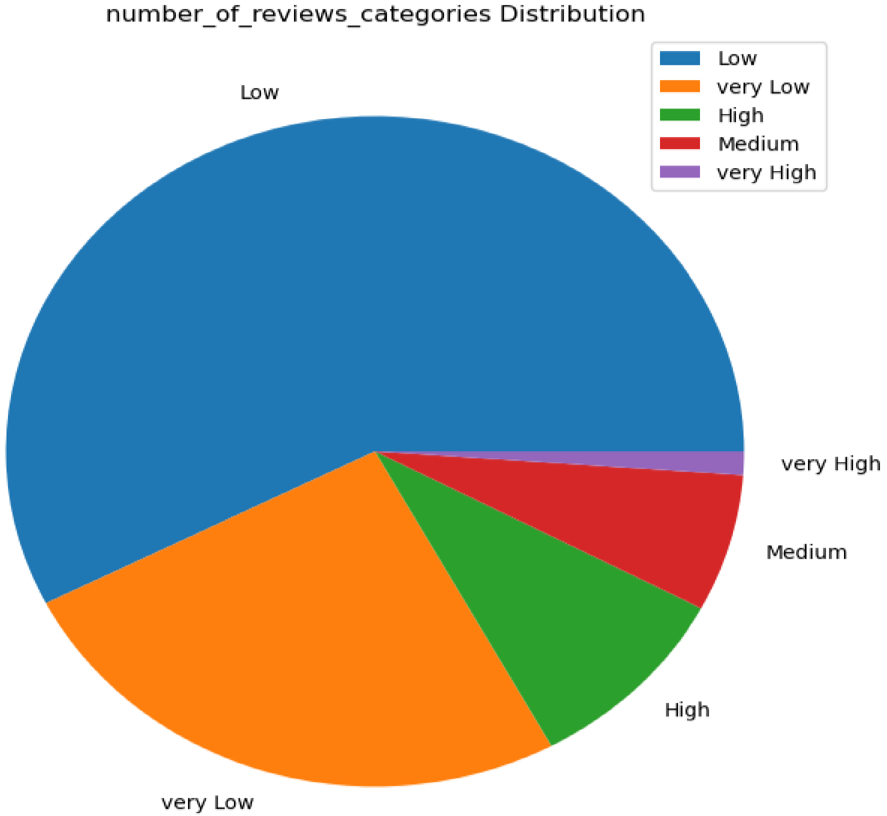
- What is the pricing ranges preferred by customers?
- 'Low' price ranges are preferred by custumers followed by very 'Low' price ranges.

5.15 number_of_reviews_categories

```
Airbnb_data.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=(8,8))
plt.title('number_of_reviews_categories Distribution')
plt.pie(x = Airbnb_data1.number_of_reviews_categories.value_counts(),labels=Airbnb_data1.number_of_reviews_categories.value_count
plt.legend()
plt.show()
```



6.Bivariate and Multivariate Analysis

6.1 Finding the correlations

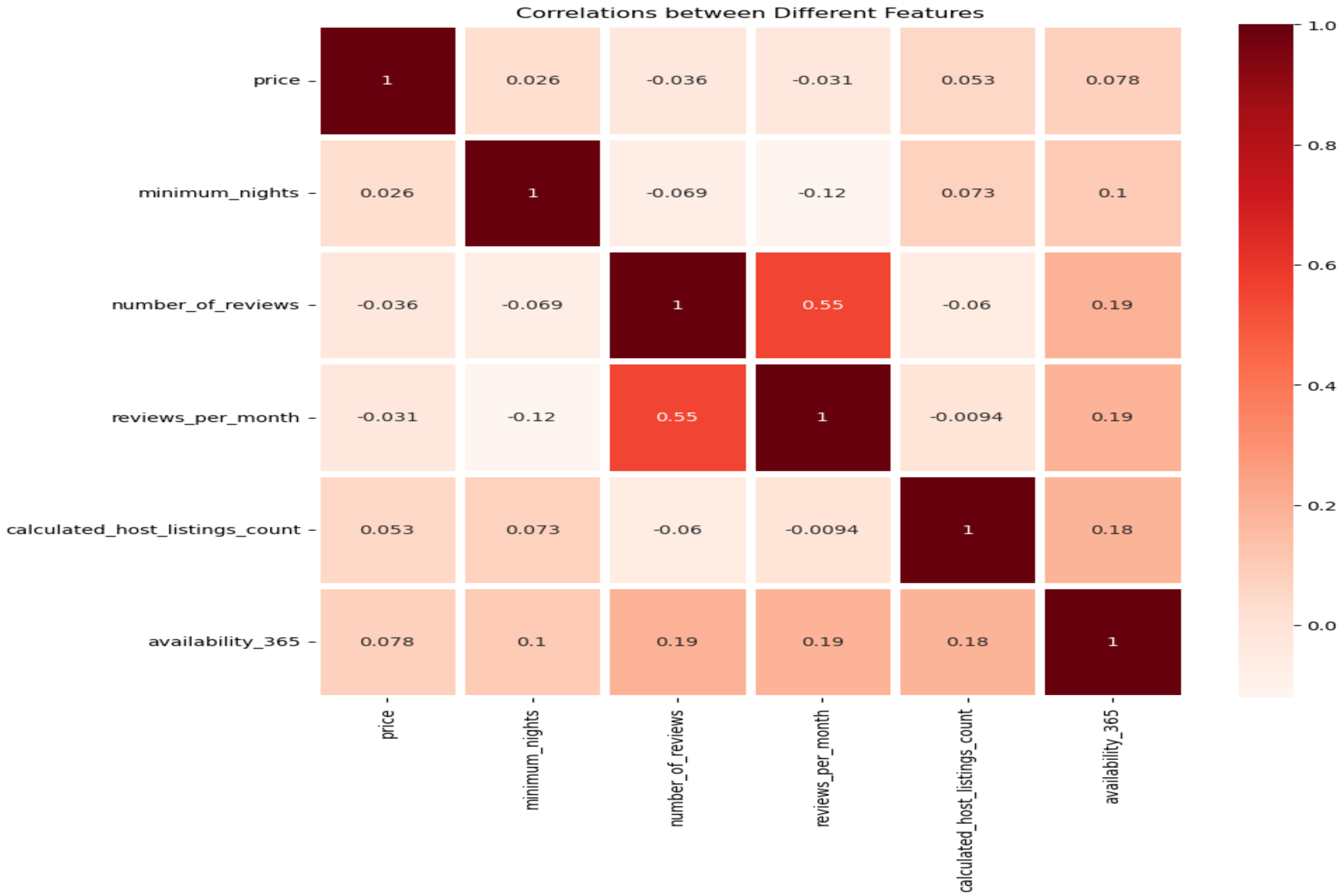
```
numerical_columns = Airbnb_data1.columns[[9,10,11,13,14,15]]
Airbnb_data1[numerical_columns].head()
```

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
0	149	1	9	0.21	6	365
1	225	1	45	0.38	2	355
3	89	1	270	4.64	1	194
4	80	10	9	0.10	1	0
5	200	3	74	0.59	1	129

```
Airbnb_data1[numerical_columns].corr()
```

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
price	1.000000	0.025501	-0.035924	-0.030623	0.052895	0.078276
minimum_nights	0.025501	1.000000	-0.069366	-0.121712	0.073474	0.101658
number_of_reviews	-0.035924	-0.069366	1.000000	0.549699	-0.059796	0.193409
reviews_per_month	-0.030623	-0.121712	0.549699	1.000000	-0.009442	0.185896
calculated_host_listings_count	0.052895	0.073474	-0.059796	-0.009442	1.000000	0.182981
availability_365	0.078276	0.101658	0.193409	0.185896	0.182981	1.000000

```python
plt.figure(figsize=(10,10))
plt.title("Correlations between Different Features")
sns.heatmap(data = Airbnb_data1[numerical_columns].corr(), annot=True, cmap="Reds", linecolor="White", linewidths=5)
plt.show()
```



6.2 Finding Top correlations

```
corr_matrix = Airbnb_data1[numerical_columns].corr().abs()

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

data = (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	0.025501	0.035924	0.030623	0.052895	0.078276
minimum_nights	0.025501	1.000000	0.069366	0.121712	0.073474	0.101658
number_of_reviews	0.035924	0.069366	1.000000	0.549699	0.059796	0.193409
reviews_per_month	0.030623	0.121712	0.549699	1.000000	0.009442	0.185896
calculated_host_listings_count	0.052895	0.073474	0.059796	0.009442	1.000000	0.182981
availability_365	0.078276	0.101658	0.193409	0.185896	0.182981	1.000000

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

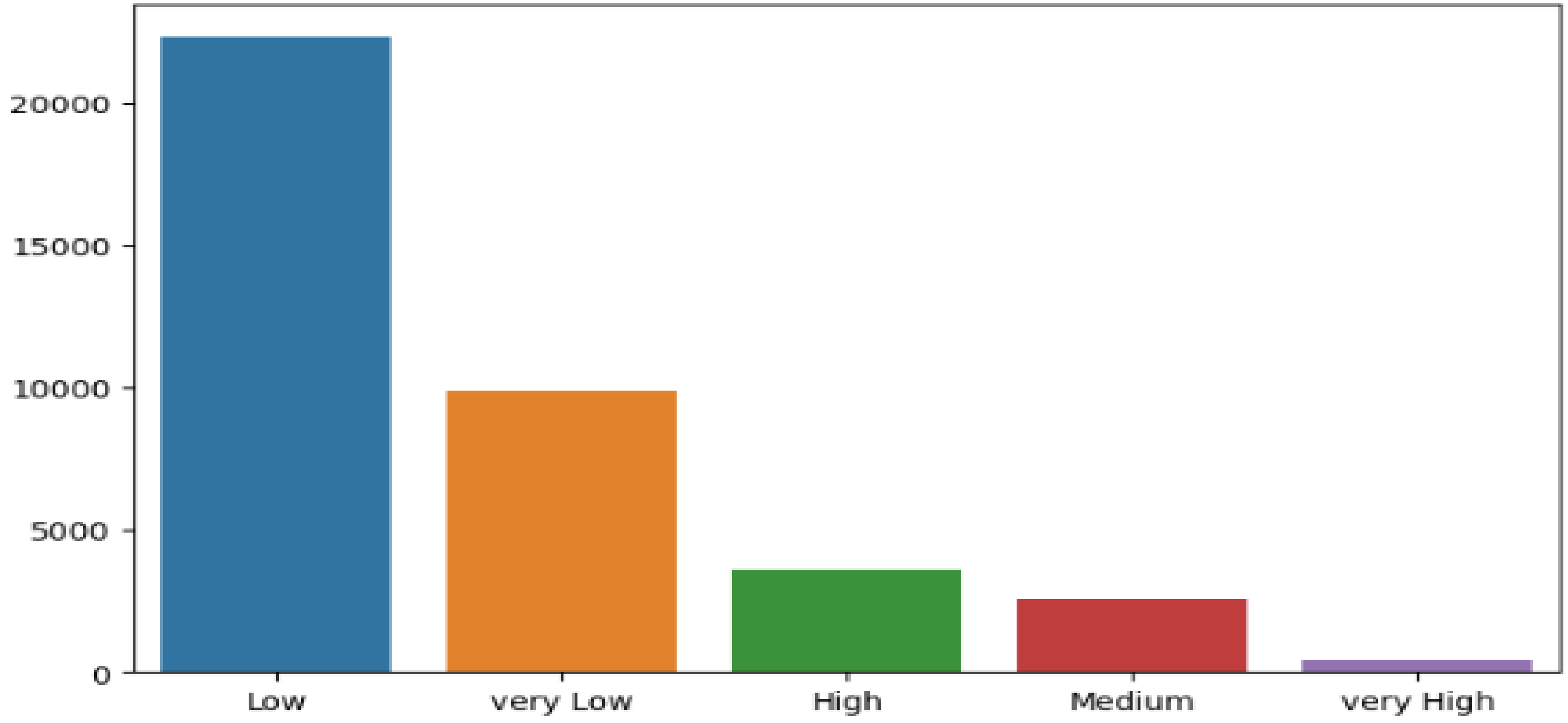
```
number_of_reviews               availability_365                  0.193409
reviews_per_month               availability_365                  0.185896
calculated_host_listings_count  availability_365                  0.182981
minimum_nights                  reviews_per_month                 0.121712
                                availability_365                  0.101658
price                           availability_365                  0.078276
minimum_nights                  calculated_host_listings_count    0.073474
dtype: float64
```

6.3 number_of_reviews_categories and prices

```
# prices for each of reviews_categories
y1 = Airbnb_data1.number_of_reviews_categories.value_counts()
y1
```

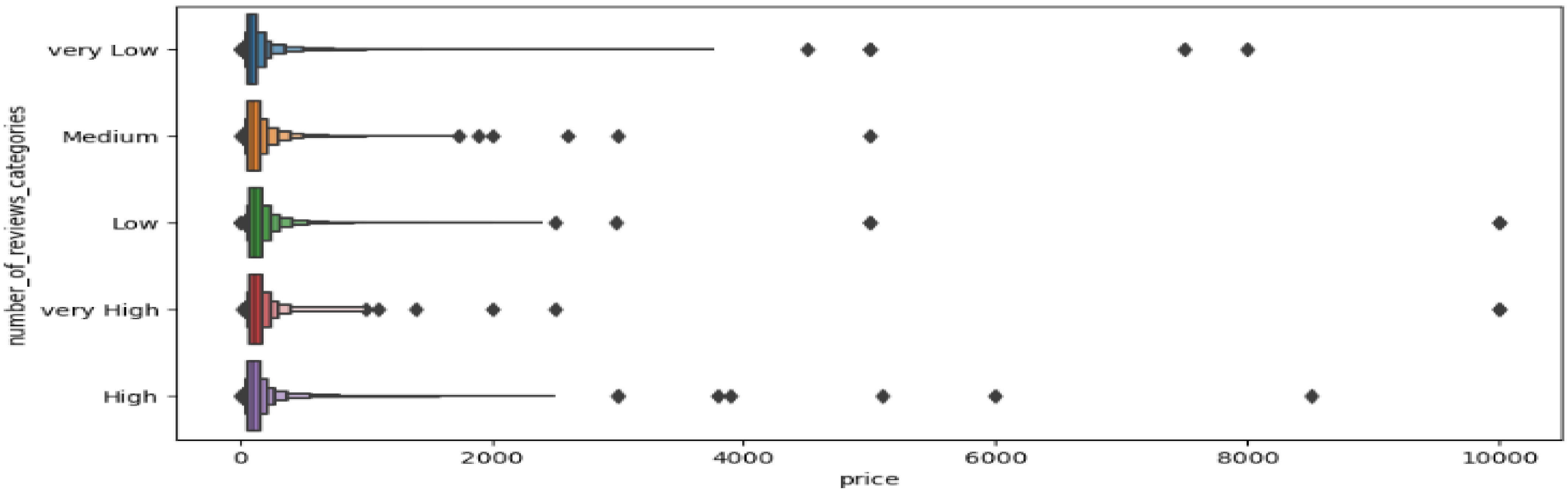
```
Low           22314
very Low       9878
High           3618
Medium         2572
very High       439
Name: number_of_reviews_categories, dtype: int64
```

```
plt.figure(figsize=(8,5))
sns.barplot(x = y1.index,y = y1.values)
plt.show()
```



```python
plt.figure(figsize=(10,5))
sns.boxenplot(y = Airbnb_data1.number_of_reviews_categories , x = Airbnb_data1.price)
```

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




```python
Airbnb_data1.groupby('number_of_reviews_categories').price.mean().sort_values()
```

```
number_of_reviews_categories
very Low      122.647702
Medium        140.857309
High          146.001106
Low           149.489558
very High     199.886105
Name: price, dtype: float64
```

```python
Airbnb_data1.groupby('number_of_reviews_categories').price.median().sort_values()
```

```
number_of_reviews_categories
very Low       85.0
Medium        107.0
High          110.0
Low           119.0
very High     120.0
Name: price, dtype: float64
```

Insights:

- What is the pricing ranges preferred by customers?
- The total price for 'Low' or 'very Low' number_of_reviews_categories are high.

6.4 ('room_type' and 'number_of_reviews_categories')

```
Airbnb_data1.room_type.value_counts()
```

```
Entire home/apt    20321
Private room       17654
Shared room          846
Name: room_type, dtype: int64
```

```
pd.crosstab(Airbnb_data1['room_type'], Airbnb_data1['number_of_reviews_categories'])
```

number_of_reviews_categories	High	Low	Medium	very High	very Low
room_type					
Entire home/apt	2329	12978	1479	322	3213
Private room	1218	9052	1069	109	6206
Shared room	71	284	24	8	459

```
Airbnb_data1.groupby('room_type').number_of_reviews.sum()
```

```
room_type
Entire home/apt    579856
Private room       537965
Shared room         19256
Name: number_of_reviews, dtype: int64
```

```
Airbnb_data1.groupby('room_type').number_of_reviews.sum()/Airbnb_data.room_type.value_counts()
```

```
room_type
Entire home/apt    22.820890
Private room       24.095897
Shared room        16.600000
dtype: float64
```

Insights:

- 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 likey to give reviews. only 16 %

6.5 'room_type' and 'price_categories'

```
pd.crosstab(Airbnb_data1['room_type'], Airbnb_data1['price_categories'])
```

price_categories	High	Low	Medium	very High	very Low
room_type					
Entire home/apt	2245	11495	3301	67	3213
Private room	984	8140	2302	22	6206
Shared room	62	254	71	0	459

6.6 'room_type' and 'reviews_per_month'

```
Airbnb_data1.room_type.value_counts()
```

```
Entire home/apt    20321
Private room       17654
Shared room          846
Name: room_type, dtype: int64
```

```
Airbnb_data1.groupby('room_type').reviews_per_month.mean()
```

```
room_type
Entire home/apt    1.306712
Private room       1.445075
Shared room        1.471726
Name: reviews_per_month, dtype: float64
```

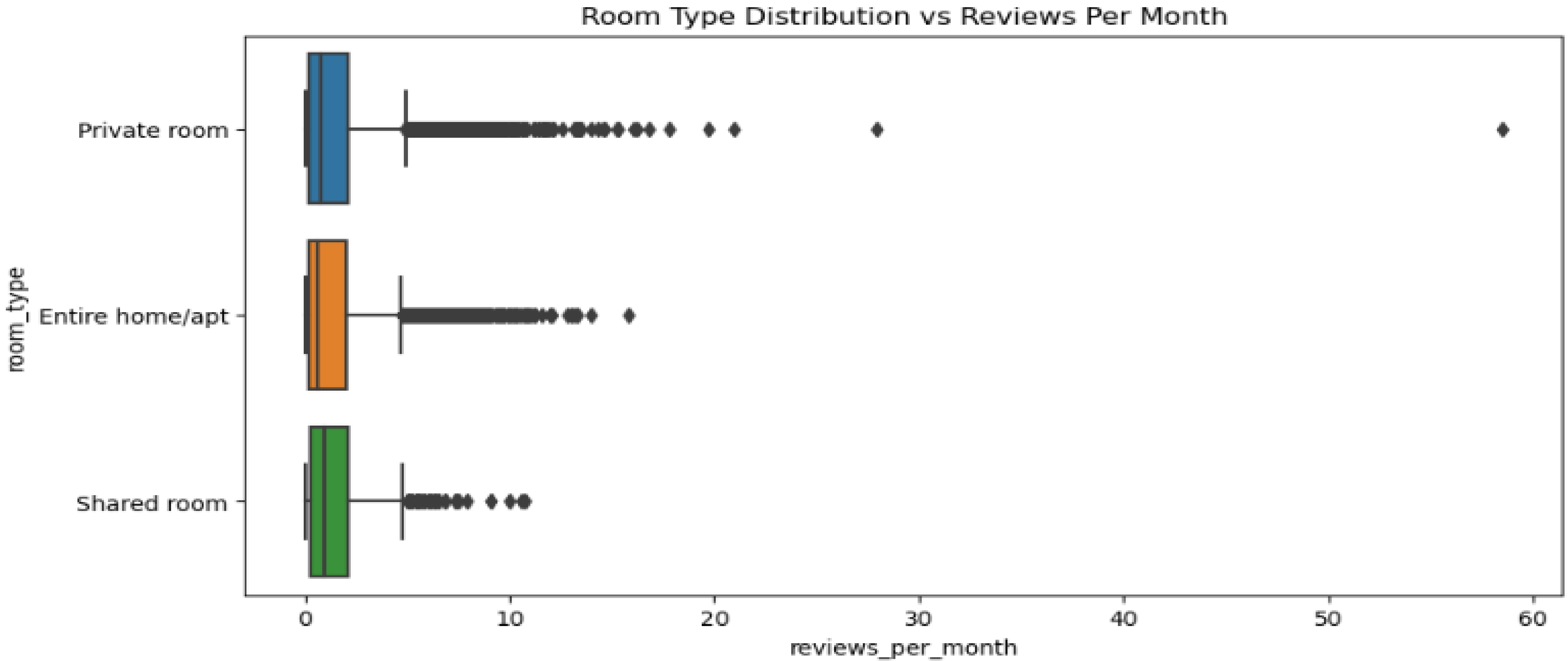
```
Airbnb_data1.groupby('room_type').reviews_per_month.median()
```

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

```
Airbnb_data1.groupby('room_type').reviews_per_month.sum()
```

```
room_type
Entire home/apt    26553.69
Private room       25511.36
Shared room         1245.08
Name: reviews_per_month, dtype: float64
```

```
plt.figure(figsize=(10,5))
plt.title("Room Type Distribution vs Reviews Per Month")
sns.boxplot(data = Airbnb_data1, y = 'room_type' ,x = 'reviews_per_month')
plt.show()
```



Insights:

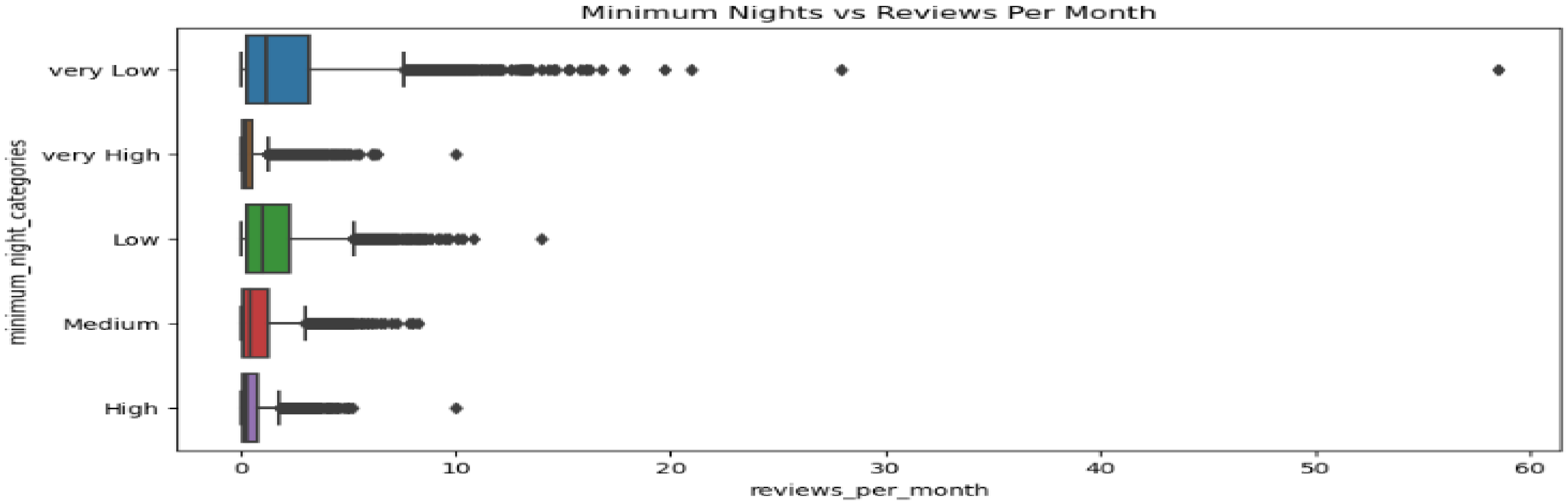
- For each 'room_type' there are ~1.4 reviews per month on average.

6.7 minimum_night_categories and reviews_per_month

```
Airbnb_data1.groupby('minimum_night_categories').reviews_per_month.sum().sort_values()
```

```
minimum_night_categories
High           1227.48
very High      2234.45
Medium         4686.12
very Low      20377.86
Low           24784.22
Name: reviews_per_month, dtype: float64
```

```
plt.figure(figsize=(10,5))
plt.title("Minimum Nights vs Reviews Per Month")
sns.boxplot(data = Airbnb_data1, y = 'minimum_night_categories' ,x = 'reviews_per_month')
plt.show()
```



Insights

- Customer's 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

6.8 'availability_365_categories', 'price_categories' and 'reviews_per_month'

```
Airbnb_data1.availability_365_categories.value_counts()
```

```
very Low     13042
Low          10062
very High     6124
Medium        4963
High          4630
Name: availability_365_categories, dtype: int64
```

```python
pd.DataFrame(Airbnb_data1.groupby(['availability_365_categories','price_categories']).reviews_per_month.mean())
```

availability_365_categories	price_categories	reviews_per_month
High	High	0.598431
	Low	2.200411
	Medium	1.056111
	very High	0.342308
	very Low	3.289381
Low	High	0.638307
	Low	1.784057
	Medium	0.883844
	very High	0.803750
	very Low	2.897139
Medium	High	0.591070
	Low	1.993565
	Medium	1.157492
	very High	0.517500
	very Low	2.893918
very High	High	0.428236
	Low	1.489812
	Medium	0.694283
	very High	0.276571
	very Low	2.207028
very Low	High	0.337780
	Low	0.506156
	Medium	0.277142
	very High	0.480588
	very Low	0.670527

Insights

- If availability and price both is very high, reviews_per_month is low on average.
- Very high availability and very low price are likely to getting more reviews.