upGrad

airbnb

NYC

Story Telling Case Study

PPT - I

Agenda

1.	Objective
2.	Data Life Cycle
3.	Analysis Method
4.	Recommendations
5.	Data Methodology

Objective

1. **Data Analysis Objective:** Conduct a comprehensive analysis of the New York Airbnb dataset through effective questioning to extract actionable insights.

2. **Question Framing:** Develop precise, effective questions aimed at revealing vital information within the dataset.

3. **Data Exploration:** Utilize statistical and analytical techniques to delve deeply into the dataset, ensuring a complete understanding of its contents.

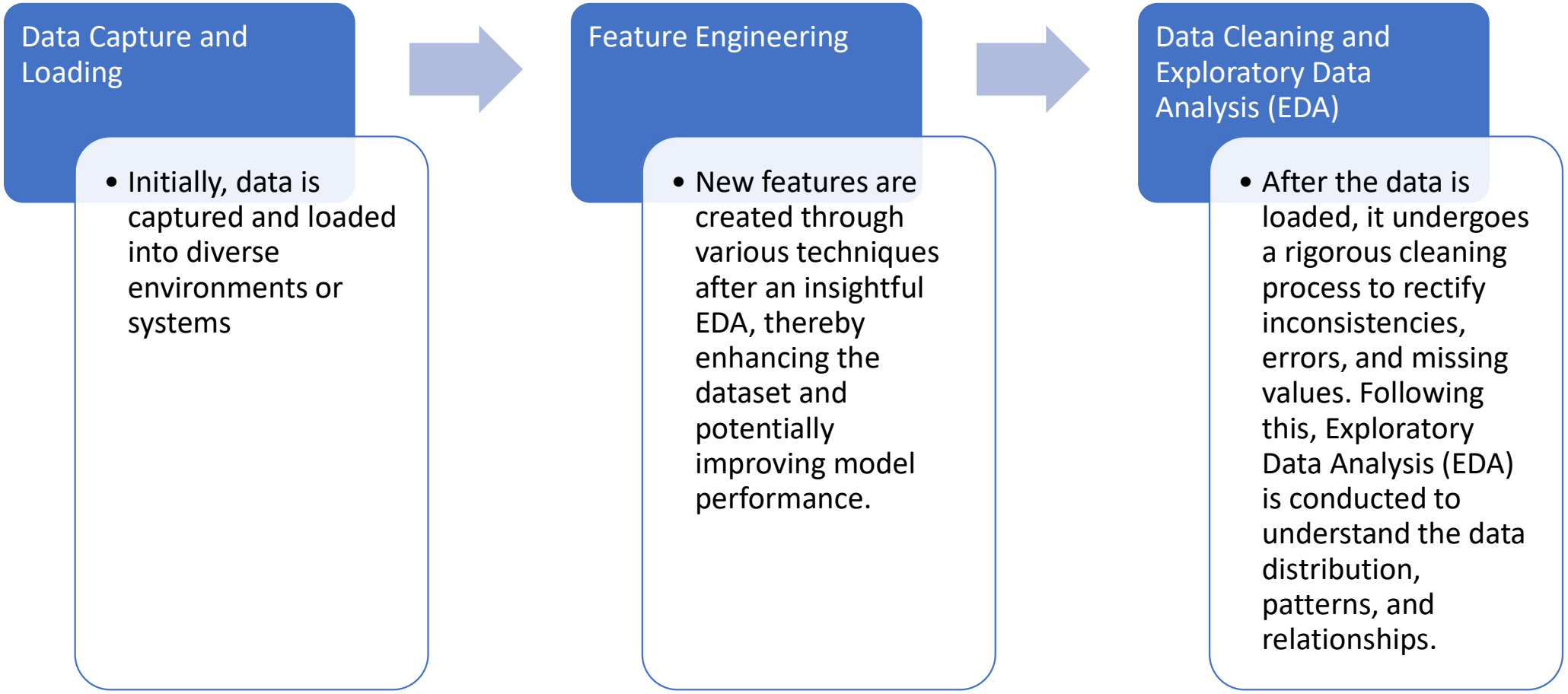
4. **Visualization Techniques:** Employ diverse graphical representations to visually showcase the insights extracted from the data.

5. **Effective Communication**: Utilize data visualization and analytical methods to effectively communicate the findings, ensuring clarity and actionable understanding for the audience.

6. **Insightful Interpretation**: Present the results in an actionable manner, enabling the audience to comprehend and act upon the insights gained from the New York Airbnb dataset.

7. **Comprehensive Understanding**: The goal is to provide a complete understanding of the dataset, derived insights, and their relevance for decision-making and strategic planning.

Data Life Cycle

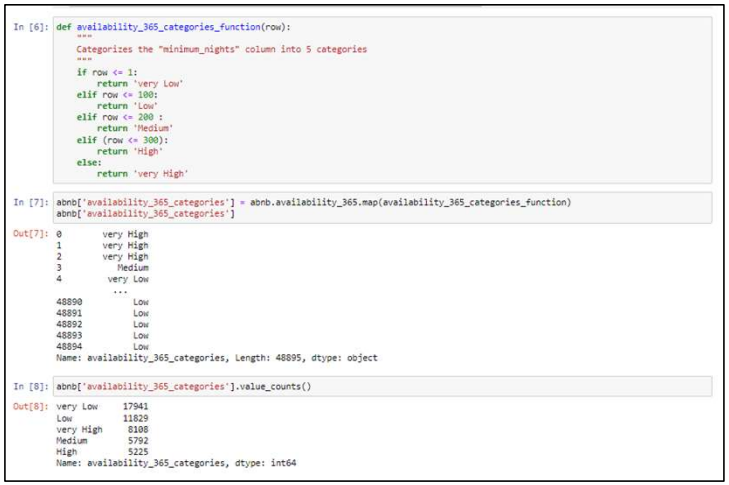


1. Data Capture and Loading

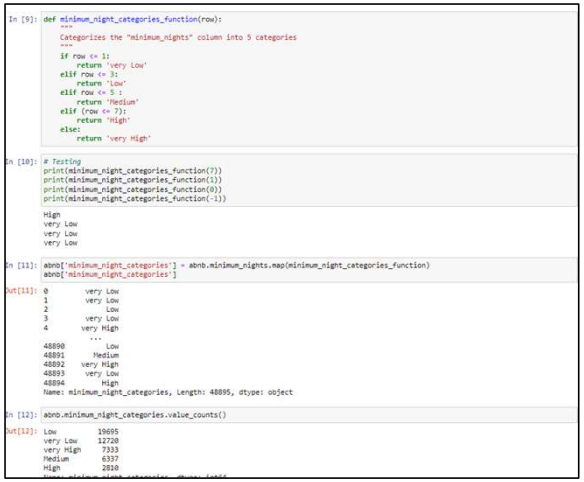
1. Importing libraries and reading the data

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

In [3]: abnb = pd.read_csv('AB_NYC_2019.csv')

In [4]: abnb.head(10)
```

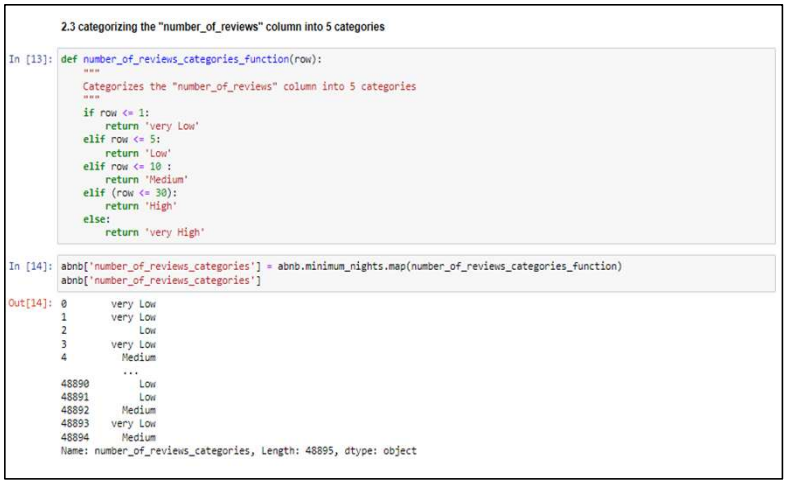
2. Feature Engineering



```
In [6]: 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'

In [7]: abnb['availability_365_categories'] = abnb.availability_365.map(availability_365_categories_function)
        abnb['availability_365_categories']

Out[7]: 0        very High
        1        very High
        2        very High
        3           Medium
        4         very Low
                   ...
        48890          Low
        48891          Low
        48892          Low
        48893          Low
        48894          Low
        Name: availability_365_categories, Length: 48895, dtype: object

In [8]: abnb['availability_365_categories'].value_counts()

Out[8]: very Low     17941
        Low          11829
        very High     8108
        Medium        5792
        High          5225
        Name: availability_365_categories, dtype: int64
```



```
In [9]: 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'

In [10]: # Testing
         print(minimum_night_categories_function(7))
         print(minimum_night_categories_function(1))
         print(minimum_night_categories_function(0))
         print(minimum_night_categories_function(-1))

         High
         very Low
         very Low
         very Low

In [11]: abnb['minimum_night_categories'] = abnb.minimum_nights.map(minimum_night_categories_function)
         abnb['minimum_night_categories']

Out[11]: 0         very Low
         1         very Low
         2              Low
         3         very Low
         4        very High
                    ...
         48890          Low
         48891       Medium
         48892    very High
         48893    very Low
         48894         High
         Name: minimum_night_categories, Length: 48895, dtype: object

In [12]: abnb.minimum_night_categories.value_counts()

Out[12]: Low          19695
         very Low     12720
         very High     7333
         Medium        6337
         High          2810
```



2.3 categorizing the "number_of_reviews" column into 5 categories

```
In [13]: 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'

In [14]: abnb['number_of_reviews_categories'] = abnb.minimum_nights.map(number_of_reviews_categories_function)
         abnb['number_of_reviews_categories']

Out[14]: 0         very Low
         1         very Low
         2              Low
         3         very Low
         4           Medium
                     ...
         48890          Low
         48891          Low
         48892       Medium
         48893    very Low
         48894       Medium
         Name: number_of_reviews_categories, Length: 48895, dtype: object
```

3. Data Cleaning and EDA

3. Fixing columns

```
In [20]: # Check Non-Null counts and data types
         abnb.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
```

Observation: reviews_per_month is of float data type. Correct data type would be datetime64.

4. Data types

4.1 Categorical

```
In [23]: # Categorical
         categorical_columns = abnb.columns[[0,1,3,4,5,8,16,17,18,19]]
         categorical_columns
```

```
Out[23]: Index(['id', 'name', 'host_name', 'neighbourhood_group', 'neighbourhood',
                'room_type', 'availability_365_categories', 'minimum_night_categories',
                'number_of_reviews_categories', 'price_categories'],
               dtype='object')
```

```
In [24]: # To see the first few rows of categorical columns
         abnb[categorical_columns].head()
```

Out[24]:

	id	name	host_name	neighbourhood_group	neighbourhood	room_type	availability_365_categories	minimum_night_categories	number_of_revi
0	2539	Clean & quiet apt home by the park	John	Brooklyn	Kensington	Private room	very High	very Low	
1	2595	Skylit Midtown Castle	Jennifer	Manhattan	Midtown	Entire home/apt	very High	very Low	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	Elisabeth	Manhattan	Harlem	Private room	very High	Low	
3	3831	Cozy Entire Floor of Brownstone	LisaRoxanne	Brooklyn	Clinton Hill	Entire home/apt	Medium	very Low	
4	5022	Entire Apt: Spacious Studio/Loft by central park	Laura	Manhattan	East Harlem	Entire home/apt	very Low	very High	

4.2 Numerical

```
In [25]: numerical_columns = abnb.columns[[9,10,11,13,14,15]]
         numerical_columns
```

```
Out[25]: Index(['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month',
                'calculated_host_listings_count', 'availability_365'],
               dtype='object')
```

5. Missing values

```
In [29]: # Check if the data has any missing value (% of null values)
         round((abnb.isnull().sum()/len(abnb))*100,2)
```

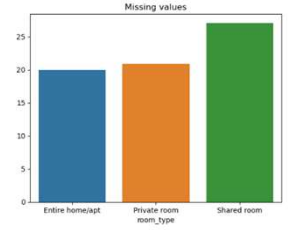
```
Out[29]: id                                0.00
         name                              0.03
         host_id                           0.00
         host_name                         0.04
         neighbourhood_group               0.00
         neighbourhood                     0.00
         latitude                          0.00
         longitude                         0.00
         room_type                         0.00
         price                             0.00
         minimum_nights                    0.00
         number_of_reviews                 0.00
         last_review                      20.56
         reviews_per_month                20.56
         calculated_host_listings_count    0.00
         availability_365                  0.00
         availability_365_categories       0.00
         minimum_night_categories          0.00
         number_of_reviews_categories      0.00
         price_categories                  0.00
         dtype: float64
```

5.3 Missing values Analysis ('room_type' feature)

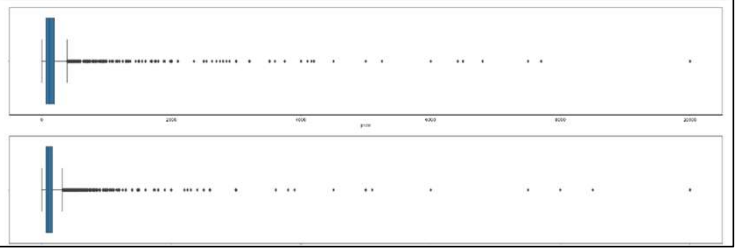
```
In [36]: # Count of 'room_type' with missing values
         mval = (mvk.groupby('room_type').room_type.count()/abnb.groupby('room_type').room_type.count())*100
         mval
```

```
Out[36]: room_type
         Entire home/apt    19.981109
         Private room       20.877004
         Shared room        27.068966
         Name: room_type, dtype: float64
```

```
In [37]: plt.title('Missing values')
         sns.barplot(x = mval.index, y = mval.values)
         plt.show()
```




```
In [39]: plt.figure(figsize=(30,10))

         plt.subplot(2,1,1)
         sns.boxplot(data = abnb[abnb['last_review'].isnull()] , x = 'price',  width= .8)

         plt.subplot(2,1,2)
         sns.boxplot(data = abnb[abnb['last_review'].notnull()] , x = 'price',  width= .8)

         plt.show()
```



3. Data Cleaning and EDA (continued…)

6. Univariate Analysis

6.1 Host Name and Value

```
In [40]: abnb.name.value_counts()
```

```
Out[40]: 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
```

```
In [41]: abnb.host_name.value_counts()
```

```
Out[41]: 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
```

```
In [42]: abnb.host_name.value_counts().index[:10]
```

```
Out[42]: Index(['Michael', 'David', 'Sonder (NYC)', 'John', 'Alex', 'Blueground',
                'Sarah', 'Daniel', 'Jessica', 'Maria'],
               dtype='object')
```

```
In [43]: # Top 10 hosts
         plt.figure(figsize=(15,5))
         sns.barplot(x = abnb.host_name.value_counts().index[:10] , y = abnb.host_name.value_counts().values[:10])
         plt.show()
```

7. Bivariate and Multivariate Analysis

7.1 Finding the correalations

```
In [98]: abnb[numerical_columns].head()
```

Out[98]:

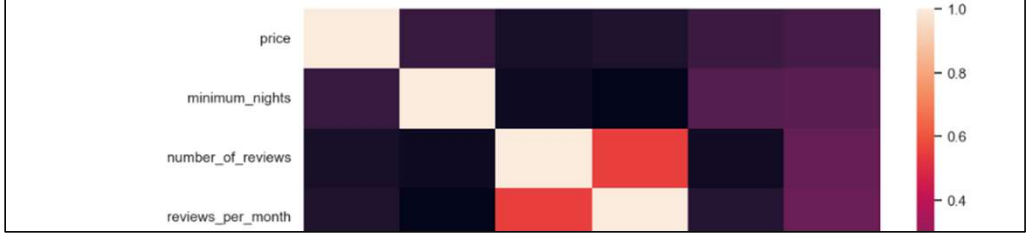
	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
0	149	1	9	0.21	6	365
1	225	1	45	0.38	2	355
2	150	3	0	NaN	1	365
3	89	1	270	4.64	1	194
4	80	10	9	0.10	1	0

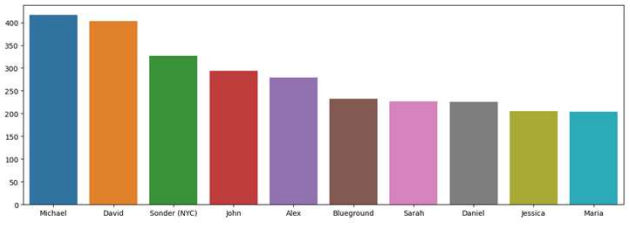
```
In [99]: abnb[numerical_columns].corr()
```

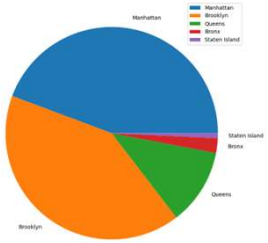
Out[99]:

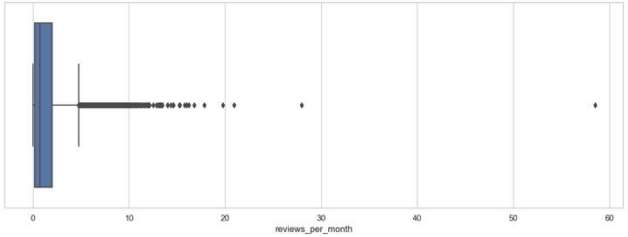
	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
price	1.000000	0.042799	-0.047954	-0.030608	0.057472	0.081829
minimum_nights	0.042799	1.000000	-0.080116	-0.121702	0.127960	0.144303
number_of_reviews	-0.047954	-0.080116	1.000000	0.549868	-0.072376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

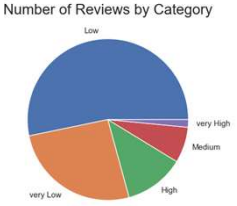
```
In [100]: plt.figure(figsize=(10,5))
          sns.heatmap(data = abnb[numerical_columns].corr())
          plt.show()
```

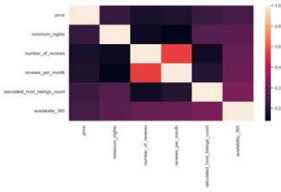












Analysis Method

Data Collection and Loading:

- Gathered and imported the Airbnb dataset, consisting of New York listings, into a working environment like Python using Pandas.
- Ensured data readiness for analysis by loading it into a DataFrame for further processing.

Data Cleaning and Exploratory Data Analysis (EDA):

- Performed data cleaning procedures to handle missing values and ensure data consistency.
- Conducted EDA using statistical measures and visualization tools to comprehend distributions, relationships, and patterns within the dataset.

Feature Engineering:

- Created new features from the available dataset, such as categorizing variables (e.g., 'availability_365' and 'minimum_nights') into distinct categories to gain more insights.

Statistical Techniques and Correlation Analysis:

- Utilized statistical methods to establish correlations between variables within the dataset.
- Examined correlations using tools like heatmap visualizations to identify key relationships and meaningful associations.

Categorical Analysis:

- Analyzed categorical features to identify patterns and preferences, such as the 'room_type' distribution and preferences among users.

Visualization and Reporting:

- Utilized various visualizations including histograms, box plots, pie charts, and bar plots to summarize and illustrate key findings for better comprehension.

Conclusion and Recommendation:

- Summarized insights, conclusions, and any recommended actions based on the analysis conducted on the dataset

Recommendations

1. **Price Adjustment:** Adjust pricing strategies to align with the 'very low' or 'low' categories, which are more likely to attract reviews and potentially increase demand.

2. **Property Diversification:** Encourage more listings under 'Entire home/apt' categories as they are in high demand. Adjust existing listings or focus on acquiring similar properties.

3. **Optimize Shared Room Listings:** Enhance the appeal of shared rooms by either adjusting pricing strategies or introducing promotional activities to increase their traction.

4. **Improve User Engagement:** Optimize the property listings for better user experience to attract more customers. Highlight properties with high availability and competitive pricing to improve customer traction.

Data Methodology

Introduction

The methodology aims to conduct a comprehensive Exploratory Data Analysis (EDA) on the Airbnb NYC dataset. The dataset includes various features related to properties listed on Airbnb in New York City.

Data Collection and Preparation

- Data Importing: Utilized Pandas library to read the dataset ('AB_NYC_2019.csv').
- Data Understanding: Displayed the first few rows to gain an initial understanding of the data structure.

Feature Engineering and Categorization

- Availability, Nights, Reviews, and Pricing Categories: Categorized columns including 'availability_365', 'minimum_nights', 'number_of_reviews', and 'price' into five distinct categories using defined conditional functions.

Data Cleaning and Column Adjustments

- Data Type Corrections: Transformed the 'last_review' column to 'datetime64' data type.
- Column Adjustments: Identified categorical, numerical, coordinates, and date columns and displayed their content for a preliminary understanding.

Missing Values Analysis

- Evaluation of Missing Values: Investigated the presence of missing values in columns, specifically focusing on 'last_review' and 'reviews_per_month' columns.

Data Methodology

Univariate Analysis

- Exploration of Features: Analyzed individual features including 'host_name', 'neighbourhood_group', 'price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', and 'availability_365'.
- Visualization: Utilized various visualizations such as bar plots, box plots, histograms, and pie charts to understand the distributions and characteristics of different features.

Bivariate and Multivariate Analysis

- Correlation Analysis: Explored correlations among numerical columns using correlation matrices and visualizations.
- Top Correlations: Identified and examined top meaningful correlations within the dataset.
- Relationship Analysis: Studied the relationships between room types, number of reviews, prices, and availability for further insights.

Data Saving

- Export of Updated Data: Saved the manipulated dataset to 'AB_NYC_2019_updated.csv' after categorization and adjustments.

Conclusion and Recommendations

- Insights: Derived insights into the impact of various categories on prices, reviews, and customer preferences.
- Recommendations: Suggested adjustments such as keeping minimum nights lower, modifying prices for higher availability, and understanding property features more customer-oriented.

Data Methodology

Implications

- This methodology provides a foundation for making data-driven decisions and more in-depth analyses related to Airbnb's business operations.

Further Steps

The methodology serves as a basis for future advanced analyses, predictive modelling, and strategic decision-making in the context of Airbnb's property listings.

1. Importing libraries and reading the data

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
abnb = pd.read_csv('AB_NYC_2019.csv')
```

In [5]:
```
abnb.head(10)
```

Out[5]:

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

2. Creating features

Note: Through categorization, we enhance our comprehension of the interconnections between elements and improve our ability to articulate our discoveries

2.1 categorizing the "availability_365" column into 5 categories

In [6]:
```
abnb.head()
```

Out[6]:

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

In [7]:
```
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'
```

In [8]:
```
abnb['availability_365_categories'] = abnb.availability_365.map(availability_365_categor
abnb['availability_365_categories']
```

Out[8]:
```
0         very High
1         very High
2         very High
3            Medium
4          very Low
            ...
48890           Low
48891           Low
48892           Low
48893           Low
48894           Low
Name: availability_365_categories, Length: 48895, dtype: object
```

In [9]:
```
abnb['availability_365_categories'].value_counts()
```

Out[9]:
```
very Low     17941
Low          11829
very High     8108
Medium        5792
```

```
High          5225
Name: availability_365_categories, dtype: int64
```

2.2 categorizing the "minimum_nights" column into 5 categories

In [10]:
```
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'
```

In [11]:
```
# Testing
print(minimum_night_categories_function(7))
print(minimum_night_categories_function(1))
print(minimum_night_categories_function(0))
print(minimum_night_categories_function(-1))
```

```
High
very Low
very Low
very Low
```

In [12]:
```
abnb['minimum_night_categories'] = abnb.minimum_nights.map(minimum_night_categories_func
abnb['minimum_night_categories']
```

Out[12]:
```
0         very Low
1         very Low
2              Low
3         very Low
4        very High
           ...
48890          Low
48891       Medium
48892    very High
48893     very Low
48894         High
Name: minimum_night_categories, Length: 48895, dtype: object
```

In [13]:
```
abnb.minimum_night_categories.value_counts()
```

Out[13]:
```
Low          19695
very Low     12720
very High     7333
Medium        6337
High          2810
Name: minimum_night_categories, dtype: int64
```

2.3 categorizing the "number_of_reviews" column into 5 categories

In [14]:
```
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'
```

In [15]:
```
abnb['number_of_reviews_categories'] = abnb.minimum_nights.map(number_of_reviews_categor
abnb['number_of_reviews_categories']
```

Out[15]:
```
0         very Low
1         very Low
2              Low
3         very Low
4           Medium
            ...
48890          Low
48891          Low
48892       Medium
48893     very Low
48894       Medium
Name: number_of_reviews_categories, Length: 48895, dtype: object
```

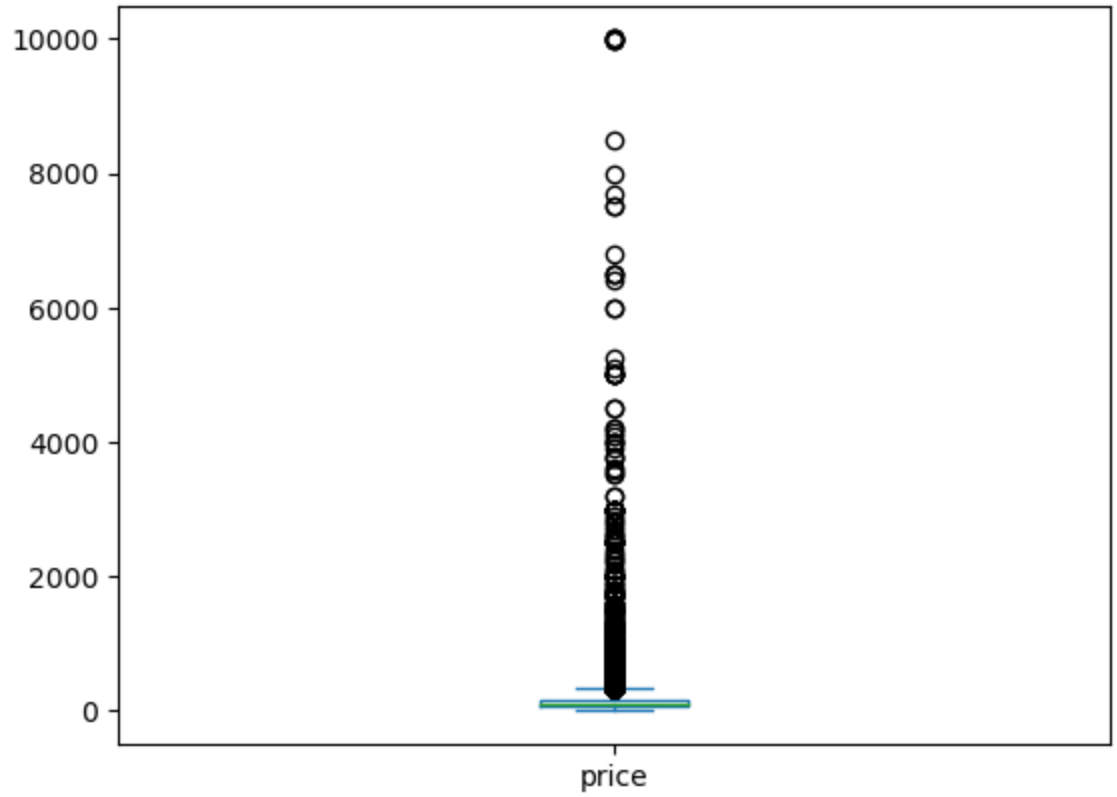
2.4 categorizing the "price" column into 5 categories

In [16]:
```
abnb.price.describe()
```

Out[16]:
```
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
```

In [17]:
```
abnb.price.plot.box()
```

Out[17]:
```
<Axes: >
```



In [18]:
```
abnb[abnb.price == 0].shape
```

Out[18]:
```
(11, 19)
```

In [19]:
```
def price_categories_function(row):
    """
    Categorizes the "number_of_reviews" column into 5 categories
    """
    if row <= 1:
        return 'very Low'
    elif row <= 4:
        return 'Low'
    elif row <= 15 :
        return 'Medium'
    elif (row <= 100):
        return 'High'
    else:
        return 'very High'
```

In [20]:
```
abnb['price_categories'] = abnb.minimum_nights.map(price_categories_function)
abnb['price_categories']
```

Out[20]:
```
0         very Low
1         very Low
2              Low
3         very Low
4           Medium
            ...
48890          Low
48891          Low
48892       Medium
48893     very Low
48894       Medium
Name: price_categories, Length: 48895, dtype: object
```

3. Fixing columns

In [21]:
```
# Check Non-Null counts and data types
abnb.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
```

Observation: reviews_per_month is of float data type. Correct data type would be datetime64.

In [22]:
```
abnb.last_review = pd.to_datetime(abnb.last_review)
abnb.last_review
```

```
C:\Users\sggao\AppData\Local\Temp\ipykernel_13760\1107023067.py:1: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to ensure consistent parsing.
  abnb.last_review = pd.to_datetime(abnb.last_review)
```

Out[22]:
```
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]
```

In [23]:
```
abnb.columns
```

Out[23]:
```
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')
```

4. Data types

4.1 Categorical

In [24]:
```
# Categorical
categorical_columns = abnb.columns[[0,1,3,4,5,8,16,17,18,19]]
categorical_columns
```

Out[24]:
```
Index(['id', 'name', 'host_name', 'neighbourhood_group', 'neighbourhood',
       'room_type', 'availability_365_categories', 'minimum_night_categories',
       'number_of_reviews_categories', 'price_categories'],
      dtype='object')
```

In [25]:
```
# To see the first few rows of categorical columns
abnb[categorical_columns].head()
```

Out[25]:

	id	name	host_name	neighbourhood_group	neighbourhood	room_type	availability_365_categorie
0	2539	Clean & quiet apt home by the park	John	Brooklyn	Kensington	Private room	very Hig
1	2595	Skylit Midtown Castle	Jennifer	Manhattan	Midtown	Entire home/apt	very Hig
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	Elisabeth	Manhattan	Harlem	Private room	very Hig
3	3831	Cozy Entire Floor of Brownstone	LisaRoxanne	Brooklyn	Clinton Hill	Entire home/apt	Mediur
4	5022	Entire Apt: Spacious Studio/Loft by central park	Laura	Manhattan	East Harlem	Entire home/apt	very Lo

4.2 Numerical

In [26]:
```
numerical_columns = abnb.columns[[9,10,11,13,14,15]]
numerical_columns
```

Out[26]:
```
Index(['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month',
       'calculated_host_listings_count', 'availability_365'],
      dtype='object')
```

In [27]:
```
abnb[numerical_columns].head()
```

Out[27]:

	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
2	150	3	0	NaN	1	365
3	89	1	270	4.64	1	194
4	80	10	9	0.10	1	0

In [28]:
```
abnb[numerical_columns].describe()
```

Out[28]:

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	avail
count	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	48
mean	152.720687	7.029962	23.274466	1.373221	7.143982	
std	240.154170	20.510550	44.550582	1.680442	32.952519	
min	0.000000	1.000000	0.000000	0.010000	1.000000	
25%	69.000000	1.000000	1.000000	0.190000	1.000000	
50%	106.000000	3.000000	5.000000	0.720000	1.000000	
75%	175.000000	5.000000	24.000000	2.020000	2.000000	
max	10000.000000	1250.000000	629.000000	58.500000	327.000000	

4.3 Coordinates and date

In [29]:
```
coordinates = abnb.columns[[5,6,12]]
abnb[coordinates]
```

Out[29]:

	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

In [30]:
```
# Check if the data has any missing value (% of null values)
round((abnb.isnull().sum()/len(abnb))*100,2)
```

Out[30]:
```
id                                0.00
name                              0.03
host_id                           0.00
host_name                         0.04
neighbourhood_group               0.00
neighbourhood                     0.00
latitude                          0.00
longitude                         0.00
room_type                         0.00
price                             0.00
minimum_nights                    0.00
```

```
number_of_reviews                  0.00
last_review                       20.56
reviews_per_month                 20.56
calculated_host_listings_count     0.00
availability_365                   0.00
availability_365_categories        0.00
minimum_night_categories           0.00
number_of_reviews_categories       0.00
price_categories                   0.00
dtype: float64
```

Observations:

- 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

In [31]:
```
mva = abnb.loc[abnb.last_review.isnull(),:]
mva
```

Out[31]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitud
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.9419
19	7750	Huge 2 BR Upper East Cental Park	17985	Sing	Manhattan	East Harlem	40.79685	-73.9487
26	8700	Magnifique Suite au N de Manhattan - vue Cloitres	26394	Claude & Sophie	Manhattan	Inwood	40.86754	-73.9263
36	11452	Clean and Quiet in Brooklyn	7355	Vt	Brooklyn	Bedford-Stuyvesant	40.68876	-73.9431
38	11943	Country space in the city	45445	Harriet	Brooklyn	Flatbush	40.63702	-73.9632
...	...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.9499
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.9331
48892	36485431	Sunny Studio	23492952	Ilgar &	Manhattan	Harlem	40.81475	-73.9486

		at Historical Neighborhood		Aysel				
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.991
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.989

10052 rows × 20 columns

5.2 Missing values Analysis ('neighbourhood_group' feature)

In [32]:
```
# Count of 'neighbourhood_group' with missing values
mva.groupby('neighbourhood_group').neighbourhood_group.count()
```

Out[32]:
```
neighbourhood_group
Bronx             215
Brooklyn         3657
Manhattan        5029
Queens           1092
Staten Island      59
Name: neighbourhood_group, dtype: int64
```

In [33]:
```
# Count of 'neighbourhood_group'
abnb.groupby('neighbourhood_group').neighbourhood_group.count()
```

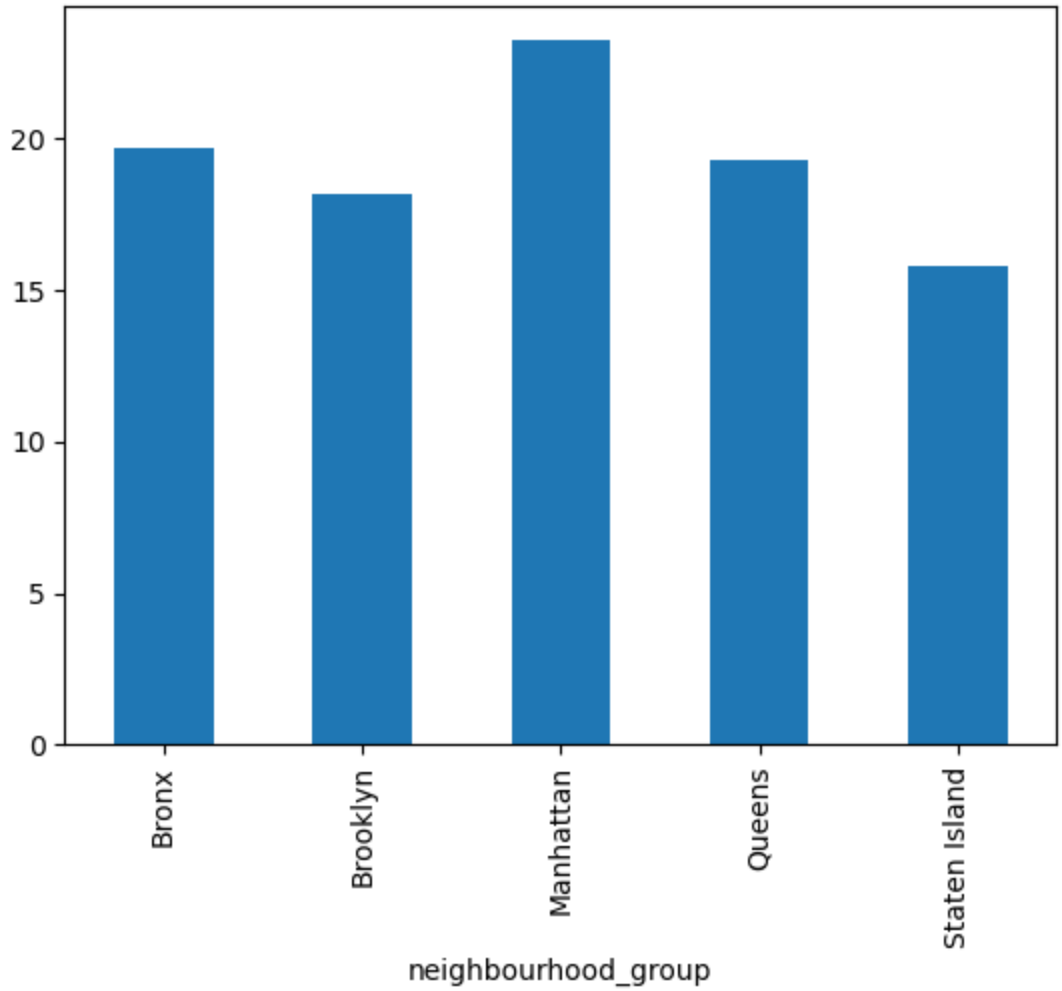
Out[33]:
```
neighbourhood_group
Bronx             1091
Brooklyn         20104
Manhattan        21661
Queens            5666
Staten Island      373
Name: neighbourhood_group, dtype: int64
```

In [34]:
```
(mva.groupby('neighbourhood_group').neighbourhood_group.count()/abnb.groupby('neighbourh
```

Out[34]:
```
neighbourhood_group
Bronx            19.706691
Brooklyn         18.190410
Manhattan        23.216841
Queens           19.272856
Staten Island    15.817694
Name: neighbourhood_group, dtype: float64
```

In [35]:
```
((mva.groupby('neighbourhood_group').neighbourhood_group.count()/abnb.groupby('neighbour
```

Out[35]:
```
<Axes: xlabel='neighbourhood_group'>
```



In [36]:
```
((mva.groupby('neighbourhood_group').neighbourhood_group.count()/abnb.groupby('neighbour
```

Out[36]:
```
19.240898461107257
```

Observation:

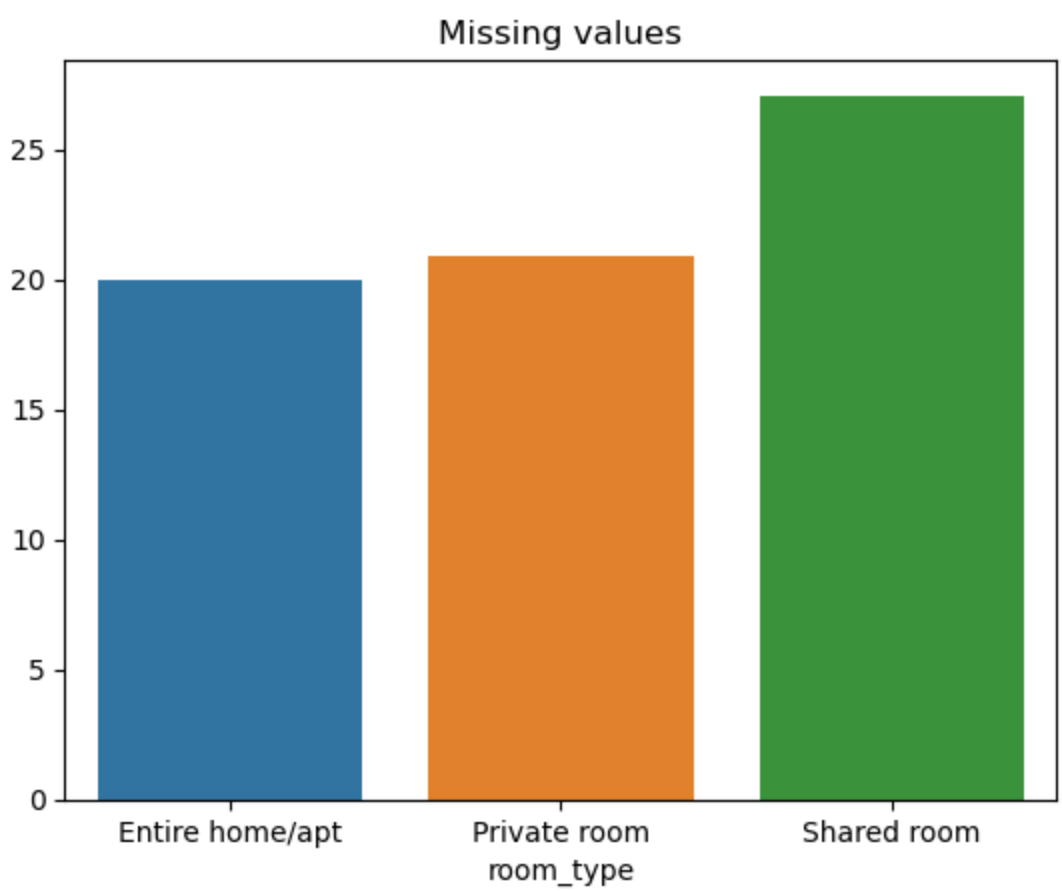
Each neighbourhood_group has about 19 % missing values in 'last_review' feature.

5.3 Missing values Analysis ('room_type' feature)

In [37]:
```
# Count of 'room_type' with missing values
mva1 = (mva.groupby('room_type').room_type.count()/abnb.groupby('room_type').room_type.c
mva1
```

Out[37]:
```
room_type
Entire home/apt     19.981109
Private room        20.877004
Shared room         27.068966
Name: room_type, dtype: float64
```

In [38]:
```
plt.title('Missing values')
sns.barplot(x = mva1.index, y = mva1.values)
plt.show()
```



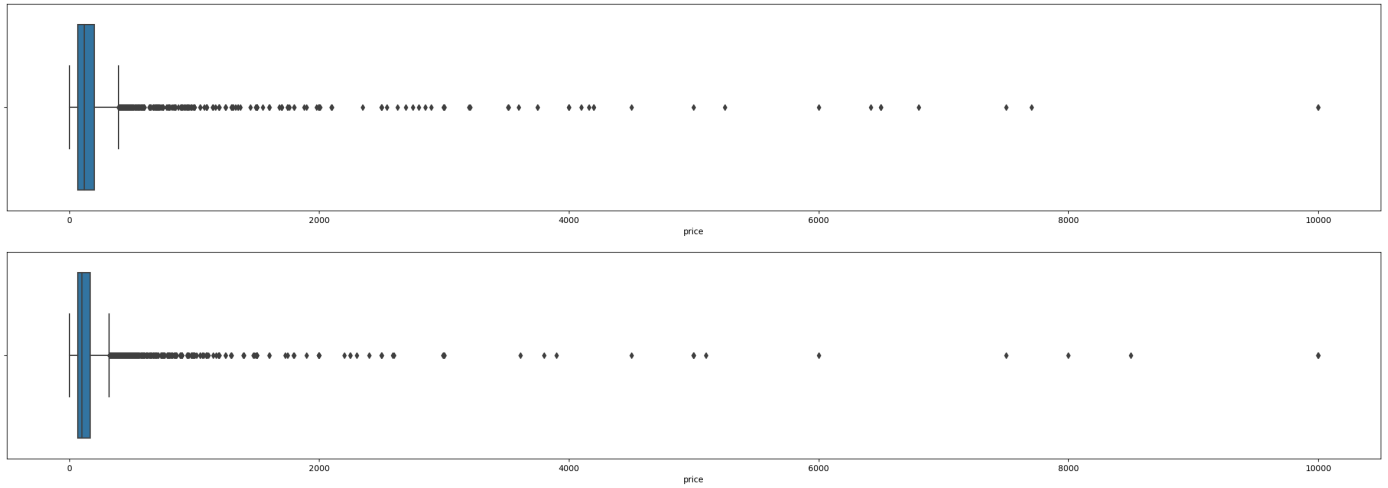
Observation

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

In [39]:
```
print('Mean and Median of prices with last_review feature missing')
print('Mean   = ', abnb[abnb['last_review'].isnull()].price.mean())
print('Median = ', abnb[abnb['last_review'].isnull()].price.median())

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

In [40]:
```
plt.figure(figsize=(30,10))

plt.subplot(2,1,1)
sns.boxplot(data = abnb[abnb['last_review'].isnull()] , x = 'price',  width= .8)

plt.subplot(2,1,2)
sns.boxplot(data = abnb[abnb['last_review'].notnull()] , x = 'price',  width= .8)

plt.show()
```



Observations:

Absence of the 'last_review' feature results in higher pricing.

Shared rooms tend to receive fewer reviews.

High prices are associated with lower review likelihood.

This analysis suggests that the missing values are not indicative of a random absence (MCAR).

6. Univariate Analysis

6.1 Host Name and Value

In [41]:
```
abnb.name.value_counts()
```

Out[41]:
```
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
```

In [42]:
```
abnb.host_name.value_counts()
```

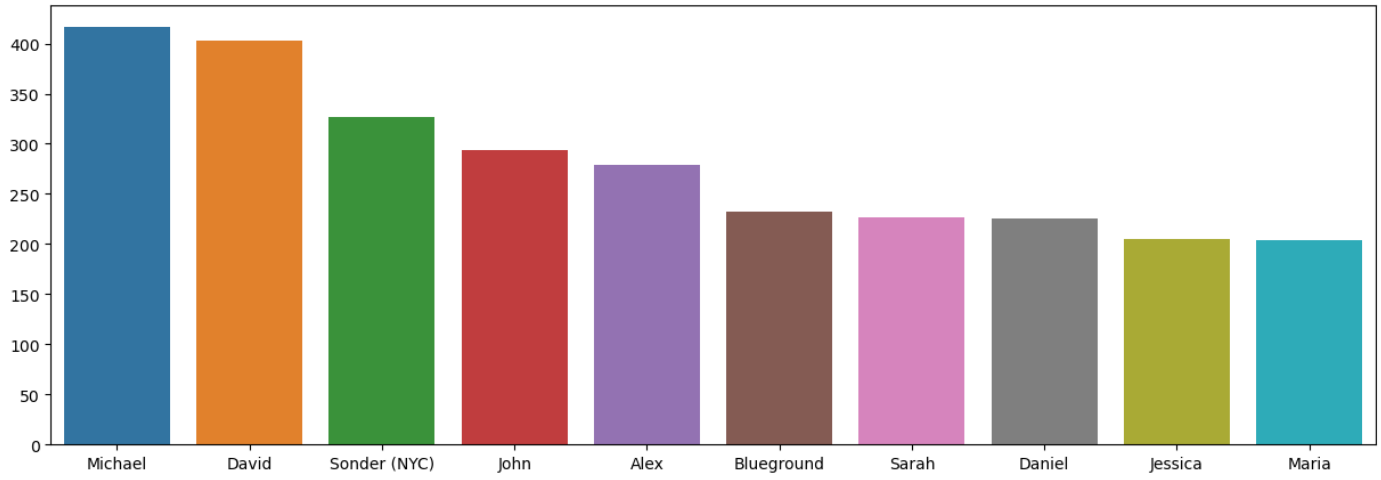
Out[42]:
```
Michael              417
David                403
Sonder (NYC)         327
John                 294
Alex                 279
                    ...
Rhonycs                1
Brandy-Courtney        1
Shanthony              1
Aurore And Jamila      1
Ilgar & Aysel          1
Name: host_name, Length: 11452, dtype: int64
```

In [43]:
```
abnb.host_name.value_counts().index[:10]
```

Out[43]:
```
Index(['Michael', 'David', 'Sonder (NYC)', 'John', 'Alex', 'Blueground',
```

```
       'Sarah', 'Daniel', 'Jessica', 'Maria'],
      dtype='object')
```

In [44]:
```
# Top 10 hosts
plt.figure(figsize=(15,5))
sns.barplot(x = abnb.host_name.value_counts().index[:10] , y = abnb.host_name.value_coun
plt.show()
```



6.2 Neighbourhood Group

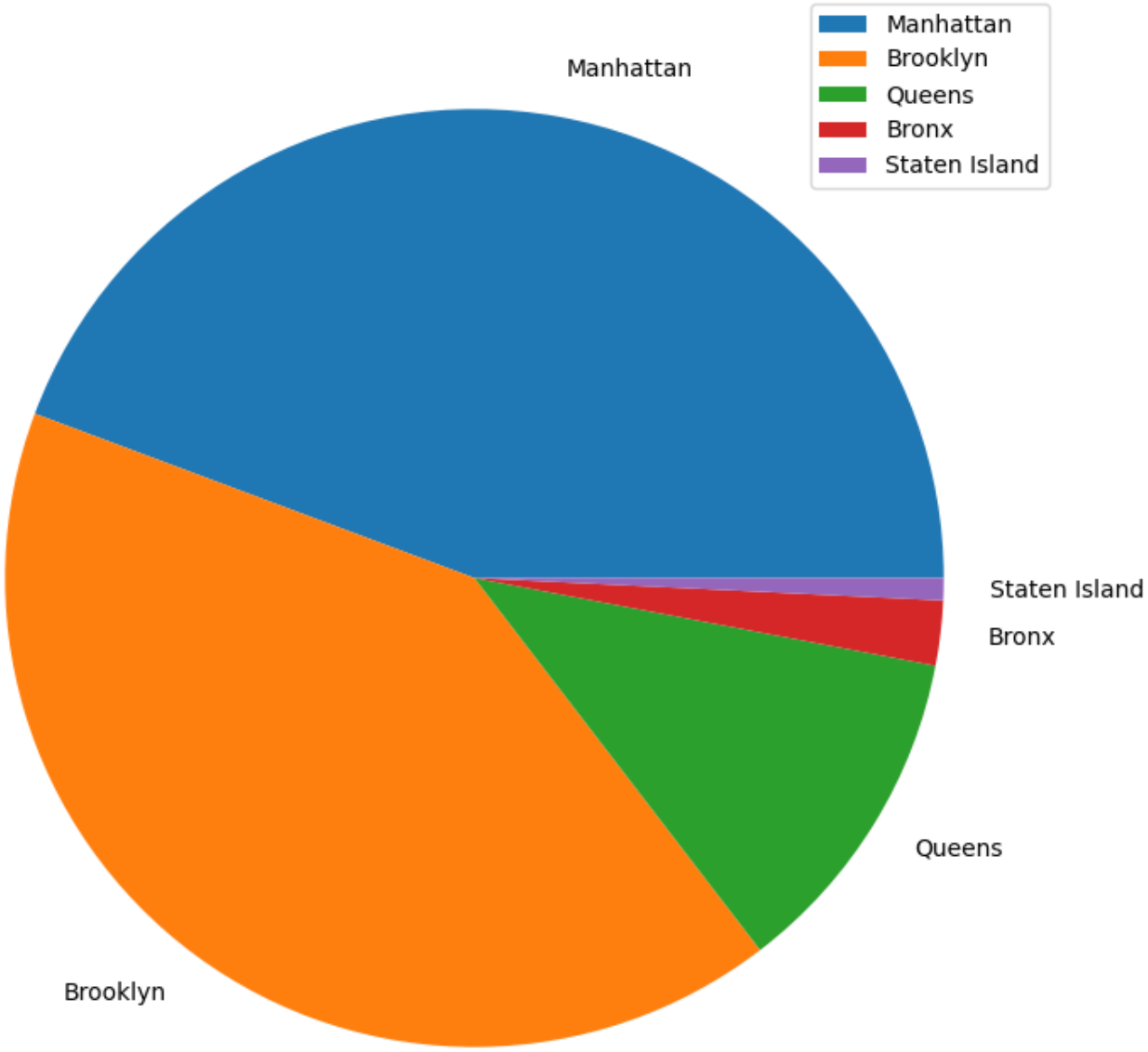
In [45]:
```
abnb.neighbourhood_group.value_counts()
```

Out[45]:
```
Manhattan        21661
Brooklyn         20104
Queens            5666
Bronx             1091
Staten Island      373
Name: neighbourhood_group, dtype: int64
```

In [46]:
```
abnb.neighbourhood_group.value_counts(normalize= True) * 100
```

Out[46]:
```
Manhattan        44.301053
Brooklyn         41.116679
Queens           11.588097
Bronx             2.231312
Staten Island     0.762859
Name: neighbourhood_group, dtype: float64
```

In [47]:
```
plt.figure(figsize=(9,9))
plt.pie(x = abnb.neighbourhood_group.value_counts(normalize= True) * 100,labels = abnb.n
plt.legend()
plt.show()
```



What are the neighbourhoods they need to target?

85 % of the listing are Manhattan and Brooklyn neighbourhood_group

6.3 Neighbourhood

In [48]:
```
abnb.neighbourhood.value_counts()
```

Out[48]:
```
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
```

6.4 Room Type

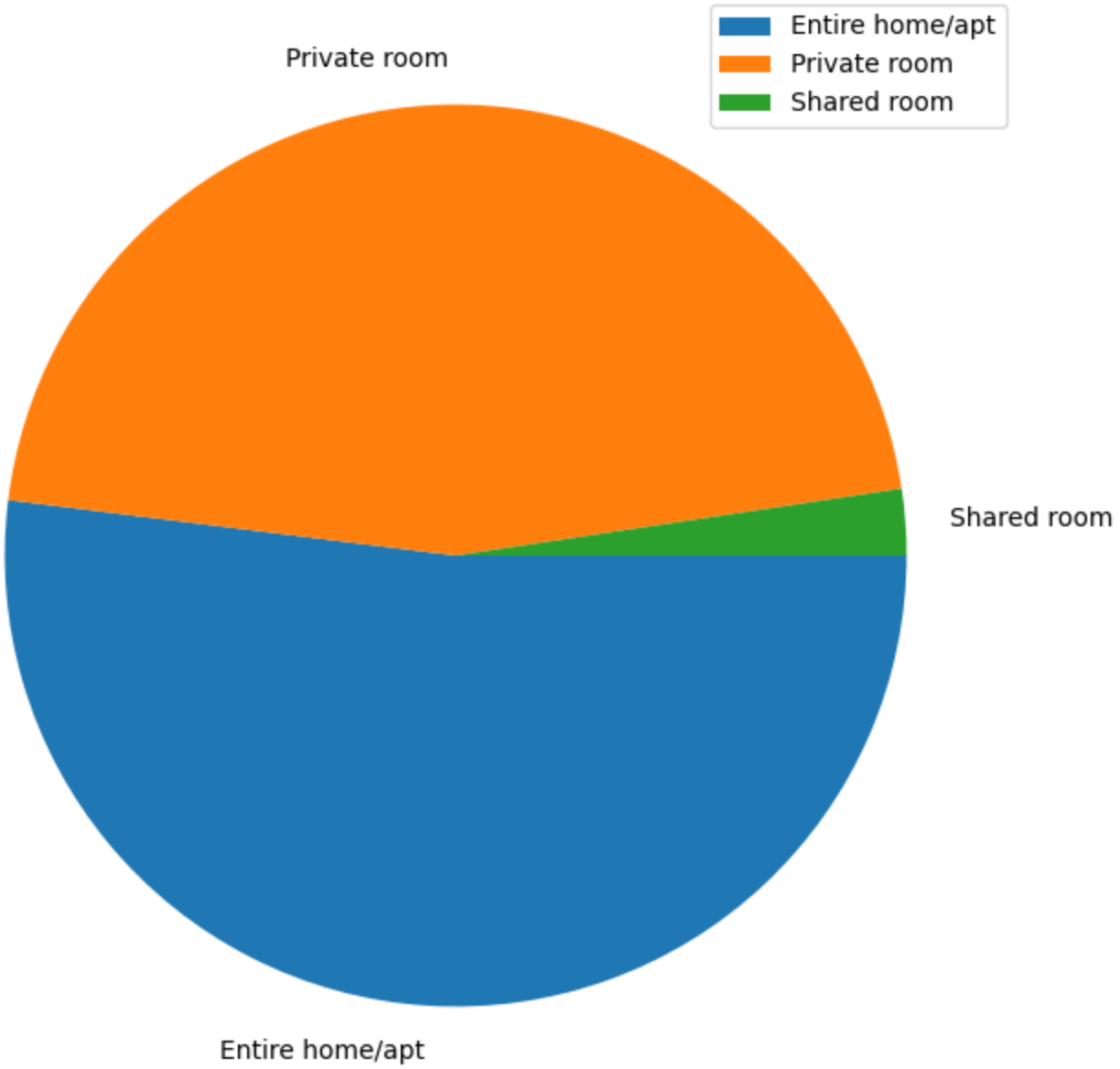
In [49]:
```
abnb.room_type.value_counts()
```

Out[49]:
```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

In [50]:
```
abnb.room_type.value_counts(normalize=True)*100
```

Out[50]:
```
Entire home/apt    51.966459
Private room       45.661111
Shared room         2.372431
Name: room_type, dtype: float64
```

In [51]:
```
plt.figure(figsize=(8,8))
plt.pie(x = abnb.room_type.value_counts(normalize= True) * 100,labels = abnb.room_type.v
plt.legend()
plt.show()
```



Observation

Entire home/apt is most listed in Airbnb.

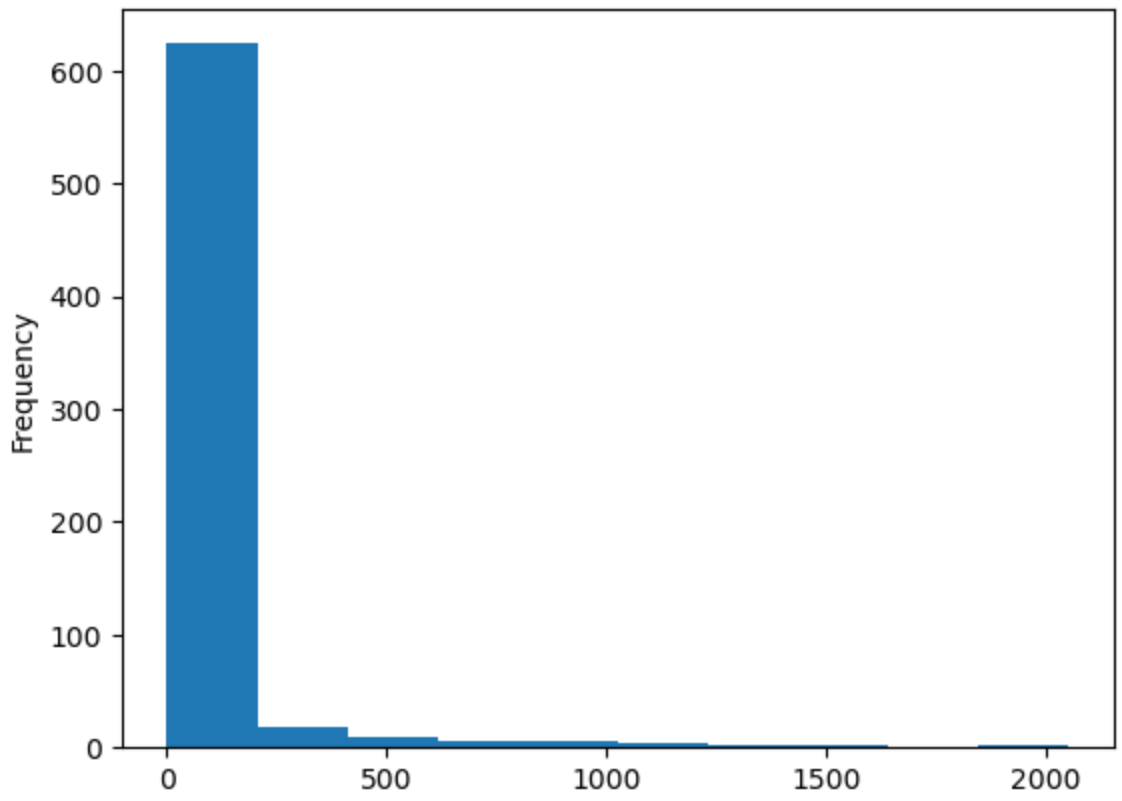
6.5 Price

In [52]:
```
abnb.price.value_counts()
```

Out[52]:
```
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
```

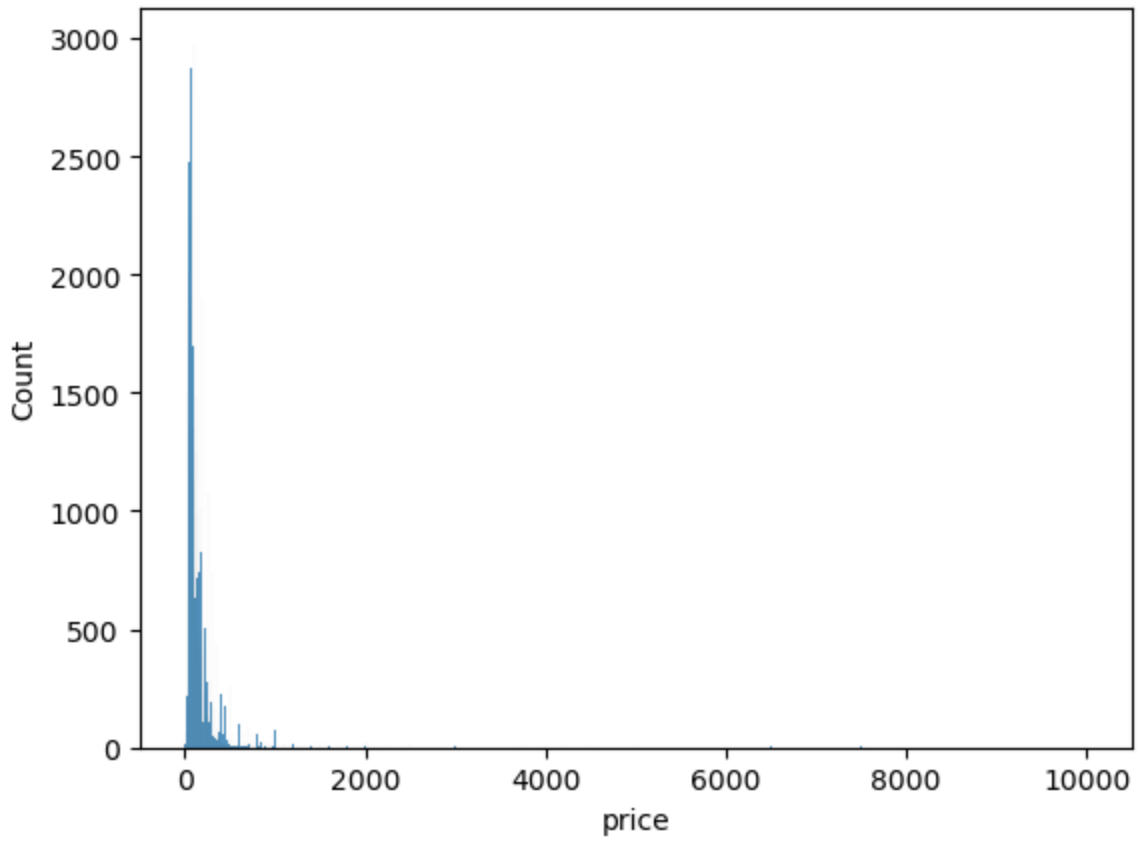
In [53]:
```
abnb.price.value_counts().plot.hist()
```

Out[53]:
```
<Axes: ylabel='Frequency'>
```



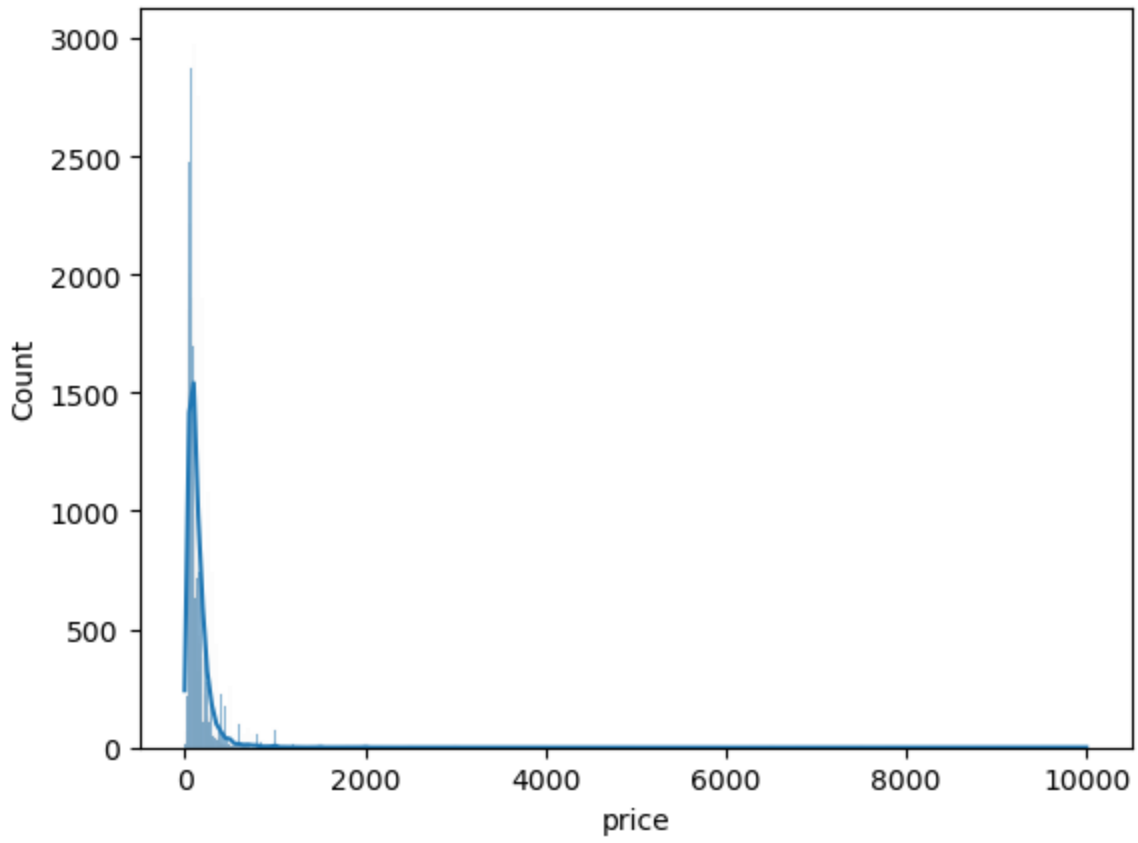
In [54]:
```
sns.histplot(data = abnb.price)
```

Out[54]:
```
<Axes: xlabel='price', ylabel='Count'>
```



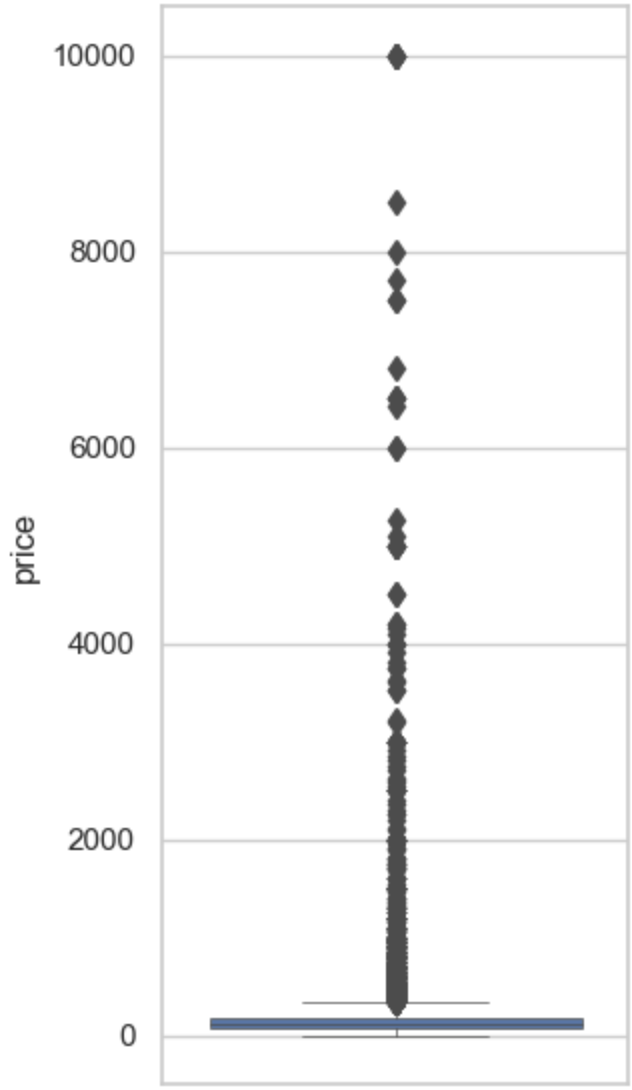
In [55]:
```
sns.histplot(data = abnb.price,kde = True)
```

Out[55]:
```
<Axes: xlabel='price', ylabel='Count'>
```



In [56]:
```
plt.figure(figsize=(3,7))
sns.set_theme(style="whitegrid")
#tips = sns.load_dataset("tips")
sns.boxplot(y = abnb.price,width=0.8,
    dodge=True,
```

```
    fliersize=6,
    linewidth=.5,
    whis=1.5,
    color=None)
plt.show()
```



6.6 Minimum Nights

In [57]:
```
abnb.minimum_nights.value_counts()
```

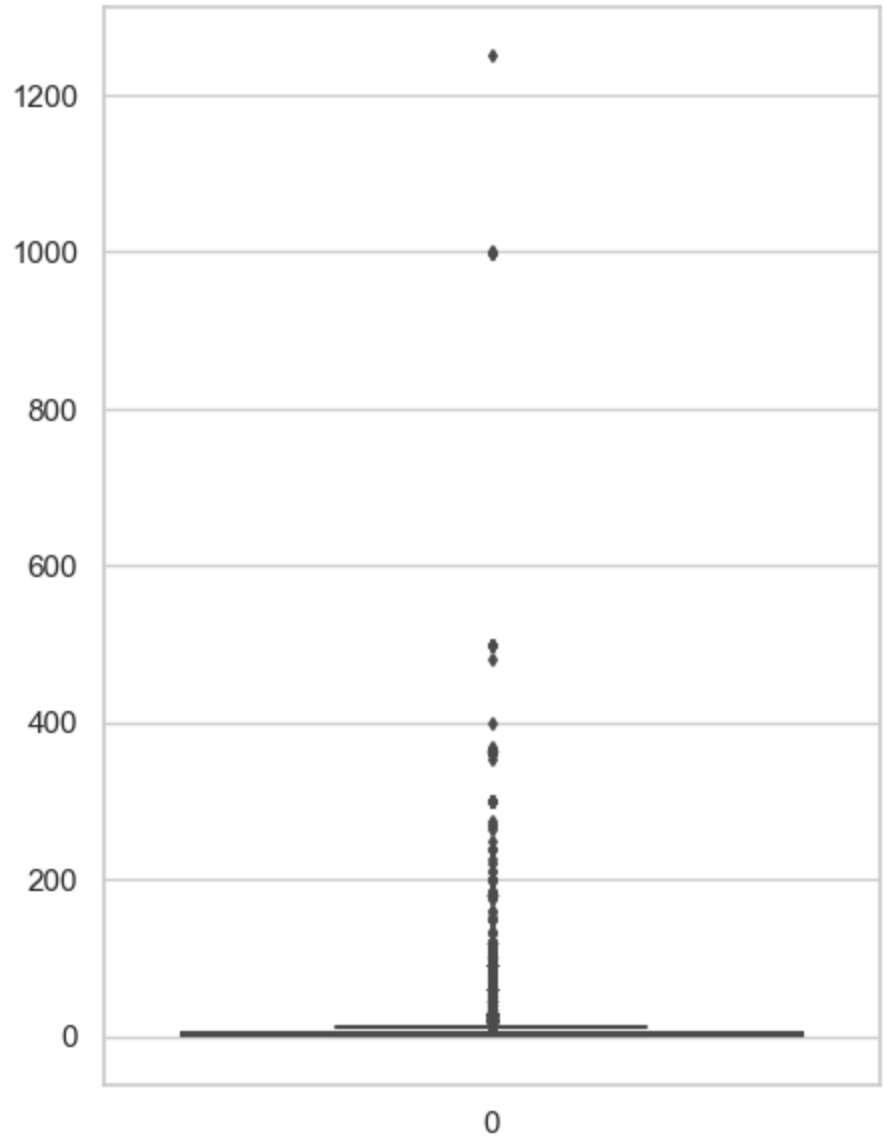
Out[57]:
```
1      12720
2      11696
3       7999
30      3760
4       3303
       ...
186        1
366        1
68         1
87         1
36         1
Name: minimum_nights, Length: 109, dtype: int64
```

In [58]:
```
abnb.minimum_nights.describe()
```

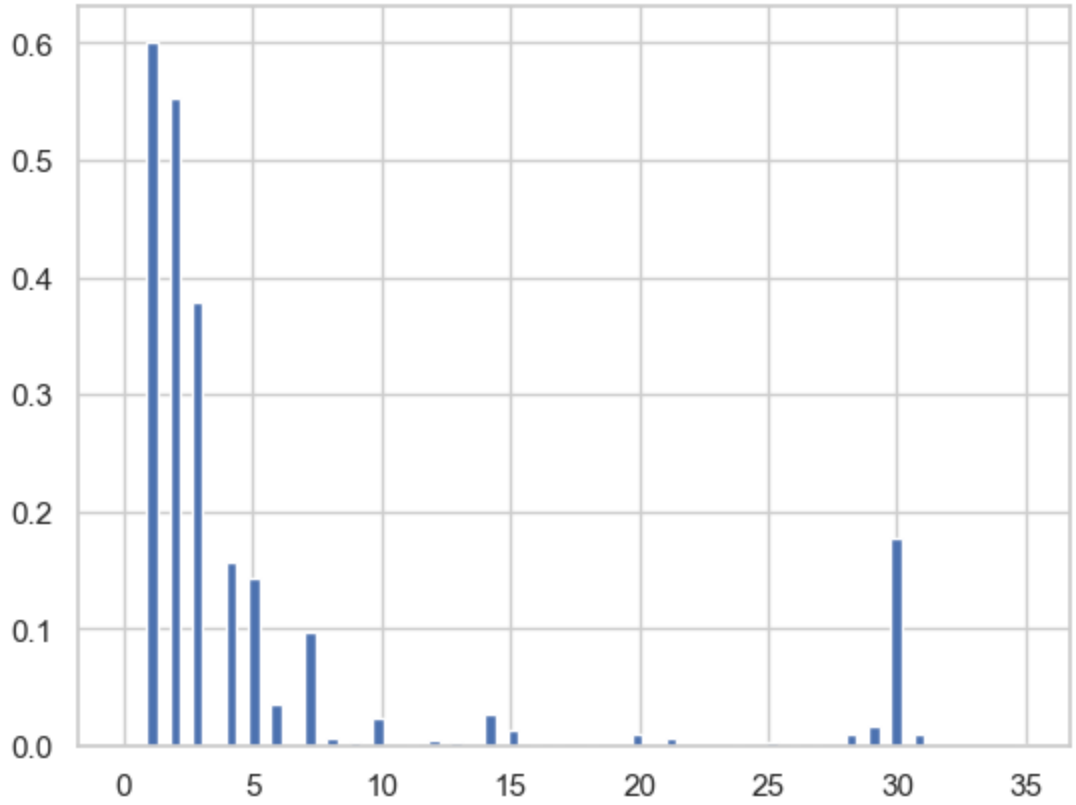
Out[58]:
```
count    48895.000000
mean         7.029962
std         20.510550
min          1.000000
```

```
25%           1.000000
50%           3.000000
75%           5.000000
max        1250.000000
Name: minimum_nights, dtype: float64
```

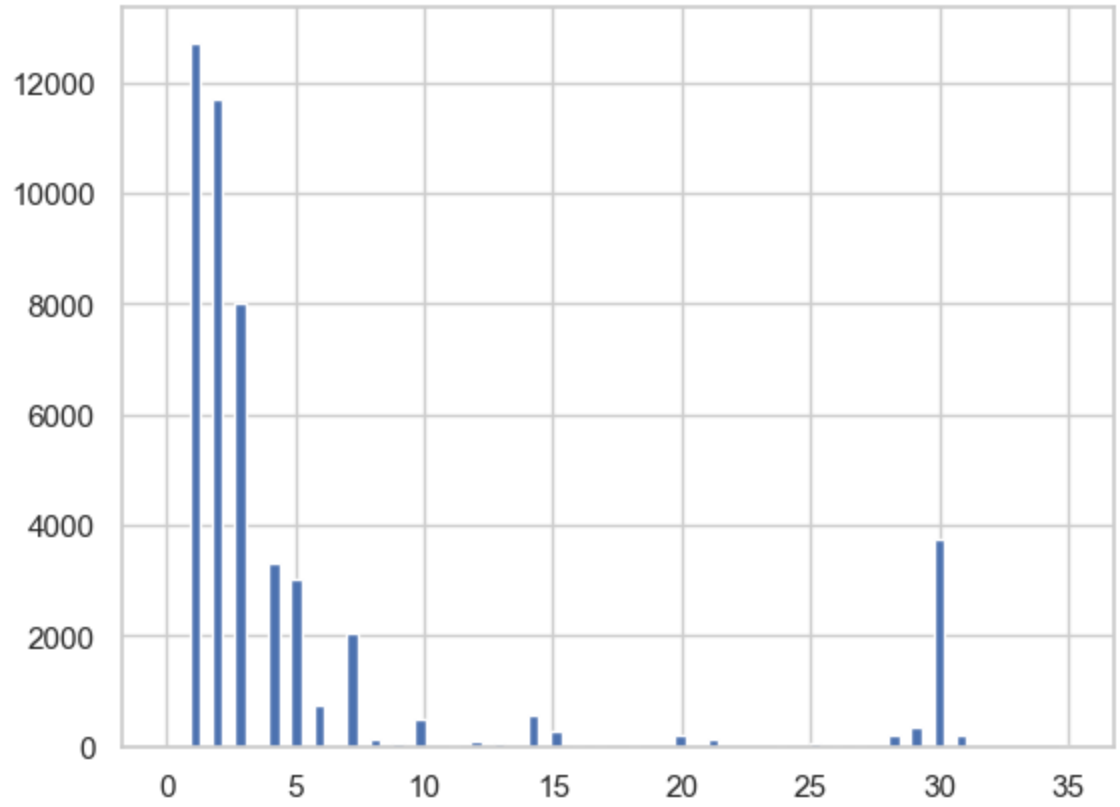
In [59]:
```
plt.figure(figsize=(5,7))
sns.boxplot(data = abnb.minimum_nights,fliersize=3)
plt.show()
```



In [60]:
```
plt.hist(data = abnb, x = 'minimum_nights',bins=80,range=(0,35),density=True)
plt.show()
```



In [61]:
```
plt.hist(data = abnb, x = 'minimum_nights',bins=80,range=(0,35))
plt.show()
```



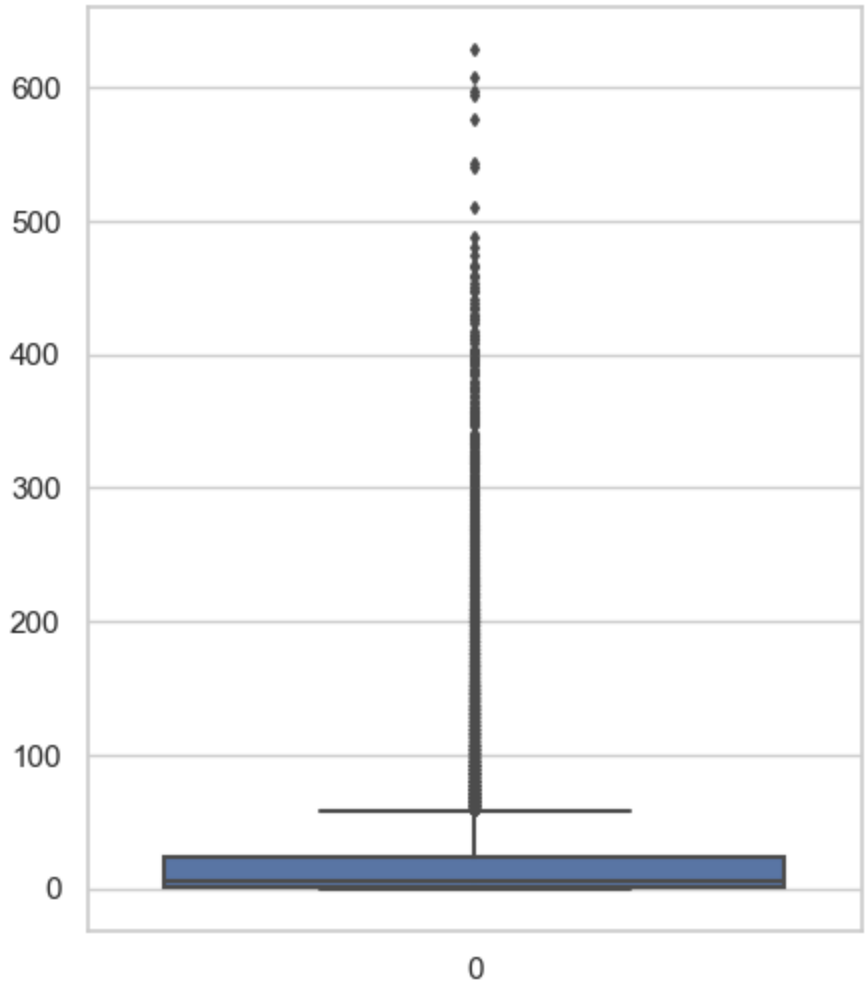
6.7 Number of Reviews

In [62]:
```
abnb.number_of_reviews.describe()
```

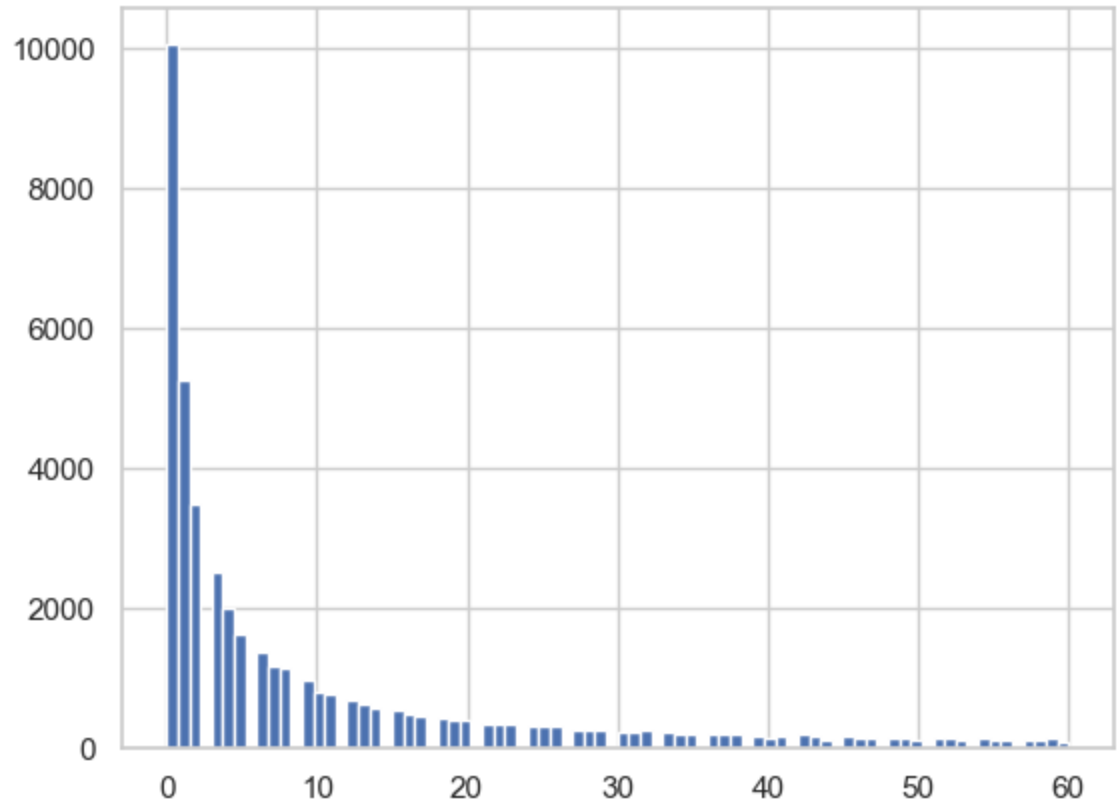
Out[62]:
```
count    48895.000000
mean        23.274466
std         44.550582
min          0.000000
```

```
25%           1.000000
50%           5.000000
75%          24.000000
max         629.000000
Name: number_of_reviews, dtype: float64
```

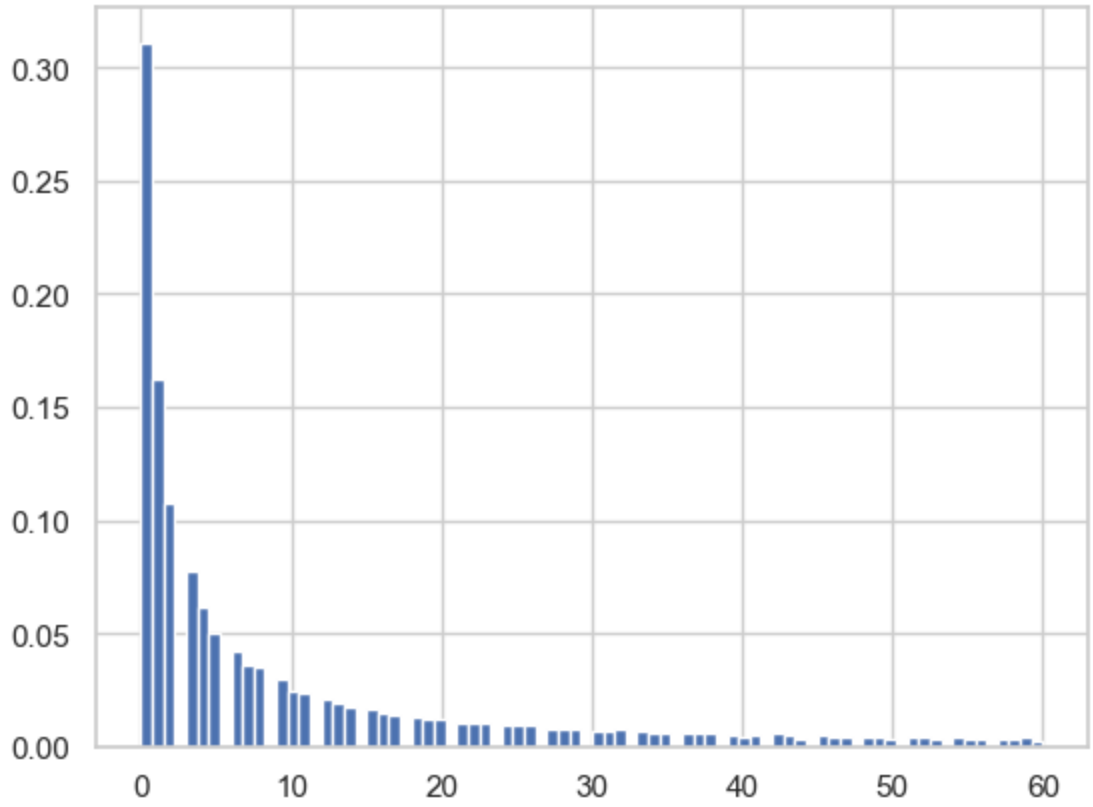
In [63]:
```
plt.figure(figsize=(5,6))
sns.boxplot(data = abnb.number_of_reviews,fliersize=3)
plt.show()
```



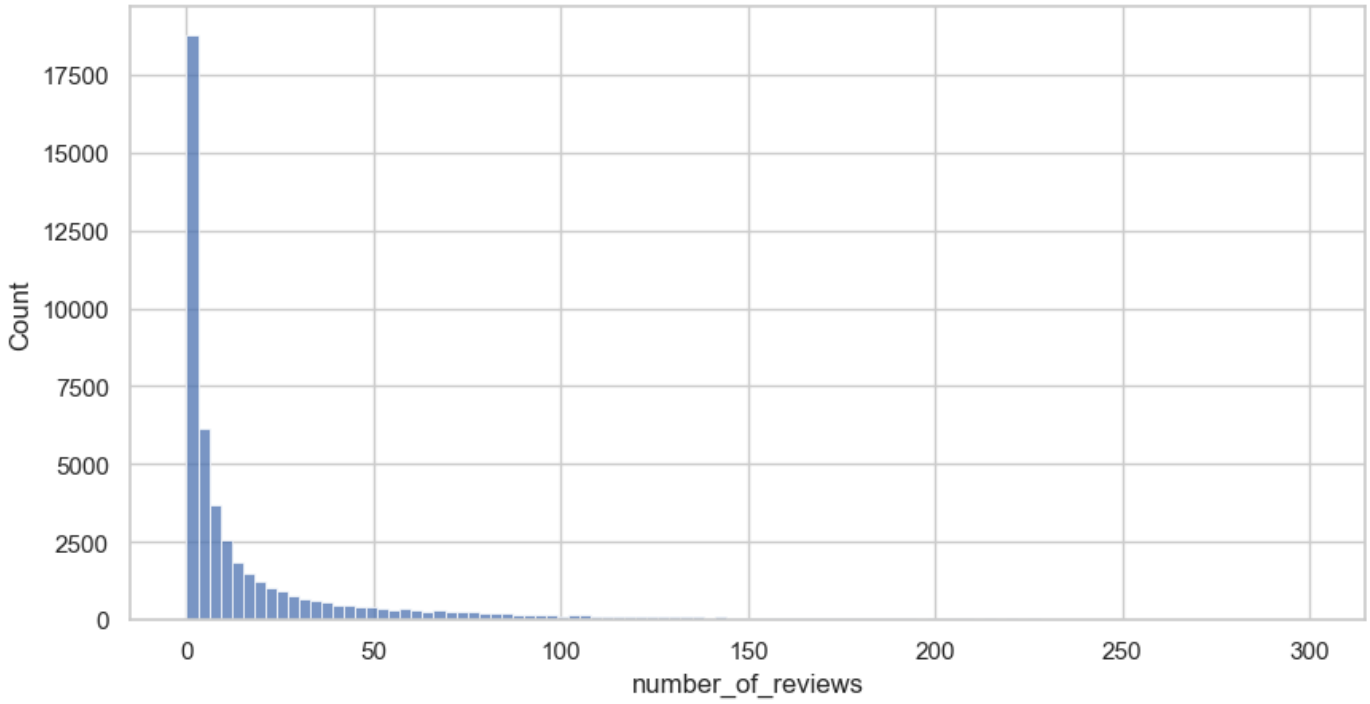
In [64]:
```
plt.hist(data = abnb, x = 'number_of_reviews',bins=80,range=(0,60))
plt.show()
```



In [65]:
```
plt.hist(data = abnb, x = 'number_of_reviews',bins=80,range=(0,60),density=True)
plt.show()
```



In [66]:
```
plt.figure(figsize = (10,5))
sns.histplot(data = abnb, x = 'number_of_reviews',bins=100,binrange=(0,300))
plt.show()
```

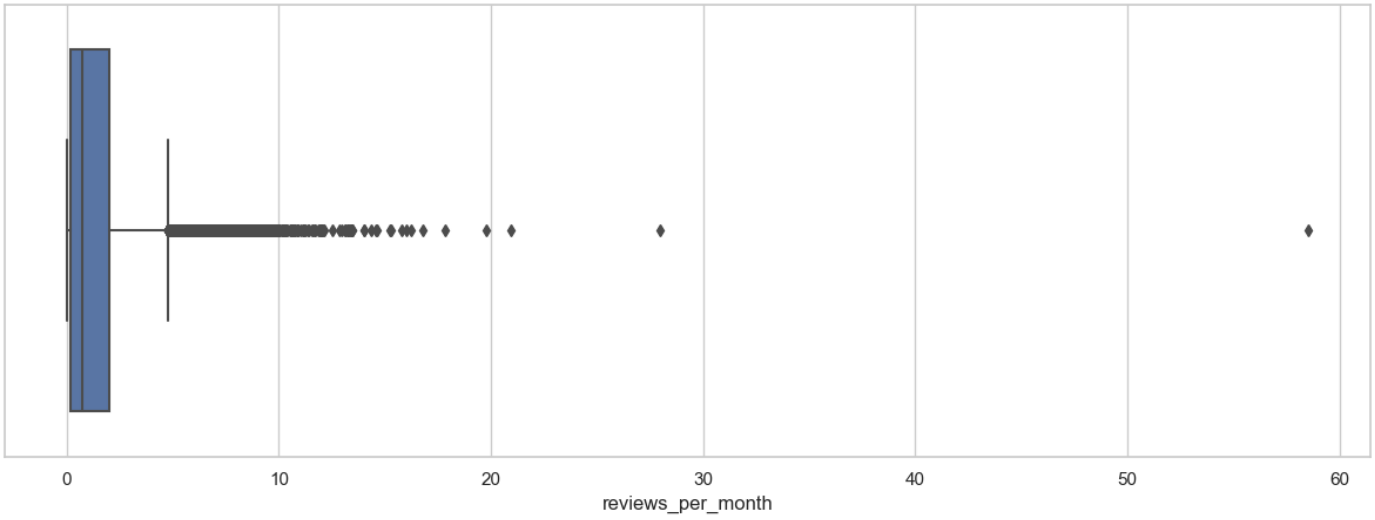


6.8 Reviews per Month

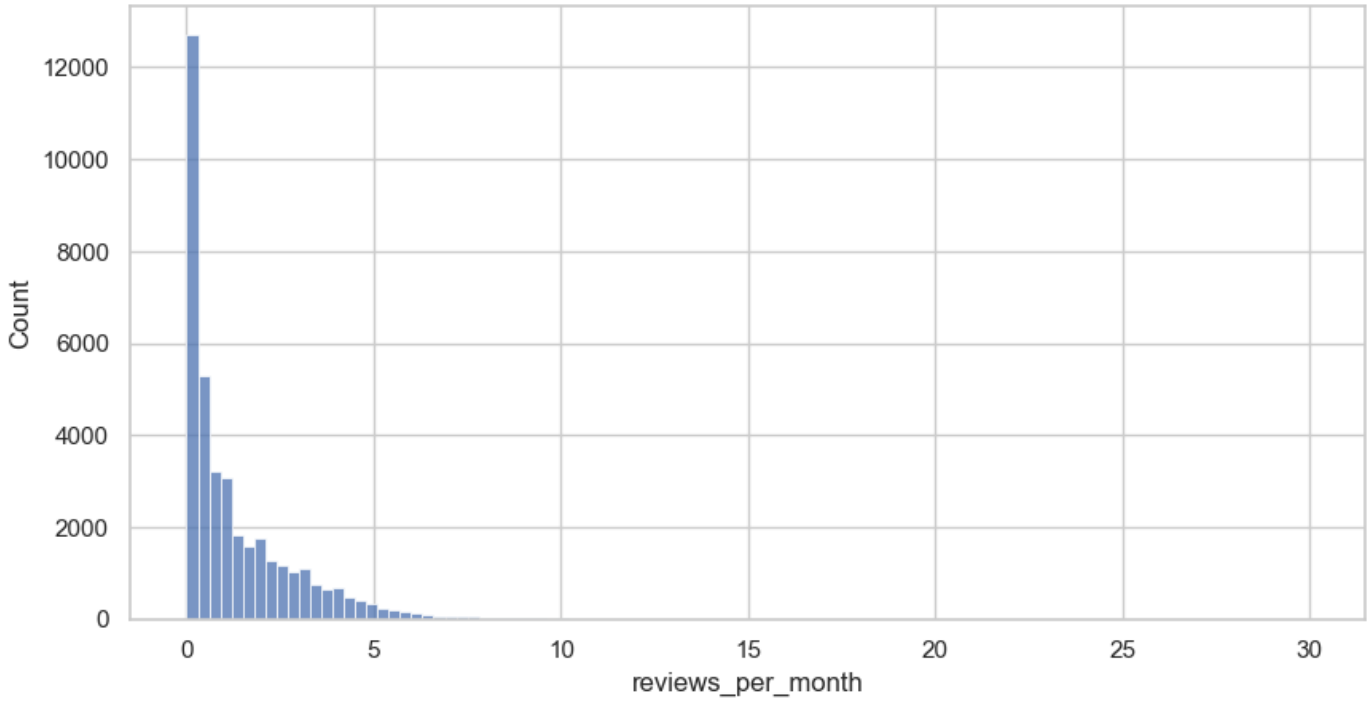
In [67]:
```
abnb.reviews_per_month.describe()
```

Out[67]:
```
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
```

In [68]:
```
plt.figure(figsize = (15,5))
sns.boxplot(data = abnb , x = 'reviews_per_month')
plt.show()
```



In [69]:
```
plt.figure(figsize = (10,5))
sns.histplot(data = abnb, x = 'reviews_per_month',bins=100,binrange=(0,30))
plt.show()
```

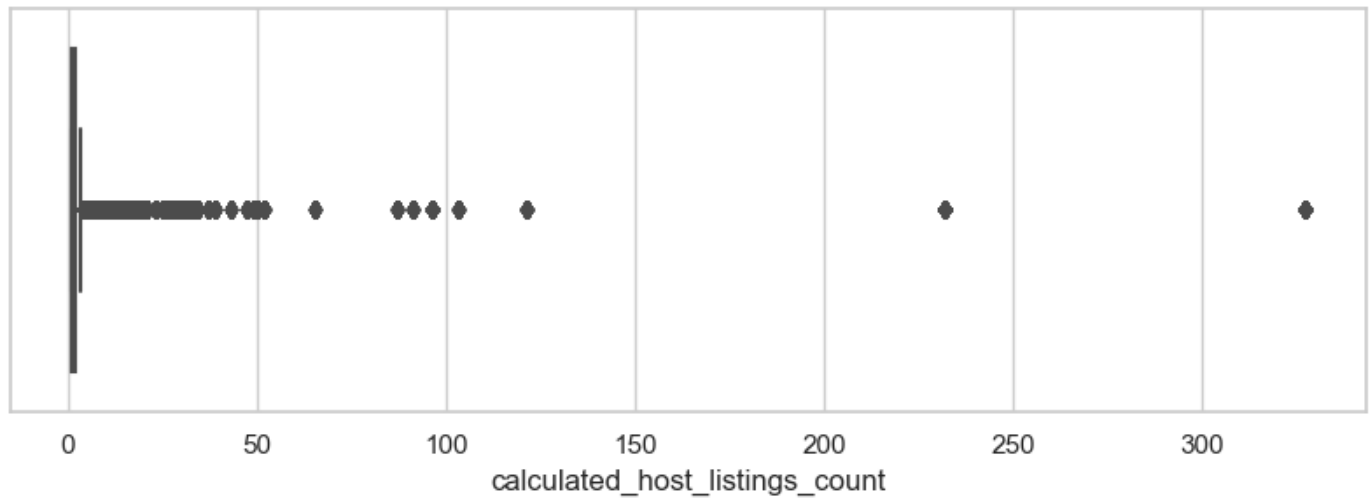


6.9 Calculated Host Listings Count

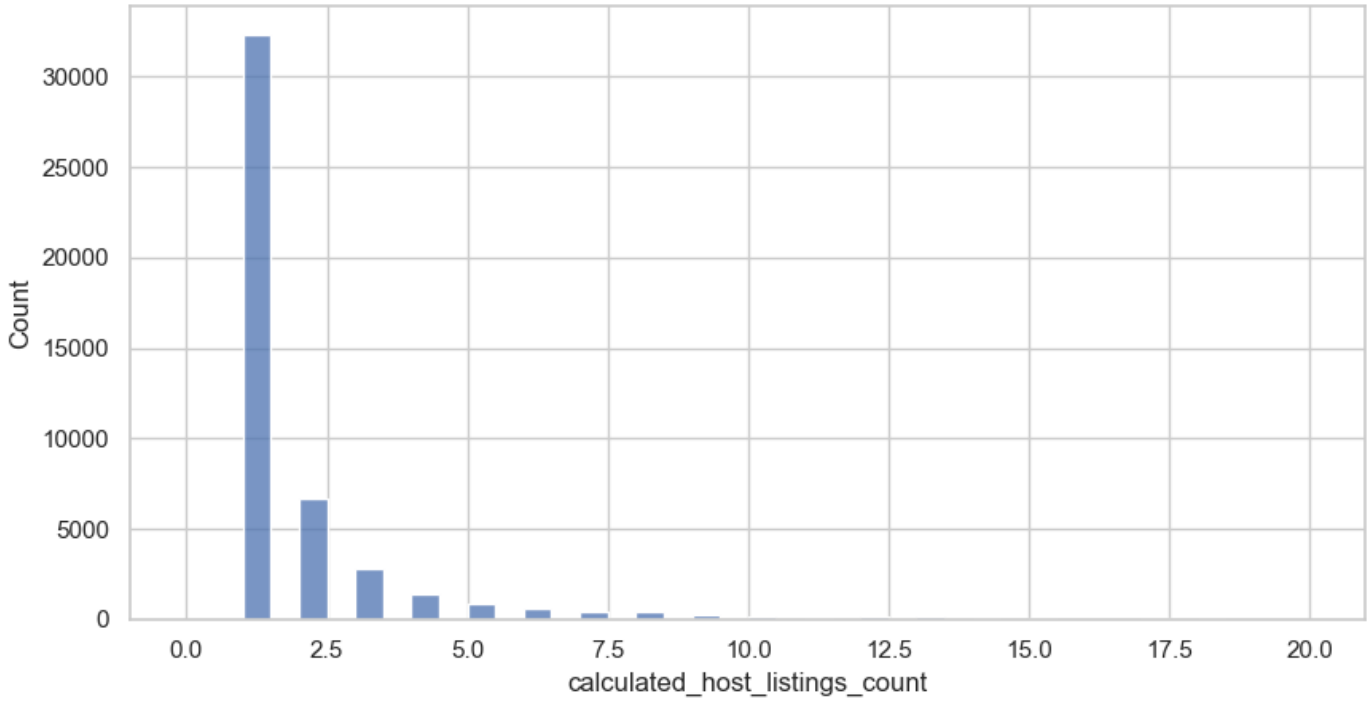
In [70]:
```
abnb.calculated_host_listings_count.describe()
```

Out[70]:
```
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
```

In [71]:
```
plt.figure(figsize = (10,3))
sns.boxplot(data = abnb , x = 'calculated_host_listings_count')
plt.show()
```



In [72]:
```
plt.figure(figsize = (10,5))
sns.histplot(data = abnb, x = 'calculated_host_listings_count',bins=40,binrange=(0,20))
plt.show()
```

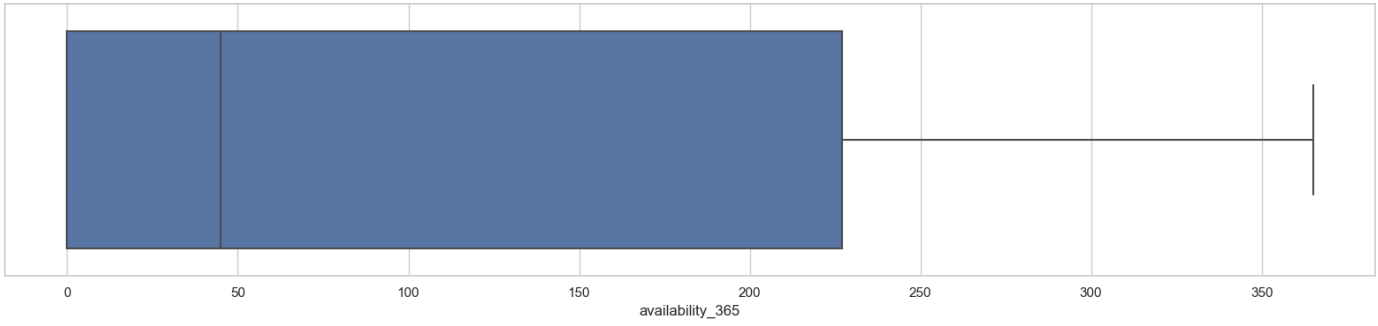


6.10 Availability_365

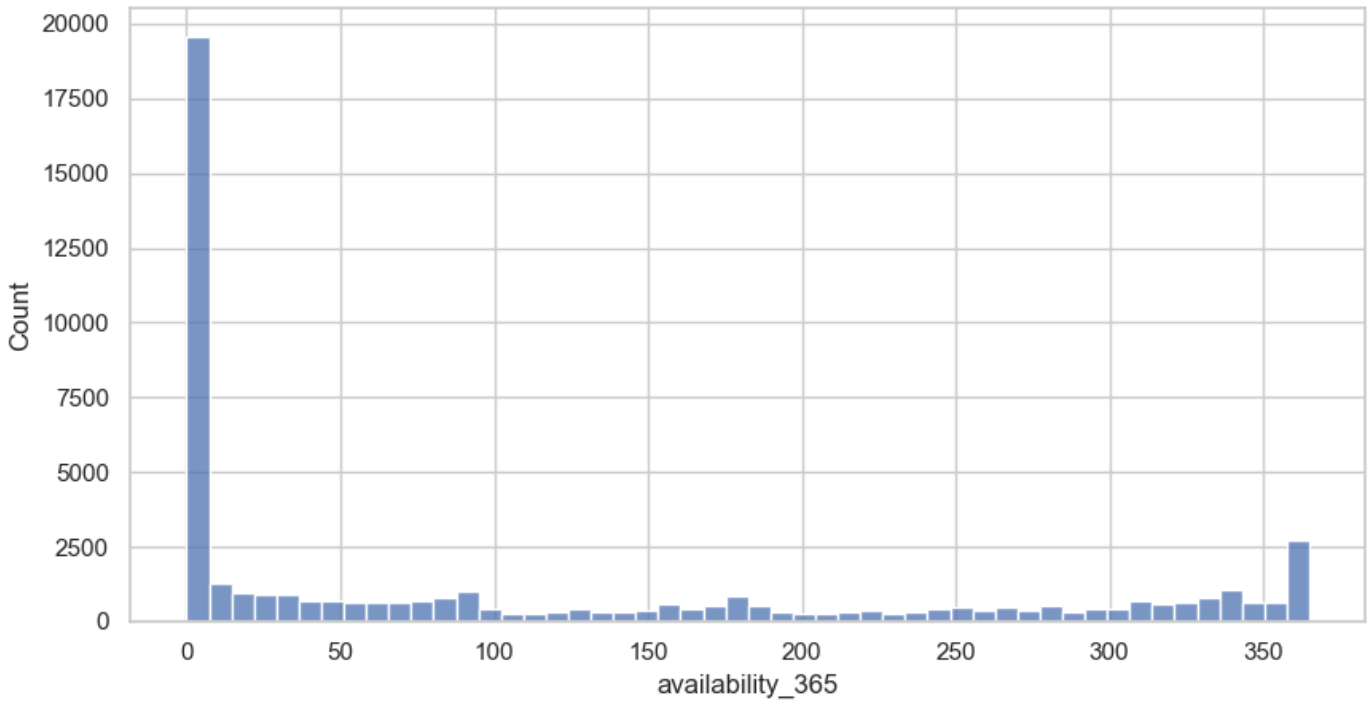
In [73]:
```
abnb.availability_365.describe()
```

Out[73]:
```
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
```

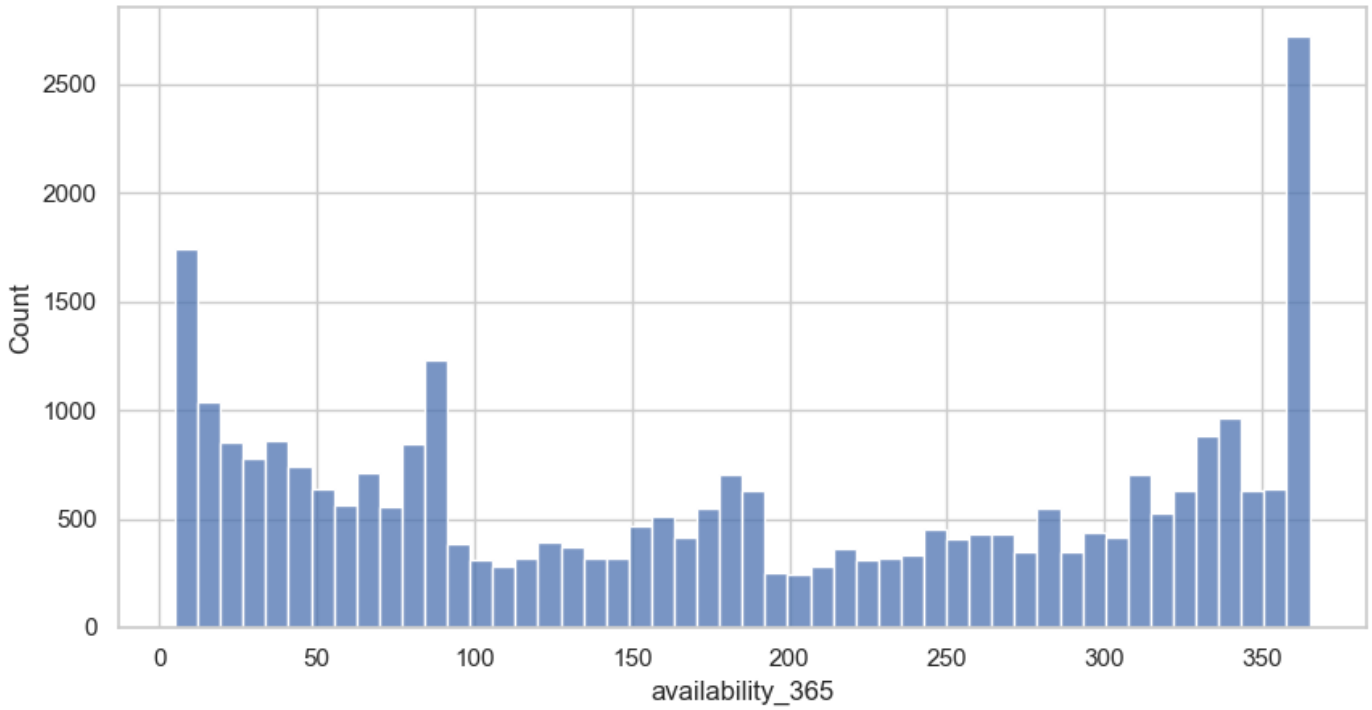
In [74]:
```
plt.figure(figsize = (20,4))
sns.boxplot(data = abnb , x = 'availability_365')
plt.show()
```



In [75]:
```
plt.figure(figsize = (10,5))
sns.histplot(data = abnb, x = 'availability_365',bins=50,binrange=(0,365))
plt.show()
```



In [76]:
```
plt.figure(figsize = (10,5))
sns.histplot(data = abnb, x = 'availability_365',bins=50,binrange=(5,365))
plt.show()
```



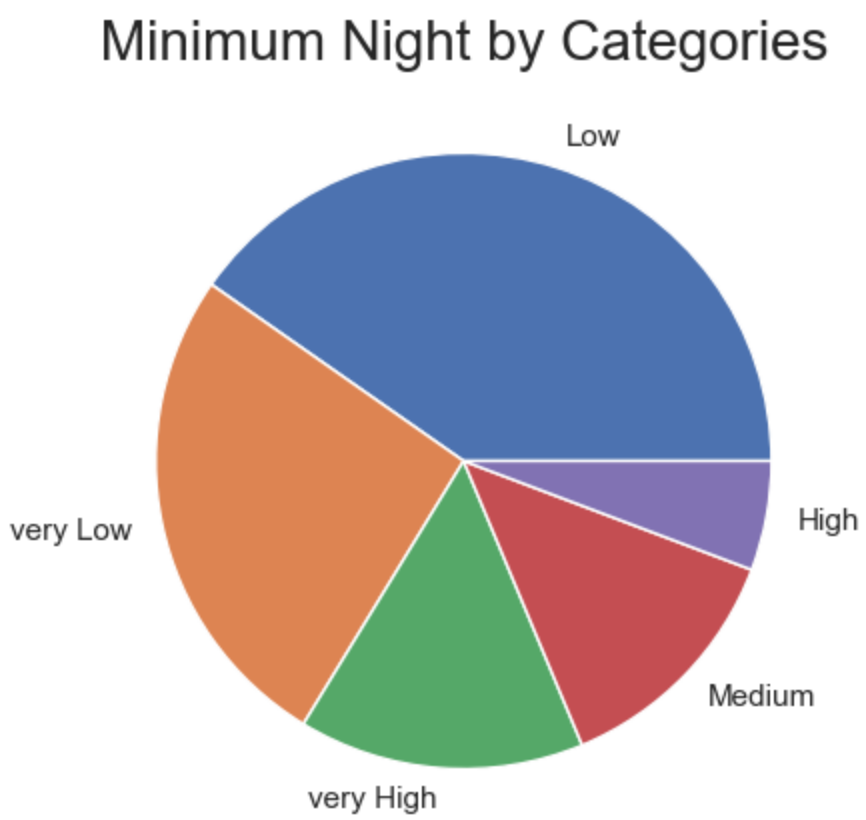
6.11 Minimum Night Categories

In [77]:
```
abnb.minimum_night_categories.value_counts(normalize= True)*100
```

Out[77]:
```
Low          40.280192
very Low     26.014930
very High    14.997444
Medium       12.960425
High          5.747009
Name: minimum_night_categories, dtype: float64
```

In [78]:
```
plt.figure(figsize=(10,5))
plt.title('Minimum Night by Categories', fontdict={'fontsize': 20})
```

```
plt.pie(x = abnb.minimum_night_categories.value_counts(),labels=abnb.minimum_night_categ
plt.show()
```



In [79]:
```
abnb.number_of_reviews.describe()
```

Out[79]:
```
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
```

6.12 Number of Reviews Categories

In [80]:
```
abnb.number_of_reviews_categories.value_counts()
```

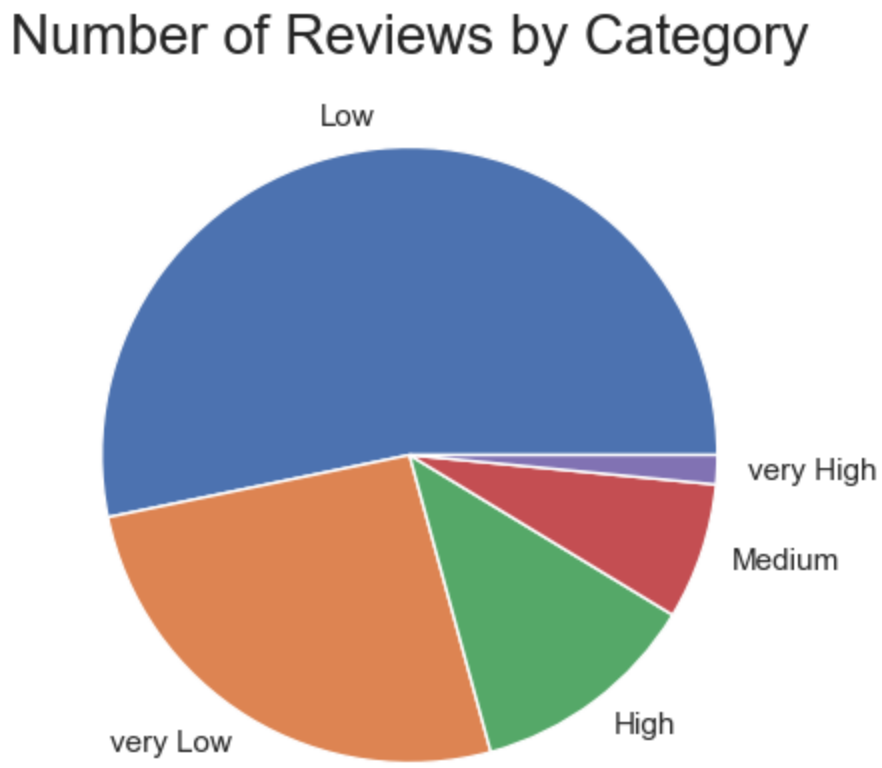
Out[80]:
```
Low          26032
very Low     12720
High          5893
Medium        3503
very High      747
Name: number_of_reviews_categories, dtype: int64
```

In [81]:
```
abnb.number_of_reviews_categories.value_counts(normalize=True)*100
```

Out[81]:
```
Low          53.240618
very Low     26.014930
High         12.052357
Medium        7.164332
very High     1.527764
Name: number_of_reviews_categories, dtype: float64
```

In [82]:
```
plt.figure(figsize=(10,5))
plt.title('Number of Reviews by Category', fontdict={'fontsize': 20})
```

```
plt.pie(x = abnb.number_of_reviews_categories.value_counts(),labels=abnb.number_of_revie
plt.show()
```



6.13 Price Categories

In [83]:
```
abnb['price_categories'].value_counts()
```

Out[83]:
```
Low          22998
very Low     12720
Medium        7556
High          5447
very High      174
Name: price_categories, dtype: int64
```

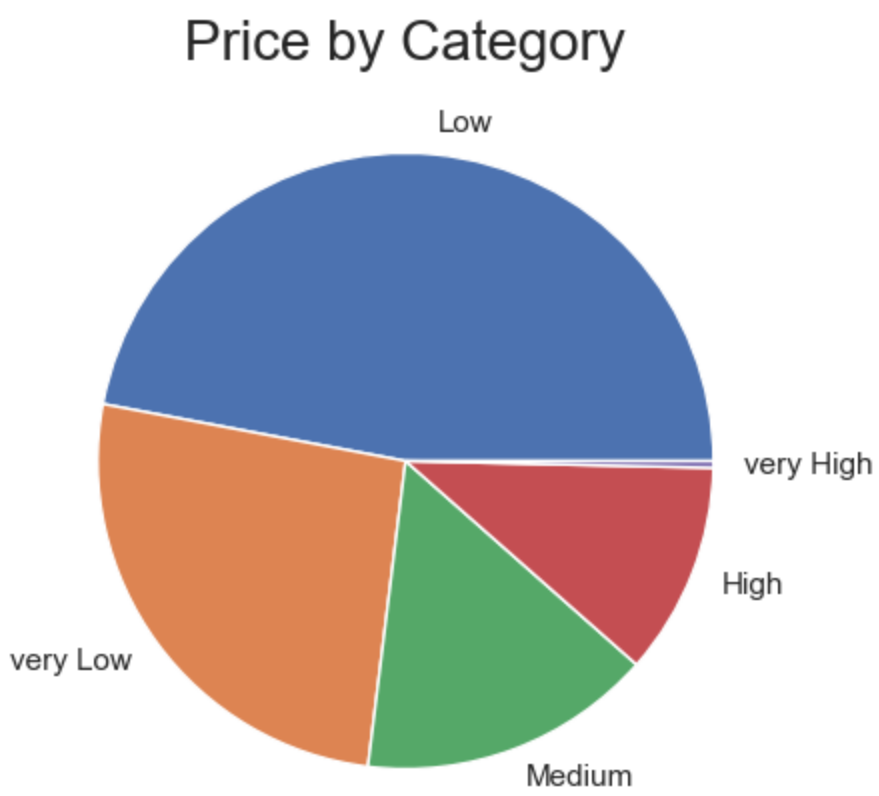
In [84]:
```
abnb['price_categories'].value_counts(normalize=True)*100
```

Out[84]:
```
Low          47.035484
very Low     26.014930
Medium       15.453523
High         11.140198
very High     0.355865
Name: price_categories, dtype: float64
```

In [85]:
```
abnb['price_categories'].describe()
```

Out[85]:
```
count      48895
unique         5
top          Low
freq       22998
Name: price_categories, dtype: object
```

In [86]:
```
plt.figure(figsize=(10,5))
plt.title('Price by Category', fontdict={'fontsize': 20})
plt.pie(x = abnb.price_categories.value_counts(),labels=abnb.price_categories.value_coun
plt.show()
```



In [87]:
```
47.035484+26.014930
```

Out[87]:
```
73.05041399999999
```

What is the pricing ranges preferred by customers?

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

7. Bivariate and Multivariate Analysis

7.1 Finding the correalations

In [88]:
```
abnb[numerical_columns].head()
```

Out[88]:

	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
2	150	3	0	NaN	1	365
3	89	1	270	4.64	1	194
4	80	10	9	0.10	1	0

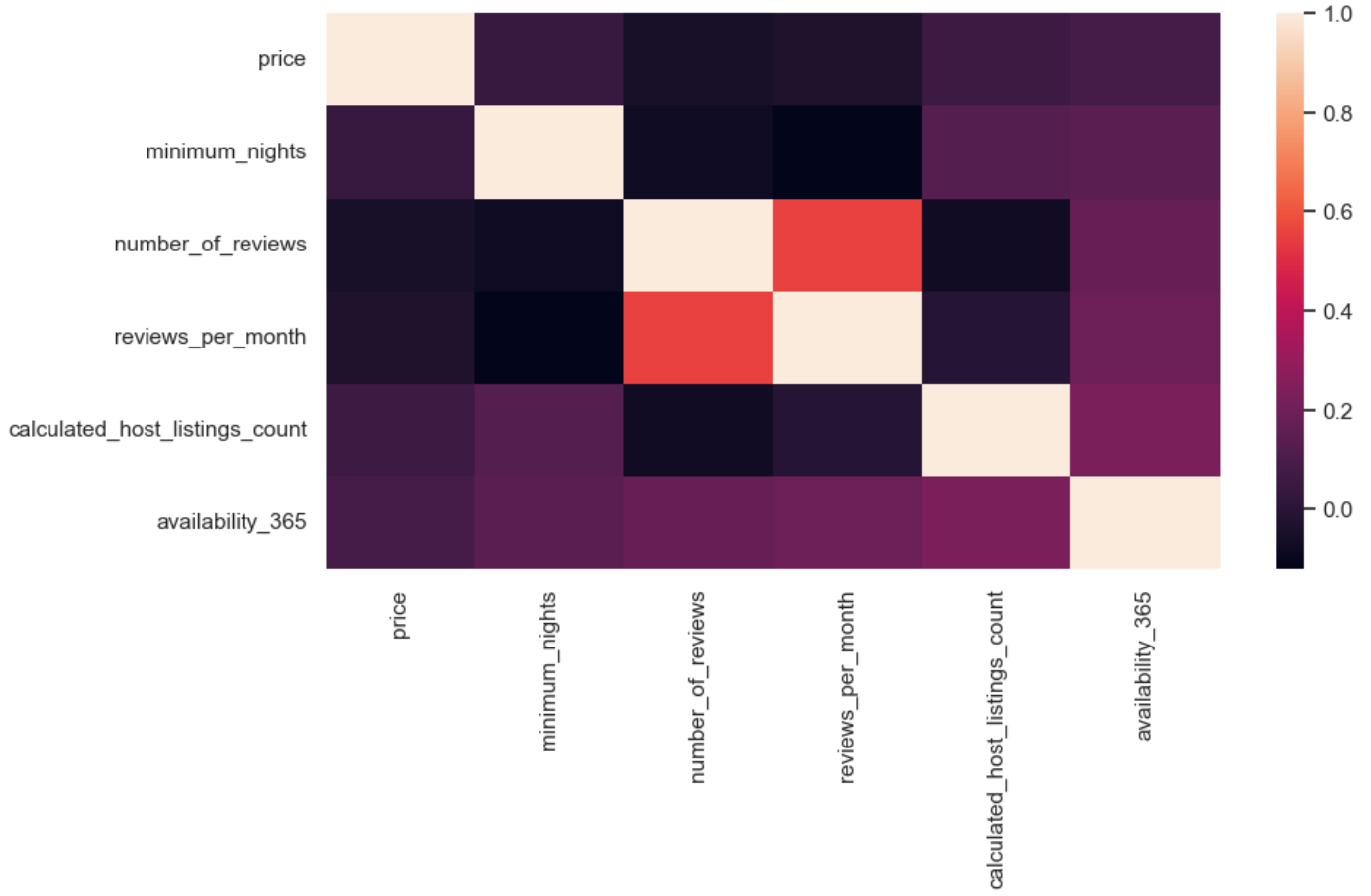
In [89]:
```
abnb[numerical_columns].corr()
```

Out[89]:

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_l
price	1.000000	0.042799	-0.047954	-0.030608	
minimum_nights	0.042799	1.000000	-0.080116	-0.121702	

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

In [90]:
```
plt.figure(figsize=(10,5))
sns.heatmap(data = abnb[numerical_columns].corr())
plt.show()
```



7.2 Finding Top correlations

In [91]:
```
corr_matrix = abnb[numerical_columns].corr().abs()

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

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

Out[91]:

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

In [92]:
```
sol
```

Out[92]:
```
number_of_reviews                reviews_per_month                0.549868
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
minimum_nights                   number_of_reviews                 0.080116
number_of_reviews                calculated_host_listings_count    0.072376
price                            calculated_host_listings_count    0.057472
                                 number_of_reviews                 0.047954
                                 minimum_nights                    0.042799
                                 reviews_per_month                 0.030608
reviews_per_month                calculated_host_listings_count    0.009421
dtype: float64
```

In [93]:
```
# Top meaningful correlations
sol[1:8]
```

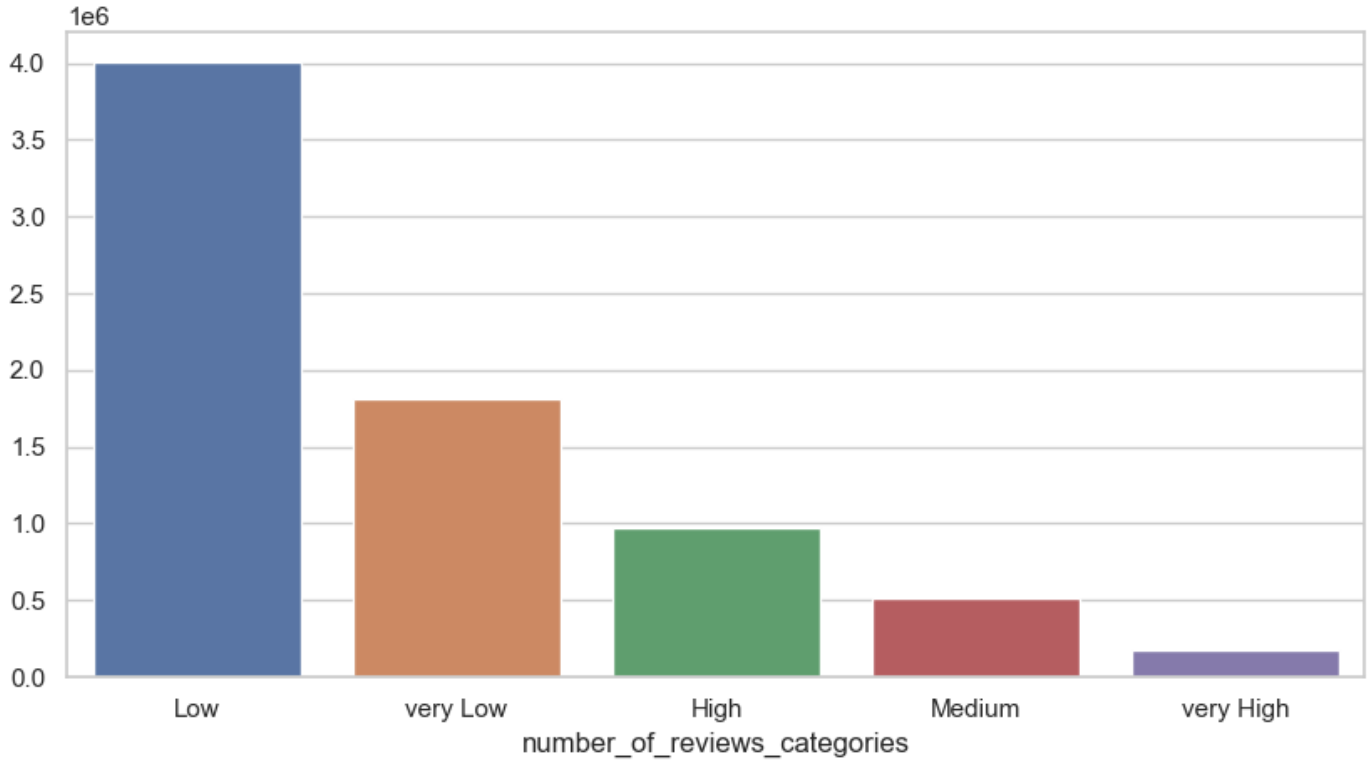
Out[93]:
```
calculated_host_listings_count   availability_365                  0.225701
reviews_per_month                availability_365                  0.185791
number_of_reviews                availability_365                  0.172028
minimum_nights                   availability_365                  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 by categories v/s Prices

In [94]:
```
# prices for each of reviews_categories
x1 = abnb.groupby('number_of_reviews_categories').price.sum().sort_values(ascending = Fa
x1
```

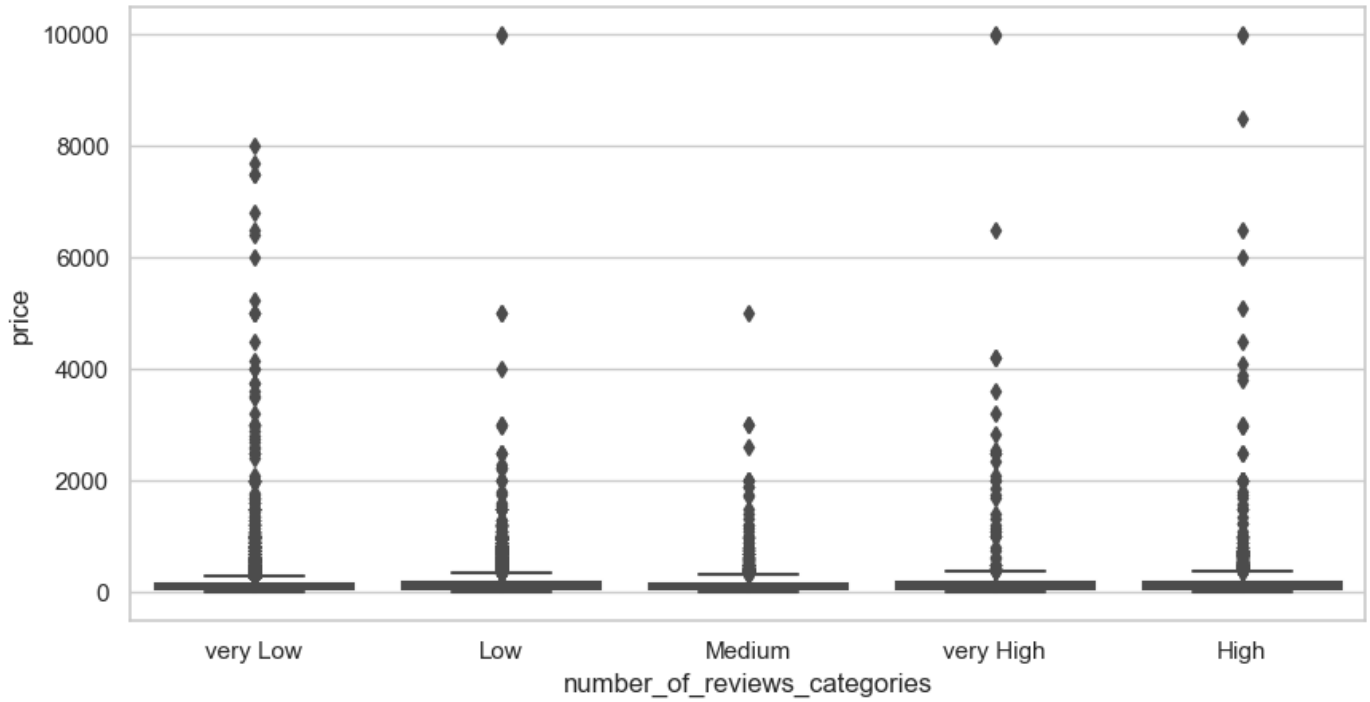
Out[94]:
```
number_of_reviews_categories
Low          4002323
very Low     1806531
High          971346
Medium        508647
very High     178431
Name: price, dtype: int64
```

In [95]:
```
plt.figure(figsize=(10,5))
sns.barplot(x = x1.index,y = x1.values)
plt.show()
```



In [96]:
```
plt.figure(figsize=(10,5))
sns.boxplot(x = abnb.number_of_reviews_categories , y = abnb.price)
```

Out[96]:
```
<Axes: xlabel='number_of_reviews_categories', ylabel='price'>
```



In [97]:
```
abnb.groupby('number_of_reviews_categories').price.mean().sort_values()
```

Out[97]:
```
number_of_reviews_categories
very Low      142.022877
Medium        145.203254
Low           153.746274
High          164.830477
very High     238.863454
Name: price, dtype: float64
```

In [98]:
```
abnb.groupby('number_of_reviews_categories').price.median().sort_values()
```

Out[98]:
```
number_of_reviews_categories
very Low       90.0
Medium        105.0
High          120.0
Low           120.0
very High     120.0
Name: price, dtype: float64
```

In [99]:
```
x2 = pd.DataFrame(x1)
x2 = x2.reset_index()
x2
```

Out[99]:

	number_of_reviews_categories	price
0	Low	4002323
1	very Low	1806531
2	High	971346
3	Medium	508647
4	very High	178431

In [100…
```
((x2.groupby('number_of_reviews_categories').price.sum()/x2.price.sum())*100).sort_value
```

Out[100]:
```
number_of_reviews_categories
very High      2.389505
Medium         6.811679
High          13.008033
very Low      24.192631
Low           53.598152
Name: price, dtype: float64
```

What is the pricing ranges preferred by customers?

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

7.4 Room Type and Number of review category

In [101…
```
abnb.head()
```

Out[101]:

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

In [102…
```
abnb.room_type.value_counts()
```

Out[102]:
```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

In [103…
```
pd.crosstab(abnb['room_type'], abnb['number_of_reviews_categories'])
```

Out[103]:

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

In [104…
```
abnb.groupby('room_type').number_of_reviews.sum()
```

Out[104]:
```
room_type
Entire home/apt    580403
Private room       538346
Shared room         19256
Name: number_of_reviews, dtype: int64
```

In [105…
```
abnb.groupby('room_type').number_of_reviews.sum()/abnb.room_type.value_counts()
```

Out[105]:
```
room_type
Entire home/apt    22.842418
Private room       24.112962
Shared room        16.600000
dtype: float64
```

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

1) Entire home/apt have more reviews than Shared rooms 2) Shared room' are less likey to give reviews. only 16 %

7.5 Room Type and Price Category

In [106…
```
pd.crosstab(abnb['room_type'], abnb['price_categories'])
```

Out[106]:

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

In [107…
```
abnb.room_type.value_counts()
```

Out[107]:
```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

In [108…
```
abnb.groupby('room_type').reviews_per_month.mean()
```

Out[108]:
```
room_type
Entire home/apt    1.306578
Private room       1.445209
Shared room        1.471726
Name: reviews_per_month, dtype: float64
```

In [109…
```
abnb.groupby('room_type').reviews_per_month.median()
```

Out[109]:
```
room_type
Entire home/apt    0.66
Private room       0.77
Shared room        0.98
Name: reviews_per_month, dtype: float64
```

In [110…
```
abnb.groupby('room_type').reviews_per_month.sum()
```

Out[110]:
```
room_type
Entire home/apt    26565.34
Private room       25529.62
Shared room         1245.08
Name: reviews_per_month, dtype: float64
```

In [114…
```
plt.figure(figsize=(70,50))
sns.boxplot(data = abnb, y = 'room_type' ,x = 'reviews_per_month')
plt.xticks(np.arange(0,100,.5))
plt.show()
```

Observation

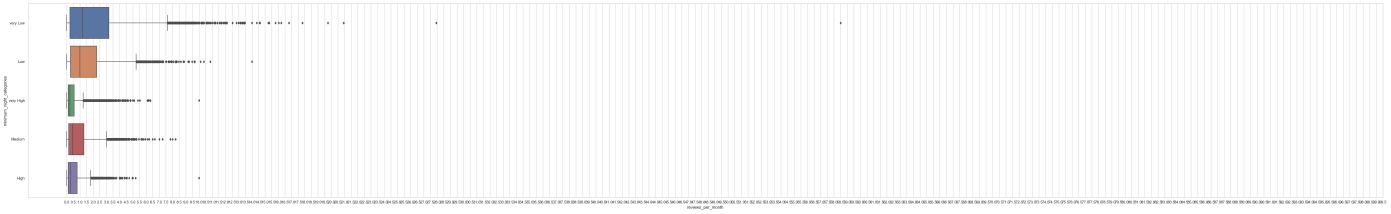
For each room type there are approx 1.4 reviews per month on average.

7.7 Minimum night category and Reviews per month

In [112…
```
abnb.groupby('minimum_night_categories').reviews_per_month.sum().sort_values()
```

Out[112]:
```
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
```

In [113…
```
plt.figure(figsize=(70,10))
sns.boxplot(data = abnb, y = 'minimum_night_categories' ,x = 'reviews_per_month')
plt.xticks(np.arange(0,100,.5))
plt.show()
```



Observation

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.

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

In [135…
```
abnb.availability_365_categories.value_counts()
```

Out[135]:
```
very Low      17941
Low           11829
very High      8108
Medium         5792
High           5225
Name: availability_365_categories, dtype: int64
```

In [136…
```
pd.DataFrame(abnb.groupby(['availability_365_categories','price_categories']).reviews_pe
```

Out[136]:

		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

Observaation

1. If the combination of availability and price is very high, reviews_per_month will be low on average.
2. Very high availability and very low price are likely to get more reviews.

In [137…
```
abnb.to_csv('AB_NYC_2019_updated.csv')
```

In [ ]:

In [ ]: