AIRBNB, NYC STORYTELLING CASE STUDY

DATA METHODOLOGY

RUSHABH PATEL

C59 BATCH, 2024

Importing libraries and reading data:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data1 = pd.read_csv('AB_NYC_2019.csv')
data1.head()
```

Out[2]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitu
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.647
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.753
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.809
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.685
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.798

In [3]:
```python
def availability_365_cat_func(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 [4]:
```python
data1['availability_365_categories'] = data1.availability_365.map(availabili
data1['availability_365_categories']
```

Out[4]:
```
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 [5]:
```python
data1['availability_365_categories'].value_counts()
```

Out[5]:
```
Very Low     17941
Low          11829
Very High     8108
Medium        5792
High          5225
Name: availability_365_categories, dtype: int64
```

EDA:

In [14]:
```
## let's check null counts and data types:

data1.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
```

In [15]:
```
## let's change dtype of last_review to Datetime:

data1.last_review = pd.to_datetime(data1.last_review)
data1.last_review
```

Out[15]:
```
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]
```

Categorical Data Types:

In [17]:
```
data1.columns
```

Out[17]:
```
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')
```

In [18]:
```
## let's differentiate categorical columns:

categorical_columns = data1.columns[[0, 1, 3,4,5,8,16,17,18,19]]
categorical_columns
```

Out[18]:
```
Index(['id', 'name', 'host_name', 'neighbourhood_group', 'neighbourhood',
       'room_type', 'availability_365_categories', 'minimum_night_categori
es',
       'number_of_reviews_categories', 'price_categories'],
      dtype='object')
```

Numerical Data Types:

In [20]:
```
numerical_columns = data1.columns[[9,10,11,13,14,15]]
numerical_columns
```

Out[20]:
```
Index(['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_mont
h',
       'calculated_host_listings_count', 'availability_365'],
      dtype='object')
```

In [21]:
```
data1[numerical_columns].head()
```

Out[21]:

	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_co
0	149	1	9	0.21	
1	225	1	45	0.38	
2	150	3	0	NaN	
3	89	1	270	4.64	
4	80	10	9	0.10	

checking missing values:

In [24]:
```
data1.isnull().sum()
```

Out[24]:
```
id                                 0
name                              16
host_id                            0
host_name                         21
neighbourhood_group                0
neighbourhood                      0
latitude                           0
longitude                          0
room_type                          0
price                              0
minimum_nights                     0
number_of_reviews                  0
last_review                    10052
reviews_per_month              10052
calculated_host_listings_count     0
availability_365                   0
availability_365_categories        0
minimum_night_categories           0
number_of_reviews_categories       0
price_categories                   0
dtype: int64
```

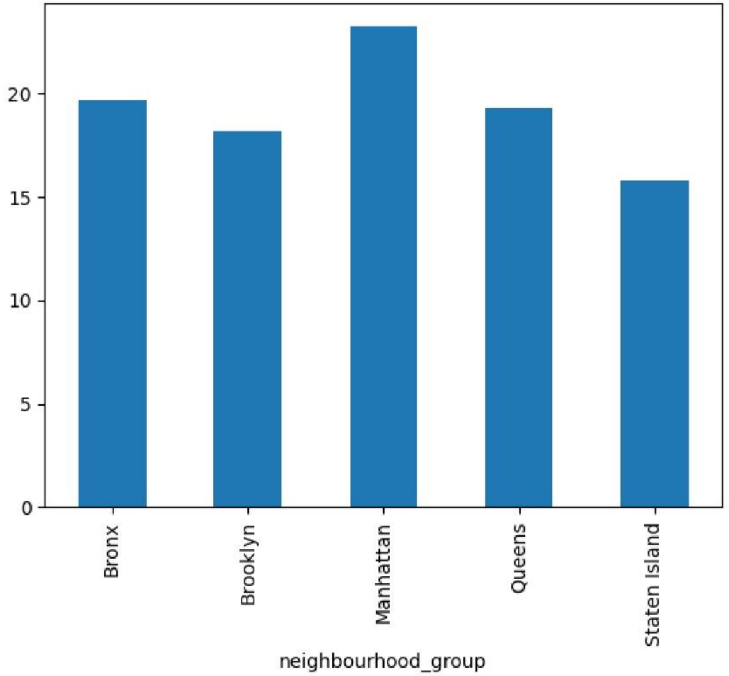
In [25]:
```
round(data1.isnull().sum() / len(data1) * 100,2)
```

Out[25]:
```
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
```

- last_review and reviews_per_month has highest 20.56% missing values.
- Here in this case study the main thing is, we're not going to develope any model so no need to drop or imputation missing values.

In [30]:
```
((data2.groupby('neighbourhood_group').neighbourhood_group.count() /
  data1.groupby('neighbourhood_group').neighbourhood_group.count())*100).plc
```

Out[30]: <Axes: xlabel='neighbourhood_group'>

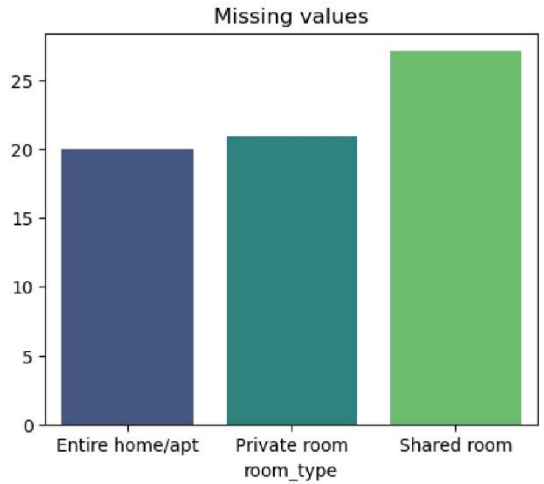


In [31]:
```
round((data2.groupby('neighbourhood_group').neighbourhood_group.count() /
  data1.groupby('neighbourhood_group').neighbourhood_group.count())*100,1).r
```

Out[31]: 19.24

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

In [34]:
```
plt.figure(figsize=[5,4])
plt.title('Missing values')
sns.barplot(x = data3.index, y = data3.values, palette='viridis')
plt.show()
```

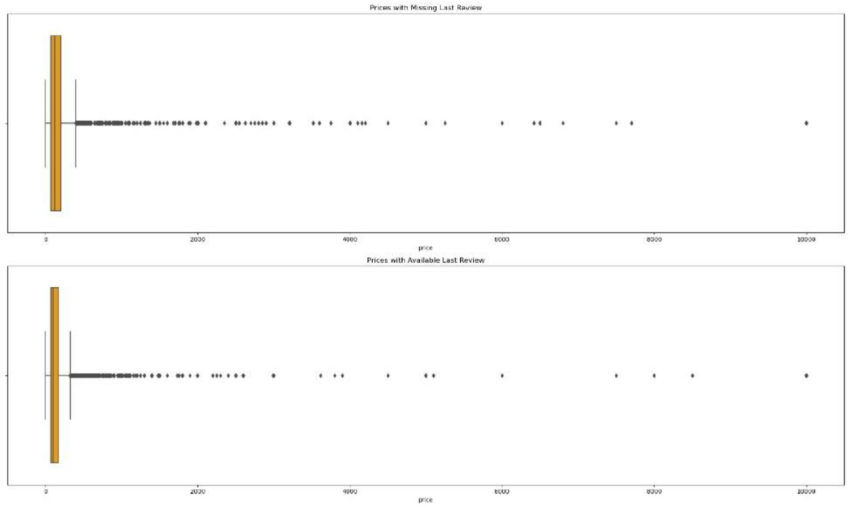


- 'Shared room' has the highest missing value percentage 27% for 'last_review' feature while to other room types has only about 20 %.

In [35]:
```
print('Mean and Median of prices with last_review feature missing')
print('Mean   = ', round(data1[data1['last_review'].isnull()].price.mean(),1
print('Median = ', data1[data1['last_review'].isnull()].price.median())

print('\nMean and Median of prices with last_review feature not missing')
print('Mean   = ', round(data1[data1['last_review'].notnull()].price.mean(),
print('Median = ', data1[data1['last_review'].notnull()].price.median())
```

```
Mean and Median of prices with last_review feature missing
Mean   =  192.9
Median =  120.0

Mean and Median of prices with last_review feature not missing
Mean   =  142.3
Median =  101.0
```

In [36]:
```
plt.figure(figsize=(20, 12))

plt.subplot(2, 1, 1)
sns.boxplot(data=data1[data1['last_review'].isnull()], x='price', width=0.8,
plt.title('Prices with Missing Last Review')

plt.subplot(2, 1, 2)
sns.boxplot(data=data1[data1['last_review'].notnull()], x='price', width=0.8
plt.title('Prices with Available Last Review')

plt.tight_layout()
plt.show()
```



Analysis scenerio:

- 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)

85% of the listing are Manhattan and Brooklyn neighbourhood_group

host_name

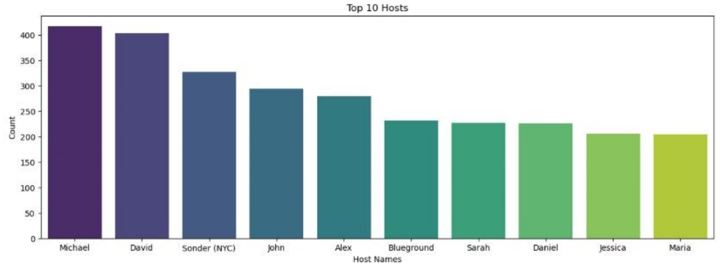
In [43]:
```
data1.host_name.value_counts()
```

Out[43]:
```
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
```

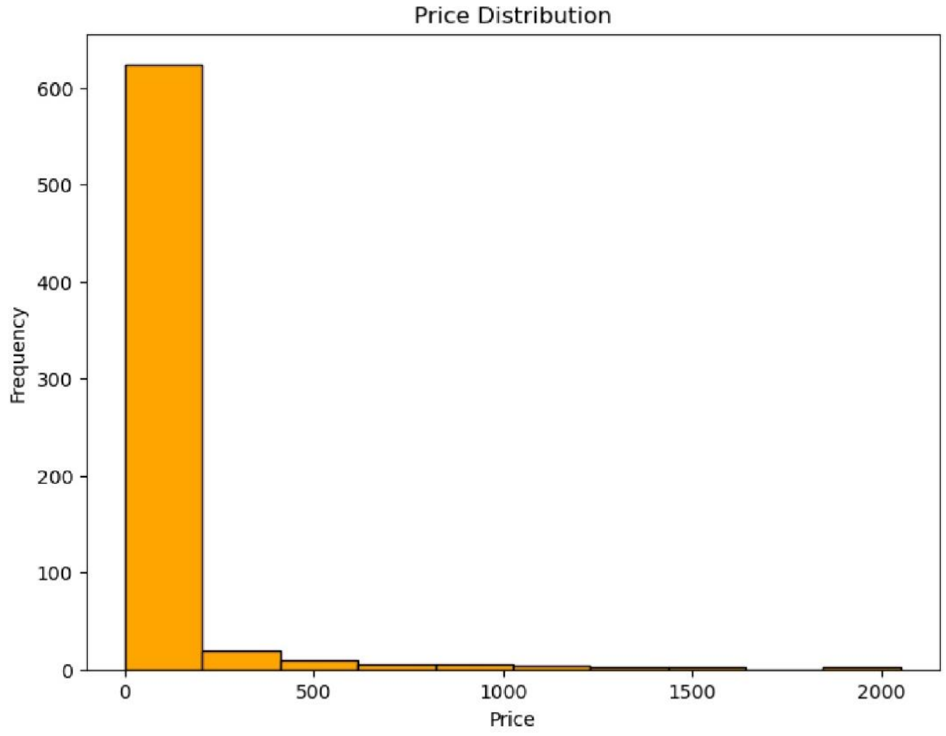
In [44]:
```
data1.host_name.value_counts().index[:10]
```

Out[44]:
```
Index(['Michael', 'David', 'Sonder (NYC)', 'John', 'Alex', 'Blueground',
       'Sarah', 'Daniel', 'Jessica', 'Maria'],
      dtype='object')
```

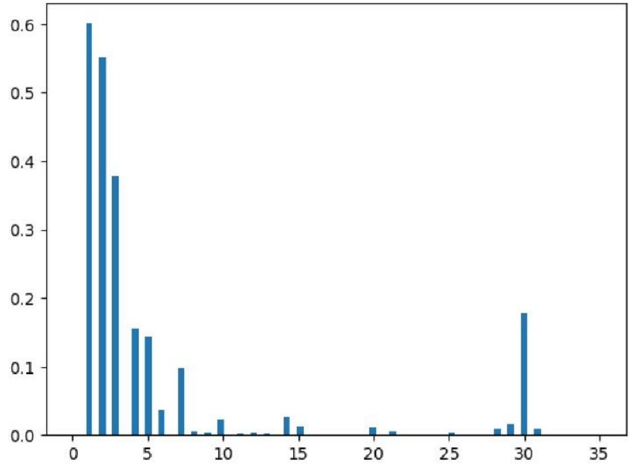
In [52]:
```
plt.figure(figsize=(15, 5))

sns.barplot(x=data1.host_name.value_counts().index[:10],
            y=data1.host_name.value_counts().values[:10],
            palette='viridis')

plt.title("Top 10 Hosts")
plt.xlabel("Host Names")
plt.ylabel("Count")
plt.show()
```



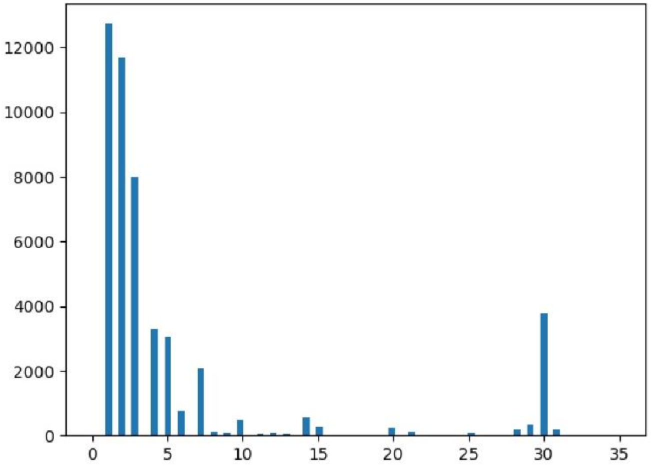
In [68]:
```
plt.figure(figsize=(8, 6))
data1.price.value_counts().plot.hist(color='orange', bins=10, edgecolor='bla
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



```
In [77]: plt.hist(data = data1, x = 'minimum_nights',bins=80, range=(0,35), density=T
         plt.show()
```



```
In [78]: plt.hist(data = data1, x = 'minimum_nights',bins=80, range=(0,35))
         plt.show()
```

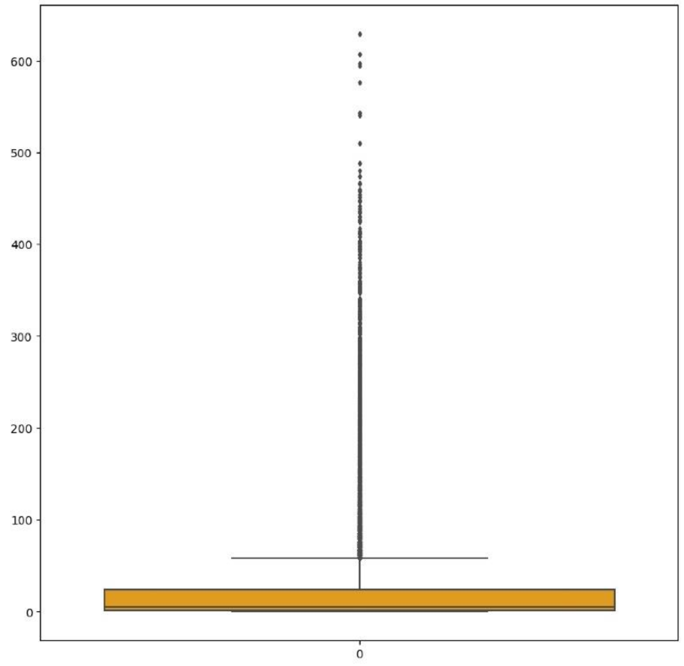


number_of_reviews

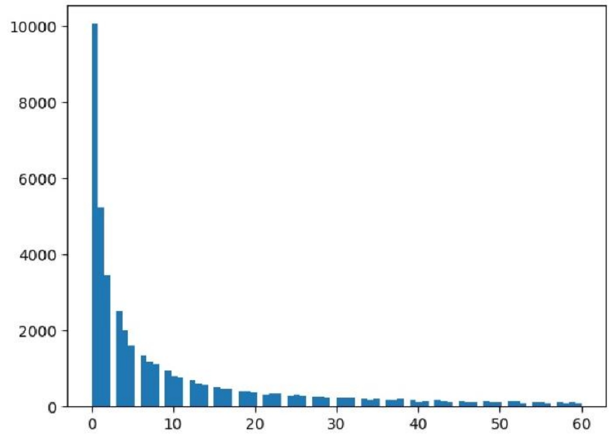
In [79]:
```
data1.number_of_reviews.describe()
```

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
```

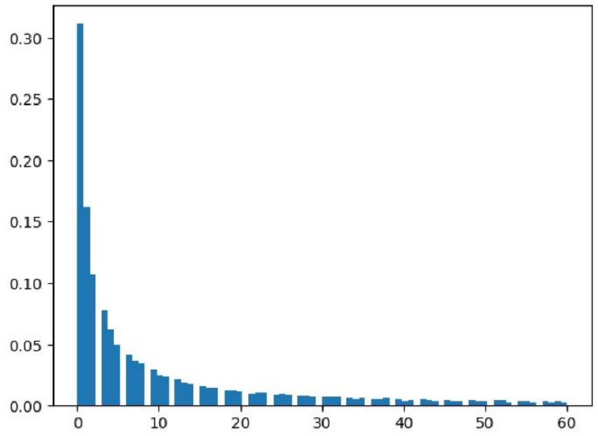
In [93]:
```
plt.figure(figsize=(10,10))
sns.boxplot(data = data1.number_of_reviews,fliersize=3, color='orange')
plt.show()
```



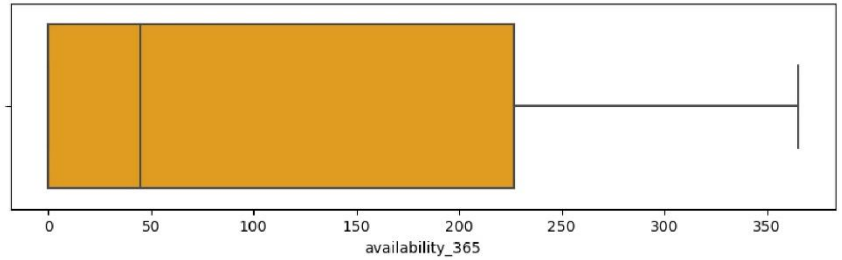
In [81]:
```
plt.hist(data = data1, x = 'number_of_reviews',bins=80,range=(0,60))
plt.show()
```



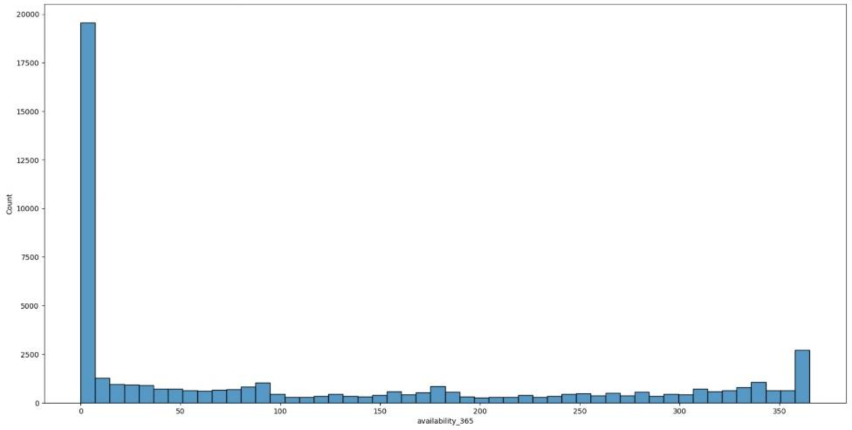
In [82]:
```
plt.hist(data = data1, x = 'number_of_reviews', bins=80, range=(0,60),densit
plt.show()
```



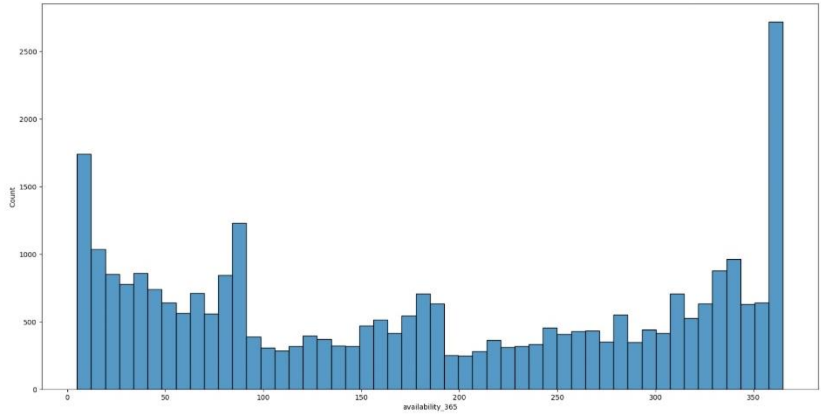
In [105]:
```
plt.figure(figsize = (10,2.5))
sns.boxplot(data = data1 , x = 'availability_365', color='orange')
plt.show()
```



In [106]:
```
plt.figure(figsize = (20,10))
sns.histplot(data = data1, x = 'availability_365',bins=50,binrange=(0,365))
plt.show()
```



In [107]:
```
plt.figure(figsize = (20,10))
sns.histplot(data = data1, x = 'availability_365',bins=50,binrange=(5,365))
plt.show()
```



number_of_reviews_categories

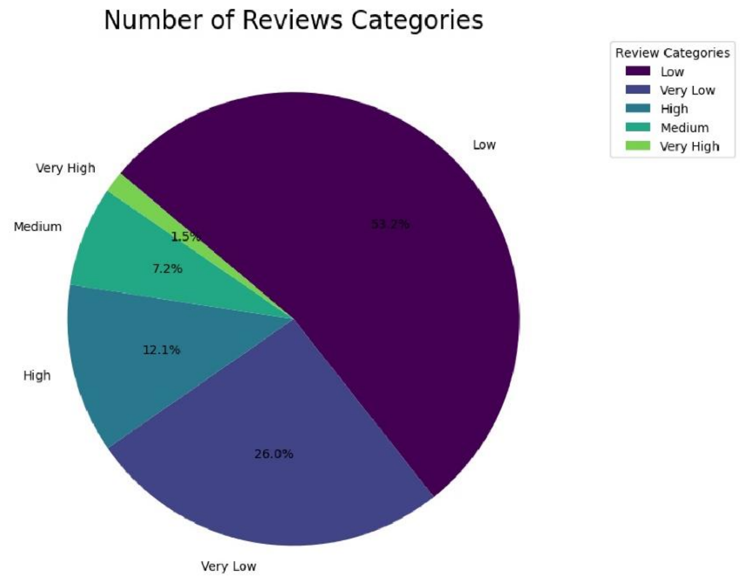
In [108]:
```
data1.number_of_reviews_categories.value_counts()
```

Out[108]:
```
Low          26032
Very Low     12720
High          5893
Medium        3503
Very High      747
Name: number_of_reviews_categories, dtype: int64
```

In [109]:
```
data1.number_of_reviews_categories.value_counts(normalize=True)*100
```

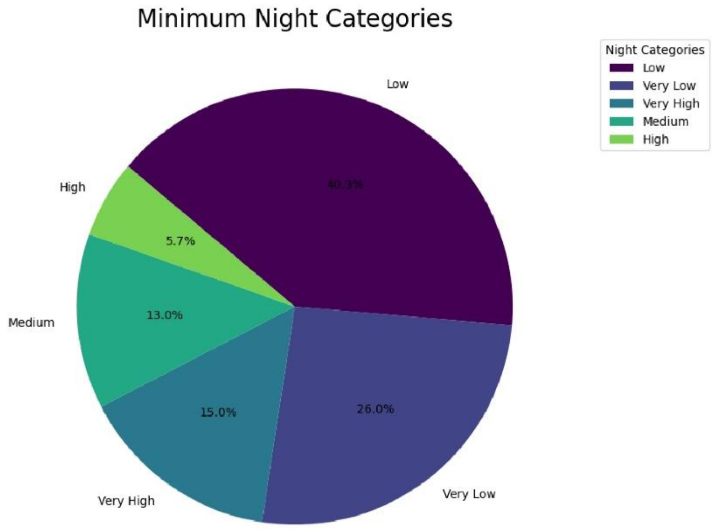
Out[109]:
```
Low          53.240618
Very Low     26.014930
High         12.052357
Medium        7.164332
Very High     1.527764
Name: number_of_reviews_categories, dtype: float64
```

In [119]:
```
labels = counts.index
sizes = counts.values

cmap = plt.get_cmap('viridis')
colors = [cmap(i / len(labels)) for i in range(len(labels))]

plt.figure(figsize=(8, 7))
plt.title('Number of Reviews Categories', fontdict={'fontsize': 20})
plt.pie(x=sizes,
        labels=labels,
        autopct='%1.1f%%',
        colors=colors,
        startangle=140,
        counterclock=False)

plt.legend(title='Review Categories', bbox_to_anchor=(1.05, 1), loc='upper l
plt.tight_layout()
plt.show()
```



In [122]:

```
counts = data1.minimum_night_categories.value_counts()
labels = counts.index
sizes = counts.values

# Generate colors from the 'viridis' colormap
cmap = plt.get_cmap('viridis')
colors = [cmap(i / len(labels)) for i in range(len(labels))]

# Create the pie chart
plt.figure(figsize=(12, 7))
plt.title('Minimum Night Categories', fontdict={'fontsize': 20})
plt.pie(x=sizes,
        labels=labels,
        autopct='%1.1f%%',  # Add percentage display
        colors=colors,  # Add colors
        startangle=140,  # Rotate the start angle for better visualization
        counterclock=False)  # Set the direction to clockwise

# Add the legend and move it outside the chart
plt.legend(title='Night Categories', bbox_to_anchor=(1.05, 1), loc='upper le
plt.tight_layout()
plt.show()
```



Bivariate and Multivariate Analysis

Let's find Correlations:

In [128]:
```
data1[numerical_columns].head()
```

Out[128]:

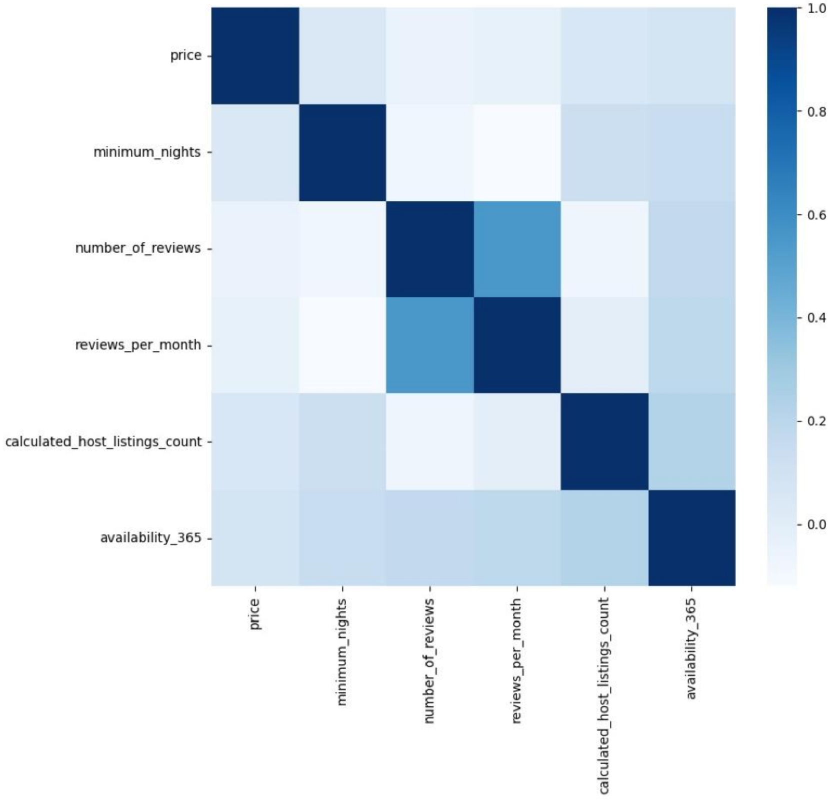
	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_co
0	149	1	9	0.21	
1	225	1	45	0.38	
2	150	3	0	NaN	
3	89	1	270	4.64	
4	80	10	9	0.10	

In [129]:
```
data1[numerical_columns].corr()
```

Out[129]:

	price	minimum_nights	number_of_reviews	reviews_per_m
price	1.000000	0.042799	-0.047954	-0.03
minimum_nights	0.042799	1.000000	-0.080116	-0.12
number_of_reviews	-0.047954	-0.080116	1.000000	0.54
reviews_per_month	-0.030608	-0.121702	0.549868	1.00
calculated_host_listings_count	0.057472	0.127960	-0.072376	-0.00
availability_365	0.081829	0.144303	0.172028	0.18

In [139]:
```
plt.figure(figsize=(9,8))
sns.heatmap(data = data1[numerical_columns].corr(), cmap='Blues')
plt.show()
```



In [141]:
```
corr_matrix
```

Out[141]:

	price	minimum_nights	number_of_reviews	reviews_per_mo
price	1.000000	0.042799	0.047954	0.030
minimum_nights	0.042799	1.000000	0.080116	0.121
number_of_reviews	0.047954	0.080116	1.000000	0.549
reviews_per_month	0.030608	0.121702	0.549868	1.000
calculated_host_listings_count	0.057472	0.127960	0.072376	0.009
availability_365	0.081829	0.144303	0.172028	0.185

In [142]:
```
sol
```

Out[142]:
```
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 [143]:
```
# Top correlations

sol[1:8]
```

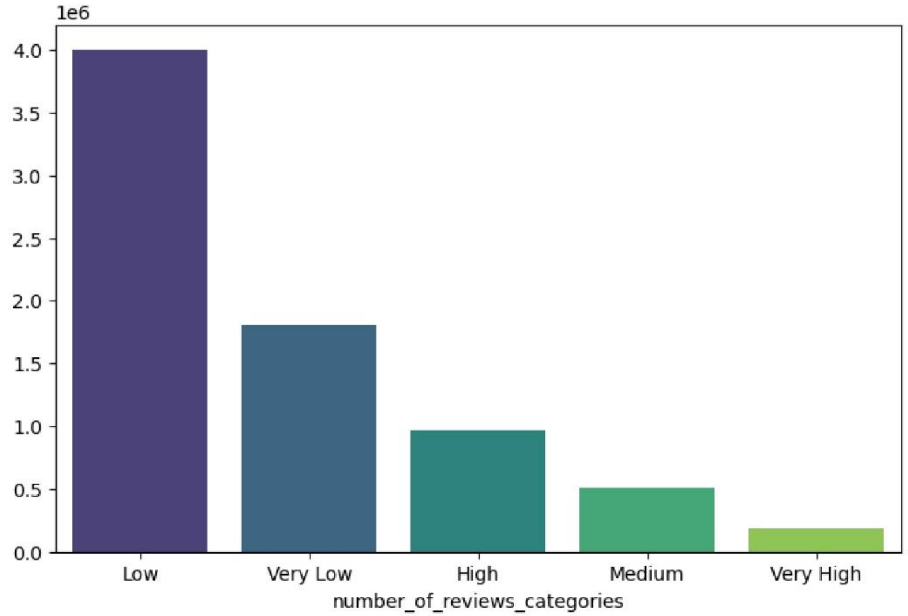
Out[143]:
```
calculated_host_listings_count  availability_365                  0.225701
reviews_per_month               availability_365                  0.185791
number_of_reviews               availability_365                  0.172028
minimum_nights                  availability_365                  0.144303
                                calculated_host_listings_count    0.127960
                                reviews_per_month                 0.121702
price                           availability_365                  0.081829
dtype: float64
```

number_of_reviews_categories and prices

In [144]:
```
# prices for each of reviews_categories

x1 = data1.groupby('number_of_reviews_categories').price.sum().sort_values(a
x1
```

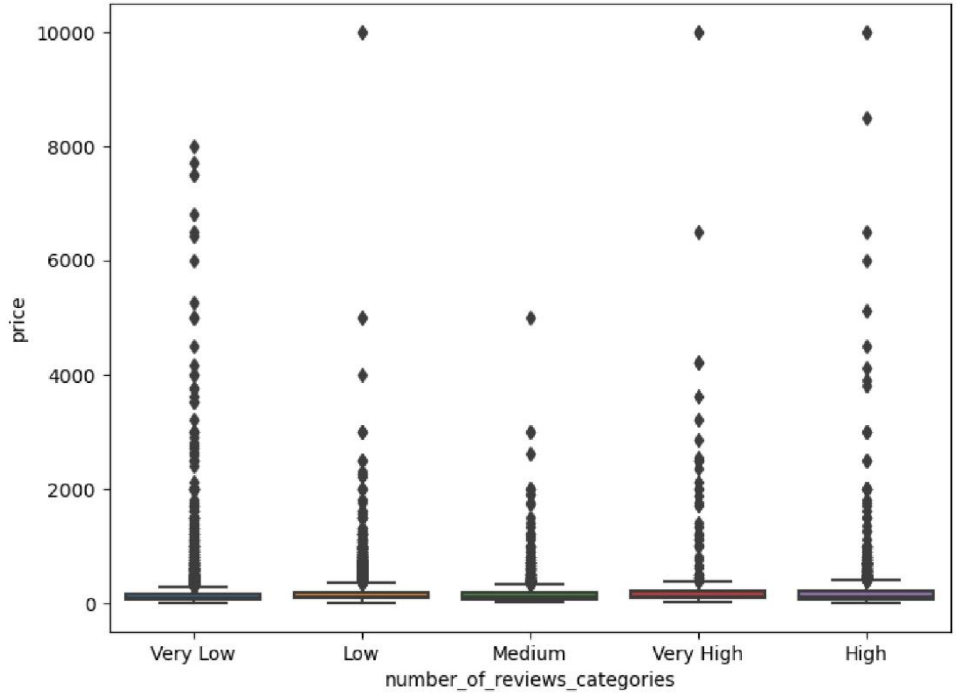
Out[144]:
```
number_of_reviews_categories
Low          4002323
Very Low     1806531
High          971346
Medium        508647
Very High     178431
Name: price, dtype: int64
```

In [146]:
```
plt.figure(figsize=(8,5))
sns.barplot(x = x1.index,y = x1.values, palette='viridis')
plt.show()
```



```
In [148]: plt.figure(figsize=(8,6))
          sns.boxplot(x = data1.number_of_reviews_categories , y = data1.price)
```

Out[148]: <Axes: xlabel='number_of_reviews_categories', ylabel='price'>



In [151]:
```
x2 = pd.DataFrame(x1)
x2 = x2.reset_index()
x2
```

Out[151]:

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

In [152]:
```
((x2.groupby('number_of_reviews_categories').price.sum()/x2.price.sum())*10(
```

Out[152]:
```
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
```

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

('room_type' and 'number_of_reviews_categories')

In [153]:
```
data1.head()
```

Out[153]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitu
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.647
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.753
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.809
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.685
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.798

'room_type' and 'reviews_per_month'

In [160]:
```
data1.room_type.value_counts()
```

Out[160]:
```
Entire home/apt    25409
Private room       22326
Shared room         1160
Name: room_type, dtype: int64
```

In [163]:
```
data1.groupby('room_type').reviews_per_month.mean()
```

Out[163]:
```
room_type
Entire home/apt    1.306578
Private room       1.445209
Shared room        1.471726
Name: reviews_per_month, dtype: float64
```

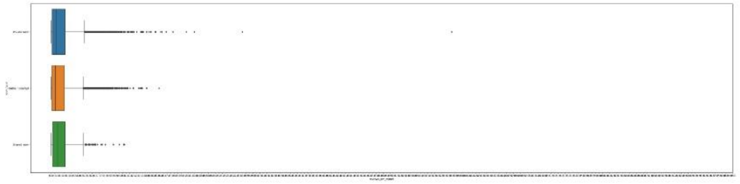
In [164]:
```
data1.groupby('room_type').reviews_per_month.median()
```

Out[164]:
```
room_type
Entire home/apt    0.66
Private room       0.77
Shared room        0.98
Name: reviews_per_month, dtype: float64
```

In [165]:
```
data1.groupby('room_type').reviews_per_month.sum()
```

Out[165]:
```
room_type
Entire home/apt    26565.34
Private room       25529.62
Shared room         1245.08
Name: reviews_per_month, dtype: float64
```

In [169]:
```
plt.figure(figsize=(50,12))
sns.boxplot(data = data1, y = 'room_type' ,x = 'reviews_per_month')
plt.xticks(np.arange(0,100,.5))
plt.show()
```



let's plot violin plot to understand better

In [176]:
```
plt.figure(figsize=(50, 12))
sns.violinplot(data=data1, y='room_type', x='reviews_per_month', palette='vi

plt.xticks(np.arange(0, 100, 0.5))
plt.show()
```

- 1.5 is the average reviews for each room type.

minimum_night_categories and reviews_per_month

In [177]:
```
data1.groupby('minimum_night_categories').reviews_per_month.sum().sort_value
```

Out[177]:
```
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 [180]:
```
plt.figure(figsize=(70,8))
sns.boxplot(data = data1, y = 'minimum_night_categories' ,x = 'reviews_per_r
plt.xticks(np.arange(0,100,.5))
plt.show()
```

- Customer's are more likely to leave reviews for low number of minimum nights
- minimum_nights should be on the lower side to make properties more customer-oriented

'availability_365_categories', 'price_categories' and 'reviews_per_month'

In [181]:
```
data1.availability_365_categories.value_counts()
```

Out[181]:
```
Very Low     17941
Low          11829
Very High     8108
Medium        5792
High          5225
Name: availability_365_categories, dtype: int64
```

In [182]:
```
pd.DataFrame(data1.groupby(['availability_365_categories','price_categories'
```

Out[182]:

		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

- Properties with high availability and low price tend to receive more reviews.
- Conversely, properties with high availability and high price typically have lower review rates.

THE END

THANK YOU!