STORY TELLING CASE STUDY

NEWYORK AIRBNBS

Problem background

- For the past few months, Airbnb has seen a major decline in revenue.
- Now that the restrictions have started lifting and people have started to travel more, Airbnb wants to make sure that it is fully prepared for this change.
- People have now started travelling again and Airbnb is aiming to bring up the business again and e ready to provide services to customers.

Objectives

- **To understand some important insights based on various attributes in the dataset so as to increase the revenue**
- **To process, analyse and share findings by data visualisation and statistical techniques.**
- **Enhance our understanding of property and host acquisitions, operations, and customer preferences.**
- **Provide early recommendations to our marketing and operations teams**

AGENDA

- DATA IMPORTATION
- ANALYSIS METHODS
- VISUALISATIONS
- RECOMMENDATIONS

METHODS INVOLVED IN DATA PURIFICATION

- IMPORTING DATA AND NECESSARY LIBRARIES
- TREATING AND COMPUTING OF MISSING VALUES
- UNDERSTANDING DATA TYPES
- EVALUATING AND TREATING OUTLIERS
- CREATING MORE FEATURES TO UNDERSTAND DATA BETTER

Importing Data and necessary libraries

Importing Data and necessary libraries

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

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
# importing data
abnyc = pd.read_csv("E:\My certificates\My projects\AB_NYC_2019.csv")
abnyc.head(5)
```

Out[3]:

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

In [4]:
```
abnyc.shape
```

Out[4]:
```
(48895, 16)
```

TREATING AND COMPUTING OF MISSING VALUES

In [4]:
```
abnyc.shape
```
Out[4]:
```
(48895, 16)
```

Analysing and computing missing values

In [5]:
```
abnyc.isnull().sum()
```
Out[5]:
```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

###values are missing in last_review and reviews_per_month, meaning these hosted sites/places have not received any reviews from the customers. Hence, these places would be least preferred by the future customers and would also be facing bad business from our side.

In [7]:
```
abnyc.describe()
```
Out[7]:

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.

In [8]:
```
# Now reviews per month contains more missing values which should be replaced with 0 respectively
abnyc.fillna({'reviews_per_month':0},inplace=True)
```

In [9]:
```
abnyc.reviews_per_month.isnull().sum()
```
Out[9]:
```
0
```

In [6]:
```
# Percentage of missing values
round((abnyc.isnull().sum()/len(abnyc))*100,2)
```
Out[6]:
```
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
dtype: float64
```

we identified two columns having an equal percentage of missing values which were last_review and reviews_per_month of around 20.56%. And also, the other two columns had quite minimal missing values which were host_name of 0.4% and name of the place of 0.3%.

In [12]:
```
### Lets do drop this column as it doesnt signify anything or any conclusion
abnyc.drop('last_review', axis = 1, inplace = True)
```

In [13]:
```
abnyc.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 15 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  reviews_per_month               48895 non-null  float64
 13  calculated_host_listings_count  48895 non-null  int64
 14  availability_365                48895 non-null  int64
dtypes: float64(3), int64(7), object(5)
memory usage: 5.6+ MB
```

UNDERSTANDING DATA TYPES

Data types

In [14]:
```
# Extracting Numeric columns:

int_cols = abnyc.select_dtypes(include=["int64","float64"]).columns
```

In [15]:
```
list(enumerate(int_cols))
```

Out[15]:
```
[(0, 'id'),
 (1, 'host_id'),
 (2, 'latitude'),
 (3, 'longitude'),
 (4, 'price'),
 (5, 'minimum_nights'),
 (6, 'number_of_reviews'),
 (7, 'reviews_per_month'),
 (8, 'calculated_host_listings_count'),
 (9, 'availability_365')]
```

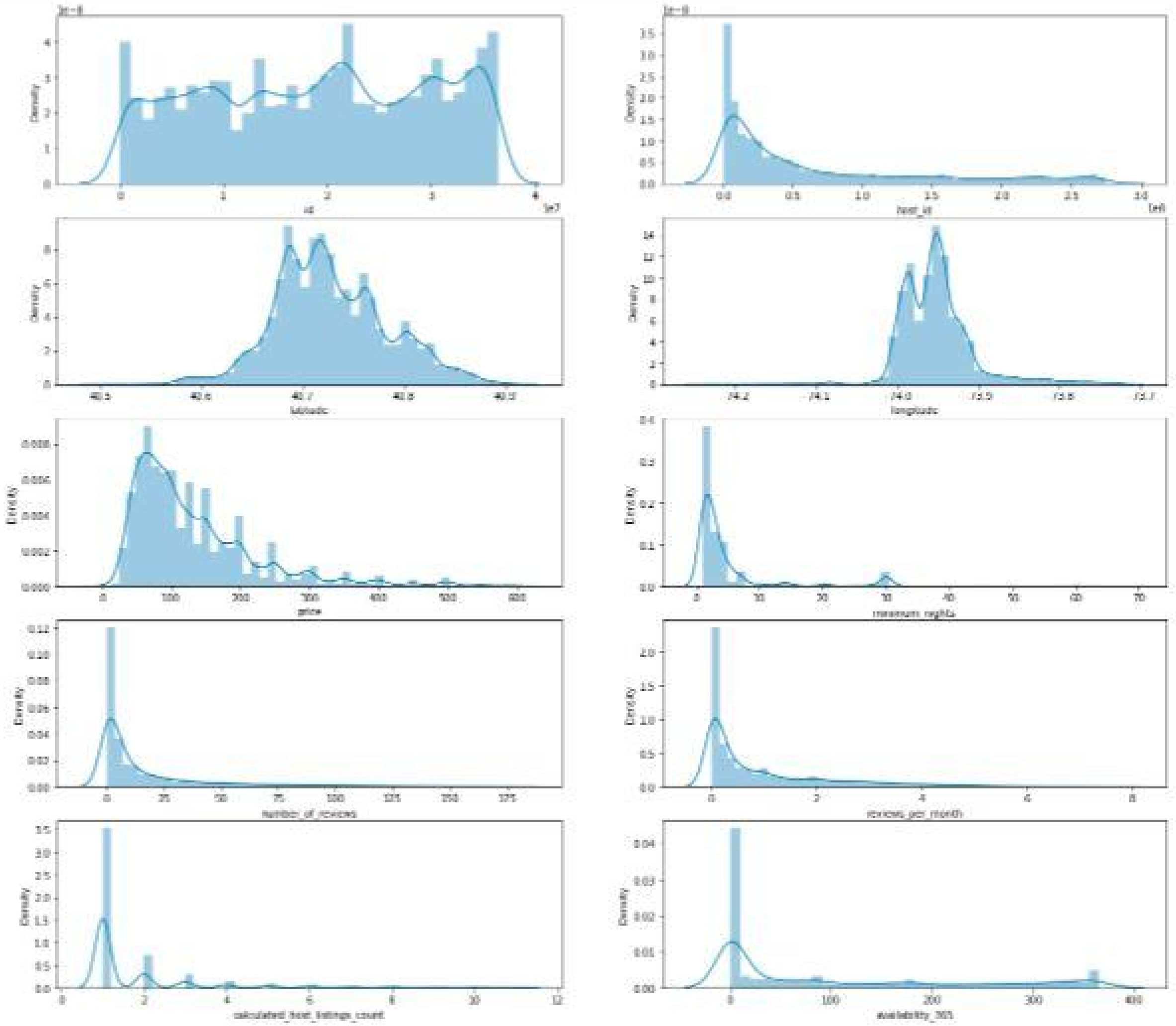
Analysing Categorical and Numeric values

In [23]:
```
cat_cols = abnyc.select_dtypes(exclude=['int64', 'float64']).columns
```

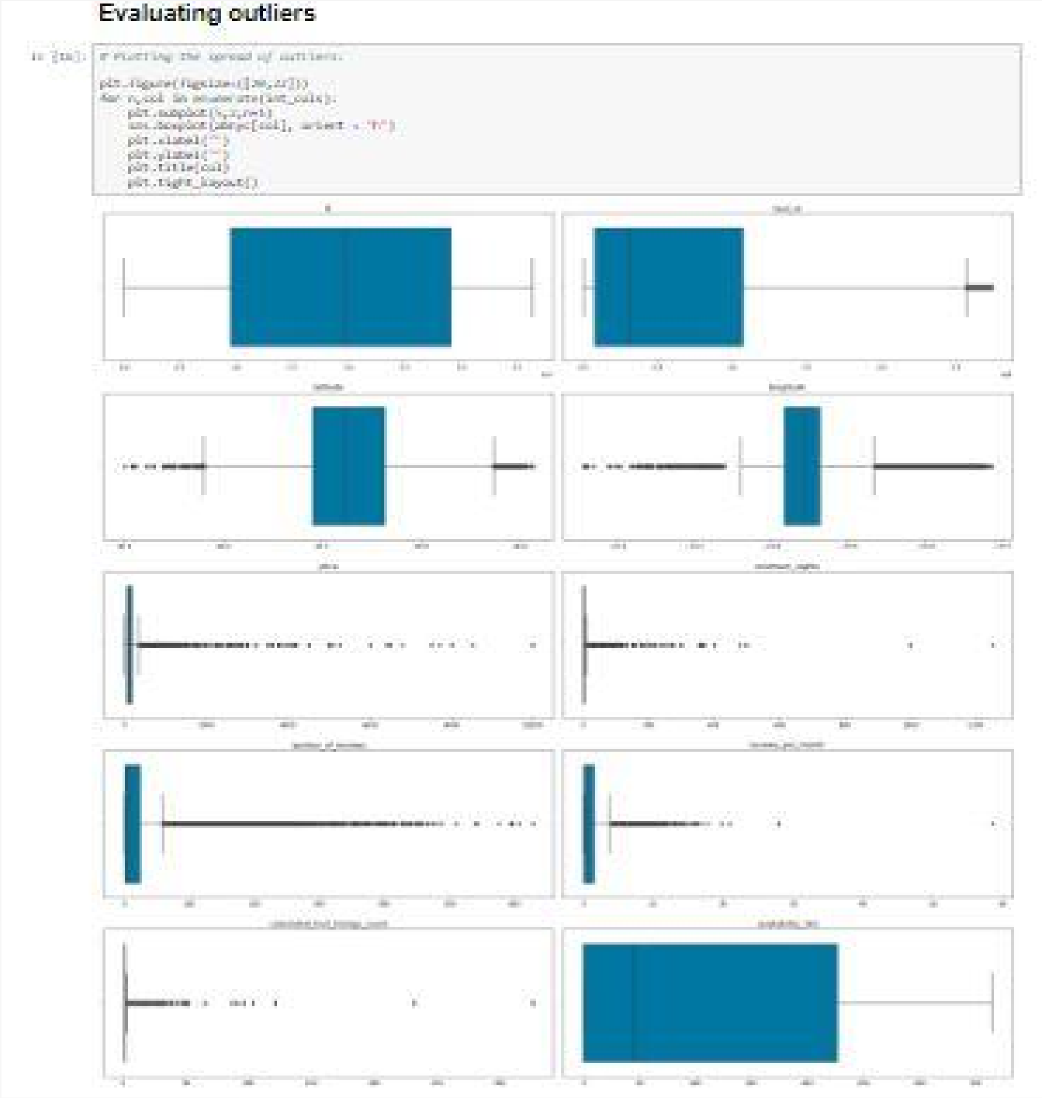
In [24]:
```
list(enumerate(cat_cols))
```

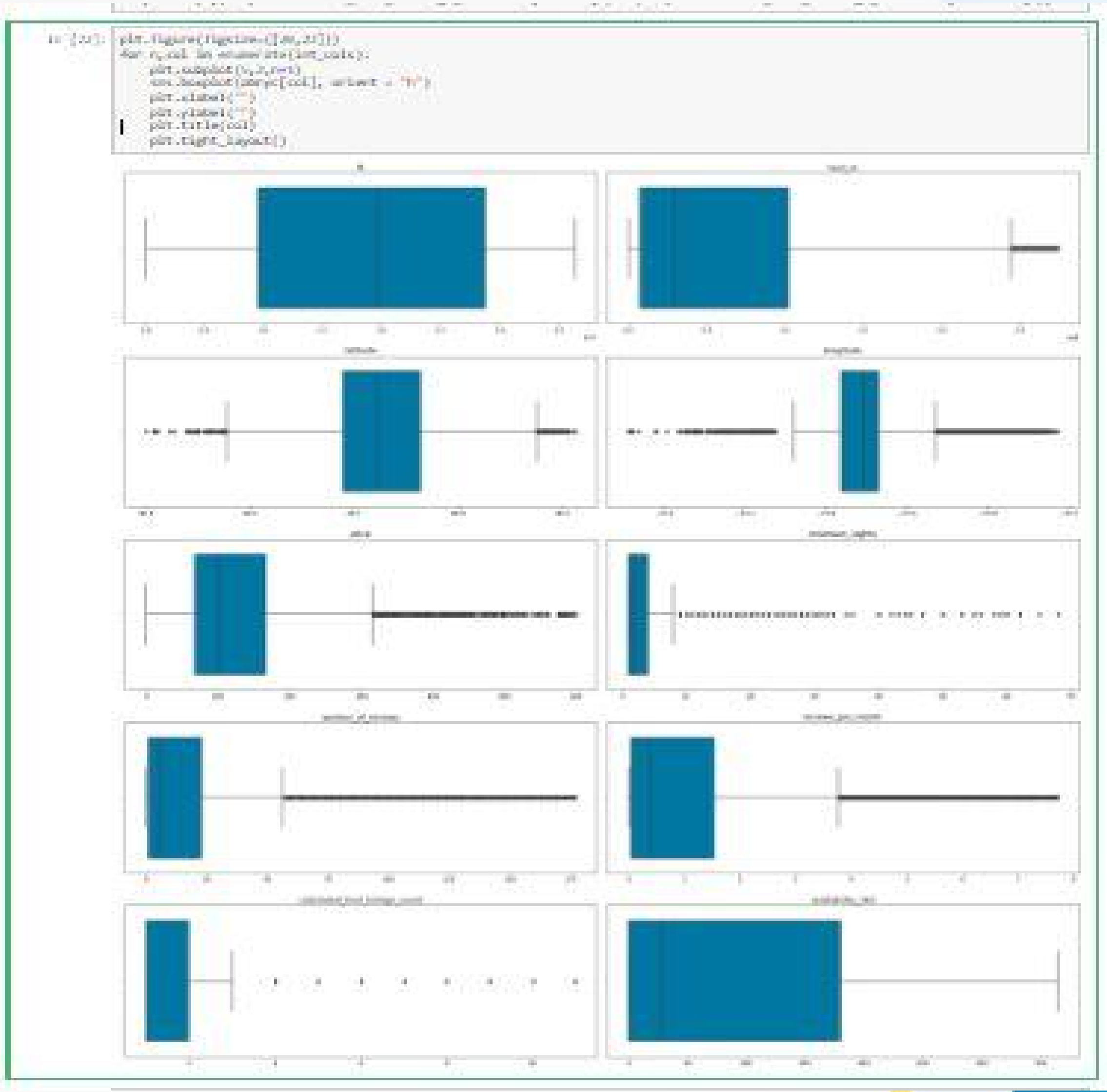
Out[24]:
```
[(0, 'name'),
 (1, 'host_name'),
 (2, 'neighbourhood_group'),
 (3, 'neighbourhood'),
 (4, 'room_type')]
```

In [25]:
```
int_cols = abnyc.select_dtypes(include=['int64', 'float64']).columns
plt.figure(figsize=[20,18])
for n,col in enumerate(int_cols):
    plt.subplot(5,2,n+1)
    sns.distplot(abnyc[col])
```



EVALUATING AND TREATING OUTLIERS





CREATING MORE FEATURES TO UNDERSTAND DATA BETTER

Creating more Features

```
In [28]: 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 [29]: abnyc['availability_365_categories'] = abnyc.availability_365.map(availability_365_categories_function)
         abnyc['availability_365_categories']

Out[29]: 0        very High
         1        very High
         2        very High
         4         very Low
         5           Medium
                   ...
         48890          Low
         48891          Low
         48892          Low
         48893          Low
         48894          Low
         Name: availability_365_categories, Length: 43912, dtype: object
```

```
In [30]: abnyc['availability_365_categories'].value_counts()

Out[30]: very Low     17523
         Low          11182
         very High     5921
         Medium        5170
         High          4116
```

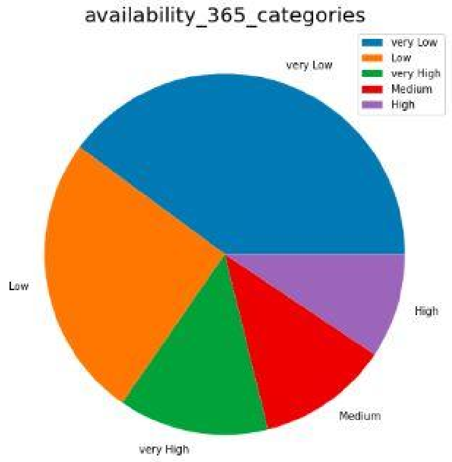
```
In [32]: def minimum_night_categories_function(row):
             """
             Categorizes the "minimum_nights" column into 5 categories
             """
             if row <= 1:
                 return 'very Low'
             elif row <= 3:
                 return 'Low'
             elif row <= 5 :
                 return 'Medium'
             elif (row <= 7):
                 return 'High'
             else:
                 return 'very High'
```

```
In [33]: abnyc['minimum_night_categories'] = abnyc.minimum_nights.map(minimum_night_categories_function)
         abnyc['minimum_night_categories']

Out[33]: 0         very Low
         1         very Low
         2              Low
         4        very High
         5              Low
                    ...
         48890          Low
         48891       Medium
         48892    very High
         48893     very Low
         48894         High
         Name: minimum_night_categories, Length: 43912, dtype: object
```

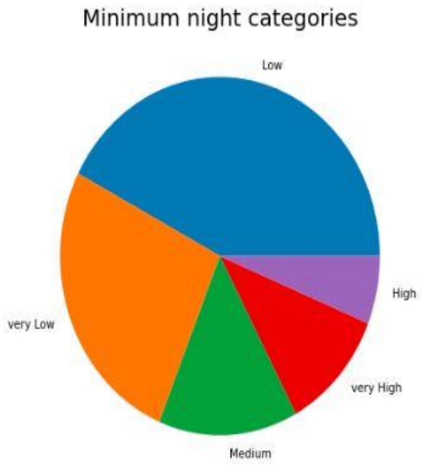
```
In [34]: abnyc.minimum_night_categories.value_counts()

Out[34]: Low          18609
         very Low     11603
         Medium        6176
         very High     4834
         High          2690
         Name: minimum_night_categories, dtype: int64
```

VISUALISATION OF DIFFERENT CATEGORIES

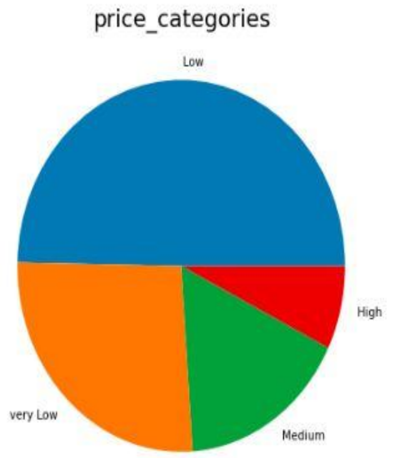
In [31]:
```
plt.figure(figsize=(8,8))
plt.title('availability_365_categories', fontdict={'fontsize': 20})
plt.pie(x = abnyc.availability_365_categories.value_counts(normalize= True) * 100,labels = abnyc.availability_365_categories.val
plt.legend()
plt.show()
```



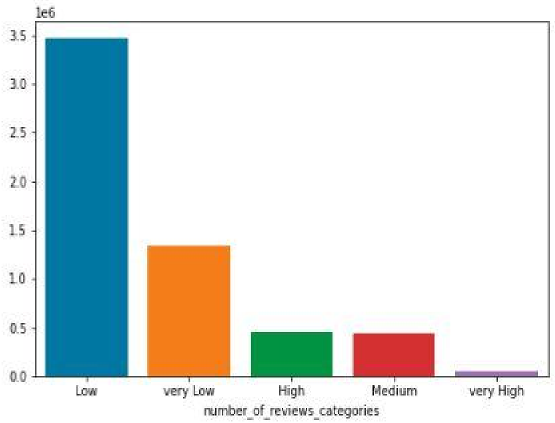
In [35]:
```
plt.figure(figsize=(12,7))
plt.title('Minimum night categories', fontdict={'fontsize': 20})
plt.pie(x = abnyc.minimum_night_categories.value_counts(),labels=abnyc.minimum_night_categories.value_counts().index)
plt.show()
```



In [40]:
```
plt.figure(figsize=(12,7))
plt.title('price_categories', fontdict={'fontsize': 20})
plt.pie(x = abnyc.price_categories.value_counts(),labels=abnyc.price_categories.value_counts().index,)
plt.show()
```



In [43]:
```
# prices for each of reviews_categories
x1 = abnyc.groupby('number_of_reviews_categories').price.sum().sort_values(ascending = False)
plt.figure(figsize=(8,5))
sns.barplot(x = x1.index,y = x1.values)
plt.show()
```



Availability_365_Categories vs Price_Categories vs Reviews per month

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

In [44]:
```
pd.DataFrame(abnyc.groupby(['availability_365_categories','price_categories']).reviews_per_month.mean())
```

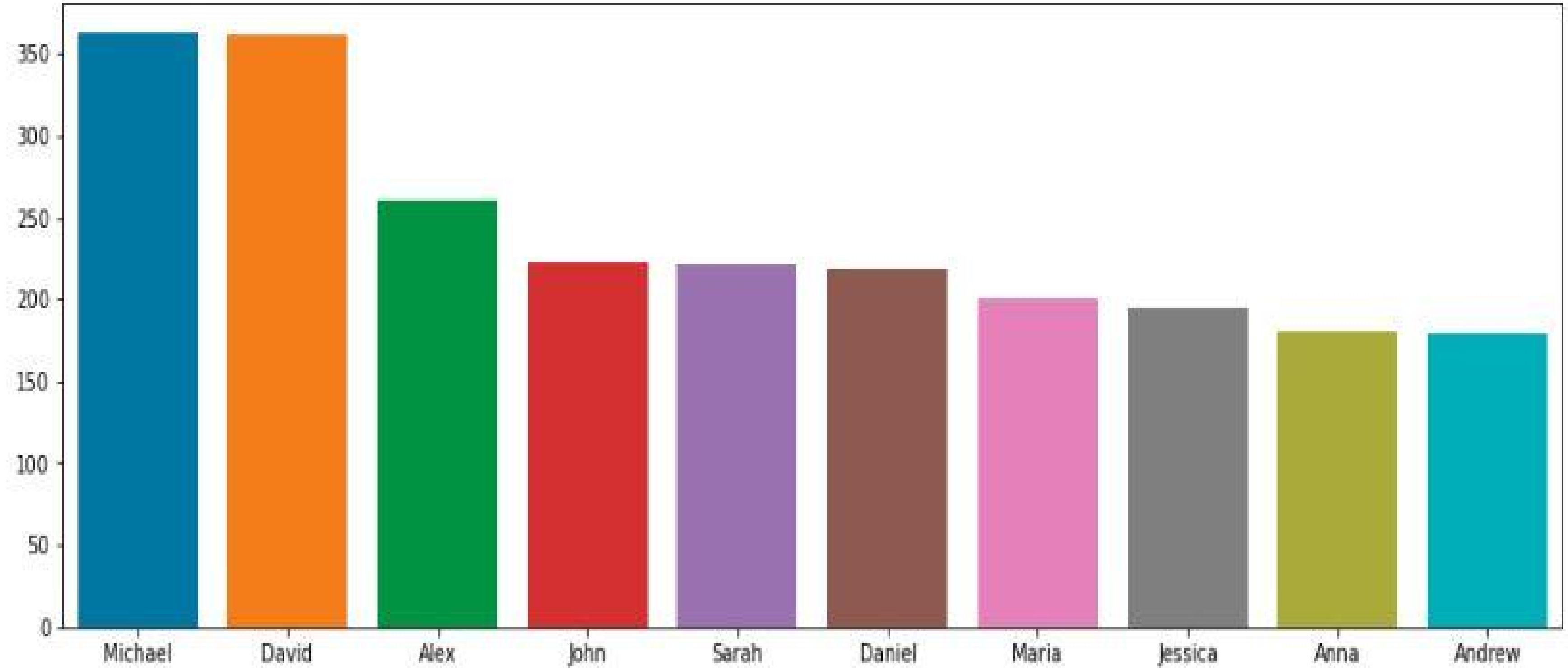
Out[44]:

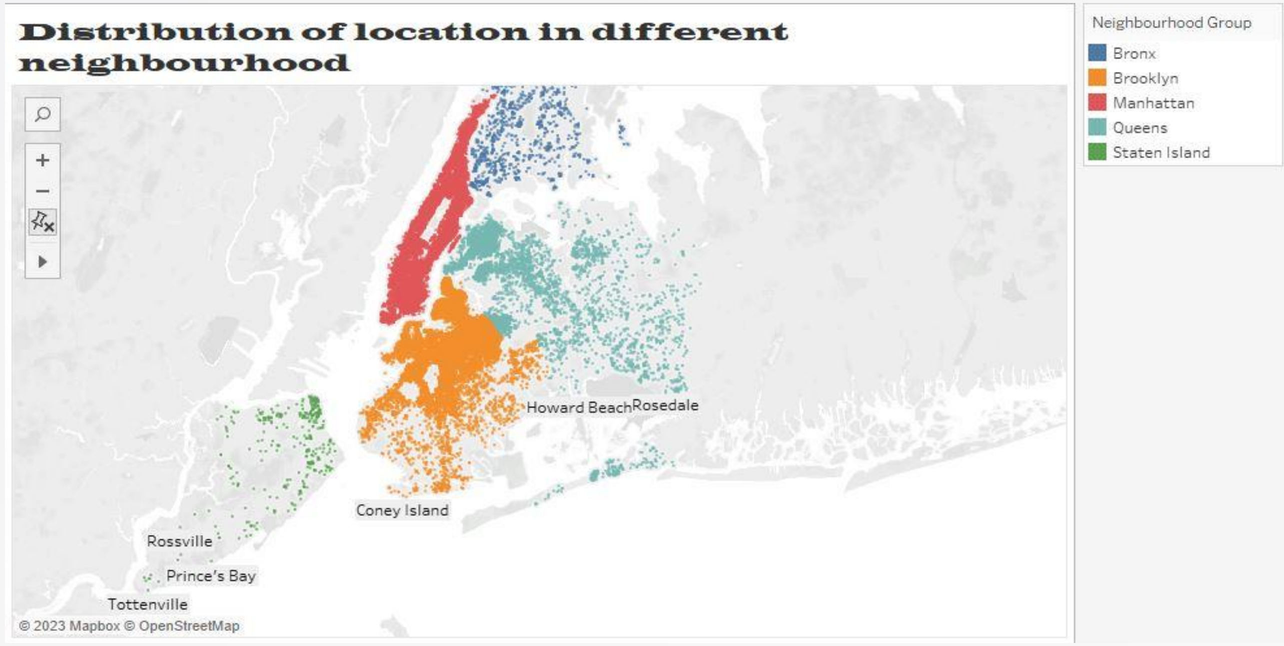
availability_365_categories	price_categories	reviews_per_month
High	High	0.618385
	Low	2.011989
	Medium	0.898256
	very Low	2.477938
Low	High	0.444719
	Low	1.545853
	Medium	0.696910
	very Low	2.233417
Medium	High	0.490754
	Low	1.748611
	Medium	0.968775
	very Low	2.169168
very High	High	0.359710
	Low	1.262194
	Medium	0.556535
	very Low	1.599074
very Low	High	0.201468
	Low	0.401511
	Medium	0.179748
	very Low	0.400113

Top ten performing hosts

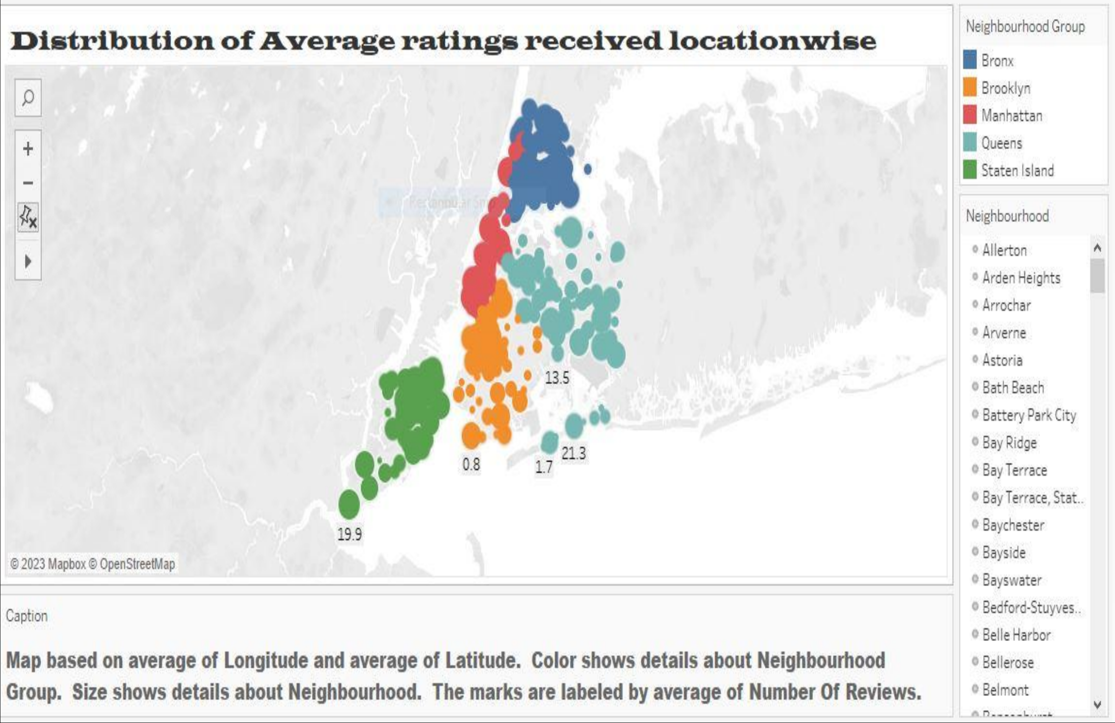
In [42]:

```
# Top 10 host's
plt.figure(figsize=(15,5))
sns.barplot(x = abnyc.host_name.value_counts().index[:10] , y = abnyc.host_name.value_counts().values[:10])
plt.show()
```



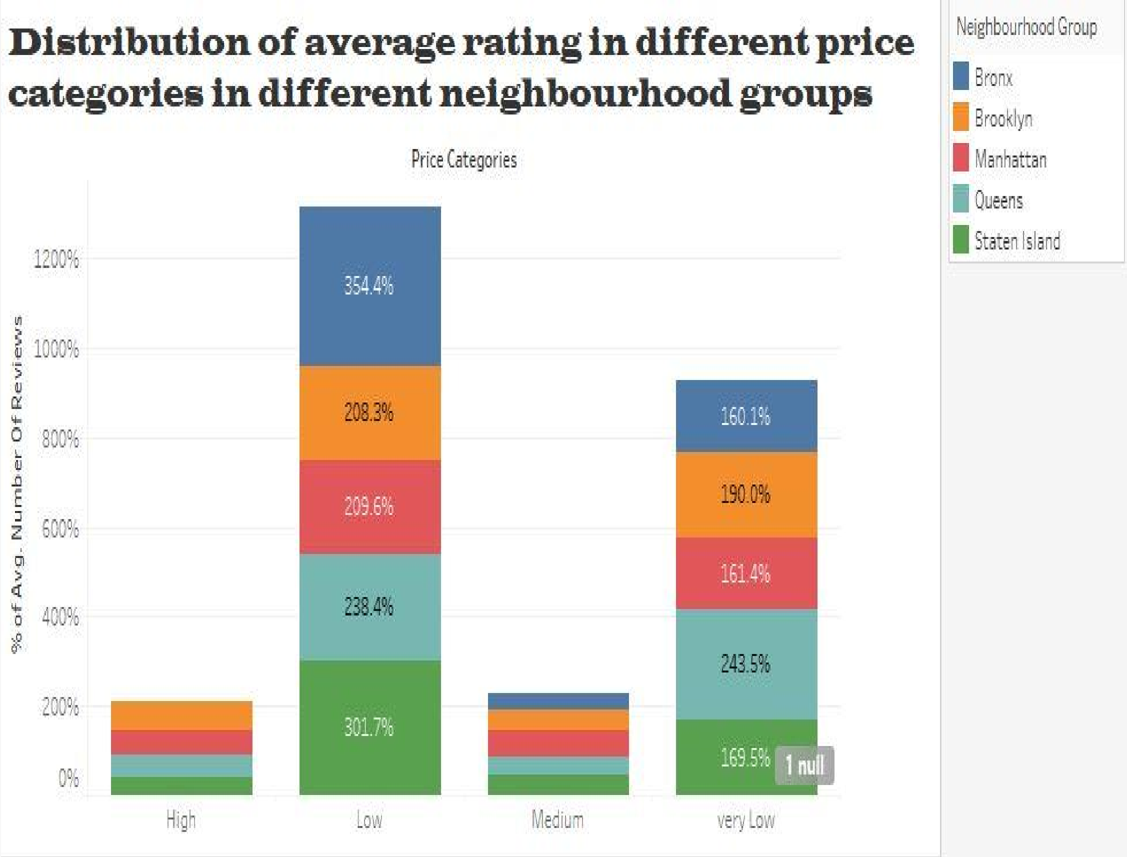


- The distribution of locations clearly indicates density.
- We see that, Airbnb has good presence in Manhattan, Brooklyn & Queens.
- Listings are maximum in Manhattan & Brooklyn owing to the high population density and it being the financial and tourism hub of NYC. Staten Island has the least number of listings.



- The distribution indicates average ratings across different neighbourhood groups
- The higher number of customer reviews imply higher satisfaction in these localities.

Price Categories Vs Average rating Vs Neighbourhood groups



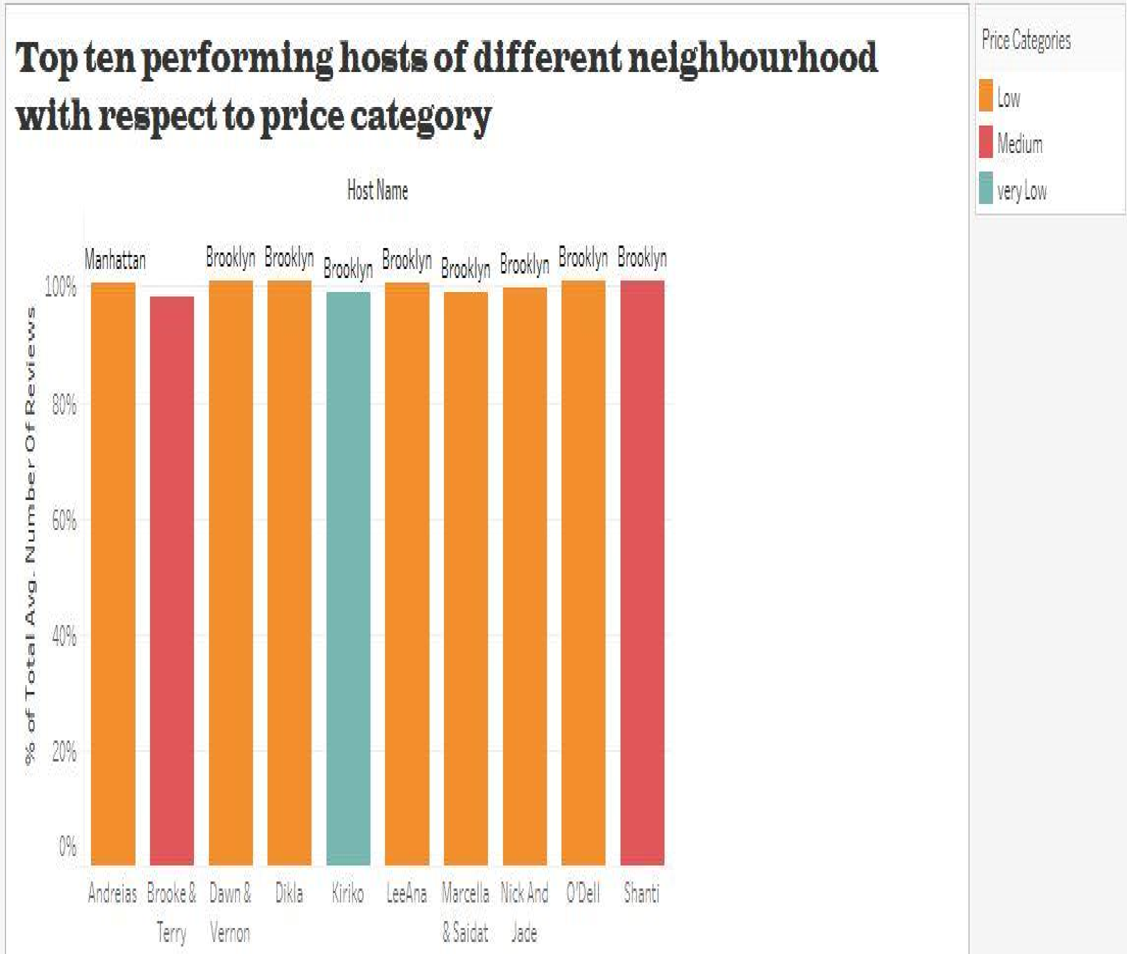
Caption

% of Avg. Number Of Reviews for each Price Categories. Color shows details about Neighbourhood Group.

- **Low Price category has the highest average ratings**
- **Neighbourhood group Bronx has the highest share followed by Queens**

Low Performing of Hosts and localities

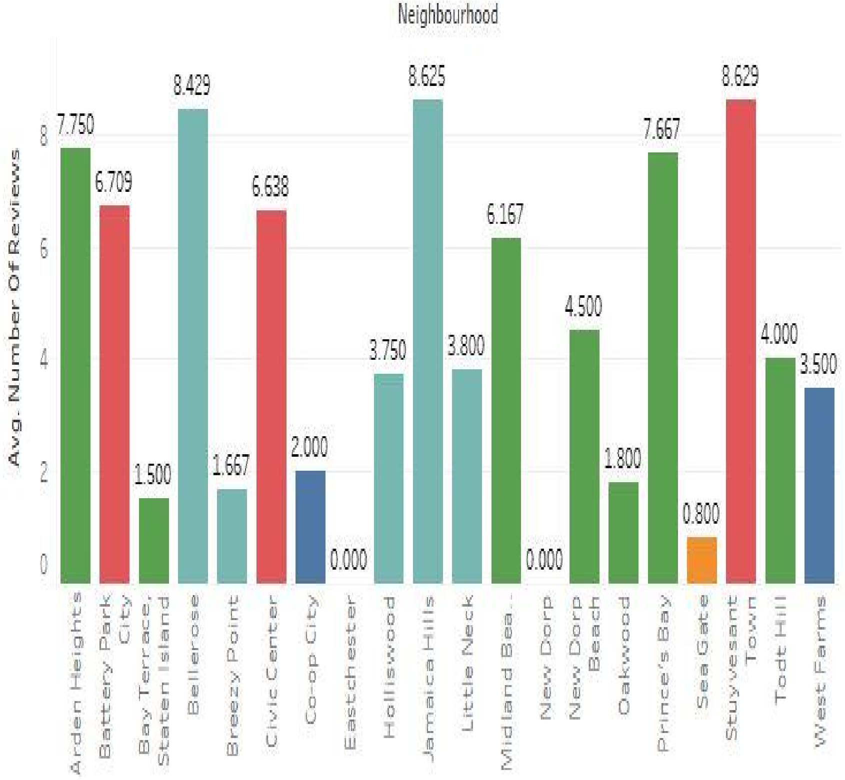
Top ten performing hosts of different neighbourhood with respect to price category



Caption

% of Total Avg. Number Of Reviews for each Host Name. Color shows details about Price Categories. The marks are labeled by Neighbourhood Group. The view is filtered on Host Name, which keeps 10 of 11,047 members.

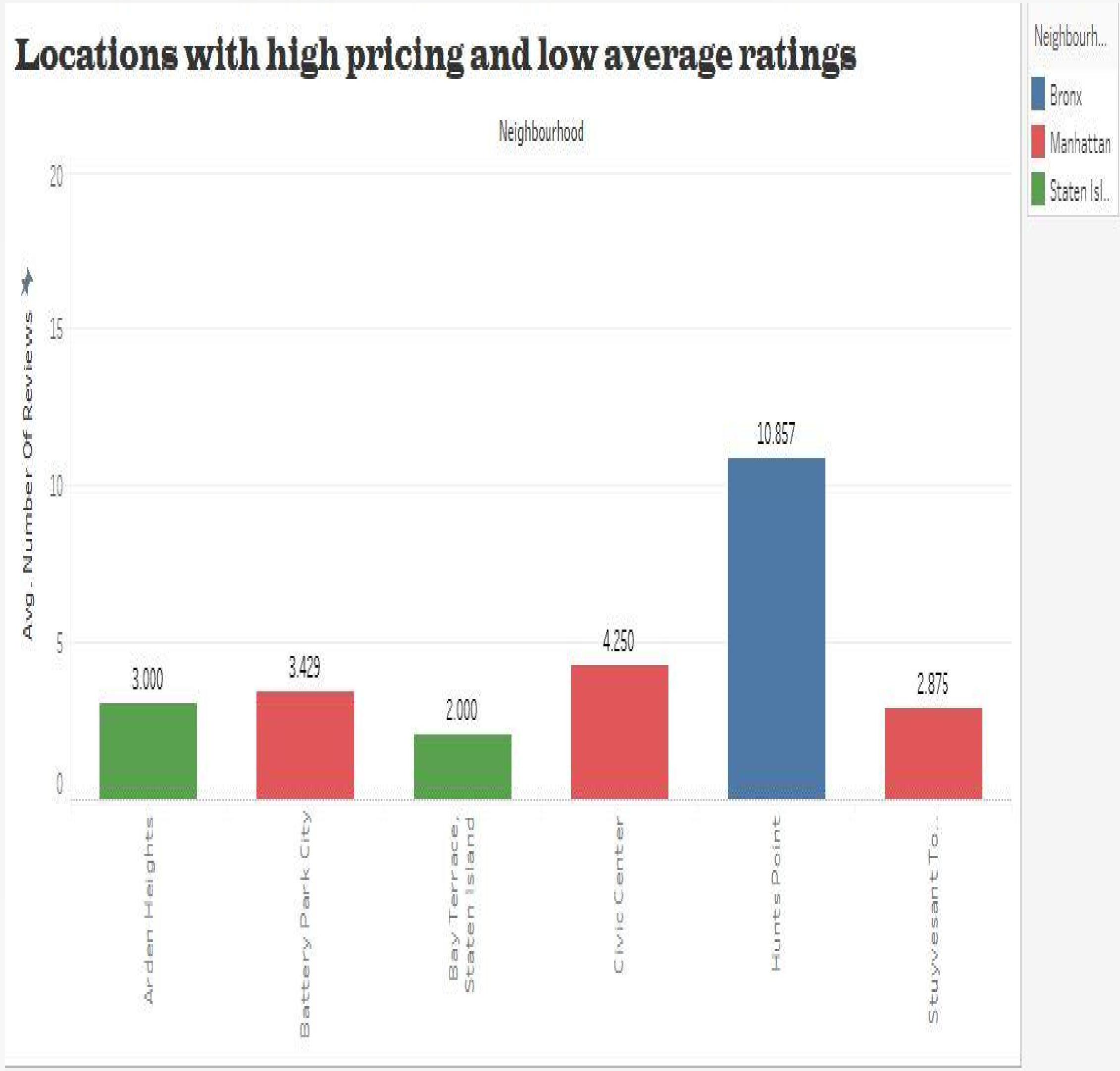
Least performing 20 locations in their respective neighbourhood



Caption

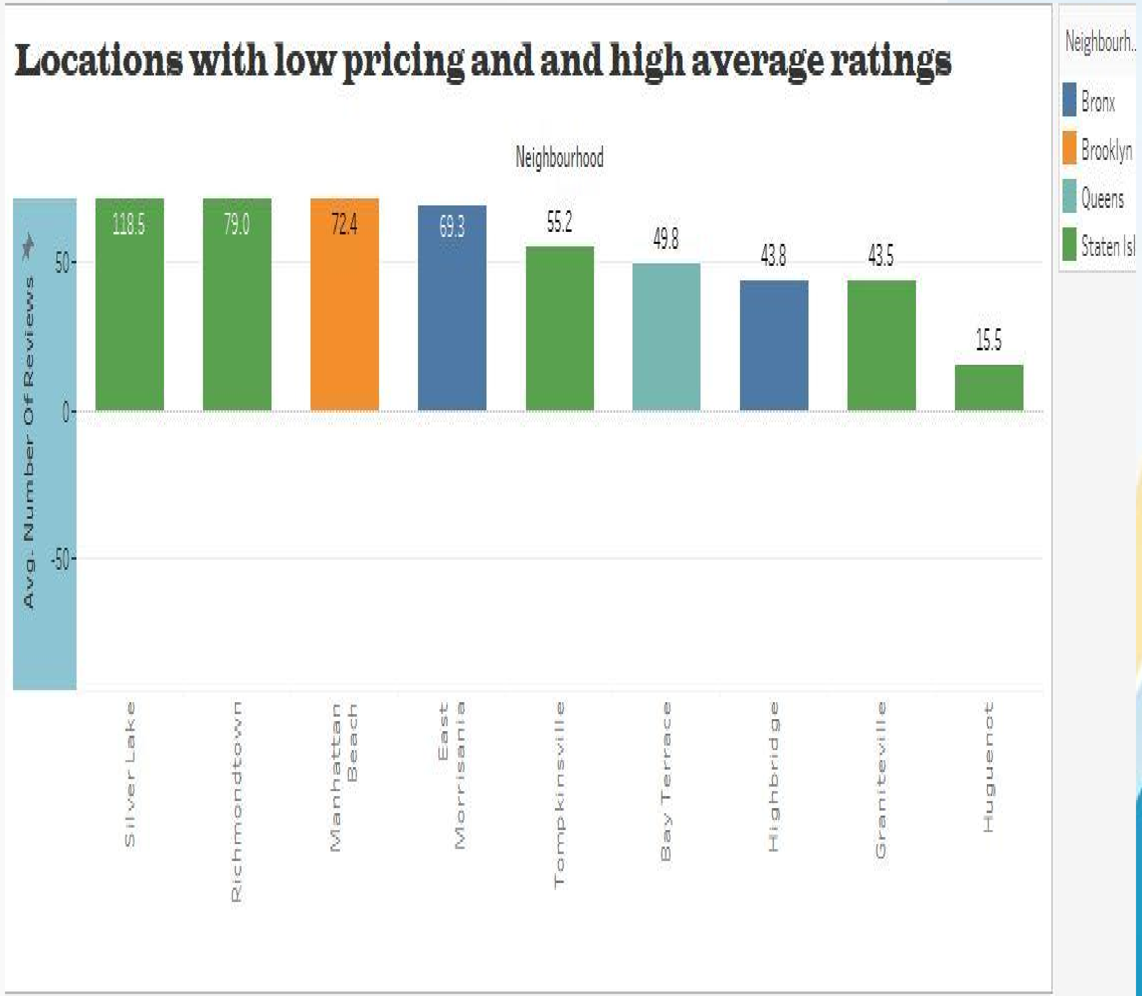
Average of Number Of Reviews for each Neighbourhood. Color shows details about Neighbourhood Group. The view is filtered on Neighbourhood, which keeps 20 of 219 members.

Pricing Vs Average reviews received



Caption

Average of Number Of Reviews for each Neighbourhood. Color shows details about Neighbourhood Group. Details

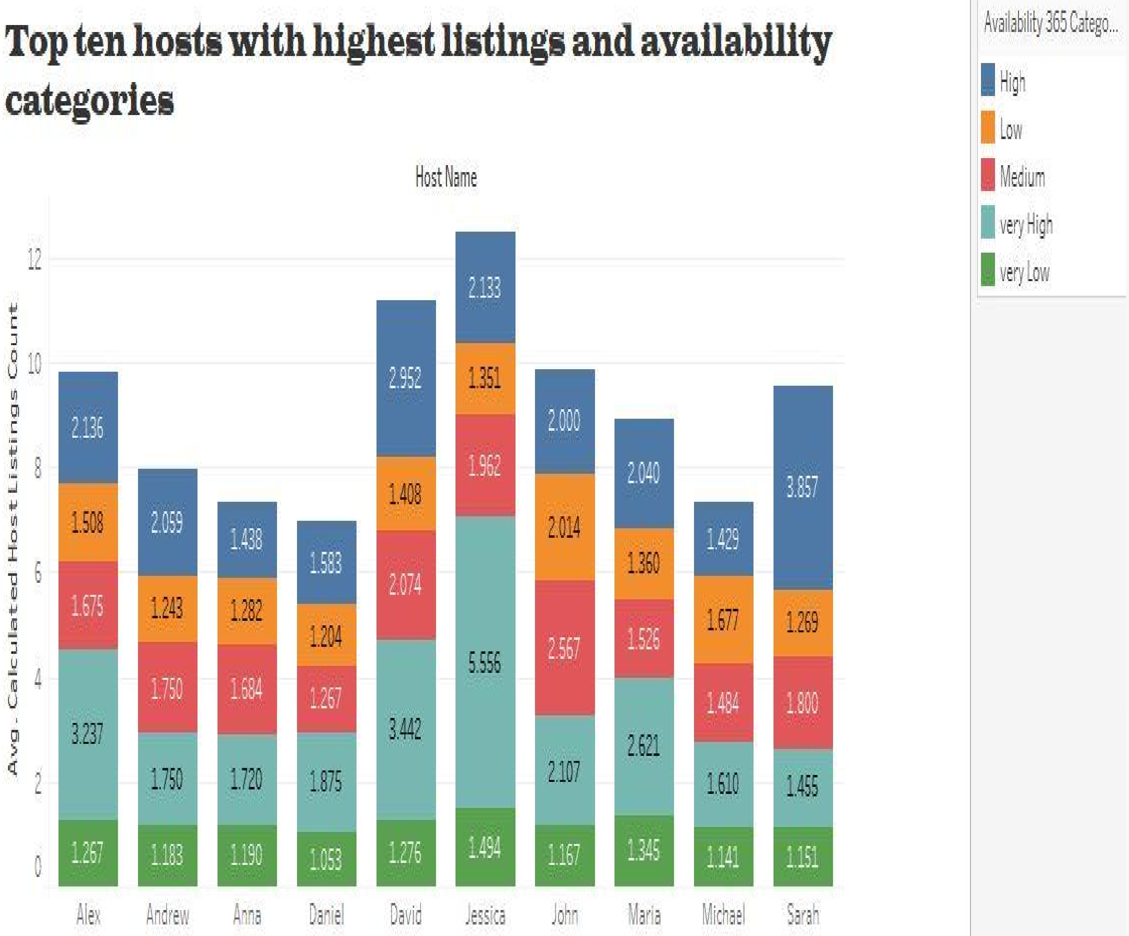


Caption

Average of Number Of Reviews for each Neighbourhood. Color shows details about Neighbourhood Group. Details are shown for Price Categories. The view is filtered on Neighbourhood and Price Categories. The Neighbourhood filter keeps 30 of 219 members. The Price Categories filter keeps High.

Host listing count and their availibility in respective neighbourhood

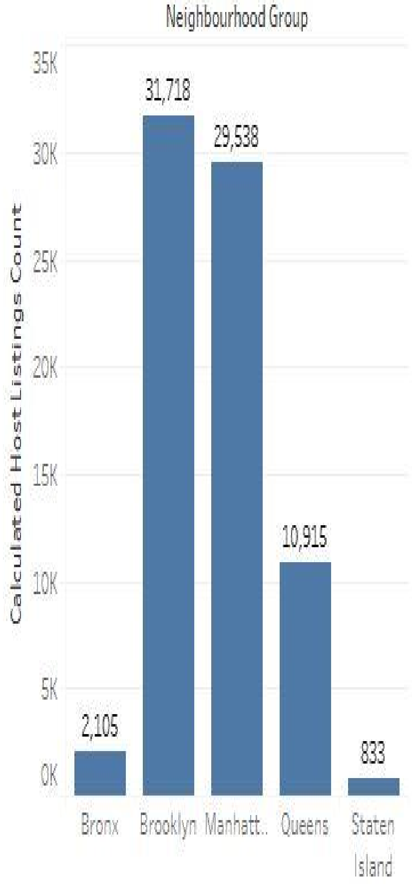
Top ten hosts with highest listings and availability categories



Caption

Average of Calculated Host Listings Count for each Host Name. Color shows details about Availability 365 Categories. The view is filtered on Host Name, which keeps 10 of 11,047 members.

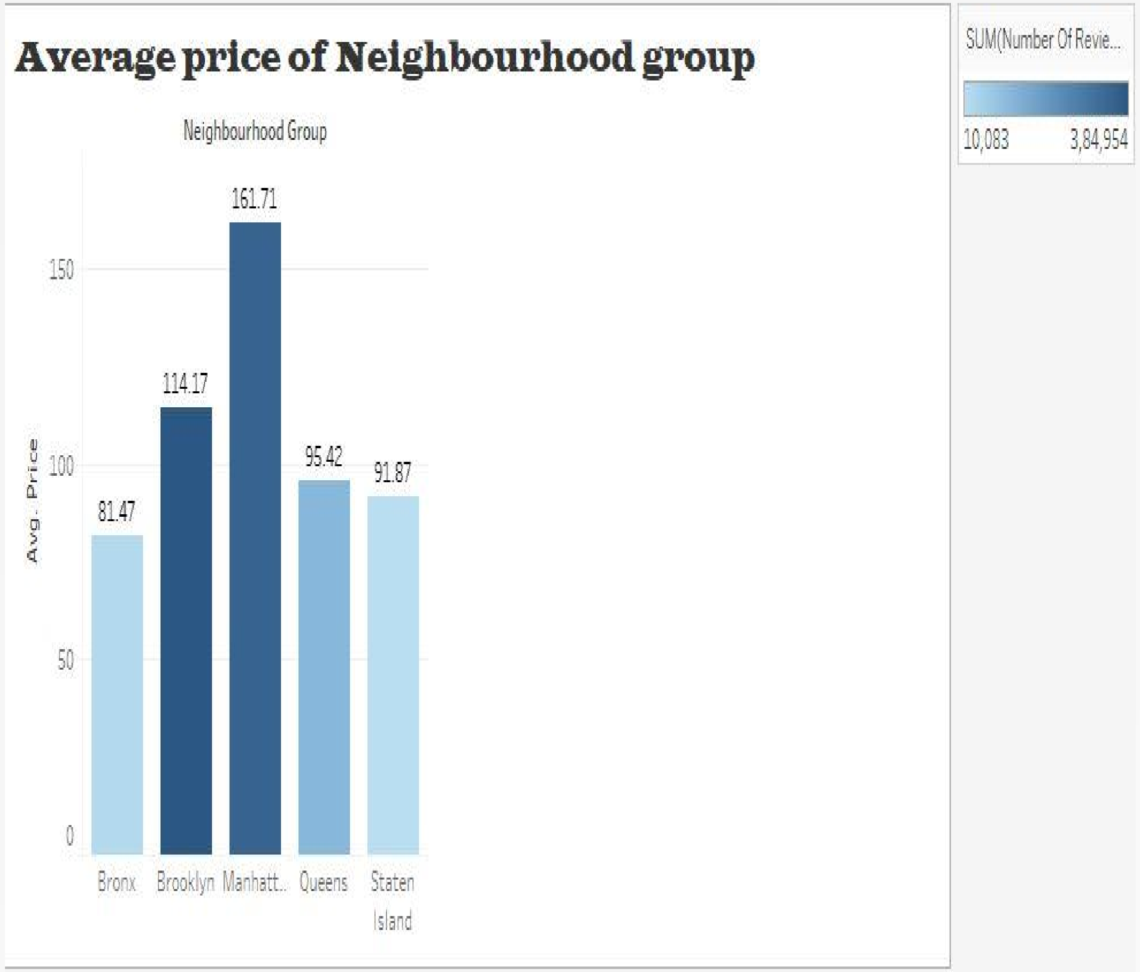
Calculated host listing Vs Neighbourhood groups



Caption

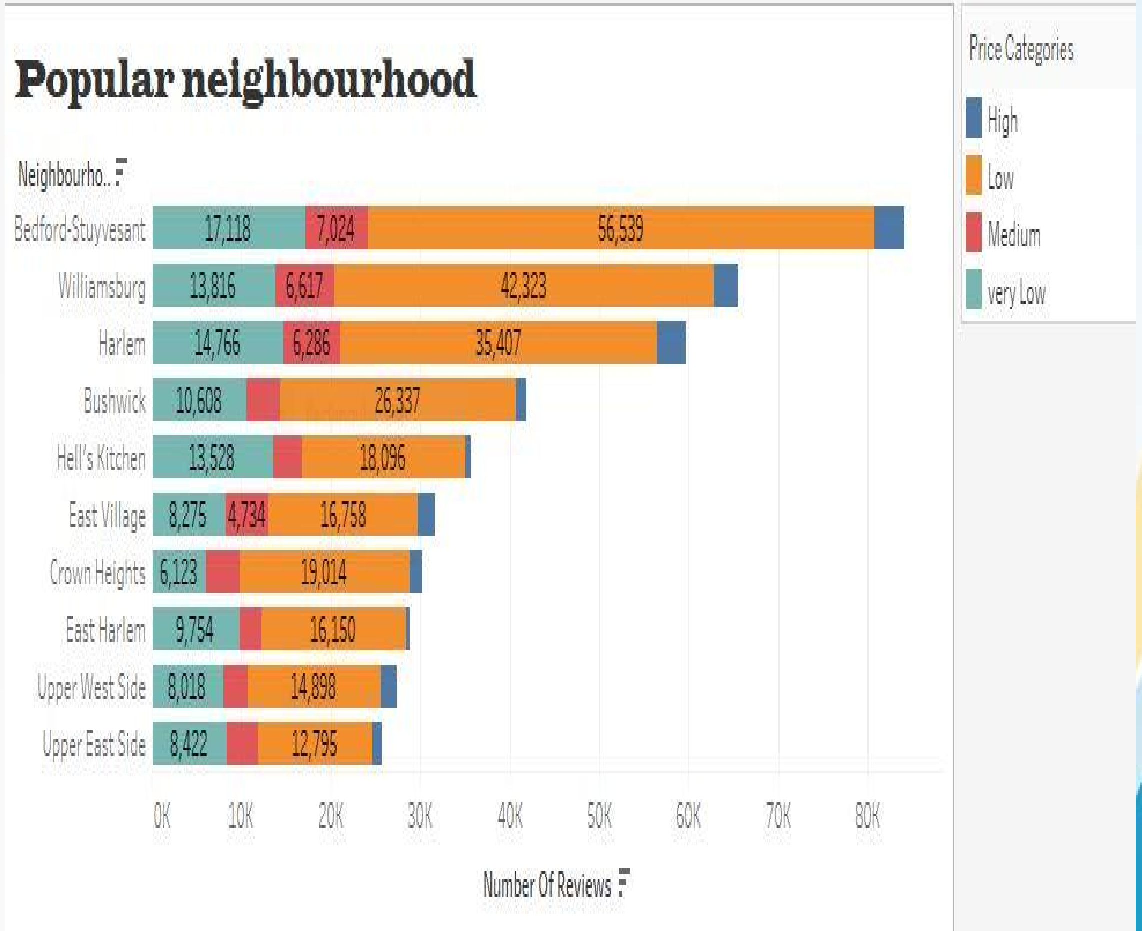
Sum of Calculated Host Listings Count for each Neighbourhood Group.

Neighbourhood average price and popular neighbour hood



Caption

Average of Price for each Neighbourhood Group. Color shows sum of Number Of Reviews.



Recommendations and conclusions

- **Low Pricing category has recieved better avg reviews, Moderate price or reduction in price may be considered to attract more customers.**
- **identified Least performing locations may be looked upon and initiate necessary improvements**
- **Shared rooms is a vital area lagging behind, these to be checked upon.**
- **The cumulative contribution of all hosts is better than a few hosts doing well.**
- **More than 80 % of the listing are Manhattan and Brooklyn neighborhood group.**
- **Minimum nights threshold should be on the lower side to make propertiesmore customer-oriented.**
- **Data collection team should collect data about review scores so that it can strengthen the later analysis.**
- **A clustering machine learning model to identify groups of similar objects in datasets with two or more variable quantities can be made.**