Methodology adopted in to analyse a dataset consisting of various Airbnb listings in New York.

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)
```

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
```

```
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%.

###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	n
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	

In [8]:

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

In [9]:

```
abnyc.reviews_per_month.isnull().sum()
```

Out[9]:

```
0
```

In [10]:

```
round((abnyc.isnull().sum()/len(abnyc))*100,2)
```

Out[10]:

```
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                  0.00
calculated_host_listings_count     0.00
availability_365                   0.00
dtype: float64
```

In [11]:

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

Out[11]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	l
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	4
19	7750	Huge 2 BR Upper East Cental Park	17985	Sing	Manhattan	East Harlem	4
26	8700	Magnifique Suite au N de Manhattan - vue Cloitres	26394	Claude & Sophie	Manhattan	Inwood	4
36	11452	Clean and Quiet in Brooklyn	7355	Vt	Brooklyn	Bedford-Stuyvesant	4
38	11943	Country space in the city	45445	Harriet	Brooklyn	Flatbush	4
...	...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	4
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	4
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	4
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	4
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	4

10052 rows × 16 columns

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
```

Data types

In [14]:

```
# Extracting Numeric columns:

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

In [15]:

```
list(enumerate(int_cols))
```

Out[15]:

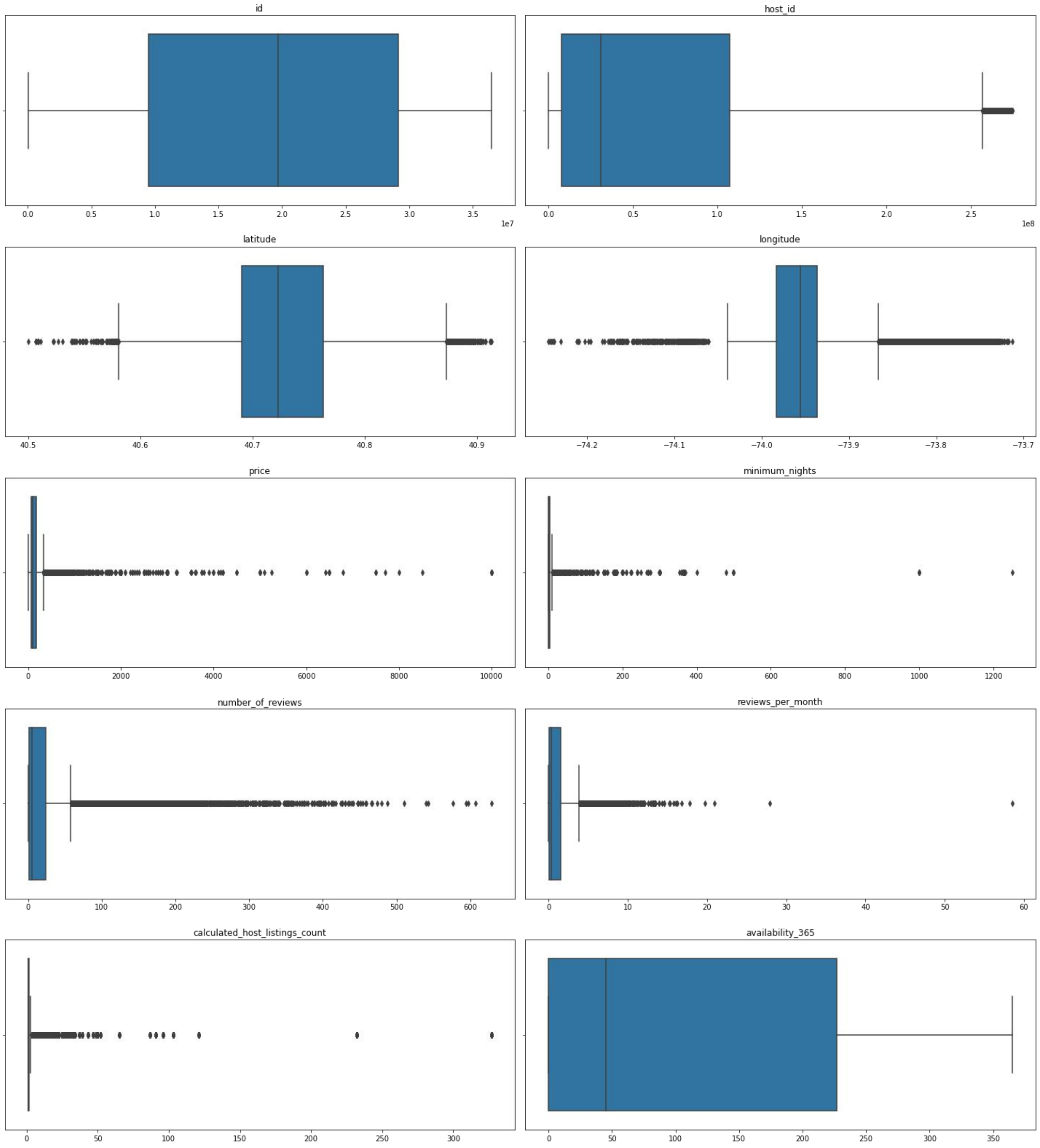
```
[(0, 'id'),
 (1, 'host_id'),
 (2, 'latitude'),
 (3, 'longitude'),
 (4, 'price'),
 (5, 'minimum_nights'),
 (6, 'number_of_reviews'),
 (7, 'reviews_per_month'),
 (8, 'calculated_host_listings_count'),
 (9, 'availability_365')]
```

Evaluating outliers

In [16]:

```
# Plotting the spread of outliers:

plt.figure(figsize=([20,22]))
for n,col in enumerate(int_cols):
    plt.subplot(5,2,n+1)
    sns.boxplot(abnyc[col], orient  =  "h")
    plt.xlabel("")
    plt.ylabel("")
    plt.title(col)
    plt.tight_layout()
```



In [17]:
```
# outlier treatment for price:
Q1  =  abnyc.price.quantile(0.10)
Q3  =  abnyc.price.quantile(0.90)
IQR  = Q3-Q1
abnyc = abnyc[(abnyc.price >= Q1-1.5*IQR) & (abnyc.price <= Q3 + 1.5*IQR)]
```

In [18]:
```
# outlier treatment for minimum_nights:
Q1  =  abnyc.minimum_nights.quantile(0.10)
Q3  =  abnyc.minimum_nights.quantile(0.90)
IQR  = Q3-Q1
abnyc = abnyc[(abnyc.minimum_nights >= Q1-1.5*IQR) & (abnyc.minimum_nights <= Q3 +
```

In [19]:
```
# outlier treatment for minimum_nights:
Q1  =  abnyc.number_of_reviews.quantile(0.10)
Q3  =  abnyc.number_of_reviews.quantile(0.90)
IQR  = Q3-Q1
abnyc  =  abnyc[(abnyc.number_of_reviews  >=  Q1-1.5*IQR)  &  (abnyc.number_of_reviews  <
```

In [20]:
```
# outlier treatment for reviews_per_month:
Q1  =  abnyc.reviews_per_month.quantile(0.10)
Q3  =  abnyc.reviews_per_month.quantile(0.90)
```

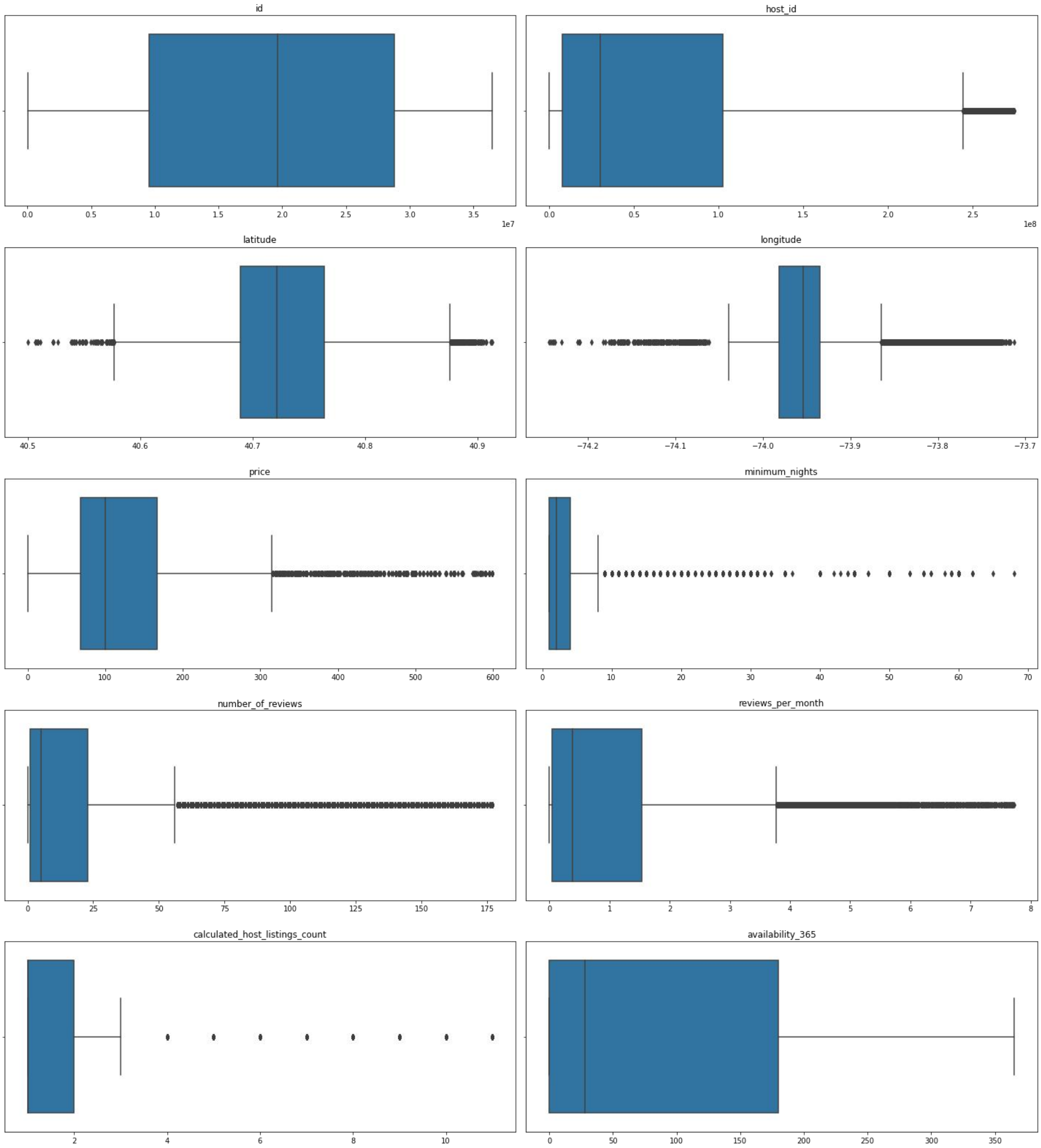
```
IQR = Q3-Q1
abnyc = abnyc[(abnyc.reviews_per_month >= Q1-1.5*IQR) & (abnyc.reviews_per_month <
```

In [21]:

```
# outlier treatment for calculated_host_listings_count:
Q1  =  abnyc.calculated_host_listings_count.quantile(0.10)
Q3  =  abnyc.calculated_host_listings_count.quantile(0.90)
IQR = Q3-Q1
abnyc= abnyc[(abnyc.calculated_host_listings_count >= Q1-1.5*IQR) & (abnyc.calcula
```

In [22]:

```
plt.figure(figsize=([20,22]))
for n,col in enumerate(int_cols):
    plt.subplot(5,2,n+1)
    sns.boxplot(abnyc[col], orient = "h")
    plt.xlabel("")
    plt.ylabel("")
    plt.title(col)
    plt.tight_layout()
```



Analysing Categorical and Numeric values

In [23]:

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

In [24]:

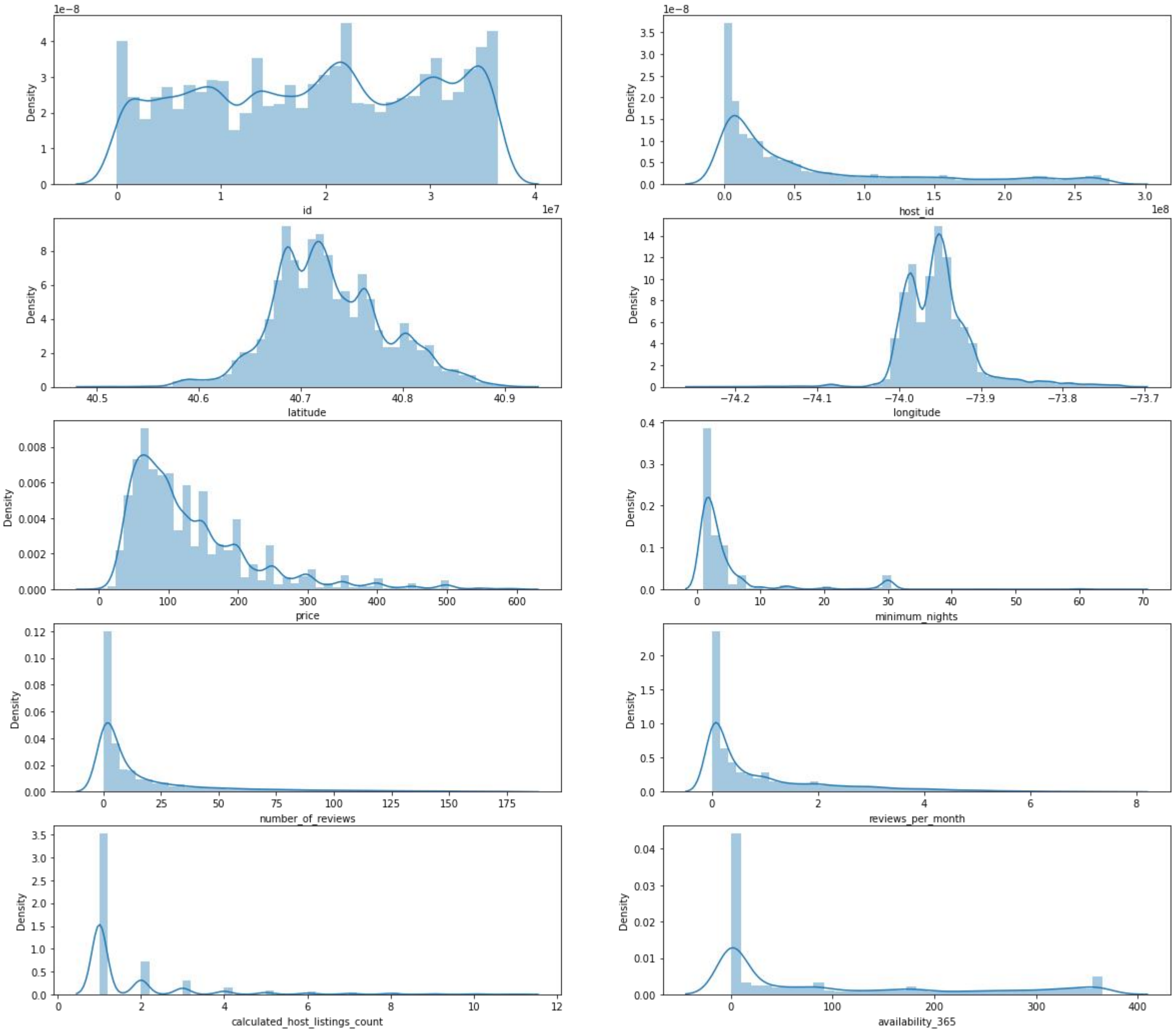
```
list(enumerate(cat_cols))
```

Out[24]:

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

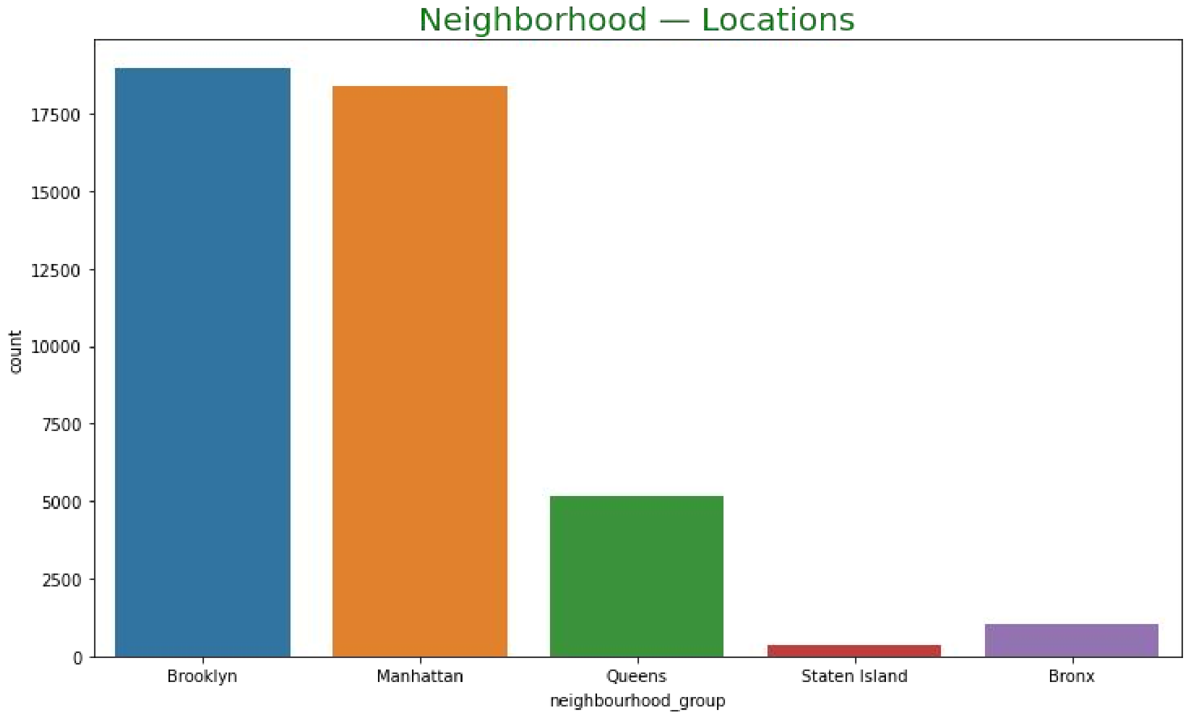
In [25]:

```
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])
```

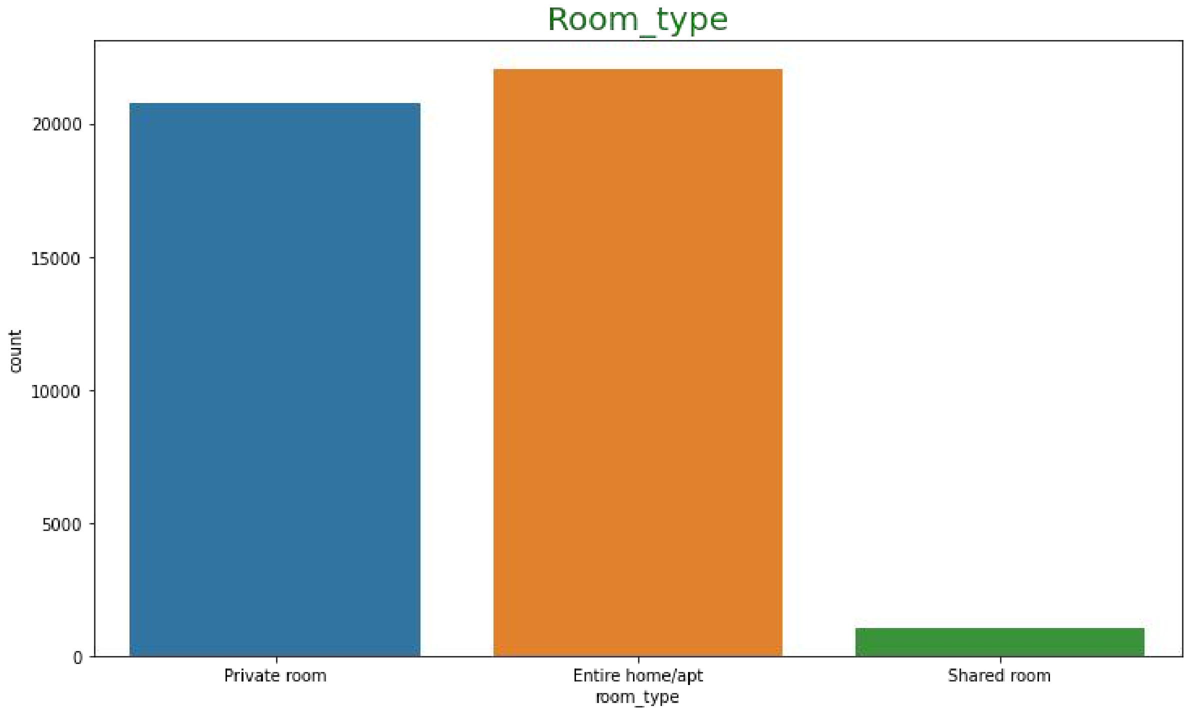


In [26]:

```
plt.figure(figsize=[12,7])
sns.countplot(abnyc.neighbourhood_group)
plt.title('Neighborhood – Locations', fontdict={'fontsize': 20, 'fontweight': 5, '
plt.show()
```



In [27]:
```
plt.figure(figsize=[12,7])
sns.countplot(abnyc.room_type)
plt.title('Room_type', fontdict={'fontsize': 20, 'fontweight': 5, 'color': 'Green'
plt.show()
```



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
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]:

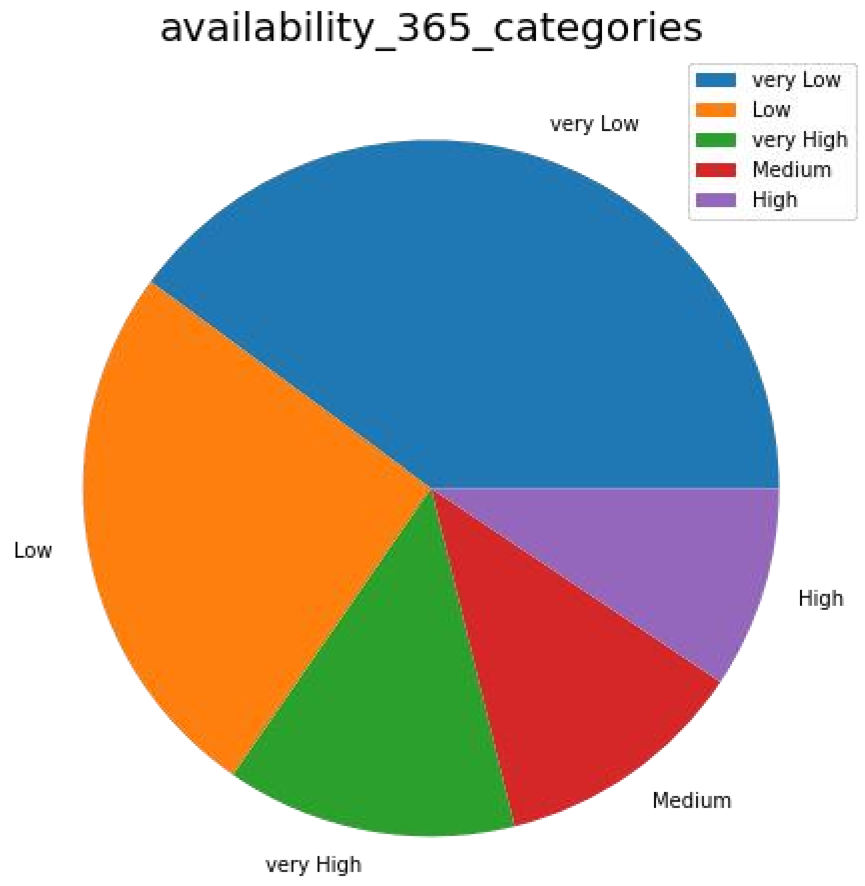
```
abnyc['availability_365_categories'].value_counts()
```

Out[30]:

```
very Low     17523
Low          11182
very High     5921
Medium        5170
High          4116
Name: availability_365_categories, dtype: int64
```

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,
plt.legend()
plt.show()
```



categorizing the "minimum_nights" column into 5 categories

```
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_categor
         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]:

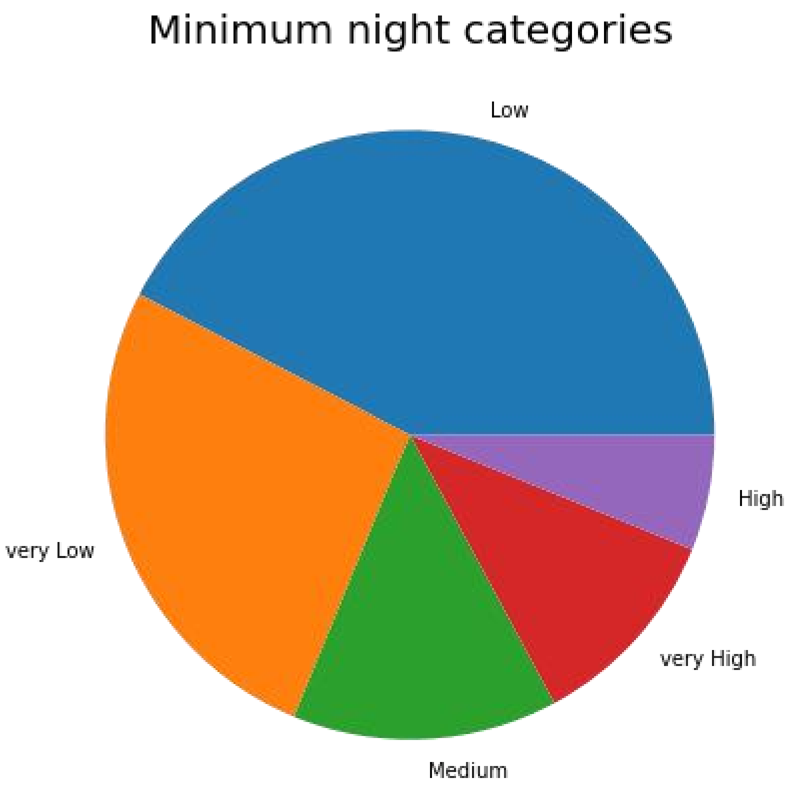
```
abnyc.minimum_night_categories.value_counts()
```

Out[34]:

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

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_nig
plt.show()
```



In [36]:

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

In [37]:

```
abnyc['number_of_reviews_categories'] = abnyc.minimum_nights.map(number_of_reviews
abnyc['number_of_reviews_categories']
```

Out[37]:

```
0         very Low
1         very Low
2              Low
4           Medium
5              Low
            ...   
48890          Low
48891          Low
48892       Medium
48893     very Low
48894       Medium
Name:  number_of_reviews_categories,  Length:  43912,  dtype:  object
```

In [38]:

```
###categorizing the "price" column into 5 categories
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 [39]:

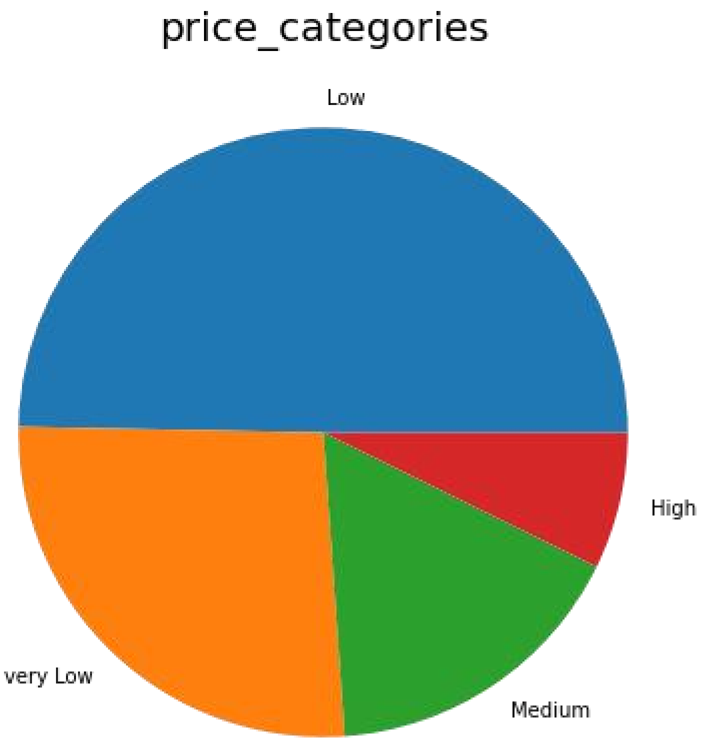
```
abnyc['price_categories'] = abnyc.minimum_nights.map(price_categories_function)
abnyc['price_categories']
```

Out[39]:

```
0         very Low
1         very Low
2              Low
4           Medium
5              Low
            ...   
48890          Low
48891          Low
48892       Medium
48893     very Low
48894       Medium
Name: price_categories, Length: 43912, dtype: object
```

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.va
plt.show()
```



In [41]:

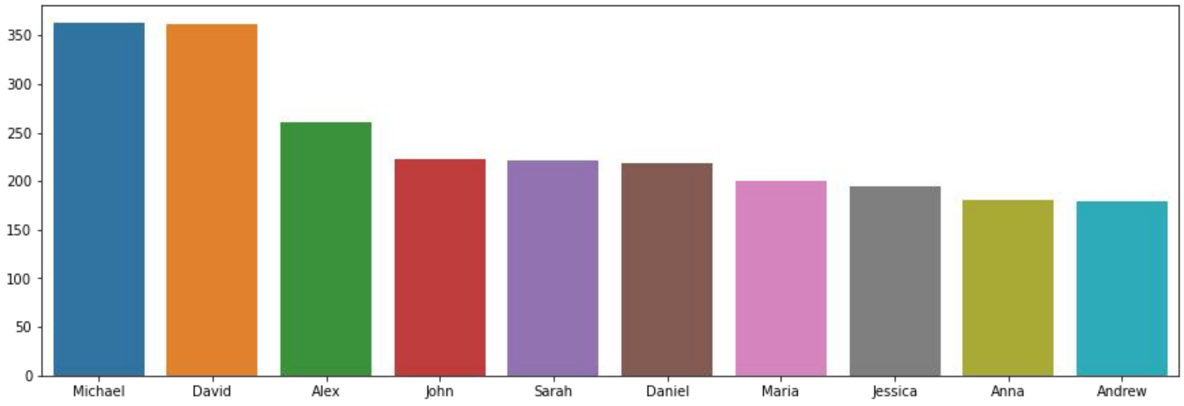
```
abnyc.host_name.value_counts()
```

Out[41]:

```
Michael          363
David            362
Alex             260
John             223
Sarah            221
                ... 
Nkoli              1
Lyles              1
Ubi                1
Yah                1
Ilgar & Aysel      1
Name: host_name, Length: 11046, dtype: int64
```

In [42]:

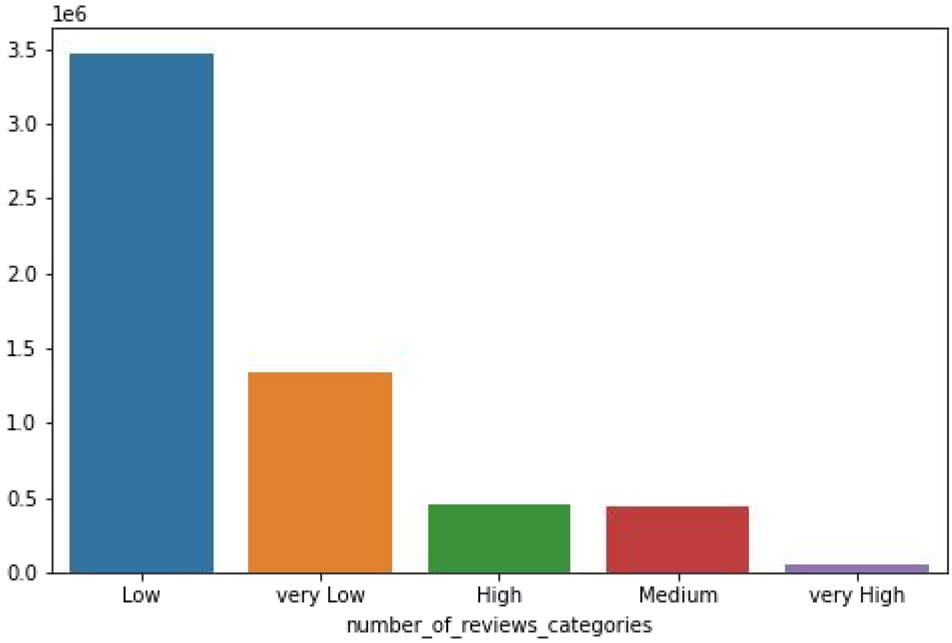
```
# Top 10 host's
plt.figure(figsize=(15,5))
sns.barplot(x = abnyc.host_name.value_counts().index[:10] , y = abnyc.host_name.va
plt.show()
```



In [43]:

```
# prices for each of reviews_categories
x1 = abnyc.groupby('number_of_reviews_categories').price.sum().sort_values(ascendi
plt.figure(figsize=(8,5))
```

```
sns.barplot(x = x1.index,y = x1.values)
plt.show()
```



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

Out[44]:

		reviews_per_month
availability_365_categories	price_categories	
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

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 [45]:

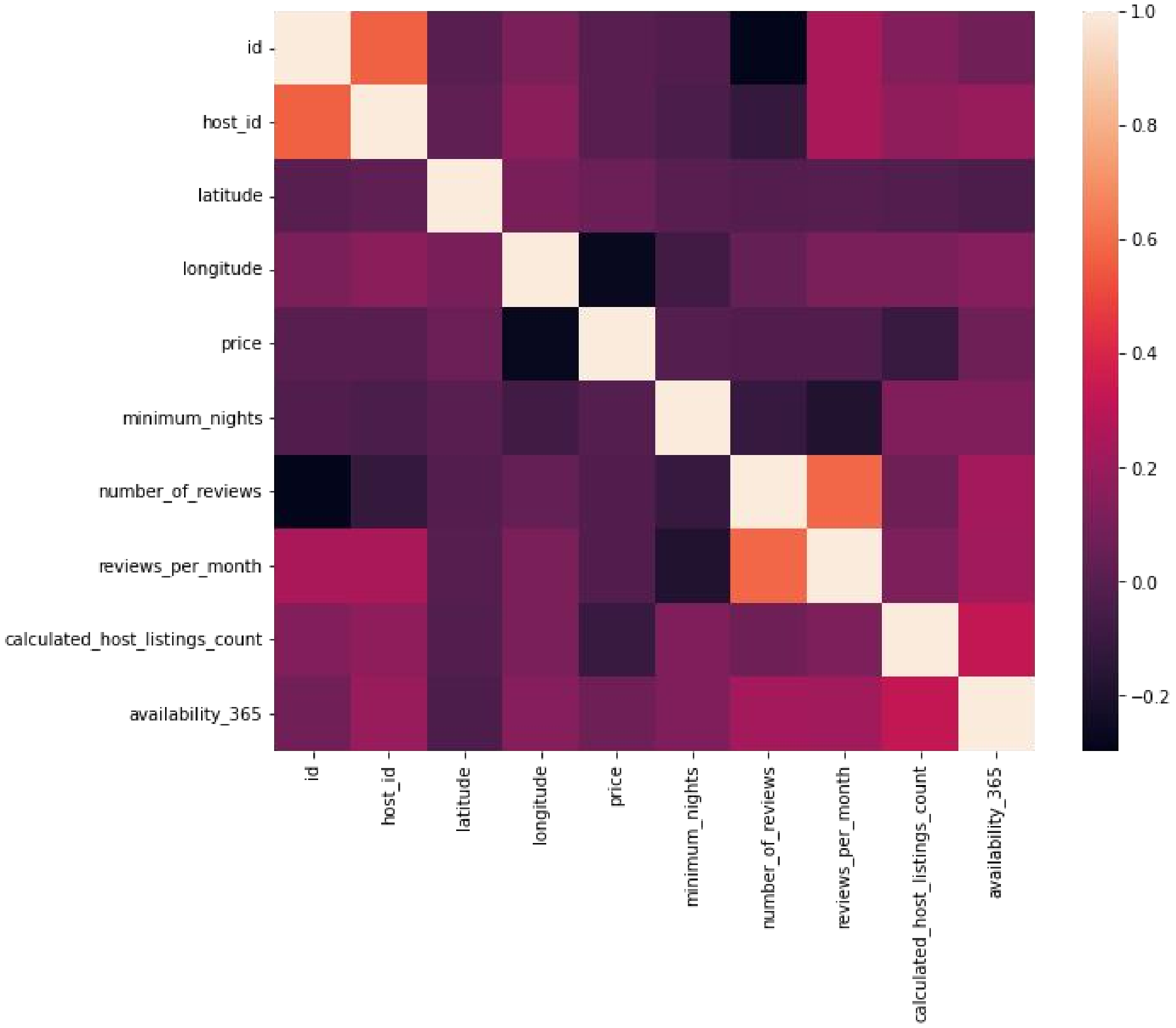
```
abnyc.groupby('minimum_night_categories').reviews_per_month.sum().sort_values()
```

Out[45]:

```
minimum_night_categories
High          1131.35
very High     1793.87
Medium        4432.61
very Low     15780.71
Low          22350.91
Name: reviews_per_month, dtype: float64
```

In [46]:

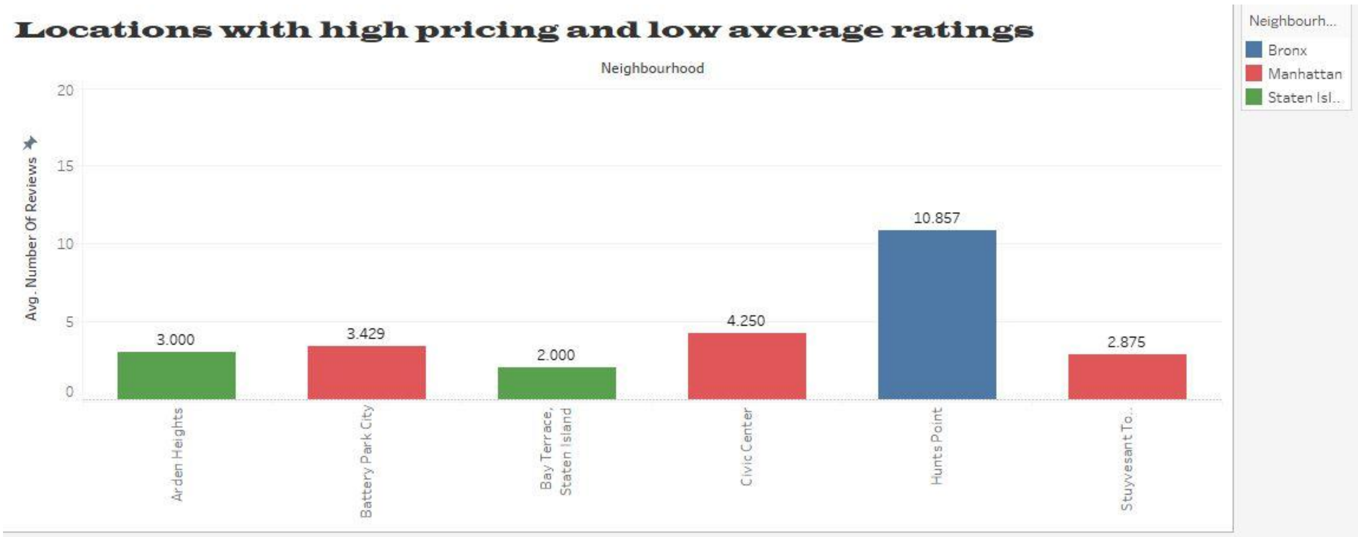
```
plt.figure(figsize=(10,8))
sns.heatmap(data = abnyc[int_cols].corr())
plt.show()
```



This shows correlation among the features

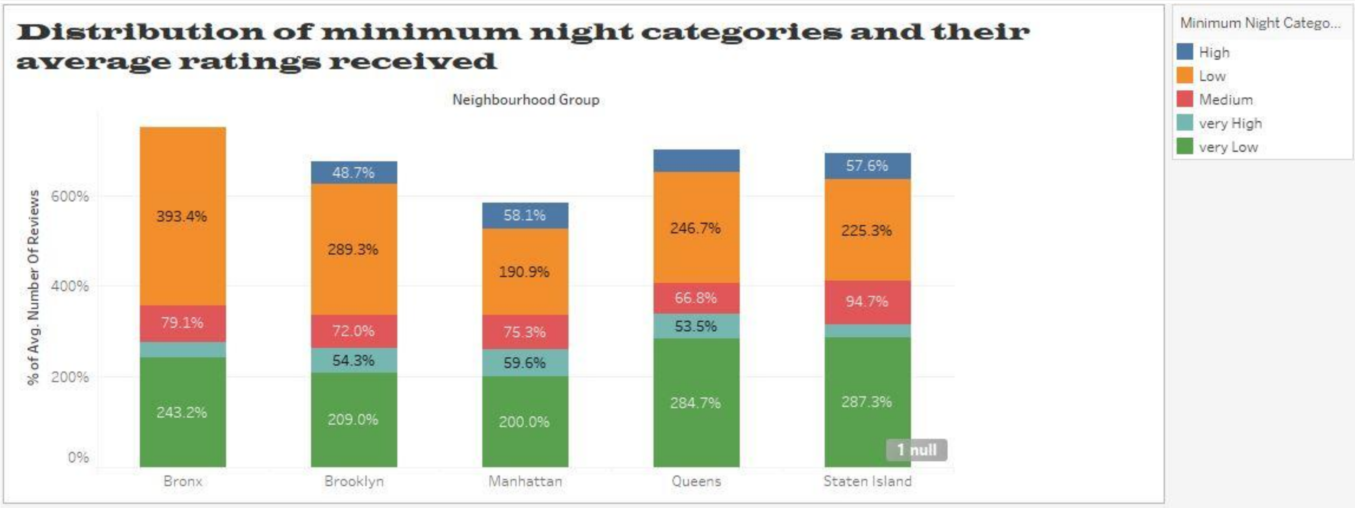
In [47]:
```
abnyc.to_csv('AB_NYC_2019_processed.csv')
```

In [ ]:


Locations with high pricing and low average ratings

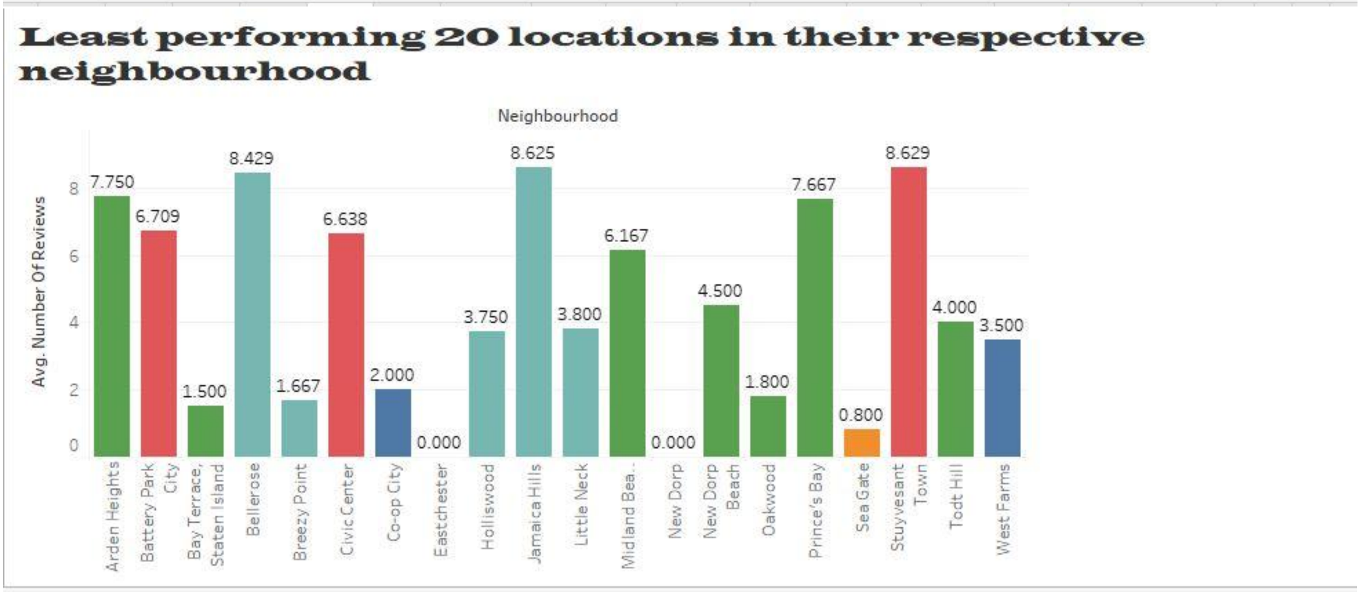
Caption

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


Distribution of minimum night categories and their average ratings received

Caption

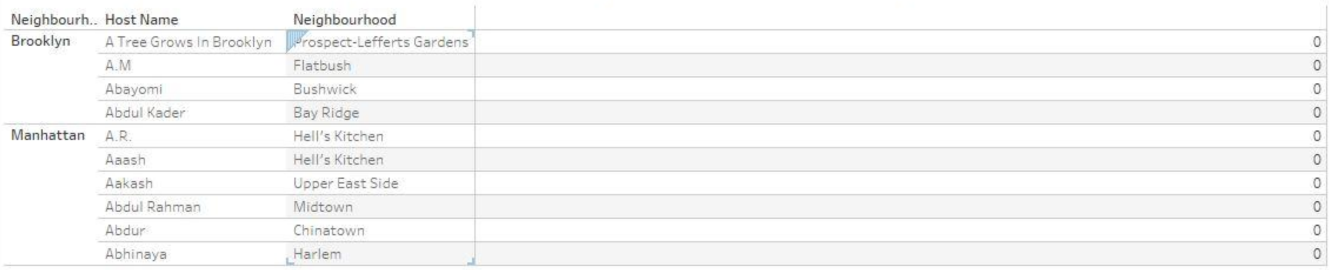
% of Avg. Number Of Reviews for each Neighbourhood Group. Color shows details about Minimum Night Categories.


Least performing 20 locations in their respective neighbourhood

Caption

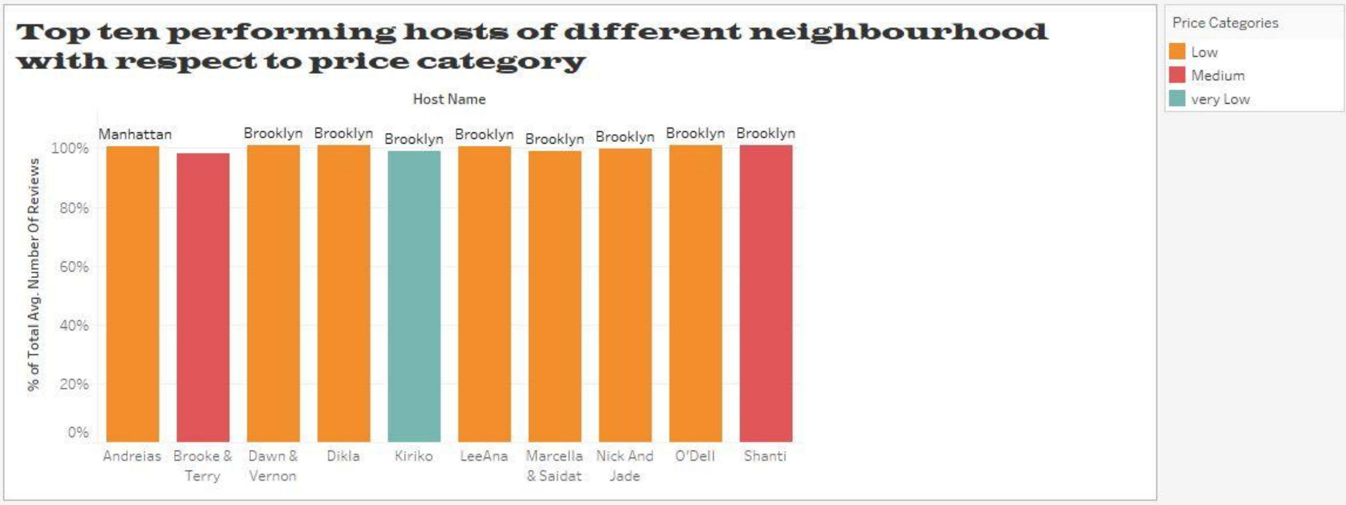
Average of Number Of Reviews for each Neighbourhood. Color shows details about Neighbourhood Group. The view is filtered on Neighbourhood, which keeps 20 of 219 members.

Least performing ten hosts as per review ratings



Neighbourh..	Host Name	Neighbourhood	
Brooklyn	A Tree Grows In Brooklyn	Prospect-Lefferts Gardens	0
	A.M	Flatbush	0
	Abayomi	Bushwick	0
	Abdul Kader	Bay Ridge	0
Manhattan	A.R.	Hell's Kitchen	0
	Aaash	Hell's Kitchen	0
	Aakash	Upper East Side	0
	Abdul Rahman	Midtown	0
	Abdur	Chinatown	0
	Abhinaya	Harlem	0

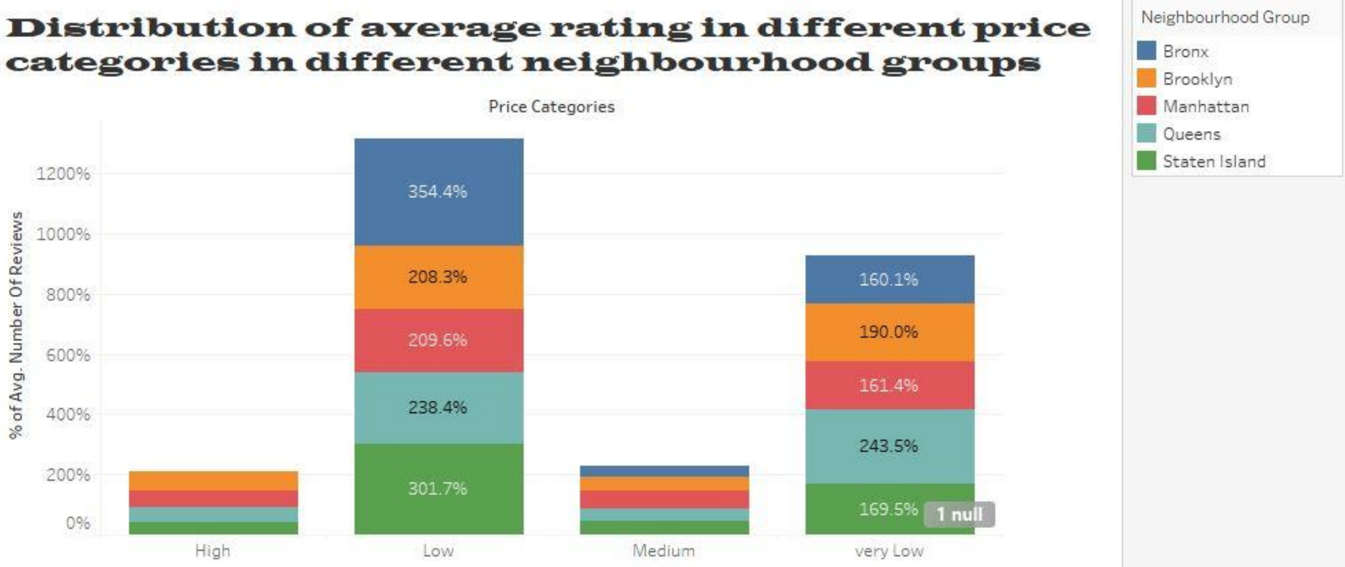
Top ten performing hosts of different neighbourhood with respect to price category



Caption

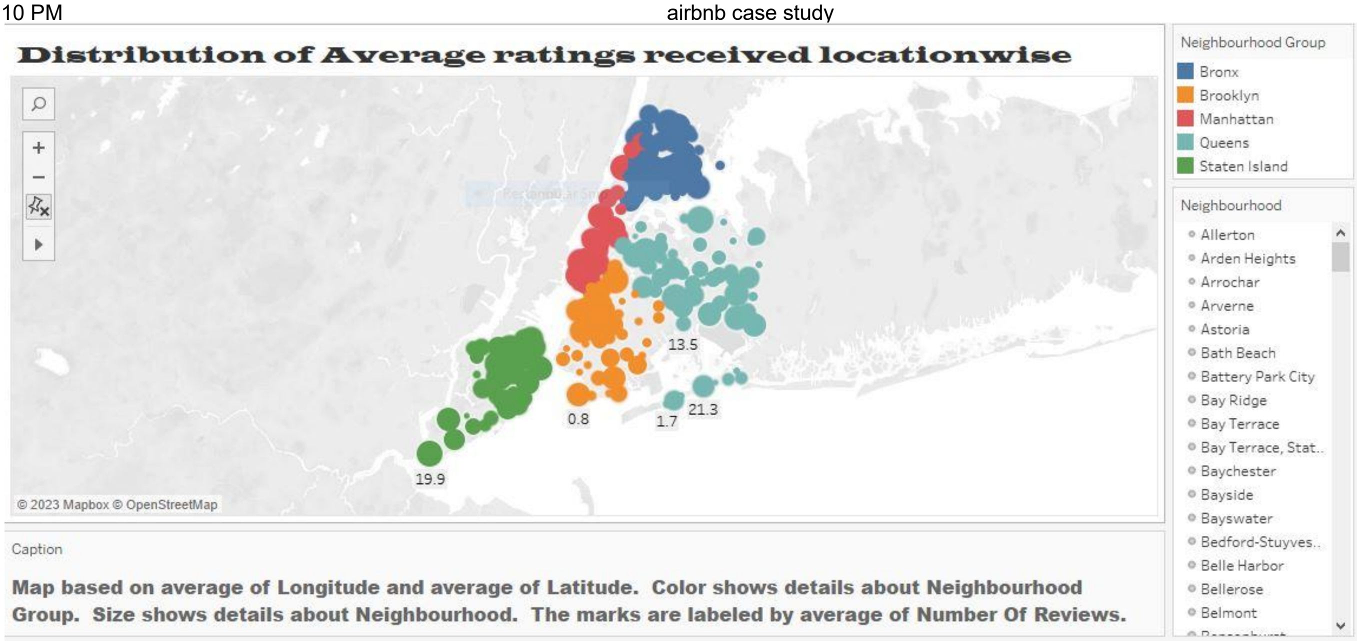
% of Total Avg. Number Of Reviews for each Host Name. Color shows details about Price Categories. The marks are labeled by Neighbourhood Group. The view is filtered on Host Name, which keeps 10 of 11,047 members.

Distribution of average rating in different price categories in different neighbourhood groups



Caption

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



Distribution of Average ratings received locationwise

Caption

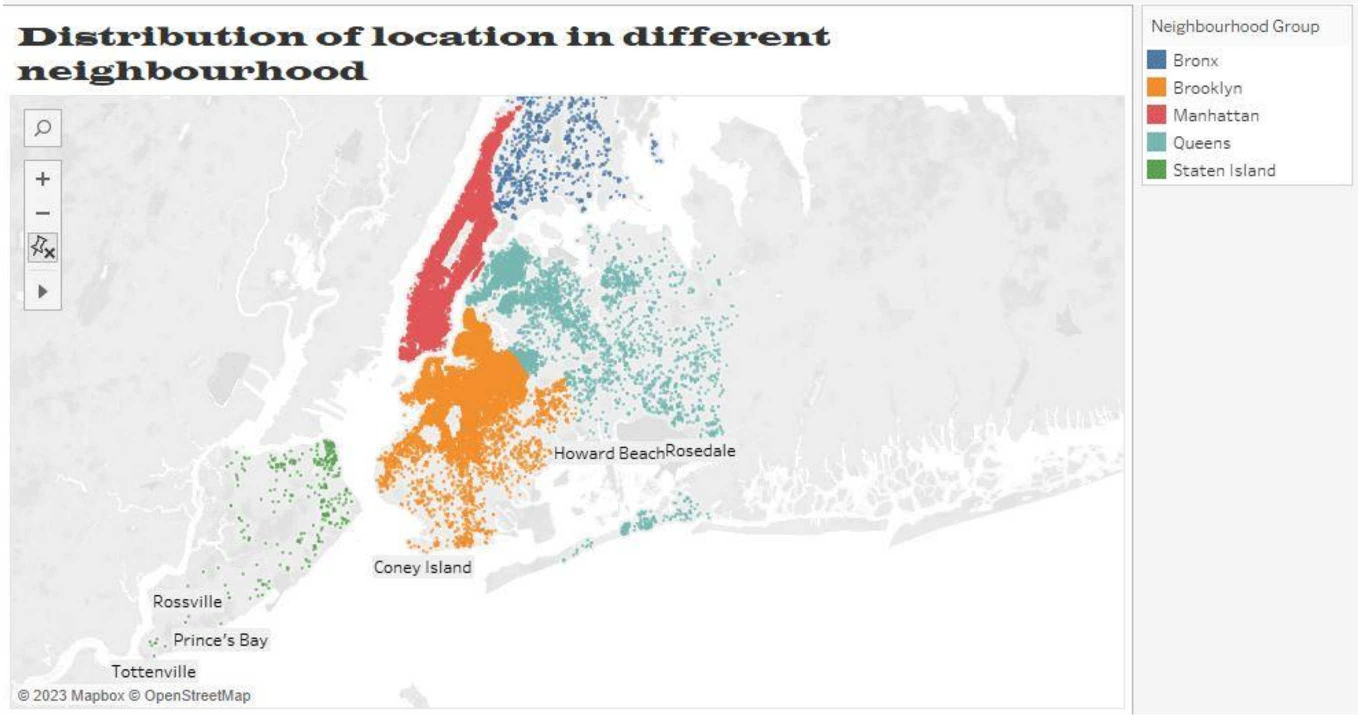
Map based on average of Longitude and average of Latitude. Color shows details about Neighbourhood Group. Size shows details about Neighbourhood. The marks are labeled by average of Number Of Reviews.

Neighbourhood Group

- 🟦 Bronx
- 🟧 Brooklyn
- 🟥 Manhattan
- 🟦 Queens
- 🟩 Staten Island

Neighbourhood

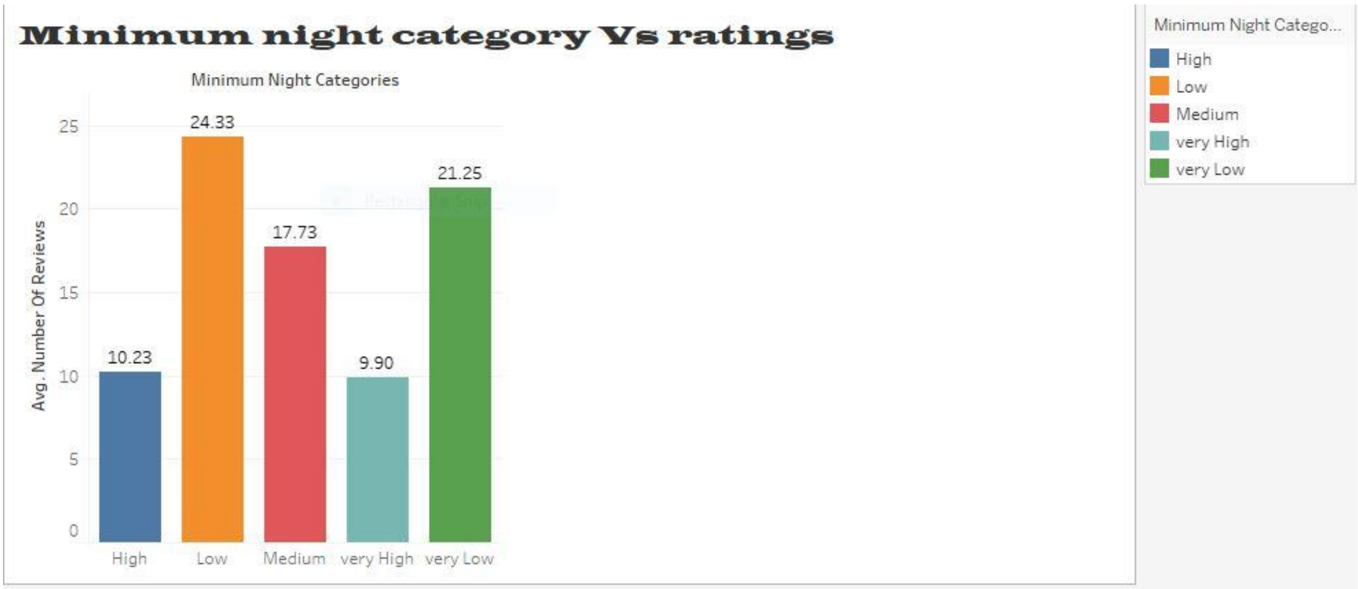
- Allerton
- Arden Heights
- Arrochar
- Arverne
- Astoria
- Bath Beach
- Battery Park City
- Bay Ridge
- Bay Terrace
- Bay Terrace, Stat..
- Baychester
- Bayside
- Bayswater
- Bedford-Stuyves..
- Belle Harbor
- Bellerose
- Belmont



Distribution of location in different neighbourhood

Neighbourhood Group

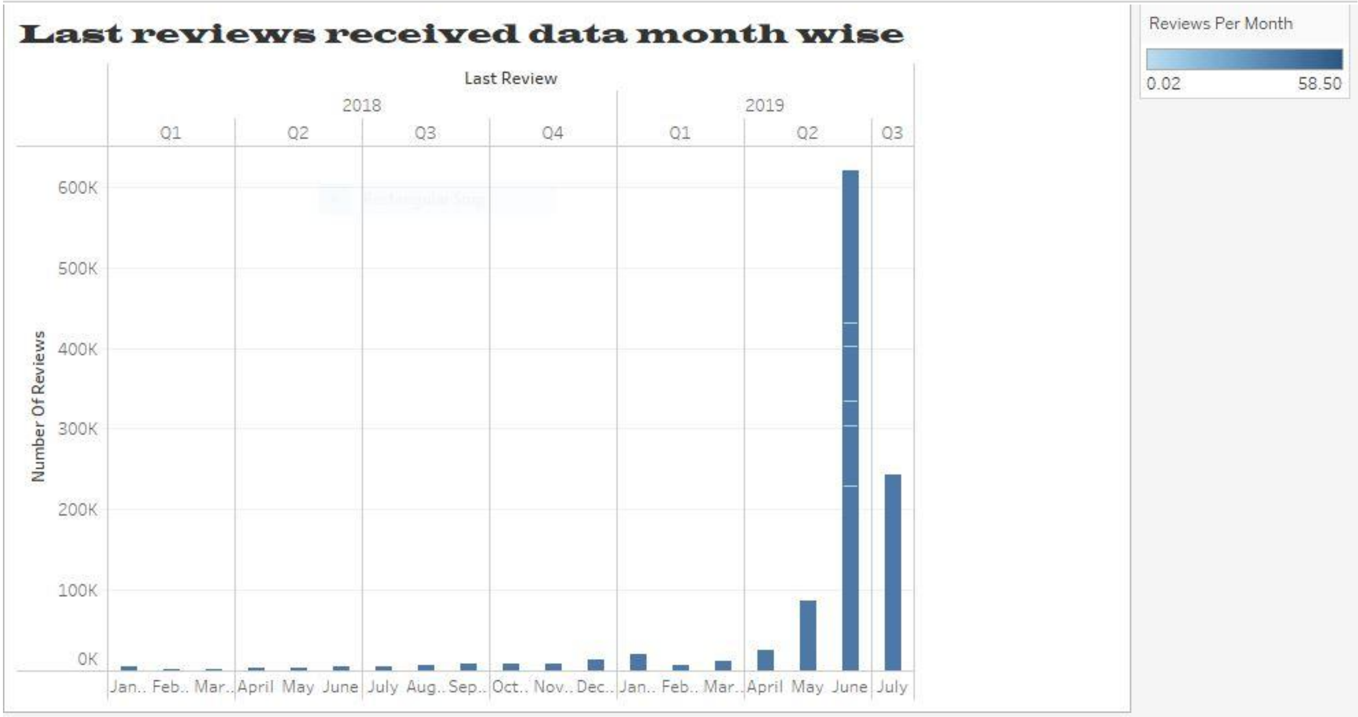
- 🟦 Bronx
- 🟧 Brooklyn
- 🟥 Manhattan
- 🟦 Queens
- 🟩 Staten Island



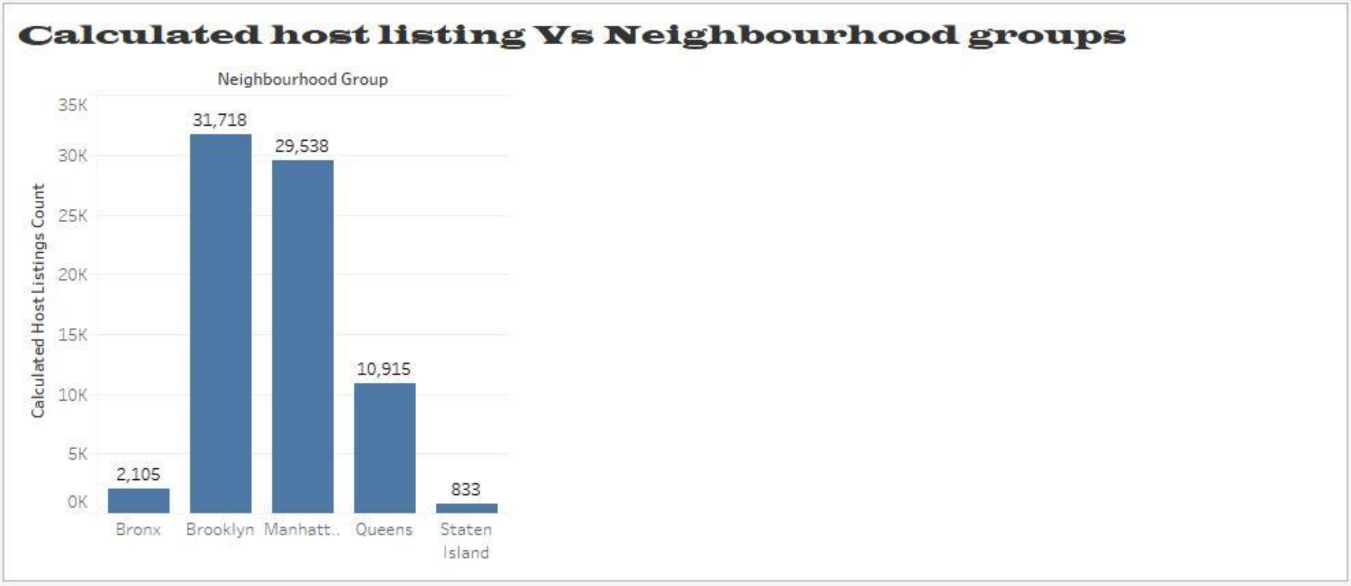
Minimum night category Vs ratings

Caption

Average of Number Of Reviews for each Minimum Night Categories. Color shows details about Minimum Night Categories.

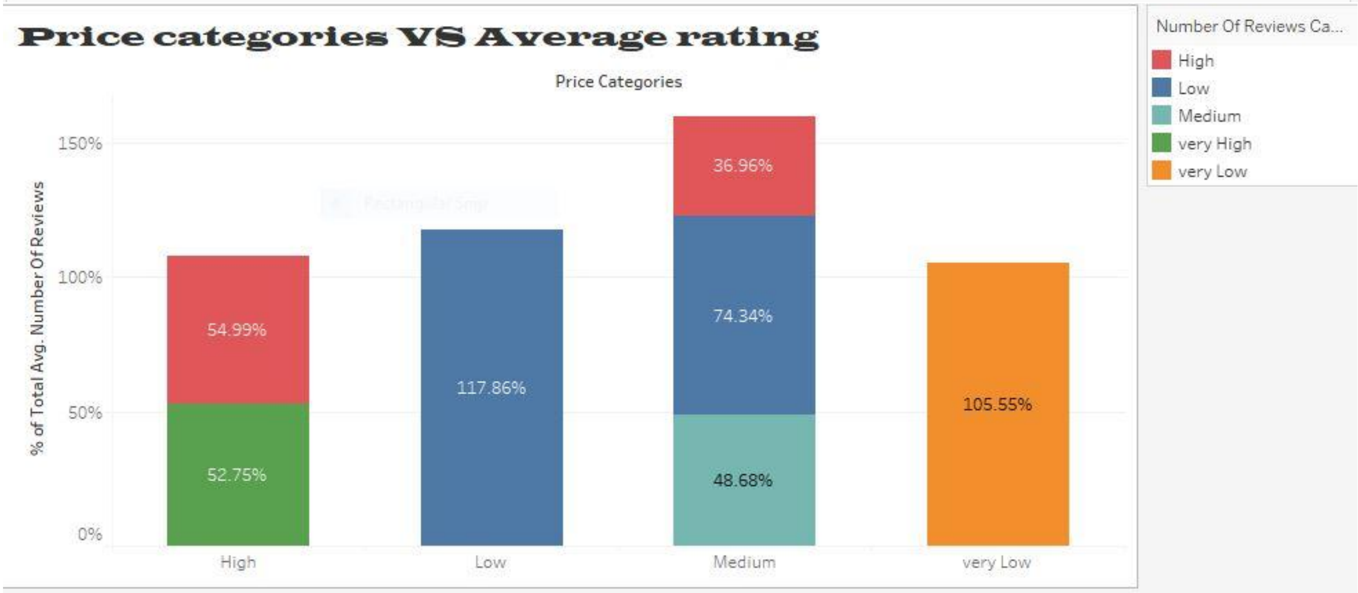


Last reviews received data month wise



Caption

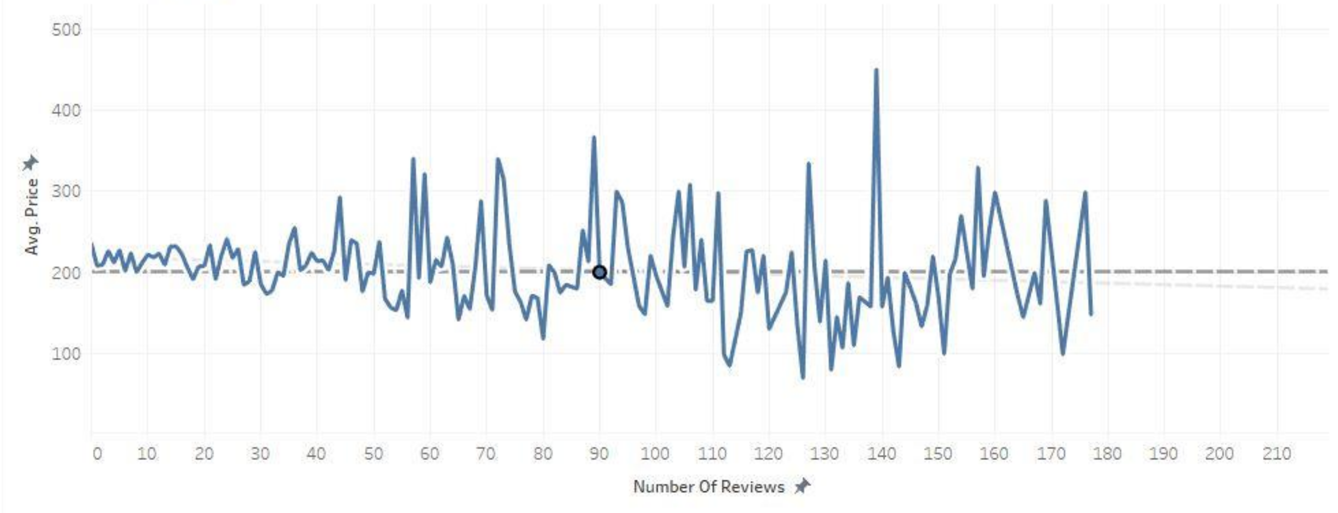
Sum of Calculated Host Listings Count for each Neighbourhood Group.



Caption

Percentile of Avg. Number Of Reviews for each Price Categories. Color shows details about Number Of Reviews Categories.

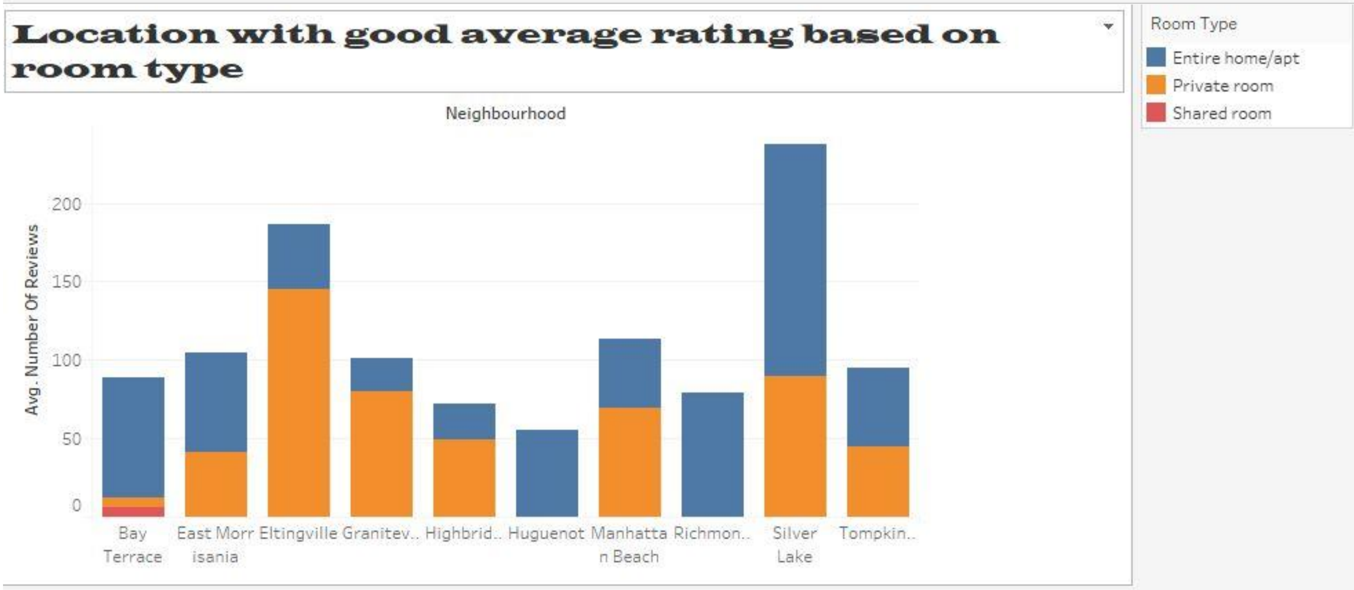
Average price vs Number of Reviews



Caption

The trend of average of Price for Number Of Reviews. The data is filtered on Neighbourhood, which keeps 10 of 219 members.

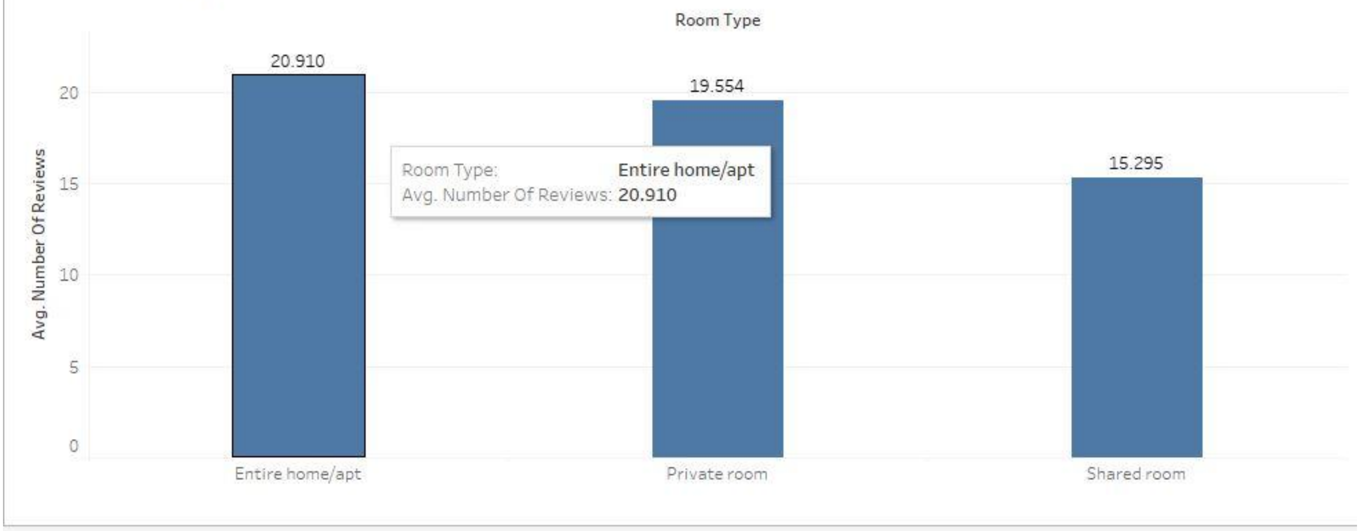
Location with good average rating based on room type



Caption

Average of Number Of Reviews for each Neighbourhood. Color shows details about Room Type. The view is filtered on Neighbourhood, which keeps 10 of 219 members.

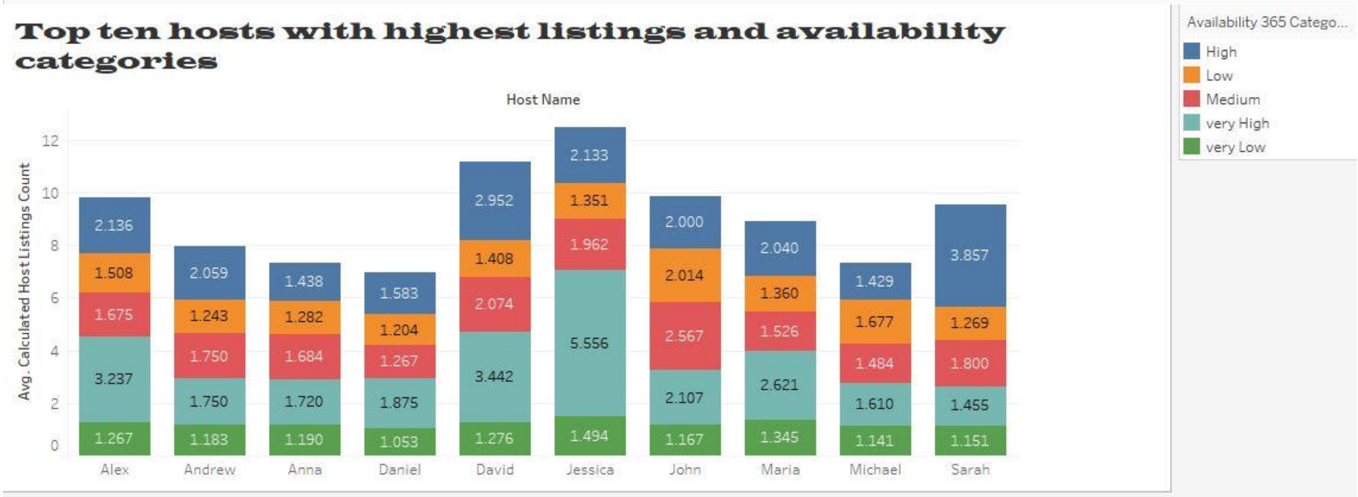
Room type VS Average ratings



Caption

Average of Number Of Reviews for each Room Type.

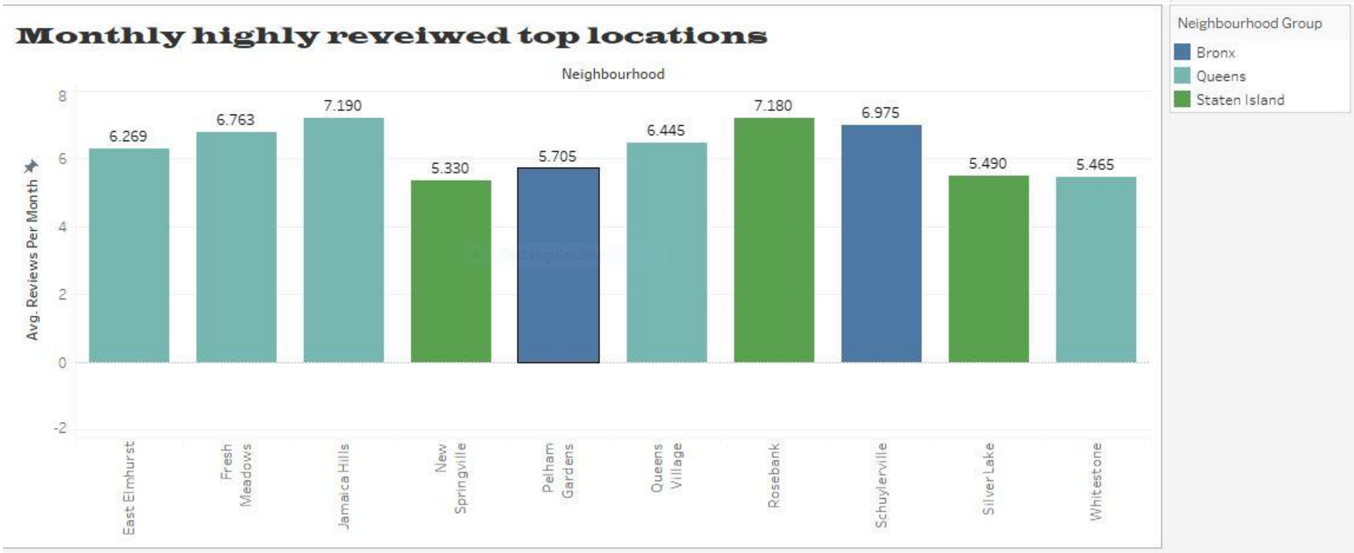
Top ten hosts with highest listings and availability categories



Caption

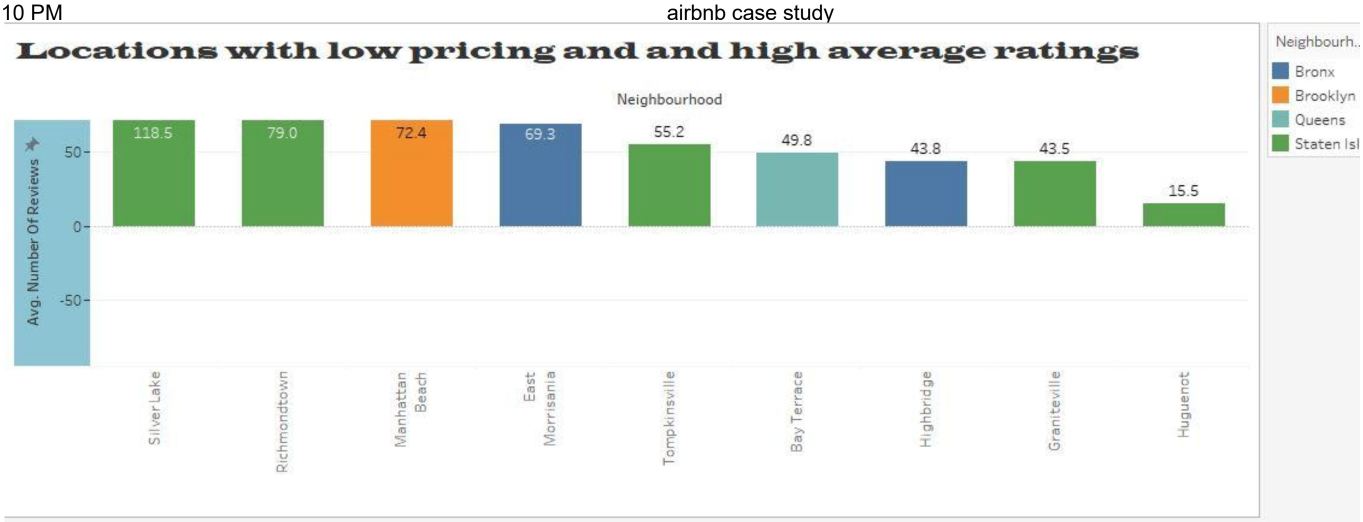
Average of Calculated Host Listings Count for each Host Name. Color shows details about Availability 365 Categories. The view is filtered on Host Name, which keeps 10 of 11,047 members.

Monthly highly reveiwed top locations



Caption

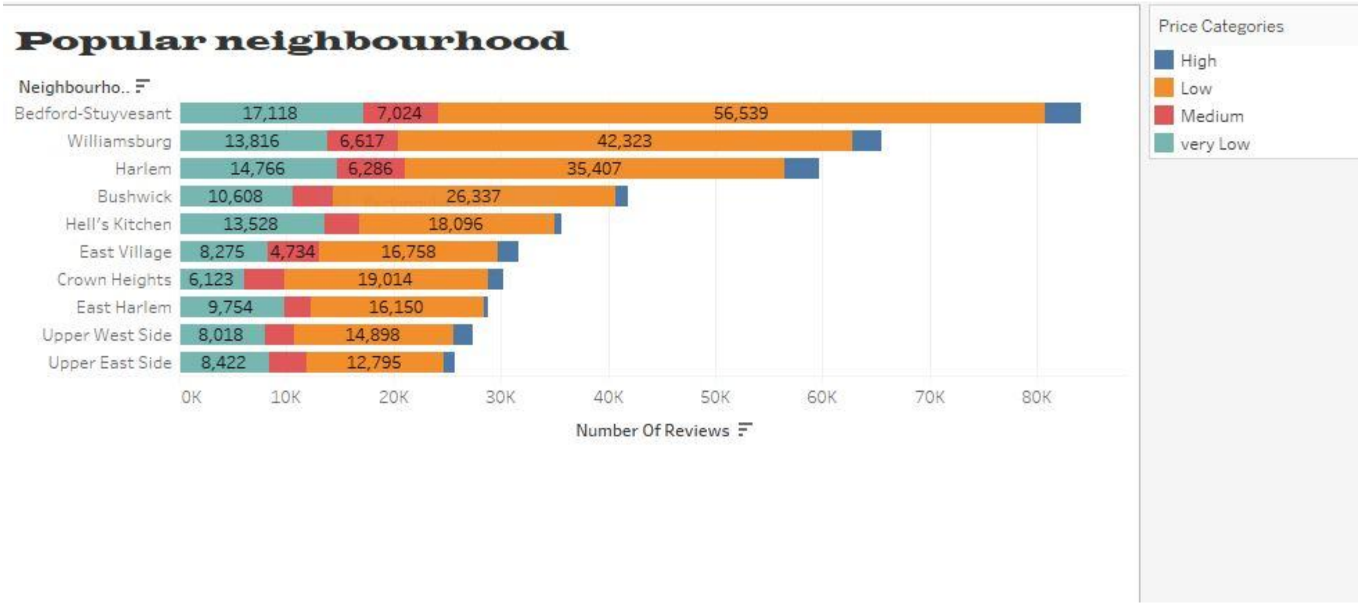
Average of Reviews Per Month for each Neighbourhood. Color shows details about Neighbourhood Group. The data is filtered on Reviews Per Month, which ranges from 5 to 7.72. The view is filtered on Neighbourhood, which keeps 10 of 219 members.



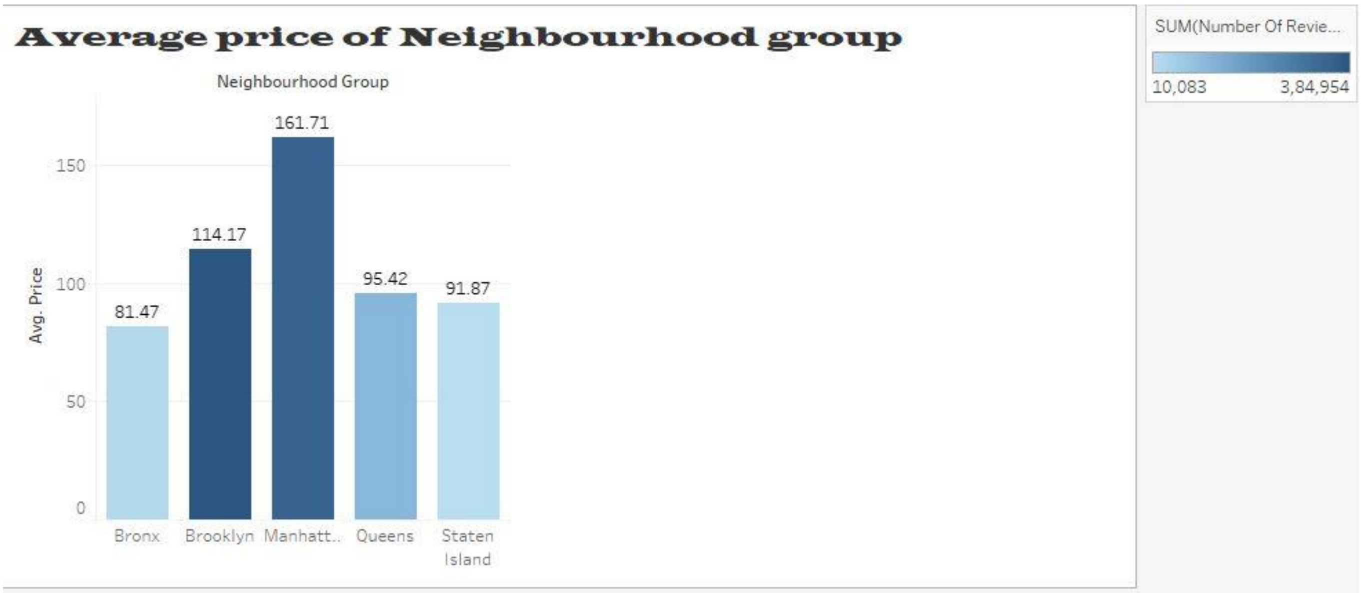
Locations with low pricing and and high average ratings

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.



Popular neighbourhood



Average price of Neighbourhood group

Caption

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