AIR BNB CASE STUDY-METHODOLOGY

Airbnb Case Study

Objective

To prepare for the next best steps that Airbnb needs to take as a business, we have been asked to analyse a dataset consisting of various Airbnb listings in New York.

Problem Statement

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.

Tools Used

For the analysis we have used following tools

• Python Jupiter Notebook

• Tableau

• Microsoft Excel

Mainly to perform Data Cleaning and Data Analysis to come up with useful insights and business recommendations.

Derived/Calculated fields in Tableau

• Number of hosts: count [host Name]

• Number of properties listed: count [Neighbourhood Group]

• Revenue per stay: to check the revenue generated through each booking by multiplying min. of nights with price

Let’s dive into the details and approach we have used in step-wise manner:

1. **Importing the data into Pandas Data frame**

1. Importing libraries and reading the data

```
#Importing Libraries
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Reading the data
nyc = pd.read_csv('AB_NYC_2019.csv')
nyc.head()
```

To check the number of rows and columns present in the data.

```
nyc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 19 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
dtypes: float64(3), int64(7), object(9)
memory usage: 7.1+ MB
```

2. **Categorization of availability_365, minimum_nights and number_of_reviews columns:**

```python
def availability_365_cat(n):
    if n <= 1:
        return 'very Low'
    elif n <= 100:
        return 'Low'
    elif n <= 200 :
        return 'Medium'
    elif (n <= 300):
        return 'High'
    else:
        return 'very High'
```

```python
def minimum_night_cat(n):
    if n <= 1:
        return 'very Low'
    elif n <= 3:
        return 'Low'
    elif n <= 5 :
        return 'Medium'
    elif (n <= 7):
        return 'High'
    else:
        return 'very High'
```

```python
def number_of_reviews_cat(n):
    if n <= 1:
        return 'very Low'
    elif n <= 5:
        return 'Low'
    elif n <= 10 :
        return 'Medium'
    elif (n <= 30):
        return 'High'
    else:
        return 'very High'
```

3. Fixing Columns

NOTE: last_review should be date type instead of object

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

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_40692\804760938.py:
t=False (the default) was specified. This may lead to inconsiste
g.
  nyc.last_review = pd.to_datetime(nyc.last_review)
```

```
0        2018-10-19
1        2019-05-21
2               NaT
3        2019-05-07
4        2018-11-19
            ...
48890           NaT
48891           NaT
48892           NaT
48893           NaT
48894           NaT
Name: last_review, Length: 48895, dtype: datetime64[ns]
```

Changed the last_review column to date type.

4. **Data Types**

4.1 Categorical

```
nyc.columns
```

```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365', 'availability_365_categories',
       'minimum_night_categories', 'number_of_reviews_categories'],
      dtype='object')
```

```
# Categorical nominal
categorical_columns = nyc.columns[[1,3,4,5,8,16,17,18]]
categorical_columns
```

```
Index(['name', 'host_name', 'neighbourhood_group', 'neighbourhood',
       'room_type', 'availability_365_categories', 'minimum_night_categories',
       'number_of_reviews_categories'],
      dtype='object')
```

4.2 Numerical

```
numerical_columns = nyc.columns[[0,2,9,10,11,13,14,15]]
numerical_columns
```

```
Index(['id', 'host_id', 'price', 'minimum_nights', 'number_of_reviews',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365'],
      dtype='object')
```

4.3 Coordinates and date

```
cdate = nyc.columns[[6,12]]
nyc[cdate]
```

	latitude	last_review
0	40.64749	2018-10-19
1	40.75362	2019-05-21
2	40.80902	NaT
3	40.68514	2019-05-07
4	40.79851	2018-11-19
...	...	...
48890	40.67853	NaT
48891	40.70184	NaT
48892	40.81475	NaT
48893	40.75751	NaT
48894	40.76404	NaT

48895 rows × 2 columns

5. Missing Values

- Missing values Next checking the null value count and percentage of null values in those column

```
# To see the number of missing values
nyc.isnull().sum()
```

```
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
dtype: int64
```

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.

- If the analysis is primarily for storytelling and no predictive model is being created, imputing missing values may not be necessary. In such cases, the missing data itself can tell an important story, such as:

1)Why are higher-priced listings less reviews? 2)Why are shared rooms getting fewer reviews?

-Imputing values in this scenario might obscure these insights. Instead, you could focus on explaining the reasons behind the missing data and what it indicates about customer behavior. Highlighting the missing data can provide valuable context for decision-making rather than trying to fill it in.

6. Analysis of Missing Values:

```
nyc1 = nyc.loc[nyc.last_review.isnull(),:]
nyc1
```

7. Missing values Analysis ('neighbourhood_group' feature)

```
# Count of 'neighbourhood_group'with  missing values
nyc1.groupby('neighbourhood_group').neighbourhood_group.count()
```

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

```
# Count of 'neighbourhood_group'
nyc.groupby('neighbourhood_group').neighbourhood_group.count()
```

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

```
(nyc1.groupby('neighbourhood_group').neighbourhood_group.count()/nyc.groupby('neighbourhood_group').neighbourhood_group.count())
```

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

```
((nyc1.groupby('neighbourhood_group').neighbourhood_group.count()/nyc.groupby('neighbourhood_group').neighbourhood_group.count()
```

```
((nyc1.groupby('neighbourhood_group').neighbourhood_group.count()/nyc.groupby('neighbourhood_group').neighbourhood_group.count()
```

```
19.240898461107257
```

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

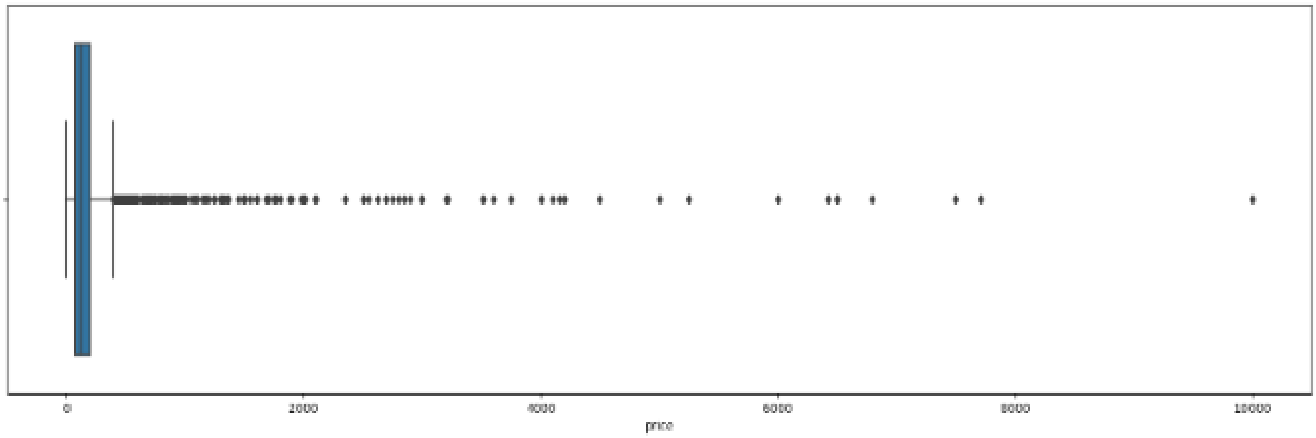
5.3 Missing values Analysis ('room_type' feature)

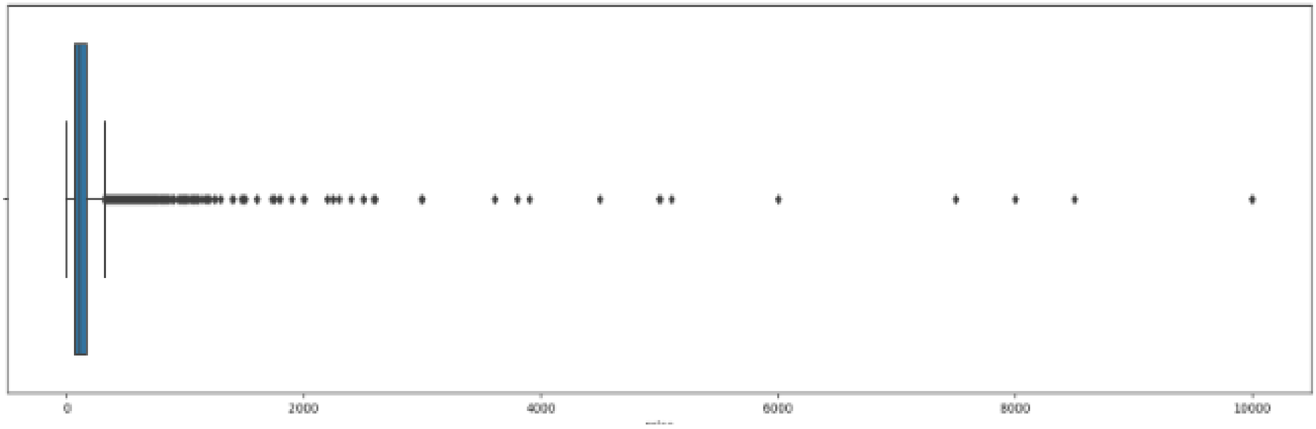
```
# Count of 'room_type' with missing values
nyc3 = (nyc1.groupby('room_type').room_type.count()/nyc.groupby('room_type').room_type.count())*100
nyc3
```

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

```
plt.title('Missing values')
sns.barplot(x = nyc3.index, y = nyc3.values)
plt.show()
```

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





-Higher prices are linked to missing last_review values, indicating that high-priced listings are less likely to receive reviews.

-Shared rooms tend to have fewer reviews, which contributes to the missing last_review data for these room types.

-As prices increase, the likelihood of receiving reviews decreases, possibly due to fewer bookings or higher customer expectations.

-The missing values in last_review are not random but influenced by factors like price and room type.

-This suggests the missing data is not MAR (Missing at Random), where missingness depends on observable features.

-Airbnb could encourage more reviews from high-priced properties and shared room listings to better capture customer feedback.

6. Univariate Analysis

6.1 name

```
nyc.name.value_counts()
```

```
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
```

6.2 host_id

```
nyc.host_id.value_counts()
```

```
219517861    327
107434423    232
30283594     121
137358866    103
16098958      96
            ...
23727216       1
89211125       1
19928013       1
1017772        1
68119814       1
Name: host_id, Length: 37457, dtype: int64
```

6.3 host_name

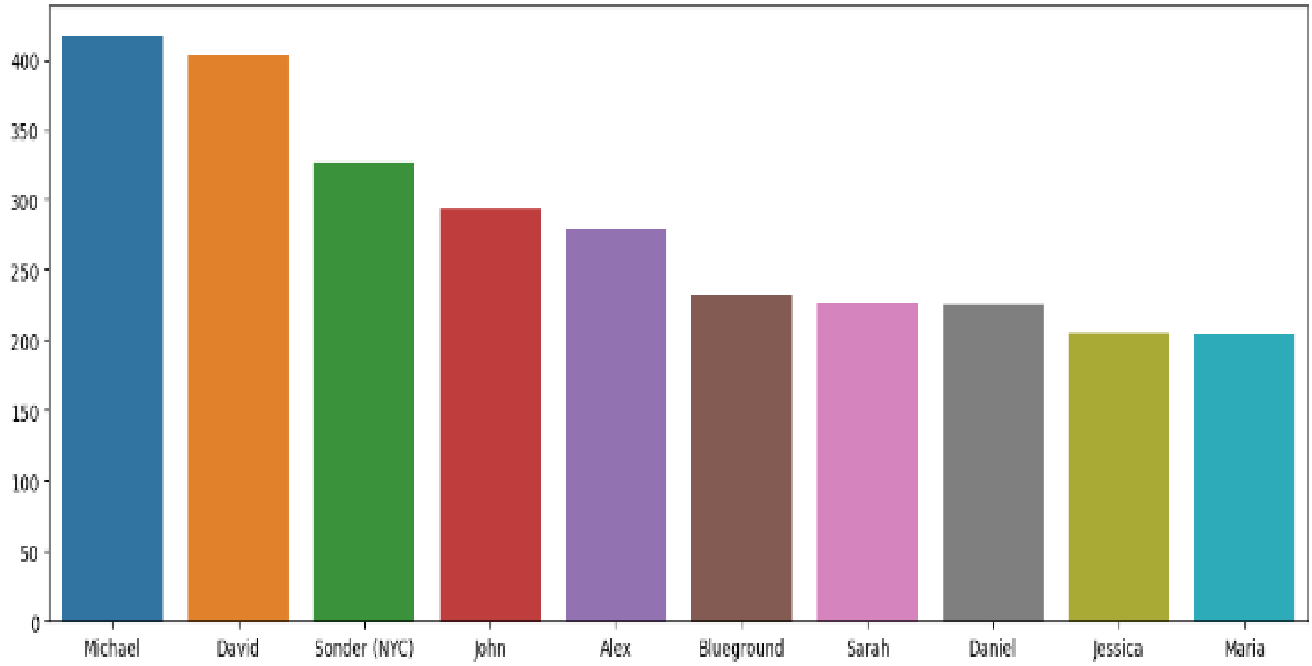
```
nyc.host_name.value_counts()
```

```
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
```

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

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

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



6.4 neighbourhood_group

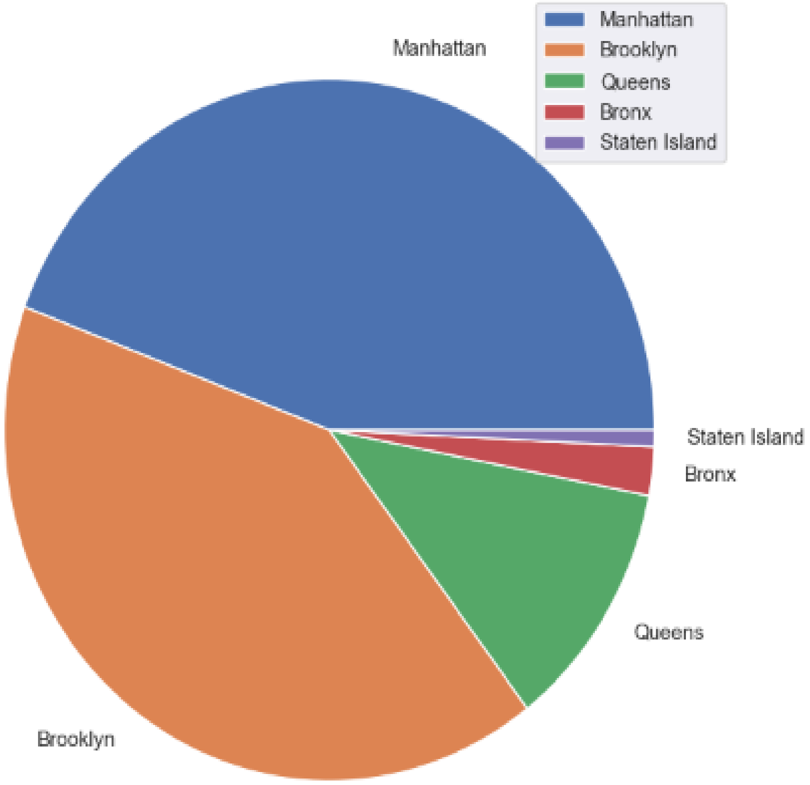
```
nyc.neighbourhood_group.value_counts()
```

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

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

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

```
plt.figure(figsize=(8,8))
plt.pie(x = nyc.neighbourhood_group.value_counts(normalize= True) * 100,labels = nyc.neighbourhood_group.value_counts(normalize=
plt.legend()
plt.show()
```



6.5 neighbourhood ¶

```
nyc.neighbourhood.value_counts()
```

```
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
```

6.6 room_type

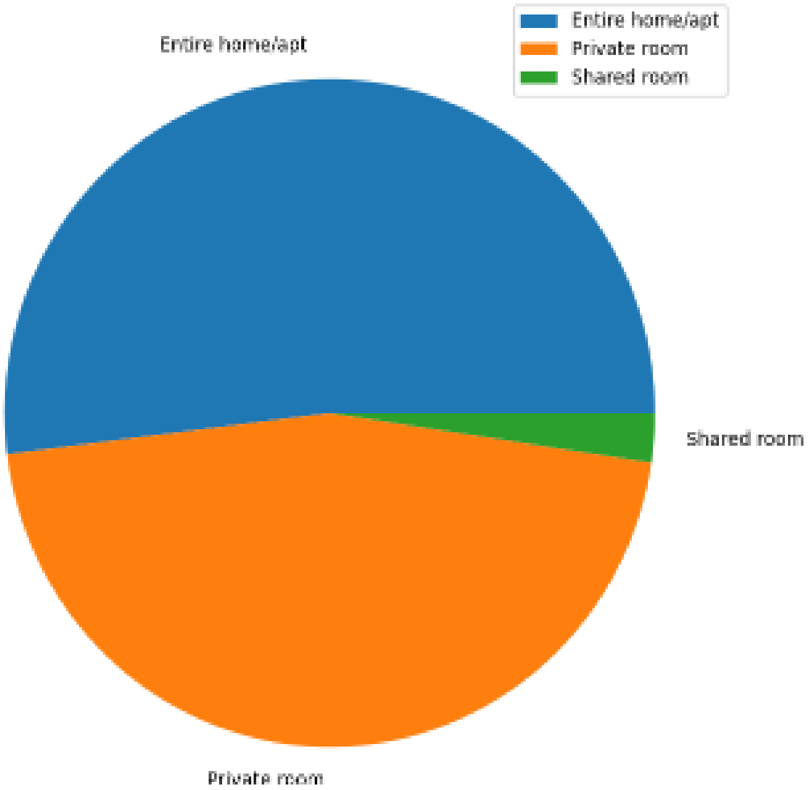
```
nyc.room_type.value_counts()
```

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

```
nyc.room_type.value_counts(normalize=True)*100
```

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

```
plt.figure(figsize=(8,8))
plt.pie(x = nyc.room_type.value_counts(normalize= True) * 100,labels = nyc.room_type.value_counts(normalize= True).index)
plt.legend()
plt.show()
```



Private and Entire Home/apt has the maximum contribution, whereas shared only have 2% Contribution.

6.11 calculated_host_listings_count

```
nyc.calculated_host_listings_count.describe()
```

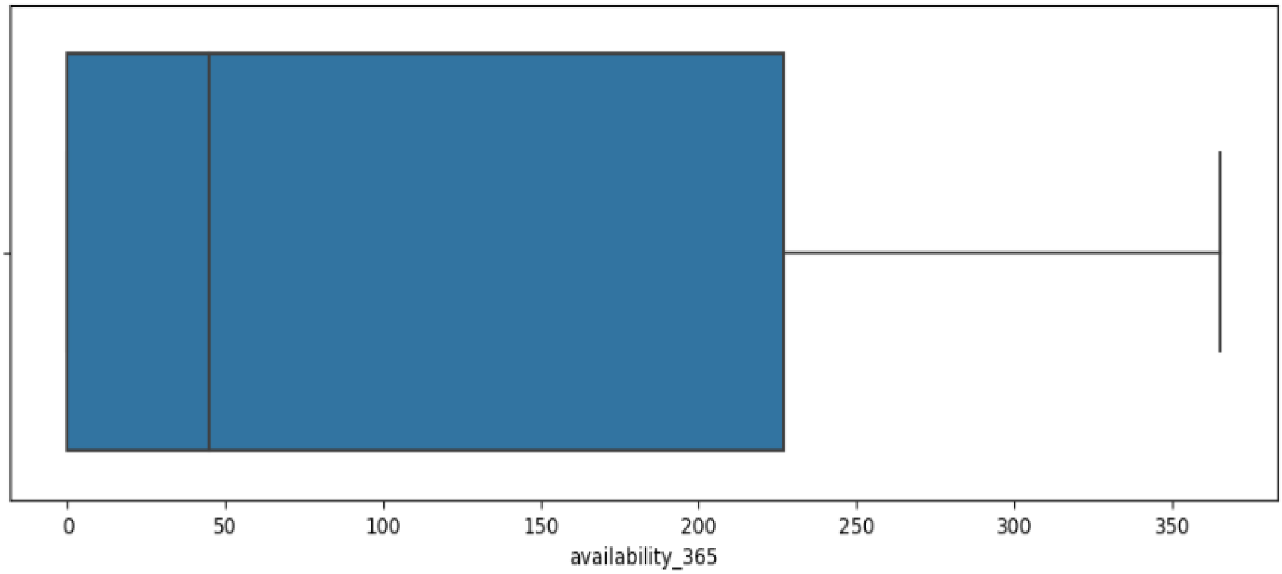
```
count    48895.000000
mean         7.143982
std         32.952519
min          1.000000
25%          1.000000
50%          1.000000
75%          2.000000
max        327.000000
Name: calculated_host_listings_count, dtype: float64
```

6.12 availability_365

```
nyc.availability_365.describe()
```

```
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
```

```
plt.figure(figsize = (12,4))
sns.boxplot(data = nyc, x = 'availability_365')
plt.show()
```

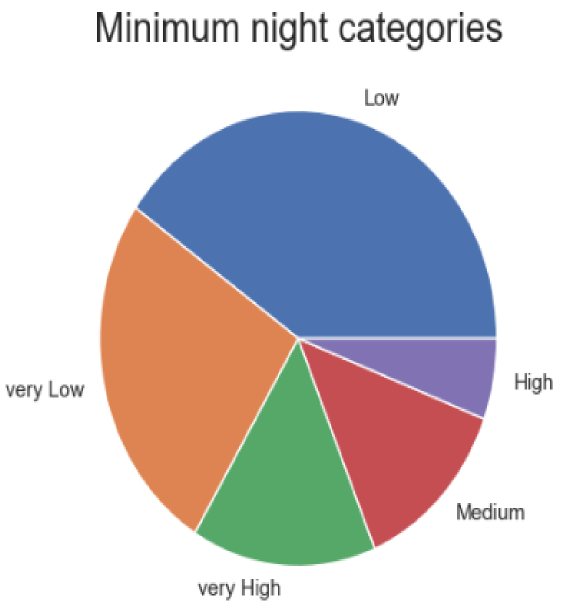


6.13 minimum_night_categories

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

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

```
plt.figure(figsize=(7,5))
plt.title('Minimum night categories', fontdict={'fontsize': 20})
plt.pie(x = nyc.minimum_night_categories.value_counts(),labels=nyc.minimum_night_categories.value_counts().index)
plt.show()
```

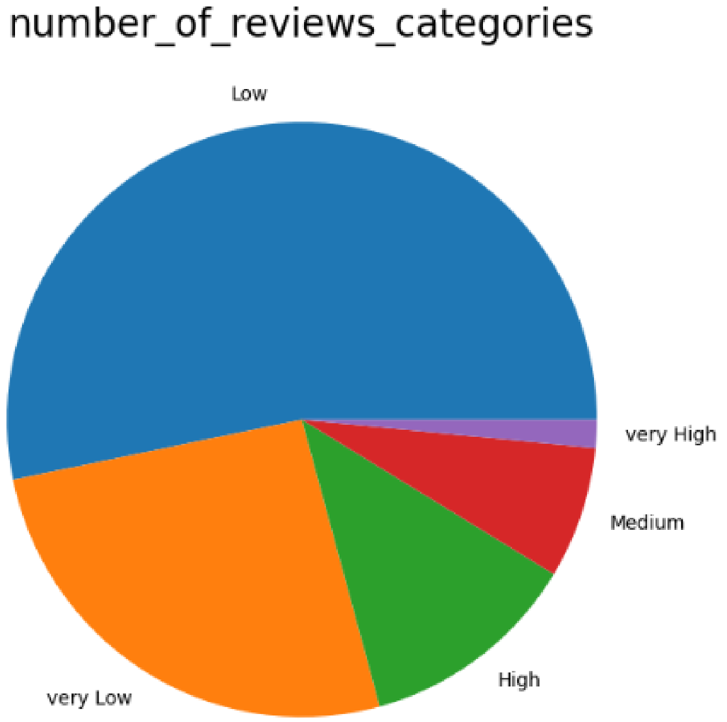


6.14 number_of_reviews_categories

```
nyc.number_of_reviews_categories.value_counts(normalize=True)*100
```

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

```python
plt.figure(figsize=(12,7))
plt.title('number_of_reviews_categories', fontdict={'fontsize': 20})
plt.pie(x = nyc.number_of_reviews_categories.value_counts(),labels=nyc.number_of_reviews_categories.value_counts().index)
plt.show()
```

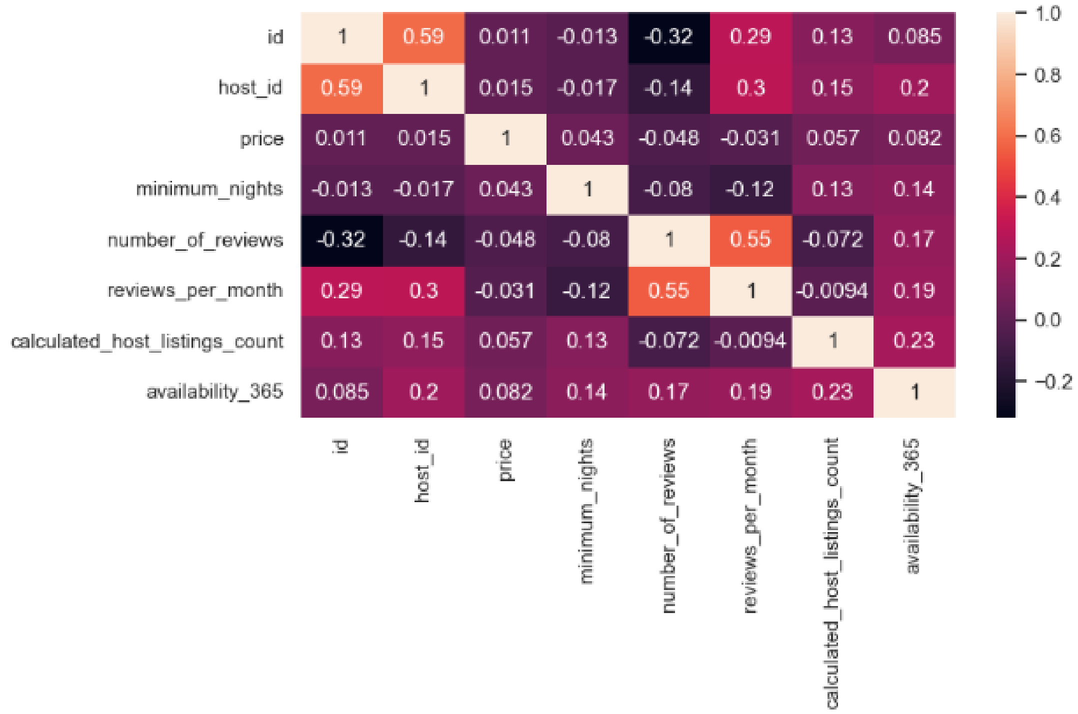


Bivariate and Multivariate Analysis

```python
nyc[numerical_columns].corr()
```

	id	host_id	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availabil
id	1.000000	0.588290	0.010619	-0.013224	-0.319760	0.291828	0.133272	0.
host_id	0.588290	1.000000	0.015309	-0.017364	-0.140106	0.296417	0.154950	0.
price	0.010619	0.015309	1.000000	0.042799	-0.047954	-0.030608	0.057472	0.
minimum_nights	-0.013224	-0.017364	0.042799	1.000000	-0.080116	-0.121702	0.127960	0.
number_of_reviews	-0.319760	-0.140106	-0.047954	-0.080116	1.000000	0.549868	-0.072376	0.
reviews_per_month	0.291828	0.296417	-0.030608	-0.121702	0.549868	1.000000	-0.009421	0.
calculated_host_listings_count	0.133272	0.154950	0.057472	0.127960	-0.072376	-0.009421	1.000000	0.
availability_365	0.085468	0.203492	0.081829	0.144303	0.172028	0.185791	0.225701	1.

```
plt.figure(figsize=(8,4))
sns.heatmap(data = nyc[numerical_columns].corr(),annot=True)
plt.show()
```



From our analysis we have observed that there is a negative correlation between price, minimum nights and number of reviews. And on the other hand, we can see there is a positive correlation between Calculated_host_listings_count and minimum_nights & availability _365 columns

7.2 Finding Top correlations

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

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

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

```
corr_matrix
```

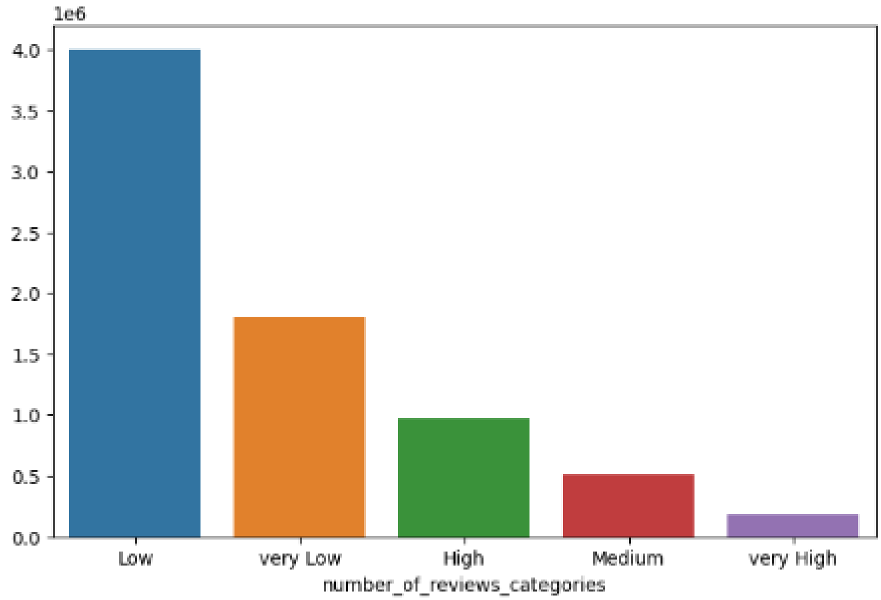
	id	host_id	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability
id	1.000000	0.588290	0.010619	0.013224	0.319760	0.291828	0.133272	0.0
host_id	0.588290	1.000000	0.015309	0.017364	0.140106	0.296417	0.154950	0.2
price	0.010619	0.015309	1.000000	0.042799	0.047954	0.030608	0.057472	0.0
minimum_nights	0.013224	0.017364	0.042799	1.000000	0.080116	0.121702	0.127960	0.1
number_of_reviews	0.319760	0.140106	0.047954	0.080116	1.000000	0.549868	0.072376	0.1
reviews_per_month	0.291828	0.296417	0.030608	0.121702	0.549868	1.000000	0.009421	0.1
calculated_host_listings_count	0.133272	0.154950	0.057472	0.127960	0.072376	0.009421	1.000000	0.2
availability_365	0.085468	0.203492	0.081829	0.144303	0.172028	0.185791	0.225701	1.0

7.3 number_of_reviews_categories and prices

```
# prices for each of reviews_categories
x1 = nyc.groupby('number_of_reviews_categories').price.sum().sort_values(ascending = False)
x1
```

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

```
plt.figure(figsize=(8,5))
sns.barplot(x = x1.index,y = x1.values)
plt.show()
```



7.4 ('room_type' and 'number_of_reviews_categories')¶

```
nyc.room_type.value_counts()
```

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

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

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

Entire home/apt properties receive significantly more reviews compared to shared rooms, indicating higher customer engagement.

Shared rooms account for only 16% of total reviews, showing that guests in shared spaces are less likely to leave feedback.

This suggests that customer preference leans towards entire home/apt listings, which are perceived as more desirable and receive more reviews.

7.5'room_type' and 'reviews_per_month'

```
nyc.room_type.value_counts()
```

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

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

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

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

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

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

7.6 minimum_night_categories and reviews_per_month

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

```
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
```

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'

```
nyc.availability_365_categories.value_counts()
```

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

availability_365_categories	price_categories	reviews_per_month
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

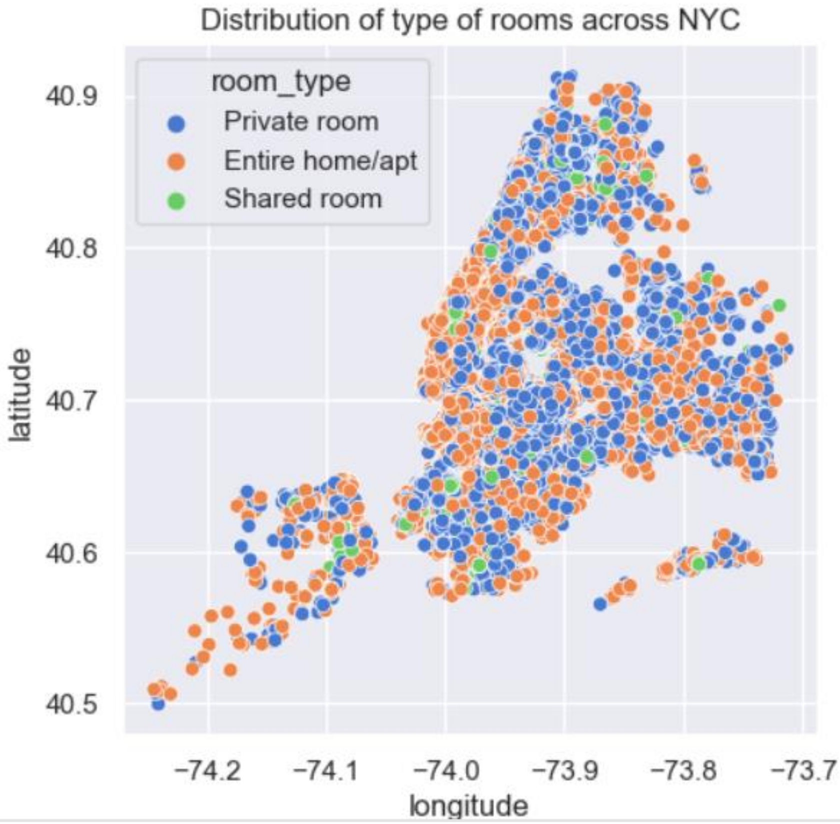
If the combination of availability and price is very high, reviews_per_month will be low on average.

Very high availability and very low price are likely to get more reviews.

7.9 Coordinates with respect to Room type and Neighbourhood groups

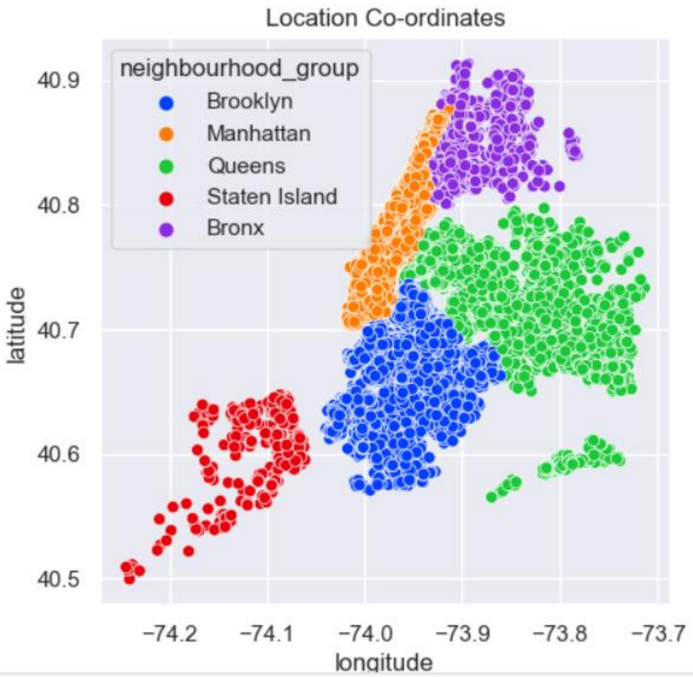
```
#Room Types distribution geographically
sns.set(rc={"figure.figsize": (5, 5)})
ax= sns.scatterplot(x=nyc.longitude, y=nyc.latitude,hue=nyc.room_type,palette='muted')
ax.set_title('Distribution of type of rooms across NYC')
```

```
Text(0.5, 1.0, 'Distribution of type of rooms across NYC')
```



```
#trying to find where the coordinates belong from the latitude and longitude
sns.set(rc={"figure.figsize": (5,5)})
ax= sns.scatterplot(data=nyc, x="longitude", y="latitude",hue='neighbourhood_group',palette='bright')
ax.set_title('Location Co-ordinates')
```

```
Text(0.5, 1.0, 'Location Co-ordinates')
```



By the two scatterplots of latitude vs longitude we can infer there's is very less shared room throughout NYC as compared to private and Entire home/apt.

95% of the listings on Airbnb are either Private room or Entire/home apt. Very few guests had opted for shared rooms on Airbnb.

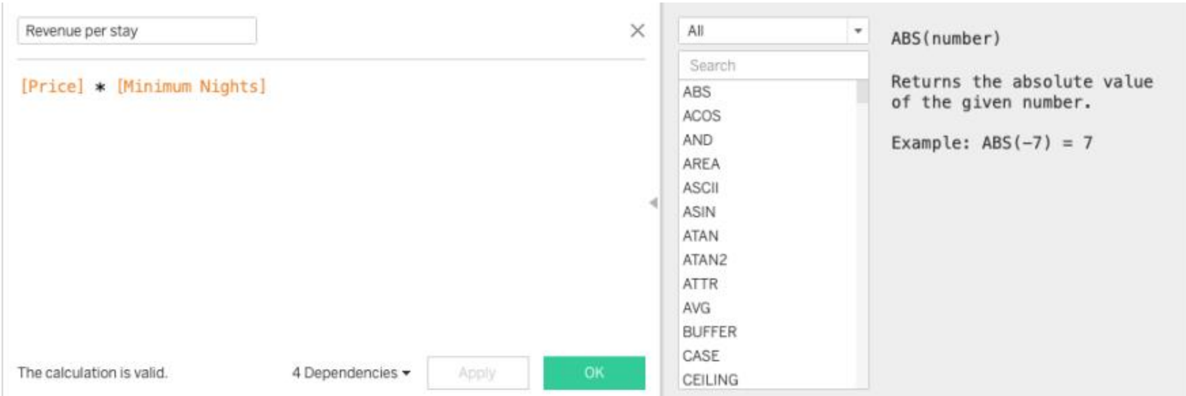
Also, guests mostly prefer this room types when they are looking for a rent on Airbnb as we found out previously in our analysis.

We can infer that there are high range of prices across Manhattan being the most costliest place to stay in NYC

Data Visualization and Analysis using Tableau:

We have used tableau to visualize the data for the assignment. We will use Tableau for Data Visualization and Analysis to come up with Insights and observations. Recommendations are made from the insights and observations drawn from the Analysis.

Derived Column Calculation Revenue Per Stay:



Count of Neighbourhood Group and host names:

Describe Field

Number of properties listed

Role: Continuous Measure
Type: Calculated Field
Status: Valid

Formula

```
count([Neighbourhood Group])
```

Domain

The single value 48,895.

Describe Field

Hosts

Role: Continuous Measure
Type: Calculated Field
Status: Valid

Formula

```
count([Host Name])
```

Domain

The single value 48,874.

Analysis & Insights:

We have created visualizations for each separately and then created two dashboards:

Analysis of each are given below:

1. **Properties based on Neighbourhood Group and Room Type:**

Observation: About 96% of the properties falls under Entire Home/apt. category. Private rooms are the second largest category and very few properties are listed under Shared room across all the Neighbourhood

2. **Host having Maximum Properties**

Observation/Insights: Based upon the number of reviews received we have identified top 20 most preferred host. Micheael has received the maximum number of reviews indicating that he provided good stay experience to the visitors

3. **Maximum Properties listed Neighbourhood Group**

Observation/Insights: 85% of listings are Manhattan and Brooklyn Neighbourhood Groups

The low number of listings in Staten island but high prices indicates an untapped market.

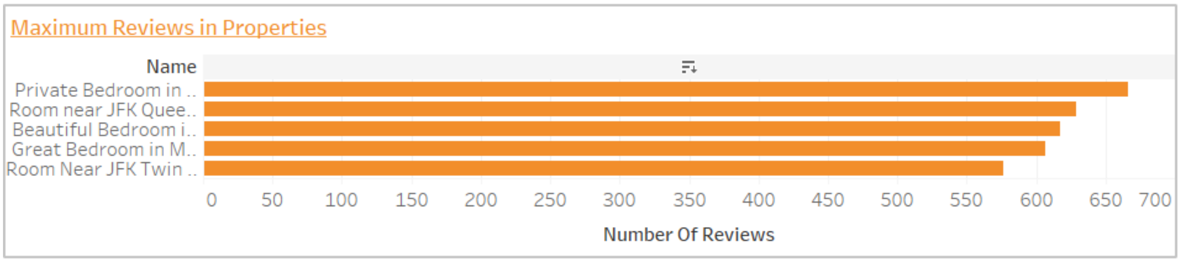
4. **Host Having maximum Properties in Manhattan**

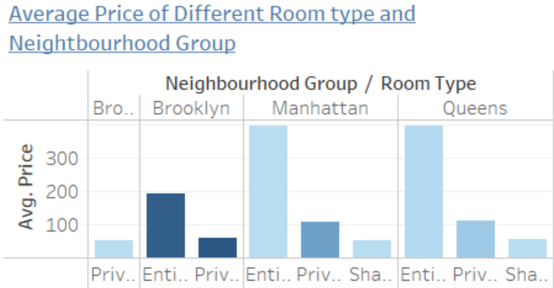
Observation: Sonder has the highest number of properties in Manhattan.

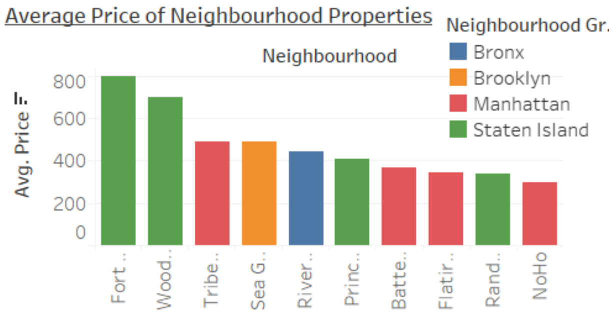
Maximum Number of reviews is provided for either entire home/apt or Private rooms in Manhattan and Brooklyn

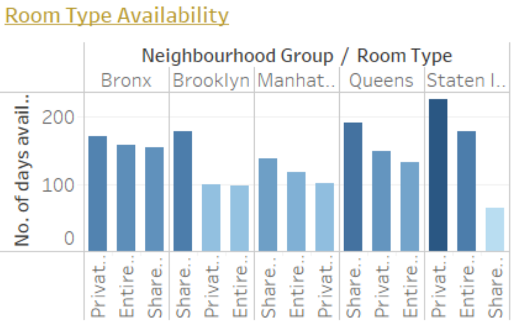
5. **Average Prices of Neighbourhood Groups**

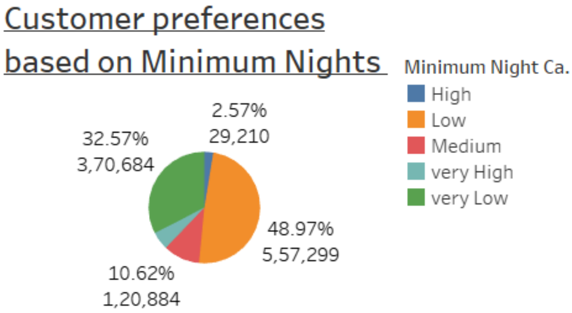
Observations: We can observe that the avg rate of properties in Manhattan across all category is higher compared to other neighbourhoods











Analysis of above charts:

1) High Price listings have lower number of reviews and minimum nights is low/very low provided by hosts.

2) Availability is low in Brooklyn and Manhattan, making it customer preferrable
3) It is possible that many visitor or people are interested in staying for less number of nights. So by reducing the minimum nights required for booking for most of the property in order to increase the revenue. Assuming that, we can restart the service with less minimum nights required in order to increase the visitor and revenue
4) We have identified the Top 10 properties based upon the number of reviews received by them. And found Private Room in Manhattan is the most preferred property for stay among the visitors
5) Maximum Properties are for Minimum night stay 1,2 and 3 days and quite few property offer minimum nights required for booking more than 4 nights for all the neighbourhood groups and Room Type

6) The average price for Entire Home/Apt show the similar trend across the minimum nights required for booking. However, Private room category has almost equivalent average price for long duration stay (>30 days)The hike in the revenue is for Minimum stays 1 to 5 days , mainly 1, 2 and 3 night stay and 30 days. Business can think about the properties having min night stays in these categories across all locations.

Recommendations to Data analysis Managers and lead data analysts:

1. **Segment Market Trends**: Based on above analysis on post-COVID trends to identify shifts in traveller preferences and booking patterns. Focus on the recovery rate of luxury vs. budget accommodations.
2. **Dynamic Pricing Strategies**: Implement machine learning algorithms to optimize pricing based on demand fluctuations, special events, and local economic conditions.
3. **Promote Long-term Stays**: Increase marketing efforts for long-term stay options, as there's been a rise in remote working and extended vacations.
4. **Enhance Data Collection**: Improve granularity of data collection to better track customer behavior, booking frequency, and length of stay. Integrate data from customer feedback and reviews.
5. **High Missing values in "last_Review " column:**Take customer feedback before the check out so that the facilities can be improved in case of customer dissatisfaction.
6. **Competitive Benchmarking**: Compare performance with key competitors in the market to identify gaps and opportunities for revenue growth.

Recommendations for Head of User Experience and operations

Enhance Value for Money:

- Offer competitive pricing with high-quality facilities.
- Highlight premium amenities in property descriptions and photos.

Optimize Pricing in High-Demand Areas:

- Implement dynamic pricing in Brooklyn and Manhattan.
- Use promotions during off-peak times and adjust rates based on demand

Enhance Private Room Experience:

Improve and promote private rooms as an affordable yet comfortable option, appealing to solo travelers and couples

Expand Shared Room Listings:

Increase the number of shared room options to attract budget-conscious travelers and solo adventurers, filling a gap in the current market

Promote Staten Island's Unique Selling Points:

Highlight Staten Island's attractions, such as scenic views and cultural sites, in marketing materials