**Market Analysis Report.** (Word count: 975)

## 1. Introduction

The board of directors at Airbnb are actively seeking to analyse specific segments of the real estate market for potential investment opportunities. Particularly, they are interested in exploring the feasibility of investing in the construction or acquisition of long-term rental properties, specifically targeting "entire home/apartment" formats, in New York City. In this report, we will address a series of business questions they have posed in order to provide them with comprehensive insights and data-driven analysis. Our goal is to facilitate informed decision-making regarding potential investments.

## 2. Business Questions: Analysis and Findings

*A.* **Overall Market Landscape:** *What is the current economic landscape of the New York City market for long-term rental properties exceeding 30 days or 1 month in the "entire home/apartment" format?*

Out of 44,046 presented listings, 23,156 (52.6%) are classified as "entire home/apartment", and only 478 listings (approximately 2.1%) identified as long-term rental properties.
With 311 listings, Manhattan is the leading neighbourhood group for long-term rental properties, indicating a robust market for entire home/apartment rentals. Brooklyn follows Manhattan with 140 listings, offering a significant alternative to Manhattan. Queens has a smaller market share with only 17 listings, indicating limited availability compared to Manhattan and Brooklyn. The Bronx has just 9 listings, suggesting a less developed market for long-term rentals in the entire home/apartment format. Among the neighbourhood groups, Staten Island has the fewest listings with only 1, indicating a highly limited market.

Visualisation: Appendix 1

*B.* **Neighbourhood Group Analysis:** *Which neighbourhood groups within New York City demonstrate notable popularity and demand for long-term rentals of "entire home/apartment" properties, and to what extent?*

Manhattan (311 listings and 3,538 reviews) stands out as the most popular neighbourhood group, indicating a high demand for long-term rentals in this area. Brooklyn (140 listings and 2,437 reviews) showcasing notable popularity. Queens (17 listings and 299 reviews) indicating a moderate level of popularity. The Bronx (9 listings and 52 reviews) demonstrates a relatively lower level of demand compared to Manhattan and Brooklyn. Finally, Staten Island (1 listing and no reviews) indicating the lowest level of popularity.
To summarise, Manhattan and Brooklyn are the most popular neighbourhood groups in New York City for long-term rentals of "entire home/apartment" properties, while Queens, the Bronx, and Staten Island have a lower level of demand.

Visualisation: Appendix 2

*C.* **Neighbourhood Analysis:** *Which specific neighbourhoods within New York City hold investment potential for the construction or purchase of long-term rental properties in the "entire home/apartment" format? Furthermore, what is the level of desirability for such properties in these neighbourhoods?*

The level of desirability based on their total number of listings and the corresponding number of reviews.
In Manhattan, the neighbourhoods with the highest number of reviews, indicating popularity and demand, are: Harlem: 676 reviews (significant investment potential); Upper West Side: 514 reviews (desirability for long-term rentals); West Village, Chelsea, and East Harlem are

other noteworthy neighbourhoods with substantial numbers of reviews.
In Brooklyn, the neighbourhoods with notable investment potential and desirability for long-term rentals are: Bedford-Stuyvesant: 953 reviews (high demand); Bushwick and Williamsburg are also popular neighbourhoods with significant numbers of reviews.
Queens, despite having a smaller number of listings, has neighbourhoods with moderate levels of desirability based on reviews. The neighbourhoods of Ditmars Steinway and Astoria have the highest number of reviews among the Queens neighbourhoods, suggesting a certain level of investment potential.
The Bronx and Staten Island have a relatively lower number of listings and reviews compared to Manhattan and Brooklyn, indicating a lower level of demand and investment potential.

<u>Visualisation:</u> Appendix 3

D. **Pricing Analysis***: What are the average market prices for "entire home/apartment" long-term rentals within each identified neighbourhood group and specific neighbourhood?*

The average prices for each neighbourhood group and specific neighbourhood:
<u>In Manhattan:</u> Neighbourhoods such as Nolita, Hell's Kitchen, and Greenwich Village have higher average prices, ranging from $294 to $348; Roosevelt Island stands out with an average price of $762.50, indicating a higher-end rental market; other neighbourhoods have relatively lower average prices, ranging from $79.50 to $287.31.
<u>In Brooklyn:</u> Neighbourhoods like Crown Heights, Carroll Gardens, and Windsor Terrace have average prices ranging from $114 to $266; Prospect-Lefferts Gardens and Greenpoint show higher average prices of $568 and $190.60, respectively; Some neighbourhoods have lower average prices, such as Downtown Brooklyn with $97 and Clinton Hill with $89.
<u>In Queens:</u> Ditmars Steinway and Astoria have average prices of $69.50 and $89.50, respectively; Long Island City shows a slightly higher average price of $139, indicating a relatively more expensive rental market.
<u>In The Bronx and Staten Island:</u> The Bronx neighbourhoods have average prices ranging from $54 to $91.50, representing relatively more affordable options; Staten Island's St. George neighbourhood has an average price of $100.

<u>Visualisation:</u> Appendix 3

*E.* **Additional Factors***: Are there any noteworthy findings or factors that should be taken into consideration which may significantly impact investment decisions in this context?*

While we have successfully addressed all the business questions during the exploratory data analysis (EDA), we decided to conduct further investigation to uncover potentially hidden patterns. To achieve this, we employed clustering algorithms and conducted an in-depth analysis of the resulting clusters. The cluster analysis revealed four major clusters among the 478 property listings; however, it solely relied on the geographical division based on neighbourhoods and did not uncover any hidden relationships. Consequently, the obtained results did not provide any additional information that could be utilised in our report.

<u>Visualisation:</u> Appendix 4

## 3. Conclusion and Recommendations

The analysis of the New York City real estate market for long-term rental properties indicates investment potential in the "entire home/apartment" format. Manhattan and Brooklyn are the most popular areas, while Queens, the Bronx, and Staten Island show varying levels of demand. Detailed consideration requires additional data on market trends, rental regulations, and economic indicators. Gathering more data will enhance the accuracy of investment evaluations and inform decision-making.

# 4. Appendices
## Appendix 1.

Total Number of 'Entire home/apt' Rental Properties in New York City: 478

| room_type | neighbourhood_group | Total Number of Listings | Percentage |
|---|---|---|---|
| Entire home/apt | Manhattan | 311 | 65.06 |
| | Brooklyn | 140 | 29.29 |
| | Queens | 17 | 3.56 |
| | Bronx | 9 | 1.88 |
| | Staten Island | 1 | 0.21 |

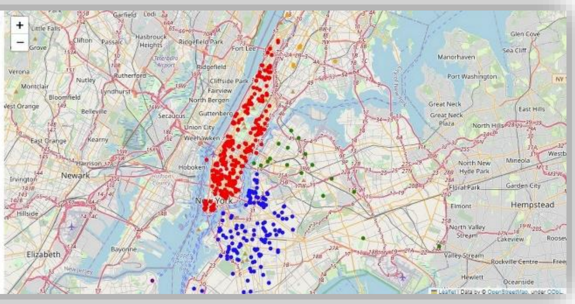

*Figure 1 Total Number of Listings*

## Appendix 2.

Total Number of Popular Neighbourhood Groups in New York City:
Total Number of Listings: 478

| room_type | neighbourhood_group | Total Number of Listings | Total Number of Reviews |
|---|---|---|---|
| Entire home/apt | Manhattan | 311 | 3538 |
| | Brooklyn | 140 | 2437 |
| | Queens | 17 | 299 |
| | Bronx | 9 | 52 |
| | Staten Island | 1 | 0 |

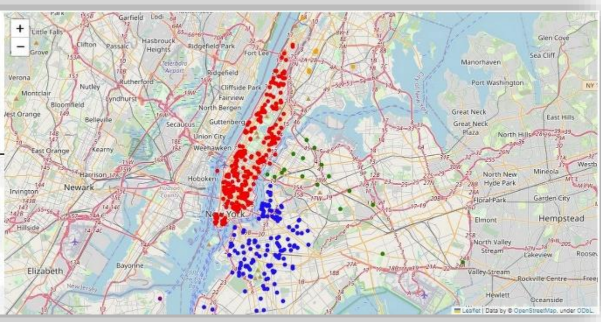

*Figure 2 Total Number of Popular Neighbourhood Groups*

## Appendix 3.

Total Number of 'Entire home/apt' Rental Properties in Manhattan: 311
Neighbourhoods in Manhattan by Number of Reviews (Highest to Lowest):

| neighbourhood_group | neighbourhood | Total Number of Reviews | Mean Price |
|---|---|---|---|
| Manhattan | Harlem | 676 | 236.437500 |
| | Upper West Side | 514 | 287.310345 |
| | West Village | 312 | 188.000000 |
| | Chelsea | 299 | 163.680000 |
| | East Harlem | 243 | 172.600000 |

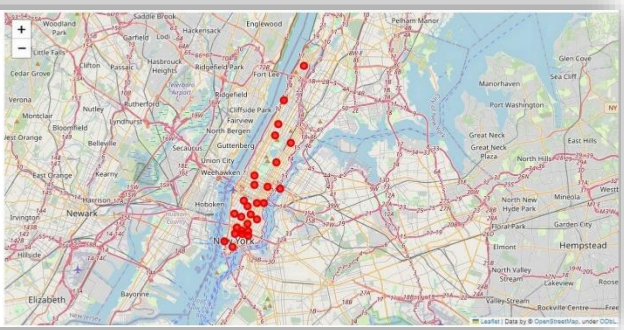

*Figure 3 Manhattan*

Total Number of 'Entire home/apt' Rental Properties in Brooklyn: 140
Neighbourhoods in Brooklyn by Number of Reviews (Highest to Lowest):

| neighbourhood_group | neighbourhood | Total Number of Reviews | Mean Price |
|---|---|---|---|
| Brooklyn | Bedford-Stuyvesant | 953 | 148.470588 |
| | Bushwick | 256 | 128.900000 |
| | Williamsburg | 216 | 177.029412 |
| | Crown Heights | 180 | 266.076923 |
| | Carroll Gardens | 138 | 187.000000 |

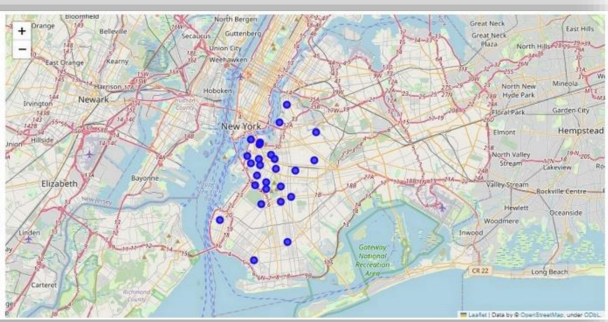

*Figure 4 Brooklyn*

```
Total Number of 'Entire home/apt' Rental Properties in Queens: 17
Neighbourhoods in Queens by Number of Reviews (Highest to Lowest):
```

| neighbourhood_group | neighbourhood | Total Number of Reviews | Mean Price |
|---|---|---|---|
| Queens | Ditmars Steinway | 200 | 69.5 |
| | Astoria | 48 | 89.5 |
| | Long Island City | 32 | 139.0 |
| | Rosedale | 15 | 350.0 |
| | Elmhurst | 2 | 100.0 |

*Figure 5 Queens*

```
Total Number of 'Entire home/apt' Rental Properties in Bronx: 9
Neighbourhoods in Bronx by Number of Reviews (Highest to Lowest):
```

| neighbourhood_group | neighbourhood | Total Number of Reviews | Mean Price |
|---|---|---|---|
| Bronx | Claremont Village | 28 | 91.5 |
| | Fordham | 10 | 71.0 |
| | Mount Hope | 7 | 54.0 |
| | Spuyten Duyvil | 7 | 79.0 |
| | Parkchester | 0 | 70.0 |

*Figure 6 Bronx*

```
Total Number of 'Entire home/apt' Rental Properties in Staten Island: 1
Neighbourhoods in Staten Island by Number of Reviews (Highest to Lowest):
```

| neighbourhood_group | neighbourhood | Total Number of Reviews | Mean Price |
|---|---|---|---|
| Staten Island | St. George | 0 | 100 |

*Figure 7 Staten Island*

## Appendix 4.

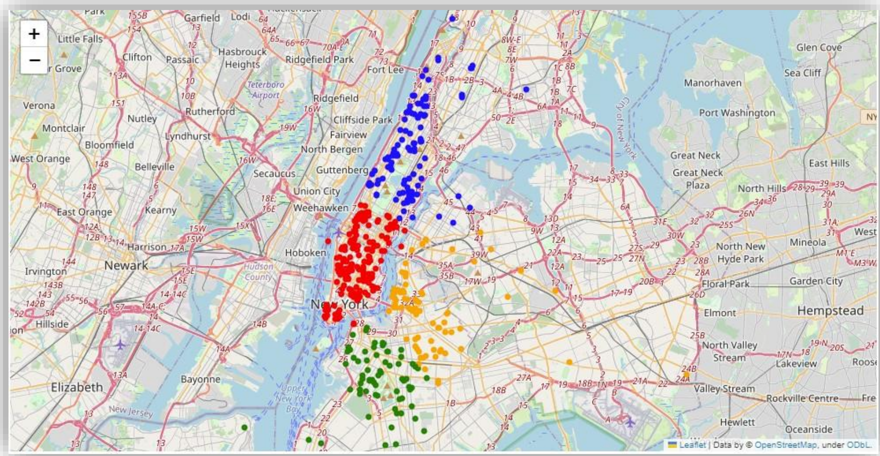

*Figure 8 All clusters*

**Appendix 6.**

We kindly request your attention to the fact that we have included an additional ***ML Assignment 1 AirBnB.ipynb*** file in the University Submission Form, which contains all the necessary code. We appreciate your consideration.

```
#pip install folium
#pip install yellowbrick
import random
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as colors
import folium

sns.set()
import scipy.stats as st
from scipy.stats import norm
from scipy.stats import iqr
%matplotlib inline

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.cluster import KMeans, DBSCAN
from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer

# report warnings
import warnings
warnings.filterwarnings('ignore')

class Settings:
    """Settings is the class for EDA.
    This class has 2 attributes:
    - distplot_attributes
    - boxplot_attributes
    """

    def distplot_attributes(self, data):
        """Plot multiple attributes using distplot"""
        cols = []
        # Iterate over columns in the data
        for i in df.columns:
            # Check if the column data type is float or int
            if df[i].dtypes == "float64" or df[i].dtypes == 'int64':
                cols.append(i)
        # Create a figure for the subplots
        gp = plt.figure(figsize=(15, 10))
        gp.subplots_adjust(wspace=0.2, hspace=0.4)
        # Iterate over the selected columns
        for i in range(1, len(cols) + 1):
            # Add a subplot to the figure
            ax = gp.add_subplot(3, 4, i)
            # Plot the distribution using distplot
            sns.distplot(data[cols[i - 1]], fit=norm, kde=False)
            # Set the title of the subplot
            ax.set_title('{}'.format(cols[i - 1]))

    def boxplot_attributes(self, data):
        """Plot multiple attributes using boxplot"""
        cols = []
        # Iterate over columns in the data
        for i in df.columns:
            # Check if the column data type is float or int
            if df[i].dtypes == "float64" or df[i].dtypes == 'int64':
```

```python
                cols.append(i)
        # Create a figure for the subplots
        gp = plt.figure(figsize=(20, 15))
        gp.subplots_adjust(wspace=0.2, hspace=0.4)
        # Iterate over the selected columns
        for i in range(1, len(cols) + 1):
            # Add a subplot to the figure
            ax = gp.add_subplot(3, 4, i)
            # Plot the boxplot using boxplot
            sns.boxplot(x=cols[i - 1], data=df)
            # Set the title of the subplot
            ax.set_title('{}'.format(cols[i - 1]))

# Read data from 'dataset.csv' file
df = pd.read_csv("AB_NYC_2019.csv")

# Info
df.info()

# Describe
df.describe()

# Head
df.head()

# Check for unique values
df.nunique()
#
# Set 'id' as the index
df.set_index('id', inplace=True)

# Drop unnecessary columns
columns_to_drop = ['name', 'host_id', 'host_name']
df.drop(columns=columns_to_drop, inplace=True)

# Missing values
total =
df.isnull().sum().sort_values(ascending=False)[df.isnull().sum().sort_values(ascend
ing=False) != 0]
percent = round(df.isnull().sum().sort_values(ascending=False) / len(df) * 100,
2)[round(df.isnull().sum().sort_values(ascending=False) / len(df) * 100, 2) != 0]
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data

# Visualize fields with missing values
sns.set_style('darkgrid')
missing = df.isnull().sum()
missing = missing[missing > 0]
missing.sort_values(inplace=True)
missing.plot.bar(color='#5081ac')

# Simplify 'last_review' to year component
df['last_review'] = pd.to_datetime(df['last_review']).dt.year
#
# Calculate number and percentage of '0' values in 'availability_365'
condition = df[df['availability_365'] == 0]
num_rows = len(condition)
percentage = round((num_rows / len(df)) * 100, 2)
zero_data = pd.DataFrame({'Total': [num_rows], 'Percent': [percentage]})
print(zero_data)

# Analyze instances with 'availability_365' = 0 and null values in 'last_review',
'reviews_per_month', and 'number_of_reviews'
zero_null_data = df[(df['availability_365'] == 0) & (df['last_review'].isnull()) &
(df['reviews_per_month'].isnull()) & (df['number_of_reviews'] == 0)]
zero_null_data
```

```python
# It shows that there are a total of 17,533 instances where the attribute
'availability_365' equals 0, which accounts for approximately 35.86% of the
dataset.
#
# Now let's examine how many of the 17,533 instances where 'availability_365'
equals 0 also have null values for the attributes 'last_review' and
'reviews_per_month', and where 'number_of_reviews' equals 0.

condition = df[(df['number_of_reviews'] == 0) & df['last_review'].isnull() &
(df['reviews_per_month'].isnull()) & (df['availability_365'] == 0)]

# Counting the number of rows that satisfy the conditions
num_rows = len(condition)

# Calculating the percentage of filtered rows in the whole dataset
percentage = (num_rows / len(df)) * 100

# Displaying the results
print("Total: ", num_rows)
print("Percent: ", percentage, "%")

# It shows that there are a total of 4845 instances. 9.9% of data can be considered
as missing data. It is less than 10% of the whole dataset. So we can drop it.

# Dropping the rows from the original DataFrame
df = df.drop(condition.index)

# Let's recheck the remaining number of instances where 'availability_365' equals
0.

# Filtering the dataset where 'availability_365' equals 0
condition = df[df['availability_365'] == 0]

# Calculating the number of rows and percentage
num_rows = len(condition)
percentage = round((num_rows / len(df)) * 100, 2)

# Displaying the results
zero_data = pd.DataFrame({'Total': [num_rows], 'Percent': [percentage]})
print(zero_data)

# It shows that there are a total of 12688 instances where the attribute
'availability_365' equals 0, which accounts for approximately 28.8% of the dataset.

# Now let's focus on identifying data that can be treated as "Fully booked". We
will consider instances where 'minimum_nights' is greater than or equal to 10,
'number_of_reviews' is greater or equal to 37, 'last_review' is greater than or
equal to 2019, and 'availability_365' is equal to 0. When we find it we will change
'0' to '365'.

# Miltering the dataset based on the given conditions
condition = df[(df['minimum_nights'] >= 10) & (df['number_of_reviews'] >= 37) &
(df['last_review'] >= 2019) & (df['availability_365'] == 0)]

# Modifying the 'availability_365' column to change '0' to '365' for the filtered
rows
df.loc[condition.index, 'availability_365'] = 365

# Counting the number of rows that satisfy the conditions
num_rows = len(condition)

# Calculating the percentage of filtered rows in the whole dataset
percentage = (num_rows / len(df)) * 100

# Displaying the results
```

```
print("Total: ", num_rows)
print("Percent: ", percentage, "%")

# The conditions we provided were met by only 18 instances in the dataset, and
their 'availability_365' values were successfully changed from '0' to '365'. The
remaining 12670 instances or 28.76% where 'availability_365' is still equal to 0
can be considered as "Not Available" based on the context of our analysis.

# Before conducting our analysis, the attribute 'availability_365' had 17,533
instances, accounting for 35.86% of values being equal to 0. After the analysis, we
categorized the instances as follows:

# Not Available: There are 12,670 instances, which accounts for 28.76% of the
dataset.
# Fully Booked: Only 18 instances, representing 0.04% of the dataset, were
identified as fully booked.
# Missing Data: There are 4,845 instances, accounting for 9.9% of the dataset,
where the 'availability_365' attribute is missing or undefined.

# Now we can drop unnecessary columns

# drop the specified columns
columns_to_drop = ['reviews_per_month', 'last_review']
df.drop(columns=columns_to_drop, inplace=True)

# Check for missing values

# check missing values again
df.isnull().sum().sum()

# Part 2: Exploratory Data Analysis.

# Numerical and Categorical attributes

# check for Numerical and Categorical attributes in the dataset
numerical_feats = df.dtypes[df.dtypes != 'object'].index
print('Quantity of Numerical features: ', len(numerical_feats))
print()
print(df[numerical_feats].columns)
print()
categorical_feats = df.dtypes[df.dtypes == 'object'].index
print('Quantity of Categorical features: ', len(categorical_feats))
print()
print(df[categorical_feats].columns)

# Visualisation of Numerical attributes

# visualisation of numerical attributes
data = Settings()
data.distplot_attributes(df)

data.boxplot_attributes(df)

# Visualization for categorical attributes we will explore a bit later.

# Outliers

# The only attribute we will examine for outliers is 'price'.

# 'Price' attribute (visualisation: box plot)

plt.figure(figsize=(16, 6))
sns.boxplot(x="neighbourhood_group", y="price", data=df)

# 'Price' attribute shows inconsistencies.
```

```
# Brooklyn:
# Likely: An entire house or apartment in Brooklyn, near Greenpoint, could cost
10,000 dollars.
# A film location in Brooklyn can cost 8,000 dollars.

# Unlikely: A private room in Brooklyn cannot cost 7,500 dollars. A private room in
Brooklyn, near the Williamsburg Bridge, cannot cost 5,000 dollars.

# Manhattan:
# Likely: An entire house or apartment in Manhattan could cost 10,000 dollars.

# Unlikely: A private room in Manhattan cannot cost 9,999 dollars.

# Queens:
# Unlikely: A private furnished room in Queens cannot cost 10,000 dollars.

# Staten Island:
# Likely: An entire house or apartment in Staten Island could cost 5,000 dollars.

# Bronx:
# Likely: A private room in the Bronx could cost 2,500 dollars.

# Therefore, it has been decided to drop the rows that contain these
inconsistencies.

# drop outliers
df = df.drop(df[(df['price'] >= 5000) & (df['neighbourhood_group'] == 'Brooklyn') &
(df['room_type'] == 'Private room')].index)
df = df.drop(df[(df['price'] == 9999) & (df['neighbourhood_group'] == 'Manhattan')
& (df['room_type'] == 'Private room')].index)
df = df.drop(df[(df['price'] == 10000) & (df['neighbourhood_group'] == 'Queens') &
(df['room_type'] == 'Private room')].index)

# Now, we can plot a scatterplot that illustrates the variation in listing prices
in New York City, without considering extreme prices. By doing so, the scatterplot
provides more informative insights into the price differences across different
areas of the city. Please refer to the scatterplot below to visualize this.

# visualisation 'longitude', 'latitude', and 'price'
plt.figure(figsize=(14,10))
ax = plt.gca()
df.plot(kind='scatter', x='longitude', y='latitude', c='price', ax=ax,
cmap=plt.get_cmap('RdBu'), colorbar=True, alpha=0.7);

# Visualisation

# 'neighbourhood_group' attribute (categorical)

# visualisation of 'neighbourhood_group'
df['neighbourhood_group'].value_counts().plot(x=df['neighbourhood_group'],
kind='bar')

#  Calculate average price per neighbourhood_group

# visualisation of 'neighbourhood_group' and 'price'
avg_price = df.groupby('neighbourhood_group')['price'].mean()

# create bar plot
plt.bar(avg_price.index, avg_price)
plt.xlabel('Neighbourhood Group')
plt.ylabel('Average Price per Night')
plt.title('Average Price per Night by Neighbourhood Group')
plt.xticks(rotation=45)
plt.show()

# describe 'neighbourhood_group'
```

```python
set(df['neighbourhood_group'])
{'Bronx', 'Brooklyn', 'Manhattan', 'Queens', 'Staten Island'}
df.groupby('neighbourhood_group')['price'].describe()

# The table above provides a clear overview of the price distribution within each
neighbourhood group as well as overall. The majority of Airbnb listings are
concentrated in Manhattan and Brooklyn, which also happen to have the highest
prices compared to the other regions. This can be attributed to the high demand in
these areas, prompting more hosts to offer their rooms or apartments for rent.

# 'room_type' attribute (categorical)

# To provide a more precise representation of the pricing, the table also includes
a breakdown based on room type, ensuring a more detailed analysis of the prices.

# visualisation 'room_type' and 'neighbourhood_group'
fig = plt.subplots(figsize=(12, 5))
sns.countplot(x='room_type', hue='neighbourhood_group', data=df)

# visualisation 'neighbourhood_group', 'room_type', and 'price'
plt.figure(figsize=(15, 10))
sns.boxplot(x='neighbourhood_group', y='price', hue='room_type', data=df)
plt.title('Price of Different Room Types in Each Neighbourhood Group')
plt.show()

# This countplot and boxplot shows that the highest number of private rooms is in
Brooklyn, while the highest number of entire homes/apartments and shared rooms is
in Manhattan.

# pivot table 'neighbourhood_group', 'room_type', and 'price'
df.pivot_table(index='neighbourhood_group', columns='room_type', values='price',
aggfunc='mean')


# The pivot table indicates that Manhattan has the highest mean price for entire
homes/apartments, private rooms, and shared rooms.

# 'minimum_nights' attribute

# Let's investigate the average minimum number of nights per listing across
different neighbourhood groups and room types.

# visualisation 'minimum_nights' and 'room_type'
sns.catplot('neighbourhood_group', 'minimum_nights', hue='room_type', data=df,
            kind='bar', ci=None, linewidth=1, edgecolor='w', height=8.27,
aspect=11.7/8.27)
plt.xlabel('Borough', fontsize=15, labelpad=15)
plt.xticks(fontsize=13)
plt.ylabel('Average minimum nights per listing', fontsize=17, labelpad=14)
plt.show()

# describe 'neighbourhood_group' and 'minimum_nights'
df.groupby('neighbourhood_group')['minimum_nights'].describe()

# The data indicates that more than 25% of Airbnb listings require only 1 night,
while over half of the listings require 2 or 3 nights. This aligns with the
fundamental principle of Airbnb as a short-term accommodation service.

# 'number_of_reviews' attribute

# Let's examine the number of reviews for different neighbourhood groups and room
types.

# visualisation 'number_of_reviews', 'neighbourhood_group', and 'room_type'
sns.catplot('neighbourhood_group', y='number_of_reviews', hue='room_type',
kind='bar',
```

```
             ci=None, data=df, linewidth=1, edgecolor='w', height=8.27,
aspect=11.7/8.27)
plt.xlabel('Borough', fontsize=15, labelpad=15)
plt.xticks(fontsize=13)
plt.ylabel('Average number of reviews per listing', fontsize=17, labelpad=14)
plt.show()

# The highest number of reviews for private rooms and entire home/apartments is
observed in Staten Island, while for shared rooms, it is highest in Manhattan.

# Price and reviews

# visualisation 'number_of_reviews' and 'price'
plt.figure(figsize=(8, 8))
sns.scatterplot(x='price', y='number_of_reviews', data=df[df.price <= 10000])
plt.title("Relationship Between Price and Number of Reviews (For Properties Priced
Below $1000)", fontsize=15)
plt.xlabel("Price", fontsize=12)
plt.ylabel("Number of review", fontsize=12)
plt.show()

# Based on the plot above, it is evident that properties within the price range of
0-400 USD tend to have higher numbers of reviews. As the price for the property
increases, the maximum number of reviews observed is 200.

# 'longitude' and 'latitude' attributes

# Let's explore the relationship between price, latitude/longitude, and
neighbourhood_group.

# visualisation 'longitude' and 'latitude'
fig = plt.figure(figsize=(15, 5))
fig.subplots_adjust(wspace=0.2, hspace=0.2)
a1 = fig.add_subplot(1, 2, 1)
a2 = fig.add_subplot(1, 2, 2)

sns.scatterplot(x='longitude', y='price', hue='neighbourhood_group', data=df,
ax=a1)
sns.scatterplot(x='latitude', y='price', hue='neighbourhood_group', data=df, ax=a2)

a1.set_title('price along a longitude')
a2.set_title('price along a latitude')
plt.show()

# It is evident that Manhattan and Brooklyn are the two neighborhood groups with
high prices for Airbnb listings.

# The scatterplot of the listings of our dataset:

# visualisation 'longitude', 'latitude' and 'neighbourhood_group'
fig, ax = plt.subplots(figsize=(10, 10))
sns.scatterplot(x='longitude', y='latitude', hue='neighbourhood_group', ax=ax,
s=20, alpha=0.2, data=df)
plt.title('The density of Airbnb listings in New York')
plt.show()

# The areas marked by thicker circles indicate a high number of Airbnb listings.

# 'availability_365' attribute

# describe 'neighbourhood_group' and 'availability_365'
df.groupby('neighbourhood_group')['availability_365'].describe()

# pivot table 'availability_365', 'room_type', and 'price'
df.pivot_table(index='availability_365', columns='room_type', values='price',
aggfunc='mean')
```

```python
# 'neighbourhood' attribute (categorical)

# visualisation 'neighbourhood'
neigh = df['neighbourhood'].value_counts()
neigh[neigh > 100].plot(figsize=(15, 5), kind='bar')
plt.title('Most common neighbourhood Airbnb')
plt.show()

# Williamsburg and Bedford-Stuyvesant are the neighborhoods that have a higher
number of Airbnb listings compared to others.

# visualisation 'neighbourhood' and 'price'
d = df.groupby('neighbourhood')['price'].mean().sort_values(ascending=False)
plt.figure(figsize=(12, 6))
sns.barplot(d.index.tolist()[:20], d.values[:20], palette=("Blues_d"))
plt.xticks(rotation=40, ha="right")
plt.title('Top 20 highest-priced neighborhoods')
plt.show()

# The table above showcases the top 20 neighborhoods with the highest prices.

# Business Questions Analysis

# Now, let's explore our business questions.

# Business Question A. Overall Market Landscape: What is the current economic
landscape of the New York City market for long-term rental properties exceeding 30
days or 1 month in the "entire home/apartment" format?

# Business Question A pivot table
new_df = df[(df['room_type'] == 'Entire home/apt') & (df['minimum_nights'] > 30)]

by_room_type = pd.pivot_table(new_df,
                              index=['room_type', 'neighbourhood_group'],
                              aggfunc={'neighbourhood_group': np.size})

by_room_type = by_room_type.rename(columns={'neighbourhood_group': 'Total Number of
Listings'})
by_room_type = by_room_type.sort_values(by='Total Number of Listings',
ascending=False)

total_listings = by_room_type['Total Number of Listings'].sum()

# calculate percentages with two decimal places
by_room_type['Percentage'] = round((by_room_type['Total Number of Listings'] /
total_listings) * 100, 2)

print("Total Number of 'Entire home/apt' Rental Properties in New York City:
{}".format(total_listings))
print(by_room_type)

# Answer: Out of 44,046 presented listings, 23,156 (52.6%) are classified as
"entire home/apartment", and only 478 listings (approximately 2.1%) identified as
long-term rental properties.

# With 311 listings, Manhattan is the leading neighbourhood group for long-term
rental properties, indicating a robust market for entire home/apartment rentals.
Brooklyn follows Manhattan with 140 listings, offering a significant alternative to
Manhattan. Queens has a smaller market share with only 17 listings, indicating
limited availability compared to Manhattan and Brooklyn. The Bronx has just 9
listings, suggesting a less developed market for long-term rentals in the entire
home/apartment format. Among the neighbourhood groups, Staten Island has the fewest
listings with only 1, indicating a highly limited market.
```

```
# Business Question B. Neighbourhood Group Analysis: Which neighbourhood groups
within New York City demonstrate notable popularity and demand for long-term
rentals of "entire home/apartment" properties, and to what extent?

# Business Question B pivot table
new_df = df[(df['room_type'] == 'Entire home/apt') & (df['minimum_nights'] > 30)]

by_room_type = pd.pivot_table(new_df,
                              index=['room_type', 'neighbourhood_group'],
                              values='number_of_reviews',
                              aggfunc={'neighbourhood_group': np.size,
'number_of_reviews': np.sum})

by_room_type = by_room_type.rename(columns={'neighbourhood_group': 'Total Number of
Listings', 'number_of_reviews': 'Total Number of Reviews'})

# Sort by total number of listings in descending order
by_room_type = by_room_type.sort_values(by='Total Number of Listings',
ascending=False)

total_listings = by_room_type['Total Number of Listings'].sum()

print("Total Number of Popular Neighbourhood Groups in New York City:")
print("Total Number of Listings: {}".format(total_listings))
print(by_room_type)

# Answer: Manhattan (311 listings and 3,538 reviews) stands out as the most popular
neighbourhood group, indicating a high demand for long-term rentals in this area.
Brooklyn (140 listings and 2,437 reviews) showcasing notable popularity. Queens (17
listings and 299 reviews) indicating a moderate level of popularity. The Bronx (9
listings and 52 reviews) demonstrates a relatively lower level of demand compared
to Manhattan and Brooklyn. Finally, Staten Island (1 listing and no reviews)
indicating the lowest level of popularity.

# In summary, Manhattan and Brooklyn are the most popular neighbourhood groups in
New York City for long-term rentals of "entire home/apartment" properties, while
Queens, the Bronx, and Staten Island have a lower level of demand.

# Business Question C. Neighbourhood Analysis: Which specific neighbourhoods within
New York City hold investment potential for the construction or purchase of long-
term rental properties in the "entire home/apartment" format? Furthermore, what is
the level of desirability for such properties in these neighbourhoods?

# Business Question D. Pricing Analysis: What are the average market prices for
"entire home/apartment" long-term rentals within each identified neighbourhood
group and specific neighbourhood?

# Manhattan

# Business Question C&D pivot table for Manhattan
new_df_manhattan = new_df[new_df['neighbourhood_group'] == 'Manhattan']

total_listings_manhattan = new_df_manhattan.shape[0]

by_room_type_manhattan = pd.pivot_table(new_df_manhattan,
                                        index=['neighbourhood_group',
'neighbourhood'],
                                        values=['number_of_reviews', 'price'],
                                        aggfunc={'number_of_reviews': np.sum,
'price': np.mean})

by_room_type_manhattan =
by_room_type_manhattan.rename(columns={'number_of_reviews': 'Total Number of
Reviews', 'price': 'Mean Price'})
by_room_type_manhattan = by_room_type_manhattan.sort_values(by='Total Number of
Reviews', ascending=False)
```

```
print("Total Number of 'Entire home/apt' Rental Properties in Manhattan:
{}".format(total_listings_manhattan))
print("Neighbourhoods in Manhattan by Number of Reviews (Highest to Lowest):")
print(by_room_type_manhattan)


# Brooklyn

# Business Question C&D pivot table for Brooklyn
new_df_brooklyn = new_df[new_df['neighbourhood_group'] == 'Brooklyn']

total_listings_brooklyn = new_df_brooklyn.shape[0]

by_room_type_brooklyn = pd.pivot_table(new_df_brooklyn,
                                       index=['neighbourhood_group',
'neighbourhood'],
                                       values=['number_of_reviews', 'price'],
                                       aggfunc={'number_of_reviews': np.sum,
'price': np.mean})

by_room_type_brooklyn = by_room_type_brooklyn.rename(columns={'number_of_reviews':
'Total Number of Reviews', 'price': 'Mean Price'})
by_room_type_brooklyn = by_room_type_brooklyn.sort_values(by='Total Number of
Reviews', ascending=False)

print("Total Number of 'Entire home/apt' Rental Properties in Brooklyn:
{}".format(total_listings_brooklyn))
print("Neighbourhoods in Brooklyn by Number of Reviews (Highest to Lowest):")
print(by_room_type_brooklyn)

# Queens

# Business Question C&D pivot table for Queens
new_df_queens = new_df[new_df['neighbourhood_group'] == 'Queens']

total_listings_queens = new_df_queens.shape[0]

by_room_type_queens = pd.pivot_table(new_df_queens,
                                     index=['neighbourhood_group',
'neighbourhood'],
                                     values=['number_of_reviews', 'price'],
                                     aggfunc={'number_of_reviews': np.sum, 'price':
np.mean})

by_room_type_queens = by_room_type_queens.rename(columns={'number_of_reviews':
'Total Number of Reviews', 'price': 'Mean Price'})
by_room_type_queens = by_room_type_queens.sort_values(by='Total Number of Reviews',
ascending=False)

print("Total Number of 'Entire home/apt' Rental Properties in Queens:
{}".format(total_listings_queens))
print("Neighbourhoods in Queens by Number of Reviews (Highest to Lowest):")
print(by_room_type_queens)

# Bronx

# Business Question C&D pivot table for Bronx
new_df_bronx = new_df[new_df['neighbourhood_group'] == 'Bronx']
total_listings_bronx = new_df_bronx.shape[0]

by_room_type_bronx = pd.pivot_table(new_df_bronx,
                                    index=['neighbourhood_group', 'neighbourhood'],
                                    values=['number_of_reviews', 'price'],
                                    aggfunc={'number_of_reviews': np.sum, 'price':
np.mean})
```

```python
by_room_type_bronx = by_room_type_bronx.rename(columns={'number_of_reviews': 'Total
Number of Reviews', 'price': 'Mean Price'})
by_room_type_bronx = by_room_type_bronx.sort_values(by='Total Number of Reviews',
ascending=False)

print("Total Number of 'Entire home/apt' Rental Properties in Bronx:
{}".format(total_listings_bronx))
print("Neighbourhoods in Bronx by Number of Reviews (Highest to Lowest):")
print(by_room_type_bronx)

# Staten Island

# Business Question C&D pivot table for Staten Island
new_df_staten_island = new_df[new_df['neighbourhood_group'] == 'Staten Island']

total_listings_staten_island = new_df_staten_island.shape[0]

by_room_type_staten_island = pd.pivot_table(new_df_staten_island,
                                            index=['neighbourhood_group',
'neighbourhood'],
                                            values=['number_of_reviews', 'price'],
                                            aggfunc={'number_of_reviews': np.sum,
'price': np.mean})

by_room_type_staten_island =
by_room_type_staten_island.rename(columns={'number_of_reviews': 'Total Number of
Reviews', 'price': 'Mean Price'})
by_room_type_staten_island = by_room_type_staten_island.sort_values(by='Total
Number of Reviews', ascending=False)

print("Total Number of 'Entire home/apt' Rental Properties in Staten Island:
{}".format(total_listings_staten_island))
print("Neighbourhoods in Staten Island by Number of Reviews (Highest to Lowest):")
print(by_room_type_staten_island)

# **Answer for business question C:** The level of desirability based on their
total number of listings and the corresponding number of reviews.

# **In Manhattan**, the neighbourhoods with the highest number of reviews,
indicating popularity and demand, are:
# * Harlem: 676 reviews (significant investment potential);
# * Upper West Side: 514 reviews (desirability for long-term rentals);
# * West Village, Chelsea, and East Harlem are other noteworthy neighbourhoods with
substantial numbers of reviews.

# **In Brooklyn**, the neighbourhoods with notable investment potential and
desirability for long-term rentals are:
# * Bedford-Stuyvesant: 953 reviews (high demand);
# * Bushwick and Williamsburg are also popular neighbourhoods with significant
numbers of reviews,

# **Queens**, despite having a smaller number of listings, has neighbourhoods with
moderate levels of desirability based on reviews. The neighbourhoods of Ditmars
Steinway and Astoria have the highest number of reviews among the Queens
neighbourhoods, suggesting a certain level of investment potential.

# **The Bronx and Staten Island** have a relatively lower number of listings and
reviews compared to Manhattan and Brooklyn, indicating a lower level of demand and
investment potential.

# **Answer for business question D:** The average prices for each neighbourhood
group and specific neighbourhood:
# * **In Manhattan:** Neighbourhoods such as Nolita, Hell's Kitchen, and Greenwich
Village have higher average prices, ranging from 294 dollars to 348 dollars;
Roosevelt Island stands out with an average price of 762.50 dollars, indicating a
```

```
higher-end rental market; other neighbourhoods have relatively lower average
prices, ranging from 79.50 dollars to 287.31 dollars.

# * **In Brooklyn:** Neighbourhoods like Crown Heights, Carroll Gardens, and
Windsor Terrace have average prices ranging from 114 dollars to 266 dollars;
Prospect-Lefferts Gardens and Greenpoint show higher average prices of 568 dollars
and 190.60 dollars, respectively; Some neighbourhoods have lower average prices,
such as Downtown Brooklyn with 97 dollars and Clinton Hill with 89 dollars.

# * **In Queens:** Ditmars Steinway and Astoria have average prices of 69.50
dollars and 89.50 dollars, respectively; Long Island City shows a slightly higher
average price of 139 dollars, indicating a relatively more expensive rental market.

# * **In The Bronx and Staten Island:** The Bronx neighbourhoods have average
prices ranging from 54 dollars to 91.50 dollars, representing relatively more
affordable options; Staten Island's St. George neighbourhood has an average price
of 100 dollars.

# ### Encoding of the Categorical data

# For our purposes, we are applying a **label encoding** technique.

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

# label encoding for 'room_type'
le_room_type = LabelEncoder()
df['room_type'] = le_room_type.fit_transform(df.room_type.values)
df.head()

# **room_type == 0 means Entire home/apt**
# **room_type == 1 means Private room**
# **room_type == 2 means Shared room**

# We understand that a good Exploratory Data Analysis (EDA) should include Skewness
and Kurtosis, data distribution analysis, correlation list and matrix, and the use
of StandardScaler. However, we will skip these steps for the following reasons:
# 1. If we apply log transformation to attributes such as "price" and
"minimum_nights", we will not be able to visualize the actual prices and number of
nights accurately.
# 2. If we use Label Encoding for all categorical attributes, we will lose the
ability to identify the actual names of neighbourhoods and neighbourhood groups in
our visualizations. We have only applied Label Encoding to the "room_type"
attribute for our analysis.
# 3. Applying StandardScaler would prevent us from accurately locating the clusters
on the real map of New York City. Therefore, we have chosen to preserve the
original values for better visualization and understanding of the cluster
locations.
# 4. The presence of a correlation list and matrix is not relevant to our business
questions, so we have decided not to include it in our solution.
# Considering these reasons, we have decided to omit these specific steps in our
EDA to focus on addressing our business objectives.

# dataset shape
df.shape

# # Part 3: Machine Learning.

# We have chosen the **K-Means algorithm** to implement our solution. Although we
also explored the **DBSCAN** algorithm, K-Means has shown better results in
addressing our business questions.

# K-Means algorithm
```

```python
# select relevant columns for clustering
fields = ['latitude', 'longitude']

# perform k-means clustering with elbow method for optimal cluster selection
def clustering_kelbow_score(data, min_clusters=2, max_clusters=10, fields=[]):
    model = KMeans()
    visualizer = KElbowVisualizer(model, k=(min_clusters, max_clusters))
    visualizer.fit(data[fields])
    visualizer.show()
    return visualizer.elbow_value_

# filter data based on room type and minimum nights
cluster_df = df[(df['room_type'] == 0) & (df['minimum_nights'] > 30)]

# Determine the optimal number of clusters
estimated_clusters_num = clustering_kelbow_score(cluster_df, fields=fields)

# perform k-means clustering
kmeans = KMeans(n_clusters=estimated_clusters_num, random_state=0)
cluster_df['cluster_labels'] = kmeans.fit_predict(cluster_df[fields])

# The elbow point, which is found at **4 clusters**, indicates that using 4
clusters strikes a good balance in capturing the underlying patterns and structure
in the data. As well as this choice avoids excessive complexity or overfitting.

# returns an array containing the cluster labels
print(kmeans.labels_)

# print the cluster labels
print("Cluster Labels:")
print(cluster_df['cluster_labels'])

# calculate Silhouette Score
from sklearn.metrics import silhouette_score
silhouette_avg = silhouette_score(cluster_df[fields], cluster_df['cluster_labels'])
print("Silhouette Score:", silhouette_avg)

# visualize Silhouette Scores
fig, ax = plt.subplots(figsize=(8, 6))
visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick')
visualizer.fit(cluster_df[fields])
visualizer.show()

# The Silhouette Score of 0.4638275849406755 indicates a moderate level of
separation and cohesion. It implies that the K-Means algorithm has successfully
grouped the data points, but there may still be some overlapping or ambiguity in
the assignments. Further analysis and fine-tuning of the clustering parameters
could potentially improve the cluster quality and enhance the separation among the
clusters.

# display head of clusters
clusters_head = cluster_df.groupby('cluster_labels').head()
clusters_head.head()

# visualisation all clusters
# this function is used to help plot the maps
def embed_map(m, file_name):
    from IPython.display import IFrame
    m.save(file_name)
    return IFrame(file_name, width='100%', height='500px')

# set color for clusters
num_clusters = len(cluster_df['cluster_labels'].unique())
palette = ['red', 'blue', 'green', 'orange']  # add or modify colors in the palette
```

```
# create the folium map
map_cluster = folium.Map(location=[40.730610, -73.935242], zoom_start=10)

# set the marker for the map
markers_colors = []
for lat, lng, cluster in zip(cluster_df['latitude'], cluster_df['longitude'],
cluster_df['cluster_labels']):
    label = folium.Popup('Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker([lat, lng],
                        radius=2,
                        popup=label,
                        color=palette[cluster],
                        fill=True,
                        fill_color=palette[cluster],
                        fill_opacity=0.7).add_to(map_cluster)

embed_map(map_cluster, 'map_cluster.html')

# #### Cluster 0

# cluster 0 head
cluster_0_data = cluster_df[cluster_df['cluster_labels'] == 0]
print("Cluster 0 Count:", len(cluster_0_data))
cluster_0_data.head()

# describe cluster 0
cluster_0_data = cluster_df[cluster_df['cluster_labels'] == 0]
print("Cluster 0 Count:", len(cluster_0_data))
print("Descriptive Statistics for Cluster 0:")
print(cluster_0_data.describe())

# visualisation cluster 0
# this function is used to help plot the maps
def embed_map(m, file_name):
    from IPython.display import IFrame
    m.save(file_name)
    return IFrame(file_name, width='100%', height='500px')

# set color for the desired cluster
cluster_color = 'red'

# filter data for the desired cluster
desired_cluster_data = cluster_0_data

# create the folium map
map_cluster = folium.Map(location=[40.730610, -73.935242], zoom_start=10)

# set the marker for the map
for lat, lng in zip(desired_cluster_data['latitude'],
desired_cluster_data['longitude']):
    label = folium.Popup('Cluster 0', parse_html=True)
    folium.CircleMarker([lat, lng],
                        radius=2,
                        popup=label,
                        color=cluster_color,
                        fill=True,
                        fill_color=cluster_color,
                        fill_opacity=0.7).add_to(map_cluster)

embed_map(map_cluster, 'map_cluster.html')

# #### Cluster 1

# cluster 1 head
cluster_1_data = cluster_df[cluster_df['cluster_labels'] == 1]
print("Cluster 1 Count:", len(cluster_1_data))
```

```
cluster_1_data.head()

# describe cluster 1
cluster_1_data = cluster_df[cluster_df['cluster_labels'] == 1]
print("Cluster 1 Count:", len(cluster_1_data))
print("Descriptive Statistics for Cluster 1:")
print(cluster_1_data.describe())

# visualisation cluster 1
# this function is used to help plot the maps
def embed_map(m, file_name):
    from IPython.display import IFrame
    m.save(file_name)
    return IFrame(file_name, width='100%', height='500px')

# set color for the desired cluster
cluster_color = 'blue'

# filter data for the desired cluster
desired_cluster_data = cluster_1_data

# create the folium map
map_cluster = folium.Map(location=[40.730610, -73.935242], zoom_start=10)

# set the marker for the map
for lat, lng in zip(desired_cluster_data['latitude'],
desired_cluster_data['longitude']):
    label = folium.Popup('Cluster 1', parse_html=True)
    folium.CircleMarker([lat, lng],
                        radius=2,
                        popup=label,
                        color=cluster_color,
                        fill=True,
                        fill_color=cluster_color,
                        fill_opacity=0.7).add_to(map_cluster)

embed_map(map_cluster, 'map_cluster.html')

# #### Cluster 2

# cluster 2 head
cluster_2_data = cluster_df[cluster_df['cluster_labels'] == 2]
print("Cluster 2 Count:", len(cluster_2_data))
cluster_2_data.head()

# describe cluster 2
cluster_2_data = cluster_df[cluster_df['cluster_labels'] == 2]
print("Cluster 2 Count:", len(cluster_2_data))
print("Descriptive Statistics for Cluster 2:")
print(cluster_2_data.describe())

# visualisation cluster 2
# this function is used to help plot the maps
def embed_map(m, file_name):
    from IPython.display import IFrame
    m.save(file_name)
    return IFrame(file_name, width='100%', height='500px')

# set color for the desired cluster
cluster_color = 'green'

# filter data for the desired cluster
desired_cluster_data = cluster_2_data

# create the folium map
map_cluster = folium.Map(location=[40.730610, -73.935242], zoom_start=10)
```

```python
# set the marker for the map
for lat, lng in zip(desired_cluster_data['latitude'],
desired_cluster_data['longitude']):
    label = folium.Popup('Cluster 2', parse_html=True)
    folium.CircleMarker([lat, lng],
                        radius=2,
                        popup=label,
                        color=cluster_color,
                        fill=True,
                        fill_color=cluster_color,
                        fill_opacity=0.7).add_to(map_cluster)

embed_map(map_cluster, 'map_cluster.html')

# #### Cluster 3

# cluster 3 head
cluster_3_data = cluster_df[cluster_df['cluster_labels'] == 3]
print("Cluster 3 Count:", len(cluster_3_data))
cluster_3_data.head()

# describe cluster 3
cluster_3_data = cluster_df[cluster_df['cluster_labels'] == 3]
print("Cluster 3 Count:", len(cluster_3_data))
print("Descriptive Statistics for Cluster 3:")
print(cluster_3_data.describe())

# visualisation cluster 3
# this function is used to help plot the maps
def embed_map(m, file_name):
    from IPython.display import IFrame
    m.save(file_name)
    return IFrame(file_name, width='100%', height='500px')

# set color for the desired cluster
cluster_color = 'orange'

# filter data for the desired cluster
desired_cluster_data = cluster_3_data

# create the folium map
map_cluster = folium.Map(location=[40.730610, -73.935242], zoom_start=10)

# set the marker for the map
for lat, lng in zip(desired_cluster_data['latitude'],
desired_cluster_data['longitude']):
    label = folium.Popup('Cluster 0', parse_html=True)
    folium.CircleMarker([lat, lng],
                        radius=2,
                        popup=label,
                        color=cluster_color,
                        fill=True,
                        fill_color=cluster_color,
                        fill_opacity=0.7).add_to(map_cluster)

embed_map(map_cluster, 'map_cluster.html')

# #### All clusters

# visualisation all clusters
cluster_sizes = [len(cluster_0_data), len(cluster_1_data), len(cluster_2_data),
len(cluster_3_data)]
cluster_names = ['Cluster 0', 'Cluster 1', 'Cluster 2', 'Cluster 3']
```

```
df_clu = pd.DataFrame({'Cluster Number': cluster_names, 'Total Number of Entire
Home/Apt': cluster_sizes})

plt.figure(figsize=(10, 8))
ax = sns.barplot(data=df_clu, x='Cluster Number', y='Total Number of Entire
Home/Apt')
plt.xlabel('Cluster Number', fontsize=12)
plt.ylabel('Total Number of Entire Home/Apt for All Clusters', fontsize=12)
plt.title('Total Number of Entire Home/Apt for All Clusters', fontsize=12)

# add value labels to the bars
for i, v in enumerate(cluster_sizes):
    ax.text(i, v + 10, str(v), ha='center', va='bottom', fontsize=10)

plt.show()

# Two clustering algorithms, namely **K-Means and DBSCAN**, were employed in this
analysis. However, for a cleaner and more focused solution, we have decided to
present only the results obtained from the **K-Means** algorithm.

# The **"k-means++"** initialization method has been selected for K-Means
clustering. Various experiments on the dataset showed a degraded performance and
lower **Silhouette Score** for random centroid initialization. Updated of
**"n_init"** setting, which defines how many times k-means runs with different
centroids, also did not improve either the clustering result or the score. Both
**"lloyd"** and **"elkan"** algorithms for the "algorithm" setting showed similar
performance, and the default settings were used for further clustering. Experiments
with the number of clusters showed degraded performance in all cases when the value
different from the **Elbow method** result was used.

# For **DBSCAN**, default settings were used, except the maximum distance between
samples and the minimal count of samples to form a cluster. Other settings
manipulation did not show any visible difference in results.
# For **"eps"** and **"min_samples"**, the values were chosen to correspond to the
nature of the dataset and particular parameters for clustering. However, the
heterogeneous distribution of the geo data, with significant gaps in between
coordinates and a low count of samples,  provided questionable results, either one
big cluster and big chunks of noise data or many clusters with most samples in a
big one.

# save cluster_df as a .csv file
cluster_df.to_csv('cluster_dataset.csv', index=False)
```