Development Team Project:

Project Report

Nikos Archontas, Fabian Narel, Matilda Nilsson, João Torres and Maryum Rasool

Introduction

The rise of online platforms like Airbnb has transformed the hotel industry by giving customers more control over pricing, reviews, and location research (Sharma & Gupta, 2021). Airbnb has become a top accommodation option worldwide, especially in New York City (Jiao & Bai, 2019). This report analyzes Airbnb listings in NYC, focusing on pricing, availability, location, reviews, and property types. Understanding these factors can help Airbnb optimize pricing and improve guest engagement.

1. Proposed Business Analytic Question:

Which NYC neighborhood had the highest average Airbnb listing price in 2019, and what factors contributed to its high pricing? Additionally, is there a statistical relationship between Airbnb prices and the number of reviews a listing receives?

2. Data Analysis

Data Preprocessing

The dataset initially contained 48,895 rows and 16 columns. We removed records with zero reviews, zero prices, and extreme outliers. After preprocessing, the dataset was reduced to 38,833 entries.

Exploratory Data Analysis (EDA)

The number of Airbnb listings by neighborhood groups showed that Manhattan had the highest number of listings, closely followed by Brooklyn. Staten Island and the Bronx had considerably fewer listings (Figure 1).

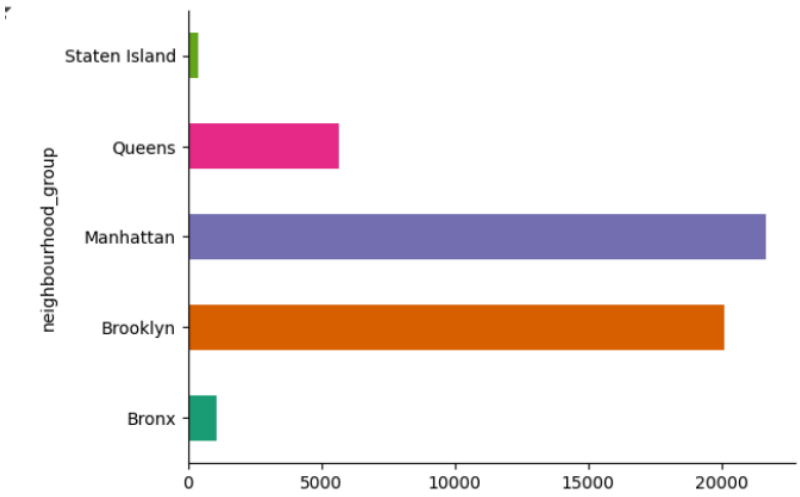


Figure 1: Visualisation of the number of Airbnb listings per neighborhood group

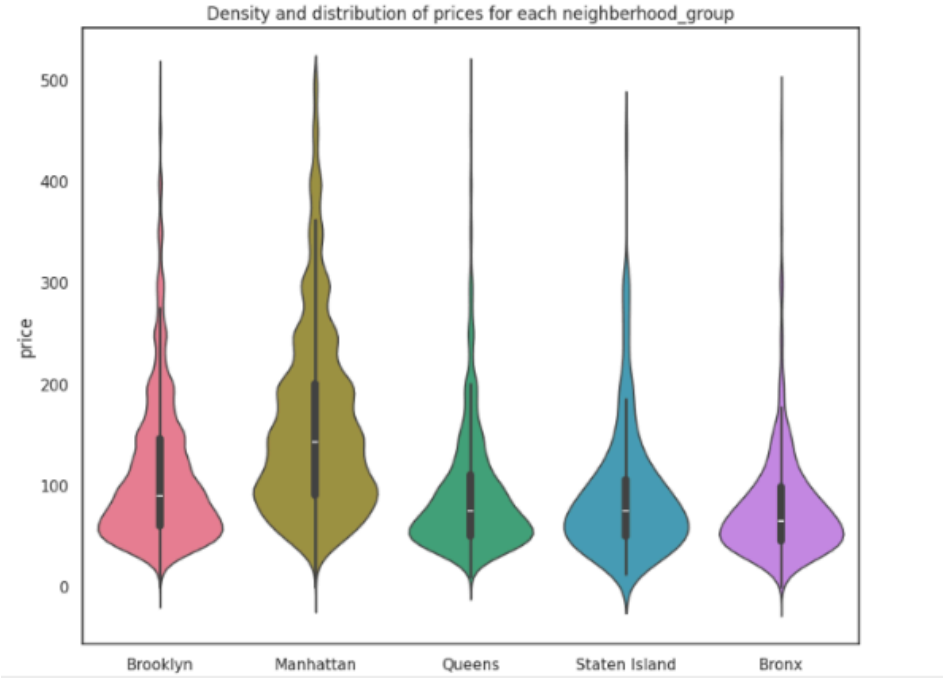


Figure 2: Violin Plot depicting the density and distribution of prices for airbnb listings for each New York neighborhood group in 2019

Manhattan had the highest median price at $150 per night, while the Bronx had the lowest at $65 per night. Summary statistics revealed the overall median price was $142.33, with a standard deviation of $196.99, indicating significant price differences across listings (Figure 2, Figure 3).

```
[ ] data.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365	last_review_day	last_review_month	last_review_year
count	3.882100e+04	3.882100e+04	38821.000000	38821.000000	38821.000000	38821.000000	38821.000000	38821	38821.000000	38821.000000	38821.000000	38821.000000	38821.000000	38821.000000
mean	1.810081e+07	6.424582e+07	40.728129	-73.951149	142.332526	5.869220	29.290255	2018-10-04 07:23:47.072975872	1.373229	5.166611	114.886299	15.792509	6.173901	2018.288323
min	2.539000e+03	2.438000e+03	40.506410	-74.244420	0.000000	1.000000	1.000000	2011-03-28 00:00:00	0.010000	1.000000	0.000000	1.000000	1.000000	2011.000000
25%	8.721444e+06	7.029525e+06	40.688640	-73.982460	69.000000	1.000000	3.000000	2018-07-09 00:00:00	0.190000	1.000000	0.000000	6.000000	5.000000	2018.000000
50%	1.887286e+07	2.837092e+07	40.721710	-73.954810	101.000000	2.000000	9.000000	2019-05-19 00:00:00	0.720000	1.000000	55.000000	17.000000	6.000000	2019.000000
75%	2.756746e+07	1.018905e+08	40.762990	-73.935020	170.000000	4.000000	33.000000	2019-06-23 00:00:00	2.020000	2.000000	229.000000	24.000000	7.000000	2019.000000
max	3.645581e+07	2.738417e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	2019-07-08 00:00:00	58.500000	327.000000	365.000000	31.000000	12.000000	2019.000000
std	1.069372e+07	7.589752e+07	0.054991	0.046693	196.994756	17.389026	48.182900	NaN	1.680328	26.302954	129.529950	9.881894	2.531374	1.180598

Figure 3: Descriptive statistics on the dataset, achieved through the data.describe command.

In Manhattan and Brooklyn, room type had a significant impact on pricing. Entire homes and apartments had a moderate positive correlation with price (r=0.476), while private rooms had a negative correlation (r=-0.450), supporting the preference for privacy in these neighborhoods (Perez-Sanchez et al., 2019). Longitude also showed a negative correlation with price, reflecting Manhattan's property value decline from expensive areas like SoHo to more affordable locations like East Harlem (Coles et al., 2017) (Figure 4). The number of reviews had a weak correlation with price.

Count	38821
Mean	$142.33
Standard Deviation	196.99
25th Percentile	$69

50th Percentile	$101
75th Percentile	$170
Max	$10000

Table 1: Descriptive statistics on Airbnb listing prices.

Box plots confirmed that Entire Home/Apartment listings had more than double the median price of Private and Shared rooms. The number of reviews had no significant impact on price. Minimum nights had a weak correlation with price, showing lower prices for properties that allowed single-night stays (Figure 5).

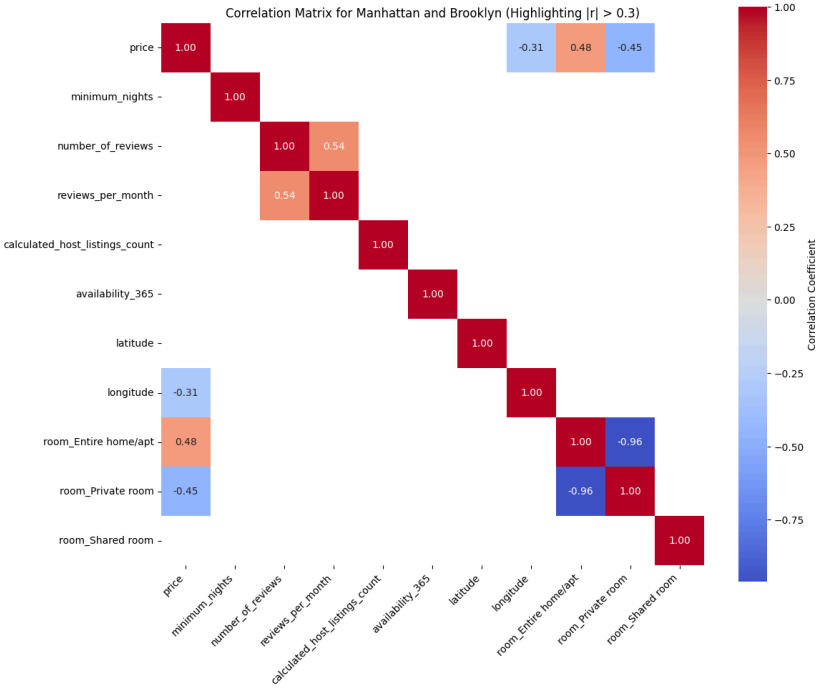


Figure 4: Correlation Matrix for Manhattan and Brooklyn data.

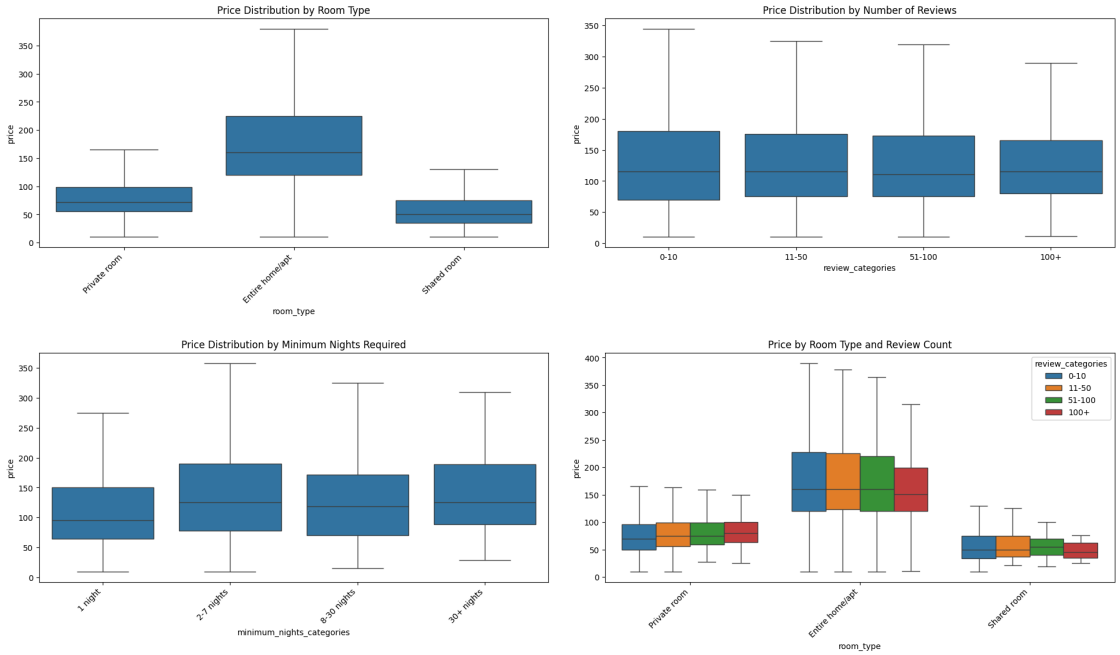


Figure 5: Box plots comparing Price Distribution by Room Type, Number of Reviews, Minimum Nights Required and Room Type combined with Number of Reviews

Figure 6: Histogram of nightly prices for NYC Airbnb listings (prices capped at $1000).

The distribution of NYC Airbnb prices was highly skewed, with most listings priced under $1,000, though a few luxury properties exceeded $10,000. The median price was inflated by high-end properties. The market reflected a mix of budget and premium options, with 75% of listings allowing stays of five nights or fewer (Figure 6).

Correlation Analysis

Expensive listings tend to have fewer reviews, suggesting lower booking frequency. In the 2019 NYC data, the weak negative correlation ($r \approx -0.058$, $p \approx 2.6 \times 10^{-37}$) indicates that budget-friendly listings attract more guests and reviews, while high-priced ones cater to a niche market (Martinez et al., 2017). This aligns with research on tourism demand dynamics, where lower prices can boost occupancy, but higher prices reduce the guest pool (Toader et al., 2022).

Figure 7: Visualisation of scatter plot of price vs. number of reviews

The scatter plot (Figure 7) supports this, showing a dense cluster of low-priced, well-reviewed listings and scattered high-priced listings with few reviews. Many expensive listings had 0–5 reviews, reflecting rare bookings. This reinforces the price-occupancy trade-off, where lower prices attract more bookings, while higher prices target fewer guests. Price showed minimal correlation with other features, aside from the slight negative link with reviews.

Machine Learning Analysis

To further explore patterns at the neighborhood level, we performed K-Means clustering on neighborhoods based on average price and average number of reviews per listing as well as linear regression to examine the relationship between prices and various predictors, including neighborhood group, number of reviews, and host listing count, with the idea to determine whether these factors impact price (Schroeder et al., 2017).

Linear Regression

The linear regression model tests whether neighborhood group, reviews, and host listing count influence price. While statistically significant ($F = 205.263$, $p < 0.001$), it explains only 9.3% of price variance ($R^2 = 0.093$), indicating many unaccounted factors (Chatterjee & Hadi, 2006). The high RMSE (112.457) reflects substantial predictive error. We reject the null hypothesis, confirming that at least one predictor impacts price (Field, 2024).

Predictor	Coefficient	P value	Interpretation
Number of reviews	-0.116	<0.001	Every additional review results in the price dropping by $0.12 on average. P value suggests significant impact.
Reviews per month	-0.073	0.917	Each additional review decreases the price by $0.07. Not statistically significant.
Neighbourhood group (Brooklyn)	41.529	<0.001	$41.53 more expensive than the baseline.
Neighbourhood group (Manhattan)	99.871	<0.001	Listings are $99.87 more expensive that the baseline reference
Neighbourhood group (Queens)	16.440	16.440	Queens listing are 16.44$ more expensive But have a weaker impact
Neighbourhood group (Staten Island)	10.920	0.392	Does not significantly impact price ($p > 0.05$)

Table 2: Regression table highlighting the effects individuals have on price

The analysis confirms significant price differences among neighborhood groups, with Manhattan and Brooklyn commanding higher prices than the Bronx. Listings with more reviews tend to have lower prices, indicating budget-friendly properties attract more guests (Perez-Sanchez et al., 2018; Coles et al., 2017), while "reviews per month" had no significant impact.

Linear regression identifies neighborhood groups as a key price predictor, with Manhattan and Brooklyn as premium markets. However, its low explanatory power (R^2 = 9.3%) suggests future models should include factors like room type, amenities, seasonality, and demand for better accuracy (Chatterjee & Hadi, 2006; Schroeder et al., 2017).

K-Means Clustering

Performing K-Means clustering on NYC neighborhoods based on average price and number of reviews to segment the market into three distinct groups:

- **Cluster 1 – "Luxury Markets"**: High price, low reviews. Representing about 1% of neighborhoods, these areas have ultra-expensive listings with minimal bookings, catering to an exclusive clientele (e.g., Fort Wadsworth, Staten Island).
- **Cluster 2 – "Mid-Range Markets"**: Moderate price, moderate reviews. The largest group (79%) includes areas like Manhattan and Brooklyn, offering moderately priced listings with steady guest traffic (e.g., popular parts of Brooklyn and Lower Manhattan).
- **Cluster 3 – "Budget-Friendly/High-Occupancy"**: Lower price, high reviews. Comprising 20% of neighborhoods, these areas attract budget travelers and high occupancy, with listings priced under $100 and many reviews (e.g., East Elmhurst, Queens).

These clusters represent different market segments, from high-end to budget-friendly options, and illustrate varying pricing strategies and guest demands across the city.

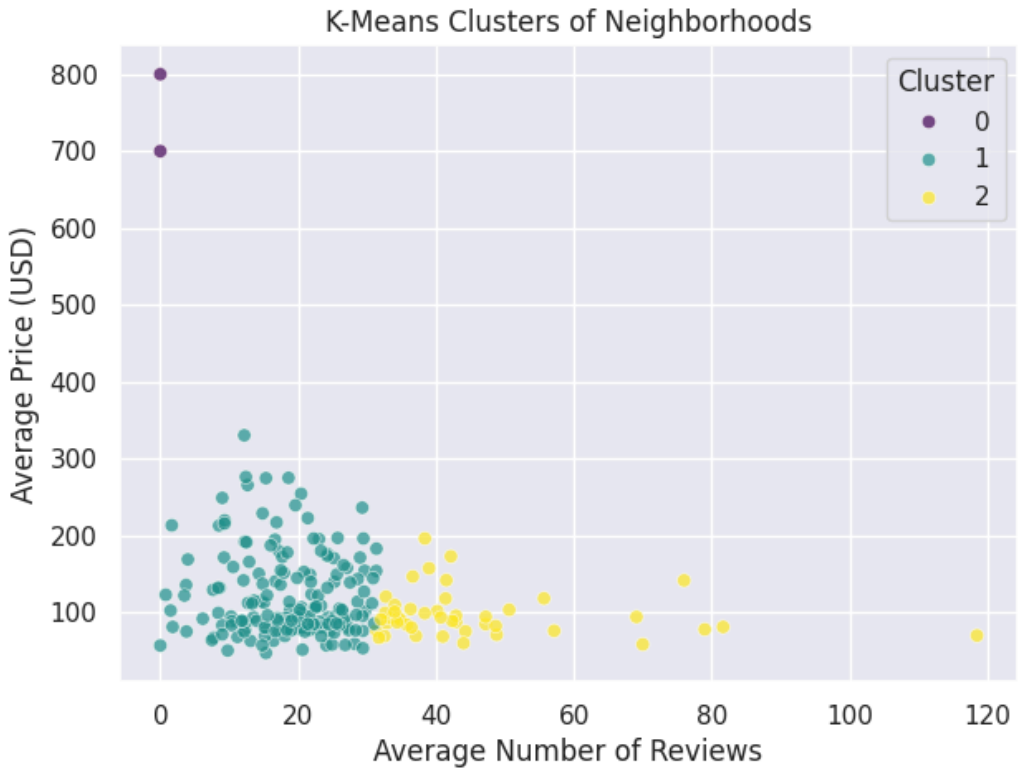


Figure 8: Visualisation of K-means clustering of neighborhoods

The cluster visualization (Figure 8) shows three distinct groups: budget (low-price, high-review), mid-range (moderate price and reviews), and luxury (high-price, low-reviews). This segmentation offers strategic insights for Airbnb hosts and stakeholders. Hosts in mid-range areas should differentiate their listings to stand out, while budget hosts should focus on high occupancy and positive reviews. Luxury hosts should emphasize exclusive amenities and experiences. Airbnb can tailor marketing and pricing strategies based on these segments, targeting budget travelers with promotions and luxury segments through niche channels.

3. Findings and Recommendations

Airbnb has significant growth potential in New York City. The dataset reveals high demand near major landmarks, which could be leveraged to increase profits by targeting accommodations in these areas. Additionally, offering a mix of premium housing in high-demand sectors and smaller rooms in less popular areas could help Airbnb remain competitive through dynamic pricing. By tracking location-specific trends and local events, Airbnb can better plan promotional offers tailored to neighborhoods and hot spots.

4. Conclusion

This report identifies key business opportunities in New York City for Airbnb, including targeted marketing strategies and revenue optimization. By focusing on specific areas, Airbnb can enhance customer satisfaction and increase revenue. The report also demonstrates the use of data analysis and machine learning for strategic decision-making and business growth.

Appendix

Correlation and Machine Learning Analysis

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

A histogram for price distribution (<= 1000)

```
plt.figure(figsize=(7,5))
sns.histplot(df_eda['price'], bins=50, kde=True)
plt.title("Overall Price Distribution (<= $1000)")
plt.xlabel("Price (USD)")
plt.show()
```

Correlation analysis between price and numeric features number_of_reviews, minimum_nights, availability_365, etc.

```
feat_cols = ['price','minimum_nights','number_of_reviews','reviews_per_month','calculated_host_listings_count','availability_365']
corr_mat = df_eda[feat_cols].corr()
corr_mat['price'].sort_values(ascending=False)
```

	price
price	1.000000
calculated_host_listings_count	0.130687
availability_365	0.117916
minimum_nights	0.019487
reviews_per_month	-0.055841
number_of_reviews	-0.057829

dtype: float64

And plotting

```
plt.figure(figsize=(7,5))
sns.scatterplot(x='number_of_reviews', y='price', data=df_eda, alpha=0.5)
plt.title("Price vs Number of Reviews")
plt.xlabel("Number of Reviews")
plt.ylabel("Price (USD)")
plt.show()
```

Pearson correlation for Price vs Numbers

```
from scipy.stats import pearsonr

r_value, p_value = pearsonr(df_eda['price'], df_eda['number_of_reviews'])
print(f"Pearson correlation = {r_value:.3f}, p-value = {p_value:.5e}")
if p_value < 0.05:
    print("Correlation is statistically significant.")
else:
    print("No statistically significant correlation.")
```

```
Pearson correlation = -0.058, p-value = 2.57252e-37
Correlation is statistically significant.
```

K-means clustering by aggregating data by neighbourhood to compute mean price and mean reviews. Feature scaling and clustering.

```
# 1. Aggregate
neigh_stats = df_eda.groupby('neighbourhood').agg({
    'price':'mean',
    'number_of_reviews':'mean'
}).reset_index()

neigh_stats.rename(columns={'price':'avg_price','number_of_reviews':'avg_reviews'}, inplace=True)
print(neigh_stats.head())
print("Unique neighborhoods:", len(neigh_stats))
```

```
   neighbourhood   avg_price  avg_reviews
0       Allerton   87.595238    42.928571
1  Arden Heights   67.250000     7.750000
2       Arrochar  115.000000    14.619048
3        Arverne  154.302632    29.644737
4        Astoria  100.635045    21.543527
Unique neighborhoods: 221
```

```
# 2. Scale the features (avg_price, avg_reviews)
from sklearn.preprocessing import StandardScaler
X = neigh_stats[['avg_price','avg_reviews']].values
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 3. K-Means with k=3
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
neigh_stats['cluster'] = kmeans.fit_predict(X_scaled)
neigh_stats.head()
```

	neighbourhood	avg_price	avg_reviews	cluster
0	Allerton	87.595238	42.928571	2
1	Arden Heights	67.250000	7.750000	1
2	Arrochar	115.000000	14.619048	1
3	Arverne	154.302632	29.644737	1
4	Astoria	100.635045	21.543527	1

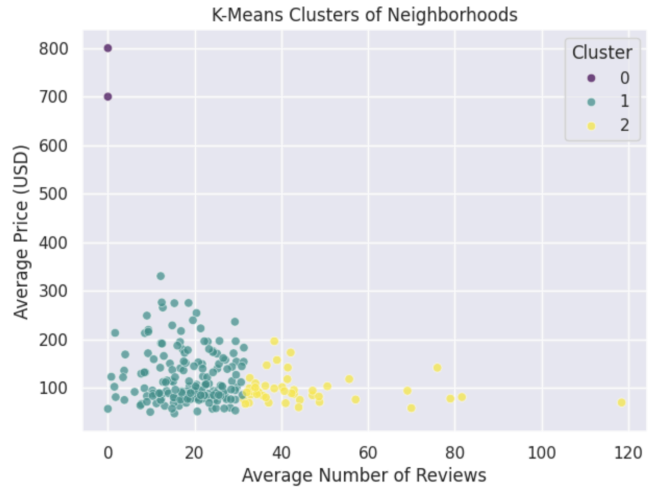
Cluster profiling.

```
centers_scaled = kmeans.cluster_centers_
centers_orig = scaler.inverse_transform(centers_scaled)

for idx, c in enumerate(centers_orig):
    print(f"Cluster {idx}: avg_price=${c[0]:.2f}, avg_reviews={c[1]:.1f}")
```

```
Cluster 0: avg_price=$750.00, avg_reviews=0.0
Cluster 1: avg_price=$121.49, avg_reviews=18.6
Cluster 2: avg_price=$97.36, avg_reviews=45.4
```

Plotting:

```
plt.figure(figsize=(7,5))
sns.scatterplot(
    x='avg_reviews', y='avg_price',
    hue='cluster', data=neigh_stats,
    palette='viridis', alpha=0.7
)
plt.title("K-Means Clusters of Neighborhoods")
plt.xlabel("Average Number of Reviews")
plt.ylabel("Average Price (USD)")
plt.legend(title="Cluster")
plt.show()
```



Data Pre-Processing

Libraries used, Data import and initial analysis:

```
import pandas as pd
import numpy as np
from scipy import stats

import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
[ ]  sns.set_theme()
```

```
[ ]  data = pd.read_csv("/content/AB_NYC_2019.csv")
     data.head()
```

	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_hos
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	2018-10-19	0.21	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	0.38	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	NaN	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05	4.64	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	0.10	

```
[ ] print("Number of rows     :{}".format(data.shape[0]))
    print("Number of columns  :{}".format(data.shape[1]))
```

```
Number of rows     :48895
Number of columns  :16
```

```
[ ] data.isnull().sum()
```

	0
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

Descriptive statistics of the data set:

```
[ ] data.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365	last_review_day	last_review_month	last_review_year
count	3.882100e+04	3.882100e+04	38821.000000	38821.000000	38821.000000	38821.000000	38821.000000	38821	38821.000000	38821.000000	38821.000000	38821.000000	38821.000000	38821.000000
mean	1.810081e+07	6.424582e+07	40.728129	-73.951149	142.332526	5.869220	29.290255	2018-10-04 07:23:47.072975872	1.373229	5.166611	114.886299	15.792509	6.173901	2018.288323
min	2.539000e+03	2.438000e+03	40.506410	-74.244420	0.000000	1.000000	1.000000	2011-03-28 00:00:00	0.010000	1.000000	0.000000	1.000000	1.000000	2011.000000
25%	8.721444e+06	7.029525e+06	40.688640	-73.982460	69.000000	1.000000	3.000000	2018-07-09 00:00:00	0.190000	1.000000	0.000000	6.000000	5.000000	2018.000000
50%	1.887286e+07	2.837092e+07	40.721710	-73.954810	101.000000	2.000000	9.000000	2019-05-19 00:00:00	0.720000	1.000000	55.000000	17.000000	6.000000	2019.000000
75%	2.756746e+07	1.018905e+08	40.762990	-73.935020	170.000000	4.000000	33.000000	2019-06-23 00:00:00	2.020000	2.000000	229.000000	24.000000	7.000000	2019.000000
max	3.645581e+07	2.738417e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	2019-07-08 00:00:00	58.500000	327.000000	365.000000	31.000000	12.000000	2019.000000
std	1.069372e+07	7.589752e+07	0.054991	0.046693	196.994756	17.389026	48.182900	NaN	1.680328	26.302954	129.529950	9.881894	2.531374	1.180598

```python
# Check basic statistics of the dataset
data.describe()
```

✓ 0.0s Python

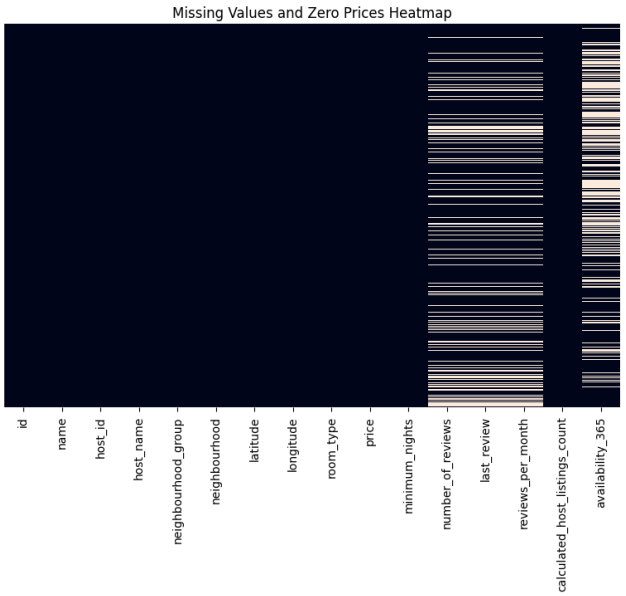
	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	rev
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	

Code to create missing zeroes heatmap:

```python
# Create a mask where True represents either null values or zero prices
mask = data.isnull() | (data == 0)

# For the price column specifically, mark zeros as True
mask['price'] = (data['price'] == 0) | data['price'].isnull()

plt.figure(figsize=(12, 6))
sns.heatmap(mask, yticklabels=False, cbar=True)
plt.title('Missing Values and Zero Prices Heatmap')
plt.show()

# Print summary
print("\nSummary of problematic values:")
print("-" * 50)
print("\nNull values count:")
print(data.isnull().sum())
print("\nZero values count in price:", len(data[data['price'] == 0]))
print("\nTotal problematic values in price:",
      len(data[data['price'].isnull() | (data['price'] == 0)]))
```

✓ 0.4s Python

Visualization:




```
Summary of problematic values:
----------------------------------------------------

Null values count:
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

Zero values count in price: 11

Total problematic values in price: 11
```

Pre-processing function:

```python
def preprocess_airbnb_data(data):
    # Create a copy of the data to avoid modifying the original
    df = data.copy()

    # 1. Handle price outliers and zero prices
    # Remove any zero prices as they're likely errors
    df = df[df['price'] > 0]

    # Cap prices at 99th percentile to handle extreme outliers
    price_99th = df['price'].quantile(0.99)
    df['price'] = df['price'].clip(upper=price_99th)

    # 2. Drop rows with null values in last_review and reviews_per_month
    df = df.dropna(subset=['last_review', 'reviews_per_month'])

    # 3. Drop unnecessary columns
    columns_to_drop = ['id', 'name', 'host_name']
    df = df.drop(columns=columns_to_drop)

    return df

# Apply preprocessing
processed_data = preprocess_airbnb_data(data)
```

✓ 0.0s Python

Info on the pre-processed data:

```
processed_data.info()
```

✓ 0.0s Python

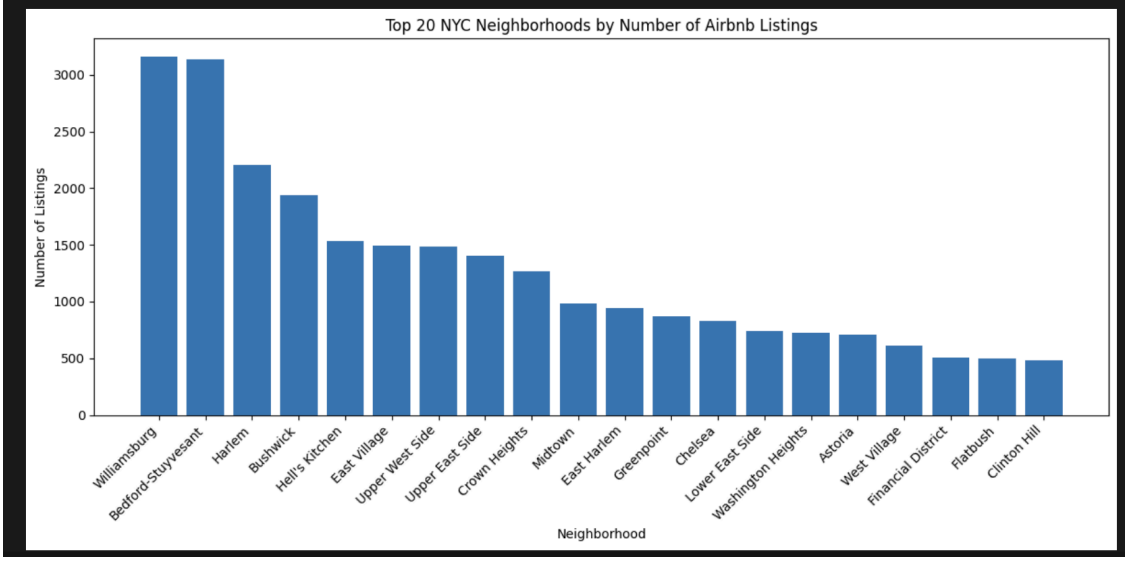
```
<class 'pandas.core.frame.DataFrame'>
Index: 38833 entries, 0 to 48852
Data columns (total 13 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   host_id                         38833 non-null  int64
 1   neighbourhood_group             38833 non-null  object
 2   neighbourhood                   38833 non-null  object
 3   latitude                        38833 non-null  float64
 4   longitude                       38833 non-null  float64
 5   room_type                       38833 non-null  object
 6   price                           38833 non-null  int64
 7   minimum_nights                  38833 non-null  int64
 8   number_of_reviews               38833 non-null  int64
 9   last_review                     38833 non-null  object
 10  reviews_per_month               38833 non-null  float64
 11  calculated_host_listings_count  38833 non-null  int64
 12  availability_365                38833 non-null  int64
dtypes: float64(3), int64(6), object(4)
memory usage: 4.1+ MB
```

EDA

Code and visualization of top 20 NYC Neighborhoods by number of listings.

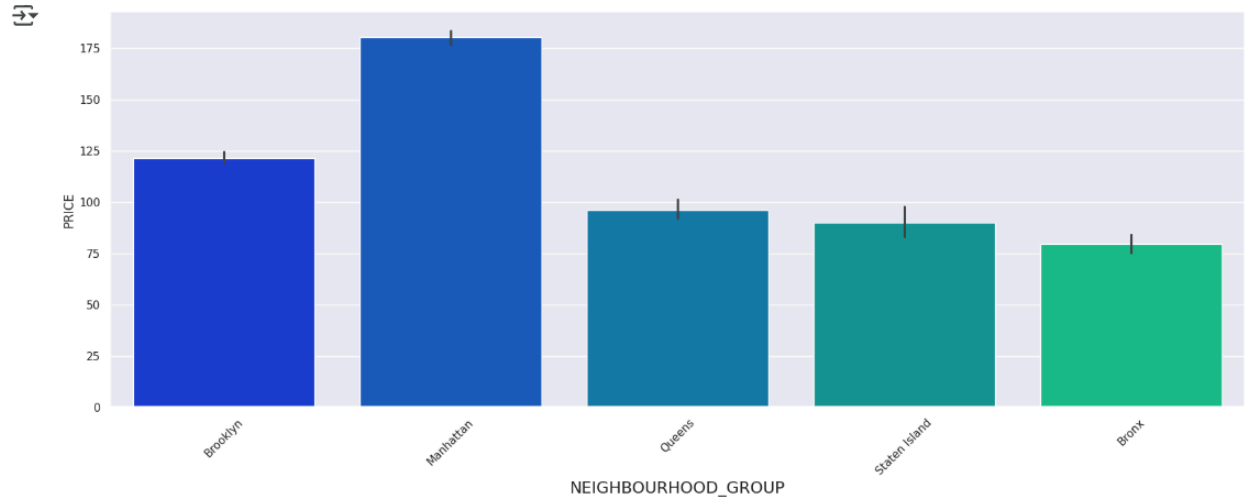
```python
# Count listings per neighborhood
neighborhood_counts = processed_data['neighbourhood'].value_counts()

# Show only top 20 neighborhoods to avoid overcrowded histogram
top_20_neighborhoods = neighborhood_counts.head(20)

plt.figure(figsize=(12, 6))
plt.bar(top_20_neighborhoods.index, top_20_neighborhoods.values)
plt.xticks(rotation=45, ha='right')
plt.title('Top 20 NYC Neighborhoods by Number of Airbnb Listings')
plt.xlabel('Neighborhood')
plt.ylabel('Number of Listings')
plt.tight_layout()
plt.show()

# Show the plot
plt.show()
```

✓ 0.1s Pytho



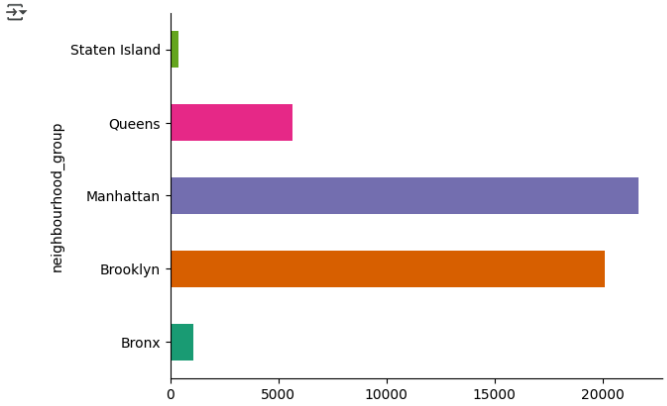
Code and visualization of Neighborhood groups by number of listings.

```
[ ]  plt.figure(figsize=(20,7))
     sns.barplot(x = data["neighbourhood_group"], y = data["price"], palette = "winter")
     #sns.barplot(x = data["neighbourhood_group"], y = data["price"], color = "blue")
     #sns.scatterplot(x = data["neighbourhood_group"], y = data["price"], color = "crimson")
     plt.xlabel("NEIGHBOURHOOD_GROUP", fontsize = 16)
     plt.ylabel("PRICE", fontsize = 12)
     plt.xticks(rotation = 45)
     plt.show()
```



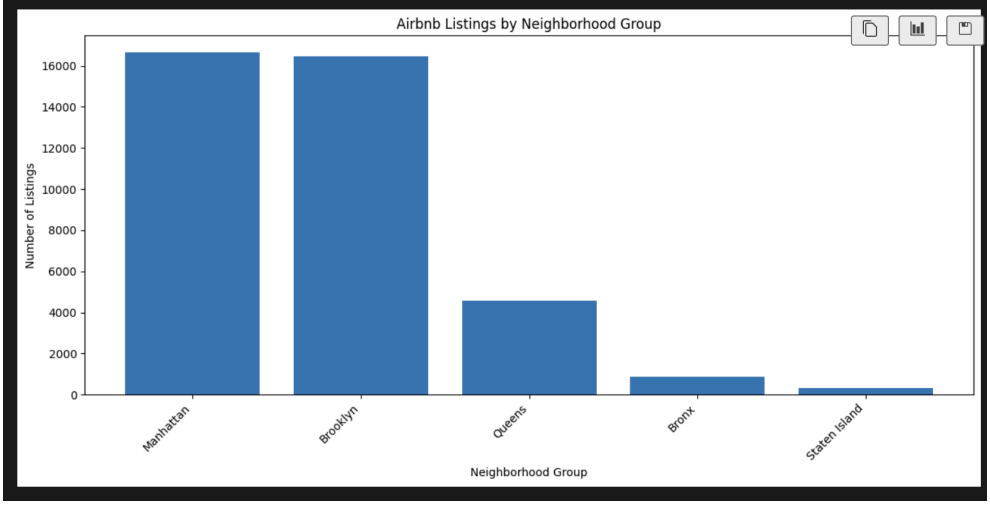
neighbourhood_group

```
# @title neighbourhood_group

from matplotlib import pyplot as plt
import seaborn as sns
data.groupby('neighbourhood_group').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



```
neighborhood_groupcounts = processed_data['neighbourhood_group'].value_counts()
plt.figure(figsize=(12, 6))
plt.bar(neighborhood_groupcounts.index, neighborhood_groupcounts.values)
plt.xticks(rotation=45, ha='right')
plt.title('Airbnb Listings by Neighborhood Group')
plt.xlabel('Neighborhood Group')
plt.ylabel('Number of Listings')
plt.tight_layout()
plt.show()

# Show the plot
plt.show()
```

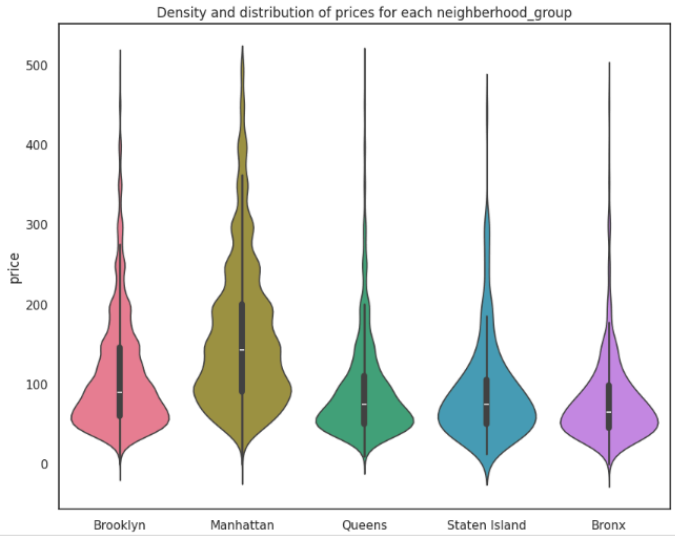
✓ 0.0s Pytho



Code and visualization of price distribution per neighborhood group:

```
# prompt: generate the sam violin plot but with different colours for each nrighbourhood

# Use a color palette with distinct colors for each neighborhood group
palette = sns.color_palette("husl", len(sub_6['neighbourhood_group'].unique()))

viz_2 = sns.violinplot(data=sub_6, x='neighbourhood_group', y='price', palette=palette)
viz_2.set_title('Density and distribution of prices for each neighberhood_group')
```

```
<ipython-input-33-d1697328d79e>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  viz_2 = sns.violinplot(data=sub_6, x='neighbourhood_group', y='price', palette=palette)
Text(0.5, 1.0, 'Density and distribution of prices for each neighberhood_group')
```



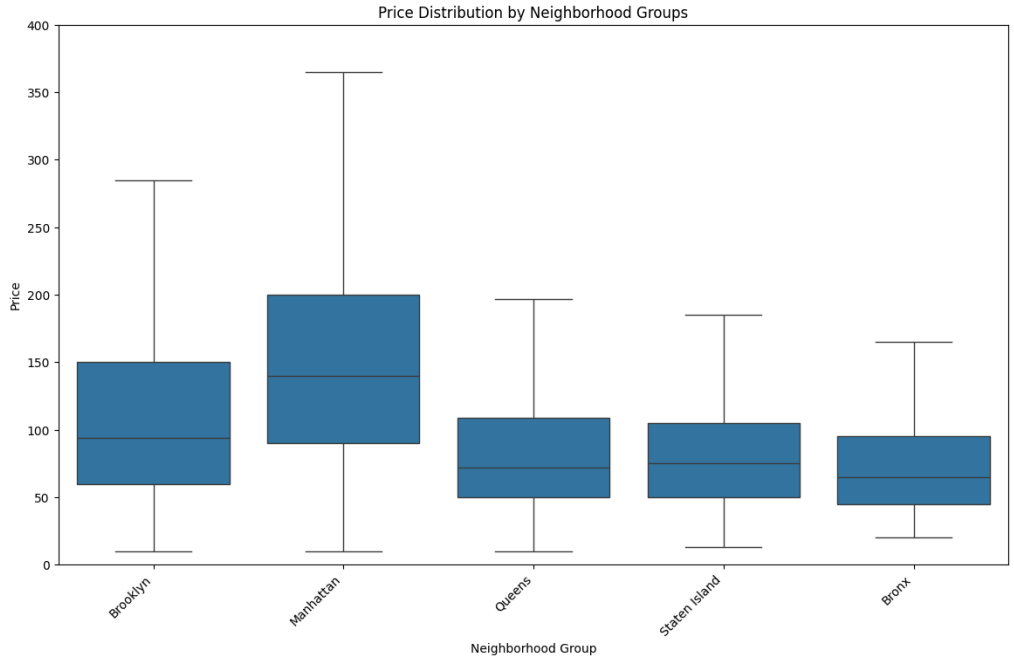
```
[ ] import holoviews as hv
    import geoviews as gv
    import datashader as ds
    from colorcet import fire, rainbow, bgy, bjy, bkr, kb, kr
    from datashader.colors import colormap_select, Greys9
    from holoviews.streams import RangeXY
    from holoviews.operation.datashader import datashade, dynspread, rasterize
    from bokeh.io import push_notebook, show, output_notebook
    output_notebook()
    hv.extension('bokeh')
```

```
[ ] NewYork = data
    agg_name = "price"
    NewYork[agg_name].describe().to_frame()
```

	price
count	38821.000000
mean	142.332526
std	196.994756
min	0.000000
25%	69.000000
50%	101.000000
75%	170.000000
max	10000.000000

```python
# Create figure with larger size
plt.figure(figsize=(12, 8))

# Create boxplot without outliers
sns.boxplot(data=processed_data,
            x='neighbourhood_group',
            y='price',
            showfliers=False)

# Get 97th percentile of prices for y-axis limit
y_limit = processed_data['price'].quantile(0.97)
plt.ylim(0, y_limit)

# Rotate x-axis labels and adjust their position
plt.xticks(rotation=45, ha='right')
plt.title('Price Distribution by Neighborhood Groups')
plt.xlabel('Neighborhood Group')
plt.ylabel('Price')

# Adjust layout with specific margins
plt.tight_layout(pad=2.0)
plt.show()

# Print summary statistics for neighborhood groups
print("\nSummary statistics for neighborhood groups:")
print(data.groupby('neighbourhood_group')
      .agg({
          'price': ['count', 'median', 'mean', 'std'],
          'room_type': 'count'
      })
      .round(2))
```

✓ 0.1s Python



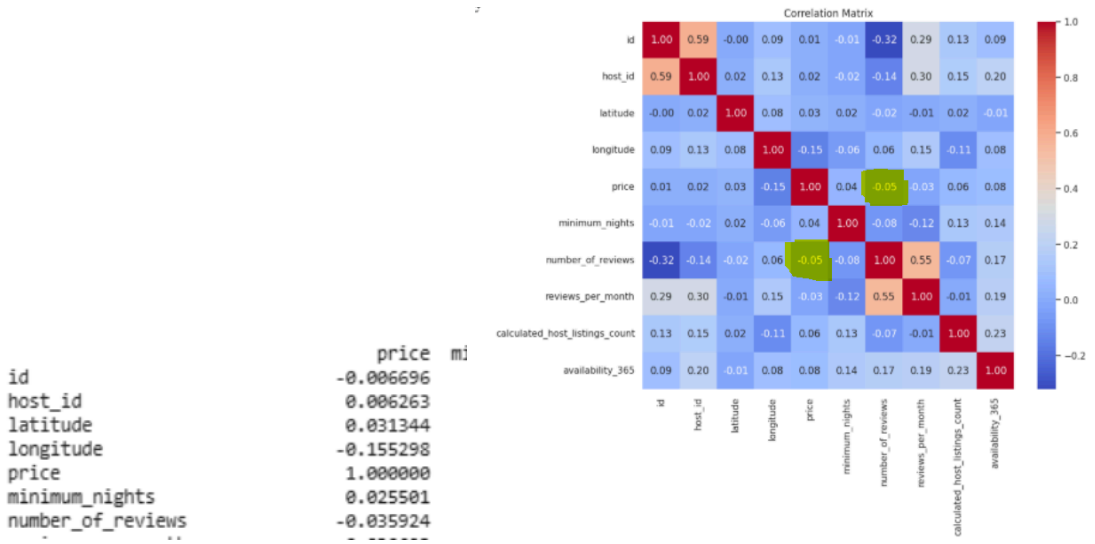
Summary statistics for neighborhood groups:

	price				room_type
	count	median	mean	std	count
neighbourhood_group					
Bronx	1091	65.0	87.50	106.71	1091
Brooklyn	20104	90.0	124.38	186.87	20104
Manhattan	21661	150.0	196.88	291.38	21661
Queens	5666	75.0	99.52	167.10	5666
Staten Island	373	75.0	114.81	277.62	373

Calculation of the correlation matrix:

```python
# Calculate the correlation matrix
correlation_matrix = numerical_data.corr()

# Display the correlation matrix
print(correlation_matrix)
```

```
                                     id   host_id  latitude  longitude  \
id                             1.000000  0.591480 -0.010085   0.103336
host_id                        0.591480  1.000000  0.012823   0.141276
latitude                      -0.010085  0.012823  1.000000   0.088259
longitude                      0.103336  0.141276  0.088259   1.000000
price                         -0.006696  0.006263  0.031344  -0.155298
minimum_nights                -0.073937 -0.051673  0.024893  -0.055414
number_of_reviews             -0.330019 -0.141973 -0.008559   0.054746
reviews_per_month              0.291786  0.296274 -0.010117   0.146228
calculated_host_listings_count  0.098446  0.149417  0.004340  -0.093348
availability_365               0.006624  0.155361 -0.021921   0.102573
last_review_day               -0.018368 -0.014804  0.000112  -0.013533
last_review_month             -0.051582 -0.030352 -0.002249   0.000573
last_review_year               0.407245  0.265914 -0.020545   0.101605

                                   price  minimum_nights  number_of_reviews  \
id                            -0.006696       -0.073937          -0.330019
host_id                        0.006263       -0.051673          -0.141973
latitude                       0.031344        0.024893          -0.008559
longitude                     -0.155298       -0.055414           0.054746
price                          1.000000        0.025501          -0.035924
minimum_nights                 0.025501        1.000000          -0.069366
number_of_reviews             -0.035924       -0.069366           1.000000
reviews_per_month             -0.030623       -0.121712           0.549699
calculated_host_listings_count  0.052895        0.073474          -0.059796
availability_365               0.078276        0.101658           0.193409
last_review_day               -0.007274        0.003170           0.025769
last_review_month              0.003720       -0.013313          -0.005288
last_review_year              -0.016673       -0.048504           0.272038

                              reviews_per_month  \
id                                     0.291786
host_id                                0.296274
latitude                              -0.010117
longitude                              0.146228
price                                 -0.030623
minimum_nights                        -0.121712
number_of_reviews                      0.549699
reviews_per_month                      1.000000
calculated_host_listings_count        -0.009442
availability_365                       0.185896
last_review_day                       -0.020906
last_review_month                      0.003225
last_review_year                       0.423909

                              calculated_host_listings_count  \
id                                                  0.098446
host_id                                             0.149417
latitude                                            0.004340
longitude                                          -0.093348
price                                               0.052895
minimum_nights                                      0.073474
number_of_reviews                                  -0.059796
```



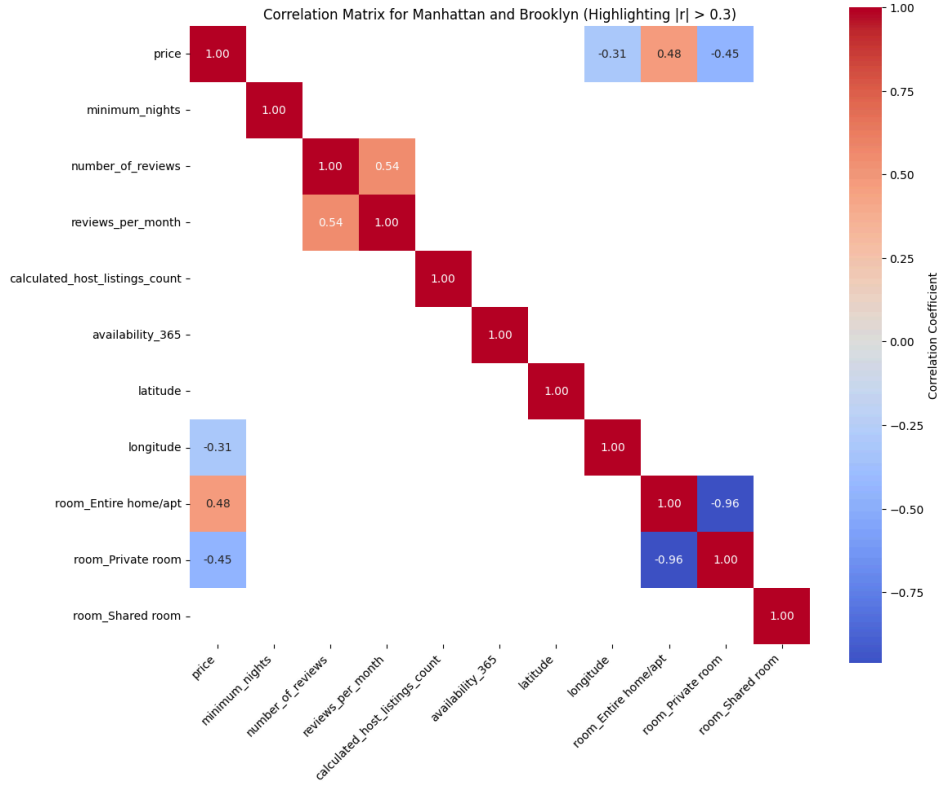

```
                                price  mi
id                          -0.006696
host_id                      0.006263
latitude                     0.031344
longitude                   -0.155298
price                        1.000000
minimum_nights               0.025501
number_of_reviews           -0.035924
```

Code of the Correlation matrix used for only for Brooklyn and Manhattan

```python
# Filter for Manhattan and Brooklyn
filtered_data = processed_data[processed_data['neighbourhood_group'].isin(['Manhattan', 'Brooklyn'])]

# Create one-hot encoded variables for room_type
room_type_dummies = pd.get_dummies(filtered_data['room_type'], prefix='room')

# Combine numerical columns with room type dummies
numerical_cols = ['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month',
                  'calculated_host_listings_count', 'availability_365', 'latitude', 'longitude']

# Combine all features
combined_data = pd.concat([filtered_data[numerical_cols], room_type_dummies], axis=1)

# Create correlation matrix
plt.figure(figsize=(12, 10))
correlation_matrix = combined_data.corr()

# Create mask for strong correlations (absolute value > 0.3)
strong_mask = np.abs(correlation_matrix) > 0.3

# Create heatmap
sns.heatmap(correlation_matrix,
            annot=True,
            cmap='coolwarm',
            center=0,
            fmt='.2f',
            square=True,
            mask=~strong_mask,  # Mask weak correlations
            cbar_kws={'label': 'Correlation Coefficient'})

plt.title('Correlation Matrix for Manhattan and Brooklyn (Highlighting |r| > 0.3)')
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()

# Print strongest correlations
print("\nStrongest correlations (|r| > 0.3):")
upper = correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape), k=1).astype(bool))
strong_correlations = upper[np.abs(upper) > 0.3]
print(strong_correlations.unstack().dropna().sort_values(key=abs, ascending=False))
```

✓ 0.2s Python

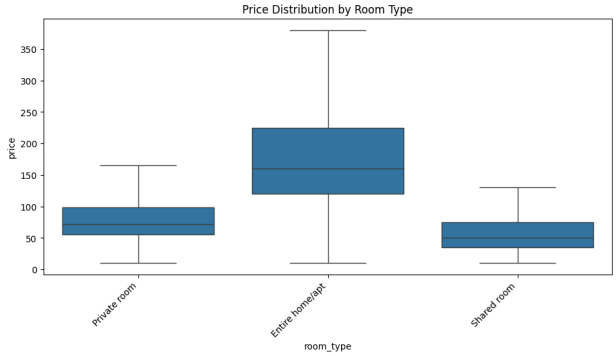
Visualisation:

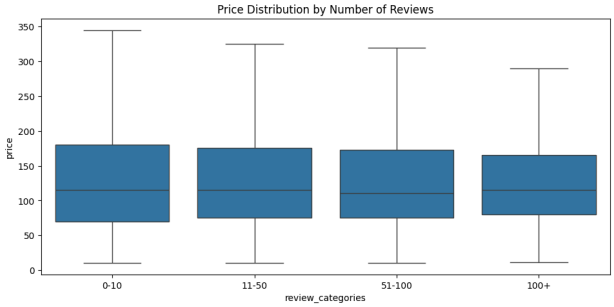


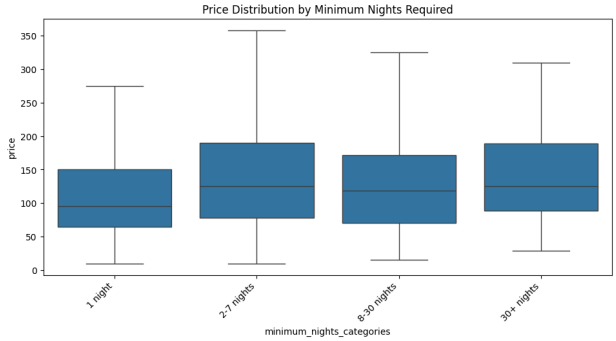
```
Strongest correlations (|r| > 0.3):
room_Private room     room_Entire home/apt   -0.961286
reviews_per_month     number_of_reviews       0.543811
room_Entire home/apt  price                   0.475979
room_Private room     price                  -0.450208
longitude             price                  -0.309007
dtype: float64
```

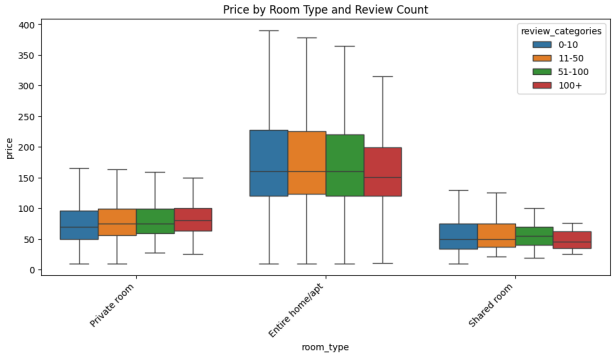
Code and visualization for Box plots comparing Price Distribution by Room Type, Number of Reviews, Minimum Nights Required and Room Type combined with Number of Reviews

```python
# Create a figure with multiple subplots
fig = plt.figure(figsize=(20, 12))

# 1. Room Type vs Price
plt.subplot(2, 2, 1)
sns.boxplot(data=filtered_data, x='room_type', y='price', showfliers=False)
plt.xticks(rotation=45, ha='right')
plt.title('Price Distribution by Room Type')

# 2. Number of Reviews vs Price
plt.subplot(2, 2, 2)
# Create categories for number of reviews
filtered_data['review_categories'] = pd.cut(filtered_data['number_of_reviews'],
                                            bins=[0, 10, 50, 100, float('inf')],
                                            labels=['0-10', '11-50', '51-100', '100+'])
sns.boxplot(data=filtered_data, x='review_categories', y='price', showfliers=False)
plt.title('Price Distribution by Number of Reviews')

# 3. Minimum Nights vs Price
plt.subplot(2, 2, 3)
# Create categories for minimum nights
filtered_data['minimum_nights_categories'] = pd.cut(filtered_data['minimum_nights'],
                                                    bins=[0, 1, 7, 30, float('inf')],
                                                    labels=['1 night', '2-7 nights', '8-30 nights', '30+ nights'])
sns.boxplot(data=filtered_data, x='minimum_nights_categories', y='price', showfliers=False)
plt.xticks(rotation=45, ha='right')
plt.title('Price Distribution by Minimum Nights Required')

# 4. Room Type and Review Count Combined
plt.subplot(2, 2, 4)
sns.boxplot(data=filtered_data, x='room_type', y='price', hue='review_categories', showfliers=False)
plt.xticks(rotation=45, ha='right')
plt.title('Price by Room Type and Review Count')

# Adjust layout
plt.tight_layout(pad=3.0)
plt.show()

# Print summary statistics
print("\nMedian Prices by Room Type:")
print(filtered_data.groupby('room_type')['price'].median().round(2))

print("\nMedian Prices by Review Category:")
print(filtered_data.groupby('review_categories')['price'].median().round(2))
```





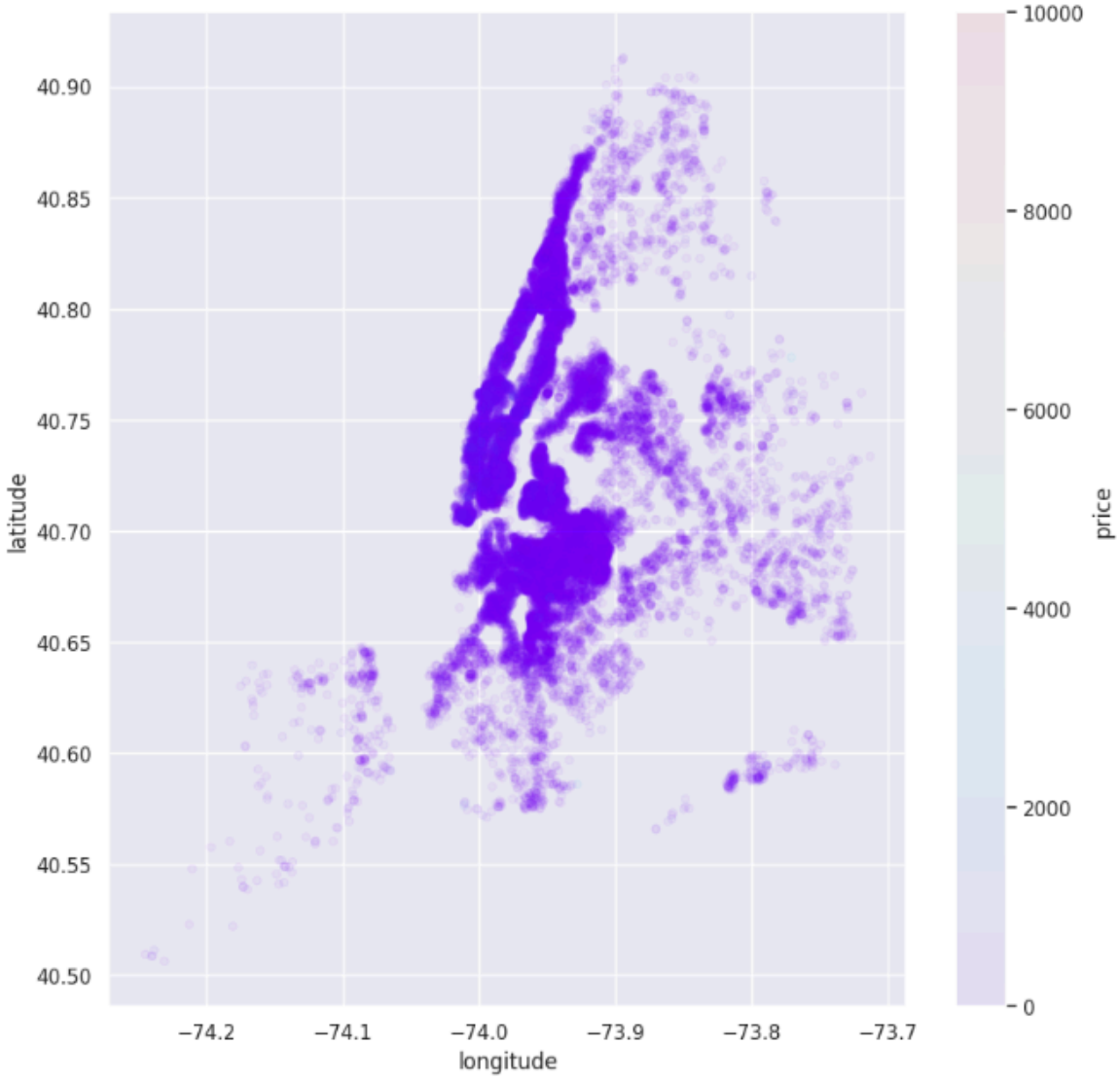





```
Median Prices by Room Type:
room_type
Entire home/apt    160.0
Private room        72.0
Shared room         50.0
Name: price, dtype: float64

Median Prices by Review Category:
review_categories
0-10      115.0
11-50     115.0
51-100    111.0
100+      115.0
Name: price, dtype: float64

Median Prices by Minimum Nights Category:
minimum_nights_categories
1 night         95.0
2-7 nights     125.0
8-30 nights    119.0
30+ nights     125.0
Name: price, dtype: float64
```

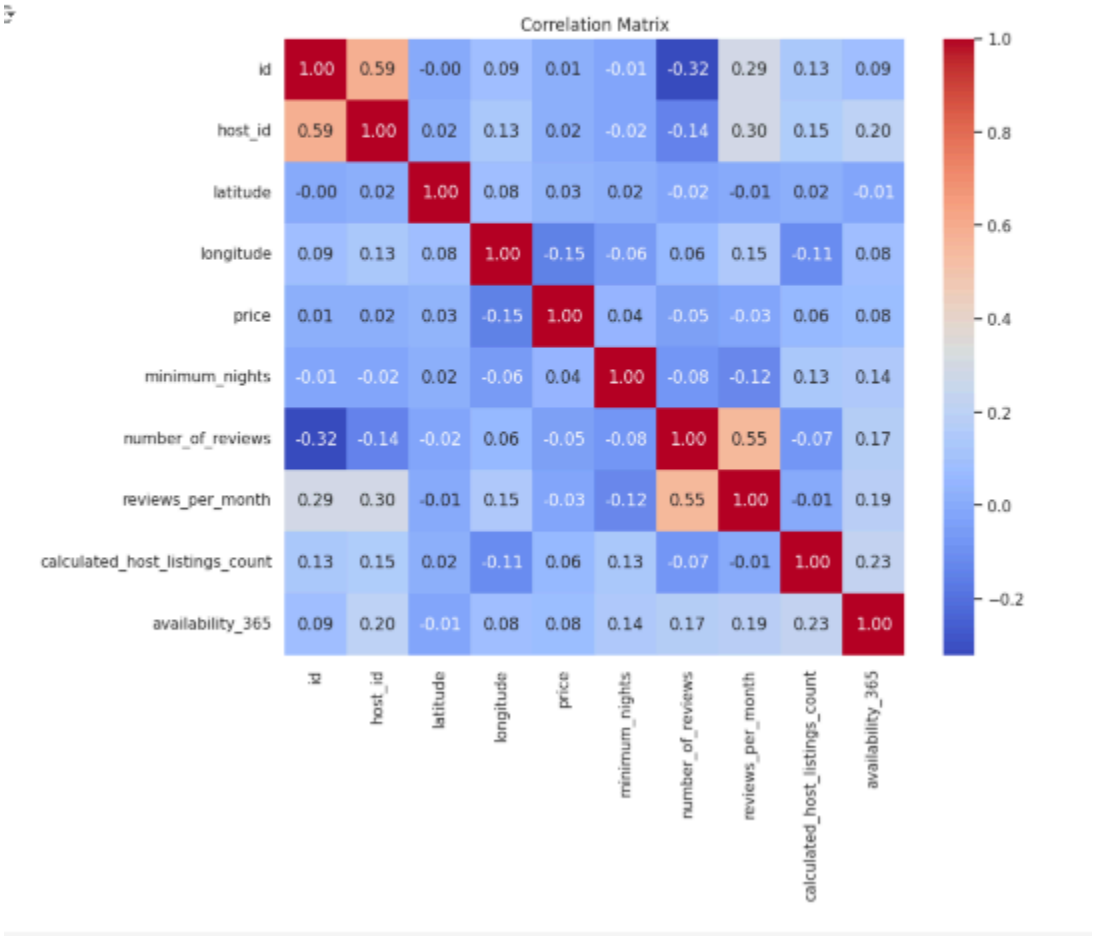
```
data.plot(kind = "scatter", x = "longitude", y = "latitude", alpha = 0.05,
          figsize = (10,10), c = "price", cmap = plt.get_cmap("rainbow"),
           colorbar = True
          )
```

```
<Axes: xlabel='longitude', ylabel='latitude'>
```



Data Analysis

```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming 'correlation_matrix' is already calculated as in your provided code

plt.figure(figsize=(10, 8))  # Adjust figure size as needed
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



```python
import statsmodels.api as sm

# Assuming 'data' DataFrame is already loaded and preprocessed as in your code.

# Define the independent and dependent variables
X = data['number_of_reviews']  # Independent variable
y = data['price']  # Dependent variable

# Add a constant to the independent variable (intercept term)
X = sm.add_constant(X)

# Fit the linear regression model
model = sm.OLS(y, X).fit()

# Print the regression results
print(model.summary())
```

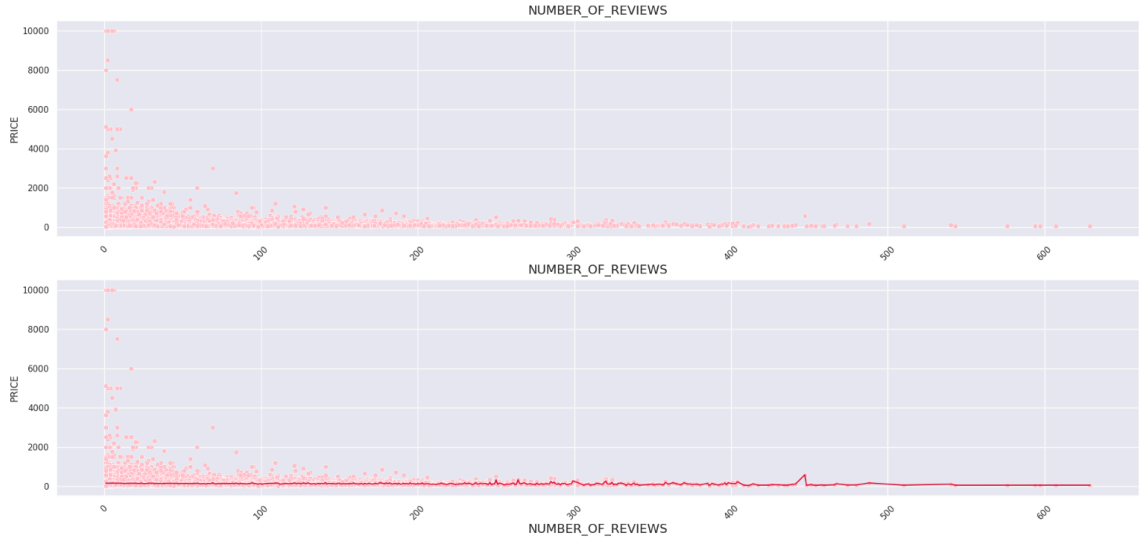
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     112.7
Date:                Sat, 01 Mar 2025   Prob (F-statistic):           2.69e-26
Time:                        17:30:41   Log-Likelihood:            -3.3733e+05
No. Observations:               48895   AIC:                         6.747e+05
Df Residuals:                   48893   BIC:                         6.747e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        158.7372      1.224    129.692      0.000     156.338     161.136
number_of_reviews    -0.2585      0.024    -10.616      0.000      -0.306      -0.211
==============================================================================
Omnibus:                   105110.317   Durbin-Watson:                   1.837
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        704079197.888
Skew:                          19.129   Prob(JB):                         0.00
Kurtosis:                     589.628   Cond. No.                         56.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Model Fit (R-squared = 0.002, Adjusted R-squared = 0.002)

```
# Assuming 'data' DataFrame is already loaded and preprocessed as in your code.
import statsmodels.api as sm
import pandas as pd # Import pandas for data manipulation

# Define the independent and dependent variables
X = data['neighbourhood_group']  # Independent variable
y = data['price']  # Dependent variable

# Convert the categorical variable 'neighbourhood_group' to numerical using dummy variables
X = pd.get_dummies(X, drop_first=True)

# Add a constant to the independent variables (for the intercept term)
X = sm.add_constant(X)

# Fit the linear regression model
# Ensure all columns in X are numeric before fitting the model
X = X.astype(float) # Convert all columns in X to float
model = sm.OLS(y, X).fit()

# Print the regression results
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.028
Model:                            OLS   Adj. R-squared:                  0.028
Method:                 Least Squares   F-statistic:                     355.0
Date:                Sat, 01 Mar 2025   Prob (F-statistic):          7.72e-302
Time:                        17:30:49   Log-Likelihood:            -3.3669e+05
No. Observations:               48895   AIC:                         6.734e+05
Df Residuals:                   48890   BIC:                         6.734e+05
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            87.4968      7.168     12.207      0.000      73.448     101.546
Brooklyn         36.8864      7.360      5.012      0.000      22.462      51.311
Manhattan       109.3790      7.346     14.890      0.000      94.981     123.777
Queens           12.0209      7.827      1.536      0.125      -3.321      27.363
Staten Island    27.3155     14.200      1.924      0.054      -0.517      55.148
==============================================================================
Omnibus:                   106502.529   Durbin-Watson:                   1.839
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        771576005.180
Skew:                          19.721   Prob(JB):                         0.00
Kurtosis:                     617.142   Cond. No.                         19.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

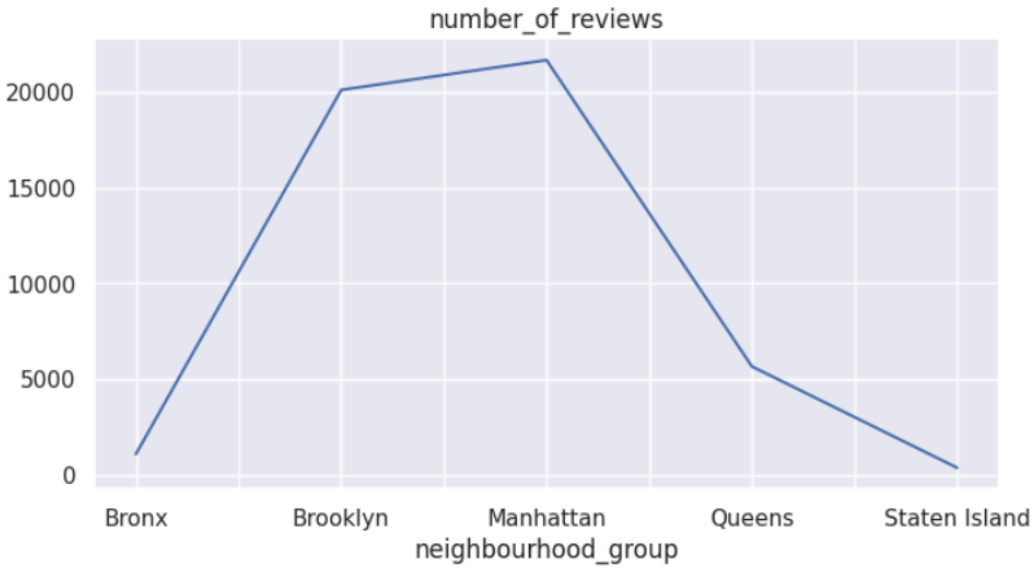
```
plt.figure(figsize=(25,17))

plt.subplot(3,1,1)
sns.lineplot(x = data["number_of_reviews"], y = data["price"], color = "crimson")
#sns.scatterplot(x = data["number_of_reviews"], y = data["price"], color = "purple")
plt.xlabel("NUMBER_OF_REVIEWS", fontsize = 16)
plt.ylabel("PRICE", fontsize = 12)
plt.xticks(rotation = 45)

plt.subplot(3,1,2)
#sns.lineplot(x = data["number_of_reviews"], y = data["price"], color = "crimson")
sns.scatterplot(x = data["number_of_reviews"], y = data["price"], color = "pink")
plt.xlabel("NUMBER_OF_REVIEWS", fontsize = 16)
plt.ylabel("PRICE", fontsize = 12)
plt.xticks(rotation = 45)

plt.subplot(3,1,3)
sns.lineplot(x = data["number_of_reviews"], y = data["price"], color = "crimson")
sns.scatterplot(x = data["number_of_reviews"], y = data["price"], color = "pink")
plt.xlabel("NUMBER_OF_REVIEWS", fontsize = 16)
plt.ylabel("PRICE", fontsize = 12)
plt.xticks(rotation = 45)
plt.show()
```



```python
# prompt: plt.subplot(3,1,2)
# #sns.lineplot(x = data["number_of_reviews"], y = data["neighbourhood_group"], color = "crimson")
# sns.scatterplot(x = data["number_of_reviews"], y = data["neighbourhood_group"], color = "red")
# plt.xlabel("NUMBER_OF_REVIEWS", fontsize = 16)
# plt.ylabel("NEIGHBOURHOOD GROUP", fontsize = 12)
# plt.xticks(rotation = 45) provide a table not a plot

import pandas as pd

# Assuming 'data' DataFrame is already loaded as in the provided code

# Create the table
table_data = pd.pivot_table(data, values='number_of_reviews', index='neighbourhood_group', aggfunc='count')

# Display the table
table_data
```

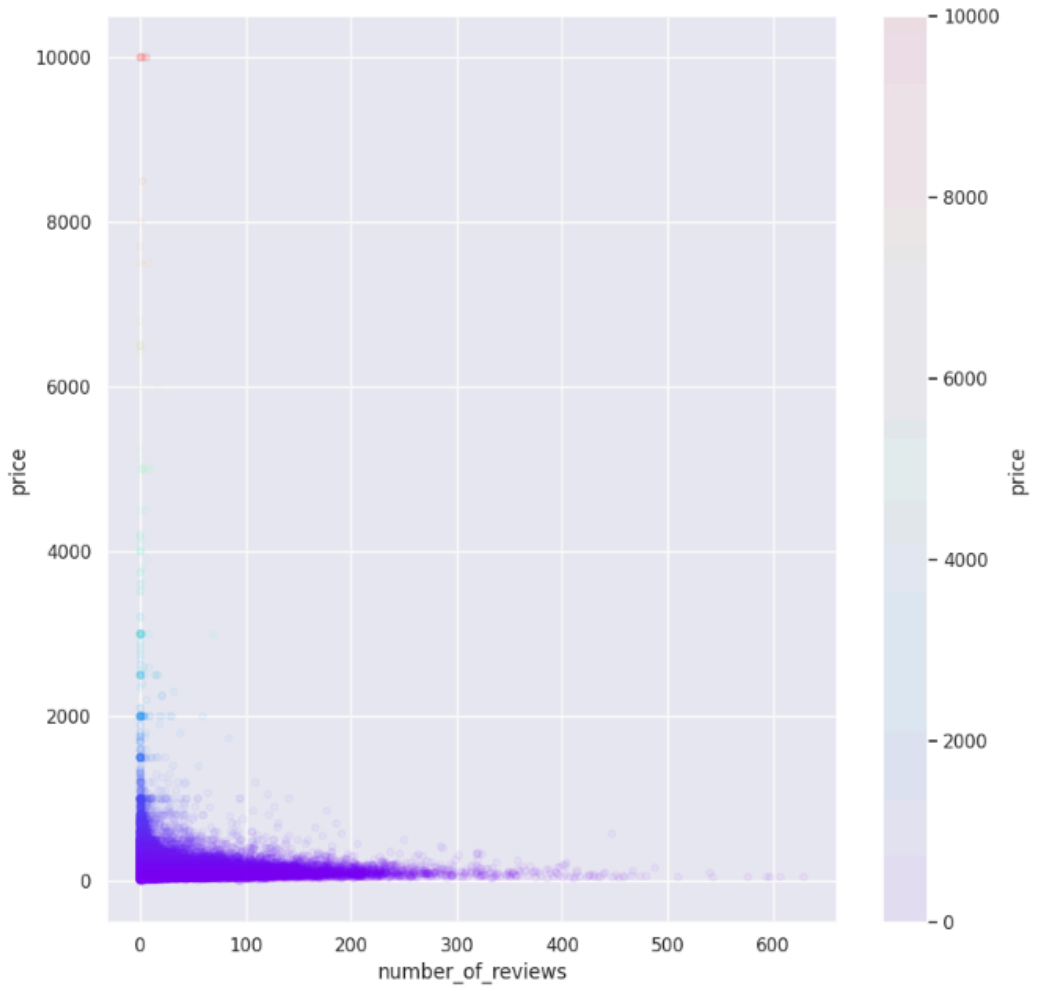
neighbourhood_group	number_of_reviews
Bronx	1091
Brooklyn	20104
Manhattan	21661
Queens	5666
Staten Island	373

```python
from matplotlib import pyplot as plt
table_data['number_of_reviews'].plot(kind='line', figsize=(8, 4), title='number_of_reviews')
plt.gca().spines[['top', 'right']].set_visible(False)
```



```
data.plot(kind = "scatter", x = "number_of_reviews", y = "price", alpha = 0.05,
          figsize = (10,10), c = "price", cmap = plt.get_cmap("rainbow"),
          colorbar = True)
```

```
<Axes: xlabel='number_of_reviews', ylabel='price'>
```



```
# Define X (independent variables) and Y (dependent variable)
X = df_eda[[ 'number_of_reviews',
            'reviews_per_month','neighbourhood_group_Brooklyn',
            'neighbourhood_group_Manhattan', 'neighbourhood_group_Queens',
            'neighbourhood_group_Staten Island']]
y = df_eda['price']

# split dataset into training and testing portions (80%/20% split)
#for machine learning model evaluation.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

#evaluate model performance predcict test data
y_pred = model.predict(X_test)

# get coefficients
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)

# Evaluate model performance
```

✓ 0s completed at 1:07 AM

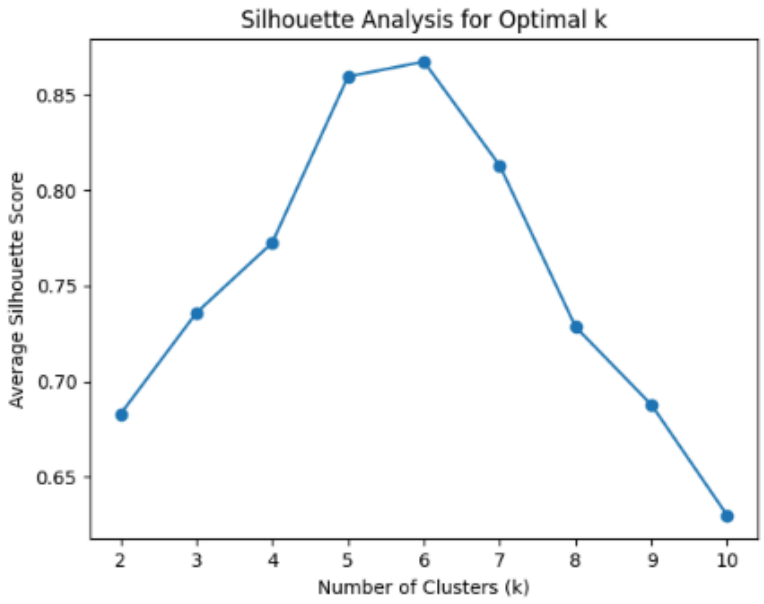
```
#create and train the model
model = LinearRegression()
model.fit(X_train, y_train)

#evaluate model performance predcict test data
y_pred = model.predict(X_test)

# get coefficients
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)

# Evaluate model performance
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
print(f'Root Mean Squared Error (RMSE): {rmse}')
print(f'R-squared Score: {r2}')
```

Unsupervised learning-Clustering

Silhouette Analysis

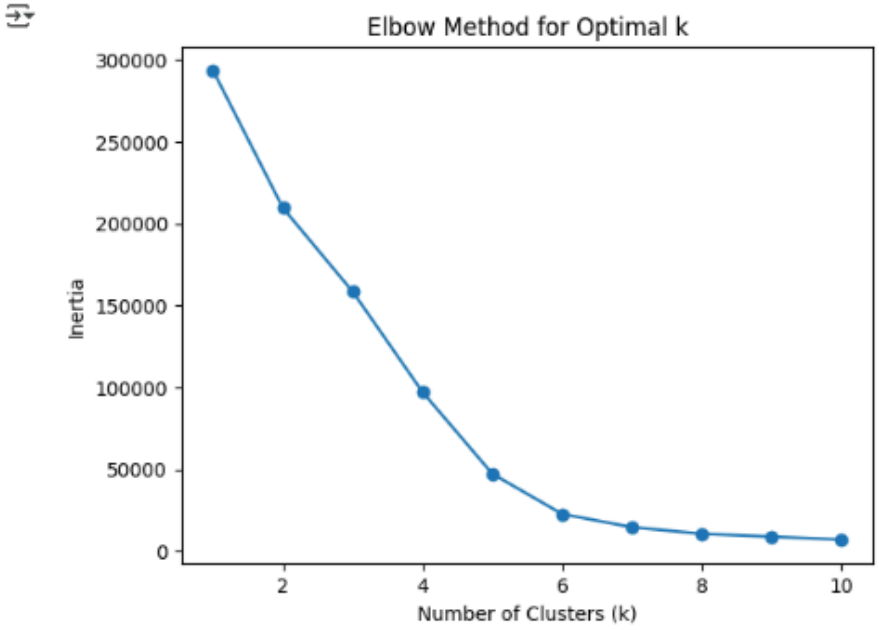
```
from sklearn.metrics import silhouette_score

# ... (previous code)

# Silhouette analysis
silhouette_avg = []
for k in range(2, 11):  # Test k from 2 to 10 clusters
    kmeans = KMeans(n_clusters=k, random_state=0, n_init=10) # Increased n_init
    cluster_labels = kmeans.fit_predict(X_scaled[:1000]) # Subset for silhouette
    silhouette_avg.append(silhouette_score(X_scaled[:1000], cluster_labels))

plt.plot(range(2, 11), silhouette_avg, marker='o')
plt.title('Silhouette Analysis for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Average Silhouette Score')
plt.show()

# ... (rest of the code)
```

[2]



Elbow Method

```
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.show()
```



References:

Carvache-Franco, M., Regalado-Pezúa, O., Sirkis, G., Carvache-Franco, O., & Carvache-Franco, W. (2023). Market segmentation in urban tourism: A study in Latin America. *PLOS ONE, 18*(5), e0285138. https://doi.org/10.1371/journal.pone.0285138 (Accessed: 7 March 2025).

Chatterjee, S., & Hadi, A. S. (2006). *Regression Analysis by Example* (4th ed.). Hoboken, NJ: Wiley-Interscience.

Coles, P. A., Egesdal, M., Ellen, I. G., & Sundararajan, A. (2017). Airbnb usage across New York City neighborhoods: Geographic patterns and regulatory implications. *Cambridge Handbook on the Law of the Sharing Economy*. https://doi.org/10.2139/ssrn.3048397

Field, A. P. (2024). *Discovering statistics using IBM SPSS statistics* (6th ed.). Los Angeles: Sage.

Jaureguizar Cervera, D., Pérez-Bustamante Yábar, D.C., & de Esteban Curiel, J. (2022). Factors affecting short-term rental first price: A revenue management model. *Frontiers in Psychology, 13*, article 994910. https://doi.org/10.3389/fpsyg.2022.994910 (Accessed: 7 March 2025).

Jiao, J., & Bai, S. (2019). Cities reshaped by Airbnb: A case study in New York City, Chicago, and Los Angeles. *Environment and Planning A: Economy and Space, 52*(1), 10–13. https://doi.org/10.1177/0308518x19853275

Martinez, R.D., Carrington, A., Kuo, T., Tarhuni, L., & Abdel-Motaal, N.A.Z. (2017). The

impact of an Airbnb host's listing description "sentiment" and length on occupancy rates. *arXiv preprint arXiv:1711.09196* [cs.CY]. https://arxiv.org/abs/1711.09196 (Accessed: 7 March 2025).

Nivón, T. (2019). Data Analysis on Airbnb in NYC. NYC Data Science Academy Blog, Dec 2, 2019. (Analysis of Inside Airbnb NYC September 2019 data). Available at: nycdatascience.com.

Perez-Sanchez, V.R., Serrano-Estrada, L., Marti, P., & Mora-Garcia, R-T. (2018). The what, where, and why of Airbnb price determinants. *Sustainability, 10*(12), 4596. https://doi.org/10.3390/su10124596

Schroeder, L. D., Sjoquist, D. L., & Stephan, P. E. (2017). *Understanding regression analysis: An introductory guide* (2nd ed.). Los Angeles, CA: SAGE Publications, Inc.

Sharma, U., & Gupta, D. (2021). Analyzing the applications of the internet of things in the hotel industry. *Journal of Physics: Conference Series*. https://doi.org/10.1088/1742-6596%2F1969%2F1%2F012041

Toader, V., Negruşa, A.L., Bode, O.R., & Rus, R.V. (2022). Analysis of price determinants in the case of Airbnb listings. *Economic Research-Ekonomska Istraživanja, 35*(1), 2487–2509.

Wedel, M., & Kamakura, W.A. (2000). *Market segmentation: Conceptual and methodological foundations* (2nd ed.). Boston: Kluwer Academic Publishers.