DEVELOPMENT TEAM PROJECT: AIRBNB NEW YORK CITY 2019 ANALYTIC REPORT

Maria Ingold, Murthy Kanuri, Khoi Dang (Danty), Ahmed Husain
Unit 6
Machine Learning
University of Essex Online
2 December 2024

CONTENTS

INTRODUCTION

Founded in 2008, Airbnb has generated $250 billion revenue from five million hosts (Airbnb, N.D.a; Ding et al., 2023). Despite revenue increases (Figure 1), year-over-year (Y/Y) growth declined below pre-COVID levels, prompting further Artificial Intelligence (AI) research (Airbnb, N.D.b).

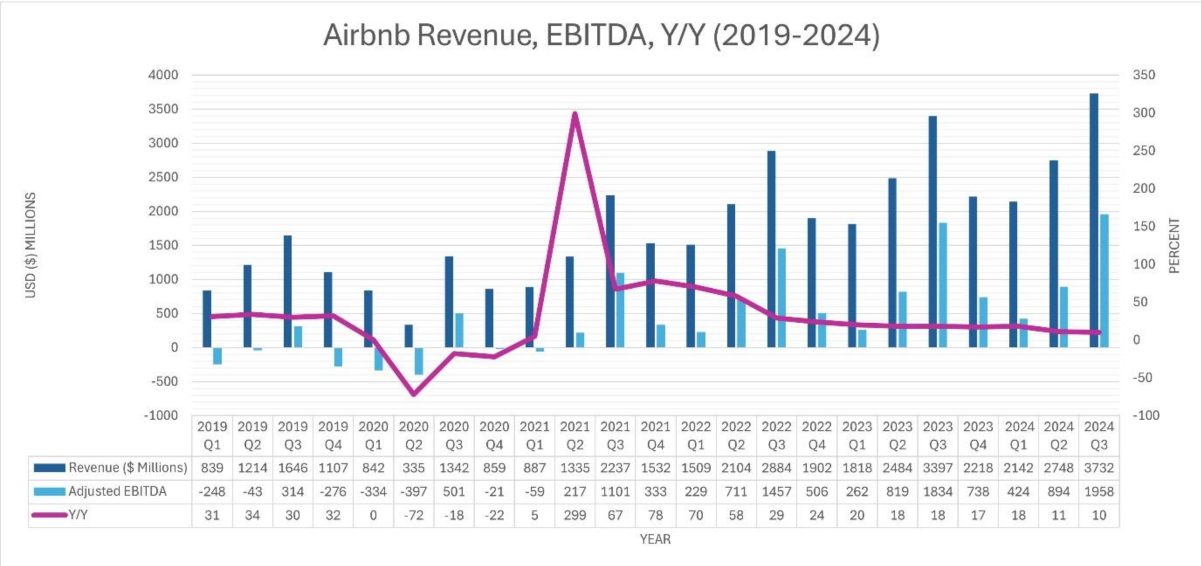


FIGURE 1 | Airbnb

This report applies Cross Industry Standard Process for Data Mining (CRISP-DM) (Figure 2) to the 2019 New York City (NYC) Airbnb dataset (Niakšu, 2015; Kaggle, 2021). Through exploratory data analysis (EDA) and unsupervised machine learning (ML), it visualises trends and generates pricing recommendations to optimise revenue.

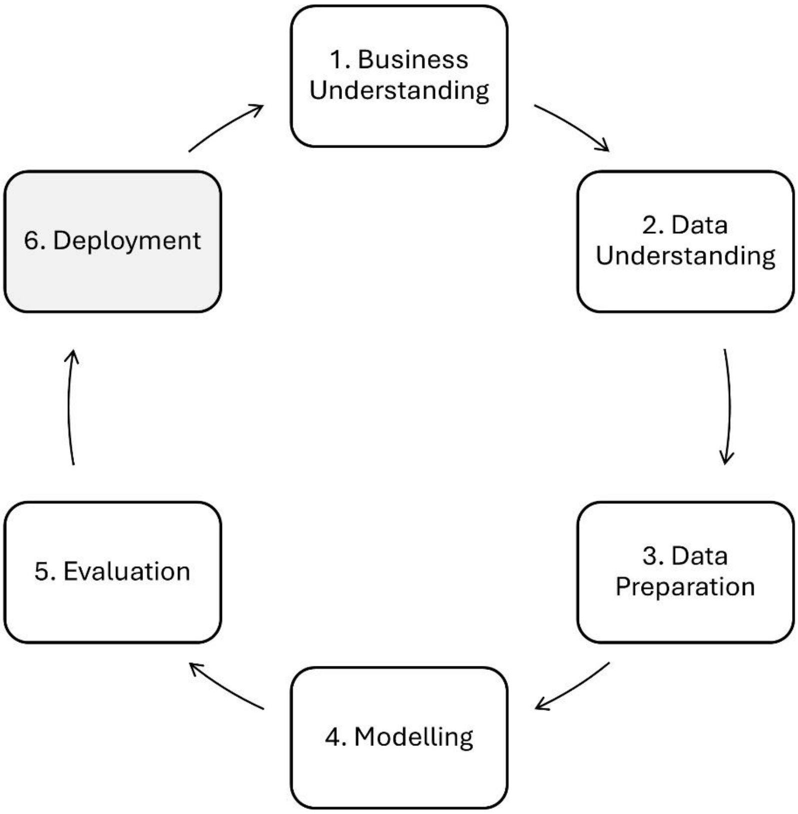


FIGURE 2 | CRISP-DM

RATIONALE

Quarterly year-over-year revenue growth (Figure 1) reflects seasonal trends and company health (Taylor & Almeida, 2022). NYC's price-inelasticity enables raising prices to increase revenue, however, market dynamics need consideration (Gunter et al., 2020; Akalın & Alptekin, 2024). Therefore, this study addresses: "*How can Airbnb NYC optimise revenue growth through tailored pricing recommendations based on location, room type, and host characteristics?*"

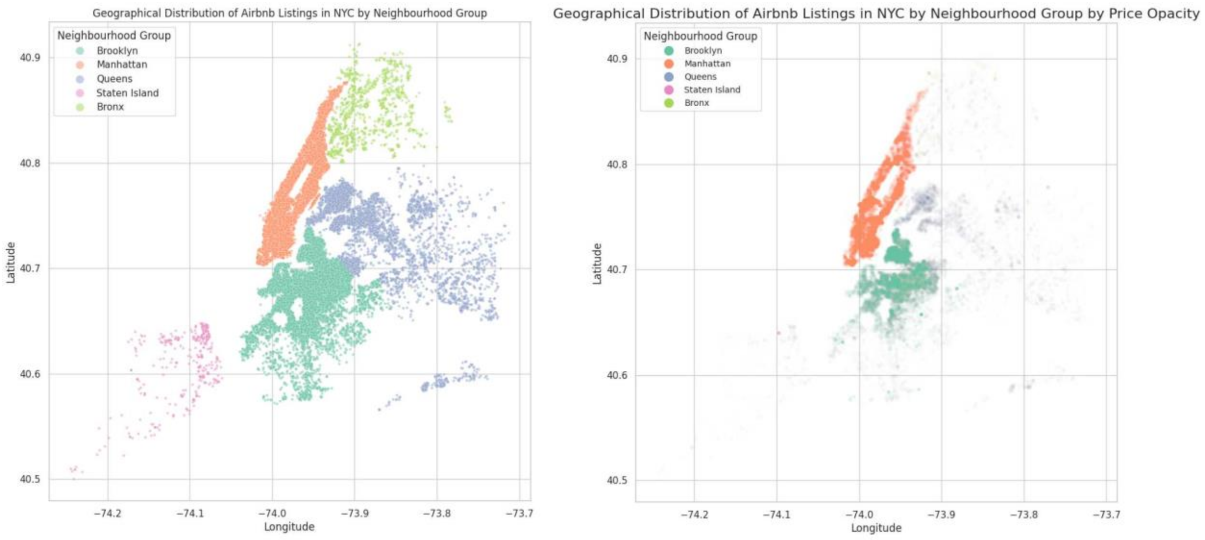


FIGURE 3 | Airbnb listing neighbourhoods and highest price densities

Location affects pricing (Figure 3), including proximity to amenities (Akalın & Alptekin, 2024). Room type influences market positioning, impacting competition and regulation (Gunter et al., 2020). Host characteristics like listing number, availability, and reviews potentially reveal professional hosts targeted for regulation (Table 1).

TABLE 1 | Potential price influencers

Location	• General: `neighbourhood_group` • Specific: `neighbourhood`, `latitude`, `longitude`
Room Type	• `room_type`: private, shared, entire
Host Characteristics	• `calculated_host_listings_count` • `minimum_nights` • `availability_365` • `number_of_reviews`, `reviews_per_month`, `last_review`

DATA UNDERSTANDING AND PREPARATION

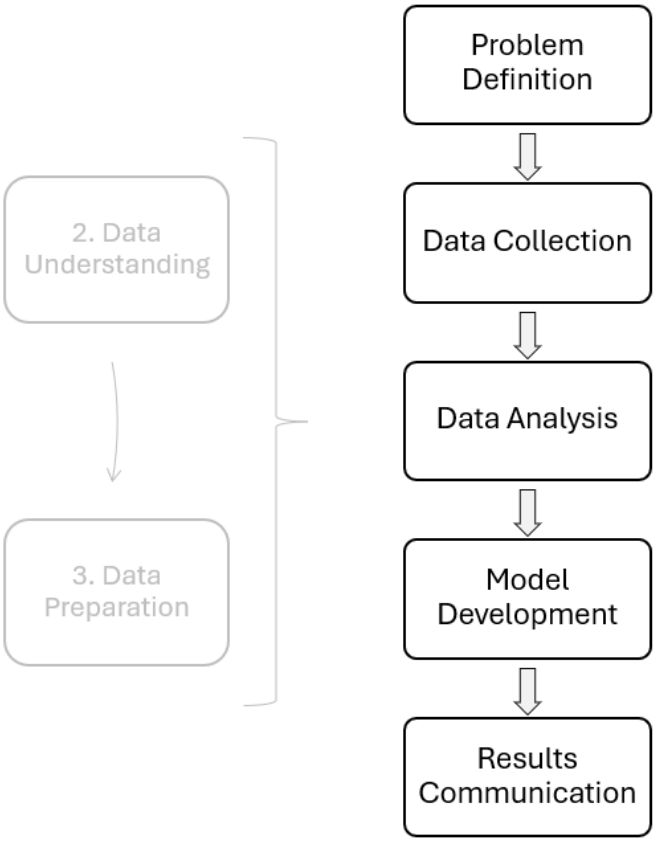


FIGURE 4 | EDA

EDA follows CRISP-DM (Figure 4) with data understanding identifying missing values (Table 2). (Mukhiya & Ahmed, 2020; Oluleye, 2023). Data preparation set missing `reviews_per_month` to 0 and imputed `last_review` with earliest date and adding a `has_review` flag. Non-sensitive Personally Identifiable Information (PII) `host_name` and `name` were removed (IBM, N.D.).

TABLE 2 | Missing data

Count	48,895 entries	
Columns	16 columns	10 numeric 6 categorical
Missing data	4 columns	`last_review` 10052 `reviews_per_month` 10052 `host_name` 21 `name` 16

Statistical analysis indicated non-normal distribution, confirmed by skew, kurtosis, and boxplots (Figure 5) (Mukhiya & Ahmed, 2020). Anderson-Darling confirmed highly significant deviation across all features at a 1% significance level (Berenson et al., 2019; Bobbitt, 2019). Because outliers shift the mean, Table 3 shows mode and median are more reliable measures of central tendency (Holmes et al., 2022). Outliers were removed using Interquartile Range (IQR), reducing the dataset to 29,408 entries (Bruce et al., 2020).

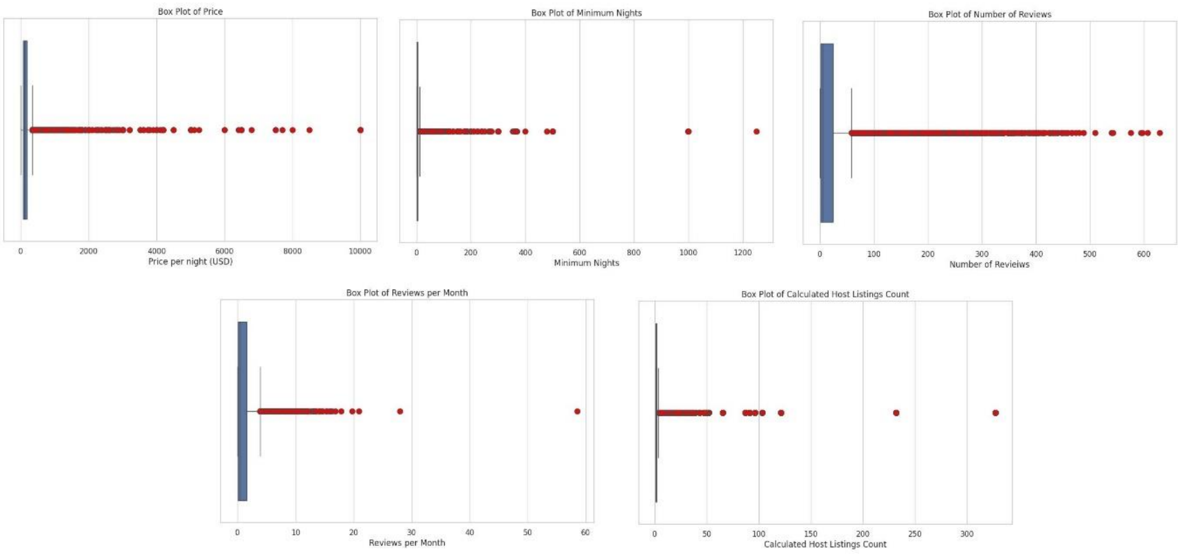


FIGURE 5 | Outliers (red) in box plots (`price`, `minimum_nights`, `number_of_reviews`, `reviews_per_month`, and `calculated_host_listings_count`)

TABLE 3 | Price

Mode	$100.00
Median	$106.00
Mean	$152.72
Min	$0
Max	$10,000
Upper band of price outliers	$334.00
Number of outliers	2,972

DATA LEARNINGS

Price

Figure 6 shows a modal peak at $100 (Oluleye, 2023). The long right tail starts at $334, with 2972 outliers reaching up to $10,000. This indicates price variability from luxury listings.

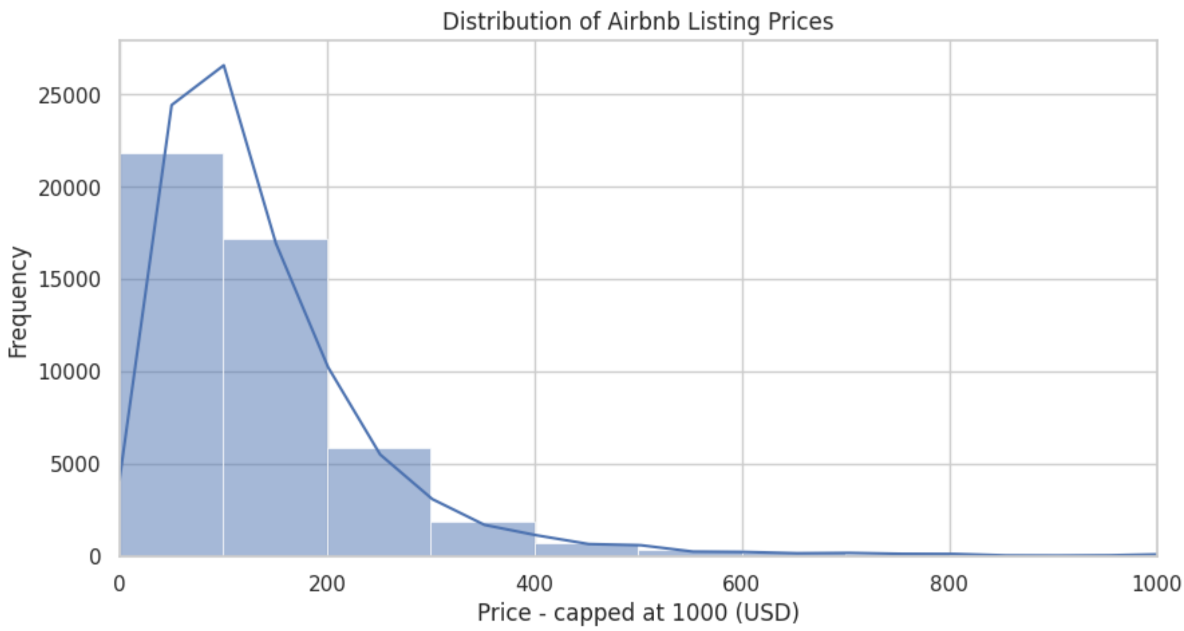


FIGURE 6 | Price

Location

Most listings are in Manhattan and Brooklyn (Figure 7), with high density and pricing indicating strong demand in these popular boroughs (Figure 3). Median neighbourhood prices vary (Figure 8), with Staten Island's Fort Wadsworth and Woodrow reflecting higher Zillow (N.D.) property values (Figure 9).

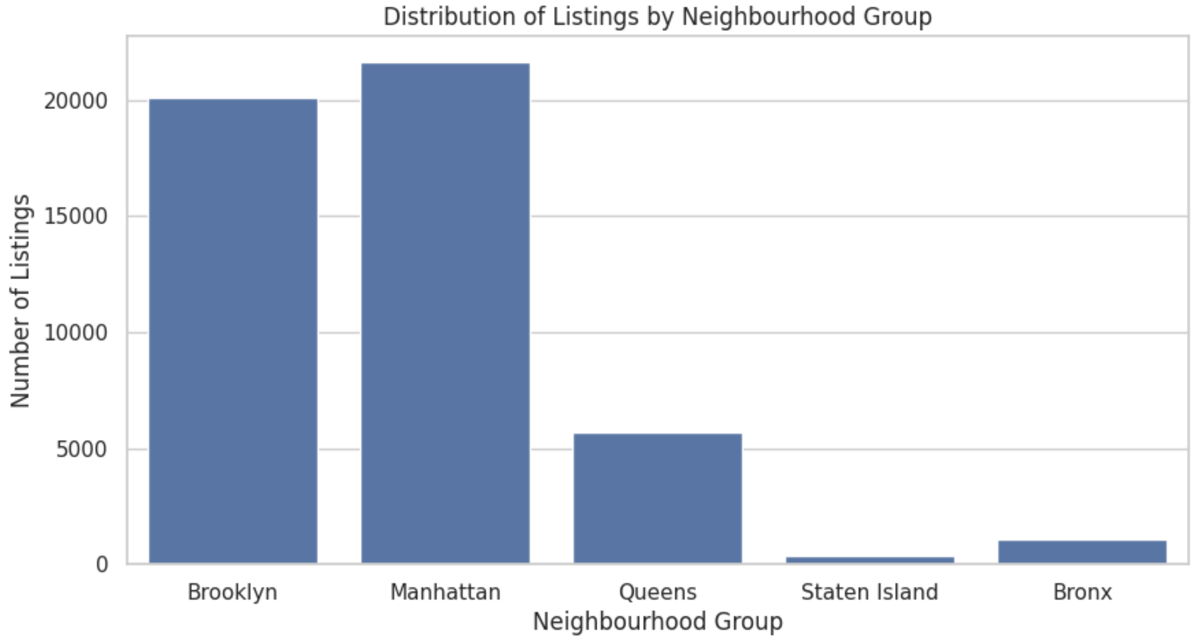


FIGURE 7 | Neighbourhood Group

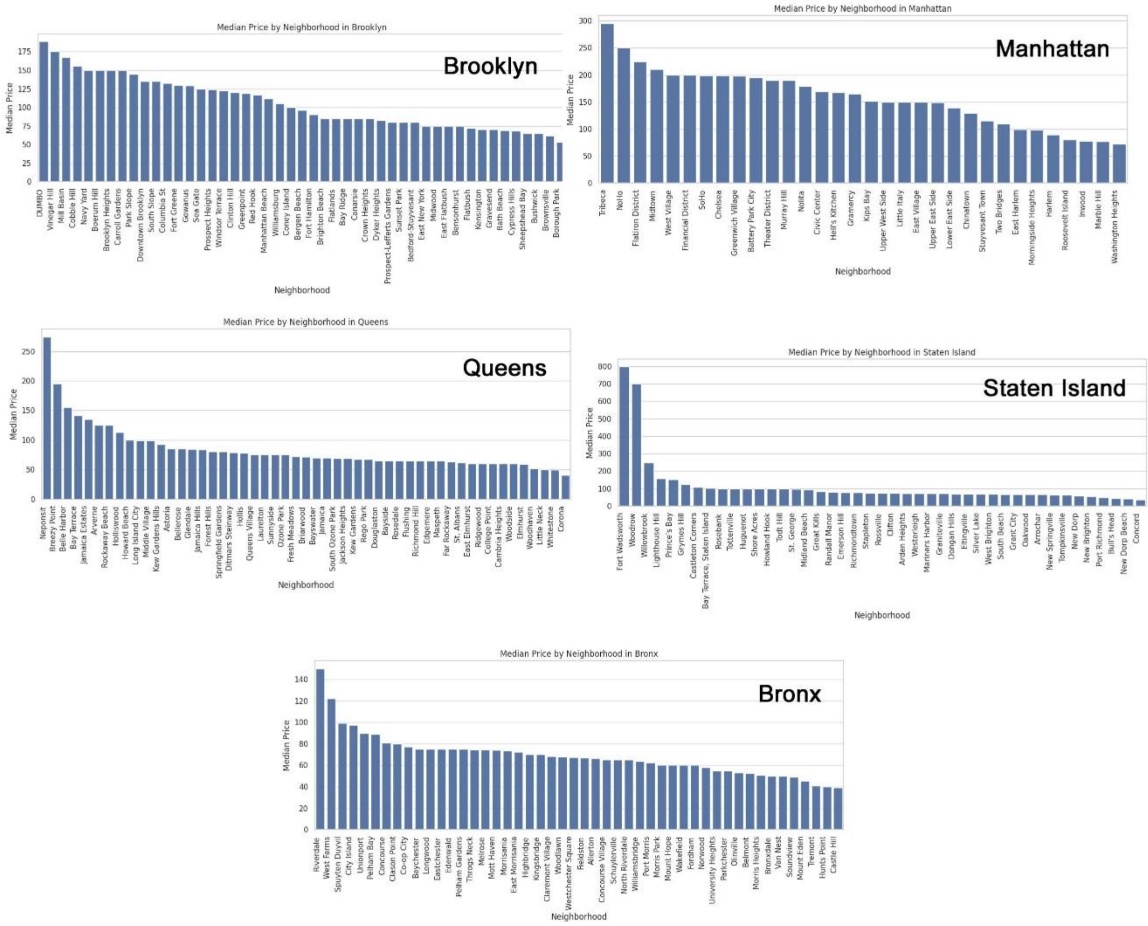


FIGURE 8 | Neighbourhood median prices

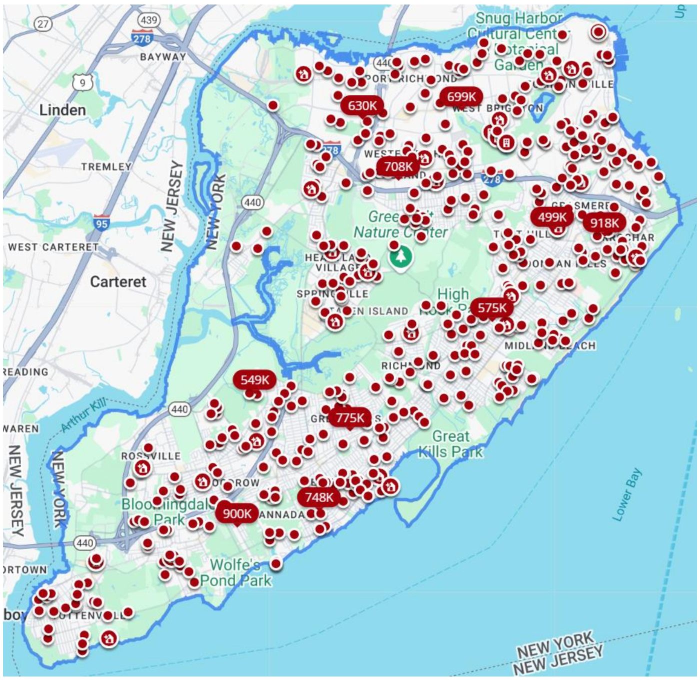


FIGURE 9 | Staten Island property prices

Room Type

Most listings are entire homes, followed by private rooms (Figure 10). Manhattan's entire homes fetch the highest prices (Figure 11), highlighting demand for private, exclusive, central accommodation.

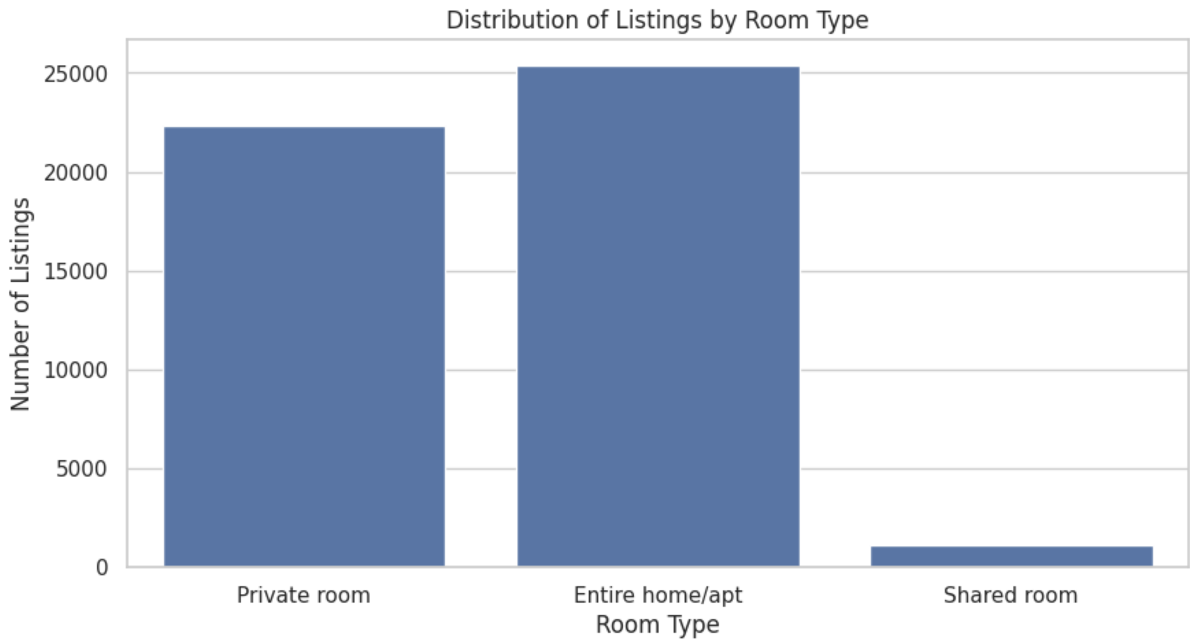


FIGURE 10 | Room type

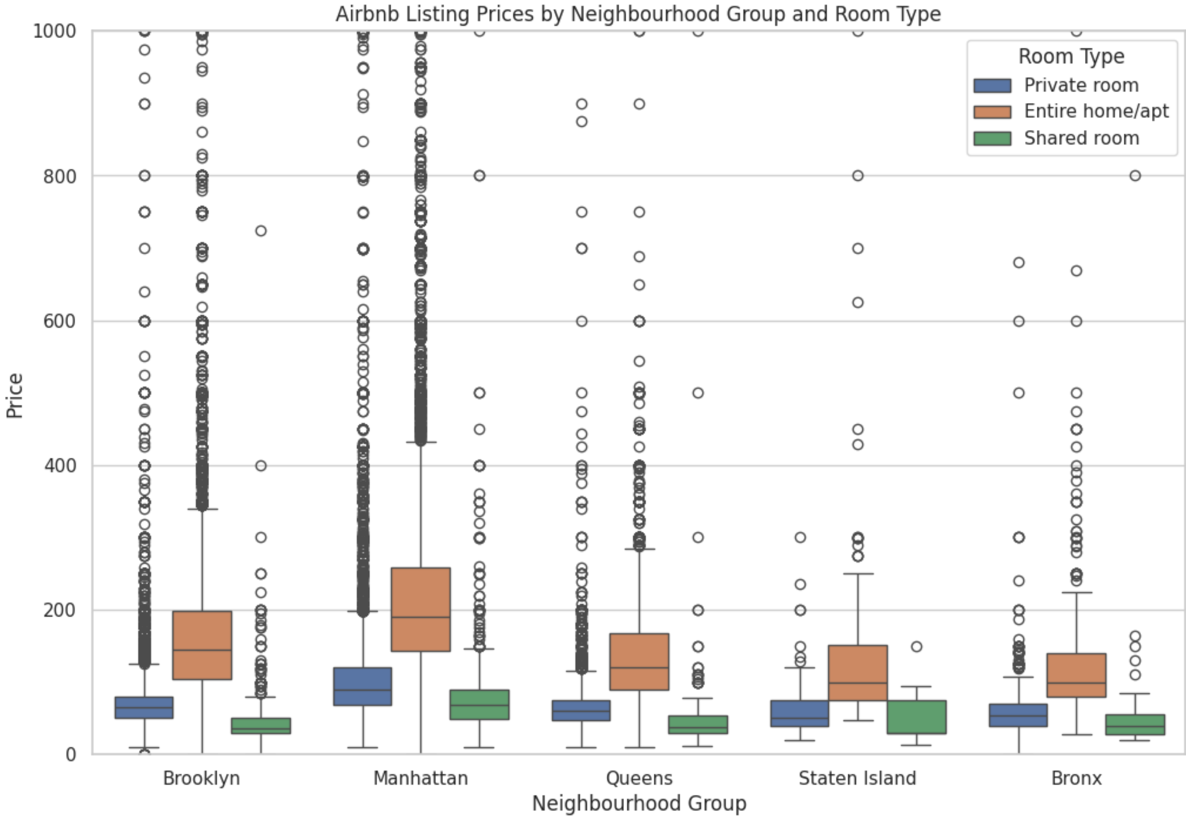


FIGURE 11 | Price per neighbourhood group by room type

Host Characteristics

Of 37,457 hosts, 5,154 professionals manage multiple listings (Figure 12) (Abrate et al., 2022). Removing outliers above 3.5 listings reduces hosts with full-year availability (Figure 13, Table 4). Regardless of outliers, correlation between price and number of listings per host is weak.

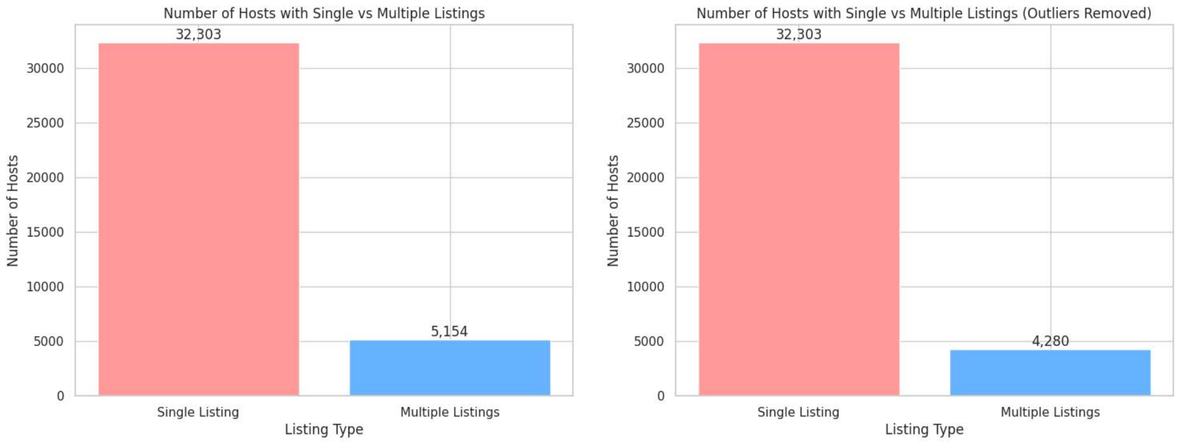


FIGURE 12 | Single versus multiple listings (left), outliers removed (right)

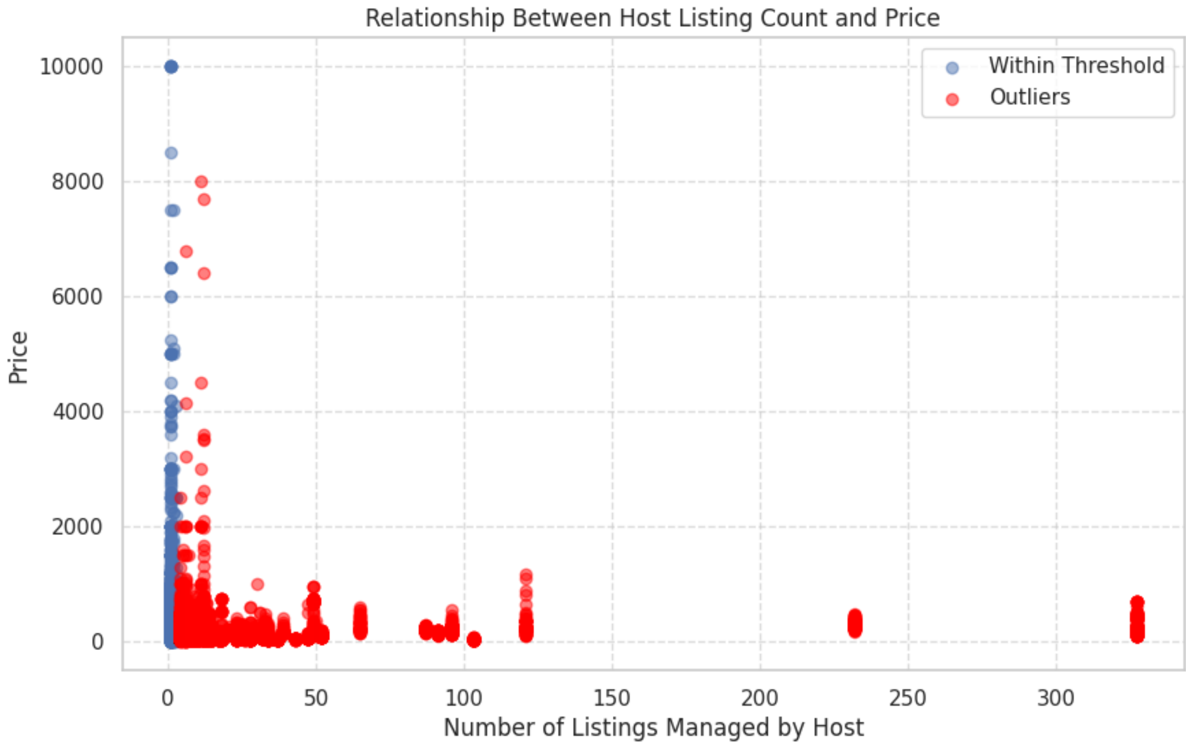


FIGURE 13 | Price per host listing count

TABLE 4 | Host characteristics

Metric	Before Outlier Removal	After Outlier Removal
Total Listings	48895.00	41814.00
Unique Hosts	37457.00	36583.00
Hosts with 1 Listing	32303.00	32303.00
Hosts with More Than 1 Listing	5154.00	4280.00
Average Reviews Per Host	1.03	1.02
Hosts with Full Year Availability	894.00	741.00

Correlation

Figure 14 indicates weak correlations, with a notable negative correlation (-0.60), between `room_type_encoded` and `price`. Log transformation (Figure 13) had minimal impact on correlation strength.

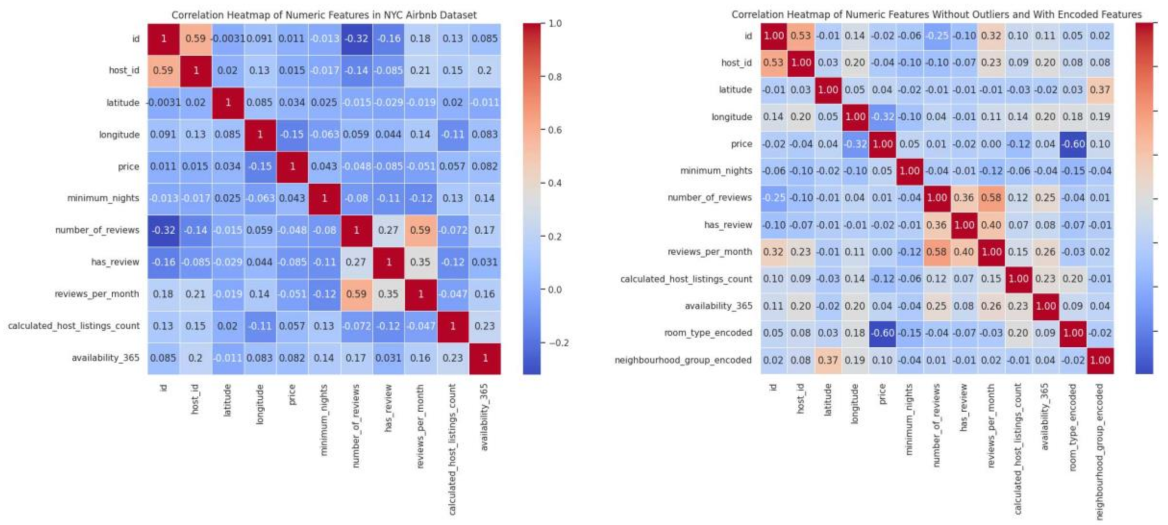


FIGURE 14 | Linear correlation (left); without outliers and with encoded categoricals (right)

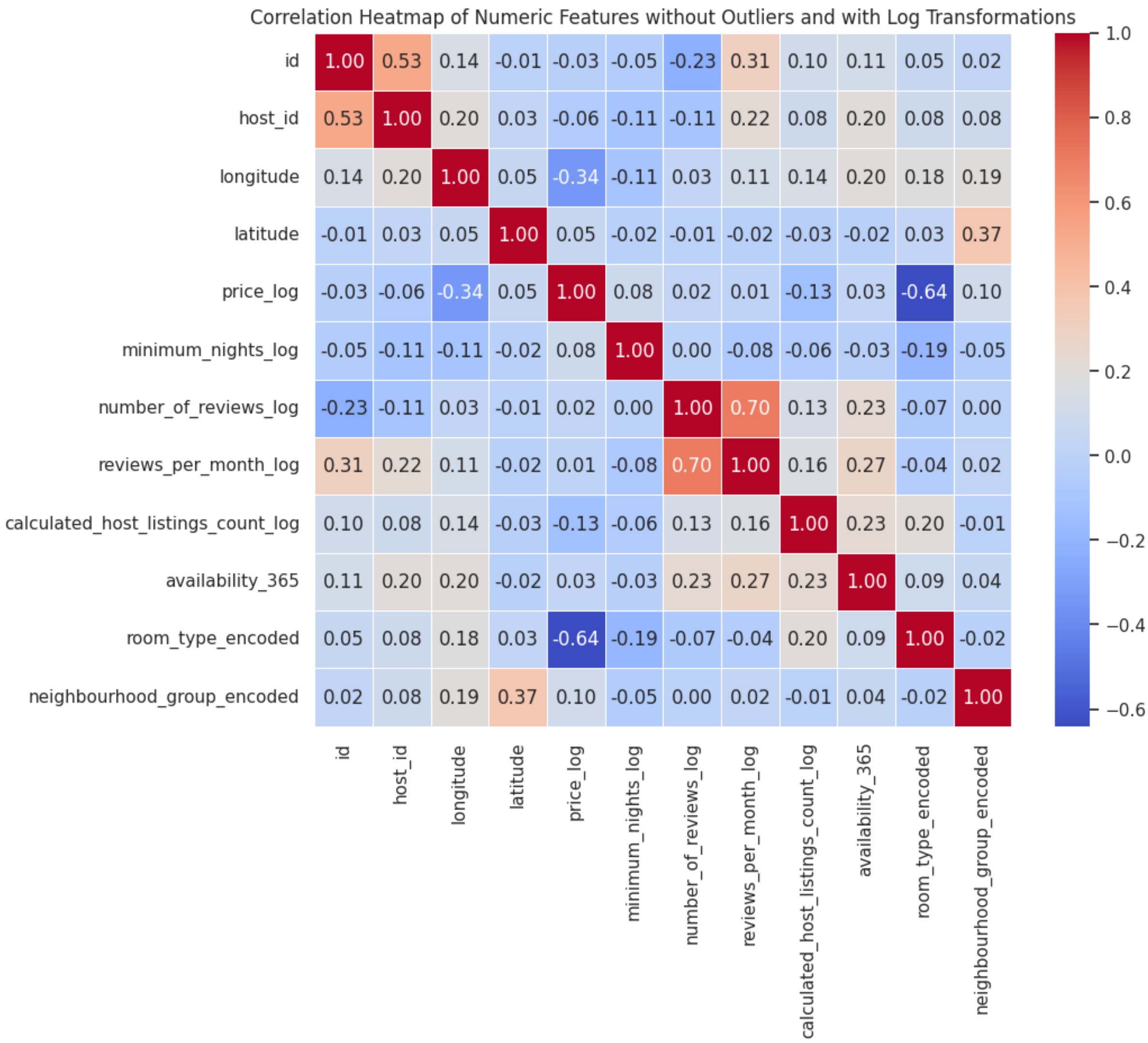


FIGURE 15 | Log-linear correlation

Clustering

Three optimal *k*-means clusters were identified (Figure 16), with numerical encoding enhancing analysis (Table 5). Cluster 0, half the size, shows higher activity with the most reviews and availability (Figures 17, 18). Cluster 1 features higher pricing concentrated in Manhattan and Brooklyn, while Cluster 2 represents budget-friendly options spread throughout NYC (Figures 18, 19).

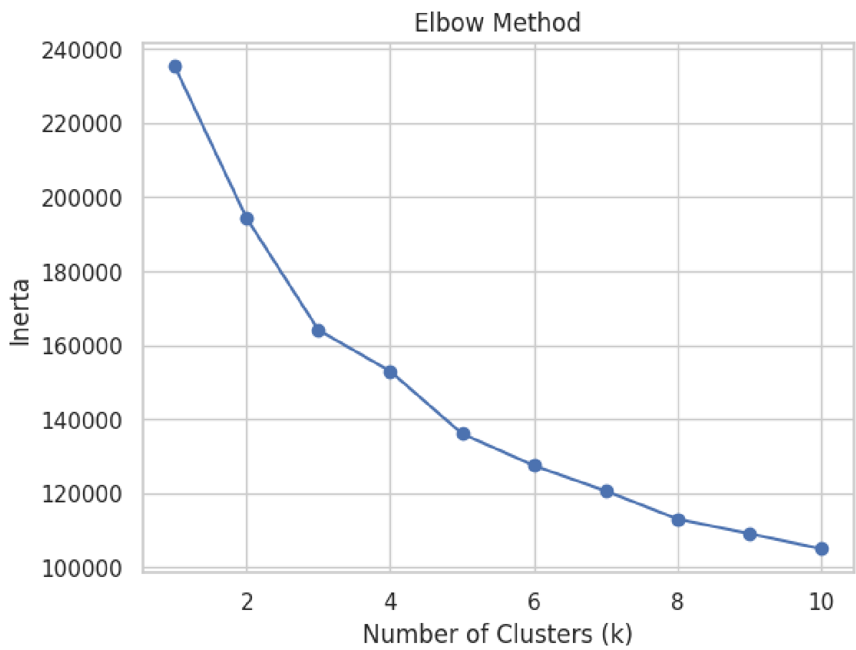


FIGURE 16 | Elbow method

TABLE 5 | Encoding

`room_type`	Entire: 0, Private: 1, Shared: 2
`neighbourhood_group`	Bronx: 0, Brooklyn: 1, Manhattan: 2, Queens: 3, Staten Island: 4

FIGURE 17 | Cluster sizes

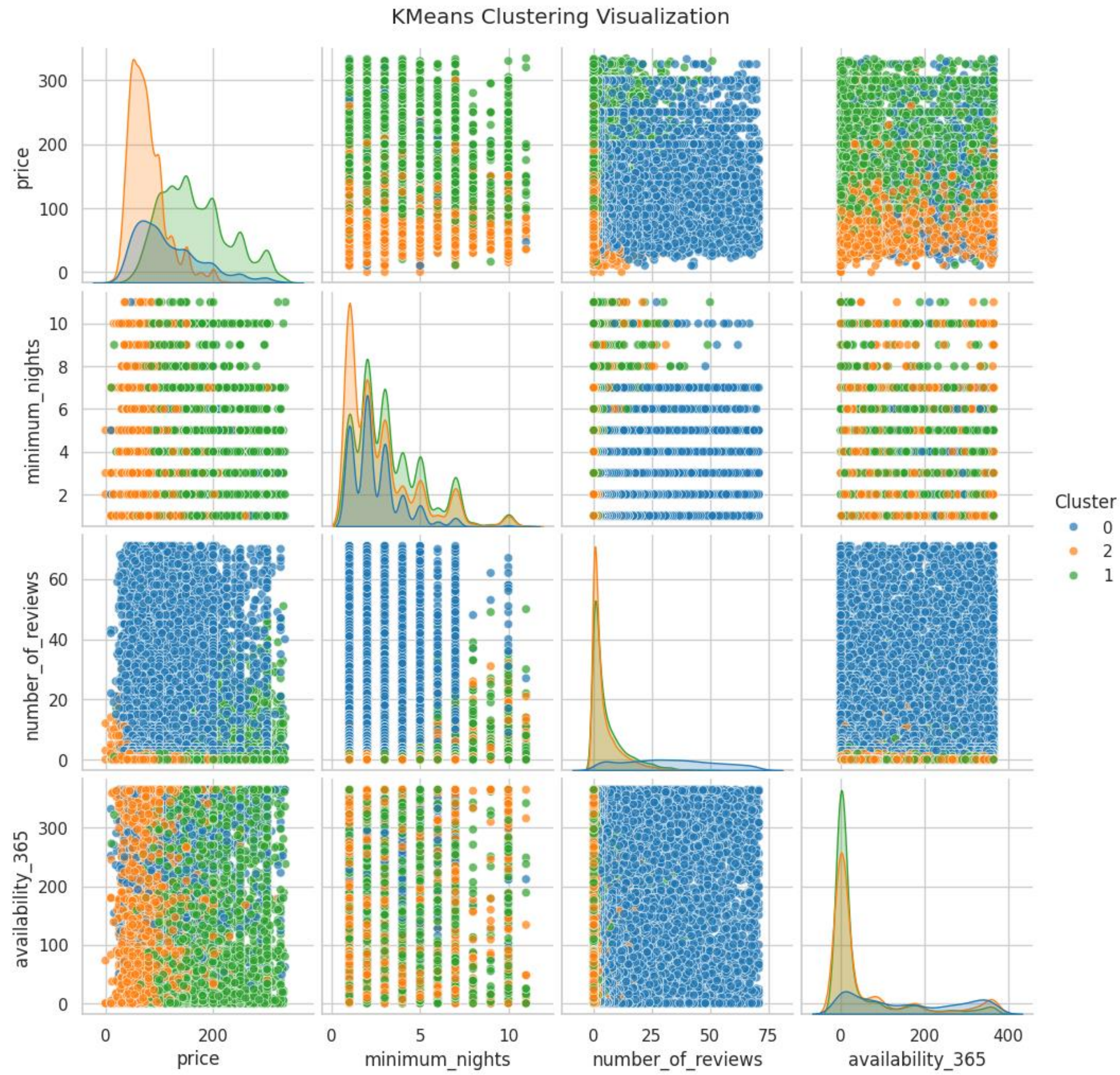


FIGURE 18 | Clustering

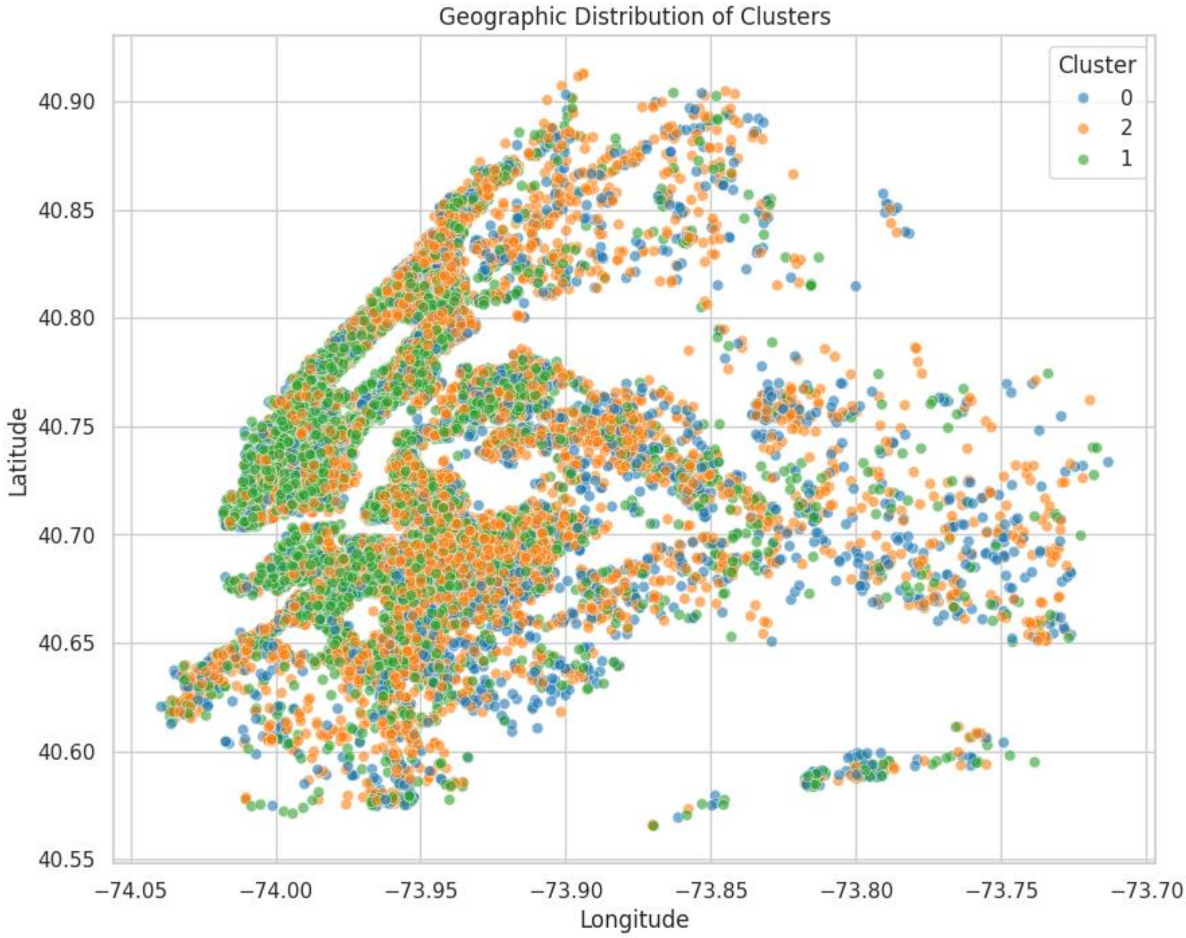


FIGURE 19 | Geographic clustering

INSIGHTS

Revenue optimisation requires cluster-specific strategies (Tables 6, 7). Cluster 1's exclusivity commands highest prices, but Cluster 0's moderate pricing and higher availability outpaces revenue at the same booking rate. Half of Cluster 0 and 1 bookings have reviews, while Cluster 2 has one-fifth, suggesting potential to improve relationships. Dynamic pricing benefits all hosts, particularly Cluster 1 during peaks, with Clusters 0 and 2 benefiting seasonally (Abrate et al., 2022). NYC's Local Law 18 and market saturation pose challenges as expansion markets outpace core markets (Airbnb, N.D.b; Voltes-Dorta, 2024). Opportunities exist through data integration, like census, and advanced ML for improved targeting and revenue strategies (Zhu et al., 2020; Akalın & Alptekin, 2024).

TABLE 6 | Cluster mean analysis

Cluster (Listings)	Price	Minimum Nights	Number of Reviews	Has Review	Reviews per Month	Calculated Host Listings Count	Availability 365	Room Type Encoded	Neighbourhood Group Encoded
0 Moderately priced, available (5807)	£117.37	2.5	33	1	1.73	1.5	156.2	0.53 Split entire home and private room	1.66 Majority Brooklyn, some Manhattan
1 High-priced homes, limited availability (11,934)	£166.34	3.3	6	0.77	0.33	1.11	40.5	0.02 Entire home	1.65 Majority Brooklyn, some Manhattan
2 Budget-friendly rooms, moderate availability (11,658)	£77.08	2.7	4	0.7	0.28	1.3	59.6	1.04 Mostly private room / few shared	1.57 Less Manhattan, Brooklyn, Bronx

TABLE 7 | Example revenue

	Price x Listings x Availability	Booking Rate (%)	ANNUAL REVENUE (USD millions)
Cluster 0	$106.46m	50%	$53.23m
Cluster 1	$80.40m	50%	$40.20m
Cluster 2	$53.56m	50%	$26.78m

CONCLUSION

This analysis aimed to optimise Airbnb's NYC revenue through EDA and cluster-specific insights. Tailored pricing strategies—exclusivity for luxury, enhanced booking rates for moderate, improved engagement for budget, and dynamic pricing for all—can boost profitability. Challenges like Local Law 18 and market saturation underscore the need for continuous data-driven adaptation to continue revenue growth in the home-sharing market (Voltes-Dorta, 2024).

REFERENCES

Abrate, G., Sainaghi, R. & Mauri, A.G. (2022) Dynamic pricing in Airbnb: Individual versus professional hosts, *Journal of Business Research* 141: 191–199. DOI: https://doi.org/10.1016/J.JBUSRES.2021.12.012.

Airbnb (N.D.a) *About Us*. Available from: https://news.airbnb.com/about-us/ [Accessed 25 November 2024].

Airbnb (N.D.b) *Airbnb Investor: Quarterly Results*. Available from: https://investors.airbnb.com/financials/default.aspx#quarterly [Accessed 25 November 2024].

Akalın, Ö. & Alptekin, G.I. (2024) Enhancing Airbnb Price Predictions with Location-Based Data: A Case Study of Istanbul 207–212. DOI: https://doi.org/10.15439/2024F7603.

Berenson, M., Levine, D., Szabat, K. & Stephan, D. (2019) *Basic Business Statistics, Global Edition*. 14th ed. Harlow: Pearson International Content.

Bobbitt, Z. (2019) *How to Conduct an Anderson-Darling Test in R*. Available from: https://www.statology.org/anderson-darling-test-r/ [Accessed 27 November 2024].

Bruce, P., Bruce, A. & Gedeck, P. (2020) *Practical Statistics for Data Scientists*. 2nd ed. O’Reilly Media. Available from: vbk://9781492072898 [Accessed 24 May 2024].

Ding, K., Niu, Y. & Choo, W.C. (2023) The evolution of Airbnb research: A systematic literature review using structural topic modeling, *Heliyon* 9(6). DOI: https://doi.org/10.1016/j.heliyon.2023.e17090.

Gunter, U., Önder, I. & Zekan, B. (2020) Modeling Airbnb demand to New York City while employing spatial panel data at the listing level, *Tourism Management* 77: 104000. DOI: https://doi.org/10.1016/J.TOURMAN.2019.104000.

Holmes, A., Illowsky, B. & Dean, S. (2022) *Introductory Business Statistics*. OpenStax. Available from: https://openstax.org/details/books/introductory-business-statistics [Accessed 25 April 2024].

IBM (N.D.) *What is Personally Identifiable Information (PII)?* Available from: https://www.ibm.com/topics/pii [Accessed 27 November 2024].

Kaggle (2021) *New York City Airbnb Open Data*, *Kaggle*. Available from: https://www.kaggle.com/code/whyalwaysme/ab-nyc-2019 [Accessed 26 November 2024].

Mukhiya, S.K. & Ahmed, U. (2020) *Hands-On Exploratory Data Analysis with Python*. Packt Publishing.

Niakšu, O. (2015) CRISP Data Mining Methodology Extension for Medical Domain, *Baltic J Modern Computing* 3(2): 92–109. Available from: https://www.researchgate.net/publication/277775478_CRISP_Data_Mining_Methodology_Extension_for_Medical_Domain [Accessed 30 October 2023].

Oluleye, A. (2023) *Exploratory Data Analysis with Python Cookbook*. Pockt Publishing. Available from: https://learning.oreilly.com/library/view/exploratory-data-analysis/9781803231105/ [Accessed 12 November 2024].

Taylor, B. & Almeida, A. (2022) *Revenue Growth*, *Stock Analysis*. Available from: https://stockanalysis.com/term/revenue-growth/ [Accessed 26 November 2024].

Voltes-Dorta, A. (2024) Price competition among short-term Airbnb listings in New York City following Local Law 18, *Current Issues in Tourism* [Preprint]. DOI: https://doi.org/10.1080/13683500.2024.2354521.

Zhu, A., Li, R. & Xie, Z. (2020) Machine Learning Prediction of New York Airbnb Prices, *Proceedings - 2020 3rd International Conference on Artificial Intelligence for Industries, AI4I 2020* 1–5. DOI: https://doi.org/10.1109/AI4I49448.2020.00007.

Zillow (N.D.) *Staten Island Property Prices*. Available from: https://www.zillow.com/staten-island-new-york-ny/ [Accessed 28 November 2024].

APPENDIX

Code: https://github.com/mariaingold/AirbnbNYC

Dataset: AB_NYC_2019.csv

Colab:

https://colab.research.google.com/github/mariaingold/AirbnbNYC/blob/main/AirbnbNYC.ipynb

MACHINE LEARNING: AIRBNB ANALYSIS PROJECT
UoEO Machine Learning
Assignment Due Date: 2 December, 2024
TEAM 2 AUTHORS

- Ahmed Husain
- Dinh (Danty) Khoi
- Maria Ingold
- Murthy Kanuri

PROJECT DESCRIPTION
Development Team Project
Determine interesting business analytic substantive question answered by dataset that is useful for the Airbnb executive board.
Business Analytic Question
Agreed by: Ahmed, Danty, Maria, Murthy
How can Airbnb NYC optimise revenue growth through tailored pricing recommendations based on location, room type, and host characteristics?
Source Dataset: Airbnb NYC 2019

Code

CONTRIBUTIONS

ACTION	**WHO**	**DESCRIPTION**
Setup	Maria	Google Colab, GitHub (connection, collaboration, Readme), structure, initial setup, basic data exploration.
EDA	Danty	Exploratory Data Analysis (EDA)
	Ahmed	Exploratory Data Analysis
	Maria	Basic EDA as part of setup, comment detail, outlier boxplots, added neighbourhood, added elbow method
	Murthy	Mode and Range
Machine Learning	Danty	Clustering

ACTION	WHO	DESCRIPTION
	Ahmed	Clustering
Review	Ahmed	EDA and clustering
	Danty	EDA and clustering
	Maria	Merged initial notebook with Danty's and Ahmed's EDA and clustering, Murthy's mode and range analysis, removed duplications, consolidated variable names, fixed warnings, added structure and additional comments.
	Murthy	EDA and clustering
Analysis	Ahmed	EDA and clustering
	Danty	EDA and clustering. Comment detail. Input into report draft.
	Maria	EDA and clustering. Detailed comments to provide analysis of results. Further research into outliers especially for pricing. Refined missing data handling. Added more on cluster mean analysis.
	Murthy	EDA and clustering. Input into report draft.

SETUP

Import Libraries

In [1]:

```
import pandas as pd                  # data manipulation
import numpy as np                   # numerical computation / inear algebra
import seaborn as sns                # enhanced visuals and statistics
import geopandas as gpd              # geospatial data
import matplotlib.lines as mlines    # customised legend lines / markers
import folium                        # interactive maps
from matplotlib import pyplot as plt # plots, charts and figures
from scipy import stats              # Anderson-Darling test for normality
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, LabelEncoder, MinMaxScaler
from sklearn.metrics import silhouette_score
from shapely.geometry import Point
```

Import Data

Manually upload AB_NYC_2019.csv into Session Storage in Google Colab

Read Data

In [2]:

```
nyc_airbnb_df = pd.read_csv("AB_NYC_2019.csv")
```

EXPLORATORY DATA ANALYSIS

Overview

Column overview

The NYC Airbnb dataset has 48,895 entries and 16 columns. Key columns include:

- Host Information: host_id and host_name.
- Location Information: neighbourhood_group, neighbourhood, latitude, longitude.
- Listing Characteristics: room_type, price, minimum_nights, number_of_reviews, last_review, and reviews_per_month.
- Listing Metrics: calculated_host_listings_count (number of listings by the host) and availability_365 (days available per year).

Some columns, such as name, host_name, and last_review, contain missing values. Specifically:

- name and host_name have a few missing entries.
- last_review and reviews_per_month have more substantial missing values, likely due to properties with no reviews.

EDA Plan for This Dataset

1. Data Understanding: Structure and contents: columns, rows, column names and types
2. Data Cleaning: Identify and handle missing values.
3. Descriptive Anaysis: Descriptive statistical analysis like skew and kurtosis for numeric features.
4. Univariate Analysis: Examine distributions for numerical columns (e.g., price, minimum_nights).
5. Bivariate Analysis: Examine two variables.

6. Multivariate Analysis: Clustering and more.

Steps to Address Question

1. Analyze Price Distribution: Examine price to identify outliers and distribution shape.
2. Price by Neighbourhood Group: Analyze how prices vary by neighbourhood_group.
3. Price by Room Type: Examine price trends across different room types.
4. Price vs Host Characteristics. Calculated Host Listings Count: Investigate if hosts with multiple listings set different price levels.
5. Interactions: Neighbourhood Group & Room Type: Analyze combined influence on prices.
6. Cluster analysis.

DATA UNDERSTANDING

Get a sense of the data structure and contents by summarising the data.

Number of columns and rows

In [3]:

```
print("Number of columns: ", len(nyc_airbnb_df.columns))
print("Number of rows: ", len(nyc_airbnb_df))
```

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

Column names, non-nulls and types

In [4]:

```
nyc_airbnb_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   name                            48879 non-null  object
 2   host_id                         48895 non-null  int64
 3   host_name                       48874 non-null  object
 4   neighbourhood_group             48895 non-null  object
 5   neighbourhood                   48895 non-null  object
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object
 9   price                           48895 non-null  int64
 10  minimum_nights                  48895 non-null  int64
 11  number_of_reviews               48895 non-null  int64
 12  last_review                     38843 non-null  object
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365                48895 non-null  int64
dtypes: float64(3), int64(7), object(6)
memory usage: 6.0+ MB
```

5-Figure Summary per Column

- Mean (mean)
- Standard Deviation (std)
- Minimum and Maximum (min, max)
- Median (50%)
- Quartile 1 (25%)
- Quartile 3 (75%)

In [5]:

```
nyc_airbnb_df.describe()
```

Out[5]:

	id	**host_id**	**latitude**	**longitude**	**price**	**minimum_nights**	**number_of_reviews**	**reviews_per_month**	**calculated_host_listings_count**	**availability_365**
count	4.889500e+04	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000	48895.000000
mean	1.901714e+07	6.762001e+07	40.728949	-73.952170	152.720687	7.029962	23.274466	1.373221	7.143982	112.781327

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365
std	1.098311e+07	7.861097e+07	0.054530	0.046157	240.154170	20.510550	44.550582	1.680442	32.952519	131.622289
min	2.539000e+03	2.438000e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.010000	1.000000	0.000000
25%	9.471945e+06	7.822033e+06	40.690100	-73.983070	69.000000	1.000000	1.000000	0.190000	1.000000	0.000000
50%	1.967728e+07	3.079382e+07	40.723070	-73.955680	106.000000	3.000000	5.000000	0.720000	1.000000	45.000000
75%	2.915218e+07	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	2.020000	2.000000	227.000000
max	3.648724e+07	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.500000	327.000000	365.000000

Mode

This is the first mode for each column.

In [6]:

```
# Calculate the mode for each column
mode_values = nyc_airbnb_df.mode().iloc[0]

# Display the mode values
print(mode_values)
```

```
id                                              2539
name                                  Hillside Hotel
host_id                                  219517861.0
host_name                                    Michael
neighbourhood_group                        Manhattan
neighbourhood                           Williamsburg
latitude                                    40.71813
longitude                                  -73.95677
room_type                            Entire home/apt
price                                          100.0
minimum_nights                                   1.0
number_of_reviews                                0.0
last_review                               2019-06-23
reviews_per_month                               0.02
calculated_host_listings_count                   1.0
availability_365                                 0.0
Name: 0, dtype: object
```

Range

In [7]:

```
# Calculate the range for each numerical column
range_values = nyc_airbnb_df.select_dtypes(include='number').apply(lambda x: x.max() - x.min())

# Convert the range values to avoid exponential notation
range_values_formatted = range_values.apply(lambda x: f"{x:.0f}" if x >= 1 else f"{x:.6f}")

# Display the range values
print("Range for each numerical column:")
print(range_values_formatted)
```

```
Range for each numerical column:
id                                  36484706
host_id                            274318875
latitude                            0.413270
```

```
longitude                           0.531430
price                                  10000
minimum_nights                          1249
number_of_reviews                        629
reviews_per_month                         58
calculated_host_listings_count           326
availability_365                         365
dtype: object
```

Review Data (head and tail)

Reviewing both head and tail shows a more complete picture.

Head

Head shows mostly complete data.

In [8]:

```
nyc_airbnb_df.head()
```

Out[8]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
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	6	365
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	355
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	NaN	NaN	1	365
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	1	194
4	50	Entire Apt	71	Laura	Manhattan	East	40.7	-73.9	Entire	80	10	9	2018-	0.10	1	0

id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
22	: Spacious Studio/Loft by central park	92			Harlem	9851	4399	home/apt				11-19			

Tail

Tail shows more NaN -- more missing data.

In [9]:

```
nyc_airbnb_df.tail()
```

Out[9]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	Bedford-Stuyvesant	40.67853	-73.94995	Private room	70	2	0	NaN	NaN	2	9
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	Bushwick	40.70184	-73.93317	Private room	40	4	0	NaN	NaN	2	36

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	Harlem	40.81475	-73.94867	Entire home/apt	115	10	0	NaN	NaN	1	27
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	Hell's Kitchen	40.75751	-73.99112	Shared room	55	1	0	NaN	NaN	6	2
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	Hell's Kitchen	40.76404	-73.98933	Private room	90	7	0	NaN	NaN	1	23

DATA CLEANING

Identify missing values

In [10]:

```
# Identify missing values in the dataset
missing_values = nyc_airbnb_df.isnull().sum()
missing_values = missing_values[missing_values > 0].sort_values(ascending=False)
missing_values
```

Out[10]:

	0
last_review	10052
reviews_per_month	10052
host_name	21
name	16

dtype: int64

reviews_per_month

Missing reviews likely means none have been made, so set to 0.

In [11]:

```
# Set missing reviews per month to 0
nyc_airbnb_df.loc[nyc_airbnb_df['reviews_per_month'].isnull(), 'reviews_per_month'] = 0
```

last_review

- Missing last review likely means there is none. Set date to be before 2008 where it can't exist, but keeps datetime type.
- Don't want to lose that it has no review so add has_review column.

In [12]:

```
# Fill NaNs with a dummy date outside the range to keep datetime dtype
nyc_airbnb_df['last_review'] = nyc_airbnb_df['last_review'].fillna(pd.Timestamp.min)

# Add column for feature engineering with 0 for no review, or 1 for has review.
nyc_airbnb_df['has_review'] = nyc_airbnb_df['last_review'].apply(lambda x: 0 if x ==
pd.Timestamp.min else 1)

# Reorder columns to place 'has_review' after 'last_review'
cols = list(nyc_airbnb_df.columns)
last_review_index = cols.index('last_review')
cols.insert(last_review_index + 1, cols.pop(cols.index('has_review')))
nyc_airbnb_df = nyc_airbnb_df.reindex(columns=cols)
```

host_name and name

- host_name is personally identifiable information (PII) and should be removed.
- name (of listing) is not being used for this analysis, so has been removed.

In [13]:

```
# drop host_name and name
nyc_airbnb_df = nyc_airbnb_df.drop(['host_name', 'name'], axis=1, errors='ignore')
```

Verify missing values after handling

In [14]:

```
missing_values_after_cleaning = nyc_airbnb_df.isnull().sum()
missing_values_after_cleaning
```

Out[14]:

	0
id	0
host_id	0
neighbourhood_group	0
neighbourhood	0
latitude	0
longitude	0
room_type	0
price	0
minimum_nights	0
number_of_reviews	0
last_review	0
has_review	0

	0
reviews_per_month	0
calculated_host_listings_count	0
availability_365	0

dtype: int64

In [15]:

```
nyc_airbnb_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 15 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              48895 non-null  int64  
 1   host_id                         48895 non-null  int64  
 2   neighbourhood_group             48895 non-null  object 
 3   neighbourhood                   48895 non-null  object 
 4   latitude                        48895 non-null  float64
 5   longitude                       48895 non-null  float64
 6   room_type                       48895 non-null  object 
 7   price                           48895 non-null  int64  
 8   minimum_nights                  48895 non-null  int64  
 9   number_of_reviews               48895 non-null  int64  
 10  last_review                     48895 non-null  object 
 11  has_review                      48895 non-null  int64  
 12  reviews_per_month               48895 non-null  float64
 13  calculated_host_listings_count  48895 non-null  int64  
 14  availability_365                48895 non-null  int64  
dtypes: float64(3), int64(8), object(4)
memory usage: 5.6+ MB
```

In [16]:

```
nyc_airbnb_df.head()
```

Out[16]:

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	has_review	reviews_per_month	calculated_host_listings_count	availability_365
0	2539	2787	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	2018-10-19	1	0.21	6	365
1	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	1	0.38	2	355
2	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	1677-09-21 00:12:43.145224193	0	0.00	1	365

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	has_review	reviews_per_month	calculated_host_listings_count	availability_365
3	3831	4869	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	2019-07-05	1	4.64	1	194
4	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	1	0.10	1	0

Numeric Variables

In [17]:

```
# Selecting only numeric columns for analysis
numeric_columns_nyc = nyc_airbnb_df.select_dtypes(include=['float64', 'int64']).columns
```

DESCRIPTIVE ANALYTICS

Skew and Kurtosis

Descriptive analysis on numerical columns to examine distributions and calculate skewness and kurtosis for each.

SKEWNESS

- Asymmetry of distribution of dataset.
- Positive skew: right skewed (long right tail)
- Negative skew: left skewed (long left tail)
- Zero skew: normal distribution

High Positive Skew (large outliers towards right)

- price
- minimum_nights
- number_of_reviews
- reviews_per_month
- calculated_host_listings_count

Positive Skew (some outliers towards right)

- host_id
- longitude

Slight Postive Skew (closer to normal, some outliers to right)

- latitude
- availability_365

Very Slightly Negative (almost normal)

- id

KURTOSIS

- Peakedness and tailedness of dataset distribution
- Postive kurtosis: peaked with more tails (outliers)
- Negative kurtosis: flatter and lighter tails (concentrated at mean)
- Zero kurosis: normal peakedness in distribution

Very High Postive Kurtosis (sharp peak, heavy tails --> outliers)

- price
- minimum_nights
- reviews_per_month
- calculated_host_listings_count

High Positive Kurtosis (sharp peak, heavey tails --> outliers)

- longitude

Slight Positive Kurtosis (closer to normal, but outliers)

- host_id

- latitude

Sighlty Negative Kurtosis

- id
- has_review
- availability_365

These findings suggest the need for potential transformations or outlier handling in these variables.

In [18]:

```
# Calculation of skewness and kurtosis for numerical columns

# Calculate skewness and kurtosis
skew_kurtosis_nyc = nyc_airbnb_df[numeric_columns_nyc].agg(['skew', 'kurtosis']).transpose()

# Display skewness and kurtosis results
skew_kurtosis_nyc
```

Out[18]:

	skew	kurtosis
id	-0.090257	-1.227748
host_id	1.206214	0.169106
latitude	0.237167	0.148845
longitude	1.284210	5.021646
price	19.118939	585.672879
minimum_nights	21.827275	854.071662
number_of_reviews	3.690635	19.529788
has_review	-1.457094	0.123127
reviews_per_month	3.300723	43.531611
calculated_host_listings_count	7.933174	67.550888
availability_365	0.763408	-0.997534

UNIVARIATE ANALYSIS

Oluleye (2023) for visualising one variable:

- summary table
- bar chart
- boxplot (good for finding outliers)
- histogram (good for finding outliers)
- pie chart
- violin plot

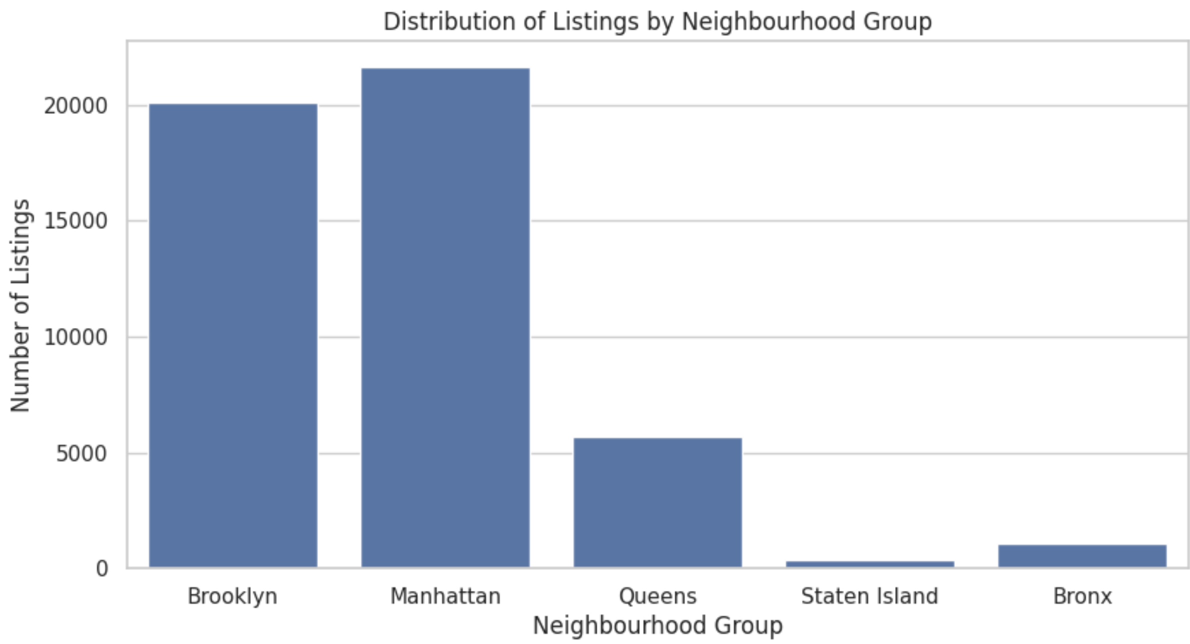
Categorical Analysis

Neighbourhood Group

The majority of listings are concentrated in Manhattan and Brooklyn, with fewer in Queens, the Bronx, and Staten Island.

In [19]:

```
# Set a general style for the plots
sns.set(style="whitegrid")

# Plotting the distribution of listings by neighbourhood group
plt.figure(figsize=(10, 5))
sns.countplot(x='neighbourhood_group', data=nyc_airbnb_df)
plt.title("Distribution of Listings by Neighbourhood Group")
plt.xlabel("Neighbourhood Group")
plt.ylabel("Number of Listings")
plt.show()
```

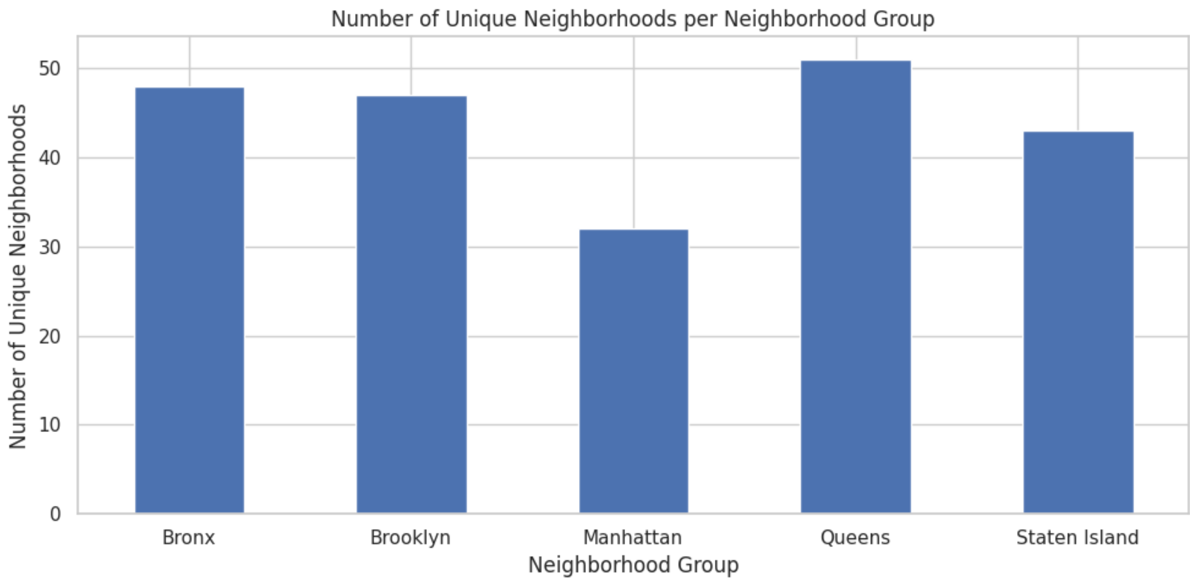


Unique Neighbourhoods in Neighbourhood Groups

In [20]:

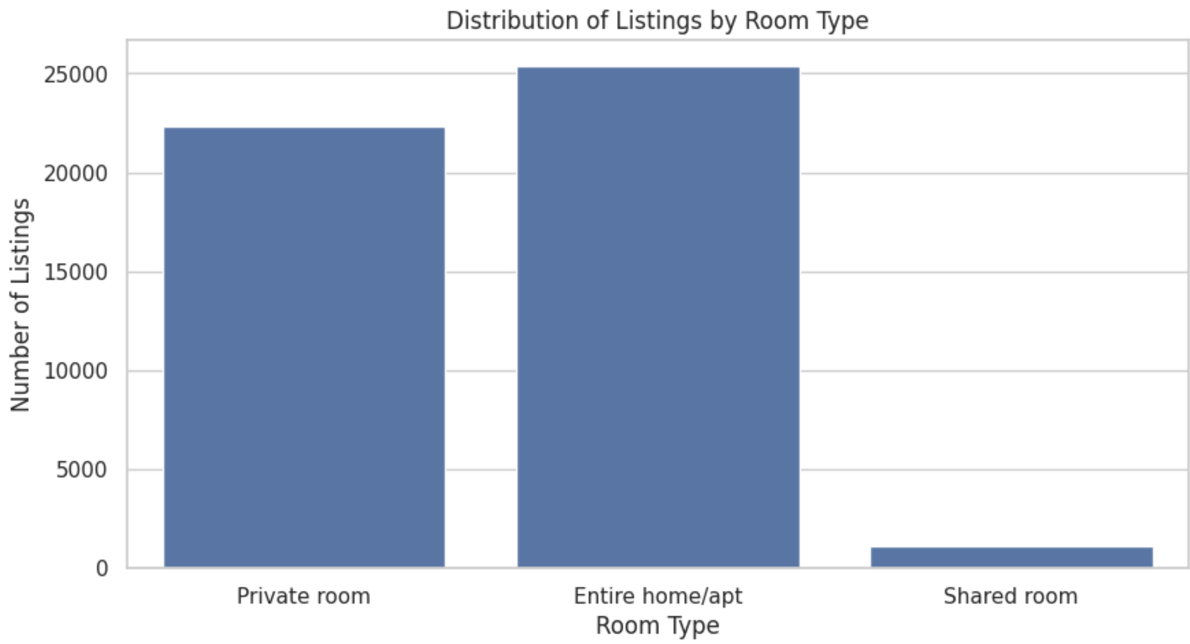
```
# Group by neighborhood_group and get unique neighborhoods
neighborhood_counts = nyc_airbnb_df.groupby('neighbourhood_group')['neighbourhood'].nunique()

# Print the results
print("Unique Neighborhoods per Neighborhood Group:\n", neighborhood_counts)

# Visualization using a bar chart
plt.figure(figsize=(10, 5))
neighborhood_counts.plot(kind='bar')
plt.title('Number of Unique Neighborhoods per Neighborhood Group')
plt.xlabel('Neighborhood Group')
plt.ylabel('Number of Unique Neighborhoods')
plt.xticks(rotation=0)  # Keep x-axis labels horizontal
plt.tight_layout()
plt.show()
```

```
Unique Neighborhoods per Neighborhood Group:
 neighbourhood_group
Bronx            48
Brooklyn         47
Manhattan        32
Queens           51
Staten Island    43
Name: neighbourhood, dtype: int64
```



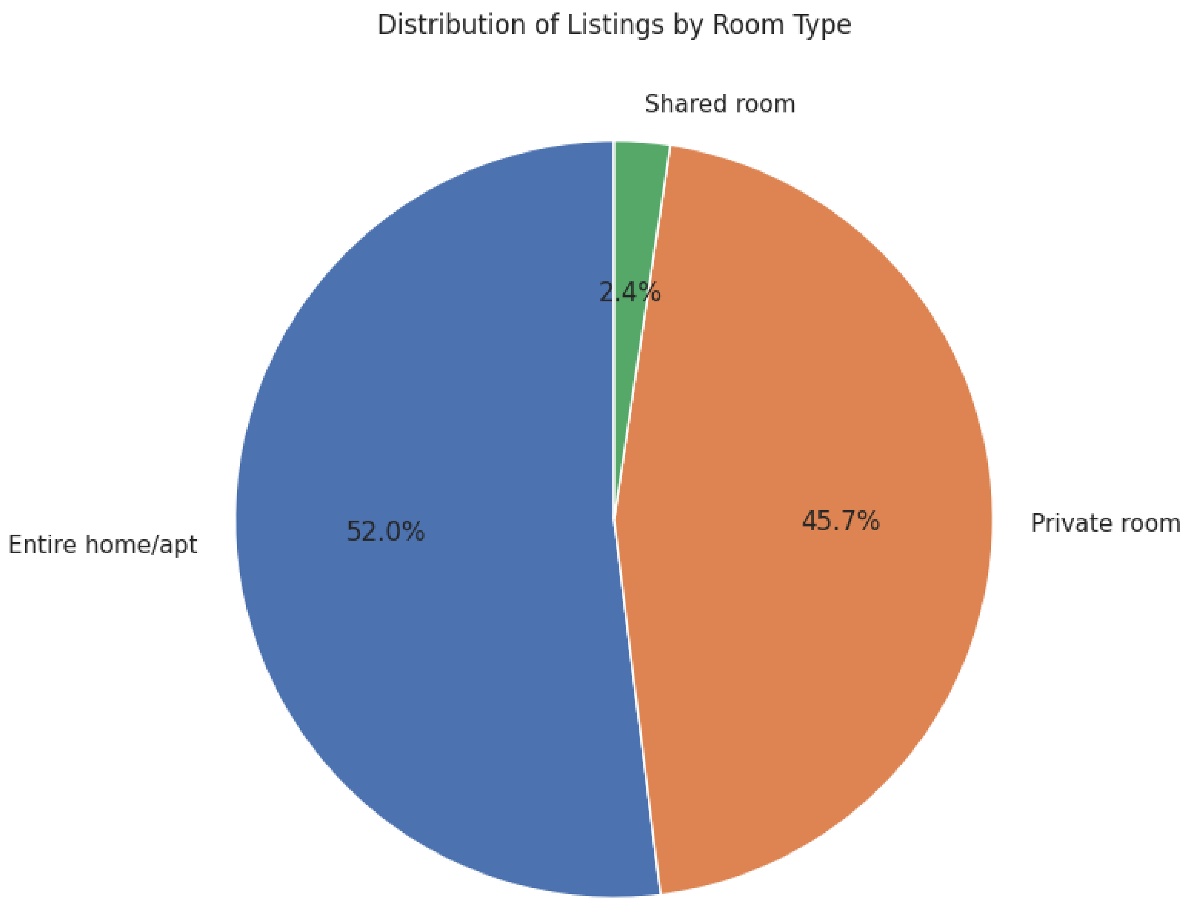
Room Type
Most listings are for "Entire home/apt," followed by "Private room," with fewer listings for "Shared room."
In [21]:

```
# Plotting the distribution of listings by room type
plt.figure(figsize=(10, 5))
sns.countplot(x='room_type', data=nyc_airbnb_df)
plt.title("Distribution of Listings by Room Type")
plt.xlabel("Room Type")
plt.ylabel("Number of Listings")
plt.show()
```



In [22]:

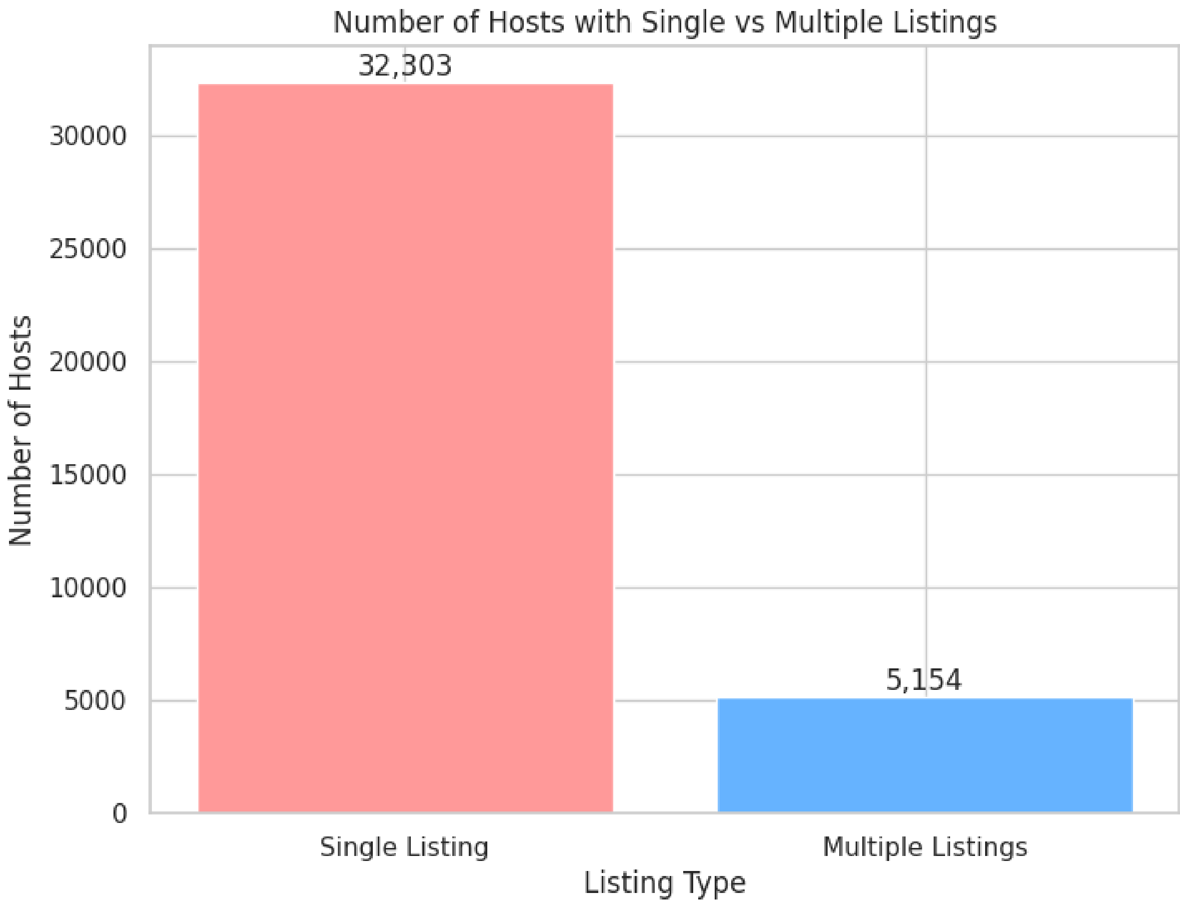
```
# Calculate the counts for each room type
room_type_counts = nyc_airbnb_df['room_type'].value_counts()

# Create the pie chart
plt.figure(figsize=(8, 8))  # Adjust figure size as needed
plt.pie(room_type_counts, labels=room_type_counts.index, autopct='%1.1f%%', startangle=90)
plt.title("Distribution of Listings by Room Type")
plt.show()
```



Numerical Analysis
Calculate host listings count manually

In [23]:

```
# Group by 'host_id' and count the number of listings for each host
host_listings_count = nyc_airbnb_df.groupby('host_id')['id'].count()

# Create a new column to classify hosts as having a single listing or multiple listings
single_listing = host_listings_count[host_listings_count == 1]
multiple_listings = host_listings_count[host_listings_count > 1]

# Calculate counts
single_listing_count = len(single_listing)
multiple_listings_count = len(multiple_listings)

# Bar chart data
labels = ['Single Listing', 'Multiple Listings']
counts = [single_listing_count, multiple_listings_count]

# Create bar chart
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
plt.bar(labels, counts, color=['#ff9999', '#66b3ff'])
plt.title('Number of Hosts with Single vs Multiple Listings')
plt.xlabel('Listing Type')
plt.ylabel('Number of Hosts')

# Add count labels on top of bars
for i, v in enumerate(counts):
    plt.text(i, v + 2, f'{v:,}', ha='center', va='bottom')  # Adjust positioning as needed

plt.show()
```



In [24]:

```
# Group by 'host_id' and count the number of listings for each host
host_listings_count = nyc_airbnb_df.groupby('host_id')['id'].count()

print(host_listings_count.value_counts().sort_index())
```

```
id
1       32303
2        3329
3         951
4         360
5         169
6          95
7          57
8          52
9          26
10         21
11         10
12         15
13         10
14          5
15          5
16          1
17          4
18          3
19          1
20          2
21          1
23          3
25          2
26          1
27          1
28          2
29          1
30          1
31          2
32          1
```

```
33        3
34        2
37        1
39        1
43        1
47        1
49        2
50        1
52        2
65        1
87        1
91        1
96        2
103       1
121       1
232       1
327       1
Name: count, dtype: int64
```

Calculate outliers for number of listings per host

There are so many at 1 that below or above 1 is an outlier.

In [25]:

```
host_listings_count.info()
```

```
<class 'pandas.core.series.Series'>
Index: 37457 entries, 2438 to 274321313
Series name: id
Non-Null Count  Dtype
--------------  -----
37457 non-null  int64
dtypes: int64(1)
memory usage: 585.3 KB
```

In [26]:

```
host_listings_count.describe()
```

Out[26]:

	id
count	37457.000000
mean	1.305363
std	2.760747
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	327.000000

dtype: float64

In [27]:

```
# Calculate outlier thresholds using IQR
Q1 = host_listings_count.quantile(0.25)
Q3 = host_listings_count.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Print outlier thresholds
print(f"Lower Outlier Threshold: {lower_bound}")
print(f"Upper Outlier Threshold: {upper_bound}")
```

```
Lower Outlier Threshold: 1.0
Upper Outlier Threshold: 1.0
```

Calculated Host Listings Count

- As a note, calculated_host_listings_count shows that there are 327 listings with 1 host that has 327 listings denoted for each entry. So there is only one host with 327, but if count, then it appears 327 times. So, if need to count, can't use it like this.

In [28]:

```
print(nyc_airbnb_df['calculated_host_listings_count'].value_counts().sort_index())
```

```
calculated_host_listings_count
1      32303
2       6658
3       2853
4       1440
5        845
6        570
7        399
8        416
9        234
10       210
11       110
12       180
13       130
14        70
15        75
16        16
17        68
18        54
19        19
20        40
21        21
23        69
25        50
26        26
27        27
28        56
29        29
30        30
31        62
32        32
33        99
34        68
37        37
39        39
43        43
47        47
49        98
50        50
52       104
65        65
87        87
91        91
96       192
103      103
121      121
232      232
327      327
Name: count, dtype: int64
```

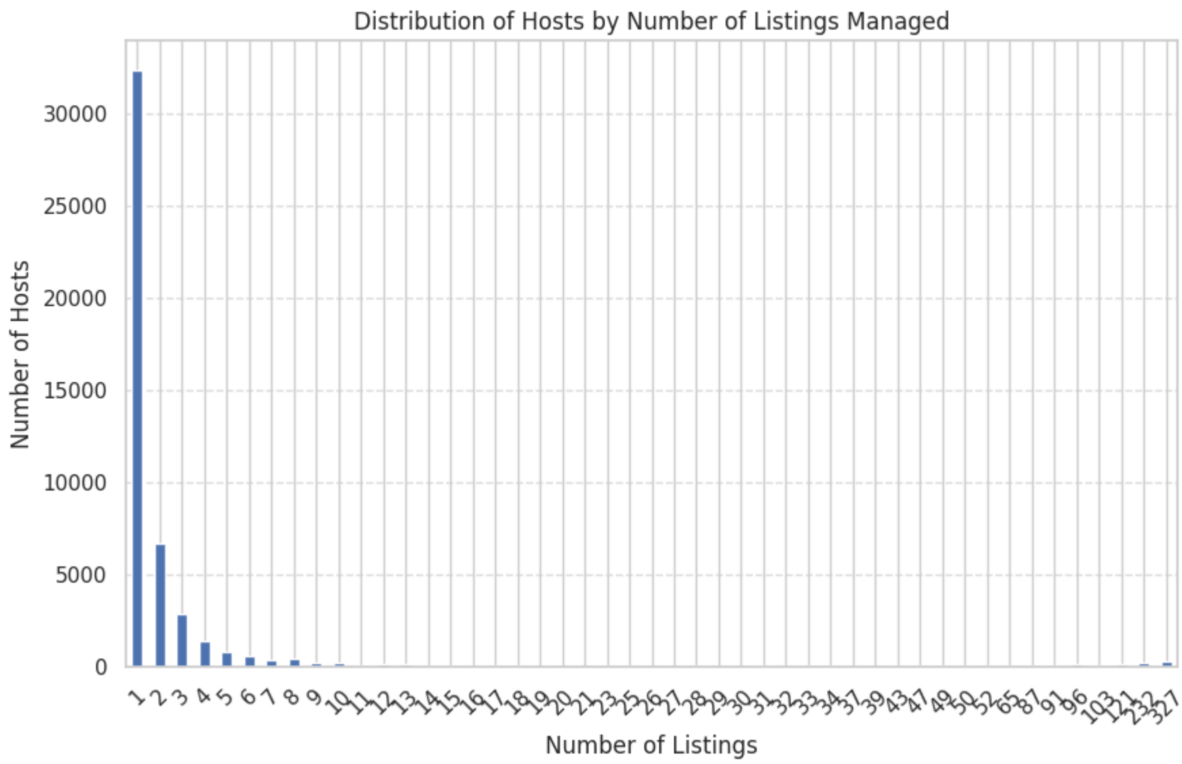
The number of listings that have hosts with this many listings.

- 32,303 listings have 1 listing per host.
- 327 listings have 327 listed but this is all for 1 host.

This isn't a particularly useful visualisation.

In [29]:

```
# Distribution of hosts based on the number of listings they manage
host_listings_dist = nyc_airbnb_df['calculated_host_listings_count'].value_counts()

plt.figure(figsize=(10, 6))
host_listings_dist.sort_index().plot(kind='bar')
plt.title('Distribution of Hosts by Number of Listings Managed')
plt.xlabel('Number of Listings')
plt.ylabel('Number of Hosts')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



In [30]:

```
calc_host_listings_count = nyc_airbnb_df['calculated_host_listings_count']
calc_host_listings_count.describe()
```

Out[30]:

	calculated_host_listings_count
count	48895.000000
mean	7.143982
std	32.952519
min	1.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	327.000000

dtype: float64

Remove Outliers

In [31]:

```
# Calculate quartiles
Q1 = nyc_airbnb_df['calculated_host_listings_count'].quantile(0.25)
Q3 = nyc_airbnb_df['calculated_host_listings_count'].quantile(0.75)
IQR = Q3 - Q1

# Calculate upper and lower bounds
upper_bound = Q3 + 1.5 * IQR
lower_bound = Q1 - 1.5 * IQR

print(f"Lower Outlier Threshold: {lower_bound}")
print(f"Upper Outlier Threshold: {upper_bound}")
```

```
# Filter data based on IQR bounds
filtered_data_iqr = nyc_airbnb_df[
    (nyc_airbnb_df['calculated_host_listings_count'] >= lower_bound) &
    (nyc_airbnb_df['calculated_host_listings_count'] <= upper_bound)
]
```

```
Lower Outlier Threshold: -0.5
Upper Outlier Threshold: 3.5
```

In [32]:

```
def calculate_host_characteristics(df):
    """Calculates host characteristics for a given DataFrame."""
    unique_hosts = df['host_id'].nunique()
    multi_listing_hosts = df[df['calculated_host_listings_count'] > 1]['host_id'].nunique()
    hosts_with_one_listing = unique_hosts - multi_listing_hosts
    average_reviews_per_host = df.groupby('host_id')['reviews_per_month'].mean().mean()
    always_available_hosts = df[df['availability_365'] == 365]['host_id'].nunique()

    return {
        "Total Listings": len(df),
        "Unique Hosts": unique_hosts,
        "Hosts with 1 Listing": hosts_with_one_listing,
        "Hosts with More Than 1 Listing": multi_listing_hosts,
        "Average Reviews Per Host": average_reviews_per_host,
        "Hosts with Full Year Availability": always_available_hosts
    }

# Calculate characteristics for original data
original_characteristics = calculate_host_characteristics(nyc_airbnb_df)

# Calculate characteristics for filtered data
filtered_characteristics = calculate_host_characteristics(filtered_data_iqr)

# Set display options to prevent line wrapping
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)

# Create comparison table
comparison_table = pd.DataFrame({
    "Metric": list(original_characteristics.keys()),
    "Before Outlier Removal": list(original_characteristics.values()),
    "After Outlier Removal": list(filtered_characteristics.values())
})

# Format float columns to 2 decimal places
comparison_table_formatted = comparison_table.style.format({
    "Before Outlier Removal": "{:.2f}",
    "After Outlier Removal": "{:.2f}"
})

# Display the formatted table using display()
display(comparison_table_formatted)
```

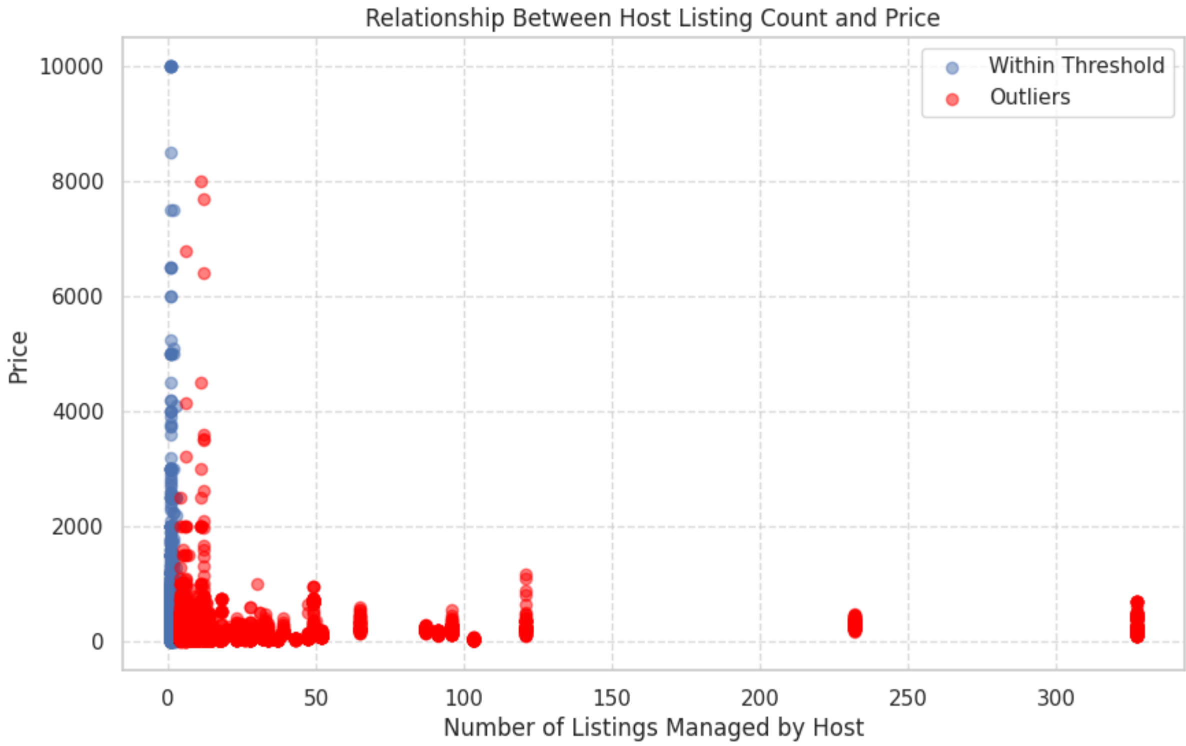
	Metric	Before Outlier Removal	After Outlier Removal
0	Total Listings	48895.00	41814.00
1	Unique Hosts	37457.00	36583.00
2	Hosts with 1 Listing	32303.00	32303.00
3	Hosts with More Than 1 Listing	5154.00	4280.00
4	Average Reviews Per Host	1.03	1.02
5	Hosts with Full Year Availability	894.00	741.00

Price by Listings Count (STEP 4)

In [33]:

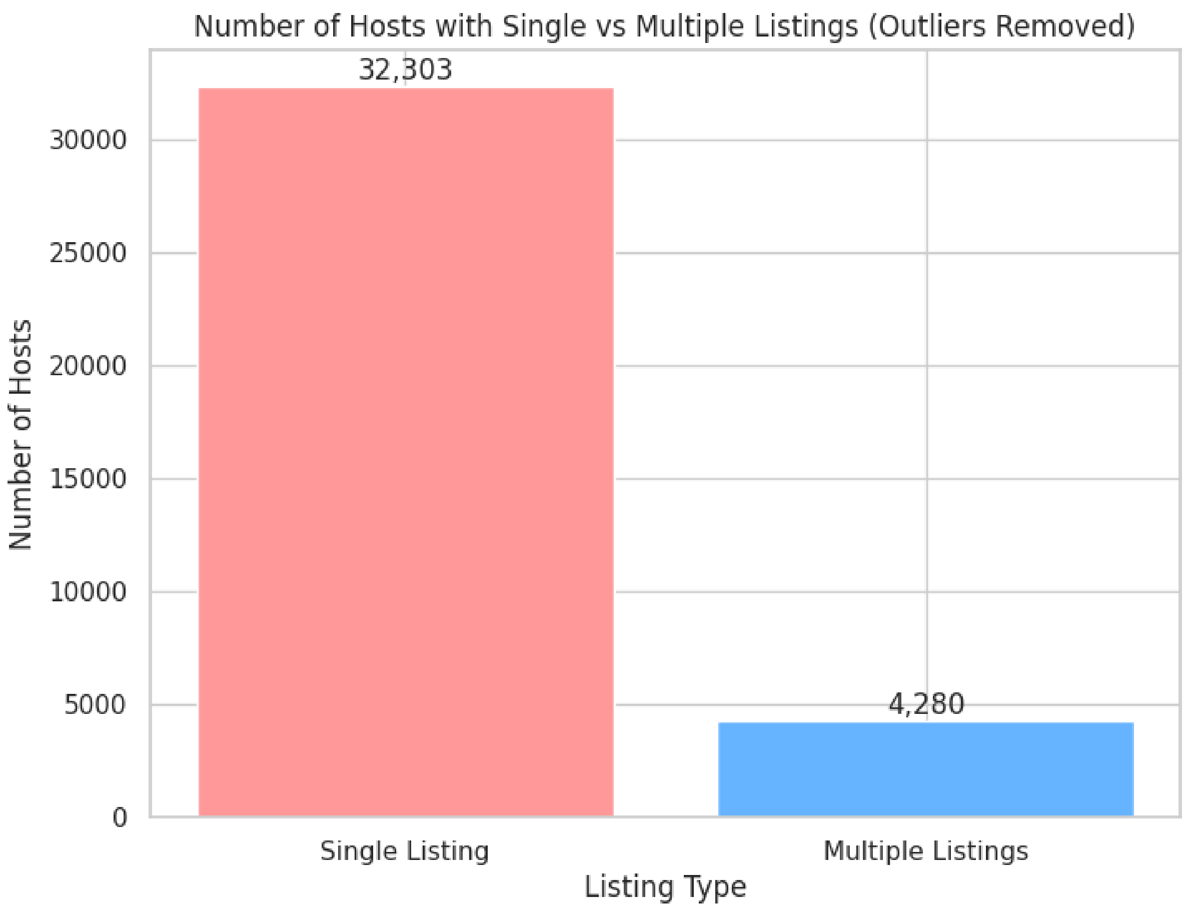
```
# Set the upper outlier threshold
upper_threshold = upper_bound  # Adjust this value as needed

# Scatter plot with outlier threshold
```

```
plt.figure(figsize=(10, 6))

# Plot data points within the threshold
plt.scatter(
    nyc_airbnb_df[nyc_airbnb_df['calculated_host_listings_count'] <=
upper_threshold]['calculated_host_listings_count'],
    nyc_airbnb_df[nyc_airbnb_df['calculated_host_listings_count'] <=
upper_threshold]['price'],
    alpha=0.5,
    label='Within Threshold'
)

# Plot data points beyond the threshold (outliers)
plt.scatter(
    nyc_airbnb_df[nyc_airbnb_df['calculated_host_listings_count'] >
upper_threshold]['calculated_host_listings_count'],
    nyc_airbnb_df[nyc_airbnb_df['calculated_host_listings_count'] > upper_threshold]['price'],
    alpha=0.5,
    color='red',  # Highlight outliers in red
    label='Outliers'
)

plt.title('Relationship Between Host Listing Count and Price')
plt.xlabel('Number of Listings Managed by Host')
plt.ylabel('Price')
plt.grid(linestyle='--', alpha=0.7)
plt.legend()  # Add legend to distinguish data points
plt.show()

# Correlation between host listing count and price
correlation = nyc_airbnb_df[['calculated_host_listings_count', 'price']].corr().iloc[0, 1]
print("Correlation:", correlation)

# Correlation below the outlier threshold
filtered_data = nyc_airbnb_df[nyc_airbnb_df['calculated_host_listings_count'] <=
upper_threshold]
correlation_below_threshold = filtered_data[['calculated_host_listings_count',
'price']].corr().iloc[0, 1]
print("Correlation below threshold:", correlation_below_threshold)
```




```
Correlation: 0.057471688368068104
Correlation below threshold: -0.055767553095915426
In [34]:

# Group by 'host_id' and count the number of listings for each host
```

```
host_listings_count_filtered = filtered_data_iqr.groupby('host_id')['id'].count()

# Create a new column to classify hosts as having a single listing or multiple listings
single_listing_filtered = host_listings_count_filtered[host_listings_count_filtered == 1]
multiple_listings_filtered = host_listings_count_filtered[host_listings_count_filtered > 1]

# Calculate counts
single_listing_count_filtered = len(single_listing_filtered)
multiple_listings_count_filtered = len(multiple_listings_filtered)

# Bar chart data
labels = ['Single Listing', 'Multiple Listings']
counts = [single_listing_count_filtered, multiple_listings_count_filtered]

# Create bar chart
plt.figure(figsize=(8, 6))  # Adjust figure size as needed
plt.bar(labels, counts, color=['#ff9999', '#66b3ff'])
plt.title('Number of Hosts with Single vs Multiple Listings (Outliers Removed)')
plt.xlabel('Listing Type')
plt.ylabel('Number of Hosts')

# Add count labels on top of bars
for i, v in enumerate(counts):
    plt.text(i, v + 2, f'{v:,}', ha='center', va='bottom')  # Adjust positioning as needed

plt.show()
```



Price (STEP 1)
Statisical Descriptive Analysis on Price
In [35]:

```
# Mean price
print(f"Mean price: ${nyc_airbnb_df['price'].mean():.2f}")

# Median price
print(f"Median price: ${nyc_airbnb_df['price'].median():.2f}")

# Price range
```

```
print(f"Price range: ${nyc_airbnb_df['price'].min()} to ${nyc_airbnb_df['price'].max()}")

# Calculate quartiles
Q1 = nyc_airbnb_df['price'].quantile(0.25)
print(f"First Quartile (Q1) of price: ${Q1:.2f}")

Q3 = nyc_airbnb_df['price'].quantile(0.75)
print(f"Third Quartile (Q3) of price: ${Q3:.2f}")

# Calculate IQR
IQR = Q3 - Q1
print(f"Interquartile Range (IQR) of price: ${IQR:.2f}")

# Calculate outlier bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
print(f"Lower bound of price ouliers: ${lower_bound:.2f}")
print(f"Upper bound of price outliers: ${upper_bound:.2f}")

# Number of outliers
outlier_below = nyc_airbnb_df[nyc_airbnb_df['price'] < (Q1 - 1.5 * IQR)]
outlier_above = nyc_airbnb_df[nyc_airbnb_df['price'] > (Q3 + 1.5 * IQR)]
print(f"Number of outliers below the IQR: {len(outlier_below)}")
print(f"Number of outliers above the IQR: {len(outlier_above)}")

# Total number of non-null price rows
non_null_price_count = nyc_airbnb_df['price'].notnull().sum()
print("Out of total number of non-null price rows:", non_null_price_count)
```

```
Mean price: $152.72
Median price: $106.00
Price range: $0 to $10000
First Quartile (Q1) of price: $69.00
Third Quartile (Q3) of price: $175.00
Interquartile Range (IQR) of price: $106.00
Lower bound of price ouliers: $-90.00
Upper bound of price outliers: $334.00
Number of outliers below the IQR: 0
Number of outliers above the IQR: 2972
Out of total number of non-null price rows: 48895
```

Review the high skew outliers (numeric).

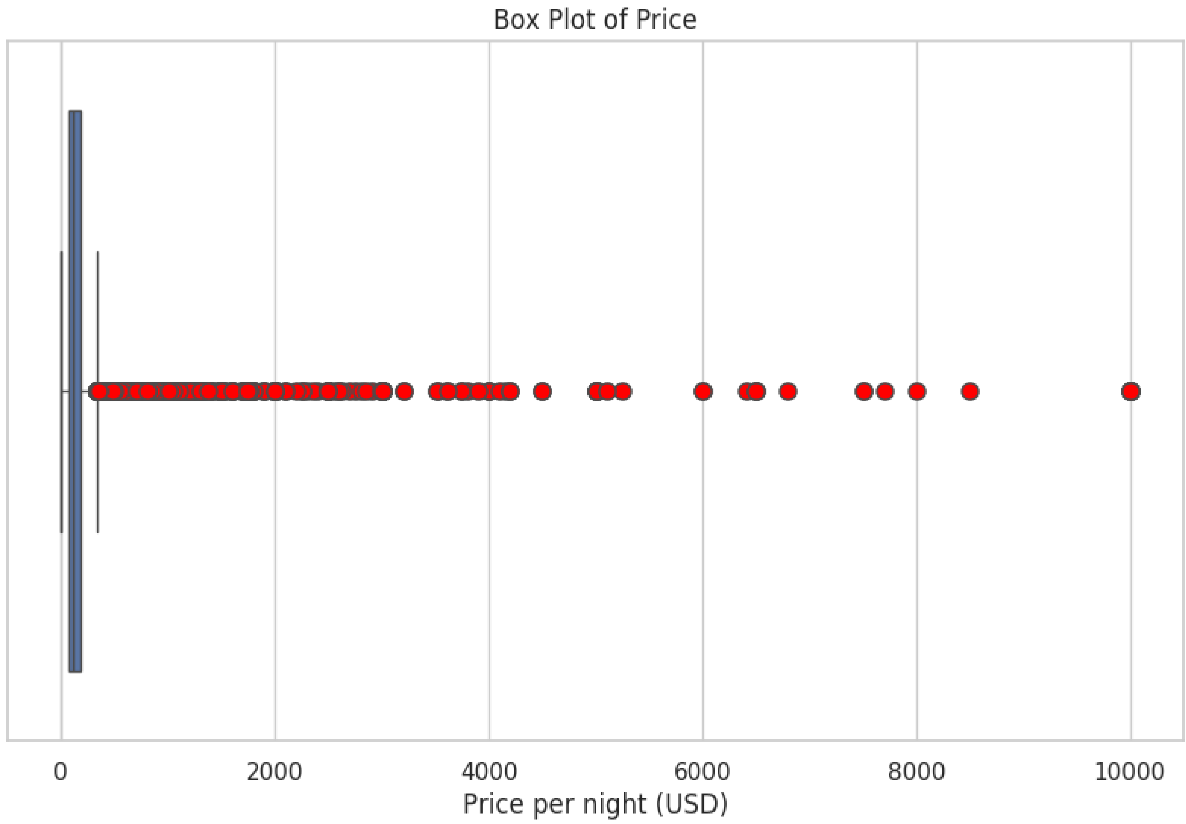
- price
- minimum_nights
- number_of_reviews
- reviews_per_month
- calculated_host_listings_count

Price Box Plot

High priced listings could be luxury properties or in prime locations. Removing them will impact revenue. But they also skew average prices, making it harder to price new listings. ACTION: Better understand what generates a higher price in bivariate. Is it a location, for instance?

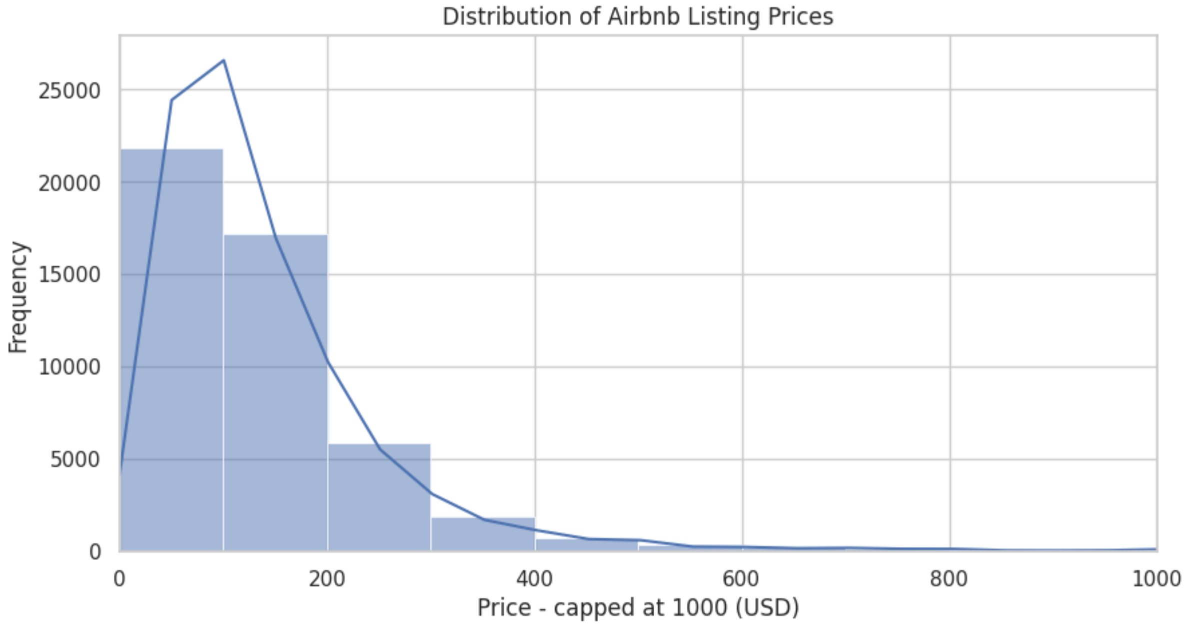
In [36]:

```
# Create the box plot
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
sns.boxplot(x='price', data=nyc_airbnb_df,  # Use x='price' for horizontal orientation
            flierprops={'marker': 'o', 'markerfacecolor': 'red', 'markersize': 8}) # Customize
outlier markers
plt.title('Box Plot of Price')
plt.xlabel('Price per night (USD)')  # Change ylabel to xlabel
plt.show()
```



Price Distribution (Zoomed in to 0-1000 USD)
As seen in the calculations above, 334 USD is the upper bound above which there are 2,972 price outliers out of 48,895 non-null prices.
The zoomed in histogram shows that most listings are indeed priced below 400 USD, with a high concentration at lower prices.
In [37]:

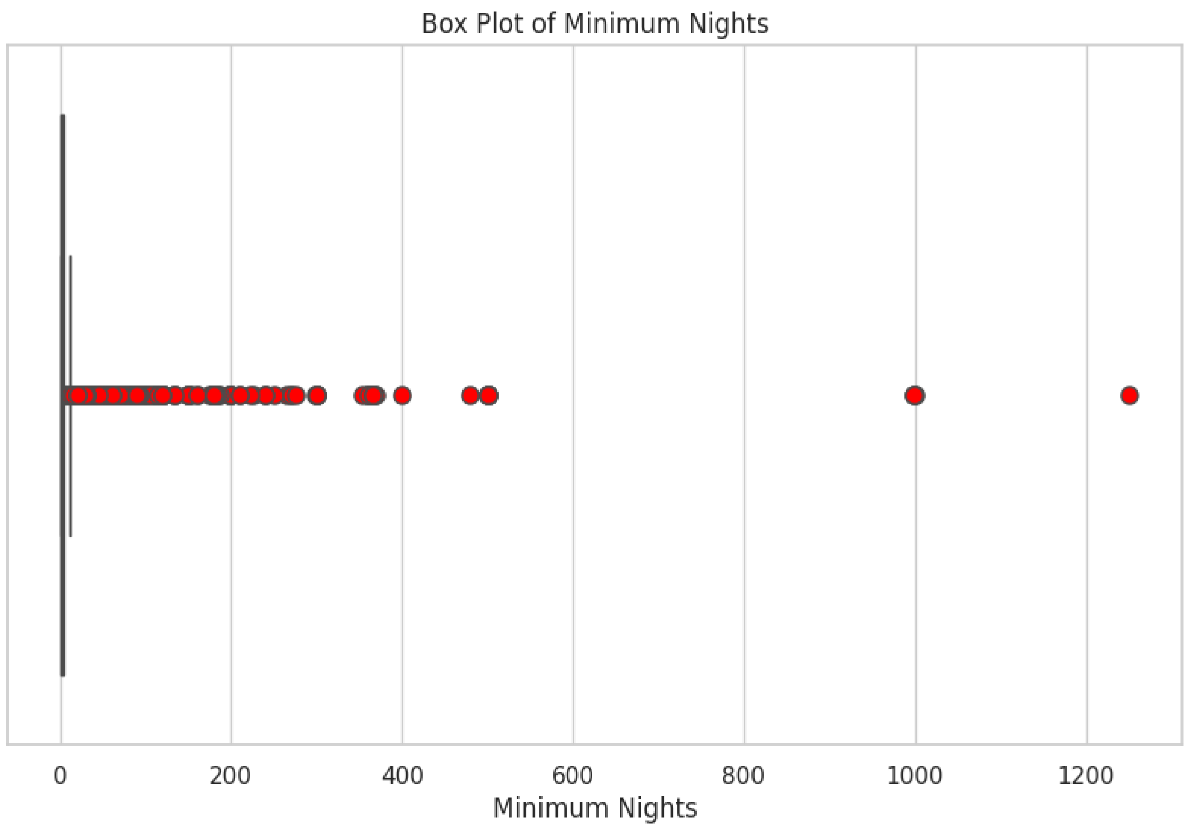
```
# Plot the distribution of 'price' with a histogram capped at 1000 USD
plt.figure(figsize=(10, 5))
sns.histplot(nyc_airbnb_df['price'], bins=100, kde=True)
plt.xlim(0, 1000)  # Limit x-axis for a clearer view (removes extreme outliers)
plt.title("Distribution of Airbnb Listing Prices")
plt.xlabel("Price - capped at 1000 (USD)")
plt.ylabel("Frequency")
plt.show()
```



Minimum nights box plot

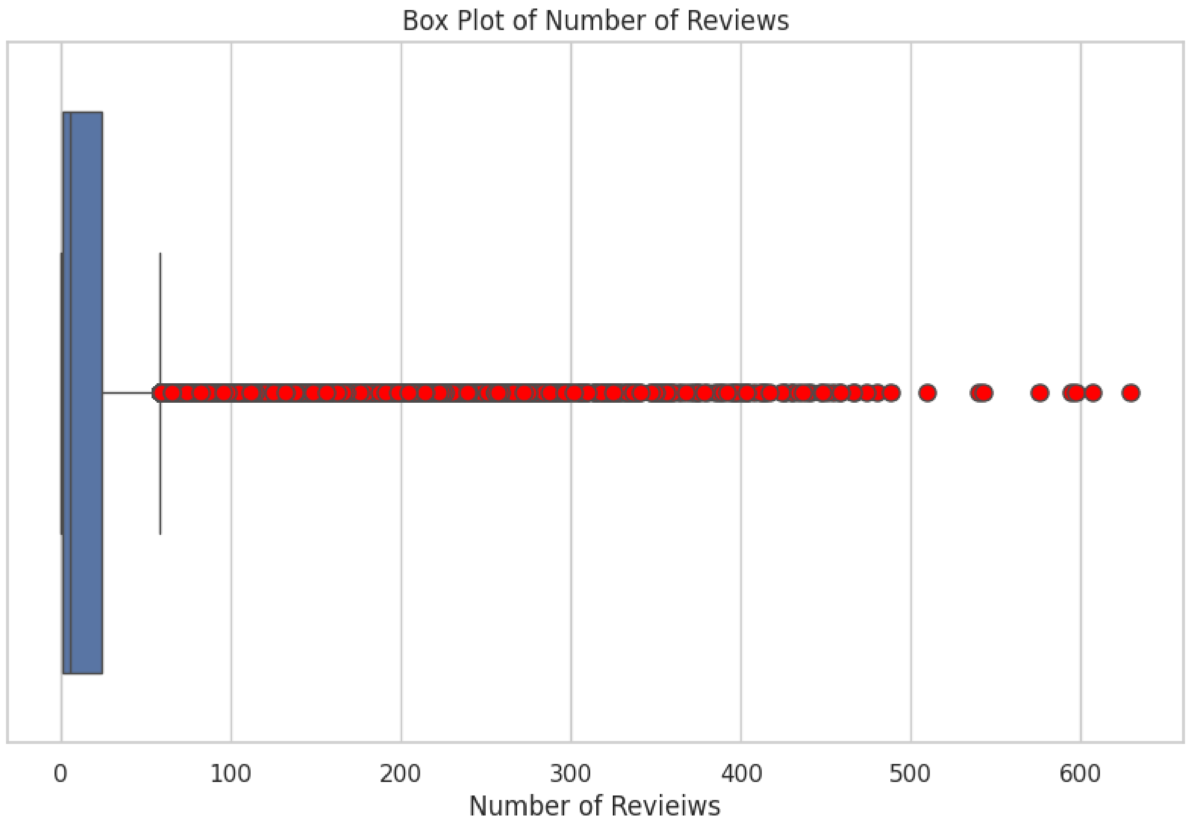
Very high minimum nights could be targeting specific customer segments like long-term rental.
ACTION: May want to split this into rental periods.
In [38]:

```
# Create the box plot
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
sns.boxplot(x='minimum_nights', data=nyc_airbnb_df,  # Use x='price' for horizontal
orientation
            flierprops={'marker': 'o', 'markerfacecolor': 'red', 'markersize': 8}) # Customize
outlier markers
plt.title('Box Plot of Minimum Nights')
plt.xlabel('Minimum Nights')  # Change ylabel to xlabel
plt.show()
```



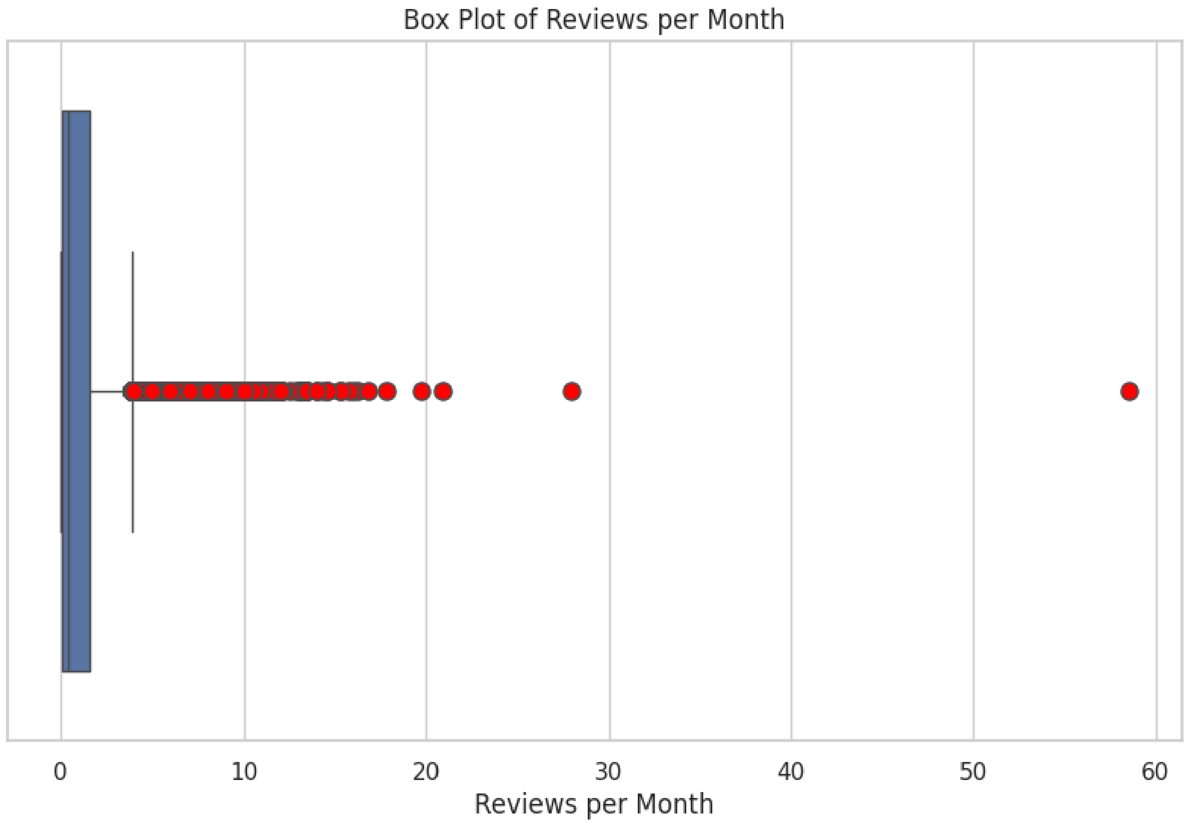
Number of Reviews
High number of reviews could mean popular listings or professional hosts (number listings) or rental for fewer days (minimum nights).
ACTION: Could compare to number of listings (i.e. professional hosts?), or see if more reviews have higher or lower prices, or see if it's just related to minimum nights (more frequently rented perhaps).
In [39]:

```
# Create the box plot
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
sns.boxplot(x='number_of_reviews', data=nyc_airbnb_df,  # Use x='price' for horizontal orientation
            flierprops={'marker': 'o', 'markerfacecolor': 'red', 'markersize': 8}) # Customize outlier markers
plt.title('Box Plot of Number of Reviews')
plt.xlabel('Number of Revieiws')  # Change ylabel to xlabel
plt.show()
```



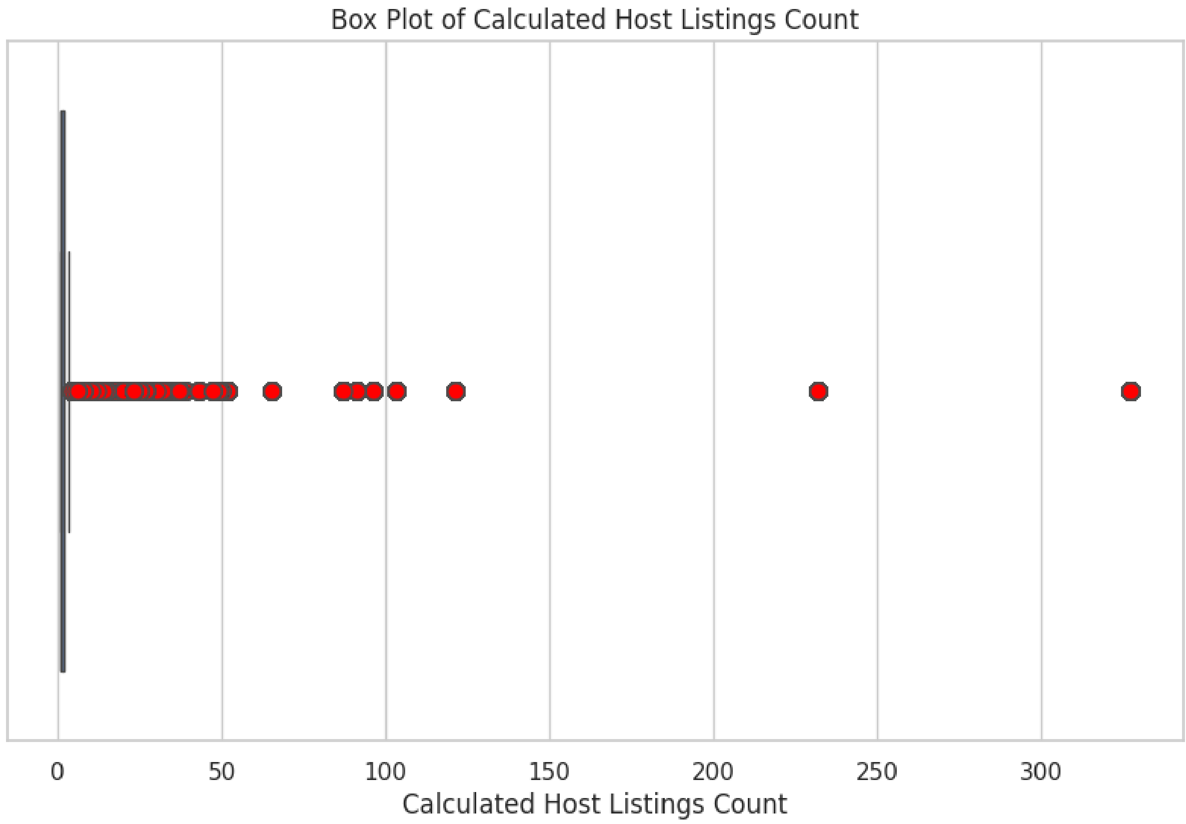
Reviews per month
High number of reviews per month could mean popular listings or professional hosts (number listings) or rental for fewer days (minimum nights).
ACTION: Could compare to number of listings (i.e. professional hosts?), or see if more reviews have higher or lower prices, or see if it's just related to minimum nights (more frequently rented perhaps).
In [40]:

```
# Create the box plot
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
sns.boxplot(x='reviews_per_month', data=nyc_airbnb_df,  # Use x='price' for horizontal orientation
            flierprops={'marker': 'o', 'markerfacecolor': 'red', 'markersize': 8}) # Customize outlier markers
plt.title('Box Plot of Reviews per Month')
plt.xlabel('Reviews per Month')  # Change ylabel to xlabel
plt.show()
```



Calculated Host Listings Count
High number of listings probably means professional hosts with multiple properties.
ACTION: Could get a better idea of how many in each group to see how many professional hosts.
In [41]:

```
# Create the box plot
plt.figure(figsize=(10, 6))  # Adjust figure size if needed
sns.boxplot(x='calculated_host_listings_count', data=nyc_airbnb_df,  # Use
x='calculated_host_listings_count' for horizontal orientation
            flierprops={'marker': 'o', 'markerfacecolor': 'red', 'markersize': 8}) # Customize
outlier markers
plt.title('Box Plot of Calculated Host Listings Count')
plt.xlabel('Calculated Host Listings Count')  # Change ylabel to xlabel
plt.show()
```



BIVARIATE ANALYSIS
Oluleye (2023) for analysing two variables:

- crosstab / two-way tables
- pivot table
- scatter plot
- bar chart
- correlation analysis
- pairplots
- boxplots
- histograms

Corrlation Heatmap
A correlation heatmap for the numeric columns helps understand their relationships better.

- +1: Perfect positive corrlation
- +0.5 to +1: Strong positive correlation
- 0 to +0.5: Weak positive correlation
- 0: No correlation
- -0.5 to 0: Weak negative correlation
- -1 to -0.5: Weak negative correlation = -1: Perfect negative correlation

Potential for correlation:

- Could ID and number of reviews be how long someone has been a host?
- ID and host_id are related positively, so there seems to be a relationship in ID creation.
- Number of reviews and reviews per month are positively related that makes sense.

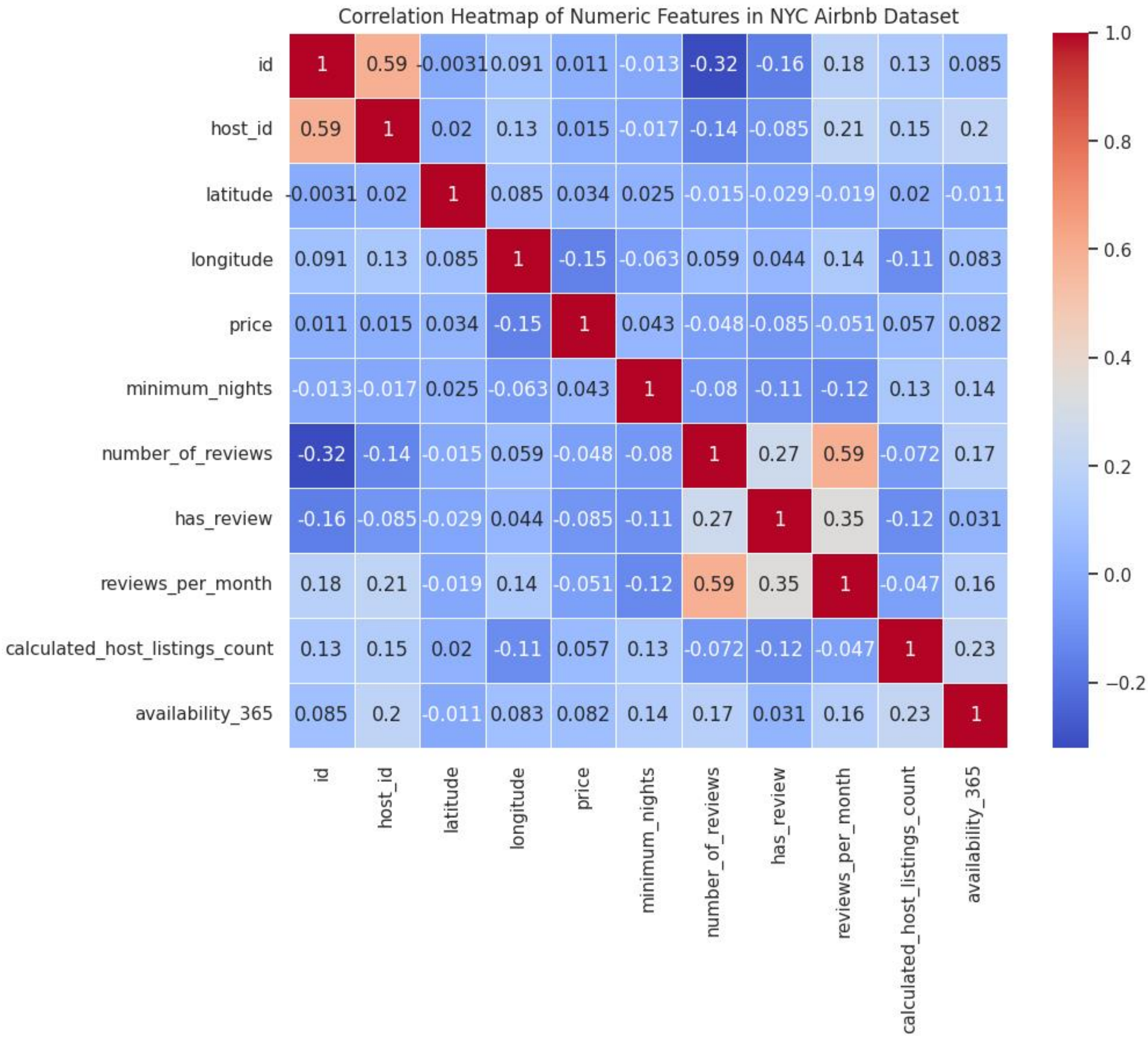
The correlation heatmap indicates the following relationships:

- Latitude and Longitude: They are not correlated with other variables in a meaningful way.
- Calculated Host Listings Count and Availability: Low correlation, suggesting that hosts's availability does not strongly correlate with their number of listings.
- Price and Minimum Nights: Surprisingly, there is no strong correlation between price and minimum_nights, which may be further investigated for insights.

In [42]:

```
# Calculate correlation matrix
correlation_matrix_nyc = nyc_airbnb_df[numeric_columns_nyc].corr()
```

```
# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_nyc, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap of Numeric Features in NYC Airbnb Dataset")
plt.show()
```

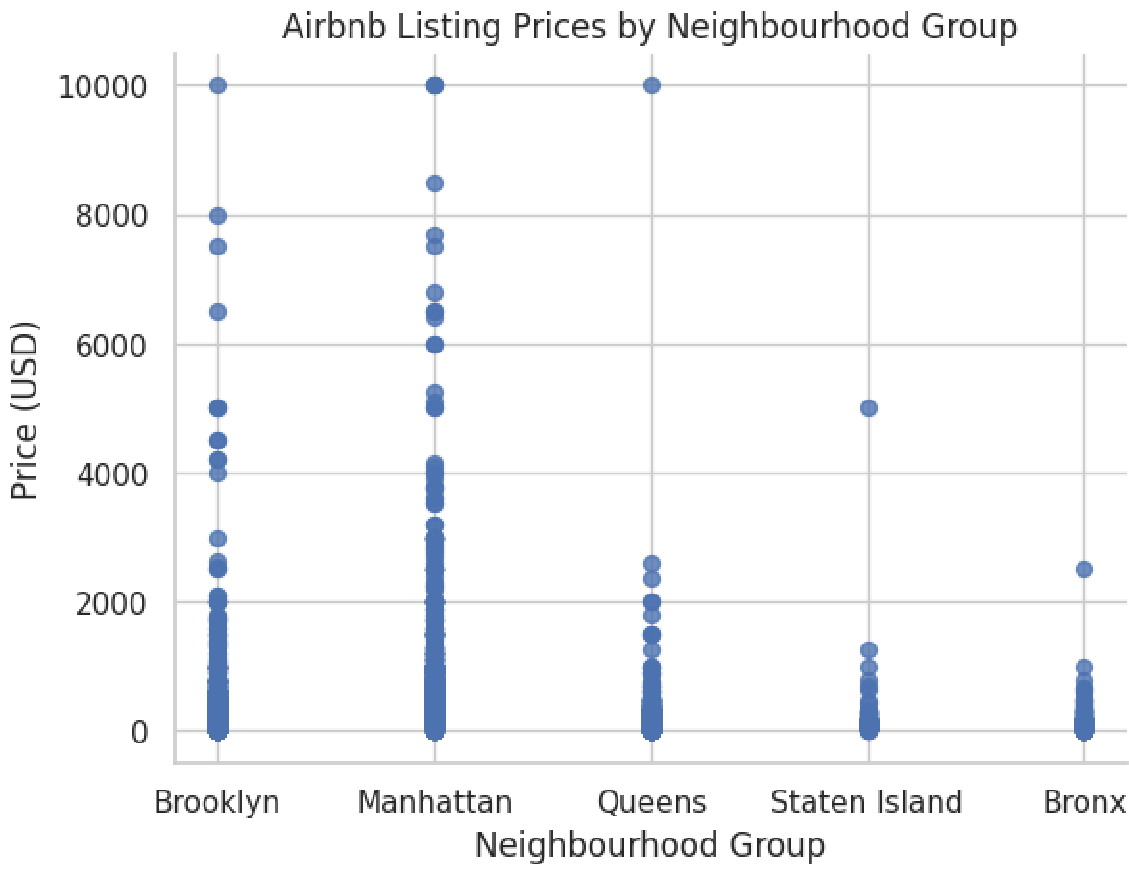


	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	has_review	reviews_per_month	calculated_host_listings_count	availability_365
id	1	0.59	-0.0031	0.091	0.011	-0.013	-0.32	-0.16	0.18	0.13	0.085
host_id	0.59	1	0.02	0.13	0.015	-0.017	-0.14	-0.085	0.21	0.15	0.2
latitude	-0.0031	0.02	1	0.085	0.034	0.025	-0.015	-0.029	-0.019	0.02	-0.011
longitude	0.091	0.13	0.085	1	-0.15	-0.063	0.059	0.044	0.14	-0.11	0.083
price	0.011	0.015	0.034	-0.15	1	0.043	-0.048	-0.085	-0.051	0.057	0.082
minimum_nights	-0.013	-0.017	0.025	-0.063	0.043	1	-0.08	-0.11	-0.12	0.13	0.14
number_of_reviews	-0.32	-0.14	-0.015	0.059	-0.048	-0.08	1	0.27	0.59	-0.072	0.17
has_review	-0.16	-0.085	-0.029	0.044	-0.085	-0.11	0.27	1	0.35	-0.12	0.031
reviews_per_month	0.18	0.21	-0.019	0.14	-0.051	-0.12	0.59	0.35	1	-0.047	0.16
calculated_host_listings_count	0.13	0.15	0.02	-0.11	0.057	0.13	-0.072	-0.12	-0.047	1	0.23
availability_365	0.085	0.2	-0.011	0.083	0.082	0.14	0.17	0.031	0.16	0.23	1

Price by Location (STEP 2)
Neighbourhood Group Versus Price (Zoomed Out)

In [43]:

```
nyc_airbnb_df.plot(kind='scatter', x='neighbourhood_group', y='price', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
plt.title("Airbnb Listing Prices by Neighbourhood Group")
plt.xlabel("Neighbourhood Group")
plt.ylabel("Price (USD)")
plt.show()
```

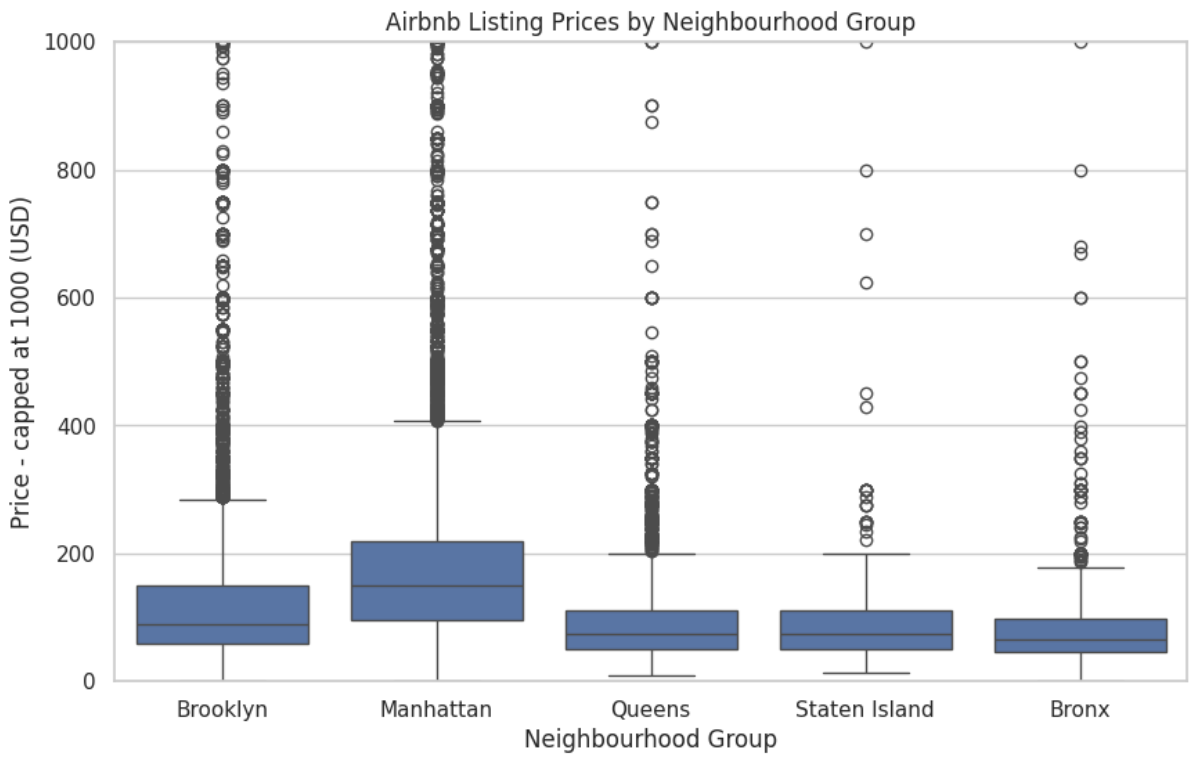


Airbnb Listing Prices by Neighbourhood Group - Zoomed In
Examine how prices vary across different neighbourhood groups to see if location significantly influences price.
The boxplot indicates the following trends:

- Manhattan listings generally have the highest median prices, with more variability and outliers at the high end.
- Brooklyn follows Manhattan with moderately high prices. Queens, the Bronx, and Staten Island have lower median prices, with fewer outliers.
- This suggests that location (specifically, neighborhood) influences number of listings and prices.

Price by Neighbourhood Group (Zoomed in 0-1000 USD)
In [44]:

```
# Boxplot to show price distribution by neighbourhood group
plt.figure(figsize=(10, 6))
sns.boxplot(x='neighbourhood_group', y='price', data=nyc_airbnb_df)
plt.ylim(0, 1000)  # Limit y-axis for a clearer view (removes extreme outliers)
plt.title("Airbnb Listing Prices by Neighbourhood Group")
plt.xlabel("Neighbourhood Group")
plt.ylabel("Price - capped at 1000 (USD)")
plt.show()
```

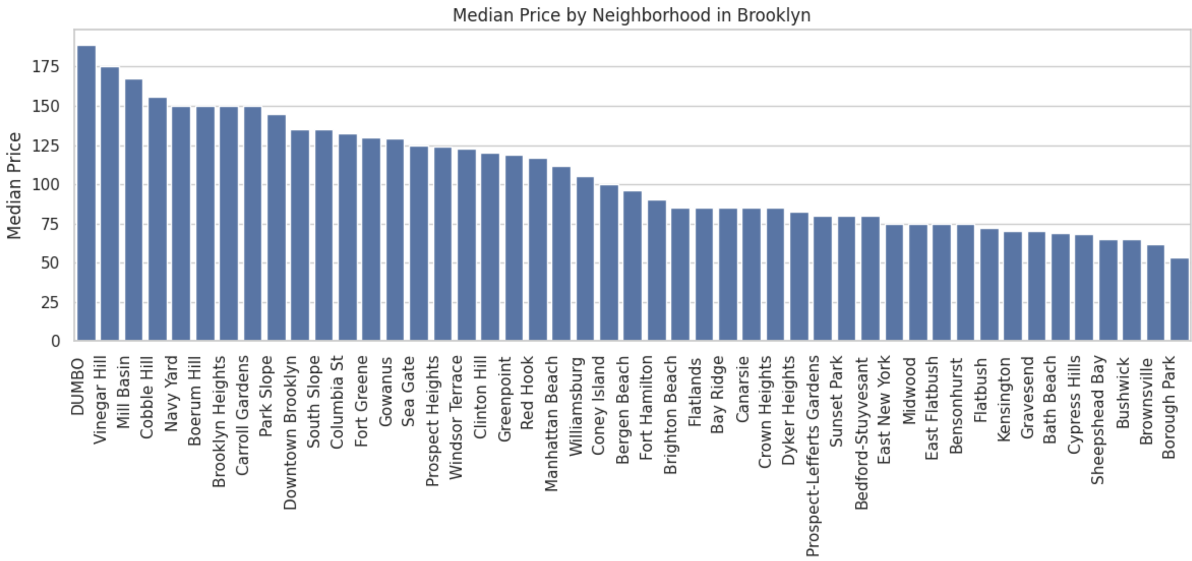


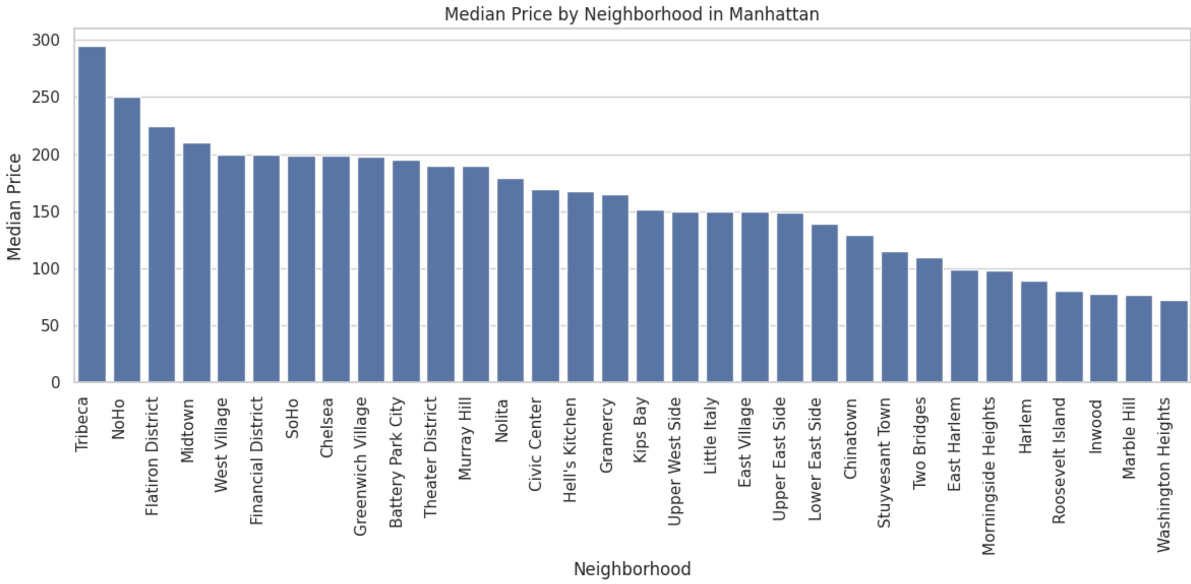
Median Price by Neighbourhood by Neighbourhood Group

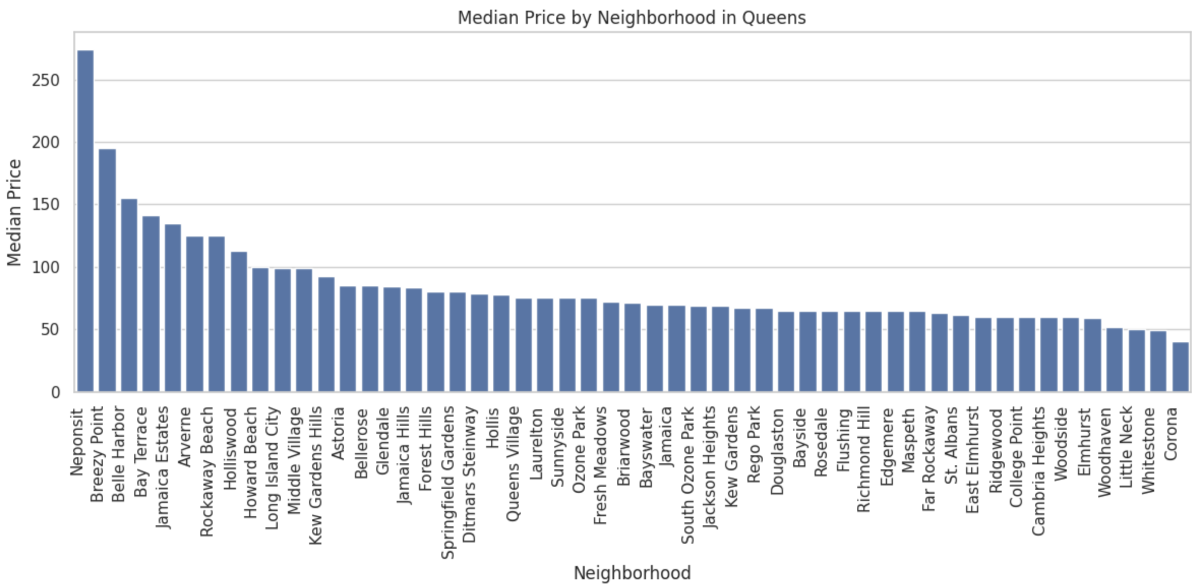
Median rather than mean as less succeptivle to outliers.

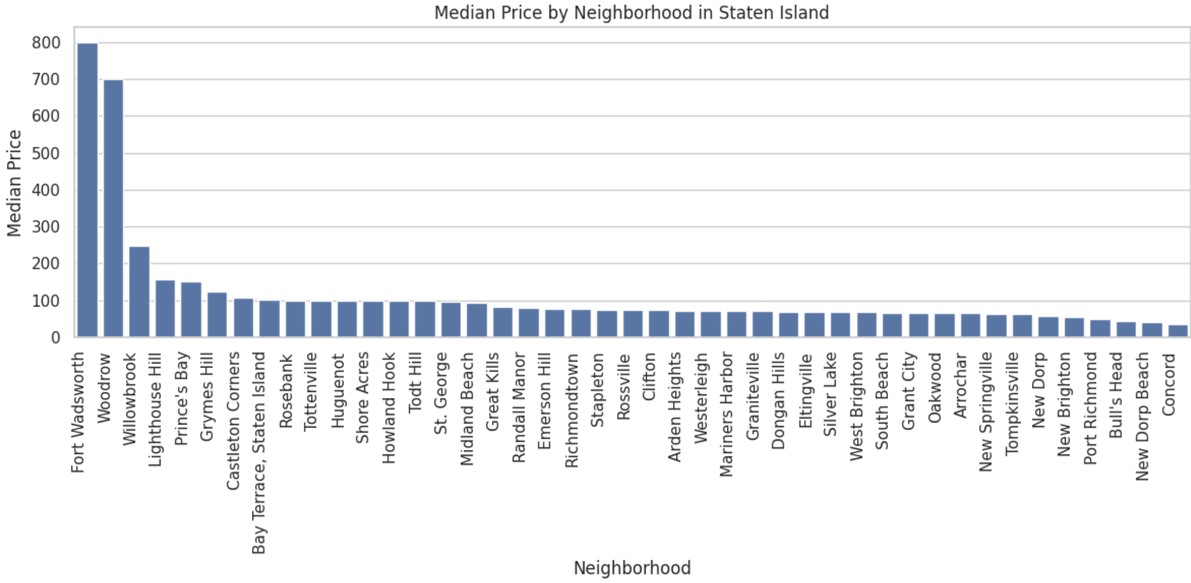
In [45]:

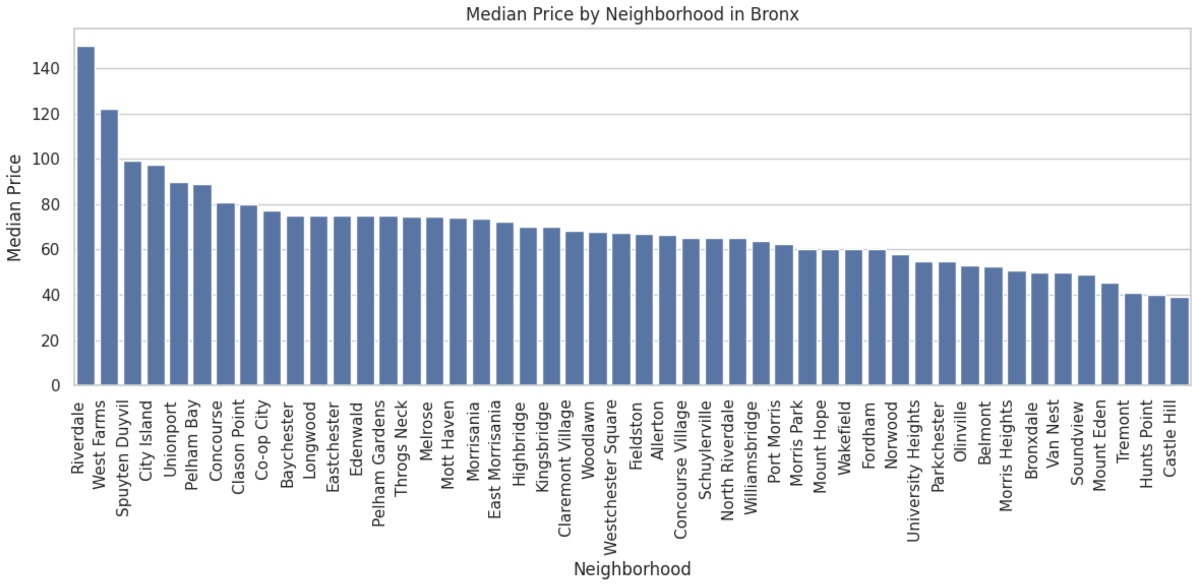
```
# Get unique neighborhood groups
neighborhood_groups = nyc_airbnb_df['neighbourhood_group'].unique()

# Loop through neighborhood groups and create plots
for group in neighborhood_groups:
    # Filter data for the current group
    group_data = nyc_airbnb_df[nyc_airbnb_df['neighbourhood_group'] == group]

    # Calculate average price per neighborhood within the group
    median_price_by_neighborhood =
group_data.groupby('neighbourhood')['price'].median().sort_values(ascending=False)

    # Create the bar chart
    plt.figure(figsize=(12, 6))  # Adjust figure size as needed
    sns.barplot(x=median_price_by_neighborhood.index, y=median_price_by_neighborhood.values)
    plt.title(f'Median Price by Neighborhood in {group}')
    plt.xlabel('Neighborhood')
    plt.ylabel('Median Price')
    plt.xticks(rotation=90, ha='right')  # Rotate x-axis labels for better readability
    plt.tight_layout()
    plt.show()
```


Median Price by Neighborhood in Brooklyn
Median Price
Neighborhood
DUMBO
Vinegar Hill
Mill Basin
Cobble Hill
Navy Yard
Boerum Hill
Brooklyn Heights
Carroll Gardens
Park Slope
Downtown Brooklyn
South Slope
Columbia St
Fort Greene
Gowanus
Sea Gate
Prospect Heights
Windsor Terrace
Clinton Hill
Greenpoint
Red Hook
Manhattan Beach
Williamsburg
Coney Island
Bergen Beach
Fort Hamilton
Brighton Beach
Flatlands
Bay Ridge
Canarsie
Crown Heights
Dyker Heights
Prospect-Lefferts Gardens
Sunset Park
Bedford-Stuyvesant
East New York
Midwood
East Flatbush
Bensonhurst
Flatbush
Kensington
Gravesend
Bath Beach
Cypress Hills
Sheepshead Bay
Bushwick
Brownsville
Borough Park


Median Price by Neighborhood in Manhattan
Median Price
Neighborhood
Tribeca
NoHo
Flatiron District
Midtown
West Village
Financial District
SoHo
Chelsea
Greenwich Village
Battery Park City
Theater District
Murray Hill
Nolita
Civic Center
Hell's Kitchen
Gramercy
Kips Bay
Upper West Side
Little Italy
East Village
Upper East Side
Lower East Side
Chinatown
Stuyvesant Town
Two Bridges
East Harlem
Morningside Heights
Harlem
Roosevelt Island
Inwood
Marble Hill
Washington Heights


Median Price by Neighborhood in Queens
Median Price
Neighborhood
Neponsit
Breezy Point
Belle Harbor
Bay Terrace
Jamaica Estates
Arverne
Rockaway Beach
Holliswood
Howard Beach
Long Island City
Middle Village
Kew Gardens Hills
Astoria
Bellerose
Glendale
Jamaica Hills
Forest Hills
Springfield Gardens
Ditmars Steinway
Hollis
Queens Village
Laurelton
Sunnyside
Ozone Park
Fresh Meadows
Briarwood
Bayswater
Jamaica
South Ozone Park
Jackson Heights
Kew Gardens
Rego Park
Douglaston
Bayside
Rosedale
Flushing
Richmond Hill
Edgemere
Maspeth
Far Rockaway
St. Albans
East Elmhurst
Ridgewood
College Point
Cambria Heights
Woodside
Elmhurst
Woodhaven
Little Neck
Whitestone
Corona





Top 3 highest priced neighbourhoods per neighbourhood group

What are the most desirable neighbourhoods? Are they real, outliers or just numbers thrown in as holding numbers? Could do further literary research on popular areas in NYC. (Not coding, but articles.)

In [46]:

```
def top_3_highest_price_neighborhoods(df):
    """
    Finds the top 3 neighborhoods with the highest individual listing prices
    within each neighborhood group, and the number of listings at that price.

    Args:
        df: The input DataFrame containing Airbnb listing data.

    Returns:
        A DataFrame with the top 3 neighborhoods, their highest prices, and the
        number of listings at that price for each neighborhood group.
    """

    top_neighborhoods = []
    for group in df['neighbourhood_group'].unique():
        group_data = df[df['neighbourhood_group'] == group]

        # Sort by price in descending order and get the top 3 unique prices
        top_3_prices = group_data['price'].sort_values(ascending=False).unique()[:3]
```

```
        for price in top_3_prices:
            # Find neighborhoods with listings at that price
            neighborhoods_at_price = group_data[group_data['price'] ==
price]['neighbourhood'].unique()

            for neighborhood in neighborhoods_at_price:
                # Count listings at that price in the neighborhood
                count = group_data[(group_data['price'] == price) &
(group_data['neighbourhood'] == neighborhood)].shape[0]
                top_neighborhoods.append([group, neighborhood, price, count])

    return pd.DataFrame(top_neighborhoods, columns=['Neighborhood Group', 'Neighborhood',
'Highest Price', 'Count'])

# Get the top 3 neighborhoods by highest price with counts
top_neighborhoods_df = top_3_highest_price_neighborhoods(nyc_airbnb_df)

# Print the results
print(top_neighborhoods_df)
```

	Neighborhood Group	Neighborhood	Highest Price	Count
0	Brooklyn	Greenpoint	10000	1
1	Brooklyn	Clinton Hill	8000	1
2	Brooklyn	East Flatbush	7500	1
3	Manhattan	Upper West Side	10000	1
4	Manhattan	East Harlem	9999	1
5	Manhattan	Lower East Side	9999	2
6	Manhattan	Tribeca	8500	1
7	Queens	Astoria	10000	1
8	Queens	Bayside	2600	1
9	Queens	Forest Hills	2350	1
10	Staten Island	Randall Manor	5000	1
11	Staten Island	Prince's Bay	1250	1
12	Staten Island	St. George	1000	1
13	Bronx	Riverdale	2500	1
14	Bronx	City Island	1000	1
15	Bronx	Riverdale	800	1

Geospatial Scatterplot

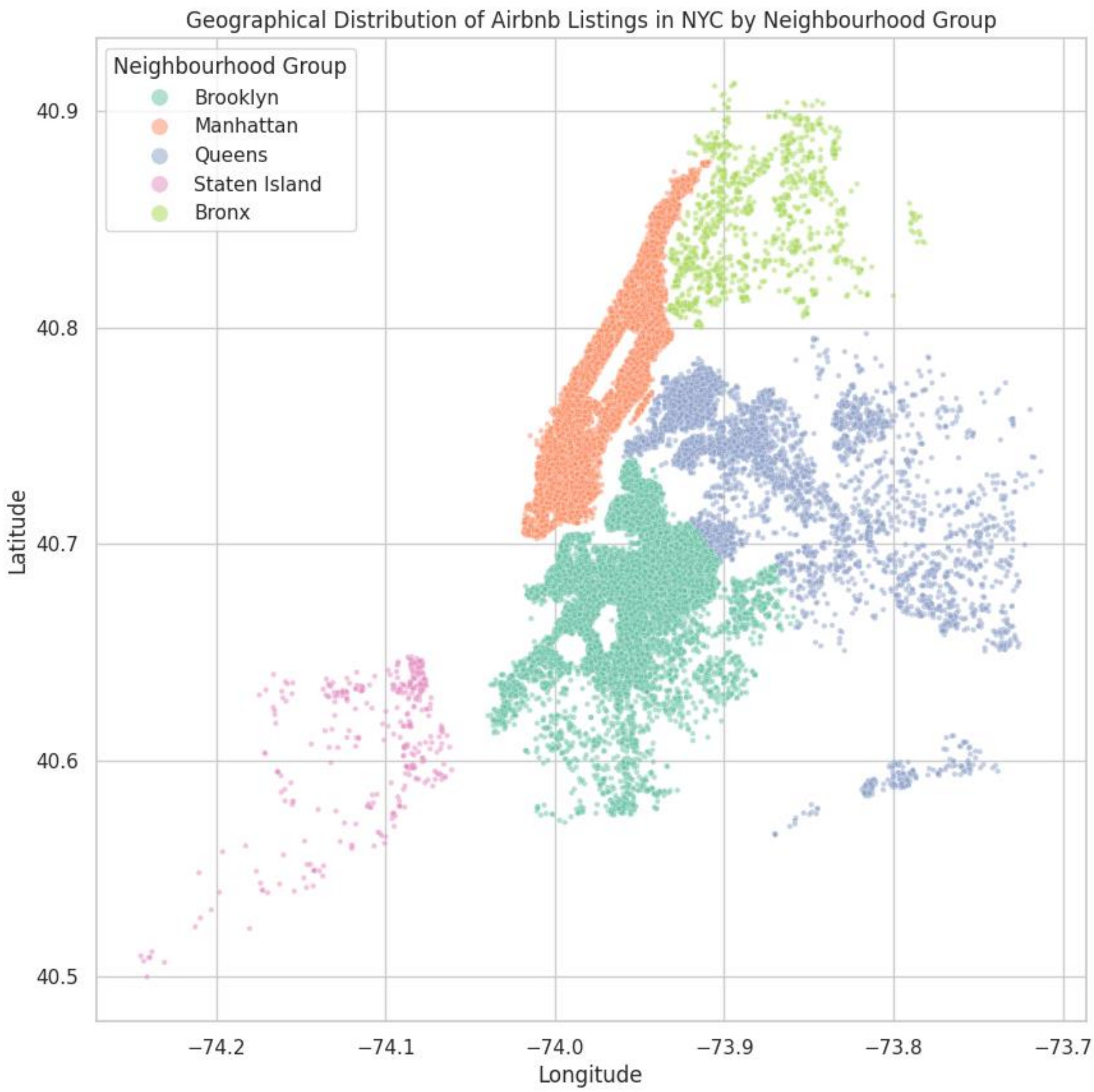
Basic geospatial analysis by plotting the locations of listings based on their latitude and longitude.

The geospatial plot reveals the distribution of Airbnb listings across New York City:
Manhattan has a dense concentration of listings, especially in central and lower Manhattan. Brooklyn also has a significant number of listings, particularly in neighborhoods closer to Manhattan. Queens, the Bronx, and Staten Island have fewer listings, which are more spread out.

This map highlights the primary areas of Airbnb activity, aligning with the neighbourhoods in NYC.

In [47]:

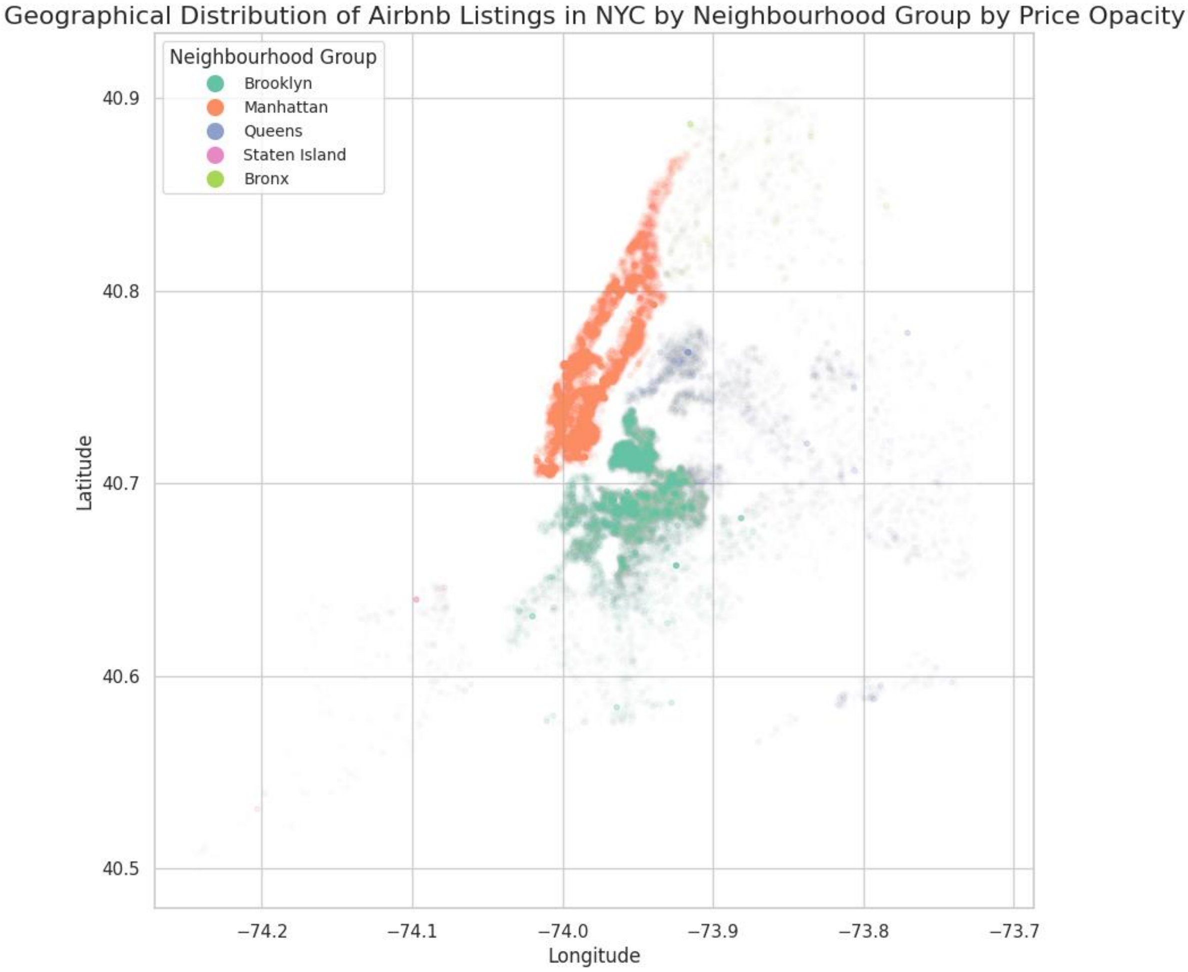
```
# Geospatial analysis - plot listings by latitude and longitude
plt.figure(figsize=(10, 10))
sns.scatterplot(x='longitude', y='latitude', hue='neighbourhood_group', data=nyc_airbnb_df,
palette="Set2", s=10, alpha=0.5)
plt.title("Geographical Distribution of Airbnb Listings in NYC by Neighbourhood Group")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend(title='Neighbourhood Group', markerscale=3, scatterpoints=1)
plt.show()
```



In [48]:

```
plt.figure(figsize=(10, 10))
# Calculate normalized prices for alpha values beforehand
normalized_prices = nyc_airbnb_df['price'] / nyc_airbnb_df['price'].max()

# Get unique neighbourhood groups and their corresponding colors from the palette
neighbourhood_groups = nyc_airbnb_df['neighbourhood_group'].unique()
palette = sns.color_palette("Set2", n_colors=len(neighbourhood_groups))
group_colors = dict(zip(neighbourhood_groups, palette))

# Plot each neighbourhood group separately with varying transparency
for group in neighbourhood_groups:
    group_data = nyc_airbnb_df[nyc_airbnb_df['neighbourhood_group'] == group]
    plt.scatter(
        x=group_data['longitude'],
        y=group_data['latitude'],
        c=[group_colors[group]],  # Use a single color for each group
        s=10,
        alpha=normalized_prices[group_data.index].values,  # Use normalized prices for alpha
        label=group
    )

# Create custom legend handles with colors
legend_handles = [
    mlines.Line2D([], [], color=color, marker='o', linestyle='None', markersize=10,
label=group)
```

```
    for group, color in group_colors.items()
]

plt.title("Geographical Distribution of Airbnb Listings in NYC by Neighbourhood Group by Price
Opacity", fontsize=16)
plt.xlabel("Longitude", fontsize=12)
plt.ylabel("Latitude", fontsize=12)
plt.legend(handles=legend_handles, title='Neighbourhood Group', fontsize=10)  # Use custom
handles
plt.show()
```



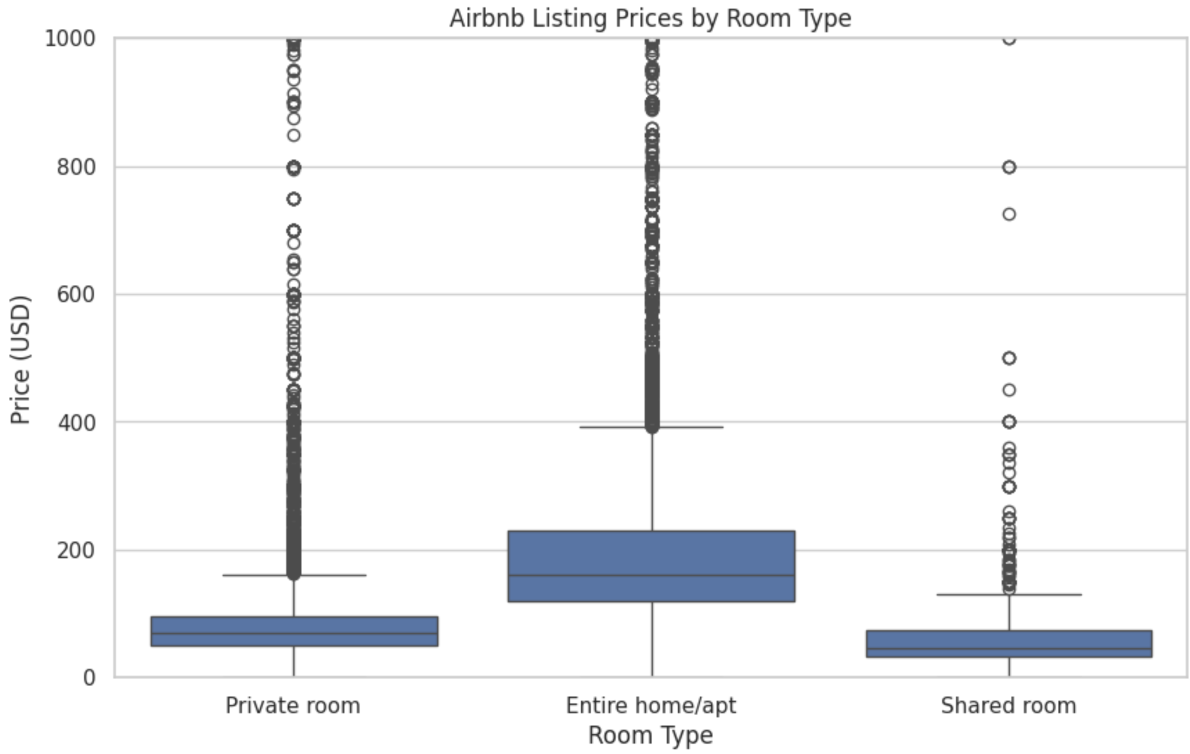
Price by Room Type (STEP 3)
Next, analyze the price distribution by room type to assess if certain types of accommodations are priced higher than others
The boxplot reveals that:

- Entire home/apartment listings have the highest median prices, with a wide range and more high-priced outliers.
- Private rooms are generally priced lower, with less variability than entire homes.
- Shared rooms have the lowest median prices and minimal variability.

This suggests that room type impacts listing prices, with entire homes commanding the highest rates.
The scatter plot suggests that hosts with a larger number of listings generally set lower prices, with most high-priced listings concentrated among hosts with fewer listings. This could indicate that individual or small-scale hosts may price their properties higher, while larger hosts with multiple listings might offer competitive pricing.
Price by Room Type (Zoomed In)
Midline of box shows median (not mean) It is more robust to outliers, which makes it a better measure of central tendency than mean, which can be pulled higher.
In [49]:

```
# Boxplot to show price distribution by room type
plt.figure(figsize=(10, 6))
sns.boxplot(x='room_type', y='price', data=nyc_airbnb_df)
plt.ylim(0, 1000)   # Limit y-axis for a clearer view
plt.title("Airbnb Listing Prices by Room Type")
```

```
plt.xlabel("Room Type")
plt.ylabel("Price (USD)")
plt.show()
```



Neighbourhood Group by Room Type by Price (STEP 5)

Finally, examine the interaction between neighbourhood group and room type to see if combining these factors provides additional insights into pricing

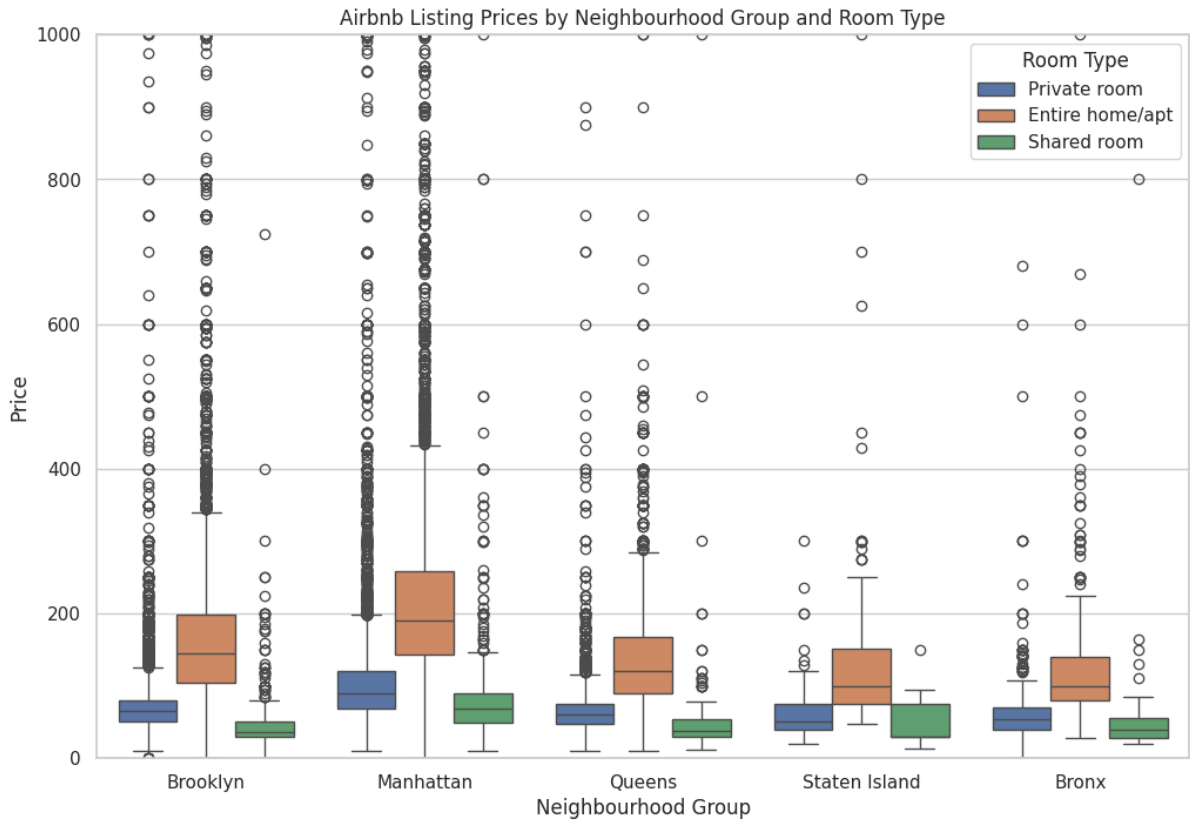
The boxplot highlights the combined influence of neighbourhood group and room type on pricing:

- Entire homes/apartments in Manhattan are priced significantly higher compared to other room types and neighborhoods, followed by Brooklyn.
- Private rooms show moderate pricing across neighborhoods, with Manhattan and Brooklyn remaining on the higher end.
- Shared rooms are generally the most affordable across all neighborhoods, with prices varying less between locations.

This interaction analysis reinforces that both neighborhood and room type play substantial roles in determining listing prices.

In [50]:

```
# Step 5: Price by Neighbourhood Group and Room Type Interaction

# Boxplot to show price distribution by neighbourhood group and room type
plt.figure(figsize=(12, 8))
sns.boxplot(x='neighbourhood_group', y='price', hue='room_type', data=nyc_airbnb_df)
plt.ylim(0, 1000)  # Limit y-axis for clearer view
plt.title("Airbnb Listing Prices by Neighbourhood Group and Room Type")
plt.xlabel("Neighbourhood Group")
plt.ylabel("Price")
plt.legend(title="Room Type")
plt.show()
```



MULTIVARIATE ANALYSIS (MACHINE LEARNING)

Oluleye (2023) describes multivariate using multiple variable techniques:

- k-means cluster analysis
- Principle Component Analysis (PCA)
- Factor analysis

Here, we only provide k-means cluster analysis.

Kmeans Cluster Analysis

Encode categorical features

In [51]:

```
# Encode and create mapping tables for categorical features
for col in ['room_type', 'neighbourhood_group']:
    encoder = LabelEncoder()
    nyc_airbnb_df[col + '_encoded'] = encoder.fit_transform(nyc_airbnb_df[col])
    mapping = dict(zip(encoder.classes_, encoder.transform(encoder.classes_)))
    print(f"{col.title()} Mapping:", mapping)

Room_Type Mapping: {'Entire home/apt': 0, 'Private room': 1, 'Shared room': 2}
Neighbourhood_Group Mapping: {'Bronx': 0, 'Brooklyn': 1, 'Manhattan': 2, 'Queens': 3, 'Staten
Island': 4}
```

Feature Selection

In [52]:

```
# Define selected features for analysis
# Choose from: 'latitude', 'longitude', 'price', 'minimum_nights', 'availability_365',
'number_of_reviews', 'reviews_per_month', 'room_type_encoded', 'neighbourhood_group_encoded'
# features = ['price', 'minimum_nights', 'number_of_reviews', 'availability_365']
features = ['price', 'minimum_nights', 'availability_365', 'number_of_reviews',
            'reviews_per_month', 'calculated_host_listings_count', 'room_type_encoded',
'neighbourhood_group_encoded']

# Drop rows with NaN values in the selected features
data_cleaned = nyc_airbnb_df.dropna(subset=features).copy()
```

Normality Test

- Can not use Shapiro-Wilk here because it is designed for data sets less than 5000.
- Use Anderson-Darling because large dataset with outliers.

Anderson-Darling shows:

- Choose 5.0% significance level 0.787

- Null hypothesis: If Anderson-Darling statistic is > critical value at 5%, reject null hypothesis of normality.
- Alternative: If Anderson-Darling is <= critical value at 5%, fail to reject null.
- all features are > 0.787 ==> Reject null, not normal.

As seen throughout, they are not normal.
In [53]:

```
# Normality test on the selected features
normality_results = {}
for col in features:
    # Drop NaN values before the test
    data_col = data_cleaned[col].dropna()

    # Perform Anderson-Darling test
    statistic, critical_values, significance_level = stats.anderson(data_col, dist='norm')

    # Store results in the dictionary
    normality_results[col] = {
        'Anderson-Darling Statistic': statistic,
        'Critical Values': critical_values,
        'Significance Level': significance_level
    }

# Convert the results to a DataFrame for better readability
normality_results_df = pd.DataFrame(normality_results).T
normality_results_df.reset_index(inplace=True)
normality_results_df.rename(columns={'index': 'Feature'}, inplace=True)

# Display the normality results in a clean table format
print("Normality Test Results (Anderson-Darling):")
print(normality_results_df)
```

```
Normality Test Results (Anderson-Darling):
                          Feature Anderson-Darling Statistic                      Critical
Values  \
0                           price                7277.445657  [0.576, 0.656, 0.787, 0.918,
1.092]
1                  minimum_nights               10324.332785  [0.576, 0.656, 0.787, 0.918,
1.092]
2                availability_365                4044.550982  [0.576, 0.656, 0.787, 0.918,
1.092]
3               number_of_reviews                7077.730483  [0.576, 0.656, 0.787, 0.918,
1.092]
4               reviews_per_month                4416.952658  [0.576, 0.656, 0.787, 0.918,
1.092]
5  calculated_host_listings_count               15451.861681  [0.576, 0.656, 0.787, 0.918,
1.092]
6               room_type_encoded                7335.071753  [0.576, 0.656, 0.787, 0.918,
1.092]
7     neighbourhood_group_encoded                3937.089193  [0.576, 0.656, 0.787, 0.918,
1.092]

            Significance Level
0  [15.0, 10.0, 5.0, 2.5, 1.0]
1  [15.0, 10.0, 5.0, 2.5, 1.0]
2  [15.0, 10.0, 5.0, 2.5, 1.0]
3  [15.0, 10.0, 5.0, 2.5, 1.0]
4  [15.0, 10.0, 5.0, 2.5, 1.0]
5  [15.0, 10.0, 5.0, 2.5, 1.0]
6  [15.0, 10.0, 5.0, 2.5, 1.0]
7  [15.0, 10.0, 5.0, 2.5, 1.0]
```

Remove Outliers

- k-means is sensitive to outliers.
- Remove outliers using Interquartile Range (IQR) method.

In [54]:

```
# Remove outliers using IQR and standardize the data
def remove_outliers_iqr(dataframe, columns):
    cleaned_data = dataframe.copy()
    for col in columns:
        Q1 = cleaned_data[col].quantile(0.25)
        Q3 = cleaned_data[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
```

```
        cleaned_data = cleaned_data[(cleaned_data[col] >= lower_bound) & (cleaned_data[col] <=
upper_bound)]
    return cleaned_data

# Applying the outlier removal
data_no_outliers = remove_outliers_iqr(data_cleaned, features)
data_no_outliers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 29408 entries, 1 to 48894
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              29408 non-null  int64
 1   host_id                         29408 non-null  int64
 2   neighbourhood_group             29408 non-null  object
 3   neighbourhood                   29408 non-null  object
 4   latitude                        29408 non-null  float64
 5   longitude                       29408 non-null  float64
 6   room_type                       29408 non-null  object
 7   price                           29408 non-null  int64
 8   minimum_nights                  29408 non-null  int64
 9   number_of_reviews               29408 non-null  int64
 10  last_review                     29408 non-null  object
 11  has_review                      29408 non-null  int64
 12  reviews_per_month               29408 non-null  float64
 13  calculated_host_listings_count  29408 non-null  int64
 14  availability_365                29408 non-null  int64
 15  room_type_encoded               29408 non-null  int64
 16  neighbourhood_group_encoded     29408 non-null  int64
dtypes: float64(3), int64(10), object(4)
memory usage: 4.0+ MB
In [55]:
```

```
# Display the data without outliers
data_no_outliers.reset_index(drop=True, inplace=True)  # Resetting index for clarity
data_no_outliers.head()  # Displaying the first few rows of the dataset without outliers
```

Out[55]:

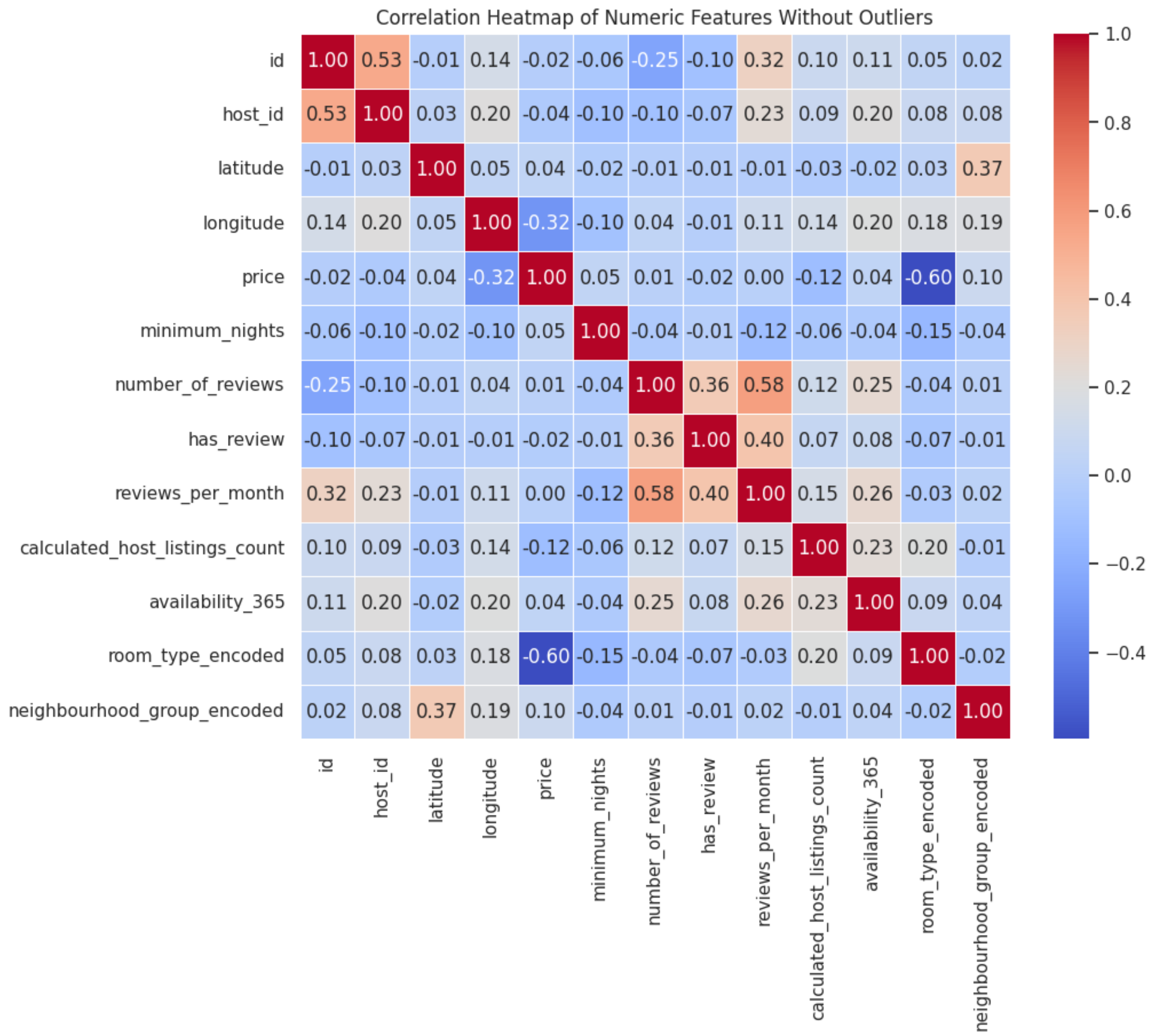
	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	has_review	reviews_per_month	calculated_host_listings_count	availability_365	room_type_encoded	neighbourhood_group_encoded
0	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	1	0.38	2	355	0	2
1	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	1677-09-21 00:12:43.145224193	0	0.00	1	365	1	2

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	has_review	reviews_per_month	calculated_host_listings_count	availability_365	room_type_encoded	neighbourhood_group_encoded
2	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	1	0.10	1	0	0	2
3	5295	7702	Manhattan	Upper West Side	40.80316	-73.96545	Entire home/apt	135	5	53	2019-06-22	1	0.43	1	6	0	2
4	7750	17985	Manhattan	East Harlem	40.79685	-73.94872	Entire home/apt	190	7	0	1677-09-21 00:12:43.145224193	0	0.00	2	249	0	2

Linear Correlation Map without outliers

In [57]:

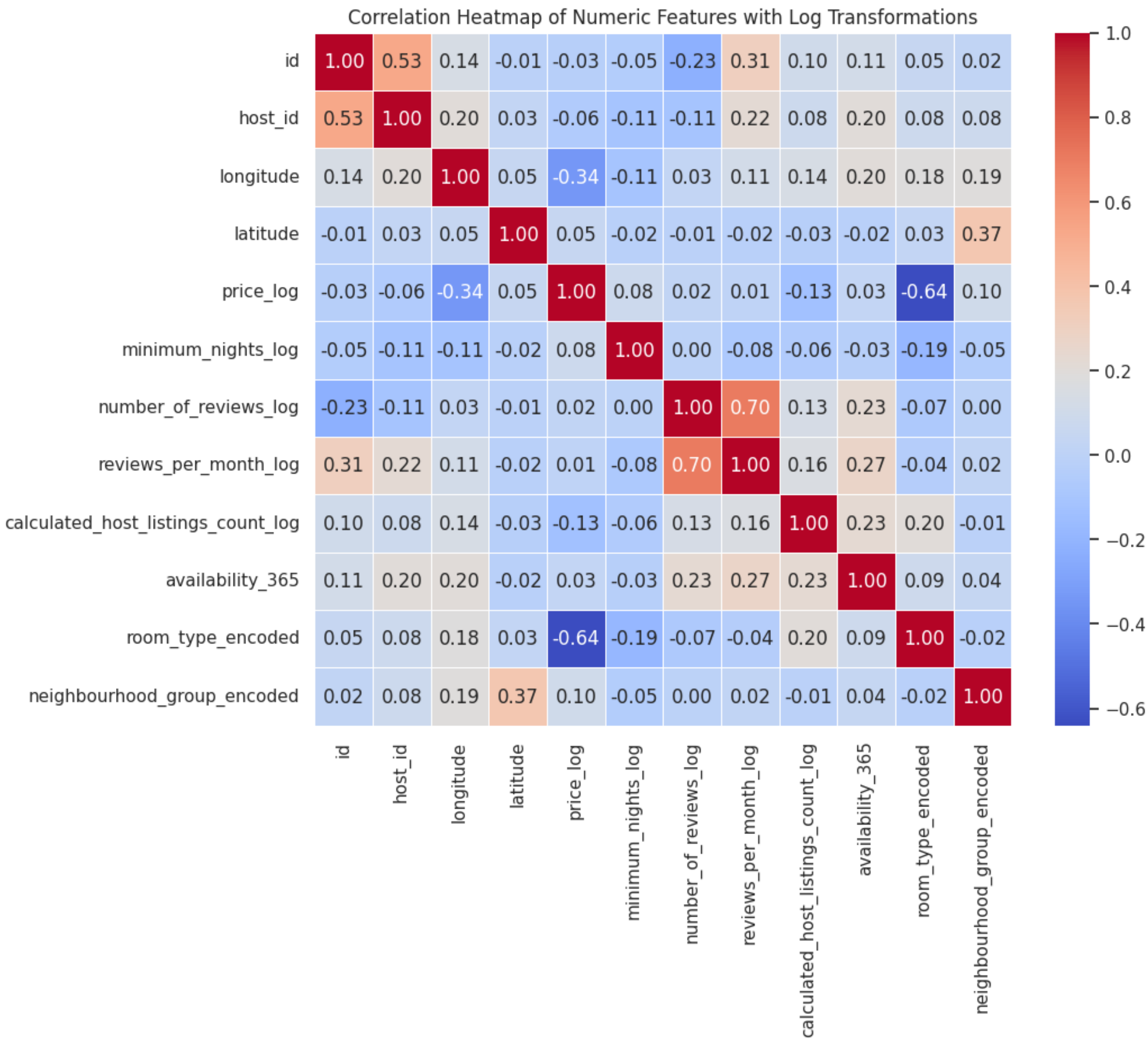
```
# Get new numeric list with encoded categoricals
numeric_columns_nyc = data_no_outliers.select_dtypes(include=['float64', 'int64']).columns

# Outlier free correlation matrix
correlation_matrix_no_outliers = data_no_outliers[numeric_columns_nyc].corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_no_outliers, annot=True, cmap="coolwarm", linewidths=0.5,
fmt=".2f")

plt.title("Correlation Heatmap of Numeric Features Without Outliers")
plt.show()
```



Log Correlation Map without outliers

In [58]:

```
# Create a copy of the DataFrame to avoid modifying the original
data_log_transformed = data_no_outliers.copy()

# Apply log transformations to selected columns
for col in ['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month',
'calculated_host_listings_count']:
    data_log_transformed[col + '_log'] = np.log1p(data_log_transformed[col])

# Create a list of log-transformed features
log_transformed_features = ['price_log', 'minimum_nights_log', 'number_of_reviews_log',
                            'reviews_per_month_log', 'calculated_host_listings_count_log']

# Select columns to include and remove NaN values
data_for_correlation = data_log_transformed[['id', 'host_id', 'longitude', 'latitude'] +
log_transformed_features +
['availability_365','room_type_encoded','neighbourhood_group_encoded']].dropna()

# ... Update numeric_columns_nyc to include 'price_log' and others
numeric_columns_nyc = data_for_correlation.select_dtypes(include=['float64', 'int64']).columns

# Calculate correlation matrix
correlation_matrix_log_transformed = data_for_correlation[numeric_columns_nyc].corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))  # Increased figure size to accommodate labels
sns.heatmap(correlation_matrix_log_transformed.round(2), annot=True, cmap="coolwarm",
linewidths=0.5, fmt=".2f")
plt.title("Correlation Heatmap of Numeric Features with Log Transformations")
plt.show()
```



Standardise Data

In [59]:

```
# Standardizing the data
scaler = StandardScaler()
scaled_data_no_outliers = scaler.fit_transform(data_no_outliers[features])

print("Outlier removal and standardization are complete.")
```

```
Outlier removal and standardization are complete.
```

Display standardised data without outliers

In [60]:

```
# Display the standardised data without outliers
scaled_data_no_outliers_df = pd.DataFrame(scaled_data_no_outliers, columns=features)
scaled_data_no_outliers_df.head()
```

Out[60]:

	price	minimum_nights	availability_365	number_of_reviews	reviews_per_month	calculated_host_listings_count	room_type_encoded	neighbourhood_group_encoded
0	1.542496	-0.972669	2.506412	2.260172	-0.271607	1.329224	-0.975070	0.540583
1	0.427050	0.045149	2.594634	-0.691617	-0.773451	-0.470823	0.887417	0.540583

	price	minimum_nights	availability_365	number_of_reviews	reviews_per_month	calculated_host_listings_count	room_type_encoded	neighbourhood_group_encoded
2	-0.614034	3.607511	-0.625488	-0.101259	-0.641387	-0.470823	-0.975070	0.540583
3	0.203960	1.062967	-0.572555	2.784934	-0.205575	-0.470823	-0.975070	0.540583
4	1.021954	2.080785	1.571253	-0.691617	-0.773451	1.329224	-0.975070	0.540583

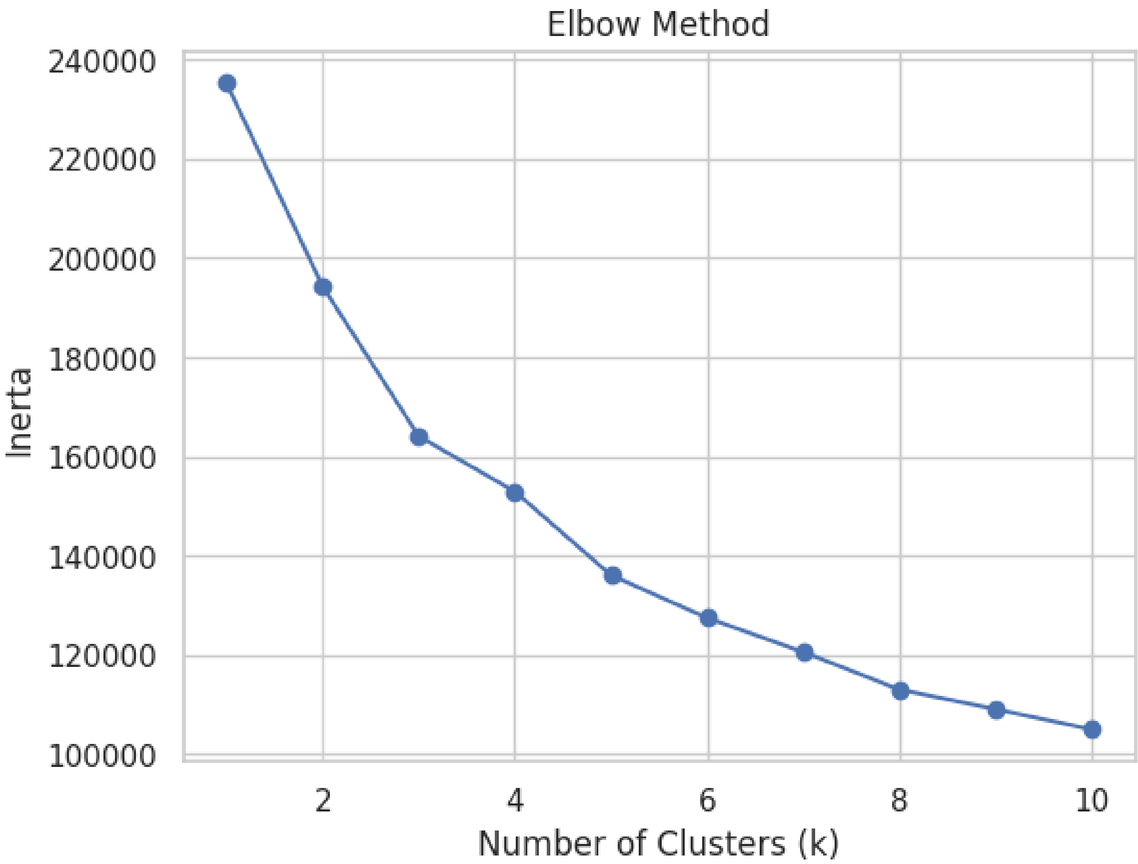
Select k

Elbow Method

- Use Elbow to select optimal clusters (k)
- Look for where decrease in inertia starts

In [61]:

```
wcss = []
for i in range(1, 11):  # Try k values from 1 to 10
  kmeans = KMeans(n_clusters=i, random_state=42)  # Initialize KMeans
  kmeans.fit(scaled_data_no_outliers)  # Fit to your data
  wcss.append(kmeans.inertia_)  # Append Within Cluster Sum of Squares (WCSS) aka inertia

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



Conduct k-means

- Selected k as per elbow for feature selection.

K = 3

Choose K

In [62]:

```
# Choose k from Elbow graph where it bends up.
k = 3

# Perform KMeans clustering
kmeans_no_outliers = KMeans(n_clusters=k, random_state=42)
clusters_no_outliers = kmeans_no_outliers.fit_predict(scaled_data_no_outliers)

# Add cluster labels to the cleaned dataset
data_no_outliers['Cluster'] = clusters_no_outliers

# Display the dataset with the assigned clusters
data_no_outliers.head()
```

Out[62]:

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	has_review	reviews_per_month	calculated_host_listings_count	availability_365	room_type_encoded	neighbourhood_group_encoded	Cluster
0	2595	2845	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	2019-05-21	1	0.38	2	355	0	2	0
1	3647	4632	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	1677-09-21 00:12:43.145224193	0	0.00	1	365	1	2	2
2	5022	7192	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9	2018-11-19	1	0.10	1	0	0	2	1

	id	host_id	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	has_review	reviews_per_month	calculated_host_listings_count	availability_365	room_type_encoded	neighbourhood_group_encoded	Cluster
3	5295	7702	Manhattan	Upper West Side	40.80316	-73.96545	Entire home/apt	135	5	53	2019-06-22	1	0.43	1	6	0	2	0
4	7750	17985	Manhattan	East Harlem	40.79685	-73.94872	Entire home/apt	190	7	0	1677-09-21 00:12:43.145224193	0	0.00	2	249	0	2	1

k-means Insights

Cluster sizes

In [63]:

```
# Cluster labels
cluster_labels = data_no_outliers['Cluster']

# Get unique cluster labels and their counts
unique_clusters, cluster_counts = np.unique(cluster_labels, return_counts=True)

# Print the cluster sizes
for cluster, count in zip(unique_clusters, cluster_counts):
    print(f"Cluster {cluster}: {count} listings")
```

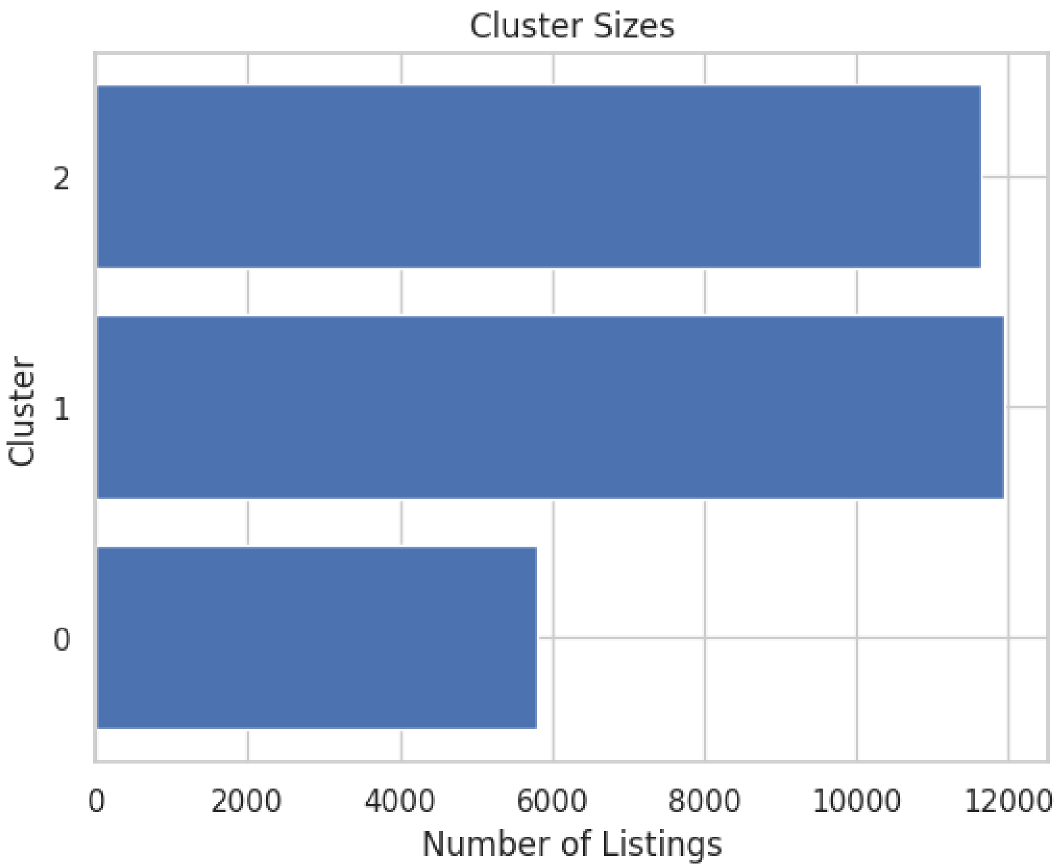
```
Cluster 0: 5807 listings
Cluster 1: 11943 listings
Cluster 2: 11658 listings
```

Visualise Cluster Sizes

In [64]:

```
# Get unique cluster labels and their counts
unique_clusters, cluster_counts = np.unique(cluster_labels, return_counts=True)

# Create horizontal bar chart
plt.barh(unique_clusters, cluster_counts)

# Add labels and title
plt.xlabel('Number of Listings')
plt.ylabel('Cluster')
plt.title('Cluster Sizes')

# Set y-axis ticks to integers only
plt.yticks(unique_clusters)

# Display the chart
```

```
plt.show()
```



Calculate means to get insight into meaning of clusters.

Cluster 0: Popular, moderately priced and desirable

- Moderate price ($117)
- Short minimum nights (2.5)
- High reviews (32) and per month (1.7)
- More than one listing (1.5)
- Available half of year (156)
- Mix room types but also entire homes
- Popular neighbourhood

Cluster 1: Desirable but expensive with limited availability

- Highest priced ($166)
- Slightly longer minimum nights (3.3)
- Few reviews (6) and per month (.33)
- Just over one listing (1.1)
- Low availability (40)
- Mostly entire homes
- Popular neighbourhood

Cluster 2: Cheapest but less desirable with limited availability

- Cheapest ($77)
- Short minimum nights (2.7)
- Fewest reviews (4) and per month (.27)
- Moderate listings (1.3)
- Moderately low availability (60)
- Mostly private or shared rooms
- Less popular or central neighbourhoods

In [65]:

```
cluster_means = data_no_outliers.groupby('Cluster').mean(numeric_only=True)
cluster_means
```

Out[65]:

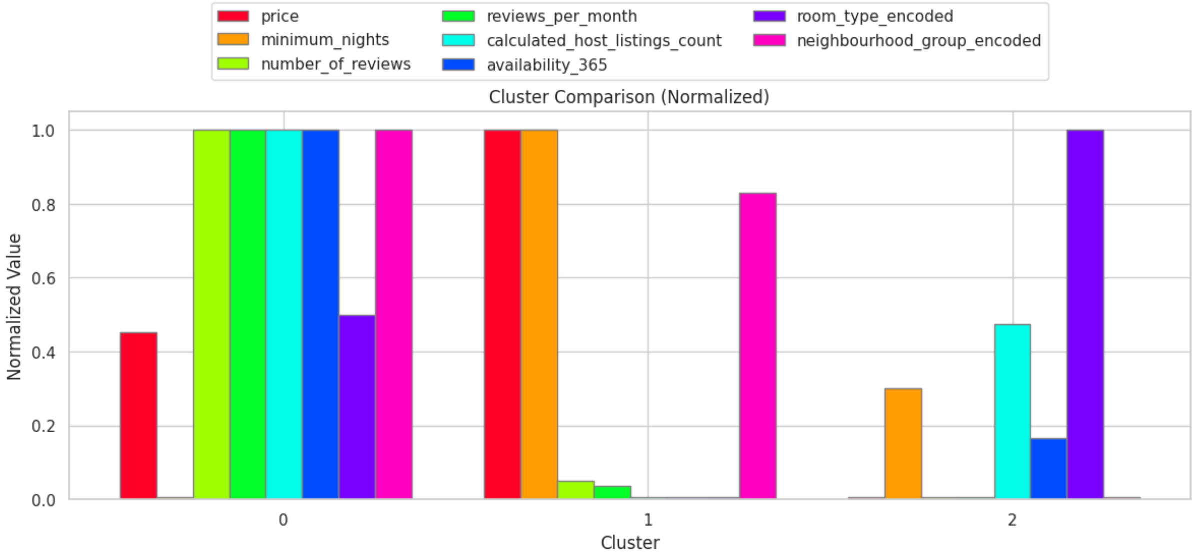
	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	has_review	reviews_per_month	calculated_host_listings_count	availability_365	room_type_encoded	neighbourhood_group_encoded
Cluster													
0	1.929124e+07	7.334394e+07	40.728015	-73.940484	117.370415	2.476494	32.828138	1.000000	1.734293	1.501808	156.214052	0.528672	1.661271
1	1.778771e+07	4.874774e+07	40.729367	-73.962235	166.340367	3.304697	5.785983	0.767395	0.330198	1.110525	40.474002	0.018002	1.645232
2	1.896497e+07	6.172734e+07	40.729437	-73.944573	77.081060	2.724824	4.317550	0.704066	0.275226	1.296620	59.571110	1.038857	1.567765

Normalise means and visualise clusters

In [66]:

```
# features to visualise
features_to_visualize = ['price', 'minimum_nights', 'number_of_reviews', 'reviews_per_month',
                         'calculated_host_listings_count', 'availability_365',
'room_type_encoded', 'neighbourhood_group_encoded']

# Normalize the data using MinMaxScaler and store in 'normalised_cluster_means'
scaler = MinMaxScaler()
normalised_data = scaler.fit_transform(cluster_means[features_to_visualize])
normalised_cluster_means = pd.DataFrame(normalised_data, columns=features_to_visualize,
index=cluster_means.index)

# Add the 'Cluster' column back to the normalized data
normalised_cluster_means['Cluster'] = cluster_means.index

# Explicitly cast all columns to float
normalised_cluster_means = normalised_cluster_means.astype(float)

# Add a small offset to zero values
normalised_cluster_means[normalised_cluster_means == 0] = 0.005  # Adjust the offset value as
needed

# Set the width of the bars
bar_width = 0.1

# Set the positions of the bars on the x-axis (adjusted for more features)
r = np.arange(len(cluster_means.index))
positions = [r + i * bar_width for i in range(len(features_to_visualize))]

# Create the bar chart with adjusted figure size
plt.figure(figsize=(12, 6))  # Increased figure width for better visibility

# Color map for the bars
num_features = len(features_to_visualize)
cmap = plt.get_cmap('gist_rainbow', num_features)
colors = [cmap(i) for i in range(num_features)]
```

```
# Create bars for each feature
for i, feature in enumerate(features_to_visualize):
    plt.bar(positions[i], normalised_cluster_means[feature], color=colors[i], width=bar_width,
edgecolor='grey', label=feature)

# Add labels, title, and legend
plt.xlabel('Cluster')
plt.ylabel('Normalized Value')
plt.title('Cluster Comparison (Normalized)')
plt.xticks([r + bar_width * (len(features_to_visualize) / 2) for r in
range(len(cluster_means.index))], cluster_means.index)  # Center x-axis ticks
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.3), ncol=3)  # Adjust legend position

plt.tight_layout()
plt.show()
```

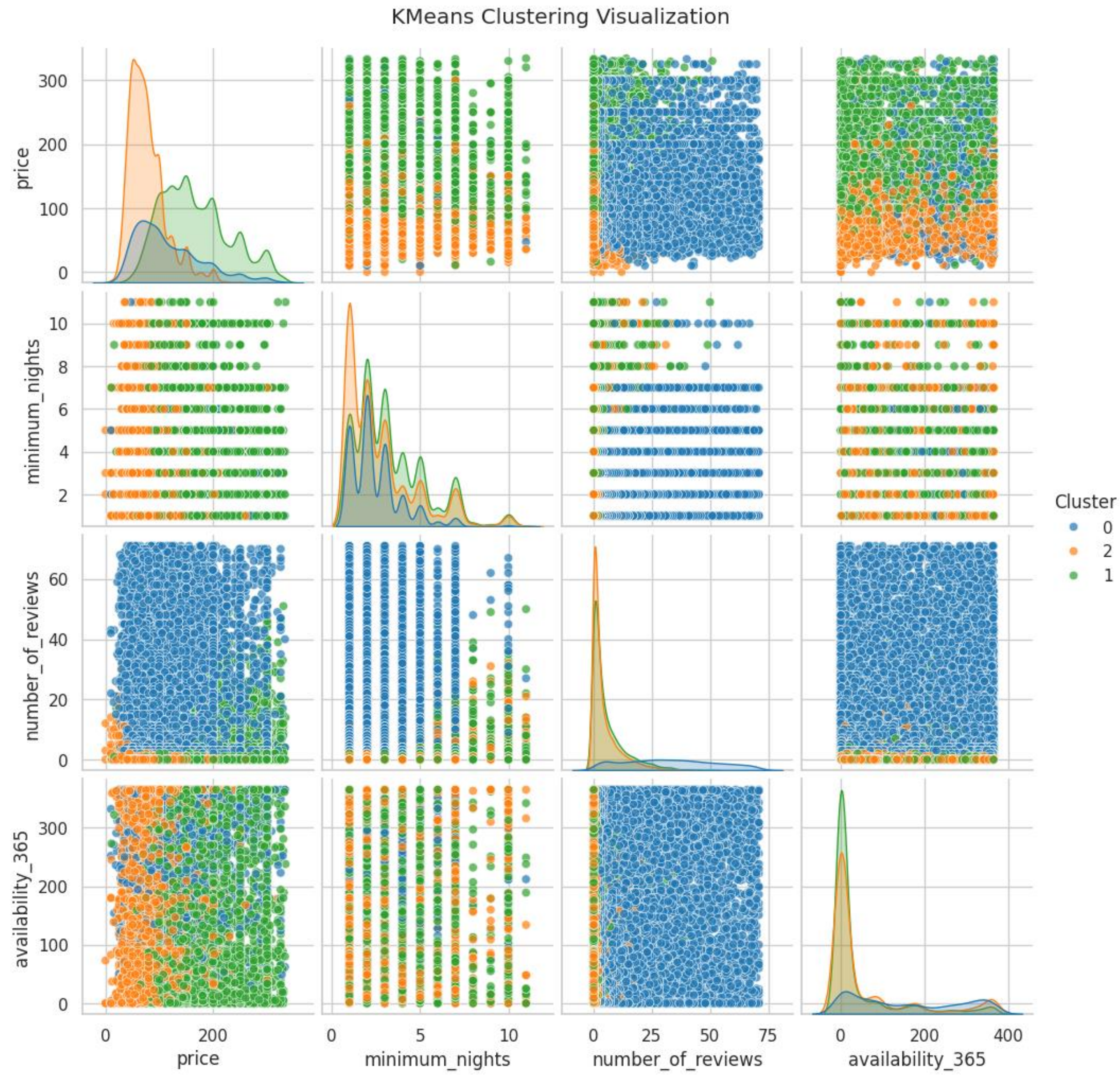


Visualise k-means using pairplot

- This takes a bit of time to run
- Creates pair-plots for the four features and five clusters

In [67]:

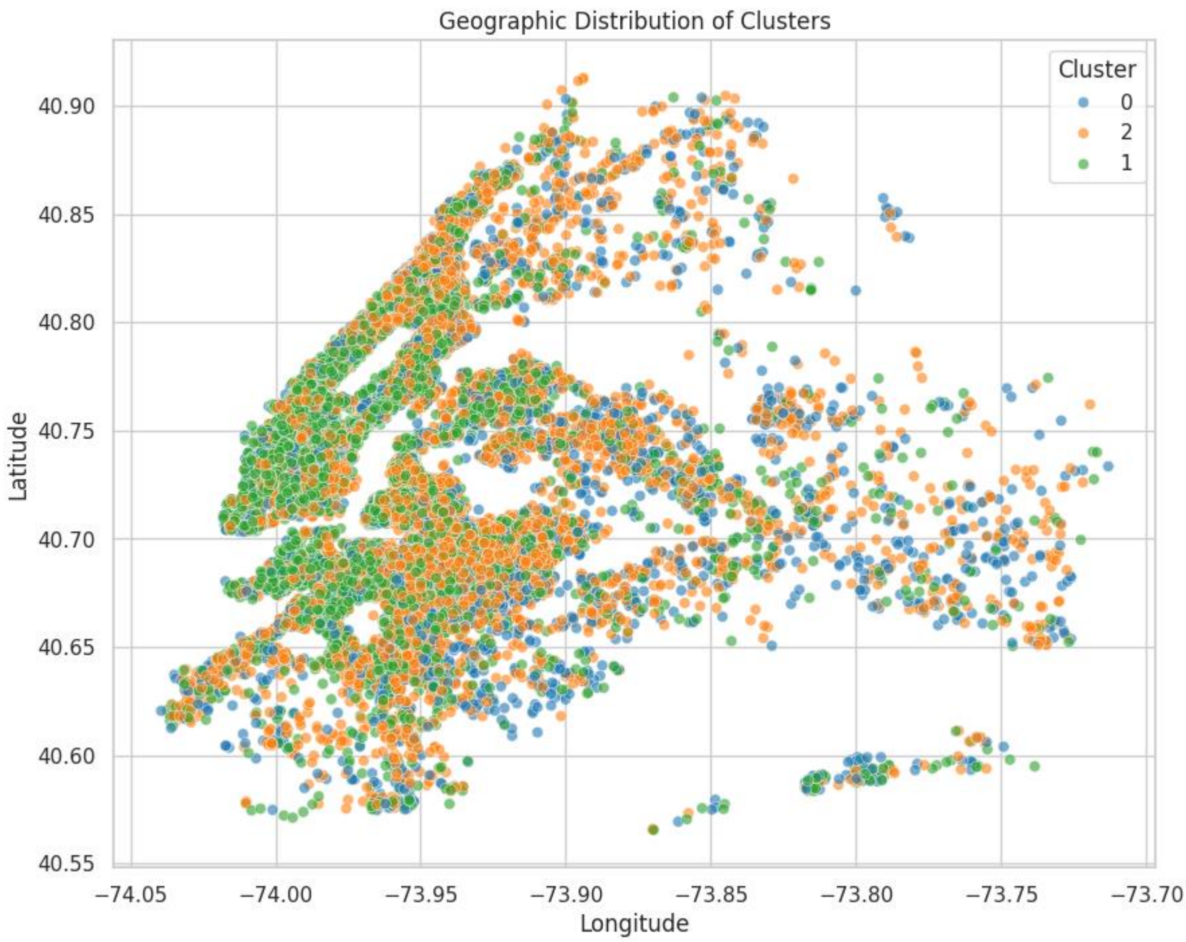
```
# Add cluster labels for visualization
data_no_outliers['Cluster'] = data_no_outliers['Cluster'].astype(str)

# Pairplot to visualize clusters based on selected features
sns.set(style="whitegrid")
pairplot = sns.pairplot(
    data_no_outliers,
    vars=['price', 'minimum_nights', 'number_of_reviews', 'availability_365'],
    hue='Cluster',
    palette='tab10',
    diag_kind='kde',
    plot_kws={'alpha': 0.7}
)

# Show the pairplot
plt.suptitle("KMeans Clustering Visualization", y=1.02)
plt.show()
```



Geographic distribution of clusters

In [68]:

```
# Geographic distribution of clusters
plt.figure(figsize=(10, 8))
sns.scatterplot(
    x=data_no_outliers['longitude'],
    y=data_no_outliers['latitude'],
    hue=data_no_outliers['Cluster'],
    palette='tab10',
    alpha=0.6
)
plt.title("Geographic Distribution of Clusters")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend(title="Cluster")
plt.show()
```



Geographic Visualisation

- Takes a bit of time to visualise.
- WARNING! WARNING! WARNING! If this has been run, can't check in to GitHub.

In [ ]:

```
# Create a GeoDataFrame for geographic visualization
geometry = [Point(xy) for xy in zip(data_no_outliers['longitude'],
data_no_outliers['latitude'])]
geo_df = gpd.GeoDataFrame(data_no_outliers, geometry=geometry)

# Base map creation using Folium
map_clusters = folium.Map(location=[data_no_outliers['latitude'].mean(),
data_no_outliers['longitude'].mean()], zoom_start=11)

# Add cluster points to the map
colors = ['blue', 'green', 'red']
for _, row in geo_df.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=5,
        color=colors[int(row['Cluster'])],
        fill=True,
        fill_opacity=0.6,
        popup=f"Cluster: {row['Cluster']}\nPrice: {row['price']}\nReviews:
{row['number_of_reviews']}"
    ).add_to(map_clusters)

# Display the map inline
map_clusters
```

REFERENCES

Oluleye, A. (2023) *Exploratory Data Analysis with Python Cookbook*. Pockt Publishing.