Project Name - Airbnb Bookings Analysis

Project Type - Exploratory Data Analysis (EDA)

Contribution - Individual

Name - Ravikant Shrivas

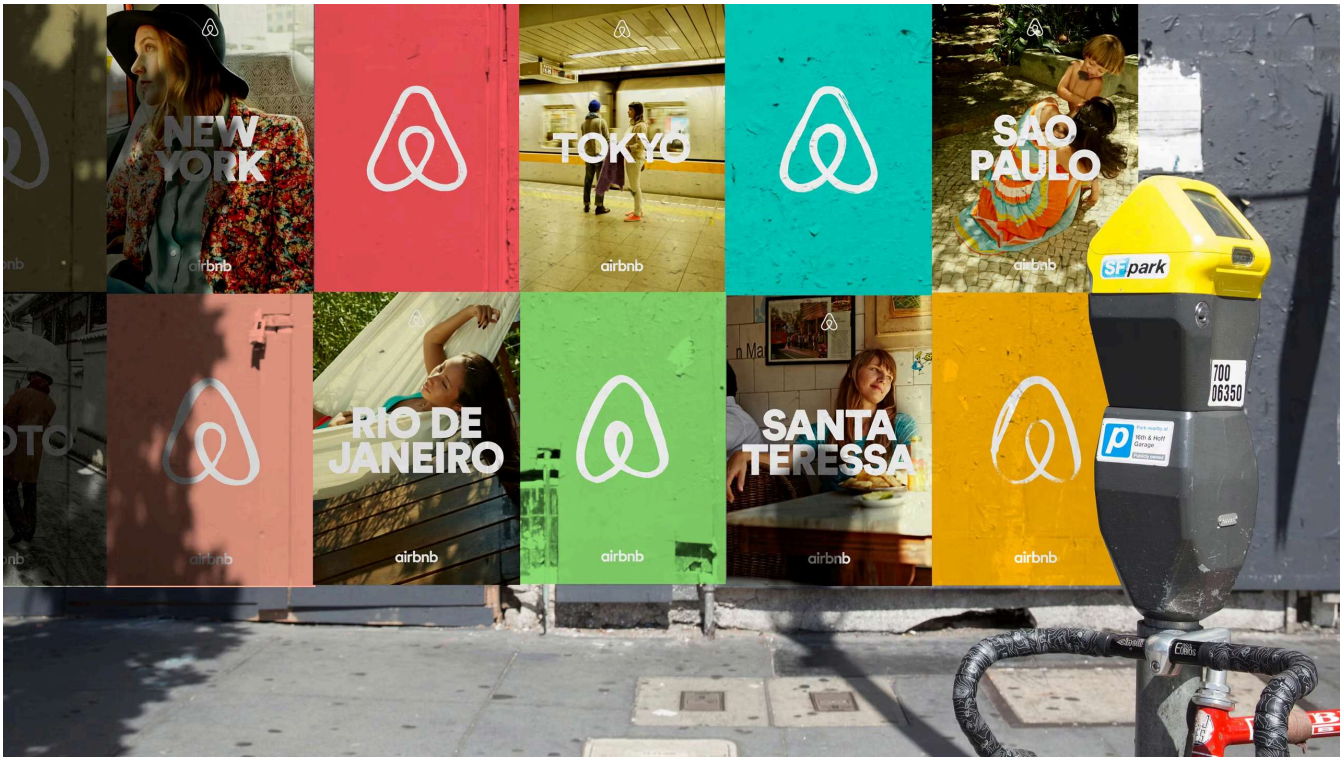
Project Summary -

- **The purpose of the analysis:** understanding the factors that influence Airbnb prices in New York City, or identifying patterns of all variables and Our analysis provides useful information for travelers and hosts in the city and also provides some best insights for Airbnb business.
- This project involved exploring and cleaning a dataset to prepare it for analysis. The data exploration process involved identifying and understanding the characteristics of the data, such as the data types, missing values, and distributions of values. The data cleaning process involved identifying and addressing any issues or inconsistencies in the data, such as errors, missing values, or duplicate records and remove outliers.
- Through this process, we were able to identify and fix any issues with the data, and ensure that it was ready for further analysis. This is an important step in any data analysis project, as it allows us to work with high-quality data and avoid any potential biases or errors that could affect the results. The clean and prepared data can now be used to answer specific research.
- Once the data has been cleaned and prepared, now begin exploring and summarizing it with describe the data and creating visualizations, and identifying patterns and trends in the data. in explore the data, may develop the relationships between different variables or the underlying causes of certain patterns or trends and other methods.
- using data visualization to explore and understand patterns in Airbnb data. We created various graphs and charts to visualize the data, and wrote observations and insights below each one to help us better understand the data and identify useful insights and patterns.
- Through this process, we were able to uncover trends and relationships in the data that would have been difficult to identify through raw data alone, for example factors affecting prices and availability. We found that minimum nights, number of reviews, and host listing count are important for determining prices, and that availability varies significantly across neighborhoods. Our analysis provides useful information for travelers and hosts in the city.
- The observations and insights we identified through this process will be useful for future analysis and decision-making related to Airbnb. and also Our analysis provides useful information for travelers and hosts in the city.

Problem Statements -

1. What are the most popular neighborhoods for Airbnb rentals in New York City? How do prices and availability vary by neighborhood?
2. How has the Airbnb market in New York City changed over time? Have there been any significant trends in terms of the number of listings, prices, or occupancy rates?
3. Are there any patterns or trends in terms of the types of properties that are being rented out on Airbnb in New York City? Are certain types of properties more popular or more expensive than others?
4. Are there any factors that seem to be correlated with the prices of Airbnb rentals in New York City?
5. the best area in New York City for a host to buy property at a good price rate and in an area with high traffic ?
6. How do the lengths of stay for Airbnb rentals in New York City vary by neighborhood? Do certain neighborhoods tend to attract longer or shorter stays?
7. How do the ratings of Airbnb rentals in New York City compare to their prices? Are higher-priced rentals more likely to have higher ratings?
8. Find the total numbers of Reviews and Maximum Reviews by Each Neighborhood Group.
9. Find Most reviewed room type in Neighborhood groups per month.

there is a lot of problem statements and we have to finds information and insights through different different problem statements so now lets start...



Importing Data From Kaggle

```
from google.colab import files
files.upload()
```

```
Choose Files kaggle.json
  • kaggle.json(application/json) - 68 bytes, last modified: 2/27/2025 - 100% done
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"ravikant1500","key":"62b8fc86c6c52fb4feb64889325f3e17"}'}
```

```
import os
os.makedirs("/root/.kaggle", exist_ok=True)
!mv kaggle.json /root/.kaggle/
!chmod 600 /root/.kaggle/kaggle.json  # Secure the file
```

```
!kaggle datasets download -d vrindakallu/new-york-dataset
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.7.4 / client 1.6.17)
Dataset URL: https://www.kaggle.com/datasets/vrindakallu/new-york-dataset
License(s): CC-BY-SA-4.0
new-york-dataset.zip: Skipping, found more recently modified local copy (use --force to force download)
```

```
import zipfile
with zipfile.ZipFile("new-york-dataset.zip", 'r') as zip_ref:
  zip_ref.extractall("dataset")
```

```
import os
os.listdir("dataset")
```

```
['new_york_listings_2024.csv']
```

```
import pandas as pd
df = pd.read_csv("/content/dataset/new_york_listings_2024.csv")
df.head(5)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	...	l
0	1312228	Rental unit in Brooklyn · ★5.0 · 1 bedroom	7130382	Walter	Brooklyn	Clinton Hill	40.683710	-73.964610	Private room	55.0	...	
1	45277537	Rental unit in New York · ★4.67 · 2 bedrooms ·...	51501835	Jeniffer	Manhattan	Hell's Kitchen	40.766610	-73.988100	Entire home/apt	144.0	...	
2	971353993633883038	Rental unit in New York · ★4.17 · 1 bedroom · ...	528871354	Joshua	Manhattan	Chelsea	40.750764	-73.994605	Entire home/apt	187.0	...	
3	3857863	Rental unit in New York · ★4.64 · 1 bedroom · ...	19902271	John And Catherine	Manhattan	Washington Heights	40.835600	-73.942500	Private room	120.0	...	
4	40896611	Condo in New York · ★4.91 · Studio · 1 bed · 1...	61391963	Stay With Vibe	Manhattan	Murray Hill	40.751120	-73.978600	Entire home/apt	85.0	...	

5 rows × 22 columns

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


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20758 entries, 0 to 20757
Data columns (total 22 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              20758 non-null  int64  
 1   name                            20758 non-null  object 
 2   host_id                         20758 non-null  int64  
 3   host_name                       20758 non-null  object 
 4   neighbourhood_group             20758 non-null  object 
 5   neighbourhood                   20758 non-null  object 
 6   latitude                        20758 non-null  float64
 7   longitude                       20758 non-null  float64
 8   room_type                       20758 non-null  object 
 9   price                           20758 non-null  float64
 10  minimum_nights                  20758 non-null  int64  
 11  number_of_reviews               20758 non-null  int64  
 12  last_review                     20758 non-null  object 
 13  reviews_per_month               20758 non-null  float64
 14  calculated_host_listings_count  20758 non-null  int64  
 15  availability_365                20758 non-null  int64  
 16  number_of_reviews_ltm           20758 non-null  int64  
 17  license                         20758 non-null  object 
 18  rating                          20758 non-null  object 
 19  bedrooms                        20758 non-null  object 
 20  beds                            20758 non-null  int64  
 21  baths                           20758 non-null  object 
dtypes: float64(4), int64(8), object(10)
memory usage: 3.5+ MB
```

Data Cleaning & Transformation

```
df.shape
```

```
(20758, 22)
```

```
df.columns.tolist()
```

```
['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',
 'number_of_reviews_ltm',
 'license',
 'rating',
 'bedrooms',
 'beds',
 'baths']
```

Cleaning The Data Missing Vlues...

```
df['bedrooms'].unique()
```

```
array(['1', '2', 'Studio', '3', '6', '5', '4', '7', '15', '9', '8', '14'],
      dtype=object)
```

```
df.loc[(df['rating'] == 'No rating') | (df['rating'] == 'New '), 'rating'] = 0
df.loc[(df['bedrooms'] == 'Studio'), 'bedrooms'] = 0
df = df.rename(columns={'neighbourhood_group': 'n_group', 'number_of_reviews':'total_reviews','calculated_host_listings_count':'host_listing
df = df[df['availability_365'] != 0]  # Excluding rows with no availability
```

```
df['rating'] = pd.to_numeric(df['rating'])
df['bedrooms'] = pd.to_numeric(df['bedrooms'])
```

```
num_records = len(df)
num_records
```

```
18276
```

```
df.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	total_reviews	reviews_per_month	host_listi
count	1.827600e+04	1.827600e+04	18276.000000	18276.000000	18276.000000	18276.000000	18276.000000	18276.000000	182
mean	3.193323e+17	1.762031e+08	40.726368	-73.938620	188.547056	28.556303	41.911195	1.233938	
std	3.944108e+17	1.745000e+08	0.060786	0.062194	1081.574573	32.442013	70.268809	1.730621	
min	2.595000e+03	2.234000e+03	40.500314	-74.249840	10.000000	1.000000	1.000000	0.010000	
25%	2.767965e+07	2.012958e+07	40.683731	-73.980638	80.000000	30.000000	4.000000	0.230000	
50%	5.099994e+07	1.082499e+08	40.722305	-73.949385	125.000000	30.000000	14.000000	0.670000	
75%	7.369364e+17	3.279947e+08	40.763106	-73.916735	199.000000	30.000000	49.000000	1.790000	
max	1.054376e+18	5.504035e+08	40.911147	-73.713650	100000.000000	1124.000000	1865.000000	75.490000	7

```
 ## Minimum and Maximum Prices (Price Range).
price_range = df['price'].agg(['min', 'max'])
price_range
```

	price
min	10.0
max	100000.0

dtype: float64

```
df['beds'].unique()
```

```
array([ 1,  2,  3,  4,  8,  6,  9, 10,  7,  5, 12, 21, 11, 14, 13, 18, 42])
```

```
df['baths'].unique()
```

```
array(['1', '2', '4', '3.5', '1.5', '3', '0', '2.5', '5', '4.5', '6',
       '5.5', 'Not specified', '11.5', '6.5', '7', '15.5'], dtype=object)
```

```
df['bedrooms'].unique()
```

```
array([ 2,  1,  0,  6,  3,  5,  4,  7, 15,  9,  8, 14])
```

```
# Replace 'Not specified' with 0 and convert to float
df['baths'] = df['baths'].replace('Not specified', 0).astype(float)

# Convert to integer (rounding down decimal values)
df['baths'] = df['baths'].astype(int)
```

```
df['name'] = df['name'].str.extract(r'^(.*?·)')
df['name'] = df['name'].str.replace('·', '', regex=True).str.strip()
```

```
df['name'].head(3)
```

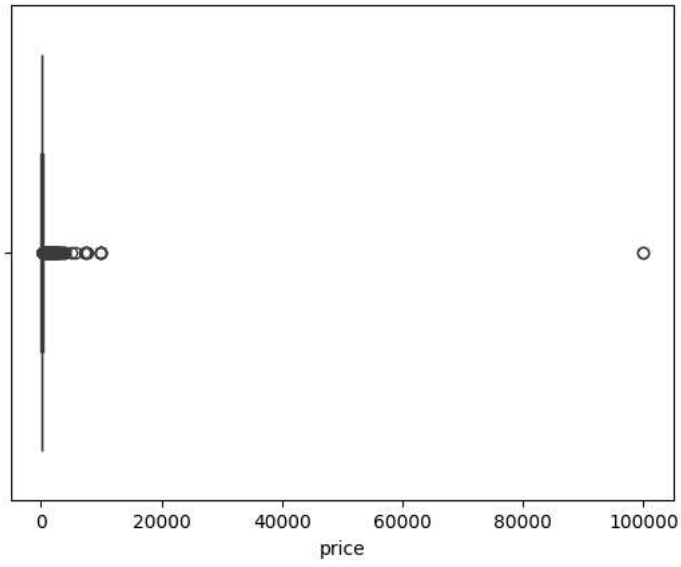
	name
1	Rental unit in New York
2	Rental unit in New York
3	Rental unit in New York

dtype: object

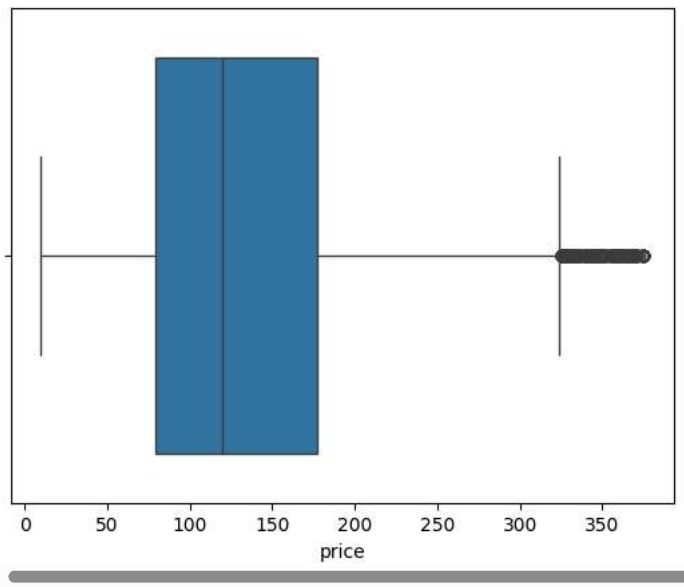
```
df.duplicated().sum()
```

```
0
```

```
sns.boxplot(x = df['price'])

plt.show()
```




```
# prompt: # writing a outlier function for removing outliers in important columns PRICE
def remove_outliers_iqr(data, column, threshold=1.5):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - threshold * IQR
    upper_bound = Q3 + threshold * IQR
    data_filtered = data[(data[column] >= lower_bound) & (data[column] <= upper_bound)]
    return data_filtered

# Example usage for 'price' column
df = remove_outliers_iqr(df, 'price')

#Now you can use df_no_outliers for further analysis
sns.boxplot(x=df['price'])
plt.show()
```




```
df['price'].agg(['min', 'max'])
```

	price
min	10.0
max	375.0

dtype: float64

EDA

(1) Distribution Of Airbnb Bookings Price Range Using Histogram

```
# Set theme and figure size
sns.set_theme(style='darkgrid')
plt.figure(figsize=(12, 5))

# Plot histogram
sns.histplot(df['price'], color='r', kde=True)

# Labels and title
plt.xlabel('Price', fontsize=14)
plt.ylabel('Density', fontsize=14)
plt.title('Distribution of Airbnb Prices', fontsize=15)

plt.show()
```

observations -->

The range of prices being charged on Airbnb appears to be from 10 to 350 dollars , with the majority of listings falling in the price range of 20 to 200 dollars.

The distribution of prices appears to have a peak in the 50 to 150 dollars range, with a relatively lower density of listings in higher and lower price ranges.

There may be fewer listings available at prices above 300 dollars, as the density of listings drops significantly in this range.

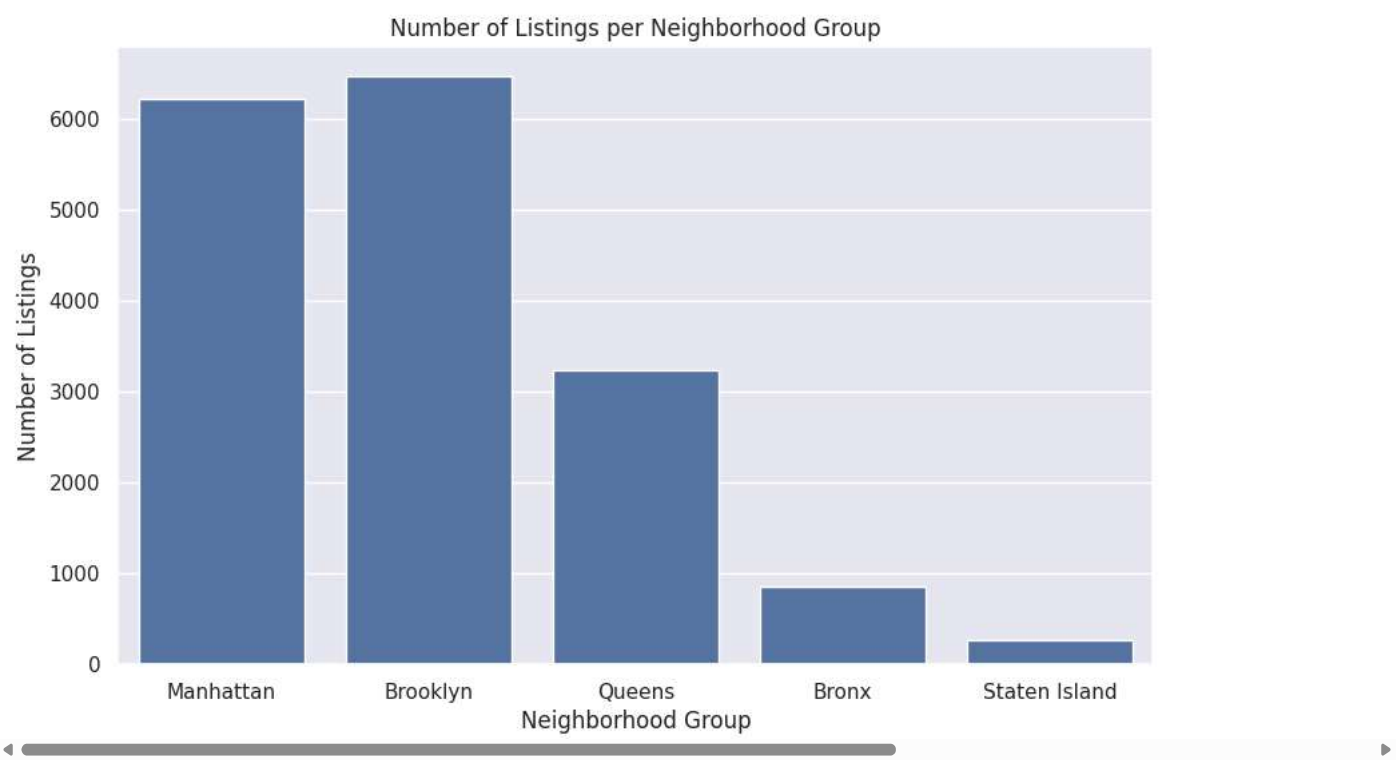
(2) Total Listing/Property count in Each Neighborhood Group using Count plot

```
# Count the occurrences of each neighborhood group
neighborhood_group_counts = df['n_group'].value_counts().reset_index()

# Rename columns
neighborhood_group_counts.columns = ['Neighborhood Group', 'Count']
neighborhood_group_counts
```

	Neighborhood Group	Count
0	Brooklyn	6466
1	Manhattan	6214
2	Queens	3239
3	Bronx	858
4	Staten Island	266

Next steps: Generate code with neighborhood_group_counts | View recommended plots | New interactive sheet

```
# You can also visualize this using a bar plot (optional)
plt.figure(figsize=(10, 6))
sns.countplot(x='n_group', data=df)
plt.title('Number of Listings per Neighborhood Group')
plt.xlabel('Neighborhood Group')
plt.ylabel('Number of Listings')
plt.show()
```



Observations -->

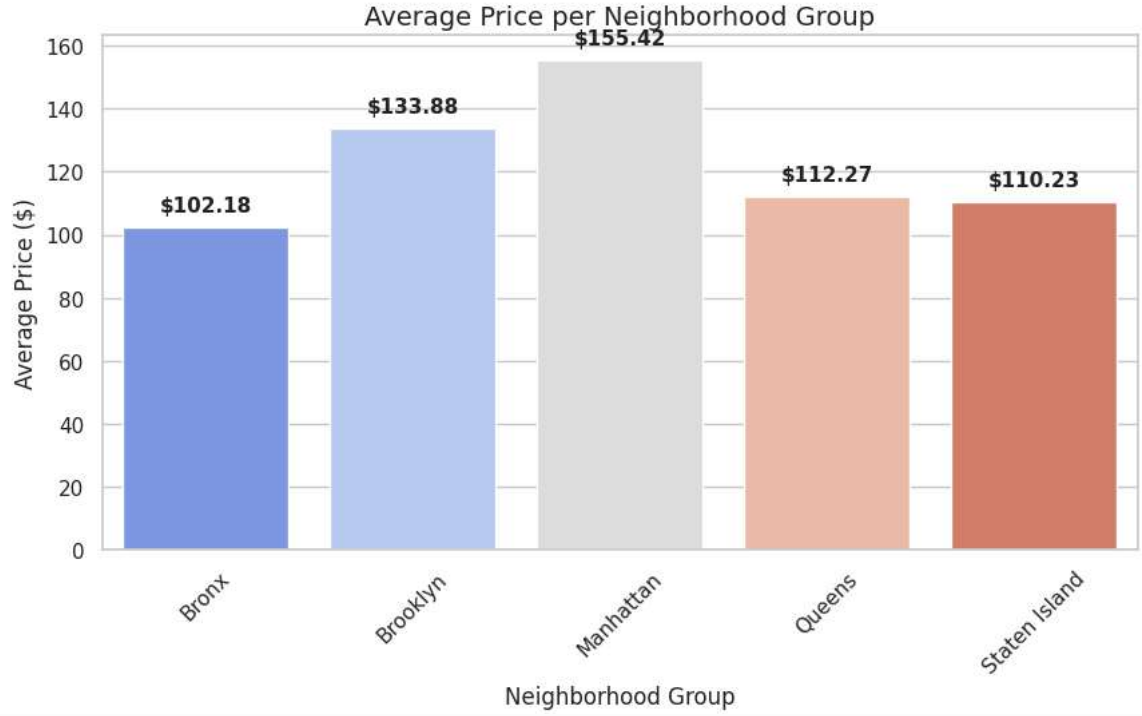
- Manhattan and Brooklyn have the highest number of listings on Airbnb, with over 6,200 listings each.
- Queens and the Bronx have significantly fewer listings compared to Manhattan and Brooklyn, with 3,000 and 1,000 listings, respectively
- Staten Island has the fewest number of listings, with only 266.
- The distribution of listings across the different neighborhood groups is skewed, with a concentration of listings in Manhattan and Brooklyn.
- This could suggest that the demand for Airbnb rentals is higher in Brooklyn compared to the other neighborhoods, leading to a higher concentration of listings in this area.
- Alternatively, it could be that the supply of listings is higher in Brooklyn due to a higher number of homeowners or property owners in this neighborhood who are willing to list their properties on Airbnb.

(3) Average Price Of Each Neighborhood Group using Point Plot

```
# Calculate the average price for each neighborhood group
avg_price_per_group = df.groupby('n_group')['price'].mean().reset_index()
avg_price_per_group.columns = ['Neighborhood Group', 'Average Price']
```

```
print(avg_price_per_group)
# Set style
sns.set_theme(style="whitegrid")

# Create the bar chart
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Neighborhood Group', y='Average Price', hue='Neighborhood Group', data=avg_price_per_group, palette='coolwarm')

# Add value annotations on bars
for i, v in enumerate(avg_price_per_group['Average Price']):
    ax.text(i, v + 5, f'${v:.2f}', ha='center', fontsize=11, fontweight='bold')

# Labels and title
plt.xlabel('Neighborhood Group', fontsize=12)
plt.ylabel('Average Price ($)', fontsize=12)
plt.title('Average Price per Neighborhood Group', fontsize=14)
plt.xticks(rotation=45)

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

```
  Neighborhood Group  Average Price
0              Bronx     102.179487
1           Brooklyn     133.879060
2          Manhattan     155.422272
3             Queens     112.274467
4      Staten Island     110.233083
```



Observations -->

- The **average price of a listing in New York City** varies significantly across different neighborhoods, with **Manhattan having the highest average price** $(155.42/day) * *and the * *Bronx having the lowest (102.18/day)$.
- In the second graph, **price distribution is very high in Manhattan and Brooklyn**. However, **Manhattan has more variety in price range**, as seen in the second violin plot.
- The **average price generally increases as you move from the outer boroughs (Bronx, Queens, Staten Island) towards the center of the city (Manhattan and Brooklyn)**.
- **Brooklyn** $(133.88/day) has a higher average price than Queens$**(112.27/day) and Staten Island ($110.23/day), but lower than Manhattan.**
- **Queens and Staten Island have relatively similar average prices**, despite being in different parts of the city.
- The data suggests that **the cost of living in New York City is highest in Manhattan**, likely due to **high demand, commercial significance, and dense population**. =

(4) Price Distribution Of Each Neighborhood Group using Violin Plot

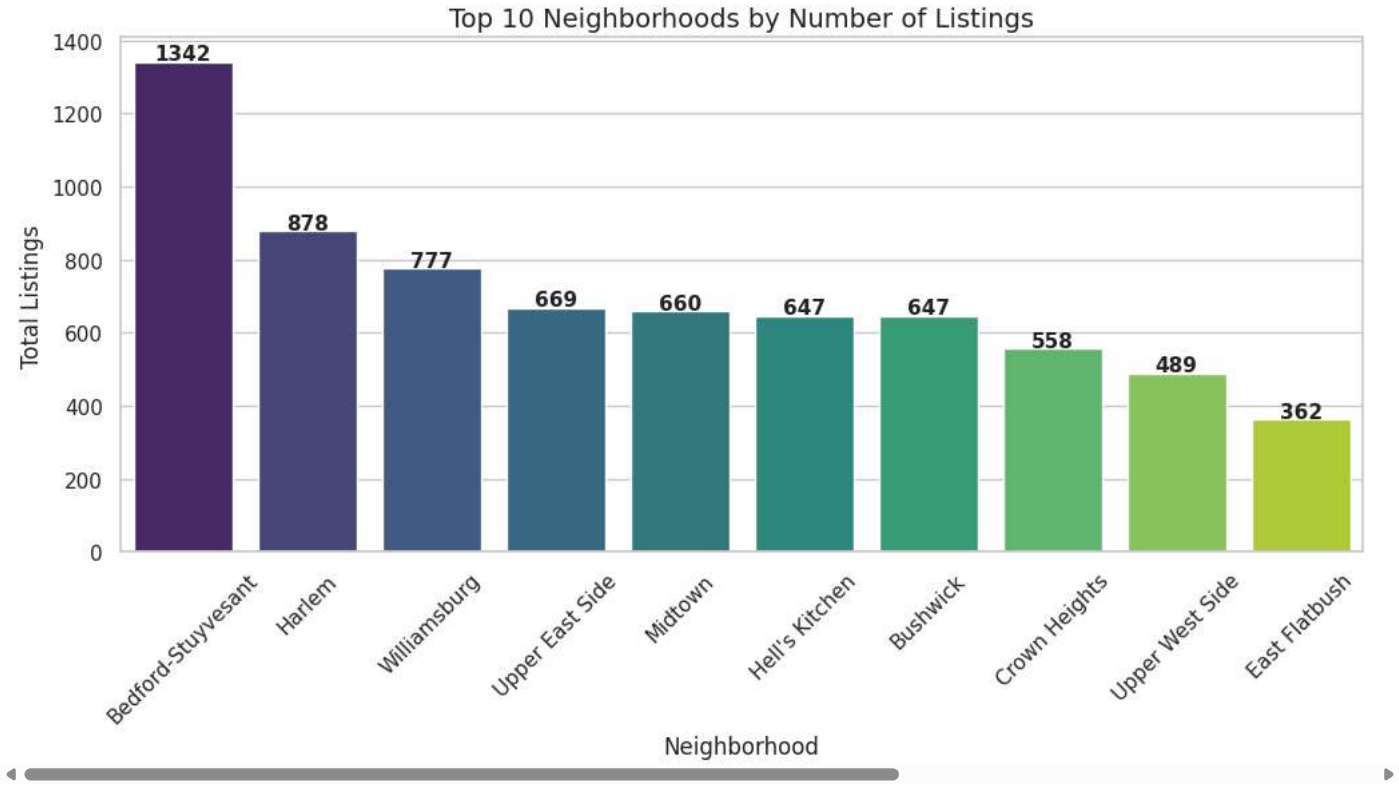
```
# Create the violin plot for price distribution in each Neighbourhood_groups
plt.figure(figsize=(12, 6))

sns.violinplot(x='n_group', y='price', data=df, palette='Set3')

plt.title('Price Distribution by Neighborhood Group', fontsize=16)
plt.xlabel('Neighborhood Group', fontsize=14)
plt.ylabel('Price', fontsize=14)
plt.show()
```

```
<ipython-input-34-3252ae8183a2>:4: FutureWarning:

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

  sns.violinplot(x='n_group', y='price', data=df, palette='Set3')
```

(4) Top Neighborhoods by Listing/property using Bar plot

```
# Count the number of listings in each neighborhood
top_neighborhoods = df['neighbourhood'].value_counts().head(10).reset_index()
top_neighborhoods.columns = ['Neighborhood', 'Total Listings']
print(top_neighborhoods)

# Set style
sns.set_theme(style="whitegrid")

# Create the bar plot
plt.figure(figsize=(12, 5))
ax = sns.barplot(x='Neighborhood', y='Total Listings', hue='Neighborhood',
                 data=top_neighborhoods, palette='viridis', legend=False)

# Add value annotations
for i, v in enumerate(top_neighborhoods['Total Listings']):
    ax.text(i, v + 5, str(v), ha='center', fontsize=11, fontweight='bold')

# Labels and title
plt.xlabel('Neighborhood', fontsize=12)
plt.ylabel('Total Listings', fontsize=12)
plt.title('Top 10 Neighborhoods by Number of Listings', fontsize=14)
plt.xticks(rotation=45)

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

```
          Neighborhood  Total Listings
0  Bedford-Stuyvesant            1342
1              Harlem             878
2        Williamsburg             777
3     Upper East Side             669
4             Midtown             660
5      Hell's Kitchen             647
6            Bushwick             647
7       Crown Heights             558
8     Upper West Side             489
9       East Flatbush             362
```



Observations -->

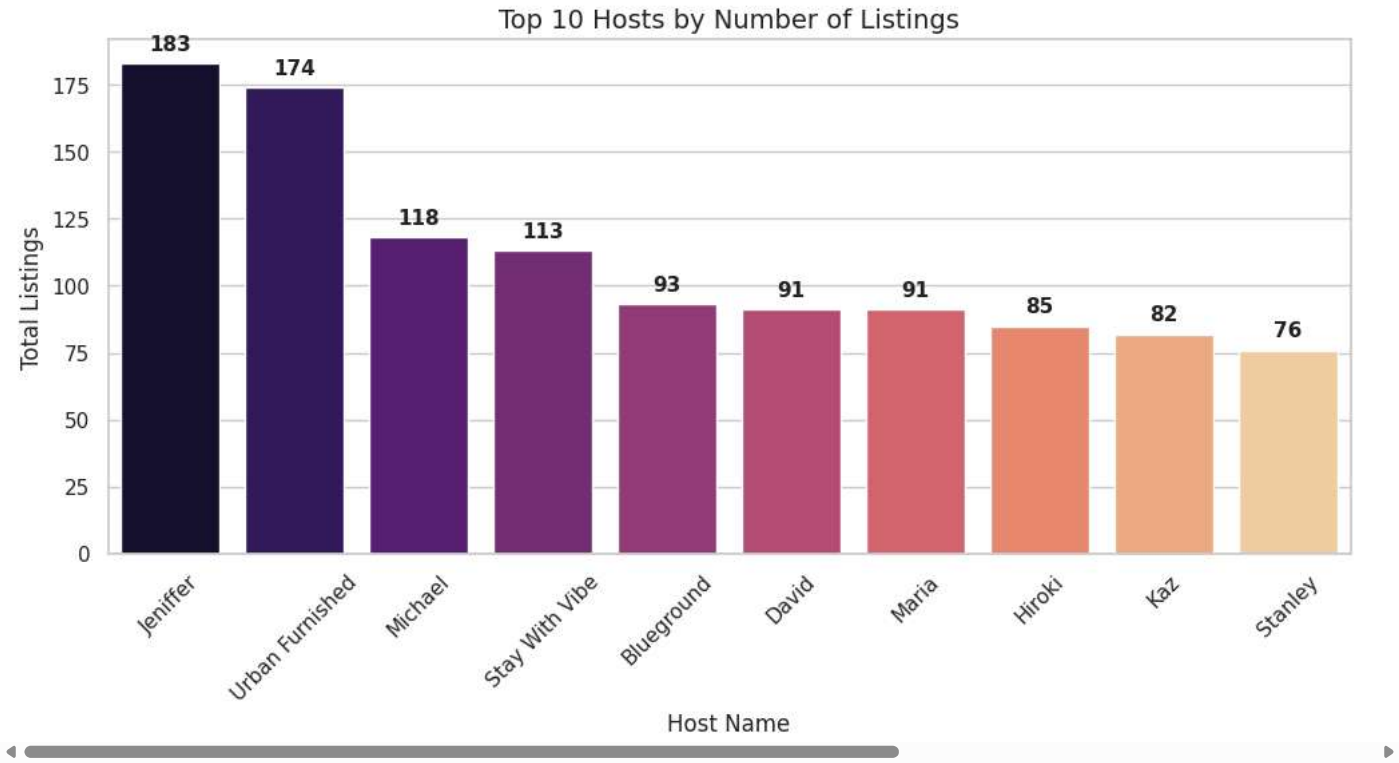
- **Bedford-Stuyvesant** has the **highest number of listings (1,342)**, making it a popular area for rentals.
- **Harlem (878) and Williamsburg (777)** also have a significant number of listings, indicating high rental activity.
- **Midtown (660) and Upper East Side (669)** are among the top-ranked neighborhoods, likely due to their central locations and accessibility.
- **Hell's Kitchen (647) and Bushwick (647) have the same number of listings**, showing strong demand in these areas.
- **Crown Heights (558) and Upper West Side (489)** have moderate listings compared to the top-ranked areas.
- **East Flatbush (362) has the lowest among the top 10**, but still a considerable rental presence.

(5) Top Hosts With More Listing/Property using Bar chart

```
# Count the number of listings per host and get the top 10
top_hosts = df['host_name'].value_counts().head(10).reset_index()
top_hosts.columns = ['Host Name', 'Total Listings']
print(top_hosts)
# Set style
sns.set_theme(style="whitegrid")

# Create the bar plot
plt.figure(figsize=(12, 5))
ax = sns.barplot(x='Host Name', y='Total Listings', hue='Host Name',
                 data=top_hosts, palette='magma', legend=False)

# Add value annotations
for i, v in enumerate(top_hosts['Total Listings']):
    ax.text(i, v + 5, str(v), ha='center', fontsize=11, fontweight='bold')

# Labels and title
plt.xlabel('Host Name', fontsize=12)
plt.ylabel('Total Listings', fontsize=12)
```

```
plt.title('Top 10 Hosts by Number of Listings', fontsize=14)
plt.xticks(rotation=45)

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

```
          Host Name  Total Listings
0         Jeniffer             183
1  Urban Furnished             174
2          Michael             118
3   Stay With Vibe             113
4       Blueground              93
5            David              91
6            Maria              91
7           Hiroki              85
8              Kaz              82
9          Stanley              76
```

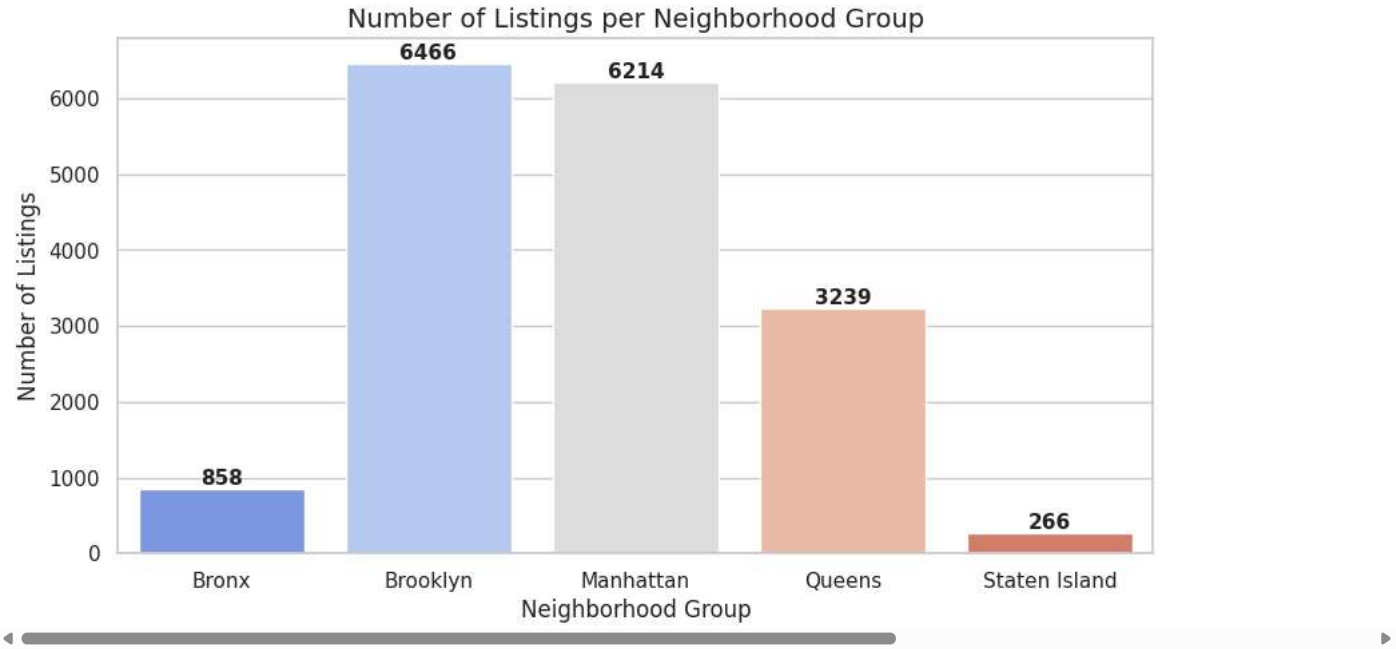


(6) Number Of Active Hosts Per Location

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

# Count listings per neighborhood group
listings_per_location = df.groupby('n_group')['id'].count().reset_index()
listings_per_location.columns = ['Neighborhood Group', 'Listing Count']
print(listings_per_location)
# Set theme
sns.set_theme(style="whitegrid")

# Create the bar plot
plt.figure(figsize=(10, 5))
ax = sns.barplot(x='Neighborhood Group', y='Listing Count', hue='Neighborhood Group',
                 data=listings_per_location, palette='coolwarm', legend=False)

# Add value annotations
for i, v in enumerate(listings_per_location['Listing Count']):
    ax.text(i, v + 50, str(v), ha='center', fontsize=11, fontweight='bold')

# Labels and title
plt.xlabel('Neighborhood Group', fontsize=12)
plt.ylabel('Number of Listings', fontsize=12)
plt.title('Number of Listings per Neighborhood Group', fontsize=14)

# Show the plot
```

```
plt.show()
```

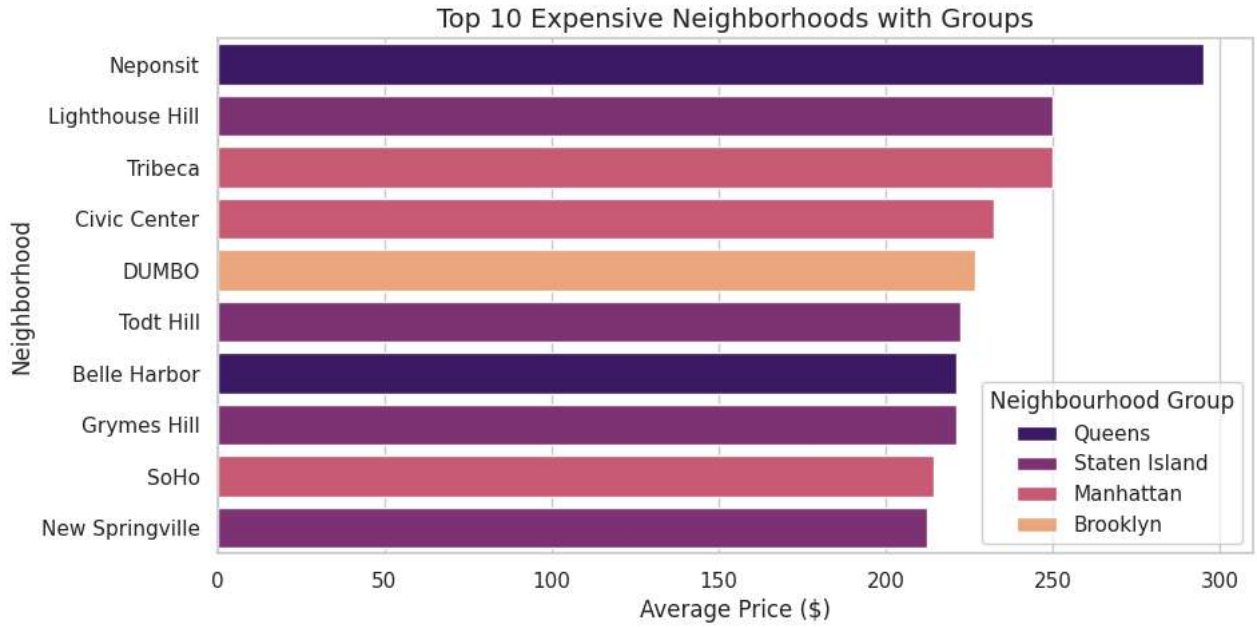
```
  Neighborhood Group  Listing Count
0              Bronx            858
1           Brooklyn           6466
2          Manhattan           6214
3             Queens           3239
4      Staten Island            266
```



(7) Top 10 Expensive Neighborhoods with Groups

```
# Calculate average price per neighborhood with neighborhood group
avg_price_neighborhood = df.groupby(['n_group', 'neighbourhood'])['price'].mean().reset_index()

# Rename columns
avg_price_neighborhood.columns = ['Neighbourhood Group', 'Neighborhood', 'Average Price']

# Sort by highest average price and select the top 10
top_10_neighborhoods = avg_price_neighborhood.sort_values(by='Average Price', ascending=False).head(10)

# Display table without index
print(top_10_neighborhoods.to_string(index=False))

# Plot the top 10 expensive neighborhoods
plt.figure(figsize=(10, 5))
sns.barplot(y='Neighborhood', x='Average Price', hue='Neighbourhood Group', data=top_10_neighborhoods, palette='magma')

# Title and labels
plt.title('Top 10 Expensive Neighborhoods with Groups', fontsize=14)
plt.xlabel('Average Price ($)', fontsize=12)
plt.ylabel('Neighborhood', fontsize=12)

# Show legend
plt.legend(title='Neighbourhood Group')

# Show plot
plt.show()
```

```
Neighbourhood Group    Neighborhood  Average Price
             Queens        Neponsit     295.000000
      Staten Island Lighthouse Hill     250.000000
          Manhattan         Tribeca     249.971429
          Manhattan    Civic Center     232.200000
           Brooklyn           DUMBO     226.727273
      Staten Island       Todt Hill     222.500000
             Queens    Belle Harbor     221.285714
      Staten Island     Grymes Hill     221.000000
          Manhattan            SoHo     214.146067
      Staten Island New Springville     212.500000
```



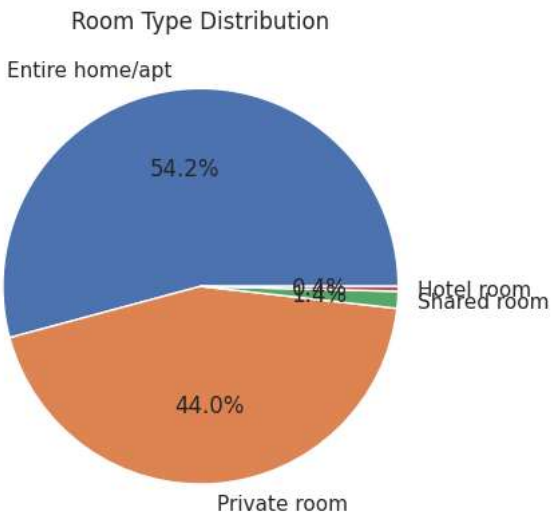
(8) Total Counts Of Each Room Type *

```
# create a new DataFrame that displays the number of listings of each room type in the Airbnb NYC dataset
top_room_type = df['room_type'].value_counts().reset_index()

# rename the columns of the resulting DataFrame to 'Room_Type' and 'Total_counts'
top_room_type.columns = ['Room_Type', 'Total_counts']

# display the resulting DataFrame
top_room_type
```

	Room_Type	Total_counts
0	Entire home/apt	9234
1	Private room	7505
2	Shared room	235
3	Hotel room	69

Next steps: Generate code with top_room_type | View recommended plots | New interactive sheet

```
plt.pie(top_room_type['Total_counts'], labels=top_room_type['Room_Type'], autopct='%1.1f%%')
plt.title('Room Type Distribution')
plt.show()
```



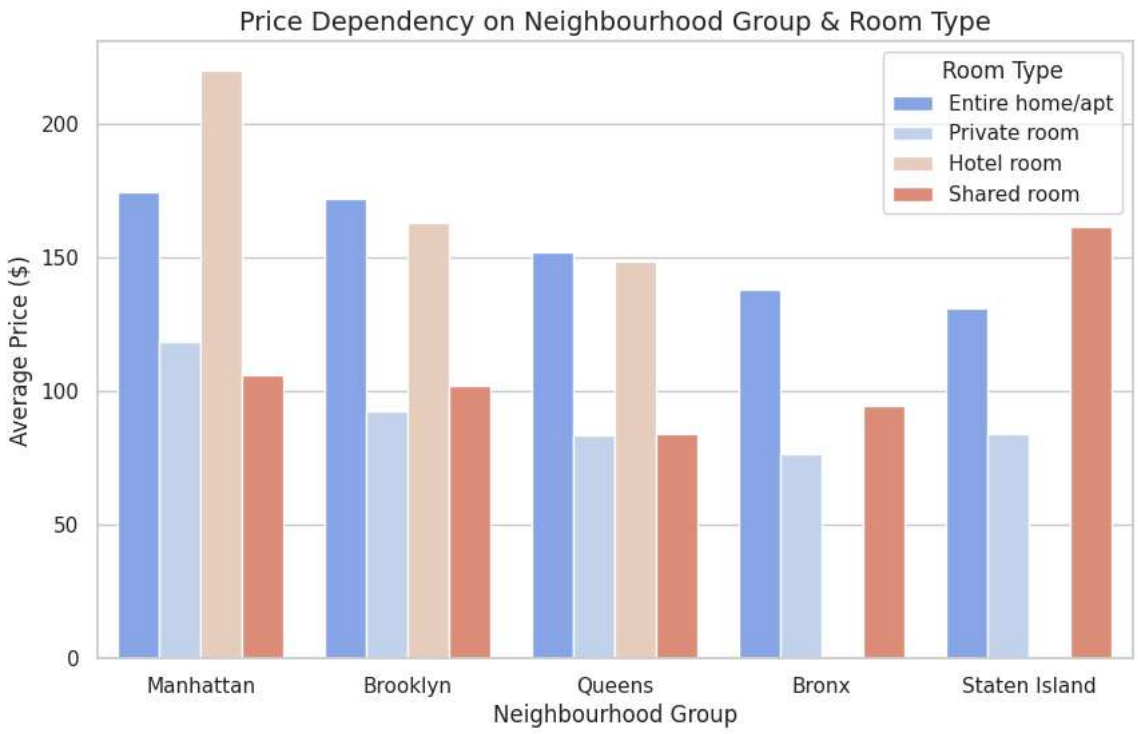
(9) Average price per room type

```
# Calculate the average price per room type
avg_price_per_room_type = df.groupby('room_type')['price'].mean().reset_index()

# Rename columns
avg_price_per_room_type.columns = ['Room Type', 'Average Price']

# Print the result
avg_price_per_room_type
```

	Room Type	Average Price
0	Entire home/apt	168.095300
1	Hotel room	212.275362
2	Private room	96.210660
3	Shared room	98.744681

Next steps: Generate code with avg_price_per_room_type | View recommended plots | New interactive sheet

```
plt.figure(figsize=(10, 6))
sns.set_theme(style="whitegrid")

sns.barplot(
    data=df,
    x='n_group',
    y='price',
    hue='room_type',
    palette='coolwarm',
    estimator=np.mean,
    errorbar=None  # Removes error bars for a cleaner look
)

plt.xlabel('Neighbourhood Group', fontsize=12)
plt.ylabel('Average Price ($)', fontsize=12)
plt.title('Price Dependency on Neighbourhood Group & Room Type', fontsize=14)
plt.legend(title='Room Type')


plt.show()
```



(10) Average price per bed

```
# average price per bed
df['price per bed']= df['price']/df['beds']
df.groupby(by='n_group')['price per bed'].mean()
```

n_group	price per bed
Bronx	72.833934
Brooklyn	92.477666
Manhattan	117.208203
Queens	73.859139
Staten Island	67.798568

dtype: float64

(11) Ratings

Generate | average rating per n_group and neighbourhood | Close

1 of 1 | Undo Changes | Use code with caution

```
# Calculate average rating per n_group and neighbourhood
average_rating = df.groupby(['n_group'])['rating'].mean().reset_index()
average_rating
```

	n_group	rating
0	Bronx	4.092692
1	Brooklyn	4.040633
2	Manhattan	3.625695
3	Queens	4.065181
4	Staten Island	4.305489

Next steps: Generate code with average_rating | View recommended plots | New interactive sheet

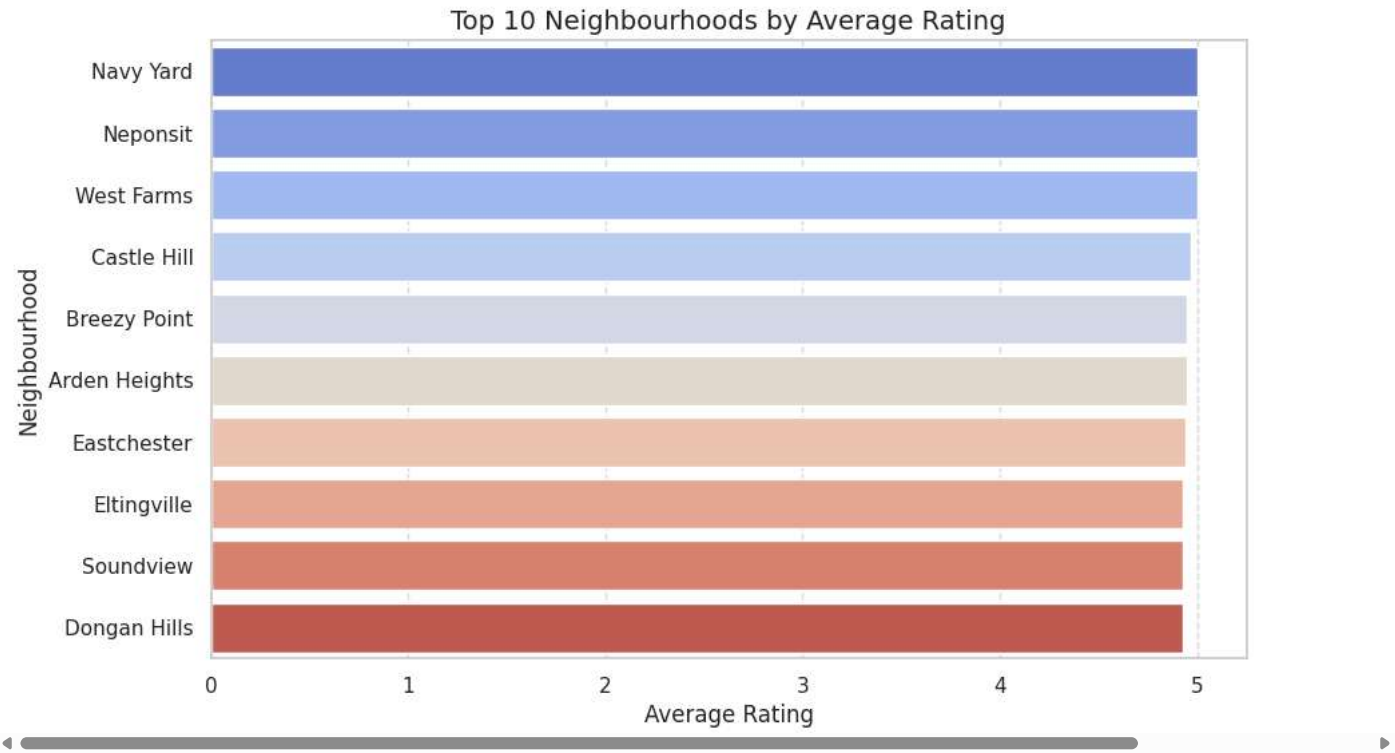
(12) 'Top 10 Neighbourhoods by Average Rating'

```
top_10_avg_rating = average_rating.nlargest(10, 'rating')
print(top_10_avg_rating)
```

```
     neighbourhood    rating
139      Navy Yard  5.000000
140       Neponsit  5.000000
208     West Farms  5.000000
32     Castle Hill  4.970000
21    Breezy Point  4.947500
1    Arden Heights  4.945000
66     Eastchester  4.943333
70     Eltingville  4.926667
179      Soundview  4.926250
56    Dongan Hills  4.925000
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=top_10_avg_rating, x='rating', y='neighbourhood', palette='coolwarm')

plt.xlabel('Average Rating', fontsize=12)
plt.ylabel('Neighbourhood', fontsize=12)
plt.title('Top 10 Neighbourhoods by Average Rating', fontsize=14)
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.show()
```

```
<ipython-input-60-8d2f308752f1>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.barplot(data=top_10_avg_rating, x='rating', y='neighbourhood', palette='coolwarm')
```



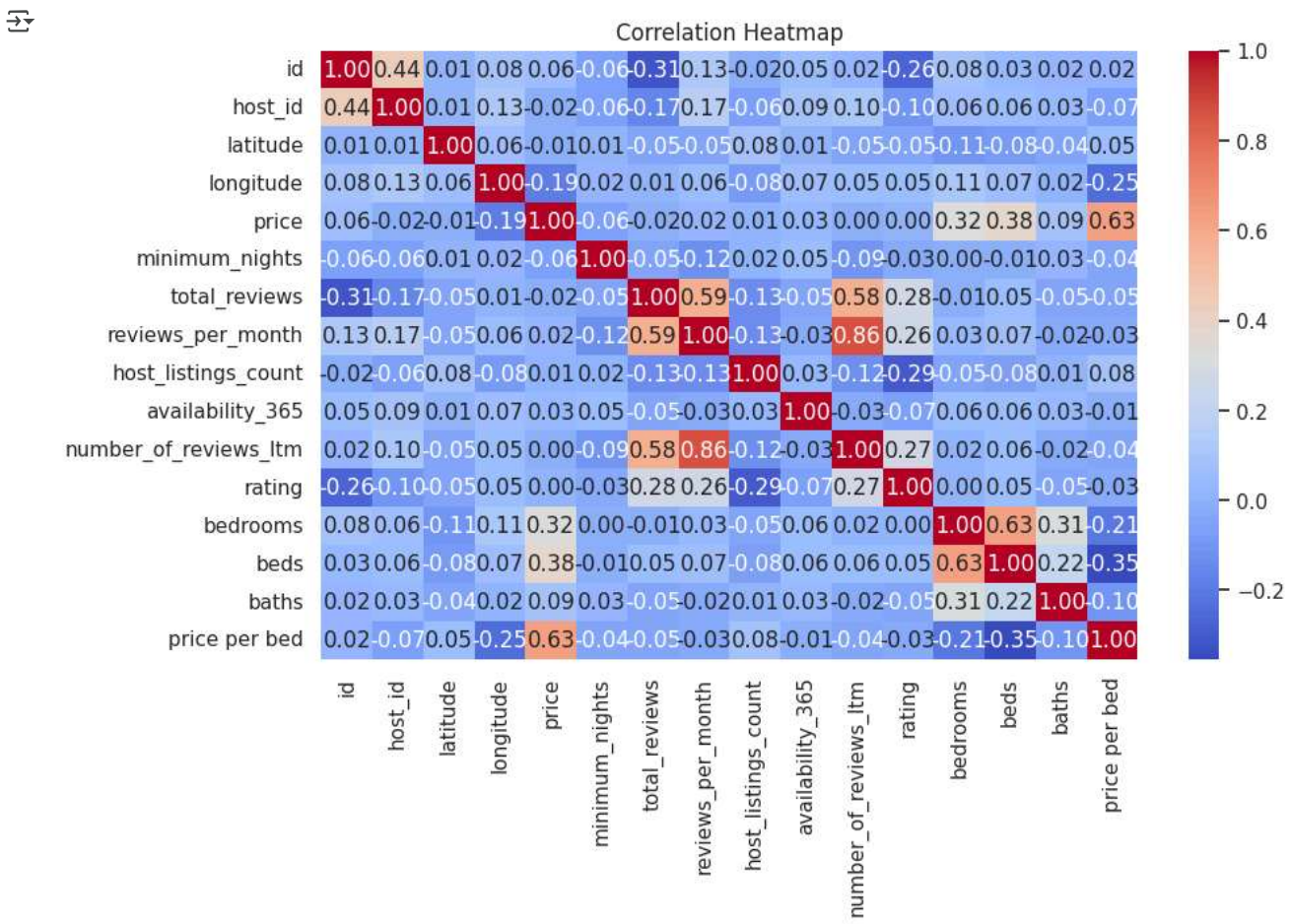
(13) Top 10 Reviews per Month

```
top_10_reviews = df.groupby('neighbourhood')['reviews_per_month'].mean().nlargest(10).reset_index()
print(top_10_reviews)
```

```
         neighbourhood  reviews_per_month
0             Huguenot           3.005000
1       New Dorp Beach           2.855000
2    Theater District            2.566442
3              Clifton           2.513333
```

```
4       Graniteville          2.513333
5       East Elmhurst         2.403019
6     Mariners Harbor         2.333000
7  University Heights         2.146000
8          Morrisania         2.128333
9         Shore Acres         2.115000
```

```
numeric_df = df.select_dtypes(include=np.number)
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```




Must See Things in Manhattan A-Z
Sygic Travel
A Statue of Liberty
B Ellis Island
C World Financial Center
D 9/11 Memorial
E Wall Street
F Battery Park
G Chelsea Market
H Chinatown
I St Patrick's Old Cathedral
J Madison Square Garden
K Macy's
L Empire State Building
M Flatiron Building
N Metropolitan Opera House
O Broadway
P Carnegie Hall
Q Times Square
R Rockefeller Center
S Chrysler Building
T United Nations Headquarters
U Strawberry Fields
V Central Park
W Metropolitan Museum of Art
X Guggenheim Museum
Y Brooklyn Bridge
Z East River Park
HOBOKEN
NEW YORK
ELLIS ISLAND
LIBERTY ISLAND
GOVERNORS ISLAND
BELMONT ISLAND
ROOSEVELT ISLAND
500 m
2000 ft
© Tripomatic, 2012. Built using data from Tripomatic, CloudMade and OpenStreetMap.org contributors. Licensed under CC-BY-SA 2.0

Observations -->