What is AirBnB Platform?

In [ ]:
```
AirBnB is a leading online marketplace that connects people looking to rent ou
Founded in 2008, the platform has grown exponentially, offering over 7 millior
AirBnB provides a wide range of lodging options, from single rooms and apartme
This diversity allows travelers to find accommodations that suit their prefere
```

Importance of Analyzing Booking Data

In [ ]:
```
Analyzing booking data on AirBnB is crucial for several reasons:
```

1.Understanding Market Trends:

By examining booking patterns, one can identify peak seasons, popular destinations, and emerging travel trends. This information is invaluable for hosts to optimize their listings and pricing strategies.

2.Enhancing Guest Experience:

Insights from booking data help identify what guests value most, enabling hosts to tailor their offerings to meet guest expectations. For instance, understanding the demand for certain amenities or property types can lead to more targeted and effective listings.

3.Improving Host Performance:

Analysis of booking data can reveal key factors that contribute to higher occupancy rates and better reviews. Hosts can use this information to improve their property management practices and increase their revenue.

4.Strategic Decision-Making:

For AirBnB as a platform, booking data analysis is essential for strategic planning. It helps in understanding the competitive landscape, assessing the effectiveness of marketing campaigns, and making informed decisions about platform enhancements.

5.Enhancing Safety and Compliance

By analyzing booking data, AirBnB can detect unusual patterns that may indicate fraudulent activity or violations of local regulations. This proactive approach ensures a safer and more reliable platform for both hosts and guests.

➡ Problem statements

(1) Find Distribution Of Airbnb Bookings Price Range.

(2) Find Total Listing/Property count in Each Neighborhood Group in NYC.

(3) Find Average Price Of listings/property in each Neighborhood Groups and also Neighborhoods.

(4) Find Top Neighborhoods and Hosts by Listing/property in entire NYC.

(5) Find the Number Of Active Hosts Per Location by Each Neighborhood Groups.

(6) Find Total Counts Of Each Room Types in entire NYC.

(7) Find Stay Requirement counts by Minimum Nights.

(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.

(10) Find Best location listing/property location for travelers.

(11) Find also best location listing/property location for Hosts.

(12) Find Price variations in NYC Neighborhood groups.

Exploring the AirBnB Dataset in Python

Dataset Link -Airbnb

Step 1: Importing Necessary Libraries and Loading the AirBNB Dataset

This script imports essential data manipulation and visualization libraries, loads an Airbnb dataset from a CSV file, and displays the first few rows of the dataset. This initial step helps in understanding the structure and content of the data before performing further analysis or visualization.

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv('Airbnb_Analysis (1).csv')
```

In [4]:
```
df
```

Out[4]:

	id	name	host_id	host_name	neighbourhood_group	n
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	
...	...	...	...	...	...	
48890	36484665	Charming one bedroom - newly renovated rowhouse	8232441	Sabrina	Brooklyn	
48891	36485057	Affordable room in Bushwick/East Williamsburg	6570630	Marisol	Brooklyn	
48892	36485431	Sunny Studio at Historical Neighborhood	23492952	Ilgar & Aysel	Manhattan	
48893	36485609	43rd St. Time Square-cozy single bed	30985759	Taz	Manhattan	
48894	36487245	Trendy duplex in the very heart of Hell's Kitchen	68119814	Christophe	Manhattan	

48895 rows × 16 columns

Step 2: Check the column names in the Dataset

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365'],
      dtype='object')
```

Step 3: Check for Missing Values

In [6]:
```
print(df.isnull().sum())
```

```
id                                    0
name                                 16
host_id                               0
host_name                            21
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                       10052
reviews_per_month                 10052
calculated_host_listings_count        0
availability_365                      0
dtype: int64
```

Step 4: Handle Missing Values

In [7]:
```
df['name'].value_counts()
```

Out[7]:
```
name
Hillside Hotel                                        18
Home away from home                                   17
New york Multi-unit building                          16
Brooklyn Apartment                                    12
Loft Suite @ The Box House Hotel                      11
                                                      ..
Brownstone garden 2 bedroom duplex, Central Park       1
Bright Cozy Private Room near Columbia Univ            1
1 bdrm/large studio in a great location                1
Cozy Private Room #2 Two Beds Near JFK and J Train     1
Trendy duplex in the very heart of Hell's Kitchen      1
Name: count, Length: 47896, dtype: int64
```

In [8]:
```
df['name'].fillna(df['name'].mode()[0],inplace=True)
```

```
C:\Users\subha\AppData\Local\Temp\ipykernel_17640\3835375673.py:1: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chain
ed assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df['name'].fillna(df['name'].mode()[0],inplace=True)
```

In [16]:
```
print(df.isnull().sum())
```

```
id                                0
name                              0
host_id                           0
host_name                         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
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

In [10]:
```
df['host_name'].fillna(df['host_name'].mode()[0],inplace=True)
```

```
C:\Users\subha\AppData\Local\Temp\ipykernel_17640\2917131993.py:1: FutureWarnin
g: A value is trying to be set on a copy of a DataFrame or Series through chain
ed assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df['host_name'].fillna(df['host_name'].mode()[0],inplace=True)
```

In [15]:
```
# Convert 'last review' to datetime and handle errors
df['last_review'] = pd.to_datetime(df['last_review'], errors='coerce')

# Fill missing values
```

```
df.fillna({'reviews_per_month': 0, 'last_review': df['last_review'].min()}, in

df.fillna({'last_review': 0, 'last_review': df['last_review'].min()}, inplace=
```

Step 5: Correct Data Types

Ensure that all columns have the correct data types.

In [17]:
```
df['price'] = df['price'].replace('[\$,]', '', regex=True).astype(float)
```

```
<>:1: SyntaxWarning: invalid escape sequence '\$'
<>:1: SyntaxWarning: invalid escape sequence '\$'
C:\Users\subha\AppData\Local\Temp\ipykernel_17640\2337335738.py:1: SyntaxWarnin
g: invalid escape sequence '\$'
  df['price'] = df['price'].replace('[\$,]', '', regex=True).astype(float)
```

Step 6: Remove Duplicates

Check for and remove any duplicate records.

In [18]:
```
df.drop_duplicates(inplace=True)
```

Step 7: Confirm Data Cleaning

Verify that the data cleaning steps were successful.

In [19]:
```
print(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                            48895 non-null  object
 2   host_id                         48895 non-null  int64
 3   host_name                       48895 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  float64
 10  minimum_nights                  48895 non-null  int64
 11  number_of_reviews               48895 non-null  int64
 12  last_review                     48895 non-null  datetime64[ns]
 13  reviews_per_month               48895 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365                48895 non-null  int64
dtypes: datetime64[ns](1), float64(4), int64(6), object(5)
memory usage: 6.0+ MB
None
```

Step 8: Descriptive Statistics

The df.describe() function in pandas generates descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values. This function is useful for understanding the basic statistical properties of the data.

In [20]:
```
print(df.describe())
```

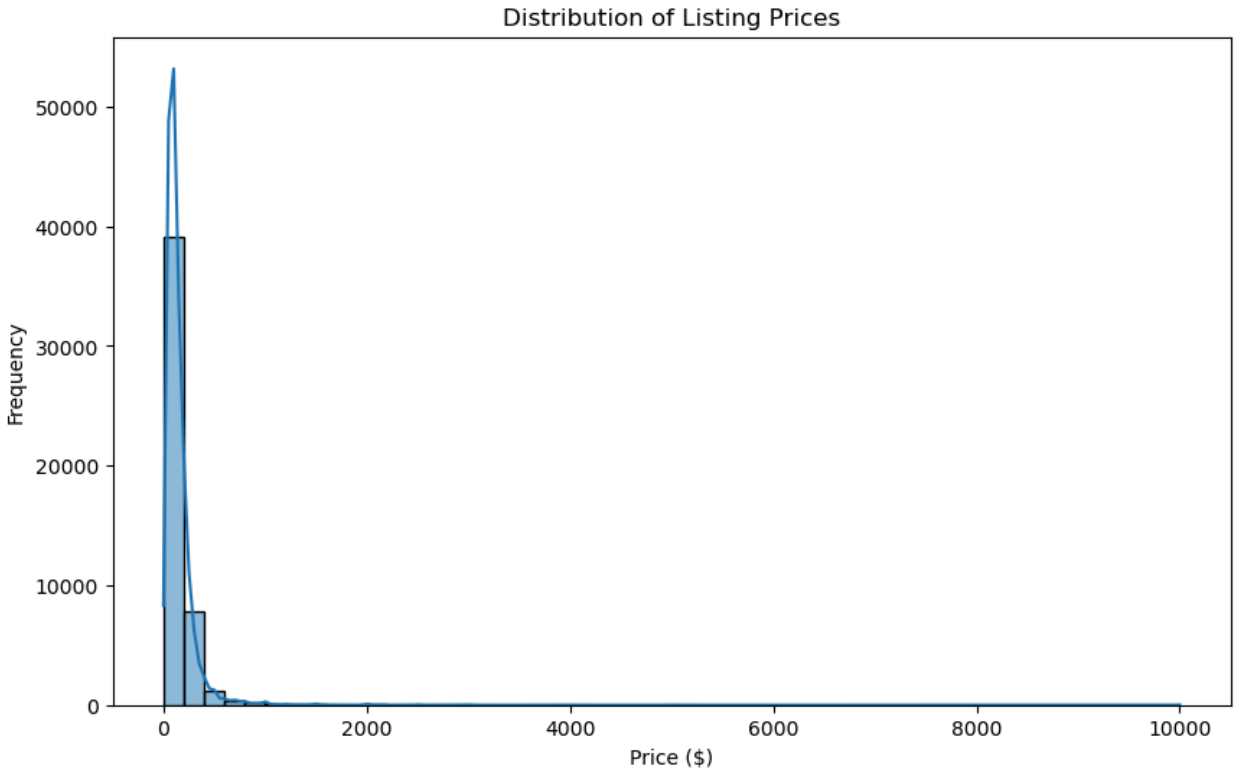
```
                 id       host_id      latitude     longitude         price  \
count  4.889500e+04  4.889500e+04  48895.000000  48895.000000  48895.000000
mean   1.901714e+07  6.762001e+07     40.728949    -73.952170    152.720687
min    2.539000e+03  2.438000e+03     40.499790    -74.244420      0.000000
25%    9.471945e+06  7.822033e+06     40.690100    -73.983070     69.000000
50%    1.967728e+07  3.079382e+07     40.723070    -73.955680    106.000000
75%    2.915218e+07  1.074344e+08     40.763115    -73.936275    175.000000
max    3.648724e+07  2.743213e+08     40.913060    -73.712990  10000.000000
std    1.098311e+07  7.861097e+07      0.054530      0.046157    240.154170

       minimum_nights  number_of_reviews                    last_review  \
count    48895.000000       48895.000000                          48895
mean         7.029962          23.274466  2017-03-18 07:43:12.191430656
min          1.000000           0.000000            2011-03-28 00:00:00
25%          1.000000           1.000000            2016-03-24 00:00:00
50%          3.000000           5.000000            2019-01-03 00:00:00
75%          5.000000          24.000000            2019-06-19 00:00:00
max       1250.000000         629.000000            2019-07-08 00:00:00
std         20.510550          44.550582                            NaN

       reviews_per_month  calculated_host_listings_count  availability_365
count       48895.000000                    48895.000000      48895.000000
mean            1.090910                        7.143982        112.781327
min             0.000000                        1.000000          0.000000
25%             0.040000                        1.000000          0.000000
50%             0.370000                        1.000000         45.000000
75%             1.580000                        2.000000        227.000000
max            58.500000                      327.000000        365.000000
std             1.597283                       32.952519        131.622289
```

Step 9: Visualization

Distribution of Prices

Plot the distribution of listing prices.

In [21]:
```
plt.figure(figsize=(10, 6))
sns.histplot(df['price'], bins=50, kde=True)
plt.title('Distribution of Listing Prices')
plt.xlabel('Price ($)')
plt.ylabel('Frequency')
plt.show()
```

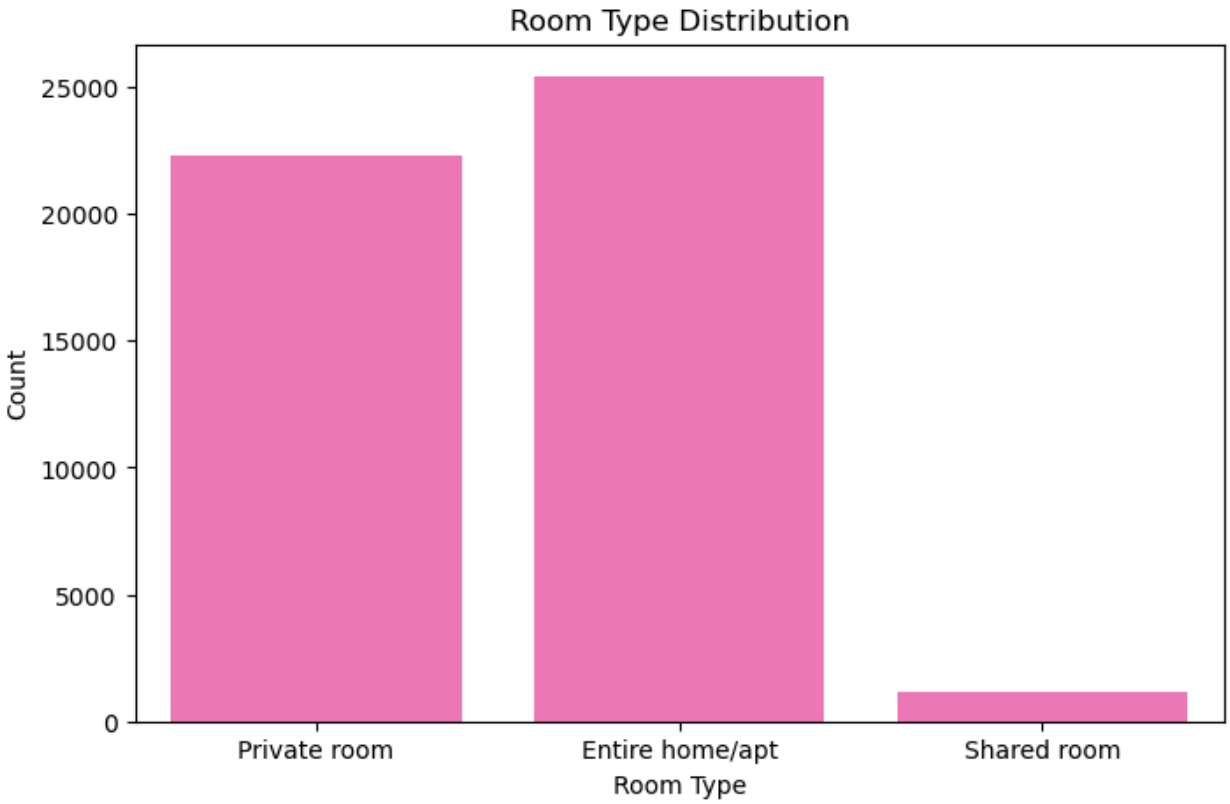


The histogram shows a fairly even distribution of listing prices across different price ranges, indicating no particular concentration of listings in any specific price range. The KDE line helps visualize this even spread more clearly, confirming that the dataset contains listings with a wide variety of prices.

Room Type Analysis

Analyze the distribution of different room types.

In [22]:
```
plt.figure(figsize=(8, 5))
sns.countplot(x='room_type', data=df , color='hotpink')
plt.title('Room Type Distribution')
plt.xlabel('Room Type')
plt.ylabel('Count')
plt.show()
```

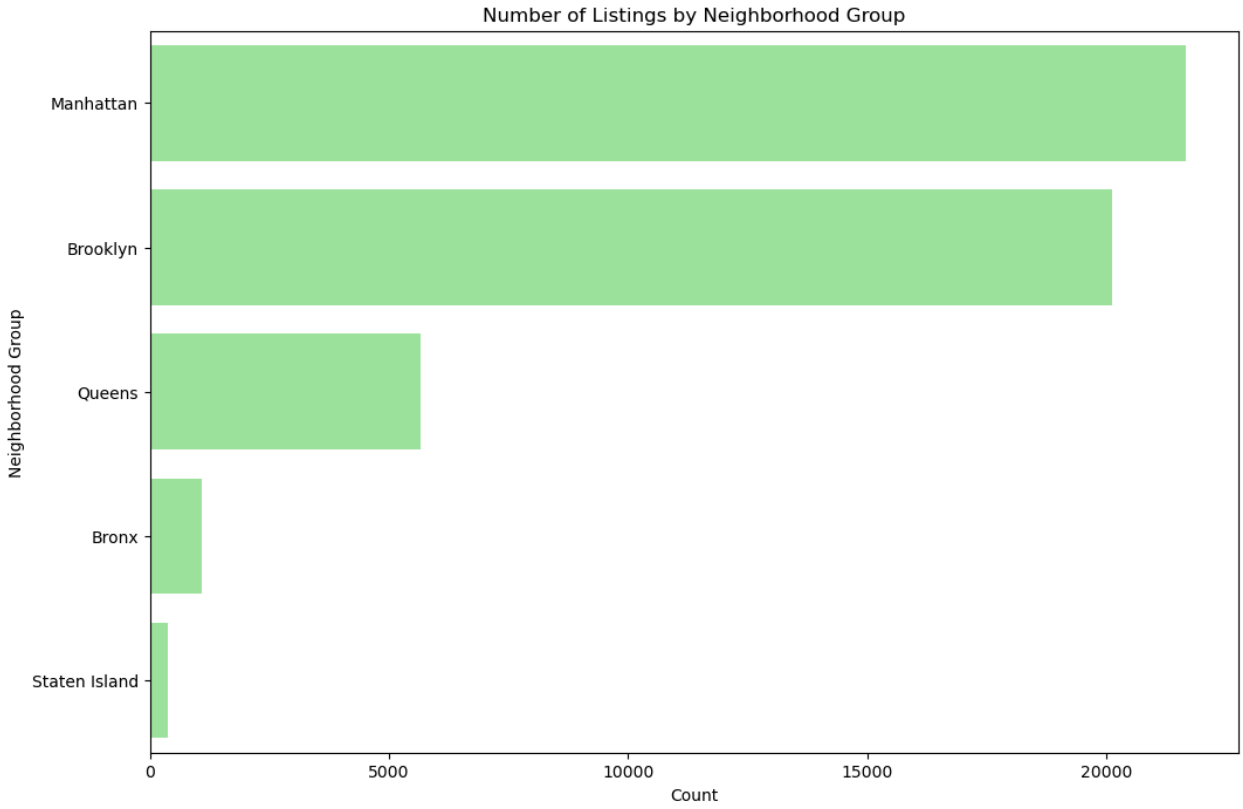


The count plot shows a clear distribution of the different room types available in the Airbnb dataset. The majority of listings are for 'Entire home/apt' and 'Private room', with 'Shared room' and 'Hotel room' being much less common. This insight can be useful for understanding the availability and popularity of different types of accommodations on Airbnb.

Neighborhood Analysis

Examine how listings are distributed across different neighborhoods.

In [23]:
```
plt.figure(figsize=(12, 8))
sns.countplot(y='neighbourhood_group', data=df,color="lightgreen" , order=df['
plt.title('Number of Listings by Neighborhood Group')
plt.xlabel('Count')
plt.ylabel('Neighborhood Group')
plt.show()
```



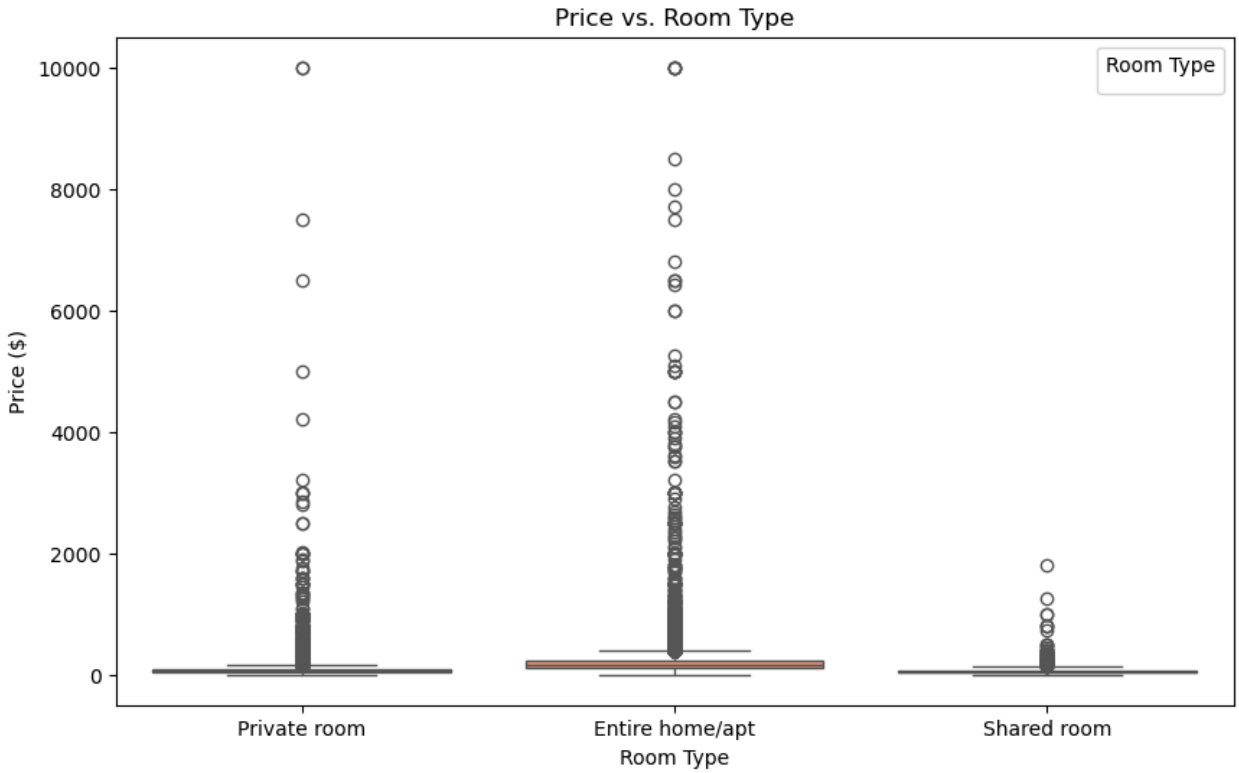
The count plot shows a clear distribution of the number of listings across different neighborhood groups. Manhattan and Brooklyn dominate the listings, suggesting they are prime locations for Airbnb. Queens, Bronx, and Staten Island have fewer listings, indicating less availability or popularity.

Price vs. Room Type

Visualize the relationship between price and room type.

In [24]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='room_type', y='price', hue='room_type', data=df, palette='Set2'
plt.title('Price vs. Room Type')
plt.xlabel('Room Type')
plt.ylabel('Price ($)')
plt.legend(title='Room Type')
plt.show()
```

```
C:\Users\subha\AppData\Local\Temp\ipykernel_17640\2734788128.py:6: UserWarning:
No artists with labels found to put in legend.  Note that artists whose label s
tart with an underscore are ignored when legend() is called with no argument.
  plt.legend(title='Room Type')
```

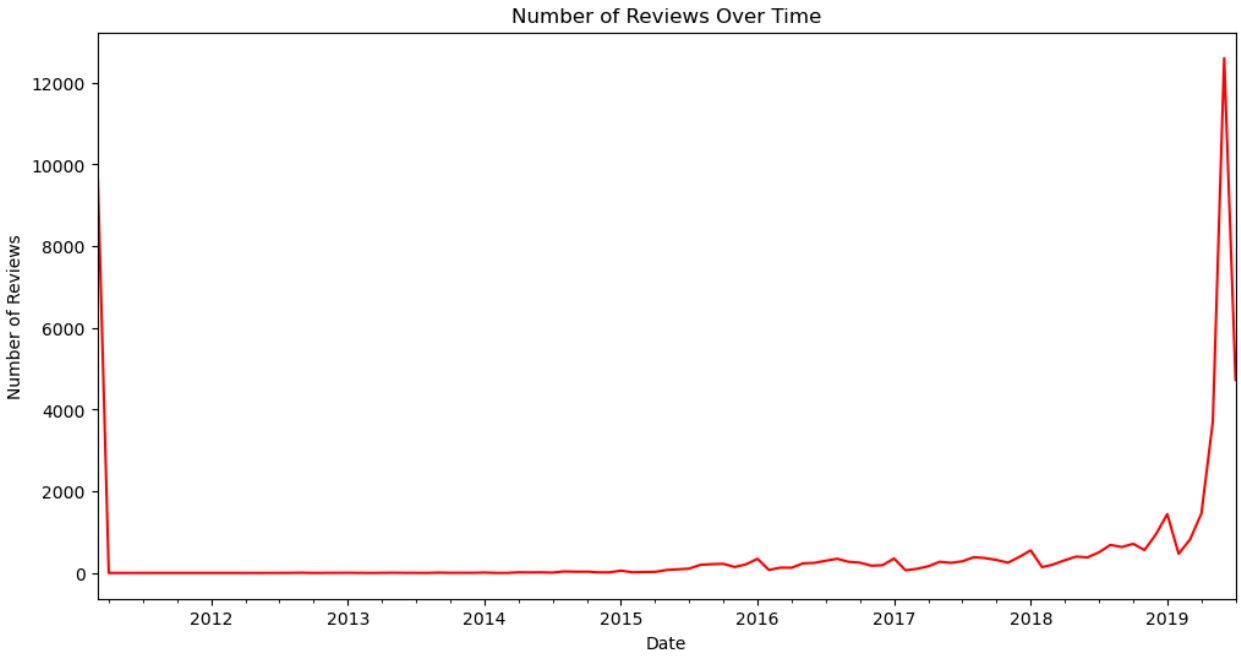


The box plot provides a detailed view of how prices vary across different room types in the Airbnb dataset. It shows that while 'Shared room' tends to have lower prices, 'Private room', 'Entire home/apt', and 'Hotel room' have higher and more varied price ranges. This visualization helps in understanding the pricing dynamics for different types of accommodations on Airbnb.

Reviews Over Time

Plot the number of reviews over time.

In [27]:
```
df['last_review'] = pd.to_datetime(df['last_review'])
reviews_over_time = df.groupby(df['last_review'].dt.to_period('M')).size()

plt.figure(figsize=(12, 6))
reviews_over_time.plot(kind='line',color='red')
plt.title('Number of Reviews Over Time')
plt.xlabel('Date')
plt.ylabel('Number of Reviews')
plt.show()
```



The line plot provides a clear visualization of the number of reviews over time. It helps identify trends and patterns in review activity, such as periods of high or low activity. This information can be useful for understanding the dynamics of user engagement and the popularity of Airbnb listings over time. The significant spikes and drops in reviews might be worth further investigation to understand the underlying causes, such as changes in Airbnb policies, market conditions, or external events.

In [ ]: