About this Dataset

This Airbnb dataset consists of information collected from 2011-2023 about House and Apartment owners in New York City who rent their properties to guests to stay for short period of time.

- **id :** ID of hosted houses or apartments.
- **name :** Names of the Hosted house or apartments.
- **host_id:** ID of the Host.
- **host_name :** Name of the Host.
- **neighbourhood_group:** Boroughs of NYC.
- **neighbourhood:** Neighourhoods in the Boroughs of NYC.
- **latitude:** Latitude.
- **longitude:** Longitude.
- **room_type:** Type of hosted Houses.
- **price:** Price of the hosted Houses.
- **minimum_nights:** Minimum number of nights spent at the Hosted Homes.
- **number_of_reviews :** Total number of reviews.
- **last_review:** Last date of the review posted.
- **reviews_per_month:** number of reviews per month
- **calculated_host_listings_count:** Number of accommodations hosted by the Host.
- **availability_365:** number of days.
- **number_of_reviews_ltm:** Number of reviews in the last n months.
- **license:** Accommodation License.
  'Special': Only one person has a license.

Import Libraries

In [1]:
```
# For data cleaning
import pandas as pd
import numpy as np

# For data visualization
import seaborn as sns
import matplotlib.pyplot as plt

# For visualizing NaN values
import missingno as msno

# For wordcloud generation
from wordcloud import WordCloud, ImageColorGenerator
from PIL import Image
import plotly.express as px
```

In [2]:
```
# Importing the Data
df = pd.read_csv(r"C:\Users\admin\Desktop\AI ML\Data analytics Projects\Airbnb\NY(
df.head()
```

Out[2]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longit
0	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75356	-73.98
1	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40.68535	-73.95
2	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Upper West Side	40.80380	-73.96
3	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Midtown	40.76457	-73.98
4	5136	Large Sunny Brooklyn Duplex, Patio + Garden	7378	Rebecca	Brooklyn	Sunset Park	40.66265	-73.99

In [3]:
```
# Shape of data
df.shape
```

Out[3]:
```
(42931, 18)
```

The tuple returned **(42931, 18)**. This means the DataFrame has 42931 rows and 18 columns.

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42931 entries, 0 to 42930
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              42931 non-null  int64  
 1   name                            42919 non-null  object 
 2   host_id                         42931 non-null  int64  
 3   host_name                       42926 non-null  object 
 4   neighbourhood_group             42931 non-null  object 
 5   neighbourhood                   42931 non-null  object 
 6   latitude                        42931 non-null  float64
 7   longitude                       42931 non-null  float64
 8   room_type                       42931 non-null  object 
 9   price                           42931 non-null  int64  
 10  minimum_nights                  42931 non-null  int64  
 11  number_of_reviews               42931 non-null  int64  
 12  last_review                     32627 non-null  object 
 13  reviews_per_month               32627 non-null  float64
 14  calculated_host_listings_count  42931 non-null  int64  
 15  availability_365                42931 non-null  int64  
 16  number_of_reviews_ltm           42931 non-null  int64  
 17  license                         1 non-null      object 
dtypes: float64(3), int64(8), object(7)
memory usage: 5.9+ MB
```

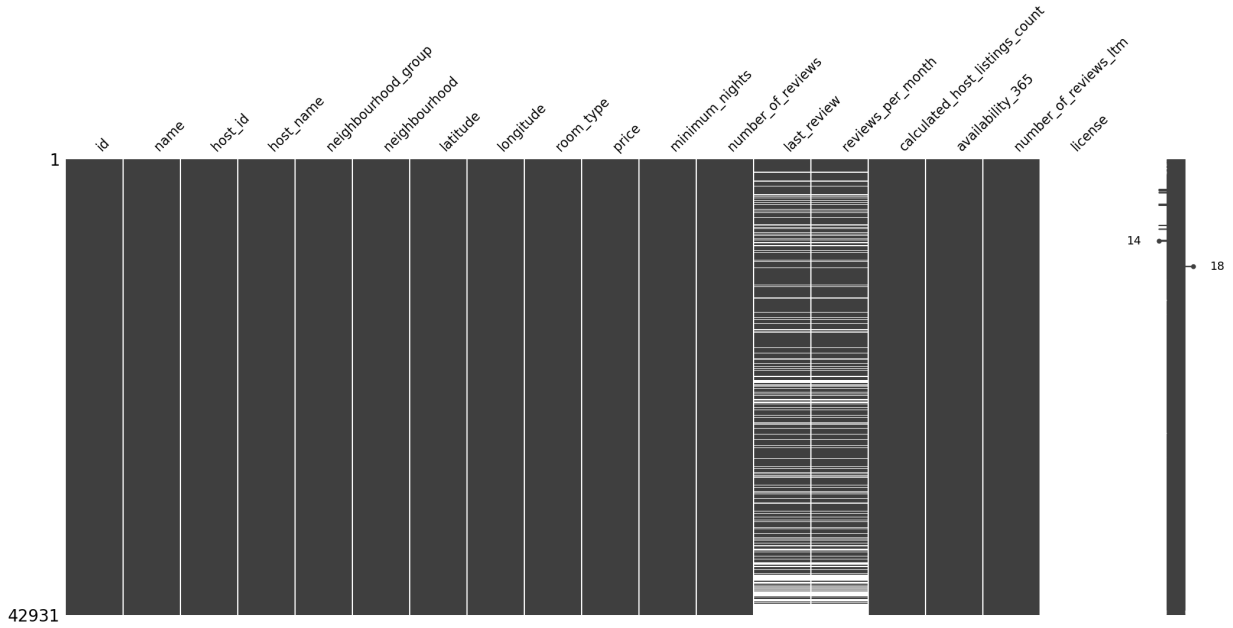
Data Cleaning

1. Handling NaN values

- NaN also known as Not a Number represents missing values in the data that cannot be converted into any datatype other than float.
- When no information is provided for one or more features or for the entire unit, this is referred to as missing data.
- Missing data poses a serious issue in real-world situations.Many datasets have missing data when they are imported into DataFrame, either because the data was never gathered or because it was present but was not captured.
- NaN values can impact various data operations, such as arithmetic computations, statistical analysis, and machine learning algorithms, skew results and introduce bias in statistical measures like means, medians, and correlations.
- There are many methods for dealing with NaN values. One such way is to replace NaN values with mode as it represents the most occuring value, making sure the integrity and reliability of data.

In [5]:
```
msno.matrix(df)
```

Out[5]: <Axes: >



In [6]:
```
df.isna().sum()
```

Out[6]:
```
id                                  0
name                               12
host_id                             0
host_name                           5
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     10304
reviews_per_month               10304
calculated_host_listings_count      0
availability_365                    0
number_of_reviews_ltm               0
license                         42930
dtype: int64
```

- **We can see that Maximum amount of NaN values present in the License Column, so that column will be dropped.**

In [7]:
```
# ALso dropping host_id and id because these are not needed

drop_col = ['host_id','id','license']
df = df.drop(columns = drop_col)
df.head()
```

Out[7]:

	name	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type
0	Skylit Midtown Castle	Jennifer	Manhattan	Midtown	40.75356	-73.98559	Entire home/apt
1	BlissArtsSpace!	Garon	Brooklyn	Bedford-Stuyvesant	40.68535	-73.95512	Private room
2	Cozy Clean Guest Room - Family Apt	MaryEllen	Manhattan	Upper West Side	40.80380	-73.96751	Private room
3	Large Furnished Room Near B'way	Shunichi	Manhattan	Midtown	40.76457	-73.98317	Private room
4	Large Sunny Brooklyn Duplex, Patio + Garden	Rebecca	Brooklyn	Sunset Park	40.66265	-73.99454	Entire home/apt

In [8]:
```
# Filling the NaN values with mode
df.reviews_per_month.mode()
```

Out[8]:
```
0    0.02
Name: reviews_per_month, dtype: float64
```

In [9]:
```
df.reviews_per_month = df.reviews_per_month.fillna(0.02)
```

In [10]:
```
df.isna().sum()

# Keeping last_review for further analysis
```

Out[10]:
```
name                                12
host_name                            5
neighbourhood_group                  0
neighbourhood                        0
latitude                             0
longitude                            0
room_type                            0
price                                0
minimum_nights                       0
number_of_reviews                    0
last_review                      10304
reviews_per_month                    0
calculated_host_listings_count       0
availability_365                     0
number_of_reviews_ltm                0
dtype: int64
```

2. Handling Duplicate Values

In [11]:
```
df.duplicated().any()
```

Out[11]:
```
True
```

In [12]:
```
# Checking the number of dupliate rows
df[df.duplicated()].shape
```

Out[12]:
```
(10, 15)
```

In [13]:
```
df = df.drop_duplicates()
```

In [14]:
```
df.shape
```

Out[14]:
```
(42921, 15)
```

3. Handling the Datatypes

In [15]:
```
df.dtypes
```

Out[15]:
```
name                                object
host_name                           object
neighbourhood_group                 object
neighbourhood                       object
latitude                           float64
longitude                          float64
room_type                           object
price                                int64
minimum_nights                       int64
number_of_reviews                    int64
last_review                         object
reviews_per_month                  float64
calculated_host_listings_count       int64
availability_365                     int64
number_of_reviews_ltm                int64
dtype: object
```

In [16]:
```
df['last_review'].unique()
```

Out[16]:
```
array(['2022-06-21', '2019-12-02', '2017-07-21', ..., '2022-02-23',
       '2022-03-10', '2022-03-18'], dtype=object)
```

In [17]:
```
# Converting object to datetime64[ns]
df['last_review'] = pd.to_datetime(df['last_review'])
```

In [18]:
```
# Creating the 'year' column for analysis
df['year'] = df['last_review'].dt.year
df['year'].unique()
```

Out[18]:
```
array([2022., 2019., 2017., 2023., 2021.,   nan, 2020., 2011., 2013.,
       2014., 2018., 2016., 2015., 2012.])
```

In [19]:
```
df.dtypes
# year in float because of NaN values
```

Out[19]:
```
name                                 object
host_name                            object
neighbourhood_group                  object
neighbourhood                        object
latitude                            float64
longitude                           float64
room_type                            object
price                                 int64
minimum_nights                        int64
number_of_reviews                     int64
last_review                  datetime64[ns]
reviews_per_month                   float64
calculated_host_listings_count        int64
availability_365                      int64
number_of_reviews_ltm                 int64
year                                float64
dtype: object
```

In [20]:
```
df.drop(columns = 'last_review',axis=1)
```

Out[20]:

	name	host_name	neighbourhood_group	neighbourhood	latitude	longitude	roo
0	Skylit Midtown Castle	Jennifer	Manhattan	Midtown	40.753560	-73.985590	ho
1	BlissArtsSpace!	Garon	Brooklyn	Bedford-Stuyvesant	40.685350	-73.955120	
2	Cozy Clean Guest Room - Family Apt	MaryEllen	Manhattan	Upper West Side	40.803800	-73.967510	
3	Large Furnished Room Near B'way	Shunichi	Manhattan	Midtown	40.764570	-73.983170	
4	Large Sunny Brooklyn Duplex, Patio + Garden	Rebecca	Brooklyn	Sunset Park	40.662650	-73.994540	ho
...	...	...	...	...	...	...	
42926	bright studio in Williamsburg	Jean	Brooklyn	Williamsburg	40.718976	-73.963985	ho
42927	Room in the heart of LES with Gym& Rooftop BBQ	Charlene	Manhattan	East Village	40.721703	-73.981473	
42928	Fantastic 3BD apt in Brooklyn	Jose	Brooklyn	Bushwick	40.688700	-73.907650	ho
42929	The Coziest Home	Remmy	Staten Island	Bull's Head	40.616911	-74.164652	ho
42930	378-2L-Red	Mikey	Brooklyn	Williamsburg	40.713091	-73.957205	

42921 rows × 15 columns

The **describe()** function is used to provide a set of descriptive statistics, which include measures such as the count, mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile), 75th percentile (Q3), and maximum.

In [21]:
```
df.describe()
```

Out[21]:

	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_
count	42921.000000	42921.000000	42921.000000	42921.000000	42921.000000	42921.0
mean	40.728267	-73.943658	200.189651	18.115165	25.862002	0.8
std	0.057646	0.056631	895.147544	27.464470	56.621574	1.6
min	40.500314	-74.251907	0.000000	1.000000	0.000000	0.0
25%	40.687480	-73.981750	75.000000	2.000000	1.000000	0.0
50%	40.724020	-73.952600	125.000000	7.000000	5.000000	0.2
75%	40.762298	-73.924020	200.000000	30.000000	24.000000	1.1
max	40.911380	-73.710870	99000.000000	1250.000000	1842.000000	86.6

Data Visualization

- Data Visualization refers to graphical representation of data with the help of bar charts, pie charts, line graphs, heatmaps, geographical maps etc.

Host Listings Count by Room Types and Neighbourhood Groups

In [22]:
```
room_list = df.groupby("room_type")["calculated_host_listings_count"].sum()
df_room_list = pd.DataFrame(room_list).reset_index()
column_name = ["room", "listings"]
df_room_list.columns = column_name
df_room_list
```

Out[22]:

	room	listings
0	Entire home/apt	555985
1	Hotel room	2809
2	Private room	471145
3	Shared room	2615

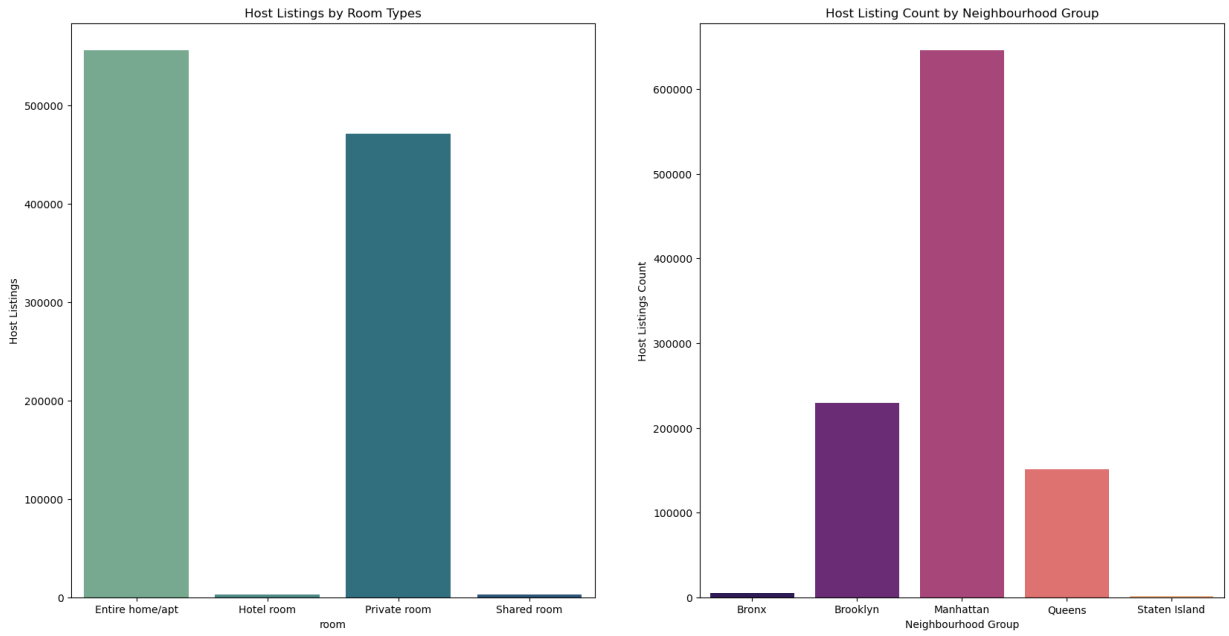
In [24]:
```
list_neighb = df.groupby("neighbourhood_group")["calculated_host_listings_count"]
df_neighb_list = pd.DataFrame(list_neighb).reset_index()
column_names_2 = ["group", "listings"]
df_neighb_list.columns = column_names_2
df_neighb_list
```

Out[24]:

	group	listings
0	Bronx	5103
1	Brooklyn	229209
2	Manhattan	645602
3	Queens	151472
4	Staten Island	1168

In [53]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 10))

sns.barplot(data = df_room_list,
            x = df_room_list['room'],
            y = df_room_list['listings'],
            ax = ax1, label = "Hosting Listings Count by Room Types",
            palette = "crest")

ax1.set_title("Host Listings by Room Types")
ax1.set_ylabel("Host Listings")

sns.barplot(data = df_neighb_list,
            x = df_neighb_list['group'],
            y = df_neighb_list['listings'],
            ax = ax2, label = "Host Listings Count by Neighbourhood groups",
            palette = "magma")

ax2.set_title("Host Listing Count by Neighbourhood Group")
ax2.set_xlabel("Neighbourhood Group")
ax2.set_ylabel("Host Listings Count")
```

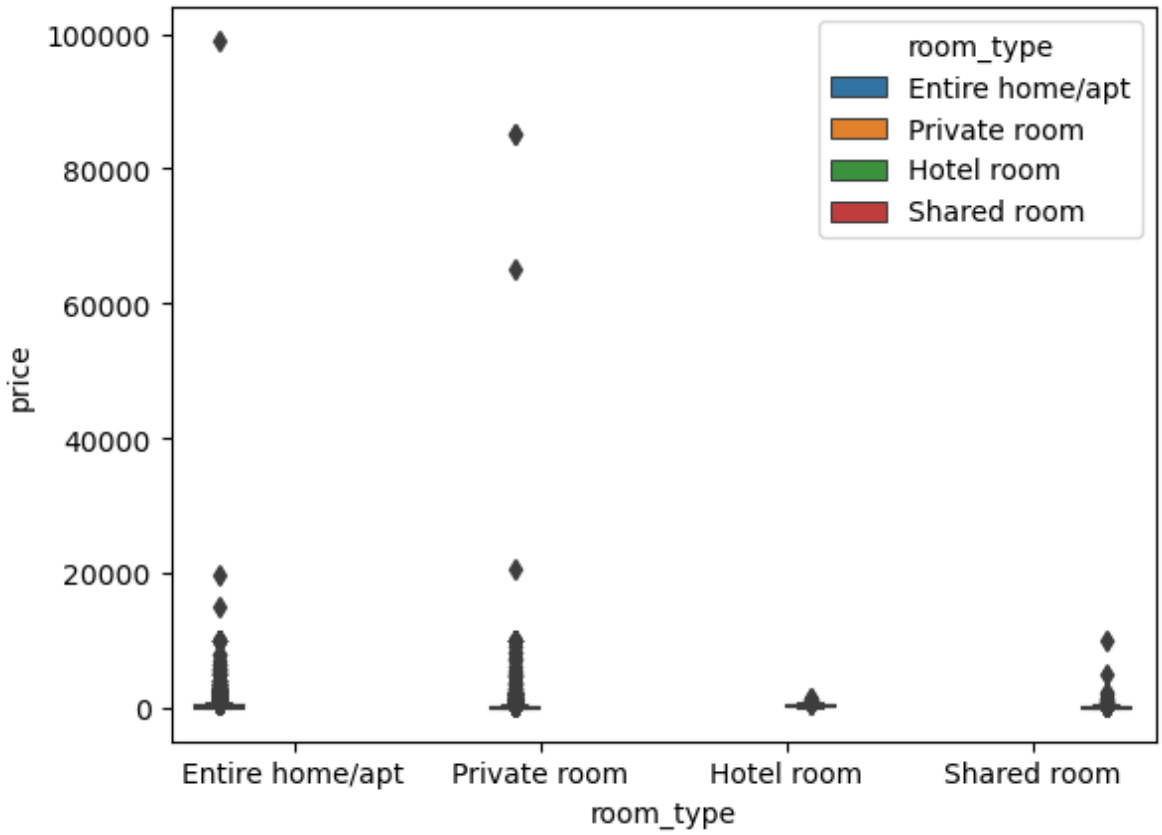
Out[53]: Text(0, 0.5, 'Host Listings Count')



Price Distribution by Room Type

In [26]:
```
sns.boxplot(data = df, x = df['room_type'], y = df['price'], hue = df['room_type'
```

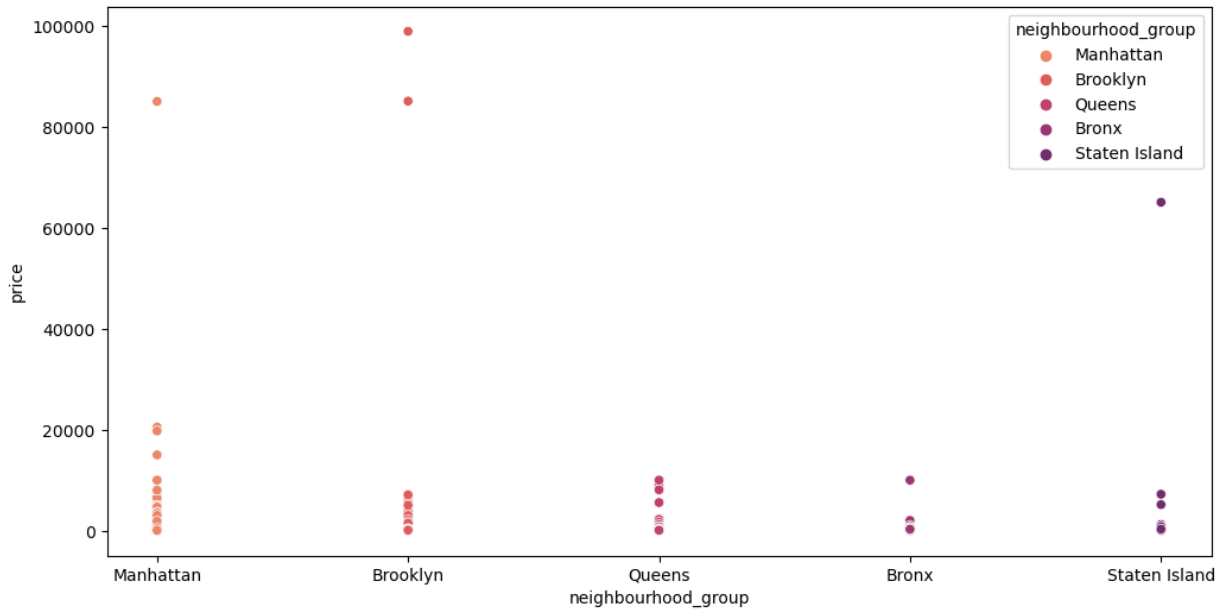
Out[26]:
```
<Axes: xlabel='room_type', ylabel='price'>
```



Price Distribution by Neighbourhood Groups

In [55]:
```
plt.figure(figsize = (12,6))
sns.scatterplot(data = df,
                x = df['neighbourhood_group'],
                y = df['price'],
                hue = df['neighbourhood_group'],
                palette = 'flare')
```

Out[55]:
```
<Axes: xlabel='neighbourhood_group', ylabel='price'>
```



HeatMap

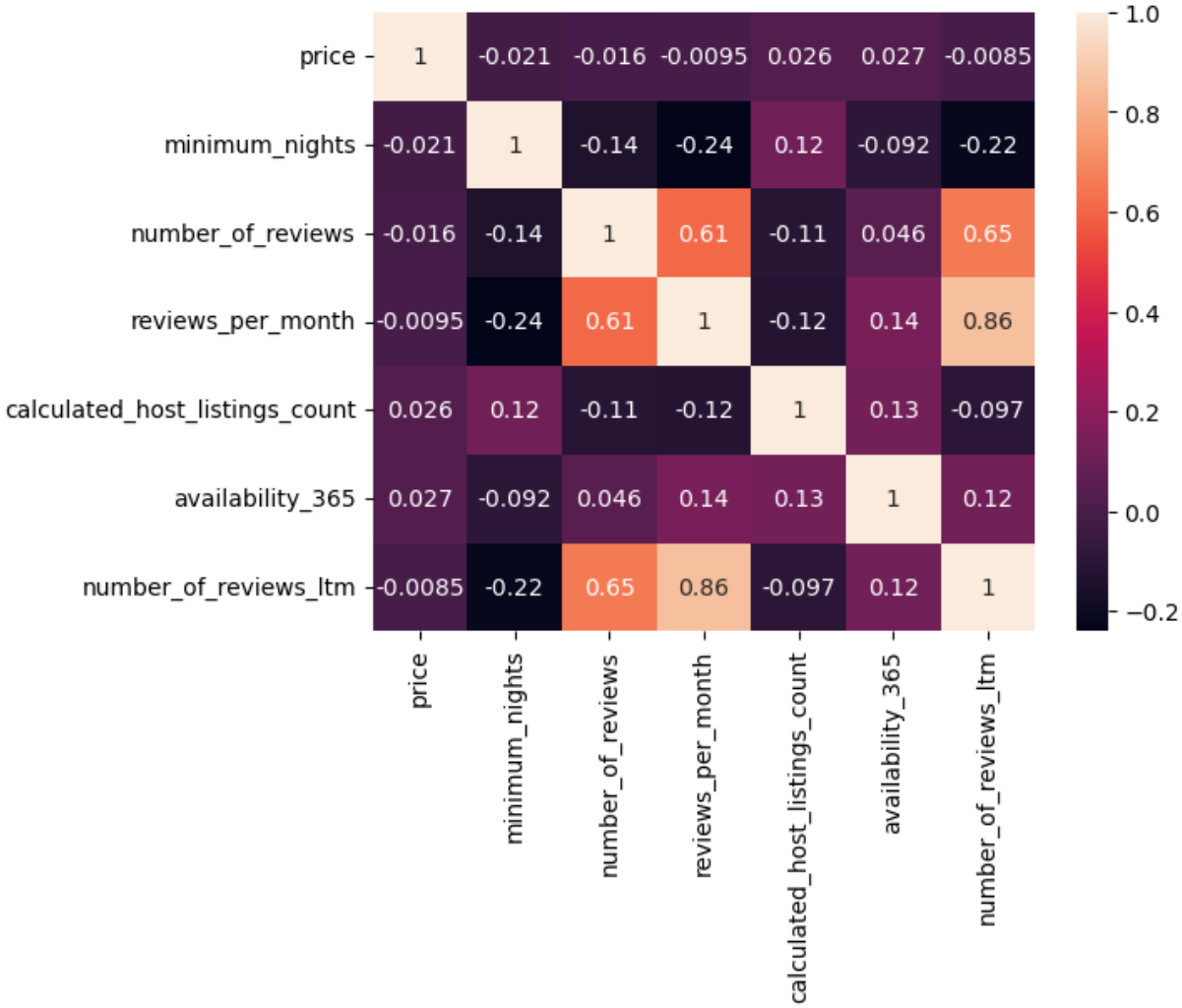
In [28]:
```
corr_table = df[['price','minimum_nights','number_of_reviews','reviews_per_month'
                 'availability_365','number_of_reviews_ltm']].corr()
corr_table
```

Out[28]:

	price	minimum_nights	number_of_reviews	reviews_per_month	c
price	1.000000	-0.020675	-0.016406	-0.009522	
minimum_nights	-0.020675	1.000000	-0.138867	-0.240618	
number_of_reviews	-0.016406	-0.138867	1.000000	0.610516	
reviews_per_month	-0.009522	-0.240618	0.610516	1.000000	
calculated_host_listings_count	0.026185	0.119949	-0.111158	-0.120731	
availability_365	0.027153	-0.092412	0.046174	0.138872	
number_of_reviews_ltm	-0.008535	-0.216611	0.652936	0.858727	

In [29]:
```
heat_map = sns.heatmap(data = corr_table, annot = True)
heat_map
```

Out[29]: <Axes: >



Geographical Map

In [30]:
```
location = df[['latitude','longitude','neighbourhood_group']]
location
```

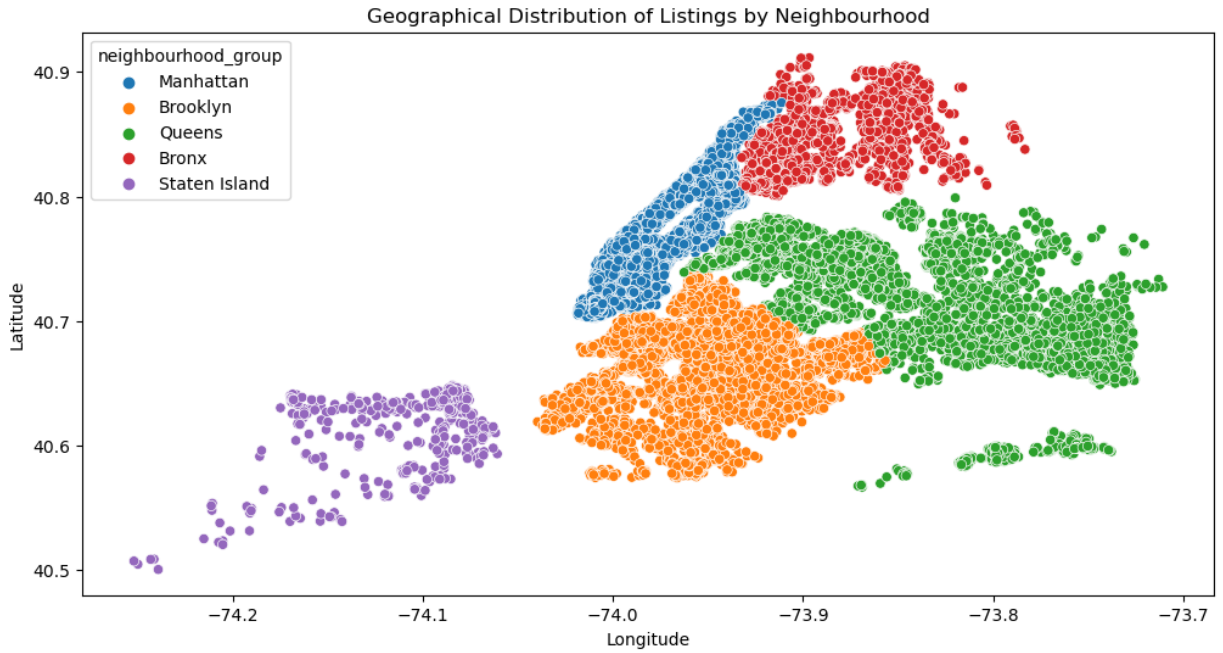
Out[30]:

	latitude	longitude	neighbourhood_group
0	40.753560	-73.985590	Manhattan
1	40.685350	-73.955120	Brooklyn
2	40.803800	-73.967510	Manhattan
3	40.764570	-73.983170	Manhattan
4	40.662650	-73.994540	Brooklyn
...	...	...	...
42926	40.718976	-73.963985	Brooklyn
42927	40.721703	-73.981473	Manhattan
42928	40.688700	-73.907650	Brooklyn
42929	40.616911	-74.164652	Staten Island
42930	40.713091	-73.957205	Brooklyn

42921 rows × 3 columns

In [31]:
```
plt.figure(figsize = (12,6))
geo_map = sns.scatterplot(location, x = "longitude", y = "latitude", hue = "neighl
                          color = 'viridis' )

plt.title("Geographical Distribution of Listings by Neighbourhood")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
```

Out[31]:
```
Text(0, 0.5, 'Latitude')
```



Mean Price Distribution by Neighbourhood Groups and Room Types

In [32]:
```
av_price_room = df.groupby("room_type")["price"].mean()
df_av_room = pd.DataFrame(av_price_room).reset_index()
column_names = ["room_type", "price"]
df_av_room.columns = column_names
df_av_room
```

Out[32]:

	room_type	price
0	Entire home/apt	249.255365
1	Hotel room	309.959391
2	Private room	134.696234
3	Shared room	126.250000

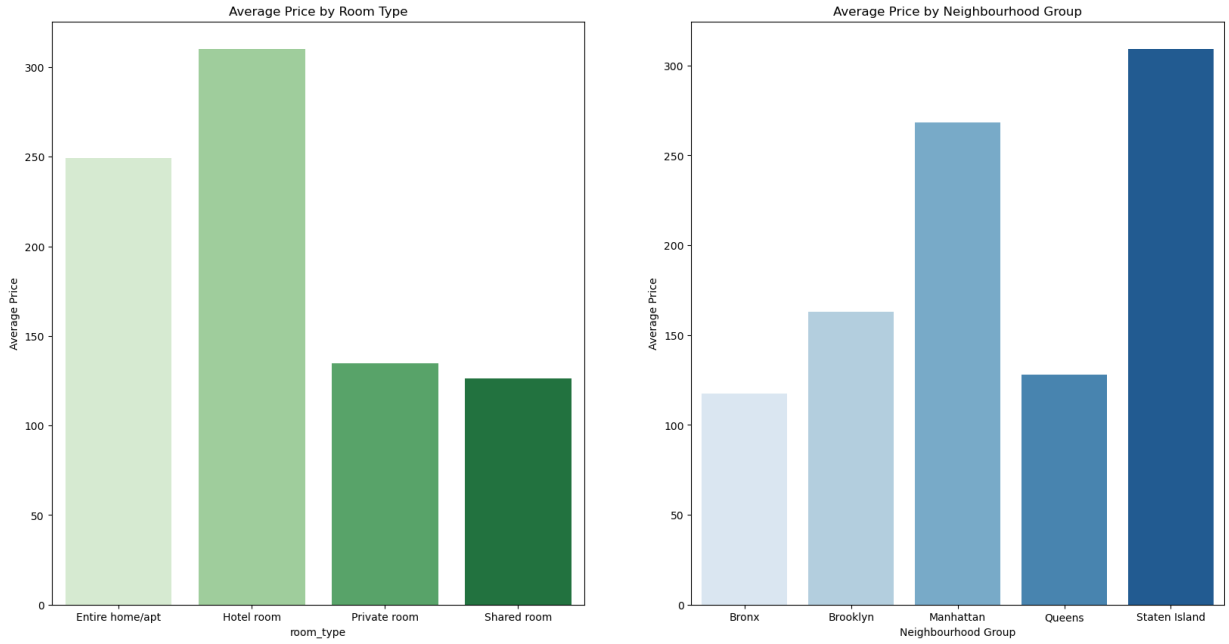
In [33]:
```
av_price_neighb = df.groupby("neighbourhood_group")["price"].mean()
df_av_neighb = pd.DataFrame(av_price_neighb).reset_index()
column_names = ["neighbourhood_group", "price"]
df_av_neighb.columns = column_names
df_av_neighb
```

Out[33]:

	neighbourhood_group	price
0	Bronx	117.512123
1	Brooklyn	162.766829
2	Manhattan	268.118540
3	Queens	128.173655
4	Staten Island	309.037296

In [34]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (20, 10))

sns.barplot(data = df_av_room, x = df_av_room["room_type"],
            y = df_av_room["price"],
            ax = ax1, palette="Greens",
            label="Average Price by Room Type")

ax1.set_title("Average Price by Room Type")
ax1.set_ylabel("Average Price")

sns.barplot(data = df_av_neighb, x = "neighbourhood_group", y = "price",
            ax = ax2, palette = "Blues",
            label = "Average Price by Neighbourhood Group")

ax2.set_title("Average Price by Neighbourhood Group")
ax2.set_xlabel("Neighbourhood Group")
ax2.set_ylabel("Average Price")
```

Out[34]:
```
Text(0, 0.5, 'Average Price')
```



Number of reviews by Room Type and Neighbourhood Locations

In [35]:
```
rev_room_type = df.groupby("room_type")["number_of_reviews"].sum()
df_rev_room_type = pd.DataFrame(rev_room_type).reset_index()
column_names = ["room_type", "number_of_reviews"]
df_rev_room_type.columns = column_names
df_rev_room_type
```

Out[35]:

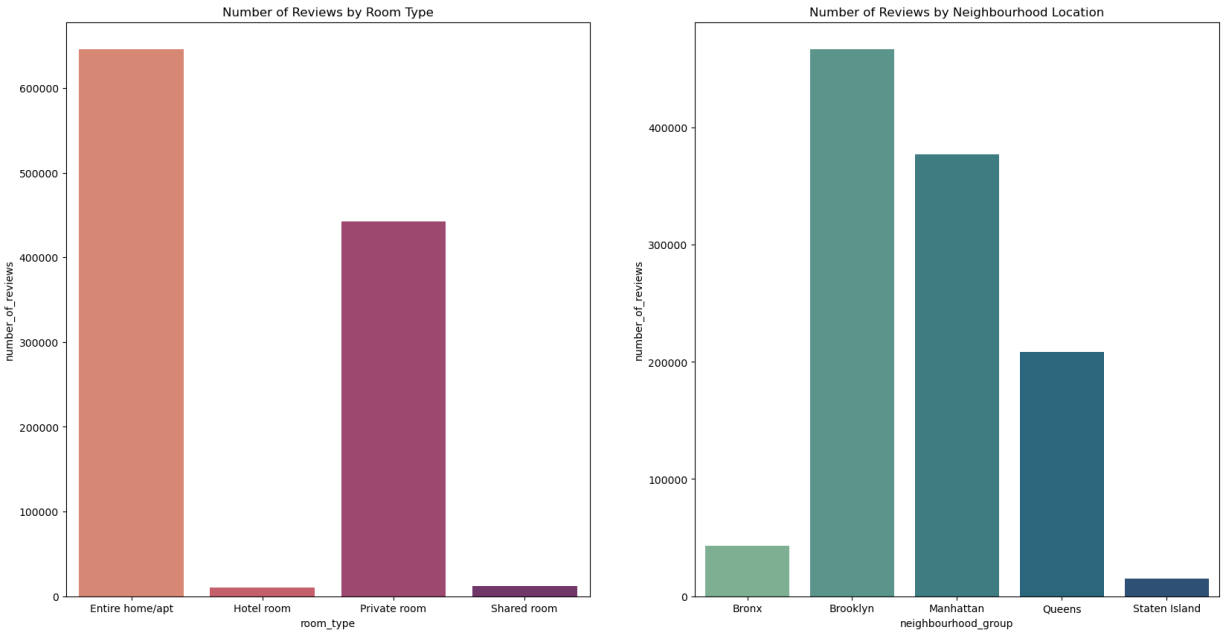
	room_type	number_of_reviews
0	Entire home/apt	645605
1	Hotel room	10292
2	Private room	442186
3	Shared room	11940

In [36]:
```
rev_neighb = df.groupby("neighbourhood_group")["number_of_reviews"].sum()
df_rev_neighb = pd.DataFrame(rev_neighb).reset_index()
column_names = ["neighbourhood_group", "number_of_reviews"]
rev_neighb.columns = column_names
df_rev_neighb
```

Out[36]:

	neighbourhood_group	number_of_reviews
0	Bronx	43047
1	Brooklyn	466643
2	Manhattan	376780
3	Queens	208344
4	Staten Island	15209

In [37]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
sns.barplot(data=df_rev_room_type, x="room_type", y="number_of_reviews", ax=ax1, 
ax1.set_title("Number of Reviews by Room Type")

sns.barplot(data=df_rev_neighb, x="neighbourhood_group", y="number_of_reviews", a
ax2.set_title("Number of Reviews by Neighbourhood Location")

plt.show()
```



Availability of Rooms

In [38]:
```
room_365 = df.groupby("room_type")["availability_365"].sum()
df_room_365 = pd.DataFrame(room_365).reset_index()
column_names_3 = ["room_type", "availability"]
df_room_365.columns = column_names_3
df_room_365
```

Out[38]:

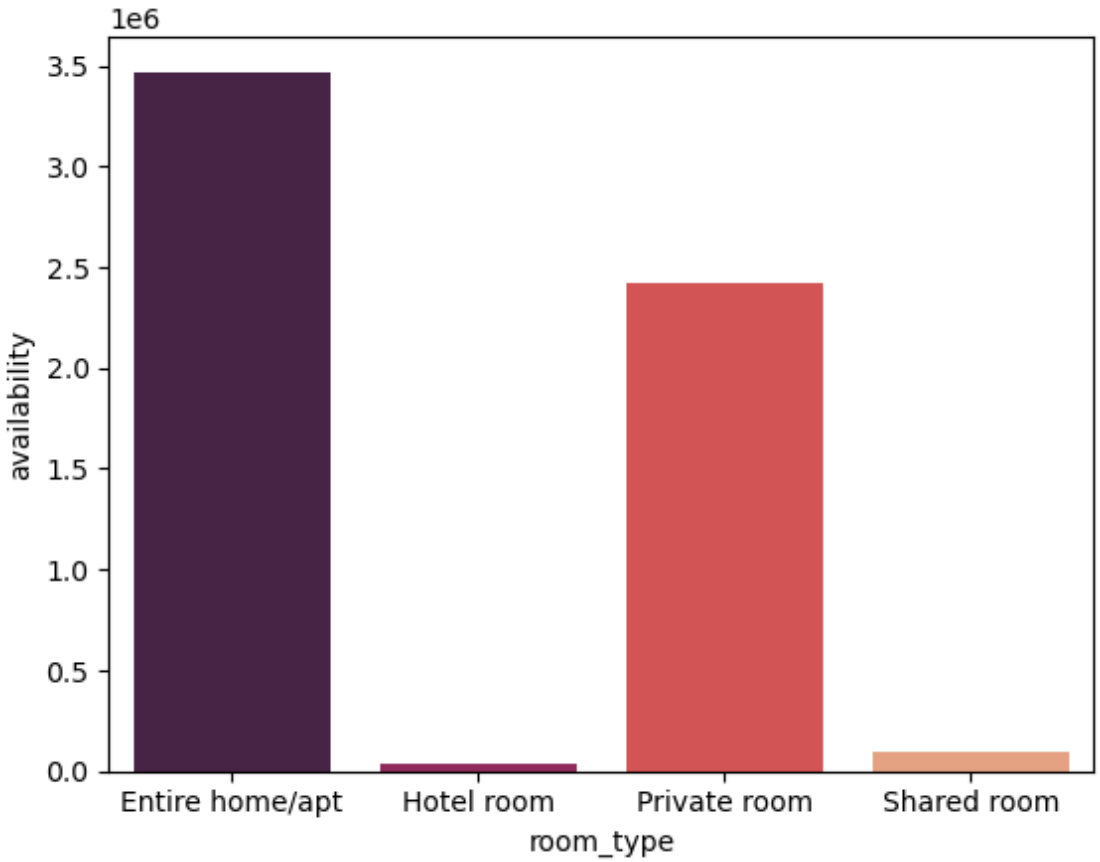
	room_type	availability
0	Entire home/apt	3468510
1	Hotel room	40942
2	Private room	2417111
3	Shared room	93359

In [58]:
```
df
```

Out[58]:

.type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listi
:ntire e/apt	150	30	49	2022-06-21	0.30	
ivate room	60	30	50	2019-12-02	0.30	
ivate room	75	2	118	2017-07-21	0.72	
ivate room	68	2	575	2023-02-19	3.41	
:ntire e/apt	275	60	3	2022-08-10	0.03	
...	...	...	...	...	...	
:ntire e/apt	76	7	0	NaT	0.02	
ivate room	32	30	0	NaT	0.02	
:ntire e/apt	127	3	0	NaT	0.02	
:ntire e/apt	280	1	0	NaT	0.02	
ivate room	78	90	0	NaT	0.02	

In [39]:
```
bar_room_365 = sns.barplot(data = df_room_365,
                           x = df_room_365['room_type'],
                           y = df_room_365['availability'],
                          palette = "rocket")
```



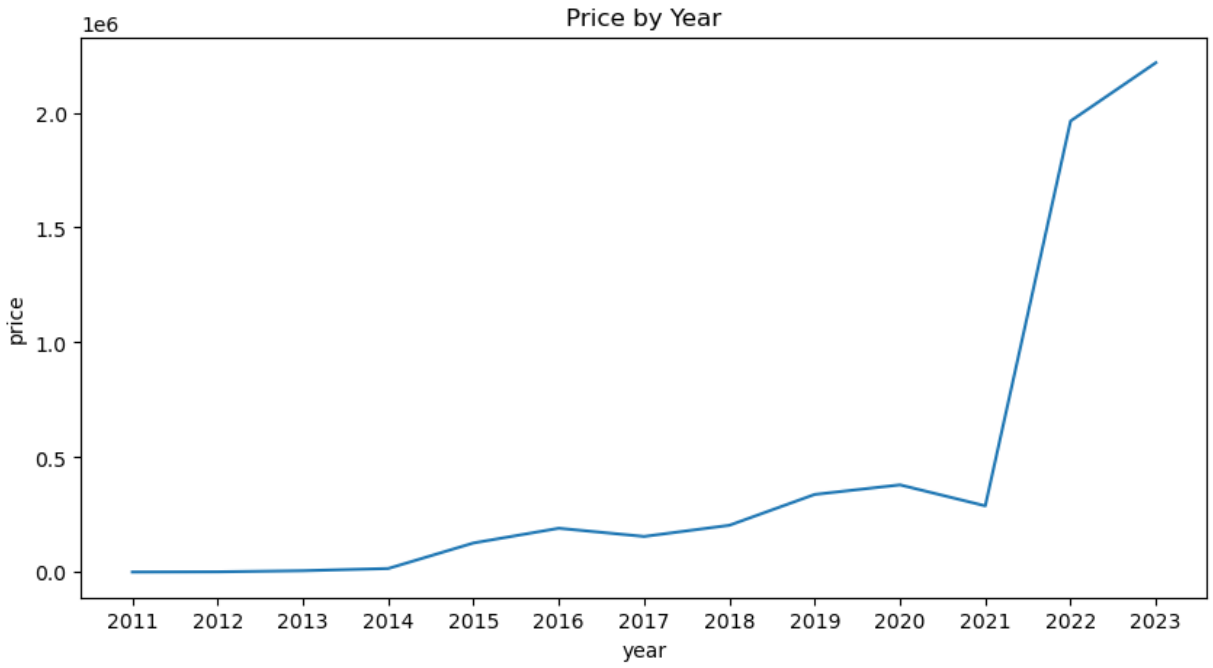
Price by Year

In [40]:
```
price_year = df.groupby("year")["price"].sum()
df_price_year = pd.DataFrame(price_year).reset_index()
column_names = ["year", "price"]
df_price_year.columns = column_names
df_price_year
```

Out[40]:

	year	price
0	2011.0	844
1	2012.0	1696
2	2013.0	6430
3	2014.0	16068
4	2015.0	127204
5	2016.0	191224
6	2017.0	155581
7	2018.0	204155
8	2019.0	338396
9	2020.0	380013
10	2021.0	288765
11	2022.0	1963554
12	2023.0	2217188

In [41]:
```
plt.figure(figsize=(10, 5))

ax = sns.lineplot(data=df_price_year, x="year", y="price")
ax.set_xticks(list(range(2011, 2024)))

plt.rcParams["figure.figsize"] = (10, 4)

plt.title("Price by Year")
plt.show()
```



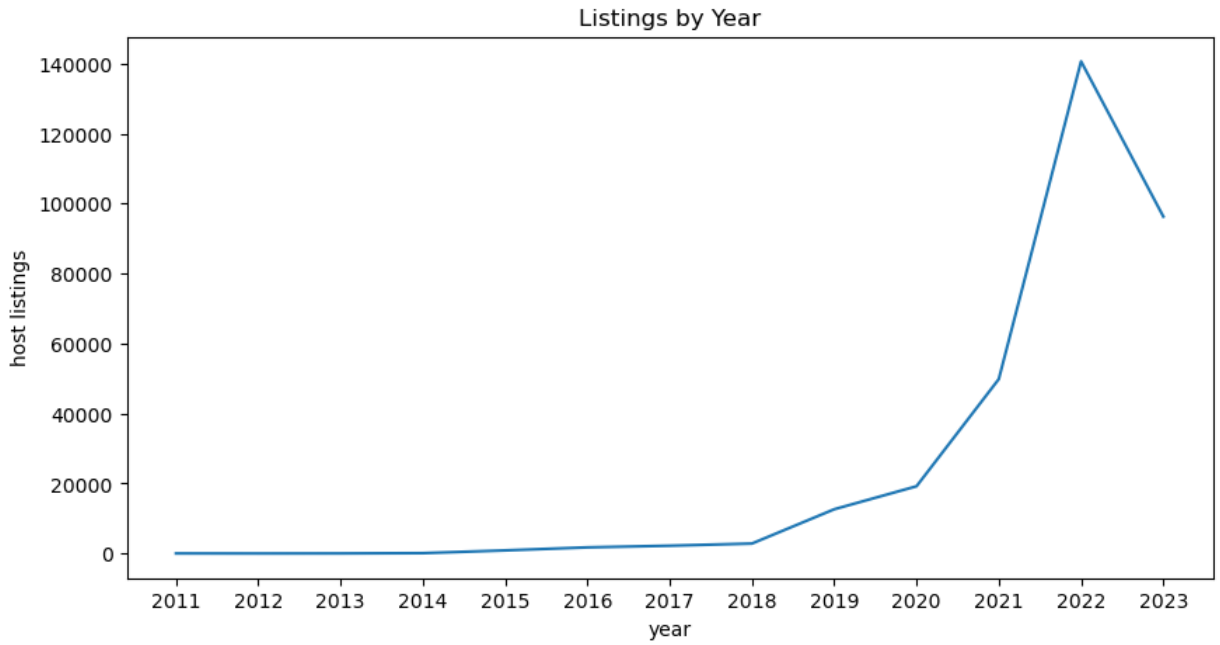
Host Listings by Year

In [42]:
```
host_year = df.groupby("year")["calculated_host_listings_count"].sum()
df_host_year = pd.DataFrame(host_year).reset_index()
column_names = ["year", "host listings"]
df_host_year.columns = column_names
df_host_year
```

Out[42]:

	year	host listings
0	2011.0	32
1	2012.0	16
2	2013.0	33
3	2014.0	112
4	2015.0	898
5	2016.0	1764
6	2017.0	2238
7	2018.0	2855
8	2019.0	12653
9	2020.0	19224
10	2021.0	49855
11	2022.0	140638
12	2023.0	96291

In [43]:
```
plt.figure(figsize=(10, 5))

ax = sns.lineplot(data=df_host_year, x="year", y="host listings")
ax.set_xticks(list(range(2011, 2024)))

plt.title("Listings by Year")
plt.show()
```



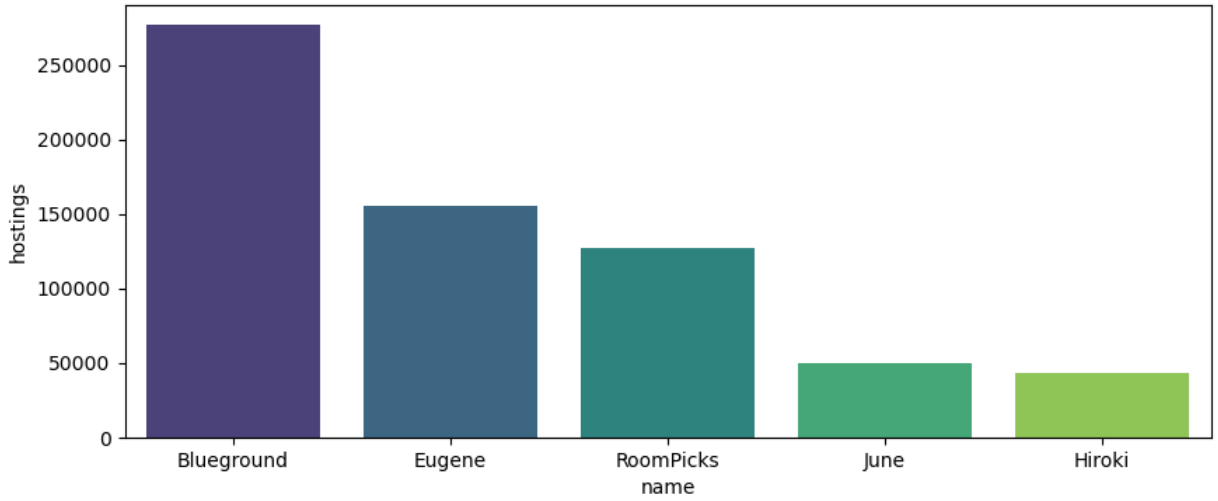
Top Five Listings by Host names

In [44]:
```
host_list = df.groupby("host_name")["calculated_host_listings_count"].sum().sort_
df_host_list = pd.DataFrame(host_list).reset_index()
column_names_4 = ["name", "hostings"]
df_host_list.columns = column_names_4
df_host_list
```

Out[44]:

	name	hostings
0	Blueground	276676
1	Eugene	155242
2	RoomPicks	127313
3	June	49300
4	Hiroki	42849

In [45]:
```
bar_host_list = sns.barplot(data = df_host_list,
                            x = df_host_list['name'],
                            y = df_host_list['hostings'],
                           palette = "viridis")
```



WordCloud for Host Names

In [47]:
```
names = ' '.join(df['host_name'].astype(str))
```

In [48]:
```
# WordCloud() is a class used to generate word clouds
# generate() takes list of words as input to generate the word cloud
wordcloud = WordCloud(width = 800, height = 400, background_color ='black').gener
```

In [49]:
```
# imshow() is used to display the wordcloud
fig = px.imshow(wordcloud.to_array())
fig.update_layout(title_text='Word Cloud for host names')
# showticklabels specify whether tick labels should be shown or not
fig.update_xaxes(showticklabels = False)
fig.update_yaxes(showticklabels = False)
fig.show()
```

Word Cloud for host names

In [50]:
```
mask_image = np.array(Image.open(r"C:\Users\admin\Desktop\AI ML\Data analytics Pr
# Creating a wordcloud object using the mask_image
wordcloud = WordCloud(mask = mask_image, background_color ='black').generate(name
# ImageColorGenerator() is used to map words in wordcloud to colors of the mask i
image_colors = ImageColorGenerator(mask_image)
# Converting wordcloud to array
wordcloud_image = wordcloud.to_array()

fig = px.imshow(wordcloud_image)
fig.update_layout(title_text ='Word Cloud with Mask Image')
fig.update_xaxes(showticklabels = False)
fig.update_yaxes(showticklabels = False)
fig.show()
```

In [ ]:
```

```