Capstone Project Creation

IBM SkillsBuild Europe Delivery - Data Analytics

Pre-requisite

- Understanding of Python, Power BI or Tableau
- Understanding of Data Cleaning
- Understanding Data Visualization

Data Analytics of Airbnb Data:

Objective:

In this exericise, you will be performing Data Analytics on an Open Dataset dataset coming from Airbnb. Some of the tasks include

- Data Cleaning.
- Data Transformation
- Data Visualization.

Overview of Airbnb Data:

People's main criteria when visiting new places are reasonable accommodation and food. Airbnb (Air-Bed-Breakfast) is an online marketplace created to meet this need of people by renting out their homes for a short term. They offer this facility at a relatively lower price than hotels. Further people worldwide prefer the homely and economical service offered by them. They offer services across various geographical locations

Dataset Source

You can get the dataset for this assessment using the following link: https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata (https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata)

This dataset contains information such as the neighborhood offering these services, room type, price,avaliabilty, reviews, service fee, cancellation policy and rules to use the house. This analysis will help airbnb in improving its services.

In [1]:
```
### PLEASE NOTE:
# All comments with Two hashtags (##) where added by the Supervisor.
# All comments with One hashtag (#) were added by me.
```

Task 1: Data Loading (Python)

1. Read the csv file and load it into a pandas dataframe.
2. Display the first five rows of your dataframe.
3. Display the data types of the columns.

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import warnings
from warnings import filterwarnings
filterwarnings("ignore")
```

In [3]:
```
## Read the csv file


df = pd.read_csv(r"../IBM_Capstone_Project_2023/Dirty_Airbnb_Open_Data.csv")
```

In [4]:
```
# AS AN EXTRA BONUS: Setting this option will print all collumns of the dataframe


pd.set_option("display.max.columns", None)
```

In [5]:
```
## Display the first 5 rows

df.head()
```

Out[5]:

	id	NAME	host id	host_identity_verified	host name	neighbourhood group	neighbourhood	lat	long	country	country code
0	1001254	Clean & quiet apt home by the park	80014485718	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	-73.97237	United States	US
1	1002102	Skylit Midtown Castle	52335172823	verified	Jenna	Manhattan	Midtown	40.75362	-73.98377	United States	US
2	1002403	THE VILLAGE OF HARLEM....NEW YORK !	78829239556	NaN	Elise	Manhattan	Harlem	40.80902	-73.94190	United States	US
3	1002755	NaN	85098326012	unconfirmed	Garry	Brooklyn	Clinton Hill	40.68514	-73.95976	United States	US
4	1003689	Entire Apt: Spacious Studio/Loft by central park	92037596077	verified	Lyndon	Manhattan	East Harlem	40.79851	-73.94399	United States	US

In [6]:
```
## Display the data types of the columns


df.dtypes
```

Out[6]:
```
id                                int64
NAME                             object
host id                           int64
host_identity_verified           object
host name                        object
neighbourhood group              object
neighbourhood                    object
lat                             float64
long                            float64
country                          object
country code                     object
instant_bookable                 object
cancellation_policy              object
room type                        object
Construction year               float64
price                            object
service fee                      object
minimum nights                  float64
number of reviews               float64
last review                      object
reviews per month               float64
review rate number              float64
calculated host listings count  float64
availability 365                float64
house_rules                      object
license                          object
dtype: object
```

Task 2a: Data Cleaning (Any Tool)

1. Drop some of the unwanted columns. These include `host id` , `id` , `country` and `country code` from the dataset.
2. State the reason for not including these columns for your Data Analytics.

If using Python for this exercise, please include the code in the cells below. If using any other tool, please include screenshots before and after

In [7]:
```
## Drop some of the unwanted columns. These include host id, id, country and country code from the dataset.

df.drop(["id", "host id", "country", "country code"], axis="columns", inplace=True)



## State the reason for not including these columns for your Data Analytics.


# Number One reason for not including the host_id and id columns in my Data Analytics:

# These columns contain unique identifiers for hosts and listings. They are specific to
# individual hosts and listings and do not provide any meaningful information about the
# total number of listings available per year. Including these columns in the analysis would
# not contribute to understanding the overall trend or pattern of listings over time.


# Number Two reason for not including the country and country_code columns:

# These columns contain information about the country and country codes associated with the
# listings. While these columns might be useful for analyzing the geographical distribution of
# listings, they are not directly relevant to understanding the total amount of listings available
# per year. If the focus is solely on the number of listings over time, including these columns could
# introduce unnecessary complexity to the analysis without providing meaningful insights.
```

In [8]:
```
# AS AN EXTRA BONUS: I am viewing all the columns of my dataframe after droping those unwanted columns


df.columns
```

Out[8]:
```
Index(['NAME', 'host_identity_verified', 'host name', 'neighbourhood group',
       'neighbourhood', 'lat', 'long', 'instant_bookable',
       'cancellation_policy', 'room type', 'Construction year', 'price',
       'service fee', 'minimum nights', 'number of reviews', 'last review',
       'reviews per month', 'review rate number',
       'calculated host listings count', 'availability 365', 'house_rules',
       'license'],
      dtype='object')
```

Task 2b: Data Cleaning (Python)

- Check for missing values in the dataframe and display the count in ascending order. **If the values are missing, impute the values as per the datatype of the columns.**
- Check whether there are any duplicate values in the dataframe and, if present, remove them.
- Display the total number of records in the dataframe before and after removing the duplicates.

In [9]:
```
## Check for missing values in my dataframe and display the count in ascending order.


df.isnull().sum().sort_values(ascending=True)
```

Out[9]:
```
room type                           0
lat                                 8
long                                8
neighbourhood                      16
neighbourhood group                29
cancellation_policy                76
instant_bookable                  105
number of reviews                 183
Construction year                 214
price                             247
NAME                              250
service fee                       273
host_identity_verified            289
calculated host listings count    319
review rate number                326
host name                         406
minimum nights                    409
availability 365                  448
reviews per month               15879
last review                     15893
house_rules                     52131
license                        102597
dtype: int64
```

In [10]:
```
## If the values are missing, impute the values as per the datatype of the columns.


# Identifying columns with missing values
missing_values = df.isnull().sum()

# Imputing missing values for numerical columns using for loop
for column in missing_values[missing_values > 0].index:
    if df[column].dtype in [np.int64, np.float64]:
        df[column].fillna(df[column].mean(), inplace=True)


# Imputing missing values for categorical columns using for loop
for column in missing_values[missing_values > 0].index:
    if df[column].dtype == "object" or pd.api.types.is_categorical_dtype(df[column]):
        df[column].fillna(df[column].mode().iloc[0], inplace=True)
```

In [11]:
```
## Checking whether there are any duplicate values in the dataframe


duplicate_values = df.duplicated()
any_duplicates = duplicate_values.any()
print("Any duplicates exist in my dataset?:", any_duplicates)
```

```
Any duplicates exist in my dataset?: True
```

In [12]:
```
## Display the total number of records in my dataframe before removing the duplicates values.


df.shape[0]
```

Out[12]:
```
102599
```

In [13]:
```
## And if duplicate value is presect, remove them.


df = df.drop_duplicates()
```

In [14]:
```
## Display the total number of records in my dataframe after removing the duplicates values.


df.shape[0]
```

Out[14]: 99146

In [15]:
```
# AS AN EXTRA BONUS: I am again checking whether there are any duplicate values in the dataframe


duplicate_values = df.duplicated()
any_duplicates = duplicate_values.any()
print("Any duplicates exist in my dataset?:", any_duplicates)
```

```
Any duplicates exist in my dataset?: False
```

Task 3: Data Transformation (Any Tool)

- Rename the column `availability 365` to `days_booked`
- Convert all column names to lowercase and replace the spaces in the column names with an underscore "_".
- Remove the dollar sign and comma from the columns `price` and `service_fee` . If necessary, convert these two columns to the appropriate data type.

If using Python for this exercise, please include the code in the cells below. If using any other tool, please include screenshoots of your work.

In [16]:
```
## Rename the availability 365 column to days_booked column.


df.rename(columns={ "availability 365" : "days_booked" }, inplace=True)
```

In [17]:
```
## Converting all column names to lowercase and replace the spaces with an underscore "_"


df.columns = df.columns.str.lower()
df.columns = df.columns.str.strip().str.replace(' ', '_')
df.head(4)
```

Out[17]:

	name	host_identity_verified	host_name	neighbourhood_group	neighbourhood	lat	long	instant_bookable	cancellation_po
0	Clean & quiet apt home by the park	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	-73.97237	False	s
1	Skylit Midtown Castle	verified	Jenna	Manhattan	Midtown	40.75362	-73.98377	False	mode
2	THE VILLAGE OF HARLEM....NEW YORK !	unconfirmed	Elise	Manhattan	Harlem	40.80902	-73.94190	True	flex
3	Home away from home	unconfirmed	Garry	Brooklyn	Clinton Hill	40.68514	-73.95976	True	mode

In [18]:
```
## Removing the dollar sign and comma from the price and service columns.


df['price'] = df['price'].str.replace(',', '').str.replace('$', '')
df['service_fee'] = df['service_fee'].str.replace(',', '').str.replace('$', '')
```

In [19]:
```
## If necessary, convert the price and service columns to their appropriate data type.


def change_column_to_float(df, col_name):
    df[col_name] = df[col_name].astype(float)
    return df
```

In [20]:
```
# Calling change_column_to_float function on the price columns to change it appropriate data type to float.


price_df = change_column_to_float(df, "price")
price_df.head(4)
```

Out[20]:

	name	host_identity_verified	host_name	neighbourhood_group	neighbourhood	lat	long	instant_bookable	cancellation_po
0	Clean & quiet apt home by the park	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	-73.97237	False	s
1	Skylit Midtown Castle	verified	Jenna	Manhattan	Midtown	40.75362	-73.98377	False	mode
2	THE VILLAGE OF HARLEM....NEW YORK !	unconfirmed	Elise	Manhattan	Harlem	40.80902	-73.94190	True	flex
3	Home away from home	unconfirmed	Garry	Brooklyn	Clinton Hill	40.68514	-73.95976	True	mode

In [21]:
```
# Calling change_column_to_float function on the service_fee columns to change it appropriate data type to float.


service_df = change_column_to_float(df, "service_fee")
service_df.head(4)
```

Out[21]:

	name	host_identity_verified	host_name	neighbourhood_group	neighbourhood	lat	long	instant_bookable	cancellation_po
0	Clean & quiet apt home by the park	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	-73.97237	False	s
1	Skylit Midtown Castle	verified	Jenna	Manhattan	Midtown	40.75362	-73.98377	False	mode
2	THE VILLAGE OF HARLEM....NEW YORK !	unconfirmed	Elise	Manhattan	Harlem	40.80902	-73.94190	True	flex
3	Home away from home	unconfirmed	Garry	Brooklyn	Clinton Hill	40.68514	-73.95976	True	mode

In [22]:
```
# AS AN EXTRA BONUS: I am displaying the data types of the columns again to make sure everything is ok.


df.dtypes
```

Out[22]:
```
name                                object
host_identity_verified              object
host_name                           object
neighbourhood_group                 object
neighbourhood                       object
lat                                float64
long                               float64
instant_bookable                      bool
cancellation_policy                 object
room_type                           object
construction_year                  float64
price                              float64
service_fee                        float64
minimum_nights                     float64
number_of_reviews                  float64
last_review                         object
reviews_per_month                  float64
review_rate_number                 float64
calculated_host_listings_count     float64
days_booked                        float64
house_rules                         object
license                             object
dtype: object
```

In [23]:
```
# AS AN EXTRA BONUS: I am viewing all the columns of my dataframe again to make sure everything is ok.


df.columns
```

Out[23]:
```
Index(['name', 'host_identity_verified', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'lat', 'long', 'instant_bookable',
       'cancellation_policy', 'room_type', 'construction_year', 'price',
       'service_fee', 'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'review_rate_number',
       'calculated_host_listings_count', 'days_booked', 'house_rules',
       'license'],
      dtype='object')
```

Task 4: Exploratory Data Analysis (Any Tool)

- List the count of various room types available in the dataset.
- Which room type has the most strict cancellation policy?
- List the average price per neighborhood group, and highlight the most expensive neighborhood to rent from.

If using Python for this exercise, please include the code in the cells below. If using any other tool, please include screenshoots of your work.

In [24]:
```
## List the count of the various room types avaliable in the dataset.


df["room_type"].value_counts()
```

Out[24]:
```
Entire home/apt    51995
Private room       44887
Shared room         2149
Hotel room           115
Name: room_type, dtype: int64
```

In [25]:
```
## Which room type adheres to more strict cancellation policy in dataset


# Filtering the relevant columns
relevant_columns = ['room_type', 'cancellation_policy']
filtered_df = df[relevant_columns]

# Group data by room_type and cancellation_policy and count the combinations
group_df = filtered_df.groupby(['room_type', 'cancellation_policy']).size().reset_index(name='count')

# Sort the data by count in descending order
sorted_df = group_df.sort_values('count', ascending=False)

# Select the room type with the most strict cancellation policy
most_strict_room_type = sorted_df.iloc[0]['room_type']

# Print the result
print("The room type with the most strict cancellation policy is:", most_strict_room_type)


# The below print message shows that the Entire home/apt room type has the most strict cancellation policy
```

```
The room type with the most strict cancellation policy is: Entire home/apt
```

In [26]:
```
## Listing the average price per neighborhood group in the dataset


average_prices = df.groupby("neighbourhood_group")["price"].mean()

for place, average_price in average_prices.items():
    average_price = format(average_price, ".2f")
    print(f"{place}: {average_price}")
```

```
Bronx: 625.27
Brooklyn: 625.45
Manhattan: 621.64
Queens: 628.67
Staten Island: 625.06
brookln: 580.00
manhatan: 460.00
```

In [27]:
```
## List the most expensive neighborhood group for Airbnb rentals


grouped_df = df.groupby("neighbourhood_group")["price"].mean().reset_index()
sorted_df = grouped_df.sort_values("price", ascending=False)
most_expensive_group = sorted_df.iloc[0]["neighbourhood_group"]
print(f"The most expensive neighborhood group for Airbnb rentals is:", most_expensive_group)

# The below print message shows that Queens is the most expensive neighborhood group
```

```
The most expensive neighborhood group for Airbnb rentals is: Queens
```

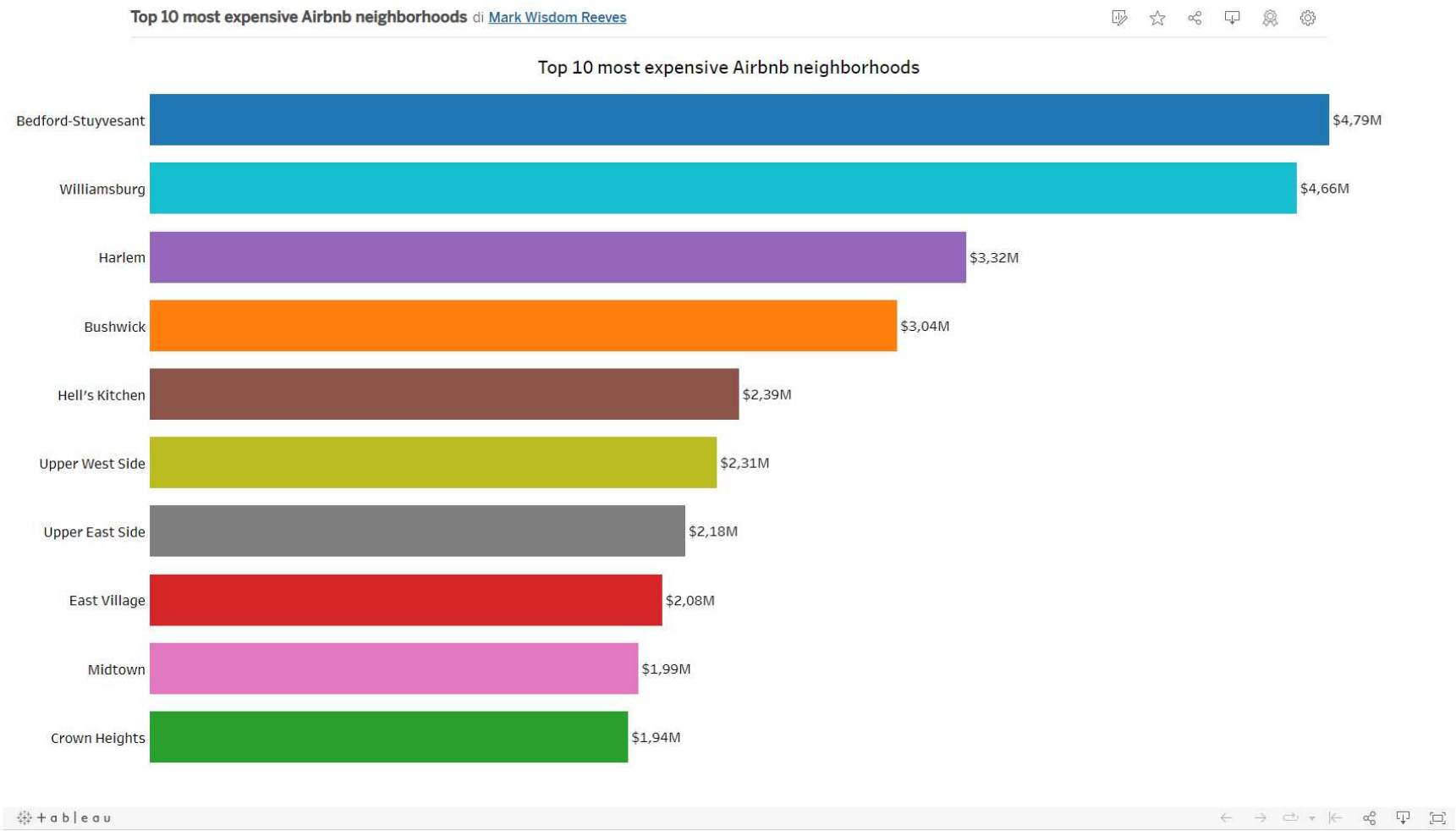
Task 5a: Data Visualization (Any Tool)

- Create a horizontal bar chart to display the top 10 most expensive neighborhoods in the dataset.
  - Create another chart with the 10 cheapest neighborhoods in the dataset.
- Create a box and whisker chart that showcases the price distribution of all listings split by room type.

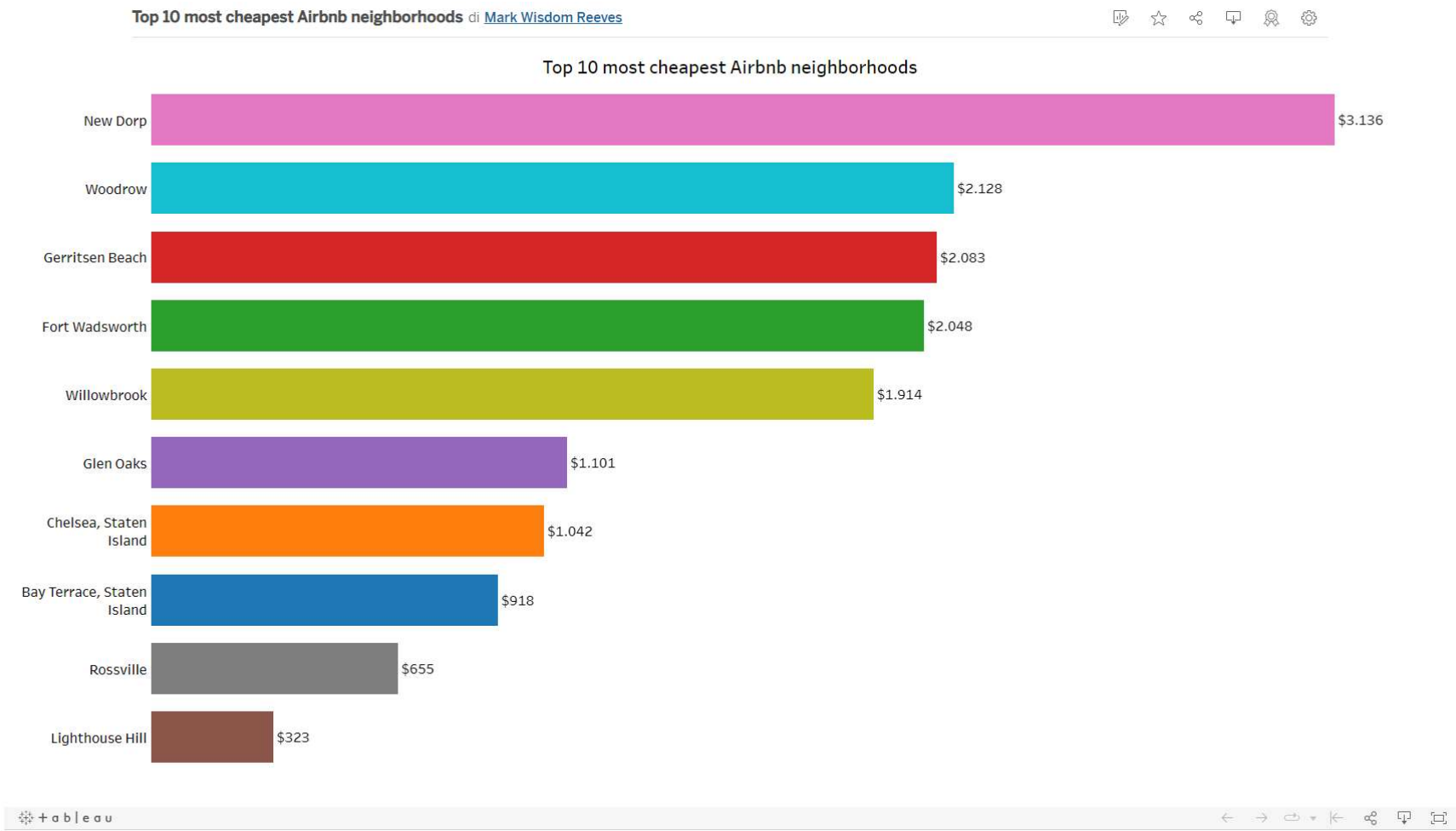
If using Python for this exercise, please include the code in the cells below. If using any other tool, please include screenshoots of your work.

In [28]:
```
# AS AN EXTRA BINUS: I am creating a cleaned dataset as a csv file and re-loadeding it again for virtualizations


df.to_csv("../IBM_Capstone_Project_2023/Clean_Airbnb_Open_Data.csv", index=False)


df = pd.read_csv("../IBM_Capstone_Project_2023/Clean_Airbnb_Open_Data.csv")
```

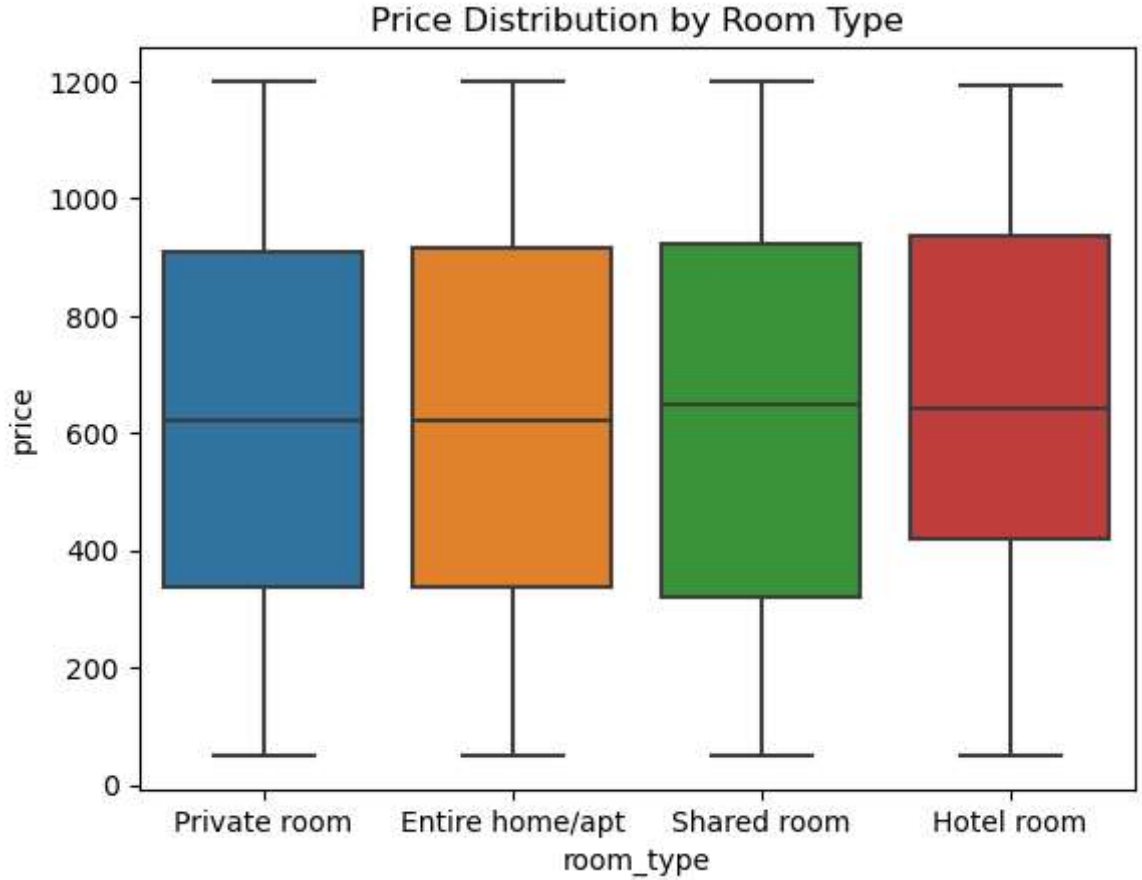
In [29]:
```
## Create a horizontal bar chart to display the top 10 most expensive neighborhoods in the dataset

# I used Tableau to create the chart in this section

# Below is the link to the live and interactive project on my Tableau profile:

# https://public.tableau.com/app/profile/mark.wisdom.reeves
```



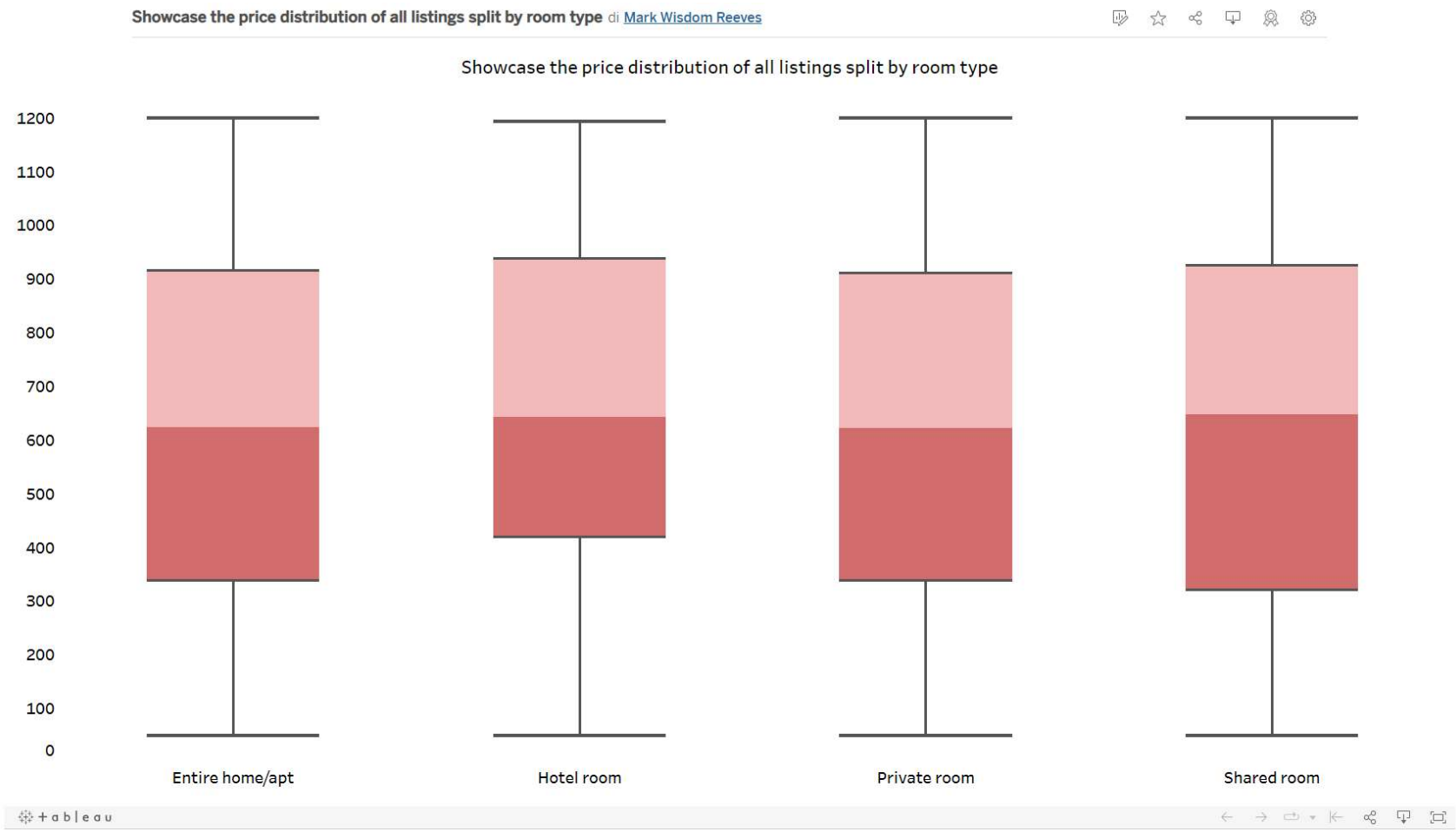
In [30]:
```
## Create a horizontal bar chart to display the top cheapest neighborhoods in the dataset

# I used Tableau to create the chart in this section

# Below is the link to the live and interactive project on my Tableau profile:

# https://public.tableau.com/app/profile/mark.wisdom.reeves
```



In [31]:
```
## Create a box and whisker chart that showcases the price distribution of all listings split by room type.


# Plot the box and whisker chart using Seaborn
sns.boxplot(x='room_type', y='price', data=df)

# Add labels and title to the chart
plt.xlabel('room_type')
plt.ylabel('price')
plt.title('Price Distribution by Room Type')

# Display the chart
plt.show()
```



In [32]:
```
## Create a box and whisker chart that showcases the price distribution of all listings split by room type.

# AS AN EXTRA BONUS: Below is the Tableau Version of this section

# Below is the link to the live and interactive project on my Tableau profile:

# https://public.tableau.com/app/profile/mark.wisdom.reeves
```



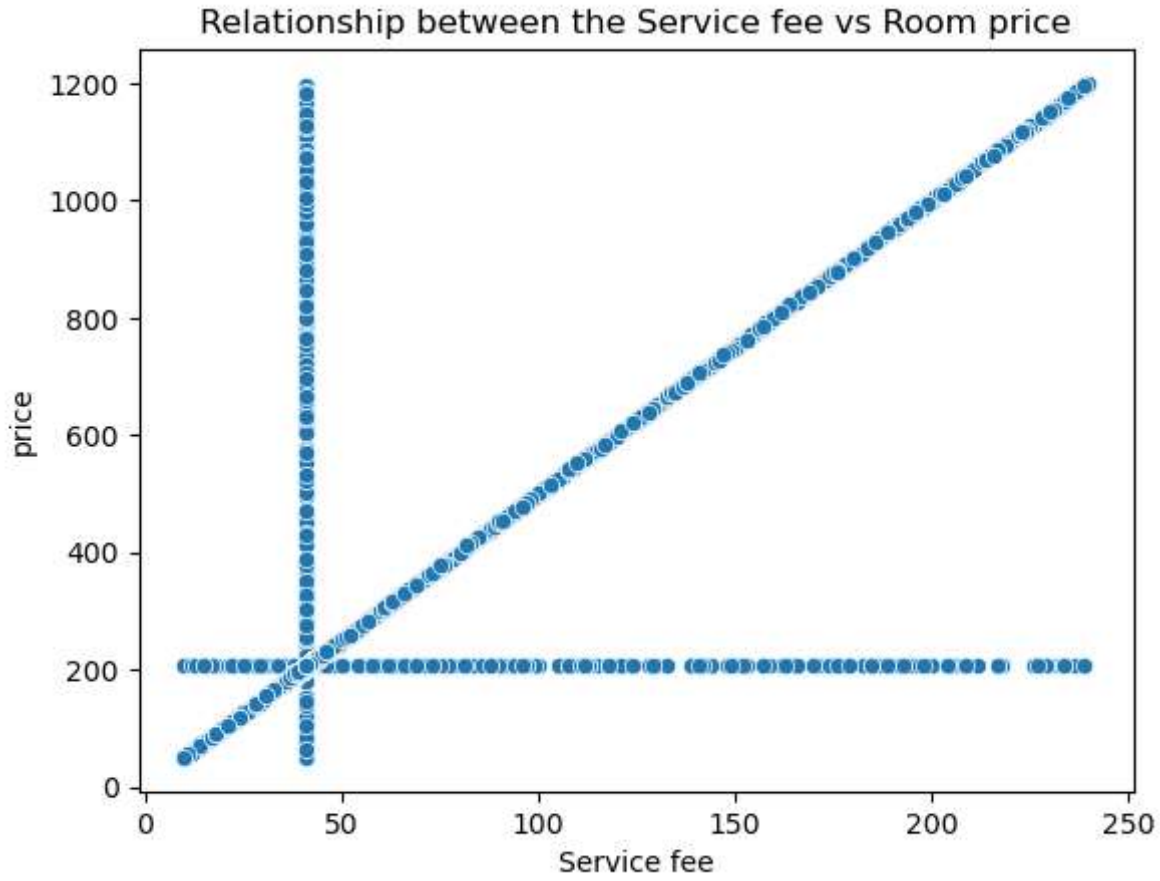
Task 5b: Data Visualization (Any Tool)

- Create a scatter plot to illustrate the relationship between the service fee and the room price and write down the kind of correlation, if any, that you see.
- Create a line chart to showcase the total amount of listings available per year.

If using Python for this exercise, please include the code in the cells below. If using any other tool, please include screenshoots of your work.

In [33]:
```
## Create a scatter plot to illustrate the relationship between the service fee and the room price


# Plot the Scatterplot
ax = sns.scatterplot(x="service_fee", y="price", data=df)

# Set the axis labels and title
ax.set_title("Relationship between the Service fee vs Room price")
ax.set_xlabel("Service fee")
```

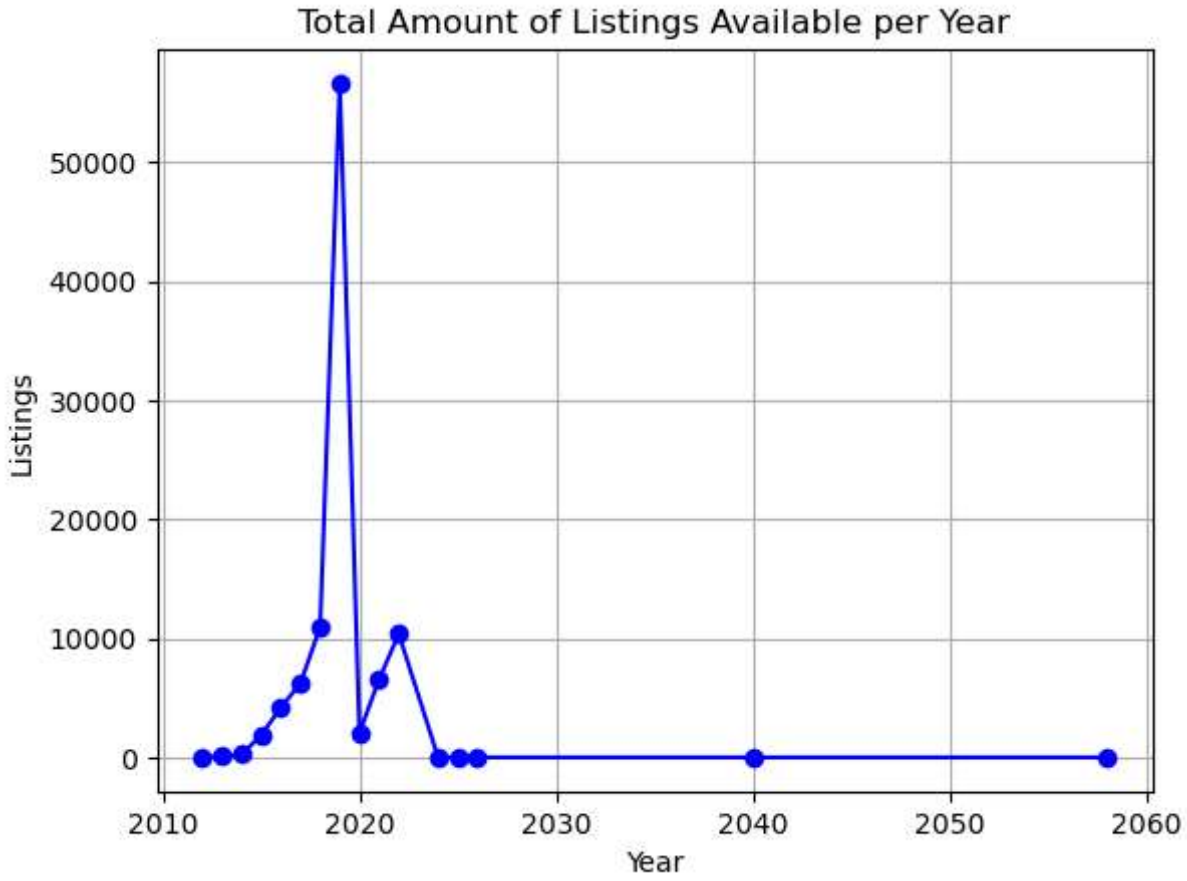
Out[33]:
```
Text(0.5, 0, 'Service fee')
```



In [34]:
```
## Display any correlation between the relationship of the service fee and the room price.


service_fee = df["service_fee"]

price = df["price"]

# Calculate the correlation coefficient
correlation = np.corrcoef(service_fee, price)[0, 1]

# Print Correlation coefficient
print("Correlation coefficient:", correlation)
```

```
Correlation coefficient: 0.9939395285701236
```

In [35]:
```
## Create a line chart to showcase the total amount of listings available per year.


# Convert 'last_review' column to datetime
last_review_date = pd.to_datetime(df['last_review'])

# Extract the year from 'last_review' column and create a new 'year' column
year = last_review_date.dt.year

# Group by 'year' and calculate the total amount of listings per year
listings_per_year = df.groupby(year).size()

# Create line chart
listings_per_year.plot(marker='o', linestyle='-', color='b')

# Customize chart
plt.title('Total Amount of Listings Available per Year')
plt.xlabel('Year')
plt.ylabel('Listings')
plt.grid(True)

# Show chart
plt.show()
```

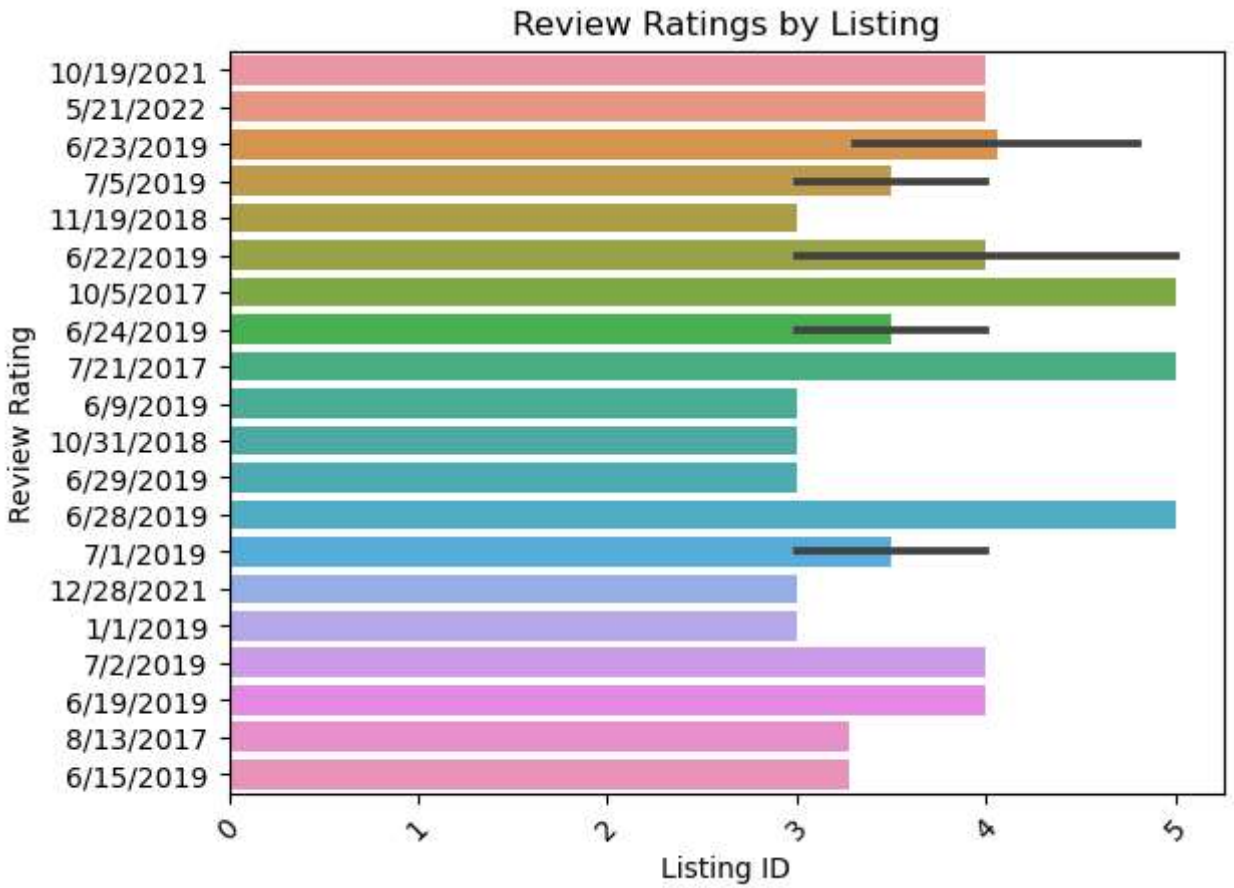


Task 5c: Data Visualization (Any Tool)

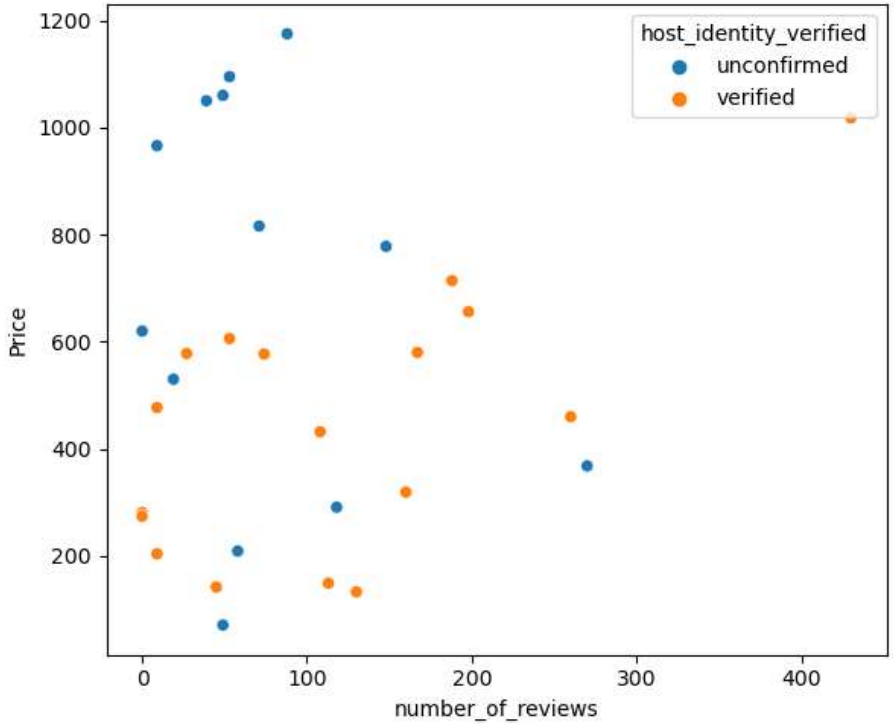
- Create a data visualization of your choosing using one of the review columns in isolation or in combination with another column.
- Create a visualization to compare at least two different variables between super hosts and regular hosts.

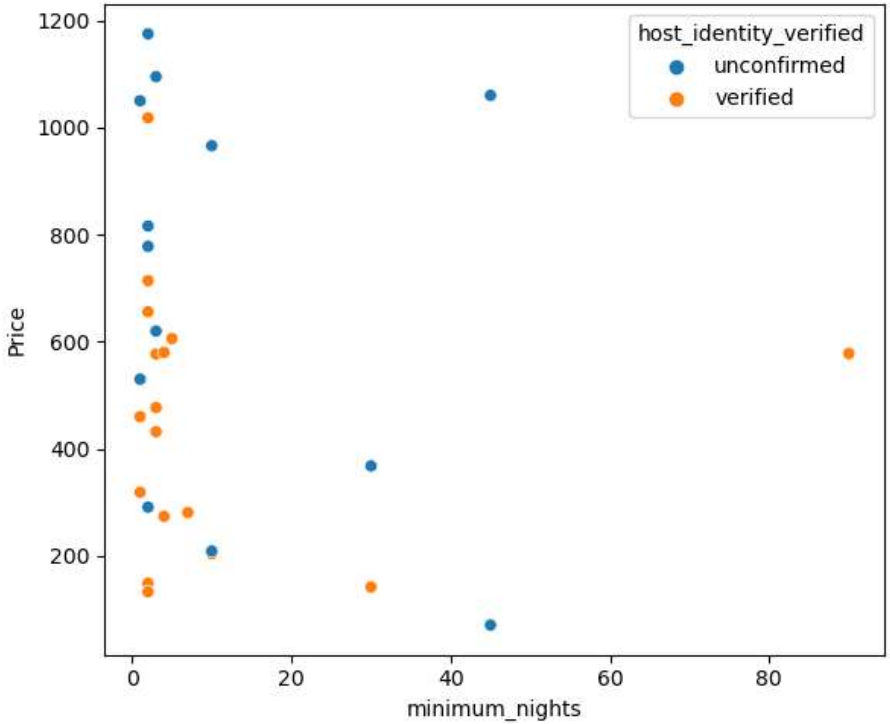
If using Python for this exercise, please include the code in the cells below. If using any other tool, please include screenshoots of your work.

In [36]:
```
## Create a data viz of your choosing by using one of the review columns with another column.


# Here, an displaying a sample of 30 rows of my dataset.
df = df[0:30]

# Convert 'last_review' column to datetime
last_review_date = pd.to_datetime(df['last_review'])

# Extract the year from 'last_review' column and create a new 'year' column
year = last_review_date.dt.year

# Create a bar chart using Seaborn
sns.barplot(x="review_rate_number", y="last_review", data=df)

# Set the axis labels and title
plt.xlabel("Listing ID")
plt.ylabel("Review Rating")
plt.title("Review Ratings by Listing")

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45)

# Display the chart
plt.show()
```


Review Ratings by Listing
10/19/2021
5/21/2022
6/23/2019
7/5/2019
11/19/2018
6/22/2019
10/5/2017
6/24/2019
7/21/2017
6/9/2019
10/31/2018
6/29/2019
6/28/2019
7/1/2019
12/28/2021
1/1/2019
7/2/2019
6/19/2019
8/13/2017
6/15/2019
Review Rating
0
1
2
3
4
5
Listing ID

In [37]:
```
## Create a visualization to compare at least two different variables between verified and unconfirmed.


# Select variables
variables = ['number_of_reviews', 'minimum_nights']

# Create subplots
fig, axes = plt.subplots(nrows=1, ncols=len(variables), figsize=(12, 5))

# Generate scatter plots for each variable
for i, var in enumerate(variables):
    sns.scatterplot(x=var, y='price', hue='host_identity_verified', data=df, ax=axes[i])
    axes[i].set_xlabel(var)
    axes[i].set_ylabel('Price')


plt.tight_layout()

plt.show()
```


host_identity_verified
unconfirmed
verified
Price
number_of_reviews
1200
1000
800
600
400
200
0
100
200
300
400


host_identity_verified
unconfirmed
verified
Price
minimum_nights
1200
1000
800
600
400
200
0
20
40
60
80

In [ ]:

In [ ]: