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.

So all the best for your Data Analytics Journey on Airbnb data!!!

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 all libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:

```
import warnings

# Filter out specific warning
warnings.filterwarnings("ignore")
```

In [4]:

```
## Read the csv file
df=pd.read_csv("/Users/myyntiimac/Desktop/IBM project/Airbnb_Open_Data.csv")

```

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

5 rows × 26 columns

In [6]:

```
## Display the data types
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 screenshoots before and after the elimination of the columns.

In [7]:

```
columns_to_drop = ["host id", "id", "country", "country code"]
df1 = df.drop(columns_to_drop, axis=1)
```

In [8]:

```
df1.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')
```

Host id and Id both are same but there is anouther column called host name so all 3 column containing identity information of host , so we can keep host name and delete others

As all the city neighbourhood are in USA so we can delete the country and countrycode , we can perform the analysis by considering Neghbourhood city

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 the dataframe and display the count in ascending order.
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102599 entries, 0 to 102598
Data columns (total 22 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   NAME                            102349 non-null  object 
 1   host_identity_verified          102310 non-null  object 
 2   host name                       102193 non-null  object 
 3   neighbourhood group             102570 non-null  object 
 4   neighbourhood                   102583 non-null  object 
 5   lat                             102591 non-null  float64
 6   long                            102591 non-null  float64
 7   instant_bookable                102494 non-null  object 
 8   cancellation_policy             102523 non-null  object 
 9   room type                       102599 non-null  object 
 10  Construction year               102385 non-null  float64
 11  price                           102352 non-null  object 
 12  service fee                     102326 non-null  object 
 13  minimum nights                  102190 non-null  float64
 14  number of reviews               102416 non-null  float64
 15  last review                     86706 non-null   object 
 16  reviews per month               86720 non-null   float64
 17  review rate number              102273 non-null  float64
 18  calculated host listings count  102280 non-null  float64
 19  availability 365                102151 non-null  float64
 20  house_rules                     50468 non-null   object 
 21  license                         2 non-null       object 
dtypes: float64(9), object(13)
memory usage: 17.2+ MB
```

In [10]:

```
missvalue = df1.isnull().sum().sort_values(ascending=True)
missvalue
```

Out[10]:

```
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
```

Insight: we saw almost all variable contain missing value , so instad of single impution , used for for loop for time save First catagorical imputation than numercal , catagorical fill with mode and numerical filled with mean

In [11]:

```
# Find categorical columns in the DataFrame
categorical_columns = df1.select_dtypes(include=['object']).columns

# Perform imputation for each categorical column
for column in categorical_columns:
    most_frequent_category = df1[column].mode()[0]
    df1[column].fillna(most_frequent_category, inplace=True)


```

In [12]:

```
# Find numerical columns in the DataFrame
numerical_columns = df1.select_dtypes(include=['int64', 'float64']).columns

# Perform imputation for each numerical column
for column in numerical_columns:
    mean_value = df1[column].mean()
    df1[column].fillna(mean_value, inplace=True)

```

In [13]:

```
df1.isnull().sum().sum()
```

Out[13]:

```
0
```

In [14]:

```
## Check whether there are any duplicate values in the dataframe and if present remove them.
#check the number of records before remove duplicate values
len(df1)
```

Out[14]:

```
102599
```

In [15]:

```
# delete duplication
df2= df1.drop_duplicates()

```

In [16]:

```
## Display the total number of records in the dataframe after removing the duplicates.
len(df2)
```

Out[16]:

```
99146
```

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 [17]:

```
## Rename the column.
df2.rename(columns={"availability 365": "days_booked"}, inplace=True)
```

In [35]:

```
df2.columns
```

Out[35]:

```
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')
```

In [22]:

```
## Convert all column names to lowercase and replace the spaces with an underscore "_"
df2.columns = df2.columns.str.lower().str.replace(' ', '_')
```

In [23]:

```
df2.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')
```

In [31]:

```
# Remove the dollar sign and comma from the columns
df2['price'] = df2['price'].replace('[\$,]', '', regex=True).astype(float)
df2['service_fee'] = df2['service_fee'].replace('[\$,]', '', regex=True).astype(int)


```

In [26]:

```
df2.head()
```

Out[26]:

	name	host_identity_verified	host_name	neighbourhood_group	neighbourhood	lat	long	instant_bookabl
0	Clean & quiet apt home by the park	unconfirmed	Madaline	Brooklyn	Kensington	40.64749	-73.97237	Fals
1	Skylit Midtown Castle	verified	Jenna	Manhattan	Midtown	40.75362	-73.98377	Fals
2	THE VILLAGE OF HARLEM....NEW YORK !	unconfirmed	Elise	Manhattan	Harlem	40.80902	-73.94190	Tru
3	Home away from home	unconfirmed	Garry	Brooklyn	Clinton Hill	40.68514	-73.95976	Tru
4	Entire Apt: Spacious Studio/Loft by central park	verified	Lyndon	Manhattan	East Harlem	40.79851	-73.94399	Fals

5 rows × 22 columns

Task 4: Exploratory Data Analysis (Any Tool)

- List the count of various room types avaliable 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 [42]:

```
## List the count of various room types avaliable with Airbnb
# Get the value counts of the "room_type" column
room_type_counts = df2['room_type'].value_counts()

# Display the value counts
print(room_type_counts)
```

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

In [43]:

```
## Which room type adheres to more strict cancellation policy
strict_cancellation_room = df2[df2['cancellation_policy'] == 'strict']['room_type'].value_counts().idx
strict_cancellation_room
```

Out[43]:

```
'Entire home/apt'
```

In [49]:

```
## List the prices by neighborhood group and also mention which is the most expensive neighborhood group
grouped_prices = df2.groupby("neighbourhood_group")['price'].mean().reset_index()
grouped_prices = grouped_prices.sort_values('price', ascending=False)
print(grouped_prices)
```

```
  neighbourhood_group       price
3              Queens  628.668822
1            Brooklyn  625.451927
0               Bronx  625.271511
4       Staten Island  625.060870
2           Manhattan  621.641437
5             brookln  580.000000
6            manhatan  460.000000
```

Insight: The most expensive group is Queens which represents 628.668822

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 [61]:

```
# Group the data by 'Neighborhood' and calculate the mean price
grouped_prices = df2.groupby('neighbourhood')['price'].mean().reset_index()


```

In [62]:

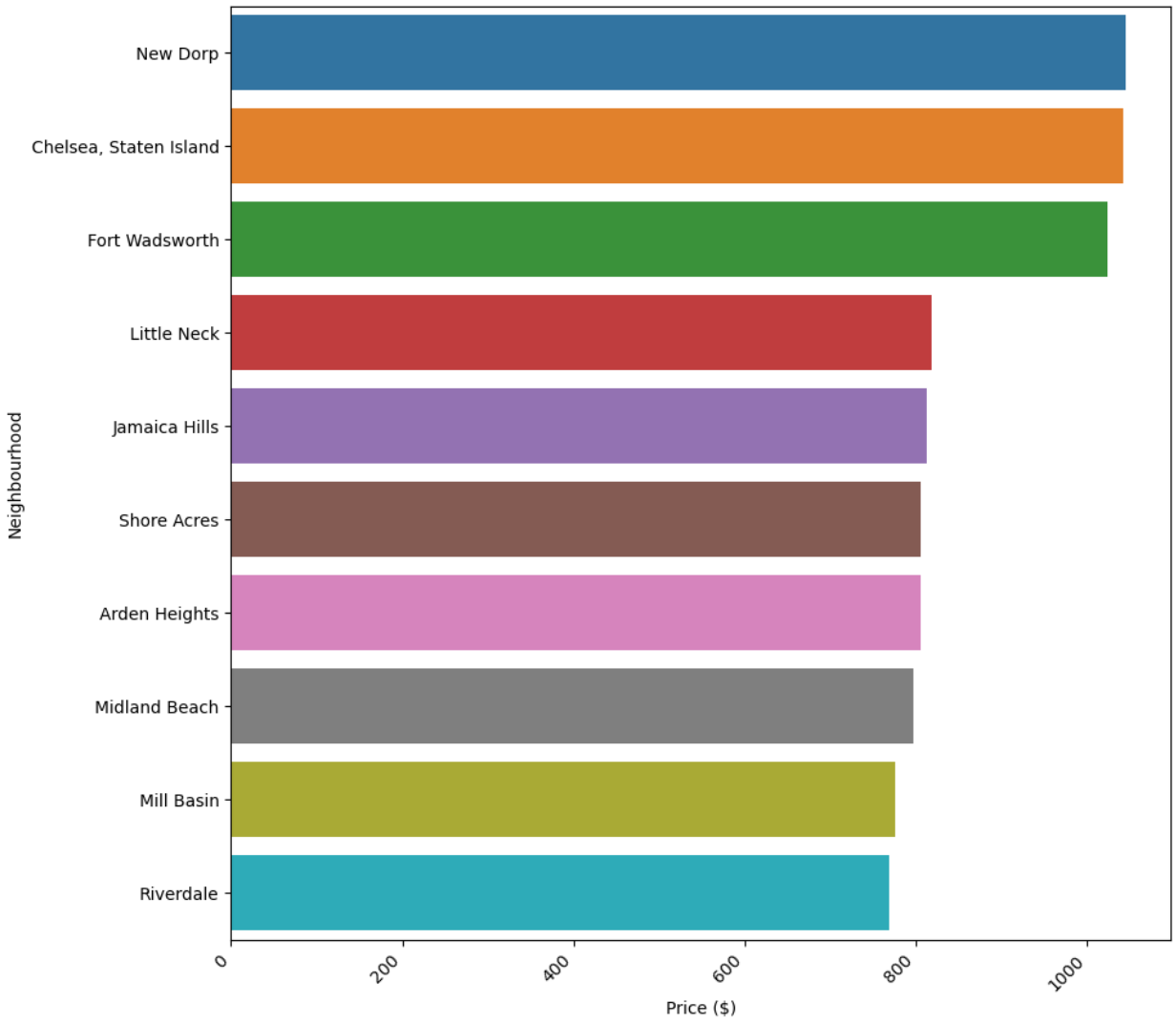
```
# Sort the data in descending order based on the mean price and select the top 10
top_10_expensive = grouped_prices.nlargest(10, 'price')
top_10_expensive

```

Out[62]:

	neighbourhood	price
144	New Dorp	1045.333333
35	Chelsea, Staten Island	1042.000000
83	Fort Wadsworth	1024.000000
119	Little Neck	817.750000
110	Jamaica Hills	812.904762
179	Shore Acres	805.142857
1	Arden Heights	804.888889
129	Midland Beach	796.176471
132	Mill Basin	775.142857
170	Riverdale	768.736842

In [68]:

```
# Display horizontal bar chart using seaborn
plt.figure(figsize=(10, 10))
sns.barplot(x='price', y='neighbourhood', data=top_10_expensive)

plt.xlabel('Price ($)')
plt.xticks(rotation=45, ha='right')
plt.ylabel('Neighbourhood')
plt.yticks(fontsize=10)
plt.show()
```




```
insight:top 10 expensive neighberhood plae where new drop is highest price and followedby other
```

In [69]:

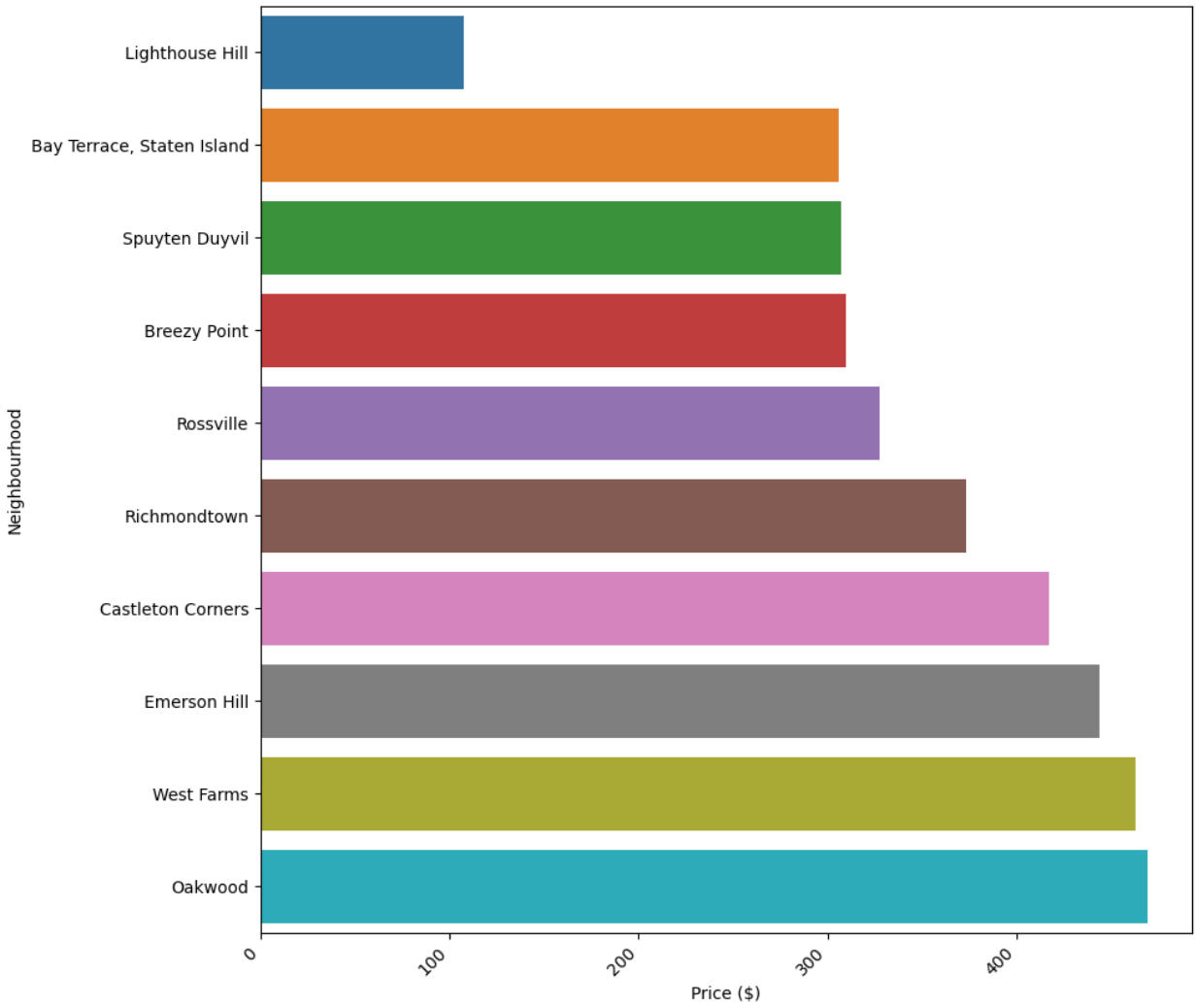
```
# Sort the data in descending order based on the mean price and select the top 10
Least_10_cheapest = grouped_prices.nsmallest(10, 'price')
Least_10_cheapest

```

Out[69]:

	neighbourhood	price
117	Lighthouse Hill	107.666667
9	Bay Terrace, Staten Island	306.000000
187	Spuyten Duyvil	307.000000
21	Breezy Point	309.888889
175	Rossville	327.500000
168	Richmondtown	373.400000
33	Castleton Corners	417.230769
71	Emerson Hill	443.800000
211	West Farms	463.166667
151	Oakwood	469.307692

In [70]:

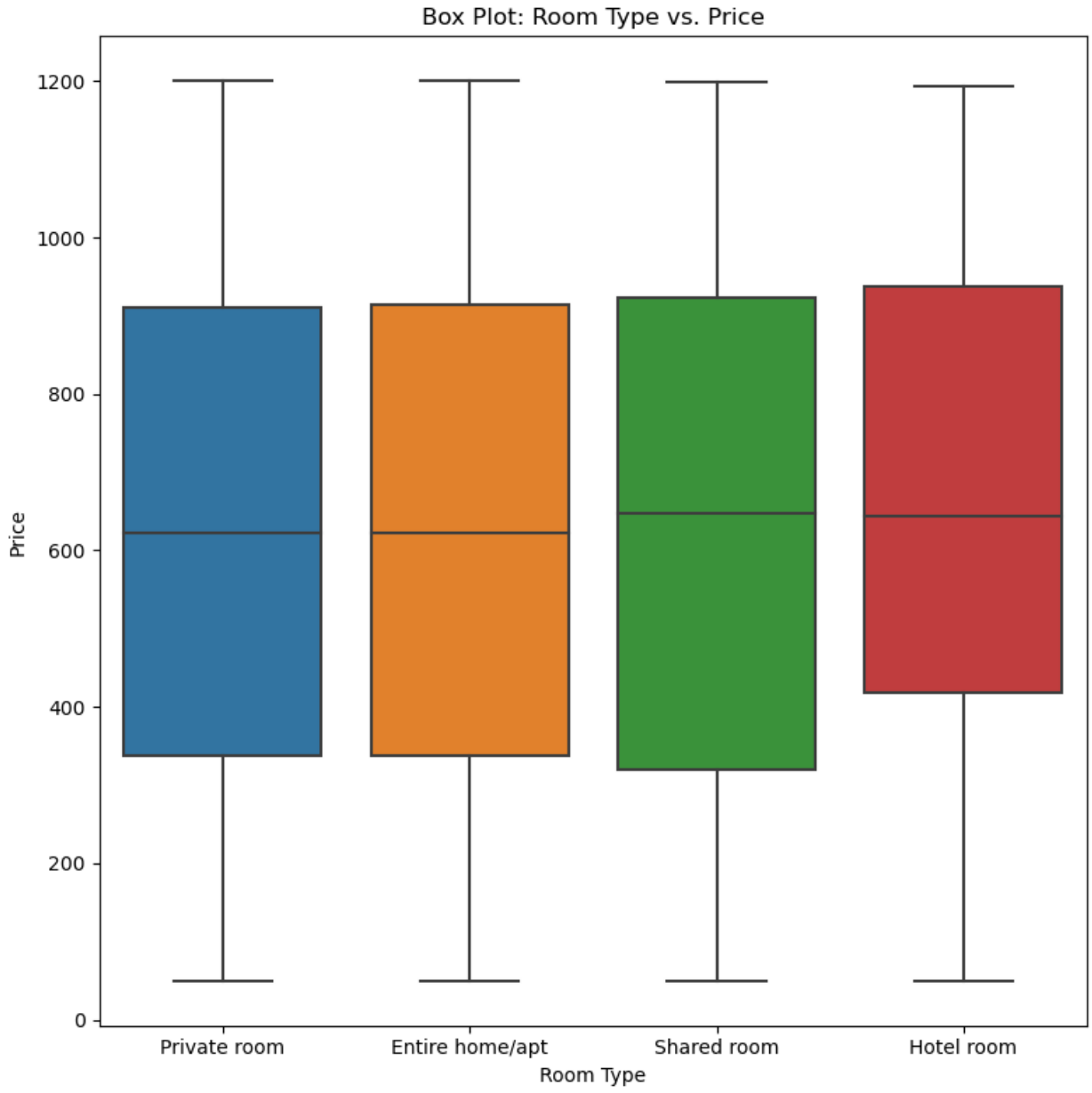
```
# Display horizontal bar chart using seaborn
plt.figure(figsize=(10, 10))
sns.barplot(x='price', y='neighbourhood', data=Least_10_cheapest)

plt.xlabel('Price ($)')
plt.xticks(rotation=45, ha='right')
plt.ylabel('Neighbourhood')
plt.yticks(fontsize=10)
plt.show()
```




```
Insight:10 cheapest neighbourhood place where lighthouse hill is most cheapest represent around 100
```

In [90]:

```
# Create the box plot
# Create the box plot
plt.figure(figsize=(8, 8))  # Adjust the figure size if needed

sns.boxplot(x='room_type', y='price', data=df2)
plt.xlabel('Room Type')
plt.ylabel('Price')
plt.title('Box Plot: Room Type vs. Price')

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




```
Insight:hotel room price is start is higher than other room type price where average price of shared roo and hotel room almost same
```

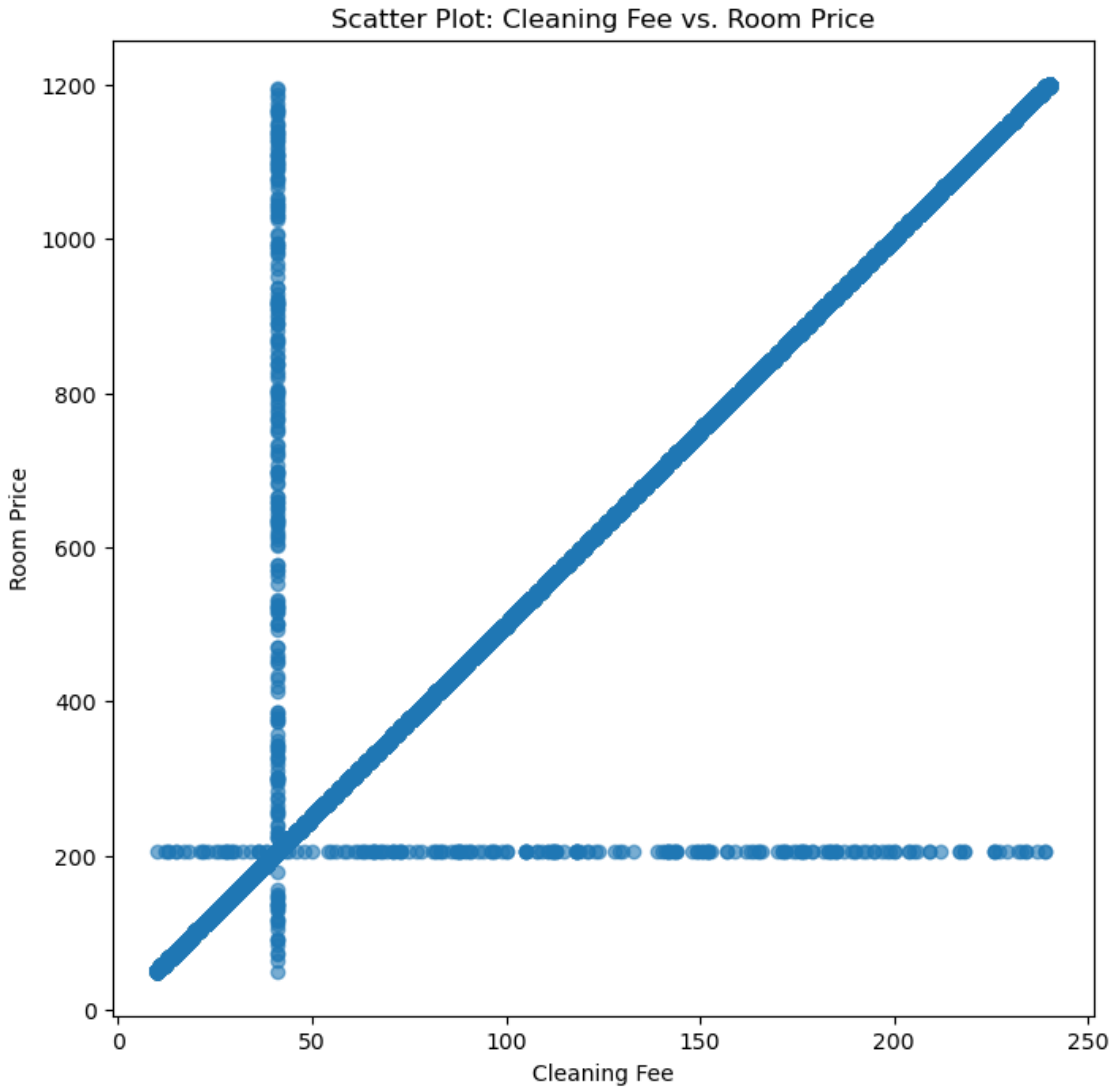
Task 5b: Data Visualization (Any Tool)

- Create a scatter plot to illustrate the relationshi between the cleaning 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 [71]:

```
# Extract the data for the scatter plot
cleaning_fee = df2['service_fee']
room_price = df2['price']

# Create the scatter plot
plt.figure(figsize=(8, 8))
plt.scatter(cleaning_fee, room_price, alpha=0.6)
plt.xlabel('Cleaning Fee')
plt.ylabel('Room Price')
plt.title('Scatter Plot: Cleaning Fee vs. Room Price')
plt.show()
```



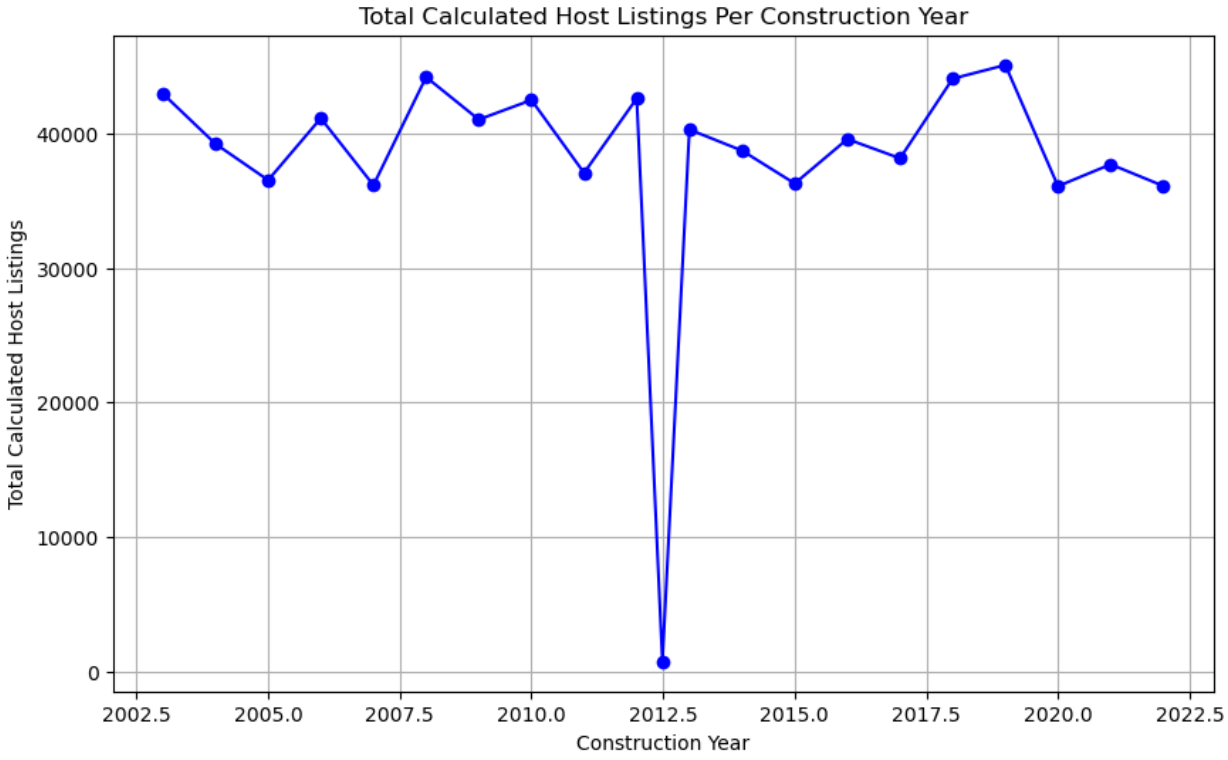
In [81]:

```
# Extract the data for the scatter plot
cleaning_fee = df2['service_fee']
room_price = df2['price']
# Calculate the correlation coefficient
correlation_coefficient = np.corrcoef(cleaning_fee, room_price)[0, 1]
correlation_type = "Positive" if correlation_coefficient > 0 else "Negative" if correlation_coefficient

print("Correlation coefficient:", correlation_coefficient)
print("Correlation type:", correlation_type)

```

```
Correlation coefficient: 0.9939395285701238
Correlation type: Positive
```

In [30]:

```
# Group the data by year and calculate the sum of calculated_host_listings_count for each year
listings_per_year = df2.groupby('construction_year')['calculated_host_listings_count'].sum()

# Plotting the line chart
plt.figure(figsize=(10, 6))
plt.plot(listings_per_year.index, listings_per_year.values, marker='o', linestyle='-', color='b')
plt.xlabel('Construction Year')
plt.ylabel('Total Calculated Host Listings')
plt.title('Total Calculated Host Listings Per Construction Year')
plt.grid(True)
plt.show()
```




```
Insight :the overall trend of host listing flactuated over the years except 2012 where listing almost
0
```

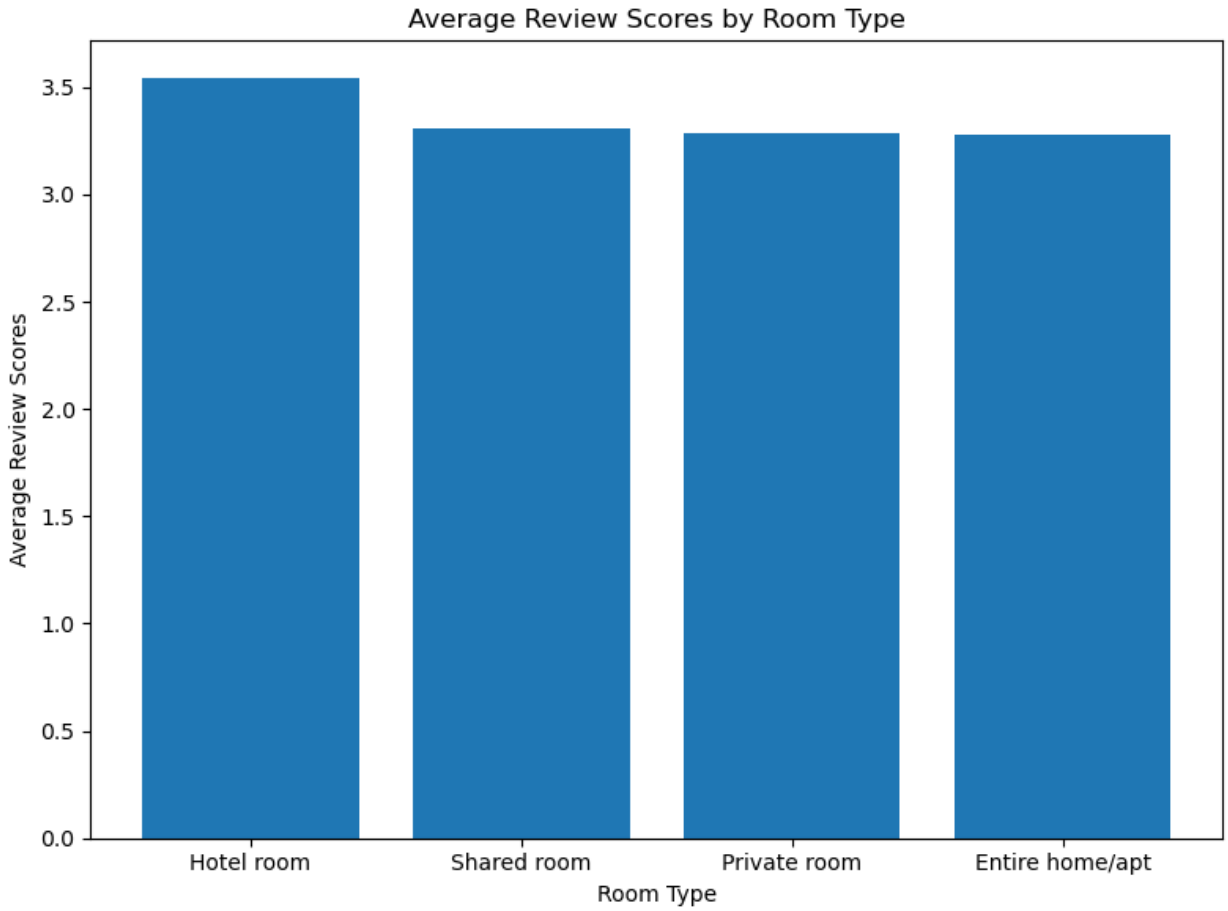
Task 5c: Data Visualization (Any Tool)

- 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 [93]:

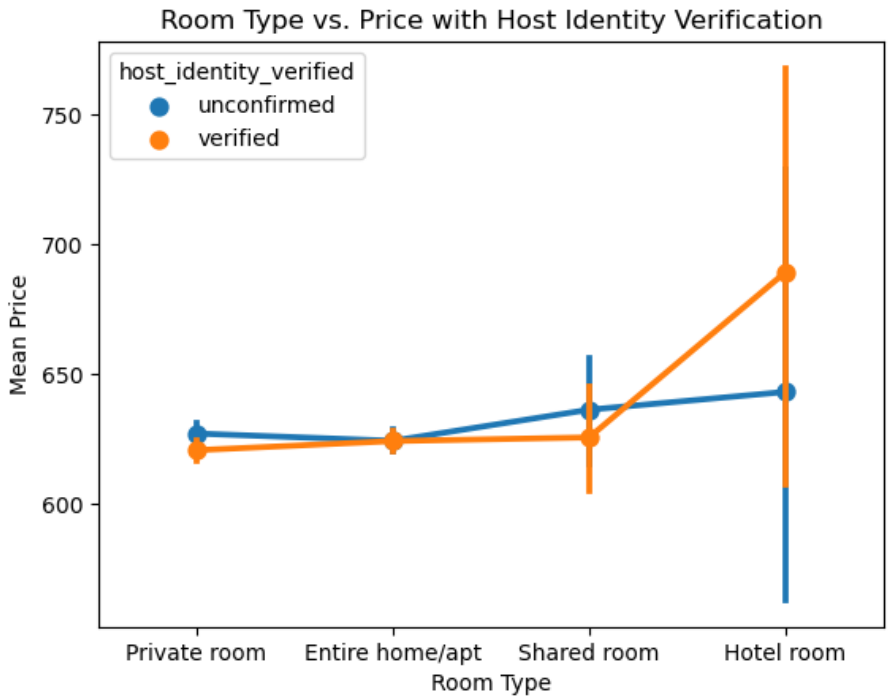
```
# Group the data by 'room_type' and calculate the mean review scores
avg_review_scores = df2.groupby('room_type')['review_rate_number'].mean().reset_index()

# Sort the data in descending order based on the mean review scores
avg_review_scores = avg_review_scores.sort_values('review_rate_number', ascending=False)

# Create the bar plot
plt.figure(figsize=(8, 6))  # Adjust the figure size if needed

plt.bar(avg_review_scores['room_type'], avg_review_scores['review_rate_number'])
plt.xlabel('Room Type')
plt.ylabel('Average Review Scores')
plt.title('Average Review Scores by Room Type')

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




```
insight:hotel room average score high than the other rooms type
```

In [27]:

```
sns.pointplot(x="room_type", y="price", hue="host_identity_verified", data=df2)
plt.title('Room Type vs. Price with Host Identity Verification')
plt.xlabel('Room Type')
plt.ylabel('Mean Price')
plt.show()
```




```
Insight:in hotel room , the host who is verified are charged more
```