London Airbnb Market Analysis Summary using UK Airbnb Open Data

Author: Javier Alessandro Parra Dicillo

LinkedIn: https://www.linkedin.com/in/javier-parra-49443b163/

Dataset: https://github.com/pjavierdicillo/London_Airbnb_Market_Analysis/blob/main/listings.csv

---

Importing libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import math
from matplotlib import pyplot as plt
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
```

Data Source Information

```
df_source = pd.read_csv('listings.csv')

#Making a working copy of the original df

df = df_source.copy()
```

Exploratory Data Analysis:

Data Preparation: Preprocessing, Cleaning and Transformation:

```
#Drop unnecessary  columns and columns with 0 non-null values
columns_to_drop =
['neighbourhood_group','license','number_of_reviews_ltm']
df = df.drop(columns=columns_to_drop, axis=1)

# Replace null values in specified columns
df['name'] = df['name'].fillna('Unknown')
df['host_name'] = df['host_name'].fillna('Unknown')
df['last_review'] = df['last_review'].fillna('2000-01-01')
df['reviews_per_month'] = df['reviews_per_month'].fillna(0)

# Display count of null values after replacement for validation
null_counts_after_replacement = df.isnull().sum()
```

```python
print("Ensuring the df has no null-values")
print(" ")
```

```
Ensuring the df has no null-values
```

```python
# Convert data types
df['name'] = df['name'].astype(str)
df['host_name'] = df['host_name'].astype(str)
df['neighbourhood'] = df['neighbourhood'].astype(str)
df['room_type'] = df['room_type'].astype(str)
df['last_review'] = pd.to_datetime(df['last_review'])
```

Now we are going to look into the data insights:

Summary Statistics:

Looking at stats of continuous variables Focus is on price, minimum nights and availability Using a temporary df to exclude some columns from the table

```python
df_short = df.copy()

#Drop unnecessary  columns
cols_to_drop = 
['id','host_id','latitude','longitude','calculated_host_listings_count
']
df_short = df.drop(columns=cols_to_drop, axis=1)

df_short.describe()
```

```
                 price  minimum_nights  number_of_reviews  
reviews_per_month  \
count  69351.000000    69351.000000       69351.000000  
69351.000000   
mean     177.208822        5.997505          17.537051  
0.669043   
std      412.823024       25.709514          40.410763  
1.172270   
min        0.000000        1.000000           0.000000  
0.000000   
25%       55.000000        1.000000           1.000000  
0.010000   
50%      100.000000        2.000000           4.000000  
0.200000   
75%      180.000000        4.000000          16.000000  
0.850000   
max    25000.000000     1125.000000        1141.000000  
51.330000
```

```
       availability_365
count      69351.000000
mean         108.520266
std          132.821088
min            0.000000
25%            0.000000
50%           32.000000
75%          228.000000
max          365.000000
```

Price: -The maximum value is 25000£ and we might not want to take it into account for the analysis as it is impacting mean,std,etc. -The minimum value is 0£ and we have to think why a property is listed as 0£ and whether we want to include them in the analysis. Zero values will be excluded.

Minimum nights: -The maximum value apparently is 1125 nights and we might not want to keep it in the analysis. Zero values will be excluded.

Availability_365: -The minimum value is 0 meaning that some properties are not available. Listings with 0 days available will be excluded and only active properties will be considered

Handling outliers for columns price, minimum nights, and availability

```
print("Handling outliers for 'price' and excluding listings with Price = 0£")

Q1_price = df['price'].quantile(0.25)
Q3_price = df['price'].quantile(0.75)

IQR_price = Q3_price - Q1_price

lower_bound_price = int(Q1_price - 1.5 * IQR_price)
upper_bound_price = int(Q3_price + 1.5 * IQR_price)

# Replacing values greater than upper_bound_price and lower than lower_bound_price
df['price'] = df['price'].clip(lower=lower_bound_price, upper=upper_bound_price)
df_short['price'] = df_short['price'].clip(lower=lower_bound_price, upper=upper_bound_price) #Temporary

df = df[df["price"] > 0]
df_short = df_short[df_short["price"] > 0]
```

```
Handling outliers for 'price' and excluding listings with Price = 0£
```

```
print("Handling outliers for 'minimum_nights' and excluding listings with minimum_nights = 0")

Q1_min_nights = df['minimum_nights'].quantile(0.25)
Q3_min_nights = df['minimum_nights'].quantile(0.75)
```

```python
IQR_min_nights = Q3_min_nights - Q1_min_nights

lower_bound_min_nights = int(Q1_min_nights - 1.5 * IQR_min_nights)
upper_bound_min_nights = int(Q3_min_nights + 1.5 * IQR_min_nights)

# Replacing values greater than upper_bound_price and lower than
lower_bound_price
df['minimum_nights'] =
df['minimum_nights'].clip(lower=lower_bound_min_nights,
upper=upper_bound_min_nights)
df_short['minimum_nights'] =
df_short['minimum_nights'].clip(lower=lower_bound_min_nights,
upper=upper_bound_min_nights) #Temporary

df = df[df['minimum_nights'] > 0]
df_short = df_short[df_short['minimum_nights'] > 0]
```

```
Handling outliers for 'minimum_nights' and excluding listings with
minimum_nights = 0
```

```python
print("Handling outliers for 'availability_365' and excluding listings
with availability_365 = 0")

Q1_availability = df['availability_365'].quantile(0.25)
Q3_availability = df['availability_365'].quantile(0.75)

IQR_availability = Q3_availability - Q1_availability

lower_bound_availability = int(Q1_availability - 1.5 *
IQR_availability)
upper_bound_availability = int(Q3_availability + 1.5 *
IQR_availability)

# Replacing values greater than upper_bound_price and lower than
lower_bound_price
df['availability_365'] =
df['availability_365'].clip(lower=lower_bound_availability,
upper=upper_bound_availability)
df_short['availability_365'] =
df_short['availability_365'].clip(lower=lower_bound_availability,
upper=upper_bound_availability)

df = df[df['availability_365'] > 0]
df_short = df_short[df_short['availability_365'] > 0]
```

```
Handling outliers for 'availability_365' and excluding listings with
availability_365 = 0
```

```python
#Now let's see how the stats were adjusted

df_short.describe()
```

```
                 price  minimum_nights  number_of_reviews  
reviews_per_month  \
count  39936.000000    39936.000000       39936.000000   
39936.000000   
mean     159.665114        3.268004          22.688527   
0.981961   
std      106.969902        2.342131          48.104226   
1.390896   
min        1.000000        1.000000           0.000000   
0.000000   
25%       71.000000        1.000000           1.000000   
0.070000   
50%      130.000000        2.000000           6.000000   
0.530000   
75%      225.000000        5.000000          22.000000   
1.290000   
max      367.000000        8.000000        1141.000000   
51.330000   

       availability_365  
count      39936.000000  
mean         188.411008  
std          124.801222  
min            1.000000  
25%           70.000000  
50%          179.000000  
75%          316.000000  
max          365.000000  
```

Price: -The maximum value is 367£ which is more reasonable for the type of properties we are interested in.

Minimum nights: -The maximum value is 8 nights which is more reasonable

Availability_365: -The maximum value is now 365 days

All variables are now greater than zero.

OBJECTIVES:

To address the analysis looking to answer to the following questions:

-What is the most popular neighborhood? -what is the most expensive neighbourhood? -What is the avg price per neighbourhood? -What is the most popular room type overall -What is the most popular room type per neighbourhood? -What is the min night per room type overall? -

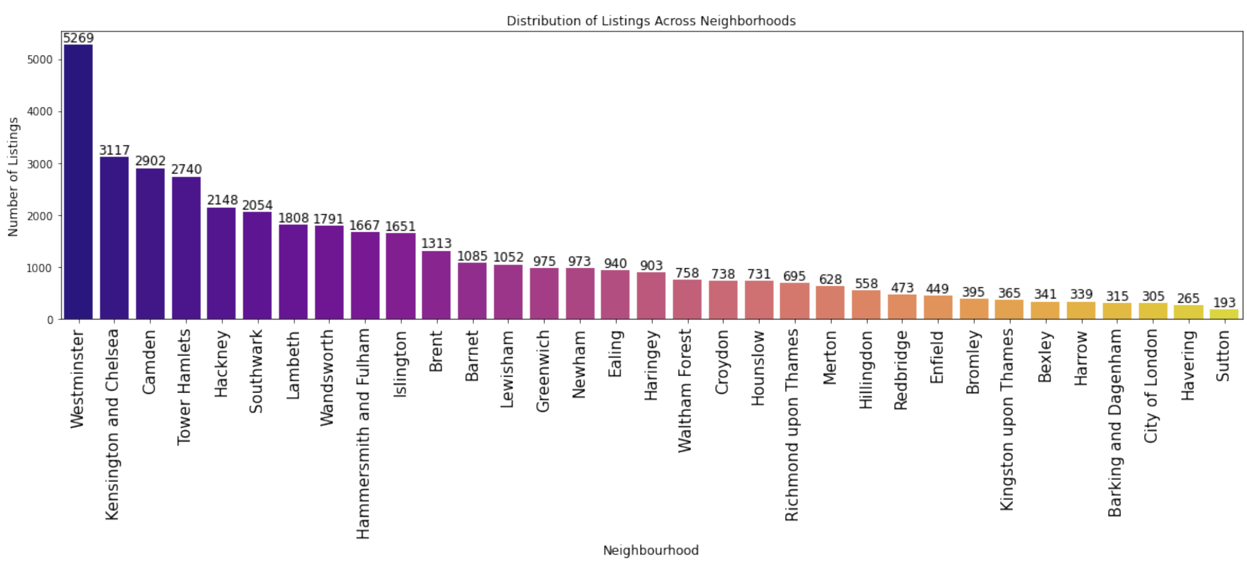
What is the most active neighbourhood? most number of last review in the 3 most recent months -what is the average revenue per property and per neighborhood

At the end of the analysis business opportunities may be identified to list a new rental property.

Data Visualisation

Visualisation 1: What is the most popular neighborhood?

```
#Listings across neighborhoods

plt.figure(figsize=(20, 5))

labels_1 = sns.countplot(x='neighbourhood', data=df,
order=df['neighbourhood'].value_counts().index,palette="plasma")

# Add count labels to each bar
for p in labels_1.patches:
    labels_1.annotate(f'{p.get_height()}', (p.get_x() +
p.get_width()/2, p.get_height()), ha='center', va='bottom',
color='black',fontsize=12)

plt.xticks(rotation=90,fontsize=15)
plt.title('Distribution of Listings Across Neighborhoods')
plt.xlabel('Neighbourhood',fontsize=12)
plt.ylabel('Number of Listings',fontsize=12)
plt.show()
```



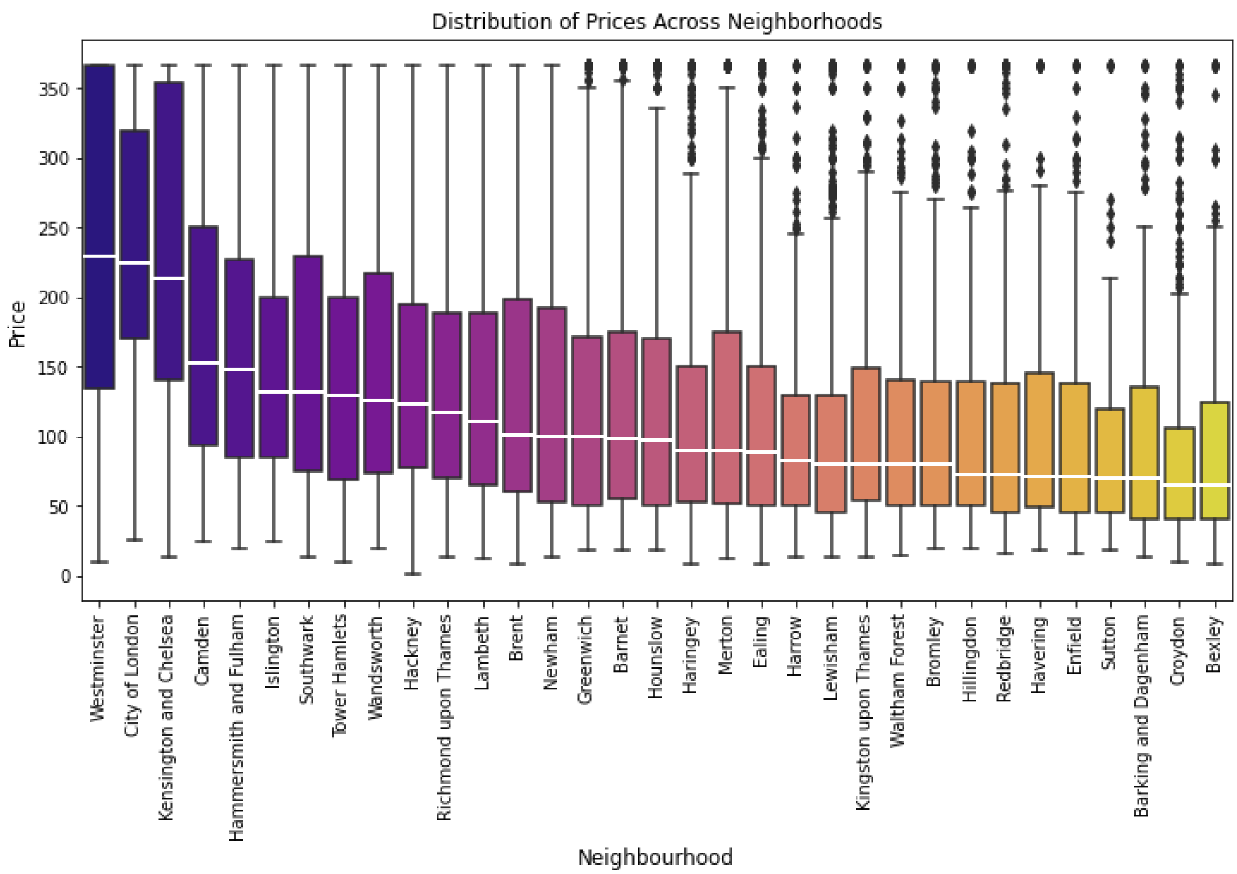
Top 3 most saturared neighborhoods: -Westminster -Tower Hamlets -Hackney

It would be interesting to consider neighbourhoods in the middle range (under 2000 properties) for business opportunities: -Brent -Lewisham -Haringey -Barnet -Ealing -Greenwich -Waltham Forest -Richmond Upon Thames -Kingston Upon Thames

---

Visualisation 2: What is the most expensive neighbourhood?

```
# Create a boxplot to visualize the distribution of prices across
neighborhoods

# Set the color for the median line
medianprops = dict(linewidth=2, color='white')

plt.figure(figsize=(12, 6))

labels_2 = sns.boxplot(x='neighbourhood', y='price', data=df,
order=df.groupby('neighbourhood')
['price'].median().sort_values(ascending=False).index,palette="plasma"
,medianprops=medianprops)

# Add count labels to each bar
for p in labels_2.patches:
    labels_2.annotate(f'{p.get_height()}', (p.get_x() +
p.get_width()/2, p.get_height()), ha='center', va='bottom',
color='black',fontsize=12)


plt.xticks(rotation=90)
plt.title('Distribution of Prices Across Neighborhoods')
plt.xlabel('Neighbourhood',fontsize=12)
plt.ylabel('Price',fontsize=12)
plt.show()
```



The top 3 most expensive neighbourhoods are: -City of London -Kensington and Chelsea -Westminster

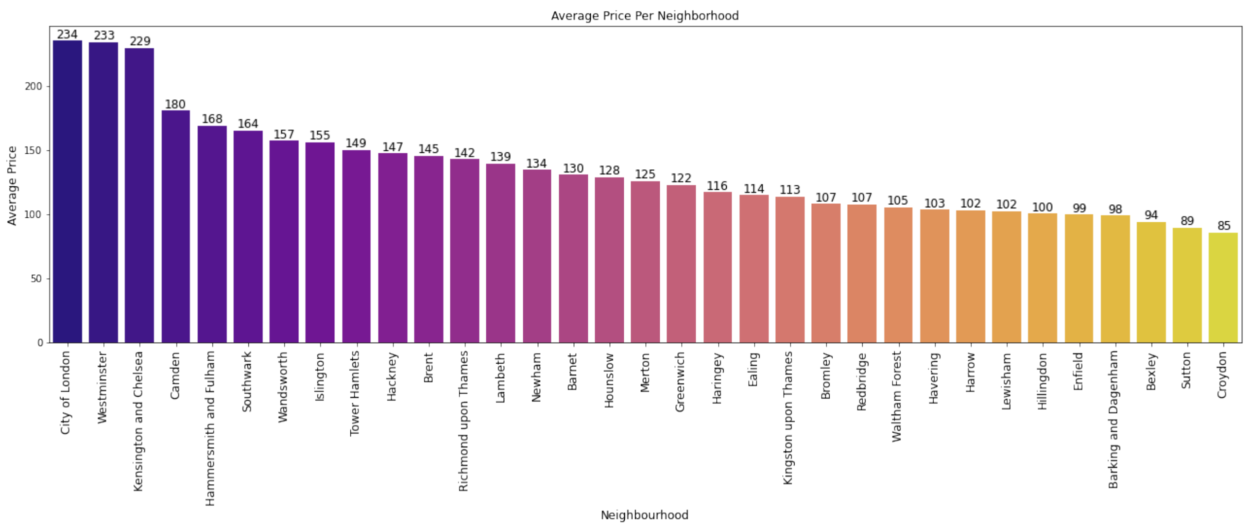
The majority of neighborhoods' data present potential outliers far away from the median and interquartile range. An interesting group of neighborhoods with median price per night ranging between 90£-125£ excluding the most saturated ones are: -Richmond upon Thames -Islington -Wandsworth -Merton -Brent -Houslow -Greenwich -Barnet -Kingston Upon Thames

---

Visualisation 3: Average price per neighbourhood

```
# Calculate the average price per neighborhood
avg_price_neighborhood = df.groupby('neighbourhood')
['price'].mean().sort_values(ascending=False)

# Plot the average price per neighborhood
plt.figure(figsize=(22, 6))

labels_3 = sns.barplot(x=avg_price_neighborhood.index,
y=avg_price_neighborhood.values,palette="plasma")

# Add count labels to each bar
for p in labels_3.patches:
```

```
    labels_3.annotate(f'{int(p.get_height())}', (p.get_x() +
p.get_width()/2, p.get_height()), ha='center', va='bottom',
color='black',fontsize=12)


plt.xticks(rotation=90,fontsize=12)
plt.title('Average Price Per Neighborhood')
plt.xlabel('Neighbourhood',fontsize=12)
plt.ylabel('Average Price',fontsize=12)
plt.show()
```



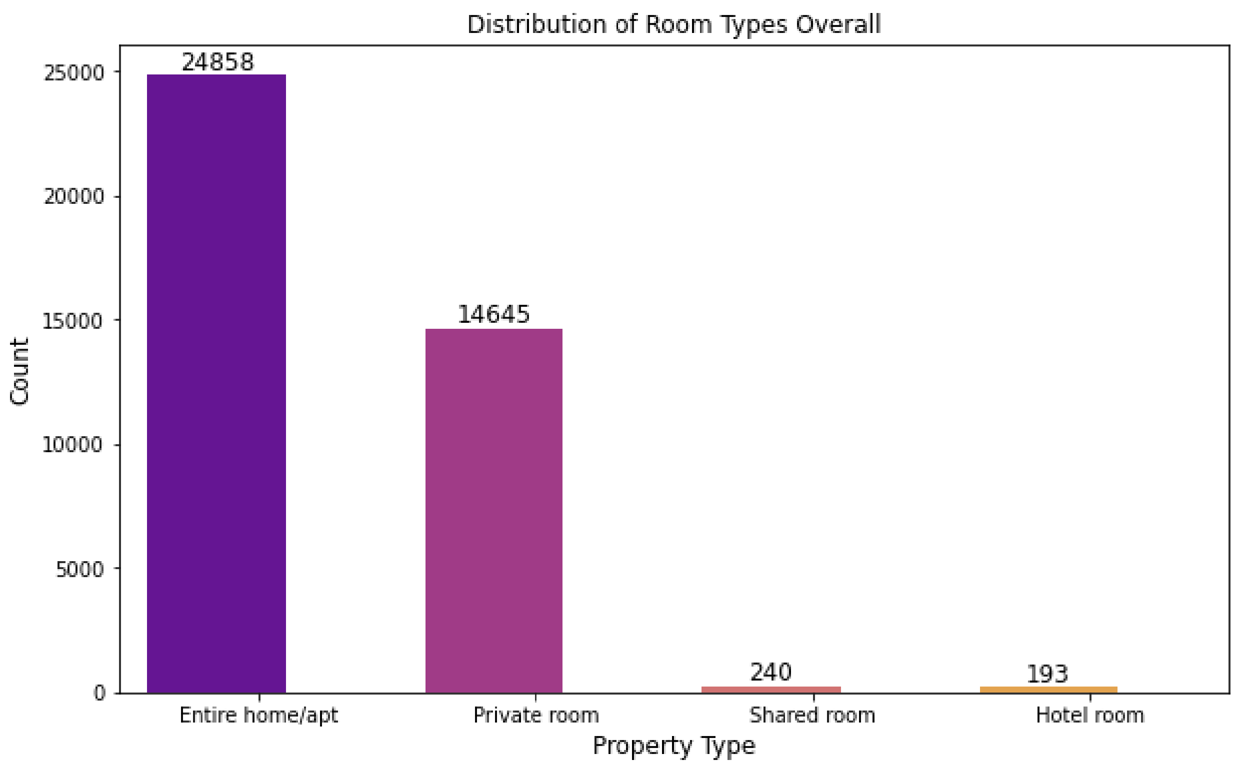
Top 3 average price per neighbourhood around 230£: -City of London -Kensington and Chelsea -Westminster

The average price in all other neighbourhoods drops from 180£ to 85£

Visualisation 4: What is the most popular room type overall

```
# Calculate the distribution of room types
room_type_distribution = df['room_type'].value_counts()

# Plot a bar chart to visualize the distribution of room types
plt.figure(figsize=(10, 6))

labels_4 = sns.barplot(x=room_type_distribution.index,
y=room_type_distribution.values,palette="plasma")

bar_width = 0.5
# Add count labels to each bar
for p in labels_4.patches:
    p.set_width(bar_width)
    labels_4.annotate(f'{int(p.get_height())}', (p.get_x() +
```

```
p.get_width()/2, p.get_height()), ha='center', va='bottom',
color='black',fontsize=12)

plt.title('Distribution of Room Types Overall')
plt.xlabel('Property Type',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.xticks(rotation=0)
plt.show()
```

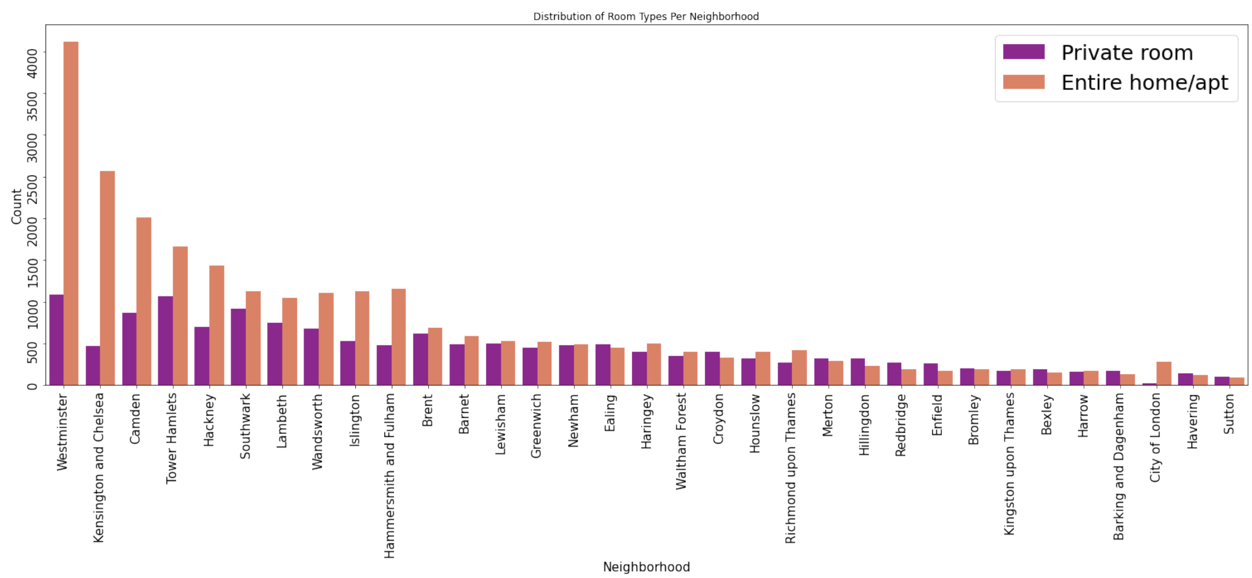


-Approx 63% of listings are Entire properties while almost 37% are private rooms within a property -Shared rooms and Hotel rooms could be ignored in the analysis.

Visualisation 5: What is the most popular room type per neighbourhood?

```
# Excluding 'Shared room' and 'Hotel room' from the DataFrame
df = df[df['room_type'].isin(['Entire home/apt', 'Private room'])]

# Create a grouped bar chart to visualize the distribution of room
types per neighborhood
plt.figure(figsize=(26, 8))

labels_5 = sns.countplot(x='neighbourhood', hue='room_type', data=df,
order=df['neighbourhood'].value_counts().index,palette="plasma")
```

```
plt.xticks(rotation=90, fontsize=15)
plt.yticks(rotation=90, fontsize=15)
plt.title('Distribution of Room Types Per Neighborhood')
plt.xlabel('Neighborhood',fontsize=15)
plt.ylabel('Count',fontsize=15)
plt.legend(loc='upper right', fontsize=25)
plt.show()
```



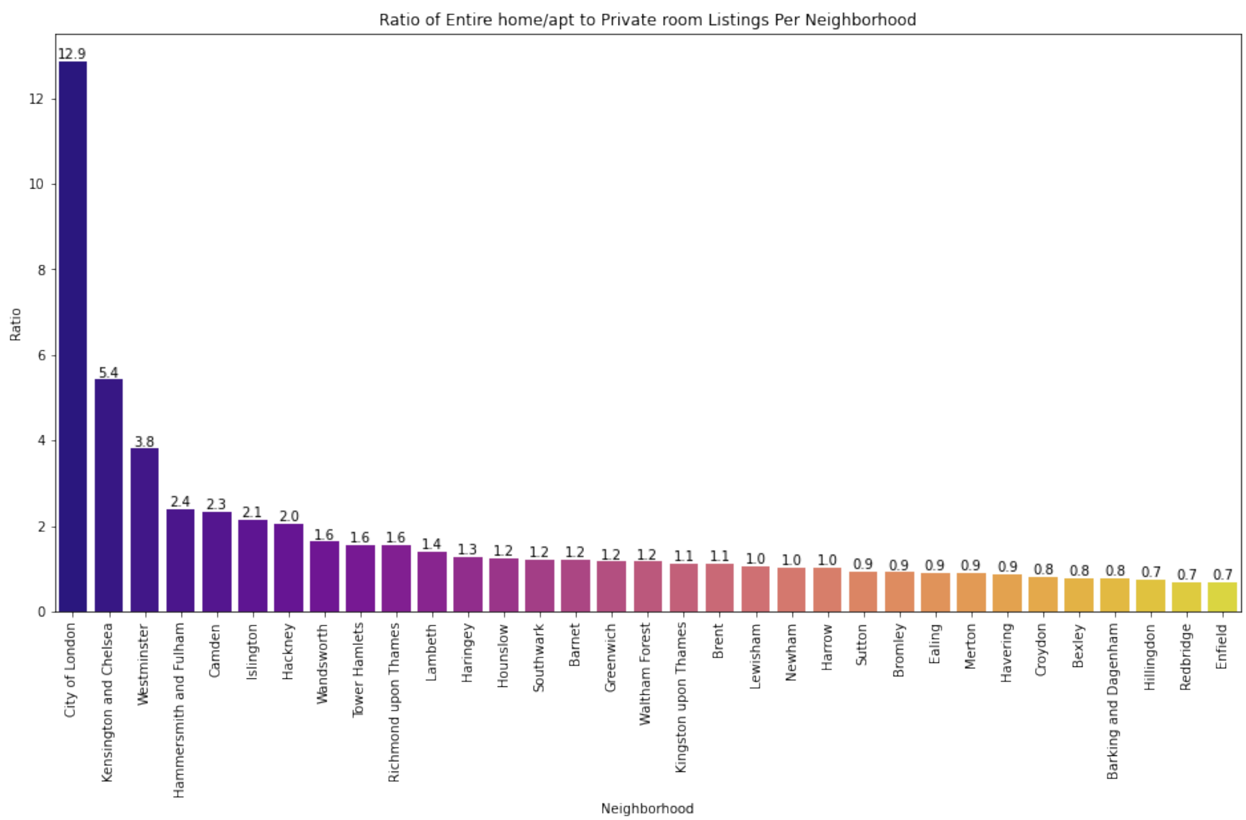
This visual presents what type of listing is most popular per neighbourhood only considering Entire home vs Private Room where Entire Homes are more common in the most relevant neighbourhoods

---

Visualisation 6: Entire Home vs Private Room ratio per neighbourhood

```
# Calculate the counts of 'Entire home/apt' and 'Private room'
listings per neighborhood
room_type_counts = df.groupby(['neighbourhood',
'room_type']).size().unstack(fill_value=0)

# Calculate the ratio of 'Entire home/apt' to 'Private room' listings
room_type_counts['ratio_entire_home_to_private'] =
room_type_counts['Entire home/apt'] / room_type_counts['Private room']

# Sort the neighborhoods by the ratio in descending order
sorted_ratio =
room_type_counts['ratio_entire_home_to_private'].sort_values(ascending
=False).index

# Plot the ratio per neighborhood
plt.figure(figsize=(16, 8))
```

```
room_type_counts_labels = sns.barplot(x=room_type_counts.index,
y='ratio_entire_home_to_private', data=room_type_counts,
order=sorted_ratio, palette="plasma")

# Add count labels to each bar
for p in room_type_counts_labels.patches:
    room_type_counts_labels.annotate(f'{round(p.get_height(),1)}',
(p.get_x() + p.get_width()/2, p.get_height()), ha='center',
va='bottom', color='black',fontsize=10)

plt.xticks(rotation=90)
plt.title('Ratio of Entire home/apt to Private room Listings Per
Neighborhood')
plt.xlabel('Neighborhood')
plt.ylabel('Ratio')
plt.show()
```



This visual shows the ratio between Entire Homes and Private Rooms.

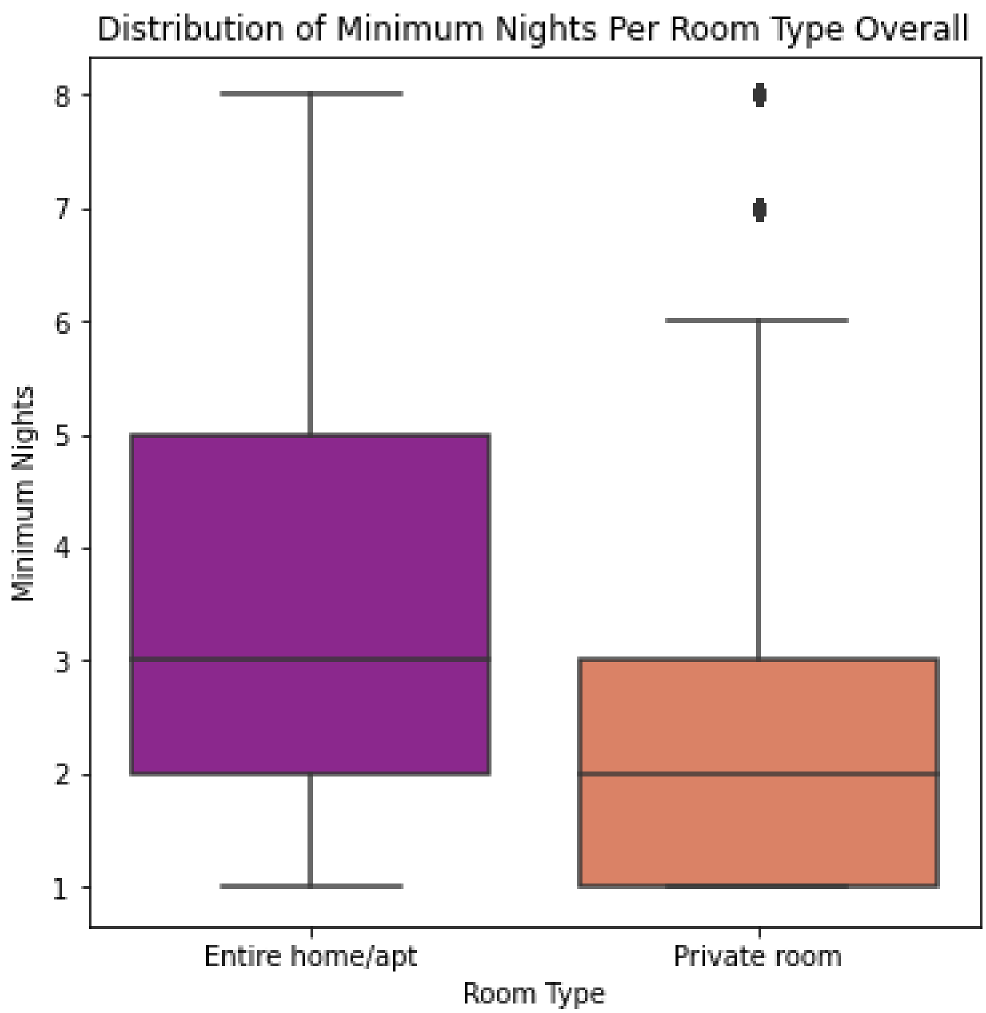
In the most expensive neighbourhoods: -City of London -Kensington and Chelsea -Westminster

Private Rooms are rarer, and Entire Homes dominate the listings.

In all other neighbourhoods on London, their presence is quite even, therefore Entire Homes and Private Rooms might work as well.

Visualisation 7: What is the min night per room type overall?

```
# Create a boxplot to visualize the distribution of minimum nights per
room type
plt.figure(figsize=(6, 6))

# Calculate the median minimum nights for each room type
median_min_nights = df.groupby('room_type')
['minimum_nights'].median().sort_values(ascending=False).index

sns.boxplot(x='room_type', y='minimum_nights', data=df,
palette='plasma', showfliers=True, order=median_min_nights)


plt.title('Distribution of Minimum Nights Per Room Type Overall')
plt.xlabel('Room Type')
plt.ylabel('Minimum Nights')
plt.show()
```



Even though Entire Homes are more expensive, the may be booked by groups, are more popular on the listings and are required to be booked by 3 nights median compared to 2 nights for the private rooms

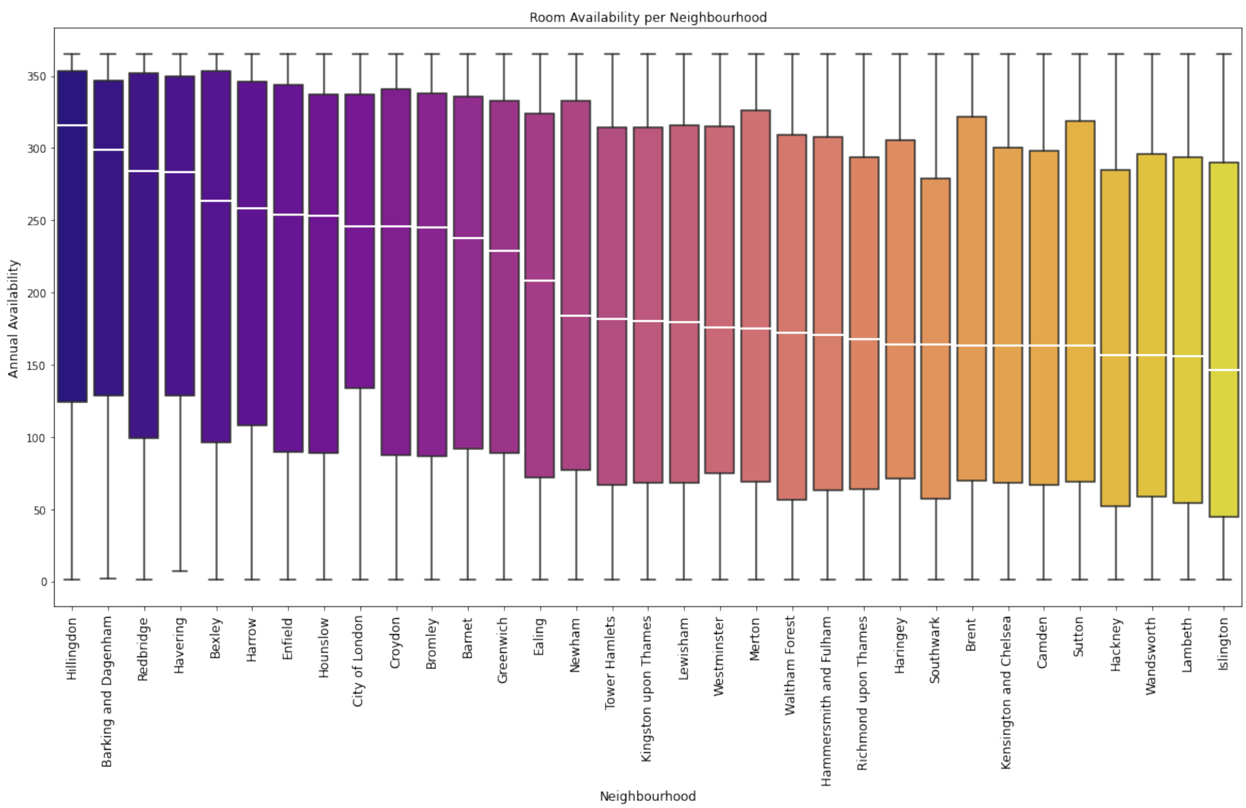
Ranges are: 2 to 5 nights for Entire Homes 1 to 3 nights for Private Rooms

---

Visualisation 8: Room availability per neighbourhood

```
print("Analysis of availability")

sorted_df = df.groupby("neighbourhood")
["availability_365"].median().sort_values(ascending=False).index

# Set the color for the median line
medianprops = dict(linewidth=2, color='white')

# Create a boxplot to visualize the distribution of availability_365
```

```
per neighborhood
plt.figure(figsize=(20, 10))
plot_av = sns.boxplot(data=df, x="neighbourhood",
y="availability_365", palette='plasma',
order=sorted_df,medianprops=medianprops)

plt.title('Room Availability per Neighbourhood')
plt.xticks(rotation=90,fontsize = 12)
plt.xlabel("Neighbourhood",fontsize = 12)
plt.ylabel("Annual Availability",fontsize = 12)
plt.show()

Analysis of availability
```



Neighbourhoods of interest with the most median availability: -Harrow -Enfield -Hounslow -Barnet -Greenwich

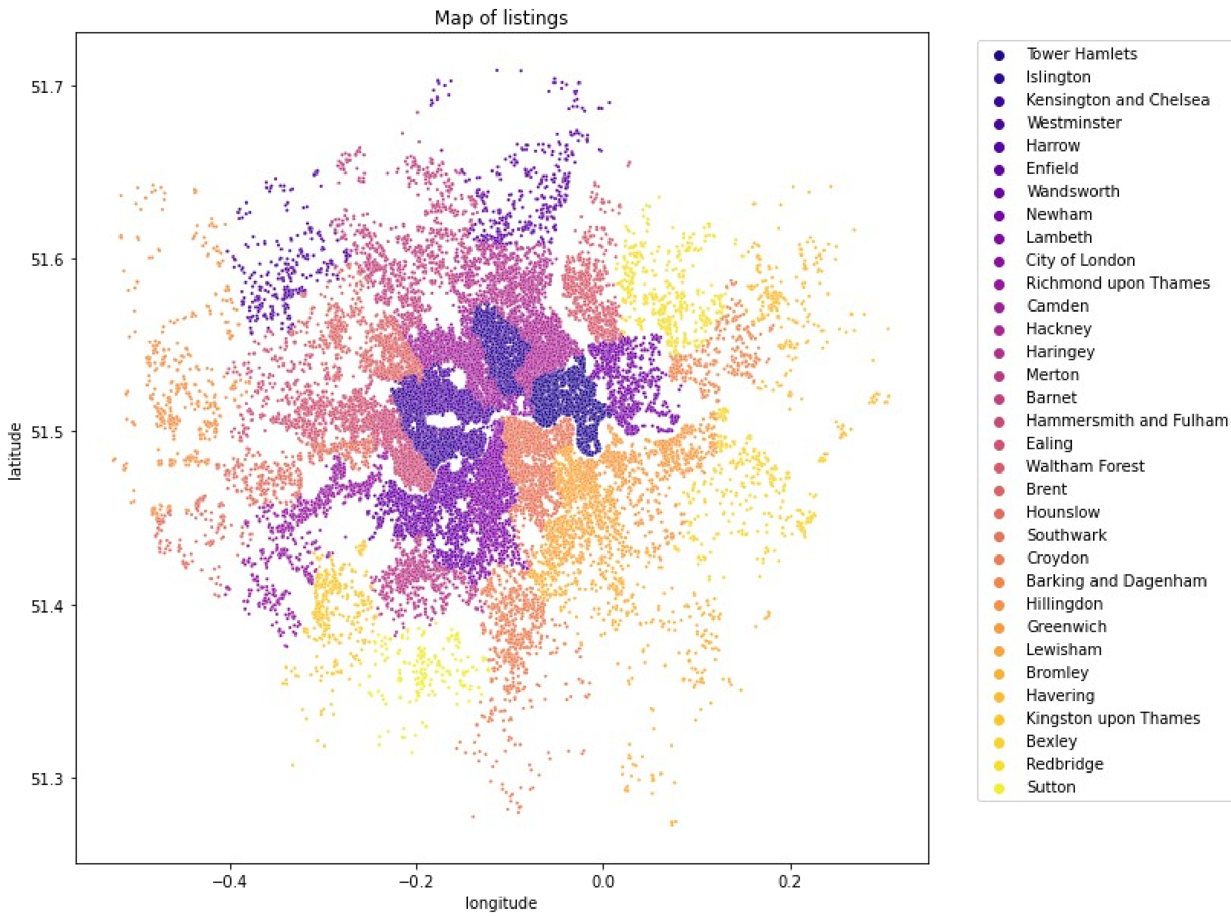
Neighbourhoods of interest with least the median availability: -Hackney -Islington -Lambeth -Wandsworth

---

Visualisation 9: Listed properties on the map

```
plt.figure(figsize=(10, 10))
```

```
plot_2 = sns.scatterplot(x=df.longitude, y=df.latitude,
hue=df.neighbourhood, palette='plasma', s=5)

plt.title('Map of listings')

plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

plt.ioff()
plt.show()
```

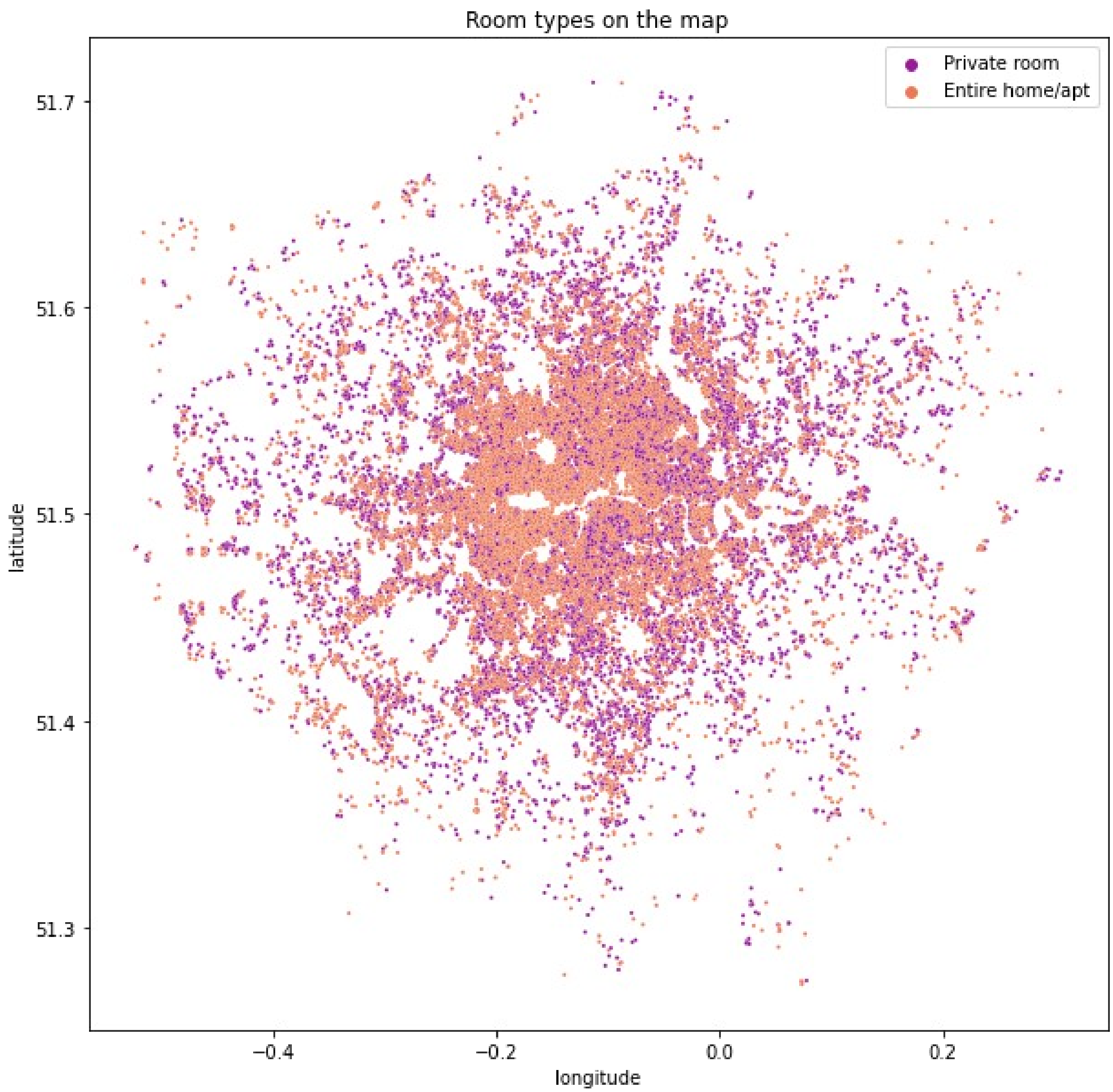


This visual provides insights into the most saturated neighbourhoods in London where we can see that in the outer circle the listings are more spread out whereas the neighbourhoods within the inner circle have the largest supply of properties.

Visualisation 10: Room types on the map

```
plt.figure(figsize=(10, 10))

plot_3 = sns.scatterplot(x=df.longitude, y=df.latitude,
hue=df.room_type, palette='plasma', s=5)
```

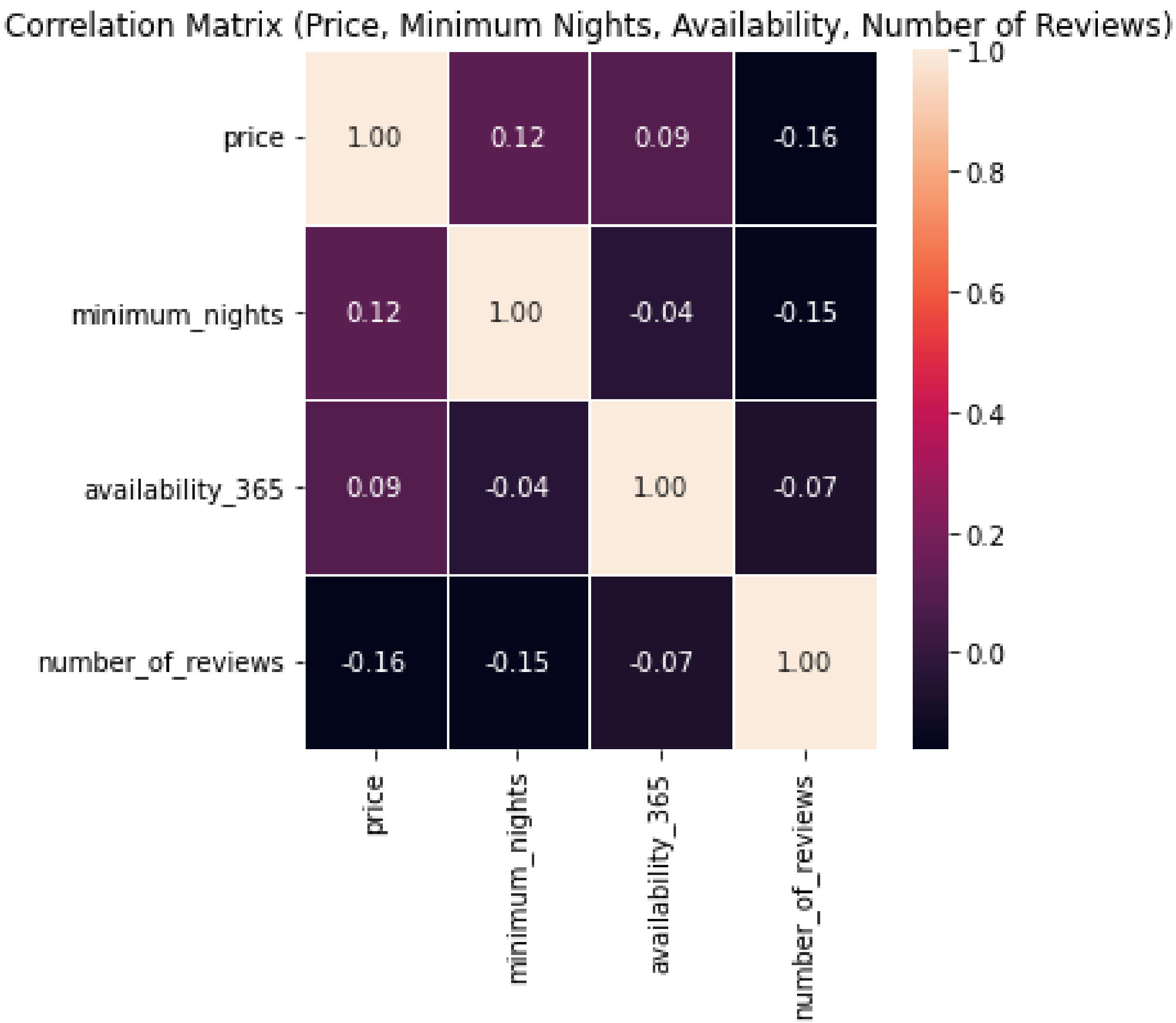
```
plt.title('Room types on the map')
plt.legend(loc='upper right', fontsize=10)
plt.show()
```



This visual suggests a higher concentration of Entire Homes within the inner areas of the city and a more even distribution of Private room vs Entire Homes towards the outer areas of the city.

Visualisation 11: Correlation Analysis:

```
# Relevant columns for correlation analysis
selected_columns = ['price', 'minimum_nights', 'availability_365',
'number_of_reviews']

# Subset of the DataFrame with selected columns
selected_df = df[selected_columns]

# Compute correlation matrix
correlation_matrix = selected_df.corr()

# Display heatmap for correlation matrix
plt.figure(figsize=(5, 5))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix (Price, Minimum Nights, Availability,
Number of Reviews)')
plt.show()
```



Negative correlation factors:

-Price vs Number of reviews. Higher-priced listings tend to receive fewer reviews. -Minimum number of nights vs Number of reviews. Listings with longer minimum night stays tend to receive fewer reviews. -Availability vs Number of reviews. Listings with higher availability tend to receive fewer reviews. -Minimum number of nights vs Availability. Listings with longer minimum night stays tend to have lower availability.

Positive correlation factors:

-Price vs Availability. On average, higher-priced listings tend to have higher availability throughout the year. -Price vs Minimum number of nights. On average, higher-priced listings tend to have longer minimum night stays.

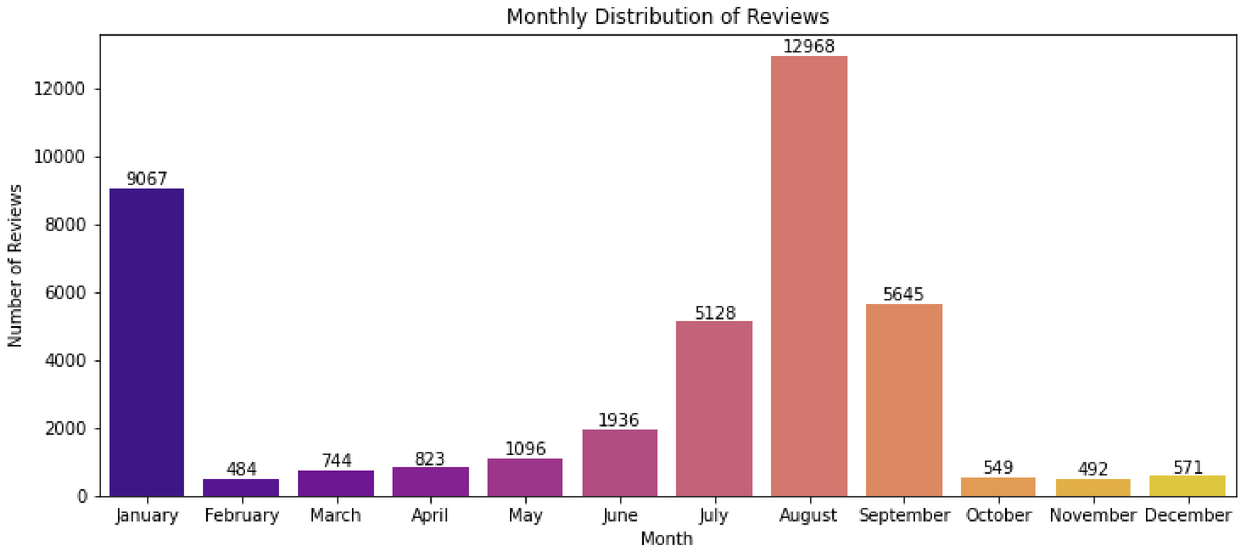
None of the correlation factors stand out significantly. All of them are very low which suggests that there seems to be no relevant correlation between the variables analysed

---

#Exploring Temporal Patterns

Visualisation 12: Seasonal Trends: Analyse if there are seasonal trends in the data by grouping the data by months or other time intervals.

```
# Define the order of months
month_mapping = {1: 'January', 2: 'February', 3: 'March', 4: 'April',
5: 'May', 6: 'June', 7: 'July', 8: 'August', 9: 'September', 10:
'October', 11: 'November', 12: 'December'}

# Extract month from 'last_review'
df['month'] = df['last_review'].dt.month.map(month_mapping)

plt.figure(figsize=(12, 5))

# Create a bar plot using Seaborn
labels6 = sns.countplot(x='month', data=df, palette='plasma',
order=month_mapping.values())

# Add count labels to each bar
for p in labels6.patches:
    labels6.annotate(f'{p.get_height()}', (p.get_x() +
p.get_width()/2, p.get_height()), ha='center', va='bottom',
color='black')

plt.title('Monthly Distribution of Reviews')
plt.xlabel('Month')
plt.ylabel('Number of Reviews')
plt.show()
```

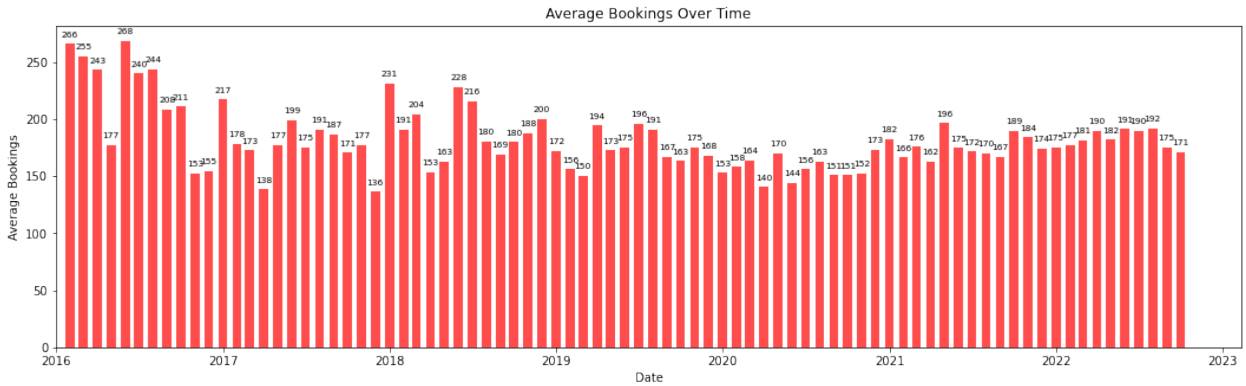


This visual shows the seasonality regarding customers reviews suggesting that the months with the most bookings are January after Christmas and August in Summer time.

Visualisation 13: Availability Over Time: Explore how availability changes over time.

```
# Filter data to include records from 2017 onwards
df_reviews_overtime = df[df['last_review'].dt.year >= 2016]

# Resample data to monthly frequency and calculate average
availability
average_bookings =
df_reviews_overtime.set_index('last_review').resample('M')
['availability_365'].mean()

# Set a wider bar width
bar_width = 20

# Plot as a bar chart with wider bars
plt.figure(figsize=(18, 5))
plt.bar(average_bookings.index, average_bookings, width=bar_width,
color='red', alpha=0.7)

# Add labels to the top of the bars
for date, value in average_bookings.items():
    plt.text(date, value + 5, f'{value:.0f}', ha='center',
va='bottom', fontsize=7.5)

plt.title('Average Bookings Over Time')
plt.xlabel('Date')
plt.ylabel('Average Bookings')
```

```
plt.xlim(pd.Timestamp('2016-01-01'))
plt.show()
```



The average monthly availability of properties follows a seasonal behaviour month-by-month confirming the seasonality seen on the previous graph.

Financial Estimates: Now we are going to look into renevue estimates per stay and per month and how this is distributed across neighbourhoods

```
# Calculate average price per night for entire homes and private rooms
in each neighborhood
avg_price_neighborhood = df.groupby(['neighbourhood', 'room_type'])
['price'].mean().reset_index()
avg_price_neighborhood['price'] =
avg_price_neighborhood['price'].round(0).astype(int)

# Calculate average minimum nights for entire homes and private rooms
in each neighborhood
avg_nights_neighborhood = df.groupby(['neighbourhood', 'room_type'])
['minimum_nights'].mean().reset_index()
avg_nights_neighborhood['minimum_nights'] =
avg_nights_neighborhood['minimum_nights'].round(0).astype(int)

# Calculate the average number of days available per year in each
neighborhood
avg_days_neighborhood = df.groupby(['neighbourhood', 'room_type'])
['availability_365'].mean().reset_index()
avg_days_neighborhood['availability_365'] =
avg_days_neighborhood['availability_365'].round(0).astype(int)
```

Calculated: average Price per night, average minimum nights per stay, average availability per year per room type per each neighbourhood

```
print("Merged all three dataframes")
```

```python
# Merge average figures Dataframes:
merge_part = pd.merge(avg_price_neighborhood,avg_nights_neighborhood,
on=['neighbourhood','room_type'])

neighborhood_figures = pd.merge(merge_part,avg_days_neighborhood,
on=['neighbourhood','room_type'])

neighborhood_figures.sample(4)
```

```
Merged all three dataframes
```

```
           neighbourhood        room_type  price  minimum_nights  \
5                 Bexley     Private room     54               2
47                Merton     Private room     68               2
53  Richmond upon Thames     Private room     76               2
14               Croydon  Entire home/apt    123               3

    availability_365
5                226
47               200
53               191
14               220
```

Calculated: -Average revenue per year -Average renevue per month using avg price x avg days available per year

```python
# Calculate average revenue per year and per month using avg price x
avg days available per year
neighborhood_figures['revenue_per_year'] =
(neighborhood_figures['price'] *
neighborhood_figures['availability_365']).round(0).astype(int)
neighborhood_figures['revenue_per_month'] =
(neighborhood_figures['revenue_per_year'] / 12).round(0).astype(int)

neighborhood_figures.head()
```

```
          neighbourhood        room_type  price  minimum_nights  \
0  Barking and Dagenham  Entire home/apt    147               3
1  Barking and Dagenham     Private room     61               3
2                Barnet  Entire home/apt    164               4
3                Barnet     Private room     90               2
4                Bexley  Entire home/apt    145               3

   availability_365  revenue_per_year  revenue_per_month
0               246             36162               3014
1               234             14274               1190
2               218             35752               2979
3               211             18990               1582
4               236             34220               2852
```

Calculated: -Average revenue per stay using avg price x avg minimum nights -Estimated Bookings per year

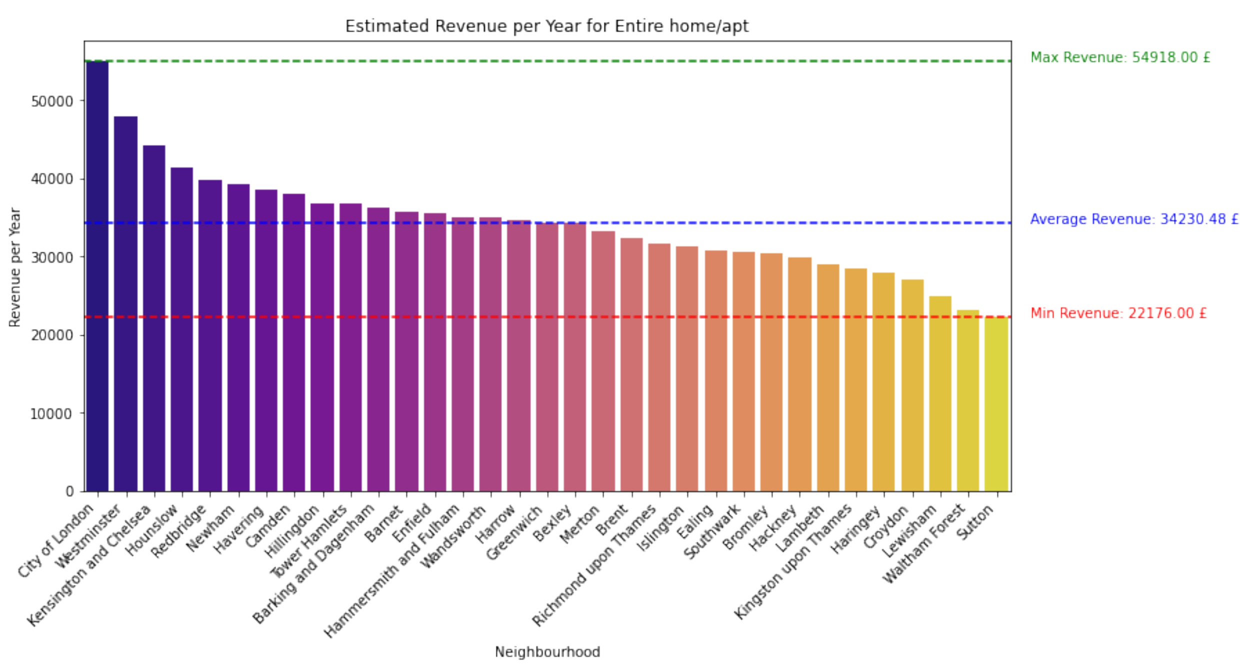
```
# Calculate average revenue per stay using avg price x avg minimum nights
neighborhood_figures['revenue_per_stay'] =
(neighborhood_figures['price'] *
neighborhood_figures['minimum_nights']).round(0).astype(int)
neighborhood_figures['estimated_bookings_year'] =
(neighborhood_figures['revenue_per_year'] /
neighborhood_figures['revenue_per_stay']).round(0).astype(int)

neighborhood_figures.sample(4)
```

```
   neighbourhood        room_type  price  minimum_nights
availability_365  \
56        Sutton  Entire home/apt    126               3
176
45      Lewisham     Private room     61               3
205
19       Enfield     Private room     64               3
216
64   Westminster  Entire home/apt    256               4
187

    revenue_per_year  revenue_per_month  revenue_per_stay  \
56             22176               1848               378
45             12505               1042               183
19             13824               1152               192
64             47872               3989              1024

    estimated_bookings_year
56                       59
45                       68
19                       72
64                       47
```

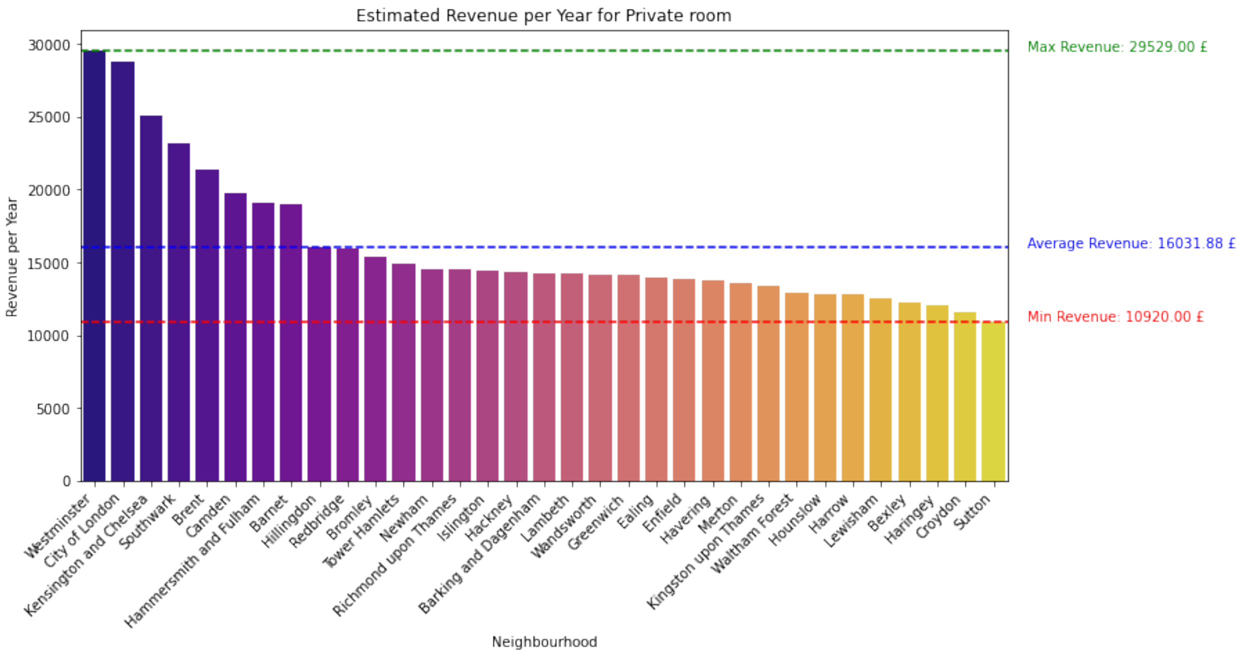
```
#Plots: Revenue per Year:

print("Visualisation 14: Estimated Revenue per Year for Entire
home/apt:")

plt.figure(figsize=(12, 6))

entire_home_year =
neighborhood_figures[neighborhood_figures['room_type'] == 'Entire
home/apt']
entire_home_year =
```

```
entire_home_year.sort_values(by='revenue_per_year',ascending=False)

sns.barplot(x='neighbourhood', y='revenue_per_year',
data=entire_home_year, palette='plasma')

#Adding lines to the plot:
max_revenue_entire_home_year =
entire_home_year['revenue_per_year'].max()
avg_revenue_entire_home_year =
entire_home_year['revenue_per_year'].mean()
min_revenue_entire_home_year =
entire_home_year['revenue_per_year'].min()

plt.axhline(max_revenue_entire_home_year, color='green',
linestyle='--', label='Max Revenue')
plt.axhline(avg_revenue_entire_home_year, color='blue',
linestyle='--', label='Average Revenue')
plt.axhline(min_revenue_entire_home_year, color='red', linestyle='--',
label='Min Revenue')

plt.text(len(entire_home_year) + 0.2, max_revenue_entire_home_year,
f'Max Revenue: {max_revenue_entire_home_year:.2f} £', color='green')
plt.text(len(entire_home_year) + 0.2, avg_revenue_entire_home_year,
f'Average Revenue: {avg_revenue_entire_home_year:.2f} £',
color='blue')
plt.text(len(entire_home_year) + 0.2, min_revenue_entire_home_year,
f'Min Revenue: {min_revenue_entire_home_year:.2f} £', color='red')

plt.title('Estimated Revenue per Year for Entire home/apt')
plt.xlabel('Neighbourhood')
plt.ylabel('Revenue per Year')
plt.xticks(rotation=45, ha='right')
plt.show()

print("Visualisation 15: Estimated Revenue per Year for Private
Room:")

plt.figure(figsize=(12, 6))

private_room_year =
neighborhood_figures[neighborhood_figures['room_type'] == 'Private
room']
private_room_year =
private_room_year.sort_values(by='revenue_per_year',ascending=False)

sns.barplot(x='neighbourhood', y='revenue_per_year',
data=private_room_year, palette='plasma')

#Adding lines to the plot:
max_revenue_private_room_year =
```

```
private_room_year['revenue_per_year'].max()
avg_revenue_private_room_year =
private_room_year['revenue_per_year'].mean()
min_revenue_private_room_year =
private_room_year['revenue_per_year'].min()

plt.axhline(max_revenue_private_room_year, color='green',
linestyle='--', label='Max Revenue')
plt.axhline(avg_revenue_private_room_year, color='blue',
linestyle='--', label='Average Revenue')
plt.axhline(min_revenue_private_room_year, color='red',
linestyle='--', label='Min Revenue')

plt.text(len(private_room_year) + 0.2, max_revenue_private_room_year,
f'Max Revenue: {max_revenue_private_room_year:.2f} £', color='green')
plt.text(len(private_room_year) + 0.2, avg_revenue_private_room_year,
f'Average Revenue: {avg_revenue_private_room_year:.2f} £',
color='blue')
plt.text(len(private_room_year) + 0.2, min_revenue_private_room_year,
f'Min Revenue: {min_revenue_private_room_year:.2f} £', color='red')

plt.title('Estimated Revenue per Year for Private room')
plt.xlabel('Neighbourhood')
plt.ylabel('Revenue per Year')
plt.xticks(rotation=45, ha='right')
plt.show()

Visualisation 14: Estimated Revenue per Year for Entire home/apt:
```



```
Visualisation 15: Estimated Revenue per Year for Private Room:
```

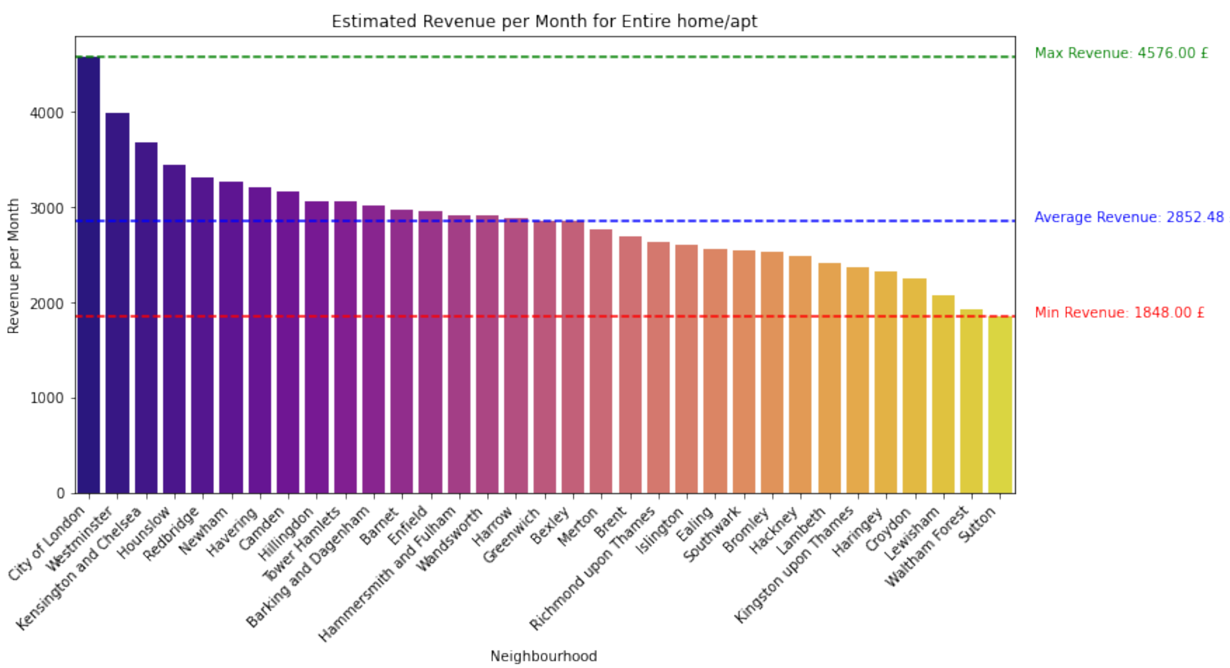



```
#Plots: Revenue per Month:

print("Visualisation 16: Estimated Revenue per Month for Entire
home/apt:")

plt.figure(figsize=(12, 6))


entire_home_month =
neighborhood_figures[neighborhood_figures['room_type'] == 'Entire
home/apt']
entire_home_month =
entire_home_month.sort_values(by='revenue_per_month',ascending=False)


sns.barplot(x='neighbourhood', y='revenue_per_month',
data=entire_home_month, palette='plasma')

#Adding lines to the plot:
max_revenue_entire_home_month =
entire_home_month['revenue_per_month'].max()
avg_revenue_entire_home_month =
entire_home_month['revenue_per_month'].mean()
min_revenue_entire_home_month =
entire_home_month['revenue_per_month'].min()

plt.axhline(max_revenue_entire_home_month, color='green',
```

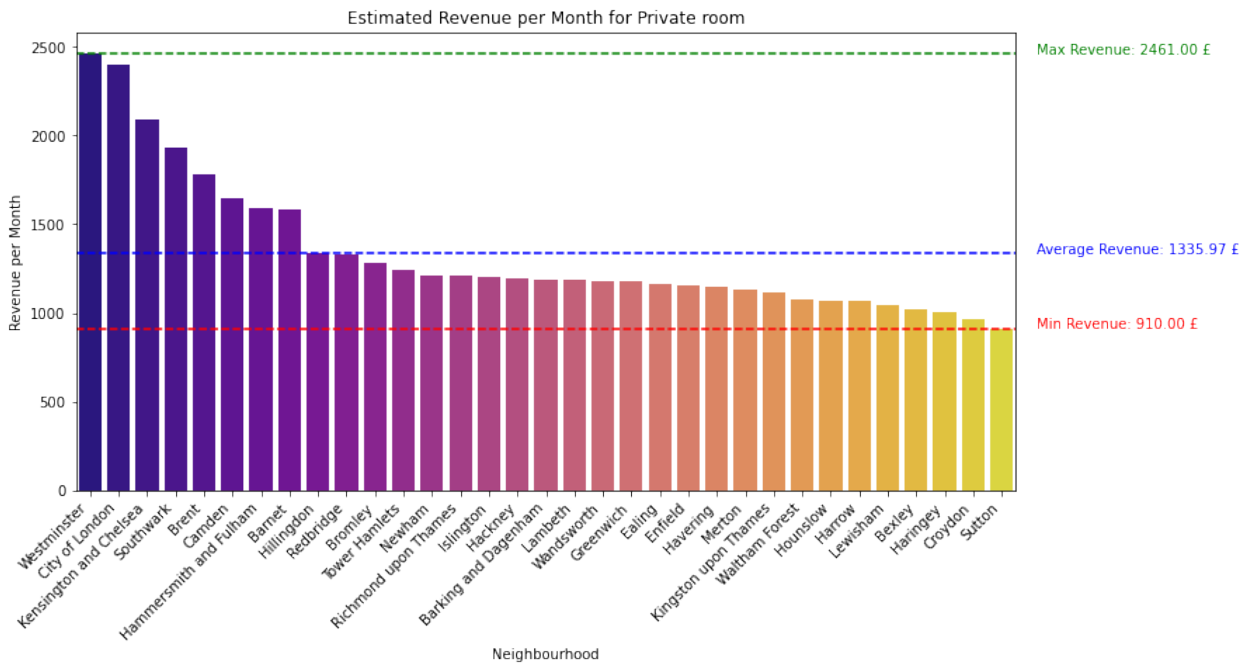
```
linestyle='--', label='Max Revenue')
plt.axhline(avg_revenue_entire_home_month, color='blue',
linestyle='--', label='Average Revenue')
plt.axhline(min_revenue_entire_home_month, color='red',
linestyle='--', label='Min Revenue')

plt.text(len(entire_home_month) + 0.2, max_revenue_entire_home_month,
f'Max Revenue: {max_revenue_entire_home_month:.2f} £', color='green')
plt.text(len(entire_home_month) + 0.2, avg_revenue_entire_home_month,
f'Average Revenue: {avg_revenue_entire_home_month:.2f} £',
color='blue')
plt.text(len(entire_home_month) + 0.2, min_revenue_entire_home_month,
f'Min Revenue: {min_revenue_entire_home_month:.2f} £', color='red')

plt.title('Estimated Revenue per Month for Entire home/apt')
plt.xlabel('Neighbourhood')
plt.ylabel('Revenue per Month')
plt.xticks(rotation=45, ha='right')
plt.show()

print("Visualisation 17: Estimated Revenue per Month for Private
room:")

plt.figure(figsize=(12, 6))


private_room_month =
neighborhood_figures[neighborhood_figures['room_type'] == 'Private
room']
private_room_month =
private_room_month.sort_values(by='revenue_per_month',ascending=False)

sns.barplot(x='neighbourhood', y='revenue_per_month',
data=private_room_month, palette='plasma')

#Adding lines to the plot:
max_revenue_private_room_month =
private_room_month['revenue_per_month'].max()
avg_revenue_private_room_month =
private_room_month['revenue_per_month'].mean()
min_revenue_private_room_month =
private_room_month['revenue_per_month'].min()

plt.axhline(max_revenue_private_room_month, color='green',
linestyle='--', label='Max Revenue')
plt.axhline(avg_revenue_private_room_month, color='blue',
linestyle='--', label='Average Revenue')
plt.axhline(min_revenue_private_room_month, color='red',
linestyle='--', label='Min Revenue')
```

```
plt.text(len(private_room_month) + 0.2,
max_revenue_private_room_month, f'Max Revenue:
{max_revenue_private_room_month:.2f} £', color='green')
plt.text(len(private_room_month) + 0.2,
avg_revenue_private_room_month, f'Average Revenue:
{avg_revenue_private_room_month:.2f} £', color='blue')
plt.text(len(private_room_month) + 0.2,
min_revenue_private_room_month, f'Min Revenue:
{min_revenue_private_room_month:.2f} £', color='red')

plt.title('Estimated Revenue per Month for Private room')
plt.xlabel('Neighbourhood')
plt.ylabel('Revenue per Month')
plt.xticks(rotation=45, ha='right')
plt.show()
```

Visualisation 16: Estimated Revenue per Month for Entire home/apt:



Visualisation 17: Estimated Revenue per Month for Private room:

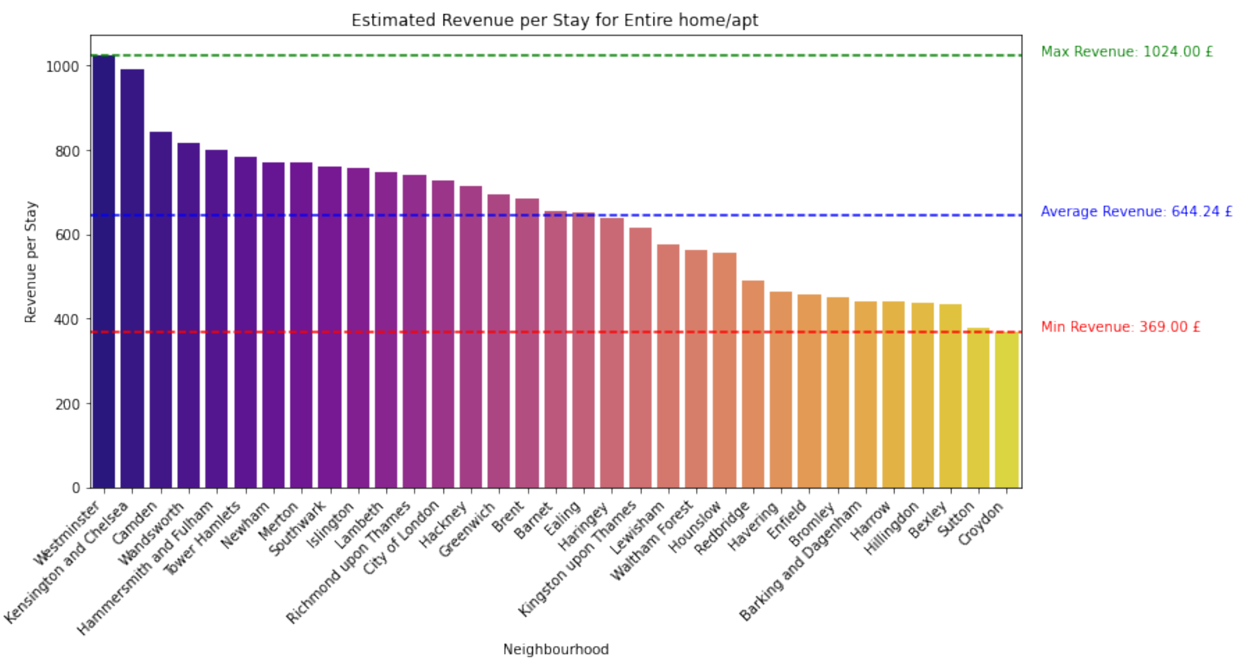



```
#Plots: Revenue per Stay:

print("Visualisation 18: Estimated Revenue per Stay for Entire
home/apt:")

plt.figure(figsize=(12, 6))

entire_home_stay =
neighborhood_figures[neighborhood_figures['room_type'] == 'Entire
home/apt']
entire_home_stay =
entire_home_stay.sort_values(by='revenue_per_stay',ascending=False)

sns.barplot(x='neighbourhood', y='revenue_per_stay',
data=entire_home_stay, palette='plasma')

#Adding lines to the plot:
max_revenue_entire_home_stay =
entire_home_stay['revenue_per_stay'].max()
avg_revenue_entire_home_stay =
entire_home_stay['revenue_per_stay'].mean()
min_revenue_entire_home_stay =
entire_home_stay['revenue_per_stay'].min()

plt.axhline(max_revenue_entire_home_stay, color='green',
linestyle='--', label='Max Revenue')
plt.axhline(avg_revenue_entire_home_stay, color='blue',
linestyle='--', label='Average Revenue')
plt.axhline(min_revenue_entire_home_stay, color='red', linestyle='--',
```

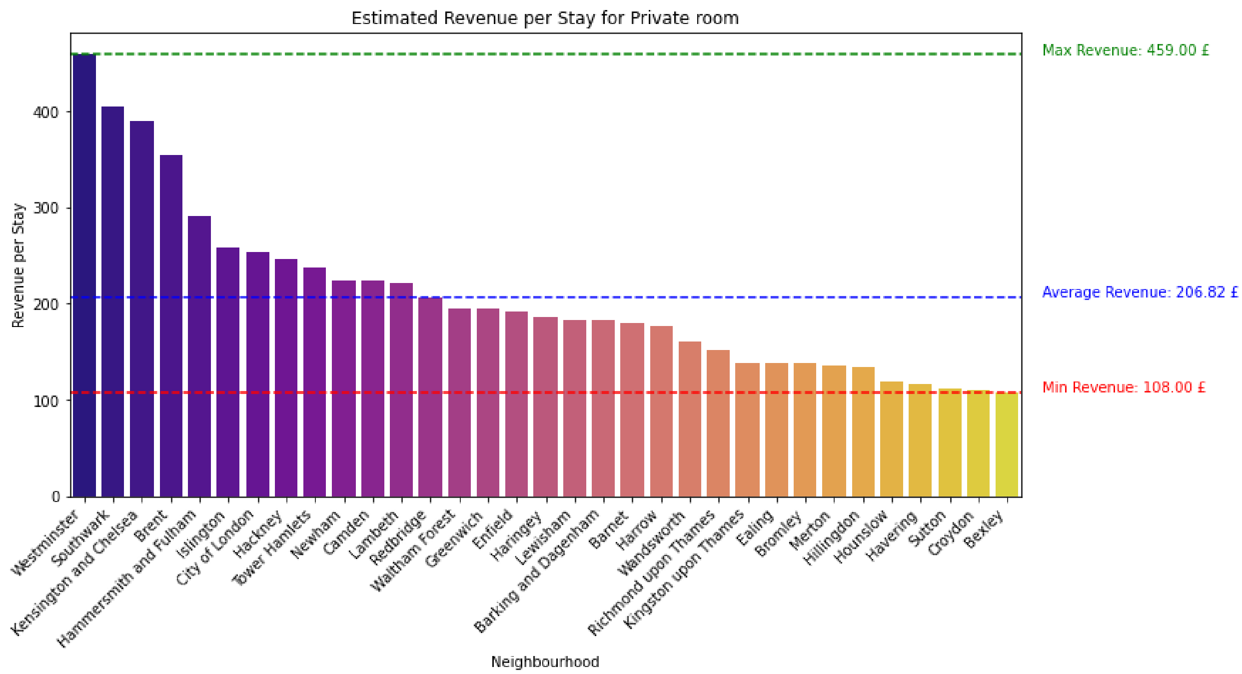
```
label='Min Revenue')

plt.text(len(entire_home_stay) + 0.2, max_revenue_entire_home_stay,
f'Max Revenue: {max_revenue_entire_home_stay:.2f} £', color='green')
plt.text(len(entire_home_stay) + 0.2, avg_revenue_entire_home_stay,
f'Average Revenue: {avg_revenue_entire_home_stay:.2f} £',
color='blue')
plt.text(len(entire_home_stay) + 0.2, min_revenue_entire_home_stay,
f'Min Revenue: {min_revenue_entire_home_stay:.2f} £', color='red')

plt.title('Estimated Revenue per Stay for Entire home/apt')
plt.xlabel('Neighbourhood')
plt.ylabel('Revenue per Stay')
plt.xticks(rotation=45, ha='right')
plt.show()


print("Visualisation 19: Estimated Revenue per Stay for Private
room:")

plt.figure(figsize=(12, 6))

private_room_stay =
neighborhood_figures[neighborhood_figures['room_type'] == 'Private
room']
private_room_stay =
private_room_stay.sort_values(by='revenue_per_stay',ascending=False)

sns.barplot(x='neighbourhood', y='revenue_per_stay',
data=private_room_stay, palette='plasma')

#Adding lines to the plot:
max_revenue_private_room_stay =
private_room_stay['revenue_per_stay'].max()
avg_revenue_private_room_stay =
private_room_stay['revenue_per_stay'].mean()
min_revenue_private_room_stay =
private_room_stay['revenue_per_stay'].min()

plt.axhline(max_revenue_private_room_stay, color='green',
linestyle='--', label='Max Revenue')
plt.axhline(avg_revenue_private_room_stay, color='blue',
linestyle='--', label='Average Revenue')
plt.axhline(min_revenue_private_room_stay, color='red',
linestyle='--', label='Min Revenue')

plt.text(len(private_room_stay) + 0.2, max_revenue_private_room_stay,
f'Max Revenue: {max_revenue_private_room_stay:.2f} £', color='green')
plt.text(len(private_room_stay) + 0.2, avg_revenue_private_room_stay,
f'Average Revenue: {avg_revenue_private_room_stay:.2f} £',
```

```
color='blue')
plt.text(len(private_room_stay) + 0.2, min_revenue_private_room_stay,
f'Min Revenue: {min_revenue_private_room_stay:.2f} £', color='red')


plt.title('Estimated Revenue per Stay for Private room')
plt.xlabel('Neighbourhood')
plt.ylabel('Revenue per Stay')
plt.xticks(rotation=45, ha='right')
plt.show()
```

Visualisation 18: Estimated Revenue per Stay for Entire home/apt:



Visualisation 19: Estimated Revenue per Stay for Private room:



Insights and Conclusions:

Insights:

The analysis uncovers several key patterns in London's Airbnb market. Notably, the dataset reveals a diverse range of hosts and properties, with 69351 unique IDs and 45229 distinct host IDs spanning 33 neighborhoods. Among property types, Entire Homes dominate the market, constituting around 60% of listings, followed by private rooms at nearly 40%. The analysis excludes shared rooms and hotel rooms due to their marginal market share.

In terms of pricing and popularity, the top three neighborhoods by listing frequency are Westminster, Tower Hamlets, and Hackney, while the most expensive areas, with a median nightly rate of £230, include City of London, Kensington and Chelsea, and Westminster. Furthermore, the data indicates a strong preference for Entire Homes in affluent neighborhoods, contrasting with a more balanced distribution in other areas. The minimum booking requirement also varies, with Entire Homes typically requiring a median stay of 3 nights compared to 2 nights for private rooms.

Trending patterns over time and Conclusions:

Seasonal trends emerge in customer bookings, with peaks observed in January post-Christmas and August during summer. This seasonality suggests opportunities for strategic rental planning, possibly combining short-term and mid-term strategies to optimize revenue. Despite fluctuations, the Airbnb market continues to grow, presenting viable business prospects. Notably, renting out Entire Homes could yield double the revenue of private rooms, making it a lucrative option, albeit with varying levels of operational commitment. Additionally, exploring underserved areas, such as Brent, Harrow, and Lewisham, could unveil untapped market potential. Ultimately, the profitability of each venture hinges on operational costs, warranting a detailed cost-benefit analysis for informed decision-making.