```
#IMPORT LIBRARIES

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import statsmodels.formula.api as smf
import re
import itertools
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
import numpy as np


#LOAD DATA

df = pd.read_csv('airbnb_Dallas.csv')


# Preprocessing
df['Property Type'] = df['Property Type'].replace({'Camper/rv':'Camper/RV'})
df['Property Type'] = df['Property Type'].replace({'Entire camper/RV':'Camper/RV'})
df['Property Type'] = df['Property Type'].replace({'Entire condominium':'Condominium'})
df['Property Type'] = df['Property Type'].replace({'Bed and breakfast':'Bed & Breakfast'})
df['Property Type'] = df['Property Type'].replace({'Entire bungalow':'Bungalow'})
df['Property Type'] = df['Property Type'].replace({'Entire place':'Entire house'})
df['Property Type'] = df['Property Type'].replace({'Entire guesthouse':'Guesthouse'})
df['Property Type'] = df['Property Type'].replace({'Entire house':'House'})
df['Property Type'] = df['Property Type'].replace({'Entire loft':'Loft'})
df['Property Type'] = df['Property Type'].replace({'Room in boutique hotel':'Boutique hotel'})
df['Property Type'] = df['Property Type'].replace({'Room in hostel':'Hostel'})
df['Property Type'] = df['Property Type'].replace({'Entire townhouse':'Townhouse'})
df['Property Type'] = df['Property Type'].replace({'Entire apartment':'Apartment'})


#NULL HANDLING

#Print columns with NA values
columns_with_nan = []
for column in df.columns:
    if df[column].isnull().any():
        columns_with_nan.append(column)
print("Columns with NaN values:", columns_with_nan)

#Replace NaN values in numeric and non-numeric columns
for column in df.columns:
    if pd.api.types.is_numeric_dtype(df[column]):
        df[column].fillna(0, inplace=True)
    else:
        df[column].fillna("NA", inplace=True)
print("DataFrame after imputing zeros for numeric columns and 'NA' for others:")
print(df)

#Check for NaN columns after imputation
columns_with_nan = []
for column in df.columns:
    if df[column].isnull().any():
        columns_with_nan.append(column)
print("Columns with NaN values:", columns_with_nan)
```

```
Columns with NaN values: ['rating_ave_pastYear', 'numReviews_pastYear', 'numCancel_pastYear', 'num_5_star_Rev_pastYear', 'pr
DataFrame after imputing zeros for numeric columns and 'NA' for others:
       Airbnb Host ID  Airbnb Property ID  City_x  superhost_period_all  \
0               18837                7273  Dallas                     5
1               18837                7273  Dallas                     6
2               18837                7273  Dallas                     7
3               18837                7273  Dallas                     8
4               18837                7273  Dallas                     9
...               ...                 ...     ...                   ...
48706         1498025            42777500  Dallas                    19
48707         1498025            42777500  Dallas                    20
48708        51662786            42780168  Dallas                    19
```

```
48709       51662786            42780168  Dallas                    20
48710      320800969            42809984  Dallas                    20

       scrapes_in_period Scraped Date  superhost_observed_in_period  \
0                      7    8/13/2016                             7
1                      3    11/9/2016                             3
2                      4     2/5/2017                             3
3                      8     5/4/2017                             6
4                      7     8/4/2017                             7
...                  ...          ...                           ...
48706                 45     2/4/2020                            45
48707                 20     5/5/2020                            20
48708                152     2/1/2020                           152
48709                129     5/1/2020                           129
48710                  7     5/4/2020                             7

       host_is_superhost_in_period  superhost_ratio  \
0                                0        0.000000
1                                1        1.000000
2                                1        1.000000
3                                1        1.000000
4                                1        1.000000
...                            ...             ...
48706                            0        0.000000
48707                            0        0.000000
48708                            1        1.000000
48709                            1        0.992248
48710                            1        1.000000

       prev_superhost_period_all  ...  prev_host_is_superhost2  \
0                              4  ...                        0
1                              5  ...                        0
2                              6  ...                        0
3                              7  ...                        0
4                              8  ...                        1
...                          ...  ...                      ...
48706                         18  ...                        0
48707                         19  ...                        0
48708                         18  ...                        0
48709                         19  ...                        0
48710                         19  ...                        0

       prev_year_superhosts  booked_days_period_city  revenue_period_city  \
0                         0                    20067              2182615
1                         0                    19879              2118369
2                         1                    25078              2902274
```

```
import pandas as pd
from tabulate import tabulate

# Assuming df is your DataFrame with a 'revenue' column
# Replace this with your actual DataFrame
# df = ...

# Calculate descriptive statistics for the 'revenue' column
revenue_stats = df['revenue'].describe().to_frame(name='Revenue Statistics')

# Display the descriptive statistics as a table using tabulate
table = tabulate(revenue_stats, headers='keys', tablefmt='pretty')

# Print the table
print(table)
```

```
+-------+--------------------+
|       | Revenue Statistics |
+-------+--------------------+
| count |      48711.0       |
| mean  | 2049.271622426146  |
|  std  | 3745.623263107884  |
|  min  |        0.0         |
|  25%  |        0.0         |
|  50%  |       787.0        |
|  75%  |       2759.0       |
|  max  |      134209.0      |
+-------+--------------------+
```

```
import pandas as pd
from tabulate import tabulate

# Assuming df is your DataFrame with a 'revenue' column
# Replace this with your actual DataFrame
# df = ...

# Calculate descriptive statistics for the 'revenue' column
revenue_stats = df['Max Guests'].describe().to_frame(name='Max Guests Statistics')

# Display the descriptive statistics as a table using tabulate
table = tabulate(revenue_stats, headers='keys', tablefmt='pretty')

# Print the table
print(table)
```

```
+-------+-----------------------+
|       | Max Guests Statistics |
+-------+-----------------------+
| count |        48711.0        |
| mean  |  3.9419227689844183   |
|  std  |  2.5689933720159845   |
|  min  |          0.0          |
|  25%  |          2.0          |
|  50%  |          4.0          |
|  75%  |          5.0          |
|  max  |         16.0          |
+-------+-----------------------+
```

```
import pandas as pd
from tabulate import tabulate

# Assuming df is your DataFrame with a 'revenue' column
# Replace this with your actual DataFrame
# df = ...

# Calculate descriptive statistics for the 'revenue' column
revenue_stats = df['Neighborhood'].describe().to_frame(name='Neighborhood Statistics')

# Display the descriptive statistics as a table using tabulate
table = tabulate(revenue_stats, headers='keys', tablefmt='pretty')

# Print the table
print(table)
```

```
+--------+-------------------------+
|        | Neighborhood Statistics |
+--------+-------------------------+
| count  |          48711          |
| unique |           18            |
|  top   |     Central Dallas      |
|  freq  |          25470          |
+--------+-------------------------+
```

```
import pandas as pd
from tabulate import tabulate

# Assuming df is your DataFrame with a 'revenue' column
# Replace this with your actual DataFrame
# df = ...

# Calculate descriptive statistics for the 'revenue' column
revenue_stats = df['Nightly Rate'].describe().to_frame(name='Nightly Rate Statistics')

# Display the descriptive statistics as a table using tabulate
table = tabulate(revenue_stats, headers='keys', tablefmt='pretty')

# Print the table
print(revenue_stats)
```

```
       Nightly Rate Statistics
count             48711.000000
mean                157.173051
std                 169.886401
min                   1.000000
25%                  69.000000
50%                 107.500000
```

```
75%                 185.125000
max                1900.000000
```

```
#CONVERT DATE TO OBTAIN QUARTERS

#Convert the "Scraped_Date" column to datetime format
df['Scraped Date'] = pd.to_datetime(df['Scraped Date'])

#Create a new column for quarters based on the month
df['Quarter'] = df['Scraped Date'].dt.month.apply(lambda x: 'Q1' if 1 <= x <= 3 else 'Q2' if 4 <= x <= 6 else 'Q3' if 7 <= x <=

#Print the updated DataFrame
print(df['Quarter'])
```

```
0        Q3
1        Q4
2        Q1
3        Q2
4        Q3
         ..
48706    Q1
48707    Q2
48708    Q1
48709    Q2
48710    Q2
Name: Quarter, Length: 48711, dtype: object
```

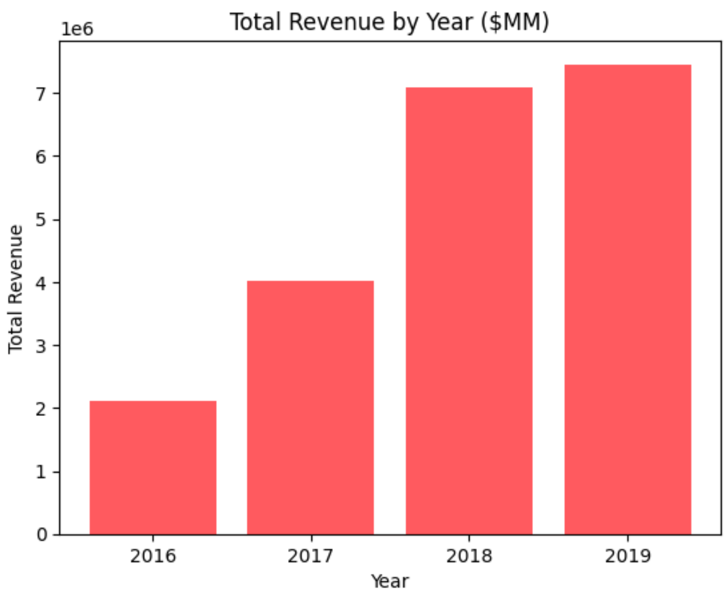
```
#CONVERT DATE TO OBTAIN YEAR

#Convert the "Scraped_Date" column to datetime format
df['Scraped Date'] = pd.to_datetime(df['Scraped Date'])

#Create a new column for the year
df['Year'] = df['Scraped Date'].dt.year

#Print the updated DataFrame
print(df['Year'])
```

```
0        2016
1        2016
2        2017
3        2017
4        2017
         ...
48706    2020
48707    2020
48708    2020
48709    2020
48710    2020
Name: Year, Length: 48711, dtype: int64
```

```
#ADJUST YEARS FOR TIME SERIES

def adjust_year(row):
    if row['Quarter'] == 'Q4':
        if row['Year'] == 2017:
            return 2017
        elif row['Year'] == 2016:
            return 2016
        elif row['Year'] == 2018:
            return 2018
        elif row['Year'] == 2019:
            return 2019
        elif row['Year'] == 2020:
            return 2020
    return 0  # Return 0 if conditions are not met

# Apply the function to create the new column 'Year_Adjusted'
df['Year_Adjusted'] = df.apply(adjust_year, axis=1)

# Filter to keep rows where 'Year_Adjusted' is not equal to 0
df = df[df['Year_Adjusted']!=0]
```

```
#Before Outliers REV BY YEAR

# Calculate total revenue by Year_Adjusted
total_revenue_by_year = df.groupby('Year_Adjusted')['revenue'].sum()

# Display total revenue by year
print(total_revenue_by_year)

# Create a bar chart
plt.bar(total_revenue_by_year.index, total_revenue_by_year.values, color='#FF5A5F')

# Set specific years to display on the x-axis
specific_years = [2016, 2017, 2018, 2019]  # Update with the years you want to display
plt.xticks(specific_years)

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Total Revenue')
plt.title('Total Revenue by Year ($MM)')

# Disable logarithmic scaling on the y-axis
plt.yscale('linear')

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

```
Year_Adjusted
2016    2118369.0
2017    4024237.0
2018    7093302.0
2019    7456803.0
Name: revenue, dtype: float64
```




```
#Remove outliers for revenue
# Assuming df is your DataFrame
# Calculate Z-scores for the 'revenue' column
#z_scores = np.abs((df['revenue'] - df['revenue'].mean()) / df['revenue'].std())

# Set a threshold for Z-score (e.g., 3)
#threshold = 3

# Remove rows with Z-scores above the threshold
#df = df[z_scores < threshold]
```

```
#Remove outliers for Nightly Rate
# Assuming df is your DataFrame
# Calculate Z-scores for the 'revenue' column
#z_scores = np.abs((df['Nightly Rate'] - df['Nightly Rate'].mean()) / df['Nightly Rate'].std())

# Set a threshold for Z-score (e.g., 3)
#threshold = 3

# Remove rows with Z-scores above the threshold
#df = df[z_scores < threshold]


#SEGMENT "Rating Overall" and "prev_Rating Overall" by deciles

# Round 'Rating Overall' to the nearest 10s place and replace the values in the DataFrame
df['Rating Overall'] = df['Rating Overall'].apply(lambda x: round(x, -1))

# Round 'prev_Rating Overall' to the nearest 10s place and replace the values in the DataFrame
df['prev_Rating Overall'] = df['prev_Rating Overall'].apply(lambda x: round(x, -1))



#Set Variables

# Load the CSV file without headers
dfvar = pd.read_csv('cat.cont.variables2.csv', header=None)

# Iterate through each row in dfvar and remove columns as specified
for index, row in dfvar.iterrows():
    variable_name = row[0]  # Assuming the variable name is in the first column
    action = row[1]  # Assuming the action (e.g., 'REMOVE') is in the second column

    if action == 'REMOVE':
        # Check if the variable exists in the DataFrame before removing
        if variable_name in df.columns:
            df.drop(variable_name, axis=1, inplace=True)
        else:
            print(f"Column '{variable_name}' not found in the DataFrame.")

# Assuming dfvar is loaded and structured as described: variable names in the first column, action in the second column

continuous_variables = []  # List to store continuous variable names
categorical_variables = []  # List to store categorical variable names

# Iterate through each row in dfvar and update variable types as specified
for index, row in dfvar.iterrows():
    variable_name = row[0]  # Assuming the variable name is in the first column
    action = row[1]  # Assuming the action (e.g., 'Cont' or 'Cat') is in the second column

    if action == 'Cont':
        # Check if the variable exists in the DataFrame before changing its type
        if variable_name in df.columns:
            df[variable_name] = df[variable_name].astype(float)  # Convert to float or numeric type for continuous
            continuous_variables.append(variable_name)  # Add to continuous_variables list
        else:
            print(f"Column '{variable_name}' not found in the DataFrame.")
    elif action == 'Cat':
        # Check if the variable exists in the DataFrame before changing its type
        if variable_name in df.columns:
            df[variable_name] = df[variable_name].astype(str)  # Convert to string type for categorical
            categorical_variables.append(variable_name)  # Add to categorical_variables list
        else:
            print(f"Column '{variable_name}' not found in the DataFrame.")

# After looping through all rows in dfvar, columns have been designated as continuous or categorical in 'df'
# Continuous variables are stored in continuous_variables list, and categorical variables are stored in categorical_variables li




# Remove specified variables from the list
variables_to_remove = ['Quarter', 'Year']

for variable in variables_to_remove:
    if variable in categorical_variables:
        categorical_variables.remove(variable)
```

```
# Exclude 'Quarter' and 'Year' columns
df = df.drop(columns=['Quarter', 'Year'])

#Drop NA neighborhoods
#df = df[df['Neighborhood'] != 'NA']

#Rename Hillside/University Neighborhood values
df['Neighborhood'] = df['Neighborhood'].replace(
    {'Hillside/University Meadows/Ridge Wood Park/North Stonewall Terrace': 'Hillside etc.'}
)

#Rename NA to other for neighborhood
df['Neighborhood'] = df['Neighborhood'].replace(
    {'NA': 'Other'}
)



#Exclude revnues of 0
#df = df[df['revenue'] != 0]

#Exclude availible days less than 21
#df = df[df['available_days'] > 20]

#Exclude booked days equal to zero
#df = df[df['booked_days'] != 0]

#Exclude hotel listing types
#df = df[df['Listing Type'] != 'Hotel room']


#Optimized guest term
# Assuming 'df' is your DataFrame
#df['Optimized Guests for BRs'] = np.where(df['Max Guests'] < 3 * df['Bedrooms'], 0, 1)
#categorical_variables.append('Optimized Guests for BRs')
#print(categorical_variables)


#Drop Airbnb Property ID and Airbnb Host ID
#df = df.drop(columns=['Airbnb Property ID', 'Airbnb Host ID'
                      #])


#Create copies of original dataframe. After Quarter and year are removed
#df2=df.copy()
# Exclude 'Quarter' and 'Year' columns
#df2 = df2.drop(columns=['Instantbook Enabled','prev_Instantbook Enabled'
                      #])


#df3=df.copy()
# Exclude 'Quarter' and 'Year' columns
#df3 = df3.drop(columns=['Instantbook Enabled','prev_Instantbook Enabled'
                      #])
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have a DataFrame named df with the required columns
# For demonstration purposes, let's assume the columns are 'Listing Type', 'booked_days', and 'Airbnb Property ID'

# Group by 'Listing Type' and calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df.agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']

print('Overall Booking ratio')
print(ratio_df['Booking Ratio'])

df_private_room = df[df['Listing Type'] == 'Private room']
# Group by 'Listing Type' and calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_private_room.agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']

print('Private room Booking ratio')
print(ratio_df['Booking Ratio'])




df_shared_room = df[df['Listing Type'] == 'Shared room']
# Group by 'Listing Type' and calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_shared_room.agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']

print('Shared room Booking ratio')
print(ratio_df['Booking Ratio'])



df_house = df[df['Listing Type'] == 'Entire home/apt']
# Group by 'Listing Type' and calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_house.agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']

print('Entire home/apt Booking ratio')
print(ratio_df['Booking Ratio'])

df_bedrooms = df[(df['Bedrooms'] != 0) & (df['Max Guests'] != 0)]
# Assuming 'df' is your DataFrame
MG_BR_df = df_bedrooms[df_bedrooms['Max Guests'] < 3 * df_bedrooms['Bedrooms']]

# Display count of instances meeting the condition
print("Count of instances where Max Guests < 3 * Bedrooms:",len(MG_BR_df)) #Short term listings can have up to 3 guests per bedr



import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame named MG_BR_df with a column 'Neighborhood'
# For demonstration purposes, let's assume the column name is 'Neighborhood'

# Group by 'Neighborhood' and calculate the count
neighborhood_count_df = MG_BR_df['Neighborhood'].value_counts().reset_index()
neighborhood_count_df.columns = ['Neighborhood', 'Count']

# Sort the DataFrame by 'Count' in descending order
sorted_count_df = neighborhood_count_df.sort_values(by='Count', ascending=False)

# Set the size of the figure
plt.figure(figsize=(10, 6))

# Display the sorted DataFrame as a table
table = plt.table(cellText=sorted_count_df.values,
                  colLabels=['Neighborhood', 'Count'],
                  cellLoc='center',
                  loc='center',
                  colColours=['#FF5A5F', '#FF5A5F'])

for (i, j), cell in table.get_celld().items():
    if i == 0:
        cell.set_text_props(fontweight='bold', color='#484848')  # Header text is bold and white
```

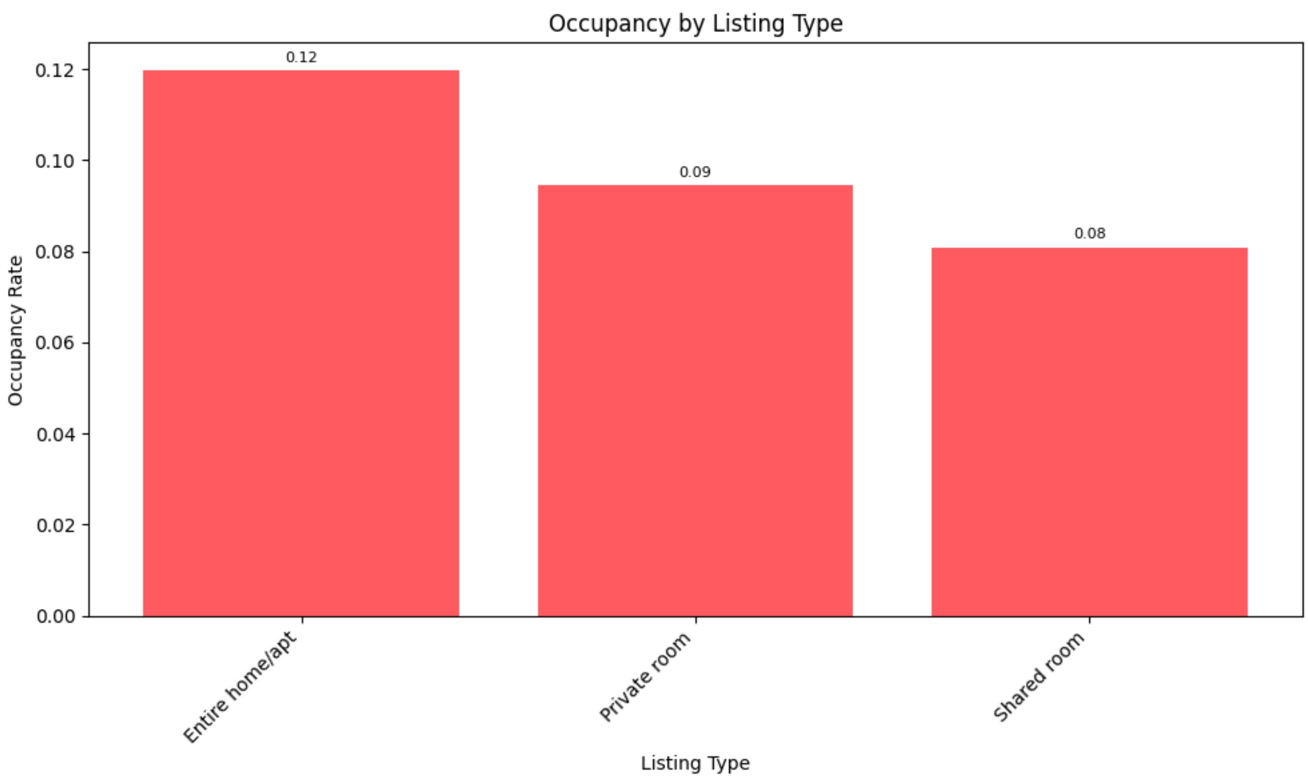
```
    else:
        cell.set_text_props(color='#484848')

# Set the font size of the table
table.auto_set_font_size(False)
table.set_fontsize(10)

# Hide axis
plt.axis('off')

plt.title('Neighborhood Count in MG_BR_df')
plt.show()
```

```
Overall Booking ratio
0.11283071735448814
Private room Booking ratio
0.09465217477470846
Shared room Booking ratio
0.08091310635969198
Entire home/apt Booking ratio
0.11983633271085842
Count of instances where Max Guests < 3 * Bedrooms: 6404
```

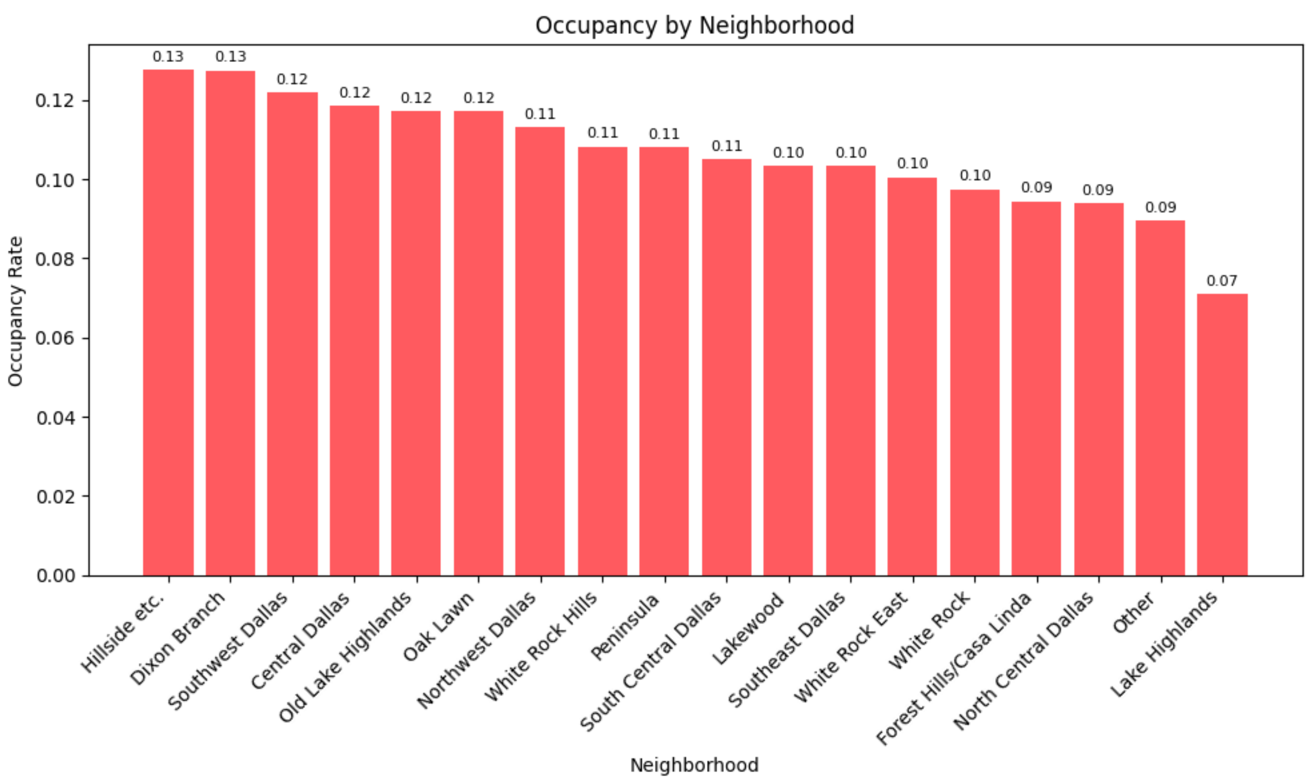
Neighborhood Count in MG_BR_df

Neighborhood	Count
Central Dallas	2515
North Central Dallas	802
Southwest Dallas	771
Oak Lawn	618
Other	520
Northwest Dallas	333
Lake Highlands	205
Southeast Dallas	174
South Central Dallas	133
Forest Hills/Casa Linda	68
Hillside etc.	50
Lakewood	50
White Rock East	39
Old Lake Highlands	37
White Rock Hills	36
Dixon Branch	35
White Rock	12
Peninsula	6

```
import matplotlib.pyplot as plt
import pandas as pd

# Assuming you have a DataFrame named df with the required columns
# For demonstration purposes, let's assume the columns are 'Listing Type', 'booked_days', and 'Airbnb Property ID'
df_hotel = df[df['Listing Type'] != 'Hotel room']

# Group by 'Listing Type' and calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_hotel.groupby('Listing Type').agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']

# Plot the bar chart with data labels
plt.figure(figsize=(10, 6))
bars = plt.bar(ratio_df['Booking Ratio'].sort_values(ascending=False).index,
               ratio_df['Booking Ratio'].sort_values(ascending=False),
               color='#FF5A5F')

# Add data labels using annotate
for bar, label in zip(bars, ratio_df['Booking Ratio'].sort_values(ascending=False)):
    plt.annotate(f'{label:.2f}',
                 xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                 xytext=(0, 3),  # 3 points vertical offset
                 textcoords="offset points",
                 ha='center', va='bottom', fontsize=8)

plt.title('Occupancy by Listing Type')
plt.xlabel('Listing Type')
plt.ylabel('Occupancy Rate')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



```
import matplotlib.pyplot as plt
import pandas as pd

# Assuming you have a DataFrame named df with the required columns
# For demonstration purposes, let's assume the columns are 'Neighborhood', 'booked_days', and 'Airbnb Property ID'

# Group by 'Neighborhood' and calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_hotel.groupby('Neighborhood').agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Neighborhood Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']

# Plot the bar chart with data labels
plt.figure(figsize=(10, 6))
bars = plt.bar(ratio_df['Neighborhood Booking Ratio'].sort_values(ascending=False).index,
               ratio_df['Neighborhood Booking Ratio'].sort_values(ascending=False),
               color='#FF5A5F')

# Add data labels using annotate
for bar, label in zip(bars, ratio_df['Neighborhood Booking Ratio'].sort_values(ascending=False)):
    plt.annotate(f'{label:.2f}',
                 xy=(bar.get_x() + bar.get_width() / 2, bar.get_height()),
                 xytext=(0, 3),  # 3 points vertical offset
                 textcoords="offset points",
                 ha='center', va='bottom', fontsize=8)

plt.title('Occupancy by Neighborhood')
plt.xlabel('Neighborhood')
plt.ylabel('Occupancy Rate')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame named df with the required columns
# For demonstration purposes, let's assume the columns are 'Listing Type', 'booked_days', 'Airbnb Property ID', and 'Neighborhoc

# Filter the DataFrame for rows where 'Listing Type' is 'Private room'
df_private_room = df[df['Listing Type'] == 'Private room']

# Group by 'Neighborhood' and 'Listing Type', then calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_private_room.groupby(['Neighborhood', 'Listing Type']).agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']
ratio_df['Booking Ratio']=ratio_df['Booking Ratio'].round(4)

# Sort the DataFrame by 'Booking Ratio' in descending order
sorted_ratio_df = ratio_df.sort_values(by='Booking Ratio', ascending=False)

# Set the size of the figure
plt.figure(figsize=(10, 6))

# Display the sorted DataFrame as a table
table = plt.table(cellText=sorted_ratio_df.reset_index()[['Neighborhood', 'Booking Ratio']].values,
                  colLabels=['Neighborhood', 'Occupancy Rate'],
                  cellLoc='center',
                  loc='center',
                  colColours=['#FF5A5F', '#FF5A5F'])

for (i, j), cell in table.get_celld().items():
    if i == 0:
        cell.set_text_props(fontweight='bold', color='#484848')  # Header text is bold and white
    else:
        cell.set_text_props(color='#484848')


# Set the font size of the table
table.auto_set_font_size(False)
table.set_fontsize(10)

# Hide axis
plt.axis('off')

plt.title('Occupancy for Private Rooms by Neighborhood')
plt.show()
```

Occupancy for Private Rooms by Neighborhood

Neighborhood	Occupancy Rate
Dixon Branch	0.2217
Old Lake Highlands	0.1217
Hillside etc.	0.1212
Southeast Dallas	0.1177
White Rock Hills	0.114
Southwest Dallas	0.1067
Central Dallas	0.1016
Oak Lawn	0.0957
Northwest Dallas	0.0948
South Central Dallas	0.0888
Other	0.087
North Central Dallas	0.0817
White Rock East	0.0807
Forest Hills/Casa Linda	0.0799
Lakewood	0.0738
Lake Highlands	0.0478
White Rock	0.0333
Peninsula	0.0

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame named df with the required columns
# For demonstration purposes, let's assume the columns are 'Listing Type', 'booked_days', 'Airbnb Property ID', and 'Neighborhoc

# Filter the DataFrame for rows where 'Listing Type' is 'Shared room'
df_shared_room = df[df['Listing Type'] == 'Shared room']

# Group by 'Neighborhood' and 'Listing Type', then calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_shared_room.groupby(['Neighborhood', 'Listing Type']).agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']
ratio_df['Booking Ratio']=ratio_df['Booking Ratio'].round(4)

# Sort the DataFrame by 'Booking Ratio' in descending order
sorted_ratio_df = ratio_df.sort_values(by='Booking Ratio', ascending=False)

# Set the size of the figure
plt.figure(figsize=(10, 6))

# Display the sorted DataFrame as a table
table = plt.table(cellText=sorted_ratio_df.reset_index()[['Neighborhood', 'Booking Ratio']].values,
                  colLabels=['Neighborhood', 'Occupancy Rate'],
                  cellLoc='center',
                  loc='center',
                  colColours=['#FF5A5F', '#FF5A5F'])

for (i, j), cell in table.get_celld().items():
    if i == 0:
        cell.set_text_props(fontweight='bold', color='#484848')  # Header text is bold and white
    else:
        cell.set_text_props(color='#484848')


# Set the font size of the table
table.auto_set_font_size(False)
table.set_fontsize(10)

# Hide axis
plt.axis('off')

plt.title('Occupancy for Shared Rooms by Neighborhood')
plt.show()
```

Occupancy for Shared Rooms by Neighborhood

Neighborhood	Occupancy Rate
Northwest Dallas	0.1202
Southwest Dallas	0.1041
Oak Lawn	0.082
South Central Dallas	0.0721
Central Dallas	0.0636
North Central Dallas	0.0445
Southeast Dallas	0.0366
Other	0.0198
Lake Highlands	0.0157
White Rock Hills	0.0041

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame named df with the required columns
# For demonstration purposes, let's assume the columns are 'Listing Type', 'booked_days', 'Airbnb Property ID', and 'Neighborhoc

# Filter the DataFrame for rows where 'Listing Type' is 'Shared room'
df_home = df[df['Listing Type'] == 'Entire home/apt']

# Group by 'Neighborhood' and 'Listing Type', then calculate the ratio of booked days to the count of Airbnb Property ID
ratio_df = df_home.groupby(['Neighborhood', 'Listing Type']).agg({'booked_days': 'sum', 'available_days': 'sum'})
ratio_df['Booking Ratio'] = ratio_df['booked_days'] / ratio_df['available_days']
ratio_df['Booking Ratio']=ratio_df['Booking Ratio'].round(4)

# Sort the DataFrame by 'Booking Ratio' in descending order
sorted_ratio_df = ratio_df.sort_values(by='Booking Ratio', ascending=False)

# Set the size of the figure
plt.figure(figsize=(10, 6))

# Display the sorted DataFrame as a table
table = plt.table(cellText=sorted_ratio_df.reset_index()[['Neighborhood', 'Booking Ratio']].values,
                  colLabels=['Neighborhood', 'Occupancy Rate'],
                  cellLoc='center',
                  loc='center',
                  colColours=['#FF5A5F', '#FF5A5F'])

for (i, j), cell in table.get_celld().items():
    if i == 0:
        cell.set_text_props(fontweight='bold', color='#484848')  # Header text is bold and white
    else:
        cell.set_text_props(color='#484848')


# Set the font size of the table
table.auto_set_font_size(False)
table.set_fontsize(10)

# Hide axis
plt.axis('off')

plt.title('Occupancy for Entire home/apt by Neighborhood')
plt.show()
```

Occupancy for Entire home/apt by Neighborhood

Neighborhood	Occupancy Rate
White Rock	0.1393
South Central Dallas	0.1346
Southwest Dallas	0.1328
Hillside etc.	0.129
Oak Lawn	0.1245
Northwest Dallas	0.1216
Central Dallas	0.1214
Peninsula	0.1178
White Rock East	0.1145
Old Lake Highlands	0.1129
Lakewood	0.1097
Dixon Branch	0.1088
Southeast Dallas	0.1069
North Central Dallas	0.1065
White Rock Hills	0.1045
Forest Hills/Casa Linda	0.099
Lake Highlands	0.0985
Other	0.0961

```
#Remove outliers for revenue
# Assuming df is your DataFrame
# Calculate Z-scores for the 'revenue' column
z_scores = np.abs((df['revenue'] - df['revenue'].mean()) / df['revenue'].std())

# Set a threshold for Z-score (e.g., 3)
threshold = 3

# Remove rows with Z-scores above the threshold
df = df[z_scores < threshold]


#Remove outliers for Nightly Rate
# Assuming df is your DataFrame
# Calculate Z-scores for the 'revenue' column
z_scores = np.abs((df['Nightly Rate'] - df['Nightly Rate'].mean()) / df['Nightly Rate'].std())

# Set a threshold for Z-score (e.g., 3)
threshold = 3

# Remove rows with Z-scores above the threshold
df = df[z_scores < threshold]


#Exclude revnues of 0
#df = df[df['revenue'] != 0]

#Exclude availible days less than 21
df = df[df['available_days'] > 20]

#Exclude booked days equal to zero
#df = df[df['booked_days'] != 0]

#Exclude hotel listing types
df = df[df['Listing Type'] != 'Hotel room']


#After Outliers REV BY YEAR

# Calculate total revenue by Year_Adjusted
total_revenue_by_year_no_outliers = df.groupby('Year_Adjusted')['revenue'].sum()

# Display total revenue by year
print(total_revenue_by_year_no_outliers)
```

```
Year_Adjusted
2016    1763821.0
2017    3355013.0
2018    5779937.0
2019    6124143.0
Name: revenue, dtype: float64
```

```
#Drop Airbnb Property ID and Airbnb Host ID
df = df.drop(columns=['Airbnb Property ID', 'Airbnb Host ID'
                       ])


#Create copies of original dataframe. After Quarter and year are removed
df2=df.copy()
# Exclude 'Quarter' and 'Year' columns
df2 = df2.drop(columns=['Instantbook Enabled','prev_Instantbook Enabled'
                       ])


df3=df.copy()
# Exclude 'Quarter' and 'Year' columns
df3 = df3.drop(columns=['Instantbook Enabled','prev_Instantbook Enabled'
                       ])
```

```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error

# Assuming 'df' is your DataFrame with one-hot encoded columns
df = pd.get_dummies(df, columns=categorical_variables, drop_first=True)

# Separate features (X) and target variable (y)
X = df.drop('revenue', axis=1)
y = df['revenue']

# Add a constant term to the features matrix
X = sm.add_constant(X)

# Split the data into a 75% training set and a 25% validation set
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.25, random_state=42)

# Fit the linear regression model on the training set
model = sm.OLS(y_train, X_train).fit()

# Print the model summary on the training set
print("Training Set Model Summary:")
print(model.summary())

# Make predictions on the training set and validation set
y_train_pred = model.predict(X_train)
y_valid_pred = model.predict(X_valid)

# Calculate and print the R-squared for the training set and validation set
train_r2 = r2_score(y_train, y_train_pred)
valid_r2 = r2_score(y_valid, y_valid_pred)

print(f"\nTraining Set R-squared: {train_r2:.4f}")
print(f"Validation Set R-squared: {valid_r2:.4f}")

# Calculate and print the Mean Squared Error (MSE) for the training set and validation set
train_mse = mean_squared_error(y_train, y_train_pred)
valid_mse = mean_squared_error(y_valid, y_valid_pred)

print(f"\nTraining Set Mean Squared Error (MSE): {train_mse:.4f}")
print(f"Validation Set Mean Squared Error (MSE): {valid_mse:.4f}")
```

```
Training Set Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.849
Model:                            OLS   Adj. R-squared:                  0.845
Method:                 Least Squares   F-statistic:                     244.6
Date:                Thu, 07 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:30:55   Log-Likelihood:                -58019.
No. Observations:                7216   AIC:                         1.164e+05
Df Residuals:                    7053   BIC:                         1.175e+05
Df Model:                         162
Covariance Type:            nonrobust
==============================================================================================================
                                                coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------
const                                       -27.2374      9.144     -2.979      0.003     -45.162      -9.313
superhost_period_all                        -14.4642      7.711     -1.876      0.061     -29.581       0.652
superhost_observed_in_period                  0.0277      0.098      0.282      0.778      -0.165       0.220
prev_superhost_period_all                    12.7732      5.024      2.542      0.011       2.924      22.622
rating_ave_pastYear                          10.4491     28.868      0.362      0.717     -46.141      67.039
numReviews_pastYear                          -1.8087      1.048     -1.726      0.084      -3.863       0.245
numCancel_pastYear                           -8.2830     14.144     -0.586      0.558     -36.009      19.443
num_5_star_Rev_pastYear                       2.6989      1.322      2.042      0.041       0.107       5.290
prop_5_StarReviews_pastYear                  13.4801    146.246      0.092      0.927    -273.206     300.167
prev_rating_ave_pastYear                     11.0374     29.637      0.372      0.710     -47.060      69.135
prev_numReviews_pastYear                      2.4028      1.316      1.825      0.068      -0.178       4.983
prev_numCancel_pastYear                      -4.5524     15.771     -0.289      0.773     -35.468      26.364
prev_num_5_star_Rev_pastYear                 -3.2768      1.662     -1.972      0.049      -6.535      -0.019
prev_prop_5_StarReviews_pastYear            -14.8165    151.320     -0.098      0.922    -311.449     281.816
numReservedDays_pastYear                      0.2041      0.033      6.158      0.000       0.139       0.269
numReserv_pastYear                           -0.4006      0.083     -4.805      0.000      -0.564      -0.237
prev_numReservedDays_pastYear                -0.1939      0.035     -5.501      0.000      -0.263      -0.125
prev_numReserv_pastYear                       0.3389      0.092      3.665      0.000       0.158       0.520
hostResponseNumber_pastYear                  -4.6822      0.761     -6.150      0.000      -6.175      -3.190
hostResponseAverage_pastYear                 -1.0546      0.608     -1.734      0.083      -2.247       0.138
```

```
prev_hostResponseNumber_pastYear        3.9156      0.682      5.744      0.000       2.579       5.252
prev_hostResponseAverage_pastYear      -0.3066      0.465     -0.659      0.510      -1.219       0.606
available_days                          0.1350      0.314      0.431      0.667      -0.480       0.750
available_days_aveListedPrice          -0.1345      0.207     -0.651      0.515      -0.539       0.270
booked_days                            85.7690      1.148     74.731      0.000      83.519      88.019
booked_days_avePrice                    6.1884      0.170     36.321      0.000       5.854       6.522
prev_available_days                     0.2961      0.333      0.889      0.374      -0.357       0.949
prev_available_days_aveListedPrice     -0.5176      0.388     -1.334      0.182      -1.278       0.243
prev_booked_days                      -16.8943      1.380    -12.239      0.000     -19.600     -14.188
prev_booked_days_avePrice              -0.7050      0.210     -3.359      0.001      -1.116      -0.294
Bedrooms                              120.3179     22.860      5.263      0.000      75.506     165.129
Bathrooms                              30.1874     25.142      1.201      0.230     -19.099      79.474
Max Guests                             61.2429      7.266      8.429      0.000      47.000      75.486
Cleaning Fee (USD)                      0.4197      0.327      1.284      0.199      -0.221       1.060
Minimum Stay                            0.2202      0.454      0.485      0.628      -0.670       1.111
Number of Photos                        0.0576      0.764      0.075      0.940      -1.441       1.556
Nightly Rate                            1.6578      0.232      7.158      0.000       1.204       2.112
prev_Nightly Rate                      -0.1807      0.068     -2.647      0.008      -0.314      -0.047
Number of Reviews                      -2.9938      0.951     -3.147      0.002      -4.859      -1.129
prev_Number of Reviews                  2.8871      0.994      2.905      0.004       0.939       4.835
occupancy_rate                       -484.9137    127.085     -3.816      0.000    -734.038    -235.790
prev_revenue                            0.1513      0.007     21.992      0.000       0.138       0.165
prev_occupancy_rate                   154.7936    154.223      1.004      0.316    -147.530     457.118
```

```
#All predictors included in model

# Assuming 'df' is your DataFrame with one-hot encoded columns
#df = pd.get_dummies(df, columns=categorical_variables, drop_first=True)


# Separate features (X) and target variable (y)
#X = df.drop('revenue', axis=1)
#y = df['revenue']

# Add a constant term to the features matrix
#X = sm.add_constant(X)

# Fit the linear regression model
#model = sm.OLS(y, X).fit()

# Print the model summary
#print(model.summary())
```

```
#Manually selected variables to remove
#Created interaction terms

# Exclude 'Quarter' and 'Year' columns
df2 = df2.drop(columns=['superhost_period_all', 'prev_superhost_period_all', 'numCancel_pastYear',
                        'prev_prop_5_StarReviews_pastYear', 'prev_numCancel_pastYear',
                        'prev_num_5_star_Rev_pastYear', 'prev_numReservedDays_pastYear',
                        'prev_numReserv_pastYear', 'hostResponseNumber_pastYear',
                        'hostResponseAverage_pastYear', 'prev_hostResponseNumber_pastYear',
                        'prev_hostResponseAverage_pastYear', 'Rating Overall', 'prev_Rating Overall', 'Zipcode'
                        ])

# Remove specified variables from the list
variables_to_remove = ['superhost_period_all', 'prev_superhost_period_all', 'numCancel_pastYear',
                       'prev_prop_5_StarReviews_pastYear', 'prev_numCancel_pastYear',
                       'prev_num_5_star_Rev_pastYear', 'prev_numReservedDays_pastYear',
                       'prev_numReserv_pastYear', 'hostResponseNumber_pastYear',
                       'hostResponseAverage_pastYear', 'prev_hostResponseNumber_pastYear',
                       'prev_hostResponseAverage_pastYear', 'Rating Overall', 'prev_Rating Overall', 'Zipcode','Instantbook Enab

categorical_variables2 = categorical_variables
continuous_variables2 = continuous_variables

for variable in variables_to_remove:
    if variable in categorical_variables2:
        categorical_variables2.remove(variable)
    if variable in continuous_variables2:
        continuous_variables2.remove(variable)


# Assuming 'df' is your DataFrame with one-hot encoded columns
df2 = pd.get_dummies(df2, columns=categorical_variables2, drop_first=True)

neighborhood_columns = [col for col in df2.columns if col.startswith('Neighborhood_')]
# Create interaction terms for each one-hot encoded 'Neighborhood' column with 'Max Guests'
for col in neighborhood_columns:
    df2[f'{col}_Max_Guests'] = df2[col] * df2['Max Guests']

listing_type_columns = [col for col in df2.columns if col.startswith('Listing Type_')]

# Create interaction terms for each combination of 'Neighborhood' and 'Listing Type'
for neighborhood_col in neighborhood_columns:
    for listing_type_col in listing_type_columns:
        interaction_col_name = f'{neighborhood_col}_{listing_type_col}'
        df2[interaction_col_name] = df2[neighborhood_col] * df2[listing_type_col]

# List of one-hot encoded 'Neighborhood' columns
#neighborhood_columns = [col for col in df2.columns if col.startswith('Neighborhood_')]
# List of one-hot encoded 'Property Type' columns
#property_type_columns = [col for col in df2.columns if col.startswith('Property Type_')]
# Create interaction terms for each combination of 'Neighborhood' and 'Property Type'
#for neighborhood_col in neighborhood_columns:
    #for property_type_col in property_type_columns:
        #interaction_col_name = f'{neighborhood_col}_{property_type_col}'
        #df2[interaction_col_name] = df2[neighborhood_col] * df2[property_type_col]


# Create interaction terms for each one-hot encoded 'Neighborhood' column with 'Max Guests'
#for col in neighborhood_columns:
    #df2[f'{col}_Nightly Rate'] = df2[col] * df2['Nightly Rate']


# Add interaction terms
#df2['Neighborhood_Max_Guests'] = df2['Neighborhood'] * df2['Max Guests']
#df2['Neighborhood_Property_Type'] = df2['Neighborhood'] * df2['Property Type']
#df2['Nightly_Rate_Neighborhood'] = df2['Nightly Rate'] * df2['Neighborhood']
df2['Max_Guests_Occupancy_Rate'] = df2['Max Guests'] * df2['occupancy_rate']
#df2['Prop_5starreviews_prevyear_Minimum_Stay'] = df2['prop_5_StarReviews_pastYear'] * df2['Minimum Stay']

df2 = df2.drop(columns=['occupancy_rate'])

# Separate features (X) and target variable (y)
X = df2.drop('revenue', axis=1)
y = df2['revenue']

# Add a constant term to the features matrix
X = sm.add_constant(X)
```

```
# Get the initial list of columns
initial_columns = X.columns.tolist()

while True:
    # Fit the linear regression model
    model = sm.OLS(y, X).fit()

    # Find variables with coefficients equal to 0
    zero_coeff_vars = model.params[model.params == 0].index.tolist()

    # Break the loop if no variables with coefficient 0 are found
    if not zero_coeff_vars:
        break

    # Remove variables with coefficients equal to 0
    X = X.drop(columns=zero_coeff_vars)
    print(f"Removed variables with coefficients equal to 0: {zero_coeff_vars}")

# Print the final model summary
print(model.summary())


# Get the coefficients and sort them in descending order
coefficients = model.params.sort_values(ascending=False)

# Print the list of variables and coefficients
print("Variable\tCoefficient")
print("=============================")
for variable, coefficient in coefficients.items():
    print(f"{variable}\t{coefficient}")

# Assuming 'coefficients' is the pandas Series containing coefficients
coefficients_df = pd.DataFrame({'Variable': coefficients.index, 'Coefficient': coefficients.values})

# Filter for neighborhood coefficients excluding specified strings
neighborhood_coefficients_df = coefficients_df[coefficients_df['Variable'].str.startswith('Neighborhood_') & ~coefficients_df['V

# Sort the DataFrame by 'Coefficient' in descending order
neighborhood_coefficients_df = neighborhood_coefficients_df.sort_values(by='Coefficient', ascending=False)

# Display the table
print(neighborhood_coefficients_df)
```

```
Removed variables with coefficients equal to 0: ['Neighborhood_Peninsula_Listing Type_Shared room', 'Neighborhood_White Rock
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.856
Model:                            OLS   Adj. R-squared:                  0.854
Method:                 Least Squares   F-statistic:                     429.0
Date:                Thu, 07 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:30:57   Log-Likelihood:                -77165.
No. Observations:                9622   AIC:                         1.546e+05
Df Residuals:                    9489   BIC:                         1.555e+05
Df Model:                         132
Covariance Type:            nonrobust
==============================================================================================================
                                                            coef    std err          t      P>|t|      [0.025
--------------------------------------------------------------------------------------------------------------
const                                                   -67.1967      9.046     -7.429      0.000     -84.928      -
superhost_observed_in_period                              0.0524      0.052      1.003      0.316      -0.050
rating_ave_pastYear                                     -40.9606     18.369     -2.230      0.026     -76.969
numReviews_pastYear                                       0.0732      0.166      0.441      0.659      -0.252
num_5_star_Rev_pastYear                                   0.1430      0.182      0.784      0.433      -0.215
prop_5_StarReviews_pastYear                             254.1544     91.942      2.764      0.006      73.929      4
prev_rating_ave_pastYear                                  6.7479      6.028      1.119      0.263      -5.068
prev_numReviews_pastYear                                 -0.1644      0.111     -1.486      0.137      -0.381
numReservedDays_pastYear                                  0.0054      0.005      1.155      0.248      -0.004
numReserv_pastYear                                       -0.0185      0.011     -1.611      0.107      -0.041
available_days                                            3.6627      0.243     15.100      0.000       3.187
available_days_aveListedPrice                            -0.7040      0.165     -4.263      0.000      -1.028
booked_days                                              68.2402      0.757     90.141      0.000      66.756
booked_days_avePrice                                      5.9363      0.140     42.368      0.000       5.662
prev_available_days                                      -1.7316      0.270     -6.410      0.000      -2.261
prev_available_days_aveListedPrice                        0.2695      0.299      0.903      0.367      -0.316
prev_booked_days                                        -12.4052      1.144    -10.842      0.000     -14.648      -
prev_booked_days_avePrice                                -0.2308      0.171     -1.352      0.176      -0.565
```

```
Bedrooms                                   67.4409     19.148      3.522      0.000      29.906       1
Bathrooms                                  68.1817     21.594      3.157      0.002      25.853       1
Max Guests                                  2.6774      8.178      0.327      0.743     -13.353
Cleaning Fee (USD)                          0.0540      0.268      0.201      0.840      -0.471
Minimum Stay                                0.3475      0.384      0.906      0.365      -0.404
Number of Photos                            1.5523      0.591      2.626      0.009       0.394
Nightly Rate                                1.9433      0.183     10.615      0.000       1.584
prev_Nightly Rate                          -0.1599      0.062     -2.582      0.010      -0.281
Number of Reviews                          -2.2924      0.809     -2.833      0.005      -3.879
prev_Number of Reviews                      2.2471      0.851      2.642      0.008       0.580
prev_revenue                                0.1439      0.006     25.441      0.000       0.133
prev_occupancy_rate                      -166.8543    130.272     -1.281      0.200    -422.216
prev_year_superhosts                       -3.7874      7.918     -0.478      0.632     -19.309
booked_days_period_city                    -0.3079      0.020    -15.346      0.000      -0.347
revenue_period_city                         0.0025      0.000     15.051      0.000       0.002
host_is_superhost_in_period_1              -3.6617     10.237     -0.358      0.721     -23.728
prev_host_is_superhost_in_period_1         18.6119     11.682      1.593      0.111      -4.287
Superhost_1                                -3.6617     10.237     -0.358      0.721     -23.728
prev_host_is_superhost_1                   18.6119     11.682      1.593      0.111      -4.287
superhost_change_1                          9.6206     16.605      0.579      0.562     -22.928
superhost_change_lose_superhost_1          15.9471     12.204      1.307      0.191      -7.975
superhost_change_gain_superhost_1          -6.3265     10.741     -0.589      0.556     -27.382
Property Type_Bed & Breakfast             265.8221    202.991      1.310      0.190    -132.083       6
Property Type_Boutique hotel              197.0002    231.773      0.850      0.395    -257.325       6
```

```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

# ... (Previous code remains unchanged)

# Separate features (X) and target variable (y)
X = df2.drop('revenue', axis=1)
y = df2['revenue']

# Add a constant term to the features matrix
X = sm.add_constant(X)

# Split the data into a 75% training set and a 25% validation set
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.25, random_state=42)


while True:
    # Fit the linear regression model
    model = sm.OLS(y_train, X_train).fit()

    # Find variables with coefficients equal to 0
    zero_coeff_vars = model.params[model.params == 0].index.tolist()

    # Break the loop if no variables with coefficient 0 are found
    if not zero_coeff_vars:
        break

    # Remove variables with coefficients equal to 0
    X_train = X_train.drop(columns=zero_coeff_vars)
    X_valid = X_valid.drop(columns=zero_coeff_vars)
    print(f"Removed variables with coefficients equal to 0: {zero_coeff_vars}")


# Fit the linear regression model on the training set
#model = sm.OLS(y_train, X_train).fit()

# Print the final model summary on the training set
print("Training Set Model Summary:")
print(model.summary())

# Make predictions on the validation set
y_valid_pred = model.predict(X_valid)
y_train_pred = model.predict(X_train)

# Calculate and print the R-squared on the validation set
valid_r2 = r2_score(y_valid, y_valid_pred)
print(f"\nValidation Set R-squared: {valid_r2:.4f}")

# Calculate and print the Mean Squared Error (MSE) for the training set and validation set
train_mse = mean_squared_error(y_train, y_train_pred)
valid_mse = mean_squared_error(y_valid, y_valid_pred)

print(f"\nTraining Set Mean Squared Error (MSE): {train_mse:.4f}")
print(f"Validation Set Mean Squared Error (MSE): {valid_mse:.4f}")


coefficients = model.params.sort_values(ascending=False)
# Assuming 'coefficients' is the pandas Series containing coefficients
coefficients_df = pd.DataFrame({'Variable': coefficients.index, 'Coefficient': coefficients.values})

# Filter for neighborhood coefficients excluding specified strings
neighborhood_coefficients_df = coefficients_df[
    coefficients_df['Variable'].str.startswith('Neighborhood_') &
    ~coefficients_df['Variable'].str.contains('Nightly Rate|Listing Type|Max_Guests')
]

# Sort the DataFrame by 'Coefficient' in descending order
neighborhood_coefficients_df = neighborhood_coefficients_df.sort_values(by='Coefficient', ascending=False)

# Display the table using tabulate with 'pretty' format
table = tabulate(neighborhood_coefficients_df, headers='keys', tablefmt='pretty')

# Print the table
print(table)
```

```
# Assuming 'coefficients' is the pandas Series containing coefficients
coefficients_df = pd.DataFrame({'Variable': coefficients.index, 'Coefficient': coefficients.values})

# Filter for neighborhood coefficients ending with 'Private room'
private_room_coefficients_df = coefficients_df[
    coefficients_df['Variable'].str.startswith('Neighborhood_') &
    coefficients_df['Variable'].str.endswith('Private room')]

# Sort the DataFrame by 'Coefficient' in descending order
private_room_coefficients_df = private_room_coefficients_df.sort_values(by='Coefficient', ascending=False)

# Display the table for 'Private room'
print("Private Room Coefficients:")
print(tabulate(private_room_coefficients_df, headers='keys', tablefmt='pretty'))

# Filter for neighborhood coefficients ending with 'Shared room'
shared_room_coefficients_df = coefficients_df[
    coefficients_df['Variable'].str.startswith('Neighborhood_') &
    coefficients_df['Variable'].str.endswith('Shared room')]

# Sort the DataFrame by 'Coefficient' in descending order
shared_room_coefficients_df = shared_room_coefficients_df.sort_values(by='Coefficient', ascending=False)

# Display the table for 'Shared room'
print("\nShared Room Coefficients:")
print(tabulate(shared_room_coefficients_df, headers='keys', tablefmt='pretty'))


# Assuming 'coefficients' is the pandas Series containing coefficients
coefficients_df = pd.DataFrame({'Variable': coefficients.index, 'Coefficient': coefficients.values})

# Filter for neighborhood coefficients ending with 'Private room'
private_room_coefficients_df = coefficients_df[
    coefficients_df['Variable'].str.startswith('Max_Guests_Occupancy_Rate')]

# Sort the DataFrame by 'Coefficient' in descending order
private_room_coefficients_df = private_room_coefficients_df.sort_values(by='Coefficient', ascending=False)

# Display the table for 'Private room'
print("Max_Guests_Occupancy_Rate Coefficients:")
print(tabulate(private_room_coefficients_df, headers='keys', tablefmt='pretty'))
```

```
Removed variables with coefficients equal to 0: ['Neighborhood_Peninsula_Listing Type_Shared room', 'Neighborhood_White Rock
Training Set Model Summary:
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.856
Model:                            OLS   Adj. R-squared:                  0.853
Method:                 Least Squares   F-statistic:                     321.2
Date:                Thu, 07 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:30:59   Log-Likelihood:                -57847.
No. Observations:                7216   AIC:                         1.160e+05
Df Residuals:                    7084   BIC:                         1.169e+05
Df Model:                         131
Covariance Type:            nonrobust
=====================================================================================================================
                                                                 coef    std err          t      P>|t|      [0.025
---------------------------------------------------------------------------------------------------------------------
const                                                        -73.1813     10.508     -6.964      0.000     -93.781     -
superhost_observed_in_period                                   0.0688      0.061      1.122      0.262      -0.051
rating_ave_pastYear                                          -13.0069     21.540     -0.604      0.546     -55.232
numReviews_pastYear                                            0.0152      0.193      0.079      0.937      -0.362
num_5_star_Rev_pastYear                                        0.1593      0.210      0.758      0.448      -0.253
prop_5_StarReviews_pastYear                                  133.3608    108.054      1.234      0.217     -78.458     3
prev_rating_ave_pastYear                                       1.4209      6.954      0.204      0.838     -12.210
prev_numReviews_pastYear                                      -0.1236      0.127     -0.969      0.332      -0.373
numReservedDays_pastYear                                       0.0038      0.005      0.699      0.484      -0.007
numReserv_pastYear                                            -0.0150      0.013     -1.124      0.261      -0.041
available_days                                                 3.6835      0.281     13.106      0.000       3.133
available_days_aveListedPrice                                 -0.6776      0.185     -3.657      0.000      -1.041
booked_days                                                   68.6749      0.879     78.126      0.000      66.952
booked_days_avePrice                                           5.8534      0.165     35.568      0.000       5.531
prev_available_days                                           -1.6980      0.315     -5.386      0.000      -2.316
prev_available_days_aveListedPrice                             0.1384      0.344      0.402      0.688      -0.537
```

```
prev_booked_days                          -13.1126      1.324     -9.908      0.000    -15.707      -
prev_booked_days_avePrice                  -0.4067      0.197     -2.064      0.039     -0.793
Bedrooms                                   82.1168     22.086      3.718      0.000     38.822      1
Bathrooms                                  67.4952     24.838      2.717      0.007     18.805      1
Max Guests                                  6.1197      9.452      0.647      0.517    -12.408
Cleaning Fee (USD)                          0.1369      0.311      0.441      0.659     -0.472
Minimum Stay                                0.3544      0.437      0.811      0.417     -0.502
Number of Photos                            1.6898      0.687      2.459      0.014      0.343
Nightly Rate                                1.9792      0.213      9.299      0.000      1.562
prev_Nightly Rate                          -0.1473      0.065     -2.272      0.023     -0.274
Number of Reviews                          -2.0352      0.918     -2.217      0.027     -3.835
prev_Number of Reviews                      2.0754      0.962      2.159      0.031      0.191
prev_revenue                                0.1501      0.007     22.692      0.000      0.137
prev_occupancy_rate                      -154.7192    149.238     -1.037      0.300   -447.271      1
prev_year_superhosts                       -0.9675      9.119     -0.106      0.916    -18.843
booked_days_period_city                    -0.3254      0.023    -14.086      0.000     -0.371
revenue_period_city                         0.0026      0.000     13.823      0.000      0.002
host_is_superhost_in_period_1               2.2128     11.812      0.187      0.851    -20.942
prev_host_is_superhost_in_period_1         23.7654     13.628      1.744      0.081     -2.949
Superhost_1                                 2.2128     11.812      0.187      0.851    -20.942
prev_host_is_superhost_1                   23.7654     13.628      1.744      0.081     -2.949
superhost_change_1                          7.1439     19.168      0.373      0.709    -30.431
superhost_change_lose_superhost_1          14.3483     14.049      1.021      0.307    -13.192
superhost_change_gain_superhost_1          -7.2044     12.491     -0.577      0.564    -31.691
Property Type_Bed & Breakfast             337.5458    240.961      1.401      0.161   -134.811      8
Property Type_Boutique hotel               16.3562    287.155      0.057      0.955   -546.553      5
```

```
#Previous 5 star reviews regression
#What drives previous 5 star reviews?

# Remove specified variables from the list
variables_to_remove = ['superhost_period_all','prev_superhost_period_all','numCancel_pastYear',
                       'prev_prop_5_StarReviews_pastYear','prev_numCancel_pastYear',
                       'prev_num_5_star_Rev_pastYear','prev_numReservedDays_pastYear',
                       'prev_numReserv_pastYear','hostResponseNumber_pastYear',
                       'hostResponseAverage_pastYear','prev_hostResponseNumber_pastYear',
                       'prev_hostResponseAverage_pastYear',
                       'Instantbook Enabled','prev_Instantbook Enabled','Rating Overall','prev_Rating Overall','Zipcode'
                       ]

categorical_variables3 = categorical_variables
continuous_variables3 = continuous_variables

for variable in variables_to_remove:
    if variable in categorical_variables3:
        categorical_variables2.remove(variable)
    if variable in continuous_variables3:
        continuous_variables2.remove(variable)

# Exclude 'Quarter' and 'Year' columns
df3 = df3.drop(columns=['superhost_period_all','prev_superhost_period_all','numCancel_pastYear',
                       'prev_prop_5_StarReviews_pastYear','prev_numCancel_pastYear',
                       'prev_num_5_star_Rev_pastYear','prev_numReservedDays_pastYear',
                       'prev_numReserv_pastYear','hostResponseNumber_pastYear',
                       'hostResponseAverage_pastYear','prev_hostResponseNumber_pastYear',
                       'prev_hostResponseAverage_pastYear','Rating Overall', 'prev_Rating Overall','Zipcode'
                       ])
#'Pets Allowed','Instantbook Enabled','prev_Instantbook Enabled'

# Assuming 'df' is your DataFrame with one-hot encoded columns
df3 = pd.get_dummies(df3, columns=categorical_variables3, drop_first=True)


# Separate features (X) and target variable (y)
X = df3.drop('prop_5_StarReviews_pastYear', axis=1)
y = df3['prop_5_StarReviews_pastYear']

# Add a constant term to the features matrix
X = sm.add_constant(X)

# Fit the linear regression model
model = sm.OLS(y, X).fit()

# Print the model summary
print(model.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:     prop_5_StarReviews_pastYear   R-squared:                       0.946
Model:                                     OLS   Adj. R-squared:                  0.946
```

```
Method:                 Least Squares   F-statistic:                     1886.
Date:                Thu, 07 Dec 2023   Prob (F-statistic):               0.00
Time:                        22:31:00   Log-Likelihood:                 10345.
No. Observations:                9622   AIC:                        -2.051e+04
Df Residuals:                    9532   BIC:                        -1.986e+04
Df Model:                          89
Covariance Type:            nonrobust
==============================================================================================================
```