In [1]:

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
Abn=pd.read_csv("Airbin.csv")
```

In [3]:

```
Abn
```

Out[3]:

	City	Price	Day	Room Type	Shared Room	Private Room	Person Capacity	Superhost	Mu Rc
0	Amsterdam	194.033698	Weekday	Private room	False	True	2.0	False	
1	Amsterdam	344.245776	Weekday	Private room	False	True	4.0	False	
2	Amsterdam	264.101422	Weekday	Private room	False	True	2.0	False	
3	Amsterdam	433.529398	Weekday	Private room	False	True	4.0	False	
4	Amsterdam	485.552926	Weekday	Private room	False	True	2.0	True	
...	...	...	...	...	...	...	...	...	
41709	Vienna	715.938574	Weekend	Entire home/apt	False	False	6.0	False	
41710	Vienna	304.793960	Weekend	Entire home/apt	False	False	2.0	False	
41711	Vienna	637.168969	Weekend	Entire home/apt	False	False	2.0	False	
41712	Vienna	301.054157	Weekend	Private room	False	True	2.0	False	
41713	Vienna	133.230489	Weekend	Private room	False	True	4.0	True	

41714 rows × 19 columns

In [4]:

```
Abn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41714 entries, 0 to 41713
Data columns (total 19 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   City                         41714 non-null  object 
 1   Price                        41714 non-null  float64
 2   Day                          41714 non-null  object 
 3   Room Type                    41714 non-null  object 
 4   Shared Room                  41714 non-null  bool   
 5   Private Room                 41714 non-null  bool   
 6   Person Capacity              41714 non-null  float64
 7   Superhost                    41714 non-null  bool   
 8   Multiple Rooms               41714 non-null  int64  
 9   Business                     41714 non-null  int64  
 10  Cleanliness Rating           41714 non-null  float64
 11  Guest Satisfaction           41714 non-null  float64
 12  Bedrooms                     41714 non-null  int64  
 13  City Center (km)             41714 non-null  float64
 14  Metro Distance (km)          41714 non-null  float64
 15  Attraction Index             41714 non-null  float64
 16  Normalised Attraction Index  41714 non-null  float64
 17  Restraunt Index              41714 non-null  float64
 18  Normalised Restraunt Index   41714 non-null  float64
dtypes: bool(3), float64(10), int64(3), object(3)
memory usage: 5.2+ MB
```

In [5]:

```
Abn.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%
Price	41714.0	260.094423	279.408493	34.779339	144.016085	203.819274	297.373358
Person Capacity	41714.0	3.237235	1.299459	2.000000	2.000000	3.000000	4.000000
Multiple Rooms	41714.0	0.295273	0.456171	0.000000	0.000000	0.000000	1.000000
Business	41714.0	0.341180	0.474112	0.000000	0.000000	0.000000	1.000000
Cleanliness Rating	41714.0	9.442274	0.889173	2.000000	9.000000	10.000000	10.000000
Guest Satisfaction	41714.0	93.103179	8.141745	20.000000	90.000000	95.000000	98.000000
Bedrooms	41714.0	1.165939	0.638157	0.000000	1.000000	1.000000	1.000000
City Center (km)	41714.0	2.679792	1.996684	0.015045	1.275913	2.253237	3.584489
Metro Distance (km)	41714.0	0.603921	0.706206	0.002301	0.236693	0.391220	0.678702
Attraction Index	41714.0	293.905990	235.750055	15.152201	124.371614	228.920599	394.000201
Normalised Attraction Index	41714.0	11.719658	8.379161	0.926301	5.510735	9.951086	15.467009
Restraunt Index	41714.0	626.692618	520.644720	19.576924	210.459574	519.583509	860.708156
Normalised Restraunt Index	41714.0	25.553590	18.484572	0.592757	11.132052	21.814414	36.821356

In [6]:

```
Abn.duplicated().sum()
```

Out[6]:

```
0
```

In [7]:

```
Abn.isna().sum()
```

Out[7]:

```
City                          0
Price                         0
Day                           0
Room Type                     0
Shared Room                   0
Private Room                  0
Person Capacity               0
Superhost                     0
Multiple Rooms                0
Business                      0
Cleanliness Rating            0
Guest Satisfaction            0
Bedrooms                      0
City Center (km)              0
Metro Distance (km)           0
Attraction Index              0
Normalised Attraction Index   0
Restraunt Index               0
Normalised Restraunt Index    0
dtype: int64
```

In [8]:

```
ab=Abn.copy()
```

In [9]:

```
ab
```

Out[9]:

	City	Price	Day	Room Type	Shared Room	Private Room	Person Capacity	Superhost	Mu Rc
0	Amsterdam	194.033698	Weekday	Private room	False	True	2.0	False	
1	Amsterdam	344.245776	Weekday	Private room	False	True	4.0	False	
2	Amsterdam	264.101422	Weekday	Private room	False	True	2.0	False	
3	Amsterdam	433.529398	Weekday	Private room	False	True	4.0	False	
4	Amsterdam	485.552926	Weekday	Private room	False	True	2.0	True	
...	...	...	...	...	...	...	...	...	
41709	Vienna	715.938574	Weekend	Entire home/apt	False	False	6.0	False	
41710	Vienna	304.793960	Weekend	Entire home/apt	False	False	2.0	False	
41711	Vienna	637.168969	Weekend	Entire home/apt	False	False	2.0	False	
41712	Vienna	301.054157	Weekend	Private room	False	True	2.0	False	
41713	Vienna	133.230489	Weekend	Private room	False	True	4.0	True	

41714 rows × 19 columns

In [10]:

```
ab.drop(["Shared Room","Private Room"],inplace=True,axis=1)
```

In [11]:

```
ab.columns
```

Out[11]:

```
Index(['City', 'Price', 'Day', 'Room Type', 'Person Capacity', 'Superhos
t',
       'Multiple Rooms', 'Business', 'Cleanliness Rating',
       'Guest Satisfaction', 'Bedrooms', 'City Center (km)',
       'Metro Distance (km)', 'Attraction Index',
       'Normalised Attraction Index', 'Restraunt Index',
       'Normalised Restraunt Index'],
      dtype='object')
```

1. Which city has the highest average price for private rooms?

In [12]:

```
#1)Which city has the highest average price for private rooms?
room_type=ab[ab["Room Type"]=="Private room"]
highest_average=room_type.groupby(["City","Room Type"]).agg({"Price":"mean"}).head(100).s
highest_average
```

Out[12]:

		Price
City	**Room Type**	
Amsterdam	**Private room**	383.468718
Paris	**Private room**	299.218870
Barcelona	**Private room**	214.714206
Vienna	**Private room**	190.426164
Berlin	**Private room**	180.520256
Lisbon	**Private room**	148.902035
Rome	**Private room**	148.787371
Athens	**Private room**	112.838060
Budapest	**Private room**	109.137759

2. What is the total number of shared rooms available in the dataset?

In [13]:

```
#2)What is the total number of shared rooms available in the dataset?
shared_rooms=ab[ab["Room Type"]=="Shared room"]
shared_rooms.count()
```

Out[13]:

```
City                          316
Price                         316
Day                           316
Room Type                     316
Person Capacity               316
Superhost                     316
Multiple Rooms                316
Business                      316
Cleanliness Rating            316
Guest Satisfaction            316
Bedrooms                      316
City Center (km)              316
Metro Distance (km)           316
Attraction Index              316
Normalised Attraction Index   316
Restraunt Index               316
Normalised Restraunt Index    316
dtype: int64
```

3. Which superhost has the most listings with multiple rooms?

In [14]:

```
#3)Which superhost has the most listings with multiple rooms?
most_listing=ab.groupby(["Superhost","Multiple Rooms"]).agg({"Multiple Rooms":"count"})
most_listing
```

Out[14]:

		Multiple Rooms
Superhost	**Multiple Rooms**	
False	**0**	21924
	1	8131
True	**0**	7473
	1	4186

4. How many listings have a person capacity greater than or equal to 4, and what is their average price?

In [15]:

```
#4)How many listings have a person capacity greater than or equal to 4, and what is their
for person in ab["Person Capacity"]:
    if person>=4:
        print(person)
ab["New_person"]=person
```

```
4.0
4.0
4.0
6.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
4.0
```

In [16]:

```
ab
```

Out[16]:

	City	Price	Day	Room Type	Person Capacity	Superhost	Multiple Rooms	Business
0	Amsterdam	194.033698	Weekday	Private room	2.0	False	1	0
1	Amsterdam	344.245776	Weekday	Private room	4.0	False	0	0
2	Amsterdam	264.101422	Weekday	Private room	2.0	False	0	1
3	Amsterdam	433.529398	Weekday	Private room	4.0	False	0	1
4	Amsterdam	485.552926	Weekday	Private room	2.0	True	0	0
...	...	...	...	...	...	...	...	...
41709	Vienna	715.938574	Weekend	Entire home/apt	6.0	False	0	1
41710	Vienna	304.793960	Weekend	Entire home/apt	2.0	False	0	0
41711	Vienna	637.168969	Weekend	Entire home/apt	2.0	False	0	0
41712	Vienna	301.054157	Weekend	Private room	2.0	False	0	0
41713	Vienna	133.230489	Weekend	Private room	4.0	True	1	0

41714 rows × 18 columns

In [17]:

```
#4)How many listings have a person capacity greater than or equal to 4, and what is their
listing=ab.groupby(["Person Capacity","New_person"]).agg({"Price":"mean"}).sort_values("P
listing
```

Out[17]:

		Price
Person Capacity	**New_person**	
6.0	**4.0**	389.981481
5.0	**4.0**	299.261527
4.0	**4.0**	292.046341
2.0	**4.0**	219.364248
3.0	**4.0**	219.194727

5. Which city has the highest percentage of listings that are marked as "Business"?

In [18]:

```
#5)Which city has the highest percentage of listings that are marked as "Business"?
business_listing=ab[["City","Business"]].value_counts(normalize=True).mul(100).round(1)
df=pd.DataFrame(business_listing,columns=["Percentage"])
df.rank(na_option="top")
```

Out[18]:

		Percentage
City	**Business**	
Rome	**0**	18.0
Paris	**0**	17.0
Lisbon	**1**	16.0
Athens	**0**	15.0
Rome	**1**	14.0
Budapest	**0**	13.0
Lisbon	**0**	12.0
Vienna	**0**	11.0
Berlin	**0**	10.0
Athens	**1**	9.0
Barcelona	**0**	8.0
Amsterdam	**0**	7.0
Paris	**1**	6.0
Budapest	**1**	5.0
Vienna	**1**	4.0
Barcelona	**1**	3.0
Berlin	**1**	2.0
Amsterdam	**1**	1.0

In [19]:

```
ab
```

Out[19]:

	City	Price	Day	Room Type	Person Capacity	Superhost	Multiple Rooms	Business
0	Amsterdam	194.033698	Weekday	Private room	2.0	False	1	0
1	Amsterdam	344.245776	Weekday	Private room	4.0	False	0	0
2	Amsterdam	264.101422	Weekday	Private room	2.0	False	0	1
3	Amsterdam	433.529398	Weekday	Private room	4.0	False	0	1
4	Amsterdam	485.552926	Weekday	Private room	2.0	True	0	0
...	...	...	...	...	...	...	...	...
41709	Vienna	715.938574	Weekend	Entire home/apt	6.0	False	0	1
41710	Vienna	304.793960	Weekend	Entire home/apt	2.0	False	0	0
41711	Vienna	637.168969	Weekend	Entire home/apt	2.0	False	0	0
41712	Vienna	301.054157	Weekend	Private room	2.0	False	0	0
41713	Vienna	133.230489	Weekend	Private room	4.0	True	1	0

41714 rows × 18 columns

In [ ]:

```

```

6. For each city in the dataset, calculate the top 5 most common room types and their corresponding average prices. Additionally, for each room type, calculate the percentage of listings in the dataset that have that room type, and display this information alongside the corresponding average price.

In [20]:

```
# 6)For each city in the dataset, calculate the top 5 most common
# room types and their corresponding average prices.
# Additionally, for each room type, calculate the percentage of
# listings in the dataset that have that room type, and display this
# information alongside the corresponding average price.
price=np.average(ab["Price"])
ab["Average_Price"]=price
common_room=ab[["City","Room Type","Average_Price"]].value_counts(normalize=True).mul(100
total_value=pd.DataFrame(common_room,columns=['Percentage'])
total_value.rank(method="average",ascending=True,na_option='top').head(5)
```

Out[20]:

			Percentage
City	**Room Type**	**Average_Price**	
Rome	**Entire home/apt**	**260.094423**	27.0
Paris	**Entire home/apt**	**260.094423**	26.0
Athens	**Entire home/apt**	**260.094423**	25.0
Lisbon	**Entire home/apt**	**260.094423**	24.0
Budapest	**Entire home/apt**	**260.094423**	23.0

7. For each combination of city and room type in the dataset, calculate the median price and the 25th and 75th percentiles of the price. Display the results in a table with multi-level row and column headers.

In [21]:

```
# 7)For each combination of city and room type in the dataset,
# calculate the median price and the 25th and 75th percentiles of
# the price. Display the results in a table with multi-level row and
# column headers
median=np.median(ab["Price"])
Price=ab["Price"].quantile(0.25)
Price
price1=ab["Price"].quantile(0.75)
multi_level=ab[["City","Room Type"]]
multi_level["25%"]=Price
multi_level["75%"]=price1
multi_level["median"]=median
multi_level
```

Out[21]:

	City	Room Type	25%	75%	median
0	Amsterdam	Private room	144.016085	297.373358	203.819274
1	Amsterdam	Private room	144.016085	297.373358	203.819274
2	Amsterdam	Private room	144.016085	297.373358	203.819274
3	Amsterdam	Private room	144.016085	297.373358	203.819274
4	Amsterdam	Private room	144.016085	297.373358	203.819274
...	...	...	...	...	...
41709	Vienna	Entire home/apt	144.016085	297.373358	203.819274
41710	Vienna	Entire home/apt	144.016085	297.373358	203.819274
41711	Vienna	Entire home/apt	144.016085	297.373358	203.819274
41712	Vienna	Private room	144.016085	297.373358	203.819274
41713	Vienna	Private room	144.016085	297.373358	203.819274

41714 rows × 5 columns

8. Calculate the average price for each day of the week, and for each city in the dataset. Display the results in a heatmap, where the x-axis corresponds to the day of the week, the y-axis corresponds to the city, and the color of each cell corresponds to the average price.

In [22]:

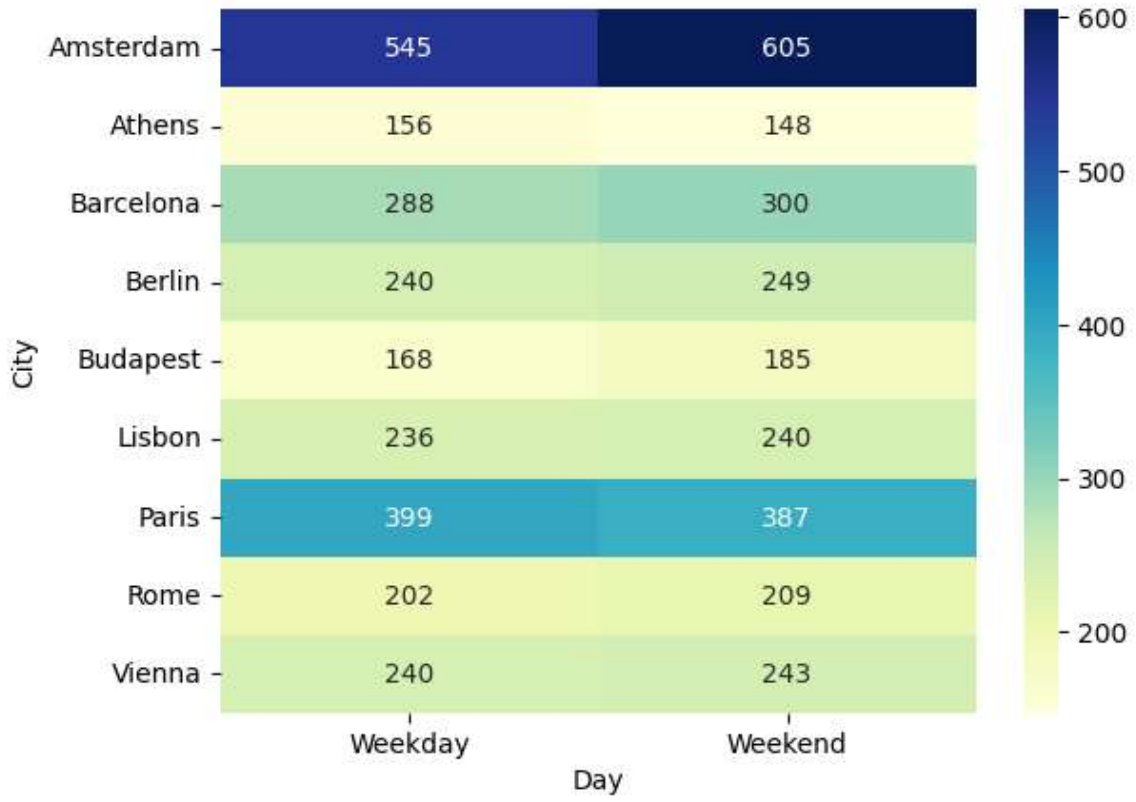
```
import seaborn as sns
```

In [23]:

```
import matplotlib.pyplot as plt
```

In [24]:

```
# 8)Calculate the average price for each day of the week, and for
# each city in the dataset. Display the results in a heatmap, where
# the x-axis corresponds to the day of the week, the y-axis
# corresponds to the city, and the color of each cell corresponds
# to the average price.
pivot_table=ab.pivot_table(values="Price",index="City",columns="Day",aggfunc="mean")
pivot_table
sns.heatmap(pivot_table,cmap="YlGnBu", annot=True, fmt=".0f")
plt.show()
```



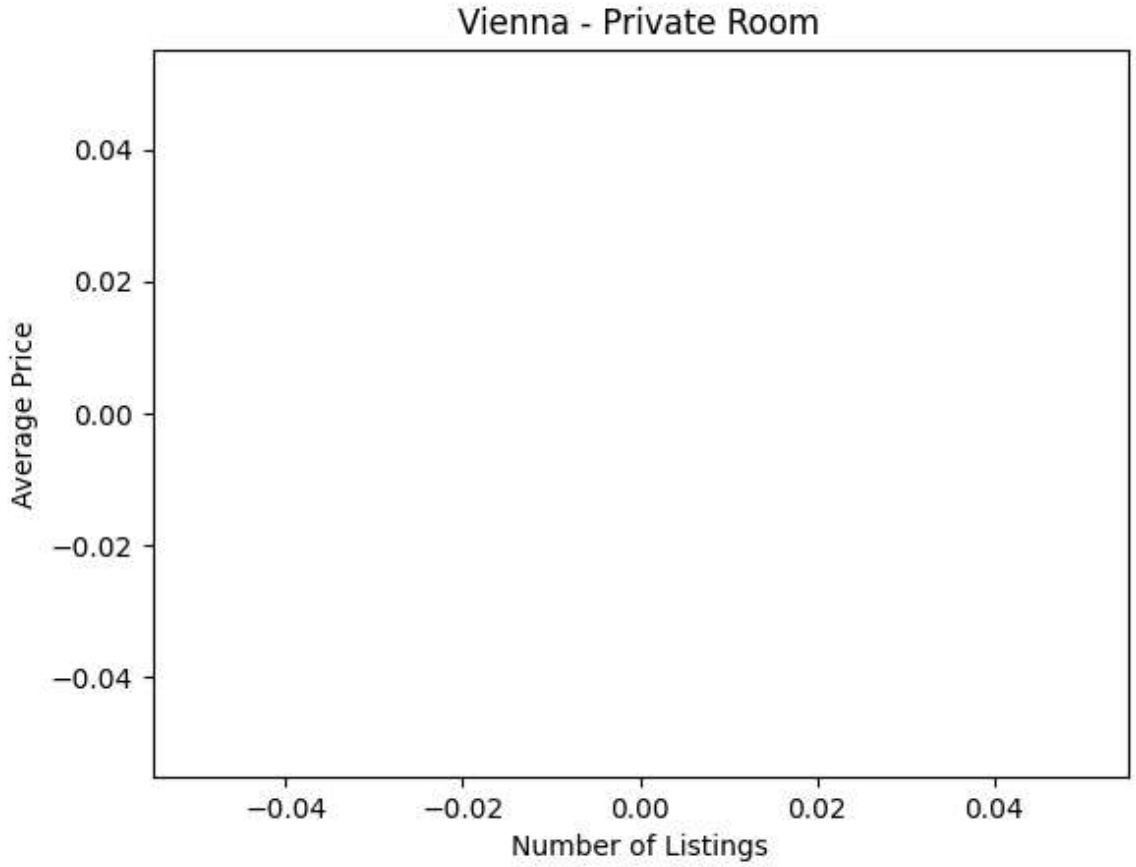
9. Create a function that takes a list of cities as input, and returns a summary table that shows the average price, minimum price,and maximum price for each city, broken down by room

type.The function should also display a scatter plot that shows the relationship between the number of listings and the average

In [26]:

```
# # 9)Create a function that takes a list of cities as input, and returns a
# summary table that shows the average price, minimum price,
# and maximum price for each city, broken down by room type.
# The function should also display a scatter plot that shows the
# relationship between the number of listings and the average
# price for each room type in each city.

def summarize_prices_by_city(cities):
    filtered_data = ab[ab["City"].isin(cities)]

    summary_table = filtered_data.groupby(["City", "Room Type"]).agg({"Price": ["mean", "
    summary_table.columns = ["Average Price", "Minimum Price", "Maximum Price"]
    print(summary_table)
    grouped_data = filtered_data.groupby(["City", "Room Type", "Person Capacity"]).agg({"

    for city in cities:
        for room_type in ["Shared Room", "Private Room"]:
            city_room_data = grouped_data[(grouped_data["City"] == city) & (grouped_data[
            num_listings = city_room_data["Person Capacity"]
            mean_price = city_room_data["Price"]



    plt.scatter(num_listings, mean_price)
    plt.title(f"{city} - {room_type}")
    plt.xlabel("Number of Listings")
    plt.ylabel("Average Price")
    plt.show()
summarize_prices_by_city(ab['City'])
```

City	Room Type	Average Price	Minimum Price	Maximum Price
Amsterdam	Entire home/apt	734.699030	128.887118	8130.668104
	Private room	383.468718	143.650552	1769.971645
	Shared room	280.903616	184.425749	479.225740
Athens	Entire home/apt	155.079543	51.086167	18545.450285
	Private room	112.838060	42.884259	609.518900
	Shared room	78.610490	60.225435	85.768519
Barcelona	Entire home/apt	629.855642	161.984779	6943.700980
	Private room	214.714206	69.588289	1619.847790
	Shared room	124.068083	80.992390	171.294249
Berlin	Entire home/apt	363.205813	111.246144	5857.483407
	Private room	180.520256	64.971487	2319.341872
	Shared room	153.192356	75.021034	417.640460
Budapest	Entire home/apt	184.573180	37.129295	3751.233727
	Private room	109.137759	34.779339	695.351788
	Shared room	126.830447	53.343986	233.115571
Lisbon	Entire home/apt	282.495813	112.570356	1681.050657
	Private room	148.902035	70.590994	887.429644
	Shared room	103.062725	72.701689	193.480300
Paris	Entire home/apt	425.107824	109.749278	16445.614689
	Private room	299.218870	105.322024	1952.418678
	Shared room	152.356013	92.739305	591.154814
Rome	Entire home/apt	240.784170	80.658359	2418.348023
	Private room	148.787371	46.057092	2311.738714
	Shared room	96.731583	69.202534	115.493419
Vienna	Entire home/apt	256.557459	74.094851	12942.991375
	Private room	190.426164	69.653834	13664.305916
	Shared room	145.253372	63.576654	324.661665



10. Identify the top 10% of hosts in the dataset based on the number of listings they have, and calculate the average price for each of these hosts. Then, calculate the correlation between the number of listings and the average price for these hosts, and display the results in a scatter plot.

In [63]:

```
ab1=ab.copy()
```

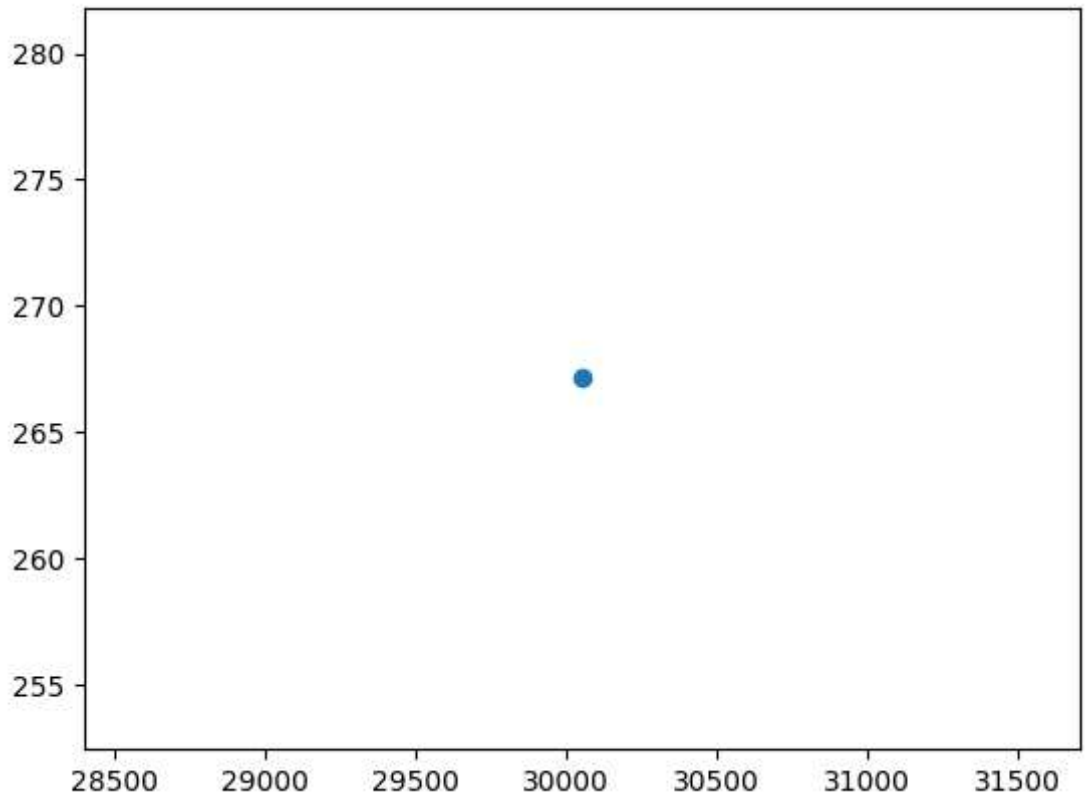
In [182]:

```
num_listings = ab1['Superhost'].value_counts()

# Identify the top 10% of hosts
top_hosts = num_listings[num_listings >= num_listings.quantile(0.9)].index

# Filter the dataframe for the top hosts
top_hosts_df = ab1[ab1['Superhost'].isin(top_hosts)]
avg_price = top_hosts_df.groupby('Superhost')['Price'].mean()

# Convert the series to a dataframe
avg_price_df = pd.DataFrame({'Superhost': avg_price.index, 'Price': avg_price.values})
# Calculate the correlation between the number of listings and the average price for thes
correlation = num_listings[top_hosts].corr(avg_price)

# Print the correlation
print("Correlation between number of listings and average price:", correlation)
plt.scatter(num_listings[top_hosts], avg_price)
```

```
Correlation between number of listings and average price: nan
```

Out[182]:

```
<matplotlib.collections.PathCollection at 0x2578255eed0>
```



In [ ]: