

**Group Details:**

Name: קונייבסקי ID: 315117341 Name: סיון חממי ID: 206842056 Name: הגר פרינץ ID: 323967604  
 Name: גיא אליחי ID: 316279199

## > Helper Functions and Imports

[Show code](#)

## ✓ Introduction to Data Science - Lab #2

### Exploratory Data Analysis

#### ✓ Case Study: Rental Listings in Jerusalem

In this lab we will practice our exploratory data analysis skills using real data!

We will explore data of rental pricings in Jersualem. The dataset consists of listings published in <https://www.komo.co.il/> during the summer of 2022.

We will use two python packages for visualizing the data: `matplotlib` (and specifically its submodule `pyplot` imported here as `plt`) and `seaborn` (imported as `sns`). Seaborn is a package that "wraps" `matplotlib` and introduces more convenient functions for quickly creating standard visualizations based on dataframes.

Please **briefly** go over this [quick start guide](#) to `matplotlib`, the [first](#) `seaborn` introduction page until the "Multivariate views on complex datasets" section (not included), and the [second](#) introduction page until the "Combining multiple views on the data" section.

#### ✓ Loading the dataset

```
#@title Loading the dataset
rent_df = load_df(RENT_ID)[['propertyID', 'neighborhood', 'monthlyRate', 'mefarsem', 'rooms', 'f]
rent_df = rent_df.drop_duplicates(subset='propertyID').reset_index(drop=True)
rent_df_backup_for_exercise = rent_df.copy()
clean_df_area_filtered = None
clean_df = None
```

Let's print a random sample:

```
np.random.seed(2)
rent_df.sample(5)
```



	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	des
403	3981729	גבעת מרדכי	4500.0	private	3.0	6.0	62.0	10/08/2022	צת
457	3981612	גבעת מרדכי	3000.0	private	3.5	3.0	NaN	NaN	7

And print some summary statistics:

```
rent_df.describe(include='all')
```



	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	
<b>count</b>	6.120000e+02	612	612.000000	612	612.000000	611.000000	295.0
<b>unique</b>	NaN	54	NaN	2	NaN	NaN	
<b>top</b>	NaN	קריית יובל	NaN	private	NaN	NaN	
<b>freq</b>	NaN	66	NaN	600	NaN	NaN	
<b>mean</b>	3.981582e+06	NaN	4717.393791	NaN	2.927288	1.916530	87.6
<b>std</b>	6.525543e+04	NaN	2195.215139	NaN	1.007350	1.581006	277.0
<b>min</b>	2.494041e+06	NaN	0.000000	NaN	1.000000	-2.000000	1.0
<b>25%</b>	3.981694e+06	NaN	3500.000000	NaN	2.000000	1.000000	42.0
<b>50%</b>	3.987901e+06	NaN	4400.000000	NaN	3.000000	2.000000	60.0
<b>75%</b>	3.992605e+06	NaN	5800.000000	NaN	3.500000	2.000000	85.0

The variables we will focus on are:

1. neighborhood: The hebrew name of the neighborhood in jerusalem where the listing is located
2. monthlyRate: The monthly rate (שכר דירה) in shekels
3. rooms: The number of rooms in the apartment
4. floor: The floor in which the apartment is located

5. area: The area of the apartment in squared meters
6. numFloors: The total number of floors in the building

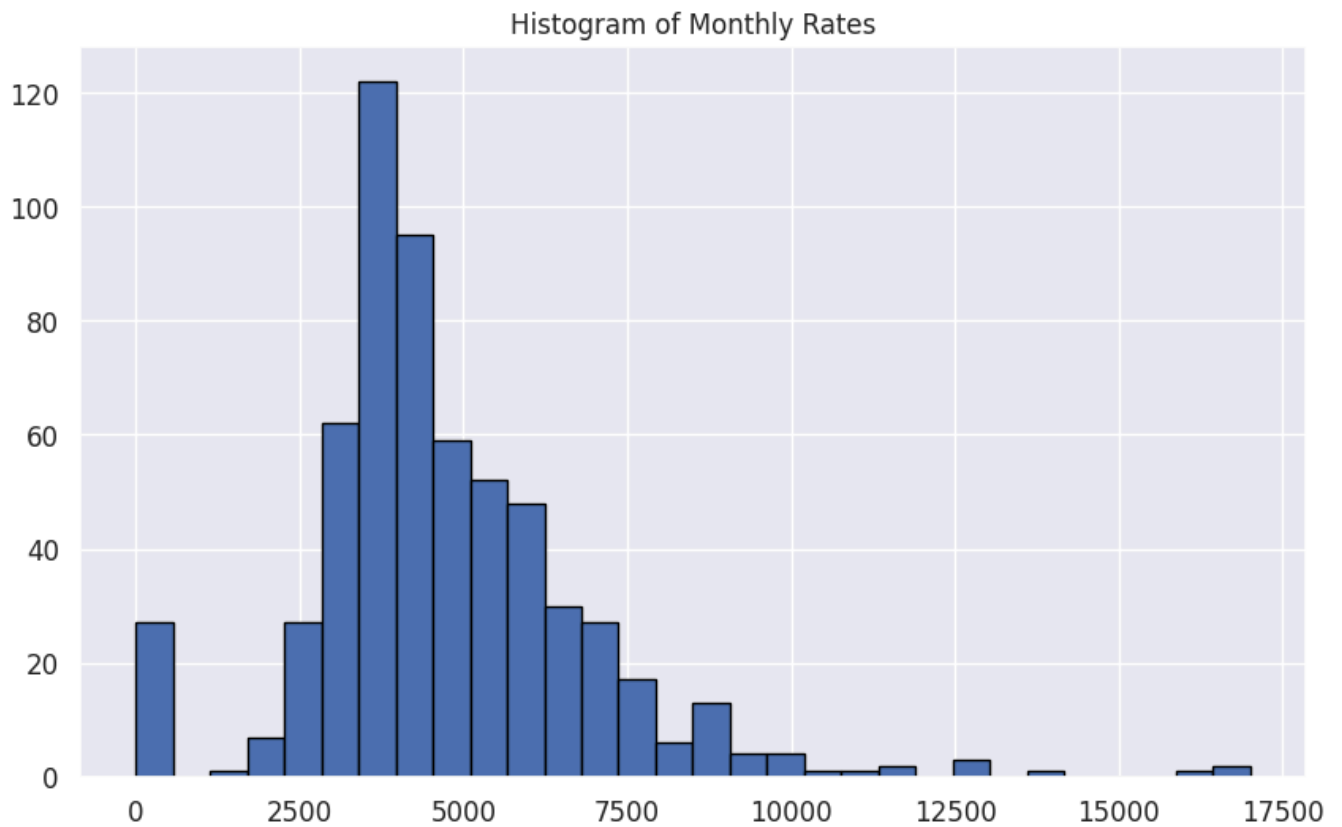
### What is the distribution of prices in this dataset?

Q: Plot a histogram with 30 bins of the monthly rates in this dataset:

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

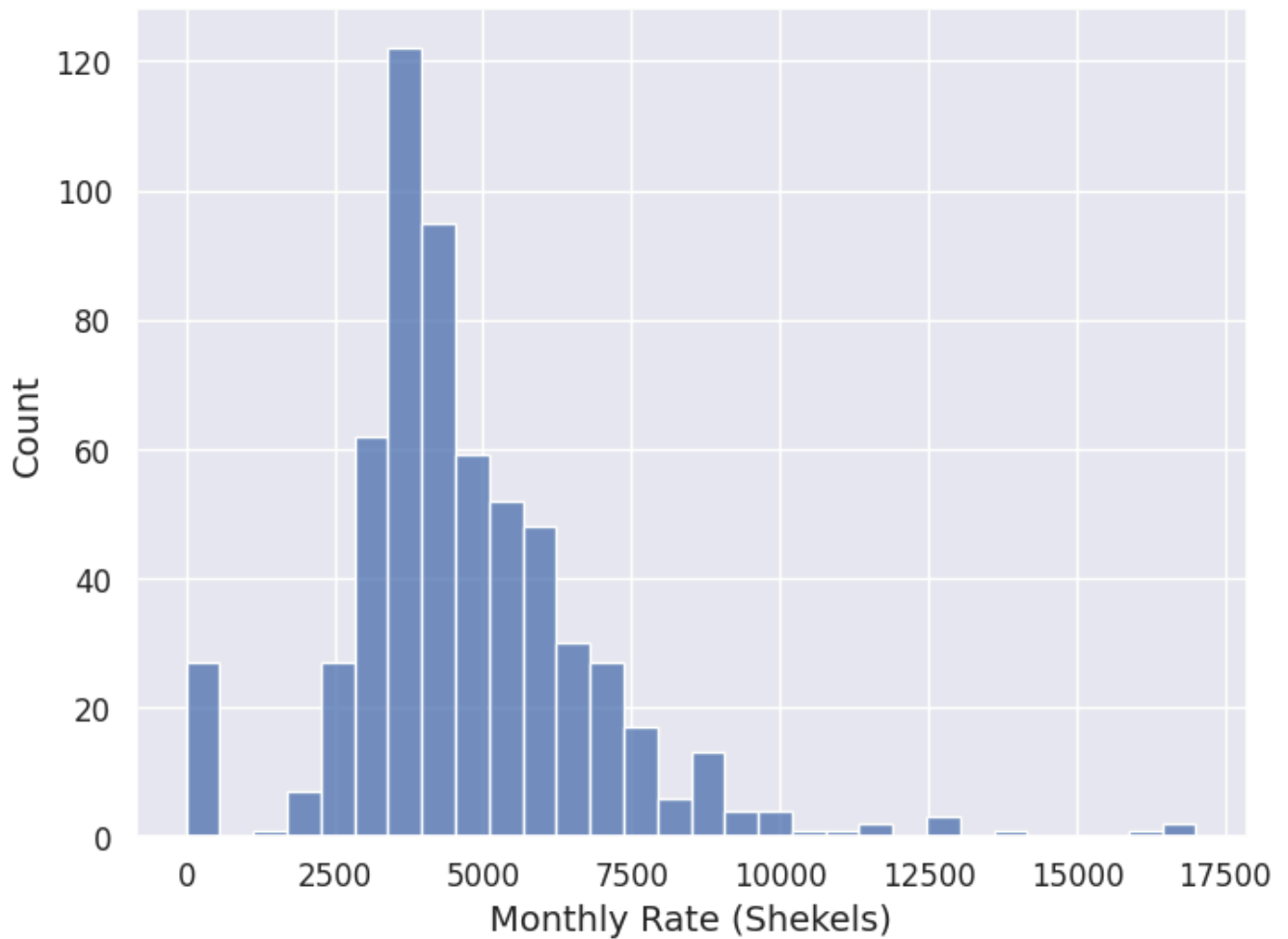
```
plt.figure(figsize=(10, 6))  
plt.hist(rent_df['monthlyRate'], bins=30, edgecolor='black')  
plt.title('Histogram of Monthly Rates')
```

↗ Text(0.5, 1.0, 'Histogram of Monthly Rates')



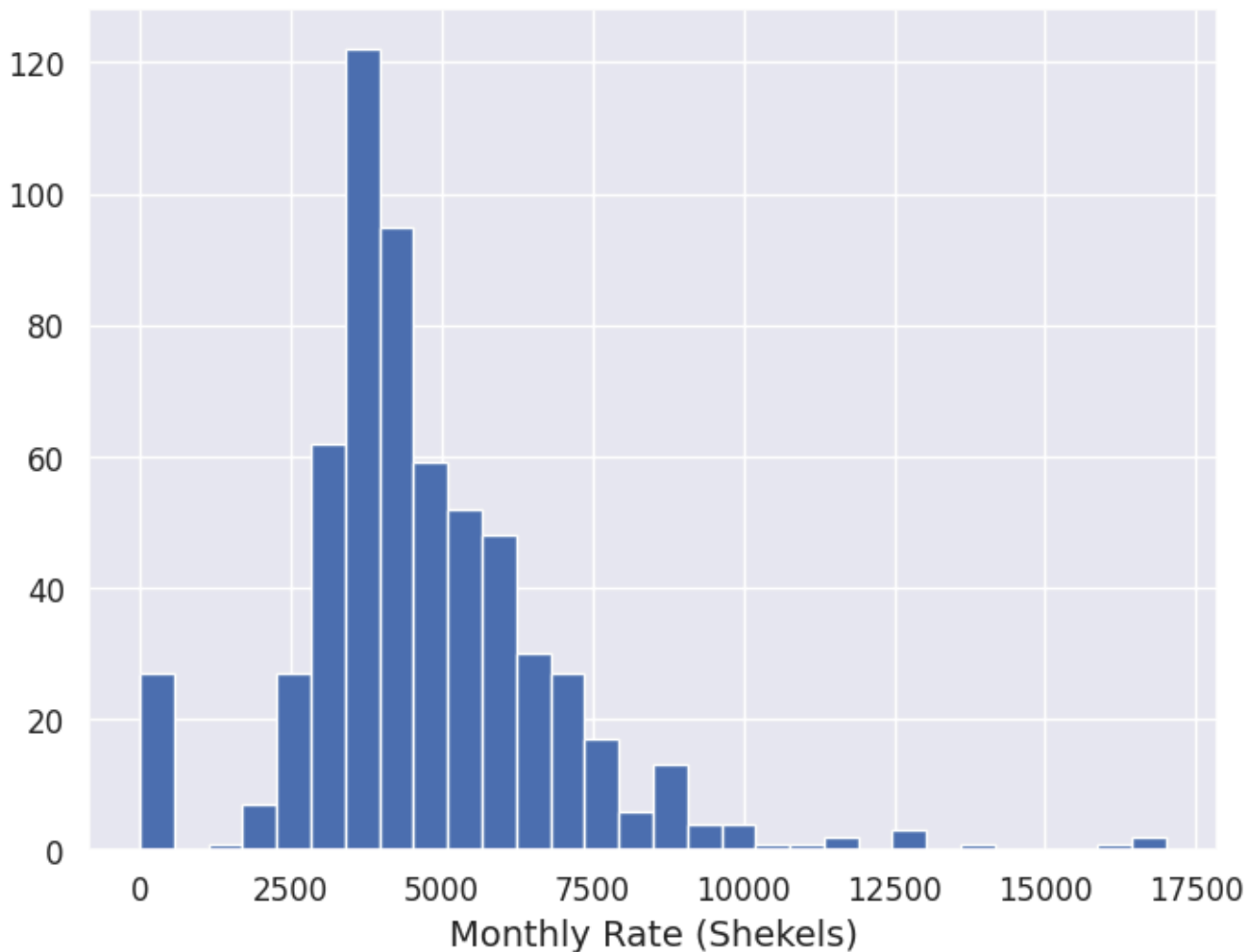
### ✓ Solution 1

```
# @title Solution 1
plt.figure(figsize=(8,6))
sns.histplot(rent_df['monthlyRate'], bins=30)
plt.xlabel("Monthly Rate (Shekels)");
```



## ✓ Solution 2

```
# @title Solution 2
rent_df["monthlyRate"].hist(bins=30, figsize=(8,6))
plt.xlabel("Monthly Rate (Shekels)");
```



We see that the prices distribution peaks around ~3500 Shekels and that it is right skewed, as there are some very expensive apartments. We can also see a peak at zero which makes sense as sometimes listings do not include a price. We would want to filter those out when we analyze prices later on.

Q: Print the number of listings that have no monthly rate:

## ✓ Solution

```
# @title Solution
print("Number of apartments without a price: ", rent_df['monthlyRate'].value_counts()[0].rou
```

➦ Number of apartments without a price: 25

We want to remove those listings, but we don't want to lose these entries, as we might want to know how many and what type of outliers we originally removed. So we create another dataframe that has the listings we removed and the reason for removal.

```
outlier_df = pd.DataFrame(columns=rent_df.columns.to_list()+['reason']) # will save the outl

outliers = rent_df[rent_df['monthlyRate'] <= 0].reset_index(drop=True)
outliers['reason'] = "monthlyRate <= 0"
outlier_df = pd.concat([outlier_df, outliers], axis=0, ignore_index=True).drop_duplicates().
outlier_df.tail()
```

➦ [Show hidden output](#)

We will now remove those listings and save the result to a new variable `clean_df`:

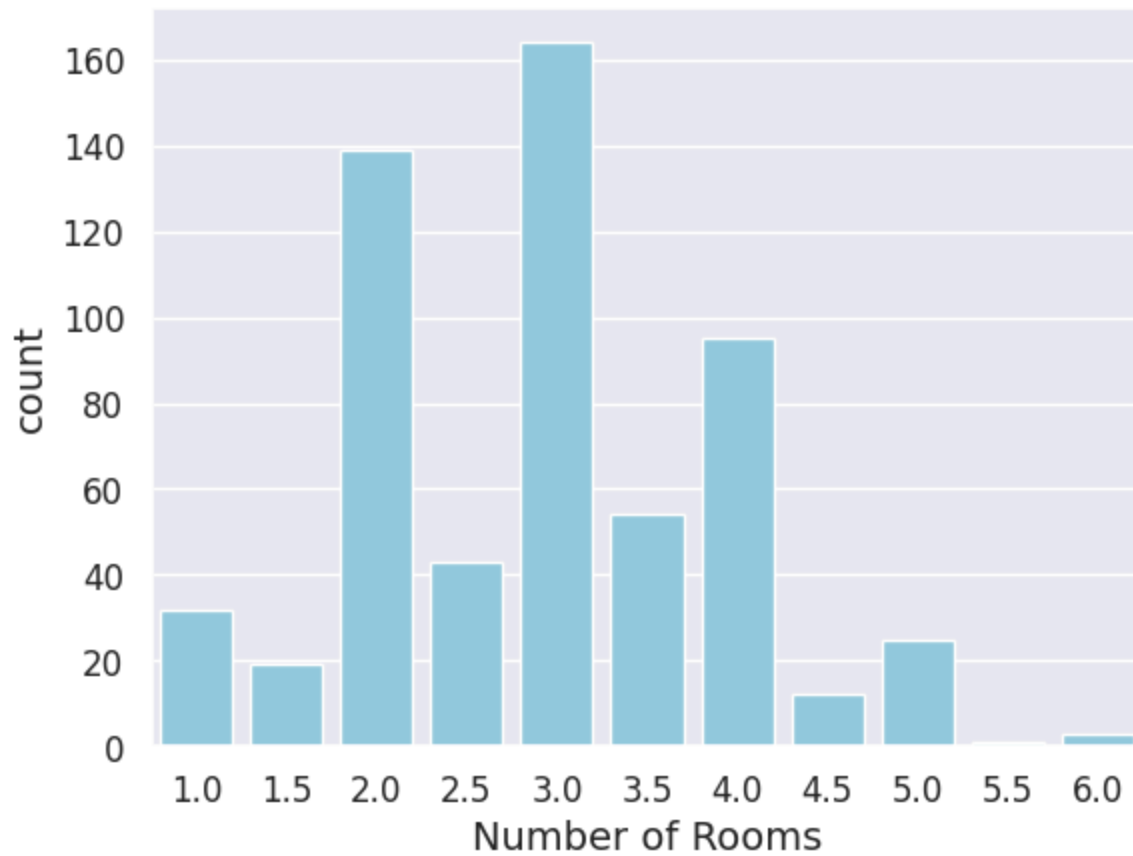
```
clean_df = rent_df[rent_df['monthlyRate'] > 0].reset_index(drop=True)
```

### What is the distribution of the number of rooms?

Q: Use `sns.countplot` to compare the counts of listings with different numbers of rooms. Plot all bars in the same [color](#) of your choice.

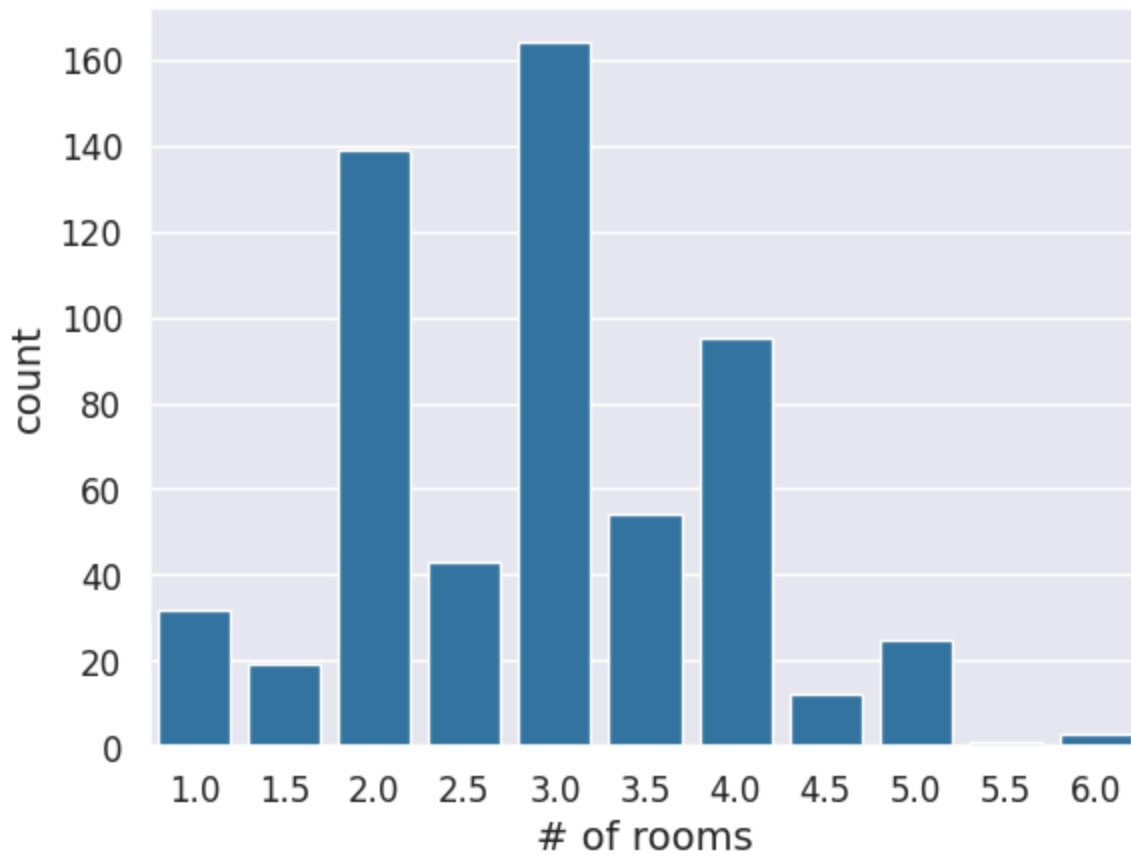
```
sns.countplot(x='rooms', data=clean_df, color='skyblue')
plt.xlabel('Number of Rooms')
```

Text(0.5, 0, 'Number of Rooms')



## ✓ Solution

```
# @title Solution
if clean_df is None:
    print("Can't run until 'clean_df' is created!")
else:
    sns.countplot(x='rooms', data=clean_df, color='tab:blue')
    plt.xlabel("# of rooms");
```

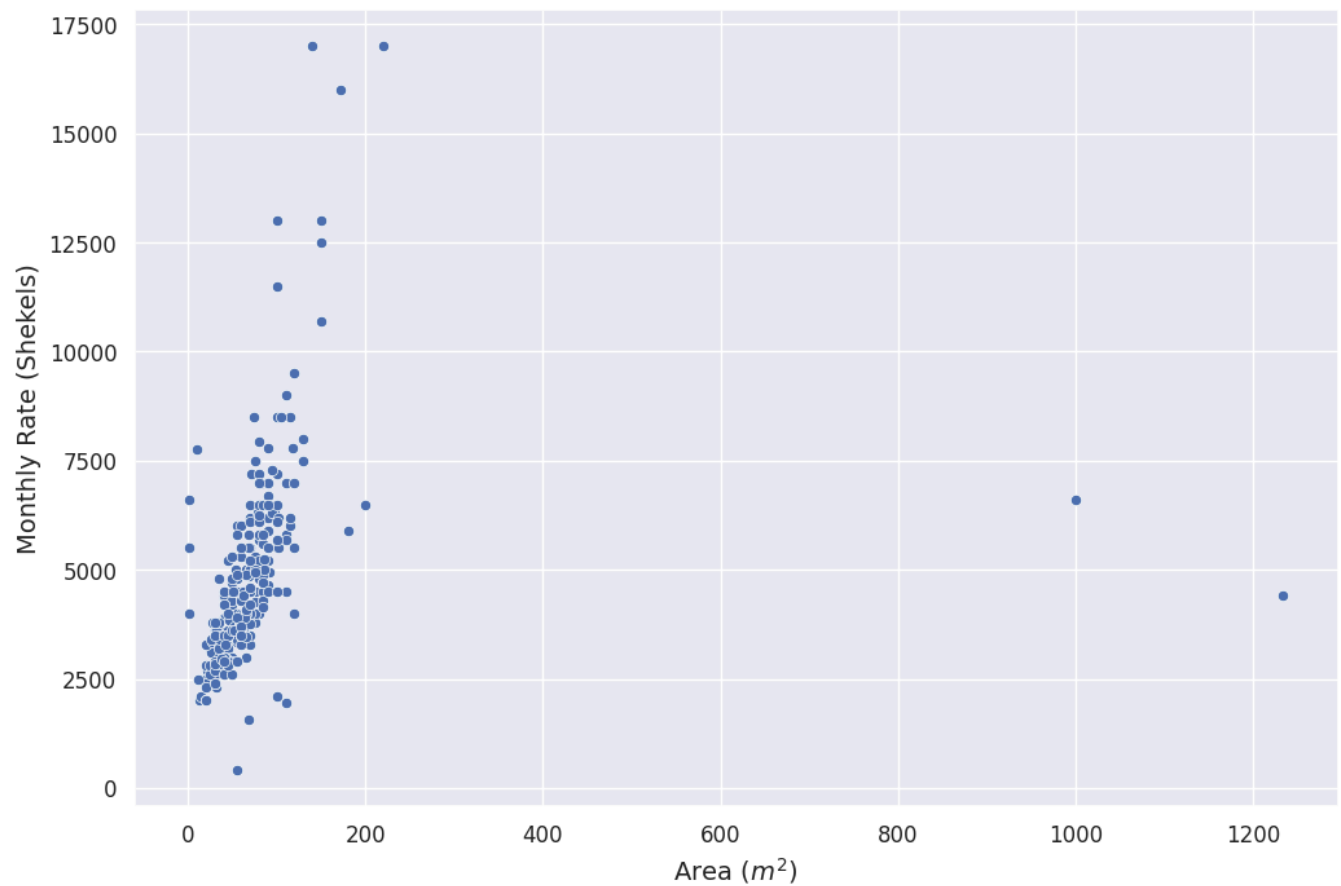


The distribution peaks at three rooms and we also see that "half rooms" are less common.

### Can we see an association between apartment area and price?

```
if clean_df is None:
    print("Can't run until 'clean_df' is created!")
else:
    plt.figure(figsize=(12,8))
    sns.scatterplot(x='area', y='monthlyRate', data=clean_df)
    plt.ylabel("Monthly Rate (Shekels)")
    plt.xlabel("Area ($m^2$)");
```





We see clear outliers here! We know that area is measured in squared meters and it is unlikely that there are any apartments of  $\sim 1000m^2$ .

Let's look at those samples to see if we can understand what happened there:

```
if clean_df is None:
    print("Can't run until 'clean_df' is created!")
else:
    display(clean_df.sort_values('area', ascending=False).head(4))
```



	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	de
185	3964340	תלפיות	4400.0	private	2.0	2.0	1234.0	10/08/2022	
543	3956561	זכרון משה	6600.0	private	3.5	3.0	1000.0	01/07/2022	



And inspect the description of one of those listings:

```
if clean_df is None:
    print("Can't run until 'clean_df' is created!")
else:
    display(clean_df.at[543,'description'])
```



דירה מהממת בלב ירושלים. צמודה לרכבת הקלה- תחנת הדוידקה. 3 חדרים ענקיים ולכל חדר מרפסת גדולה. חלל כניסה ע'  
'ח פיוח ישירה מחאימה מאוד ל- 3 שוחרים

Clearly not a 1000 m<sup>2</sup> apartment...

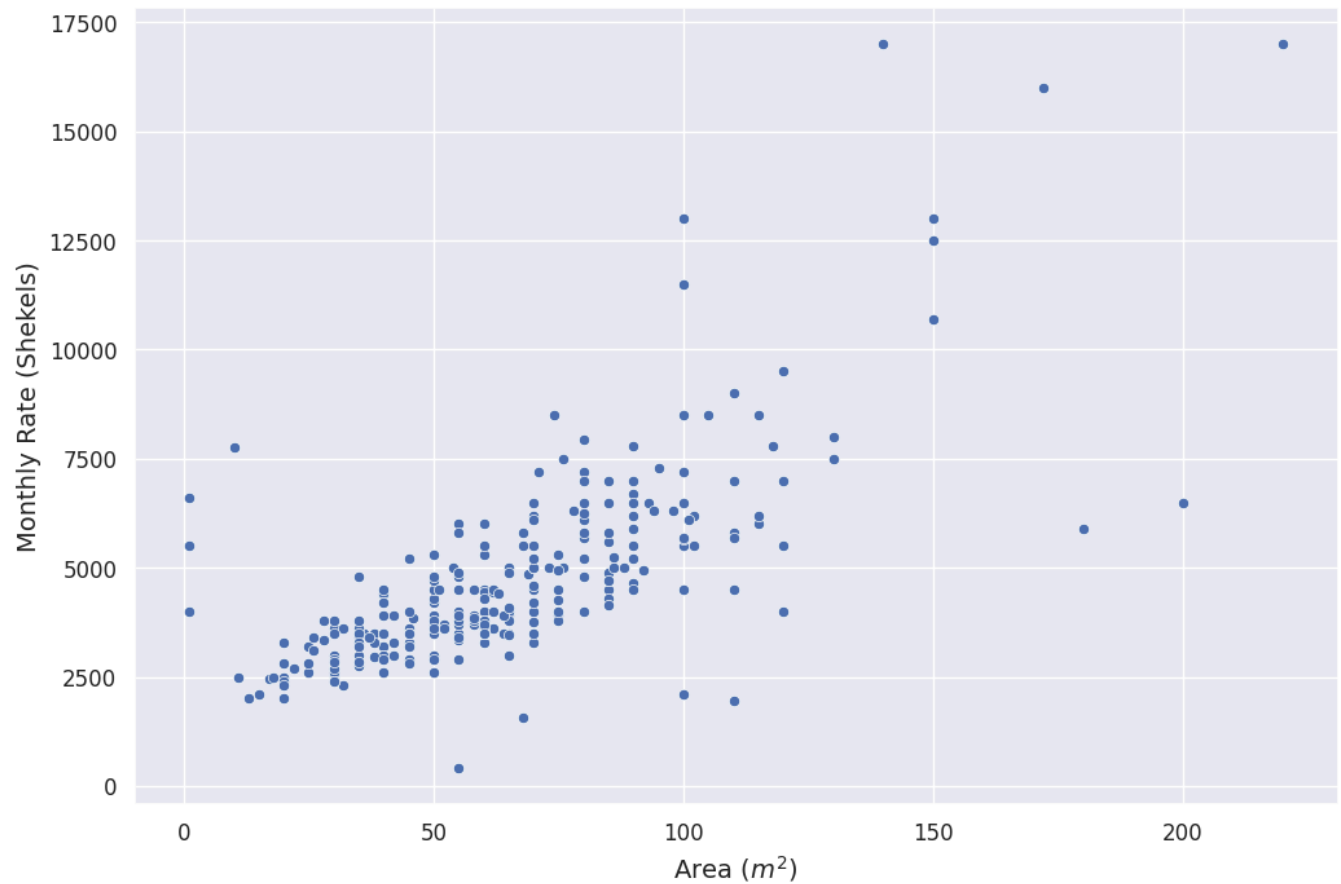
Q: Save a new dataframe named `clean_df_area_filtered` with all listings with area smaller than 800 m<sup>2</sup>. Again, add the removed outliers to the `outliers_df` dataframe.

Plot again the scatter of area vs. monthly rate after removing the outliers.

✓ Solution

```
# @title Solution
if clean_df is None:
    print("Can't run until 'clean_df' is created!")
elif outlier_df is None:
    print("Can't run until 'outlier_df' is created!")
else:
    # save outliers
    outliers = clean_df[clean_df['area'] >= 800].reset_index(drop=True)
    outliers['reason'] = "'area' >= 800"
    outlier_df = pd.concat([outlier_df, outliers], axis=0, ignore_index=True).drop_duplicates()

    # remove the outliers from the dataset
    clean_df_area_filtered = clean_df[clean_df['area'] < 800].reset_index(drop=True)
    plt.figure(figsize=(12,8))
    sns.scatterplot(x='area', y='monthlyRate', data=clean_df_area_filtered)
    plt.xlabel("Area ($m^2$)")
    plt.ylabel("Monthly Rate (Shekels)");
```



Again, we see some strange behavior of apartments with almost zero area but with a high monthly rate. Let's check them out:

We start with all apartments with an area between 0 to 25  $m^2$ :

```
# Show all apartments with area between 0 and 25
clean_df_area_filtered[clean_df_area_filtered['area'].between(0,25)]
```



	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	des
0	3994505	קריית יובל	2000.0	private	1.0	2.0	13.0	10/08/2022	חוב ה
1	3981298	רחביה	2450.0	private	1.0	1.0	17.0	10/08/2022	1 ה
3	3993997	בית וגן	2100.0	private	1.0	0.0	15.0	10/08/2022	15-: ו קט,
5	3993552	הר נוף	2000.0	private	1.0	0.0	20.0	10/08/2022	צת שרד 
6	3972039	גבעת שאול	2700.0	private	1.0	0.0	22.0	10/08/2022	יסה וועד
7	3988096	המושבה הגרמנית	2500.0	private	1.0	0.0	18.0	10/08/2022	ם זיות
8	3992809	נחלאות	3200.0	private	1.0	2.0	25.0	10/08/2022	י ה
10	3983516	הגבעה הצרפתית	2000.0	private	1.0	2.0	20.0	10/08/2022	בת: ויד



Some make sense and others do not. Let's focus on the expensive ones (between 5,000 and 10,000 shekels):

```
# Show all apartments with area between 0 and 25 that also have a price between 5000 and 10000
if clean_df_area_filtered is None:
    print("Can't run until 'clean_df_area_filtered' is created!")
else:
    display(clean_df_area_filtered[clean_df_area_filtered['area'].between(0,25) & clean_df_area_filtered['price'].between(5000,10000)])
```



	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	des
197	3984483	ארנונה	6600.0	private	4.0	2.0	1.0	01/09/2022	ימים



Those are clearly wrong too... Besides that the relationship between the area and the price seems linear. Let's remove these outliers too:

```
#remove the outliers
if clean_df_area_filtered is None:
    print("Can't run until 'clean_df_area_filtered' is created!")
elif outlier_df is None:
    print("Can't run until 'outlier_df' is created!")
else:
    non_outliers = clean_df_area_filtered['area'] > 10 # get non outliers series of true/false

    # save outliers
    outliers = clean_df_area_filtered[~non_outliers].reset_index(drop=True) # get the outliers
    outliers['reason'] = "'area' <= 10"
    outlier_df = pd.concat([outlier_df, outliers], axis=0, ignore_index=True).drop_duplicates()

    # remove them
    clean_df_area_filtered = clean_df_area_filtered[non_outliers].reset_index(drop=True)
```

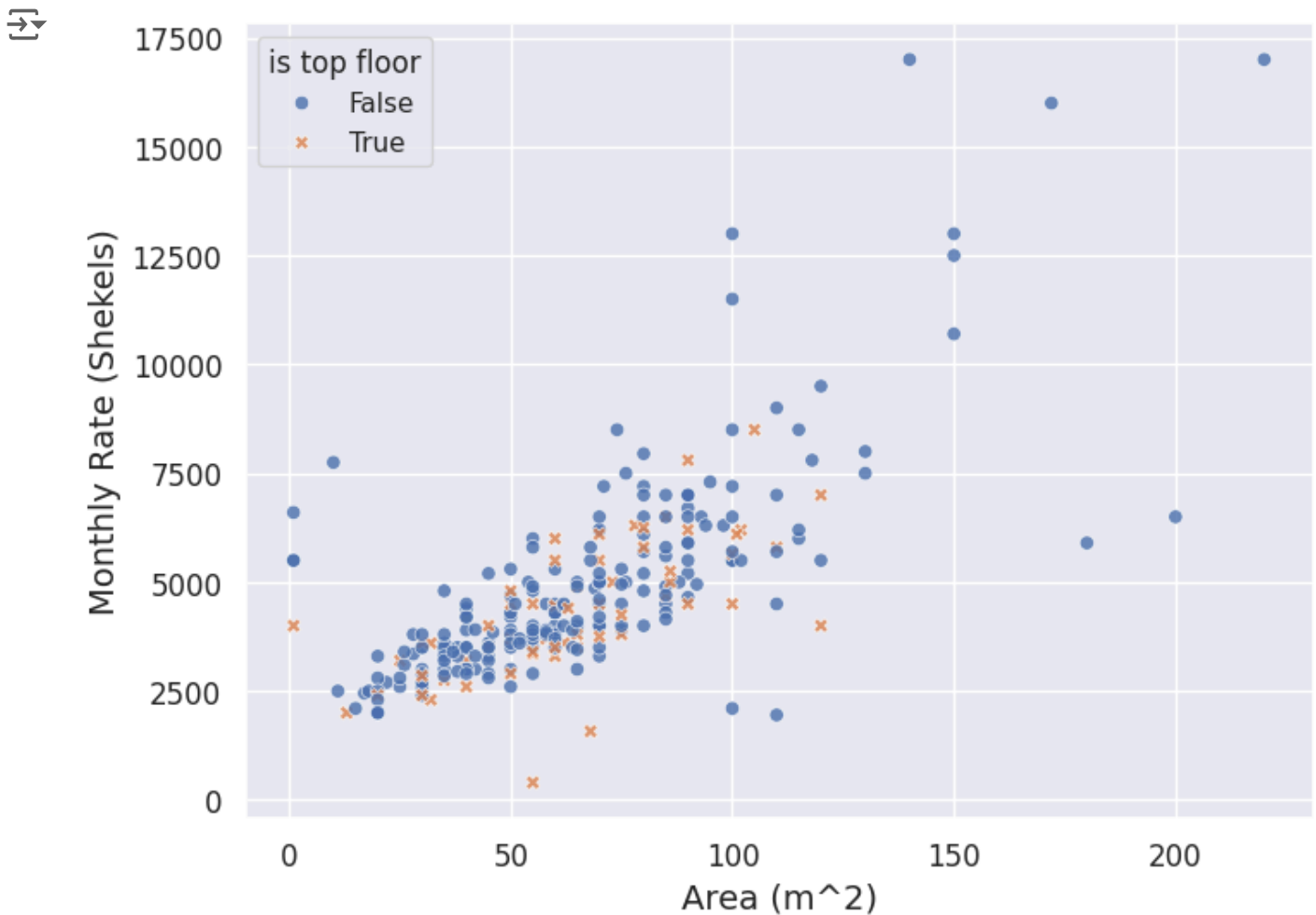
### ✓ Can we see a different pattern for top floor apartments?

Q: Plot again a scatter of area vs. monthly rate. This time distinguish (by color / marker style or both) between apartments that are in the top floor and the rest of the apartments. (To do that you should create a new column in `clean_df_area_filtered` called `is_top_floor` and set it to 1 if the apartment is in the top floor and 0 otherwise.)

### ✓ Solution

```
# @title Solution
```

```
if clean_df_area_filtered is None:
    print("Can't run until 'clean_df_area_filtered' is created!")
else:
    clean_df_area_filtered['is top floor'] = clean_df_area_filtered['floor'] == clean_df_area_
    plt.figure(figsize=(8,6))
    sns.scatterplot(x='area', y='monthlyRate', data=clean_df_area_filtered, alpha=0.8, hue='is
    plt.xlabel("Area (m^2)")
    plt.ylabel("Monthly Rate (Shekels)");
```



We can take a deeper look on the apartments with the very high monthly rate (to see if those are outliers or not):

```
if clean_df_area_filtered is None:
    print("Can't run until 'clean_df_area_filtered' is created!")
else:
    display(clean_df_area_filtered[clean_df_area_filtered['monthlyRate'] > 11000])
```



	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	des
200	3956418	רחביה	13000.0	agent	4.0	1.0	100.0	NaN	
240	3985051	טלביה	17000.0	private	4.0	4.0	140.0	10/08/2022	4

We can see some representation of the more expensive neighborhoods of Jerusalem here.. More on the neighborhoods later on!

### Is there also a relation between the number of rooms and the listing price?

Q: Create a visualization that compares the distribution of prices for different number of rooms. Your visualization should provide information about central tendency (mean/median/mode) and some information about the distribution of individual values around it (standard deviation/interquartile range) for each number of rooms. Also, show the real prices of the listings per number of rooms.

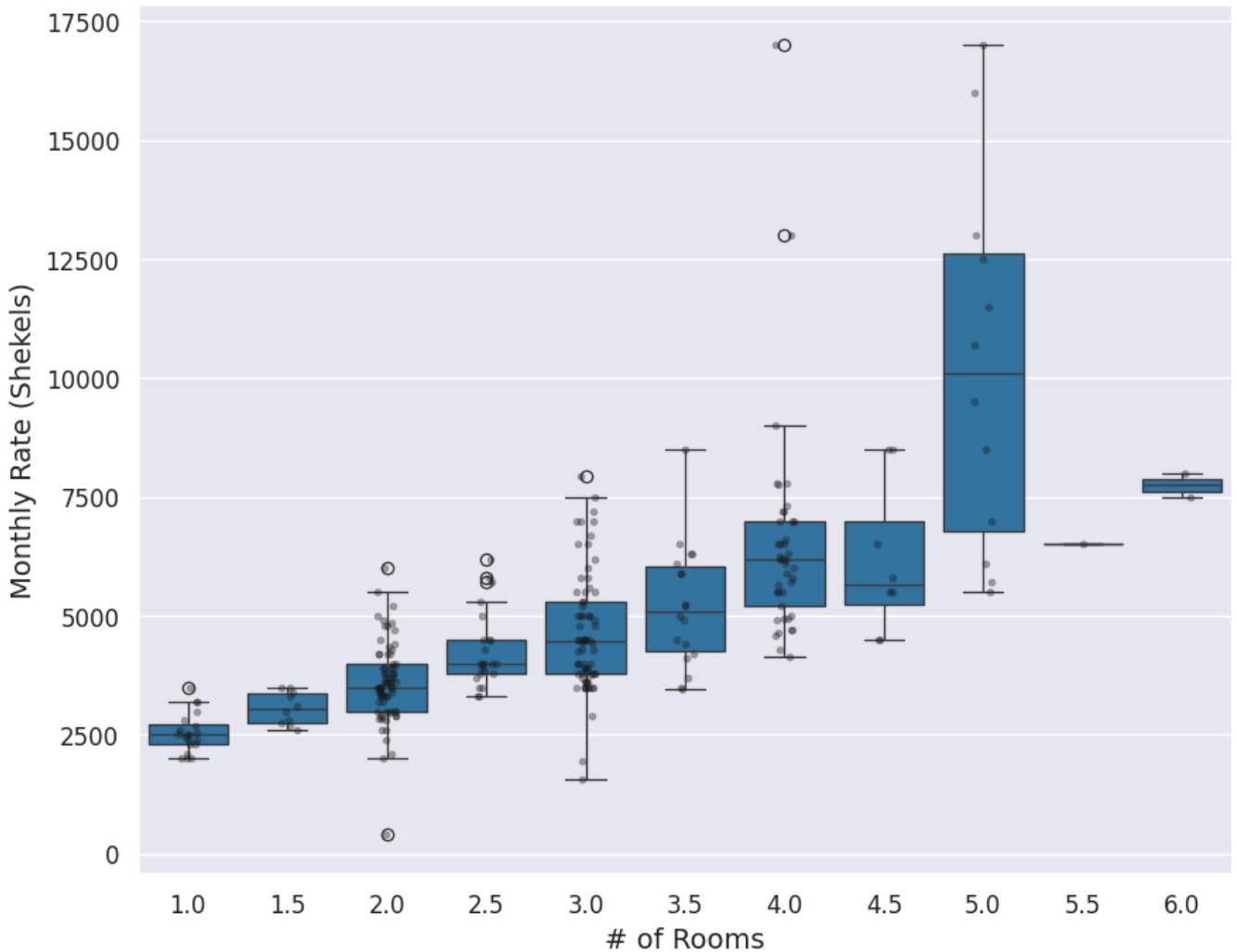
### ✓ Solution



```
# @title Solution
if clean_df_area_filtered is None:
    print("Can't run until 'clean_df_area_filtered' is created!")
else:
    plt.figure(figsize=(10,8))
    sns.boxplot(x='rooms', y='monthlyRate', data=clean_df_area_filtered, color='tab:blue')
    sns.stripplot(x='rooms', y='monthlyRate', alpha=0.4 ,size=4,color='k',data=clean_df_area_f
    plt.xlabel("# of Rooms")
    plt.ylabel("Monthly Rate (Shekels)");

# Or:
# plt.figure(figsize=(10,8))
# sns.barplot(x='rooms', y='monthlyRate', data=clean_df_area_filtered, color='tab:blue', e
# # Can also use mean but median is more informative in this case as prices are skewed...
# sns.stripplot(x='rooms', y='monthlyRate', alpha=0.4 ,color='k',data=clean_df_area_filter
# plt.xlabel("# of Rooms")
# plt.ylabel("Monthly Rate (Shekels)");

#Violin plot completely fails for very small subsets:
# plt.figure(figsize=(10,8))
# sns.violinplot(x='rooms', y='monthlyRate', data=clean_df_area_filtered, color='tab:blue'
# plt.xlabel("# of Rooms")
# plt.ylabel("Monthly Rate (Shekels)");
```



Now that we finished pre-processing the data, we can see the state of our outliers VS the data that remains:

```
if outlier_df is None:
    print("Can't run until 'outlier_df' is created!")
else:
    # describe the outlier data
    display(outlier_df.groupby('reason').describe())
    print(f"Proportion removed: {100*len(outlier_df) / (len(outlier_df)+len(clean_df_area_filt
```



	monthlyRate								rooms	
	count	mean	std	min	25%	50%	75%	max	count	mean
reason										
'area' >= 800	2.0	5500.0	1555.634919	4400.0	4950.0	5500.0	6050.0	6600.0	2.0	2.75
monthlyRate <= 0	25.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	25.0	3.25

2 rows × 11 columns

## ✓ Submission Exercises

### ✓ Part 1: Diving deeper into rental prices

We will create a copy of the dataset and work on that. We want to make sure that we do not modify the original dataset.

#### ✓ Part 1 - Create a DataFrame

```
# @title Part 1 - Create a DataFrame
part1_df = rent_df_backup_for_exercise.copy()
```

Let's go back to the distribution of monthly rental prices in the dataset. Are there interesting trends in the distribution that we missed in the visualizations before?

**Use only `part1_df` for the coding questions in this part**

#### ✓ Question 1

Plot 3 different histograms of the monthly prices with 20, 60 and 120 bins respectively, each in a different axis/figure.

```
# Part 1 - Question 1
```

```
# 20 bins
```


```
plt.figure(figsize=(10,5))  
sns.histplot(part1_df['monthlyRate'], bins=20)  
plt.title("20 bins")
```

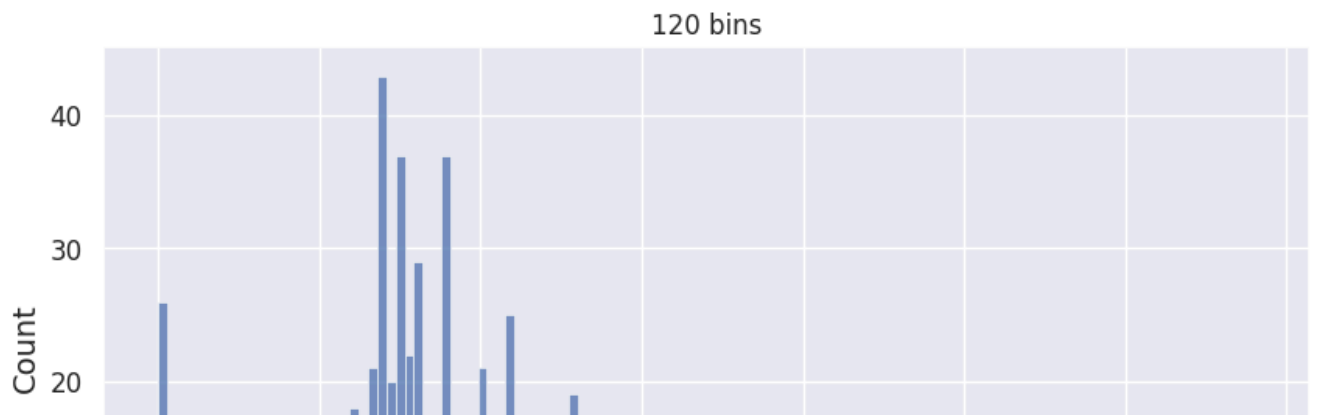
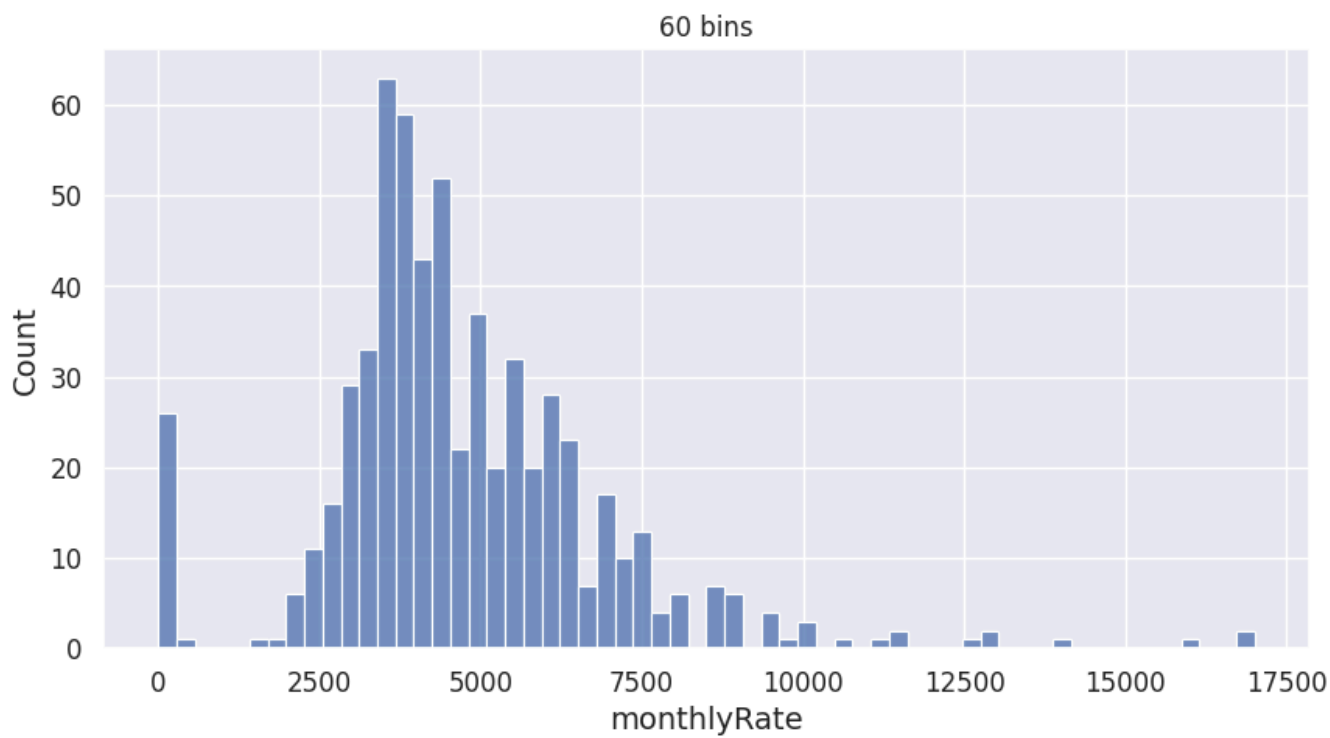
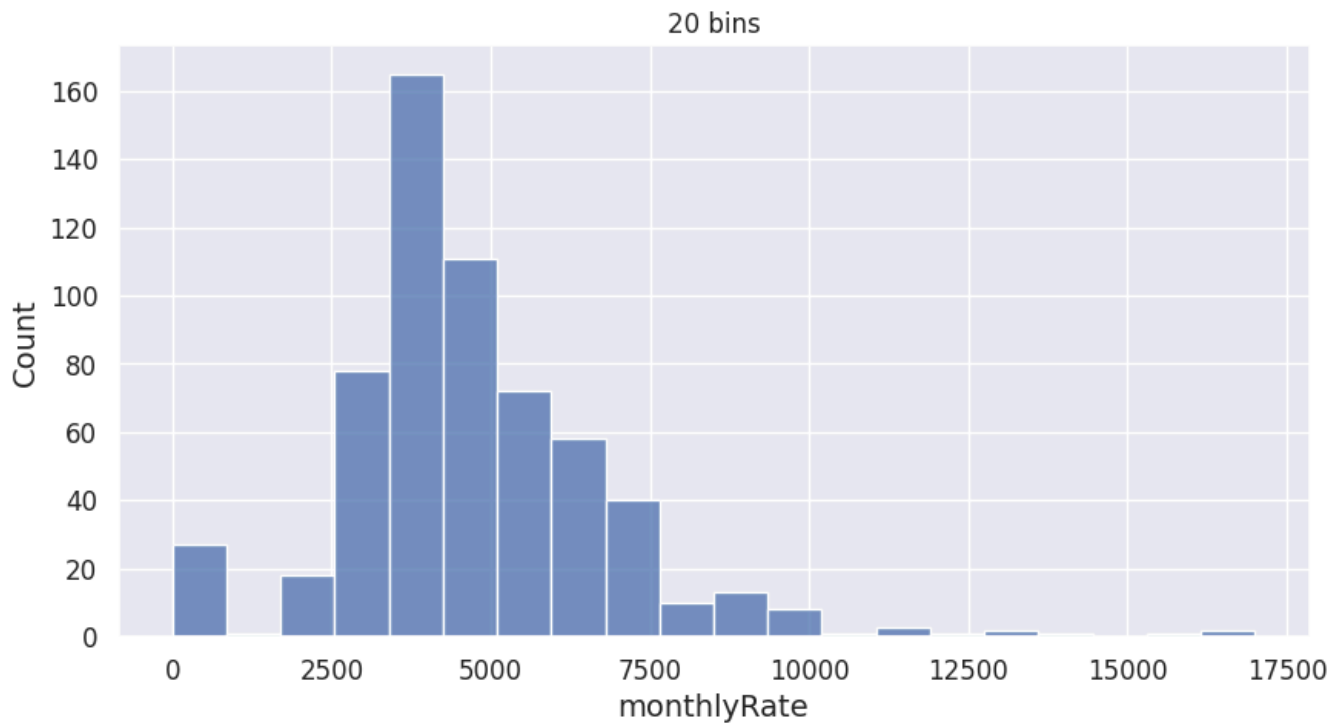
```
# 60 bins
```

```
plt.figure(figsize=(10,5))  
sns.histplot(part1_df['monthlyRate'], bins=60)  
plt.title("60 bins")
```

```
# 120 bins
```

```
plt.figure(figsize=(10,5))  
sns.histplot(part1_df['monthlyRate'], bins=120)  
plt.title("120 bins")
```

 Text(0.5, 1.0, '120 bins')





## ✓ Question 2

For 60 and 120 bins, you can see a repeating pattern of "peaks" and "vallies" in the distribution (mostly in the range between 500 and 7000). Is this pattern due to people rounding the rental prices? Please create a visualization that answers this question. Describe in words how the graph shows what the answer is (Hint: you can use the '%' operator to compute the remainder of dividing values in a pandas Series by a scalar number).

- **extra hint:** please open this cell only after discussing with the course staff the best solution you could come up with

[Show code](#)

```
# Part 1 - Question 2
```

```
# Your code goes here:
```

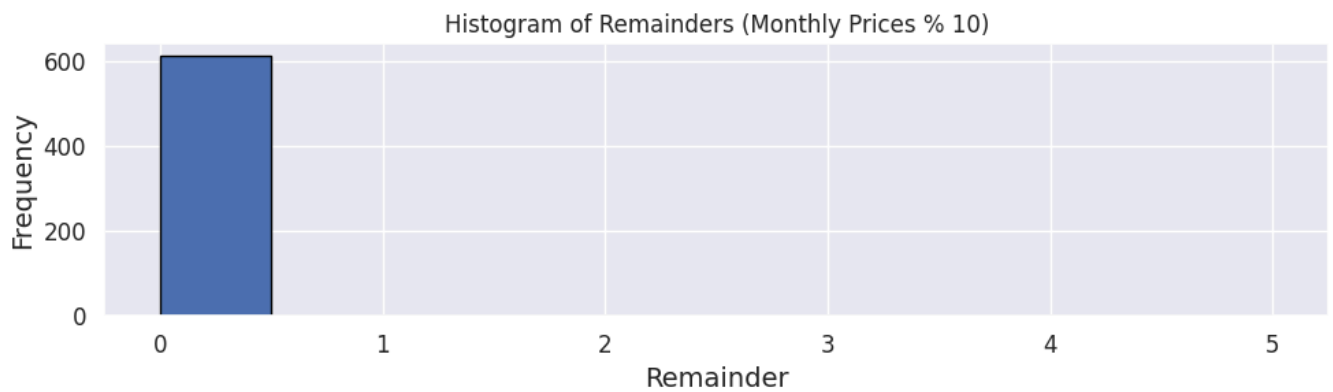
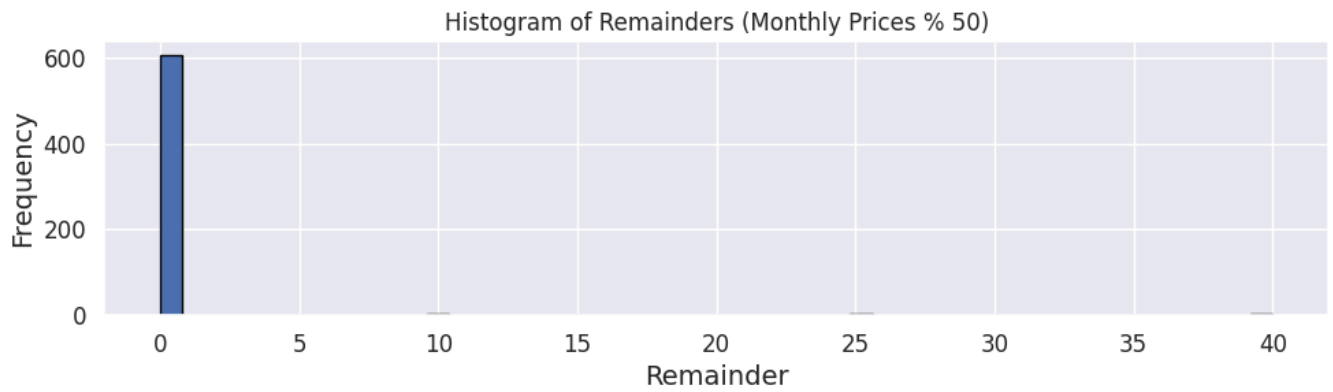
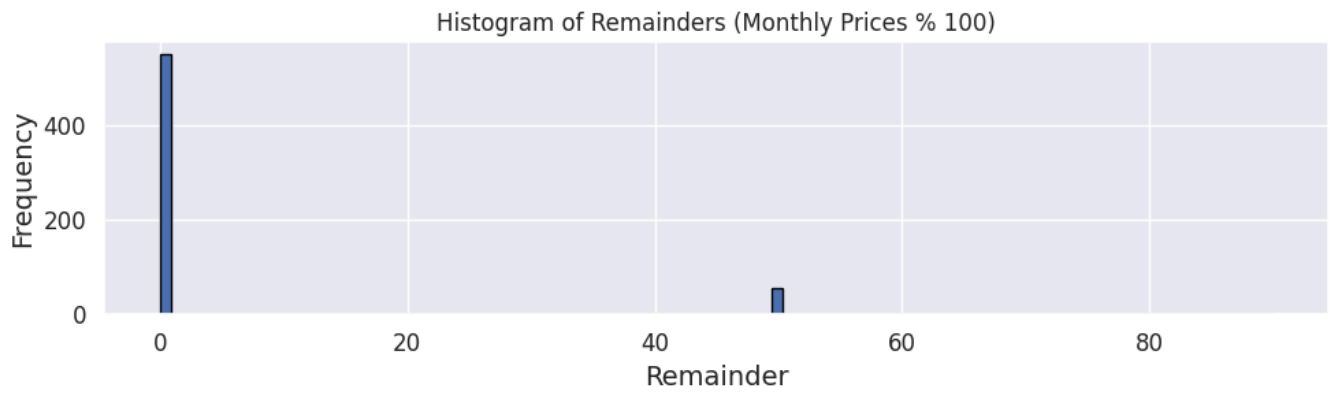
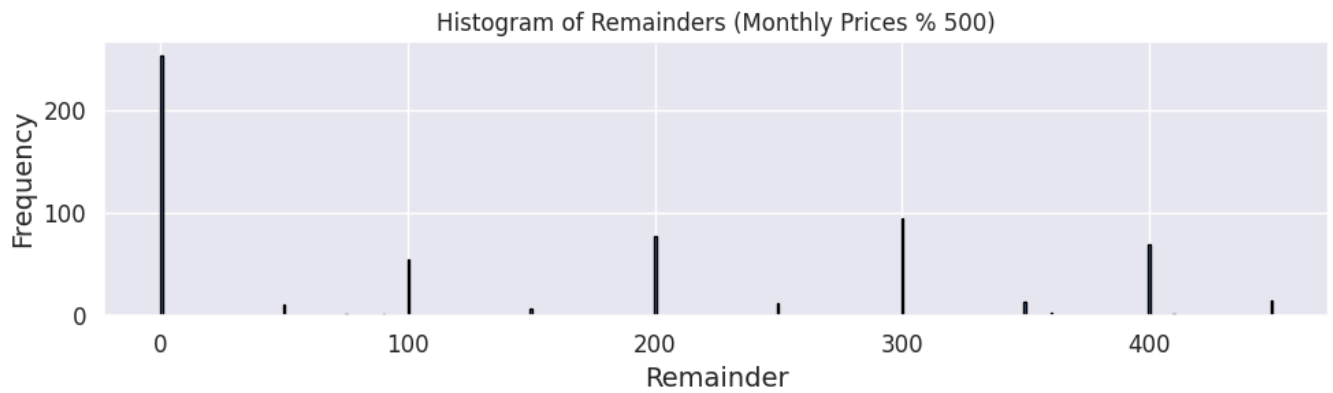
```
part1_df['remainder_10'] = part1_df['monthlyRate'] % 10
part1_df['remainder_50'] = part1_df['monthlyRate'] % 50
part1_df['remainder_100'] = part1_df['monthlyRate'] % 100
part1_df['remainder_500'] = part1_df['monthlyRate'] % 500
# Create subplots
fig, axes = plt.subplots(4, 1, figsize=(10, 12))
#divided by 500
axes[0].hist(part1_df['remainder_500'], bins=500, edgecolor='black')
axes[0].set_title('Histogram of Remainders (Monthly Prices % 500)')
axes[0].set_xlabel('Remainder')
axes[0].set_ylabel('Frequency')

# divided by 100
axes[1].hist(part1_df['remainder_100'], bins=100, edgecolor='black')
axes[1].set_title('Histogram of Remainders (Monthly Prices % 100)')
axes[1].set_xlabel('Remainder')
axes[1].set_ylabel('Frequency')

# divided by 50
axes[2].hist(part1_df['remainder_50'], bins=50, edgecolor='black')
axes[2].set_title('Histogram of Remainders (Monthly Prices % 50)')
axes[2].set_xlabel('Remainder')
axes[2].set_ylabel('Frequency')

# divided by 10
axes[3].hist(part1_df['remainder_10'], bins=10, edgecolor='black')
axes[3].set_title('Histogram of Remainders (Monthly Prices % 10)')
axes[3].set_xlabel('Remainder')
axes[3].set_ylabel('Frequency')

plt.tight_layout()
```





## ✓ Part 1 Question 2 - textual Answer:

*Write your answer here:*

כדי לבדוק האם יש הטיה לעיגול סכומי שכר הדירה לקחנו את הערכים וחילקנו במספר ערכים. תחילה, חילקנו ב-500 וראינו כי לא ניתן כלל להסיק כי רוב הערכים מעוגלים, כיוון שיש הרבה סוגים של שאריות. בהמשך, חילקנו ב-100 והצגנו את היסטוגרמת שארית החלוקה. ההיסטוגרמה הראתה שרוב הערכים אכן מתחלקים ב-100 ללא שארית, פרט לכמה ערכים שישבו שהשארית שלהם 50. ביצענו את אותה הפעולה שוב וחילקנו ב-50, וראינו שרוב הערכים מתחלקים ב-50 ללא שארית, פרט לכמה בודדים שמתחלקים ב-10. חילקנו ב-10 וראינו שכל הערכים מתחלקים בו ללא שארית. אנו מסיקים מכך שיש נטייה לעיגול הערכים. ניתן לראות בגרפים שרוב השאריות הן 0 ולכן ניתן להסיק כי רוב האנשים מעגלים. בנוסף, ניתן לראות כי ככל שהמספר עולה יש יותר שכיחויות בשאריות ומכאן ניתן להסיק כי אנשים מעגלים למספרים שלמים קרובים יחסית ולא בקפיצות מאוד גדולות (כמו 500).

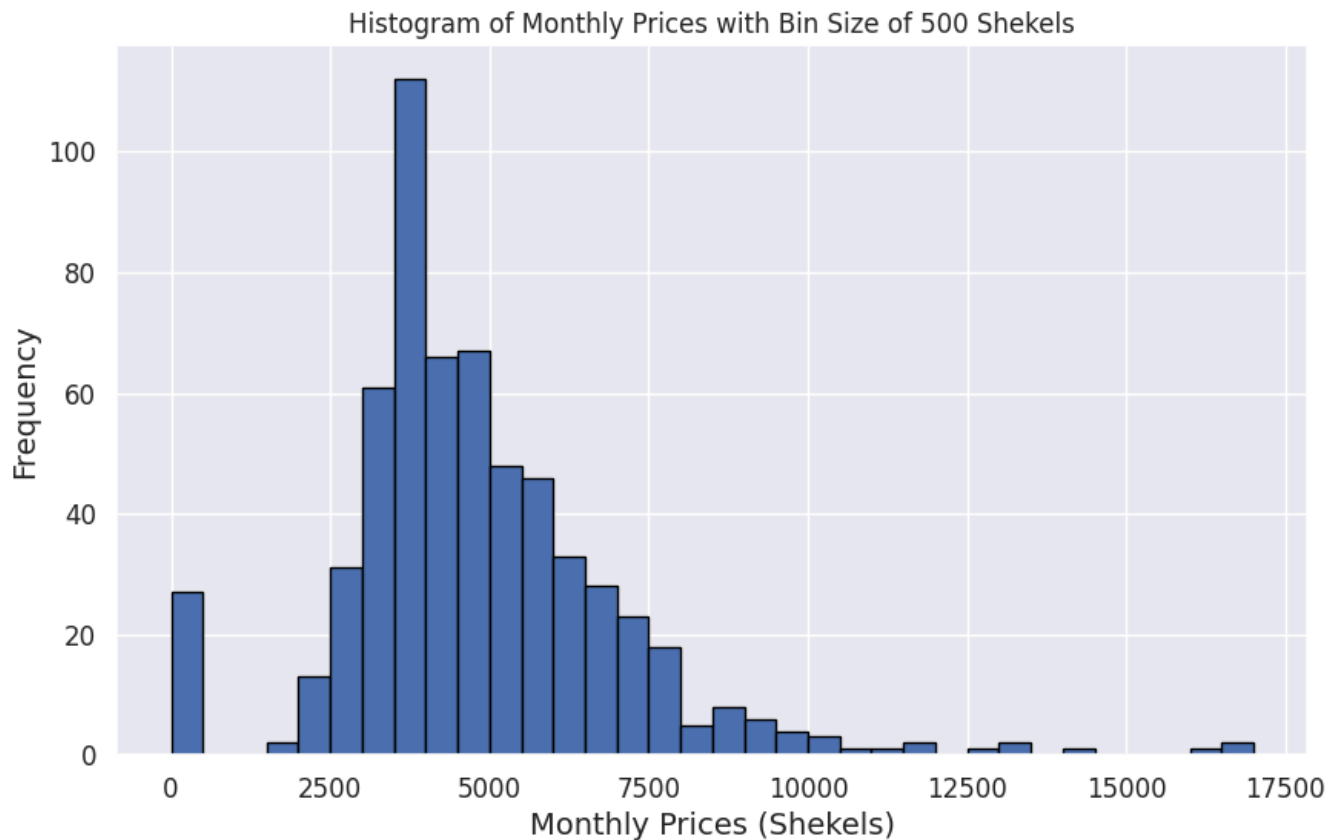
## ✓ Question 3

We expect to see a "drop" in prices frequency near the 5000 Shekels mark due to tax considerations (See [here](#) for an explanation). Create a histogram visualization of the data with the smallest possible bins such that every bin will include exactly one multiplication of 500 (Hint: read the `bins` parameter documentation and what types it accepts). Explain why does this choice of bin size ensures that we will not see rounding effects. Do you see a "drop" around 5000 Shekels? Are there other "drops"?

```
# Part 1 - Question 3
# Your code goes here:
```

```
min_price = part1_df['monthlyRate'].min()
max_price = part1_df['monthlyRate'].max()
bin_edges = np.arange(min_price, max_price + 500, 500)

# Plot the histogram
plt.figure(figsize=(10, 6))
plt.hist(part1_df['monthlyRate'], bins=bin_edges, edgecolor='black')
plt.title('Histogram of Monthly Prices with Bin Size of 500 Shekels')
plt.xlabel('Monthly Prices (Shekels)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



### ✓ Part 1 Question 3 - textual Answer:

*Write your answer here:*

ראשית, ניתן לראות בשאלה 2 שכאשר חילקנו את הערכים ב-500 לא הייתה עדות לנטייה לעיגול ערכים. לכן, כאשר הבינים בגודל 500 ניתן להתעלם מתופעת עיגול הערכים. ניתן לראות עוד דרופ באזור 7500 ש"ח. אנו משערים כי זה קשור לחוקי מס ע"פ הפסקה הבאה: "בעלי דירה יחידה שמשכירים אותה למגורים, ובמקביל שוכרים דירה אחרת, ואינם זכאים לפטור או לא ממשים אותו, יכולים להפחית לצרכי מס את סכום השכירות שהם משלמים כשוכרים (עד תקרה של 7,500 ש"ח) מסכום השכירות שהם מקבלים כמשכירים, ולשלם מס בשיעור 10% רק על ההפרש ולא על מלוא דמי השכירות שהם מקבלים."

### ✓ Part 2: Size or number of rooms?

## ✓ Part 2 - Create a DataFrame for Part 2

```
# @title Part 2 - Create a DataFrame for Part 2
```

```
# Create the dataframe and remove the outliers we found in the intro part:
part2_df = rent_df_backup_for_exercise.copy()
part2_df = part2_df[part2_df['monthlyRate'] > 0].reset_index(drop=True);
part2_df = part2_df[part2_df['area'] < 800].reset_index(drop=True)
part2_df = part2_df[part2_df['area'] > 10].reset_index(drop=True)
```

We saw that both the number of rooms and the area of an apartment are strongly associated with the monthly rate. We now want to check if those are just two perspectives of the same relation (how big is the apartment) or is there something more to it. We will use the cleaned dataframe for this exercise.

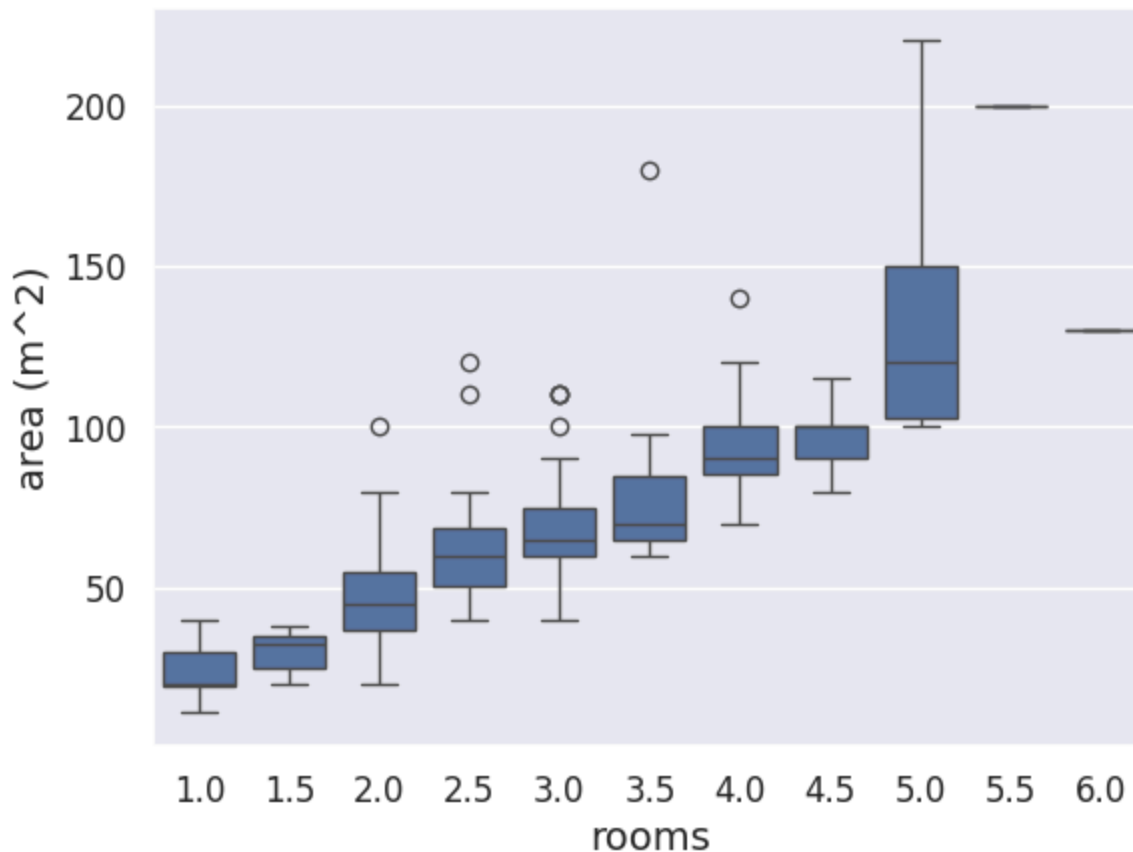
**Use only `part2_df` for the coding questions in this part**

## ✓ Question 1

Generate a visualization to show that there is a strong association between the number of rooms and the area of the apartment. Explain your choice of plot type and your conclusion from the graph.

```
# Part 2 - Question 1
# Your code goes here:
#
#
sns.boxplot(x='rooms', y='area', data=part2_df)
plt.xlabel("rooms")
plt.ylabel("area (m^2)")
```

```
Text(0, 0.5, 'area\xa0(m^2)')
```



## ✓ Part 2 Question 1 - textual Answer:

*Write your answer here:*

הצגנו את הדאטא בוויסקר פלוט, כדי להציג את התפלגות השטח של כל דירה לפי קטגוריות של מספר החדרים. נראה בגרף כי אין הרבה תצפיות חריגות, החציון שלל כל התפלגות נמצא מעל לחציון שקדם לו, משמאל לימין. כלומר, רוב הדירות בכל קטגוריה של חדרים הראו מגמה לפיה ככל שמספר החדרים עולה כך גם שטח הדירה. לפיכך, ניתן להסיק קשר לינארי חזק.

## ✓ Question 2

Add a new column to the dataframe named "averageRoomSize" with the average room size in the given listing.

# Part 2 - Question 2

# Your code goes here:

```
part2_df['averageRoomSize'] = part2_df['area'] / part2_df['rooms']
print(part2_df.head())
```

```

↩
propertyID neighborhood monthlyRate mefarsem rooms floor area \
0 3994505 2000.0 יובל קריית private 1.0 2.0 13.0
1 3981298 2450.0 רחביה private 1.0 1.0 17.0
2 3981623 2550.0 מלחה private 1.0 0.0 30.0
3 3993997 2100.0 בית וגן private 1.0 0.0 15.0
4 3994399 2300.0 פסגת זאב private 1.0 1.0 32.0

entry description numFloors \
0 10/08/2022 2.0 יחידת דיור להשכרה ברחוב הראשי של קריית יובל, ה...
1 10/08/2022 3.0 דירת יחיד 17 מטר כולל מרפסת קטנה
2 10/08/2022 2.0 דירה יפה ומטופחת, לדיירת שקטה לטווח ארוך, ללא ...
3 10/08/2022 3.0 דירת חדר, כ-15 מ"ר, במיקום מרכזי אך שקט, משופצ...
4 10/08/2022 1.0 בס"ד בפסגת זאב מזרח דירת חדר גדולה משופצת ויפ...

averageRoomSize
0 13.0
1 17.0
2 30.0
3 15.0
4 32.0
```

### ✓ Question 3

Create a plot of the relation between the average room size and the monthly rate.

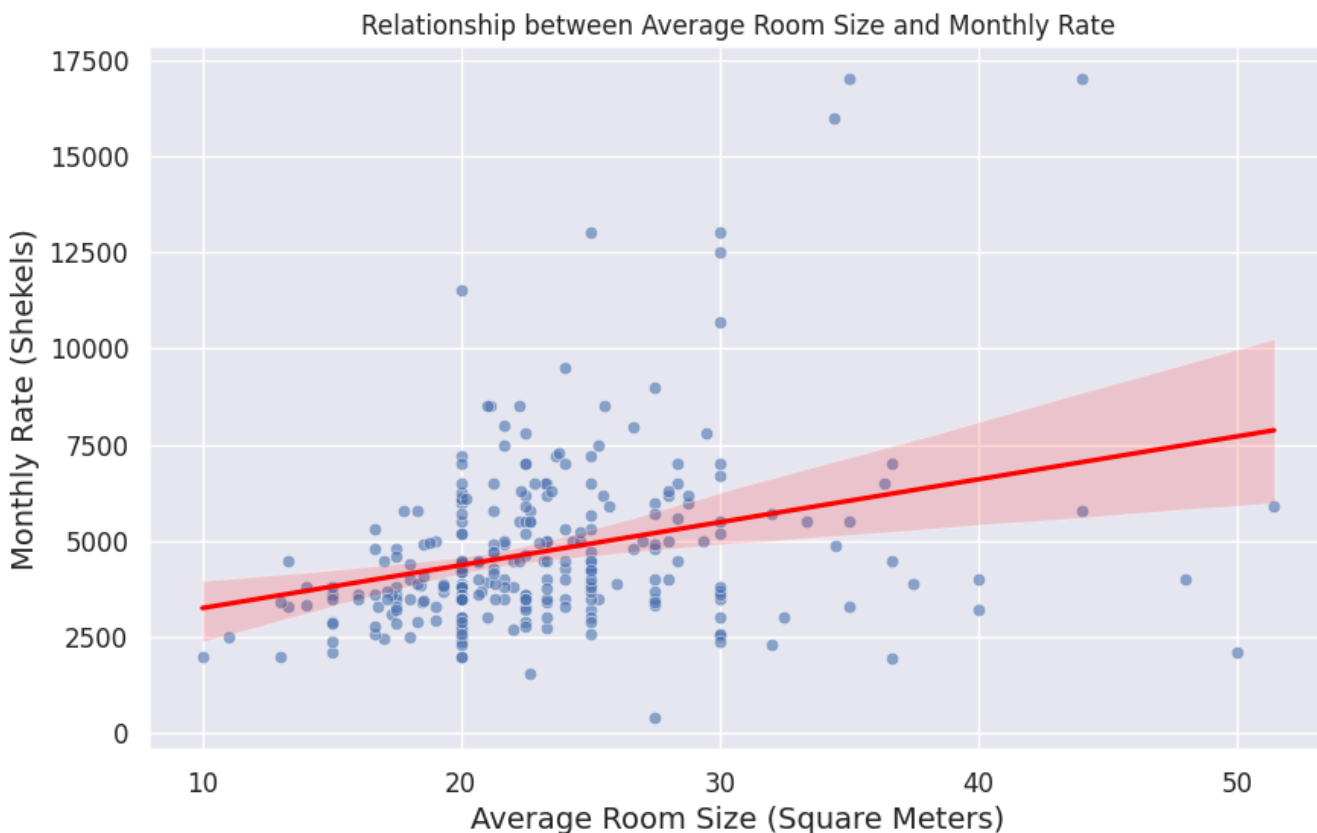
# Part 2 - Question 3

# Your code goes here:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='averageRoomSize', y='monthlyRate', data=part2_df, alpha=0.6)

# Add a regression line to highlight the trend
sns.regplot(x='averageRoomSize', y='monthlyRate', data=part2_df, scatter=False, color='red')

plt.title('Relationship between Average Room Size and Monthly Rate')
plt.xlabel('Average Room Size (Square Meters)')
plt.ylabel('Monthly Rate (Shekels)')
plt.grid(True)
plt.show()
```



#### ✓ Question 4 - **bonus**

We can see that the variance of the monthly rate increases with the average room size.

Suggest what might be the reason for the increase in the variance and create a visualization to support or refute your suggestion.

```
# Part 2 - Question 4
# Your code goes here:
#
#
```

---

Part 2 Question 4 - textual Answer:

*Write your answer here:*

---

## ✓ Part 3: Neighborhoods

### ✓ Part 3 - Function Definitions and DataFrame Creation

```
# @title Part 3 - Function Definitions and DataFrame Creation
def reverse_string(a):
    return a[::-1]

socialrank_df = load_df(SOCIORANK_ID)
neighborhood_ranks = {k: v for k,v in zip(socialrank_df['neighborhood'], socialrank_df['soci

def get_neighborhood_rank(neighborhood):
    if neighborhood in neighborhood_ranks:
        return neighborhood_ranks[neighborhood]
    else:
        return None

# Create the dataframe and remove the outliers we found in the intro part:
part3_df = rent_df_backup_for_exercise.copy()
part3_df = part3_df[part3_df['monthlyRate'] > 0].reset_index(drop=True);
part3_df = part3_df[part3_df['area'] < 800].reset_index(drop=True)
part3_df = part3_df[part3_df['area'] > 10].reset_index(drop=True)
part3_df["neighborhood_flipped"] = part3_df["neighborhood"].apply(reverse_string) # making t
```

We now want to focus on the differences between different neighborhoods in Jerusalem.

**Use only part3\_df for the coding questions in this part**

\*Use the "neighborhood\_flipped" column for visualizations as seaborn will flip the order of letters in hebrew.

### ✓ Question 1

Print the number of unique neighborhoods that appear in the dataset.

```
# Part 3 - Question 1
# Your code goes here:
#
#
unique_neighborhoods = part3_df['neighborhood_flipped'].nunique()

print(f"Number of unique neighborhoods: {unique_neighborhoods}")
```

➞ Number of unique neighborhoods: 46

## ✓ Question 2

Visualize the number of listings per neighborhood in a way that will allow you to easily identify those with the highest count.

```
# Part 3 - Question 2
# Your code goes here:
#
#
# Count the number of listings per neighborhood
neighborhood_counts = part3_df['neighborhood_flipped'].value_counts().reset_index()
neighborhood_counts.columns = ['neighborhood_flipped', 'count']

# Sort neighborhoods by the number of listings
neighborhood_counts = neighborhood_counts.sort_values(by='count', ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x='count', y='neighborhood_flipped', data=neighborhood_counts, palette='viridis')

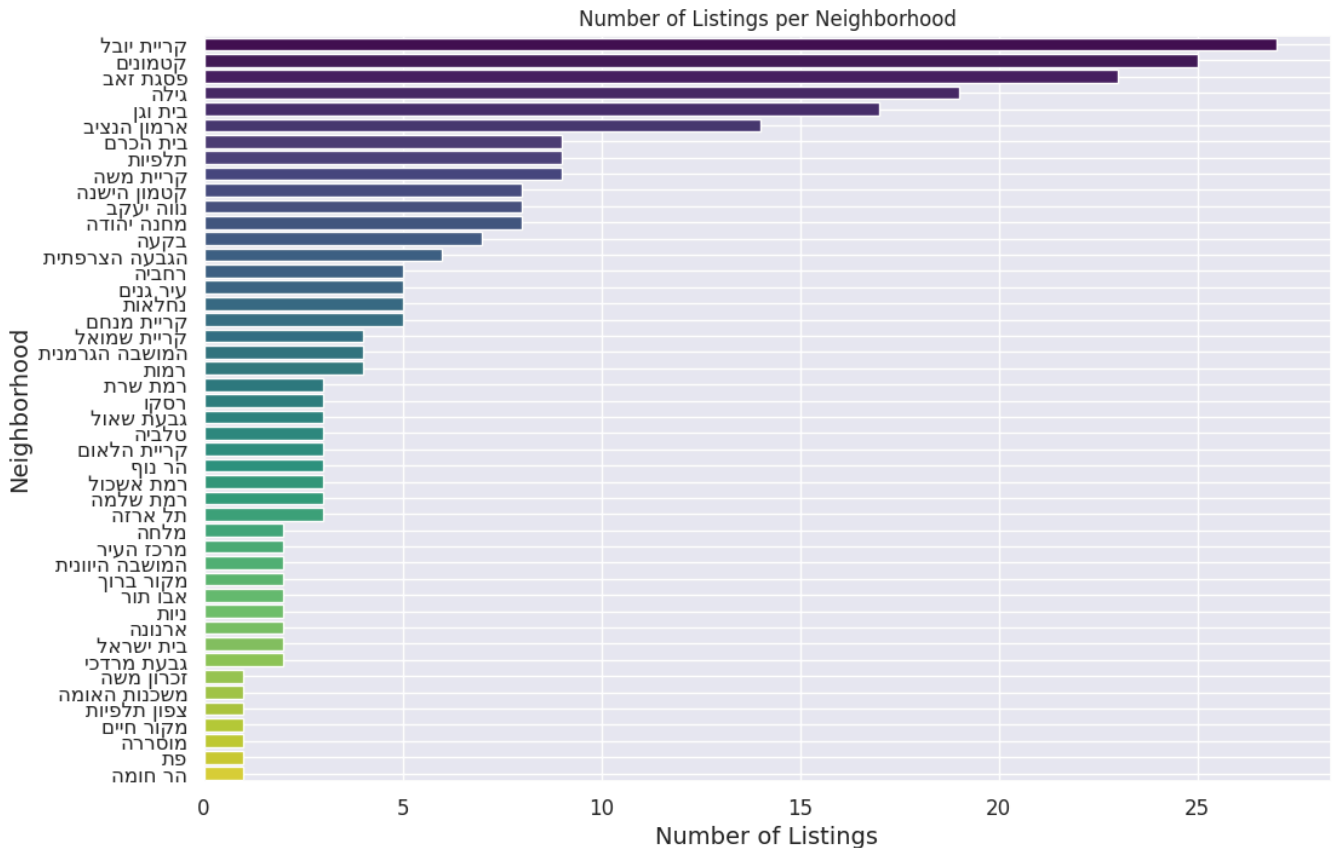
plt.title('Number of Listings per Neighborhood')
plt.xlabel('Number of Listings')
plt.ylabel('Neighborhood')
plt.grid(True)
plt.show()
```



 <ipython-input-17-bc21fb831055>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

```
sns.barplot(x='count', y='neighborhood_flipped', data=neighborhood_counts, palette='vi
```



### ✓ Question 3 - Heavy-tailed distributions

Print the number of neighborhoods with less than 5 listings and the fraction of their total number of listings out of the total number of listings. Also print the fraction of listings from the 8 most frequent neighborhoods out of the total number of listings.

```

# Part 3 - Question 3
# Your code goes here:

group_less5 = 0
listings_less5 = 0

for listing in neighborhood_counts['count']:
    if listing < 5:
        group_less5 += 1
        listings_less5 += listing

print("number of neighborhoods with less than 5 listings:", group_less5)

total_listings = neighborhood_counts['count'].sum()
fraction_less5 = listings_less5 / total_listings

print("fraction of less than 5 listings from total listings:", fraction_less5)

listings_highest8 = neighborhood_counts['count'][:8].sum()
fraction_highest8 = listings_highest8 / total_listings

print("fraction of 8 most frequent neighborhoods listings from total listings:", fraction_hi

```

```

➞ number of neighborhoods with less than 5 listings: 28
fraction of less than 5 listings from total listings: 0.23443223443223443
fraction of 8 most frequent neighborhoods listings from total listings: 0.52380952380952

```

Those types of distributions where there are many categories that appear only a few times but together take a large portion of the distribution are called heavy-tailed (or long-tailed) distributions. This is a real issue in many data science applications, since even if we have a large dataset there are still some sub-populations or sub-categories that are not well represented.

## ✓ Question 4

Create a new filtered dataframe with listings from only the 8 most frequent neighborhoods.

```
# Part 3 - Question 4
# Your code goes here:
#
neighborhood_counts = part3_df['neighborhood_flipped'].value_counts().reset_index()
neighborhood_counts.columns = ['neighborhood_flipped', 'count']

df_highest8 = neighborhood_counts.head(8)['neighborhood_flipped']

listings_highest8 = part3_df[part3_df['neighborhood_flipped'].isin(df_highest8)]
```

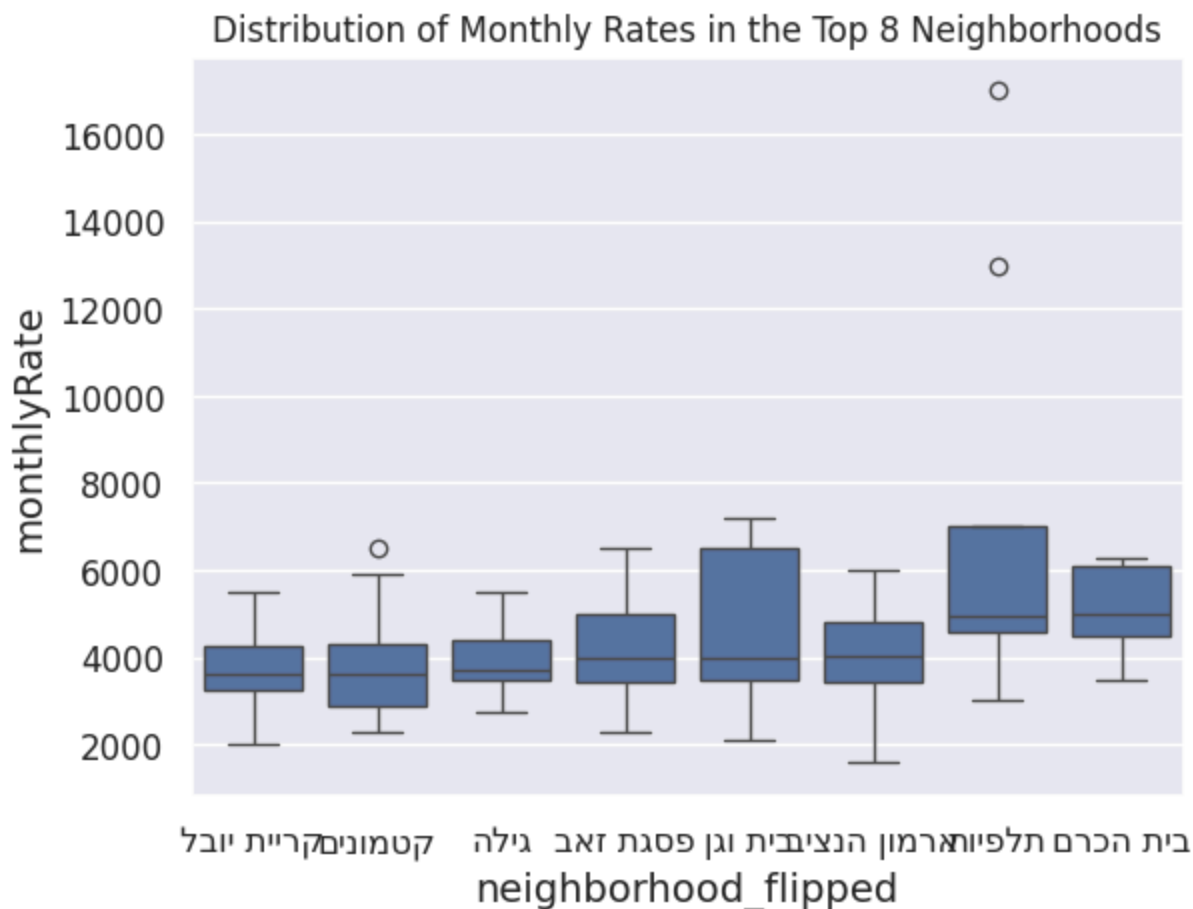
## ✓ Question 5

Plot a graph to check whether there are different distributions of monthly rates in the eight neighborhoods. Explain your choice for the visualization and your conclusions. Note: Make sure that the neighborhoods are ordered in the plot based on their tendency for higher or lower monthly rates.

Hint: Which is a better descriptor of the central tendency of monthly rates when the distributions are skewed?

```
# Part 3 - Question 5
# Your code goes here:
#
ordered_neighborhoods = listings_highest8.groupby('neighborhood_flipped')['monthlyRate'].median()
sns.boxplot(x='neighborhood_flipped', y='monthlyRate', data=listings_highest8, order=ordered_neighborhoods)
plt.title('Distribution of Monthly Rates in the Top 8 Neighborhoods')
```

```
Text(0.5, 1.0, 'Distribution of Monthly Rates in the Top 8 Neighborhoods')
```



### Part 3 Question 5 - textual Answer:

Write your answer here: כשהפיזור מוטה עדיף להתחשב לפי חציון ולא ממוצע.

בחרנו בבוקס פלוט מכיוון ואפשר לבחור לפי מה יהיה האמצע - במקרה זה בחרנו את החציון. בנוסף אפשר לראות את הפיזור כולל האאוטליירים וכך קל להשוות בין הקבוצות השונות.

### ✓ Question 6

Now that we compared the different distributions of monthly rates between neighborhoods, we can check whether we can explain some of the differences using our common-sense and the data we already have. For example, perhaps different neighborhoods have different distributions of apartment sizes?

Think of a new variable that will allow you to check the relationship between neighborhoods and prices fairly, factoring different apartment sizes out of the equation. Save this measure into the dataframe and create a new visualization to answer the question.

```
# Part 3 - Question 6
# Your code goes here:
#
#
```

```
listings_highest8['price_per_room'] = listings_highest8['monthlyRate'] / listings_highest8['
ordered_by_rooms']
ordered_by_rooms = listings_highest8.groupby('neighborhood_flipped')['price_per_room'].median
sns.boxplot(x='neighborhood_flipped', y='price_per_room', data=listings_highest8, order=order)
plt.title('Distribution of price per room in the Top 8 Neighborhoods')
```



<ipython-input-21-c6b3f6c0f751>:6: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)  
listings\_highest8['price\_per\_room'] = listings\_highest8['monthlyRate'] / listings\_highest8['ordered\_by\_rooms']  
Text(0.5, 1.0, 'Distribution of price per room in the Top 8 Neighborhoods')



### ✓ Part 3 Question 6 - textual Answer:

*Write your answer here:*

---

Given the conclusions from the previous steps, we may think that the apartment's neighborhood gives us additional information about the expected monthly rate. But the sample size for most neighborhoods is rather small. So let's examine another way to utilize the location information. Luckily, we also have data about the socio-economic rank of most neighborhoods (between 1 and 10).

### ✓ Question 7 - **bonus**

Use again the full dataset (without filtering by neighborhood).

Create an aggregated dataframe where every record represents a neighborhood, with columns for:

1. neighborhood name
2. flipped neighborhood name
3. The number of listings in a neighborhood
4. The median monthly rate for listings in this neighborhood.

Add a column with the neighborhood socio-economic rank to the dataframe (you can use the provided `get_neighborhood_rank` function that takes as an input a neighborhood name and returns its socio-economic rank.) Use this dataframe to visualize the association between socio-economic rank and pricing for all neighborhoods with at least 5 listings. What is your conclusion?

```
# Part 3 - Question 7
# Your code goes here:
#
#
```

---

### Part 3 Question 7 - textual Answer:

*Write your answer here:*

---

### ✓ Part 4: Are private houses more expensive than apartments?

## ✓ Part 4 - Create a DataFrame and remove outliers for Part 4

```
# @title Part 4 - Create a DataFrame and remove outliers for Part 4
part4_df = rent_df_backup_for_exercise.copy()
part4_df = part4_df[part4_df['monthlyRate'] > 0].reset_index(drop=True);
part4_df = part4_df[part4_df['area'] < 800].reset_index(drop=True)
part4_df = part4_df[part4_df['area'] > 10].reset_index(drop=True)
display(part4_df)
```



	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	des
0	3994505	קריית יובל	2000.0	private	1.0	2.0	13.0	10/08/2022	חוב , ה
1	3981298	רחביה	2450.0	private	1.0	1.0	17.0	10/08/2022	1 ה
2	3981623	מלחה	2550.0	private	1.0	0.0	30.0	10/08/2022	ה
3	3993997	בית וגן	2100.0	private	1.0	0.0	15.0	10/08/2022	זי



Finally, we want to check if listings in private houses tend to be more expensive than apartments in a building.

**Use only `part4_df` for the coding questions in this part**

## ✓ Question 1

The current dataset doesn't include a variable that describes whether a listing is in a building or a private house but this can be inferred from the existing variables. Create a new column named "is\_a\_house" with value of `True` if a listing is in the first (or zero) floor in a building with only one floor. Print the number of private houses and print the descriptions of three random listings with 'is\_a\_house' equal to `True`.

```
# Part 4 - Question 1
# Your code goes here:
#
#
```

```
part4_df['is_a_house'] = (part4_df['floor']<=1) & (part4_df['numFloors']==1)
numTrue = (part4_df['is_a_house'] == True).sum()
print("number of private houses:", numTrue)
```

```
random_private = part4_df[part4_df['is_a_house']==True].sample(3)
print("3 random private houses:\n", random_private)
display(part4_df)
```



number of private houses: 17

3 random private houses:

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	\
48	3985356	3600.0	גילה	private	2.0	1.0	60.0	