Group Details:

- Name: Carmel Ben Yosef ID: 315116368
- Name: Atalya Uzan ID: 318783420
- Name: Adi Zemel ID: 211999735
- Name: Shahed Higaze ID: 213766561

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 **breifly** 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',
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
403	3981729	גבעת מרדכי	4500.0	private	3.0	6.0	62.0	10/08/2022
457	3991612	קריית משה	3000.0	private	3.5	3.0	NaN	NaN

And print some summary statistics:

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

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	
count	6.120000e+02	612	612.000000	612	612.000000	611.000000	29
unique	NaN	54	NaN	2	NaN	NaN	
top	NaN	קריית יובל	NaN	private	NaN	NaN	
freq	NaN	66	NaN	600	NaN	NaN	
mean	3.981582e+06	NaN	4717.393791	NaN	2.927288	1.916530	8
std	6.525543e+04	NaN	2195.215139	NaN	1.007350	1.581006	27
min	2.494041e+06	NaN	0.000000	NaN	1.000000	-2.000000	
25%	3.981694e+06	NaN	3500.000000	NaN	2.000000	1.000000	4
50%	3.987901e+06	NaN	4400.000000	NaN	3.000000	2.000000	6
75%	3.992605e+06	NaN	5800.000000	NaN	3.500000	3.000000	8

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

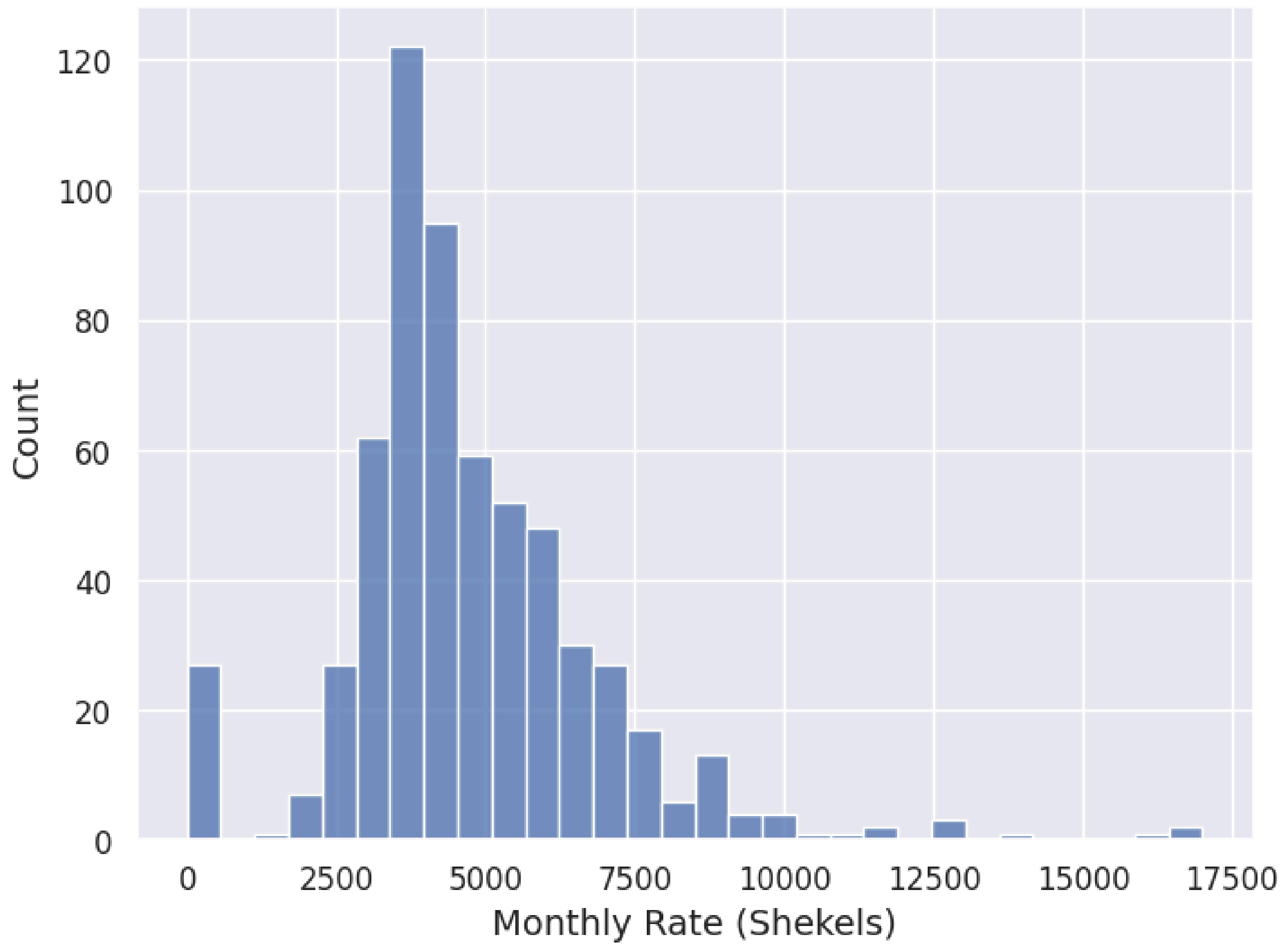
What is the distribution of prices in this dataset?

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

Solution 1

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

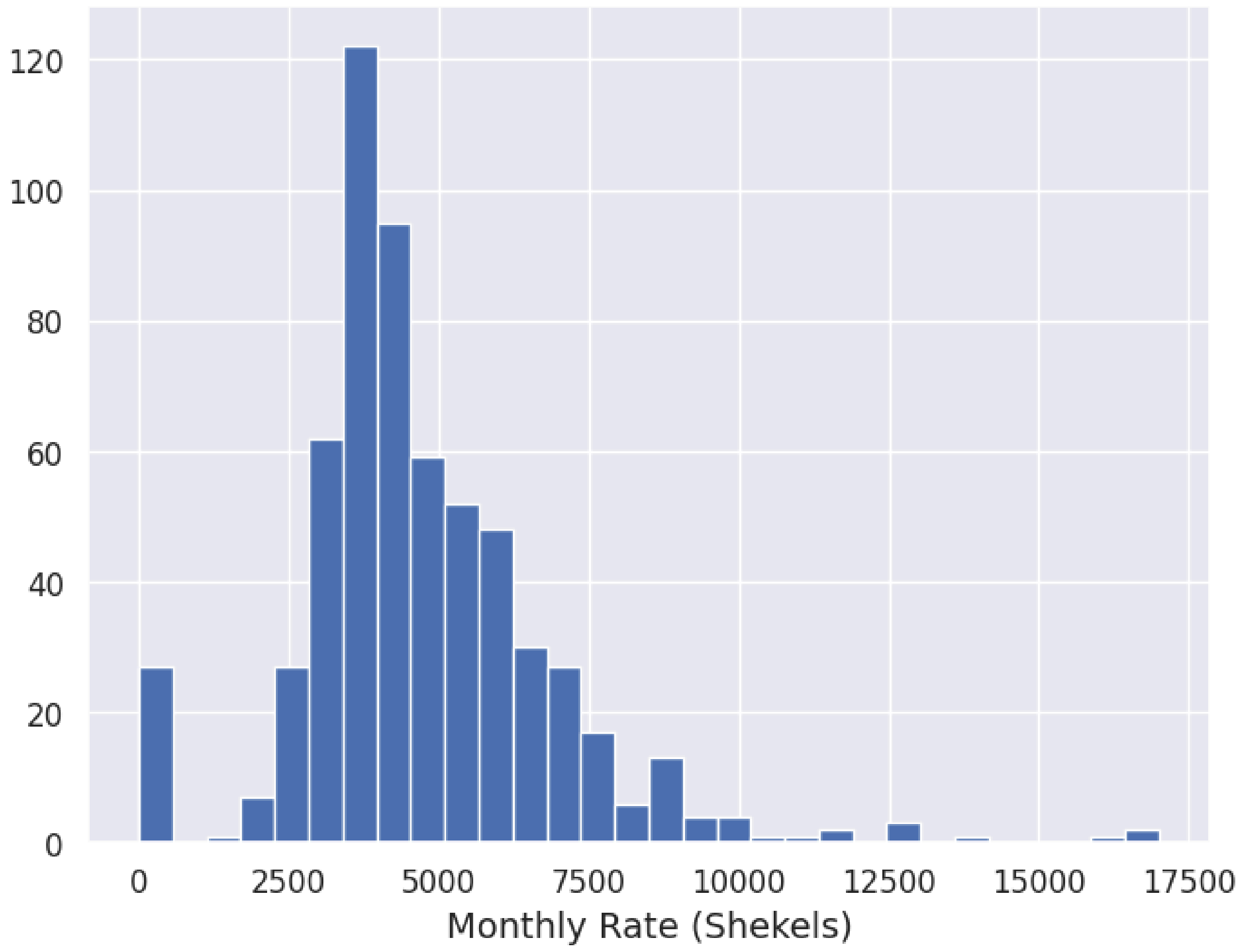
```
Text(0.5, 0, 'Monthly Rate (Shekels)')
```



Solution 2

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

```
Text(0.5, 0, '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].
```

```
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 c

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()
```

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	d
20	3983978	קריית משה	0.0	private	4.0	3.0	100.0	10/08/2022	
21	3985184	נווה יעקב	0.0	private	4.0	1.0	68.0	10/08/2022	

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

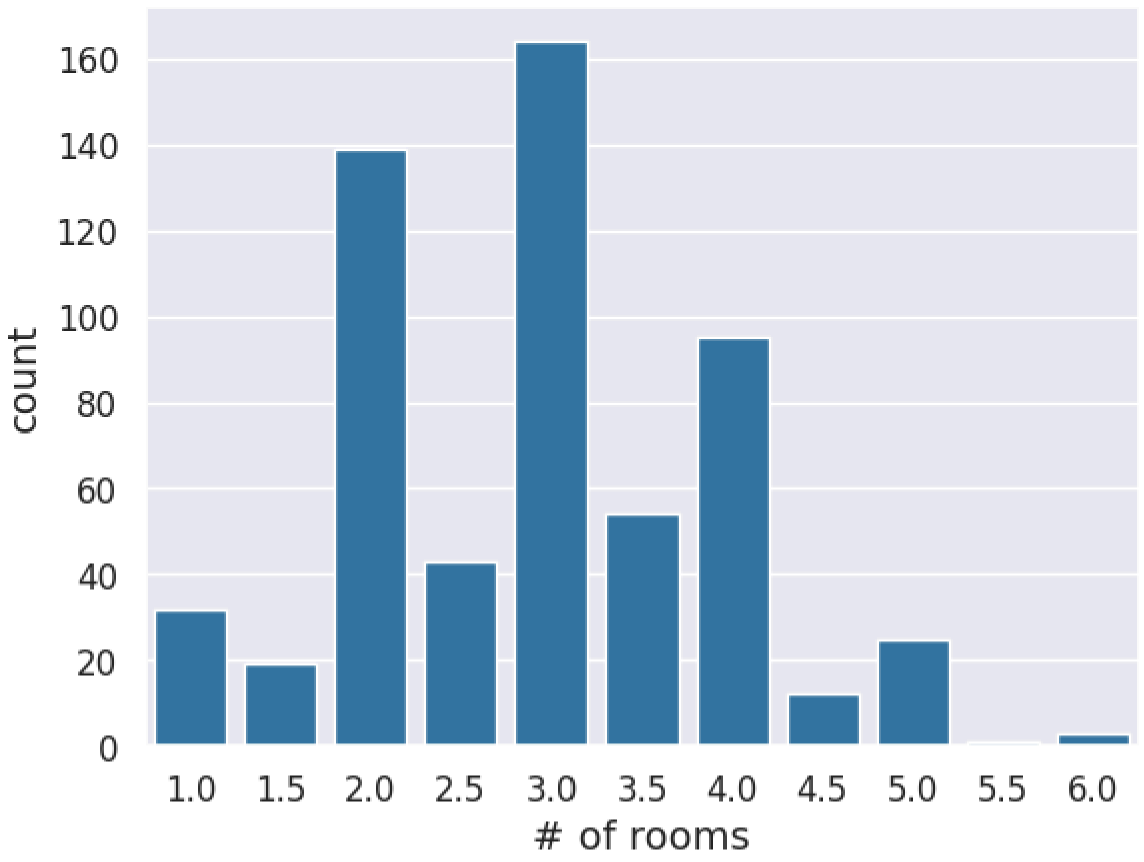
```
clean_df = rent_df[rent_df['monthlyRate'] > 0].reset_index(drop=True)
```

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.

Solution

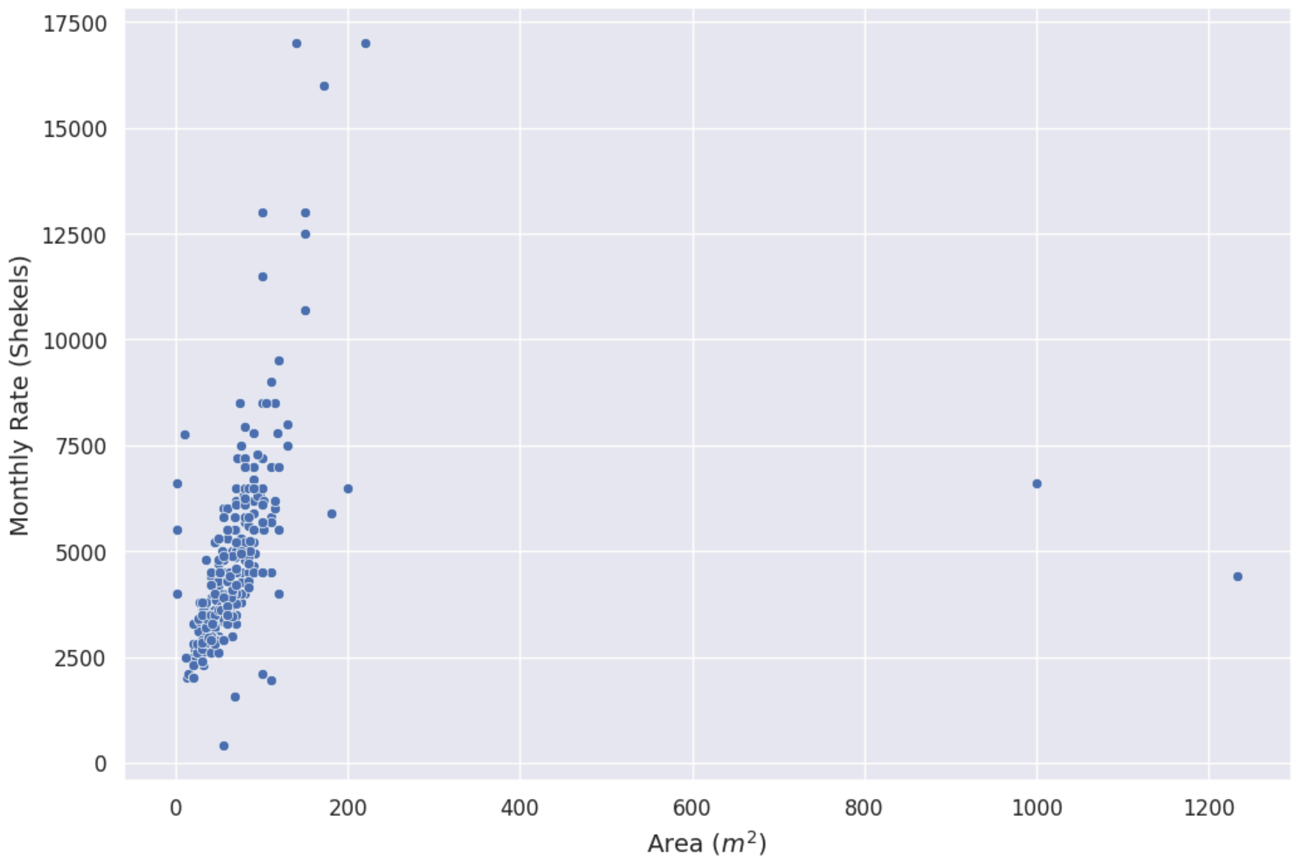
```
# @title Solution
if clean_df is None:
  print("Can't run until 'clean_df' is created!")
else:
  sns.countplot(x='rooms', data=clean_df, color='tab:blue')
  plt.xlabel("# of rooms")
```



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 ~$1000m^2$.

Let's look at those samples to see if we can understand what happend 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
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 כניס
ה עם פינת ישיבה. מתאימה מאוד ל- 3 שותפים'
```

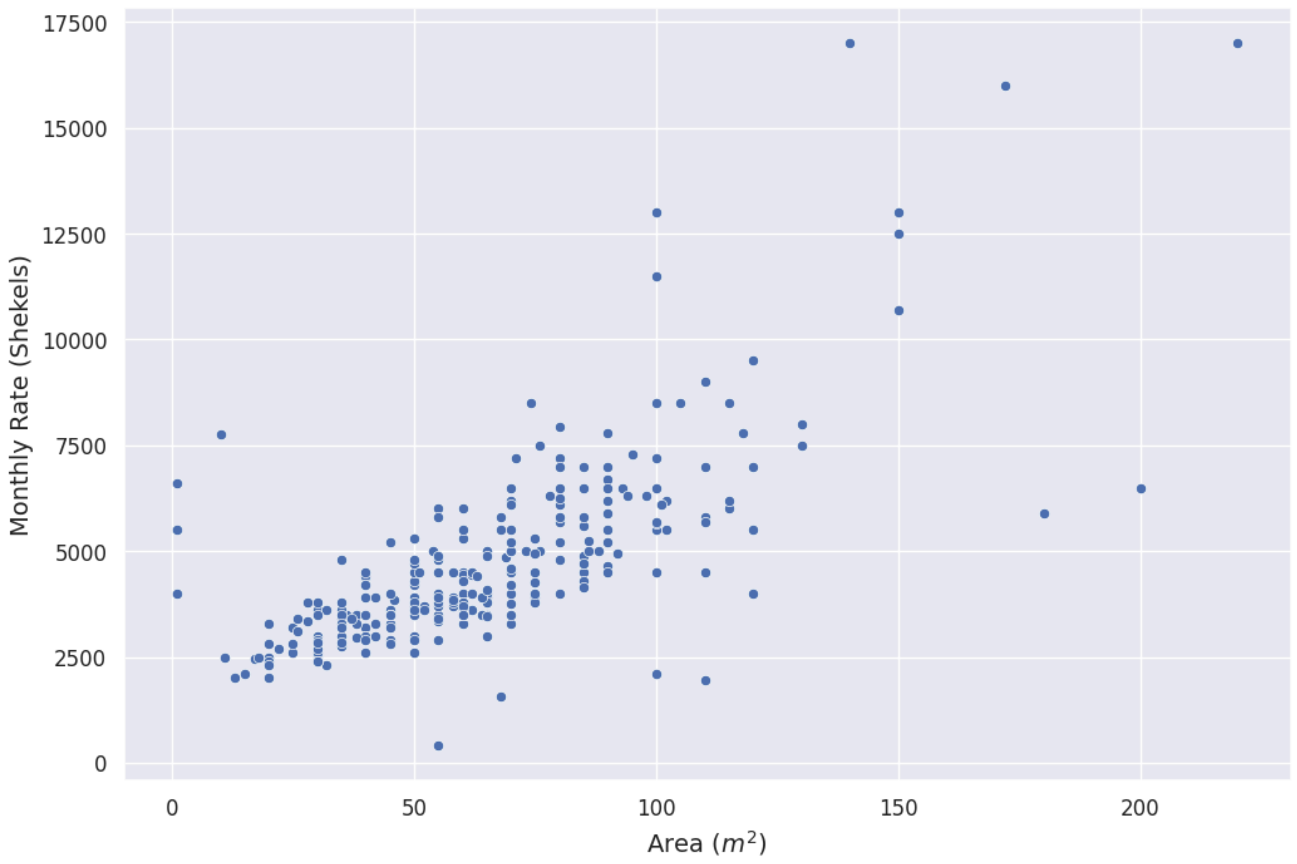
Clearly not a 1000 m^2 apartment...

Q: Save a new dataframe named `clean_df_area_filtered` with all listings with area smaller than 800 m^2. 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_duplicat

  # 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
0	3994505	קריית יובל	2000.0	private	1.0	2.0	13.0	10/08/2022
1	3981298	רחביה	2450.0	private	1.0	1.0	17.0	10/08/2022
3	3993997	בית וגן	2100.0	private	1.0	0.0	15.0	10/08/2022
5	3993552	הר נוף	2000.0	private	1.0	0.0	20.0	10/08/2022
6	3972039	גבעת שאול	2700.0	private	1.0	0.0	22.0	10/08/2022
7	3988096	המושבה הגרמנית	2500.0	private	1.0	0.0	18.0	10/08/2022
8	3992809	נחלאות	3200.0	private	1.0	2.0	25.0	10/08/2022
10	3983516	הגבעה הצרפתית	2000.0	private	1.0	2.0	20.0	10/08/2022

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
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_
```

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

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

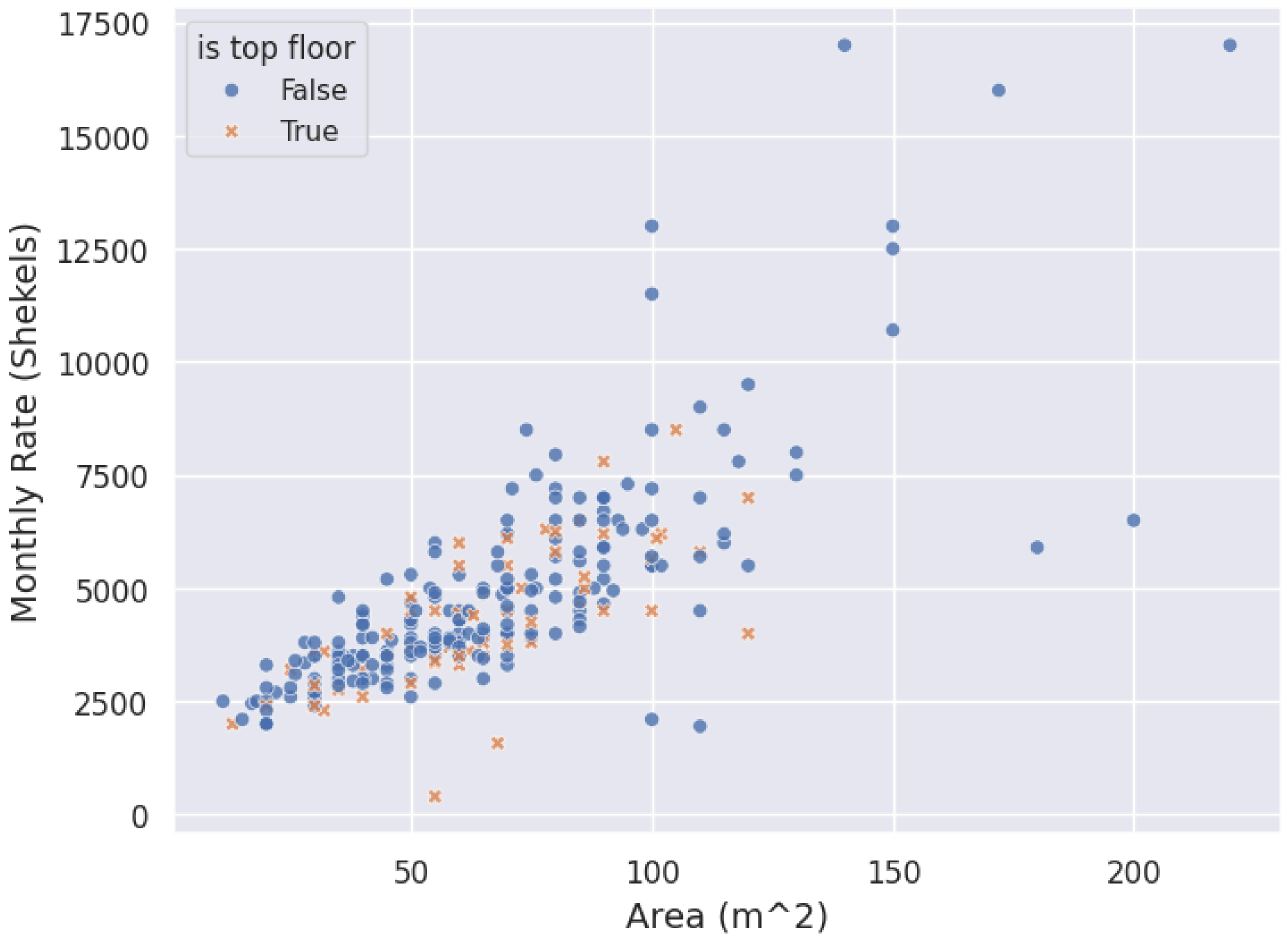
```
#remove the outliers
if clean_df_area_filtered is None:
  print("Can't run until 'clean_df_area_filtered' is created!")
elif outlier_df is None:
  print("Can't run until 'outlier_df' is created!")
else:
  non_ouliers = clean_df_area_filtered['area'] > 10 # get non outliers series of true/fal

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

  # remove them
  clean_df_area_filtered = clean_df_area_filtered[non_ouliers].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_ar
  plt.figure(figsize=(8,6))
  sns.scatterplot(x='area', y='monthlyRate', data=clean_df_area_filtered, alpha=0.8, hue=
  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
198	3956418	רחביה	13000.0	agent	4.0	1.0	100.0	NaN
236	3985051	טלביה	17000.0	private	4.0	4.0	140.0	10/08/2022

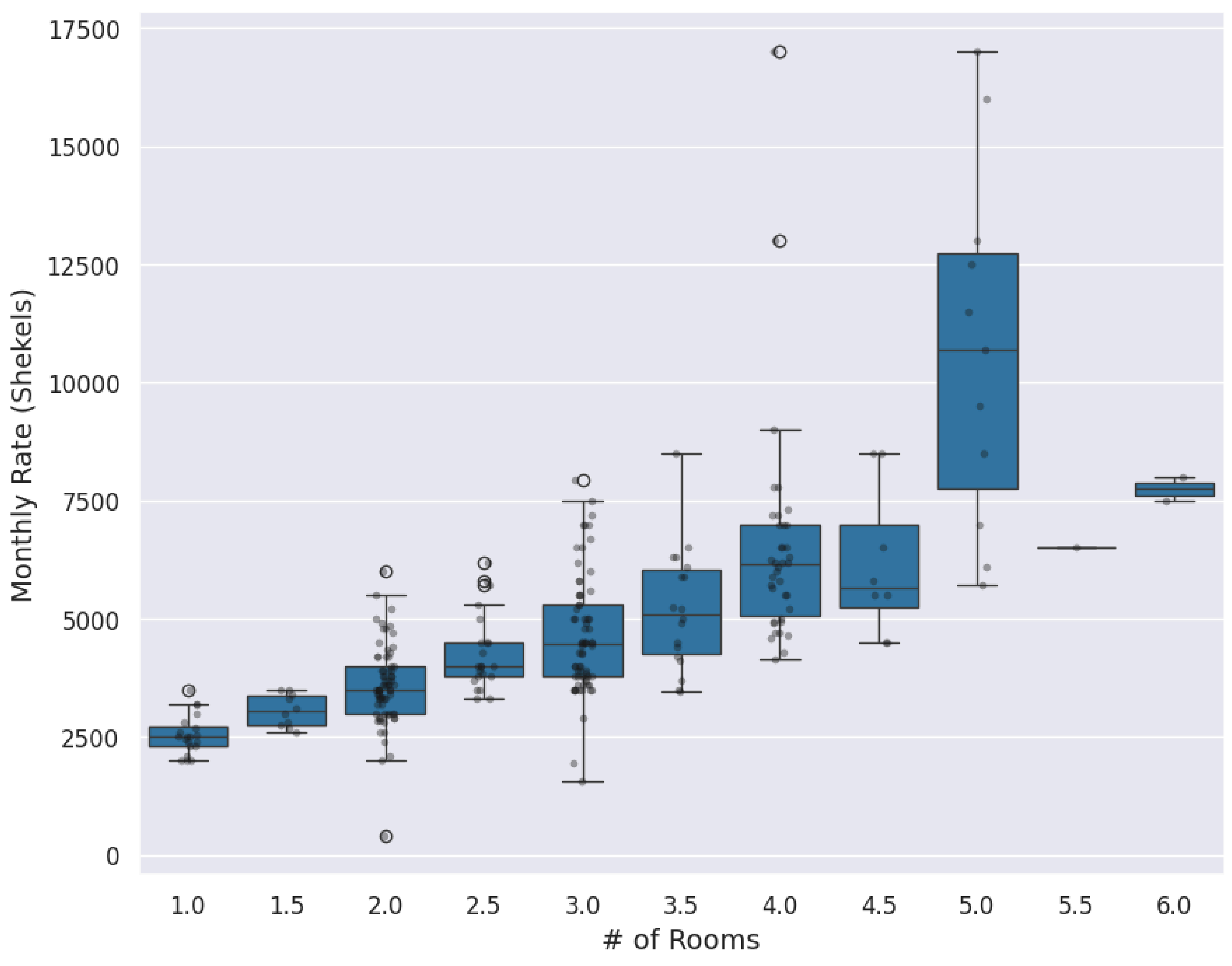
We can see some representation of the more expensive neighborhoods of Jerusalem here.. More on the neighborhoods later on!

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_are
  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'
  # # 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_fil
  # plt.xlabel("# of Rooms")
  # plt.ylabel("Monthly Rate (Shekels)");

  #Violin plot completly fails for very small subsets:
  # plt.figure(figsize=(10,8))
  # sns.violinplot(x='rooms', y='monthlyRate', data=clean_df_area_filtered, color='tab:bl
  # 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_f
```

	monthlyRate								rooms	
	count	mean	std	min	25%	50%	75%	max	count	m
reason										
'area' <= 10	5.0	5870.0	1399.821417	4000.0	5500.0	5500.0	6600.0	7750.0	5.0	[illegible]
'area' >= 800	2.0	5500.0	1555.634919	4400.0	4950.0	5500.0	6050.0	6600.0	2.0	[illegible]
monthlyRate <= 0	25.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	25.0	[illegible]

3 rows × 40 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?

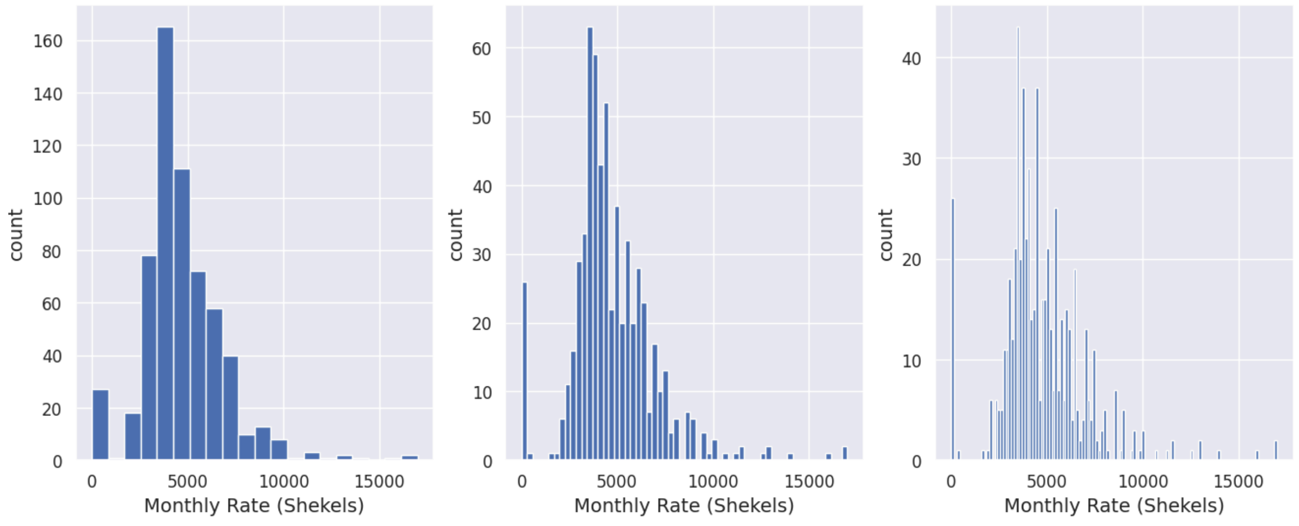
Use only `part1_df` for the coding questions in this part

Question 1

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

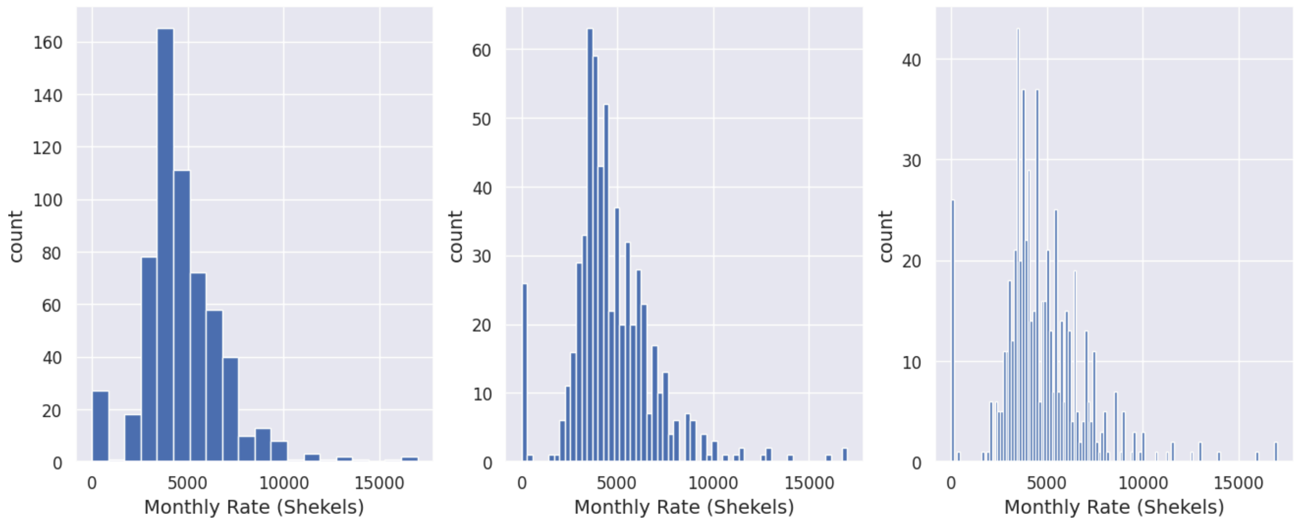
```
fig, axes = plt.subplots(1, 3, figsize=(16, 6))

axes[0].hist(part1_df["monthlyRate"], bins=20)
axes[0].set_xlabel("Monthly Rate (Shekels)")
axes[0].set_ylabel("count")

axes[1].hist(part1_df["monthlyRate"], bins=60)
axes[1].set_xlabel("Monthly Rate (Shekels)")
axes[1].set_ylabel("count")

axes[2].hist(part1_df["monthlyRate"], bins=120)
axes[2].set_xlabel("Monthly Rate (Shekels)")
axes[2].set_ylabel("count")
```

```
Text(0, 0.5, 'count')
```



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

fig, axes = plt.subplots(1, 3, figsize=(16, 6))

axes[0].hist(part1_df["monthlyRate"], bins=20)
axes[0].set_xlabel("Monthly Rate (Shekels)")
axes[0].set_ylabel("count")

axes[1].hist(part1_df["monthlyRate"], bins=60)
axes[1].set_xlabel("Monthly Rate (Shekels)")
axes[1].set_ylabel("count")

axes[2].hist(part1_df["monthlyRate"], bins=120)
axes[2].set_xlabel("Monthly Rate (Shekels)")
axes[2].set_ylabel("count")
```

```
Text(0, 0.5, 'count')
```



Question 2

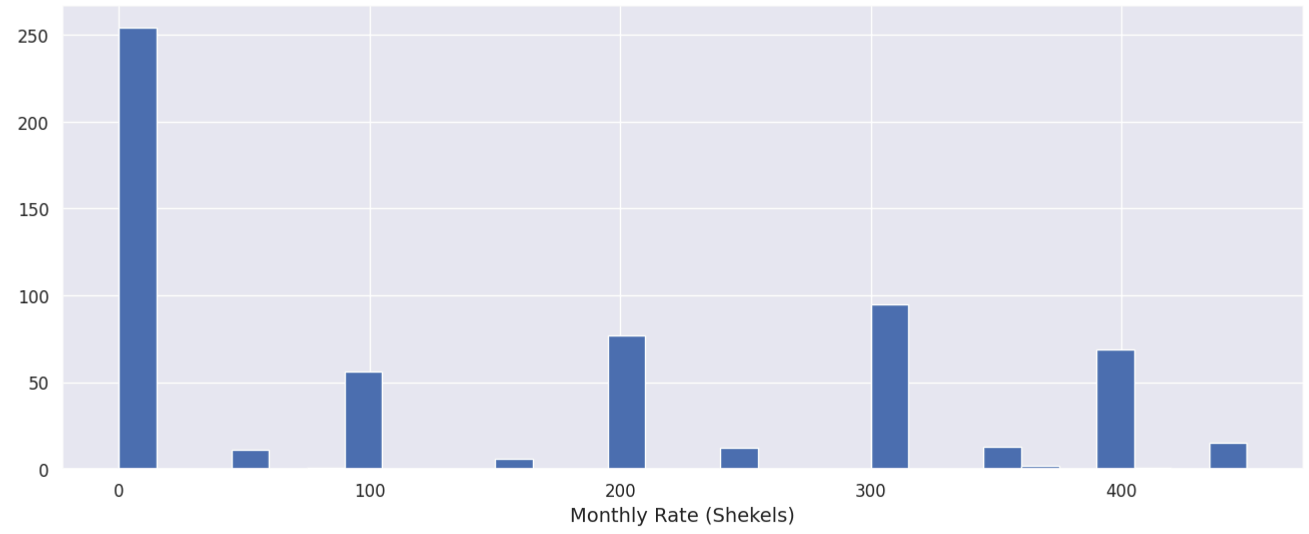
For 60 and 120 bins, you can see a repeating pattern of "peaks" and "vallies" in the distribution (mostly in the range between 500 and 7000). Is this pattern due to people rounding the rental prices? Please create a visualization that answers this question. Describe in words how the graph shows what 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["monthlyRate"] % 500).hist(bins=30 ,figsize=(16, 6))
plt.xlabel("Monthly Rate (Shekels)")
```

```
Text(0.5, 0, 'Monthly Rate (Shekels)')
```



Part 1 Question 2 - textual Answer:

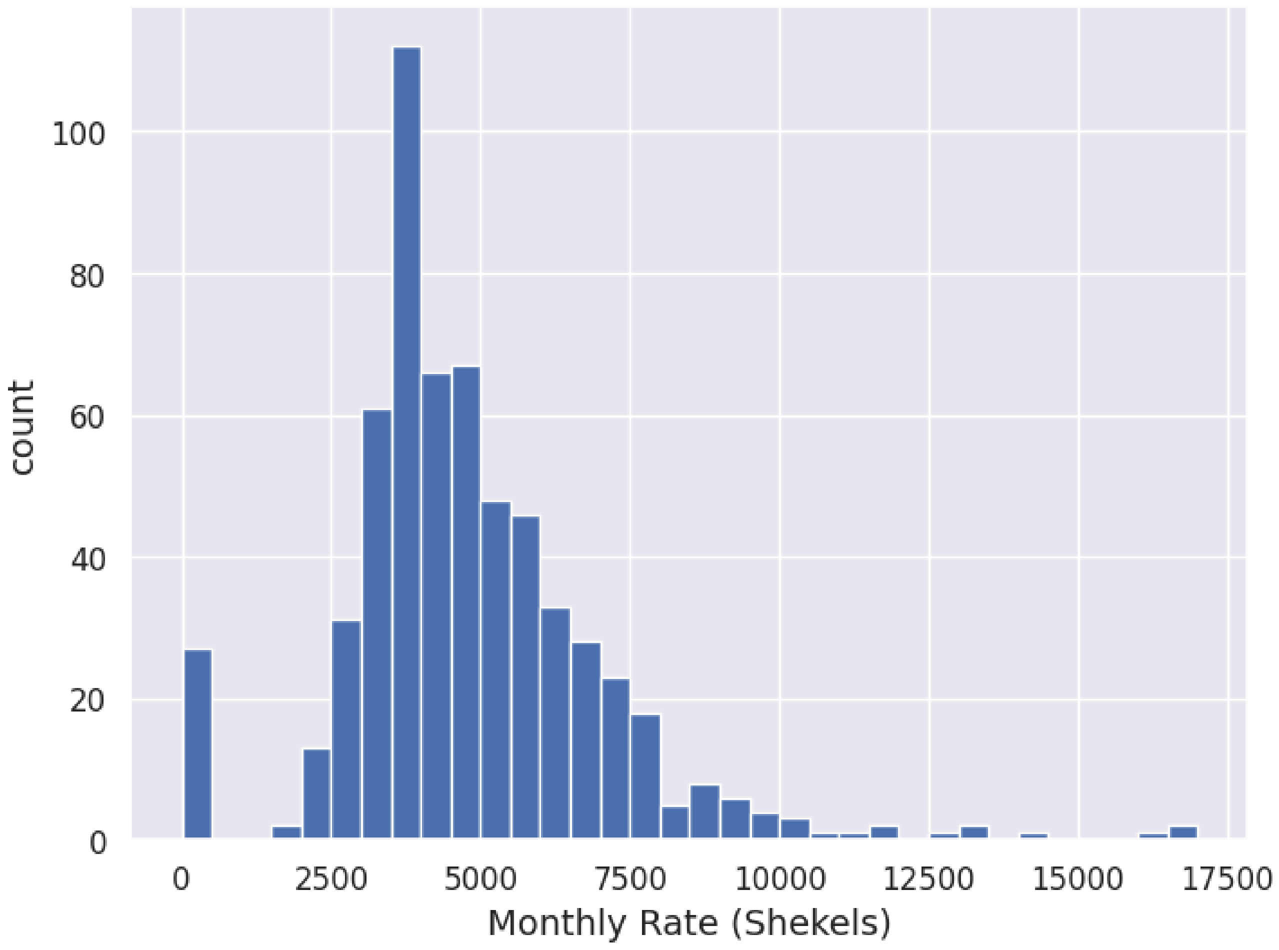
Write your answer here: אכן יש נטייה לעגל את המחירים. כאשר אנו מחלקים את הדירות ב-100, ניתן לראות כי הרוב המוחלט של הדירות מתחלקות ב-100 ללא שארית, והשאר מתחלקות ב-50. כלומר רוב האנשים מעגלים את מחיר הדירות למאות, ושאר האנשים מעגלים ב-50. אין בדאטה הנוכחי אנשים שלא מעגלים את הדירות במספר נמוך מ-50.

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:
#
#
(part1_df["monthlyRate"]).hist(bins= np.arange(0, max(part1_df["monthlyRate"]) + 500, 50
plt.xlabel("Monthly Rate (Shekels)")
plt.ylabel("count")
```

```
Text(0, 0.5, 'count')
```



Part 1 Question 3 - textual Answer:

Write your answer here: אנחנו אכן רואים ירידה מערך 5000 לערך 55000, כך שניתן להסיק כי יש ירידה כלשהי לטובת הנחת מס. גודל הבין מבטיח שלא נראה בעיות בעיגול הדירות כי הראנו בסעיף הקודם כי רוב הדירות מעוגלות ל-500. ולכן נצפה לראות ירידה בין 5000 ל-5500 על מנת לקבל את טובת המס.

ניתן לראות ירידה נוספת סביב 4000 ל-4500, וכן סביב 6000, 7500 ו-500.

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.

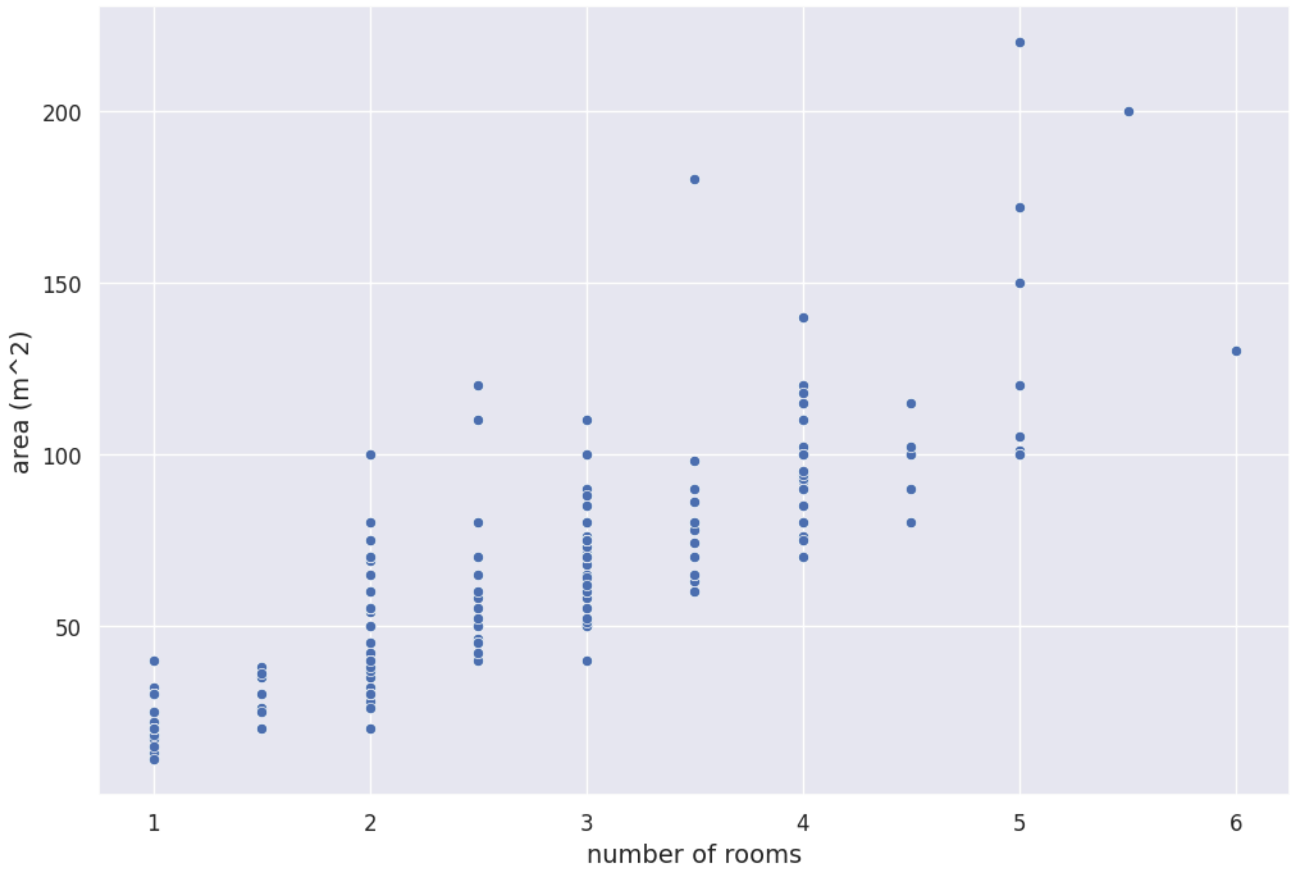
Use only `part2_df` for the coding questions in this part

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:
plt.figure(figsize=(12,8))
sns.scatterplot(x='rooms', y='area', data=part2_df)
plt.ylabel("area (m^2)")
plt.xlabel("number of rooms")
```

```
Text(0.5, 0, 'number of rooms')
```



Part 2 Question 1 - textual Answer:

Write your answer here: We selected a scatter plot to illustrate the relationship between the "number of rooms" and "area (m^2)" because the number of rooms is a continuous variable. The scatter plot shows that as the number of rooms increases, the area of the house also becomes larger.

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"]
display(part2_df)
```

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

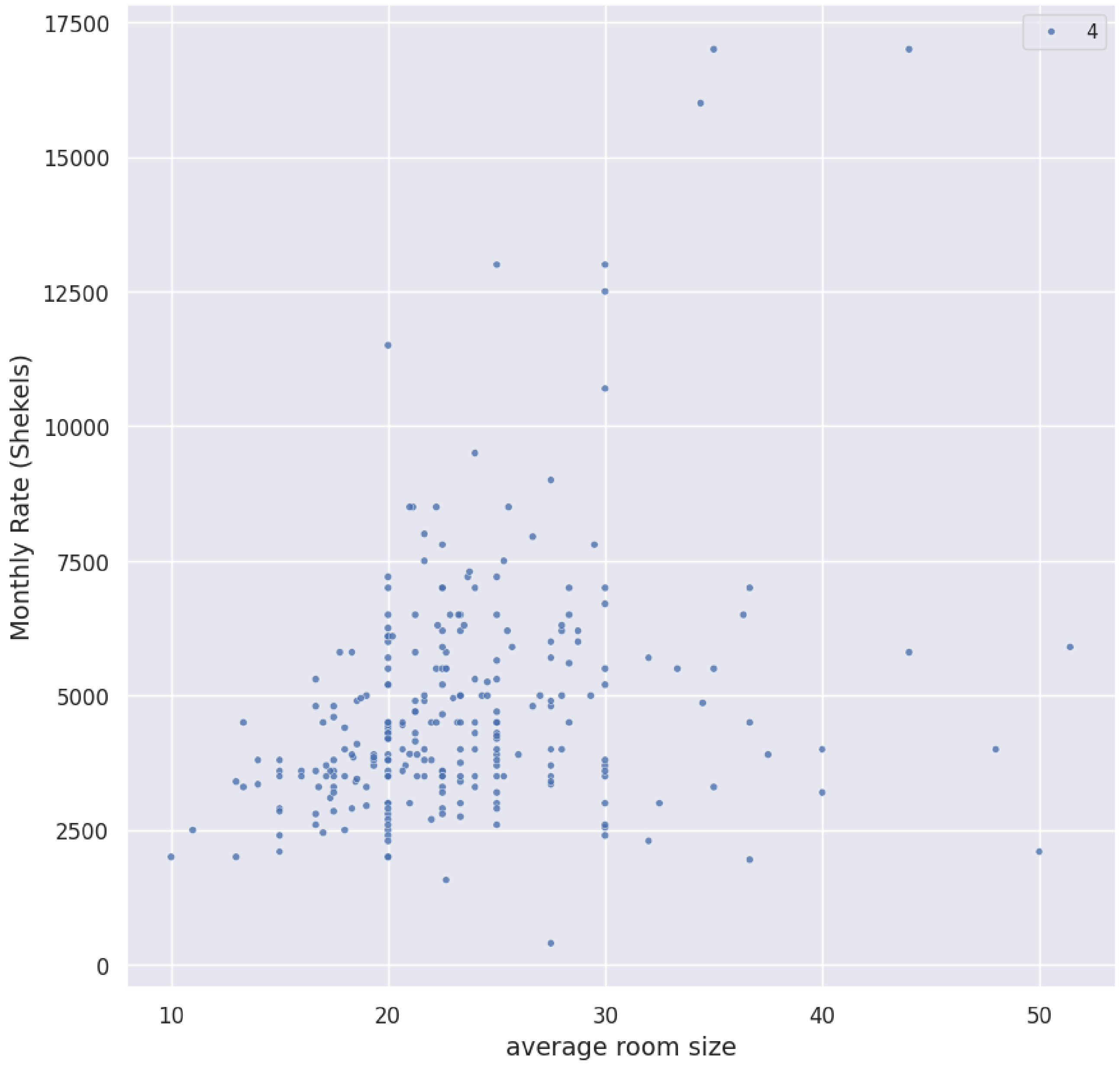
Next steps: View recommended plots

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=(10,10))
sns.scatterplot(x='averageRoomSize', y='monthlyRate', alpha=0.8 ,size=4,color='b',data=pa
plt.xlabel("average room size")
plt.ylabel("Monthly Rate (Shekels)")
```

```
Text(0, 0.5, 'Monthly Rate (Shekels)')
```



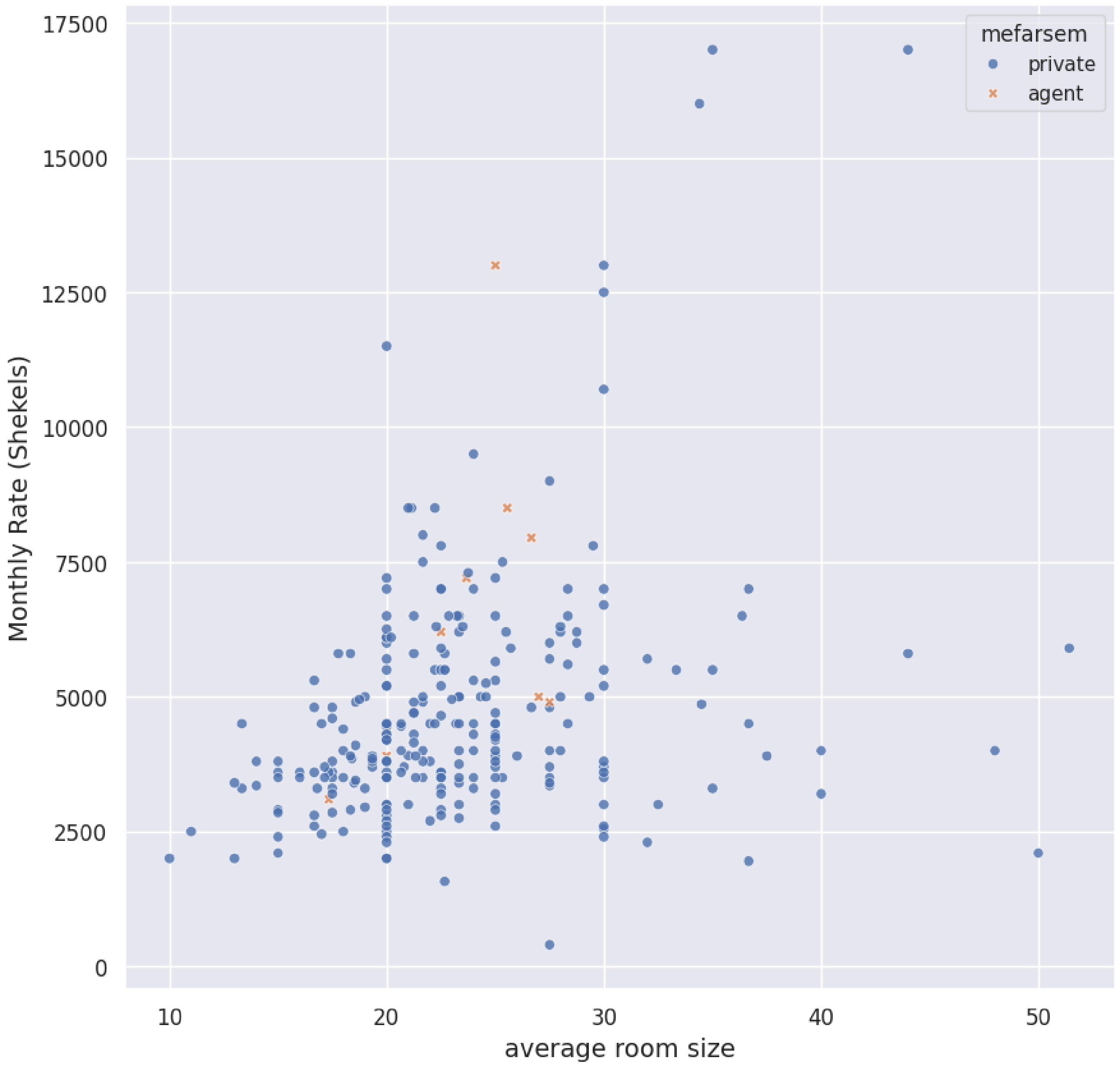
Question 4 - **bonus**

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:
plt.figure(figsize=(10,10))
sns.scatterplot(x='averageRoomSize', y='monthlyRate', alpha=0.8 ,color='b',data=part2_df,
plt.xlabel("average room size")
plt.ylabel("Monthly Rate (Shekels)")
```

```
Text(0, 0.5, 'Monthly Rate (Shekels)')
```



Part 2 Question 4 - textual Answer:

Write your answer here:

I decided to investigate whether the "mefarsem" variable could explain the increase in the "monthly rate" as the average number of rooms rises. However, it doesn't appear to be the case. :(

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['s

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) # makin
```

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:
#
#
print("neighborhoods:")
print(len(part3_df["neighborhood"].unique()))
```

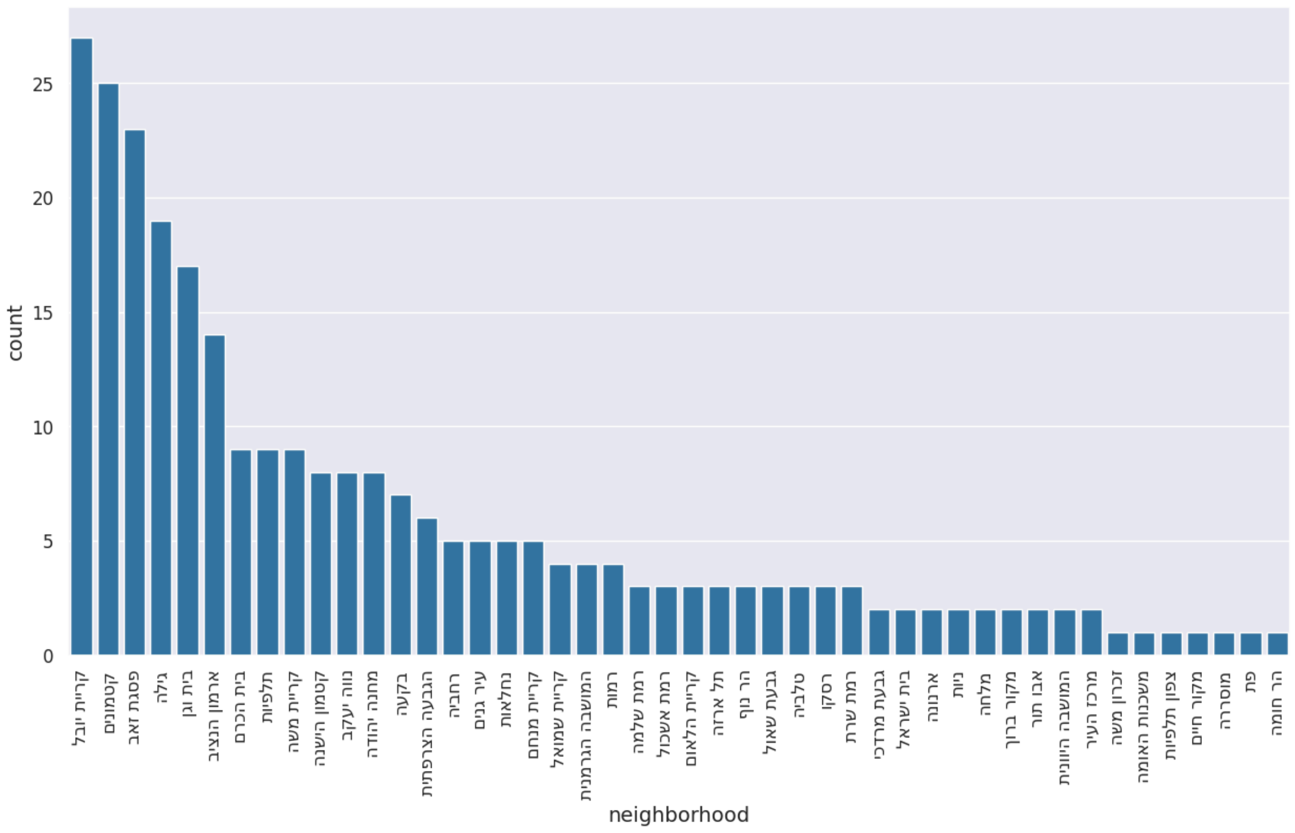
```
neighborhoods:
46
```

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:
#
#
top_neighborhood = list(part3_df['neighborhood_flipped'].value_counts(sort=True).keys())
plt.figure(figsize=(15,8))
sns.countplot(x='neighborhood_flipped',order=top_neighborhood, data=part3_df, color='tab:
plt.xticks(rotation=90)
plt.xlabel("neighborhood")
```

```
Text(0.5, 0, 'neighborhood')
```



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:
#
#
neighborhood_counts = part3_df['neighborhood_flipped'].value_counts()
neighborhoods_with_few_apartments = neighborhood_counts[neighborhood_counts < 5].index
few_listings = neighborhood_counts[neighborhood_counts < 5].sum()
print("number of neighborhoods with less than 5 listings:", len(neighborhoods_with_few_ap
print("their fraction:", few_listings/len(part3_df))
print("the fraction of listings from the 8 most frequent neighborhoods:")
top_neighborhood = part3_df['neighborhood_flipped'].value_counts(sort=True).keys()[:8]
print(len(part3_df[part3_df['neighborhood_flipped'].isin(top_neighborhood)])/len(part3_df
```

```
number of neighborhoods with less than 5 listings: 28
their fraction: 0.23443223443223443
the fraction of listings from the 8 most frequent neighborhoods:
0.5238095238095238
```

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:
#
#
top_neighborhood = part3_df['neighborhood_flipped'].value_counts(sort=True).keys()[:8]
new_df = part3_df[part3_df['neighborhood_flipped'].isin(top_neighborhood)]
```

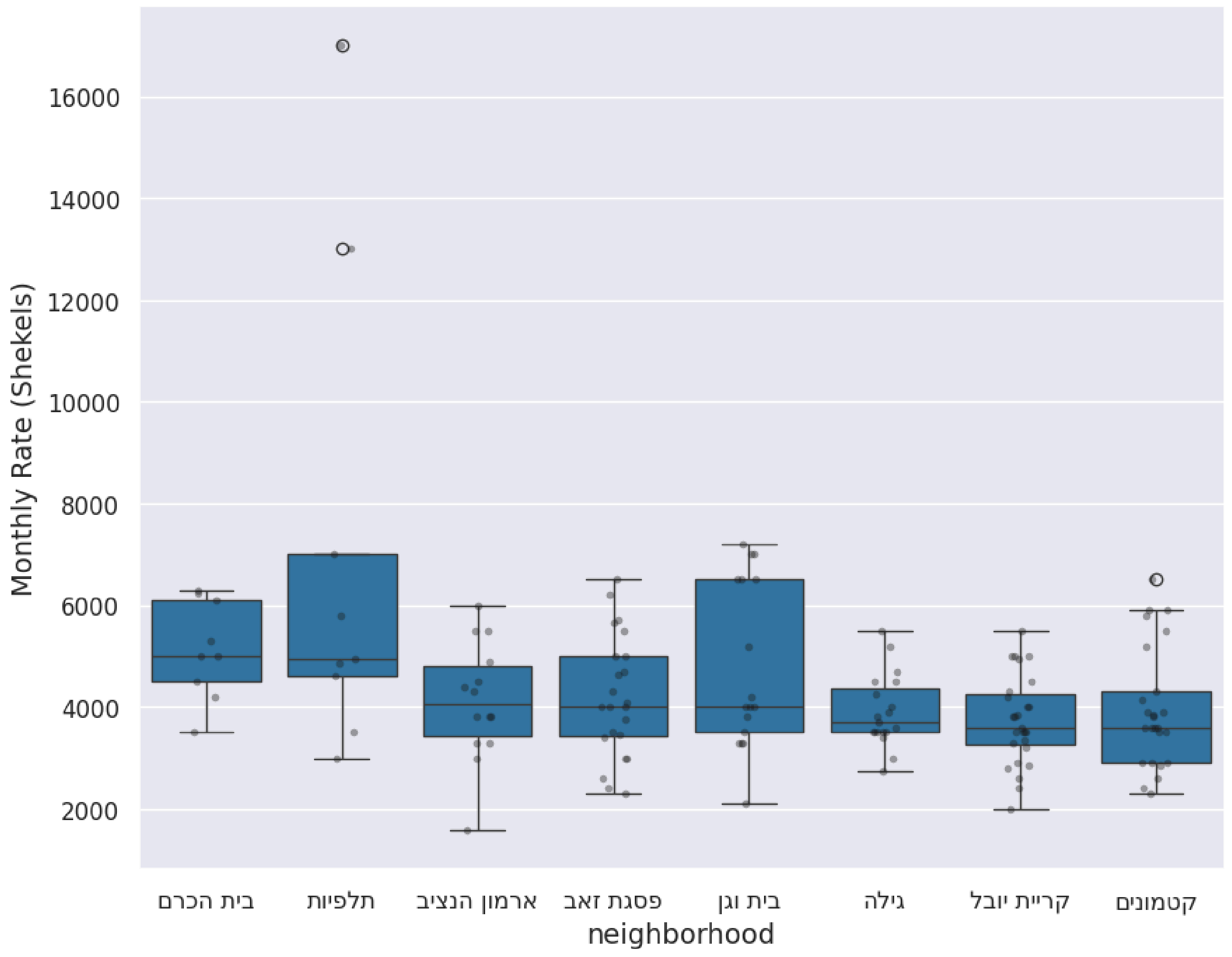
Question 5

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:
#
#
median_order = list(new_df.groupby('neighborhood_flipped')['monthlyRate'].median().sort_v
plt.figure(figsize=(10,8))
sns.boxplot(x= "neighborhood_flipped", y='monthlyRate', data=new_df, color='tab:blue', or
sns.stripplot(x="neighborhood_flipped", y='monthlyRate', alpha=0.4 ,size=4,color='k',data
plt.xlabel("neighborhood")
plt.ylabel("Monthly Rate (Shekels)")
```

```
Text(0, 0.5, 'Monthly Rate (Shekels)')
```



Part 3 Question 5 - textual Answer:

Write your answer here:

We chose to use a box whiskers plot that describes the different distributions of monthly rates in the eight neighborhoods. It includes the median, the interquartile range, the maximum and the minimum as well as outliers that are marked with dots. Since the distribution is skewed, we chose to arrange the boxes according to the medians (the median is less affected by extreme values). There seem to be differences between the distributions, there are neighborhoods with greater dispersion like Beit va Gan and there are neighborhoods with less dispersion like Gila. The median rent is the highest in the Beit Hakerem neighborhood and the lowest in the Katamonim neighborhood.

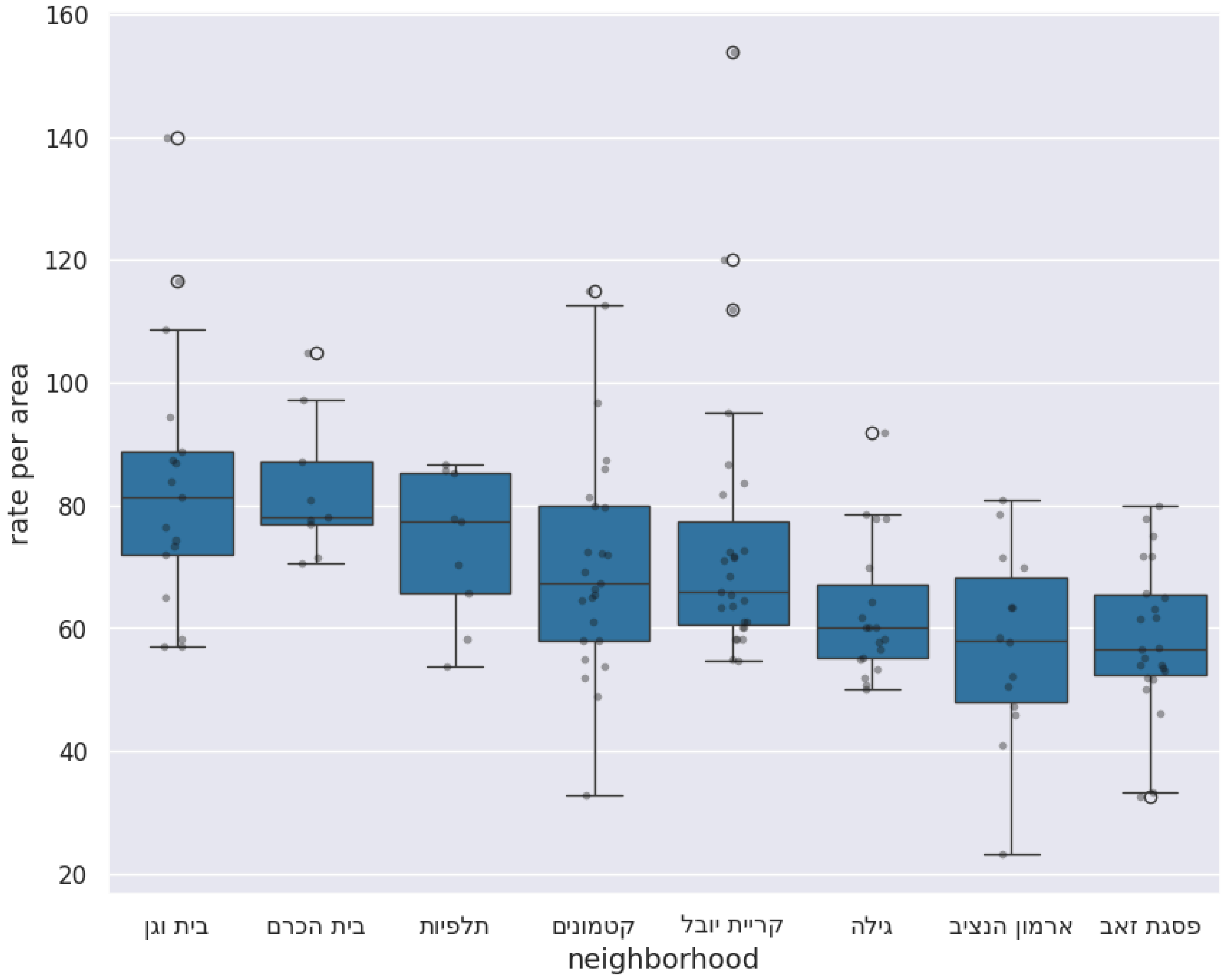
Question 6

Now that we compared the different distributions of monthly rates betwen 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:
#
#
new_df["rate_by_area"] = new_df["monthlyRate"]/new_df["area"]
median_order = list(new_df.groupby('neighborhood_flipped')['rate_by_area'].median().sort_
plt.figure(figsize=(10,8))
sns.boxplot(x= "neighborhood_flipped", y='rate_by_area', data=new_df, color='tab:blue', c
sns.stripplot(x="neighborhood_flipped", y='rate_by_area', alpha=0.4 ,size=4,color='k',dat
plt.xlabel("neighborhood")
plt.ylabel("rate per area")
```

```
<ipython-input-115-426df44c8ea4>:5: 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/us
  new_df["rate_by_area"] = new_df["monthlyRate"]/new_df["area"]
Text(0, 0.5, 'rate per area')
```



Part 3 Question 6 - textual Answer:

Write your answer here:

There may be a different distribution of apartment areas in different neighborhoods, which could explain the previous results.

Therefore, we checked if there is a difference between the distributions of price per square meter in different neighborhoods, so we could compare the neighborhoods.

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).

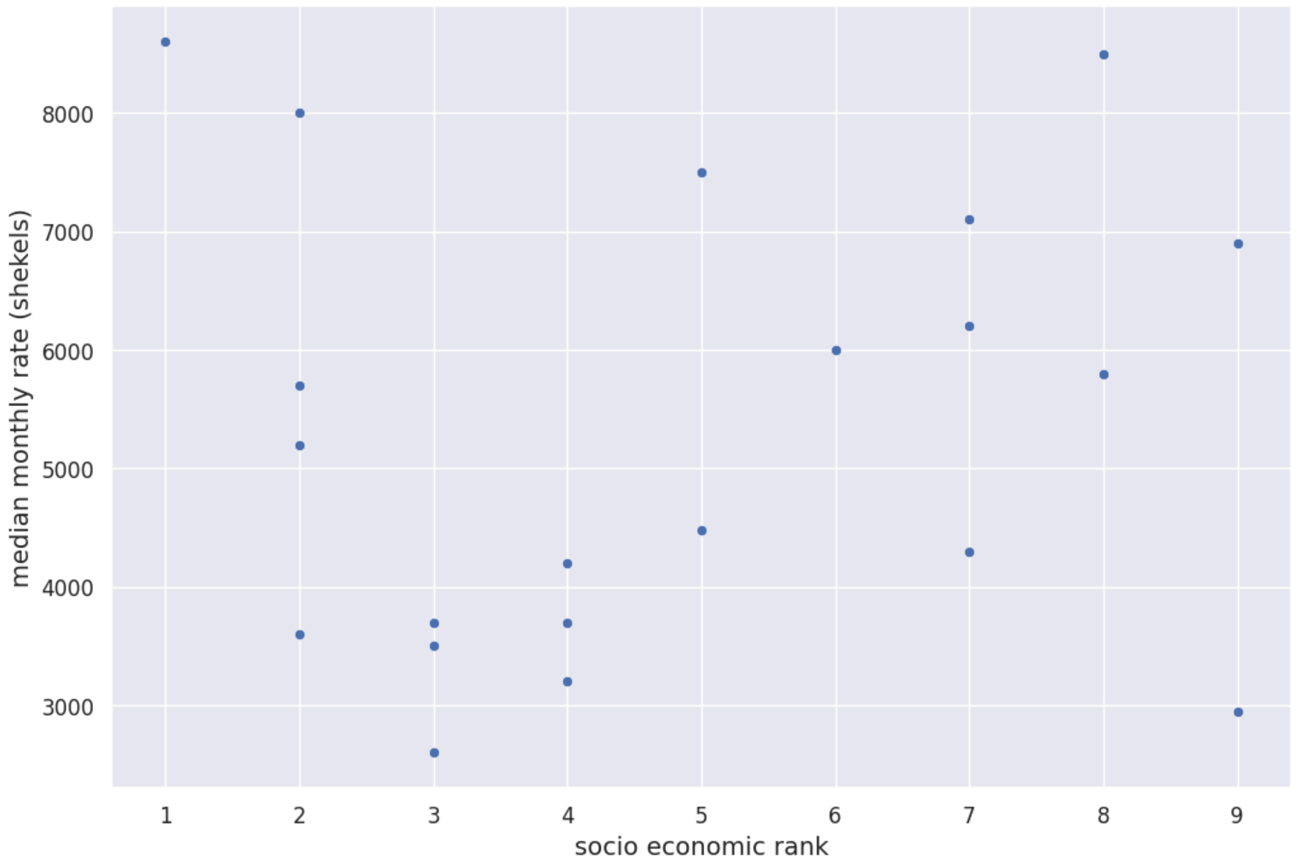
Create an aggregated dataframe where every record represents a neighborhood, with columns for:

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 you conclusion?

```
# Part 3 - Question 7
# Your code goes here:
#
#
aggregated_df = part3_df.groupby(['neighborhood', 'neighborhood_flipped']).agg(
    num_listings=('neighborhood', 'size'),
    median_monthly_rate=('monthlyRate', 'median')
).reset_index()
aggregated_df["socio_economic"] = aggregated_df["neighborhood"].apply(get_neighborhood_ra

aggregated_df_filtered = aggregated_df[aggregated_df["num_listings"] <= 5]

plt.figure(figsize=(12,8))
sns.scatterplot(x='socio_economic', y='median_monthly_rate', data=aggregated_df_filtered)
plt.ylabel("median monthly rate (shekels)")
plt.xlabel("socio economic rank")
```

```
Text(0.5, 0, 'socio economic rank')
```



Part 3 Question 7 - textual Answer:

Write your answer here:

According to the graph, it appears that there is a weak positive association between socio-economic rank and pricing for all neighborhoods with at least 5 listings, but there is a lot of scatter.

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)
```

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'] == 0) | (part4_df['floor'] == 1)) & (part4_d
print("The number of private houses:", len(part4_df[part4_df["is_a_house"] == True]))
sample_data = part4_df[part4_df["is_a_house"]].sample(3)
display(sample_data['description'])
```

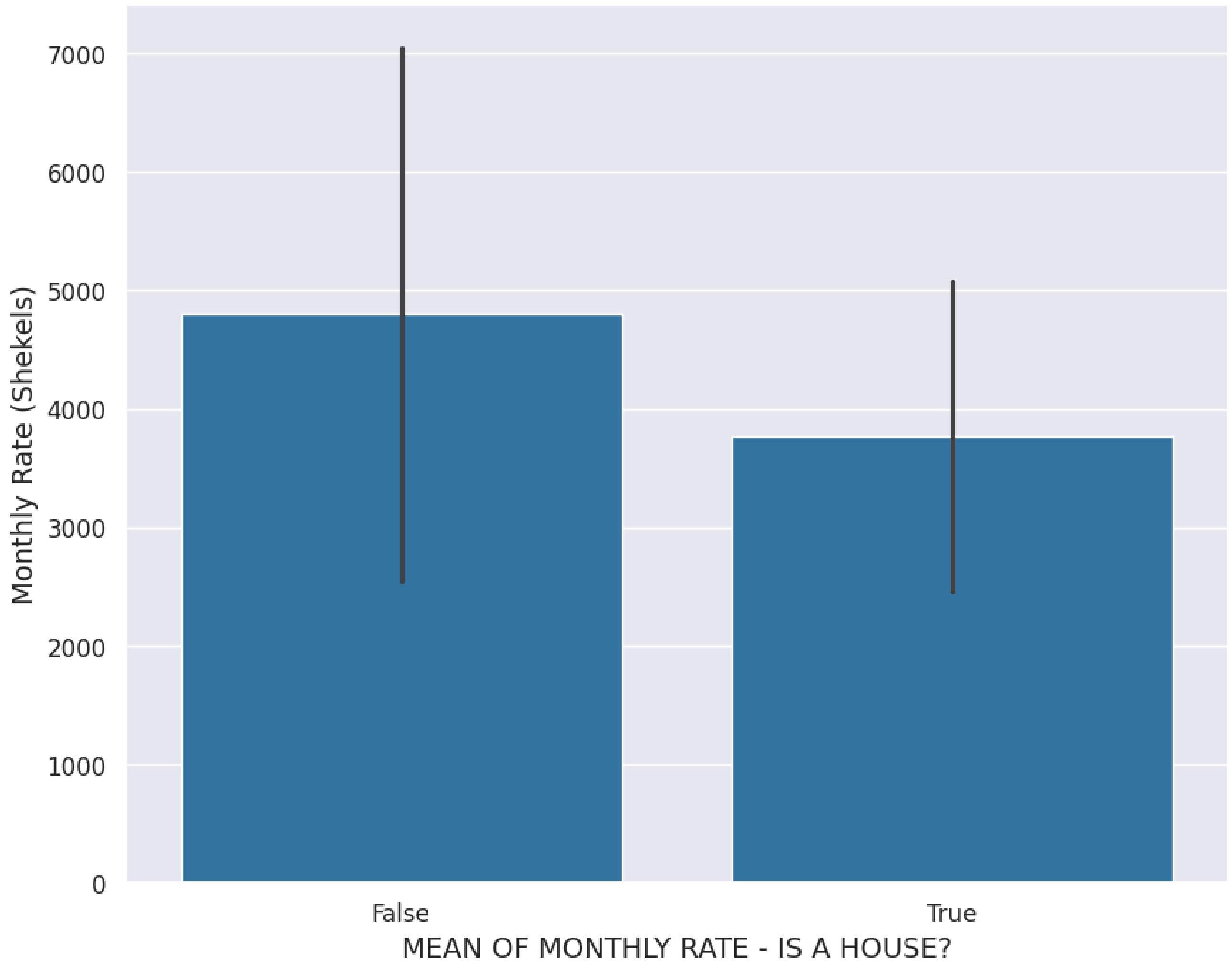
```
The number of private houses: 17
70                    להשכרה, דירה, קומה ראשונה, בירושלים
118                   להשכרה, דירה, קומה ראשונה, בירושלים
4      בס"ד  בפסגת זאב מזרח דירת חדר גדולה משופצת ויפ...
Name: description, dtype: object
```

Question 2

Create a visualization that compares the **average** monthly rates in houses vs. apartments. Which are more expensive on average?

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

plt.figure(figsize=(10,8))
sns.barplot(x='is_a_house', y='monthlyRate', data=part4_df, color='tab:blue', errorbar ='
plt.xlabel("MEAN OF MONTHLY RATE - IS A HOUSE?")
plt.ylabel("Monthly Rate (Shekels)")
```

```
Text(0, 0.5, 'Monthly Rate (Shekels)')
```



Part 4 Question 2 - textual Answer:

Write your answer here:

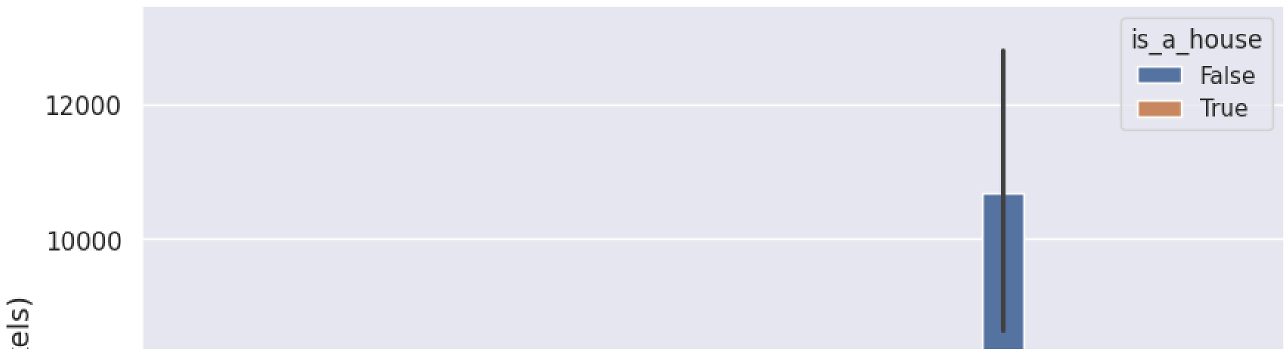
ניתן לראות כי הממוצע של דירות יותר גבוה מבתים פרטיים, אך כן ניתן לראות כי השונות בדירות הרבה יותר גדולה, וכן יש הרבה יותר דירות בדאטה מאשר בתים פרטיים.

Question 3

Now, let's look at the data in a higher resolution. Create a visualization that compares the average monthly rates of houses vs. apartments separetly for any number of rooms. Do the results align with the results from the previous question?

```
# Part 4 - Question 3
# Your code goes here:
#
#


plt.figure(figsize=(10,8))
sns.barplot(x='rooms', y='monthlyRate', data=part4_df, hue='is_a_house')
plt.xlabel("# of Rooms")
plt.ylabel("Monthly Rate (Shekels)")
```

```
Text(0, 0.5, 'Monthly Rate (Shekels)')
```


is_a_house
False
True
els)