Group Details:

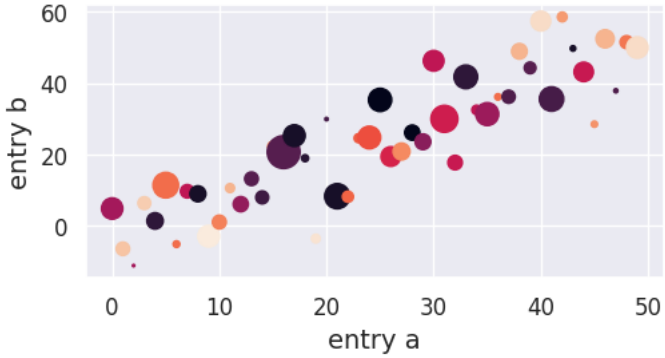
- Name: Shani Cohen, ID: 318341468
- Name: Tal Nir, ID: 208270322
- Name: Yaakov Wilshinsky, ID: 208969949
- Name: Jonathan Shomer Schwartz, ID: 212269591

Helper Functions and Imports

Show code

```
np.random.seed(19680801)  # seed the random number generator.
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

fig, ax = plt.subplots(figsize=(5, 2.7), layout='constrained')
ax.scatter('a', 'b', c='c', s='d', data=data)
ax.set_xlabel('entry a')
ax.set_ylabel('entry b')
```

```
Text(0, 0.5, 'entry b')
```



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','floor','area','entry','description','numFloors']]
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	descriptic
403	3981729	גבעת מרדכי	4500.0	private	3.0	6.0	62.0	10/08/2022	רה יפה, יה, ומשופצת אחרונה. תאים למשפחות
457	3991612	[illegible]	3000.0	private	3.5	3.0	NaN	NaN	מחיר חסר קדים! דירה

And print some summary statistics:

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

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area
count	6.120000e+02	612	612.000000	612	612.000000	611.000000	295.000000
unique	NaN	54	NaN	2	NaN	NaN	NaN
top	NaN	קריית יובל	NaN	private	NaN	NaN	NaN
freq	NaN	66	NaN	600	NaN	NaN	NaN
mean	3.981582e+06	NaN	4717.393791	NaN	2.927288	1.916530	87.664407
std	6.525543e+04	NaN	2195.215139	NaN	1.007350	1.581006	277.004591
min	2.494041e+06	NaN	0.000000	NaN	1.000000	-2.000000	1.000000
25%	3.981694e+06	NaN	3500.000000	NaN	2.000000	1.000000	42.000000
50%	3.987901e+06	NaN	4400.000000	NaN	3.000000	2.000000	60.000000
75%	3.992605e+06	NaN	5800.000000	NaN	3.500000	3.000000	85.000000

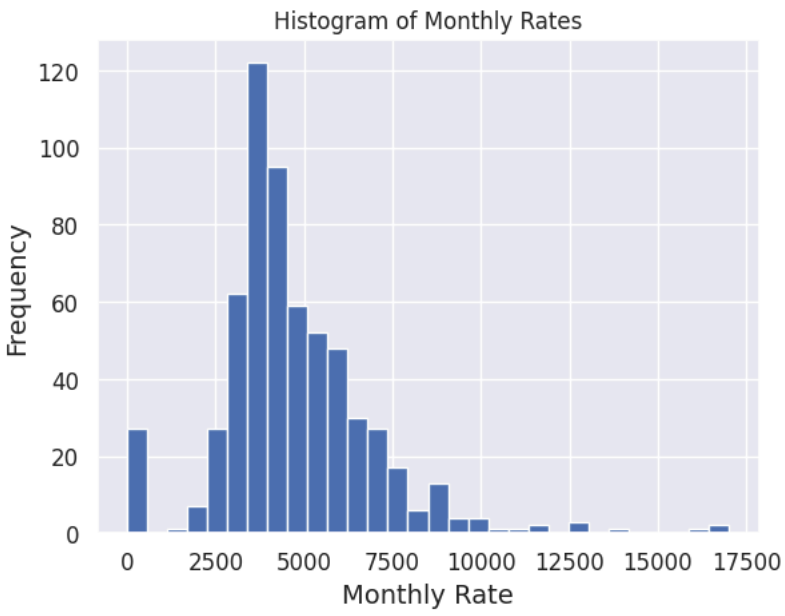
The variables we will focus on are:

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:

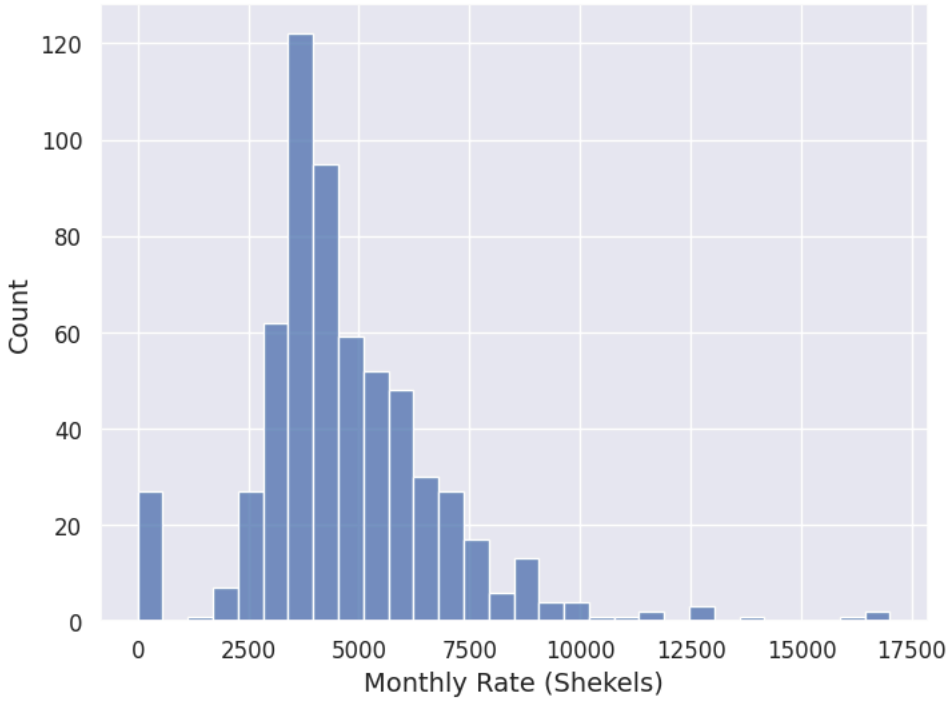
```
# Plotting
plt.hist(rent_df['monthlyRate'], bins=30)
plt.xlabel('Monthly Rate')
plt.ylabel('Frequency')
plt.title('Histogram of Monthly Rates')
plt.show()
```



Solution 1

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

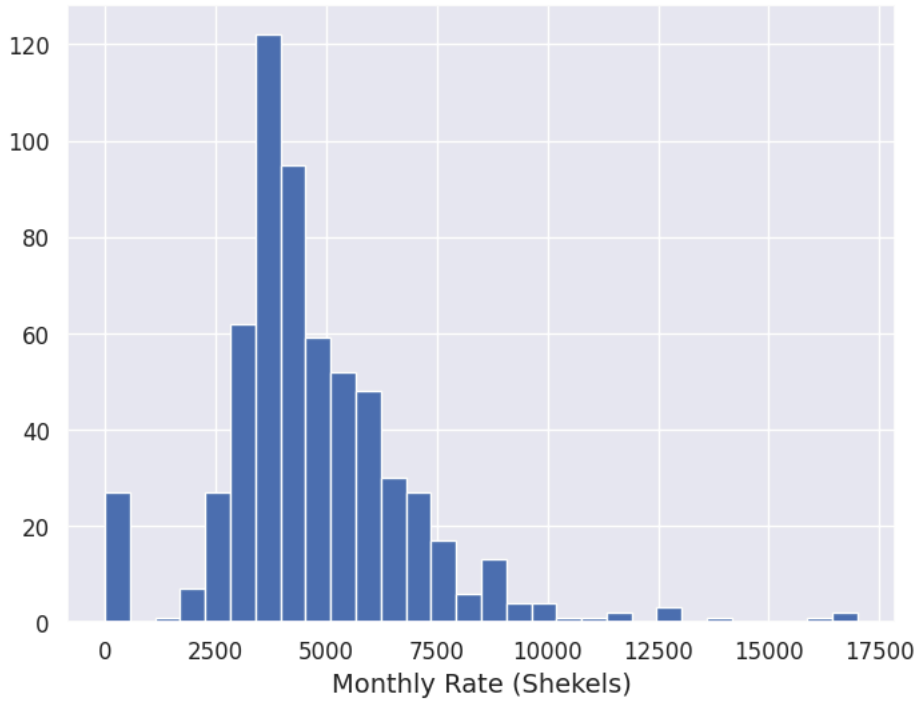
```
Text(0.5, 0, 'Monthly Rate (Shekels)')
```



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].round(3))
```

```
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 outliers

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().reset_index(drop=True)
outlier_df.tail()
```

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	descriptior
20	3983978	קריית משה	0.0	private	4.0	3.0	100.0	10/08/2022	ירת 4 חדרים במצב מצויין - שמורה ביותר .. כולל
21	3985184	נווה יעקב	0.0	private	4.0	1.0	68.0	10/08/2022	ירה במצב שמור מאד. מוזגת, 2

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

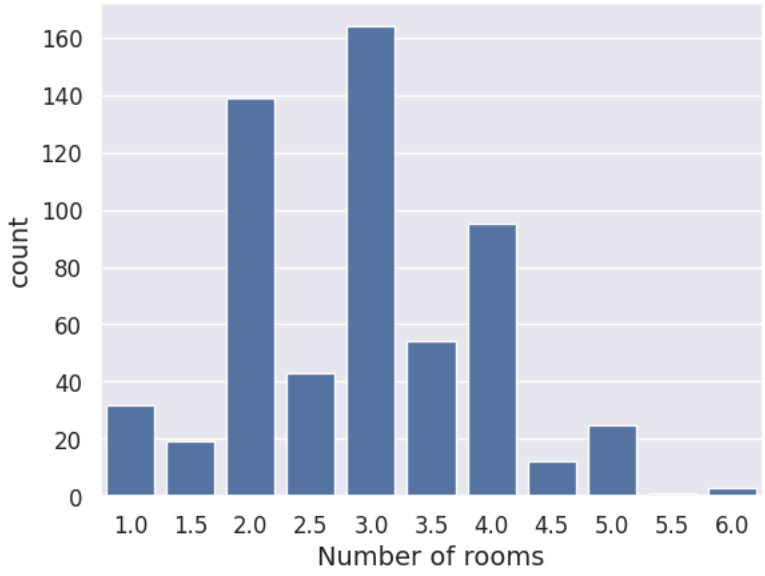
```
clean_df = rent_df[rent_df['monthlyRate'] > 0].reset_index(drop=True)
```

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.

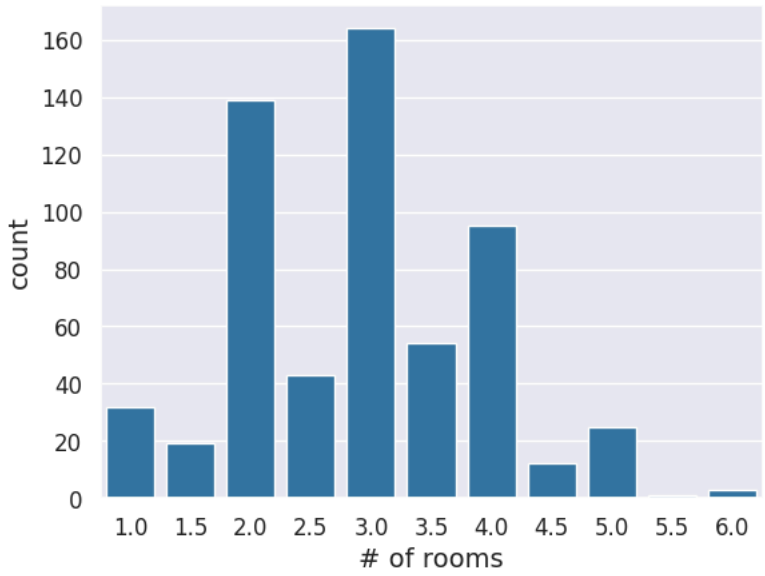
```
sns.countplot(x='rooms', data=clean_df)
plt.xlabel("Number of rooms")
```

```
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")
```

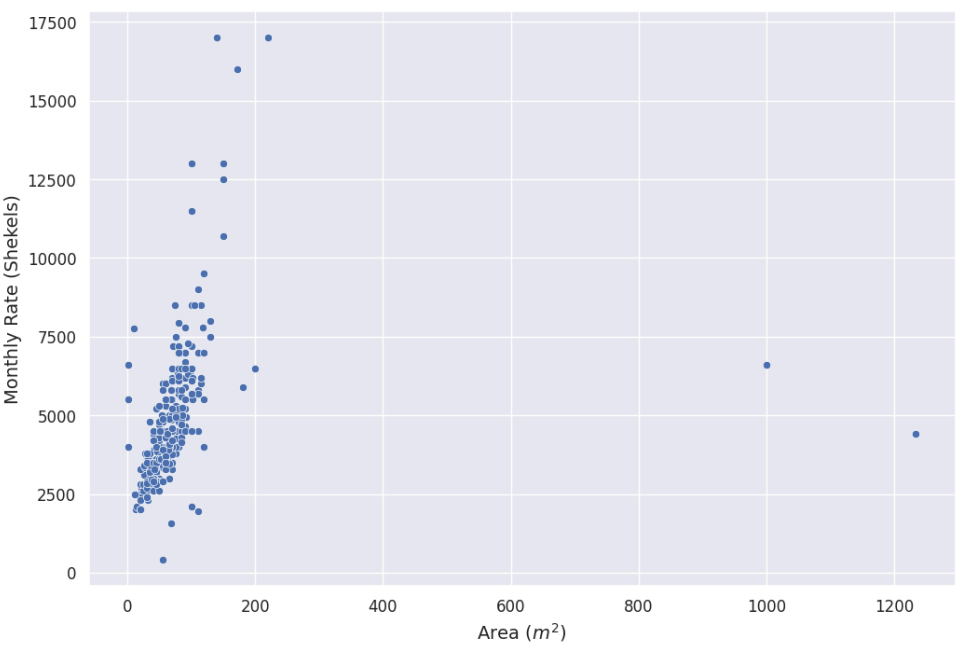


The distribution peaks at three rooms and we also see that "half rooms" are less common.

Can we see an association between apartment area and price?

1. List item
2. List item

```
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	descripti
185	3964340	תלפיות	4400.0	private	2.0	2.0	1234.0	10/08/2022	N
543	3956561	זכרון משה	6600.0	private	3.5	3.0	1000.0	01/07/2022	ה מהממת : ירושלים. דה לרכבת קלה- תחנת

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 עם פינת יש
```

```
clean_df.loc[543,"description"]
```

```
'דירה מהממת בלב ירושלים. צמודה לרכבת הקלה- תחנת הדוידקה. 3 חדרים ענקיים ולכל חדר מרפסת גדולה. חלל כניסה עם פינת יש
```

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.

```
clean_df_area_filtered = clean_df[clean_df['area'] < 800].reset_index(drop=True)
```

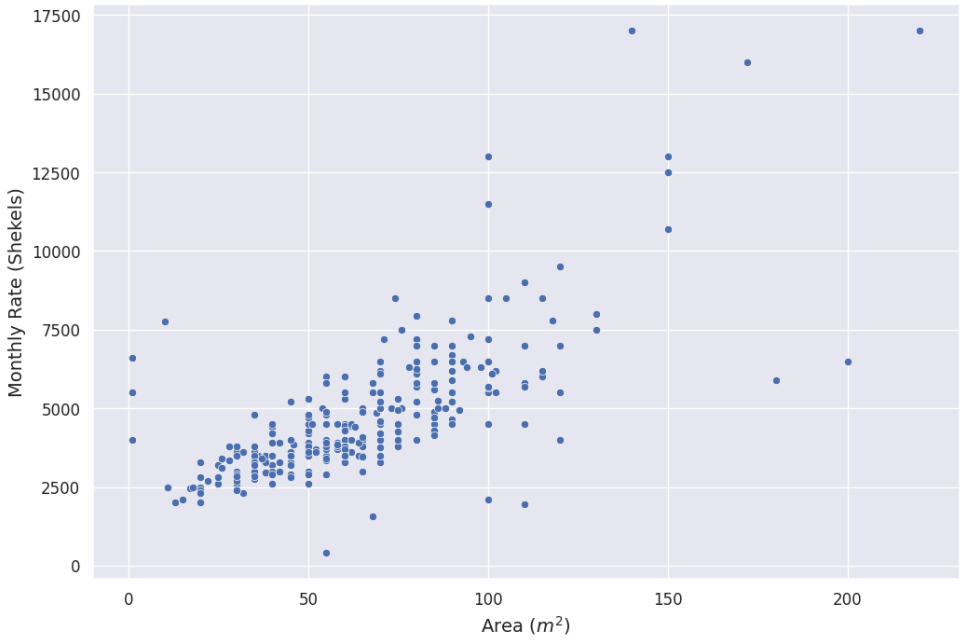
```
outliers_area = rent_df[rent_df['area'] >= 800].reset_index(drop=True)
outliers_area['reason']= "area >= 800"
outlier_df = pd.concat([outlier_df, outliers_area], axis=0, ignore_index=True).drop_duplicates().reset_index(drop=True)
outlier_df
```

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	descriptio
0	3978742	תלפיות	0.0	private	2.0	3.0	NaN	NaN	רה מהממת טיות, חיד/ה או זוג שווה לראות
1	3988036	תלפיות	0.0	private	2.0	0.0	40.0	10/08/2022	רת שני רים עם חצר
2	3993781	זכרון משה	0.0	private	2.0	1.0	NaN	NaN	ס למגורים, 2 רים 4 פסות קשה לשלוח ווא
3	3993281	בקעה	0.0	private	2.0	2.0	4554.0	10/08/2022	שכרה, דירה, מה 2, רושלים
4	3988345	המושבה היוונית	0.0	private	2.0	1.0	40.0	10/08/2022	רת צימר, מת קרקע, 2 רים, חצר אחורית, חצר
5	3987824	קטמונים	0.0	private	2.0	1.0	NaN	NaN	רה 2 חדרים שופצת כניסה טית
6	3989409	קריית יובל	0.0	private	2.0	1.0	33.0	10/08/2022	רת 2 חדרים שכרה מיקום ב 33 מטר
7	3961291	קריית מנחם	0.0	private	2.5	1.0	65.0	10/08/2022	Na
8	3973844	גילה	0.0	private	2.5	4.0	60.0	10/08/2022	רת 2.5 רים, במיקום כזי, שופצת, מרווחת ו
9	3994795	קריית יובל	0.0	private	3.0	1.0	1.0	10/08/2022	חפשת תפ/ה לדירה רושלים , ית יובל. 3 חד
10	3984453	מאה שערים	0.0	private	3.0	1.0	55.0	10/08/2022	תאים לכל טרה משרד חסן וכו
									רת מגורים

Next steps: View recommended plots

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().reset_index(drop=True)

  # 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	descriptic
0	3994505	קריית יובל	2000.0	private	1.0	2.0	13.0	10/08/2022	ידת דיור השכרה ברחוב ראשי של קריית יובל, ה
1	3981298	רחביה	2450.0	private	1.0	1.0	17.0	10/08/2022	רת יחיד 17 טר כולל ספת קטנה
3	3993997	בית וגן	2100.0	private	1.0	0.0	15.0	10/08/2022	רת חדר, כ-15 ר, במיקום כזי אך שקט, משופצ
5	3993552	הר נוף	2000.0	private	1.0	0.0	20.0	10/08/2022	ידה משופצת חיד או למשרד מיקום מצוין
6	3972039	גבעת שאול	2700.0	private	1.0	0.0	22.0	10/08/2022	רת חדר זדשה , כניסה רדת ללא וועד בית , מו
7	3988096	המושבה הגרמנית	2500.0	private	1.0	0.0	18.0	10/08/2022	וונטי לנשים בד. ללא שון. ללא חיות מחמד.
8	3992809	נחלאות	3200.0	private	1.0	2.0	25.0	10/08/2022	ירה המגניבה חלאות, תפנה אחרי קופה ארוכה
10	3983516	הגבעה הצרפתית	2000.0	private	1.0	2.0	20.0	10/08/2022	רת חדר טנה, מסוגננת חמדה, תאימה ליחיד בל

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['monthlyRate'].between(5000, 10000)])
```

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	descriptio
197	3984483	ארנונה	6600.0	private	4.0	2.0	1.0	01/09/2022	שכונת רנונה, רח' ילום יהודה, ירת 4 חדרים מש.

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

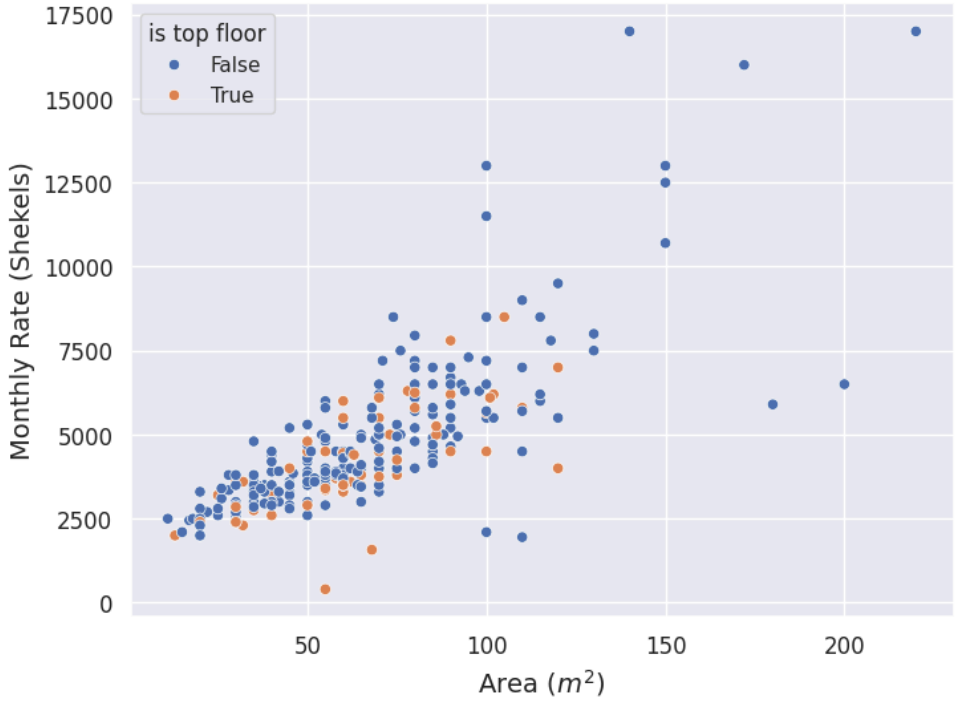
```
#remove the outliers
if clean_df_area_filtered is None:
  print("Can't run until 'clean_df_area_filtered' is created!")
elif outlier_df is None:
  print("Can't run until 'outlier_df' is created!")
else:
  non_ouliers = clean_df_area_filtered['area'] > 10 # get non outliers series of true/false

  # save outliers
  outliers = clean_df_area_filtered[~non_ouliers].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().reset_index(drop=True)

  # 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.)

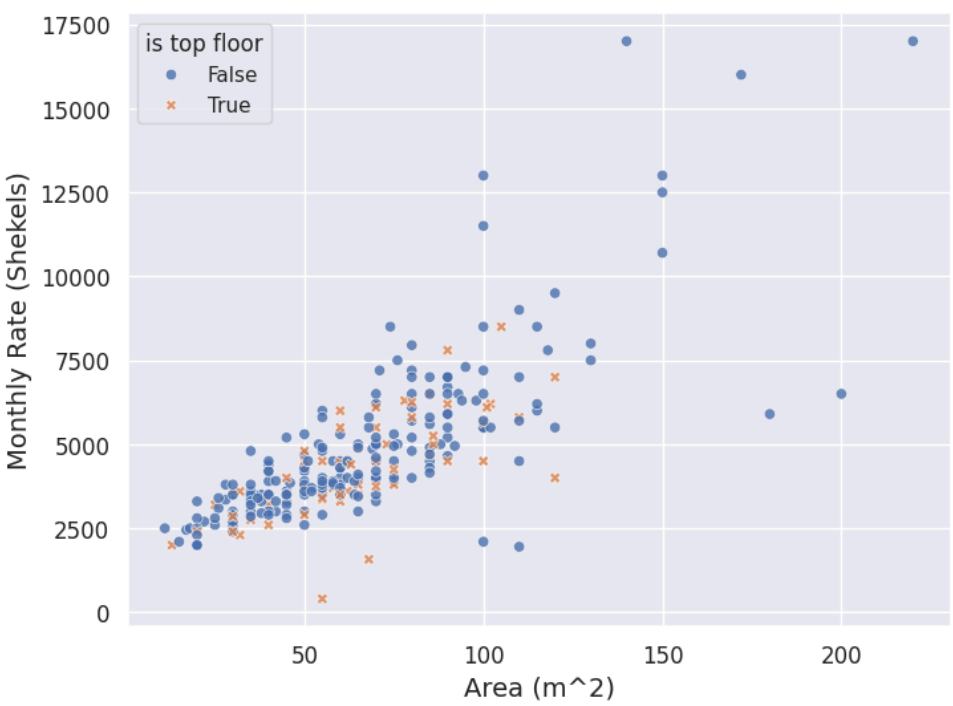
```
clean_df_area_filtered['is top floor'] = clean_df_area_filtered['floor'] == clean_df_area_filtered['numFloors']
plt.figure(figsize=(8,6))
sns.scatterplot(x='area', y='monthlyRate', data=clean_df_area_filtered, hue='is top floor')
plt.xlabel("Area ($m^2$)")
plt.ylabel("Monthly Rate (Shekels)")
```

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



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_filtered['numFloors']
  plt.figure(figsize=(8,6))
  sns.scatterplot(x='area', y='monthlyRate', data=clean_df_area_filtered, alpha=0.8, hue='is top floor', style="is top floor");
  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	descriptic
198	3956418	רחביה	13000.0	agent	4.0	1.0	100.0	NaN	Beautifu renovate furnishe authent arab
236	3985051	טלביה	17000.0	private	4.0	4.0	140.0	10/08/2022	קינג דיוד דנס דירת 4 רים

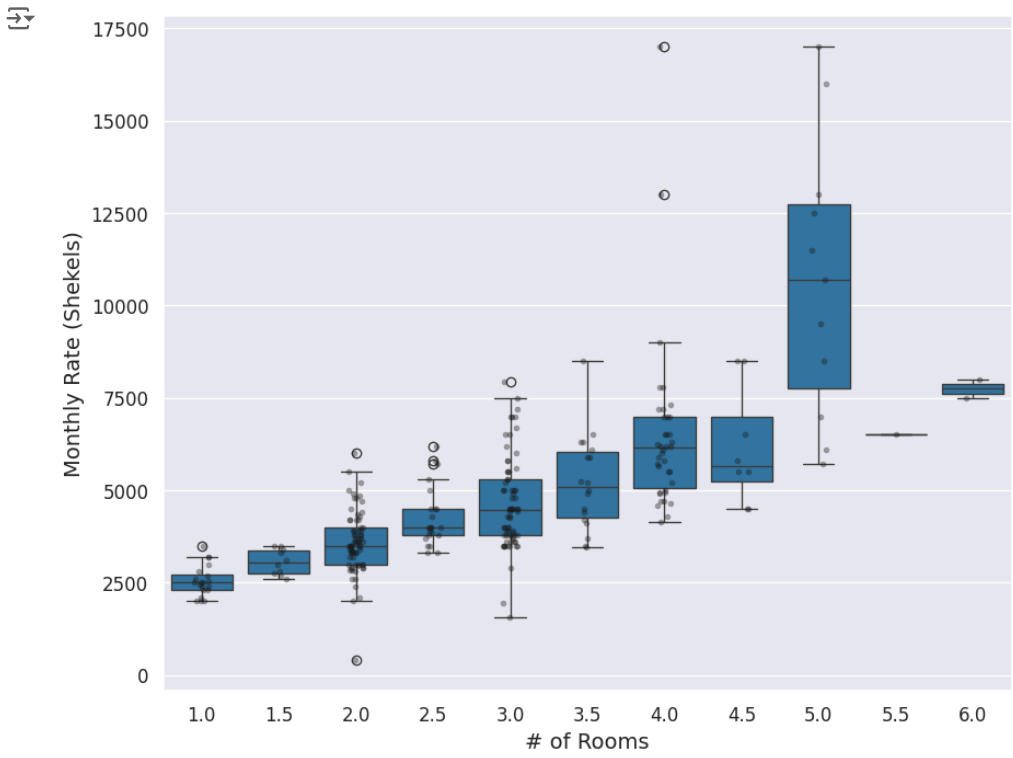
We can see some representation of the more expensive neighborhoods of Jerusalem here.. More on the neighborhoods later on!

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_filtered)
  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', errorbar=None, estimator='median')
  # # 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_filtered)
  # 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: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

1. List item
2. List item

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_filtered)):.0f} %")
```

	monthlyRate								rooms	
	count	mean	std	min	25%	50%	75%	max	count	mean
reason										
'area' <= 10	5.0	5870.000000	1399.821417	4000.0	5500.0	5500.0	6600.0	7750.0	5.0	3.90
'area' >= 800	2.0	5500.000000	1555.634919	4400.0	4950.0	5500.0	6050.0	6600.0	2.0	2.75
area >= 800	3.0	3666.666667	3360.555510	0.0	2200.0	4400.0	5500.0	6600.0	3.0	2.50
monthlyRate <= 0	25.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	25.0	3.28

4 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()
```

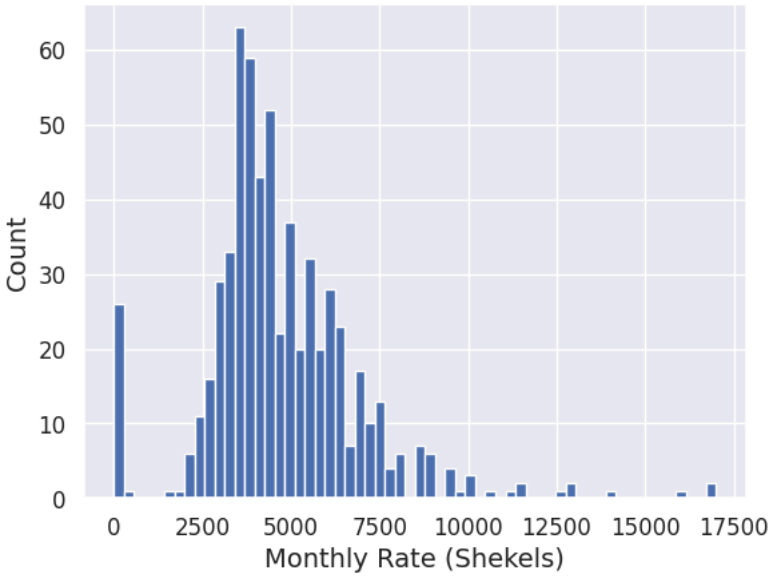
Let's go back to the distribution of monthly rental prices in the dataset. Are there interesting trends in the distribution that we missed in the visualizations before?

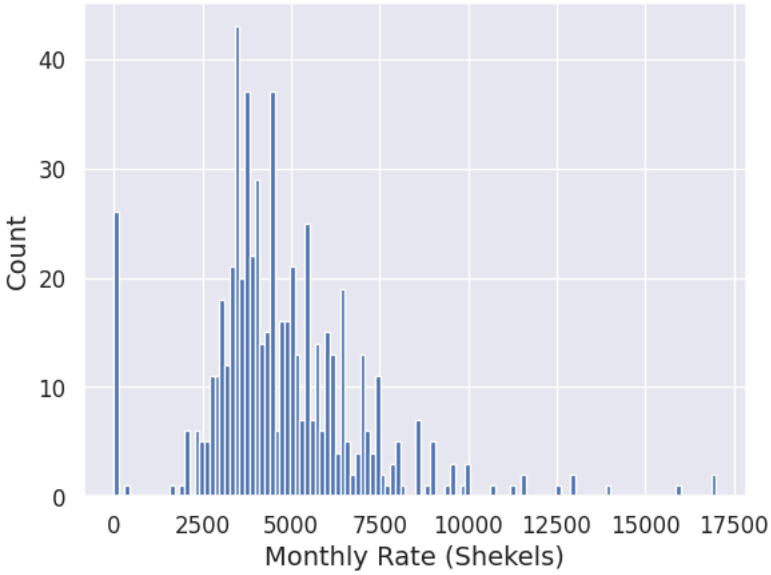
Use only `part1_df` for the coding questions in this part

Question 1

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

```
# Part 1 - Question 1
# Your code goes here:
for i in [20, 60, 120]:
  plt.figure()
  part1_df["monthlyRate"].hist(bins=i)
  plt.xlabel("Monthly Rate (Shekels)")
  plt.ylabel("Count")
```



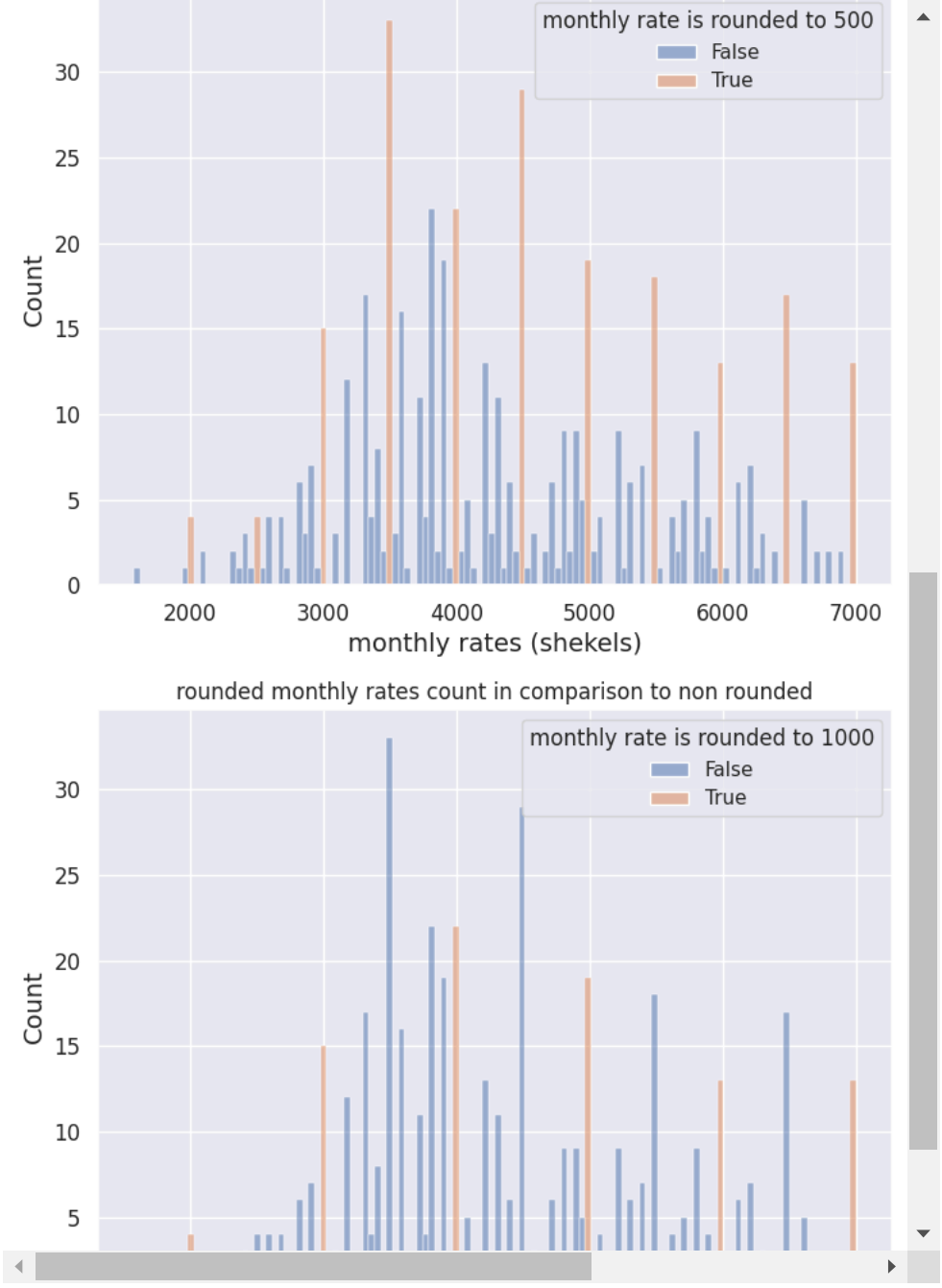


Question 2

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

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

#
# Plot the distribution of values of the 'monthlyRate' column modulu (%) 1000
#


# Part 1 - Question 2
# Your code goes here:

# plt.figure()
# (part1_df["monthlyRate"]%1000).hist(bins=60)
# plt.xlabel("Reminder of Monthly Rate / 1000 (Shekels)")
# plt.ylabel("Count")

round_check_df = part1_df[part1_df["monthlyRate"].between(500,7000)]

for n in (100, 500, 1000):
  plt.figure(figsize = (8,6))
  round_check_df[f'monthly rate is rounded to {n}'] = round_check_df['monthlyRate'] % n == 0
  sns.histplot(x = "monthlyRate", data = round_check_df, bins = 120, hue = f'monthly rate is rounded to {n}')
  plt.title("rounded monthly rates count in comparison to non rounded")
  plt.xlabel("monthly rates (shekels)")
```



rounded monthly rates count in comparison to non rounded

Part 1 Question 2 - textual Answer:

Write your answer here:

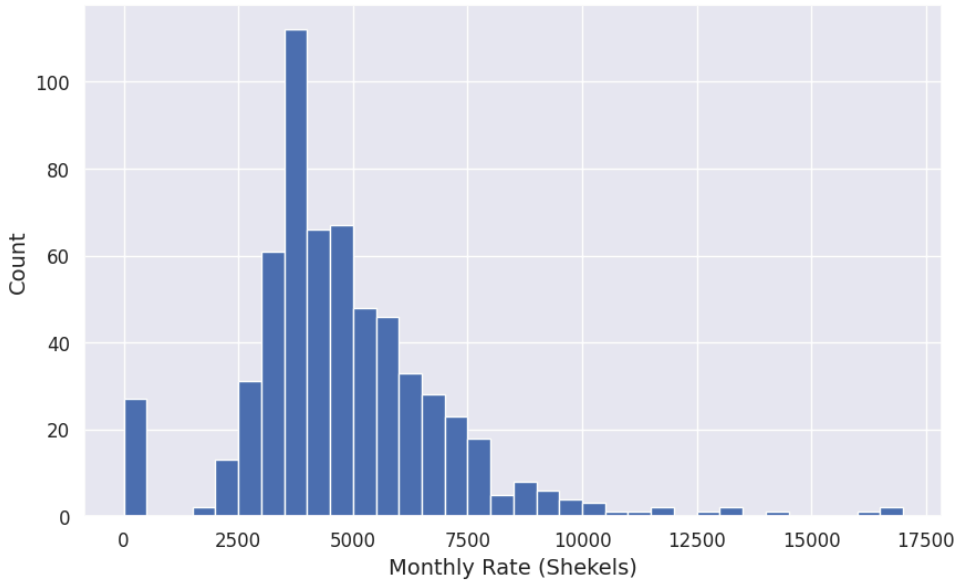
We can see that rounding to 500 includes almost any peak of the mothly rates between 500 to 7000

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 = min(part1_df["monthlyRate"])
max_price = max(part1_df["monthlyRate"])

num_bins = int((max_price - min_price) / 500) + 1
bin_edges = np.arange(min_price, max_price+500, 500)

plt.figure(figsize=(10, 6))
plt.hist(part1_df["monthlyRate"], bins=bin_edges)
plt.xlabel("Monthly Rate (Shekels)")
plt.ylabel("Count")
```

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



Part 1 Question 3 - textual Answer:

Write your answer here: Here, when the bins are exactly in multiplications of 500, each bin has only 1 rounded price, such that the peaks caused by the roundind are canceled.

We can see a "drop" in prices near 5000 in the graph. nevertheless, it is unclear to our opinion if the 5000 "drop" is due to tax considerations because of two reasons.

1. there are significant "drop" also near 4000, and 8000, without ant relevance to tax.
2. the 3 big "drops" of 4000, 5000, 8000 could also just fit to a right tailed function.

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.

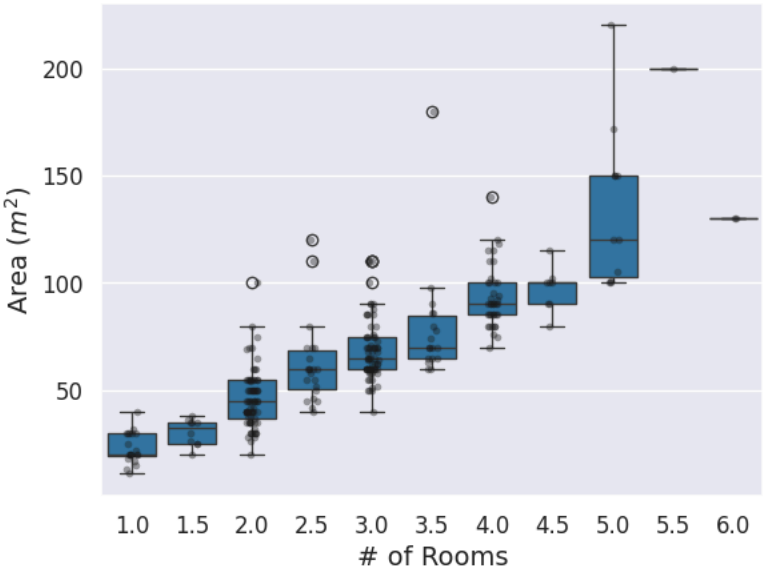
Use only `part2_df` for the coding questions in this part

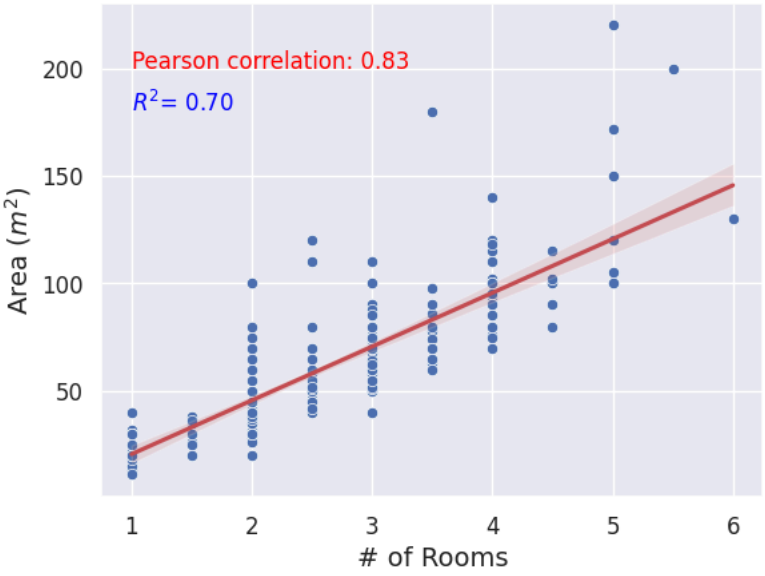
Question 1

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:

from scipy.stats import pearsonr
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

plt.figure()
sns.boxplot(x='rooms', y='area', data=part2_df, color='tab:blue')
sns.stripplot(x='rooms', y='area', alpha=0.4 ,size=4,color='k',data=part2_df)
plt.xlabel("# of Rooms")
plt.ylabel("Area ($m^2$)")

plt.figure()
sns.scatterplot(x='rooms', y='area', data=part2_df)

model = LinearRegression()
X = part2_df[['rooms']]
y = part2_df['area']
model.fit(X, y)

corr, _ = pearsonr(part2_df['rooms'], part2_df['area'])

sns.regplot(x='rooms', y='area', data=part2_df, scatter=False, color='r', line_kws={'label': 'Linear Regression'})

y_pred = model.predict(X)
r_squared = r2_score(y, y_pred)

plt.text(1, 200, f'Pearson correlation: {corr:.2f}', fontsize=12, color='red')
plt.text(1, 180, f'$R^2$= {r_squared:.2f}', fontsize=12, color='blue')

plt.xlabel("# of Rooms")
plt.ylabel("Area ($m^2$)")
```

```
Text(0, 0.5, 'Area ($m^2$)')
```





Part 2 Question 1 - textual Answer:

Write your answer here: בחרנו בתרשים הראשון כדי להציג קשר בין משתנה קטגוריאלי (מספר החדרים) לבין משתנה רציף (השטח). הבוקס פלוט מאפשר לראות מדדי סיכום רלוונטיים של גודל השטח עבור כל מספר חדר (חציון, טווח, וכו'). הסטריפלוט מאפשר תצוגה מפורטת יותר של התפלגות הנתונים, וכך לראות בכל קטגוריה את הצפיפות והפיזור של הערכים, כולל תצפיות קיצוניות. השילוב של שניהם יחד מאפשר תצוגה מקיפה של הקשר בין מספר החדרים לשטח. ניתן לראות בתרשים שאכן קיים קשר בין השניים - ככל שמספר החדרים גדל כך השטח גדל. ניתן לראות זאת גם כיוון שהחציונים מסודרים במגמת עליה, וגם לפי פיזור הנקודות עצמן. את הקשר ניתן לראות גם בתרשים השני בו מוצגת ריגרסיה לינארית עם קורלציה מובהקת. ניתן לראות ש70% מהשונות בשטח מוסברת ע"י מספר החדרים.

Question 2

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

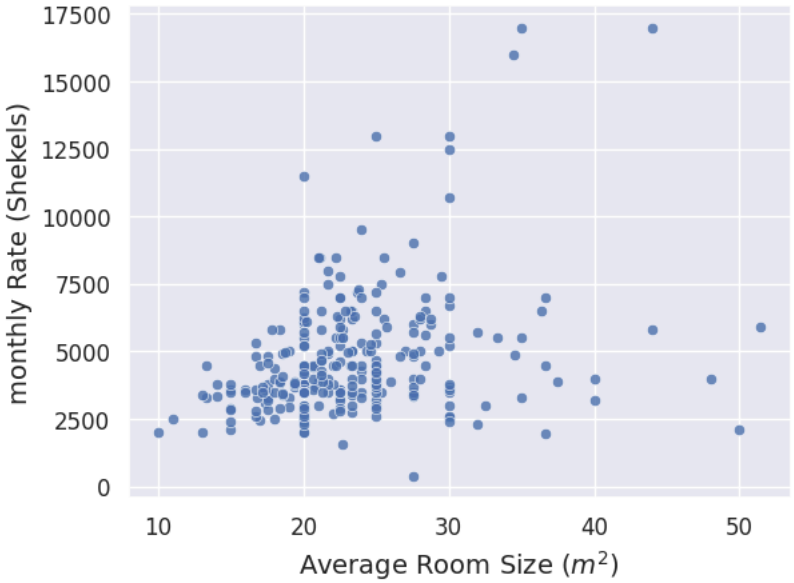
```
# Part 2 - Question 2
# Your code goes here:

part2_df["averageRoomSize"] = part2_df['area']/part2_df['rooms']
```

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()
sns.scatterplot(x='averageRoomSize', y='monthlyRate', data=part2_df, alpha=0.8)
plt.xlabel("Average Room Size ($m^2$)")
plt.ylabel("monthly Rate (Shekels)")
```

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



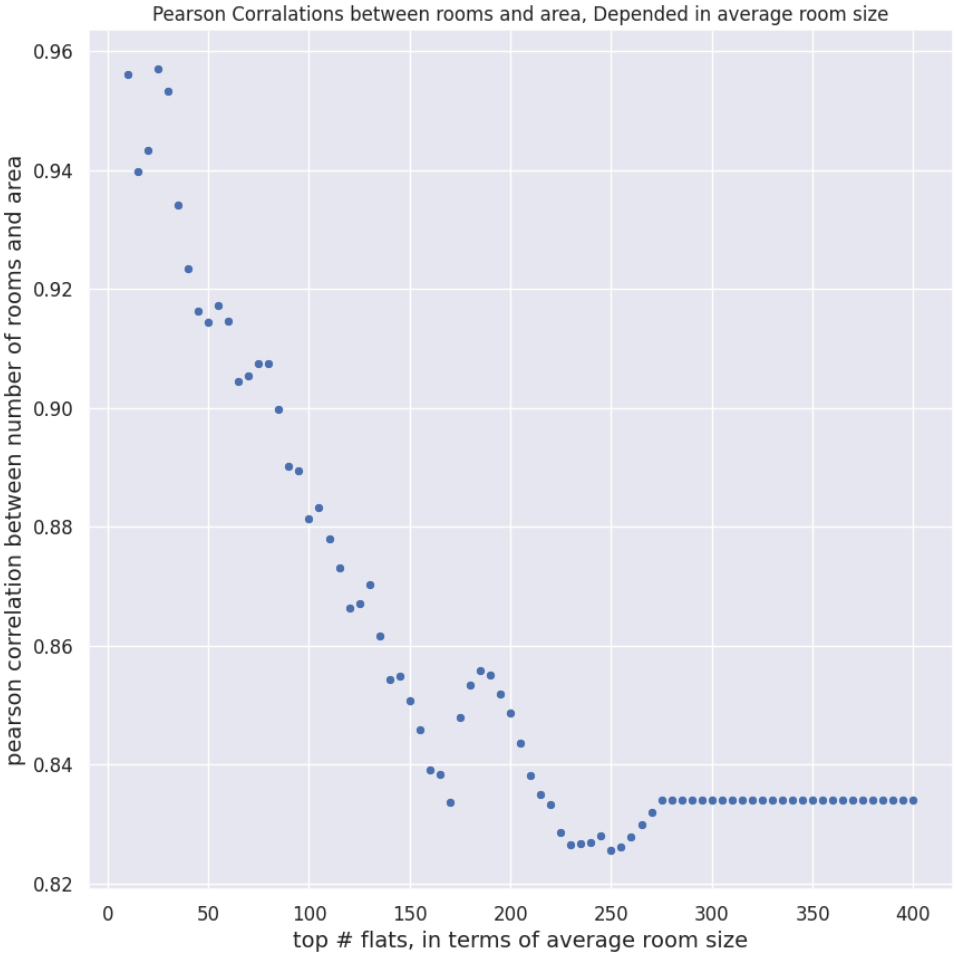
Question 4 - **bonus**

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:

correlations = []
for n in np.arange(400,5,-5):
  top_ones = part2_df.nlargest(n, 'averageRoomSize')
  cor, _ = pearsonr(top_ones['area'], top_ones['rooms'])
  correlations.append([n,cor])
correlations = np.array(correlations)[::-1].T
plt.figure(figsize = (10,10))
sns.scatterplot(x = correlations[0],y = correlations[1])


plt.title(f'Pearson Corralations between rooms and area, Depended in average room size')
plt.xlabel('top # flats, in terms of average room size')
plt.ylabel('pearson correlation between number of rooms and area')
```

```
Text(0, 0.5, 'pearson correlation between number of rooms and area')
```



Part 2 Question 4 - textual Answer:

Write your answer here:

Our hypothesis was that the increse of the variance of the monthly rates with the average room size was due to the decrese of correlation between rooms and area in those cases. if so, the feature of "average room size" ability to explain the monthly rates will weaken, due to its explanatory power dividing between "rooms" and "area". therefore, we plotted the conection between the correlation of rooms-area and the average room numbers, and found a refuting evidece - the rooms-area correlation remained high, end even higher, in the top "average room size" flats.

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['socioEconomicRank'])}

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 the neighborhood names readable
```

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.

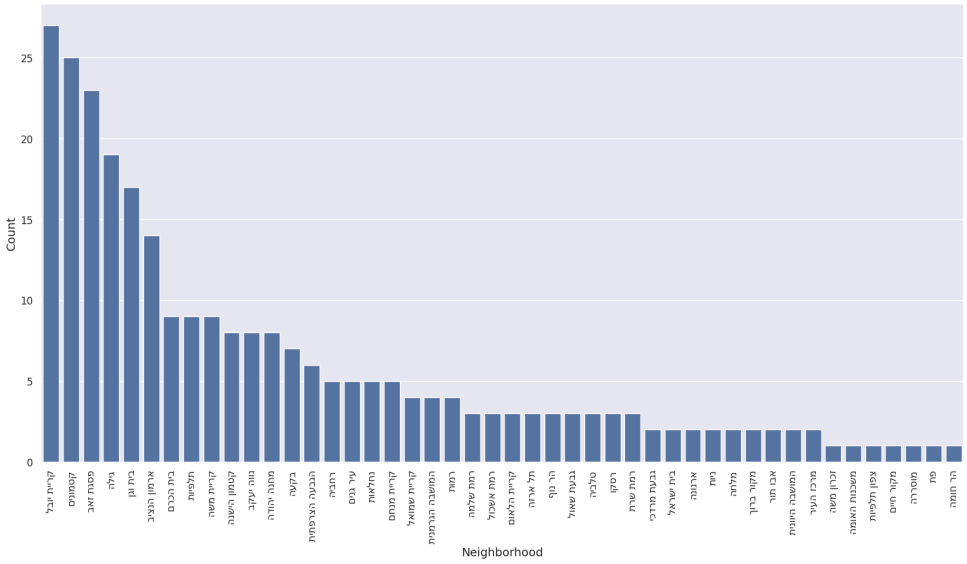
```
# Part 3 - Question 1
# Your code goes here:

len(part3_df["neighborhood"].unique())
```

```
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:

neighborhood_counts = part3_df['neighborhood_flipped'].value_counts()
sorted_neighborhoods = neighborhood_counts.index
plt.figure(figsize=(20, 10))
sns.countplot(data=part3_df, x='neighborhood_flipped', order=sorted_neighborhoods)
plt.xlabel('Neighborhood')
plt.ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```



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'].value_counts()
neighborhood_less_than_5 = neighborhood_counts[neighborhood_counts < 5].index
print("The number of neighborhoods with less than 5 listings:", len(neighborhood_less_than_5))

filtered_df = part3_df[part3_df['neighborhood'].isin(neighborhood_less_than_5)]
print("Propotion Of listings from <5 listings neighborhoods:", round(len(filtered_df) / len(part3_df), 3))

top_8_neighborhoods = neighborhood_counts.nlargest(8)
top_8_neighborhoods = top_8_neighborhoods.index
frequent_neighborhoods_df = part3_df[part3_df['neighborhood'].isin(top_8_neighborhoods)]
print("Propotion Of listings from top 8 neighborhoods:", round(len(frequent_neighborhoods_df) / len(part3_df), 3))
```

```
The number of neighborhoods with less than 5 listings: 28
Propotion Of listings from <5 listings neighborhoods: 0.234
Propotion Of listings from top 8 neighborhoods: 0.524
```

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:

frequent_neighborhoods_df
```

	propertyID	neighborhood	monthlyRate	mefarsem	rooms	floor	area	entry	descripti
0	3994505	קריית יובל	2000.0	private	1.0	2.0	13.0	10/08/2022	דת דיור שכרה ברחוב אשי של קריית יובל, ה
3	3993997	בית וגן	2100.0	private	1.0	0.0	15.0	10/08/2022	ת חדר, כ-15 ר, במיקום כזי אך שקט, משופצ
4	3994399	פסגת זאב	2300.0	private	1.0	1.0	32.0	10/08/2022	"ד בפסגת ב מזרח דירת ר גדולה משופצת ויפ
11	3986231	קריית יובל	2600.0	private	1.0	1.0	30.0	10/08/2022	ירה שטופת וש, מגיעה והטת- מיטה, ארון בגד
									ת חדר

Next steps: View recommended plots

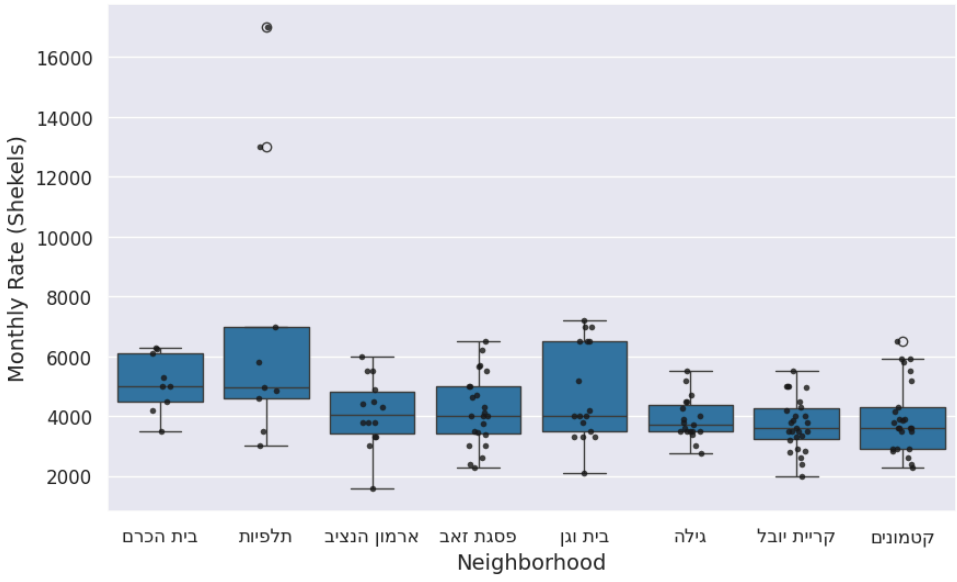
Question 5

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_monthly_rate = frequent_neighborhoods_df.groupby('neighborhood_flipped')['monthlyRate'].median()
median_monthly_rate_sorted = median_monthly_rate.sort_values(ascending=False).index

plt.figure(figsize=(10,6))
sns.boxplot(x='neighborhood_flipped', y='monthlyRate', data=frequent_neighborhoods_df, color='tab:blue', order=median_monthly_rate_sorted)
sns.stripplot(x='neighborhood_flipped', y='monthlyRate', alpha=0.8 ,size=4,color='k',data=frequent_neighborhoods_df, order=median_monthly_rate_sorted)
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: בחרנו בתרשים כדי להציג קשר בין משתנה קטגוריאלי (השכונה) לבין משתנה רציף (שכ"ד). הבוקס פלוט מאפשר לראות את החציון, אשר פחות מושפע מקצוות קיצוניים ביחס לממוצע, כמו שאכן יש לנו (בתלפיות לדוגמא, כפי שניתן לראות בגרף). בנוסף, הסטריפלוט מאפשר לראות תצוגה מפורטת יותר של התפלגות הנתונים, וכך לראות בכל שכונה את הצפיפות והפיזור של הערכים, כולל תצפיות קיצוניות. לפי הגרף נראה שהשכונות בהן שכ"ד הוא היקר ביותר הן בית הכרם ותלפיות, ואחריהן כל השאר עם שכ"ד חציוני די דומה. כמו כן, ניתן לראות שפיזור המחירים משתנה בין השכונות. בבית וגן לדוגמא פיזור המחירים גדול יחסית, ואילו בגילה הוא קטן יחסית.

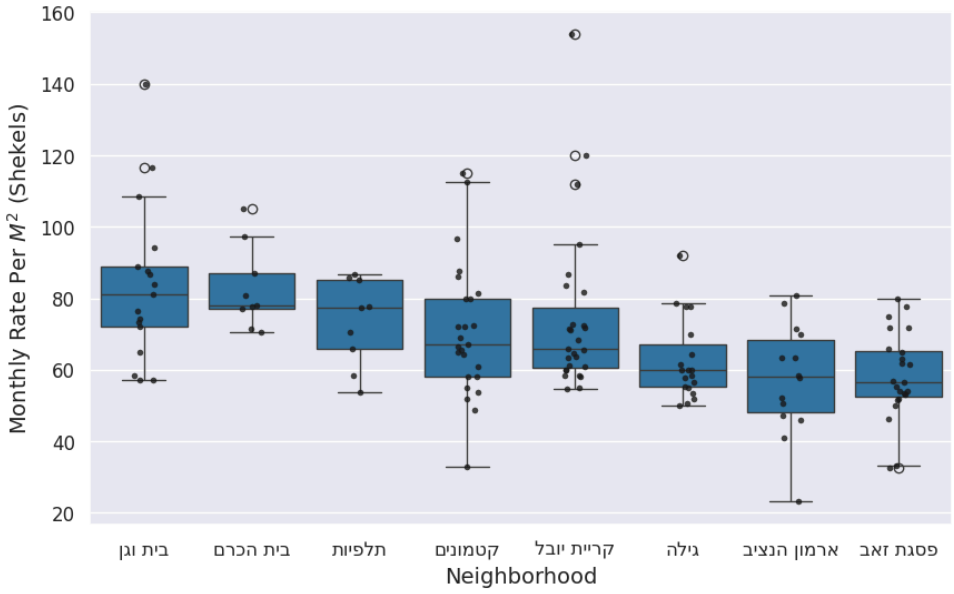
Question 6

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:

frequent_neighborhoods_df["monthlyRatePerMeter"] = frequent_neighborhoods_df['monthlyRate'] / frequent_neighborhoods_df['area']

median_monthly_rate = frequent_neighborhoods_df.groupby('neighborhood_flipped')['monthlyRatePerMeter'].median()
median_monthly_rate_sorted = median_monthly_rate.sort_values(ascending=False).index

plt.figure(figsize=(10,6))
sns.boxplot(x='neighborhood_flipped', y='monthlyRatePerMeter', data=frequent_neighborhoods_df, color='tab:blue', order=median_monthly_rate_sorted)
sns.stripplot(x='neighborhood_flipped', y='monthlyRatePerMeter', alpha=0.8 ,size=4,color='k',data=frequent_neighborhoods_df, order=median_monthly_rate
plt.xlabel("Neighborhood")
plt.ylabel("Monthly Rate Per $M^2$ (Shekels)")
```

```
<ipython-input-44-bf7879e03517>:4: 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/i
  frequent_neighborhoods_df["monthlyRatePerMeter"] = frequent_neighborhoods_df['monthlyRate'] /
Text(0, 0.5, 'Monthly Rate Per $M^2$ (Shekels)')
```



Part 3 Question 6 - textual Answer:

Write your answer here: כדי להוציא את הפרמטר של שטח הדירה מחוץ למשוואה, הוספנו מדד של מחיר למטר רבוע. ניתן לראות שדירוג השכונות לפי מדד זה שונה מהדירוג שראינו בשאלה הקודמת (כאשר שכר הדירה לא היה מנורמל ביחס לשטחה).

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).

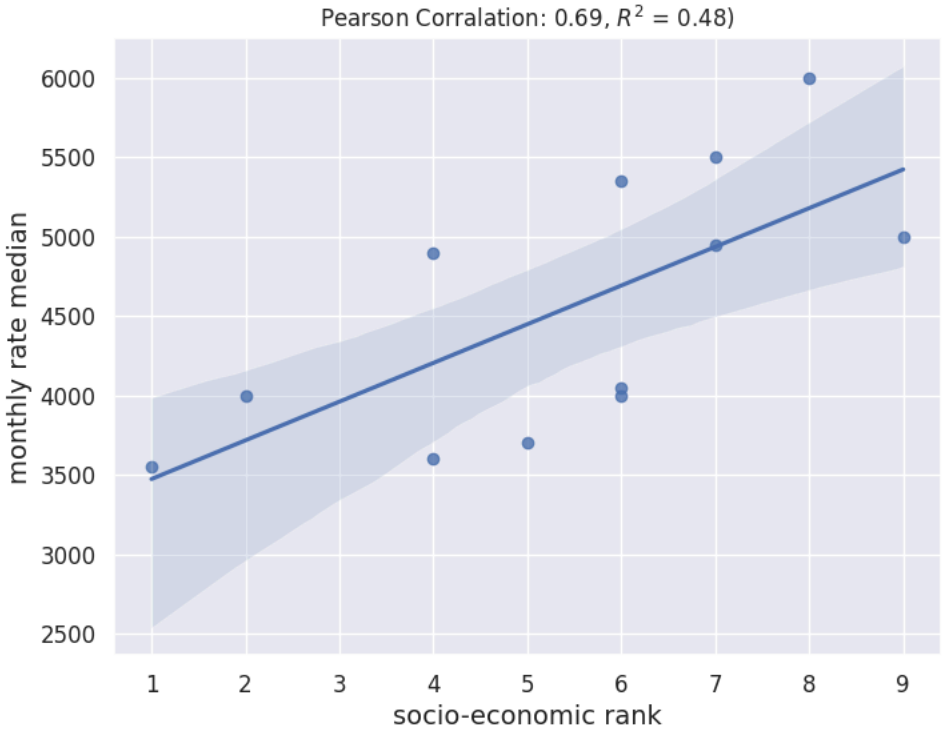
Create an aggregated dataframe where every record represents a neighborhood, with columns for:

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:

def get_median(name, column):
  df = part3_df[part3_df['neighborhood'] == name]
  median = np.median(df[column])
  return median

neighborhoods_df = part3_df['neighborhood'].value_counts().reset_index()
neighborhoods_df = neighborhoods_df[neighborhoods_df['count']>5]
neighborhoods_df['flipped neighborhood name'] = neighborhoods_df['neighborhood'].apply(lambda name: name[::-1])
neighborhoods_df['monthly rate median'] =  neighborhoods_df['neighborhood'].apply(lambda name: get_median(name, 'monthlyRate'))
neighborhoods_df['socio-economic rank'] = neighborhoods_df['neighborhood'].apply(lambda name: get_neighborhood_rank(name))
neighborhoods_df_cleared = neighborhoods_df[~neighborhoods_df['socio-economic rank'].isna()]

plt.figure(figsize= (8,6))
sns.regplot(x = 'socio-economic rank', y = 'monthly rate median', data = neighborhoods_df_cleared)
cor, _ = pearsonr(neighborhoods_df_cleared['socio-economic rank'], neighborhoods_df_cleared['monthly rate median'])
plt.title(f'Pearson Corralation: {cor.round(2)}, $R^2$ = {round(cor**2,2)})')
```

```
Text(0.5, 1.0, 'Pearson Corralation: 0.69, $R^2$ = 0.48)')
```



Part 3 Question 7 - textual Answer:

Write your answer here: we can see a clear corralation, when the socio-economic rank of neghbohoods explaines 48% of the varience in monthly rate medians of them.

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.head(4)
part4_df["is_a_house"] = (part4_df['floor'].isin([1.0,0.0]) & part4_df['numFloors'].isin([1.0]))
private_houses_df = part4_df[part4_df["is_a_house"]]
print("the number of private houses:", len(private_houses_df))
print('')
print("descriptions of three random listings with 'is_a_house' equal to True:")
random_houses_description = part4_df.loc[part4_df['is_a_house'] == True, "description"].sample(3)
for description in random_houses_description:
    print(str(description))
```

```
the number of private houses: 17

descriptions of three random listings with 'is_a_house' equal to True:
nan
דירה בבית פרטי, כניסה פרטית, כניסה מיידית. המחיר כולל ארנונה
להשכרה, דירה, קומה ראשונה, בירושלים
```

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:

apartments_df = part4_df[~part4_df["is_a_house"]]
plt.figure(figsize=(6,8))
sns.barplot(x='is_a_house', y='monthlyRate', data=part4_df, color='tab:blue', estimator='mean')
stripplot = sns.stripplot(x='is_a_house', y='monthlyRate', alpha=0.4 ,color='g',data=part4_df)
stripplot.set_xticklabels(['Apartment', 'Private House'])
plt.ylabel('average monthly rate (shekels)')
plt.xlabel('')
plt.title('Average Monthly Rates in Houses vs. Apartments')
ratio = np.mean(private_houses_df['monthlyRate']) / np.mean(apartments_df['monthlyRate'])
plt.text(1.6, 0, f'Ratio between private houses and apartmens monthly rates avarages: {ratio.round(2)}', rotation=90, verticalalignment='bottom')
```