**Group Details:**

- Name: Yonah Huppert ID: 328949813
- Name:Ben Towler ID: 345576433
- Name: Hallel Azulai ID:212667497
- Name: Ido Zchut ID: 206679029

## 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','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 |
|---|---|---|---|---|---|---|---|---|
| **403** | 3981729 | גבעת מרדכי | 4500.0 | private | 3.0 | 6.0 | 62.0 | 10/08/2022 |
| **457** | 3991612 | קרית 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 | area | entry | description | numFloors |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 6.120000e+02 | 612 | 612.000000 | 612 | 612.000000 | 611.000000 | 295.000000 | 292 | 596 | 610.000000 |
| unique | NaN | 54 | NaN | 2 | NaN | NaN | NaN | 17 | 578 | NaN |
| top | NaN | קריית יובל | NaN | private | NaN | NaN | NaN | 10/08/2022 | להשכרה, דירה,<br>קומה ראשונה,<br>בירושלים | NaN |
| freq | NaN | 66 | NaN | 600 | NaN | NaN | NaN | 259 | 8 | NaN |
| mean | 3.981582e+06 | NaN | 4717.393791 | NaN | 2.927288 | 1.916530 | 87.664407 | NaN | NaN | 3.908197 |
| std | 6.525543e+04 | NaN | 2195.215139 | NaN | 1.007350 | 1.581006 | 277.004591 | NaN | NaN | 1.978065 |
| min | 2.494041e+06 | NaN | 0.000000 | NaN | 1.000000 | -2.000000 | 1.000000 | NaN | NaN | 1.000000 |
| 25% | 3.981694e+06 | NaN | 3500.000000 | NaN | 2.000000 | 1.000000 | 42.000000 | NaN | NaN | 3.000000 |
| 50% | 3.987901e+06 | NaN | 4400.000000 | NaN | 3.000000 | 2.000000 | 60.000000 | NaN | NaN | 4.000000 |
| 75% | 3.992605e+06 | NaN | 5800.000000 | NaN | 3.500000 | 3.000000 | 85.000000 | NaN | NaN | 4.000000 |

The variables we will focus on are:

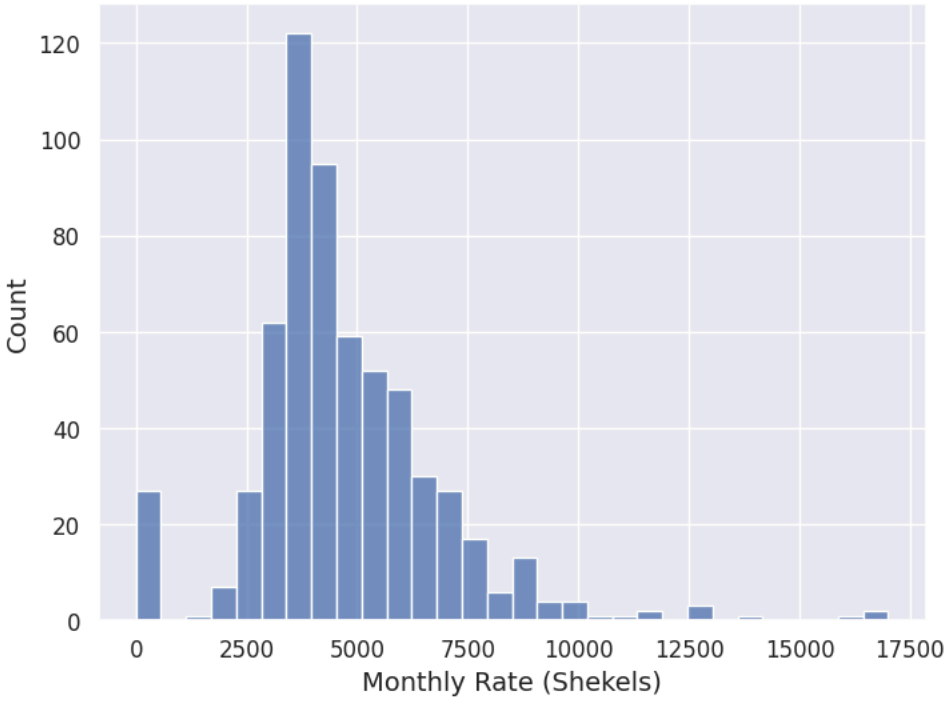
1. neighborhood: The hebrew name of the neighborhood in jerusalem where the listing is located
2. monthlyRate: The monthly rate (שכר דירה) in shekels
3. rooms: The number of rooms in the apartment
4. floor: The floor in which the apartment is located
5. area: The area of the apartment in squared meters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

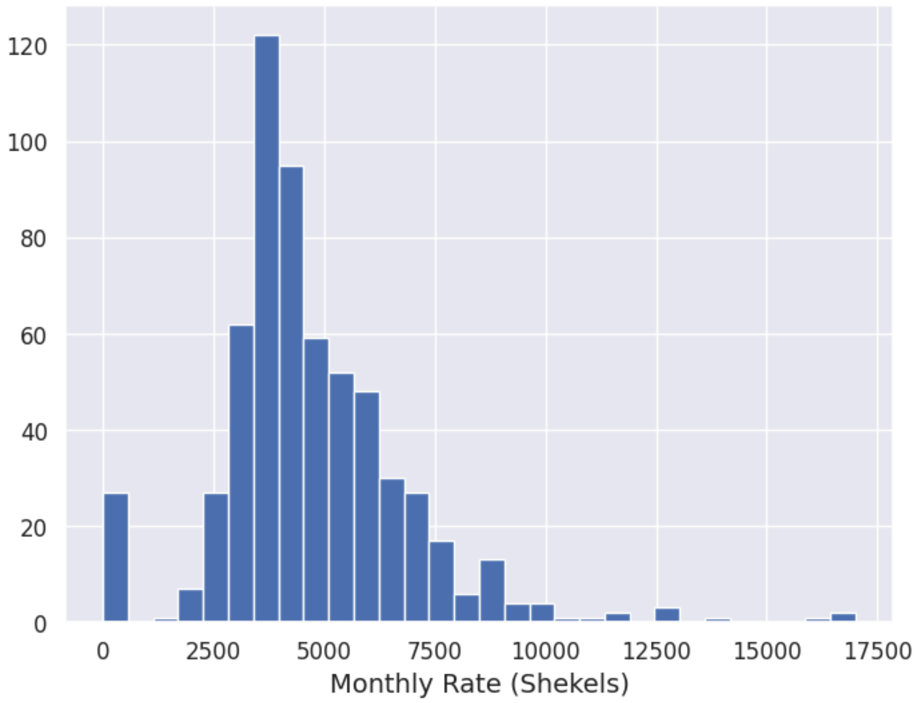
## Solution 1

```
# @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].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 | description | numFloors | reason |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **20** | 3983978 | קריית משה | 0.0 | private | 4.0 | 3.0 | 100.0 | 10/08/2022 | דירת 4 חדרים - במצב מצויין - שמורה ביותר כולל ... | 5.0 | monthlyRate <= 0 |
| **21** | 3985184 | נווה יעקב | 0.0 | private | 4.0 | 1.0 | 68.0 | 10/08/2022 | דירה במצב שמור מאד. ממוזגת, 2 חדרי שירותים (אח... | 2.0 | monthlyRate <= 0 |
| **22** | 3952750 | מוסררה | 0.0 | private | 5.5 | 1.0 | 180.0 | 10/08/2022 | להשכרה דירה מפוארת, במרכז העיר מרחק | 4.0 | monthlyRate |

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

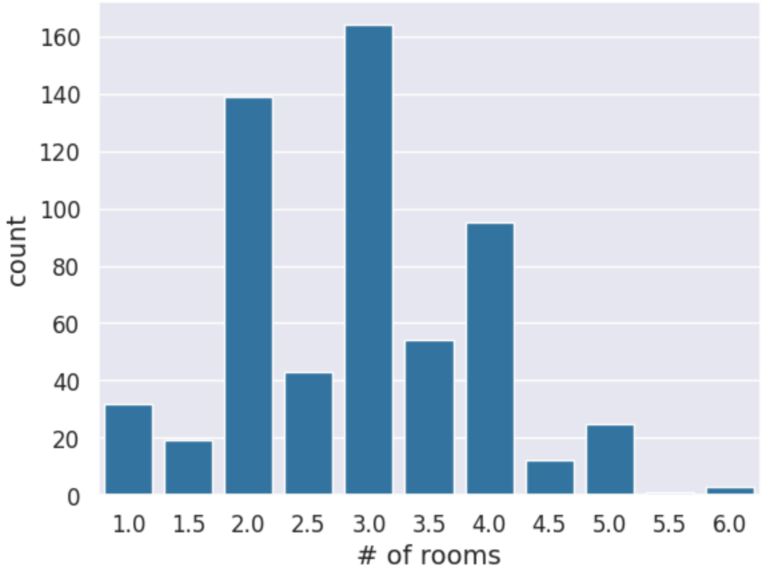
```
clean_df = rent_df[rent_df['monthlyRate'] > 0].reset_index(drop=True)
```

**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

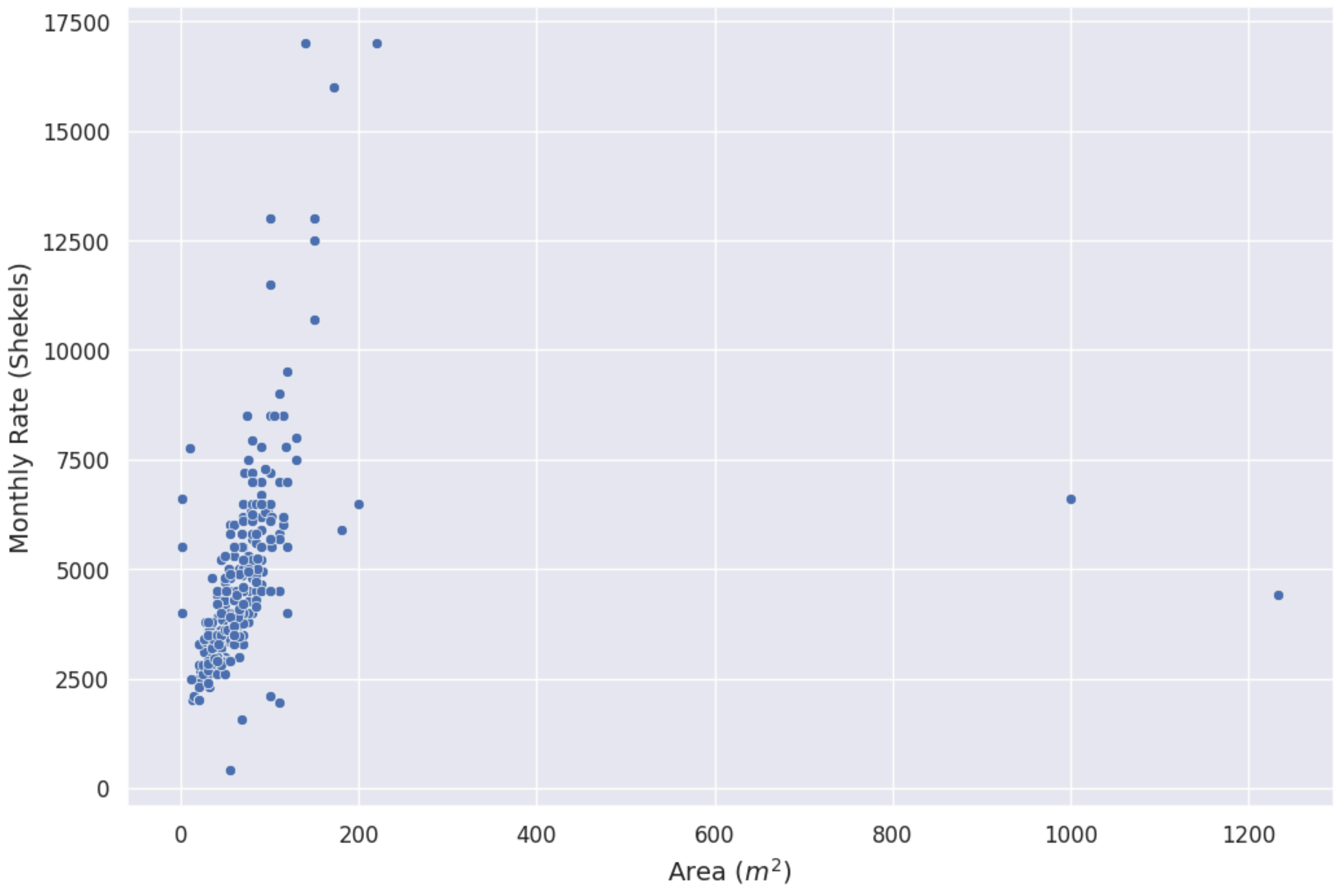
```
# @title Solution
if clean_df is None:
  print("Can't run until 'clean_df' is created!")
else:
  sns.countplot(x='rooms', data=clean_df, color='tab:blue')
  plt.xlabel("# of rooms");
```



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 ~1000$m^2$.

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 | description | numFloors |
|---|---|---|---|---|---|---|---|---|---|---|
| **185** | 3964340 | תלפיות | 4400.0 | private | 2.0 | 2.0 | 1234.0 | 10/08/2022 | NaN | 3.0 |
| **543** | 3956561 | זכרון משה | 6600.0 | private | 3.5 | 3.0 | 1000.0 | 01/07/2022 | דירה מהממת בלב ירושלים. צמודה לרכבת הקלה- תחנת... | 3.0 |
| **576** | 3974914 | תלפיות | 17000.0 | private | 5.0 | 3.0 | 220.0 | 10/08/2022 | דירת 5 חדרים ענקית ומהממת, בבנין בוטיק ויחודי ... | 4.0 |

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 ישיבה. מתאימה מאוד ל- 3 שותפים'
```

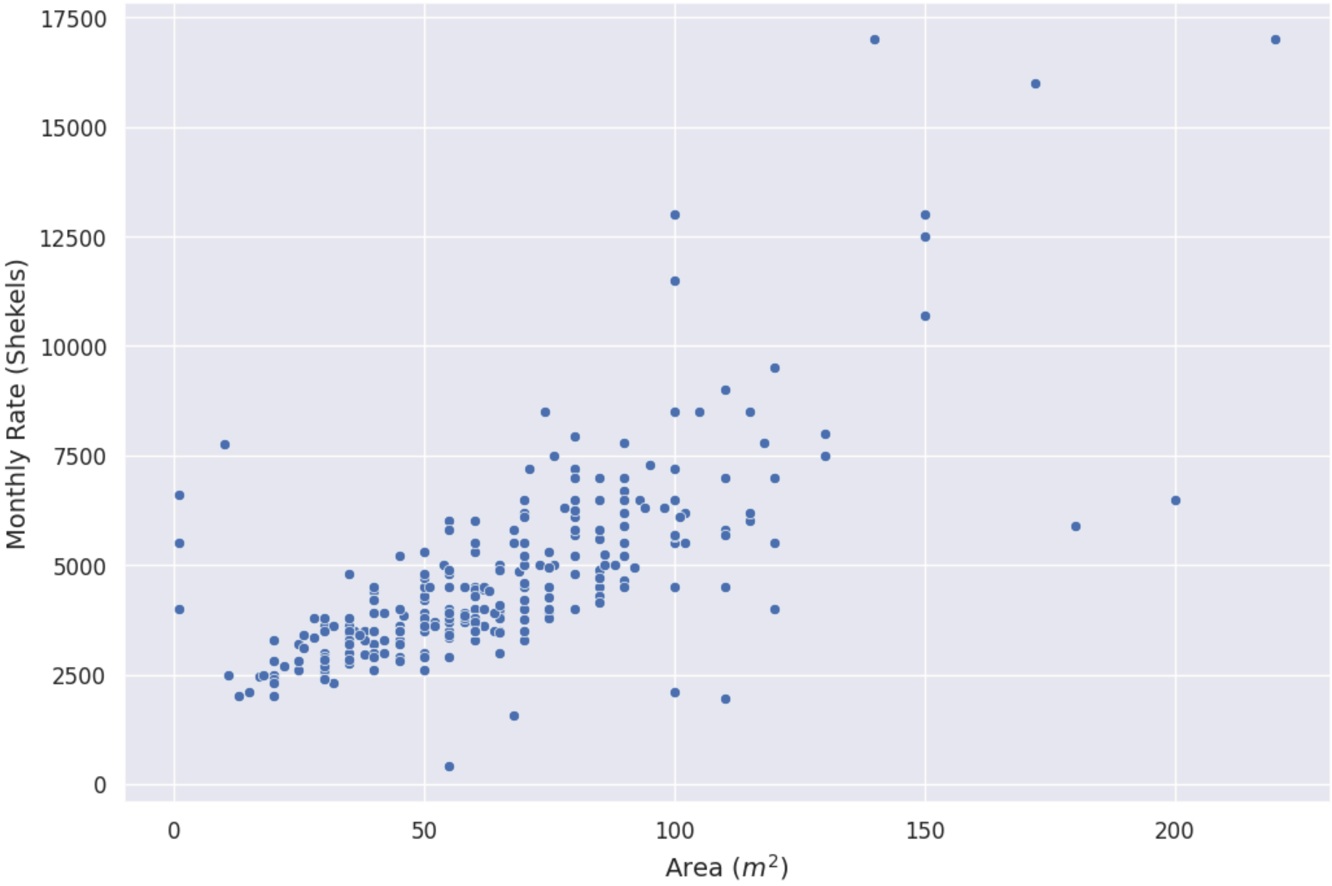
Clearly not a 1000 m^2 apartment...

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_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 | description | numFloors |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3994505 | קריית יובל | 2000.0 | private | 1.0 | 2.0 | 13.0 | 10/08/2022 | יחידת דיור להשכרה ברחוב הראשי של קריית יובל, ה... | 2.0 |
| **1** | 3981298 | רחביה | 2450.0 | private | 1.0 | 1.0 | 17.0 | 10/08/2022 | דירת יחיד 17 מטר כולל מרפסת קטנה | 3.0 |
| **3** | 3993997 | בית וגן | 2100.0 | private | 1.0 | 0.0 | 15.0 | 10/08/2022 | דירת חדר, כ-15 מ"ר, במיקום מרכזי אך שקט, משופצ... | 3.0 |
| **5** | 3993552 | הר נוף | 2000.0 | private | 1.0 | 0.0 | 20.0 | 10/08/2022 | יחידה משופצת ליחיד או למשרד , מיקום מצוין | 4.0 |
| **6** | 3972039 | גבעת שאול | 2700.0 | private | 1.0 | 0.0 | 22.0 | 10/08/2022 | דירת חדר כחדשה , כניסה נפרדת ללא וועד בית , מו... | 1.0 |
| **7** | 3988096 | המושבה הגרמנית | 2500.0 | private | 1.0 | 0.0 | 18.0 | 10/08/2022 | רלוונטי לנשים בלבד. ללא עישון. ללא חיות מחמד. ... | 1.0 |
| **8** | 3992809 | נחלאות | 3200.0 | private | 1.0 | 2.0 | 25.0 | 10/08/2022 | הדירה המגניבה בנחלאות, מתפנה אחרי תקופה ארוכה ... | 2.0 |
| **10** | 3983516 | הגבעה הצרפתית | 2000.0 | private | 1.0 | 2.0 | 20.0 | 10/08/2022 | דירת חדר קטנה, מסוגננת ונחמדה, מתאימה ליחיד בל... | 13.0 |
| **12** | 3987706 | נחלאות | 2800.0 | private | 1.0 | 0.0 | 20.0 | 10/08/2022 | להשכרה, דירה, בירושלים ברחוב חצור בנחלאות. דיר... | 2.0 |
| **13** | 3991842 | קטמון הישנה | 2500.0 | private | 1.0 | 1.0 | 20.0 | 10/08/2022 | דירת חדר ליחיד באזור יפייפה ושקט בקטמון הישנה.... | 4.0 |
| **14** | 3992479 | קריית יובל | 2400.0 | private | 1.0 | 1.0 | 20.0 | 10/08/2022 | דירת חדר חמודה עם גינה קטנה משותפת,מתאים ליחיד... | 1.0 |
| **15** | 3974372 | תל ארזה | 2500.0 | private | 1.0 | 0.0 | 11.0 | 10/08/2022 | להשכרה, דירת חדר , בירושלים | 4.0 |
| **19** | 3985106 | קטמונים | 2300.0 | private | 1.0 | 0.0 | 20.0 | 10/08/2022 | דירת חדר חמודה עם חצר במיקום מרכזי. כמה דק' ה... | 3.0 |
| **21** | 3985295 | מוסררה | 2600.0 | private | 1.5 | 0.0 | 25.0 | 10/08/2022 | אזור חרדי. מיקום מרכזי קרוב להכול. מרוהטת מלא | 4.0 |
| **27** | 3982008 | קריית יובל | 2800.0 | private | 1.5 | 0.0 | 25.0 | 10/08/2022 | הדירה נמצאת ברחוב שקט 7 דקות הליכה מעין כרם יש... | 3.0 |

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, 1000
```

| | propertyID | neighborhood | monthlyRate | mefarsem | rooms | floor | area | entry | description | numFloors |
|---|---|---|---|---|---|---|---|---|---|---|
| **197** | 3984483 | ארנונה | 6600.0 | private | 4.0 | 2.0 | 1.0 | 01/09/2022 | בשכונת ארנונה, רח' שלום יהודה, דירת 4 חדרים מש... | 4.0 |
| **234** | 3985019 | פסגת זאב | 5500.0 | private | 4.0 | 3.0 | 1.0 | 10/08/2022 | להשכרה 4 חדרים מרווחת עם מרפסת סוכה שטופת שמש ... | 4.0 |
| **235** | 3944204 | בית הכרם | 7750.0 | private | 4.0 | 3.0 | 10.0 | 01/09/2022 | בבניין חדש, מושקעת, מוארת, מאווררת רח' שקט, רח... | 5.0 |

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

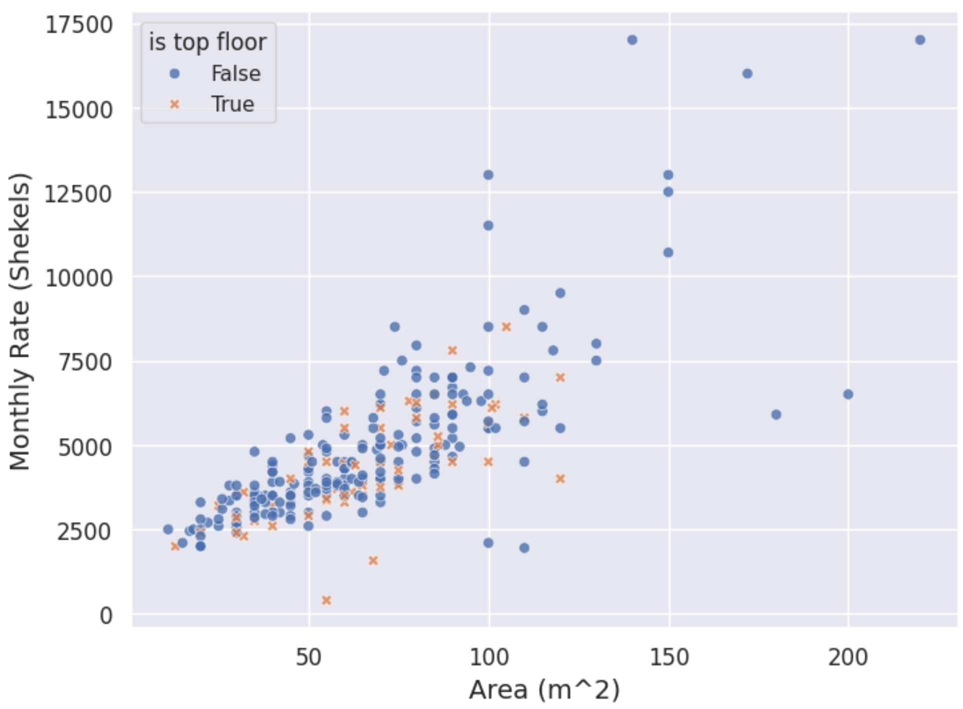
```
#remove the outliers
if clean_df_area_filtered is None:
  print("Can't run until 'clean_df_area_filtered' is created!")
elif outlier_df is None:
  print("Can't run until 'outlier_df' is created!")
else:
  non_ouliers = clean_df_area_filtered['area'] > 10 # get non outliers series of true/false

  # 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.)

## 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 | description | numFloors | is top floor |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **198** | 3956418 | רחביה | 13000.0 | agent | 4.0 | 1.0 | 100.0 | NaN | Beautifully renovated furnished authentic arab... | 3.0 | False |
| **236** | 3985051 | טלביה | 17000.0 | private | 4.0 | 4.0 | 140.0 | 10/08/2022 | בקינג דיוד רזדנס דירת 4 חדרים להשכרה ללא תיו... | 10.0 | False |
| **257** | 3982363 | רחביה | 16000.0 | private | 5.0 | 2.0 | 172.0 | 10/08/2022 | הזדמנות נדירה!!! ברחוב רד"ק, בנקודה הכי קרובה ... | 5.0 | False |
| **260** | 3980016 | אבו תור | 12500.0 | private | 5.0 | 0.0 | 150.0 | 10/08/2022 | דירת דופלקס ענקית 150 מ"ר עם כניסה נפרדת. | 4.0 | False |

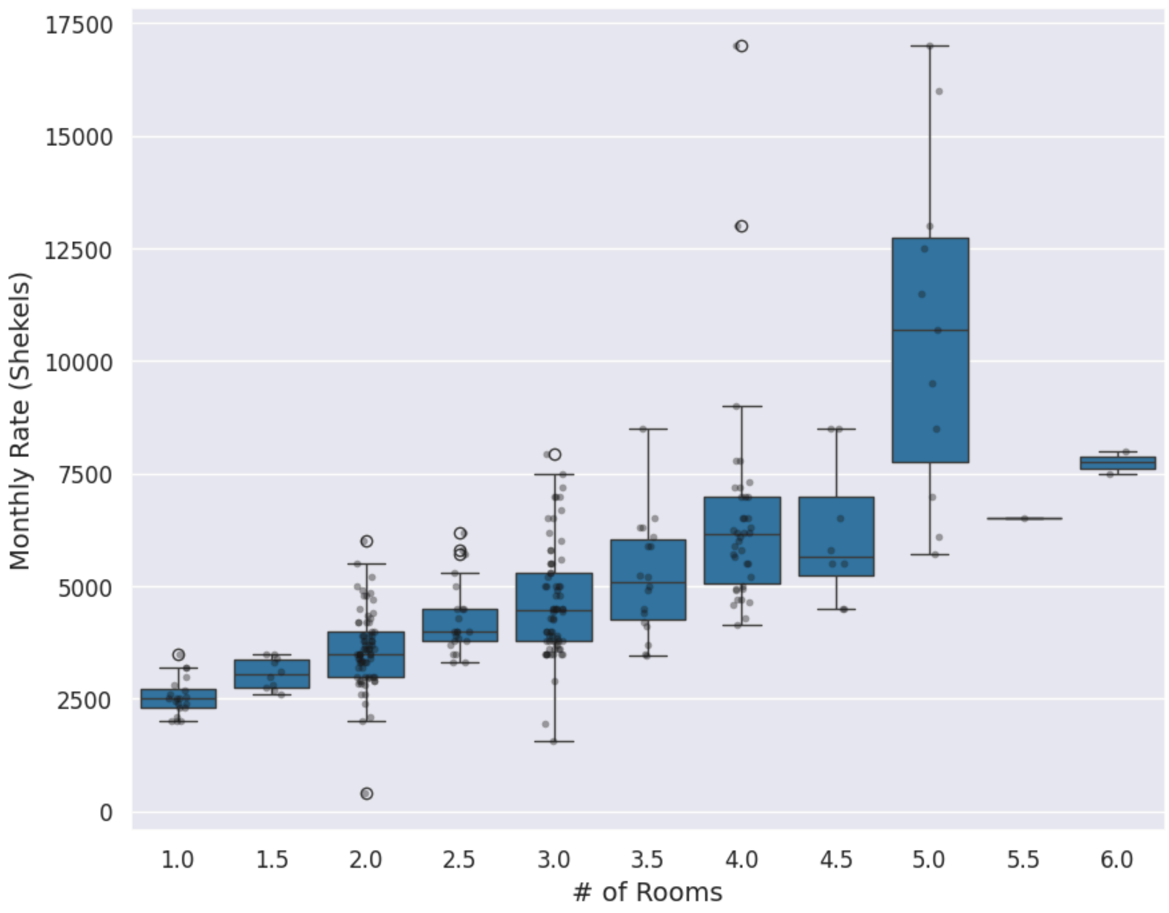
We can see some representation of the more expensive neighborhoods of Jerusalem here.. More on the neighborhoods later on!

**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 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        ...  area             numFloors
                   count  mean    std          min     25%     50%     75%     max      count  mean   ...  75%     max      count  mean  std
       reason
'area' <= 10       5.0    5870.0  1399.821417  4000.0  5500.0  5500.0  6600.0  7750.0   5.0    3.90   ...  1.0     10.0     5.0    3.80  1.095445
'area' >= 800      2.0    5500.0  1555.634919  4400.0  4950.0  5500.0  6050.0  6600.0   2.0    2.75   ...  1175.5  1234.0   2.0    3.00  0.000000
monthlyRate
    <= 0           25.0   0.0     0.000000     0.0     0.0     0.0     0.0     0.0      25.0   3.28   ...  99.0    4554.0   25.0   3.48  2.293469

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

Show code

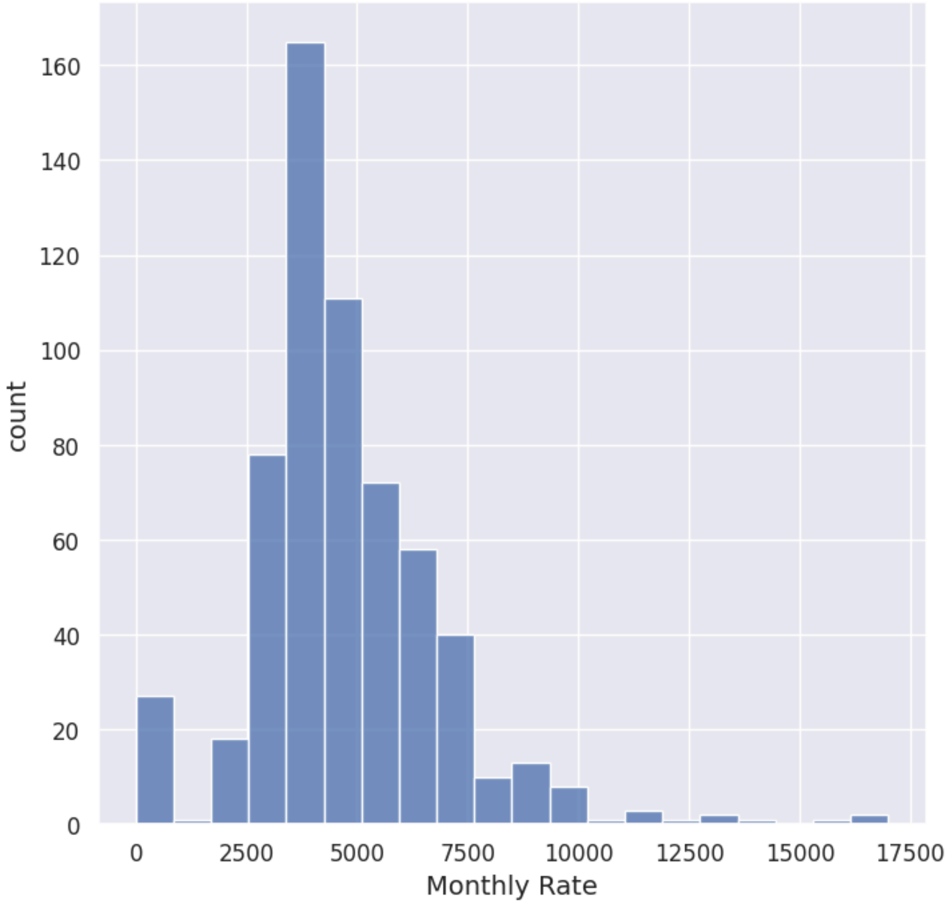
Let's go back to the distribution of monthly rental prices in the dataset. Are there interesting trends in the distribution that we missed in the visualizations before?

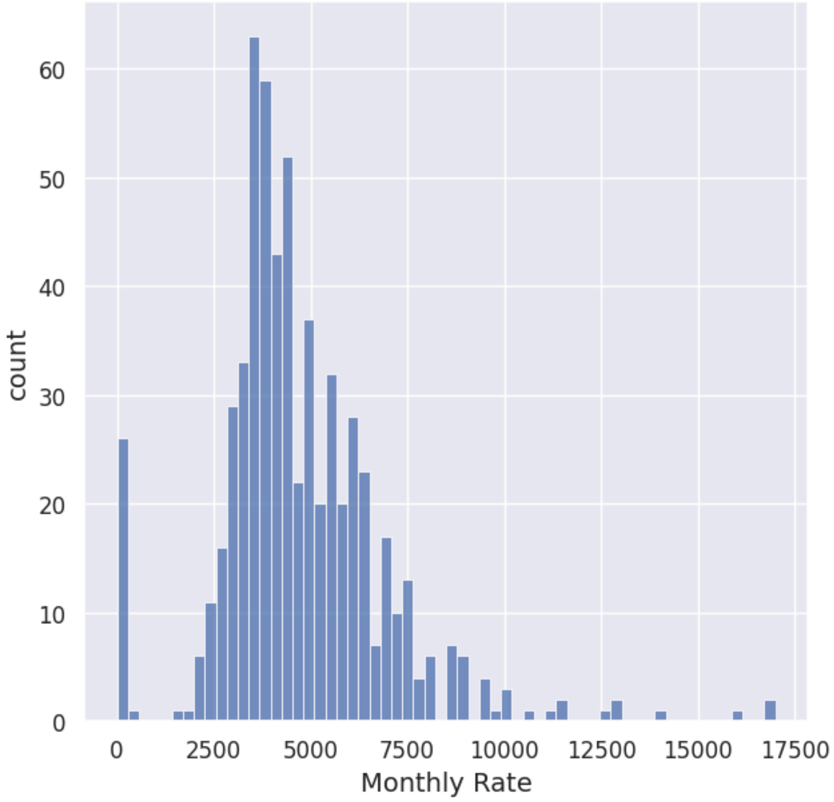
**Use only `part1_df` for the coding questions in this part**

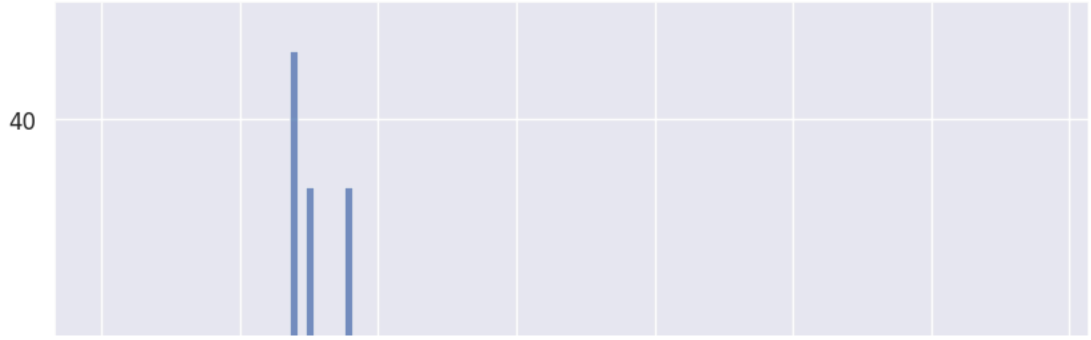
### Question 1

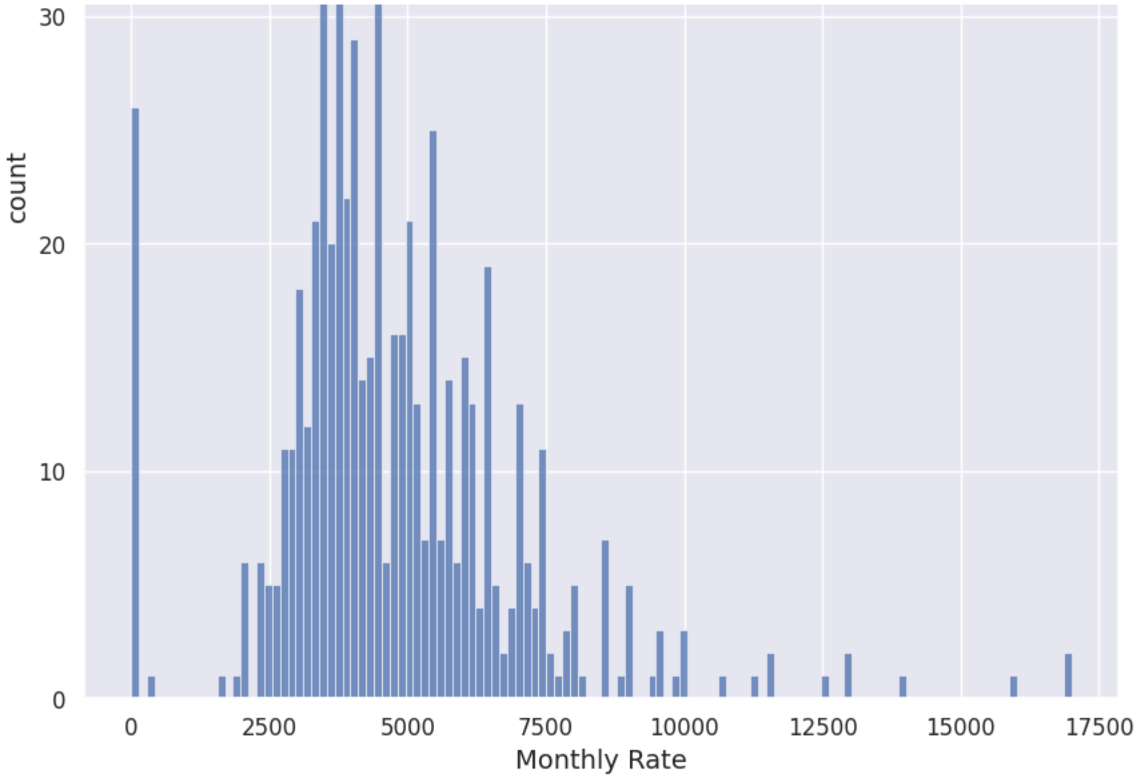
Plot 3 different histograms of the monthly prices with 20, 60 and 120 bins respectively, each in a different axis/figure.

```
# Part 1 - Question 1
# Your code goes here:
#hist1
fig = plt.figure(figsize=(8,8))
sns.histplot(x='monthlyRate', data=part1_df, bins=20)
plt.xlabel("Monthly Rate")
plt.ylabel("count");

#hist2
fig = plt.figure(figsize=(7,7))
sns.histplot(x='monthlyRate', data=part1_df, bins=60)
plt.xlabel("Monthly Rate")
plt.ylabel("count");

#hist3
fig = plt.figure(figsize=(10,10))
sns.histplot(x='monthlyRate', data=part1_df, bins=120)
plt.xlabel("Monthly Rate")
plt.ylabel("count");
```


count
0
20
40
60
80
100
120
140
160
0
2500
5000
7500
10000
12500
15000
17500
Monthly Rate


count
0
10
20
30
40
50
60
0
2500
5000
7500
10000
12500
15000
17500
Monthly Rate


40



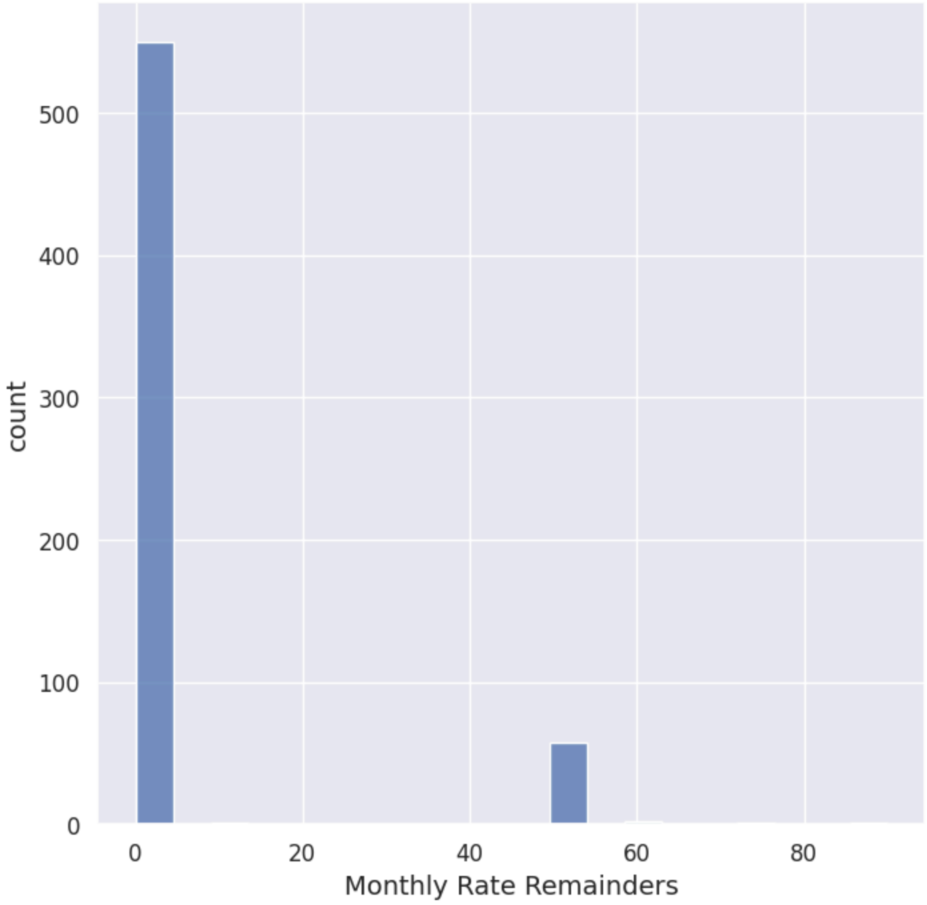
## Question 2

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:
fig = plt.figure(figsize=(8,8))
part1_df['remainders'] = part1_df['monthlyRate'] % 100
sns.histplot(x='remainders', data=part1_df, bins=20)
plt.xlabel("Monthly Rate Remainders")
plt.ylabel("count");
```



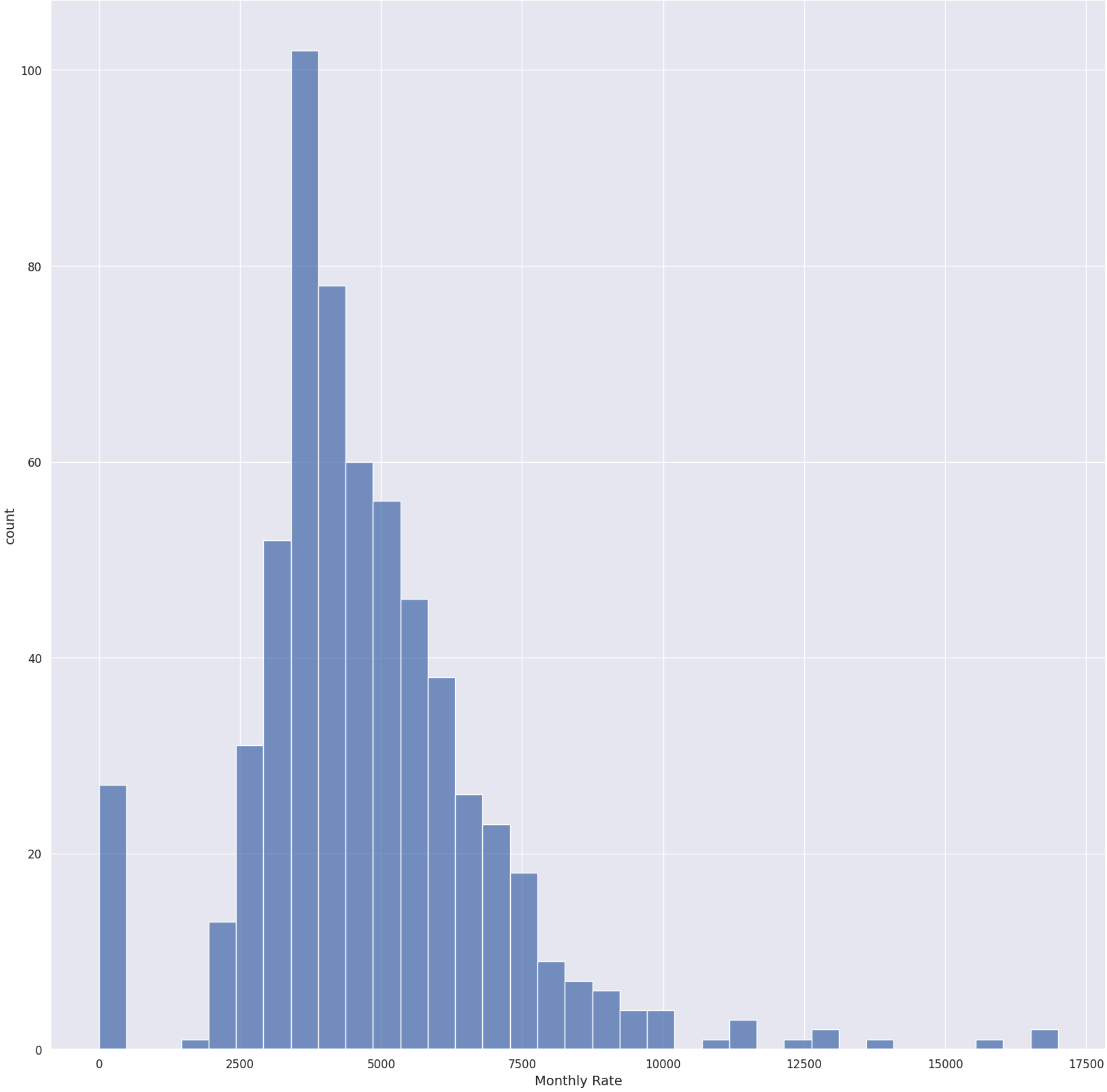
Part 1 Question 2 - textual Answer:

We created a new column in the data that presents the remainder after dividing the price of the apartment by 100. The assumption is that if most of the data has a remainder of zero, it means that the apartment price was likely rounded to the nearest hundred. Our results show this is indeed the case.

## 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:
fig = plt.figure(figsize=(20,20))
sns.histplot(x='monthlyRate', data=part1_df, bins = 35)
plt.xlabel("Monthly Rate")
plt.ylabel("count");
#part1_df["monthlyRate"].hist(bins=np.arange(0, part1_df["monthlyRate"].max(), 500))
```



Part 1 Question 3 - textual Answer:

``` This bin choice avoids an exaggerated rounding effect and allows us to see trend at a reasonable rate of increase of price (per 500 shekel).

We do see a drop at 5000 shekel, and another significant drop after 7500 and 10000, where it drops almost entirely, then only sporadic instances of higher values appear.

## 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)
```

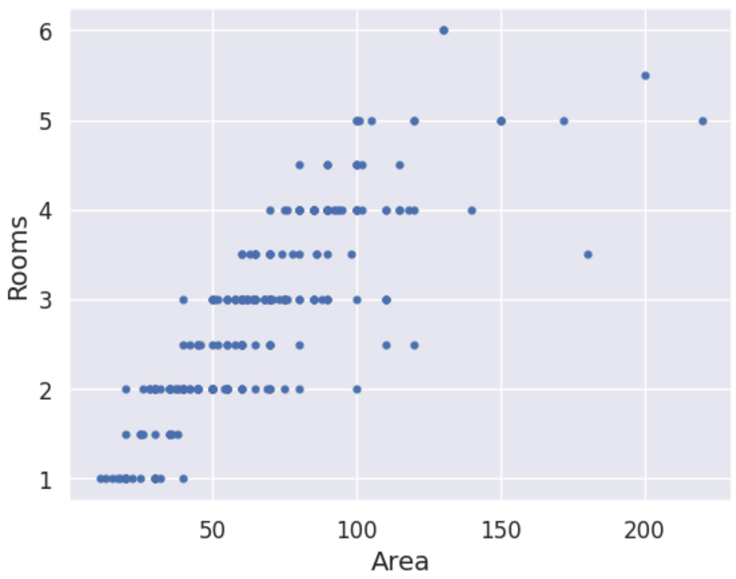
We saw that both the number of rooms and the area of an apartment are strongly associated with the monthly rate. We now want to check if those are just two perspectives of the same relation (how big is the apartment) or is there something more to it. We will use the cleaned dataframe for this exercise.

**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.scatter(part2_df['area'], part2_df['rooms'], s = 12)
plt.xlabel("Area")
plt.ylabel("Rooms");
```



Part 2 Question 1 - textual Answer:

Based on this scatter plot, it is easy to see a trend of more rooms as the size of the apartment increases.

### Question 2

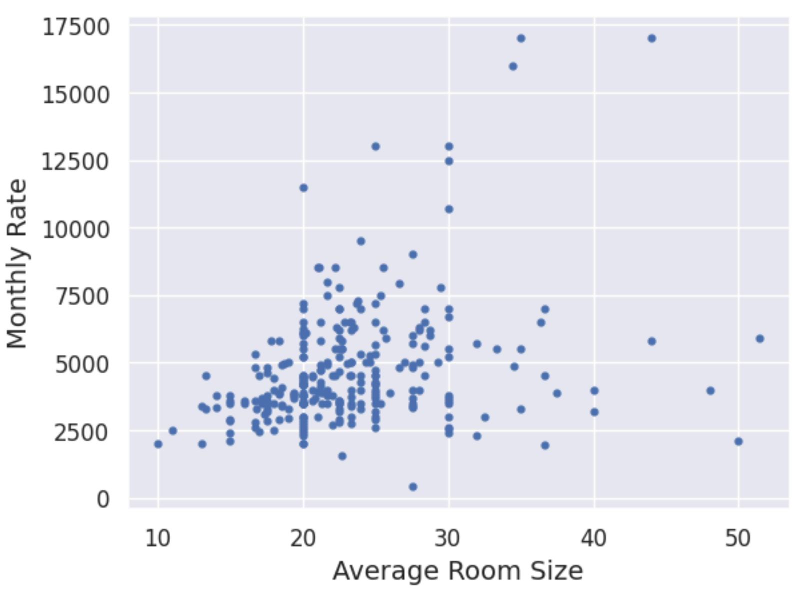
Add a new column to the dataframe named `"averageRoomSize"` with the average room size in the given listing.

```
# 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.scatter(part2_df['averageRoomSize'], part2_df['monthlyRate'], s = 12)
plt.xlabel("Average Room Size")
plt.ylabel("Monthly Rate");
```



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

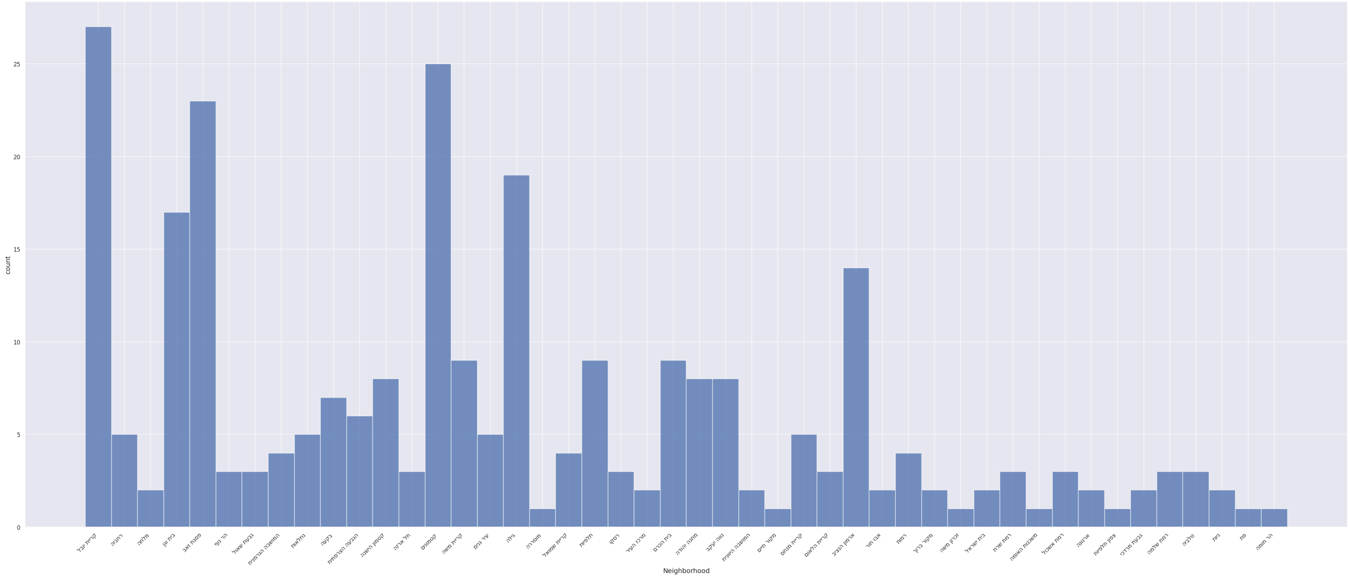
```
46
```

## 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:
fig = plt.figure(figsize=(50,20))
sns.histplot(x='neighborhood_flipped', data=part3_df)
plt.xlabel("Neighborhood")
plt.ylabel("count");
plt.xticks(rotation=45, ha='right')
```

```
([0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  39,
  40,
  41,
  42,
  43,
  44,
  45],
 [Text(0, 0, 'לבוי תיירק'),
  Text(1, 0, 'היבחר'),
  Text(2, 0, 'החלמ'),
  Text(3, 0, 'ןגו תיב'),
  Text(4, 0, 'באז תגספ'),
  Text(5, 0, 'ףונ רה'),
  Text(6, 0, 'לואש תעבג'),
  Text(7, 0, 'תינמרגה הבשומה'),
  Text(8, 0, 'תואלחנ'),
  Text(9, 0, 'העקב'),
  Text(10, 0, 'תיתפרצה העבגה'),
  Text(11, 0, 'הנשיה ןומטק'),
  Text(12, 0, 'הזרא לת'),
  Text(13, 0, 'םינומטק'),
  Text(14, 0, 'השמ תיירק'),
  Text(15, 0, 'םינג ריע'),
  Text(16, 0, 'הליג'),
  Text(17, 0, 'הררסומ'),
  Text(18, 0, 'לאומש תיירק'),
  Text(19, 0, 'תויפלת'),
  Text(20, 0, 'וקסר'),
  Text(21, 0, 'ריעה זכרמ'),
  Text(22, 0, 'םרכה תיב'),
  Text(23, 0, 'הדוהי הנחמ'),
  Text(24, 0, 'בקעי הוונ'),
  Text(25, 0, 'תינוויה הבשומה'),
  Text(26, 0, 'םייח רוקמ'),
  Text(27, 0, 'םחנמ תיירק'),
  Text(28, 0, 'םואלה תיירק'),
  Text(29, 0, 'ביצנה ןומרא'),
  Text(30, 0, 'רות ובא'),
  Text(31, 0, 'תומר'),
  Text(32, 0, 'ךורב רוקמ'),
  Text(33, 0, 'השמ ןורכז'),
  Text(34, 0, 'לארשי תיב'),
  Text(35, 0, 'תרש תמר'),
  Text(36, 0, 'המואה תונכשמ'),
  Text(37, 0, 'לוכשא תמר'),
  Text(38, 0, 'הנונרא'),
  Text(39, 0, 'תויפלת ןופצ'),
  Text(40, 0, 'יכדרמ תעבג'),
  Text(41, 0, 'המלש תמר'),
  Text(42, 0, 'היבלט'),
  Text(43, 0, 'תוינ'),
       Text(44, 0, 'תפ'),
       Text(45, 0, 'המוח רה')])
```



## 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 = 0
listing = 0
counts = part3_df.groupby('neighborhood')['monthlyRate'].count()
for amount in counts:
  if amount < 5:
    neighborhood += 1
    listing += amount
print("Number of neighborhoods with less than 5 listings: ", neighborhood)
print( "Fraction of listings with less than 5 listings out of total listings: ", listing/sum(counts))
print("Fraction of listings from top 8 neighborhoods:", sum(counts.nlargest(8))/sum(counts))
```

```
Number of neighborhoods with less than 5 listings:  28
Fraction of listings with less than 5 listings out of total listings:  0.23443223443223443
Fraction of listings from top 8 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:

counts = part3_df.groupby('neighborhood')['monthlyRate'].count().nlargest(8).sort_values(ascending = False).index
filtered_listings = part3_df[part3_df['neighborhood'].isin(counts)]
```

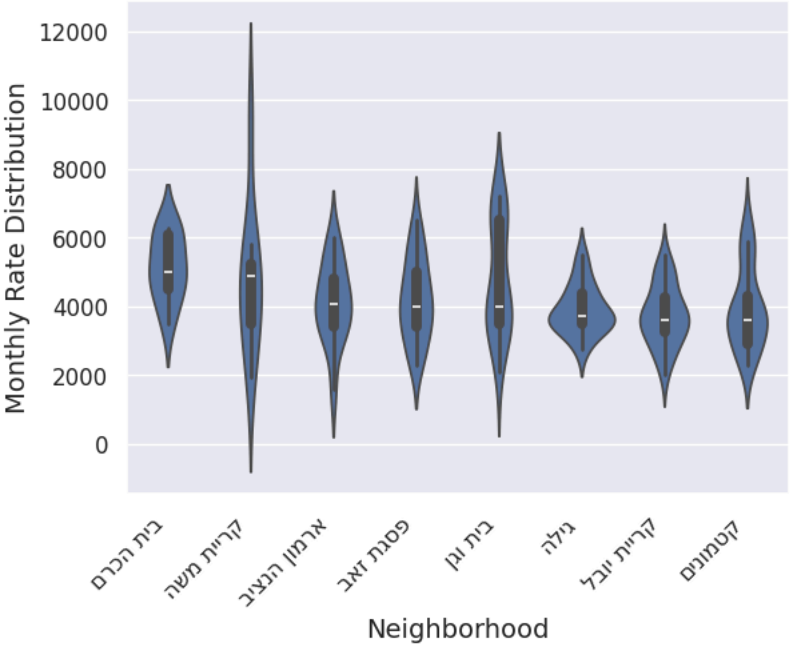
## Question 5

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:
sns.violinplot(x='neighborhood_flipped', y='monthlyRate', data=filtered_listings, order = filtered_listings.groupby('neighborhood_flipped
plt.xlabel("Neighborhood")
plt.ylabel("Monthly Rate Distribution")
plt.xticks(rotation=45, ha='right')
```

```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'םרכה תיב'),
  Text(1, 0, 'השמ תיירק'),
  Text(2, 0, 'ביצנה ןומרא'),
  Text(3, 0, 'באז תגספ'),
  Text(4, 0, 'ןגו תיב'),
  Text(5, 0, 'הליג'),
  Text(6, 0, 'לבוי תיירק'),
  Text(7, 0, 'םינומטק')])
```



Part 3 Question 5 - textual Answer:

A violin plot allows us to easily compare the distributions of neighborhood prices to each other, centered around the median. It is clear that Beit Hakerem has the largest distribution, and the others can be easily visualized and compared as well.

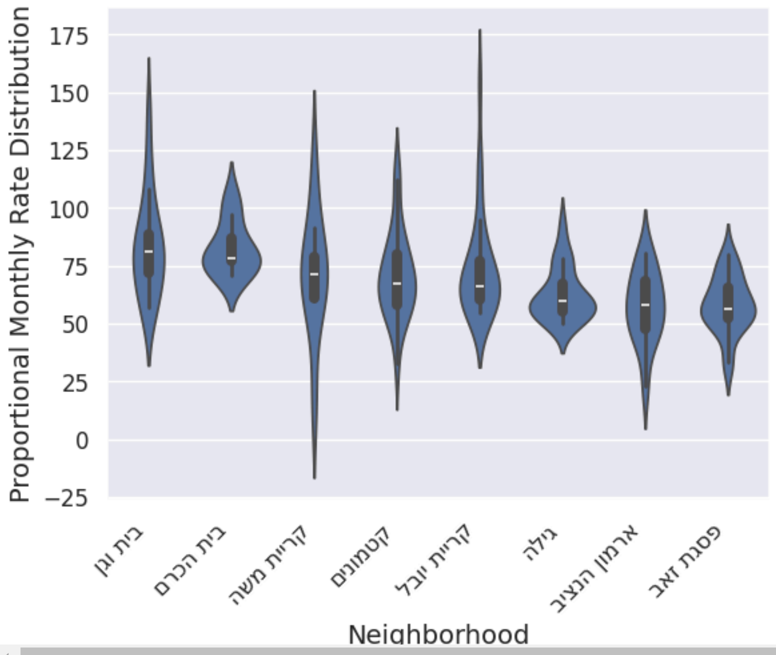
## Question 6

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:
filtered_listings['proportional_price'] = filtered_listings['monthlyRate'] / filtered_listings['area']

sns.violinplot(x='neighborhood_flipped', y='proportional_price', data=filtered_listings, order = filtered_listings.groupby('neighborhoo
plt.xlabel("Neighborhood")
plt.ylabel("Proportional Monthly Rate Distribution")
plt.xticks(rotation=45, ha='right')
```

```
<ipython-input-171-0584c03434bf>:3: 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-
  filtered_listings['proportional_price'] = filtered_listings['monthlyRate'] / filtered_listings['area']
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'ןגו תיב'),
  Text(1, 0, 'םרכה תיב'),
  Text(2, 0, 'השמ תיירק'),
  Text(3, 0, 'םינומטק'),
  Text(4, 0, 'לבוי תיירק'),
  Text(5, 0, 'הליג'),
  Text(6, 0, 'ביצנה ןומרא'),
  Text(7, 0, 'באז תגספ')])
```



## Part 3 Question 6 - textual Answer:

By dividing the monthly rate by the apartment area, we created a new value that shows the monthly cost of the apartment proportional to its area, to remove its influece on our results. With this new information, the same comparison shows that Bait Vegan has the highest median rent per square meter.

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:
#
#
```

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)
```

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'] < 2.0 ) & (part4_df['numFloors']  == 1.0)
print(len(part4_df[part4_df['is_a_house'] == True]))

np.random.seed(0)
print(part4_df[part4_df['is_a_house'] == True].sample(3)['description'])
```

```
17
6     דירת חדר כחדשה , כניסה נפרדת ללא וועד בית , מו...
34    בלב נחלאות, דירה מתוקה להשכרה עם כניסה פרטית. ...
48    מקום מדהים, קומה ראשונה,תחנת אוטובוס,גני ילדים...
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:
sns.boxplot(x='is_a_house', y='monthlyRate', data=part4_df, meanline=True, showmeans=True)
plt.xlabel("Is it a house?")
plt.ylabel("Monthly Rate")
```