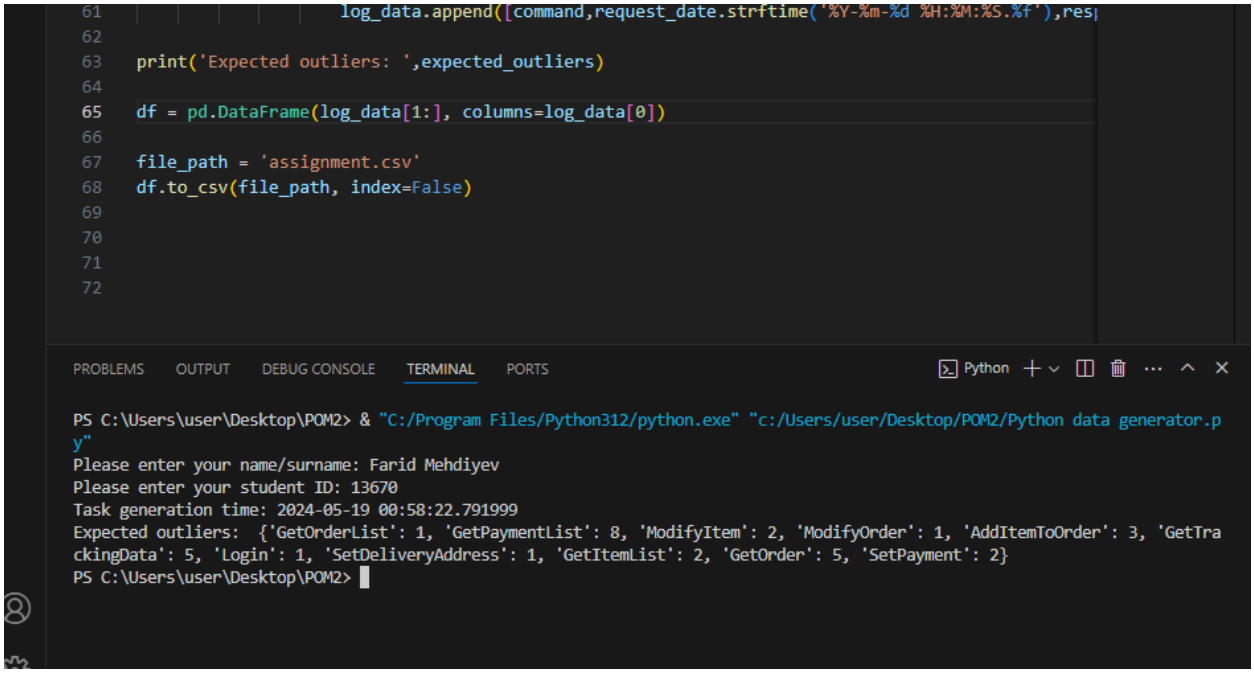
Farid Mehdiyev

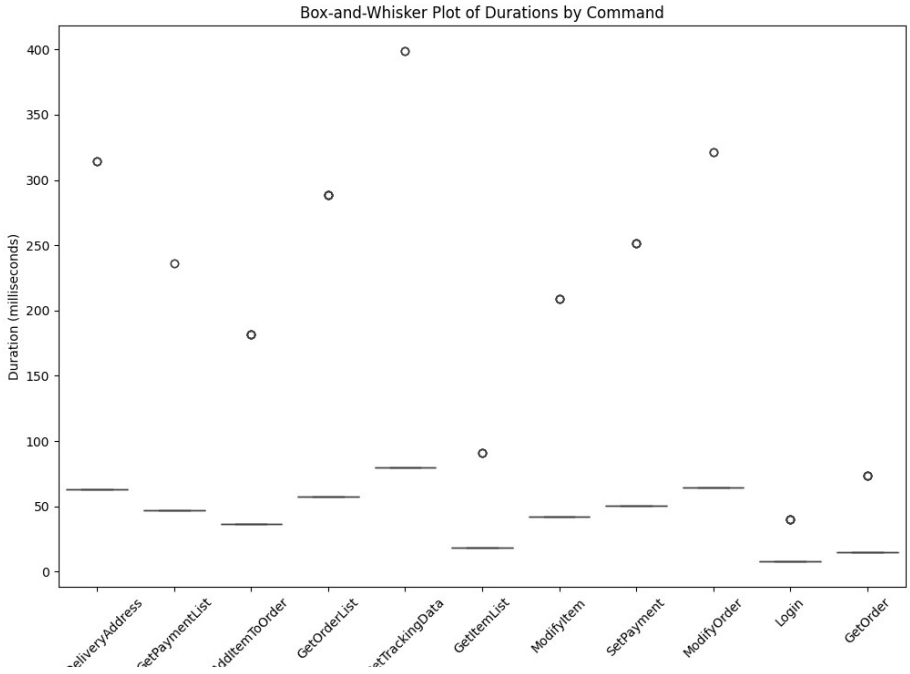
Statistical Data Analysis

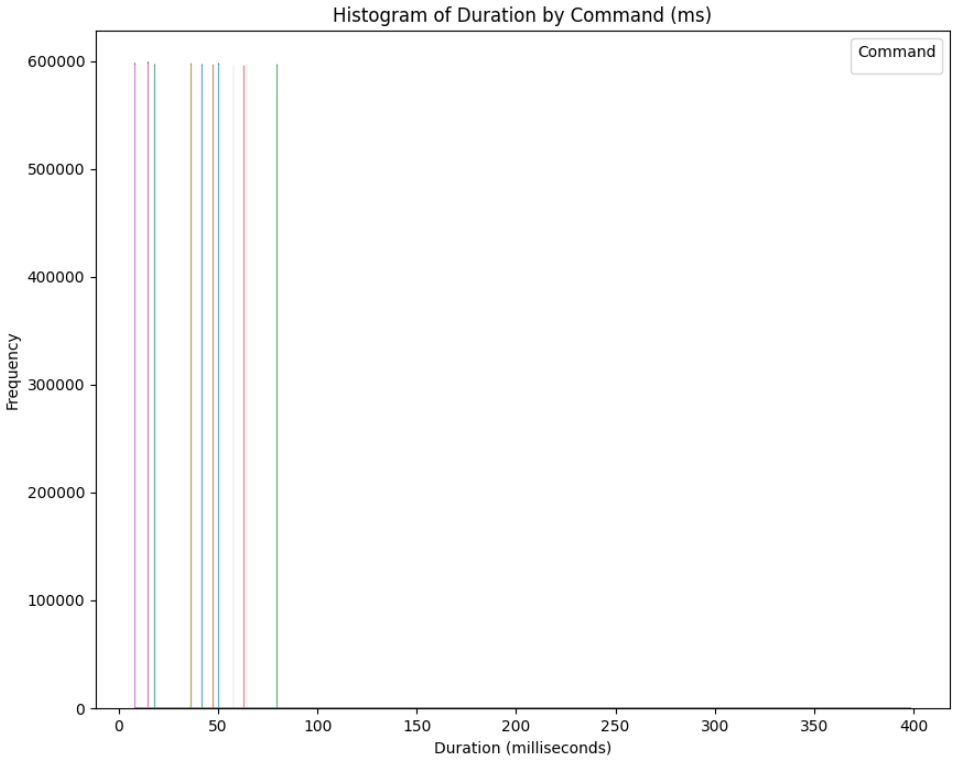
1. **First of all, we have to generate CSV file that is mentioned in code. Lets do it:**



First part is done, I will upload the “assignement.csv” file to OneDrive

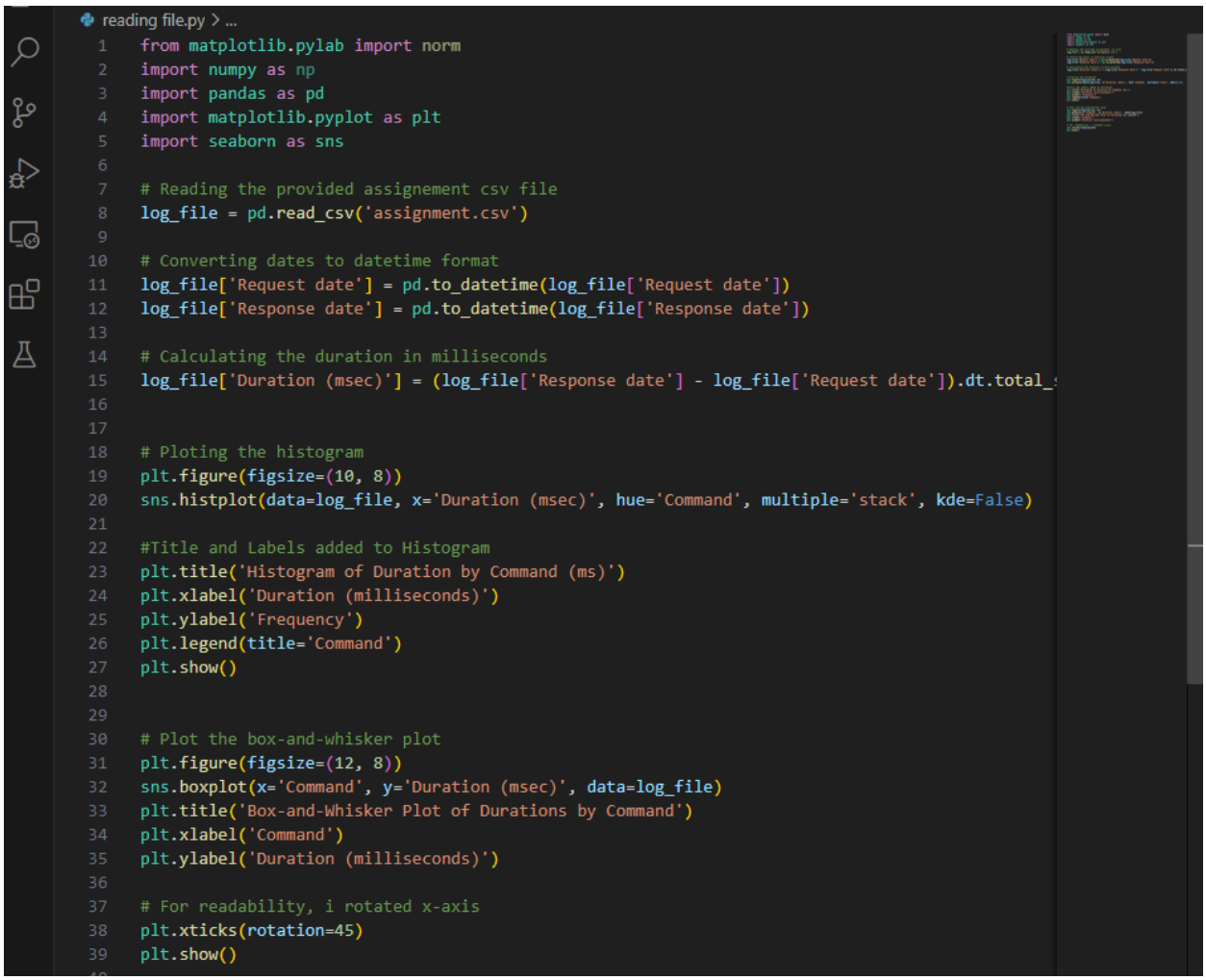
2. Visualize the distribution and range of values for each command type (histogram and box and whisker diagram):





Commands are in order, as a

“GetDeliveryAddress,
GetPaymentList,
AddItemtoOrde,
GetTrackingData,
GetItemList,
ModifyItem,
SetPayment,
ModifyOrder,
Login and GetOrder”



```python
from matplotlib.pylab import norm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Reading the provided assignement csv file
log_file = pd.read_csv('assignment.csv')

# Converting dates to datetime format
log_file['Request date'] = pd.to_datetime(log_file['Request date'])
log_file['Response date'] = pd.to_datetime(log_file['Response date'])

# Calculating the duration in milliseconds
log_file['Duration (msec)'] = (log_file['Response date'] - log_file['Request date']).dt.total_

# Ploting the histogram
plt.figure(figsize=(10, 8))
sns.histplot(data=log_file, x='Duration (msec)', hue='Command', multiple='stack', kde=False)

#Title and Labels added to Histogram
plt.title('Histogram of Duration by Command (ms)')
plt.xlabel('Duration (milliseconds)')
plt.ylabel('Frequency')
plt.legend(title='Command')
plt.show()


# Plot the box-and-whisker plot
plt.figure(figsize=(12, 8))
sns.boxplot(x='Command', y='Duration (msec)', data=log_file)
plt.title('Box-and-Whisker Plot of Durations by Command')
plt.xlabel('Command')
plt.ylabel('Duration (milliseconds)')

# For readability, i rotated x-axis
plt.xticks(rotation=45)
plt.show()
```

Provided Python code generating both a histogram and box-and-whisker plot for visualize the distribution of durations by commands from a dataset that is provided as a “assignement.csv” file. Duration in milliseconds is calculated by the finding difference between “Response date” and “Request date” columns and that is converted the results to the millisecond.

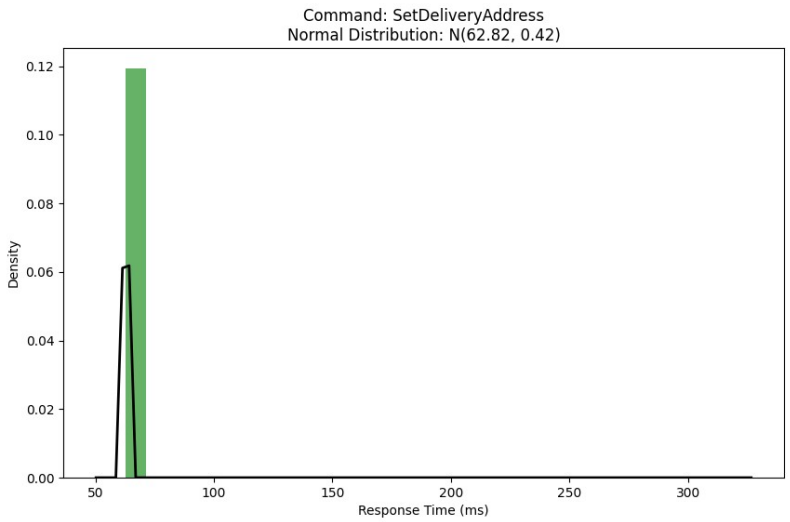
Histogram - for visualization, I have used Matplotlib and Seaborn Python libraries and they are very useful. Histogram is created using Seaborn’s ‘histplot’ function. Customization of the histogram is x and y axis labels for enhancing readability.

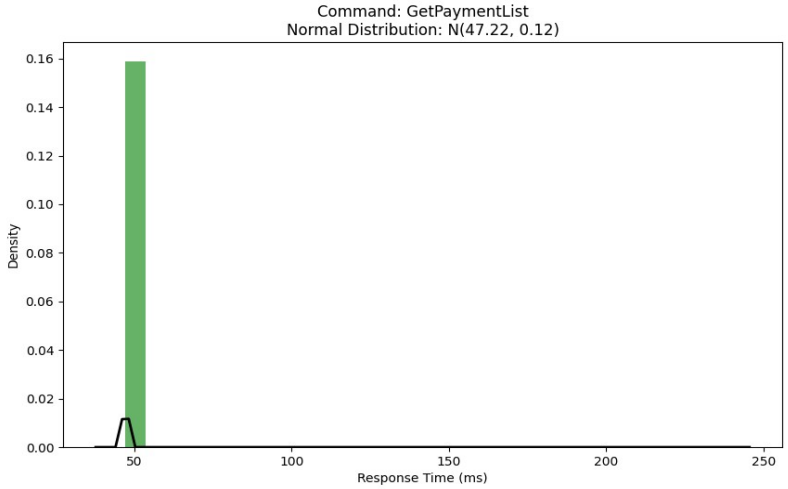
Box-and-Whisker Plot – it is the same as the histogram as you can see that – same data reading and processing log data. I have used ‘boxplot’ function for box-and-whisker plot and customization is same as histogram.

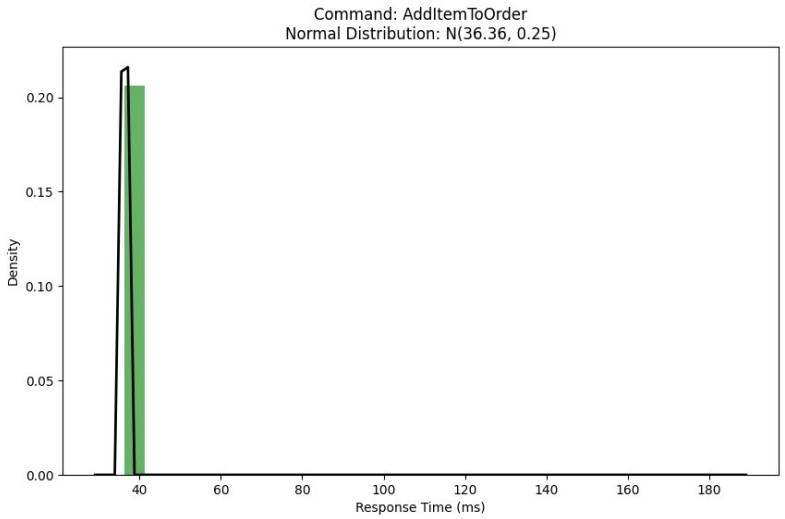
3. Show the distribution of each command as Normal distribution: N(m, variation).

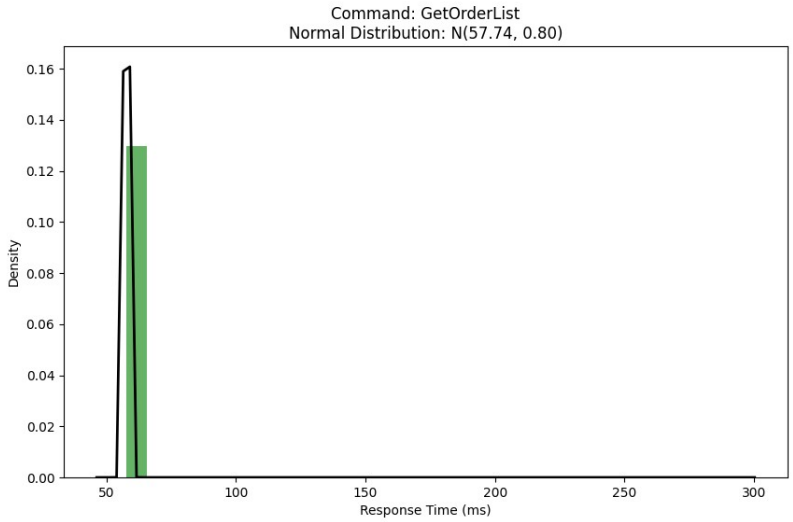
As we can observe that, code is used Pandas to read the CSV file into a DataFrame name **'df'**. For the statistical analysis by the calculating mean and variance of response times for each unique command in the dataset and likewise it iterates unique commands and subsets the *DataFrame* for calculating statistics for each command by separately. '**mean()**' and **'var()**' functions are used for finding mean and variance. For the visualization part I have used SciPy's statistical functions.

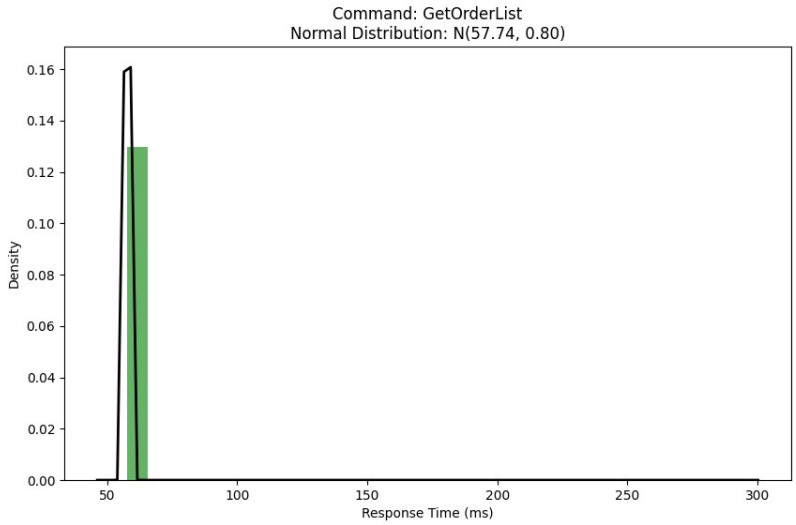
```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Loading the generated CSV file that is provided
df = pd.read_csv('assignment.csv')

# Converting datetime strings to datetime objects
df['Request date'] = pd.to_datetime(df['Request date'])
df['Response date'] = pd.to_datetime(df['Response date'])

# Calculating response times in milliseconds
df['Response time'] = (df['Response date'] - df['Request date']).dt.total_seconds() * 1000

# Initializing a dictionary to store mean and variance for each command
command_stats = {}

# Calculating mean and variance for each command
for command in df['Command'].unique():
    response_times = df[df['Command'] == command]['Response time']
    mean = response_times.mean()
    variance = response_times.var()
    command_stats[command] = (mean, variance)

    # Plot the distribution of response times
    plt.figure(figsize=(10, 6))
    plt.hist(response_times, bins=30, density=True, alpha=0.6, color='g')

    # Plot the normal distribution with calculated mean and variance
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = stats.norm.pdf(x, mean, np.sqrt(variance))
    plt.plot(x, p, 'k', linewidth=2)
    plt.title(f'Command: {command}\nNormal Distribution: N({mean:.2f}, {variance:.2f})')
    plt.xlabel('Response Time (ms)')
    plt.ylabel('Density')
    plt.show()

```

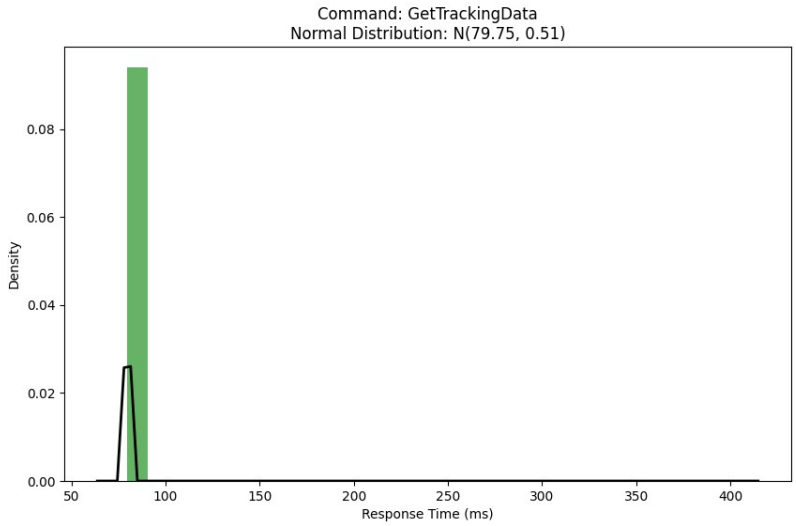

Command: SetDeliveryAddress
Normal Distribution: N(62.82, 0.42)
Density
Response Time (ms)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
50
100
150
200
250
300

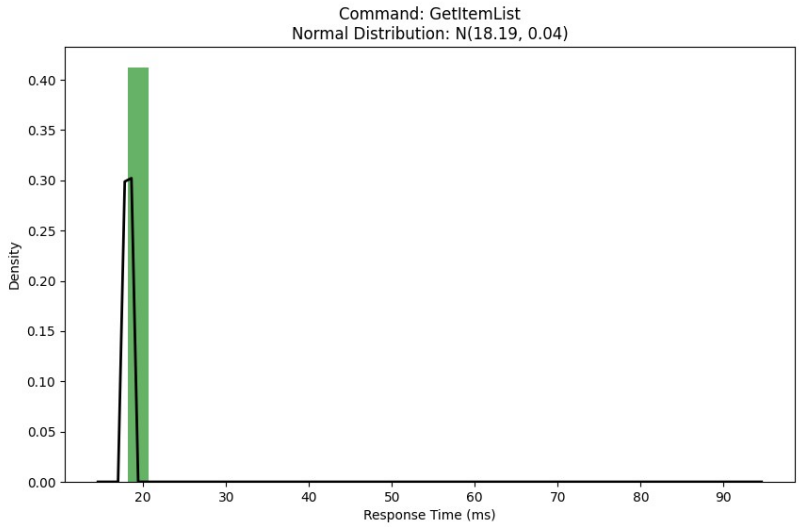

Command: GetPaymentList
Normal Distribution: N(47.22, 0.12)
Density
Response Time (ms)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
50
100
150
200
250

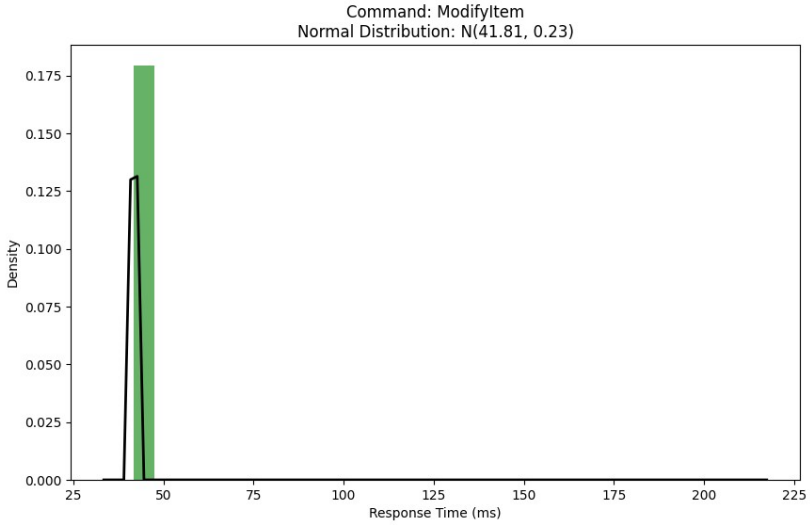

Command: AddItemToOrder
Normal Distribution: N(36.36, 0.25)
Density
Response Time (ms)
0.00
0.05
0.10
0.15
0.20
40
60
80
100
120
140
160
180

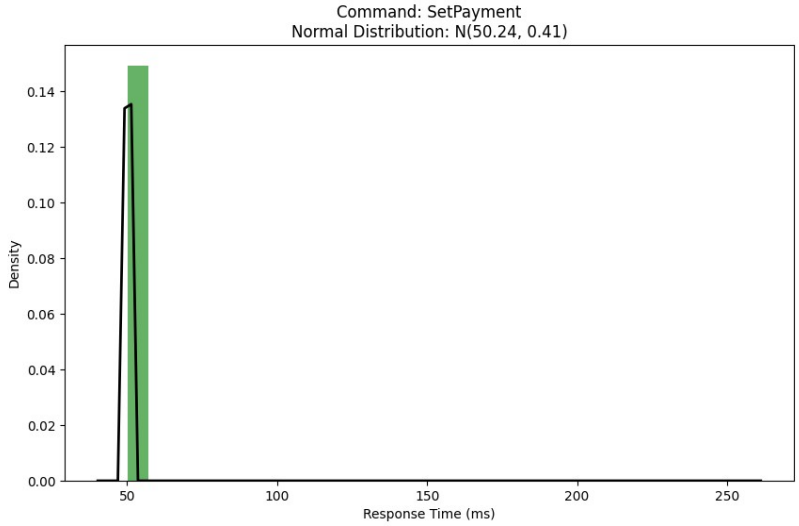

Command: GetOrderList
Normal Distribution: N(57.74, 0.80)
Density
Response Time (ms)

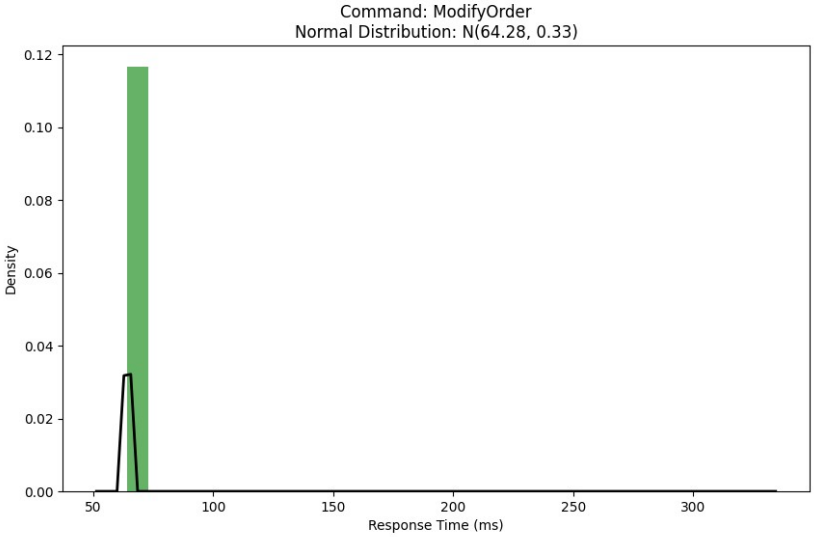

Command: GetOrderList
Normal Distribution: N(57.74, 0.80)
Density
Response Time (ms)

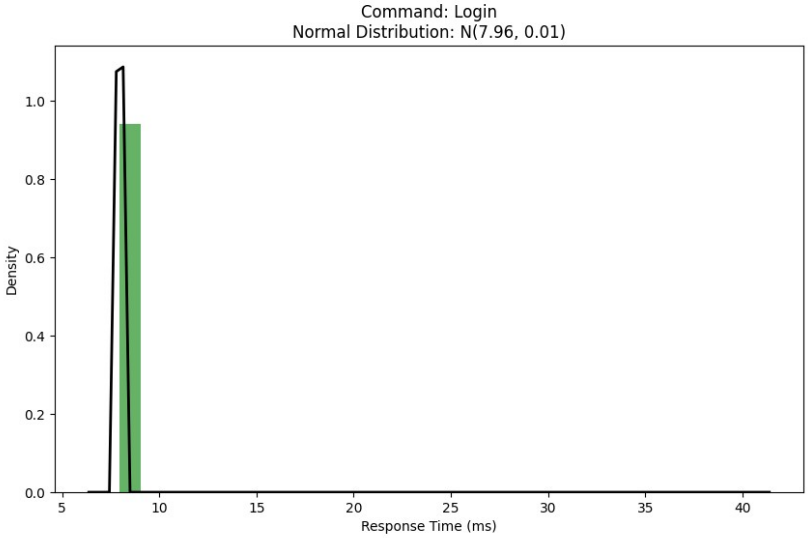

Command: GetTrackingData
Normal Distribution: N(79.75, 0.51)
Density
Response Time (ms)

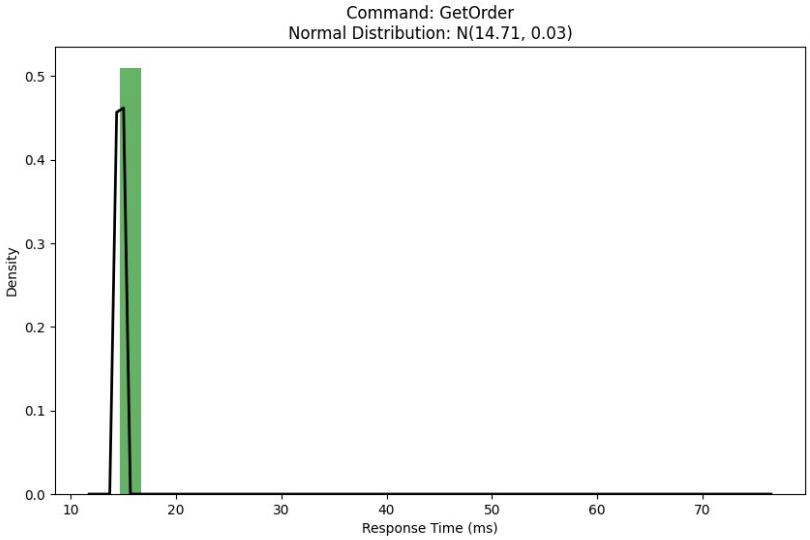

Command: GetItemList
Normal Distribution: N(18.19, 0.04)
Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
20
30
40
50
60
70
80
90
Response Time (ms)


Command: ModifyItem
Normal Distribution: N(41.81, 0.23)
Density
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
25
50
75
100
125
150
175
200
225
Response Time (ms)


Command: SetPayment
Normal Distribution: N(50.24, 0.41)
Density
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
50
100
150
200
250
Response Time (ms)

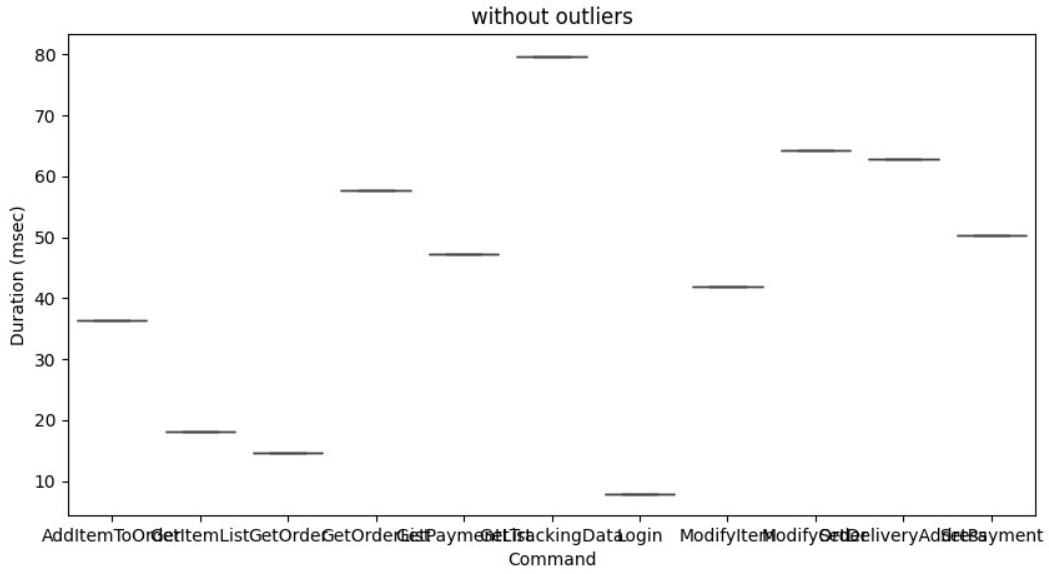

Command: ModifyOrder
Normal Distribution: N(64.28, 0.33)
Density
Response Time (ms)


Command: Login
Normal Distribution: N(7.96, 0.01)
Density
Response Time (ms)

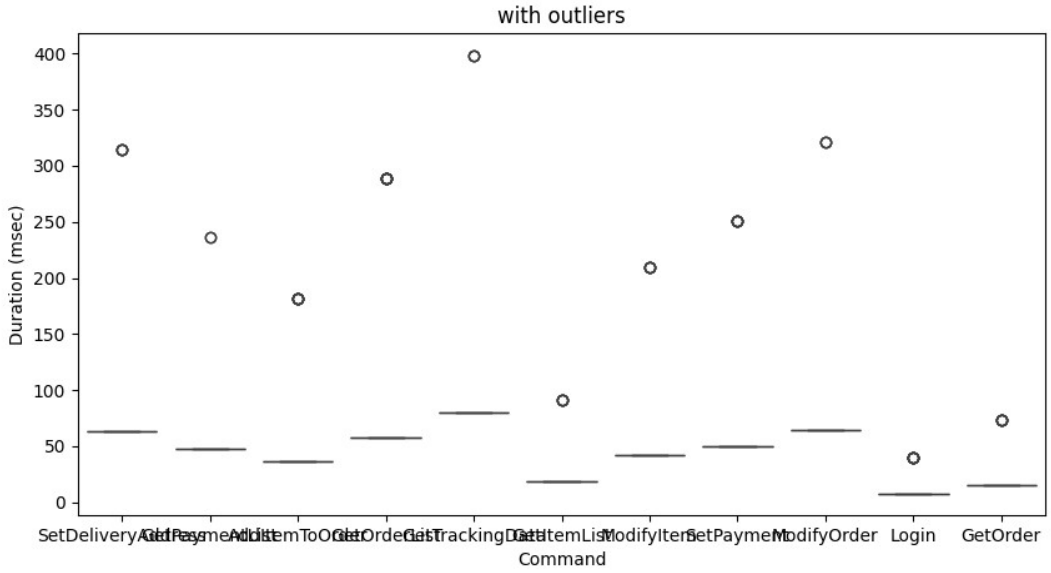

Command: GetOrder
Normal Distribution: N(14.71, 0.03)
Density
Response Time (ms)

4. Show the outliers for each command and then perform item 2 after excluding the outliers.

Without outliers



With outliers



```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Reading the provided assignment CSV file
log_file = pd.read_csv('assignment.csv')

# Converting dates to datetime format
log_file['Request date'] = pd.to_datetime(log_file['Request date'])
log_file['Response date'] = pd.to_datetime(log_file['Response date'])

# Calculating the duration in milliseconds
log_file['Duration (msec)'] = (log_file['Response date'] - log_file['Request date']).dt.total_seconds() * 1000

def filter_outliers(df):

    Quartile1 = df['Duration (msec)'].quantile(0.25) # Calculating first quartile as a Q1
    Quartile3 = df['Duration (msec)'].quantile(0.75) # Same thing that i mentioned for Q3

    InterQuartileRange = Quartile3 - Quartile1 # Calculating the InterQuartile Range (IQR)
    LB = Quartile1 - 1.5 * InterQuartileRange
    upper_bound = Quartile3 + 1.5 * InterQuartileRange


    return df[(df['Duration (msec)'] >= LB)
              & (df['Duration (msec)'] <= upper_bound)]

# Visualizing data with outliers

plt.figure(figsize=(10, 5))
sns.boxplot(x='Command',
            y='Duration (msec)',
              data=log_file)
plt.title('with outliers')
plt.show()

# Filtering out the outliers

clean_data = log_file.groupby('Command').apply(lambda x: filter_outliers(x)).reset_index(drop=True)

# Visualizing data without outliers

plt.figure(figsize=(10, 5))
sns.boxplot(x='Command'
            , y='Duration (msec)'
            ,data=clean_data)

plt.title('without outliers')
plt.show()

```

As the previous files, first of all we have to reading csv file from ‘assignment.csv’ file into a dataframe named ‘**log_file’**. Converting dates I have used Pandas **‘to_datetime’** functionality. A function named ‘**filter_outliers’** is defined to filter outliers from the date and function takes a dataframe as a **‘df’** input.

For calculating **Quartile 1 (Q1)** and **Quartile 3 (Q3)** are calculated using the quantile function. The **Interquartile Range (IQR**) is calculated as the *difference* between *Q3 and Q1.*