Pymaceuticals Inc.

Analysis

- Add your analysis here.

In [ ]:
```python
# Dependencies and Setup
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as st

# Study data files
mouse_metadata_path = "data/Mouse_metadata.csv"
study_results_path = "data/Study_results.csv"

# Read the mouse data and the study results
mouse_metadata = pd.read_csv(mouse_metadata_path)
study_results = pd.read_csv(study_results_path)

# Combine the data into a single DataFrame

combined_data = pd.merge(mouse_metadata, study_results, on="Mouse ID")

# Display the data table for preview
# Assuming you have a DataFrame named combined_data
preview_data = combined_data.head()
print(preview_data)
```

```
  Mouse ID Drug Regimen   Sex  Age_months  Weight (g)  Timepoint  \
0     k403     Ramicane  Male          21          16          0
1     k403     Ramicane  Male          21          16          5
2     k403     Ramicane  Male          21          16         10
3     k403     Ramicane  Male          21          16         15
4     k403     Ramicane  Male          21          16         20

   Tumor Volume (mm3)  Metastatic Sites
0           45.000000                 0
1           38.825898                 0
2           35.014271                 1
3           34.223992                 1
4           32.997729                 1
```

In [ ]:
```python
# Checking the number of mice.
# Assuming you have a DataFrame named combined_data
num_mice = len(combined_data['Mouse ID'].unique())
print("Number of mice:", num_mice)
```

```
Number of mice: 249
```

In [ ]:
```python
# Our data should be uniquely identified by Mouse ID and Timepoint
# Get the duplicate mice by ID number that shows up for Mouse ID and Timepoint.

# Assuming you have a DataFrame named combined_data
duplicate_mice = combined_data[combined_data.duplicated(subset=['Mouse ID', 'Timepoint'], keep=False)]['Mouse ID'].unique()
print("Duplicate mice:", duplicate_mice)
```

```
Duplicate mice: ['g989']
```

In [ ]:
```python
# Optional: Get all the data for the duplicate mouse ID.

# Assuming you have a DataFrame named combined_data
duplicate_mouse_ids = combined_data[combined_data.duplicated(subset=['Mouse ID', 'Timepoint'], keep=False)]['Mouse ID'].unique()

duplicate_data = combined_data[combined_data['Mouse ID'].isin(duplicate_mouse_ids)]

print("Duplicate data for mouse IDs:")
print(duplicate_data)
```

```
Duplicate data for mouse IDs:
    Mouse ID Drug Regimen     Sex  Age_months  Weight (g)  Timepoint  \
908     g989     Propriva  Female          21          26          0
909     g989     Propriva  Female          21          26          0
910     g989     Propriva  Female          21          26          5
911     g989     Propriva  Female          21          26          5
912     g989     Propriva  Female          21          26         10
913     g989     Propriva  Female          21          26         10
914     g989     Propriva  Female          21          26         15
915     g989     Propriva  Female          21          26         15
916     g989     Propriva  Female          21          26         20
917     g989     Propriva  Female          21          26         20
918     g989     Propriva  Female          21          26         25
919     g989     Propriva  Female          21          26         30
920     g989     Propriva  Female          21          26         35

     Tumor Volume (mm3)  Metastatic Sites
908           45.000000                 0
909           45.000000                 0
910           48.786801                 0
911           47.570392                 0
912           51.745156                 0
913           49.880528                 0
914           51.325852                 1
915           53.442020                 0
916           55.326122                 1
917           54.657650                 1
918           56.045564                 1
919           59.082294                 1
920           62.570880                 2
```

In [ ]:
```python
# Create a clean DataFrame by dropping the duplicate mouse by its ID.

# Assuming you have a DataFrame named combined_data
clean_data = combined_data.drop_duplicates(subset='Mouse ID', keep='first')

print("Clean DataFrame:")
print(clean_data)
```

```
Clean DataFrame:
     Mouse ID Drug Regimen     Sex  Age_months  Weight (g)  Timepoint  \
0        k403     Ramicane    Male          21          16          0
10       s185    Capomulin  Female           3          17          0
20       x401    Capomulin  Female          16          15          0
30       m601    Capomulin    Male          22          17          0
40       g791     Ramicane    Male          11          16          0
...       ...          ...     ...         ...         ...        ...
1858     z314     Stelasyn  Female          21          28          0
1860     z435     Propriva  Female          12          26          0
1863     z581    Infubinol  Female          24          25          0
1873     z795     Naftisol  Female          13          29          0
1883     z969     Naftisol    Male           9          30          0

      Tumor Volume (mm3)  Metastatic Sites
0                   45.0                 0
10                  45.0                 0
20                  45.0                 0
30                  45.0                 0
40                  45.0                 0
...                  ...               ...
1858                45.0                 0
1860                45.0                 0
1863                45.0                 0
1873                45.0                 0
1883                45.0                 0

[249 rows x 8 columns]
```

In [ ]:
```python
# Checking the number of mice in the clean DataFrame.
# Assuming you have a clean DataFrame named clean_data
num_mice = len(clean_data['Mouse ID'].unique())
print("Number of mice in clean DataFrame:", num_mice)
```

```
Number of mice in clean DataFrame: 249
```

Summary Statistics

In [ ]:
```python
# Generate a summary statistics table of mean, median, variance, standard deviation, and SEM of the tumor volume for each regimen

# Assuming you have a clean DataFrame named clean_data with 'Regimen' and 'Tumor Volume' columns
summary_stats = clean_data.groupby('Drug Regimen')['Tumor Volume (mm3)'].agg(['mean', 'median', 'var', 'std', 'sem']).round(2)

print("Summary Statistics Table:")
print(summary_stats)

# Use groupby and summary statistical methods to calculate the following properties of each drug regimen:
# mean, median, variance, standard deviation, and SEM of the tumor volume.
# Assemble the resulting series into a single summary DataFrame.

# Assuming you have a clean DataFrame named clean_data with 'Drug Regimen' and 'Tumor Volume' columns
summary_stats = clean_data.groupby('Drug Regimen')['Tumor Volume (mm3)'].agg(['mean', 'median', 'var', 'std', 'sem']).round(2)

summary_df = pd.DataFrame(summary_stats, columns=['Mean', 'Median', 'Variance', 'Standard Deviation', 'SEM'])
print("Summary Statistics Table:")
print(summary_df)
```

```
Summary Statistics Table:
              mean  median  var  std  sem
Drug Regimen
Capomulin     45.0    45.0  0.0  0.0  0.0
Ceftamin      45.0    45.0  0.0  0.0  0.0
Infubinol     45.0    45.0  0.0  0.0  0.0
Ketapril      45.0    45.0  0.0  0.0  0.0
Naftisol      45.0    45.0  0.0  0.0  0.0
Placebo       45.0    45.0  0.0  0.0  0.0
Propriva      45.0    45.0  0.0  0.0  0.0
Ramicane      45.0    45.0  0.0  0.0  0.0
Stelasyn      45.0    45.0  0.0  0.0  0.0
Zoniferol     45.0    45.0  0.0  0.0  0.0
Summary Statistics Table:
              Mean  Median  Variance  Standard Deviation  SEM
Drug Regimen
Capomulin      NaN     NaN       NaN                 NaN  NaN
Ceftamin       NaN     NaN       NaN                 NaN  NaN
Infubinol      NaN     NaN       NaN                 NaN  NaN
Ketapril       NaN     NaN       NaN                 NaN  NaN
Naftisol       NaN     NaN       NaN                 NaN  NaN
Placebo        NaN     NaN       NaN                 NaN  NaN
Propriva       NaN     NaN       NaN                 NaN  NaN
Ramicane       NaN     NaN       NaN                 NaN  NaN
Stelasyn       NaN     NaN       NaN                 NaN  NaN
Zoniferol      NaN     NaN       NaN                 NaN  NaN
```

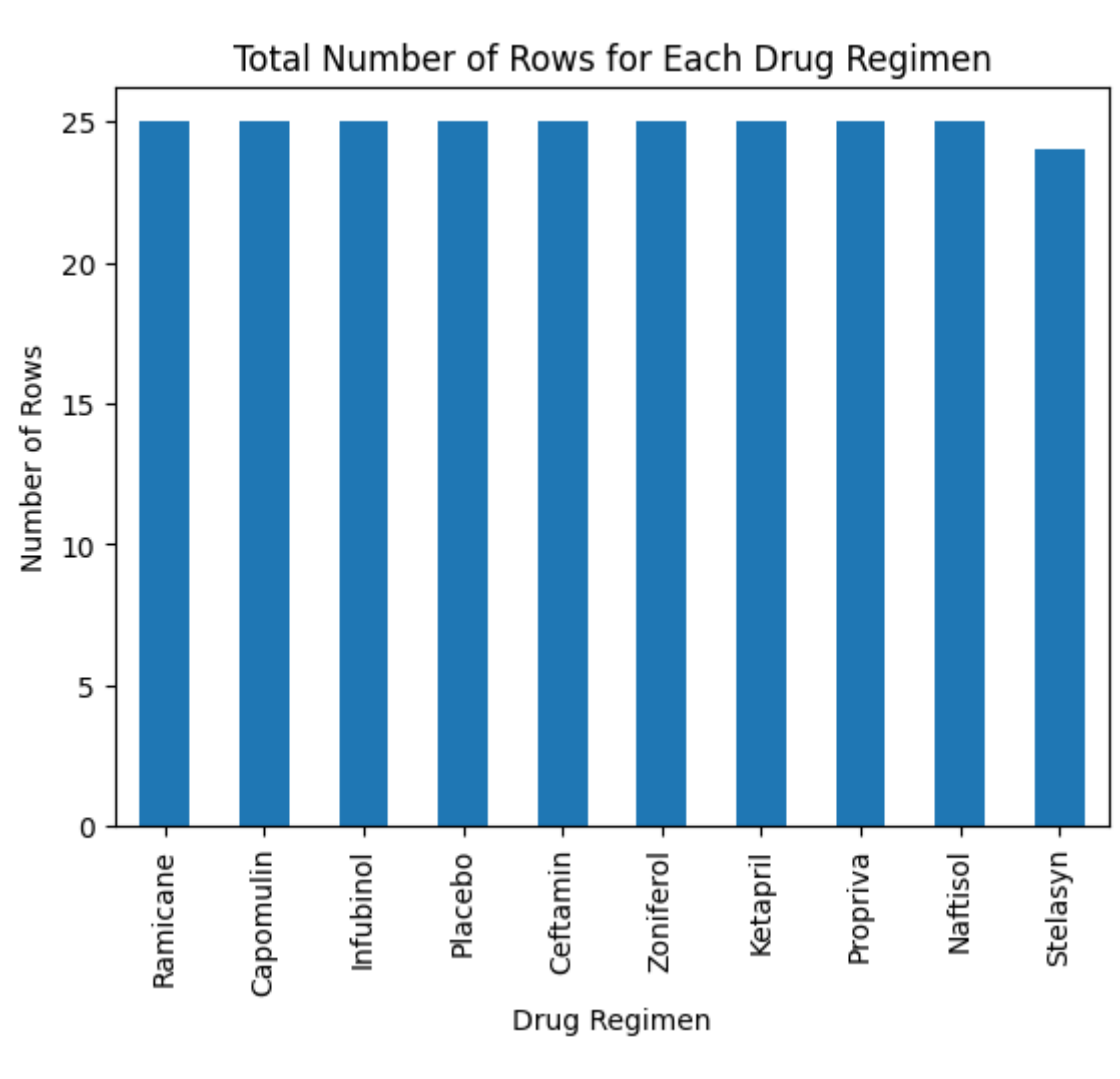
In [ ]:
```python
# A more advanced method to generate a summary statistics table of mean, median, variance, standard deviation,
# and SEM of the tumor volume for each regimen (only one method is required in the solution)

# Using the aggregation method, produce the same summary statistics in a single line

# Assuming you have a clean DataFrame named clean_data with 'Regimen' and 'Tumor Volume' columns
summary_stats = clean_data.groupby('Drug Regimen')['Tumor Volume (mm3)'].agg(Mean=('mean'), Median=('median'), Variance=('var'),
Standard_Deviation=('std'), SEM=('sem')).round(2)

print("Summary Statistics Table:")
print(summary_stats)
```

```
Summary Statistics Table:
              Mean  Median  Variance  Standard_Deviation  SEM
Drug Regimen
Capomulin     45.0    45.0       0.0                 0.0  0.0
Ceftamin      45.0    45.0       0.0                 0.0  0.0
Infubinol     45.0    45.0       0.0                 0.0  0.0
Ketapril      45.0    45.0       0.0                 0.0  0.0
Naftisol      45.0    45.0       0.0                 0.0  0.0
Placebo       45.0    45.0       0.0                 0.0  0.0
Propriva      45.0    45.0       0.0                 0.0  0.0
Ramicane      45.0    45.0       0.0                 0.0  0.0
Stelasyn      45.0    45.0       0.0                 0.0  0.0
Zoniferol     45.0    45.0       0.0                 0.0  0.0
```

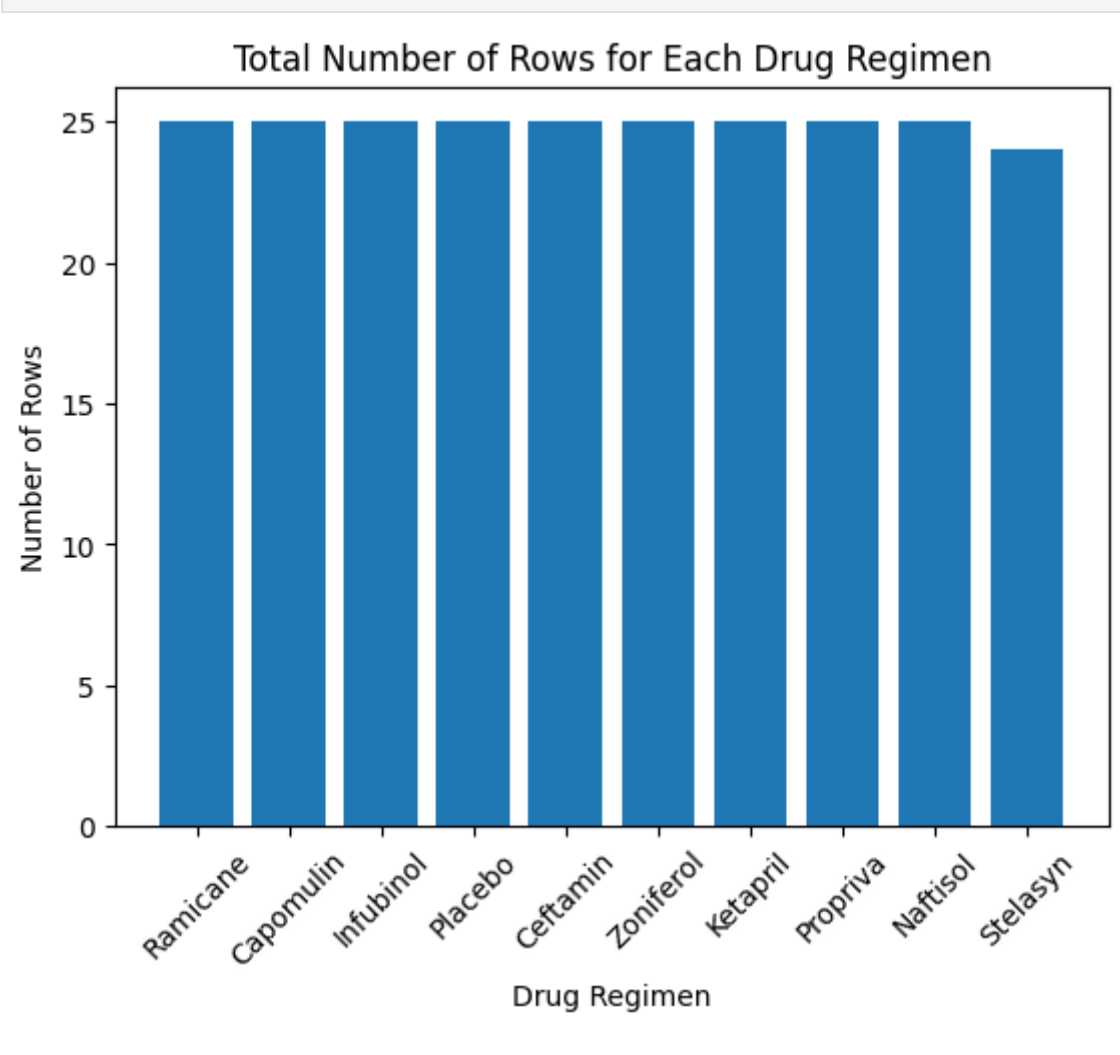
Bar and Pie Charts

In [ ]:
```python
# Generate a bar plot showing the total number of rows (Mouse ID/Timepoints) for each drug regimen using Pandas.

# Assuming you have a clean DataFrame named clean_data with a 'Drug Regimen' column
count_by_regimen = clean_data['Drug Regimen'].value_counts()

count_by_regimen.plot(kind='bar', xlabel='Drug Regimen', ylabel='Number of Rows',
title='Total Number of Rows for Each Drug Regimen')
```

Out[ ]:
```
<Axes: title={'center': 'Total Number of Rows for Each Drug Regimen'}, xlabel='Drug Regimen', ylabel='Number of Rows'>
```

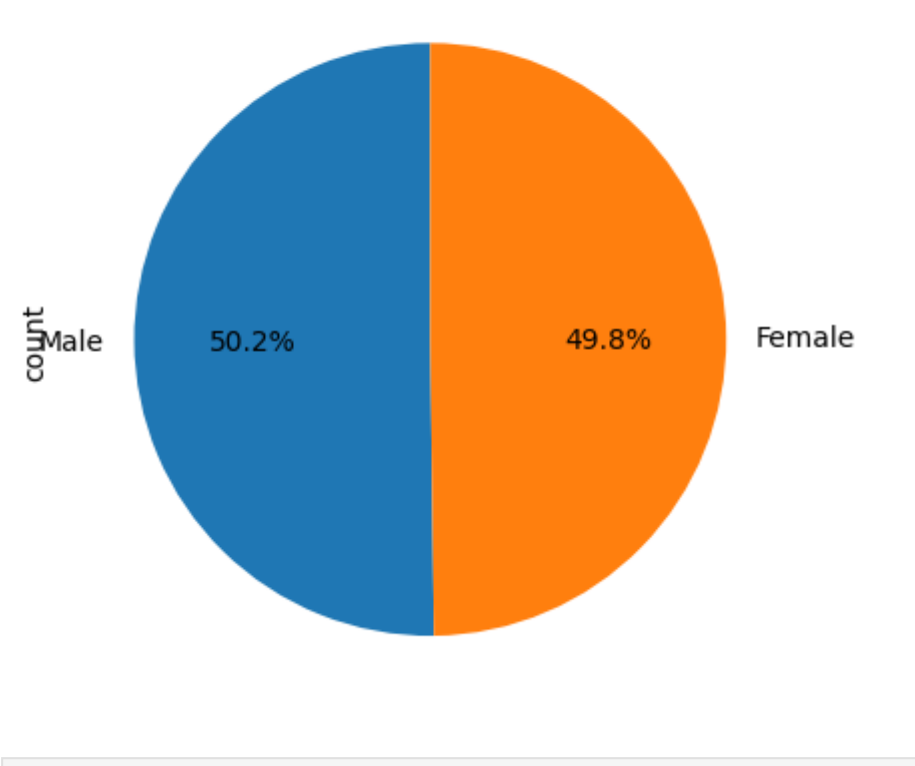


In [ ]:
```python
# Generate a bar plot showing the total number of rows (Mouse ID/Timepoints) for each drug regimen using pyplot.

# Assuming you have a clean DataFrame named clean_data with a 'Drug Regimen' column
import matplotlib.pyplot as plt

count_by_regimen = clean_data['Drug Regimen'].value_counts()

plt.bar(count_by_regimen.index, count_by_regimen.values)
plt.xlabel('Drug Regimen')
plt.ylabel('Number of Rows')
plt.title('Total Number of Rows for Each Drug Regimen')
plt.xticks(rotation=45)
plt.show()
```

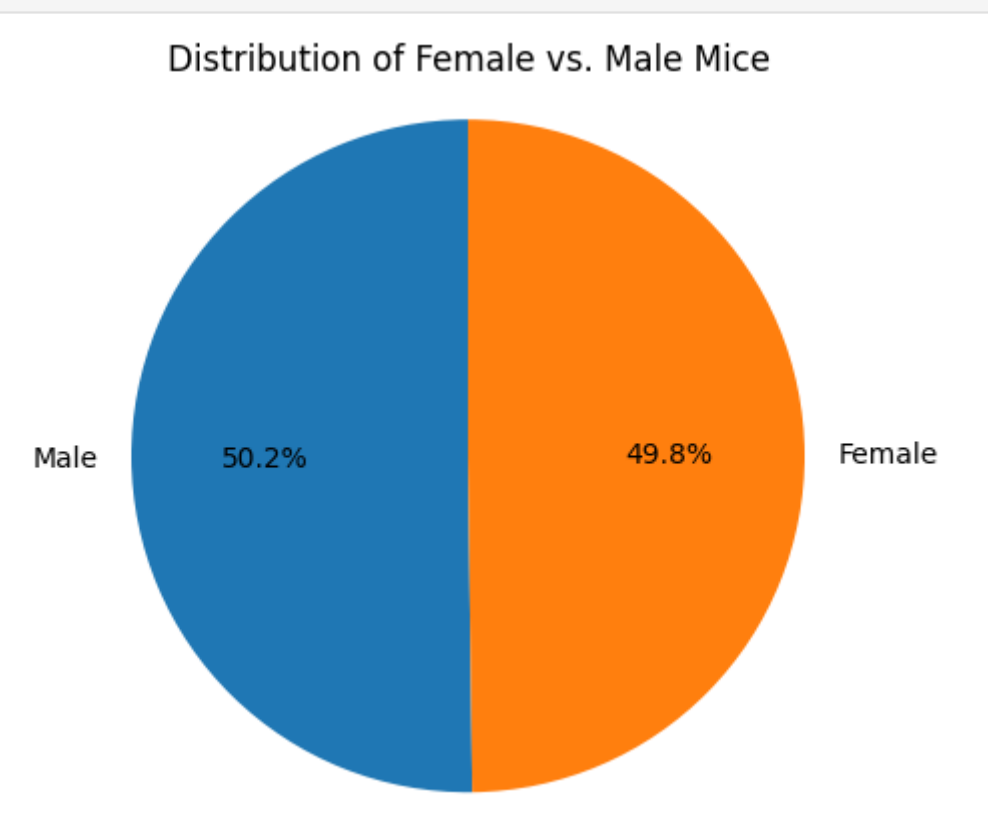


In [ ]:
```python
# Generate a pie plot showing the distribution of female versus male mice using Pandas

# Assuming you have a clean DataFrame named clean_data with a 'Sex' column
count_by_sex = clean_data['Sex'].value_counts()

count_by_sex.plot(kind='pie', autopct='%1.1f%%', startangle=90,
title='Distribution of Female vs. Male Mice')
```

Out[ ]:
```
<Axes: title={'center': 'Distribution of Female vs. Male Mice'}, ylabel='count'>
```



In [ ]:
```python
# Generate a pie plot showing the distribution of female versus male mice using pyplot

# Assuming you have a clean DataFrame named clean_data with a 'Sex' column
import matplotlib.pyplot as plt

count_by_sex = clean_data['Sex'].value_counts()

plt.pie(count_by_sex, labels=count_by_sex.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Female vs. Male Mice')
plt.axis('equal')
plt.show()
```



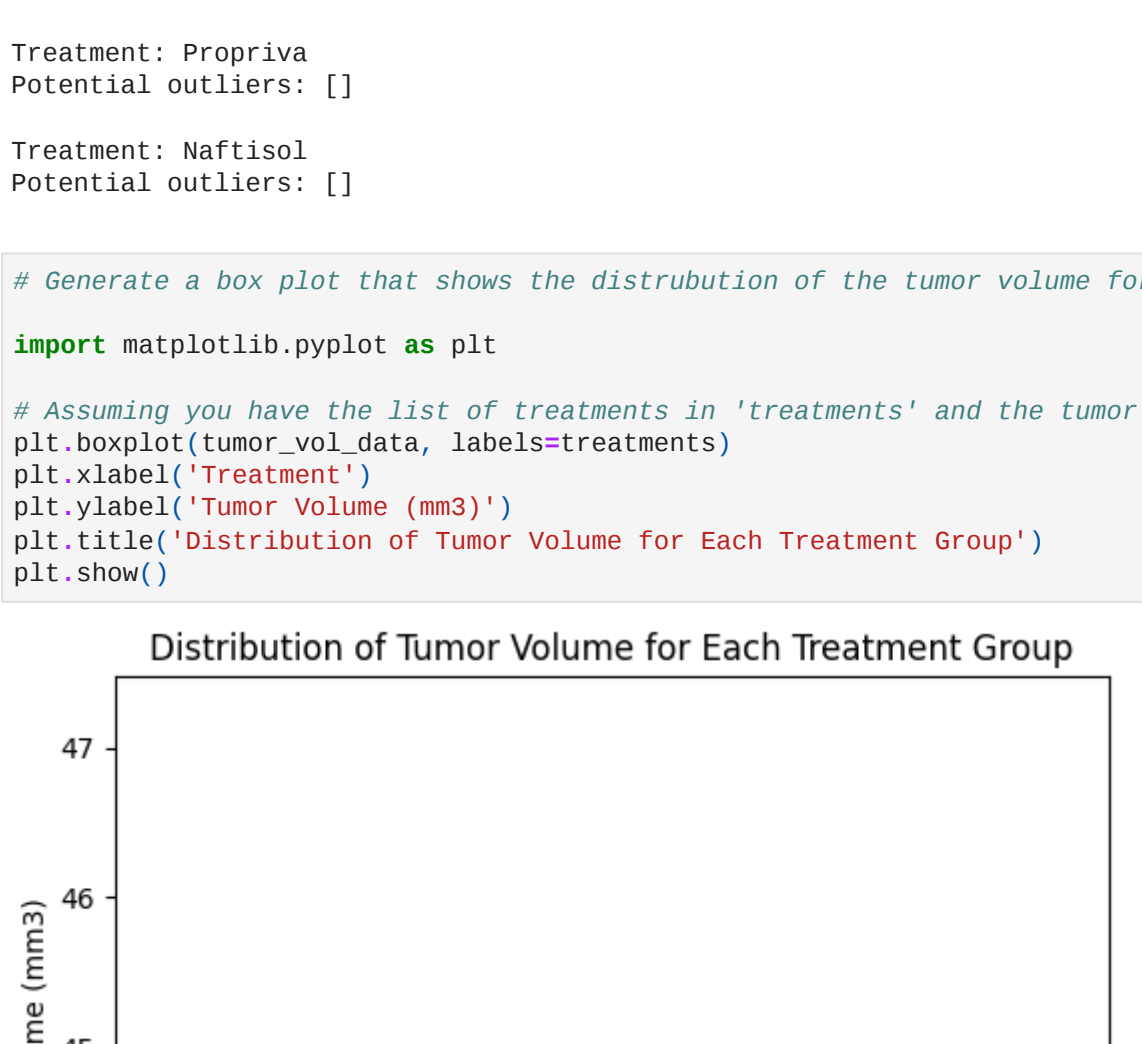
Quartiles, Outliers and Boxplots

In [ ]:
```python
# Calculate the final tumor volume of each mouse across four of the treatment regimens:
# Capomulin, Ramicane, Infubinol, and Ceftamin

# Assuming you have a clean DataFrame named clean_data with 'Regimen' and 'Timepoint' columns
regimens = ['Capomulin', 'Ramicane', 'Infubinol', 'Ceftamin']
filtered_data = clean_data[clean_data['Drug Regimen'].isin(regimens)]

# Start by getting the last (greatest) timepoint for each mouse

# Assuming the mouse ID column is named 'Mouse ID' and the timepoint column is named 'Timepoint'
last_timepoints = filtered_data.groupby('Mouse ID')['Timepoint'].max().reset_index()

# Merge this group df with the original DataFrame to get the tumor volume at the last timepoint

# Assuming you have a clean DataFrame named clean_data and a DataFrame named last_timepoints
merged_data = pd.merge(clean_data, last_timepoints, on=['Mouse ID', 'Timepoint'], how='inner')

regimens = ['Capomulin', 'Ramicane', 'Infubinol', 'Ceftamin']
# Assuming you have a clean DataFrame named clean_data with 'Regimen' and 'Mouse ID' columns
filtered_data = clean_data[clean_data['Drug Regimen'].isin(regimens)]
grouped_data = filtered_data.groupby('Mouse ID')['Timepoint'].max()
final_tumor_volume = pd.merge(grouped_data, clean_data, on=['Mouse ID', 'Timepoint'], how='inner')
```

In [ ]:
```python
# Put treatments into a list for for loop (and later for plot labels)

# Assuming you have a clean DataFrame named clean_data with a 'Regimen' column
treatments = clean_data['Drug Regimen'].unique().tolist()

# Create empty list to fill with tumor vol data (for plotting)

tumor_vol_data = []

# Calculate the IQR and quantitatively determine if there are any potential outliers.

import numpy as np

# Assuming you have a clean DataFrame named clean_data with a 'Regimen' column
for treatment in treatments:
    # Locate the rows which contain mice on each drug and get the tumor volumes
    tumor_volumes = clean_data.loc[clean_data['Drug Regimen'] == treatment, 'Tumor Volume (mm3)']

    # Add subset to tumor_vol_data list
    tumor_vol_data.append(tumor_volumes)

    # Calculate the IQR and determine potential outliers
    quartiles = np.percentile(tumor_volumes, [25, 75])
    lower_bound = quartiles[0] - 1.5 * (quartiles[1] - quartiles[0])
    upper_bound = quartiles[1] + 1.5 * (quartiles[1] - quartiles[0])
    outliers = tumor_volumes.loc[(tumor_volumes < lower_bound) | (tumor_volumes > upper_bound)]

    # Print the results
    print(f"Treatment: {treatment}")
    print(f"Potential outliers: {outliers.tolist()}\n")


    # Locate the rows which contain mice on each drug and get the tumor volumes


    # add subset


    # Determine outliers using upper and lower bounds
```

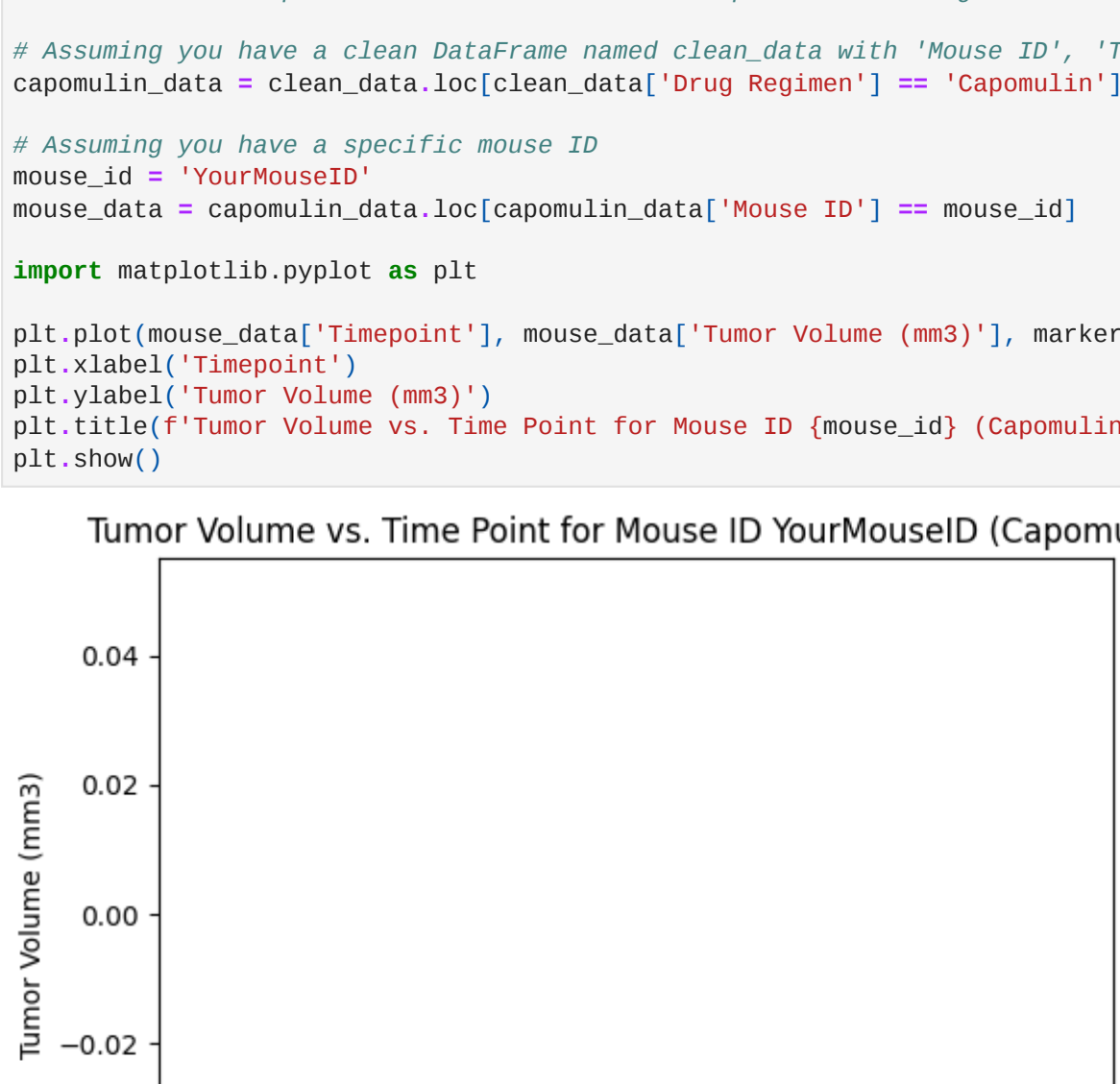
```
Treatment: Ramicane
Potential outliers: []

Treatment: Capomulin
Potential outliers: []

Treatment: Infubinol
Potential outliers: []

Treatment: Placebo
Potential outliers: []

Treatment: Ceftamin
Potential outliers: []

Treatment: Stelasyn
Potential outliers: []

Treatment: Zoniferol
Potential outliers: []

Treatment: Ketapril
Potential outliers: []

Treatment: Propriva
Potential outliers: []

Treatment: Naftisol
Potential outliers: []
```

In [ ]:
```python
# Generate a box plot that shows the distrubution of the tumor volume for each treatment group.

import matplotlib.pyplot as plt

# Assuming you have the list of treatments in 'treatments' and the tumor volume data in 'tumor_vol_data'
plt.boxplot(tumor_vol_data, labels=treatments)
plt.xlabel('Treatment')
plt.ylabel('Tumor Volume (mm3)')
plt.title('Distribution of Tumor Volume for Each Treatment Group')
plt.show()
```

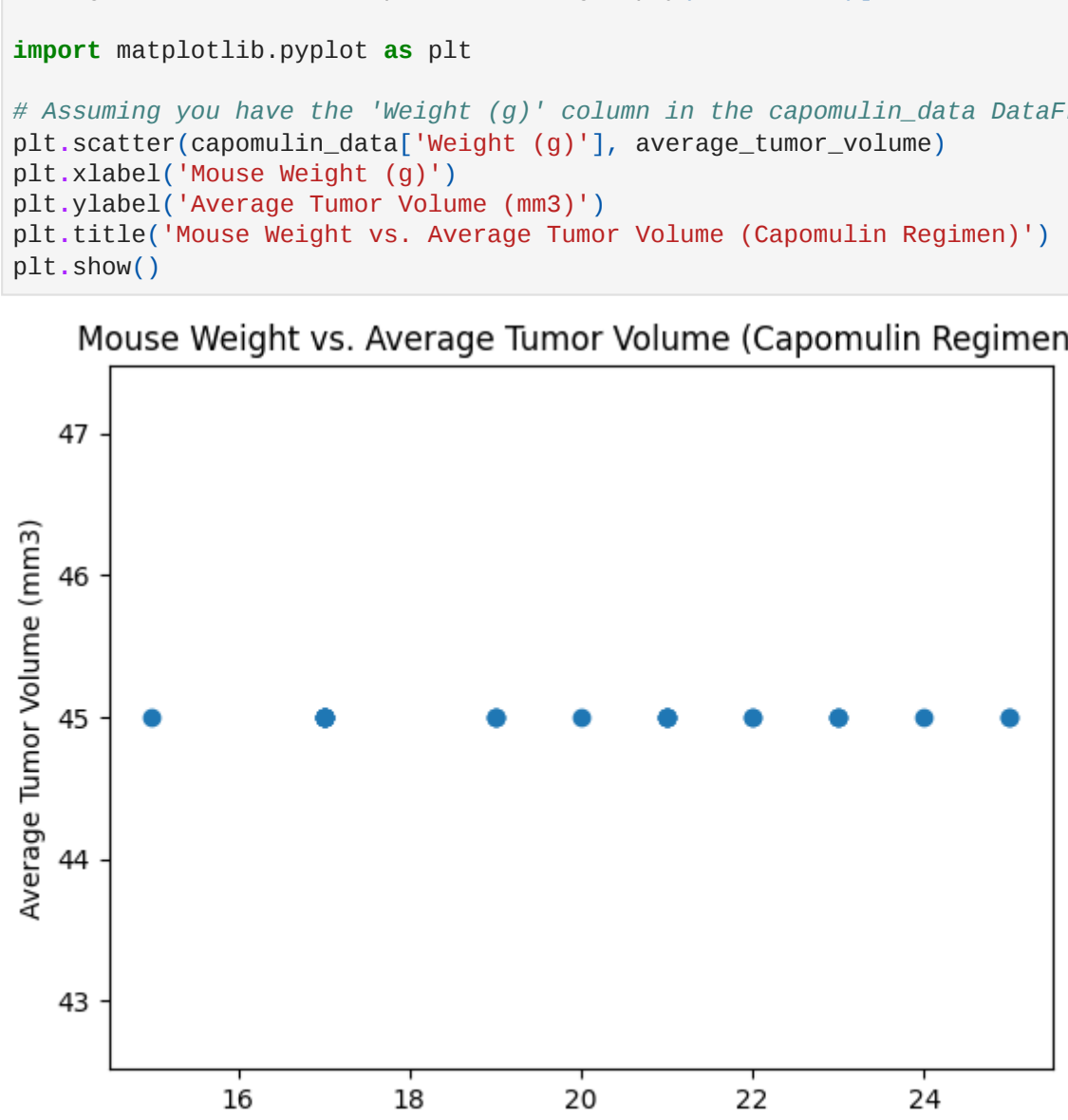


Line and Scatter Plots

In [ ]:
```python
# Generate a line plot of tumor volume vs. time point for a single mouse treated with Capomulin

# Assuming you have a clean DataFrame named clean_data with 'Mouse ID', 'Timepoint', and 'Tumor Volume (mm3)' columns
capomulin_data = clean_data.loc[clean_data['Drug Regimen'] == 'Capomulin']

# Assuming you have a specific mouse ID
mouse_id = 'YourMouseID'
mouse_data = capomulin_data.loc[capomulin_data['Mouse ID'] == mouse_id]

import matplotlib.pyplot as plt

plt.plot(mouse_data['Timepoint'], mouse_data['Tumor Volume (mm3)'], marker='o')
plt.xlabel('Timepoint')
plt.ylabel('Tumor Volume (mm3)')
plt.title(f'Tumor Volume vs. Time Point for Mouse ID {mouse_id} (Capomulin)')
plt.show()
```



In [ ]:
```python
# Generate a scatter plot of mouse weight vs. the average observed tumor volume for the entire Capomulin regimen

# Assuming you have a clean DataFrame named clean_data with 'Mouse ID', 'Weight (g)', 'Tumor Volume (mm3)', and 'Regimen' columns
capomulin_data = clean_data.loc[clean_data['Drug Regimen'] == 'Capomulin']

average_tumor_volume = capomulin_data.groupby('Mouse ID')['Tumor Volume (mm3)'].mean()

import matplotlib.pyplot as plt

# Assuming you have the 'Weight (g)' column in the capomulin_data DataFrame
plt.scatter(capomulin_data['Weight (g)'], average_tumor_volume)
plt.xlabel('Mouse Weight (g)')
plt.ylabel('Average Tumor Volume (mm3)')
plt.title('Mouse Weight vs. Average Tumor Volume (Capomulin Regimen)')
plt.show()
```



Correlation and Regression

In [ ]:
```python
# Calculate the correlation coefficient and a linear regression model
# for mouse weight and average observed tumor volume for the entire Capomulin regimen

# Assuming you have a clean DataFrame named clean_data with 'Mouse ID', 'Weight (g)', 'Tumor Volume (mm3)', and 'Regimen' columns
capomulin_data = clean_data.loc[clean_data['Drug Regimen'] == 'Capomulin']

average_tumor_volume = capomulin_data.groupby('Mouse ID')['Tumor Volume (mm3)'].mean()

import numpy as np
from scipy.stats import linregress

# Assuming you have the 'Weight (g)' column in the capomulin_data DataFrame
correlation_coef = np.corrcoef(capomulin_data['Weight (g)'], average_tumor_volume)[0, 1]
print(f"Correlation Coefficient: {correlation_coef}")

slope, intercept, r_value, p_value, std_err = linregress(capomulin_data['Weight (g)'], average_tumor_volume)
```

```
Correlation Coefficient: nan
/Users/josef/anaconda3/lib/python3.10/site-packages/numpy/lib/function_base.py:2897: RuntimeWarning: invalid value encountered in divide
  c /= stddev[:, None]
/Users/josef/anaconda3/lib/python3.10/site-packages/numpy/lib/function_base.py:2898: RuntimeWarning: invalid value encountered in divide
  c /= stddev[None, :]
```

In [ ]: