In [1]:

```
import pandas as pd
```

In [1]:

```
import numpy as np
```

In [3]:

```
import matplotlib as mpl
```

In [4]:

```
import matplotlib.pyplot as plt
```

In [5]:

```
import seaborn as sns
```

In [4]:

```
#1.1
import pandas as pd
weights_series = pd.read_csv(r"C:\Users\gabed\.ipython\weights.tsv")
print(weights_series)
```

```
    164
0   158
1   172
2   153
3   144
4   156
5   189
6   163
7   134
8   159
9   143
10  176
11  177
12  162
13  141
14  151
15  182
16  185
17  171
18  152
```

In [8]:

```
#1.1
data = pd.Series([164, 158,
172, 153, 144, 156, 189, 163, 134, 159, 143, 176, 177, 162, 141, 151, 182, 185, 171,
152])
data
```

Out[8]:

```
0     164
1     158
2     172
3     153
4     144
5     156
6     189
7     163
8     134
9     159
10    143
11    176
12    177
13    162
14    141
15    151
16    182
17    185
18    171
19    152
dtype: int64
```

In [2]:

```
#1.2
import pandas as pd

weight_lbs = ([164, 158,
172, 153, 144, 156, 189, 163, 134, 159, 143, 176, 177, 162, 141, 151, 182, 185, 171,
152])

weight_kg = [round(weight * 0.453592,2) for weight in weight_lbs]

weight_series = pd.Series(weight_kg)

print(weight_series)
```

```
0     74.39
1     71.67
2     78.02
3     69.40
4     65.32
5     70.76
6     85.73
7     73.94
8     60.78
9     72.12
10    64.86
11    79.83
12    80.29
13    73.48
14    63.96
15    68.49
16    82.55
17    83.91
18    77.56
19    68.95
dtype: float64
```

In [39]:

```
#1.3
import pandas as pd

weight_lbs = ([164, 158,
172, 153, 144, 156, 189, 163, 134, 159, 143, 176, 177, 162, 141, 151, 182, 185, 171,
152])

weight_series_lbs = pd.Series(weight_lbs)

weight_kg = weight_series_lbs * 0.453592

mean_lbs = weight_series_lbs.mean()
median_lbs = weight_series_lbs.median()
std_dev_lbs = weight_series_lbs.std()

mean_kg = weight_kg.mean()
median_kg = weight_kg.median()
std_dev_kg = weight_kg.std()


print("weight in lbs")
print("Mean:",mean_lbs)
print("Median:", median_lbs)
print("Standard Deviation:", std_dev_lbs)
print("weight in kg:")
print("Mean:", mean_kg)
print("Median:", median_kg)
print("Standard Deviation:", std_dev_kg)
```

```
weight in lbs
Mean: 161.6
Median: 160.5
Standard Deviation: 15.44906742203316
weight in kg:
Mean: 73.30046720000001
Median: 72.80151599999999
Standard Deviation: 7.007573390094864
```

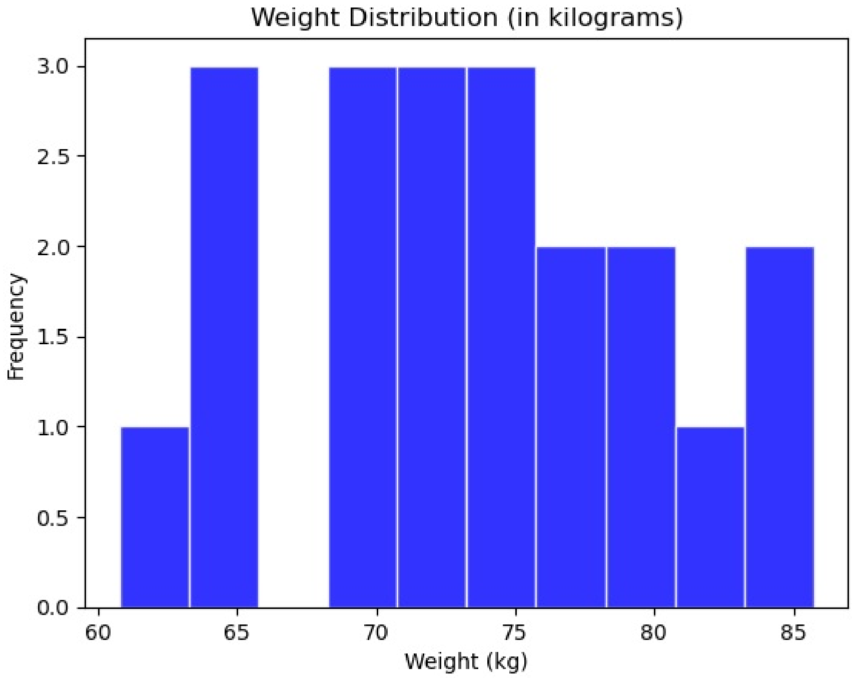
In [20]:

```
#1.4
import matplotlib.pyplot as plt

weights_lbs = [164, 158, 172, 153, 144, 156, 189, 163, 134, 159, 143, 176, 177, 162, 141, 151, 182, 185, 171, 1

weights_kg = [weight * 0.453592 for weight in weights_lbs]


plt.hist(weights_kg, bins=10, color='blue', edgecolor='white', alpha=0.8)
plt.title('Weight Distribution (in kilograms)')
plt.xlabel('Weight (kg)')
plt.ylabel('Frequency')
plt.show()
```



In [18]:

```
#2.1
import pandas as pd
df = pd.read_csv(r"C:\Users\gabed\.ipython\boston.csv")
num_rows, num_cols = df.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_cols)
```

```
Number of rows: 506
Number of columns: 13
```

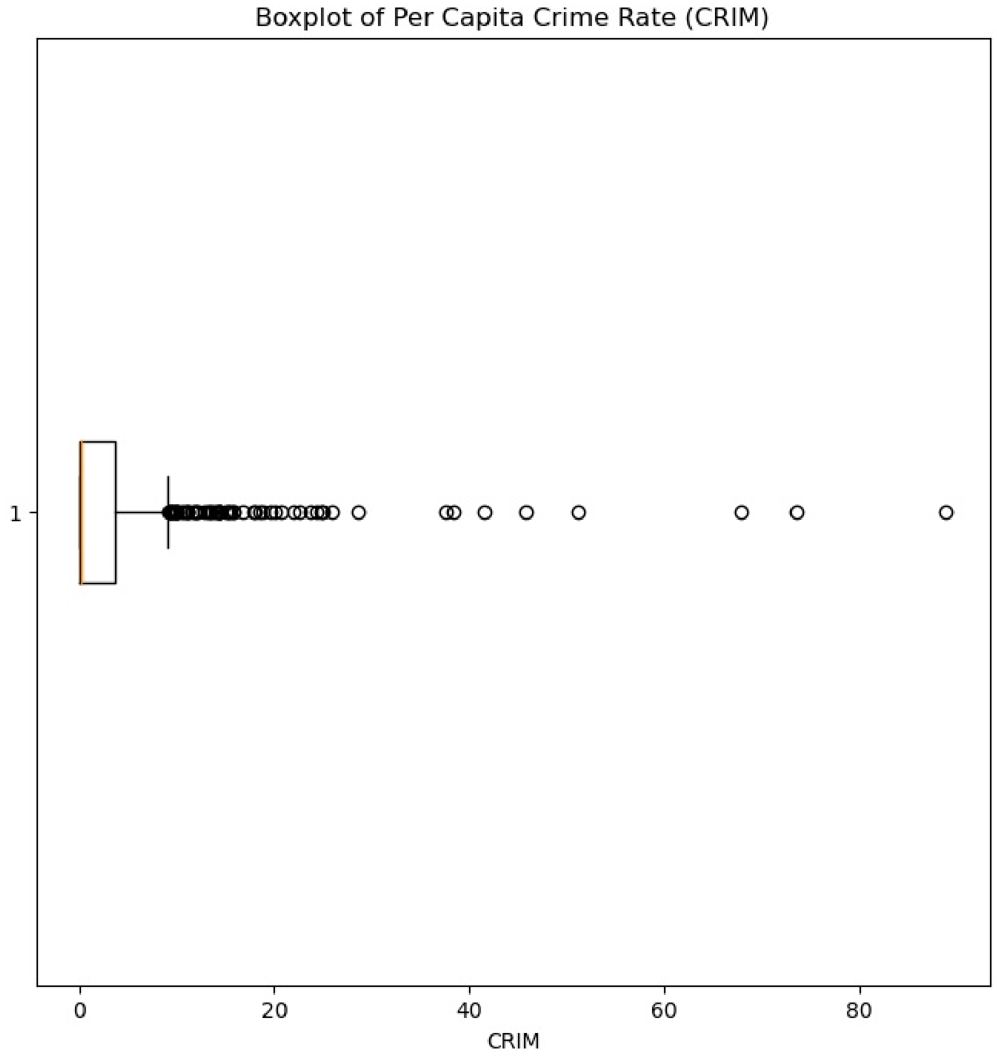
In [14]:

```
#2.2
import pandas as pd
df = pd.read_csv(r"C:\Users\gabed\.ipython\boston.csv")
index_lowest_nox = df['NOX'].idxmin()
medv_lowest_nox = df.loc[index_lowest_nox, 'MEDV']
print("owner-occupied home value (MEDV) for the lowest nitric oxide concentration (NOX):", medv_lowest_nox)
```

```
owner-occupied home value (MEDV) for the lowest nitric oxide concentration (NOX): 20.1
```

In [19]:

```
#2.3
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\gabed\.ipython\boston.csv")
plt.figure(figsize=(8, 8))
plt.boxplot(df['CRIM'], vert=False)
plt.title('Boxplot of Per Capita Crime Rate (CRIM)')
plt.xlabel('CRIM')
plt.show()
Q1 = df['CRIM'].quantile(0.25)
Q3 = df['CRIM'].quantile(0.75)
IQR = Q3 - Q1
print("Interquartile Range (IQR) for Crime Rate (CRIM):", IQR)
```



```
Interquartile Range (IQR) for Crime Rate (CRIM): 3.5950375
```

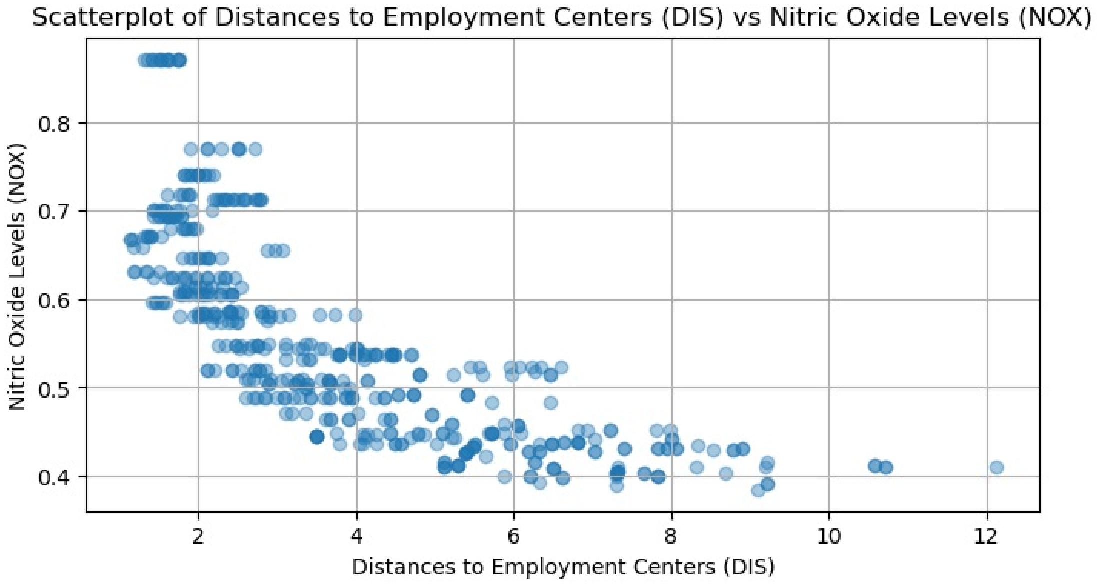
In [11]:

```
#2.4
import pandas as pd
df = pd.read_csv(r"C:\Users\gabed\.ipython\boston.csv")
Q1 = df['CRIM'].quantile(0.25)
Q3 = df['CRIM'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_df = df[(df['CRIM'] < lower_bound) | (df['CRIM'] > upper_bound)]
mean_age_original = df['AGE'].mean()
mean_age_outliers = outliers_df['AGE'].mean()
print("Mean AGE in original dataframe:", mean_age_original)
print("Mean AGE in subsetted dataframe (with outliers):", mean_age_outliers)
```

```
Mean AGE in original dataframe: 68.57490118577078
Mean AGE in subsetted dataframe (with outliers): 94.23333333333335
```

In [13]:

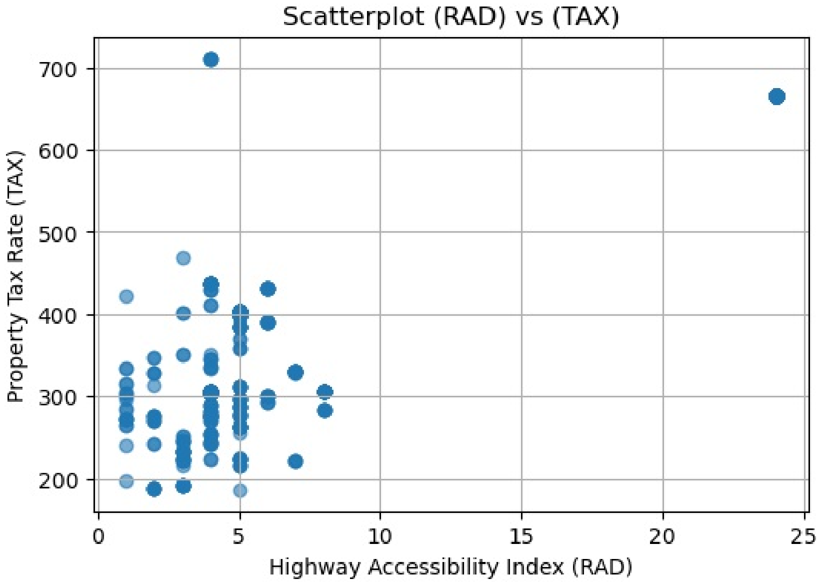
```
#2.5
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\gabed\.ipython\boston.csv")
plt.figure(figsize=(8, 4))
plt.scatter(df['DIS'], df['NOX'], alpha=0.4)
plt.title('Scatterplot of Distances to Employment Centers (DIS) vs Nitric Oxide Levels (NOX)')
plt.xlabel('Distances to Employment Centers (DIS)')
plt.ylabel('Nitric Oxide Levels (NOX)')
plt.grid(True)
plt.show()
correlation_index = df['DIS'].corr(df['NOX'])
print("Correlation index between DIS and NOX:", correlation_index)
```



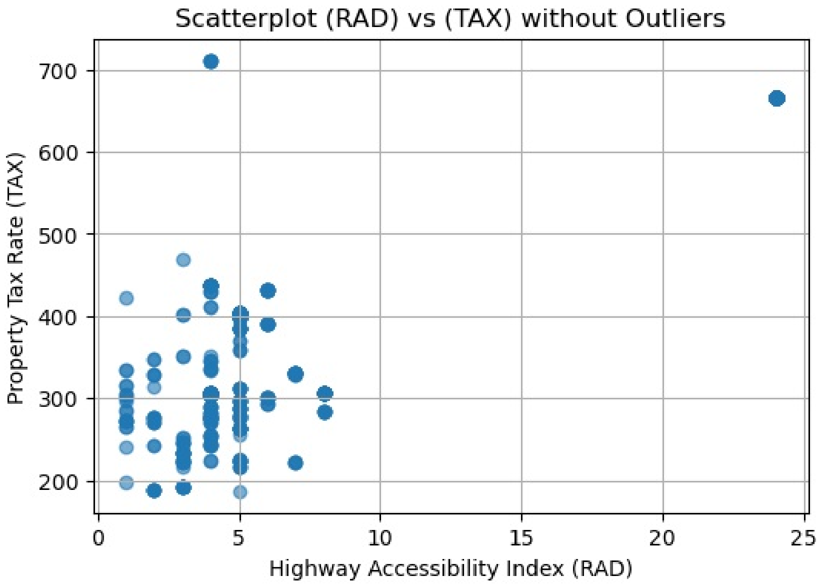
```
Correlation index between DIS and NOX: -0.7692301132258278
```

In [5]:

```
#2.6
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\gabed\.ipython\boston.csv")
plt.figure(figsize=(6, 4))
plt.scatter(df['RAD'], df['TAX'], alpha=0.6)
plt.title('Scatterplot (RAD) vs (TAX)')
plt.xlabel('Highway Accessibility Index (RAD)')
plt.ylabel('Property Tax Rate (TAX)')
plt.grid(True)
plt.show()
correlation_index = df['RAD'].corr(df['TAX'])
print("Correlation index between RAD and TAX:", correlation_index)
upper_bound = df['TAX'].quantile(0.75) + 1.5 * (df['TAX'].quantile(0.75) - df['TAX'].quantile(0.25))
outlier_condition = df['TAX'] > upper_bound
df_cleaned = df[~outlier_condition]
plt.figure(figsize=(6, 4))
plt.scatter(df_cleaned['RAD'], df_cleaned['TAX'], alpha=0.6)
plt.title('Scatterplot (RAD) vs (TAX) without Outliers')
plt.xlabel('Highway Accessibility Index (RAD)')
plt.ylabel('Property Tax Rate (TAX)')
plt.grid(True)
plt.show()
correlation_index_cleaned = df_cleaned['RAD'].corr(df_cleaned['TAX'])
print("Correlation index between RAD and TAX after removing outliers:", correlation_index_cleaned)
```



```
Correlation index between RAD and TAX: 0.9102281885331835
```



```
Correlation index between RAD and TAX after removing outliers: 0.9102281885331835
```

In [17]:

```
#3.1
import seaborn as sns
tips_df= sns.load_dataset('tips')
tips_df.head()
tips_df['tip_percent'] = (tips_df['tip'] / tips_df['total_bill']) * 100
tips_df['tip_percent'] = tips_df['tip_percent'].round(2)
print(tips_df.head())
```

```
   total_bill   tip     sex smoker  day    time  size  tip_percent
0       16.99  1.01  Female     No  Sun  Dinner     2         5.94
1       10.34  1.66    Male     No  Sun  Dinner     3        16.05
2       21.01  3.50    Male     No  Sun  Dinner     3        16.66
3       23.68  3.31    Male     No  Sun  Dinner     2        13.98
4       24.59  3.61  Female     No  Sun  Dinner     4        14.68
```

In [8]:

```
#3.2
import seaborn as sns
tips_df = sns.load_dataset('tips')
mean_bill_per_day = tips_df.groupby('day')['total_bill'].mean()
day_highest_mean_bill = mean_bill_per_day.idxmax()
highest_mean_bill = mean_bill_per_day.max()
print("Days in the dataset:", mean_bill_per_day.index.tolist())
print("Day with the highest bill mean:", day_highest_mean_bill)
print("Highest mean bill amount:", highest_mean_bill)
```

```
Days in the dataset: ['Thur', 'Fri', 'Sat', 'Sun']
Day with the highest bill mean: Sun
Highest mean bill amount: 21.41
```

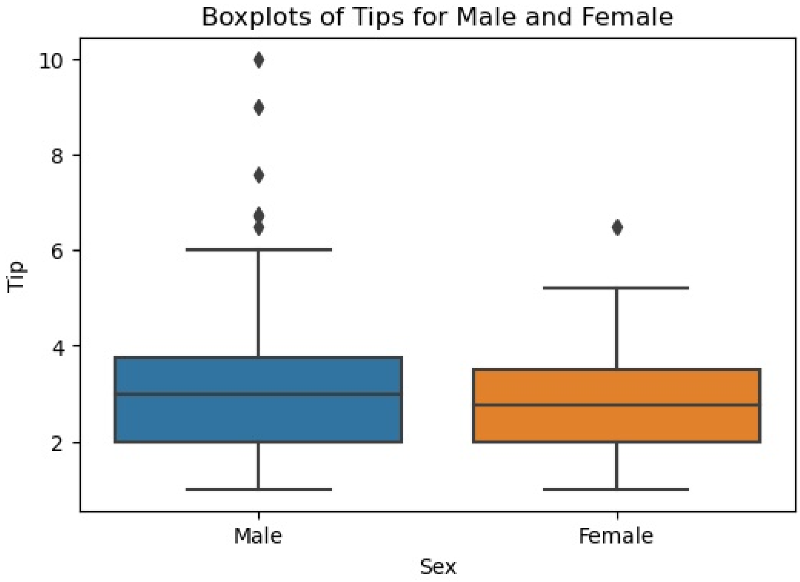
In [9]:

```
#3.3
import seaborn as sns
import pandas as pd
tips_df = sns.load_dataset('tips')
time_counts = tips_df['time'].value_counts()
smokers_during_lunch = tips_df[tips_df['time'] == 'Lunch']['smoker'].value_counts()
smokers_during_dinner = tips_df[tips_df['time'] == 'Dinner']['smoker'].value_counts()
ime_of_day_df = pd.DataFrame({'time_of_day': time_counts.index, 'count': time_counts.values})
smokers_lunch_df = pd.DataFrame({'smoker': smokers_during_lunch.index, 'count': smokers_during_lunch.values})
smokers_dinner_df = pd.DataFrame({'smoker': smokers_during_dinner.index, 'count': smokers_during_dinner.values}
merged_df = pd.merge(smokers_lunch_df, smokers_dinner_df, on='smoker', suffixes=('_lunch', '_dinner'))
merged_df['percent_lunch'] = (merged_df['count_lunch'] / time_counts['Lunch']) * 100
merged_df['percent_dinner'] = (merged_df['count_dinner'] / time_counts['Dinner']) * 100
print("Percentage of smokers during lunch and dinner:")
print(merged_df)
```

```
Percentage of smokers during lunch and dinner:
  smoker  count_lunch  count_dinner  percent_lunch  percent_dinner
0     No           45           106      66.176471       60.227273
1    Yes           23            70      33.823529       39.772727
```

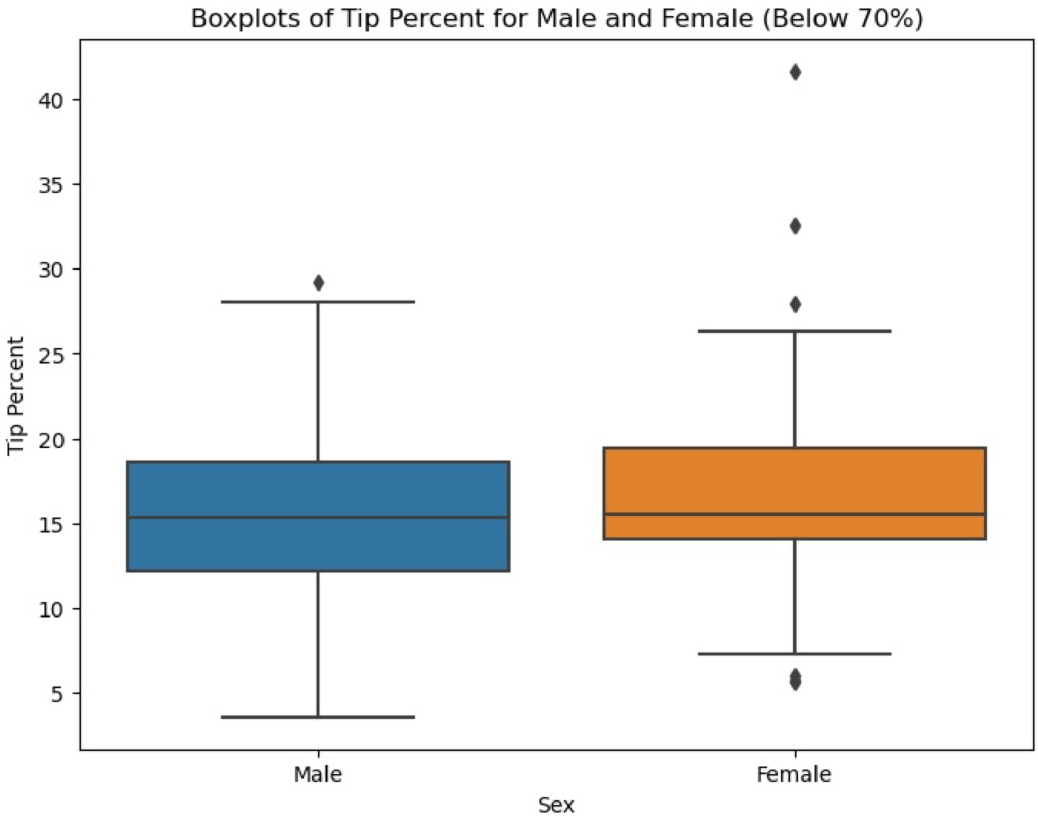
In [11]:

```
#3.4
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(6, 4))
sns.boxplot(x='sex', y='tip', data=tips_df)
plt.title('Boxplots of Tips for Male and Female')
plt.xlabel('Sex')
plt.ylabel('Tip')
plt.show()
```



In [3]:

```
#3.5
import seaborn as sns
import matplotlib.pyplot as plt
tips_df = sns.load_dataset('tips')
tips_df['tip_percent'] = (tips_df['tip'] / tips_df['total_bill']) * 100
filtered_tips_df = tips_df[tips_df['tip_percent'] < 70]
plt.figure(figsize=(8, 6))
sns.boxplot(x='sex', y='tip_percent', data=filtered_tips_df)
plt.title('Boxplots of Tip Percent for Male and Female (Below 70%)')
plt.xlabel('Sex')
plt.ylabel('Tip Percent')
plt.show()
```



In [12]:

```
#4.1
import pandas as pd
avocado_series = pd.read_csv(r"C:\Users\gabed\.ipython\avocado.csv")
df = pd.read_csv(r"C:\Users\gabed\.ipython\avocado.csv")
print(avocado_series)
missing_values = df.isnull()
missing_values_count = missing_values.sum()
```

```
print("Count of missing values per column:")
print(missing_values_count)
numeric_columns = df.select_dtypes(include=['number']).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())
missing_values_after_imputation = df.isnull().sum()
print("Count of missing values after mean imputation:")
print(missing_values_after_imputation)
text_columns = df.select_dtypes(include=['object']).columns
placeholder_value = "Unknown"  # Placeholder value for missing text or object values
df[text_columns] = df[text_columns].fillna(placeholder_value)
missing_values_after_imputation = df.isnull().sum()
print("Count of missing values after place holder imputation:")
print(missing_values_after_imputation)
```

```
             Date  AveragePrice  TotalVolume    Small      Large  AllSizes  \
0      2015-12-27          1.33     64236.62  1036.74   54454.85     48.16   
1      2015-12-20          1.35     54876.98   674.28   44638.81     58.33   
2      2015-12-13          0.93    118220.22   794.70  109149.67       NaN   
3      2015-12-06          1.08     78992.15  1132.00   71976.41     72.58   
4      2015-11-29          1.28     51039.60   941.48   43838.39     75.78   
...           ...           ...          ...      ...        ...       ...   
18244  2018-02-04          1.63     17074.83  2046.96    1529.20      0.00   
18245  2018-01-28          1.71     13888.04  1191.70    3431.50      0.00   
18246  2018-01-21          1.87     13766.76  1191.92    2452.79    727.94   
18247  2018-01-14          1.93     16205.22  1527.63    2981.04    727.01   
18248  2018-01-07          1.62     17489.58  2894.77    2356.13    224.53   

       TotalBags          Type    Year            Region  
0        8696.87  conventional  2015.0            Albany  
1        9505.56  conventional     NaN            Albany  
2        8145.35  conventional  2015.0            Albany  
3        5811.16  conventional  2015.0            Albany  
4        6183.95  conventional  2015.0            Albany  
...          ...           ...     ...               ...  
18244   13498.67       organic  2018.0  WestTexNewMexico  
18245    9264.84       organic  2018.0  WestTexNewMexico  
18246    9394.11       organic  2018.0  WestTexNewMexico  
18247   10969.54       organic  2018.0  WestTexNewMexico  
18248   12014.15       organic  2018.0  WestTexNewMexico  

[18249 rows x 10 columns]
Count of missing values per column:
Date            176
AveragePrice    184
TotalVolume     192
Small           194
Large           178
AllSizes        184
TotalBags       184
Type            204
Year            196
Region          169
dtype: int64
Count of missing values after mean imputation:
Date            176
AveragePrice      0
TotalVolume       0
Small             0
Large             0
AllSizes          0
TotalBags         0
Type            204
Year              0
Region          169
dtype: int64
Count of missing values after place holder imputation:
Date            0
AveragePrice    0
TotalVolume     0
Small           0
Large           0
AllSizes        0
TotalBags       0
Type            0
Year            0
Region          0
dtype: int64
```

In [14]:

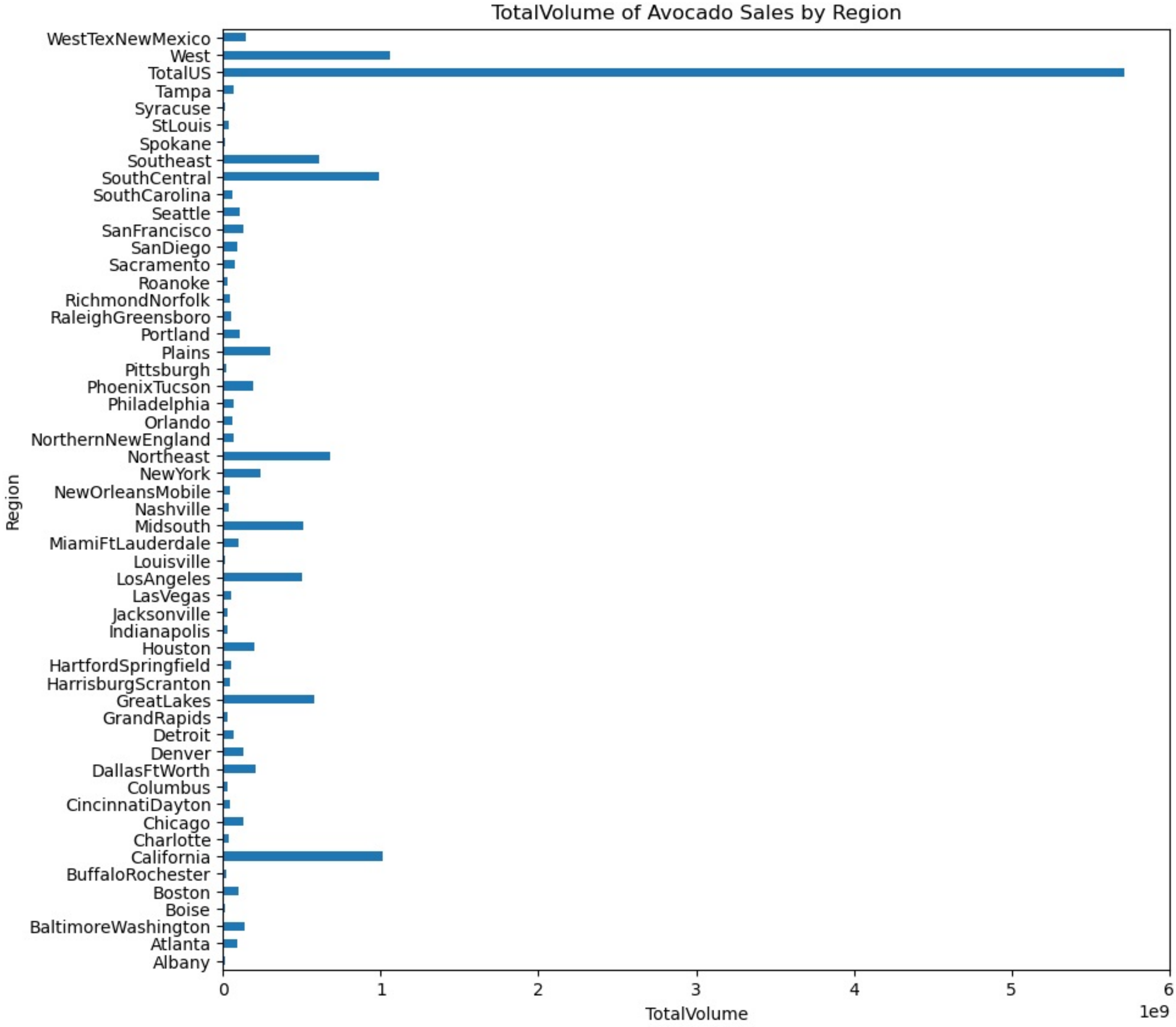
```
#4.2
import pandas as pd
avocado_series = pd.read_csv(r"C:\Users\gabed\.ipython\avocado.csv")
df = pd.read_csv(r"C:\Users\gabed\.ipython\avocado.csv")
df['Type'] = df['Type'].astype('category')
df['Year'] = df['Year'].astype('category')
df['Region'] = df['Region'].astype('category')
filtered_df = df[~df['Region'].isin(['TotalUS', 'West'])]
filtered_df = filtered_df.sort_values(by='Date')
mean_price_2016 = filtered_df[filtered_df['Year'] == 2016]['AveragePrice'].mean()
mean_price_2017 = filtered_df[filtered_df['Year'] == 2017]['AveragePrice'].mean()
```

```
if mean_price_2017 > mean_price_2016:
    print("The mean price of an avocado is higher in 2017 compared to 2016.")
elif mean_price_2017 < mean_price_2016:
    print("The mean price of an avocado is higher in 2016 compared to 2017.")
else:
    print("The mean price of an avocado is the same in 2016 and 2017.")
```

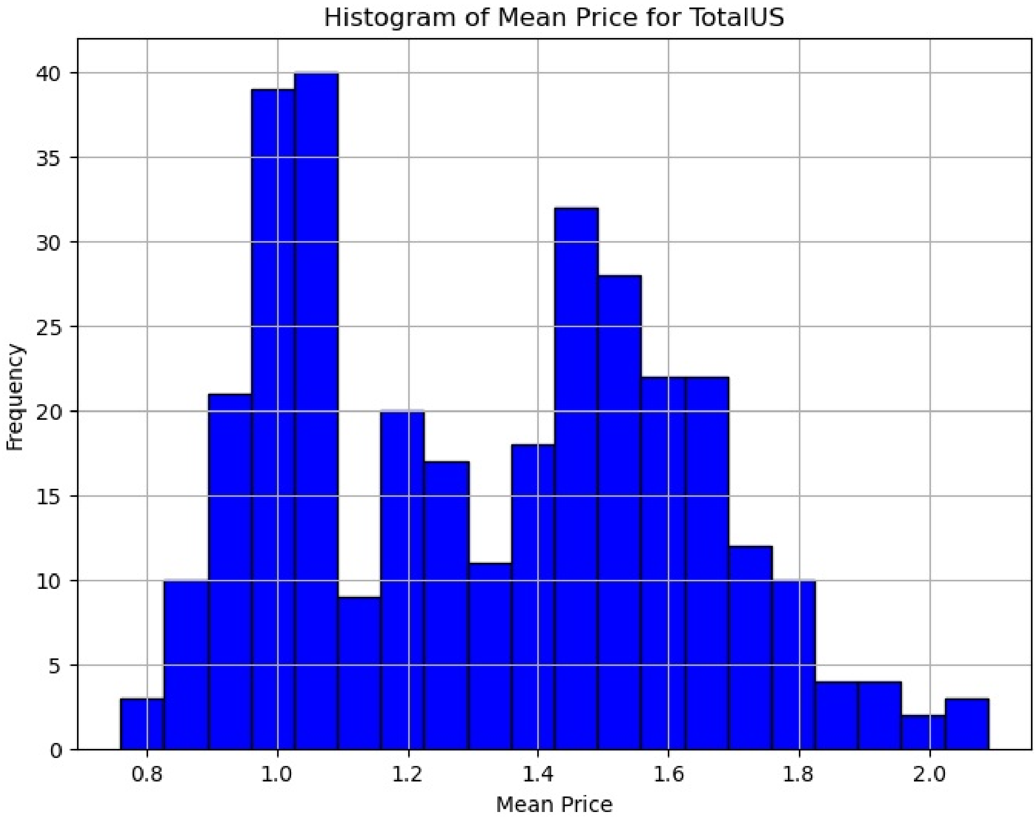
```
The mean price of an avocado is higher in 2017 compared to 2016.
```

In [33]:

```
#4.3
import pandas as pd
import matplotlib.pyplot as plt
avocado_series = pd.read_csv(r"C:\Users\gabed\.ipython\avocado.csv")
sales_by_Region = avocado_series.groupby('Region')['TotalVolume'].sum()
plt.figure(figsize=(10, 10))
sales_by_region.plot(kind='barh')
plt.title('TotalVolume of Avocado Sales by Region')
plt.xlabel('TotalVolume')
plt.ylabel('Region')
plt.show()
highest_sales_region = sales_by_region.idxmax()
print("State with the highest sales of avocados by volume:", highest_sales_region)
subset_state_data = avocado_series[avocado_series['Region'] == highest_sales_region]
plt.figure(figsize=(8, 6))
plt.hist(subset_state_data['AveragePrice'], bins=20, color='blue', edgecolor='black')
plt.title('Histogram of Mean Price for {}'.format(highest_sales_region))
plt.xlabel('Mean Price')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
correlation_index = subset_state_data['AveragePrice'].corr(subset_state_data['TotalVolume'])
print("Correlation index between mean price and total volume for state {}: {:.2f}".format(highest_sales_region,
```



```
State with the highest sales of avocados by volume: TotalUS
```



```
Correlation index between mean price and total volume for state TotalUS: -0.80
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js