Data Preparation Week 3 & 4

April 7, 2024

1 Activity 3.01

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Load the Boston Housing dataset
df = pd.read_csv('/Users/nickblackford/Desktop/Python/Boston_housing.csv')
```

[2]:
```
# Check the first 10 records
print(df.head(10))
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM    AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575   65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421   78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185   61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998   45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147   54.2  6.0622    3  222     18.7
5  0.02985   0.0   2.18     0  0.458  6.430   58.7  6.0622    3  222     18.7
6  0.08829  12.5   7.87     0  0.524  6.012   66.6  5.5605    5  311     15.2
7  0.14455  12.5   7.87     0  0.524  6.172   96.1  5.9505    5  311     15.2
8  0.21124  12.5   7.87     0  0.524  5.631  100.0  6.0821    5  311     15.2
9  0.17004  12.5   7.87     0  0.524  6.004   85.9  6.5921    5  311     15.2

        B  LSTAT  PRICE
0  396.90   4.98   24.0
1  396.90   9.14   21.6
2  392.83   4.03   34.7
3  394.63   2.94   33.4
4  396.90   5.33   36.2
5  394.12   5.21   28.7
6  395.60  12.43   22.9
7  396.90  19.15   27.1
8  386.63  29.93   16.5
9  386.71  17.10   18.9
```

[3]:
```
# Find the total number of records
len(df)
```

[3]:
```
506
```

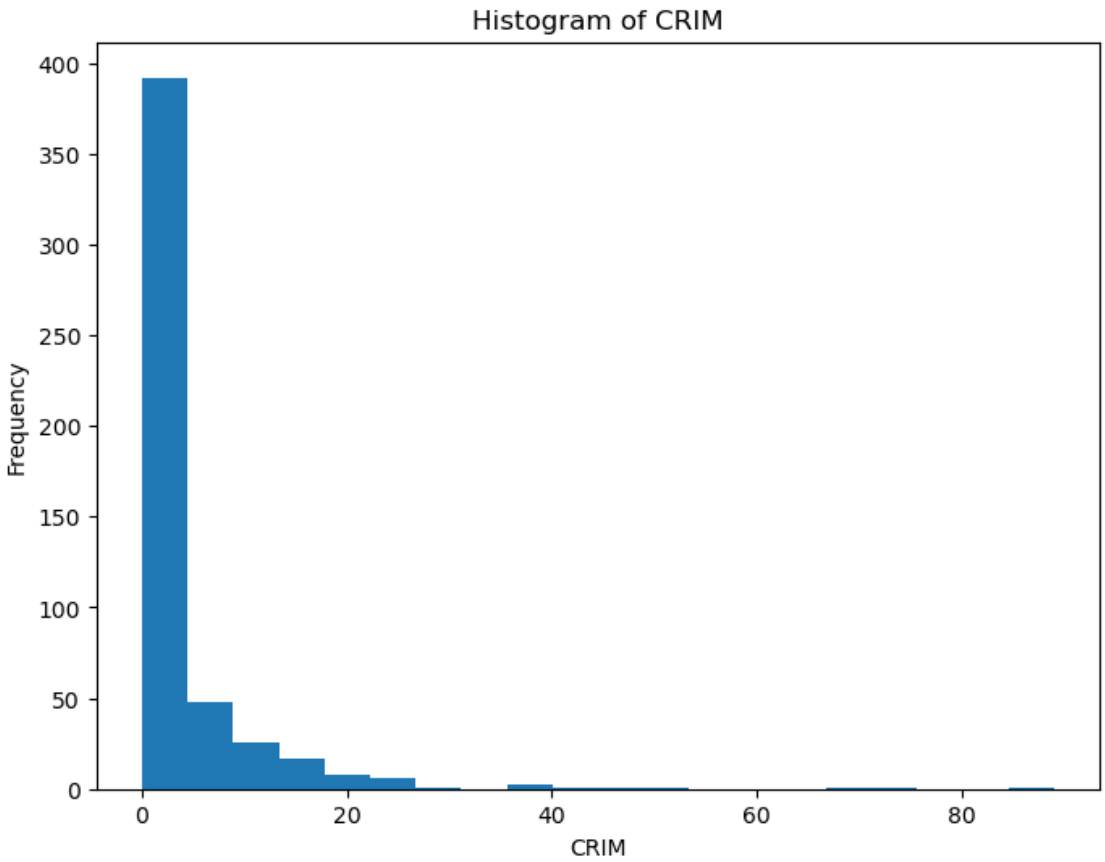
[4]:
```
# Create a smaller DataFrame without CHAS, NOX, B, and LSTAT
df_small = df.drop(columns=['CHAS', 'NOX', 'B', 'LSTAT'])
```

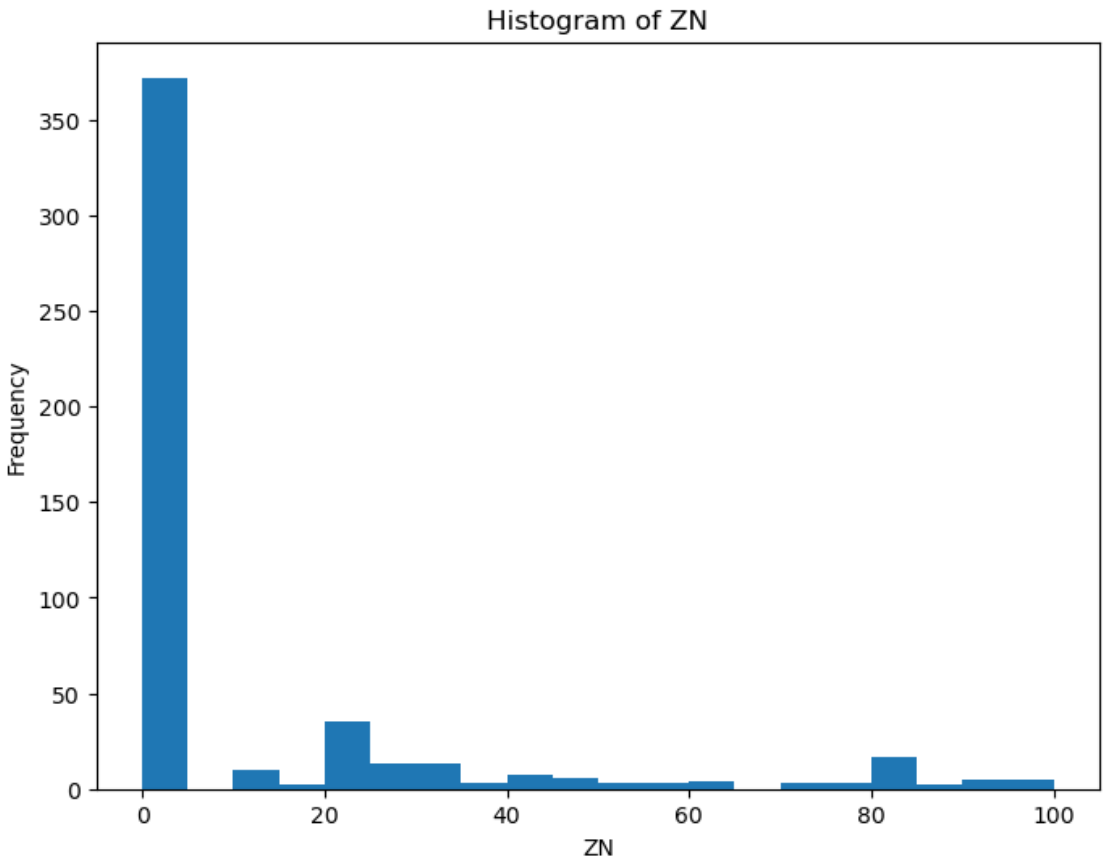
[5]:
```
df_small.tail(7)
```

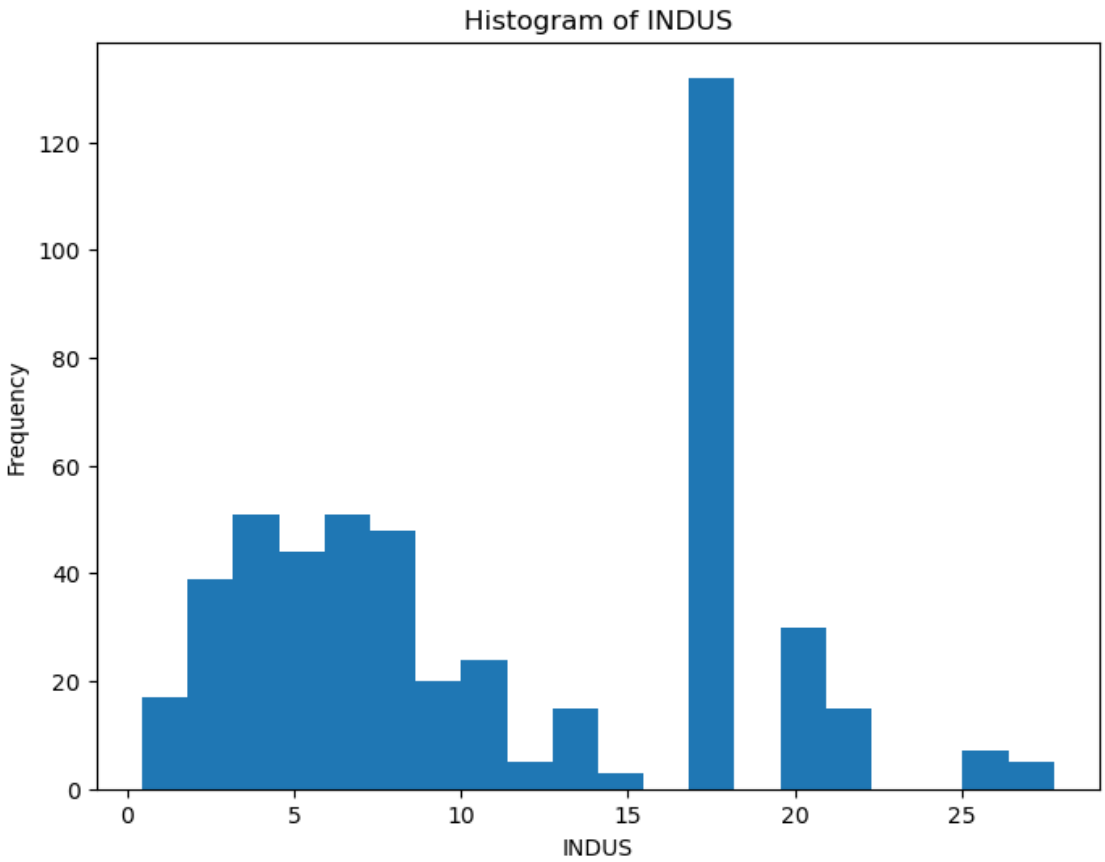
[5]:

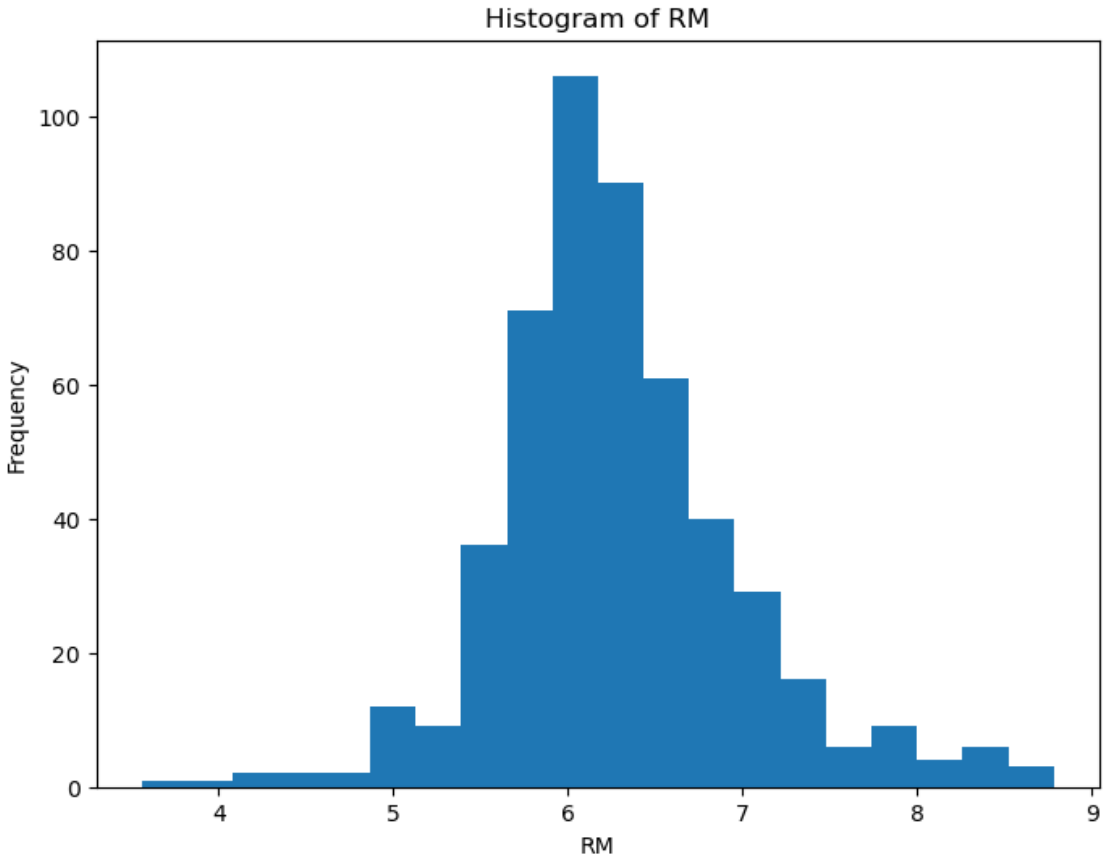
	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
499	0.17783	0.0	9.69	5.569	73.5	2.3999	6	391	19.2	17.5
500	0.22438	0.0	9.69	6.027	79.7	2.4982	6	391	19.2	16.8
501	0.06263	0.0	11.93	6.593	69.1	2.4786	1	273	21.0	22.4
502	0.04527	0.0	11.93	6.120	76.7	2.2875	1	273	21.0	20.6
503	0.06076	0.0	11.93	6.976	91.0	2.1675	1	273	21.0	23.9
504	0.10959	0.0	11.93	6.794	89.3	2.3889	1	273	21.0	22.0
505	0.04741	0.0	11.93	6.030	80.8	2.5050	1	273	21.0	11.9

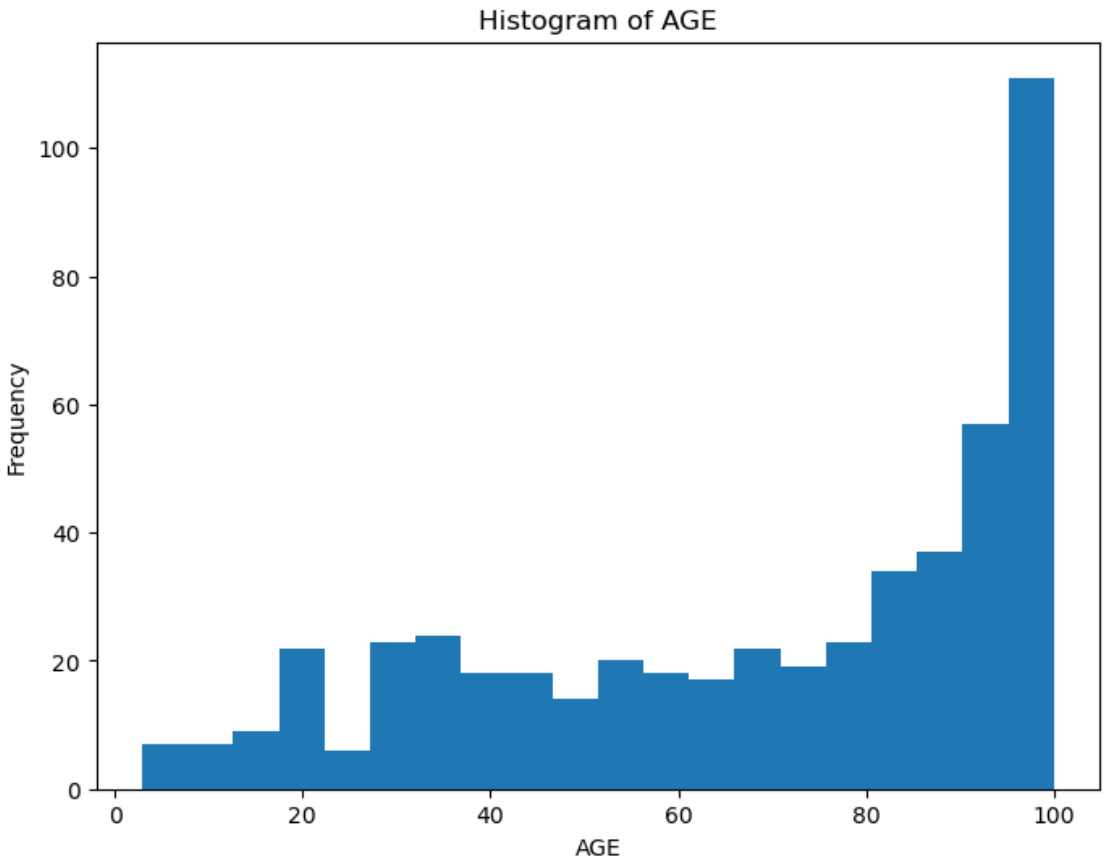
[6]:
```
# Plot the histograms of all the variables in the new DataFrame
for column in df_small.columns:
    plt.figure(figsize=(8, 6))
    plt.hist(df_small[column], bins=20)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

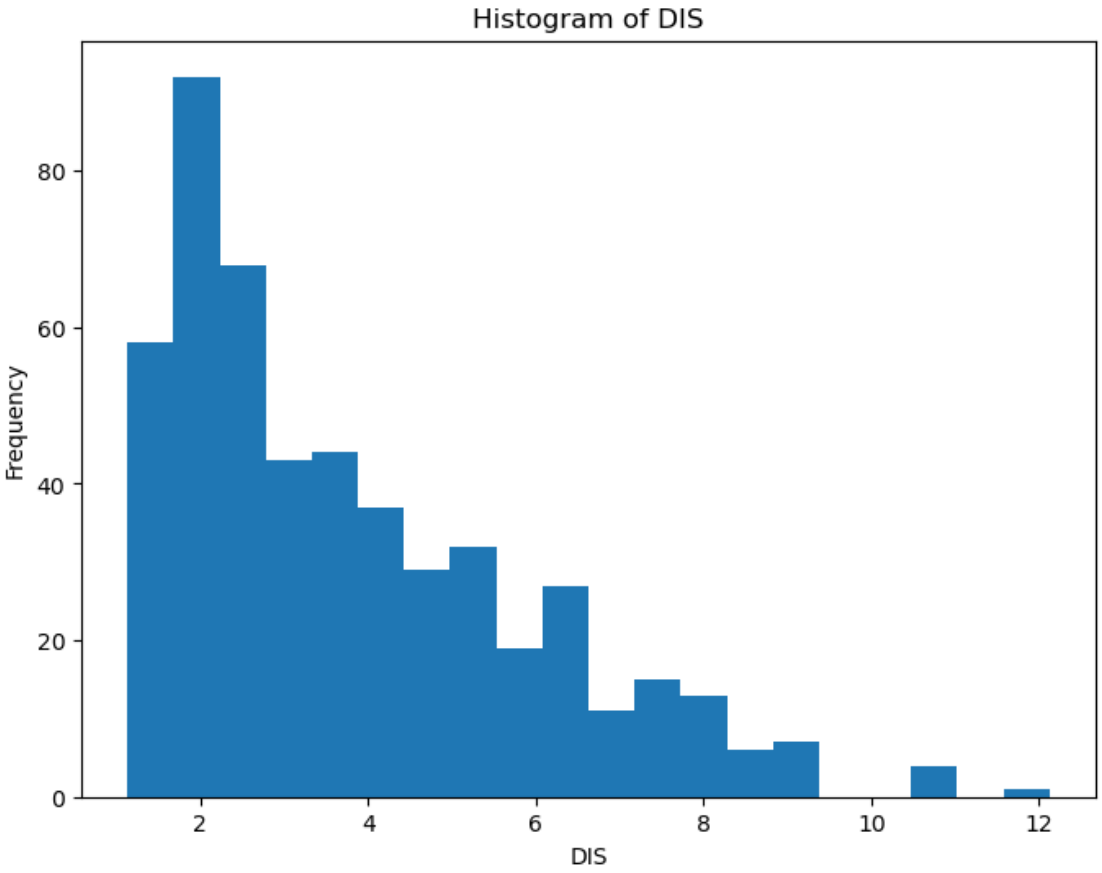

Histogram of CRIM
Frequency
CRIM
0
50
100
150
200
250
300
350
400
0
20
40
60
80

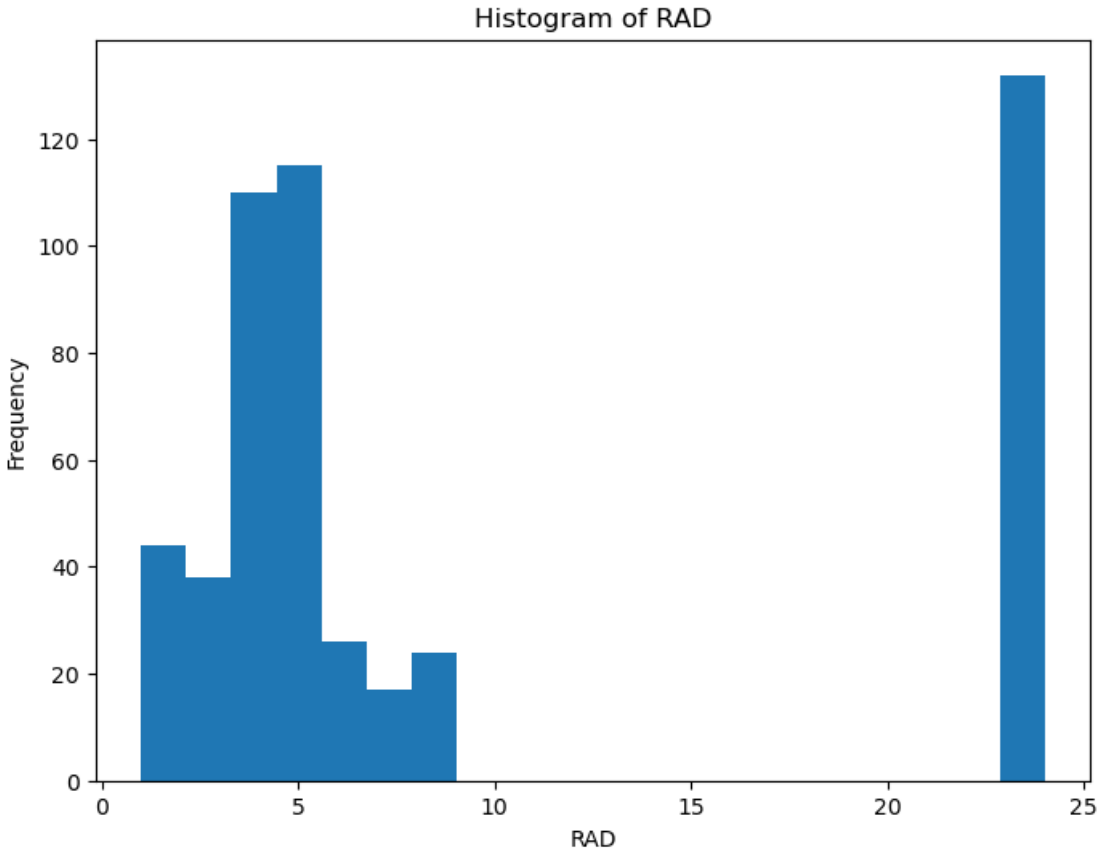

Histogram of ZN
Frequency
350
300
250
200
150
100
50
0
0
20
40
60
80
100
ZN

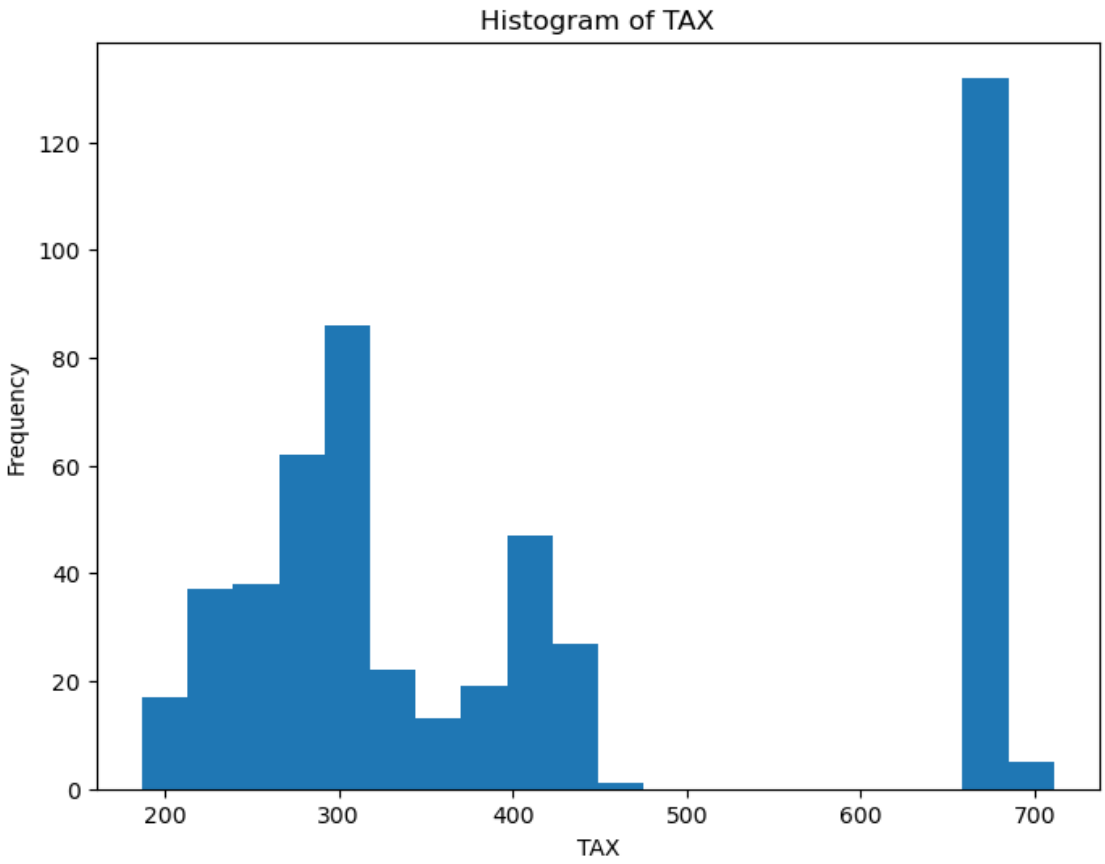

Histogram of INDUS
Frequency
INDUS
0
20
40
60
80
100
120
0
5
10
15
20
25

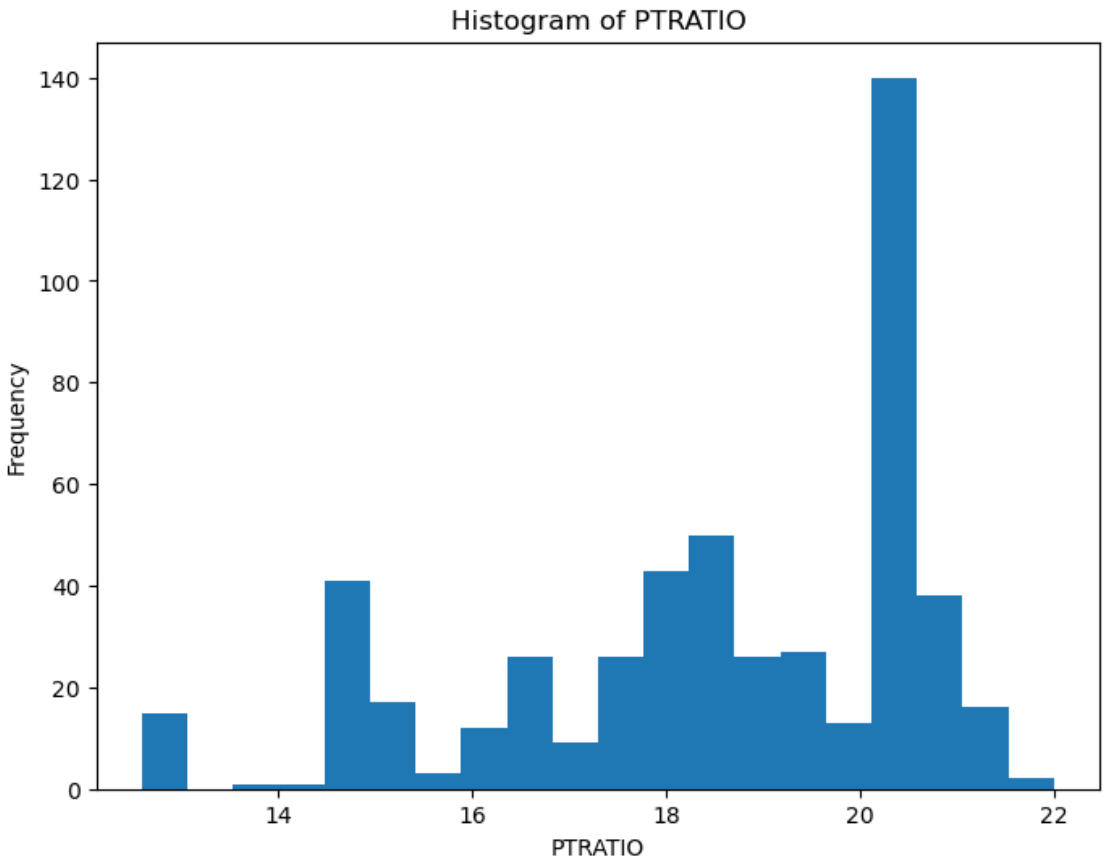

Histogram of RM
Frequency
RM
0
20
40
60
80
100
4
5
6
7
8
9

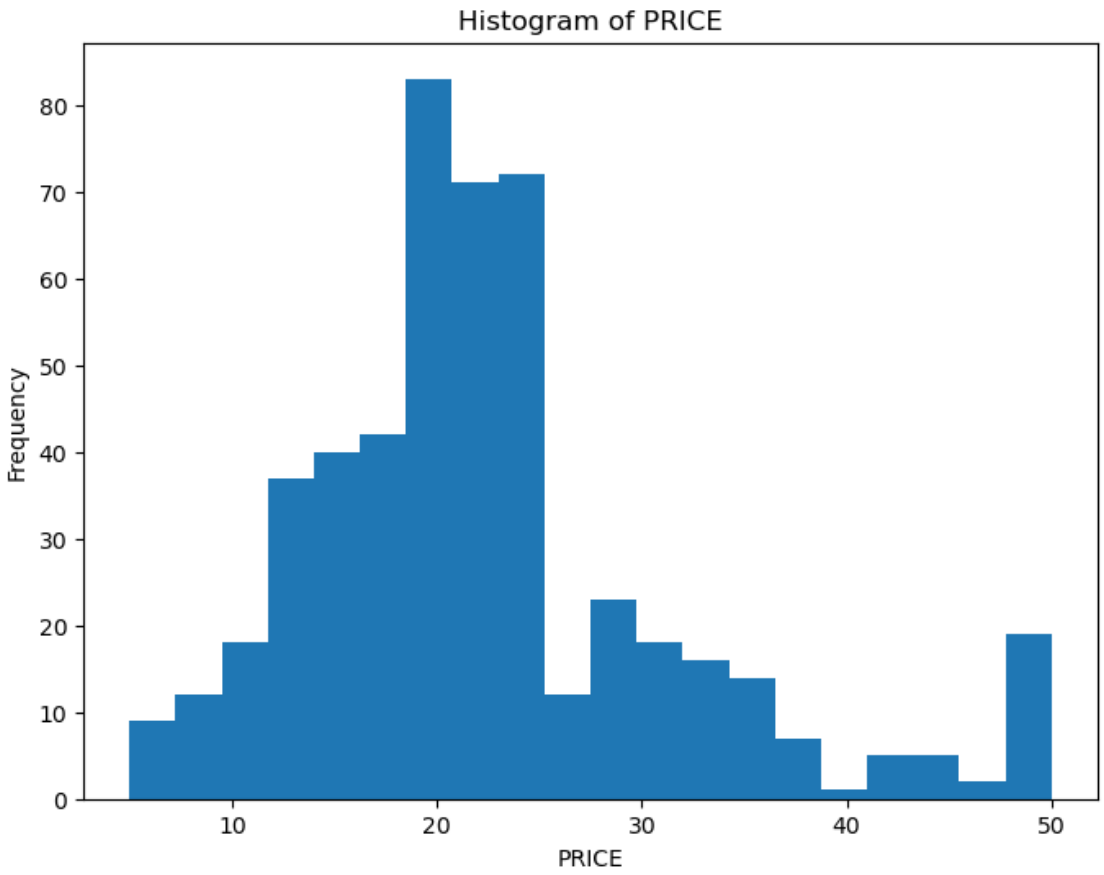

Histogram of AGE
Frequency
AGE
0
20
40
60
80
100


Histogram of DIS
Frequency
0
20
40
60
80
2
4
6
8
10
12
DIS

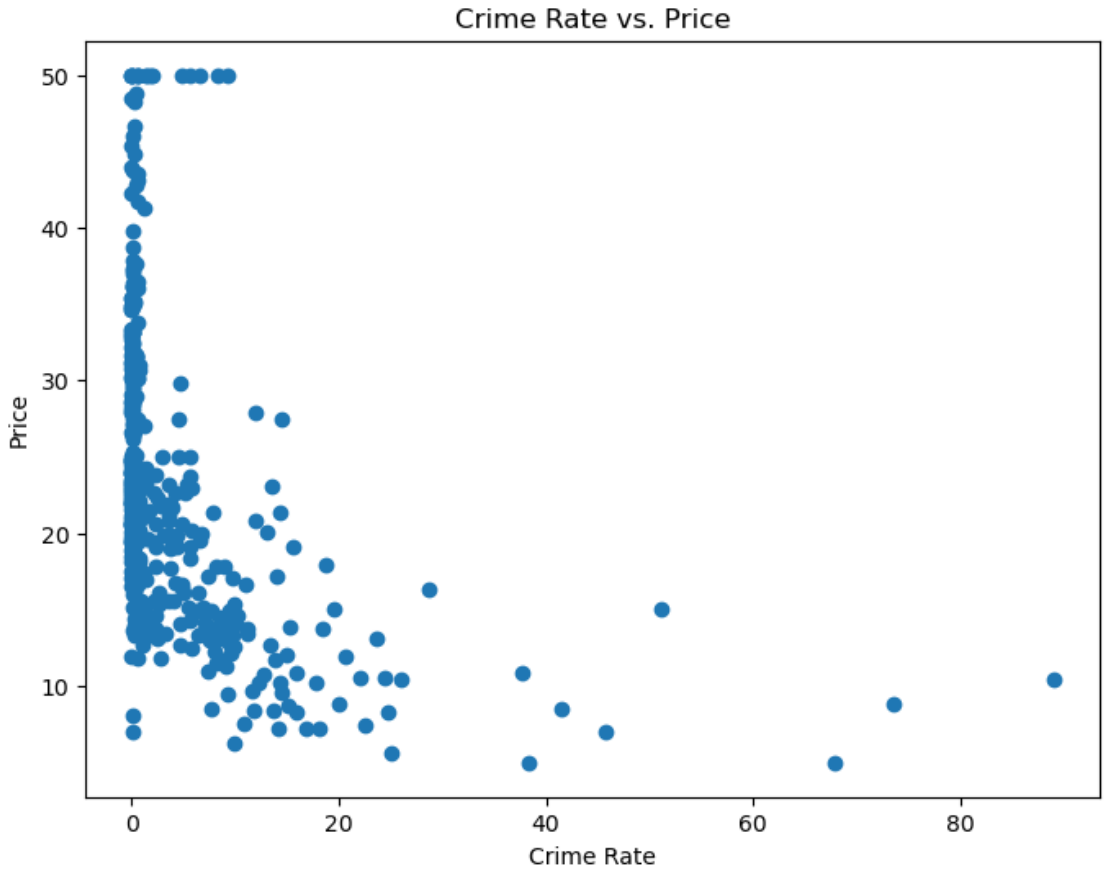

Histogram of RAD
Frequency
RAD
0
20
40
60
80
100
120
0
5
10
15
20
25


Histogram of TAX
Frequency
0
20
40
60
80
100
120
200
300
400
500
600
700
TAX

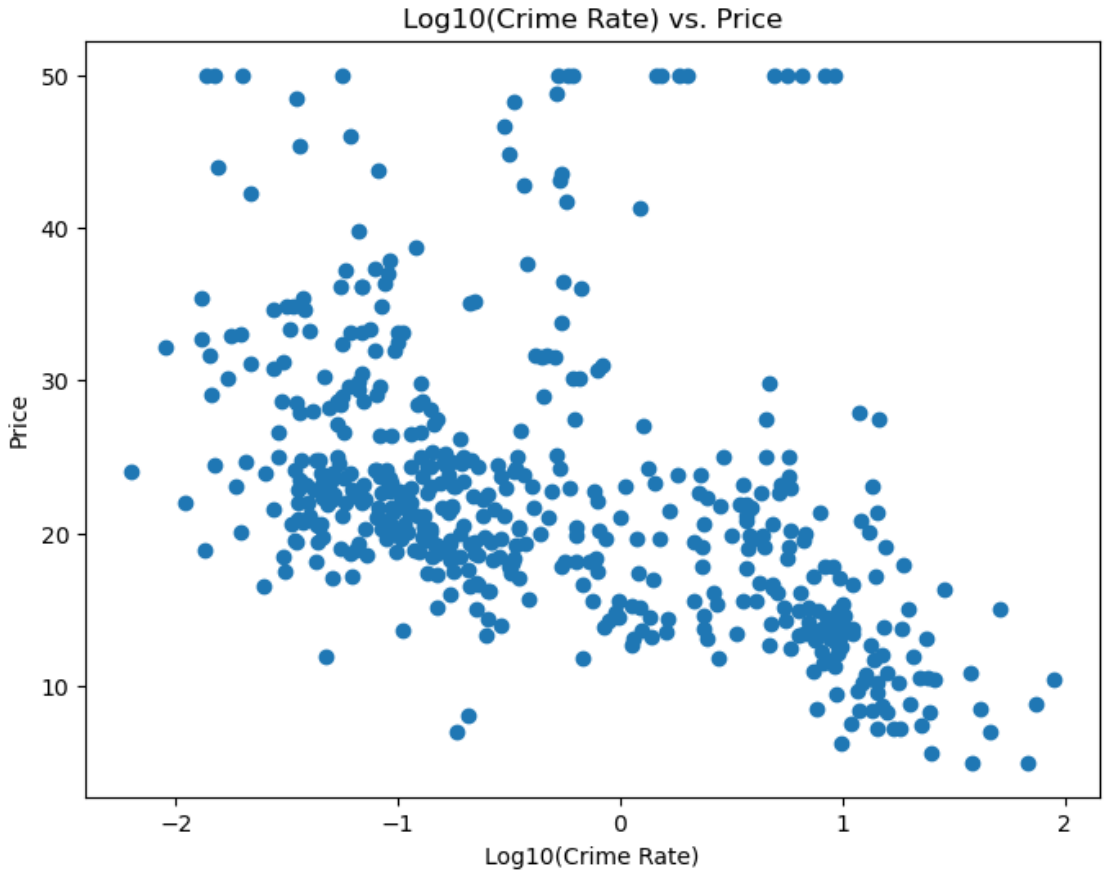

Histogram of PTRATIO
Frequency
PTRATIO
0
20
40
60
80
100
120
140
14
16
18
20
22



[7]:
```
# Create a scatter plot of crime rate versus price
plt.figure(figsize=(8, 6))
plt.scatter(df['CRIM'], df['PRICE'])
plt.xlabel('Crime Rate')
plt.ylabel('Price')
plt.title('Crime Rate vs. Price')
plt.show()
```



[8]:
```
# Plot log10(crime) versus price
plt.figure(figsize=(8, 6))
plt.scatter(np.log10(df['CRIM']), df['PRICE'])
plt.xlabel('Log10(Crime Rate)')
plt.ylabel('Price')
plt.title('Log10(Crime Rate) vs. Price')
plt.show()
```



[9]:
```
# Calculate useful statistics
mean_rooms = df['RM'].mean()
median_age = df['AGE'].median()
mean_distance = df['DIS'].mean()
percentage_low_price = (df['PRICE'] < 20).mean() * 100

print("Mean rooms per dwelling:", mean_rooms)
print("Median age:", median_age)
print("Mean distances to five Boston employment centers:", mean_distance)
print("Percentage of houses with a low price (<$20,000):", percentage_low_price)
```

```
Mean rooms per dwelling: 6.284634387351779
Median age: 77.5
Mean distances to five Boston employment centers: 3.795042687747036
Percentage of houses with a low price (<$20,000): 41.50197628458498
```

2 Activity 4.01

[35]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the necessary libraries and read the adult income dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.
 ↪data"
column_names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num',␣
 ↪'marital-status', 'occupation',
                'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',␣
 ↪'hours-per-week', 'native-country', 'Income']
df2 = pd.read_csv(url, names=column_names)
```

[36]:
```
# Create a script that will read a text file line by line
with open('adult_income_data.txt', 'w') as f:
    for line in df.to_string(index=False):
        f.write(line)
```

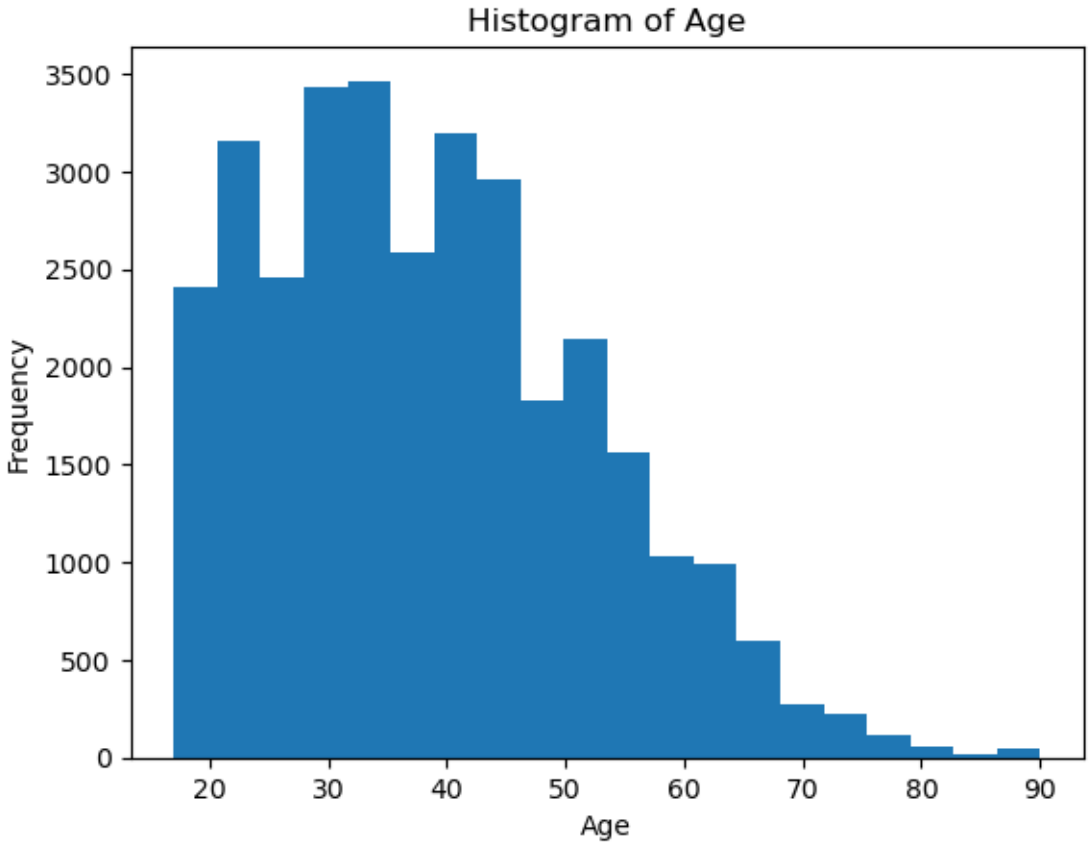
[37]:
```
# Find the missing values
print("Missing values:\n", df2.isnull().sum())
```

```
Missing values:
 age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
Income            0
dtype: int64
```

[38]:
```
# Create a DataFrame with only age, education, and occupation by using␣
 ↪subsetting
df_subset = df2[['age', 'education', 'occupation']]
```

[39]:
```
# Plot a histogram of age with a bin size of 20
plt.hist(df2['age'], bins=20)
```

```
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Histogram of Age')
plt.show()
```



[40]:
```
# Create a function to strip the whitespace characters
def strip_whitespace(s):
    return s.strip()
```

[41]:
```
# Use the apply method to apply this function to all the columns with string␣
 ↪values
df2 = df2.applymap(lambda x: strip_whitespace(x) if type(x) == str else x)
```

[42]:
```
# Find the number of people who are aged between 30 and 50
print("Number of people aged between 30 and 50:", df2[(df2['age'] >= 30) &␣
 ↪(df2['age'] <= 50)].shape[0])
```

```
Number of people aged between 30 and 50: 16390
```

[44]:
```
# Group the records based on age and education to find how the mean age is␣
↪distributed
numeric_cols = df2.select_dtypes(include=[np.number]).columns
print("Mean age distribution by age and education:\n", df2.groupby(['age',␣
↪'education'])[numeric_cols].mean())
```

```
Mean age distribution by age and education:
                     age         fnlwgt  education-num  capital-gain  \
age education
17  10th            17.0  187111.630435            6.0    266.659420
    11th            17.0  188696.555556            7.0     30.411111
    12th            17.0  178750.702703            8.0      0.000000
    5th-6th         17.0  270942.000000            3.0      0.000000
    7th-8th         17.0  127804.000000            4.0      0.000000
…                    …            …              …            …
90  Bachelors       90.0  165312.333333           13.0   2327.000000
    HS-grad         90.0  132201.285714            9.0   1394.142857
    Masters         90.0  150378.250000           14.0   5012.750000
    Prof-school     90.0   87372.000000           15.0  20051.000000
    Some-college    90.0  153924.333333           10.0      0.000000

                    capital-loss  hours-per-week
age education
17  10th               46.434783       21.543478
    11th               36.911111       19.927778
    12th               46.513514       20.189189
    5th-6th             0.000000       48.000000
    7th-8th             0.000000       31.000000
…                          …              …
90  Bachelors           0.000000       31.666667
    HS-grad           468.714286       37.428571
    Masters             0.000000       47.500000
    Prof-school         0.000000       72.000000
    Some-college        0.000000       32.833333

[965 rows x 6 columns]
```

[45]:
```
# Group by occupation and show the summary statistics of age
occupation_stats = df2.groupby('occupation')['age'].describe()
print("Summary statistics of age by occupation:\n", occupation_stats)
```

```
Summary statistics of age by occupation:
                    count       mean        std   min   25%   50%   75%   max
occupation
?                  1843.0  40.882800  20.336350  17.0  21.0  35.0  61.0  90.0
Adm-clerical       3770.0  36.964456  13.362998  17.0  26.0  35.0  46.0  90.0
Armed-Forces          9.0  30.222222   8.089774  23.0  24.0  29.0  34.0  46.0
```

```
Craft-repair       4099.0  39.031471  11.606436  17.0  30.0  38.0  47.0  90.0
Exec-managerial    4066.0  42.169208  11.974548  17.0  33.0  41.0  50.0  90.0
Farming-fishing     994.0  41.211268  15.070283  17.0  29.0  39.0  52.0  90.0
Handlers-cleaners  1370.0  32.165693  12.372635  17.0  23.0  29.0  39.0  90.0
Machine-op-inspct  2002.0  37.715285  12.068266  17.0  28.0  36.0  46.0  90.0
Other-service      3295.0  34.949621  14.521508  17.0  22.0  32.0  45.0  90.0
Priv-house-serv     149.0  41.724832  18.633688  17.0  24.0  40.0  57.0  81.0
Prof-specialty     4140.0  40.517633  12.016676  17.0  31.0  40.0  48.0  90.0
Protective-serv     649.0  38.953775  12.822062  17.0  29.0  36.0  47.0  90.0
Sales              3650.0  37.353973  14.186352  17.0  25.0  35.0  47.0  90.0
Tech-support        928.0  37.022629  11.316594  17.0  28.0  36.0  44.0  73.0
Transport-moving   1597.0  40.197871  12.450792  17.0  30.0  39.0  49.0  90.0
```

[46]:
```
# Find which profession has the oldest workers on average
print("Profession with the oldest workers on average:",␣
 ↪occupation_stats['mean'].idxmax())
```

```
Profession with the oldest workers on average: Exec-managerial
```

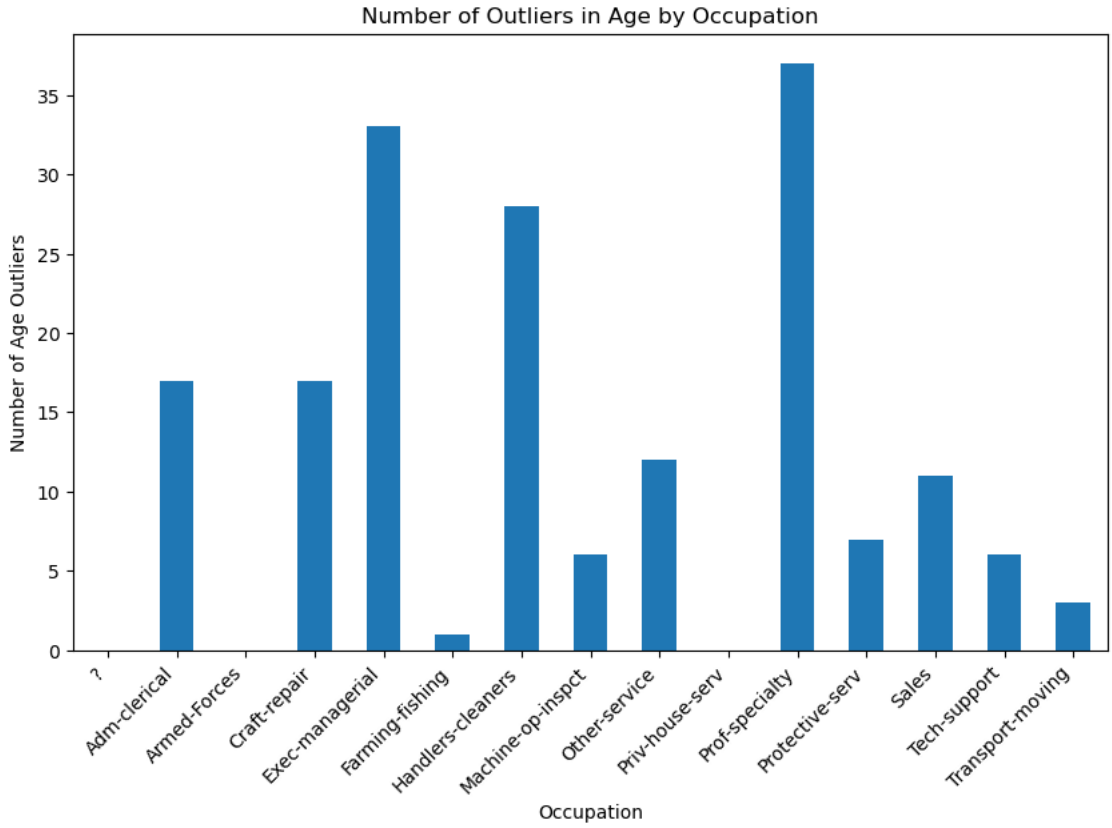
[48]:
```
# Find which profession has its largest share of the workforce above the 75th␣
 ↪percentile
print("Profession with largest share of workforce above 75th percentile:",␣
 ↪occupation_stats['75%'].idxmax())
```

```
Profession with largest share of workforce above 75th percentile: ?
```

[53]:
```
def count_outliers(group):
    q1 = group.quantile(0.25)
    q3 = group.quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = group[(group < lower_bound) | (group > upper_bound)]
    return len(outliers)

# Group by occupation and count the number of outliers in age for each group
outlier_counts = df2.groupby('occupation')['age'].apply(count_outliers)
```

[54]:
```
# Plot the number of outliers in each job category
outlier_counts.plot(kind='bar', figsize=(10, 6))
plt.xlabel('Occupation')
plt.ylabel('Number of Age Outliers')
plt.title('Number of Outliers in Age by Occupation')
plt.xticks(rotation=45, ha='right')
plt.show()
```



[55]:
```
# Merge the two datasets using common keys (columns)
merged_df = pd.merge(df2, df_subset, how='inner', on=['age', 'education',␣
 ↪'occupation'])

# Drop duplicate values
merged_df.drop_duplicates(inplace=True)
```

[55]:
```
Empty DataFrame
Columns: [age, workclass, fnlwgt, education, education-num, marital-status,
occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week,
native-country, Income]
Index: []
```

3 3. Create a series

[56]:
```
# Create Series 1
series1 = pd.Series([7.3, -2.5, 3.4, 1.5], index=['a', 'c', 'd', 'e'])

# Create Series 2
```

```
series2 = pd.Series([-2.1, 3.6, -1.5, 4, 3.1], index=['a', 'c', 'e', 'f', 'g'])
```

[60]:
```
# Add/Subtract Series 1 and Series 2
add_result = series1.add(series2, fill_value=0)
sub_result = series2.subtract(series1, fill_value=0)
```

[61]:
```
# Print the results
print("Addition of Series 1 and Series 2:\n", add_result)
print("\nSubtraction of Series 1 from Series 2:\n", sub_result)
```

```
Addition of Series 1 and Series 2:
 a    5.2
c    1.1
d    3.4
e    0.0
f    4.0
g    3.1
dtype: float64

Subtraction of Series 1 from Series 2:
 a   -9.4
c    6.1
d   -3.4
e   -3.0
f    4.0
g    3.1
dtype: float64
```