Student Name: Balakrishna Mupparaju

Assignment: Week 3 & 4

1. The Data Wrangling Workshop: Activity 3.01, page 155

Suppose you are working with the Boston Housing price dataset. This dataset is famous in the machine learning community. Many regression problems can be formulated, and machine learning algorithms can be run on this dataset. You will perform a basic data wrangling activity (including plotting some trends) on this dataset (.csv file) by reading it as a pandas DataFrame. We will perform a few statistical operations on this data frame.

In [4]:

```
#import required libraries
"""NumPy (numpy): Used for numerical operations and advanced mathematical cc
Pandas (pandas): A library for data manipulation and analysis. Helps in work
Matplotlib (matplotlib.pyplot): A plotting library to visualize the data in 

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [5]:

```
#Reads the CSV file containing the Boston Housing dataset using pandas and s
df=pd.read_csv("/Users/balakrishnamupparaju/Downloads/Boston_housing.csv")
```

In [8]:

```
#Displays the first 10 rows of the dataset, helping you understand its struc
df.head(10)
```

Out[8]:

	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

In [10]:

```python
#Returns the dimensions of the DataFrame (rows, columns), indicating the tot
df.shape
```

Out[10]:

```
(506, 14)
```

In [12]:

```python
#create a new DataFrame (df1) that contains a subset of columns from the ori
df1=df[['CRIM','ZN','INDUS','RM','AGE','DIS',
'RAD','TAX','PTRATIO','PRICE']]
```

In [14]:

```python
#Shows the last 7 rows of the DataFrame df1.
df1.tail(7)
```

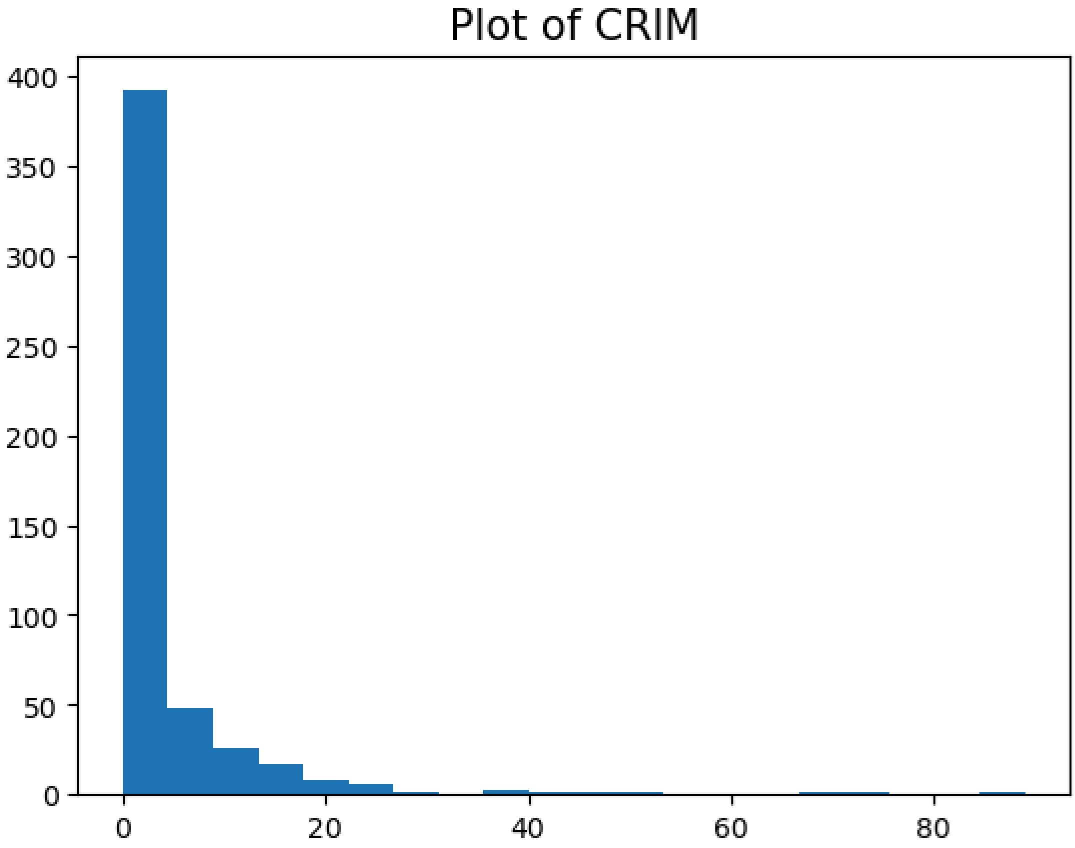
Out[14]:

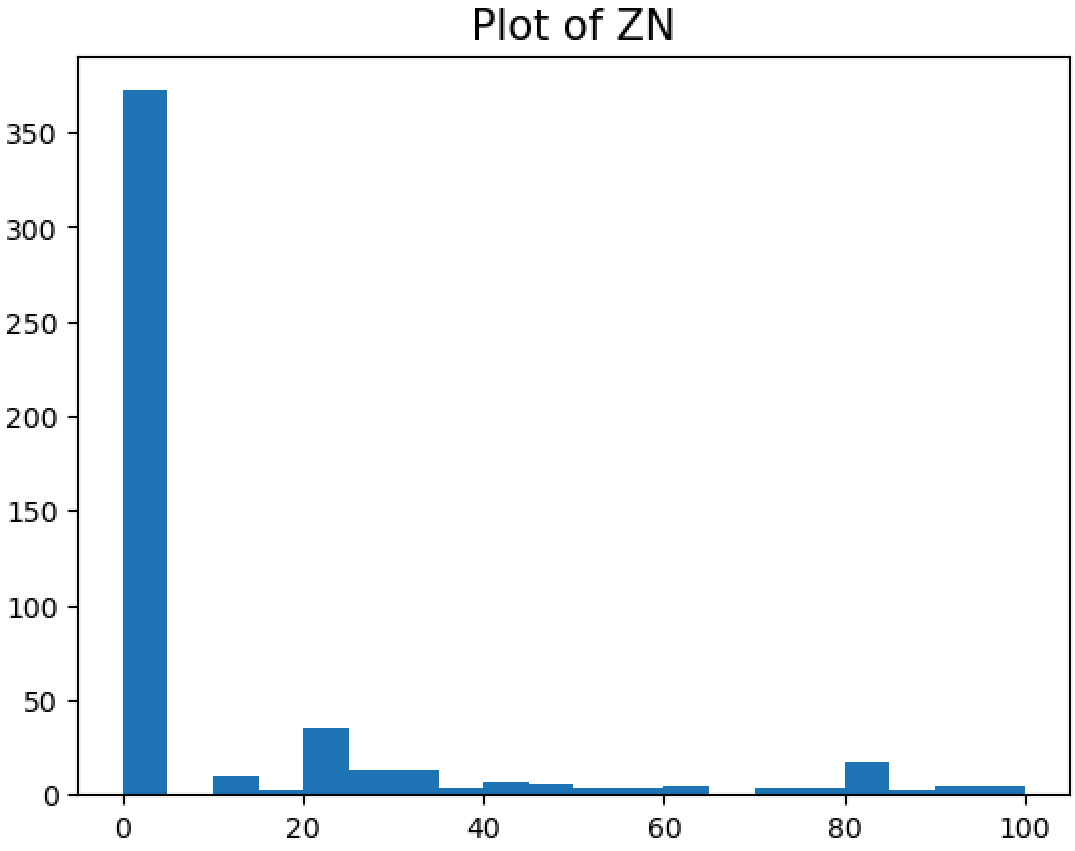
	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
499	0.17783	0.0	9.69	5.569	73.5	2.3999	6	391	19.2	17.5
500	0.22438	0.0	9.69	6.027	79.7	2.4982	6	391	19.2	16.8
501	0.06263	0.0	11.93	6.593	69.1	2.4786	1	273	21.0	22.4
502	0.04527	0.0	11.93	6.120	76.7	2.2875	1	273	21.0	20.6
503	0.06076	0.0	11.93	6.976	91.0	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

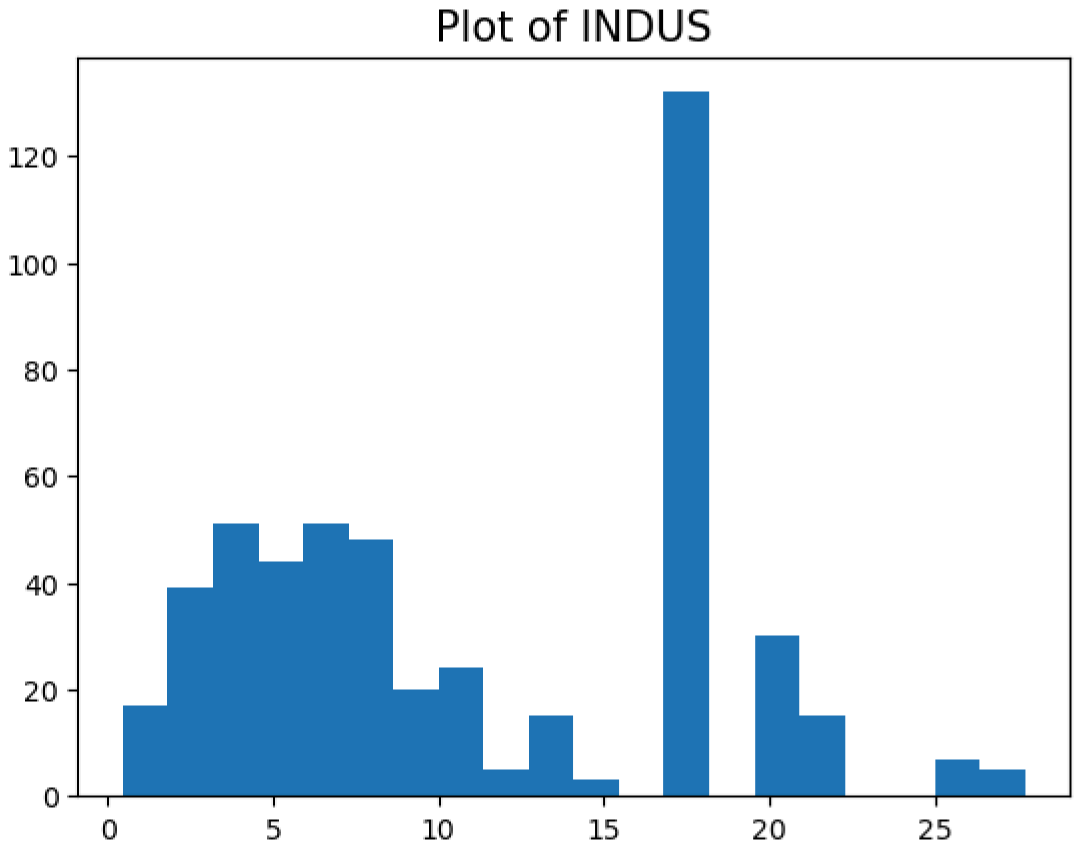
In [16]:

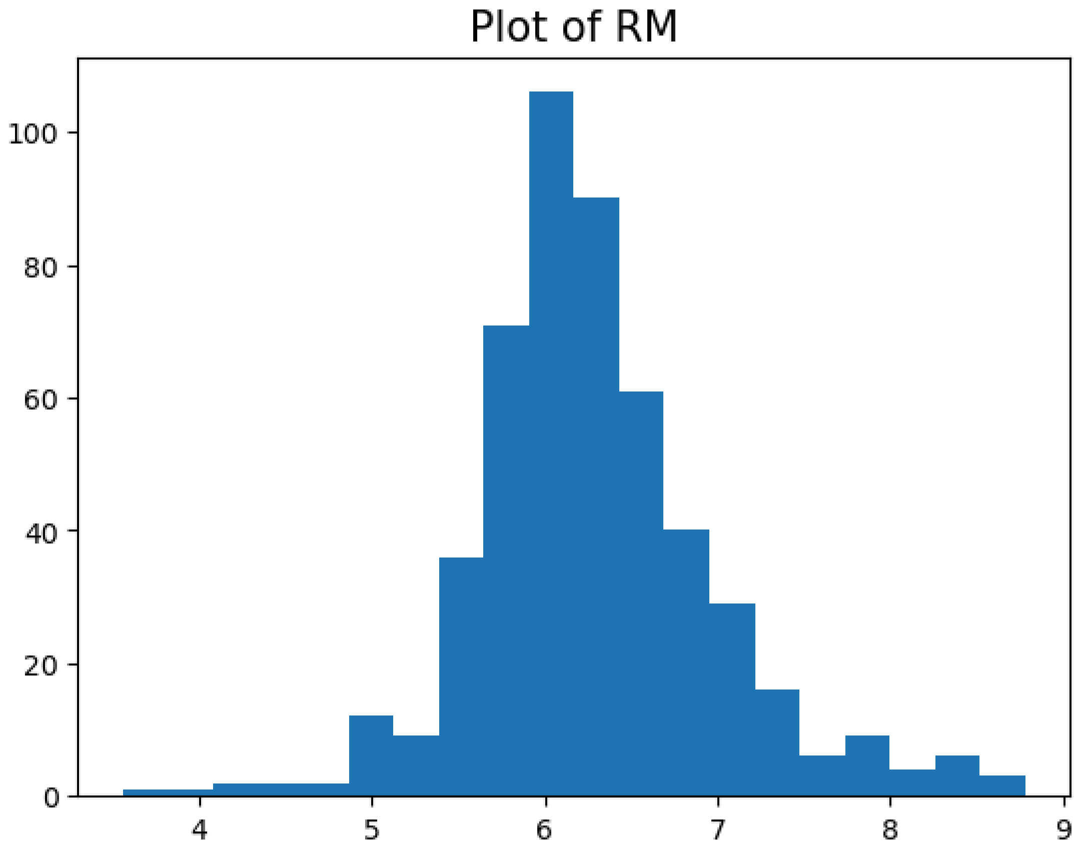
```python
"""Iterates through all the columns in the DataFrame.

For each column, a histogram is plotted to visualize the distribution of dat
plt.title: Sets the title for each plot.
plt.hist: Creates a histogram with 20 bins (i.e., dividing the data range ir
plt.show(): Displays the histogram plot.
"""
```

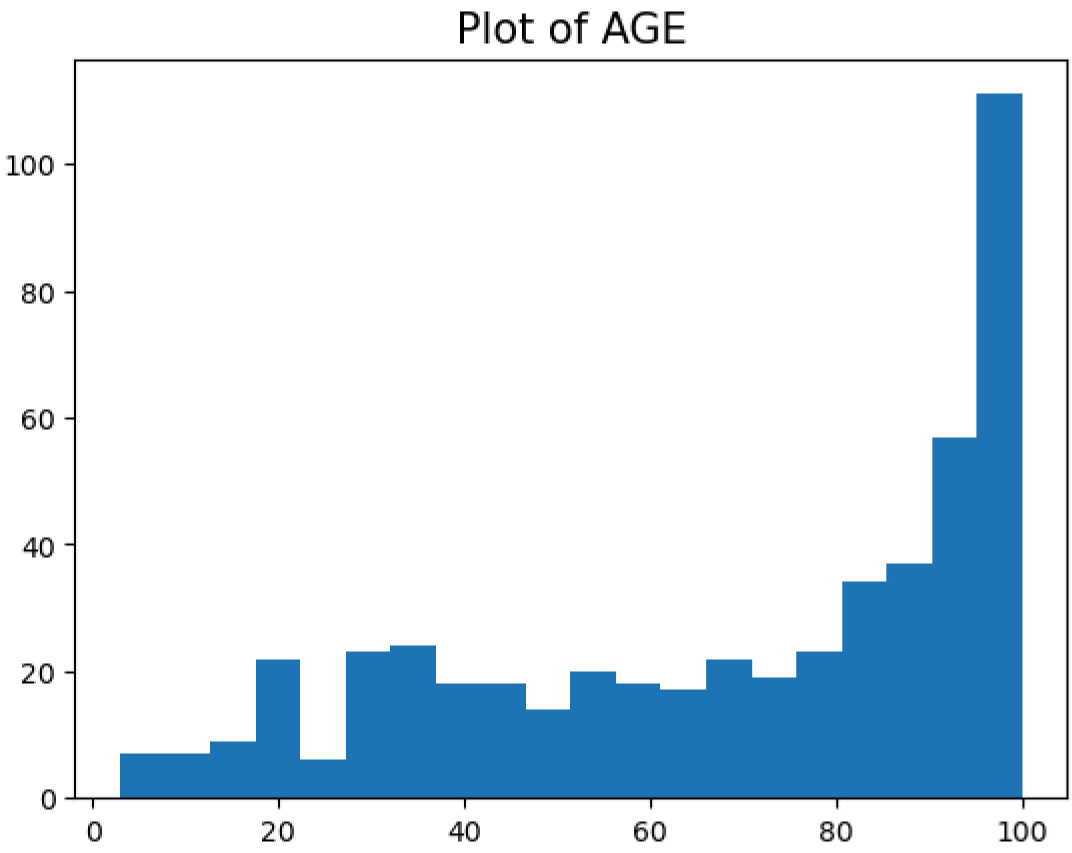
```
for c in df1.columns:
    plt.title("Plot of "+c,fontsize=15)
    plt.hist(df1[c],bins=20)
    plt.show()
```

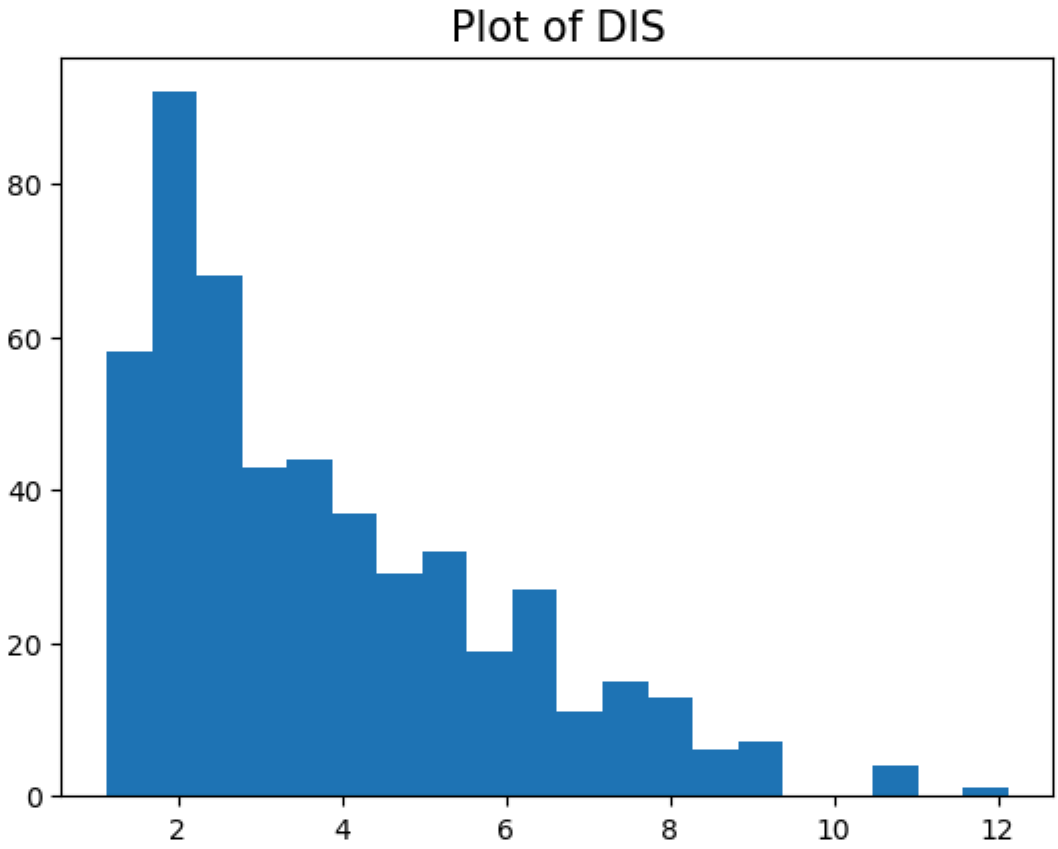

Plot of CRIM

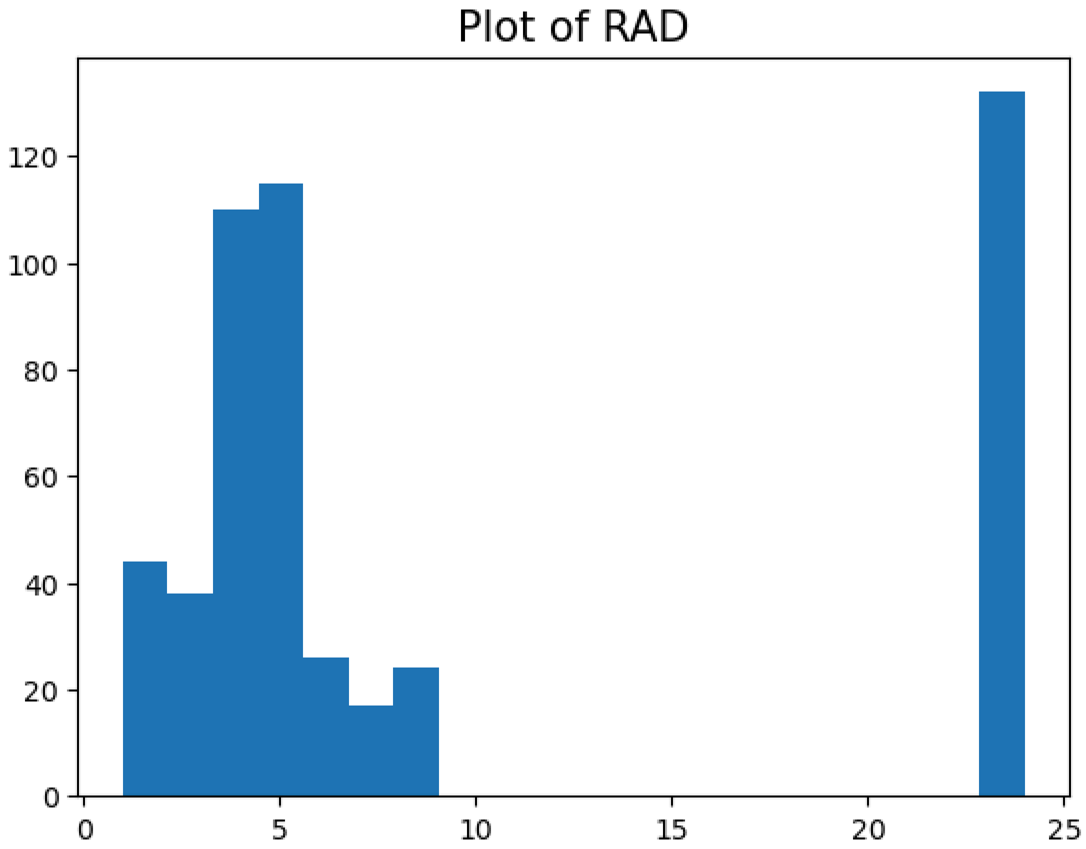

Plot of ZN


Plot of INDUS

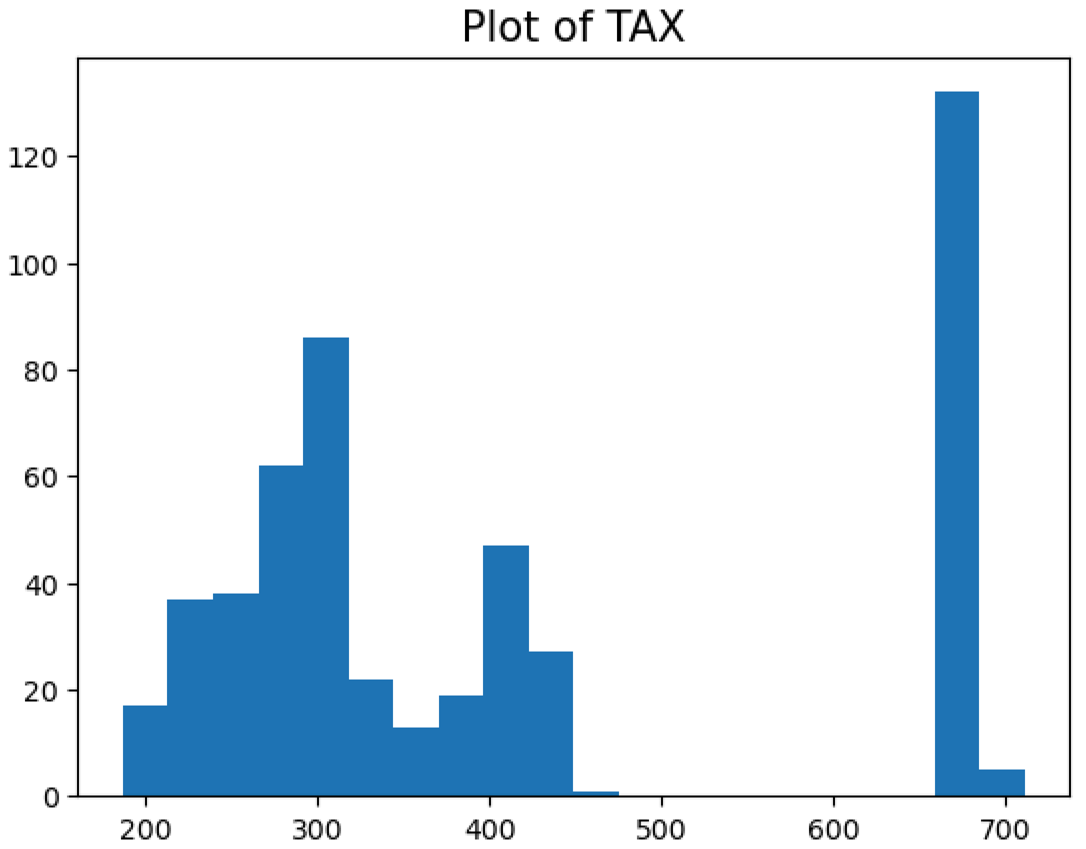

Plot of RM


Plot of AGE
0
20
40
60
80
100
0
20
40
60
80
100

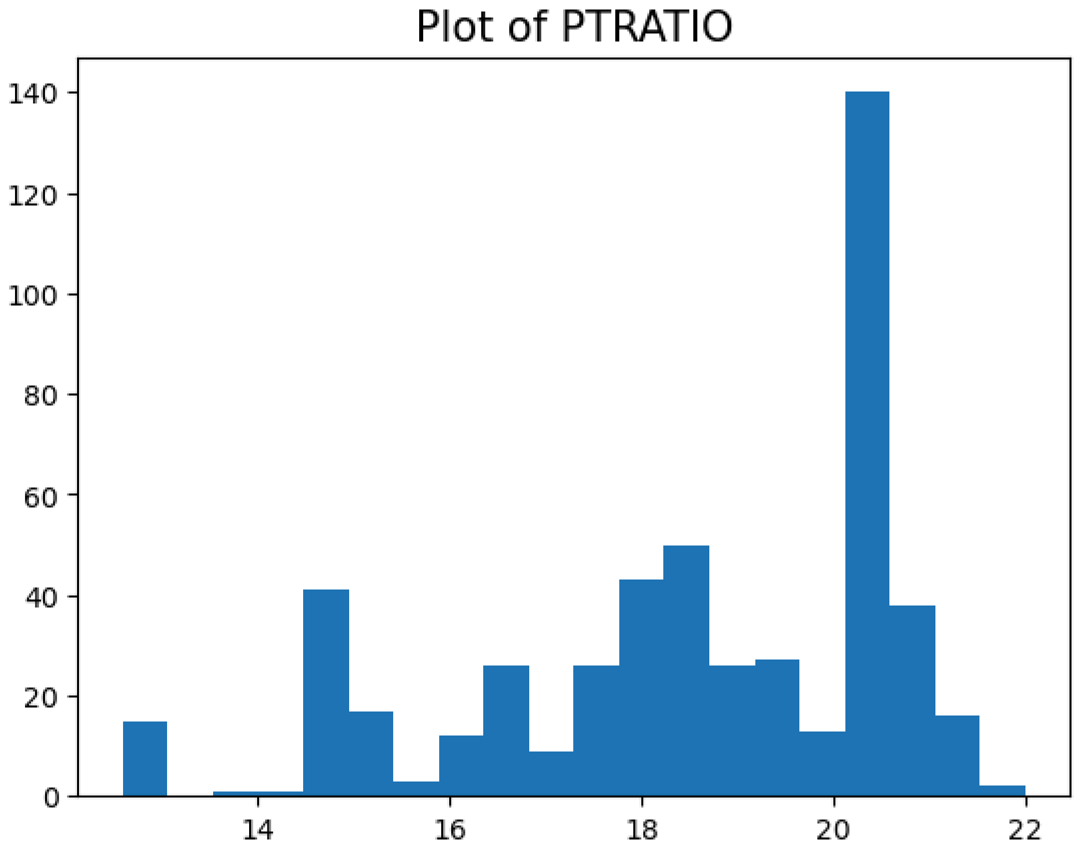

Plot of DIS
0
20
40
60
80
2
4
6
8
10
12

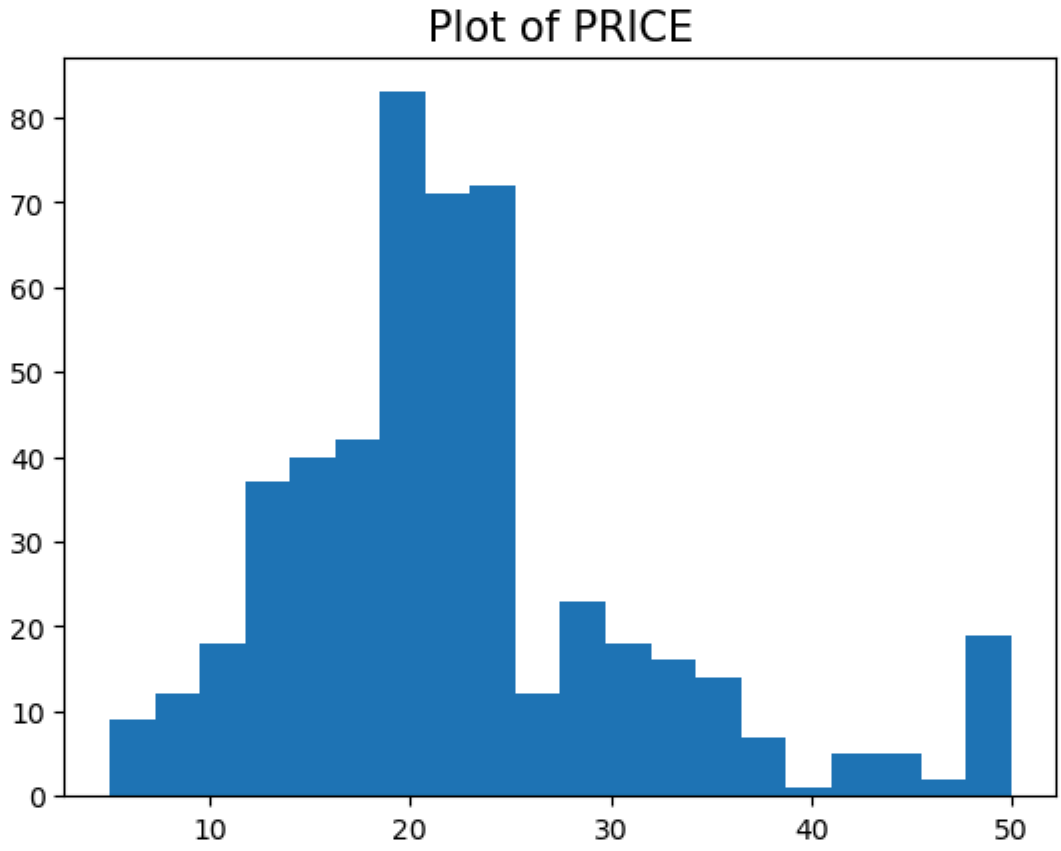


Plot of RAD



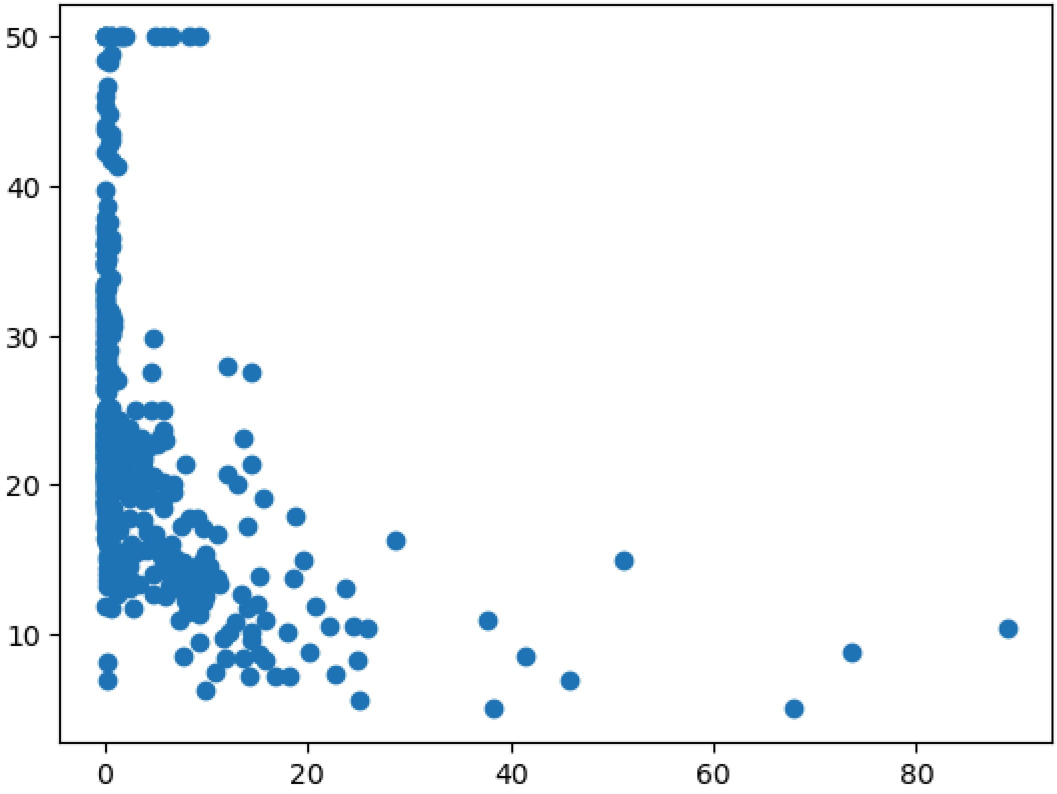
Plot of TAX



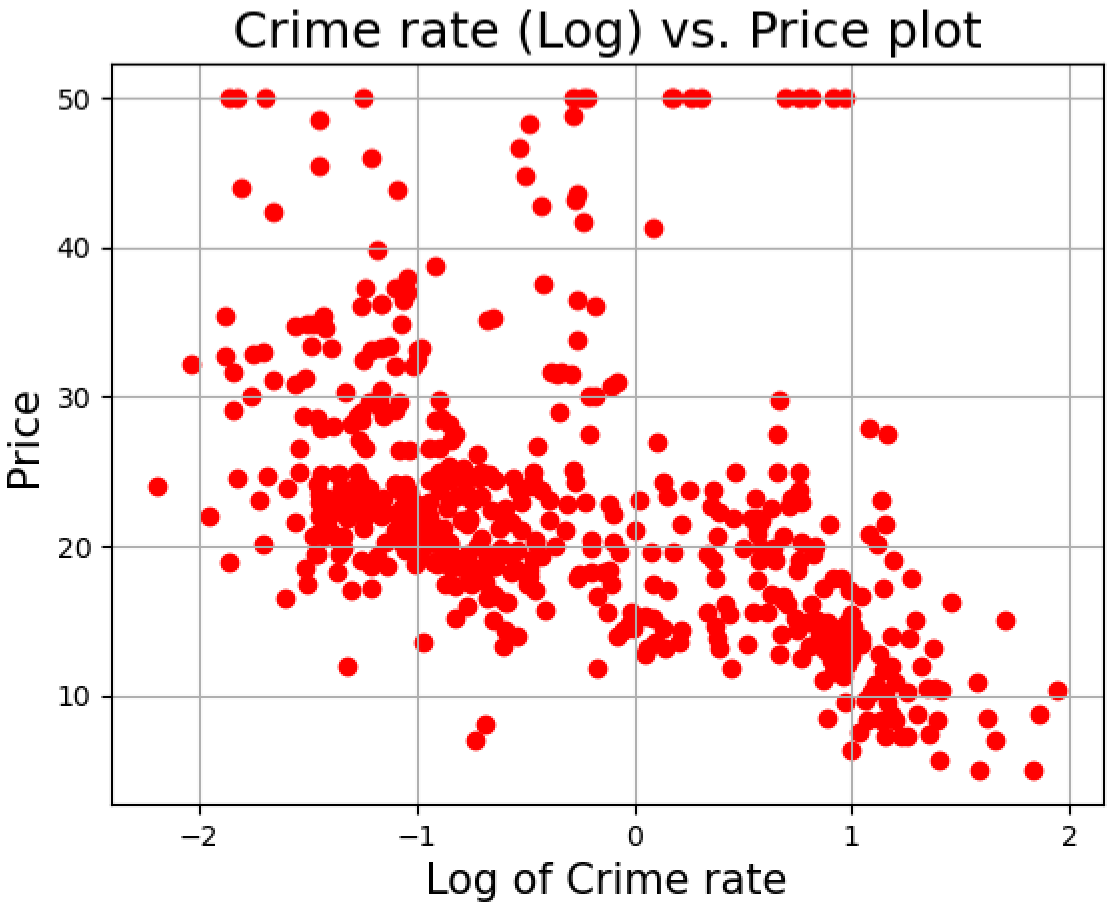


In [17]:

```
#Plots a scatter plot between the CRIM (Crime rate per capita) and PRICE (ho
plt.scatter(df1['CRIM'],df1['PRICE'])
plt.show()
```



In [18]:
```
"""Applies a logarithmic transformation (np.log10) to the CRIM data to handl
Uses c='red' to color the scatter points in red.
Adds a title, labels, and gridlines for better presentation."""

plt.scatter(np.log10(df1['CRIM']),df1['PRICE'],c='red')
plt.title("Crime rate (Log) vs. Price plot", fontsize=18)
plt.xlabel("Log of Crime rate",fontsize=15)
plt.ylabel("Price",fontsize=15)
plt.grid(True)
plt.show()
```



In [22]:

```
#Computes the mean (average) number of rooms (RM).
print("Mean is :"+ str(df1['RM'].mean()))
#Finds the median (middle value) of the AGE column.
print("Median is :"+ str(df1['AGE'].median()))
#Calculates the mean of the DIS column
print("Mean is :"+ str(df1['DIS'].mean()))
```

```
Mean is :6.284634387351779
Median is :77.5
Mean is :3.795042687747036
```

In [24]:

```
"""Creates a boolean mask low_price where True indicates houses priced below
print(low_price) outputs this mask, showing True or False for each house."""
low_price=df1['PRICE']<20
print(low_price)
```

```
0      False
1      False
2      False
3      False
4      False
       ...
501    False
502    False
503    False
504    False
505     True
Name: PRICE, Length: 506, dtype: bool
```

In [26]:
```
# That many houses are priced below 20,000. So that is the answer.
print("\n Mean of Houses priced below 20000 is:" , low_price.mean())
# Converts the proportion into a percentage by multiplying it by 100, then p
pcnt=low_price.mean()*100
print("\nPercentage of house with <20,000 price is: ",pcnt)
```

```
 Mean of Houses priced below 20000 is: 0.4150197628458498

Percentage of house with <20,000 price is:  41.50197628458498
```

2. The Data Wrangling Workshop: Activity 4.01, page 233

In this activity, we will detect outliers in the Adult Income Dataset from the UCI machine learning portal https://packt.live/2N9lRUU.

You can find a description of the dataset https://packt.live/2N9lRUU. We will use the concepts we've learned throughout this chapter, such as subsetting, applying user-defined functions, summary statistics, visualizations, boolean indexing, and group by to find a whole group of outliers in a dataset. We will create a bar plot to plot this group of outliers. Finally, we will merge two datasets by using a common key.

In [29]:
```
"""NumPy (np): Utilized for numerical operations.
Pandas (pd): A versatile library for managing and analyzing tabular data.
Matplotlib (plt): For creating visualizations such as histograms, box plots,

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [31]:
```
#Loads the Adult Income Dataset into a pandas DataFrame df using pd.read_csv
df = pd.read_csv("/Users/balakrishnamupparaju/Downloads/adult_income_data.cs
#displays the first five rows, giving a snapshot of the data.
df.head()
```

Out[31]:

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Female	0

In [33]:

```
"""Opens the file containing column names (adult_income_names.txt).
Iterates through each line, extracts variable names, and appends them to the
This is useful because the dataset may not have proper headers by default.""

names = []
with open('/Users/balakrishnamupparaju/Downloads/adult_income_names.txt', 'r
    for line in f:
        f.readline()
        var = line.split(":")[0]
        names.append(var)
names
```

Out[33]:

```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'sex',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country']
```

In [35]:

```
#Adds the Income column (binary label indicating income class).
#Reloads the dataset, this time with proper column headers defined in names.
names.append('Income')
df = pd.read_csv("/Users/balakrishnamupparaju/Downloads/adult_income_data.cs
df.head()
```

Out[35]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife

In [37]:

```python
#Computes summary statistics for numeric columns, such as mean, standard dev
df.describe()
```

Out[37]:

	age	fnlwgt	education-num	capital-gain	capital-loss	h
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	3256
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	4
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	4
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	4
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	4
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	9

In [39]:

```python
"""
Specifies a list of categorical variables.
"""
vars_class = ['workclass', 'education', 'marital-status',
              'occupation', 'relationship', 'sex', 'native-country']

"""For each variable, prints:
Unique classes (categories in that column).
Count of unique classes.
Helps understand the categorical data distribution.
"""
for v in vars_class:
    classes = df[v].unique()
    num_classes = df[v].nunique()
    print("There are {} classes in the \"{}\" column. They are: {}".format(r
    print("-" * 100)
```

```
There are 9 classes in the "workclass" column. They are: [' State-gov' ' Sel
f-emp-not-inc' ' Private' ' Federal-gov' ' Local-gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked']
----------------------------------------------------------------------------
------------------------
There are 16 classes in the "education" column. They are: [' Bachelors' ' HS
-grad' ' 11th' ' Masters' ' 9th' ' Some-college'
 ' Assoc-acdm' ' Assoc-voc' ' 7th-8th' ' Doctorate' ' Prof-school'
 ' 5th-6th' ' 10th' ' 1st-4th' ' Preschool' ' 12th']
----------------------------------------------------------------------------
------------------------
There are 7 classes in the "marital-status" column. They are: [' Never-marri
ed' ' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed']
----------------------------------------------------------------------------
------------------------
There are 15 classes in the "occupation" column. They are: [' Adm-clerical'
' Exec-managerial' ' Handlers-cleaners' ' Prof-specialty'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support' ' ?'
 ' Protective-serv' ' Armed-Forces' ' Priv-house-serv']
----------------------------------------------------------------------------
------------------------
There are 6 classes in the "relationship" column. They are: [' Not-in-famil
y' ' Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']
----------------------------------------------------------------------------
------------------------
There are 2 classes in the "sex" column. They are: [' Male' ' Female']
----------------------------------------------------------------------------
------------------------
There are 42 classes in the "native-country" column. They are: [' United-Sta
tes' ' Cuba' ' Jamaica' ' India' ' ?' ' Mexico' ' South'
 ' Puerto-Rico' ' Honduras' ' England' ' Canada' ' Germany' ' Iran'
 ' Philippines' ' Italy' ' Poland' ' Columbia' ' Cambodia' ' Thailand'
 ' Ecuador' ' Laos' ' Taiwan' ' Haiti' ' Portugal' ' Dominican-Republic'
 ' El-Salvador' ' France' ' Guatemala' ' China' ' Japan' ' Yugoslavia'
 ' Peru' ' Outlying-US(Guam-USVI-etc)' ' Scotland' ' Trinadad&Tobago'
 ' Greece' ' Nicaragua' ' Vietnam' ' Hong' ' Ireland' ' Hungary'
 ' Holand-Netherlands']
----------------------------------------------------------------------------
------------------------
```

In [41]:

```
#Calculates the total number of null values in each column.
#A critical step to identify any data quality issues.
df.isnull().sum()
```

Out[41]:

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

In [43]:

```
#Creates a smaller DataFrame df_subset containing only the age, education, a
df_subset = df[['age', 'education', 'occupation']]
df_subset.head()
```

Out[43]:

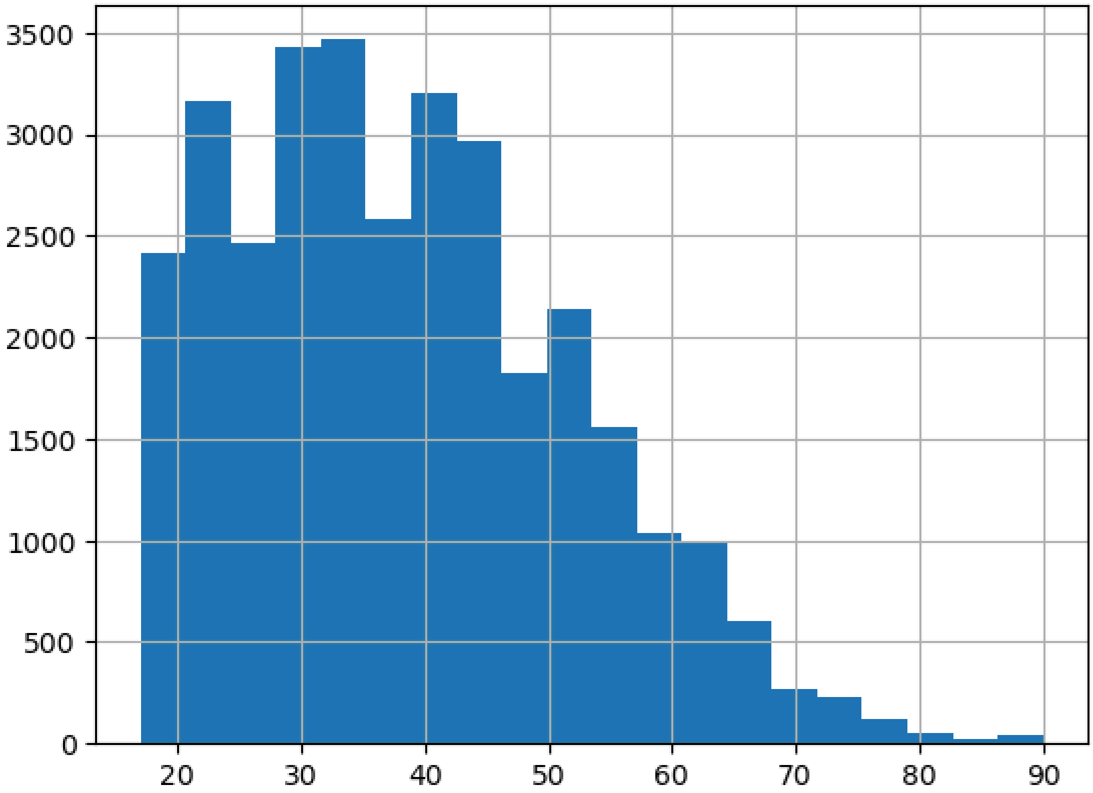
	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners
4	28	Bachelors	Prof-specialty

In [45]:

```
#Creates a histogram of the age column, dividing the data into 20 bins.
df_subset['age'].hist(bins=20)
```

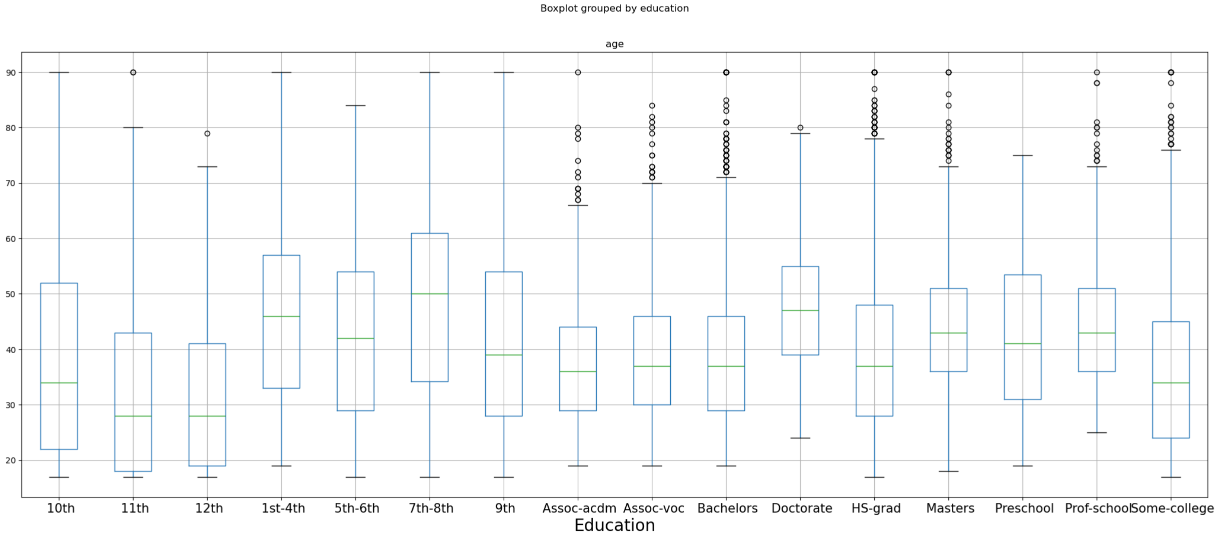
Out[45]:

```
<Axes: >
```



In [47]:

```
#Generates a box plot of age grouped by education.
#Adjusts the figure size and font for better readability.
df_subset.boxplot(column='age', by='education', figsize=(25, 10))
plt.xticks(fontsize=15)
plt.xlabel("Education", fontsize=20)
plt.show()
```



In [49]:

```
#Defines a function to strip leading/trailing whitespace from strings.
def strip_whitespace(s):
    return s.strip()
```

In [51]:

```
#Applies strip_whitespace to clean up the education and occupation columns.
#Drops the temporary columns after cleaning.
```

```python
import warnings
warnings.filterwarnings("ignore")

df_subset['education_stripped'] = df['education'].apply(strip_whitespace)
df_subset['education'] = df_subset['education_stripped']
df_subset.drop(labels=['education_stripped'], axis=1, inplace=True)

df_subset['occupation_stripped'] = df['occupation'].apply(strip_whitespace)
df_subset['occupation'] = df_subset['occupation_stripped']
df_subset.drop(labels=['occupation_stripped'], axis=1, inplace=True)
```

In [53]:

```python
#Filters rows where age is between 30 and 50 (inclusive).
#The result is stored in df_filtered.
df_filtered = df_subset[(df_subset['age'] >= 30) & (df_subset['age'] <= 50)]
df_filtered.head()
```

Out[53]:

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
5	37	Masters	Exec-managerial
6	49	9th	Other-service

In [55]:

```python
#Calculates and prints the total number of rows that meet the filtering crit
answer_1 = df_filtered.shape[0]
print("There are {} people of age between 30 and 50 in this dataset.".format
```

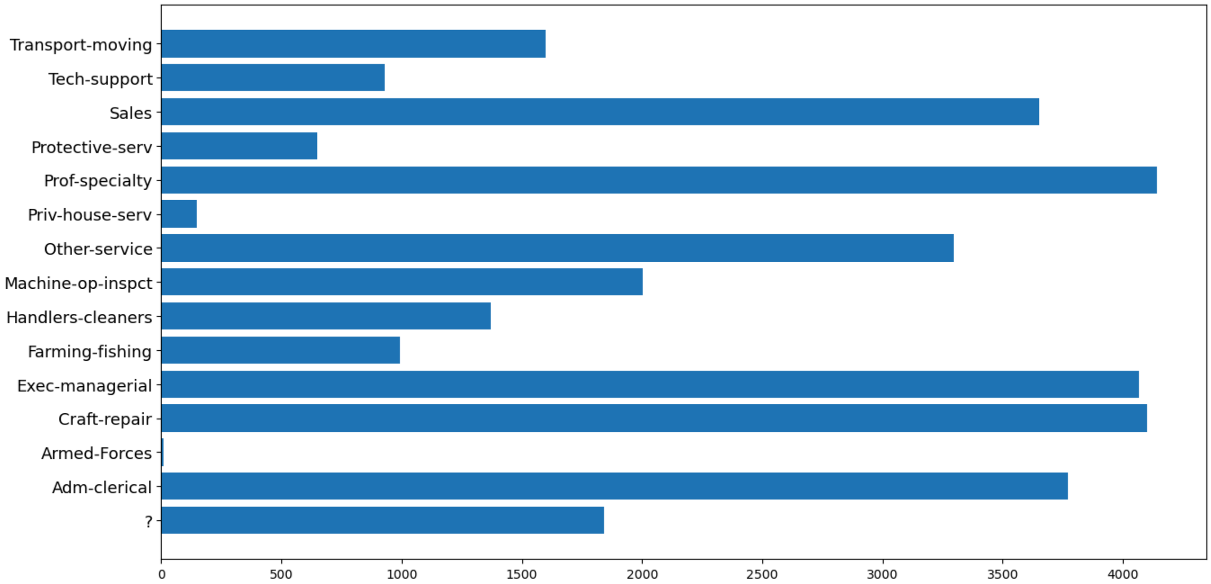
```
There are 16390 people of age between 30 and 50 in this dataset.
```

In [57]:

```python
#Groups the data by the occupation column and computes summary statistics fo
occupation_stats = df_subset.groupby('occupation').describe()['age']
```

In [59]:

```python
"""Creates a horizontal bar plot where:
Y-axis represents occupations.
X-axis represents the count of people in each occupation.
"""
plt.figure(figsize=(15, 8))
plt.barh(y=occupation_stats.index, width=occupation_stats['count'])
plt.yticks(fontsize=13)
plt.show()
```



In [61]:

```
#Randomly samples 5 rows from two subsets of the original DataFrame (df_1 ar
df_1 = df[['age', 'workclass', 'occupation']].sample(5, random_state=101)
df_2 = df[['education', 'occupation']].sample(5, random_state=101)
#df_1.head()
#df_2.head()
```

In [63]:

```
#Merges df_1 and df_2 on the common key (occupation) using an inner join.
#Removes duplicate rows from the result.
df_merged = pd.merge(df_1, df_2, on='occupation', how='inner').drop_duplicat
df_merged
```

Out[63]:

	age	workclass	occupation	education
0	51	Private	Machine-op-inspct	HS-grad
1	19	Private	Sales	11th
2	40	Private	Exec-managerial	HS-grad
3	17	Private	Handlers-cleaners	10th
4	61	Private	Craft-repair	7th-8th

3. Create a series and practice basic arithmetic steps

In [66]:

```
#We will use pandas to create the series and perform arithmetic operations.
import pandas as pd
```

In [68]:

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

In [70]:

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

In [72]:

```
#Adds corresponding values for matching indices.
#For indices that do not overlap, the result will contain NaN (indicating mi
result_add = series1 + series2
print("Result of Addition:\n", result_add)
```

```
Result of Addition:
 a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64
```

In [75]:

```
#Subtracts series1 values from series2 for matching indices.
#For indices that do not overlap, the result will contain NaN.
result_subtract = series2 - series1
print("Result of Subtraction:\n", result_subtract)
```

```
Result of Subtraction:
 a   -9.4
c    6.1
d    NaN
e   -3.0
f    NaN
g    NaN
dtype: float64
```