DSC540-T303-Data-Preparation-Week3-4

December 16, 2024

[173]:
```
# Weeks 3 & 4 Exercises
```

[174]:
```
# 1. The Data Wrangling Workshop: Activity 3.01, page 155
# Generating statistics from a csv file.

#Load the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[175]:
```
#Read the Boston housing dataset from the csv file.
```

[176]:
```
df=pd.read_csv("Boston_housing.csv")
```

[177]:
```
#Check the first 10 records
```

[178]:
```
df.head(10)
```

[178]:
```
      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
```

[179]:
```
#Find the total number of records
```

[180]:
```
df.shape
```

[180]:
```
(506, 14)
```

[181]:
```
#Create a smaller dataset which exclude CHAS, NOX, B, LSTAT
```

[182]:
```
df1=df[['CRIM','ZN','INDUS','RM','AGE','DIS',
'RAD','TAX','PTRATIO','PRICE']]
```

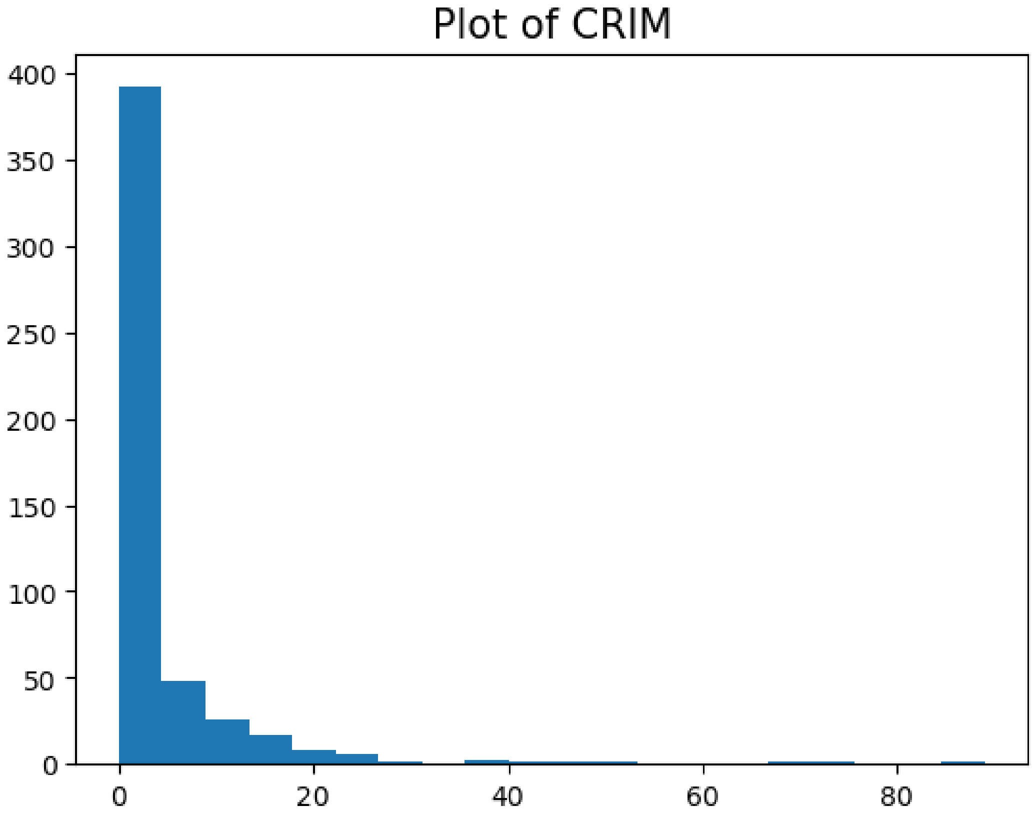
[183]:
```
# Check the last 7 records of the new DataFrame you just created
```

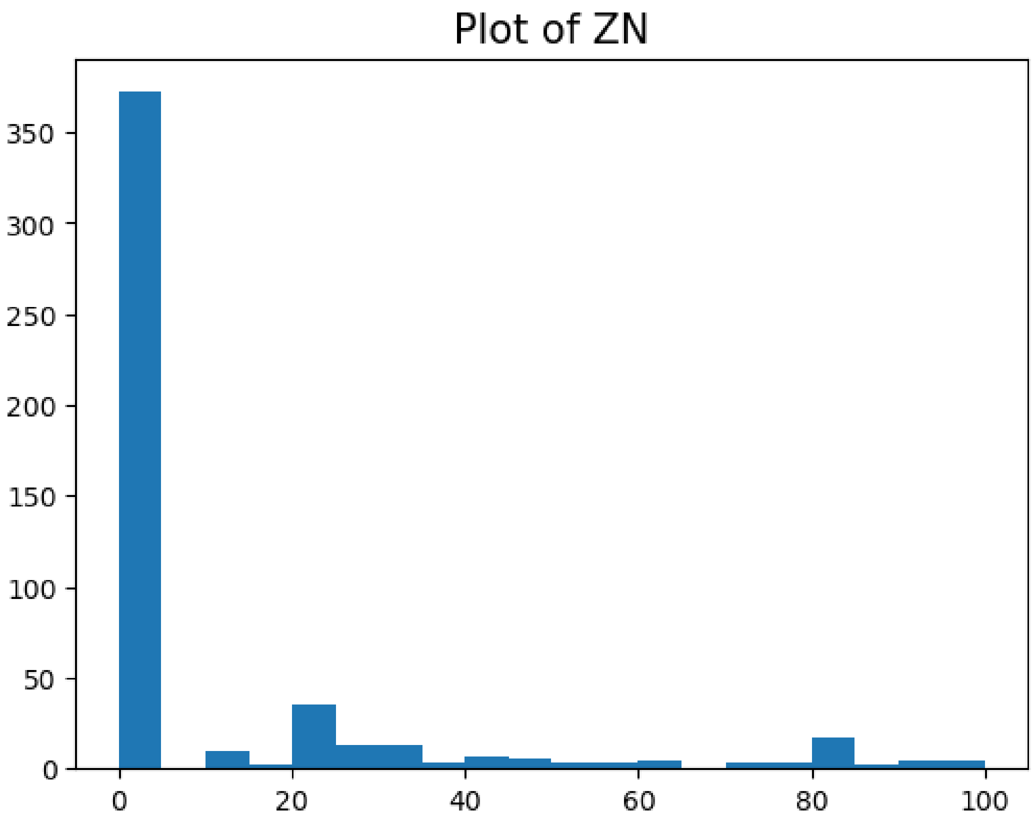
[184]:
```
df1.tail(7)
```

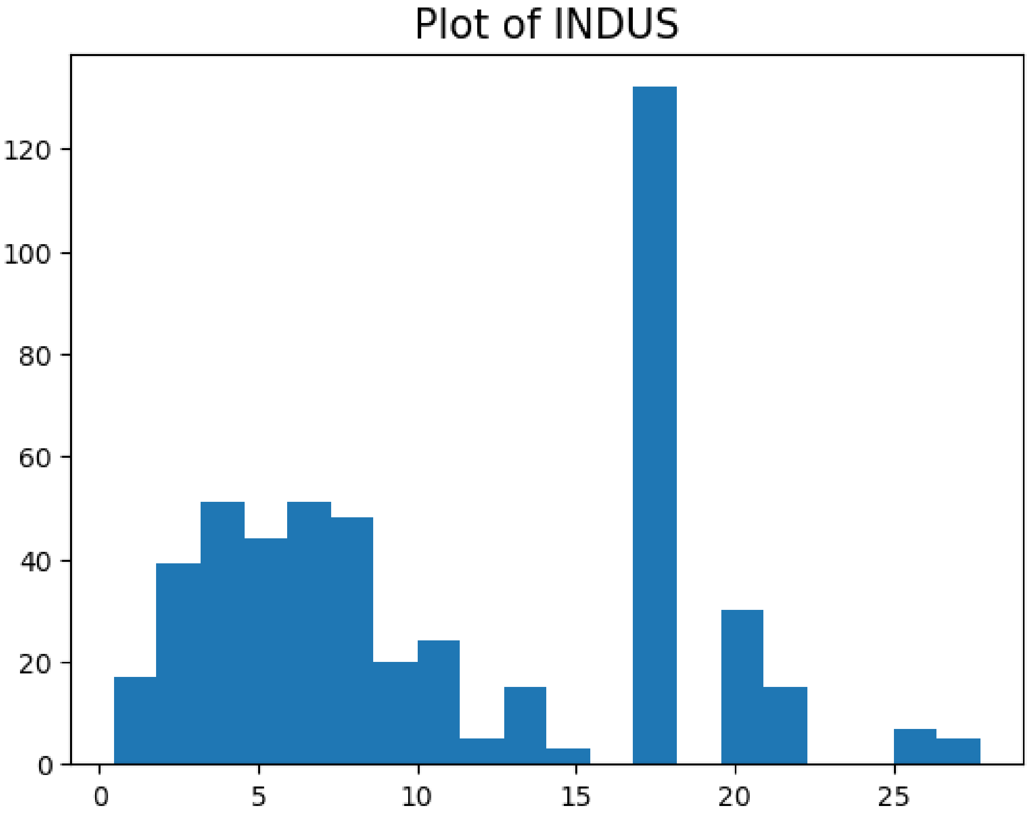
[184]:
```
        CRIM   ZN  INDUS     RM   AGE     DIS  RAD  TAX  PTRATIO  PRICE
499  0.17783  0.0   9.69  5.569  73.5  2.3999    6  391     19.2   17.5
500  0.22438  0.0   9.69  6.027  79.7  2.4982    6  391     19.2   16.8
501  0.06263  0.0  11.93  6.593  69.1  2.4786    1  273     21.0   22.4
502  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
```

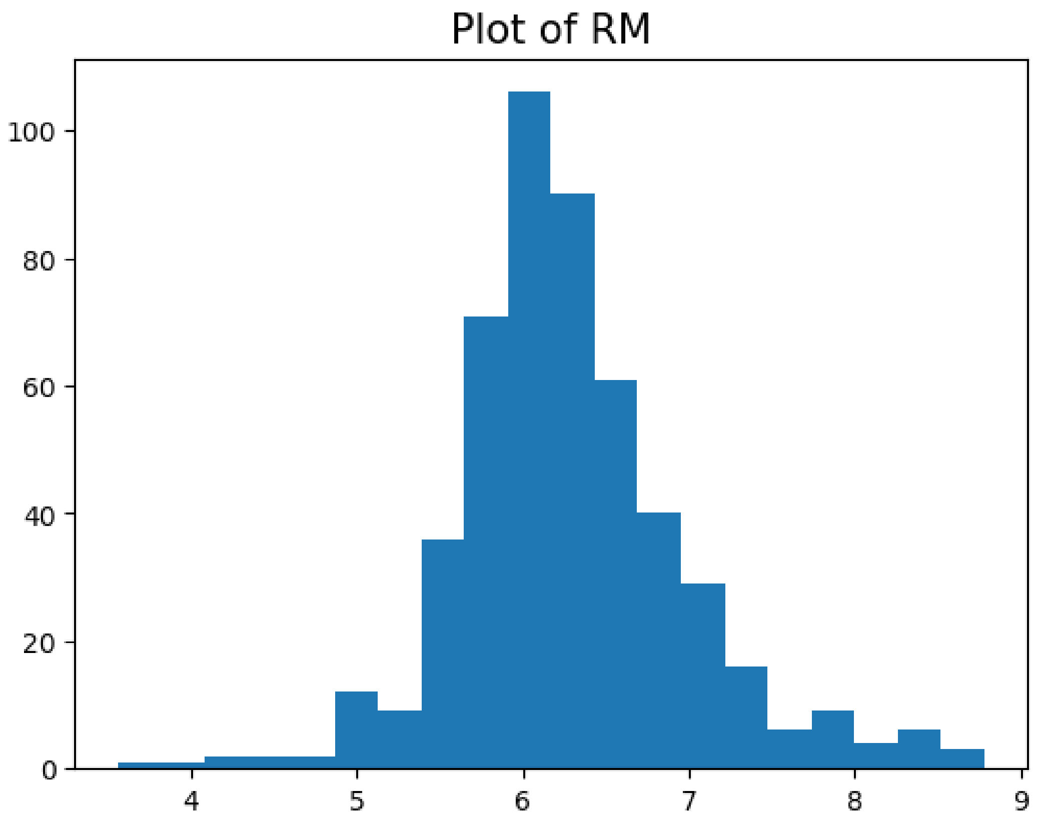
[185]:
```
#Plot hs(columns) in the new DataFrame by using a for loopistograms of all the␣
 ↪variables(columns) in the new DataFrame by using a for loop
```

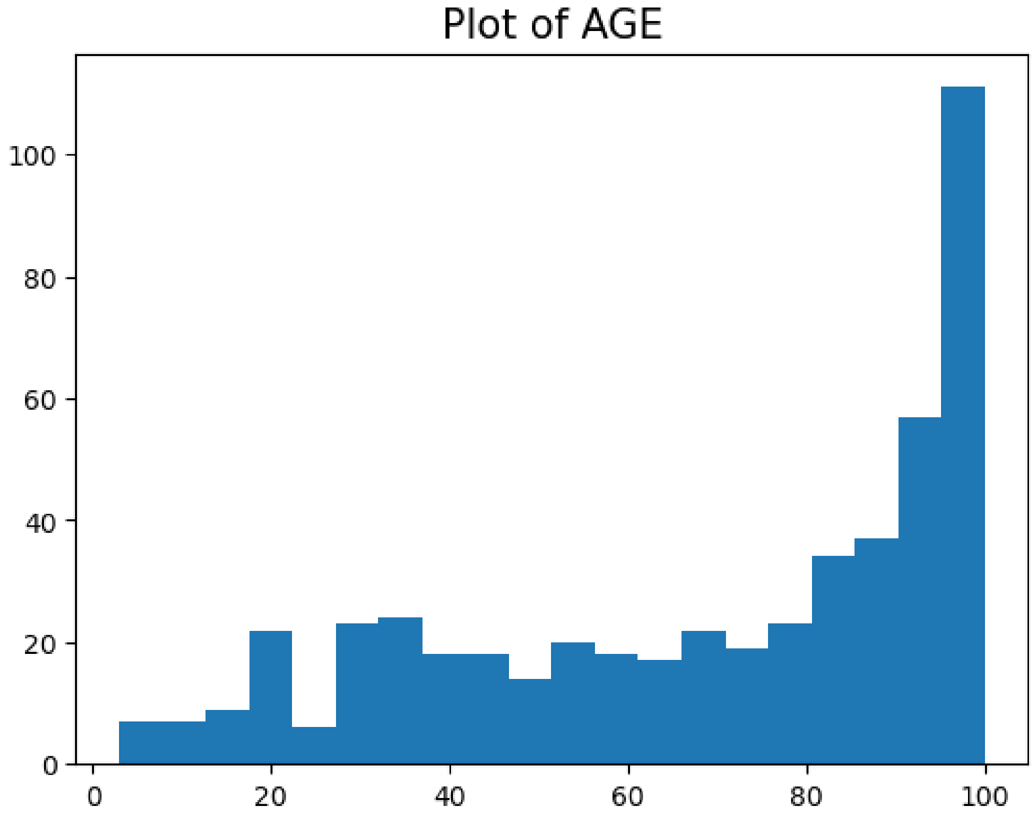
[186]:
```
for c in df1.columns:
    plt.title("Plot of "+c,fontsize=15)
    plt.hist(df1[c],bins=20)
    plt.show()
```

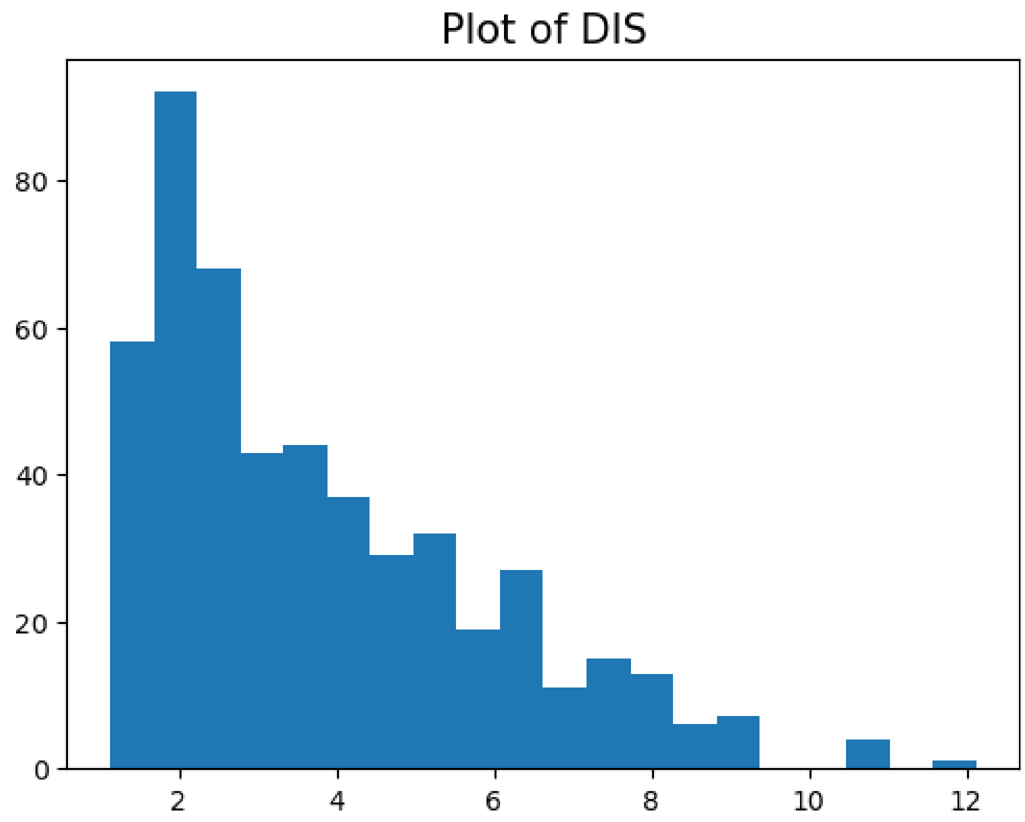

Plot of CRIM
400
350
300
250
200
150
100
50
0
0
20
40
60
80

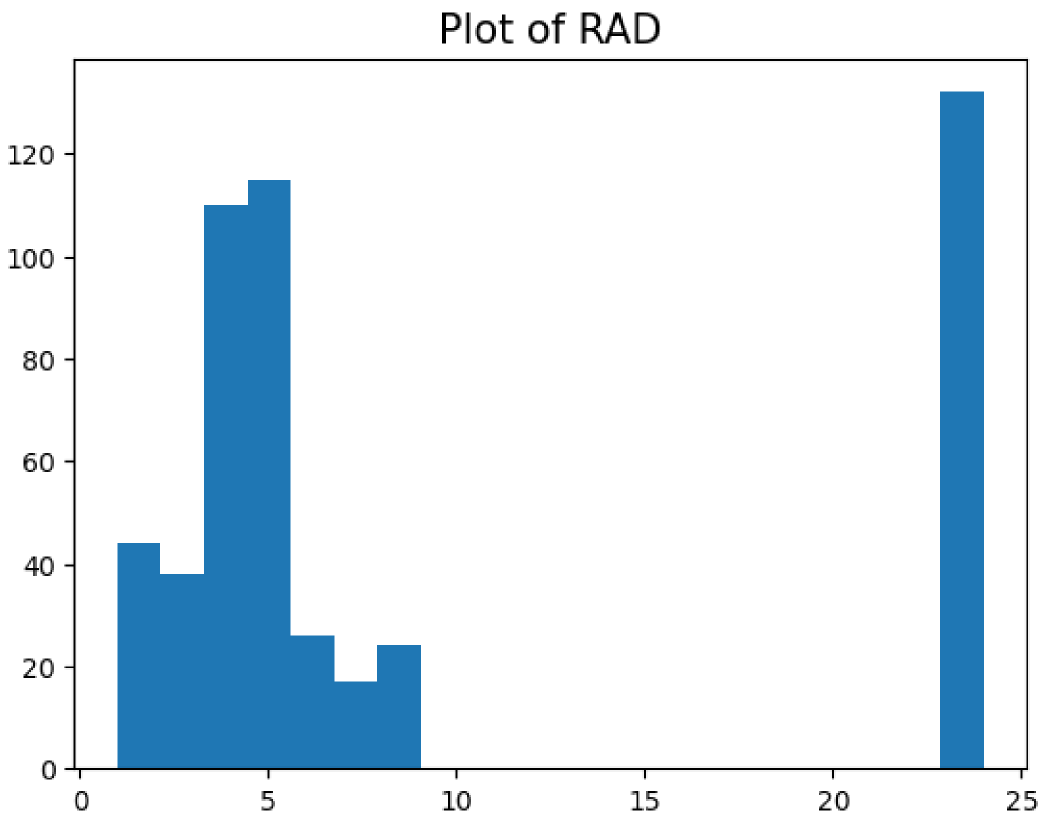

Plot of ZN
0
50
100
150
200
250
300
350
0
20
40
60
80
100

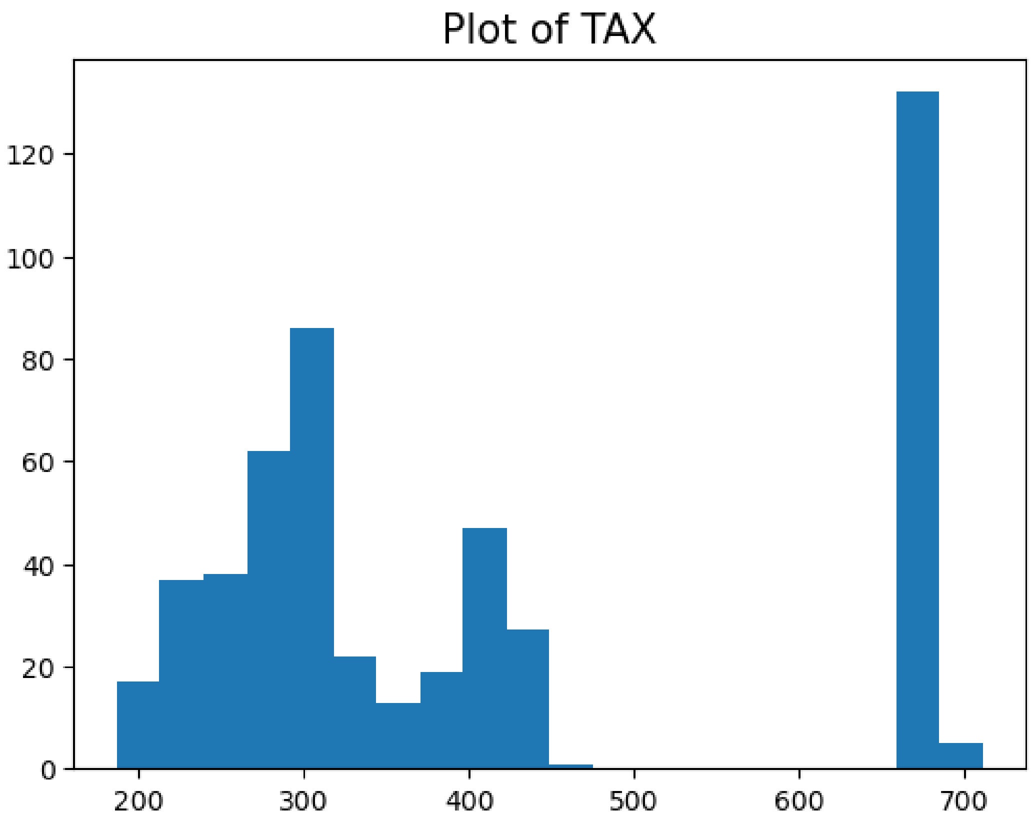

Plot of INDUS
0
20
40
60
80
100
120
0
5
10
15
20
25

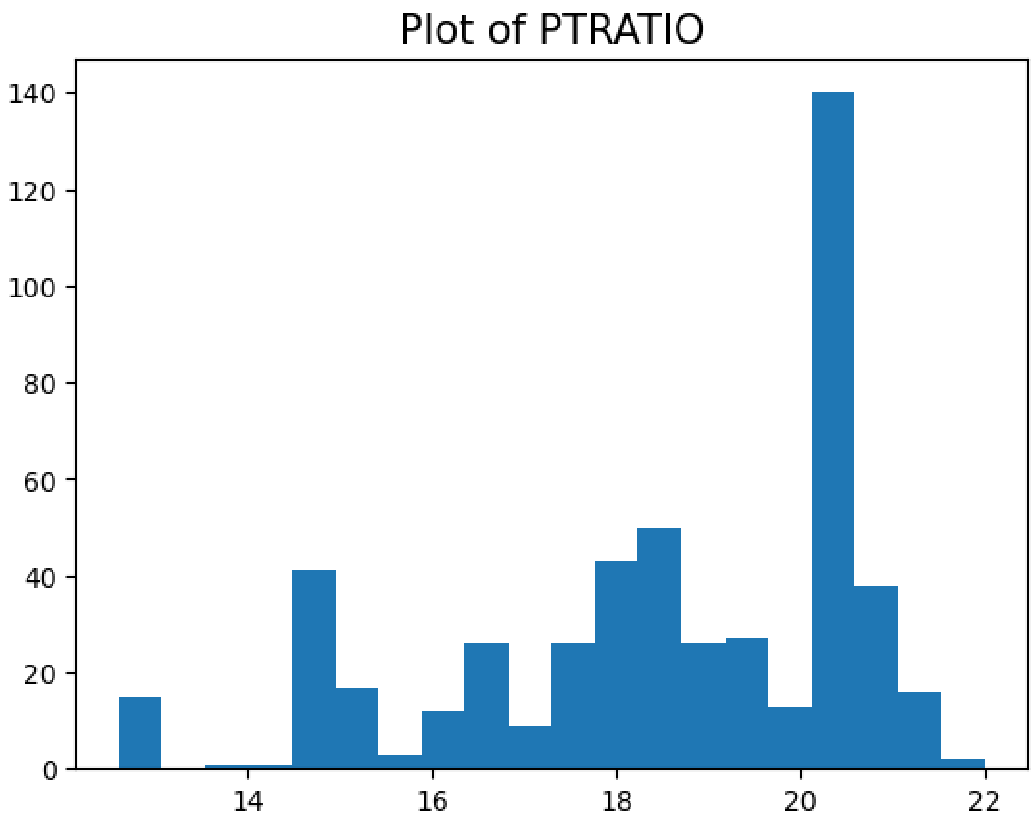

Plot of RM
0
20
40
60
80
100
4
5
6
7
8
9

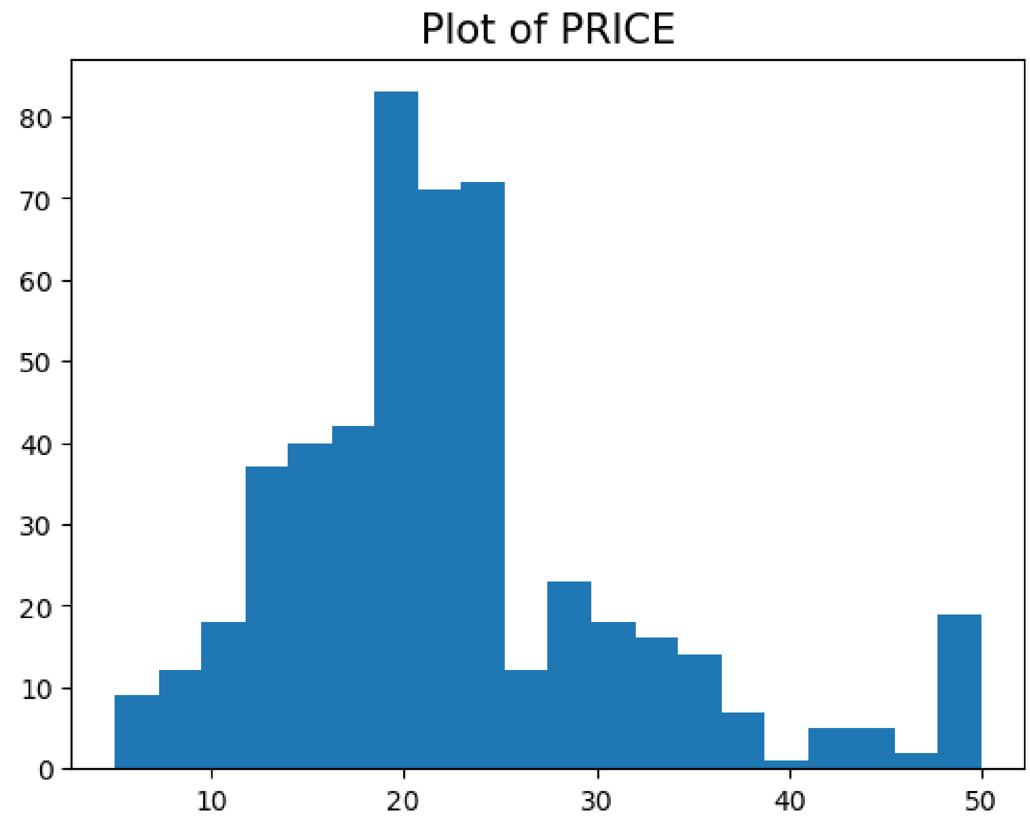

Plot of AGE
0
20
40
60
80
100
0
20
40
60
80
100


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


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


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


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

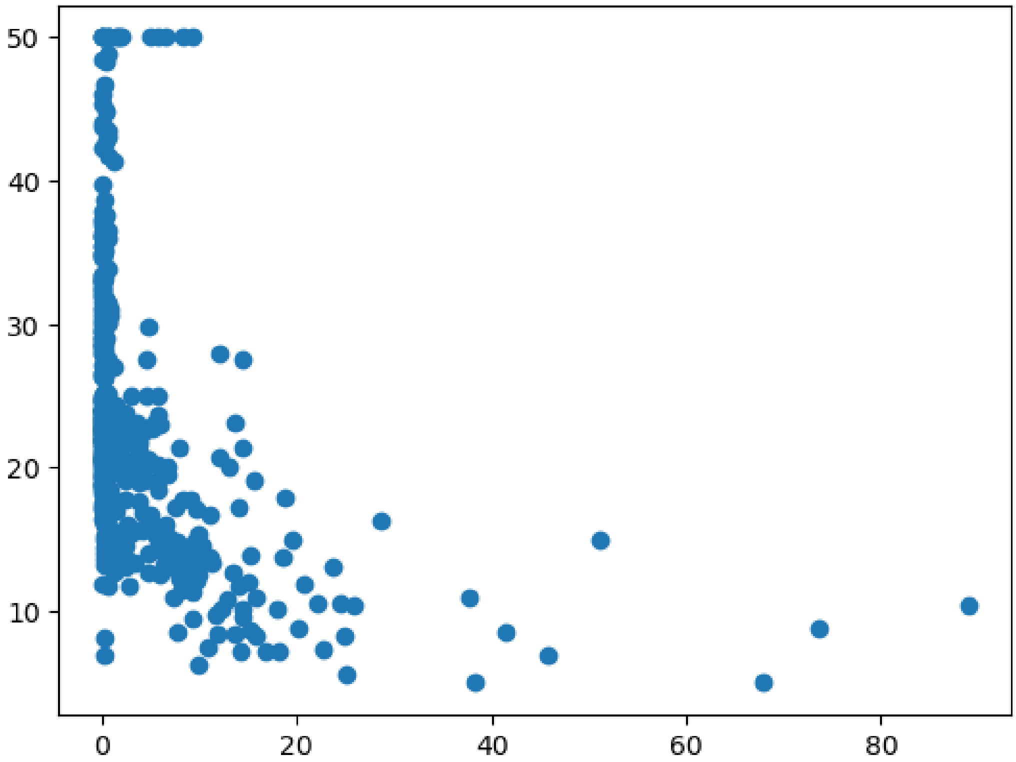


[187]:

```
# Create a scatter plot of crime rate versus price.
```

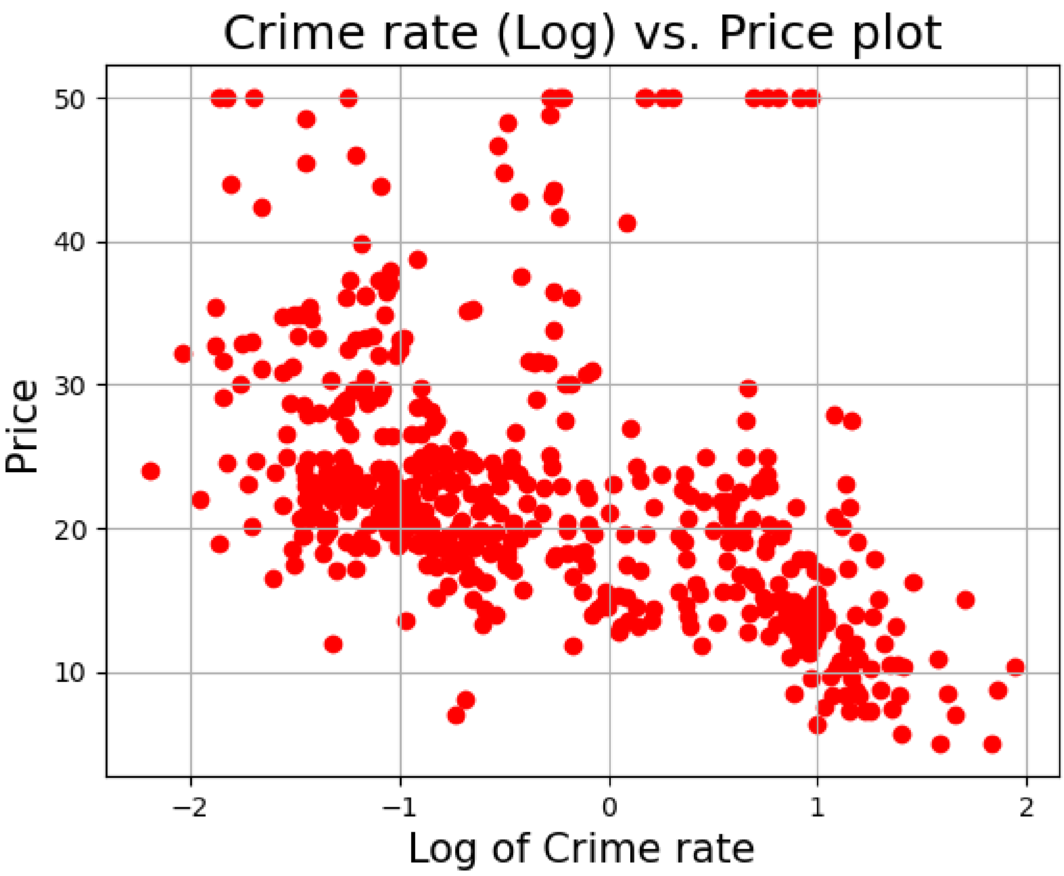
[188]:

```
plt.scatter(df1['CRIM'],df1['PRICE'])
plt.show()
```



[189]:
```
#Create a plot of log10 (crime) veru price.
```

[190]:
```
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()
```



[191]:
```
#Calculate the mean room per dwelling
```

[192]:
```
df1['RM'].mean()
```

[192]:
```
6.284634387351779
```

[193]:
```
#Calculate the median age
```

[194]:
```
df1['AGE'].median()
```

[194]:
```
77.5
```

[195]:
```
# Calculate the average mean distances to five Bostone employment centers
```

[196]:
```
df1['DIS'].mean()
```

[196]:
```
3.795042687747036
```

[197]:
```
#Calculate the price of the housing that's than 20
```

[198]:

```
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
```

[199]:

```
#Calculate the mean of this array
```

[200]:

```
# That many houses are priced below 20,000. So that is the answer.
low_price.mean()
```

[200]:

```
0.4150197628458498
```

[201]:

```
#Calculate the percentage of houses with a low price < $20000
```

[202]:

```
# You can convert that into percentage by multiplying with 100
pcnt=low_price.mean()*100
print("\nPercentage of house with <20,000 price is: ",pcnt)
```

```

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

[203]:

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

[204]:

```
# Read the adult income dataset
df = pd.read_csv("adult_income_data.csv")
df.head()
```

[204]:

```
   39         State-gov   77516   Bachelors  13        Never-married  \
0  50  Self-emp-not-inc   83311   Bachelors  13   Married-civ-spouse
1  38           Private  215646     HS-grad   9             Divorced
2  53           Private  234721        11th   7   Married-civ-spouse
3  28           Private  338409   Bachelors  13   Married-civ-spouse
4  37           Private  284582     Masters  14   Married-civ-spouse

         Adm-clerical   Not-in-family     Male  2174  0  40   United-States  \
0     Exec-managerial         Husband     Male     0  0  13   United-States
1   Handlers-cleaners   Not-in-family     Male     0  0  40   United-States
```

```
2   Handlers-cleaners         Husband     Male     0  0  40   United-States
3      Prof-specialty            Wife   Female     0  0  40            Cuba
4     Exec-managerial            Wife   Female     0  0  40   United-States

     <=50K
0    <=50K
1    <=50K
2    <=50K
3    <=50K
4    <=50K
```

[205]:

```
#Create a script that will read a text file line by line and exxtract the first␣
 ↪line which is the header of the .csv file
```

[206]:

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

[206]:

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

[207]:

```
# Add the name of "Income" for the response variable(last column) to the␣
 ↪dataset by using the "append" comman
```

[208]:

```
names.append('Income')
```

[209]:

```
# Read the new file again
df = pd.read_csv("adult_income_data.csv",names=names)
df.head()
```

[209]:

```
   age          workclass  fnlwgt   education  education-num  \
0   39          State-gov   77516   Bachelors             13
```

```
1   50   Self-emp-not-inc   83311   Bachelors             13
2   38            Private   215646     HS-grad              9
3   53            Private   234721        11th              7
4   28            Private   338409   Bachelors             13

        marital-status          occupation    relationship      sex  \
0        Never-married        Adm-clerical   Not-in-family     Male
1   Married-civ-spouse     Exec-managerial         Husband     Male
2             Divorced   Handlers-cleaners   Not-in-family     Male
3   Married-civ-spouse   Handlers-cleaners         Husband     Male
4   Married-civ-spouse      Prof-specialty            Wife   Female

   capital-gain  capital-loss  hours-per-week  native-country  Income
0          2174             0              40   United-States   <=50K
1             0             0              13   United-States   <=50K
2             0             0              40   United-States   <=50K
3             0             0              40   United-States   <=50K
4             0             0              40            Cuba   <=50K
```

[210]:
```
# Use the describe command to get the stastical summary of the dataset.
```

[211]:
```
df.describe()
```

[211]:
```
                age        fnlwgt  education-num  capital-gain  capital-loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000  32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844     87.303830
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
50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    32561.000000
mean        40.437456
std         12.347429
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```

[212]:
```
# Make a list of all variable
```

[213]:
```
# Make a list of all variables with classes
vars_class = ['workclass','education','marital-status',
                    'occupation','relationship','sex','native-country']
```

[214]:

```
# Create a loop to count and print them by uing the following command
```

[215]:

```
for v in vars_class:
    classes=df[v].unique()
    num_classes = df[v].nunique()
    print("There are {} classes in the \"{}\" column. They are: {}".
 ↪format(num_classes,v,classes))
    print("-"*100)
```

```
There are 9 classes in the "workclass" column. They are: [' State-gov' ' Self-
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-married'
' 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-family' '
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-States'
' 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'
 ' Holland-Netherlands']
--------------------------------------------------------------------------------
--------------------
```

[216]:
```
# Find the missing values by using the follwoing command
```

[217]:
```
df.isnull().sum()
```

[217]:
```
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
```

[218]:
```
# Create a DataFrame with only age,education and occupation by using subnetting
```

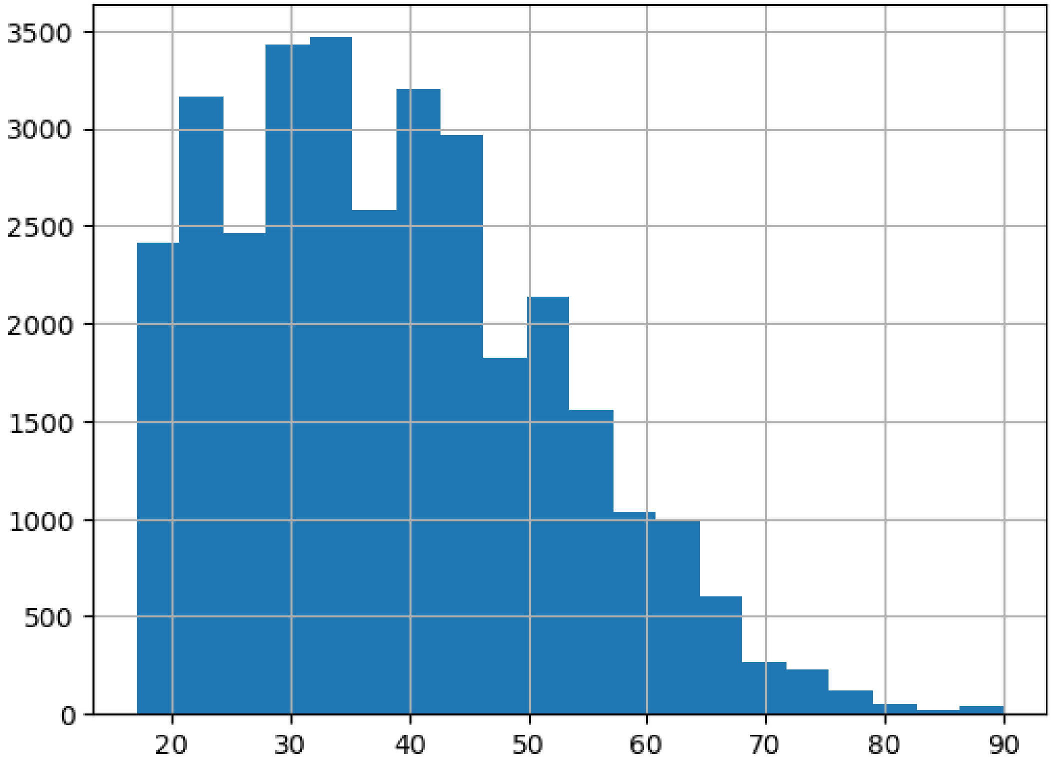
[219]:
```
df_subset = df[['age','education','occupation']]
df_subset.head()
```

[219]:
```
   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
```

[220]:
```
# Plot a histogram of age with a bin size of 20
```

[221]:
```
df_subset['age'].hist(bins=20)
```

[221]:
```
<Axes: >
```

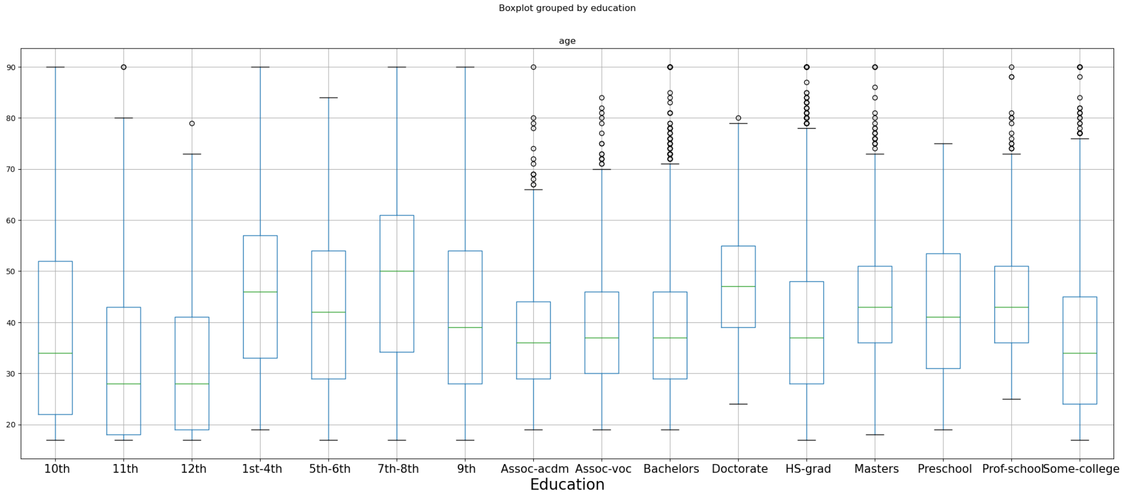


[222]:

```
# PLot box plots for age grouped by education
```

[223]:

```
df_subset.boxplot(column='age',by='education',figsize=(25,10))
plt.xticks(fontsize=15)
plt.xlabel("Education",fontsize=20)
plt.show()
```



[224]:

```
# create a function to strip the whitespace charecters
```

[225]:

```
def strip_whitespace(s):
    return s.strip()
```

[226]:

```
# Use the apply method to apply the function to all the columns
```

[227]:

```
import warnings
# Suppress all warnings
warnings.filterwarnings("ignore")

# Education column
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)

# Occupation column
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)
```

[228]:

```
# Find the number of people who are aged between 30 & 50
```

[229]:

```
# Conditional clauses and join them by & (AND)
df_filtered=df_subset[(df_subset['age']>=30) & (df_subset['age']<=50)]
```

[230]:

```
# Check the contents of the new datasets
```

[231]:

```
df_filtered.head()
```

[231]:

```
   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
```

[232]:

```
# Find the shape of the filtered DataFrame and specify the index of the tuple␣
 ↪as 0 to return the first element
```

[233]:

```
answer_1=df_filtered.shape[0]
answer_1
```

[233]: 16390

[234]:

```
# Print the number of people of age between 30 and 50 in this dataset
```

[235]:

```
print("There are {} people of age between 30 and 50 in this dataset".
 ↪format(answer_1))
```

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

[236]:

```
# Group by occupation and show the summary statistics by age
```

[237]:

```
df_subset.groupby('occupation').describe()['age']
```

[237]:

```
                     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
```

[238]:

```
# Use subsets and groupby to find the outliers
```

[239]:

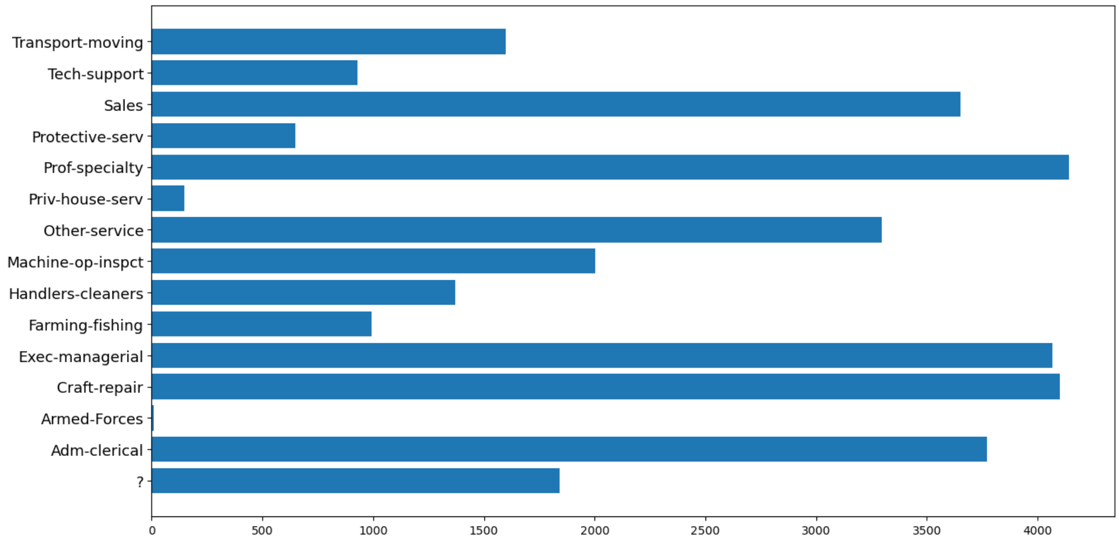
```
occupation_stats= df_subset.groupby(
    'occupation').describe()['age']
```

[240]:

```
# Plot the values on a bar chart
```

[241]:

```
plt.figure(figsize=(15,8))
plt.barh(y=occupation_stats.index,
         width=occupation_stats['count'])
plt.yticks(fontsize=13)
plt.show()
```



```
[242]: #create new dataset where occupation is column. first create two such datasets␣
         ↪by taking random samples from the full datasets
```

```
[243]: df_1 = df[['age',
                   'workclass',
                   'occupation']].sample(5,random_state=101)
        df_1.head()
```

```
[243]:        age workclass           occupation
       22357   51   Private    Machine-op-inspct
       26009   19   Private                Sales
       20734   40   Private      Exec-managerial
       17695   17   Private    Handlers-cleaners
       27908   61   Private         Craft-repair
```

```
[244]: # The second dataset
```

```
[245]: df_2 = df[['education',
                   'occupation']].sample(5,random_state=101)
        df_2.head()
```

```
[245]:      education           occupation
       22357   HS-grad    Machine-op-inspct
       26009      11th                Sales
       20734   HS-grad      Exec-managerial
       17695      10th    Handlers-cleaners
       27908   7th-8th         Craft-repair
```

```
[246]: # Merge the two datasets together
```

[247]:

```
df_merged = pd.merge(df_1,df_2,
                     on='occupation',
                     how='inner').drop_duplicates()
df_merged
```

[247]:

```
   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
```

[ ]:

```

```

[248]:

```
# 3. Create a series and practice basic arithmetic steps
# a. Series 1 = 7.3, -2.5, 3.4, 1.5
# i. Index = ‘a’, ‘c’, ‘d’, ‘e’
# b. Series 2 = -2.1, 3.6, -1.5, 4, 3.1
# i. Index = ‘a’, ‘c’, ‘e’, ‘f’, ‘g’
# c. Add Series 1 and Series 2 together and print the results
# d. Subtract Series 1 from Series 2 and print the results
```

[249]:

```
# Creating Series 1
series_1 = pd.Series([7.3, -2.5, 3.4, 1.5], index=['a', 'c', 'd', 'e'])
print("Series 1:")
print(series_1)
```

```
Series 1:
a    7.3
c   -2.5
d    3.4
e    1.5
dtype: float64
```

[250]:

```
# Creating Series 2
series_2 = pd.Series([-2.1, 3.6, -1.5, 4, 3.1], index=['a', 'c', 'e', 'f', 'g'])
print("Series 2:")
print(series_2)
```

```
Series 2:
a   -2.1
c    3.6
e   -1.5
f    4.0
g    3.1
dtype: float64
```

[251]:

```
# c. Add Series 1 and Series 2 together
```

[252]:

```python
# Adding Series 1 and Series 2
result_addition = series_1 + series_2
print("\nResult of adding Series 1 and Series 2:")
print(result_addition)
```

```
Result of adding Series 1 and Series 2:
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64
```

[253]:

```python
# d. Subtract Series 1 from Series 2
```

[254]:

```python
# Subtracting Series 1 from Series 2
result_subtraction = series_2 - series_1
print("\nResult of subtracting Series 1 from Series 2:")
print(result_subtraction)
```

```
Result of subtracting Series 1 from Series 2:
a   -9.4
c    6.1
d    NaN
e   -3.0
f    NaN
g    NaN
dtype: float64
```

[ ]:

```python

```