DSC 540-Week 3 & 4 Exercises

Kannur, Gyan

Data Wrangling with Python: Activity 5

```
import numpy as np
from pandas import read_csv
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

## reading boston data
#column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
boston_data = read_csv('./datasets/Boston_housing.csv')

## Check the first 10 records
boston_data.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
```

```
## total number of records
boston_data.shape
```

```
(506, 14)
```

```
boston_data.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
PRICE      0
dtype: int64
```

```
## Create a smaller DataFrame with columns that do not include CHAS,
NOX, B, and LSTAT.
subset_data = boston_data.drop(["CHAS", "NOX", "B", "LSTAT"], axis=1)

## Check the last seven records of the new DataFrame you just created
subset_data.tail(7)
```

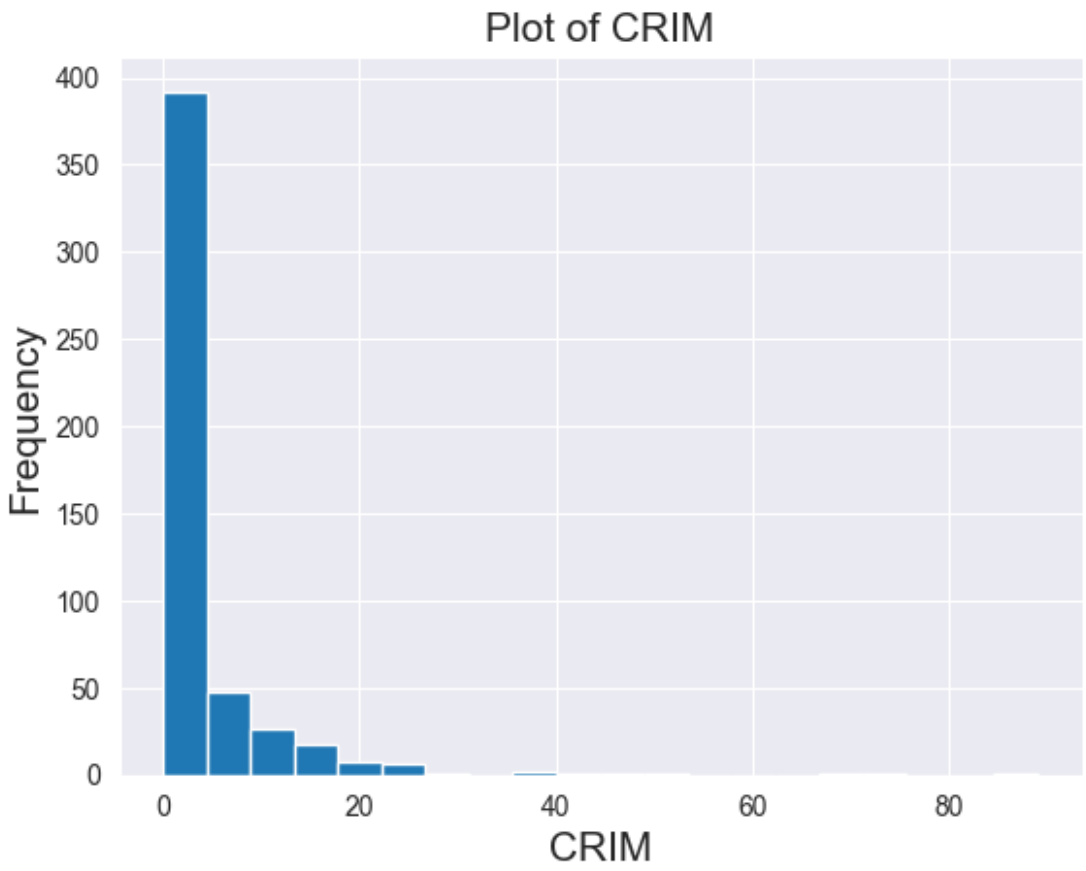
```
        CRIM   ZN  INDUS     RM   AGE     DIS  RAD  TAX  PTRATIO
PRICE
499  0.17783  0.0   9.69  5.569  73.5  2.3999    6  391     19.2
17.5
500  0.22438  0.0   9.69  6.027  79.7  2.4982    6  391     19.2
16.8
501  0.06263  0.0  11.93  6.593  69.1  2.4786    1  273     21.0
22.4
```

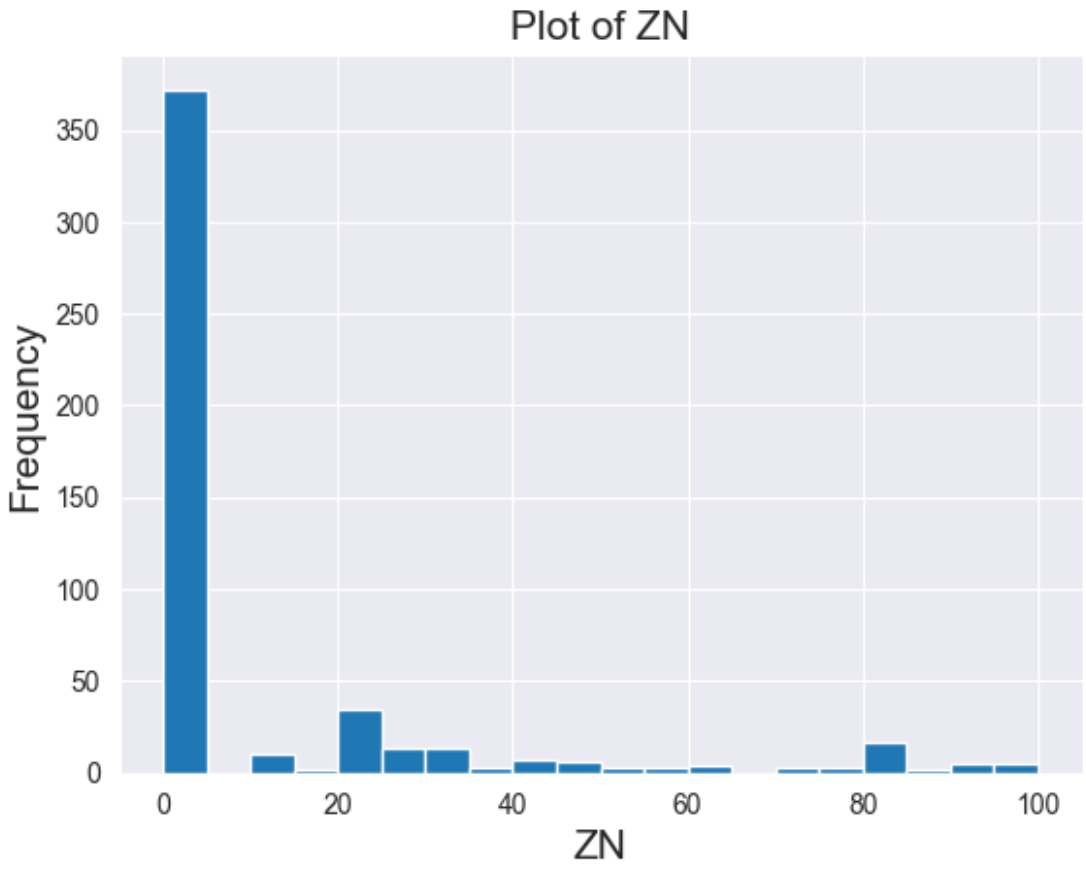
```
502  0.04527  0.0  11.93  6.120  76.7  2.2875    1  273     21.0
20.6
503  0.06076  0.0  11.93  6.976  91.0  2.1675    1  273     21.0
23.9
504  0.10959  0.0  11.93  6.794  89.3  2.3889    1  273     21.0
22.0
505  0.04741  0.0  11.93  6.030  80.8  2.5050    1  273     21.0
11.9
```

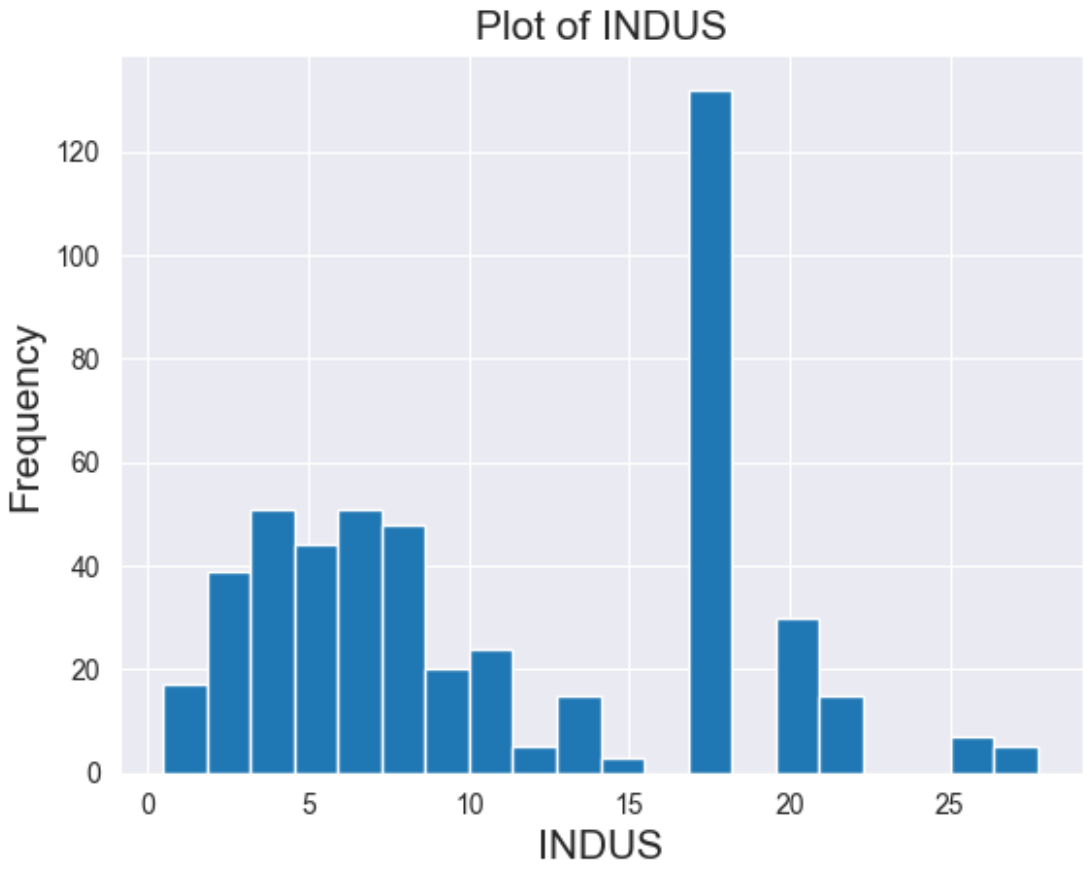
Data Observation: two data coulmns show interesting summeries.

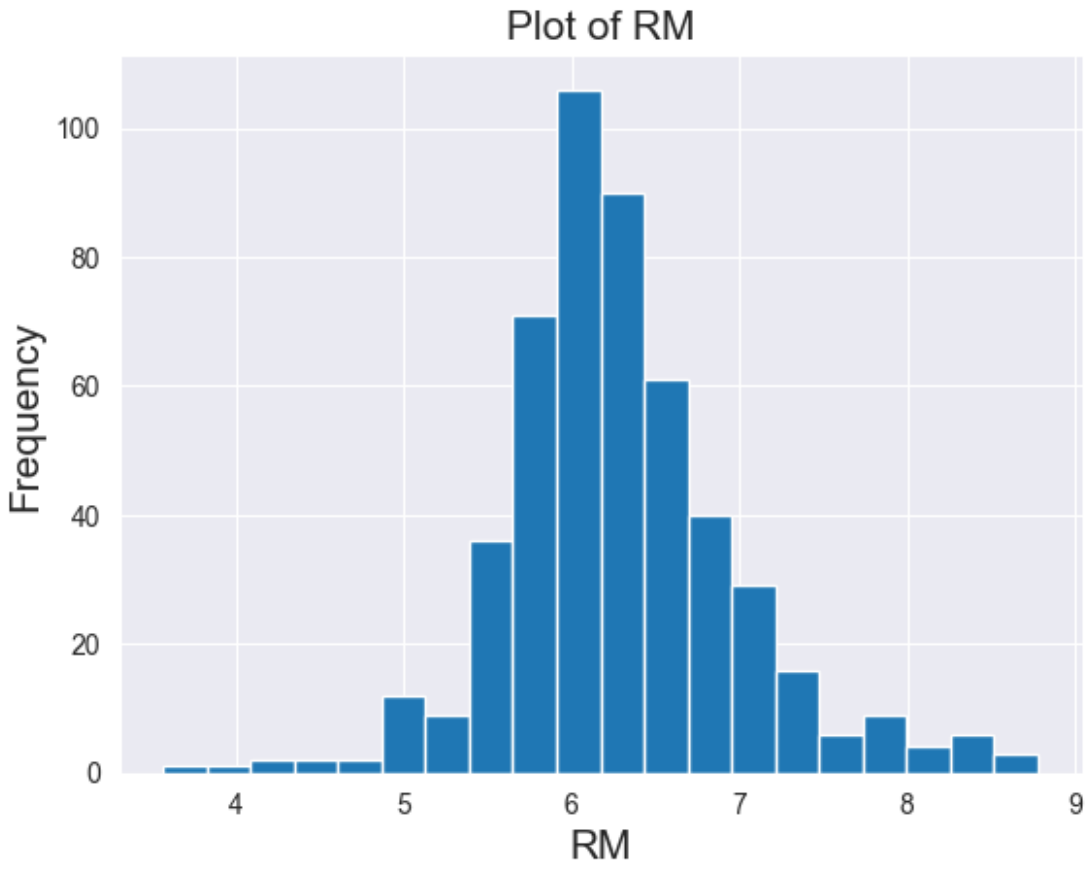
1. ZN (proportion of residential land zoned for lots over 25,000 sq.ft.) with 0 for 25th, 50th percentiles.
2. CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise) with 0 for 25th, 50th and 75th percentiles. These summeries are understandable as both variables are conditional + categorical variables.

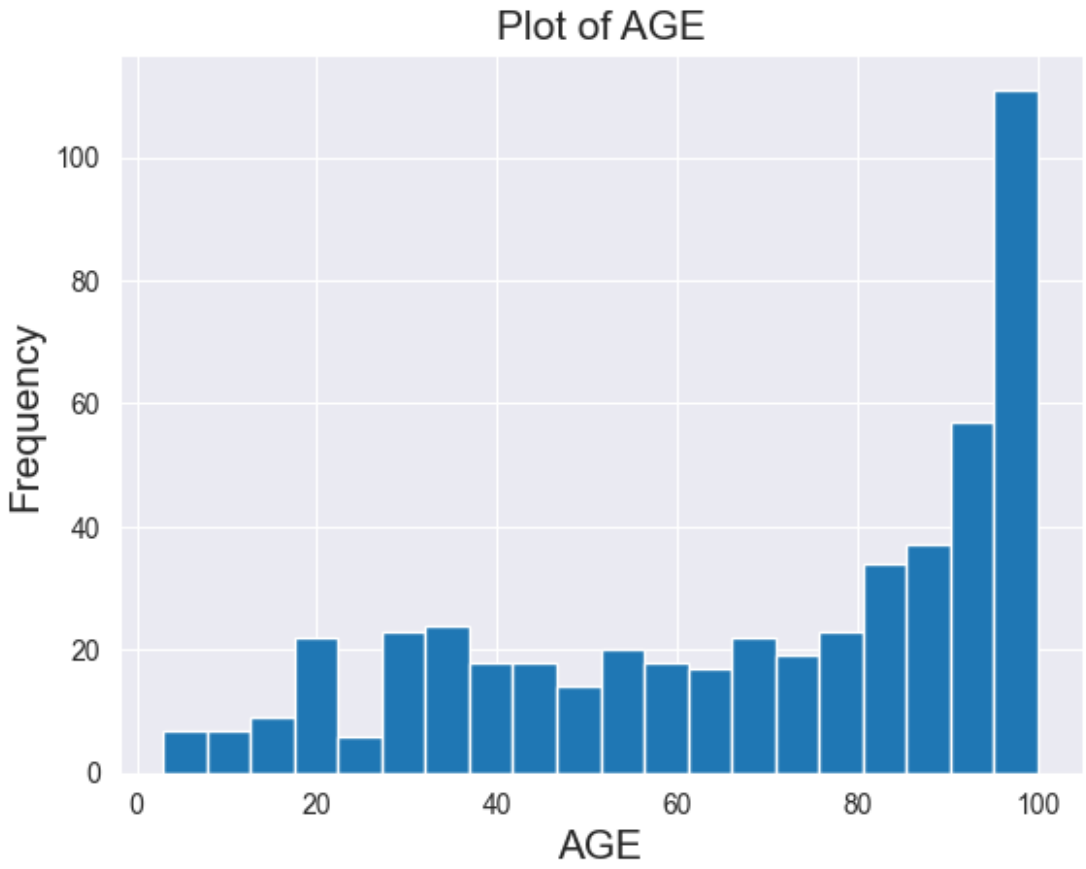
```
# ## Plot the histograms of all the variables (columns) in the new
DataFrame.
# for c in subset_data.columns:
#     plt.figure(figsize=(20,10))
#     plt.title('Plot of '+c)
#     sns.histplot(subset_data[c])
#     plt.xticks(rotation=45)
#     plt.show()

for c in subset_data.columns:
    plt.title("Plot of "+c,fontsize=15)
    plt.grid(True)
    plt.xlabel(c,fontsize=15)
    plt.ylabel("Frequency",fontsize=15)
    plt.hist(subset_data[c],bins=20)
    plt.show()
```

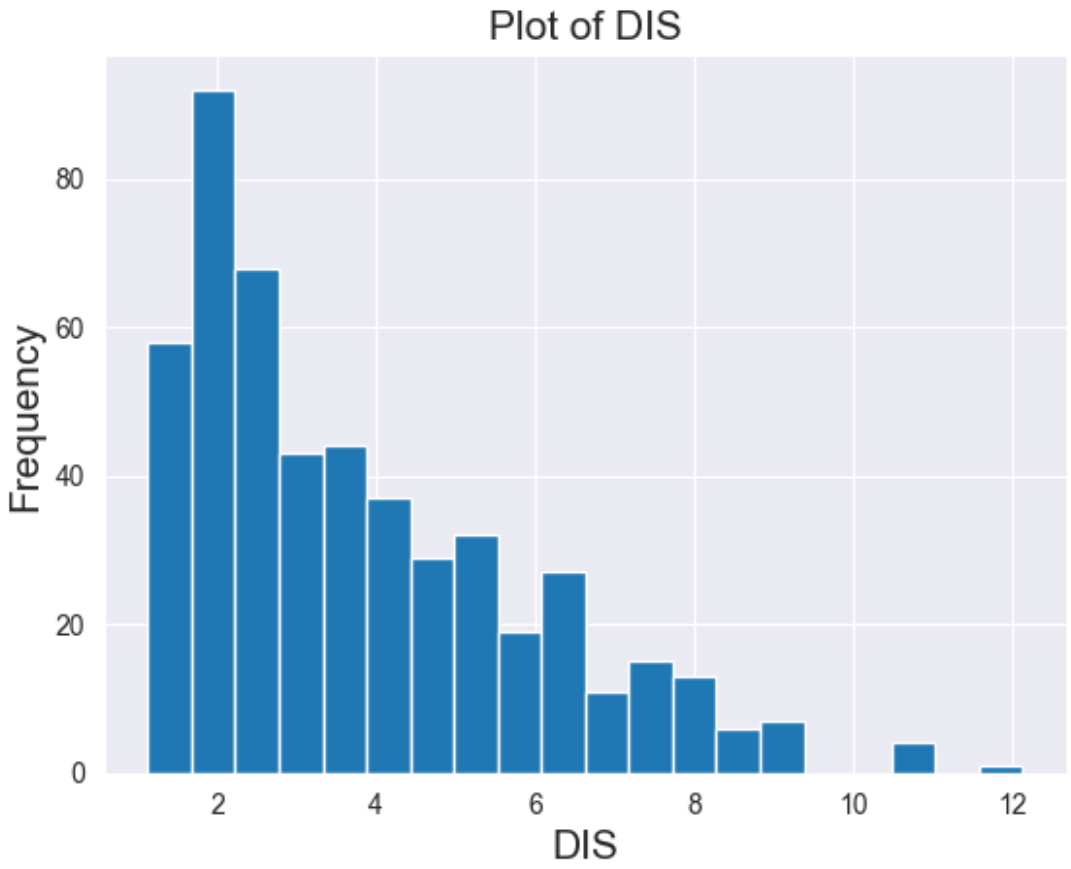

Plot of CRIM
Frequency
400
350
300
250
200
150
100
50
0
0
20
40
60
80
CRIM

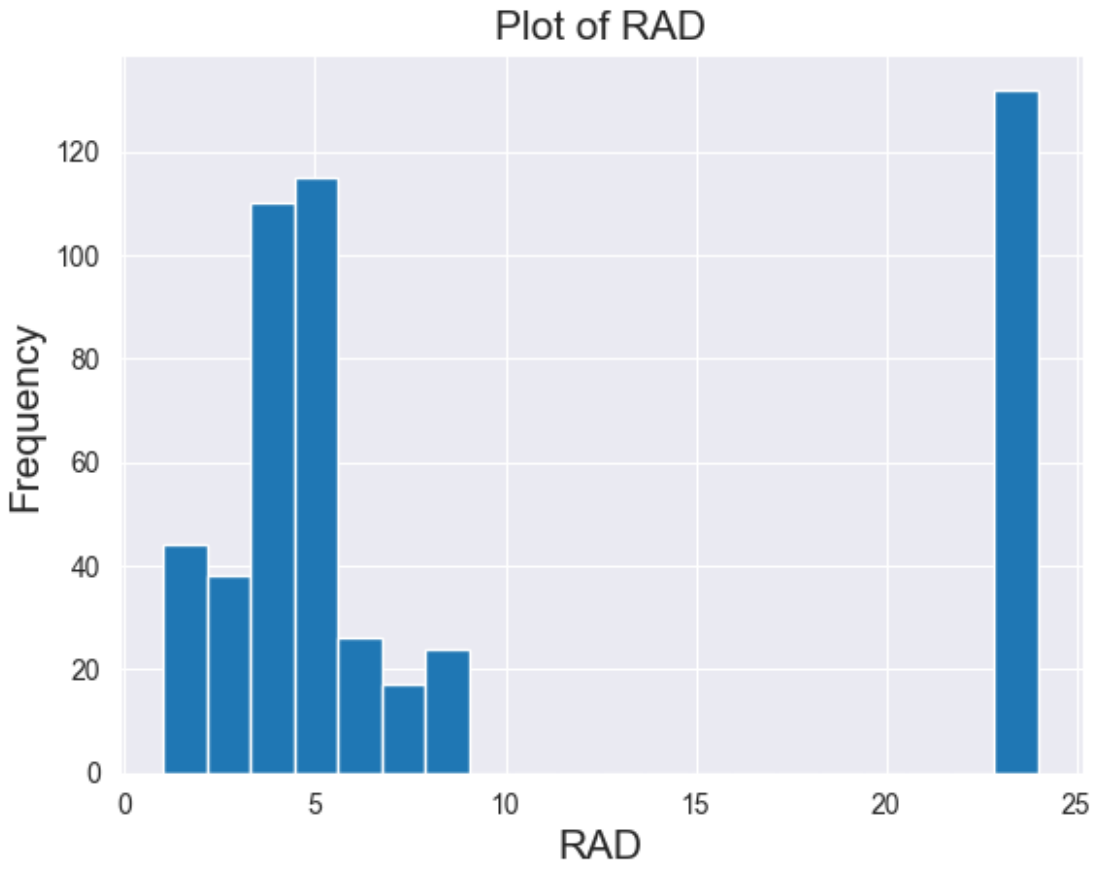

Plot of ZN
Frequency
350
300
250
200
150
100
50
0
0
20
40
60
80
100
ZN

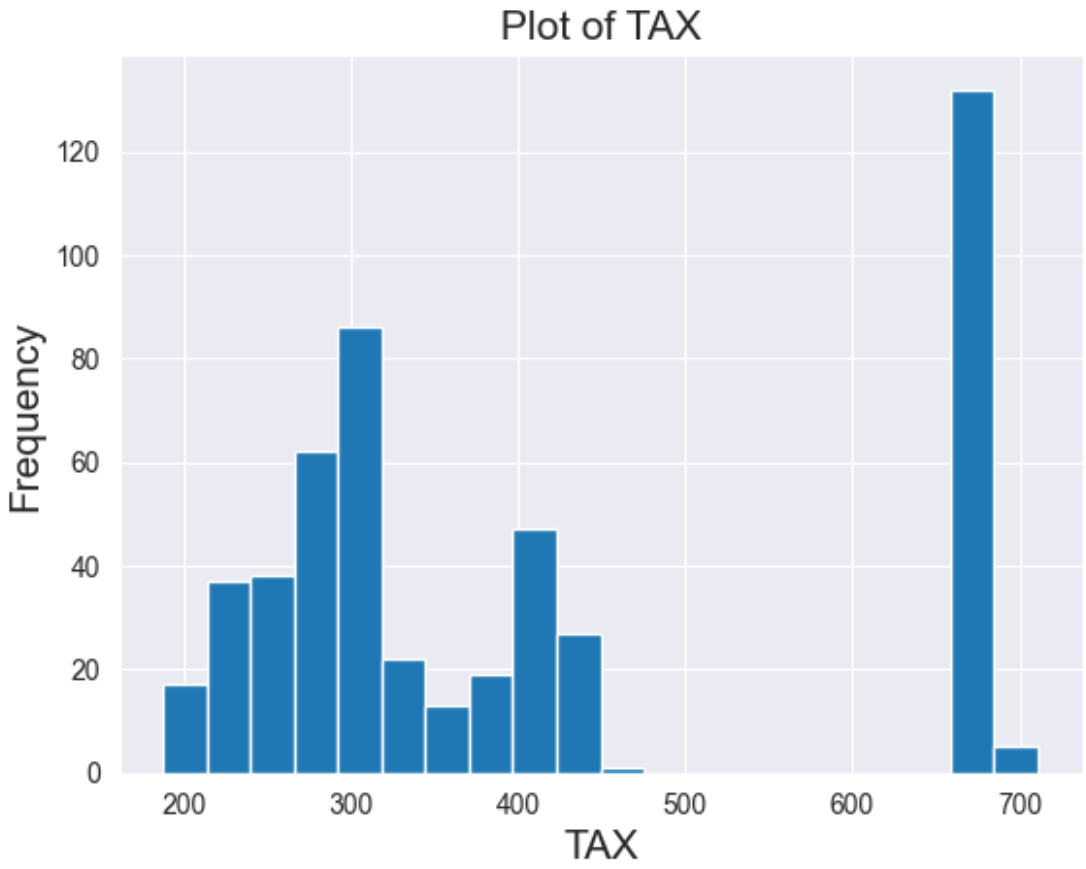

Plot of INDUS
Frequency
120
100
80
60
40
20
0
0
5
10
15
20
25
INDUS

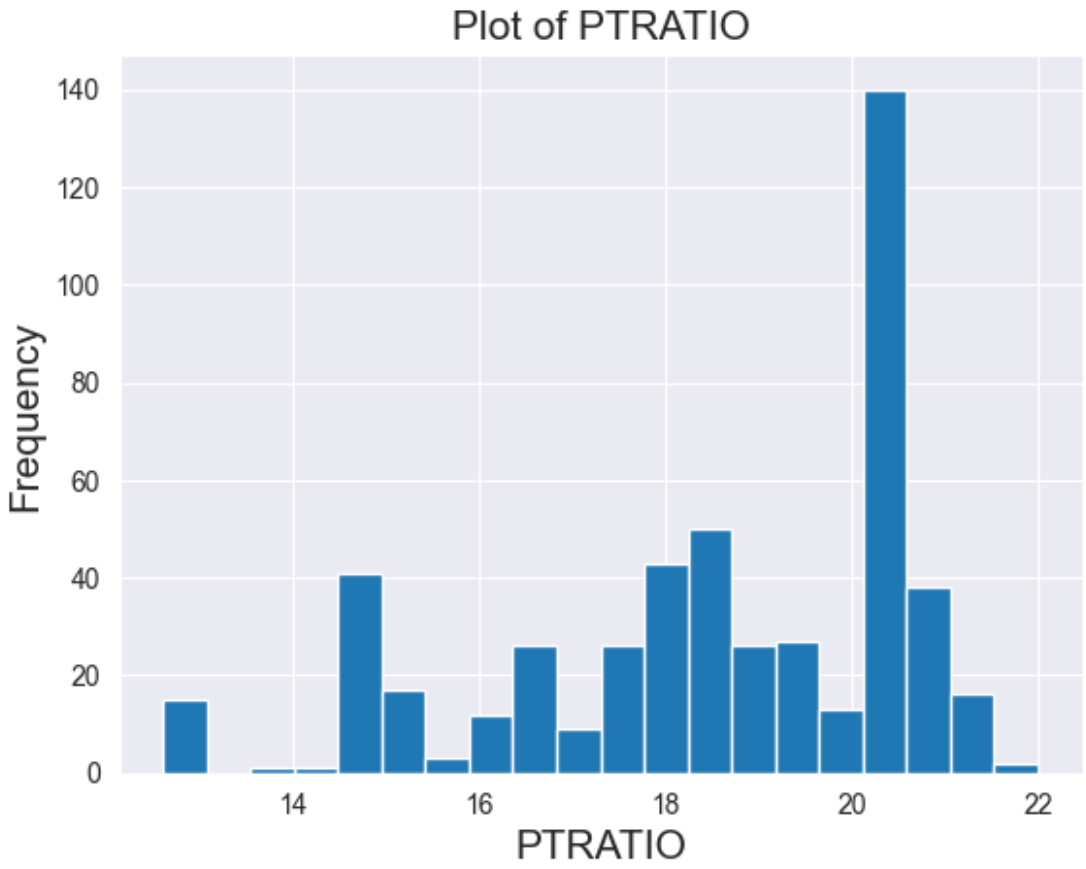

Plot of RM
Frequency
RM
0
20
40
60
80
100
4
5
6
7
8
9

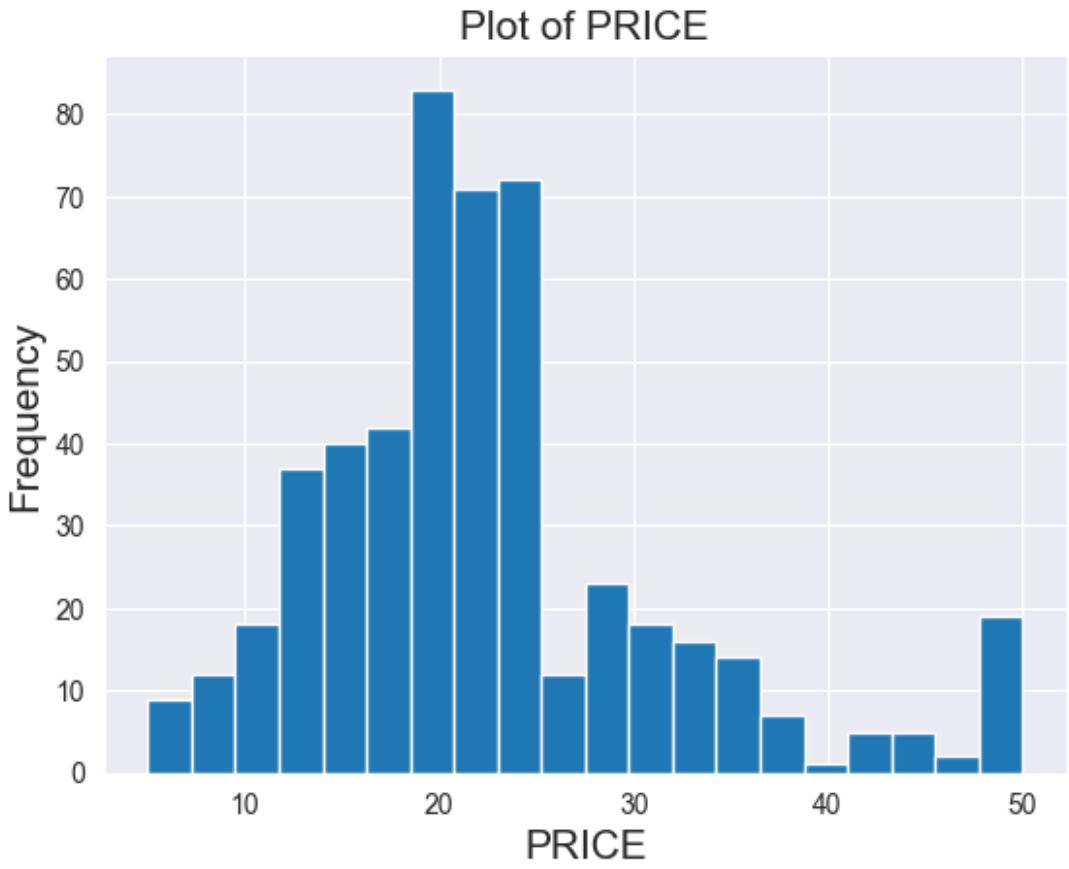

Plot of AGE
Frequency
AGE
0
20
40
60
80
100


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


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


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

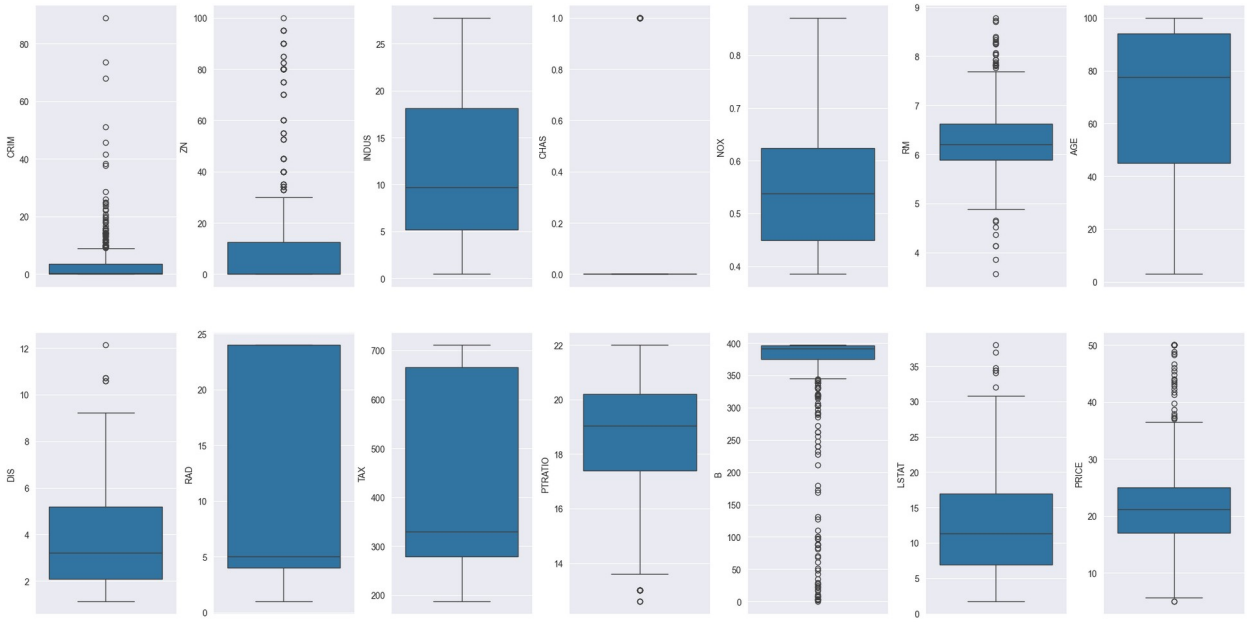

Plot of PTRATIO
Frequency
140
120
100
80
60
40
20
0
14
16
18
20
22
PTRATIO




```
## Create a scatter plot of crime rate versus price
plt.scatter(x=subset_data['CRIM'], y=subset_data['PRICE'])
plt.title("Crime Rate Vs Housing Price Plot", fontsize=15)
plt.xlabel("Crime Rate", fontsize=15)
plt.ylabel("Housing Price", fontsize=15)
plt.grid(True)
plt.show()
```

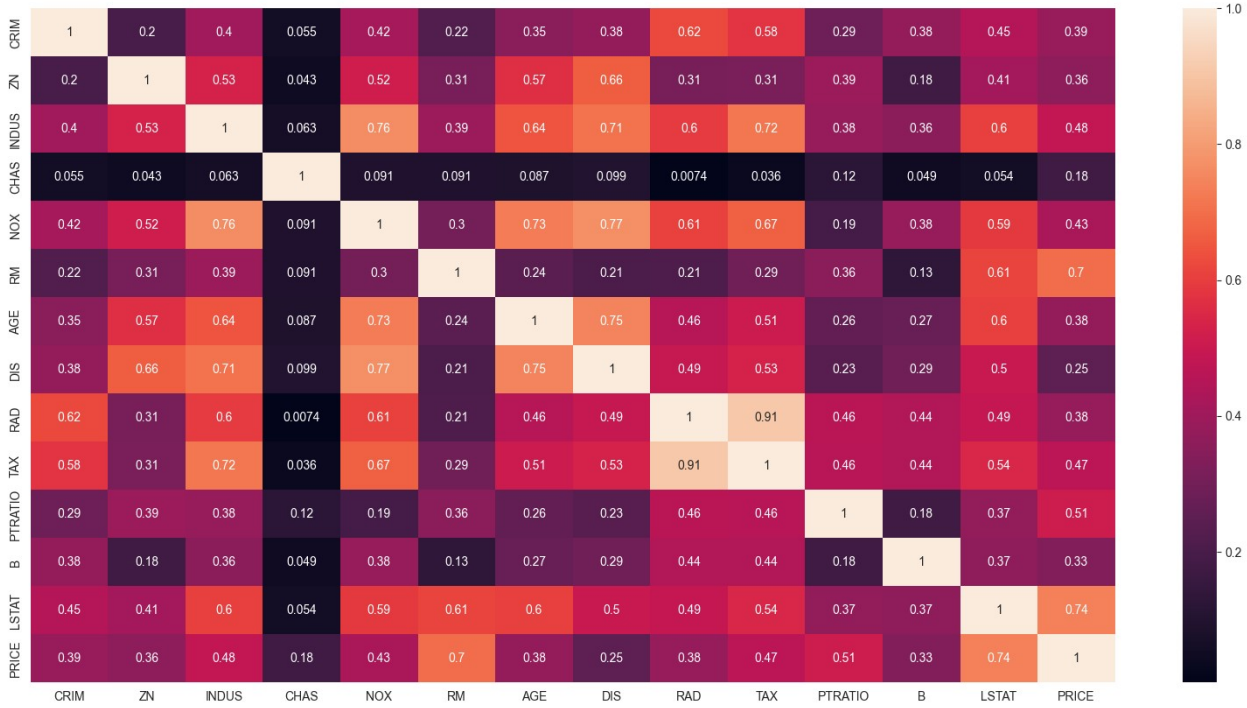

```
## Plot using log10(crime) versus price
plt.scatter(np.log10(subset_data['CRIM']),subset_data['PRICE'],c='red'
)
plt.title('Crime rate (Log) vs. Price plot', fontsize=18)
plt.xlabel('Log of Crime rate',fontsize=30)
plt.ylabel('Price',fontsize=30)
plt.show()
```


```
# create box plots
fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for k,v in boston_data.items():
    sns.boxplot(y=k, data=boston_data, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```




```
# Plotting Heatmap to identify variable behaviour at a threshold of correlation score= 0.5
plt.figure(figsize=(20, 10))
sns.heatmap(boston_data.corr().abs(),  annot=True)
```

```
<Axes: >
```



useful statistics - mean(), median()

```
### mean rooms per dwelling
subset_data['RM'].mean()
```

```
6.284634387351779
```

```
### median age
subset_data['AGE'].median()
```

```
77.5
```

```
### mean distances to five Boston employment centers
subset_data['DIS'].mean()
```

```
3.795042687747036
```

```
### percentage of houses with a low price
low_price=subset_data['PRICE']<20

low_price.mean()*100
```

```
41.50197628458498
```

Data Wrangling with Python: Activity 6

```
# load the packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# Reading csv file
adult_obs = pd.read_csv('./datasets/adult_income_data.csv')
adult_obs.head()

# Create a script that will read a text file line by line
names = []
with open('./datasets/adult_income_names.txt','r') as f:
    for line in f:
        f.readline()
        var=line.split(':')[0]
        names.append(var)

names


# Add a name of Income for the response variable to the dataset
names.append('Income')
names
```

```
adult_data =
pd.read_csv('./datasets/adult_income_data.csv',names=names)

adult_data.head()

# Find the missing values
adult_data.isnull().sum()


# Create a DataFrame with only age, education, and occupation by using
subsetting
adult_subset_data = adult_data[['age', 'education', 'occupation']]



for c in adult_subset_data.columns:

    plt.title('Plot of '+c,fontsize=15)
    plt.hist(adult_subset_data[c],bins=20,facecolor='blue', alpha=0.5)
    plt.show()

# Create a function to strip the whitespace characters
def strip_whitespace(spaces):
    return spaces.strip()


# Education column
adult_subset_data['education_stripped']=adult_data['education'].apply(
strip_whitespace)

adult_subset_data['education']=adult_subset_data['education_stripped']

adult_subset_data.drop(labels=['education_stripped'],axis=1,inplace=Tr
ue)

# Occupation column

adult_subset_data['occupation_stripped']=adult_data['occupation'].appl
y(strip_whitespace)

adult_subset_data['occupation']=adult_subset_data['occupation_stripped
']

adult_subset_data.drop(labels=['occupation_stripped'],axis=1,inplace=T
rue)


# Find the number of people who are aged between ['age']>=30 and
['age']<=50
filtered_data = adult_subset_data[(adult_subset_data['age']>=30) &
```

```
(adult_subset_data['age']<=50)]
filtered_data.shape[0]

# Group the records based on age and education
adult_subset_data.groupby('education').mean()['age']

# Group by occupation and show the summary statistics of age
adult_subset_data.groupby('occupation').describe()['age']

# Use subset and groupby to find outliers
outlier_stats= adult_subset_data.groupby('occupation').describe()
['age']
outlier_stats

# Plot the values on a bar chart
plt.figure(figsize=(15,8))
plt.barh(y=outlier_stats.index, width=outlier_stats['count'])
plt.show()

# Merge the subset data and the original adult data using occupation
as the common key
merged_data_frame = pd.merge(adult_data,adult_subset_data,
on='occupation', how='inner')
merged_data_frame.head()

# Question 3.a
import pandas as pd
series1 = {'a':7.3, 'b':-2.5, 'c':3.4, 'd':1.5}
series1 = pd.Series(data=series1, index=['a', 'b', 'c', 'd'])
series1

# Question 3.b
series2 = {'a':-2.1, 'c':3.6, 'e':-1.5, 'f':4, 'g':3.1}
series2 = pd.Series(data=series2, index=['a', 'c', 'e', 'f', 'g'])
series2

## Question 3.c adding two series
total = series1 + series2
print(total)

## Question 3.d subtracting two series
diff = series2 - series1
print(diff)
```