Data Wrangling with Python-Activity 5

July 2, 2023

[51]:
```
# Carlos Cano
# DSC 540
# Activity 5
```

[3]:
```
## Import needed library for necessary functions for assignment
```

[18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[5]:
```
## Read CSV file into Notebook
```

[6]:
```
df = pd.read_csv("Boston_housing.csv")
```

[ ]:
```
## Read First Data Rows
```

[7]:
```
df.head(10)
```

[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
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			

```
7  396.90  19.15   27.1
8  386.63  29.93   16.5
9  386.71  17.10   18.9
```

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

[8]:
```
df.shape
```

[8]:
```
(506, 14)
```

[9]:
```
# Remove Variables from Data
```

[10]:
```
df1 = df[['CRIM','ZN','INDUS','RM','AGE','DIS','RAD','TAX','PTRATIO','PRICE']]
```

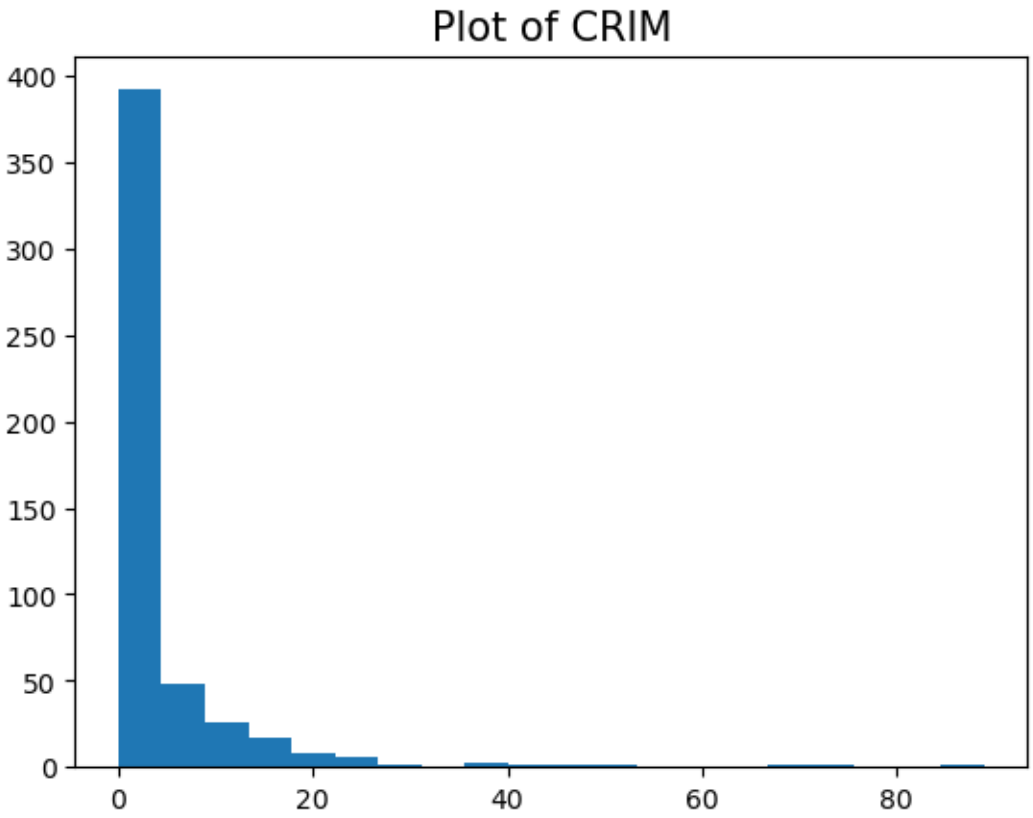
[11]:
```
df1.tail(7)
```

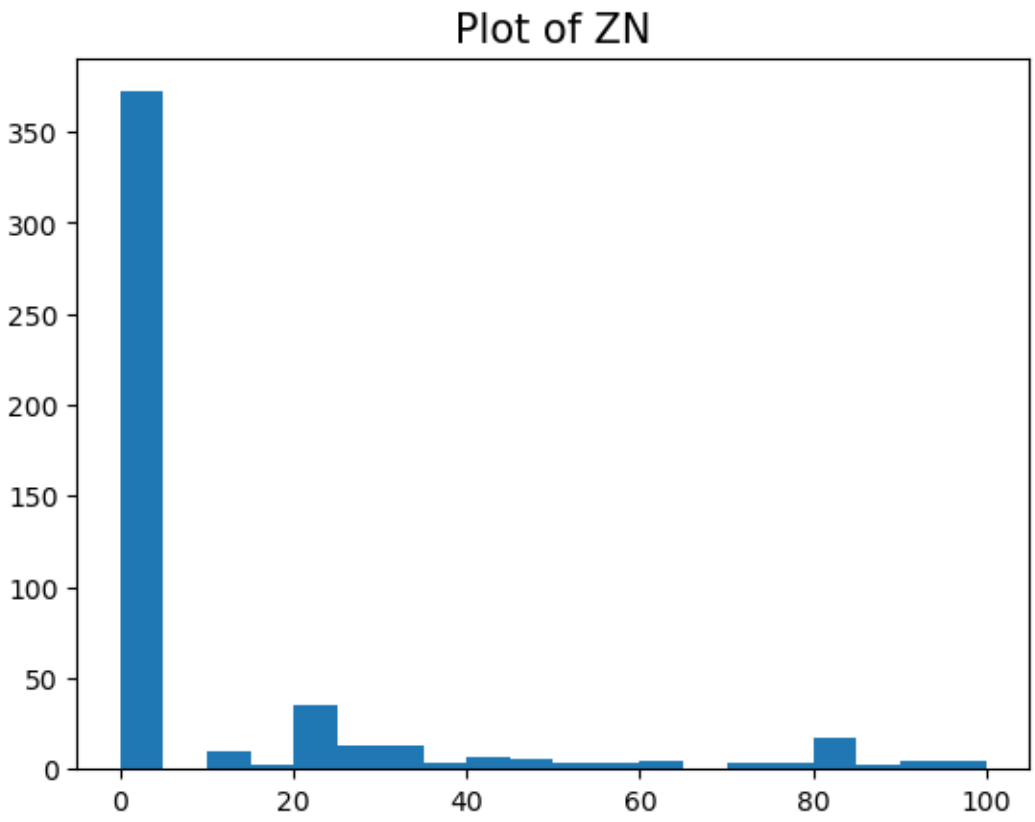
[11]:

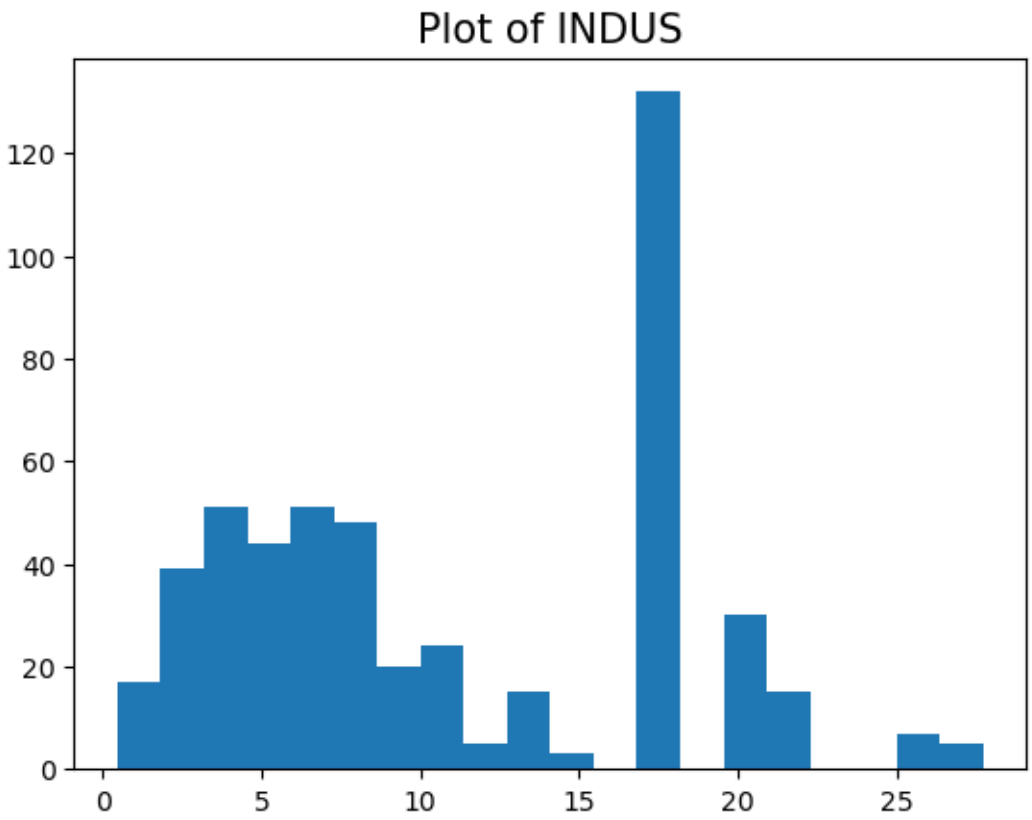
	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
499	0.17783	0.0	9.69	5.569	73.5	2.3999	6	391	19.2	17.5
500	0.22438	0.0	9.69	6.027	79.7	2.4982	6	391	19.2	16.8
501	0.06263	0.0	11.93	6.593	69.1	2.4786	1	273	21.0	22.4
502	0.04527	0.0	11.93	6.120	76.7	2.2875	1	273	21.0	20.6
503	0.06076	0.0	11.93	6.976	91.0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

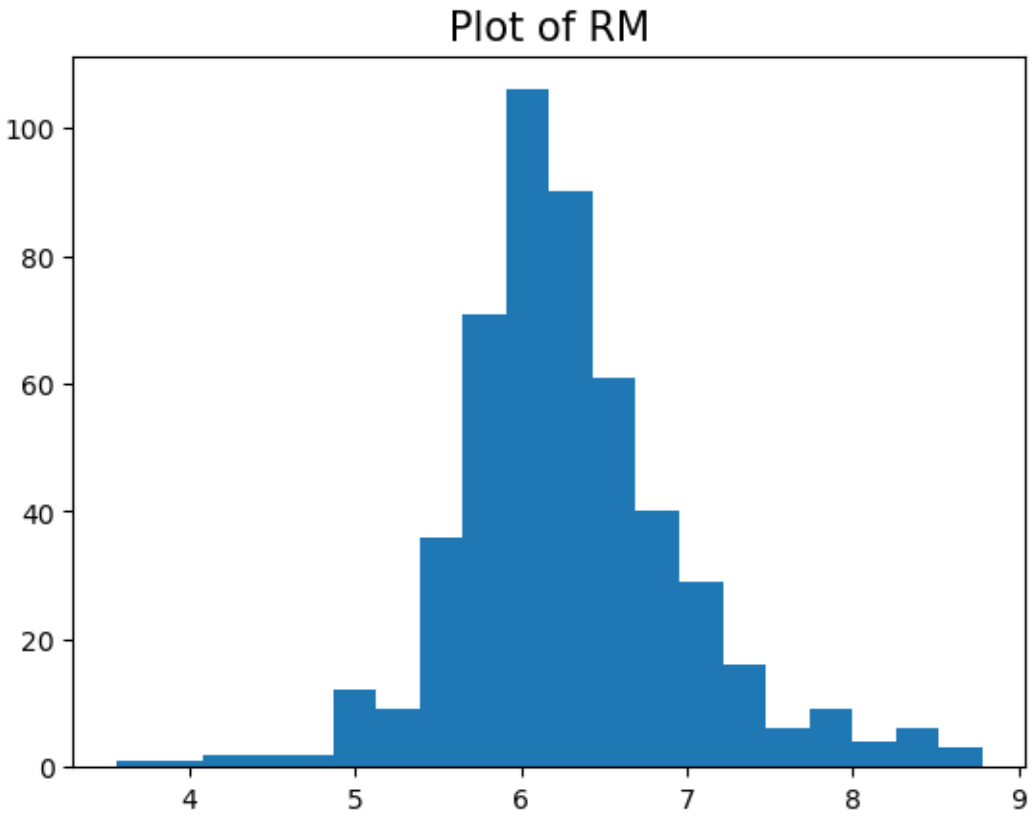
[12]:
```
## Plot Histograms of all variables
```

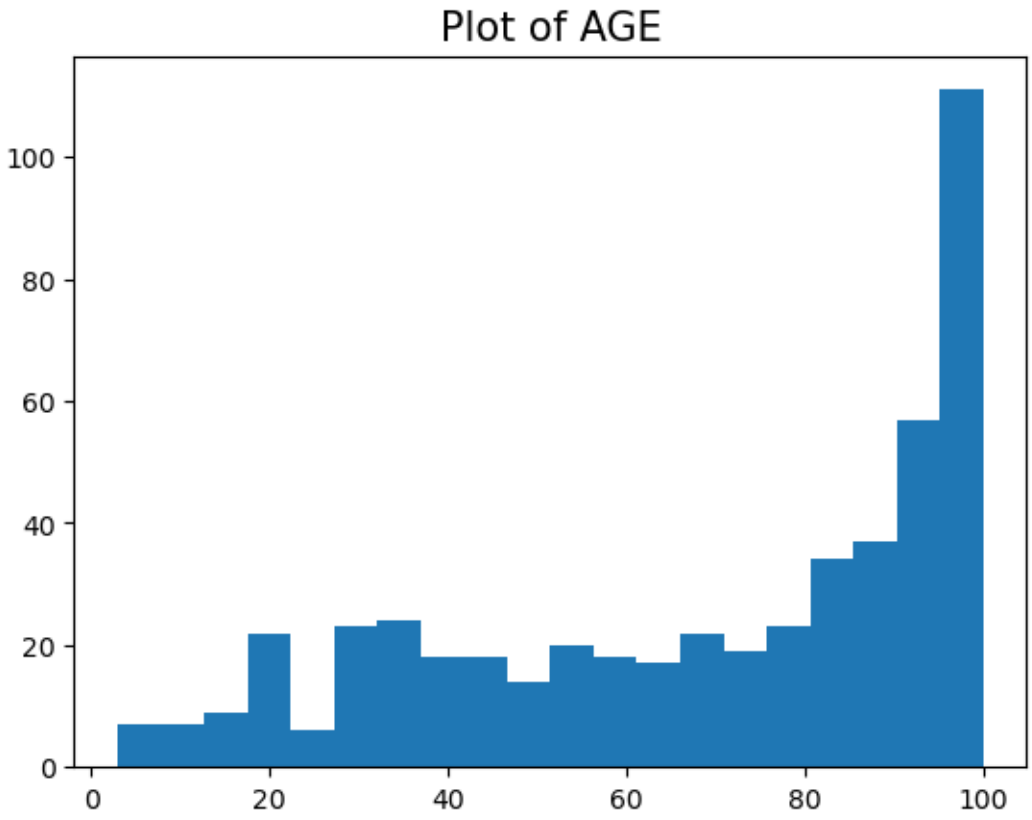
[24]:
```
for c in df1.columns:
    plt.title("Plot of " + c,fontsize=15)
    plt.hist(df1[c], bins=20)
    plt.show()
```

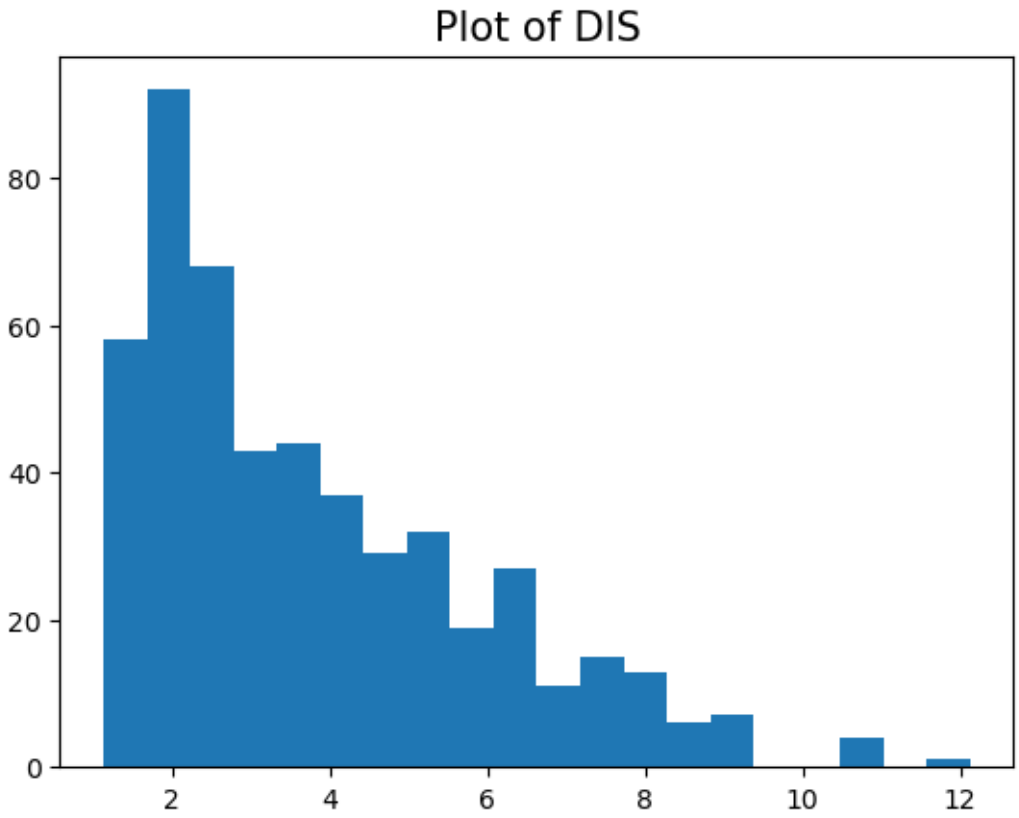

Plot of CRIM
400
350
300
250
200
150
100
50
0
0
20
40
60
80

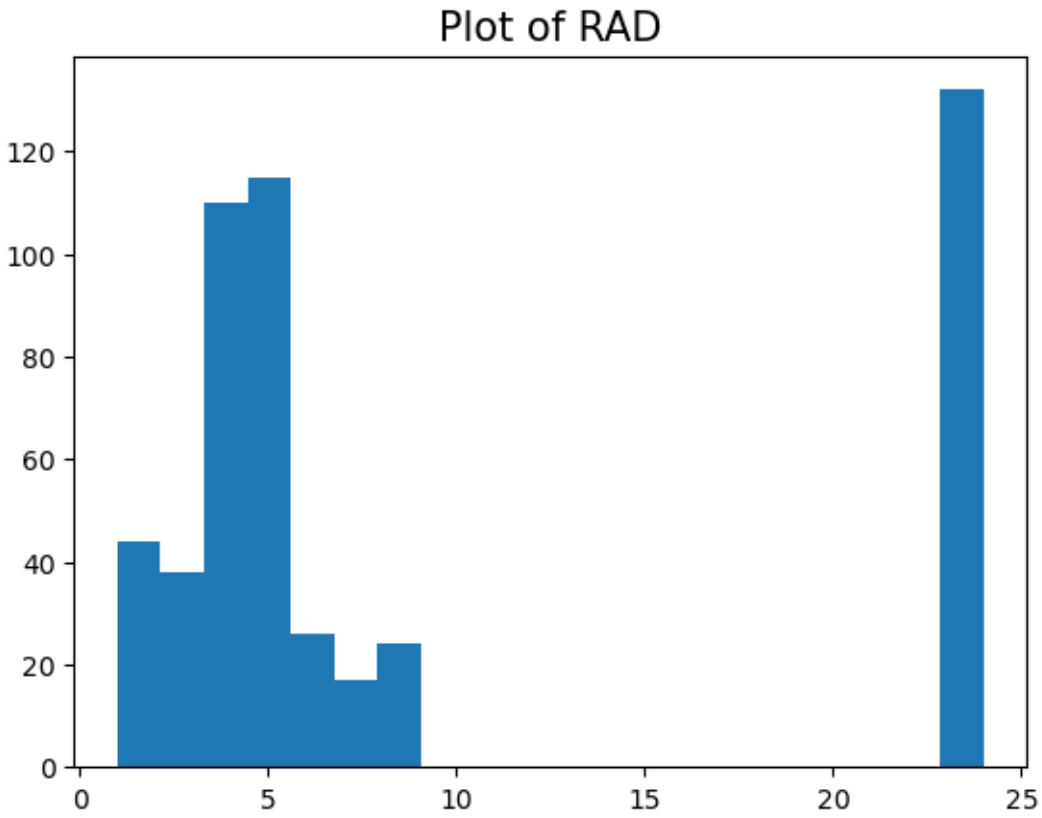

Plot of ZN
350
300
250
200
150
100
50
0
0
20
40
60
80
100

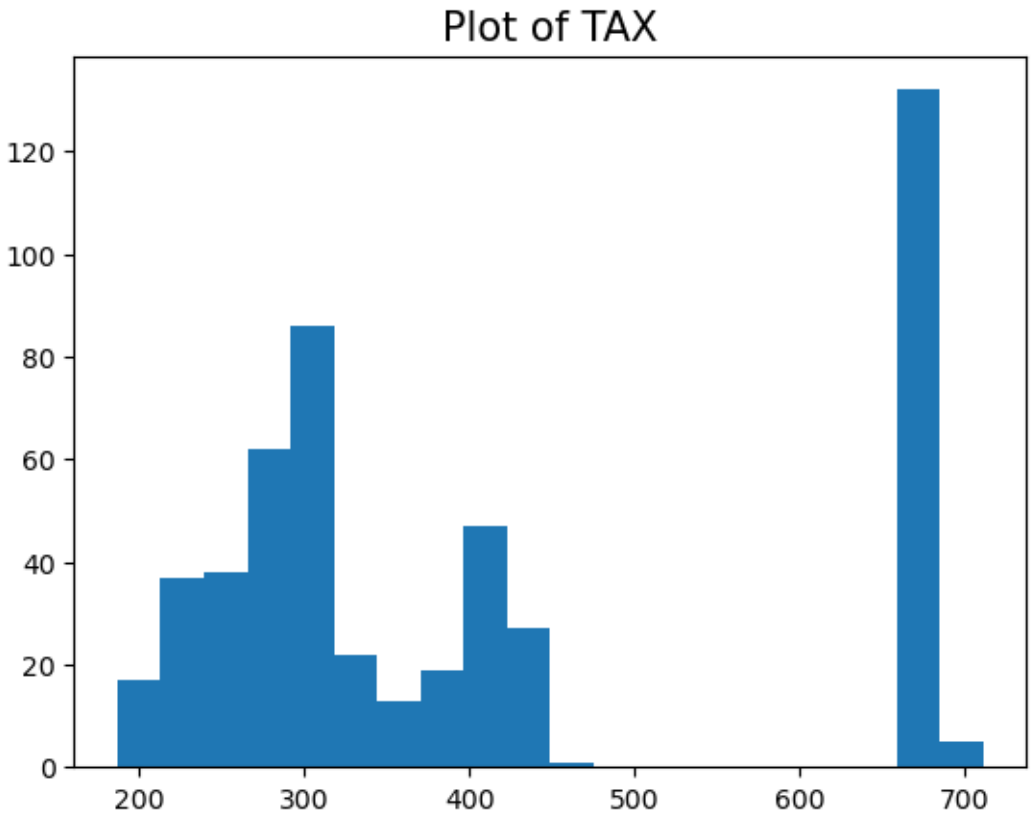

Plot of INDUS
120
100
80
60
40
20
0
0
5
10
15
20
25

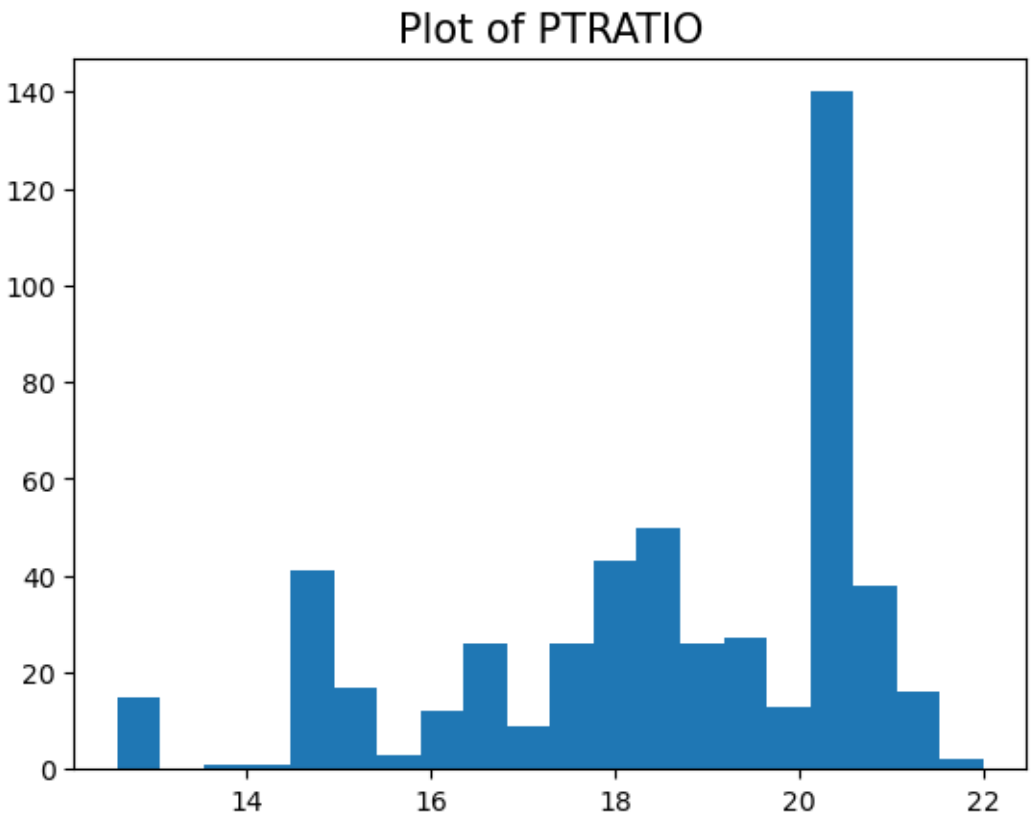

Plot of RM
100
80
60
40
20
0
4
5
6
7
8
9

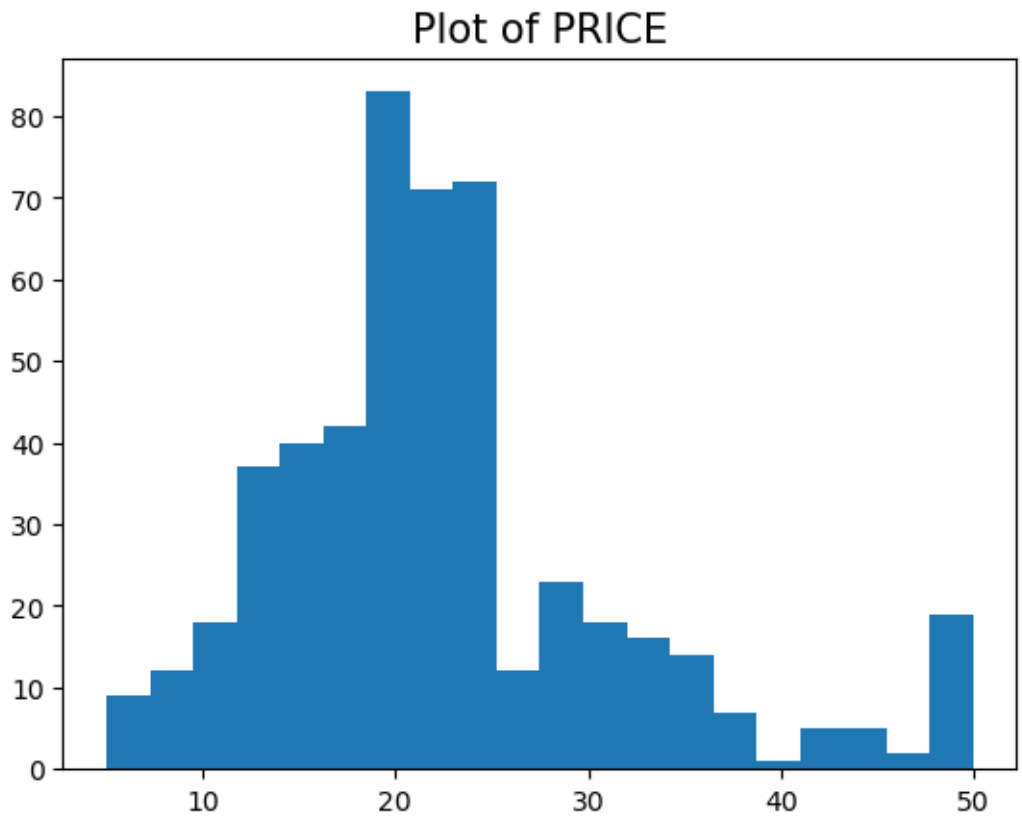

Plot of AGE
100
80
60
40
20
0
0
20
40
60
80
100


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


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

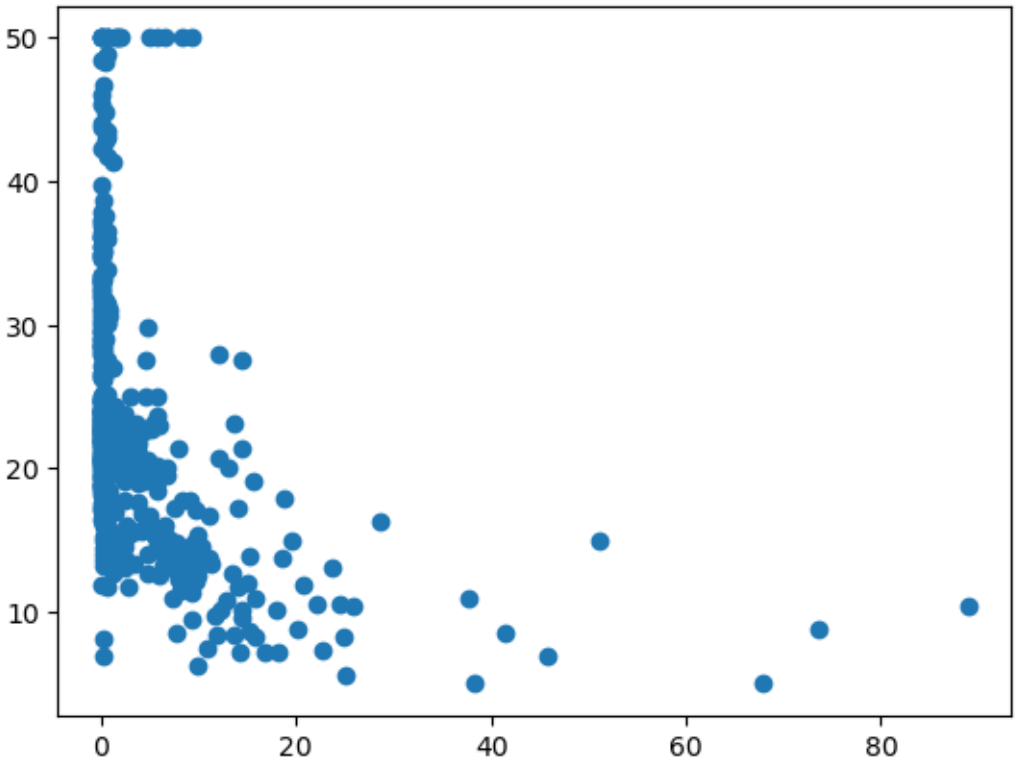

Plot of TAX
120
100
80
60
40
20
0
200
300
400
500
600
700


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



[25]:
```
## Scatterplot Crime Rate vs  Price
```

[34]:
```
plt.scatter(df1['CRIM'],df1['PRICE'])
plt.show()
```



[52]:
```
## Plot using log10(crime) versus Price
```

[50]:
```
plt.scatter(np.log10(df1['CRIM']),df1['PRICE'],c='red')
plt.title("Crime rate (Log) vs Price plot", fontsize=12)
plt.xlabel("Log of Crime Rate")
plt.ylabel("Price")
plt.grid(True)
plt.show()
```

[ ]:
```
## Calc Mean Room per Dwelling
```

[42]:
```
df1['RM'].mean()
```

[42]: 6.284634387351787

[43]:
```
## Calc Median Age
```

[44]:
```
df1['AGE'].median()
```

[44]: 77.5

[46]:
```
## Calc mean distance to five Boston Employment Center
```

[47]:
```
df1['DIS'].mean()
```

[47]: 3.795042687747034

[ ]:
```

```

Data Wrangling with Python-Activity 6

July 2, 2023

[117]:
```
# Carlos Cano
# DSC 540
# Activity 6
```

[118]:
```
## Load Libraries
```

[119]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[120]:
```
## Import Data & Read head
```

[121]:
```
df=pd.read_csv("adult_income_data.csv")
df.head()
```

[121]:

	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

[122]:
```
## Read Text File with first line extraction
```

[123]:
```
names = []
with open ('adult_income_names.txt','r') as f:
    for line in f:
        f.readline()
        var=line.split(":")[0]
        names.append(var)
names
```

[123]:
```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'sex',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country']
```

[124]:
```
## Add Income Variable to Dataset
```

[125]:
```
names.append('Income')
```

[126]:
```
## Import data to variable df & Read
```

[127]:
```
df = pd.read_csv("adult_income_data.csv",names=names)
df.head()
```

[127]:

	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
```

[128]:
```
## Gather Basic Data
```

[129]:
```
df.describe()
```

[129]:
```
                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
```

[130]:
```
## Create Subset of Data
```

[131]:
```
vars_class =␣
↪['workclass','education','marital-status','occupation','relationship','sex','native-country']
```

[132]:
```
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'
 ' Holand-Netherlands']
--------------------------------------------------------------------------------
--------------------
```

[133]:
```
## Check for missing data
```

[134]:
```
df.isnull().sum()
```

[134]:
```
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
```

[135]:
```
## Create subset
```

[136]:
```
df_subset = df[['age','education','occupation']]
df_subset.head()
```

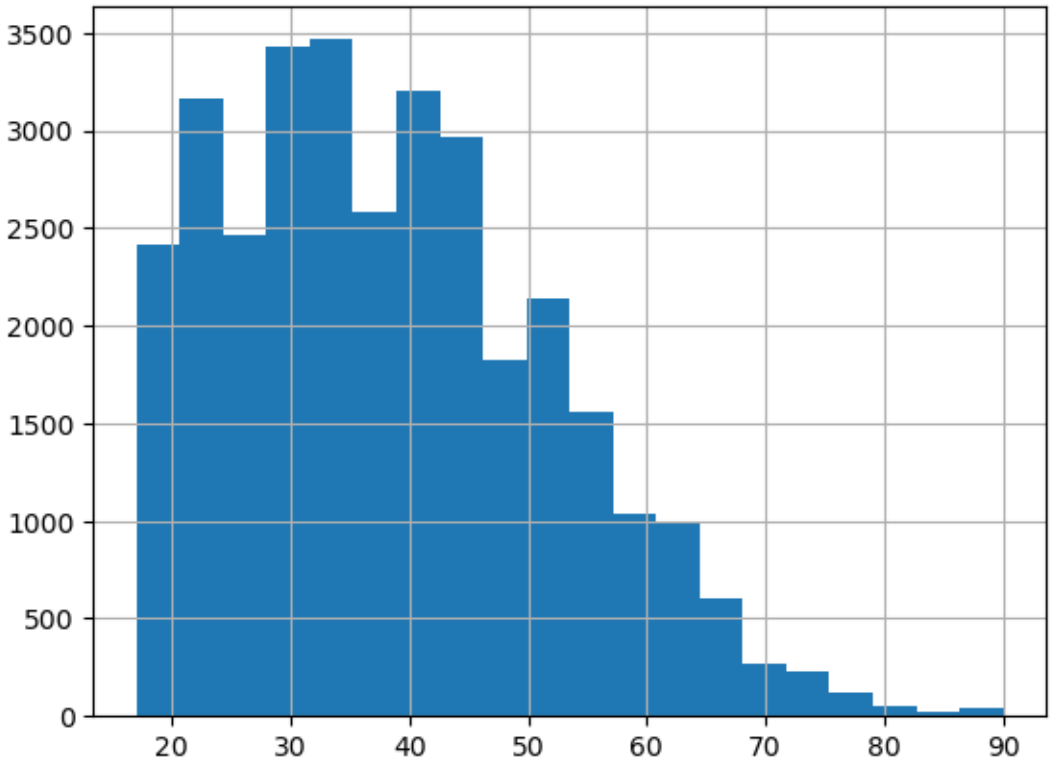
[136]:

	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

[137]:
```
## Visualize Age Histogram
```

[138]:
```
df_subset['age'].hist(bins=20)
```

[138]:
```
<AxesSubplot:>
```



[139]:
```
## Strip Function
```

[140]:
```
def strip_whitespace(s):
    return s.strip()
```

[141]:
```
## Strip Columns
```

[142]:
```
# 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)
```

```
/var/folders/q7/c71x7n2901x74v60v338kcfm0000gn/T/ipykernel_40434/1940179211.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
```

```
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset['education_stripped']=df['education'].apply(strip_whitespace)
/var/folders/q7/c71x7n2901x74v60v338kcfm0000gn/T/ipykernel_40434/1940179211.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset['education']=df_subset['education_stripped']
/var/folders/q7/c71x7n2901x74v60v338kcfm0000gn/T/ipykernel_40434/1940179211.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset.drop(labels=['education_stripped'],axis=1,inplace=True)
/var/folders/q7/c71x7n2901x74v60v338kcfm0000gn/T/ipykernel_40434/1940179211.py:7
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset['occupation_stripped']=df['occupation'].apply(strip_whitespace)
/var/folders/q7/c71x7n2901x74v60v338kcfm0000gn/T/ipykernel_40434/1940179211.py:8
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset['occupation']=df_subset['occupation_stripped']
/var/folders/q7/c71x7n2901x74v60v338kcfm0000gn/T/ipykernel_40434/1940179211.py:9
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_subset.drop(labels=['occupation_stripped'],axis=1,inplace=True)
```

[143]:
```
## Above Error Expected
```

[144]:
```
## Filter for age between 30 & 50
```

[145]:
```
df_filtered=df_subset[(df_subset['age']>=30) & (df_subset['age']<=50)]
```

[146]:
```
## Read Output
```

[147]:
```
df_filtered.head()
```

[147]:

	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

[148]:
```
## set filtered data to variable specific to shape
```

[149]:
```
answer_1=df_filtered.shape[0]
```

[150]:
```
answer_1
```

[150]: 16390

[151]:
```
print("There are {} people between the age of 30 and 50 in this dataset.".
 ↪format(answer_1))
```

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

[152]:
```
## Express group of education by age
```

[153]:
```
df_subset.groupby('education').describe()['age']
```

[153]:

education	count	mean	std	min	25%	50%	75%	max
10th	933.0	37.429796	16.720713	17.0	22.00	34.0	52.0	90.0
11th	1175.0	32.355745	15.545485	17.0	18.00	28.0	43.0	90.0
12th	433.0	32.000000	14.334625	17.0	19.00	28.0	41.0	79.0
1st-4th	168.0	46.142857	15.615625	19.0	33.00	46.0	57.0	90.0
5th-6th	333.0	42.885886	15.557285	17.0	29.00	42.0	54.0	84.0
7th-8th	646.0	48.445820	16.092350	17.0	34.25	50.0	61.0	90.0
9th	514.0	41.060311	15.946862	17.0	28.00	39.0	54.0	90.0
Assoc-acdm	1067.0	37.381443	11.095177	19.0	29.00	36.0	44.0	90.0
Assoc-voc	1382.0	38.553546	11.631300	19.0	30.00	37.0	46.0	84.0
Bachelors	5355.0	38.904949	11.912210	19.0	29.00	37.0	46.0	90.0
Doctorate	413.0	47.702179	11.784716	24.0	39.00	47.0	55.0	80.0
HS-grad	10501.0	38.974479	13.541524	17.0	28.00	37.0	48.0	90.0
Masters	1723.0	44.049913	11.068935	18.0	36.00	43.0	51.0	90.0
Preschool	51.0	42.764706	15.126914	19.0	31.00	41.0	53.5	75.0
Prof-school	576.0	44.746528	11.962477	25.0	36.00	43.0	51.0	90.0
Some-college	7291.0	35.756275	13.474051	17.0	24.00	34.0	45.0	90.0

[154]:
```
## Express group of occupation by age
```

[155]:
```
df_subset.groupby('occupation').describe()['age']
```

[155]:

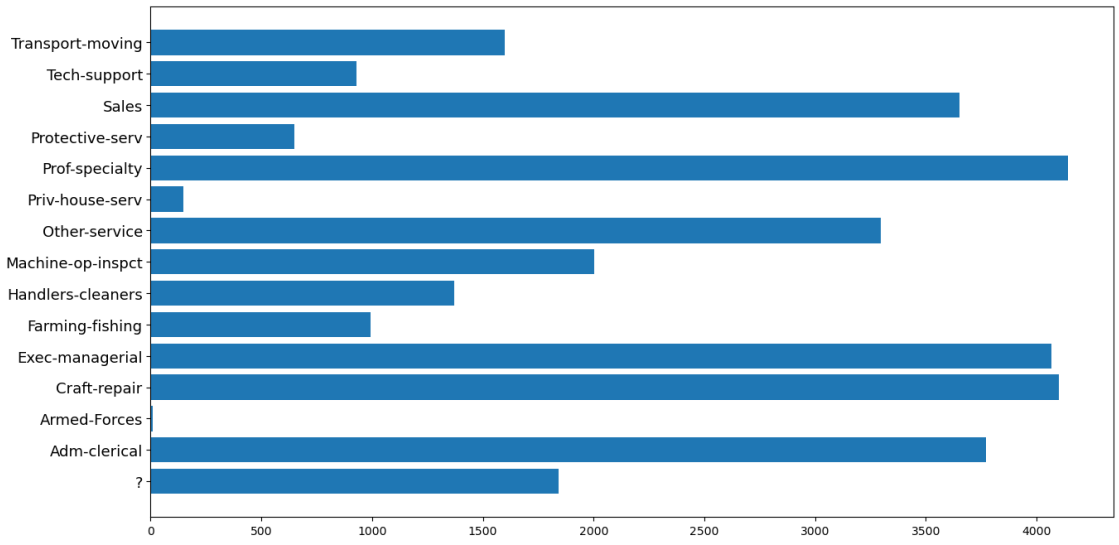
occupation	count	mean	std	min	25%	50%	75%	max
?	1843.0	40.882800	20.336350	17.0	21.0	35.0	61.0	90.0
Adm-clerical	3770.0	36.964456	13.362998	17.0	26.0	35.0	46.0	90.0
Armed-Forces	9.0	30.222222	8.089774	23.0	24.0	29.0	34.0	46.0
Craft-repair	4099.0	39.031471	11.606436	17.0	30.0	38.0	47.0	90.0
Exec-managerial	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

[156]:
```
## assign previous groupby to variable for plotting
```

[157]:
```
occupation_stats= df_subset.groupby('occupation').describe()['age']
```

[158]:
```
## plotted data
```

[159]:
```
plt.figure(figsize=(15,8))
plt.barh(y=occupation_stats.index,width=occupation_stats['count'])
plt.yticks(fontsize=13)
plt.show()
```



[160]:
```
## subset data & read
```

[161]:
```
df_1 = df[['age','workclass','occupation']].sample(5,random_state=101)
```

[162]:
```
df_1.head()
```

[162]:

	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

[163]:
```
## subset data & read
```

[164]:
```
df_2 = df[['education','occupation']].sample(5,random_state=101)
```

[165]:
```
df_2
```

[165]:

	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

[166]:
```
## merge data & print
```

[167]:
```
df_merged = pd.merge(df_1,df_2,on='occupation',how='inner').drop_duplicates()
```

[168]:
```
df_merged
```

[168]:

	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

[ ]:
```

```

Data Wrangling with Python-Activity 7

July 2, 2023

[69]:
```
# Carlos Cano
# DSC 540
# Activity 7
```

[70]:
```
## Import libraries
```

[71]:
```
from bs4 import BeautifulSoup
import pandas as pd
```

[72]:
```
## Open the Wikipedia file
```

[73]:
```
fd = open("List of countries by GDP (nominal) - Wikipedia.htm", "rb")
soup = BeautifulSoup(fd)
fd.close()
```

[74]:
```
## Calculate the tables
```

[75]:
```
all_tables = soup.find_all("table")
print("Total number of tables are {} ".format(len(all_tables)))
```

```
Total number of tables are 9
```

[76]:
```
## Find the class
```

[77]:
```
data_table = soup.find("table", {"class": '"wikitable"|}'})
print(type(data_table))
```

```
<class 'bs4.element.Tag'>
```

[78]:
```
## Separate the source & actual data
```

[79]:
```
sources = data_table.tbody.findAll('tr', recursive=False)[0]
sources_list = [td for td in sources.findAll('td')]
print(len(sources_list))
```

```
3
```

[80]:
```
## Find all function
```

[81]:
```
data = data_table.tbody.findAll('tr', recursive=False)[1].findAll('td',␣
↪recursive=False)
```

[82]:
```
## Find all function
```

[83]:
```
data_tables = []
for td in data:
    data_tables.append(td.findAll('table'))
```

[84]:
```
## Find the length of the table
```

[85]:
```
len(data_tables)
```

[85]: 3

[86]:
```
## set findAll data to variable and print for data sources
```

[87]:
```
source_names = [source.findAll('a')[0].getText() for source in sources_list]
print(source_names)
```

```
['International Monetary Fund', 'World Bank', 'United Nations']
```

[88]:
```
## pull data header info
```

[89]:
```
header1 = [th.getText().strip() for th in data_tables[0][0].findAll('thead')[0].
↪findAll('th')]
header1
```

[89]: ['Rank', 'Country', 'GDP(US$MM)']

[90]:
```
rows1 = data_tables[0][0].findAll('tbody')[0].findAll('tr')[1:]
```

[91]:
```
data_rows1 = [[td.get_text().strip() for td in tr.findAll('td')] for tr in rows1]
```

[92]:
```
## assign data from pull into datafram & print
```

[93]:
```
df1 = pd.DataFrame(data_rows1, columns=header1)
```

[94]:
```
df1.head()
```

[94]:

	Rank	Country	GDP(US$MM)
0	1	United States	19,390,600
1	2	China[n 1]	12,014,610
2	3	Japan	4,872,135
3	4	Germany	3,684,816
4	5	United Kingdom	2,624,529

[95]:
```
## pull data header info
```

[96]:
```
header2 = [th.getText().strip() for th in data_tables[1][0].findAll('thead')[0].
↪findAll('th')]
header2
```

[96]:
```
['Rank', 'Country', 'GDP(US$MM)']
```

[97]:
```
rows2 = data_tables[1][0].findAll('tbody')[0].findAll('tr')[1:]
```

[98]:
```
def find_right_text(i, td):
    if i == 0:
        return td.getText().strip()
    elif i == 1:
        return td.getText().strip()
    else:
        index = td.text.find("♠")
        return td.text[index+1:].strip()
```

[99]:
```
data_rows2 = [[find_right_text(i, td) for i, td in enumerate(tr.findAll('td'))]␣
↪for tr in rows2]
```

[100]:
```
## assign data from pull into datafram & print
```

[101]:
```
df2 = pd.DataFrame(data_rows2, columns=header2)
```

[102]:
```
df2.head()
```

[102]:

	Rank	Country	GDP(US$MM)
0	1	United States	19,390,604
1		European Union[23]	17,277,698
2	2	China[n 4]	12,237,700
3	3	Japan	4,872,137
4	4	Germany	3,677,439

[103]:
```
## pull data header info
```

[104]:
```
header3 = [th.getText().strip() for th in data_tables[2][0].findAll('thead')[0].
↪findAll('th')]
header3
```

[104]:
```
['Rank', 'Country', 'GDP(US$MM)']
```

[105]:
```
rows3 = data_tables[2][0].findAll('tbody')[0].findAll('tr')[1:]
```

[106]:
```
data_rows3 = [[find_right_text(i, td) for i, td in enumerate(tr.findAll('td'))]␣
↪for tr in rows3]
```

[107]:
```
## assign data from pull into datafram & print
```

[108]:
```
df3 = pd.DataFrame(data_rows3, columns=header3)
df3.head()
```

[108]:

	Rank	Country	GDP(US$MM)
0	1	United States	18,624,475
1	2	China[n 4]	11,218,281
2	3	Japan	4,936,211
3	4	Germany	3,477,796
4	5	United Kingdom	2,647,898

[ ]:

Data Wrangling with Python-Activity 8

July 2, 2023

[28]:
```
# Carlos Cano
# DSC 540
# Activity 8
```

[29]:
```
## Import Libraries
```

[30]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[31]:
```
df = pd.read_csv("visit_data.csv")
```

[32]:
```
df.head()
```

[32]:

	id	first_name	last_name	email	gender	ip_address	visit
0	1	Sonny	Dahl	sdahl0@mysql.com	Male	135.36.96.183	1225.0
1	2	NaN	NaN	dhoovart1@hud.gov	NaN	237.165.194.143	919.0
2	3	Gar	Armal	garmal2@technorati.com	NaN	166.43.137.224	271.0
3	4	Chiarra	Nulty	cnulty3@newyorker.com	NaN	139.98.137.108	1002.0
4	5	NaN	NaN	sleaver4@elegantthemes.com	NaN	46.117.117.27	2434.0

[33]:
```
## Check for Duplicates
```

[34]:
```
print("First name is duplictaed - {}".format(any(df.first_name.duplicated())))
print("Last name is duplictaed - {}".format(any(df.last_name.duplicated())))
print("Email is duplictaed - {}".format(any(df.email.duplicated())))
```

```
First name is duplictaed - True
Last name is duplictaed - True
Email is duplictaed - False
```

[35]:
```
## Check for Missinh Data
```

[36]:
```
print("The column Email contains NaN - %r " % df.email.isnull().values.any())
print("The column IP Address contains NaN - %s " % df.ip_address.isnull().values.
 ↪any())
print("The column Visit contains NaN - %s " % df.visit.isnull().values.any())
```

```
The column Email contains NaN - False
The column IP Address contains NaN - False
The column Visit contains NaN - True
```

[37]:
```
## Remove Outliers
```

[38]:
```
size_prev = df.shape
df = df[np.isfinite(df['visit'])] #This is an inplace operation. After this␣
 ↪operation the original DataFrame is lost.
size_after = df.shape
```

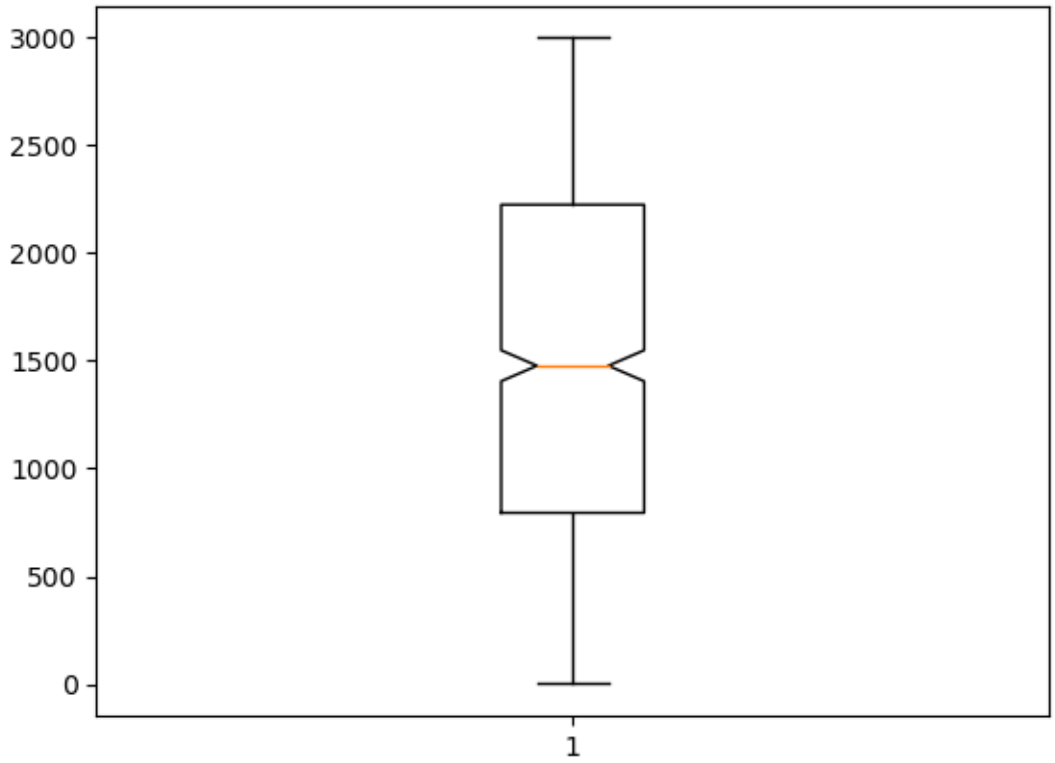
[39]:
```
## Size Difference
```

[40]:
```
print("The size of previous data was - {prev[0]} rows and the size of the new␣
 ↪one is - {after[0]} rows".
      format(prev=size_prev, after=size_after))
```

```
The size of previous data was - 1000 rows and the size of the new one is - 974
rows
```

[41]:
```
plt.boxplot(df.visit, notch=True)
```

[41]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fc56823d7c0>,
  <matplotlib.lines.Line2D at 0x7fc56823da90>],
 'caps': [<matplotlib.lines.Line2D at 0x7fc56823dd60>,
  <matplotlib.lines.Line2D at 0x7fc56824c070>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fc56823d4f0>],
 'medians': [<matplotlib.lines.Line2D at 0x7fc56824c340>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fc56824c610>],
 'means': []}
```



[42]:
```
df1 = df[(df['visit'] <= 2900) & (df['visit'] >= 100)]
```

[43]:
```
print("Remaining data is - {}".format(*df1.shape))
```

```
Remaining data is - 923
```

[44]:
```
## print final dataset condensed and refined
```

[45]:
```
df1.head(50)
```

[45]:

	id	first_name	last_name	email	gender
0	1	Sonny	Dahl	sdahl0@mysql.com	Male
1	2	NaN	NaN	dhoovart1@hud.gov	NaN
2	3	Gar	Armal	garmal2@technorati.com	NaN
3	4	Chiarra	Nulty	cnulty3@newyorker.com	NaN
4	5	NaN	NaN	sleaver4@elegantthemes.com	NaN
5	6	Raymund	Ingerfield	ringerfield5@microsoft.com	NaN
6	7	Wilhelmina	Dagnan	wdagnan6@nytimes.com	Female
7	8	NaN	NaN	mdewilde7@creativecommons.org	Female
8	9	Gunter	Lisamore	glisamore8@disqus.com	NaN
9	10	Luelle	Scinelli	lscinelli9@issuu.com	Female
10	11	Kayne	Charlick	kcharlicka@privacy.gov.au	NaN

11	12	NaN	NaN	jmccotterb@ning.com	NaN
12	13	Katya	Rewcassell	krewcassellc@dyndns.org	Female
13	14	NaN	NaN	gkippied@infoseek.co.jp	NaN
14	15	Felice	Chaffin	fchaffine@shutterfly.com	NaN
15	16	NaN	NaN	wclowleyf@usda.gov	Male
16	17	NaN	NaN	ldickerlineg@youtu.be	Female
17	18	Forrester	Randleson	frandlesonh@cnet.com	Male
18	19	Rabbi	Lawton	rlawtoni@over-blog.com	Male
19	20	Geoff	Scholtz	gscholtzj@google.com.au	NaN
21	22	NaN	NaN	jtwistl@cbc.ca	Male
22	23	NaN	NaN	nheepsm@slideshare.net	NaN
24	25	Evelin	Ludgrove	eludgroveo@slate.com	NaN
26	27	Linda	Rampton	lramptonq@businessweek.com	Female
27	28	Damian	Shawl	dshawlr@amazon.co.uk	Male
28	29	Mitchel	Beynkn	mbeynkns@usgs.gov	NaN
30	31	Marwin	Guilliatt	mguilliattu@cdc.gov	Male
32	33	NaN	NaN	cbortolazziw@redcross.org	Male
33	34	NaN	NaN	dsynckex@example.com	NaN
34	35	Casi	Harses	charsesy@ft.com	Female
35	36	Jourdan	Barock	jbarockz@ebay.co.uk	NaN
36	37	Bob	Cammock	bcammock10@alexa.com	NaN
37	38	Myrtle	Huffadine	mhuffadine11@reference.com	NaN
38	39	Harrie	Ughelli	hughelli12@bandcamp.com	Female
39	40	Quinn	Hesse	qhesse13@abc.net.au	NaN
41	42	NaN	NaN	welizabeth15@twitpic.com	Female
42	43	NaN	NaN	hrusbridge16@auda.org.au	NaN
43	44	Cullin	Oades	coades17@bandcamp.com	Male
44	45	Filip	O'Lennane	folennane18@behance.net	NaN
45	46	Marlowe	Gilardi	mgilardi19@uol.com.br	Male
46	47	NaN	NaN	omilillo1a@moonfruit.com	Male
47	48	Hyman	Valentim	hvalentim1b@sohu.com	NaN
48	49	NaN	NaN	clarive1c@pbs.org	NaN
51	52	Giacomo	Laden	gladen1f@shutterfly.com	NaN
52	53	NaN	NaN	sswateridge1g@adobe.com	NaN
53	54	NaN	NaN	cskelhorn1h@ucsd.edu	NaN
54	55	Theobald	Seekings	tseekings1i@surveymonkey.com	NaN
55	56	Gayle	Petchell	gpetchell1j@desdev.cn	Male
56	57	Daniel	Brunini	dbrunini1k@163.com	Male
57	58	Romy	Alastair	ralastair1l@skyrock.com	Female

	ip_address	visit
0	135.36.96.183	1225.0
1	237.165.194.143	919.0
2	166.43.137.224	271.0
3	139.98.137.108	1002.0
4	46.117.117.27	2434.0
5	90.100.118.215	451.0

6	88.133.77.243	1540.0
7	229.215.244.227	537.0
8	134.185.44.82	743.0
9	160.130.58.61	1507.0
10	32.242.11.185	913.0
11	180.112.224.129	127.0
12	68.203.78.150	661.0
13	248.75.123.182	1867.0
14	25.46.111.146	1917.0
15	45.37.121.91	1331.0
16	177.43.87.9	933.0
17	133.200.143.251	303.0
18	227.242.70.247	433.0
19	81.31.175.252	810.0
21	125.95.76.155	1626.0
22	194.65.165.56	2482.0
24	177.66.83.108	1003.0
26	134.196.180.179	708.0
27	198.159.170.249	2556.0
28	213.232.25.221	2507.0
30	29.29.195.151	1096.0
32	62.211.45.21	1071.0
33	12.7.170.17	327.0
34	231.212.221.219	2338.0
35	48.154.135.166	1356.0
36	68.216.220.101	358.0
37	215.104.187.189	455.0
38	127.189.174.165	1417.0
39	103.249.143.153	1984.0
41	33.32.239.61	2231.0
42	236.238.215.118	2195.0
43	124.67.131.185	2175.0
44	185.206.69.199	2196.0
45	122.174.183.193	258.0
46	160.225.253.133	783.0
47	47.38.144.39	623.0
48	136.56.223.95	977.0
51	78.132.222.73	1472.0
52	18.139.235.174	1517.0
53	242.73.159.107	2392.0
54	228.22.41.182	2875.0
55	54.176.87.111	2699.0
56	51.69.198.8	2533.0
57	143.127.21.206	2550.0

Week 3 Exercise 1

July 2, 2023

[29]:
```
# Carlos Cano
# DSC 540
# Week 3 Exercise 1
```

[30]:
```
## Import library
```

[31]:
```
import pandas as pd
```

[32]:
```
## Create function that calculates the addition and subtraction of two series
```

[33]:
```
def Add_Sub_Func():

## Input of values

    Series_1 = [7.3, -2.5, 3.4, 1.5]

## Input for Index for Series_1

    Index_1 = ['a','c','d','e']

## Input of values Series_2

    Series_2 =[-2.1, 3.6, -1.5, 4, 3.1]

## Input for Index for Series_2

    Index_2 = ['a', 'c', 'e', 'f', 'g']

## Create a series using series_1 and index_1

    S_1 = pd.Series(Series_1,Index_1)

## Create a series using series_2 and index_2

    S_2 = pd.Series(Series_2,Index_2)

## Add the series s1 and s2 print the result
```

```
    print("Addition of Series is:\n",S_1.add(S_2))

    print()

## Subtratct the s1-s2 and print the result

    print("Subtraction of Series is:\n",S_1.subtract(S_2))
```

[34]:
```
## main function
if __name__ == "__main__":
## Call the Add_sub_series function
    Add_Sub_Func()
```

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

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

[ ]:

Week 3 Exercise 2

July 2, 2023

[65]:
```
# Carlos Cano
# DSC 540
# Week 3 Exercise 2
```

[66]:
```
## Import Libraries
```

[67]:
```
import sqlite3 as sq
```

[68]:
```
## Create Databased
```

[69]:
```
conn = sq.connect('customers.db')
```

[70]:
```
## Create Variables
```

[71]:
```
conn.execute('''CREATE TABLE customers
             (name TEXT, address TEXT, city TEXT, state TEXT, zip TEXT,␣
 ↪phone_number TEXT)''')
```

[71]:
```
<sqlite3.Cursor at 0x7fce691d8c00>
```

[72]:
```
## Data to be imported
```

[73]:
```
data = [('Billy Jole', '704 Hauser St', 'New York', 'NY', '12345',␣
 ↪'888-888-0001'),
       ('Anna Nicole', '221B Baker', 'London', 'UK', '32145', '888-888-0002'),
       ('Tim Barker', '129 W. 81st St', 'New York', 'NY', '45654',␣
 ↪'888-888-0003'),
       ('Don King', '124 Conch St.', 'Bikini Bottom', 'PO', '67544',␣
 ↪'888-888-0004'),
       ('Bill Gates', '1600 Pennsylvania Ave', 'Washington DC', 'DC', '56789',␣
 ↪'888-555-0005'),
       ('Jack Nickelson', '485 Maple Dr', 'Mayberry', 'NC', '54321',␣
 ↪'888-888-0006'),
       ('Jason Bourne', '698 Candlewood Lane', 'Cabot Cove', 'FL', '00044',␣
 ↪'888-888-0007'),
       ('Bruce Wayne', '607 S. Maple St', 'Hollywood', 'CA', '90028',␣
 ↪'888-888-0008'),
```

```
        ('Clark Kent', '79 Wistful Vista', 'Tampa', 'FL', '90210',␣
↪'888-888-0009'),
        ('Tony Stark', '200 Chesternut Dr', 'Oakbridge', 'NE', '33789',␣
↪'888-888-0010')]
conn.executemany('INSERT INTO customers VALUES (?,?,?,?,?,?)', data)
```

[73]:
```
<sqlite3.Cursor at 0x7fce691d8ea0>
```

[74]:
```
cursor = conn.execute("SELECT * from customers")
rows = cursor.fetchall()
for row in rows:
    print(row)
```

```
('Billy Jole', '704 Hauser St', 'New York', 'NY', '12345', '888-888-0001')
('Anna Nicole', '221B Baker', 'London', 'UK', '32145', '888-888-0002')
('Tim Barker', '129 W. 81st St', 'New York', 'NY', '45654', '888-888-0003')
('Don King', '124 Conch St.', 'Bikini Bottom', 'PO', '67544', '888-888-0004')
('Bill Gates', '1600 Pennsylvania Ave', 'Washington DC', 'DC', '56789',
'888-555-0005')
('Jack Nickelson', '485 Maple Dr', 'Mayberry', 'NC', '54321', '888-888-0006')
('Jason Bourne', '698 Candlewood Lane', 'Cabot Cove', 'FL', '00044',
'888-888-0007')
('Bruce Wayne', '607 S. Maple St', 'Hollywood', 'CA', '90028', '888-888-0008')
('Clark Kent', '79 Wistful Vista', 'Tampa', 'FL', '90210', '888-888-0009')
('Tony Stark', '200 Chesternut Dr', 'Oakbridge', 'NE', '33789', '888-888-0010')
```

[ ]: