In [149…
```
#Shaquiel Pashtunyar
#DSC550
#Week 3 and 4 homework
```

In [150…
```
#basic imports for packets
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [151…
```
#getting the boston data from the packt github page
url = 'https://raw.githubusercontent.com/tirthajyoti/Packt-Data_Wrangling/master/Lesso
boston = pd.read_csv(url)
```

In [152…
```
#first 10 rows of the data set
boston.head(10)
```

Out[152]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PR
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	

In [153…
```
#Shape gives us data set shape, so 506 rows and 13 columns
boston.shape
```

Out[153]:
```
(506, 14)
```

In [154…
```
boston2= boston.drop(['NOX', 'B', 'LSTAT', 'CHAS'], axis=1)
```

In [155…
```
#new dataset with less columns
boston2
```

Out[155]:

	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
0	0.00632	18.0	2.31	6.575	65.2	4.0900	1	296	15.3	24.0
1	0.02731	0.0	7.07	6.421	78.9	4.9671	2	242	17.8	21.6
2	0.02729	0.0	7.07	7.185	61.1	4.9671	2	242	17.8	34.7
3	0.03237	0.0	2.18	6.998	45.8	6.0622	3	222	18.7	33.4
4	0.06905	0.0	2.18	7.147	54.2	6.0622	3	222	18.7	36.2
...	...	...	...	...	...	...	...	...	...	...
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

506 rows × 10 columns

In [156…

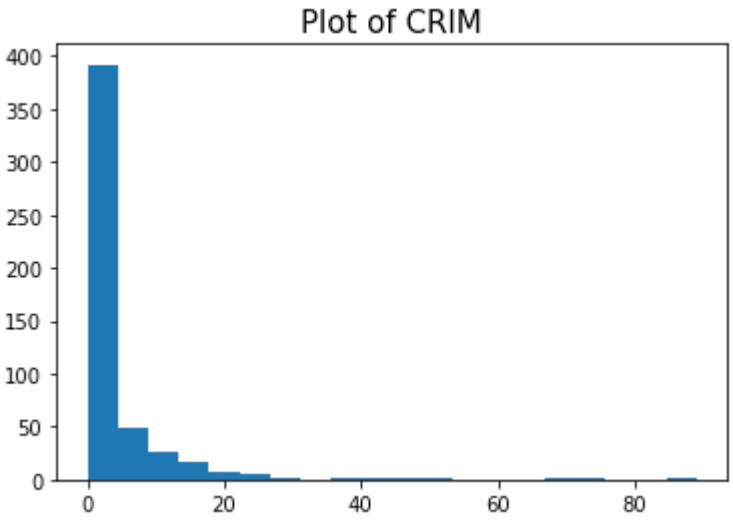
```
#final 7 rows of dataset
boston2.tail(7)
```

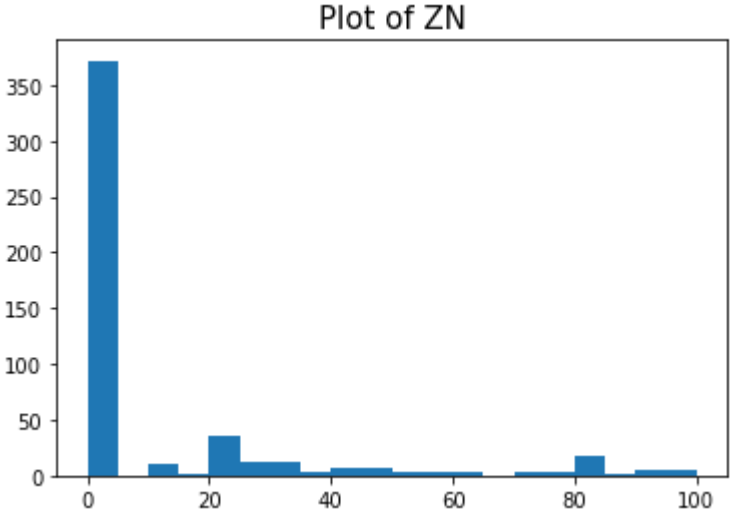
Out[156]:

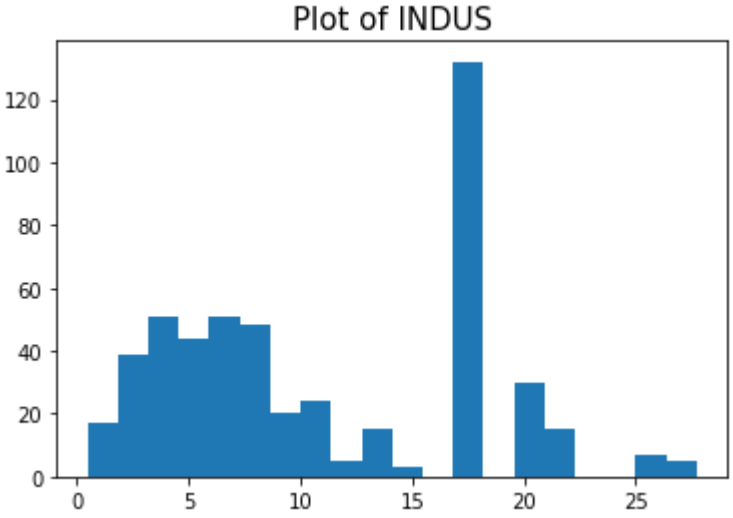
	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
499	0.17783	0.0	9.69	5.569	73.5	2.3999	6	391	19.2	17.5
500	0.22438	0.0	9.69	6.027	79.7	2.4982	6	391	19.2	16.8
501	0.06263	0.0	11.93	6.593	69.1	2.4786	1	273	21.0	22.4
502	0.04527	0.0	11.93	6.120	76.7	2.2875	1	273	21.0	20.6
503	0.06076	0.0	11.93	6.976	91.0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

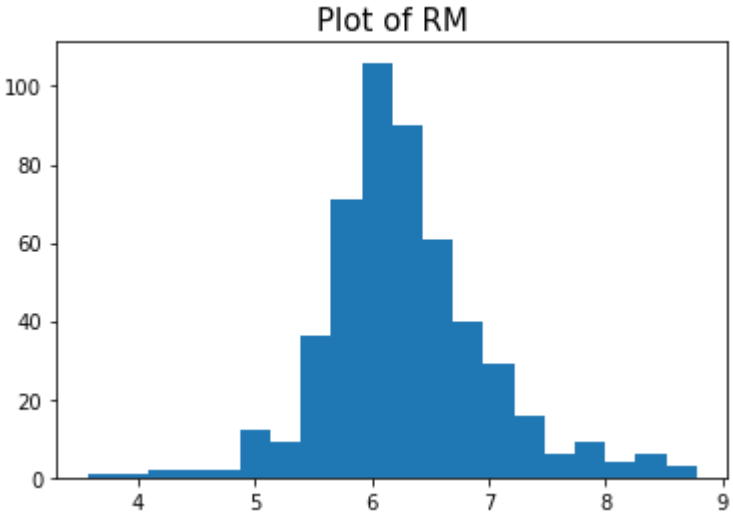
In [157…

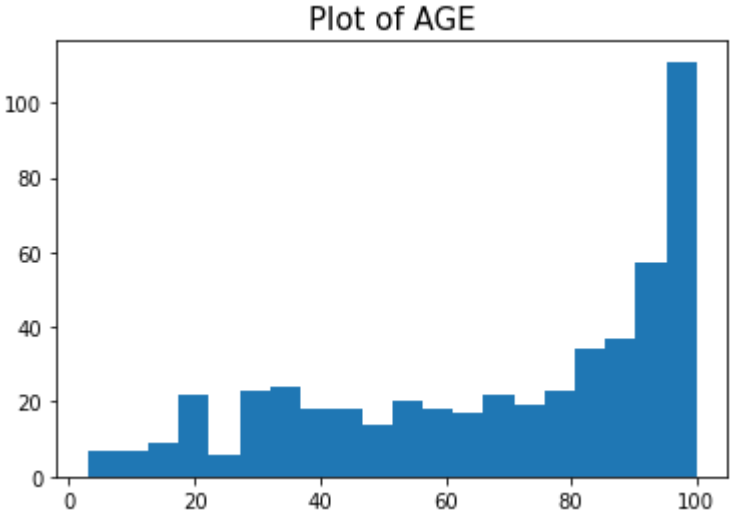
```
#Wrote a loop that creates all of the plots for me
for c in boston2.columns:
    plt.title("Plot of "+c,fontsize=15)
    plt.hist(boston2[c],bins=20)
    plt.show()
```

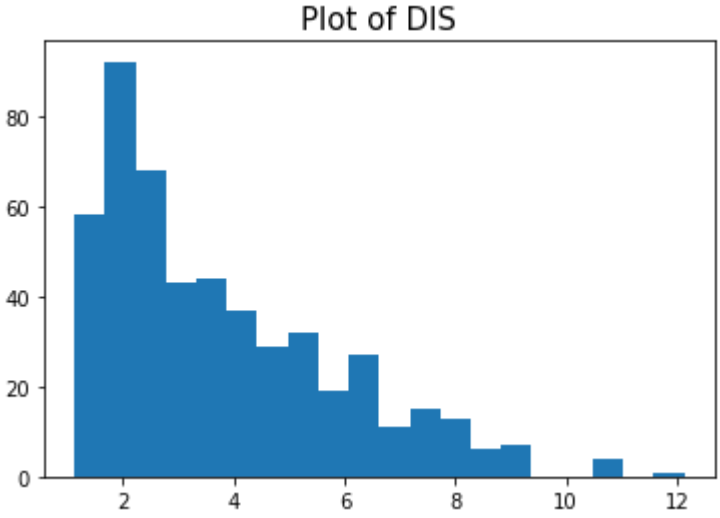

Plot of CRIM
400
350
300
250
200
150
100
50
0
0
20
40
60
80

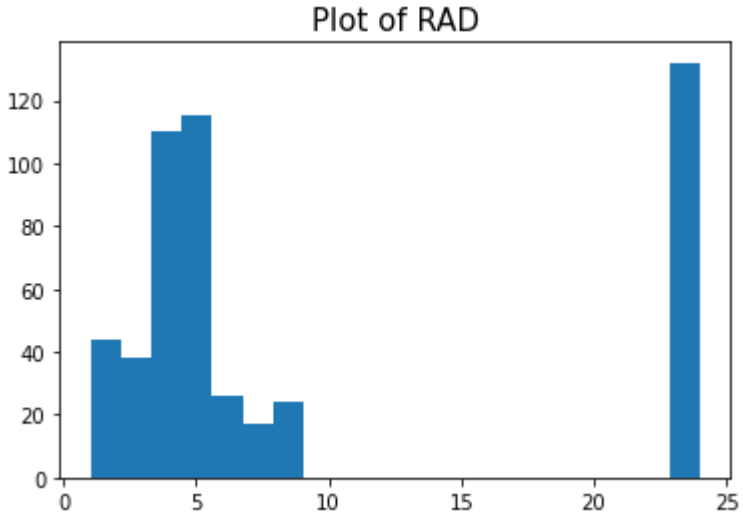

Plot of ZN
350
300
250
200
150
100
50
0
0
20
40
60
80
100

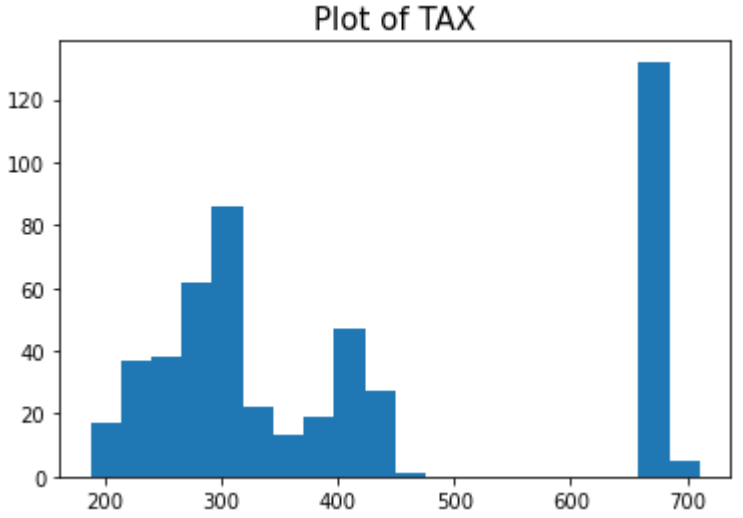

Plot of INDUS
120
100
80
60
40
20
0
0
5
10
15
20
25

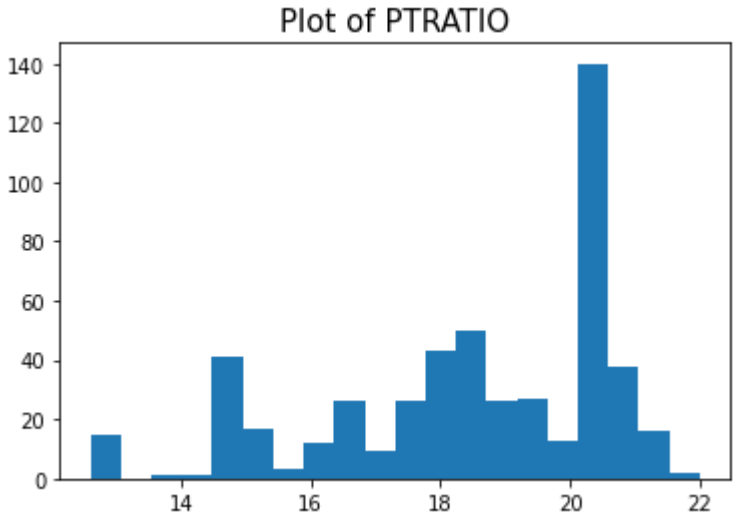

Plot of RM
100
80
60
40
20
0
4
5
6
7
8
9

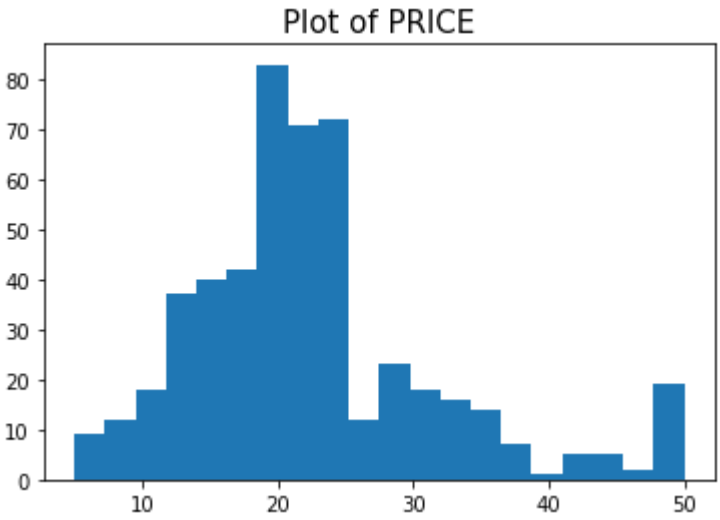

Plot of AGE
100
80
60
40
20
0
0
20
40
60
80
100


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



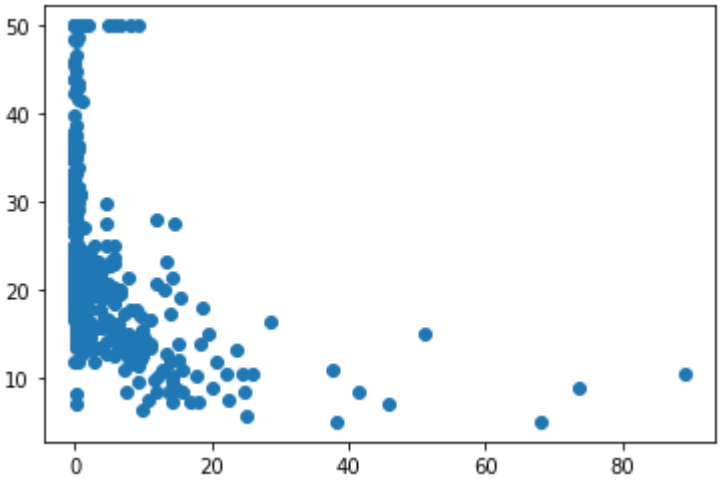
In [158…
```
boston2.head()
```

Out[158]:

	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
0	0.00632	18.0	2.31	6.575	65.2	4.0900	1	296	15.3	24.0
1	0.02731	0.0	7.07	6.421	78.9	4.9671	2	242	17.8	21.6
2	0.02729	0.0	7.07	7.185	61.1	4.9671	2	242	17.8	34.7
3	0.03237	0.0	2.18	6.998	45.8	6.0622	3	222	18.7	33.4
4	0.06905	0.0	2.18	7.147	54.2	6.0622	3	222	18.7	36.2

In [159…
```
#Scatter plot creation
plt.scatter(boston2['CRIM'],boston2['PRICE'])
plt.show
```

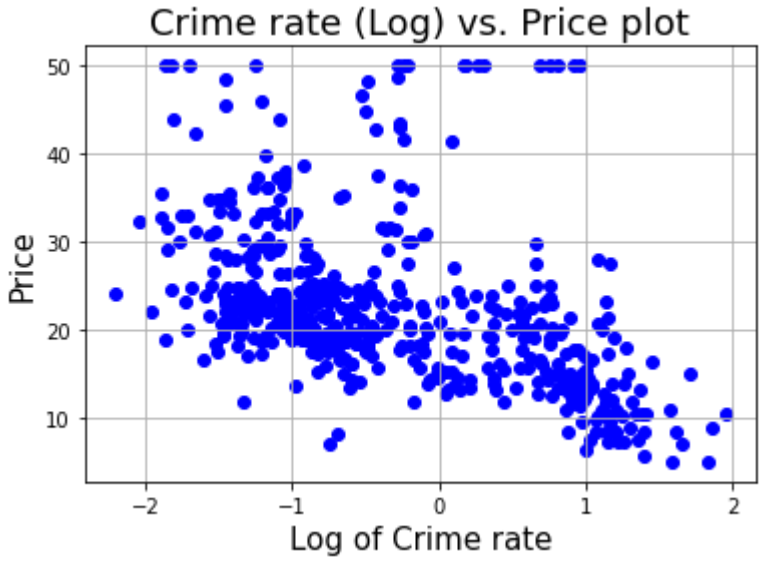
Out[159]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [160…
```
display(boston2.dtypes)
```

```
CRIM       float64
ZN         float64
INDUS      float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX          int64
PTRATIO    float64
PRICE      float64
dtype: object
```

In [161…
```
#Creating a logrimtic scale to the scatter plot for crim rate. Price going up doesn't
plt.scatter(np.log10(boston2['CRIM']),boston2['PRICE'],c='blue')
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 [162…
```
boston2.describe
```

Out[162]:
```
<bound method NDFrame.describe of         CRIM    ZN  INDUS     RM   AGE     DIS  RAD
TAX  PTRATIO  PRICE
0    0.00632  18.0   2.31  6.575  65.2  4.0900    1  296     15.3   24.0
1    0.02731   0.0   7.07  6.421  78.9  4.9671    2  242     17.8   21.6
2    0.02729   0.0   7.07  7.185  61.1  4.9671    2  242     17.8   34.7
3    0.03237   0.0   2.18  6.998  45.8  6.0622    3  222     18.7   33.4
4    0.06905   0.0   2.18  7.147  54.2  6.0622    3  222     18.7   36.2
..       ...   ...    ...    ...   ...     ...  ...  ...      ...    ...
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

[506 rows x 10 columns]>
```

In [163…
```
#mean number of rooms
boston2['RM'].mean()
```

Out[163]:
```
6.284634387351787
```

In [164…
```
# median age of house
boston2['AGE'].median()
```

Out[164]:
```
77.5
```

In [165…
```
# distance of 5 boston employment centers
boston2['DIS'].mean()
```

Out[165]:
```
3.795042687747034
```

In [166…
```
#create a field with true or false boolean on low price
low_price=boston2['PRICE']<20
# This creates a Boolean array of True, False
print(low_price)
# True = 1, False = 0, so now if you take an average of this Numpy array, you will kno
# That many houses are priced below 20,000. So that is the answer.
# 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)
```

```
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

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

In [167…
```
#downloading data file from aadult csv
income = pd.read_csv('adult.csv')
```

In [168…
```
income.head()
```

Out[168]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gende
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Ma
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Ma
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Ma
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Ma
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Fema

In [169…

```
#creating script to read txt line by line
#text file open
names = []
with open('adultnames.txt','r') as f:
    for line in f:
        f.readline()
        var=line.split(":")[0]
        names.append(var)
```

In [170…

```
#column names
names
```

Out[170]:

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

In [171…

```
#adding income by using append
names.append('Income')
```

In [173…

```
#using name column that we created above
url = 'https://raw.githubusercontent.com/TrainingByPackt/Data-Wrangling-with-Python/ma
income = pd.read_csv(url,names=names)
```

In [174…

```
income.head()
```

Out[174]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0

In [175…

```
#info on the dataset
income.describe()
```

Out[175]:

	age	fnlwgt	education-num	sex	capital-gain	capital-loss	Income
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000	0.0
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456	NaN
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429	NaN
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000	NaN
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000	NaN
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000	NaN
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000	NaN
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000	NaN

In [177…

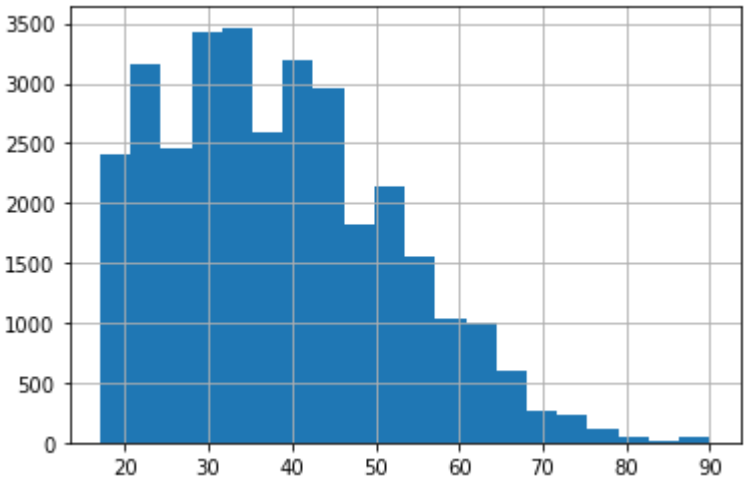
```
#number of missing datapoints looks like none except for income
income.isnull().sum()
```

Out[177]:
```
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          32561
dtype: int64
```

In [178…
```
#Subset data frame with only select columns
income_subset = income[['age','education','occupation','race']]
```

In [179…
```
#age histogram with a bin of 20
income_subset['age'].hist(bins=20)
```

Out[179]:
```
<AxesSubplot:>
```



In [180…
```
#whitespace striper
def strip_whitespace(s):
    return s.strip()
```

In [183…
```
#applying the whitespace stripper to the 3 columns in my subset
# Education column
income_subset['education_stripped']=income['education'].apply(strip_whitespace)
income_subset['education']=income_subset['education_stripped']
income_subset.drop(labels=['education_stripped'],axis=1,inplace=True)

# Occupation column
income_subset['occupation_stripped']=income['occupation'].apply(strip_whitespace)
income_subset['occupation']=income_subset['occupation_stripped']
income_subset.drop(labels=['occupation_stripped'],axis=1,inplace=True)

# Race column
income_subset['race_stripped']=income['race'].apply(strip_whitespace)
```

```
income_subset['race']=income_subset['race_stripped']
income_subset.drop(labels=['race_stripped'],axis=1,inplace=True)
```

```
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:3: SettingWithCo
pyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset['education_stripped']=income['education'].apply(strip_whitespace)
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:4: SettingWithCo
pyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset['education']=income_subset['education_stripped']
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:5: SettingWithCo
pyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset.drop(labels=['education_stripped'],axis=1,inplace=True)
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:8: SettingWithCo
pyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset['occupation_stripped']=income['occupation'].apply(strip_whitespace)
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:9: SettingWithCo
pyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset['occupation']=income_subset['occupation_stripped']
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:10: SettingWithC
opyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset.drop(labels=['occupation_stripped'],axis=1,inplace=True)
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:13: SettingWithC
opyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset['race_stripped']=income['race'].apply(strip_whitespace)
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:14: SettingWithC
opyWarning:
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/us
```

```
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset['race']=income_subset['race_stripped']
C:\Users\spashtunyar\AppData\Local\Temp\ipykernel_34080\240269240.py:15: SettingWithC
opyWarning:
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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  income_subset.drop(labels=['race_stripped'],axis=1,inplace=True)
```

In [186…
```
# people between 30 and 50
income_filtered=income_subset[(income_subset['age']>=30) & (income_subset['age']<=50)]
```

In [187…
```
income_filtered.head()
```

Out[187]:

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

In [188…
```
#number of people who are between 30 and 50
income_filtered.shape[0]
```

Out[188]:
```
16390
```

In [193…
```
#the mean column give us the average age at each educational level
income_subset.groupby('education').describe()['age']
```

Out[193]:

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

In [200…
```
#creating data series
Data1 = 7.3, -2.5, 3.4, 1.5
Data2 = -2.1, 3.6, -1.5, 4, 3.1
```

In [202…
```
#pandas dataset
series1 = pd.Series(Data1);

series2 = pd.Series(Data2);
```

In [205…
```
series1
```

Out[205]:
```
0    7.3
1   -2.5
2    3.4
3    1.5
dtype: float64
```

In [212…
```
#add them together
Seriesadd = np.add(series1,series2)
Seriesadd
```

Out[212]:
```
0    5.2
1    1.1
2    1.9
3    5.5
4    NaN
dtype: float64
```

In [211…

```
#subtract the two
Seriessub = np.subtract(series1,series2)
Seriessub
```

Out[211]:

```
0    9.4
1   -6.1
2    4.9
3   -2.5
4    NaN
dtype: float64
```

In [ ]: