In [7]:
```
#Name: Paul Galvez
#Date: 4/7/23
#Week 3 and 4
#DSC 540
```

In [8]:
```
#1. Data Wrangling with Python: Activity 5, page 116
#loading the necessay libraries - in this case I will be using the following libraries:
# numpy, pandas, and matplotlib

import numpy as np
```

In [9]:
```
import pandas as pd
```

In [10]:
```
import matplotlib.pyplot as plt
```

In [12]:
```
#reading the Boston Housing CSV file into the workbook

df=pd.read_csv("Boston_housing.csv")
df
```

Out[12]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

In [14]:
```
#the first ten records are shown for the dataset df.head(10)

df.head(10)
```

Out[14]:

	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	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9

In [17]:
```
df.tail()
```

Out[17]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

In [16]:
```
#the total number of records are 505 rows and 13 columns

df.shape
```

Out[16]:
```
(506, 14)
```

In [19]:
```
#Create a smaller DataFrame with columns which do not include 'CHAS', 'NOX', 'B', and 'LSTAT'

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

Out[19]:

	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

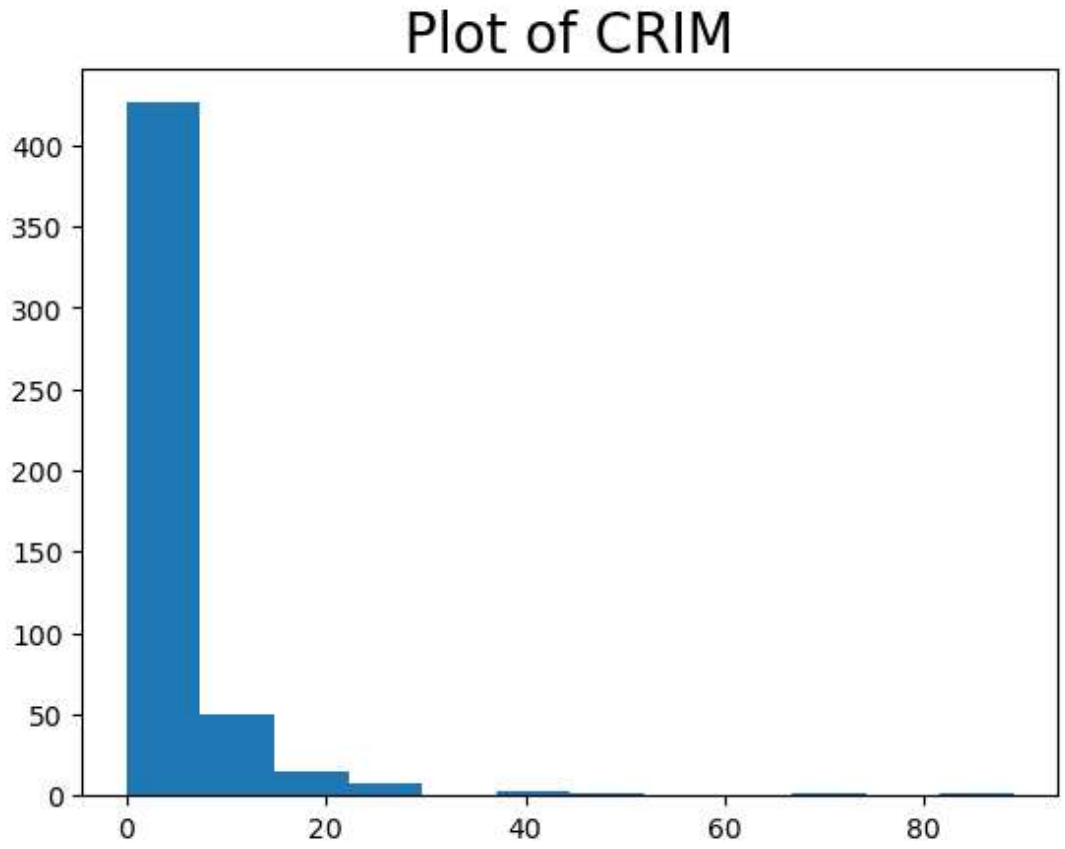
506 rows × 10 columns

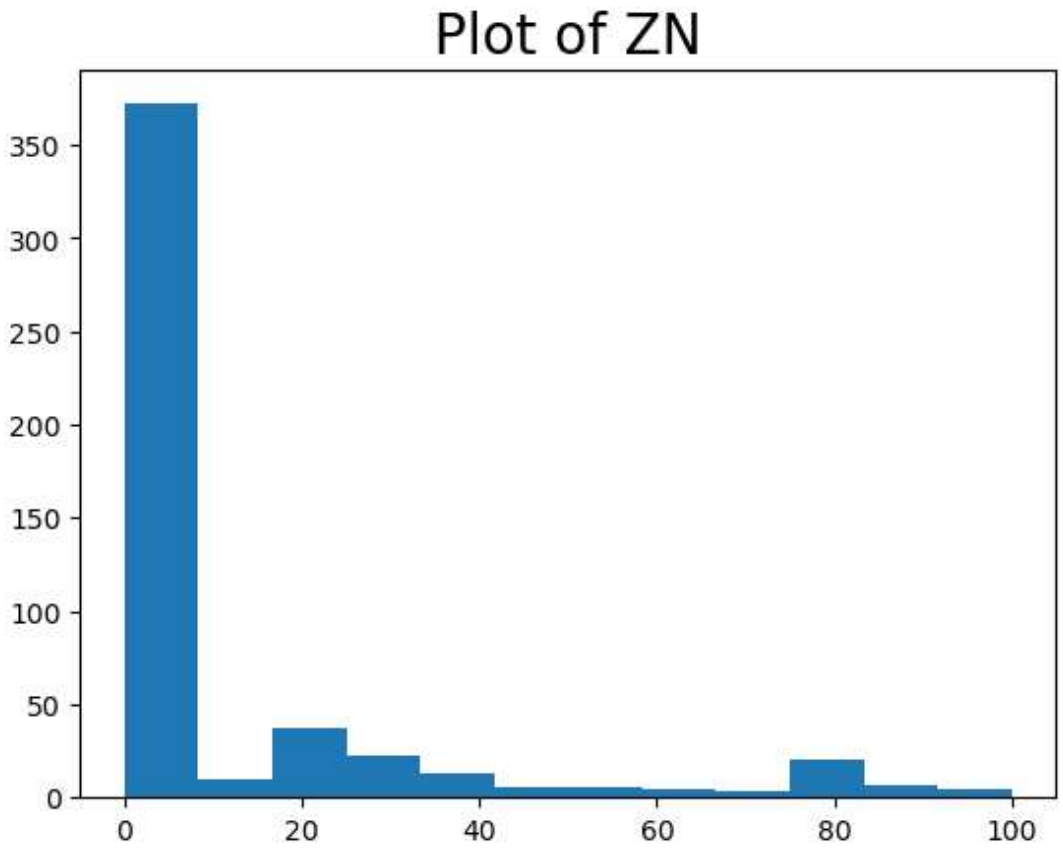
In [20]:
```
#Check the last 7 records of the new DataFrame you just created

df.tail(7)
```

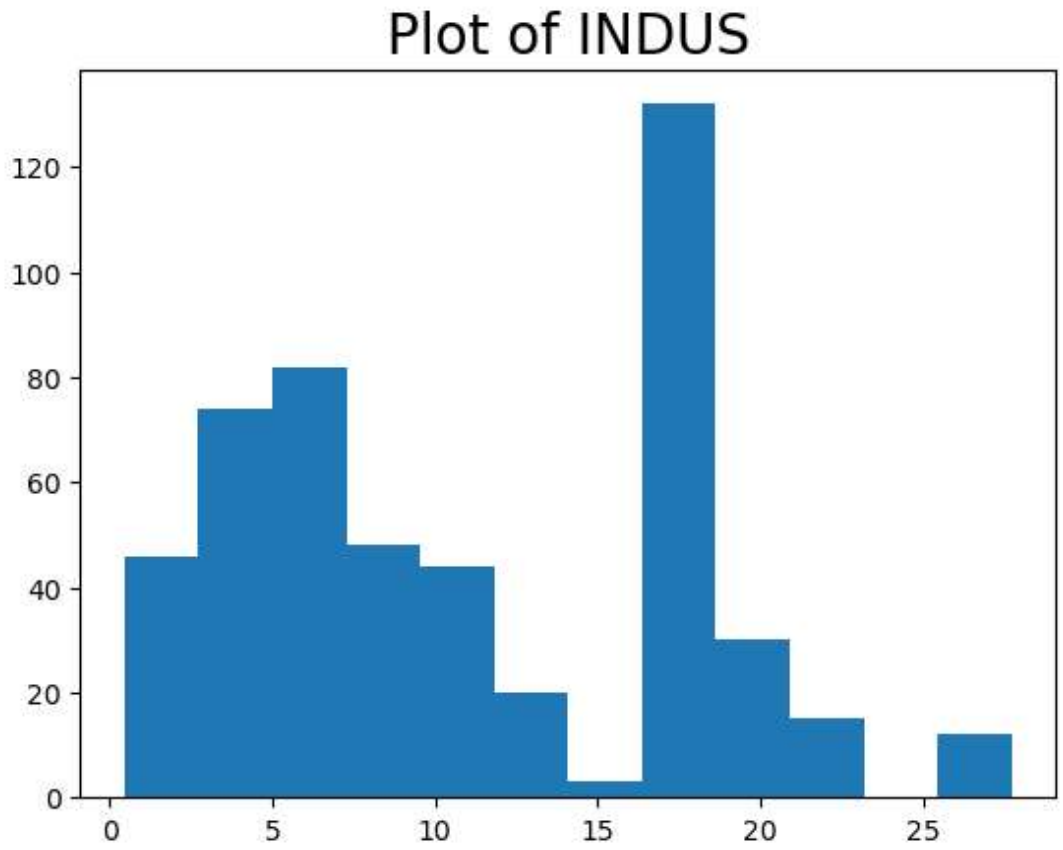
Out[20]:

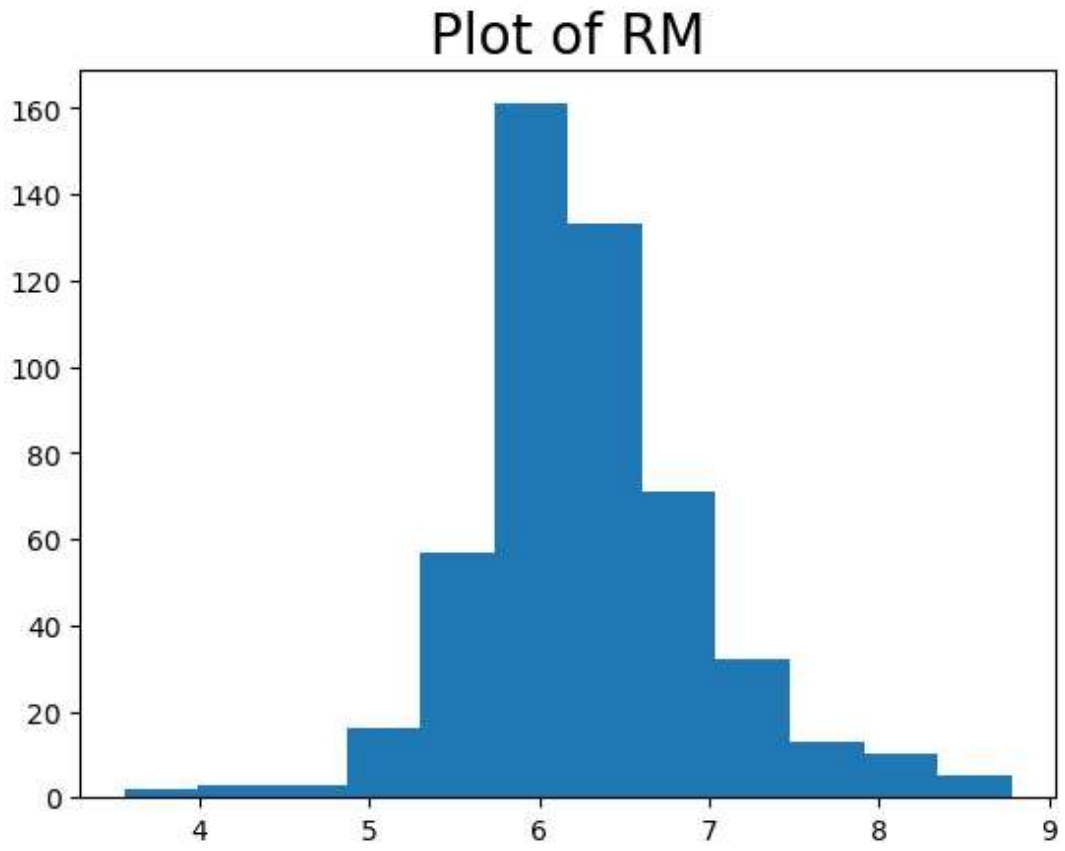
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
499	0.17783	0.0	9.69	0	0.585	5.569	73.5	2.3999	6	391	19.2	395.77	15.10	17.5
500	0.22438	0.0	9.69	0	0.585	6.027	79.7	2.4982	6	391	19.2	396.90	14.33	16.8
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

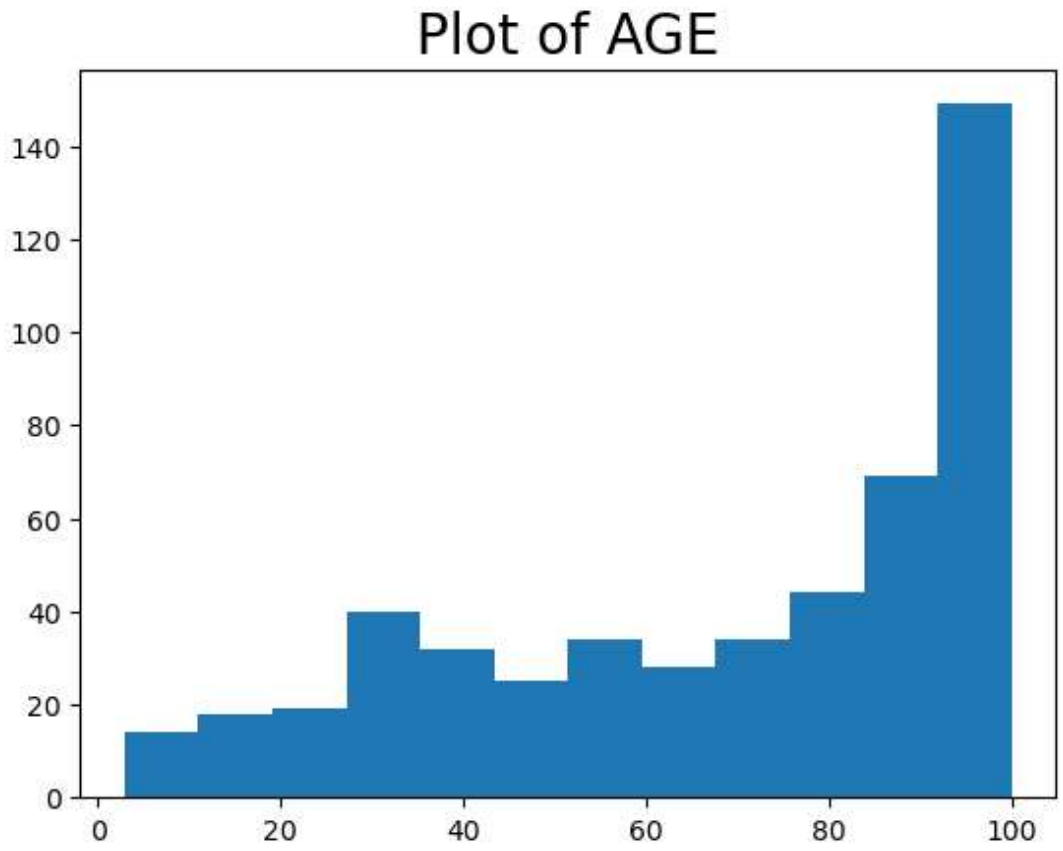
In [32]:
```
#Can you plot histograms of all the variables (columns) in the new DataFrame?
#using a for loop ever the data and the remaining variables in the new dataframe and to
#avoid plotting each variable individually.

for c in smll_df.columns:
    plt.title("Plot of "+c,fontsize = 20)
    plt.hist(smll_df[c],bins = 12)
    plt.show()
```

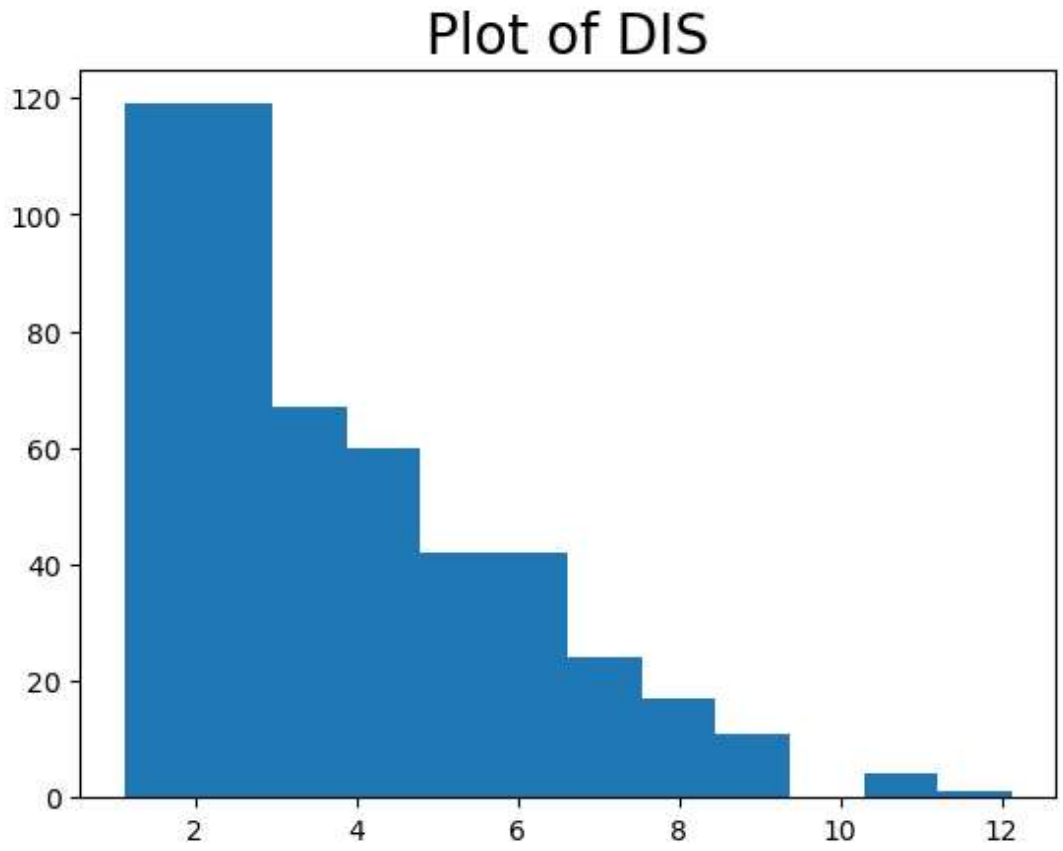


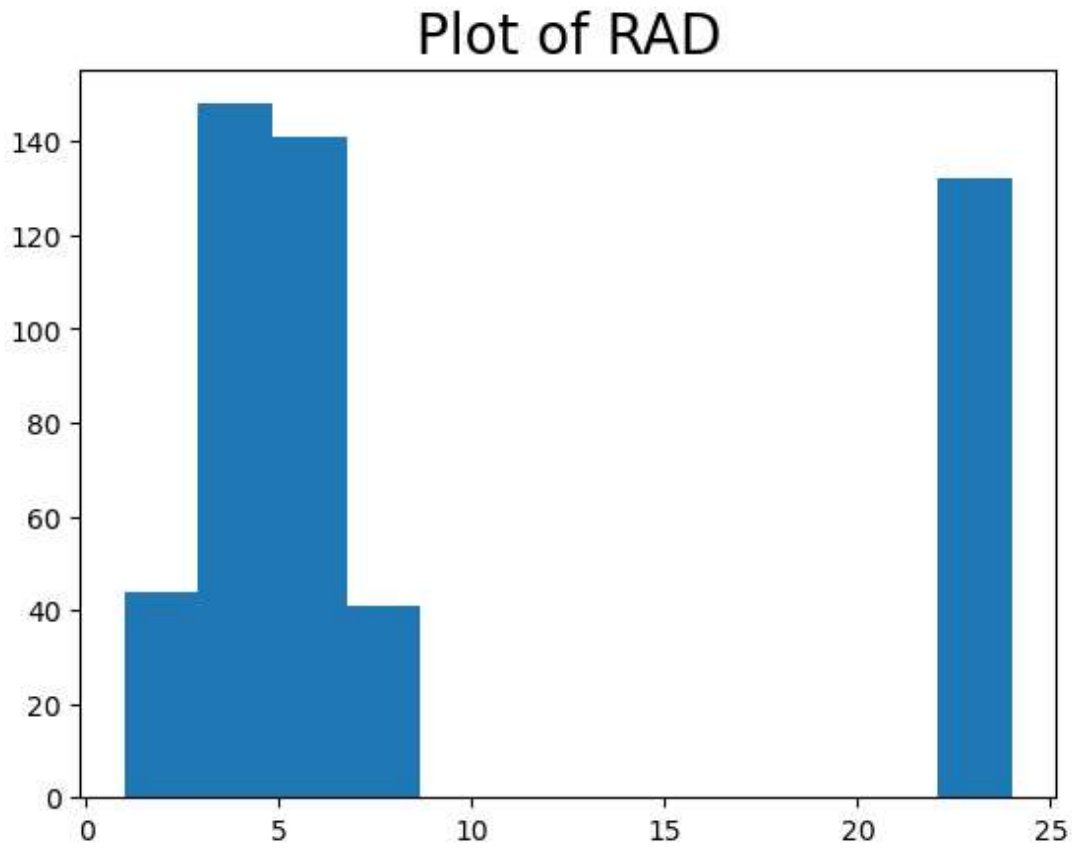

Plot of ZN
350
300
250
200
150
100
50
0
0
20
40
60
80
100

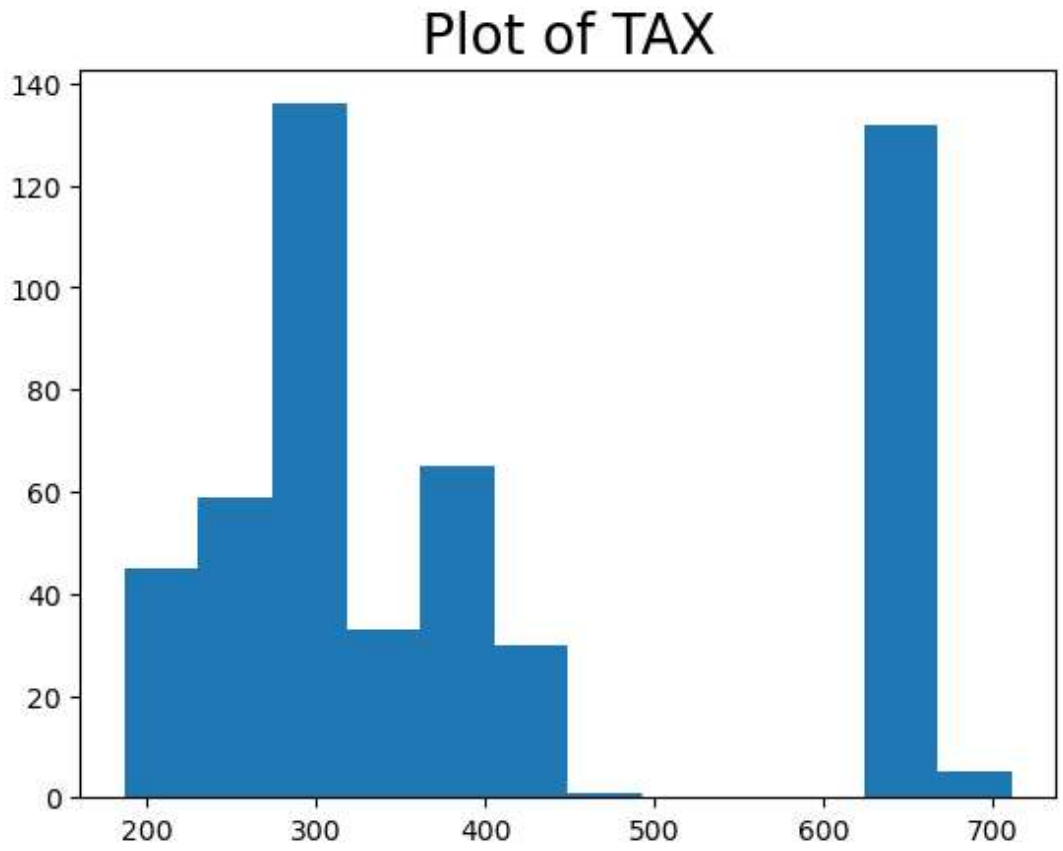

Plot of INDUS
120
100
80
60
40
20
0
0
5
10
15
20
25

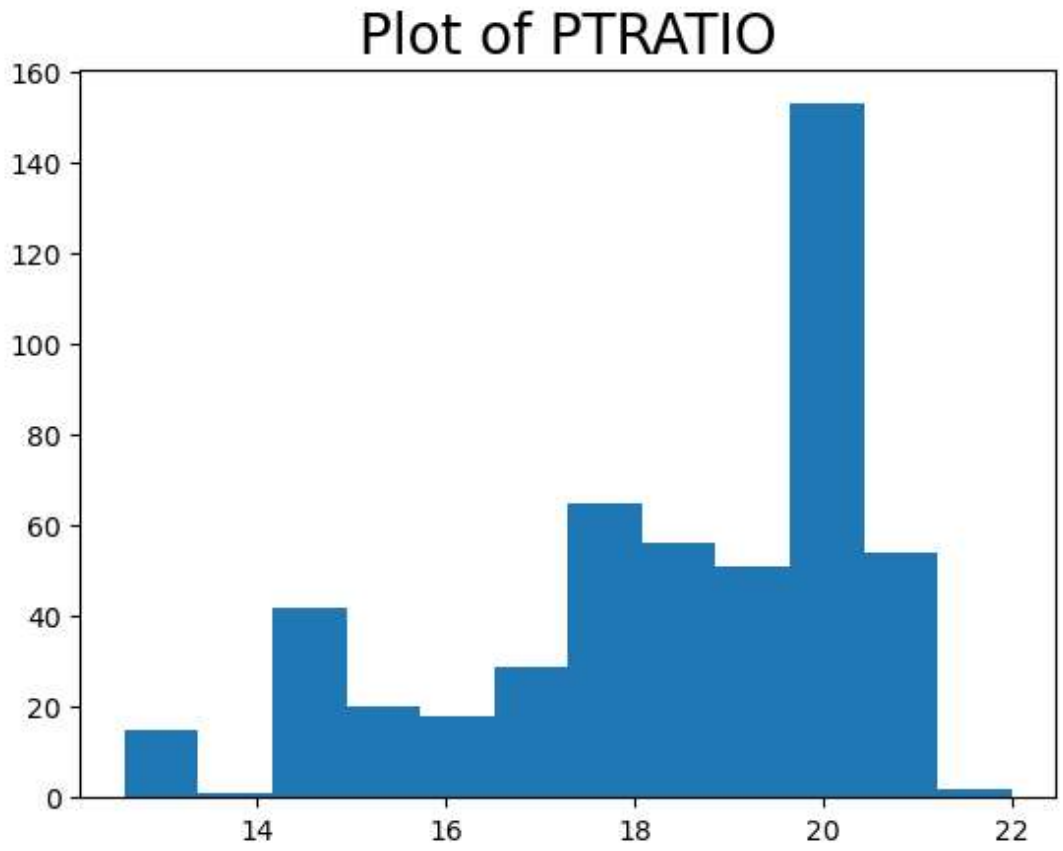

Plot of RM
160
140
120
100
80
60
40
20
0
4
5
6
7
8
9

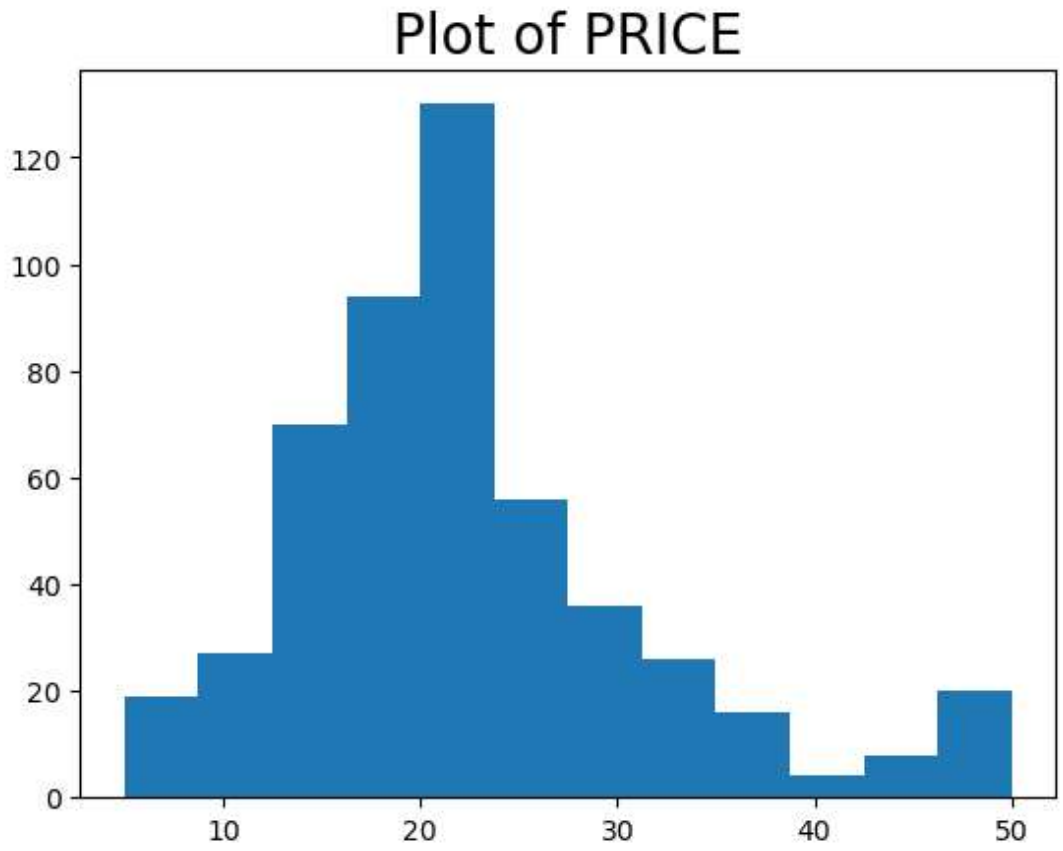

Plot of AGE
140
120
100
80
60
40
20
0
0
20
40
60
80
100


Plot of DIS
120
100
80
60
40
20
0
2
4
6
8
10
12

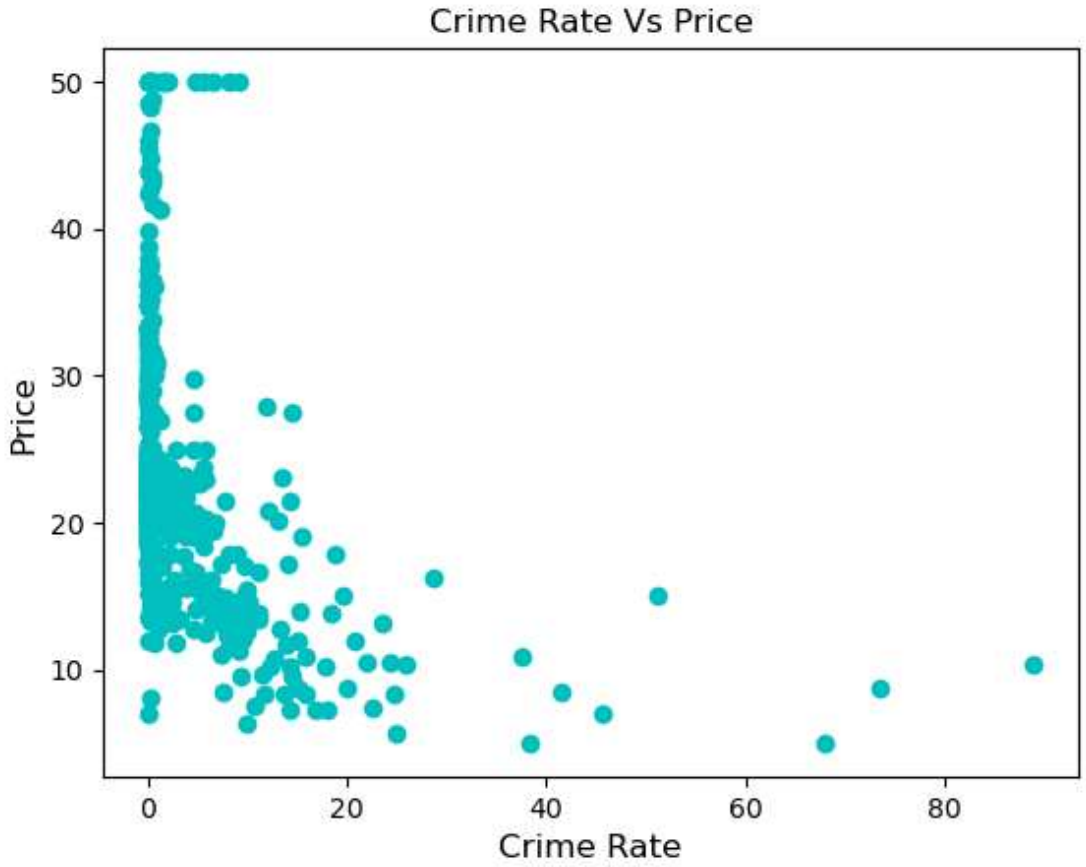

Plot of RAD
140
120
100
80
60
40
20
0
0
5
10
15
20
25


Plot of TAX
140
120
100
80
60
40
20
0
200
300
400
500
600
700


Plot of PTRATIO
160
140
120
100
80
60
40
20
0
14
16
18
20
22


Plot of PRICE
120
100
80
60
40
20
0
10
20
30
40
50

In [38]:
```
#The scatter plot below shows the crime rate vs the price of housing for the dataset.
#We can see the crime has a correlative relarionship with housing pricing.

plt.scatter(smll_df['CRIM'],smll_df['PRICE'], color = 'c')
plt.title('Crime Rate Vs Price')
plt.xlabel('Crime Rate', fontsize = 12)
plt.ylabel('Price', fontsize = 12)
plt.show()
```



In [57]:
```
#We can understand the relationship better if we plot log10(crime) vs. Price.
#Create that plot and make it nice. Give proper title, x-axis, y-axis label,
#make data points a color of your choice, etc...

plt.scatter(np.log10(smll_df['CRIM']), smll_df['PRICE'],c='orange')
plt.title('Crime rate vs. Home Price', fontsize = 20)
plt.xlabel('Crime Rate',fontsize = 20)
plt.ylabel('Home Price',fontsize = 20)
plt.grid()
plt.show()
```

In [58]:
```
#Can you calculate the mean rooms per dwelling?
#the mean of room per dwelling is 6.284634387351787

smll_df['RM'].mean()
```

Out[58]:
```
6.284634387351787
```

In [60]:
```
#Can you calculate median Age?
#the median age is 77.5

smll_df['AGE'].median()
```

Out[60]:
```
77.5
```

In [63]:
```
#Can you calculate average (mean) distances to five Boston employment centres?
#the mean dist to five Boston emply. centers is 3.795042687747034

smll_df['DIS'].mean()
```

Out[63]:
```
3.795042687747034
```

In [82]:
```
#Tricky question: Can you calculate the percentage of houses with low price (< $20,000)?
#creating a boolean for the cost of homes being over or under 20k. This will help to determine
#the percentage of homes over and under 20k. We can see the data type as dtype:bool

lw_hme_pri = smll_df['PRICE'] < 20
lw_hme_pri
```

Out[82]:
```
0      False
1      False
2      False
3      False
4      False
       ...
501    False
502    False
503    False
504    False
505     True
Name: PRICE, Length: 506, dtype: bool
```

In [90]:
```
#we want to find the mean of the home pricing for under 20k. I used the mean of price times 100
#to determine the that Percentage of homes that are lower than 20k is:  41.50197628458498

hm_pcnt = lw_hme_pri.mean() * 100
print("Percentage of homes that are lower than 20k is:", hm_pcnt)
```

```
Percentage of homes that are lower than 20k is: 41.50197628458498
```

In [ ]:
```

```

In [ ]:
```
#2. Data Wrangling with Python: Activity 6, page 171
#the necessary libraries are loaded up top in the second - fourth cell
```

In [111]:
```
#Reading the adult income data and viewing the set as df2 - dataframe

df2 = pd.read_csv("adult_income_data.csv")
df2
```

Out[111]:

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174	0	40	United-States	<=50K
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0	0	40	United-States	<=50K
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0	0	40	Cuba	<=50K
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Female	0	0	40	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	Female	0	0	38	United-States	<=50K
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	Male	0	0	40	United-States	>50K
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	Female	0	0	40	United-States	<=50K
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	Male	0	0	20	United-States	<=50K
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	Female	15024	0	40	United-States	>50K

32560 rows × 14 columns

In [112]:
```
df2.head()
```

Out[112]:

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174	0	40	United-States	<=50K
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	13	United-States	<=50K
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0	0	40	United-States	<=50K
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0	0	40	United-States	<=50K
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0	0	40	Cuba	<=50K
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Female	0	0	40	United-States	<=50K

In [113]:
```
df2.tail()
```

Out[113]:

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174	0	40	United-States	<=50K
32555	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	Female	0	0	38	United-States	<=50K
32556	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	Male	0	0	40	United-States	>50K
32557	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	Female	0	0	40	United-States	<=50K
32558	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	Male	0	0	20	United-States	<=50K
32559	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	Female	15024	0	40	United-States	>50K

In [114]:
```
#Time to read in the text file with data descriptions and extract header names
#Write a file reading script which reads the text file line by line,
#and extracts the first phrase which is the header name
#opening the text file for income names. The list of names is the output.

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

Out[114]:
```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'sex',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country']
```

In [115]:
```
#Add a name ("Income") for the response variable (last column) to the dataset
#and read it again with the column names supplied

names.append("Income")
names
```

Out[115]:
```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'sex',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country',
 'Income']
```

In [117]:
```
df2 = pd.read_csv("adult_income_data.csv",names = names)
df2
```

Out[117]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capital-gain	capital-loss	hours-per-week	native-country	Inc
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174	0	40	United-States	<=
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	13	United-States	<=
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0	0	40	United-States	<=
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0	0	40	United-States	<=
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0	0	40	Cuba	<=
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	Female	0	0	38	United-States	<=
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	Male	0	0	40	United-States	>
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	Female	0	0	40	United-States	<=
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	Male	0	0	20	United-States	<=
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	Female	15024	0	40	United-States	>

32561 rows × 14 columns

In [118]:
```
df2.head()
```

Out[118]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capital-gain	capital-loss	hours-per-week	native-country	Income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0	0	40	Cuba	<=50K

In [119]:
```
df2.tail()
```

Out[119]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	sex	capital-gain	capital-loss	hours-per-week	native-country	Inc
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	Female	0	0	38	United-States	<=
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	Male	0	0	40	United-States	>
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	Female	0	0	40	United-States	<=
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	Male	0	0	20	United-States	<=
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	Female	15024	0	40	United-States	>

In [120]:
```
#finding the missing values using describe stats as a starting point. We can see descriptive stats such as mean

df2.describe()
```

Out[120]:

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

In [131]:
```
#Many variables in the dataset have multiple factors or classes. Can you write a loop to count and print them?

vari_classes = ['workclass','education','marital-status','occupation','relationship','sex','native-country']
```

In [132]:
```
for v in vari_classes:
    classes=df2[v].unique()
    num_classes = df2[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']
----------------------------------------------------------------------------------------------------
```

In [133]:
```
#Is there any missing (NULL) data in the dataset? Write a single line of code to show this for all coumns

df2.isnull().sum()
```

Out[133]:
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
Income            0
dtype: int64
```

In [139]:
```
#Create a dataframe with only age, edu., and occupation by using subsetting

df2_subset = df2[['age','education','occupation']]
df2_subset
```

Out[139]:

	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
...	...	...	...
32556	27	Assoc-acdm	Tech-support
32557	40	HS-grad	Machine-op-inspct
32558	58	HS-grad	Adm-clerical
32559	22	HS-grad	Adm-clerical
32560	52	HS-grad	Exec-managerial

32561 rows × 3 columns

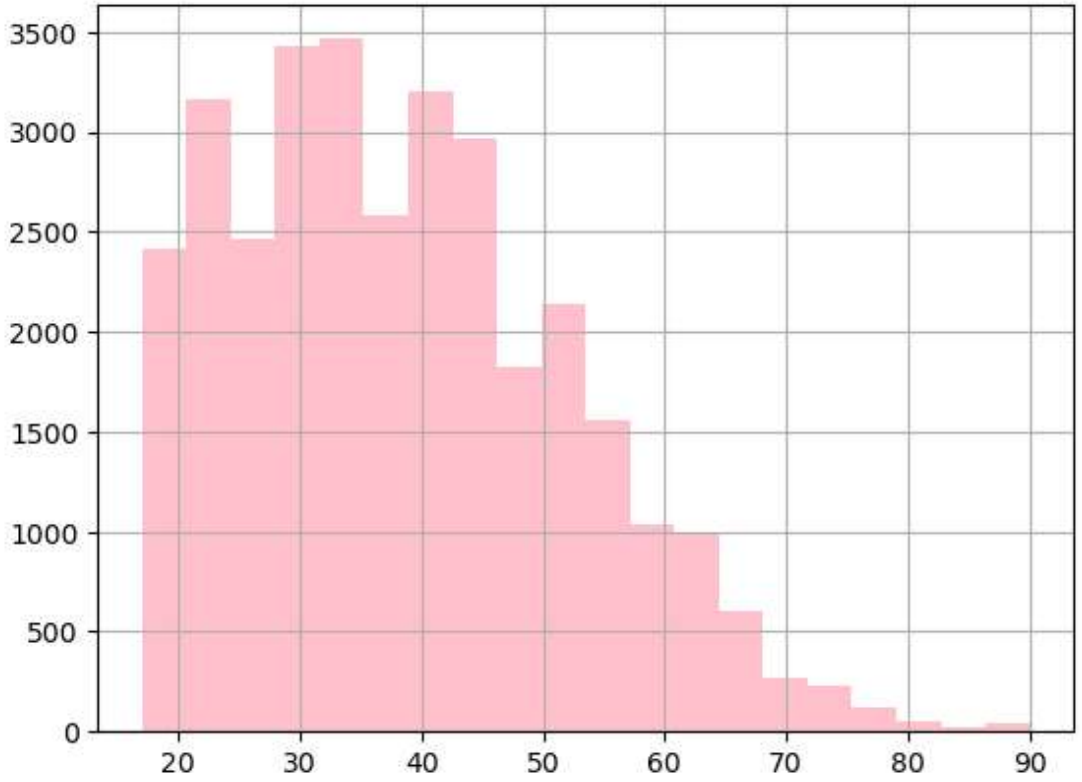
In [140]:
```
df2_subset.head(10)
```

Out[140]:

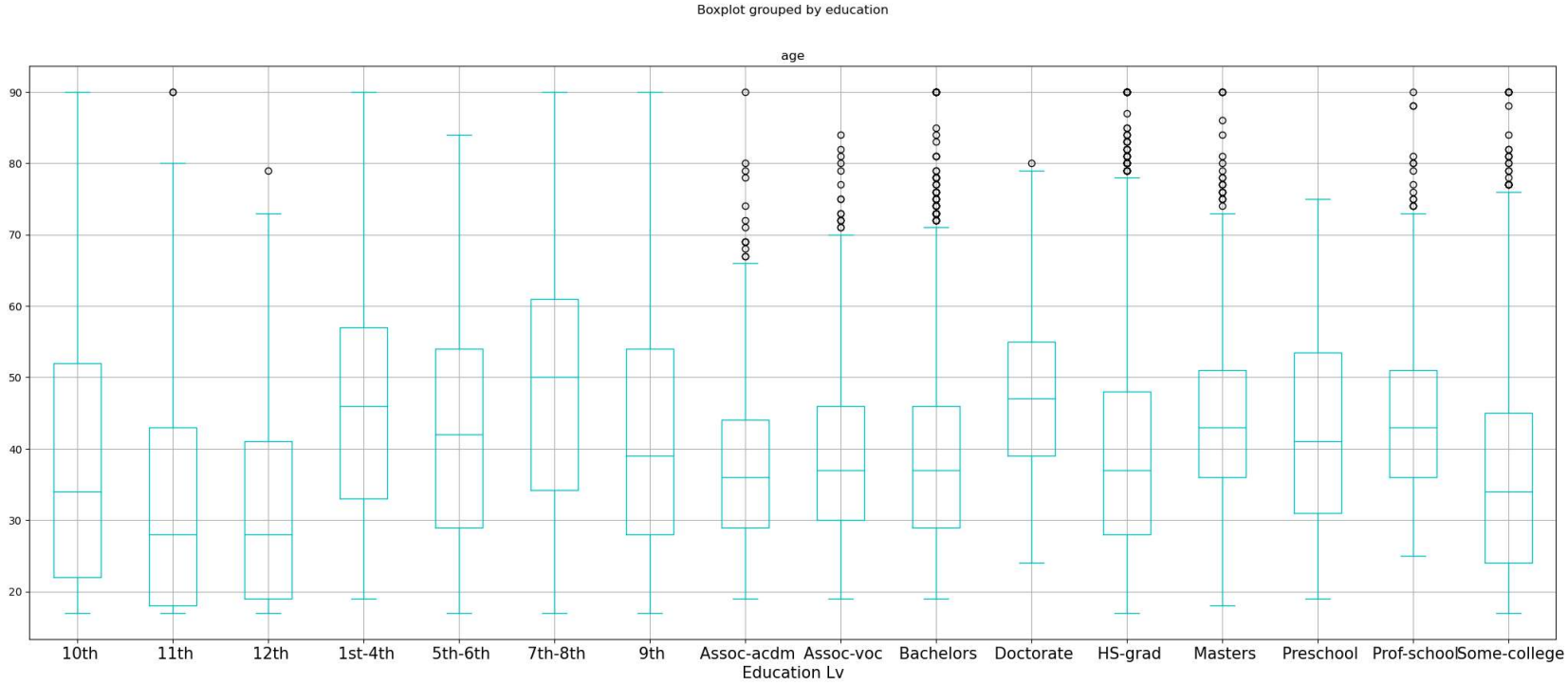
	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners
4	28	Bachelors	Prof-specialty
5	37	Masters	Exec-managerial
6	49	9th	Other-service
7	52	HS-grad	Exec-managerial
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial

In [146]:
```
#plot a histogram of the age with a bin size of 20 and using pink as the color

df2_subset['age'].hist(bins = 20, color = 'pink')
```

Out[146]: <AxesSubplot:>



In [157]:
```
#Plot boxplots for age grouped by education. Use a long fig. size 25x10
#and make x ticks font size 15

df2_subset.boxplot(column='age',by='education',figsize = (25,10), color = 'c')
plt.xticks(fontsize = 15)
plt.xlabel("Education Lv",fontsize = 15)
plt.show()
```



In [158]:
```
#create a function to strip the whitespace chrts.

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

In [161]:
```
#Using the apply method for the columns w/string values. The colunms are going to be education
#and occupation
```

In [159]:
```
#the education column with the apply method - ignoring the message below

df2_subset['education_stripped']=df2['education'].apply(strip_whitespace)
df2_subset['education']=df2_subset['education_stripped']
df2_subset.drop(labels=['education_stripped'],axis=1,inplace=True)
```

```
C:\Users\paul_\AppData\Local\Temp\ipykernel_17560\450515344.py:1: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2_subset['education_stripped']=df2['education'].apply(strip_whitespace)
C:\Users\paul_\AppData\Local\Temp\ipykernel_17560\450515344.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2_subset['education']=df2_subset['education_stripped']
C:\Users\paul_\AppData\Local\Temp\ipykernel_17560\450515344.py:3: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2_subset.drop(labels=['education_stripped'],axis=1,inplace=True)
```

In [160]:
```
#the occupation column with the apply method - ignoring the message below

df2_subset['occupation_stripped']=df2['occupation'].apply(strip_whitespace)
df2_subset['occupation']=df2_subset['occupation_stripped']
df2_subset.drop(labels=['occupation_stripped'],axis=1,inplace=True)
```

```
C:\Users\paul_\AppData\Local\Temp\ipykernel_17560\1728677784.py:1: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2_subset['occupation_stripped']=df2['occupation'].apply(strip_whitespace)
C:\Users\paul_\AppData\Local\Temp\ipykernel_17560\1728677784.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2_subset['occupation']=df2_subset['occupation_stripped']
C:\Users\paul_\AppData\Local\Temp\ipykernel_17560\1728677784.py:3: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df2_subset.drop(labels=['occupation_stripped'],axis=1,inplace=True)
```

In [172]:
```
#find the number of people who are aged between 30-50
#the following command filters the subset for age 30 - 50

df2_filtered = df2_subset[(df2_subset['age'] >=30) & (df2_subset['age'] <=50)]
df2_filtered
```

Out[172]:

	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
...	...	...	...
32550	43	Some-college	Craft-repair
32551	32	10th	Handlers-cleaners
32552	43	Assoc-voc	Sales
32553	32	Masters	Tech-support
32557	40	HS-grad	Machine-op-inspct

16390 rows × 3 columns

In [173]:
```
df2_filtered.head(10)
```

Out[173]:

	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
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial
10	37	Some-college	Exec-managerial
11	30	Bachelors	Prof-specialty
13	32	Assoc-acdm	Sales

In [175]:
```
#there are 16390 people between the ages of 30-50.

shp_df = df2_filtered.shape[0]
shp_df
```

Out[175]:
```
16390
```

In [177]:
```
#Group the records based on age and education to find how the mean age is distributed

df2_subset.groupby('occupation').describe()['age']
```

Out[177]:

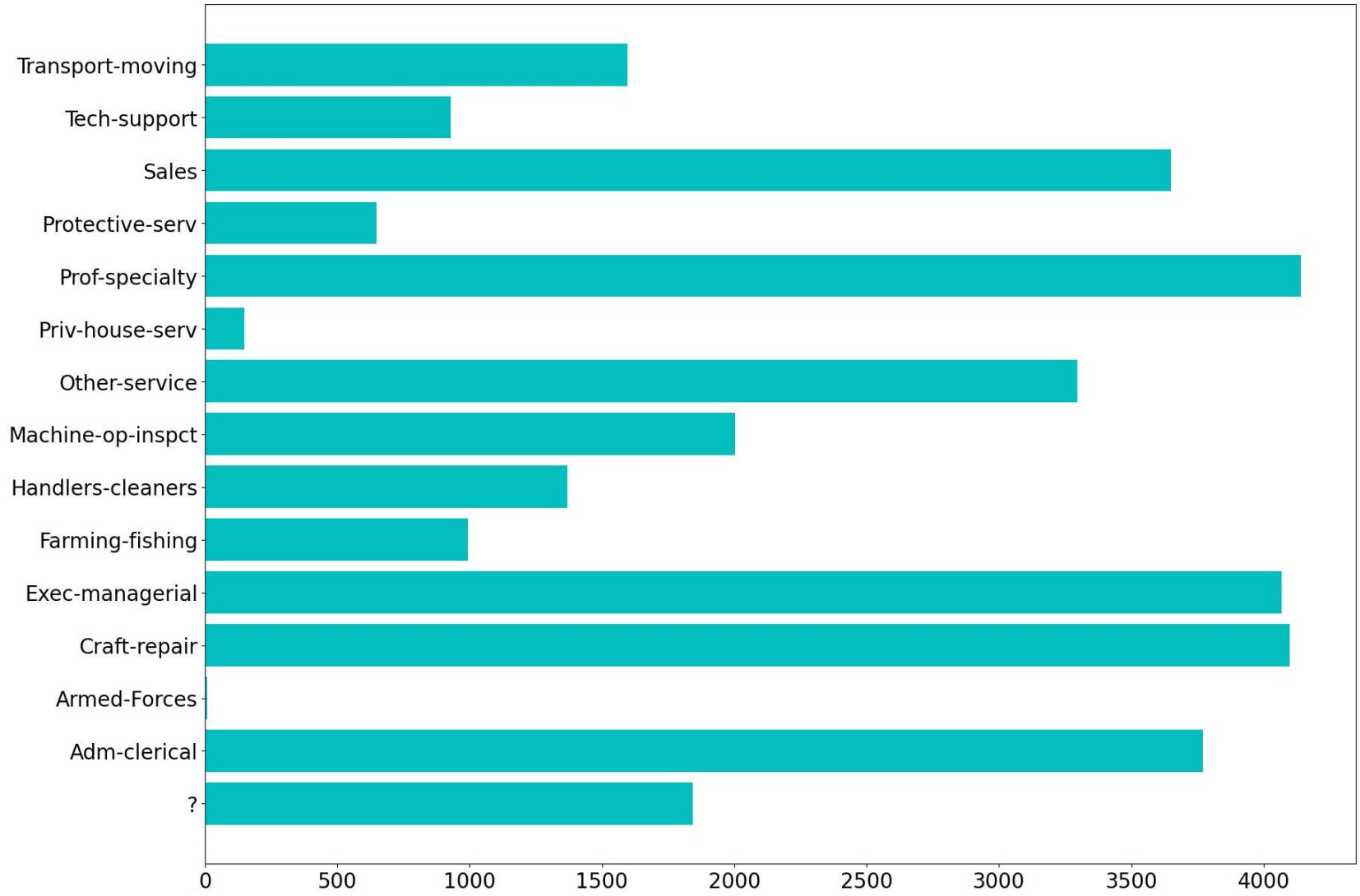
occupation	count	mean	std	min	25%	50%	75%	max
?	1843.0	40.882800	20.336350	17.0	21.0	35.0	61.0	90.0
Adm-clerical	3770.0	36.964456	13.362998	17.0	26.0	35.0	46.0	90.0
Armed-Forces	9.0	30.222222	8.089774	23.0	24.0	29.0	34.0	46.0
Craft-repair	4099.0	39.031471	11.606436	17.0	30.0	38.0	47.0	90.0
Exec-managerial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

In [181]:
```
occ_sts = df2_subset.groupby('occupation').describe()['age']
occ_sts
```

Out[181]:

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

In [191]:
```
#plotting the values on a bar chart

plt.figure(figsize=(20,15))
plt.barh(y=occupation_stats.index,width=occupation_stats['count'], color = 'c')
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.show()
```



In [199]:
```
#Practice for merging. We have age, workclass, and occupation with the printed results, Sample size will be 10

df_3 = df2[['age','workclass','occupation']].sample(10,random_state=101)
df_3
```

Out[199]:

	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
27225	58	?	?
13108	37	Local-gov	Other-service
27552	22	Private	Adm-clerical
14043	22	Private	Sales
30313	26	?	?

In [200]:
```
#merging education and occupation. Sample size is also 10

df_4 = df2[['education','occupation']].sample(10,random_state=101)
df_4
```

Out[200]:

	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
27225	Some-college	?
13108	HS-grad	Other-service
27552	Assoc-voc	Adm-clerical
14043	HS-grad	Sales
30313	Some-college	?

In [201]:
```
#merging the new dataframes for the final result with the new table showing the merged results
#from the previous two cells

mg_df = pd.merge(df_3,df_4,on='occupation',how='inner').drop_duplicates()
mg_df
```

Out[201]:

	age	workclass	occupation	education
0	51	Private	Machine-op-inspct	HS-grad
1	19	Private	Sales	11th
2	19	Private	Sales	HS-grad
3	22	Private	Sales	11th
4	22	Private	Sales	HS-grad
5	40	Private	Exec-managerial	HS-grad
6	17	Private	Handlers-cleaners	10th
7	61	Private	Craft-repair	7th-8th
8	58	?	?	Some-college
10	26	?	?	Some-college
12	37	Local-gov	Other-service	HS-grad
13	22	Private	Adm-clerical	Assoc-voc

In [ ]:

In [219]:
```
#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
```

In [243]:
```
a = pd.Series([7.3, -2.5, 3.4, 1.5], index=['a', 'c', 'd', 'e'])
a
```

Out[243]:
```
a    7.3
c   -2.5
d    3.4
e    1.5
dtype: float64
```

In [244]:
```
b = pd.Series([-2.1, 3.6, -1.5, 4, 3.1], index=['a', 'c', 'e', 'f', 'g'])
b
```

Out[244]:
```
a   -2.1
c    3.6
e   -1.5
f    4.0
g    3.1
dtype: float64
```

In [245]:
```
#c. Add Series 1 and Series 2 together and print the results

a.add(b, fill_value=0)
```

Out[245]:
```
a    5.2
c    1.1
d    3.4
e    0.0
f    4.0
g    3.1
dtype: float64
```

In [246]:
```
#d. Subtract Series 1 from Series 2 and print the results

a.subtract(b, fill_value=0)
```

Out[246]:
```
a    9.4
c   -6.1
d    3.4
e    3.0
f   -4.0
g   -3.1
dtype: float64
```