In [1]:
```
"""
Created on Fri Aug 26 12:59:26 2022

@author: Brandon Botzer - btb5103

"""
```

Out[1]:
```
'\nCreated on Fri Aug 26 12:59:26 2022\n\n@author: Brandon Botzer - btb5103\n
\n'
```

In [2]:

```
"""
Perform the following actions:
Import data mtcars.csv  Download mtcars.csv into Python. (10 points)
"""

#import the relevant libraries / packages
import os
from pandas import Series, DataFrame
import pandas as pd
import numpy as np


#Set the path for the CSV file

readPath = "J:\DSDegree\PennState\DAN_862\Week 3\Homework"

#Change the directory
os.chdir(readPath)

#Read the CSV file in

mtcars = pd.read_csv("mtcars.csv")

print(mtcars)
```

```
                 model   mpg  cyl   disp   hp  drat     wt   qsec  vs  am  \
0            Mazda RX4  21.0    6  160.0  110  3.90  2.620  16.46   0   1
1        Mazda RX4 Wag  21.0    6  160.0  110  3.90  2.875  17.02   0   1
2           Datsun 710  22.8    4  108.0   93  3.85  2.320  18.61   1   1
3       Hornet 4 Drive  21.4    6  258.0  110  3.08  3.215  19.44   1   0
4    Hornet Sportabout  18.7    8  360.0  175  3.15  3.440  17.02   0   0
5              Valiant  18.1    6  225.0  105  2.76  3.460  20.22   1   0
6           Duster 360  14.3    8  360.0  245  3.21  3.570  15.84   0   0
7            Merc 240D  24.4    4  146.7   62  3.69  3.190  20.00   1   0
8             Merc 230  22.8    4  140.8   95  3.92  3.150  22.90   1   0
9             Merc 280  19.2    6  167.6  123  3.92  3.440  18.30   1   0
10            Merc 280C  17.8    6  167.6  123  3.92  3.440  18.90   1   0
11           Merc 450SE  16.4    8  275.8  180  3.07  4.070  17.40   0   0
12           Merc 450SL  17.3    8  275.8  180  3.07  3.730  17.60   0   0
13          Merc 450SLC  15.2    8  275.8  180  3.07  3.780  18.00   0   0
14   Cadillac Fleetwood  10.4    8  472.0  205  2.93  5.250  17.98   0   0
15  Lincoln Continental  10.4    8  460.0  215  3.00  5.424  17.82   0   0
16    Chrysler Imperial  14.7    8  440.0  230  3.23  5.345  17.42   0   0
17             Fiat 128  32.4    4   78.7   66  4.08  2.200  19.47   1   1
18         Honda Civic  30.4    4   75.7   52  4.93  1.615  18.52   1   1
19       Toyota Corolla  33.9    4   71.1   65  4.22  1.835  19.90   1   1
20        Toyota Corona  21.5    4  120.1   97  3.70  2.465  20.01   1   0
21     Dodge Challenger  15.5    8  318.0  150  2.76  3.520  16.87   0   0
22          AMC Javelin  15.2    8  304.0  150  3.15  3.435  17.30   0   0
23           Camaro Z28  13.3    8  350.0  245  3.73  3.840  15.41   0   0
24     Pontiac Firebird  19.2    8  400.0  175  3.08  3.845  17.05   0   0
25            Fiat X1-9  27.3    4   79.0   66  4.08  1.935  18.90   1   1
26        Porsche 914-2  26.0    4  120.3   91  4.43  2.140  16.70   0   1
27         Lotus Europa  30.4    4   95.1  113  3.77  1.513  16.90   1   1
28       Ford Pantera L  15.8    8  351.0  264  4.22  3.170  14.50   0   1
29         Ferrari Dino  19.7    6  145.0  175  3.62  2.770  15.50   0   1
30        Maserati Bora  15.0    8  301.0  335  3.54  3.570  14.60   0   1
```

```
31           Volvo 142E  21.4    4  121.0  109  4.11  2.780  18.60   1   1
```

	gear	carb
0	4	4
1	4	4
2	4	1
3	3	1
4	3	2
5	3	1
6	3	4
7	4	2
8	4	2
9	4	4
10	4	4
11	3	3
12	3	3
13	3	3
14	3	4
15	3	4
16	3	4
17	4	1
18	4	2
19	4	1
20	3	1
21	3	2
22	3	2
23	3	4
24	3	2
25	4	1
26	5	2
27	5	2
28	5	4
29	5	6
30	5	8
31	4	2

In [3]:
```
"""
Explore the data and perform a statistical analysis of the data. (30 points)
"""

print("Explore the data and perform a statistical analysis of the data. (30 point

#I'll look at MPG for the statistical analysis:
    #A classic statistical analysis consists of:
        #mean, min, q1, median, q3, max, std dev across all car types


avgMPG = mtcars['mpg'].mean()
stdMPG = mtcars['mpg'].std()
minMPG = mtcars['mpg'].min()
q1MPG = mtcars['mpg'].quantile(0.25)
meadMPG = mtcars['mpg'].median()
q3MPG = mtcars['mpg'].quantile(0.75)
maxmPG = mtcars['mpg'].max()

print("Average MPG: " + str(avgMPG) +
      "\nFive number summary: " + str(minMPG) + ", " + str(q1MPG) + ", " +
      str(meadMPG) + ", " + str(q3MPG) + ", " + str(maxmPG) +
      "\nStandard Deviation: " + str(stdMPG))
```

```
Explore the data and perform a statistical analysis of the data. (30 points)
Average MPG: 20.090624999999996
Five number summary: 10.4, 15.425, 19.2, 22.8, 33.9
Standard Deviation: 6.026948052089105
```

In [4]:
```
#Function to be used later... probably should have used it earlier...
def stats_Analysis(dataColumn):

    avg = dataColumn.mean()
    std = dataColumn.std()
    minimum = dataColumn.min()
    q1 = dataColumn.quantile(0.25)
    mead = dataColumn.median()
    q3 = dataColumn.quantile(0.75)
    maximum = dataColumn.max()

    print("Average: " + str(avg) +
          "\nFive number summary: " + str(minimum) + ", " + str(q1) + ", " +
          str(mead) + ", " + str(q3) + ", " + str(maximum) +
          "\nStandard Deviation: " + str(std))
```

In [5]:
```
"""
Analyze mpg for cars with different gears, and show your findings. (20 points)
"""

print("\n\nAnalyze mpg for cars with different gears, and show your findings. (20

#Perform statistical analysis again based on gear instead of type


#Create a new trimmed array with just mpg and gear data
#gearData = mtcars[['mpg', 'gear']]   #Used for testing

#First determine which unique gear listings exist
uniqueGears = mtcars['gear'].unique()
#sort these in ascending
uniqueGears.sort()

#Full data of Gear 4 (can change mtcars -> gearData)
#test = mtcars[mtcars['gear'] == uniqueGears[0]]   #Used for testing

for i in range(0,len(uniqueGears)):

    print("Here is the statistical analysis for MPG based on " + str(uniqueGears[

    #remade a dataColumn of the data to be analyzed (mpg) for easier changes in t
    dataC = mtcars['mpg'][mtcars['gear'] == uniqueGears[i]]

    #Run the statistical analysis function
    stats_Analysis(dataC)

    #Spacer
    print()
```

```
Analyze mpg for cars with different gears, and show your findings. (20 points)
Here is the statistical analysis for MPG based on 3 Gears:
Average: 16.106666666666666
Five number summary: 10.4, 14.5, 15.5, 18.4, 21.5
Standard Deviation: 3.371618235181665

Here is the statistical analysis for MPG based on 4 Gears:
Average: 24.53333333333333
Five number summary: 17.8, 21.0, 22.8, 28.075000000000003, 33.9
Standard Deviation: 5.276764389684498

Here is the statistical analysis for MPG based on 5 Gears:
Average: 21.380000000000003
Five number summary: 15.0, 15.8, 19.7, 26.0, 30.4
Standard Deviation: 6.65897890070242
```

In [6]:
```
"""
Analyze mpg for cars with different carbs, and show your findings. (20 points)
"""

print("\n\nAnalyze mpg for cars with different carbs, and show your findings. (20

#Perform statistical analysis again based on carb instead of gear / type

#Create a new trimmed array with just mpg and gear data
#gearData = mtcars[['mpg', 'gear']]   #Used for testing

#First determine which unique gear listings exist
uniqueCarb = mtcars['carb'].unique()
#sort these in ascending
uniqueCarb.sort()

#Full data of Gear 4 (can change mtcars -> gearData)
#test = mtcars[mtcars['gear'] == uniqueGears[0]]   #Used for testing

for i in range(0,len(uniqueCarb)):

    print("Here is the statistical analysis for MPG based on " + str(uniqueCarb[i

    #remade a dataColumn of the data to be analyzed for easier changes in the fut
    dataC = mtcars['mpg'][mtcars['carb'] == uniqueCarb[i]]

    #Run the statistical analysis function
    stats_Analysis(dataC)

    #Spacer
    print()
```

```
Analyze mpg for cars with different carbs, and show your findings. (20 points)
Here is the statistical analysis for MPG based on 1 Carbs:
Average: 25.342857142857145
Five number summary: 18.1, 21.45, 22.8, 29.85, 33.9
Standard Deviation: 6.001349054686827

Here is the statistical analysis for MPG based on 2 Carbs:
Average: 22.4
Five number summary: 15.2, 18.825, 22.1, 25.6, 30.4
Standard Deviation: 5.472151719794001

Here is the statistical analysis for MPG based on 3 Carbs:
Average: 16.3
Five number summary: 15.2, 15.799999999999999, 16.4, 16.85, 17.3
Standard Deviation: 1.0535653752852745

Here is the statistical analysis for MPG based on 4 Carbs:
Average: 15.790000000000003
Five number summary: 10.4, 13.55, 15.25, 18.849999999999998, 21.0
Standard Deviation: 3.911081123622413
```

Here is the statistical analysis for MPG based on 6 Carbs:
Average: 19.7
Five number summary: 19.7, 19.7, 19.7, 19.7, 19.7
Standard Deviation: nan

Here is the statistical analysis for MPG based on 8 Carbs:
Average: 15.0
Five number summary: 15.0, 15.0, 15.0, 15.0, 15.0
Standard Deviation: nan

In [7]:

```
"""
Find out which attribute has the most impact on mpg. (20 points)
"""

print("\n\nFind out which attribute has the most impact on mpg. (20 points)")


#Compute the Correclation matrix

corrCoef = mtcars.corr()

print("The Correlation Matrix: \n " + str(corrCoef))

#Compute the Covariance matrix

covCoef = mtcars.cov()

print("\nThe Covariance Matrix:" + str(covCoef))

#Note: the correlation is the covariance divided by the std dev of the
#two std devs being compared.  Thus I'll stick to the correlation Coef
#for my final analysis
print("""\n\nNote: the correlation is the covariance divided by the two std devs
      of the variables being compared.  Thus I'll stick to the
      correlation Coef for my final analysis\n\n""")



#We can display all of the relevant correlation Coef
#I have dropped the mpg corr value here as it is natually 1.0 and irrelevant
print("We can display all of the relevant correlation coef")
print(str(corrCoef['mpg'].drop('mpg')))

#If we are just looking at the MOST impact, let us look at these from an absolute
print("\nAbsolute values of the corr coef:\n" + str(corrCoef['mpg'].drop('mpg').a

#If we'd like these ordered
print("\nOrdered absolute values of the corr coef: \n" + str(corrCoef['mpg'].drop

#Largest correclation Coefficient
print("\nThe largest value of the correlation coefficients is from: " + str(corrC
```

```
Find out which attribute has the most impact on mpg. (20 points)
The Correlation Matrix:
            mpg       cyl      disp        hp      drat        wt      qsec
\
mpg   1.000000 -0.852162 -0.847551 -0.776168  0.681172 -0.867659  0.418684
cyl  -0.852162  1.000000  0.902033  0.832447 -0.699938  0.782496 -0.591242
disp -0.847551  0.902033  1.000000  0.790949 -0.710214  0.887980 -0.433698
hp   -0.776168  0.832447  0.790949  1.000000 -0.448759  0.658748 -0.708223
drat  0.681172 -0.699938 -0.710214 -0.448759  1.000000 -0.712441  0.091205
wt   -0.867659  0.782496  0.887980  0.658748 -0.712441  1.000000 -0.174716
qsec  0.418684 -0.591242 -0.433698 -0.708223  0.091205 -0.174716  1.000000
vs    0.664039 -0.810812 -0.710416 -0.723097  0.440278 -0.554916  0.744535
```

```
am    0.599832 -0.522607 -0.591227 -0.243204  0.712711 -0.692495 -0.229861
gear  0.480285 -0.492687 -0.555569 -0.125704  0.699610 -0.583287 -0.212682
carb -0.550925  0.526988  0.394977  0.749812 -0.090790  0.427606 -0.656249

            vs        am      gear      carb
mpg   0.664039  0.599832  0.480285 -0.550925
cyl  -0.810812 -0.522607 -0.492687  0.526988
disp -0.710416 -0.591227 -0.555569  0.394977
hp   -0.723097 -0.243204 -0.125704  0.749812
drat  0.440278  0.712711  0.699610 -0.090790
wt   -0.554916 -0.692495 -0.583287  0.427606
qsec  0.744535 -0.229861 -0.212682 -0.656249
vs    1.000000  0.168345  0.206023 -0.569607
am    0.168345  1.000000  0.794059  0.057534
gear  0.206023  0.794059  1.000000  0.274073
carb -0.569607  0.057534  0.274073  1.000000

The Covariance Matrix:              mpg         cyl          disp            hp
drat  \
mpg    36.324103   -9.172379   -633.097208  -320.732056   2.195064
cyl    -9.172379    3.189516    199.660282   101.931452  -0.668367
disp -633.097208  199.660282  15360.799829  6721.158669 -47.064019
hp   -320.732056  101.931452   6721.158669  4700.866935 -16.451109
drat    2.195064   -0.668367    -47.064019   -16.451109   0.285881
wt     -5.116685    1.367371    107.684204    44.192661  -0.372721
qsec    4.509149   -1.886855    -96.051681   -86.770081   0.087141
vs      2.017137   -0.729839    -44.377621   -24.987903   0.118649
am      1.803931   -0.465726    -36.564012    -8.320565   0.190151
gear    2.135685   -0.649194    -50.802621    -6.358871   0.275988
carb   -5.363105    1.520161     79.068750    83.036290  -0.078407

              wt       qsec         vs         am       gear       carb
mpg    -5.116685   4.509149   2.017137   1.803931   2.135685  -5.363105
cyl     1.367371  -1.886855  -0.729839  -0.465726  -0.649194   1.520161
disp  107.684204 -96.051681 -44.377621 -36.564012 -50.802621  79.068750
hp     44.192661 -86.770081 -24.987903  -8.320565  -6.358871  83.036290
drat   -0.372721   0.087141   0.118649   0.190151   0.275988  -0.078407
wt      0.957379  -0.305482  -0.273661  -0.338105  -0.421081   0.675790
qsec   -0.305482   3.193166   0.670565  -0.204960  -0.280403  -1.894113
vs     -0.273661   0.670565   0.254032   0.042339   0.076613  -0.463710
am     -0.338105  -0.204960   0.042339   0.248992   0.292339   0.046371
gear   -0.421081  -0.280403   0.076613   0.292339   0.544355   0.326613
carb    0.675790  -1.894113  -0.463710   0.046371   0.326613   2.608871


Note: the correlation is the covariance divided by the two std devs
      of the variables being compared.  Thus I'll stick to the
      correlation Coef for my final analysis


We can display all of the relevant correlation coef
cyl    -0.852162
disp   -0.847551
hp     -0.776168
drat    0.681172
wt     -0.867659
qsec    0.418684
```

```
vs      0.664039
am      0.599832
gear    0.480285
carb   -0.550925
Name: mpg, dtype: float64

Absolute values of the corr coef:
cyl     0.852162
disp    0.847551
hp      0.776168
drat    0.681172
wt      0.867659
qsec    0.418684
vs      0.664039
am      0.599832
gear    0.480285
carb    0.550925
Name: mpg, dtype: float64

Ordered absolute values of the corr coef:
wt      0.867659
cyl     0.852162
disp    0.847551
hp      0.776168
drat    0.681172
vs      0.664039
am      0.599832
carb    0.550925
gear    0.480285
qsec    0.418684
Name: mpg, dtype: float64

The largest value of the correlation coefficients is from: wt
```