Assignment 3

1. Import data mtcars.csv using read_csv function. (10 points)

In [1]:
```
import os
import pandas as pd
```

In [2]:
```
os.chdir("E:/GoogleDrive/PSU/DAAN862/Course contents/Lesson 3")
mtcars = pd.read_csv("mtcars.csv")
```

In [3]:
```
mtcars.head()
```

Out[3]:

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

2. Explore the data and perform a statistical analysis of the data. (30 points)

Explore the basic information of data:

In [4]:
```
mtcars.columns
```

Out[4]:
```
Index(['model', 'mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am',
       'gear', 'carb'],
      dtype='object')
```

In [5]:
```
mtcars.shape
```

Out[5]:
```
(32, 12)
```

Only disp, hp, drat, wt, qsec are numeric variables:

In [6]:
```
mtcars.loc[:, "disp":"qsec"].describe().round(2)   # Rount is used to round a
 DataFrame to a variable number of decimal places.
```

Out[6]:

	disp	hp	drat	wt	qsec
count	32.00	32.00	32.00	32.00	32.00
mean	230.72	146.69	3.60	3.22	17.85
std	123.94	68.56	0.53	0.98	1.79
min	71.10	52.00	2.76	1.51	14.50
25%	120.82	96.50	3.08	2.58	16.89
50%	196.30	123.00	3.70	3.32	17.71
75%	326.00	180.00	3.92	3.61	18.90
max	472.00	335.00	4.93	5.42	22.90

For categorical variable we perform:

In [7]:
```
mtcars['model'].unique().size
```

Out[7]: 32

The number of unique values in Model equals to the number of rows. All models are unique.Model works like the ID numbers, since the it has uniqe value for each row.

For the rest variables:

In [8]:
```
for name in ['cyl', 'vs', 'am', 'gear', 'carb']:
    print("The value counts for", name, "are :")
    print(mtcars[name].value_counts(), '\n')
```

```
The value counts for cyl are :
8    14
4    11
6     7
Name: cyl, dtype: int64

The value counts for vs are :
0    18
1    14
Name: vs, dtype: int64

The value counts for am are :
0    19
1    13
Name: am, dtype: int64

The value counts for gear are :
3    15
4    12
5     5
Name: gear, dtype: int64

The value counts for carb are :
4    10
2    10
1     7
3     3
8     1
6     1
Name: carb, dtype: int64
```

3. Analyze mpg for cars with different gear, and show your findings. (20 points)

In [9]:
```
gear3 = mtcars.loc[mtcars.gear == 3, ]
gear4 = mtcars.loc[mtcars.gear == 4, ]
gear5 = mtcars.loc[mtcars.gear == 5, ]
GearCompar = pd.DataFrame(columns = ['gear3', 'gear4', 'gear5'])
GearCompar['gear3'] = gear3.mpg.describe()
GearCompar['gear4'] = gear4.mpg.describe()
GearCompar['gear5'] = gear5.mpg.describe()
GearCompar
```

Out[9]:

	gear3	gear4	gear5
count	15.000000	12.000000	5.000000
mean	16.106667	24.533333	21.380000
std	3.371618	5.276764	6.658979
min	10.400000	17.800000	15.000000
25%	14.500000	21.000000	15.800000
50%	15.500000	22.800000	19.700000
75%	18.400000	28.075000	26.000000
max	21.500000	33.900000	30.400000

Here I manually created each columns. If there are more categories, it is not feasible. In Q4, I tried to write a for loop to add categories.

The purpose of Q3 and Q5 is to practice manually selection of data. This can be easily done by groupby method.

In [10]:
```
mtcars[['mpg', 'gear']].groupby(['gear']).agg('describe')
```

Out[10]:

	mpg							
	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
gear								
3	15.0	16.106667	3.371618	10.4	14.5	15.5	18.400	21.5
4	12.0	24.533333	5.276764	17.8	21.0	22.8	28.075	33.9
5	5.0	21.380000	6.658979	15.0	15.8	19.7	26.000	30.4

4. Analyze mpg for cars with different carb, and show your findings. (20 points)

In [11]:
```
columns = []
carb_unique = mtcars.carb.unique()
carb_unique.sort()
for i in carb_unique:
    columns.append('carb' + str(i))
columns
```

Out[11]: ['carb1', 'carb2', 'carb3', 'carb4', 'carb6', 'carb8']

In [12]:
```
carbCompar = pd.DataFrame(columns = columns)
for i in range(len(carb_unique)):
    carbCompar[columns[i]] = mtcars.loc[mtcars.carb == carb_unique[i], ].mpg.d
escribe()
carbCompar
```

Out[12]:

	carb1	carb2	carb3	carb4	carb6	carb8
count	7.000000	10.000000	3.000000	10.000000	1.0	1.0
mean	25.342857	22.400000	16.300000	15.790000	19.7	15.0
std	6.001349	5.472152	1.053565	3.911081	NaN	NaN
min	18.100000	15.200000	15.200000	10.400000	19.7	15.0
25%	21.450000	18.825000	15.800000	13.550000	19.7	15.0
50%	22.800000	22.100000	16.400000	15.250000	19.7	15.0
75%	29.850000	25.600000	16.850000	18.850000	19.7	15.0
max	33.900000	30.400000	17.300000	21.000000	19.7	15.0

Since we only two datapoints for 6 and 8 carb, it is difficult to draw any conclusion. With increase of carb, the mean mpg descrese.

In [13]:
```
mtcars[['mpg', 'carb']].groupby(['carb']).agg('describe')
```

Out[13]:

	mpg							
	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
carb								
1	7.0	25.342857	6.001349	18.1	21.450	22.80	29.85	33.9
2	10.0	22.400000	5.472152	15.2	18.825	22.10	25.60	30.4
3	3.0	16.300000	1.053565	15.2	15.800	16.40	16.85	17.3
4	10.0	15.790000	3.911081	10.4	13.550	15.25	18.85	21.0
6	1.0	19.700000	NaN	19.7	19.700	19.70	19.70	19.7
8	1.0	15.000000	NaN	15.0	15.000	15.00	15.00	15.0

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

In [14]:
```
mtcars.corrwith(mtcars.mpg).abs().sort_values(ascending = False)
```

Out[14]:
```
mpg     1.000000
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
dtype: float64
```

Based on what we learn so far, correlation is an appropriate tool to solve this problem. The wt has highest correlation coefficient with mpg, therefor it has the most impact on mpg.