The Data Science Environment

SET A

In [1]:
```
#Q1. Create and view a data frame
#import the library
import pandas as pd
import numpy as np
#Enter Data
data_values={'Name':['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'],
 'Age' : [26, 28, 20, 15, 20, 16, 18, 17, 22, 21],
 'Percentage' : [56,62,42,74,32,63,74,84,96,21]
 }
#Create empty dataframe with column names
data=pd.DataFrame.from_dict(data_values)
data #To view the data frame
```

Out[1]:

	Name	Age	Percentage
0	A	26	56
1	B	28	62
2	C	20	42
3	D	15	74
4	E	20	32
5	F	16	63
6	G	18	74
7	H	17	84
8	I	22	96
9	J	21	21

In [2]:
```
#Q2.
#print shape >> number of rows - columns
data.shape
```

Out[2]:
```
(10, 3)
```

In [3]:
```
print("Size = {} \n Shape = {}\n Number of rows = {} \n Number of Columns = {}".
format(data.size, data.shape, data.shape[0], data.shape[1]))
```

```
Size = 30
 Shape = (10, 3)
 Number of rows = 10
 Number of Columns = 3
```

In [4]:
```
#feature names
print("data types")
data.dtypes
```

Out[4]:
```
data types
Name          object
Age            int64
Percentage     int64
dtype: object
```

In [5]:
```
print("Feature Names = {}, {}, {}".
 format(data.columns[0], data.columns[1], data.columns[2]))
```

```
Feature Names = Name, Age, Percentage
```

In [6]:
```
print("Description of Data")
data.info()
```

```
Description of Data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Name        10 non-null     object
 1   Age         10 non-null     int64
 2   Percentage  10 non-null     int64
dtypes: int64(2), object(1)
memory usage: 368.0+ bytes
```

In [7]:
```
#Number of columns with null entries = 0
#Number of columns with numeric data = 2
#Number of columns with categorical data = 1
#Q3. obtaining basic statistical details of the data
data.describe()
```

Out[7]:

	Age	Percentage
count	10.000000	10.000000
mean	20.300000	60.400000
std	4.191261	23.381854
min	15.000000	21.000000
25%	17.250000	45.500000
50%	20.000000	62.500000
75%	21.750000	74.000000
max	28.000000	96.000000

In [8]:
```
# Mean Age = 20.3 yrs ; Mean % = 60.4 %
# Standard Deviation : sd(Age) = 4.191261 ;sd(%) = 23.381854
# Minimum Age =15 yrs ; Maximum Age = 28 yrs
# Minimum % = 21% ; Maximum % = 96%
```

In [9]:
```
#Q4. Adding 5 rows and 1 column
data.loc[10] = ['K',21,56 ]
data.loc[11] = ['L',21,None]
data.loc[12] = ['M',None, 45]
data.loc[13] = ['K',21,56]
data.loc[14] = ['O',25,84]
data["Remarks"] = None
data #data display
```

Out[9]:

	Name	Age	Percentage	Remarks
0	A	26	56	None
1	B	28	62	None
2	C	20	42	None
3	D	15	74	None
4	E	20	32	None
5	F	16	63	None
6	G	18	74	None
7	H	17	84	None
8	I	22	96	None
9	J	21	21	None
10	K	21	56	None
11	L	21	None	None
12	M	None	45	None
13	K	21	56	None
14	O	25	84	None

In [10]:
```
#Q5.
print("Number of Observations = ", len(data.index))
print(" \nTotal missing values in a DataFrame : \n\n", data.isnull().sum().sum())
print(data.duplicated().value_counts()) #number of duplicate values
```

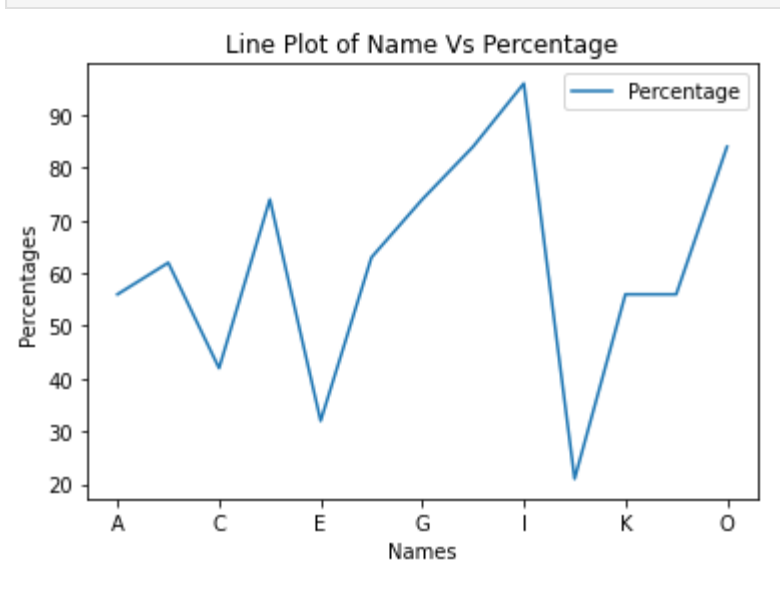
```
Number of Observations =  15

Total missing values in a DataFrame :

 17
False    14
True      1
dtype: int64
```

In [11]:
```
#duplicate observations = 1
#Q6. Removing a column and missing values
data2=data.drop(columns='Remarks')
data2=data2.dropna(axis=0)
#print modified data
data2
```

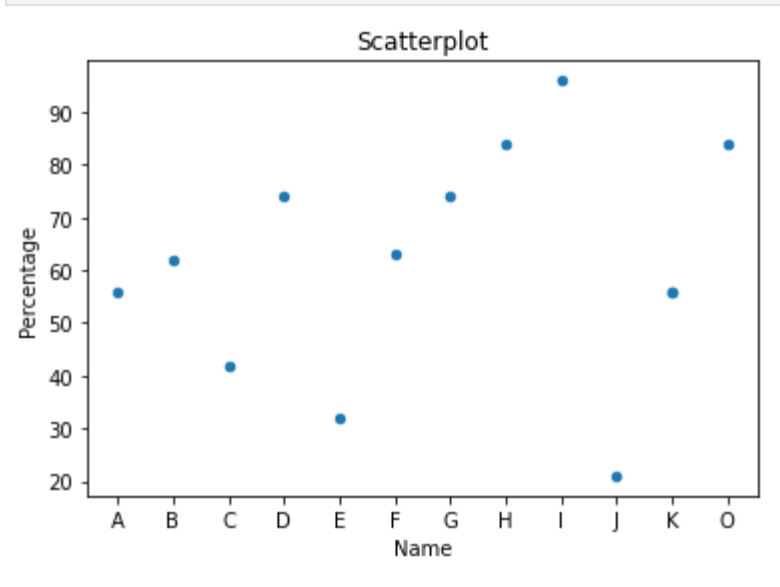
Out[11]:

	Name	Age	Percentage
0	A	26	56
1	B	28	62
2	C	20	42
3	D	15	74
4	E	20	32
5	F	16	63
6	G	18	74
7	H	17	84
8	I	22	96
9	J	21	21
10	K	21	56
13	K	21	56
14	O	25	84

In [12]:
```
#Q7. Line plot
import matplotlib.pyplot as plt
data2.plot(x="Name",y="Percentage",
 title="Line Plot of Name Vs Percentage")
plt.xlabel("Names")
plt.ylabel("Percentages")
plt.show()
```



In [13]:
```
#Q8. Scatterplot
data2.plot.scatter(x='Name',y='Percentage',
                   title = "Scatterplot")
plt.show()
```



SET B

In [14]:
```
import pandas as pd
data=pd.read_csv("C:\\Downloads\\SOCR-HeightWeight.csv")
```

In [15]:
```
data.head(10) #print first 10 rows
```

Out[15]:

	Index	Height(Inches)	Weight(Pounds)
0	1	65.78331	112.9925
1	2	71.51521	136.4873
2	3	69.39874	153.0269
3	4	68.21660	142.3354
4	5	67.78781	144.2971
5	6	68.69784	123.3024
6	7	69.80204	141.4947
7	8	70.01472	136.4623
8	9	67.90265	112.3723
9	10	66.78236	120.6672

In [16]:
```
data.tail(10) #print last 10 rows
```

Out[16]:

	Index	Height(Inches)	Weight(Pounds)
24990	24991	69.97767	125.3672
24991	24992	71.91656	128.2840
24992	24993	70.96218	146.1936
24993	24994	66.19462	118.7974
24994	24995	67.21126	127.6603
24995	24996	69.50215	118.0312
24996	24997	64.54826	120.1932
24997	24998	64.69855	118.2655
24998	24999	67.52918	132.2682
24999	25000	68.87761	124.8742

In [17]:
```
data.sample(20) #print 20 random rows
```

Out[17]:

	Index	Height(Inches)	Weight(Pounds)
16406	16407	69.20068	133.18500
22628	22629	71.92476	137.72010
7103	7104	68.92581	110.00910
15084	15085	69.17547	142.27360
16767	16768	70.57005	145.80510
2305	2306	65.63764	117.85290
21448	21449	67.68578	122.56810
5770	5771	69.72285	134.59230
621	622	64.79753	122.37060
1179	1180	65.74615	134.77170
22207	22208	67.67203	121.01700
14634	14635	67.80629	118.86780
21634	21635	66.40240	117.36940
9823	9824	71.61882	165.87160
12166	12167	67.15786	114.81410
12875	12876	65.09478	133.71830
8137	8138	68.68597	124.68490
12714	12715	67.63379	136.18380
20084	20085	64.30312	99.33579
20497	20498	70.37349	131.55900

In [18]:
```
#Q2.
print("Size = {} \n Shape of DataFrame Object = {}\n Number of rows = {} \n Number of Columns = {}".
format(data.size, data.shape, data.shape[0], data.shape[1]))
print("\n Datatypes of dataframe object")
data.dtypes
```

Out[18]:
```
Size = 75000
 Shape of DataFrame Object = (25000, 3)
 Number of rows = 25000
 Number of Columns = 3

 Datatypes of dataframe object
Index               int64
Height(Inches)    float64
Weight(Pounds)    float64
dtype: object
```

In [19]:
```
#Q3.
data.describe() #basic statistical details
```

Out[19]:

	Index	Height(Inches)	Weight(Pounds)
count	25000.000000	25000.000000	25000.000000
mean	12500.500000	67.993114	127.079421
std	7217.022701	1.901679	11.660898
min	1.000000	60.278360	78.014760
25%	6250.750000	66.704397	119.308675
50%	12500.500000	67.995700	127.157750
75%	18750.250000	69.272958	134.892850
max	25000.000000	75.152800	170.924000

In [20]:
```
#Mean Height = 67.9931 Inches ; Mean Weight = 127.0794 Pounds
#sd(Height) = 1.9017 ; sd(Weight) = 11.6609
#Minimum Height = 60.2784 Inches ; Minimum Weight = 78.0148 Pounds
#Maximum Height = 75.1528 Inches ; Maximum Weight = 170.924 Pounds


#Q4.
print("\n Description of Data")
data.info()
print("\n Number of Observations = ", len(data.index))
print(" \nTotal missing values in a DataFrame = ",data.isnull().sum().sum())
print("Number of duplicate values \n ", data.duplicated().value_counts())
```

```
 Description of Data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Index           25000 non-null  int64
 1   Height(Inches)  25000 non-null  float64
 2   Weight(Pounds)  25000 non-null  float64
dtypes: float64(2), int64(1)
memory usage: 586.1 KB

 Number of Observations =  25000

Total missing values in a DataFrame =  0
Number of duplicate values
  False    25000
dtype: int64
```

In [21]:
```
#Q5.
#Add column "BMI"
data2=data.assign(BMI=data['Weight(Pounds)']/(data['Height(Inches)']*data['Height(Inches)']))
data2.head(1)
```

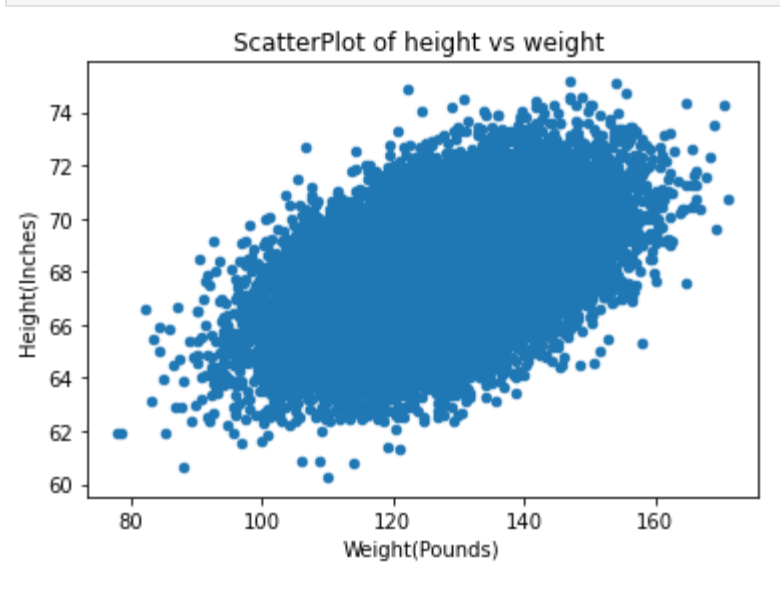
Out[21]:

	Index	Height(Inches)	Weight(Pounds)	BMI
0	1	65.78331	112.9925	0.026111

In [22]:
```
#Q6.
print("Maximum BMI = ",max(data2['BMI']))
print("\n Minimum BMI = ",min(data2['BMI']))
```

```
Maximum BMI =  0.03701443692089851

 Minimum BMI =  0.018591137267932455
```

In [23]:
```
#Q7.
data.plot(x='Weight(Pounds)',y='Height(Inches)',kind="scatter", title = "ScatterPlot of height vs weight ")
plt.show()
```



In [ ]: