Out[61]:

	Days	Places	Visitors
count	6.000000	6	6.00000
unique	NaN	5	NaN
top	NaN	Bangalore	NaN
freq	NaN	2	NaN
mean	3.500000	NaN	3507.00000
std	1.870829	NaN	1856.05334
min	1.000000	NaN	1000.00000
25%	2.250000	NaN	2387.50000
50%	3.500000	NaN	3546.00000
75%	4.750000	NaN	4598.00000
max	6.000000	NaN	6000.00000

Working with Datasets:-

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
# to read the dataset.
df = pd.read_csv(r"C:\Users\CTTC\Downloads\iris\iris.data")
df
```

Out[2]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

149 rows × 5 columns

In the above code 👆 as our dataset doesnot contain the columns name, the 1st row data is being interpreted as column's name.

Therefore, inorder to overcome this we are using "header = None" as below 👇.

In [3]:
```
# to read the dataset.
df = pd.read_csv(r"C:\Users\CTTC\Downloads\iris\iris.data",header=None)
df
```

Out[3]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [5]:
```
# Renaming the columns name:
df.columns = ['Sepal_Length','Sepal_Width','Petal_Length'
              ,'Petal_Width','Class']
```

In [6]:
```
df
```

Out[6]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [8]:
```
# to check the shape of the Dataframe.
# returns the no.of rows and columns in dataset.
df.shape
```

Out[8]:
```
(150, 5)
```

In [10]:
```
# to check the first top values:
df.head(10)
```

Out[10]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

In [12]:
```
# returns the last 5 values.
df.tail()
```

Out[12]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

In [13]:
```
# to check the null values:-
df.isnull()
```

Out[13]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [15]:
```
# returns the total no.of null values present in each col.
df.isnull().sum()
```

Out[15]:
```
Sepal_Length    0
Sepal_Width     0
Petal_Length    0
Petal_Width     0
Class           0
dtype: int64
```

In [17]:
```
# checking the datatype of each column.
df.dtypes
```

Out[17]:
```
Sepal_Length    float64
Sepal_Width     float64
Petal_Length    float64
Petal_Width     float64
Class            object
dtype: object
```

In [19]:
```
# to print the total information of our dataset.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Sepal_Length  150 non-null    float64
 1   Sepal_Width   150 non-null    float64
 2   Petal_Length  150 non-null    float64
 3   Petal_Width   150 non-null    float64
 4   Class         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [20]:
```
# to check the unique values.
df.Sepal_Length.unique()
```

Out[20]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

In [21]:
```
df.Sepal_Width.unique()
```

Out[21]:
```
array([3.5, 3. , 3.2, 3.1, 3.6, 3.9, 3.4, 2.9, 3.7, 4. , 4.4, 3.8, 3.3,
       4.1, 4.2, 2.3, 2.8, 2.4, 2.7, 2. , 2.2, 2.5, 2.6])
```

In [22]:
```
df.columns
```

Out[22]:
```
Index(['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width', 'Class'], dty
pe='object')
```

In [25]:
```
# to find the unique values of all columns at a time.
for i in df.columns:
    print(f"{i}: \n {df[i].unique()}\n")
```

```
Sepal_Length:
 [5.1 4.9 4.7 4.6 5.  5.4 4.4 4.8 4.3 5.8 5.7 5.2 5.5 4.5 5.3 7.  6.4 6.9
 6.5 6.3 6.6 5.9 6.  6.1 5.6 6.7 6.2 6.8 7.1 7.6 7.3 7.2 7.7 7.4 7.9]

Sepal_Width:
 [3.5 3.  3.2 3.1 3.6 3.9 3.4 2.9 3.7 4.  4.4 3.8 3.3 4.1 4.2 2.3 2.8 2.4
 2.7 2.  2.2 2.5 2.6]

Petal_Length:
 [1.4 1.3 1.5 1.7 1.6 1.1 1.2 1.  1.9 4.7 4.5 4.9 4.  4.6 3.3 3.9 3.5 4.2
 3.6 4.4 4.1 4.8 4.3 5.  3.8 3.7 5.1 3.  6.  5.9 5.6 5.8 6.6 6.3 6.1 5.3
 5.5 6.7 6.9 5.7 6.4 5.4 5.2]

Petal_Width:
 [0.2 0.4 0.3 0.1 0.5 0.6 1.4 1.5 1.3 1.6 1.  1.1 1.8 1.2 1.7 2.5 1.9 2.1
 2.2 2.  2.4 2.3]

Class:
 ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

In [27]:
```
# to check the statistical data of the dataset.
# it only returns the statistical values of numeric cols.
```

```
df.describe()
```

Out[27]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [28]:
```
df.describe(include='all')
```

Out[28]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-setosa
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

In [30]:
```
# returns the no.of datas present for each category.
df['Class'].value_counts()
```

Out[30]:
```
Class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

Auto_MPG dataset:-

In [31]:
```
# importing the libraries:-
import numpy as np
import pandas as pd
```

In [32]:
```
# Reading the dataset
auto = pd.read_csv(r"C:\Users\CTTC\Downloads\archive\auto-mpg.csv")
```

In [33]:
```
auto
```

Out[33]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	r
0	18.0	8	307.0	130	3504	12.0	70	1	chev che m
1	15.0	8	350.0	165	3693	11.5	70	1	sk
2	18.0	8	318.0	150	3436	11.0	70	1	plym sat
3	16.0	8	304.0	150	3433	12.0	70	1	reb
4	17.0	8	302.0	140	3449	10.5	70	1	t
...	...	...	...	...	...	...	...	...	
393	27.0	4	140.0	86	2790	15.6	82	1	mus
394	44.0	4	97.0	52	2130	24.6	82	2	p
395	32.0	4	135.0	84	2295	11.6	82	1	d ram
396	28.0	4	120.0	79	2625	18.6	82	1	ra
397	31.0	4	119.0	82	2720	19.4	82	1	che

398 rows × 9 columns

In [34]:
```
# to check the null values:
auto.isnull().sum()
```

Out[34]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

In [35]:
```
# to check the datatype:
auto.dtypes
```

Out[35]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

In [36]:
```
auto.head()
```

Out[36]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	nar
0	18.0	8	307.0	130	3504	12.0	70	1	chevro cheve mali
1	15.0	8	350.0	165	3693	11.5	70	1	bu skyl 3
2	18.0	8	318.0	150	3436	11.0	70	1	plymou satell
3	16.0	8	304.0	150	3433	12.0	70	1	a rebel
4	17.0	8	302.0	140	3449	10.5	70	1	fo tori

In [37]:
```
# find the unique values:-
for i in auto.columns:
    print(f"{i}:\n {auto[i].unique()}\n")
```

```
mpg:
 [18.  15.  16.  17.  14.  24.  22.  21.  27.  26.  25.  10.  11.   9.
 28.  19.  12.  13.  23.  30.  31.  35.  20.  29.  32.  33.  17.5 15.5
 14.5 22.5 24.5 18.5 29.5 26.5 16.5 31.5 36.  25.5 33.5 20.5 30.5 21.5
 43.1 36.1 32.8 39.4 19.9 19.4 20.2 19.2 25.1 20.6 20.8 18.6 18.1 17.7
 27.5 27.2 30.9 21.1 23.2 23.8 23.9 20.3 21.6 16.2 19.8 22.3 17.6 18.2
 16.9 31.9 34.1 35.7 27.4 25.4 34.2 34.5 31.8 37.3 28.4 28.8 26.8 41.5
 38.1 32.1 37.2 26.4 24.3 19.1 34.3 29.8 31.3 37.  32.2 46.6 27.9 40.8
 44.3 43.4 36.4 44.6 40.9 33.8 32.7 23.7 23.6 32.4 26.6 25.8 23.5 39.1
 39.  35.1 32.3 37.7 34.7 34.4 29.9 33.7 32.9 31.6 28.1 30.7 24.2 22.4
 34.  38.  44. ]

cylinders:
 [8 4 6 3 5]

displacement:
 [307.  350.  318.  304.  302.  429.  454.  440.  455.  390.  383.  340.
 400.  113.  198.  199.  200.   97.  110.  107.  104.  121.  360.  140.
  98.  232.  225.  250.  351.  258.  122.  116.   79.   88.   71.   72.
  91.   97.5  70.  120.   96.  108.  155.   68.  114.  156.   76.   83.
  90.  231.  262.  134.  119.  171.  115.  101.  305.   85.  130.  168.
 111.  260.  151.  146.   80.   78.  105.  131.  163.   89.  267.   86.
 183.  141.  173.  135.   81.  100.  145.  112.  181.  144. ]

horsepower:
 ['130' '165' '150' '140' '198' '220' '215' '225' '190' '170' '160' '95'
 '97' '85' '88' '46' '87' '90' '113' '200' '210' '193' '?' '100' '105'
 '175' '153' '180' '110' '72' '86' '70' '76' '65' '69' '60' '80' '54'
 '208' '155' '112' '92' '145' '137' '158' '167' '94' '107' '230' '49' '75'
 '91' '122' '67' '83' '78' '52' '61' '93' '148' '129' '96' '71' '98' '115'
 '53' '81' '79' '120' '152' '102' '108' '68' '58' '149' '89' '63' '48'
 '66' '139' '103' '125' '133' '138' '135' '142' '77' '62' '132' '84' '64'
 '74' '116' '82']

weight:
 [3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 3563 3609 3761 3086
 2372 2833 2774 2587 2130 1835 2672 2430 2375 2234 2648 4615 4376 4382
 4732 2264 2228 2046 2634 3439 3329 3302 3288 4209 4464 4154 4096 4955
 4746 5140 2962 2408 3282 3139 2220 2123 2074 2065 1773 1613 1834 1955
 2278 2126 2254 2226 4274 4385 4135 4129 3672 4633 4502 4456 4422 2330
 3892 4098 4294 4077 2933 2511 2979 2189 2395 2288 2506 2164 2100 4100
 3988 4042 3777 4952 4363 4237 4735 4951 3821 3121 3278 2945 3021 2904
 1950 4997 4906 4654 4499 2789 2279 2401 2379 2124 2310 2472 2265 4082
 4278 1867 2158 2582 2868 3399 2660 2807 3664 3102 2875 2901 3336 2451
 1836 2542 3781 3632 3613 4141 4699 4457 4638 4257 2219 1963 2300 1649
 2003 2125 2108 2246 2489 2391 2000 3264 3459 3432 3158 4668 4440 4498
 4657 3907 3897 3730 3785 3039 3221 3169 2171 2639 2914 2592 2702 2223
 2545 2984 1937 3211 2694 2957 2671 1795 2464 2572 2255 2202 4215 4190
 3962 3233 3353 3012 3085 2035 3651 3574 3645 3193 1825 1990 2155 2565
 3150 3940 3270 2930 3820 4380 4055 3870 3755 2045 1945 3880 4060 4140
 4295 3520 3425 3630 3525 4220 4165 4325 4335 1940 2740 2755 2051 2075
 1985 2190 2815 2600 2720 1800 2070 3365 3735 3570 3535 3155 2965 3430
 3210 3380 3070 3620 3410 3445 3205 4080 2560 2230 2515 2745 2855 2405
 2830 3140 2795 2135 3245 2990 2890 3265 3360 3840 3725 3955 3830 4360
 4054 3605 1925 1975 1915 2670 3530 3900 3190 3420 2200 2150 2020 2595
 2700 2556 2144 1968 2120 2019 2678 2870 3003 3381 2188 2711 2434 2110
```

```
 2800 2085 2335 2950 3250 1850 2145 1845 2910 2420 2500 2905 2290 2490
 2635 2620 2725 2385 1755 1875 1760 2050 2215 2380 2320 2210 2350 2615
 3230 3160 2900 3415 3060 3465 2605 2640 2575 2525 2735 2865 3035 1980
 2025 1970 2160 2205 2245 1965 1995 3015 2585 2835 2665 2370 2790 2295
 2625]

acceleration:
 [12.  11.5 11.  10.5 10.   9.   8.5  8.   9.5 15.  15.5 16.  14.5 20.5
 17.5 12.5 14.  13.5 18.5 19.  13.  19.5 18.  17.  23.5 16.5 21.  16.9
 14.9 17.7 15.3 13.9 12.8 15.4 17.6 22.2 22.1 14.2 17.4 16.2 17.8 12.2
 16.4 13.6 15.7 13.2 21.9 16.7 12.1 14.8 18.6 16.8 13.7 11.1 11.4 18.2
 15.8 15.9 14.1 21.5 14.4 19.4 19.2 17.2 18.7 15.1 13.4 11.2 14.7 16.6
 17.3 15.2 14.3 20.1 24.8 11.3 12.9 18.8 18.1 17.9 21.7 23.7 19.9 21.8
 13.8 12.6 16.1 20.7 18.3 20.4 19.6 17.1 15.6 24.6 11.6]

model year:
 [70 71 72 73 74 75 76 77 78 79 80 81 82]

origin:
 [1 3 2]

car name:
 ['chevrolet chevelle malibu' 'buick skylark 320' 'plymouth satellite'
 'amc rebel sst' 'ford torino' 'ford galaxie 500' 'chevrolet impala'
 'plymouth fury iii' 'pontiac catalina' 'amc ambassador dpl'
 'dodge challenger se' "plymouth 'cuda 340" 'chevrolet monte carlo'
 'buick estate wagon (sw)' 'toyota corona mark ii' 'plymouth duster'
 'amc hornet' 'ford maverick' 'datsun pl510'
 'volkswagen 1131 deluxe sedan' 'peugeot 504' 'audi 100 ls' 'saab 99e'
 'bmw 2002' 'amc gremlin' 'ford f250' 'chevy c20' 'dodge d200' 'hi 1200d'
 'chevrolet vega 2300' 'toyota corona' 'ford pinto'
 'plymouth satellite custom' 'ford torino 500' 'amc matador'
 'pontiac catalina brougham' 'dodge monaco (sw)'
 'ford country squire (sw)' 'pontiac safari (sw)'
 'amc hornet sportabout (sw)' 'chevrolet vega (sw)' 'pontiac firebird'
 'ford mustang' 'mercury capri 2000' 'opel 1900' 'peugeot 304' 'fiat 124b'
 'toyota corolla 1200' 'datsun 1200' 'volkswagen model 111'
 'plymouth cricket' 'toyota corona hardtop' 'dodge colt hardtop'
 'volkswagen type 3' 'chevrolet vega' 'ford pinto runabout'
 'amc ambassador sst' 'mercury marquis' 'buick lesabre custom'
 'oldsmobile delta 88 royale' 'chrysler newport royal' 'mazda rx2 coupe'
 'amc matador (sw)' 'chevrolet chevelle concours (sw)'
 'ford gran torino (sw)' 'plymouth satellite custom (sw)'
 'volvo 145e (sw)' 'volkswagen 411 (sw)' 'peugeot 504 (sw)'
 'renault 12 (sw)' 'ford pinto (sw)' 'datsun 510 (sw)'
 'toyouta corona mark ii (sw)' 'dodge colt (sw)'
 'toyota corolla 1600 (sw)' 'buick century 350' 'chevrolet malibu'
 'ford gran torino' 'dodge coronet custom' 'mercury marquis brougham'
 'chevrolet caprice classic' 'ford ltd' 'plymouth fury gran sedan'
 'chrysler new yorker brougham' 'buick electra 225 custom'
 'amc ambassador brougham' 'plymouth valiant' 'chevrolet nova custom'
 'volkswagen super beetle' 'ford country' 'plymouth custom suburb'
 'oldsmobile vista cruiser' 'toyota carina' 'datsun 610' 'maxda rx3'
 'mercury capri v6' 'fiat 124 sport coupe' 'chevrolet monte carlo s'
 'pontiac grand prix' 'fiat 128' 'opel manta' 'audi 100ls' 'volvo 144ea'
 'dodge dart custom' 'saab 99le' 'toyota mark ii' 'oldsmobile omega'
```

```
'chevrolet nova' 'datsun b210' 'chevrolet chevelle malibu classic'
'plymouth satellite sebring' 'buick century luxus (sw)'
'dodge coronet custom (sw)' 'audi fox' 'volkswagen dasher' 'datsun 710'
'dodge colt' 'fiat 124 tc' 'honda civic' 'subaru' 'fiat x1.9'
'plymouth valiant custom' 'mercury monarch' 'chevrolet bel air'
'plymouth grand fury' 'buick century' 'chevroelt chevelle malibu'
'plymouth fury' 'buick skyhawk' 'chevrolet monza 2+2' 'ford mustang ii'
'toyota corolla' 'pontiac astro' 'volkswagen rabbit' 'amc pacer'
'volvo 244dl' 'honda civic cvcc' 'fiat 131' 'capri ii' 'renault 12tl'
'dodge coronet brougham' 'chevrolet chevette' 'chevrolet woody'
'vw rabbit' 'dodge aspen se' 'ford granada ghia' 'pontiac ventura sj'
'amc pacer d/l' 'datsun b-210' 'volvo 245' 'plymouth volare premier v8'
'mercedes-benz 280s' 'cadillac seville' 'chevy c10' 'ford f108'
'dodge d100' 'honda accord cvcc' 'buick opel isuzu deluxe'
'renault 5 gtl' 'plymouth arrow gs' 'datsun f-10 hatchback'
'oldsmobile cutlass supreme' 'dodge monaco brougham'
'mercury cougar brougham' 'chevrolet concours' 'buick skylark'
'plymouth volare custom' 'ford granada' 'pontiac grand prix lj'
'chevrolet monte carlo landau' 'chrysler cordoba' 'ford thunderbird'
'volkswagen rabbit custom' 'pontiac sunbird coupe'
'toyota corolla liftback' 'ford mustang ii 2+2' 'dodge colt m/m'
'subaru dl' 'datsun 810' 'bmw 320i' 'mazda rx-4'
'volkswagen rabbit custom diesel' 'ford fiesta' 'mazda glc deluxe'
'datsun b210 gx' 'oldsmobile cutlass salon brougham' 'dodge diplomat'
'mercury monarch ghia' 'pontiac phoenix lj' 'ford fairmont (auto)'
'ford fairmont (man)' 'plymouth volare' 'amc concord'
'buick century special' 'mercury zephyr' 'dodge aspen' 'amc concord d/l'
'buick regal sport coupe (turbo)' 'ford futura' 'dodge magnum xe'
'datsun 510' 'dodge omni' 'toyota celica gt liftback' 'plymouth sapporo'
'oldsmobile starfire sx' 'datsun 200-sx' 'audi 5000' 'volvo 264gl'
'saab 99gle' 'peugeot 604sl' 'volkswagen scirocco' 'honda accord lx'
'pontiac lemans v6' 'mercury zephyr 6' 'ford fairmont 4'
'amc concord dl 6' 'dodge aspen 6' 'ford ltd landau'
'mercury grand marquis' 'dodge st. regis' 'chevrolet malibu classic (sw)'
'chrysler lebaron town @ country (sw)' 'vw rabbit custom'
'maxda glc deluxe' 'dodge colt hatchback custom' 'amc spirit dl'
'mercedes benz 300d' 'cadillac eldorado' 'plymouth horizon'
'plymouth horizon tc3' 'datsun 210' 'fiat strada custom'
'buick skylark limited' 'chevrolet citation' 'oldsmobile omega brougham'
'pontiac phoenix' 'toyota corolla tercel' 'datsun 310' 'ford fairmont'
'audi 4000' 'toyota corona liftback' 'mazda 626' 'datsun 510 hatchback'
'mazda glc' 'vw rabbit c (diesel)' 'vw dasher (diesel)'
'audi 5000s (diesel)' 'mercedes-benz 240d' 'honda civic 1500 gl'
'renault lecar deluxe' 'vokswagen rabbit' 'datsun 280-zx' 'mazda rx-7 gs'
'triumph tr7 coupe' 'ford mustang cobra' 'honda accord'
'plymouth reliant' 'dodge aries wagon (sw)' 'toyota starlet'
'plymouth champ' 'honda civic 1300' 'datsun 210 mpg' 'toyota tercel'
'mazda glc 4' 'plymouth horizon 4' 'ford escort 4w' 'ford escort 2h'
'volkswagen jetta' 'renault 18i' 'honda prelude' 'datsun 200sx'
'peugeot 505s turbo diesel' 'volvo diesel' 'toyota cressida'
'datsun 810 maxima' 'oldsmobile cutlass ls' 'ford granada gl'
'chrysler lebaron salon' 'chevrolet cavalier' 'chevrolet cavalier wagon'
'chevrolet cavalier 2-door' 'pontiac j2000 se hatchback' 'dodge aries se'
'ford fairmont futura' 'amc concord dl' 'volkswagen rabbit l'
'mazda glc custom l' 'mazda glc custom' 'plymouth horizon miser'
'mercury lynx l' 'nissan stanza xe' 'honda civic (auto)' 'datsun 310 gx'
```

```
'buick century limited' 'oldsmobile cutlass ciera (diesel)'
'chrysler lebaron medallion' 'ford granada l' 'toyota celica gt'
'dodge charger 2.2' 'chevrolet camaro' 'ford mustang gl' 'vw pickup'
'dodge rampage' 'ford ranger' 'chevy s-10']
```

In [38]:
```
# to find the total no.of missing values (?) in dataset.
sum(auto['mpg']=='?')
```

Out[38]: 0

In [39]:
```
for i in auto.columns:
    print(f"{i} : {sum(auto[i]=='?')}")
```

```
mpg : 0
cylinders : 0
displacement : 0
horsepower : 6
weight : 0
acceleration : 0
model year : 0
origin : 0
car name : 0
```

In [40]:
```
des = auto.describe(include='all')
des
```

Out[40]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	
count	398.000000	398.000000	398.000000	398	398.000000	398.000000	398.(
unique	NaN	NaN	NaN	94	NaN	NaN	
top	NaN	NaN	NaN	150	NaN	NaN	
freq	NaN	NaN	NaN	22	NaN	NaN	
mean	23.514573	5.454774	193.425879	NaN	2970.424623	15.568090	76.(
std	7.815984	1.701004	104.269838	NaN	846.841774	2.757689	3.(
min	9.000000	3.000000	68.000000	NaN	1613.000000	8.000000	70.(
25%	17.500000	4.000000	104.250000	NaN	2223.750000	13.825000	73.(
50%	23.000000	4.000000	148.500000	NaN	2803.500000	15.500000	76.(
75%	29.000000	8.000000	262.000000	NaN	3608.000000	17.175000	79.(
max	46.600000	8.000000	455.000000	NaN	5140.000000	24.800000	82.(

In [41]:
```
# replacing the '?' with top repeated value.
auto['horsepower'].replace('?',des['horsepower'][2])
```

```
C:\Users\CTTC\AppData\Local\Temp\ipykernel_4964\4048010339.py:2: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  auto['horsepower'].replace('?',des['horsepower'][2])
```

Out[41]:
```
0      130
1      165
2      150
3      150
4      140
      ...
393     86
394     52
395     84
396     79
397     82
Name: horsepower, Length: 398, dtype: object
```

In [42]:
```
for i in auto.columns:
    print(f"{i} : {sum(auto[i]=='?')}")
```

```
mpg : 0
cylinders : 0
displacement : 0
horsepower : 6
weight : 0
acceleration : 0
model year : 0
origin : 0
car name : 0
```

In [43]:
```
# replacing the '?' with top repeated value.
auto['horsepower'].replace('?',des['horsepower'][2],inplace=True)
```

```
C:\Users\CTTC\AppData\Local\Temp\ipykernel_4964\4201554134.py:2: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels (consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  auto['horsepower'].replace('?',des['horsepower'][2],inplace=True)
```

In [44]:
```
for i in auto.columns:
    print(f"{i} : {sum(auto[i]=='?')}")
```

```
mpg : 0
cylinders : 0
displacement : 0
horsepower : 0
weight : 0
acceleration : 0
model year : 0
origin : 0
car name : 0
```

In [45]:
```
auto.dtypes
```

Out[45]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

In [46]:
```
# to change the datatype of 'horsepower' explicitly.
auto.horsepower = auto.horsepower.astype('float')
auto.dtypes
```

Out[46]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight            int64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

In [47]:
```
auto.describe(include='all')
```

Out[47]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.(
unique	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	NaN	
mean	23.514573	5.454774	193.425879	105.155779	2970.424623	15.568090	76.(
std	7.815984	1.701004	104.269838	38.600986	846.841774	2.757689	3.(
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.(
25%	17.500000	4.000000	104.250000	76.000000	2223.750000	13.825000	73.(
50%	23.000000	4.000000	148.500000	95.000000	2803.500000	15.500000	76.(
75%	29.000000	8.000000	262.000000	130.000000	3608.000000	17.175000	79.(
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.(

Wine Quality Dataset:-

In [48]:
```
import numpy as np
import pandas as pd
```

In [49]:
```
wine = pd.read_csv(r"C:\Users\CTTC\Downloads\wine+quality\winequality-red.csv")
wine
```

Out[49]:

	fixed acidity;"volatile acidity";"citric acid";"residual sugar";"chlorides";"free sulfur dioxide";"total sulfur dioxide";"density";"pH";"sulphates";"alcohol";"quality"
0	7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
1	7.8;0.88;0;2.6;0.098;25;67;0.9968;3.2;0.68;9.8;5
2	7.8;0.76;0.04;2.3;0.092;15;54;0.997;3.26;0.65;...
3	11.2;0.28;0.56;1.9;0.075;17;60;0.998;3.16;0.58...
4	7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
...	...
1594	6.2;0.6;0.08;2;0.09;32;44;0.9949;3.45;0.58;10.5;5
1595	5.9;0.55;0.1;2.2;0.062;39;51;0.99512;3.52;0.76...
1596	6.3;0.51;0.13;2.3;0.076;29;40;0.99574;3.42;0.7...
1597	5.9;0.645;0.12;2;0.075;32;44;0.99547;3.57;0.71...
1598	6;0.31;0.47;3.6;0.067;18;42;0.99549;3.39;0.66;...

1599 rows × 1 columns

To separate each columns along with datas from semicolon ';'

In [50]:
```
wine = pd.read_csv(r"C:\Users\CTTC\Downloads\wine+quality\winequality-red.csv",deli
wine
```

Out[50]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphate
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.6
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.6
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.5
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.5
...	...	...	...	...	...	...	...	...	...	.
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.7
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.7
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.7
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.6

1599 rows × 12 columns

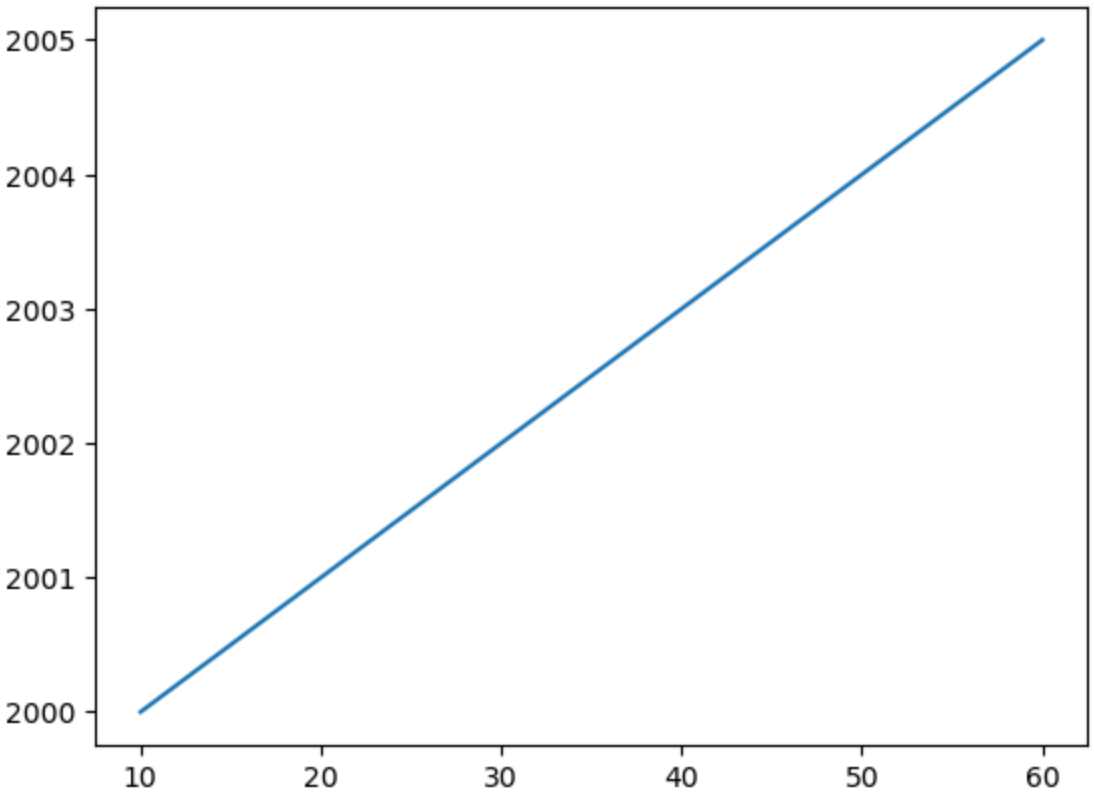
Matplotlib :- It is Data Visualization Tool.

Used to plot different graphs.

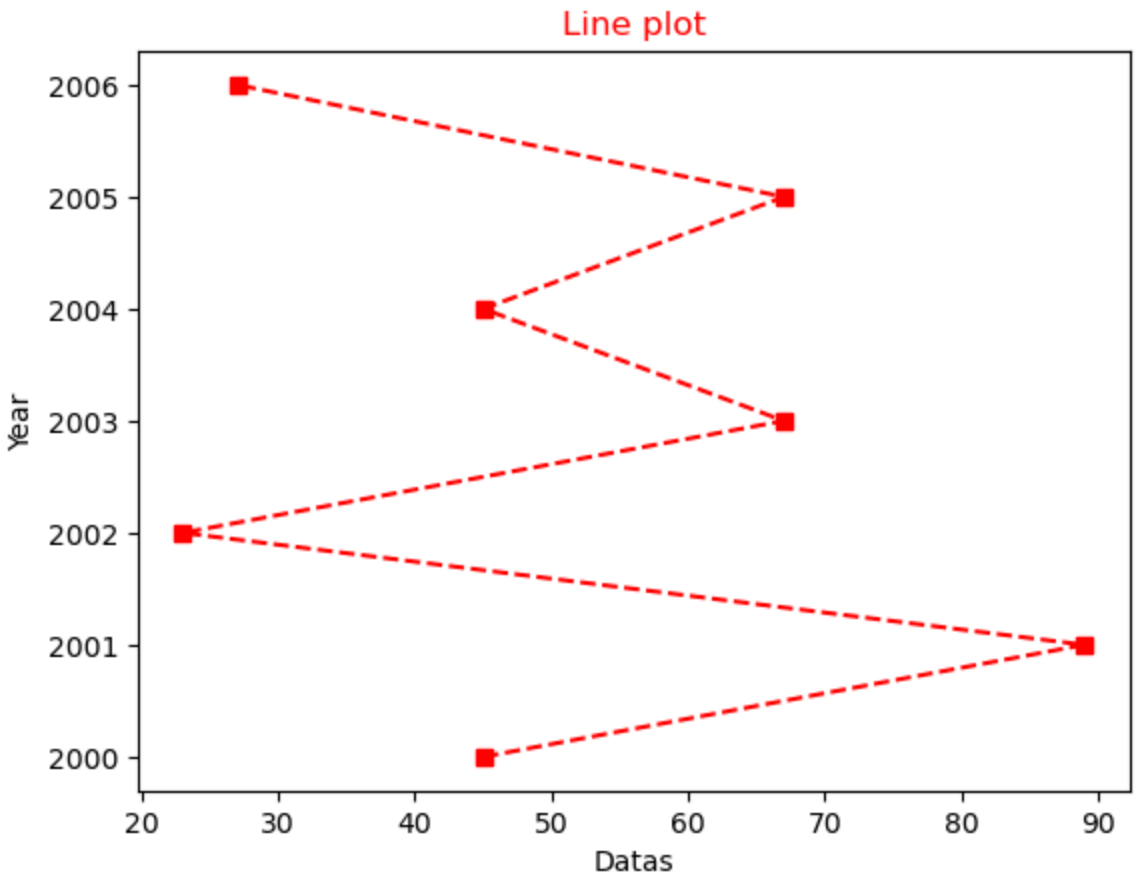
In [51]:
```
# Import the libraries:-
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Line Plot :- is used to visualize a particular trend over time.

In [52]:
```
x = [10,20,30,40,50,60]
y = [2000,2001,2002,2003,2004,2005]
plt.plot(x,y) # to the graph using x & y data.
plt.show()  # to plot the graph
```

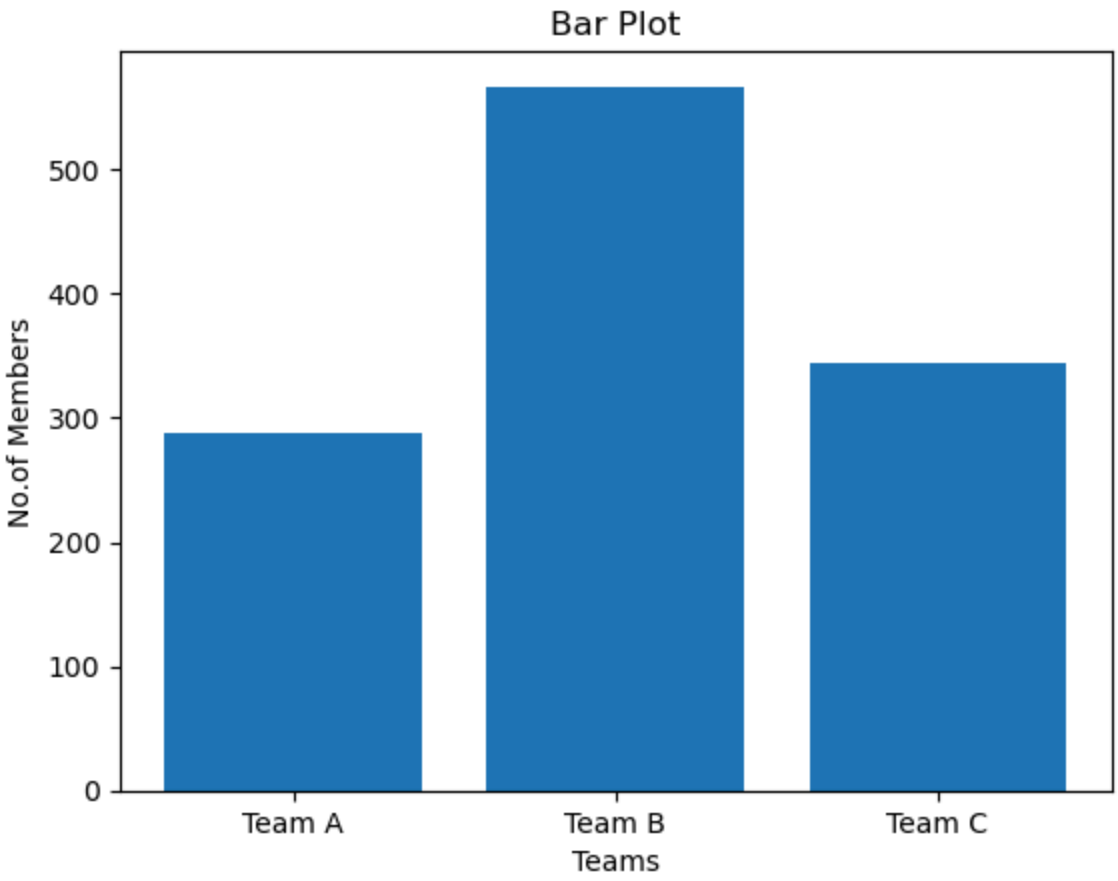


In [61]:
```
x = [45,89,23,67,45,67,27]
y = [2000,2001,2002,2003,2004,2005,2006]
plt.title('Line plot',c='red') # gives a heading to the graph
plt.plot(x,y,marker='s',linestyle='--',c='r')
plt.xlabel('Datas') # gives a name to x-axis
plt.ylabel('Year') # gives a name to y-axis
plt.show()
```

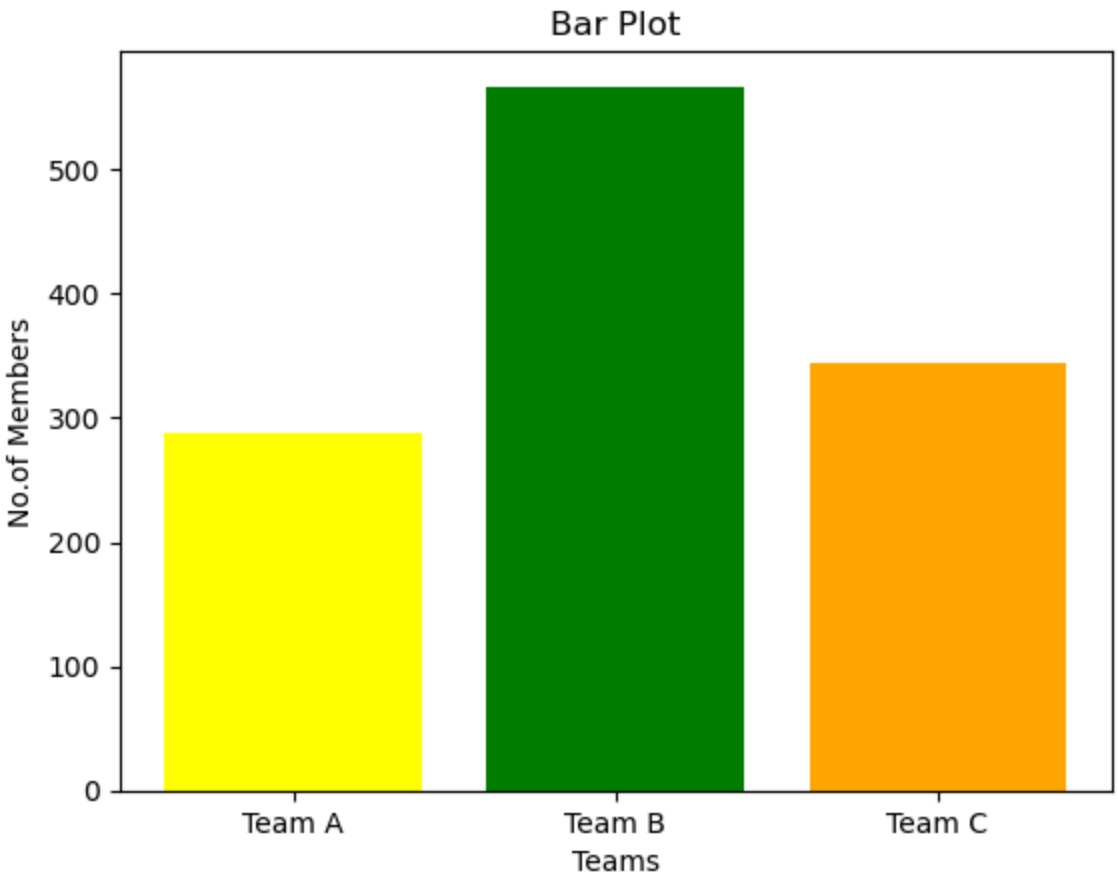


Bar plot:- used for comparision of categories.

In [62]:
```
x = ['Team A','Team B','Team C']
y = [288,567,345]
plt.bar(x,y) # plot a bar graph
plt.title('Bar Plot')
plt.xlabel('Teams')
plt.ylabel('No.of Members')
plt.show()
```



In [63]:
```
x = ['Team A','Team B','Team C']
y = [288,567,345]
colours = ['yellow','green','orange']
plt.bar(x,y,color = colours ) # plot a bar graph
plt.title('Bar Plot')
plt.xlabel('Teams')
plt.ylabel('No.of Members')
plt.show()
```


Bar Plot
No.of Members
0
100
200
300
400
500
Team A
Team B
Team C
Teams

In [ ]: