Name: Saloni Vishwakarma

Roll No: C1-13

Aim: To implement a Machine Learning Regression model using a Simple Linear regression algorithm

Import Libraries

In [1]:
```
import pandas as pd
import pandas as pd
from matplotlib import pyplot as plt
```

Import Dataset

In [2]:
```
url  = "https://raw.githubusercontent.com/devzohaib/Simple-Linear-Regression/master
d1 = pd.read_csv(url)
```

In [3]:
```
print(d1)
```

```
        TV  Sales
0    230.1   22.1
1     44.5   10.4
2     17.2    9.3
3    151.5   18.5
4    180.8   12.9
..     ...    ...
195   38.2    7.6
196   94.2    9.7
197  177.0   12.8
198  283.6   25.5
199  232.1   13.4

[200 rows x 2 columns]
```

In [4]:
```
d1.head(10)
```

Out[4]:

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	9.3
3	151.5	18.5
4	180.8	12.9
5	8.7	7.2
6	57.5	11.8
7	120.2	13.2
8	8.6	4.8
9	199.8	10.6

Exploratory Data Analysis(EDA)

In [5]:
```
d1.head()
```

Out[5]:

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	9.3
3	151.5	18.5
4	180.8	12.9

In [6]:
```
d1.tail()
```

Out[6]:

	TV	Sales
195	38.2	7.6
196	94.2	9.7
197	177.0	12.8
198	283.6	25.5
199	232.1	13.4

In [7]:
```
d1.describe()
```

Out[7]:

	TV	Sales
count	200.000000	200.000000
mean	147.042500	14.022500
std	85.854236	5.217457
min	0.700000	1.600000
25%	74.375000	10.375000
50%	149.750000	12.900000
75%	218.825000	17.400000
max	296.400000	27.000000

In [8]:

```
d1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   TV      200 non-null    float64
 1   Sales   200 non-null    float64
dtypes: float64(2)
memory usage: 3.3 KB
```

In [9]:

```
print(d1.shape)
```

```
(200, 2)
```

In [10]:

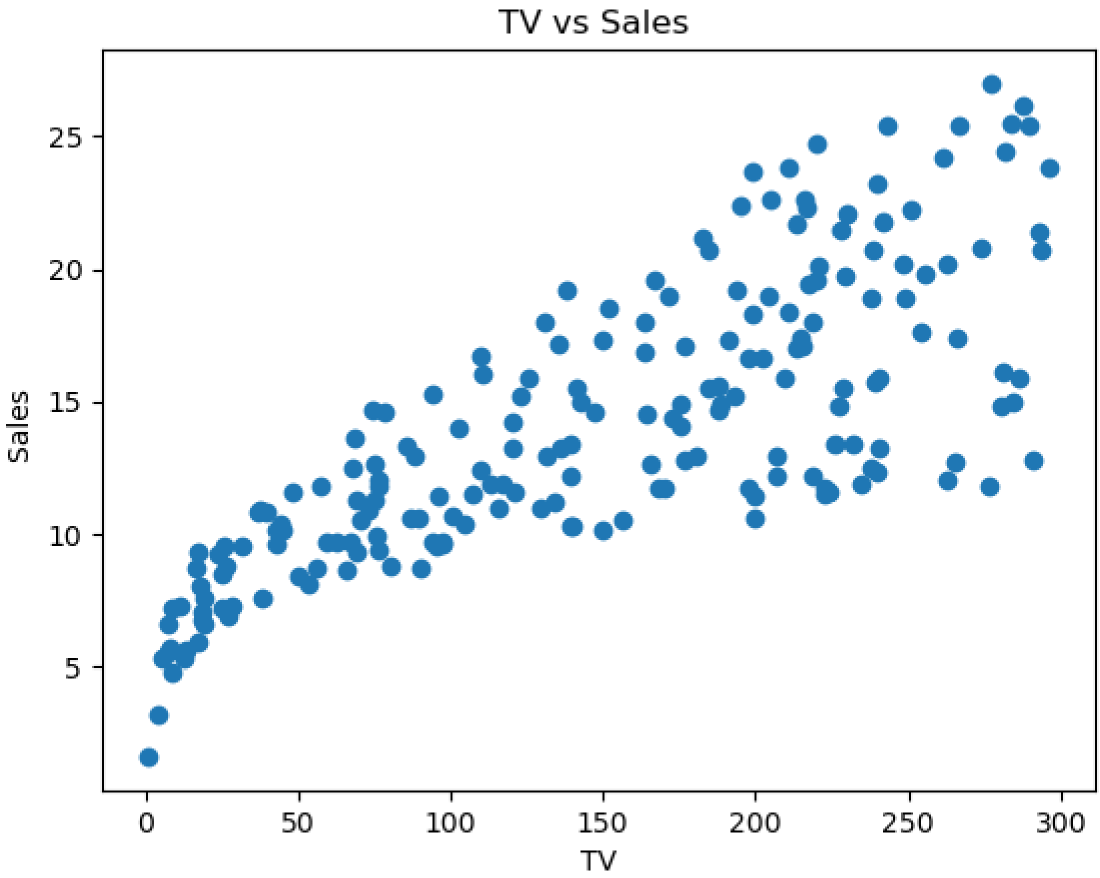
```
d1.corr()
```

Out[10]:

	TV	Sales
TV	1.000000	0.782224
Sales	0.782224	1.000000

In [11]:

```
from matplotlib import pyplot as plt
plt.scatter(d1['TV'],d1["Sales"])
plt.title('TV vs Sales ')
plt.xlabel("TV")
plt.ylabel('Sales ')
plt.show()
```



Select attribute as dependent and independent

In [12]:
```
d1.head()
```

Out[12]:

	TV	Sales
0	230.1	22.1
1	44.5	10.4
2	17.2	9.3
3	151.5	18.5
4	180.8	12.9

In [13]:
```
x=d1.iloc[:,:1].values
y=d1.iloc[:,1].values
```

In [14]:
```
print(x)
```

```
[[230.1]
 [ 44.5]
 [ 17.2]
 [151.5]
 [180.8]
 [  8.7]
 [ 57.5]
 [120.2]
 [  8.6]
 [199.8]
 [ 66.1]
 [214.7]
 [ 23.8]
 [ 97.5]
 [204.1]
 [195.4]
 [ 67.8]
 [281.4]
 [ 69.2]
 [147.3]
 [218.4]
 [237.4]
 [ 13.2]
 [228.3]
 [ 62.3]
 [262.9]
 [142.9]
 [240.1]
 [248.8]
 [ 70.6]
 [292.9]
 [112.9]
 [ 97.2]
 [265.6]
 [ 95.7]
 [290.7]
 [266.9]
 [ 74.7]
 [ 43.1]
 [228. ]
 [202.5]
 [177. ]
 [293.6]
 [206.9]
 [ 25.1]
 [175.1]
 [ 89.7]
 [239.9]
 [227.2]
 [ 66.9]
 [199.8]
 [100.4]
 [216.4]
 [182.6]
 [262.7]
 [198.9]
 [  7.3]
 [136.2]
 [210.8]
 [210.7]
 [ 53.5]
 [261.3]
 [239.3]
 [102.7]
```

```
[131.1]
[ 69. ]
[ 31.5]
[139.3]
[237.4]
[216.8]
[199.1]
[109.8]
[ 26.8]
[129.4]
[213.4]
[ 16.9]
[ 27.5]
[120.5]
[  5.4]
[116. ]
[ 76.4]
[239.8]
[ 75.3]
[ 68.4]
[213.5]
[193.2]
[ 76.3]
[110.7]
[ 88.3]
[109.8]
[134.3]
[ 28.6]
[217.7]
[250.9]
[107.4]
[163.3]
[197.6]
[184.9]
[289.7]
[135.2]
[222.4]
[296.4]
[280.2]
[187.9]
[238.2]
[137.9]
[ 25. ]
[ 90.4]
[ 13.1]
[255.4]
[225.8]
[241.7]
[175.7]
[209.6]
[ 78.2]
[ 75.1]
[139.2]
[ 76.4]
[125.7]
[ 19.4]
[141.3]
[ 18.8]
[224. ]
[123.1]
[229.5]
[ 87.2]
[  7.8]
[ 80.2]
```

```
[220.3]
[ 59.6]
[  0.7]
[265.2]
[  8.4]
[219.8]
[ 36.9]
[ 48.3]
[ 25.6]
[273.7]
[ 43. ]
[184.9]
[ 73.4]
[193.7]
[220.5]
[104.6]
[ 96.2]
[140.3]
[240.1]
[243.2]
[ 38. ]
[ 44.7]
[280.7]
[121. ]
[197.6]
[171.3]
[187.8]
[  4.1]
[ 93.9]
[149.8]
[ 11.7]
[131.7]
[172.5]
[ 85.7]
[188.4]
[163.5]
[117.2]
[234.5]
[ 17.9]
[206.8]
[215.4]
[284.3]
[ 50. ]
[164.5]
[ 19.6]
[168.4]
[222.4]
[276.9]
[248.4]
[170.2]
[276.7]
[165.6]
[156.6]
[218.5]
[ 56.2]
[287.6]
[253.8]
[205. ]
[139.5]
[191.1]
[286. ]
[ 18.7]
[ 39.5]
[ 75.5]
```

```
 [ 17.2]
 [166.8]
 [149.7]
 [ 38.2]
 [ 94.2]
 [177. ]
 [283.6]
 [232.1]]
```

In [15]:
```
print(y)
```

```
[22.1 10.4  9.3 18.5 12.9  7.2 11.8 13.2  4.8 10.6  8.6 17.4  9.2  9.7
 19.  22.4 12.5 24.4 11.3 14.6 18.  12.5  5.6 15.5  9.7 12.  15.  15.9
 18.9 10.5 21.4 11.9  9.6 17.4  9.5 12.8 25.4 14.7 10.1 21.5 16.6 17.1
 20.7 12.9  8.5 14.9 10.6 23.2 14.8  9.7 11.4 10.7 22.6 21.2 20.2 23.7
  5.5 13.2 23.8 18.4  8.1 24.2 15.7 14.  18.   9.3  9.5 13.4 18.9 22.3
 18.3 12.4  8.8 11.  17.   8.7  6.9 14.2  5.3 11.  11.8 12.3 11.3 13.6
 21.7 15.2 12.  16.  12.9 16.7 11.2  7.3 19.4 22.2 11.5 16.9 11.7 15.5
 25.4 17.2 11.7 23.8 14.8 14.7 20.7 19.2  7.2  8.7  5.3 19.8 13.4 21.8
 14.1 15.9 14.6 12.6 12.2  9.4 15.9  6.6 15.5  7.  11.6 15.2 19.7 10.6
  6.6  8.8 24.7  9.7  1.6 12.7  5.7 19.6 10.8 11.6  9.5 20.8  9.6 20.7
 10.9 19.2 20.1 10.4 11.4 10.3 13.2 25.4 10.9 10.1 16.1 11.6 16.6 19.
 15.6  3.2 15.3 10.1  7.3 12.9 14.4 13.3 14.9 18.  11.9 11.9  8.  12.2
 17.1 15.   8.4 14.5  7.6 11.7 11.5 27.  20.2 11.7 11.8 12.6 10.5 12.2
  8.7 26.2 17.6 22.6 10.3 17.3 15.9  6.7 10.8  9.9  5.9 19.6 17.3  7.6
  9.7 12.8 25.5 13.4]
```

Splitting Data into Training and Testing Dataset

In [16]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [17]:
```
print("x_train")
print(x_train)
print("x_test")
print(x_test)
```

```
x_train
[[ 36.9]
 [ 31.5]
 [142.9]
 [209.6]
 [215.4]
 [102.7]
 [  8.6]
 [ 16.9]
 [125.7]
 [104.6]
 [109.8]
 [229.5]
 [253.8]
 [184.9]
 [ 44.7]
 [ 62.3]
 [292.9]
 [172.5]
 [202.5]
 [  7.3]
 [265.2]
 [197.6]
 [218.5]
 [147.3]
 [171.3]
 [217.7]
 [262.7]
 [163.5]
 [100.4]
 [ 76.3]
 [184.9]
 [134.3]
 [273.7]
 [296.4]
 [ 96.2]
 [109.8]
 [255.4]
 [204.1]
 [240.1]
 [193.7]
 [191.1]
 [ 89.7]
 [ 43. ]
 [ 38.2]
 [ 13.1]
 [239.3]
 [ 17.2]
 [210.7]
 [ 25.6]
 [177. ]
 [206.9]
 [ 66.1]
 [149.7]
 [129.4]
 [ 94.2]
 [276.7]
 [276.9]
 [  7.8]
 [250.9]
 [175.7]
 [ 11.7]
 [ 75.5]
 [199.8]
```

```
[230.1]
[107.4]
[225.8]
[163.3]
[131.1]
[206.8]
[177. ]
[216.8]
[ 66.9]
[227.2]
[193.2]
[ 97.5]
[ 85.7]
[228.3]
[139.5]
[ 48.3]
[218.4]
[195.4]
[  5.4]
[238.2]
[216.4]
[222.4]
[ 27.5]
[151.5]
[139.2]
[117.2]
[283.6]
[ 57.5]
[237.4]
[213.5]
[ 18.8]
[  4.1]
[164.5]
[ 93.9]
[ 28.6]
[232.1]
[214.7]
[ 19.4]
[280.2]
[290.7]
[136.2]
[ 69. ]
[ 44.5]
[141.3]
[188.4]
[293.6]
[137.9]
[  8.4]
[168.4]
[281.4]
[ 43.1]
[219.8]
[182.6]
[149.8]
[220.3]
[ 95.7]
[248.8]
[ 78.2]
[121. ]
[112.9]
[ 17.9]
[ 80.2]
[248.4]
[ 97.2]
```

```
 [220.5]
 [284.3]
 [243.2]
 [ 70.6]
 [135.2]
 [ 75.3]
 [116. ]
 [ 75.1]
 [ 38. ]
 [166.8]
 [ 26.8]
 [120.5]
 [262.9]
 [234.5]
 [239.8]
 [286. ]
 [222.4]
 [ 39.5]
 [228. ]
 [210.8]
 [ 73.4]
 [ 88.3]
 [199.1]
 [110.7]
 [266.9]
 [237.4]
 [199.8]
 [187.9]
 [139.3]
 [ 17.2]
 [ 76.4]
 [239.9]
 [ 19.6]]
x_test
[[ 69.2]
 [ 50. ]
 [ 90.4]
 [289.7]
 [170.2]
 [ 56.2]
 [  8.7]
 [240.1]
 [ 23.8]
 [197.6]
 [261.3]
 [ 87.2]
 [156.6]
 [187.8]
 [ 76.4]
 [120.2]
 [265.6]
 [  0.7]
 [ 74.7]
 [213.4]
 [287.6]
 [140.3]
 [175.1]
 [131.7]
 [ 53.5]
 [123.1]
 [165.6]
 [205. ]
 [224. ]
 [ 25.1]
```

```
 [ 67.8]
 [198.9]
 [280.7]
 [241.7]
 [ 13.2]
 [ 18.7]
 [ 59.6]
 [180.8]
 [ 68.4]
 [ 25. ]]
```

In [18]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(160, 1)
(40, 1)
(160,)
(40,)
```

In [19]:
```
print("X:",x)

print("x_train:",x_train)
print("x_test",x_test)
```

```
X: [[230.1]
 [ 44.5]
 [ 17.2]
 [151.5]
 [180.8]
 [  8.7]
 [ 57.5]
 [120.2]
 [  8.6]
 [199.8]
 [ 66.1]
 [214.7]
 [ 23.8]
 [ 97.5]
 [204.1]
 [195.4]
 [ 67.8]
 [281.4]
 [ 69.2]
 [147.3]
 [218.4]
 [237.4]
 [ 13.2]
 [228.3]
 [ 62.3]
 [262.9]
 [142.9]
 [240.1]
 [248.8]
 [ 70.6]
 [292.9]
 [112.9]
 [ 97.2]
 [265.6]
 [ 95.7]
 [290.7]
 [266.9]
 [ 74.7]
 [ 43.1]
 [228. ]
 [202.5]
 [177. ]
 [293.6]
 [206.9]
 [ 25.1]
 [175.1]
 [ 89.7]
 [239.9]
 [227.2]
 [ 66.9]
 [199.8]
 [100.4]
 [216.4]
 [182.6]
 [262.7]
 [198.9]
 [  7.3]
 [136.2]
 [210.8]
 [210.7]
 [ 53.5]
 [261.3]
 [239.3]
 [102.7]
```

```
[131.1]
[ 69. ]
[ 31.5]
[139.3]
[237.4]
[216.8]
[199.1]
[109.8]
[ 26.8]
[129.4]
[213.4]
[ 16.9]
[ 27.5]
[120.5]
[  5.4]
[116. ]
[ 76.4]
[239.8]
[ 75.3]
[ 68.4]
[213.5]
[193.2]
[ 76.3]
[110.7]
[ 88.3]
[109.8]
[134.3]
[ 28.6]
[217.7]
[250.9]
[107.4]
[163.3]
[197.6]
[184.9]
[289.7]
[135.2]
[222.4]
[296.4]
[280.2]
[187.9]
[238.2]
[137.9]
[ 25. ]
[ 90.4]
[ 13.1]
[255.4]
[225.8]
[241.7]
[175.7]
[209.6]
[ 78.2]
[ 75.1]
[139.2]
[ 76.4]
[125.7]
[ 19.4]
[141.3]
[ 18.8]
[224. ]
[123.1]
[229.5]
[ 87.2]
[  7.8]
[ 80.2]
```

```
[220.3]
[ 59.6]
[  0.7]
[265.2]
[  8.4]
[219.8]
[ 36.9]
[ 48.3]
[ 25.6]
[273.7]
[ 43. ]
[184.9]
[ 73.4]
[193.7]
[220.5]
[104.6]
[ 96.2]
[140.3]
[240.1]
[243.2]
[ 38. ]
[ 44.7]
[280.7]
[121. ]
[197.6]
[171.3]
[187.8]
[  4.1]
[ 93.9]
[149.8]
[ 11.7]
[131.7]
[172.5]
[ 85.7]
[188.4]
[163.5]
[117.2]
[234.5]
[ 17.9]
[206.8]
[215.4]
[284.3]
[ 50. ]
[164.5]
[ 19.6]
[168.4]
[222.4]
[276.9]
[248.4]
[170.2]
[276.7]
[165.6]
[156.6]
[218.5]
[ 56.2]
[287.6]
[253.8]
[205. ]
[139.5]
[191.1]
[286. ]
[ 18.7]
[ 39.5]
[ 75.5]
```

```
 [ 17.2]
 [166.8]
 [149.7]
 [ 38.2]
 [ 94.2]
 [177. ]
 [283.6]
 [232.1]]
x_train: [[ 36.9]
 [ 31.5]
 [142.9]
 [209.6]
 [215.4]
 [102.7]
 [  8.6]
 [ 16.9]
 [125.7]
 [104.6]
 [109.8]
 [229.5]
 [253.8]
 [184.9]
 [ 44.7]
 [ 62.3]
 [292.9]
 [172.5]
 [202.5]
 [  7.3]
 [265.2]
 [197.6]
 [218.5]
 [147.3]
 [171.3]
 [217.7]
 [262.7]
 [163.5]
 [100.4]
 [ 76.3]
 [184.9]
 [134.3]
 [273.7]
 [296.4]
 [ 96.2]
 [109.8]
 [255.4]
 [204.1]
 [240.1]
 [193.7]
 [191.1]
 [ 89.7]
 [ 43. ]
 [ 38.2]
 [ 13.1]
 [239.3]
 [ 17.2]
 [210.7]
 [ 25.6]
 [177. ]
 [206.9]
 [ 66.1]
 [149.7]
 [129.4]
 [ 94.2]
 [276.7]
```

```
[276.9]
[  7.8]
[250.9]
[175.7]
[ 11.7]
[ 75.5]
[199.8]
[230.1]
[107.4]
[225.8]
[163.3]
[131.1]
[206.8]
[177. ]
[216.8]
[ 66.9]
[227.2]
[193.2]
[ 97.5]
[ 85.7]
[228.3]
[139.5]
[ 48.3]
[218.4]
[195.4]
[  5.4]
[238.2]
[216.4]
[222.4]
[ 27.5]
[151.5]
[139.2]
[117.2]
[283.6]
[ 57.5]
[237.4]
[213.5]
[ 18.8]
[  4.1]
[164.5]
[ 93.9]
[ 28.6]
[232.1]
[214.7]
[ 19.4]
[280.2]
[290.7]
[136.2]
[ 69. ]
[ 44.5]
[141.3]
[188.4]
[293.6]
[137.9]
[  8.4]
[168.4]
[281.4]
[ 43.1]
[219.8]
[182.6]
[149.8]
[220.3]
[ 95.7]
[248.8]
```

```
 [ 78.2]
 [121. ]
 [112.9]
 [ 17.9]
 [ 80.2]
 [248.4]
 [ 97.2]
 [220.5]
 [284.3]
 [243.2]
 [ 70.6]
 [135.2]
 [ 75.3]
 [116. ]
 [ 75.1]
 [ 38. ]
 [166.8]
 [ 26.8]
 [120.5]
 [262.9]
 [234.5]
 [239.8]
 [286. ]
 [222.4]
 [ 39.5]
 [228. ]
 [210.8]
 [ 73.4]
 [ 88.3]
 [199.1]
 [110.7]
 [266.9]
 [237.4]
 [199.8]
 [187.9]
 [139.3]
 [ 17.2]
 [ 76.4]
 [239.9]
 [ 19.6]]
x_test [[ 69.2]
 [ 50. ]
 [ 90.4]
 [289.7]
 [170.2]
 [ 56.2]
 [  8.7]
 [240.1]
 [ 23.8]
 [197.6]
 [261.3]
 [ 87.2]
 [156.6]
 [187.8]
 [ 76.4]
 [120.2]
 [265.6]
 [  0.7]
 [ 74.7]
 [213.4]
 [287.6]
 [140.3]
 [175.1]
 [131.7]
```

```
 [ 53.5]
 [123.1]
 [165.6]
 [205. ]
 [224. ]
 [ 25.1]
 [ 67.8]
 [198.9]
 [280.7]
 [241.7]
 [ 13.2]
 [ 18.7]
 [ 59.6]
 [180.8]
 [ 68.4]
 [ 25. ]]
```

In [20]:

```
print("Y:",y)

print("y_train:",y_train)
print("y_test",y_test)
```

```
Y: [22.1 10.4  9.3 18.5 12.9  7.2 11.8 13.2  4.8 10.6  8.6 17.4  9.2  9.7
 19.  22.4 12.5 24.4 11.3 14.6 18.  12.5  5.6 15.5  9.7 12.  15.  15.9
 18.9 10.5 21.4 11.9  9.6 17.4  9.5 12.8 25.4 14.7 10.1 21.5 16.6 17.1
 20.7 12.9  8.5 14.9 10.6 23.2 14.8  9.7 11.4 10.7 22.6 21.2 20.2 23.7
  5.5 13.2 23.8 18.4  8.1 24.2 15.7 14.  18.   9.3  9.5 13.4 18.9 22.3
 18.3 12.4  8.8 11.  17.   8.7  6.9 14.2  5.3 11.  11.8 12.3 11.3 13.6
 21.7 15.2 12.  16.  12.9 16.7 11.2  7.3 19.4 22.2 11.5 16.9 11.7 15.5
 25.4 17.2 11.7 23.8 14.8 14.7 20.7 19.2  7.2  8.7  5.3 19.8 13.4 21.8
 14.1 15.9 14.6 12.6 12.2  9.4 15.9  6.6 15.5  7.  11.6 15.2 19.7 10.6
  6.6  8.8 24.7  9.7  1.6 12.7  5.7 19.6 10.8 11.6  9.5 20.8  9.6 20.7
 10.9 19.2 20.1 10.4 11.4 10.3 13.2 25.4 10.9 10.1 16.1 11.6 16.6 19.
 15.6  3.2 15.3 10.1  7.3 12.9 14.4 13.3 14.9 18.  11.9 11.9  8.  12.2
 17.1 15.   8.4 14.5  7.6 11.7 11.5 27.  20.2 11.7 11.8 12.6 10.5 12.2
  8.7 26.2 17.6 22.6 10.3 17.3 15.9  6.7 10.8  9.9  5.9 19.6 17.3  7.6
  9.7 12.8 25.5 13.4]
y_train: [10.8  9.5 15.  15.9 17.1 14.   4.8  8.7 15.9 10.4 12.4 19.7 17.6 15.5
 10.1  9.7 21.4 14.4 16.6  5.5 12.7 11.7 12.2 14.6 19.  19.4 20.2 18.
 10.7 12.  20.7 11.2 20.8 23.8 11.4 16.7 19.8 19.  15.9 19.2 17.3 10.6
  9.6  7.6  5.3 15.7  9.3 18.4  9.5 12.8 12.9  8.6 17.3 11.   9.7 11.8
 27.   6.6 22.2 14.1  7.3  9.9 11.4 22.1 11.5 13.4 16.9 18.  12.2 17.1
 22.3  9.7 14.8 15.2  9.7 13.3 15.5 10.3 11.6 18.  22.4  5.3 20.7 22.6
 11.7  6.9 18.5 12.2 11.9 25.5 11.8 18.9 21.7  7.   3.2 14.5 15.3  7.3
 13.4 17.4  6.6 14.8 12.8 13.2  9.3 10.4 15.5 14.9 20.7 19.2  5.7 11.7
 24.4 10.1 19.6 21.2 10.1 24.7  9.5 18.9 14.6 11.6 11.9  8.   8.8 20.2
  9.6 20.1 15.  25.4 10.5 17.2 11.3 11.  12.6 10.9 19.6  8.8 14.2 12.
 11.9 12.3 15.9 11.5 10.8 21.5 23.8 10.9 12.9 18.3 16.  25.4 12.5 10.6
 14.7 13.4  5.9  9.4 23.2  7.6]
y_test [11.3  8.4  8.7 25.4 11.7  8.7  7.2 13.2  9.2 16.6 24.2 10.6 10.5 15.6
 11.8 13.2 17.4  1.6 14.7 17.  26.2 10.3 14.9 12.9  8.1 15.2 12.6 22.6
 11.6  8.5 12.5 23.7 16.1 21.8  5.6  6.7  9.7 12.9 13.6  7.2]
```

Creating Linear Regression Model

y=aX+b

Where X=Predictor/Independant variable, Y=Response/Dependant Variable, a= Coefficient and b=Intercept

In this example, X is TV and Y is Sales

In [21]:
```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train) # Training the algorithm
```

Out[21]:
```
▾ LinearRegression
LinearRegression()
```

Interpreting Model Coefficients

In [22]:
```
print(regressor.intercept_)
```

```
7.2924937735593645
```

In [23]:
```
print(regressor.coef_)
```

```
[0.04600779]
```

Y=ax+b

Sales=0.04600779(TV)+7.2924937735593645

Making Predictions with Our Model

In [24]:
```
y_pred=regressor.predict(x_test)
```

In [25]:
```
print(y_pred)
```

```
[10.47623281  9.59288325 11.45159795 20.62095042 15.12301956  9.87813155
  7.69276154 18.33896406  8.38747917 16.383633   19.3143292  11.30437303
 14.49731363 15.93275666 10.8074889  12.82263008 19.51216269  7.32469923
 10.72927566 17.11055607 20.52433406 13.74738665 15.34845773 13.35171966
  9.75391052 12.95605267 14.91138373 16.72409064 17.59823864  8.44728929
 10.41182191 16.44344313 20.20688032 18.41257652  7.8997966   8.15283944
 10.03455803 15.61070213 10.43942658  8.44268851]
```

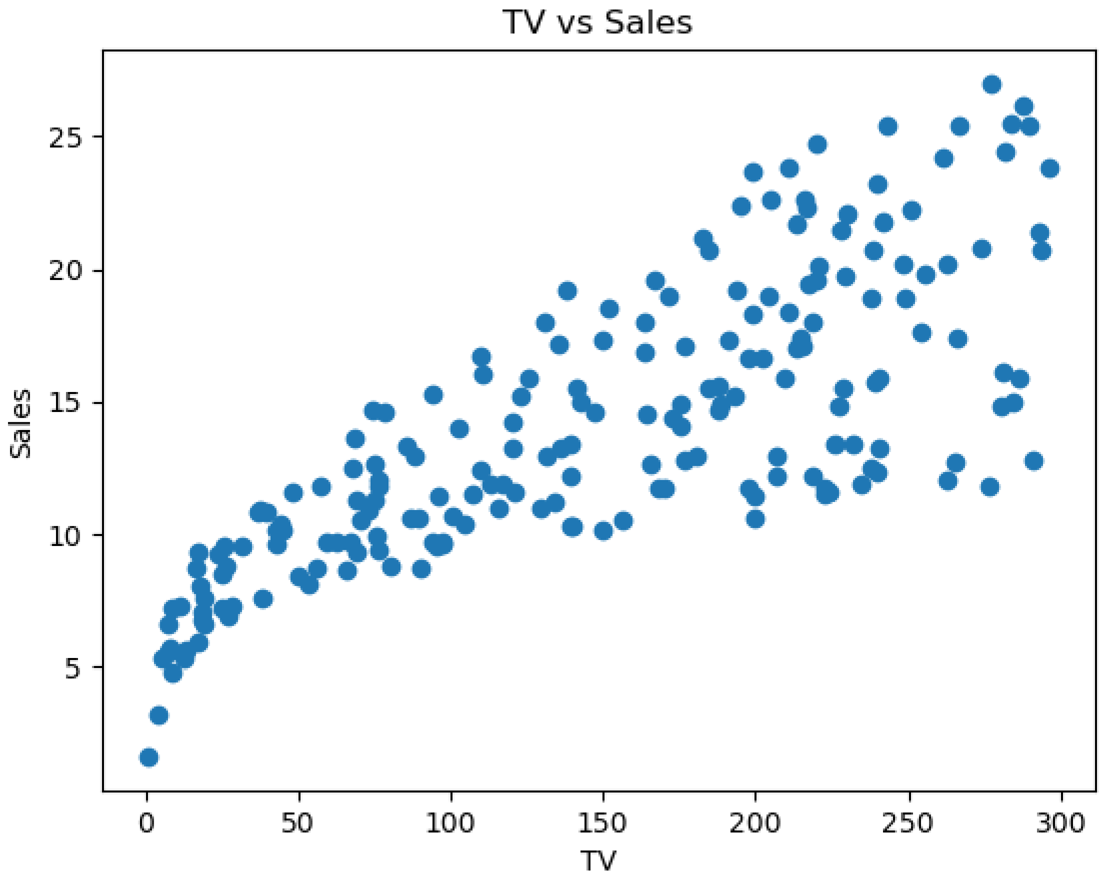
In [26]:
```
df=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
print(df)
```

```
    Actual  Predicted
0     11.3  10.476233
1      8.4   9.592883
2      8.7  11.451598
3     25.4  20.620950
4     11.7  15.123020
5      8.7   9.878132
6      7.2   7.692762
7     13.2  18.338964
8      9.2   8.387479
9     16.6  16.383633
10    24.2  19.314329
11    10.6  11.304373
12    10.5  14.497314
13    15.6  15.932757
14    11.8  10.807489
15    13.2  12.822630
16    17.4  19.512163
17     1.6   7.324699
18    14.7  10.729276
19    17.0  17.110556
20    26.2  20.524334
21    10.3  13.747387
22    14.9  15.348458
23    12.9  13.351720
24     8.1   9.753911
25    15.2  12.956053
26    12.6  14.911384
27    22.6  16.724091
28    11.6  17.598239
29     8.5   8.447289
30    12.5  10.411822
31    23.7  16.443443
32    16.1  20.206880
33    21.8  18.412577
34     5.6   7.899797
35     6.7   8.152839
36     9.7  10.034558
37    12.9  15.610702
38    13.6  10.439427
39     7.2   8.442689
```

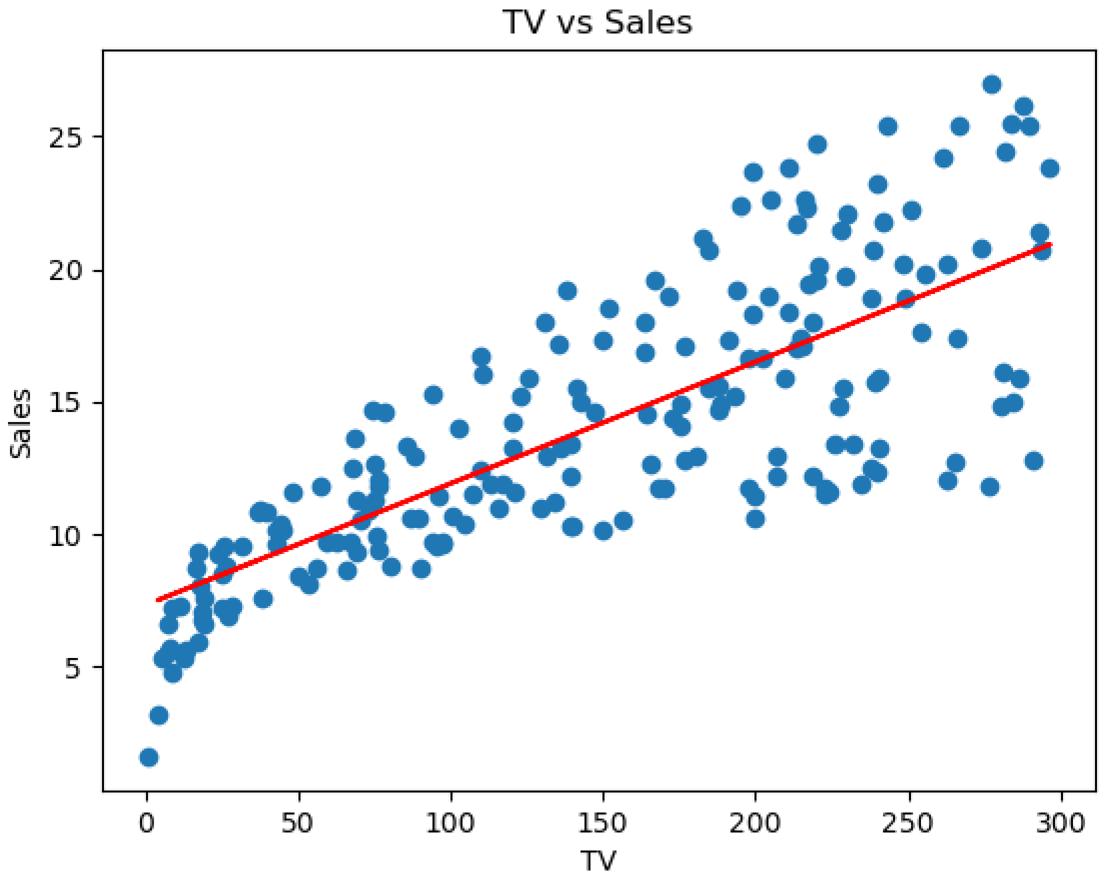
Ploting Linear Regression Model best line¶

In [27]:

```
from matplotlib import pyplot as plt
plt.scatter(d1['TV'],d1["Sales"])
plt.title('TV vs Sales')
plt.xlabel("TV")
plt.ylabel('Sales')
plt.show()
```



In [28]:
```
from matplotlib import pyplot as plt
plt.plot(x_train,regressor.predict(x_train), color='red')
plt.scatter(d1['TV'],d1["Sales"])
plt.title('TV vs Sales')
plt.xlabel("TV")
plt.ylabel('Sales')
plt.show()
```



Model Evaluation Metrics

In [29]:
```
from sklearn import metrics
import numpy as np
print('Mean Absolute Error',metrics.mean_absolute_error(y_test,y_pred))
```

```
Mean Absolute Error 2.5054181789660026
```

In [30]:
```
print('Mean Squared Error',metrics.mean_squared_error(y_test,y_pred))
```

```
Mean Squared Error 10.186181934530222
```

In [31]:
```
# bNo direct function for RMSE
print('Root Mean Squared Error',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
Root Mean Squared Error 3.191579849311344
```

In [32]:
```
print(x_test)
```

```
[[ 69.2]
 [ 50. ]
 [ 90.4]
 [289.7]
 [170.2]
 [ 56.2]
 [  8.7]
 [240.1]
 [ 23.8]
 [197.6]
 [261.3]
 [ 87.2]
 [156.6]
 [187.8]
 [ 76.4]
 [120.2]
 [265.6]
 [  0.7]
 [ 74.7]
 [213.4]
 [287.6]
 [140.3]
 [175.1]
 [131.7]
 [ 53.5]
 [123.1]
 [165.6]
 [205. ]
 [224. ]
 [ 25.1]
 [ 67.8]
 [198.9]
 [280.7]
 [241.7]
 [ 13.2]
 [ 18.7]
 [ 59.6]
 [180.8]
 [ 68.4]
 [ 25. ]]
```

In [33]:

```
train_score_lr = regressor.score(x_train, y_train)
test_score_lr = regressor.score(x_test, y_test)

print("The train score for lr model is: ", train_score_lr)
print("The test score for lr model is: ",test_score_lr)
```

```
The train score for lr model is:  0.5884742462828709
The test score for lr model is:  0.6763151577939721
```

In [34]:

```
actual_minus_predicted = sum((y_test - y_pred)**2)
actual_minus_actual_mean = sum((y_test - y_test.mean())**2)
r2 = 1 - actual_minus_predicted/actual_minus_actual_mean
print('R²:', r2)
```

```
R²: 0.6763151577939721
```

In [35]:

```
from sklearn.metrics import r2_score
# r2 score
print(" R2 Score",r2_score(y_test,y_pred) )
r2=r2_score(y_test,y_pred)
```

```
 R2 Score 0.6763151577939721
```

Adjusted R2 Score

In [36]:
```
# Adjusted R2 Score
x_test.shape
```

Out[36]:
```
(40, 1)
```

In [37]:
```
1-((1-r2)*(6-1)/(6-1-1))
```

Out[37]:
```
0.5953939472424652
```

In [38]:
```
plt.scatter(y_test,y_pred)
plt.xlabel('Test')
plt.ylabel('Predict')
```

Out[38]:
```
Text(0, 0.5, 'Predict')
```