In [1]:

```
# Required Librarys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
#From pc read csv file the data
data=pd.read_csv("C:/Users/MyPc/Documents/Python Scripts/score.csv")
```

In [3]:

```
# first 5 rows by default Show with Columns names
data.head()
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [5]:

```
#Columns Name Show
data.columns
```

Out[5]:

```
Index(['Hours', 'Scores'], dtype='object')
```

In [6]:

```
data.tail()
```

Out[6]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [7]:

```
#show complete data table
data
```

Out[7]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67
19	7.4	69
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [8]:

```
data.describe()
```

Out[8]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [9]:

```
data.corr()
```

Out[9]:

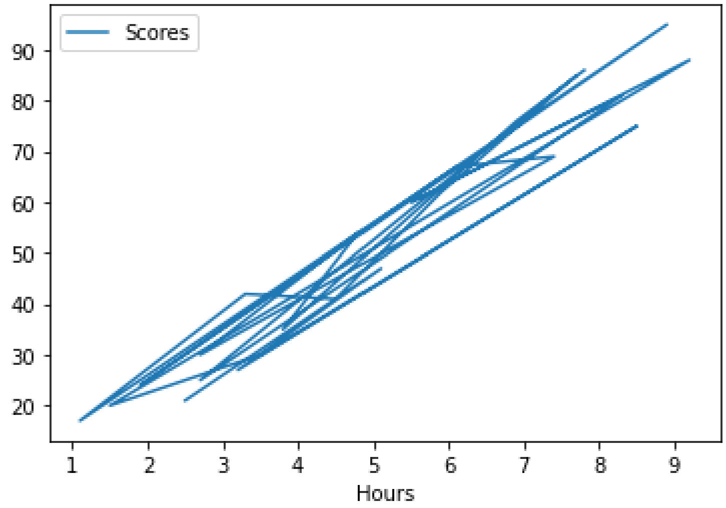
	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [10]:

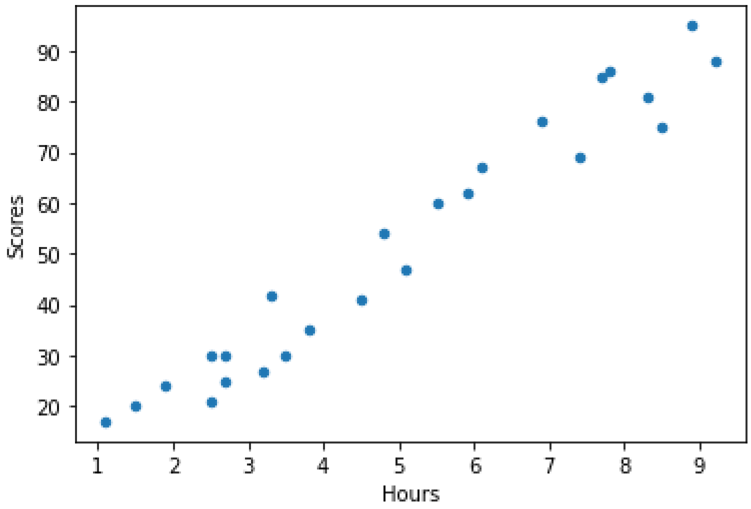
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [12]:
```
#plot data table with respect to x and y axis kind line
data.plot(kind='line',x='Hours',y='Scores');
plt.show()
```



In [13]:
```
#plot data table with respect to x and y axis kind scatter
data.plot(kind='scatter',x='Hours',y='Scores');
plt.show()
```



In [14]:
```
data.shape
```

Out[14]:
```
(25, 2)
```

In [15]:

```
x=data.iloc[0:,:-1].values
x
```

Out[15]:

```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [16]:

```
y=data.iloc[:,1].values
y
```

Out[16]:

```
array([21, 47, 27, 75, 30, 20, 88, 60, 81, 25, 85, 62, 41, 42, 17, 95, 30,
       24, 67, 69, 30, 54, 35, 76, 86], dtype=int64)
```

In [17]:

```
# these library use for machine training, testing and split using linear model wi
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

In [20]:

```
#Train test and split with respect to x axis randomly
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=30)
```

In [21]:

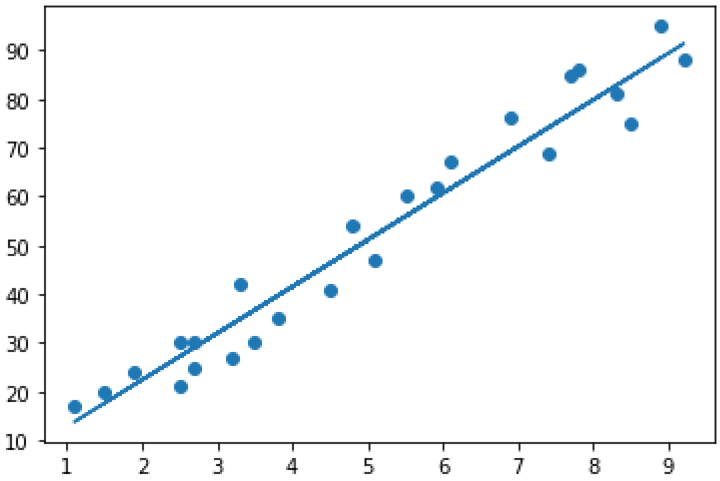
```
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[21]:

```
LinearRegression()
```

In [22]:

```
# show data in graphical shape both dot and line
a=reg.coef_
b=reg.intercept_
line=a*x+b
plt.scatter(x,y)
plt.plot(x,line)
plt.show()
```



In [23]:

```
pred_y=reg.predict(x_test)
```

In [24]:

```
# printing predict results
print(np.concatenate((pred_y.reshape(len(pred_y),1),y_test.reshape(len(y_test),1
```

```
[[76.97173986 85.        ]
 [27.17172289 30.        ]
 [74.09866196 69.        ]
 [27.17172289 21.        ]
 [69.31019879 76.        ]]
```

In [25]:

```
#What will be predicted score if a student studies for 9.25 hrs/ day?
hr=[9.25]
result=reg.predict([hr])
print("The predicted score of a student who studies for 9.25hr/day = {}".format(
```

```
The predicted score of a student who studies for 9.25hr/day = 91.81597568811604
```

In [26]:

```
#library for result error metrics showing
from sklearn import metrics
from sklearn.metrics import r2_score
```

In [27]:

```
# Errors
print("Mean Absolure error: ",metrics.mean_absolute_error(y_test,pred_y))
print("Mean Squared error: ",metrics.mean_squared_error(y_test,pred_y))
print("R2 Score: ",r2_score(y_test,pred_y))
```

```
Mean Absolure error:  5.763344662175538
Mean Squared error:  36.25841394546992
R2 Score:  0.9452422164650992
```

In [ ]: