Task#2 :- To Explore Supervised Learning

In this regression task we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables.

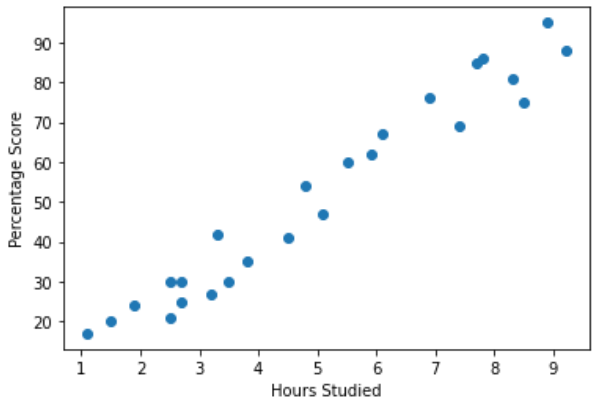
In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

list_1 = [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]

list_2 = [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]

df = pd.DataFrame(list(zip(list_1, list_2)),
               columns =['Hours', 'Scores'])
df
```

```
Matplotlib is building the font cache; this may take a moment.
```

Out[6]:

	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 [7]:
```
plt.scatter(df.Hours,df.Scores)
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```

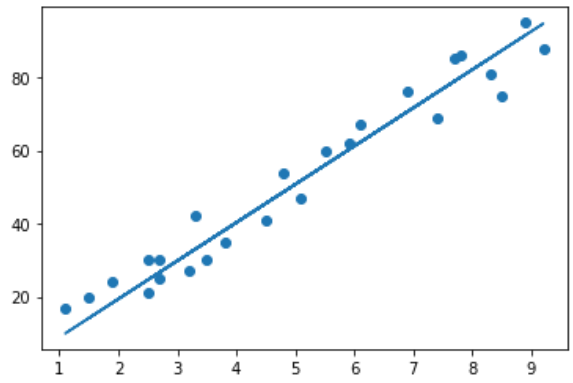


In [8]:
```
x= df.iloc[:, :-1].values
y= df.iloc[:, 1].values
```

In [10]:
```
from sklearn.linear_model import LinearRegression
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=1)
Linear_reg=LinearRegression()
Linear_reg.fit(X_train,y_train)
```

Out[10]:
```
LinearRegression()
```

In [12]:
```
line = Linear_reg.coef_*x+Linear_reg.intercept_

plt.scatter(x, y)
plt.plot(x, line);
plt.show()
```



In [13]:
```
y_pred=Linear_reg.predict(X_test)
```

In [14]:
```
from sklearn.metrics import r2_score
print("R-Squared Value for Training Set: {:.3f}".format(Linear_reg.score(X_train,y_train)))
print("R-Squared Value for Test Set: {:.3f}".format(Linear_reg.score(X_test,y_test)))
```

```
R-Squared Value for Training Set: 0.964
R-Squared Value for Test Set: 0.842
```

In [15]:
```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 7.882398086270432
```

In [16]:
```
y1_pred=Linear_reg.predict([[9.25]])
```

In [17]:
```
print('predicted score :',y1_pred)
```

```
predicted score : [95.22829438]
```

In [ ]: