# PREDICTION USING SUPERVISED ML

## problem statement

predict the percentage of an student based on the no. of study hours

### IMPORTING ALL ESSENTIAL LIBRARIES

In [38]:
```
#PANDAS FOR CONVERTING AND ANALYSING DATASETS
import pandas as pd

#NUMPY FOR SCIENTIFIC COMPUTING
import numpy as np

#MATPLOTLIB FOR PLOTTING GRAPHS
import matplotlib.pyplot as plt

#SEABORN FOR DATA VISUALIZATIONS
import seaborn as sns

#SKLEARN FOR TRAINING OUR MODULE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error


#PRINTING OUR DATASET CONTENT FROM CSV FILE USING read_csv
dataset = pd.read_csv("https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_scores%20-%20student_scor
print(dataset)
```

```
    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
```

**the first five values in the 'student_score' dataset**

In [39]:
```
dataset.head()
```

Out[39]:

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

**number of rows and columns**

In [40]:
```
dataset.shape
```

Out[40]:
```
(25, 2)
```

# VISUALIZAION

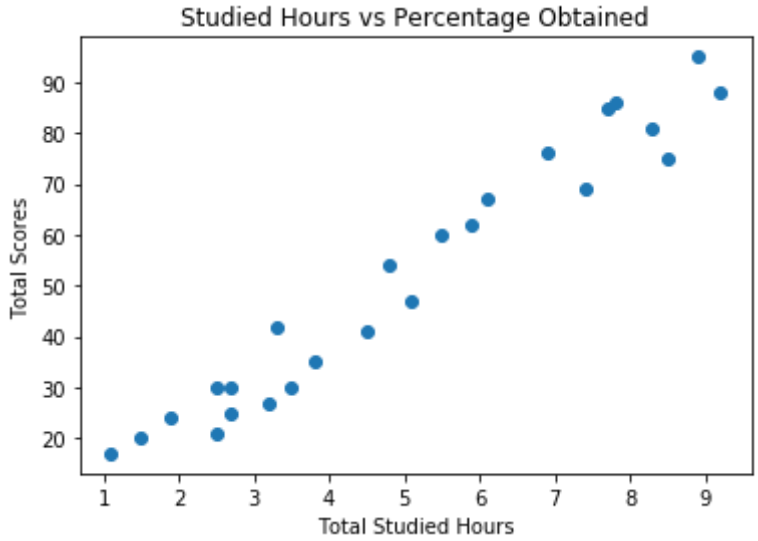
In [41]:
```
#Hours Vs Percentage of Scores

#PLOTTING SCATTER PLOT VISUALIZATION
plt.scatter(dataset['Hours'], dataset['Scores'])

#TITLE OF PLOT
plt.title('Studied Hours vs Percentage Obtained')

#LABELLING X-AXIS AND Y-AXIS
plt.xlabel('Total Studied Hours')
plt.ylabel('Total Scores')

#ACTUAL PLOT
plt.show()
```




# Train-Test Split

# APPLYING SIMPLE LINEAR REGRESSION ON OUR TRAINING DATASET

In [42]:
```
#X will take all the values except for the last column which is our dependent variable (target variable)

#DEPENDENT VARIABLE = X
#INDEPENDENT VARIABLE = Y

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

#IMPORTING train_test_split model from sklearn library
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)

#applying linear regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

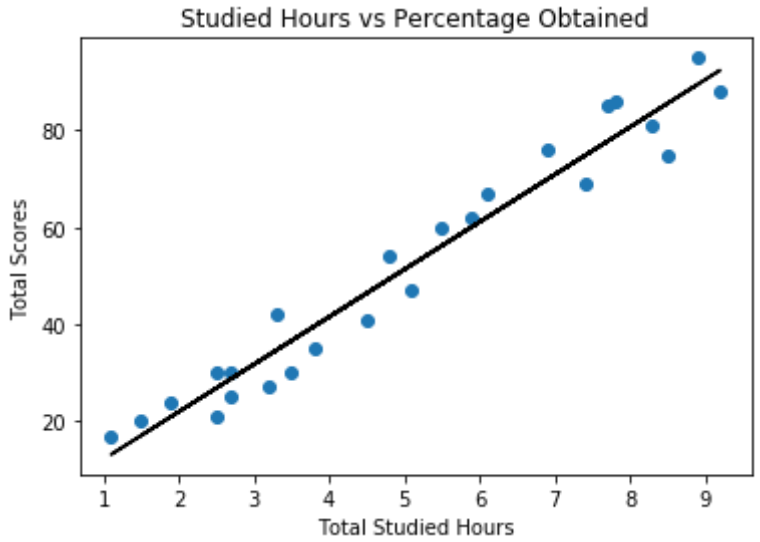
Out[42]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [43]:
```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_

# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line,color = 'black');


#TITLE OF PLOT
plt.title('Studied Hours vs Percentage Obtained')

#LABELLING X-AXIS AND Y-AXIS
plt.xlabel('Total Studied Hours')
plt.ylabel('Total Scores')
plt.show()
```



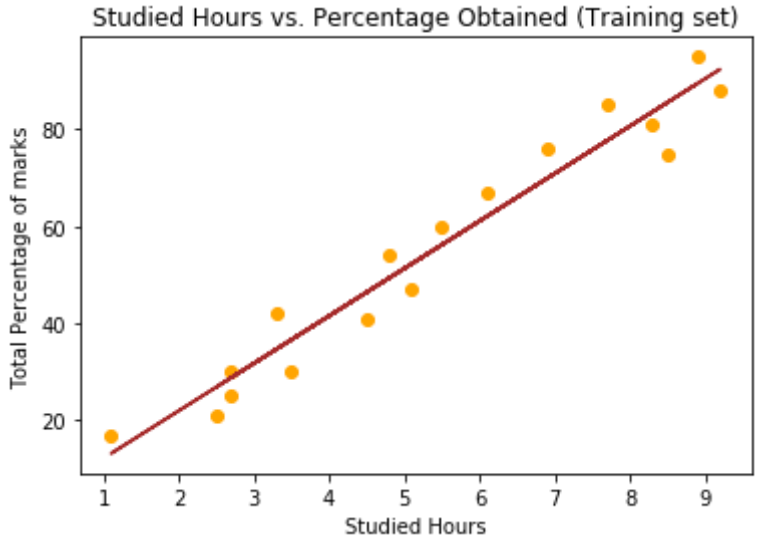

**TEST SET RESULT PREDICTION**

In [44]:
```
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[17.05366541 33.69422878 74.80620886 26.8422321  60.12335883 39.56736879
 20.96909209 78.72163554]
```

# Visualising the Training set results

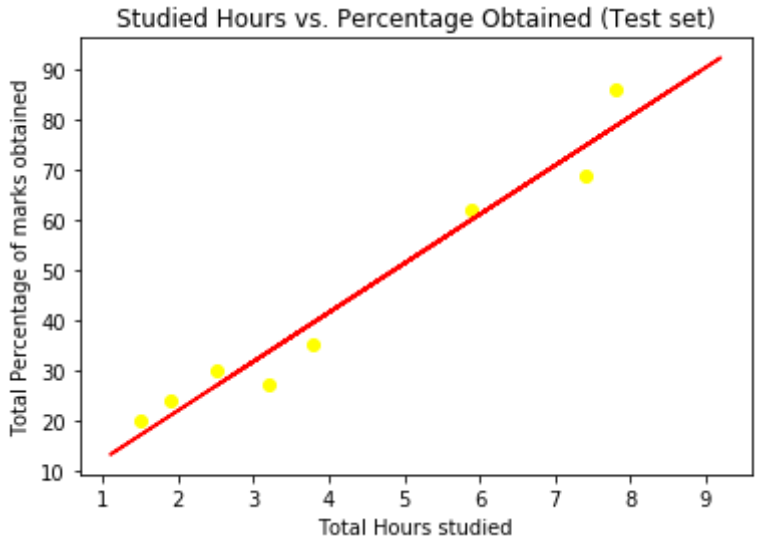
In [45]:
```
plt.scatter(X_train, y_train, color = 'orange')
plt.plot(X_train, regressor.predict(X_train), color = 'brown')
plt.title('Studied Hours vs. Percentage Obtained (Training set)')
plt.xlabel('Studied Hours')
plt.ylabel('Total Percentage of marks')
plt.show()
```




### Visualising the Test set results

In [46]:
```
#scatter plot of dataset
plt.scatter(X_test, y_test, color = 'yellow')
plt.plot(X_train, regressor.predict(X_train), color = 'red')
plt.title('Studied Hours vs. Percentage Obtained (Test set)')

#labelling x-axis and y-axis
plt.xlabel('Total Hours studied')
plt.ylabel('Total Percentage of marks obtained')
plt.show()
```




## Comparing the actual values with the predicted ones.

In [47]:
```
dataset = pd.DataFrame({'Actual Values': y_test, 'Predicted Values': y_pred})
dataset
```

Out[47]:

| | Actual Values | Predicted Values |
|---|---|---|
| **0** | 20 | 17.053665 |
| **1** | 27 | 33.694229 |
| **2** | 69 | 74.806209 |
| **3** | 30 | 26.842232 |
| **4** | 62 | 60.123359 |
| **5** | 35 | 39.567369 |
| **6** | 24 | 20.969092 |
| **7** | 86 | 78.721636 |

# FINAL CONCLUSION: predicting the score

In [48]:
```
dataset = np.array(9.25)
dataset = dataset.reshape(-1, 1)
pred = regressor.predict(dataset)
print("If the student studies for 9.25 hours per day, the score is {}.".format(pred))
```

```
If the student studies for 9.25 hours per day, the score is [92.91505723].
```

## Error Metrics

***FIDING MEAN ABSOLUTE ERROR***

In [49]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.419727808027652
```

***FIDING R-square ERROR***

In [50]:
```
from sklearn.metrics import r2_score
print("The R-Square of the model is: ",r2_score(y_test,y_pred))
```

```
The R-Square of the model is:  0.9568211104435257
```

In [ ]:
```

```