Exploratory Data Analysis on given Dataset

In [2]:
```
#Importing all libraries required in this notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
#Reading data from dataset
url = "http://bit.ly/w-data"
df = pd.read_csv(url)
```

In [21]:
```
#check top 5 rows of data
df.head()
```

Out[21]:

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [5]:
```
#Shape of dataframe
df.shape
```

Out[5]:
```
(25, 2)
```

In [6]:
```
df.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 [7]:
```
df.describe()
```

Out[7]:

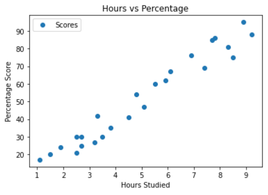
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [8]:
```
#Checking correlation between Hours and Scores
df.corr()
```

Out[8]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [10]:
```
#Plotting the distribution of score
df.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



from the above graph,we can see that there is positive relation between hours studied and percentage score

Preparing the Data

In [12]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, 1].values
```

the next step is to split this data into training and test sets. We'll do this by using Scikit-Learn's built-in train_test_split() method:

In [13]:
```
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)
```

Training the Algorithm

We have split our data into training and testing sets, and now is finally the time to train our algorithm.

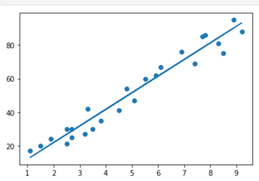
In [14]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

In [15]:
```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_

# Plotting for the test data
plt.scatter(X,y)
plt.plot(X, line);
plt.show()
```



Making Predictions

In [16]:
```
# Testing data - In Hours
print(X_test)

# Predicting the scores
y_pred = regressor.predict(X_test)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

comparing Actual vs Predicted

In [17]:
```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[17]:

	Actual	Predicted
0	20	16.884145
1	27	33.732261
2	69	75.357018
3	30	26.794801
4	62	60.491033

In [20]:
```
# You can also test with your own data
hours = 9.25
own_prediction = regressor.predict([[hours]])
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(own_prediction[0]))
```

```
No of Hours = 9.25
Predicted Score = 93.69173248737538
```

In [23]:
```
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

In [ ]:
```

```