Task #1 = Predict the percentage of an student based on the no. of study hours by The Spark Foundation.

Abhishek Mohan Ambawale

Importing Libraries.

In [1]:

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

Importing Dataset.

In [2]:

```
df = pd.read_csv("https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/studen
t_scores%20-%20student_scores.csv")
df.head()
```

Out[2]:

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

In [3]:

```
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 [4]:

```
df.duplicated()
```

Out[4]:

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
```

```
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
dtype: bool
```

Splitting the data into dependent and independent variables.

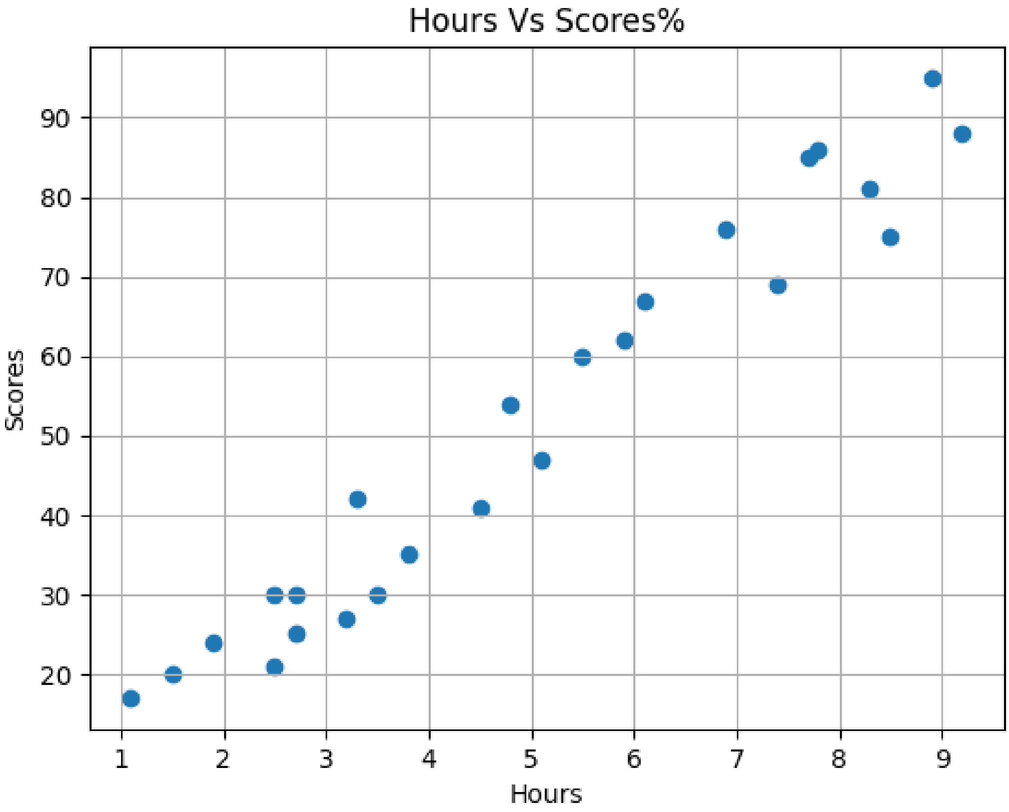
In [5]:

```
x = df['Hours'].values.reshape(-1,1)
y = df['Scores'].values.reshape(-1,1)
```

Visualising the data.

In [6]:

```
plt.scatter(x,y)
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Hours Vs Scores%")
plt.grid()
plt.show()
```



The above graph shows a Linear Relation between Hours and Scores. As we can see the data is quite linear so it doesn't need any Feature Scaling.

Splitting the data into Train and Test data

Splitting the data into Train and Test data.

In [7]:

```
from sklearn.model_selection import train_test_split
```

In [8]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [9]:

```
from sklearn.linear_model import LinearRegression
```

In [10]:

```
lr = LinearRegression()
```

In [11]:

```
lr.fit(x_train,y_train)
```

Out[11]:

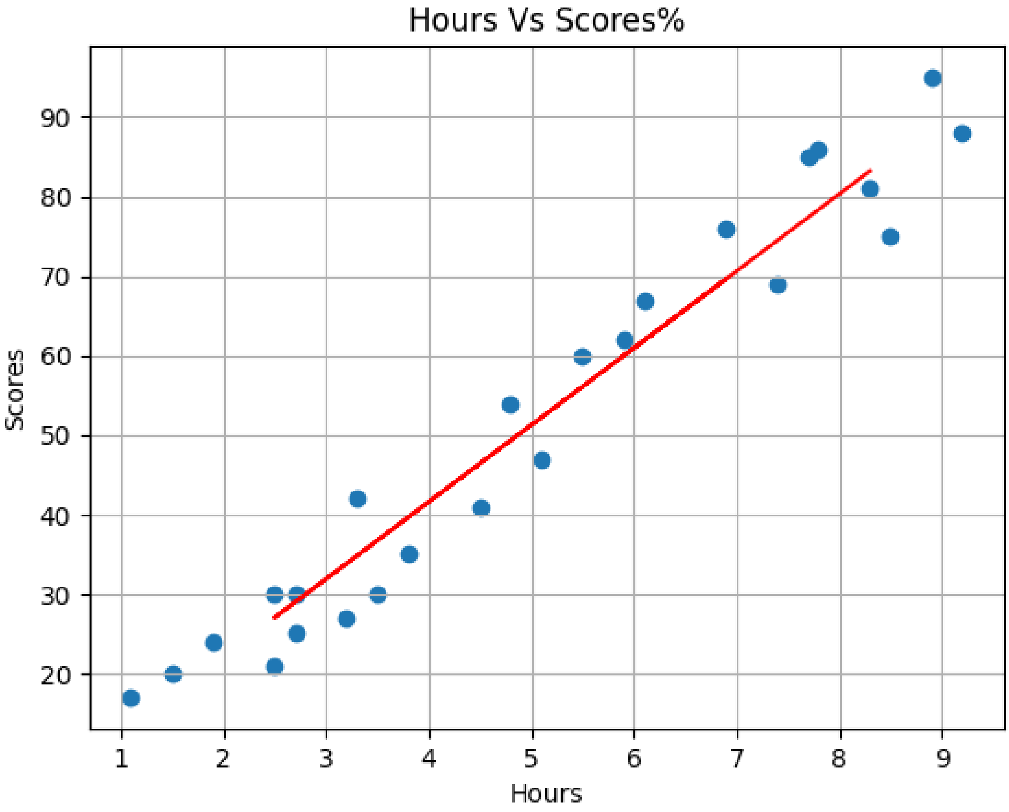
```
▾ LinearRegression
LinearRegression()
```

In [12]:

```
y_pred = lr.predict(x_test)
```

In [13]:

```
plt.scatter(x,y)
plt.plot(x_test,lr.predict(x_test),color='r')
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.title("Hours Vs Scores%")
plt.grid()
plt.show()
```



Evaluating the performance of a linear regression model.

In [14]:

```
from sklearn.metrics import r2_score
```

In [15]:

```
score = r2_score(y_test,y_pred)
```

In [16]:

```
score
```

Out[16]:

```
0.9678055545167994
```

In [17]:

```
df_predict = pd.DataFrame({"Hours": x_test.reshape(1,-1)[0] , "Actual Score" : y_test.reshape(1,-1)[0] , "Predicted Score" : y_pred.reshape(1,-1)[0]})
df_predict
```

Out[17]:

	Hours	Actual Score	Predicted Score
0	8.3	81	83.188141
1	2.5	30	27.032088
2	2.5	21	27.032088
3	6.9	76	69.633232
4	5.9	62	59.951153

Sample Predictions

In [18]:

```
lr.predict(np.array([[8.3 ]]))[0]
```

Out[18]:

```
array([83.18814104])
```