In [1]:

```
"""score prediction of students"""
# Author - Ketki Kale
# Importing all libraries required
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

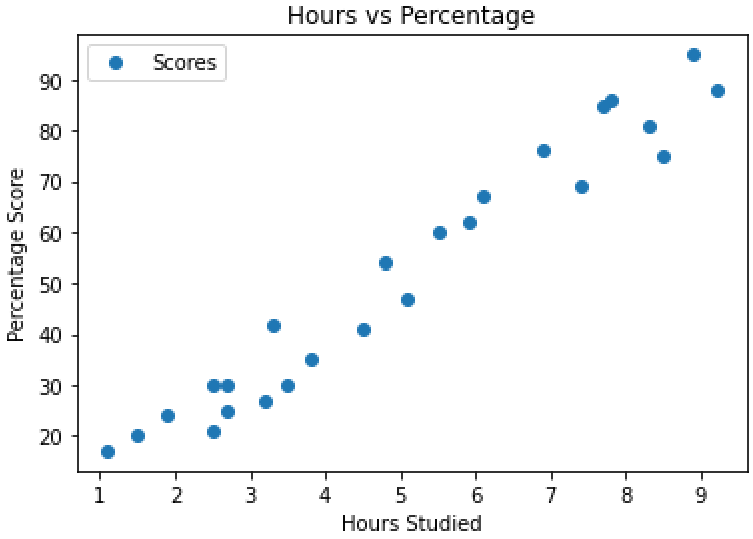
In [2]:

```
#  Load dataset into dataframe
student_score_df = pd.read_csv("students_scores.csv")
print(student_score_df.head(5))
```

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

In [3]:

```
# Plotting the distribution of scores
student_score_df.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



In [4]:

```
# Creating feature variable(X) and outcome variable(y)
X = student_score_df.iloc[:, :-1].values
y = student_score_df.iloc[:, 1].values
print(X)
print(y)
```

```
[[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]]
[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]
```

In [5]:

```
# Splitting dataset into Training and Validation Sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, rand
om_state=0)
```

In [6]:

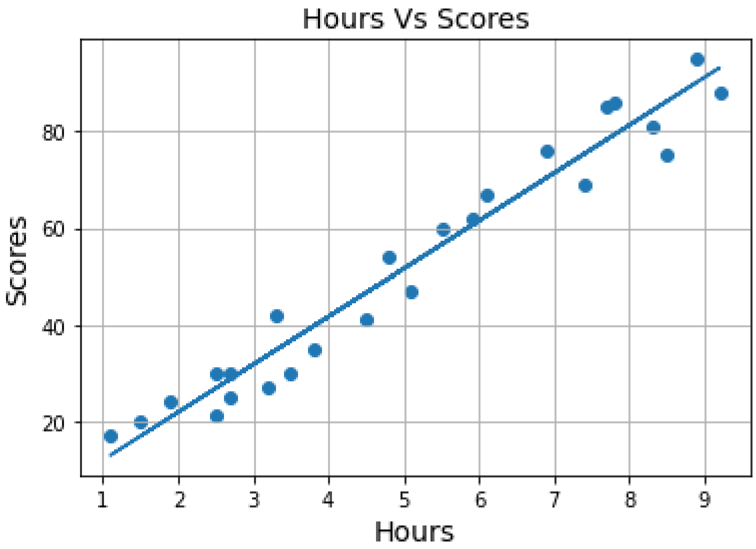
```
# Fitting of regression model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor = regressor.fit(X_train, y_train)
beta = regressor.coef_
print('Value of beta(regression coefficient): {}'.format(beta))
intercept = regressor.intercept_
print('Value of intercept : {}'.format(intercept))
```

```
Value of beta(regression coefficient): [9.91065648]
Value of intercept : 2.018160041434662
```

In [7]:

```
# Plotting the regression line(y = ax  + b)(y= beta*X + intercept)(y=beta1X +
 beta0)
line = regressor.coef_*X+regressor.intercept_
```

In [8]:
```
# Plotting for the test data
plt.scatter(X, y)
plt.plot(X, line)
plt.title('Hours Vs Scores', fontsize=14)
plt.xlabel('Hours', fontsize=14)
plt.ylabel('Scores', fontsize=14)
plt.grid(True)
plt.show()
```



In [9]:
```
# LR is sensitive to outliner analysis. Influential observation
from scipy.stats import zscore
student_score_df['z_score_percent'] = zscore(student_score_df.Scores)
b = student_score_df[(student_score_df.z_score_percent > 3.0) | (student_score
_df.z_score_percent < -3.0)]
print('z_score is {}'.format(b))
```

```
z_score is Empty DataFrame
Columns: [Hours, Scores, z_score_percent]
Index: []
```

In [10]:

```python
# Testing data - In Hours
print(X_test)
# Predicting the scores
y_pred = regressor.predict(X_test)
print(y_pred)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(df)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
[16.88414476 33.73226078 75.357018   26.79480124 60.49103328]
   Actual  Predicted
0      20  16.884145
1      27  33.732261
2      69  75.357018
3      30  26.794801
4      62  60.491033
```

In [11]:

```python
# Prediction of student's percentage who studies for 9.25 hours
hours = [[9.25]]
own_pred = regressor.predict(hours)
print("No of Hours = {}".format(hours))
print("Predicted Score = {}".format(own_pred[0]))
```

```
No of Hours = [[9.25]]
Predicted Score = 93.69173248737539
```

In [12]:

```python
# Different measures for accuracy of prediction
accuracy = regressor.score(X_test, y_test)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.9454906892105354
```

In [13]:

```python
# Formula for Mean Absolute Percentage Error
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
```

In [14]:

```
from sklearn import metrics
print('Mean Absolute Error: \t \t \t',
      metrics.mean_absolute_error(y_test, y_pred))

from sklearn.metrics import r2_score, mean_squared_error
print('r2_score: \t \t \t \t', np.abs(r2_score(y_test, y_pred)))

print('Mean Squared Error: \t \t \t', np.sqrt(mean_squared_error(y_test, y_pre
d)))
print('Mean Absolute Percentage Error: \t', mean_absolute_percentage_error(y_t
est, y_pred))
```

```
Mean Absolute Error:                     4.183859899002982
r2_score:                                0.9454906892105354
Mean Squared Error:                      4.647447612100373
Mean Absolute Percentage Error:          12.568891617045686
```