GRIP spark foundation

Task - 1

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

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

from sklearn.model_selection import train_test_split
from sklearn. linear_model import LinearRegression

%matplot inline
```

In [12]:
```
df = pd.read_excel('C:\\Users\\Administrator\\Desktop\\GRIP spark foundation\\task 1\\Book1.xlsx')
```

In [13]:
```
df.head()
```

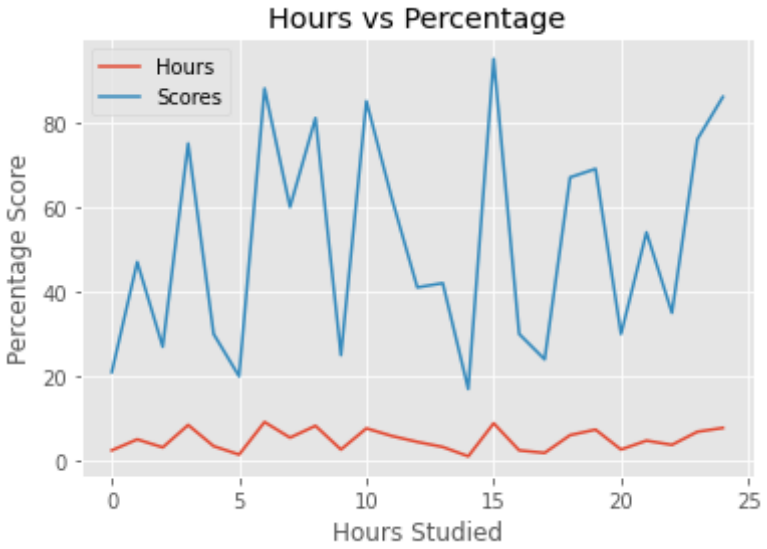
Out[13]:

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

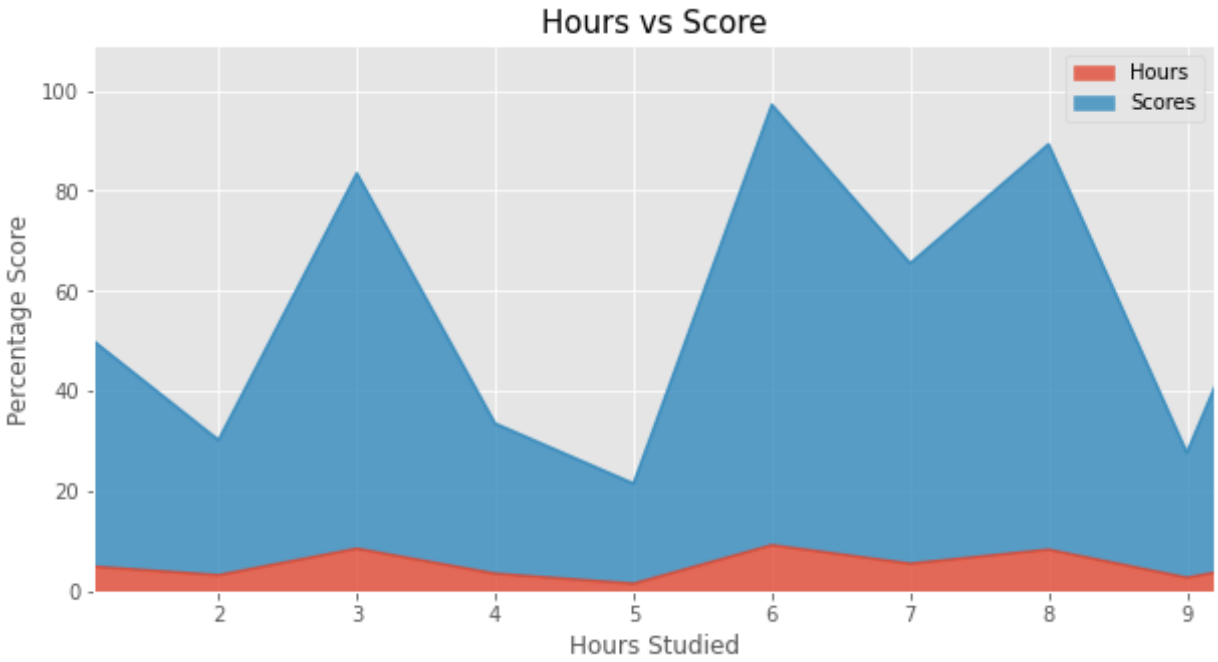
In [14]:
```
# Data Visualization
```

In [15]:
```
plt.style.use('ggplot')
```

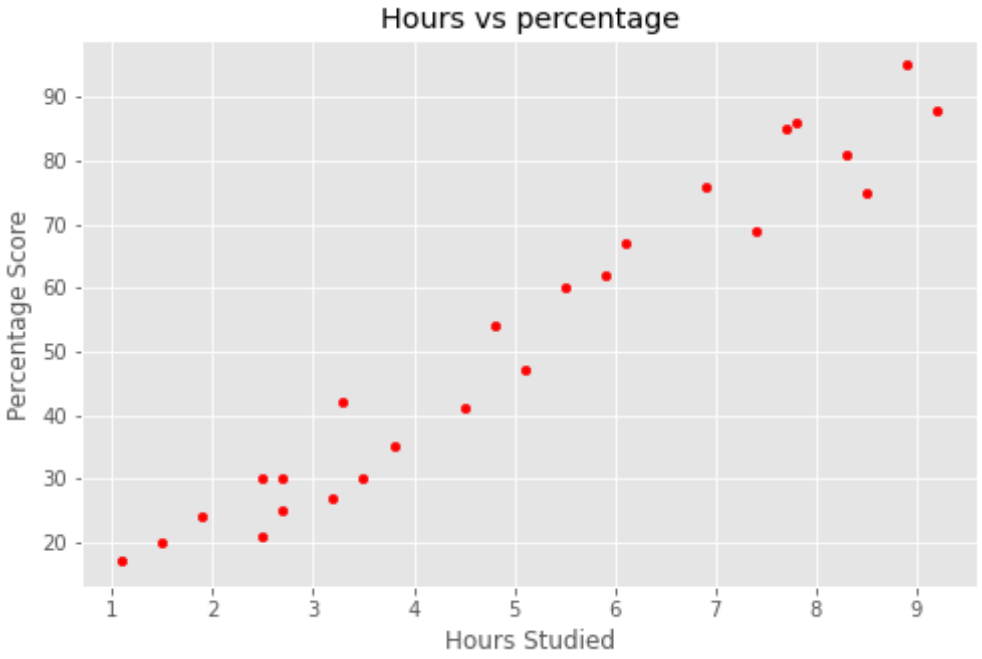
```
df.plot(kind='line')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



In [16]:
```
xmin=min(df.Hours)
xmax=max(df.Hours)
df.plot(kind='area',alpha=0.8, stacked=True,figsize=(10,5),xlim=(xmin,xmax))
plt.title('Hours vs Score',size=15)
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



In [17]:
```
df.plot(kind='scatter',x='Hours', y='Scores',color='r',figsize=(8,5))
plt.title('Hours vs percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



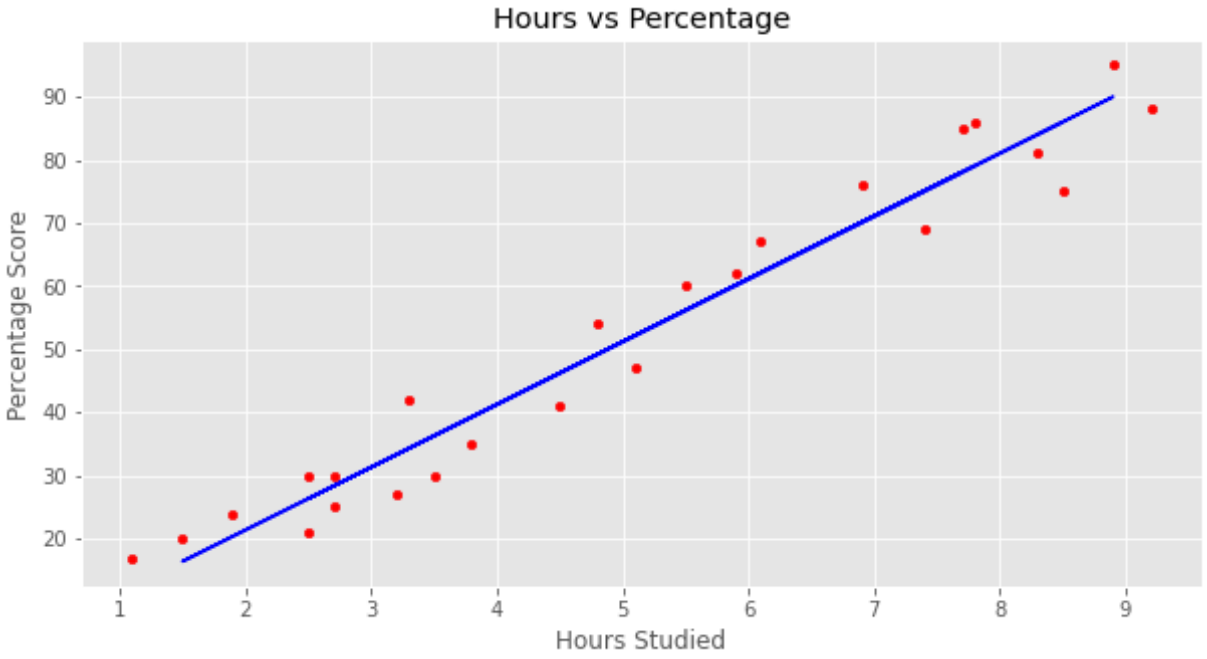
by visualization we come to know that this problem can be easily solved by linear regression.

In [18]:
```
x=np.asanyarray(df[['Hours']])
y=np.asanyarray(df['Scores'])

train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2,random_state=2)

regressor = LinearRegression()
regressor.fit(train_x,train_y)

print('Training completed\n')
print('Coefficients: ',regressor.coef_)
print('intercept:',regressor.intercept_)
```

```
Training completed
```

```
Coefficients:  [9.94061514]
intercept: 1.5079104828268655
```

In [19]:
```
df.plot(kind='scatter',x='Hours', y='Scores',figsize=(10,5),color='r')
plt.plot(train_x, regressor.coef_[0]*train_x + regressor.intercept_,color='b')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



the blue line is the best fit line for this data

Evaluation of the model

In [20]:
```
# using metrics to find mean absolute error and r2 to see the accuracy

from sklearn import metrics
```

```
from sklearn.metrics import r2_score

y_pred=regressor.predict(test_x)
print('Mean Absolute Error :{}'.format(metrics.mean_absolute_error(y_pred,test_y)))
print('R2-score: %.2f' %r2_score(y_pred, test_y) )
```

```
Mean Absolute Error :4.877039354964483
R2-score: 0.98
```

*Mean absolute Error - it is mean of absolute value of errors r2-score: ot is not error but its the metric for accuracy for the model. Higher the r2 value higher is the accuracy of model. Best score is 1

In [26]:
```
#comparing actual vs predicted

df2 = pd.DataFrame({'actual': test_y, "Predicted": y_pred})
df2
```

Out[26]:

	actual	Predicted
0	17	12.442587
1	21	26.359448
2	24	20.395079
3	88	92.961570
4	76	70.098155

Predicting the Score with the single input value

In [28]:
```
hours= 9.5
predicted_score=regressor.predict([[hours]])

print(f'No. of hours = {hours}')
print(f'predicted Score={predicted_score[0]}')
```

```
No. of hours = 9.5
predicted Score=95.94375434264262
```

In [ ]:
```

```

In [ ]: