Task 1- GRIP at Sparks Foundation

predict the percentage of marks that a student is expected to score based upon the number of hours they studied. This is a simple linear regression task as it involves just two variables

In [48]:
```
# Importing all libraries required in this notebook
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [49]:
```
# read dataset
df = pd.read_csv("D://DK//dddddd//Task_grip//data1.csv")
```

In [50]:
```
df.head()
```

Out[50]:

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

In [51]:
```
df.tail()
```

Out[51]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

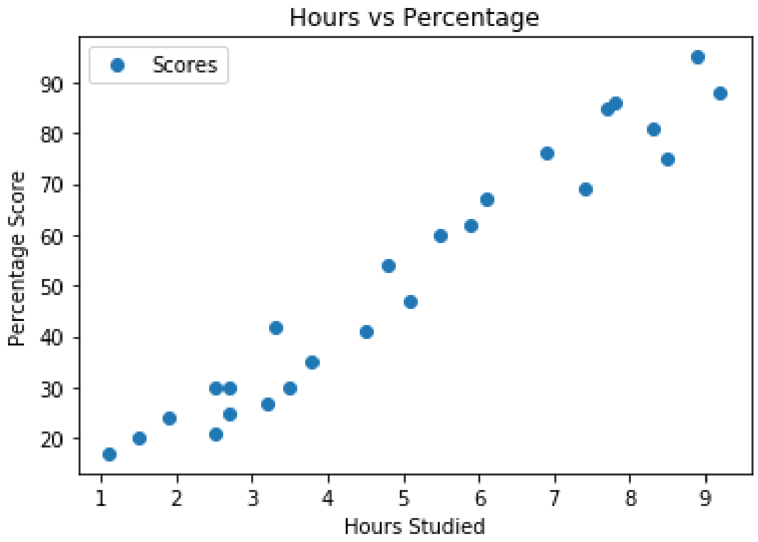
In [33]:
```
df.columns
```

Out[33]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [52]:
```
df.shape
```

Out[52]:
```
(25, 2)
```

In [53]:
```
# Plotting the distribution of scores
df.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```



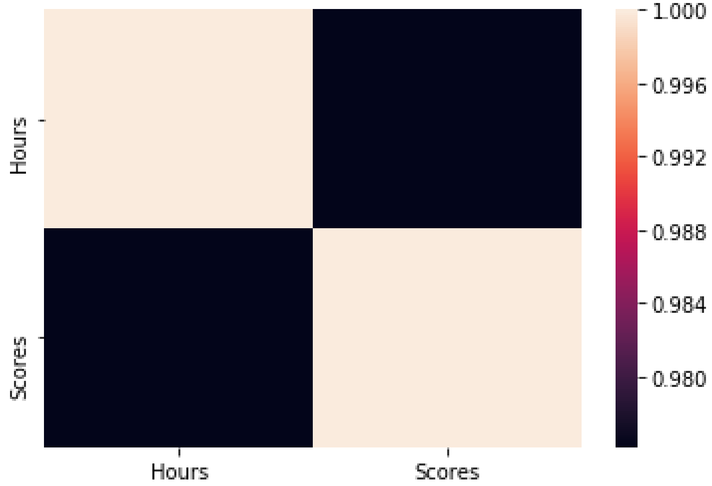
From the graph above, we can clearly see that there is a positive linear relation between the number of hours studied and percentage of score

In [54]:

```
###corraltion matrix
df_corr= df.corr()['Hours'][:-1]
corr= df.drop('Hours',axis=1).corr
corr = df.corr()
corr = (corr)
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)

corr
```

Out[54]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000



Model Building and Evaluation

In [55]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, 1].values
```

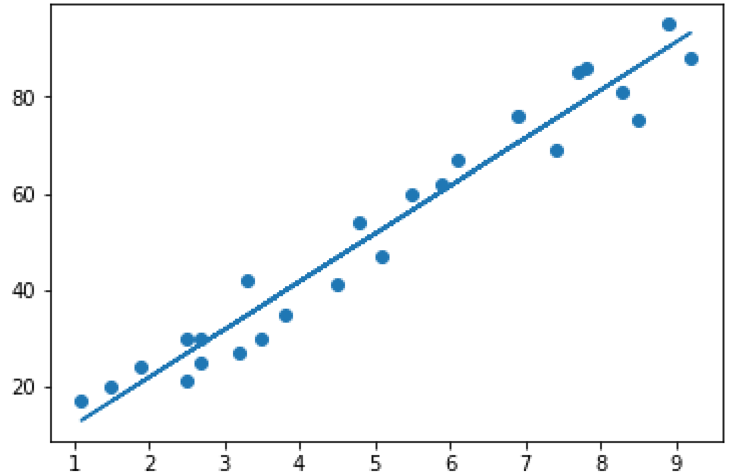
In [56]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                            test_size=0.2, random_state=0)
```

Training the ML Algorithm_Linear Regression

In [57]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

print("Training complete.")
```

```
Training complete.
```

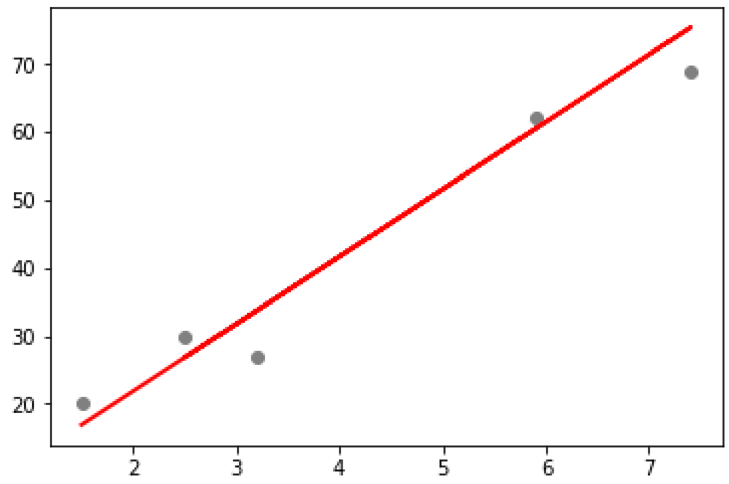
In [58]:
```
# Plotting the regression line
line = regressor.coef_*X+regressor.intercept_

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



In [59]:
```
print(X_test) # Testing data - In Hours
y_pred = regressor.predict(X_test) # Predicting the scores
```

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

In [60]:
```
plt.scatter(X_test, y_test,  color='gray')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.show()
```



In [61]:
```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[61]:

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

In [62]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pre
d))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred
))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_
test, y_pred)))
```

```
Mean Absolute Error: 4.183859899002975
Mean Squared Error: 21.5987693072174

Root Mean Squared Error: 4.6474476121003665
```

support vecotor machine

In [63]:
```
from sklearn.svm import SVR
```

In [64]:
```
regressor  = SVR(kernel='rbf')
fitSVR = regressor .fit(X_train, y_train)
y_pred = fitSVR.predict(X_test)
```

In [65]:
```
print(X_test) # Testing data - In Hours
y_pred = regressor.predict(X_test) # Predicting the scores
```

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

In [66]:
```
# Comparing Actual vs Predicted
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Out[66]:

	Actual	Predicted
0	20	46.344282
1	27	45.545300
2	69	58.849555
3	30	45.157705

	Actual	Predicted
4	62	54.771739

In [67]:
```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pre
d))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred
))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_
test, y_pred)))
```

```
Mean Absolute Error: 15.485198578582589
Mean Squared Error: 284.5969278896682
Root Mean Squared Error: 16.87000082660544
```