!

SPARK FOUNDATION

Predict the percentage of an student based on the no. of study hours(TASK 1)

step 1 :

Import the reqiured library

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv("data_list.csv")
```

Analyze the data

In [3]:
```
data.head(19)
```

Out[3]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25
10	7.7	85
11	5.9	62
12	4.5	41
13	3.3	42
14	1.1	17
15	8.9	95
16	2.5	30
17	1.9	24
18	6.1	67

In [4]:
```
data.describe()
```

Out[4]:

	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 [5]:
```
data.shape
```

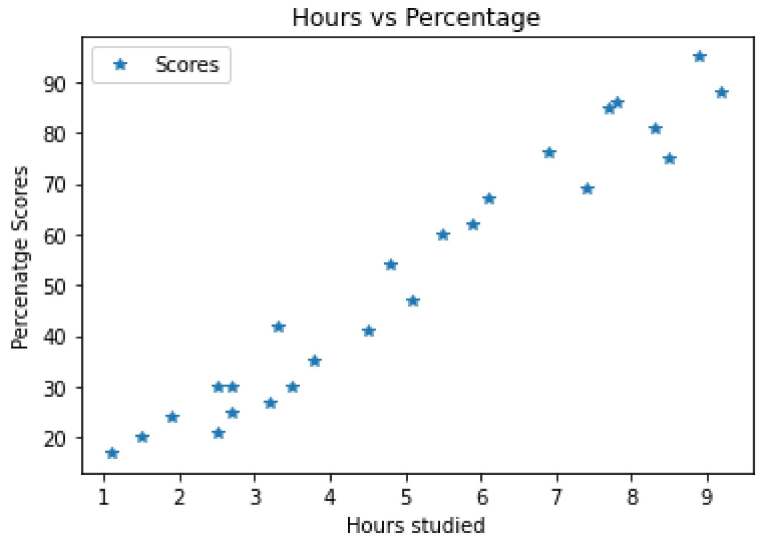
Out[5]:
```
(25, 2)
```

In [6]:
```
data.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
```

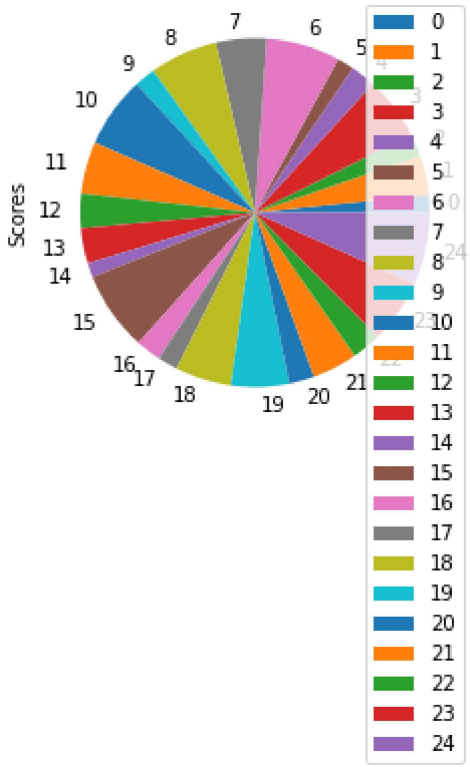
Plotting The Data In Various Type

In [7]:
```
data.plot(x='Hours',y='Scores',style='*')
plt.title('Hours vs Percentage')
plt.xlabel('Hours studied')
plt.ylabel('Percenatge Scores')
plt.show()
```



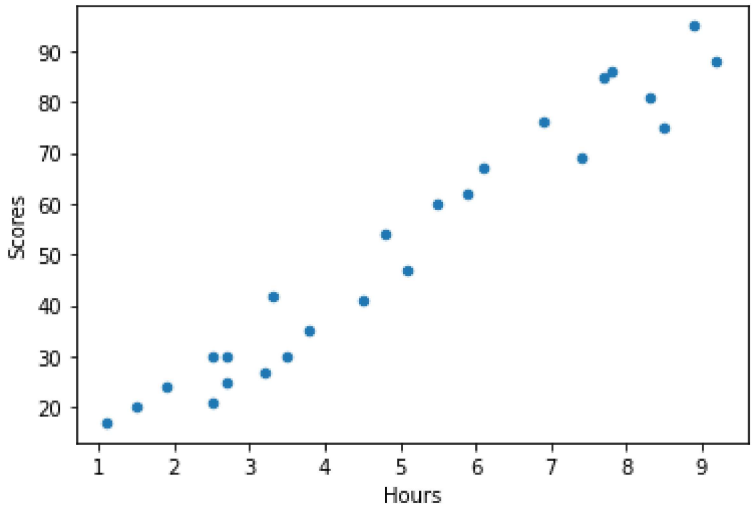
In [8]:
```
data.plot.pie(x='Hours',y='Scores')
```

Out[8]: <AxesSubplot:ylabel='Scores'>



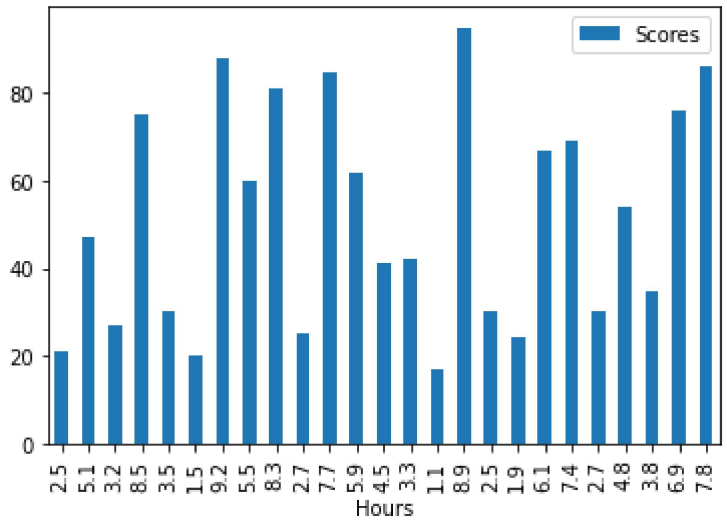
In [9]:
```
data.plot.scatter(x='Hours',y='Scores')
```

Out[9]: <AxesSubplot:xlabel='Hours', ylabel='Scores'>



In [10]:
```
data.plot.bar(x='Hours',y='Scores')
```

Out[10]: <AxesSubplot:xlabel='Hours'>



Preparing The Data

In [11]:
```
X = data.iloc[:,:-1].values
y = data.iloc[:,1].values
```

Splitting The Data

In [12]:
```
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_s
```

Check Train DataSet

In [13]:
```
from sklearn.linear_model import LinearRegression
mo = LinearRegression()
mo.fit(X_train, y_train)
```

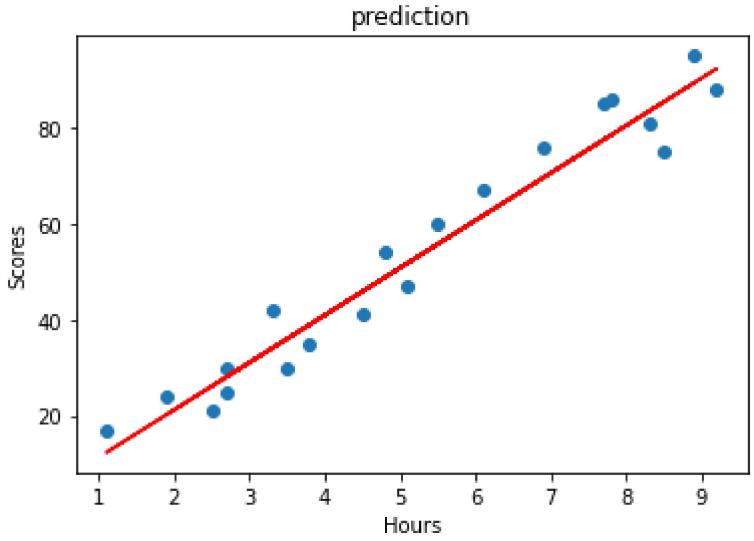
Out[13]:
```
LinearRegression()
```

In [14]:
```
print(mo.intercept_)
print(mo.coef_)
```

```
2.0181600414346974
[9.91065648]
```

Plotting The Regression Line

In [15]:
```
plt.scatter(X_train, y_train)
plt.plot(X_train,1.495142109236383 + 9.87171443*X_train, 'r' )
plt.title("prediction")
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.show()
```



Making Predictions

In [16]:
```
print(X_test)
y_pred = mo.predict(X_test)
```

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

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

Predicting the score for the study of 9.25 hours

In [18]:
```
pred_score = mo.predict([[9.25]])
print("The predictedn score is :",pred_score)
```

```
The predictedn score is : [93.69173249]
```

Evaluating the Model

In [19]:
```
from sklearn import metrics
print('Mean Absolute Erorr:',metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Erorr: 4.183859899002975
```