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

In [2]:
```
url = "https://raw.githubusercontent.com/AdiPersonalWorks/Random/master/student_s
df = pd.read_csv(url)
df.sample(10)
```

Out[2]:

	Hours	Scores
20	2.7	30
21	4.8	54
23	6.9	76
22	3.8	35
10	7.7	85
15	8.9	95
14	1.1	17
18	6.1	67
6	9.2	88
24	7.8	86

In [3]:
```
#analysis
df.describe()
```

Out[3]:

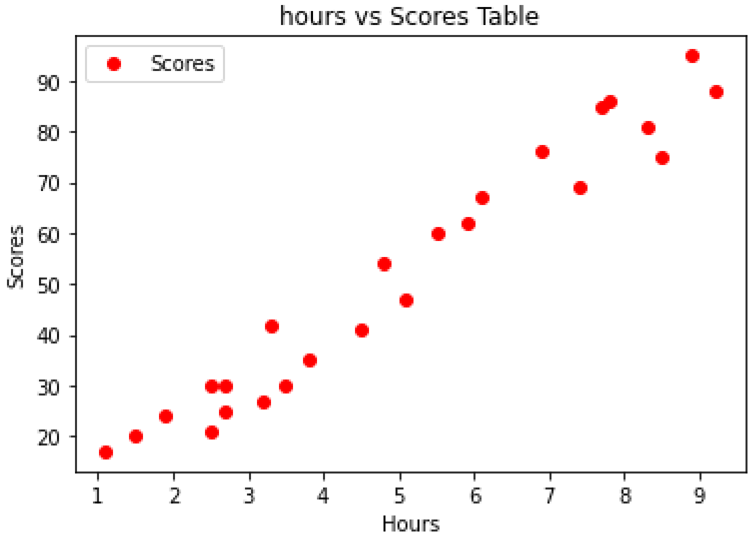
	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [4]:
```
df.dtypes
```

Out[4]:
```
Hours      float64
Scores       int64
dtype: object
```

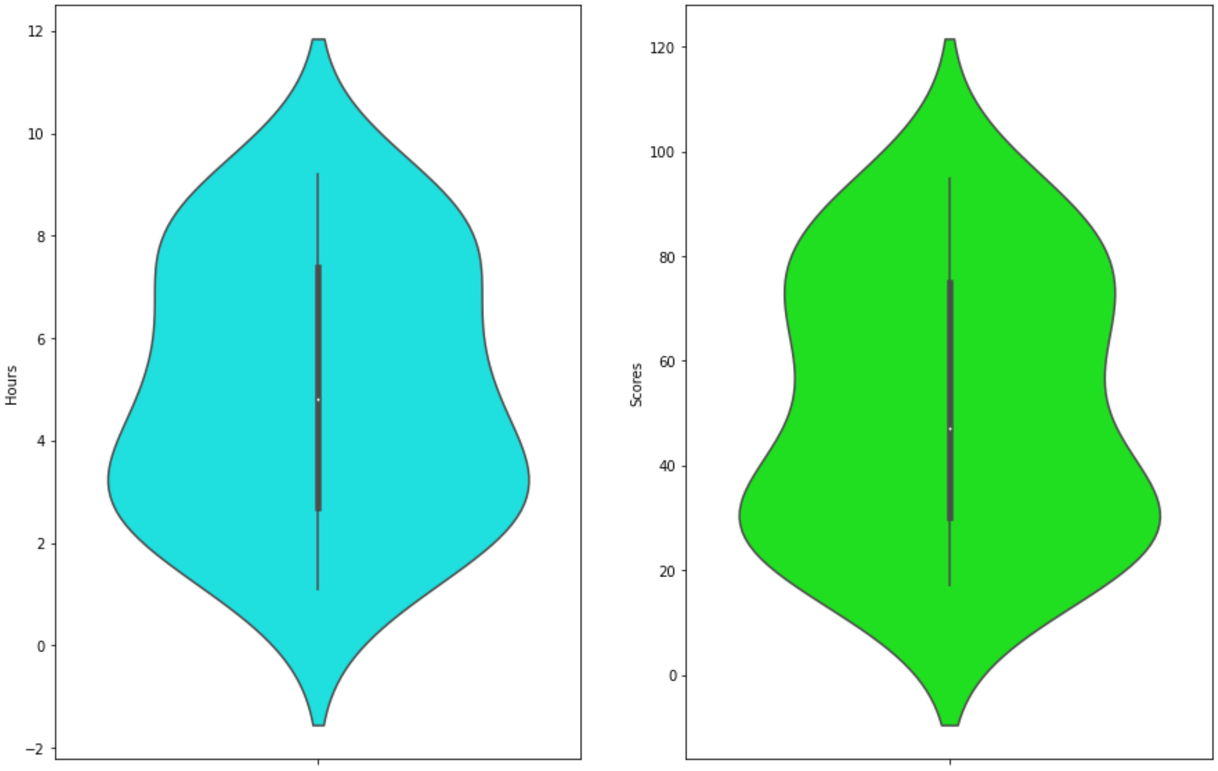
plotting the graph

In [5]:
```
df.plot(x="Hours",y="Scores",style="o",color="r")
plt.title("hours vs Scores Table")
plt.xlabel("Hours")
plt.ylabel("Scores")
plt.show()
```



In [6]:
```
fig,axs=plt.subplots(1,2,figsize=(15,10))
sns.violinplot(y='Hours',data=df,ax=axs[0],color="#00FFFF")
sns.violinplot(y='Scores',data=df,ax=axs[1],color="#00FF00")
```

Out[6]:
```
<AxesSubplot:ylabel='Scores'>
```



Both the attributes are distributed similarly

In [7]:

```
X = df.iloc[:, :-1].values
Y = df.iloc[:, 1].values
```

TRAIN AND TEST DEFINING

In [8]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=2)
```

MODEL SELECTION AND TRAINING

In [9]:

```
from sklearn.linear_model import LinearRegression
doregression = LinearRegression()
doregression.fit(X_train,Y_train)

print("the training of the model has been sucessfully completed")
```

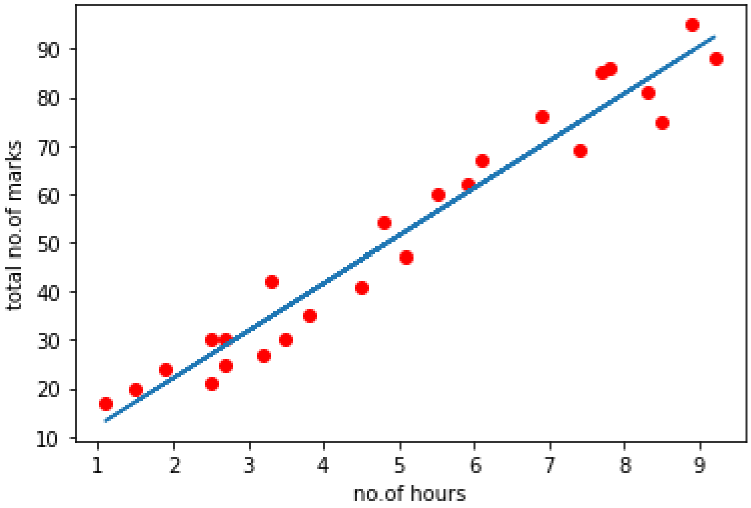
```
the training of the model has been sucessfully completed
```

plotting the scatter plot

In [10]:

```
#plotting the scatter plot
line = doregression.coef_*X+doregression.intercept_

plt.scatter(X,Y,color ="r")
plt.plot(X,line)
plt.xlabel("no.of hours")
plt.ylabel("total no.of marks")
plt.show()
```



making predictions

In [11]:
```
print(X_test)
y_predict = doregression.predict(X_test)
```

```
[[2.7]
 [1.9]]
```

In [12]:
```
dt = pd.DataFrame({'Actual':Y_test,'Predicted':y_predict})
```

In [13]:
```
dt
```

Out[13]:

	Actual	Predicted
0	25	28.919618
1	24	21.099157

now let's check logistic regression

In [14]:
```
from sklearn.linear_model import LogisticRegression
regression = LogisticRegression(max_iter=500)
regression.fit(X_train,Y_train)
print("executed sucessfully")
```

```
executed sucessfully
```

In [15]:
```
y_predict1 = regression.predict(X_test)
```

In [16]:
```
dp = pd.DataFrame({"Actual":Y_test,"predicted":y_predict1})
dp
```

Out[16]:

	Actual	predicted
0	25	30
1	24	30

evaluating the linear regression

In [17]:

```
from sklearn import metrics
print('Mean Absolute Error:',
      metrics.mean_absolute_error(Y_test, y_predict))
```

```
Mean Absolute Error: 3.4102305612975066
```

evaluating the logistic regression

In [18]:

```
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(Y_test,y_predict1))
```

```
Mean Absolute Error: 5.5
```

Conclusion:

Based on the above two model metrics linear regression performs well in this case