```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv("Scores.csv")
df
```

```
    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
19    7.4      69
20    2.7      30
21    4.8      54
22    3.8      35
23    6.9      76
24    7.8      86
```

```
df.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
```

```
#Write down the interpretation
```

```
df.describe()
```

```
           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
```

```
#Write down the interpretation
```

```
def getgrade(mark):
    if mark>=75:
        return "A"
    elif mark>=60 and mark<75:
        return "B"
    elif mark>=35 and mark<60:
        return "C"

    else:
        return "F"
```

```
df["Grade"]=df["Scores"].apply(getgrade)
```

```
df
```

```
    Hours  Scores Grade
0     2.5      21     F
1     5.1      47     C
2     3.2      27     F
3     8.5      75     A
4     3.5      30     F
5     1.5      20     F
6     9.2      88     A
7     5.5      60     B
8     8.3      81     A
9     2.7      25     F
10    7.7      85     A
11    5.9      62     B
12    4.5      41     C
13    3.3      42     C
```

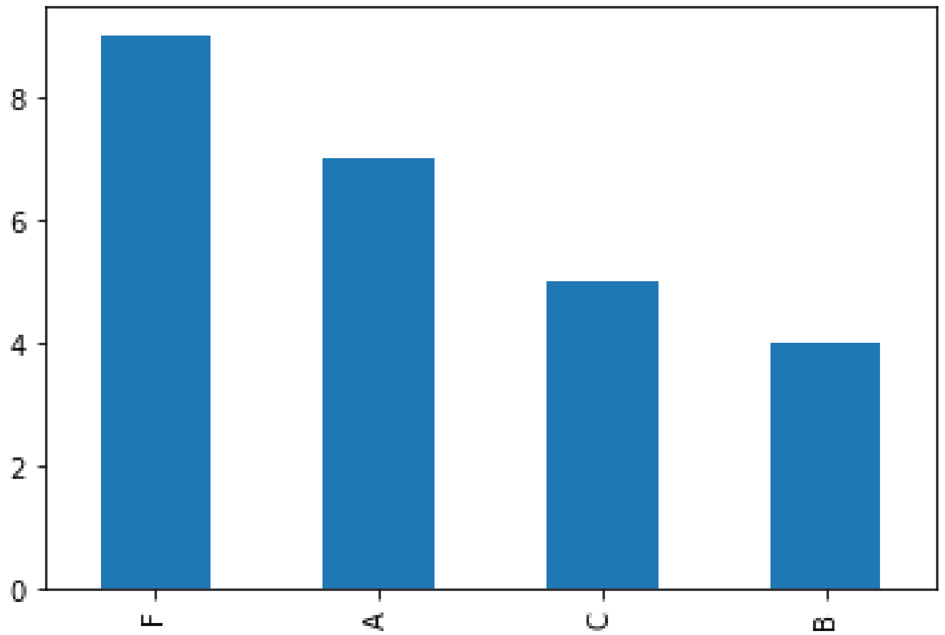
```
14     1.1       17      F
15     8.9       95      A
16     2.5       30      F
17     1.9       24      F
18     6.1       67      B
19     7.4       69      B
20     2.7       30      F
21     4.8       54      C
22     3.8       35      C
23     6.9       76      A
24     7.8       86      A
```

```
df["Grade"].value_counts()
```

```
F    9
A    7
C    5
B    4
Name: Grade, dtype: int64
```

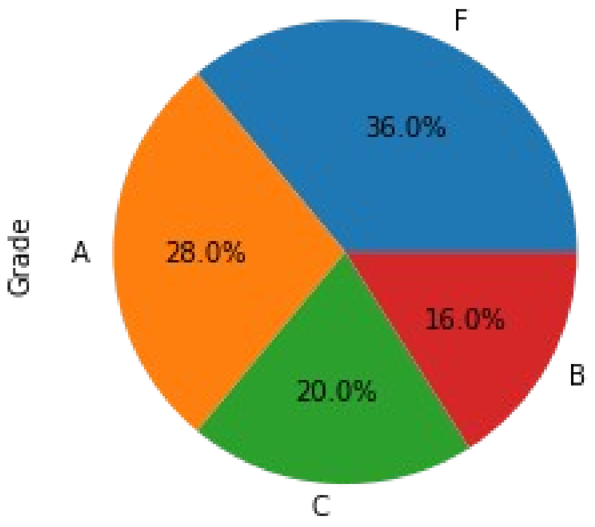
```
df["Grade"].value_counts().plot(kind="bar")
```

```
<AxesSubplot: >
```



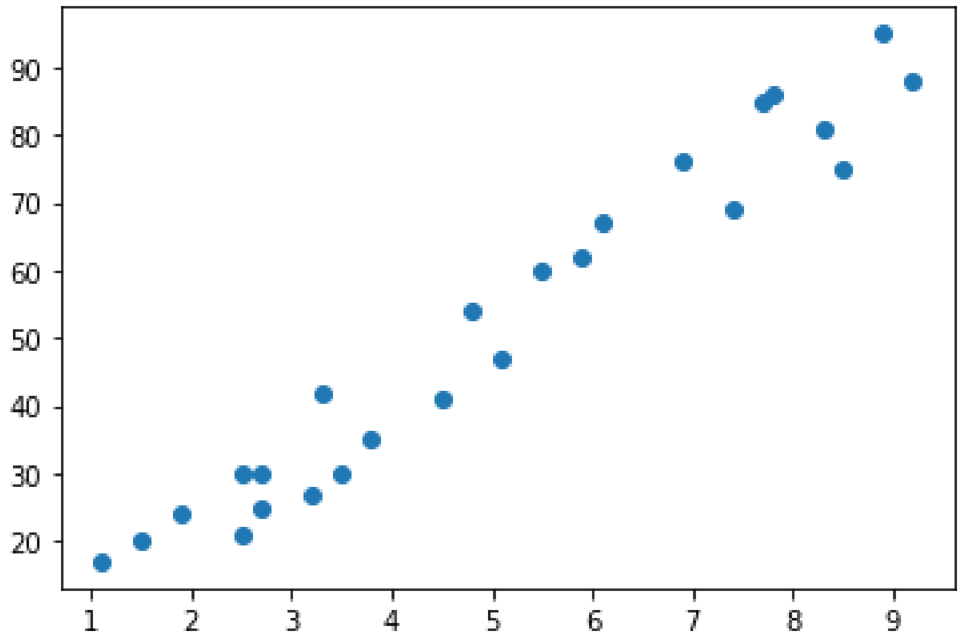
```
df["Grade"].value_counts().plot(kind="pie",autopct="%1.1f%%")
```

```
<AxesSubplot: ylabel='Grade'>
```



```
plt.scatter(df["Hours"],df["Scores"])
```

```
<matplotlib.collections.PathCollection at 0x1623f6805e0>
```



```
df.corr()
```

```
           Hours    Scores
Hours   1.000000  0.976191
Scores  0.976191  1.000000
```

Separation of X and Y

```
df.head()
```

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

```
x = df.iloc[:,:-2]  #2D
y = df.iloc[:,-2]   #1D
```

```
x
```

```
    Hours
0     2.5
1     5.1
2     3.2
3     8.5
4     3.5
5     1.5
6     9.2
7     5.5
8     8.3
9     2.7
10    7.7
11    5.9
12    4.5
13    3.3
14    1.1
15    8.9
16    2.5
17    1.9
18    6.1
19    7.4
20    2.7
21    4.8
22    3.8
23    6.9
24    7.8
```

```
y
```

```
0      21
1      47
2      27
3      75
4      30
5      20
```

```
6     88
7     60
8     81
9     25
10    85
11    62
12    41
13    42
14    17
15    95
16    30
17    24
18    67
19    69
20    30
21    54
22    35
23    76
24    86
Name: Scores, dtype: int64
```

Train Test Split the Data

```python
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_st
ate=1)

xtrain
```

```
    Hours
4     3.5
2     3.2
20    2.7
6     9.2
7     5.5
22    3.8
1     5.1
16    2.5
0     2.5
15    8.9
24    7.8
23    6.9
9     2.7
8     8.3
12    4.5
11    5.9
5     1.5
```

```
xtest
```

```
    Hours
14    1.1
13    3.3
17    1.9
3     8.5
21    4.8
10    7.7
18    6.1
19    7.4
```

Linear Reg Model

```
#Step1 :- Import the Model
from sklearn.linear_model import LinearRegression

#Step2:- Create Object Of The Model
linreg = LinearRegression()

#Step3:- Train The Model -->m and c
linreg.fit(xtrain,ytrain)

#Step4:- Make Prediction
ypred = linreg.predict(xtest)
```

```
linreg.coef_
```

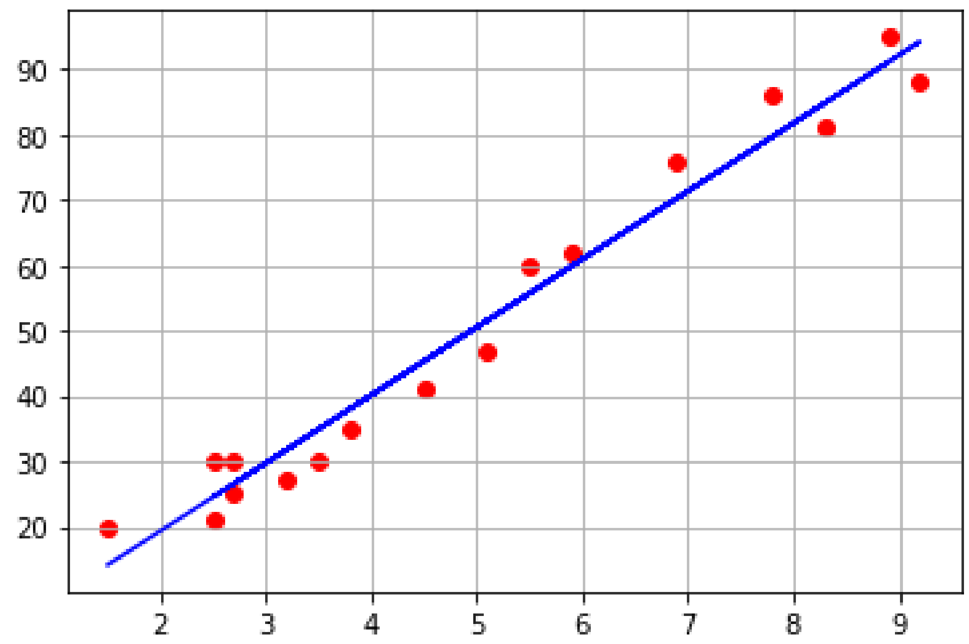
```
array([10.41075981])
```

```
linreg.intercept_
```

```
-1.5123061161277889
```

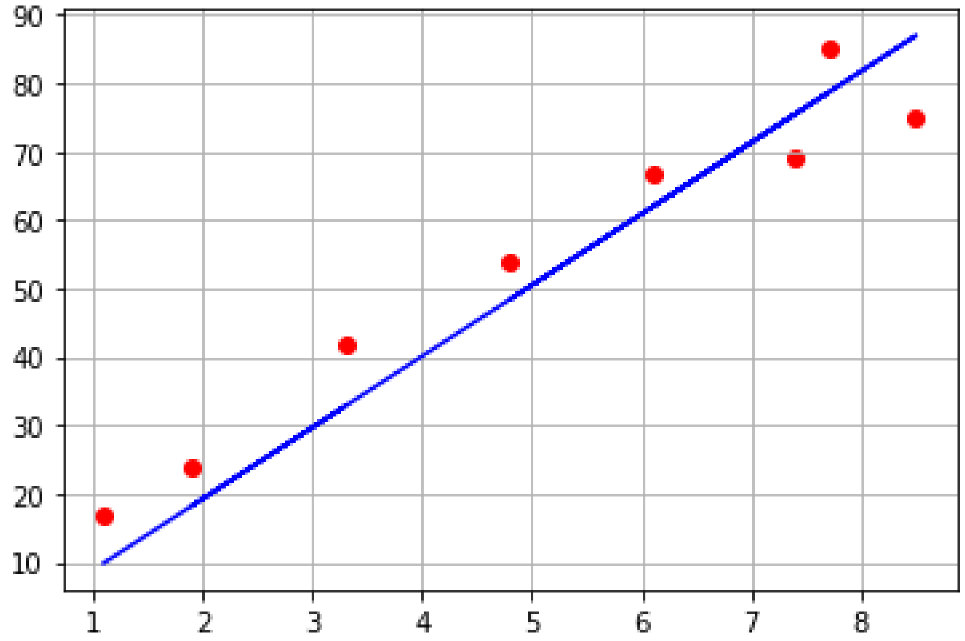
Model Performing on The Training Set

```
plt.scatter(xtrain,ytrain,color="red")
plt.plot(xtrain,linreg.predict(xtrain),color="blue")
plt.grid()
plt.show()
```



Model Performing on The Testing Set

```
plt.scatter(xtest,ytest,color="red")
plt.plot(xtest,linreg.predict(xtest),color="blue")
plt.grid()
plt.show()
```



Model EValuation

```
from sklearn.metrics import
mean_absolute_error,mean_squared_error,r2_score

mae = mean_absolute_error(ytest,ypred)
mse = mean_squared_error(ytest,ypred)
rmse = np.sqrt(mse)

r2 = r2_score(ytest,ypred)

print(f"MAE:- {mae}\n MSE:- {mse}\n RMSE:- {rmse}\n Accuracy :- {r2}")
```

```
MAE:- 7.169048271425507
 MSE:- 56.092330905646705
 RMSE:- 7.489481350911204
 Accuracy :- 0.8933827573294114
```

Model Testing On New Obsercvation

```
newob = 5
linreg.predict([[newob]])
```

```
array([50.54149294])
```

```
def makeprediction():
    newob = float(input("Enter No of Hrs Of Study:-  "))
    yp = linreg.predict([[newob]])[0]
    print(f"If You Study of {newob} hrs, you will score arround
{yp:.2f} marks")
    return print(f"{yp:.2f}")

makeprediction()
```

```
Enter No of Hrs Of Study:-  4
If You Study of 4.0 hrs, you will score arround 40.13 marks
40.13
```