```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('https://raw.githubusercontent.com/arib168/data/main/50_Startups.csv')
df
```

1 to 25 of 50 entries Filter

index	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.2	136897.8	471784.1	New York	192261.83
1	162597.7	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94
5	131876.9	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.6
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.4
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.9
19	86419.7	153514.11	0.0	New York	122776.86
20	76253.86	113867.3	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04

Show 25 per page 1 2

Like what you see? Visit the data table notebook to learn more about interactive tables.

```
df.head()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

```
df.tail()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0.00	116983.80	45173.06	California	14681.40

```
df.dtypes
```

```
R&D Spend          float64
Administration     float64
Marketing Spend    float64
State               object
Profit             float64
dtype: object
```

Find missing values

```
df.isna().sum()
```

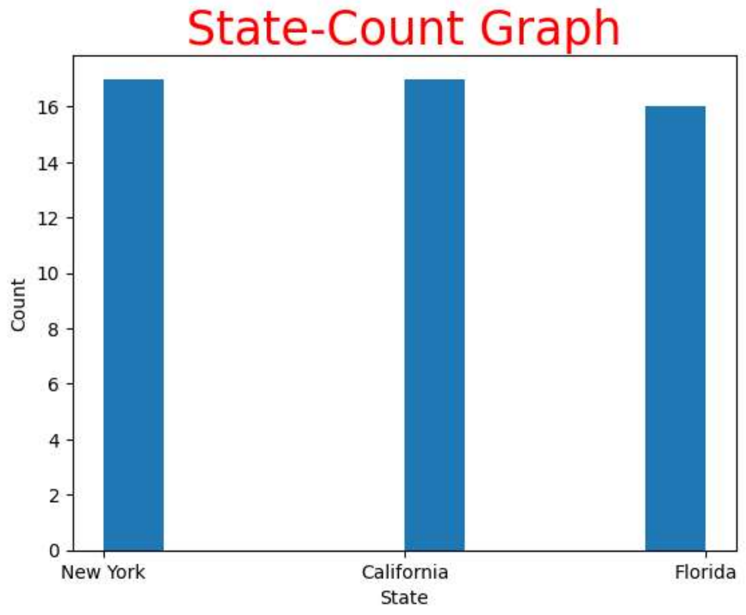
```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

```
df['State'].value_counts()
```

```
New York      17
California    17
Florida       16
Name: State, dtype: int64
```

```
plt.hist(df['State'])
plt.xlabel('State')
plt.ylabel('Count')
plt.title('State-Count Graph',color='red',size=25)
```

```
Text(0.5, 1.0, 'State-Count Graph')
```

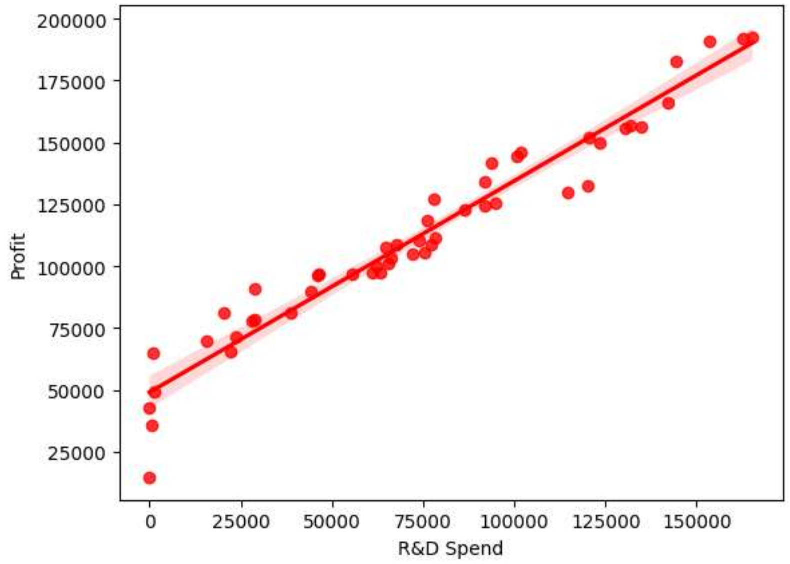


Seperate input and output

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

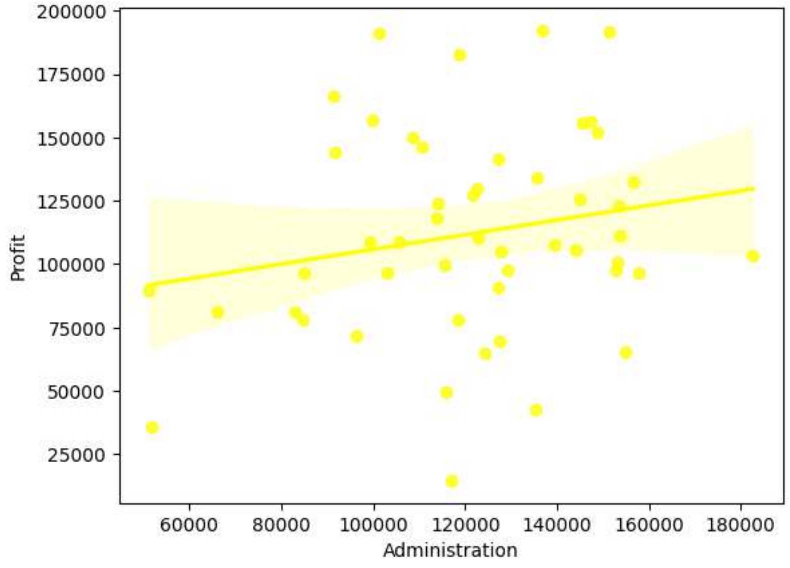
```
sns.regplot(x=df['R&D Spend'],y=y,color='red')
```

```
<Axes: xlabel='R&D Spend', ylabel='Profit'>
```



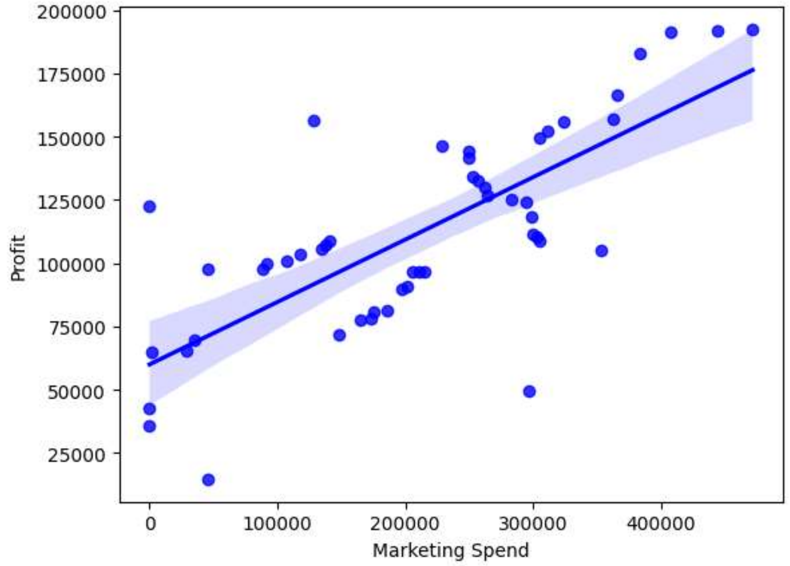
```
sns.regplot(x=df['Administration'],y=y,color='yellow')
```

```
<Axes: xlabel='Administration', ylabel='Profit'>
```




```
sns.regplot(x=df['Marketing Spend'],y=y,color='blue')
```

```
<Axes: xlabel='Marketing Spend', ylabel='Profit'>
```



One Hot Encoding

```
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder
col_trans=make_column_transformer((OneHotEncoder(handle_unknown='ignore'),['State']),remainder='passthrough')
x=col_trans.fit_transform(x)
x
```

```
array([[0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.6534920e+05,
        1.3689780e+05, 4.7178410e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.6259770e+05,
        1.5137759e+05, 4.4389853e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.5344151e+05,
        1.0114555e+05, 4.0793454e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.4437241e+05,
        1.1867185e+05, 3.8319962e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.4210734e+05,
        9.1391770e+04, 3.6616842e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.3187690e+05,
        9.9814710e+04, 3.6286136e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.3461546e+05,
        1.4719887e+05, 1.2771682e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.3029813e+05,
        1.4553006e+05, 3.2387668e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.2054252e+05,
        1.4871895e+05, 3.1161329e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.2333488e+05,
        1.0867917e+05, 3.0498162e+05],
```

```
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.0191308e+05,
        1.1059411e+05, 2.2916095e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 1.0067196e+05,
        9.1790610e+04, 2.4974455e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.3863750e+04,
        1.2732038e+05, 2.4983944e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 9.1992390e+04,
        1.3549507e+05, 2.5266493e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.1994324e+05,
        1.5654742e+05, 2.5651292e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.1452361e+05,
        1.2261684e+05, 2.6177623e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 7.8013110e+04,
        1.2159755e+05, 2.6434606e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 9.4657160e+04,
        1.4507758e+05, 2.8257431e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.1749160e+04,
        1.1417579e+05, 2.9491957e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 8.6419700e+04,
        1.5351411e+05, 0.0000000e+00],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 7.6253860e+04,
        1.1386730e+05, 2.9866447e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.8389470e+04,
        1.5377343e+05, 2.9973729e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.3994560e+04,
        1.2278275e+05, 3.0331926e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.7532530e+04,
        1.0575103e+05, 3.0476873e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.7044010e+04,
        9.9281340e+04, 1.4057481e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 6.4664710e+04,
        1.3955316e+05, 1.3796262e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.5328870e+04,
        1.4413598e+05, 1.3405007e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 7.2107600e+04,
        1.2786455e+05, 3.5318381e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.6051520e+04,
        1.8264556e+05, 1.1814820e+05],
```

Splitting to Training and Testing

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
x_train
x_test
```

```
    array([[1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 9.1992390e+04,
        1.3549507e+05, 2.5266493e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 3.8558510e+04,
        8.2982090e+04, 1.7499930e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 6.1994480e+04,
        1.1564128e+05, 9.1131240e+04],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.0002300e+03,
        1.2415304e+05, 1.9039300e+03],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 9.4657160e+04,
        1.4507758e+05, 2.8257431e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 5.4205000e+02,
        5.1743150e+04, 0.0000000e+00],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 7.5328870e+04,
        1.4413598e+05, 1.3405007e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 6.4664710e+04,
        1.3955316e+05, 1.3796262e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 6.3408860e+04,
        1.2921961e+05, 4.6085250e+04],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 8.6419700e+04,
        1.5351411e+05, 0.0000000e+00],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 9.3863750e+04,
        1.2732038e+05, 2.4983944e+05],
       [0.0000000e+00, 1.0000000e+00, 0.0000000e+00, 1.4210734e+05,
        9.1391770e+04, 3.6616842e+05],
       [1.0000000e+00, 0.0000000e+00, 0.0000000e+00, 4.4069950e+04,
        5.1283140e+04, 1.9702942e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.2054252e+05,
        1.4871895e+05, 3.1161329e+05],
       [0.0000000e+00, 0.0000000e+00, 1.0000000e+00, 1.4437241e+05,
        1.1867185e+05, 3.8319962e+05]])
```

Model Creation

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

```
array([126187.39411505,  85788.82259512,  99777.02815178,  45706.12238329,
       127062.20722772,  51891.83884459, 109114.62977495, 100600.61123702,
        97953.99874716, 111730.5770681 , 128818.49200668, 174195.35772631,
        93736.28538438, 148381.0409716 , 172313.87139388])
```

```
df1=pd.DataFrame({'Actual_value':y_test,'Pred_value':y_pred,'Error':y_test-y_pred})
df1
```

	Actual_value	Pred_value	Error
13	134307.35	126187.394115	8119.955885
39	81005.76	85788.822595	-4783.062595
30	99937.59	99777.028152	160.561848
45	64926.08	45706.122383	19219.957617
17	125370.37	127062.207228	-1691.837228
48	35673.41	51891.838845	-16218.428845
26	105733.54	109114.629775	-3381.089775
25	107404.34	100600.611237	6803.728763
32	97427.84	97953.998747	-526.158747
19	122776.86	111730.577068	11046.282932
12	141585.52	128818.492007	12767.027993
4	166187.94	174195.357726	-8007.417726
37	89949.14	93736.285384	-3787.145384
8	152211.77	148381.040972	3830.729028
3	182901.99	172313.871394	10588.118606

Performance Evaluation

MAE, MSE, RMSE, R2 Score

```
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error,mean_squared_error,r2_score
print('Mean absolute Error is',mean_absolute_error(y_test,y_pred))
print('Error percentage is',mean_absolute_percentage_error(y_test,y_pred))
print('Mean Squared Error is',mean_squared_error(y_test,y_pred))
root=mean_squared_error(y_test,y_pred)
print('Root mean squared error is',np.sqrt(root))
print('r2 score is',r2_score(y_test,y_pred))
```

```
Mean absolute Error is 7395.433531521974
Error percentage is 0.08929865344172414
Mean Squared Error is 84826955.03529756
Root mean squared error is 9210.154995183173
r2 score is 0.9397108063356046
```