```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline

import warnings
```

Import Dataset

In [2]:
```
data = pd.read_csv('50_Startups.csv')
```

Out[2]:

	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

In [ ]:
```
```

To find information about dataset

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

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

Out[5]:
```
(50, 5)
```

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

Out[6]:
```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profit'],
dtype='object')
```

In [7]:
```
data.describe()
```

Out[7]:

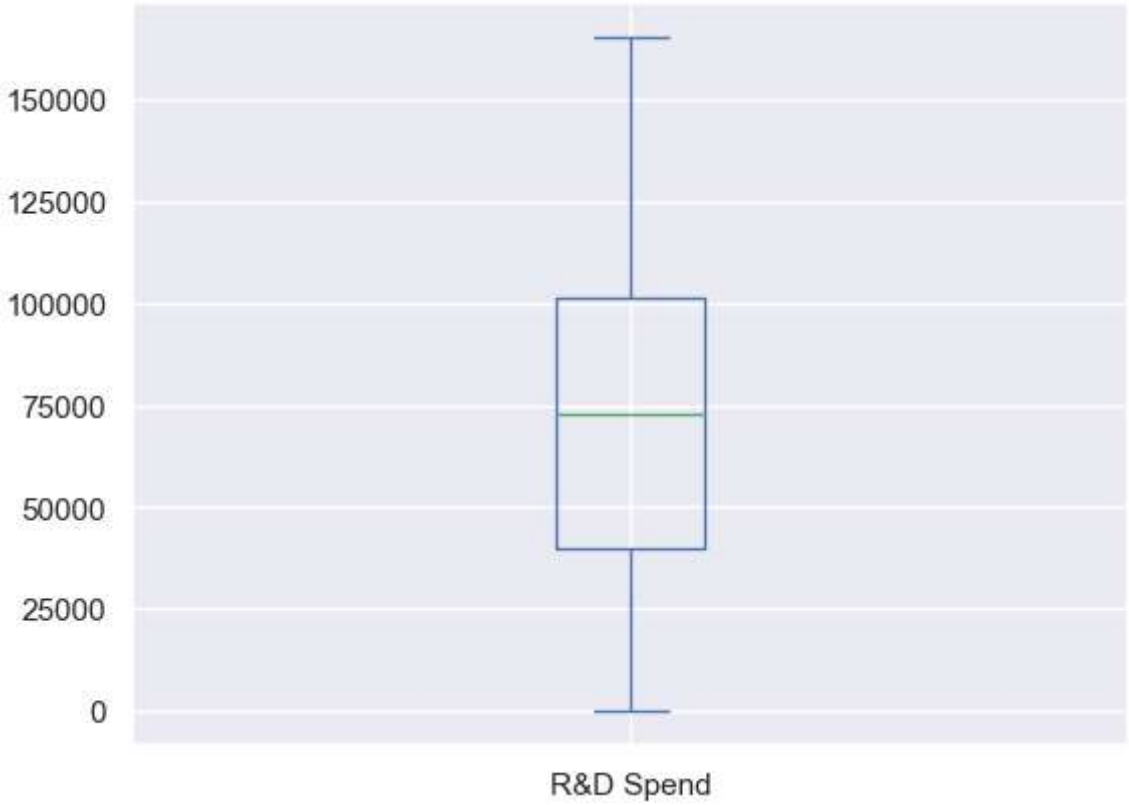
	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000

In [ ]:
```
```

check outliers

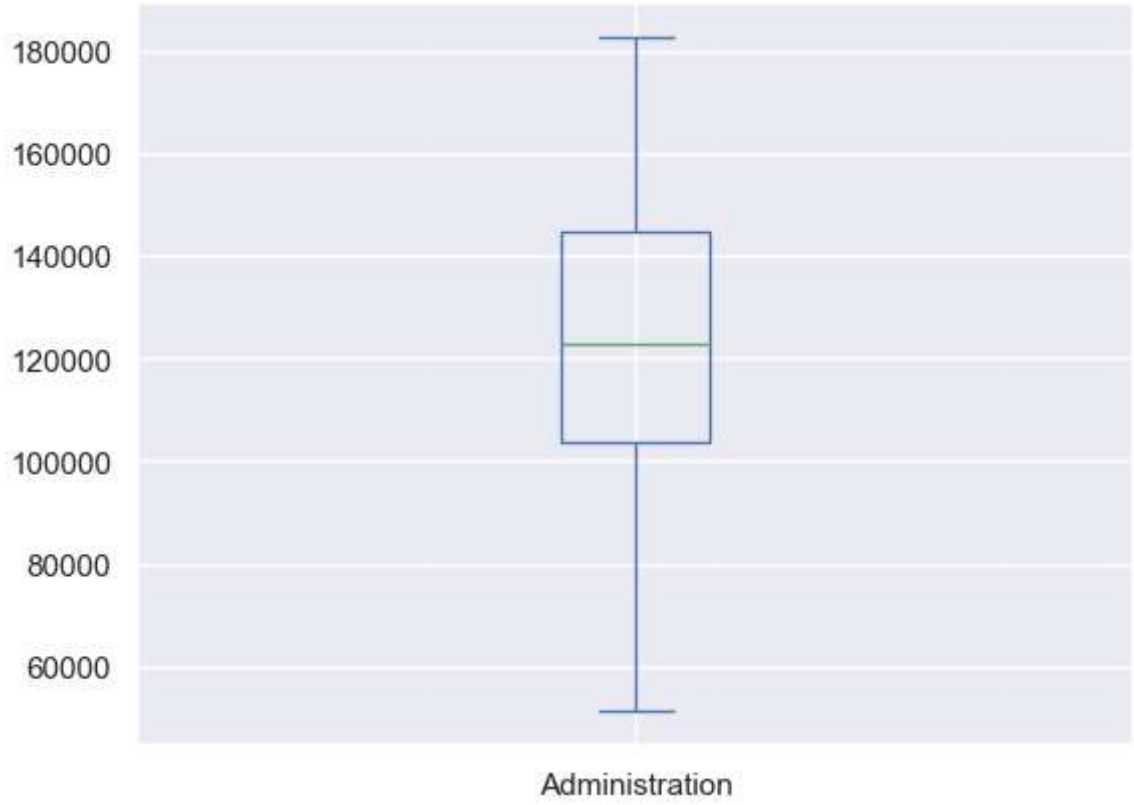
In [8]:
```
data['R&D Spend'].plot(kind='box')
```

Out[8]: <Axes: >



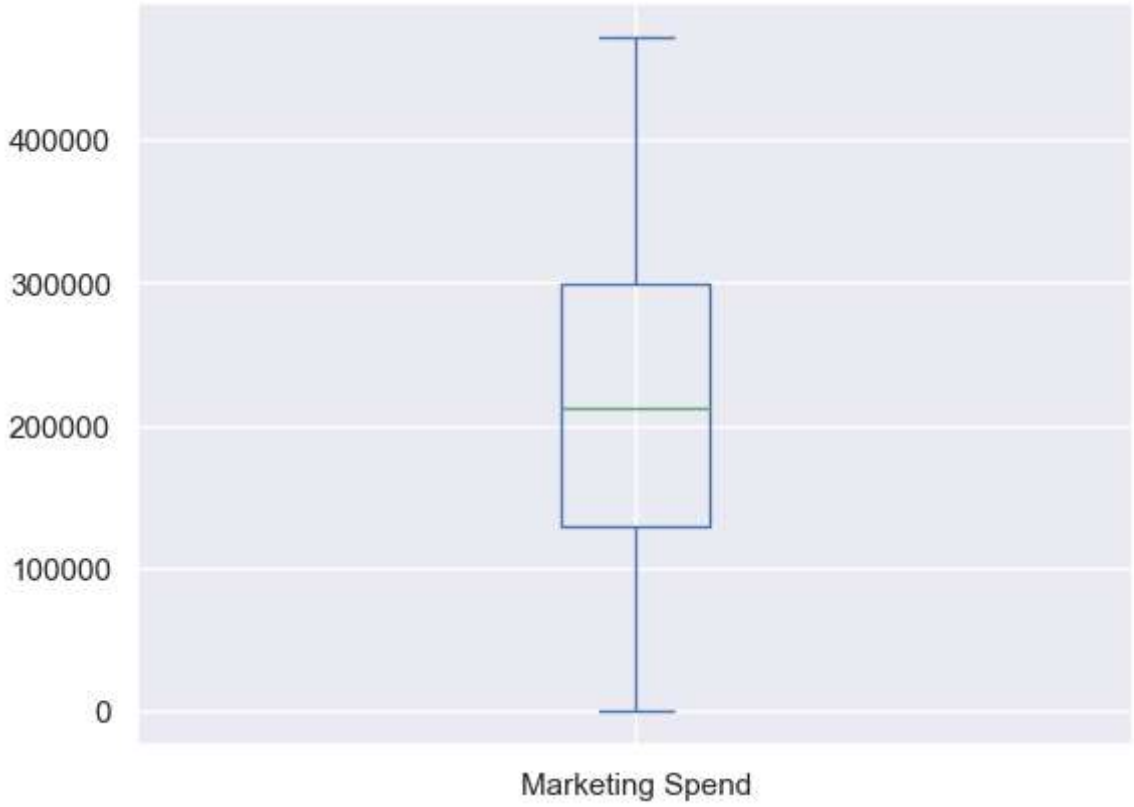
In [9]:
```
data['Administration'].plot(kind='box')
```

Out[9]: <Axes: >



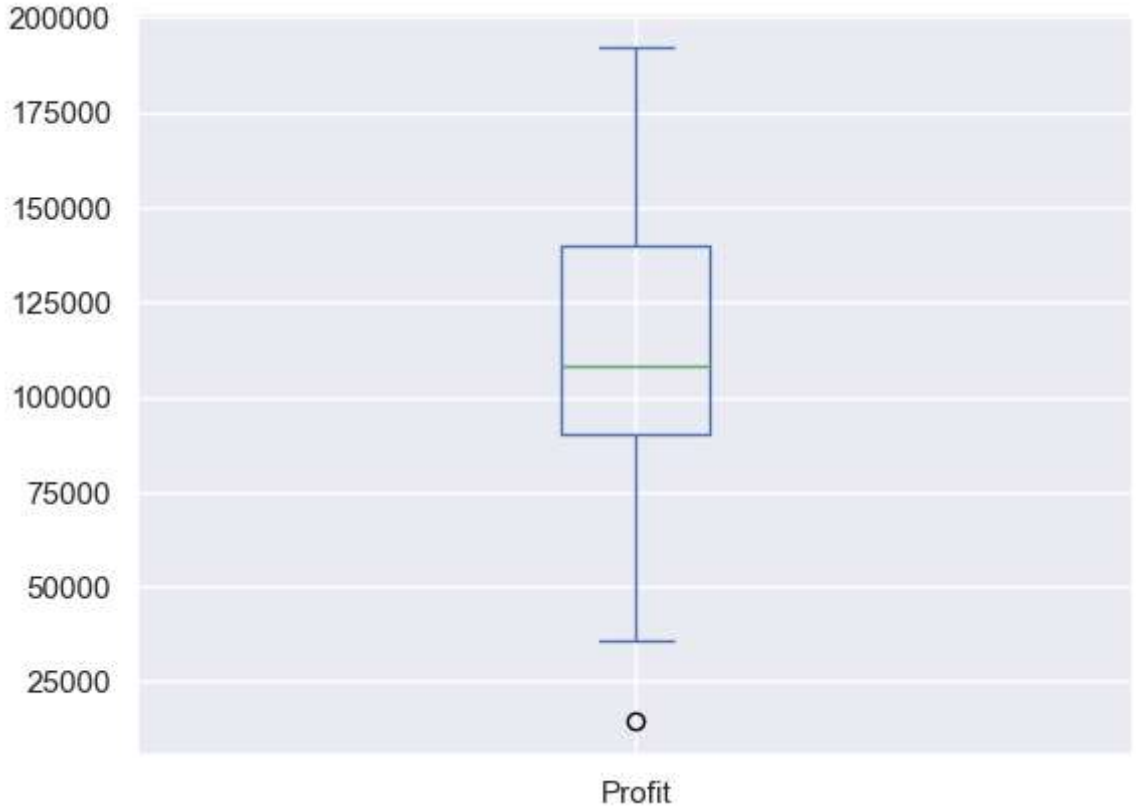
In [10]:
```
data['Marketing Spend'].plot(kind='box')
```

Out[10]: <Axes: >



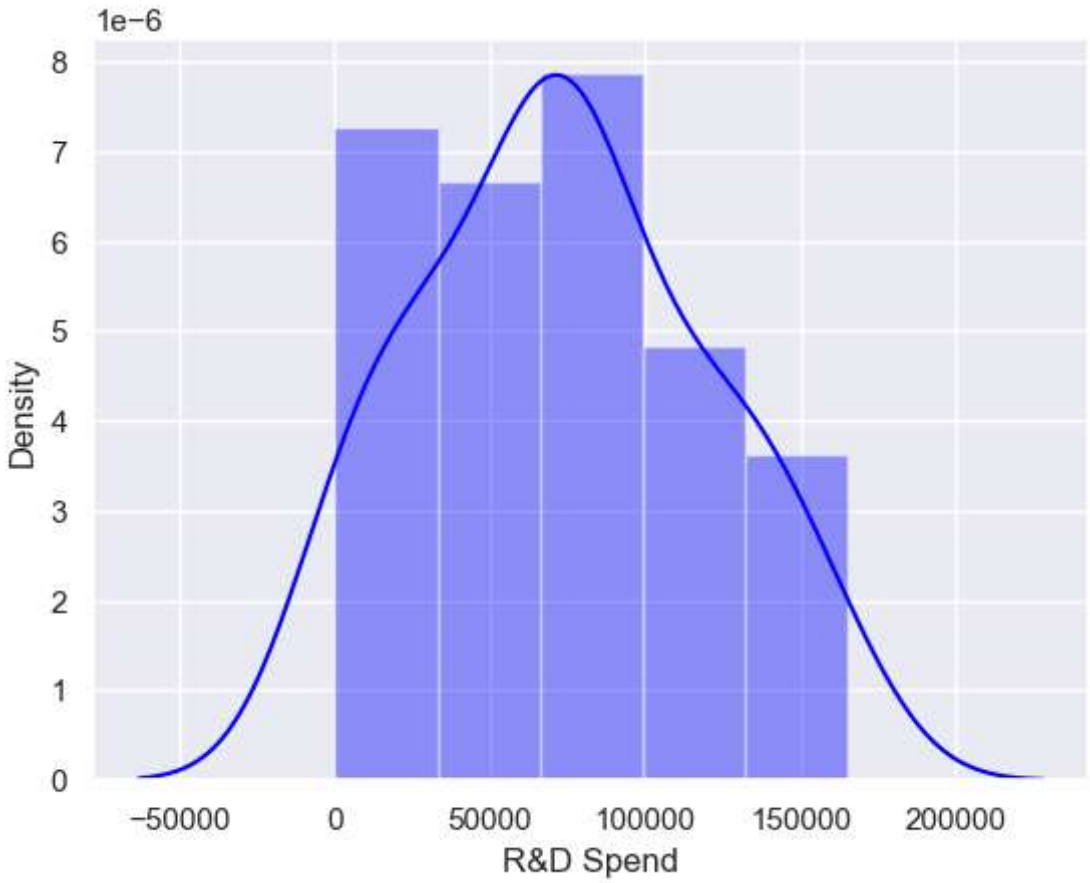
In [11]:
```
data['Profit'].plot(kind='box')
```

Out[11]: <Axes: >



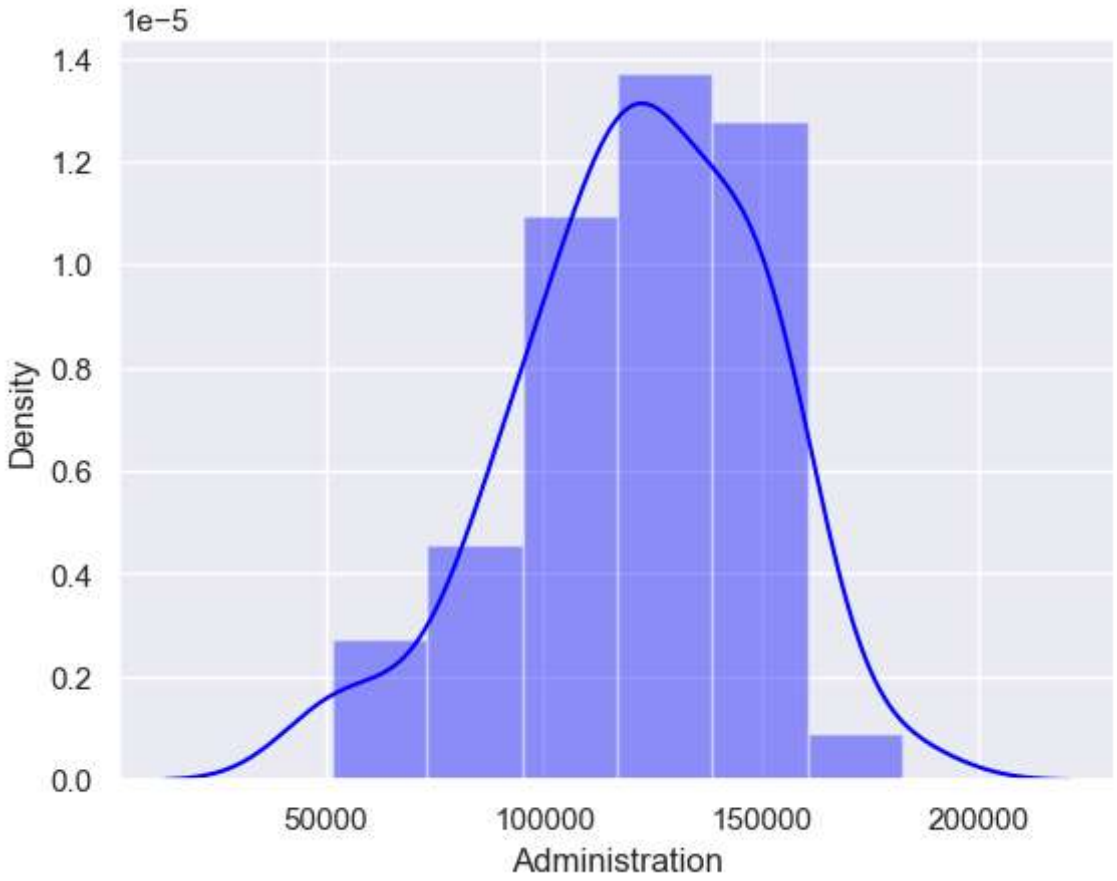
In [12]:

Out[12]: <Axes: xlabel='R&D Spend', ylabel='Density'>



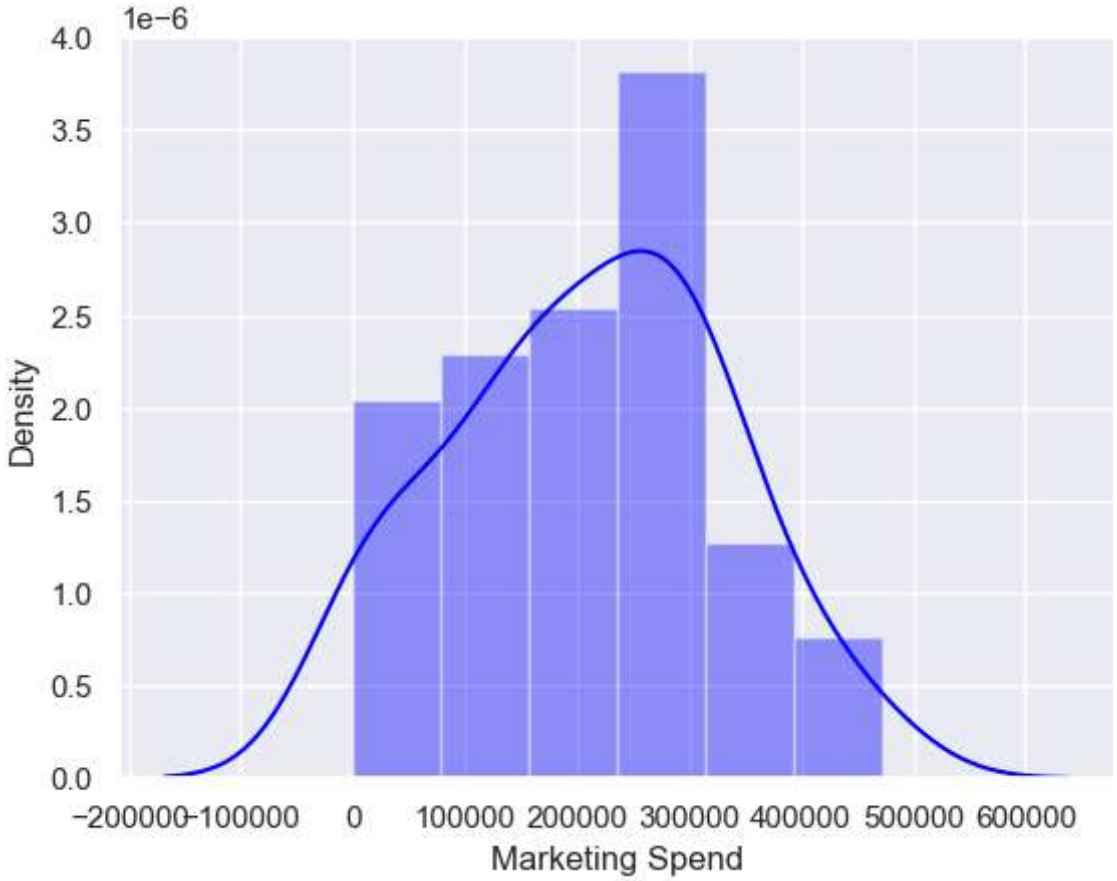
In [13]:
```
sns.distplot(data['Administration'],color='blue')
```

Out[13]: <Axes: xlabel='Administration', ylabel='Density'>



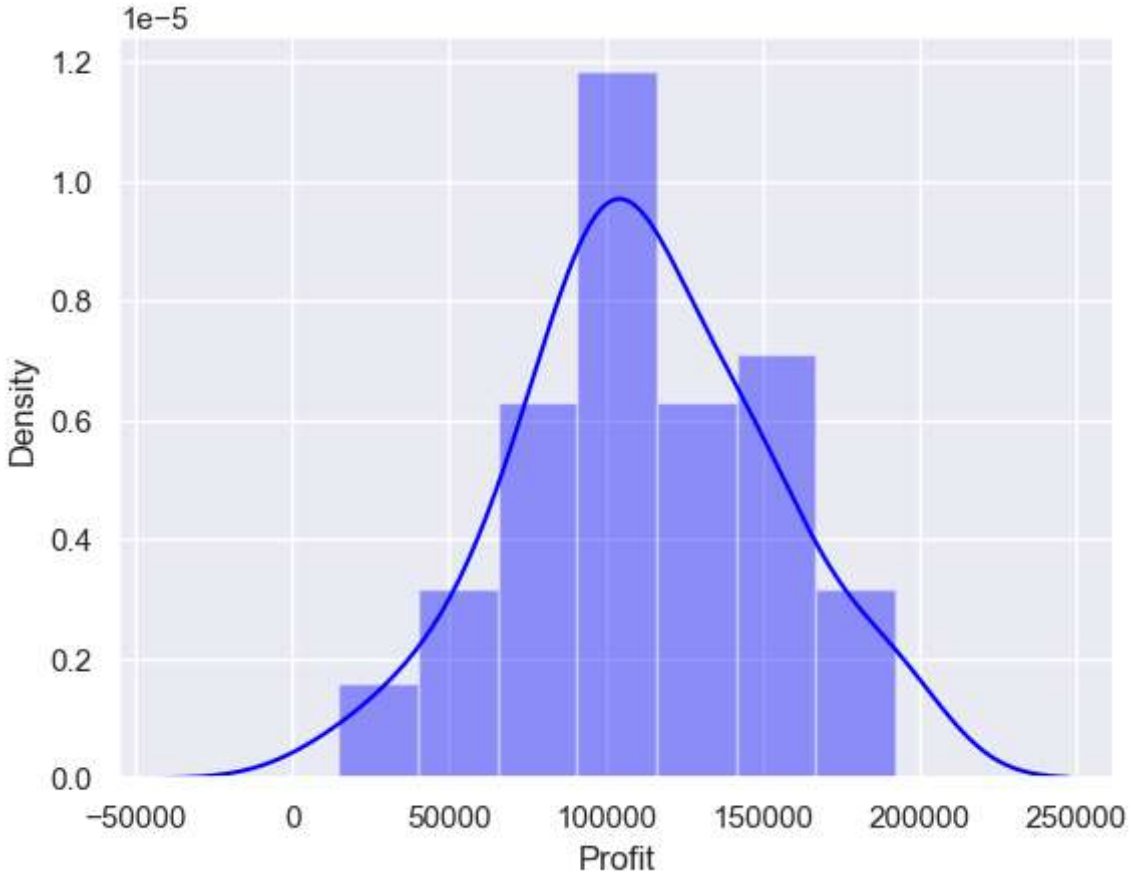
In [14]:
```
sns.distplot(data['Marketing Spend'],color='blue')
```

Out[14]:
```
<Axes: xlabel='Marketing Spend', ylabel='Density'>
```

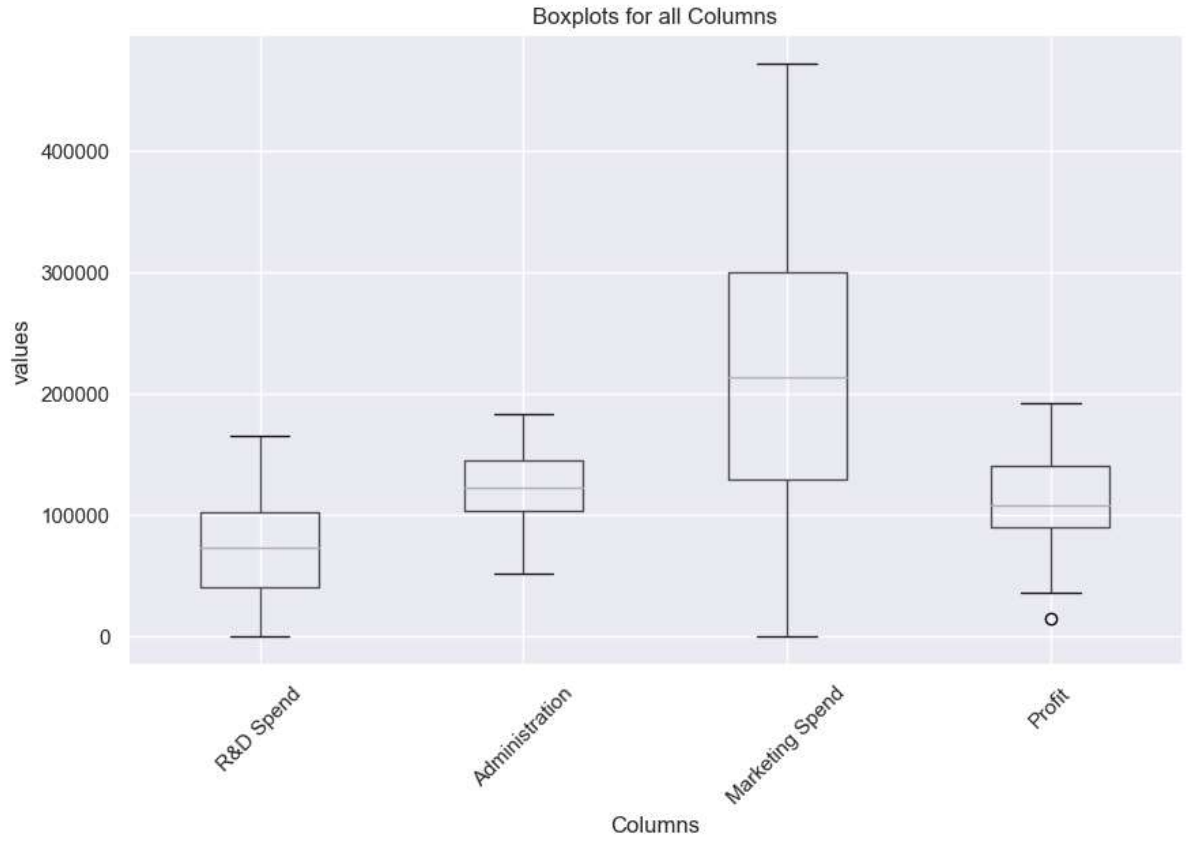


In [15]:
```
sns.distplot(data['Profit'],color='blue')
```

Out[15]: <Axes: xlabel='Profit', ylabel='Density'>



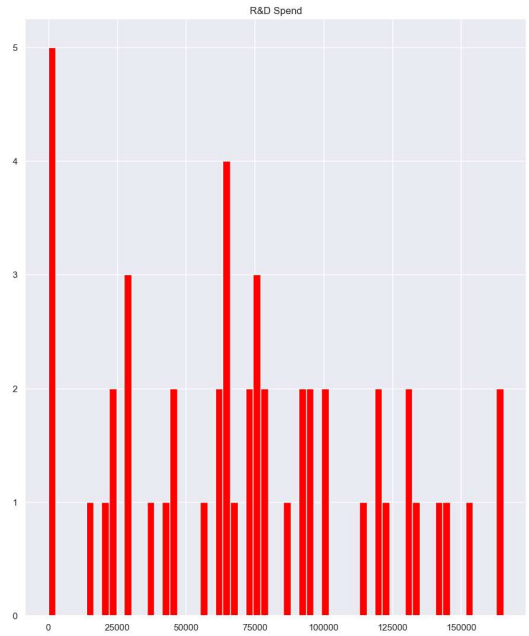
In [16]:
```
num_col = data.select_dtypes(exclude='object').columns.tolist()
plt.figure(figsize = (10,6))
data.boxplot(column = num_col)
plt.title('Boxplots for all Columns')
plt.xticks(rotation = 45)
plt.xlabel('Columns')
plt.ylabel('values')
plt.grid(True)
```

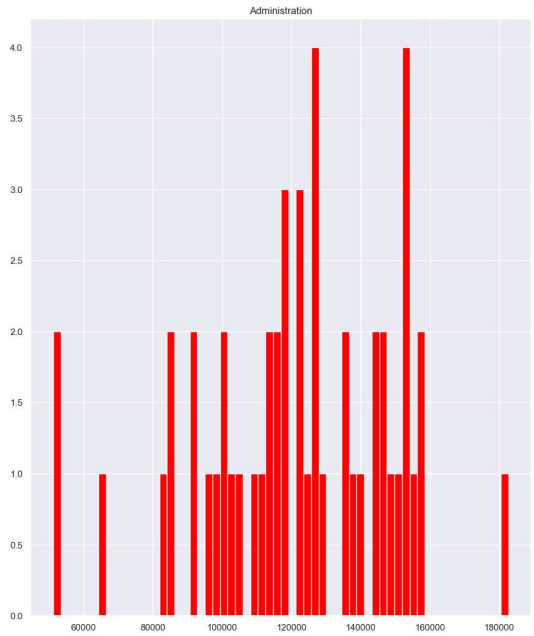


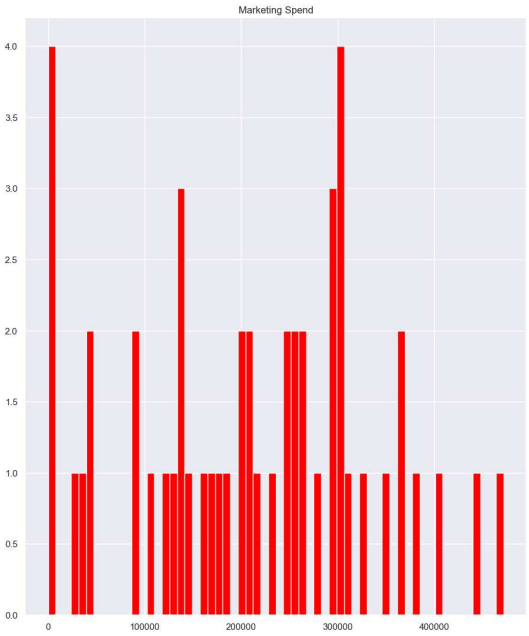
In [ ]:
```
```

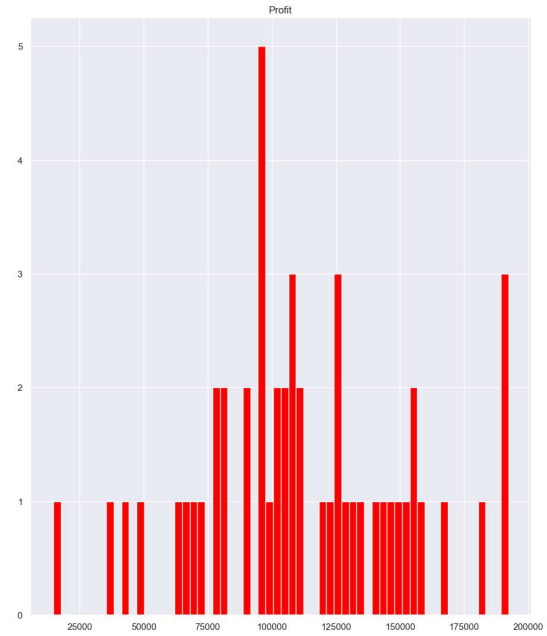
In [17]:
```
# no outliers in R&D Spend
# no outliers in Administration
# no outliers in Marketing Spend
```

In [18]:
```
data.hist(bins=60,figsize=(25,30),color='red')
```









In [19]:
```
for i in data.columns:
    print("**********************************************************",i,
          "**********************************************************")
    print()
    print(set(data[i].tolist()))
    print()
```

```
********************************************************** R&D Spend *********
*****************************************

{0.0, 66051.52, 20229.59, 119943.24, 73994.56, 15505.73, 86419.7, 64664.71, 1
01913.08, 142107.34, 38558.51, 542.05, 22177.74, 1315.46, 131876.9, 162597.7,
44069.95, 93863.75, 61994.48, 72107.6, 63408.86, 78389.47, 78013.11, 46014.0
2, 100671.96, 75328.87, 94657.16, 65605.48, 123334.88, 55493.95, 67532.53, 61
136.38, 28754.33, 134615.46, 91992.39, 23640.93, 46426.07, 114523.61, 76253.8
6, 120542.52, 153441.51, 165349.2, 91749.16, 1000.23, 144372.41, 77044.01, 27
892.92, 28663.76, 130298.13}

********************************************************** Administration ****
*************************************************

{110594.11, 156547.42, 135426.92, 108679.17, 144135.98, 91790.61, 118671.85,
103057.49, 118546.05, 152701.92, 127382.3, 105751.03, 101145.55, 65947.93, 12
2782.75, 51743.15, 139553.16, 91391.77, 82982.09, 153514.11, 153773.43, 15769
3.92, 145077.58, 154806.14, 85047.44, 115641.28, 96189.63, 136897.8, 129219.6
1, 135495.07, 153032.06, 113867.3, 127056.21, 151377.59, 99281.34, 51283.14,
127320.38, 99814.71, 84710.77, 115816.21, 148718.95, 127864.55, 182645.56, 11
6983.8, 122616.84, 124153.04, 145530.06, 121597.55, 147198.87, 114175.79}

********************************************************** Marketing Spend ***
*************************************************

{256512.92, 0.0, 304768.73, 107138.38, 118148.2, 46085.25, 294919.57, 249744.
55, 261776.23, 174999.3, 264346.06, 88218.23, 297114.46, 140574.81, 353183.8
1, 148001.11, 134050.07, 323876.68, 197029.42, 201126.82, 229160.95, 298664.4
7, 28334.72, 185265.1, 311613.29, 205517.64, 282574.31, 35534.17, 304981.62,
303319.26, 366168.42, 299737.29, 383199.62, 127716.82, 471784.1, 137962.62, 2
14634.81, 362861.36, 210797.67, 249839.44, 1903.93, 45173.06, 164470.71, 2526
64.93, 443898.53, 91131.24, 407934.54, 172795.67}

********************************************************** State ************
*****************************************

{'California', 'Florida', 'New York'}

********************************************************** Profit ************
*****************************************

{144259.4, 192261.83, 105733.54, 108552.04, 96778.92, 107404.34, 101004.64, 1
26992.93, 141585.52, 110352.25, 152211.77, 97427.84, 122776.86, 64926.08, 782
39.91, 134307.35, 166187.94, 191792.06, 105008.31, 65200.33, 125370.37, 10873
3.99, 156991.12, 42559.73, 96712.8, 146121.95, 191050.39, 118474.03, 97483.5
6, 81229.06, 71498.49, 111313.02, 49490.75, 90708.19, 35673.41, 156122.51, 14
681.4, 89949.14, 96479.51, 99937.59, 77798.83, 155752.6, 124266.9, 81005.76,
103282.38, 182901.99, 132602.65, 129917.04, 69758.98, 149759.96}
```

Data preprocessing

1) Missing value treatment

In [20]:
```
data.isnull().sum()/len(data)*100
```

Out[20]:
```
R&D Spend          0.0
Administration     0.0
Marketing Spend    0.0
State              0.0
Profit             0.0
dtype: float64
```

In [21]:
```
# there is no missing values
```

Encoding concept

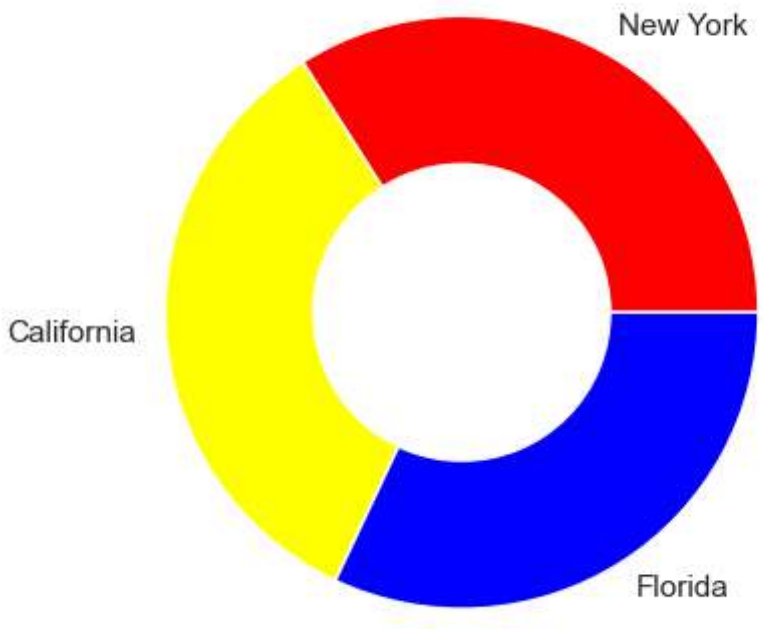
In [22]:
```
data['State'].value_counts()
```

Out[22]:
```
New York      17
California    17
Florida       16
Name: State, dtype: int64
```

In [ ]:
```

```

In [23]:
```
State=['New York','California','Florida']
quantity=[17,17,16]
plt.pie(quantity,labels=State,radius=1,colors=['red','yellow','blue'])
plt.pie([1],colors=['w'],radius=0.5)
```



In [24]:
```
#State = ['New York','Florida']
#quantity = [17,16]
#plt.pie(quantity,labels=State,autopct='%0.1f%%',colors=['yellow','blue'])
```

In [ ]:
```
```

In [25]:
```
# State is categorical feature
```

One Hot Encoder

In [26]:
```
data['State'].value_counts()
```

Out[26]:
```
New York      17
California    17
Florida       16
Name: State, dtype: int64
```

In [27]:
```
data=pd.get_dummies(data,columns=['State'])
```

In [28]:
```
data.head()
```

Out[28]:

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

In [29]:
```
#one  = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
#two  = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]
#three= [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
#data = list([one,two,three])
#plt.violinplot(data,showmedians = True)
```

In [30]:
```
data.head()
```

Out[30]:

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

In [ ]:
```

```

Dummy variable(n-1)

split data dependent and independent variables

In [31]:
```
x=data.drop(['Profit','State_California'],axis=1)
```

In [32]:
```
x.head()
```

Out[32]:

	R&D Spend	Administration	Marketing Spend	State_Florida	State_New York
0	165349.20	136897.80	471784.10	0	1
1	162597.70	151377.59	443898.53	0	0
2	153441.51	101145.55	407934.54	1	0
3	144372.41	118671.85	383199.62	0	1
4	142107.34	91391.77	366168.42	1	0

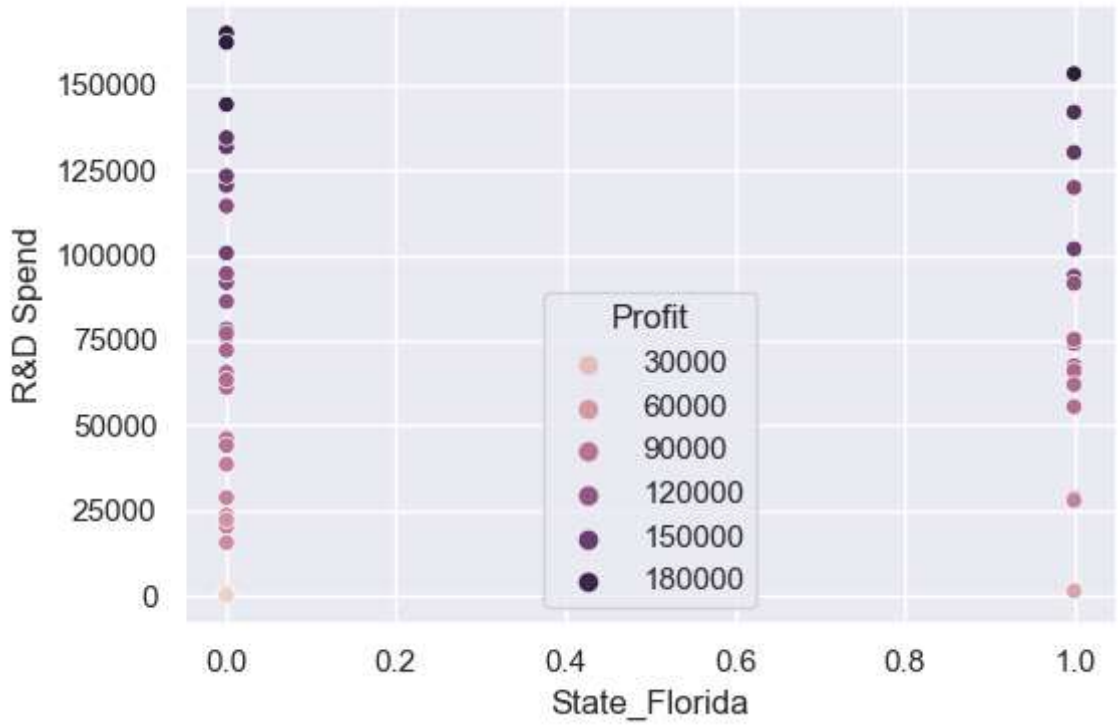
In [ ]:
```
```

In [33]:
```
y.head()
```

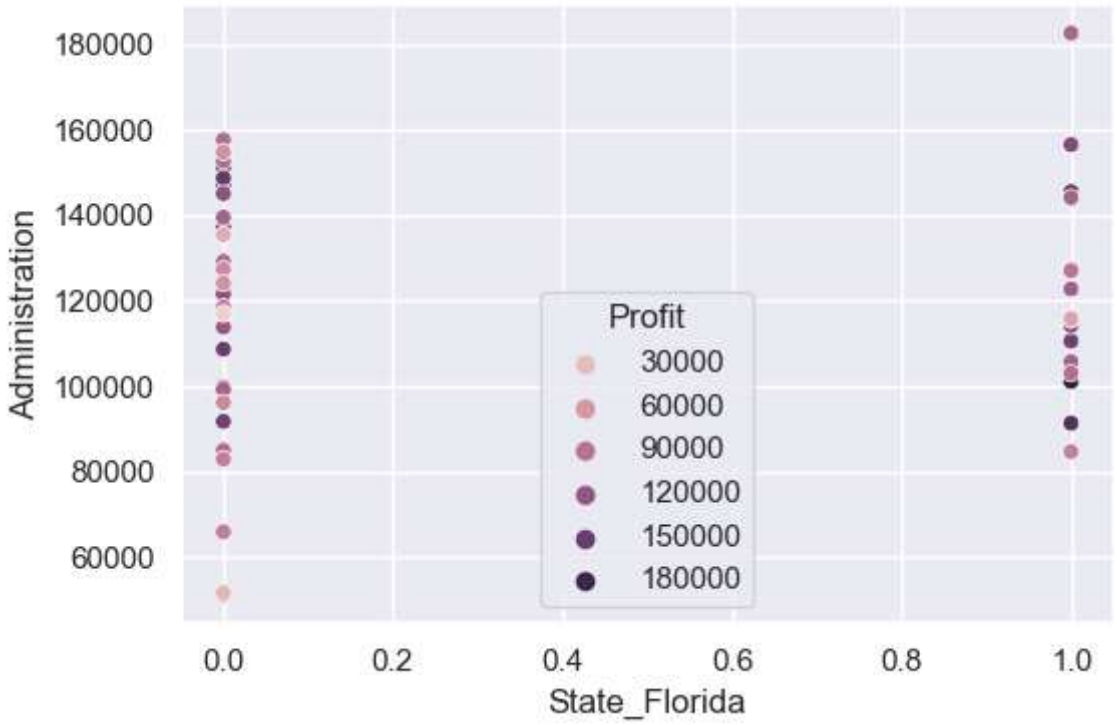
Out[33]:
```
0    192261.83
1    191792.06
2    191050.39
3    182901.99
4    166187.94
Name: Profit, dtype: float64
```

In [34]:
```
# correlation between State_Florida with R&D Spend,Administration,Marketing Sp
```

In [35]:
```
plt.figure(figsize=(6,4),dpi=100)
sns.scatterplot(x='State_Florida',y='R&D Spend',hue='Profit',data=data)
plt.xlabel('State_Florida')
plt.ylabel('R&D Spend')
```



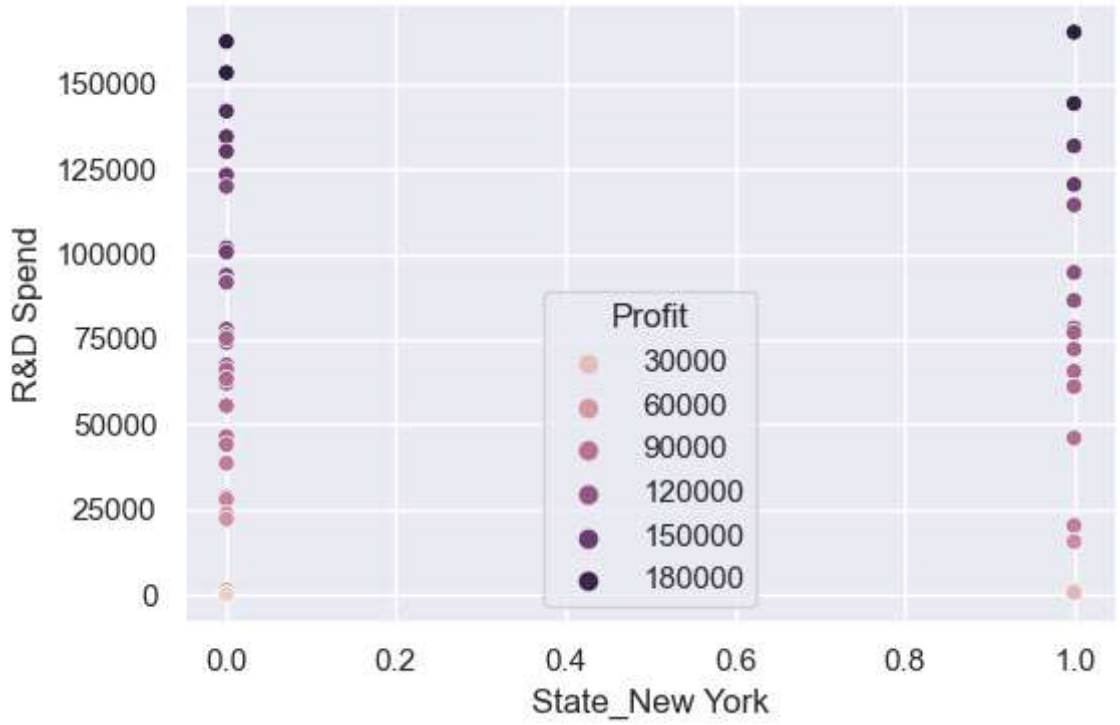
In [36]:
```
plt.figure(figsize=(6,4),dpi=100)
sns.scatterplot(x='State_Florida',y='Administration',hue='Profit',data=data)
plt.xlabel('State_Florida')
plt.ylabel('Administration')
```



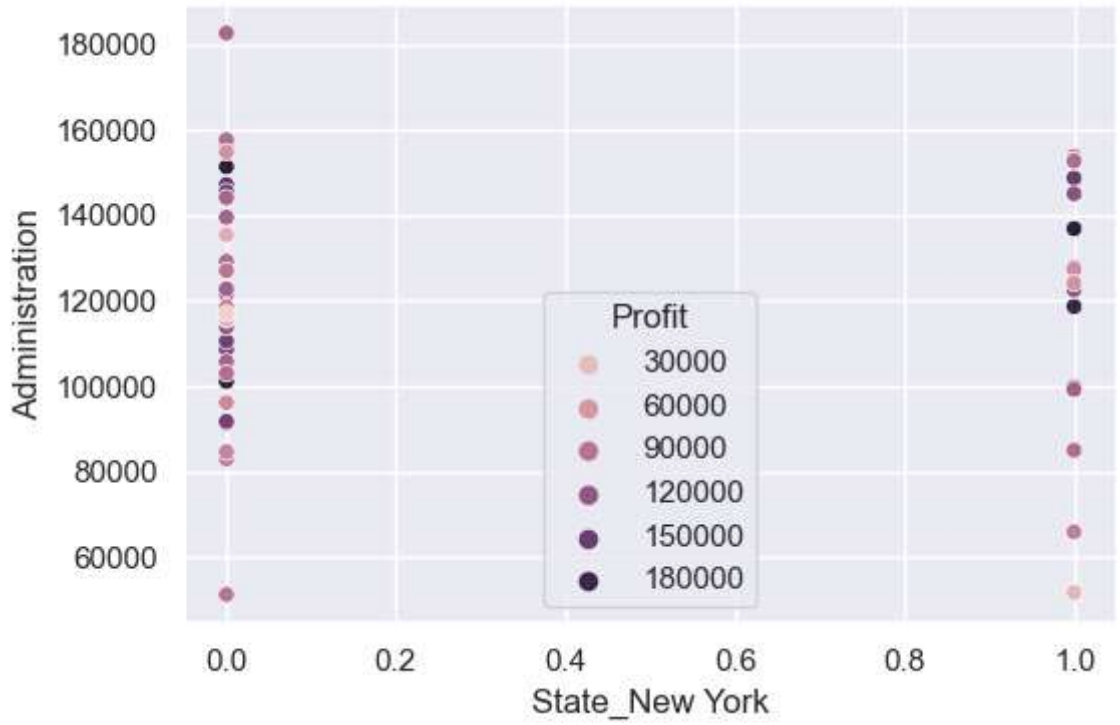
In [37]:
```
plt.figure(figsize=(6,4),dpi=100)
sns.scatterplot(x='State_Florida',y='Marketing Spend',hue='Profit',data=data)
plt.xlabel('State_Florida')
plt.ylabel('Marketing Spend')
```

In [38]:
```
# correlation between State_Florida with R&D Spend,Administration,Marketing Sp
```

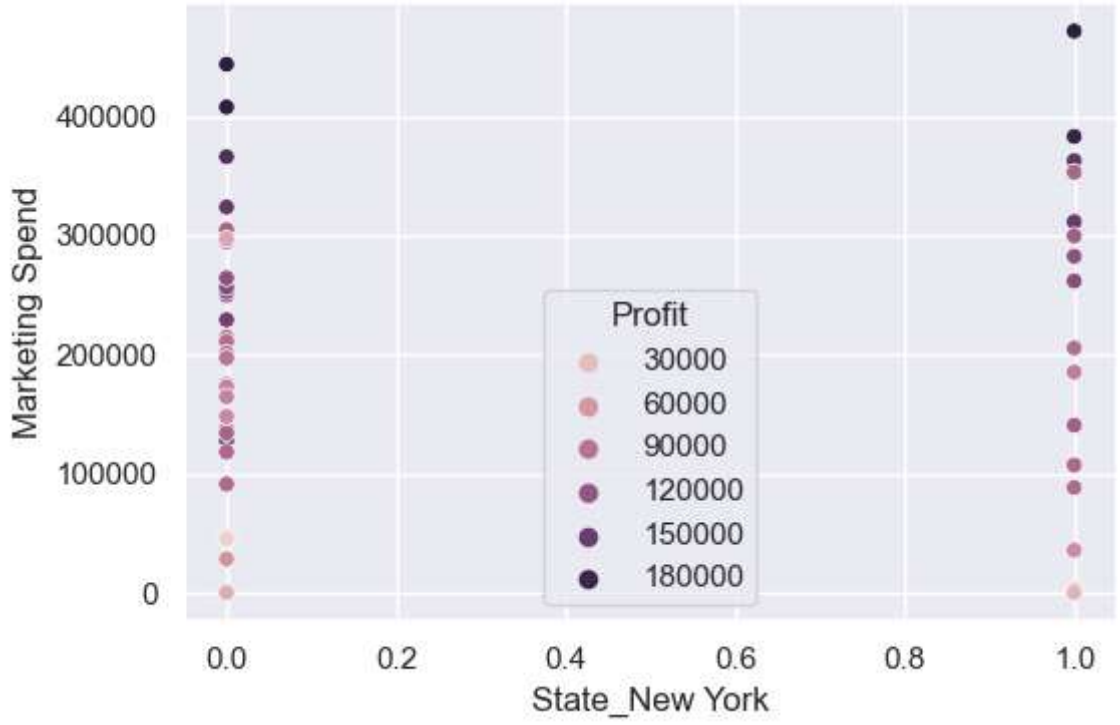
In [39]:
```
plt.figure(figsize=(6,4),dpi=100)
sns.scatterplot(x='State_New York',y='R&D Spend',hue='Profit',data=data)
plt.xlabel('State_New York')
plt.ylabel('R&D Spend')
```



In [40]:
```
plt.figure(figsize=(6,4),dpi=100)
sns.scatterplot(x='State_New York',y='Administration',hue='Profit',data=data)
plt.xlabel('State_New York')
plt.ylabel('Administration')
```

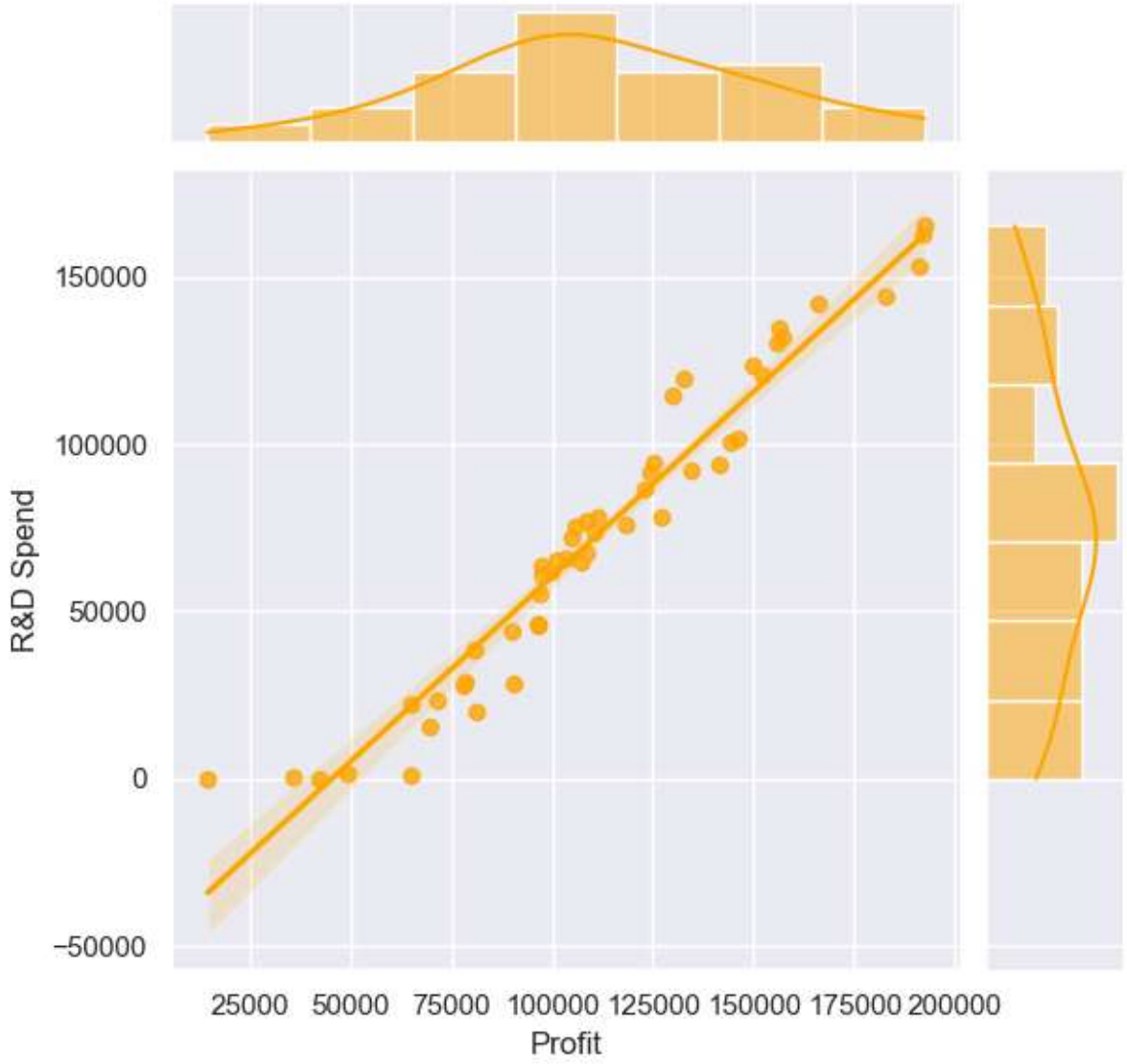


In [41]:
```
plt.figure(figsize=(6,4),dpi=100)
sns.scatterplot(x='State_New York',y='Marketing Spend',hue='Profit',data=data)
plt.xlabel('State_New York')
plt.ylabel('Marketing Spend')
```



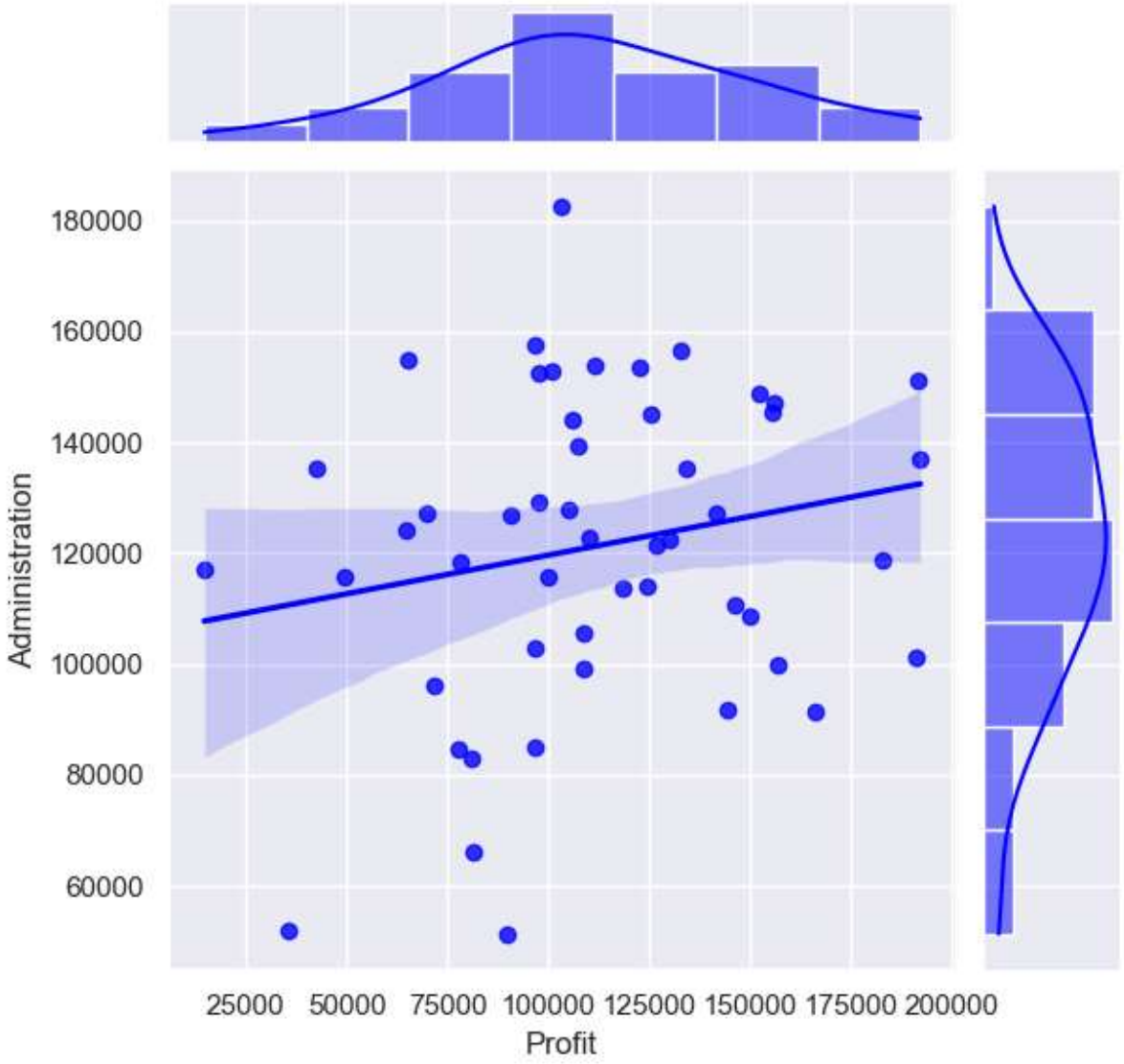
In [42]:
```
sns.jointplot(x='Profit',y='R&D Spend',data=data,kind='reg',color='orange')
```

Out[42]: <seaborn.axisgrid.JointGrid at 0x1912edeeaa0>



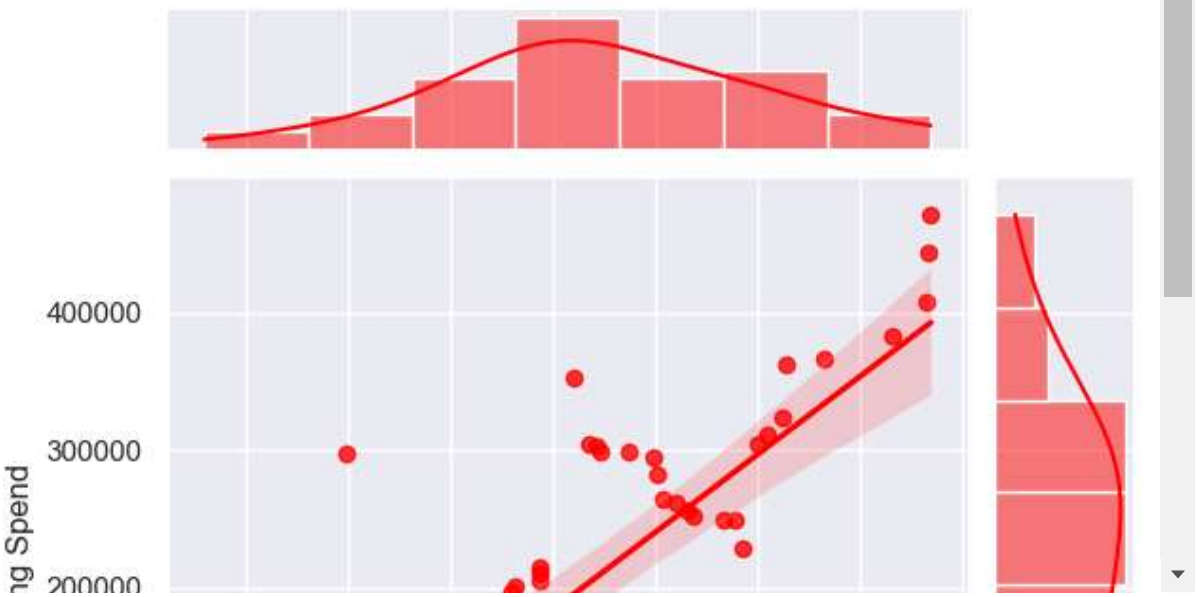
In [43]:
```
sns.jointplot(x='Profit',y='Administration',data=data,kind='reg',color='blue')
```

Out[43]: <seaborn.axisgrid.JointGrid at 0x1912eec6350>



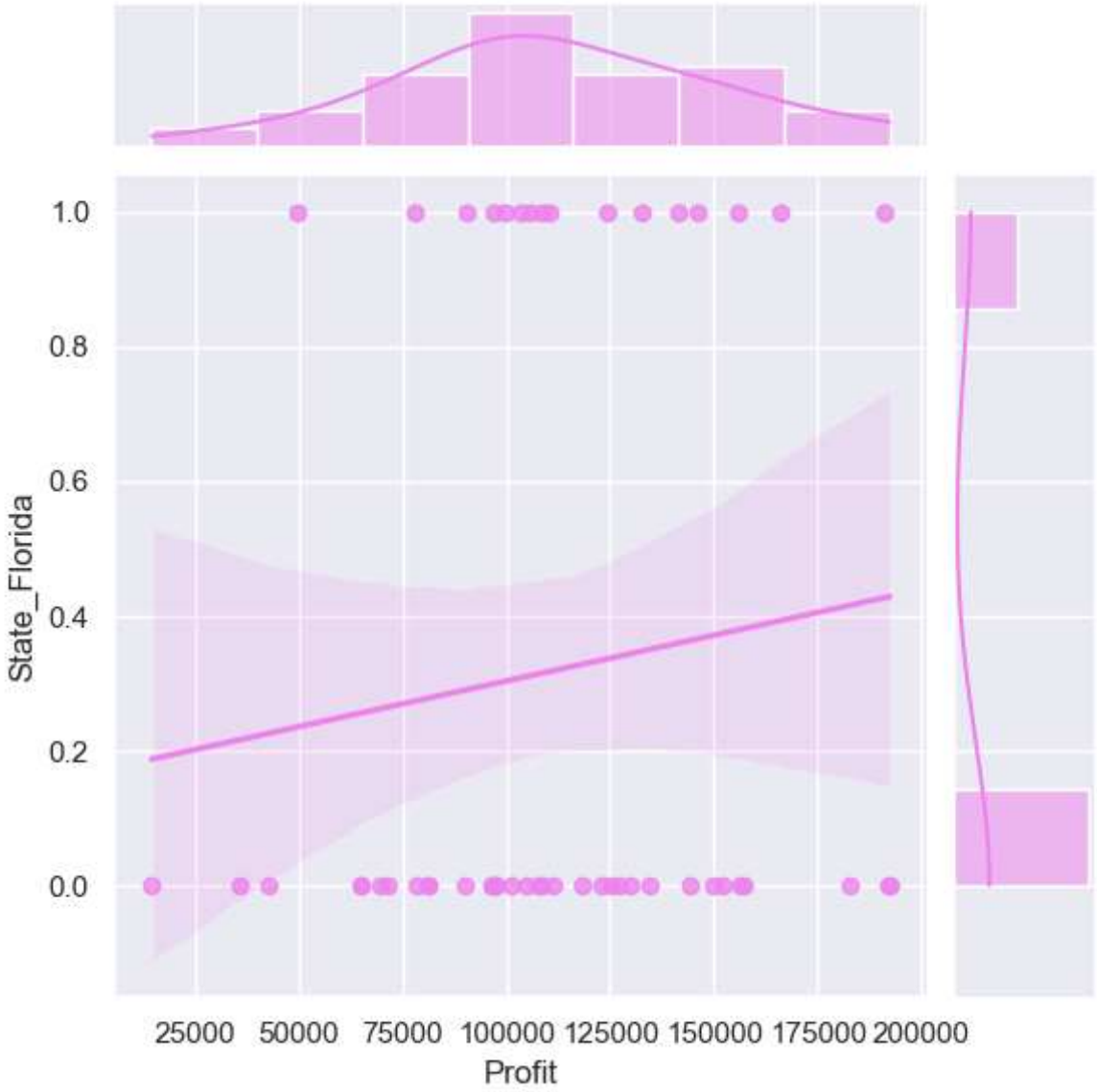
In [44]:
```
sns.jointplot(x='Profit',y='Marketing Spend',data=data,kind='reg',color='red')
```

Out[44]: <seaborn.axisgrid.JointGrid at 0x19130441780>

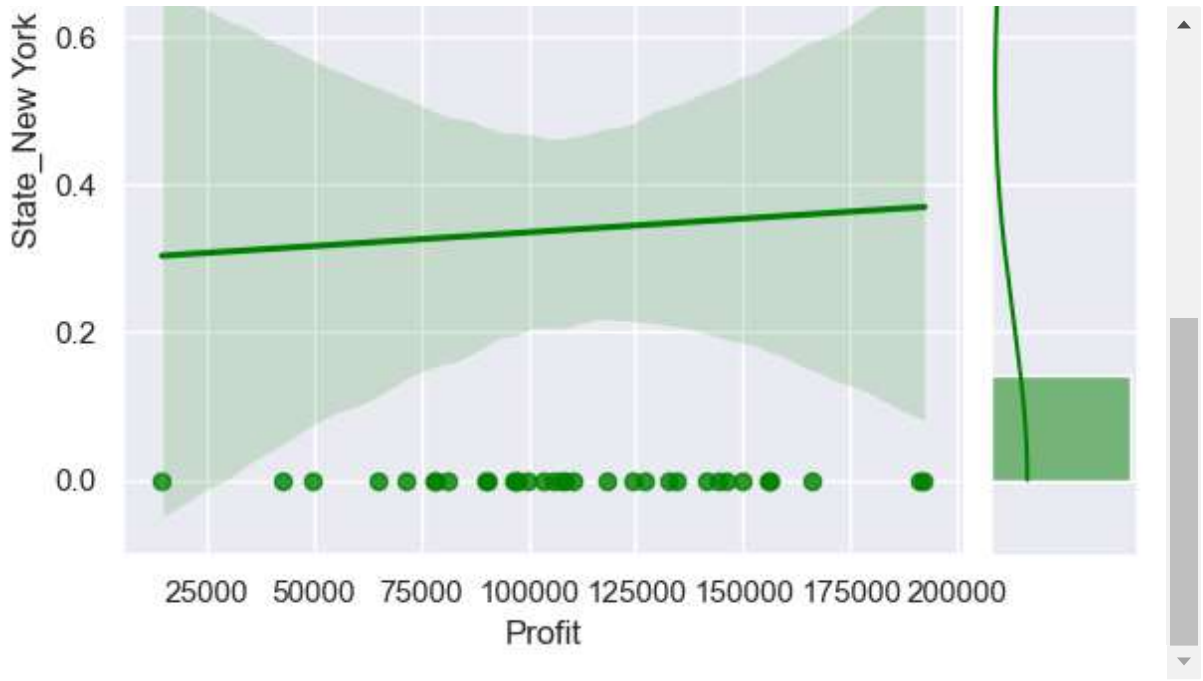


In [45]:
```
sns.jointplot(x='Profit',y='State_Florida',data=data,kind='reg',color='violet'
```

Out[45]: <seaborn.axisgrid.JointGrid at 0x1913088a4d0>

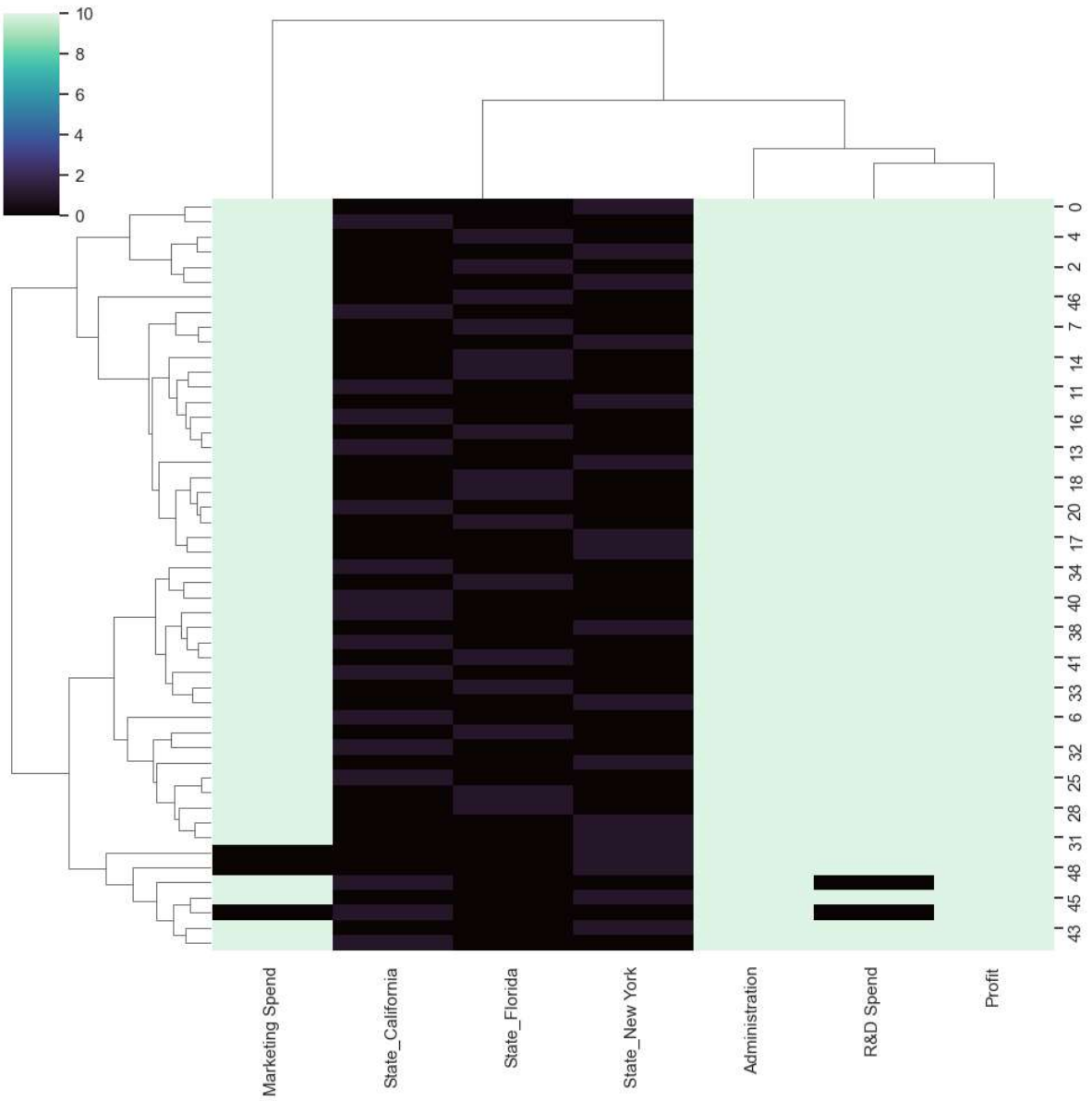


In [46]:
```
sns.jointplot(x='Profit',y='State_New York',data=data,kind='reg',color='green'
```



In [47]:

Out[47]: <seaborn.matrix.ClusterGrid at 0x19130e6a740>



Feature scaling

In [48]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder, PolynomialFeatu
from sklearn.linear_model import LinearRegression,Lasso, Ridge, ElasticNet,SGDR
from sklearn.metrics import r2_score,mean_absolute_percentage_error, mean_squa
```

In [49]:
```
x=data.drop(['Profit','State_California'],axis=1)
```

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

Out[50]:

	R&D Spend	Administration	Marketing Spend	State_Florida	State_New York
0	165349.20	136897.80	471784.10	0	1
1	162597.70	151377.59	443898.53	0	0
2	153441.51	101145.55	407934.54	1	0
3	144372.41	118671.85	383199.62	0	1
4	142107.34	91391.77	366168.42	1	0

In [51]:
```
y.head()
```

Out[51]:
```
0    192261.83
1    191792.06
2    191050.39
3    182901.99
4    166187.94
Name: Profit, dtype: float64
```

In [52]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc_x = sc.fit_transform(x)
```

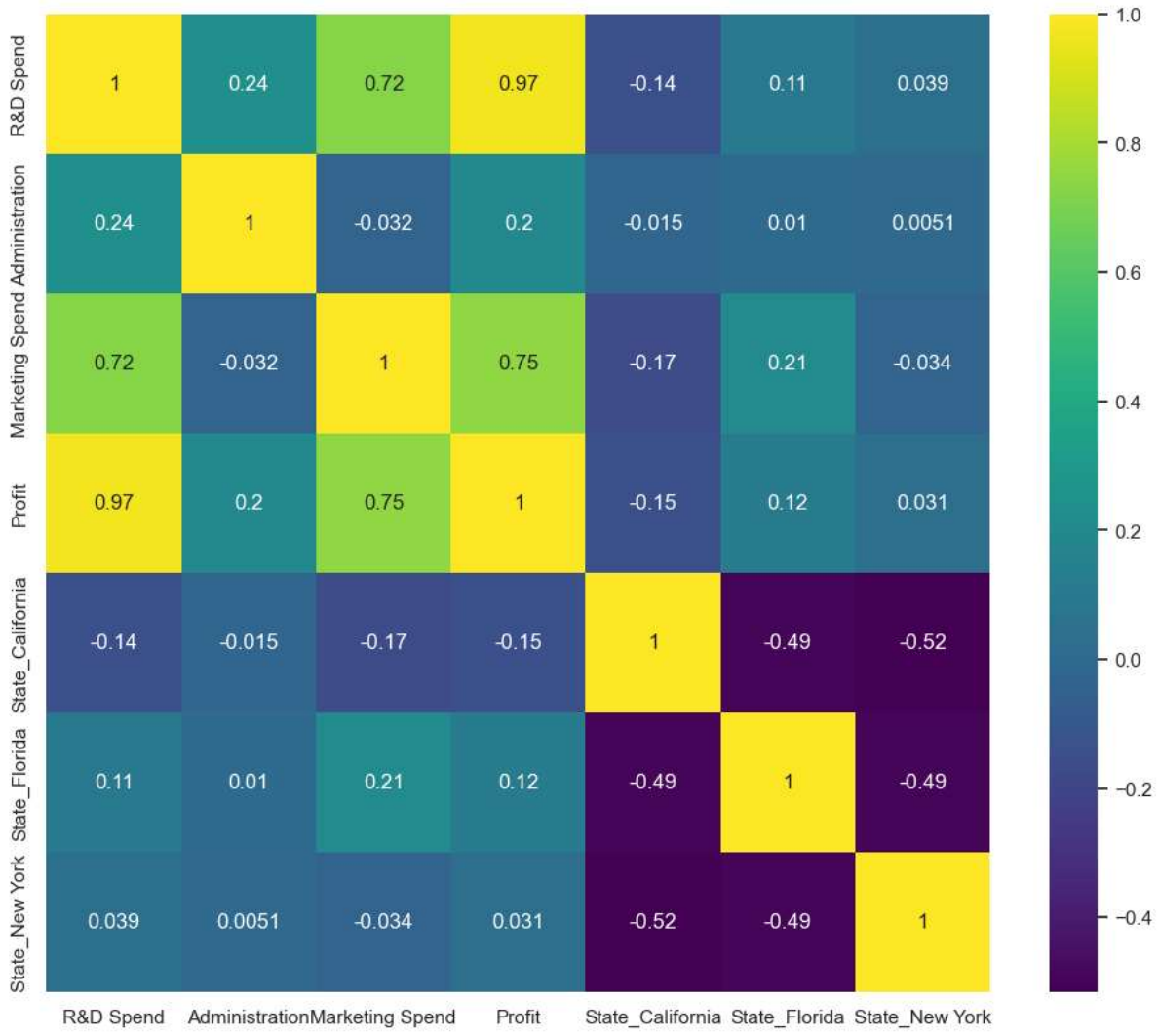
Out[52]:

	0	1	2	3	4
0	2.016411	0.560753	2.153943	-0.685994	1.393261
1	1.955860	1.082807	1.923600	-0.685994	-0.717741
2	1.754364	-0.728257	1.626528	1.457738	-0.717741
3	1.554784	-0.096365	1.422210	-0.685994	1.393261
4	1.504937	-1.079919	1.281528	1.457738	-0.717741
5	1.279800	-0.776239	1.254210	-0.685994	1.393261
6	1.340066	0.932147	-0.688150	-0.685994	-0.717741
7	1.245057	0.871980	0.932186	1.457738	-0.717741
8	1.030369	0.986952	0.830887	-0.685994	1.393261
9	1.091819	-0.456640	0.776107	-0.685994	-0.717741
10	0.620398	-0.387599	0.149807	1.457738	-0.717741
11	0.593085	-1.065540	0.319834	-0.685994	-0.717741
12	0.443260	0.215449	0.320617	1.457738	-0.717741
13	0.402078	0.510179	0.343957	-0.685994	-0.717741
14	1.017181	1.269199	0.375742	1.457738	-0.717741
15	0.897913	0.045868	0.419219	-0.685994	1.393261
16	0.094441	0.009118	0.440446	-0.685994	-0.717741
17	0.460720	0.855666	0.591017	-0.685994	1.393261
18	0.396725	-0.258465	0.692992	1.457738	-0.717741
19	0.279442	1.159837	-1.743127	-0.685994	1.393261
20	0.055726	-0.269588	0.723926	-0.685994	-0.717741
21	0.102724	1.169186	0.732788	-0.685994	1.393261
22	0.006007	0.051850	0.762376	1.457738	-0.717741
23	-0.136201	-0.562211	0.774349	1.457738	-0.717741
24	0.073115	-0.795469	-0.581939	-0.685994	1.393261
25	-0.199312	0.656489	-0.603517	-0.685994	-0.717741
26	0.035370	0.821718	-0.635835	1.457738	-0.717741
27	-0.035519	0.235069	1.174271	-0.685994	1.393261
28	-0.168793	2.210141	-0.767189	1.457738	-0.717741
29	-0.178609	1.142457	-0.858134	-0.685994	1.393261
30	-0.258074	-0.205629	-0.990357	1.457738	-0.717741
31	-0.276958	1.130554	-1.014419	-0.685994	1.393261
32	-0.226949	0.283924	-1.362450	-0.685994	-0.717741
33	-0.401129	-0.659324	0.029817	1.457738	-0.717741
34	-0.600682	1.310535	-0.001879	-0.685994	-0.717741

	0	1	2	3	4
35	-0.609750	-1.308658	-0.045493	-0.685994	1.393261
36	-0.991570	0.205925	-0.081763	1.457738	-0.717741
37	-0.652532	-2.525994	-0.115608	-0.685994	-0.717741
38	-1.177178	-1.997270	-0.212785	-0.685994	1.393261
39	-0.773820	-1.383122	-0.297583	-0.685994	-0.717741
40	-0.989577	-0.100900	-0.315786	-0.685994	-0.717741
41	-1.008534	-1.320796	-0.384552	1.457738	-0.717741
42	-1.102106	-0.906938	-0.520596	-0.685994	-0.717741
43	-1.281134	0.217682	-1.449605	-0.685994	1.393261
44	-1.134305	1.206419	-1.509074	-0.685994	-0.717741
45	-1.600350	0.101254	-1.727400	-0.685994	1.393261
46	-1.593413	-0.199322	0.711122	1.457738	-0.717741
47	-1.622362	0.507722	-1.743127	-0.685994	-0.717741
48	-1.610433	-2.509409	-1.743127	-0.685994	1.393261
49	-1.622362	-0.157226	-1.369985	-0.685994	-0.717741

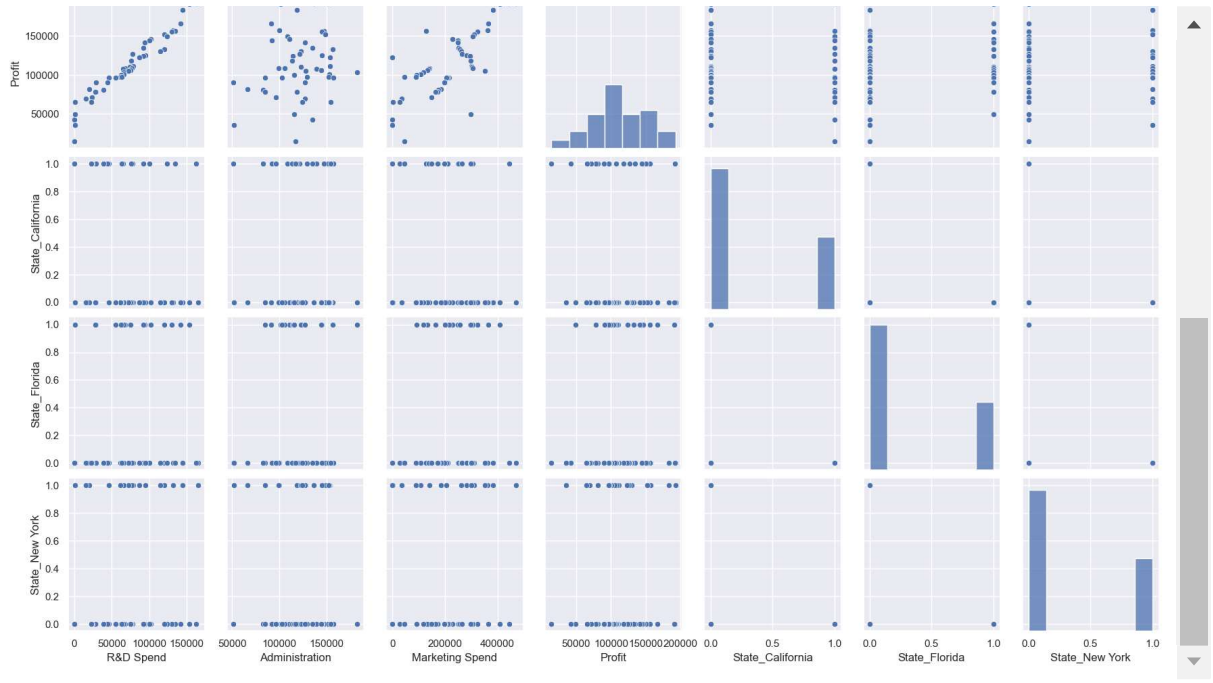
Finding correlation

In [53]:
```
plt.figure(figsize=(12,10))
corr = data.corr()
sns.heatmap(corr, annot=True, cmap='viridis')
```

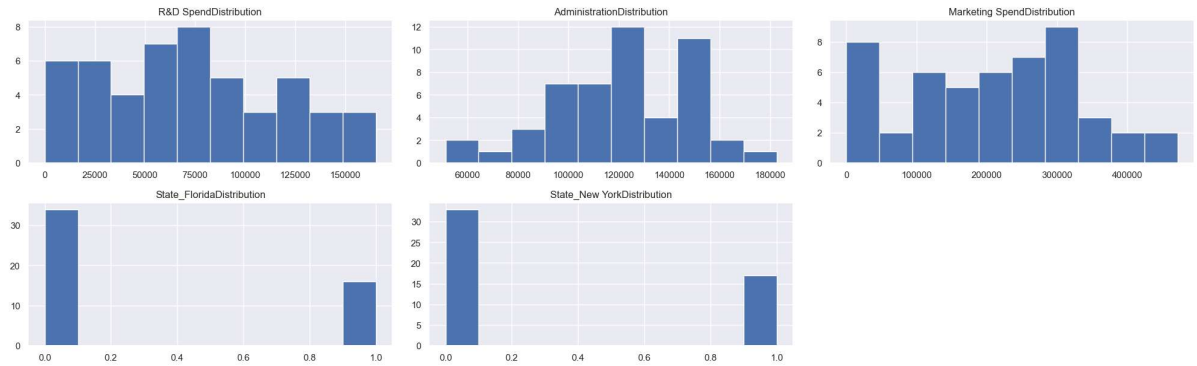


In [54]:
```
#sns.pairplot(data, hue='smoker')
```

In [55]:
```
sns.pairplot(data)
```



In [56]:
```
def histograms(dataset, variables, n_rows, n_cols):
    fig = plt.figure(figsize=(20,15))
    for i , var_name in enumerate(variables):
        ax = fig.add_subplot(n_rows, n_cols, i+1)
        dataset[var_name].hist(bins=10, ax=ax)
        ax.set_title(var_name +"Distribution")
    fig.tight_layout()
    plt.show()
```



VIF

In [57]:
```
variable = sc_x
```

Out[57]:
```
(50, 5)
```

In [58]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
variable = sc_x

vif = pd.DataFrame()

vif['Variance Inflation Factor'] = [variance_inflation_factor(variable, i ) fo
```

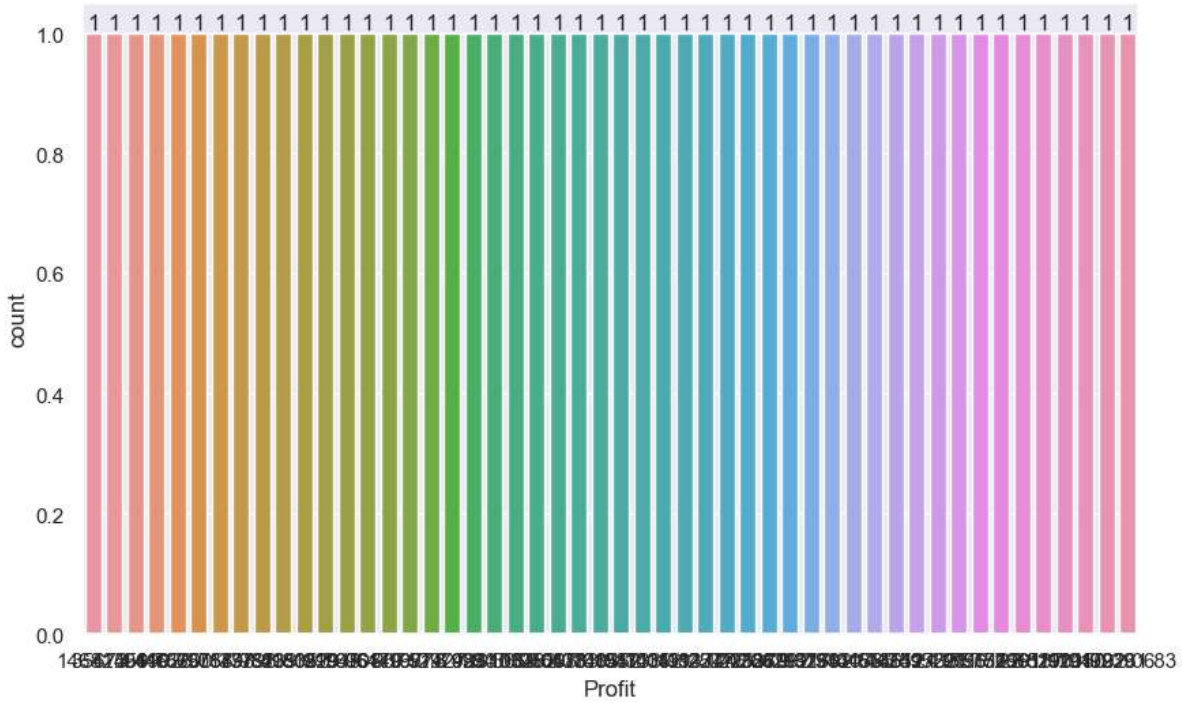
In [59]:
```
vif
```

Out[59]:

	Variance Inflation Factor	Features
0	2.495511	R&D Spend
1	1.177766	Administration
2	2.416797	Marketing Spend
3	1.387641	State_Florida
4	1.335061	State_New York

In [60]:
```
plt.figure(figsize=(10,6),dpi=100)
ax=sns.countplot(x='Profit',data=data)

for i in ax.containers:
```



Split the data into training and test for building the model and for prediction

In [61]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, rando
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(35, 5) (15, 5) (35,) (15,)
```

LinearRegression

In [62]:
```
# Linear Regression Model
lr = LinearRegression()
```

Out[62]:
```
LinearRegression
LinearRegression()
```

In [63]:
```
y_pred = lr.predict(x_test)
```

In [64]:
```
r2_score(y_test, y_pred)
```

Out[64]:
```
0.9241937845107013
```

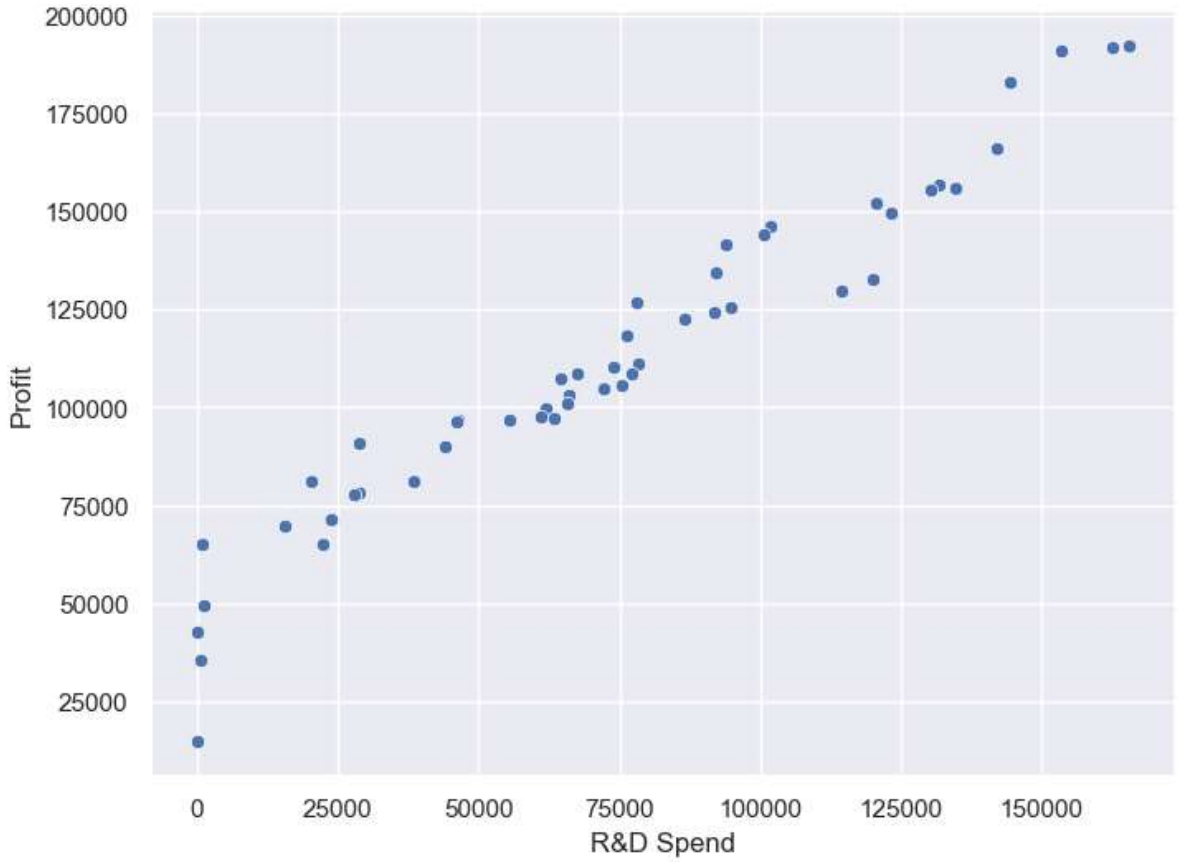
In [65]:
```
r2_score(y_train, y_pred_train)
```

Out[65]:
```
0.9517559349311887
```

In [ ]:

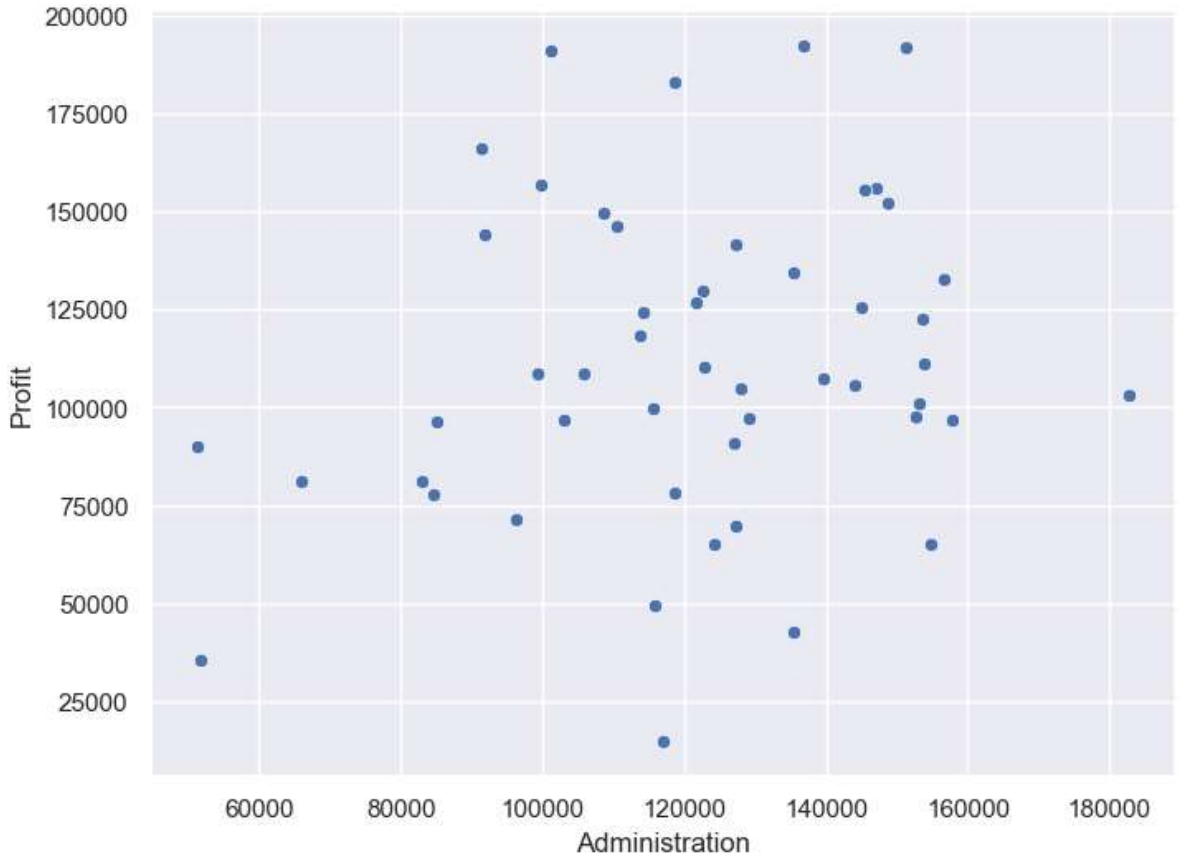
In [66]:
```
plt.figure(figsize = (8,6))
```

Out[66]: <Axes: xlabel='R&D Spend', ylabel='Profit'>



In [67]:
```
plt.figure(figsize = (8,6))
```

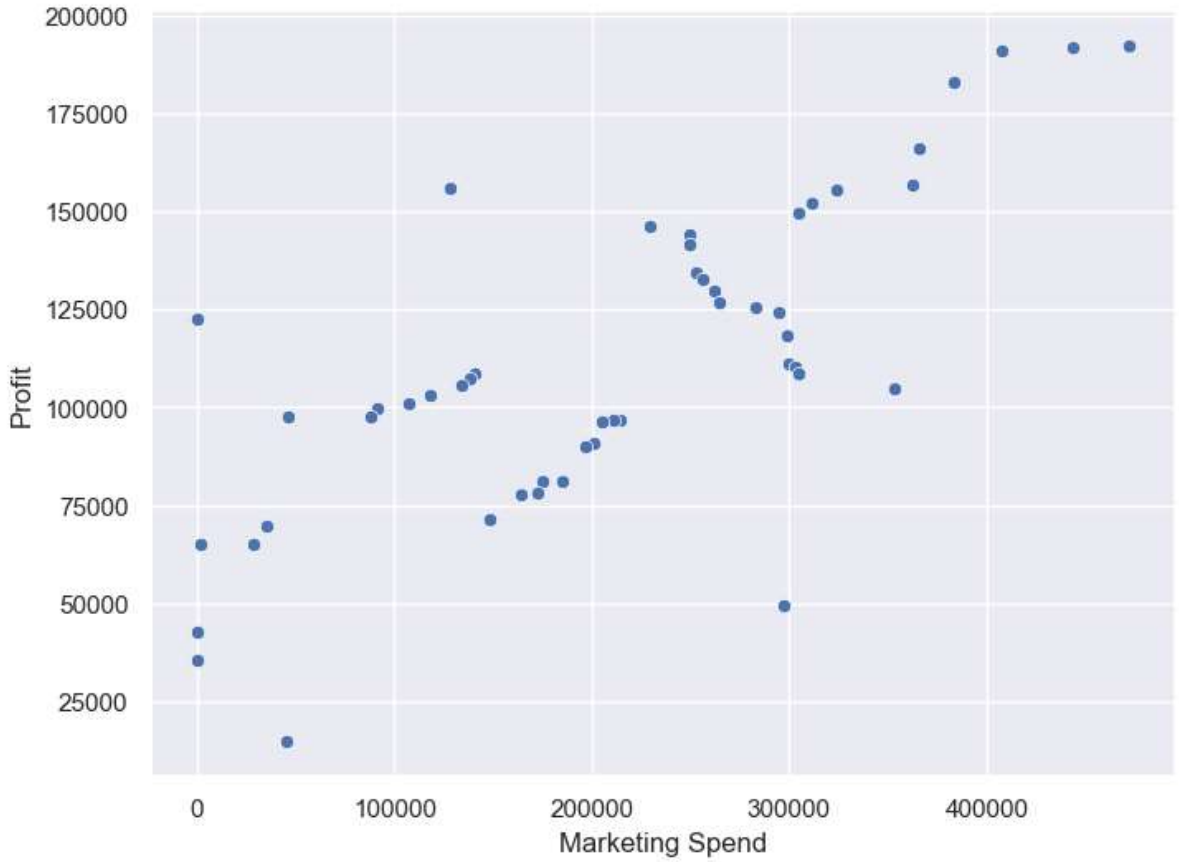
Out[67]: <Axes: xlabel='Administration', ylabel='Profit'>



In [ ]:

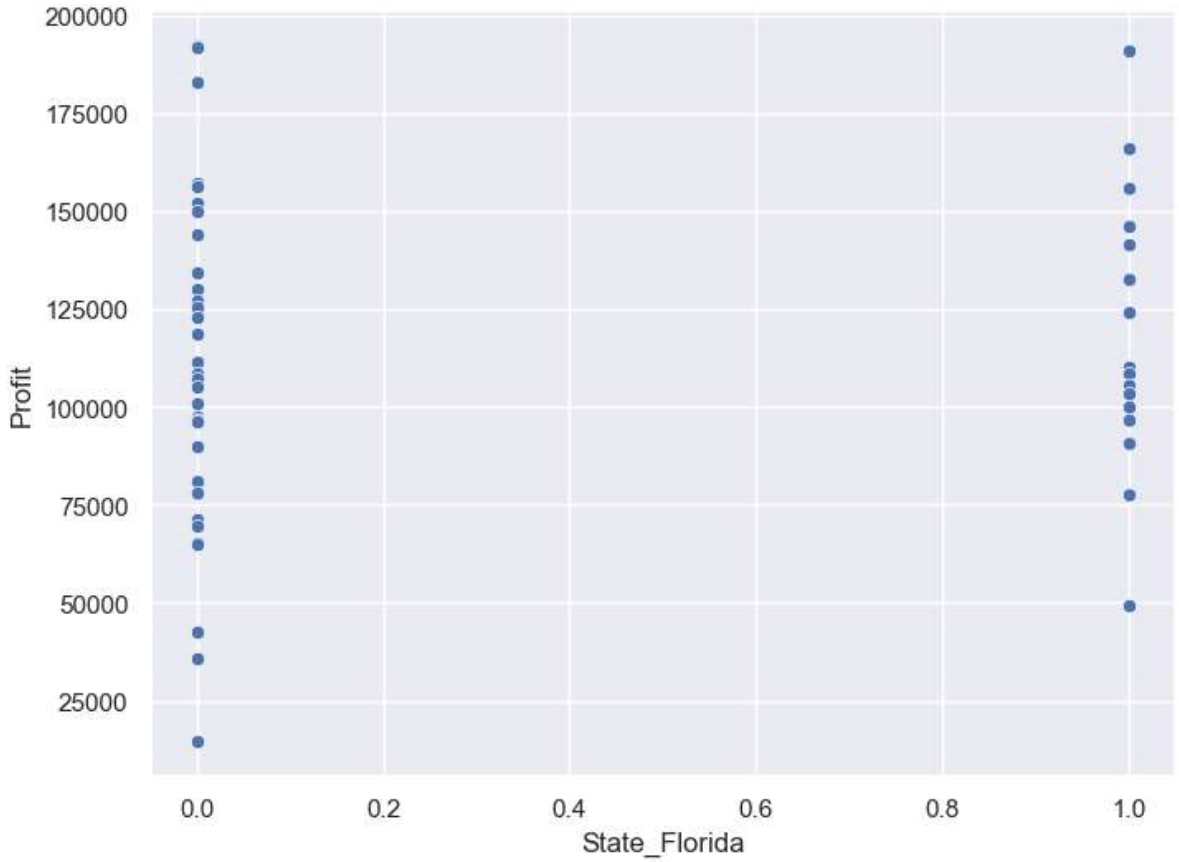
In [68]:
```
plt.figure(figsize = (8,6))
```

Out[68]: <Axes: xlabel='Marketing Spend', ylabel='Profit'>



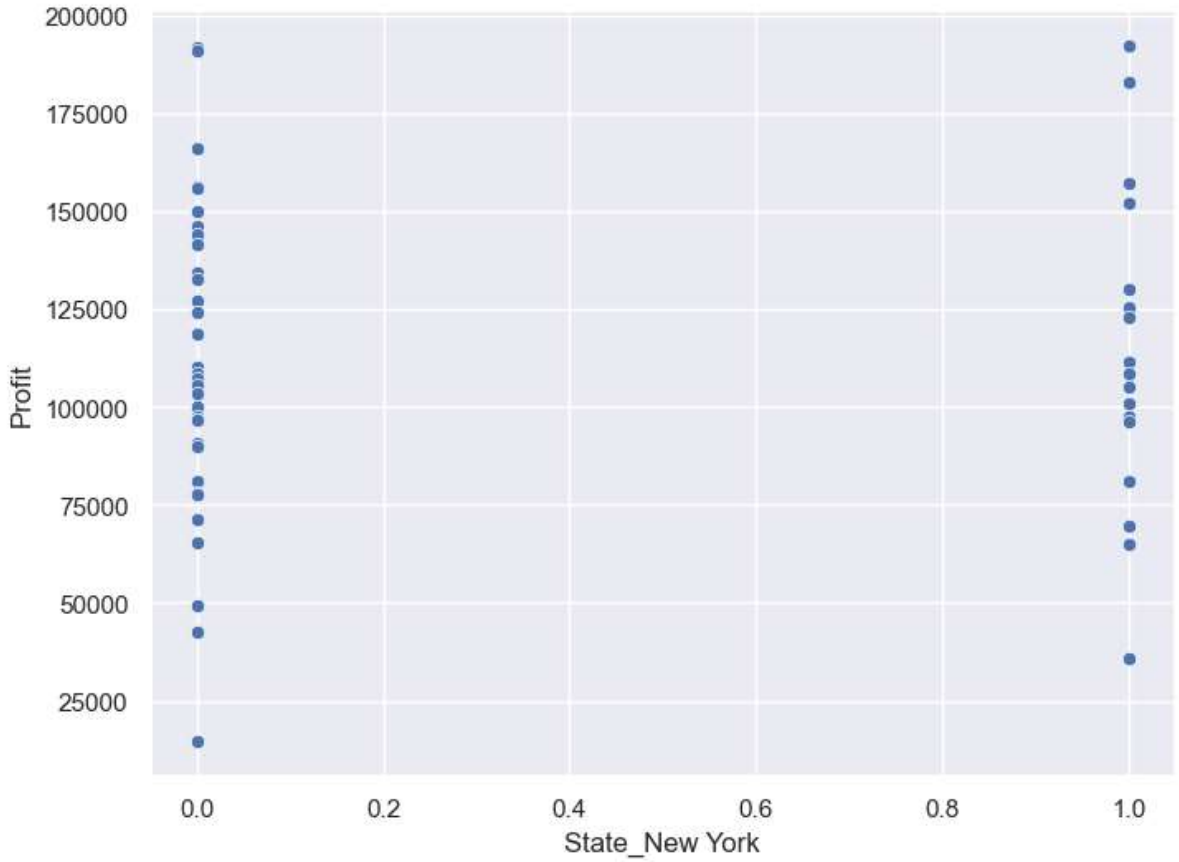
In [69]:
```
plt.figure(figsize = (8,6))
```

Out[69]: <Axes: xlabel='State_Florida', ylabel='Profit'>



In [70]:
```
plt.figure(figsize = (8,6))
```

Out[70]:
```
<Axes: xlabel='State_New York', ylabel='Profit'>
```



Polynomial Regression

In [71]:
```
poly = PolynomialFeatures()
x_train_trans = poly.fit_transform(x_train)
```

In [72]:
```
# Linear Regression Model
lr = LinearRegression()
```

Out[72]:
```
LinearRegression
LinearRegression()
```

In [73]:
```
r2_score(y_test, y_pred)
```

Out[73]:
```
0.9241937845107013
```

In [74]:
```
y_pred_train = lr.predict(x_train_trans)
```

In [75]:
```
r2_score(y_train, y_pred_train)
```

Out[75]:
```
0.9656034783412273
```

Performance matrix¶

In [76]:
```
print("RMSE :", np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
RMSE : 10495.033138078295
```

In [77]:
```
print("MAPE :", metrics.mean_absolute_error(y_test, y_pred)/100)
```

```
MAPE : 83.53874382822251
```

In [78]:
```
print("MSE :", metrics.mean_squared_error(y_test, y_pred))
```

```
MSE : 110145720.56936155
```

In [79]:
```
print("MAE :", metrics.mean_absolute_error(y_test, y_pred))
```

```
MAE : 8353.87438282225
```

In [80]:
```
r2_score(y_test, y_pred)
```

Out[80]:
```
0.9241937845107013
```

In [81]:
```
y_pred_train = lr.predict(x_train_trans)
```

In [82]:
```
r2_score(y_train, y_pred_train)
```

Out[82]:
```
0.9656034783412273
```

2.(OLS) Ordinary Least Square Method

In [83]:
```
from statsmodels.regression.linear_model import OLS
```

In [84]:
```
reg_model = smf.OLS(endog = y_train, exog=x_train).fit()
```

In [85]:
```
reg_model.summary()
```

Out[85]:

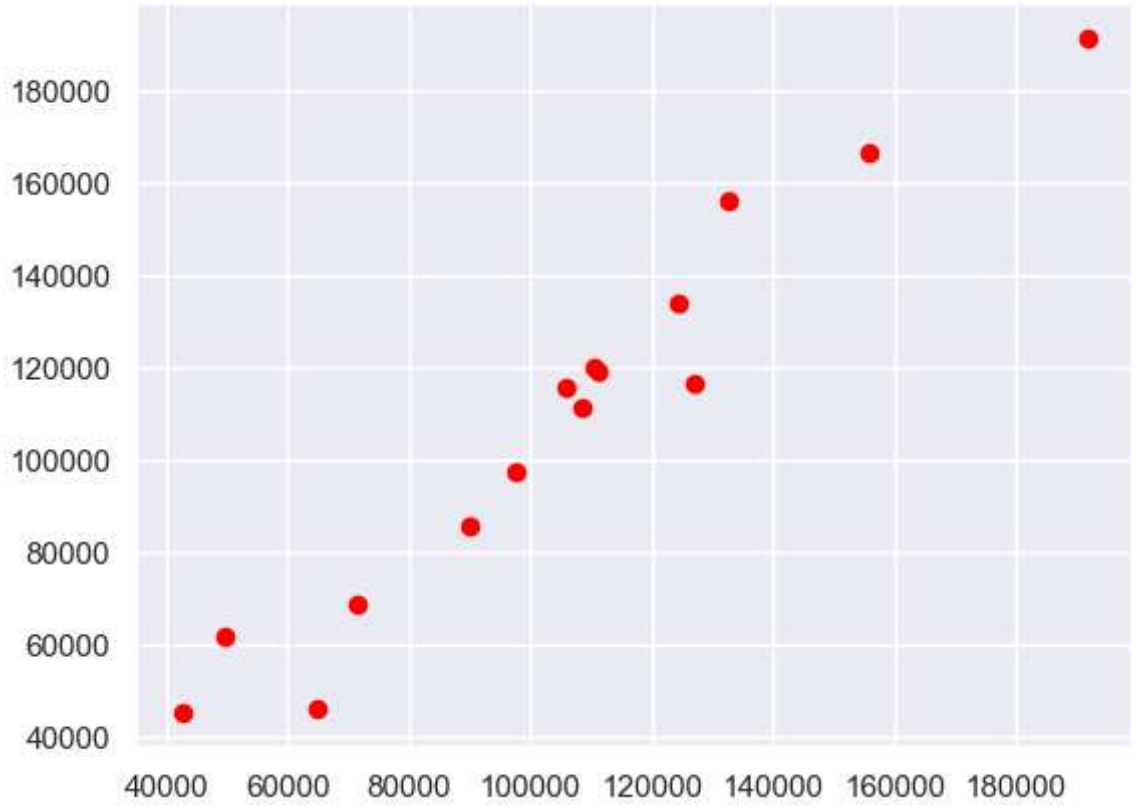
OLS Regression Results

Dep. Variable:	Profit	**R-squared (uncentered):**	0.991
Model:	OLS	**Adj. R-squared (uncentered):**	0.989
Method:	Least Squares	**F-statistic:**	632.3
Date:	Sat, 05 Aug 2023	**Prob (F-statistic):**	1.92e-29
Time:	09:41:51	**Log-Likelihood:**	-377.80
No. Observations:	35	**AIC:**	765.6
Df Residuals:	30	**BIC:**	773.4
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
R&D Spend	0.7241	0.076	9.575	0.000	0.570	0.879

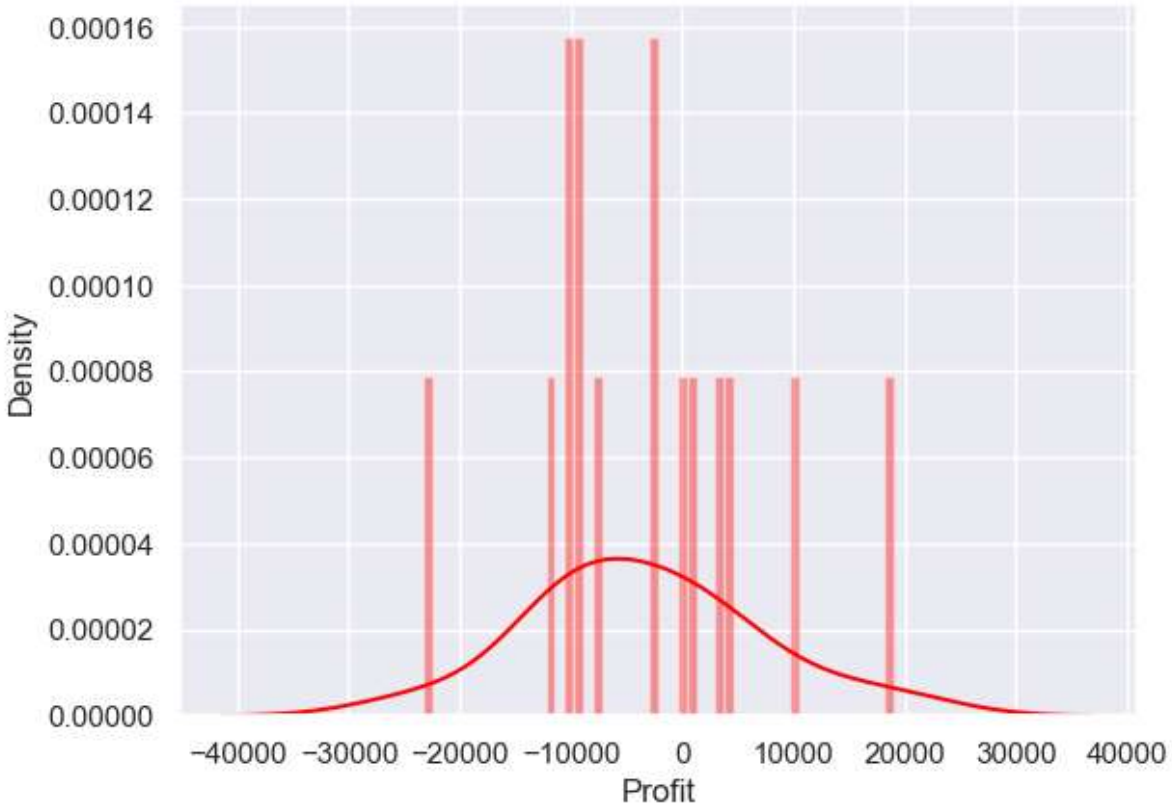
In [86]:
```
# Check linearity
```

Out[86]:
```
<matplotlib.collections.PathCollection at 0x19134777b50>
```



In [87]:
```
# Normality of Residual

sns.distplot((y_test - y_pred), bins=50,color='red')
```



Lasso regularization

In [88]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=0.1)
lasso.fit(x_train, y_train)
```

```
Lasso Model : [8.01207462e-01 1.96276308e-02 3.51978665e-02 5.32695947e+03
 2.20125249e+02]
```

In [89]:
```
y_pred_train_lasso = lasso.predict(x_train)
```

In [90]:
```
print("Training Accuracy :", r2_score(y_train, y_pred_train_lasso))
print()
```

```
Training Accuracy : 0.9517559348109744

Test Accuracy : 0.9241969817600443
```

Ridge Regression (L2- Regularization)

In [91]:
```
# Part 2 : Ridge Regression (L2- Regularization)
# closure to zero but not exact zero
# penalty - 0.3
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=0.3)
ridge.fit(x_train, y_train)
```

```
Ridge Model : [8.01364720e-01 1.91148004e-02 3.53046258e-02 5.02555051e+03
 7.23465281e+01]
```

In [92]:
```
y_pred_train_ridge = ridge.predict(x_train)
```

In [93]:
```
print("Training Accuracy :", r2_score(y_train, y_pred_train_ridge))
print()
```

```
Training Accuracy : 0.9517478664733822

Test Accuracy : 0.9254451373878588
```

ElasticNet

In [94]:
```
from sklearn.linear_model import ElasticNet
elastic = ElasticNet(alpha=0.3, l1_ratio=0.1)
```

Out[94]:
```
ElasticNet(alpha=0.3, l1_ratio=0.1)
```

In [95]:
```
y_pred_train_elastic = elastic.predict(x_train)
```

In [96]:
```
print("Training Accuracy :", r2_score(y_train, y_pred_train_elastic))
print()
```

```
Training Accuracy : 0.9507311124761139

Test Accuracy : 0.9383852133807632
```

Gradient Descent

In [97]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(sc_x, y, test_size=0.25, r
```

```
(37, 5) (13, 5) (37,) (13,)
```

In [98]:
```
from sklearn.linear_model import SGDRegressor
```

In [99]:
```
gd_model = SGDRegressor()
```

Out[99]:
```
SGDRegressor
SGDRegressor()
```

In [100]:
```
y_pred_gd_train = gd_model.predict(x_train)
```

In [101]:
```
print("GD Trainging Accuracy :", r2_score(y_train, y_pred_gd_train))

print()
```

```
GD Trainging Accuracy : 0.9473189683483982

GD Test Accuracy : 0.9414059052087741
```

Building Model - DecisionTree Regression Model

In [102]:
```
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor()
```

Out[102]:
```
DecisionTreeRegressor
DecisionTreeRegressor()
```

In [117]:
```
y_pred_train = dtree.predict(x_train)
```

In [104]:
```
# Evaluate the model
```

In [105]:
```
print(r2_score(y_train, y_pred_train))
print()
```

```
1.0

0.8677893894484192
```

Random Forest Regression

In [106]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
```

Out[106]:
```
RandomForestRegressor
RandomForestRegressor()
```

In [107]:
```
# predict
y_pred_train_rf = rf.predict(x_train)
```

In [108]:
```
# applying cross validation method in RandomForest
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(rf, x_train, y_train, cv=10)
```

Out[108]:
```
0.8554824675909524
```

In [109]:
```
print(r2_score(y_train, y_pred_train))
print()
```

```
1.0

0.8677893894484192
```

Summary of the Model Building

In [ ]:
```
#Performance matrix
# RMSE : 10495.033138078295
# MAPE : 83.53874382822251
# MSE : 110145720.56936155
```

In [ ]:
```
# Summary of the Model Building
# Linear regeression
# test accurancy : 0.9241937845107013
# train accurancy : 0.9517559349311887

# Polynomial Regression
# test accurancy : 0.9241937845107013
# train accurancy :0.9656034783412273

#( OLS) Ordinary Least Square Method
# R-squared (uncentered):   0.991
# Adj. R-squared (uncentered):  0.989

# Lasso regularization
# Training Accuracy : 0.9517559348109744
# Test Accuracy : 0.9241969817600443

# ElasticNet
#Training Accuracy : 0.9507311124761139
#Test Accuracy : 0.9383852133807632

# Gradient Descent
#GD Trainging Accuracy : 0.9473189683483982
#GD Test Accuracy : 0.9414059052087741

# DecisionTree Regression Model
# train accurancy : 1.0
# train accurancy :0.8677893894484192

# Random Forest Regression
# train accurancy : 1.0
# train accurancy :0.8677893894484192
```