Analysis on Company's Profit Prediction

In This Model, we are analyzing the profit for a Company by using it's Spend in R&D, Administration and Marketing in different States. For this, I am using the Regression Technique of Machine Learning.

Importing Dependencies

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

from sklearn.preprocessing import LabelEncoder , OneHotEncoder
from sklearn.compose import ColumnTransformer

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

Data Collection

In [2]:
```
companies = pd.read_csv('1000_Companies.csv')
```

In [3]:
```
companies.head()
```

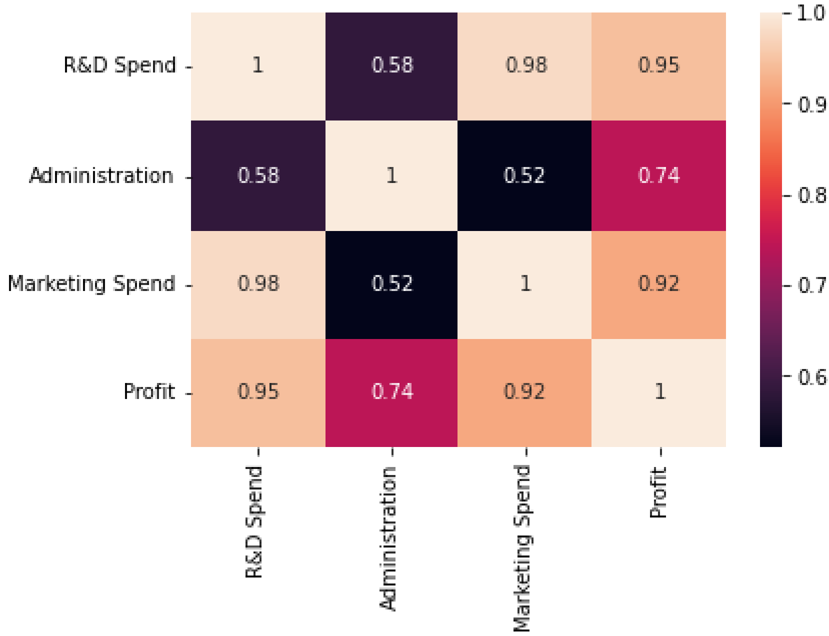
Out[3]:

	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 [4]:
```
companies.shape
```

Out[4]:
```
(1000, 5)
```

In [5]:
```
sns.heatmap(companies.corr(),annot=True)
plt.show()
```



In [6]:
```
companies.describe()
```

Out[6]:

	R&D Spend	Administration	Marketing Spend	Profit
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	81668.927200	122963.897612	226205.058419	119546.164656
std	46537.567891	12613.927535	91578.393542	42888.633848
min	0.000000	51283.140000	0.000000	14681.400000
25%	43084.500000	116640.684850	150969.584600	85943.198543
50%	79936.000000	122421.612150	224517.887350	117641.466300
75%	124565.500000	129139.118000	308189.808525	155577.107425
max	165349.200000	321652.140000	471784.100000	476485.430000

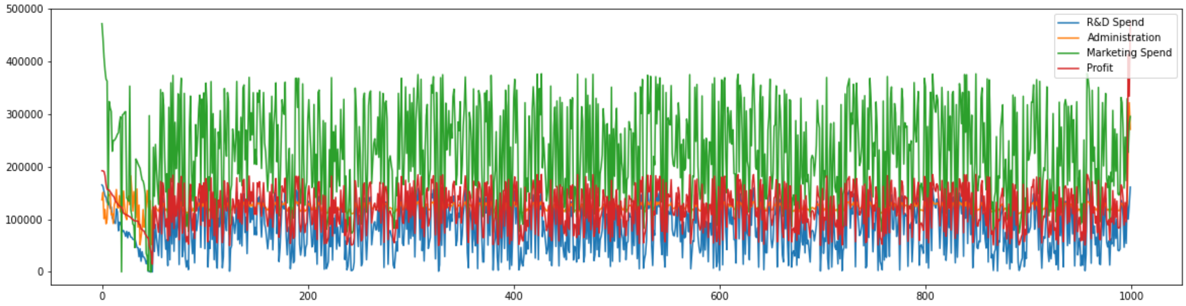
In [7]:
```
companies_num = companies.drop(columns = 'State')
```

In [8]:
```
companies_num.mean()
```

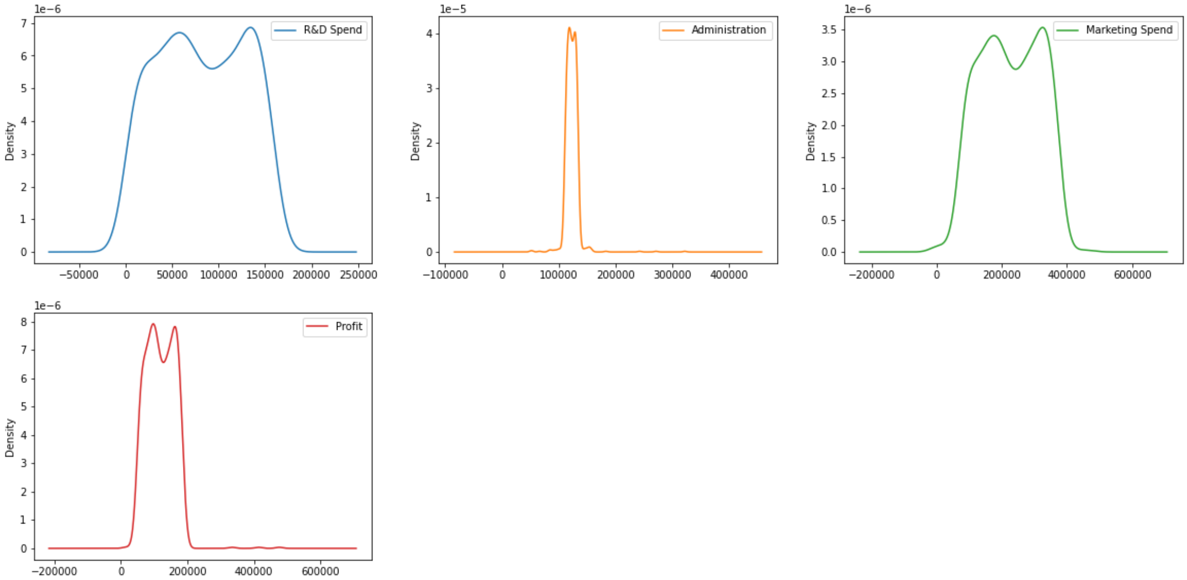
Out[8]:
```
R&D Spend           81668.927200
Administration     122963.897612
Marketing Spend    226205.058419
Profit             119546.164656
dtype: float64
```

Data Analyzation

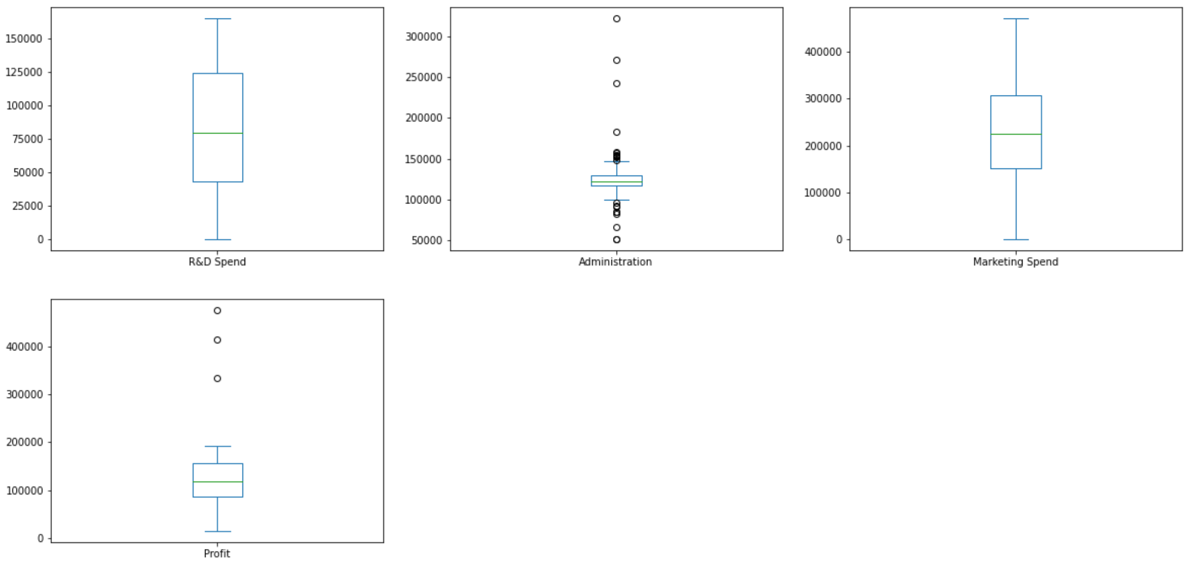
In [9]:
```
companies_num.plot(figsize=(20, 5))
plt.show()
```



In [10]:
```
companies_num.plot(kind='density', subplots=True, layout=(3,3), figsize=(20, 15), s
plt.show()
```



In [11]:
```
companies_num.plot(kind= 'box' , subplots=True, layout=(3,3), sharex=False, sharey=
plt.show()
```



In [12]:
```
corr_analysis = companies_num.corr()
corr_analysis
```

Out[12]:

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.582434	0.978407	0.945245
Administration	0.582434	1.000000	0.520465	0.741560
Marketing Spend	0.978407	0.520465	1.000000	0.917270
Profit	0.945245	0.741560	0.917270	1.000000

Note: Since, the correlation values between every label is strong. Thus this will lead to cause Multicollinearity. Hence, we will have to drop the higher most correlated label/column.

KEY TAKEAWAYS

1. A variance inflation factor (VIF) provides a measure of multicollinearity among the independent variables in a multiple regression model.
2. Detecting multicollinearity is important because while multicollinearity does not reduce the explanatory power of the model, it does reduce the statistical significance of the independent variables.
3. A large variance inflation factor (VIF) on an independent variable indicates a highly collinear relationship to the other variables that should be considered or adjusted for in the structure of the model and selection of independent variables

Data Preprocessing

Label_and_Coding: Process to giving weightage to particluar fields

_Newyork = 1 , _California = 2 , _Florida = 3

This causes biased result towards florida bcoz of more value given to it Hence we have to do One_Hot_Encoding

_Newyork = (1,0,0) , _California = (0,1,0) , _Florida = (0,0,1)

In [13]:
```
le = LabelEncoder()
data = companies
```

In [14]:
```
data['State'] = le.fit_transform(data['State'])
```

In [15]:
```
columnTransformer = ColumnTransformer([('encoder',OneHotEncoder(),[3])],remainder=
```

In [16]:
```
data = np.array(columnTransformer.fit_transform(data),dtype=np.float64)
```

In [17]:
```
X = data[:,:-1]
y = data[:,-1]
```

In [18]:
```
print(X)
```

```
[[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]
 ...
 [1.0000000e+00 0.0000000e+00 0.0000000e+00 1.0027547e+05 2.4192631e+05
  2.2714282e+05]
 [1.0000000e+00 0.0000000e+00 0.0000000e+00 1.2845623e+05 3.2165214e+05
  2.8169232e+05]
 [0.0000000e+00 0.0000000e+00 1.0000000e+00 1.6118172e+05 2.7093986e+05
  2.9544217e+05]]
```

In [19]:

```
print(y)
```

```
[192261.83    191792.06    191050.39    182901.99    166187.94
 156991.12    156122.51    155752.6     152211.77    149759.96
 146121.95    144259.4     141585.52    134307.35    132602.65
 129917.04    126992.93    125370.37    124266.9     122776.86
 118474.03    111313.02    110352.25    108733.99    108552.04
 107404.34    105733.54    105008.31    103282.38    101004.64
  99937.59     97483.56     97427.84     96778.92     96712.8
  96479.51     90708.19     89949.14     81229.06     81005.76
  78239.91     77798.83     71498.49     69758.98     65200.33
  64926.08     49490.75     42559.73     35673.41     14681.4
 123485.2464   82155.48418 125867.0108  104976.1696   89803.10053
  75297.23305 114284.5283  171985.0761   72337.96774 169566.5772
 158670.9451  114522.8756   85842.60573 101106.2297   59328.81874
 157142.6178   68669.64059 177717.3712   94409.4396  183945.1553
  82484.38635 144515.3371  105333.2634  122331.0988  168459.4156
  60947.70089 162733.9549  181574.4968   73577.54452  84782.43014
 168870.3298   72607.06952  56788.15621  67473.63267  52731.98078
 140237.9002  166598.769   102990.7964   78406.85364 111764.3688
  63662.63887 142575.2414  115980.2967  132915.7689  155954.2985
 167412.0544   88710.46186 164139.2642  131574.5314  169314.5613
  86636.24242 177468.7724  157979.8234   56944.49153  98500.64098
  87218.86913 178759.6067  101668.3534  151782.7938   68872.96194
 139016.2635   69109.60065  55091.53354 136286.8026  122307.1786
 154356.7737  114806.5004   55623.75707  73896.1952  172901.7308
 129480.6633  115890.5961  169404.2619  161666.9449   50116.99489
  86613.17655  91640.68127 138793.2935  128986.8828  111461.9497
 129804.4397  178847.5987  101028.4891  136845.5092   94579.44358
 148748.3508   91139.21223 173343.3994  175906.2735  112883.4905
 138079.1059   78689.62408 140832.487   182316.0217  129232.9188
 152520.9015  133849.5093  169431.5992  109333.0556  121505.8533
 108917.0157  169324.8128  161423.4719  171478.4813  137670.7546
 105582.7164  141472.3512  169572.5573  126993.8211  119743.4502
 103155.6747   86313.32028 105674.9799  152268.8856  101971.6268
 166402.282    84757.65569 142289.0538  155518.6099  177675.5109
 111138.1732  142490.6665   65814.59883  88870.21435 125271.5697
 163462.6654  154539.592   151150.6182  178296.5808  106401.1276
  76644.45054 110228.3529  122570.3004   54205.63339  73850.06347
 117353.9972   63814.70273  67199.40514 181391.6784  181550.5766
 156935.8793  181579.6225   94376.97653  76229.26494  89583.54765
 176839.1597  113578.8837   74477.11333  88648.95289  94066.01447
 146874.0356  163549.8031  138872.7426  121482.7874   89594.65343
  94297.52743 159509.0049   81169.63193 117700.8395   61393.64099
 169661.4036  171651.9025  128265.0066   75270.75002 109658.5406
 145976.1753   53395.76517 144038.6425   54300.45973  58223.36571
  66007.66868 121491.3304   89915.86699 182098.1774   68160.48294
 136575.5531  111422.6523  141814.0678   85830.64565 121927.8732
  96616.9285   90687.2921  155424.6379  122954.7315  113188.4725
 163883.8311   68984.01982  99306.23774  52325.33808  58694.93455
  80229.05713 104723.2993   51256.61958  51336.06868  53225.76119
  64672.41127 173263.9503  155791.1289  118334.7237   90164.46578
 140872.6387  116492.8715   58918.75889  64349.48913 164375.9029
 169059.9825  109663.6663   68719.18949 102833.6068  158338.6258
 168686.6571  170883.8945   92949.45565 110023.323   126506.0207
 130602.3478  117289.0711  117645.3106  148944.8378  114970.5243
 170174.8327   92758.09438 142762.3312  168802.8408   56138.89476
 160646.0668  162479.376   146925.293    61291.12602 112642.5803
 116273.3187  109285.2152  103264.1697   60111.34964  55227.36587
  73733.02555  98129.02423  82589.46419 168876.3098  104996.6726
 147736.0156  181512.1335   95260.31381  91977.27207 126295.865
 174329.2516  118265.5261  153825.4044   94093.35179 167934.0264
 103243.6667  160935.6715  107019.6345  173336.5651   73154.67029
 145877.9318  164180.2702  184632.0056  151992.0952  100693.6069
 180684.3252   83391.64379 121140.2166   66656.07583 123814.1486
 150292.0553  159514.1307   87019.81924  96793.76682 105154.7165
```

```
 175999.3913  169169.3318  182911.4628  134979.7368   70244.95388
 169623.8148  135817.7966   50070.86316  60368.49134 182028.1255
  92289.08842 184555.1194  135591.4094   73790.26307 160667.424
 121248.7116  135506.8346  154351.6479  116529.6061  142869.1176
 131062.8109  165941.819   172384.0302  142712.7823   91790.18226
 100435.6109  144750.2672  137468.2876  161824.1346   92920.40974
 108421.5267  141046.0598  156147.3684  182641.5067  150475.728
  79794.22283  55832.20416 152920.7099  145711.345   128195.809
  80504.99325 172615.5432  102213.3913   91240.87291 158160.0789
  54244.9308  171176.9165  167377.0285  154029.5801  105503.2673
  65204.63479  65181.56893 169235.1122  144807.5047  171235.8626
  92485.57543 161936.0467  107910.6604  183965.6583  152134.7618
  51913.56964  94320.5933  103906.5968  136632.7906   72949.64036
  51690.5996  106212.3292  149849.5324   64574.16776 120375.6258
 154765.9792   86983.939   171557.9304  157690.2186   67839.26938
 175747.3753   53177.92087 123648.4161   59636.36364 105740.7603
 165029.4358  180633.922   126997.2382  171687.7827  125636.3521
  52225.38599  59981.49735  88971.02073  63306.39937  58666.74293
  77017.77587 161008.2863   77847.29279  97478.0542   85976.72948
 149792.2949  128937.3339   80082.11902 154238.8815  174414.6808
 156182.3943   71376.03566 145680.5905  185028.3968   68921.65655
 172937.611   150707.2409  185352.1732  106018.405   131659.9605
 114839.8177   97372.12207  78308.61013  85769.13667 126436.8231
  67226.74247 174996.4532   59225.44949 102740.489   141180.1836
 157325.4362   90312.25819 162440.0786  126117.3181   58784.63514
 165530.0505   90550.60548 161123.6156   87309.42402  72735.21323
  97238.85262  68648.28331 157529.6118   85529.93509  80539.1649
 163525.0287  139080.3354  165186.6254   68347.57275  90544.62544
  88027.88306  64997.04199 184979.7022   70555.91594  91209.26412
 120790.8114  157615.041   145240.6305   65085.034    75968.70607
 181243.886   102274.046   167257.4277   95537.9585  158961.4042
  80106.89347 127075.833   184887.4387   67282.27141  77242.4545
 164713.348    84909.71956 160000.2225   73872.27504 101119.044
 107252.8561  117043.8895  155881.6838  141832.0079   81963.26862
 142494.938   143333.8521   72478.92582 134122.8826   57893.60924
 174050.7527   66273.3533  139130.7386   99032.8645  141073.3972
 156340.4382   88072.30621 126393.2542   65664.24355 134894.3077
 110963.8978  103171.0519   84808.91317 139555.3214   68705.52083
  96214.55727  98444.25774 145073.1894  138588.2636  138736.9103
  77629.44849 135641.8126   91657.76709 169403.4076  133617.1421
  70509.78421 124269.4859   54119.34996 174320.7087  151804.1511
 180524.5727   54971.93275  69995.5008   77837.89559 100556.066
 184099.7821  151774.2509  153976.614   183093.4268   91843.14832
  59342.48741 115751.3466  108629.9738  182876.4368  143061.3332
 176623.024   154901.8116  181946.9678   99906.80457  77132.25091
 175771.2955   78153.1291   56852.22806 169564.8687   75445.02546
 148158.0355  144203.5207  142226.6905  175396.2616  118130.5481
 108776.0576  107382.7084  120617.3903  150393.716   106070.5168
 160842.5538  146499.0017   97712.98433 125981.4858   71885.19331
 130070.1243  104151.7784  106169.6146  147194.3948  141527.0259
 113981.2549  184581.6024   54060.40386  76376.20305  99301.96628
  98464.76074 174007.1838  128337.6214  176344.525   107669.7503
  90482.26217  79710.50227  84583.38025 166679.9267   74052.53052
 129889.8688   84552.62576 131051.7051  116205.8296   65471.1737
 128060.831   127773.7891  108896.5127  176833.1797   54991.58145
 181929.0277  102873.7585   57541.6412   65513.03398 162182.9369
  51671.80519 138022.7227  180257.1795   60408.64304 178552.8682
 182059.7343  114403.2748   85101.93511 157086.2346  150024.6621
 127248.3999  154190.1869  126058.372   138213.2296  101352.2656
 100958.4372   63373.88839 179522.4889  184865.2271  102619.1797
 175826.8244  165119.1364  134268.9664   61306.50327 165495.0246
  77667.8916  154051.7917  111643.0594  106517.3112   68575.66854
 155915.0011  124695.7773  172769.3156  185068.5485  157591.1208
 106865.8621  104824.1057  120990.7156   56620.7151  159613.2285
```

```
 175705.515    98776.57709 165987.9507   52609.81711 117842.6519
  80045.38449  80066.74177 137719.4492  149477.9157  180783.423
 103404.2734   88302.96488 100992.6089  141082.7944  142586.3472
 175166.4572   56991.47755 109105.8141  167373.6113  153627.2089
 126846.883    76545.35274  65255.03798 171780.0462  118944.6878
 168285.1402   95226.99644  59264.74689 163470.354   162747.6235
  56775.34184  79808.74578  83356.61784  74502.74207 145965.9238
  59784.15604 133834.1321   71717.75221  91441.63138 164779.9827
  98397.27172  92502.66126 136315.8485   79796.7857  144805.7962
  94294.11026 100589.3834   81824.87341  64950.05596 113472.9515
 116685.0871  114373.3746   99277.19183 109998.5485  124765.8292
 163155.1205  144820.3191  139915.8323   50468.96294  92362.55748
 158565.8673   79167.17296 103903.1796   98368.22581 182202.4009
  70232.13951  53483.75718 120345.7256  143415.8641   56789.0105
 110848.5685  144287.2413   90108.93684  85570.94108  98808.18587
  61960.03617 127725.0945   70282.5427  159431.2644   50994.35213
  83632.55396  88481.51178 168402.1781   87716.921   173519.3834
 177513.1956  181258.409   119914.3085  152243.2568  142852.0318
 176432.517   164424.5975   57258.01646 177442.2894  149190.8738
 130689.4855  107994.381   111257.774   123690.2763  163138.0347
  91370.72519 166902.0425   55641.69719  58179.79685 156545.4682
 104862.5488  181441.2273  164884.2062  132897.8287   81580.54608
 127455.1384  116045.2229  161630.2104  140594.1397   51276.26828
 121248.7116   65868.41919  60243.7648   52276.64348 162399.9269
  68210.88613 119961.2945   72657.47271 182979.8061  112054.8279
  74208.01155 164330.6254  174254.9283  119482.037   117637.622
 160252.2384  152175.7678  106433.5906  120445.6777  141408.2794
  52086.1365  133019.1381  172966.6569  163028.6854  109278.3809
 144130.0517  140289.1577  141273.3013   57992.70704 125971.2343
 120723.3224  128977.4856   65219.15775 100404.0021  177911.2954
 137842.4672  127909.6214  154173.101   170343.1281   59766.21593
  90712.92084  93280.9207   87045.44798  79170.59013 181485.6505
 174421.5151   63924.90632 140964.9022   55195.75709 140870.0758
 169745.1241   83444.60985 185032.6682  168246.6971  156488.2307
 152900.2069  167391.5514  163226.881   130427.2181   87219.72342
 125467.2024  173897.8345  142927.2094   76356.55435 158208.7735
  67733.33725  97340.51329 170464.4375  120738.6997  149669.2769
  79513.16096 168145.8907  170465.2918  125667.9609   63742.94226
 159389.4041   72870.19126 124577.0308  177000.6208  166579.1203
 143338.9779  169673.3637   74324.19518 184669.5944  136050.1639
 184516.6762   95065.53537 170547.3037  146196.5825  109344.1613
 150516.734    55771.54948  88082.55771 109877.2392  185272.7241
  88592.56966  86750.71746  99201.1599   97955.60308 164695.4079
 173531.3435  110363.331   144184.7263  102771.2435   87654.55773
 181102.0736   64966.2875  180257.1795  140589.8682   71772.42685
  77627.73991  51286.51978  80859.52417  86101.45602 132471.5373
  50428.81124  99424.12995  97446.44542 138908.6228  148134.1153
 159173.2684   82229.80752 100759.3874  113444.7599   62531.55709
  87112.08271 128995.4258   53649.48971 104159.467    63978.72668
 167407.783   116603.0751  165584.7252   88737.79919 152894.2269
  64769.80049  51324.1086   57067.50948  66357.07385  71235.93187
 166415.9507   57463.90068  80804.84952  70902.75824 159247.5918
 112487.9536  116260.5043  121916.7675  161488.398   161456.7893
 176018.1857  118219.3944  134983.154   180378.4888  139722.7625
  84378.35032 178978.3053   97882.98831  54932.63535  95148.40164
 106235.395   103813.479   134723.4494  174364.2776   82522.82946
  65743.69265  96581.04826  70153.5447   93606.40571 151891.2888
  79299.58812  83553.10486  60650.40749 123228.959    52481.67341
 161467.0408  107682.5646  140522.3792  102118.5649   57143.54142
 159227.0888  163673.6754  157493.7316   91623.59544 157949.9232
  94974.98049 148975.5923  158516.3184  110682.8359  146690.3629
  58605.23395 120412.3603  161783.1286   76487.26093  95178.30183
 104231.2275   58963.18204  76017.40068  59803.80475 129642.9786
  51003.74933  77362.05529 185502.5285  180753.5228  172495.0881
```

```
 140251.5689   63093.68082 171416.9724  111814.772   123671.4819
  92903.32391 105457.9899   74425.00156 173861.9543   62223.15791
  60869.96038 110395.794   161076.6296  107704.7762  141344.2075
 168760.9805   97599.36358  89558.7732   99322.46927  60065.21791
 102489.3274   94400.89669 154569.4922   90808.60147 138855.6568
 103378.6447  134808.0242   84305.73556  83178.92524  86221.9111
 165330.1463  161035.6236  138841.9881   89012.02672 132077.709
  95279.96251 164336.6055  413956.48    333962.19    476485.43    ]
```

Data Training and Testing

In [20]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

In [21]:
```
lr = LinearRegression()
```

In [22]:
```
lr.fit(x_train,y_train)
```

Out[22]:
```
▾ LinearRegression
LinearRegression()
```

In [23]:
```
y_pred=lr.predict(x_test)
```

In [24]:
```
print("Coefficient = ",lr.coef_)
print("Intercept = ",lr.intercept_)
```

```
Coefficient =  [ 4.46921768e+02 -3.42694235e+02 -1.04227533e+02  5.26047095e-01
  9.78530820e-01  9.80946128e-02]
Intercept =  -66123.76082362703
```

Model Accuracy Evaluation

In [25]:
```
#Finding Coefficient of Determination
r2 = r2_score(y_test,y_pred)
print('Coefficient of Determination: ',r2)
```

```
Coefficient of Determination:  0.931112023626835
```

In [26]:
```
#Finding Correlation Coefficient
r = r2**(0.5)
print('Correlation Coefficient: ',r)
```

```
Correlation Coefficient:  0.9649414612435487
```

Performing Modelling with handling Multicollinearity

Data Preprocessing

In [27]:
```
corr_analysis = companies_num.corr()
corr_analysis
```

Out[27]:

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.582434	0.978407	0.945245
Administration	0.582434	1.000000	0.520465	0.741560
Marketing Spend	0.978407	0.520465	1.000000	0.917270
Profit	0.945245	0.741560	0.917270	1.000000

Since, from the above result, 'R&D Spend' is highly correlated with 'Administration' & 'Marketing Spend'. So we will drop one column for our input data, i.e. 'Marketing Spend', remaining columns will be our x_labels and 'Profit' will be our output data as y_label.

Why did we drop 'Marketing Spend'? --> It is because the 'R&D Spend' is highly correlated with 'Marketing Spend' (corr = 0.94)

In [28]:
```
df = pd.DataFrame(data)
```

In [29]:
```
X = companies[['R&D Spend','Administration']]
y = companies['Profit']
```

Data Training and Testing

In [30]:
```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

In [31]:
```
lr = LinearRegression()
```

In [32]:
```
lr.fit(x_train,y_train)
```

Out[32]:
```
▾ LinearRegression
LinearRegression()
```

In [33]:
```
y_pred=lr.predict(x_test)
```

In [34]:
```
print("Coefficient = ",lr.coef_)
print("Intercept = ",lr.intercept_)
```

```
Coefficient =  [0.72620044 0.90205365]
Intercept =  -50908.53791012036
```

Model Accuracy Evaluation after handling Multicollinearity

In [35]:
```
#Finding Coefficient of Determination
r2_2 = r2_score(y_test,y_pred)
print('Coefficient of Determination: ',r2_2)
```

```
Coefficient of Determination:  0.9312577287934511
```

In [36]:
```
#Finding Correlation Coefficient
r_2 = r2_2**(0.5)
print('Correlation Coefficient: ',r_2)
```

```
Correlation Coefficient:  0.9650169577750699
```

Conclusion (Before vs After Multicollinearity Handling)

Here, after dropping one column of 'Marketing Spend', we are getting improvement in our Accuracy Results.

In [37]:
```
#Before handling Multicollinearity

print("Accuracy Before : ",r2*100)
```

```
Accuracy Before :  93.1112023626835
```

In [38]:
```
#After handling Multicollinearity

print("Accuracy After : ",r2_2*100)
```

```
Accuracy After :  93.12577287934512
```