In [ ]:
```
!pip install scikit-learn
```

In [11]:
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\anisha\anaconda3\lib\site-packages (3.4.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\anisha\anaconda3\lib\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\anisha\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\anisha\anaconda3\lib\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: numpy>=1.16 in c:\users\anisha\anaconda3\lib\site-packages (from matplotlib) (1.20.3)
Requirement already satisfied: pillow>=6.2.0 in c:\users\anisha\anaconda3\lib\site-packages (from matplotlib) (9.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\anisha\anaconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: six in c:\users\anisha\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import numpy as np
from sklearn.linear_model import LinearRegression
```

# Multiple Linear Regression on startup dataset

In [3]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\india\anaconda3\lib\site-packages (0.25.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\india\anaconda3\lib\site-packages (from pandas) (2019.3)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\india\anaconda3\lib\site-packages (from pandas) (2.8.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\india\anaconda3\lib\site-packages (from pandas) (1.16.5)
Requirement already satisfied: six>=1.5 in c:\users\india\anaconda3\lib\site-packages (from python-dateutil>=2.6.1->pandas) (1.12.0)
```

In [1]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\Anisha\Downloads\50_Startups.csv")
df
```

Out[1]:

| | 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 |

| | R&D Spend | Administration | Marketing Spend | State | Profit |
|---|---|---|---|---|---|
| **5** | 131876.90 | 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 | 155752.60 |
| **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 | 144259.40 |
| **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 | 124266.90 |
| **19** | 86419.70 | 153514.11 | 0.00 | New York | 122776.86 |
| **20** | 76253.86 | 113867.30 | 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 |
| **25** | 64664.71 | 139553.16 | 137962.62 | California | 107404.34 |
| **26** | 75328.87 | 144135.98 | 134050.07 | Florida | 105733.54 |
| **27** | 72107.60 | 127864.55 | 353183.81 | New York | 105008.31 |
| **28** | 66051.52 | 182645.56 | 118148.20 | Florida | 103282.38 |
| **29** | 65605.48 | 153032.06 | 107138.38 | New York | 101004.64 |
| **30** | 61994.48 | 115641.28 | 91131.24 | Florida | 99937.59 |
| **31** | 61136.38 | 152701.92 | 88218.23 | New York | 97483.56 |
| **32** | 63408.86 | 129219.61 | 46085.25 | California | 97427.84 |
| **33** | 55493.95 | 103057.49 | 214634.81 | Florida | 96778.92 |
| **34** | 46426.07 | 157693.92 | 210797.67 | California | 96712.80 |
| **35** | 46014.02 | 85047.44 | 205517.64 | New York | 96479.51 |
| **36** | 28663.76 | 127056.21 | 201126.82 | Florida | 90708.19 |
| **37** | 44069.95 | 51283.14 | 197029.42 | California | 89949.14 |
| **38** | 20229.59 | 65947.93 | 185265.10 | New York | 81229.06 |
| **39** | 38558.51 | 82982.09 | 174999.30 | California | 81005.76 |
| **40** | 28754.33 | 118546.05 | 172795.67 | California | 78239.91 |

| | R&D Spend | Administration | Marketing Spend | State | Profit |
|---|---|---|---|---|---|
| **41** | 27892.92 | 84710.77 | 164470.71 | Florida | 77798.83 |
| **42** | 23640.93 | 96189.63 | 148001.11 | California | 71498.49 |
| **43** | 15505.73 | 127382.30 | 35534.17 | New York | 69758.98 |
| **44** | 22177.74 | 154806.14 | 28334.72 | California | 65200.33 |
| **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 |

In [2]:

```
shape=df.shape
print("Dataset contains {} rows and {} columns".format(shape[0],shape[1]))
```

```
Dataset contains 50 rows and 5 columns
```

In [3]:

```
df.columns
```

Out[3]:

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

In [4]:

```
df.describe()
```

Out[4]:

| | 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 [5]:

```
c = df.corr()
c
#Inference: We can see that all three columns have a direct relationship with the pr
```

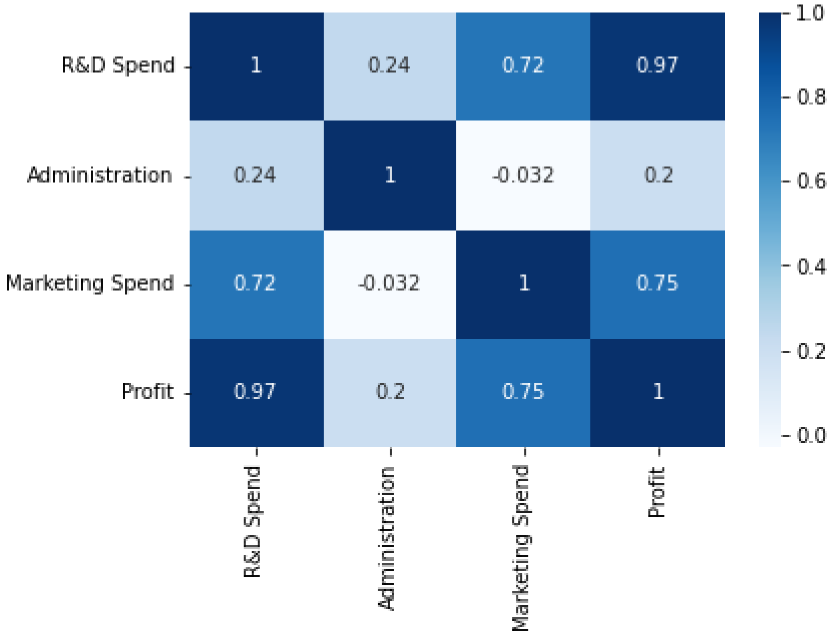
Out[5]:

| | R&D Spend | Administration | Marketing Spend | Profit |
|---|---|---|---|---|
| **R&D Spend** | 1.000000 | 0.241955 | 0.724248 | 0.972900 |
| **Administration** | 0.241955 | 1.000000 | -0.032154 | 0.200717 |
| **Marketing Spend** | 0.724248 | -0.032154 | 1.000000 | 0.747766 |
| **Profit** | 0.972900 | 0.200717 | 0.747766 | 1.000000 |

Correlation Matrix

In [13]:
```
import seaborn as sns # for visualization
import matplotlib as plt
sns.heatmap(c,annot=True,cmap='Blues')
#plt.show()
```

Out[13]: <AxesSubplot:>




In [14]:
```
# spliting Dataset in Dependent & Independent Variables
X = df.iloc[:, :-1].values
y = df.iloc[:, 4].values
```

In [15]:
```
X
```

Out[15]:
```
array([[165349.2, 136897.8, 471784.1, 'New York'],
       [162597.7, 151377.59, 443898.53, 'California'],
       [153441.51, 101145.55, 407934.54, 'Florida'],
       [144372.41, 118671.85, 383199.62, 'New York'],
       [142107.34, 91391.77, 366168.42, 'Florida'],
       [131876.9, 99814.71, 362861.36, 'New York'],
       [134615.46, 147198.87, 127716.82, 'California'],
       [130298.13, 145530.06, 323876.68, 'Florida'],
       [120542.52, 148718.95, 311613.29, 'New York'],
       [123334.88, 108679.17, 304981.62, 'California'],
       [101913.08, 110594.11, 229160.95, 'Florida'],
       [100671.96, 91790.61, 249744.55, 'California'],
       [93863.75, 127320.38, 249839.44, 'Florida'],
       [91992.39, 135495.07, 252664.93, 'California'],
       [119943.24, 156547.42, 256512.92, 'Florida'],
       [114523.61, 122616.84, 261776.23, 'New York'],
       [78013.11, 121597.55, 264346.06, 'California'],
       [94657.16, 145077.58, 282574.31, 'New York'],
       [91749.16, 114175.79, 294919.57, 'Florida'],
       [86419.7, 153514.11, 0.0, 'New York'],
       [76253.86, 113867.3, 298664.47, 'California'],
       [78389.47, 153773.43, 299737.29, 'New York'],
       [73994.56, 122782.75, 303319.26, 'Florida'],
       [67532.53, 105751.03, 304768.73, 'Florida'],
```

```
       [77044.01, 99281.34, 140574.81, 'New York'],
       [64664.71, 139553.16, 137962.62, 'California'],
       [75328.87, 144135.98, 134050.07, 'Florida'],
       [72107.6, 127864.55, 353183.81, 'New York'],
       [66051.52, 182645.56, 118148.2, 'Florida'],
       [65605.48, 153032.06, 107138.38, 'New York'],
       [61994.48, 115641.28, 91131.24, 'Florida'],
       [61136.38, 152701.92, 88218.23, 'New York'],
       [63408.86, 129219.61, 46085.25, 'California'],
       [55493.95, 103057.49, 214634.81, 'Florida'],
       [46426.07, 157693.92, 210797.67, 'California'],
       [46014.02, 85047.44, 205517.64, 'New York'],
       [28663.76, 127056.21, 201126.82, 'Florida'],
       [44069.95, 51283.14, 197029.42, 'California'],
       [20229.59, 65947.93, 185265.1, 'New York'],
       [38558.51, 82982.09, 174999.3, 'California'],
       [28754.33, 118546.05, 172795.67, 'California'],
       [27892.92, 84710.77, 164470.71, 'Florida'],
       [23640.93, 96189.63, 148001.11, 'California'],
       [15505.73, 127382.3, 35534.17, 'New York'],
       [22177.74, 154806.14, 28334.72, 'California'],
       [1000.23, 124153.04, 1903.93, 'New York'],
       [1315.46, 115816.21, 297114.46, 'Florida'],
       [0.0, 135426.92, 0.0, 'California'],
       [542.05, 51743.15, 0.0, 'New York'],
       [0.0, 116983.8, 45173.06, 'California']], dtype=object)
```

In [16]:

```
from sklearn.preprocessing import LabelEncoder
```

In [17]:

```
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X[:, 3])
X1 = pd.DataFrame(X)
X1.head()
```

Out[17]:

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0** | 165349.2 | 136897.8 | 471784.1 | 2 |
| **1** | 162597.7 | 151377.59 | 443898.53 | 0 |
| **2** | 153441.51 | 101145.55 | 407934.54 | 1 |
| **3** | 144372.41 | 118671.85 | 383199.62 | 2 |
| **4** | 142107.34 | 91391.77 | 366168.42 | 1 |

In [18]:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(X,y,train_size=0.7,random_state=0)
x_train
```

Out[18]:

```
array([[130298.13, 145530.06, 323876.68, 1],
       [119943.24, 156547.42, 256512.92, 1],
       [1000.23, 124153.04, 1903.93, 2],
       [542.05, 51743.15, 0.0, 2],
       [65605.48, 153032.06, 107138.38, 2],
       [114523.61, 122616.84, 261776.23, 2],
       [61994.48, 115641.28, 91131.24, 1],
       [63408.86, 129219.61, 46085.25, 0],
       [78013.11, 121597.55, 264346.06, 0],
       [23640.93, 96189.63, 148001.11, 0],
```

```
       [76253.86, 113867.3, 298664.47, 0],
       [15505.73, 127382.3, 35534.17, 2],
       [120542.52, 148718.95, 311613.29, 2],
       [91992.39, 135495.07, 252664.93, 0],
       [64664.71, 139553.16, 137962.62, 0],
       [131876.9, 99814.71, 362861.36, 2],
       [94657.16, 145077.58, 282574.31, 2],
       [28754.33, 118546.05, 172795.67, 0],
       [0.0, 116983.8, 45173.06, 0],
       [162597.7, 151377.59, 443898.53, 0],
       [93863.75, 127320.38, 249839.44, 1],
       [44069.95, 51283.14, 197029.42, 0],
       [77044.01, 99281.34, 140574.81, 2],
       [134615.46, 147198.87, 127716.82, 0],
       [67532.53, 105751.03, 304768.73, 1],
       [28663.76, 127056.21, 201126.82, 1],
       [78389.47, 153773.43, 299737.29, 2],
       [86419.7, 153514.11, 0.0, 2],
       [123334.88, 108679.17, 304981.62, 0],
       [38558.51, 82982.09, 174999.3, 0],
       [1315.46, 115816.21, 297114.46, 1],
       [144372.41, 118671.85, 383199.62, 2],
       [165349.2, 136897.8, 471784.1, 2],
       [0.0, 135426.92, 0.0, 0],
       [22177.74, 154806.14, 28334.72, 0]], dtype=object)
```

In [19]:

```python
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(x_train,y_train)
print('Model has been trained successfully')
```

```
Model has been trained successfully
```

In [20]:

```python
y_pred = model.predict(x_test)
y_pred
```

Out[20]:

```
array([104055.1842384 , 132557.60289702, 133633.01284474,  72336.28081054,
       179658.27210893, 114689.63133397,  66514.82249033,  98461.69321326,
       114294.70487032, 169090.51127461,  96281.907934  ,  88108.30057881,
       110687.1172322 ,  90536.34203081, 127785.3793861 ])
```

In [21]:

```python
testing_data_model_score = model.score(x_test, y_test)
print("Model Score/Performance on Testing data",testing_data_model_score)

training_data_model_score = model.score(x_train, y_train)
print("Model Score/Performance on Training data",training_data_model_score)
```

```
Model Score/Performance on Testing data 0.9355139722149947
Model Score/Performance on Training data 0.9515496105627431
```

In [27]:

```python
# Compare predicted and actual values
df1 = pd.DataFrame(data={'Predicted value':y_pred.flatten(),'Actual Value':y_test.fl
df1
```

Out[27]:

| | Predicted value | Actual Value |
|---|---|---|
| 0 | 104055.184238 | 103282.38 |
| 1 | 132557.602897 | 144259.40 |

| | Predicted value | Actual Value |
|---|---|---|
| 2 | 133633.012845 | 146121.95 |
| 3 | 72336.280811 | 77798.83 |
| 4 | 179658.272109 | 191050.39 |
| 5 | 114689.631334 | 105008.31 |
| 6 | 66514.822490 | 81229.06 |
| 7 | 98461.693213 | 97483.56 |
| 8 | 114294.704870 | 110352.25 |
| 9 | 169090.511275 | 166187.94 |
| 10 | 96281.907934 | 96778.92 |
| 11 | 88108.300579 | 96479.51 |
| 12 | 110687.117232 | 105733.54 |
| 13 | 90536.342031 | 96712.80 |
| 14 | 127785.379386 | 124266.90 |

Model evaluation

R2 score: R2 score – R squared score. It is one of the statistical approaches by which we can find the variance or the spread of the target and feature data.

In [22]:

```
from sklearn.metrics import r2_score

r2Score = r2_score(y_pred, y_test)
print("R2 score of model is :" ,r2Score*100)
```

```
R2 score of model is : 93.39448007716634
```

MSE: MSE – Mean Squared Error. By using this approach we can find that how much the regression best fit line is close to all the residual.

In [29]:

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_pred, y_test)
print("Mean Squarred Error is :" ,mse*100)
```

```
Mean Squarred Error is : 6224496238.946446
```

RMSE: RMSE – Root Mean Squared Error. This is similar to the Mean squared error(MSE) approach, the only difference is that here we find the root of the mean squared error i.e. root of the Mean squared error is equal to Root Mean Squared Error. The reason behind finding the root is to find the more close residual to the values found by mean squared error.

In [30]:

```
import numpy as np
rmse = np.sqrt(mean_squared_error(y_pred, y_test))
print("Root Mean Squarred Error is : ",rmse*100)
```

```
Root Mean Squarred Error is :  788954.7666974607
```

MAE: MAE – Mean Absolute Error. By using this approach we can find the difference between the actual values and predicted values but that difference is absolute i.e. the difference is positive.

In [31]:

```
from sklearn.metrics import mean_absolute_error

mae = mean_absolute_error(y_pred,y_test)
print("Mean Absolute Error is :" ,mae)
```

```
Mean Absolute Error is : 6503.57732358002
```

In [ ]: