In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
from statsmodels.graphics.regressionplots import influence_plot
import warnings
warnings.filterwarnings ("ignore")
```

In [2]:
```
data1 = pd.read_csv(r"C:\Users\vishal gajarmal\Downloads\50_Startups.csv")
```

In [3]:
```
data1.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]:
```
data1.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]:
```
data1.isna().sum()
```

Out[5]:
```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

In [6]:
```
data1.corr()
```

Out[6]:

| | 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 |

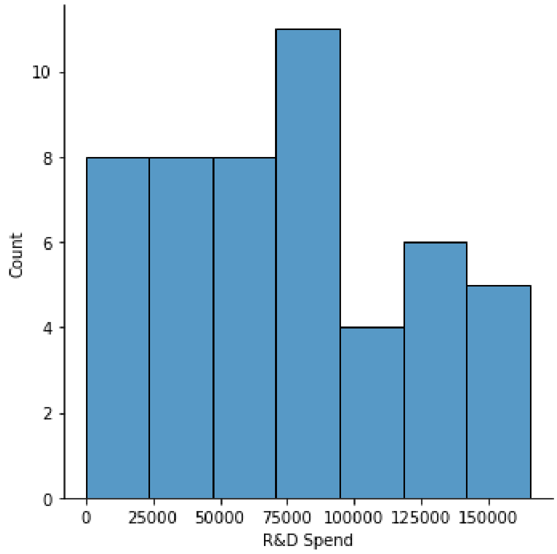
In [7]:
```
data1.describe()
```

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 |

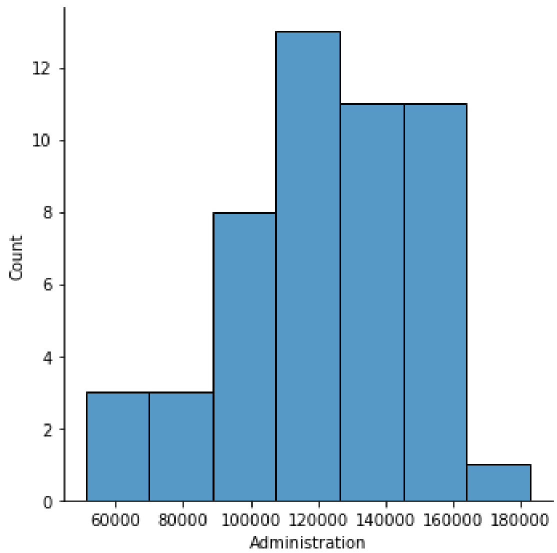
In [8]:
```
sns.displot(data1['R&D Spend'])
```

Out[8]:<seaborn.axisgrid.FacetGrid at 0x23d298c0df0>




In [9]:
```
sns.displot(data1['Administration'])
```

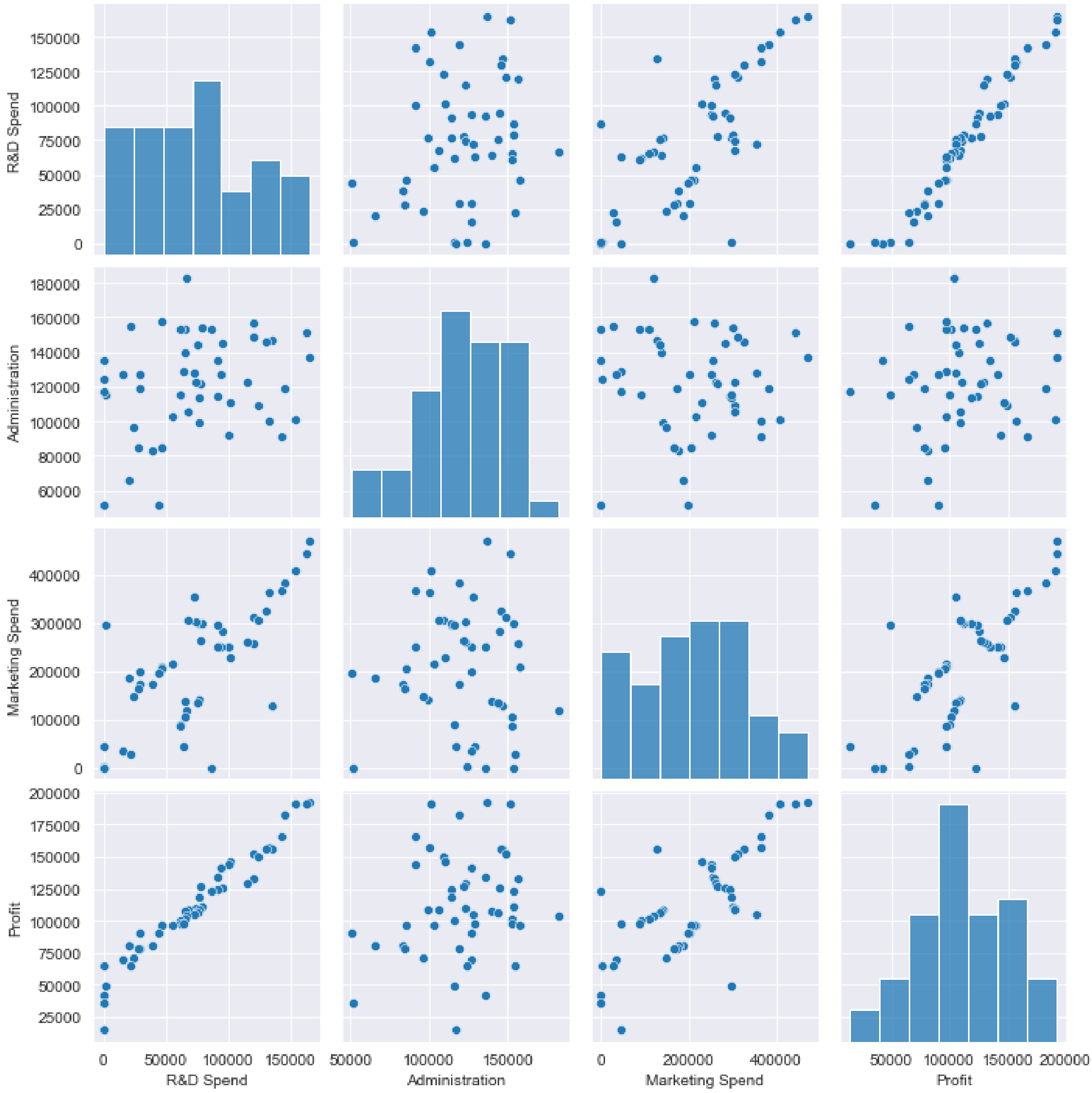
Out[9]:<seaborn.axisgrid.FacetGrid at 0x23d2a0090d0>




# Use scatterplot between variables along with hist

In [10]:
```
sns.set_style(style='darkgrid')
sns.pairplot(data1)
```

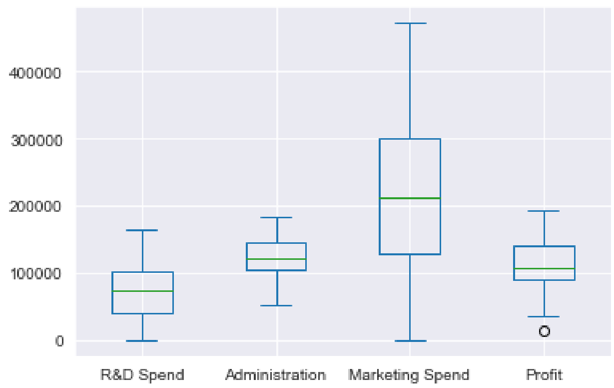
Out[10]:<seaborn.axisgrid.PairGrid at 0x23d2a13abb0>




# Finding outliers

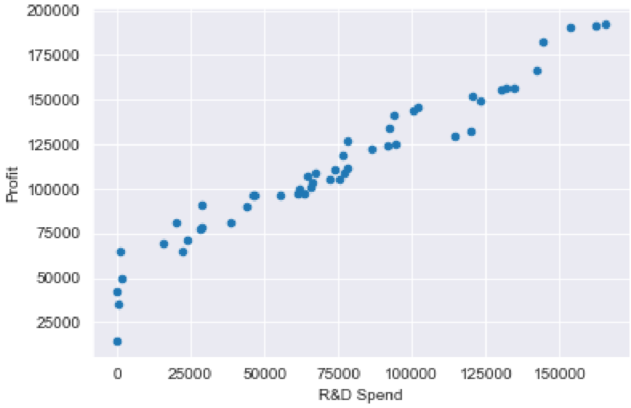
In [11]: data1.plot(kind='box')

Out[11]:<AxesSubplot:>




In [12]: data1.plot(kind='scatter',x='R&D Spend',y='Profit')

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




In [18]:
```
data1=data1.rename({'R&D Spend':'RDS','Administration':'ADMS','Marketing Spend':'MKTS'},axis=1)
```

In [19]:
```
data1["State"]=data1['State'].astype("category")
```

In [20]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RDS     50 non-null     float64
 1   ADMS    50 non-null     float64
 2   MKTS    50 non-null     float64
 3   State   50 non-null     category
 4   Profit  50 non-null     float64
dtypes: category(1), float64(4)
memory usage: 1.9 KB
```

# Preparing a model

In [21]:
```
model=smf.ols("Profit~MKTS+ADMS+RDS",data=data1).fit()
```

In [22]:
```
model.fittedvalues
```

```
Out[22]:0     192521.252890
        1     189156.768232
        2     182147.279096
        3     173696.700026
        4     172139.514183
        5     163580.780571
        6     158114.096669
        7     160021.363048
        8     151741.699699
        9     154884.684110
        10    135509.016367
        11    135573.712961
        12    129138.054182
        13    127487.991663
        14    149548.646335
        15    146235.159985
        16    116915.405401
        17    130192.447208
        18    129014.226806
        19    115635.216367
        20    116639.669231
        21    117319.451640
        22    114706.981717
        23    109996.615221
        24    113362.966113
        25    102237.725065
        26    110600.575350
        27    114408.071457
        28    101660.026005
        29    101794.983452
        30     99452.372936
        31     97687.856276
        32     99001.328985
        33     97915.007805
        34     89039.273741
        35     90511.599568
        36     75286.174585
        37     89619.537708
        38     69697.430648
        39     83729.011977
        40     74815.953991
        41     74802.556239
        42     70620.411821
        43     60167.039963
        44     64611.354916
        45     47650.649687
        46     56166.206853
        47     46490.588983
        48     49171.388158
        49     48215.134111
        dtype: float64
In [23]: model.resid
```

```
Out[23]:0      -259.422890
        1      2635.291768
        2      8903.110904
        3      9205.289974
        4     -5951.574183
        5     -6589.660571
        6     -1991.586669
        7     -4268.763048
        8       470.070301
        9     -5124.724110
        10    10612.933633
        11     8685.687039
        12    12447.465818
        13     6819.358337
        14   -16945.996335
        15   -16318.119985
        16    10077.524599
        17    -4822.077208
        18    -4747.326806
        19     7141.643633
        20     1834.360769
        21    -6006.431640
        22    -4354.731717
        23    -1262.625221
        24    -4810.926113
        25     5166.614935
        26    -4867.035350
        27    -9399.761457
        28     1622.353995
        29     -790.343452
        30      485.217064
        31     -204.296276
        32    -1573.488985
        33    -1136.087805
        34     7673.526259
        35     5967.910432
        36    15422.015415
        37      329.602292
        38    11531.629352
        39    -2723.251977
        40     3423.956009
        41     2996.273761
        42      878.078179
        43     9591.940037
        44      588.975084
        45    17275.430313
        46    -6675.456853
        47    -3930.858983
        48   -13497.978158
        49   -33533.734111
        dtype: float64
```

In [24]:

```
model.params
```

```
Out[24]:Intercept    50122.192990
        MKTS             0.027228
        ADMS            -0.026816
        RDS              0.805715
        dtype: float64
```

In [25]:

```
(model.rsquared,model.rsquared_adj)
```

```
Out[25]:(0.9507459940683246, 0.9475337762901719)
```

# t and p - values

In [26]:

```
print(model.tvalues,'\n',model.pvalues)
```

```
Intercept     7.626218
MKTS          1.655077
ADMS         -0.525507
RDS          17.846374
dtype: float64
 Intercept    1.057379e-09
MKTS         1.047168e-01
ADMS         6.017551e-01
RDS          2.634968e-22
dtype: float64
```

# Build SLR and MLR models for insignificant variables 'ADMS' and 'MKTS'

# Also find their tvalues and pvalues

In [27]:
```
slr_a=smf.ols("Profit~ADMS",data=data1).fit()
slr_a.tvalues , slr_a.pvalues
```

Out[27]:
```
(Intercept    3.040044
 ADMS         1.419493
 dtype: float64,
 Intercept    0.003824
 ADMS         0.162217
 dtype: float64)
```

In [28]:
```
slr_m=smf.ols("Profit~MKTS",data=data1).fit()
slr_m.tvalues , slr_m.pvalues
```

Out[28]:
```
(Intercept    7.808356
 MKTS         7.802657
 dtype: float64,
 Intercept    4.294735e-10
 MKTS         4.381073e-10
 dtype: float64)
```

In [29]:
```
mlr_am=smf.ols("Profit~ADMS+MKTS",data=data1).fit()
mlr_am.tvalues , mlr_am.pvalues
```

Out[29]:
```
(Intercept    1.142741
 ADMS         2.467779
 MKTS         8.281039
 dtype: float64,
 Intercept    2.589341e-01
 ADMS         1.729198e-02
 MKTS         9.727245e-11
 dtype: float64)
```

# calculating VIF(variance inflation factor)

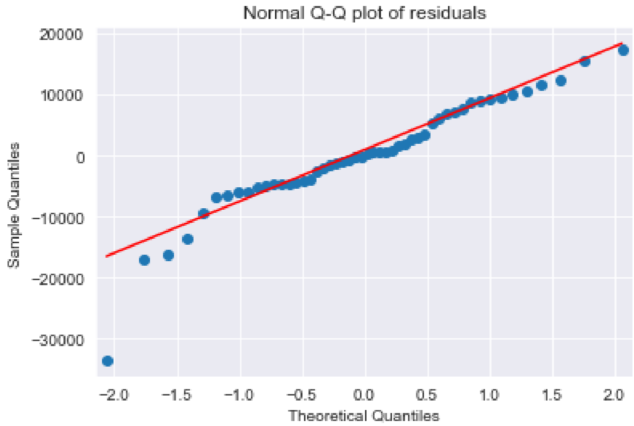
In [30]:
```
rsq_r=smf.ols("RDS~ADMS+MKTS",data=data1).fit().rsquared
vif_r=1/(1-rsq_r)

rsq_a=smf.ols("ADMS~RDS+MKTS",data=data1).fit().rsquared
vif_a=1/(1-rsq_a)

rsq_m=smf.ols("MKTS~RDS+ADMS",data=data1).fit().rsquared
vif_m=1/(1-rsq_m)

# Putting the values in Dataframe format
d1={'Variables':['RDS','ADMS','MKTS'],'Vif':[vif_r,vif_a,vif_m]}
Vif_df=pd.DataFrame(d1)
Vif_df
```

Out[30]:

| | Variables | Vif |
|---|---|---|
| **0** | RDS | 2.468903 |
| **1** | ADMS | 1.175091 |
| **2** | MKTS | 2.326773 |

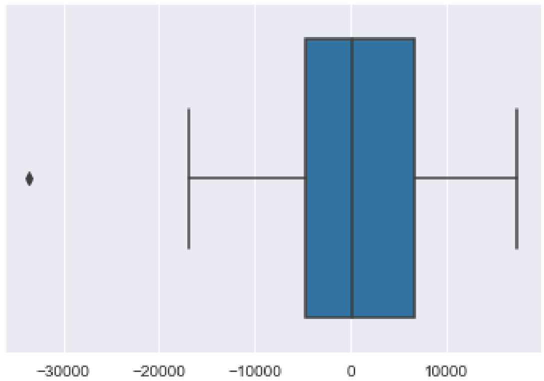
# Test for Normality of Residuals (Q-Q Plot) using residual model (model.resid)

In [32]:
```
import statsmodels.api as sm
sm.qqplot(model.resid,line='q')
plt.title("Normal Q-Q plot of residuals")
plt.show()
```



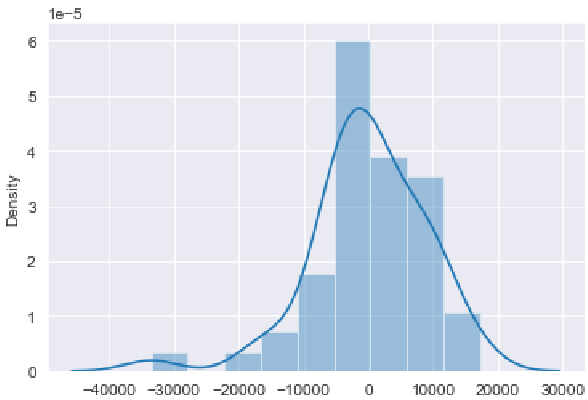

In [33]:
```
sns.boxplot(model.resid)
```

Out[33]:<AxesSubplot:>




In [34]:
```
sns.distplot(model.resid)
```

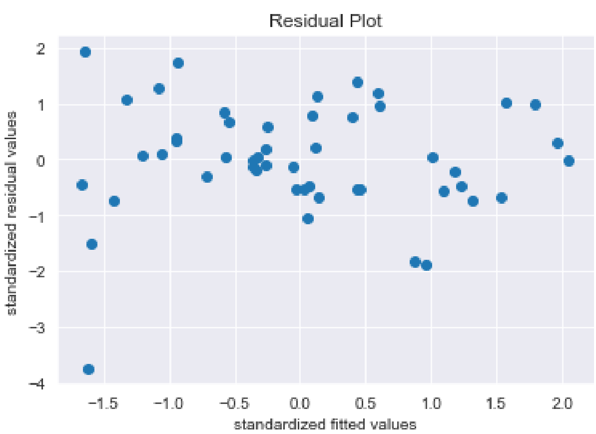
Out[34]:<AxesSubplot:ylabel='Density'>




# Residual plot for Homoscedasticity

In [35]:
```
def standard_values(vals) : return (vals-vals.mean())/vals.std()
```

In [36]:
```
plt.scatter(standard_values(model.fittedvalues),standard_values(model.resid))
plt.title('Residual Plot')
plt.xlabel('standardized fitted values')
plt.ylabel('standardized residual values')
plt.show()
```



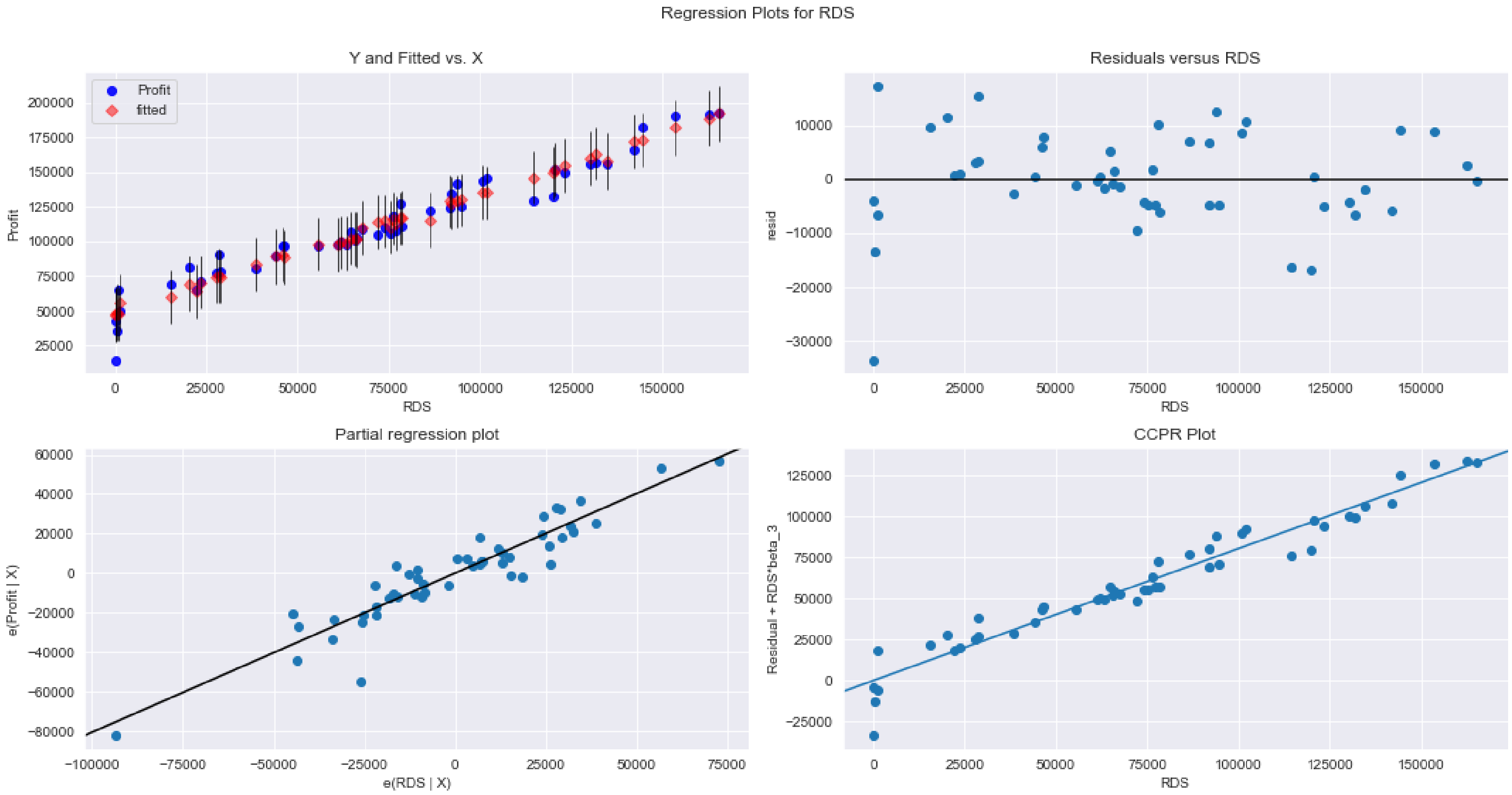

In [39]:
```
standard_values(model.resid).mean()
```

Out[39]:-1.7763568394002505e-17

# Residual vs Regression Plot

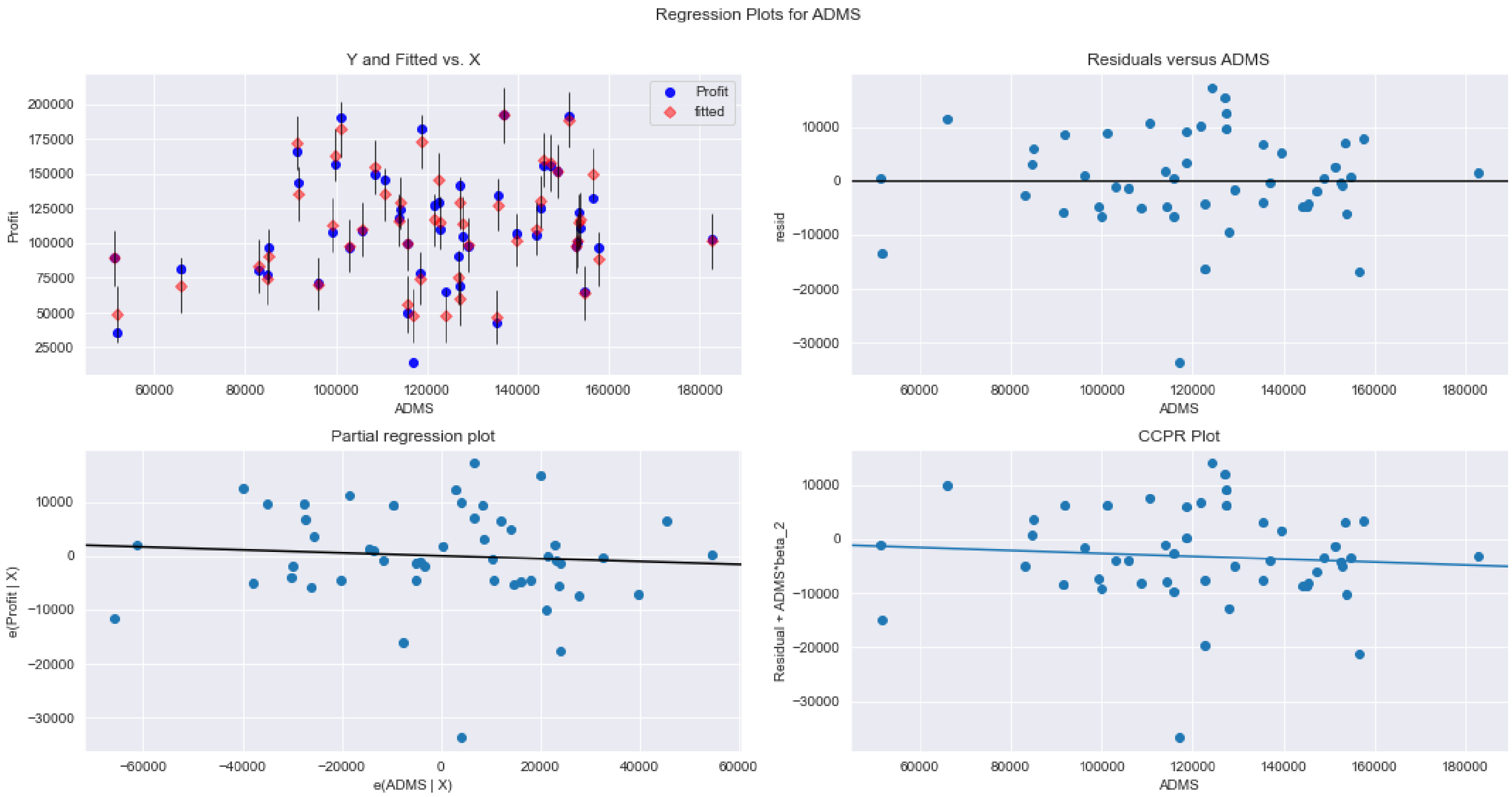
In [42]:
```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'RDS',fig=fig)
plt.show()
```

eval_env: 1



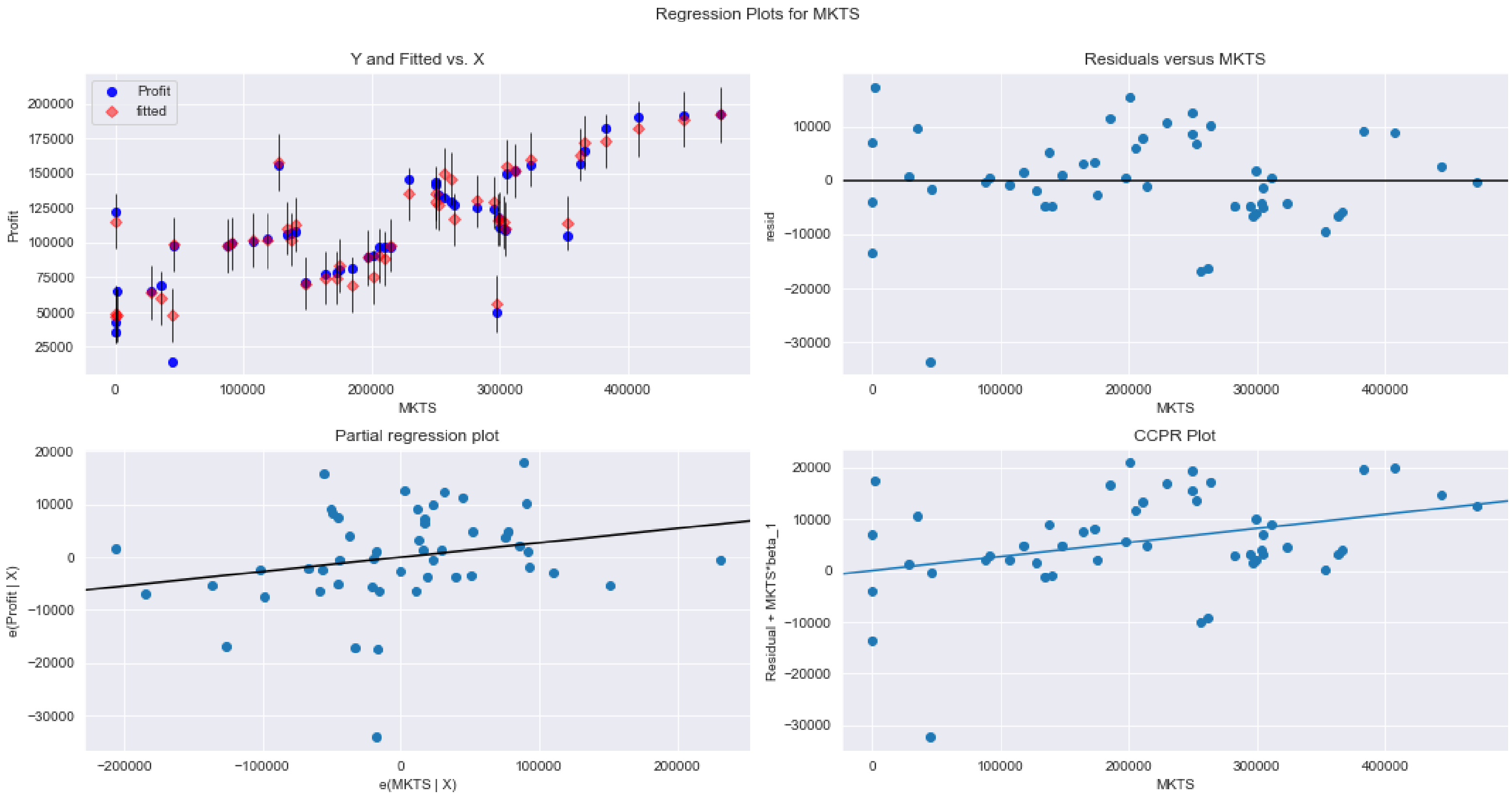

In [43]:
```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'ADMS',fig=fig)
plt.show()
```

eval_env: 1




In [44]:
```
fig=plt.figure(figsize=(15,8))
sm.graphics.plot_regress_exog(model,'MKTS',fig=fig)
plt.show()
```

eval_env: 1



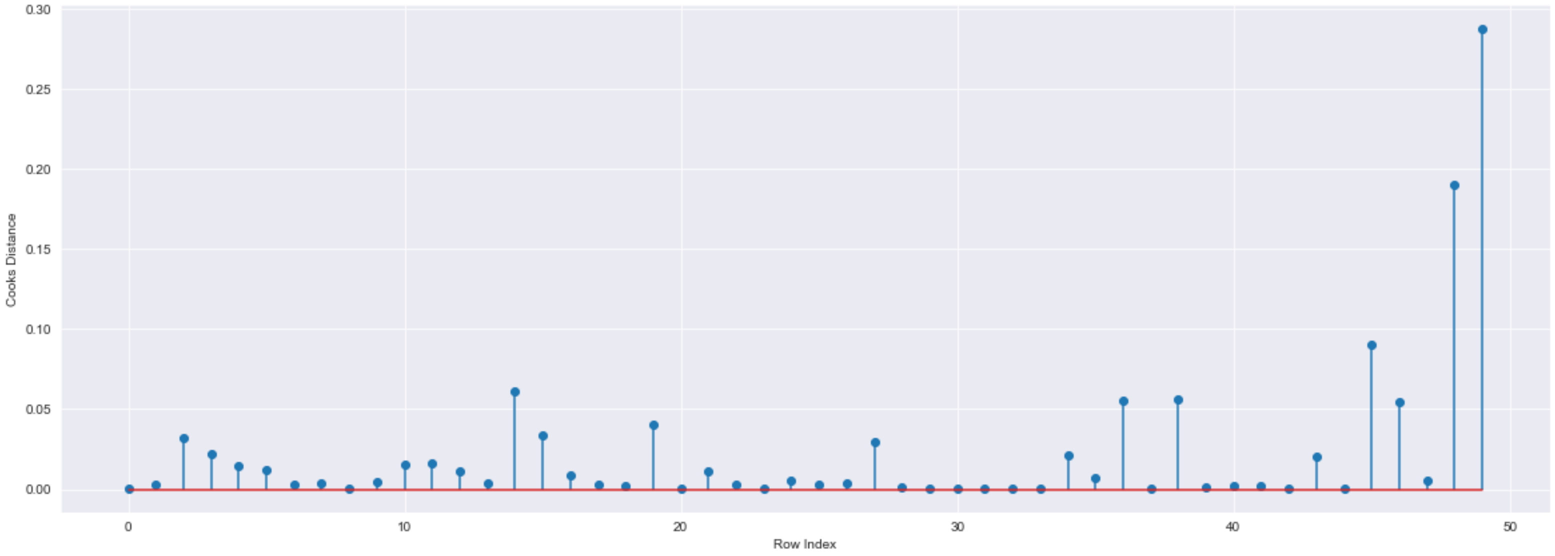

# Model Deletion Diagnostics

# Cook Distance

In [45]:
```
(c,_)=model.get_influence().cooks_distance
c
```

Out[45]:
```
array([3.21825244e-05, 3.27591036e-03, 3.23842699e-02, 2.17206555e-02,
       1.44833032e-02, 1.17158463e-02, 2.91766303e-03, 3.56513444e-03,
       4.04303948e-05, 4.86758017e-03, 1.51064757e-02, 1.63564959e-02,
       1.15516625e-02, 4.01422811e-03, 6.12934253e-02, 3.40013448e-02,
       8.33556413e-03, 3.30534399e-03, 2.16819303e-03, 4.07440577e-02,
       4.25137222e-04, 1.09844352e-02, 2.91768000e-03, 2.76030254e-04,
       5.04643588e-03, 3.00074623e-03, 3.41957068e-03, 2.98396413e-02,
       1.31590664e-03, 1.25992620e-04, 4.18505125e-05, 9.27434786e-06,
       7.08656521e-04, 1.28122674e-04, 2.09815032e-02, 6.69508674e-03,
       5.55314705e-02, 6.55050578e-05, 5.61547311e-02, 1.54279607e-03,
       1.84850929e-03, 1.97578066e-03, 1.36089280e-04, 2.05553171e-02,
       1.23156041e-04, 9.03234206e-02, 5.45303387e-02, 5.33885616e-03,
       1.90527441e-01, 2.88082293e-01])
```

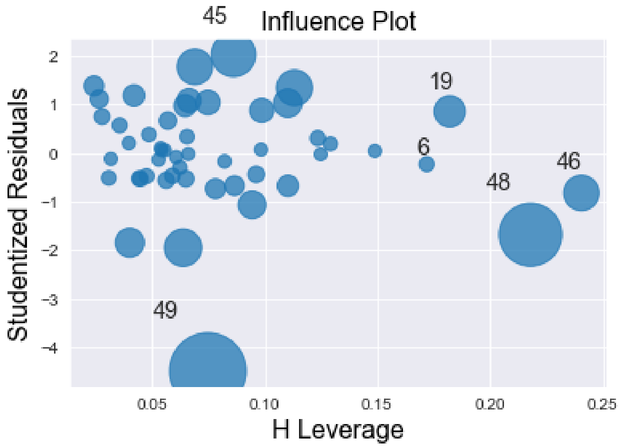
In [46]:
```
# Plot the influencers using the stem plot
fig=plt.figure(figsize=(20,7))
plt.stem(np.arange(len(data1)),np.round(c,5))
plt.xlabel('Row Index')
plt.ylabel('Cooks Distance')
plt.show()
```




In [47]:
```
# Index and value of influencer where C>0.5
np.argmax(c) , np.max(c)
```

Out[47]:(49, 0.2880822927543262)

In [48]:
```
influence_plot(model)
plt.show()
```




In [49]:
```
k=data1.shape[1]
n=data1.shape[0]
leverage_cutoff = (3*(k+1))/n
leverage_cutoff
```

Out[49]:0.36

In [51]:
```
data1[data1.index.isin([44,45,46,47,48,49])]
```

Out[51]:

| | RDS | ADMS | MKTS | State | Profit |
|---|---|---|---|---|---|
| **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 [52]:
```
data1.head()
```

Out[52]:

| | RDS | ADMS | MKTS | 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 |

# Improving the model

In [53]:
```
data2=data1.drop(data1.index[[49]],axis=0).reset_index(drop=True)
data2
```

Out[53]:

| | RDS | ADMS | MKTS | 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 |
| **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 |
| **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 |

# Building our final model

In [56]:
```
while np.max(c)>0.5 :
    model=smf.ols("Profit~RDS+ADMS+MKTS",data=data2).fit()
    (c,_)=model.get_influence().cooks_distance
    c
    np.argmax(c) , np.max(c)
    data2=data2.drop(data2.index[[np.argmax(c)]],axis=0).reset_index(drop=True)
    data2
else:
    final_model=smf.ols("Profit~RDS+ADMS+MKTS",data=data2).fit()
    final_model.rsquared , final_model.aic
    print("Thus model accuracy is improved to",final_model.rsquared)
```

Thus model accuracy is improved to 0.9613162435129847

In [57]:
```
final_model.rsquared
```

Out[57]:0.9613162435129847

In [58]:
```
data2
```

Out[58]:

| | RDS | ADMS | MKTS | 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 |
| 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 |
| 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 |

# Predict for new data

In [59]:
```
new_data=pd.DataFrame({'RDS':70000,"ADMS":90000,"MKTS":140000},index=[0])
```

```
new_data
```

Out[59]:

| | RDS | ADMS | MKTS |
|---|---|---|---|
| **0** | 70000 | 90000 | 140000 |

In [60]:

```
final_model.predict(new_data)
```

Out[60]:

```
0    108727.154753
dtype: float64
```

In [61]:

```
pred_y=final_model.predict(data2)
pred_y
```

Out[61]:

```
0     190716.676999
1     187537.122227
2     180575.526396
3     172461.144642
4     170863.486721
5     162582.583177
6     157741.338633
7     159347.735318
8     151328.826941
9     154236.846778
10    135507.792682
11    135472.855621
12    129355.599449
13    127780.129139
14    149295.404796
15    145937.941975
16    117437.627921
17    130408.626295
18    129129.234457
19    116641.003121
20    117097.731866
21    117911.019038
22    115248.217796
23    110603.139045
24    114051.073877
25    103398.054385
26    111547.638935
27    114916.165026
28    103027.229434
29    103057.621761
30    100656.410227
31     99088.213693
32    100325.741335
33     98962.303136
34     90552.307809
35     91709.288672
36     77080.554255
37     90722.503244
38     71433.021956
39     85147.375646
40     76625.510303
41     76492.145175
42     72492.394974
43     62592.049718
44     67025.731107
45     50457.297206
46     58338.443625
47     49375.776655
48     51658.096812
dtype: float64
```

In [ ]: