In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings("ignore")
```

In [3]:

```
df = pd.read_csv("Salary_Data.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [5]:

```
df.tail()
```

Out[5]:

	YearsExperience	Salary
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [6]:

```
df.info()
```

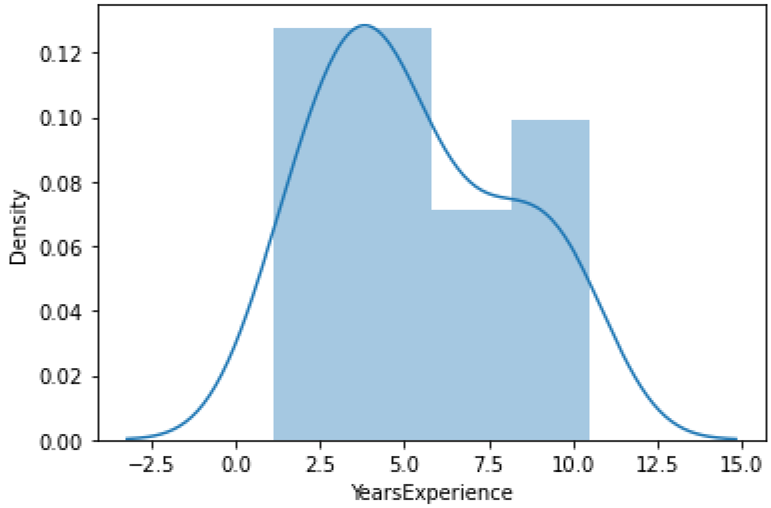
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

In [7]:

```
sns.distplot(df.YearsExperience,kde=True)
```

Out[7]:

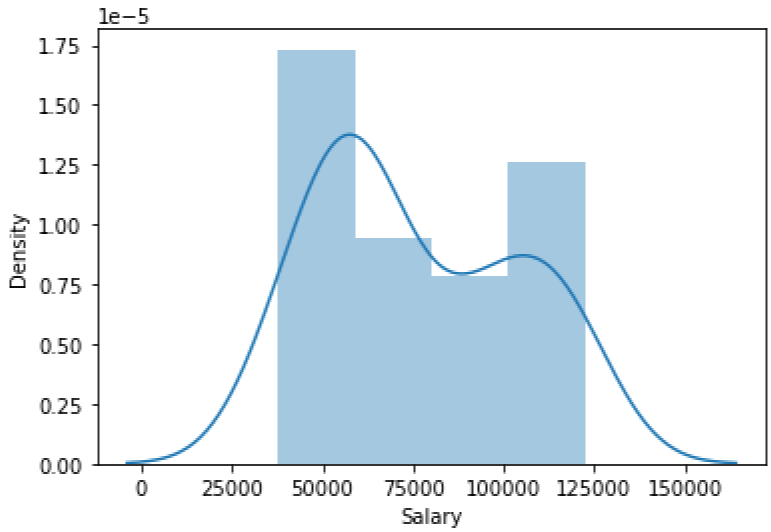
```
<AxesSubplot:xlabel='YearsExperience', ylabel='Density'>
```



In [8]:

```
sns.distplot(df.Salary,kde=True)
```

Out[8]: <AxesSubplot:xlabel='Salary', ylabel='Density'>

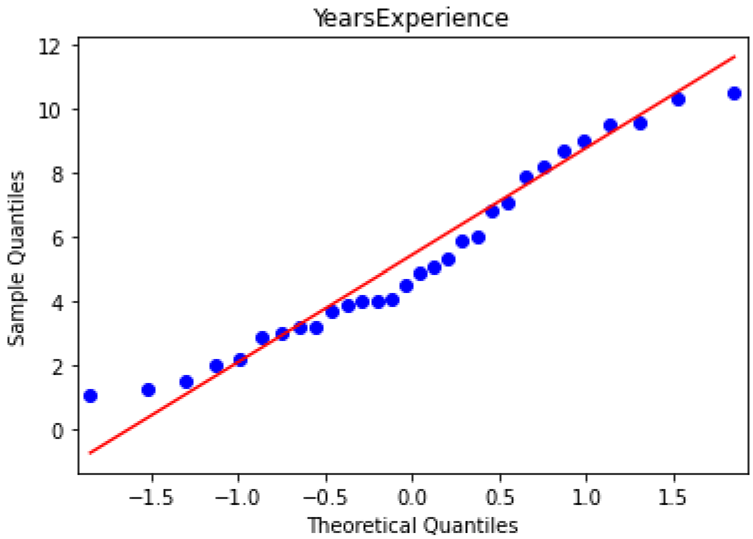


Raw Data

In [9]:

```
for feature in df:
    data = df.copy()
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```

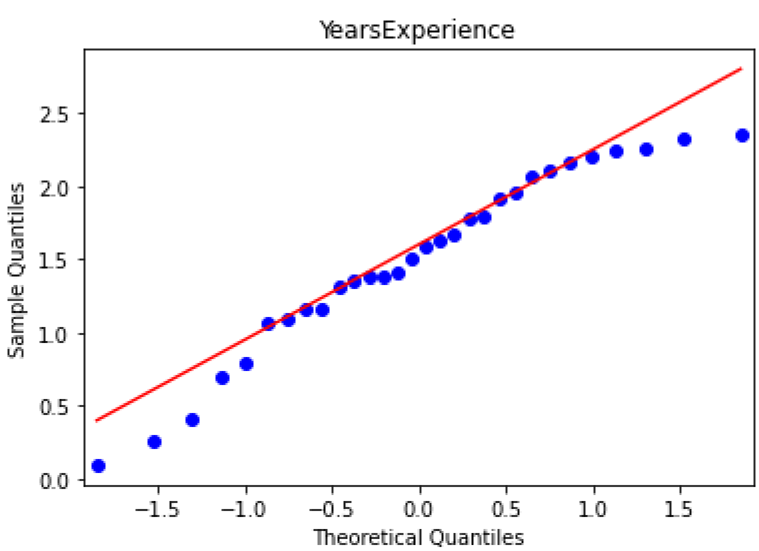
Loading [MathJax]/extensions/Safe.js

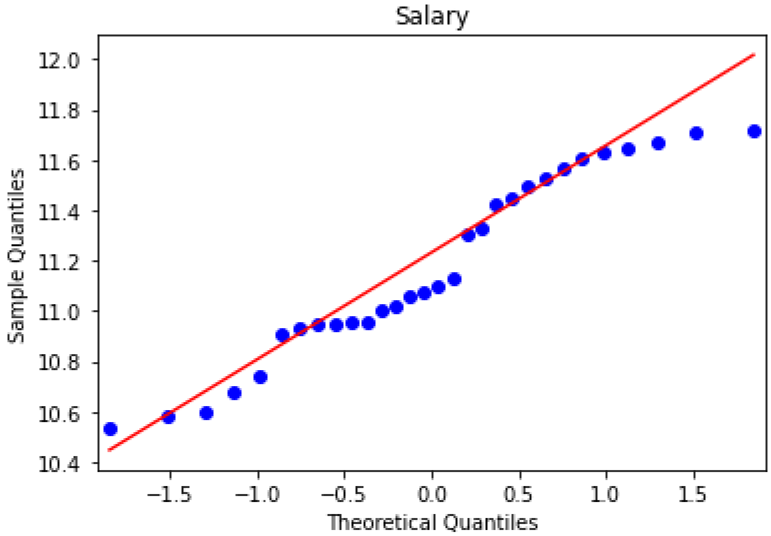


Log Transformation

In [10]:

```
for feature in df:
    data = df.copy()
    data[feature]=np.log(data[feature])
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```

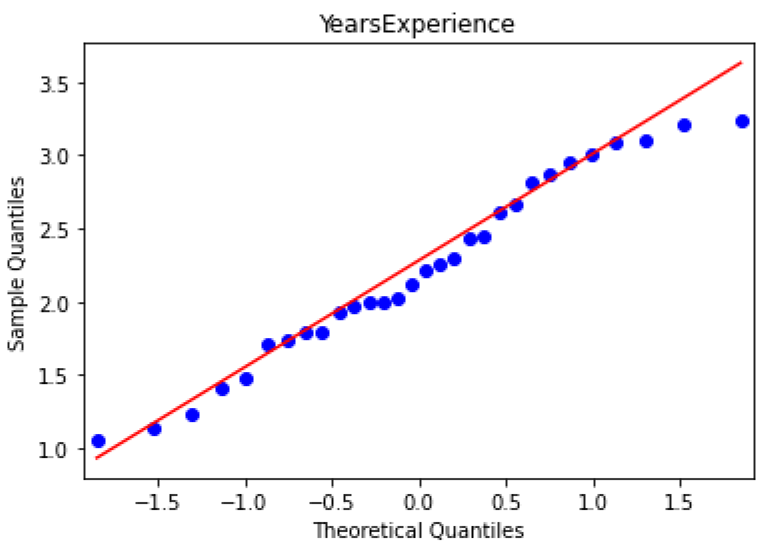




Sqareroot Transformation

In [11]:

```
for feature in df:
    data = df.copy()
    data[feature]=np.sqrt(data[feature])
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```

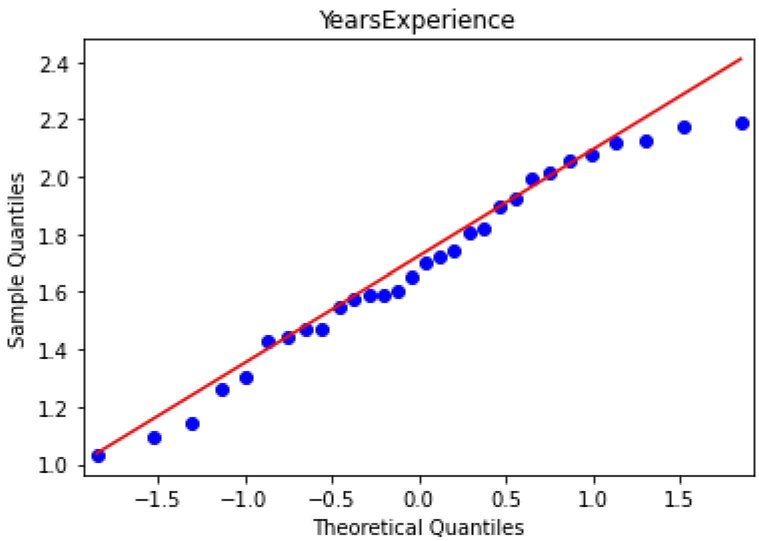


Loading [MathJax]/extensions/Safe.js

CubeRoot transformation

In [12]:

```
for feature in df:
    data = df.copy()
    data[feature]=np.cbrt(data[feature])
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```



Observation:

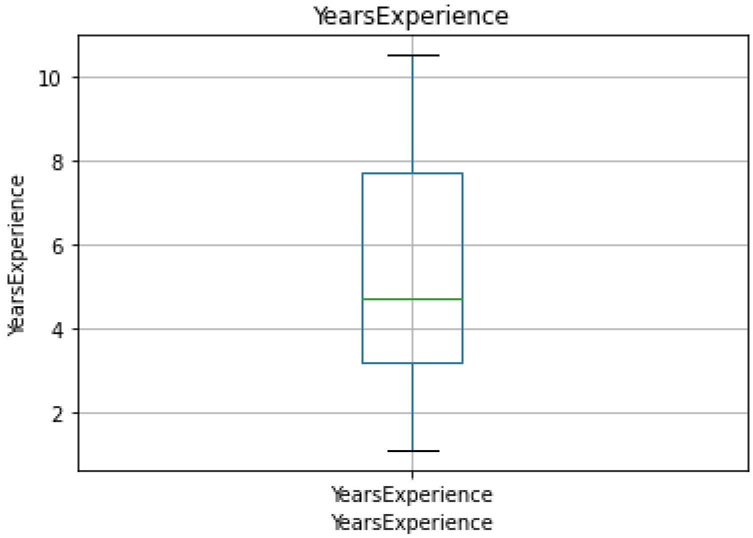
As we Does not see any much of differnce between the raw data and transformation so we can consider raw data for my predictions

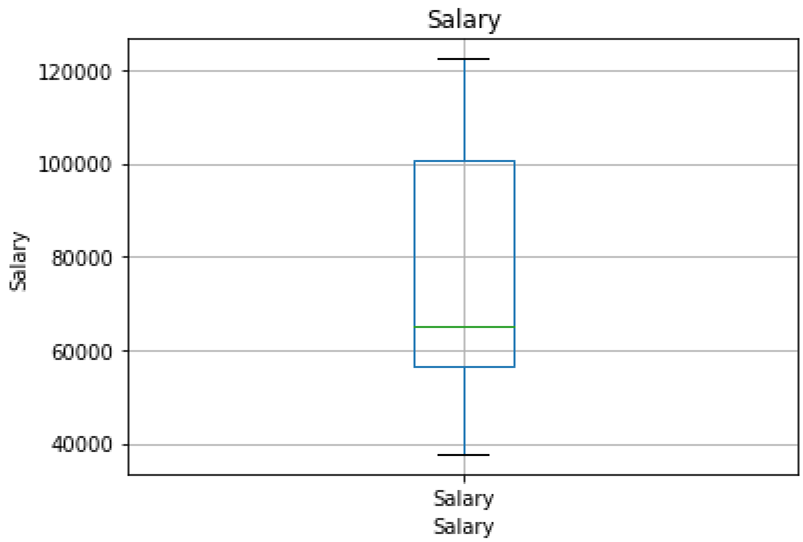
Boxplot using Raw Data

In [13]:

```
for feature in df:
    data = df.copy()
    data.boxplot(column=feature)
    plt.xlabel(feature)
    plt.ylabel(feature)
    plt.title(feature)
    plt.show()
```

Loading [MathJax]/extensions/Safe.js

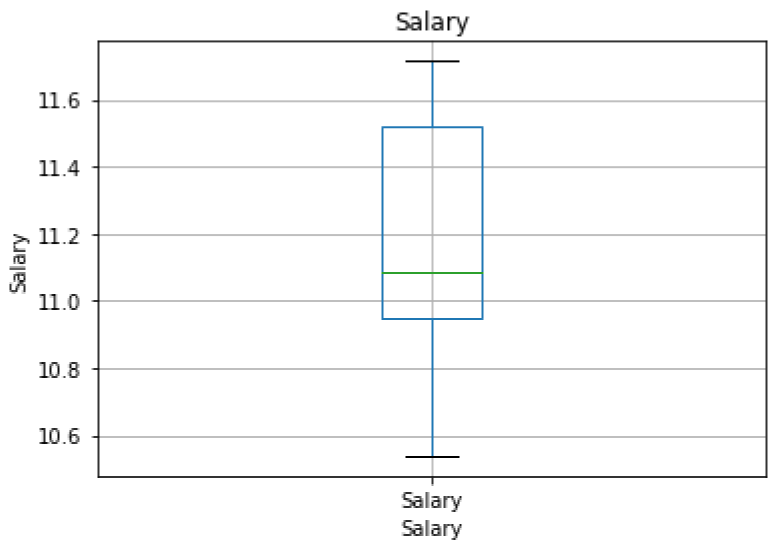




Boxplot using Log Transformation

In [15]:

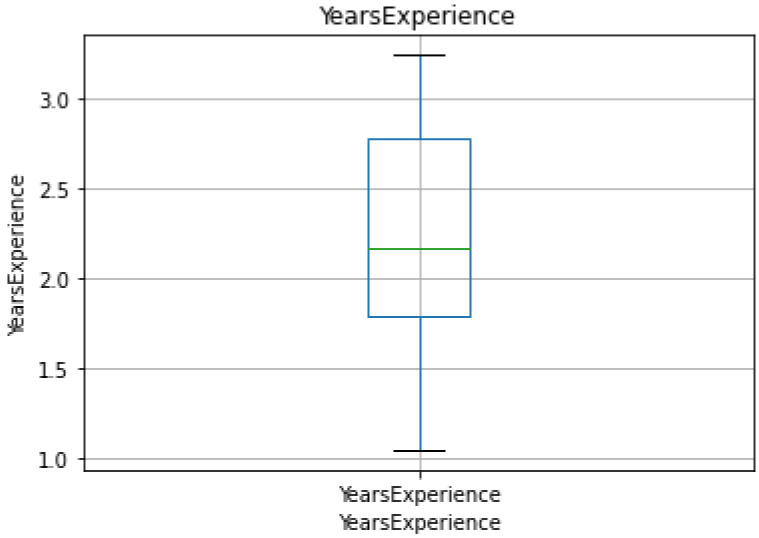
```
for feature in df:
    data = df.copy()
    data[feature]=np.log(data[feature])
    data.boxplot(column=feature)
    plt.xlabel(feature)
    plt.ylabel(feature)
    plt.title(feature)
    plt.show()
```

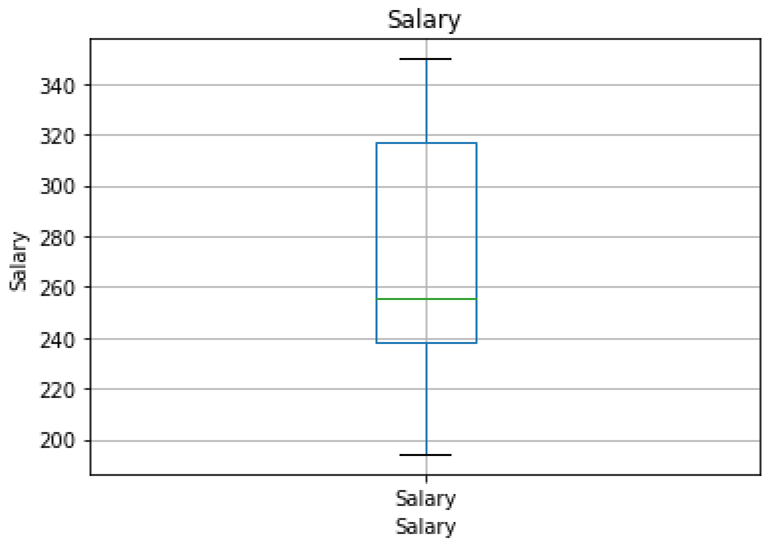


BoxPlot using Squareroot Transformation

In [16]:

```
for feature in df:
    data = df.copy()
    data[feature]=np.sqrt(data[feature])
    data.boxplot(column=feature)
    plt.xlabel(feature)
    plt.ylabel(feature)
    plt.title(feature)
    plt.show()
```





Boxplot using Cuberoot Transformation

In [17]:

```
for feature in df:
    data = df.copy()
    data[feature]=np.cbrt(data[feature])
    data.boxplot(column=feature)
    plt.xlabel(feature)
    plt.ylabel(feature)
    plt.title(feature)
    plt.show()
```

Observation:

As we can see there is no outliers in the raw data as well as after transformation of the data

Checking Colinearity

In [15]:

```
df.corr()
```

Out[15]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

In [16]:

```
df.describe()
```

Out[16]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785

Loading [MathJax]/extensions/Safe.js

	YearsExperience	Salary
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [17]:

```
df.duplicated()
```

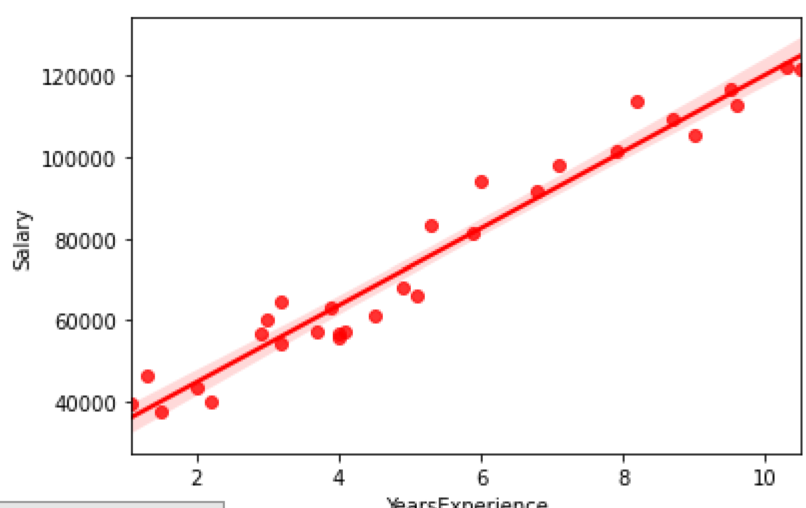
Out[17]:

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
dtype: bool
```

In [18]:

```
sns.regplot(x="YearsExperience",y="Salary",data=df,color='red')
```

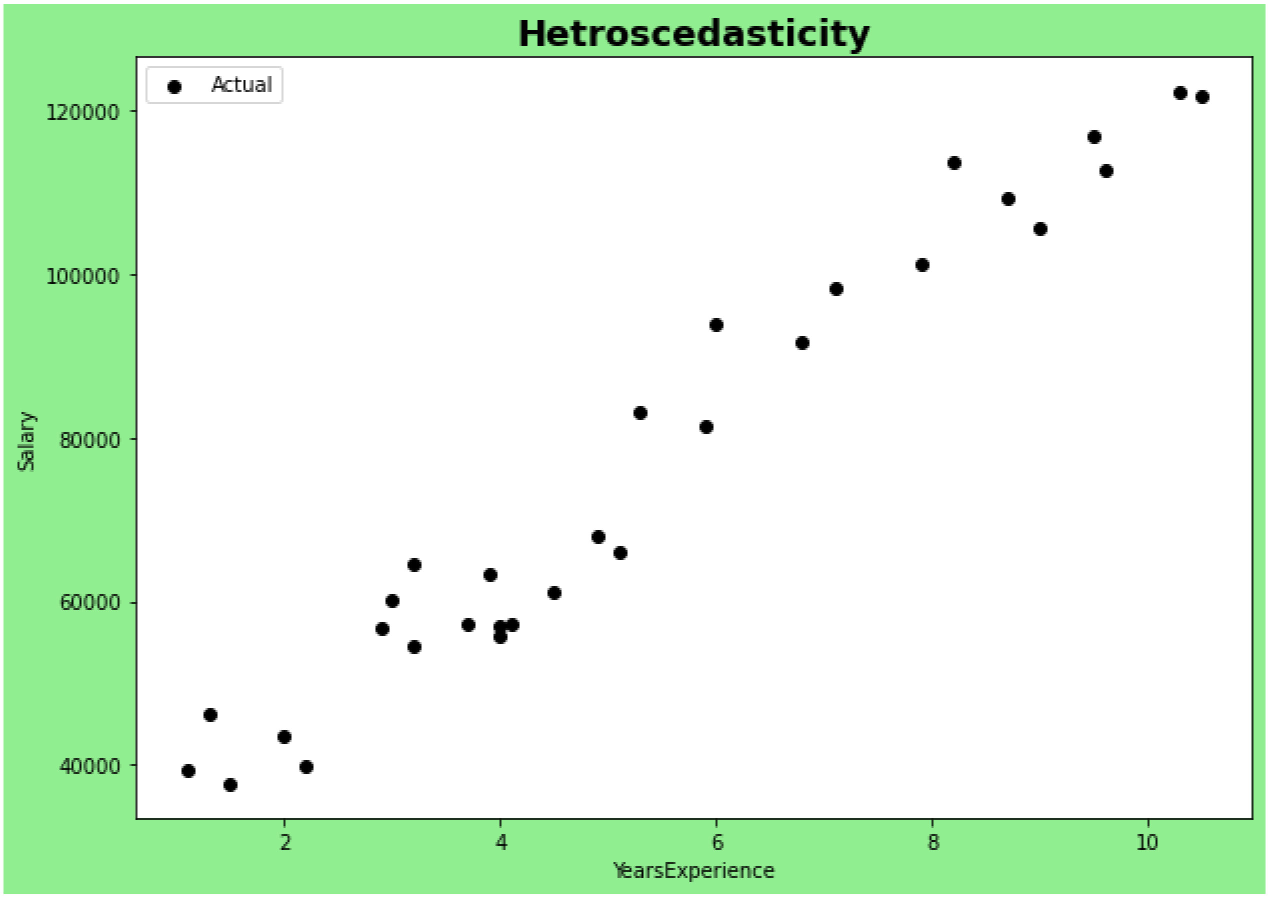
Out[18]: <AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>



In [19]:

```
plt.figure(figsize=(10,7),facecolor='lightgreen')
plt.scatter(df.YearsExperience,df.Salary,color='black',label="Actual")
plt.xlabel("YearsExperience")
plt.ylabel("Salary")
plt.legend(loc='best')
plt.title("Hetroscedasticity",fontsize=18,fontweight="bold")
```

Out[19]: Text(0.5, 1.0, 'Hetroscedasticity')



Create a Model

In [20]:

```
model = smf.ols("Salary~YearsExperience",data=df).fit()
```

In [21]:

```
model.summary()
```

Out[21]:

OLS Regression Results

Dep. Variable:	Salary	**R-squared:**	0.957
Model:	OLS	**Adj. R-squared:**	0.955
Method:	Least Squares	**F-statistic:**	622.5
Date:	Sun, 10 Apr 2022	**Prob (F-statistic):**	1.14e-20
Time:	20:08:02	**Log-Likelihood:**	-301.44
No. Observations:	30	**AIC:**	606.9
Df Residuals:	28	**BIC:**	609.7
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.579e+04	2273.053	11.347	0.000	2.11e+04	3.04e+04
YearsExperience	9449.9623	378.755	24.950	0.000	8674.119	1.02e+04

Omnibus:	2.140	**Durbin-Watson:**	1.648
Prob(Omnibus):	0.343	**Jarque-Bera (JB):**	1.569
Skew:	0.363	**Prob(JB):**	0.456
Kurtosis:	2.147	**Cond. No.**	13.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

As We See the Model accuracy is high means our R_square of the model is 0.95 so we dont need to apply transformation method

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

Out[22]:
```
Intercept          25792.200199
YearsExperience     9449.962321
dtype: float64
```

MSE

In [23]:
```
model.mse_resid
```

Out[23]:
```
33504591.13101532
```

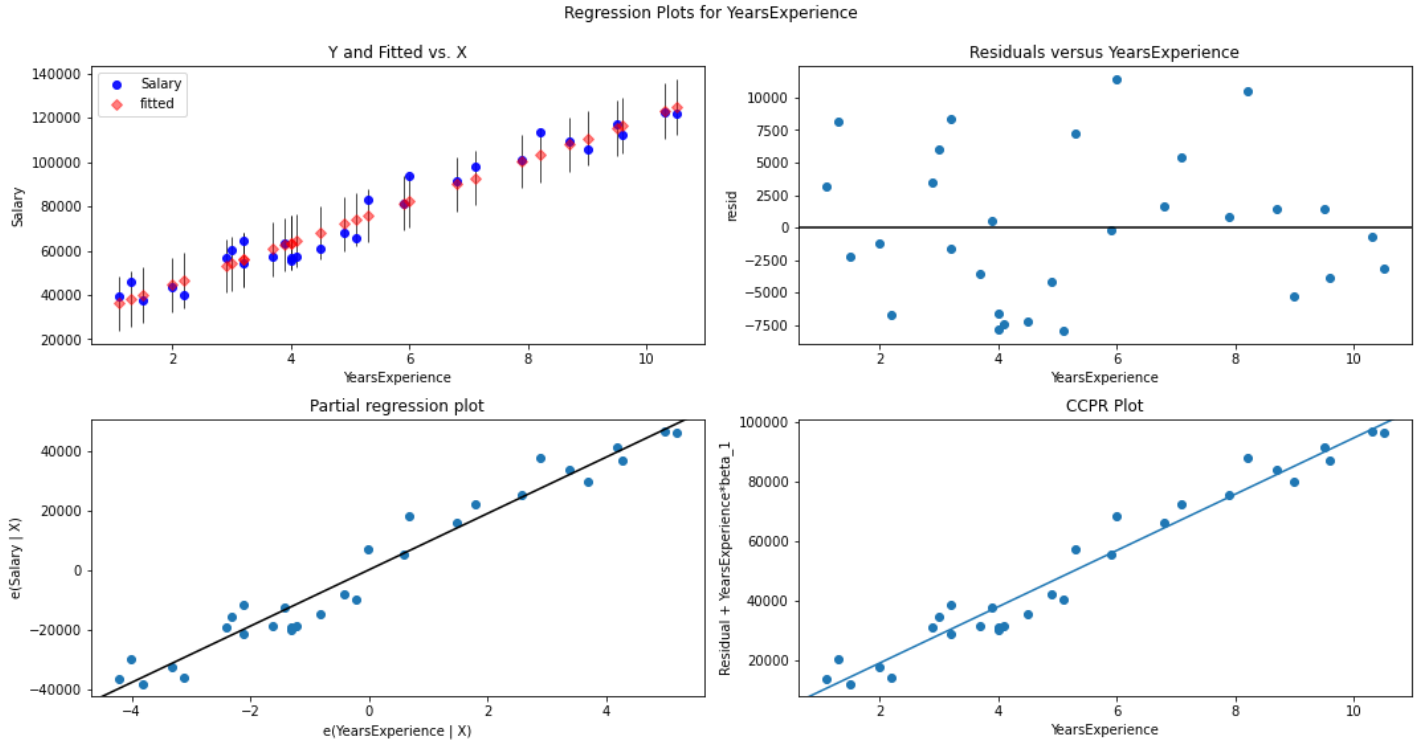
RMSE

In [24]:
```
np.sqrt(model.mse_resid)
```

Out[24]:
```
5788.315051119394
```

Residual Vs Regressor

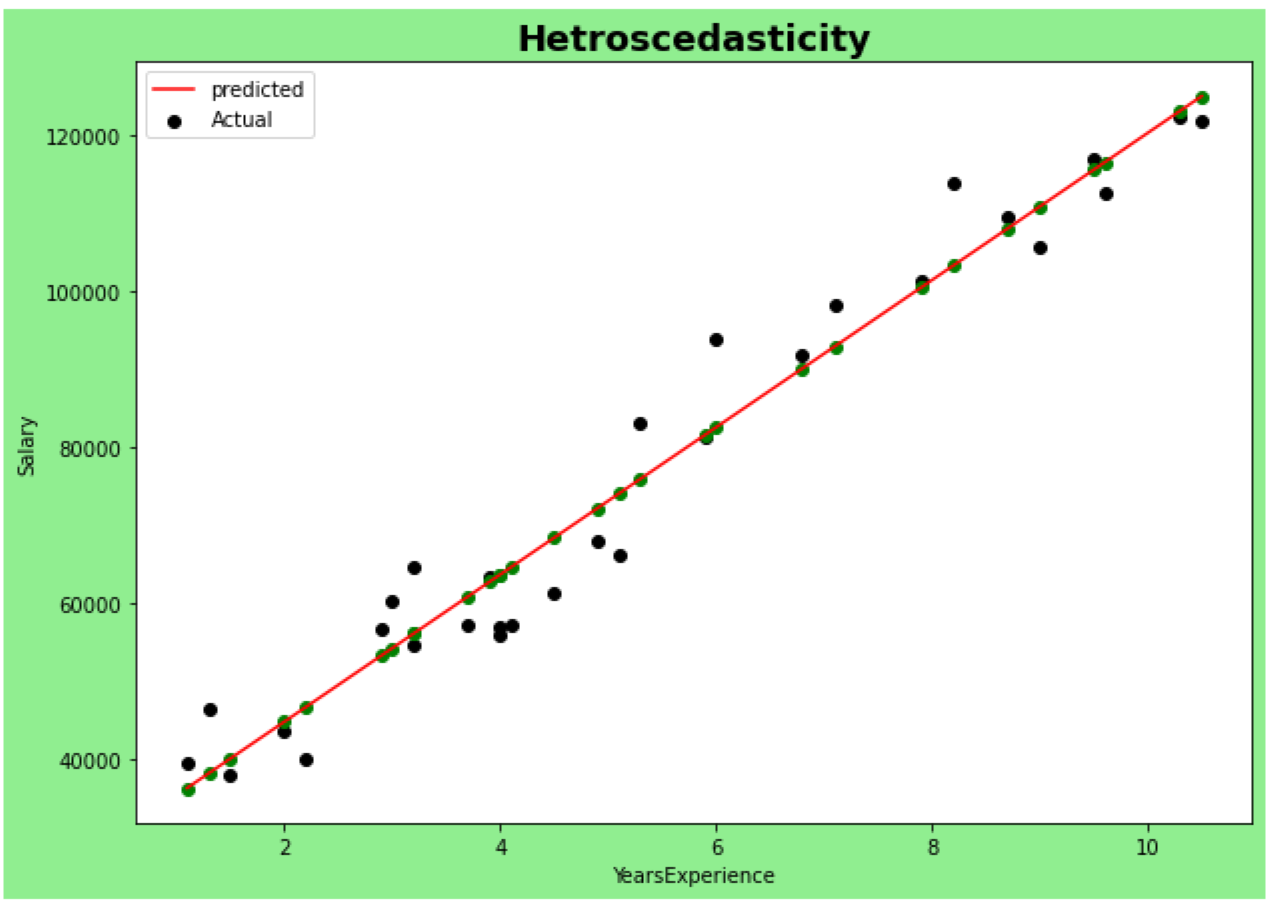
In [25]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "YearsExperience",fig=fig)
plt.show()
```



In [26]:

```
plt.figure(figsize=(10,7),facecolor='lightgreen')
plt.scatter(df.YearsExperience,df.Salary,label="Actual",color="black")
plt.plot(df.YearsExperience,model.predict(df["YearsExperience"]),color="red",linestyle='-'
plt.scatter(df.YearsExperience,model.predict(df["YearsExperience"]),color="green")
plt.xlabel("YearsExperience")
plt.ylabel("Salary")
plt.title("Hetroscedasticity",fontsize=18,fontweight='bold')
plt.legend(loc='best')
```

Out[26]: <matplotlib.legend.Legend at 0x2a8060b0a30>



Predict New Values Using Data Set

In [27]:

```
predicted2 = pd.DataFrame()
predicted2['YearsExperience'] = df.YearsExperience
predicted2['Salary'] = df.Salary
predicted2['Predicted_Salary_Hike'] = pd.DataFrame(model.predict(predicted2.YearsExperienc
predicted2
```

Out[27]:

	YearsExperience	Salary	Predicted_Salary_Hike
0	1.1	39343.0	36187.158752
1	1.3	46205.0	38077.151217
2	1.5	37731.0	39967.143681
3	2.0	43525.0	44692.124842
4	2.2	39891.0	46582.117306
5	2.9	56642.0	53197.090931
6	3.0	60150.0	54142.087163
7	3.2	54445.0	56032.079627
8	3.2	64445.0	56032.079627
9	3.7	57189.0	60757.060788
10	3.9	63218.0	62647.053252
11	4.0	55794.0	63592.049484
12	4.0	56957.0	63592.049484
13	4.1	57081.0	64537.045717
14	4.5	61111.0	68317.030645
15	4.9	67938.0	72097.015574
16	5.1	66029.0	73987.008038
17	5.3	83088.0	75877.000502
18	5.9	81363.0	81546.977895
19	6.0	93940.0	82491.974127
20	6.8	91738.0	90051.943985
21	7.1	98273.0	92886.932681
22	7.9	101302.0	100446.902538
23	8.2	113812.0	103281.891235
24	8.7	109431.0	108006.872395
25	9.0	105582.0	110841.861092
26	9.5	116969.0	115566.842252
27	9.6	112635.0	116511.838485
28	10.3	122391.0	123126.812110
29	10.5	121872.0	125016.804574

Predicting Values using Random Values

In [28]:
```
data_Predict=pd.DataFrame()
data_Predict["YearsExperience"]=pd.Series([10,11])
data_Predict
```

Out[28]:

	YearsExperience
0	10
1	11

In [29]:
```
data_Predict["Salary"]=pd.Series(model.predict(data_Predict.YearsExperience))
```

In [30]:
```
data_Predict
```

Out[30]:

	YearsExperience	Salary
0	10	120291.823413
1	11	129741.785735

In [ ]:

In [ ]:

Loading [MathJax]/extensions/Safe.js