# Assignment-04-Simple Linear Regression-1

In [5]:
```
# import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
```

In [6]:
```
# import dataset
dataset=pd.read_csv('delivery_time.csv')
dataset
```

Out[6]:

| | Delivery Time | Sorting Time |
|---|---|---|
| 0 | 21.00 | 10 |
| 1 | 13.50 | 4 |
| 2 | 19.75 | 6 |
| 3 | 24.00 | 9 |
| 4 | 29.00 | 10 |
| 5 | 15.35 | 6 |
| 6 | 19.00 | 7 |
| 7 | 9.50 | 3 |
| 8 | 17.90 | 10 |
| 9 | 18.75 | 9 |
| 10 | 19.83 | 8 |
| 11 | 10.75 | 4 |
| 12 | 16.68 | 7 |
| 13 | 11.50 | 3 |
| 14 | 12.03 | 3 |
| 15 | 14.88 | 4 |
| 16 | 13.75 | 6 |
| 17 | 18.11 | 7 |
| 18 | 8.00 | 2 |
| 19 | 17.83 | 7 |
| 20 | 21.50 | 5 |

# # EDA and Data Visualization

In [9]:
```
dataset=dataset.rename(columns={'Delivery Time': 'dt','Sorting Time': 'st' })
```

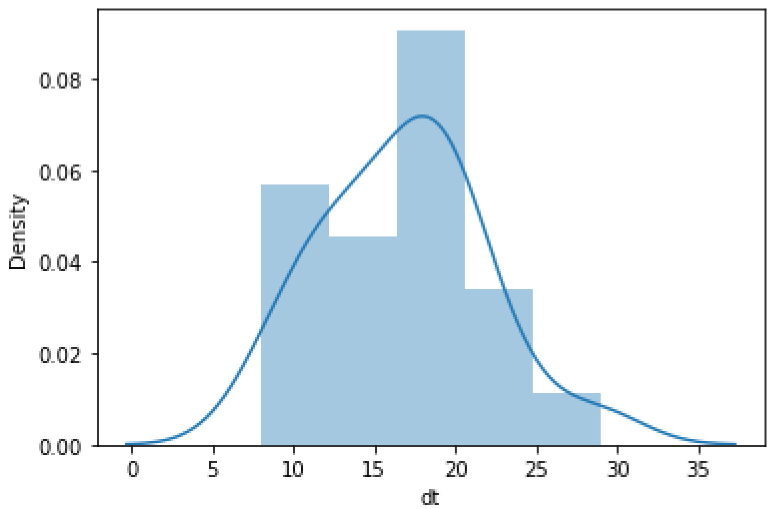
In [30]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   dt      21 non-null     float64
 1   st      21 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

In [32]:
```
sns.distplot(dataset['dt'])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

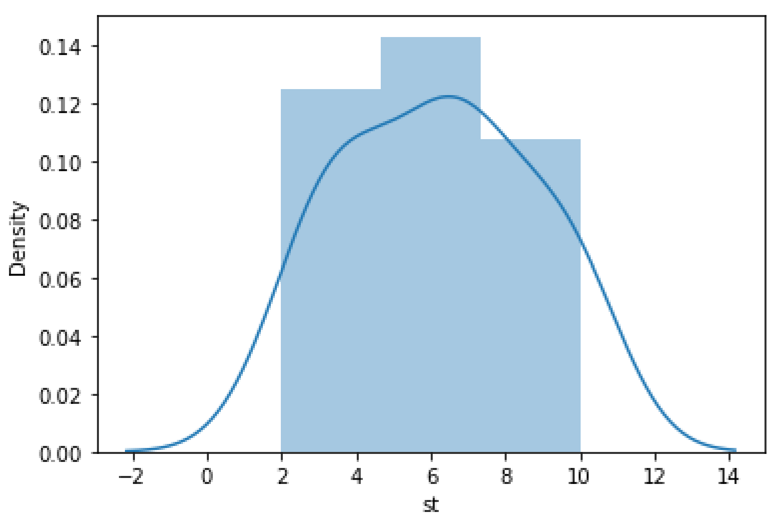
Out[32]:
```
<AxesSubplot:xlabel='dt', ylabel='Density'>
```




In [33]:
```
sns.distplot(dataset['st'])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[33]:
```
<AxesSubplot:xlabel='st', ylabel='Density'>
```




# Feature Engineering

In [34]:
```python
dataset=dataset.rename(columns={'Delivery Time': 'dt','Sorting Time': 'st' })
```

# Correlation Analysis

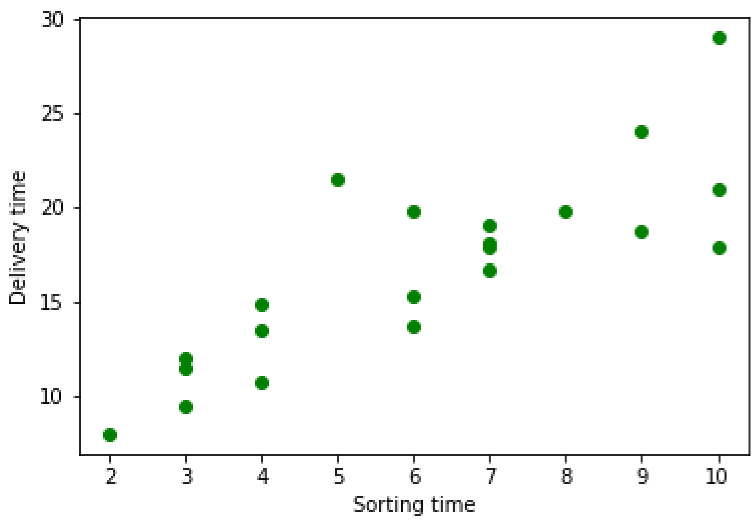
In [10]:
```python
dataset.corr()
```

Out[10]:

| | dt | st |
|---|---|---|
| **dt** | 1.000000 | 0.825997 |
| **st** | 0.825997 | 1.000000 |

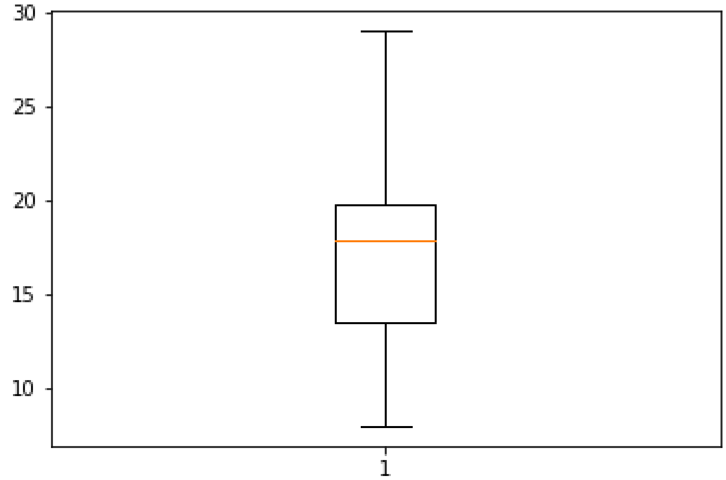
In [11]:
```python
plt.scatter(x=dataset.st, y=dataset.dt, color='green')
plt.xlabel("Sorting time")
plt.ylabel("Delivery time")
```

Out[11]:
```
Text(0, 0.5, 'Delivery time')
```



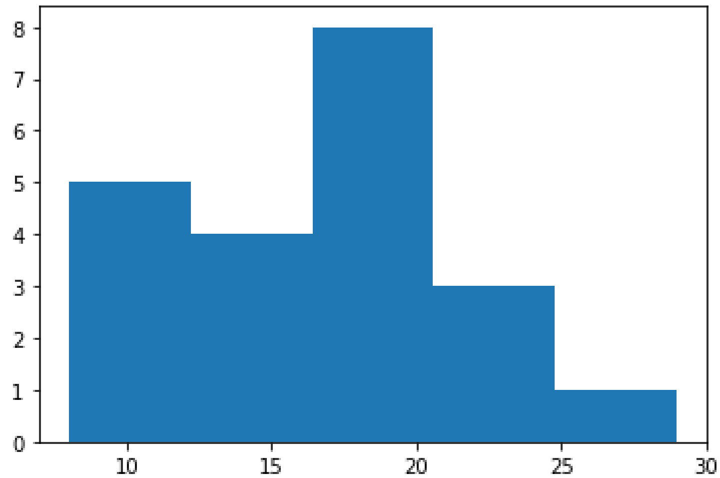

In [12]:
```python
plt.boxplot(dataset.dt)
```

Out[12]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x19226575640>,
  <matplotlib.lines.Line2D at 0x19226575940>],
 'caps': [<matplotlib.lines.Line2D at 0x19226575cd0>,
  <matplotlib.lines.Line2D at 0x19226575ee0>],
 'boxes': [<matplotlib.lines.Line2D at 0x19226575370>],
 'medians': [<matplotlib.lines.Line2D at 0x192265881f0>],
 'fliers': [<matplotlib.lines.Line2D at 0x192265884c0>],
 'means': []}
```




In [13]:
```
plt.hist(dataset.dt, bins=5)
```

Out[13]:
```
(array([5., 4., 8., 3., 1.]),
 array([ 8. , 12.2, 16.4, 20.6, 24.8, 29. ]),
 <BarContainer object of 5 artists>)
```




# Model Building

In [14]:
```
model2=smf.ols("dt~st",data=dataset).fit()
```

# Model Testing

In [35]:
```
# Finding Coefficient parameters
model2.params
```

Out[35]:
```
Intercept    6.582734
st           1.649020
dtype: float64
```

In [16]:
```
model2.summary()
```

Out[16]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | dt | **R-squared:** | 0.682 |
| **Model:** | OLS | **Adj. R-squared:** | 0.666 |
| **Method:** | Least Squares | **F-statistic:** | 40.80 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 3.98e-06 |
| **Time:** | 13:48:20 | **Log-Likelihood:** | -51.357 |
| **No. Observations:** | 21 | **AIC:** | 106.7 |
| **Df Residuals:** | 19 | **BIC:** | 108.8 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 6.5827 | 1.722 | 3.823 | 0.001 | 2.979 | 10.186 |
| **st** | 1.6490 | 0.258 | 6.387 | 0.000 | 1.109 | 2.189 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 3.649 | **Durbin-Watson:** | 1.248 |
| **Prob(Omnibus):** | 0.161 | **Jarque-Bera (JB):** | 2.086 |
| **Skew:** | 0.750 | **Prob(JB):** | 0.352 |
| **Kurtosis:** | 3.367 | **Cond. No.** | 18.3 |

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

In [18]:
```
model3=smf.ols("dt~np.log(st)",data=dataset).fit()
```

In [19]:
```
model3.params
```

Out[19]:
```
Intercept     1.159684
np.log(st)    9.043413
dtype: float64
```

In [20]:
```
model3.summary()
```

Out[20]: OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | dt | **R-squared:** | 0.695 |
| **Model:** | OLS | **Adj. R-squared:** | 0.679 |
| **Method:** | Least Squares | **F-statistic:** | 43.39 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 2.64e-06 |
| **Time:** | 13:51:47 | **Log-Likelihood:** | -50.912 |
| **No. Observations:** | 21 | **AIC:** | 105.8 |
| **Df Residuals:** | 19 | **BIC:** | 107.9 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 1.1597 | 2.455 | 0.472 | 0.642 | -3.978 | 6.297 |
| **np.log(st)** | 9.0434 | 1.373 | 6.587 | 0.000 | 6.170 | 11.917 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 5.552 | **Durbin-Watson:** | 1.427 |
| **Prob(Omnibus):** | 0.062 | **Jarque-Bera (JB):** | 3.481 |
| **Skew:** | 0.946 | **Prob(JB):** | 0.175 |
| **Kurtosis:** | 3.628 | **Cond. No.** | 9.08 |

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

In [21]:
```
model2.conf_int(0.05) # 95% confidence interval
```

Out[21]:

| | 0 | 1 |
|---|---|---|
| **Intercept** | 2.979134 | 10.186334 |
| **st** | 1.108673 | 2.189367 |

In [22]:
```
model3.conf_int(0.05) # 95% confidence interval
```

Out[22]:

| | 0 | 1 |
|---|---|---|
| **Intercept** | -3.97778 | 6.297147 |
| **np.log(st)** | 6.16977 | 11.917057 |

# Model Predictions

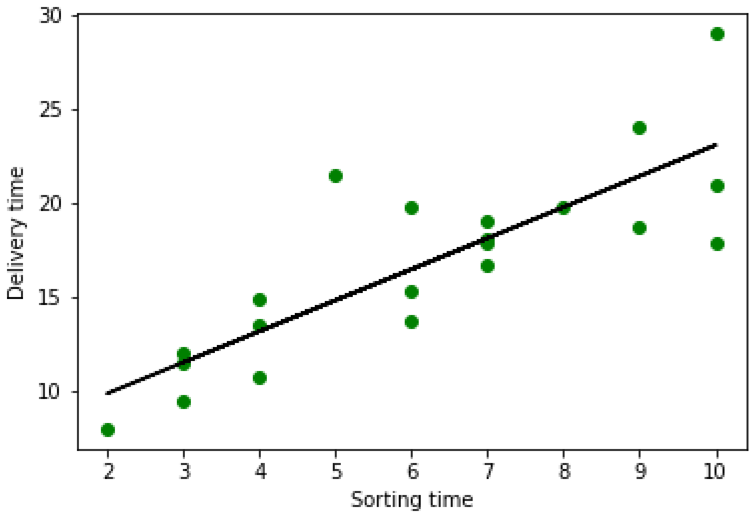
In [24]:
```
pred2 = model2.predict(dataset) # Predicted values of dt using the model
```

In [25]:
```
pred3 = model3.predict(dataset) # Predicted values of dt using the model
```

In [26]:
```
plt.scatter(x=dataset.st, y=dataset.dt, color='green')
plt.plot(dataset.st, pred2,color='black')
```

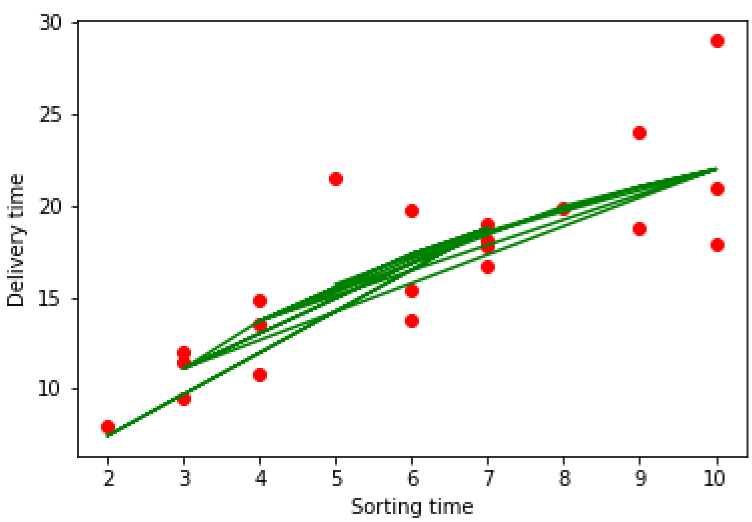
```
plt.xlabel("Sorting time")
plt.ylabel("Delivery time")
```

Out[26]:
```
Text(0, 0.5, 'Delivery time')
```




In [29]:
```
plt.scatter(x=dataset.st, y=dataset.dt, color='red')
plt.plot(dataset.st, pred3,color='green')
plt.xlabel("Sorting time")
plt.ylabel("Delivery time")
```

Out[29]:
```
Text(0, 0.5, 'Delivery time')
```




# Model Testing

In [70]:
```
# Finding Coefficient parameters
model.params
```

Out[70]:
```
Intercept          25792.200199
YearsExperience     9449.962321
dtype: float64
```

In [71]:
```
# Finding tvalues and pvalues
model.tvalues , model.pvalues
```

Out[71]:
```
(Intercept           11.346940
 YearsExperience     24.950094
 dtype: float64,
 Intercept           5.511950e-12
 YearsExperience     1.143068e-20
 dtype: float64)
```

In [72]:
```
# Finding Rsquared Values
model.rsquared , model.rsquared_adj
```

Out[72]:
```
(0.9569566641435086, 0.9554194021486339)
```

# Assignment-04-Simple Linear Regression-2

In [36]:
```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as smf
import statsmodels.formula.api as sm
import warnings
warnings.filterwarnings('ignore')
```

In [37]:
```
df = pd.read_csv('Salary_Data.csv')
df
```

Out[37]:

| | 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 |
| **5** | 2.9 | 56642.0 |
| **6** | 3.0 | 60150.0 |
| **7** | 3.2 | 54445.0 |
| **8** | 3.2 | 64445.0 |
| **9** | 3.7 | 57189.0 |
| **10** | 3.9 | 63218.0 |
| **11** | 4.0 | 55794.0 |
| **12** | 4.0 | 56957.0 |
| **13** | 4.1 | 57081.0 |
| **14** | 4.5 | 61111.0 |
| **15** | 4.9 | 67938.0 |
| **16** | 5.1 | 66029.0 |
| **17** | 5.3 | 83088.0 |
| **18** | 5.9 | 81363.0 |
| **19** | 6.0 | 93940.0 |
| **20** | 6.8 | 91738.0 |
| **21** | 7.1 | 98273.0 |
| **22** | 7.9 | 101302.0 |
| **23** | 8.2 | 113812.0 |
| **24** | 8.7 | 109431.0 |
| **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 [38]:

```
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 [39]:

```
df.describe()
```

Out[39]:

| | YearsExperience | Salary |
|---|---|---|
| **count** | 30.000000 | 30.000000 |
| **mean** | 5.313333 | 76003.000000 |
| **std** | 2.837888 | 27414.429785 |
| **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 |

# Checking for Null Values

In [40]:

```
df.isnull().sum()
```

Out[40]:

```
YearsExperience    0
Salary             0
dtype: int64
```

# Checking for Duplicate Values

In [41]:

```
df[df.duplicated()].shape
```

Out[41]:

```
(0, 2)
```

In [42]:

```
df[df.duplicated()]
```

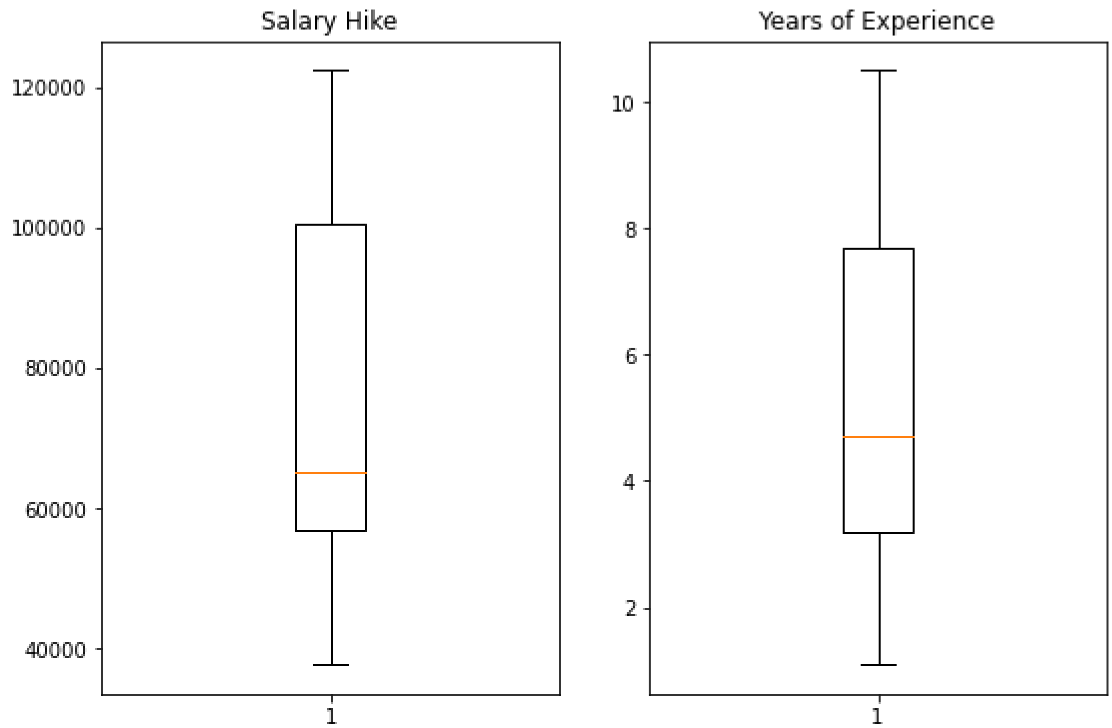
Out[42]:

| YearsExperience | Salary |
|---|---|

# Plotting the data to check for outliers

In [43]:

```
plt.subplots(figsize = (9,6))
plt.subplot(121)
plt.boxplot(df['Salary'])
plt.title('Salary Hike')
plt.subplot(122)
plt.boxplot(df['YearsExperience'])
plt.title('Years of Experience')
plt.show()
```




# Checking the Correlation between variables

In [44]:
```
df.corr()
```

Out[44]:

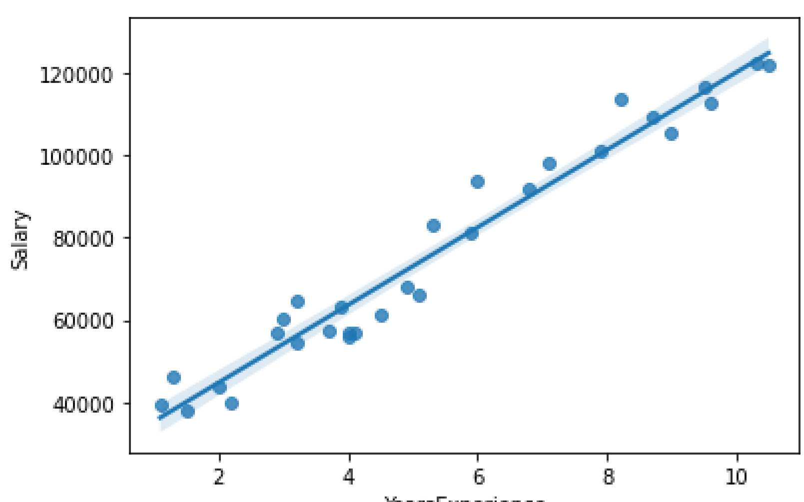
| | YearsExperience | Salary |
|---|---|---|
| **YearsExperience** | 1.000000 | 0.978242 |
| **Salary** | 0.978242 | 1.000000 |

# Visualization of Correlation beteen x and y

## regplot = regression plot

In [45]:
```
sns.regplot(x=df['YearsExperience'],y=df['Salary'])
```

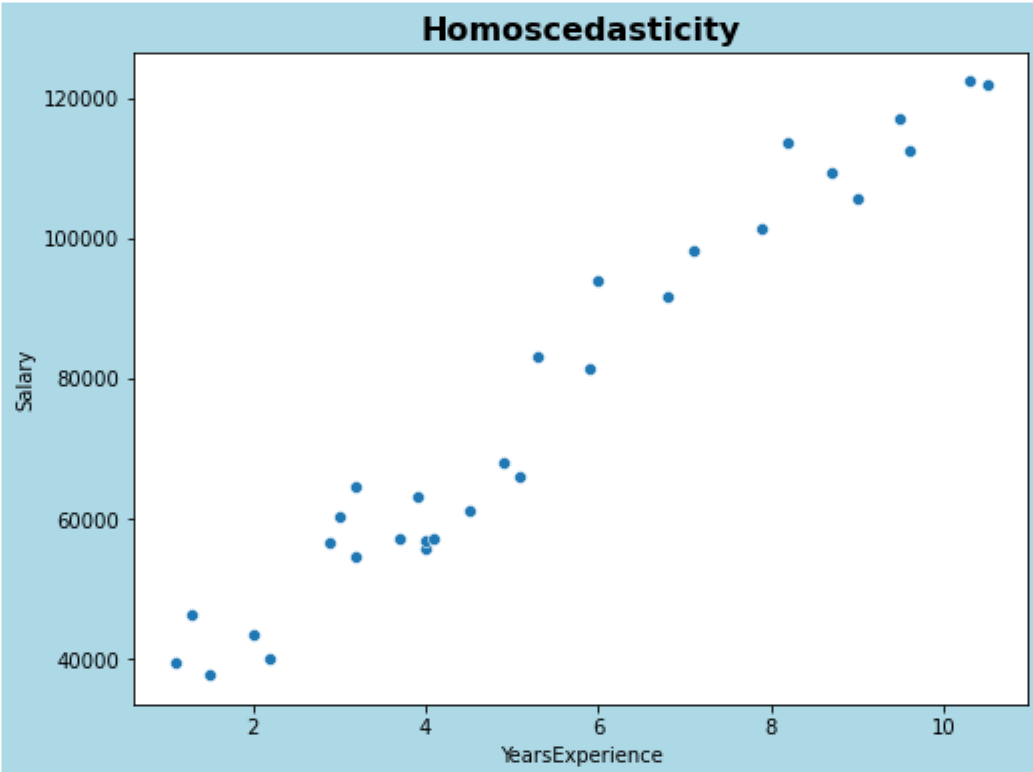
Out[45]:
```
<AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>
```




Loading [MathJax]/extensions/Safe.js

# Checking for Homoscedasticity or Hetroscedasticity

In [46]:
```
plt.figure(figsize = (8,6), facecolor = 'lightblue')
sns.scatterplot(x = df['YearsExperience'], y = df['Salary'])
plt.title('Homoscedasticity', fontweight = 'bold', fontsize = 16)
plt.show()
```

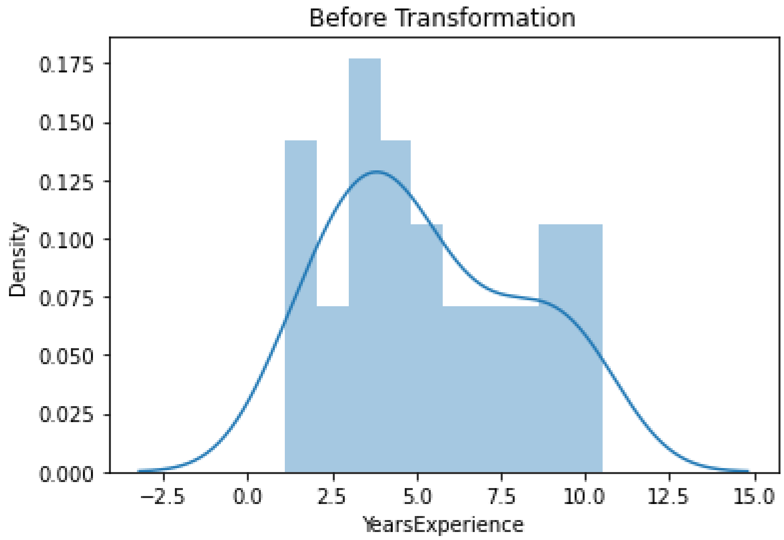


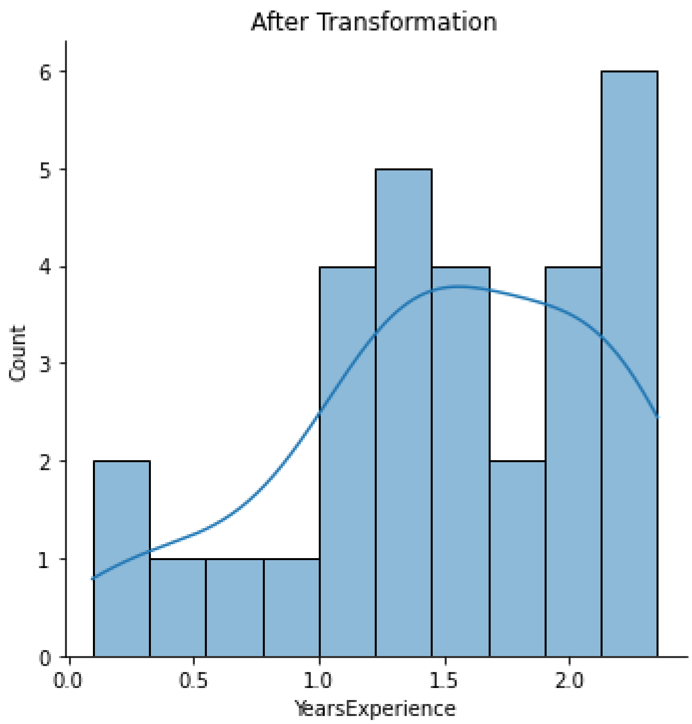

In [47]:
```
df.var()
```

Out[47]:
```
YearsExperience    8.053609e+00
Salary             7.515510e+08
dtype: float64
```

# Feature Engineering

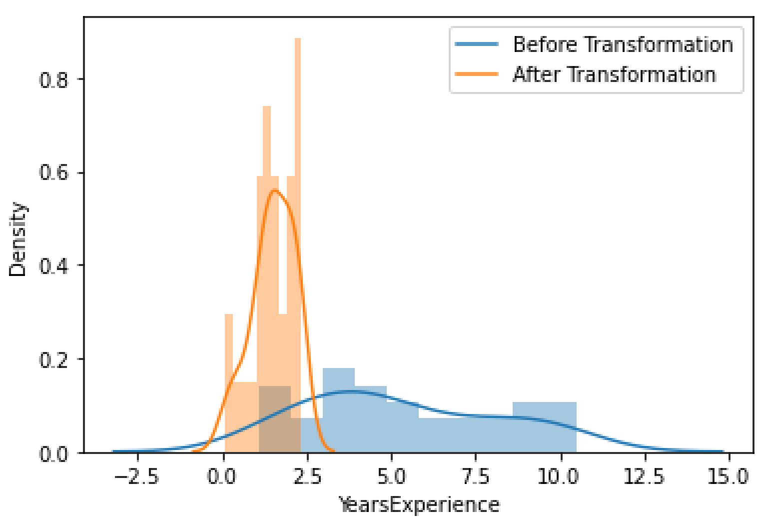
In [48]:
```
sns.distplot(df['YearsExperience'], bins = 10, kde = True)
plt.title('Before Transformation')
sns.displot(np.log(df['YearsExperience']), bins = 10, kde = True)
plt.title('After Transformation')
plt.show()
```



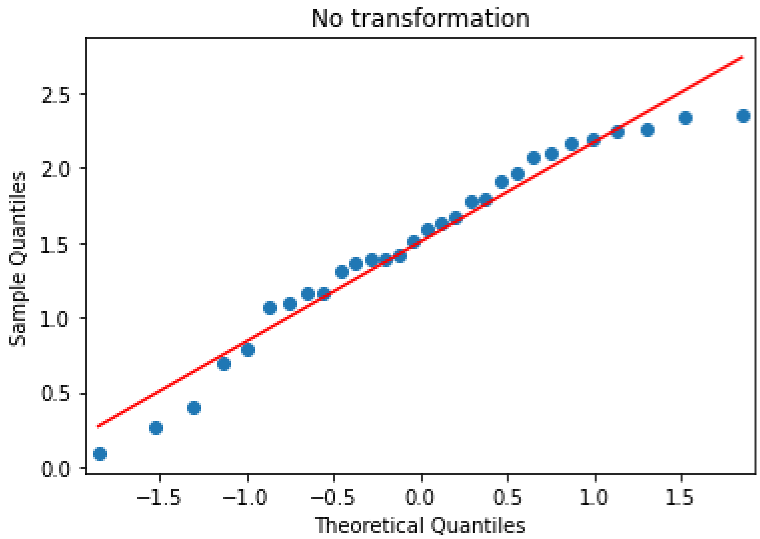



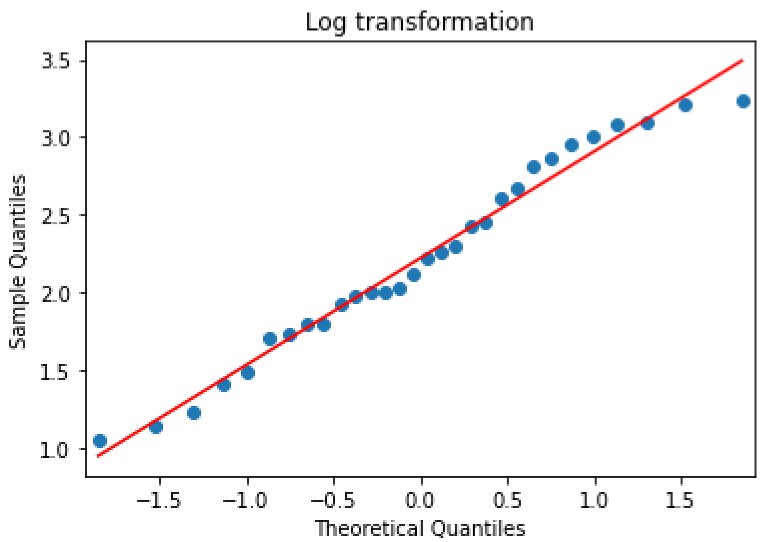

In [49]:

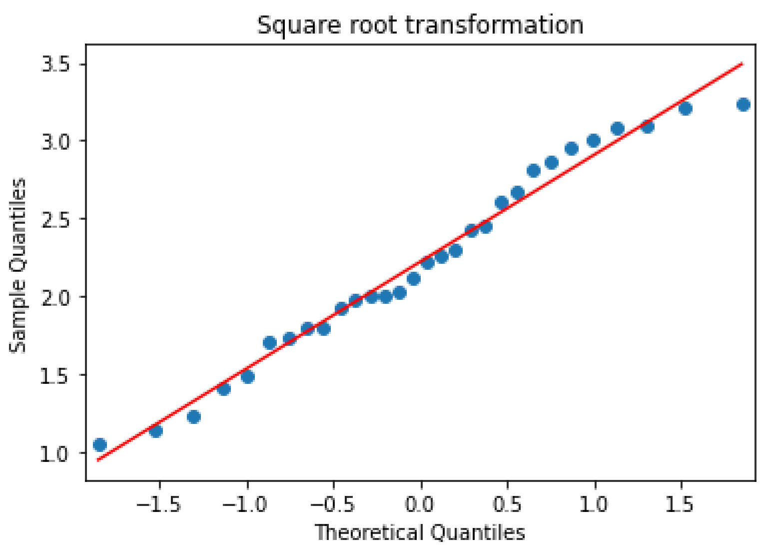
```
labels = ['Before Transformation','After Transformation']
sns.distplot(df['YearsExperience'], bins = 10, kde = True)
sns.distplot(np.log(df['YearsExperience']), bins = 10, kde = True)
plt.legend(labels)
plt.show()
```



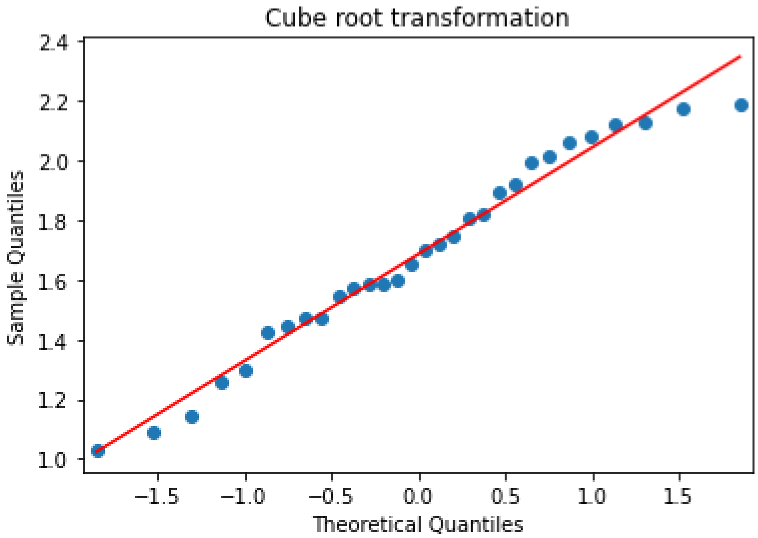
In [50]:
```
smf.qqplot(np.log(df['YearsExperience']), line = 'r')
plt.title('No transformation')
smf.qqplot(np.sqrt(df['YearsExperience']), line = 'r')
plt.title('Log transformation')
smf.qqplot(np.sqrt(df['YearsExperience']), line = 'r')
plt.title('Square root transformation')
smf.qqplot(np.cbrt(df['YearsExperience']), line = 'r')
plt.title('Cube root transformation')
plt.show()
```



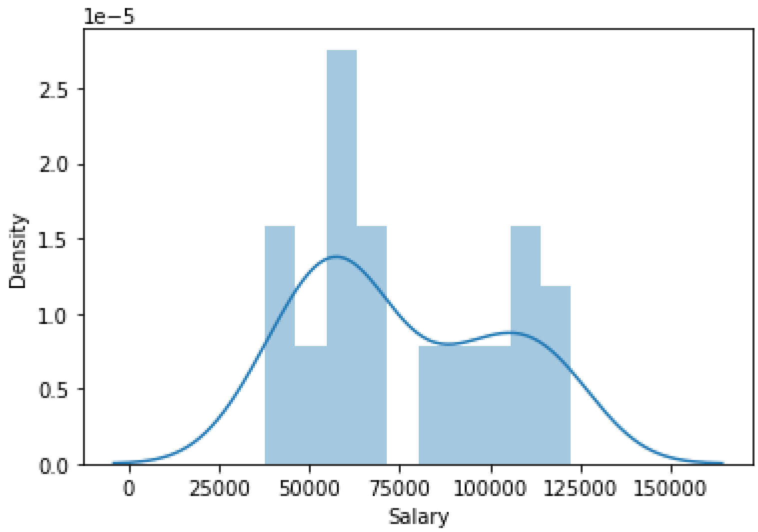



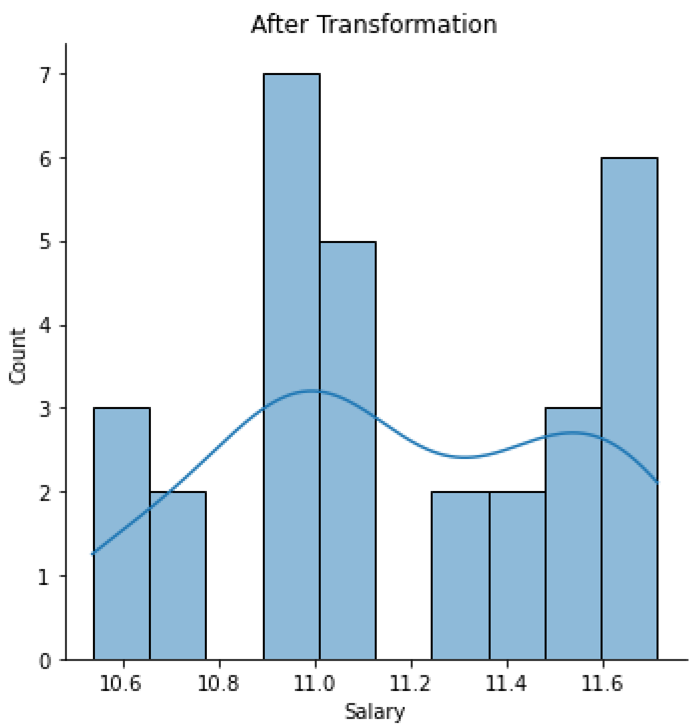

Loading [MathJax]/extensions/Safe.js



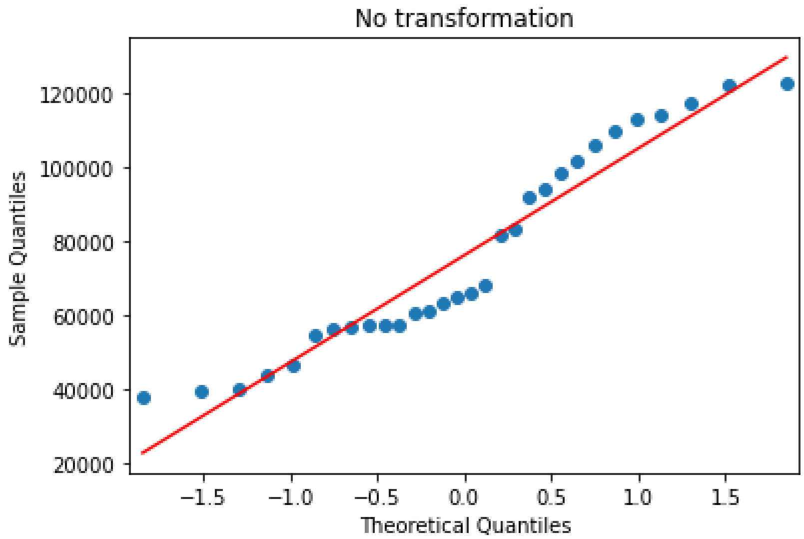

In [51]:

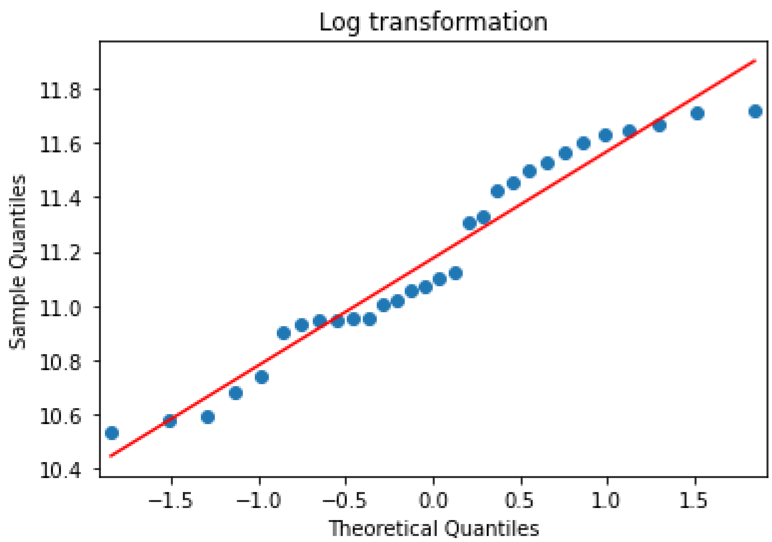
```
labels = ['Before Transformation','After Transformation']
sns.distplot(df['Salary'], bins = 10, kde = True)
sns.displot(np.log(df['Salary']), bins = 10, kde = True)
plt.title('After Transformation')
plt.show()
```

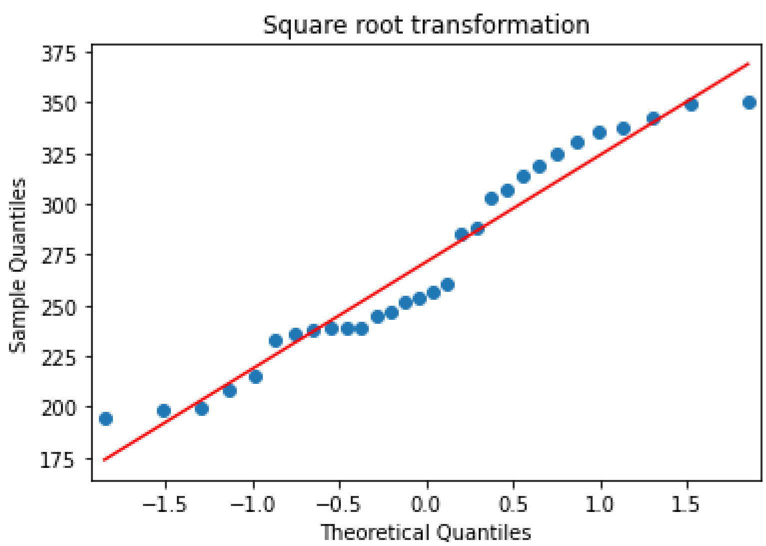


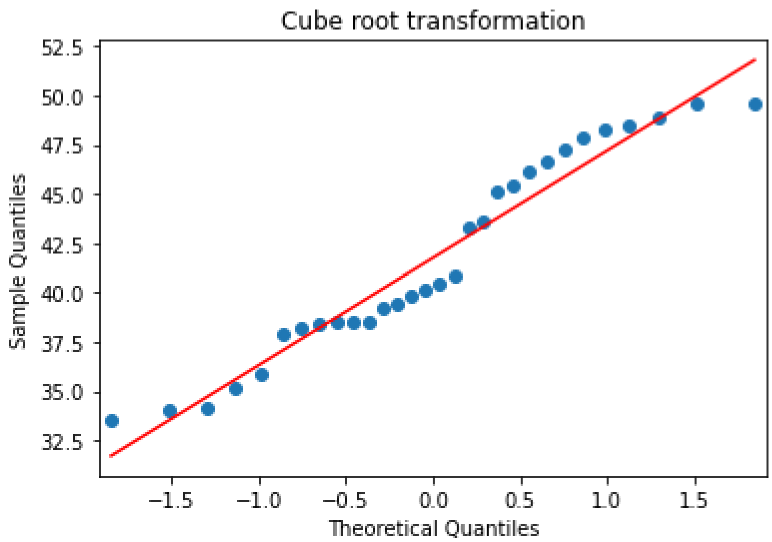
In [52]:
```
smf.qqplot(df['Salary'], line = 'r')
plt.title('No transformation')
smf.qqplot(np.log(df['Salary']), line = 'r')
plt.title('Log transformation')
smf.qqplot(np.sqrt(df['Salary']), line = 'r')
plt.title('Square root transformation')
smf.qqplot(np.cbrt(df['Salary']), line = 'r')
plt.title('Cube root transformation')
plt.show()
```




Loading [MathJax]/extensions/Safe.js




# Fitting a Linear Regression Model

In [53]:
```
import statsmodels.formula.api as sm
model = sm.ols('Salary~YearsExperience', data = df).fit()
```

In [54]:
```
model.summary()
```

Out[54]:

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:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 1.14e-20 |
| **Time:** | 15:37:31 | **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.

In [55]:
```
model1 = sm.ols('np.sqrt(Salary)~np.sqrt(YearsExperience)', data = df).fit()
```

```
model1.summary()
```

Out[55]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | np.sqrt(Salary) | **R-squared:** | 0.942 |
| **Model:** | OLS | **Adj. R-squared:** | 0.940 |
| **Method:** | Least Squares | **F-statistic:** | 454.3 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 7.58e-19 |
| **Time:** | 15:38:01 | **Log-Likelihood:** | -116.52 |
| **No. Observations:** | 30 | **AIC:** | 237.0 |
| **Df Residuals:** | 28 | **BIC:** | 239.8 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 103.5680 | 8.178 | 12.663 | 0.000 | 86.815 | 120.321 |
| **np.sqrt(YearsExperience)** | 75.6269 | 3.548 | 21.315 | 0.000 | 68.359 | 82.895 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 0.924 | **Durbin-Watson:** | 1.362 |
| **Prob(Omnibus):** | 0.630 | **Jarque-Bera (JB):** | 0.801 |
| **Skew:** | 0.087 | **Prob(JB):** | 0.670 |
| **Kurtosis:** | 2.219 | **Cond. No.** | 9.97 |

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

In [56]:

```
model2 = sm.ols('np.cbrt(Salary)~np.cbrt(YearsExperience)', data = df).fit()
model2.summary()
```

Out[56]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | np.cbrt(Salary) | **R-squared:** | 0.932 |
| **Model:** | OLS | **Adj. R-squared:** | 0.930 |
| **Method:** | Least Squares | **F-statistic:** | 386.5 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 6.37e-18 |
| **Time:** | 15:38:23 | **Log-Likelihood:** | -50.589 |
| **No. Observations:** | 30 | **AIC:** | 105.2 |
| **Df Residuals:** | 28 | **BIC:** | 108.0 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 16.6603 | 1.300 | 12.811 | 0.000 | 13.996 | 19.324 |
| **np.cbrt(YearsExperience)** | 14.8963 | 0.758 | 19.659 | 0.000 | 13.344 | 16.448 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 0.386 | **Durbin-Watson:** | 1.229 |
| **Prob(Omnibus):** | 0.824 | **Jarque-Bera (JB):** | 0.535 |
| **Skew:** | 0.070 | **Prob(JB):** | 0.765 |
| **Kurtosis:** | 2.361 | **Cond. No.** | 12.0 |

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

In [57]:

```
model3 = sm.ols('np.log(Salary)~np.log(YearsExperience)', data = df).fit()
model3.summary()
```

Out[57]: OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | np.log(Salary) | **R-squared:** | 0.905 |
| **Model:** | OLS | **Adj. R-squared:** | 0.902 |
| **Method:** | Least Squares | **F-statistic:** | 267.4 |
| **Date:** | Tue, 29 Nov 2022 | **Prob (F-statistic):** | 7.40e-16 |
| **Time:** | 15:38:44 | **Log-Likelihood:** | 23.209 |
| **No. Observations:** | 30 | **AIC:** | -42.42 |
| **Df Residuals:** | 28 | **BIC:** | -39.61 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 10.3280 | 0.056 | 184.868 | 0.000 | 10.214 | 10.442 |
| **np.log(YearsExperience)** | 0.5621 | 0.034 | 16.353 | 0.000 | 0.492 | 0.632 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 0.102 | **Durbin-Watson:** | 0.988 |
| **Prob(Omnibus):** | 0.950 | **Jarque-Bera (JB):** | 0.297 |
| **Skew:** | 0.093 | **Prob(JB):** | 0.862 |
| **Kurtosis:** | 2.549 | **Cond. No.** | 5.76 |

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

# Model Testing

In [58]:
```
model.params
```

Out[58]:
```
Intercept          25792.200199
YearsExperience     9449.962321
dtype: float64
```

In [59]:
```
print(model.tvalues,'\n',model.pvalues)
```

```
Intercept          11.346940
YearsExperience    24.950094
dtype: float64
 Intercept          5.511950e-12
YearsExperience    1.143068e-20
dtype: float64
```

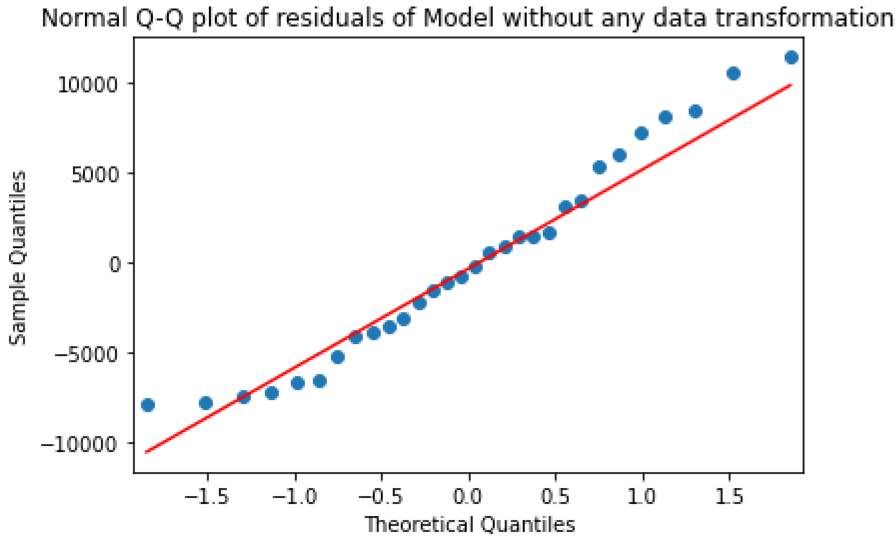
In [60]:
```
model.rsquared,model.rsquared_adj
```

Out[60]:
```
(0.9569566641435086, 0.9554194021486339)
```

# Residual Analysis

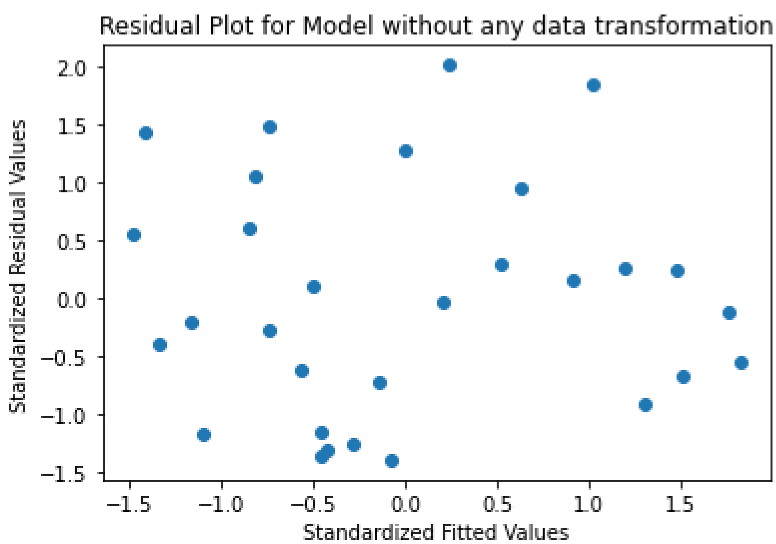
In [61]:
```
import statsmodels.api as sm
```
Loading [MathJax]/extensions/Safe.js
```
del.resid, line = 'q')
```

```
plt.title('Normal Q-Q plot of residuals of Model without any data transformation')
plt.show()
```




In [62]:
```
def get_standardized_values( vals ):
    return (vals - vals.mean())/vals.std()
```

In [63]:
```
plt.scatter(get_standardized_values(model.fittedvalues), get_standardized_values(model.r
plt.title('Residual Plot for Model without any data transformation')
plt.xlabel('Standardized Fitted Values')
plt.ylabel('Standardized Residual Values')
plt.show()
```




# Model Validation

In [64]:
```
from sklearn.metrics import mean_squared_error
```

In [65]:
```
model1_pred_y =np.square(model1.predict(df['YearsExperience']))
model2_pred_y =pow(model2.predict(df['YearsExperience']),3)
model3_pred_y =np.exp(model3.predict(df['YearsExperience']))
```

In [66]:
```
model1_rmse =np.sqrt(mean_squared_error(df['Salary'], model1_pred_y))
model2_rmse =np.sqrt(mean_squared_error(df['Salary'], model2_pred_y))
model3_rmse =np.sqrt(mean_squared_error(df['Salary'], model3_pred_y))
print('model=', np.sqrt(model.mse_resid),'\n' 'model1=', model1_rmse,'\n' 'model2=', mod
```

```
model= 5788.315051119395
model1= 5960.64709617431
model2= 6232.815455835842
model3= 7219.716974372806
```

In [67]:
```python
rmse = {'model': np.sqrt(model.mse_resid), 'model1': model1_rmse, 'model2': model3_rmse,
min(rmse, key=rmse.get)
```

Out[67]:
```
'model'
```

# Predicting values

In [69]:
```python
# first model results without any transformation
predicted2 = pd.DataFrame()
predicted2['YearsExperience'] = df.YearsExperience
predicted2['Salary'] = df.Salary
predicted2['Predicted_Salary_Hike'] = pd.DataFrame(model.predict(predicted2.YearsExperie
predicted2
```

Out[69]:

| | 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 |

In [ ]: