In [1]:
```
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 [2]:
```
df = pd.read_csv("delivery_time.csv")
```

In [3]:
```
df.head(10)
```

Out[3]:

	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

In [4]:
```
df.tail()
```

Out[4]:

	Delivery Time	Sorting Time
16	13.75	6
17	18.11	7
18	8.00	2
19	17.83	7
20	21.50	5

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Delivery Time  21 non-null     float64
 1   Sorting Time   21 non-null     int64  
```

```
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

In [6]:
```
df.shape
```

Out[6]:
```
(21, 2)
```

EDA And Feature Engineering

In [7]:
```
df1 = df.rename({"Delivery Time" : "Delivery_Time","Sorting Time" : "Sorting_Time"},axis =
```

In [8]:
```
df1
```

Out[8]:

	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

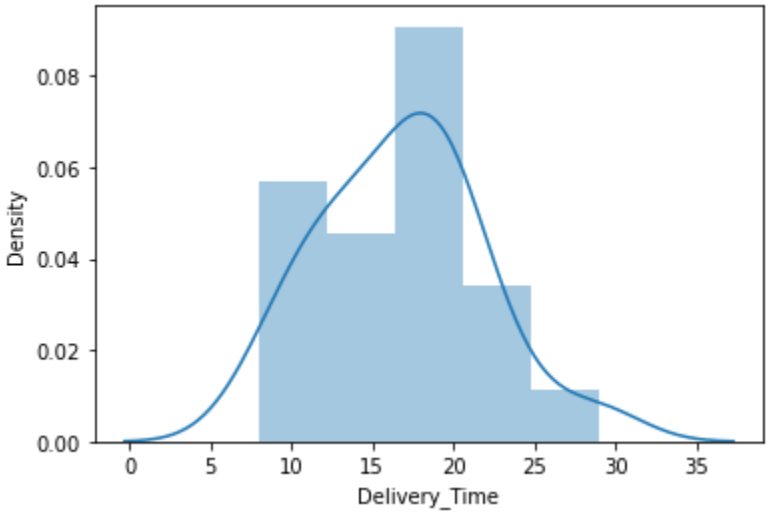
In [9]:
```
df1.corr()
```

Out[9]:

	Delivery_Time	Sorting_Time
Delivery_Time	1.000000	0.825997
Sorting_Time	0.825997	1.000000

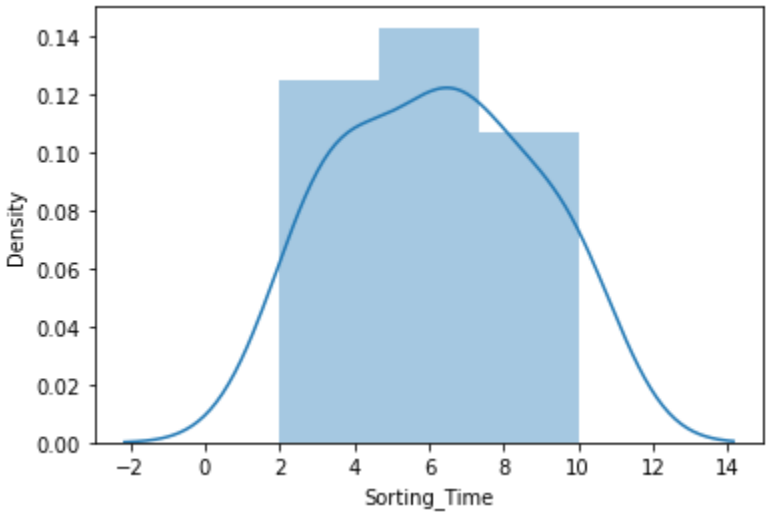
In [10]:
```
sns.distplot(df1.Delivery_Time,kde=True)
```

Out[10]:
```
<AxesSubplot:xlabel='Delivery_Time', ylabel='Density'>
```



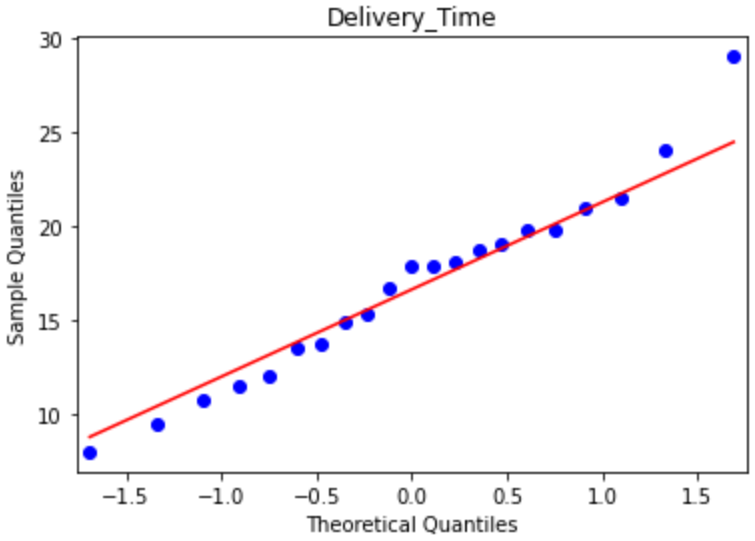
In [11]:
```
sns.distplot(df1.Sorting_Time,kde=True)
```

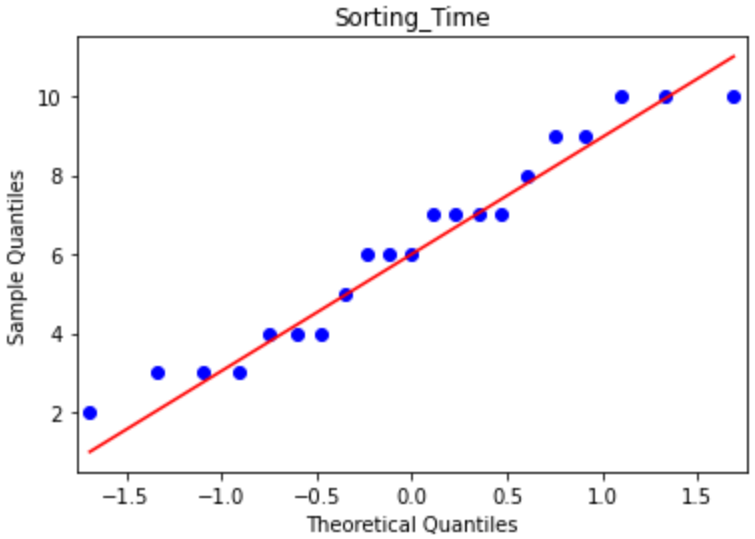
Out[11]:
```
<AxesSubplot:xlabel='Sorting_Time', ylabel='Density'>
```



QQ-Plot For Raw Data

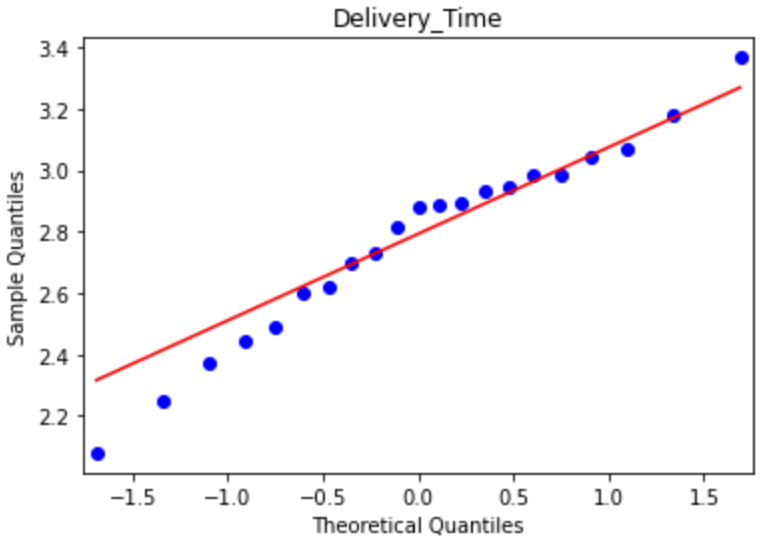
In [12]:
```
for feature in df1:
    data = df1.copy()
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```

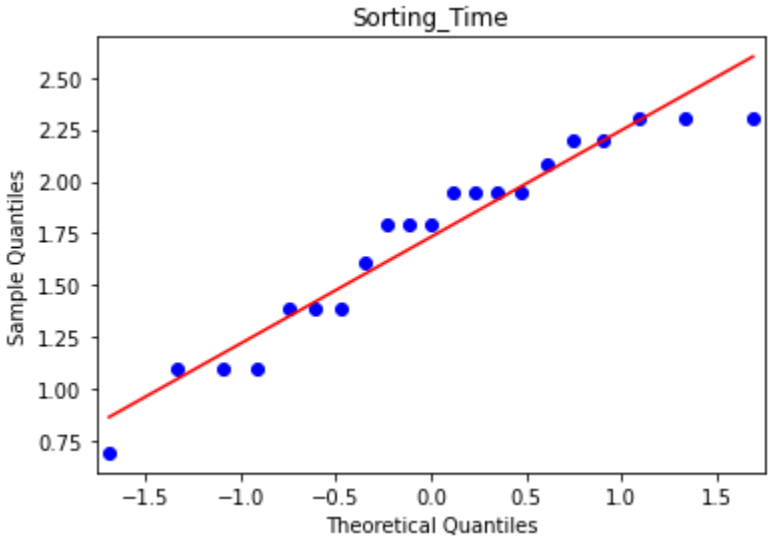




QQ-Plot For Log Transformation

In [13]:
```
for feature in df1:
    data = df1.copy()
    data[feature]=np.log(data[feature])
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```

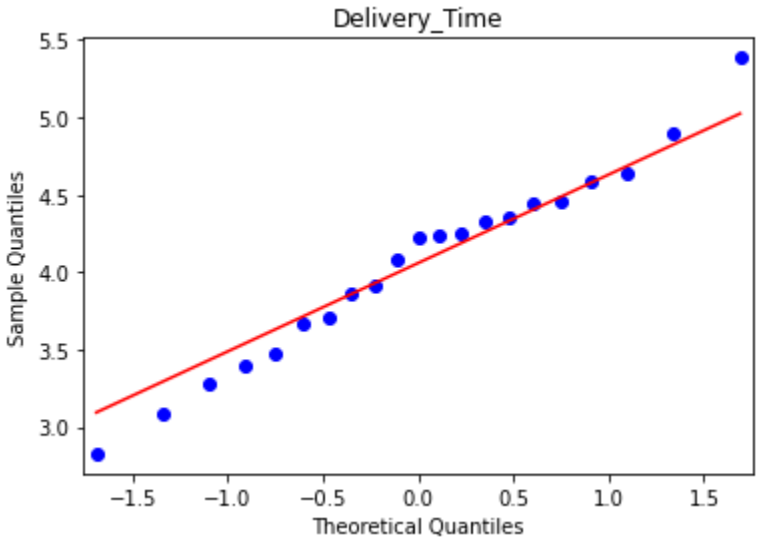


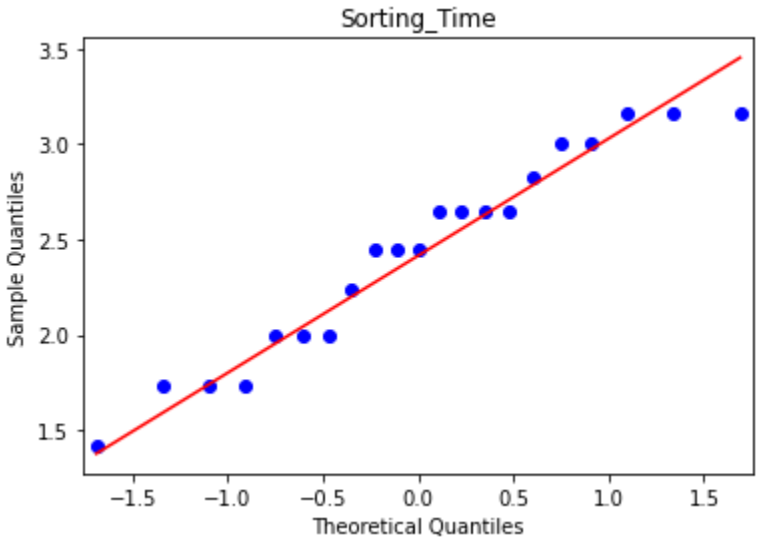


QQ-Plot for Squareroot Transformation

In [14]:

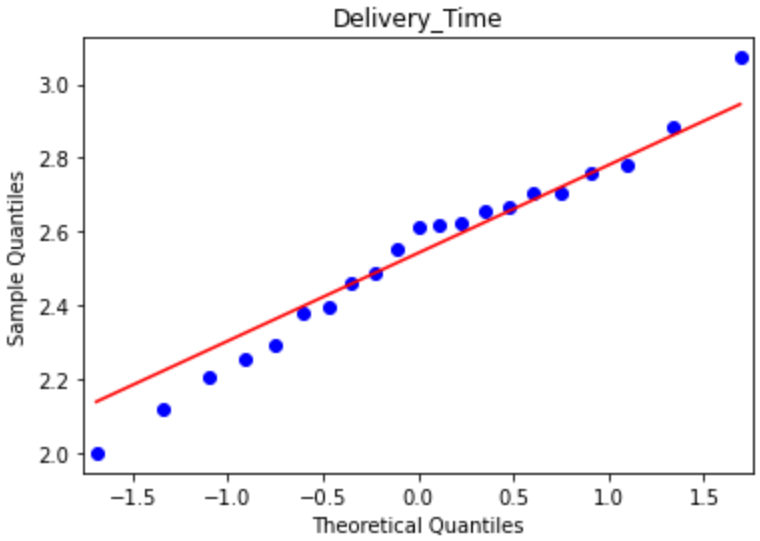
```
for feature in df1:
    data = df1.copy()
    data[feature]=np.sqrt(data[feature])
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```

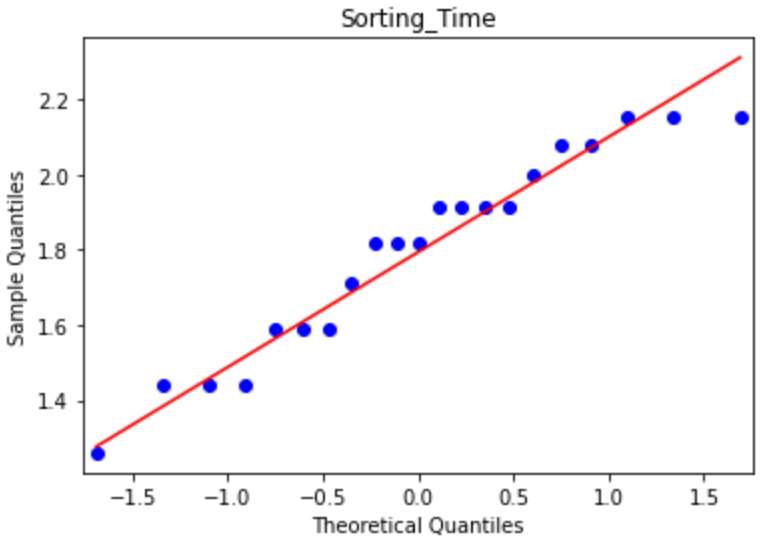




QQ-Plot For Cuberoot Transformation

In [15]:
```
for feature in df1:
    data = df1.copy()
    data[feature]=np.cbrt(data[feature])
    sm.qqplot(data[feature],line="q")
    plt.title(feature)
```





Observation

There is not much difference after transforming data so we use raw data

Checking Colinearity

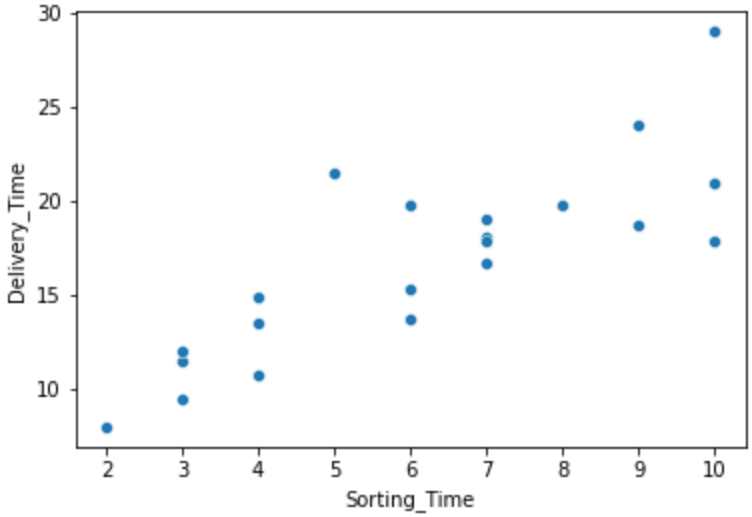
In [16]:
```
df1.corr()
```

Out[16]:

	Delivery_Time	Sorting_Time
Delivery_Time	1.000000	0.825997
Sorting_Time	0.825997	1.000000

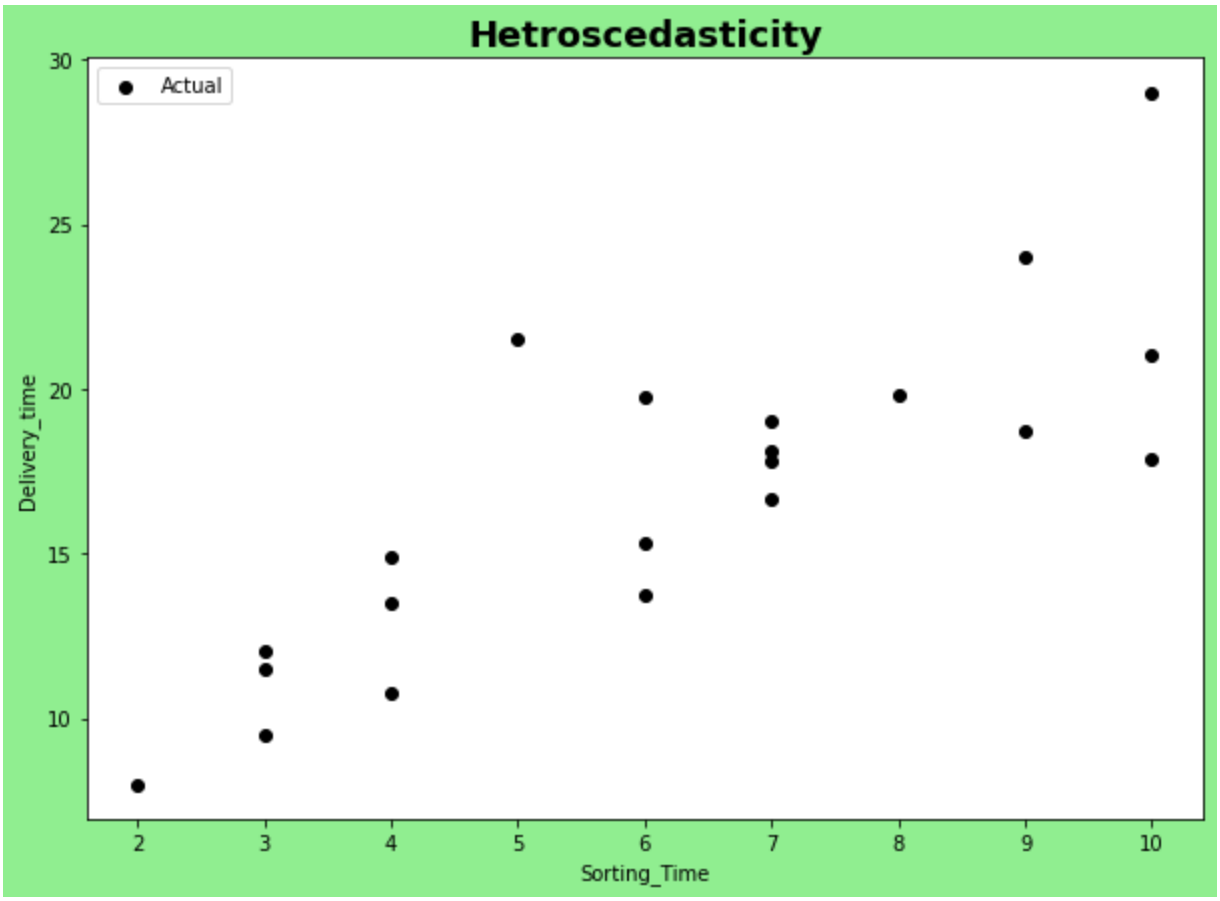
In [17]:
```
sns.scatterplot(df1.Sorting_Time,df1.Delivery_Time)
```

Out[17]:
```
<AxesSubplot:xlabel='Sorting_Time', ylabel='Delivery_Time'>
```



In [18]:
```
plt.figure(figsize=(10,7),facecolor='lightgreen')
plt.scatter(df1.Sorting_Time,df1.Delivery_Time,label = "Actual",color="black")
plt.xlabel("Sorting_Time")
plt.ylabel("Delivery_time")
plt.legend(loc="best")
plt.title("Hetroscedasticity",fontsize=18,fontweight='bold')
```

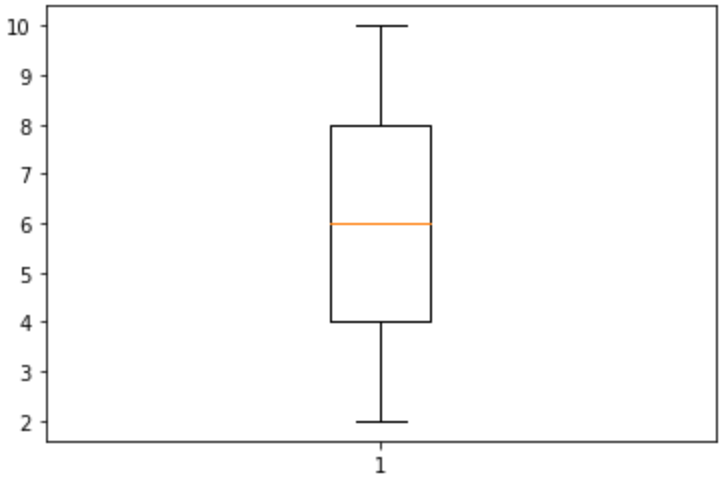
Out[18]:
```
Text(0.5, 1.0, 'Hetroscedasticity')
```



Checking Outlier in the data

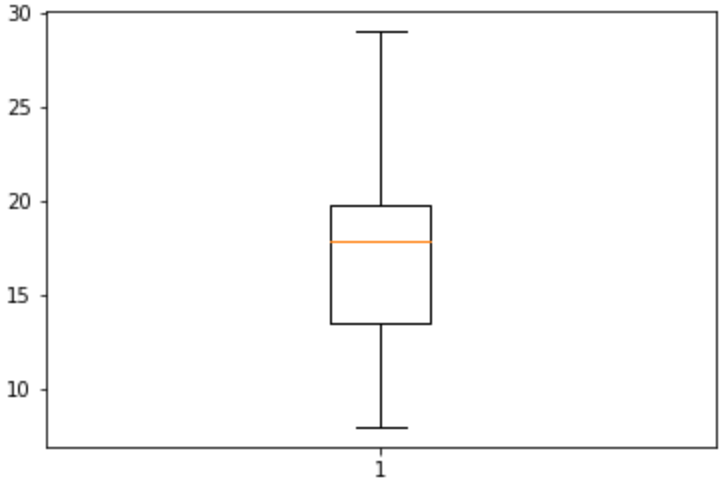
In [ ]:

Out[19]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x220d6b1d0d0>,
  <matplotlib.lines.Line2D at 0x220d6b1d430>],
 'caps': [<matplotlib.lines.Line2D at 0x220d6b1d790>,
  <matplotlib.lines.Line2D at 0x220d6b1daf0>],
 'boxes': [<matplotlib.lines.Line2D at 0x220d6b0ed30>],
 'medians': [<matplotlib.lines.Line2D at 0x220d6b1ddf0>],
 'fliers': [<matplotlib.lines.Line2D at 0x220d696a130>],
 'means': []}
```



In [20]:
```
plt.boxplot(df1.Delivery_Time)
```

Out[20]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x220d69afe50>,
  <matplotlib.lines.Line2D at 0x220d69c11f0>],
 'caps': [<matplotlib.lines.Line2D at 0x220d69c1550>,
  <matplotlib.lines.Line2D at 0x220d69c18b0>],
 'boxes': [<matplotlib.lines.Line2D at 0x220d69afaf0>],
 'medians': [<matplotlib.lines.Line2D at 0x220d69c1c10>],
 'fliers': [<matplotlib.lines.Line2D at 0x220d69c1f70>],
 'means': []}
```



In [21]:
```
df1.describe()
```

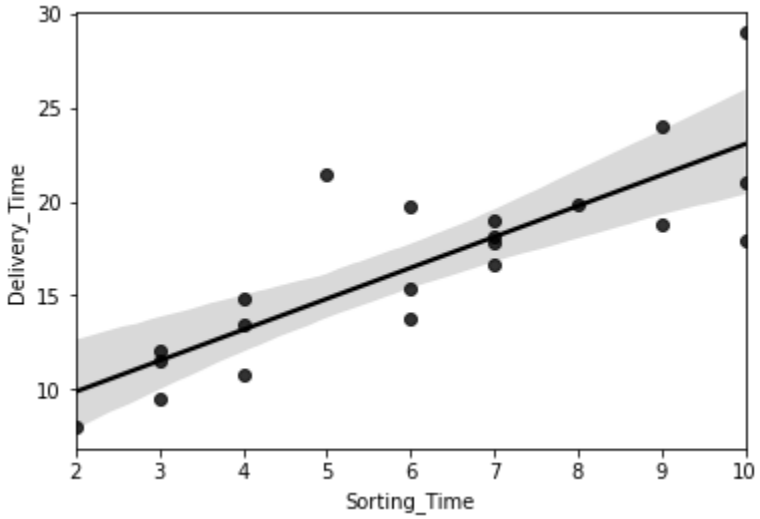
Out[21]:

	Delivery_Time	Sorting_Time
count	21.000000	21.000000
mean	16.790952	6.190476
std	5.074901	2.542028
min	8.000000	2.000000

	Delivery_Time	Sorting_Time
50%	17.830000	6.000000
75%	19.750000	8.000000
max	29.000000	10.000000

In [25]:
```
sns.regplot(x="Sorting_Time",y="Delivery_Time",data=df1,color="black")
```

Out[25]:
```
<AxesSubplot:xlabel='Sorting_Time', ylabel='Delivery_Time'>
```



Creating Model

In [26]:
```
model=smf.ols("Delivery_Time~Sorting_Time",data=df1).fit()
```

In [27]:
```
pred_train=model.predict(df1["Sorting_Time"])
```

In [28]:
```
r2_score(df1["Delivery_Time"],pred_train)
```

Out[28]:
```
0.6822714748417231
```

In [29]:
```
model.summary()
```

Out[29]:

OLS Regression Results

Dep. Variable:	Delivery_Time	**R-squared:**	0.682
Model:	OLS	**Adj. R-squared:**	0.666
Method:	Least Squares	**F-statistic:**	40.80
Date:	Sun, 10 Apr 2022	**Prob (F-statistic):**	3.98e-06
Time:	20:13:10	**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
Sorting_Time	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.

MSE

In [30]:
```
model.mse_resid
```

Out[30]:
```
8.613660132645544
```

RMSE

In [31]:
```
np.sqrt(model.mse_resid)
```

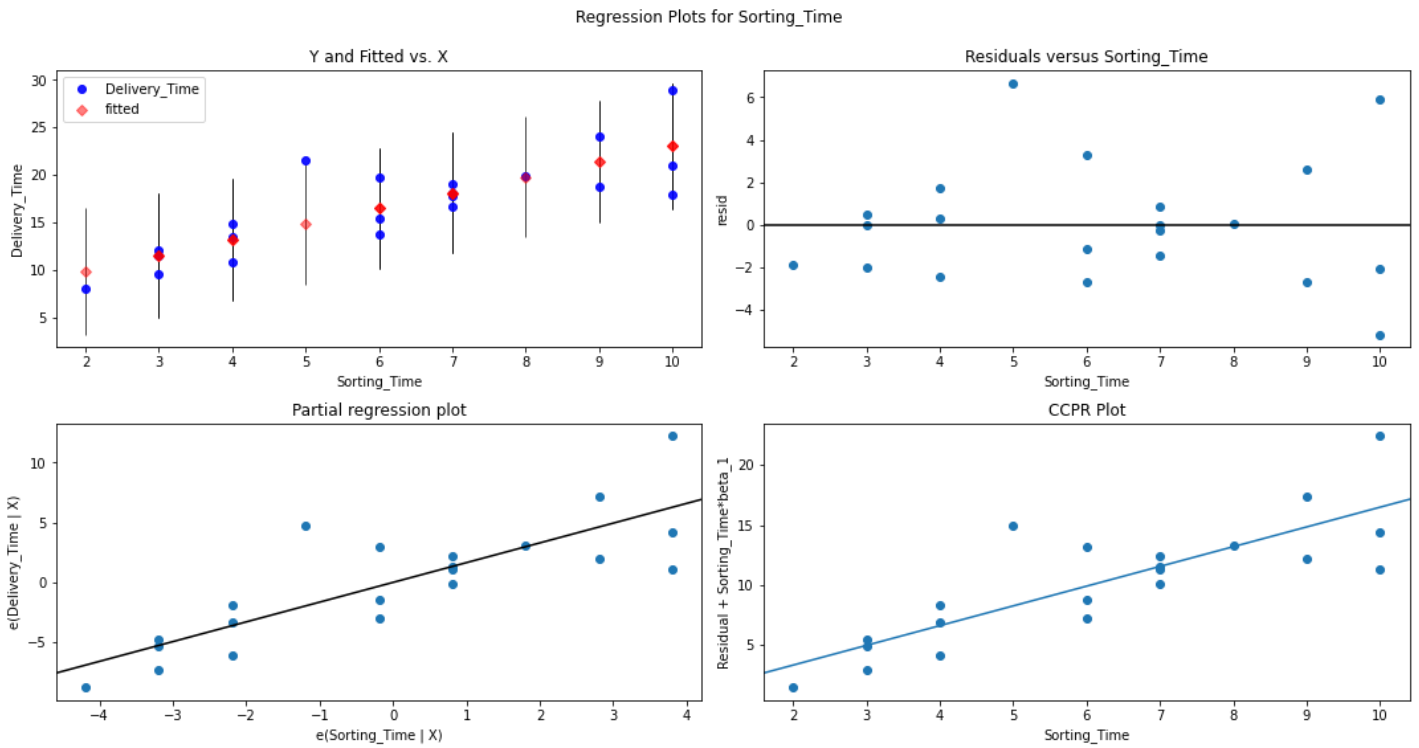
Out[31]:
```
2.9349037688901394
```

In [32]:
```
model.params
```

Out[32]:
```
Intercept       6.582734
Sorting_Time    1.649020
dtype: float64
```

Residual vs Regressor

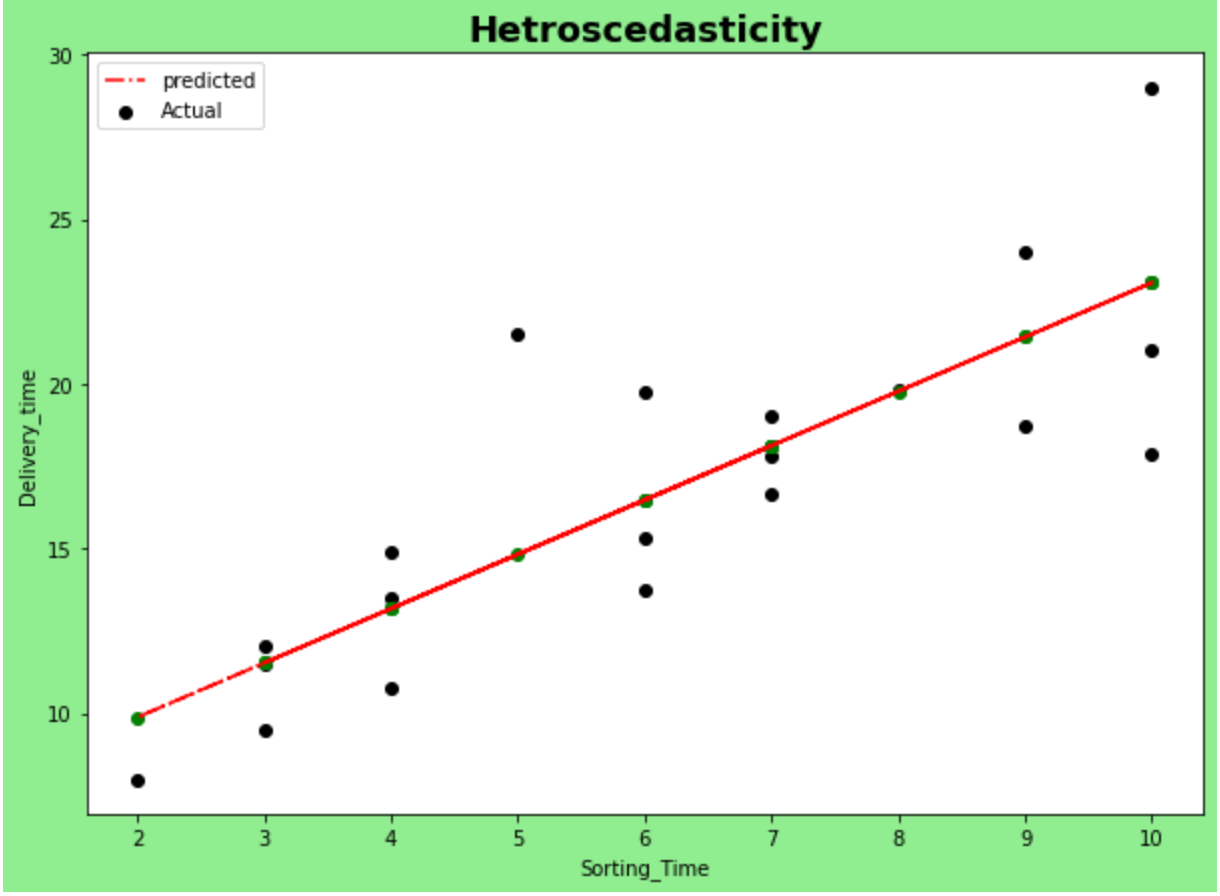
In [33]:
```
fig = plt.figure(figsize=(15,8))
fig = sm.graphics.plot_regress_exog(model, "Sorting_Time",fig=fig)
plt.show()
```



In [34]:

```
plt.figure(figsize=(10,7),facecolor='lightgreen')
plt.scatter(df1.Sorting_Time,df1.Delivery_Time,label="Actual",color="black")
plt.plot(df1.Sorting_Time,model.predict(df1["Sorting_Time"]),color="red",linestyle='-.',la
plt.scatter(df1.Sorting_Time,model.predict(df1["Sorting_Time"]),color="green")
plt.xlabel("Sorting_Time")
plt.ylabel("Delivery_time")
plt.title("Hetroscedasticity",fontsize=18,fontweight='bold')
xlim=(0,60)
ylim=(0,300)
plt.legend(loc='best')
```

Out[34]: <matplotlib.legend.Legend at 0x220d82fddf0>



Transformation

In [35]:
```
model1 = smf.ols("np.log(Delivery_Time)~Sorting_Time",data=df1).fit()
```

In [36]:
```
model1.summary()
```

Out[36]:

OLS Regression Results

Dep. Variable:	np.log(Delivery_Time)	**R-squared:**	0.711
Model:	OLS	**Adj. R-squared:**	0.696
Method:	Least Squares	**F-statistic:**	46.73
Date:	Sun, 10 Apr 2022	**Prob (F-statistic):**	1.59e-06
Time:	20:13:18	**Log-Likelihood:**	7.7920
No. Observations:	21	**AIC:**	-11.58
Df Residuals:	19	**BIC:**	-9.495
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.1214	0.103	20.601	0.000	1.906	2.337
Sorting_Time	0.1056	0.015	6.836	0.000	0.073	0.138

Omnibus:	1.238	**Durbin-Watson:**	1.325
Prob(Omnibus):	0.538	**Jarque-Bera (JB):**	0.544
			0.762

Kurtosis:	3.067	**Cond. No.**	18.3

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

MSE

In [37]:
```
model1.mse_resid
```

Out[37]:
```
0.03081093869566262
```

RMSE

In [38]:
```
np.sqrt(model1.mse_resid)
```

Out[38]:
```
0.1755304494828821
```

In [39]:
```
model2 = smf.ols("Delivery_Time~np.log(Sorting_Time)",data=df1).fit()
```

In [40]:
```
model2.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	Delivery_Time	**R-squared:**	0.695
Model:	OLS	**Adj. R-squared:**	0.679
Method:	Least Squares	**F-statistic:**	43.39
Date:	Sun, 10 Apr 2022	**Prob (F-statistic):**	2.64e-06
Time:	20:13:22	**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(Sorting_Time)	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.

MSE

In [41]:
```
model2.mse_resid
```

Out[41]:
```
8.256565933679841
```

RMSE

In [42]:
```
np.sqrt(model2.mse_resid)
```

Out[42]:
```
2.87342407828706
```

In [43]:
```
df1["Sorting_square"]=df1.Sorting_Time**2
model3=smf.ols('np.log(Delivery_Time)~Sorting_Time+Sorting_square',data=df1).fit()
model3.summary()
```

Out[43]:

OLS Regression Results

Dep. Variable:	np.log(Delivery_Time)	**R-squared:**	0.765
Model:	OLS	**Adj. R-squared:**	0.739
Method:	Least Squares	**F-statistic:**	29.28
Date:	Sun, 10 Apr 2022	**Prob (F-statistic):**	2.20e-06
Time:	20:13:26	**Log-Likelihood:**	9.9597
No. Observations:	21	**AIC:**	-13.92
Df Residuals:	18	**BIC:**	-10.79
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.6997	0.228	7.441	0.000	1.220	2.180
Sorting_Time	0.2659	0.080	3.315	0.004	0.097	0.434
Sorting_square	-0.0128	0.006	-2.032	0.057	-0.026	0.000

Omnibus:	2.548	**Durbin-Watson:**	1.369
Prob(Omnibus):	0.280	**Jarque-Bera (JB):**	1.777
Skew:	0.708	**Prob(JB):**	0.411
Kurtosis:	2.846	**Cond. No.**	373.

Notes:

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

MSE

In [44]:
```
model3.mse_resid
```

Out[44]:
```
0.026455998316946054
```

RMSE

In [45]:
```
np.sqrt(model3.mse_resid)
```

Out[45]:
```
0.16265299971702352
```

Predict new values

In [46]:
```
##Manually
data_predict=pd.DataFrame()
data_predict["Sorting_Time"]=pd.Series([2,3])
```

In [47]:
```
data_predict
```

Out[47]:

	Sorting_Time
0	2
1	3

In [48]:
```
data_predict["Delivery_Time"]=pd.Series(model.predict(data_predict))
```

In [49]:
```
data_predict
```

Out[49]:

	Sorting_Time	Delivery_Time
0	2	9.880774
1	3	11.529794

Predict New Delivery Time using DataSet

In [50]:
```
Predict = pd.DataFrame()
Predict["Sorting_Time"]=df1.Sorting_Time
Predict["Delivery_Time"]=df1.Delivery_Time
Predict["Predicted_Delivery_Time"]=pd.DataFrame(model.predict(df1.Sorting_Time))
Predict
```

Out[50]:

	Sorting_Time	Delivery_Time	Predicted_Delivery_Time
0	10	21.00	23.072933
1	4	13.50	13.178814
2	6	19.75	16.476853

	Sorting_Time	Delivery_Time	Predicted_Delivery_Time
3	9	24.00	21.423913
4	10	29.00	23.072933
5	6	15.35	16.476853
6	7	19.00	18.125873
7	3	9.50	11.529794
8	10	17.90	23.072933
9	9	18.75	21.423913
10	8	19.83	19.774893
11	4	10.75	13.178814
12	7	16.68	18.125873
13	3	11.50	11.529794
14	3	12.03	11.529794
15	4	14.88	13.178814
16	6	13.75	16.476853
17	7	18.11	18.125873
18	2	8.00	9.880774
19	7	17.83	18.125873
20	5	21.50	14.827833

In [ ]:

In [ ]: