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

In [2]:
```
delivery=pd.read_csv('delivery_time.csv')
delivery
```

Out[2]:

	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 [3]:
```
delivery.shape
```

Out[3]:
```
(21, 2)
```

In [4]:
```
delivery.isnull().sum()
```

Out[4]:
```
Delivery Time    0
Sorting Time     0
dtype: int64
```

In [5]:
```
delivery.dtypes
```

Out[5]:
```
Delivery Time    float64
Sorting Time       int64
dtype: object
```

In [6]:

```
delivery=delivery.rename({'Delivery Time':'delivery_time','Sorting Time':'sorting_time'},axis=1)
delivery
```

Out[6]:

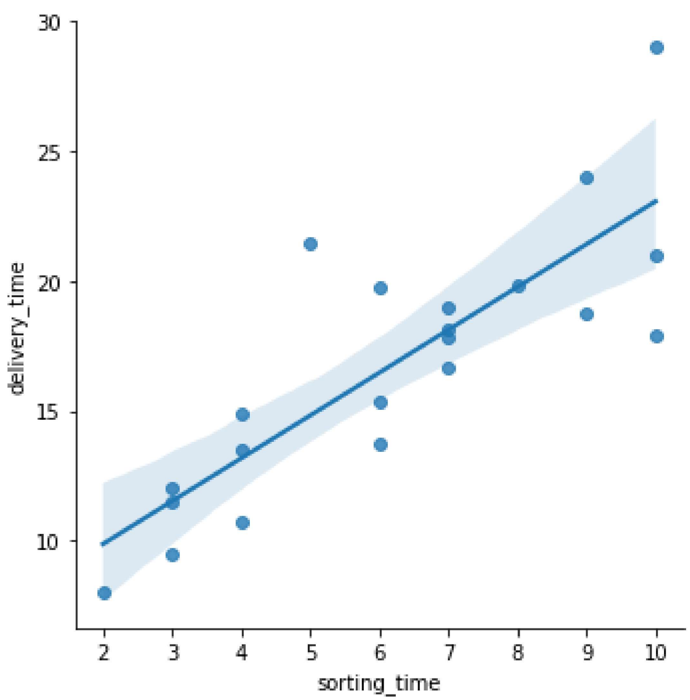
	delivery_time	sorting_time
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
4	29.00	10
5	15.35	6
6	19.00	7
7	9.50	3
8	17.90	10
9	18.75	9
10	19.83	8
11	10.75	4
12	16.68	7
13	11.50	3
14	12.03	3
15	14.88	4
16	13.75	6
17	18.11	7
18	8.00	2
19	17.83	7
20	21.50	5

ASSUMPTION CHECK

1.Linear check

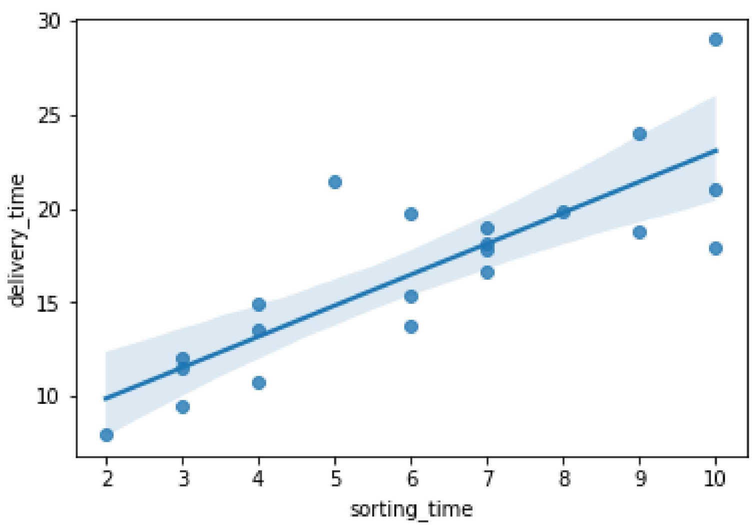
In [7]:
```
sns.lmplot(x='sorting_time',y='delivery_time',data=delivery)
plt.title(sorting_time_vs_delivery_time)
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_11336/378080106.py in <module>
      1 sns.lmplot(x='sorting_time',y='delivery_time',data=delivery)
----> 2 plt.title(sorting_time_vs_delivery_time)
      3 plt.show()

NameError: name 'sorting_time_vs_delivery_time' is not defined
```



In [8]:
```
sns.regplot(x='sorting_time',y='delivery_time',data=delivery)
plt.title(sorting_time_vs_delivery_time)
plt.show()
```

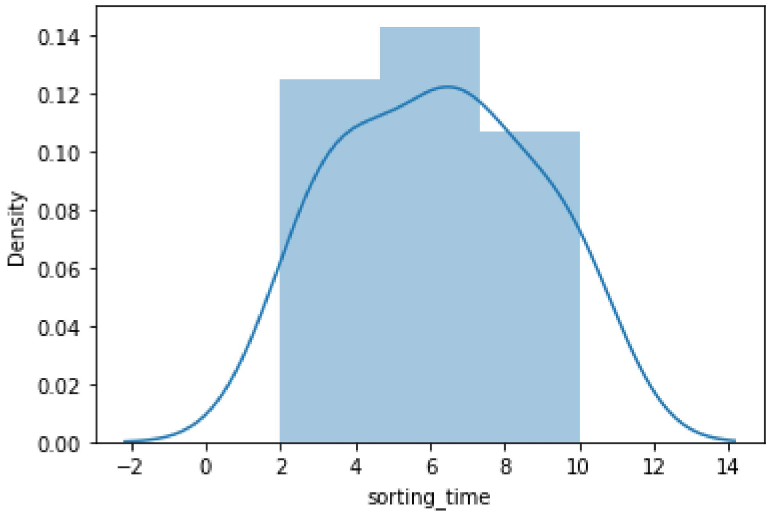
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_11336/1624518498.py in <module>
      1 sns.regplot(x='sorting_time',y='delivery_time',data=delivery)
----> 2 plt.title(sorting_time_vs_delivery_time)
      3 plt.show()

NameError: name 'sorting_time_vs_delivery_time' is not defined
```



by this we can say linearity test failed

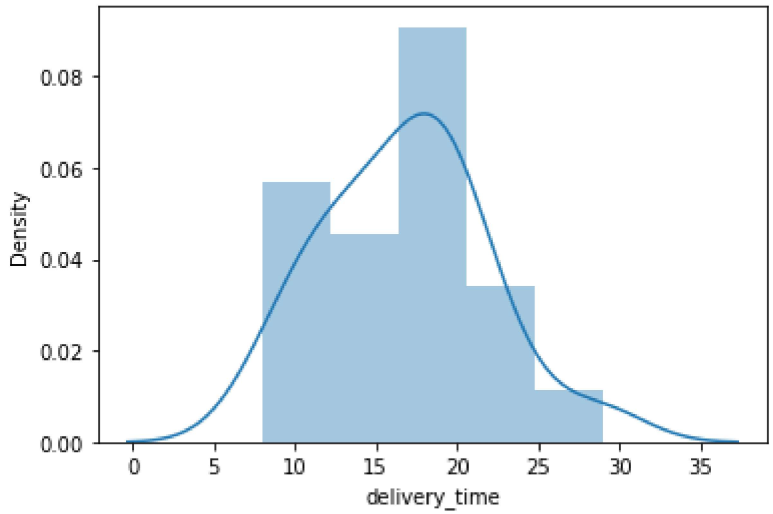
2. Distribution check

In [9]:
```
sns.distplot(a=delivery['sorting_time'],hist=True)
plt.show()
```



In [10]:
```
sns.distplot(a=delivery['delivery_time'],hist=True)
sns.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_11336/546009151.py in <module>
      1 sns.distplot(a=delivery['delivery_time'],hist=True)
----> 2 sns.show()

AttributeError: module 'seaborn' has no attribute 'show'
```



In [11]:
```
delivery.skew()
```

Out[11]:
```
delivery_time    0.352390
sorting_time     0.047115
dtype: float64
```

In [12]:
```
delivery.kurtosis()
```

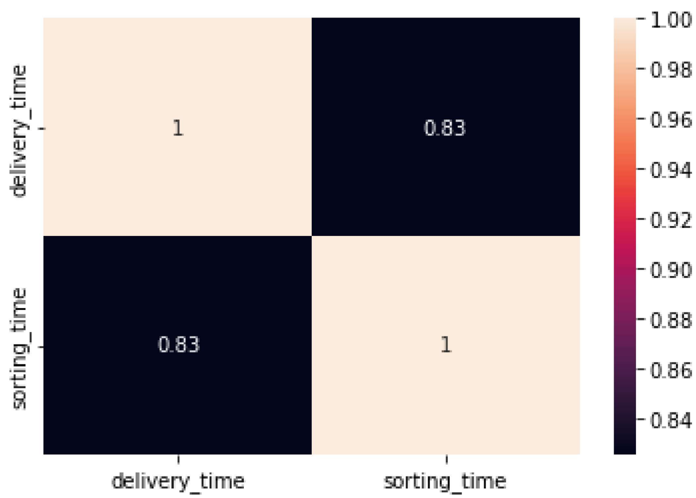
Out[12]:
```
delivery_time    0.317960
sorting_time    -1.148455
dtype: float64
```

In [13]:
```
delivery.corr()
```

Out[13]:

	delivery_time	sorting_time
delivery_time	1.000000	0.825997
sorting_time	0.825997	1.000000

In [14]:
```
sns.heatmap(delivery.corr(),annot=True)
plt.show()
```



model building || model training

In [15]:
```
model_1=smf.ols(formula='delivery_time~sorting_time', data=delivery).fit()
model_1
```

Out[15]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x2044a1ec9d0>
```

In [16]:
```
model_1.params
```

Out[16]:
```
Intercept       6.582734
sorting_time    1.649020
dtype: float64
```

In [17]:
```
model_1.pvalues
```

Out[17]:
```
Intercept       0.001147
sorting_time    0.000004
dtype: float64
```

model testing

In [18]:
```
y=(3*1.649020)+6.582734 #manual prediction for 3 sorting_time
y
```

Out[18]:
```
11.529793999999999
```

In [19]:
```
z=(5*1.649020)+6.582734  #manual prediction for 5 sorting_time
z
```

Out[19]:
```
14.827834
```

In [20]:
```
data=pd.Series([3,5])# auto prediction for 3 and 5 sorting_time
data
```

Out[20]:
```
0    3
1    5
dtype: int64
```

In [21]:
```
data_pred=pd.DataFrame(data,columns=['sorting_time'])
data_pred
```

Out[21]:

	sorting_time
0	3
1	5

In [22]:
```
model_pred=model_1.predict(data_pred)
model_pred
```

Out[22]:
```
0    11.529794
1    14.827833
dtype: float64
```

In [23]:
```
model_1=smf.ols(formula='delivery_time~sorting_time',data=delivery).fit()
print('R-square                           : ',round(model_1.rsquared,4))
print('Adjusted R-square                  : ',round(model_1.rsquared_adj,4))
print('Akaike information criterion (AIC) : ',round(model_1.aic,4))
print('Bayesian information criterion(BIC): ',round(model_1.bic,4))
```

```
R-square                           :  0.6823
Adjusted R-square                  :  0.6655
Akaike information criterion (AIC) :  106.714
Bayesian information criterion(BIC):  108.803
```

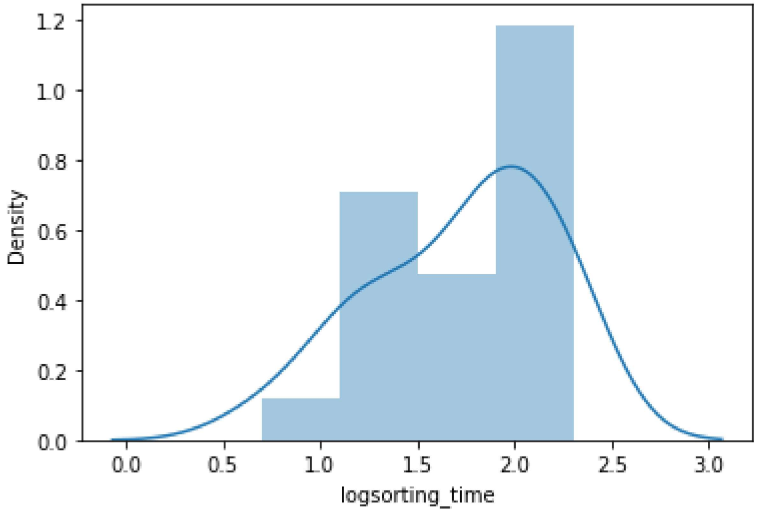
1. Log Transformations

In [24]:
```
delivery['logsorting_time']=np.log(delivery['sorting_time'])
delivery
```

Out[24]:

	delivery_time	sorting_time	logsorting_time
0	21.00	10	2.302585
1	13.50	4	1.386294
2	19.75	6	1.791759
3	24.00	9	2.197225
4	29.00	10	2.302585
5	15.35	6	1.791759
6	19.00	7	1.945910
7	9.50	3	1.098612
8	17.90	10	2.302585
9	18.75	9	2.197225
10	19.83	8	2.079442
11	10.75	4	1.386294
12	16.68	7	1.945910
13	11.50	3	1.098612
14	12.03	3	1.098612
15	14.88	4	1.386294
16	13.75	6	1.791759
17	18.11	7	1.945910
18	8.00	2	0.693147
19	17.83	7	1.945910
20	21.50	5	1.609438

In [25]:
```
sns.distplot(a=delivery['logsorting_time'],hist=True)
plt.show()
```



In [26]:
```
Log_model=smf.ols(formula='delivery_time~logsorting_time', data=delivery).fit()
Log_model
```

Out[26]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x2044a2dacd0>
```

In [27]:

```
1 Log_model.summary()
```

Out[27]:

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:	Fri, 01 Jul 2022	**Prob (F-statistic):**	2.64e-06
Time:	11:49:16	**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

In [28]:

```
1 Log_model.rsquared
```

Out[28]: 0.6954434611324223

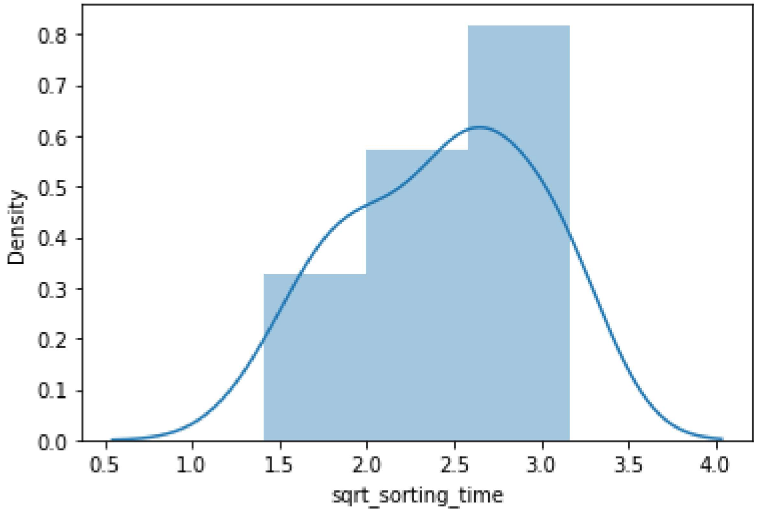
2. SQRT Model

In [29]:
```
delivery['sqrt_sorting_time']=np.sqrt(delivery['sorting_time'])
delivery
```

Out[29]:

	delivery_time	sorting_time	logsorting_time	sqrt_sorting_time
0	21.00	10	2.302585	3.162278
1	13.50	4	1.386294	2.000000
2	19.75	6	1.791759	2.449490
3	24.00	9	2.197225	3.000000
4	29.00	10	2.302585	3.162278
5	15.35	6	1.791759	2.449490
6	19.00	7	1.945910	2.645751
7	9.50	3	1.098612	1.732051
8	17.90	10	2.302585	3.162278
9	18.75	9	2.197225	3.000000
10	19.83	8	2.079442	2.828427
11	10.75	4	1.386294	2.000000
12	16.68	7	1.945910	2.645751
13	11.50	3	1.098612	1.732051
14	12.03	3	1.098612	1.732051
15	14.88	4	1.386294	2.000000
16	13.75	6	1.791759	2.449490
17	18.11	7	1.945910	2.645751
18	8.00	2	0.693147	1.414214
19	17.83	7	1.945910	2.645751
20	21.50	5	1.609438	2.236068

In [30]:
```
sns.distplot(a=delivery['sqrt_sorting_time'],hist=True)
plt.show()
```



In [31]:
```
sqrt_model=smf.ols(formula='delivery_time~sqrt_sorting_time', data=delivery).fit()
sqrt_model.summary()
```

Out[31]:

OLS Regression Results

Dep. Variable:	delivery_time	**R-squared:**	0.696
Model:	OLS	**Adj. R-squared:**	0.680
Method:	Least Squares	**F-statistic:**	43.46
Date:	Fri, 01 Jul 2022	**Prob (F-statistic):**	2.61e-06
Time:	11:49:19	**Log-Likelihood:**	-50.900
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	-2.5188	2.995	-0.841	0.411	-8.788	3.751
sqrt_sorting_time	7.9366	1.204	6.592	0.000	5.417	10.456

Omnibus:	4.658	**Durbin-Watson:**	1.318
Prob(Omnibus):	0.097	**Jarque-Bera (JB):**	2.824
Skew:	0.865	**Prob(JB):**	0.244
Kurtosis:	3.483	**Cond. No.**	13.7

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

In [32]:
```
sqrt_model.rsquared
```

Out[32]: 0.6958062276308671

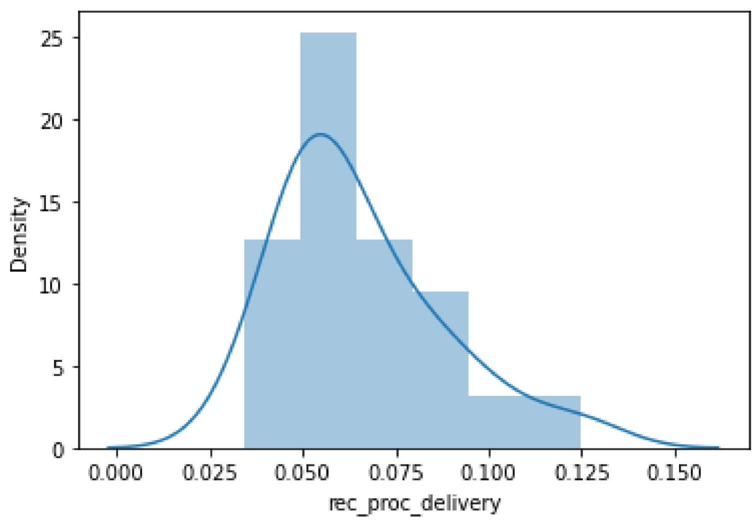
3.Reciprocal

In [33]:
```
delivery['rec_proc_delivery']=1/delivery['delivery_time']
delivery.head()
```

Out[33]:

	delivery_time	sorting_time	logsorting_time	sqrt_sorting_time	rec_proc_delivery
0	21.00	10	2.302585	3.162278	0.047619
1	13.50	4	1.386294	2.000000	0.074074
2	19.75	6	1.791759	2.449490	0.050633
3	24.00	9	2.197225	3.000000	0.041667
4	29.00	10	2.302585	3.162278	0.034483

In [34]:
```
sns.distplot(a=delivery['rec_proc_delivery'],hist=True)
plt.show()
```



In [35]:
```
rec_proc_model=smf.ols(formula='rec_proc_delivery~sorting_time', data=delivery).fit()
rec_proc_model.summary()
```

Out[35]:

OLS Regression Results

Dep. Variable:	rec_proc_delivery	**R-squared:**	0.682
Model:	OLS	**Adj. R-squared:**	0.665
Method:	Least Squares	**F-statistic:**	40.68
Date:	Fri, 01 Jul 2022	**Prob (F-statistic):**	4.06e-06
Time:	11:49:21	**Log-Likelihood:**	62.471
No. Observations:	21	**AIC:**	-120.9
Df Residuals:	19	**BIC:**	-118.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1107	0.008	14.526	0.000	0.095	0.127
sorting_time	-0.0073	0.001	-6.378	0.000	-0.010	-0.005

Omnibus:	1.096	**Durbin-Watson:**	1.555
Prob(Omnibus):	0.578	**Jarque-Bera (JB):**	0.224
Skew:	0.199	**Prob(JB):**	0.894
Kurtosis:	3.313	**Cond. No.**	18.3

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

In [36]:
```
rec_proc_model.rsquared
```

Out[36]: 0.6816508639250471

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```

=> By comparing the rsquare values of all 3 model and raw model we can say SQRT_model gives better rsquare value. so we will select Sqrt_model and do futhur calculation

The end!!!

In [ ]:
```
1
```