In [1]:
```
# Necessary Imports
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

import statsmodels.api as sm

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Import data
df = pd.read_csv('delivery_time.csv')
df.head(7)
```

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

EDA

In [3]:
```
df.shape
```

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

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

Out[4]:
```
Delivery_Time    0.352390
Sorting_Time     0.047115
dtype: float64
```

In [5]:
```
#Co-relation of Target and Feature
sns.heatmap(df.corr(), annot = True)
```

Out[5]: <AxesSubplot:>

In [6]:
```
df.info()                                          #no null values in dataset
```

```
<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 [7]:
```
df.describe()
```

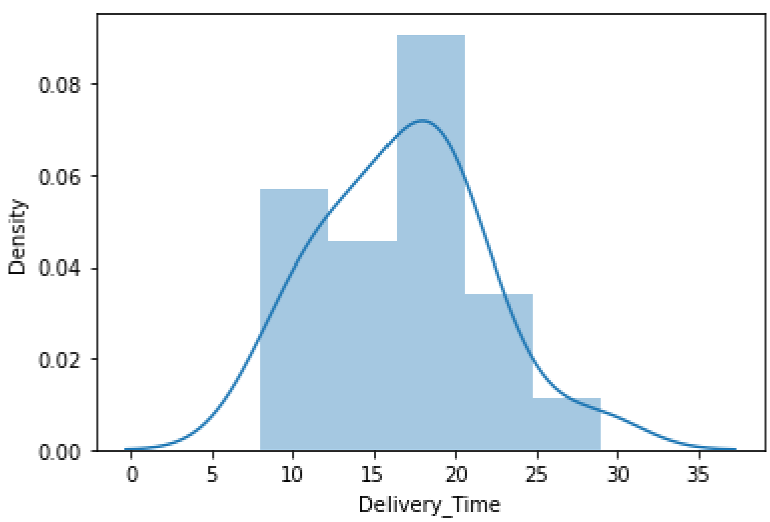
Out[7]:

	Delivery_Time	Sorting_Time
count	21.000000	21.000000
mean	16.790952	6.190476
std	5.074901	2.542028
min	8.000000	2.000000
25%	13.500000	4.000000
50%	17.830000	6.000000
75%	19.750000	8.000000
max	29.000000	10.000000

Graphical Univariate analysis on dataset

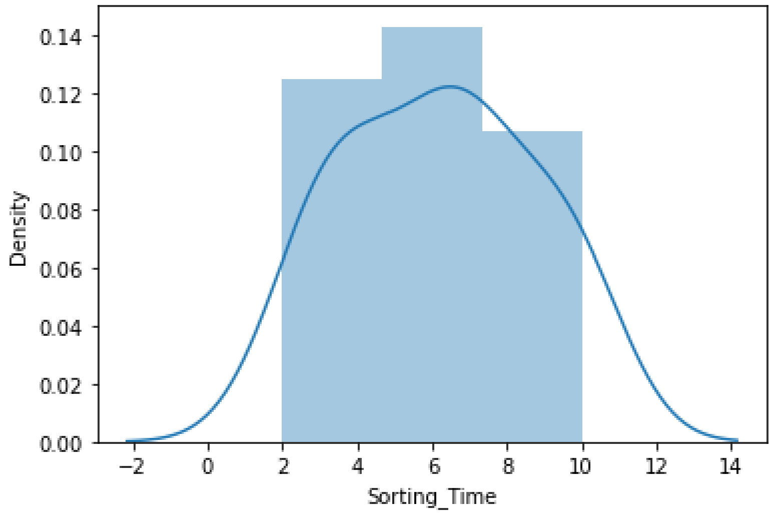
In [8]:
```
sns.distplot(df.Delivery_Time)
```

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



In [9]:
```
sns.distplot(df.Sorting_Time)
```

Out[9]: <AxesSubplot:xlabel='Sorting_Time', ylabel='Density'>



Transforming data - Log scale

In [10]:
```
df['dt'] =  np.log(df['Delivery_Time'])
df['st'] =  np.log(df['Sorting_Time'])
df.head()
```

Out[10]:

	Delivery_Time	Sorting_Time	dt	st
0	21.00	10	3.044522	2.302585
1	13.50	4	2.602690	1.386294
2	19.75	6	2.983153	1.791759
3	24.00	9	3.178054	2.197225
4	29.00	10	3.367296	2.302585

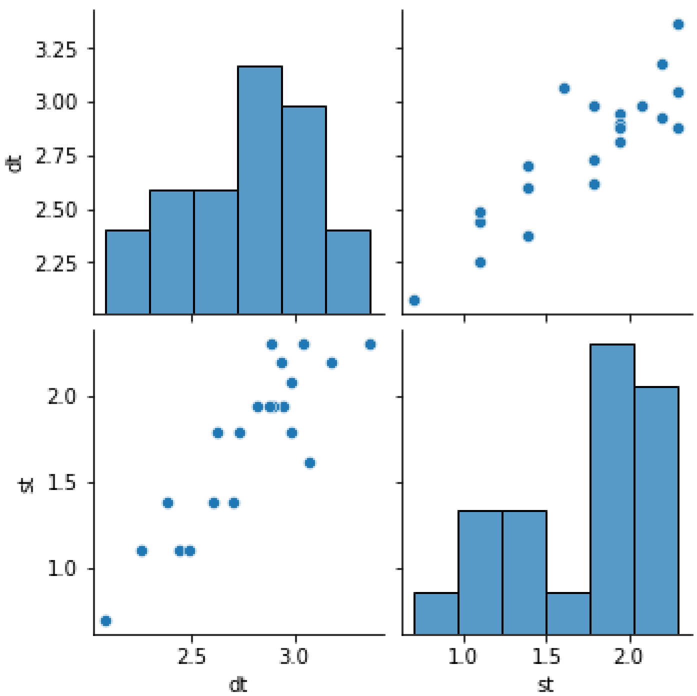
In [11]:
```
df.skew()
```

Out[11]:
```
Delivery_Time    0.352390
Sorting_Time     0.047115
dt              -0.451290
st              -0.605236
dtype: float64
```

Graphical Bi-variate analysis on dataset

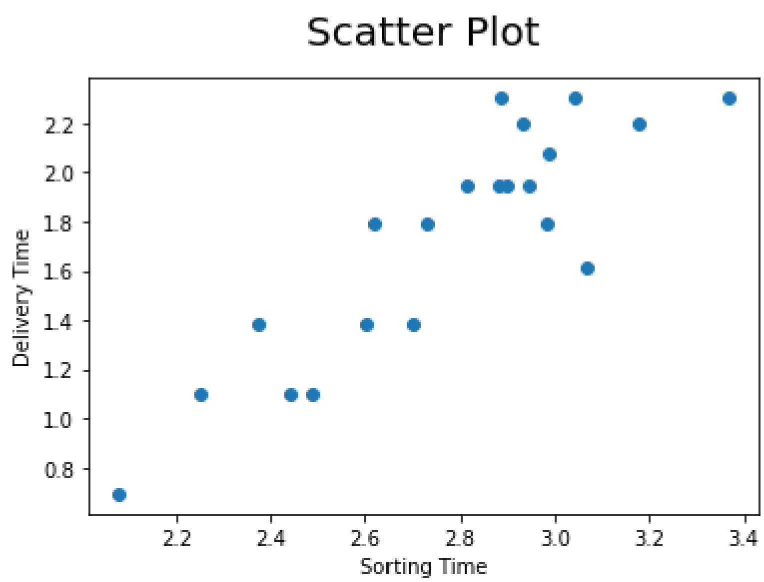
In [12]:
```
#Pairplot of predictor and target
sns.pairplot(df,vars =['dt','st'])
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x23c241481c0>
```



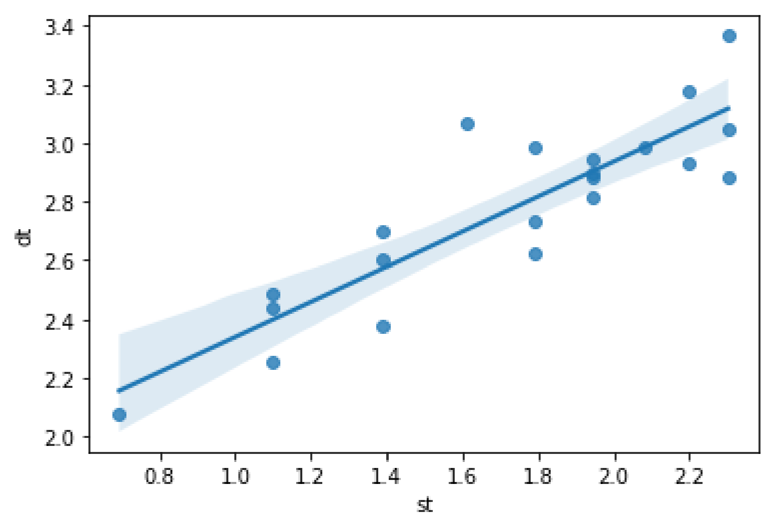
In [13]:
```
y = df['dt']
x = df['st']
plt.scatter(y,x)                                    # Scatter plot of new Delivery_t

plt.xlabel('Sorting Time', fontsize = 10)           # Named the axes
plt.ylabel('Delivery Time', fontsize = 10)
plt.title(label='Scatter Plot', fontsize=20, y=1.05)
plt.show()                                          # Show the plot
```



Fitting a Linear Regression Model

In [14]:
```
sns.regplot(x="st", y="dt", data=df);
```



In [15]:
```
import statsmodels.formula.api as smf

model = smf.ols('dt~st', data = df).fit()
model.summary()
```

Out[15]:

OLS Regression Results

Dep. Variable:	dt	**R-squared:**	0.772
Model:	OLS	**Adj. R-squared:**	0.760
Method:	Least Squares	**F-statistic:**	64.39
Date:	Fri, 21 Oct 2022	**Prob (F-statistic):**	1.60e-07
Time:	21:13:02	**Log-Likelihood:**	10.291
No. Observations:	21	**AIC:**	-16.58
Df Residuals:	19	**BIC:**	-14.49
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1.7420	0.133	13.086	0.000	1.463	2.021
st	0.5975	0.074	8.024	0.000	0.442	0.753

Omnibus:	1.871	**Durbin-Watson:**	1.322
Prob(Omnibus):	0.392	**Jarque-Bera (JB):**	1.170
Skew:	0.577	**Prob(JB):**	0.557
Kurtosis:	2.916	**Cond. No.**	9.08

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

'Sorting time' p value from summary = 0.000 Hence, the 'Sorting time' is an important feature for the Target 'Delivery Time'

The feature Const p value = 0.000 Hence, Constant is also an important feature for the Target 'Delivery Time'

R-sqaured is closer to 1 that means the regression model covers most part of the variance of the values of the response variable and can be termed as a good model.

In [16]:
```
#Co-efficients values
#beta1 and bet0 values
model.params
```

Out[16]:
```
Intercept    1.741987
st           0.597522
dtype: float64
```

In [17]:
```
#t and p-Values for intercept and Sorting Time
print(model.tvalues, '\n', model.pvalues)
```

```
Intercept    13.085552
st            8.024484
dtype: float64
 Intercept    5.921137e-11
st           1.601539e-07
dtype: float64
```