1. Import necessasry libraries

In [1]:
```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

2. Import data ¶

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

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

3. Initial Analysis

In [4]:
```
delivery_time.shape
```

Out[4]:
```
(21, 2)
```

In [5]:
```
delivery_time.isna().sum()
```

Out[5]:
```
Delivery Time    0
Sorting Time     0
dtype: int64
```

In [6]:
```
delivery_time.dtypes
```

Out[6]:
```
Delivery Time    float64
Sorting Time       int64
dtype: object
```

In [7]:
```
delivery_time.describe(include= 'all')
```

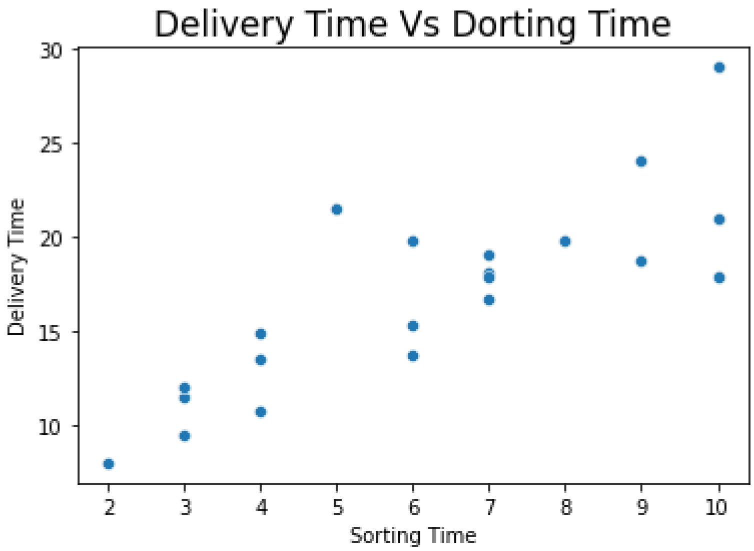
Out[7]:

	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

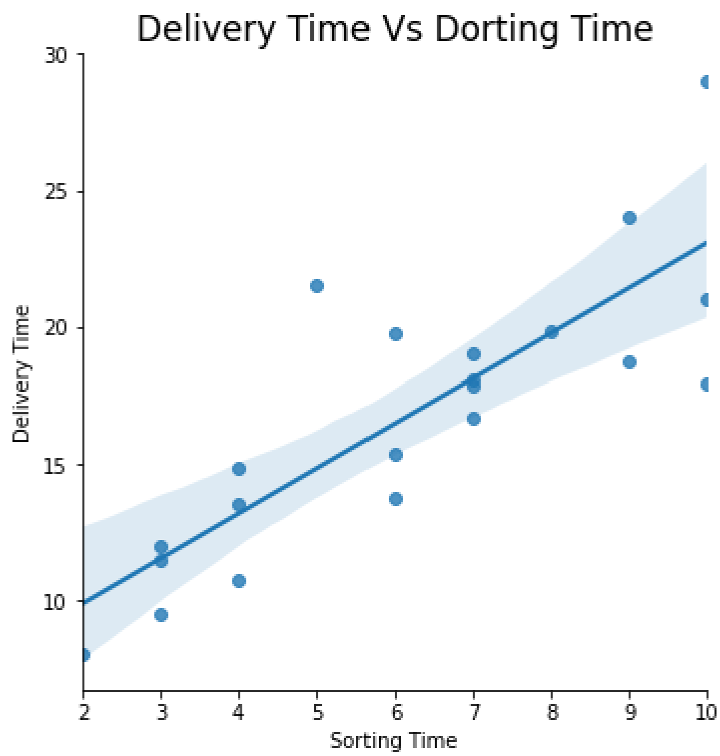
4.ASSUMPTION CHECK

1.Check for linearity

In [8]:
```
sns.scatterplot(x = 'Sorting Time',y= 'Delivery Time',data= delivery_time)
plt.title('Delivery Time Vs Dorting Time',size= 17)
plt.show()
```



In [9]:
```
sns.lmplot(x = 'Sorting Time',y= 'Delivery Time',data= delivery_time)
plt.title('Delivery Time Vs Dorting Time',size= 17)
plt.show()
```



In [10]:
```
delivery_time.corr().round(2)
```

Out[10]:

	Delivery Time	Sorting Time
Delivery Time	1.00	0.83
Sorting Time	0.83	1.00

2. Homoscedasticity - It can be checked post model building and

training.

3. No Multicollinearity - Condition satisfied

4.No Autoregression - Condition satisfied

5. Zero residual mean - It can be checked post model building and training.

4. Data Preparation

In [11]:
```
delivery_time # no unwanted parameters
```

Out[11]:

	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

6.Model Building

- There are basically 2 libraries that supports Linear Regression algorithm.
- 1.Statsmodels libraries-ols techniques.
- 2.sklearn libraries-linear regression.

using sklearn library - linear regression

In [18]:
```
x = delivery_time.drop(labels= 'Delivery Time',axis=1)
y= delivery_time[['Delivery Time']]
```

In [19]:
```
x
```

Out[19]:

	Sorting Time
0	10
1	4
2	6
3	9
4	10
5	6
6	7
7	3
8	10
9	9
10	8
11	4

In [20]:
```
y
```

Out[20]:

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

7.Model Training

In [21]:
```
from sklearn.linear_model import LinearRegression
```

In [22]:
```
Linear_model= LinearRegression() #model initialization
Linear_model.fit(x,y)#model training
```

Out[22]:
```
LinearRegression()
```

8.Model testing

training data

In [23]:
```
y_pred = Linear_model.predict(x)
y_pred
```

Out[23]:
```
array([[23.07293294],
       [13.17881356],
       [16.47685335],
       [21.42391304],
       [23.07293294],
       [16.47685335],
       [18.12587325],
       [11.52979366],
       [23.07293294],
       [21.42391304],
       [19.77489315],
       [13.17881356],
       [18.12587325],
       [11.52979366],
       [11.52979366],
       [13.17881356],
       [16.47685335],
       [18.12587325],
       [ 9.88077377],
       [18.12587325],
       [14.82783346]])
```

In [24]:
```
error = y-y_pred
error
```

Out[24]:

	Delivery Time
0	-2.072933
1	0.321186
2	3.273147
3	2.576087
4	5.927067
5	-1.126853
6	0.874127
7	-2.029794
8	-5.172933
9	-2.673913
10	0.055107
11	-2.428814
12	-1.445873
13	-0.029794
14	0.500206
15	1.701186
16	-2.726853
17	-0.015873
18	-1.880774
19	-0.295873
20	6.672167

In [25]:
```
from sklearn.metrics import mean_squared_error
```

In [26]:
```
mean_squared_error(y,y_pred) #evaluation metrix
```

Out[26]:
```
7.793311548584063
```

9.Model validation Techniques

In [27]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=
```

In [28]:
```
#Training data
x_train.shape,y_train.shape
```

Out[28]: ((16, 1), (16, 1))

In [29]:
```
#Test data
x_test.shape,y_test.shape
```

Out[29]: ((5, 1), (5, 1))

Model Training

In [30]:
```
from sklearn.linear_model import LinearRegression
```

In [31]:
```
linear_model_2 = LinearRegression() #Initialization
linear_model_2.fit(x_train,y_train) #Model Training
```

Out[31]: LinearRegression()

Model Testing

Training data

In [32]:
```
y_train_pred = Linear_model.predict(x_train)
y_train_pred
```

Out[32]:
```
array([[23.07293294],
       [11.52979366],
       [13.17881356],
       [ 9.88077377],
       [16.47685335],
       [23.07293294],
       [13.17881356],
       [18.12587325],
       [23.07293294],
       [18.12587325],
       [16.47685335],
       [21.42391304],
       [16.47685335],
       [18.12587325],
       [18.12587325],
       [13.17881356]])
```

In [33]:
```
mean_squared_error(y_train,y_train_pred)
```

Out[33]: 6.741746151715988

Test data

In [34]:
```
y_pred_test = linear_model_2.predict(x_test) #unseen by the model during the trai
y_pred_test
```

Out[34]:
```
array([[11.0825718 ],
       [21.55022193],
       [19.80561358],
       [14.57178851],
       [11.0825718 ]])
```

In [35]:
```
mean_squared_error(y_test,y_pred_test)
```

Out[35]:
```
11.70414636012243
```

Assumtion check

Homoscedaticity

In [36]:
```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
scaled_x = std_scaler.fit_transform(x)
print(scaled_x)
```

```
[[ 1.53562462]
 [-0.88298415]
 [-0.07678123]
 [ 1.13252315]
 [ 1.53562462]
 [-0.07678123]
 [ 0.32632023]
 [-1.28608562]
 [ 1.53562462]
 [ 1.13252315]
 [ 0.72942169]
 [-0.88298415]
 [ 0.32632023]
 [-1.28608562]
 [-1.28608562]
 [-0.88298415]
 [-0.07678123]
 [ 0.32632023]
 [-1.68918708]
 [ 0.32632023]
 [-0.47988269]]
```

In [37]:
```
scaled_x.mean(),scaled_x.std()
```

Out[37]:
```
(-7.137148015447435e-17, 1.0)
```

In [41]:
```
from sklearn.linear_model import LinearRegression
linear_model = LinearRegression()
linear_model.fit(scaled_x,y)
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
y_predicted = linear_model.predict(scaled_x)
y_predicted
```

Out[42]:
```
array([[23.07293294],
       [13.17881356],
       [16.47685335],
       [21.42391304],
       [23.07293294],
       [16.47685335],
       [18.12587325],
       [11.52979366],
       [23.07293294],
       [21.42391304],
       [19.77489315],
       [13.17881356],
       [18.12587325],
       [11.52979366],
       [11.52979366],
       [13.17881356],
       [16.47685335],
       [18.12587325],
       [ 9.88077377],
       [18.12587325],
       [14.82783346]])
```

In [43]:
```
error = y - y_predicted
error
```

Out[43]:

	Delivery Time
0	-2.072933
1	0.321186
2	3.273147
3	2.576087
4	5.927067
5	-1.126853
6	0.874127
7	-2.029794
8	-5.172933
9	-2.673913
10	0.055107
11	-2.428814
12	-1.445873
13	-0.029794
14	0.500206
15	1.701186
16	-2.726853
17	-0.015873
18	-1.880774
19	-0.295873
20	6.672167

In [38]:
```
import numpy as np
```

Zero residual mean.

In [39]:
```
np.mean(scaled_x)
```

Out[39]:
```
-7.137148015447435e-17
```

In [45]:
```
np.mean(error)
```

Out[45]:
```
Delivery Time   -2.622241e-15
dtype: float64
```