Importing Necessary Libraries

In [57]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
```

Business Understanding

Predicting delivary time using sorting time

Data Collection

In [58]:
```
delivery_data=pd.read_csv('delivery_time.csv')
delivery_data
```

Out[58]:

	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

Data Understanding

In [10]:
```
delivery_data.shape
```

Out[10]:
```
(21, 2)
```

In [11]:
```
delivery_data.dtypes
```

Out[11]:
```
Delivery Time    float64
Sorting Time       int64
dtype: object
```

In [12]:
```
delivery_data.isna().sum()
```

Out[12]:
```
Delivery Time    0
Sorting Time     0
dtype: int64
```

In [13]:
```
delivery_data.nunique()
```

Out[13]:
```
Delivery Time    21
Sorting Time      9
dtype: int64
```

In [16]:
```
delivery_data.describe(include = 'all')
```

Out[16]:

	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

Checking Assumptions for matching

In [68]:
```
plt.scatter(x = 'Delivery Time', y = 'Sorting Time',data = delivery_data)
plt.title('Scatter plot Delivery Time')
plt.xlabel('Sorting Time')
plt.ylabel('Delivery Time')
plt.show()
```

the data does contains some outliers, but there is potive correlation between delivery time and sorting Time

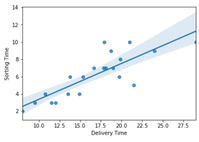
In [20]:
```
delivery_data.corr()
```

Out[20]:

	Delivery Time	Sorting Time
Delivery Time	1.000000	0.825997
Sorting Time	0.825997	1.000000

In [24]:
```
sns.regplot(x = 'Delivery Time',y='Sorting Time',data =delivery_data)
```

Out[24]:
```
<AxesSubplot:xlabel='Delivery Time', ylabel='Sorting Time'>
```



Model Training & Model Testing

```
There are basically two libraries for supporting linear regresssion algorithmns.
  statsmodel
  sklearn
```

In [41]:
```
delivery_data=delivery_data.rename({'Delivery Time':'delivery_time', 'Sorting Time':'sorting_time'},axis=1)
delivery_data
```

Out[41]:

	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 [48]:
```
#Ordinary Least square
linear_model = smf.ols(formula = 'delivery_time~sorting_time', data = delivery_data).fit()
#Model Training
```

Check for the deliverables of the training time

In [49]:
```
linear_model.params
```

Out[49]:
```
Intercept       6.582734
sorting_time    1.649020
dtype: float64
```

Model Testing

In [50]:
```
linear_model.tvalues , linear_model.pvalues #finding tvalues and pvalues
```

Out[50]:
```
(Intercept       3.823349
 sorting_time    6.387447
 dtype: float64,
 Intercept       0.001147
 sorting_time    0.000004
 dtype: float64)
```

In [51]:
```
linear_model.rsquared , linear_model.rsquared_adj #finding rsquared values
```

Out[51]:
```
(0.6822714748417231, 0.6655489208860244)
```

Model Predictions

In [52]:
```
# Manual prediction for say sorting time 5
delivery_time = (6.582734) + (1.649020)*(5)
delivery_time
```

Out[52]:
```
14.827834
```

In [53]:
```
# Automatic Prediction for say sorting time 5, 8
new_data=pd.Series([5,8])
new_data
```

Out[53]:
```
0    5
1    8
dtype: int64
```

In [54]:
```
data_pred=pd.DataFrame(new_data,columns=['sorting_time'])
data_pred
```

Out[54]:

	sorting_time
0	5
1	8

In [56]:
```
linear_model.predict(data_pred)
```

Out[56]:
```
0    14.827833
1    19.774893
dtype: float64
```