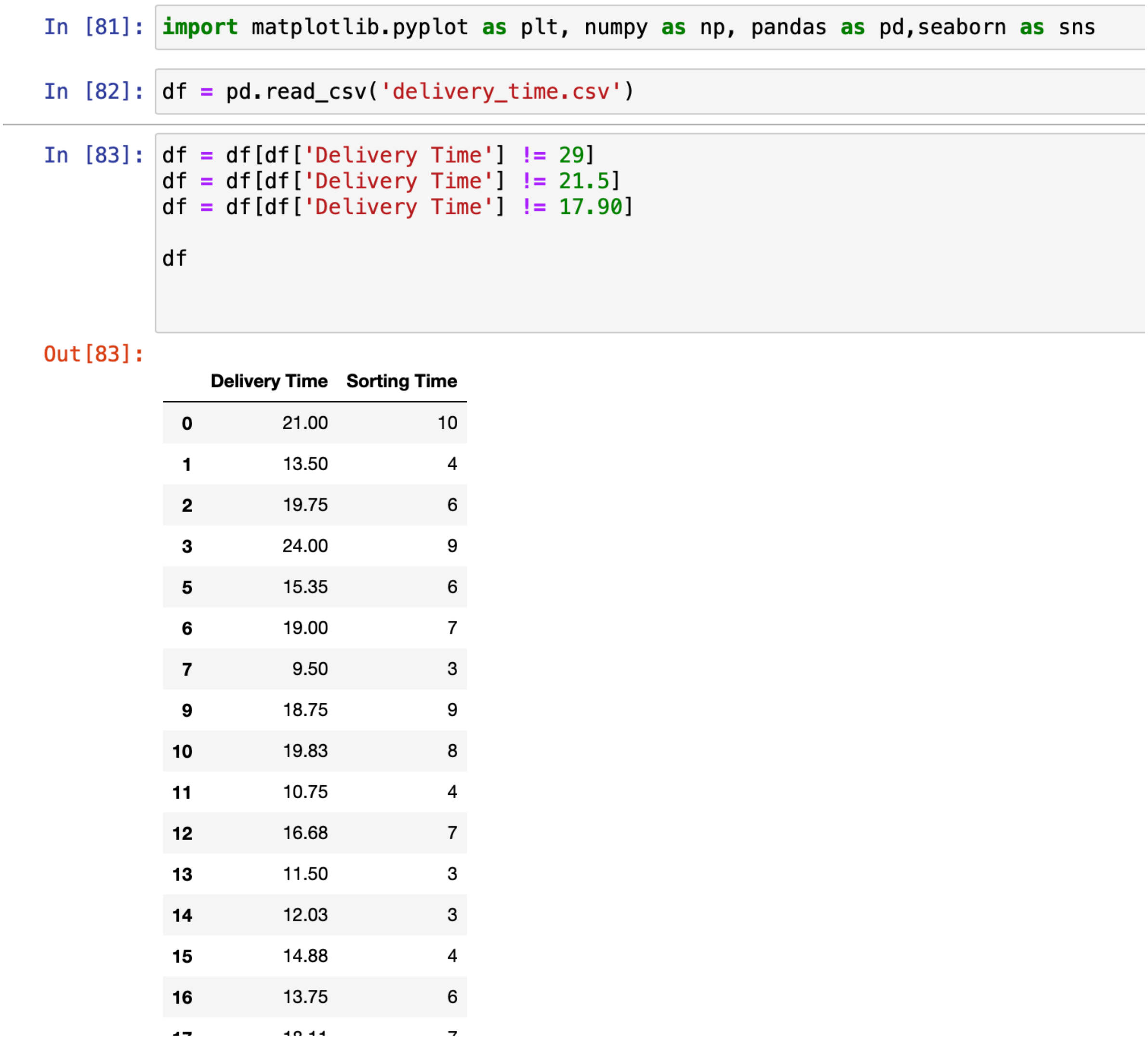
Simple Linear Regression

1) Delivery_time -> Predict delivery time using sorting time

Data Import and Cleaning:

- We start by importing the necessary Python libraries: **matplotlib.pyplot**, **numpy**, **pandas**, and **seaborn**.
- The delivery time data is read from the 'delivery_time.csv' file into a Pandas DataFrame called **df**.
- To improve data quality, we remove outliers from the 'Delivery Time' column, specifically values of 29, 21.5, and 17.90.



In [81]:
```
import matplotlib.pyplot as plt, numpy as np, pandas as pd,seaborn as sns
```

In [82]:
```
df = pd.read_csv('delivery_time.csv')
```

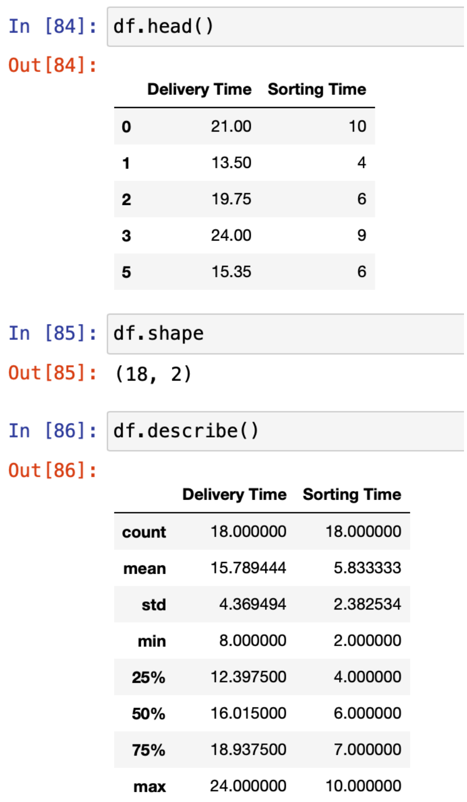
In [83]:
```
df = df[df['Delivery Time'] != 29]
df = df[df['Delivery Time'] != 21.5]
df = df[df['Delivery Time'] != 17.90]

df
```

Out[83]:

	Delivery Time	Sorting Time
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
5	15.35	6
6	19.00	7
7	9.50	3
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

Data Exploration:

- **df.head()** displays the first few rows of the dataset.
- **df.shape** provides information about the number of rows and columns in the DataFrame.
- **df.describe()** offers summary statistics for the numerical columns.
- **df.info()** reveals details about data types and potential missing values.
- **df.isnull().values** checks for any missing values within the DataFrame.



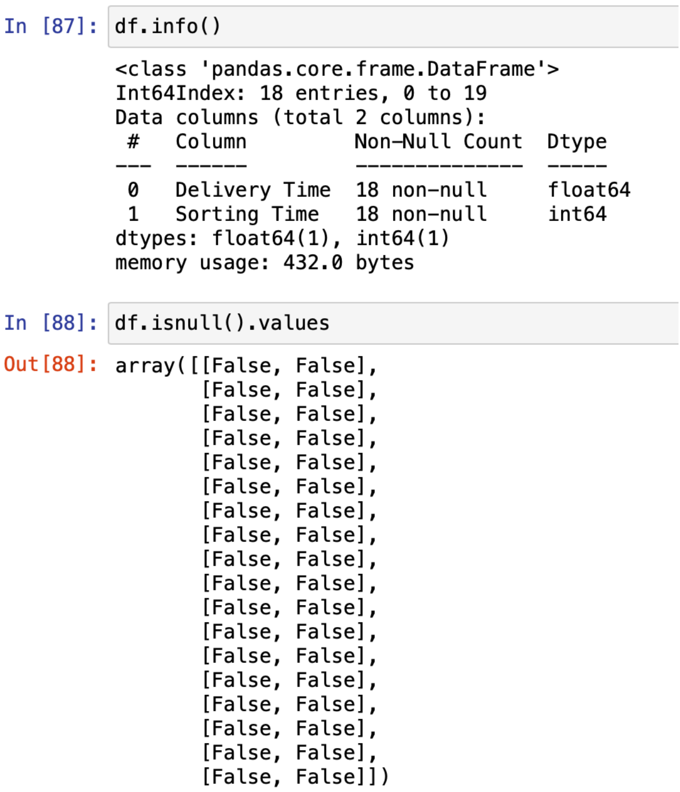
```
In [84]: df.head()
Out[84]:
```

	Delivery Time	Sorting Time
0	21.00	10
1	13.50	4
2	19.75	6
3	24.00	9
5	15.35	6

```
In [85]: df.shape
Out[85]: (18, 2)
```

```
In [86]: df.describe()
Out[86]:
```

	Delivery Time	Sorting Time
count	18.000000	18.000000
mean	15.789444	5.833333
std	4.369494	2.382534
min	8.000000	2.000000
25%	12.397500	4.000000
50%	16.015000	6.000000
75%	18.937500	7.000000
max	24.000000	10.000000



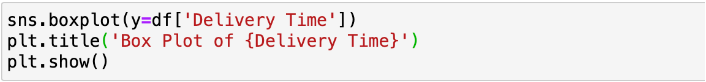
```
In [87]: df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 18 entries, 0 to 19
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Delivery Time  18 non-null     float64
 1   Sorting Time   18 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 432.0 bytes
```

```
In [88]: df.isnull().values
Out[88]: array([[False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False],
                [False, False]])
```

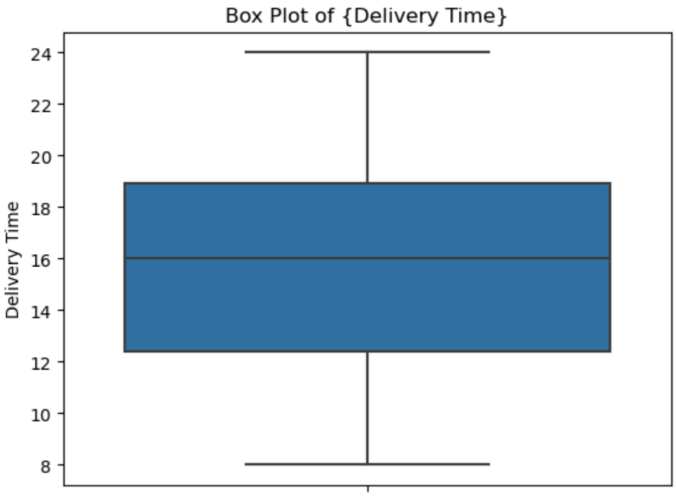
Data Visualization:

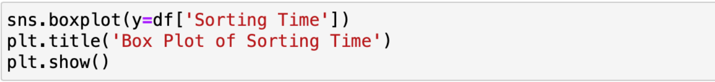
- We utilize Seaborn to create a box plot of the 'Delivery Time' column, visualizing the distribution of delivery times.

- A correlation heatmap is generated using Seaborn to visualize relationships between the numerical variables within a separate dataset called **data**.

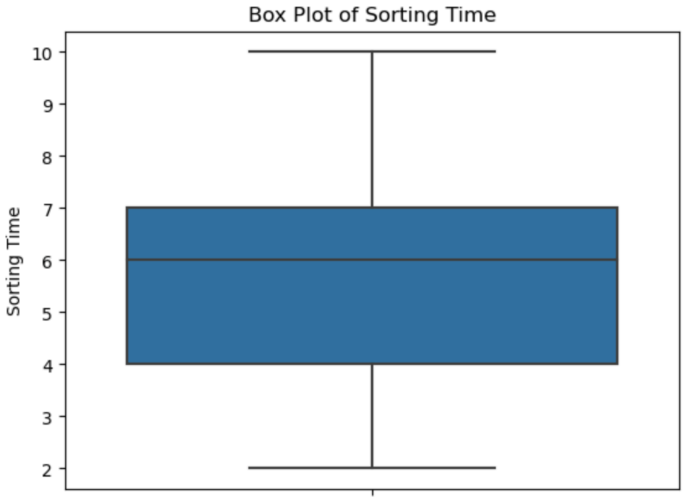


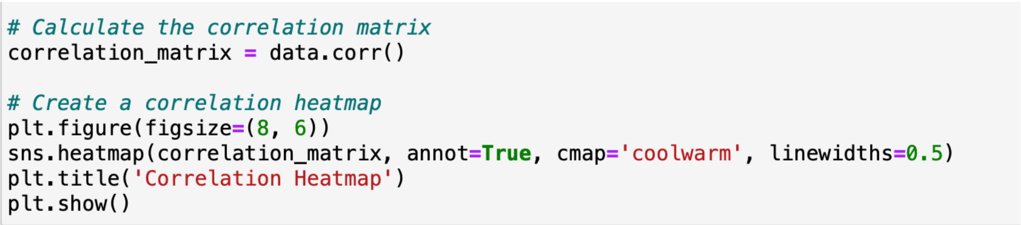
```
sns.boxplot(y=df['Delivery Time'])
plt.title('Box Plot of {Delivery Time}')
plt.show()
```



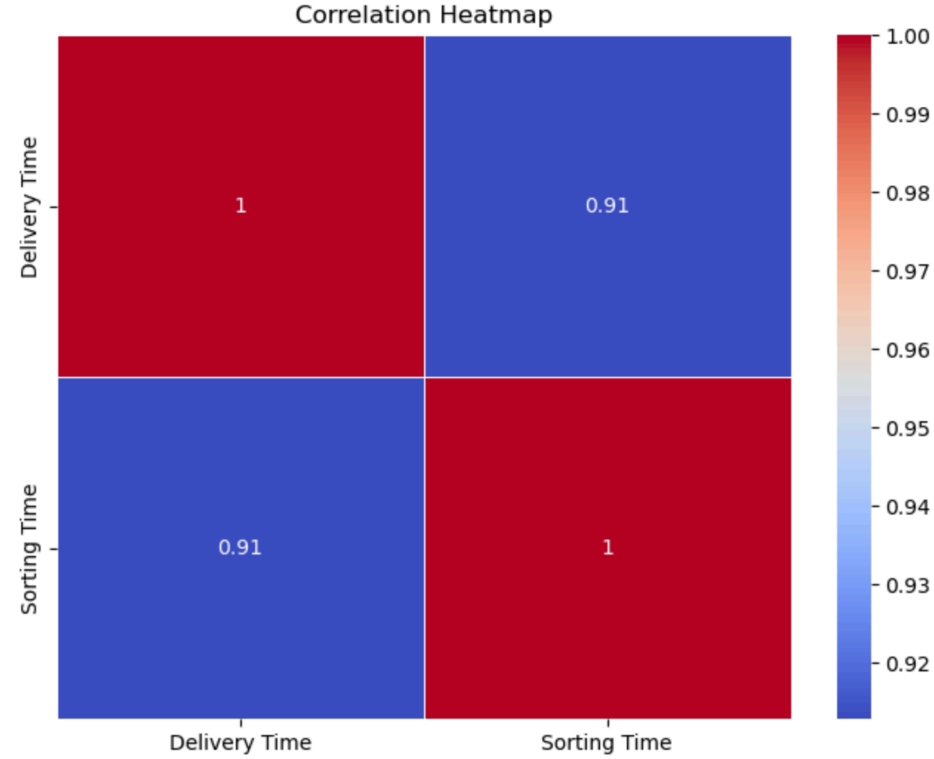


```
sns.boxplot(y=df['Sorting Time'])
plt.title('Box Plot of Sorting Time')
plt.show()
```

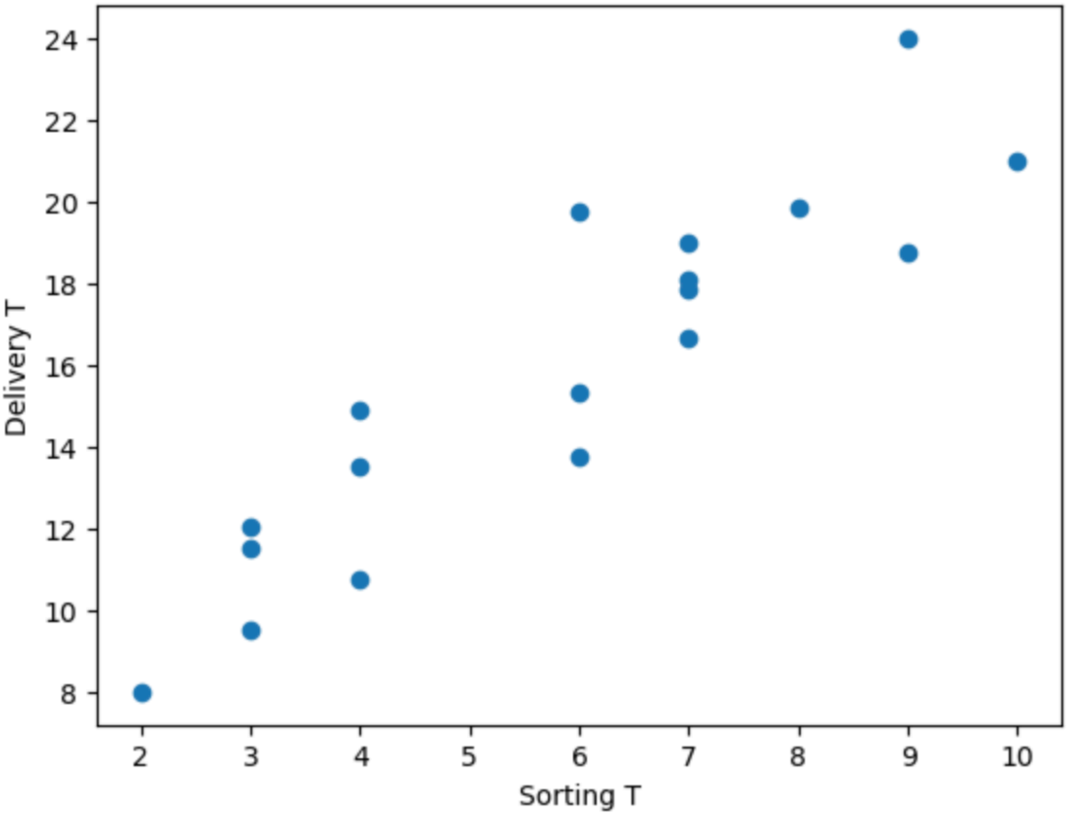




```
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Create a correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



```
In [93]: plt.scatter(x,y)
         plt.xlabel("Sorting T")
         plt.ylabel("Delivery T")

Out[93]: Text(0, 0.5, 'Delivery T')
```



Linear Regression Model:

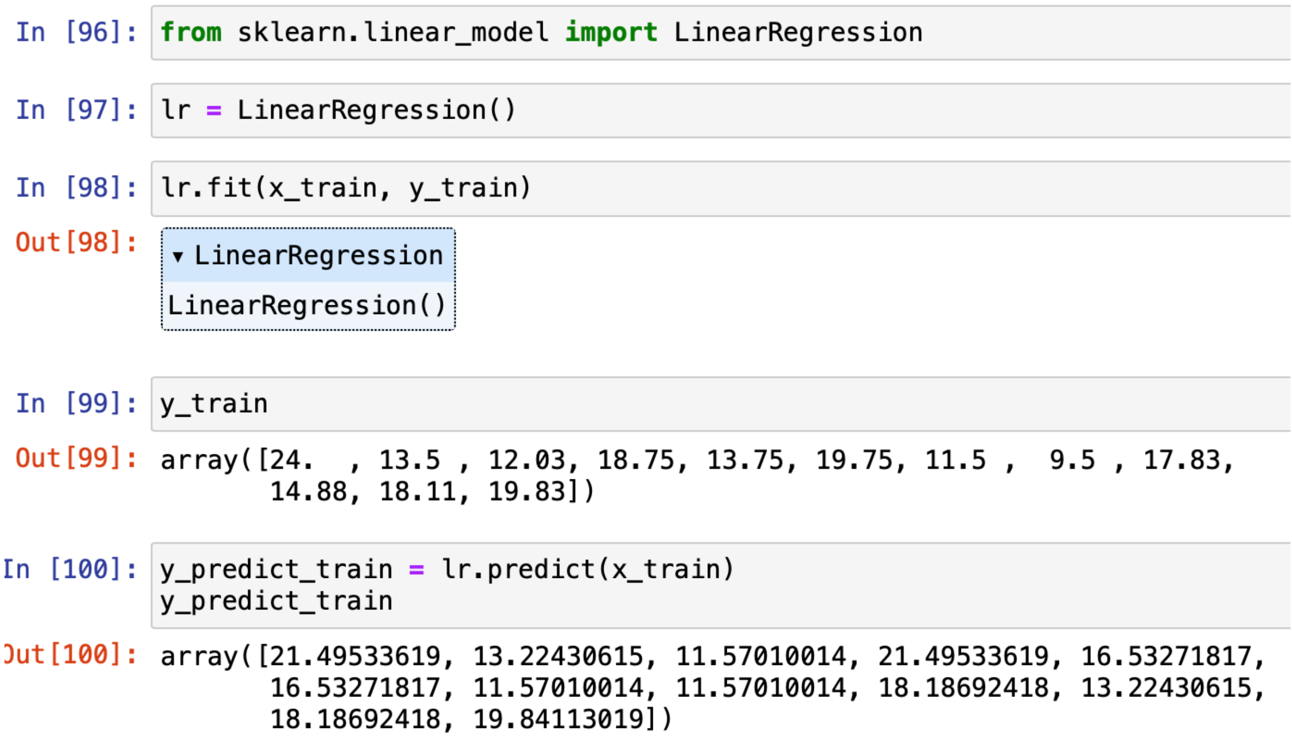
```
In [94]: from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)
```

```
In [95]: x_train = np.array(x_train).reshape(-1,1)
         x_test = np.array(x_test).reshape(-1,1)
```

- We prepare the data for building a linear regression model.
- The 'Sorting Time' column is designated as the independent variable (**x**), and the 'Delivery Time' column is designated as the dependent variable (**y**).
- Data is divided into training and testing sets using **train_test_split** from **sklearn.model_selection**.
- We reshape the independent variables in both the training and testing sets using NumPy to accommodate the model's requirements.

Linear Regression Modeling:

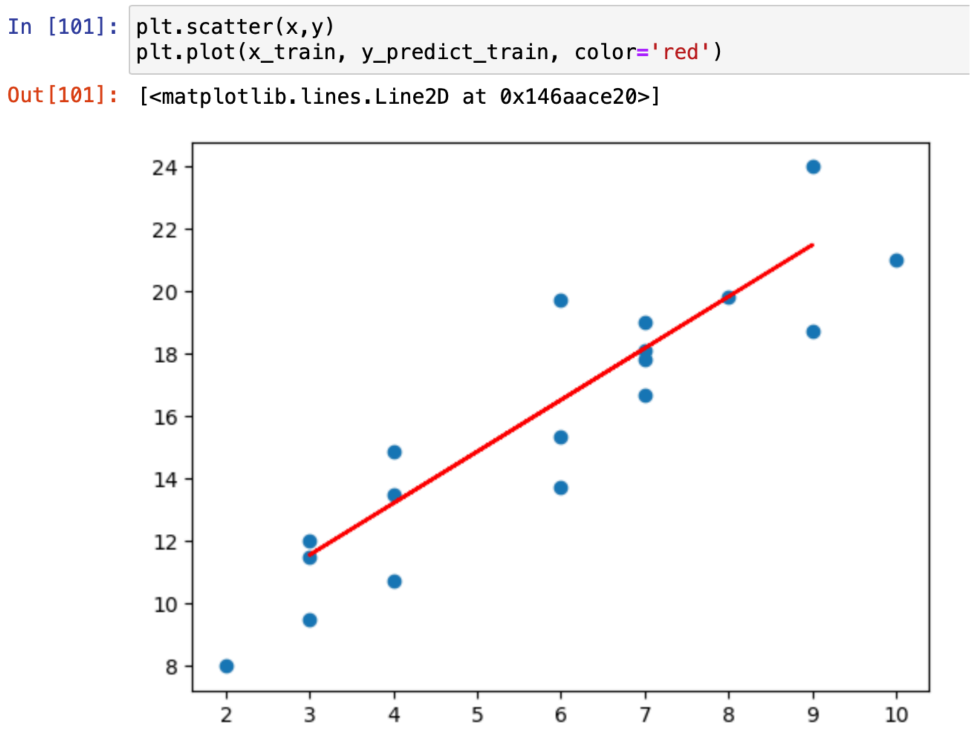
- We import the **LinearRegression** model from **sklearn.linear_model**.
- An instance of the linear regression model (**lr**) is created and fitted with the training data.



```
In [96]: from sklearn.linear_model import LinearRegression

In [97]: lr = LinearRegression()

In [98]: lr.fit(x_train, y_train)

Out[98]: ▾ LinearRegression
         LinearRegression()

In [99]: y_train

Out[99]: array([24.  , 13.5 , 12.03, 18.75, 13.75, 19.75, 11.5 ,  9.5 , 17.83,
                14.88, 18.11, 19.83])

In [100]: y_predict_train = lr.predict(x_train)
          y_predict_train

Out[100]: array([21.49533619, 13.22430615, 11.57010014, 21.49533619, 16.53271817,
                 16.53271817, 11.57010014, 11.57010014, 18.18692418, 13.22430615,
                 18.18692418, 19.84113019])
```

Model Visualization:

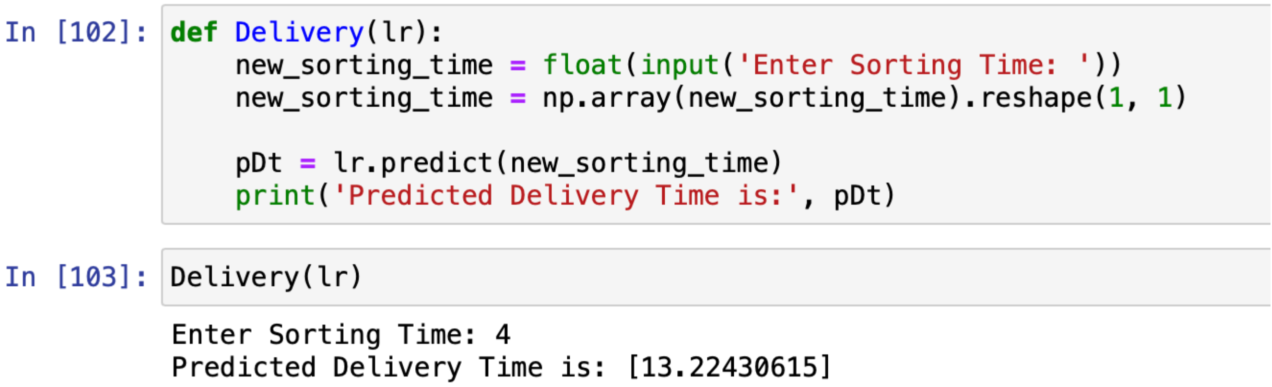
- A scatter plot is generated to visualize the relationship between 'Sorting Time' and 'Delivery Time.' Additionally, the linear regression line is overlaid on the training data for further insight.



```
In [101]: plt.scatter(x,y)
          plt.plot(x_train, y_predict_train, color='red')

Out[101]: [<matplotlib.lines.Line2D at 0x146aace20>]
```

Delivery Time Prediction:

- We define a function, **Delivery(lr)**, to make predictions for delivery times based on sorting times.
- Users are prompted to input a sorting time, and the function utilizes the trained linear regression model to predict the corresponding delivery time.
- The predicted delivery time is then displayed as 'Predicted Delivery Time.



```
In [102]: def Delivery(lr):
              new_sorting_time = float(input('Enter Sorting Time: '))
              new_sorting_time = np.array(new_sorting_time).reshape(1, 1)

              pDt = lr.predict(new_sorting_time)
              print('Predicted Delivery Time is:', pDt)

In [103]: Delivery(lr)

          Enter Sorting Time: 4
          Predicted Delivery Time is: [13.22430615]
```

Conclusion:

- The assignment is summarized by highlighting the key steps involved, including data import, cleansing, exploration, visualization, model construction, and delivery time prediction.