SALARY DATA

In [2]:

```
# import the pandas library
import pandas as pd
```

In [36]:

```
s_data=pd.read_csv("C:\\Users\\nishi\\Desktop\\Assignments\\Simple_linear_Regression\\S
```

In [37]:

```
s_data
```

Out[37]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0

	YearsExperience	Salary
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [5]:

```
s_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

In [7]:

```
s_data.corr()
```

Out[7]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

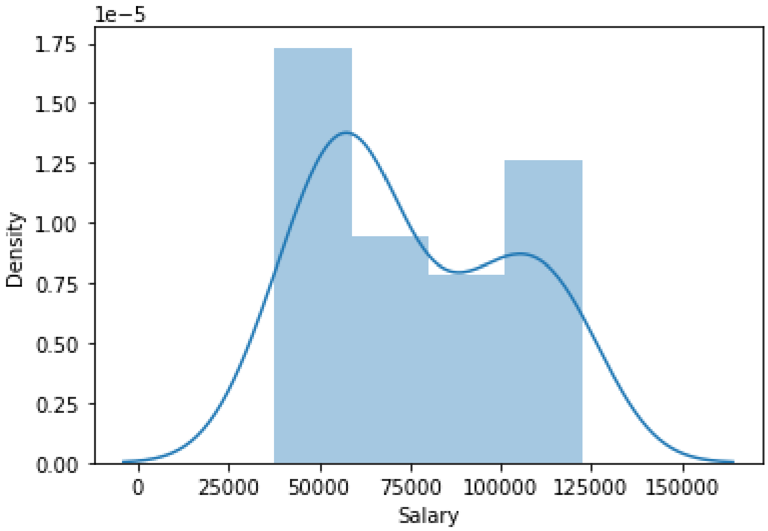
In [9]:

```
import seaborn as sns
sns.distplot(s_data["Salary"])
```

```
C:\Users\nishi\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='Salary', ylabel='Density'>
```

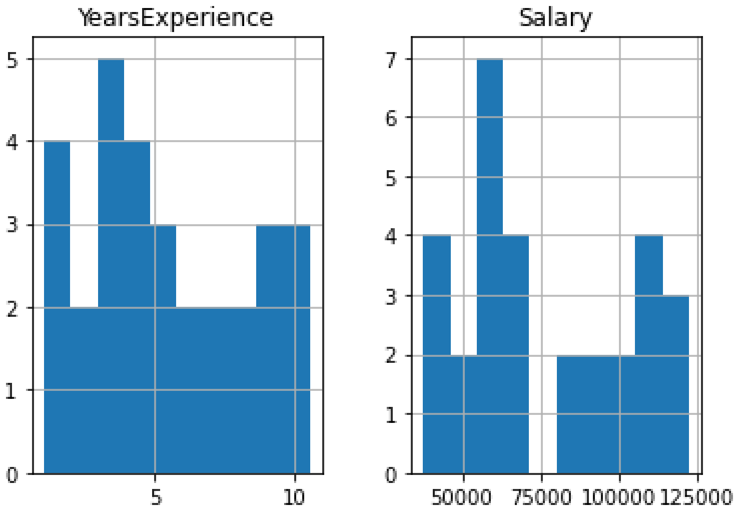


In [11]:

```
s_data.hist()
```

Out[11]:

```
array([[<AxesSubplot:title={'center':'YearsExperience'}>,
        <AxesSubplot:title={'center':'Salary'}>]], dtype=object)
```



In [38]:

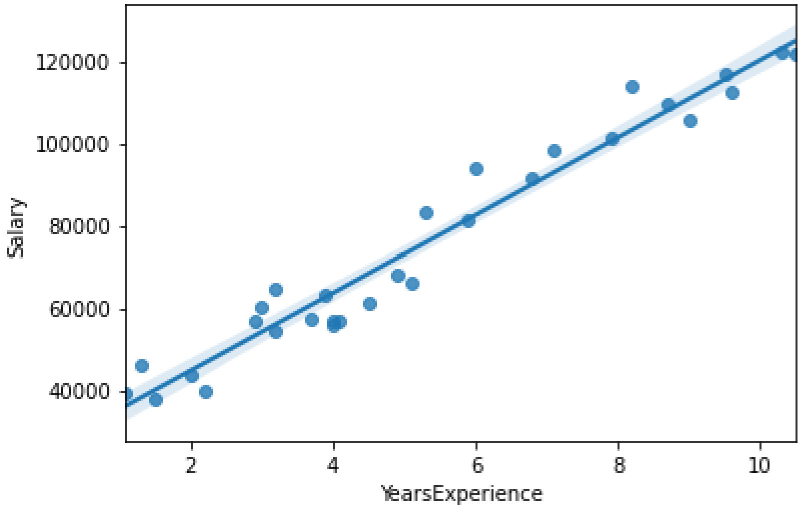
```
# ols stands for oridinary least square
import statsmodels.formula.api as sfa # this two lines are needed to perform the Linear
model=sfa.ols("Salary~YearsExperience",data=s_data).fit()
```

In [14]:

```
sns.regplot(x="YearsExperience",y="Salary",data=s_data)
```

Out[14]:

```
<AxesSubplot:xlabel='YearsExperience', ylabel='Salary'>
```



In [15]:

```
# coefficients
model.params
# irrespective of the Years of Experience there will be a min hike in the salary by 257
# with each unit increase in the Year of Experience, the Salary increases by 9449.96232
```

Out[15]:

```
Intercept          25792.200199
YearsExperience     9449.962321
dtype: float64
```

In [16]:

```
# p values for the intercept,daily,sunday
print(model.pvalues,"\n",model.tvalues)   # the p value for the model should be less th
# for the p value we need to look at the intercept value here, if p value(Intercept) is
# so the vraiable we have chosen is the correct one for the predictions.
```

```
Intercept          5.511950e-12
YearsExperience    1.143068e-20
dtype: float64
 Intercept          11.346940
YearsExperience    24.950094
dtype: float64
```

In [17]:

```
# r square values
(model.rsquared,model.rsquared_adj) # it tells us about the contribution of the data to
```

Out[17]:

```
(0.9569566641435086, 0.9554194021486339)
```

PREDICTION FOR THE NEW DATAPOINTS

In [18]:

```
new_data=pd.Series([8,9.5])
```

In [20]:

```
data_pred=pd.DataFrame(new_data,columns=['YearsExperience'])
print(model.predict(data_pred))
```

```
0    101391.898770
1    115566.842252
dtype: float64
```

In [21]:

```
# Above we have calculated the salary hike for 8 years of Experience and 9.5 years of E
```

DELIVERY TIME

In [47]:

```
# Let us import the delivery dataset
import pandas as pd
```

In [48]:

```
data1=pd.read_csv("C:\\Users\\nishi\\Desktop\\Assignments\\Simple_linear_Regression\\de
```

In [49]:

```
data1
```

Out[49]:

	DeliveryTime	SortingTime
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 [50]:

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
```

```
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   DeliveryTime  21 non-null     float64
 1   SortingTime   21 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 464.0 bytes
```

In [51]:

```
data1.corr()
```

Out[51]:

	DeliveryTime	SortingTime
DeliveryTime	1.000000	0.825997
SortingTime	0.825997	1.000000

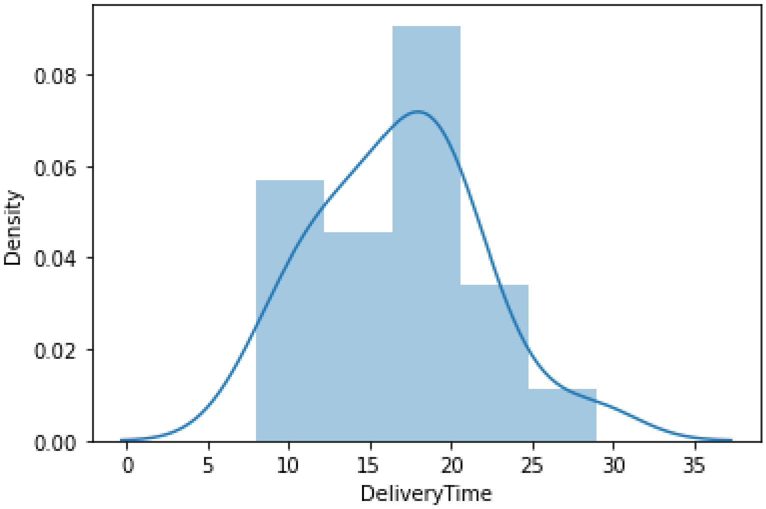
In [53]:

```
import seaborn as sns
sns.distplot(data1["DeliveryTime"])
```

```
C:\Users\nishi\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[53]:

```
<AxesSubplot:xlabel='DeliveryTime', ylabel='Density'>
```

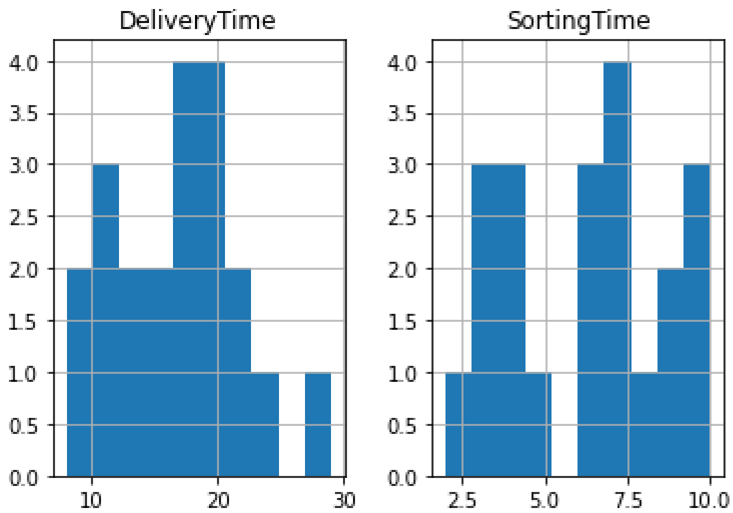


In [54]:

```
data1.hist()
```

Out[54]:

```
array([[<AxesSubplot:title={'center':'DeliveryTime'}>,
        <AxesSubplot:title={'center':'SortingTime'}>]], dtype=object)
```



In [60]:

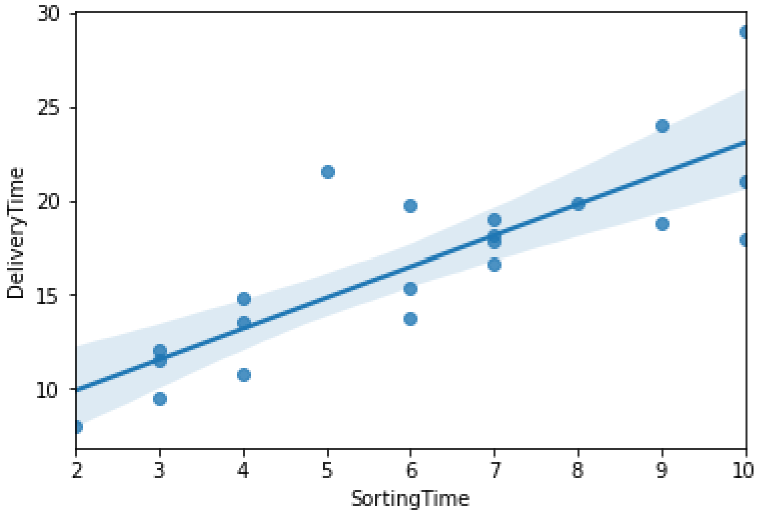
```python
# ols stands for oridinary least square
import statsmodels.formula.api as sfa # this two lines are needed to perform the Linear
model1=sfa.ols("DeliveryTime~SortingTime" ,data=data1).fit()
```

In [61]:

```python
sns.regplot(x="SortingTime",y="DeliveryTime",data=data1)
```

Out[61]:

```
<AxesSubplot:xlabel='SortingTime', ylabel='DeliveryTime'>
```



In [63]:

```python
# coefficients
model1.params
# irrespective of the SortingTime there will be a min Delivery time of 6.582734
# with each unit increase in SortingTime, the DeliveryTime increases by 1.649020
```

Out[63]:

```
Intercept      6.582734
SortingTime    1.649020
dtype: float64
```

In [64]:

```python
# p values for the intercept,DeliveryTime and SortingTime
print(model1.pvalues,"\n",model1.tvalues)   # the p value for the model should be less
# for the p value we need to look at the intercept value here, if p value(Intercept) is
# so the vraiable we have chosen is the correct one for the predictions as 0.001147<0.0
```

```
Intercept      0.001147
SortingTime    0.000004
dtype: float64
 Intercept      3.823349
SortingTime    6.387447
dtype: float64
```

In [65]:

```
# r square values
(model1.rsquared,model1.rsquared_adj) # it tells us about the contribution of the data
```

Out[65]:

```
(0.6822714748417231, 0.6655489208860244)
```

PREDICTION FOR NEW DATA POINTS

In [66]:

```
new_data1=pd.Series([11,12])
```

In [67]:

```
data_pred1=pd.DataFrame(new_data1,columns=['SortingTime'])
print(model1.predict(data_pred1))
```

```
0    24.721953
1    26.370973
dtype: float64
```