In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

Question -->1) Salary_hike -> Build a prediction model for Salary_hike

Input Variable X = YearsExperience Output Variable Y= Salary

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

Out[2]:

	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
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 [6]:
```
Salary_Data.describe()
```

Out[6]:

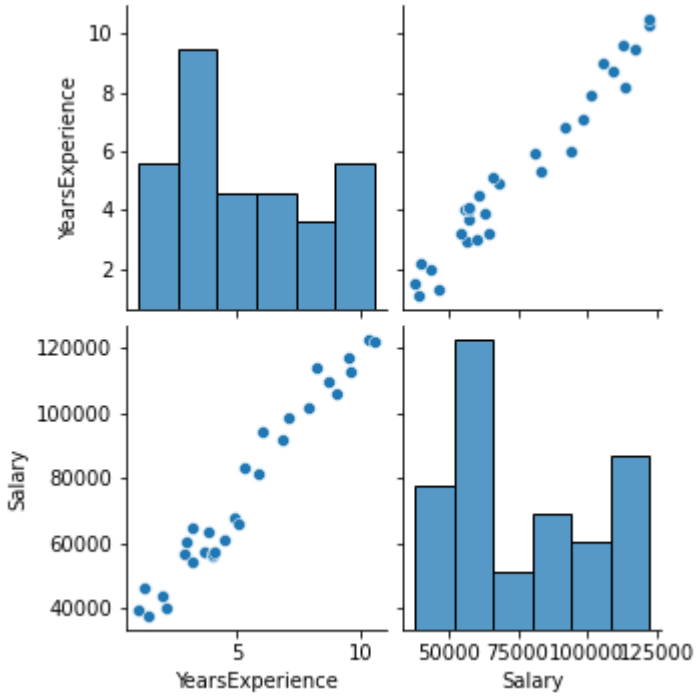
	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [7]:
```
Salary_Data.shape
```

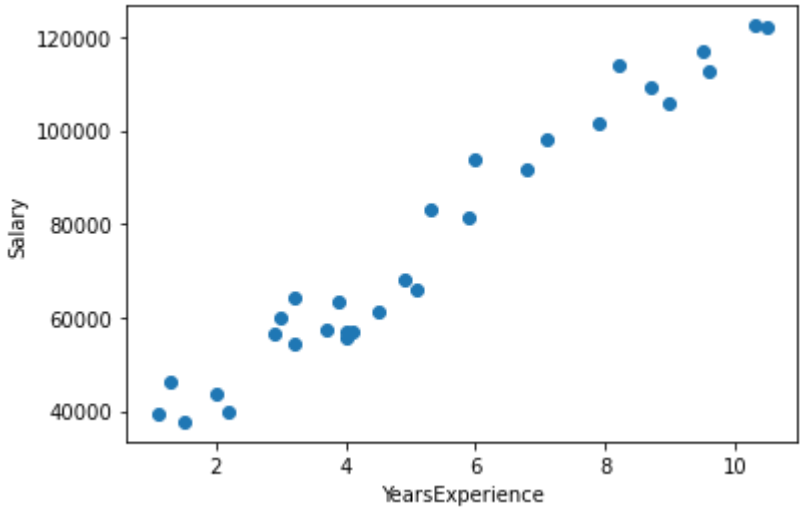
Out[7]:
```
(30, 2)
```

In [8]:
```
sns.pairplot(Salary_Data.iloc[:,0:2])
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x267e9c8e0a0>
```



In [9]:
```
plt.scatter(x = Salary_Data['YearsExperience'], y = Salary_Data['Salary'])
plt.xlabel("YearsExperience")
plt.ylabel("Salary")
plt.show()
```



Model Creation using Statsmodel

In [10]:
```
Salary_Data.YearsExperience.corr(Salary_Data.Salary)
np.corrcoef(Salary_Data.YearsExperience,Salary_Data.Salary)
```

Out[10]:
```
array([[1.        , 0.97824162],
       [0.97824162, 1.        ]])
```

In [11]:
```
linear_model=smf.ols("YearsExperience~Salary",data=Salary_Data).fit()
linear_model.params
linear_model.summary()
```

Out[11]:

OLS Regression Results

Dep. Variable:	YearsExperience	**R-squared:**	0.957
Model:	OLS	**Adj. R-squared:**	0.955
Method:	Least Squares	**F-statistic:**	622.5
Date:	Tue, 18 Jan 2022	**Prob (F-statistic):**	1.14e-20
Time:	13:47:20	**Log-Likelihood:**	-26.168
No. Observations:	30	**AIC:**	56.34
Df Residuals:	28	**BIC:**	59.14
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.3832	0.327	-7.281	0.000	-3.054	-1.713
Salary	0.0001	4.06e-06	24.950	0.000	9.3e-05	0.000

Omnibus:	3.544	**Durbin-Watson:**	1.587
Prob(Omnibus):	0.170	**Jarque-Bera (JB):**	2.094
Skew:	-0.412	**Prob(JB):**	0.351
Kurtosis:	2.003	**Cond. No.**	2.41e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.41e+05. This might indicate that there are strong multicollinearity or other numerical problems.

In [13]:
```
linear_model.conf_int(0.05)
```

Out[13]:

	0	1
Intercept	-3.053603	-1.712718
Salary	0.000093	0.000110

In [14]:
```
prediction=linear_model.predict(Salary_Data.iloc[:,1])
print(prediction)
```

```
0      1.600934
1      2.295819
2      1.437694
3      2.024427
4      1.656428
5      3.352729
6      3.707969
7      3.130248
8      4.142905
9      3.408121
10     4.018652
11     3.266856
12     3.384628
13     3.397185
14     3.805285
15     4.496626
16     4.303310
17     6.030801
18     5.856117
19     7.129735
20     6.906748
21     7.568520
22     7.875253
23     9.142087
24     8.698442
25     8.308670
26     9.461782
27     9.022897
28    10.010845
29     9.958288
dtype: float64
```

In [15]:
```
#Splitting 3/4 of the data for training and 1/4 for testing

x=np.array([Salary_Data["YearsExperience"]]).reshape(-1,1)#X value input - Years
y = np.array([Salary_Data["Salary"]]).reshape(-1,1)# Y value output - Salary Hike
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
x_train
```

Out[15]:
```
array([[ 4.1],
       [ 9.5],
       [ 6. ],
       [ 4.5],
       [ 1.3],
       [ 5.9],
       [ 3.2],
       [10.5],
       [ 9.6],
       [ 4. ],
       [ 1.5],
       [ 3. ],
       [ 6.8],
       [ 9. ],
       [ 7.9],
       [ 2.2],
       [ 4.9],
       [ 7.1],
       [ 4. ],
       [ 1.1],
       [ 2. ],
       [ 3.2]])
```

In [16]:
```
y_train
```

Out[16]:
```
array([[ 57081.],
       [116969.],
       [ 93940.],
       [ 61111.],
       [ 46205.],
       [ 81363.],
       [ 64445.],
       [121872.],
       [112635.],
       [ 56957.],
       [ 37731.],
       [ 60150.],
       [ 91738.],
       [105582.],
       [101302.],
       [ 39891.],
       [ 67938.],
       [ 98273.],
       [ 55794.],
       [ 39343.],
       [ 43525.],
       [ 54445.]])
```

In [17]:
```
model=LinearRegression().fit(x_train,y_train)
model.score(x_test,y_test)*100 # Evaluate the model
x_test
```

Out[17]:
```
array([[ 5.1],
       [ 8.2],
       [10.3],
       [ 5.3],
       [ 3.7],
       [ 3.9],
       [ 2.9],
       [ 8.7]])
```

In [18]:
```
y_test
```

Out[18]:
```
array([[ 66029.],
       [113812.],
       [122391.],
       [ 83088.],
       [ 57189.],
       [ 63218.],
       [ 56642.],
       [109431.]])
```

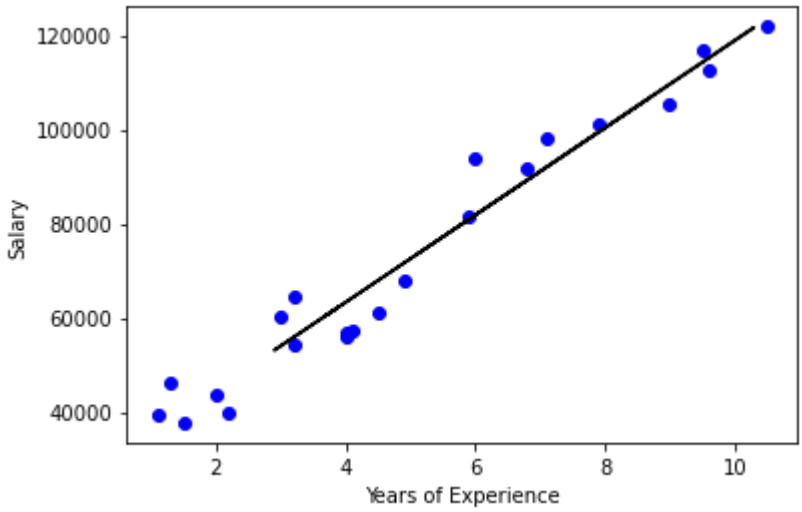
In [19]:
```
y_pred= model.predict(x_test)#Predict the model
print(y_pred)
```

```
[[ 73483.15358835]
 [102207.68106116]
 [121666.23192984]
 [ 75336.34890918]
 [ 60510.78634257]
 [ 62363.98166339]
 [ 53098.00505926]
 [106840.66936323]]
```

Visualization for the predicted model

In [20]:
```
plt.scatter(x_train,y_train,color='b')
plt.plot(x_test,y_pred,color='k')
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
```

Out[20]: Text(0, 0.5, 'Salary')



Question --> 2) Delivery_time -> Predict delivery time using sorting time

In [48]:
```
Delivery_Time_Df = pd.read_csv('delivery_time.csv')
Delivery_Time_Df
```

Out[48]:

	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 [49]:
```
Delivery_Time_Df.shape
```

Out[49]: (21, 2)

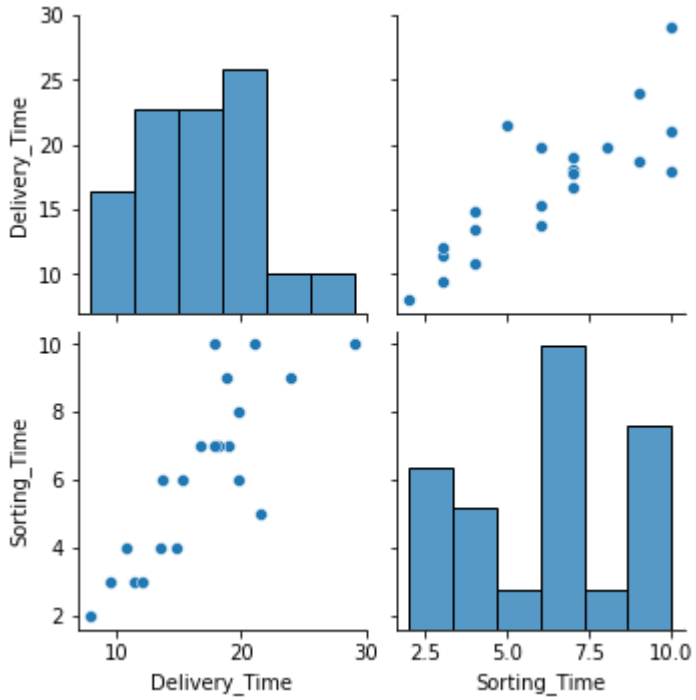
In [50]:
```
Delivery_Time_Df.describe()
```

Out[50]:

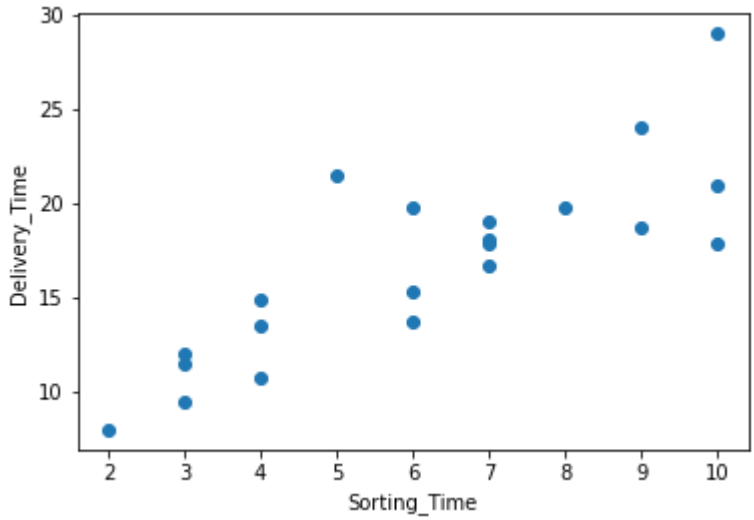
	Delivery_Time	Sorting_Time
count	21.000000	21.000000
mean	16.790952	6.190476
std	5.074901	2.542028
min	8.000000	2.000000
25%	13.500000	4.000000
50%	17.830000	6.000000
75%	19.750000	8.000000
max	29.000000	10.000000

In [51]:
```
sns.pairplot(Delivery_Time_Df.iloc[:,0:2])
```

Out[51]:
```
<seaborn.axisgrid.PairGrid at 0x267ea615eb0>
```



In [52]:
```
plt.scatter(x = Delivery_Time_Df['Sorting_Time'], y = Delivery_Time_Df['Delivery_
plt.xlabel("Sorting_Time")
plt.ylabel("Delivery_Time")
plt.show()
```



In [53]:
```
Delivery_Time_Df.Sorting_Time.corr(Delivery_Time_Df.Delivery_Time)
np.corrcoef(Delivery_Time_Df.Sorting_Time,Delivery_Time_Df.Delivery_Time)
```

Out[53]:
```
array([[1.        , 0.82599726],
       [0.82599726, 1.        ]])
```

In [54]:
```
linear_model=smf.ols("Sorting_Time~Delivery_Time",data=Delivery_Time_Df).fit()
linear_model.params
linear_model.summary()
```

Out[54]:

OLS Regression Results

Dep. Variable:	Sorting_Time	**R-squared:**	0.682
Model:	OLS	**Adj. R-squared:**	0.666
Method:	Least Squares	**F-statistic:**	40.80
Date:	Tue, 18 Jan 2022	**Prob (F-statistic):**	3.98e-06
Time:	14:01:41	**Log-Likelihood:**	-36.839
No. Observations:	21	**AIC:**	77.68
Df Residuals:	19	**BIC:**	79.77
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.7567	1.134	-0.667	0.513	-3.130	1.617
Delivery_Time	0.4137	0.065	6.387	0.000	0.278	0.549

Omnibus:	1.409	**Durbin-Watson:**	1.346
Prob(Omnibus):	0.494	**Jarque-Bera (JB):**	0.371
Skew:	0.255	**Prob(JB):**	0.831
Kurtosis:	3.405	**Cond. No.**	62.1

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

In [55]:
```
linear_model.conf_int(0.05)
```

Out[55]:

	0	1
Intercept	-3.130058	1.616712
Delivery_Time	0.278169	0.549318

In [56]:
```
prediction =linear_model.predict(Delivery_Time_Df.iloc[:,0])
print(prediction)
```

```
0      7.931943
1      4.828866
2      7.414763
3      9.173174
4     11.241892
5      5.594291
6      7.104456
7      3.173891
8      6.649338
9      7.001020
10     7.447863
11     3.691071
12     6.144570
13     4.001378
14     4.220662
15     5.399832
16     4.932302
17     6.736224
18     2.553276
19     6.620376
20     8.138815
dtype: float64
```

In [57]:
```
x=np.array([Delivery_Time_Df["Sorting_Time"]]).reshape(-1,1)
y=np.array([Delivery_Time_Df["Delivery_Time"]]).reshape(-1,1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)

x_train
```

Out[57]:
```
array([[ 7],
       [ 7],
       [ 3],
       [ 4],
       [ 5],
       [ 3],
       [ 4],
       [ 6],
       [10],
       [ 8],
       [ 4],
       [ 6],
       [10],
       [ 9],
       [ 6]], dtype=int64)
```

In [58]:
```
y_train
```

Out[58]:
```
array([[16.68],
       [18.11],
       [12.03],
       [10.75],
       [21.5 ],
       [11.5 ],
       [14.88],
       [19.75],
       [21.  ],
       [19.83],
       [13.5 ],
       [15.35],
       [29.  ],
       [24.  ],
       [13.75]])
```

In [59]:
```
model=LinearRegression().fit(x_train,y_train)
model.score(x_test,y_test)*100
```

Out[59]:
```
45.42300769005917
```

In [60]:
```
print('--------X Test---------- \n',x_test)
print('\n--------Y Test ---------\n',y_test)
```

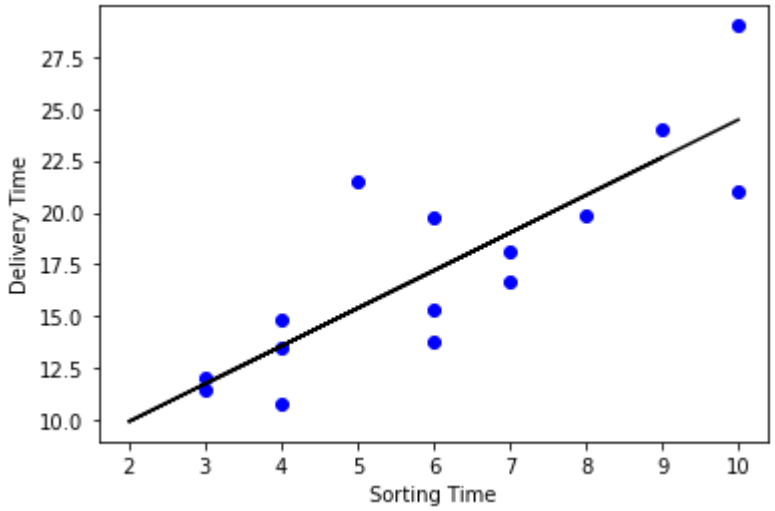
```
--------X Test----------
 [[10]
 [ 2]
 [ 9]
 [ 7]
 [ 3]
 [ 7]]

--------Y Test ---------
 [[17.9 ]
 [ 8.  ]
 [18.75]
 [19.  ]
 [ 9.5 ]
 [17.83]]
```

In [61]:
```
y_pred=model.predict(x_test)
print(y_pred)
```

```
[[24.46646655]
 [ 9.93308748]
 [22.64979417]
 [19.0164494 ]
 [11.74975986]
 [19.0164494 ]]
```

In [62]:
```
plt.scatter(x_train,y_train,color='b')
plt.plot(x_test,y_pred,color='k')
plt.xlabel("Sorting Time")
plt.ylabel("Delivery Time")
plt.show
```

Out[62]: <function matplotlib.pyplot.show(close=None, block=None)>



In [ ]: