Q1. Delivery_time -> Predict delivery time using sorting time

In [20]:
```
# importing necessary libraries

import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [21]:
```
# importing Dataset
delivery_data=pd.read_csv('delivery_time.csv')
delivery_data
```

Out[21]:

	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 [22]:
```
# Data understanding
delivery_data.dtypes
```

Out[22]:
```
Delivery_Time    float64
Sorting_Time       int64
dtype: object
```

In [23]:
```
delivery_data.shape
```

Out[23]:
```
(21, 2)
```

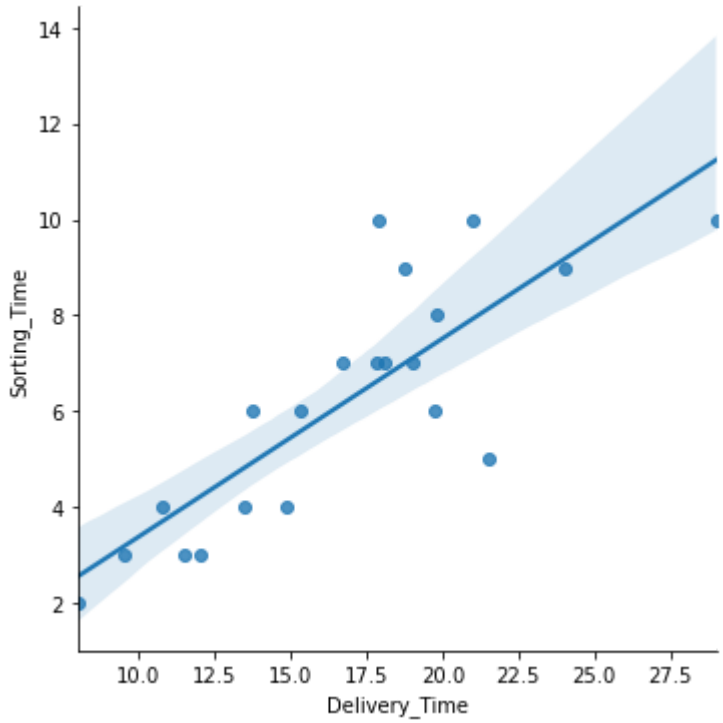
In [24]:
```
delivery_data.describe()
```

Out[24]:

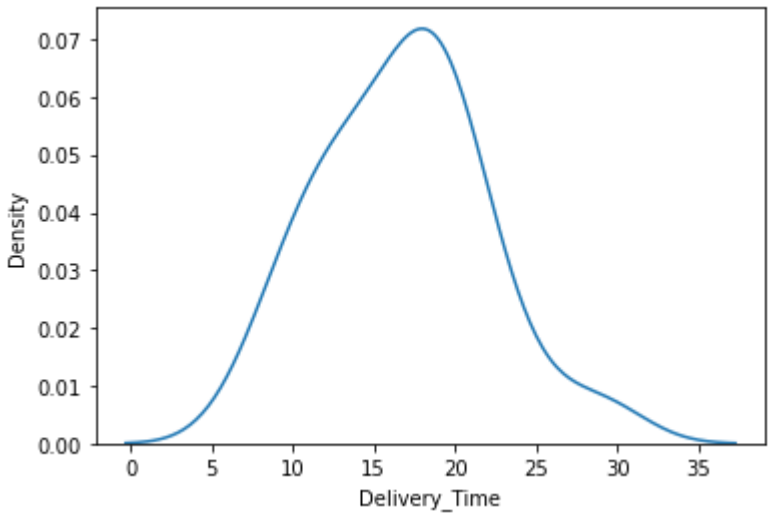
	Delivery_Time	Sorting_Time
count	21.000000	21.000000
mean	16.790952	6.190476
std	5.074901	2.542028
min	8.000000	2.000000
25%	13.500000	4.000000
50%	17.830000	6.000000
75%	19.750000	8.000000
max	29.000000	10.000000

In [26]:
```
# Linearity check
sns.scatterplot(x='Delivery_Time' ,y='Sorting_Time' ,data=delivery_data)
plt.title('Delivery Vs Sorting Time')
plt.show()
```

In [27]:
```
sns.lmplot(x='Delivery_Time' ,y='Sorting_Time' ,data=delivery_data)
plt.show()
```



In [28]:
```
# Normality check
sns.distplot(a=delivery_data['Delivery_Time'],hist=False)
plt.show()
```



In [29]:
```
delivery_data['Delivery_Time'].skew() #skewness <1 is acceptable
```

Out[29]: 0.3523900822831107

In [30]:
```
# Data Preparation

delivery_data.isna().sum()
```

Out[30]:
```
Delivery_Time    0
Sorting_Time     0
dtype: int64
```

LINEAR REGRESSION using Stats model

In [18]:
```
import statsmodels.formula.api as smf
```

In [32]:
```
linear_model = smf.ols(formula='Delivery_Time~Sorting_Time',data=delivery_da
linear_model.params
```

Out[32]:
```
Intercept       6.582734
Sorting_Time    1.649020
dtype: float64
```

In [ ]:
```

```

Q2. Salary_hike -> Build a prediction model for Salary_hike

In [33]:
```
# importing the dataset

sal_data = pd.read_csv('Salary_Data.csv')
sal_data
```

Out[33]:

	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

In [37]:
```
# Understanding the data
sal_data.info()
```

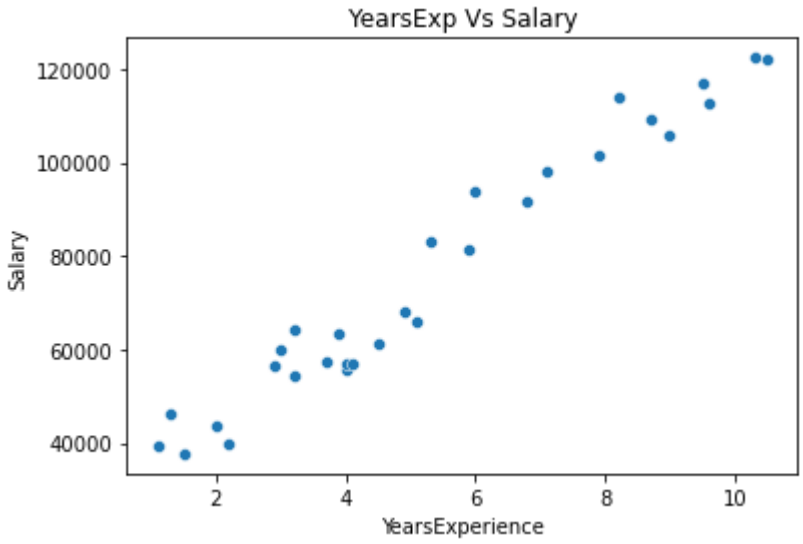
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)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

In [38]:
```
sal_data.describe()
```

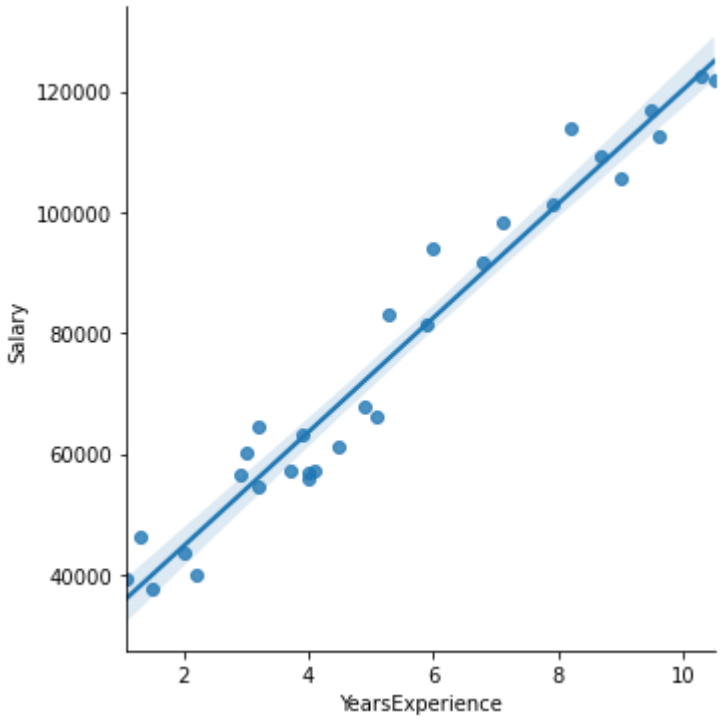
Out[38]:

	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

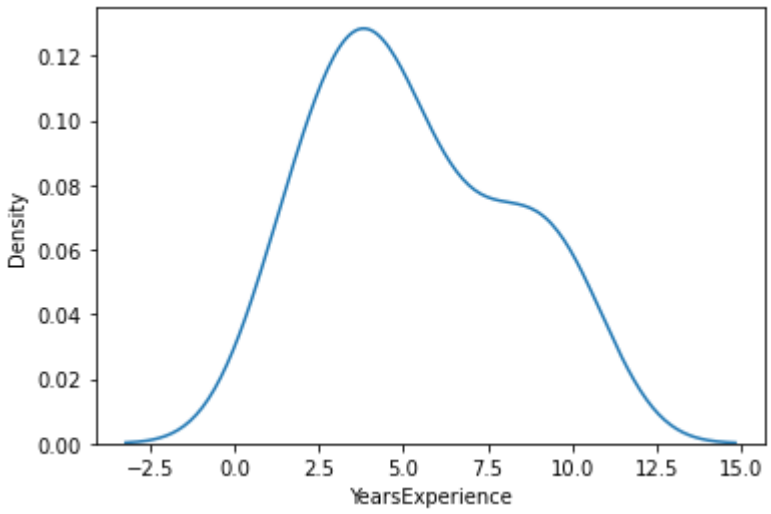
In [40]:
```
# Data Understanding
# Linearity check
sns.scatterplot(x='YearsExperience' ,y='Salary' ,data=sal_data)
plt.title('YearsExp Vs Salary')
plt.show()
```



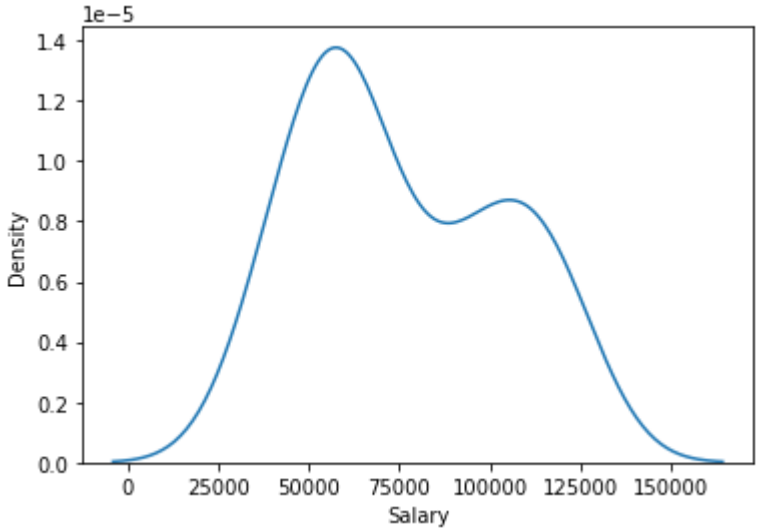
In [41]:
```
sns.lmplot(x='YearsExperience' ,y='Salary' ,data=sal_data)
plt.show()
```



In [42]:
```
# Normality check
sns.distplot(a=sal_data['YearsExperience'],hist=False)
plt.show()
```



In [43]:
```
sns.distplot(a=sal_data['Salary'],hist=False)
plt.show()
```



In [44]:
```
sal_data['Salary'].skew()
```

Out[44]:
```
0.35411967922959153
```

Linear Regression using stats model

In [45]:
```
linear_model_1 = smf.ols(formula='YearsExperience~Salary',data=sal_data).fit
linear_model_1.params
```

Out[45]:
```
Intercept   -2.383161
Salary       0.000101
dtype: float64
```

In [ ]:
```

```