1. Importing necessasary libraries

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

2. Importing data

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

3. Data Understanding

In [3]:
```
salary_data.shape
```

Out[3]:
```
(30, 2)
```

In [4]:
```
salary_data.isna().sum()
```

Out[4]:
```
YearsExperience    0
Salary             0
dtype: int64
```

In [5]:
```
salary_data.dtypes
```

Out[5]:
```
YearsExperience    float64
Salary             float64
dtype: object
```

In [6]:
```
salary_data.describe(include='all')
```

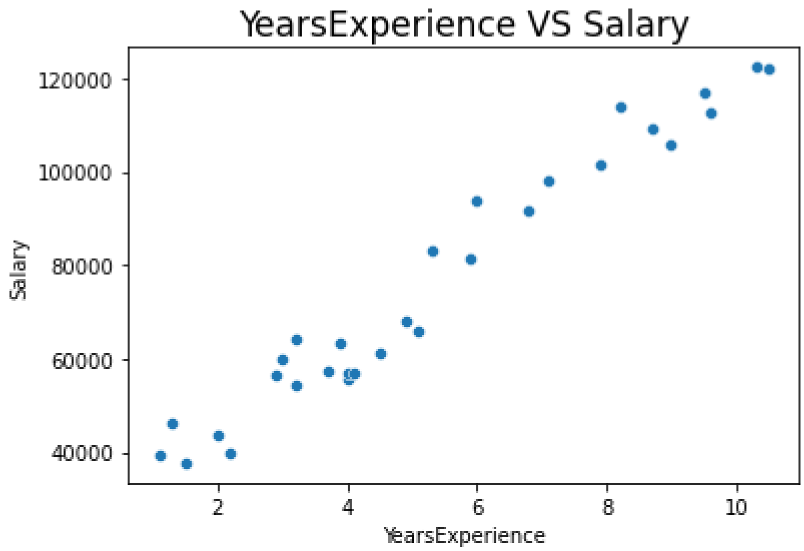
Out[6]:

	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

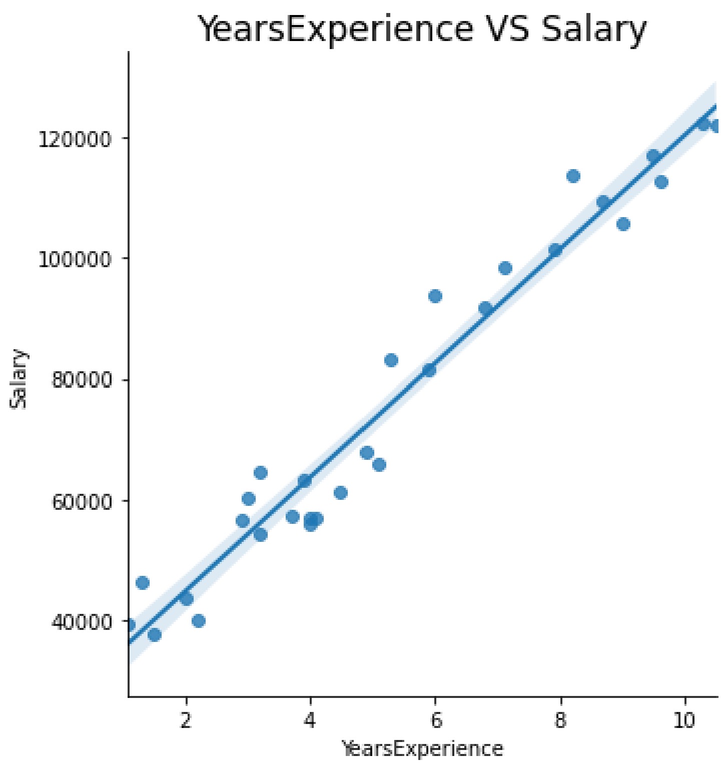
4.Check Assumptions are matching

1. Check for linearity.

In [7]:
```
sns.scatterplot(x = 'YearsExperience',y= 'Salary',data=salary_data)
plt.title('YearsExperience VS Salary',size = 17)
plt.show()
```



In [8]:
```
sns.lmplot(x = 'YearsExperience',y= 'Salary',data=salary_data)
plt.title('YearsExperience VS Salary',size = 17)
plt.show()
```



Observation = Linearity check is passed

In [9]:
```
salary_data.corr().round(2)
```

Out[9]:

	YearsExperience	Salary
YearsExperience	1.00	0.98
Salary	0.98	1.00

2.Homoscadasticity-it can be checked post model building and training

3.NO multicollinarity

4. No Autoregression

5. zero residual mean-it can be checked post model building and training

5.Data preparation

In [10]:
```
salary_data  #no unwanted parameters.
```

Out[10]:

	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

6.Model Building and Model training

There are basically 2 libraries that supports Linear Regression algorithm.

- 1.Statsmodels libraries-ols techniques.
- 2.sklearn libraries-linear regression.

In [11]:
```
import statsmodels.formula.api as smf
```

In [21]:
```
lin_reg_model = smf.ols(formula='Salary~YearsExperience', data= salary_data).fit(
```

note- with stats models model building and training is happening at the same tim

7.Model Testing

In [22]:
```
lin_reg_model.params
```

Out[22]:
```
Intercept          25792.200199
YearsExperience     9449.962321
dtype: float64
```

Manual Testing

In [15]:
```
# y = mx + c
#for x = 1.3 y=?
```

In [23]:
```
9449.962321 *1.1 +25792.200199
```

Out[23]:
```
36187.158752100004
```

In [25]:
```
9449.962321 *1.3 +25792.200199
```

Out[25]:
```
38077.1512163
```

In [26]:
```
9449.962321 *1.5 +25792.200199
```

Out[26]:
```
39967.1436805
```

In [27]:
```
9449.962321 *2 +25792.200199
```

Out[27]:
```
44692.124841
```

In [28]:
```
9449.962321 *2.2 +25792.200199
```

Out[28]:
```
46582.1173052
```

Machine Prediction

In [30]:
```
test_data = pd.read_csv('salary_test_data.csv')
test_data
```

Out[30]:

	YearsExperience
0	1.1
1	1.3
2	1.5
3	2.0
4	2.2

In [31]:
```
lin_reg_model.predict(test_data)
```

Out[31]:
```
0    36187.158752
1    38077.151217
2    39967.143681
3    44692.124842
4    46582.117306
dtype: float64
```

8. Model Evaluation

In [32]:
```
lin_reg_model.rsquared,lin_reg_model.rsquared_adj
```

Out[32]:
```
(0.9569566641435086, 0.9554194021486339)
```

In [33]:
```
lin_reg_model.aic,lin_reg_model.bic ##is an estimator of prediction error
```

Out[33]:
```
(606.882316930432, 609.6847116937563)
```

9. Model Deployement

In [34]:
```
from pickle import dump
```

In [35]:
```
dump(lin_reg_model,open('linear_regression.pkl','wb'))
```

In [36]:
```
from pickle import load
```

In [37]:
```
tested_linear_model = load(open('linear_regression.pkl','rb'))
```

In [38]:
```
tested_linear_model.predict(test_data)
```

Out[38]:
```
0    36187.158752
1    38077.151217
2    39967.143681
3    44692.124842
4    46582.117306
dtype: float64
```

Assumtion check

- Homoscedaticity

In [39]:
```
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
scaled_x = std_scaler.fit_transform(salary_data)
print(scaled_x)
```

```
[[-1.51005294 -1.36011263]
 [-1.43837321 -1.10552744]
 [-1.36669348 -1.419919  ]
 [-1.18749416 -1.20495739]
 [-1.11581443 -1.33978143]
 [-0.86493538 -0.71830716]
 [-0.82909552 -0.58815781]
 [-0.75741579 -0.79981746]
 [-0.75741579 -0.42881019]
 [-0.57821647 -0.69801306]
 [-0.50653674 -0.47433279]
 [-0.47069688 -0.74976858]
 [-0.47069688 -0.70662043]
 [-0.43485702 -0.70201994]
 [-0.29149756 -0.55250402]
 [-0.1481381  -0.29921736]
 [-0.07645838 -0.37004264]
 [-0.00477865  0.26285865]
 [ 0.21026054  0.19885989]
 [ 0.2461004   0.66547573]
 [ 0.53281931  0.58377993]
 [ 0.6403389   0.82623317]
 [ 0.92705781  0.93861127]
 [ 1.03457741  1.40274136]
 [ 1.21377673  1.24020308]
 [ 1.32129632  1.09740238]
 [ 1.50049564  1.51986835]
 [ 1.5363355   1.3590738 ]
 [ 1.78721455  1.72102849]
 [ 1.85889428  1.70177321]]
```

In [40]:
```
scaled_x.mean(),scaled_x.std()
```

Out[40]:
```
(-2.220446049250313e-17, 1.0)
```

Zero residual mean

In [44]:
```
np.mean(scaled_x)
```

Out[44]:
```
-2.2204460492503132e-17
```