In [5]:
```python
#  Expected Salary for fresher based on company employers salary

# The scenario is you are a HR officer, you got a candidate with 5 years of experience.
# Then what is the best salary you should offer to him?"
```

In [8]:
```python
# Importing Packages/Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
```

In [9]:
```python
data=pd.read_csv("E:/Github/SalaryPredLinerReg.csv")
data
```

Out[9]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

I. Getting Overview of Dataset

1. getting Datatypes of Columns

In [10]:
```python
data.dtypes
```

Out[10]:
```
YearsExperience    float64
Salary               int64
dtype: object
```

2. getting size of Dataframes(rows x Columns)

In [11]:
```python
print(data.size)
```

```
60
```

3. getting the number of rows and columns in the dataframe

In [12]:
```python
data.shape
```

Out[12]:
```
(30, 2)
```

4. getting the dimentions of the dataframe

In [13]:
```python
data.ndim
```

Out[13]:
```
2
```

5. getting Summery of Dataset

In [14]:
```python
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     int64
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

6. Informaton of Data set

In [15]:
```python
data.info
```

Out[15]:
```
<bound method DataFrame.info of     YearsExperience  Salary
0               1.1   39343
1               1.3   46205
2               1.5   37731
3               2.0   43525
4               2.2   39891
5               2.9   56642
6               3.0   60150
7               3.2   54445
8               3.2   64445
9               3.7   57189
10              3.9   63218
11              4.0   55794
12              4.0   56957
13              4.1   57081
14              4.5   61111
15              4.9   67938
16              5.1   66029
17              5.3   83088
18              5.9   81363
19              6.0   93940
20              6.8   91738
21              7.1   98273
22              7.9  101302
23              8.2  113812
24              8.7  109431
25              9.0  105582
26              9.5  116969
27              9.6  112635
28             10.3  122391
29             10.5  121872>
```

7.head() Return the first 5 rows of the DataFrame by default.

In [16]:
```python
data.head()
```

Out[16]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

8. head() Return the first n rows of the DataFrame.

In [18]:
```python
data.head(10)
```

Out[18]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189

9. head() Returning entire size of the DataFrame.

In [19]:
```python
data.head(np.size(data))
```

Out[19]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

10.tail() Return the last 5 rows of the DataFrame by default.

In [20]:
```python
data.tail()
```

Out[20]:

	YearsExperience	Salary
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

12. tail() Return the last 10 rows of the DataFrame.

In [21]:
```python
data.tail(10)
```

Out[21]:

	YearsExperience	Salary
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

13. tail() Returning entire rows of the DataFrame.

In [22]:
```python
data.tail(np.size(data))
```

Out[22]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

In [ ]:

In [ ]:

II. Data Pre-processing

1. Checking NULL Values and count

In [23]:
```python
print(data.isna())
```

```
    YearsExperience  Salary
0             False   False
1             False   False
2             False   False
3             False   False
4             False   False
5             False   False
6             False   False
7             False   False
8             False   False
9             False   False
10            False   False
11            False   False
12            False   False
13            False   False
14            False   False
15            False   False
16            False   False
17            False   False
18            False   False
19            False   False
20            False   False
21            False   False
22            False   False
23            False   False
24            False   False
25            False   False
26            False   False
27            False   False
28            False   False
29            False   False
```

In [30]:
```python
null_values=data.isna().sum()
print(null_values)
```

```
YearsExperience    0
Salary             0
dtype: int64
```

2. Categorical to Numerical Values

In [31]:
```python
print(data.isna().astype(int))
```

```
    YearsExperience  Salary
0                 0       0
1                 0       0
2                 0       0
3                 0       0
4                 0       0
5                 0       0
6                 0       0
7                 0       0
8                 0       0
9                 0       0
10                0       0
11                0       0
12                0       0
13                0       0
14                0       0
15                0       0
16                0       0
17                0       0
18                0       0
19                0       0
20                0       0
21                0       0
22                0       0
23                0       0
24                0       0
25                0       0
26                0       0
27                0       0
28                0       0
29                0       0
```

3. Reading Particular Column data

In [33]:
```python
data['YearsExperience']
```

Out[33]:
```
0      1.1
1      1.3
2      1.5
3      2.0
4      2.2
5      2.9
6      3.0
7      3.2
8      3.2
9      3.7
10     3.9
11     4.0
12     4.0
13     4.1
14     4.5
15     4.9
16     5.1
17     5.3
18     5.9
19     6.0
20     6.8
21     7.1
22     7.9
23     8.2
24     8.7
25     9.0
26     9.5
27     9.6
28    10.3
29    10.5
Name: YearsExperience, dtype: float64
```

4. Converting Particular Column data to int

In [37]:
```python
data['YearsExperience'].astype(int)
```

Out[37]:
```
0      1
1      1
2      1
3      2
4      2
5      2
6      3
7      3
8      3
9      3
10     3
11     4
12     4
13     4
14     4
15     4
16     5
17     5
18     5
19     6
20     6
21     7
22     7
23     8
24     8
25     9
26     9
27     9
28    10
29    10
Name: YearsExperience, dtype: int32
```

5. Checking Particular Column data NULL Values

In [38]:
```python
data['YearsExperience'].isna().astype(int)
```

Out[38]:
```
0     0
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
13    0
14    0
15    0
16    0
17    0
18    0
19    0
20    0
21    0
22    0
23    0
24    0
25    0
26    0
27    0
28    0
29    0
Name: YearsExperience, dtype: int32
```

In [39]:
```python
data['YearsExperience'].isna().astype(int).sum()
```

Out[39]:
```
0
```

6. Checking Particular Column data

In [40]:
```python
data['Salary']
```

Out[40]:
```
0      39343
1      46205
2      37731
3      43525
4      39891
5      56642
6      60150
7      54445
8      64445
9      57189
10     63218
11     55794
12     56957
13     57081
14     61111
15     67938
16     66029
17     83088
18     81363
19     93940
20     91738
21     98273
22    101302
23    113812
24    109431
25    105582
26    116969
27    112635
28    122391
29    121872
Name: Salary, dtype: int64
```

III. Linear Regression Initialization

In [42]:
```python
#get a copy of dataset exclude last column
# X is first column of dataset
X = data.iloc[:, :-1].values

X
```

Out[42]:
```
array([[ 1.1],
       [ 1.3],
       [ 1.5],
       [ 2. ],
       [ 2.2],
       [ 2.9],
       [ 3. ],
       [ 3.2],
       [ 3.2],
       [ 3.7],
       [ 3.9],
       [ 4. ],
       [ 4. ],
       [ 4.1],
       [ 4.5],
       [ 4.9],
       [ 5.1],
       [ 5.3],
       [ 5.9],
       [ 6. ],
       [ 6.8],
       [ 7.1],
       [ 7.9],
       [ 8.2],
       [ 8.7],
       [ 9. ],
       [ 9.5],
       [ 9.6],
       [10.3],
       [10.5]])
```

In [44]:
```python
#get array of dataset in column 1st
y = data.iloc[:, 1].values

y
```

Out[44]:
```
array([ 39343,  46205,  37731,  43525,  39891,  56642,  60150,  54445,
        64445,  57189,  63218,  55794,  56957,  57081,  61111,  67938,
        66029,  83088,  81363,  93940,  91738,  98273, 101302, 113812,
       109431, 105582, 116969, 112635, 122391, 121872], dtype=int64)
```

In [50]:
```python
# # # Splitting data
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)
```

In [52]:
```python
# # # # Fitting Simple Linear Regression to the Training set
#
# # Predicting the result of 5 Years Experience

# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[52]:
```
LinearRegression
LinearRegression()
```

In [53]:
```python
# Predicting the result of 5 Years Experience
from sklearn.linear_model import LinearRegression
y_pred = regressor.predict(X_test)
y_pred
```

Out[53]:
```
array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
       115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
        76349.68719258, 100649.1375447 ])
```

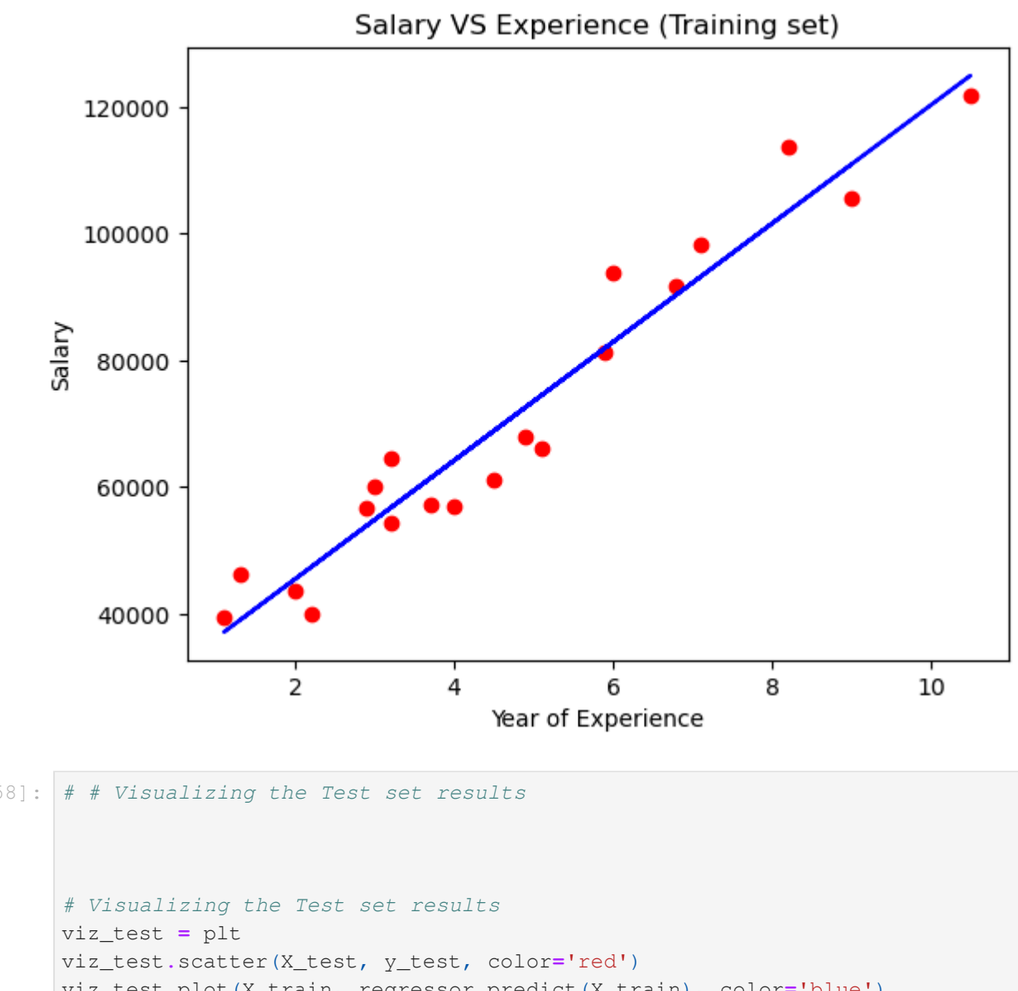
In [54]:
```python
# Predicting the result of 5 Years Experience
y_pred = regressor.predict([[5]])
y_pred
```

Out[54]:
```
array([73545.90445964])
```

In [56]:
```python
# # #  intercept and slope of a simple linear regression model in Python using scikit-learn

# print the intercept and slope
from sklearn.linear_model import LinearRegression
print("Intercept:",regressor.intercept_)
print("Slope:",regressor.coef_)

print(regressor.get_params())
```

```
Intercept: 26816.19224403119
Slope: [9345.94244312]
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'positive': False}
```

In [57]:
```python
# # # Visualizing the Training set results

# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()
```



In [58]:
```python
# # Visualizing the Test set results



# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()


# In[ ]:
```

In [ ]: