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

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

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 [3]:
```
df.head()
```

Out[3]:

	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

In [4]:
```
df.dtypes
```

Out[4]:
```
YearsExperience    float64
Salary             float64
dtype: object
```

In [6]:
```
df.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]:
```
df.isnull().sum()
```

Out[7]:
```
YearsExperience    0
Salary             0
dtype: int64
```

In [8]:

```
df.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
```

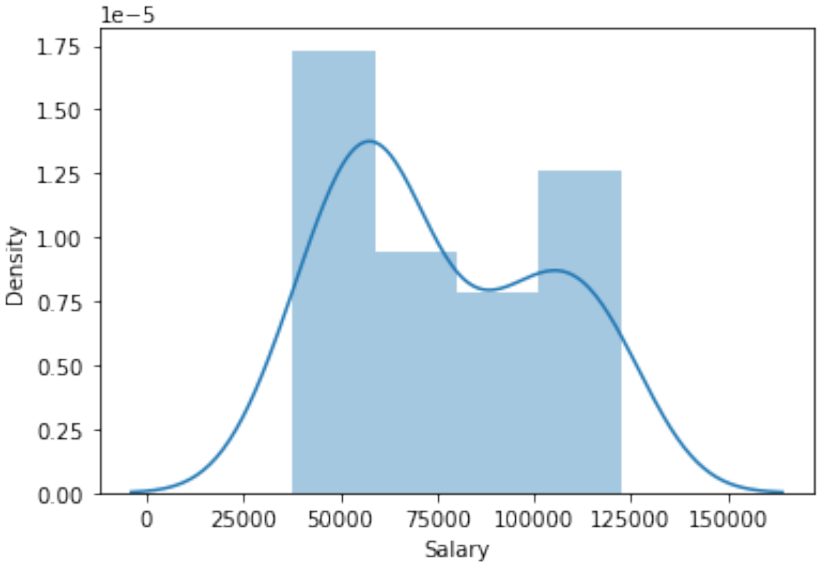
In [9]:

```
sns.distplot(df['Salary'])
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.p
y:2619: FutureWarning: `distplot` is a deprecated function and wil
l be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

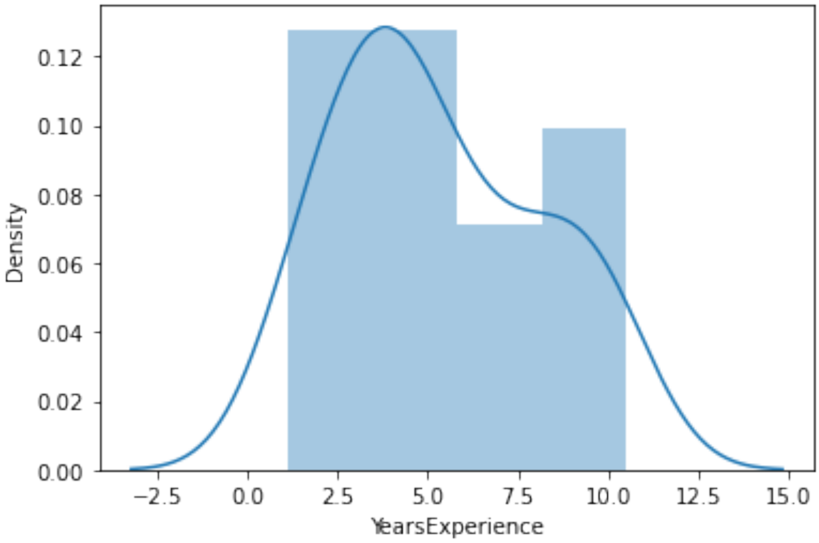
```
<AxesSubplot:xlabel='Salary', ylabel='Density'>
```



In [10]:
```
sns.distplot(df['YearsExperience'])
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.p
y:2619: FutureWarning: `distplot` is a deprecated function and wil
l be removed in a future version. Please adapt your code to use ei
ther `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:
```
<AxesSubplot:xlabel='YearsExperience', ylabel='Density'>
```



In [11]:
```
df.corr()
```

Out[11]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

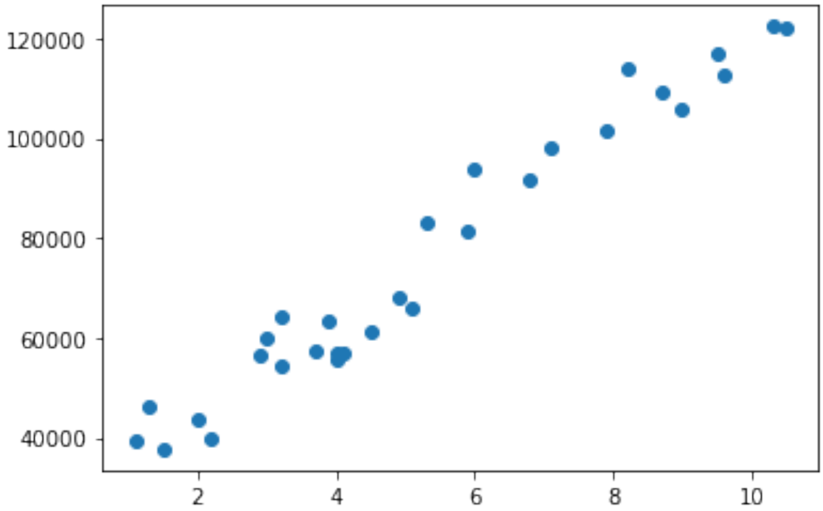
In [12]:
```
df=df.rename({'YearsExperience':'years_exp', 'Salary':'salary_hike'
df
```

Out[12]:

	years_exp	salary_hike
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

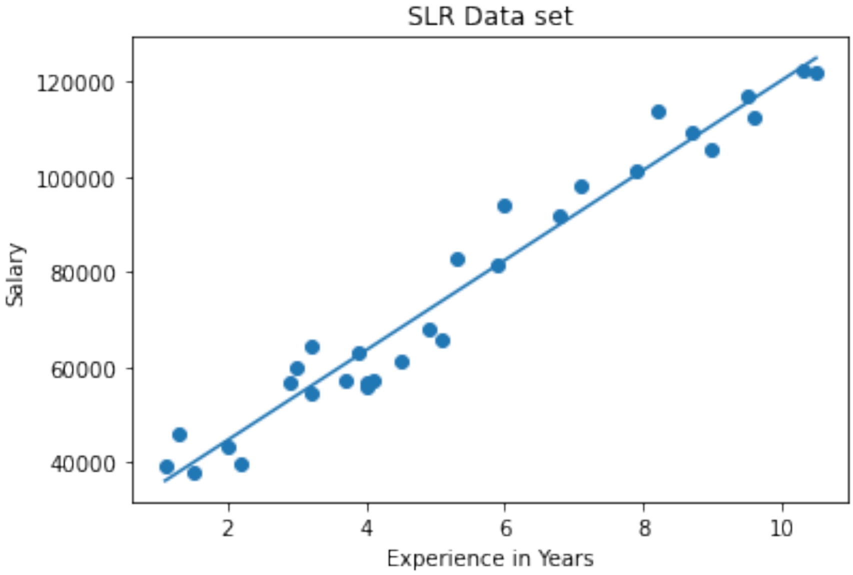
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

```
In [13]: plt.scatter(df.years_exp,df.salary_hike)

Out[13]: <matplotlib.collections.PathCollection at 0x7fda462d5e20>
```



```
In [14]: slr_model=smf.ols("df['salary_hike']~df['years_exp']",data=df).fit(
         print(slr_model.summary())
         predict=slr_model.predict(df.iloc[:,0])

         plt.scatter(df['years_exp'],df['salary_hike'])
         plt.plot(df['years_exp'],predict)
         plt.title("SLR Data set")
         plt.xlabel("Experience in Years")
         plt.ylabel("Salary")
         plt.show()
```

```
                            OLS Regression Results
==================================================================
============
Dep. Variable:      df['salary_hike']   R-squared:
0.957
Model:                            OLS   Adj. R-squared:
0.955
Method:                 Least Squares   F-statistic:
622.5
Date:                Thu, 01 Dec 2022   Prob (F-statistic):
1.14e-20
Time:                        02:19:05   Log-Likelihood:
-301.44
No. Observations:                  30   AIC:
606.9
Df Residuals:                      28   BIC:
609.7
Df Model:                           1
Covariance Type:            nonrobust
==================================================================
=================
                      coef    std err          t      P>|t|      [
0.025      0.975]
------------------------------------------------------------------
-----------------
Intercept        2.579e+04   2273.053     11.347      0.000    2.1
1e+04    3.04e+04
df['years_exp']  9449.9623    378.755     24.950      0.000    867
4.119    1.02e+04
==================================================================
============
Omnibus:                        2.140   Durbin-Watson:
1.648
Prob(Omnibus):                  0.343   Jarque-Bera (JB):
1.569
Skew:                           0.363   Prob(JB):
0.456
Kurtosis:                       2.147   Cond. No.
13.2
==================================================================
============

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

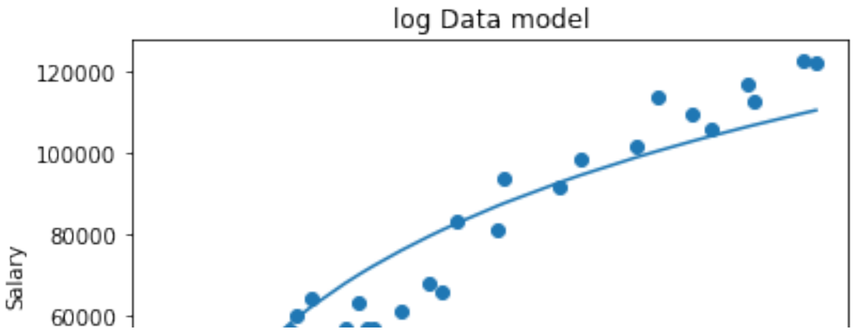


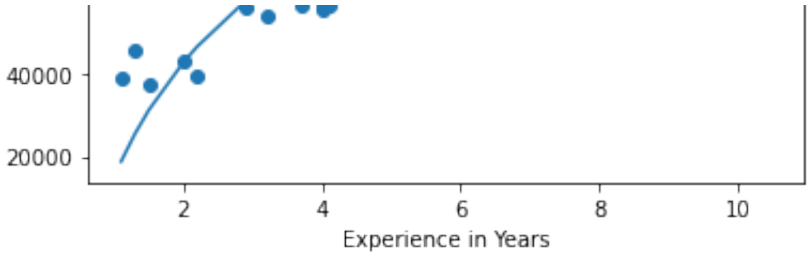
In [15]:

```
log_model=smf.ols("df['salary_hike']~np.log(df['years_exp'])", data
print(log_model.summary())
log_predict=log_model.predict(pd.DataFrame(df['years_exp']))

plt.scatter(df['years_exp'],df['salary_hike'])
plt.plot(df['years_exp'],log_predict)
plt.title("log Data model")
plt.xlabel("Experience in Years")
plt.ylabel("Salary")
plt.show()
```

```
                            OLS Regression Results
```

```
==================================================================
============
Dep. Variable:      df['salary_hike']   R-squared:
0.854
Model:                            OLS   Adj. R-squared:
0.849
Method:                 Least Squares   F-statistic:
163.6
Date:                Thu, 01 Dec 2022   Prob (F-statistic):
3.25e-13
Time:                        02:19:18   Log-Likelihood:
-319.77
No. Observations:                  30   AIC:
643.5
Df Residuals:                      28   BIC:
646.3
Df Model:                           1
Covariance Type:            nonrobust
==================================================================
=========================
                             coef    std err          t      P>|t
|      [0.025      0.975]
------------------------------------------------------------------
-------------------------
Intercept                 1.493e+04   5156.226      2.895      0.00
7    4365.921    2.55e+04
np.log(df['years_exp'])  4.058e+04   3172.453     12.792      0.00
0    3.41e+04    4.71e+04
==================================================================
============
Omnibus:                        1.094   Durbin-Watson:
0.512
Prob(Omnibus):                  0.579   Jarque-Bera (JB):
0.908
Skew:                           0.156   Prob(JB):
0.635
Kurtosis:                       2.207   Cond. No.
5.76
==================================================================
============

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

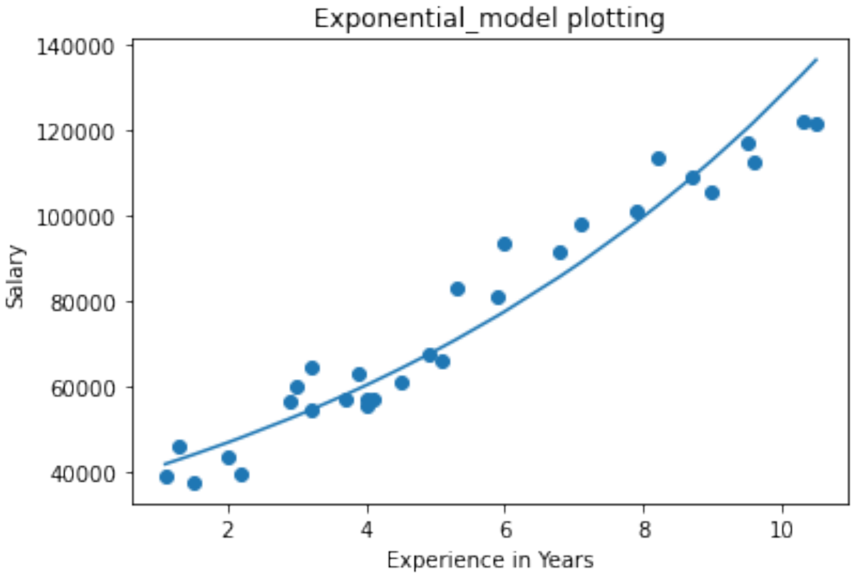




In [16]:
```
exp_model=smf.ols("np.log(df['salary_hike'])~df['years_exp']", data
print(exp_model.summary())
exp_predict=exp_model.predict(pd.DataFrame(df['years_exp']))
pred_exp=np.exp(exp_predict)

plt.scatter(df['years_exp'],df['salary_hike'])
plt.plot(df['years_exp'],np.exp(exp_predict))
plt.title("Exponential_model plotting")
plt.xlabel("Experience in Years")
plt.ylabel("Salary")
plt.show()
```

```
                                OLS Regression Results
==================================================================
===================
Dep. Variable:     np.log(df['salary_hike'])   R-squared:
0.932
Model:                                   OLS   Adj. R-squared:
0.930
Method:                        Least Squares   F-statistic:
383.6
Date:                       Thu, 01 Dec 2022   Prob (F-statistic):
7.03e-18
Time:                               02:19:28   Log-Likelihood:
28.183
No. Observations:                         30   AIC:
-52.37
Df Residuals:                             28   BIC:
-49.56
Df Model:                                  1
Covariance Type:                   nonrobust
==================================================================
=================
                      coef    std err          t      P>|t|      [
0.025      0.975]
------------------------------------------------------------------
-----------------
Intercept          10.5074      0.038    273.327      0.000      1
0.429      10.586
df['years_exp']     0.1255      0.006     19.585      0.000
0.112       0.139
==================================================================
============
Omnibus:                         0.826   Durbin-Watson:
1.438
```

```
Prob(Omnibus):                  0.661   Jarque-Bera (JB):
0.812
Skew:                           0.187   Prob(JB):
0.666
Kurtosis:                       2.286   Cond. No.
13.2
===================================================================
============

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



In [17]:

```
slr_regression=linear_model.LinearRegression()
slr_regression.fit(df[['years_exp']],df.salary_hike)

flag=True
while(flag):
    myinput=float(input("Enter Years of Experience"))
    myoutput=slr_regression.predict([[myinput]])
    print("Salary Hike precited using SLR is: ",myoutput)
    flag=int(input("press 1 to continue or press 0 to exit: "))
```

```
Enter Years of Experience5
Salary Hike precited using SLR is:  [73042.01180594]
press 1 to continue or press 0 to exit: 0
```

In [18]:

```
np.sqrt(np.mean((df.years_exp-predict)**2))
```

Out[18]:

```
80440.84508275457
```

In [19]:

```
np.sqrt(np.mean((df.years_exp-predict)**2))
```

Out[19]:

```
80440.84508275457
```

In [20]:

```
np.sqrt(np.mean((df.years_exp-log_predict)**2))
```

Out[20]:

```
79974.15496099806
```

In [ ]: