In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

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

Exploratory Data Analysis

In [3]:
```
data.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]:
```
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 [5]:
```
data.shape
```

Out[5]: (30, 2)

In [7]:
```
print(data)
```

```
    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]:
```
#null value checking
data.isna().sum()
```

Out[6]:
```
YearsExperience    0
Salary             0
dtype: int64
```

In [9]:
```
# This displays the first 5 rows of data.
data.head()
```

Out[9]:

	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 [10]:
```
# Provides some information about the columns in the data.
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 [ ]:
```

```

In [7]:
```
data.describe()
```

Out[7]:

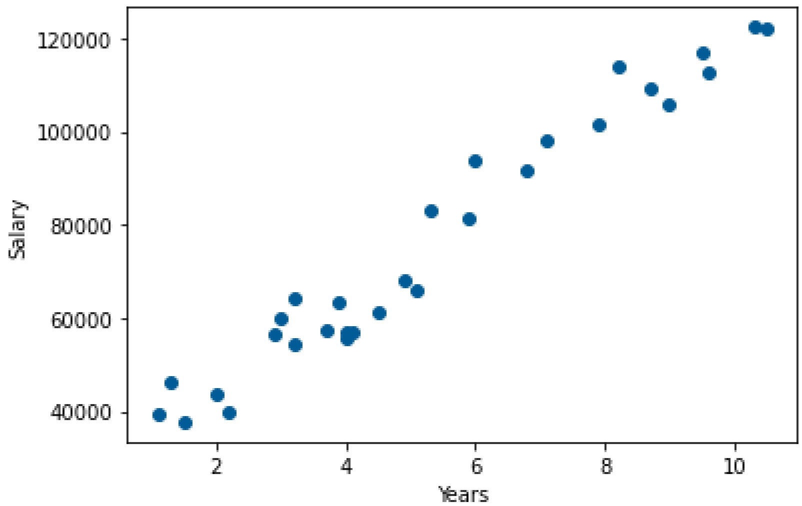
	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [8]:
```
data.describe().T
```

Out[8]:

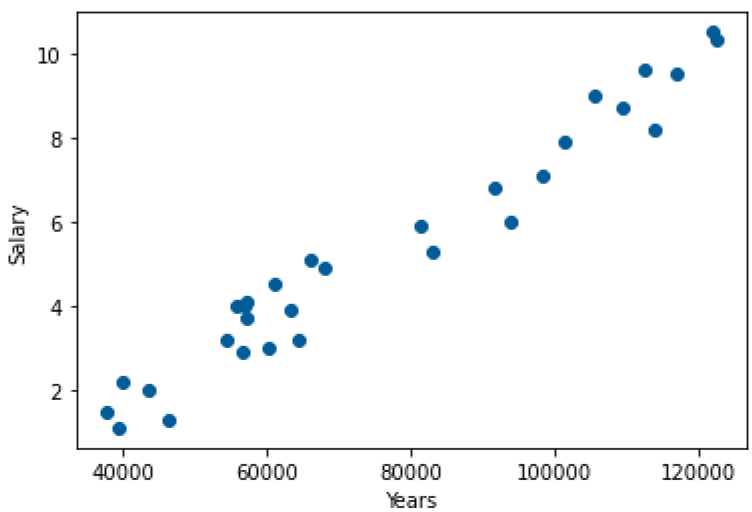
	count	mean	std	min	25%	50%	75%	
YearsExperience	30.0	5.313333	2.837888	1.1	3.20	4.7	7.70	
Salary	30.0	76003.000000	27414.429785	37731.0	56720.75	65237.0	100544.75	122

In [9]:
```
plt.scatter(data['YearsExperience'], data['Salary'], color = '#005b96')
plt.xlabel('Years')
plt.ylabel('Salary')
plt.show()
```



In [ ]:

In [10]:
```
plt.scatter( data['Salary'],data['YearsExperience'], color = '#005b96')
plt.xlabel('Years')
plt.ylabel('Salary')
plt.show()
```



In [12]:
```
x = data[['YearsExperience']]
y = data.Salary
```

In [14]:
```
from scipy.stats import pearsonr
```

In [15]:
```
corr, _ = pearsonr(data['YearsExperience'], data['Salary'])
print('Pearsons correlation: %.3f' % corr)
```

```
Pearsons correlation: 0.978
```

In [16]:
```
_
```

Out[16]:
```
1.143068109227237e-20
```

In [17]:
```
1.1430681092271564e-20<0.05
```

Out[17]:
```
True
```

In [18]:
```
np.corrcoef(data['YearsExperience'], data['Salary'])
```

Out[18]:
```
array([[1.        , 0.97824162],
       [0.97824162, 1.        ]])
```

In [19]:
```
np.corrcoef(data['YearsExperience'], data['Salary'])[0,1]
```

Out[19]:
```
0.9782416184887599
```

In [20]:
```
from scipy.stats.stats import pearsonr

pearsonr(data['YearsExperience'], data['Salary'])
```

Out[20]:
```
(0.9782416184887598, 1.143068109227237e-20)
```

In [18]:
```
1.1430681092271564e-20< 0.05
```

Out[18]:
```
True
```

In [21]:
```
data['YearsExperience'].corr(data['Salary'])
```

Out[21]:
```
0.9782416184887599
```

In [23]:
```
#define predictor and response variables
```

In [7]:
```
#x = data['ex']
#y= data['w']
```

In [23]:
```
x = data['YearsExperience']
y= data['Salary']
```

In [25]:
```
#  Model  Ordinary least squares (OLS) regression
```

In [24]:
```
 import statsmodels.api as sm
```

In [27]:
```
#add constant to predictor variables
```

In [25]:
```
x = sm.add_constant(x)
```

In [29]:
```
#fit linear regression model
```

In [27]:
```
model = sm.OLS(y, x).fit()
```

In [ ]:
```

```

In [28]:
```
#view model summary
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                 Salary   R-squared:                       0.95
7
Model:                            OLS   Adj. R-squared:                  0.95
5
Method:                 Least Squares   F-statistic:                     622.
5
Date:                Wed, 22 Jan 2025   Prob (F-statistic):           1.14e-2
0
Time:                        12:12:23   Log-Likelihood:                -301.4
4
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]
------------------------------------------------------------------------------
------
const            2.579e+04   2273.053     11.347      0.000    2.11e+04    3.
04e+04
YearsExperience  9449.9623    378.755     24.950      0.000    8674.119    1.
02e+04
==============================================================================
=
Omnibus:                        2.140   Durbin-Watson:                   1.64
8
Prob(Omnibus):                  0.343   Jarque-Bera (JB):                1.56
9
Skew:                           0.363   Prob(JB):                        0.45
6
Kurtosis:                       2.147   Cond. No.                         13.
2
==============================================================================
=

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

In [28]:
```
1.14e-20<0.05
```

Out[28]: True

In [33]:
```
#salary=9449.9623*YearsExperience+2.579e+04
```

```
y=5x+3
```

```
Here is how to interpret the rest of the model summary:

P(>|t|): This is the p-value associated with the model coefficients. Since
the p-value for hours (0.000) is less than .05, we can say that there is a
statistically significant association between YearsExperience and salary.
R-squared: This tells us the percentage of the variation in the salary can
be explained by the number of years Experience. In this case, 95.7% of the
variation in salary can be explained  YearsExperience.
F-statistic & p-value: The F-statistic ( 622.5) and the corresponding p-
value (1.14e-20) tell us the overall significance of the regression model,
i.e. whether predictor variables in the model are useful for explaining the
variation in the response variable. Since the p-value in this example is
less than .05, our model is statistically significant and YearsExperience
is deemed to be useful for explaining the variation in salary.
```

In [34]:
```
1.14e-20<0.05
```

Out[34]:
```
True
```

```
Ho:m=0
    h1:m<>0
```

In [ ]:
```
```

In [ ]:
```
0.000<0.05
```

In [29]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
#Create a model and fit it
```

In [30]:
```
lm = LinearRegression()
```

In [31]:
```
lm.fit(x, y)
```

Out[31]:
```
▾ LinearRegression
LinearRegression()
```

In [32]:
```
model1 = LinearRegression().fit(x, y)
```

In [ ]:
```
#Get results
```

In [33]:
```
r_sq = lm.score(x, y)
```

In [34]:
```
print(f"coefficient of determination: {r_sq}")
```

```
coefficient of determination: 0.9569566641435086
```

In [35]:
```
print(f"intercept: {lm.intercept_}")
```

```
intercept: 25792.20019866871
```

In [36]:
```
 print(f"slope: {lm.coef_}")
```

```
slope: [   0.         9449.96232146]
```

In [37]:
```
y_pred = lm.predict(x)
```

In [38]:
```
y_pred
```

Out[38]:
```
array([ 36187.15875227,  38077.15121656,  39967.14368085,  44692.12484158,
        46582.11730587,  53197.09093089,  54142.08716303,  56032.07962732,
        56032.07962732,  60757.06078805,  62647.05325234,  63592.04948449,
        63592.04948449,  64537.04571663,  68317.03064522,  72097.0155738 ,
        73987.00803809,  75877.00050238,  81546.97789525,  82491.9741274 ,
        90051.94398456,  92886.932681  , 100446.90253816, 103281.8912346 ,
       108006.87239533, 110841.86109176, 115566.84225249, 116511.83848464,
       123126.81210966, 125016.80457395])
```

In [ ]:
```
y
```

In [ ]:

In [ ]:

In [ ]: