import Necessary Library

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

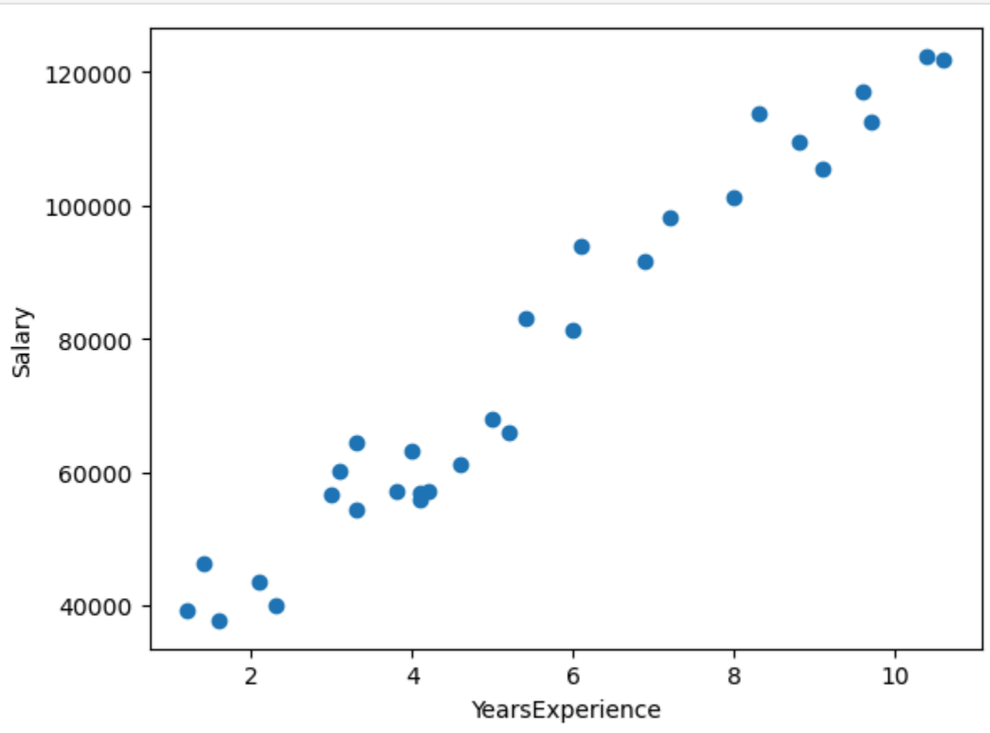
Load & Data Acquisition

In [2]:
```
df = pd.read_csv("Salary_dataset (1).csv")
df.head()
```

Out[2]:

	Unnamed: 0	YearsExperience	Salary
0	0	1.2	39344.0
1	1	1.4	46206.0
2	2	1.6	37732.0
3	3	2.1	43526.0
4	4	2.3	39892.0

In [3]:
```
# Here, we can see our dataset is sort of linear
plt.scatter(df['YearsExperience'],df['Salary'])
plt.xlabel('YearsExperience')
plt.ylabel('Salary')
plt.show()
```



In [5]:
```
# we are creating new dataframe
df1 = df[['YearsExperience','Salary']].head()
df1
```

Out[5]:

	YearsExperience	Salary
0	1.2	39344.0
1	1.4	46206.0
2	1.6	37732.0
3	2.1	43526.0
4	2.3	39892.0

In [6]:
```
# Import necessary libraries
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import tensorflow as tf

# Sample data: X (input), Y (output) for training
# Example: Let's assume we're predicting the output of a linear equation y = 2x + 1
X = df1['YearsExperience']
Y = df1['Salary']

# Build the ANN model
model = Sequential([
    Dense(units=1, input_shape=(1,), activation='linear')
    ])

# Compile the model with SGD optimizer
model.compile(optimizer='sgd', loss='mean_squared_error')

# Train the model without displaying the epochs
model.fit(X, Y, epochs=1000, verbose=0)

# Predict the output for a user input
user_input = float(input("Enter a value for prediction: "))
prediction = model.predict(np.array([[user_input]]))

# Display the prediction
print(f"Predicted output for input {user_input}: {prediction[0][0]}")
```

```
C:\Users\AMIT KUMAR\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Enter a value for prediction: 3.6
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step
Predicted output for input 3.6: 54769.53125
```

In [7]:
```
# Predict salary for each YearsExperience in the dataset
predictions = model.predict(X)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step
```

In [8]:
```
# Here, we can see clearly actual and predicted salary

df1 = pd.DataFrame({"Actual_Salary" : Y ,"Predicted_Salary": predictions.flatten()})
```

In [9]:
```
df1
```

Out[9]:

	Actual_Salary	Predicted_Salary
0	39344.0	36889.562500
1	46206.0	38379.562500
2	37732.0	39869.558594
3	43526.0	43594.550781
4	39892.0	45084.546875

In [ ]:
```

```