# Simple Linear Regression

## 2) Salary_hike -> Build a prediction model for Salary_hike

---

### Data Import and Overview:

- We begin by importing the necessary libraries: matplotlib.pyplot, numpy, pandas, and seaborn.
- The salary data is read from the 'Salary_Data.csv' file into a Pandas DataFrame named df.
- We display the DataFrame to get an initial look at the data.

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

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

In [3]:
```
df
```

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 |
| 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 |

---

### Data Exploration:

- df.head() displays the first few rows of the dataset.
- df.shape provides information about the number of rows and columns in the DataFrame.
- df.describe() gives summary statistics of the numerical columns.
- df.info() provides information about the data types and missing values.
- df.isnull().values checks for missing values in the DataFrame.

In [4]:
```python
df.head()
```

Out[4]:

| | 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 [5]:
```python
df.shape
```

Out[5]: (30, 2)

In [6]:
```python
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]:
```python
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 [8]:
```python
df.isnull().values
```

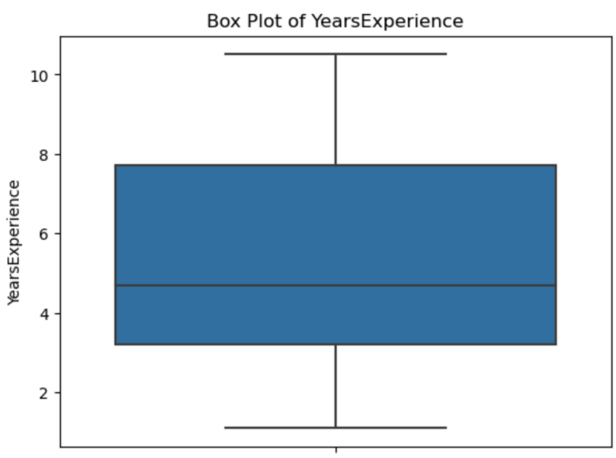
Out[8]:
```
array([[False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
       [False, False],
```

## Data Visualization:

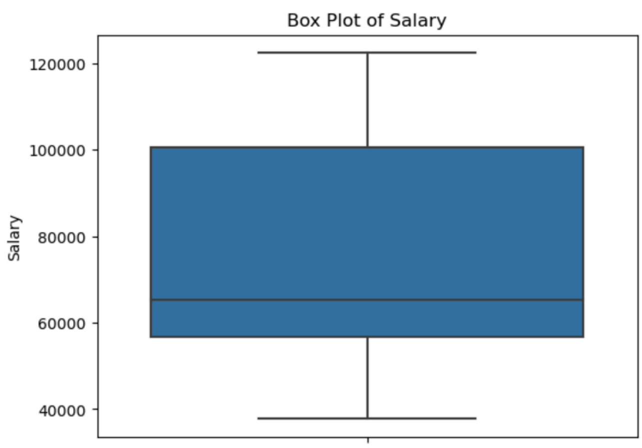
We create box plots to visualize the distribution of 'YearsExperience' and 'Salary' columns using Seaborn.

A correlation heatmap is generated to visualize the relationships between numerical variables using Seaborn.

In [9]:
```python
sns.boxplot(y=df['YearsExperience'])
plt.title('Box Plot of YearsExperience')
plt.show()
```



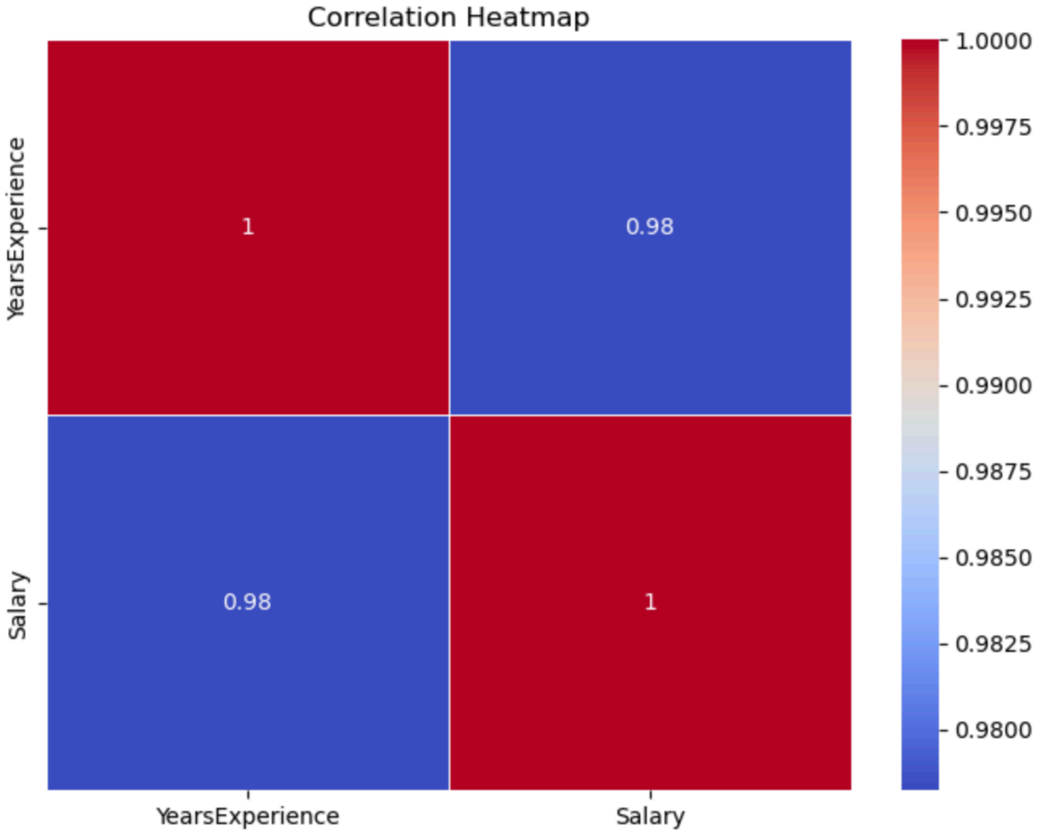

In [10]:
```python
sns.boxplot(y=df['Salary'])
plt.title('Box Plot of Salary')
plt.show()
```



In [11]:

```python
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Create a correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



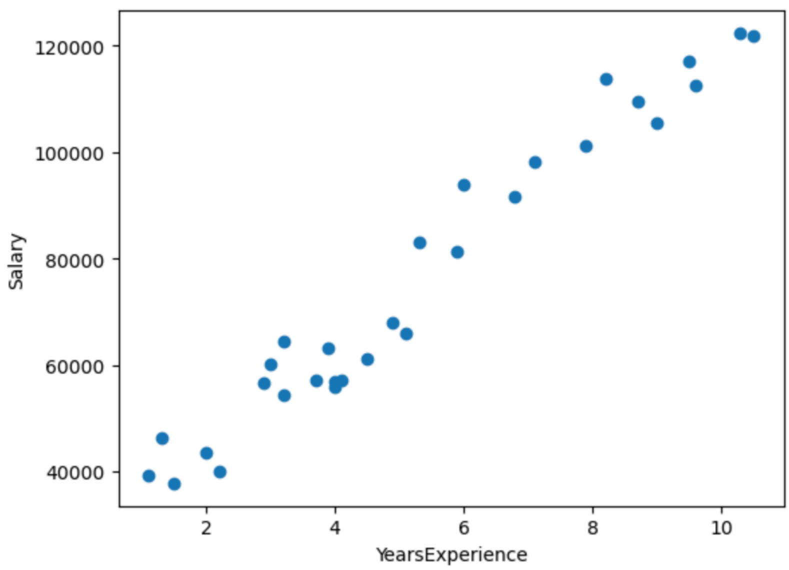

In [13]:

```python
x = df["YearsExperience"]
y = df["Salary"]
```

In [14]:

```python
plt.scatter(x,y)
plt.xlabel("YearsExperience")
plt.ylabel("Salary")
```

Out[14]: Text(0, 0.5, 'Salary')



## Linear Regression Model:

We prepare the data for building a linear regression model.

x represents the independent variable 'YearsExperience', and y represents the dependent variable 'Salary'.

Data is split into training and testing sets using train_test_split from sklearn.model_selection.

The independent variables in the training and testing sets are reshaped using NumPy.

### Linear Regression Modeling:

We import the LinearRegression model from sklearn.linear_model.
An instance of the linear regression model lr is created and fitted with the training data.

In [15]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)
```

In [16]:
```
x_train = np.array(x_train).reshape(-1,1)
x_test = np.array(x_test).reshape(-1,1)
```

In [17]:
```
from sklearn.linear_model import LinearRegression
```

In [18]:
```
lr = LinearRegression()
```

In [19]:
```
lr.fit(x_train, y_train)
```

Out[19]:
```
▾ LinearRegression
LinearRegression()
```

In [21]:
```
y_train
```

Out[21]:
```
20     91738.0
5      56642.0
27    112635.0
12     56957.0
4      39891.0
10     63218.0
16     66029.0
28    122391.0
25    105582.0
17     83088.0
2      37731.0
7      54445.0
26    116969.0
24    109431.0
18     81363.0
11     55794.0
22    101302.0
29    121872.0
13     57081.0
15     67938.0
8      64445.0
Name: Salary, dtype: float64
```

In [22]:
```
y_predict_train = lr.predict(x_train)
y_predict_train
```

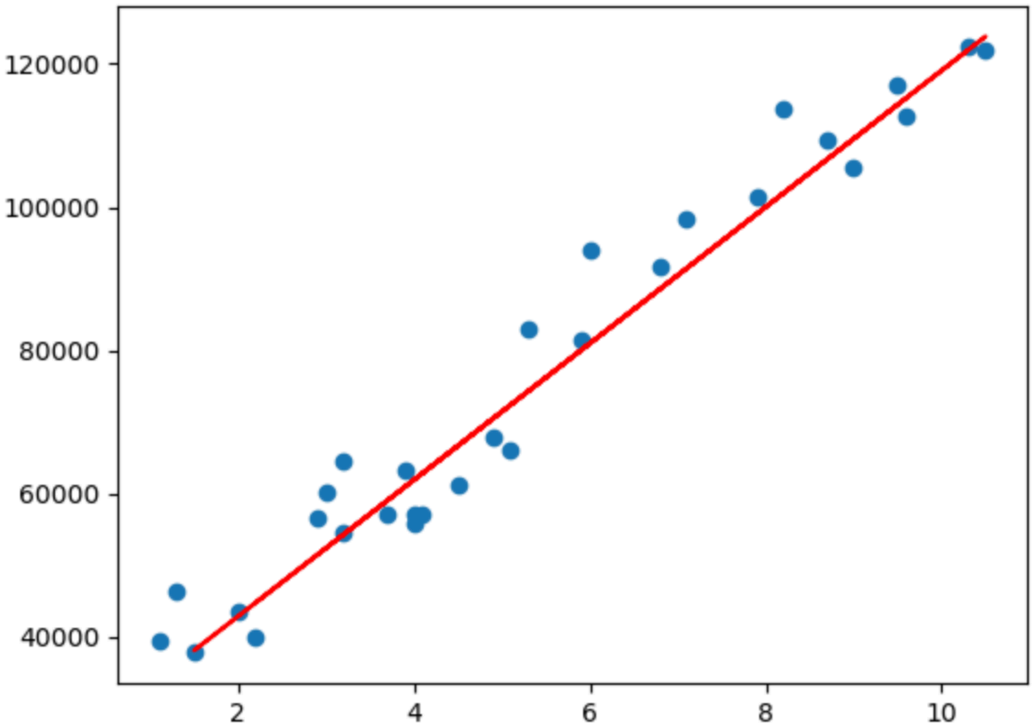
Out[22]:
```
array([ 88574.21720865,  51396.1586181 , 115266.15670955,  61882.27770774,
        44723.17374288,  60928.99415414,  72368.39679738, 121939.14158477,
       109546.45538793,  74274.96390459,  38050.18886765,  54256.00927891,
       114312.87315594, 106686.60472711,  79994.66522621,  61882.27770774,
        99060.33629829, 123845.70869198,  62835.56126135,  70461.82969018,
        54256.00927891])
```

## Model Visualization:

We plot a scatter plot of 'YearsExperience' vs. 'Salary' to visualize the data points and overlay the regression line on the training data.

In [23]:
```
plt.scatter(x,y)
plt.plot(x_train, y_predict_train, color='red')
```

Out[23]:
```
[<matplotlib.lines.Line2D at 0x11ed9a100>]
```




## Salary Prediction:

We define a function Salary(lr) to predict salaries based on years of experience.
Users are prompted to input their years of experience, and the function uses the trained linear regression model to predict their salary.
The predicted salary is printed as 'Expected Salary'.

In [24]:
```
def Salary(lr):
    new_experience = float(input('Enter Experience: '))
    new_experience = np.array(new_experience).reshape(1, 1)

    #eS expected salary
    eS = lr.predict(new_experience)
    print('Expected Salary :-',eS)
```

In [26]:
```
Salary(lr)
```

```
Enter Experience: 15
Expected Salary :- [166743.46860415]
```

In [ ]:

---

## Conclusion:

We conclude the assignment by summarizing the key steps taken, including data import, exploration, visualization, model building, and prediction

These steps provide a comprehensive analysis of the salary data and demonstrate the application of linear regression for salary prediction based on years of experience.

------------------------------------------------------------