In [2]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
df=pd.read_csv("Salary_Data[1].csv")
```

In [4]:
```
df.head(10)
```

Out[4]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0
5	29.0	Male	Bachelor's	Marketing Analyst	2.0	55000.0
6	42.0	Female	Master's	Product Manager	12.0	120000.0
7	31.0	Male	Bachelor's	Sales Manager	4.0	80000.0
8	26.0	Female	Bachelor's	Marketing Coordinator	1.0	45000.0
9	38.0	Male	PhD	Senior Scientist	10.0	110000.0

In [5]:
```
df['Gender'] = df['Gender'].map({'Male': 1, 'Female': 0})
df['Education Level'] = df['Education Level'].map({'Bachelor': 0, 'Master': 1, 'PhD': 2}
```

In [6]:
```
df = df.dropna()
#dropping the null values
```

In [57]:
```
print(df['Education Level'].unique())
#Converting categorical data to numerical data
df['Education Level'] = df['Education Level'].map({'Bachelor': 0, 'Master': 1, 'PhD': 2})
print(df)
```

```
[2.]
       Age  Gender  Education Level                         Job Title  \
2     45.0     1.0              NaN                    Senior Manager
9     38.0     1.0              NaN                   Senior Scientist
17    39.0     1.0              NaN                    Senior Engineer
28    43.0     0.0              NaN                  Senior Consultant
34    46.0     1.0              NaN                    Senior Manager
...    ...     ...              ...                               ...
6682  28.0     1.0              NaN                  Marketing Manager
6685  49.0     0.0              NaN  Senior Product Marketing Manager
6691  36.0     0.0              NaN                  Marketing Manager
6696  28.0     1.0              NaN              Sales Representative
6699  49.0     0.0              NaN             Director of Marketing

      Years of Experience    Salary
2                    15.0  150000.0
9                    10.0  110000.0
17                   12.0  115000.0
28                   15.0  140000.0
34                   20.0  170000.0
...                   ...       ...
6682                  4.0   55000.0
6685                 20.0  200000.0
6691                  9.0   95000.0
6696                  4.0   55000.0
6699                 20.0  200000.0

[1368 rows x 6 columns]
```

In [58]:
```
X = df[['Age', 'Gender', 'Years of Experience']].values
y = df['Salary'].values
```

In [59]:
```
print(X)
```

```
[[45.  1. 15.]
 [38.  1. 10.]
 [39.  1. 12.]
 ...
 [36.  0.  9.]
 [28.  1.  4.]
 [49.  0. 20.]]
```

In [60]:
```
print(y)
```

```
[150000. 110000. 115000. ...  95000.  55000. 200000.]
```

In [61]:
```
X = (X - np.mean(X, axis=0)) / np.std(X, axis=0)
```

In [62]:
```
X = np.c_[np.ones(X.shape[0]), X]
```

In [63]:
```
W = np.zeros(X.shape[1])
```

In [64]:
```
def hypothesis(X, W):
    return np.dot(X, W)
```

In [65]:
```
def cost_function(X, y, W):
    m = len(y)
    return (1 / (2 * m)) * np.sum((hypothesis(X, W) - y) ** 2)
```

In [66]:
```
# Gradient Descent Algorithm
def gradient_descent(X, y, W, alpha, num_iterations):
    m = len(y)
    cost_history = []

    for i in range(num_iterations):
        # Updating the weights
        W = W - (alpha / m) * np.dot(X.T, (hypothesis(X, W) - y))

        # Calculating cost and store in history
        cost_history.append(cost_function(X, y, W))

    return W, cost_history
```

In [67]:
```
alpha = 0.001
num_iterations = 1000

# Training the model
W, cost_history = gradient_descent(X, y, W, alpha, num_iterations)

print("Weights:", W)
print("Cost history:", cost_history[-10:])  # last 10 cost values
```

```
Weights: [104763.27538946   9784.07848236    969.10878162  10270.9667527 ]
Cost history: [2218459625.0635314, 2214658531.9170775, 2210865076.5842247, 2207079243.6
43146, 2203301017.7034383, 2199530383.406061, 2195767325.423268, 2192011828.4585447, 21
88263877.246542, 2184523456.5530114]
```

In [69]:
```
def predict(X, W): return hypothesis(X, W)
```

In [70]:
```
predicted_salaries = predict(X, W)
```

In [71]:
```
df['Predicted Salary'] = predicted_salaries
```

In [72]:
```
print("DataFrame with Predicted Salaries:\n", df.head())
```

```
DataFrame with Predicted Salaries:
      Age  Gender  Education Level          Job Title  Years of Experience  \
2   45.0     1.0              NaN     Senior Manager                 15.0
9   38.0     1.0              NaN   Senior Scientist                 10.0
17  39.0     1.0              NaN    Senior Engineer                 12.0
28  43.0     0.0              NaN  Senior Consultant                 15.0
34  46.0     1.0              NaN     Senior Manager                 20.0

      Salary  Predicted Salary
2   150000.0     112354.766614
9   110000.0      93925.862301
17  115000.0      99065.181062
28  140000.0     107858.640469
34  170000.0     123342.861048
```

In [74]:
```
df.head(10)
```

Out[74]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary	Predicted Salary
2	45.0	1.0	NaN	Senior Manager	15.0	150000.0	112354.766614
9	38.0	1.0	NaN	Senior Scientist	10.0	110000.0	93925.862301
17	39.0	1.0	NaN	Senior Engineer	12.0	115000.0	99065.181062
28	43.0	0.0	NaN	Senior Consultant	15.0	140000.0	107858.640469
34	46.0	1.0	NaN	Senior Manager	20.0	170000.0	123342.861048
48	38.0	1.0	NaN	Senior Scientist	11.0	120000.0	95875.454192
57	43.0	1.0	NaN	Senior Engineer	17.0	140000.0	113773.680436
63	47.0	1.0	NaN	Senior Data Scientist	21.0	180000.0	126532.587919
72	45.0	1.0	NaN	Research Director	16.0	190000.0	114304.358505
83	52.0	1.0	NaN	Chief Technology Officer	24.0	250000.0	138582.038491

In [ ]:
```

```