In [1]:
```
import numpy as np
```

In [2]:
```
# Function to perform single-variable linear regression
def linear_regression(X, y):
    n = len(X)
```

In [4]:
```
def linear_regression(X, y):
    n = len(X)

    # Calculate mean of X and y
    X_mean = np.mean(X)
    y_mean = np.mean(y)

    # Calculate slope (m) and y-intercept (c) using least squares method
    numerator = sum((X[i] - X_mean) * (y[i] - y_mean) for i in range(n))
    denominator = sum((X[i] - X_mean) ** 2 for i in range(n))

    m = numerator / denominator
    c = y_mean - m * X_mean

    # Calculate predicted y values
    y_pred = [m * X[i] + c for i in range(n)]

    # Calculate Sum of Squared Errors (SSE)
    sse = sum((y[i] - y_pred[i]) ** 2 for i in range(n))

    # Calculate Total Sum of Squares (SST)
    sst = sum((y[i] - y_mean) ** 2 for i in range(n))

    # Calculate R Square
    r_square = 1 - (sse / sst)

    # Calculate Adjusted R Square
    adj_r_square = 1 - (sse / (n - 2 - 1)) / (sst / (n - 1))

    return m, c, sse, sst, r_square, adj_r_square
```

In [5]:
```
# Input data
X = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
```

In [6]:
```
# Perform linear regression
m, c, sse, sst, r_square, adj_r_square = linear_regression(X, y)
```

In [7]:
```
# Print results
print("Slope (m):", m)
print("Y-intercept (c):", c)
print("Sum of Squared Errors (SSE):", sse)
print("Total Sum of Squares (SST):", sst)
print("R Square:", r_square)
print("Adjusted R Square:", adj_r_square)
```

```
Slope (m): 1.1696969696969697
Y-intercept (c): 1.2363636363636363
Sum of Squared Errors (SSE): 5.624242424242421
Total Sum of Squares (SST): 118.5
R Square: 0.952538038613988
Adjusted R Square: 0.9389774782179846
```

In [ ]: