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

```
In [3]: df = pd.read_csv(r"C:\Users\shaik\OneDrive\Desktop\classroom\(40)- 22nd simple l
```

```
In [4]: df.mean() # this will give mean of entire dataframe
```

```
Out[4]: YearsExperience        5.313333
        Salary             76003.000000
        dtype: float64
```

```
In [5]: df['Salary'].mean() # this will give us mean of that particular column
```

```
Out[5]: np.float64(76003.0)
```

Median

```
In [6]: df.median() # this will give median of entire dataframe
```

```
Out[6]: YearsExperience        4.7
        Salary             65237.0
        dtype: float64
```

```
In [7]: df['Salary'].median() # this will give us median of that particular column
```

```
Out[7]: 65237.0
```

Mode

```
In [8]:  df['Salary'].mode() # this will give us mode of that particular column
```

Out[8]:

```
0      37731
1      39343
2      39891
3      43525
4      46205
5      54445
6      55794
7      56642
8      56957
9      57081
10     57189
11     60150
12     61111
13     63218
14     64445
15     66029
16     67938
17     81363
18     83088
19     91738
20     93940
21     98273
22    101302
23    105582
24    109431
25    112635
26    113812
27    116969
28    121872
29    122391
Name: Salary, dtype: int64
```

In [9]:

```
df.var() # this will give variance of entire dataframe
```

Out[9]:

```
YearsExperience    8.053609e+00
Salary             7.515510e+08
dtype: float64
```

In [10]:

```
df['Salary'].var() # this will give us variance of that particular column
```

Out[10]:

```
751550960.4137931
```

Standard deviation

In [11]:

```
df.std() # this will give standard deviation of entire dataframe
```

Out[11]:

```
YearsExperience        2.837888
Salary             27414.429785
dtype: float64
```

In [12]:

```
df['Salary'].std() # this will give us standard deviation of that particular col
```

Out[12]:

```
27414.4297845823
```

Coefficient of variation(cv)

In [14]:

```
# for calculating cv we have to import a library first
from scipy.stats import variation
variation(df.values) # this will give cv of entire dataframe
```

Out[14]:

```
array([0.5251297 , 0.35463929])
```

In [15]:

```
variation(df['Salary']) # this will give us cv of that particular column
```

Out[15]:

```
np.float64(0.3546392938275572)
```

Correlation

In [16]:

```
df.corr() # this will give correlation of entire dataframe
```

Out[16]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

In [17]:

```
df['Salary'].corr(df['YearsExperience'])
```

Out[17]:

```
np.float64(0.9782416184887598)
```

Skewness

In [18]:

```
df.skew() # this will give skewness of entire dataframe
```

Out[18]:

```
YearsExperience    0.37956
Salary             0.35412
dtype: float64
```

In [19]:

```
df['Salary'].skew() # this will give us skewness of that particular column
```

Out[19]:

```
np.float64(0.35411967922959153)
```

Standard Error

In [20]:

```
df.sem() # this will give standard error of entire dataframe
```

Out[20]:

```
YearsExperience       0.518125
Salary             5005.167198
dtype: float64
```

In [21]:

```
df['Salary'].sem() # this will give us standard error of that particular column
```

Out[21]:

```
np.float64(5005.167198052405)
```

Z-score

In [23]:

```
pip install scipy
```

```
Requirement already satisfied: scipy in c:\users\shaik\anaconda3\lib\site-package
s (1.15.3)
Requirement already satisfied: numpy<2.5,>=1.23.5 in c:\users\shaik\anaconda3\lib
\site-packages (from scipy) (2.1.3)
Note: you may need to restart the kernel to use updated packages.
```

In [29]:

```
# For calculating Z-score we have to import a library first
import scipy.stats as stats

# This will give Z-score of the entire DataFrame
df.apply(stats.zscore)
```

Out[29]:

	YearsExperience	Salary
0	-1.510053	-1.360113
1	-1.438373	-1.105527
2	-1.366693	-1.419919
3	-1.187494	-1.204957
4	-1.115814	-1.339781
5	-0.864935	-0.718307
6	-0.829096	-0.588158
7	-0.757416	-0.799817
8	-0.757416	-0.428810
9	-0.578216	-0.698013
10	-0.506537	-0.474333
11	-0.470697	-0.749769
12	-0.470697	-0.706620
13	-0.434857	-0.702020
14	-0.291498	-0.552504
15	-0.148138	-0.299217
16	-0.076458	-0.370043
17	-0.004779	0.262859
18	0.210261	0.198860
19	0.246100	0.665476
20	0.532819	0.583780
21	0.640339	0.826233
22	0.927058	0.938611
23	1.034577	1.402741
24	1.213777	1.240203
25	1.321296	1.097402
26	1.500496	1.519868
27	1.536336	1.359074
28	1.787215	1.721028
29	1.858894	1.701773

In [31]:

```
stats.zscore(df['Salary']) # this will give us Z-score of that particular column
```

Out[31]:

```
array([-1.36011263, -1.10552744, -1.419919  , -1.20495739, -1.33978143,
       -0.71830716, -0.58815781, -0.79981746, -0.42881019, -0.69801306,
       -0.47433279, -0.74976858, -0.70662043, -0.70201994, -0.55250402,
       -0.29921736, -0.37004264,  0.26285865,  0.19885989,  0.66547573,
        0.58377993,  0.82623317,  0.93861127,  1.40274136,  1.24020308,
        1.09740238,  1.51986835,  1.3590738 ,  1.72102849,  1.70177321])
```

In [32]:

```
a = df.shape[0] # this will gives us no.of rows
b = df.shape[1] # this will give us no.of columns
degree_of_freedom = a-b
print(degree_of_freedom) # this will give us degree of freedom for entire datas
```

```
28
```

Sum of Squares Regression (SSR)

In [34]:

```
# First we have to separate dependent and independent variables
X = df.iloc[:, :-1].values  # Independent variables
y = df.iloc[:, 1].values    # Dependent variable

# This will calculate mean of dependent variable
y_mean = np.mean(y)

# Splitting dataset into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random

# Linear Regression
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)

# Predicting test set results
y_predict = reg.predict(X_test)

# Calculating SSR (Sum of Squares due to Regression)
SSR = np.sum((y_predict - y_mean) ** 2)
print(SSR)
```

```
6263152884.28413
```

Sum of Squares Error (SSE)

In [36]:

```
# First we have to separate dependent and independent variables
X = df.iloc[:, :-1].values  # Independent variables
y = df.iloc[:, 1].values    # Dependent variable (verify if column 1 is correct)

# Splitting dataset into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random

# Linear Regression
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

```
# Predicting test set results
y_predict = reg.predict(X_test)

# Calculating SSE (Sum of Squared Errors)
SSE = np.sum((y_test - y_predict) ** 2)
print(SSE)
```

```
76940473.78875929
```

In [38]:

```
y = df.iloc[:, -1].values  # or use df['target_column_name'].values
y_mean = np.mean(y)
SST = np.sum((y - y_mean) ** 2)
print("SST:", SST)
```

```
SST: 21794977852.0
```

R-Square

In [39]:

```
 r_square = SSR/SST
 r_square
```

Out[39]:

```
np.float64(0.28736679279118404)
```

In [ ]: