```
import pandas as pd

df = pd.read_csv('Simple_linear_regression.csv')
print(df)
```

```
     SAT   GPA
0   1714  2.40
1   1664  2.52
2   1760  2.54
3   1685  2.74
4   1693  2.83
..   ...   ...
79  1936  3.71
80  1810  3.71
81  1987  3.73
82  1962  3.76
83  2050  3.81

[84 rows x 2 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   SAT     84 non-null     int64  
 1   GPA     84 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 1.4 KB
```

```
df.describe()
```

	SAT	GPA
count	84.000000	84.000000
mean	1845.273810	3.330238
std	104.530661	0.271617
min	1634.000000	2.400000
25%	1772.000000	3.190000
50%	1846.000000	3.380000
75%	1934.000000	3.502500
max	2050.000000	3.810000

```
df.head()
```

	SAT	GPA
0	1714	2.40
1	1664	2.52
2	1760	2.54
3	1685	2.74
4	1693	2.83

```
df.tail()
```

	SAT	GPA
79	1936	3.71
80	1810	3.71
81	1987	3.73
82	1962	3.76
83	2050	3.81

```
#doing simple linear regression
a = df['SAT'].mean()
```

```
b = df['GPA'].mean()
print(f"the mean of SAT score is {a}")
print(f"the mean of GPA is {b}")
```

```
the mean of SAT score is 1845.2738095238096
the mean of GPA is 3.330238095238095
```

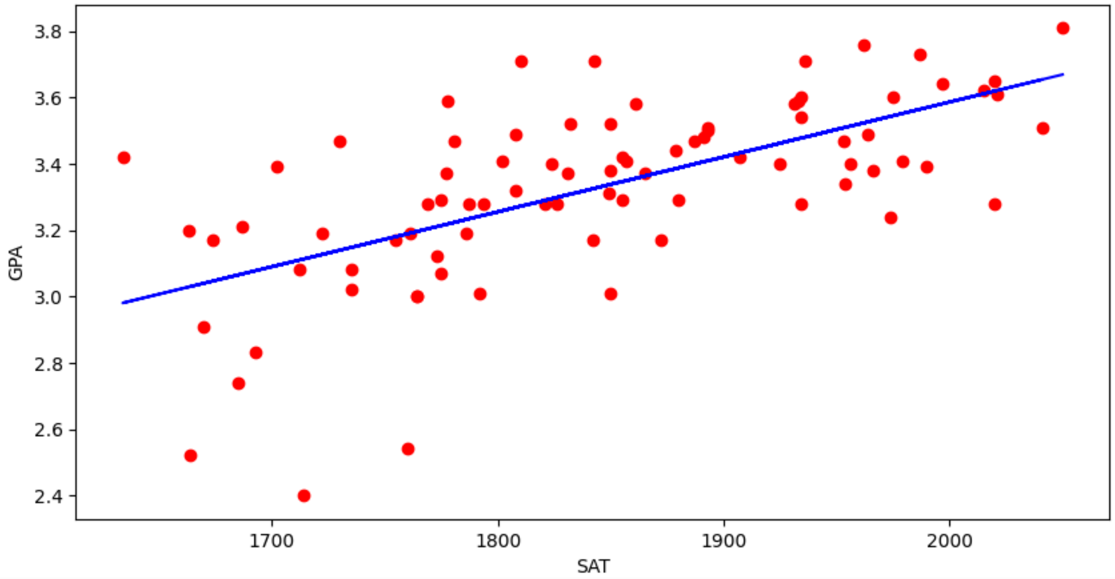
```
import numpy as np
sum_1 = 0
sum_2 = 0
for i in range(len(df)):
    sum_1 += (df['SAT'][i] - a) * (df['GPA'][i] - b)
    sum_2 += (df['SAT'][i] - a) ** 2
m = sum_1 / sum_2
print(m)
```

```
0.001655688050092815
```

```
b_not = b - m * a
print(b_not)
```

```
0.2750402996602781
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 5))
plt.scatter(df['SAT'], df['GPA'],color='red')
plt.plot(df['SAT'], m * df['SAT'] + b_not, color='blue')
plt.xlabel('SAT')
plt.ylabel('GPA')
plt.show()
```



```
y_head = 0.275+0.016*df['SAT']
df['y_head'] = y_head
df
```

	SAT	GPA	y_head
0	1714	2.40	27.699
1	1664	2.52	26.899
2	1760	2.54	28.435
3	1685	2.74	27.235
4	1693	2.83	27.363
...	...	...	...
79	1936	3.71	31.251
80	1810	3.71	29.235
81	1987	3.73	32.067
82	1962	3.76	31.667
83	2050	3.81	33.075

84 rows × 3 columns

```
error = df['SAT']-y_head
error
```

	SAT
0	1686.301
1	1637.101
2	1731.565
3	1657.765
4	1665.637
...	...
79	1904.749
80	1780.765
81	1954.933
82	1930.333
83	2016.925

84 rows × 1 columns

dtype: float64

```
#sum of errors
sum_error =sum(error)**2
print(sum_error)
```

```
23256204860.021908
```

```
#Standard error in beta1
standard_error = (sum_error/(len(df)-2))**0.5
print(standard_error)
```

```
16840.791382548243
```

```
#finding t_score
F_score = m/standard_error
print(F_score)
```

```
9.831414762423634e-08
```

```
#degree of freedom of the simple linear regression
dof = len(df)-2
print(dof)
```

```
82
```

```
#equation = 2x+3
#t_tab = 1.6
```

```
#t_calc = 0.432
```

```
#equtation = 2x+3 +some random error
x1 = np.random.randint(0,10,100)
y_new = 2*x1+3+np.random.normal(0,1,100)
df_new = pd.DataFrame({'x1':x1,'y_new':y_new})
df_new
```

	x1	y_new
0	3	6.086516
1	7	16.260355
2	6	14.787681
3	7	18.591842
4	8	19.433150
...	...	...
95	4	10.319068
96	9	20.718433
97	5	14.207255
98	7	17.682394
99	3	8.429829

100 rows × 2 columns

```
df_new.describe()
```

	x1	y_new
count	100.000000	100.000000
mean	5.020000	12.965329
std	2.651396	5.388013
min	0.000000	1.358220
25%	3.000000	8.754225
50%	5.000000	13.180374
75%	7.000000	17.515330
max	9.000000	22.164272

```
x_new_mean = np.mean(df_new['x1'])
y_new_mean = np.mean(df_new['y_new'])
print(f"{x_new_mean:.2f}")
print(f"{y_new_mean:.2f}")
```

```
5.02
12.97
```

```
#calculatiing b_not and b_one
a1 = 0
b1 = 0
for i in range(len(df_new)):
  a1 =+(df_new['x1'][i]-x_new_mean)*(df_new['y_new'][i]-y_new_mean)
  b1 =+(df_new['x1'][i]-x_new_mean)**2
  m1 = a1/b1
print(m1)
```

```
2.2452972168324674
```

```
plt.figure(figsize=(10, 5))
plt.scatter(df_new['x1'], df_new['y_new'],color='red')
plt.plot(df_new['x1'], m * df_new['x1'] + b1, color='blue')
plt.xlabel('x1')
plt.ylabel('y_new')
plt.show()
```

```
y_head1 = a1+b1*df_new['x1']
df['y_head1'] = y_head1
df_new
```

	x1	y_new
0	3	6.086516
1	7	16.260355
2	6	14.787681
3	7	18.591842
4	8	19.433150
...	...	...
95	4	10.319068
96	9	20.718433
97	5	14.207255
98	7	17.682394
99	3	8.429829

100 rows × 2 columns

```
x2 = np.random.randint(0,10,100)
y_new2 = 2*x2*x2
df_new1 = pd.DataFrame({'x2':x2,'y_new2':y_new2})
df_new1
```

	x2	y_new2
0	6	72
1	6	72
2	2	8
3	8	128
4	0	0
...	...	...
95	0	0
96	3	18
97	7	98
98	4	32
99	8	128

100 rows × 2 columns

```
df_new1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   x2      100 non-null    int64
 1   y_new2  100 non-null    int64
dtypes: int64(2)
memory usage: 1.7 KB
```

```
df_new1.describe()
```

	x2	y_new2
count	100.000000	100.000000
mean	4.670000	59.500000
std	2.832192	55.383603
min	0.000000	0.000000
25%	2.750000	15.500000
50%	4.000000	32.000000
75%	7.000000	98.000000
max	9.000000	162.000000

```
x_new_mean1 = np.mean(df_new1['x2'])
y_new_mean1 = np.mean(df_new1['y_new2'])
print(f"{x_new_mean1:.2f}")
print(f"{y_new_mean1:.2f}")
```

```
4.67
59.50
```

```
a2 = 0
b2 = 0
for i in range(len(df_new)):
  a2 =+(df_new1['x2'][i]-x_new_mean1)*(df_new1['y_new2'][i]-y_new_mean1)
  b2 =+(df_new1['x2'][i]-x_new_mean1)**2
  m2 = a2/b2
print(m2)
```

```
20.57057057057057
```

Start coding or generate with AI.