```
#Linear Regression Liberies

import numpy as np #numerical python
import pandas as pd #data processing
import seaborn as sns #for regression plot
from statsmodels.formula.api import ols


df = pd.read_csv('bert_pares.csv') #upload the data


#Check the dataset

df.head(20) #print the first 5 rows
```

	days	sales	pares_kawali	pares_flowers
0	1	9524.0	1008.0	609.0
1	2	19716.0	1041.0	529.0
2	3	6354.0	1625.0	281.0
3	4	18706.0	1143.0	1591.0
4	5	11462.0	1875.0	1629.0
5	6	17804.0	686.0	1026.0
6	7	18716.0	1318.0	1209.0
7	8	10964.0	590.0	1195.0
8	9	13183.0	1087.0	962.0
9	10	10019.0	1524.0	206.0
10	11	9087.0	748.0	616.0
11	12	NaN	496.0	1722.0
12	13	14748.0	206.0	1895.0
13	14	9043.0	1225.0	1472.0
14	15	5939.0	NaN	764.0
15	16	16495.0	229.0	988.0
16	17	NaN	1254.0	1150.0
17	18	17276.0	1680.0	738.0
18	19	11690.0	1936.0	1514.0
19	20	12181.0	NaN	751.0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#check the names of columns
df.columns
```

```
Index(['days', 'sales', 'pares_kawali', 'pares_flowers'], dtype='object')
```

```
df.dtypes
```

	0
days	int64
sales	float64
pares_kawali	float64
pares_flowers	float64

```
df.shape #(rows, columns)
```

```
(30, 4)
```

```
#Count ths missing values
df.isnull().sum()
```

	0
days	0
sales	3
pares_kawali	3
pares_flowers	1

dtype: int64

```
#Replace the missing values with mean

df['pares_flowers'].fillna(df['pares_flowers'].mean(), inplace=True)
```

```
<ipython-input-29-936de77511ee>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['pares_flowers'].fillna(df['pares_flowers'].mean(), inplace=True)
```

```
#Check for the missing values

df.isnull().sum()
```

	0
days	0
sales	3
pares_kawali	3
pares_flowers	0

```
#Compute for Summary Measures

df.describe()
```

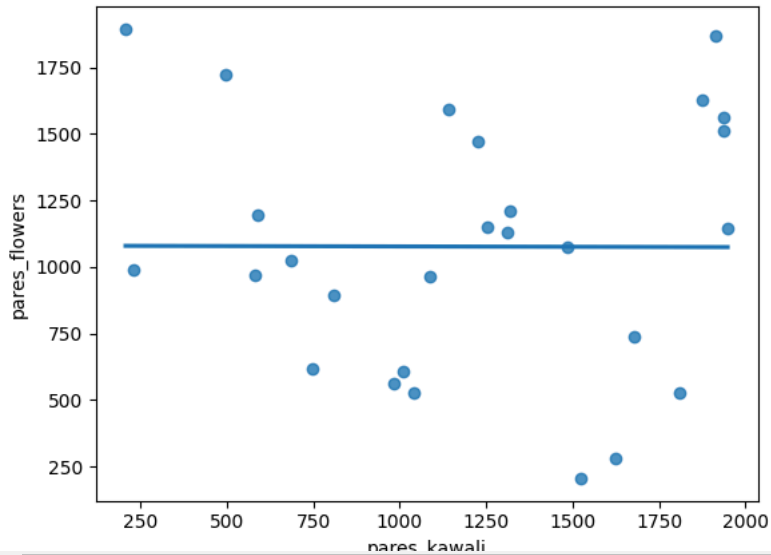
	days	sales	pares_kawali	pares_flowers
count	30.000000	27.000000	27.000000	30.000000
mean	15.500000	13193.037037	1201.777778	1072.034483
std	8.803408	4205.114464	538.420791	464.005535
min	1.000000	5124.000000	206.000000	206.000000
25%	8.250000	10290.500000	779.000000	741.250000
50%	15.500000	13183.000000	1225.000000	1049.017241
75%	22.750000	16885.500000	1652.500000	1503.500000
max	30.000000	19716.000000	1949.000000	1895.000000

```
#Compute for correlation coeffcient. Select the two variables the have the highest correlation (exclude days in the selection)
#your y is sales, select the appropriate x
df.corr()
```

	days	sales	pares_kawali	pares_flowers
days	1.000000	0.014854	0.206328	0.105899
sales	0.014854	1.000000	-0.191355	0.146615
pares_kawali	0.206328	-0.191355	1.000000	-0.003147
pares_flowers	0.105899	0.146615	-0.003147	1.000000

```
#generate a regression plot
sns.regplot(x="pares_kawali", y="pares_flowers", data=df, ci=None)
```

```
<Axes: xlabel='pares_kawali', ylabel='pares_flowers'>
```




```
#project the sales for the 32nd day using linear regression
#model: sales = intercept + slope * x
```