Importing Required libraries

In [21]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import statsmodels.formula.api as smf
```

Importing Data

In [2]:
```
calories = pd.read_csv("calories_consumed.csv")
calories
```

Out[2]:

	Weight gained (grams)	Calories Consumed
0	108	1500
1	200	2300
2	900	3400
3	200	2200
4	300	2500
5	110	1600
6	128	1400
7	62	1900
8	600	2800
9	1100	3900
10	100	1670
11	150	1900
12	350	2700
13	700	3000

Data understanding

In [3]:
```
calories.head()
```

Out[3]:

	Weight gained (grams)	Calories Consumed
0	108	1500
1	200	2300
2	900	3400
3	200	2200
4	300	2500

In [5]:
```
calories.shape
```

Out[5]:
```
(14, 2)
```

In [6]:
```
calories.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 2 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Weight gained (grams)  14 non-null     int64
 1   Calories Consumed      14 non-null     int64
dtypes: int64(2)
memory usage: 352.0 bytes
```

In [7]:
```
calories.isna().sum()
```

Out[7]:
```
Weight gained (grams)    0
Calories Consumed        0
dtype: int64
```

In [8]:
```
calories.dtypes
```

Out[8]:
```
Weight gained (grams)    int64
Calories Consumed        int64
dtype: object
```

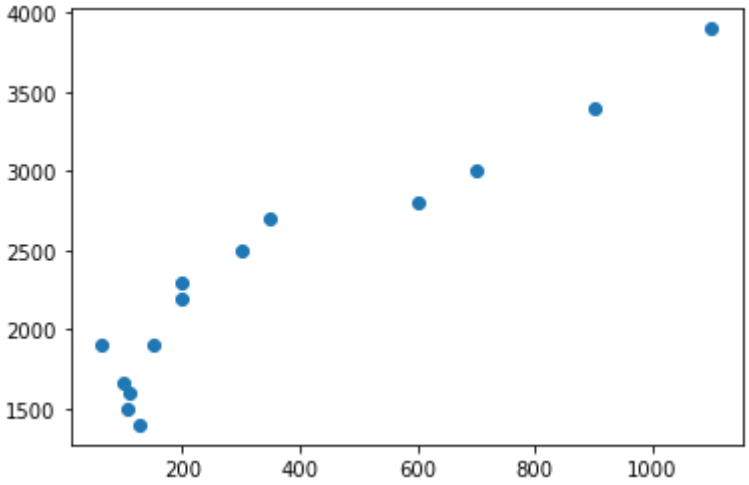
In [10]:
```
calories.describe()
```

Out[10]:

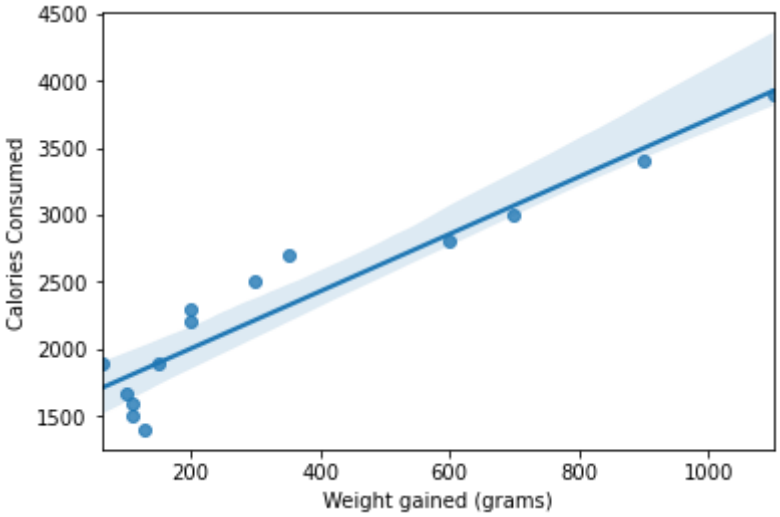
	Weight gained (grams)	Calories Consumed
count	14.000000	14.000000
mean	357.714286	2340.714286
std	333.692495	752.109488
min	62.000000	1400.000000
25%	114.500000	1727.500000
50%	200.000000	2250.000000
75%	537.500000	2775.000000
max	1100.000000	3900.000000

Checking Weater the assumptions are matching or not

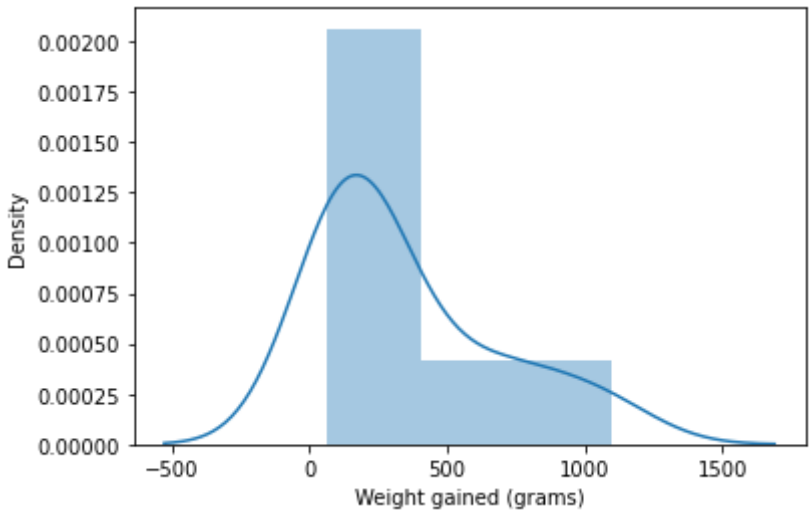
In [12]:
```
plt.scatter(x = 'Weight gained (grams)' , y = 'Calories Consumed', data = calories)
plt.show()
```



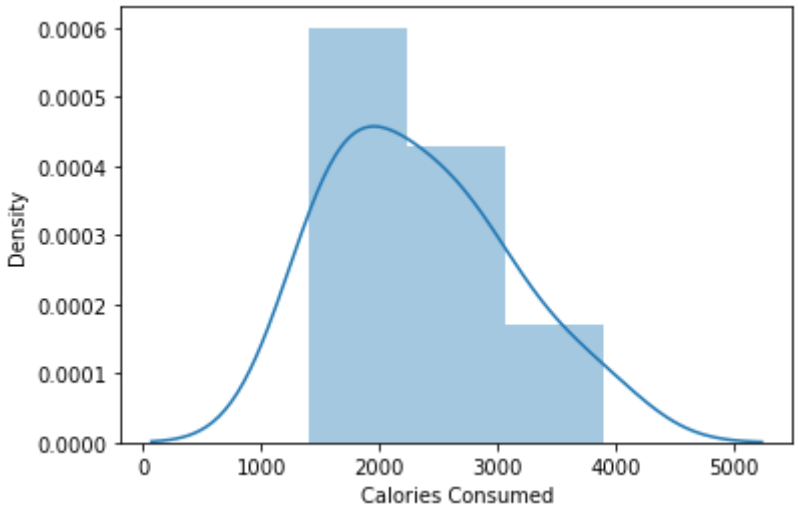
In [13]:
```
sns.regplot(x = 'Weight gained (grams)' , y = 'Calories Consumed' , data  = calories
plt.show()
```



In [18]:
```
sns.distplot(calories['Weight gained (grams)'])
plt.show()
import warnings
warnings.filterwarnings('ignore')
```



In [17]:
```
sns.distplot(calories['Calories Consumed'])
plt.show()
```



In [28]:
```
calories.corr()
```

Out[28]:

	Weight_gained_grams	Calories_Consumed
Weight_gained_grams	1.000000	0.946991
Calories_Consumed	0.946991	1.000000

Model Bulding

In [24]:
```
#Renaming columns names
calories = calories.rename({'Calories Consumed':'Calories_Consumed','Weight gained (
calories.head()
```

Out[24]:

	Weight_gained_grams	Calories_Consumed
0	108	1500
1	200	2300
2	900	3400
3	200	2200
4	300	2500

In [25]:
```
linear_model = smf.ols(formula = 'Calories_Consumed~Weight_gained_grams' , data  = c
linear_model
```

Out[25]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x209637a6640>
```

Model Testing

In [26]:
```
linear_model.params
```

Out[26]:
```
Intercept             1577.200702
Weight_gained_grams      2.134423
dtype: float64
```

In [27]:
```
linear_model.tvalues,linear_model.pvalues
```

Out[27]:
```
(Intercept              15.687195
 Weight_gained_grams    10.211269
 dtype: float64,
 Intercept              2.326102e-09
 Weight_gained_grams    2.855864e-07
 dtype: float64)
```

In [29]:
```
linear_model.rsquared,linear_model.rsquared_adj
```

Out[29]:
```
(0.8967919708530552, 0.8881913017574764)
```

Model Prediction

Sample Calculation

In [31]:
```
### y = mx+c
calories = (1577.200702+2.134423)*(5)
calories
```

Out[31]:
```
7896.675625
```

In [35]:
```
#machine prediction
pred_data = {'Weight_gained_grams':[100,200,300,400]}
pred_data
```

Out[35]:
```
{'Weight_gained_grams': [100, 200, 300, 400]}
```

In [36]:
```
new_data = pd.DataFrame(pred_data)
new_data
```

Out[36]:

	Weight_gained_grams
0	100
1	200
2	300
3	400

In [37]:
```
linear_model.predict(new_data)
```

Out[37]:
```
0    1790.642998
1    2004.085294
2    2217.527589
3    2430.969885
dtype: float64
```

In [ ]:
```

```