In [ ]:

```python

```

In [230]:

```python
# IMPORT LIBRARIES
import numpy  as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [231]:

```python
a=pd.read_csv(r"C:\Users\user\Downloads\14_Iris.csv")
a
```

Out[231]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [232]:

```
a=a.head(10)
a
```

Out[232]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [233]:

```
# to find
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             10 non-null     int64  
 1   SepalLengthCm  10 non-null     float64
 2   SepalWidthCm   10 non-null     float64
 3   PetalLengthCm  10 non-null     float64
 4   PetalWidthCm   10 non-null     float64
 5   Species        10 non-null     object 
dtypes: float64(4), int64(1), object(1)
memory usage: 608.0+ bytes
```

In [234]:

```
# to display summary of statastic
a.describe()
```

Out[234]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	10.00000	10.000000	10.000000	10.000000	10.000000
mean	5.50000	4.860000	3.310000	1.450000	0.220000
std	3.02765	0.291357	0.307137	0.108012	0.078881
min	1.00000	4.400000	2.900000	1.300000	0.100000
25%	3.25000	4.625000	3.100000	1.400000	0.200000
50%	5.50000	4.900000	3.300000	1.400000	0.200000
75%	7.75000	5.000000	3.475000	1.500000	0.200000
max	10.00000	5.400000	3.900000	1.700000	0.400000

In [235]:

```
# to display colum heading
a.columns
```

Out[235]:

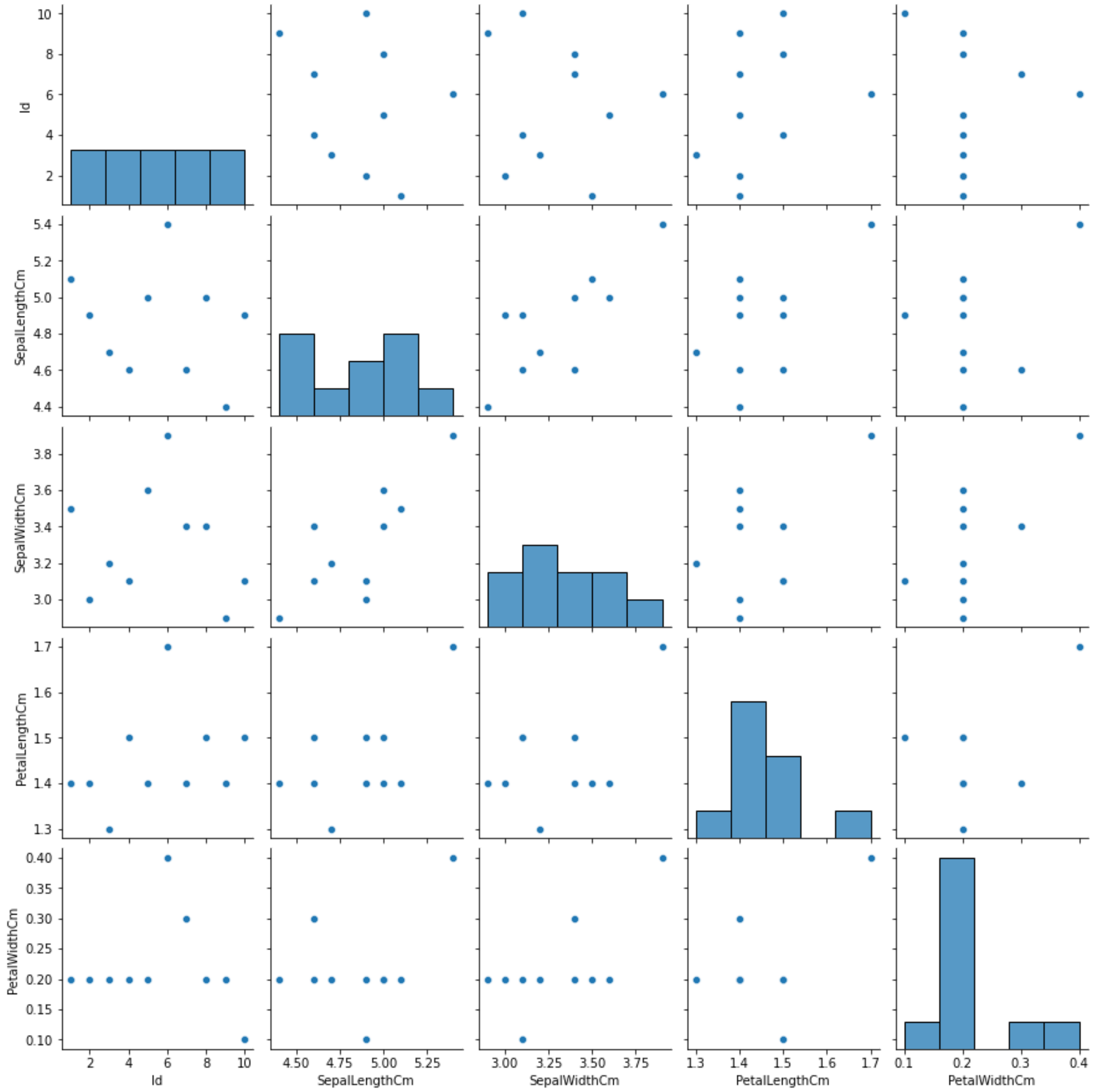
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidth
Cm',
       'Species'],
      dtype='object')
```

In [236]:

```
sns.pairplot(a)
```

Out[236]:

```
<seaborn.axisgrid.PairGrid at 0x198d38dd250>
```

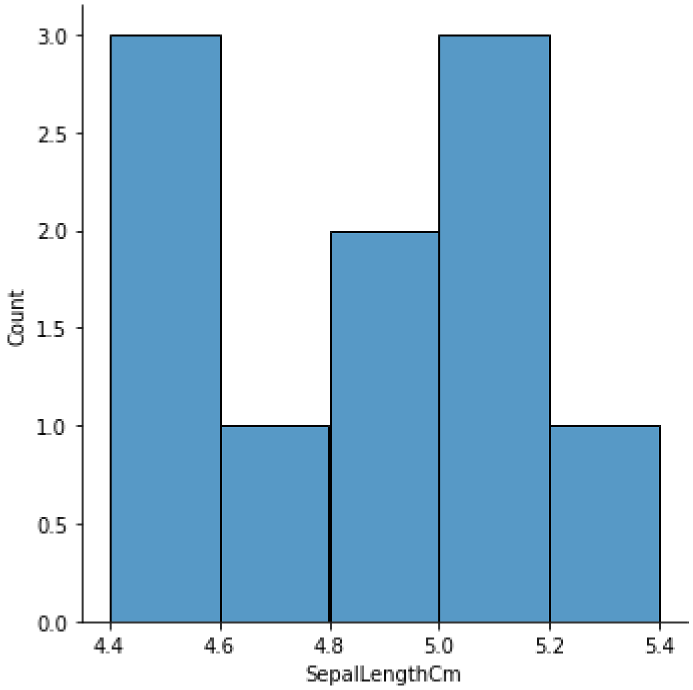


In [237]:

```
sns.displot(a["SepalLengthCm"])
```

Out[237]:

```
<seaborn.axisgrid.FacetGrid at 0x198d38f87f0>
```



In [238]:

```
b=a[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
b
```

Out[238]:

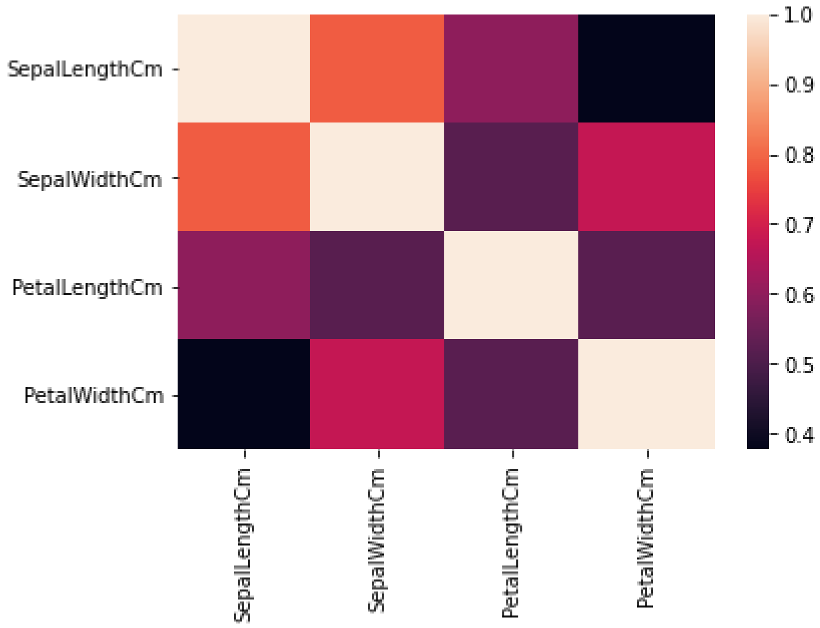
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

In [239]:

```
sns.heatmap(b.corr())
```

Out[239]:

```
<AxesSubplot:>
```



In [241]:

```
x=a[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y=a['SepalLengthCm']
```

In [242]:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [243]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[243]:

```
LinearRegression()
```

In [244]:

```
lr.intercept_
```

Out[244]:

```
-2.6645352591003757e-15
```

In [245]:

```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[245]:

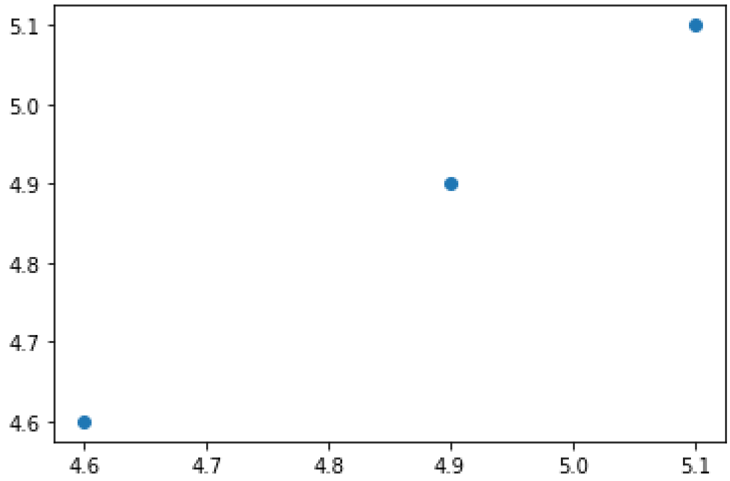
	Co-efficient
SepalLengthCm	1.000000e+00
SepalWidthCm	-6.196248e-16
PetalLengthCm	-1.300470e-16
PetalWidthCm	-3.648064e-17

In [246]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[246]:

```
<matplotlib.collections.PathCollection at 0x198d58c4310>
```



In [247]:

```
lr.score(x_test,y_test)
```

Out[247]:

```
1.0
```

In [248]:

```
lr.score(x_train,y_train)
```

Out[248]:

```
1.0
```

In [249]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [250]:

```
rr=Ridge(alpha=10)
rr.fit(x_test,y_test)
```

Out[250]:

```
Ridge(alpha=10)
```

In [251]:

```
rr.score(x_test,y_test)
```

Out[251]:

```
0.02651347184492392
```

In [252]:

```
la=Lasso(alpha=10)
la.fit(x_test,y_test)
```

Out[252]:

```
Lasso(alpha=10)
```

In [253]:

```
la.score(x_test,y_test)
```

Out[253]:

```
0.0
```

In [254]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[254]:

```
ElasticNet()
```

In [255]:

```
en.coef_
```

Out[255]:

```
array([0., 0., 0., 0.])
```

In [256]:

```
en.intercept_
```

Out[256]:

```
4.857142857142857
```

In [257]:

```
prediction=en.predict(x_test)
prediction
```

Out[257]:

```
array([4.85714286, 4.85714286, 4.85714286])
```

In [258]:

```
en.score(x_test,y_test)
```

Out[258]:

```
-0.0021482277121378512
```

EVALUATION METRICS

In [259]:

```
from sklearn import metrics
```

In [260]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.1809523809523812
```

In [261]:

```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 0.04231292517006805
```

In [262]:

```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error 0.20570105777576364
```

In [ ]:

```

```