In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("14_Iris.csv")
df
```

Out[2]:

	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 [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

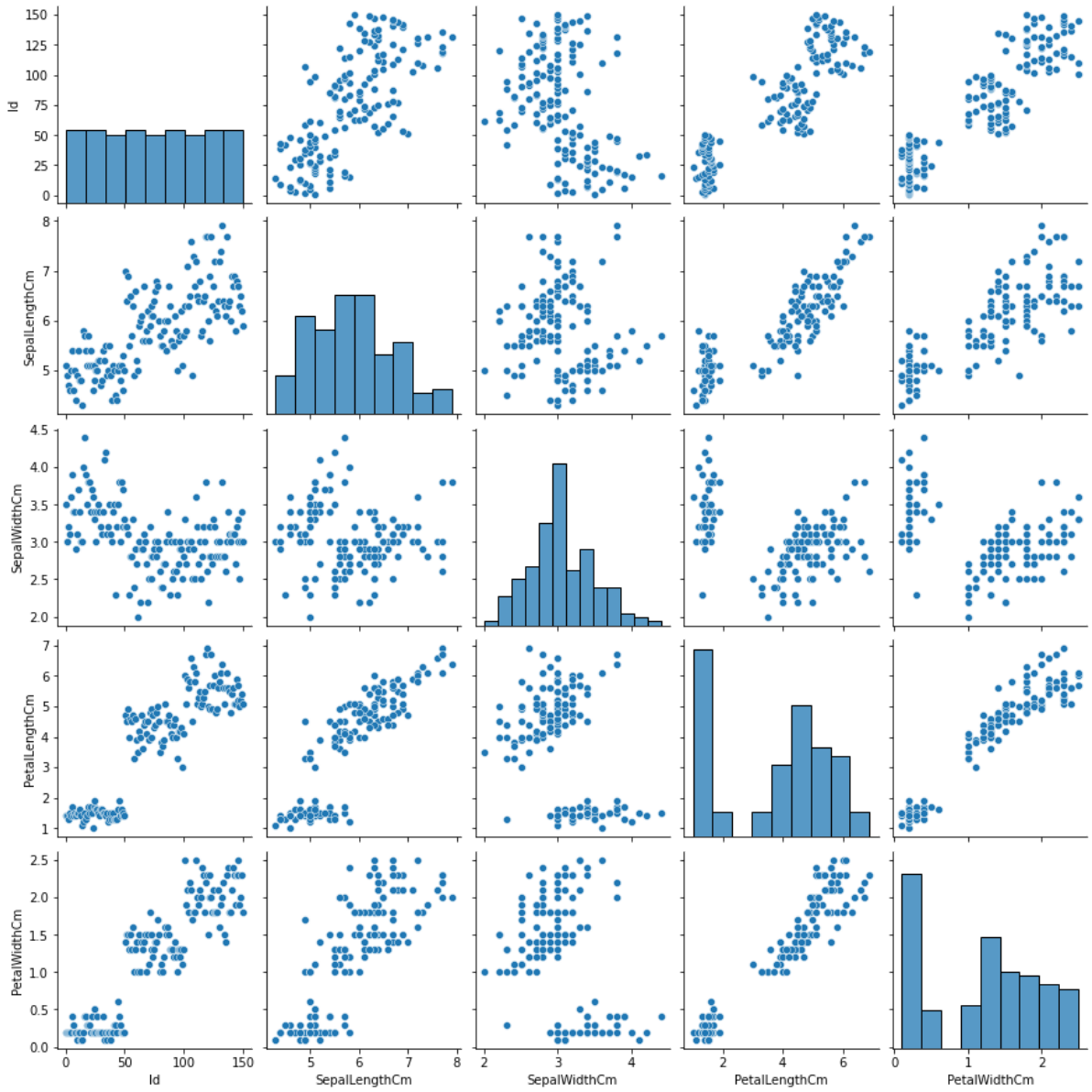
In [4]:
```
df.describe()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

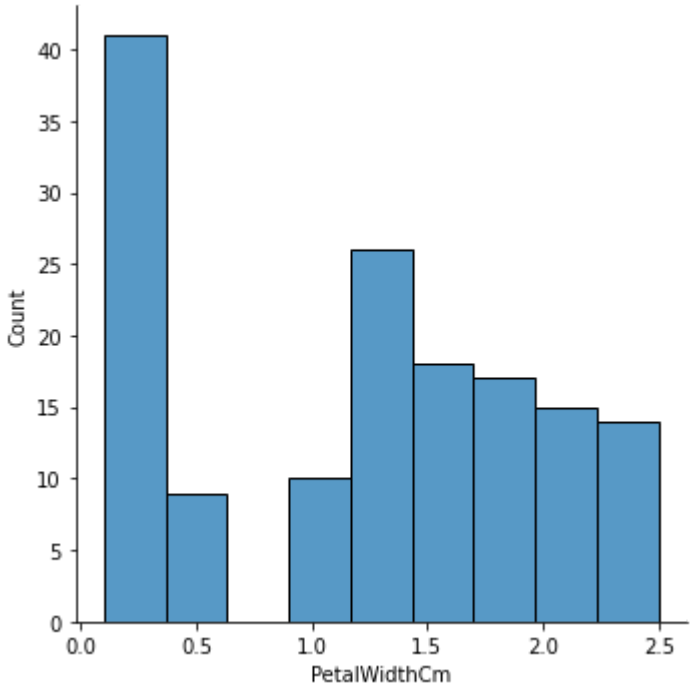
In [5]:
```
sns.pairplot(df)
```

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x1dc4a902400>
```



In [6]:
```
sns.displot(df['PetalWidthCm'])
```

Out[6]:
```
<seaborn.axisgrid.FacetGrid at 0x1dc4cc55640>
```



In [7]:
```
df1=df.drop(['Species'],axis=1)
df1
```

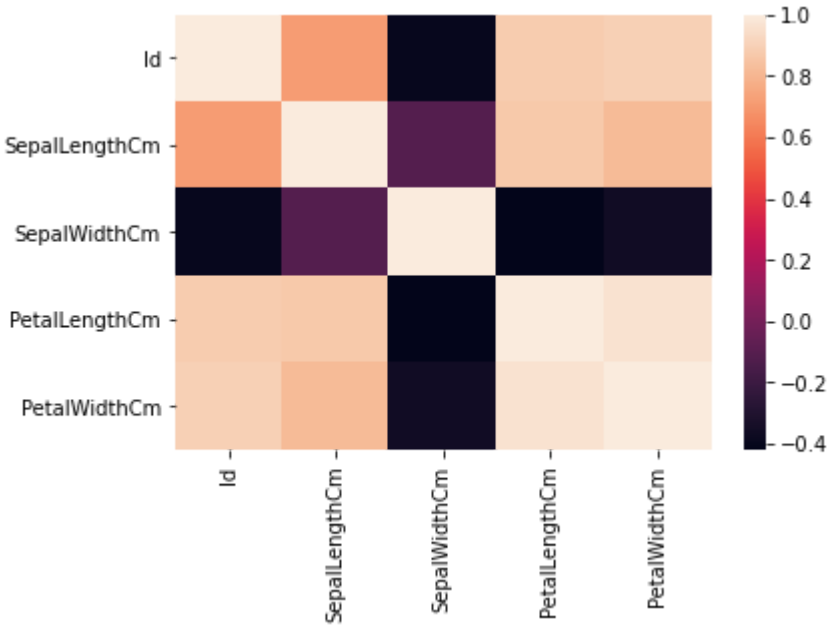
Out[7]:

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

150 rows × 5 columns

In [8]:
```
sns.heatmap(df1.corr())
```

Out[8]: <AxesSubplot:>



In [9]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [10]:
```
y=df['PetalWidthCm']
x=df1.drop(['PetalWidthCm','Id'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm
42             4.4           3.2            1.3
60             5.0           2.0            3.5
15             5.7           4.4            1.5
148            6.2           3.4            5.4
71             6.1           2.8            4.0
..             ...           ...            ...
103            6.3           2.9            5.6
118            7.7           2.6            6.9
70             5.9           3.2            4.8
145            6.7           3.0            5.2
78             6.0           2.9            4.5

[105 rows x 3 columns]
```

In [11]:
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[11]: -0.20753362311165136

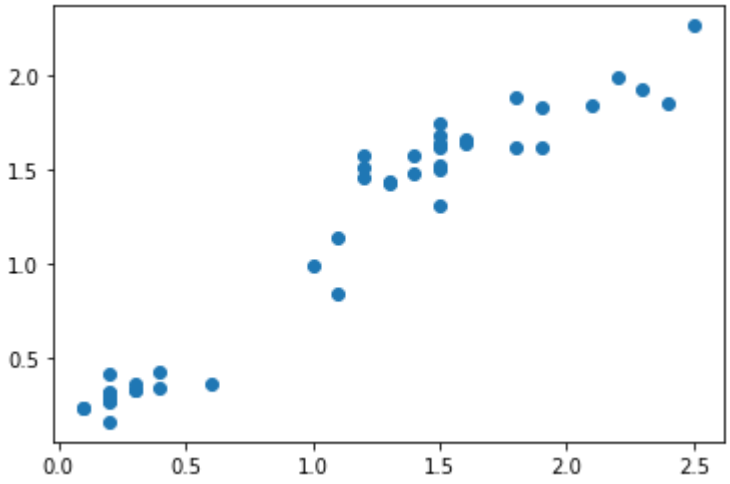
In [12]:
```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

Out[12]:

	Coefficient
SepalLengthCm	-0.223779
SepalWidthCm	0.243197
PetalLengthCm	0.525620

In [13]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[13]:
```
<matplotlib.collections.PathCollection at 0x1dc4d25de20>
```



In [14]:
```
model.score(x_test,y_test)
```

Out[14]:
```
0.935029363293646
```

In [15]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [16]:
```
rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[16]:
```
Ridge(alpha=10)
```

In [17]:
```
rr.score(x_test,y_test)
```

Out[17]:
```
0.9265166411152366
```

In [18]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[18]:
```
Lasso(alpha=10)
```

In [19]:
```
la.score(x_test,y_test)
```

Out[19]:
```
-0.04442071186610397
```

In [20]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
[ 0.         -0.          0.22233972]
0.38356605183291725
[1.36186082 1.73983835 0.69484166 0.73930961 0.71707563 0.69484166
 0.76154358 0.69484166 0.69484166 1.51749863 1.3840948  1.22845699
 1.45079671 0.76154358 1.51749863 1.42856274 1.40632877 1.42856274
 1.47303069 0.73930961 1.29515891 0.73930961 1.58420055 1.51749863
 1.3840948  1.56196657 1.3840948  0.73930961 0.69484166 1.16175508
 1.62866849 0.71707563 1.31739288 1.05058522 0.71707563 0.71707563
 0.71707563 1.49526466 0.71707563 1.3840948  1.42856274 1.49526466
 1.40632877 1.51749863 1.3840948 ]
0.6985305223464584
Mean Absolute Error: 0.1469936944918236
Mean Squared Error: 0.034530369406837694
Root Mean Squared Error: 0.18582348992212394
```

In [ ]: