In [363]:

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

In [364]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\4_drug200.csv")
a
```

Out[364]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

In [365]:

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

Out[365]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
5	22	F	NORMAL	HIGH	8.607	drugX
6	49	F	NORMAL	HIGH	16.275	drugY
7	41	M	LOW	HIGH	11.037	drugC
8	60	M	NORMAL	HIGH	15.171	drugY
9	43	M	LOW	NORMAL	19.368	drugY

In [366]:

```
# 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   Age          10 non-null     int64  
 1   Sex          10 non-null     object 
 2   BP           10 non-null     object 
 3   Cholesterol  10 non-null     object 
 4   Na_to_K      10 non-null     float64
 5   Drug         10 non-null     object 
dtypes: float64(1), int64(1), object(4)
memory usage: 608.0+ bytes
```

In [367]:

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

Out[367]:

	Age	Na_to_K
count	10.000000	10.000000
mean	42.100000	14.486100
std	13.916018	5.482634
min	22.000000	7.798000
25%	31.250000	10.344750
50%	45.000000	14.132000
75%	48.500000	17.601000
max	61.000000	25.355000

In [368]:

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

Out[368]:

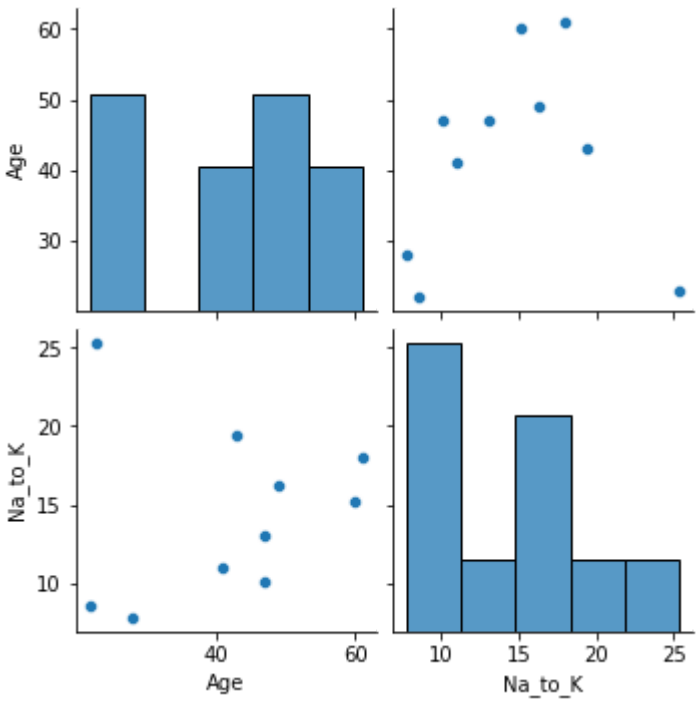
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='objec
t')
```

In [369]:

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

Out[369]:

```
<seaborn.axisgrid.PairGrid at 0x198e3344f40>
```

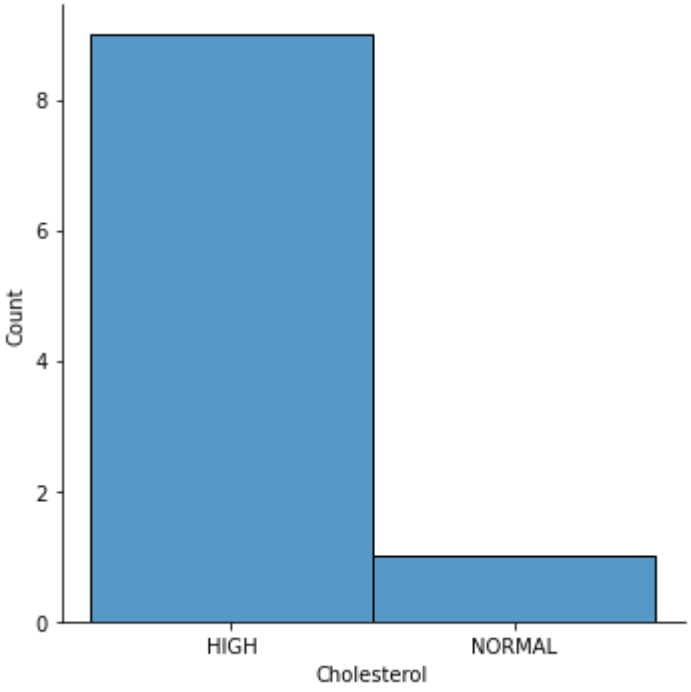


In [370]:

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

Out[370]:

```
<seaborn.axisgrid.FacetGrid at 0x198e38a7e80>
```



In [371]:

```
b=a[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']]
b
```

Out[371]:

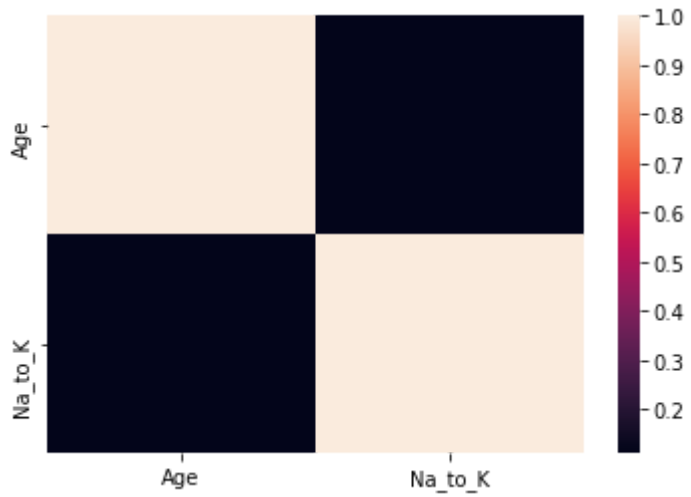
	Age	Sex	BP	Cholesterol	Na_to_K
0	23	F	HIGH	HIGH	25.355
1	47	M	LOW	HIGH	13.093
2	47	M	LOW	HIGH	10.114
3	28	F	NORMAL	HIGH	7.798
4	61	F	LOW	HIGH	18.043
5	22	F	NORMAL	HIGH	8.607
6	49	F	NORMAL	HIGH	16.275
7	41	M	LOW	HIGH	11.037
8	60	M	NORMAL	HIGH	15.171
9	43	M	LOW	NORMAL	19.368

In [372]:

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

Out[372]:

<AxesSubplot:>



In [374]:

```
x=a[['Age', 'Na_to_K']]
y=a['Age']
```

In [375]:

```
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 [376]:

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

Out[376]:

```
LinearRegression()
```

In [377]:

```
lr.intercept_
```

Out[377]:

```
-2.1316282072803006e-14
```

In [378]:

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

Out[378]:

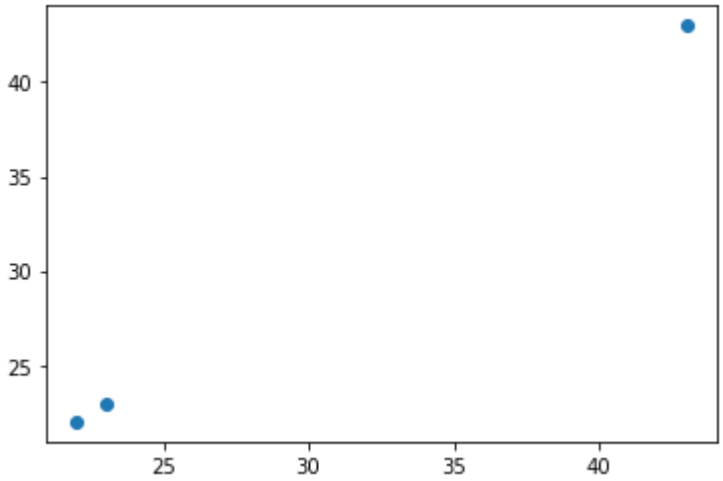
	Co-efficient
Age	1.000000e+00
Na_to_K	-6.427836e-17

In [379]:

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

Out[379]:

```
<matplotlib.collections.PathCollection at 0x198e3a84c40>
```



In [380]:

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

Out[380]:

```
1.0
```

In [381]:

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

Out[381]:

```
1.0
```

In [382]:

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

In [383]:

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

Out[383]:

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

In [384]:

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

Out[384]:

```
0.9987751603410064
```

In [385]:

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

Out[385]:

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

In [386]:

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

Out[386]:

```
0.9885748782730858
```

In [387]:

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

Out[387]:

```
ElasticNet()
```

In [388]:

```
en.coef_
```

Out[388]:

```
array([0.99087608, 0.        ])
```

In [389]:

```
en.intercept_
```

Out[389]:

```
0.434037799087605
```

In [390]:

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

Out[390]:

```
array([43.04170934, 23.22418769, 22.23331161])
```

In [391]:

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

Out[391]:

```
0.9996207811257799
```

EVALUATION METRICS

In [392]:

```
from sklearn import metrics
```

In [393]:

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

```
Mean Absolute Error: 0.16640287993048256
```

In [394]:

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

```
Mean Squared Error 0.03547803245481273
```

In [395]:

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

```
Root Mean Squared Error 0.18835613198091727
```

In [ ]:

```

```