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("4_drug.csv")
df
```

Out[2]:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    object 
 3   Cholesterol  200 non-null    object 
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

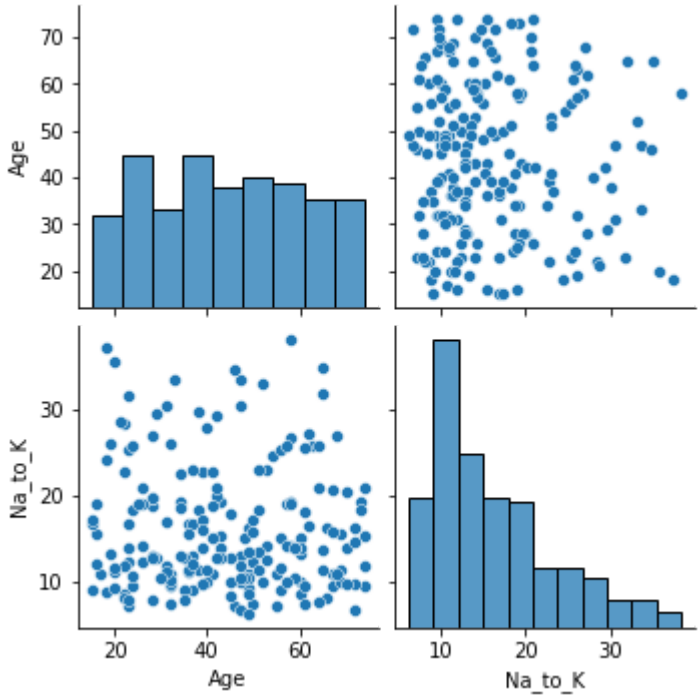
In [4]:
```
df.describe()
```

Out[4]:

	Age	Na_to_K
count	200.000000	200.000000
mean	44.315000	16.084485
std	16.544315	7.223956
min	15.000000	6.269000
25%	31.000000	10.445500
50%	45.000000	13.936500
75%	58.000000	19.380000
max	74.000000	38.247000

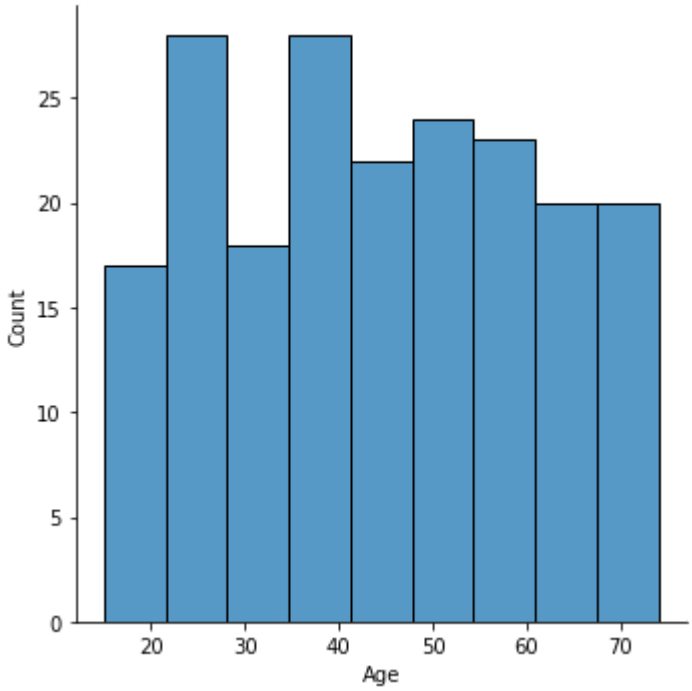
In [5]:
```
sns.pairplot(df)
```

Out[5]:
```
<seaborn.axisgrid.PairGrid at 0x1bed1a34af0>
```



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

Out[6]:
```
<seaborn.axisgrid.FacetGrid at 0x1bed1c53bb0>
```



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

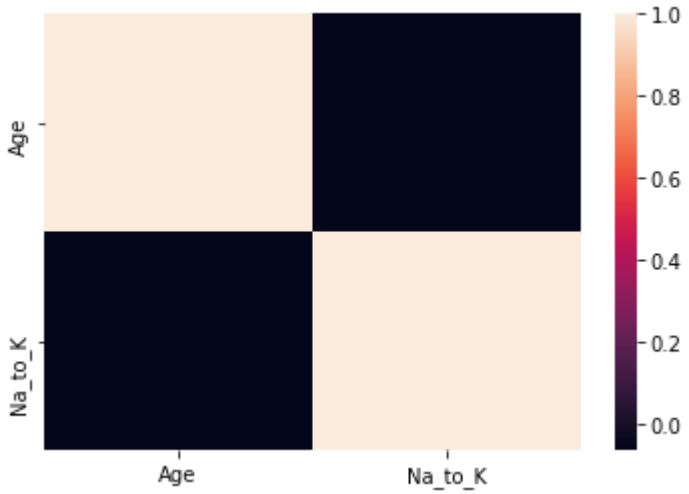
Out[7]:

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

200 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['Age']
x=df1.drop(['Age','BP','Cholesterol','Drug'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

```
     Na_to_K
62    20.693
156   11.227
187   10.403
41    14.239
118   10.292
..       ...
193    6.769
71    19.675
93    29.271
57    27.826
195   11.567

[140 rows x 1 columns]
```

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

Out[11]: 47.90162037012079

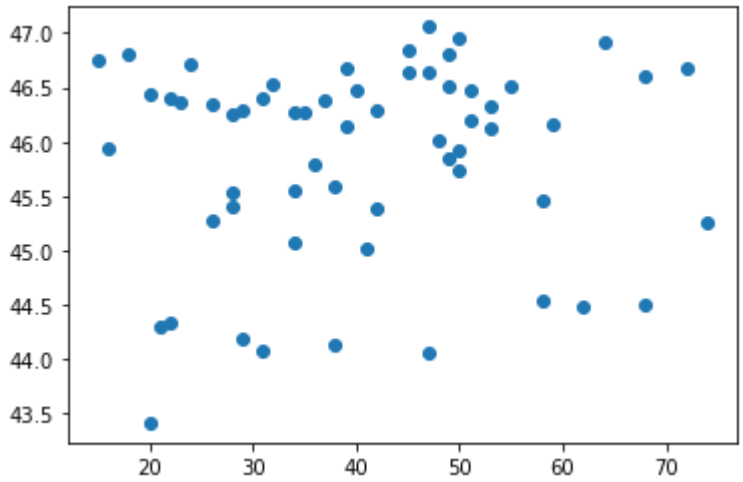
In [12]:
```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

Out[12]:

	Coefficient
Na_to_K	-0.125986

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

Out[13]:
```
<matplotlib.collections.PathCollection at 0x1bed360dee0>
```



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

Out[14]:
```
-0.11864238220653989
```

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]:
```
-0.11864770796837432
```

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]:
```
-0.12627217665641743
```

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
```

```
[-0.1153991]
47.73161475153885
[46.31682177 46.46487882 44.22740564 44.61006906 45.08839834 46.57566195
 46.12318208 46.36171202 46.76572427 45.99647387 45.90946294 46.86727548
 46.5440426  46.42264275 46.10067925 46.61489765 44.6568057  46.31139801
 45.4910258  46.16253317 46.68332932 45.85349438 45.56107306 45.9410823
 46.28970298 45.57330536 46.24365874 43.61890618 46.63820827 44.59472098
 46.35224929 46.72187262 46.96040256 44.2840666  45.74548082 46.83600233
 45.44717414 46.12941363 46.21677075 44.20409503 46.46060905 46.569892
 46.42195035 46.24804391 46.61120488 46.48068849 45.31873494 45.6203882
 44.42750769 44.33311122 45.14021254 45.79833361 46.25842983 46.38306085
 46.72764257 46.24031217 45.42213254 44.46651258 46.34613314 45.31492677]
-0.11899860556912256
Mean Absolute Error: 12.810936451640638
Mean Squared Error: 240.64483513223132
Root Mean Squared Error: 15.512731388515412
```