In [1]:

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

In [2]:

```
df=pd.read_csv(r'C:\Users\user\Downloads\4_drug200.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.head(10)
```

Out[3]:

	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 [4]:

```
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 [5]:

```
df.describe()
```

Out[5]:

	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 [6]:

```
df.columns
```

Out[6]:

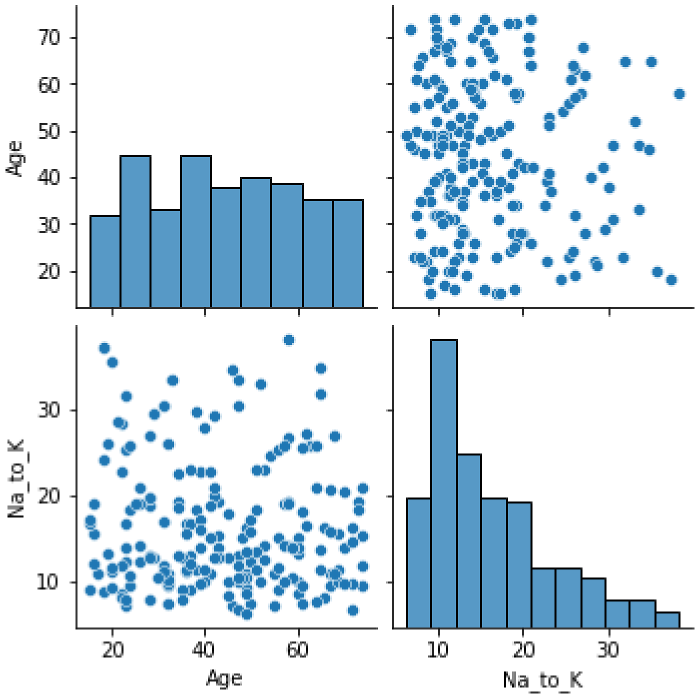
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='objec
t')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1df05a23c40>
```



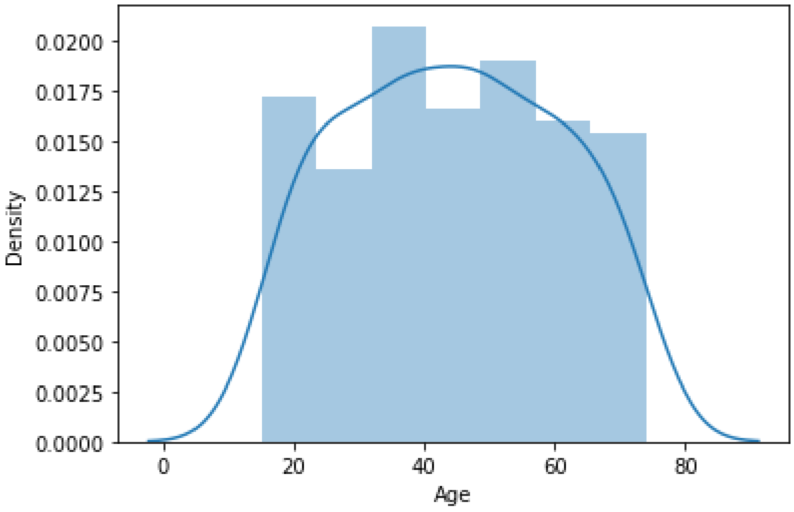
In [8]:

```
sns.distplot(df['Age'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```

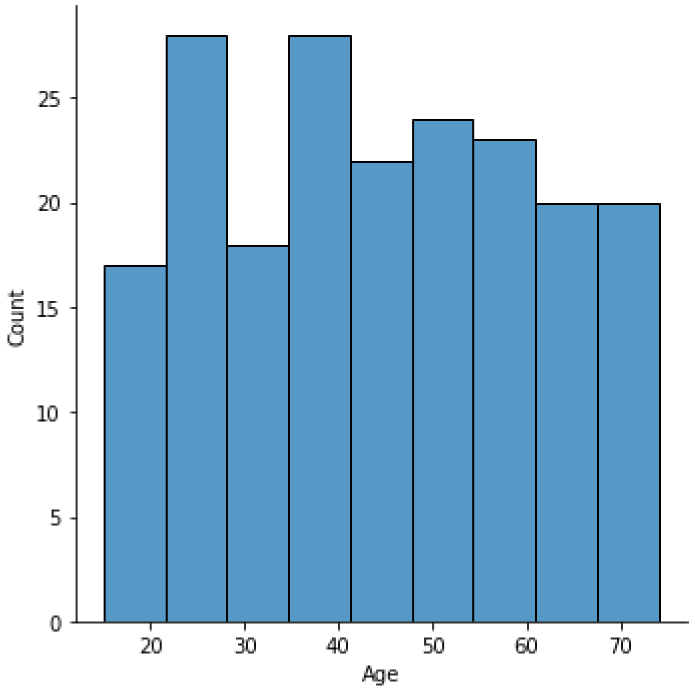


In [9]:

```
sns.displot(df["Age"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1df07426460>
```



In [10]:

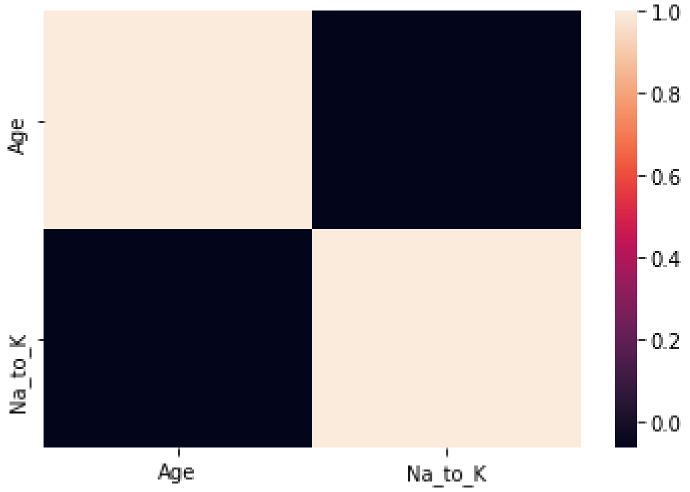
```
df1=df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']]
```

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
x=df1[['Age', 'Na_to_K']]
y=df1[['Age']]
```

In [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

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

In [15]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
print(lr.intercept_)
```

```
[-7.10542736e-15]
```

In [17]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[17]:

	0	1
0	1.0	2.176276e-17

In [18]:

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

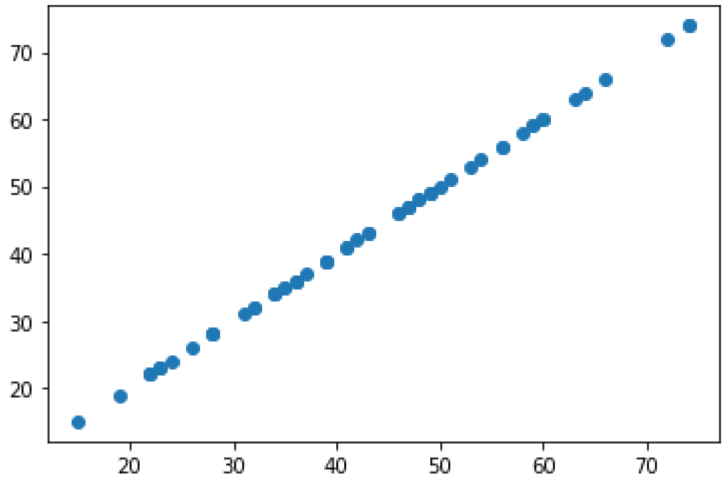
```
1.0
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x1df07d5d280>
```



In [20]:

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

Out[20]:

```
1.0
```

In [21]:

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

Out[21]:

```
1.0
```

In [22]:

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

In [23]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[23]:

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

In [24]:

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

Out[24]:

```
0.9999999401807266
```

In [25]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]:

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

In [26]:

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

Out[26]:

```
0.998831182319078
```

Elastic Net

In [27]:

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

Out[27]:

```
ElasticNet()
```

In [28]:

```
print(en.coef_)
```

```
[ 0.9966435 -0.        ]
```

In [29]:

```
print(en.intercept_)
```

```
[0.15142589]
```

In [30]:

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

```
[28.05744402 46.99367061 32.04401804 36.03059206 34.03730505 71.90975822
 23.0742265  32.04401804 47.99031411 53.97017514 34.03730505 43.00709659
 48.98695762 19.08765248 39.02052257 55.96346215 28.05744402 41.01380958
 23.0742265  58.95339266 48.98695762 34.03730505 46.99367061 42.01045309
 22.07758299 39.02052257 45.9970271  35.03394855 50.98024463 59.95003617
 63.93661019 57.95674916 22.07758299 36.03059206 43.00709659 45.9970271
 46.99367061 49.98360112 58.95339266 39.02052257 35.03394855 59.95003617
 37.02723556 47.99031411 22.07758299 28.05744402 52.97353164 24.07087
 15.10107846 62.93996668 41.01380958 73.90304523 48.98695762 65.9298972
 31.04737454 55.96346215 73.90304523 26.06415701 23.0742265  36.03059206]
```

In [31]:

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

```
0.9999883510215829
```

Evaluation Metrics

In [32]:

```
from sklearn import metrics
```

In [33]:

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

```
Mean Absolute Error: 0.041043545847445954
```

In [34]:

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

```
Mean Squared Error: 0.0024328600199539493
```

In [35]:

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

```
Root Mean Squared Error: 0.049324030856712726
```