In [1]:

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

DATA COLLECTION

In [2]:

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

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]:

```
b=a.head(100)
b
```

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
...	...	...	...	...	...	...
95	36	M	LOW	NORMAL	11.424	drugX
96	58	F	LOW	HIGH	38.247	drugY
97	56	F	HIGH	HIGH	25.395	drugY
98	20	M	HIGH	NORMAL	35.639	drugY
99	15	F	HIGH	NORMAL	16.725	drugY

100 rows × 6 columns

DATA CLEANING AND PRE-PROCESSING

In [4]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          100 non-null    int64  
 1   Sex          100 non-null    object 
 2   BP           100 non-null    object 
 3   Cholesterol  100 non-null    object 
 4   Na_to_K      100 non-null    float64
 5   Drug         100 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 4.8+ KB
```

In [5]:

```
b.describe()
```

Out[5]:

	Age	Na_to_K
count	100.000000	100.000000
mean	43.770000	16.823000
std	16.367531	7.257723
min	15.000000	7.285000
25%	30.500000	11.031250
50%	43.000000	15.025500
75%	58.000000	20.020250
max	74.000000	38.247000

In [6]:

```
b.columns
```

Out[6]:

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

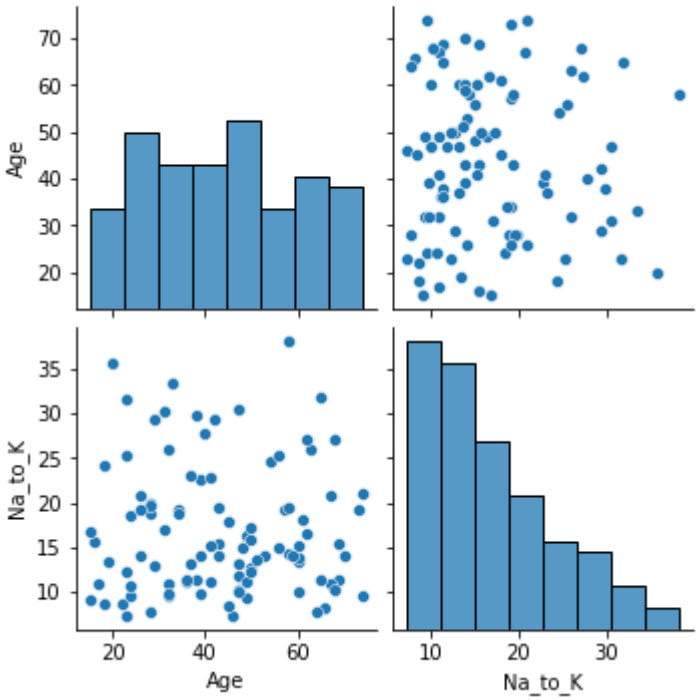
EDA AND VISUALIZATION

In [7]:

```
sns.pairplot(b)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x2365ace1c10>
```



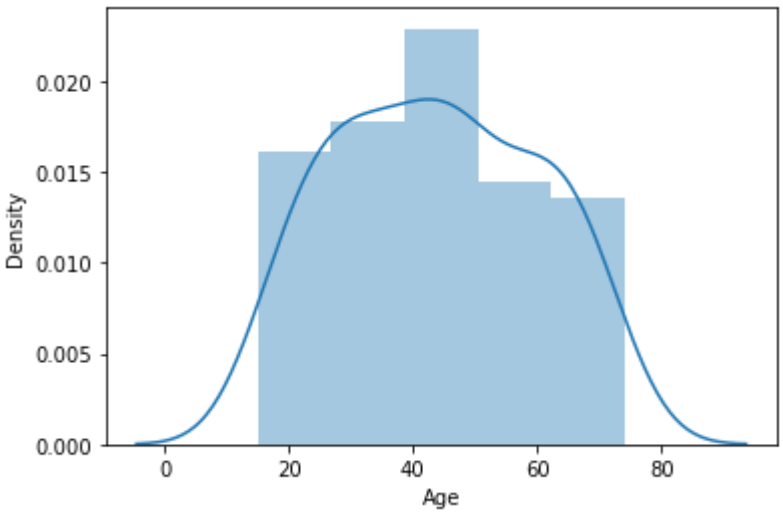
In [8]:

```
sns.distplot(b['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]:

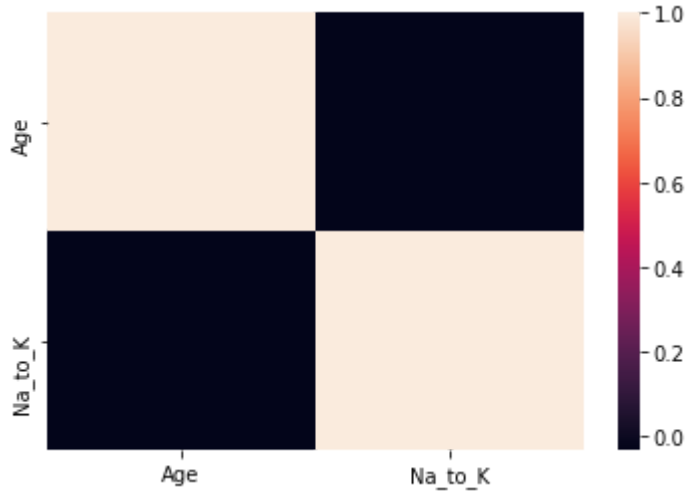
```
f=b[['Age','Na_to_K']]
```

In [10]:

```
sns.heatmap(f.corr())
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
x=f[['Age']]
y=f['Na_to_K']
```

In [12]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [13]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:

```
LinearRegression()
```

In [14]:

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

```
16.933706448803974
```

In [15]:

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

Out[15]:

	Co-efficient
Age	0.013538

In [16]:

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

Out[16]:

```
0.0008077154273242737
```

In [17]:

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

```
-0.04375868771803715
```

RIDGE REGRESSION

In [18]:

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

In [19]:

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

Out[19]:

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

In [20]:

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

Out[20]:

```
-0.043752605407668455
```

LASSO REGRESSION

In [21]:

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

Out[21]:

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

In [22]:

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

Out[22]:

```
-0.038172495936848794
```

In [23]:

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

Out[23]:

```
ElasticNet()
```

In [24]:

```
print(p.coef_)
```

```
[0.01122555]
```

In [25]:

```
print(p.intercept_)
```

```
17.037022221451593
```

In [26]:

```
print(p.predict(x_test))
```

```
[17.81158549 17.28398442 17.59829995 17.32888664 17.29520998 17.30643553
 17.35133775 17.78913438 17.86771326 17.7105555  17.80035993 17.78913438
 17.20540554 17.85648771 17.5870744  17.77790882 17.59829995 17.68810439
 17.5870744  17.7105555  17.29520998 17.76668327 17.30643553 17.32888664
 17.40746552 17.35133775 17.29520998 17.72178105 17.54217218 17.68810439
 17.39623997 17.50849551 17.82281104 17.35133775 17.20540554 17.49726996
 17.26153331 17.49726996 17.66565328 17.56462329 17.2390822  17.67687883
 17.49726996 17.51972107 17.45236774 17.41869108 17.41869108 17.59829995
 17.22785665 17.44114219]
```

In [27]:

```
prediction=p.predict(x_test)
print(p.score(x_test,y_test))
```

```
-0.042652906869034224
```

In [28]:

```
from sklearn import metrics
```

In [29]:

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

```
Mean Absolytre Error: 5.951220854106817
```

In [30]:

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

```
Mean Squared Error: 55.94555412533408
```

In [31]:

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

```
Root Mean Squared Error: 7.47967607088262
```

In [ ]:

```

```