In [1]:

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

In [2]:

```
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\4_drug200.csv")
```

In [3]:

```
data.head()
```

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

In [4]:

```
data.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]:

```
data.shape
```

Out[5]:

```
(200, 6)
```

In [6]:
```
data.describe()
```

Out[6]:

	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

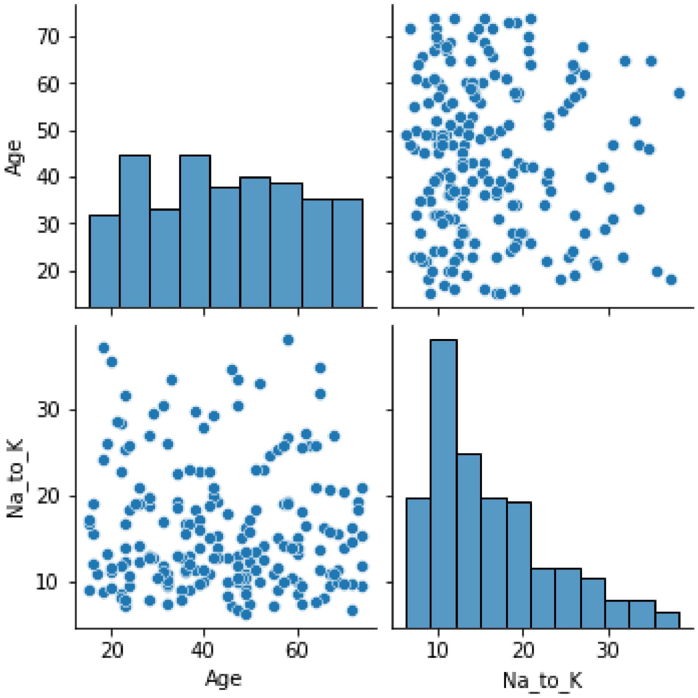
In [7]:
```
data.columns
```

Out[7]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

In [9]:
```
data1=data[['Age','Na_to_K']]
```

In [10]:
```
sns.pairplot(data1)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x223b20ba9a0>
```

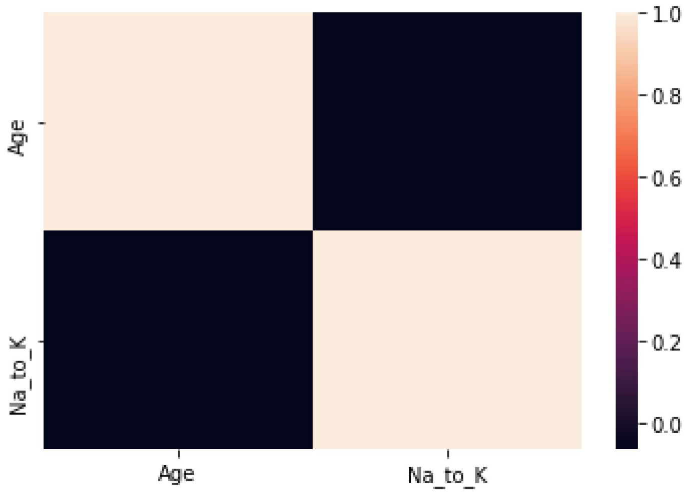


In [11]:

```
sns.heatmap(data1.corr())
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
x=data[['Age','Na_to_K']]
y=data1['Age']
```

In [13]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1)
```

In [14]:

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

Out[14]:

```
LinearRegression()
```

In [15]:

```
lr.intercept_
```

Out[15]:

```
7.105427357601002e-15
```

In [16]:

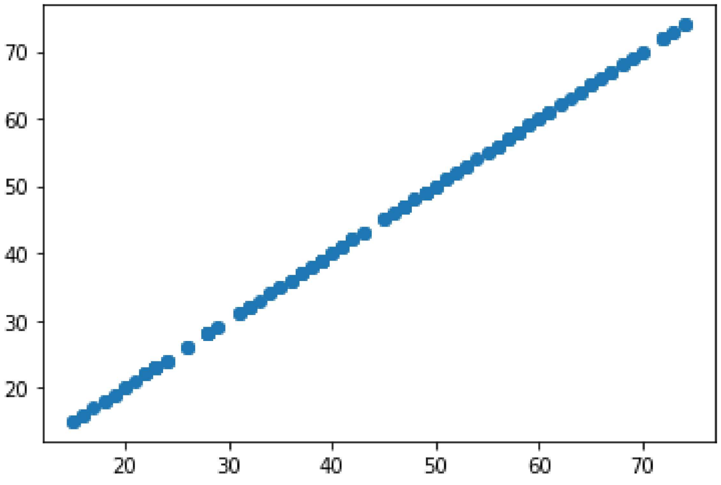
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[16]:

	Co-efficient
Age	1.000000e+00
Na_to_K	1.205361e-17

In [17]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x223b44d8610>
```



In [18]:
```
lr.score(x_test,y_test)
```

Out[18]:
```
1.0
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]:
```
1.0
```

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[21]:
```
0.9999999586455789
```

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[22]:
```
0.9986414514540869
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en= ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

In [24]:
```
print(en.coef_)
```

```
[ 0.99632325 -0.        ]
```

In [26]:
```
print(en.intercept_)
```

```
0.16271672981450536
```

In [27]:
```
prediction = en.predict(x_test)
prediction
```

Out[27]:
```
array([20.08918168, 50.97520235, 26.06712116, 43.00461637, 34.03770714,
       30.05241415, 59.94211157, 25.07079791, 73.89063704, 44.99726286,
       67.91269755, 49.9788791 , 59.94211157, 34.03770714, 69.90534405,
       46.98990936, 32.04506065, 31.0487374 , 68.9090208 , 28.05976766])
```

In [28]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-28-48d1f0543252> in <module>
----> 1 print(en.score(x_test,y_train))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in score(self, X, 
y, sample_weight)
    552         from .metrics import r2_score
    553         y_pred = self.predict(X)
--> 554         return r2_score(y, y_pred, sample_weight=sample_weight)
    555 
    556     def _more_tags(self):

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inn
er_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in 
r2_score(y_true, y_pred, sample_weight, multioutput)
    674     -3.0
    675     """
--> 676     y_type, y_true, y_pred, multioutput = _check_reg_targets(
    677         y_true, y_pred, multioutput)
    678     check_consistent_length(y_true, y_pred, sample_weight)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.py in 
_check_reg_targets(y_true, y_pred, multioutput, dtype)
     86         the dtype argument passed to check_array.
     87     """
---> 88     check_consistent_length(y_true, y_pred)
     89     y_true = check_array(y_true, ensure_2d=False, dtype=dtype)
     90     y_pred = check_array(y_pred, ensure_2d=False, dtype=dtype)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in che
ck_consistent_length(*arrays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent num
bers of"
    263                          " samples: %r" % [int(l) for l in lengths])
    264 

ValueError: Found input variables with inconsistent numbers of samples: [180, 
20]
```

In [29]:

```
from sklearn import metrics
```

```
In [30]: print("Mean Absolute error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error: 0.05349675061312968
```

```
In [31]: print("Mean Absolute Square error:",metrics.mean_squared_error(y_test,predicti
```

```
Mean Absolute Square error: 0.0037876277653470‌73
```

```
In [32]: print("Root mean Square error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root mean Square error: 0.06154370613919‌0815
```

```
In [ ]:
```