# Deena 20104016

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
```

# Problem Statement

# LINEAR REGRESSION

In [2]:
```
a = pd.read_csv("drug.csv")
```

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

# HEAD

In [3]:

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 |

# Data Cleaning and Preprocessing

In [4]:

Out[4]:

| | 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 [5]:

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 |

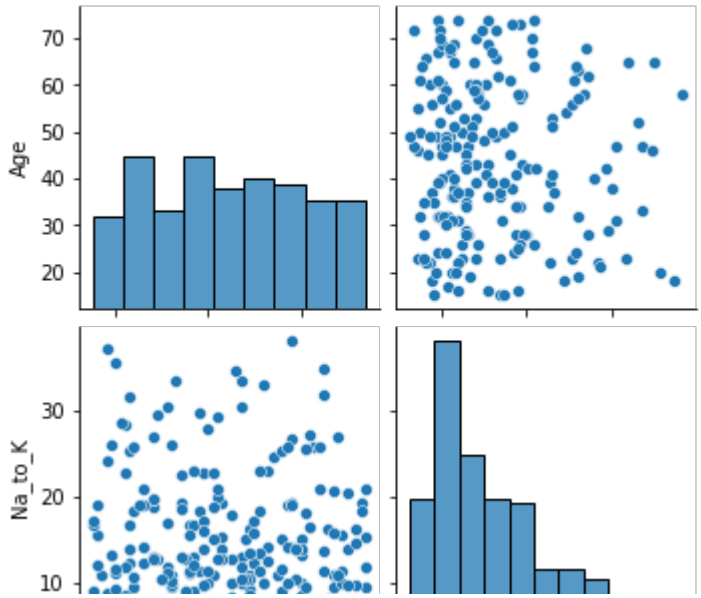
# To display heading

In [6]:

Out[6]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

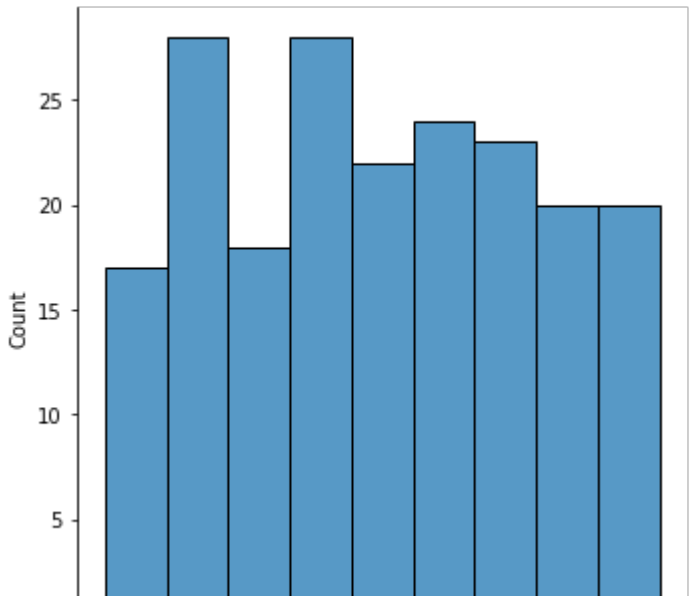
In [7]:

Out[7]: <seaborn.axisgrid.PairGrid at 0x1ab1b4a4be0>



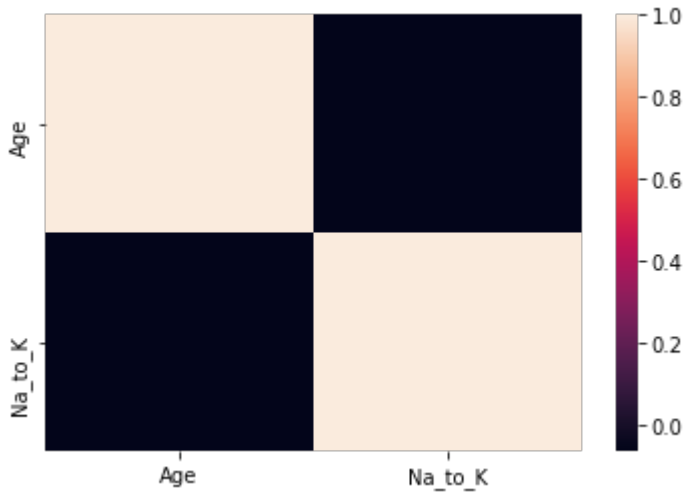

In [8]:

Out[8]: <seaborn.axisgrid.FacetGrid at 0x1ab1bd6e880>



In [9]:

Out[9]: <AxesSubplot:>




# TO TRAIN THE MODEL - MODEL BUILDING

In [10]:
```
x = a[['Age']]
```

In [11]:
```
# to split my dataset into training and test data
from sklearn.model_selection import train_test_split
```

In [12]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

Out[12]: LinearRegression()

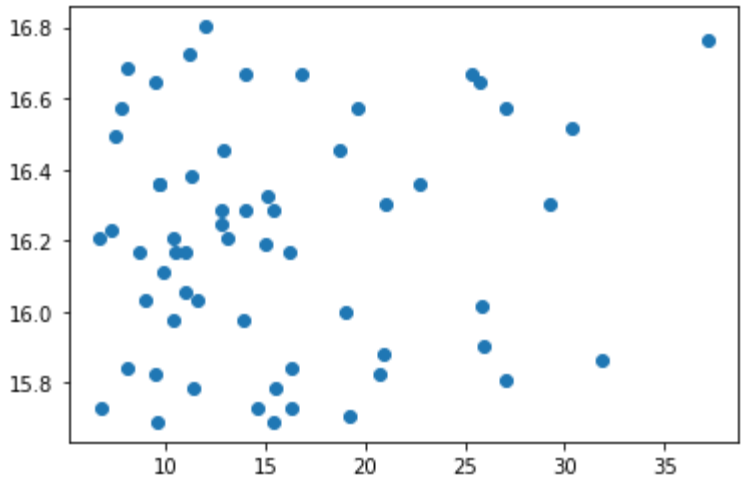
In [13]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[13]:

| | Co-efficient |
|---|---|
| **Age** | -0.019161 |

In [14]:
```
prediction= lr.predict(x_test)
```

Out[14]: <matplotlib.collections.PathCollection at 0x1ab1d71a280>




In [15]:

Out[15]: -0.00026897328465191883

# LASSO & RIDGE

In [16]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
```

Out[16]: Ridge(alpha=10)

In [17]:

Out[17]: -0.00027071646583842757

In [18]:
```
la=Lasso(alpha=10)
```

Out[18]: Lasso(alpha=10)

In [19]:

Out[19]: -0.008883331262525829

In [20]:
```
from sklearn.linear_model import ElasticNet
a=ElasticNet()
```

Out[20]: ElasticNet()

In [21]:
```
print(a.coef_)
print(a.intercept_)
print(a.score(x_test,y_test))
```

```
[-0.01722679]
17.025175504034777
-0.0009392736929447754
[15.88820764 16.24997014 16.49114514 16.47391836 16.64618621 16.07770228
 15.93988799 16.00879514 16.61173264 16.74954693 16.5428255  15.75039335
 16.3016505  16.181063   16.61173264 16.35333086 16.62895943 16.43946478
 15.75039335 16.71509336 16.28442371 15.76762014 15.90543442 16.00879514
 16.02602192 16.181063   16.21551657 16.23274335 16.37055764 16.5428255
 15.87098085 16.5428255  15.78484692 16.35333086 16.62895943 15.92266121
 16.04324871 16.43946478 16.181063   16.31887728 15.78484692 15.83652728
 16.62895943 16.35333086 16.28442371 15.78484692 16.21551657 16.21551657
 16.181063   16.68063979 16.3016505  15.87098085 16.19828978 15.83652728
 16.0604755  16.12938264 15.85375407 15.88820764 16.0604755  16.28442371]
```

In [22]:
```
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))
```

```
 Mean Absolute Error : 5.813193064840617
 Mean Squared Error : 50.19892316183593
 Root Mean Absolute Error : 2.4110564209160716
```