In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
df=pd.read_csv('1_2015-1.csv')
df
```

Out[3]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811
...	...	...	...	...	...	...	...	...	...	...	...
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22628
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16681

158 rows × 12 columns

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

Out[4]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dy Re
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2

In [5]:
```
df.tail()
```

Out[5]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dy Re
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0.22628	0
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0.18260	1
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0.47179	0
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0.19727	1
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0.16681	1

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

Out[6]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopi Residua
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.00000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.237296	2.09897
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.126685	0.55355
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.32858
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.150553	1.75941
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.216130	2.09541
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.309883	2.46241
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.795880	3.60214

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Country                        158 non-null    object 
 1   Region                         158 non-null    object 
 2   Happiness Rank                 158 non-null    int64  
 3   Happiness Score                158 non-null    float64
 4   Standard Error                 158 non-null    float64
 5   Economy (GDP per Capita)       158 non-null    float64
 6   Family                         158 non-null    float64
 7   Health (Life Expectancy)       158 non-null    float64
 8   Freedom                        158 non-null    float64
 9   Trust (Government Corruption)  158 non-null    float64
 10  Generosity                     158 non-null    float64
 11  Dystopia Residual              158 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

In [8]:
```
list(df)
```

Out[8]:
```
['Country',
 'Region',
 'Happiness Rank',
 'Happiness Score',
 'Standard Error',
 'Economy (GDP per Capita)',
 'Family',
 'Health (Life Expectancy)',
 'Freedom',
 'Trust (Government Corruption)',
 'Generosity',
 'Dystopia Residual']
```

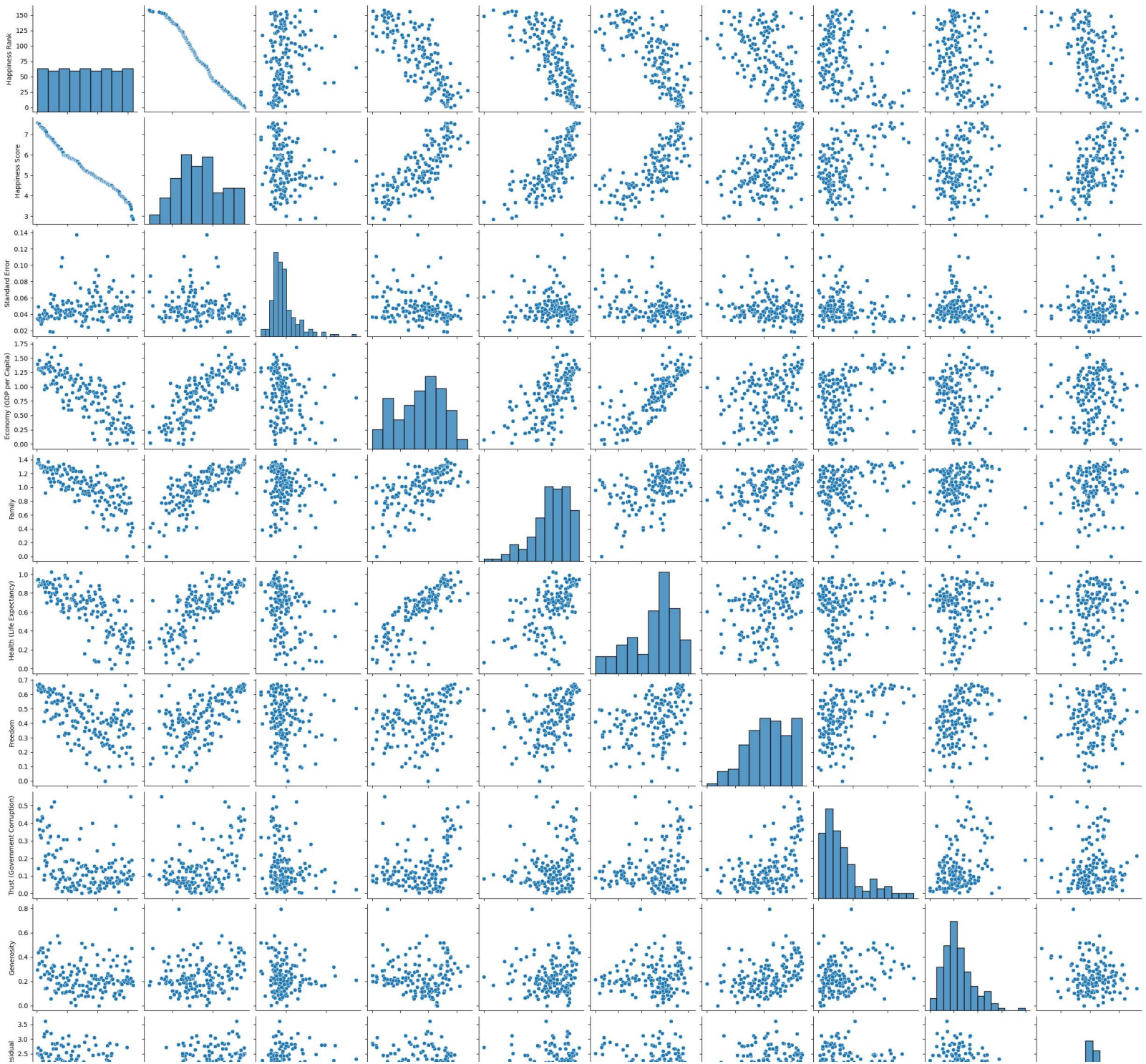
In [13]:
```
df.columns
```

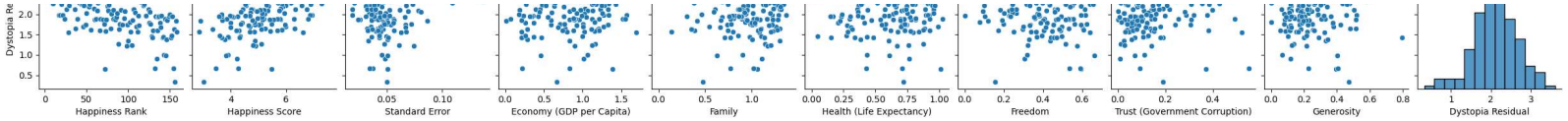
Out[13]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

visualization

In [9]:
```
sns.pairplot(df)
```

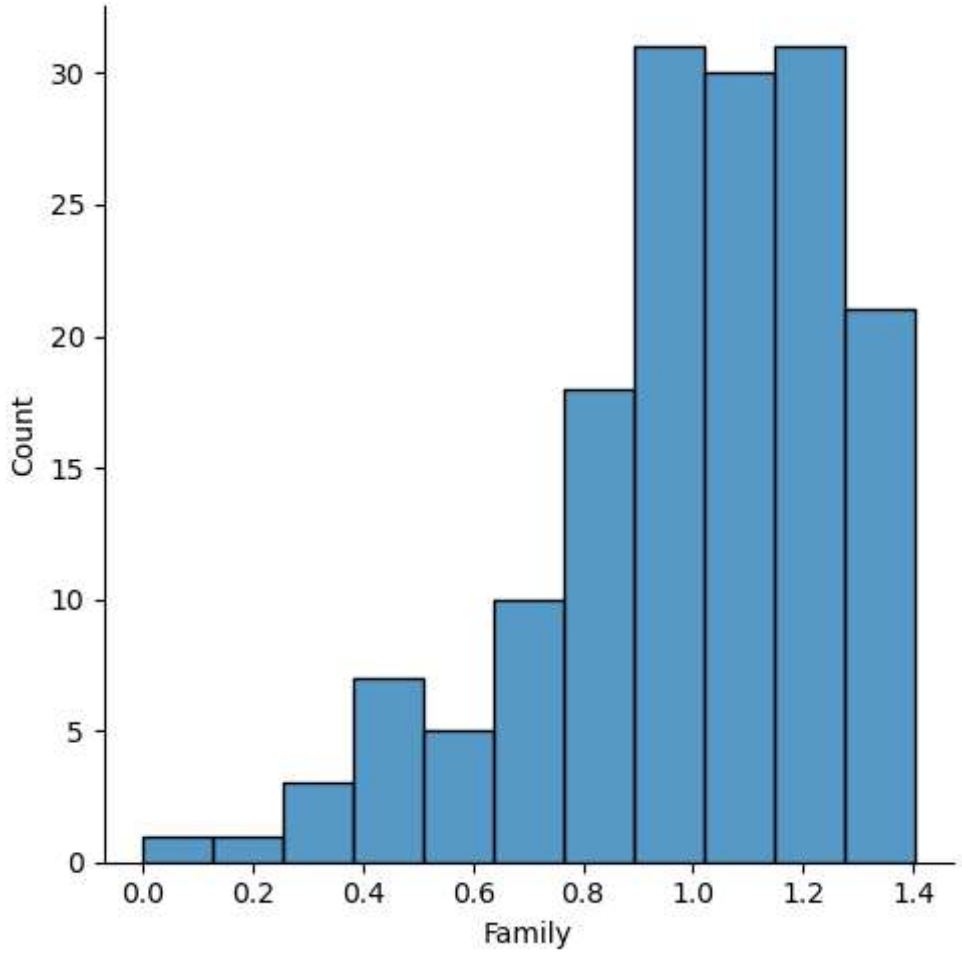
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x2075c546910>
```


Happiness Rank
Happiness Score
Standard Error
Economy (GDP per Capita)
Family
Health (Life Expectancy)
Freedom
Trust (Government Corruption)
Generosity



In [10]:
```
sns.displot(df['Family'])
```

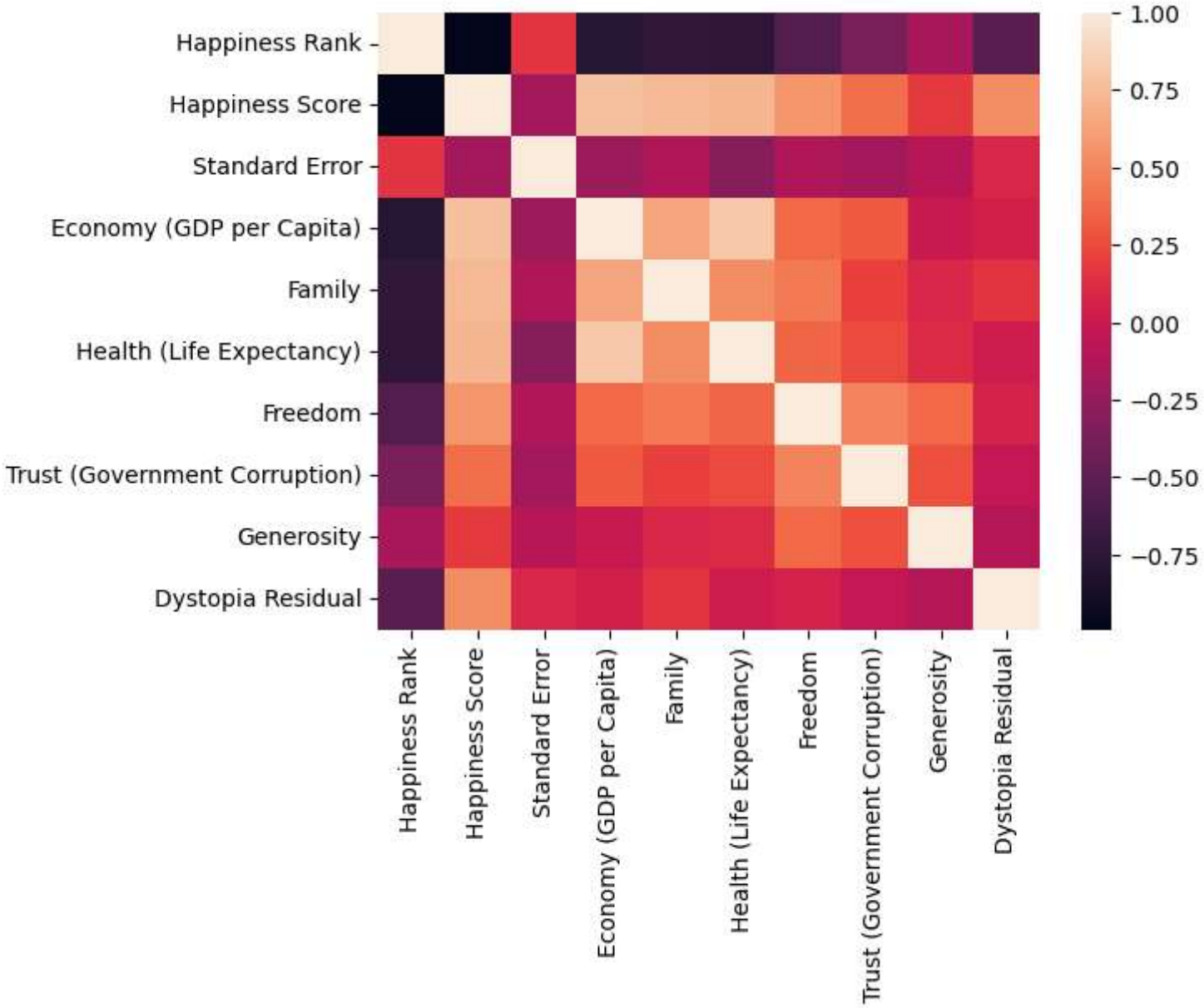
Out[10]: <seaborn.axisgrid.FacetGrid at 0x20761186e50>



In [11]:
```
sns.heatmap(df.corr())
```

```
C:\Users\Ajay\AppData\Local\Temp\ipykernel_20876\58359773.py:1: FutureWarning: The default value of num
eric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only v
alid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr())
```

Out[11]:
```
<Axes: >
```



Model building

In [20]:
```
X=df[['Happiness Rank', 'Happiness Score','Standard Error', 'Economy (GDP per Capita)','Health (Life Expe
      'Freedom', 'Trust (Government Corruption)','Generosity', 'Dystopia Residual']]
y=df['Family']
```

In [22]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.60)
```

In [23]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
```

Out[23]:
```
▾ LinearRegression

LinearRegression()
```

In [24]:
```
predX=lr.predict(X_test)
print(predX)
```

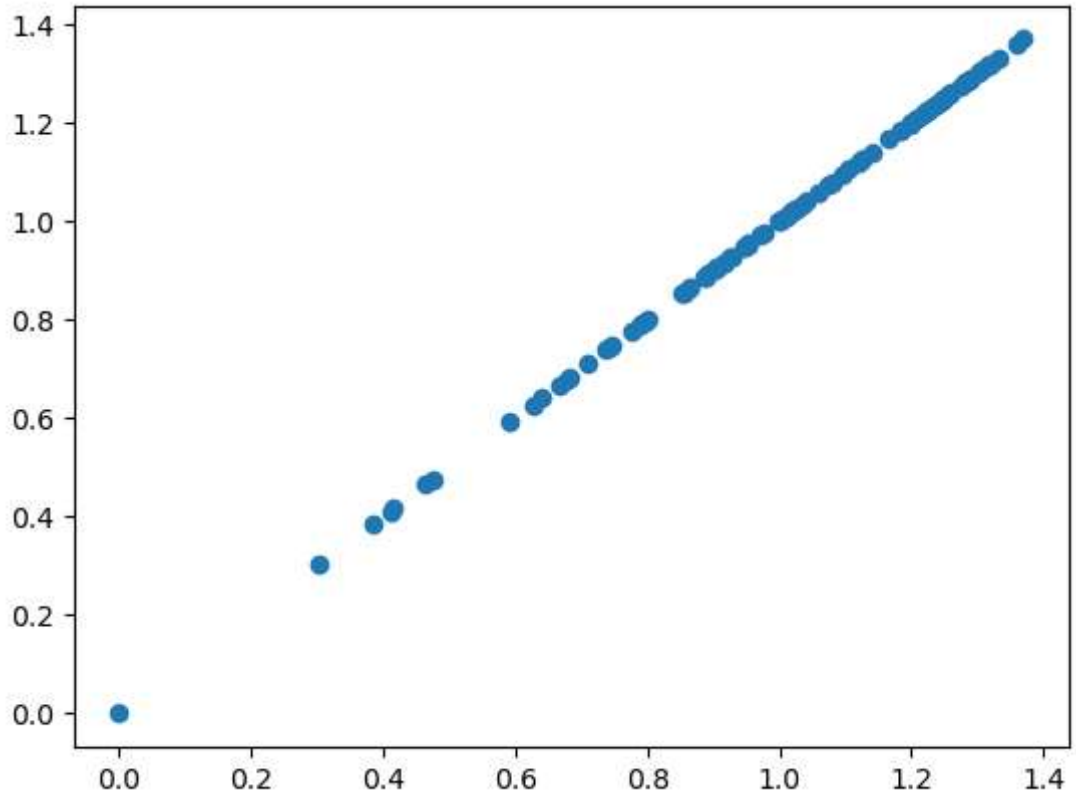
```
[9.98745468e-01 1.28525100e+00 6.68137215e-01 1.36048203e+00
 6.78992837e-01 1.31815648e+00 1.00226136e+00 1.24808319e+00
 1.19816478e+00 1.18368758e+00 7.96550444e-01 8.60272281e-01
 7.99792373e-01 8.52004398e-01 1.27998450e+00 4.05746311e-05
 1.20872845e+00 1.31384758e+00 9.53540382e-01 1.25772179e+00
 8.85653171e-01 1.00948097e+00 1.12532843e+00 1.28524970e+00
 1.22647174e+00 8.55187967e-01 7.89109611e-01 1.23807325e+00
 8.87610322e-01 9.14060882e-01 4.74680523e-01 4.64777611e-01
 7.77121600e-01 9.76455721e-01 7.46725054e-01 1.23332290e+00
 7.43088036e-01 4.10739098e-01 1.09602369e+00 9.04170190e-01
 9.45783672e-01 1.16608387e+00 1.31960075e+00 1.11813537e+00
 1.02021261e+00 1.01517804e+00 8.64110601e-01 1.33068984e+00
 4.16219521e-01 1.22412737e+00 1.27986137e+00 3.85944836e-01
 1.19756117e+00 7.09071636e-01 1.24030431e+00 6.40800399e-01
 1.03485876e+00 5.91819362e-01 8.93409939e-01 1.00287590e+00
 9.51211670e-01 1.21652289e+00 7.93052780e-01 9.71574400e-01
 1.13971419e+00 9.25773573e-01 9.28755076e-01 6.27200385e-01
 9.16104005e-01 1.01338257e+00 1.02479769e+00 1.21937702e+00
 1.02595855e+00 1.12212850e+00 1.20627701e+00 9.46356944e-01
 1.27419640e+00 1.26029986e+00 3.03208361e-01 1.25559376e+00
 1.10445077e+00 1.07824699e+00 1.30037038e+00 1.30440914e+00
 1.24651152e+00 1.28937277e+00 7.37975753e-01 1.04127886e+00
 9.04782709e-01 1.01945443e+00 6.81140386e-01 1.36921083e+00
 1.18430711e+00 1.07247591e+00 1.05844169e+00]
```

In [25]:
```
print(lr.score(X_test,y_test))
```

```
0.9999987455850263
```

In [26]:
```
plt.scatter(y_test,predX)
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x20769064910>
```



In [28]:
```
df1=pd.read_csv('3_Fitness.csv')
df1
```

Out[28]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11
0	ANU	70.0	80.0	75.0	60.0	72.0	55.0	412.0	NaN	NaN	NaN	NaN
1	BABU	30.0	48.0	35.0	45.0	25.0	37.0	220.0	NaN	NaN	NaN	1. Individual Sales using Sum()
2	CHANDRU	65.0	54.0	49.0	54.0	35.0	65.0	322.0	NaN	NaN	NaN	2. Find the pattern trend using conditional fo...
3	DAVID	85.0	71.0	68.0	77.0	88.0	73.0	462.0	NaN	NaN	NaN	3. Analyze using Pivot table as column percentage
4	EINSTEIN	55.0	25.0	45.0	50.0	53.0	30.0	258.0	NaN	NaN	NaN	4. Insert Pivot charts
5	FAROOK	35.0	45.0	15.0	45.0	45.0	25.0	210.0	NaN	NaN	NaN	5. Rank() - returns the rank of a given value ...
6	GOWTHAM	75.0	66.0	59.0	65.0	56.0	30.0	351.0	NaN	NaN	NaN	NaN
7	HARSHITH	29.0	35.0	49.0	48.0	35.0	55.0	247.0	NaN	NaN	NaN	33
8	INIYAN	35.0	35.0	50.0	59.0	67.0	73.0	319.0	NaN	NaN	NaN	NaN
9	JOHN	77.0	85.0	77.0	68.0	56.0	25.0	388.0	NaN	NaN	NaN	NaN
10	MONTHLY SALES	556.0	544.0	522.0	571.0	532.0	468.0	NaN	3193.0	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3189.0	NaN	NaN	NaN	NaN

In [29]:
```
df1.head()
```

Out[29]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11
0	ANU	70.0	80.0	75.0	60.0	72.0	55.0	412.0	NaN	NaN	NaN	NaN
1	BABU	30.0	48.0	35.0	45.0	25.0	37.0	220.0	NaN	NaN	NaN	1. Individual Sales using Sum()
2	CHANDRU	65.0	54.0	49.0	54.0	35.0	65.0	322.0	NaN	NaN	NaN	2. Find the pattern trend using conditional fo...
3	DAVID	85.0	71.0	68.0	77.0	88.0	73.0	462.0	NaN	NaN	NaN	3. Analyze using Pivot table as column percentage
4	EINSTEIN	55.0	25.0	45.0	50.0	53.0	30.0	258.0	NaN	NaN	NaN	4. Insert Pivot charts

In [30]:
```
df1.tail()
```

Out[30]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed: 11
7	HARSHITH	29.0	35.0	49.0	48.0	35.0	55.0	247.0	NaN	NaN	NaN	33
8	INIYAN	35.0	35.0	50.0	59.0	67.0	73.0	319.0	NaN	NaN	NaN	NaN
9	JOHN	77.0	85.0	77.0	68.0	56.0	25.0	388.0	NaN	NaN	NaN	NaN
10	MONTHLY SALES	556.0	544.0	522.0	571.0	532.0	468.0	NaN	3193.0	NaN	NaN	NaN
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3189.0	NaN	NaN	NaN	NaN

In [31]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   SALESMAN     11 non-null     object
 1   JAN          11 non-null     float64
 2   FEB          11 non-null     float64
 3   MAR          11 non-null     float64
 4   APR          11 non-null     float64
 5   MAY          11 non-null     float64
 6   JUN          11 non-null     float64
 7   TOTAL SALES  11 non-null     float64
 8   Unnamed: 8   1 non-null      float64
 9   Unnamed: 9   0 non-null      float64
 10  Unnamed: 10  0 non-null      float64
 11  Unnamed: 11  6 non-null      object
dtypes: float64(10), object(2)
memory usage: 1.3+ KB
```

In [32]:
```
df1.describe()
```

Out[32]:

	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10
count	11.000000	11.000000	11.000000	11.000000	11.000000	11.000000	11.000000	1.0	0.0	0.0
mean	101.090909	98.909091	94.909091	103.818182	96.727273	85.090909	579.818182	3193.0	NaN	NaN
std	152.263886	148.884153	142.770763	155.277054	145.500578	128.347540	869.142775	NaN	NaN	NaN
min	29.000000	25.000000	15.000000	45.000000	25.000000	25.000000	210.000000	3193.0	NaN	NaN
25%	35.000000	40.000000	47.000000	49.000000	40.000000	30.000000	252.500000	3193.0	NaN	NaN
50%	65.000000	54.000000	50.000000	59.000000	56.000000	55.000000	322.000000	3193.0	NaN	NaN
75%	76.000000	75.500000	71.500000	66.500000	69.500000	69.000000	400.000000	3193.0	NaN	NaN
max	556.000000	544.000000	522.000000	571.000000	532.000000	468.000000	3189.000000	3193.0	NaN	NaN

In [33]:
```
list(df1)
```

Out[33]:
```
['SALESMAN',
 'JAN',
 'FEB',
 'MAR',
 'APR',
 'MAY',
 'JUN',
 'TOTAL SALES',
 'Unnamed: 8',
 'Unnamed: 9',
 'Unnamed: 10',
 'Unnamed: 11']
```

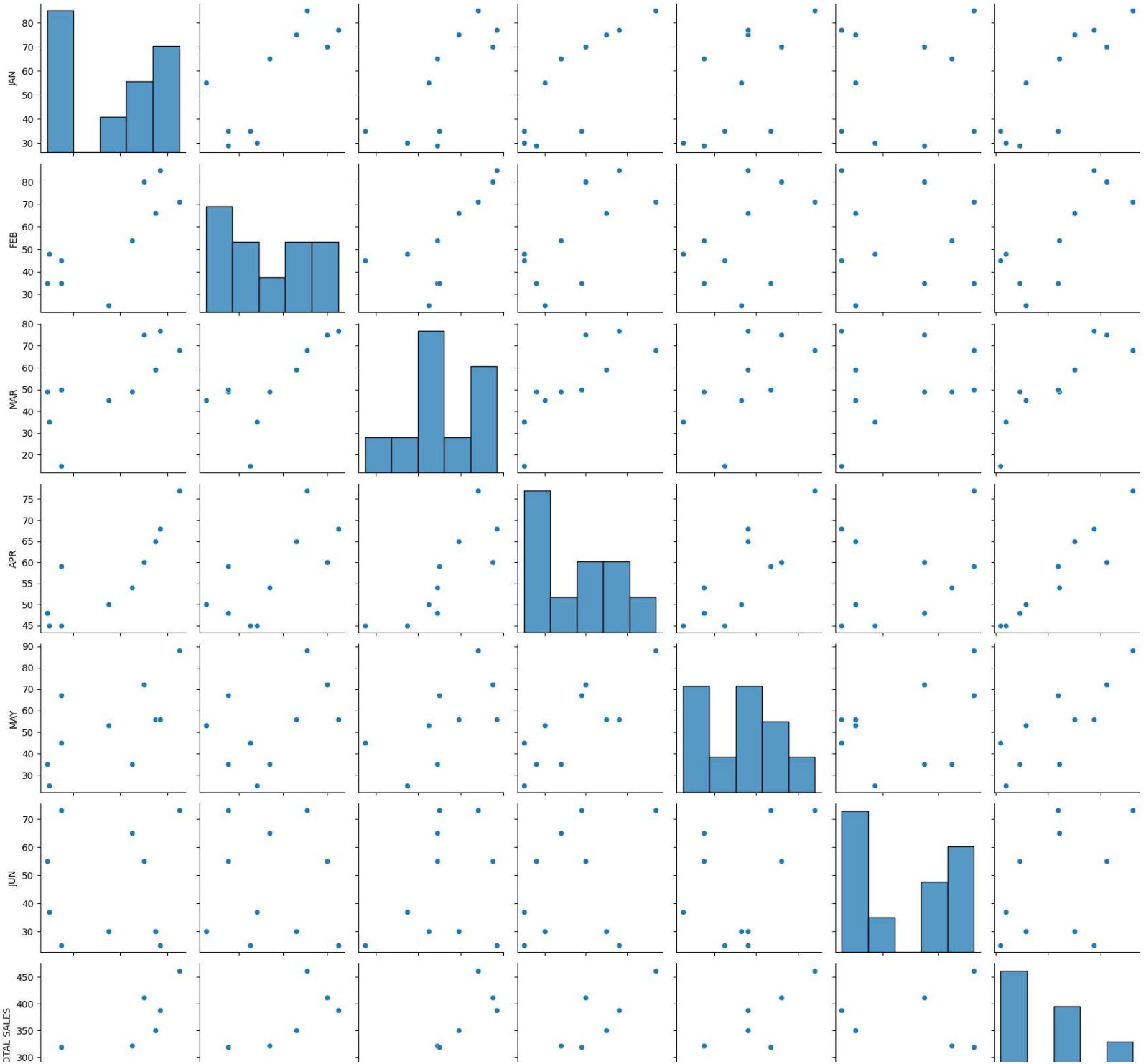
In [53]:
```
df11=df1.drop([ 'Unnamed: 8',
 'Unnamed: 9',
 'Unnamed: 10',
 'Unnamed: 11'],axis=1)
df111=df11.dropna()
df111
```

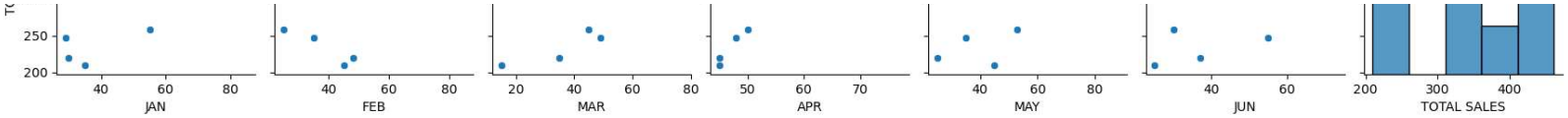
Out[53]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES
0	ANU	70.0	80.0	75.0	60.0	72.0	55.0	412.0
1	BABU	30.0	48.0	35.0	45.0	25.0	37.0	220.0
2	CHANDRU	65.0	54.0	49.0	54.0	35.0	65.0	322.0
3	DAVID	85.0	71.0	68.0	77.0	88.0	73.0	462.0
4	EINSTEIN	55.0	25.0	45.0	50.0	53.0	30.0	258.0
5	FAROOK	35.0	45.0	15.0	45.0	45.0	25.0	210.0
6	GOWTHAM	75.0	66.0	59.0	65.0	56.0	30.0	351.0
7	HARSHITH	29.0	35.0	49.0	48.0	35.0	55.0	247.0
8	INIYAN	35.0	35.0	50.0	59.0	67.0	73.0	319.0
9	JOHN	77.0	85.0	77.0	68.0	56.0	25.0	388.0

In [54]:
```
sns.pairplot(df111)
```

Out[54]:
```
<seaborn.axisgrid.PairGrid at 0x2077229e910>
```

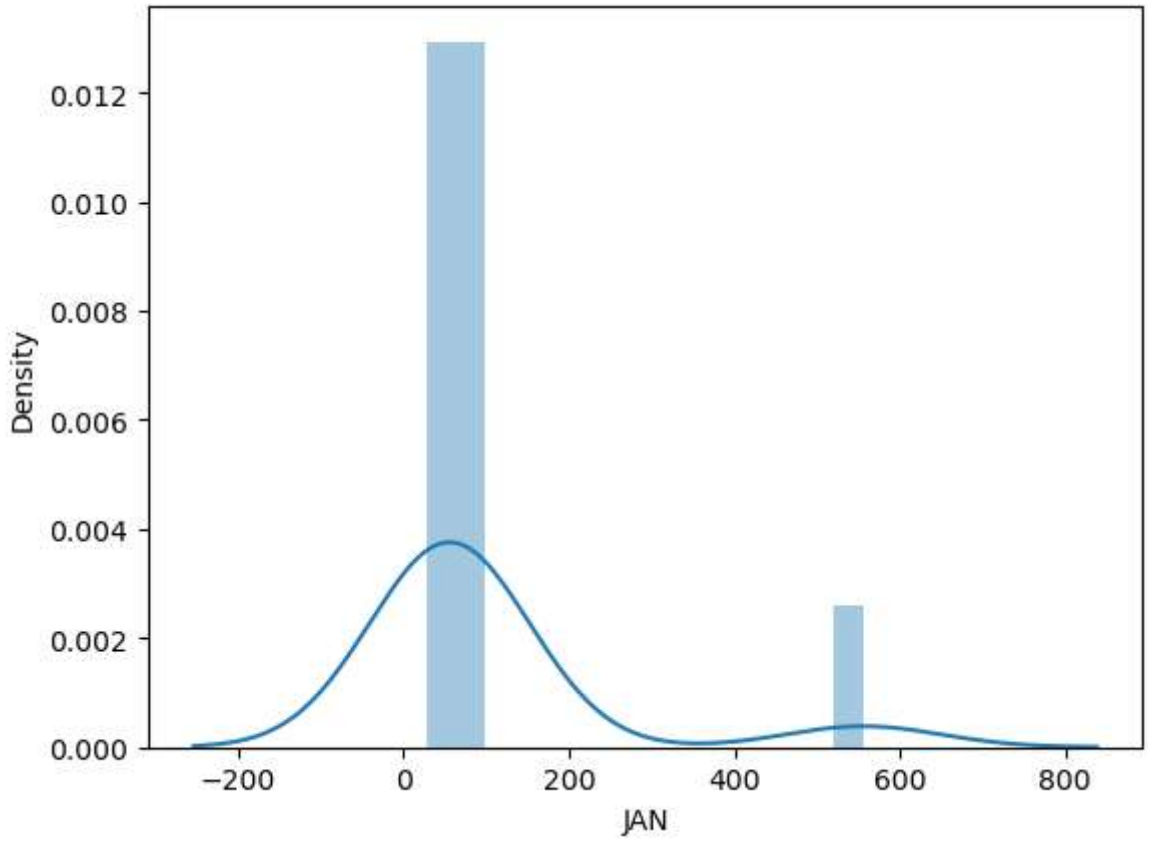

JAN
FEB
MAR
APR
MAY
JUN
OTAL SALES


250
200
40
60
80
JAN
40
60
80
FEB
20
40
60
80
MAR
50
60
70
APR
40
60
80
MAY
40
60
JUN
200
300
400
TOTAL SALES

In [37]:
```
sns.distplot(df1['JAN'])
```

```
C:\Users\Ajay\AppData\Local\Temp\ipykernel_20876\1533401960.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df1['JAN'])
```

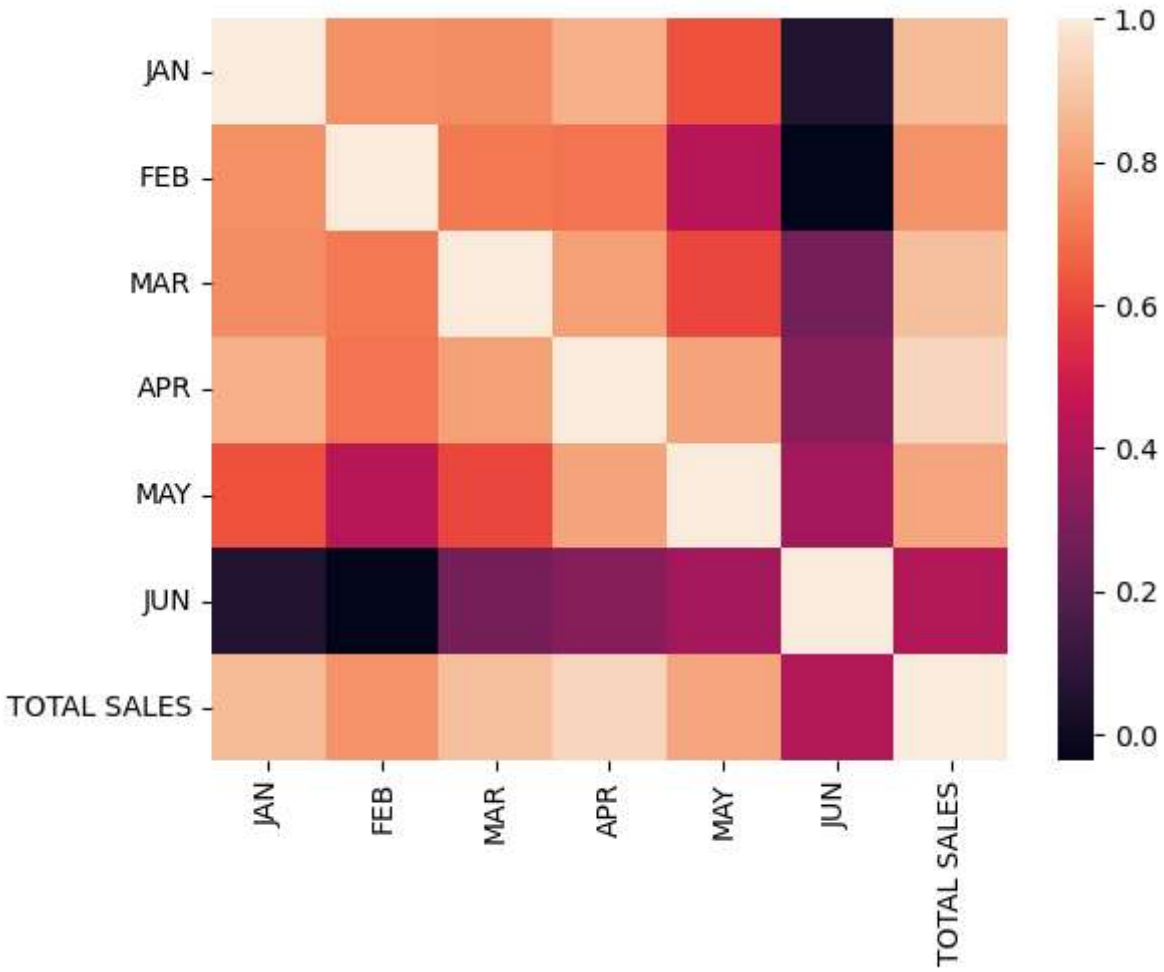
Out[37]: <Axes: xlabel='JAN', ylabel='Density'>



In [55]:
```
sns.heatmap(df111.corr())
```

```
C:\Users\Ajay\AppData\Local\Temp\ipykernel_20876\3153282829.py:1: FutureWarning: The default value of n
umeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only
valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df111.corr())
```

Out[55]: <Axes: >



In [56]:
```
X=df111[['JAN',
 'FEB',
 'MAR',
 'APR',
 'MAY',
 'JUN']]
y=df111['TOTAL SALES']
```

In [57]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.40)
```

In [58]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
```

Out[58]:
```
LinearRegression
LinearRegression()
```

In [60]:
```
predX=lr.predict(X_test)
print(predX)
```

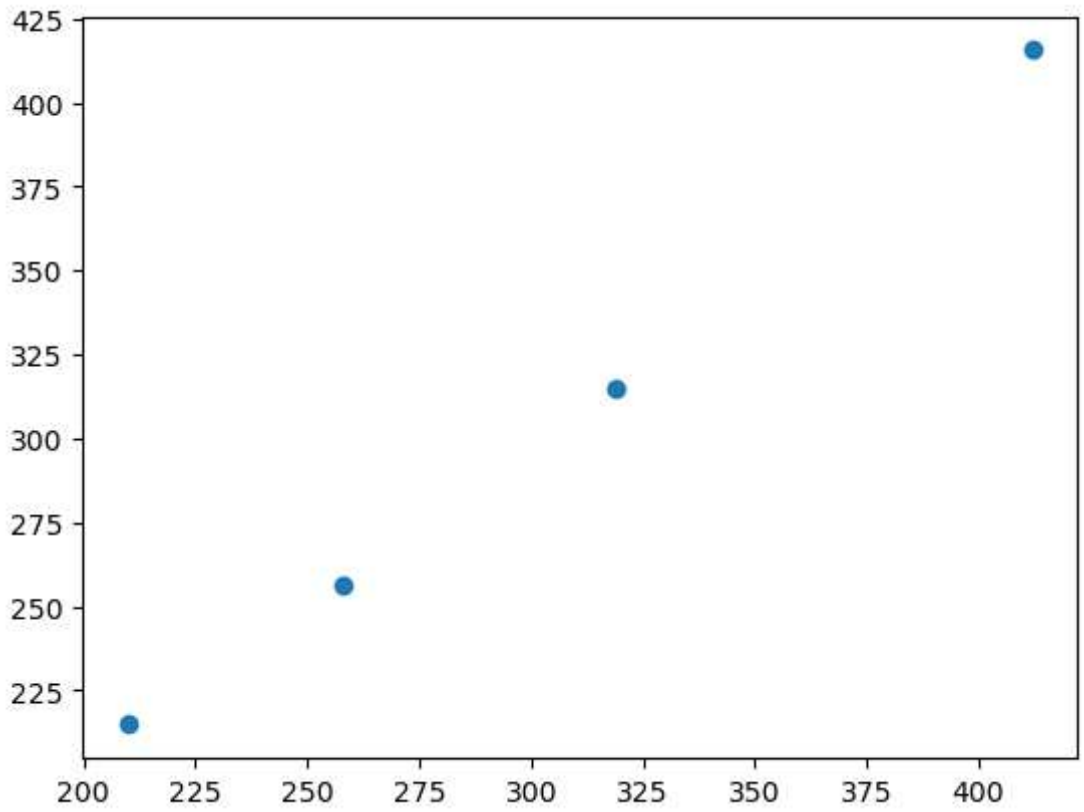
```
[315.17186693 256.21465128 214.78812605 415.63447479]
```

In [62]:
```
print(lr.score(X_test,y_test))
```

```
0.9976293107487954
```

In [63]:
```
plt.scatter(y_test,predX)
```

Out[63]: <matplotlib.collections.PathCollection at 0x20776e67350>



In [64]:
```
df2=pd.read_csv('4_Drug200.csv')
```

In [65]:
```
df2.head()
```

Out[65]:

	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 [66]:
```
df2.tail()
```

Out[66]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
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

In [67]:
```
df2.describe()
```

Out[67]:

	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 [68]:
```
df2.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 [79]:
```
df2['Sex'].unique()
df2['BP'].unique()
df2['Cholesterol'].unique()
df2['Drug'].unique()
```

Out[79]:
```
array(['drugY', 'drugC', 'drugX', 'drugA', 'drugB'], dtype=object)
```

In [80]:
```
Sex={'Sex':{'M':1,'F':2}}
df2=df2.replace(Sex)
BP={"BP":{'HIGH':1,'NORMAL':2,'LOW':3}}
df2=df2.replace(BP)
Cholesterol={"Cholesterol":{"HIGH":1,'NORMAL':2}}
df2=df2.replace(Cholesterol)
Drug={"Drug":{"drugA":1,"drugB":2,"drugC":3,"drugX":4,"drugY":5}}
df2=df2.replace(Drug)
```

In [81]:
```
df2
```

Out[81]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	2	1	1	25.355	5
1	47	1	3	1	13.093	3
2	47	1	3	1	10.114	3
3	28	2	2	1	7.798	4
4	61	2	3	1	18.043	5
...	...	...	...	...	...	...
195	56	2	3	1	11.567	3
196	16	1	3	1	12.006	3
197	52	1	2	1	9.894	4
198	23	1	2	2	14.020	4
199	40	2	3	2	11.349	4

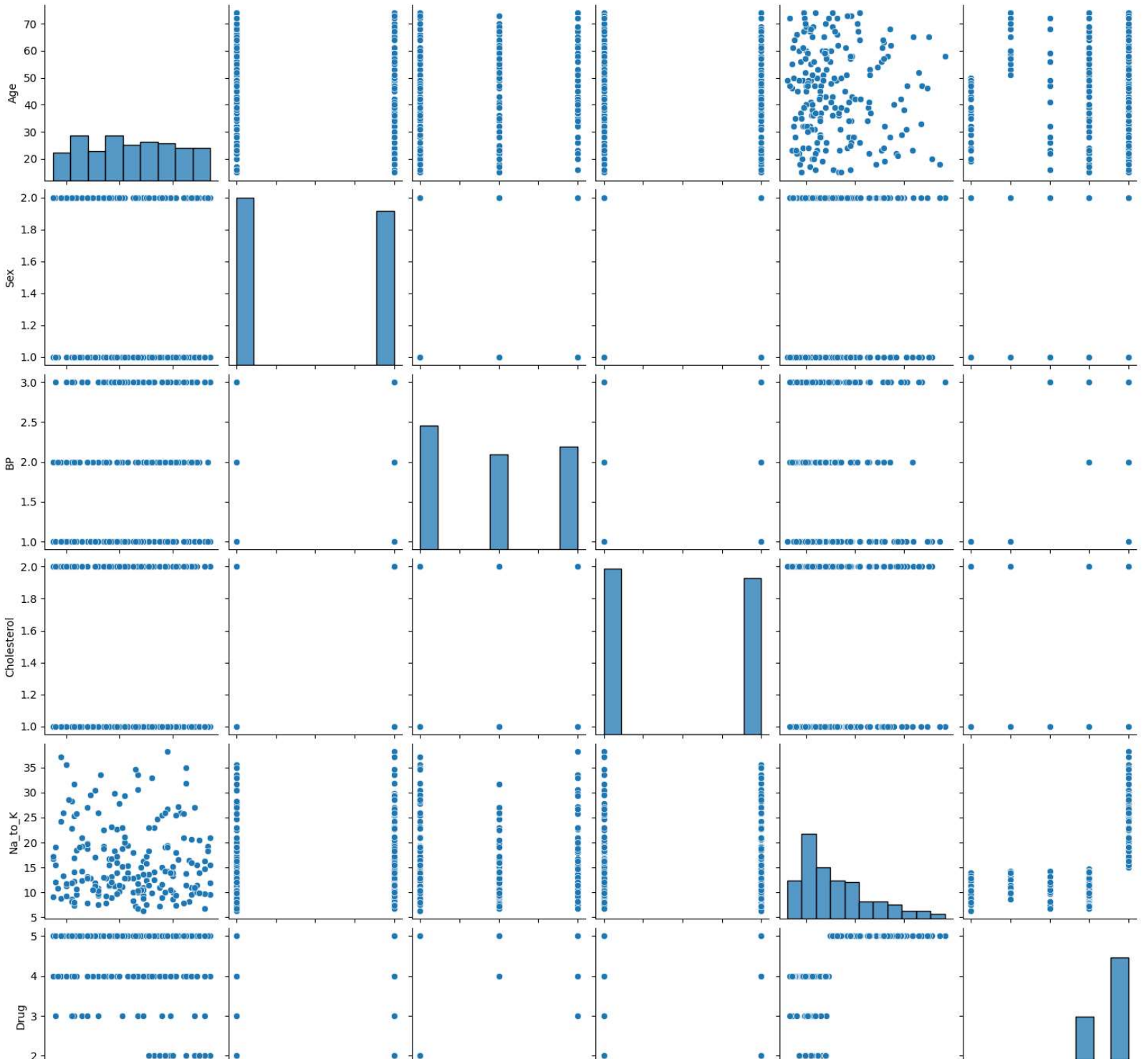
200 rows × 6 columns

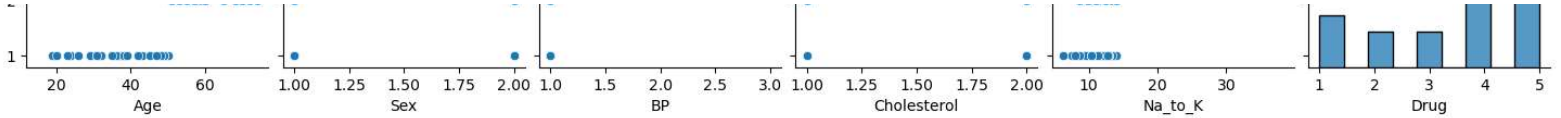
In [82]:
```
list(df2)
```

Out[82]:
```
['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']
```

In [83]:
```
sns.pairplot(df2)
```

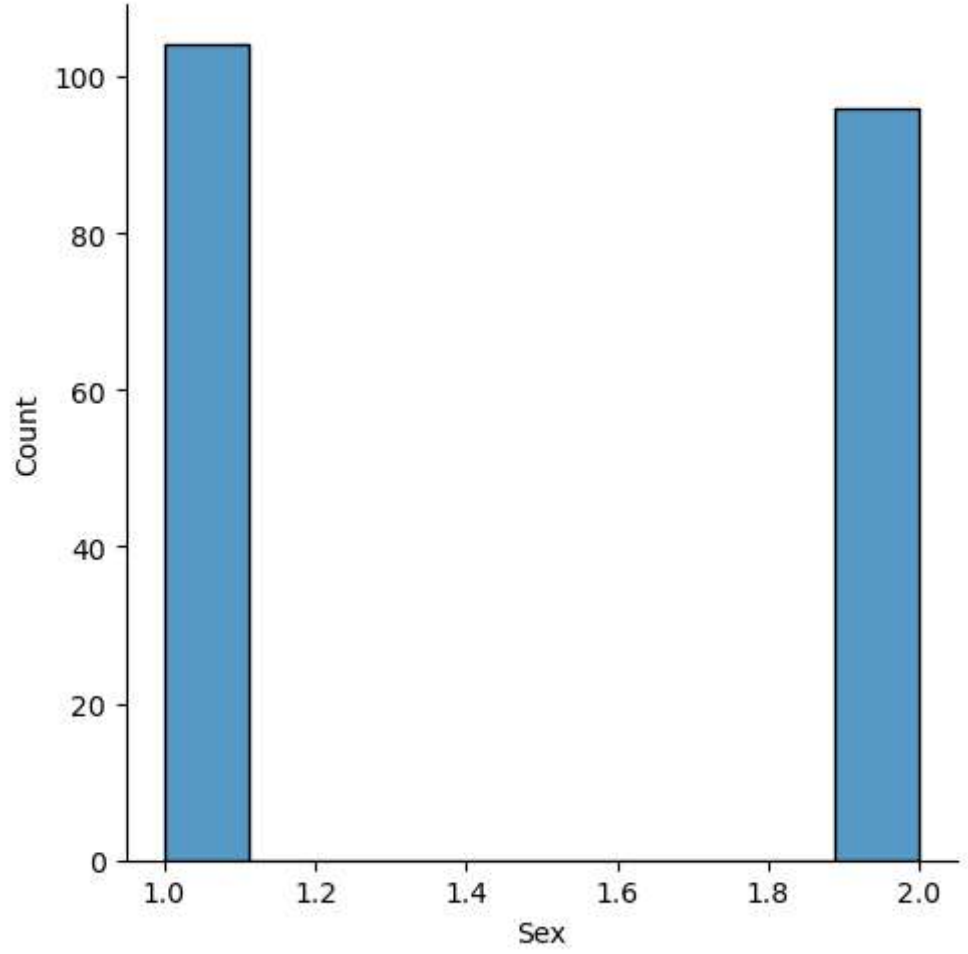
Out[83]:
```
<seaborn.axisgrid.PairGrid at 0x20776eeaed0>
```


Age
Sex
BP
Cholesterol
Na_to_K
Drug



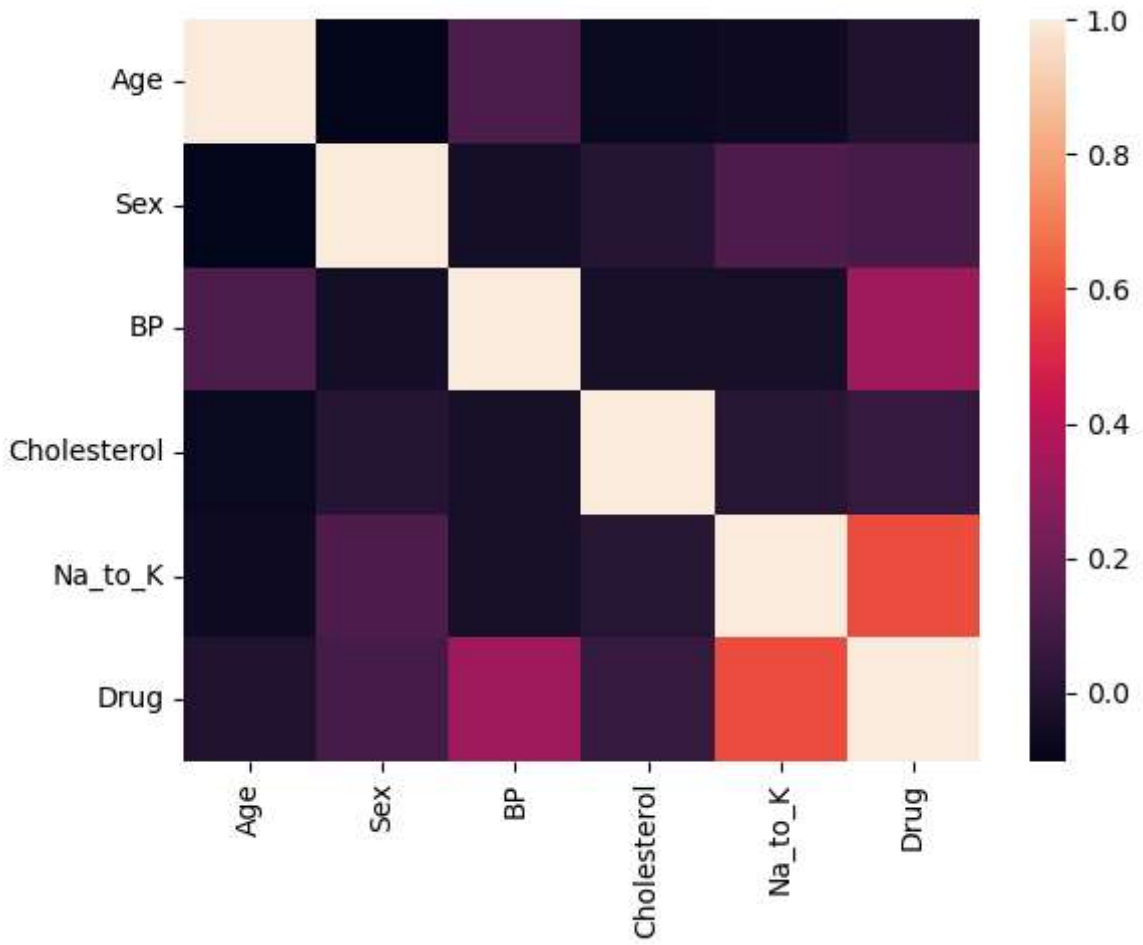
In [85]:
```
sns.displot(df2['Sex'])
```

Out[85]: <seaborn.axisgrid.FacetGrid at 0x2077228d190>



In [86]:
```
sns.heatmap(df2.corr())
```

Out[86]: <Axes: >



In [88]:
```
X=df2[[
     'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'
]]
y=df2['Age']
```

In [97]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30)
```

In [98]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
```

Out[98]:
```
LinearRegression

LinearRegression()
```

In [99]:
```
predX=lr.predict(X_test)
print(predX)
```

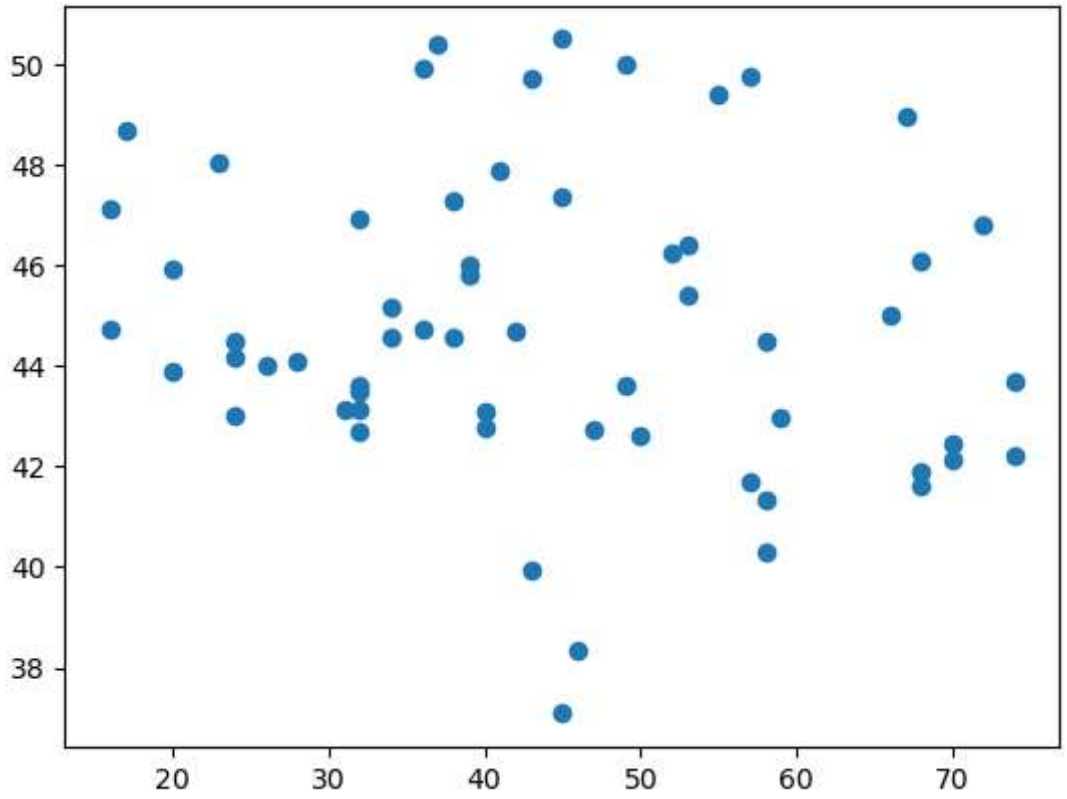
```
[50.37186222 39.94908389 44.16012064 38.34912772 42.68933958 46.9251948
 43.12178868 44.58413172 43.69635193 41.33686576 44.50041222 44.70231074
 42.77175043 41.67557221 41.91216365 44.55676249 43.47045595 43.87867014
 42.13913    40.31044448 48.93599935 46.78316637 43.61855307 49.76728735
 47.29465634 46.38131245 41.62783102 49.97154054 47.13289797 44.74064577
 49.89131967 45.81745484 45.39662651 42.72486379 43.00652658 44.02237555
 37.0874418  44.09124809 46.22603198 42.1943063  47.88638711 49.72032879
 46.09404958 43.10474822 42.98372282 44.4999303  43.61622313 45.15148024
 42.45595404 45.01773896 43.12694641 45.90119759 48.05435249 47.3619636
 50.48886728 49.37682278 44.73555859 45.98858382 48.67811862 42.59603054]
```

In [100]:
```
print(lr.score(X_test,y_test))
```

```
-0.09316356940539339
```

In [101]:
```
plt.scatter(y_test,predX)
```

Out[101]: <matplotlib.collections.PathCollection at 0x20779f29810>



In [ ]:
```

```