Name : Akash Kulkarni

Class : TE-10

Roll : 33241

Lab : DSDBAL

Problem Statement :

Perform the following operations using Python on the Air quality and Heart Diseases data sets

```
a. Data cleaning
b. Data integration
c. Data transformation
d. Error correcting
e. Data model building
```

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

In [2]:
```python
dataset = pd.read_csv('Heart.csv')
```

In [3]:
```python
dataset.head(10)
```

Out[3]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
5	6	56	1	nontypical	120	236	0	0	178	0	0.8	1	0.0	normal	No
6	7	62	0	asymptomatic	140	268	0	2	160	0	3.6	3	2.0	normal	Yes
7	8	57	0	asymptomatic	120	354	0	0	163	1	0.6	1	0.0	normal	No
8	9	63	1	asymptomatic	130	254	0	2	147	0	1.4	2	1.0	reversable	Yes
9	10	53	1	asymptomatic	140	203	1	2	155	1	3.1	3	0.0	reversable	Yes

In [4]:
```python
dataset.isna().sum()
```

Out[4]:
```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            4
Thal          2
AHD           0
dtype: int64
```

Data Cleaning

In [5]:
```python
dataset = dataset.dropna(axis=0)
```

In [6]:
```python
dataset.isnull().sum()
```

Out[6]:
```
Unnamed: 0    0
Age           0
Sex           0
ChestPain     0
RestBP        0
Chol          0
Fbs           0
RestECG       0
MaxHR         0
ExAng         0
Oldpeak       0
Slope         0
Ca            0
Thal          0
AHD           0
dtype: int64
```

In [7]:
```python
dataset.describe()
```

Out[7]:

	Unnamed: 0	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297
mean	150.673401	54.542088	0.676768	131.693603	247.350168	0.144781	0.996633	149.599327	0.326599	1.055556	1.602694	0
std	87.323283	9.049736	0.468500	17.762806	51.997583	0.352474	0.994914	22.941562	0.469761	1.166123	0.618187	0
min	1.000000	29.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0
25%	75.000000	48.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	1.000000	0
50%	150.000000	56.000000	1.000000	130.000000	243.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0
75%	226.000000	61.000000	1.000000	140.000000	276.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1
max	302.000000	77.000000	1.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3

In [8]:
```python
dataset.count()
```

Out[8]:
```
Unnamed: 0    297
Age           297
Sex           297
ChestPain     297
RestBP        297
Chol          297
Fbs           297
RestECG       297
MaxHR         297
ExAng         297
Oldpeak       297
Slope         297
Ca            297
Thal          297
AHD           297
dtype: int64
```

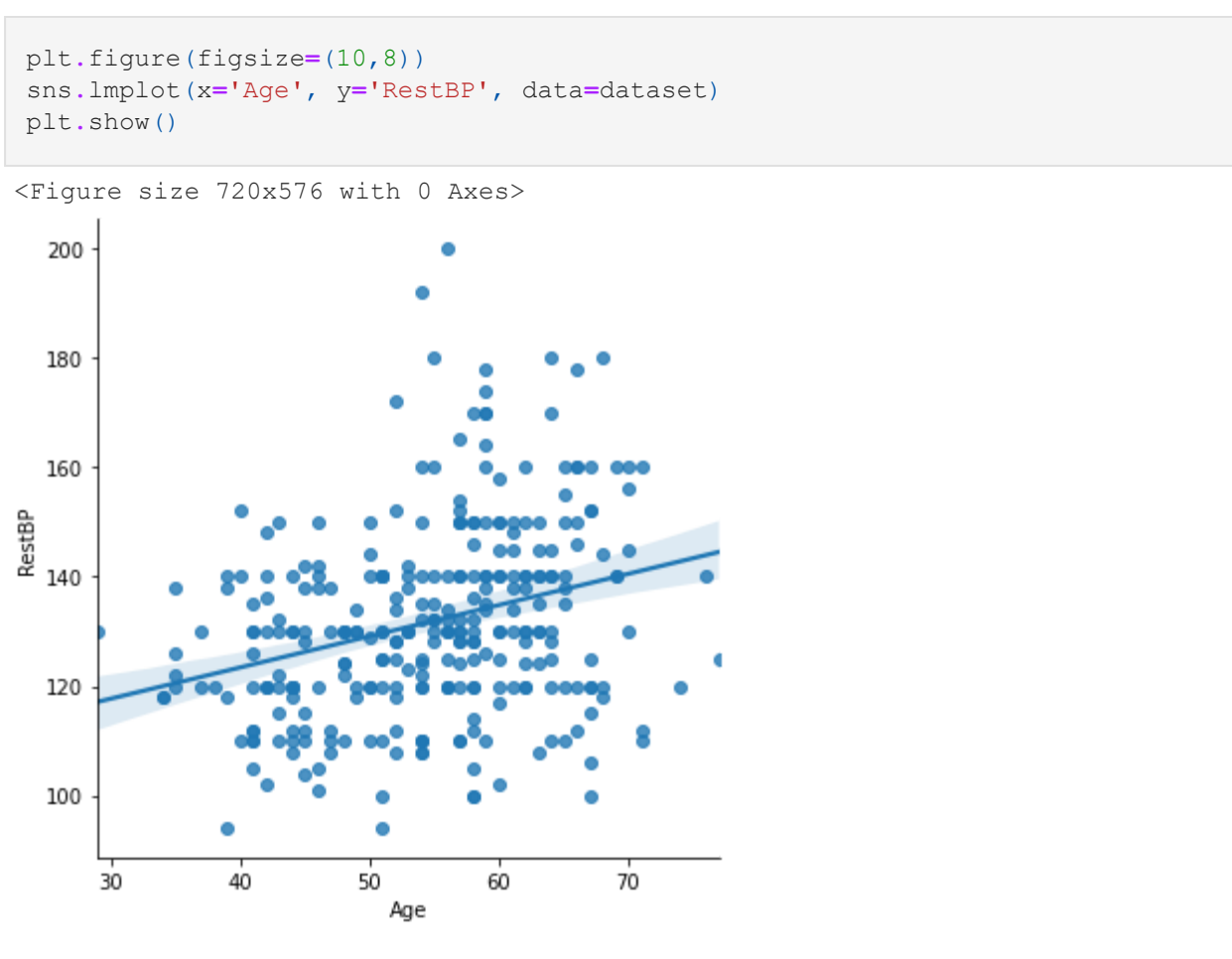
Data Visualizations

In [31]:
```python
plt.figure(figsize=(6,4))
dataset.plot(kind='box',figsize=(20,20))
```

Out[31]:
```
<AxesSubplot:>
<Figure size 432x288 with 0 Axes>
```

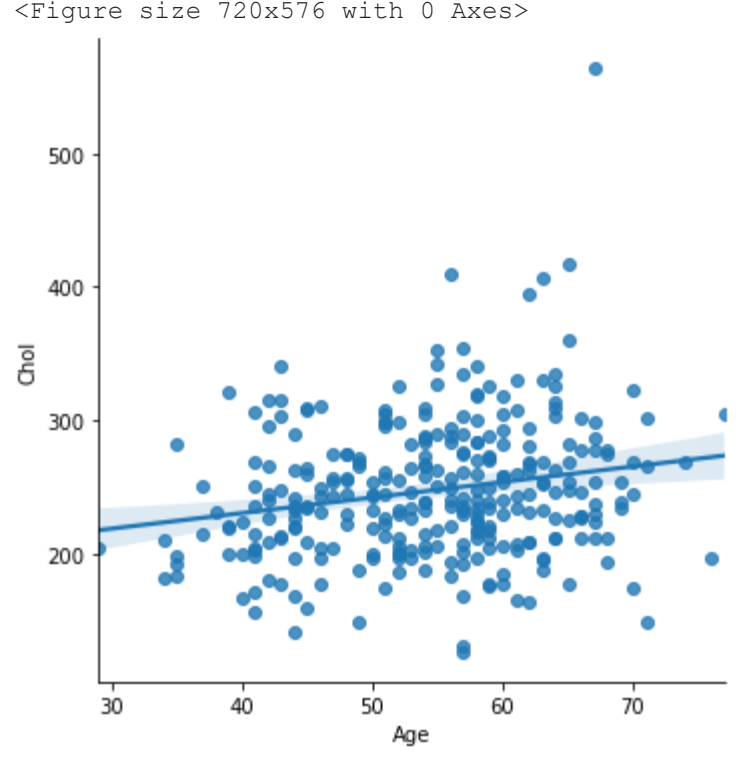
In [10]:
```python
plt.figure(figsize=(10,8))
sns.distplot(dataset['Age'])
plt.show()
```

```
C:\Users\akash\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 function for histograms).
  warnings.warn(msg, FutureWarning)
```



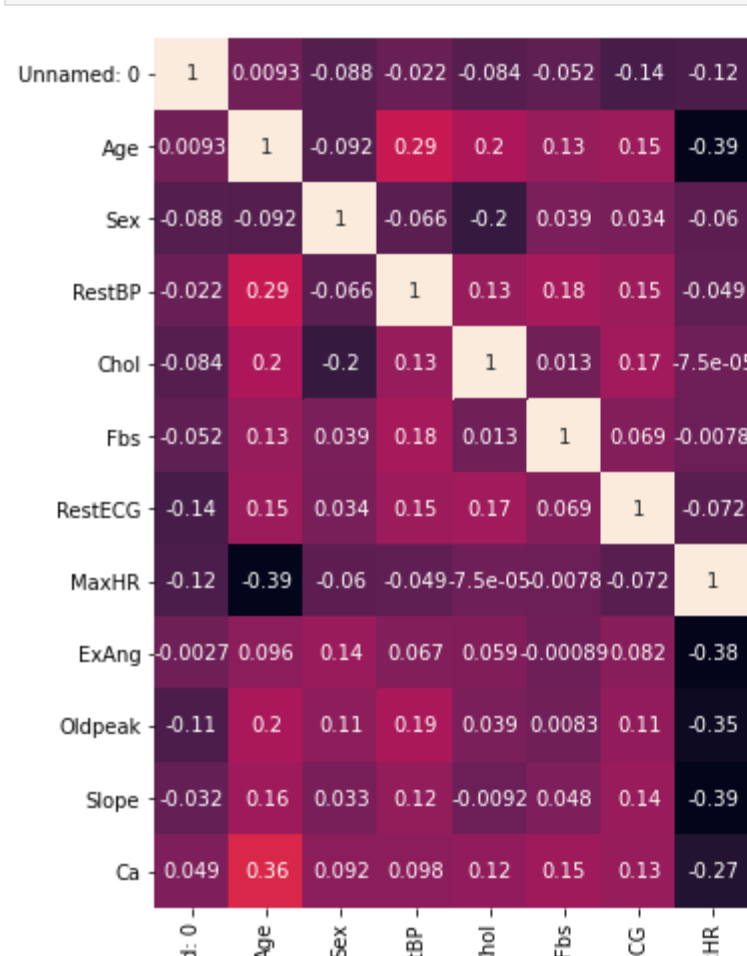
In [11]:
```python
plt.figure(figsize=(10,8))
sns.lmplot(x='Age', y='RestBP', data=dataset)
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```

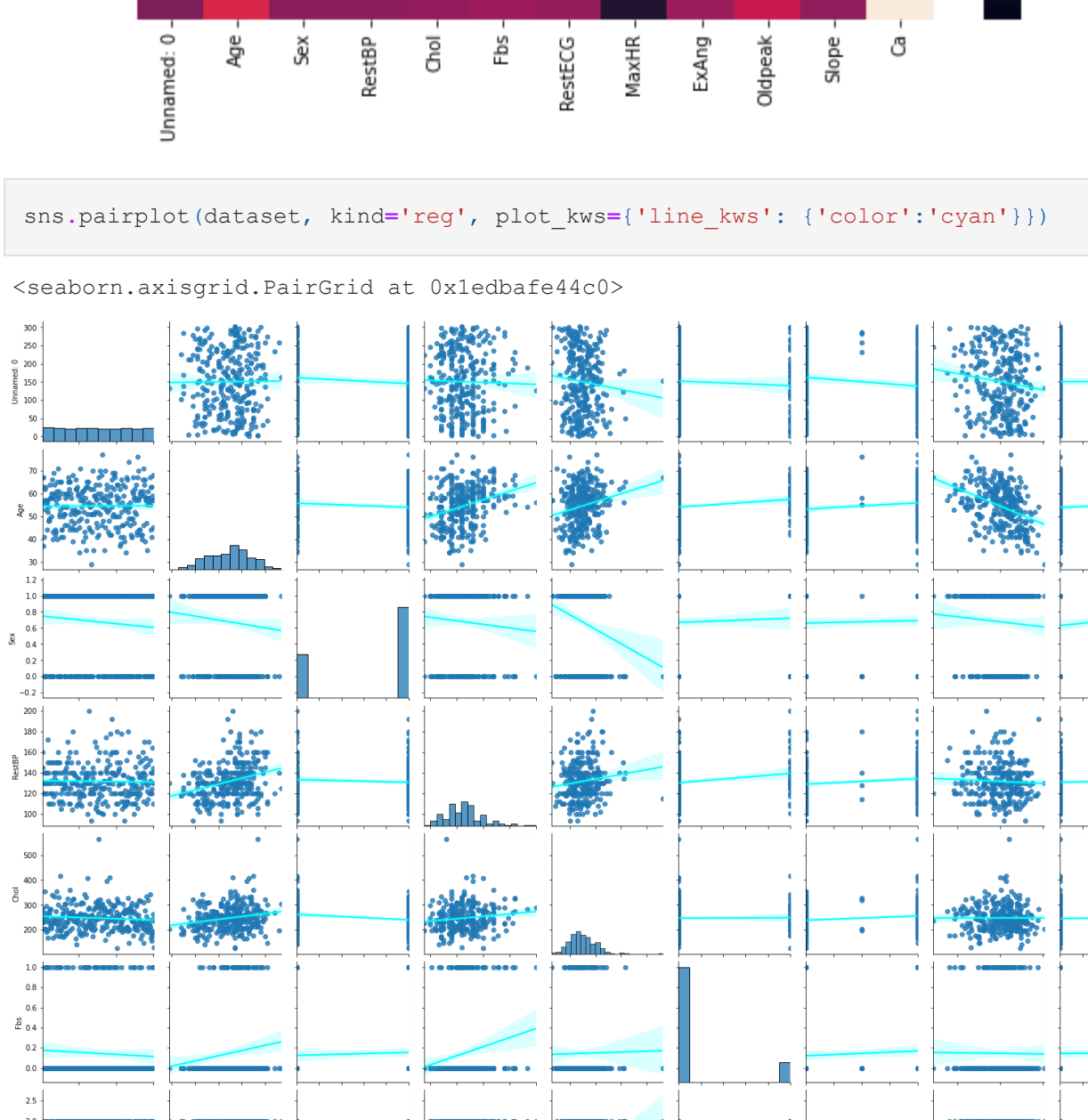


In [12]:
```python
plt.figure(figsize=(10,8))
sns.lmplot(x='Age', y='Chol', data=dataset)
plt.show()
```

```
<Figure size 720x576 with 0 Axes>
```

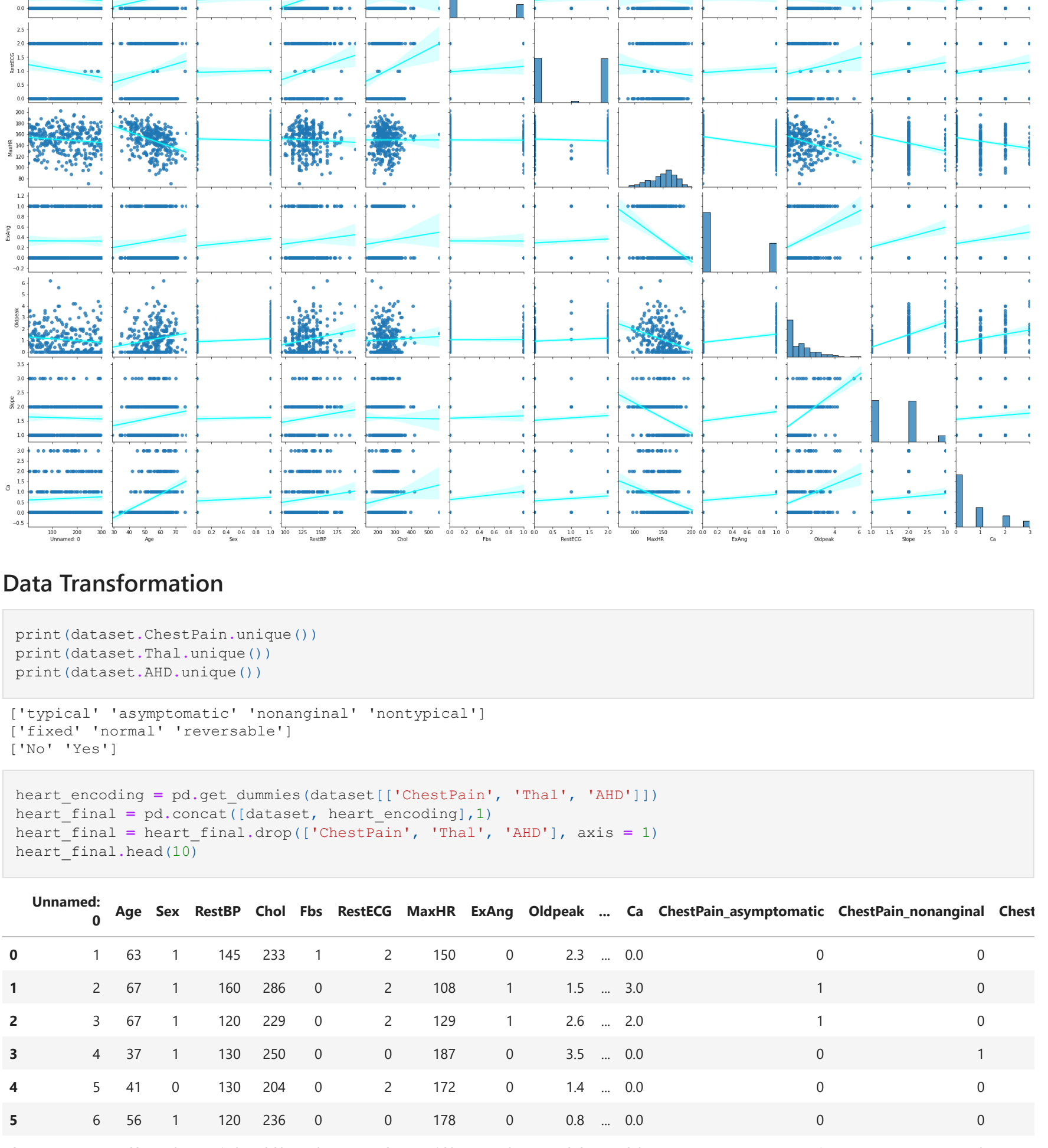


In [32]:
```python
plt.figure(figsize=(10,8))
sns.heatmap(dataset.corr(), annot=True)
plt.show()
```



In [14]:
```python
sns.pairplot(dataset, kind='reg', plot_kws={'line_kws': {'color':'cyan'}})
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x1edbafe44c0>
```



Data Transformation

In [15]:
```python
print(dataset.ChestPain.unique())
print(dataset.Thal.unique())
print(dataset.AHD.unique())
```

```
['typical' 'asymptomatic' 'nonanginal' 'nontypical']
['fixed' 'normal' 'reversable']
['No' 'Yes']
```

In [17]:
```python
heart_encoding = pd.get_dummies(dataset[['ChestPain', 'Thal', 'AHD']])
heart_final = pd.concat([dataset, heart_encoding],1)
heart_final = heart_final.drop(['ChestPain', 'Thal', 'AHD'], axis = 1)
heart_final.head(10)
```

Out[17]:

	Unnamed: 0	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	...	Ca	ChestPain_asymptomatic	ChestPain_nonanginal
0	1	63	1	145	233	1	2	150	0	2.3	...	0.0	0	0
1	2	67	1	160	286	0	2	108	1	1.5	...	3.0	1	0
2	3	67	1	120	229	0	2	129	1	2.6	...	2.0	1	0
3	4	37	1	130	250	0	0	187	0	3.5	...	0.0	0	1
4	5	41	0	130	204	0	2	172	0	1.4	...	0.0	0	0
5	6	56	1	120	236	0	0	178	0	0.8	...	0.0	0	0
6	7	62	0	140	268	0	2	160	0	3.6	...	2.0	1	0
7	8	57	0	120	354	0	0	163	1	0.6	...	0.0	1	0
8	9	63	1	130	254	0	2	147	0	1.4	...	1.0	1	0
9	10	53	1	140	203	1	2	155	1	3.1	...	0.0	1	0

10 rows × 21 columns

In [18]:
```python
heart_final.Sex.value_counts()
```

Out[18]:
```
1    201
0     96
Name: Sex, dtype: int64
```

In [19]:
```python
pd.crosstab(heart_final.AHD_Yes,heart_final.Sex)
```

Out[19]:

Sex	0	1
AHD_Yes		
0	71	89
1	25	112

In [20]:
```python
heart_final.columns
```

Out[20]:
```
Index(['Unnamed: 0', 'Age', 'Sex', 'RestBP', 'Chol', 'Fbs', 'RestECG', 'MaxHR',
       'ExAng', 'Oldpeak', 'Slope', 'Ca', 'ChestPain_asymptomatic',
       'ChestPain_nonanginal', 'ChestPain_nontypical', 'ChestPain_typical',
       'Thal_fixed', 'Thal_normal', 'Thal_reversable', 'AHD_No', 'AHD_Yes'],
      dtype='object')
```

In [21]:
```python
heart_final.dtypes
```

Out[21]:
```
Unnamed: 0                  int64
Age                         int64
Sex                         int64
RestBP                      int64
Chol                        int64
Fbs                         int64
RestECG                     int64
MaxHR                       int64
ExAng                       int64
Oldpeak                   float64
Slope                       int64
Ca                        float64
ChestPain_asymptomatic      uint8
ChestPain_nonanginal        uint8
ChestPain_nontypical        uint8
ChestPain_typical           uint8
Thal_fixed                  uint8
Thal_normal                 uint8
Thal_reversable             uint8
AHD_No                      uint8
AHD_Yes                     uint8
dtype: object
```

In [22]:
```python
df = heart_final.drop('AHD_Yes', axis=1)
df_norm = (df-df.min())/(df.max()-df.min())
df_norm = pd.concat((df_norm, heart_final.AHD_Yes), 1)
```

In [23]:
```python
df_norm.head(10)
```

Out[23]:

	Unnamed: 0	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	...	Ca	ChestPain_asymptomatic	ChestPain_no
0	0.000000	0.708333	1.0	0.481132	0.244292	1.0	1.0	0.603053	0.0	0.370968	...	0.000000	0.0	
1	0.003322	0.791667	1.0	0.622642	0.365297	0.0	1.0	0.282443	1.0	0.241935	...	1.000000	1.0	
2	0.006645	0.791667	1.0	0.245283	0.235160	0.0	1.0	0.442748	1.0	0.419355	...	0.666667	1.0	
3	0.009967	0.166667	1.0	0.339623	0.283105	0.0	0.0	0.885496	0.0	0.564516	...	0.000000	0.0	
4	0.013289	0.250000	0.0	0.339623	0.178082	0.0	1.0	0.770992	0.0	0.225806	...	0.000000	0.0	
5	0.016611	0.562500	1.0	0.245283	0.251142	0.0	0.0	0.816794	0.0	0.129032	...	0.000000	0.0	
6	0.019934	0.687500	0.0	0.433962	0.324201	0.0	1.0	0.679389	0.0	0.580645	...	0.666667	1.0	
7	0.023256	0.583333	0.0	0.245283	0.520548	0.0	0.0	0.702290	1.0	0.096774	...	0.000000	1.0	
8	0.026578	0.708333	1.0	0.339623	0.292237	0.0	1.0	0.580153	0.0	0.225806	...	0.333333	1.0	
9	0.029900	0.500000	1.0	0.433962	0.175799	1.0	1.0	0.641221	1.0	0.500000	...	0.000000	1.0	

10 rows × 21 columns

In [24]:
```python
X = df_norm.drop(['AHD_Yes', 'Unnamed: 0'], axis=1)
Y = df_norm.AHD_Yes
```

Train Test Split

In [ ]:
```python
from sklearn.model_selection import train_test_split
```

In [25]:
```python
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2 ,random_state=1)
```

KNN Modelling

In [ ]:
```python
from sklearn.neighbors import KNeighborsClassifier
```

In [26]:
```python
classifier = KNeighborsClassifier(n_neighbors=5, metric="minkowski", p=2)
classifier.fit(X_train, y_train)
```

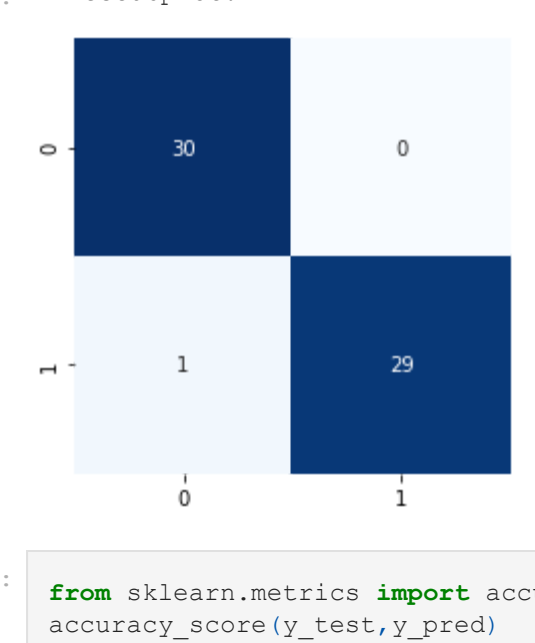
Out[26]:
```
KNeighborsClassifier()
```

In [27]:
```python
y_pred = classifier.predict(X_test)
y_pred
```

Out[27]:
```
array([0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1], dtype=uint8)
```

In [28]:
```python
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_mat, square=True, annot=True, cmap='Blues', fmt='d', cbar=False)
```

Out[28]:
```
<AxesSubplot:>
```



In [30]:
```python
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[30]:
```
0.9833333333333333
```

Conclusion : Accuracy of trained model is 98.34%