How to choose domain?

Domain should be chosen on the basis of our interest and understanding. Since all the further procedures would depend on what domain we choose, it is important that we have some prior knowledge of the domain, the methodologies followed, the developments and the need of the our domain.For this we can try working on data from different domains and narrow down to one for our future work. Along with this we need to assess the resources that are available- datasets, tools, softwares, etc and then finalise our domain and project plan.

How to choose the dataset?

Once we have finalised the domain, it is important to choose the right dataset. The source needs to be reliable and the dataset should contain as much information/variables as necessary for the analysis. We need to consider all the factors that make a good dataset- less missing values, reliability of source, more information. We can even collect data on our own based on the requirement, money and time we have. For example, I have taken a Medical Dataset here to predict the possibility of heart failure.

In [57]:

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

In [58]:

```
df=pd.read_csv('D:/Downloads/heart.csv')
df.head()
```

Out[58]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exercis
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	
4	54	M	NAP	150	195	0	Normal	122	

How to understand data?

Data can be understood in various ways. We can start by understanding our variables, their type and their descriptive statistics. We can also try and understand if their are any codes which have meanings/have been encoded. We can also understand the data by checking missing values, outliers, etc and see if they carry a meaning. Then, we can start visualising our data to understand its distribution, the relations between variables, the affect on our target variable,etc. For example:

- Univariate Analysis

o Understanding mean, median, mode and skewness of data

o Plots like boxplot, count plot etc

• Bi Variate Anaysis

o correlations

o Plots like boxplot, line plot etc

• Multivariate Analysis, etc

Here, we have a basic knowledge of the variables regarding their units and meaning. It is as follows:

Attribute Information

Age: age of the patient [years]

Sex: sex of the patient [M: Male, F: Female]

ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]

RestingBP: resting blood pressure [mm Hg]

Cholesterol: serum cholesterol [mm/dl]

FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]

RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]

MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]

ExerciseAngina: exercise-induced angina [Y: Yes, N: No]

Oldpeak: oldpeak = ST [Numeric value measured in depression]

ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]

HeartDisease: output class [1: heart disease, 0: Normal]

In [59]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

We can see that there are no null values. We can also see the various data types of the variables.

In [60]:

```
df.nunique()
```

Out[60]:

```
Age                50
Sex                 2
ChestPainType       4
RestingBP          67
Cholesterol       222
FastingBS           2
RestingECG          3
MaxHR             119
ExerciseAngina      2
Oldpeak            53
ST_Slope            3
HeartDisease        2
dtype: int64
```

Here, we can see that there are certain categorical values- the ones with unique values such as 2,3,4. So we can have a basic idea about our variables.

In [61]:

```
df.describe(include='all')
```

Out[61]:

```
              Age  Sex  ChestPainType   RestingBP  Cholesterol   FastingBS  RestingECG
count  918.000000  918            918  918.000000   918.000000  918.000000         918  91
unique        NaN    2              4         NaN          NaN         NaN           3
top           NaN    M            ASY         NaN          NaN         NaN      Normal
freq          NaN  725            496         NaN          NaN         NaN         552
mean    53.510893  NaN            NaN  132.396514   198.799564    0.233115         NaN  13
std      9.432617  NaN            NaN   18.514154   109.384145    0.423046         NaN   2
min     28.000000  NaN            NaN    0.000000     0.000000    0.000000         NaN   6
25%     47.000000  NaN            NaN  120.000000   173.250000    0.000000         NaN  12
50%     54.000000  NaN            NaN  130.000000   223.000000    0.000000         NaN  13
75%     60.000000  NaN            NaN  140.000000   267.000000    0.000000         NaN  15
max     77.000000  NaN            NaN  200.000000   603.000000    1.000000         NaN  20
```

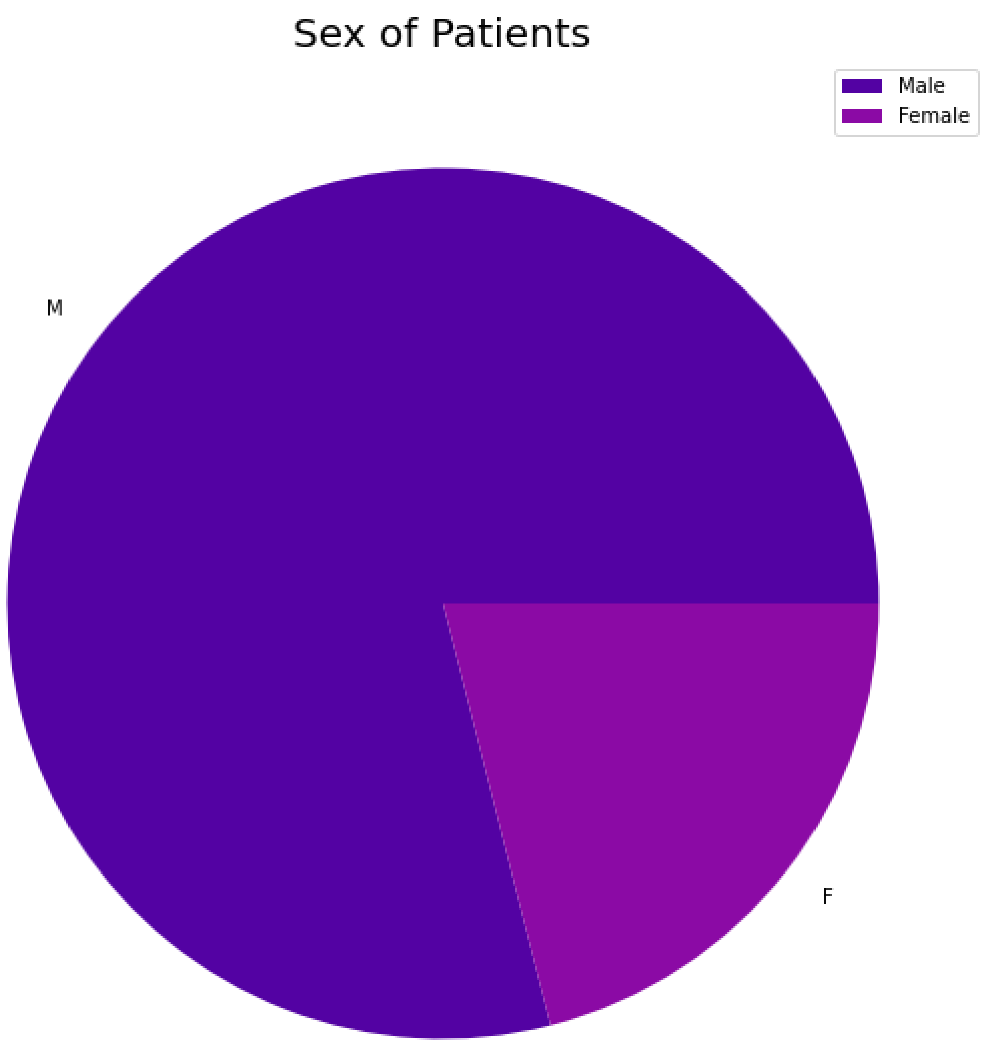
To get a statistical summary of our variables. The categorical variables would not have a mean, median ,etc. This would give us an idea about the average values, quartiles, range, etc.

In [62]:

```python
df['Sex'].value_counts()
sex = df['Sex'].value_counts()
f, ax = plt.subplots(figsize=(10,10))
chart1 = plt.pie(sex, labels = sex.index, colors = sns.color_palette("plasma"))
plt.title('Sex of Patients', fontsize=20)
plt.legend(['Male', 'Female'])
```

Out[62]:

```
<matplotlib.legend.Legend at 0x13da095baf0>
```



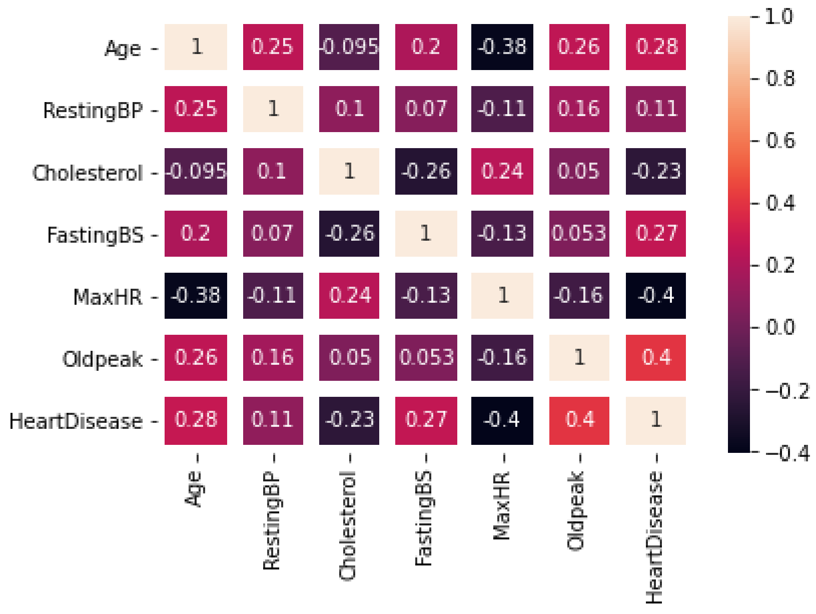
We can see that our data is male dominated and hence might produce biased results.

In [63]:

```
corr = df.corr()
sns.heatmap(corr, annot = True, linewidth=7)
```

Out[63]:

```
<AxesSubplot:>
```



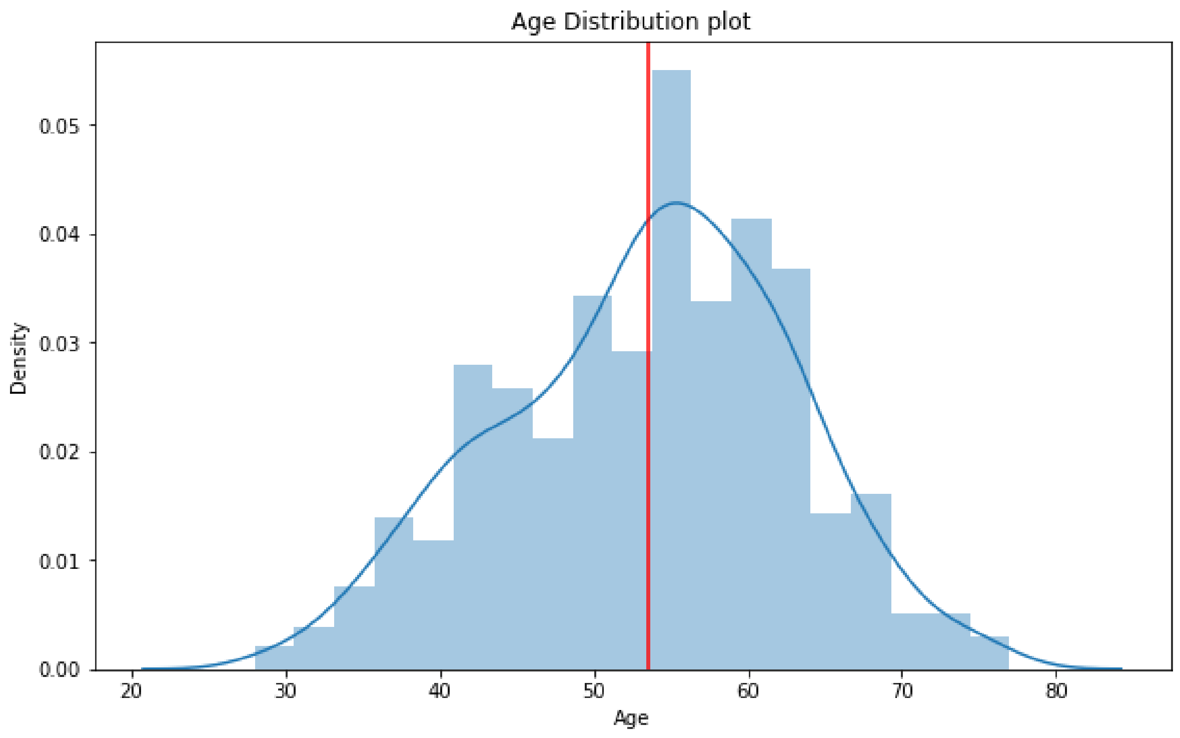
In [64]:

```python
#Distribution plot
Age_m = df['Age'].describe()
print('Mean Age of patients: {}'.format(Age_m['mean']))

plt.figure(figsize = (10, 6))
sns.distplot(df['Age'])
plt.title('Age Distribution plot')
plt.axvline(Age_m['mean'], linestyle = '-', color = "red")

```

```
Mean Age of patients: 53.510893246187365

D:\ANACONDA\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)
```

Out[64]:

```
<matplotlib.lines.Line2D at 0x13da0ac4a60>
```



We can see how the age is approximately normally distributed but is a little left skewed.

In [ ]:

```

```

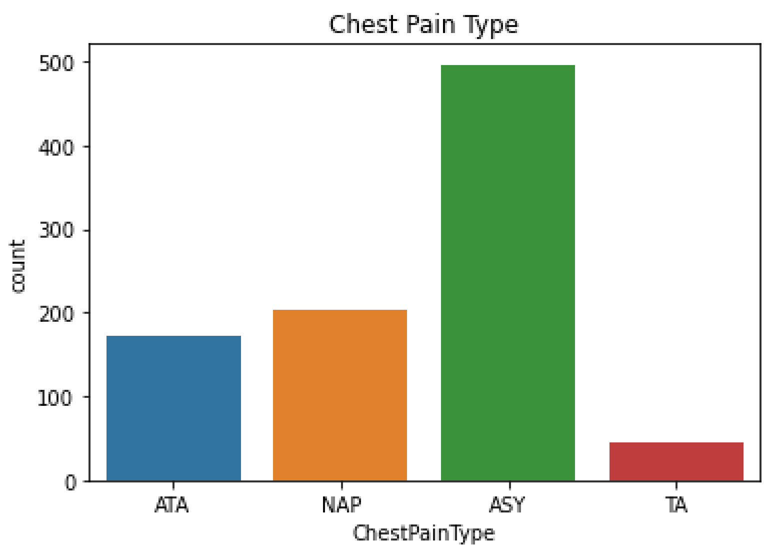
In [65]:

```python
sns.countplot(df['ChestPainType'])
plt.title('Chest Pain Type')
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only vali
d positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[65]:

```
Text(0.5, 1.0, 'Chest Pain Type')
```

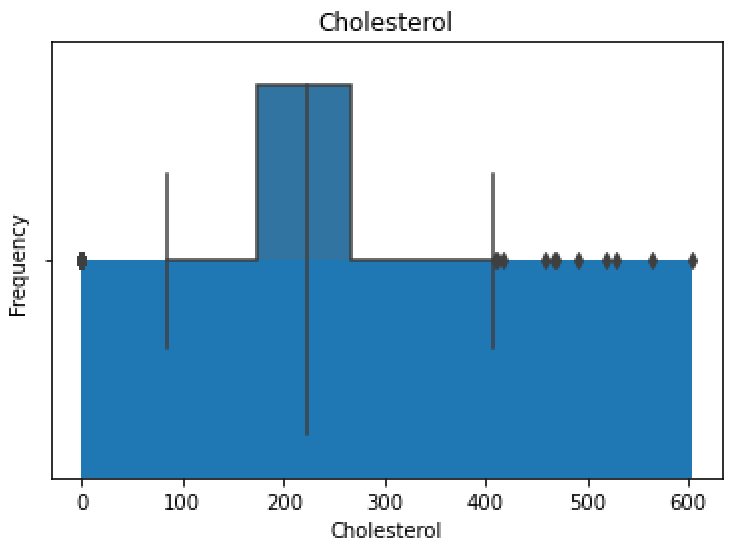


Most people experience ASY(Asymptomatic) Chest Pain Type followed by NAP(Non-Anginal Pain)

In [66]:

```
df['Cholesterol'].plot(kind = 'hist')
plt.title('Cholesterol')
print(sns.boxplot(df["Cholesterol"]))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)

D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only vali
d positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



We see that there are values which are clearly outliers and hence needs to be removed.We remove values above 420.

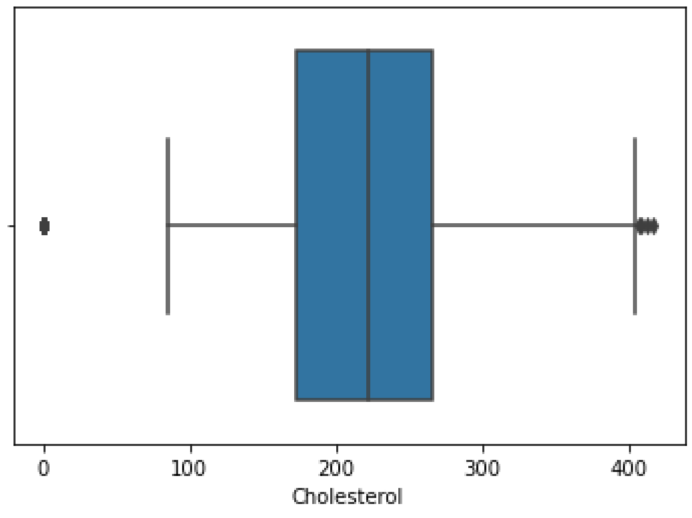
In [67]:

```
df.drop(df[df['Cholesterol'] > 420 ].index , axis = 0 , inplace = True)
sns.boxplot(df["Cholesterol"])
```

```
D:\ANACONDA\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only vali
d positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[67]:

```
<AxesSubplot:xlabel='Cholesterol'>
```



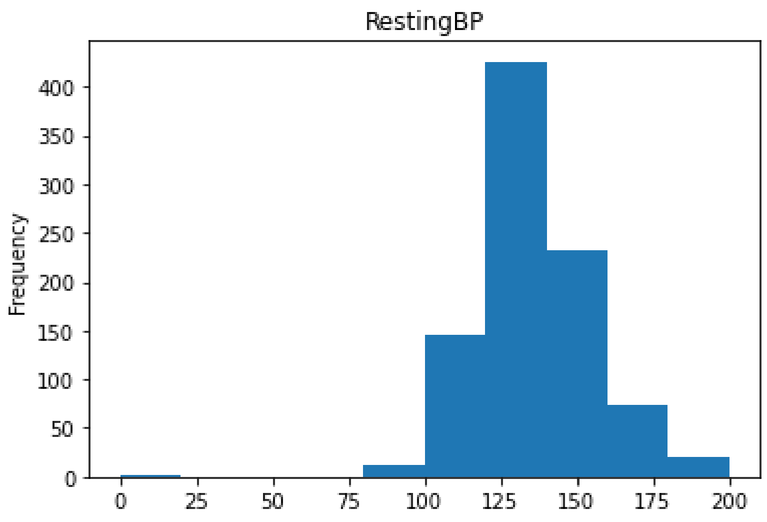
We can see that their our values which are outliers in the "Cholestrol" variable and need to be dealt with properly.

In [68]:

```
df['RestingBP'].plot(kind = 'hist')
plt.title('RestingBP')
```

Out[68]:

```
Text(0.5, 1.0, 'RestingBP')
```



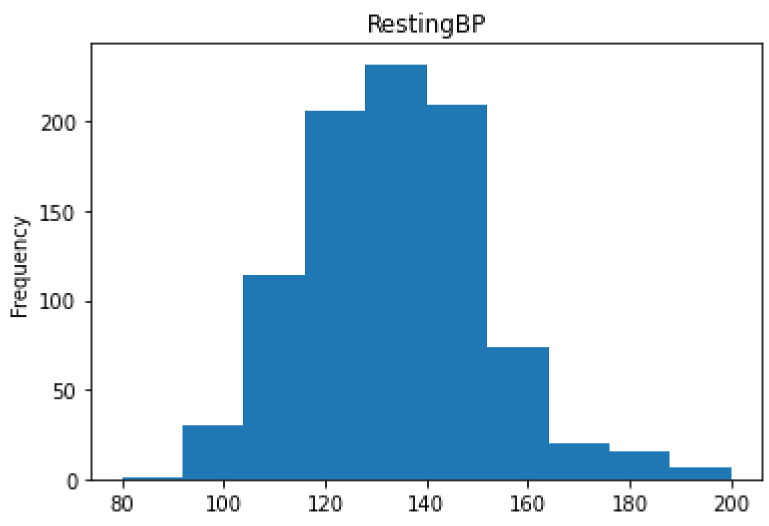
We see that value less than 80 is an outlier and affects distribution significantly. We will remove it.

In [69]:

```
df.drop(df[df['RestingBP'] < 75].index , axis = 0 , inplace = True)
df['RestingBP'].plot(kind = 'hist') #
plt.title('RestingBP')
```

Out[69]:

```
Text(0.5, 1.0, 'RestingBP')
```

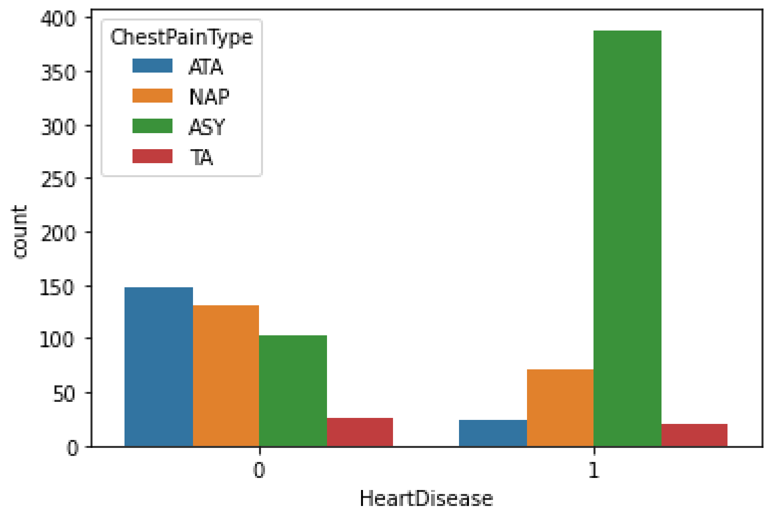


We remove the ones which are not logically or medically possible.

Bivariate Analysis

In [70]:

```
sns.countplot(data = df, x = 'HeartDisease' , hue = 'ChestPainType' );
```



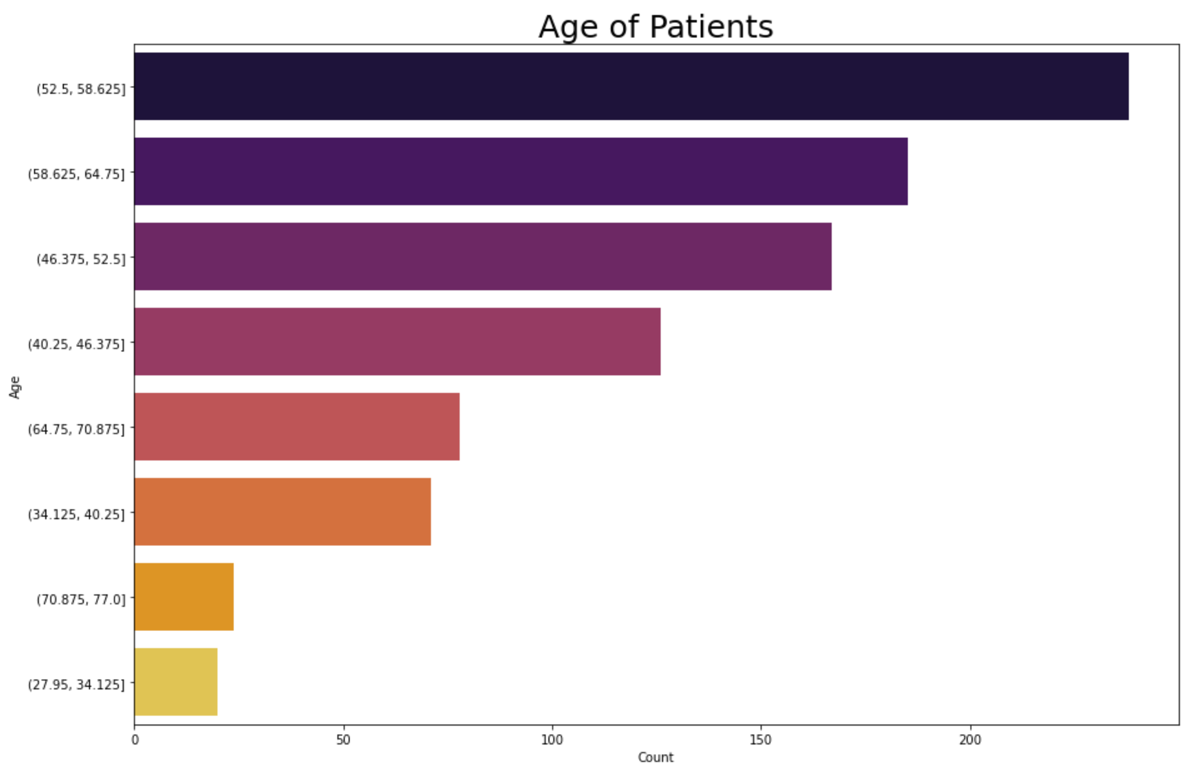
We see that most people that have ASY chest pain have heart disease

In [71]:

```python
age_bin = df['Age'].value_counts(bins=8).sort_values(ascending=False)
plot1=plt.subplots(figsize=(15,10))
plot1 = sns.barplot(x=age_bin, y=age_bin.index, palette="inferno")
plt.title('Age of Patients', fontsize=25)
plot1.set(xlabel='Count', ylabel='Age')
```

Out[71]:

```
[Text(0.5, 0, 'Count'), Text(0, 0.5, 'Age')]
```



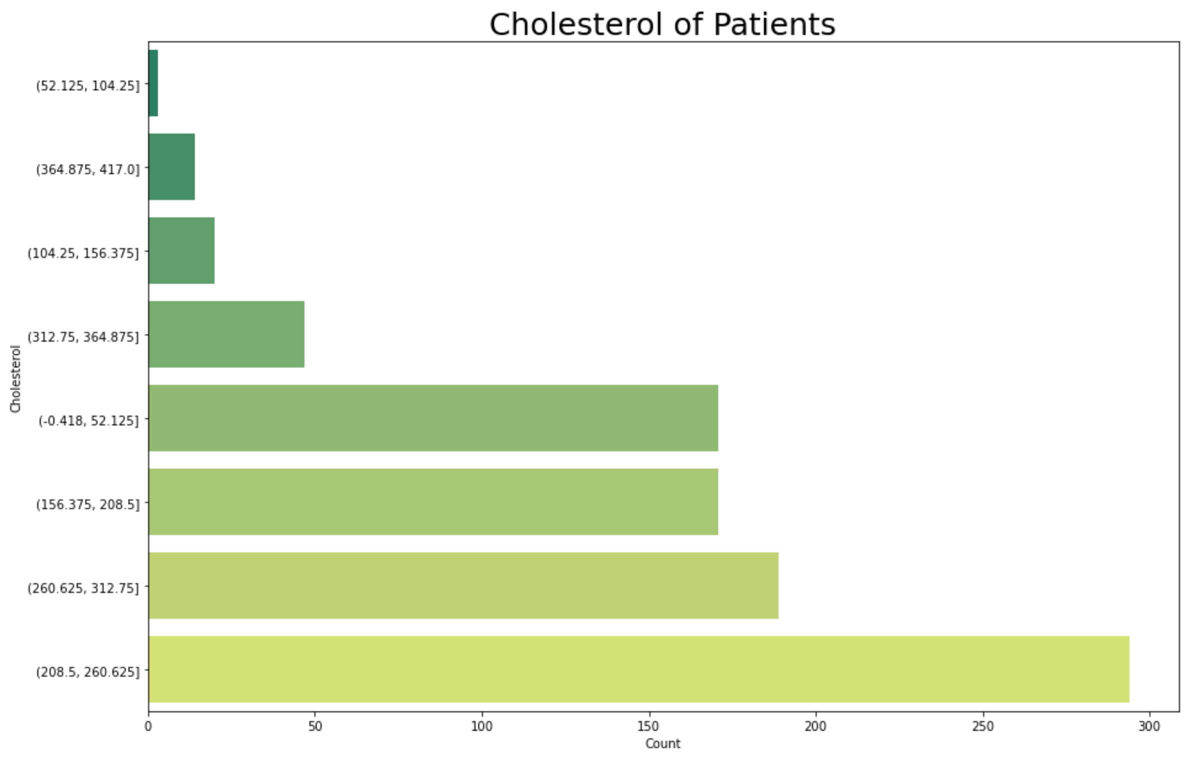
Most patients are in late adulthood i.e between 52-65 (approx). The younger group 26-35 are the least affected.

In [72]:

```
col_bin = df['Cholesterol'].value_counts(bins=8).sort_values(ascending=True)
plot1=plt.subplots(figsize=(15,10))
plot1 = sns.barplot(x=col_bin, y=col_bin.index, palette="summer")
plt.title('Cholesterol of Patients', fontsize=25)
plot1.set(xlabel='Count', ylabel='Cholesterol')
```

Out[72]:

```
[Text(0.5, 0, 'Count'), Text(0, 0.5, 'Cholesterol')]
```



No. of patients are more with higher cholesterol levels.

How to identify the proper learning method (Supervised

(classification/prediction), Unsupervised learning [Clustering/Association Rule Mining) and Reinforcement Learning (optimization/strategy/ agent based learning]

We can make use following steps to identify right learning method :

Categorize the Problem

a. Categorize by the input:

If it is a labeled data, it's a supervised learning problem. If it is not labelled, it is unsupervised learning. If it needs to train itself again and again to learn, it is reinforcement learning.

b. Categorize by output:

If the output of the model is a number, it's a regression problem.

If it is categorical, we can use Logistic Regression.

If we have to put it into certain categories, it is classification problem.

How to identify proper model [SVM/NaiveBayes/ K-means etc.]

Finding the available algorithms

The accuracy and complexity of the model.

How long does it take to build, train, and test the model?

Does the model meet the business goal?

Following are the consideration for choosing right ML algorithms:

a. Size of training data

b. Interpretability of data

c. Speed and Training time

d. Types of learning

e. Comparing the performance of model

In [ ]:

```

```