In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
df = pd.read_csv("Heart Disease data.csv")
```

In [6]:
```
df
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3

1025 rows × 14 columns

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

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	

In [8]:
```
# changing column names

df.columns = ['Age', 'Sex', 'Chest Pain Type', 'Resting BP', 'Serum Cholestoral (mg
```

In [9]:
```
df.head(10)
```

Out[9]:

	Age	Sex	Chest Pain Type	Resting BP	Serum Cholestoral (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Resting Electro-Cardiographic Results	Max Heart Rate Achieved	Exercise Induced Angina	Old Peal
0	52	1	0	125	212	0	1	168	0	1.(
1	53	1	0	140	203	1	0	155	1	3.1
2	70	1	0	145	174	0	1	125	1	2.6
3	61	1	0	148	203	0	1	161	0	0.(
4	62	0	0	138	294	1	1	106	0	1.9
5	58	0	0	100	248	0	0	122	0	1.(
6	58	1	0	114	318	0	2	140	0	4.4
7	55	1	0	160	289	0	0	145	1	0.8
8	46	1	0	120	249	0	0	144	0	0.8
9	54	1	0	122	286	0	0	116	1	3.2

In [10]:
```
df.shape
```

Out[10]:
```
(1025, 14)
```

In [11]:
```
df.describe()
```

Out[11]:

	Age	Sex	Chest Pain Type	Resting BP	Serum Cholestoral (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Cardi
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	102
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
Age                                   0
Sex                                   0
Chest Pain Type                       0
Resting BP                            0
Serum Cholestoral (mg/dl)             0
Fasting Blood Sugar > 120 mg/dl       0
Resting Electro-Cardiographic Results 0
Max Heart Rate Achieved               0
Exercise Induced Angina               0
Old-Peak                              0
Slope                                 0
No. of Major Vessels (fluoroscopy)    0
Thalium Scintigraphy                  0
Presense of Heart Disease             0
dtype: int64
```

Data Cleaning

There is not any NaN values present in our dataset so we do not need to do perform data cleaning

Data Analyzing

Percentage of people having Heart Disease

In [13]:
```
num= df.groupby('Presense of Heart Disease').size()
num
```

Out[13]:
```
Presense of Heart Disease
0    499
1    526
dtype: int64
```

In [14]:
```
df.head()
```

Out[14]:

	Age	Sex	Chest Pain Type	Resting BP	Serum Cholestoral (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Resting Electro-Cardiographic Results	Max Heart Rate Achieved	Exercise Induced Angina	Old Peak
0	52	1	0	125	212	0	1	168	0	1.0
1	53	1	0	140	203	1	0	155	1	3.1
2	70	1	0	145	174	0	1	125	1	2.6
3	61	1	0	148	203	0	1	161	0	0.0
4	62	0	0	138	294	1	1	106	0	1.9

In [15]:
```
#Converting Numerical Data into Categorical Data

def heart_disease(row):
    if row==0:
        return 'Absence'
    elif row==1:
        return 'Presence'
```

In [16]:
```
#Applying converted data into our dataset with new column - Heart_Disease

df['Heart_Disease']=df['Presense of Heart Disease'].apply(heart_disease)
df.head(10)
```

Out[16]:

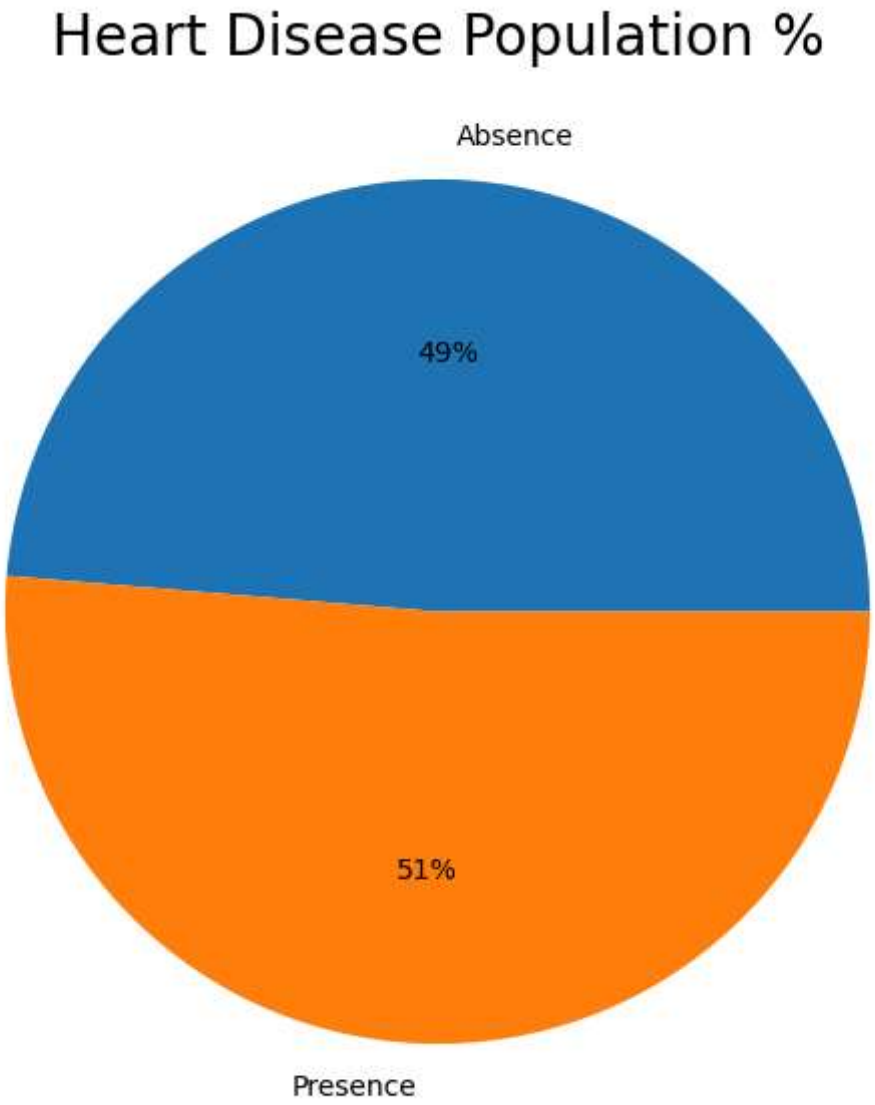
	Age	Sex	Chest Pain Type	Resting BP	Serum Cholestoral (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Resting Electro-Cardiographic Results	Max Heart Rate Achieved	Exercise Induced Angina	Old Peak
0	52	1	0	125	212	0	1	168	0	1.0
1	53	1	0	140	203	1	0	155	1	3.1
2	70	1	0	145	174	0	1	125	1	2.6
3	61	1	0	148	203	0	1	161	0	0.0
4	62	0	0	138	294	1	1	106	0	1.9
5	58	0	0	100	248	0	0	122	0	1.0
6	58	1	0	114	318	0	2	140	0	4.4
7	55	1	0	160	289	0	0	145	1	0.8
8	46	1	0	120	249	0	0	144	0	0.8
9	54	1	0	122	286	0	0	116	1	3.2

HEART DISEASE

In [17]:
```
hd=df.groupby('Heart_Disease')['Presense of Heart Disease'].count()
hd
```

Out[17]:
```
Heart_Disease
Absence     499
Presence    526
Name: Presense of Heart Disease, dtype: int64
```

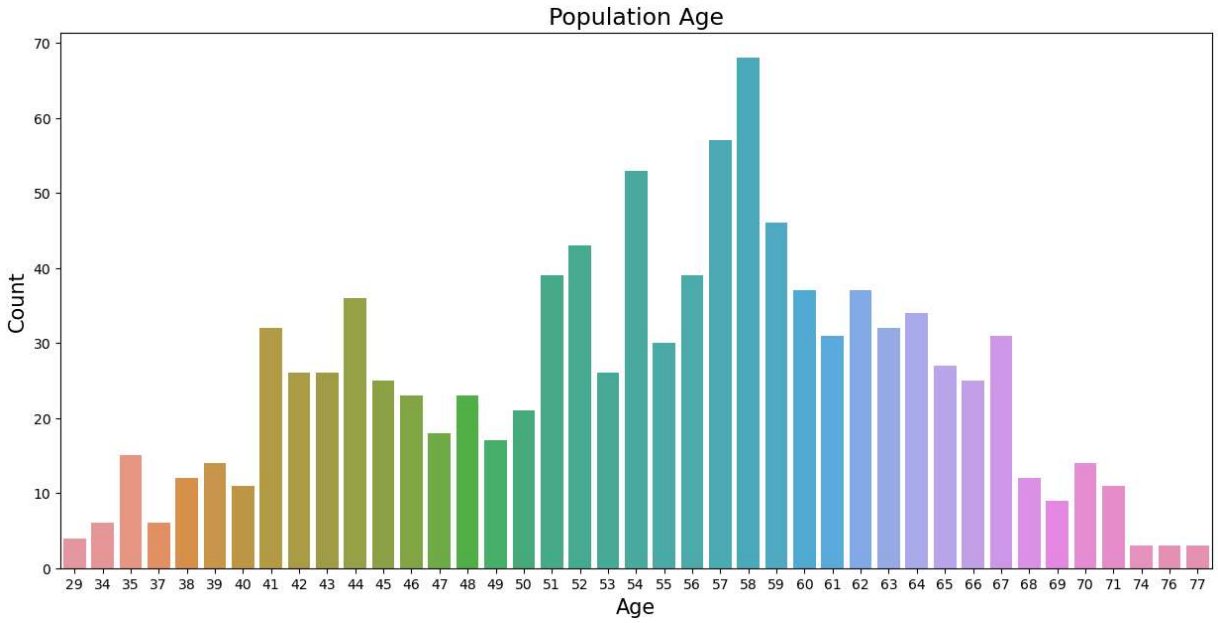
In [18]:
```
#Pie Chart Creation of Heart Disease Population % using MatplotLib

plt.figure(figsize=(10,7))
plt.pie(hd, labels=['Absence','Presence'], autopct='%0.0f%%')
plt.title('Heart Disease Population %', fontsize=20)
plt.show()
```



From the overall population, people having heart disease (49%) are lesser than those who have heart disease(51%)

AGE

In [19]:
```
#Countplot Creation of Population Age using MatplotLib and Seaborn

plt.figure(figsize=(15,7))
sns.countplot(x='Age', data=df)
plt.title('Population Age', fontsize=17)
plt.xlabel('Age', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.show()
```



In [20]:
```
#Statistical Analysis

Min_Age=df['Age'].min()
Max_Age=df['Age'].max()
Mean_Age=df['Age'].mean()
print("Minimum Age =",Min_Age)
print("Maximum Age =",Max_Age)
print("Mean Age =",Mean_Age)
```

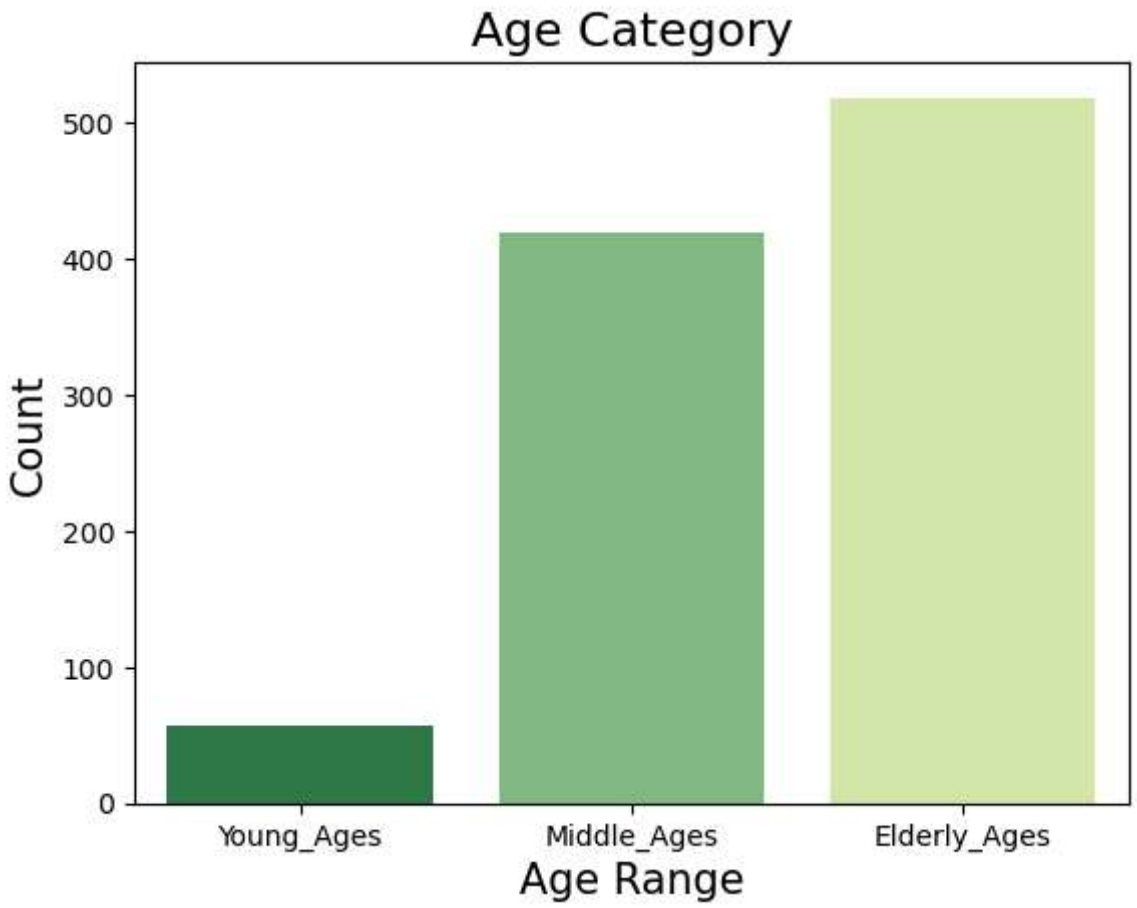
```
Minimum Age = 29
Maximum Age = 77
Mean Age = 54.43414634146342
```

In [21]:
```
#Categorical Analysis

Young_Ages=df[(df['Age']>=29) & (df['Age']<40)]
Middle_Ages=df[(df['Age']>=40) & (df['Age']<55)]
Elderly_Ages=df[(df['Age']>55)]
print('Young Ages =',len(Young_Ages))
print('Middle Ages =',len(Middle_Ages))
print('Elderly Ages =',len(Elderly_Ages))
```

```
Young Ages = 57
Middle Ages = 419
Elderly Ages = 519
```

In [22]:
```
#Bar Plot Creation of Age Category using MatplotLib and Seaborn

sns.barplot(x=['Young_Ages','Middle_Ages','Elderly_Ages'], y=[len(Young_Ages), len(
plt.title('Age Category', fontsize=17)
plt.xlabel('Age Range', fontsize=15)
plt.ylabel('Count', fontsize=15)
plt.show()
```

```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1765: FutureWarning:
unique with argument that is not not a Series, Index, ExtensionArray, or np.ndarray
is deprecated and will raise in a future version.
  order = pd.unique(vector)
```



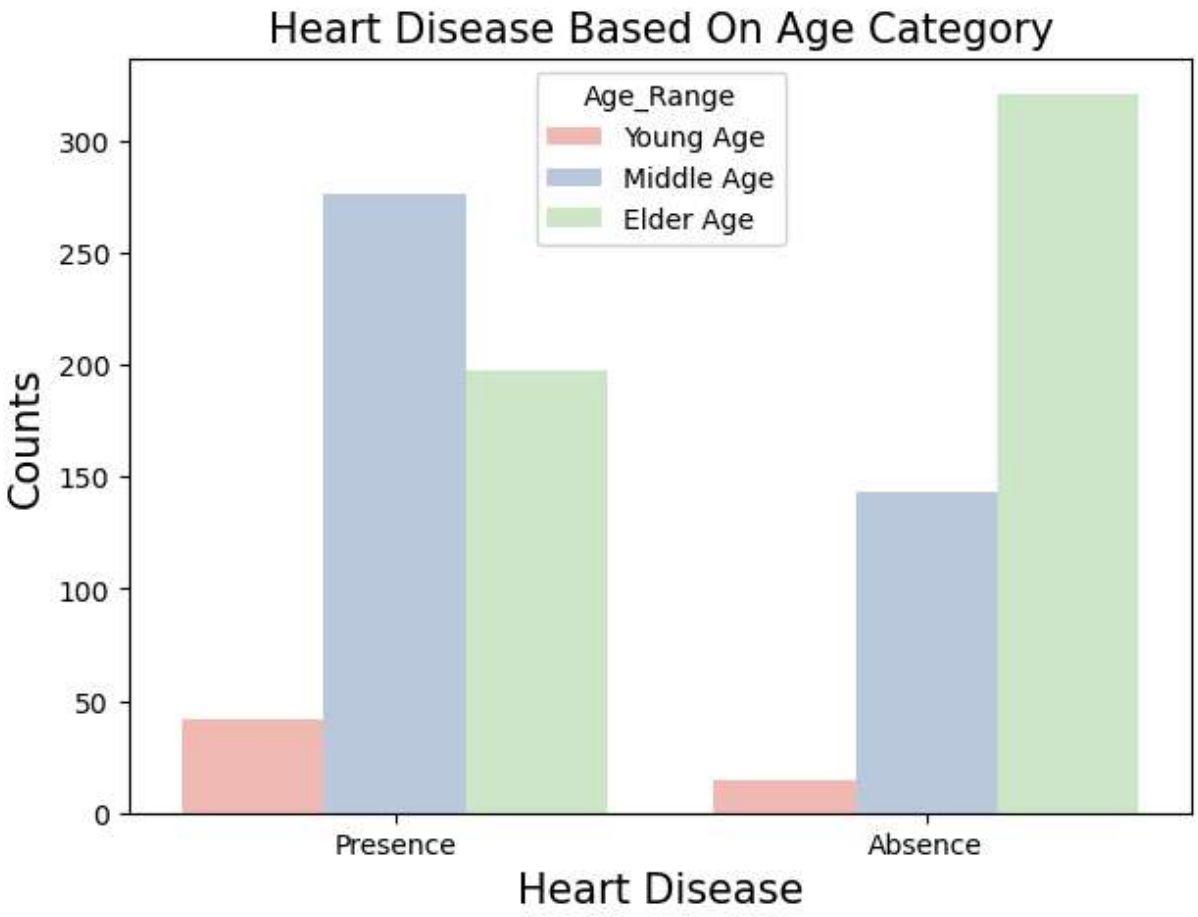
In [23]:
```
#Converting Numerical Data into Categorical Data

def age_range(row):
    if row>=29 and row<40:
        return 'Young Age'
    elif row>=40 and row<55:
        return 'Middle Age'
    elif row>55:
        return 'Elder Age'
```

In [24]:
```
#Applying converted data into our dataset with new column - Age_Range

df['Age_Range']=df['Age'].apply(age_range)
df.head()
```

Out[24]:

	Age	Sex	Chest Pain Type	Resting BP	Serum Cholestoral (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Resting Electro-Cardiographic Results	Max Heart Rate Achieved	Exercise Induced Angina	Old Peal
0	52	1	0	125	212	0	1	168	0	1.(
1	53	1	0	140	203	1	0	155	1	3.1
2	70	1	0	145	174	0	1	125	1	2.6
3	61	1	0	148	203	0	1	161	0	0.(
4	62	0	0	138	294	1	1	106	0	1.9

In [25]:
```
#Count Plot Creation of Heart Disease Based On Age Category using MatplotLib and Se

plt.figure(figsize=(7,5))
hue_order=['Young Age', 'Middle Age', 'Elder Age']
sns.countplot(x='Heart_Disease', hue='Age_Range', data=df, order=['Presence','Absen
plt.title('Heart Disease Based On Age Category', fontsize=15)
plt.xlabel('Heart Disease', fontsize=15)
plt.ylabel('Counts', fontsize=15)
plt.show()
```



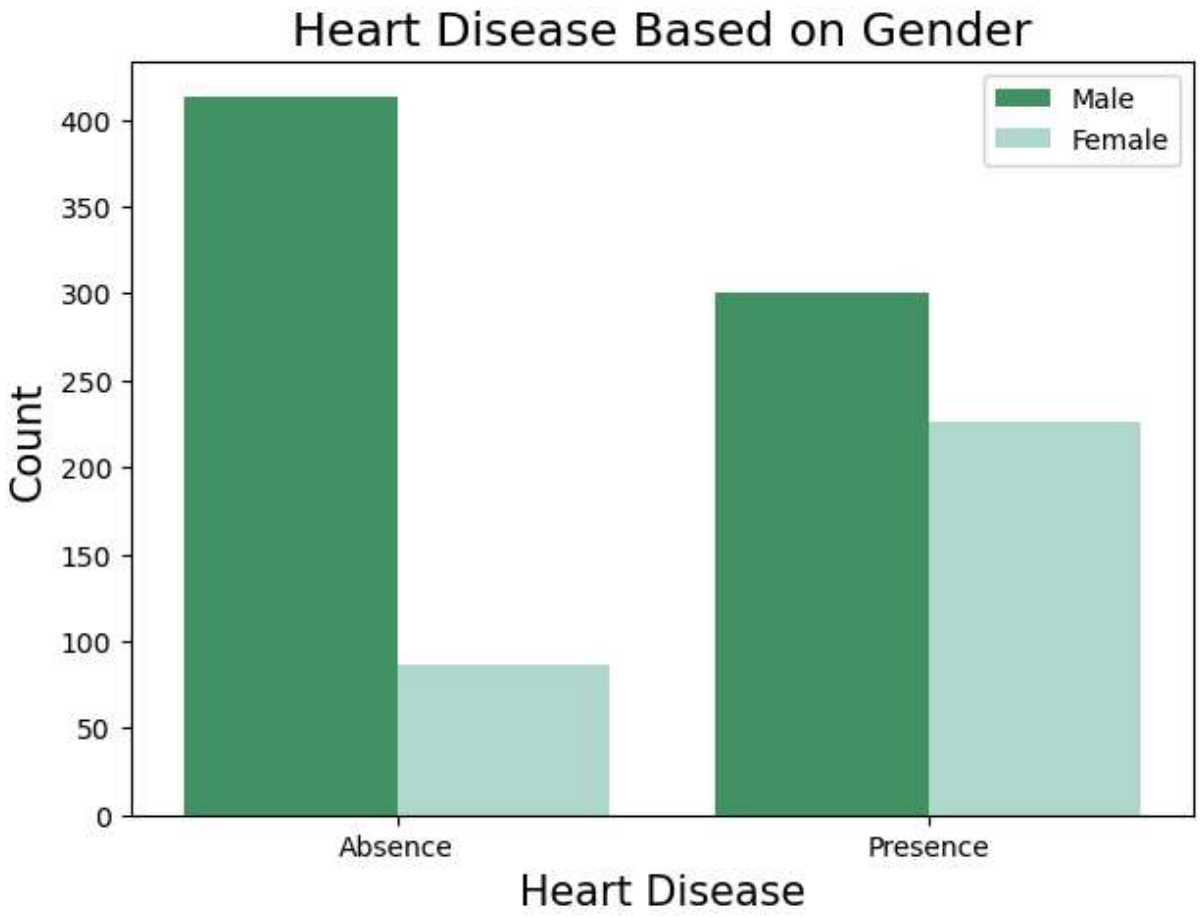
GENDER

In [26]:
```
#Converting Numerical Data into Categorical Data

def gender(row):
    if row==1:
        return 'Male'
    elif row==0:
        return 'Female'


#Applying converted data into our dataset with new column - sex1

df['Sex1']=df['Sex'].apply(gender)
df.head()
```

Out[26]:

	Age	Sex	Chest Pain Type	Resting BP	Serum Cholestoral (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Resting Electro-Cardiographic Results	Max Heart Rate Achieved	Exercise Induced Angina	Old Peak
0	52	1	0	125	212	0	1	168	0	1.0
1	53	1	0	140	203	1	0	155	1	3.1
2	70	1	0	145	174	0	1	125	1	2.6
3	61	1	0	148	203	0	1	161	0	0.0
4	62	0	0	138	294	1	1	106	0	1.9

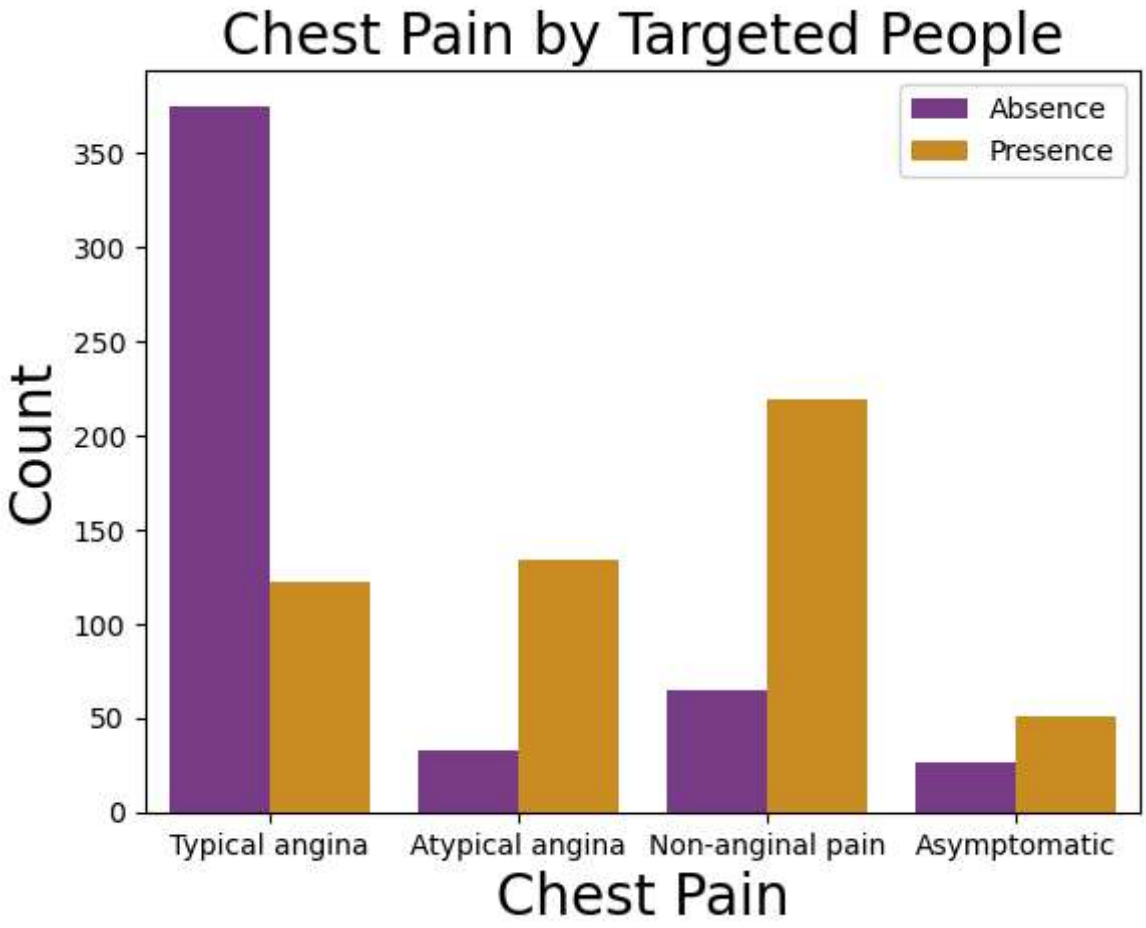
In [27]:
```
#Count Plot Creation of Heart Disease Based on Gender using MatplotLib and Seaborn

plt.figure(figsize=(7,5))
sns.countplot(x=df['Heart_Disease'], hue='Sex1', data=df, palette='BuGn_r')
plt.xlabel('Heart Disease', fontsize=15)
plt.ylabel('Count',fontsize=15)
plt.legend(labels=['Male','Female'])
plt.title('Heart Disease Based on Gender',fontsize=17)
plt.show()
```



We can see that Females are more prone to Heart Disease

CHEST PAIN

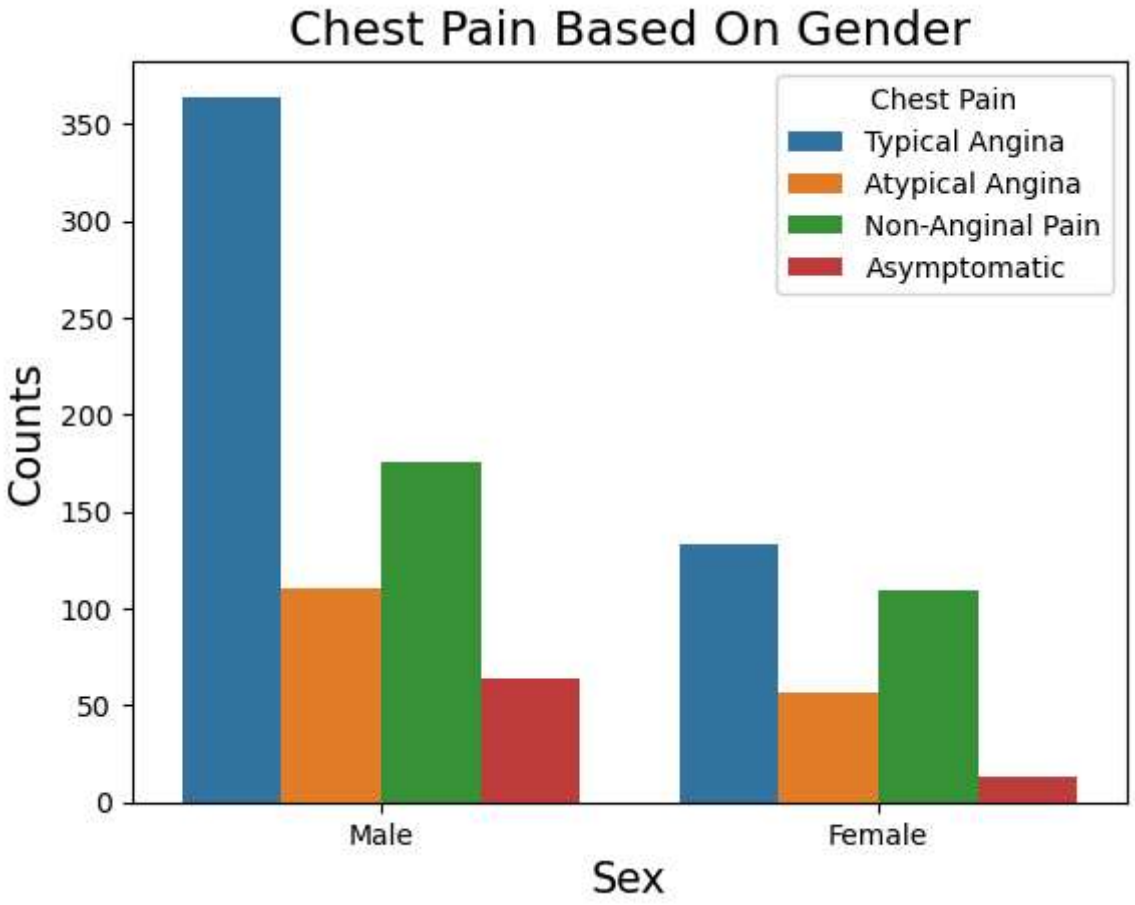
In [28]:
```
#Count Plot Creation of Chest Pain Experienced using MatplotLib and Seaborn

sns.countplot(data=df,x='Chest Pain Type',hue='Heart_Disease',palette='CMRmap')
plt.legend(labels=['Absence', 'Presence'])
plt.xticks(([0,1,2,3]),['Typical angina','Atypical angina', 'Non-anginal pain','Asy
plt.xlabel('Chest Pain',fontsize=20)
plt.ylabel('Count',fontsize=20)
plt.title('Chest Pain by Targeted People',fontsize=20)
plt.show()
```



It seems people having Non-anginal chest pain have a higher chance of heart disease

In [29]:
```
#Count Plot Creation of Chest Pain Based On Gender using MatplotLib and Seaborn

ax= sns.countplot(x='Sex1', hue='Chest Pain Type', data=df.apply(lambda x: x.astype
plt.title('Chest Pain Based On Gender', fontsize=17)
plt.xlabel('Sex', fontsize=15)
plt.ylabel('Counts', fontsize=15)

handles, labels = ax.get_legend_handles_labels()
```

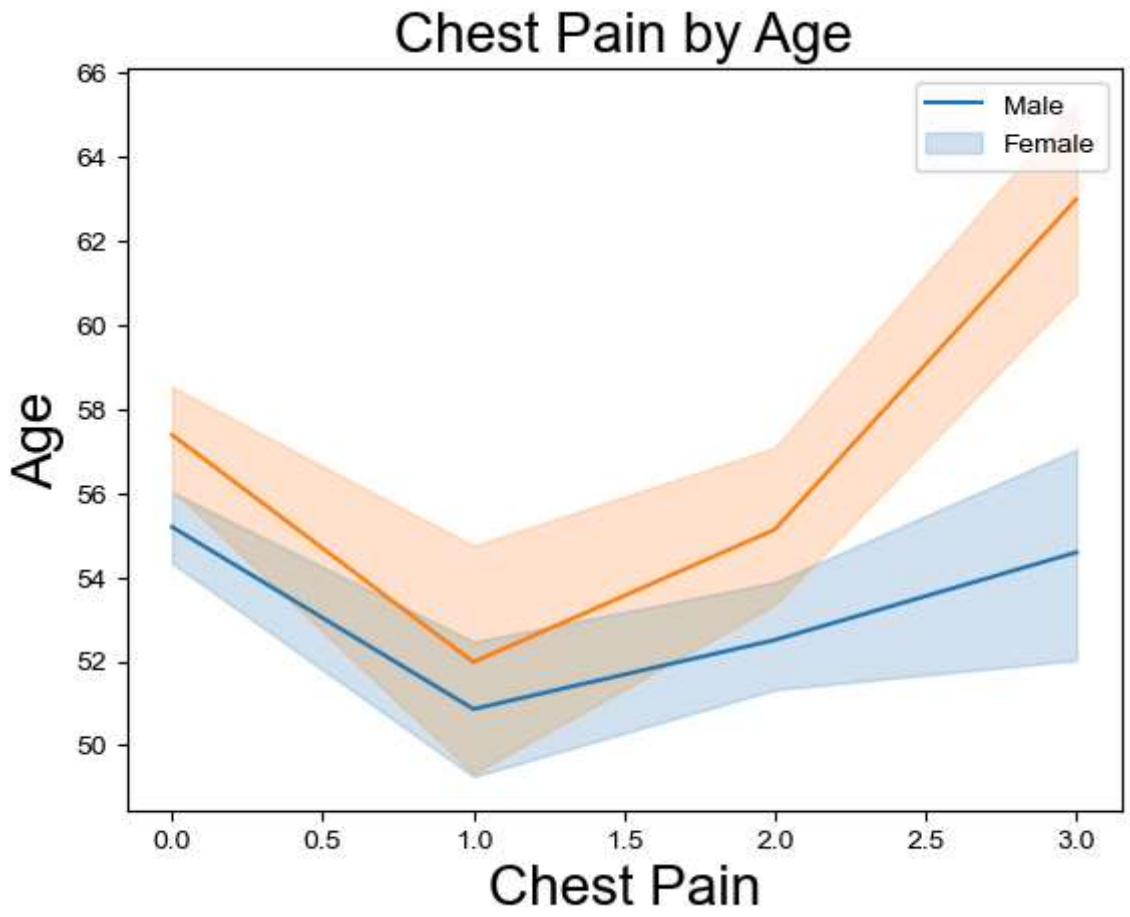
```
cp_labels = {0: 'Typical Angina', 1: 'Atypical Angina', 2: 'Non-Anginal Pain', 3: '
labels = [cp_labels[int(label)] for label in labels]  # Convert labels to strings
plt.legend(handles, labels, title='Chest Pain', loc='upper right')

plt.show()
```



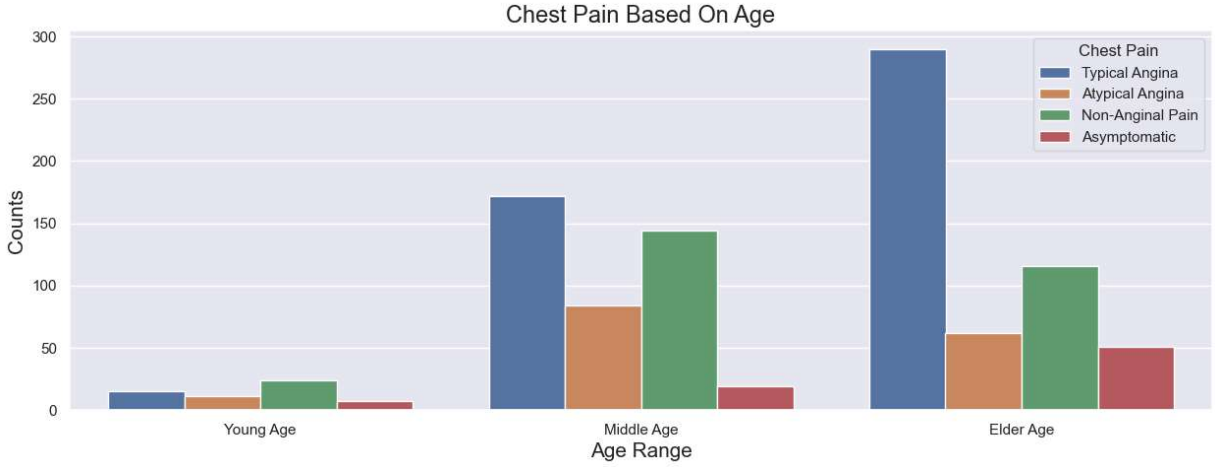
We can see that a higher number of men are suffering from Typical Angina type of Chest Pain

In [30]:
```
sns.lineplot(x='Chest Pain Type',data=df,y='Age',hue='Sex1')
plt.legend(labels=['Male','Female'])
sns.set(rc={'figure.figsize':(15,5)})
plt.xlabel('Chest Pain',fontsize=20)
plt.ylabel('Age',fontsize=20)
plt.title('Chest Pain by Age',fontsize=20)
plt.show()
```

```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [31]:
```
ax= sns.countplot(x='Age_Range', hue='Chest Pain Type', data=df.apply(lambda x: x.a
plt.title('Chest Pain Based On Age', fontsize=17)
plt.xlabel('Age Range', fontsize=15)
plt.ylabel('Counts', fontsize=15)

handles, labels = ax.get_legend_handles_labels()
cp_labels = {0: 'Typical Angina', 1: 'Atypical Angina', 2: 'Non-Anginal Pain', 3: '
labels = [cp_labels[int(label)] for label in labels]  # Convert labels to strings
plt.legend(handles, labels, title='Chest Pain', loc='upper right')

plt.show()
```



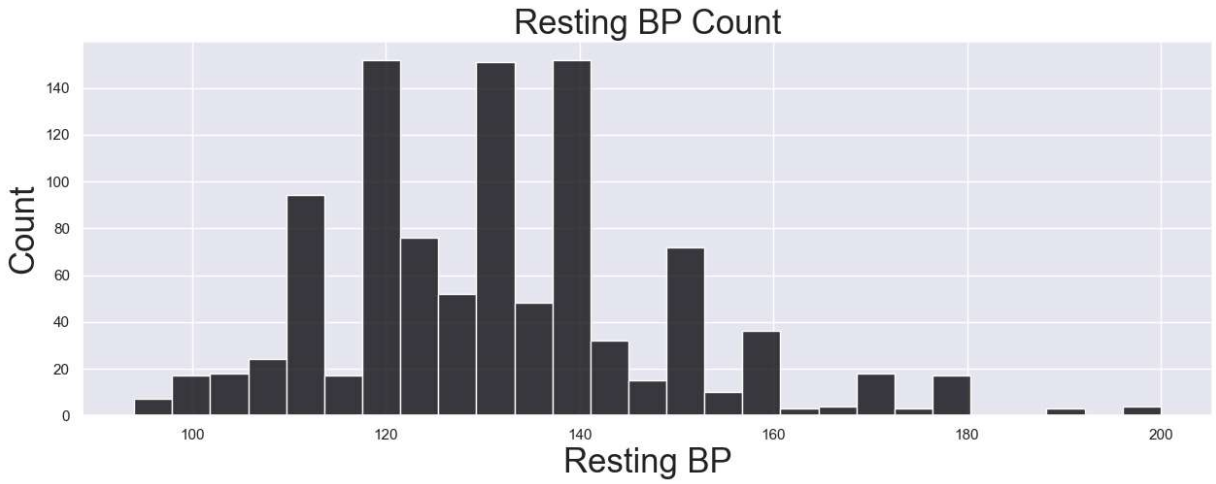
There is very high number of Typical Angina Pain in Elderly age Category

BLOOD PRESSURE

In [32]:
```
sns.histplot(x='Resting BP',data=df,color='black')
plt.xlabel('Resting BP',fontsize=25)
plt.ylabel('Count',fontsize=25)
plt.title('Resting BP Count',fontsize=25)
plt.show
```

```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[32]: <function matplotlib.pyplot.show(close=None, block=None)>



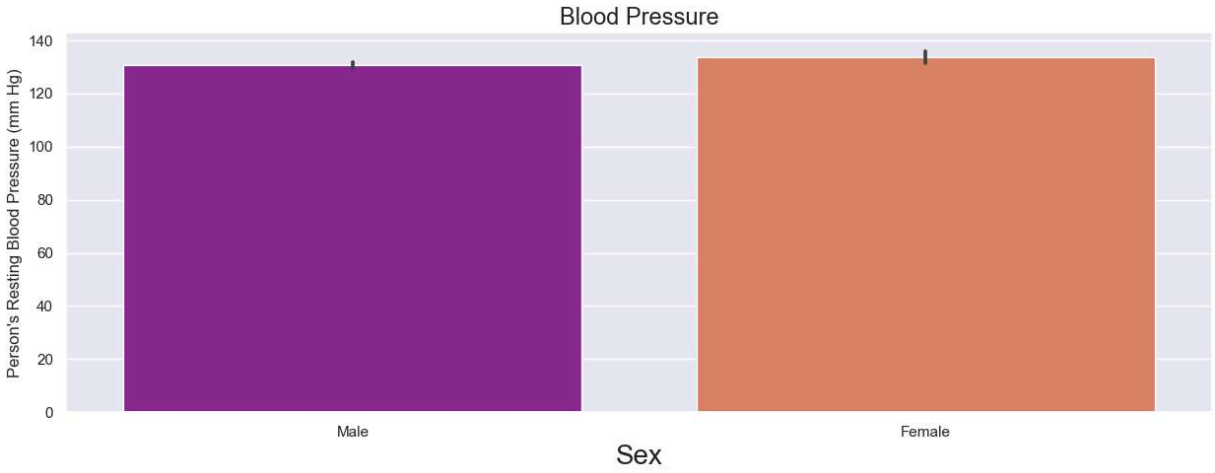
AVERAGE RESTING BLOOD PRESSURE

In [33]:
```
average_resting_blood_pressure = df['Resting BP'].mean()
print(f"Average resting blood pressure: {average_resting_blood_pressure.astype(int)
```

```
Average resting blood pressure: 131 BPM
```

GENDER

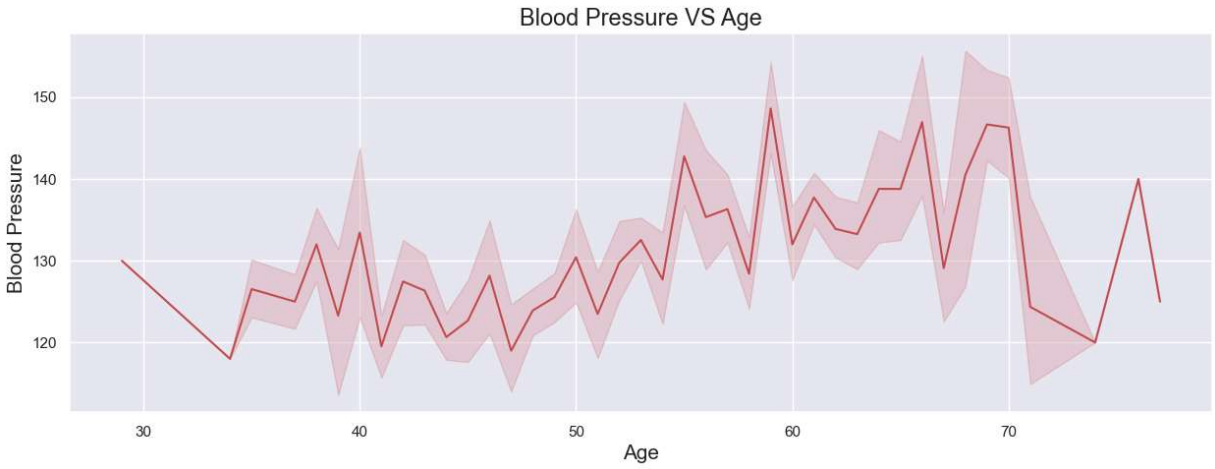
In [34]:
```
sns.barplot(x='Sex1', y='Resting BP', data=df, palette='plasma')
plt.title("Blood Pressure", fontsize=17)
plt.xlabel('Sex',fontsize=20)
plt.ylabel("Person's Resting Blood Pressure (mm Hg)", fontsize=12)
plt.show()
```



AGE

In [35]:
```
sns.lineplot(x='Age', y='Resting BP', data=df, color='r')
plt.title('Blood Pressure VS Age', fontsize=17)
plt.xlabel('Age', fontsize=15)
plt.ylabel('Blood Pressure', fontsize=15)
plt.show()
```

```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Blood Pressure increases between age of 50 to 60 and somehow continue the pattern till 70

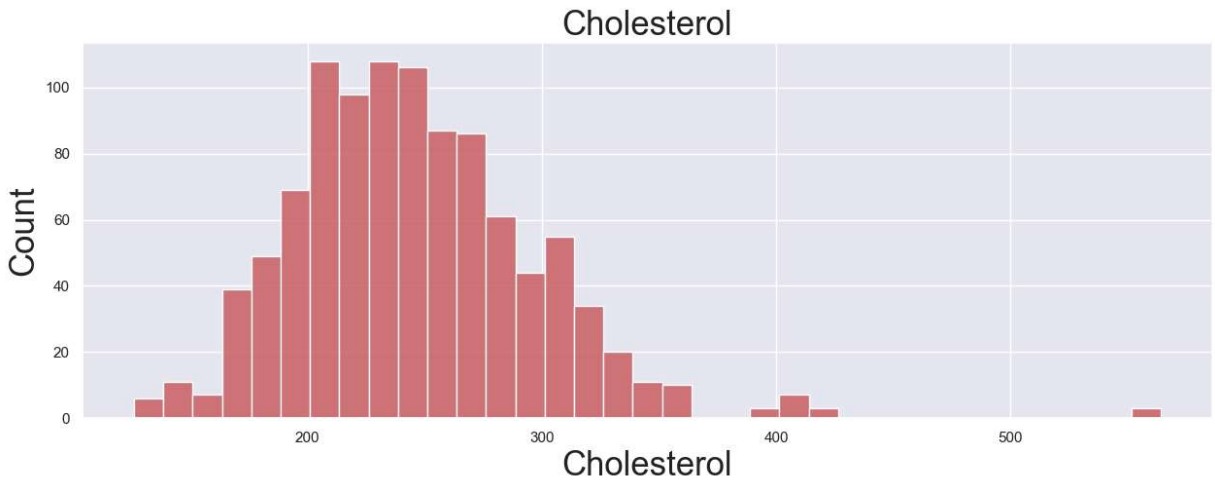
CHOLESTEROL

In [36]:
```
sns.histplot(x='Serum Cholestoral (mg/dl)',data=df,color='r')
plt.xlabel('Cholesterol',fontsize=25)
plt.ylabel('Count',fontsize=25)
```

```
plt.title('Cholesterol',fontsize=25)
plt.show
```

```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[36]: <function matplotlib.pyplot.show(close=None, block=None)>



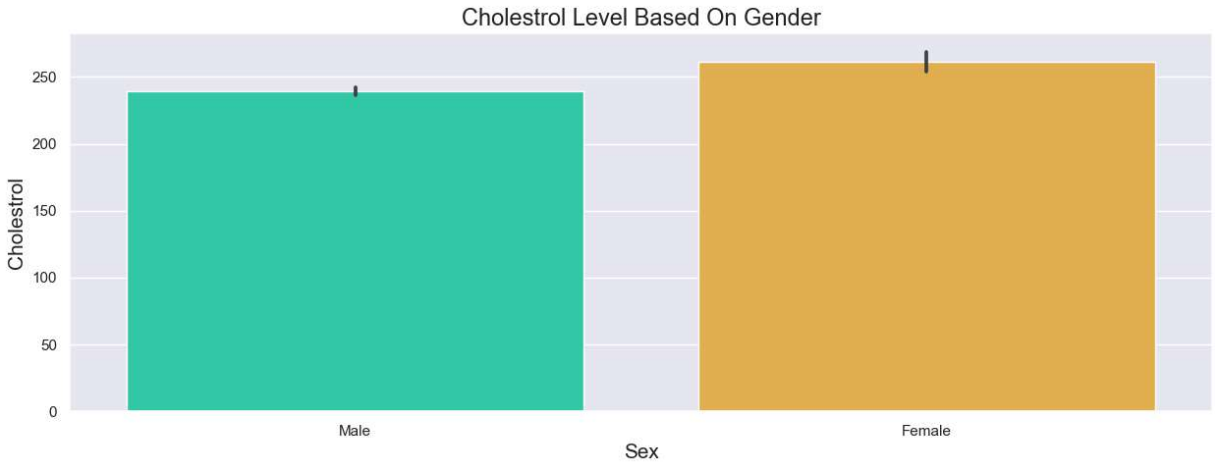
AVERAGE CHOLESTEROL

In [37]:
```
average_serum_cholesterol = df['Serum Cholestoral (mg/dl)'].mean()
print(f"Average serum cholesterol: {average_serum_cholesterol.astype(int)} mg/dl")
```

```
Average serum cholesterol: 246 mg/dl
```

GENDER

In [38]:
```
sns.barplot(x='Sex1', y='Serum Cholestoral (mg/dl)', data=df, palette='turbo')
plt.title("Cholestrol Level Based On Gender", fontsize=17)
plt.xlabel('Sex',fontsize=15)
plt.ylabel("Cholestrol", fontsize=15)
plt.show()
```

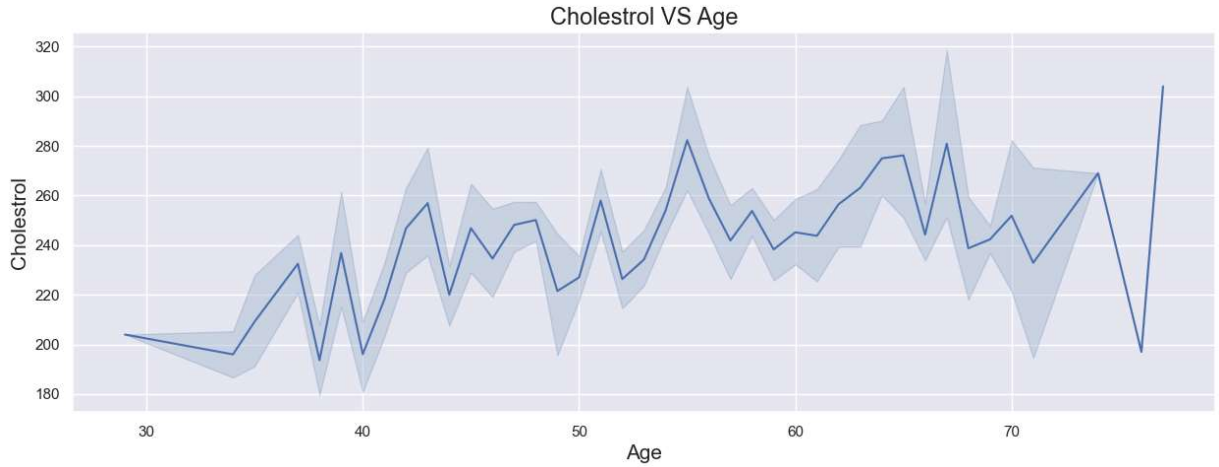


Females have little bit of higher cholesterol than males

AGE

In [39]:
```
sns.lineplot(x='Age', y='Serum Cholestoral (mg/dl)', data=df, color='b')
plt.title('Cholestrol VS Age', fontsize=17)
plt.xlabel('Age', fontsize=15)
plt.ylabel('Cholestrol', fontsize=15)
plt.show()
```

```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

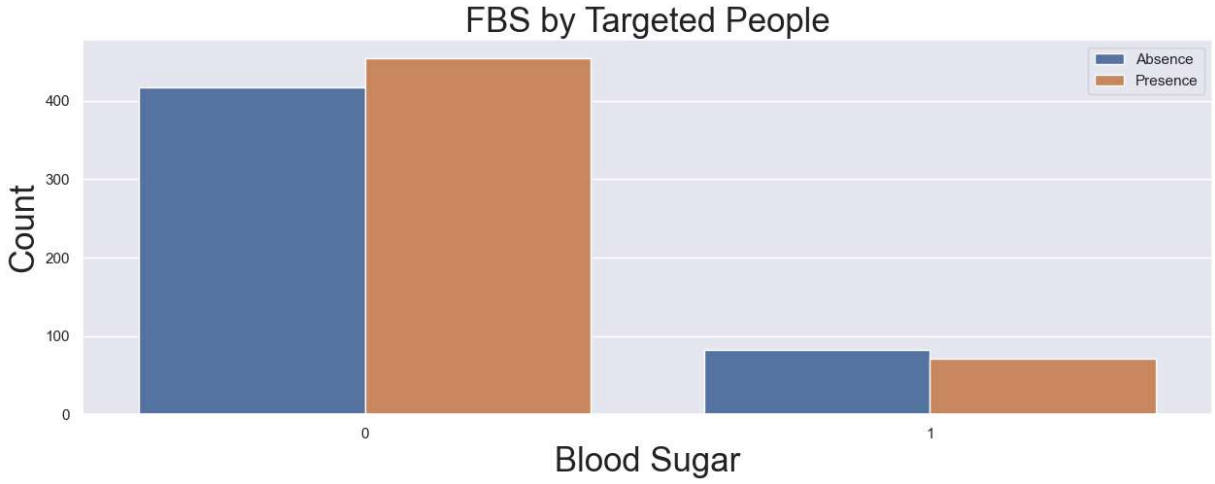


Cholestrol Increasing in the age group of 50-60

BLOOD SUGAR

In [40]:
```
sns.countplot(data=df,x='Fasting Blood Sugar > 120 mg/dl',hue='Heart_Disease')
plt.legend(labels=['Absence', 'Presence'])# fasting biood sugar distribution accord
plt.xlabel('Blood Sugar',fontsize=25)
plt.ylabel('Count',fontsize=25)
plt.title('FBS by Targeted People',fontsize=25)
plt.show
```

Out[40]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



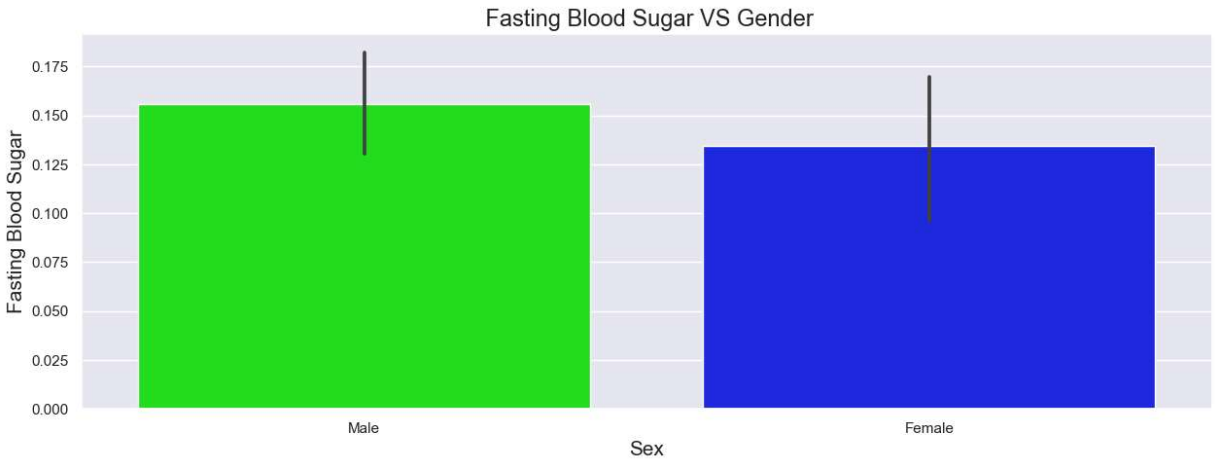
Percentage of individuals with high fasting blood sugar

In [41]:
```
count = 0
for i in range(len(df)):
    if df['Fasting Blood Sugar > 120 mg/dl'][i] == 1:
        count += 1

high_fasting_blood_sugar_rate = ((count/len(df)) * 100)
high_fasting_blood_sugar_rate = round(high_fasting_blood_sugar_rate, 1)
print(f"High fasting blood sugar rate: {high_fasting_blood_sugar_rate} %")
```

```
High fasting blood sugar rate: 14.9 %
```

GENDER

In [42]:
```
sns.barplot(y='Fasting Blood Sugar > 120 mg/dl', x='Sex1', data=df, palette='hsv')
plt.title(' Fasting Blood Sugar VS Gender', fontsize=17)
plt.xlabel('Sex', fontsize=15)
plt.ylabel('Fasting Blood Sugar', fontsize=15)
plt.show()
```

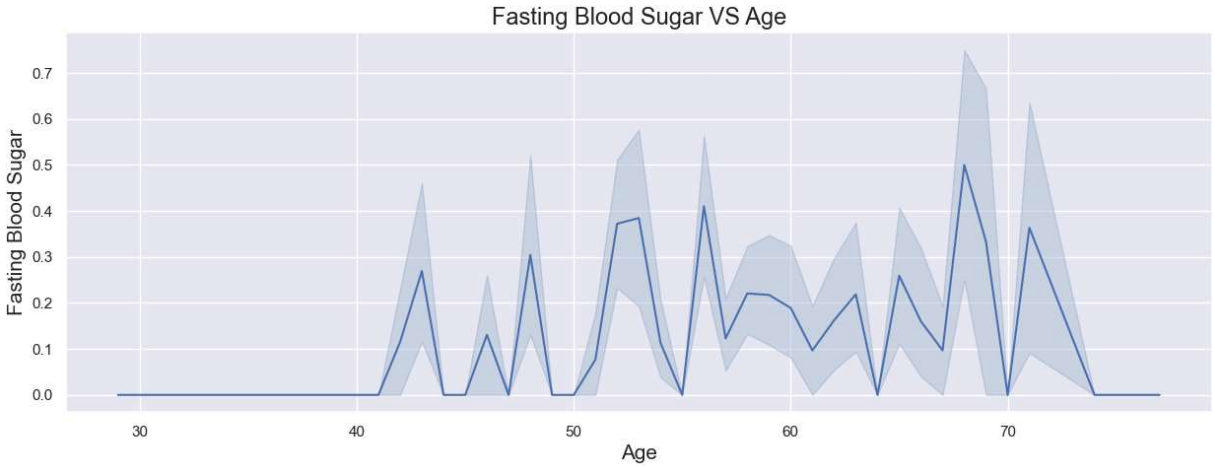


Males have high no of Fasting Blood Sugar over 120

AGE

In [43]:
```
sns.lineplot(x='Age', y='Fasting Blood Sugar > 120 mg/dl', data=df, color='b')
plt.title('Fasting Blood Sugar VS Age', fontsize=17)
plt.xlabel('Age', fontsize=15)
plt.ylabel('Fasting Blood Sugar', fontsize=15)
plt.show()
```

```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



MAXIMUM HEART RATE

In [44]:
```
female_count = 0
for i in range(len(df)):
    if df['Sex1'][i] == 0: # sex == 0 female, 1 male
        female_count += 1

print(f"Females in dataset: {female_count}")
print(f"Males in dataset: {len(df) - female_count}")

max_heart_rate = df.groupby('Sex')['Max Heart Rate Achieved'].max()
print(f"Max Heart Rate (female): {max_heart_rate[0]} BPM")
print(f"Max Heart Rate (male): {max_heart_rate[1]} BPM")
```

```
Females in dataset: 0
Males in dataset: 1025
Max Heart Rate (female): 192 BPM
Max Heart Rate (male): 202 BPM
```

ST DEPRESSION (OLD-PEAK)

Average ST depression

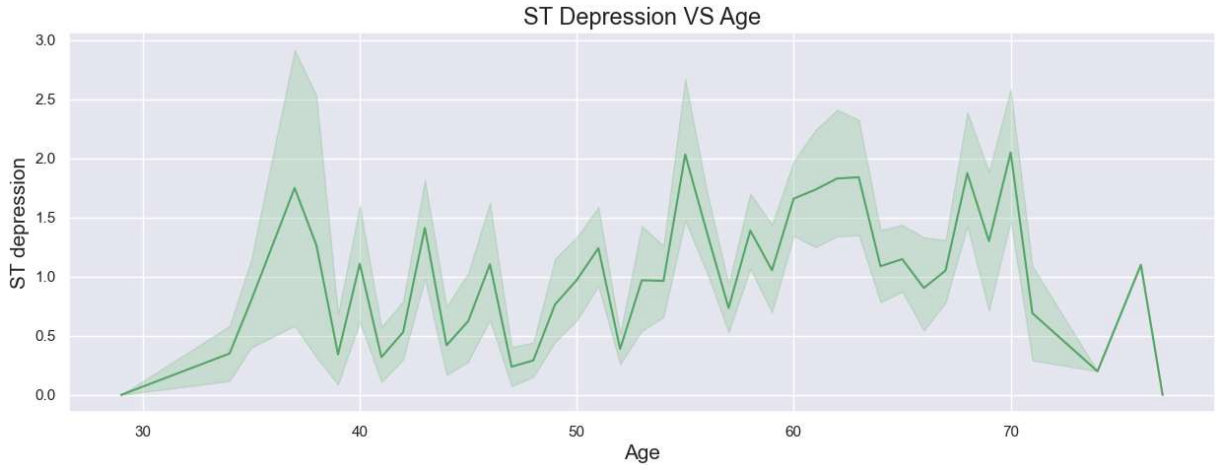
In [45]:
```
average_st_depression = df['Old-Peak'].mean()
print(f"Average ST depression: {average_st_depression.astype(int)}")
```

```
Average ST depression: 1
```

AGE

In [46]:
```
sns.lineplot(x='Age', y='Old-Peak', data=df, color='g')
plt.title('ST Depression VS Age', fontsize=17)
plt.xlabel('Age', fontsize=15)
plt.ylabel('ST depression', fontsize=15)
plt.show()
```

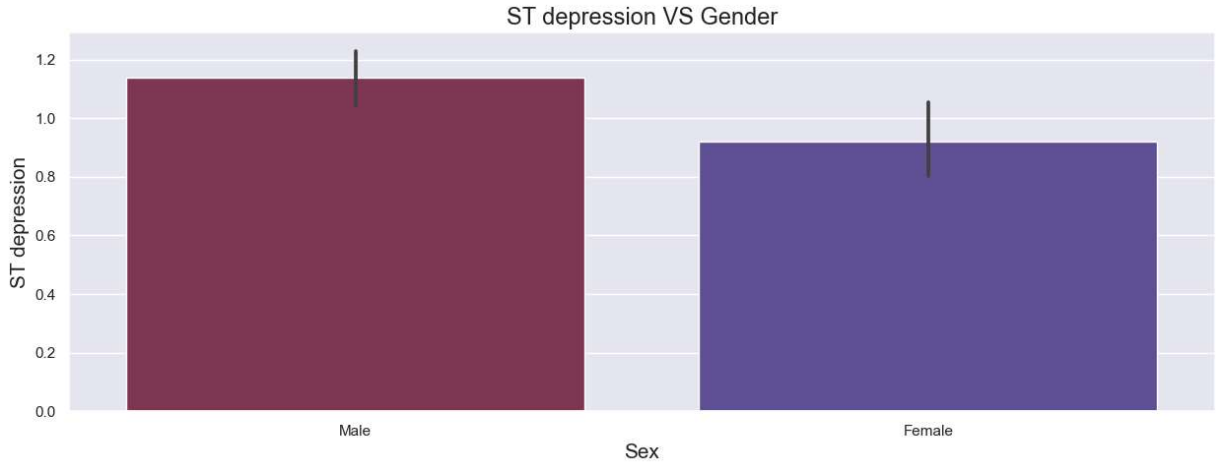
```
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\nagpa\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



ST depression mostly increases bw the age group of 30-40

GENDER

In [47]:
```
sns.barplot(x='Sex1', y='Old-Peak', data=df, palette='twilight_r')
plt.title('ST depression VS Gender', fontsize=17)
plt.xlabel('Sex', fontsize=15)
plt.ylabel('ST depression', fontsize=15)
plt.show()
```



More Males are prone to ST depression as compare to females

EXERCISE-INDUCED ANGINA

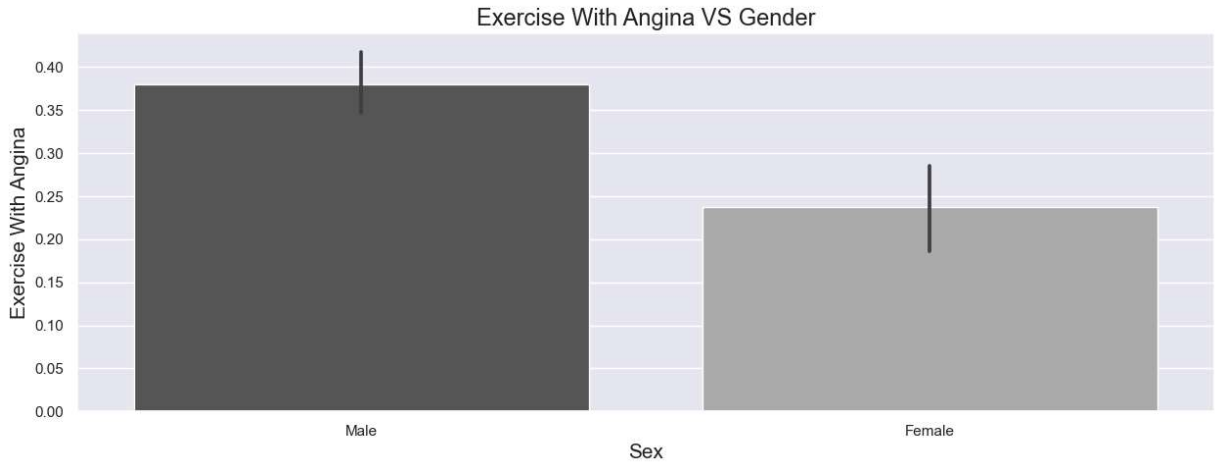
Percentage of individuals with Excercise-Induced Angina

In [48]:
```
count = 0
for i in range(len(df)):
    if df['Exercise Induced Angina'][i] == 1:
        count += 1

exercise_induced_angina_rate = ((count/len(df)) * 100)
exercise_induced_angina_rate = round(exercise_induced_angina_rate, 1)
print(f"Exercise Induced Angina Rate: {exercise_induced_angina_rate} %")
```

```
Exercise Induced Angina Rate: 33.7 %
```

GENDER

In [49]:
```
sns.barplot(x='Sex1', y='Exercise Induced Angina', data=df, palette='binary_r')
plt.title('Exercise With Angina VS Gender', fontsize=17)
plt.xlabel('Sex', fontsize=15)
plt.ylabel('Exercise With Angina', fontsize=15)
plt.show()
```



Males have have high Exercise Angina

Abnormal resting electrocardiograhic results

Percentage of individuals with abnormal resting electrocardiograhic results

In [50]:
```
count = 0
for i in range(len(df)):
    if df['Resting Electro-Cardiographic Results'][i] == 1 or df['Resting Electro-C
        count += 1

abnormal_resting_electrocardiographic_rate = ((count/len(df)) * 100)
abnormal_resting_electrocardiographic_rate = round(abnormal_resting_electrocardiogr
print(f"Abnormal Resting Electrocardiographic Rate: {abnormal_resting_electrocardio
```

```
Abnormal Resting Electrocardiographic Rate: 51.5 %
```

In [51]:
```
df.head()
```

Out[51]:

	Age	Sex	Chest Pain Type	Resting BP	Serum Cholestoral (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Resting Electro-Cardiographic Results	Max Heart Rate Achieved	Exercise Induced Angina	Old Peak
0	52	1	0	125	212	0	1	168	0	1.0
1	53	1	0	140	203	1	0	155	1	3.1
2	70	1	0	145	174	0	1	125	1	2.6
3	61	1	0	148	203	0	1	161	0	0.0
4	62	0	0	138	294	1	1	106	0	1.9

In [ ]:

In [ ]: