

In [ ]: Heart Disease Analysis

In [1]:

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

In [132...]:

```
df = pd.read_csv("heart.csv")
df
```

Out[132]:

	Age	Sex	ChestPain	Resting Blood Pressure (mm Hg)	Cholesterol (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Rest ECG	Maximum Heart Rate	Exercise Induced Angina	Previous Peak
0	63	1	Non-Typical	145	233	1	0	150	0	2.3
1	37	1	Nonanginal	130	250	0	1	187	0	3.5
2	41	0	Asymptomatic	130	204	0	0	172	0	1.4
3	56	1	Asymptomatic	120	236	0	1	178	0	0.8
4	57	0	Typical	120	354	0	1	163	1	0.6
...	...	...	...	...	...	...	...	...	...	...
298	57	0	Typical	140	241	0	1	123	1	0.2
299	45	1	Non-Typical	110	264	0	1	132	0	1.2
300	68	1	Typical	144	193	1	1	141	0	3.4
301	57	1	Typical	130	131	0	1	115	1	1.2
302	57	0	Asymptomatic	130	236	0	0	174	0	0.0

303 rows × 12 columns

In [133...]:

```
#Top 10 Data
df.head(10)
```

Out[133]:

	Age	Sex	ChestPain	Resting Blood Pressure (mm Hg)	Cholesterol (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Rest ECG	Maximum Heart Rate	Exercise Induced Angina	Previous Peak	SIC
0	63	1	Non-Typical	145	233	1	0	150	0	2.3	
1	37	1	Nonanginal	130	250	0	1	187	0	3.5	
2	41	0	Asymptomatic	130	204	0	0	172	0	1.4	
3	56	1	Asymptomatic	120	236	0	1	178	0	0.8	
4	57	0	Typical	120	354	0	1	163	1	0.6	
5	57	1	Typical	140	192	0	1	148	0	0.4	
6	56	0	Asymptomatic	140	294	0	0	153	0	1.3	
7	44	1	Asymptomatic	120	263	0	1	173	0	0.0	
8	52	1	Nonanginal	172	199	1	1	162	0	0.5	
9	57	1	Nonanginal	150	168	0	1	174	0	1.6	

In [134...]

#Bottom 10 Data  
df.tail(10)

Out[134]:

	Age	Sex	ChestPain	Resting Blood Pressure (mm Hg)	Cholesterol (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Rest ECG	Maximum Heart Rate	Exercise Induced Angina	Previous Peak
293	67	1	Nonanginal	152	212	0	0	150	0	0.8
294	44	1	Typical	120	169	0	1	144	1	2.8
295	63	1	Typical	140	187	0	0	144	1	4.0
296	63	0	Typical	124	197	0	1	136	1	0.0
297	59	1	Typical	164	176	1	0	90	0	1.0
298	57	0	Typical	140	241	0	1	123	1	0.2
299	45	1	Non-Typical	110	264	0	1	132	0	1.2
300	68	1	Typical	144	193	1	1	141	0	3.4
301	57	1	Typical	130	131	0	1	115	1	1.2
302	57	0	Asymptomatic	130	236	0	0	174	0	0.0

In [135...]

#Shape  
df.shape

Out[135]: (303, 12)

```
In [136... #No. of Rows = 303
#No. of Column = 12
```

```
In [ ]:
```

```
In [137... #Removes rows containing any missing values (NaN values)
df.dropna(inplace = True)
```

```
In [138... #Checking Missing Values
df.isna()
```

Out[138]:

	Age	Sex	ChestPain	Resting Blood Pressure (mm Hg)	Cholesterol (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Rest ECG	Maximum Heart Rate	Exercise Induced Angina	Previous Peak	SI
0	False	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	False	False	False	False	F
299	False	False	False	False	False	False	False	False	False	False	F
300	False	False	False	False	False	False	False	False	False	False	F
301	False	False	False	False	False	False	False	False	False	False	F
302	False	False	False	False	False	False	False	False	False	False	F

303 rows × 12 columns

```
In [ ]: #True : If cell has a missing value (NaN), and False : If it's not a missing value.
```

```
In [ ]:
```

```
In [139... #Count of number of missing(Nan) values in each column
df.isna().sum()
```

```
Out[139]: Age          0  
          Sex          0  
          ChestPain    0  
          Resting Blood Pressure (mm Hg) 0  
          Cholestrol (mg/dl)      0  
          Fasting Blood Sugar > 120 mg/dl 0  
          Rest ECG        0  
          Maximum Heart Rate   0  
          Exercise Induced Angina 0  
          Previous Peak       0  
          Slope           0  
          Thal            0  
          dtype: int64
```

In [ ]:

```
In [140... #Array containing all the unique values present in the 'Age' column  
  
Unique_Age = df['Age'].unique()  
  
print("Unique Age : ", Unique_Age)
```

```
Unique Age : [63 37 41 56 57 44 52 54 48 49 64 58 50 66 43 69 59 42 61 40 71 51 6  
5 53  
46 45 39 47 62 34 35 29 55 60 67 68 74 76 70 38 77]
```

In [ ]:

```
In [141... #Number of unique values in the 'Age' column:  
  
Count_Age = df['Age'].nunique()  
  
print("Count of Unique Age : ", Count_Age)
```

```
Count of Unique Age : 41
```

In [ ]:

```
In [142... #Statistical summary of the 'Age' column:  
  
df['Age'].describe()
```

```
Out[142]: count    303.000000  
          mean     54.366337  
          std      9.082101  
          min     29.000000  
          25%     47.500000  
          50%     55.000000  
          75%     61.000000  
          max     77.000000  
          Name: Age, dtype: float64
```

In [ ]:

```
In [143...]: #Bins for various age categories:  
young = df['Age'].quantile(0.25) #find the 25th percentile (Q1) of the 'Age' column  
  
mid_age = df['Age'].median() #find the median value in the 'Age' column  
  
senior = df['Age'].quantile(0.75) #find the 75th percentile (Q3) value in the 'Age'  
elder = df['Age'].max() #find the maximum (highest) value in the 'Age' column  
  
In [145...]: # Define the labels for each bin:  
bin_labels = ["Young_Adult", "Middle_Aged", "Senior_Citizen", "Elderly"]  
  
bins = [0, young, mid_age, senior, elder]  
  
In [146...]: # Create a new column 'Age_Category' based on the bins and labels  
  
df['Age_Category'] = pd.cut(df['Age'], labels = bin_labels, bins = bins)  
  
In [147...]: # Display the DataFrame with the new 'Age_Category' column  
  
print(df)
```

	Age	Sex	ChestPain	Resting Blood Pressure (mm Hg)	\	
0	63	1	Non-Typical	145		
1	37	1	Nonanginal	130		
2	41	0	Asymptomatic	130		
3	56	1	Asymptomatic	120		
4	57	0	Typical	120		
..	...	...	...	...		
298	57	0	Typical	140		
299	45	1	Non-Typical	110		
300	68	1	Typical	144		
301	57	1	Typical	130		
302	57	0	Asymptomatic	130		
	Cholestrol (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Rest ECG	\		
0	233	1	0			
1	250	0	1			
2	204	0	0			
3	236	0	1			
4	354	0	1			
..	...	...	...	...		
298	241	0	1			
299	264	0	1			
300	193	1	1			
301	131	0	1			
302	236	0	0			
	Maximum Heart Rate	Exercise Induced Angina	Previous Peak	Slope	Thal	\
0	150	0	2.3	0	1	
1	187	0	3.5	0	2	
2	172	0	1.4	2	2	
3	178	0	0.8	2	2	
4	163	1	0.6	2	2	
..	...	...	...	...	...	
298	123	1	0.2	1	3	
299	132	0	1.2	1	3	
300	141	0	3.4	1	3	
301	115	1	1.2	1	3	
302	174	0	0.0	1	2	
	Age_Category					
0	Elderly					
1	Young_Adult					
2	Young_Adult					
3	Senior_Citizen					
4	Senior_Citizen					
..	...					
298	Senior_Citizen					
299	Young_Adult					
300	Elderly					
301	Senior_Citizen					
302	Senior_Citizen					

[303 rows x 13 columns]

In [148...]: # Display the table with 'Age' and 'Age\_Category' column

```
df[['Age', 'Age_Category']]
```

Out[148]:

	Age	Age_Category
<b>0</b>	63	Elderly
<b>1</b>	37	Young_Adult
<b>2</b>	41	Young_Adult
<b>3</b>	56	Senior_Citizen
<b>4</b>	57	Senior_Citizen
...	...	...
<b>298</b>	57	Senior_Citizen
<b>299</b>	45	Young_Adult
<b>300</b>	68	Elderly
<b>301</b>	57	Senior_Citizen
<b>302</b>	57	Senior_Citizen

303 rows × 2 columns

In [149...]:

```
# Display the table with 'Age' and 'Age_Category' Top 10 values
```

```
df[['Age', 'Age_Category']].head(10)
```

Out[149]:

	Age	Age_Category
<b>0</b>	63	Elderly
<b>1</b>	37	Young_Adult
<b>2</b>	41	Young_Adult
<b>3</b>	56	Senior_Citizen
<b>4</b>	57	Senior_Citizen
<b>5</b>	57	Senior_Citizen
<b>6</b>	56	Senior_Citizen
<b>7</b>	44	Young_Adult
<b>8</b>	52	Middle_Aged
<b>9</b>	57	Senior_Citizen

In [150...]:

```
# Display the table with 'Age' and 'Age_Category' Bottom 10 values
```

```
df[['Age', 'Age_Category']].tail(10)
```

Out[150]:

	Age	Age_Category
293	67	Elderly
294	44	Young_Adult
295	63	Elderly
296	63	Elderly
297	59	Senior_Citizen
298	57	Senior_Citizen
299	45	Young_Adult
300	68	Elderly
301	57	Senior_Citizen
302	57	Senior_Citizen

In [151...]

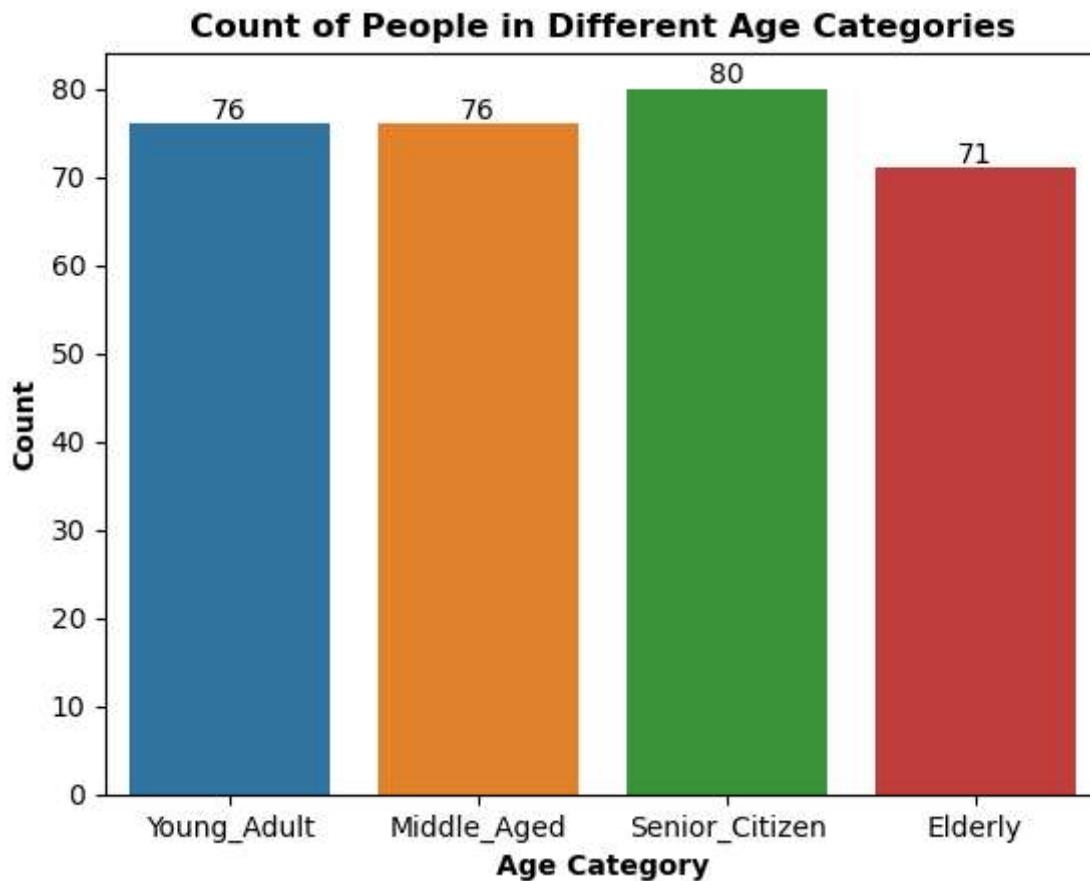
```
#Creating a visualization on Age_Category Column

fig = sns.countplot(x = "Age_Category" , data = df)

# Add count Labels on top of each bar
for bars in fig.containers:
    fig.bar_label(bars)

# Set plot Labels and title
plt.xlabel('Age Category', fontweight = 'bold')
plt.ylabel('Count', fontweight = 'bold')
plt.title('Count of People in Different Age Categories', fontweight = 'bold')

# Show the plot
plt.show()
```



In [ ]:

```
#Adding a new column 'Sex_Category' based on the values in the 'Sex' column
df['Sex_Category'] = np.where(df['Sex']==1, 'Male', 'Female')

#It assigns 'Male' to the 'Sex_Category' column if the corresponding value in the 'Sex' column is 1, otherwise 'Female'. This converts the categorical values into binary values.

#Adding a new column 'ExAng_Category' based on the values in the 'Exercise Induced Angina' column
df['ExAng_Category'] = np.where(df['Exercise Induced Angina'] == 1, 'Yes', 'No')

#It assigns 'Yes' to the 'ExAng_Category' column if the corresponding value in the 'Exercise Induced Angina' column is 1, otherwise 'No'.
```

df.head()

Out[154]:

	Age	Sex	ChestPain	Resting Blood Pressure (mm Hg)	Cholesterol (mg/dl)	Fasting Blood Sugar > 120 mg/dl	Rest ECG	Maximum Heart Rate	Exercise Induced Angina	Previous Peak	SIC
0	63	1	Non-Typical	145	233	1	0	150	0	2.3	
1	37	1	Nonanginal	130	250	0	1	187	0	3.5	
2	41	0	Asymptomatic	130	204	0	0	172	0	1.4	
3	56	1	Asymptomatic	120	236	0	1	178	0	0.8	
4	57	0	Typical	120	354	0	1	163	1	0.6	



In [ ]:

In [155...]: #Creating a visualization on ChestPain Column at different Age\_Categories

```
# Set the figure size using plt.figure
plt.figure(figsize=(10, 6))

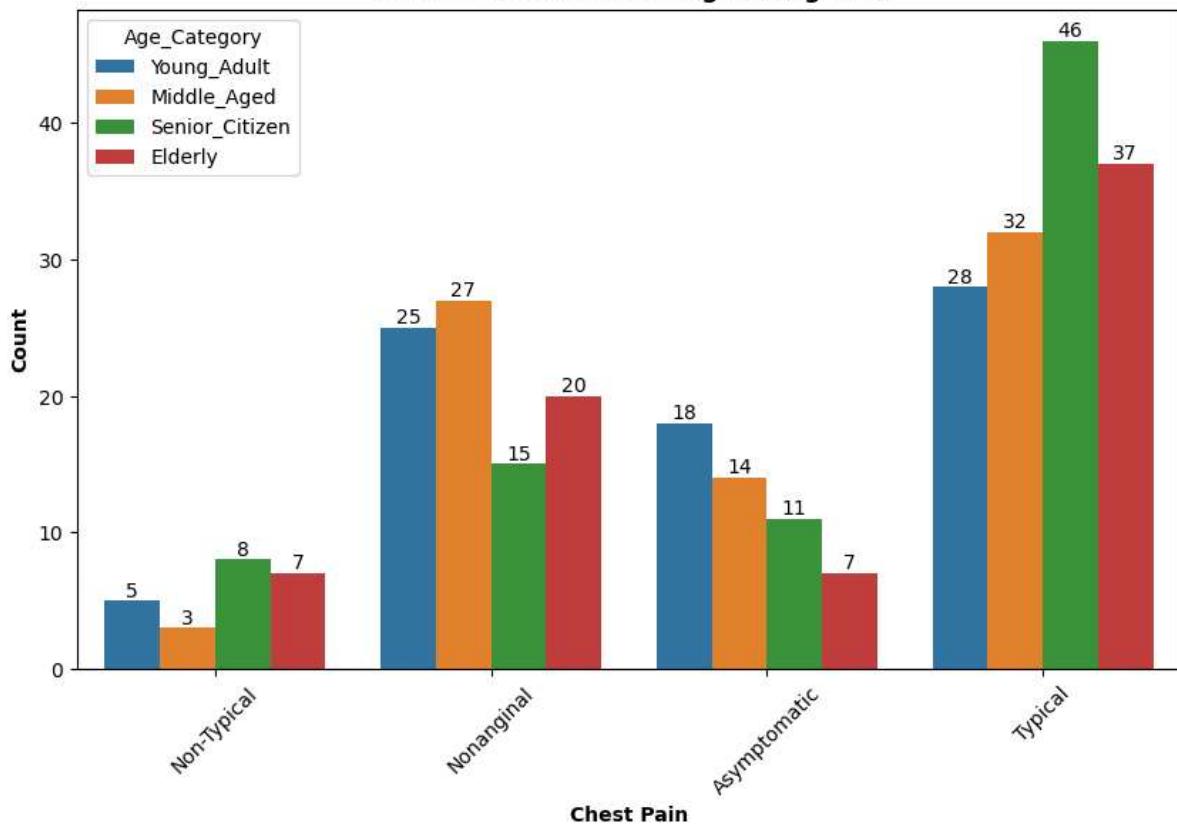
fig = sns.countplot(x = 'ChestPain', hue = 'Age_Category' , data = df)

# Add count labels on top of each bar
for bars in fig.containers:
    fig.bar_label(bars)

# Set plot labels and title
plt.xlabel('Chest Pain', fontweight = 'bold')
plt.xticks(rotation = 45)
plt.ylabel('Count', fontweight = 'bold')
plt.title('Chest Pain in Different Age Categories', fontweight = 'bold')

# Show the plot
plt.show()
```

### Chest Pain in Different Age Categories



```
In [ ]: #Typical - Characteristic chest pain associated with heart-related issues like a heart attack or angina.
```

```
#Asymptomatic - Absence of chest pain, even in the presence of heart-related conditions.
```

```
#Nonanginal - Chest pain that is not related to the heart and does not fit the typical pattern of angina.
```

```
#Non-Typical - Chest pain with atypical characteristics, not fitting the classic heart attack pattern.
```

```
In [ ]:
```

```
In [ ]: #Resting BP - BP when person at rest and not engaged in any physical activity or stress.
```

```
In [156...]: #Creating a visualization on ChestPain Column with respect to Resting Blood Pressure.
```

```
# Set the figure size using plt.figure
plt.figure(figsize=(10, 6))
```

```
fig = sns.boxplot(x = 'ChestPain', y = 'Resting Blood Pressure (mm Hg)' , data = df)
```

```
# Calculate the count for each category
```

```
count_data = df.groupby(['Resting Blood Pressure (mm Hg)', 'ChestPain']).size().reset_index()
```

```
# Set plot labels and title
```

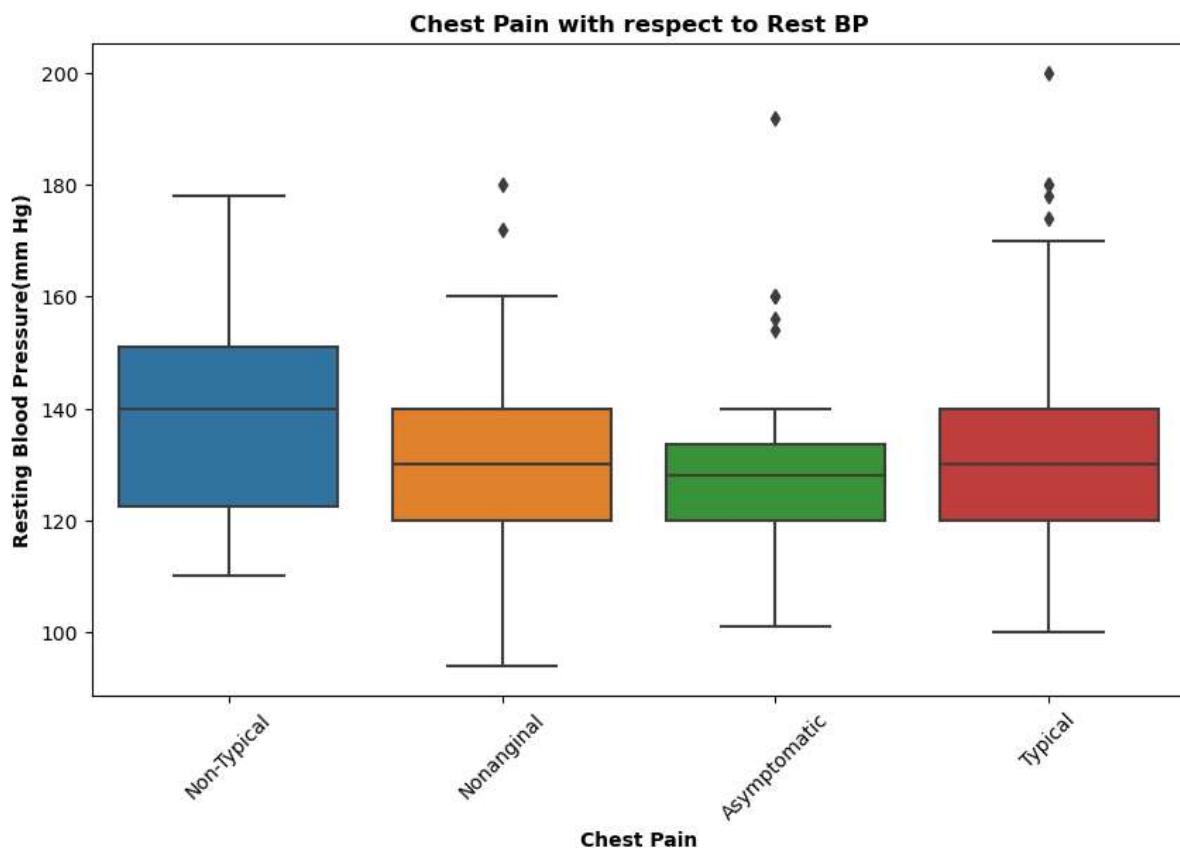
```
plt.xlabel('Chest Pain', fontweight = 'bold')
```

```
plt.xticks(rotation = 45)
```

```
plt.ylabel('Resting Blood Pressure(mm Hg)', fontweight = 'bold')
```

```
plt.title('Chest Pain with respect to Rest BP', fontweight = 'bold')

# Show the plot
plt.show()
```



In [ ]:

```
#Creating a visualization on ChestPain Column with respect to Sex_Category

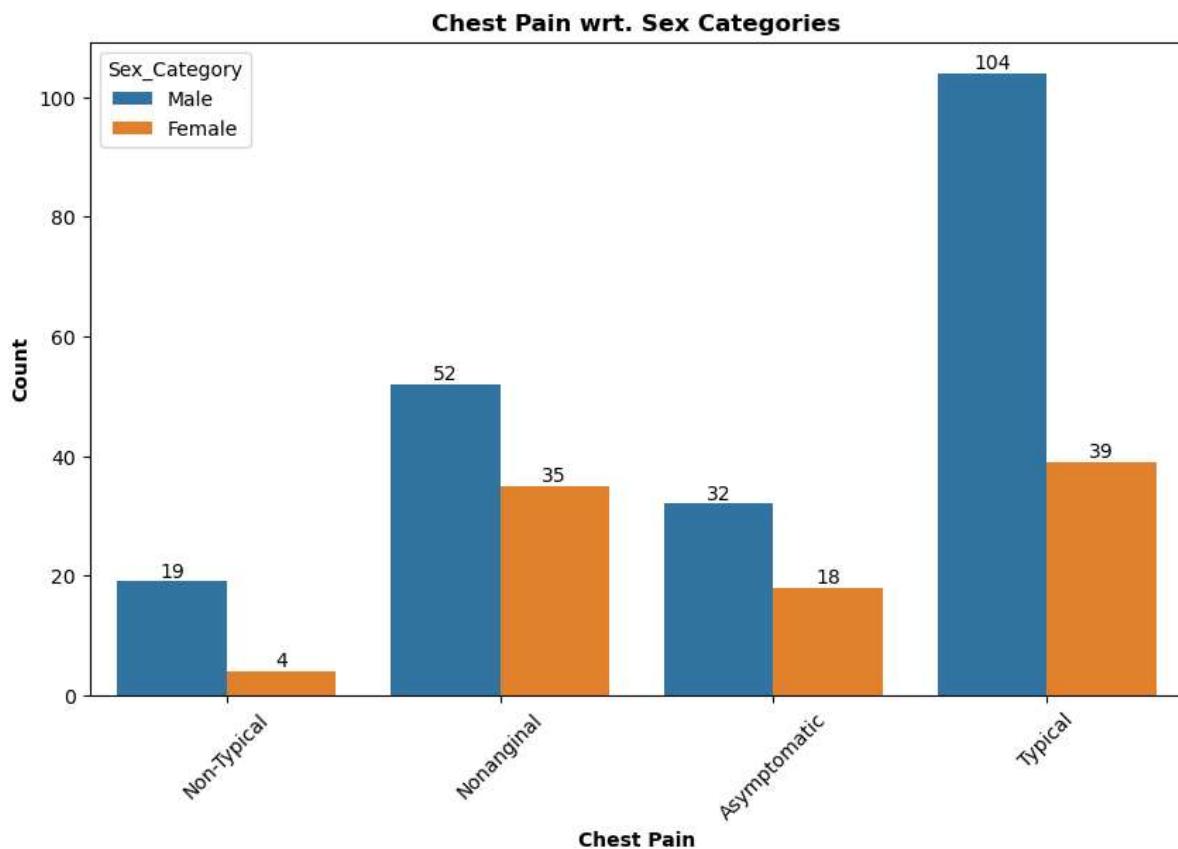
plt.figure(figsize = (10,6))

fig = sns.countplot(x = 'ChestPain', hue = 'Sex_Category', data=df)

# Add count labels on top of each bar
for bars in fig.containers:
    fig.bar_label(bars)

# Set plot labels and title
plt.xlabel('Chest Pain', fontweight = 'bold')
plt.xticks(rotation = 45)
plt.ylabel('Count', fontweight = 'bold')
plt.title('Chest Pain wrt. Sex Categories', fontweight = 'bold')

# Show the plot
plt.show()
```



In [ ]:

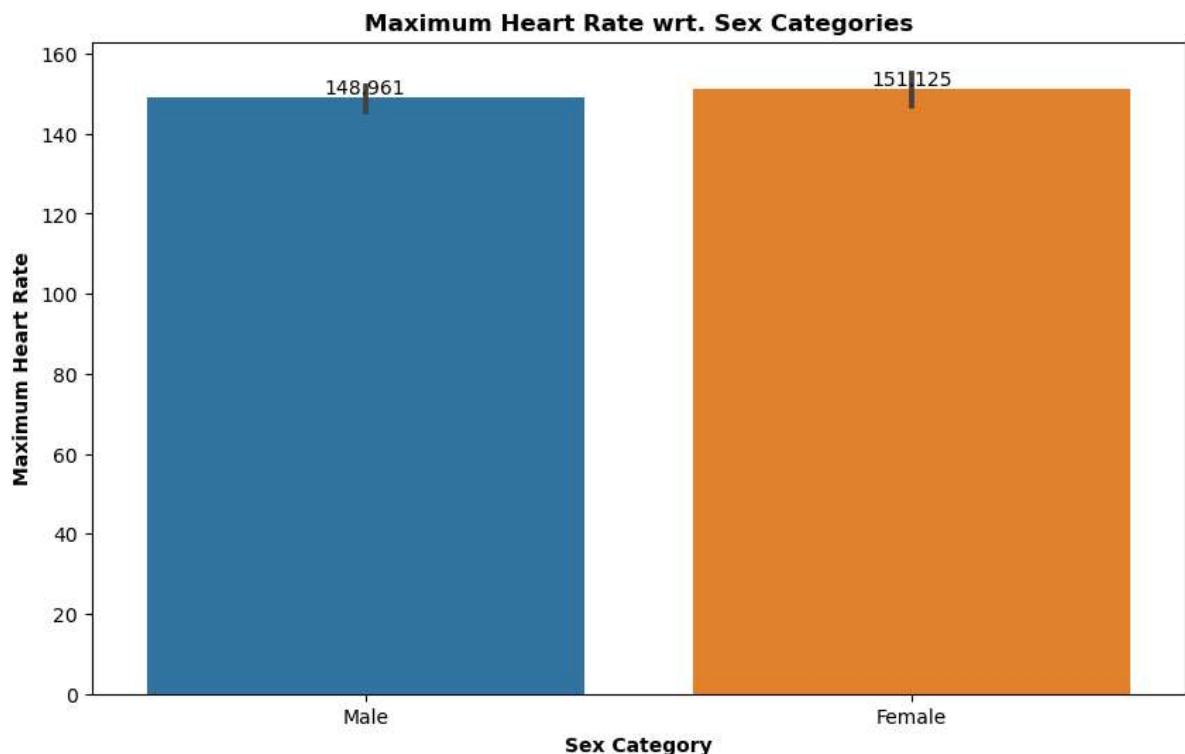
```
#Creating a visualization on Maximum Heart Rate wrt Sex Category
plt.figure(figsize = (10,6))

fig = sns.barplot(y = 'Maximum Heart Rate', x = 'Sex_Category' , data = df)

# Add count Labels on top of each bar
for bars in fig.containers:
    fig.bar_label(bars)

# Set plot Labels and title
plt.xlabel('Sex Category', fontweight = 'bold')
plt.ylabel('Maximum Heart Rate', fontweight = 'bold')
plt.title('Maximum Heart Rate wrt. Sex Categories', fontweight = 'bold')

# Show the plot
plt.show()
```



In [ ]:

```
#Summary statistics for non-numeric columns
df.describe(exclude = [np.number])
```

Out[159]:

	ChestPain	Age_Category	Sex_Category	ExAng_Category
<b>count</b>	303	303	303	303
<b>unique</b>	4	4	2	2
<b>top</b>	Typical	Senior_Citizen	Male	No
<b>freq</b>	143	80	207	204

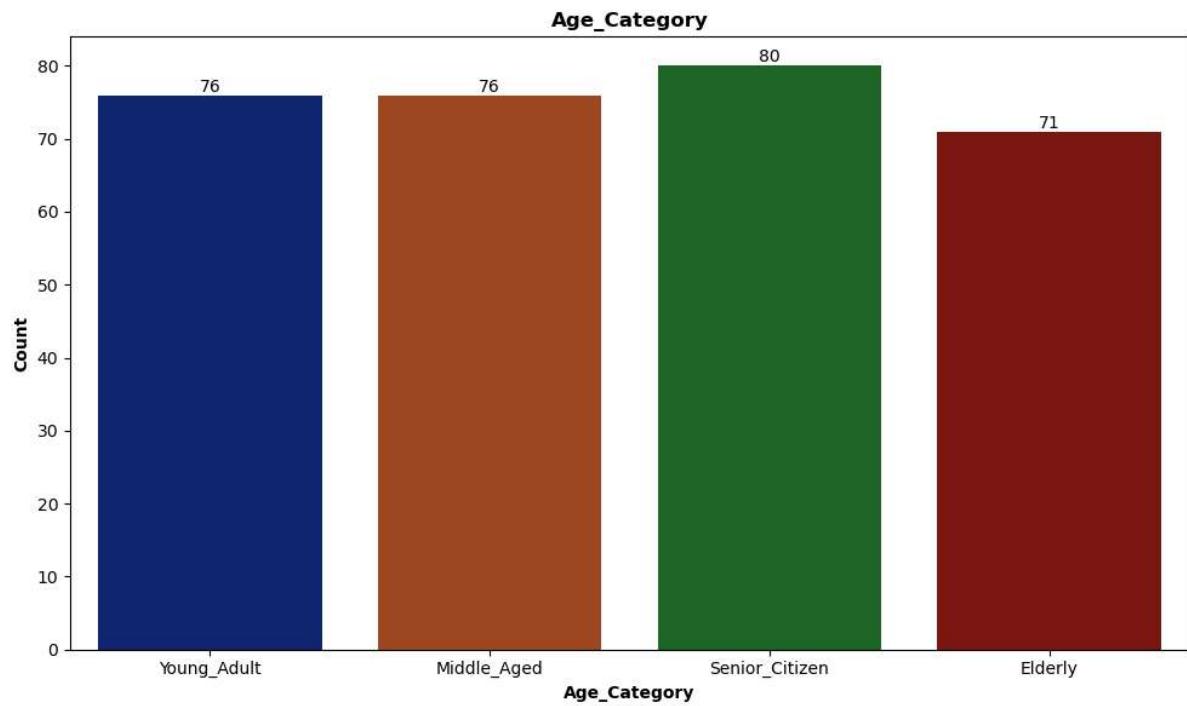
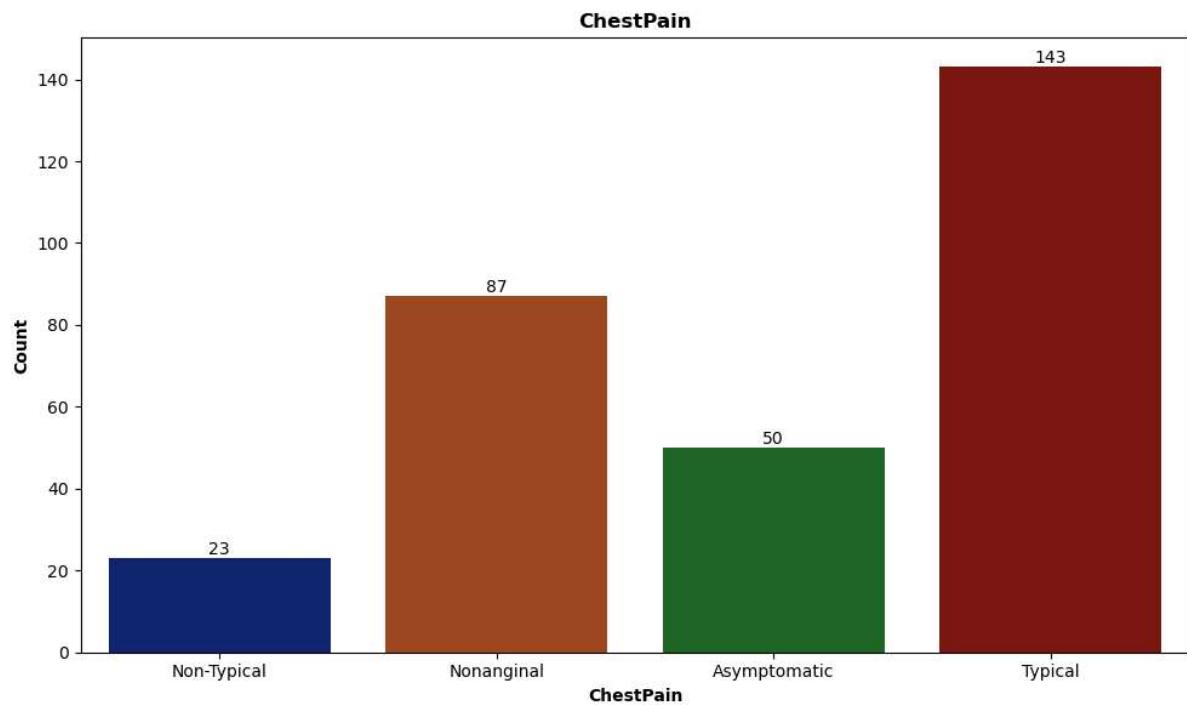
In [160...]

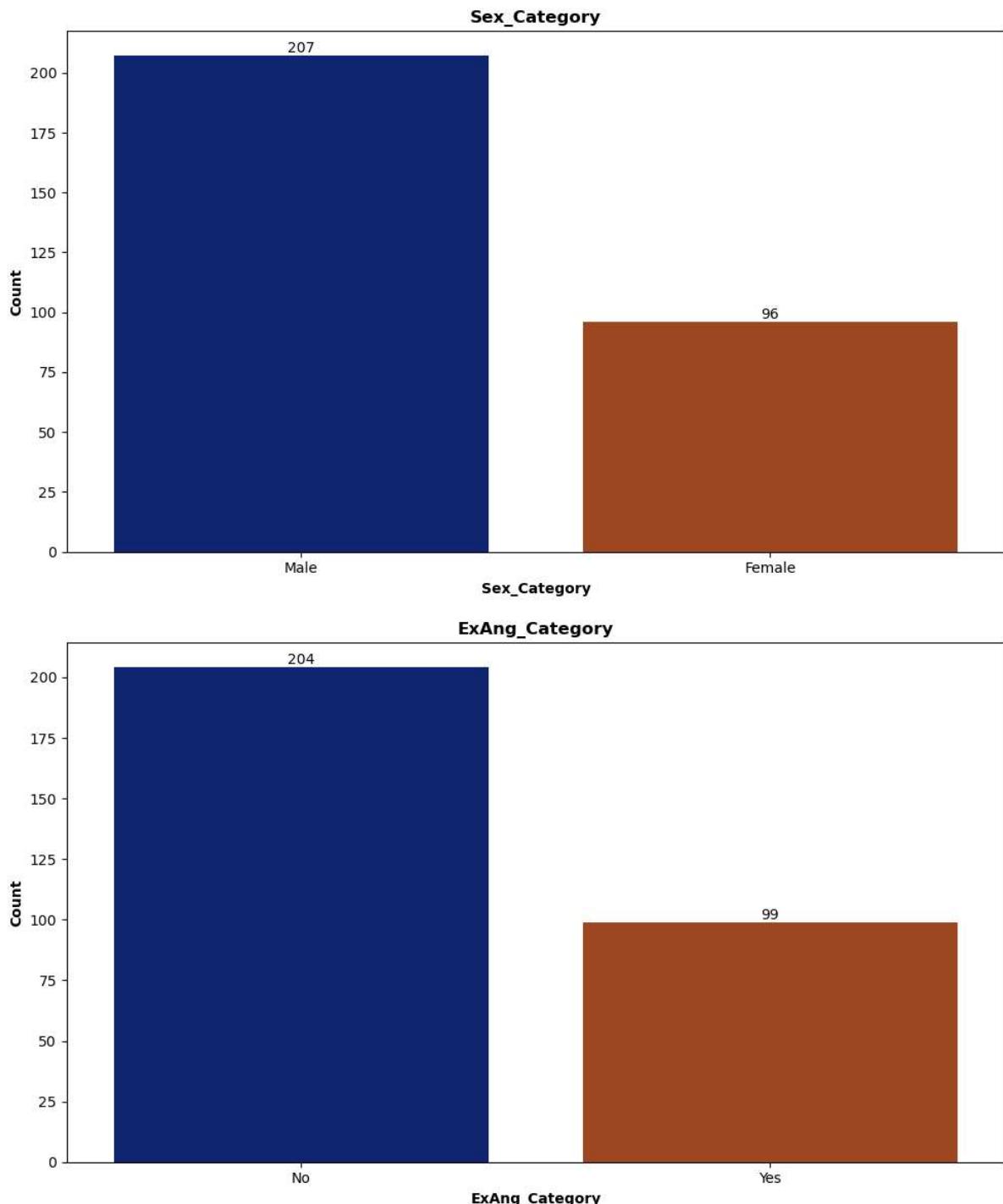
```
#Visualizations of above data
columns = ['ChestPain', 'Age_Category', 'Sex_Category', 'ExAng_Category']

def plot_cat_count():
    for c in columns :
        plt.figure(figsize=(10, 6))
        fig = sns.countplot(x = c, data = df, palette='dark')
        plt.title(c, fontweight = 'bold')
        plt.ylabel('Count', fontweight = 'bold')
        plt.xlabel(c, fontweight = 'bold')
        plt.tight_layout()

        # Add count Labels on top of each bar
        for bars in fig.containers:
            fig.bar_label(bars)
```

```
plt.show()  
plot_cat_count()
```





In [ ]:

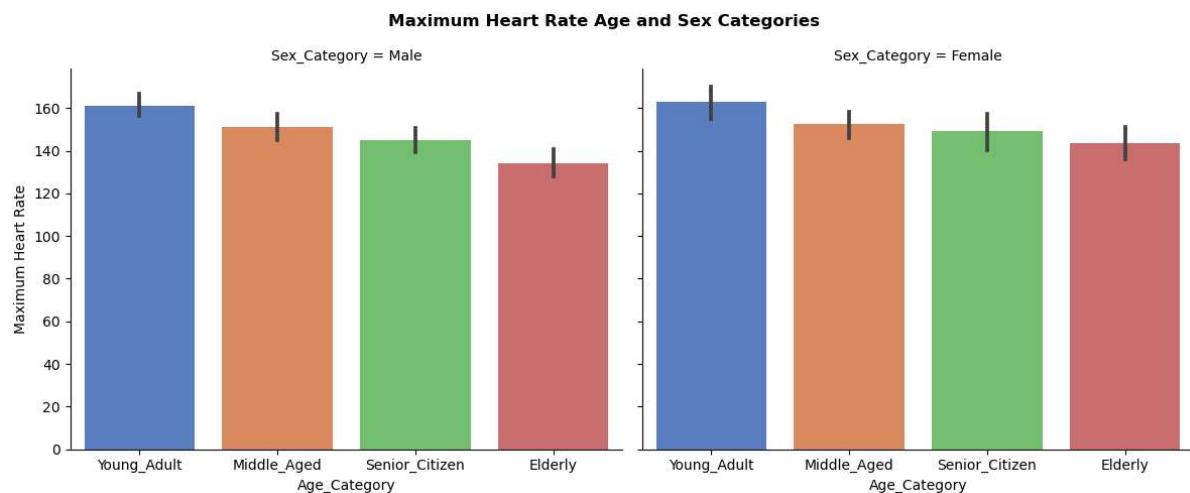
```
#Creating visualization of Maximum Heart Rate wrt Age Category for different Sex Categories
plt.figure(figsize = (10,10))
fig = sns.catplot(x='Age_Category', y='Maximum Heart Rate', data=df, col='Sex_Cat
```

```
plt.suptitle('Maximum Heart Rate Age and Sex Categories', fontweight='bold')
```

```
plt.tight_layout()
```

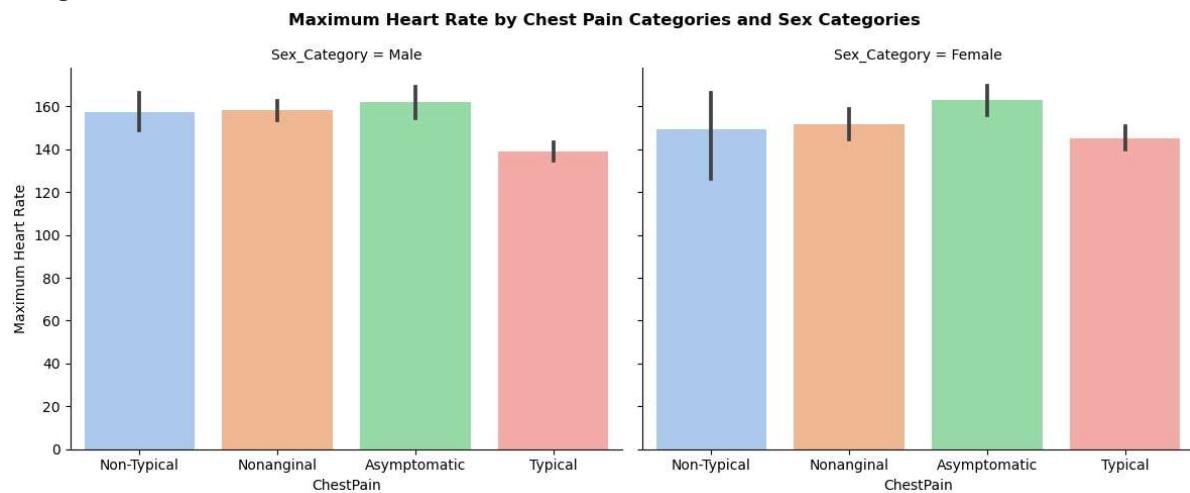
```
plt.show()
```

<Figure size 1000x1000 with 0 Axes>

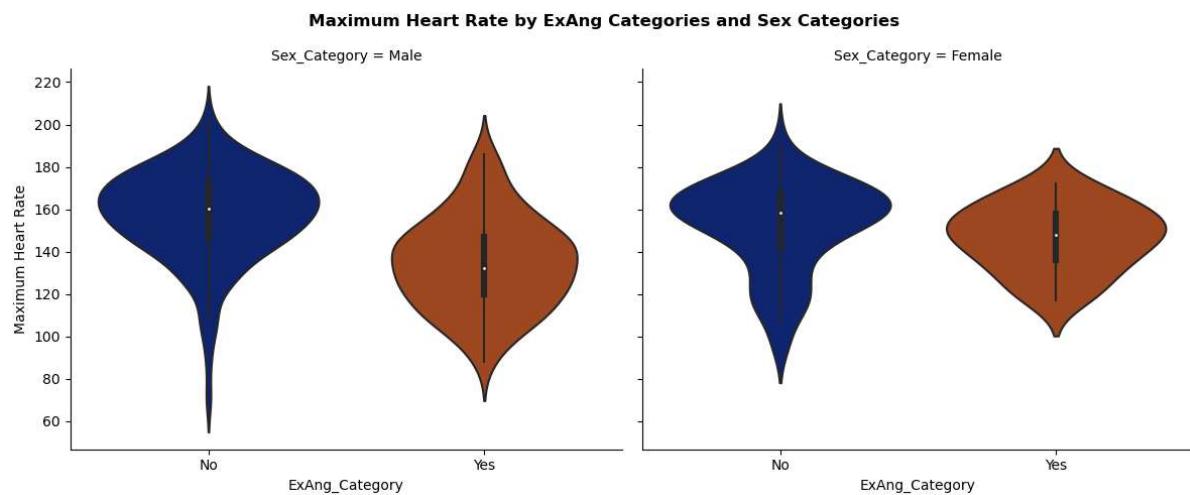


```
In [200]: #Creating visualization of Maximum Heart Rate wrt Chest pain for different Sex Cate
plt.figure(figsize = (10,10))
sns.catplot(x='ChestPain', y='Maximum Heart Rate', data=df, kind='bar', col='Sex_Cat'
plt.suptitle('Maximum Heart Rate by Chest Pain Categories and Sex Categories', font
plt.tight_layout()
plt.show()
```

<Figure size 1000x1000 with 0 Axes>



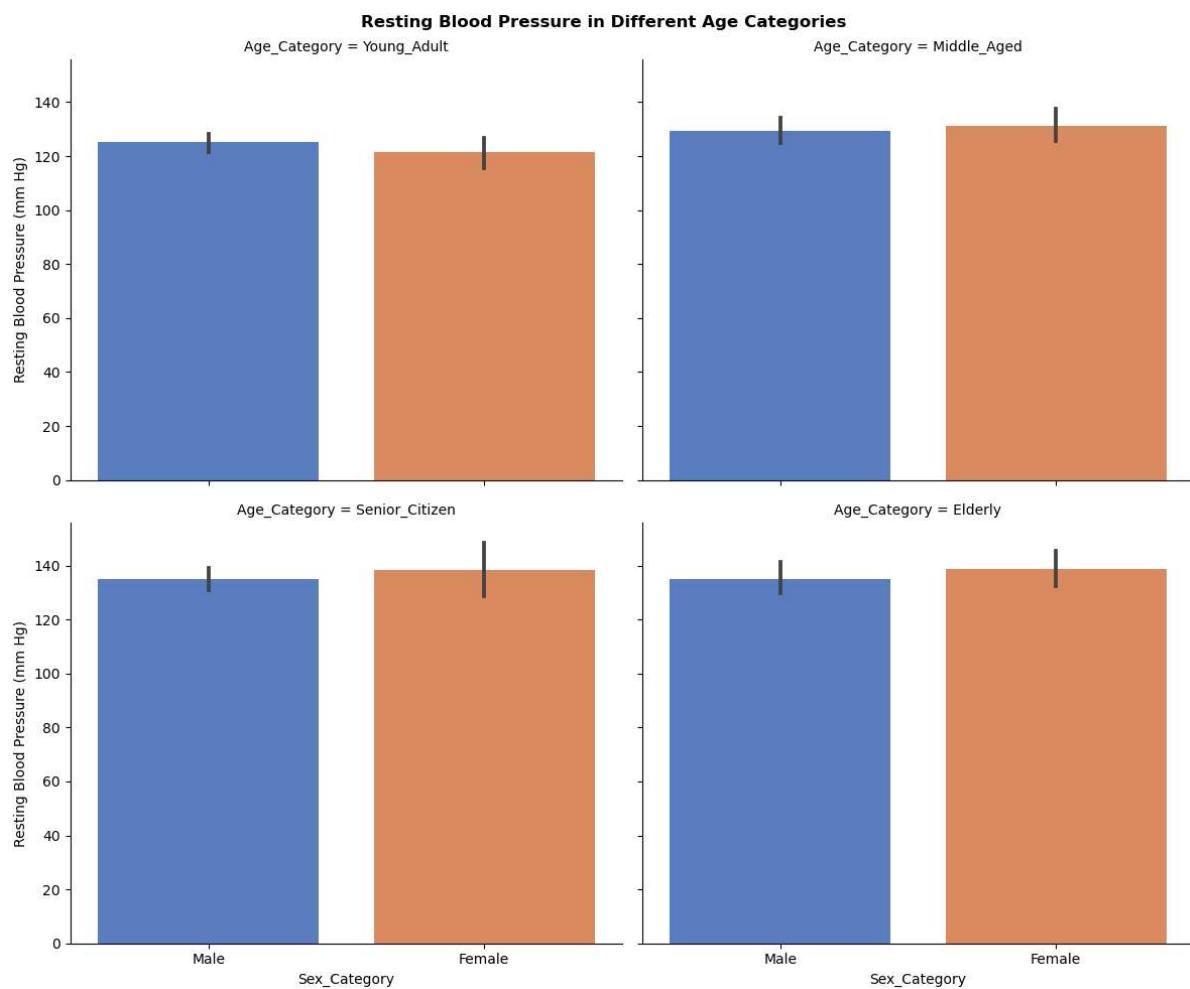
```
In [201]: #Creating visualization of Maximum Heart Rate wrt ExAng for different Sex Category
sns.catplot(x='ExAng_Category', y='Maximum Heart Rate', data=df, kind='violin', col
plt.suptitle('Maximum Heart Rate by ExAng Categories and Sex Categories', fontweigh
plt.tight_layout()
plt.show()
```



In [206...]

```
#Creating visualization of Rest BP wrt Sex Category for different Age Category

fig = sns.catplot(x='Sex_Category', y='Resting Blood Pressure (mm Hg)', data=df, co
plt.suptitle('Resting Blood Pressure in Different Age Categories', fontweight='bold'
plt.tight_layout()
plt.show()
```



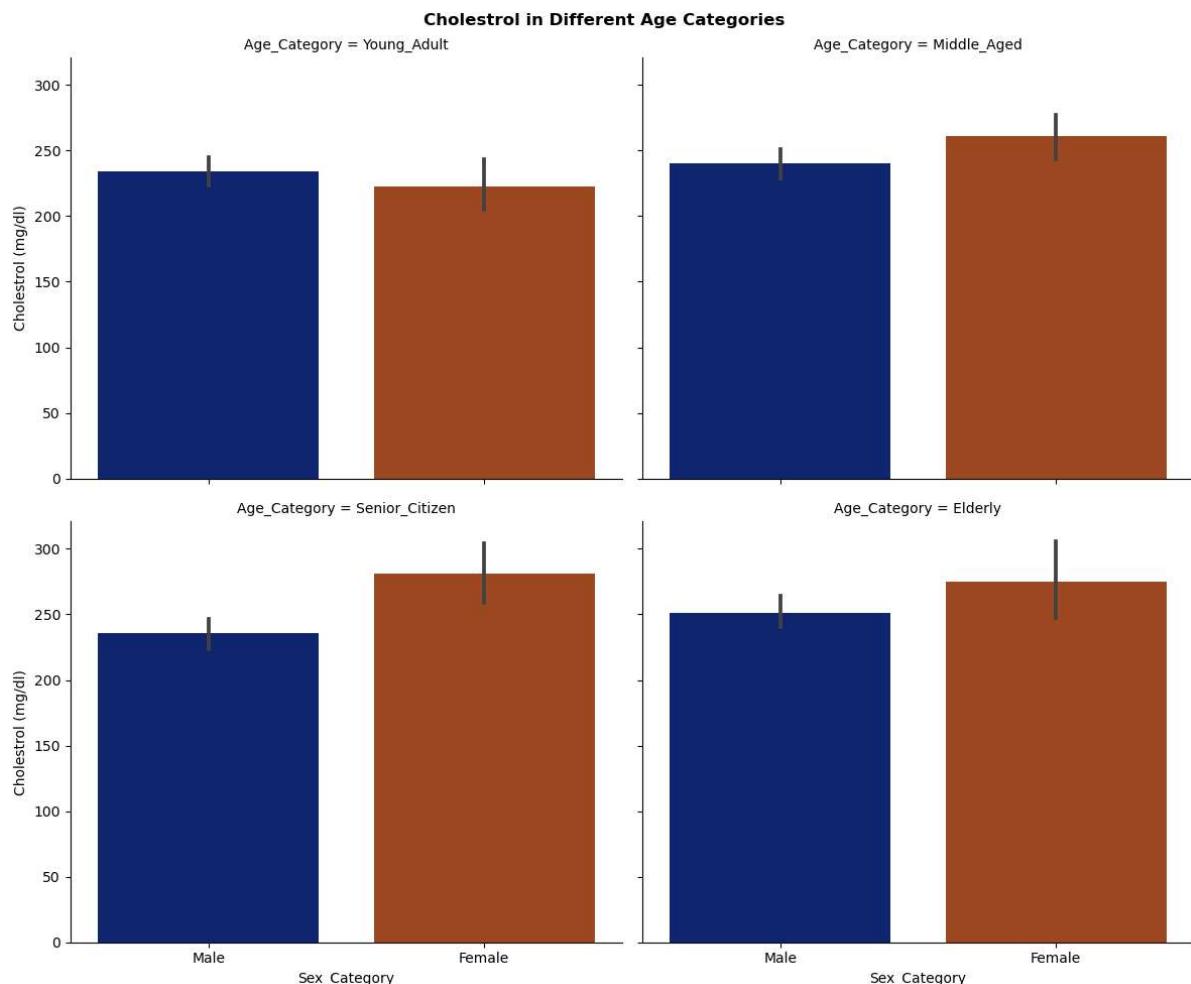
In [ ]:

```
#Creating visualization of Cholestrol wrt Sex Category for different Age Category

fig = sns.catplot(x='Sex_Category', y='Cholestrol (mg/dl)', data=df, col='Age_Categ
plt.suptitle('Cholestrol in Different Age Categories', fontweight='bold')

plt.tight_layout()

plt.show()
```



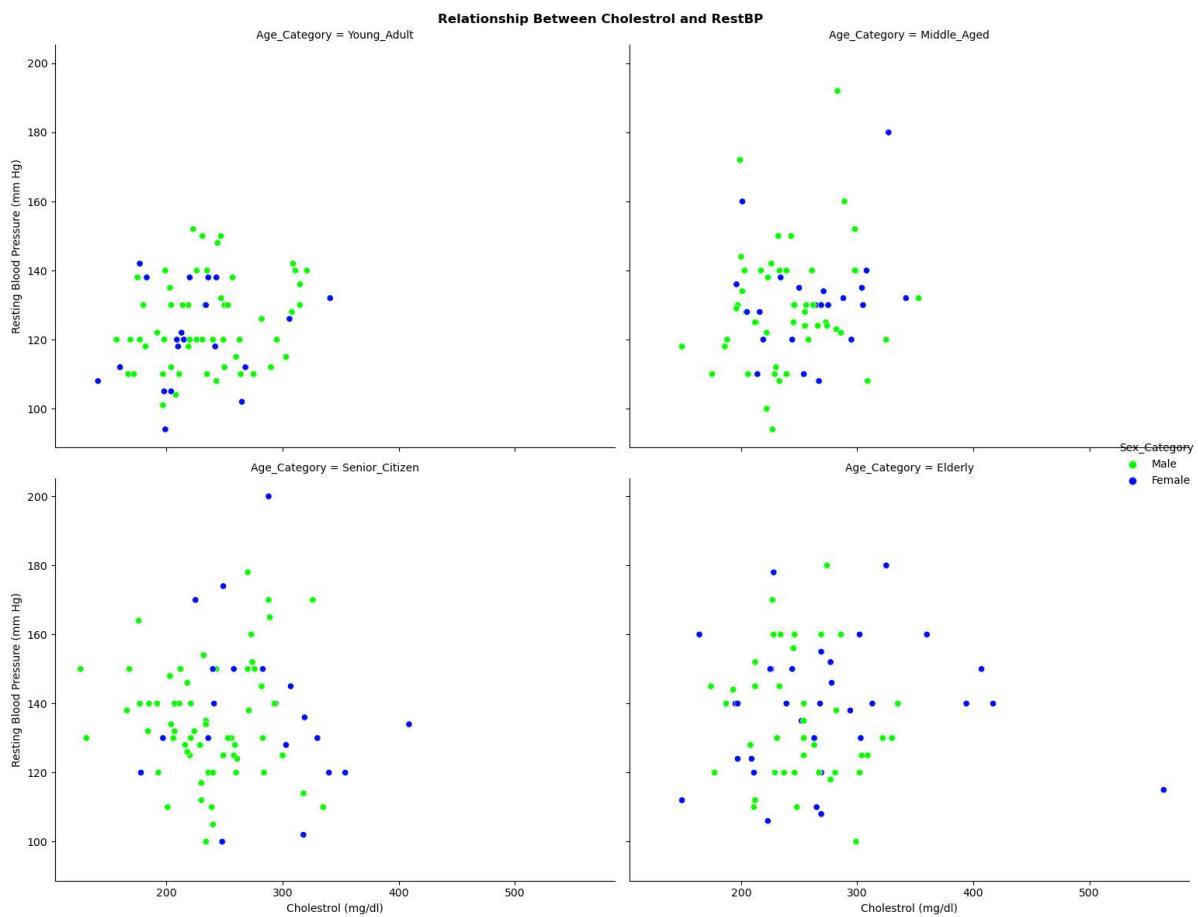
In [ ]:

```
#Creating Visualization for Relationship Between Cholestrol and RestBP

sns.relplot(x='Cholestrol (mg/dl)', y='Resting Blood Pressure (mm Hg)', data=df, ki
plt.suptitle('Relationship Between Cholestrol and RestBP', fontweight='bold')

plt.tight_layout()

plt.show()
```



In [ ]:

In [ ]: Conclusion **from** above visualizations:

1. Predominantly, both male **and** female individuals reported experiencing typical **ch** **while** a smaller proportion described non-typical symptoms.
2. On average, females exhibit a slightly higher heart rate than males.
3. The blood pressure of females shows a higher increase **with** each passing year com
4. Over the years, female cholesterol levels show a higher increase compared to mal