Heart Attack Dataset

Data Manipulation Part

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

In [2]:
```
#importing the data
df = pd.read_csv('D:/Data Sets/Python/Heart.csv')
df
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1883	60	1	0	140	207	0	0	138	1	1.9	2	1	3
1884	46	1	0	140	311	0	1	120	1	1.8	1	2	3
1885	59	1	3	134	204	0	1	162	0	0.8	2	2	2
1886	54	1	1	154	232	0	0	164	0	0.0	2	1	2
1887	53	1	0	110	335	0	1	143	1	3.0	1	1	3

1888 rows × 14 columns

In [3]:
```
#getting top 5 records
df.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

In [4]:
```
#getting last 5 records
df.tail()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	thal
1883	60	1	0	140	207	0	0	138	1	1.9	2	1	3
1884	46	1	0	140	311	0	1	120	1	1.8	1	2	3
1885	59	1	3	134	204	0	1	162	0	0.8	2	2	2
1886	54	1	1	154	232	0	0	164	0	0.0	2	1	2
1887	53	1	0	110	335	0	1	143	1	3.0	1	1	3

In [5]:
```
#getting the shape of data
df.shape
```

Out[5]: (1888, 14)

In [6]:
```
#getting the size of data
df.size
```

Out[6]: 26432

In [7]:
```
#information abut the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1888 entries, 0 to 1887
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1888 non-null   int64  
 1   sex       1888 non-null   int64  
 2   cp        1888 non-null   int64  
 3   trestbps  1888 non-null   int64  
 4   chol      1888 non-null   int64  
 5   fbs       1888 non-null   int64  
 6   restecg   1888 non-null   int64  
 7   thalachh  1888 non-null   int64  
 8   exang     1888 non-null   int64  
 9   oldpeak   1888 non-null   float64
 10  slope     1888 non-null   int64  
 11  ca        1888 non-null   int64  
 12  thal      1888 non-null   int64  
 13  target    1888 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 206.6 KB
```

In [8]:
```
#Checking the data types of each column
df.dtypes
```

Out[8]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalachh      int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [9]:
```
#checking the null values
df.isnull()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
1883	False	False	False	False	False	False	False	False	False	False	False
1884	False	False	False	False	False	False	False	False	False	False	False
1885	False	False	False	False	False	False	False	False	False	False	False
1886	False	False	False	False	False	False	False	False	False	False	False
1887	False	False	False	False	False	False	False	False	False	False	False

1888 rows × 14 columns

In [10]:
```
#counting the null values in every column
df.isna().sum()
```

Out[10]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalachh    0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [11]:
```
#getting the statistical discription of the data
df.describe()
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	r
count	1888.000000	1888.000000	1888.000000	1888.000000	1888.000000	1888.000000	1888.(
mean	54.354343	0.688559	1.279131	131.549258	246.855403	0.148305	0.:
std	9.081505	0.463205	1.280877	17.556985	51.609329	0.355496	0.(
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.(
25%	47.750000	0.000000	0.000000	120.000000	211.000000	0.000000	0.(
50%	55.000000	1.000000	1.000000	130.000000	241.000000	0.000000	1.(
75%	61.000000	1.000000	2.000000	140.000000	276.000000	0.000000	1.(
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.(

In [12]:
```
#Checking the dupicate rows in the data set
df.duplicated().any()
```

Out[12]:
```
True
```

In [13]:
```
#Removing the duplicate rows
df = df.drop_duplicates()
df.head()
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalachh	exang	oldpeak	slope	ca	thal	ta
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

Get minimum and maximum age along with mean, meadian, mode and Standard Deviation

In [14]:
```
#Calculate Mean
mean = df['age'].mean()
#Calculate Median
median = df['age'].median()
#Calculate Mode
mode = df['age'].mode().iloc[0]
#Calculate standard deviation
std = df['age'].std()
#Calculate Minimum values
minimum = df['age'].min()
#Calculate Maximum values
maximum = df.age.max()
print(f" Mean of Age : {mean}")
print(f" Median of Age : {median}")
print(f" Mode of Age : {mode}")
print(f" Standard deviation of Age : {std:.2f}")
print(f" Maximum of Age : {maximum}")
print(f" Minimum of Age : {minimum}")
```

```
Mean of Age : 54.4734219269103
Median of Age : 55.0
Mode of Age : 58
Standard deviation of Age : 9.04
Maximum of Age : 77
Minimum of Age : 29
```

Check how many males and females are in your dataset

In [15]:
```
print(df['sex'].value_counts())
```

```
sex
1    412
0    190
Name: count, dtype: int64
```

Data Visualization Part

Exploring Relationships: Heatmaps with Python for Data Visualization

In [16]:
```
#Get co-relationship of your data
df.corr()
```

Out[16]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
age	1.000000	-0.102907	0.000853	0.276338	0.214343	0.111373	0.027396	-0.38085
sex	-0.102907	1.000000	-0.019874	-0.060229	-0.202161	0.037127	-0.002663	-0.07078
cp	0.000853	-0.019874	1.000000	0.005413	0.009322	0.017099	0.259196	0.01705
trestbps	0.276338	-0.060229	0.005413	1.000000	0.137097	0.164863	0.016873	-0.04449
chol	0.214343	-0.202161	0.009322	0.137097	1.000000	0.017109	0.049201	-0.00267
fbs	0.111373	0.037127	0.017099	0.164863	0.017109	1.000000	-0.009832	0.01091
restecg	0.027396	-0.002663	0.259196	0.016873	0.049201	-0.009832	1.000000	-0.01544
thalachh	-0.380859	-0.070786	0.017050	-0.044497	-0.002679	0.010911	-0.015440	1.00000
exang	0.091769	0.144172	-0.052621	0.064775	0.068189	0.012889	0.021343	-0.38584
oldpeak	0.193927	0.112975	-0.013760	0.205781	0.049698	-0.014315	0.041418	-0.34141
slope	-0.005582	-0.011749	0.191706	0.000184	0.002933	-0.030353	0.167079	0.03653
ca	0.327055	0.085356	-0.031574	0.087626	0.116030	0.125634	0.008288	-0.23475
thal	0.055572	0.214429	0.544479	0.067002	0.050182	0.004519	0.199077	-0.14090
target	-0.032471	-0.013163	0.221884	-0.000619	0.002746	-0.019852	0.129800	0.04415

Create a heatmap to show the corelationship of data

In [17]:
```
plt.figure(figsize=(16,6))
sns.heatmap(df.corr(),annot = True)
```

Out[17]: <Axes: >

Get the Numbers affacted and not affected by heart disease

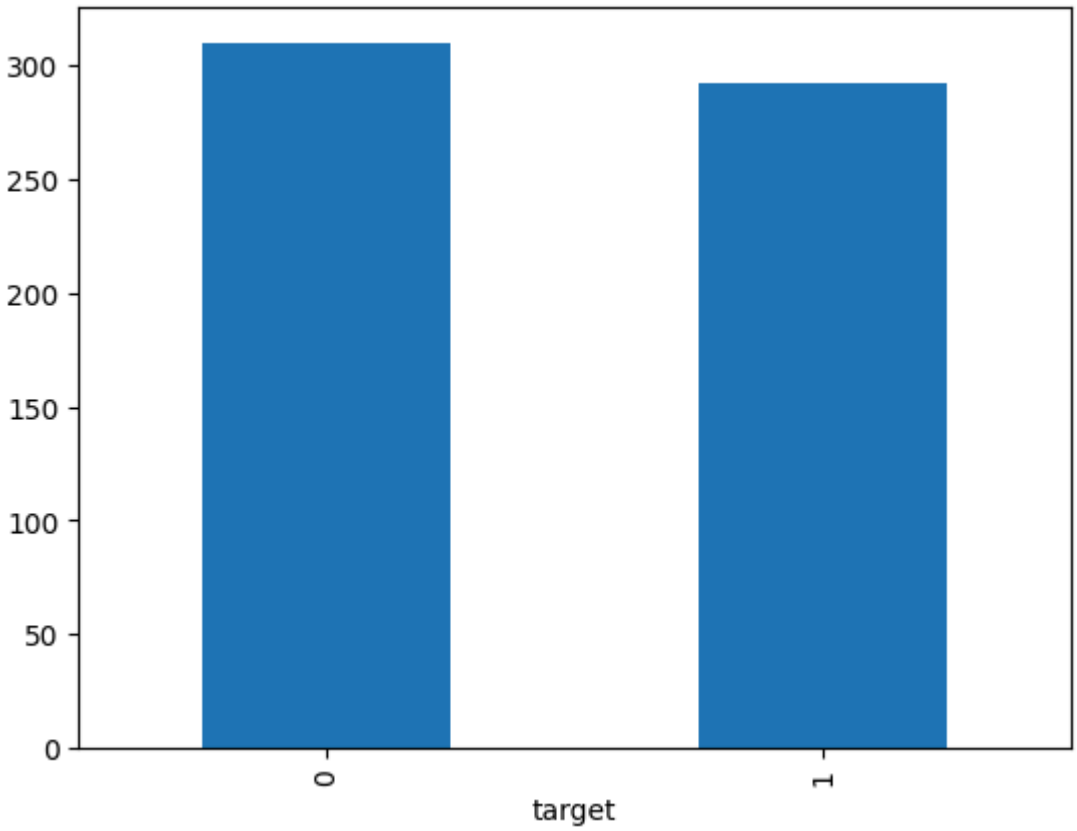
In [18]:
```
# Count the number of people affected and not affected by heart disease
affected_count = (df['target'] == 1).sum()
not_affected_count = (df['target'] == 0).sum()

# Print results
print(f"Number of people affected by heart disease: {affected_count}")
print(f"Number of people not affected by heart disease: {not_affected_
```

```
Number of people affected by heart disease: 292
Number of people not affected by heart disease: 310
```

Create bar chart to show the Numbers affacted and not affected by heart disease.

In [19]:
```
df['target'].value_counts().plot(kind = 'bar')
plt.show
```

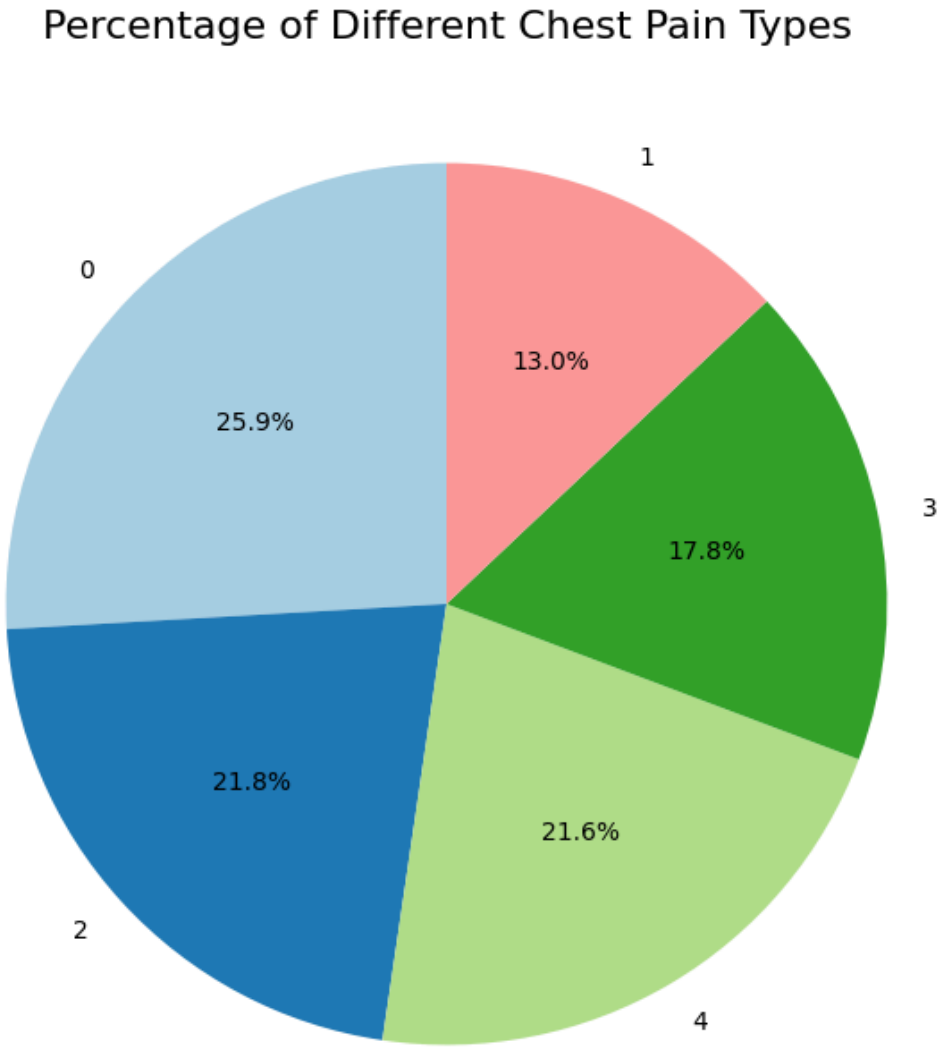
Out[19]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Percentage of different chest pain type

In [20]:

```
chest_pain_col = 'cp'  # Adjust based on your dataset

# Count the occurrences of each chest pain type
chest_pain_counts = df[chest_pain_col].value_counts()

 # Calculate percentages
chest_pain_percentages = chest_pain_counts / chest_pain_counts.sum() *

# Plot the pie chart
plt.figure(figsize=(8, 8))
plt.pie(chest_pain_percentages, labels=chest_pain_counts.index, autopc
plt.title('Percentage of Different Chest Pain Types', fontsize=16)
plt.show()
```



In [21]:
```
#Count the occurance of each chest Pain Type
chest_pain_counts = df['cp'].value_counts()
chest_pain_counts
```

Out[21]:
```
cp
0    156
2    131
4    130
3    107
1     78
Name: count, dtype: int64
```

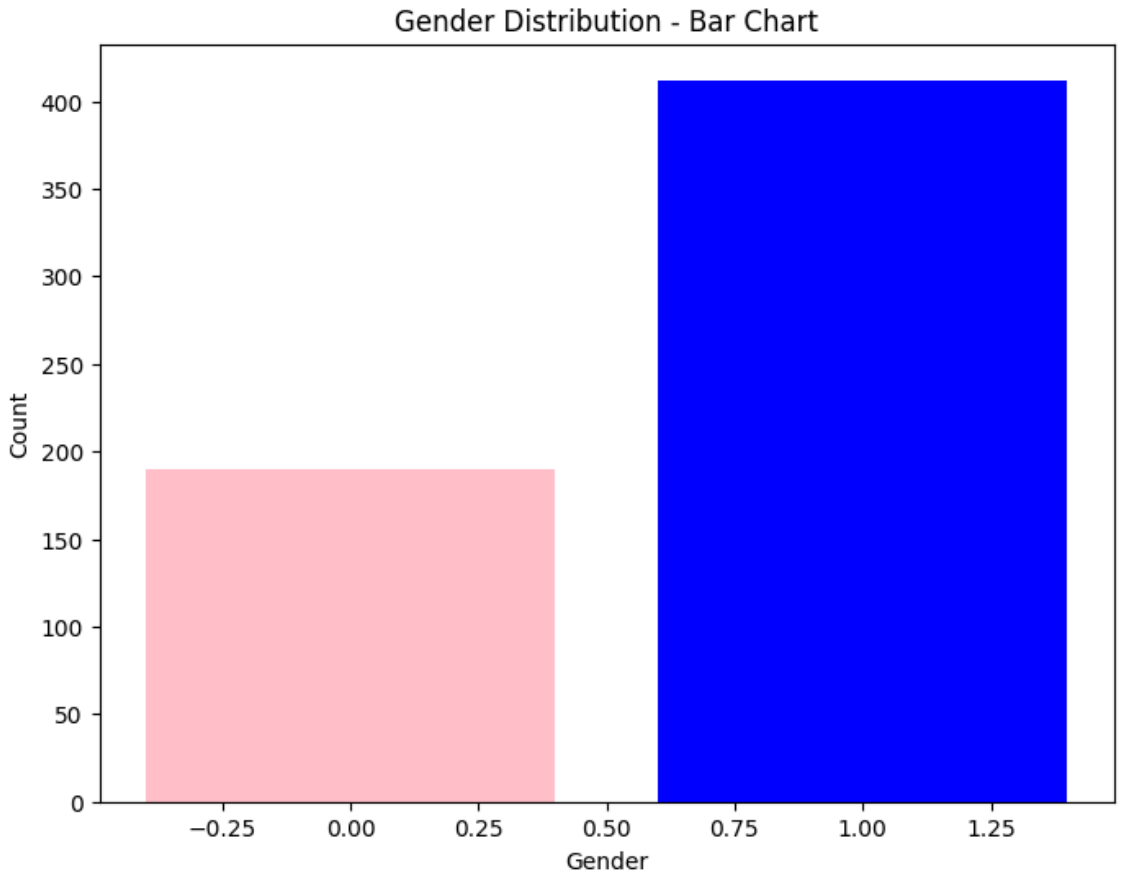
Show the Gender Distribution in dataset using charts

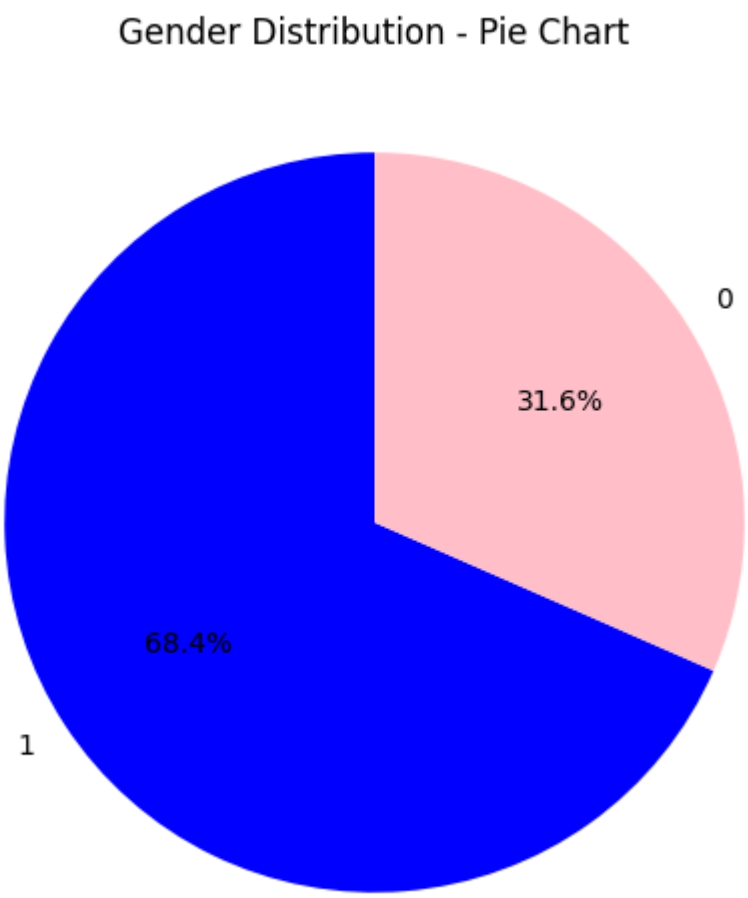
In [22]:
```
# Count the occurrences of each gender
gender_counts = df['sex'].value_counts()
gender_counts
```

Out[22]:
```
sex
1    412
0    190
Name: count, dtype: int64
```

In [23]:

```
# Bar Chart
plt.figure(figsize=(8, 6))
plt.bar(gender_counts.index, gender_counts.values, color=['blue', 'pin
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender Distribution - Bar Chart')
plt.show()

# Pie Chart
plt.figure(figsize=(8, 6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%',
plt.title('Gender Distribution - Pie Chart')
plt.show()
```





Show the Distribution of heart dieasese among males and females

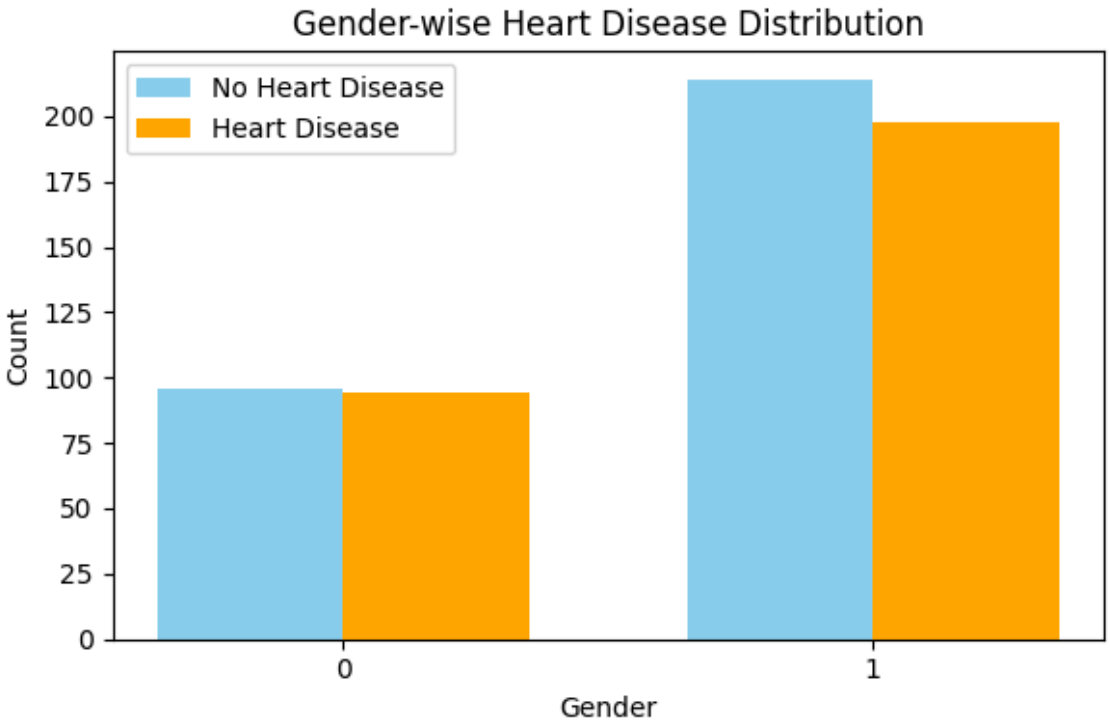
In [24]:
```
# Group by gender and target, then count occurrences
gender_target_counts = df.groupby(['sex', 'target']).size().unstack()
gender_target_counts
```

Out[24]:

target	0	1
sex		
0	96	94
1	214	198

In [25]:

```
# Plotting
bar_width = 0.35
index = np.arange(len(gender_target_counts))

plt.figure(figsize=(6, 4))

# Bars for each target category
plt.bar(index, gender_target_counts[0], bar_width, label='No Heart Dis
plt.bar(index + bar_width, gender_target_counts[1], bar_width, label='

# Adding labels and title
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender-wise Heart Disease Distribution')
plt.xticks(index + bar_width / 2, gender_target_counts.index)
plt.legend()

plt.tight_layout()
plt.show()
```

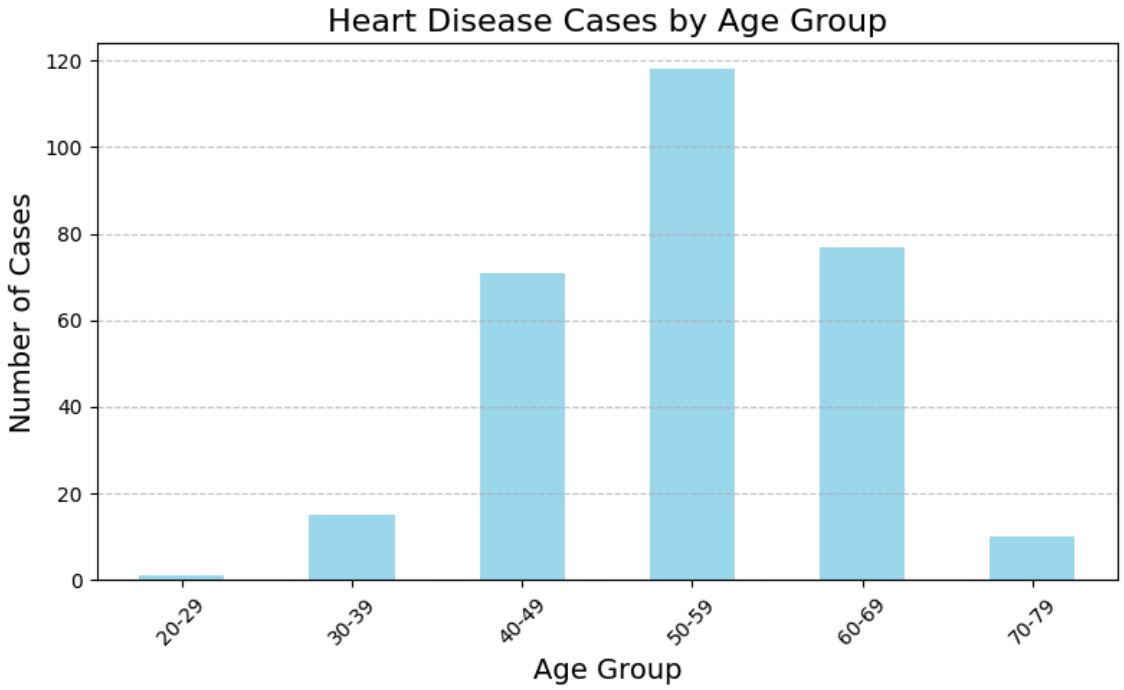


Which age group is most affected by heart desiese?

In [26]:

```
# Define age groups
bins = [20, 30, 40, 50, 60, 70, 80]
labels = ['20-29', '30-39', '40-49', '50-59', '60-69', '70-79']
df['AgeGroup'] = pd.cut(df['age'], bins=bins, labels=labels, right=Fal

# Aggregate data by age group and count heart disease cases
# Assuming 'HeartDisease' is a binary column where 1 indicates heart d
age_group_affected = df[df['target'] == 1].groupby('AgeGroup').size()

# Find the age group with the most cases
most_affected_group = age_group_affected.idxmax()
most_affected_count = age_group_affected.max()

print(f"The age group most affected by heart disease is {most_affected_

# Visualization
plt.figure(figsize=(8, 5))
age_group_affected.plot(kind='bar', color='skyblue', alpha=0.8)
plt.title('Heart Disease Cases by Age Group', fontsize=16)
plt.xlabel('Age Group', fontsize=14)
plt.ylabel('Number of Cases', fontsize=14)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

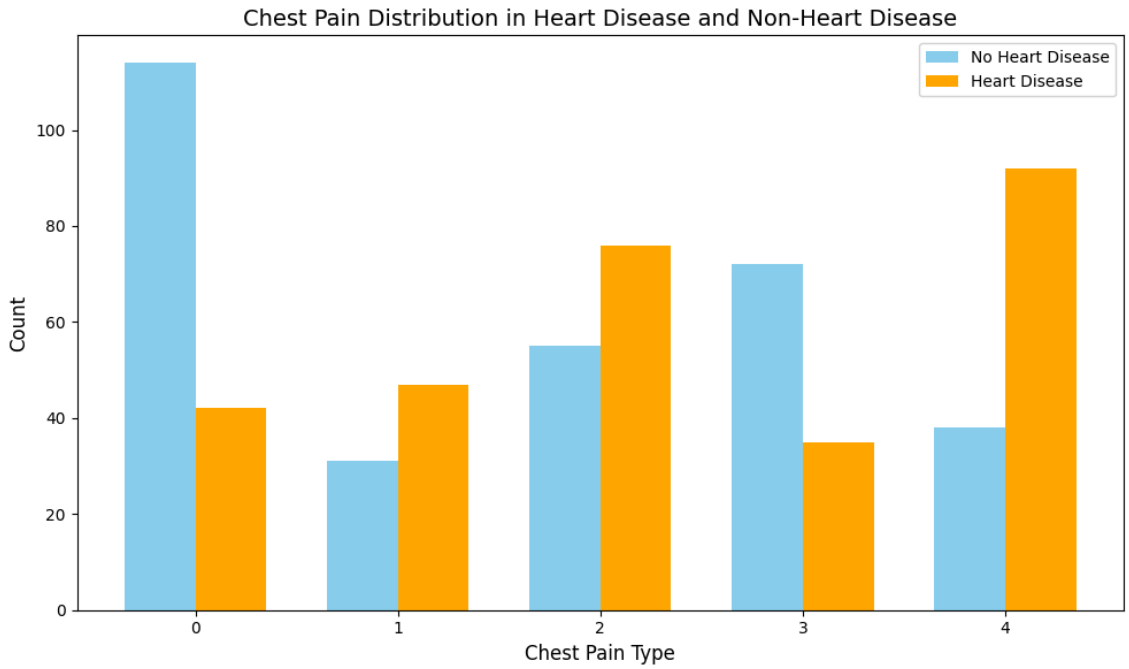
```
C:\Users\Mohammad Adil\AppData\Local\Temp\ipykernel_9900\3749715460.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df['AgeGroup'] = pd.cut(df['age'], bins=bins, labels=labels, right=Fals
e)

The age group most affected by heart disease is 50-59 with 118 cases.
```



Show Chest Pain distribution in Heart Disease vs Non Heart Disease by using graph

In [27]:

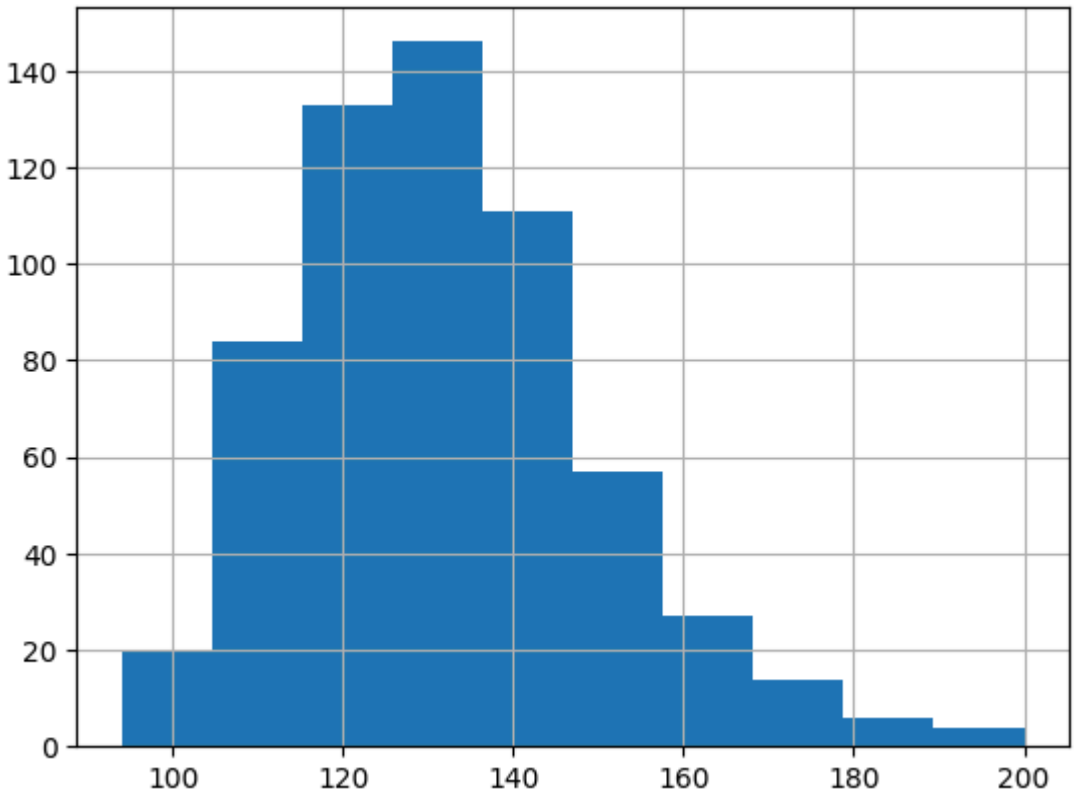
```
# Group data by chest pain type and target
chest_pain_counts = df.groupby(['cp', 'target']).size().unstack(fill_v

# Plotting
bar_width = 0.35
x = np.arange(len(chest_pain_counts))

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

# Bars for heart disease and no heart disease
plt.bar(x - bar_width / 2, chest_pain_counts[0], width=bar_width, colo
plt.bar(x + bar_width / 2, chest_pain_counts[1], width=bar_width, colo

# Adding labels, title, and legend
plt.xlabel('Chest Pain Type', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.title('Chest Pain Distribution in Heart Disease and Non-Heart Disea
plt.xticks(x, chest_pain_counts.index, fontsize=10)
plt.legend()

plt.tight_layout()
plt.show()
```



What is the Resting blood pressure (trestbps) Data Distribtion by graph

In [28]:
```
1 df['trestbps'].hist()
```

Out[28]: <Axes: >



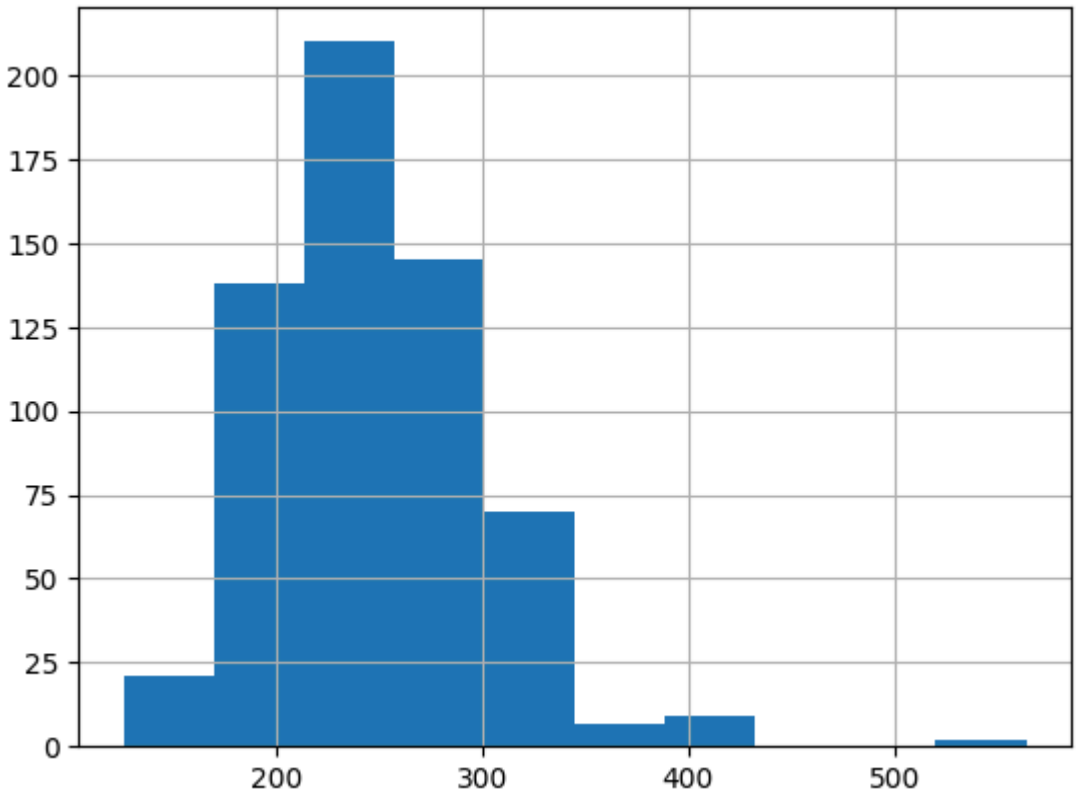
In [29]:
```
1 df.columns
```

Out[29]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach
h',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target', 'AgeGroup'],
      dtype='object')
```

Show the Serum Cholestrol (Chol) Data Distribution by graph

In [30]:
```
1 df['chol'].hist()
```

Out[30]: <Axes: >

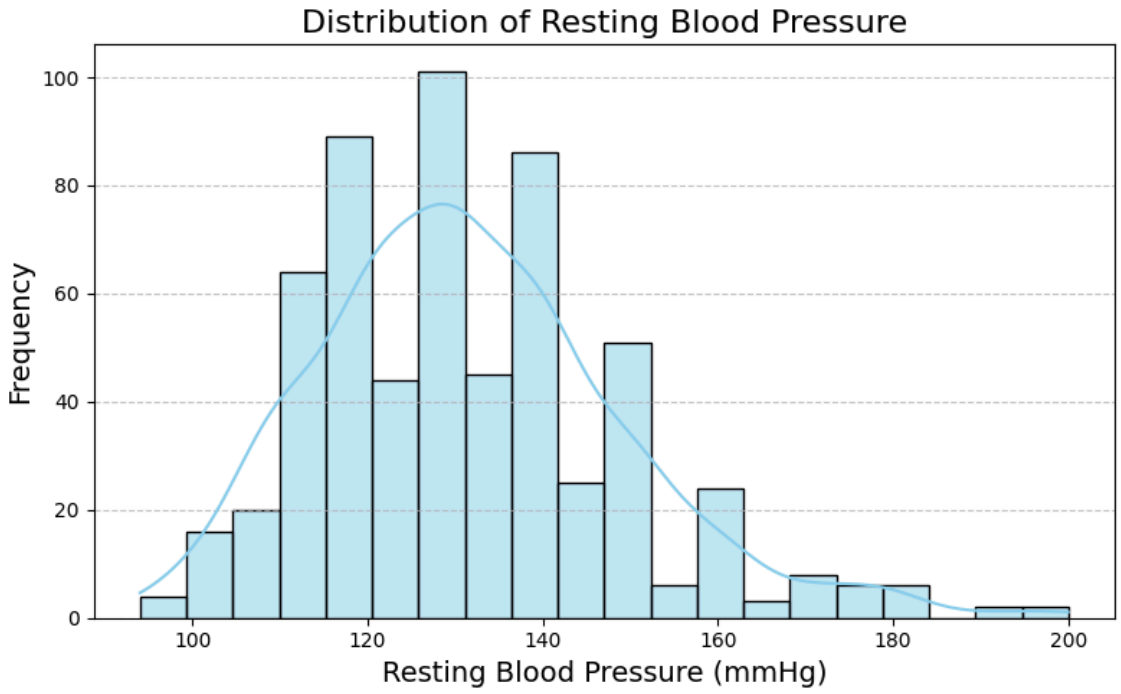


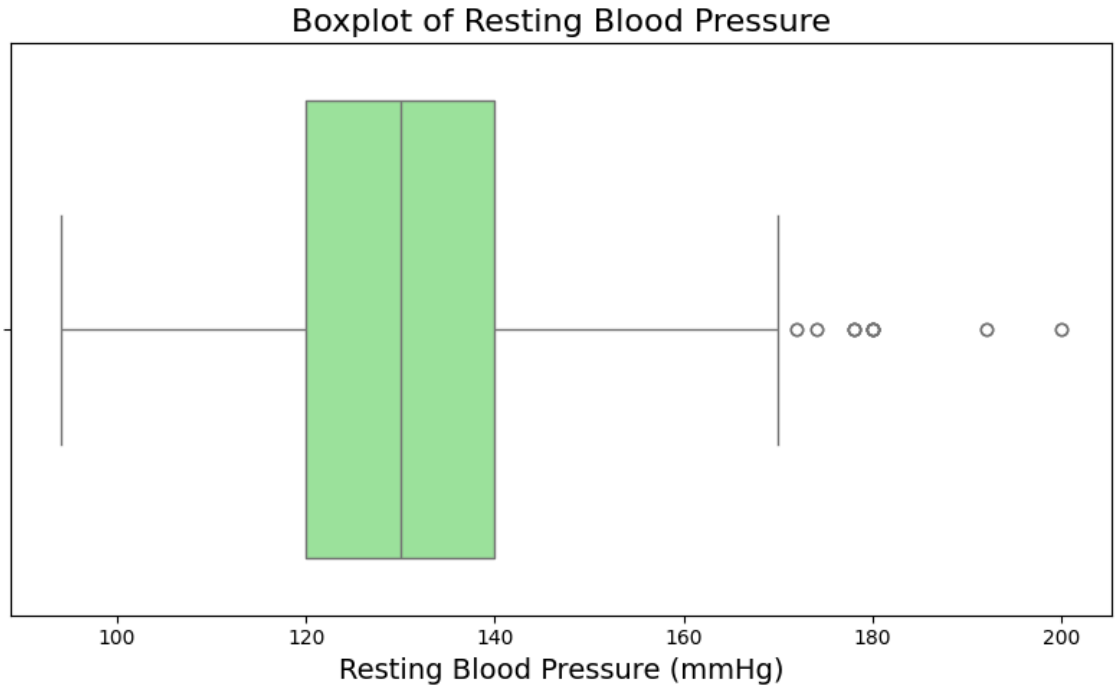
What is the distribution of resting blood pressure among patients?

In [31]:

```
# Replace 'RestingBP' with the actual column name if different
if 'trestbps' in df.columns:
    # Calculate descriptive statistics
    print("Descriptive Statistics for Resting Blood Pressure:")
    print(df['trestbps'].describe())

    # Plot the distribution using a histogram
    plt.figure(figsize=(8, 5))
    sns.histplot(df['trestbps'], bins=20, kde=True, color='skyblue')
    plt.title('Distribution of Resting Blood Pressure', fontsize=16)
    plt.xlabel('Resting Blood Pressure (mmHg)', fontsize=14)
    plt.ylabel('Frequency', fontsize=14)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()

    # Plot the distribution using a boxplot
    plt.figure(figsize=(8, 5))
    sns.boxplot(x=df['trestbps'], color='lightgreen')
    plt.title('Boxplot of Resting Blood Pressure', fontsize=16)
    plt.xlabel('Resting Blood Pressure (mmHg)', fontsize=14)
    plt.tight_layout()
    plt.show()
else:
    print("The column for resting blood pressure (e.g., 'RestingBP') i
```

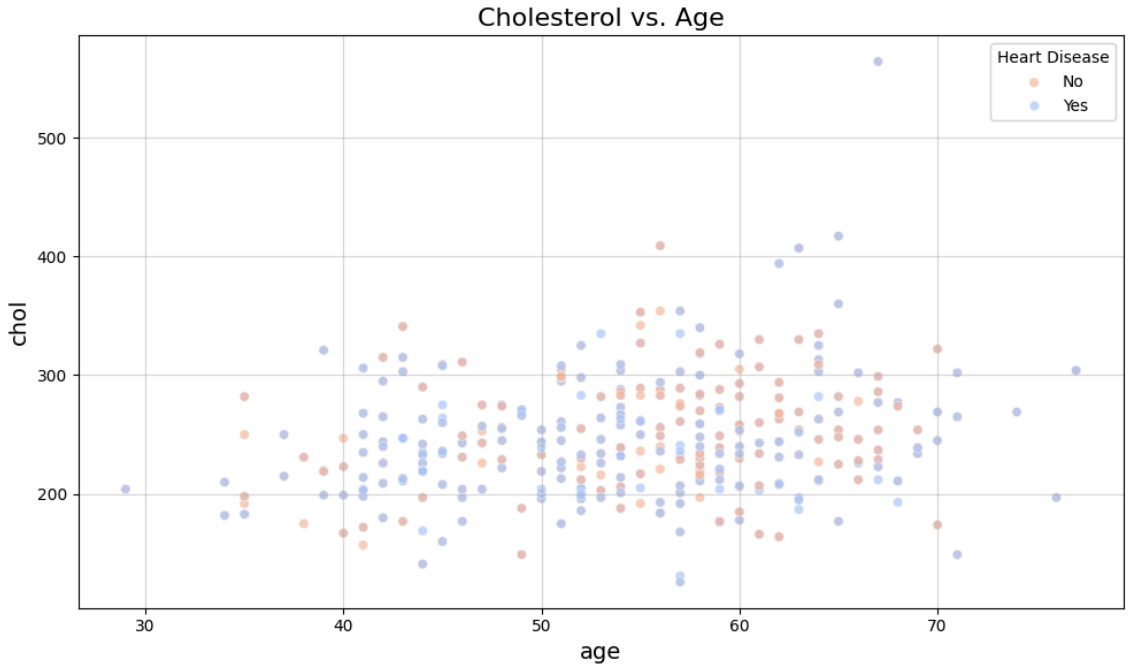
```
Descriptive Statistics for Resting Blood Pressure:
count    602.000000
mean     131.637874
std       17.509164
min       94.000000
25%      120.000000
50%      130.000000
75%      140.000000
max      200.000000
Name: trestbps, dtype: float64
```





In [32]:

```
# Scatter plot for Cholesterol vs. Age
plt.figure(figsize=(10, 6))
sns.scatterplot(x='age', y='chol', hue='target', data=df, palette='coo
plt.title('Cholesterol vs. Age', fontsize=16)
plt.xlabel('age', fontsize=14)
plt.ylabel('chol', fontsize=14)
plt.grid(alpha=0.5)
plt.legend(title='Heart Disease', labels=['No', 'Yes'])
plt.tight_layout()
plt.show()
```



Conclusion of the Project

In this project, we analyzed a heart-related dataset to understand key health metrics and their relationship to heart disease. By exploring age groups, we identified that middle-aged individuals were the most affected by heart disease. The distribution of resting blood pressure and cholesterol revealed significant variability, with some outliers indicating potential risk factors. A positive correlation between age and cholesterol levels highlighted the growing risk of heart conditions with age. Gender-based analysis showed disparities in heart disease prevalence, emphasizing the need for targeted awareness. Visualizations such as histograms, boxplots, and scatter plots effectively illustrated these trends, providing actionable insights for preventive measures. This project underscores the importance of monitoring key health indicators and adopting healthier lifestyles to mitigate heart disease risks.

Key Insights:

Demographic Trends: (e.g., The majority of patients with heart conditions were in the age group of 40–60 years. Men were more affected than women in this dataset.)

Health Metrics: (e.g., High cholesterol levels and blood pressure were observed as common factors among patients with heart conditions.)

Relationships: (e.g., A positive correlation was found between age and cholesterol levels, indicating an increased risk with age.)