

```
In [3]: # New Python Project - Heart - Walid Saleh
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
In [4]: import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
In [5]: df = pd.read_csv('/content/drive/MyDrive/Python/heart.csv')
```

```
In [6]: df.head()
```

```
Out[6]:
```

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

```
In [7]: df.dtypes
```

Out[7]:

0

age	int64
sex	int64
cp	int64
trestbps	int64
chol	int64
fbs	int64
restecg	int64
thalach	int64
exang	int64
oldpeak	float64
slope	int64
ca	int64
thal	int64
target	int64

dtype: object

In [8]: `df.columns`

Out[8]: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
 dtype='object')

In [9]: `df.isnull().sum()`

Out[9]:

	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slope	0
ca	0
thal	0
target	0

dtype: int64

```
In [10]: missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage > threshold].index
```

```
In [11]: # Drop the columns
df = df.drop(columns=columns_to_drop)

# Display the resulting dataframe after removing columns with too many missing values
```

```
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

Columns dropped: []

Remaining columns: Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'], dtype='object')

```
In [12]: import pandas as pd
        from sklearn.preprocessing import LabelEncoder

        # Assuming df is not defined, Load it here for this cell to run independently
        df = pd.read_csv('/content/drive/MyDrive/Python/heart.csv')

        label_encoder = LabelEncoder()

        for column in df.select_dtypes(include=['object']).columns:

            df[column] = label_encoder.fit_transform(df[column])

        df.head()
```

```
Out[12]:
```

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

```
In [13]: # Fill missing values in columns with mode
        for column in df.select_dtypes(include=['float64']).columns:
            mean_value = df[column].mean() # Get the mode (most frequent value)
            df[column] = df[column].fillna(mean_value)

        # Fill missing values in categorical columns with mode
        for column in df.select_dtypes(include=['object']).columns:
```

```
mode_value = df[column].mode()[0] # Get the mode (most frequent value)
df[column] = df[column].fillna(mode_value)

df.isnull().sum()
```

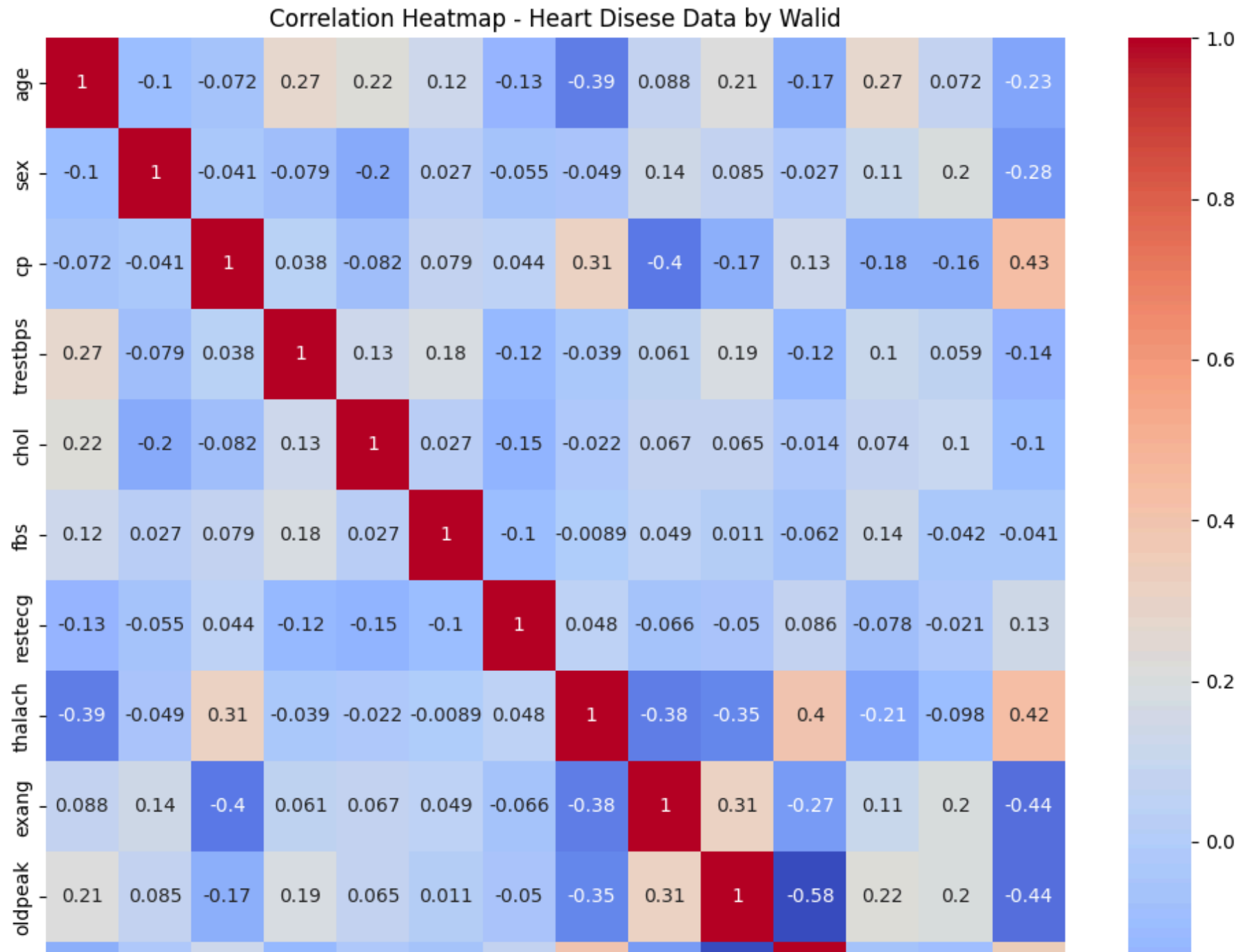
Out[13]:

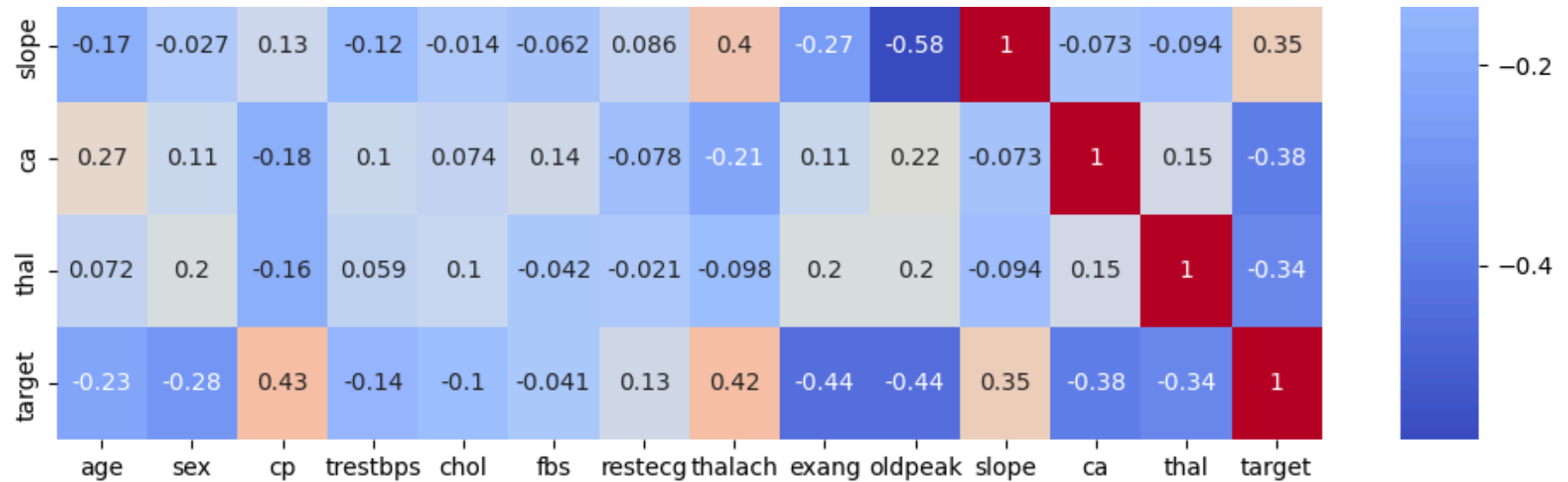
	0
age	0
sex	0
cp	0
trestbps	0
chol	0
fbs	0
restecg	0
thalach	0
exang	0
oldpeak	0
slope	0
ca	0
thal	0
target	0

dtype: int64

```
In [14]: # Data Visualization
# Correlation heatmap
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

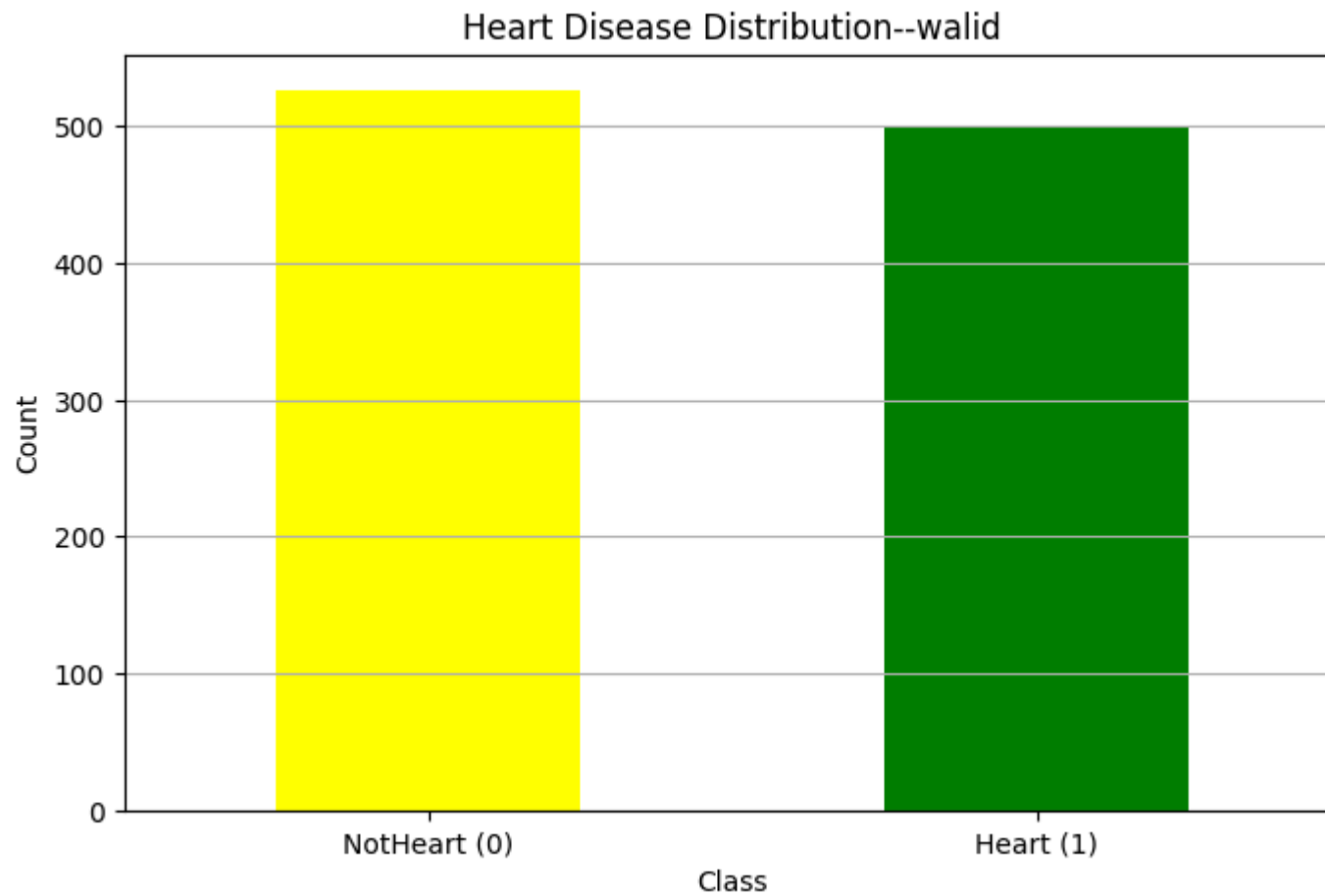
```
plt.title('Correlation Heatmap - Heart Disease Data by Walid')  
plt.show()
```





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

# Create a bar chart
plt.figure(figsize=(8, 5))
df['target'].value_counts().plot(kind='bar', color=['yellow', 'green'])
plt.xticks([0, 1], ['NotHeart (0)', 'Heart (1)'], rotation=0)
plt.title('Heart Disease Distribution--walid')
plt.xlabel('Class')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```



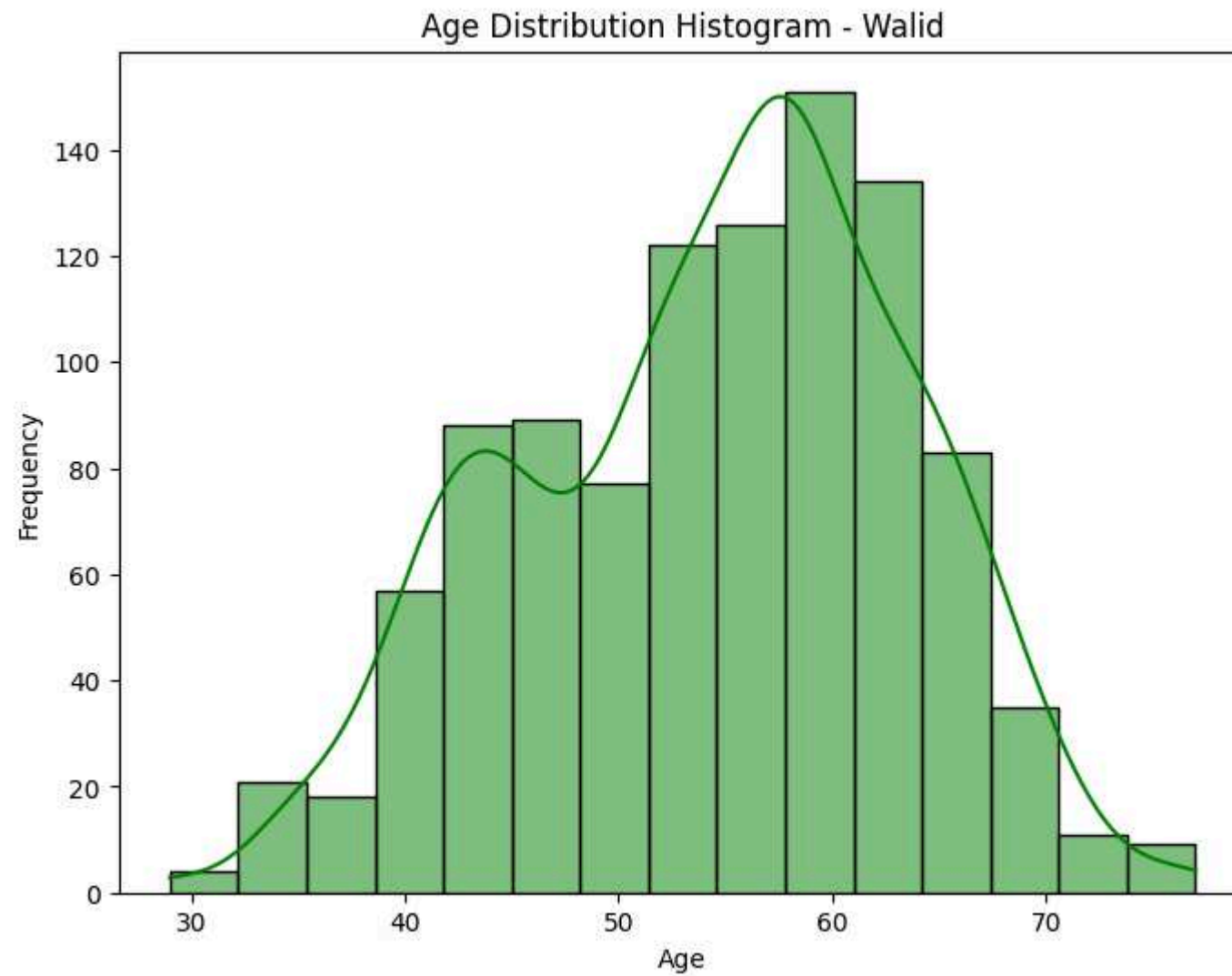
```
In [16]: #Histogram

cat_cols = [col for col in df.columns if df[col].dtype=='object']
num_cols = [col for col in df.columns if df[col].dtype!='object']

for col in cat_cols:
    print(f'{col} has {df[col].unique()} values')

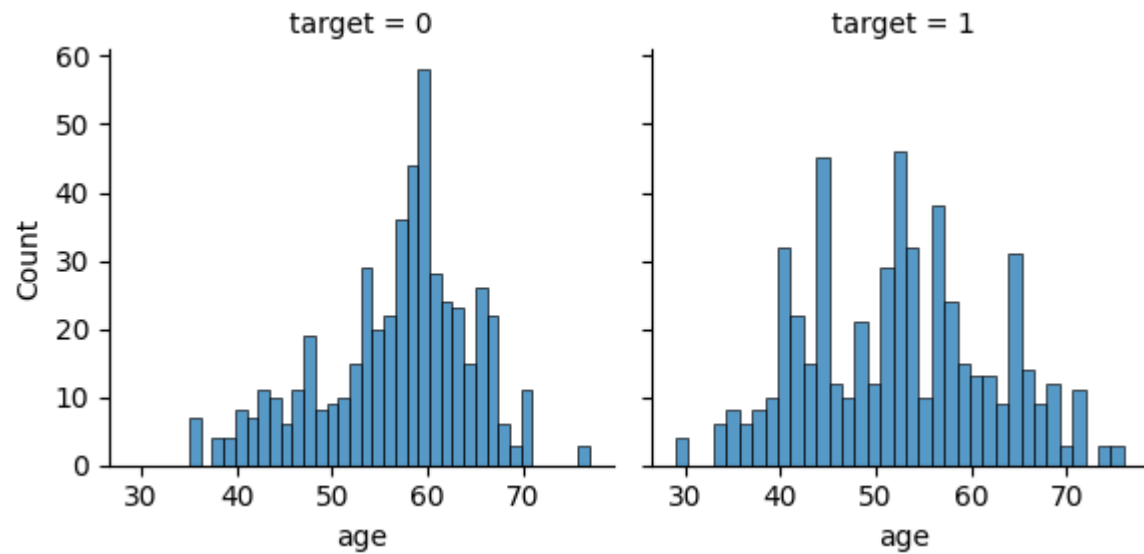
# 1. Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['age'], bins=15, kde=True, color='green')
plt.title('Age Distribution Histogram - Walid')
```

```
plt.xlabel('Age')  
plt.ylabel('Frequency')  
plt.show()
```



```
In [17]: # Histogram  
  
g = sns.FacetGrid(df,col="target")
```

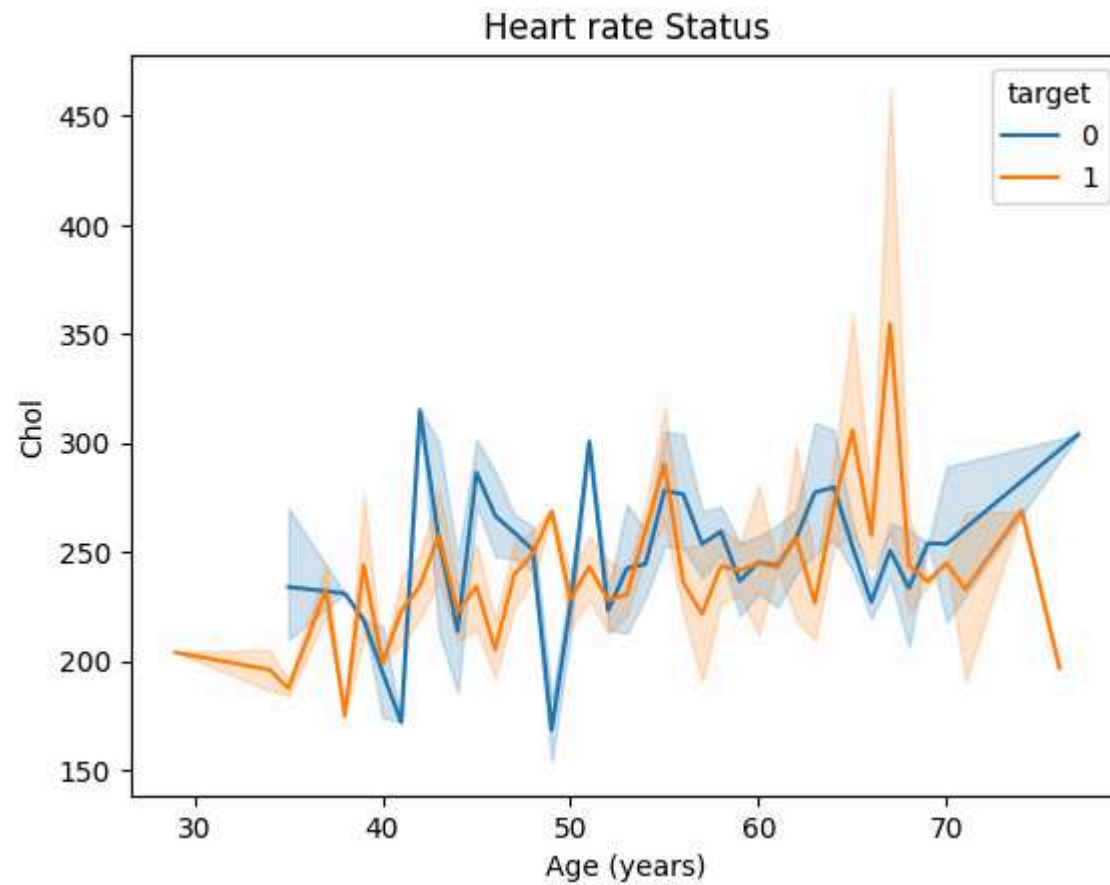
```
g.map(sns.histplot, "age", bins=35)  
plt.show()
```



```
In [18]: trestbps_counts = df['trestbps'].value_counts()  
  
plt.figure(figsize=(5,5))  
plt.pie(trestbps_counts, labels=trestbps_counts.index, autopct='%1.1f%%', startangle=90)  
plt.title("Trestbps Distribution")  
plt.show()
```



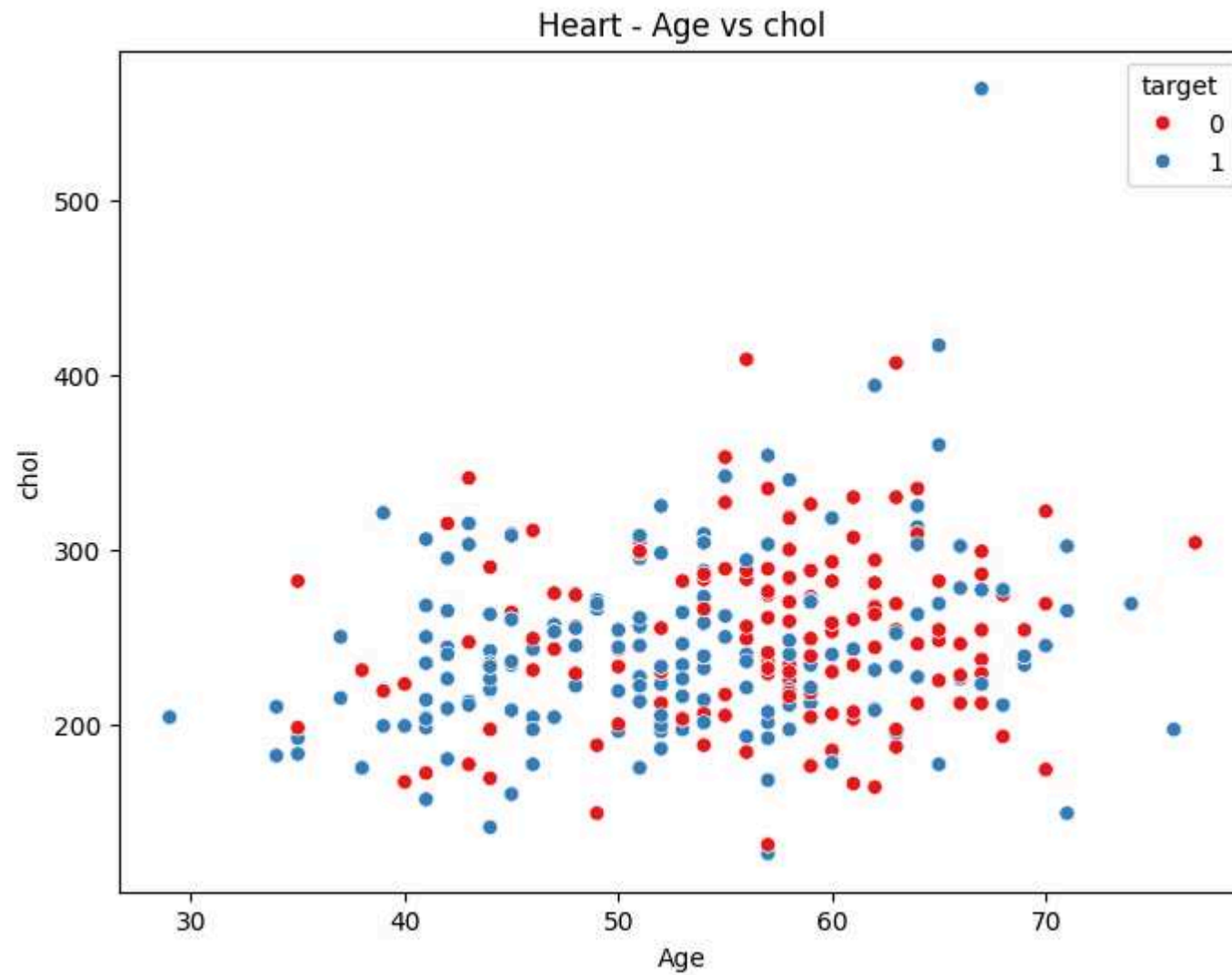
```
In [19]: #line plot
```



```
In [20]: #Scatter plot
import pandas as pd

df = pd.read_csv('/content/drive/MyDrive/Python/heart.csv')

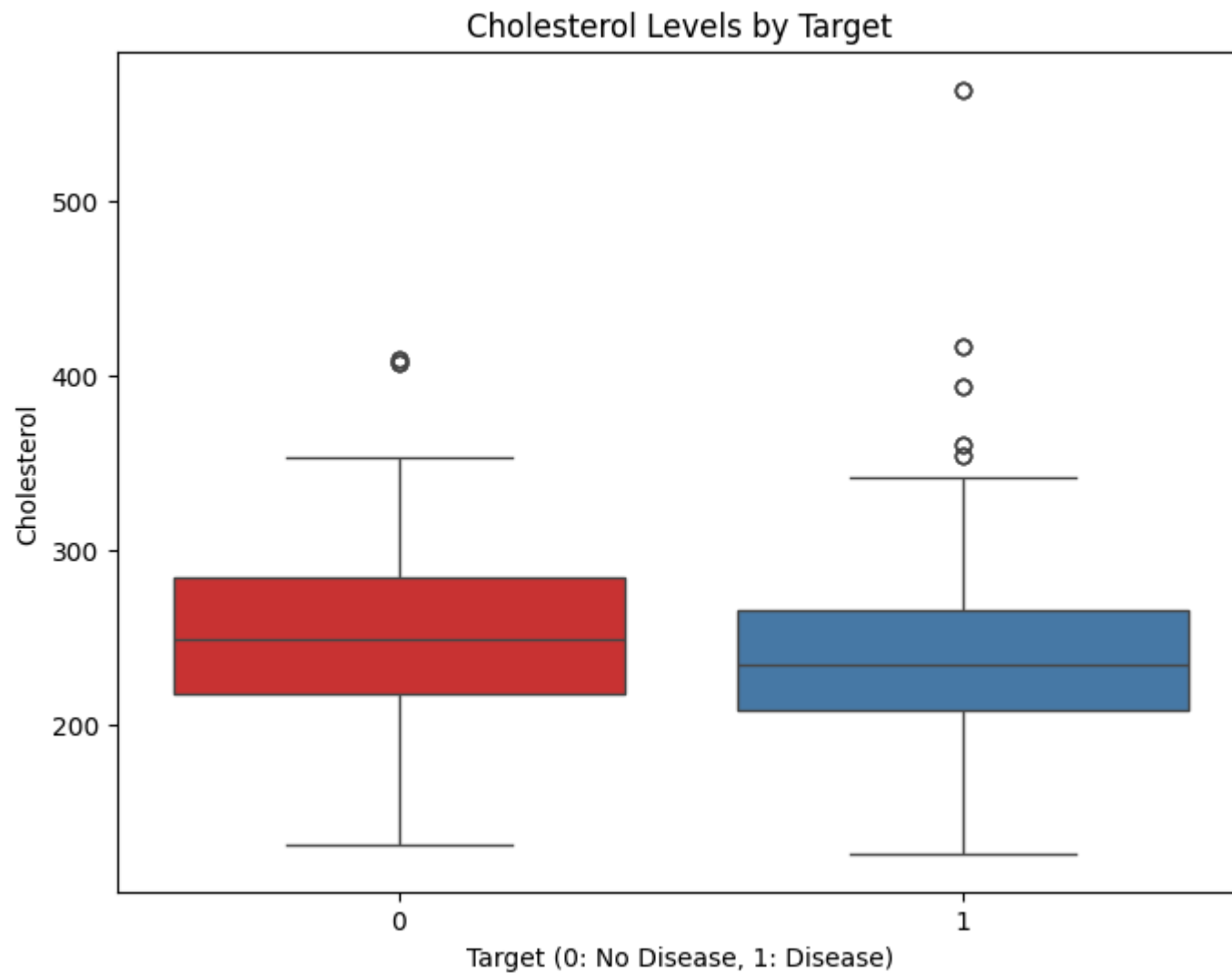
plt.figure(figsize=(8, 6))
sns.scatterplot(x='age', y='chol', data=df, hue='target', palette='Set1')
plt.title('Heart - Age vs chol')
plt.xlabel('Age')
plt.ylabel('chol')
plt.show()
```



```
In [21]: import matplotlib.pyplot as plt
import seaborn as sns

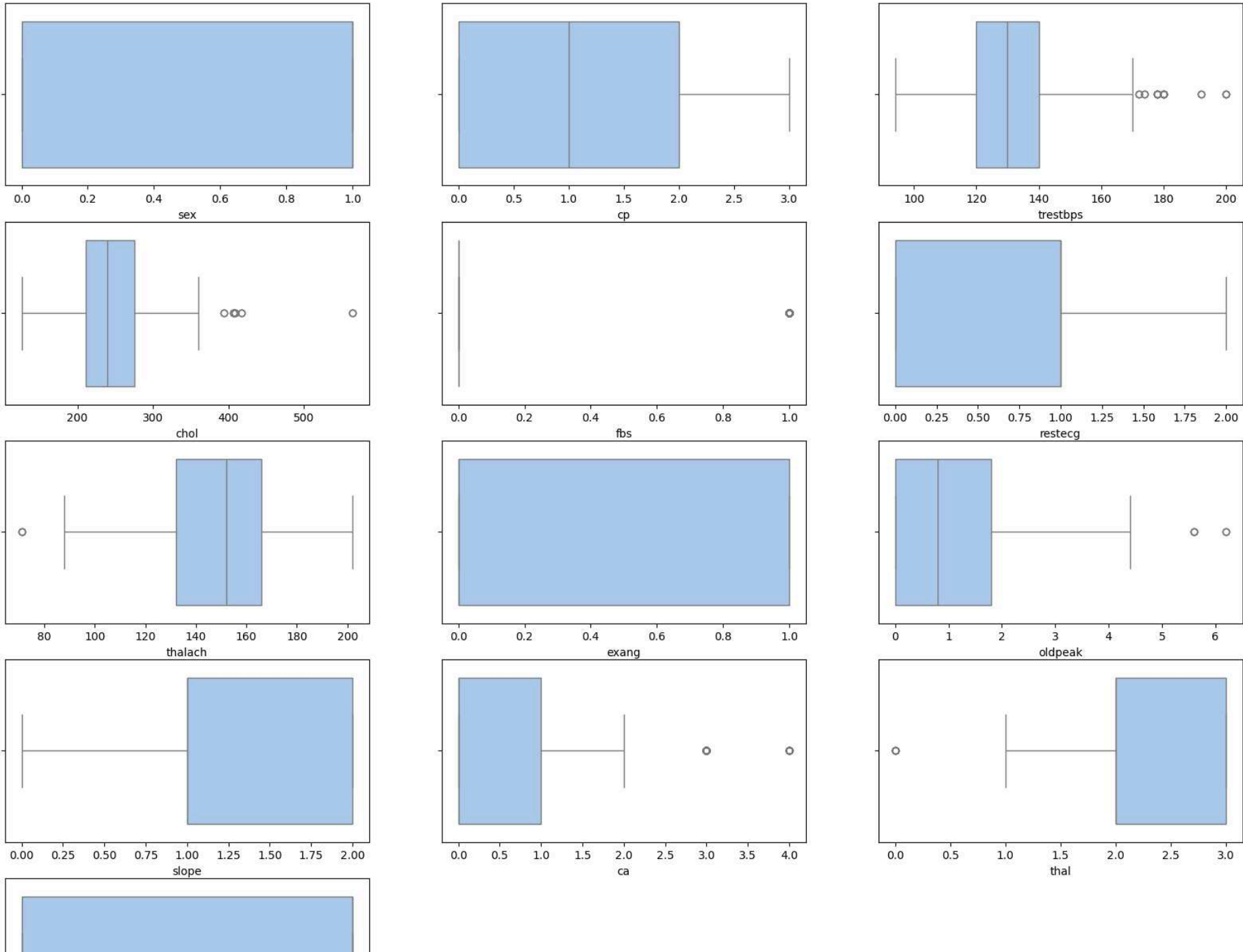
# Boxplot for Chol
plt.figure(figsize=(8, 6))
sns.boxplot(x='target', y='chol', data=df, palette='Set1')
plt.title('Cholesterol Levels by Target')
```

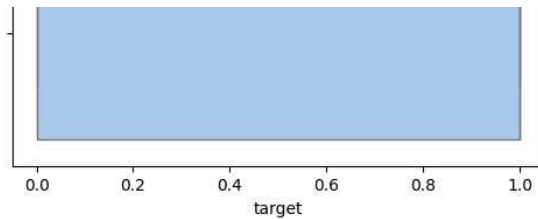
```
plt.xlabel('Target (0: No Disease, 1: Disease)')  
plt.ylabel('Cholesterol')  
plt.show()
```



```
In [22]: plt.figure(figsize=(20,35))  
plotnumber =1  
for column in df.columns[1:]:  
    ax = plt.subplot(10,3,plotnumber)
```

```
sns.boxplot(data = df, x = column, palette='pastel')  
plt.xlabel(column)  
plotnumber+=1  
plt.show()
```





```
In [23]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Example colors and data (you can replace with your actual values)
colors = ['lightcoral', 'skyblue']
labels = ['Female', 'Male']

# Set order based on actual data
order = df['sex'].value_counts().index
# --- Set up the figure ---
plt.figure(figsize=(16, 8))
plt.suptitle('Sex (Gender) Distribution', fontsize=16, fontweight='bold')

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['sex'].value_counts(), labels=labels, colors=colors, autopct='%.2f%%', startangle=90)

# Add yellow circle in the middle for donut effect
centre = plt.Circle((0, 0), 0.45, color='yellow')
plt.gca().add_artist(centre)

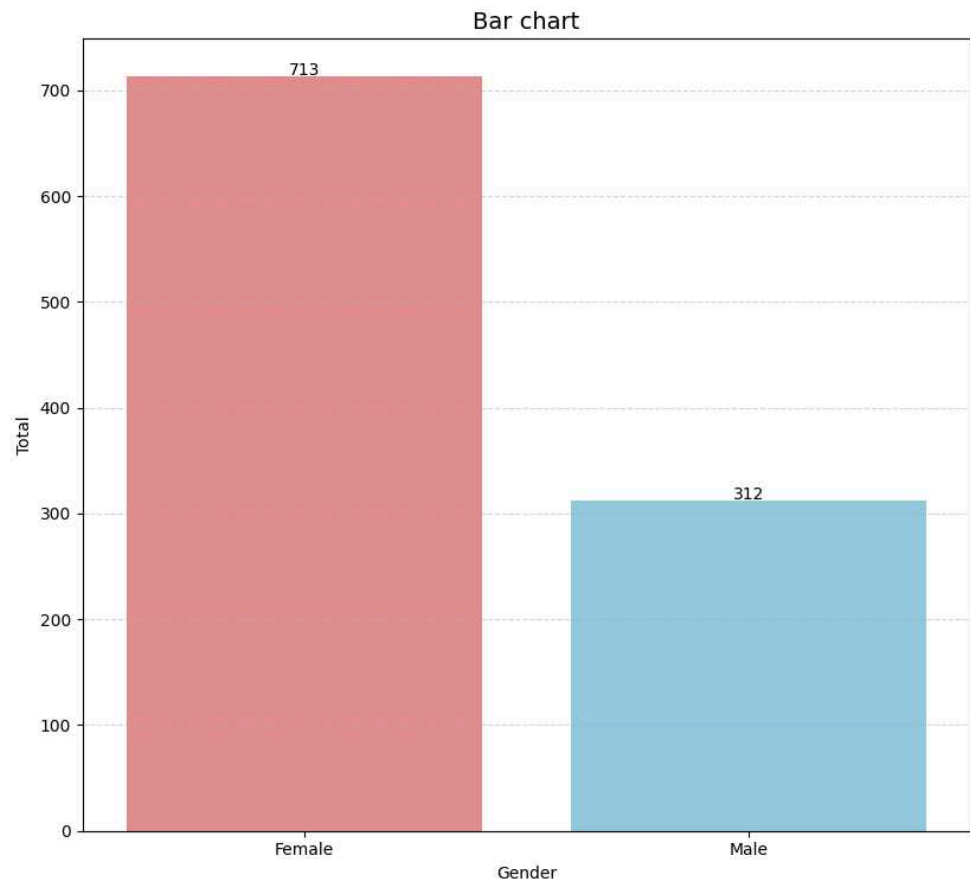
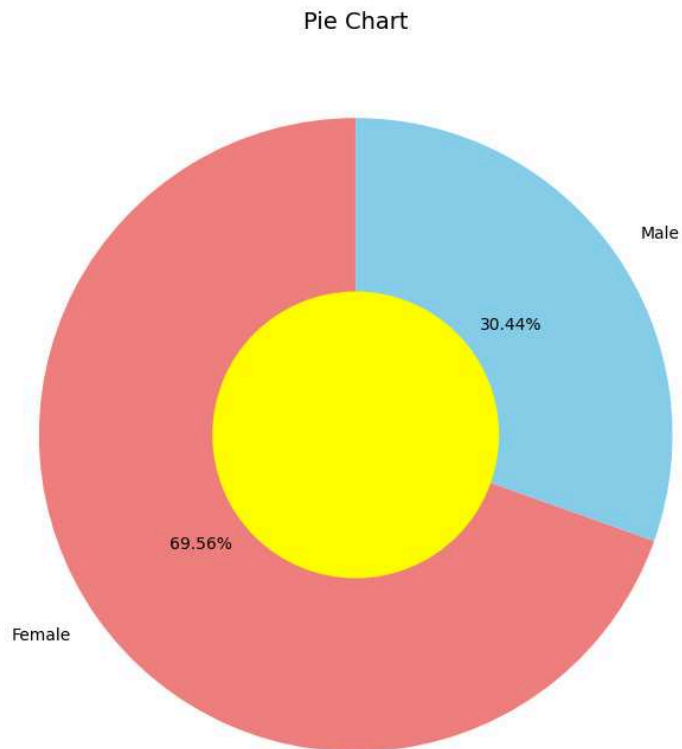
# --- Histogram (Count Plot) ---
plt.subplot(1, 2, 2)
plt.title('Bar chart', fontsize=14)
ax = sns.countplot(x='sex', data=df, palette=colors, order=order)

# Add counts on top of bars
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 1, int(height), ha='center', fontsize=10)
```

```
plt.xlabel('Gender')
plt.ylabel('Total')
plt.xticks([0, 1], labels)
plt.grid(axis='y', linestyle='--', alpha=0.5)

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

Sex (Gender) Distribution



```
In [24]: # Example setup
colors = ['lightpink', 'hotpink', 'deeppink', 'mediumvioletred']
labels = ['Type 0', 'Type 2', 'Type 1', 'Type 3']
order = df['cp'].value_counts().index # Ensure bars follow actual data order
```

```
# --- Create Figure ---
plt.figure(figsize=(16, 8))
plt.suptitle('Chest Pain Type Distribution', fontsize=16, fontweight='bold')

# --- Pie Chart ---
plt.subplot(1, 2, 1)
plt.title('Pie Chart', fontsize=14)
plt.pie(df['cp'].value_counts(), labels=labels, colors=colors, autopct='%.2f%', startangle=90)
centre = plt.Circle((0, 0), 0.45, color='white') # Donut-style centerplt.gca().add_artist(centre)

# --- Histogram (Count Plot) ---
plt.subplot(1, 2, 2)
plt.title('Histogram', fontsize=14)
ax = sns.countplot(x='cp', data=df, palette=colors, order=order)

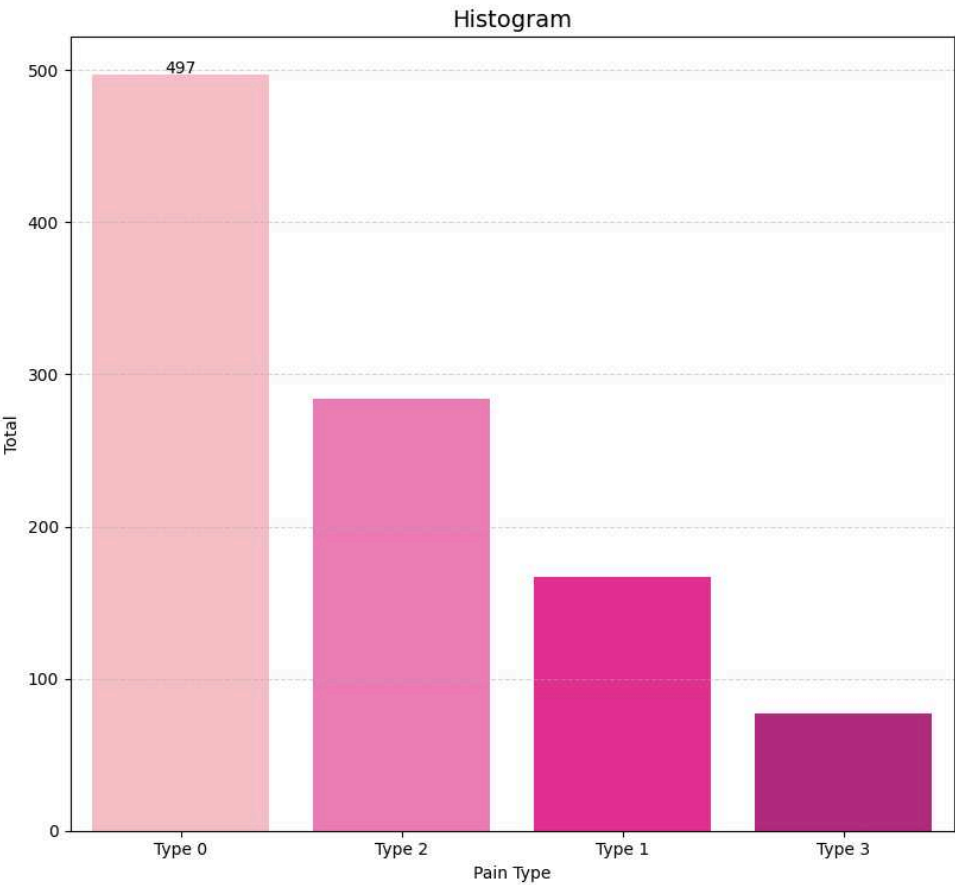
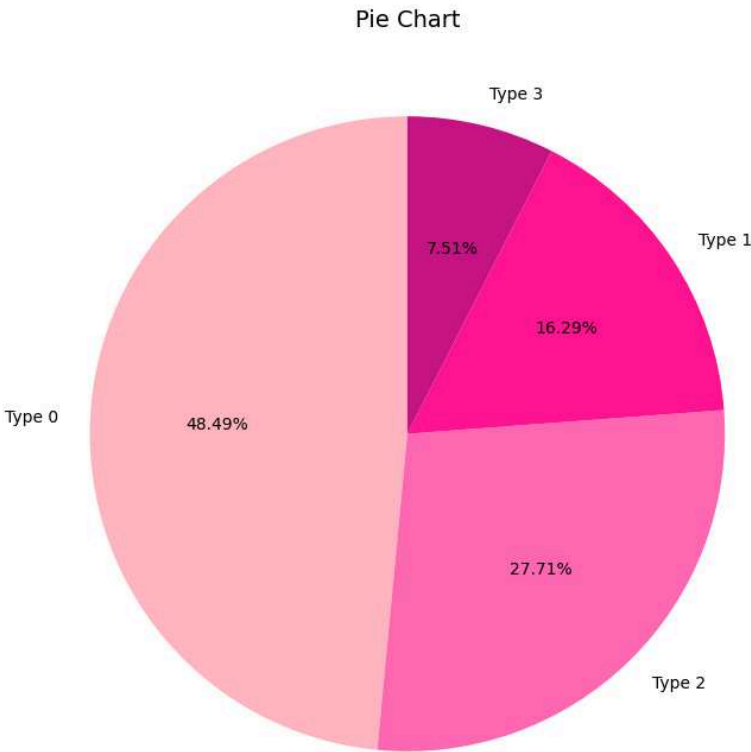
# Add count Labels on top of bars
for bar in ax.patches:
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2, height + 1, int(height), ha='center', fontsize=10)

plt.xlabel('Pain Type')
plt.ylabel('Total')
plt.xticks([0, 1, 2, 3], labels)
plt.grid(axis='y', linestyle='--', alpha=0.5)

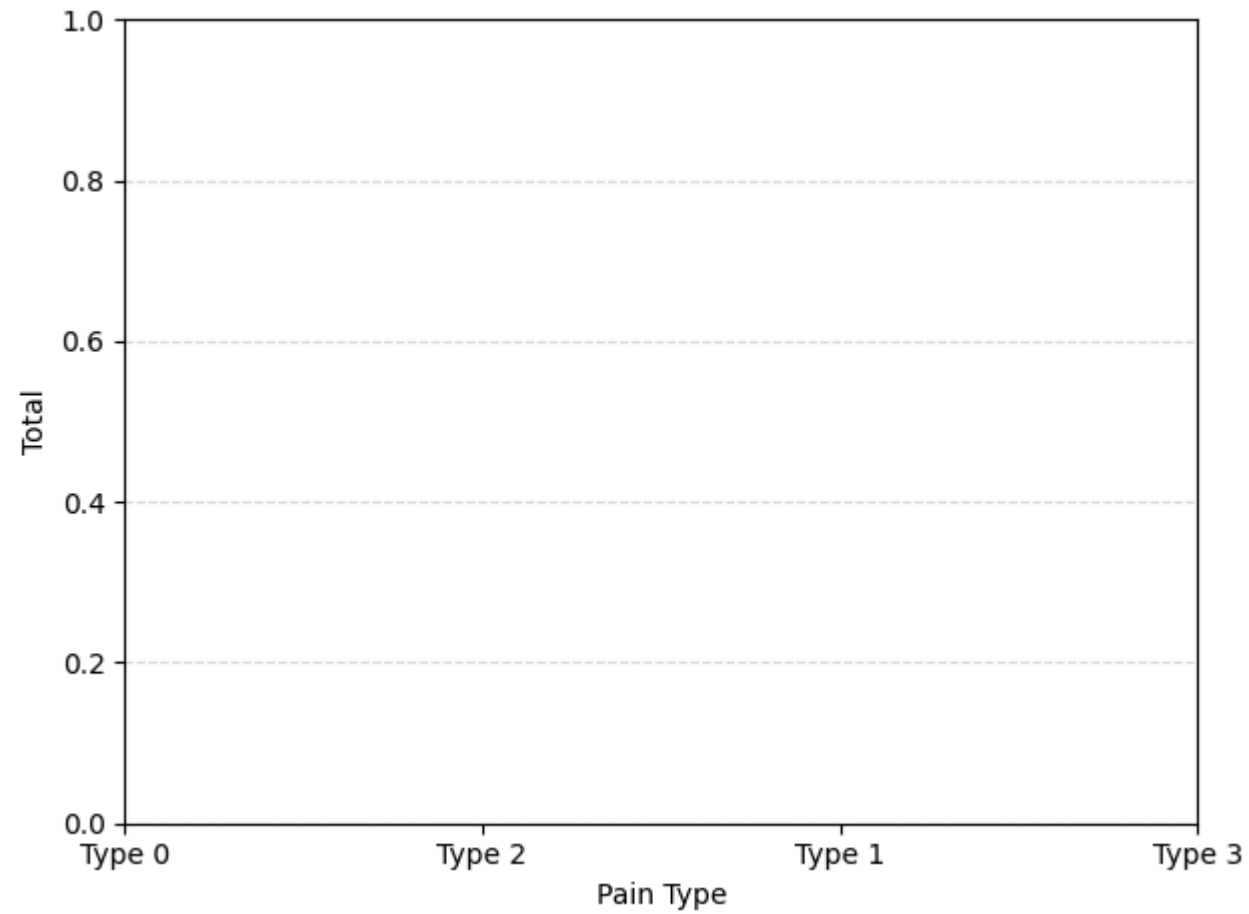
plt.tight_layout()
plt.show()

# --- Display value counts ---
print('*' * 30)
print('Chest Pain Type Total')
print('*' * 30)
print(df['cp'].value_counts(dropna=False))
```

Chest Pain Type Distribution



```
*****
Chest Pain Type Total
*****
cp
0    497
2    284
1    167
3     77
Name: count, dtype: int64
```



Chest Pain Type Total

cp

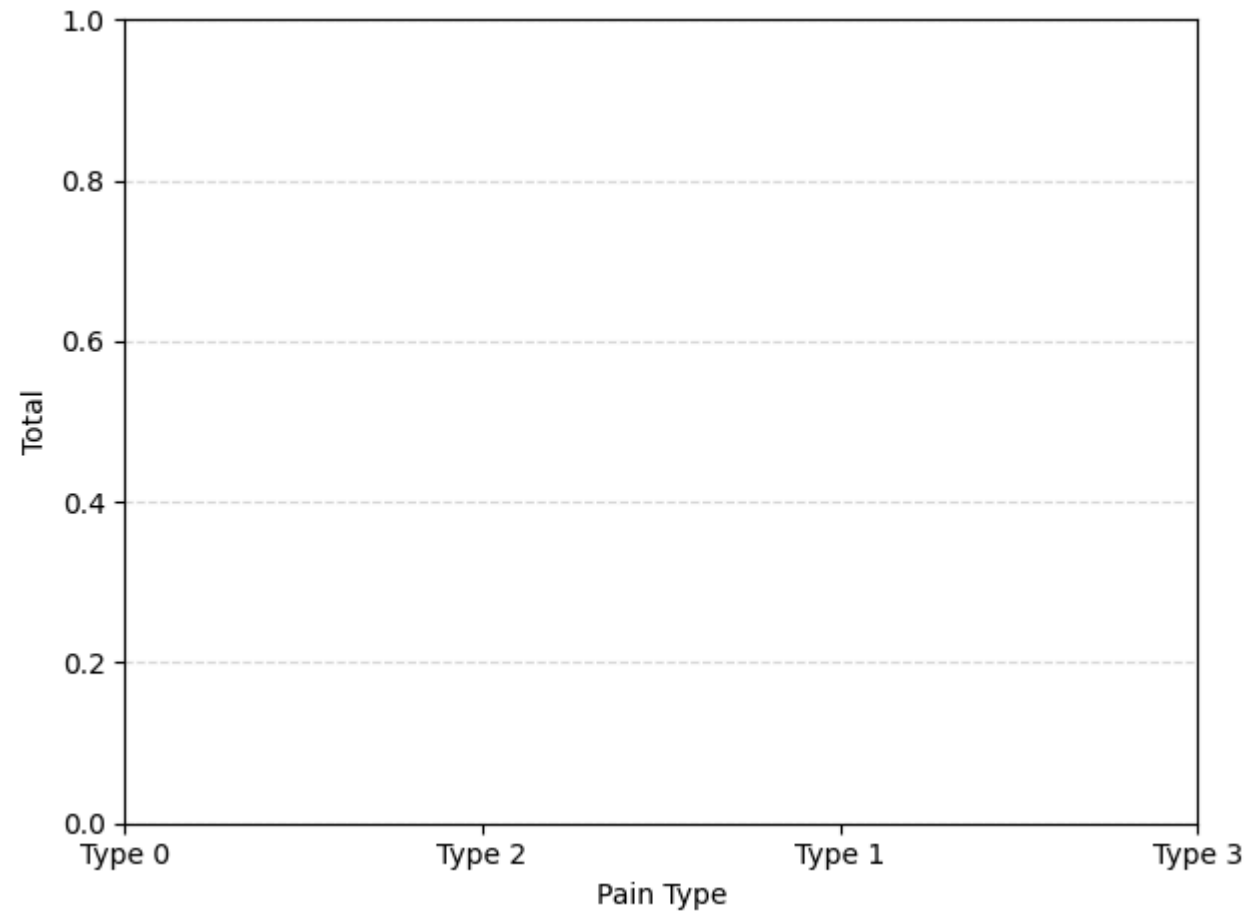
0 497

2 284

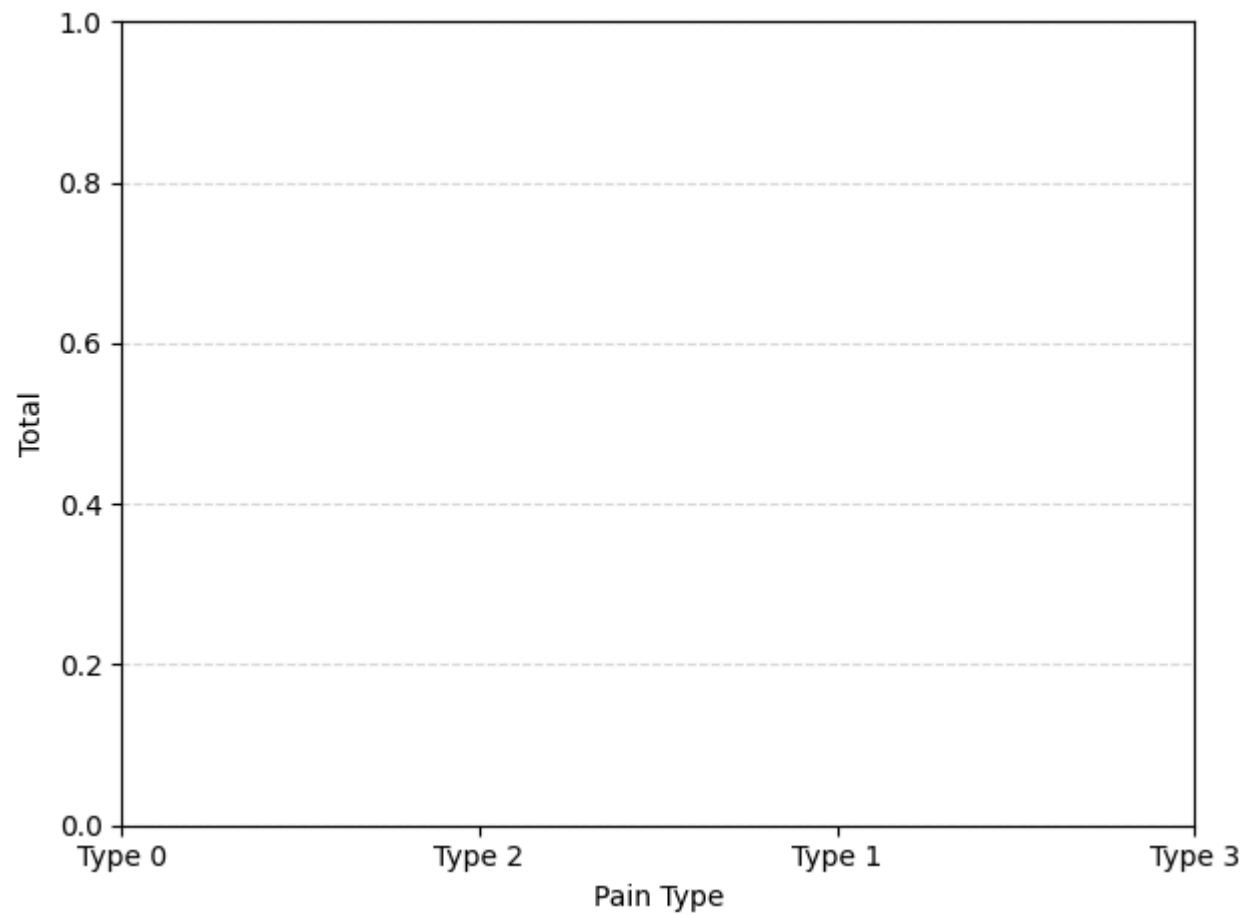
1 167

3 77

Name: count, dtype: int64



```
*****
Chest Pain Type Total
*****
cp
0    497
2    284
1    167
3     77
Name: count, dtype: int64
```



Chest Pain Type Total

cp

0 497

2 284

1 167

3 77

Name: count, dtype: int64

```
In [25]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
# --- Label Settings ---
labels = ['False', 'True'] # For target: 0 = no disease, 1 = disease
label_gender = [0, 1]
label_gender2 = ['Male', 'Female']

# --- Create Bar Chart ---
ax = pd.crosstab(df.sex, df.target).plot(
    kind='bar',
    figsize=(8, 5),
    color=['skyblue', 'lightcoral'],
    edgecolor='black',
    alpha=0.85)

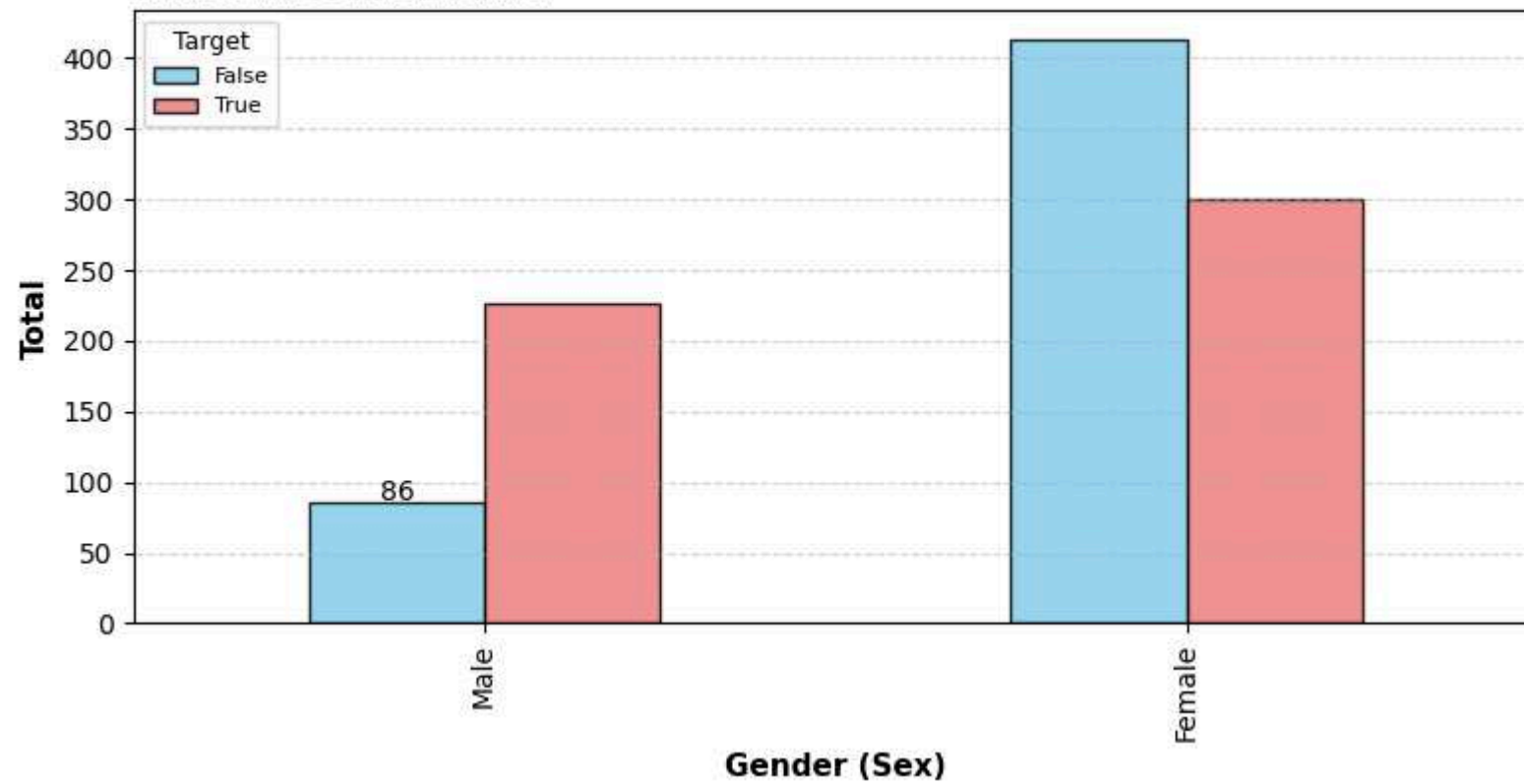
# --- Annotate Bars ---
for rect in ax.patches:
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width() / 2, height + 1, int(height), ha='center', fontsize=10)

# --- Titles and Labels ---
plt.suptitle('Heart Disease Distribution by Gender', fontsize=16, fontweight='bold', x=0.05, ha='left')
plt.title("Females tend to have heart disease more evenly distributed.\nIn males, there's more imbalance.", fontsize=9, loc=
plt.xlabel('Gender (Sex)', fontsize=11, fontweight='bold')
plt.ylabel('Total', fontsize=11, fontweight='bold')
plt.xticks(label_gender, label_gender2)
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.grid(axis='x', alpha=0)

# --- Legend ---
plt.legend(labels=labels, title='Target', fontsize=8, title_fontsize=9, loc='upper left', frameon=True)
plt.tight_layout(rect=[0, 0.04, 1, 1.02])
plt.show()
```

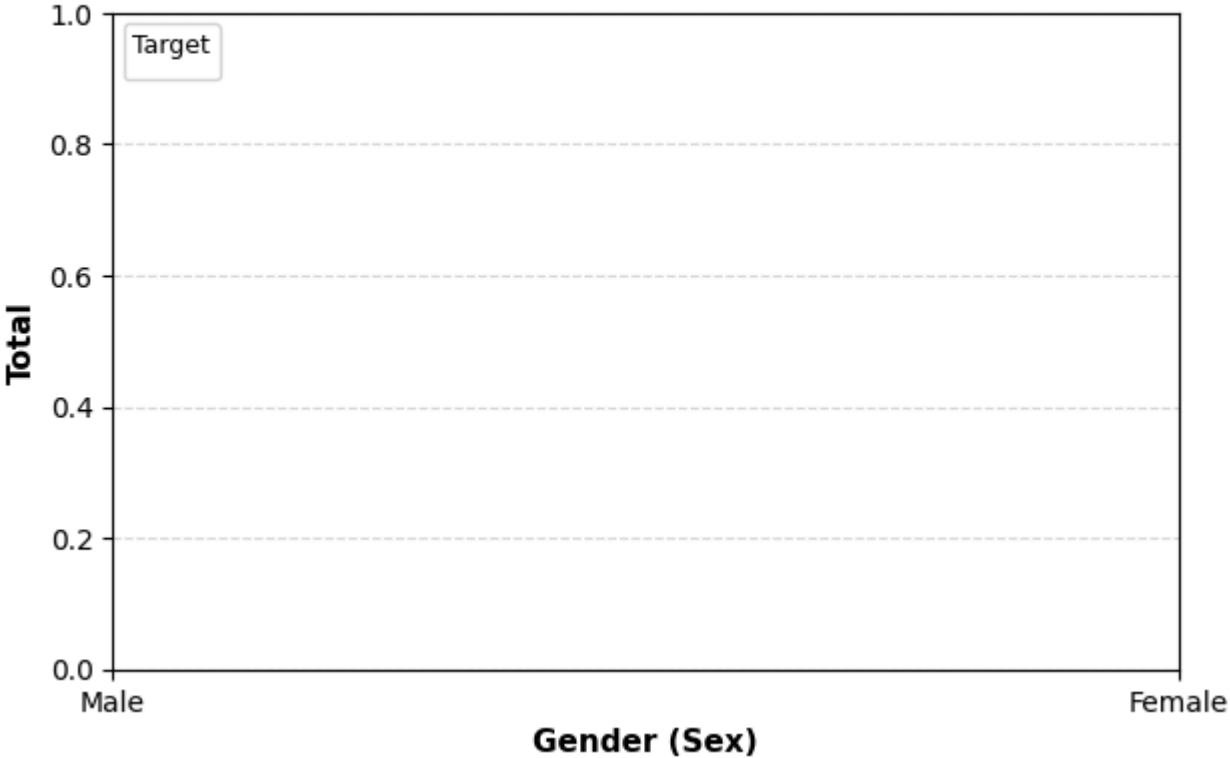
Heart Disease Distribution by Gender

Females tend to have heart disease more evenly distributed.
In males, there's more imbalance.



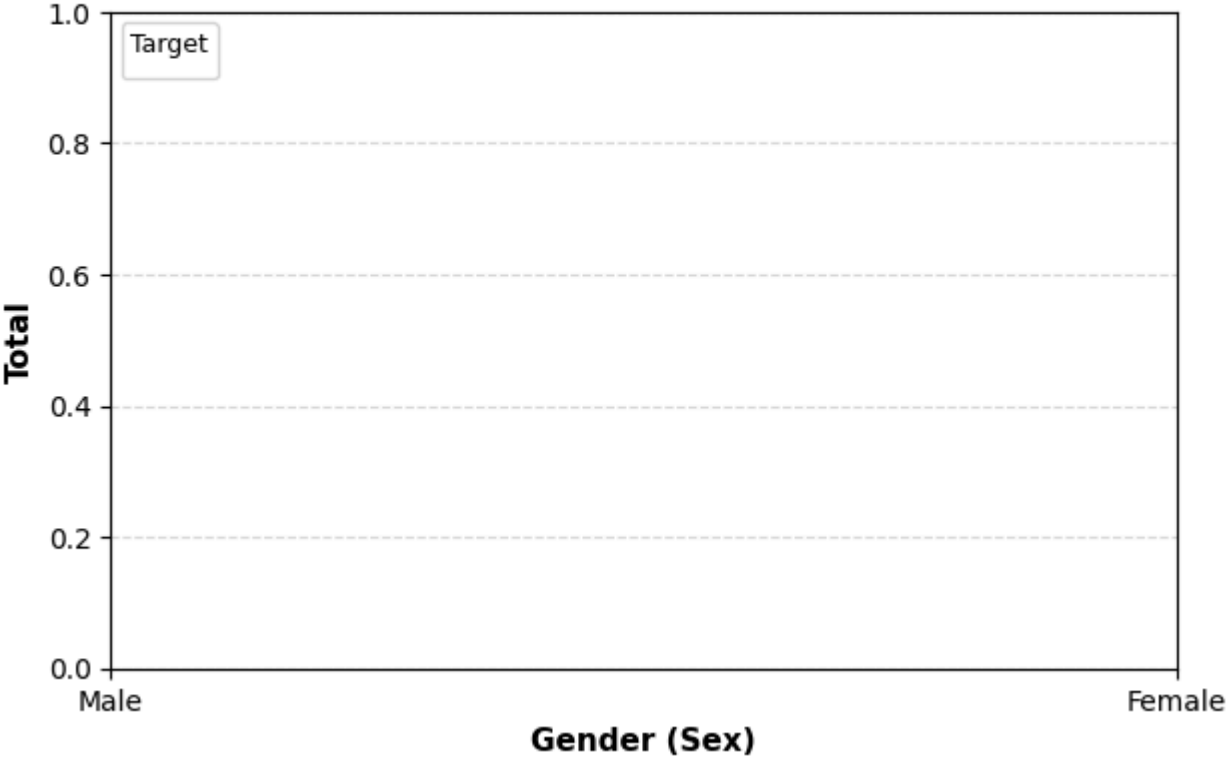
Heart Disease Distribution by Gender

Females tend to have heart disease more evenly distributed.
In males, there's more imbalance.

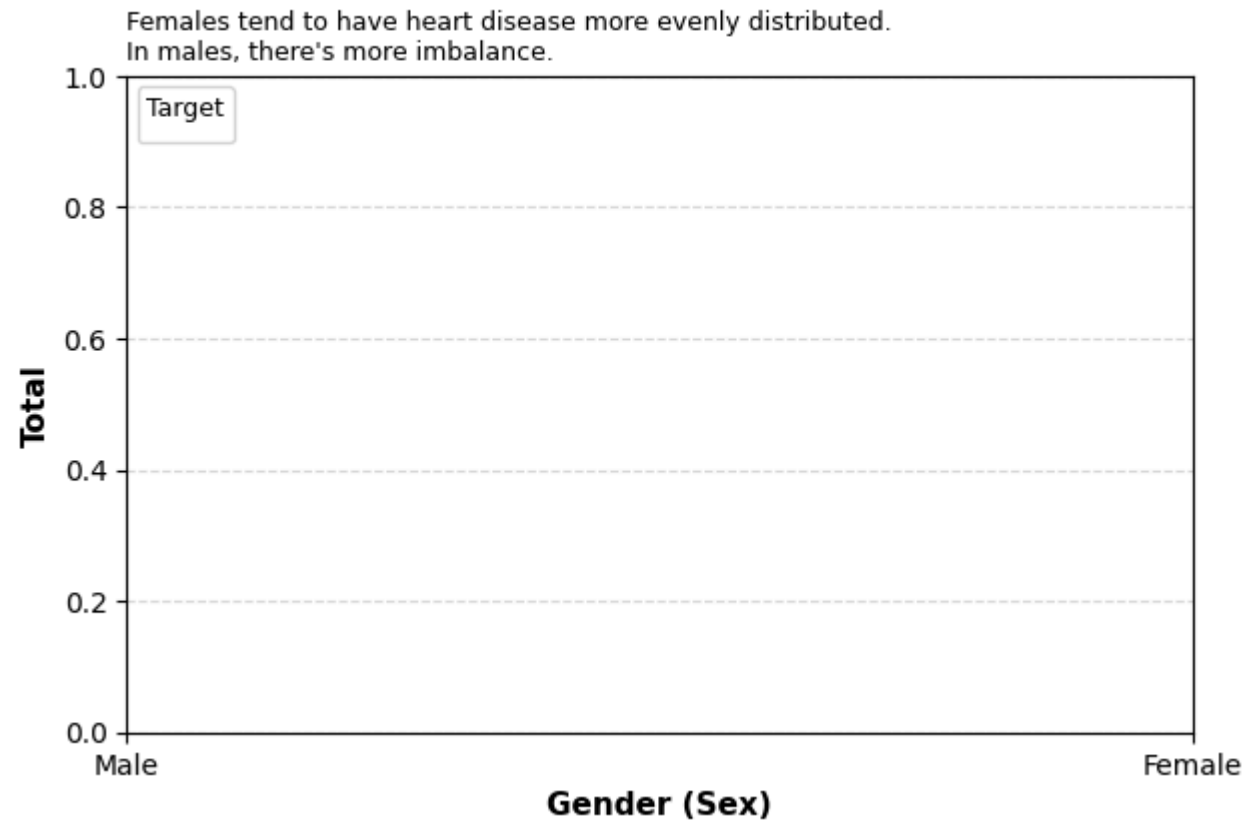


Heart Disease Distribution by Gender

Females tend to have heart disease more evenly distributed.
In males, there's more imbalance.

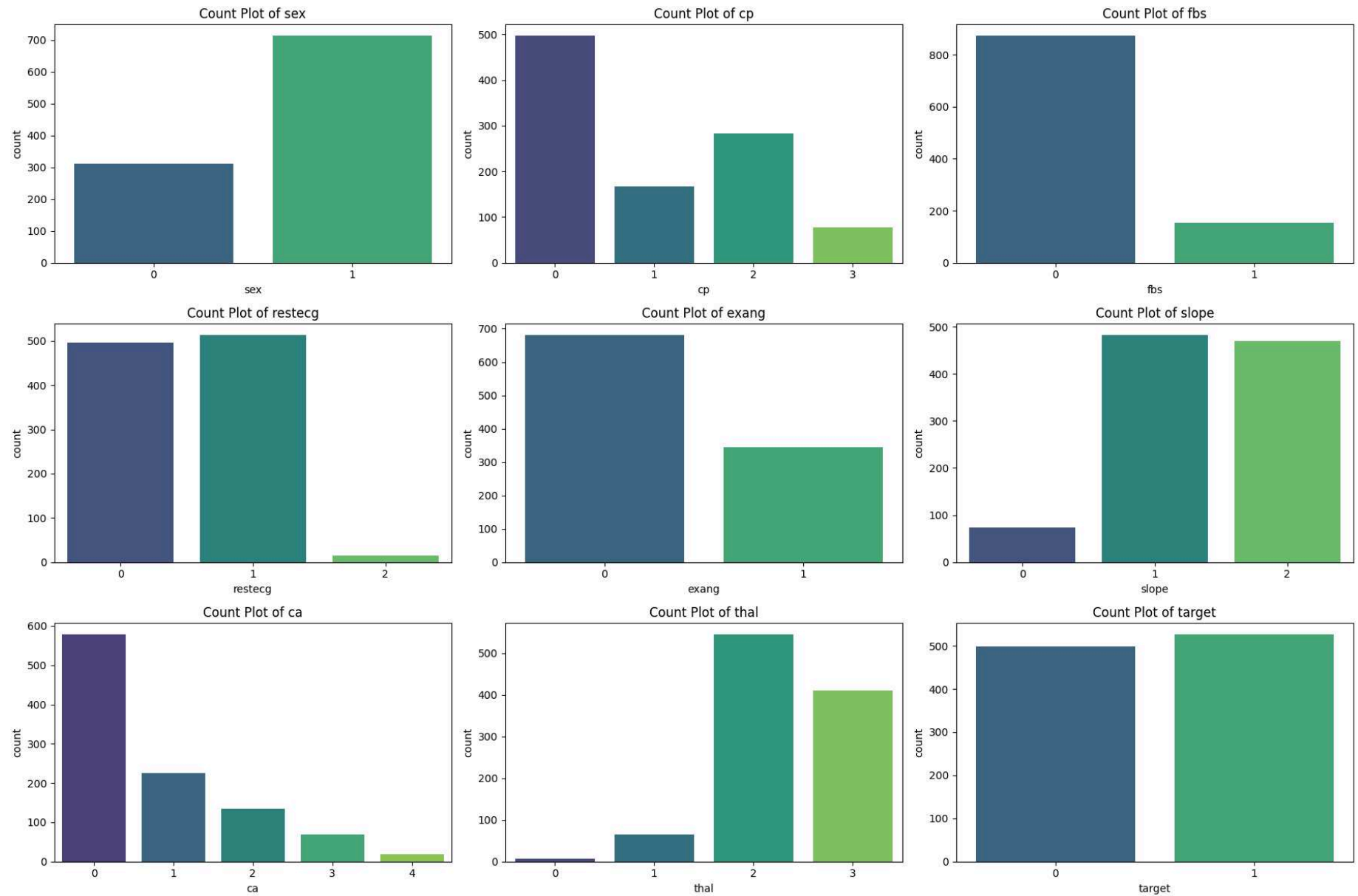


Heart Disease Distribution by Gender



```
In [26]: categorical_cols = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']

plt.figure(figsize=(18, 12))
for i, col in enumerate(categorical_cols):
    plt.subplot(3, 3, i + 1)
    sns.countplot(x=col, data=df, palette='viridis')
    plt.title(f'Count Plot of {col}')
    plt.tight_layout()
plt.show()
```



```
In [27]: # Number of categorical columns
n = len(categorical_cols)
nrows, ncols = (3, 3) # adjust as needed
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(18, 12))
```

```
axes = axes.flatten()

for i, col in enumerate(categorical_cols):
    ct = pd.crosstab(df[col], df['target'], normalize='index')

    # Create a combined pie chart for all categories in the feature
    # Each pie slice = proportion of samples in each category having target=0 or 1
    for idx, category in enumerate(ct.index):
        ax = axes[i]
        sizes = ct.loc[category]
        labels = [f'{label} ({sizes[label]*100:.1f}%)' for label in sizes.index]
        ax.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=plt.cm.Set2.colors)
        ax.set_title(f'{col}: {category}')
        break # Comment this line if you want separate subplots per category

    # Note: The next line is currently unreachable due to the 'break' statement above.
    # If you intend to set a general title for the subplot, it should be outside this inner loop
    # or the 'break' should be removed for multi-category pies per subplot.
    axes[i].set_title(f'Target Distribution for {col}')

# Remove empty axes if there are fewer categorical columns than subplots
for j in range(len(categorical_cols), len(axes)):
    fig.delaxes(axes[j])

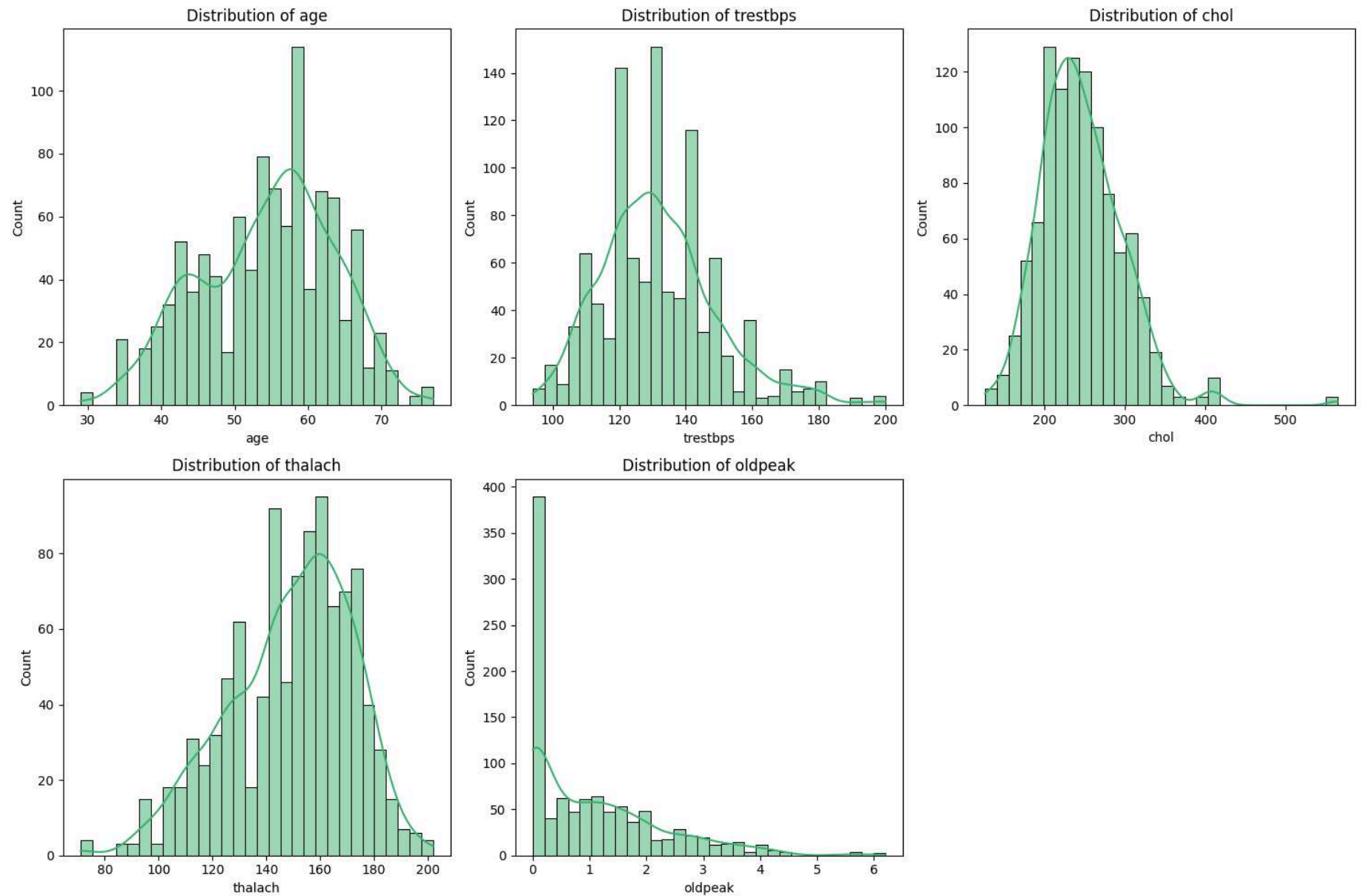
plt.tight_layout()
plt.suptitle('Categorical Features vs Heart Disease (Pie Charts)', fontsize=16, y=1.03)
plt.show()
```

Categorical Features vs Heart Disease (Pie Charts)



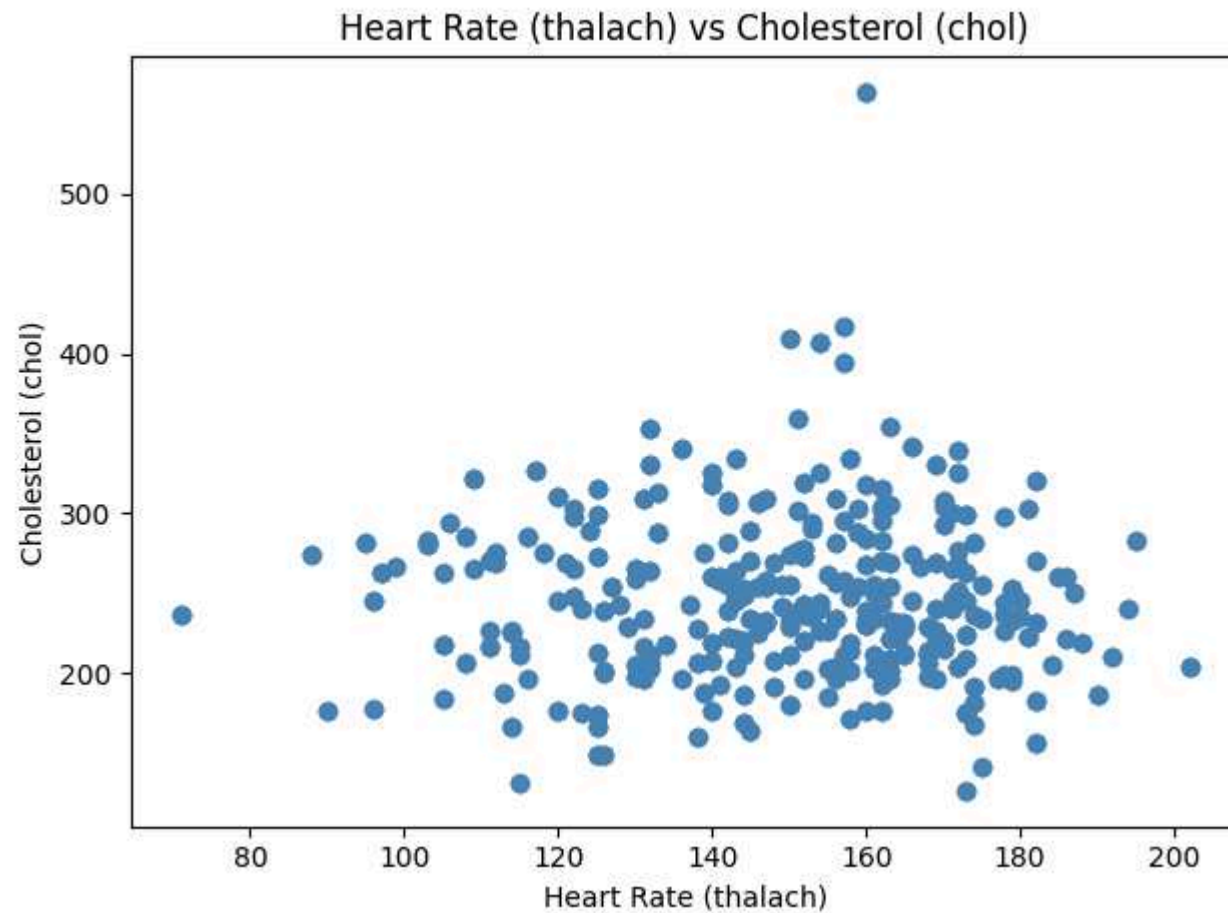
```
In [28]: numerical_cols = [col for col in df.columns if col not in categorical_cols]

plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols):
    plt.subplot(2, 3, i + 1)
    sns.histplot(df[col], kde=True, bins=30, color='mediumseagreen')
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```



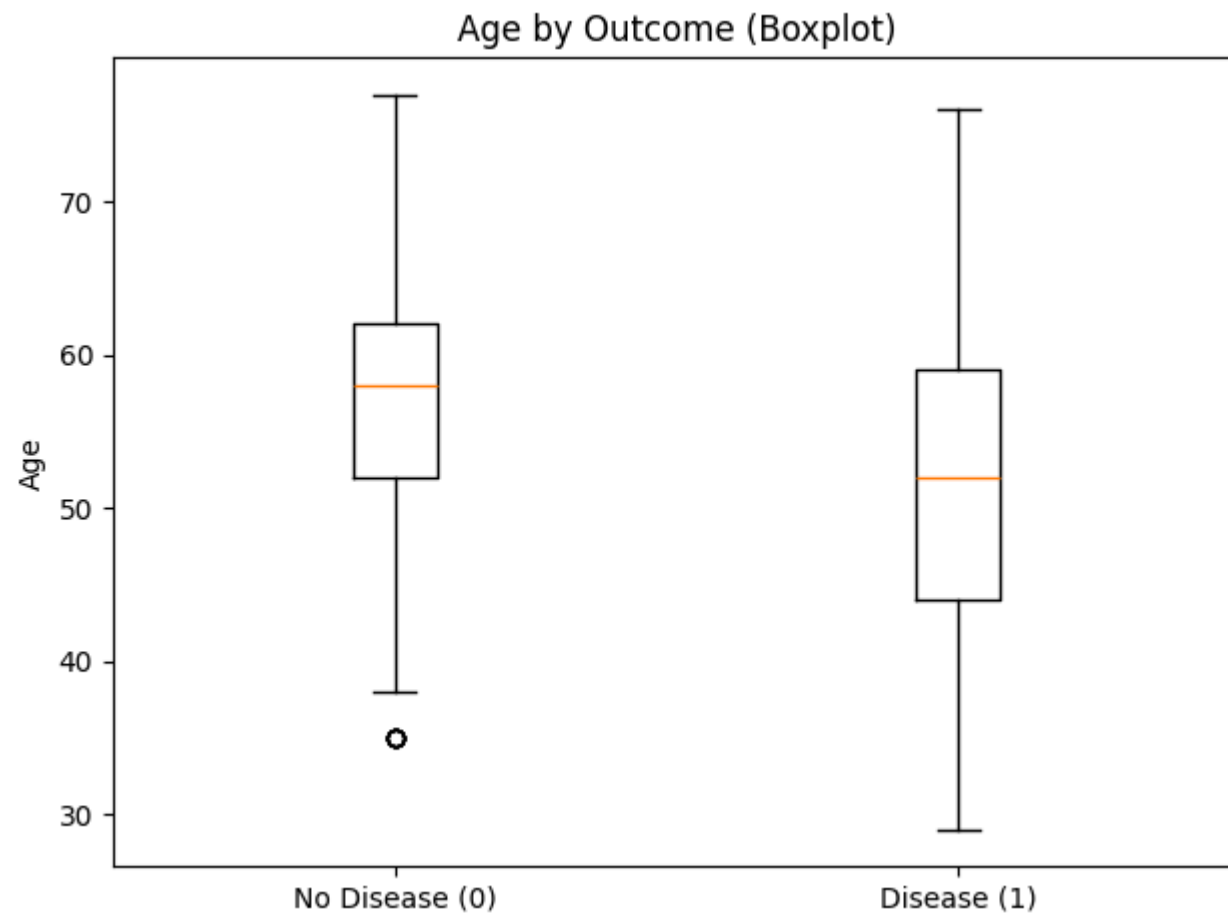
```
In [29]: # Heart Rate (thalach) vs Cholesterol (chol)
plt.figure()
plt.scatter(df["thalach"], df["chol"], color='steelblue', alpha=0.7)
plt.title("Heart Rate (thalach) vs Cholesterol (chol)")
```

```
plt.xlabel("Heart Rate (thalach)")  
plt.ylabel("Cholesterol (chol)")  
plt.tight_layout()  
plt.show()
```



```
In [30]: # 1b. Boxplot of age grouped by target  
plt.figure()  
neg = df[df['target'] == 0]['age']  
pos = df[df['target'] == 1]['age']  
data_for_box = [neg.values, pos.values]  
plt.boxplot(data_for_box, labels=["No Disease (0)", "Disease (1)"])  
plt.title("Age by Outcome (Boxplot)")
```

```
plt.ylabel("Age")  
plt.tight_layout()  
plt.show()
```



```
In [31]: !jupyter nbconvert -- to html /content/drive/MyDrive/Python/heart_proj01.ipynb
```

```
[NbConvertApp] WARNING | pattern 'to' matched no files
[NbConvertApp] WARNING | pattern 'html' matched no files
Traceback (most recent call last):
  File "/usr/local/bin/jupyter-nbconvert", line 10, in <module>
    sys.exit(main())
            ^^^^^^
  File "/usr/local/lib/python3.12/dist-packages/jupyter_core/application.py", line 284, in launch_instance
    super().launch_instance(argv=argv, **kwargs)
  File "/usr/local/lib/python3.12/dist-packages/traitlets/config/application.py", line 992, in launch_instance
    app.start()
  File "/usr/local/lib/python3.12/dist-packages/nbconvert/nbconvertapp.py", line 420, in start
    self.convert_notebooks()
  File "/usr/local/lib/python3.12/dist-packages/nbconvert/nbconvertapp.py", line 582, in convert_notebooks
    raise ValueError(msg)
ValueError: Please specify an output format with '--to <format>'.
The following formats are available: ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'qtpdf',
'qtpng', 'rst', 'script', 'slides', 'webpdf']
```