FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERIG
Department of Computer Engineering

Experiment 2 - Perform Exploratory Data Analysis of Healthcare Data

1. **Course Details:**

Academic Year	**2023 - 24**	**Estimated Time**	**Experiment No. 1 – 02 Hours**
Course & Semester	**B.E. – Sem. VII**	**Subject Name**	**Data Science for Health and Social Care Lab**
Experiment Type	**Software Performance**	**Subject Code**	**HDSSBL701**

Name of Student	Atharva Prashant Pawar	**Roll No.**	9427
Date of Performance.:		**Date of Submission.:**	
CO Mapping	**HDSSBL701.2 Clean, integrate and transform healthcare data**		

Aim: Perform Exploratory Data Analysis of Healthcare Data

Objective: The objective of this experiment is to familiarize BE Computer students with the process of performing Exploratory Data Analysis (EDA) on healthcare data. Students will learn how to import, explore, visualize, and analyze a healthcare dataset to gain insights and understand the characteristics of the data

Tools and libraries:

- Python programming language
- Jupyter Notebook or any Python IDE
- Required Python libraries: pandas, matplotlib, seaborn

Procedure:

Step 1: Data Loading and Understanding

Start by importing the necessary Python libraries: pandas, matplotlib, and seaborn.
Load the healthcare dataset into a pandas DataFrame.
Display the basic information about the dataset, such as the number of rows and columns, data types, and summary statistics

Step 2: Data Cleaning and Preprocessing

Check for any missing values in the dataset and decide how to handle them (e.g., imputation or removal).

Look for any duplicate entries in the dataset and handle them if found.
Convert data types if necessary (e.g., dates, categorical variables).

Step 3: Exploratory Data Analysis (EDA)
Generate summary statistics for relevant numerical variables (e.g., mean, median, standard deviation, etc.).

Visualize the distribution of numerical variables using histograms, box plots, or kernel density plots.
Analyze the distribution of categorical variables using bar plots or count plots
Explore the correlation between different variables using a correlation heatmap.

Step 4: Data Visualization
Create meaningful visualizations to understand relationships and trends in the data. Use scatter plots, bar plots, line plots, or any other appropriate visualization techniques.
Focus on specific aspects like the relationship between age and health conditions, gender-wise distribution of diseases etc.

Step 5: Insights and Interpretation
Based on the analysis and visualizations, derive insights and observations from the healthcare data.
Identify patterns, trends, and any interesting findings that can help in understanding the dataset better.

Result:

Data Science in Health Care : Group - 5

Team : Atharva Pawar (9427), Aditya Vyas, Harshvardhan Trivedi

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
url = "https://raw.githubusercontent.com/Aditya-K-Vyas/data_science_utils/main/heart.csv"
df = pd.read_csv(url)
```

In [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
df.describe()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.00
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512	1.385366	0.75
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.03
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.00
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000	1.000000	0.00
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000	1.000000	0.00
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.00
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.00

In [6]:
```
duplicates = df.duplicated()
print(duplicates)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
1020     True
1021     True
1022     True
1023     True
1024     True
Length: 1025, dtype: bool
```

In [7]:
```
# removing duplicate data
df = df.drop_duplicates()
```

In [8]:
```
df.shape
```

Out[8]:
```
(302, 14)
```

In [9]:
```
# checking for null values
null_counts = df.isnull().sum()
print(null_counts)
```

```
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]:
```
# final df look
df.head()
```

Out[10]:

	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

EDA

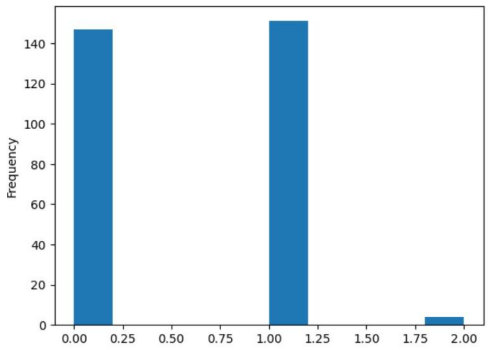
In [12]:
```
# basic df
df.describe()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	2.314
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748	0.613
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	2.000
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000	2.000
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000

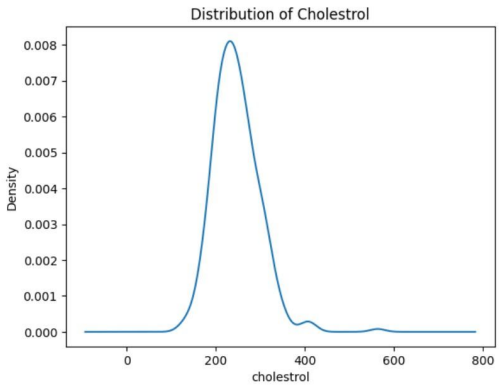
In [13]:
```
# histogram
df['restecg'].plot.hist()  # Histogram
```

Out[13]:
```
<Axes: ylabel='Frequency'>
```

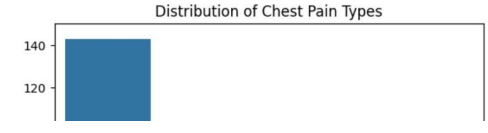


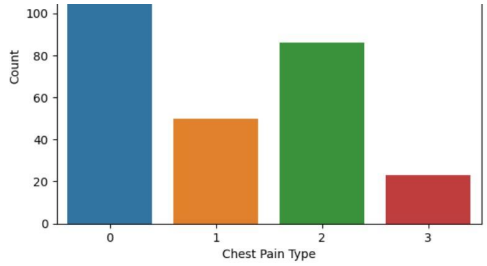
In [14]:
```
df['chol'].plot.kde()  # Kernel density plot
plt.xlabel('cholestrol')
plt.ylabel('Density')
plt.title('Distribution of Cholestrol')
```

Out[14]:
```
Text(0.5, 1.0, 'Distribution of Cholestrol')
```

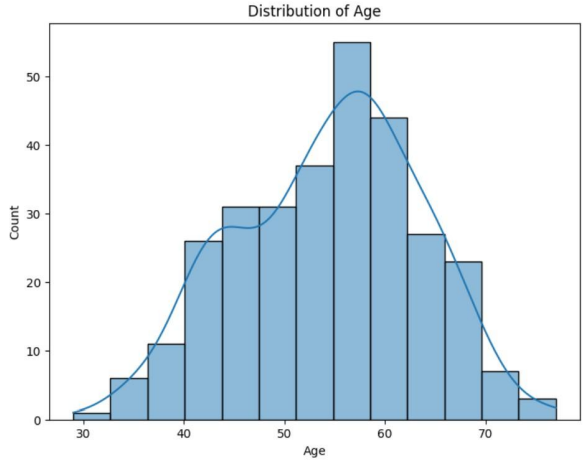


In [15]:
```
sns.countplot(data=df, x='cp')
plt.xlabel('Chest Pain Type')
plt.ylabel('Count')
plt.title('Distribution of Chest Pain Types')
plt.show()
```

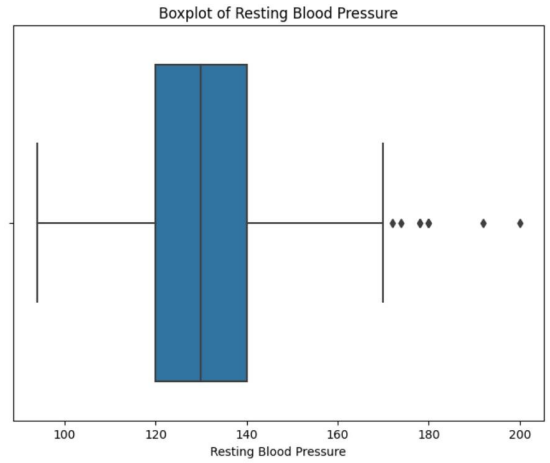




In [16]:
```
plt.figure(figsize=(8,6))
sns.histplot(data=df, x='age', kde=True)
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Distribution of Age')
plt.show()
```



In [17]:
```
plt.figure(figsize=(8,6))
sns.boxplot(data=df, x='trestbps')
plt.xlabel('Resting Blood Pressure')
plt.title('Boxplot of Resting Blood Pressure')
plt.show()
```



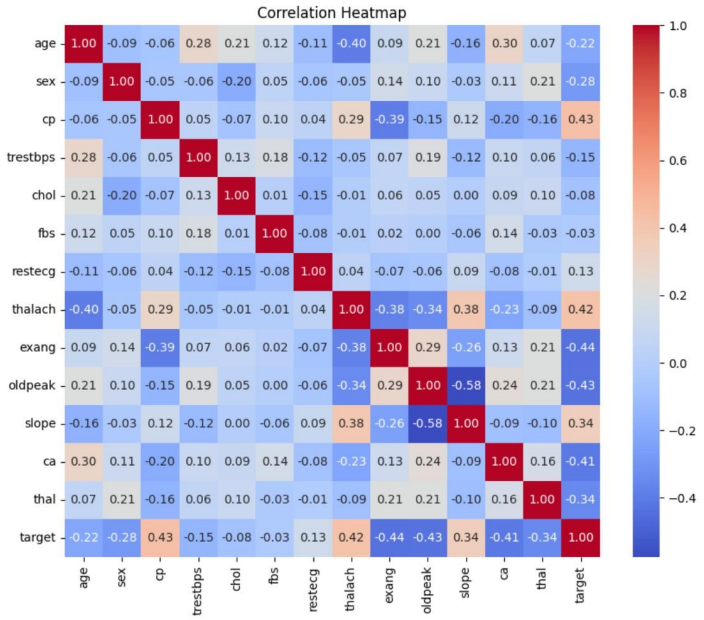
In [18]:
```
# Explore the distribution of target classes
print("\nTarget Class Distribution:")
print(df['target'].value_counts())
```

```
Target Class Distribution:
1    164
0    138
Name: target, dtype: int64
```

In [19]:
```
# Correlation matrix
print("\nCorrelation Matrix:")
correlation_matrix = df.corr()
print(correlation_matrix)
```

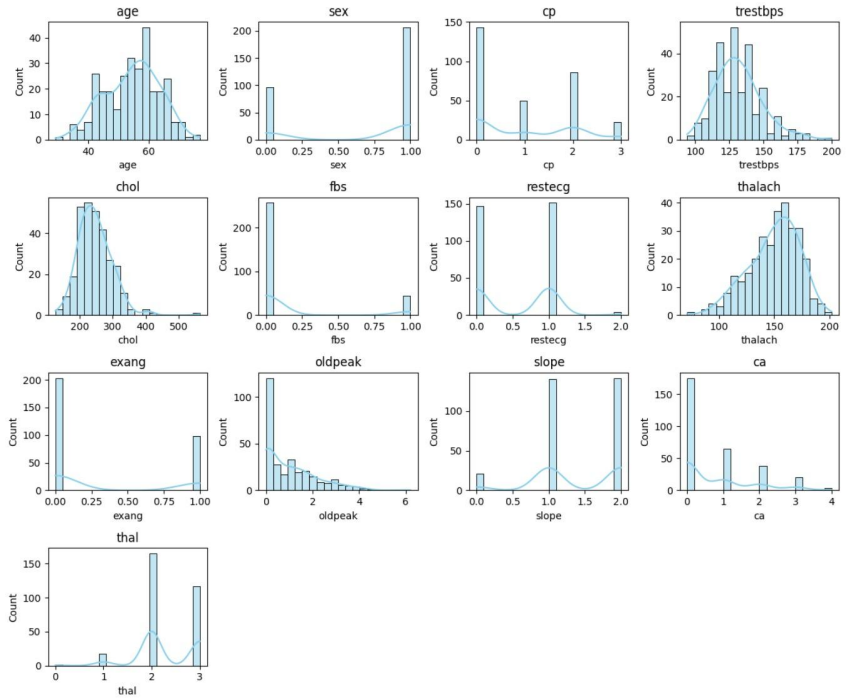
```
Correlation Matrix:
                age       sex        cp  trestbps      chol       fbs  \
age        1.000000 -0.094962 -0.063107  0.283121  0.207216  0.119492
sex       -0.094962  1.000000 -0.051740 -0.057647 -0.195571  0.046022
cp        -0.063107 -0.051740  1.000000  0.046486 -0.072682  0.096018
trestbps   0.283121 -0.057647  0.046486  1.000000  0.125256  0.178125
chol       0.207216 -0.195571 -0.072682  0.125256  1.000000  0.011428
fbs        0.119492  0.046022  0.096018  0.178125  0.011428  1.000000
restecg   -0.111590 -0.060351  0.041561 -0.115367 -0.147602 -0.083081
thalach   -0.395235 -0.046439  0.293367 -0.048023 -0.005308 -0.007169
exang      0.093216  0.143460 -0.392937  0.068526  0.064099  0.024729
oldpeak    0.206040  0.098322 -0.146692  0.194600  0.050086  0.004514
slope     -0.164124 -0.032990  0.116854 -0.122873  0.000417 -0.058654
ca         0.302261  0.113060 -0.195356  0.099248  0.086878  0.144935
thal       0.065317  0.211452 -0.160370  0.062870  0.096810 -0.032752
target    -0.221476 -0.283609  0.432080 -0.146269 -0.081437 -0.026826

             restecg   thalach     exang   oldpeak     slope        ca  \
age       -0.111590 -0.395235  0.093216  0.206040 -0.164124  0.302261
sex       -0.060351 -0.046439  0.143460  0.098322 -0.032990  0.113060
cp         0.041561  0.293367 -0.392937 -0.146692  0.116854 -0.195356
trestbps  -0.115367 -0.048023  0.068526  0.194600 -0.122873  0.099248
chol      -0.147602 -0.005308  0.064099  0.050086  0.000417  0.086878
fbs       -0.083081 -0.007169  0.024729  0.004514 -0.058654  0.144935
restecg    1.000000  0.041210 -0.068807 -0.056251  0.090402 -0.083112
thalach    0.041210  1.000000 -0.377411 -0.342201  0.384754 -0.228311
exang     -0.068807 -0.377411  1.000000  0.286766 -0.256106  0.125377
oldpeak   -0.056251 -0.342201  0.286766  1.000000 -0.576314  0.236560
slope      0.090402  0.384754 -0.256106 -0.576314  1.000000 -0.092236
ca        -0.083112 -0.228311  0.125377  0.236560 -0.092236  1.000000
thal      -0.010473 -0.094910  0.205826  0.209090 -0.103314  0.160085
target     0.134874  0.419955 -0.435601 -0.429146  0.343940 -0.408992

              thal    target
age       0.065317 -0.221476
sex       0.211452 -0.283609
cp       -0.160370  0.432080
trestbps  0.062870 -0.146269
chol      0.096810 -0.081437
fbs      -0.032752 -0.026826
restecg  -0.010473  0.134874
thalach  -0.094910  0.419955
exang     0.205826 -0.435601
oldpeak   0.209090 -0.429146
slope    -0.103314  0.343940
ca        0.160085 -0.408992
thal      1.000000 -0.343101
target   -0.343101  1.000000
```

In [20]:
```
# Heatmap of correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

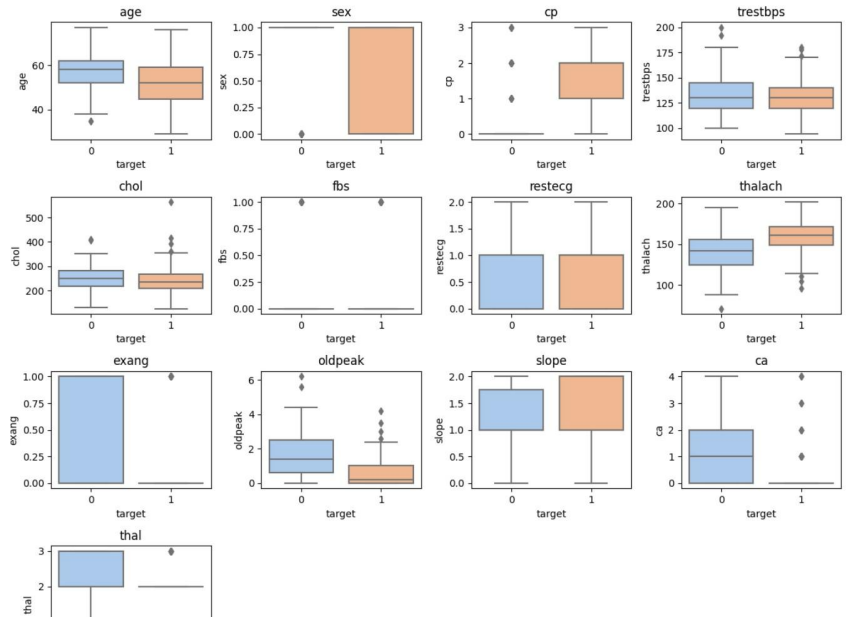


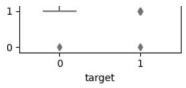
In [21]:
```
# Distribution plots for numerical features
```

```
plt.figure(figsize=(12, 10))
for i, column in enumerate(df.columns[:-1]):
    plt.subplot(4, 4, i + 1)
    sns.histplot(df[column], kde=True, bins=20, color='skyblue')
    plt.title(column)
plt.tight_layout()
plt.show()
```

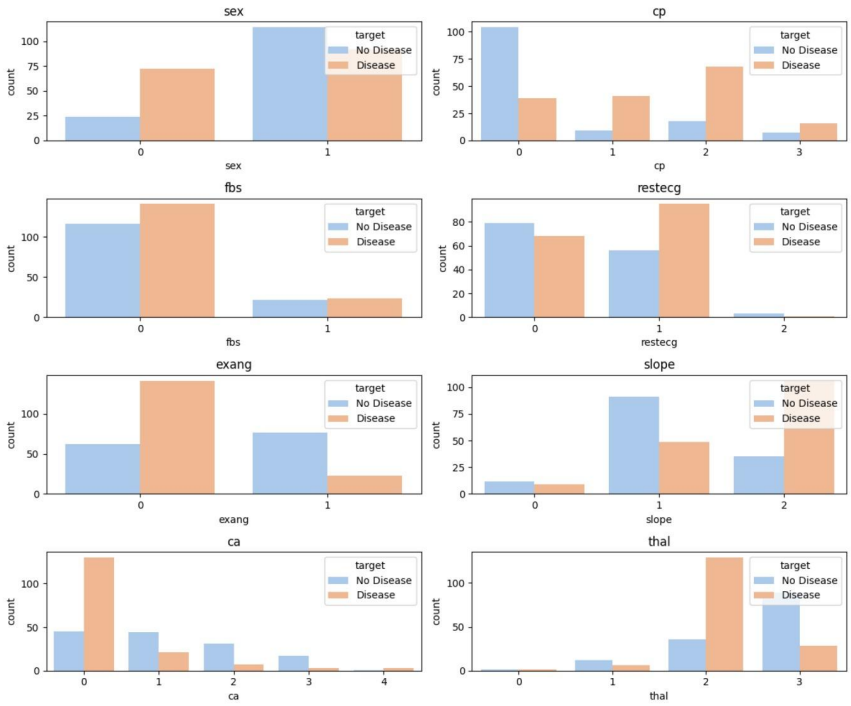


In [22]:
```
# Box plots for numerical features by target class
plt.figure(figsize=(12, 10))
for i, column in enumerate(df.columns[:-1]):
    plt.subplot(4, 4, i + 1)
    sns.boxplot(x='target', y=column, data=df, palette='pastel')
    plt.title(column)
plt.tight_layout()
plt.show()
```

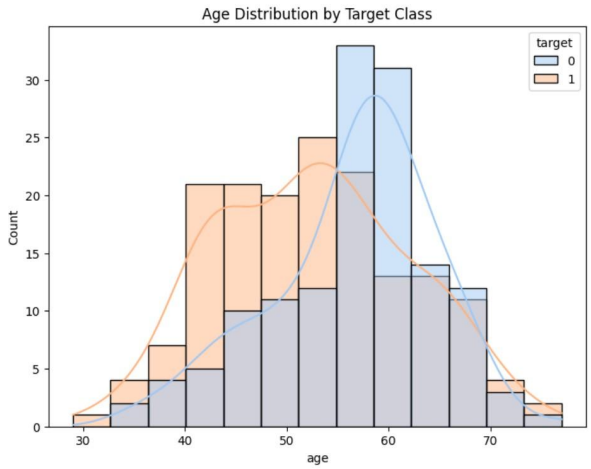




In [23]:
```
# Count plots for categorical features by target class
plt.figure(figsize=(12, 10))
for i, column in enumerate(['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']):
    plt.subplot(4, 2, i + 1)
    sns.countplot(x=column, hue='target', data=df, palette='pastel')
    plt.title(column)
    plt.legend(title='target', loc='upper right', labels=['No Disease', 'Disease'])
plt.tight_layout()
plt.show()
```



In [29]:
```
# Age distribution by target class
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='age', hue='target', kde=True, palette='pastel')
plt.title("Age Distribution by Target Class")
plt.show()
```



In [30]:
```
# Scatter plots for age vs. thalach and age vs. trestbps
plt.figure(figsize=(12, 5))
```

```
plt.subplot(1, 2, 1)
sns.scatterplot(x='age', y='thalach', hue='target', data=df, palette='pastel')
plt.title("Age vs. Thalach")

plt.subplot(1, 2, 2)
sns.scatterplot(x='age', y='trestbps', hue='target', data=df, palette='pastel')
plt.title("Age vs. Trestbps")
plt.tight_layout()
plt.show()
```