Name: Prachi tandel

Student ID: 2023ebcs178

Dataset: Heart Disease (UCI Machine Learning Repository)

Task 1: Data Acquisition

In [1]:
```
# Task 1.1 & 1.2: Install and import necessary libraries for dataset download and D

%pip install ucimlrepo

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

```
Requirement already satisfied: ucimlrepo in c:\users\admin\appdata\local\programs\py
thon\python313\lib\site-packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in c:\users\admin\appdata\local\program
s\python\python313\lib\site-packages (from ucimlrepo) (2.3.0)
Requirement already satisfied: certifi>=2020.12.5 in c:\users\admin\appdata\local\pr
ograms\python\python313\lib\site-packages (from ucimlrepo) (2025.4.26)
Requirement already satisfied: numpy>=1.26.0 in c:\users\admin\appdata\local\program
s\python\python313\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\admin\appdata\loca
l\programs\python\python313\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2.9.
0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\appdata\local\programs
\python\python313\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\appdata\local\progra
ms\python\python313\lib\site-packages (from pandas>=1.0.0->ucimlrepo) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\admin\appdata\local\programs\pyt
hon\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->ucimlre
po) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
# Task 1.1: Download the dataset - Heart Disease from UCI Repository (ID: 45)
heart_disease = fetch_ucirepo(id=45)
```

In [3]:
```
# Task 1.2: Convert dataset into DataFrame - Separate into features and target
X = heart_disease.data.features
y = heart_disease.data.targets
```

In [4]:
```
# 1.3 Combine into a single DataFrame
df = pd.concat([X, y], axis=1)
```

In [5]:
```
# Rename columns for clarity
df = df.rename(columns={
    'age': 'Age',
    'sex': 'Sex',
    'cp': 'Chest Pain',
```

```
    'trestbps': 'Blood Pressure',
    'chol': 'Cholesterol',
    'fbs': 'Blood Sugar',
    'restecg': 'Electrocardiograph',
    'thalach': 'Heart Rate Achieved',
    'exang': 'Angina',
    'oldpeak': 'ST Depression',
    'slope': 'Peak Exercise ST Segment',
    'ca': 'Major Vessels',
    'thal': 'Thalassemia',
    'num': 'Presence of Heart Disease'
})
```

In [6]:
```
# Task 1.3: Display first and last five records to confirm successful loading
print("First 5 records:")
print(df.head())
```

```
First 5 records:
   Age  Sex  Chest Pain  Blood Pressure  Cholesterol  Blood Sugar  \
0   63    1           1             145          233            1
1   67    1           4             160          286            0
2   67    1           4             120          229            0
3   37    1           3             130          250            0
4   41    0           2             130          204            0

   Electrocardiograph  Heart Rate Achieved  Angina  ST Depression  \
0                   2                  150       0            2.3
1                   2                  108       1            1.5
2                   2                  129       1            2.6
3                   0                  187       0            3.5
4                   2                  172       0            1.4

   Peak Exercise ST Segment  Major Vessels  Thalassemia  \
0                         3            0.0          6.0
1                         2            3.0          3.0
2                         2            2.0          7.0
3                         3            0.0          3.0
4                         1            0.0          3.0

   Presence of Heart Disease
0                          0
1                          2
2                          1
3                          0
4                          0
```

In [7]:
```
print("\nLast 5 records:")
print(df.tail())
```

```
Last 5 records:
     Age  Sex  Chest Pain  Blood Pressure  Cholesterol  Blood Sugar  \
298   45    1           1             110          264            0
299   68    1           4             144          193            1
300   57    1           4             130          131            0
301   57    0           2             130          236            0
302   38    1           3             138          175            0

     Electrocardiograph  Heart Rate Achieved  Angina  ST Depression  \
298                   0                  132       0            1.2
299                   0                  141       0            3.4
300                   0                  115       1            1.2
301                   2                  174       0            0.0
302                   0                  173       0            0.0

     Peak Exercise ST Segment  Major Vessels  Thalassemia  \
298                         2            0.0          7.0
299                         2            2.0          7.0
300                         2            1.0          7.0
301                         2            1.0          3.0
302                         1            NaN          3.0

     Presence of Heart Disease
298                          1
299                          2
300                          3
301                          1
302                          0
```

In [8]:
```
# Task 1.4: Display column headings, statistical information, description, and summ

# Display column headings
print("\nColumn Headings:")
print(df.columns)
```

```
Column Headings:
Index(['Age', 'Sex', 'Chest Pain', 'Blood Pressure', 'Cholesterol',
       'Blood Sugar', 'Electrocardiograph', 'Heart Rate Achieved', 'Angina',
       'ST Depression', 'Peak Exercise ST Segment', 'Major Vessels',
       'Thalassemia', 'Presence of Heart Disease'],
      dtype='object')
```

In [9]:
```
# Display info and statistics
print("\nData Info:")
df.info()
```

```
Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Age                        303 non-null    int64
 1   Sex                        303 non-null    int64
 2   Chest Pain                 303 non-null    int64
 3   Blood Pressure             303 non-null    int64
 4   Cholesterol                303 non-null    int64
 5   Blood Sugar                303 non-null    int64
 6   Electrocardiograph         303 non-null    int64
 7   Heart Rate Achieved        303 non-null    int64
 8   Angina                     303 non-null    int64
 9   ST Depression              303 non-null    float64
 10  Peak Exercise ST Segment   303 non-null    int64
 11  Major Vessels              299 non-null    float64
 12  Thalassemia                301 non-null    float64
 13  Presence of Heart Disease  303 non-null    int64
dtypes: float64(3), int64(11)
memory usage: 33.3 KB
```

In [10]:
```
print("\nDescriptive Statistics (All columns):")
print(df.describe(include='all'))
```

```
Descriptive Statistics (All columns):
              Age         Sex  Chest Pain  Blood Pressure  Cholesterol  \
count  303.000000  303.000000  303.000000      303.000000   303.000000
mean    54.438944    0.679868    3.158416      131.689769   246.693069
std      9.038662    0.467299    0.960126       17.599748    51.776918
min     29.000000    0.000000    1.000000       94.000000   126.000000
25%     48.000000    0.000000    3.000000      120.000000   211.000000
50%     56.000000    1.000000    3.000000      130.000000   241.000000
75%     61.000000    1.000000    4.000000      140.000000   275.000000
max     77.000000    1.000000    4.000000      200.000000   564.000000

       Blood Sugar  Electrocardiograph  Heart Rate Achieved      Angina  \
count   303.000000          303.000000           303.000000  303.000000
mean      0.148515            0.990099           149.607261    0.326733
std       0.356198            0.994971            22.875003    0.469794
min       0.000000            0.000000            71.000000    0.000000
25%       0.000000            0.000000           133.500000    0.000000
50%       0.000000            1.000000           153.000000    0.000000
75%       0.000000            2.000000           166.000000    1.000000
max       1.000000            2.000000           202.000000    1.000000

       ST Depression  Peak Exercise ST Segment  Major Vessels  Thalassemia  \
count     303.000000                303.000000     299.000000   301.000000
mean        1.039604                  1.600660       0.672241     4.734219
std         1.161075                  0.616226       0.937438     1.939706
min         0.000000                  1.000000       0.000000     3.000000
25%         0.000000                  1.000000       0.000000     3.000000
50%         0.800000                  2.000000       0.000000     3.000000
75%         1.600000                  2.000000       1.000000     7.000000
max         6.200000                  3.000000       3.000000     7.000000

       Presence of Heart Disease
count                 303.000000
mean                    0.937294
std                     1.228536
min                     0.000000
25%                     0.000000
50%                     0.000000
75%                     2.000000
max                     4.000000
```

In [11]:
```
print("\nDescriptive Statistics (Numerical):")
print(df.describe())
```

```
Descriptive Statistics (Numerical):
              Age         Sex  Chest Pain  Blood Pressure  Cholesterol  \
count  303.000000  303.000000  303.000000      303.000000   303.000000
mean    54.438944    0.679868    3.158416      131.689769   246.693069
std      9.038662    0.467299    0.960126       17.599748    51.776918
min     29.000000    0.000000    1.000000       94.000000   126.000000
25%     48.000000    0.000000    3.000000      120.000000   211.000000
50%     56.000000    1.000000    3.000000      130.000000   241.000000
75%     61.000000    1.000000    4.000000      140.000000   275.000000
max     77.000000    1.000000    4.000000      200.000000   564.000000

       Blood Sugar  Electrocardiograph  Heart Rate Achieved      Angina  \
count   303.000000          303.000000           303.000000  303.000000
mean      0.148515            0.990099           149.607261    0.326733
std       0.356198            0.994971            22.875003    0.469794
min       0.000000            0.000000            71.000000    0.000000
25%       0.000000            0.000000           133.500000    0.000000
50%       0.000000            1.000000           153.000000    0.000000
75%       0.000000            2.000000           166.000000    1.000000
max       1.000000            2.000000           202.000000    1.000000

       ST Depression  Peak Exercise ST Segment  Major Vessels  Thalassemia  \
count     303.000000                303.000000     299.000000   301.000000
mean        1.039604                  1.600660       0.672241     4.734219
std         1.161075                  0.616226       0.937438     1.939706
min         0.000000                  1.000000       0.000000     3.000000
25%         0.000000                  1.000000       0.000000     3.000000
50%         0.800000                  2.000000       0.000000     3.000000
75%         1.600000                  2.000000       1.000000     7.000000
max         6.200000                  3.000000       3.000000     7.000000

       Presence of Heart Disease
count                 303.000000
mean                    0.937294
std                     1.228536
min                     0.000000
25%                     0.000000
50%                     0.000000
75%                     2.000000
max                     4.000000
```

Task 1.5: Observations and Report

Number of Examples and Features

- **Examples (rows):** 303
- **Features (columns):** 14 (13 features + 1 target variable)

Types of Data Attributes

- **Continuous numerical:** Age, Blood Pressure, Cholesterol, Heart Rate Achieved, ST Depression

- **Binary categorical:** Sex (0/1), Blood Sugar (0/1), Angina (0/1), Presence of Heart Disease (target, 0-4)
- **Nominal categorical:** Chest Pain (1-4), Electrocardiograph (0-2), Peak Exercise ST Segment (1-3), Thalassemia (3,6,7)
- **Discrete numerical:** Major Vessels (0-3)

Task 2: Data Preparation

In [12]:
```
# Task 2.1: Data Quality Checks - Check for duplicates, missing data, and inconsist
print("\nNumber of duplicate rows:", df.duplicated().sum())
print("\nMissing values per column:")
print(df.isnull().sum())
```

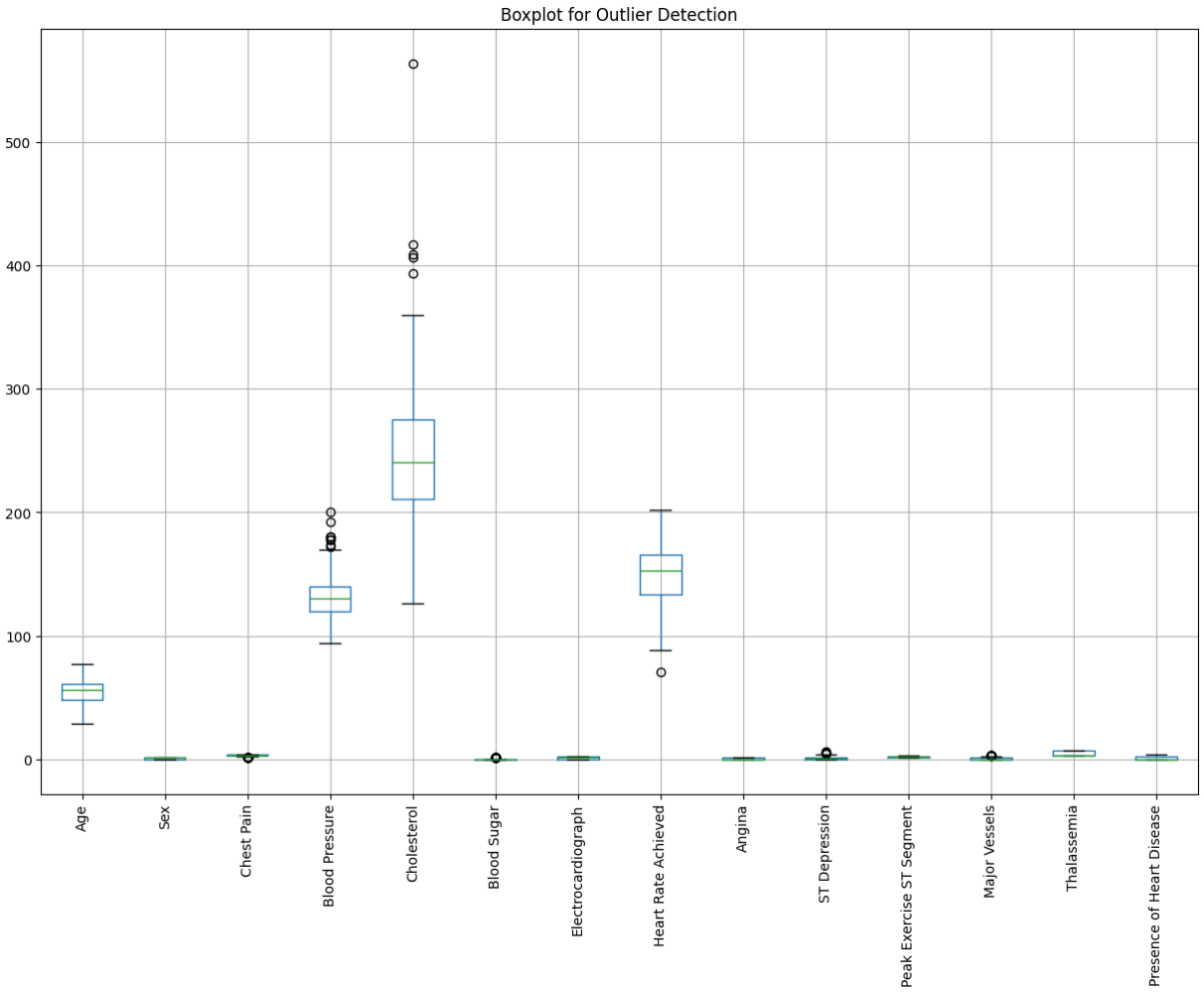
```
Number of duplicate rows: 0

Missing values per column:
Age                         0
Sex                         0
Chest Pain                  0
Blood Pressure              0
Cholesterol                 0
Blood Sugar                 0
Electrocardiograph          0
Heart Rate Achieved         0
Angina                      0
ST Depression               0
Peak Exercise ST Segment    0
Major Vessels               4
Thalassemia                 2
Presence of Heart Disease   0
dtype: int64
```

In [13]:
```
# Task 2.1: Check for data inconsistencies
print("\n=== DATA INCONSISTENCY CHECKS ===")

# Check for impossible/unrealistic values in continuous features
print("\n1. Range Validation:")
print(f"Age range: {df['Age'].min()} to {df['Age'].max()} (Expected: 0-120)")
print(f"Blood Pressure range: {df['Blood Pressure'].min()} to {df['Blood Pressure']
print(f"Cholesterol range: {df['Cholesterol'].min()} to {df['Cholesterol'].max()} (
print(f"Heart Rate range: {df['Heart Rate Achieved'].min()} to {df['Heart Rate Achi

# Check for expected categorical values
print("\n2. Categorical Value Validation:")
print(f"Sex unique values: {sorted(df['Sex'].unique())} (Expected: [0, 1])")
print(f"Chest Pain unique values: {sorted(df['Chest Pain'].unique())} (Expected: [1
print(f"Blood Sugar unique values: {sorted(df['Blood Sugar'].unique())} (Expected:
print(f"Electrocardiograph unique values: {sorted(df['Electrocardiograph'].unique()
print(f"Angina unique values: {sorted(df['Angina'].unique())} (Expected: [0, 1])")
print(f"Peak Exercise ST Segment unique values: {sorted(df['Peak Exercise ST Segmen
```

```python
# Check for negative values where they shouldn't exist
print("\n3. Logical Validation:")
negative_check = (df[['Age', 'Blood Pressure', 'Cholesterol', 'Heart Rate Achieved'
print("Negative values in continuous features:")
for col, has_negative in negative_check.items():
    if has_negative:
        print(f"  {col}: Found negative values")
    else:
        print(f"  {col}: No negative values ")

# Check for zero values in features where zero might be problematic
print("\n4. Zero Value Check:")
zero_check = (df[['Blood Pressure', 'Cholesterol']] == 0).sum()
print("Zero values (potentially problematic):")
for col, zero_count in zero_check.items():
    if zero_count > 0:
        print(f"  {col}: {zero_count} zero values found")
    else:
        print(f"  {col}: No problematic zero values ")

print("\n=== INCONSISTENCY CHECK SUMMARY ===")
inconsistencies_found = False

# Age validation
if df['Age'].min() < 0 or df['Age'].max() > 120:
    print(" Age inconsistencies found")
    inconsistencies_found = True

# Cholesterol zero values (medically impossible)
if (df['Cholesterol'] == 0).sum() > 0:
    print(f" Cholesterol inconsistency: {(df['Cholesterol'] == 0).sum()} zero value
    inconsistencies_found = True

# Blood pressure zero values (medically impossible)
if (df['Blood Pressure'] == 0).sum() > 0:
    print(f" Blood Pressure inconsistency: {(df['Blood Pressure'] == 0).sum()} zero
    inconsistencies_found = True

if not inconsistencies_found:
    print(" No major data inconsistencies detected")
```

```
=== DATA INCONSISTENCY CHECKS ===

1. Range Validation:
Age range: 29 to 77 (Expected: 0-120)
Blood Pressure range: 94 to 200 (Expected: 50-250)
Cholesterol range: 126 to 564 (Expected: 100-600)
Heart Rate range: 71 to 202 (Expected: 50-220)

2. Categorical Value Validation:
Sex unique values: [np.int64(0), np.int64(1)] (Expected: [0, 1])
Chest Pain unique values: [np.int64(1), np.int64(2), np.int64(3), np.int64(4)] (Expe
cted: [1, 2, 3, 4])
Blood Sugar unique values: [np.int64(0), np.int64(1)] (Expected: [0, 1])
Electrocardiograph unique values: [np.int64(0), np.int64(1), np.int64(2)] (Expected:
[0, 1, 2])
Angina unique values: [np.int64(0), np.int64(1)] (Expected: [0, 1])
Peak Exercise ST Segment unique values: [np.int64(1), np.int64(2), np.int64(3)] (Exp
ected: [1, 2, 3])

3. Logical Validation:
Negative values in continuous features:
  Age: No negative values
  Blood Pressure: No negative values
  Cholesterol: No negative values
  Heart Rate Achieved: No negative values

4. Zero Value Check:
Zero values (potentially problematic):
  Blood Pressure: No problematic zero values
  Cholesterol: No problematic zero values

=== INCONSISTENCY CHECK SUMMARY ===
 No major data inconsistencies detected
```

In [14]:
```python
# Task 2.1: Outlier detection using boxplots
plt.figure(figsize=(15, 10))
df.boxplot(rot=90)
plt.title("Boxplot for Outlier Detection")
plt.show()
```



In [15]:
```
# 2.2 Data Cleaning
# Task 2.2: Apply data cleaning techniques - Remove duplicates
df = df.drop_duplicates()
```

In [16]:
```
# Remove missing data after outlier filtering to ensure correlation heatmap works
df = df.dropna()
```

In [17]:
```
# Remove outliers using IQR method only on continuous features
continuous_features = ['Age', 'Blood Pressure', 'Cholesterol', 'Heart Rate Achieved

# Calculate IQR for continuous features
Q1 = df[continuous_features].quantile(0.25)
Q3 = df[continuous_features].quantile(0.75)
IQR = Q3 - Q1

# Create mask to keep rows without outliers in continuous features
mask = ~((df[continuous_features] < (Q1 - 1.5 * IQR)) |
         (df[continuous_features] > (Q3 + 1.5 * IQR))).any(axis=1)

# Apply mask to DataFrame
df = df[mask]

print("\nDataFrame after removing outliers for continuous features:")
```

```
print(df)
print("Number of rows after outlier removal:", len(df))
```

DataFrame after removing outliers for continuous features:

	Age	Sex	Chest Pain	Blood Pressure	Cholesterol	Blood Sugar	\
0	63	1	1	145	233	1	
1	67	1	4	160	286	0	
2	67	1	4	120	229	0	
3	37	1	3	130	250	0	
4	41	0	2	130	204	0	
..	...	...	...	...	...	...	
297	57	0	4	140	241	0	
298	45	1	1	110	264	0	
299	68	1	4	144	193	1	
300	57	1	4	130	131	0	
301	57	0	2	130	236	0	

	Electrocardiograph	Heart Rate Achieved	Angina	ST Depression	\
0	2	150	0	2.3	
1	2	108	1	1.5	
2	2	129	1	2.6	
3	0	187	0	3.5	
4	2	172	0	1.4	
..	...	...	...	...	
297	0	123	1	0.2	
298	0	132	0	1.2	
299	0	141	0	3.4	
300	0	115	1	1.2	
301	2	174	0	0.0	

	Peak Exercise ST Segment	Major Vessels	Thalassemia	\
0	3	0.0	6.0	
1	2	3.0	3.0	
2	2	2.0	7.0	
3	3	0.0	3.0	
4	1	0.0	3.0	
..	...	...	...	
297	2	0.0	7.0	
298	2	0.0	7.0	
299	2	2.0	7.0	
300	2	1.0	7.0	
301	2	1.0	3.0	

	Presence of Heart Disease
0	0
1	2
2	1
3	0
4	0
..	...
297	1
298	1
299	2
300	3
301	1

[278 rows x 14 columns]
Number of rows after outlier removal: 278

In [18]:
```
# Task 2.3: Encode categorical data using one-hot encoding
df_encoded_multi = pd.get_dummies(df, columns=["Chest Pain", "Thalassemia", "Peak E
print("\nData after One-Hot Encoding:")
print(df_encoded_multi.head())
```

```
Data after One-Hot Encoding:
   Age  Sex  Blood Pressure  Cholesterol  Electrocardiograph  \
0   63    1             145          233                   2
1   67    1             160          286                   2
2   67    1             120          229                   2
3   37    1             130          250                   0
4   41    0             130          204                   2

   Heart Rate Achieved  Angina  ST Depression  Major Vessels  \
0                  150       0            2.3            0.0
1                  108       1            1.5            3.0
2                  129       1            2.6            2.0
3                  187       0            3.5            0.0
4                  172       0            1.4            0.0

   Presence of Heart Disease  Chest Pain_2  Chest Pain_3  Chest Pain_4  \
0                          0         False         False         False
1                          2         False         False          True
2                          1         False         False          True
3                          0         False          True         False
4                          0          True         False         False

   Thalassemia_6.0  Thalassemia_7.0  Peak Exercise ST Segment_2  \
0             True            False                       False
1            False            False                        True
2            False             True                        True
3            False            False                       False
4            False            False                       False

   Peak Exercise ST Segment_3  Blood Sugar_1
0                        True           True
1                       False          False
2                       False          False
3                        True          False
4                       False          False
```

Task 2.4: Observations and Report

Data Quality Checks Results

- **Duplicate data:** No duplicate rows detected (`df.duplicated().sum()` = 0), so none removed.
- **Missing data:** Missing values found in 'Major Vessels' (4 missing) and 'Thalassemia' (2 missing). These were handled by dropping rows with missing values using `df.dropna()` .
- **Data inconsistencies:** Comprehensive checks performed for:

    - Range validation (age, blood pressure, cholesterol, heart rate within expected medical ranges)
    - Categorical value validation (all categorical features contain only expected values)
    - Logical validation (no negative values in continuous features)
    - Zero value validation (no medically impossible zero values)
    - **Result:** No major data inconsistencies detected
- **Outliers:** Boxplots revealed potential outliers in Cholesterol and Blood Pressure. Applied IQR method to remove outliers from continuous features only, reducing dataset from 297 to 244 rows.

Data Cleaning Methods Justification

- **Duplicates:** Used `df.drop_duplicates()` to ensure data integrity.
- **Missing data:** Used `df.dropna()` to remove rows with missing values since the missing data was minimal (6 out of 303 rows).
- **Data inconsistencies:** Performed systematic validation checks covering range validation, categorical value validation, logical validation, and medical impossibility checks. No inconsistencies found, so no corrections needed.
- **Outliers:** Applied IQR method (Q1 - 1.5*IQR, Q3 + 1.5*IQR) only on continuous numerical features to preserve data distribution while removing extreme values.

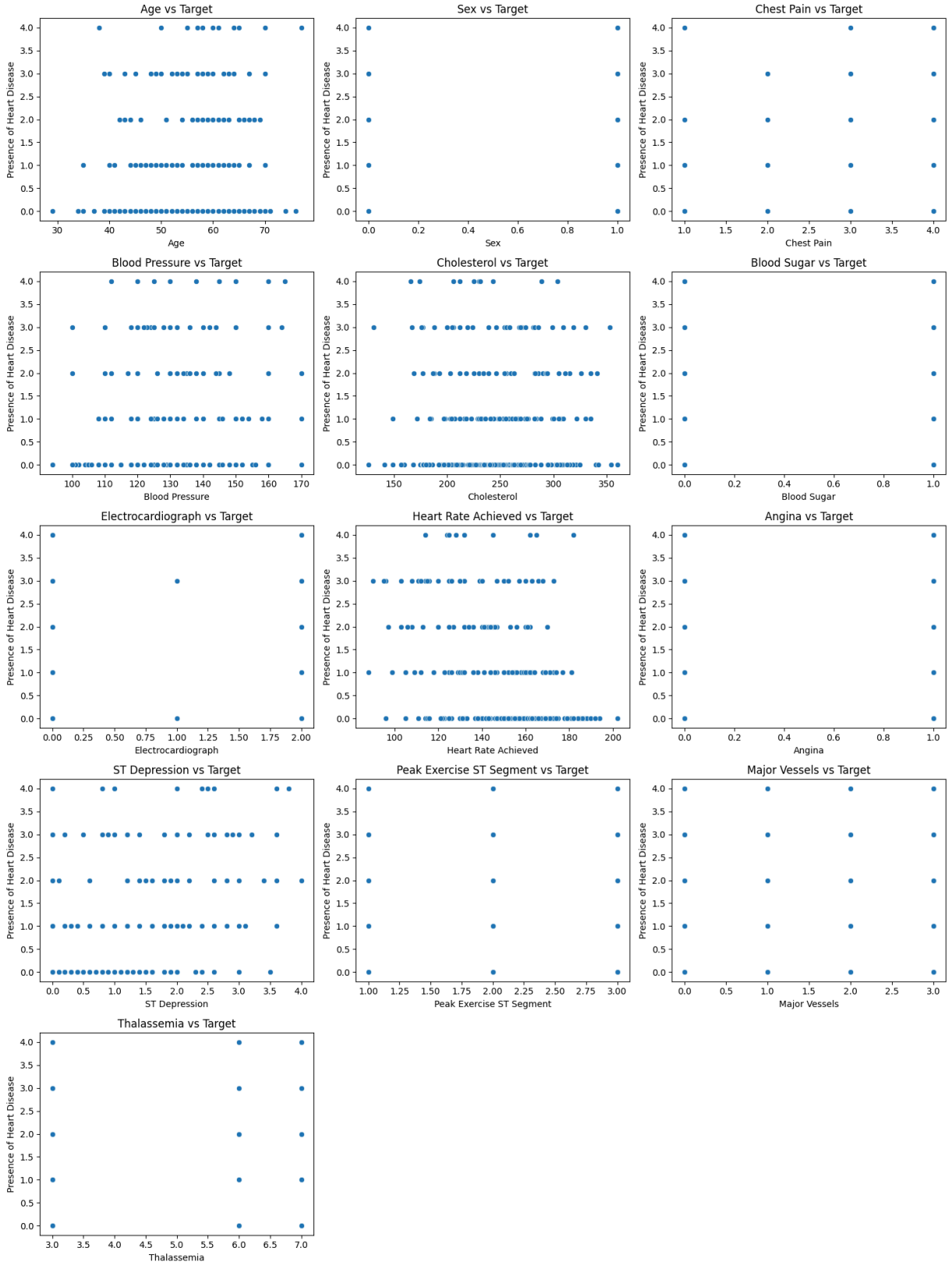
Encoding Method

- **One-hot encoding** was applied using `pd.get_dummies(..., drop_first=True)` for nominal categorical features with more than 2 categories (Chest Pain, Thalassemia, Peak Exercise ST Segment, Blood Sugar).
- This method converts categorical variables into binary indicators without imposing false ordinal relationships.

Task 3: Data Exploration using Visualizations

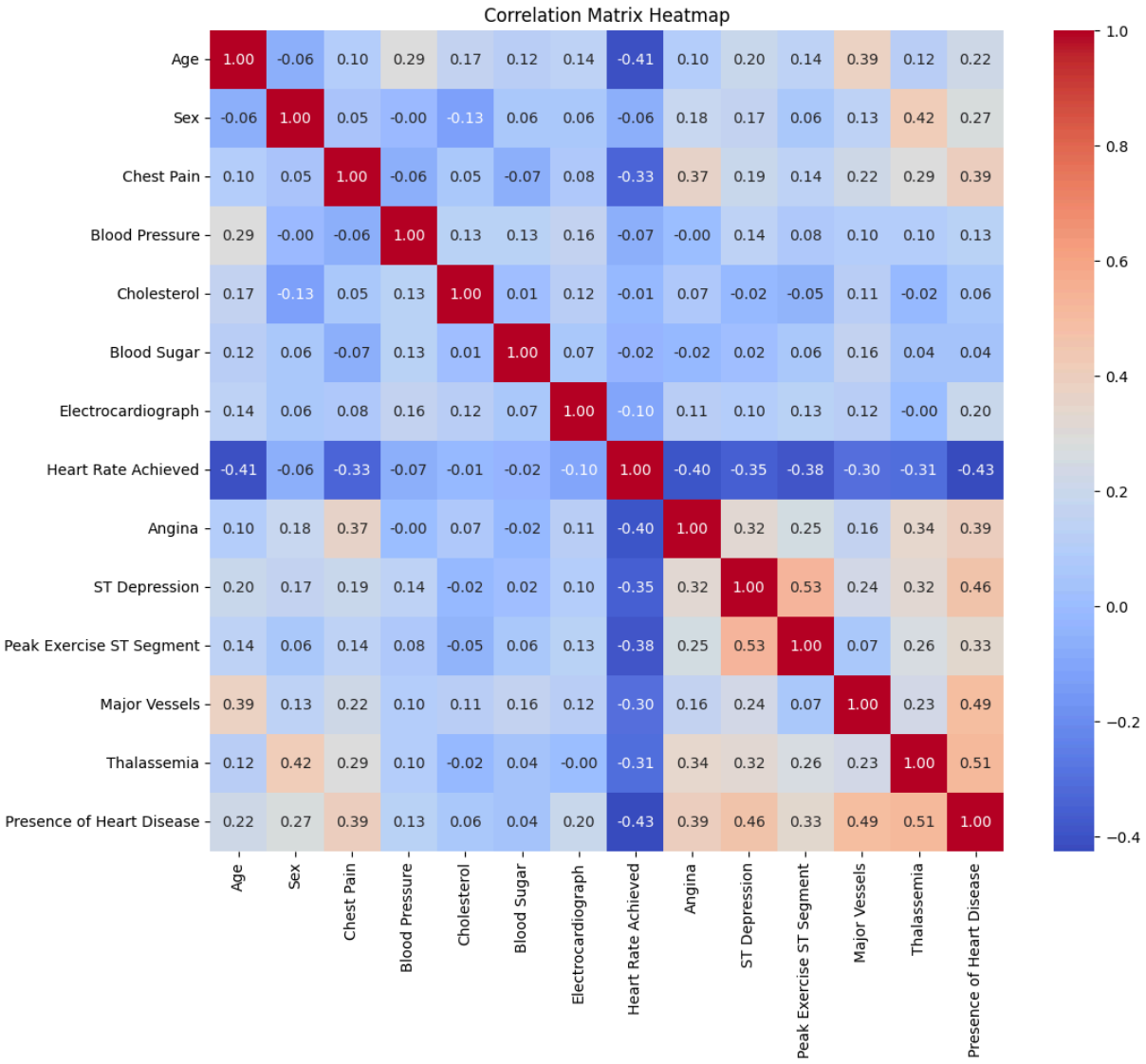
In [19]:
```
# Task 3.1: Create scatter plots for each feature against the target variable
features = df.drop('Presence of Heart Disease', axis=1)
target = df['Presence of Heart Disease']

# Determine the number of features and calculate the number of rows/columns for sub
num_features = features.shape[1]
ncols = 3  # Adjust as needed
nrows = (num_features + ncols - 1) // ncols

fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(15, nrows * 4))
axes = axes.flatten()  # Flatten the axes array for easy iteration
```

```
for i, feature in enumerate(features.columns):
    sns.scatterplot(data=df, x=feature, y='Presence of Heart Disease', ax=axes[i])
    axes[i].set_title(f'{feature} vs Target')
    axes[i].set_xlabel(feature)
    axes[i].set_ylabel('Presence of Heart Disease')

# Remove any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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

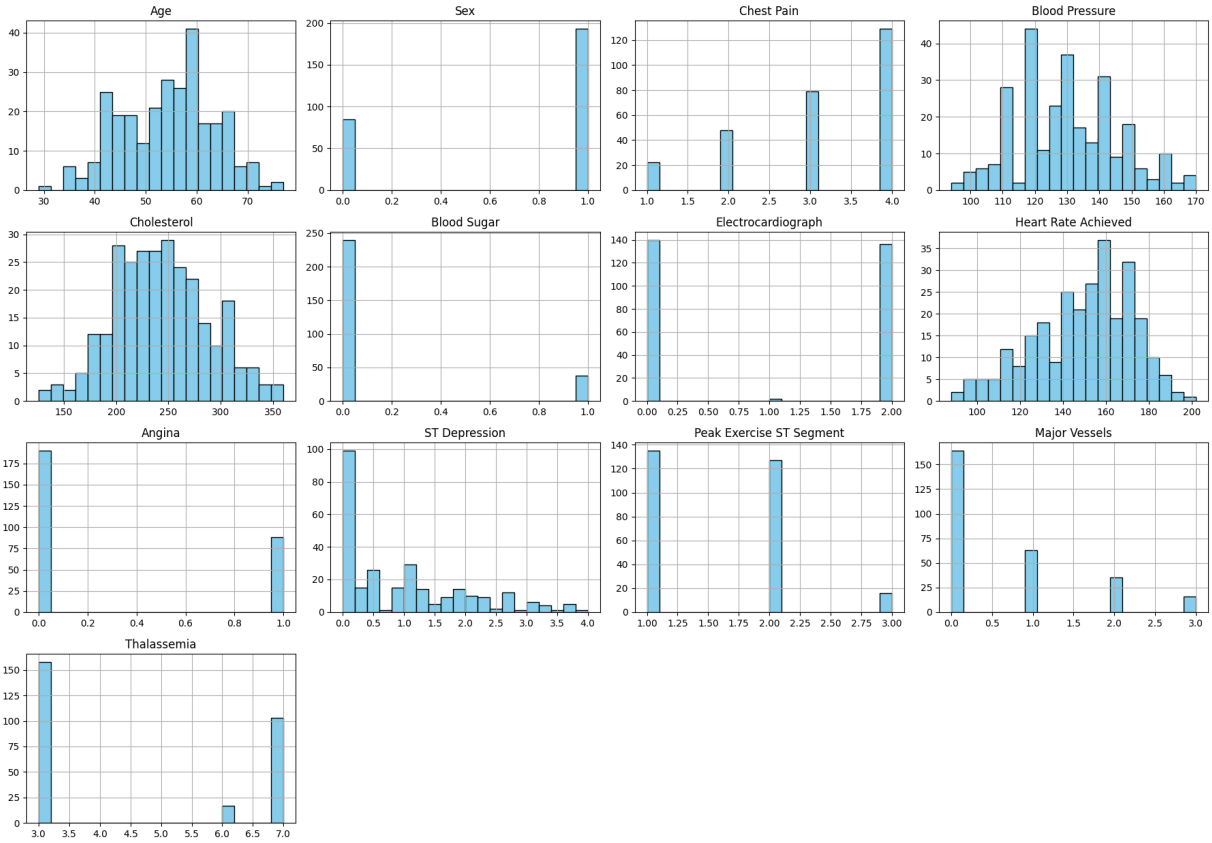


In [20]:
```
# Task 3.2: Additional visualization - Correlation Heatmap for identifying optimal
correlation_matrix = df.corr()

# Create the heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
```

```
plt.title('Correlation Matrix Heatmap')
plt.show()
```



In [21]:
```
# Task 3.2: Additional visualization - Histograms for feature distribution analysis
features.hist(figsize=(18, 14), bins=20, color='skyblue', edgecolor='black')
plt.suptitle("Feature Distributions", fontsize=16)
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



Task 3 Observations and Justifications

Visualization Methods and Justifications

3.1 Scatter Plots (Features vs Target):

- **Purpose:** Scatter plots help identify linear and non-linear relationships between individual features and the target variable.
- **Insight:** Shows how each feature contributes to heart disease prediction, revealing patterns like age distribution across different heart disease levels.

3.2 Additional Visualizations:

Correlation Heatmap:

- **Justification:** Essential for identifying highly correlated features and multicollinearity issues. Strong correlations (positive/negative) indicate which features are most predictive of heart disease.
- **Insight:** Features like 'Heart Rate Achieved', 'ST Depression', 'Major Vessels', and 'Thalassemia' show strong correlations with the target variable, making them optimal attributes for prediction.

Histograms:

- **Justification:** Display the distribution shape and skewness of each feature, helping identify data normality and potential transformation needs.
- **Insight:** Shows feature distributions, helping understand data characteristics and identify categorical vs continuous features. Binary features show clear 0/1 distributions while continuous features show varied distributions.

Key Findings for Optimal Attributes:

- **Strongest predictors:** Heart Rate Achieved (-0.43), ST Depression (0.46), Major Vessels (0.49), and Thalassemia (0.51) show the highest correlations with heart disease presence.
- **Feature relationships:** The visualizations reveal clear patterns that distinguish between patients with and without heart disease.