In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_excel('project2_Data.xlsx')
```

In [3]:
```
data.head(5)
```

Out[3]:

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

In [4]:
```
data.shape
```

Out[4]:
```
(303, 14)
```

In [5]:
```
data.isna().sum()
```

Out[5]:
```
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 [6]:
```
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   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [7]:
```
dup_data = data[data.duplicated(keep='first')]
```

In [8]:
```
dup_data
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

In [9]:
```
data.drop_duplicates(inplace=True)
```

In [10]:
```
data.shape
```

Out[10]:
```
(302, 14)
```

In [11]:
```
data.describe()
```

Out[11]:

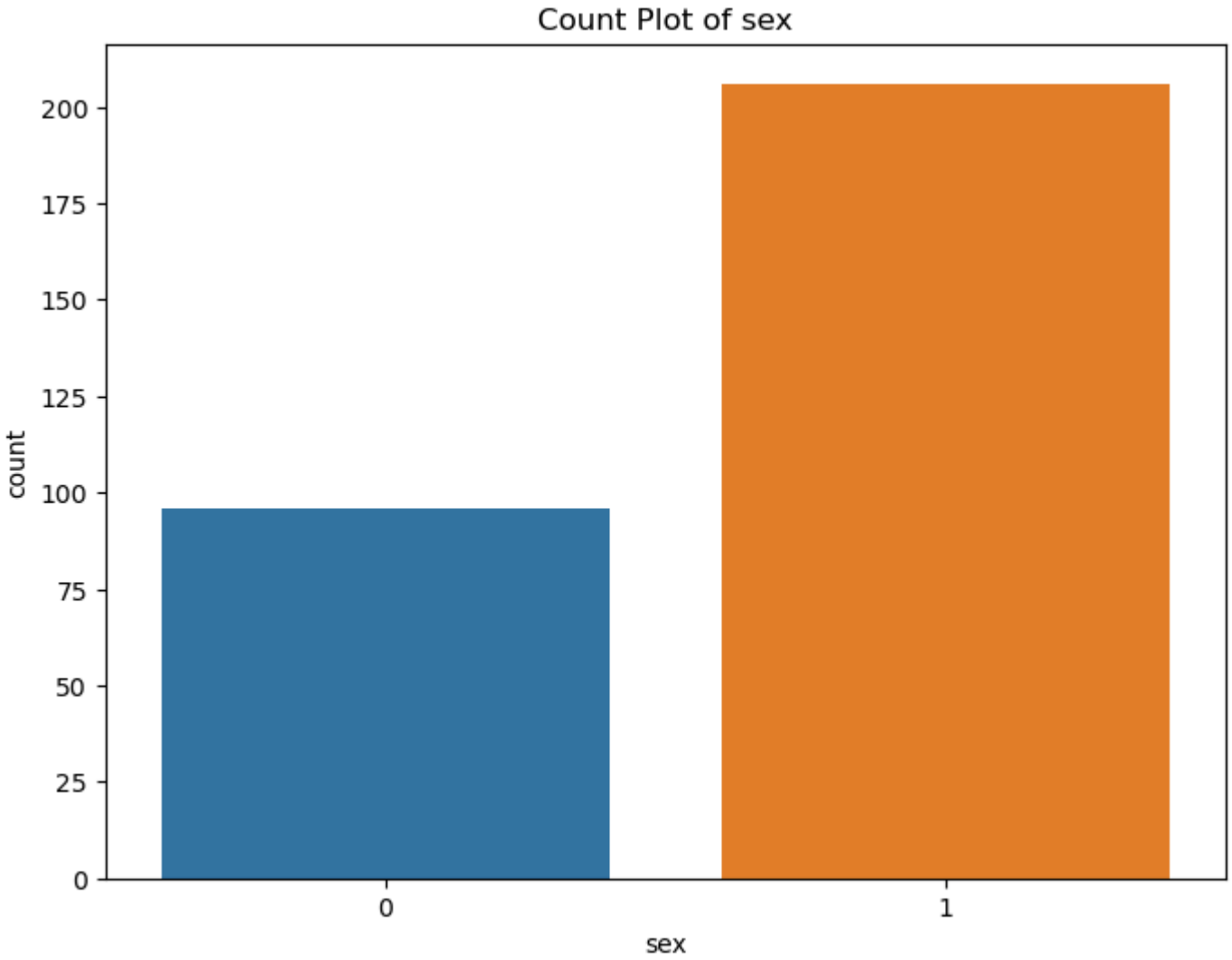
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	3(
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543	
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	
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	
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	
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	
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	
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	

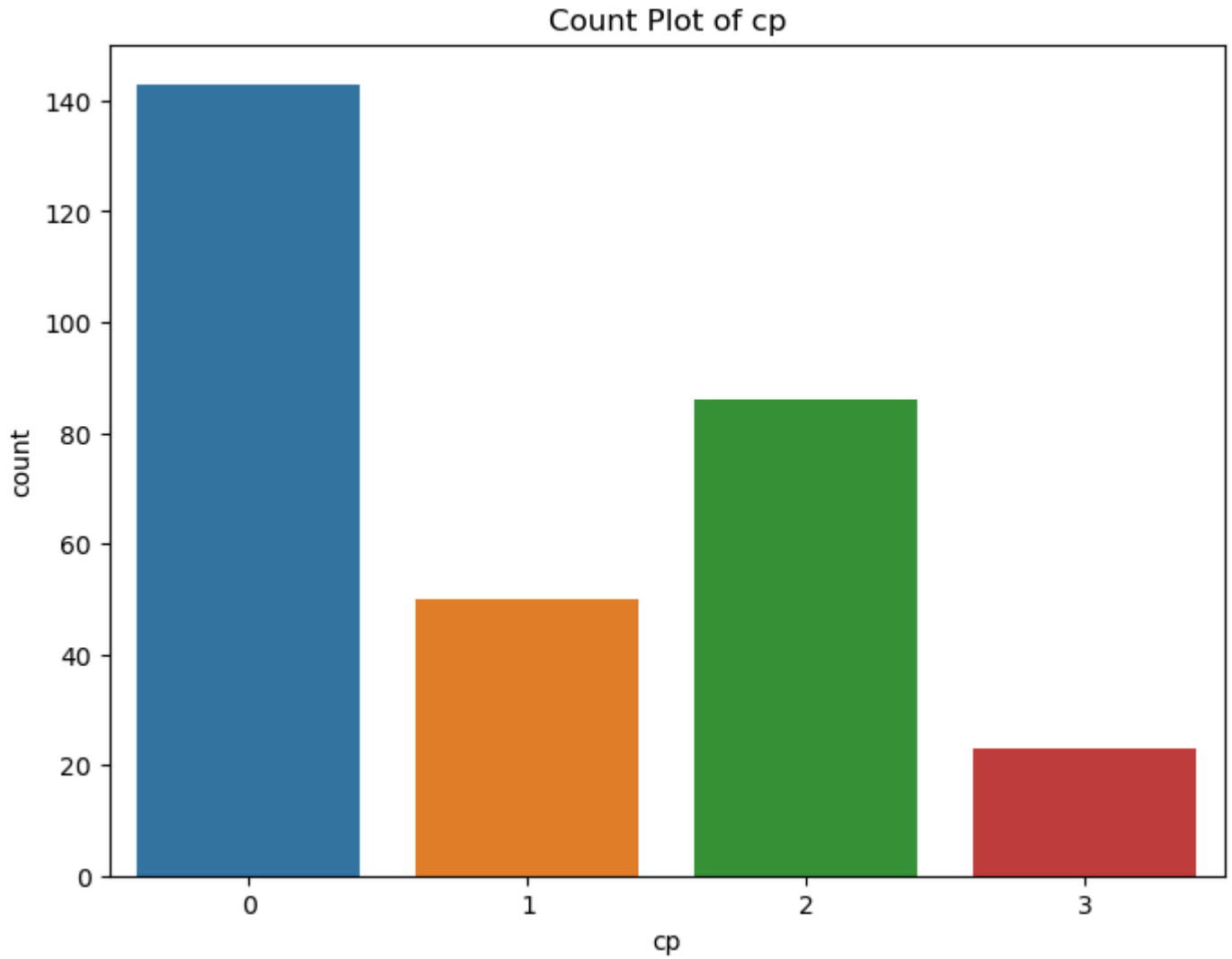
In [12]:
```
data.nunique()
```

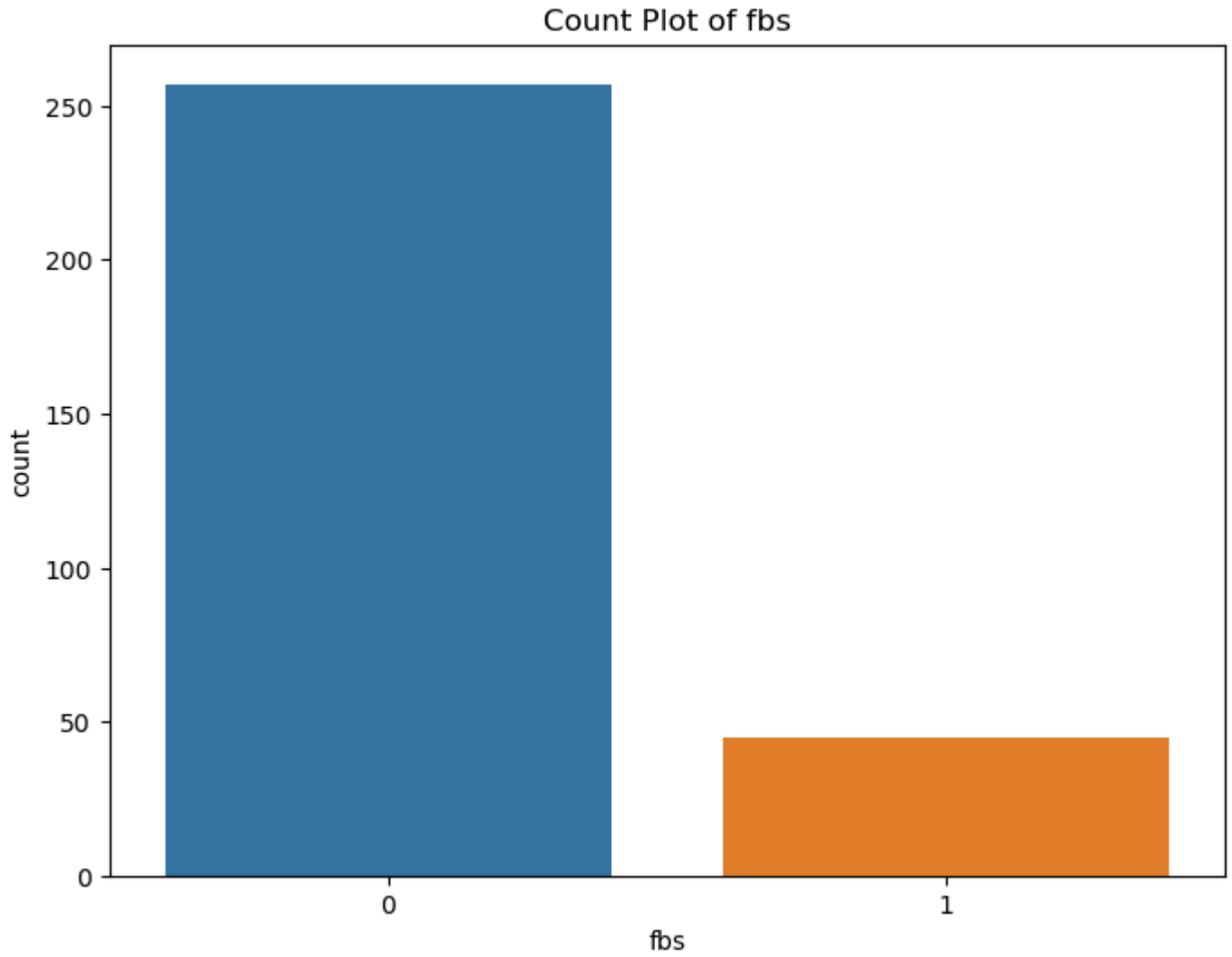
Out[12]:
```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          3
target        2
dtype: int64
```

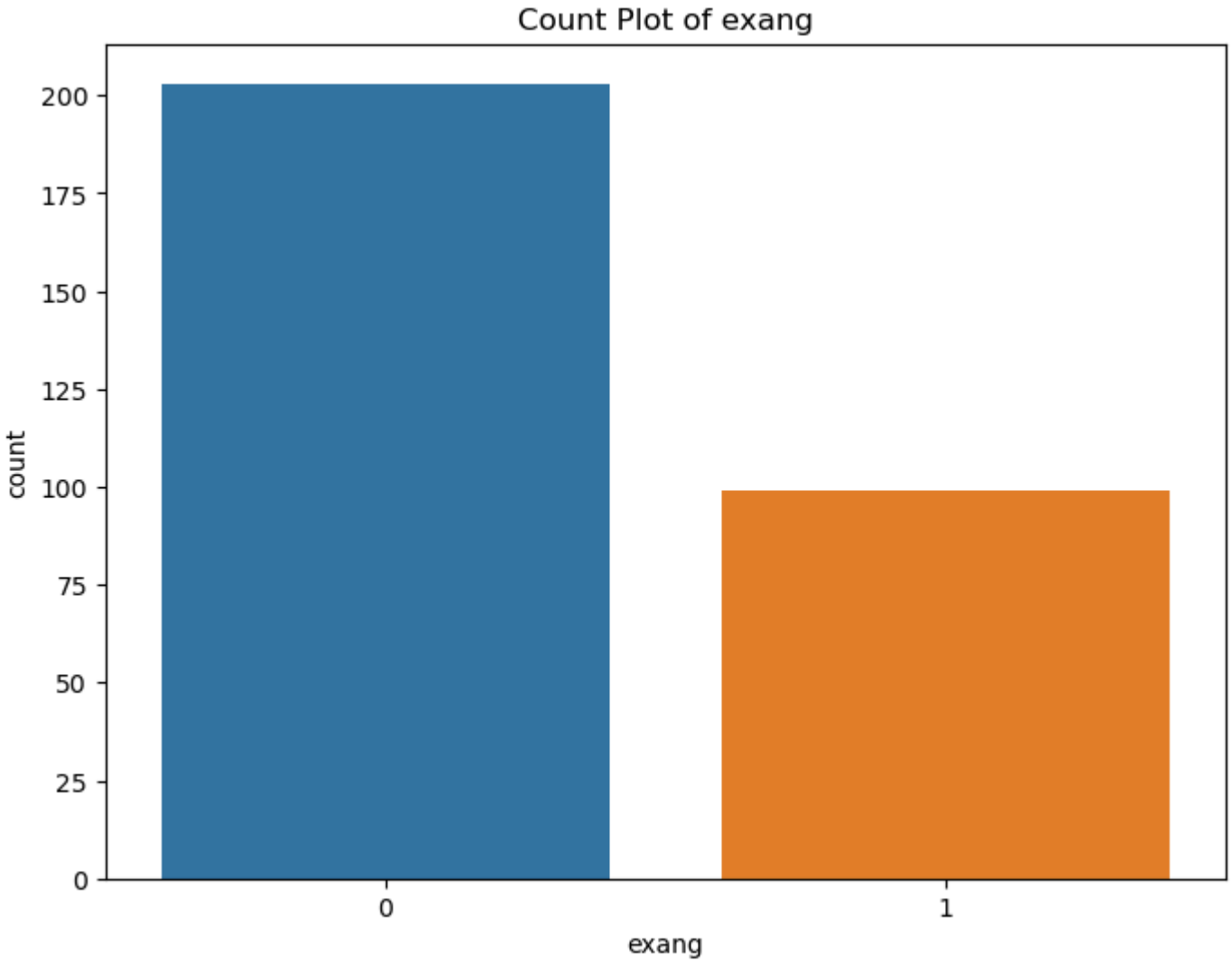
In [13]:
```
cat_vars = ['sex', 'cp', 'fbs','exang', 'thal', 'target']
```

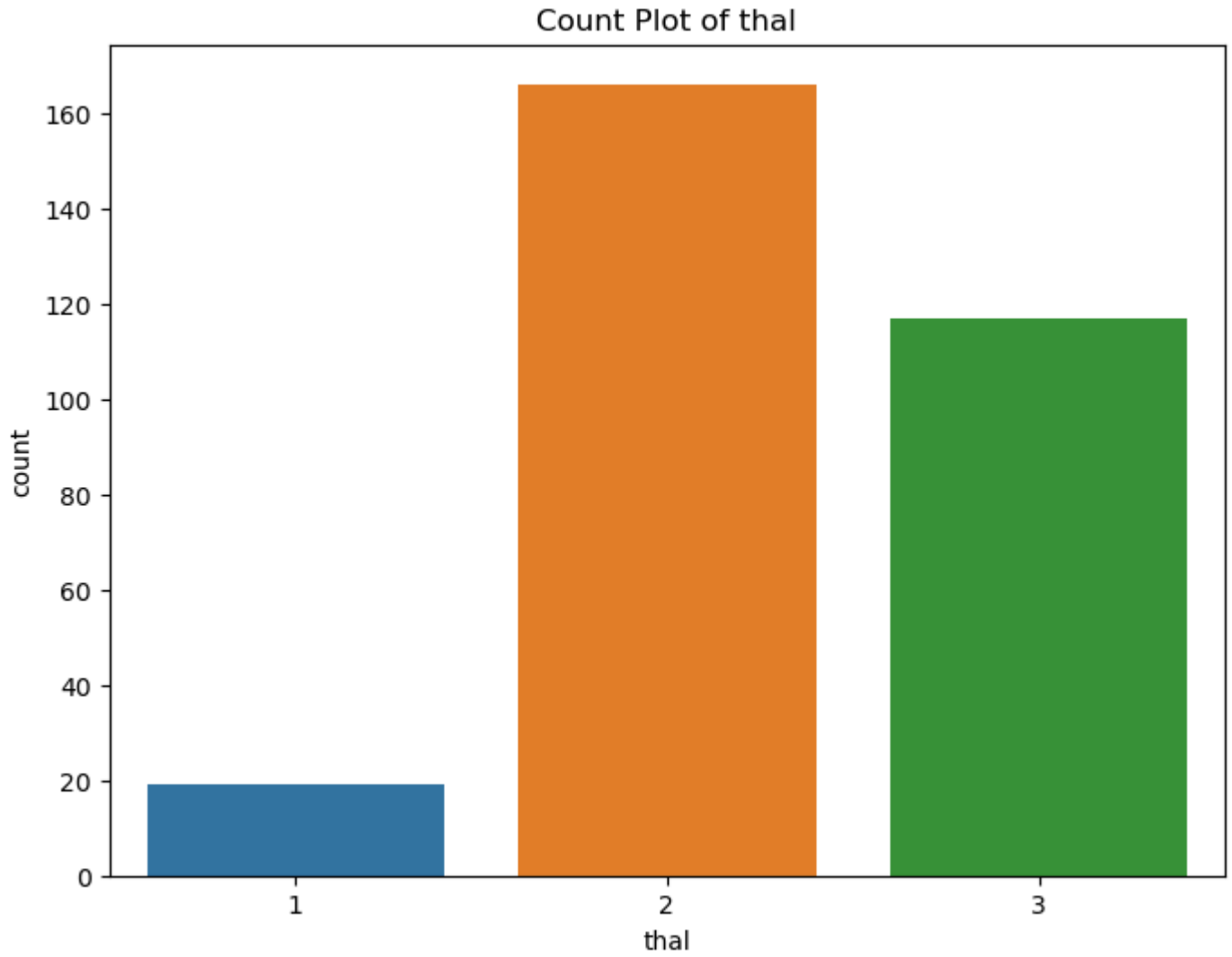
In [14]:
```
for var in cat_vars:
    plt.figure(figsize=(8, 6))
    sns.countplot(x=var, data=data)
    plt.title(f'Count Plot of {var}')
    plt.show()
```

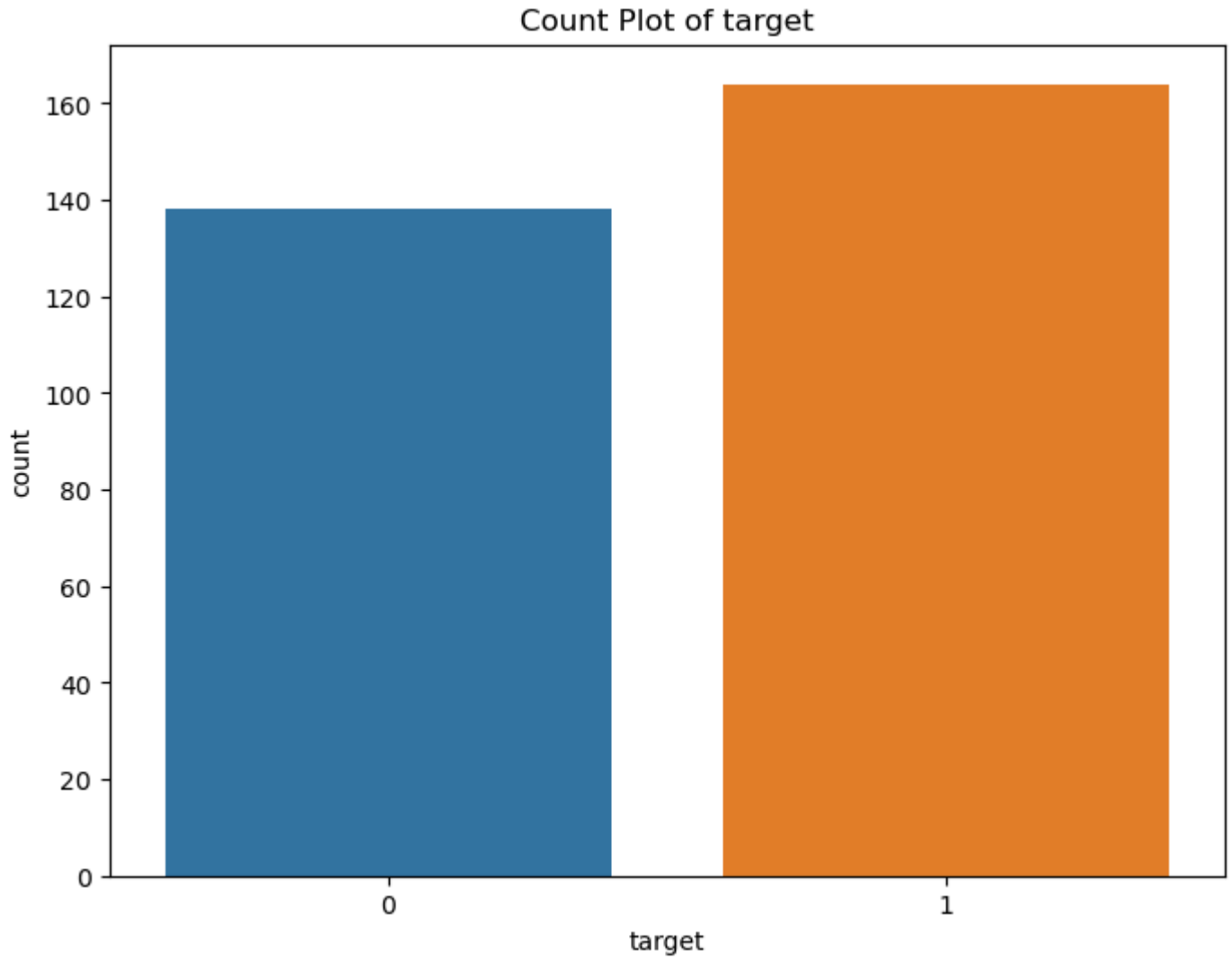

Count Plot of sex
count
200
175
150
125
100
75
50
25
0
0
1
sex


Count Plot of cp
count
cp
0
1
2
3
0
20
40
60
80
100
120
140

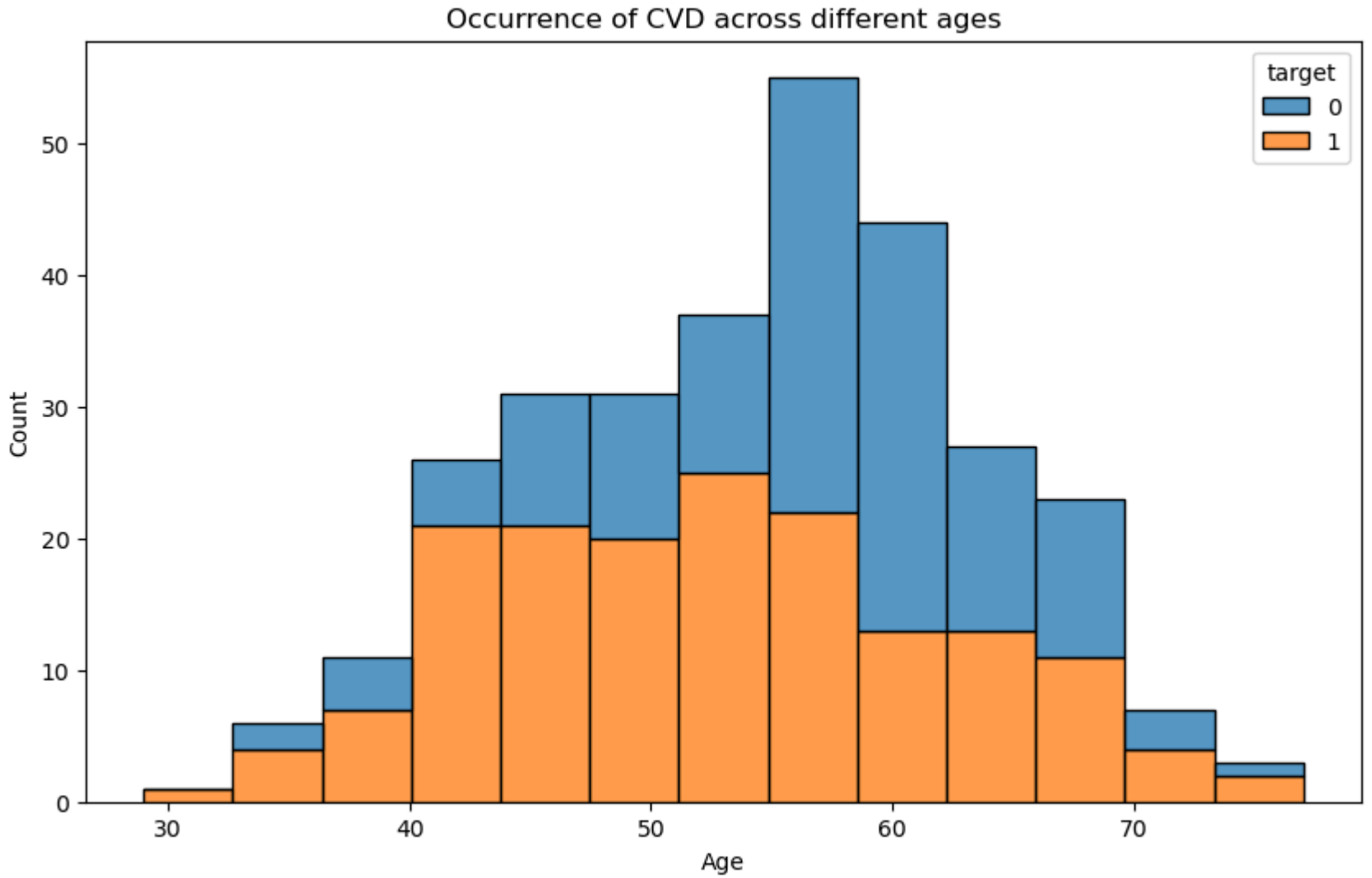

Count Plot of fbs
250
200
150
100
50
0
count
0
1
fbs


Count Plot of exang
count
200
175
150
125
100
75
50
25
0
0
1
exang

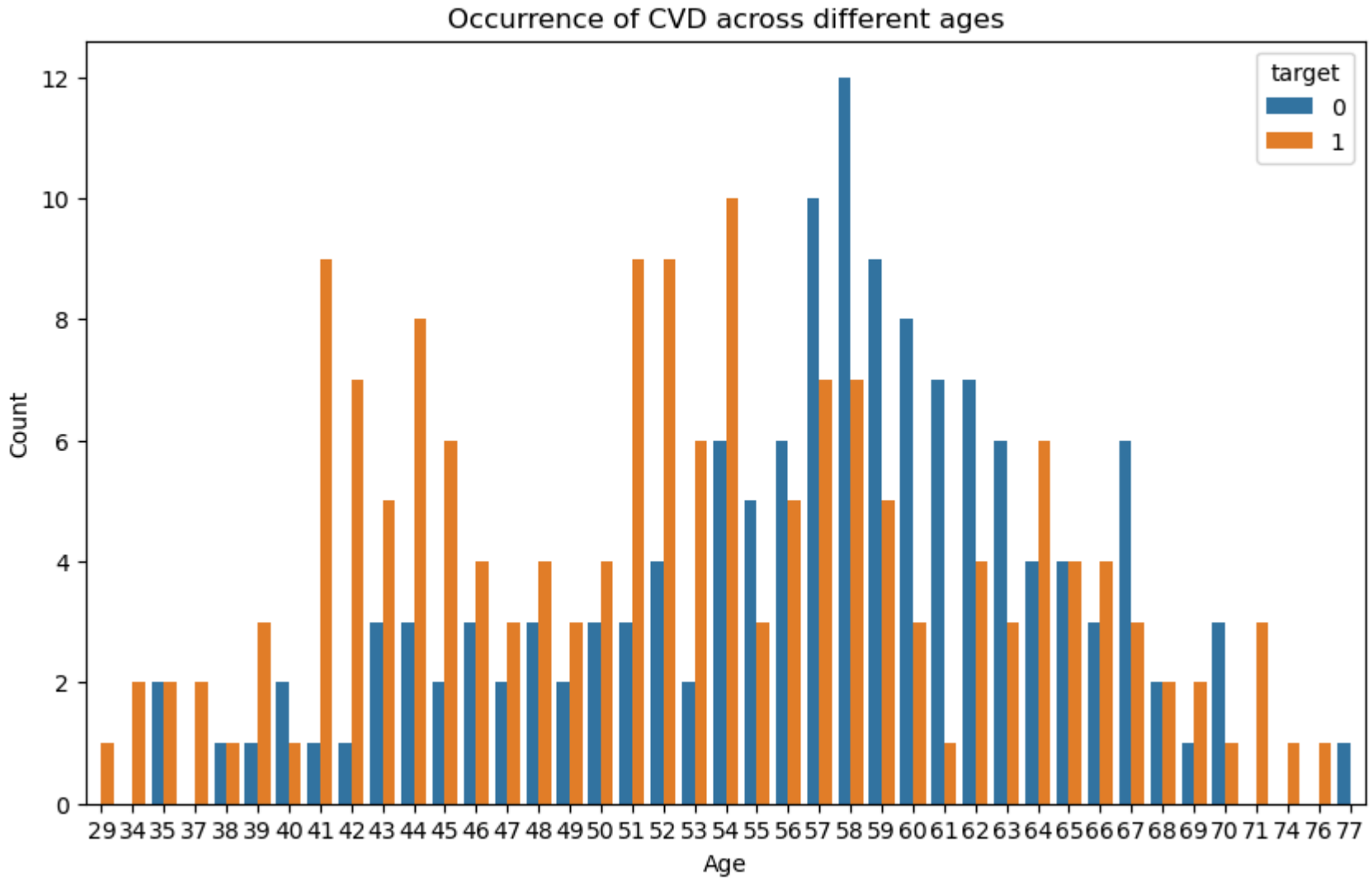

Count Plot of thal
count
thal
0
20
40
60
80
100
120
140
160
1
2
3



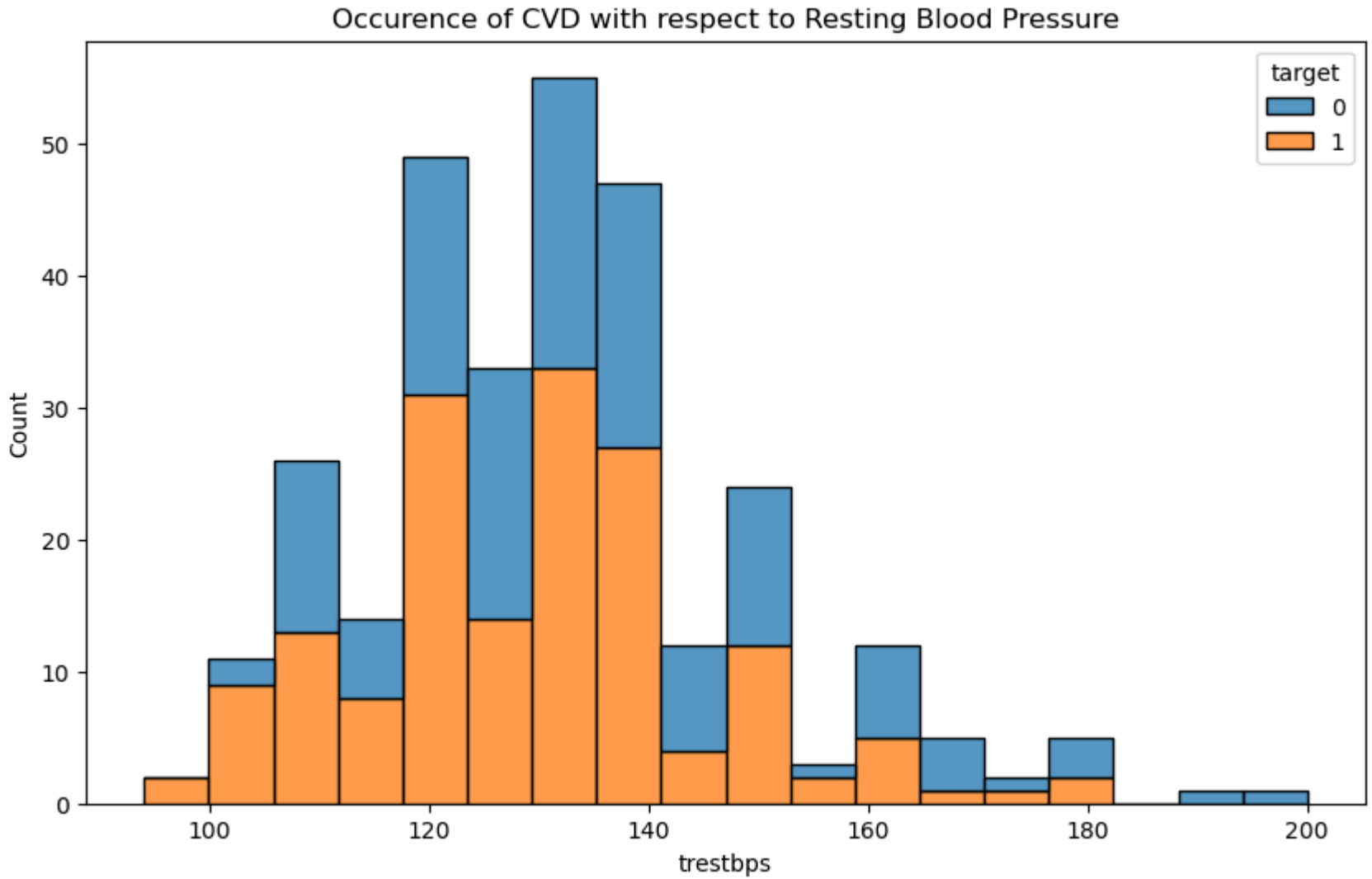
In [15]:
```
plt.figure(figsize=(10,6))
sns.histplot(x='age', hue = 'target', data = data, multiple='stack')
plt.title('Occurrence of CVD across different ages')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



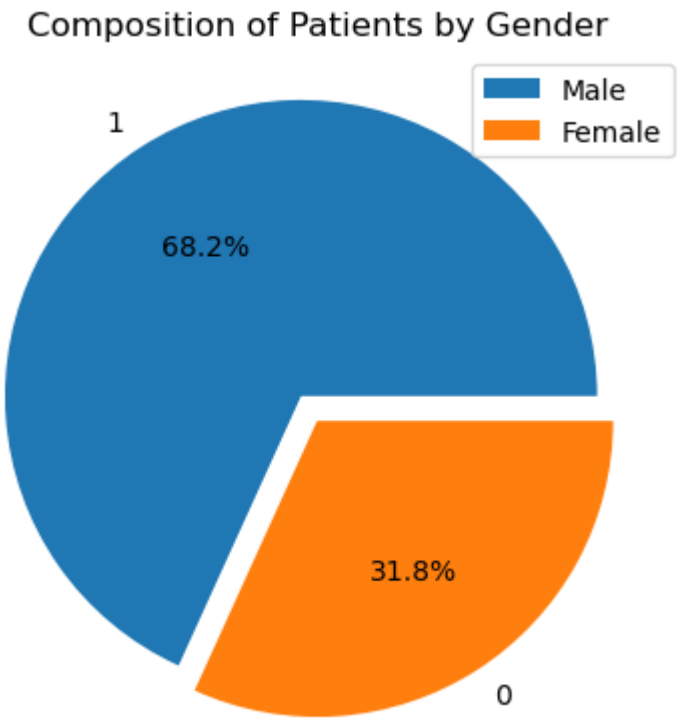
In [16]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='age', hue = 'target', data = data)
plt.title('Occurrence of CVD across different ages')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



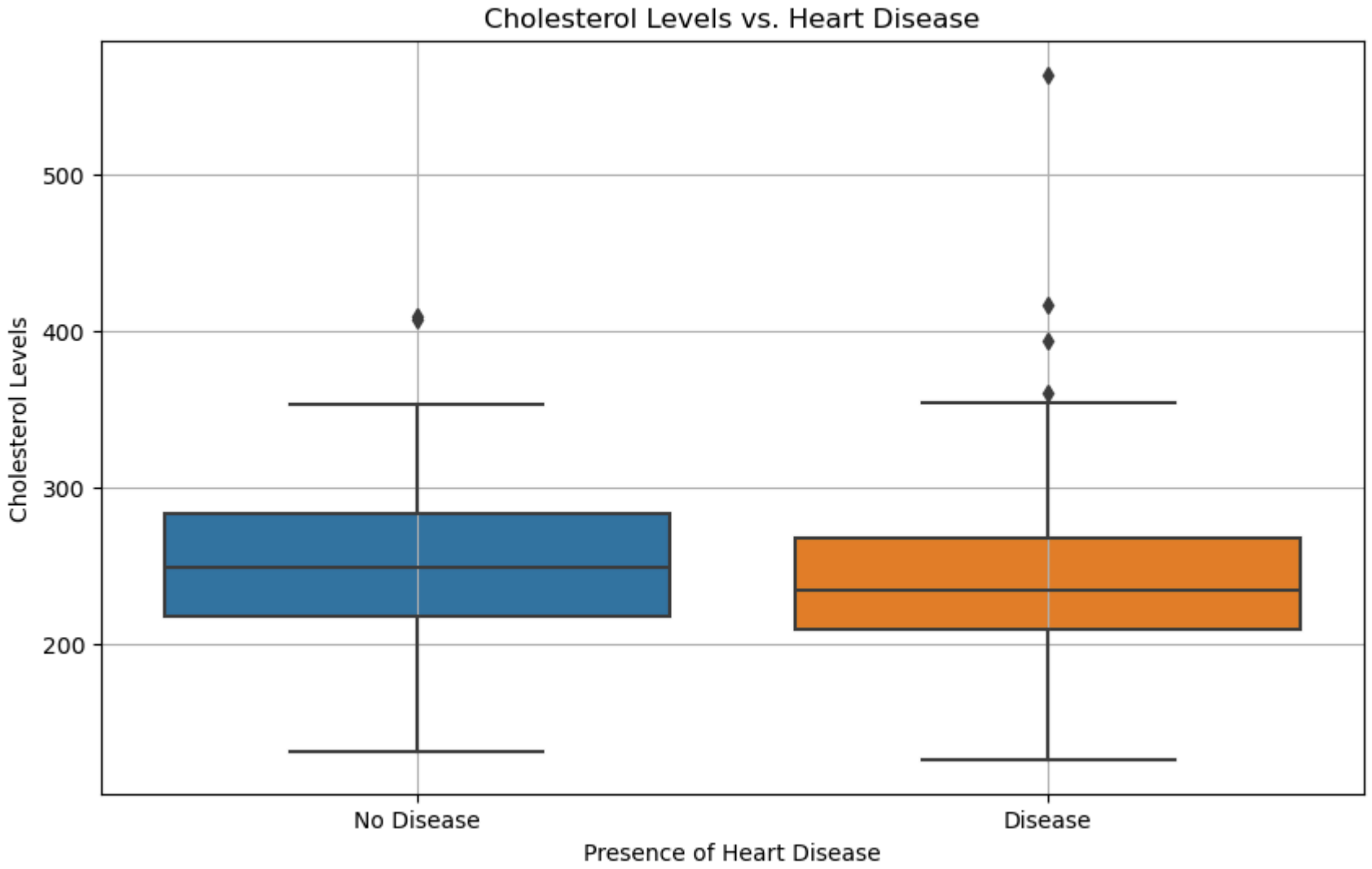
In [17]:
```
plt.figure(figsize=(10,6))
sns.histplot(x='trestbps', hue= 'target', data = data, multiple = 'stack')
plt.title("Occurence of CVD with respect to Resting Blood Pressure")
plt.show()
```



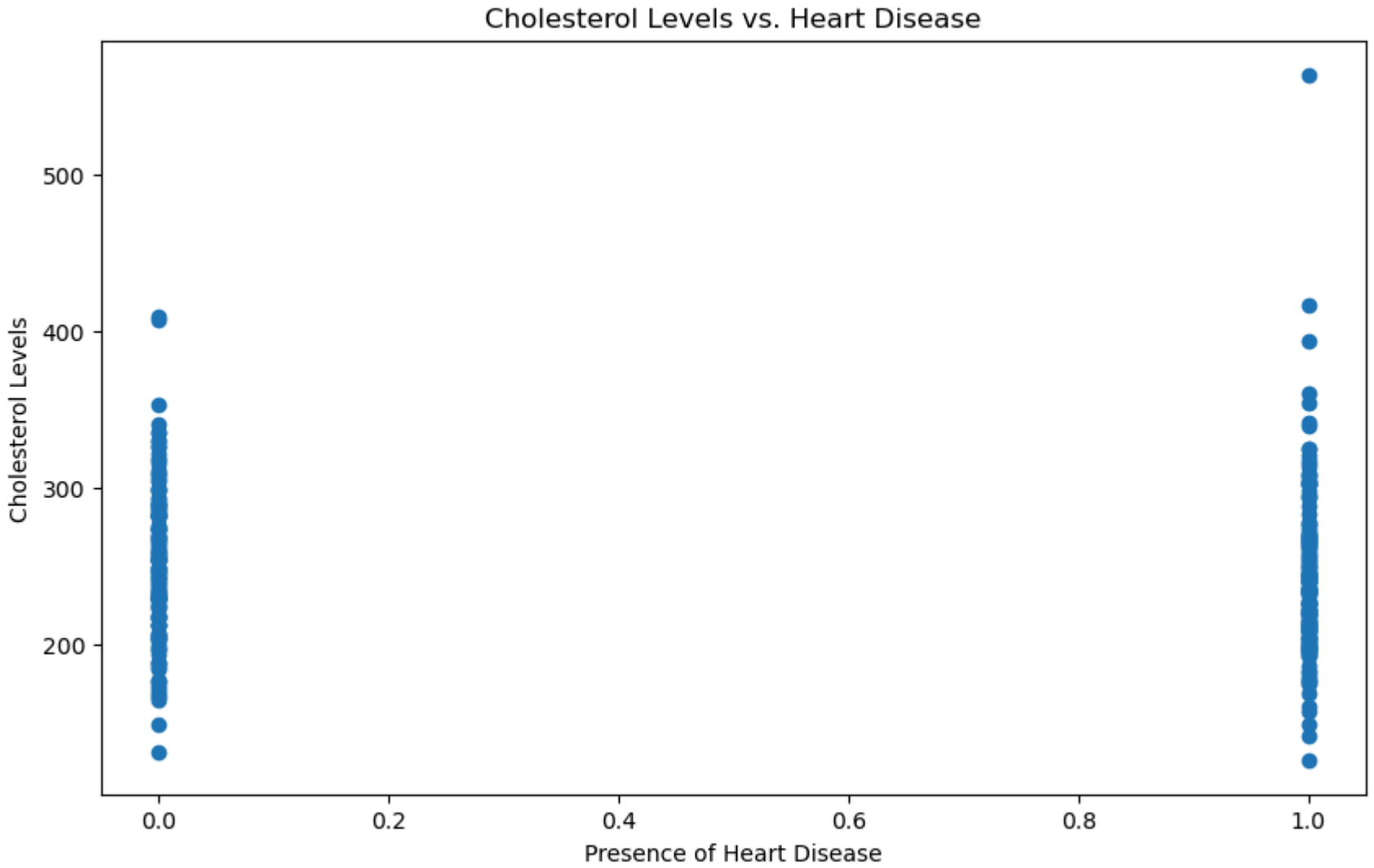
In [18]:
```
data['sex'].value_counts().plot(kind = 'pie', autopct='%1.1f%%', explode= (0.1,0))
plt.title('Composition of Patients by Gender')
plt.ylabel('')
plt.legend(labels=['Male', 'Female'], loc='upper right')
plt.show()
```



In [19]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x='target', y = 'chol', data = data)
plt.title('Cholesterol Levels vs. Heart Disease')
plt.xlabel('Presence of Heart Disease')
plt.ylabel('Cholesterol Levels')
plt.xticks([0, 1], ['No Disease', 'Disease'])
plt.grid(True)
plt.show()
```



In [20]:
```
plt.figure(figsize=(10,6))
plt.scatter(x='target', y = 'chol', data = data)
plt.title('Cholesterol Levels vs. Heart Disease')
plt.xlabel('Presence of Heart Disease')
plt.ylabel('Cholesterol Levels')
plt.show()
```



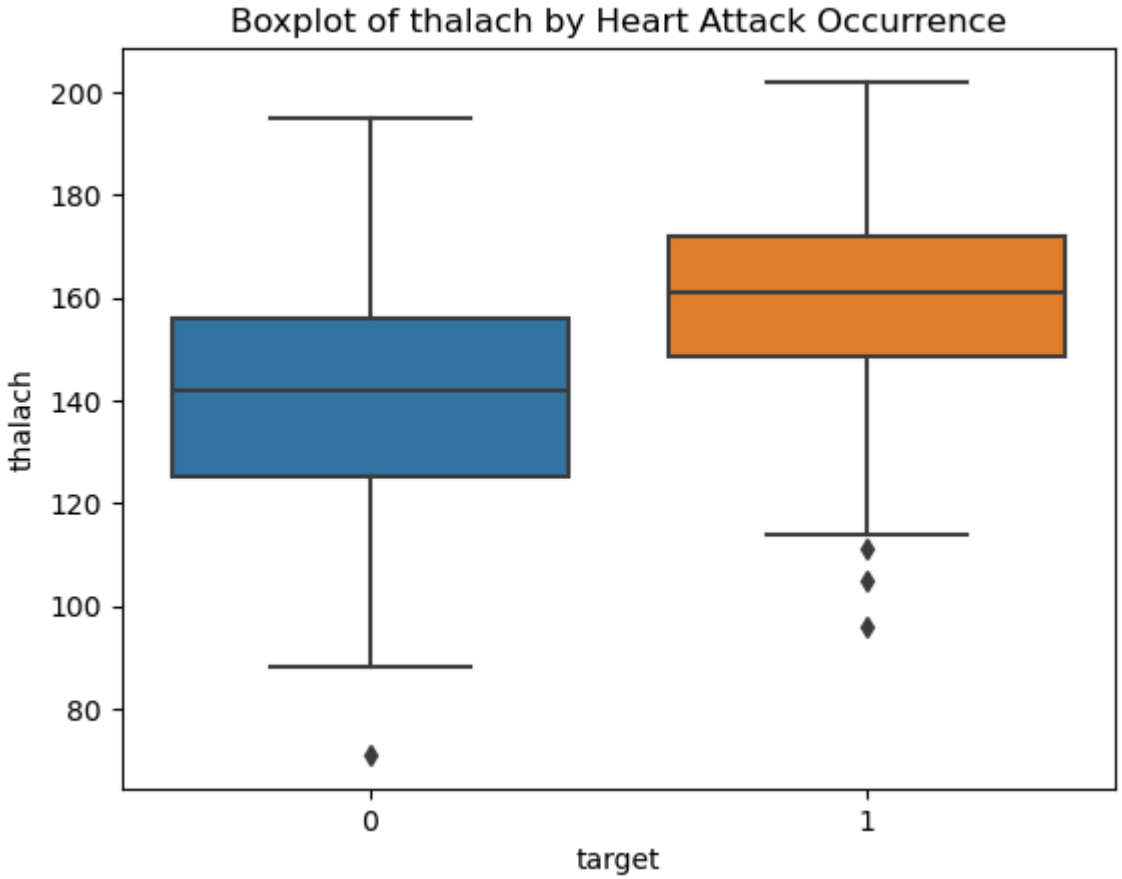
In [21]:
```
correlation = data['chol'].corr(data['target'])
print("Correlation between cholesterol and target:", correlation)
```

```
Correlation between cholesterol and target: -0.08143720051844129
```

In [22]:
```
correlation_1 = data['thalach'].corr(data['target'])
print("Correlation coefficient between thalach and target:", correlation_1)
```

```
Correlation coefficient between thalach and target: 0.41995504366386954
```

In [23]:
```
sns.boxplot(x='target', y='thalach', data=data)
plt.title("Boxplot of thalach by Heart Attack Occurrence")
plt.show()
```



In [24]:
```
X = data[['thalach']]
y = data['target']
```

In [26]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [28]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [29]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[29]:
```
LogisticRegression
LogisticRegression()
```

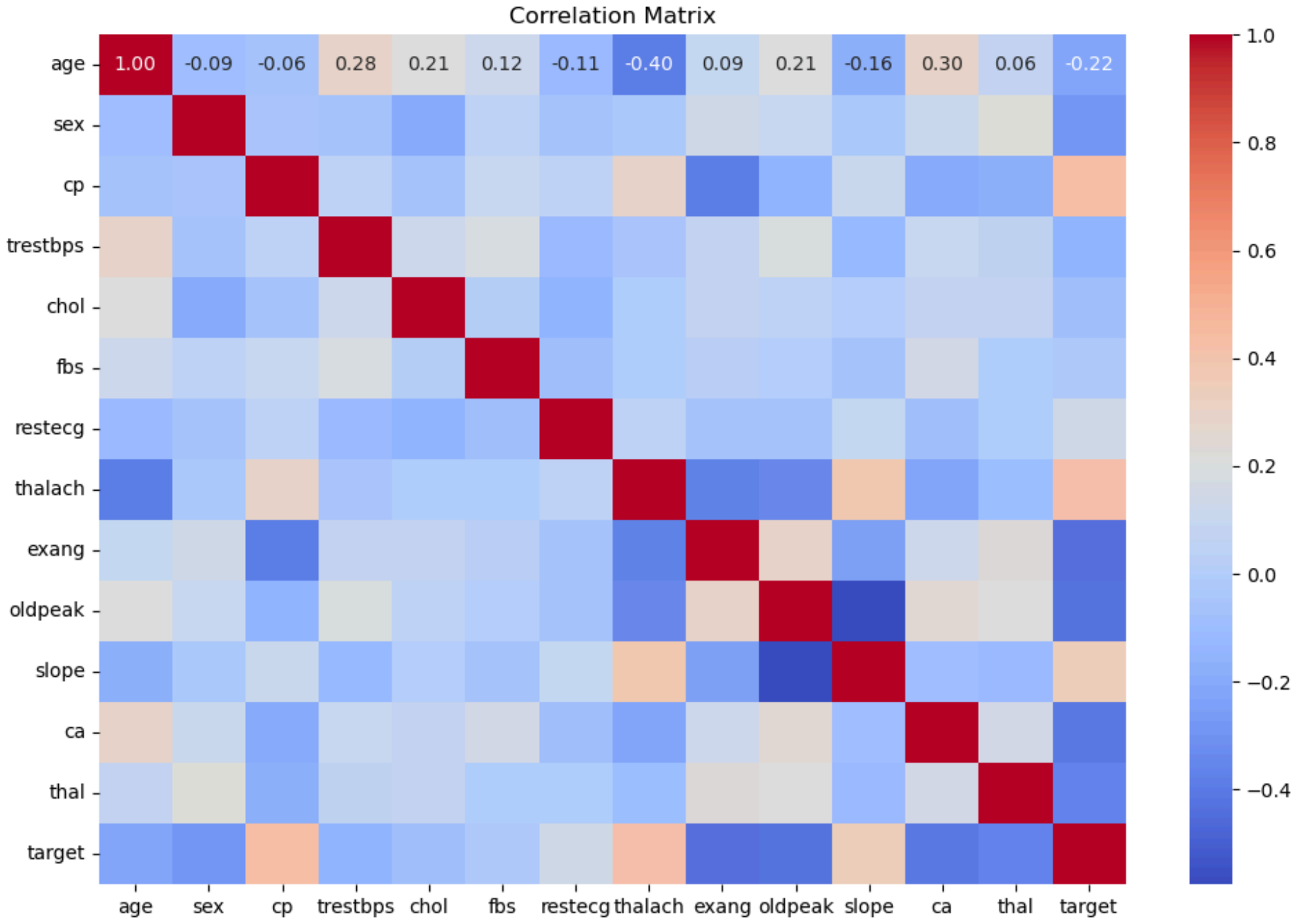
In [30]:
```
predictions = model.predict(X_test)
```

In [31]:
```
accuracy = accuracy_score(y_test, predictions)
conf_matrix = confusion_matrix(y_test, predictions)
class_report = classification_report(y_test, predictions)
```

In [32]:
```
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

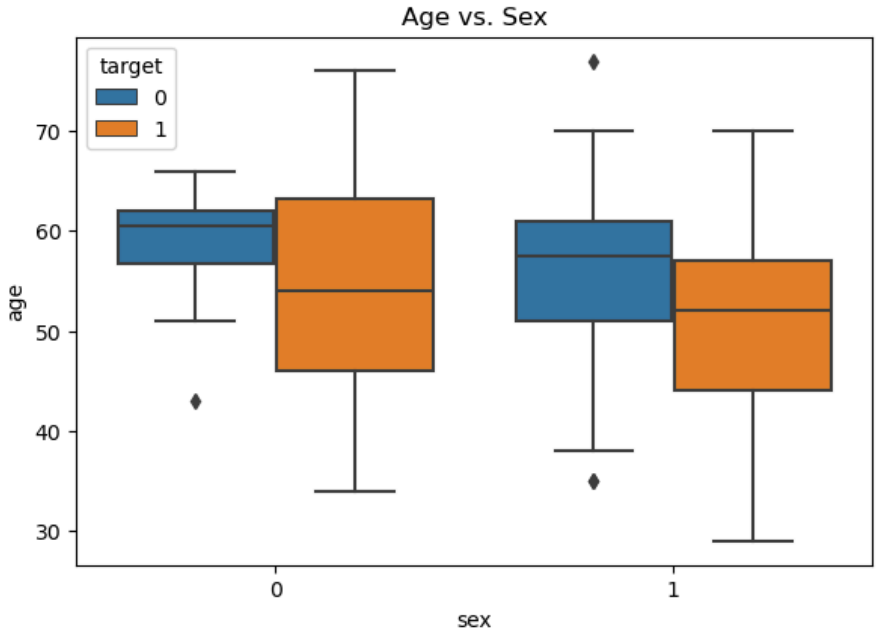
```
Accuracy: 0.7049180327868853
Confusion Matrix:
 [[16 13]
 [ 5 27]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.55      0.64        29
           1       0.68      0.84      0.75        32

    accuracy                           0.70        61
   macro avg       0.72      0.70      0.70        61
weighted avg       0.72      0.70      0.70        61
```

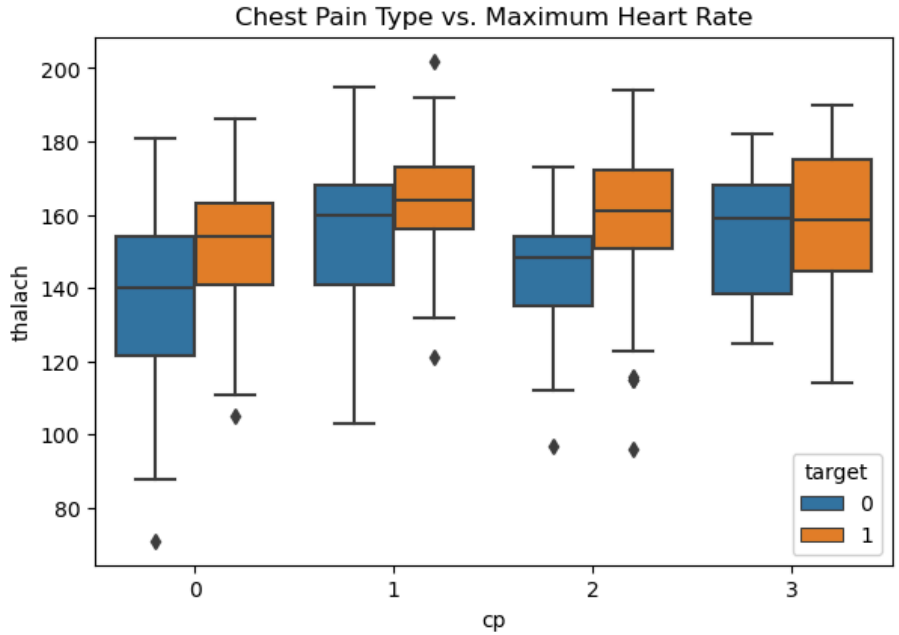
In [33]:
```
corr_matrix = data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

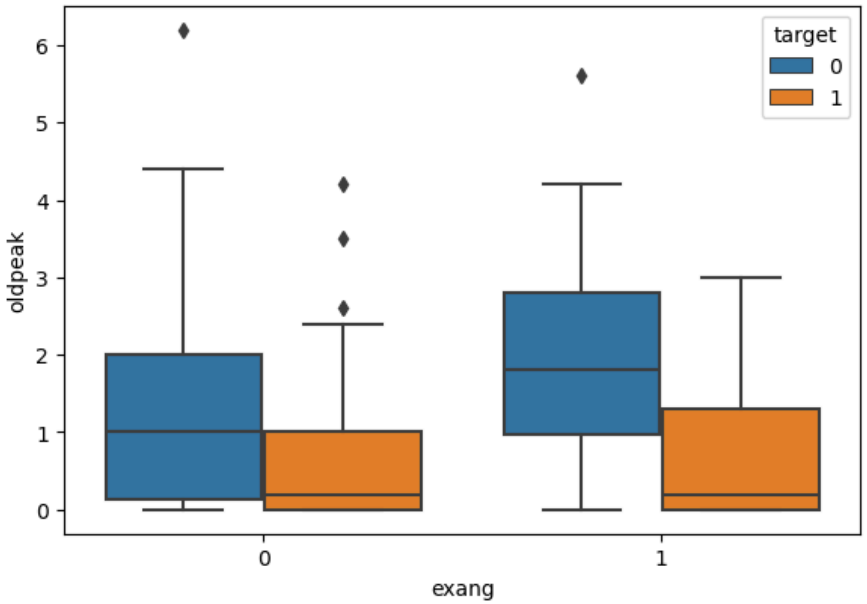


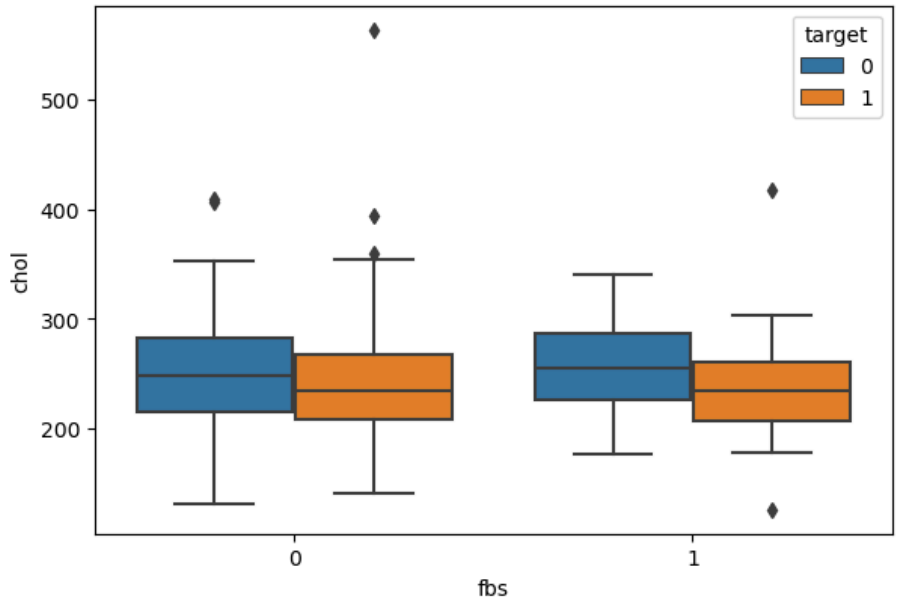
In [35]:
```
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
```

```
sns.boxplot(x='sex', y='age', hue='target', data=data)
plt.title("Age vs. Sex")

plt.subplot(2, 2, 2)
sns.boxplot(x='cp', y='thalach', hue='target', data=data)
plt.title("Chest Pain Type vs. Maximum Heart Rate")

plt.subplot(2, 2, 3)
sns.boxplot(x='exang', y='oldpeak', hue='target', data=data)
plt.title("Exercise-Induced Angina vs. ST Depression")

plt.subplot(2, 2, 4)
sns.boxplot(x='fbs', y='chol', hue='target', data=data)
plt.title("Fasting Blood Sugar vs. Cholesterol")
plt.show()
```







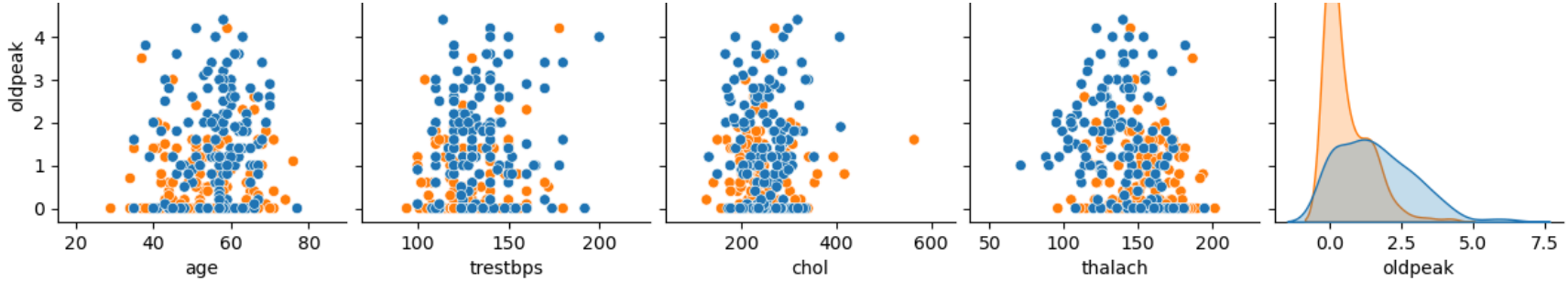


In [37]:
```
data.columns
```

Out[37]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [39]:
```
num_var=['age','trestbps', 'chol', 'thalach', 'oldpeak']
sns.pairplot(data[num_var+['target']],hue='target')
plt.show()
```


age
trestbps
chol
thalach
target
0
1



In [40]:
```
X = data.drop('target', axis=1)
y = data['target']
```

In [41]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [42]:
```
model = LogisticRegression()
```

In [44]:
```
model.fit(X_train, y_train)
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (sta
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[44]:
```
LogisticRegression
LogisticRegression()
```

In [45]:
```
y_pred = model.predict(X_test)
```

In [47]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
```

In [48]:
```
print("Confusion Matrix:")
print(conf_matrix)
```

```
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Confusion Matrix:
[[26  3]
 [ 4 28]]
Accuracy: 88.52%
```