In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import statsmodels.api as sm
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
df = pd.read_excel('CEP_1_Dataset.xlsx')

# Step 1.1: Preliminary Data Inspection
print(df.shape)
print(df.info())
print(df.duplicated().sum())
df.head(3)
```

```
(303, 14)
<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
None
1
```

Out[ ]:

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

In [ ]:

```
df.nunique(axis=0)
```

Out[ ]:

```
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          4
target        2
dtype: int64
```

In [ ]:

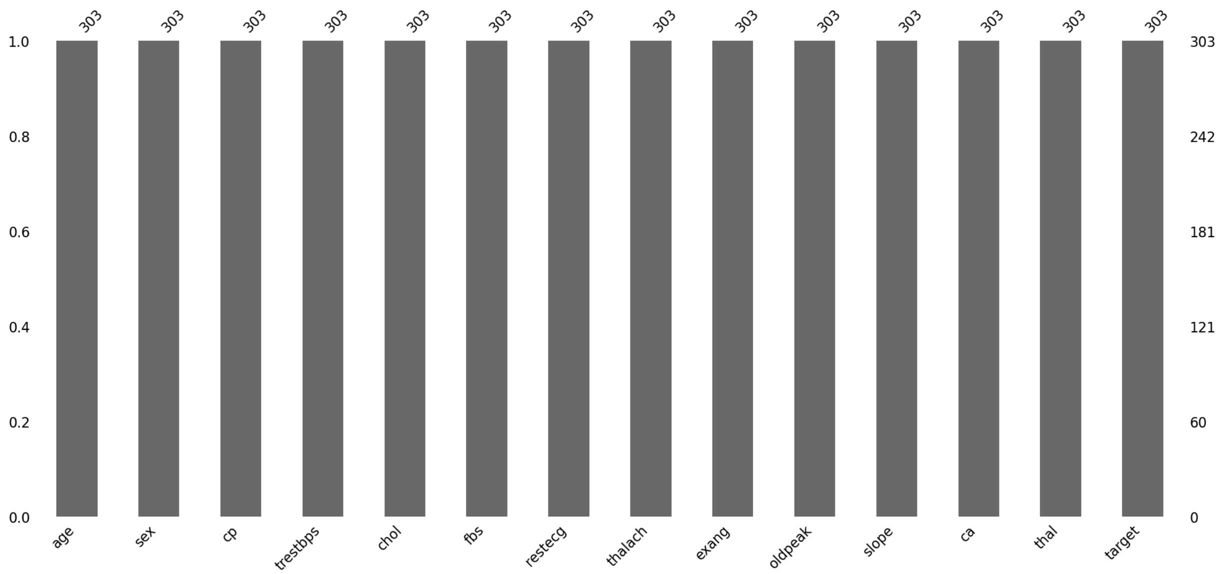
```
df.describe()
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

In [ ]:

```
msno.bar(df)
plt.show()
```

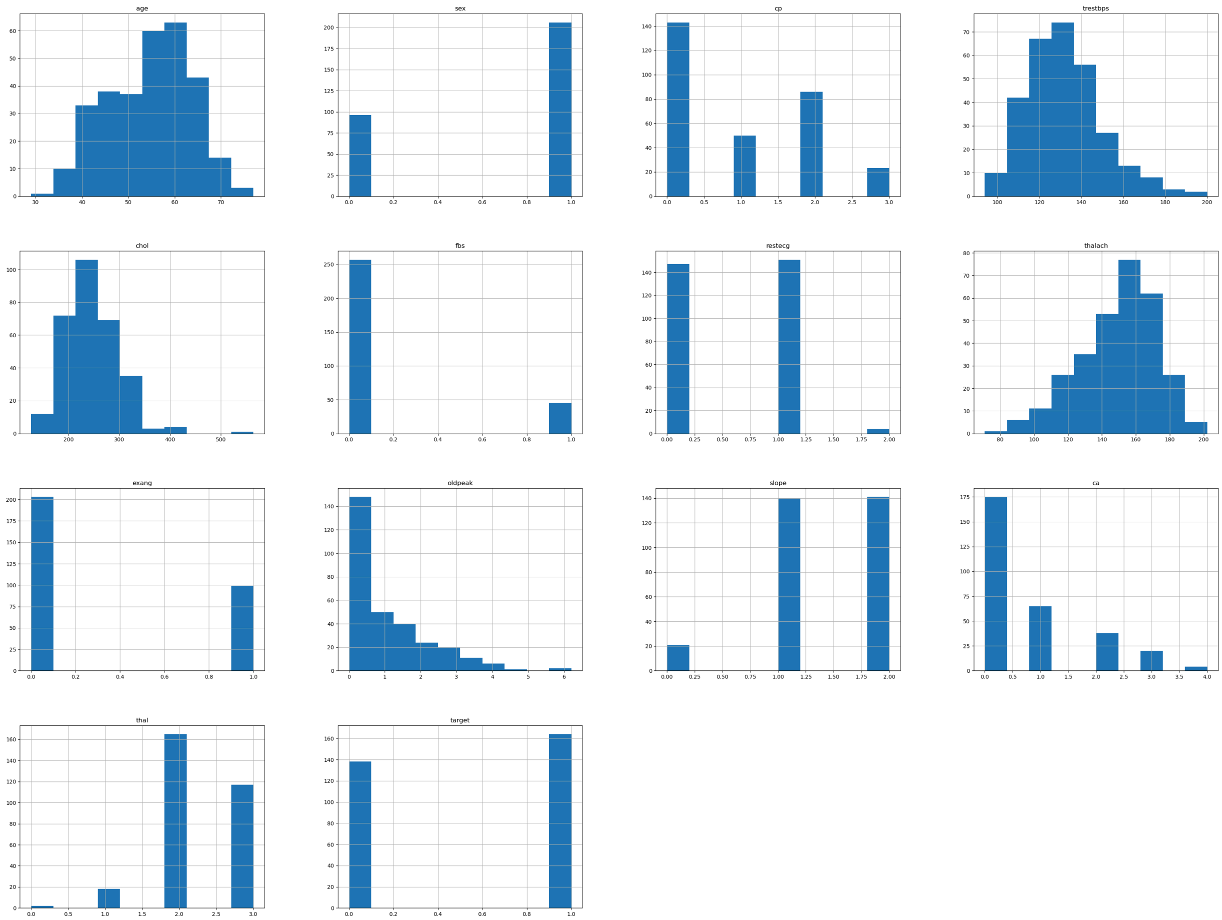


No Missing Values!

In [ ]:
```
df.drop_duplicates(inplace=True)
df.duplicated().any()
```

Out[ ]: False

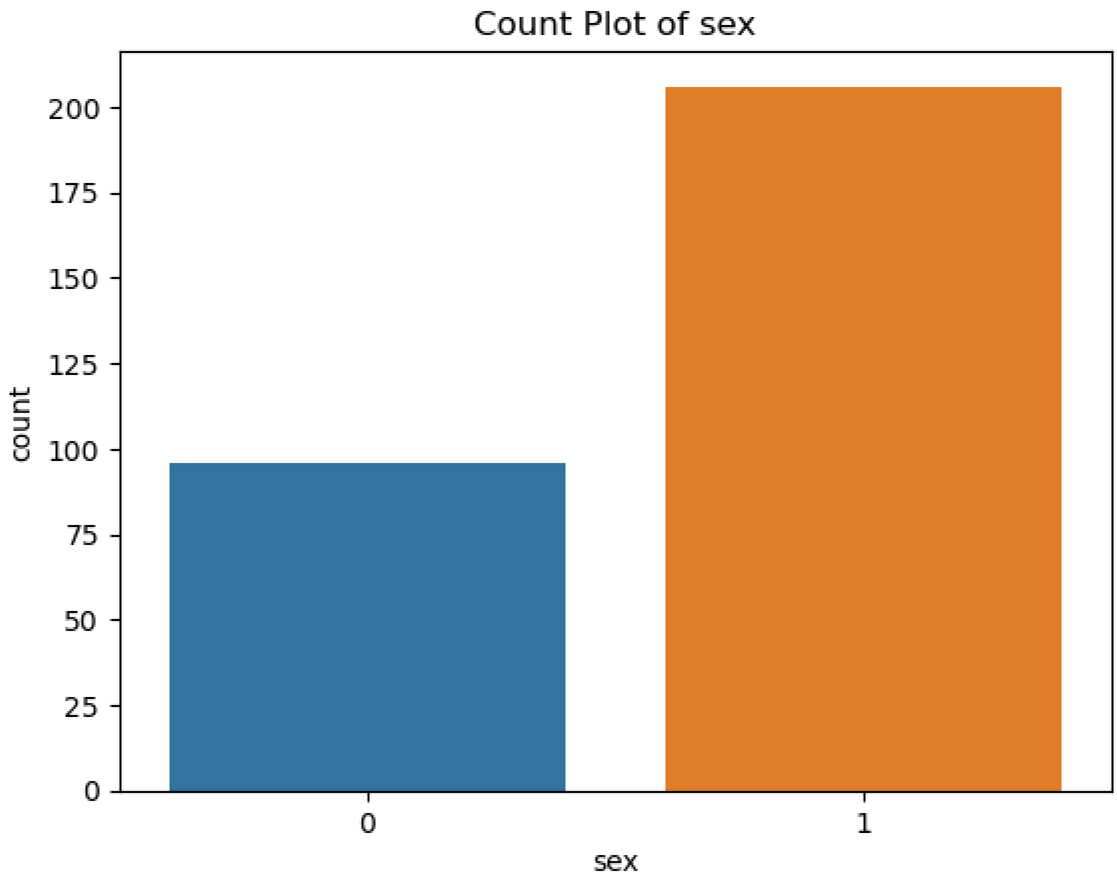
In [ ]:
```
# Data Visualization
fig = plt.figure(figsize = (40,30))
df.hist(ax = fig.gca());
```

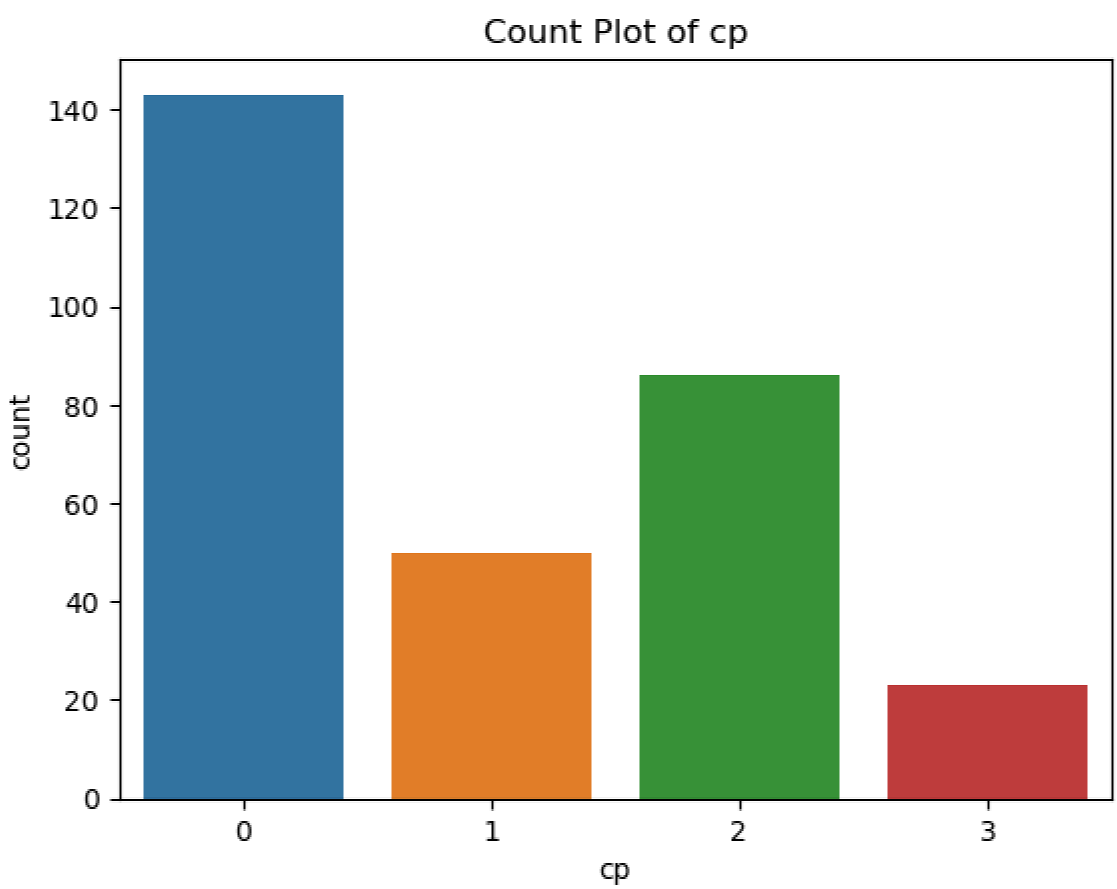


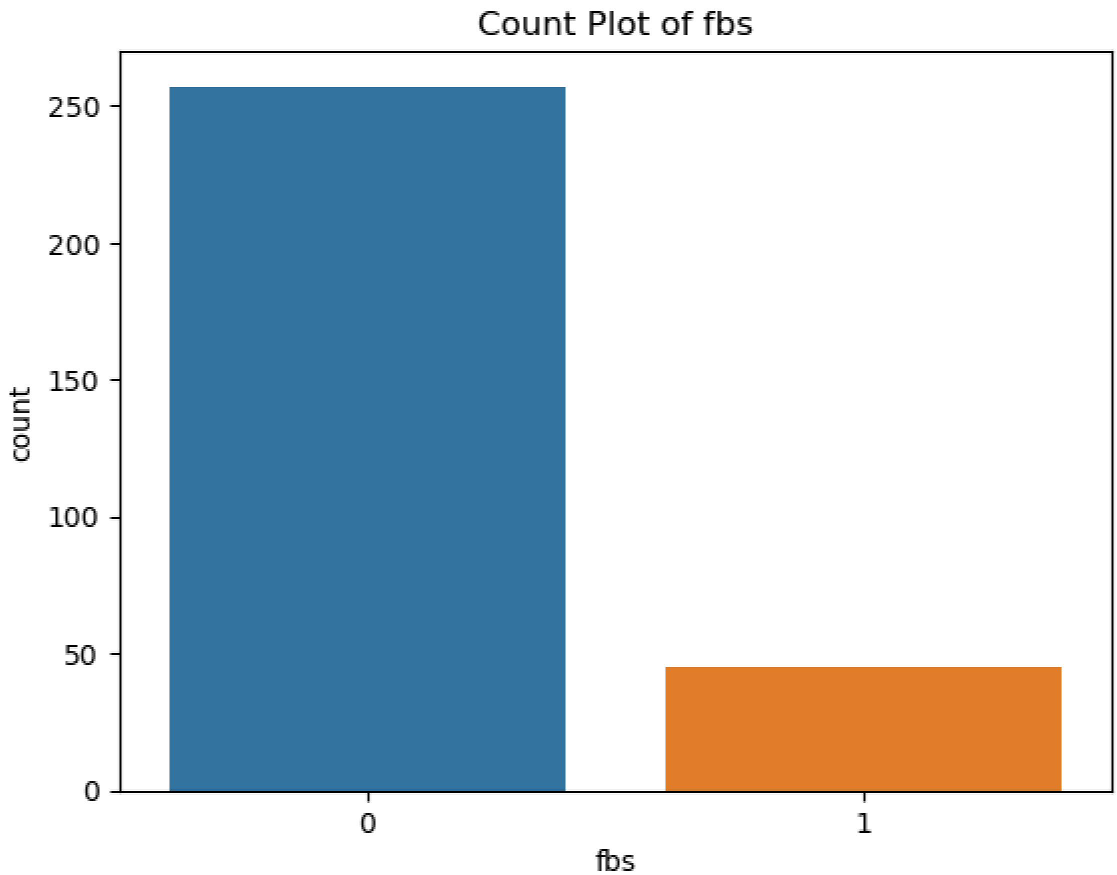
In [ ]:
```python
# Function to determine categorical variables with heuristics
def identify_categorical(df, threshold=10):
    categorical_vars = []
    for column in df.columns:
        unique_values = df[column].nunique()
        if unique_values <= threshold and df[column].dtype == 'object':
            categorical_vars.append(column)
        elif unique_values <= threshold and df[column].dtype != 'object':
            if unique_values < len(df) * 0.05:
                categorical_vars.append(column)
    return categorical_vars

categorical_vars = identify_categorical(df)
numerical_vars = [col for col in df.columns if col not in categorical_vars]

print("Identified Categorical Variables:", categorical_vars)
print("Identified Numerical Variables:", numerical_vars)
```

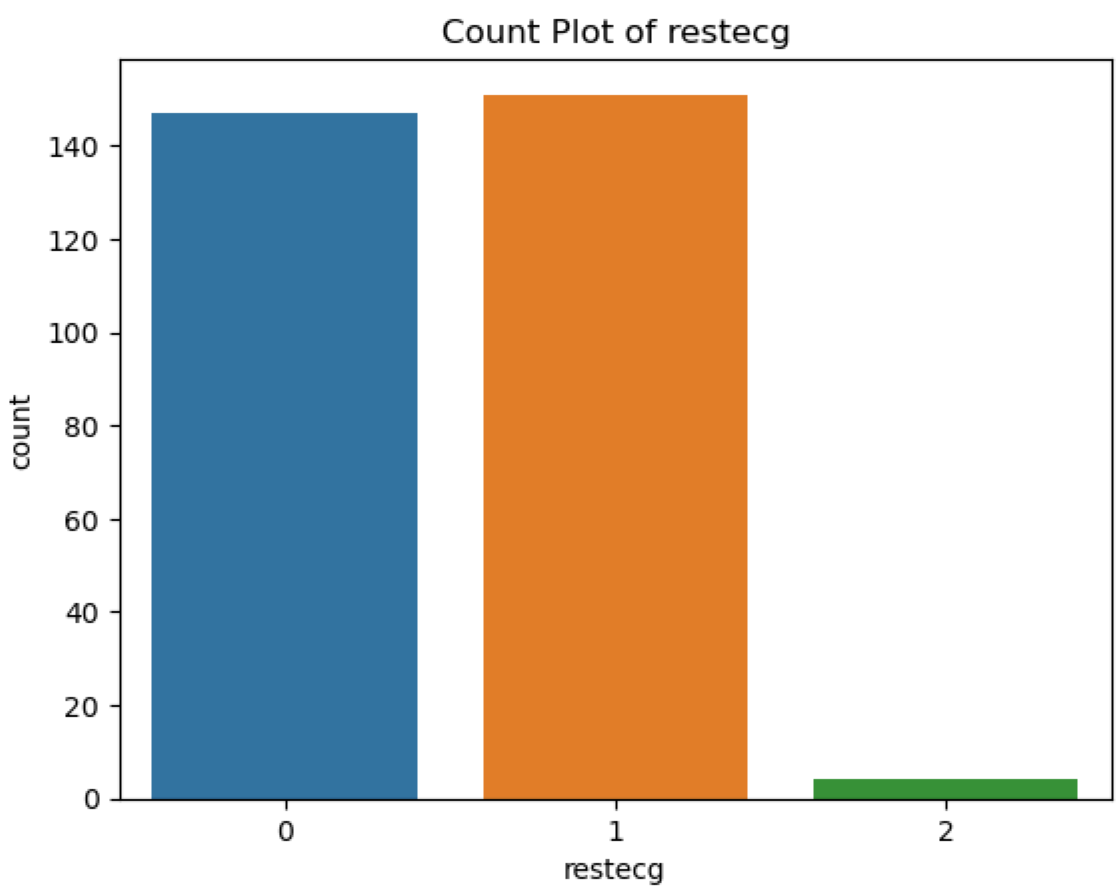
```
Identified Categorical Variables: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope',
'ca', 'thal', 'target']
Identified Numerical Variables: ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

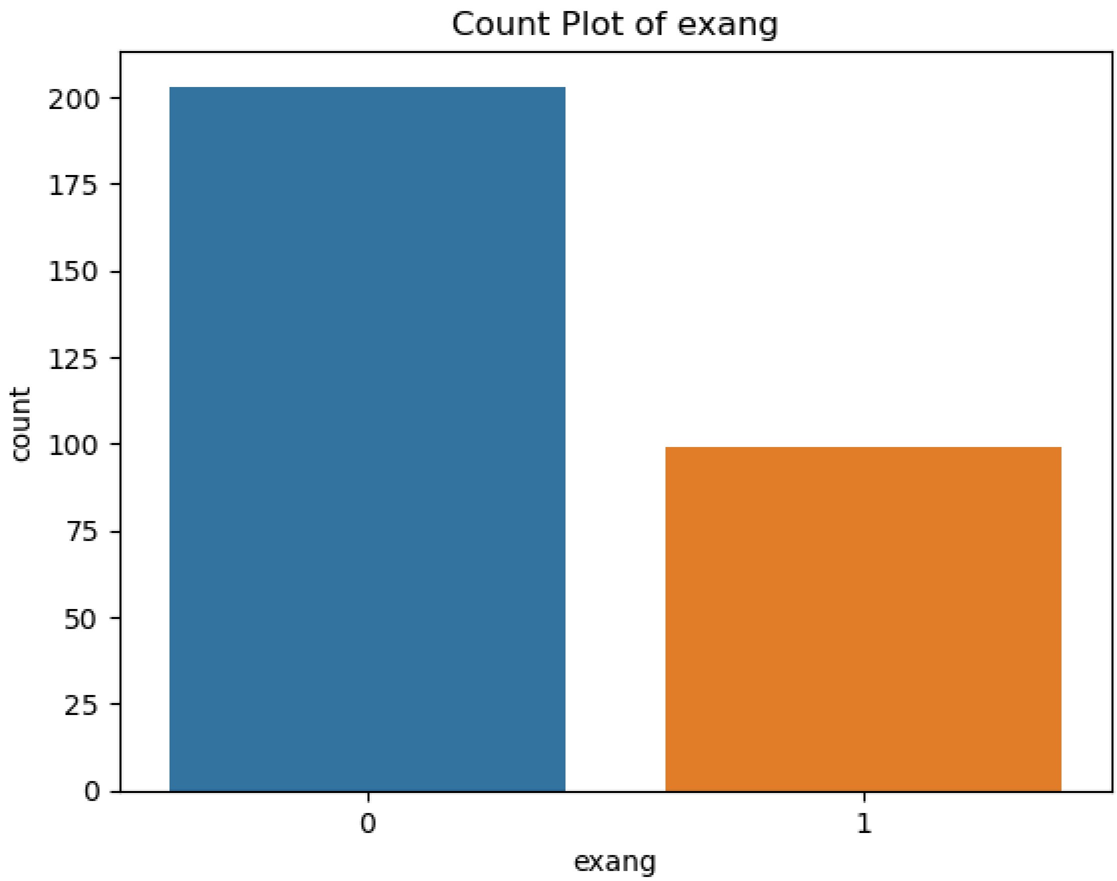
In [ ]:
```python
# Visualize categorical variables
for var in categorical_vars:
    sns.countplot(x=var, data=df)
    plt.title(f'Count Plot of {var}')
    plt.show()
```

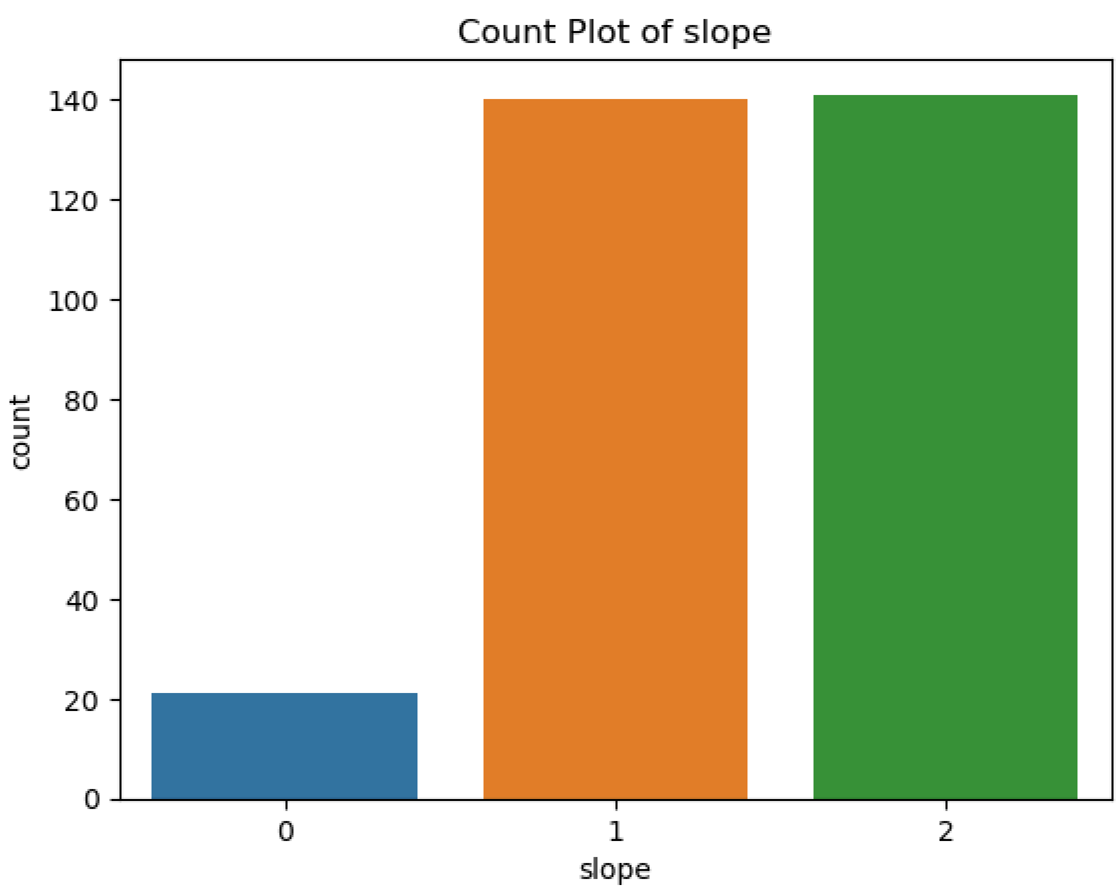

Count Plot of sex
count
200
175
150
125
100
75
50
25
0
0
1
sex

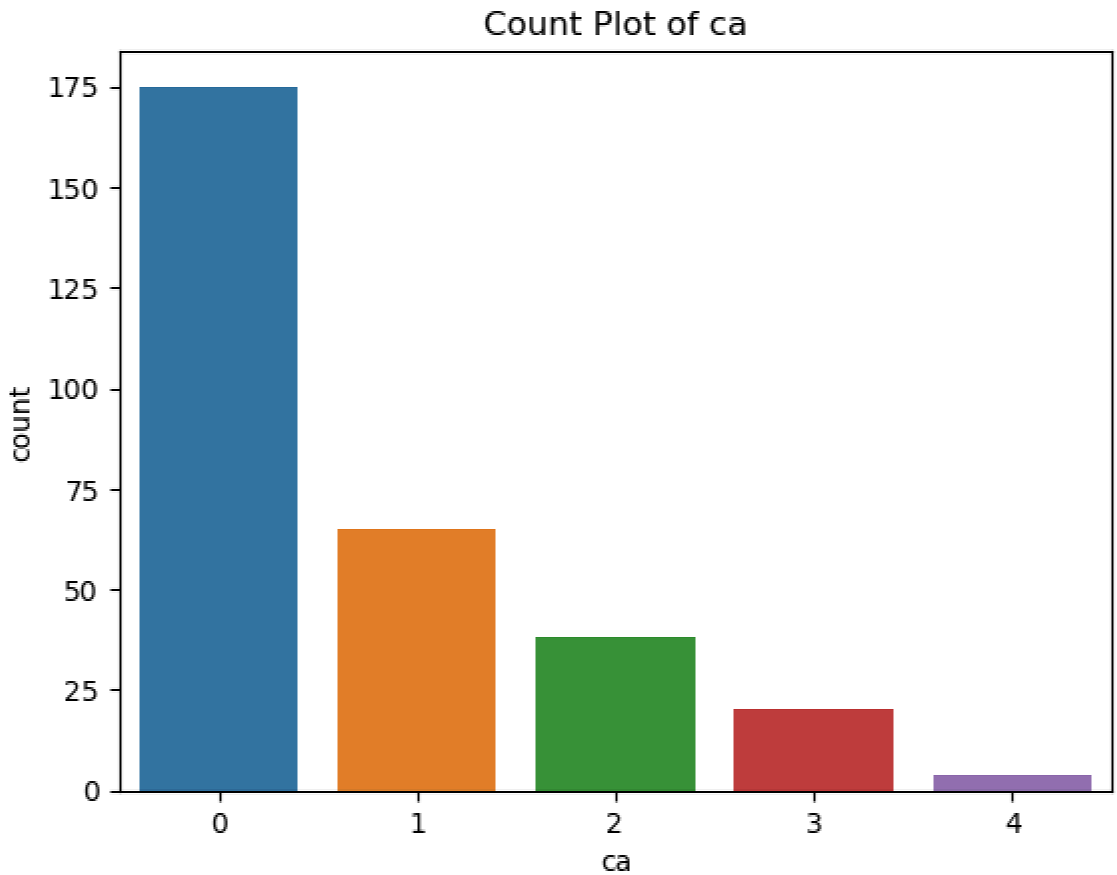

Count Plot of cp
count
140
120
100
80
60
40
20
0
0
1
2
3
cp

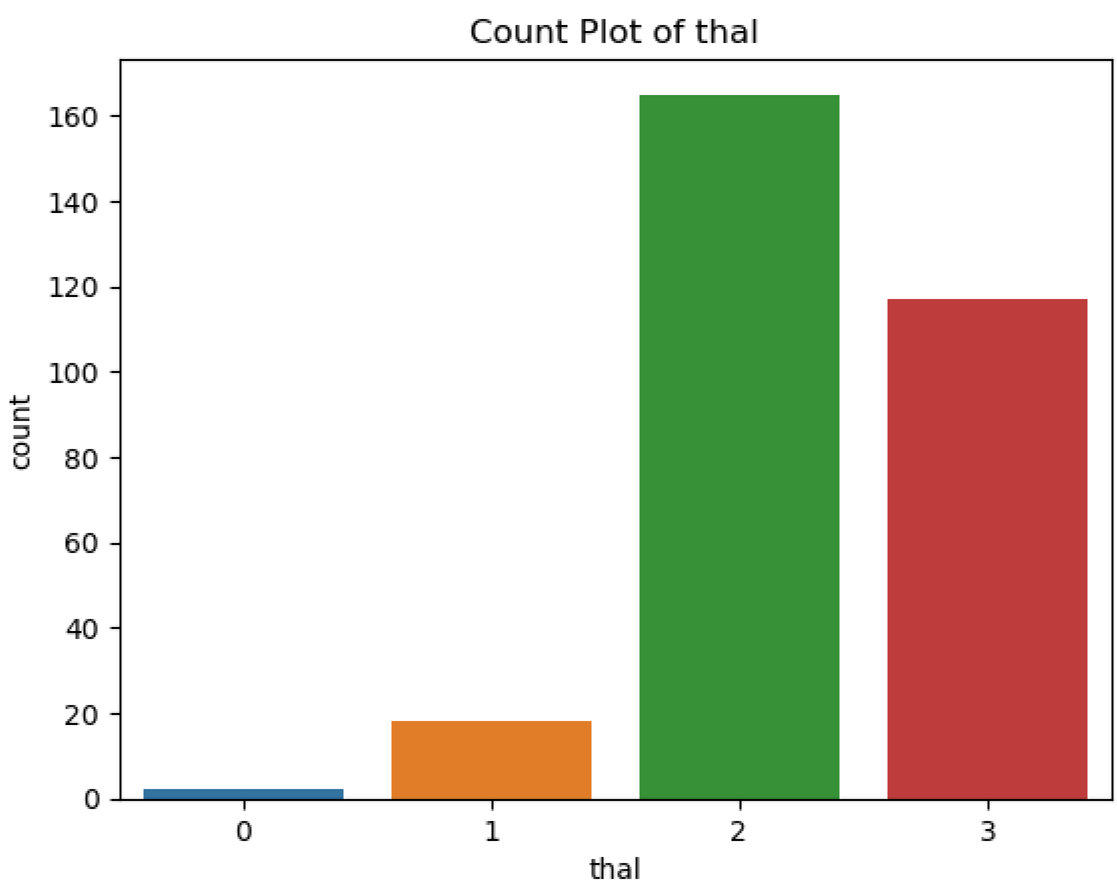

Count Plot of fbs
count
0
50
100
150
200
250
0
1
fbs

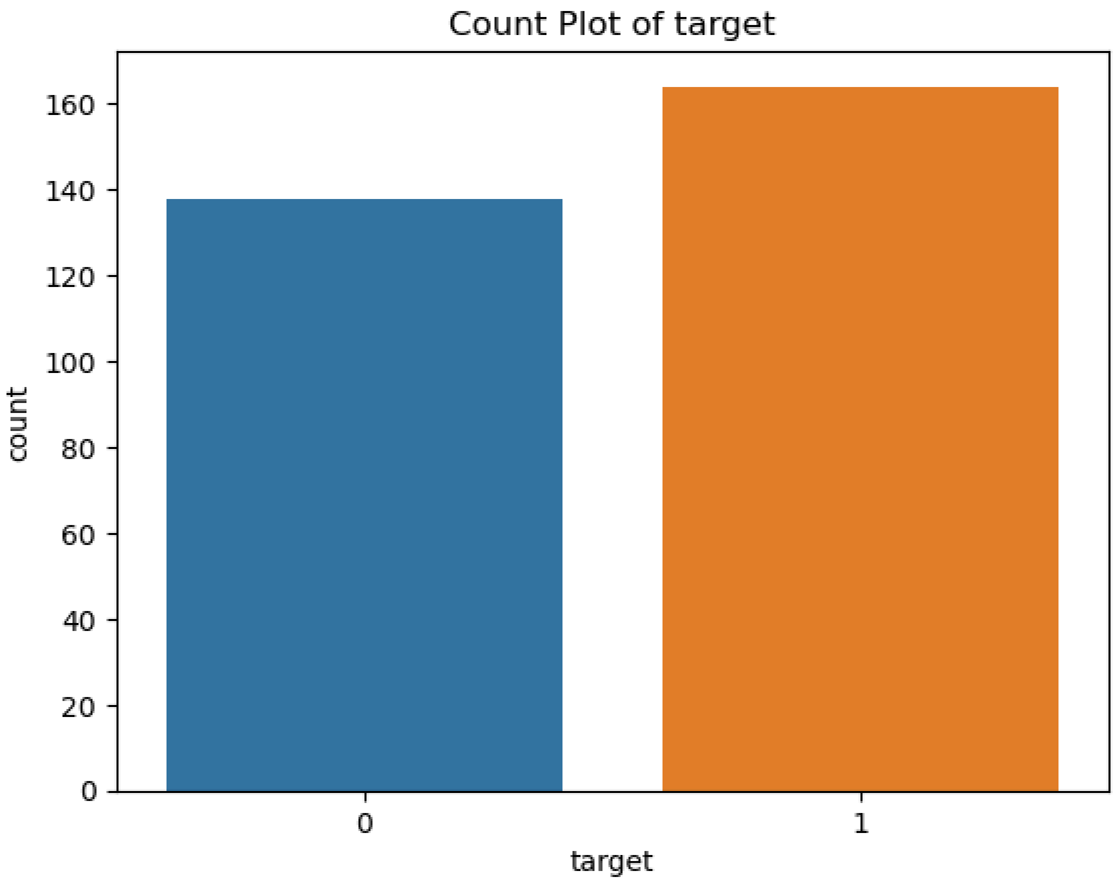

Count Plot of restecg
count
0
20
40
60
80
100
120
140
0
1
2
restecg


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


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

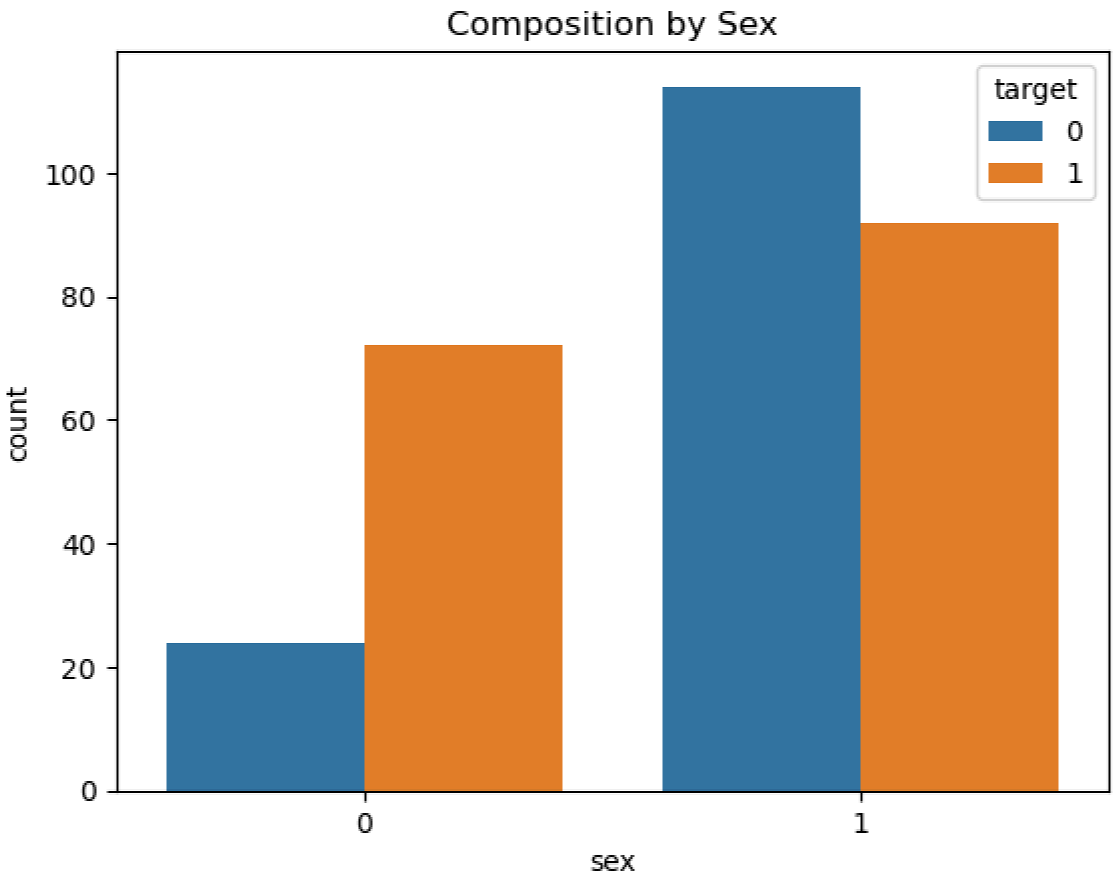

Count Plot of ca
count
175
150
125
100
75
50
25
0
0
1
2
3
4
ca


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



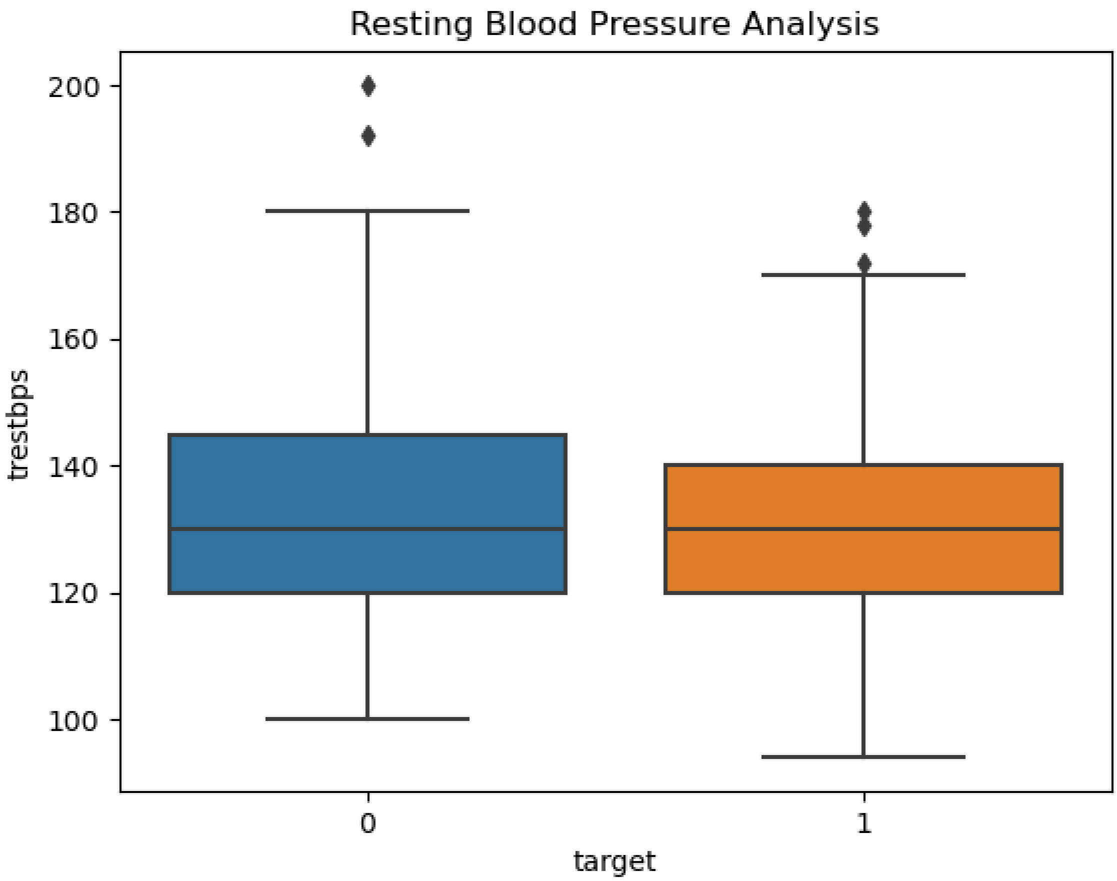
Composition of all patients with respect to Sex

In [ ]:
```
sns.countplot(x='sex', hue='target', data=df)
plt.title('Composition by Sex')
plt.show()
```

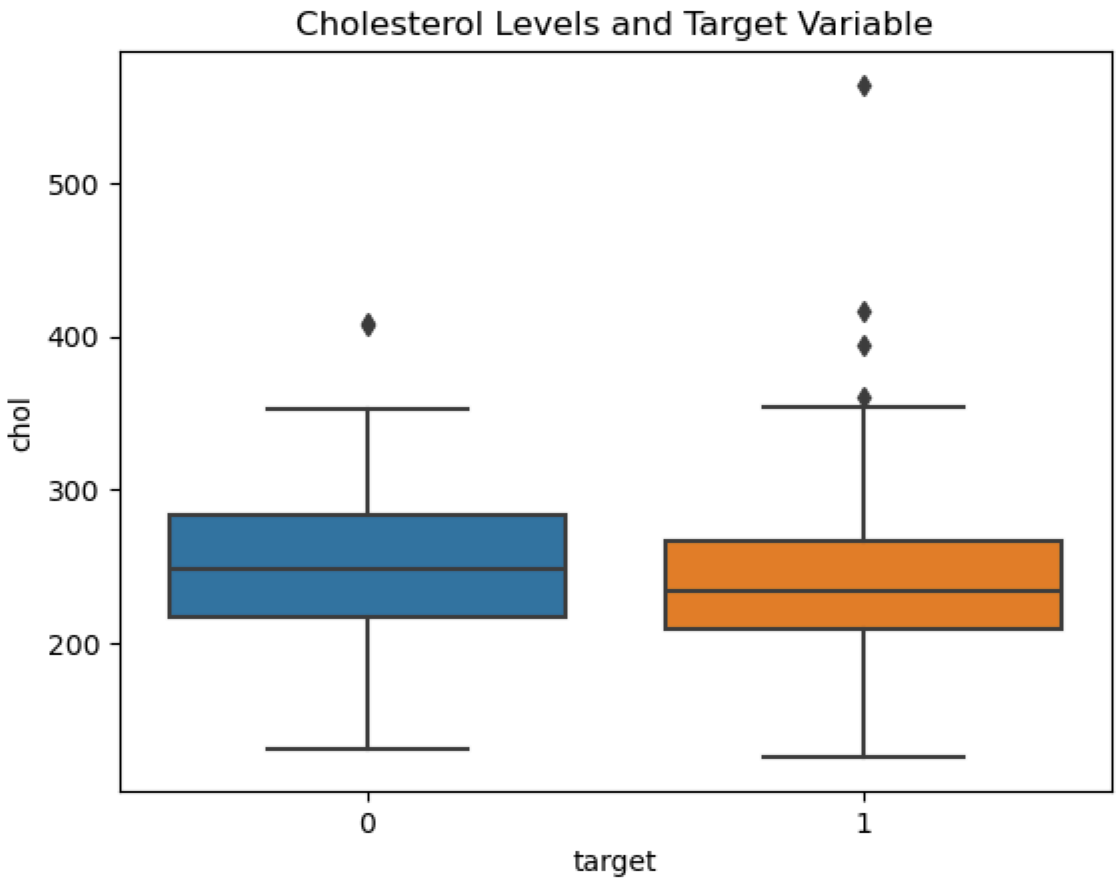


Sex vs. CVD

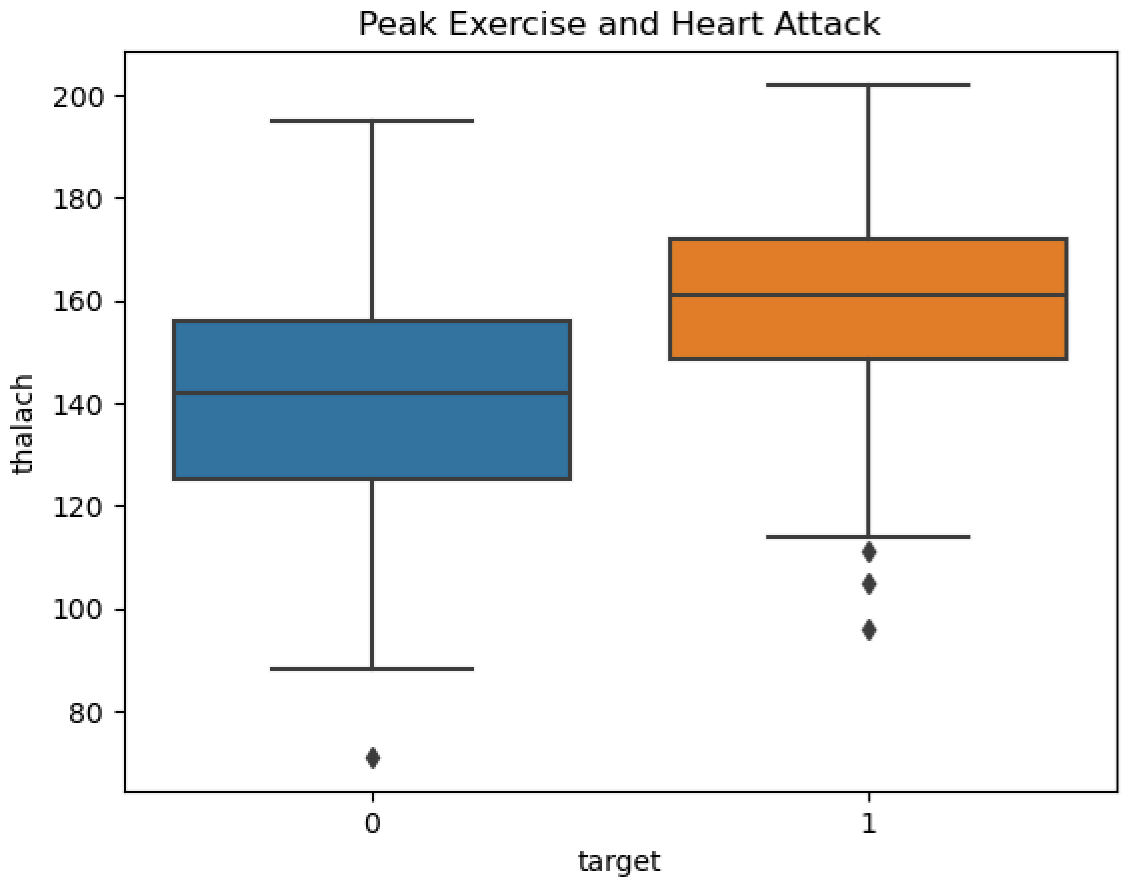
In [ ]:
```
sns.boxplot(x='target', y='trestbps', data=df)
plt.title('Resting Blood Pressure Analysis')
plt.show()
```



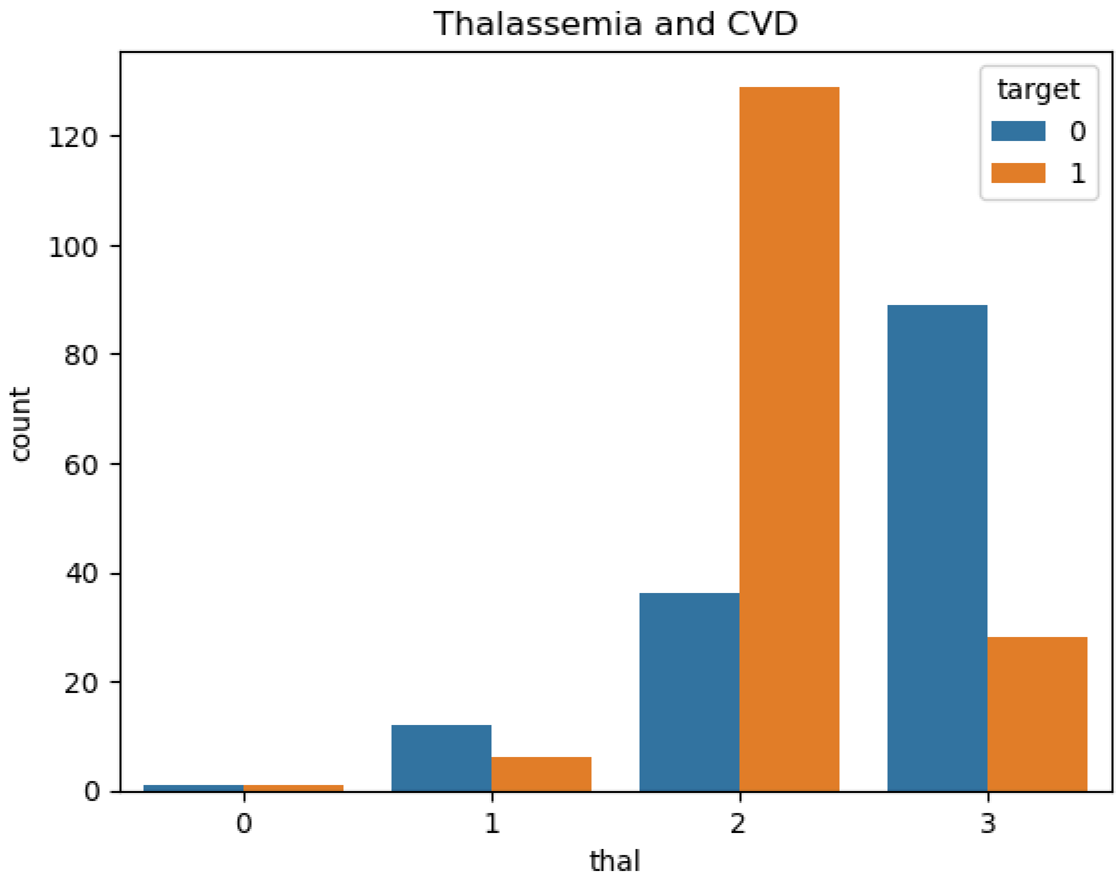
In [ ]:
```
sns.boxplot(x='target', y='chol', data=df)
plt.title('Cholesterol Levels and Target Variable')
plt.show()
```



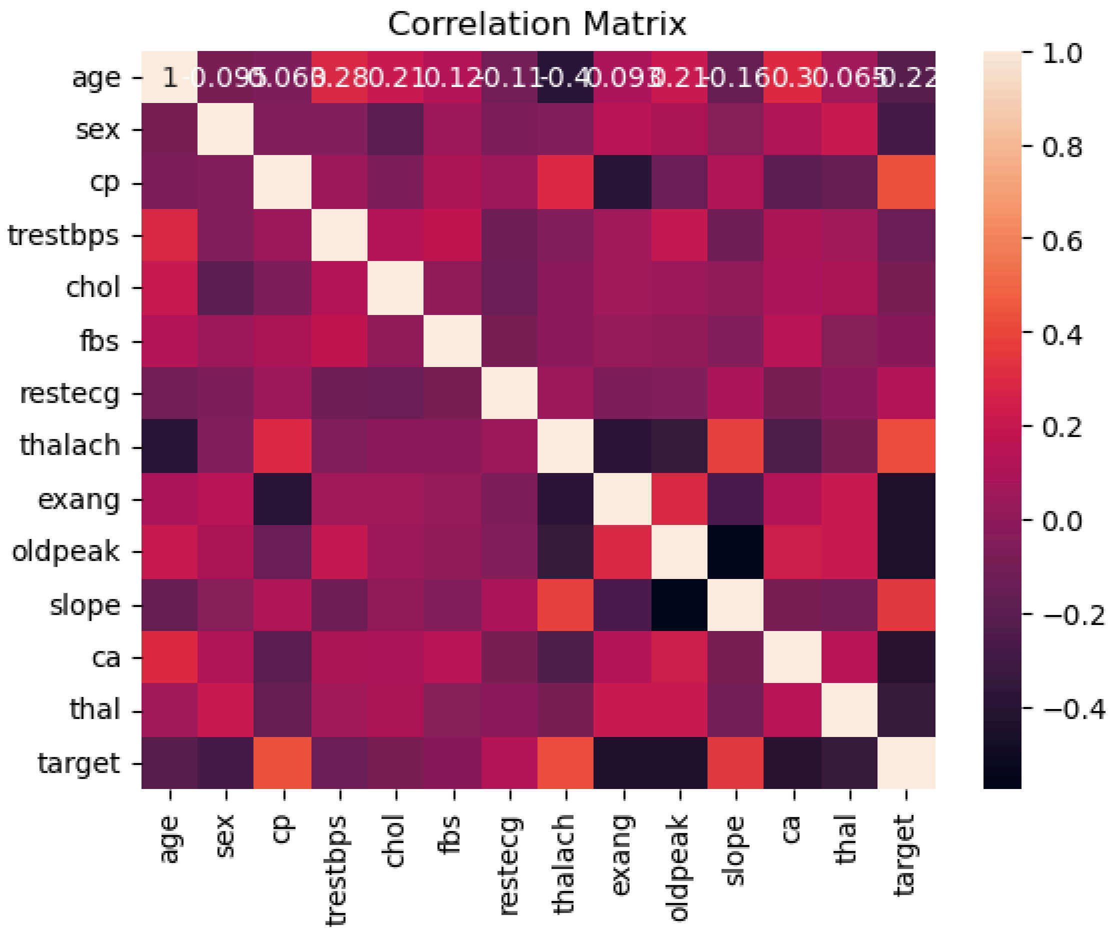
In [ ]:
```
sns.boxplot(x='target', y='thalach', data=df)
plt.title('Peak Exercise and Heart Attack')
plt.show()
```



In [ ]:
```
sns.countplot(x='thal', hue='target', data=df)
plt.title('Thalassemia and CVD')
plt.show()
```



In [ ]:
```
df_encoded = pd.get_dummies(df, drop_first=True)

# Calculate the correlation matrix after encoding
correlation_matrix = df_encoded.corr()
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation Matrix')
plt.show()

# Unstack the correlation matrix to get a long-form DataFrame
correlations = correlation_matrix.unstack()
```

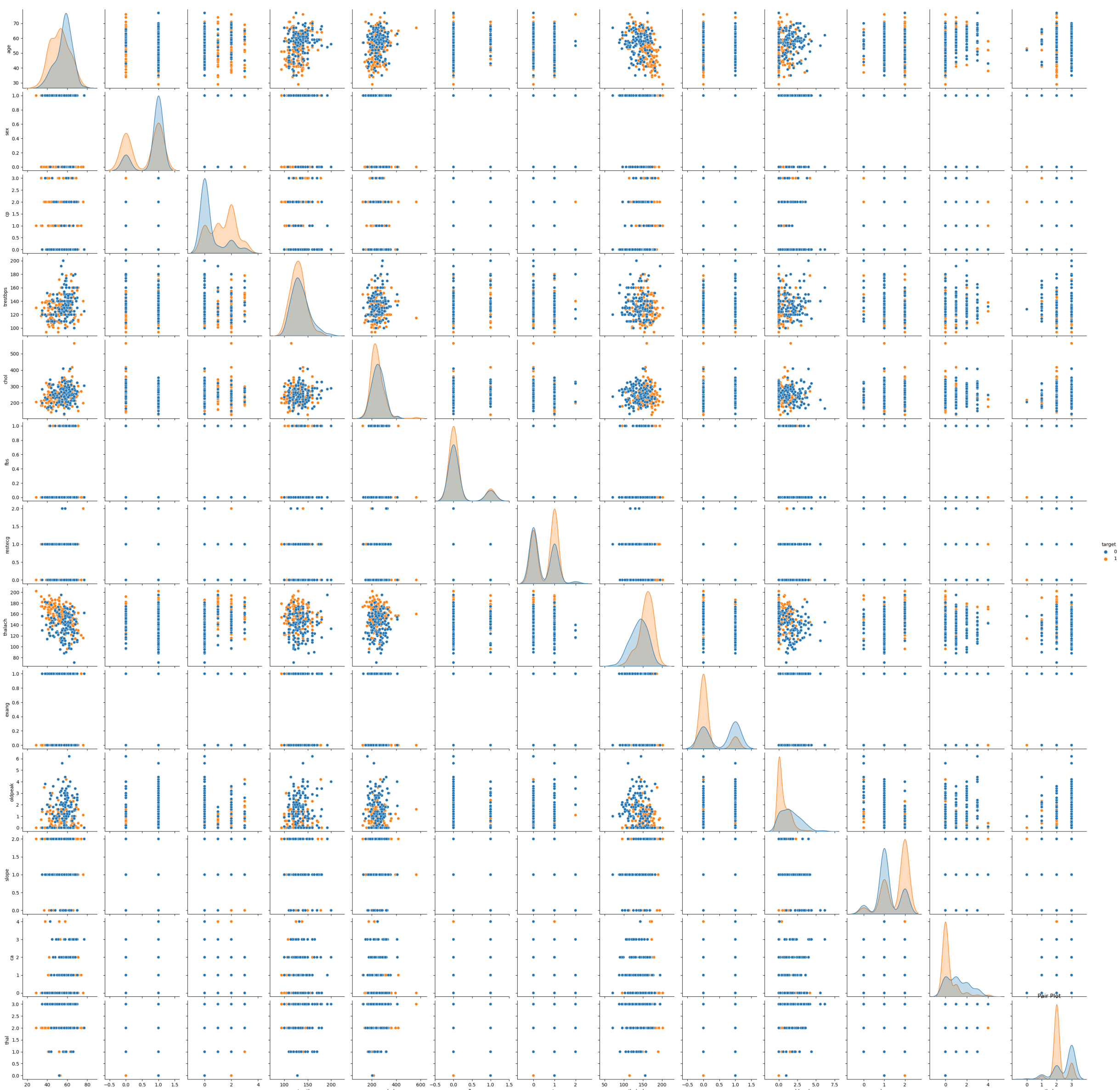


In [ ]:
```
# Unstack the correlation matrix to get a long-form DataFrame
correlations = correlation_matrix.unstack()

# Convert the Series to a DataFrame and reset index
correlations_df = pd.DataFrame(correlations, columns=['Correlation']).reset_index()

# Rename the columns for better readability
correlations_df.columns = ['Variable1', 'Variable2', 'Correlation']

# Filter out self-correlations and duplicate pairs
correlations_df = correlations_df[correlations_df['Variable1'] != correlations_df['
correlations_df = correlations_df.drop_duplicates(subset=['Correlation'], keep='fir

# Sort the correlations by absolute value in descending order
correlations_df['AbsCorrelation'] = correlations_df['Correlation'].abs()
correlations_df = correlations_df.sort_values(by='AbsCorrelation', ascending=False)

# Display the top 10 highest correlations
print(correlations_df.head(10))
```

	Variable1	Variable2	Correlation	AbsCorrelation
136	oldpeak	slope	-0.576314	0.576314
125	exang	target	-0.435601	0.435601
41	cp	target	0.432080	0.432080
139	oldpeak	target	-0.429146	0.429146
111	thalach	target	0.419955	0.419955
167	ca	target	-0.408992	0.408992
7	age	thalach	-0.395235	0.395235
36	cp	exang	-0.392937	0.392937
108	thalach	slope	0.384754	0.384754
106	thalach	exang	-0.377411	0.377411

SNS Pairplot

In [ ]:
```
sns.pairplot(df, hue='target')
plt.title('Pair Plot')
plt.show()
```



Interactive graphs of pairplots

In [ ]:
```
# Using Plotly Matrix
fig = px.scatter_matrix(df_encoded, dimensions=df_encoded.columns[:-1], color='targ
fig.update_layout(title='Pair Plot', width=1800, height=1800)  # Adjust width and h
fig.show()
```

In [ ]:
```
X = df_encoded.drop('target', axis=1)
y = df_encoded['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

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

# Predict and evaluate the model
y_pred_log_reg = log_reg.predict(X_test)
print(f'Logistic Regression Accuracy: {accuracy_score(y_test, y_pred_log_reg)}')
```

```
Logistic Regression Accuracy: 0.8524590163934426
```

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier

# Train a Random Forest model
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, y_train)

# Predict and evaluate the model
y_pred_rf = rf_clf.predict(X_test)
print(f'Random Forest Accuracy: {accuracy_score(y_test, y_pred_rf)}')
```

```
Random Forest Accuracy: 0.8852459016393442
```

In [ ]:
```
# Feature importance from Random Forest
importances = rf_clf.feature_importances_
features = X.columns
feature_importance_df = pd.DataFrame({'Feature': features, 'Importance': importance
print(feature_importance_df.sort_values(by='Importance', ascending=False))
```

```
     Feature  Importance
7    thalach    0.123748
2         cp    0.120569
11        ca    0.107202
9    oldpeak    0.106114
12      thal    0.104024
0        age    0.089463
4       chol    0.085714
3   trestbps    0.079081
8      exang    0.059152
10     slope    0.051937
1        sex    0.043754
6    restecg    0.021719
5        fbs    0.007522
```

In [ ]:
```
# Logistic Regression with statsmodels for p-values
import statsmodels.api as sm

log_reg_model = sm.Logit(y_train, X_train)
```

```
result = log_reg_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.351487
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                  241
Model:                          Logit   Df Residuals:                      228
Method:                           MLE   Df Model:                           12
Date:                Sun, 09 Jun 2024   Pseudo R-squ.:                  0.4896
Time:                        20:51:38   Log-Likelihood:                -84.708
converged:                       True   LL-Null:                       -165.95
Covariance Type:            nonrobust   LLR p-value:                 1.639e-28
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
age            0.0224      0.021      1.053      0.292      -0.019       0.064
sex           -1.7130      0.511     -3.353      0.001      -2.714      -0.712
cp             0.6956      0.202      3.452      0.001       0.301       1.091
trestbps      -0.0233      0.011     -2.120      0.034      -0.045      -0.002
chol          -0.0037      0.004     -0.879      0.380      -0.012       0.005
fbs            0.4790      0.635      0.754      0.451      -0.766       1.724
restecg        0.6161      0.388      1.587      0.113      -0.145       1.377
thalach        0.0352      0.010      3.686      0.000       0.016       0.054
exang         -0.9877      0.453     -2.182      0.029      -1.875      -0.100
oldpeak       -0.4177      0.246     -1.700      0.089      -0.899       0.064
slope          0.9236      0.387      2.388      0.017       0.166       1.682
ca            -0.8784      0.243     -3.608      0.000      -1.356      -0.401
thal          -1.0242      0.343     -2.983      0.003      -1.697      -0.351
==============================================================================
```