In [136…

```python
import pandas as pd
import numpy as np
```

In [137…

```python
a=pd.read_csv('heartiidt.csv')
```

In [138…

```python
a
```

Out[138]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

1025 rows × 14 columns

In [139…

```python
a.head()
```

Out[139]:

	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 [140…

```python
a.tail()
```

Out[140]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

In [141…

```python
a.describe()
```

Out[141]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	t
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	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	0.754146	2.323902	0.51
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053	0.617755	1.030798	0.620660	0.50
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000	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	0.000000	2.000000	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	0.000000	2.000000	1.00
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000	2.000000	1.000000	3.000000	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	4.000000	3.000000	1.00

In [142…

```python
a.isna().sum()
```

Out[142]:

```
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 [143…

```python
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [144…

```python
a['target'].value_counts()
```

Out[144]:

```
1    526
0    499
Name: target, dtype: int64
```

In [145…

```python
heart_disease_male = a[(a['target'] == 1) & (a['sex'] == 1)]
print('number of male with heart disease:')
heart_disease_male.shape[0]
```

```
number of male with heart disease:
```

Out[145]: 300

In [146…

```python
heart_disease_female = a[(a['target'] == 1) & (a['sex'] == 0)]
print('number of female with heart disease:')
heart_disease_female.shape[0]
```

```
number of female with heart disease:
```

Out[146]: 226

In [147…

```python
heart_disease_male1 = a[(a['target'] == 0) & (a['sex'] == 1)]
print('number of male with no heart disease:')
heart_disease_male1.shape[0]
```

```
number of male with no heart disease:
```

Out[147]: 413

In [148…

```python
heart_disease_female1 = a[(a['target'] == 0) & (a['sex'] == 0)]
print('number of female with no heart disease:')
heart_disease_female1.shape[0]
```

```
number of female with no heart disease:
```

Out[148]: 86

In [149…

```python
heart_disease_data = a[a['target'] == 1]
count_under_30 = heart_disease_data[heart_disease_data['age'] < 30].shape[0]
count_30_and_over = heart_disease_data[heart_disease_data['age'] >= 30].shape[0]
```

In [150…

```python
print('people with heart disease under age 30:')
count_under_30
```

```
people with heart disease under age 30:
```

Out[150]: 4

In [151…

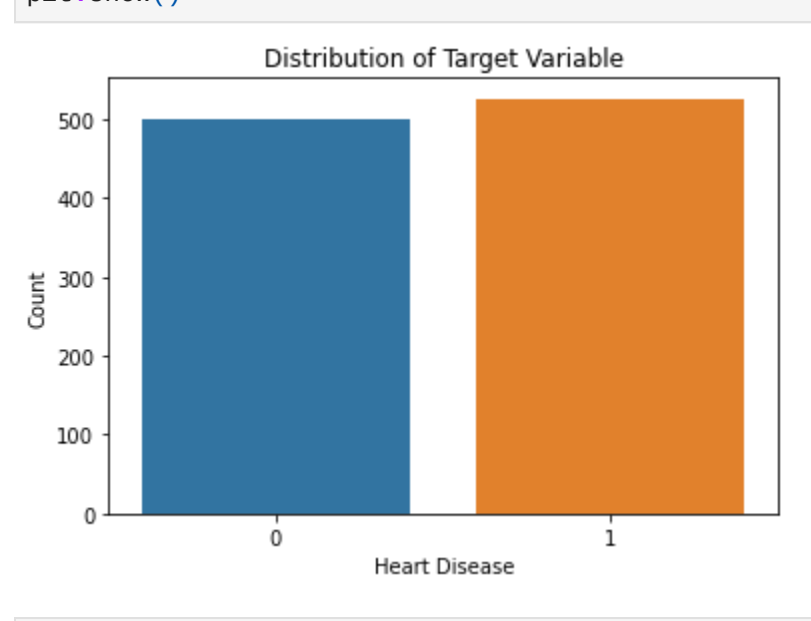
```python
print('people with heart disease above age 30:')
count_30_and_over
```

```
people with heart disease above age 30:
```

Out[151]: 522

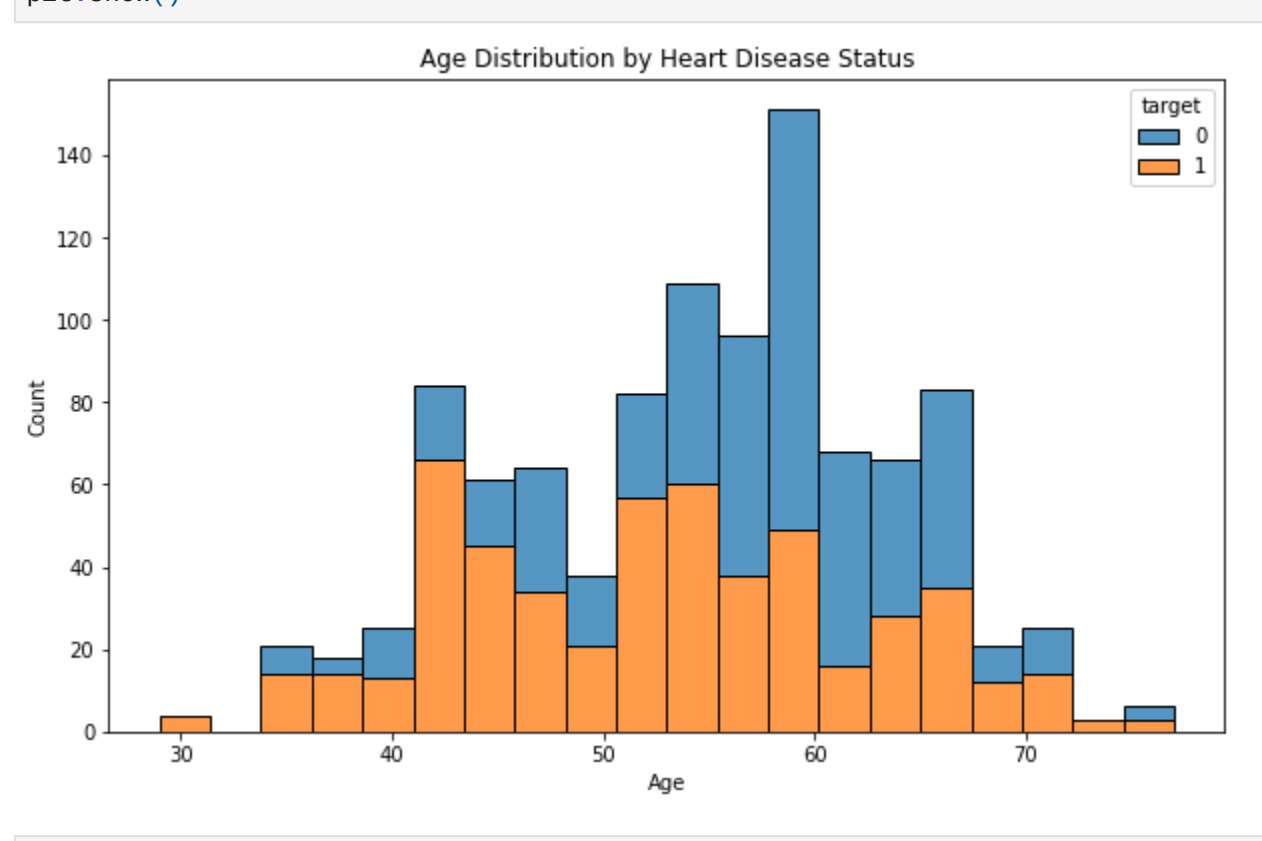
In [152…

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Distribution of Target Variable
plt.figure(figsize=(6,4))
sns.countplot(x='target', data=a)
plt.title('Distribution of Target Variable')
plt.xlabel('Heart Disease')
plt.ylabel('Count')
plt.show()
```



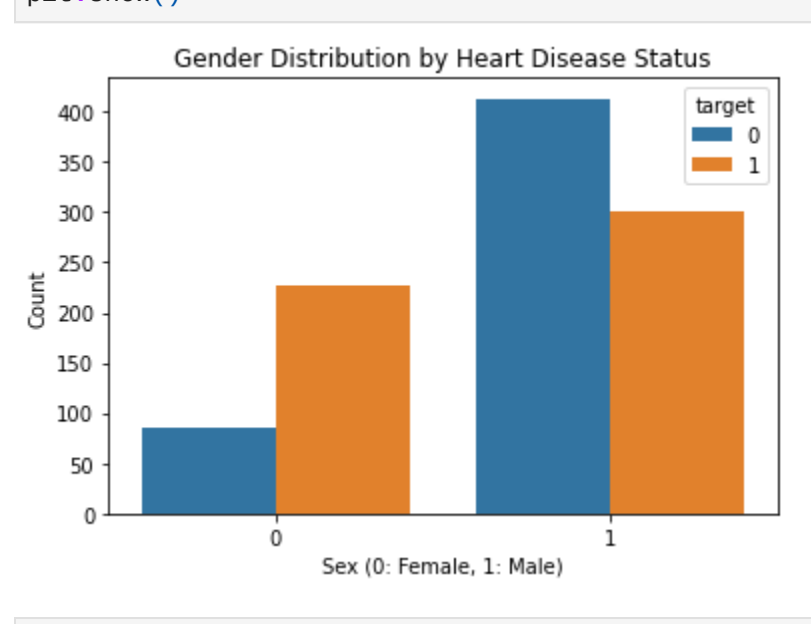
In [153…

```python
#Age Distribution by Target
plt.figure(figsize=(10,6))
sns.histplot(data=a, x='age', hue='target', multiple='stack', bins=20)
plt.title('Age Distribution by Heart Disease Status')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



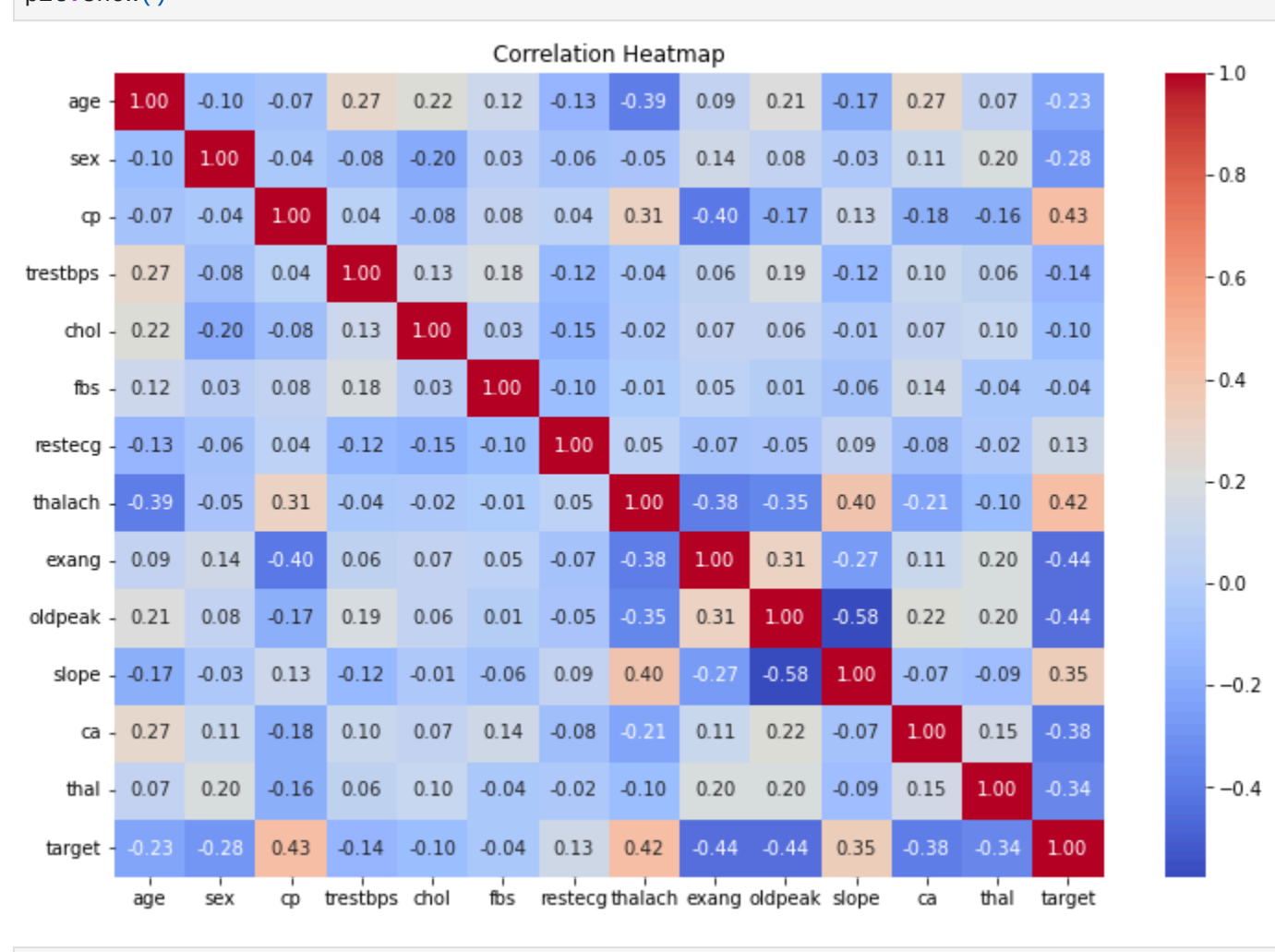
In [154…

```python
# Gender Distribution by Target
plt.figure(figsize=(6,4))
sns.countplot(x='sex', hue='target', data=a)
plt.title('Gender Distribution by Heart Disease Status')
plt.xlabel('Sex (0: Female, 1: Male)')
plt.ylabel('Count')
plt.show()
```



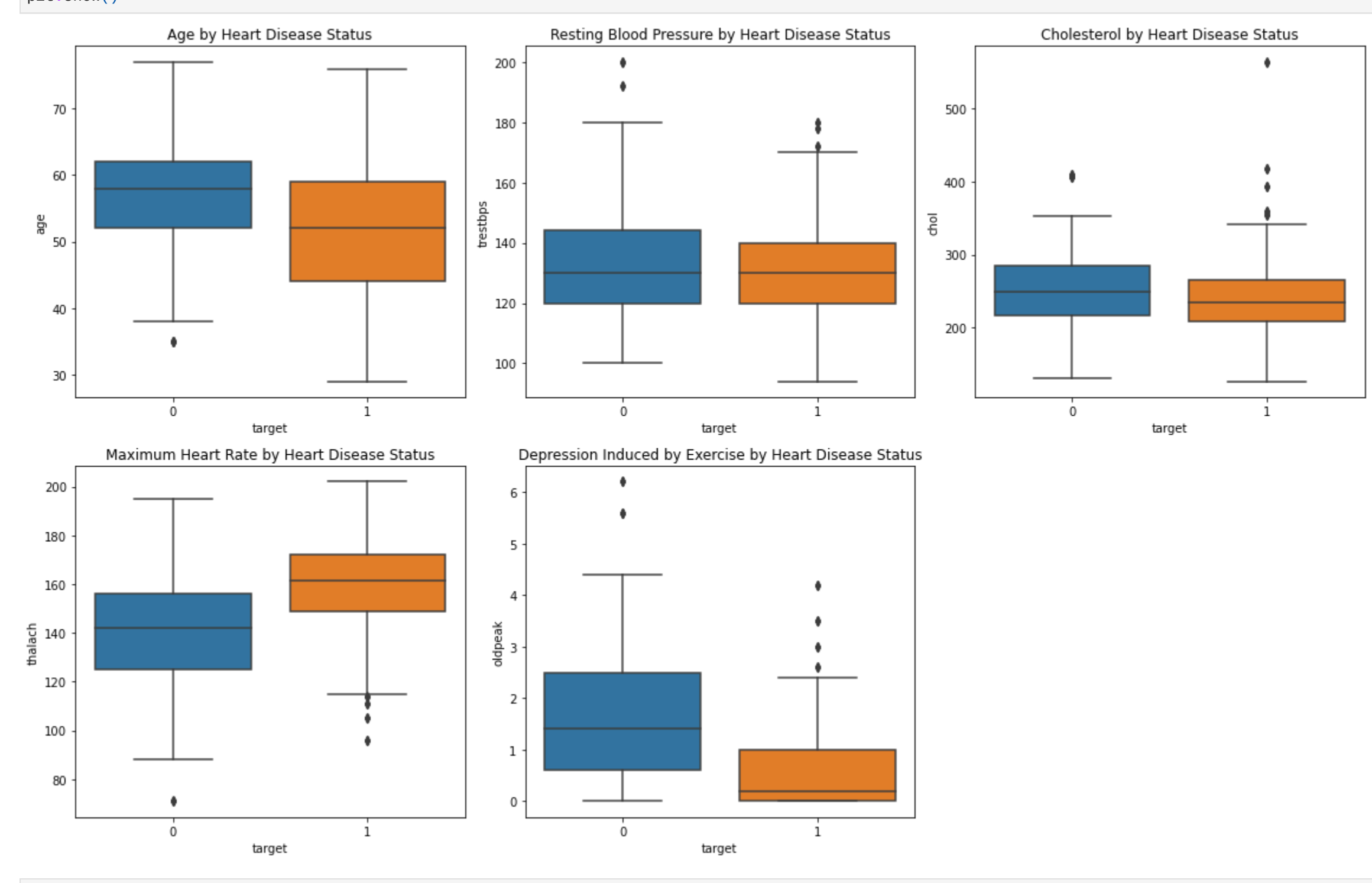
In [155…

```python
# Correlation Heatmap
plt.figure(figsize=(12,8))
corr = a.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



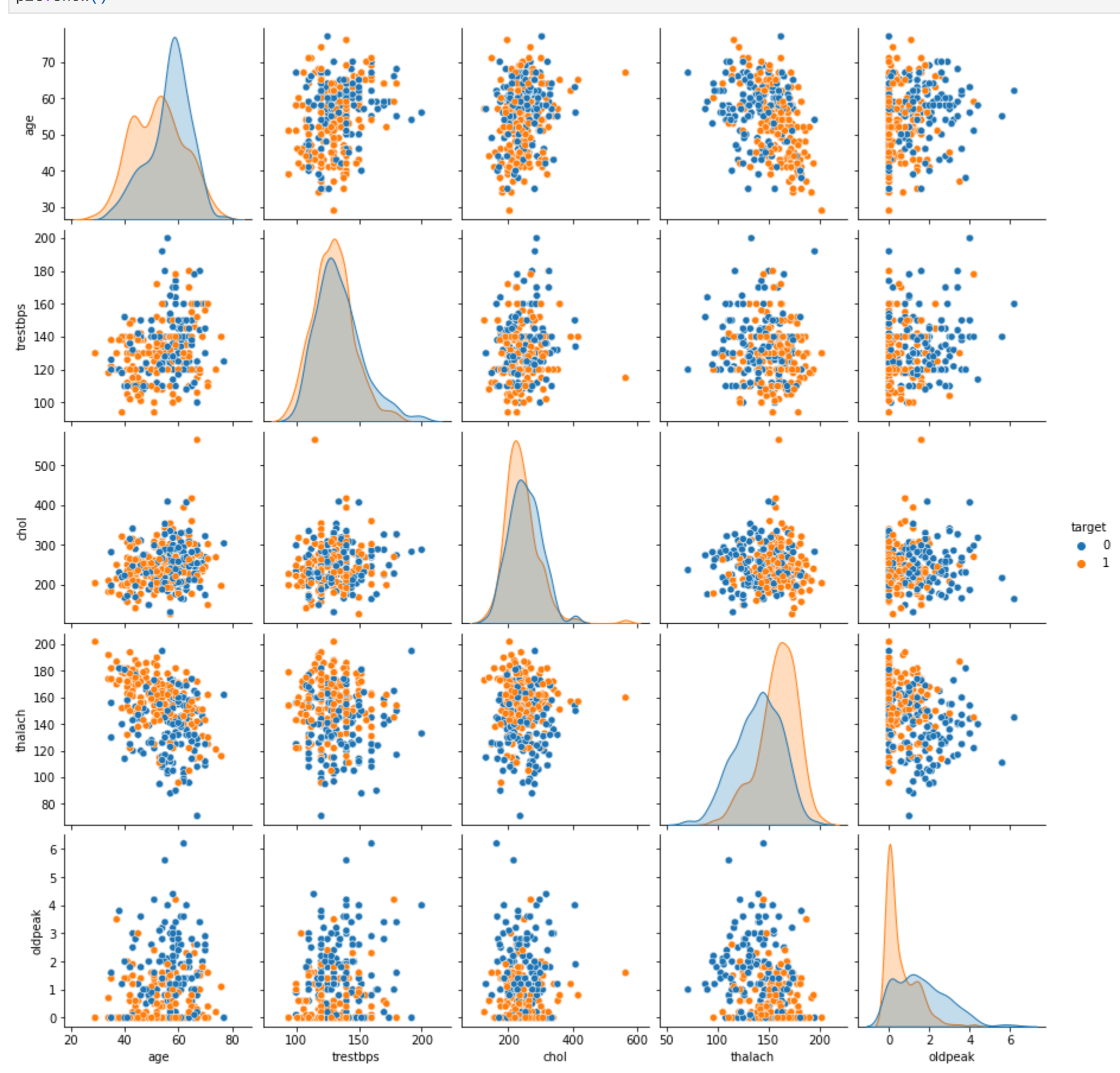
In [156…

```python
# Boxplots for Numeric Variables by Target
plt.figure(figsize=(16,10))

# Boxplot for 'age'
plt.subplot(2, 3, 1)
sns.boxplot(x='target', y='age', data=a)
plt.title('Age by Heart Disease Status')

# Boxplot for 'trestbps'
plt.subplot(2, 3, 2)
sns.boxplot(x='target', y='trestbps', data=a)
plt.title('Resting Blood Pressure by Heart Disease Status')

# Boxplot for 'chol'
plt.subplot(2, 3, 3)
sns.boxplot(x='target', y='chol', data=a)
plt.title('Cholesterol by Heart Disease Status')

# Boxplot for 'thalach'
plt.subplot(2, 3, 4)
sns.boxplot(x='target', y='thalach', data=a)
plt.title('Maximum Heart Rate by Heart Disease Status')

# Boxplot for 'oldpeak'
plt.subplot(2, 3, 5)
sns.boxplot(x='target', y='oldpeak', data=a)
plt.title('Depression Induced by Exercise by Heart Disease Status')

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



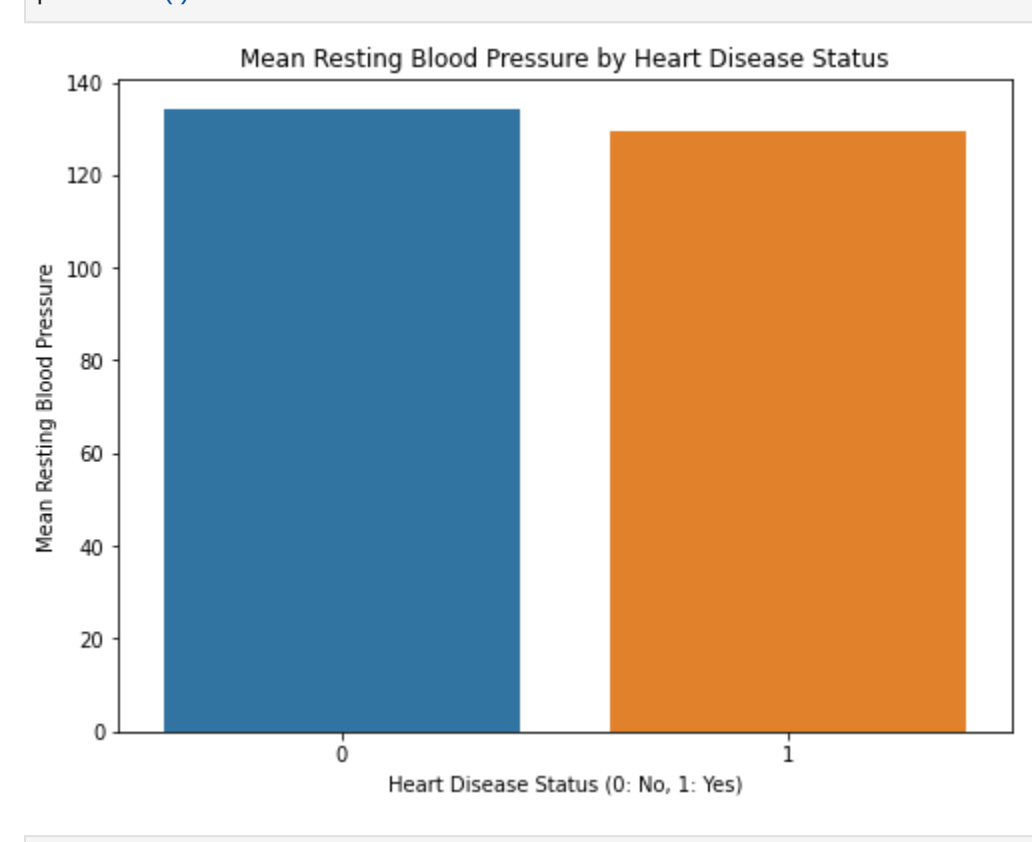
In [157…

```python
selected_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target']
sns.pairplot(a[selected_features], hue='target', diag_kind='kde')
plt.show()
```



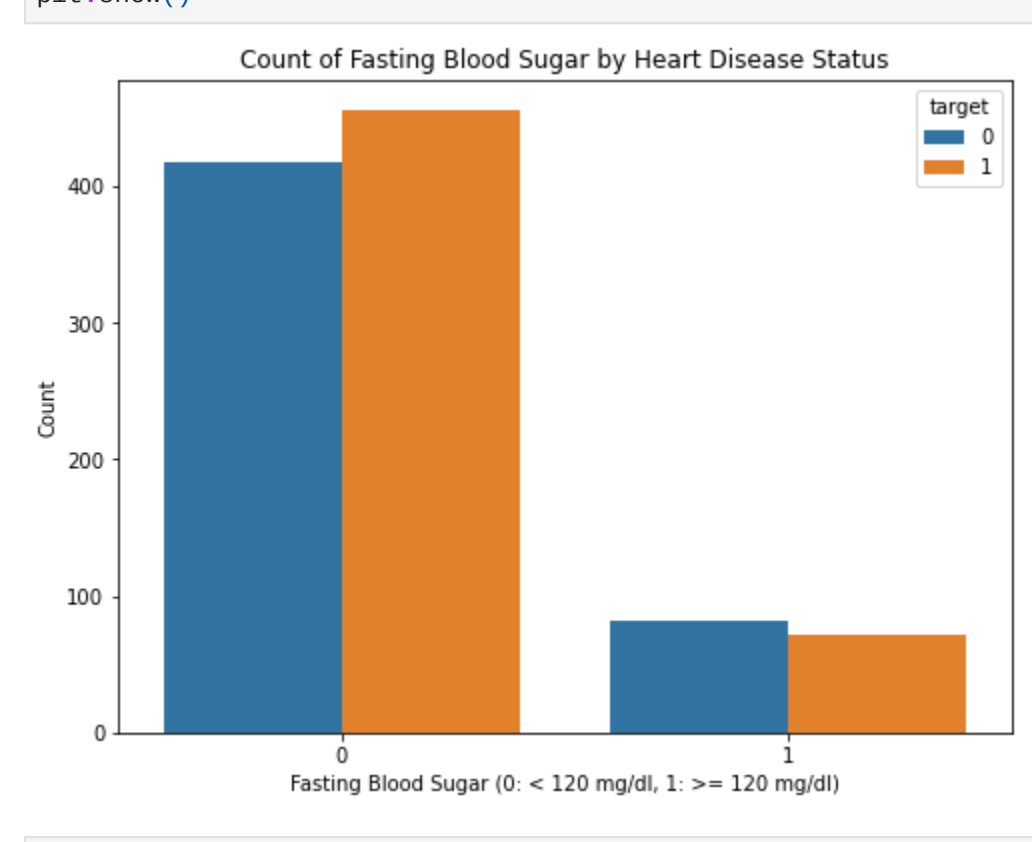
In [158…

```python
# Calculate the mean resting blood pressure for each target category
mean_trestbps = a.groupby('target')['trestbps'].mean().reset_index()

plt.figure(figsize=(8, 6))
sns.barplot(x='target', y='trestbps', data=mean_trestbps)
plt.title('Mean Resting Blood Pressure by Heart Disease Status')
plt.xlabel('Heart Disease Status (0: No, 1: Yes)')
plt.ylabel('Mean Resting Blood Pressure')
plt.show()
```



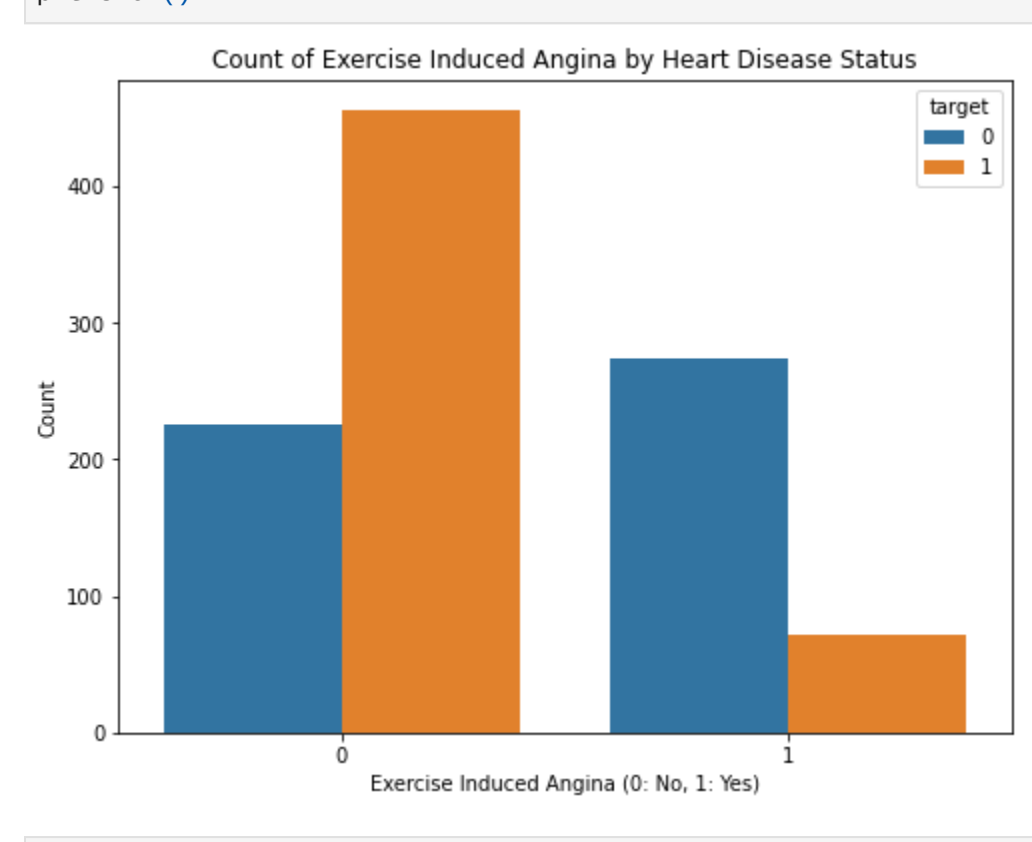
In [159…

```python
fbs_target_counts = a.groupby(['target', 'fbs']).size().reset_index(name='count')

plt.figure(figsize=(8, 6))
sns.barplot(x='fbs', y='count', hue='target', data=fbs_target_counts)
plt.title('Count of Fasting Blood Sugar by Heart Disease Status')
plt.xlabel('Fasting Blood Sugar (0: < 120 mg/dl, 1: >= 120 mg/dl)')
plt.ylabel('Count')
plt.show()
```



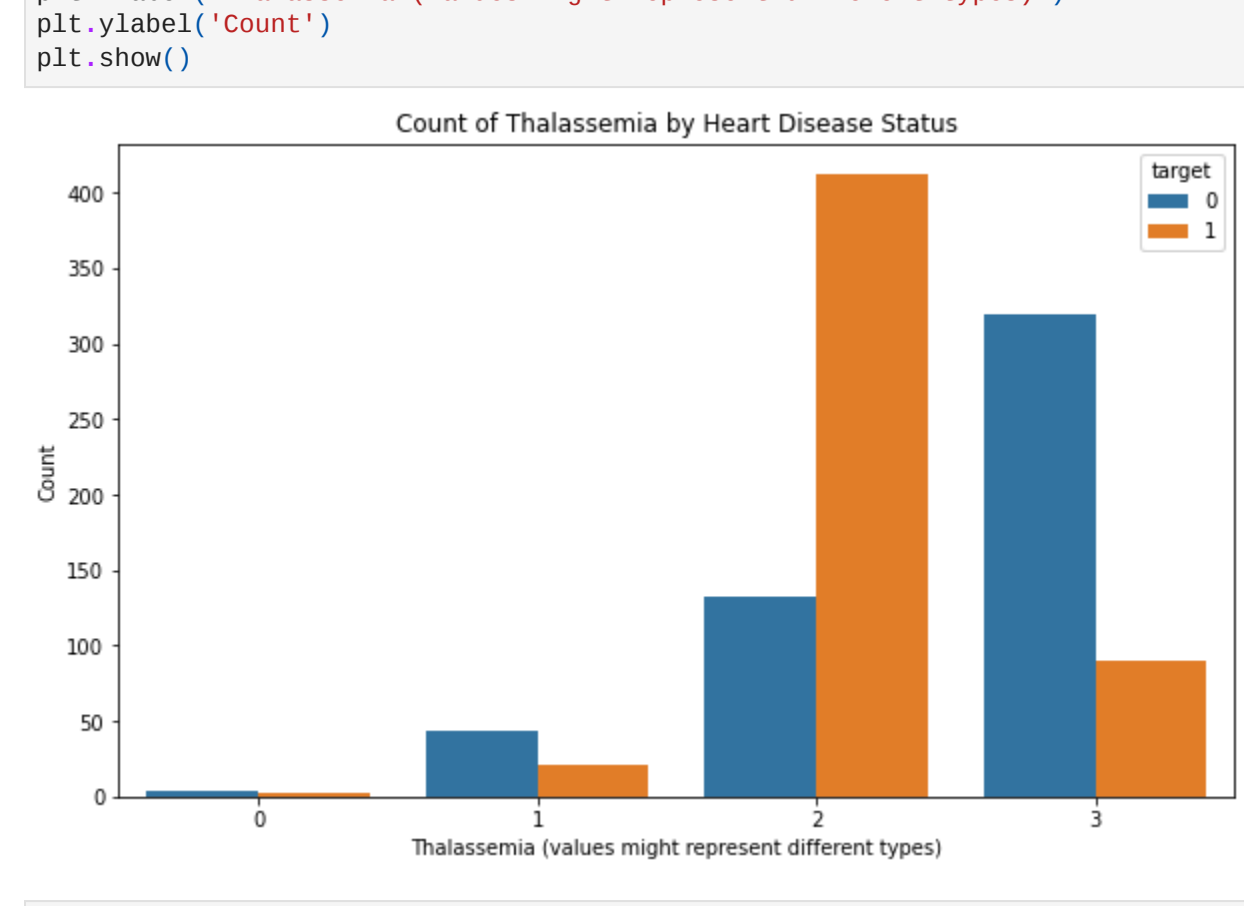
In [160…

```python
exang_target_counts = a.groupby(['target', 'exang']).size().reset_index(name='count')

plt.figure(figsize=(8, 6))
sns.barplot(x='exang', y='count', hue='target', data=exang_target_counts)
plt.title('Count of Exercise Induced Angina by Heart Disease Status')
plt.xlabel('Exercise Induced Angina (0: No, 1: Yes)')
plt.ylabel('Count')
plt.show()
```



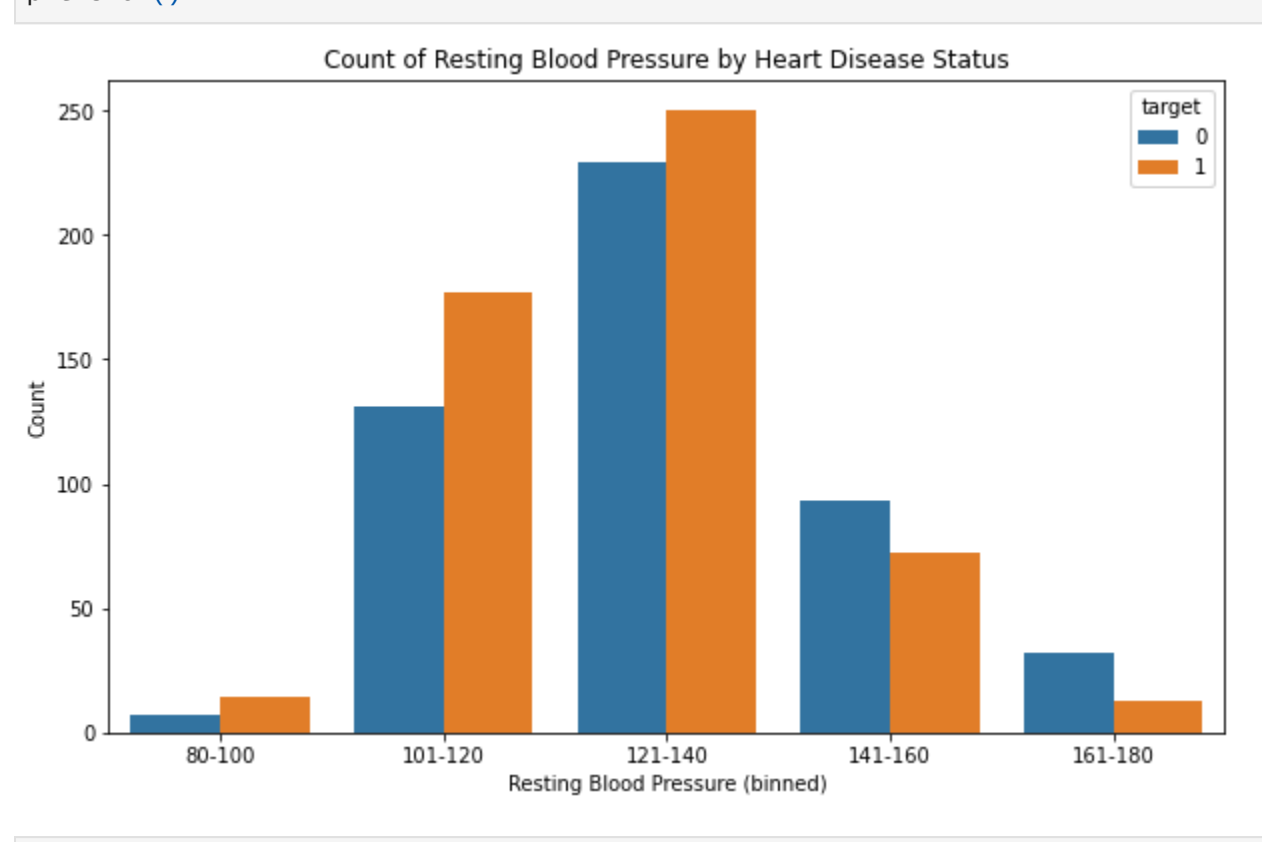
In [161…

```python
# Create a DataFrame for the counts of `thal` values by `target`
thal_target_counts = a.groupby(['target', 'thal']).size().reset_index(name='count')

plt.figure(figsize=(10, 6))
sns.barplot(x='thal', y='count', hue='target', data=thal_target_counts)
plt.title('Count of Thalassemia by Heart Disease Status')
plt.xlabel('Thalassemia (values might represent different types)')
plt.ylabel('Count')
plt.show()
```



In [162…

```python
# Bin `trestbps` into ranges
a['trestbps_bin'] = pd.cut(a['trestbps'], bins=[80, 100, 120, 140, 160, 180],
                           labels=['80-100', '101-120', '121-140', '141-160', '161-180'])

# Create a DataFrame for the counts of `trestbps_bin` values by `target`
trestbps_target_counts = a.groupby(['target', 'trestbps_bin']).size().reset_index(name='count')

plt.figure(figsize=(10, 6))
sns.barplot(x='trestbps_bin', y='count', hue='target', data=trestbps_target_counts)
plt.title('Count of Resting Blood Pressure by Heart Disease Status')
plt.xlabel('Resting Blood Pressure (binned)')
plt.ylabel('Count')
plt.show()
```



In [163…

```python
X=a.drop(columns=['target','trestbps_bin'],axis=0)
Y=a['target']
```

In [164…

```python
X.head()
```

Out[164]:

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

In [165…

```python
Y.head()
```

Out[165]:

```
0    0
1    0
2    0
3    0
4    0
Name: target, dtype: int64
```

In [166…

```python
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix

# Sample data preparation (assuming X and Y are already defined)
# X = feature matrix
# Y = target vector

# Split the data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and fit the Logistic Regression model
lg = LogisticRegression(max_iter=1000)
lg.fit(X_train_scaled, Y_train)

# Make predictions and evaluate the model
Y_pred = lg.predict(X_test_scaled)

# Print accuracy
print(f"Accuracy: {accuracy_score(Y_test, Y_pred)}")

# Print classification report for precision, recall, and F1-score
print("\nClassification Report:")
print(classification_report(Y_test, Y_pred))

# Print confusion matrix
print("\nConfusion Matrix:")
print(confusion_matrix(Y_test, Y_pred))

# Calculate precision, recall, and F1 score manually if needed
precision = precision_score(Y_test, Y_pred)
recall = recall_score(Y_test, Y_pred)
f1 = f1_score(Y_test, Y_pred)

print(f"\nPrecision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Accuracy: 0.8439024390243902

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.77      0.84       105
           1       0.79      0.92      0.85       100

    accuracy                           0.84       205
   macro avg       0.85      0.85      0.84       205
weighted avg       0.85      0.84      0.84       205


Confusion Matrix:
[[81 24]
 [ 8 92]]

Precision: 0.7931034482758621
Recall: 0.92
F1 Score: 0.8518518518518519
```

In [167…

```python
import warnings

# Suppress specific warnings
warnings.filterwarnings("ignore", category=UserWarning, module='sklearn')

# Define new data for prediction
data_input = np.array([52,1,0,125,212,0,1,168,0,1.0,2,2,3]).reshape(1, -1)

# Scale the new data using the same scaler
data_input_scaled = scaler.transform(data_input)

# Making prediction
prediction = lg.predict(data_input_scaled)
print(f"\nPrediction: {prediction[0]}")

if prediction[0] == 0:
    print('The Person does not have a heart disease')
else:
    print('The Person has a heart disease')
```

```
Prediction: 0
The Person does not have a heart disease
```

In [ ]:

In [ ]: