Heart Disease Extensive Analysis + visualization with Python

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
%matplotlib inline
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv('heart.csv')
```

In [4]:
```
# print the shape
print('The shape of the dataset: ',df.shape)
```

```
The shape of the dataset:  (303, 14)
```

In [5]:
```
df.head()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
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
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

In [6]:
```
# summary of dataset
df.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]:
```
df.dtypes
```

Out[7]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

In [8]:
```
# statistical properties of dataset
df.describe()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	reste
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.5280
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.5258
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.0000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.0000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.0000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.0000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.0000

In [9]:
```
df.columns
```

Out[9]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [10]:
```
# Univariate Analysis

df['target'].nunique()
```

Out[10]:
```
2
```

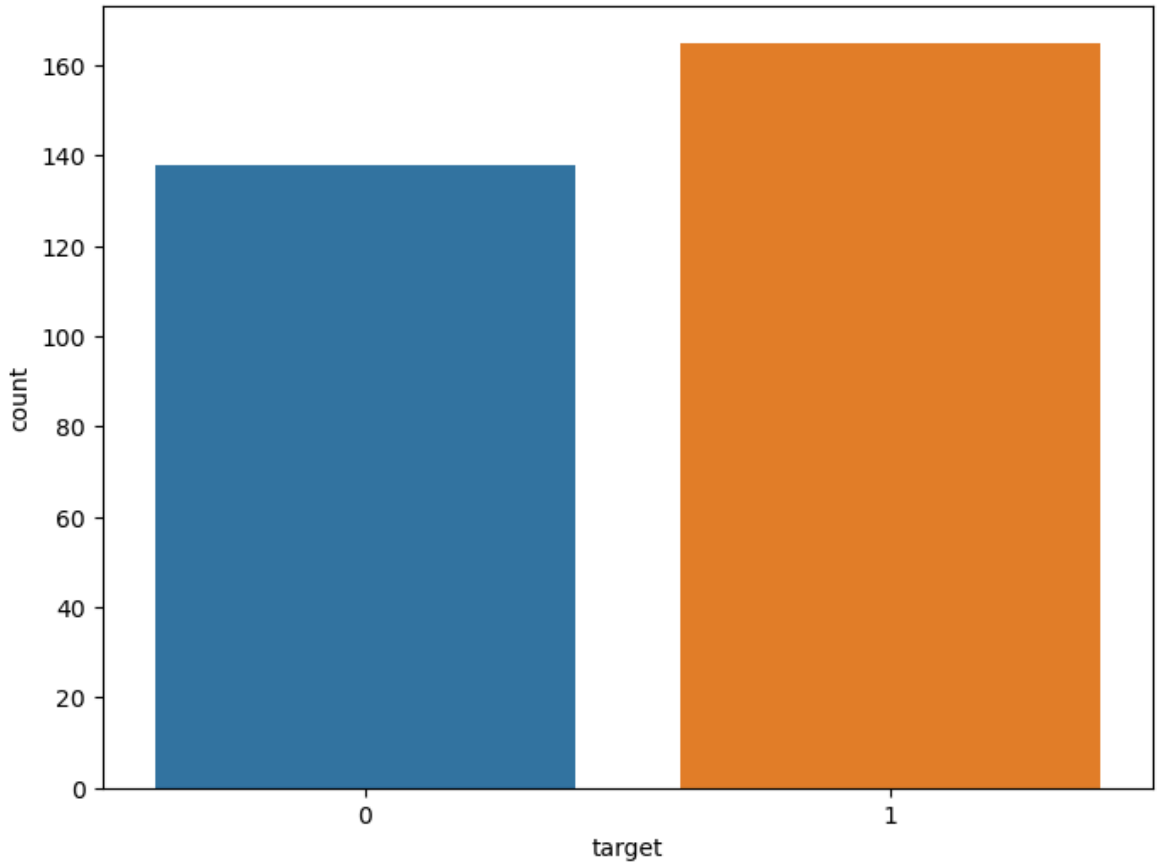
In [11]:
```
df['target'].unique()   # view the unique values in target variable
```

Out[11]:
```
array([1, 0], dtype=int64)
```

In [12]:
```
# frequency distribution of target variable

df['target'].value_counts()
```

Out[12]:
```
target
1    165
0    138
Name: count, dtype: int64
```

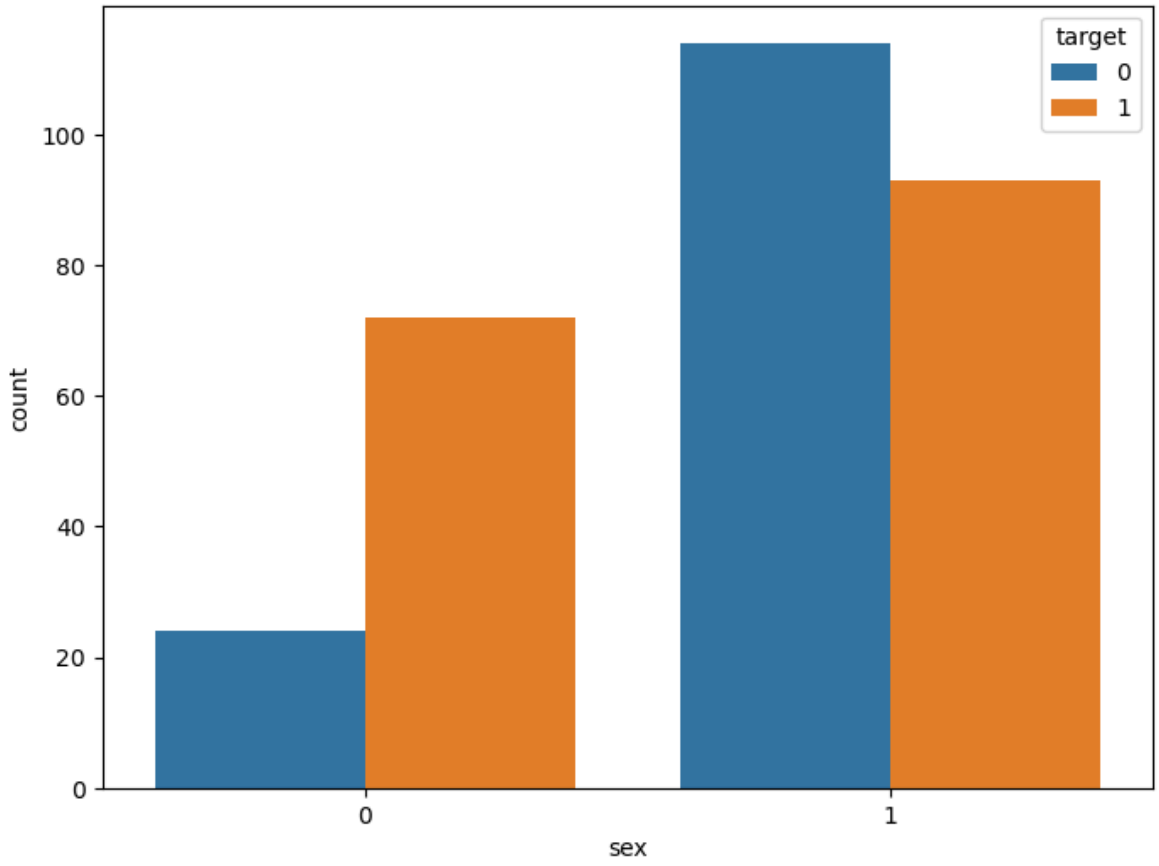
In [13]:
```
# Visualize frequency distribtuion of target variable

f,ax = plt.subplots(figsize=(8,6))
ax = sns.countplot(x="target",data=df)
plt.show()
```



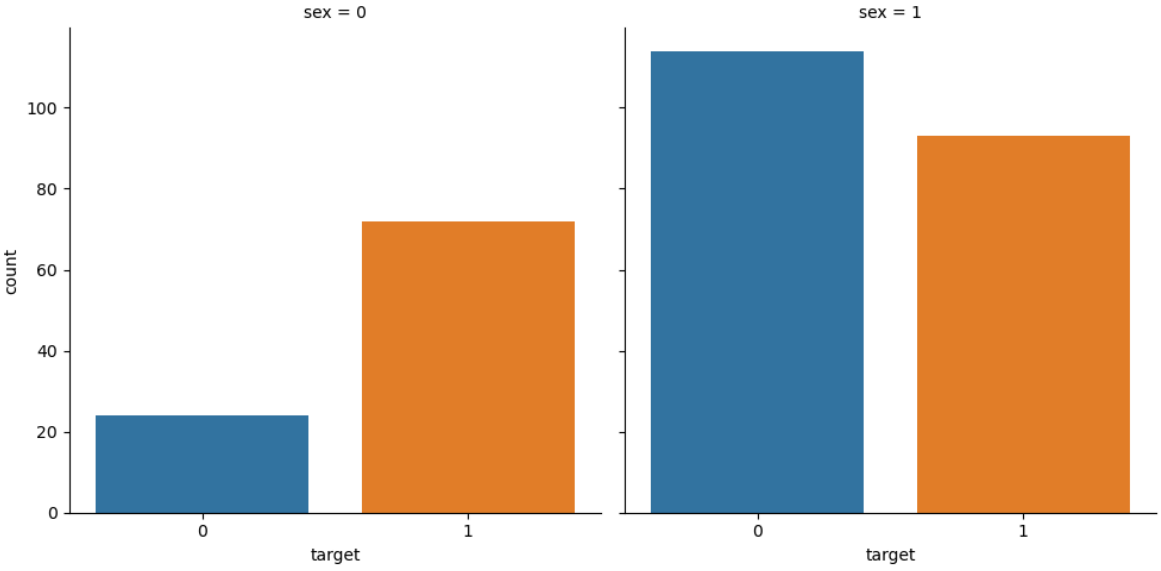
In [14]:
```
# Frequency distribution of target variable wrt sex

df.groupby('sex')['target'].value_counts()
```

Out[14]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

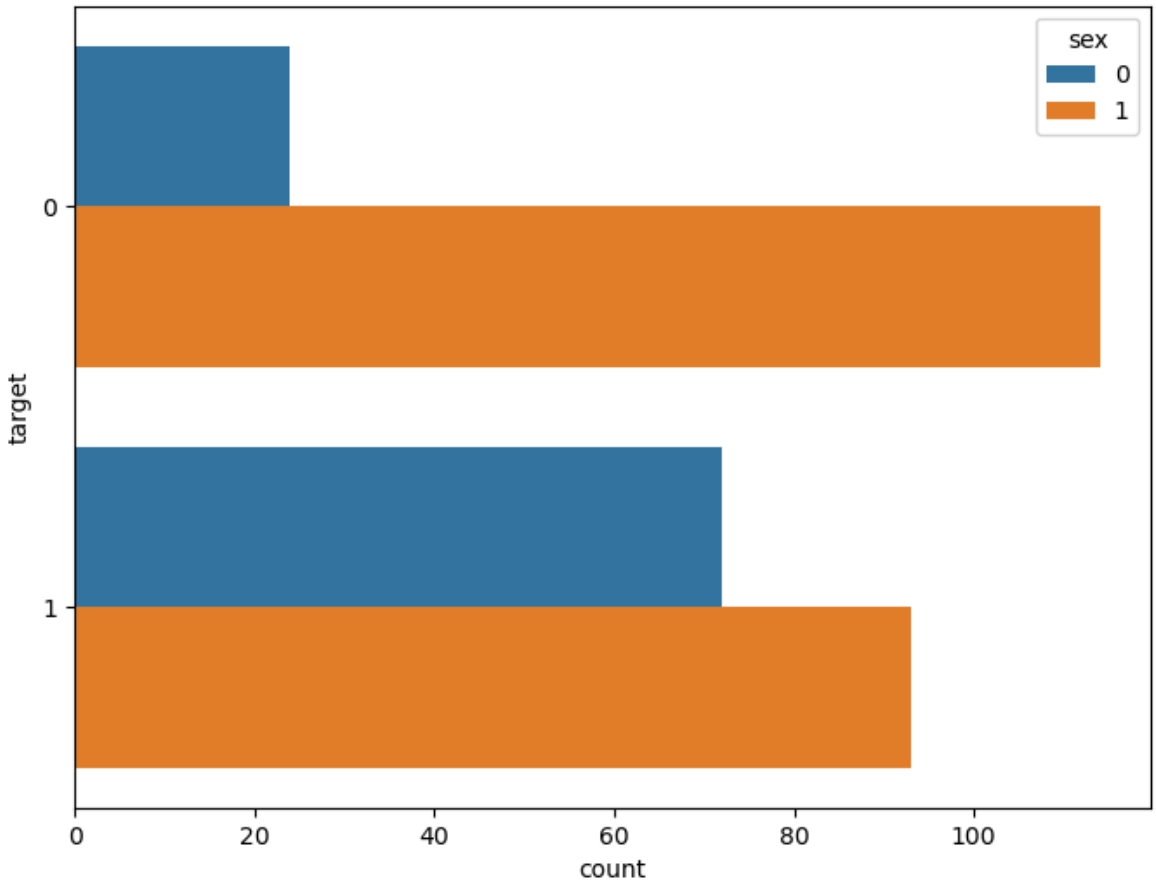
In [15]:
```
# visualize the value counts of the sex varible wrt target

f,ax = plt.subplots(figsize=(8,6))
ax = sns.countplot(x="sex", hue="target", data=df)
plt.show()
```



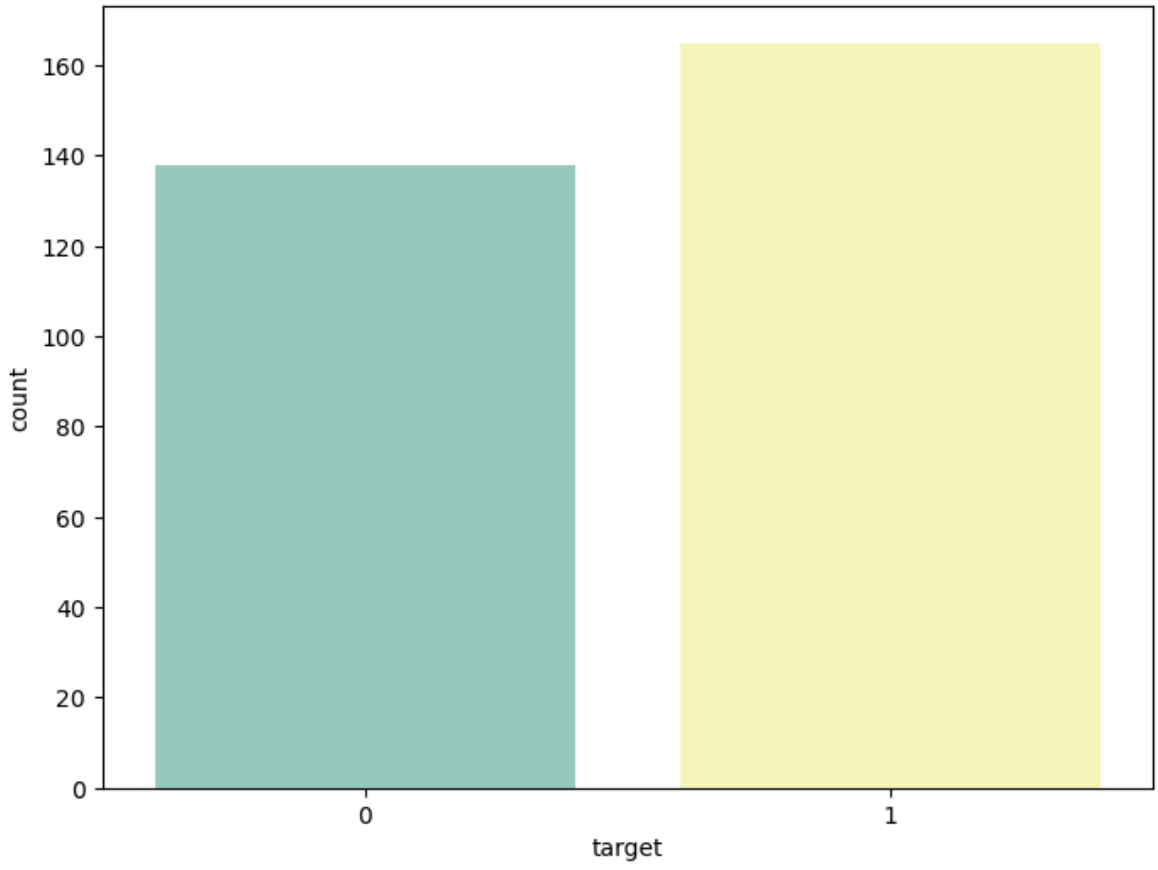
In [16]:
```
ax = sns.catplot(x="target", col="sex", data=df, kind="count", height=5, aspect=
```



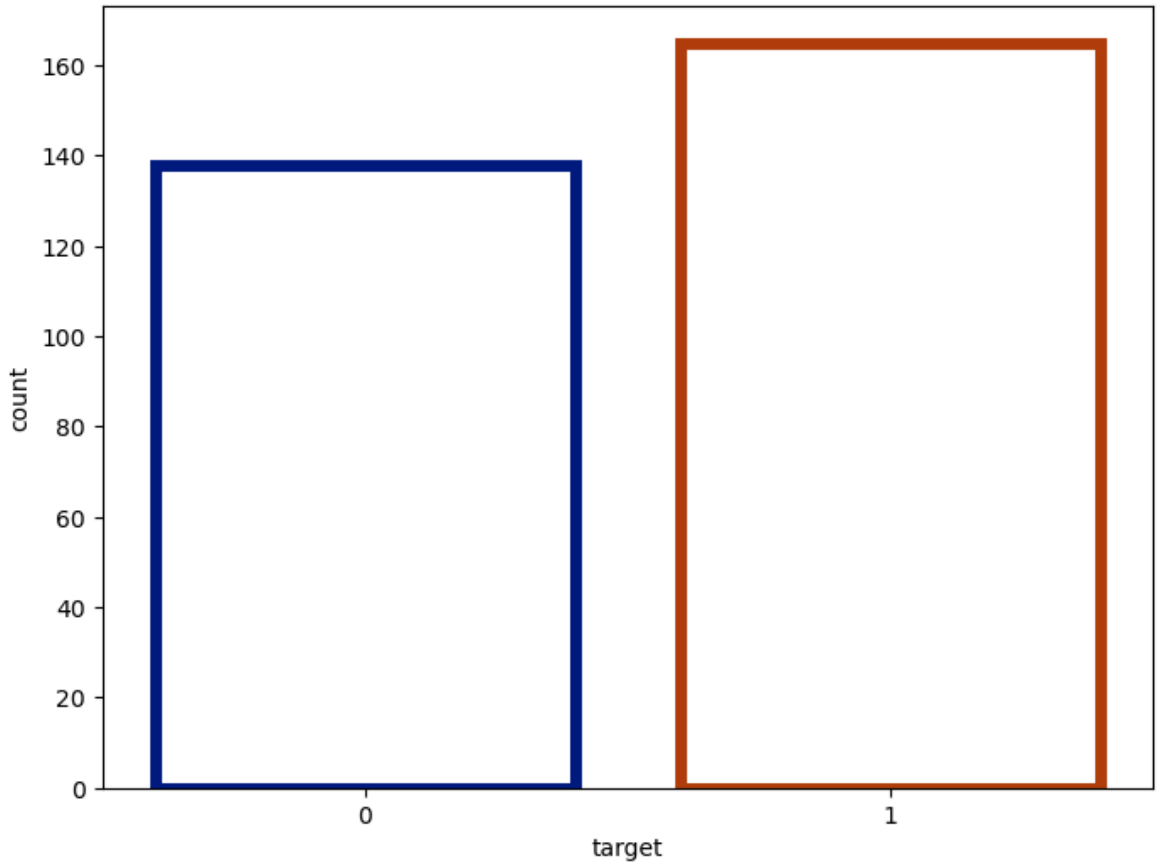
In [17]:
```
f, ax = plt.subplots(figsize=(8,6))
ax = sns.countplot(y="target", hue="sex", data=df)
plt.show()
```



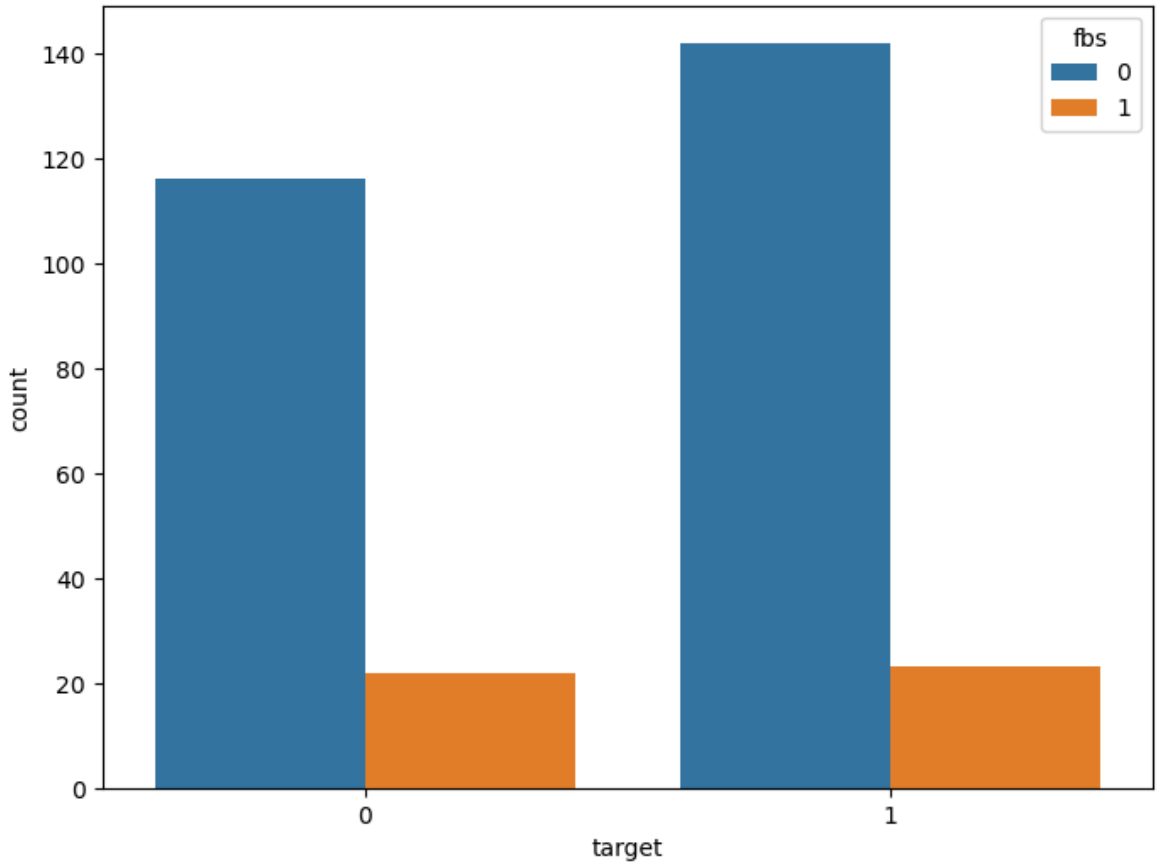
In [18]:
```
f, ax = plt.subplots(figsize=(8,6))
ax = sns.countplot(x="target", data=df, palette="Set3")
plt.show()
```



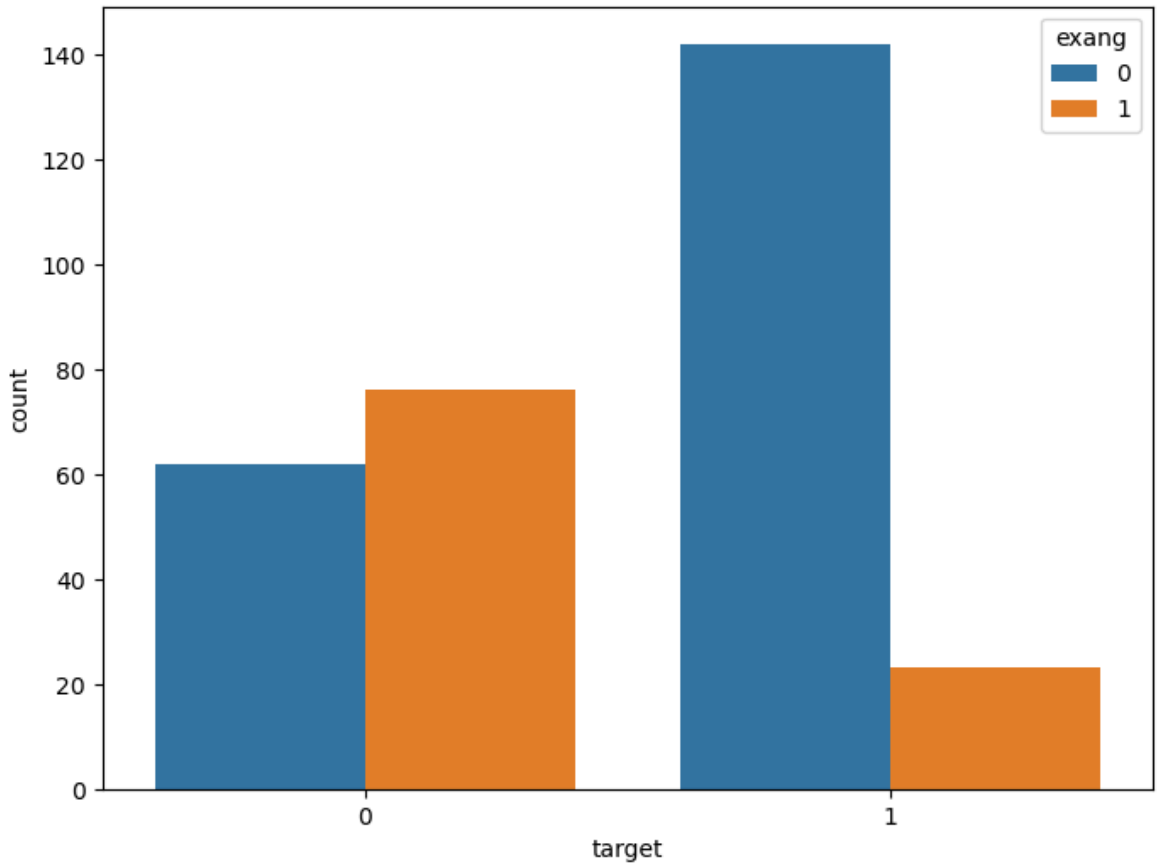
In [19]:
```
# plt.bar
f, ax = plt.subplots(figsize=(8,6))
ax = sns.countplot(x="target", data=df, facecolor=(0,0,0,0), linewidth=5, edgeco
```



In [20]:
```
f, ax = plt.subplots(figsize=(8,6))
ax = sns.countplot(x="target",hue="fbs", data=df)
plt.show()
```



In [21]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", hue="exang", data=df)
plt.show()
```



In [22]:
```
# Bivariate analysis
```

In [23]:
```
# Estimate correlation coefficient

correlation = df.corr()
```

In [24]:
```
correlation['target'].sort_values(ascending=False)
```

Out[24]:
```
target      1.000000
cp          0.433798
thalach     0.421741
slope       0.345877
restecg     0.137230
fbs        -0.028046
chol       -0.085239
trestbps   -0.144931
age        -0.225439
sex        -0.280937
thal       -0.344029
ca         -0.391724
oldpeak    -0.430696
exang      -0.436757
Name: target, dtype: float64
```

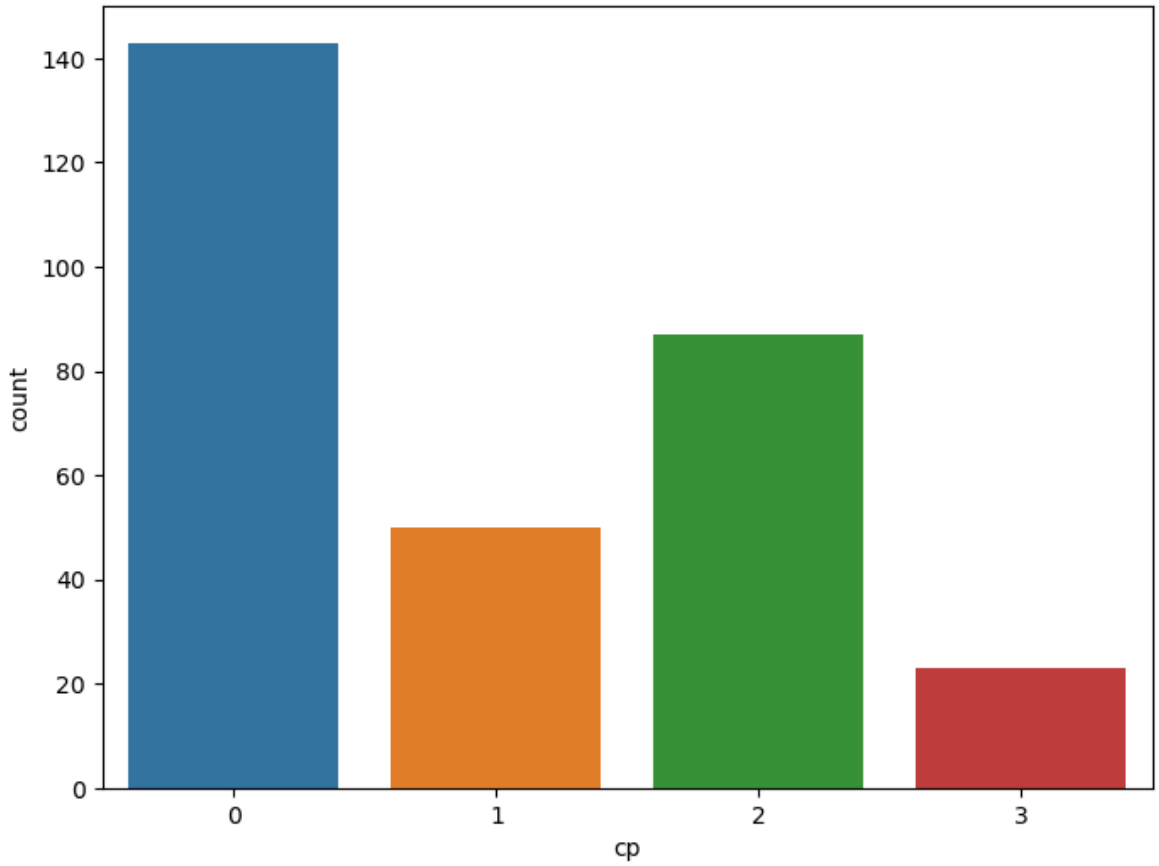
In [25]:
```
df['cp'].nunique()
```

Out[25]:
```
4
```

In [26]:
```
df['cp'].value_counts()
```

Out[26]:
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

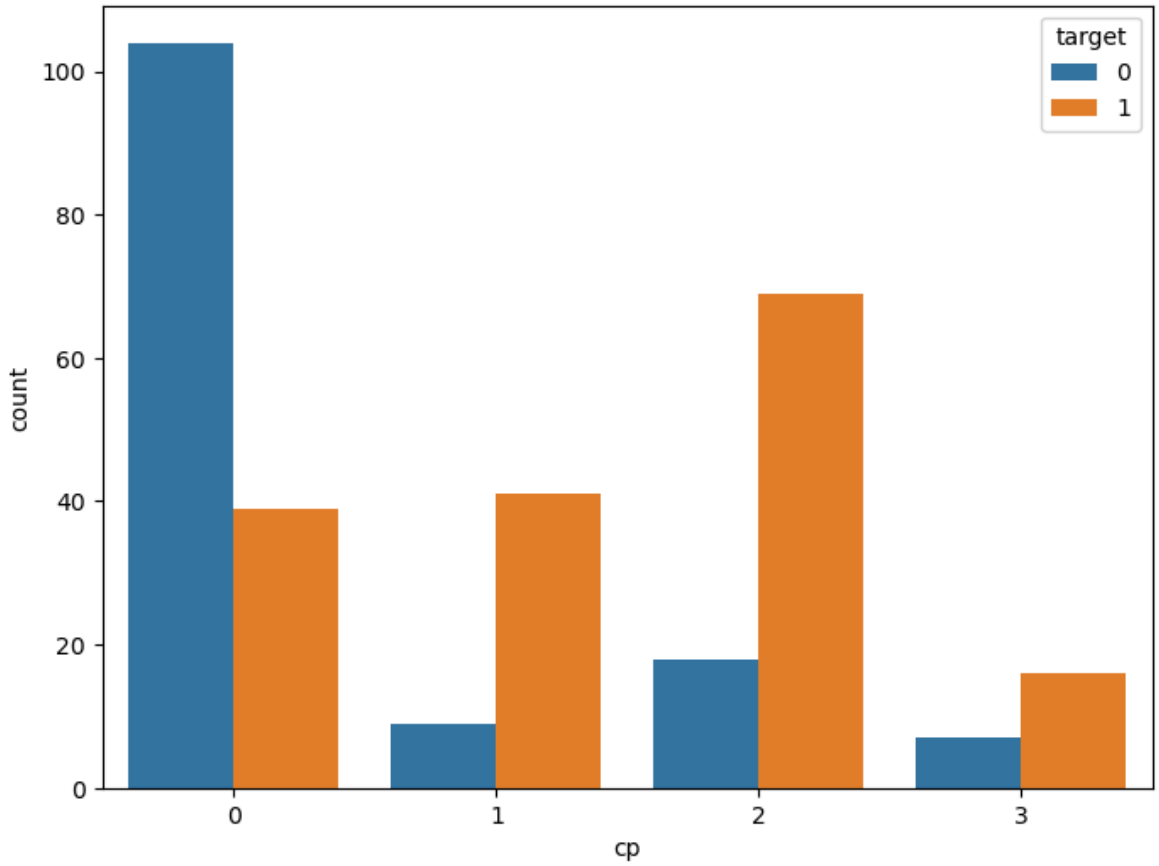
In [27]:
```
# Visualise the frequency distribution of cp variable

f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="cp", data=df)
plt.show()
```



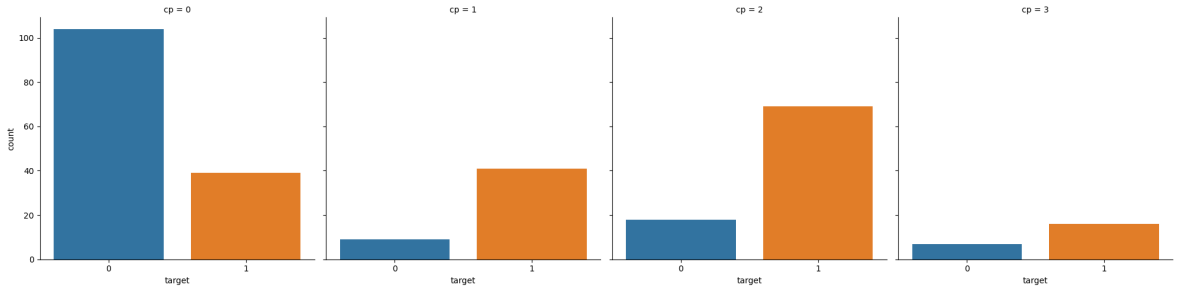
In [28]:
```
# Frequency distribution of `target` variable wrt `cp`

df.groupby('cp')['target'].value_counts()
```

Out[28]:
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: count, dtype: int64
```

In [29]:
```
# Visualise the value counts of the cp variable wrt target

f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="cp", hue="target", data=df)
plt.show()
```



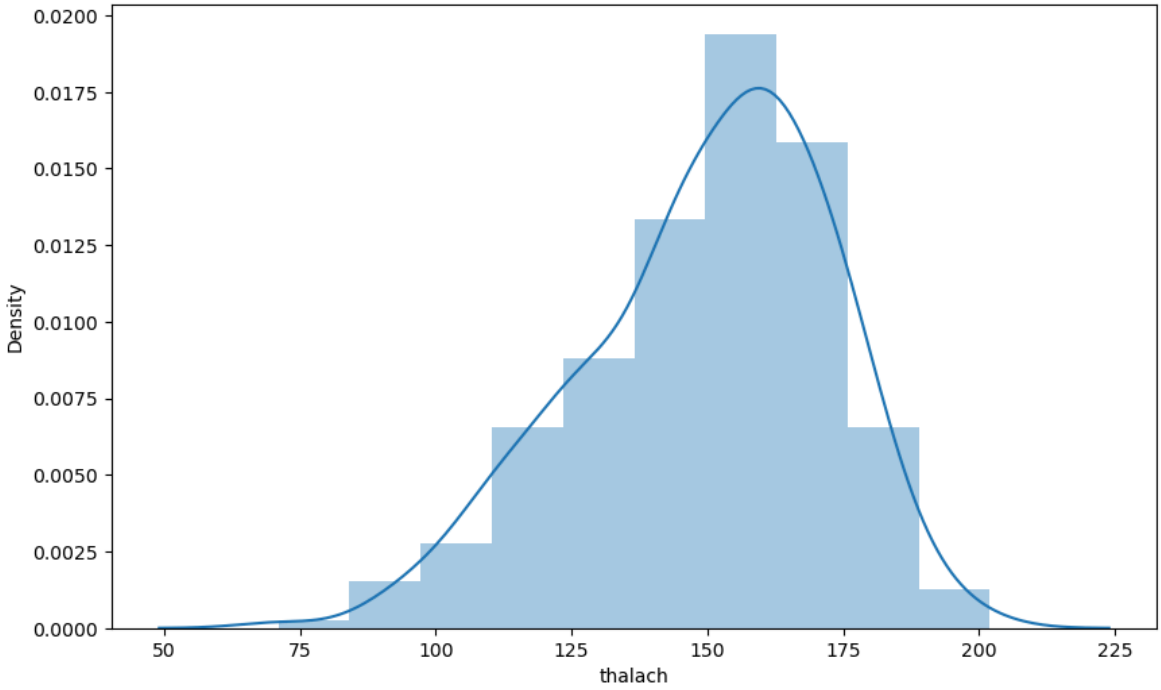
In [33]:
```
ax = sns.catplot(x="target", col="cp", data=df, kind="count", height=5, aspect=1
```



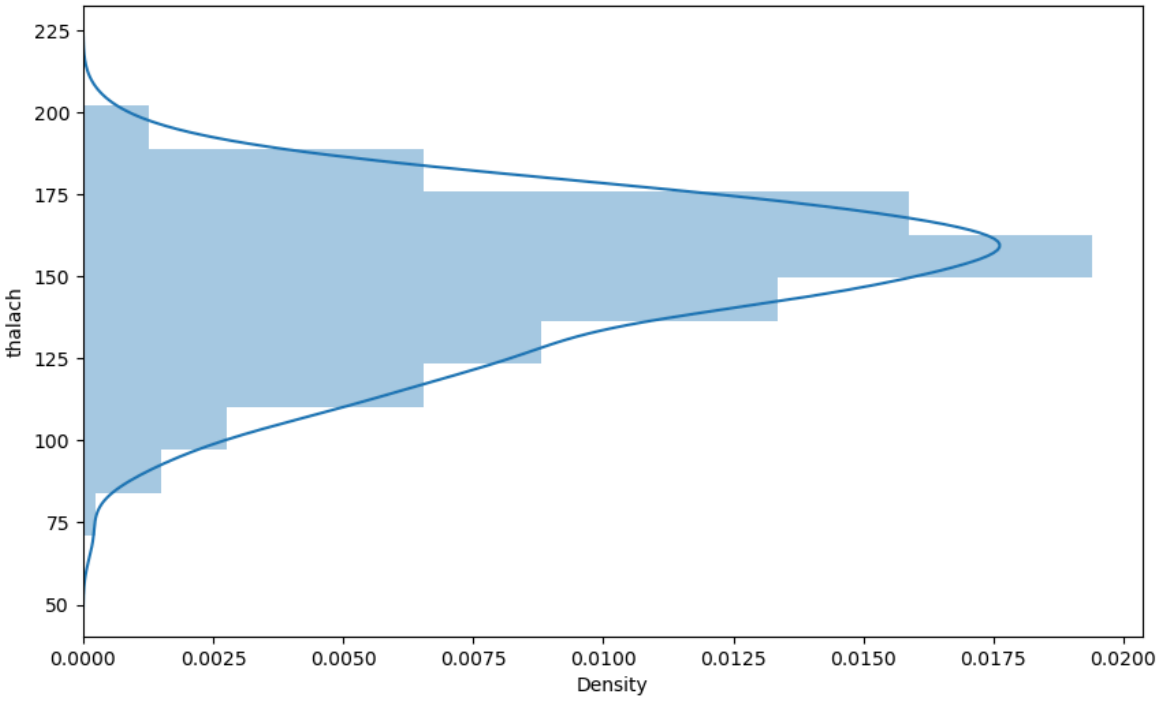
In [34]:
```
# Analysis of target and thalach variable
df['thalach'].nunique()
```

Out[34]: 91

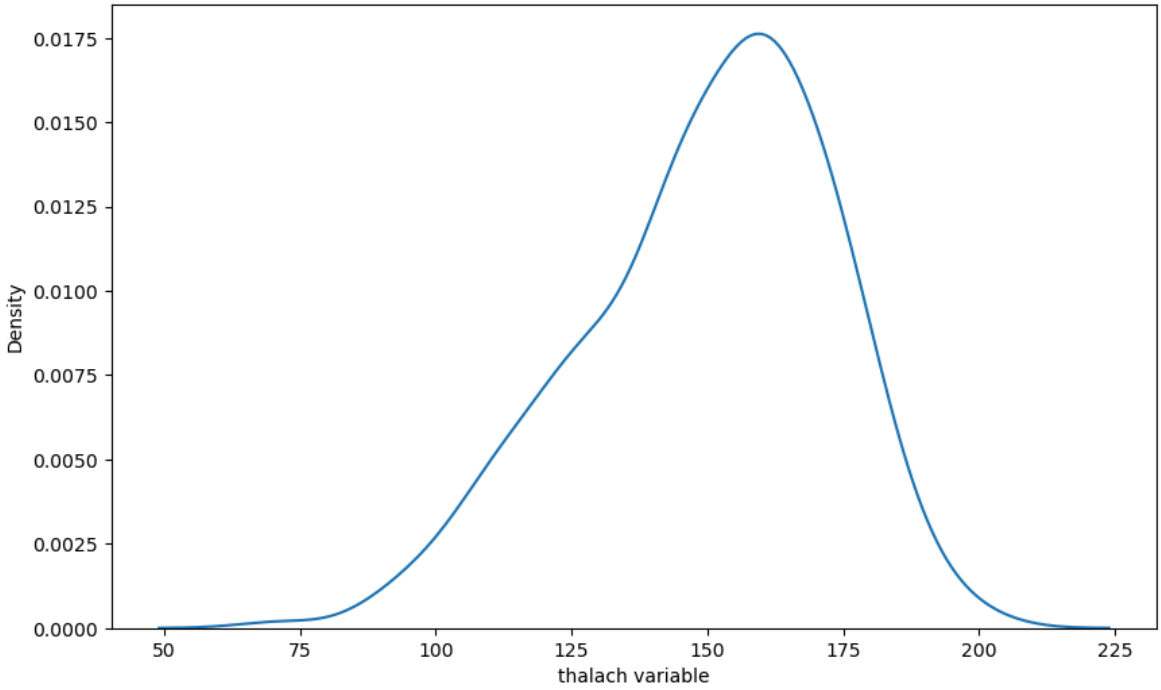
In [35]:
```
# Visualize the frequency distribution of thalach variable

f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x,bins=10)
plt.show()
```



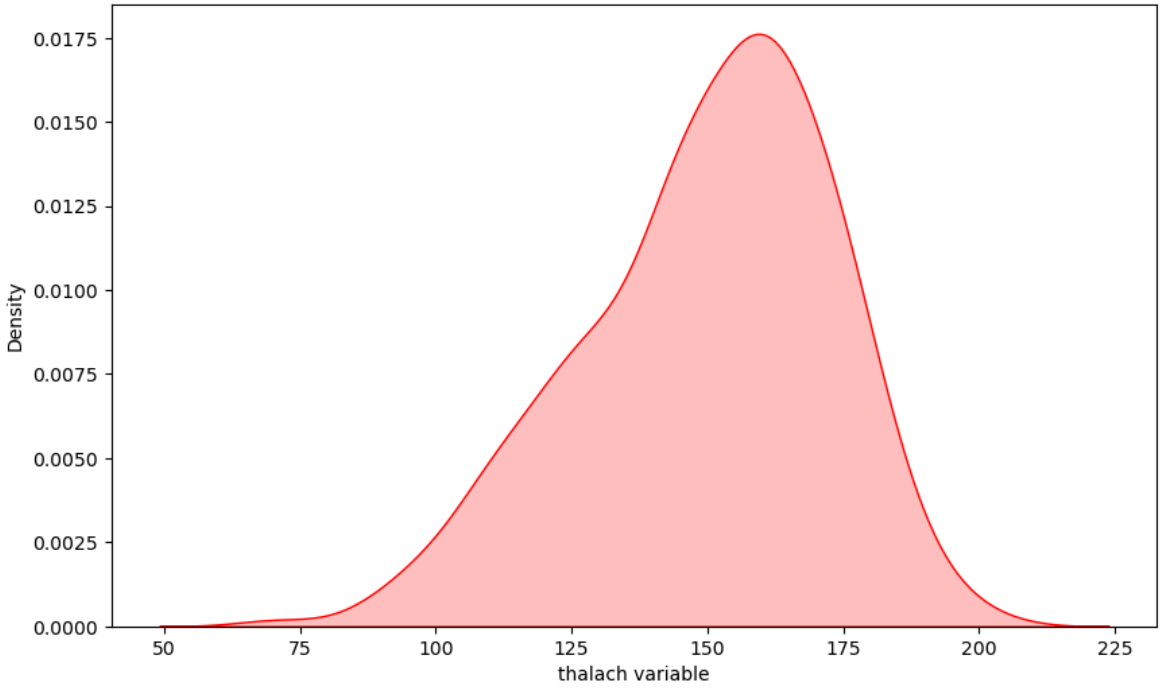
In [36]:
```
# we can plot the distribution on the vertical axis as follows:

f, ax= plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x,bins=10, vertical=True)
plt.show()
```



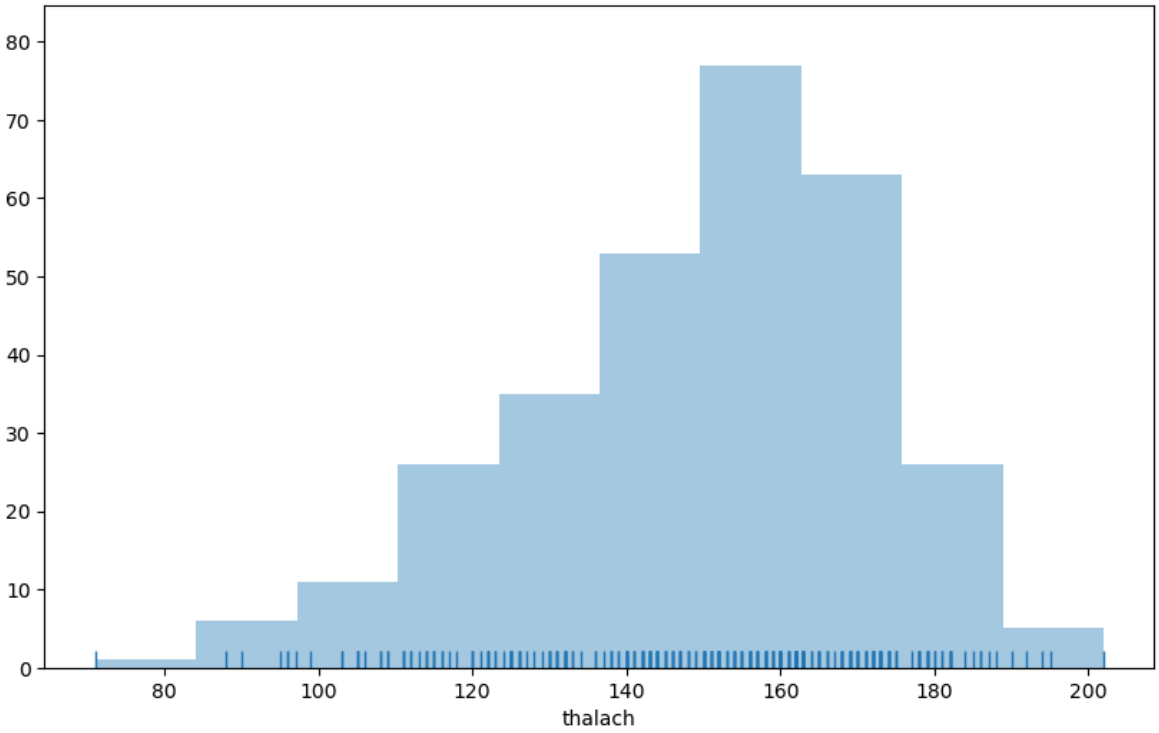
In [37]:
```
# Seaborn Kernel Density Estimation (KDE) Plot

f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x = pd.Series(x, name="thalach variable")
ax = sns.kdeplot(x)
plt.show()
```



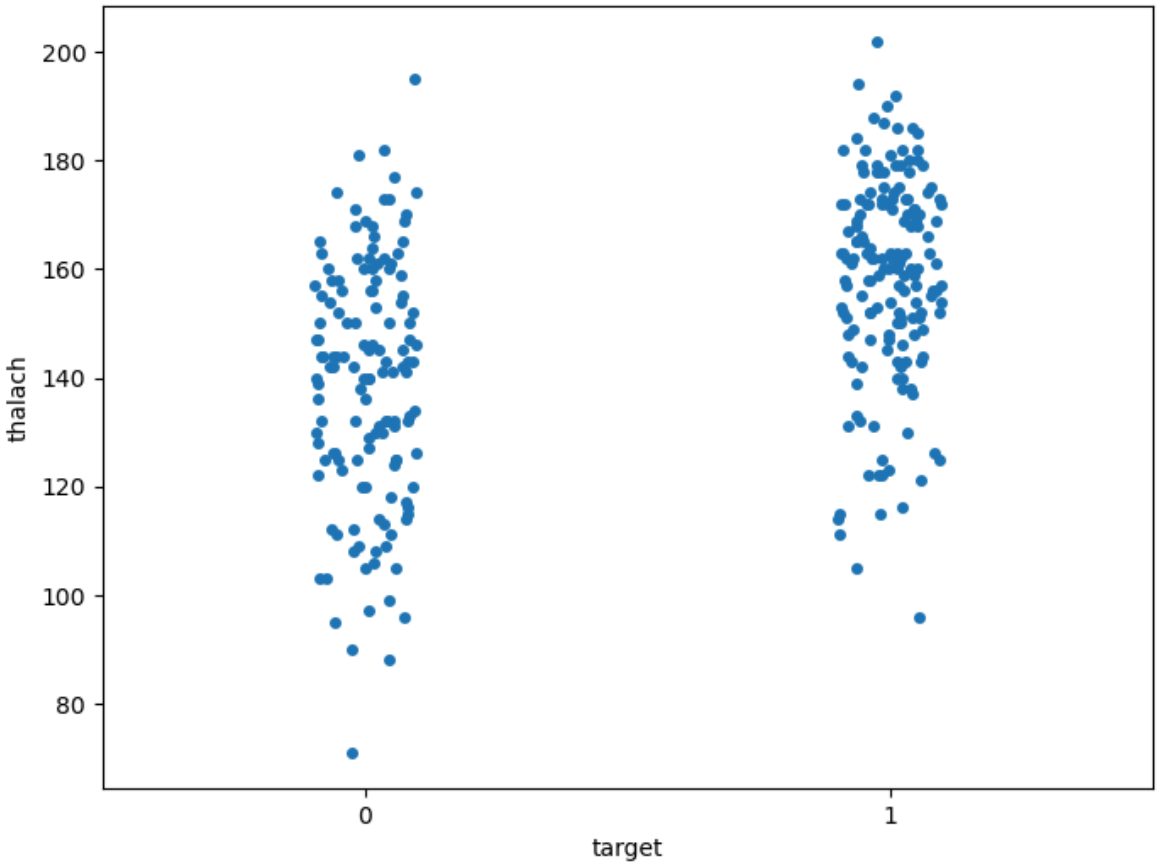
In [39]:
```
# We can shade under the density curve and use a different color:

f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x = pd.Series(x, name="thalach variable")
ax = sns.kdeplot(x, shade=True, color='r')
plt.show()
```



In [40]:
```
# Histogram

f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x, kde=False, rug=True, bins=10)
plt.show()
```

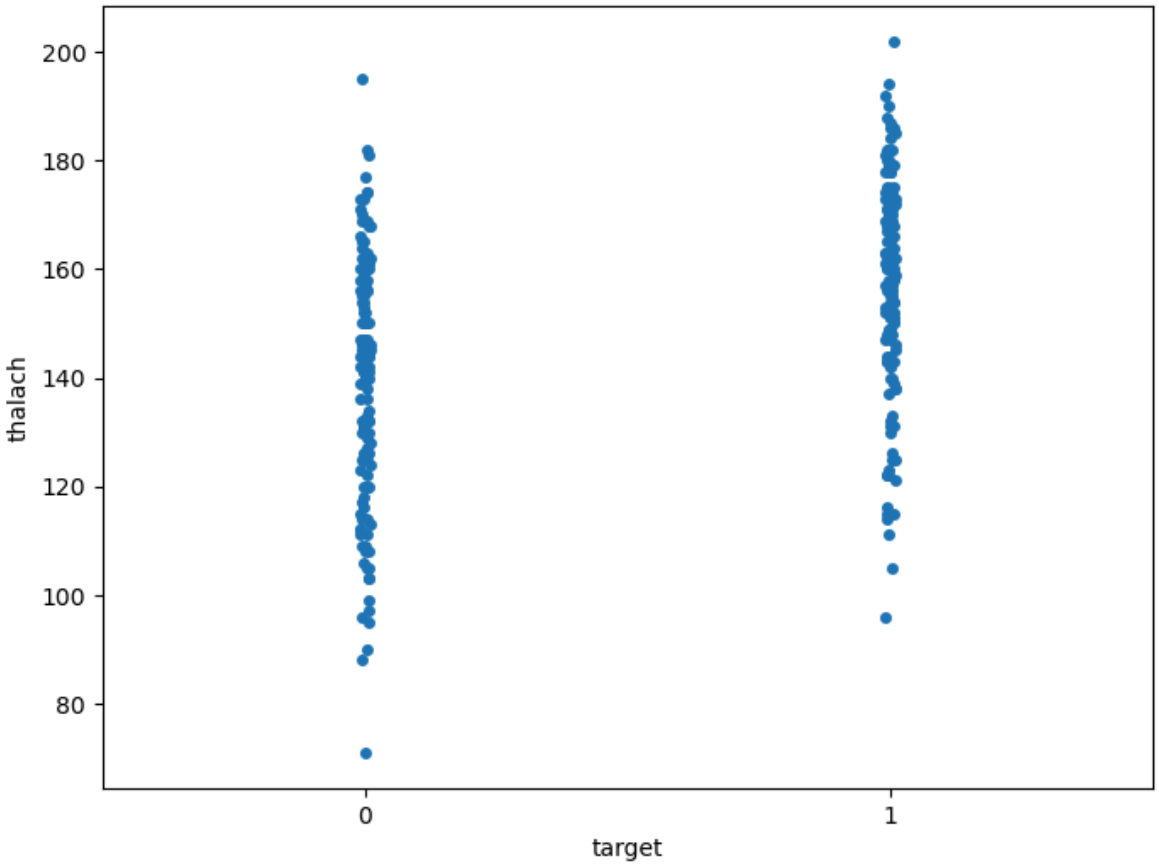


In [45]:
```
# Visualize frequency distribution of thalach variable wrt target

f,ax = plt.subplots(figsize=(8,6))
sns.stripplot(x="target", y="thalach", data=df)
plt.show()
```

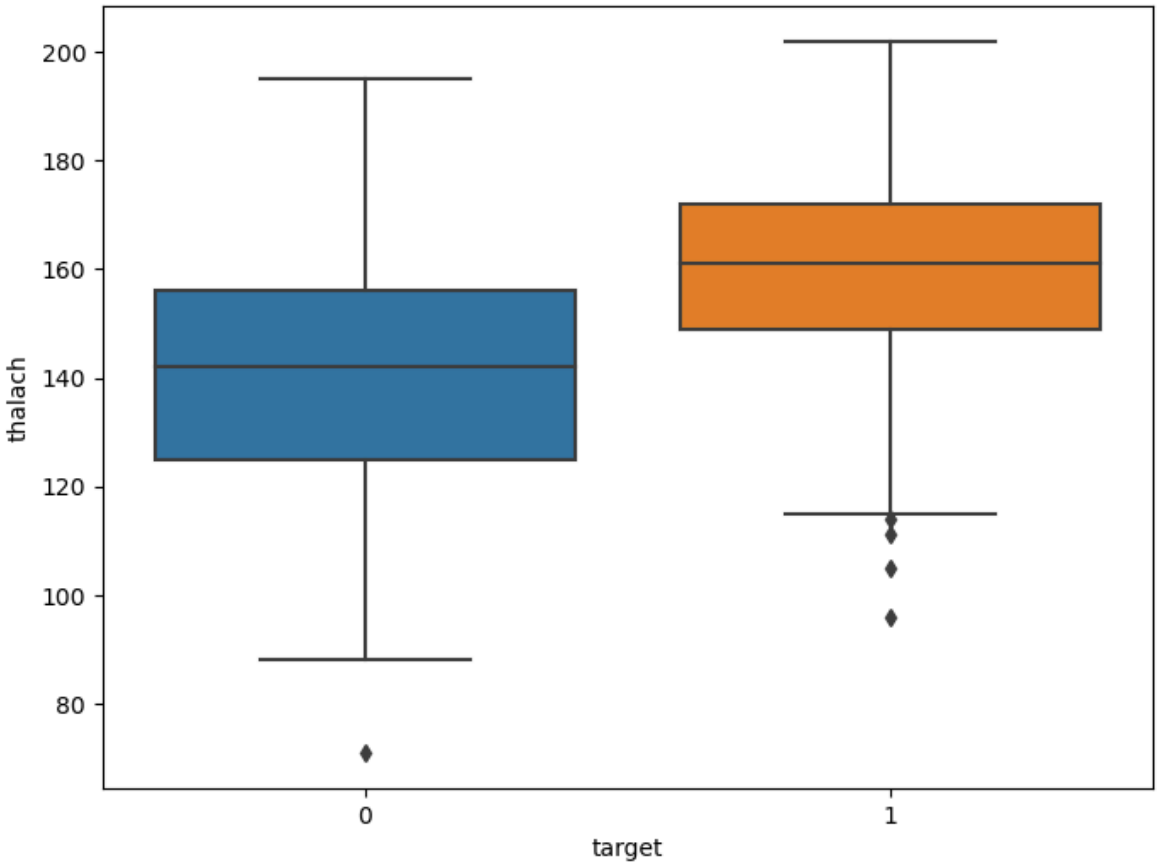


In [47]:
```
# Add the jitter to bring out the distribution of values
f,ax = plt.subplots(figsize=(8,6))
```

```
sns.stripplot(x="target", y="thalach", data=df, jitter = 0.01)
plt.show()
```

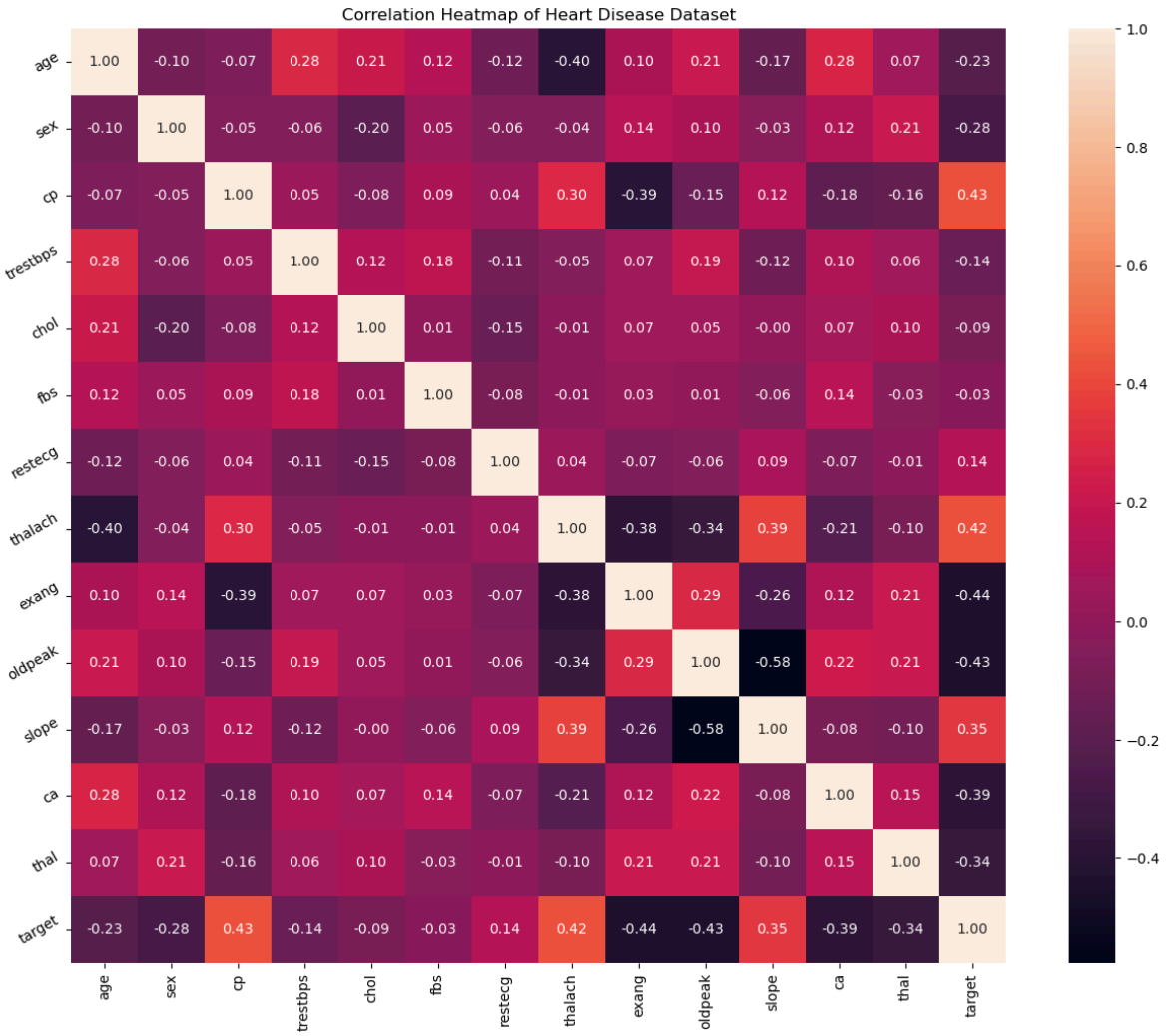


In [48]:
```
# Visualize distribution of thalach variable wrt target with boxplot

f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="target", y="thalach", data=df)
plt.show()
```

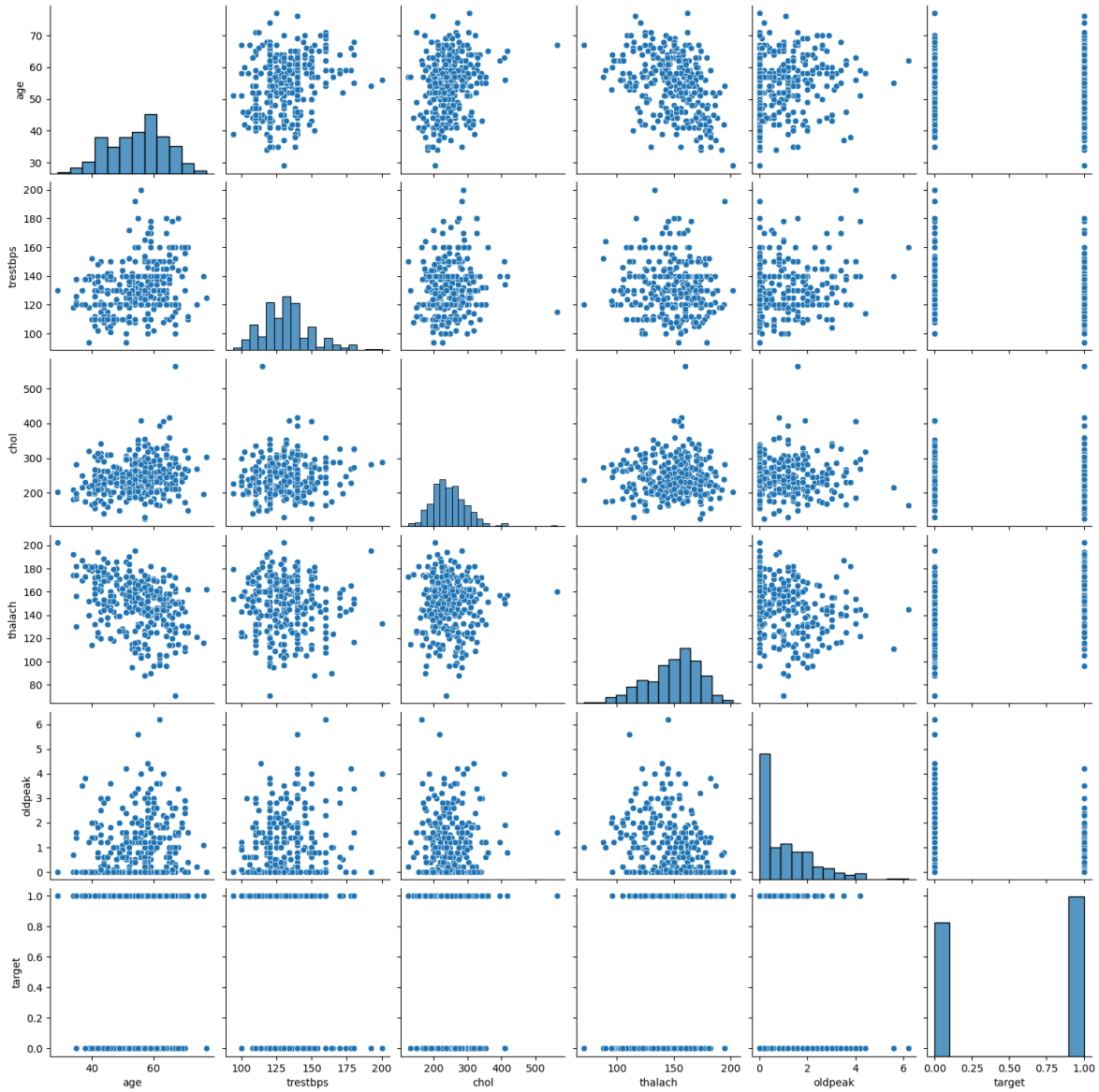


In [ ]:
```
# Multivariate analysis
```

In [51]:
```
# Heat Map

plt.figure(figsize=(16,12))
plt.title('Correlation Heatmap of Heart Disease Dataset')
a = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='whit
a.set_xticklabels(a.get_xticklabels(), rotation=90)
a.set_yticklabels(a.get_yticklabels(), rotation=30)
plt.show()
```



In [52]:
```
# Pair Plot

num_var = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target' ]
sns.pairplot(df[num_var], kind='scatter', diag_kind='hist')
plt.show()
```



In [53]:
```
# Analysis of age and other variables
```

In [54]:
```
df['age'].nunique()
```

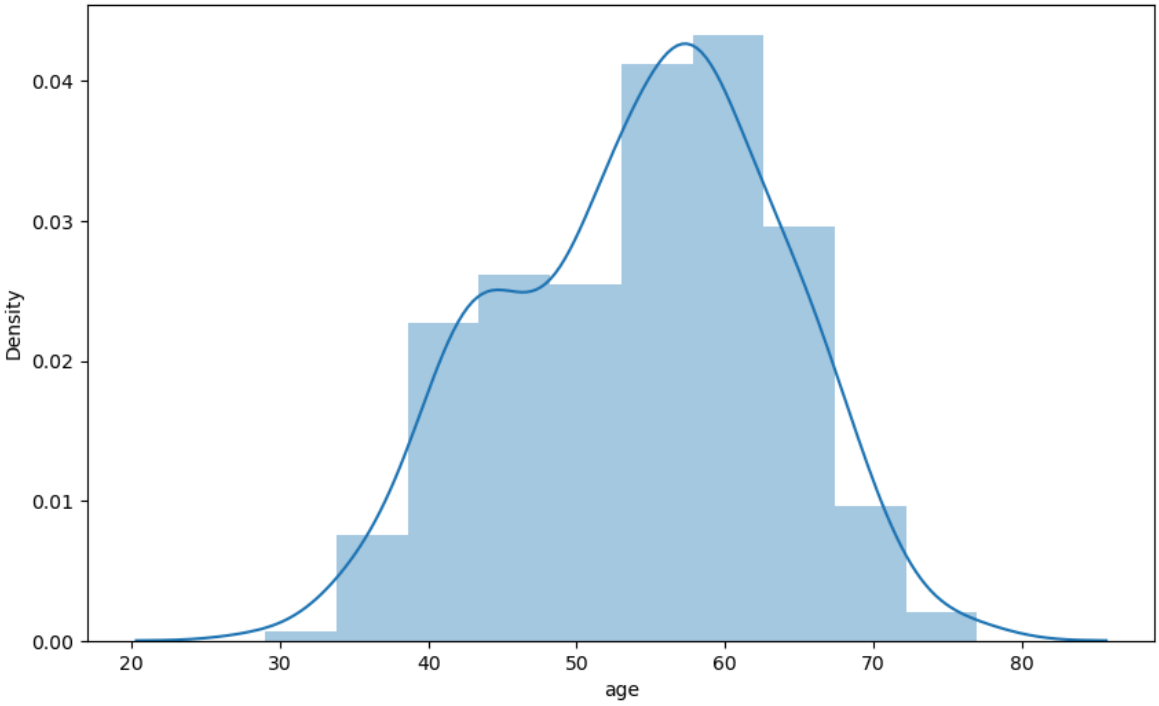
Out[54]:
```
41
```

In [55]:
```
# statistical summary of age variable

df['age'].describe()
```

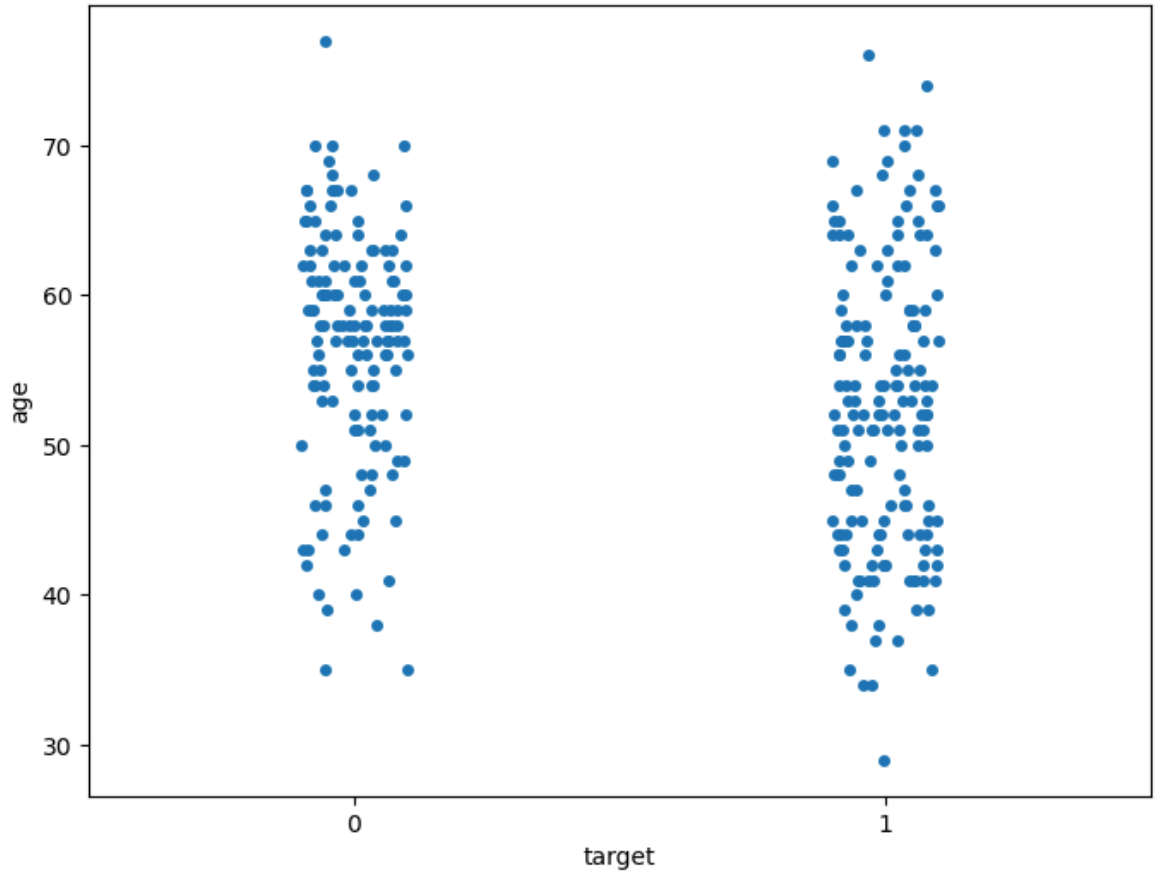
Out[55]:
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

In [56]:
```
# Plot the distribution of age variable

f, ax = plt.subplots(figsize=(10,6))
x = df['age']
```

```
ax = sns.distplot(x, bins=10)
plt.show()
```

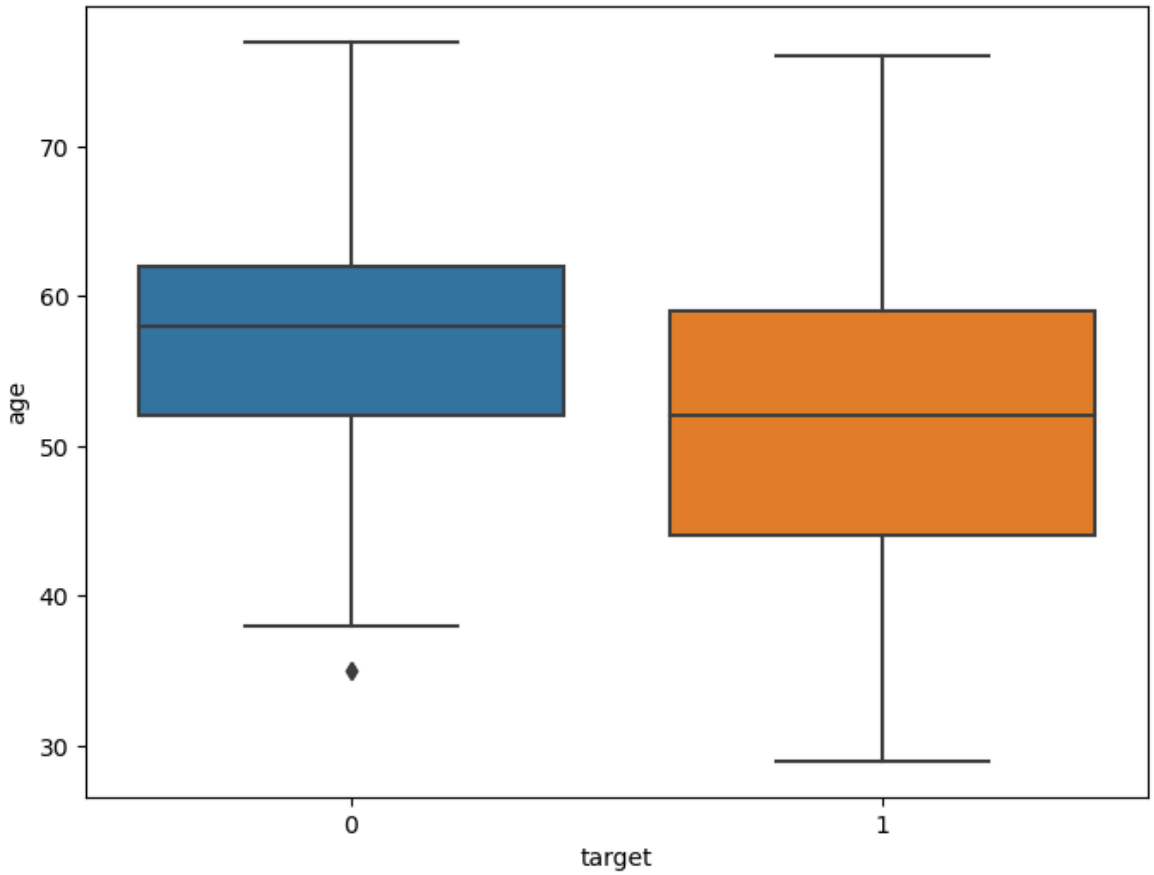


In [57]:
```
# Analyze age and target variable

f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="age", data=df)
plt.show()
```

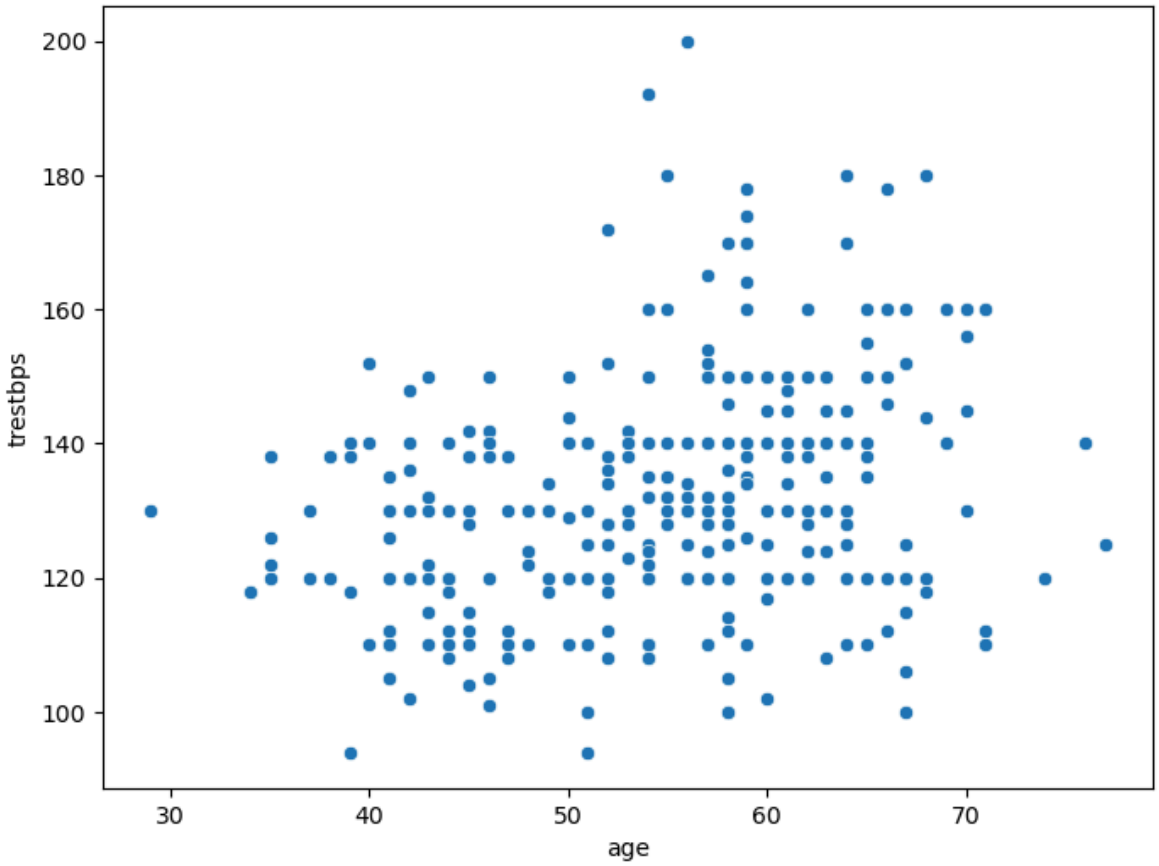


In [58]:
```
# Visualise distribution of age variable wrt target with boxplot
```

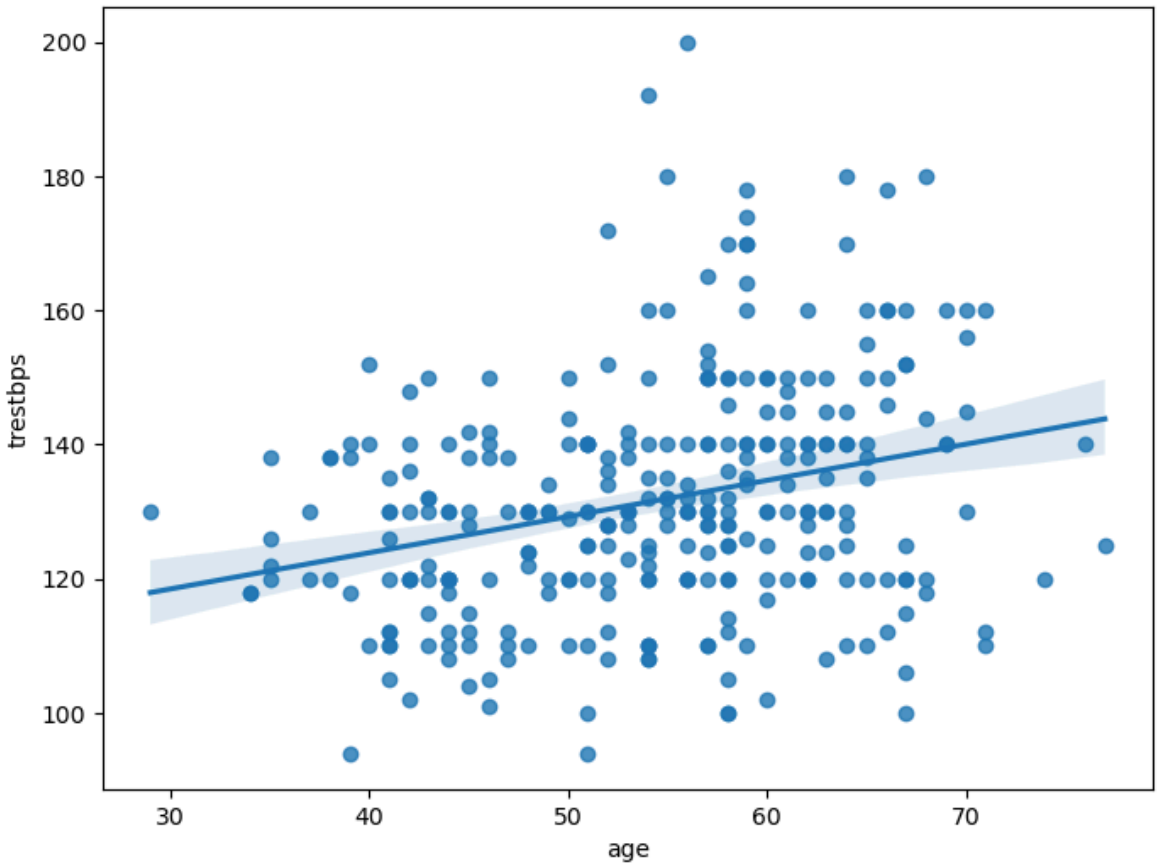
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="target", y="age", data=df)
plt.show()
```



In [59]:
```
# Analyze age and trestbps variable

f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="age", y="trestbps", data=df)
plt.show()
```

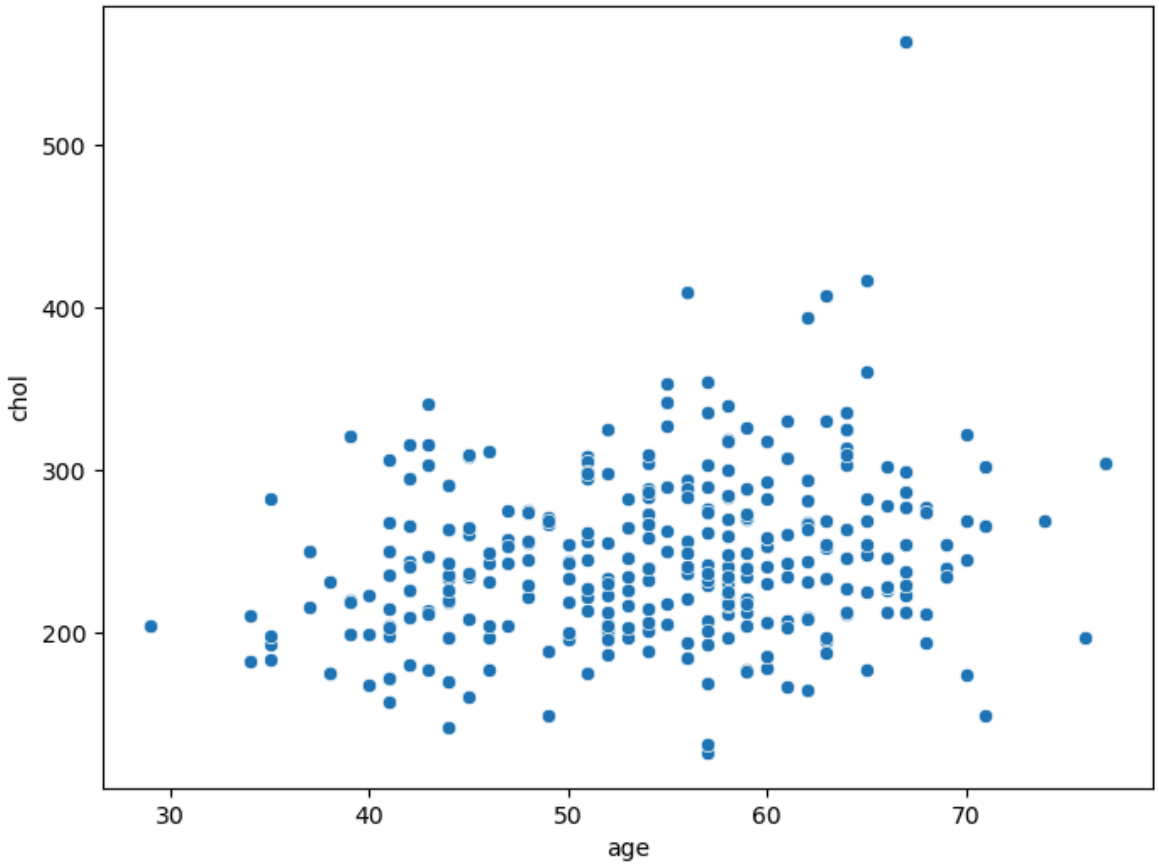


In [60]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="trestbps", data=df)
plt.show()
```

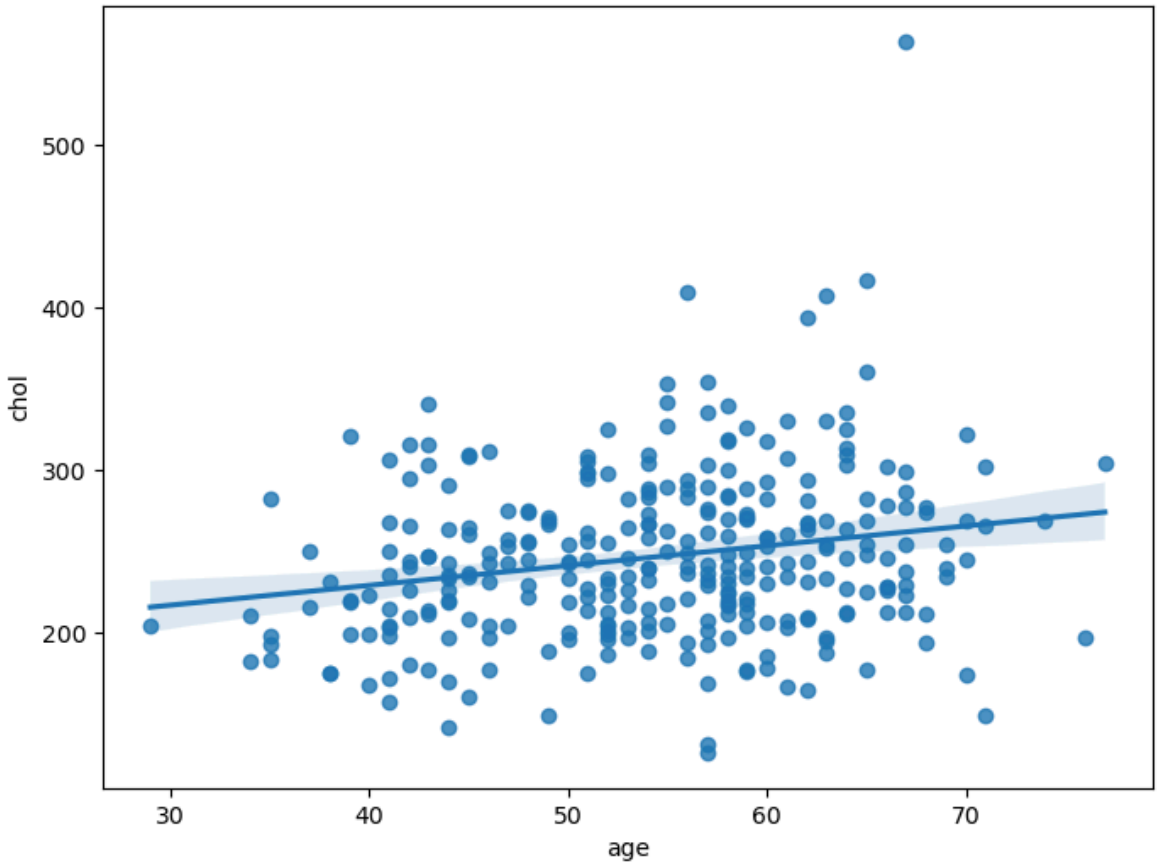


In [61]:
```
# Analyze age and chol variable
```

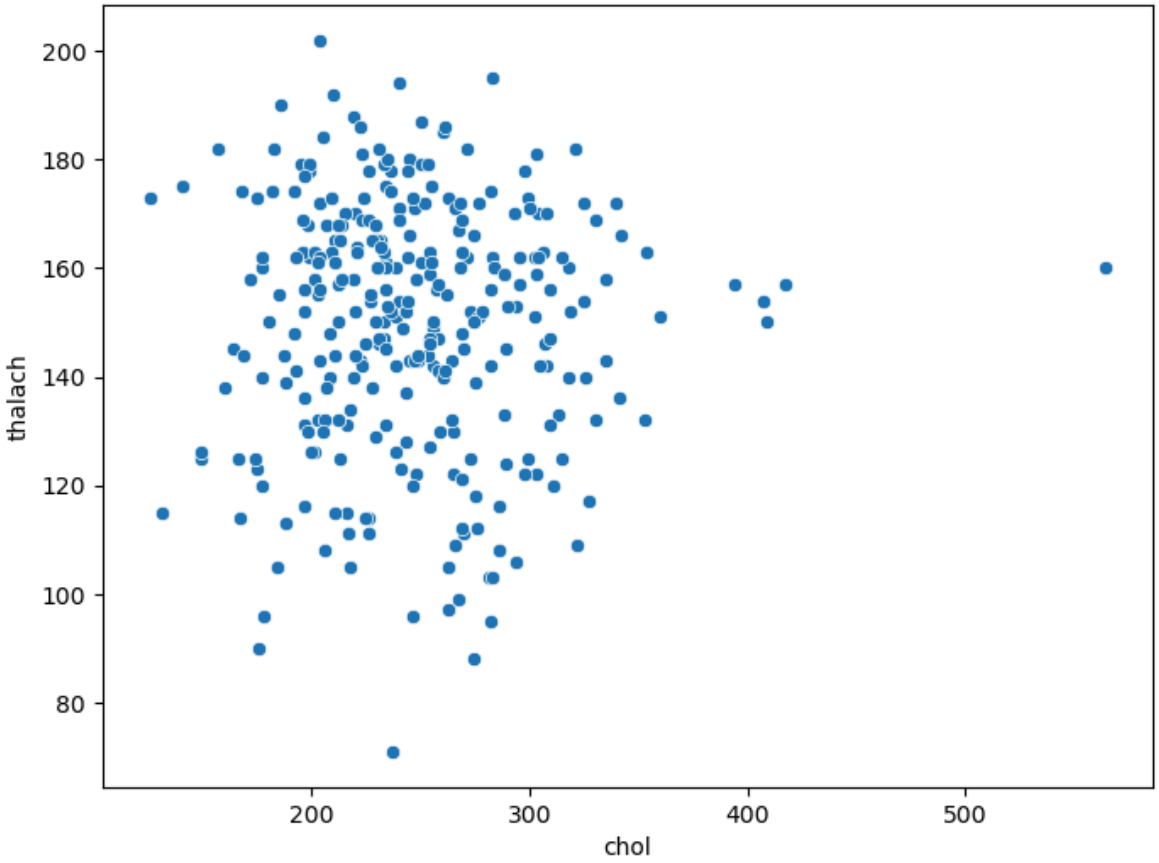
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="age", y="chol", data=df)
plt.show()
```



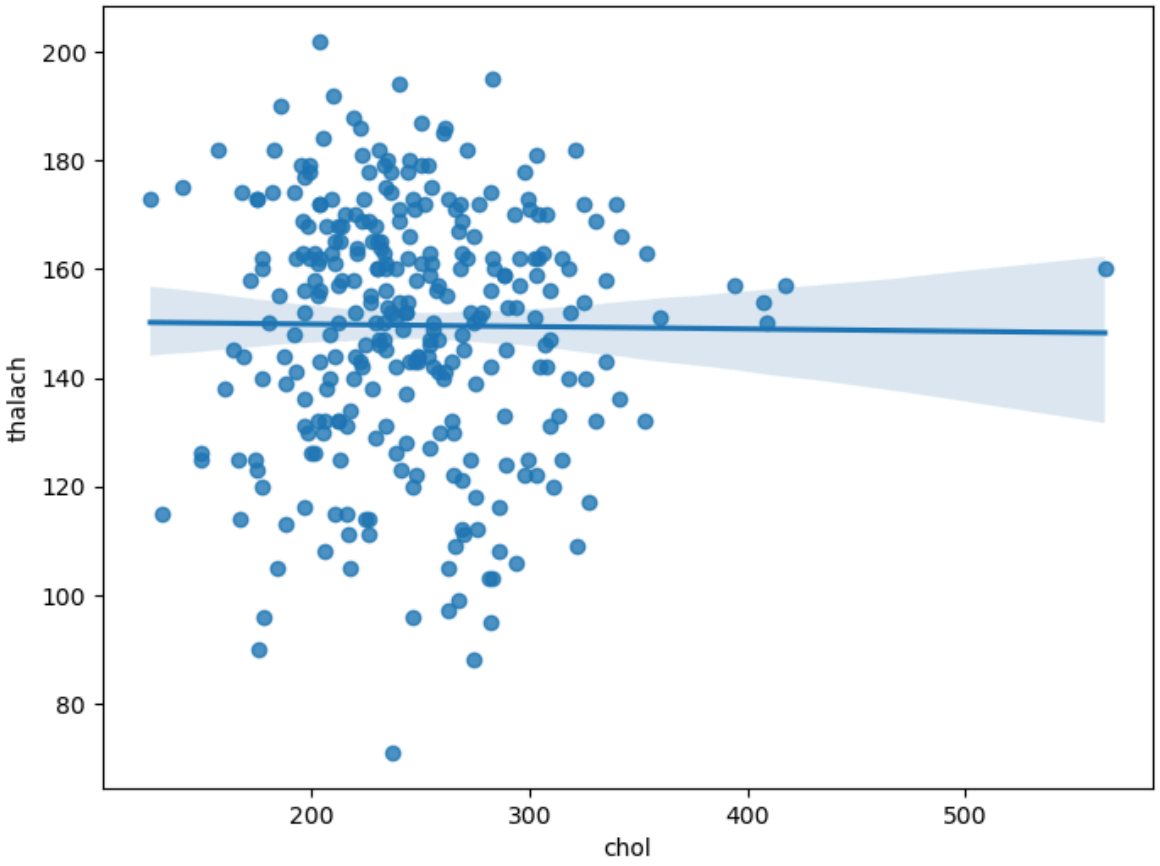
In [62]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="chol", data=df)
plt.show()
```



In [63]:
```
# Analyze chol and thalach variable

f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="chol", y = "thalach", data=df)
plt.show()
```



In [64]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="chol", y = "thalach", data=df)
plt.show()
```



In [65]:
```
# check missing values

df.isnull()
```

Out[65]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	False	False	False	False	False
299	False	False	False	False	False	False	False	False	False	False	False
300	False	False	False	False	False	False	False	False	False	False	False
301	False	False	False	False	False	False	False	False	False	False	False
302	False	False	False	False	False	False	False	False	False	False	False

303 rows × 14 columns

In [66]:
```
df.isnull().sum()
```

Out[66]:
```
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 [69]:
```
assert pd.notnull(df).all().all()
```

In [70]:
```
# assert all values are geater than or equal to 0

assert(df>=0).all().all()
```

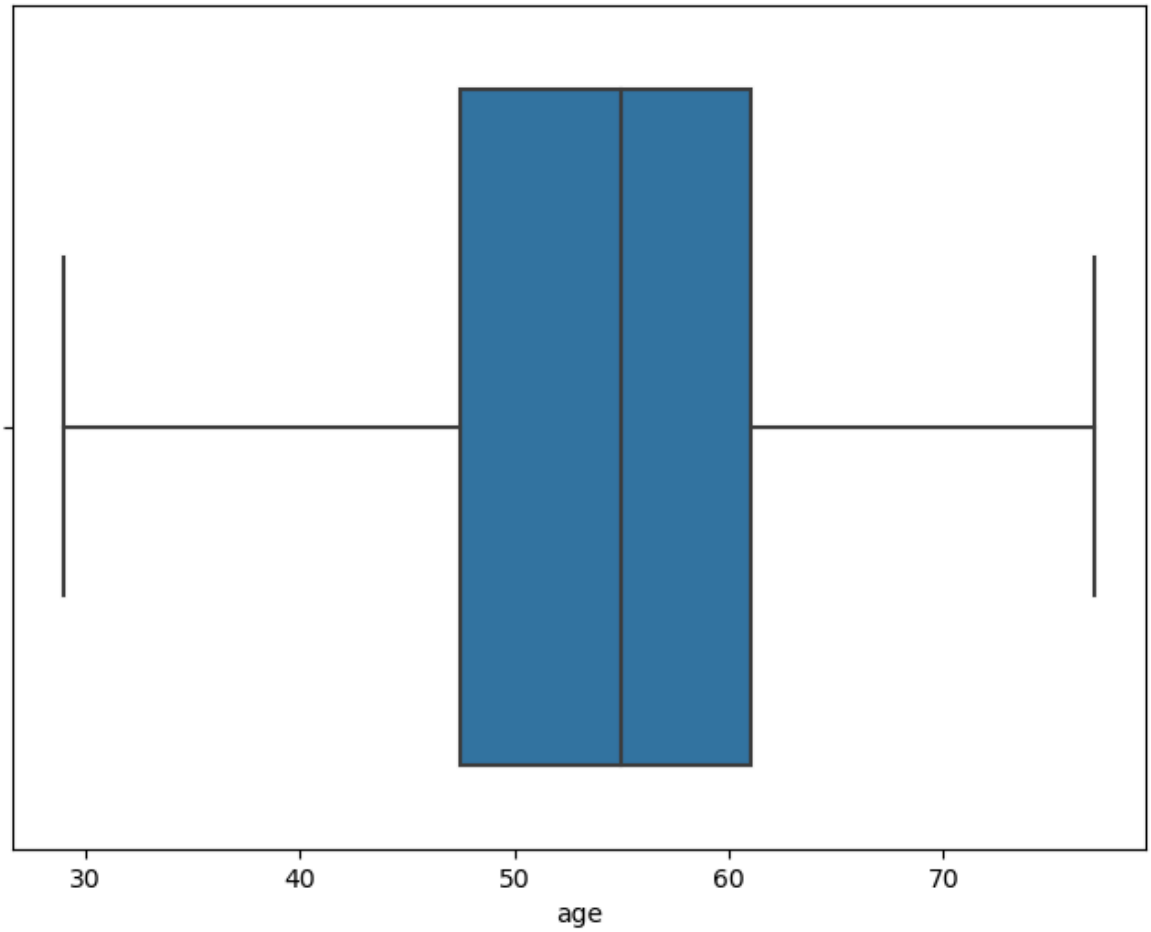
In [71]:
```
# The above two commands do not throw any error . hence it is confirmed that the
```

Outlier detection

In [72]:
```
df['age'].describe()
```

Out[72]:
```
count    303.000000
mean      54.366337
std        9.082101
min       29.000000
25%       47.500000
50%       55.000000
75%       61.000000
max       77.000000
Name: age, dtype: float64
```

In [73]:
```
# Box PLot of age variable

f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["age"])
plt.show()
```

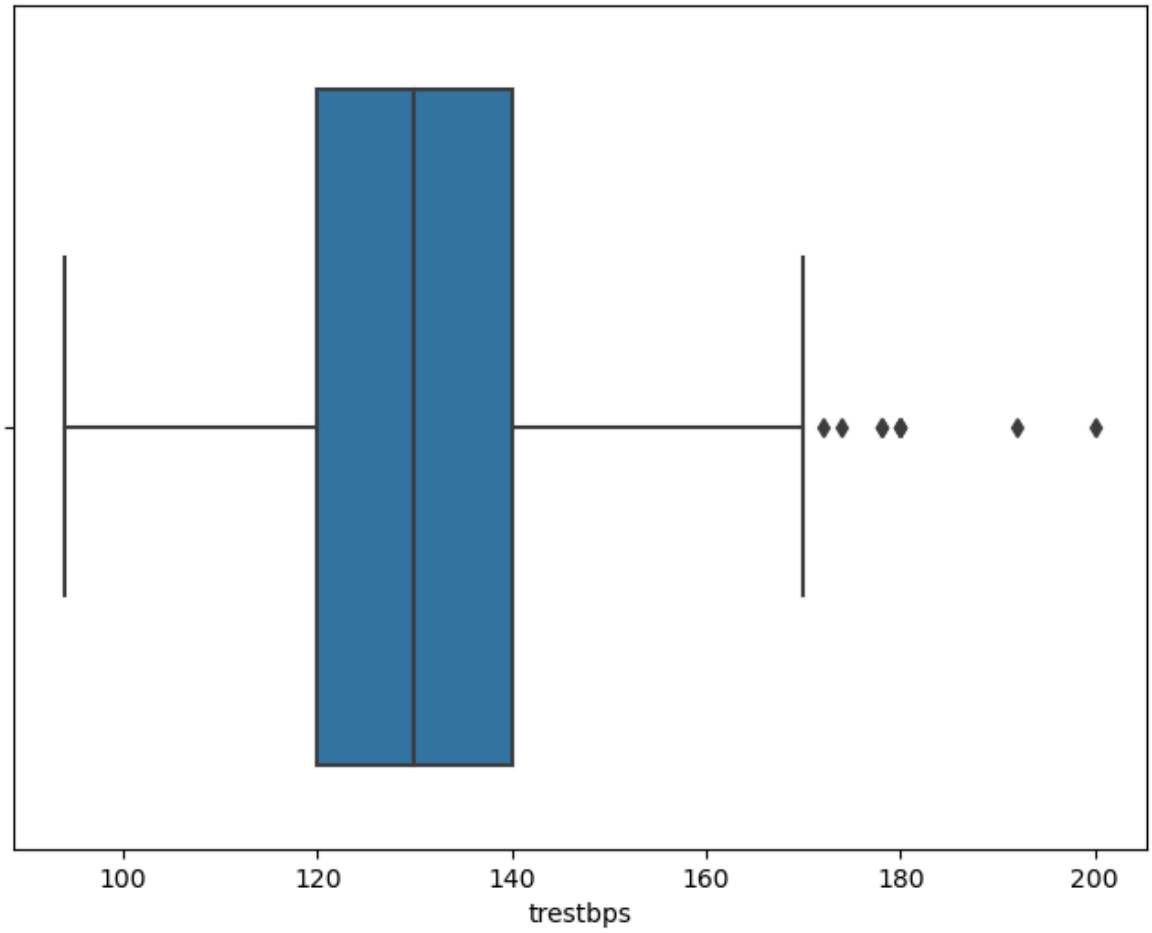


In [74]:
```
# trestbps variable
df['trestbps'].describe()
```

Out[74]:
```
count    303.000000
mean     131.623762
std       17.538143
min       94.000000
25%      120.000000
50%      130.000000
75%      140.000000
max      200.000000
Name: trestbps, dtype: float64
```

In [75]:
```
# Boxplot of trestbps variable
f, ax = plt.subplots(figsize=(8, 6))
```

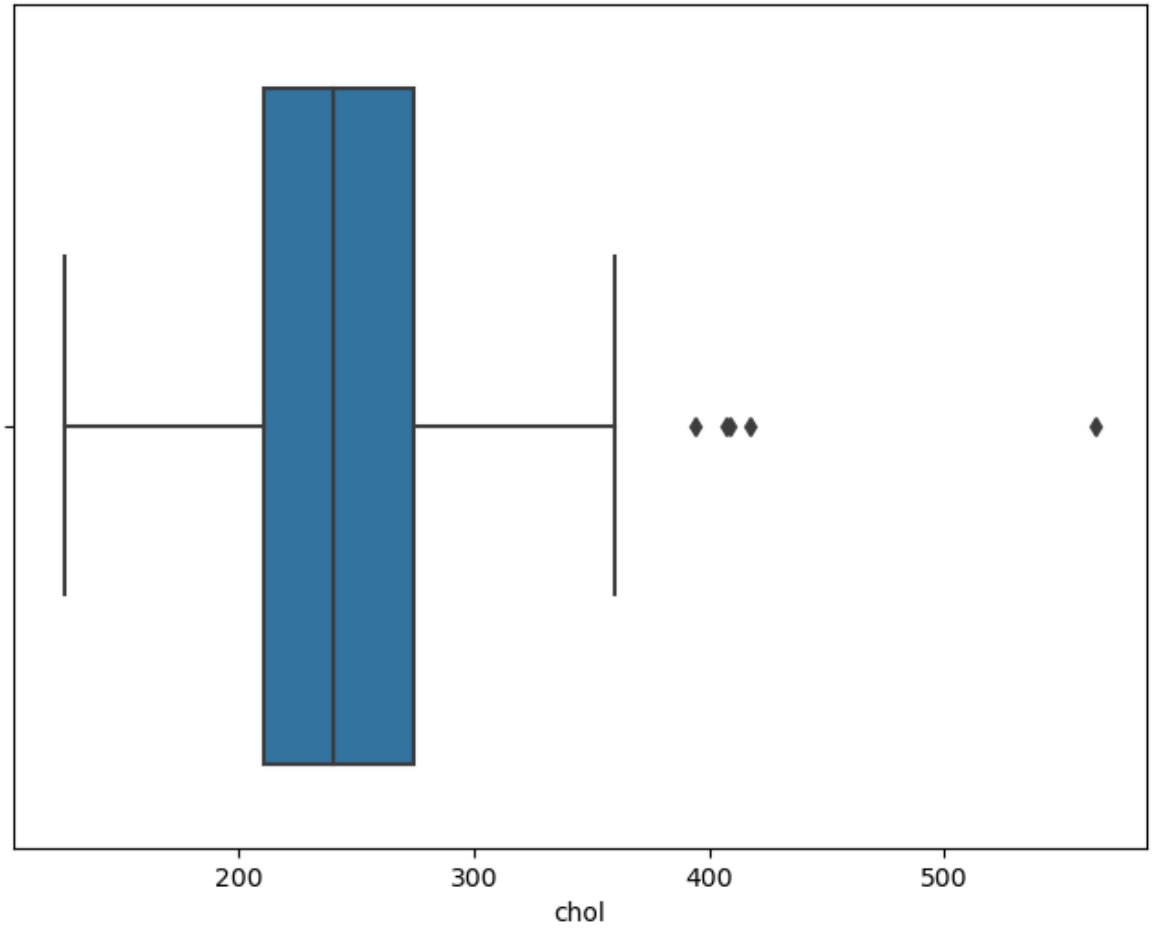
```
sns.boxplot(x=df["trestbps"])
plt.show()
```



In [76]:
```
# Chol variable

df['chol'].describe()
```

Out[76]:
```
count    303.000000
mean     246.264026
std       51.830751
min      126.000000
25%      211.000000
50%      240.000000
75%      274.500000
max      564.000000
Name: chol, dtype: float64
```

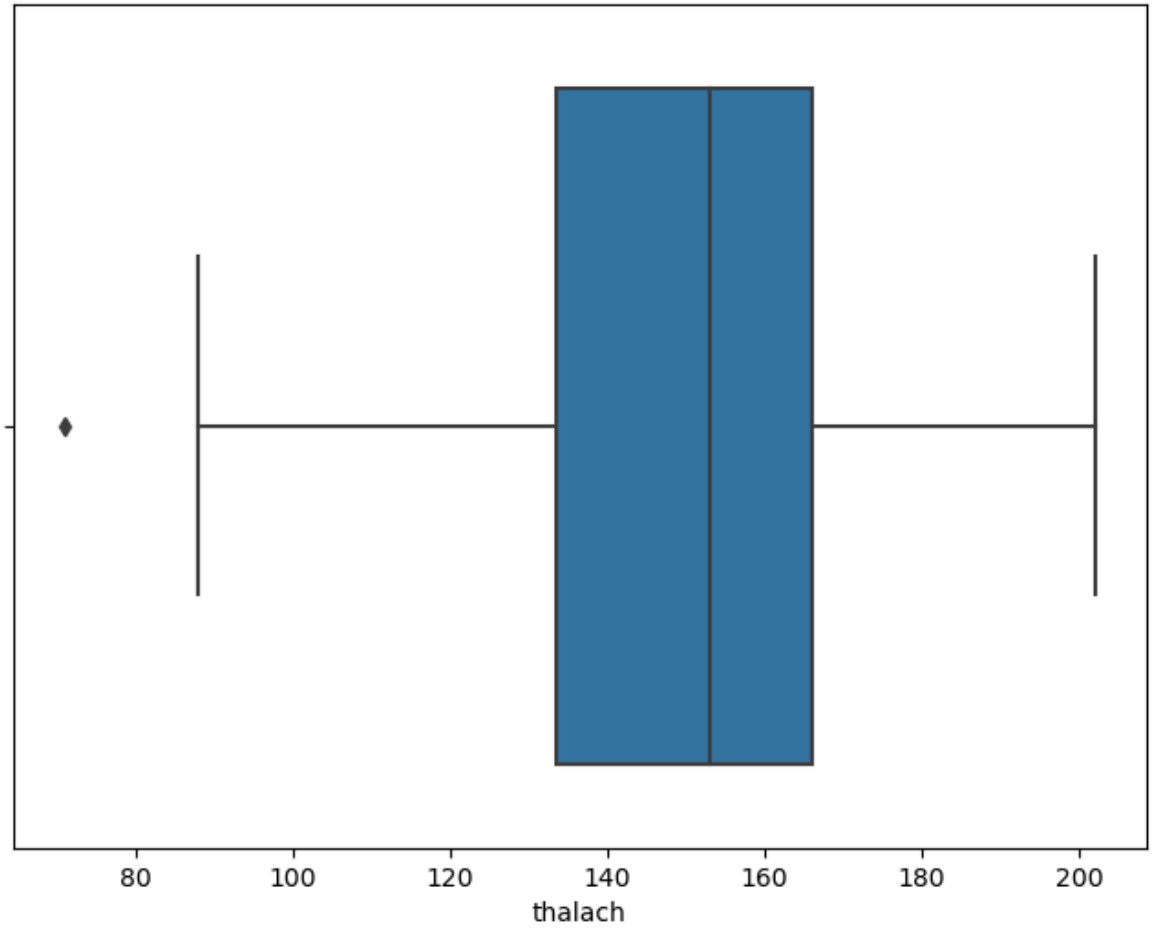
In [77]:
```
# Box plot of chol variable
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["chol"])
plt.show()
```



In [78]:
```
# thalach variable
df['thalach'].describe()
```

Out[78]:
```
count    303.000000
mean     149.646865
std       22.905161
min       71.000000
25%      133.500000
50%      153.000000
75%      166.000000
max      202.000000
Name: thalach, dtype: float64
```

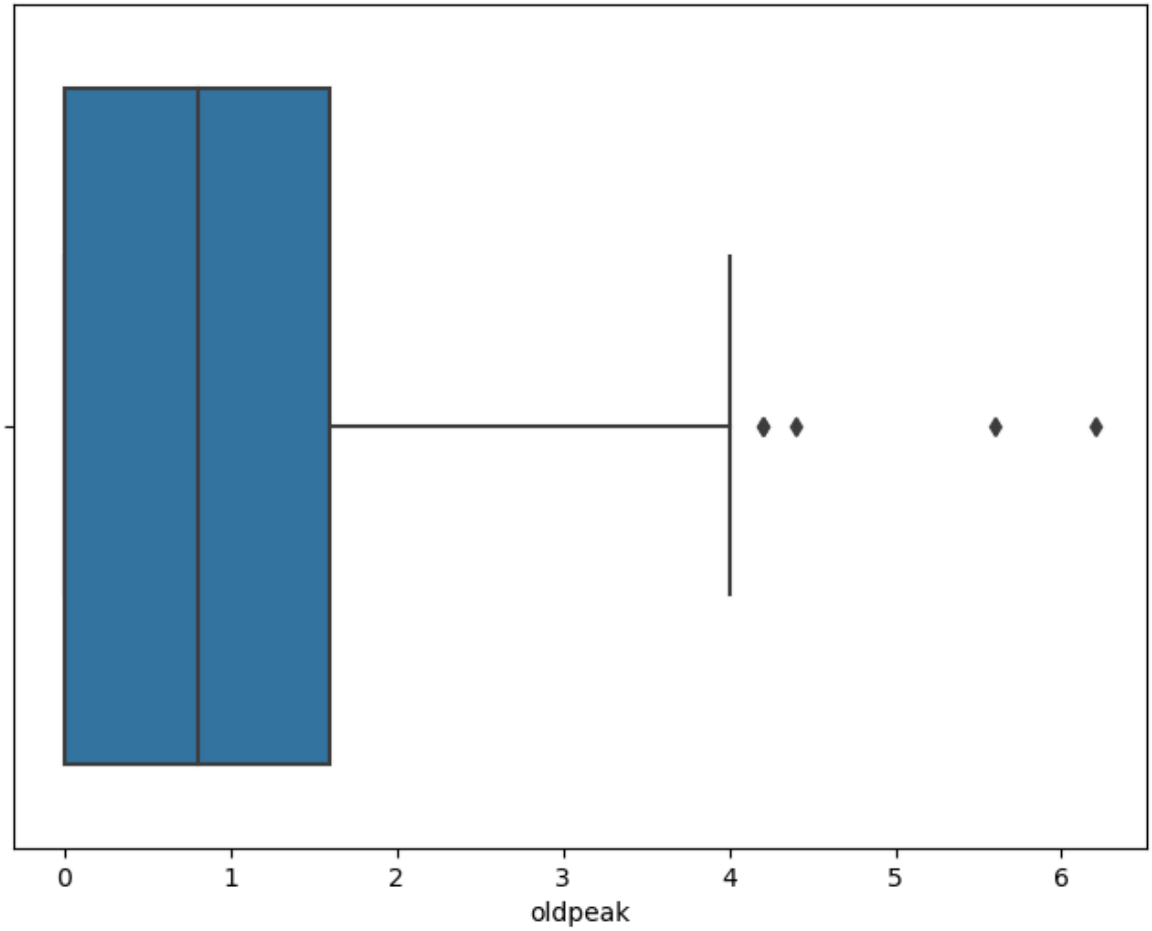
In [79]:
```
# Boxplot of thalach variable

f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["thalach"])
plt.show()
```



In [80]:
```
# oldpeak variable
df['oldpeak'].describe()
```

Out[80]:
```
count    303.000000
mean       1.039604
std        1.161075
min        0.000000
25%        0.000000
50%        0.800000
75%        1.600000
max        6.200000
Name: oldpeak, dtype: float64
```

In [81]:
```
# Box-plot of oldpeak variable

f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["oldpeak"])
plt.show()
```



In [ ]: