Exploratory Data Analysis(EDA)--Heart Disease or Cardiovascular Disease Visualization

In [195…
```
#  Import libraries

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)d
```

In [197…
```
import os
for dirname, _, filenames in os.walk('E:\Data Science & AI\Dataset files\heart.c
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [199…
```
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
%matplotlib inline

sns.set(style="whitegrid")
```

In [201…
```
# ignore warnings

import warnings
warnings.filterwarnings('ignore')
```

In [203…
```
df = pd.read_csv(r'E:\Data Science & AI\Dataset files\heart.csv')
```

In [205…
```
df
```

Out[205…

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	

303 rows × 14 columns

In [207…
```
# Exploratory Data Analysis

print('The shape of the dataset : ', df.shape)  # print the shape
```

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

In [209…
```
df.head()  # preview dataset
```

Out[209…

	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 [211…
```
df.info()  # summary of dataset
```

```
<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 [213…
```
#  Datatype of columns

df.dtypes
```

Out[213…
```
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 [215…
```
# statistical properties of dataset
df.describe()
```

Out[215…

	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 [217…
```
df.columns # view columns
```

Out[217…
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [219…
```
df['target'].nunique() # Check the number of unique values in `target` variable
```

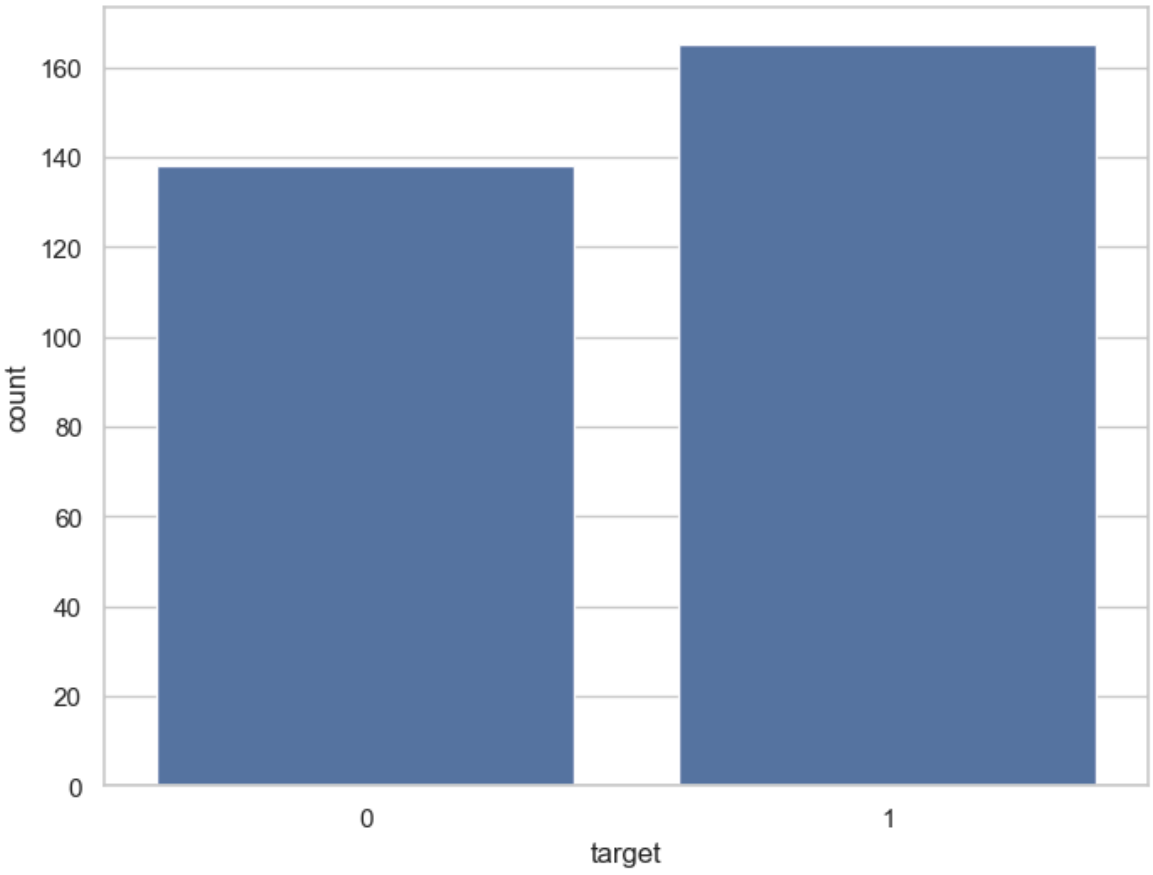
Out[219…
```
2
```

In [221…
```
df['target'].value_counts() # Frequency distribution of `target` variable
```

Out[221…
```
target
1    165
0    138
Name: count, dtype: int64
```

In [223…

```
# Visualize frequency distribution of `target` variable

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



In [225…

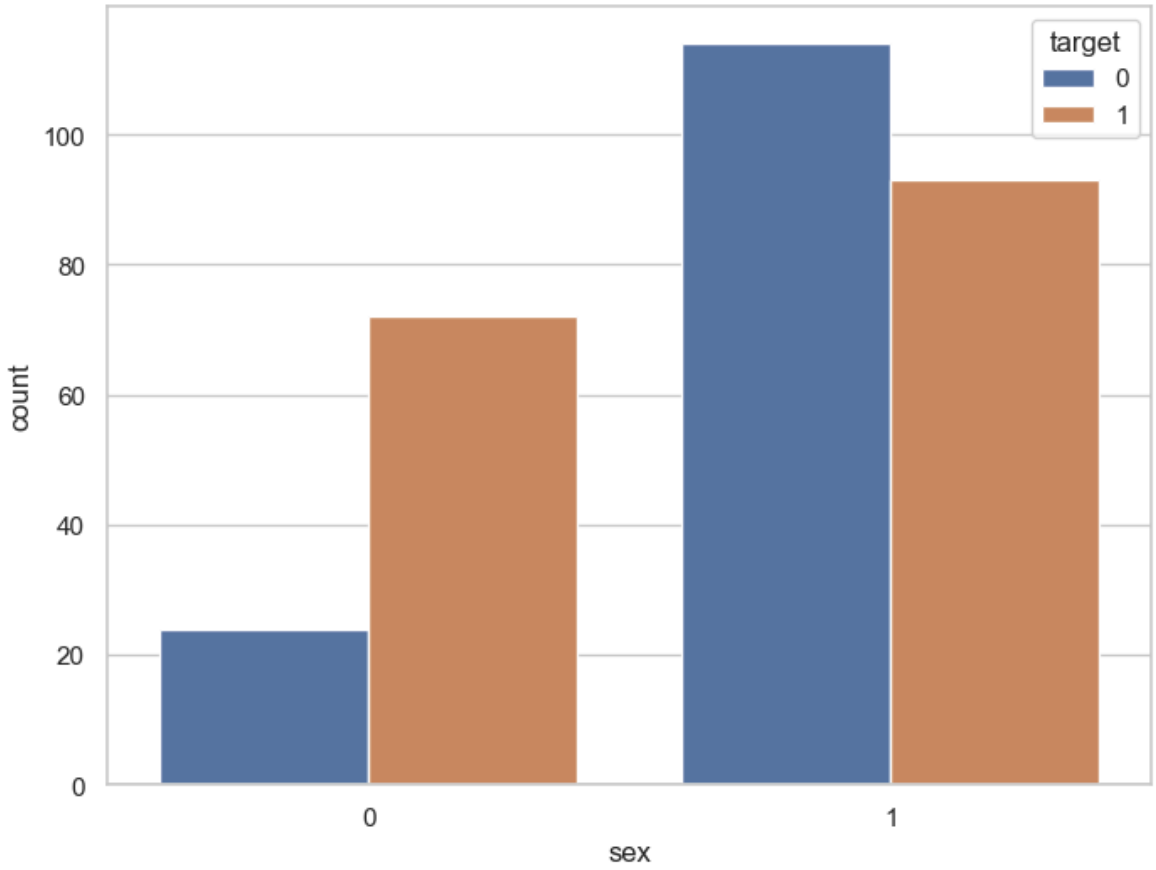
```
# Frequency distribution of `target` variable wrt `sex`

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

Out[225…

```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

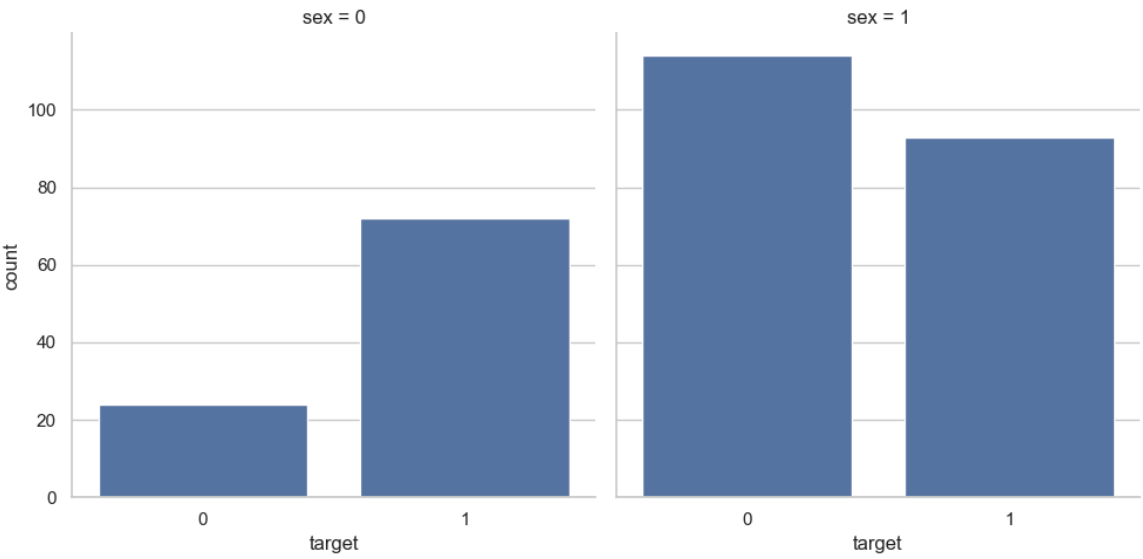
In [227…

```
# visualize the value counts of the `sex` variable wrt `target`

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



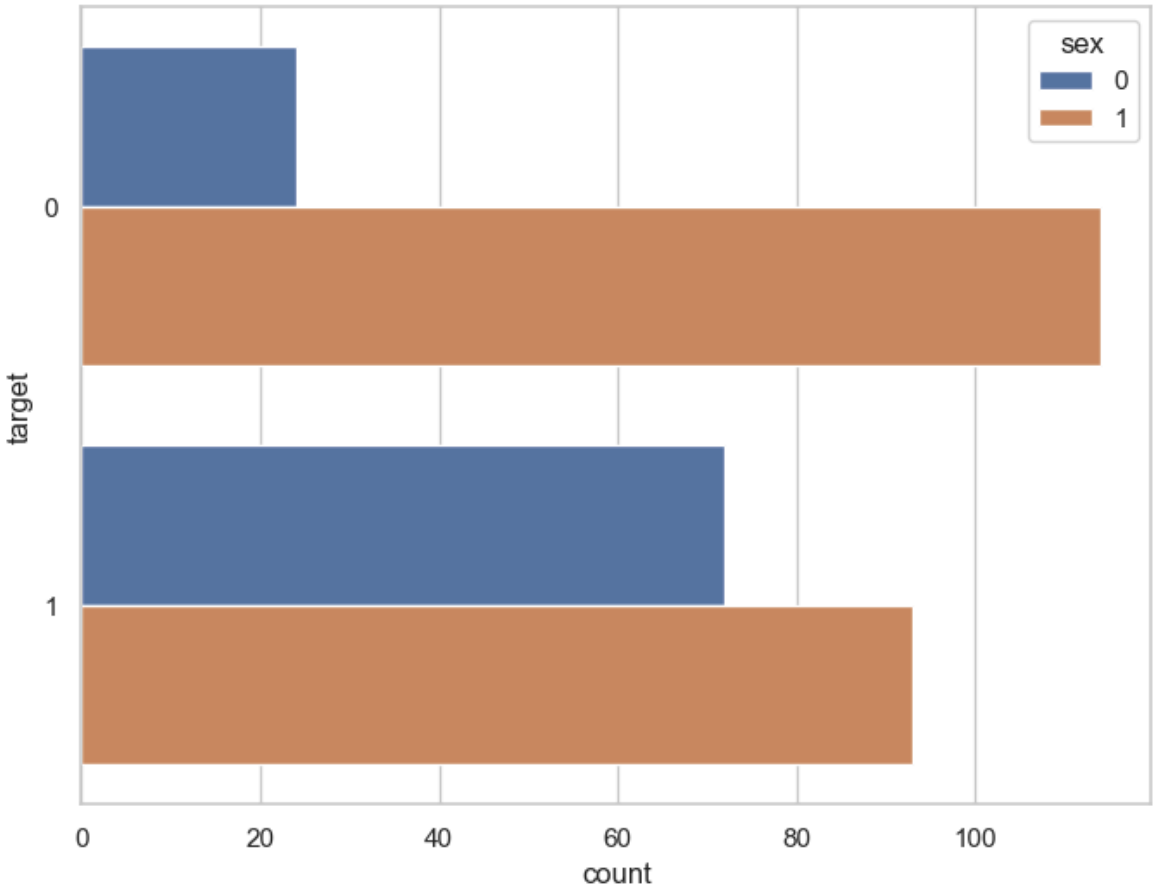
In [229…

```
# Alternatively visualize  the same information

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



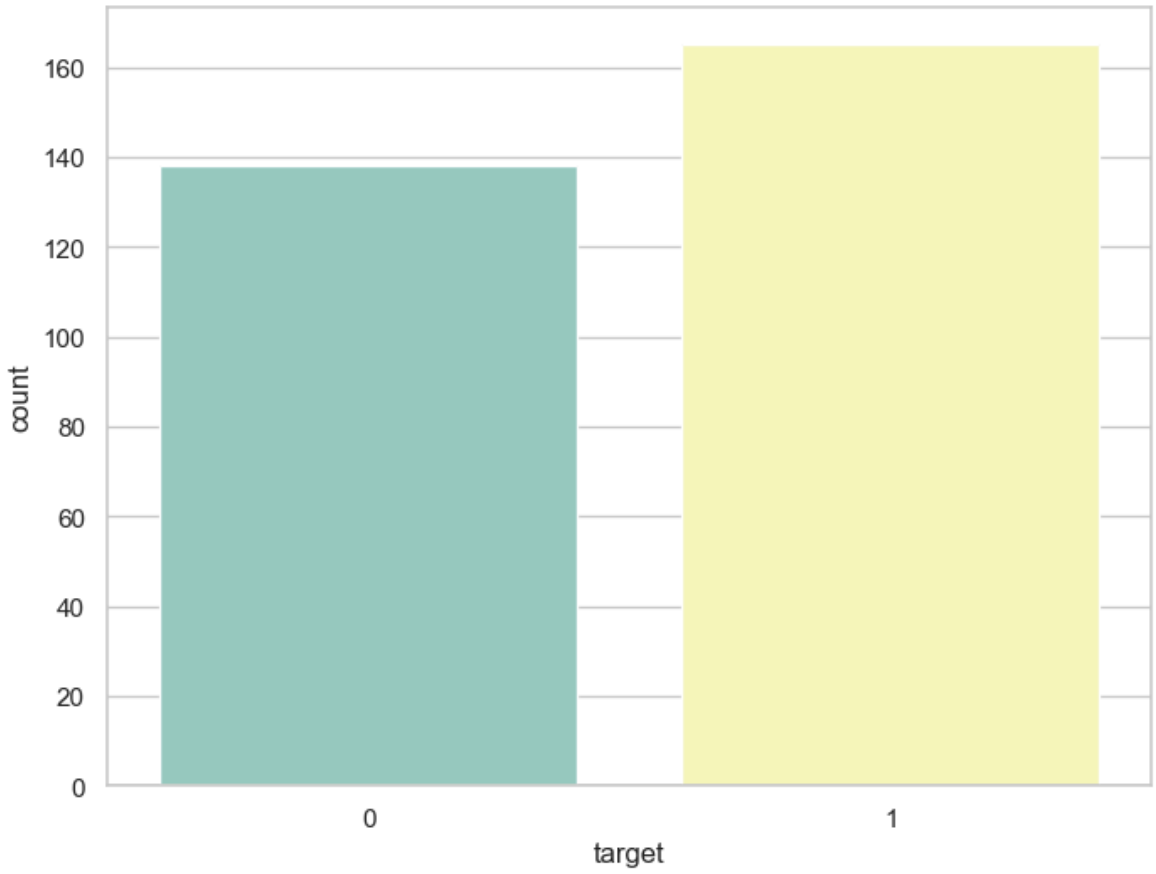
In [231…

```
#   Bars Horizontally

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



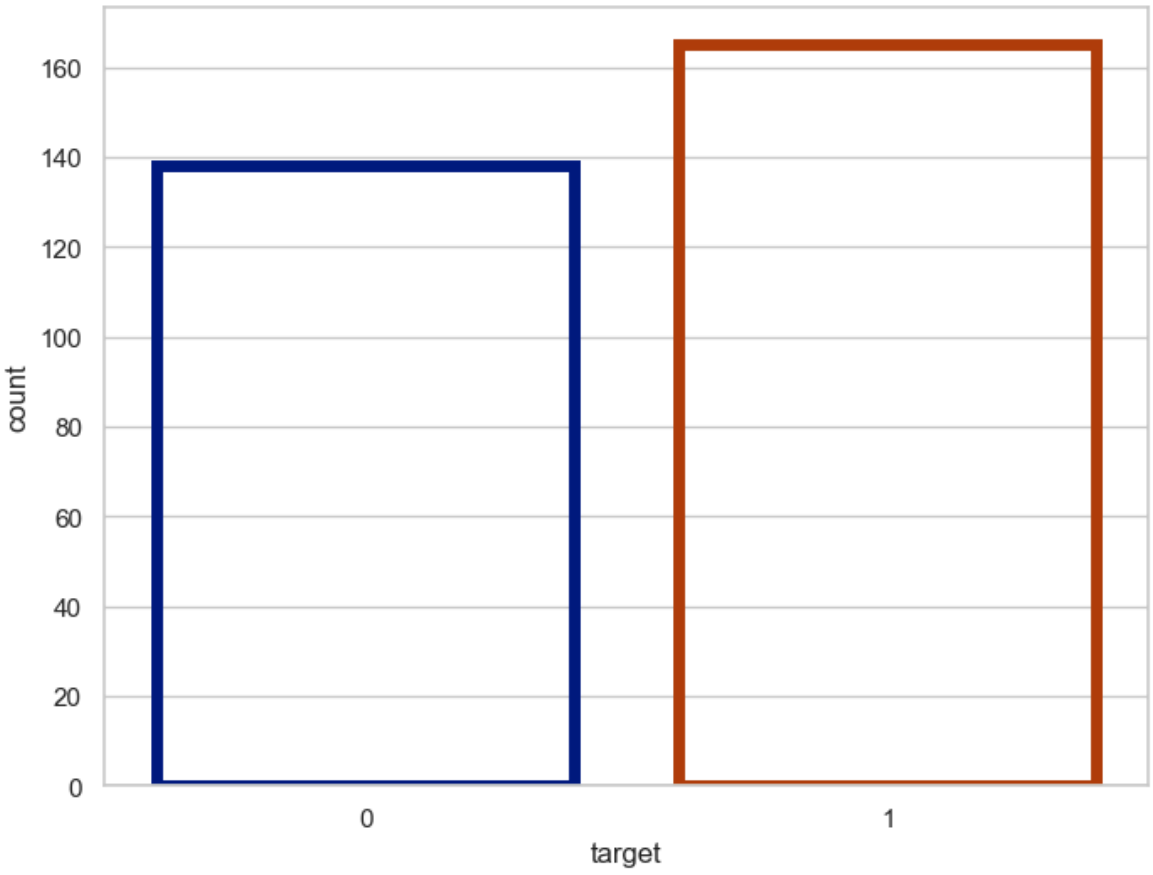
In [235…

```
#  different color palette

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



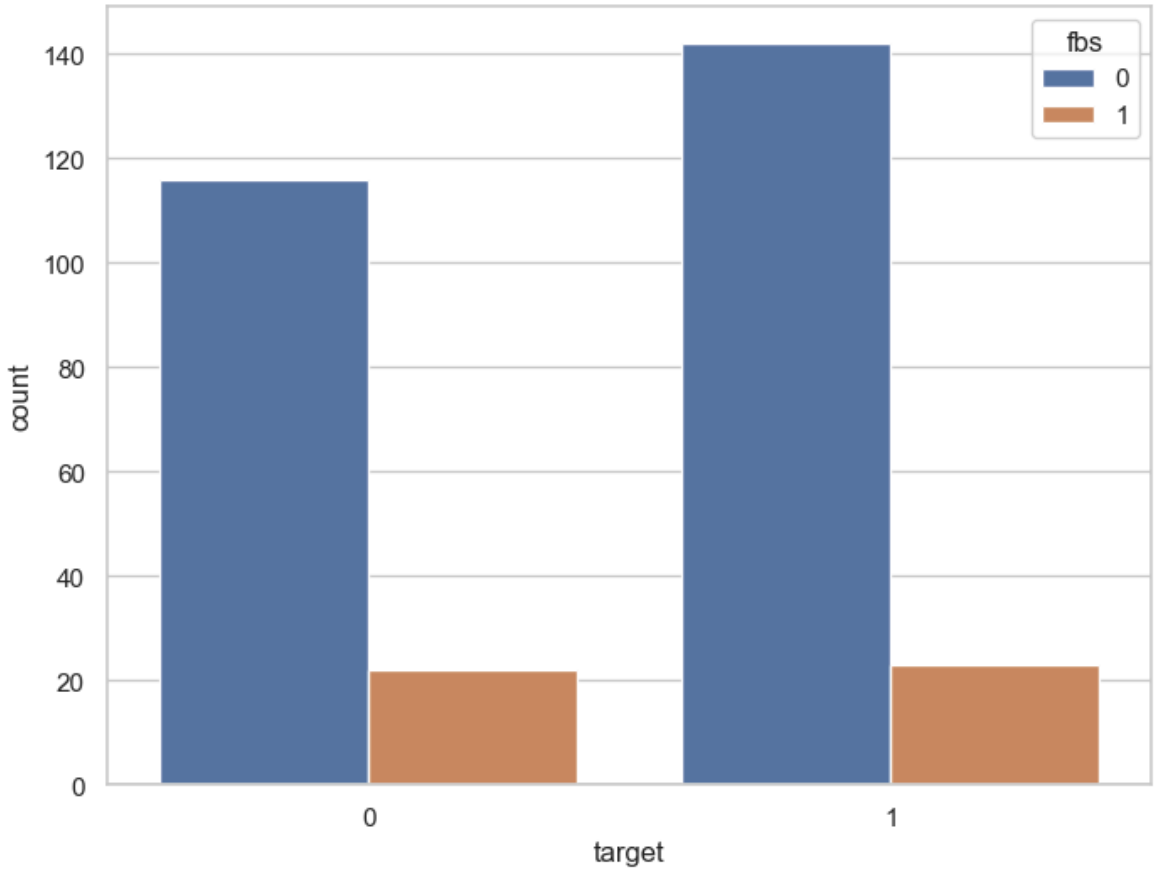
In [237…

```
# plt.bar keyword arguments

f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, facecolor=(0, 0, 0, 0), linewidth=5, edg
plt.show()
```



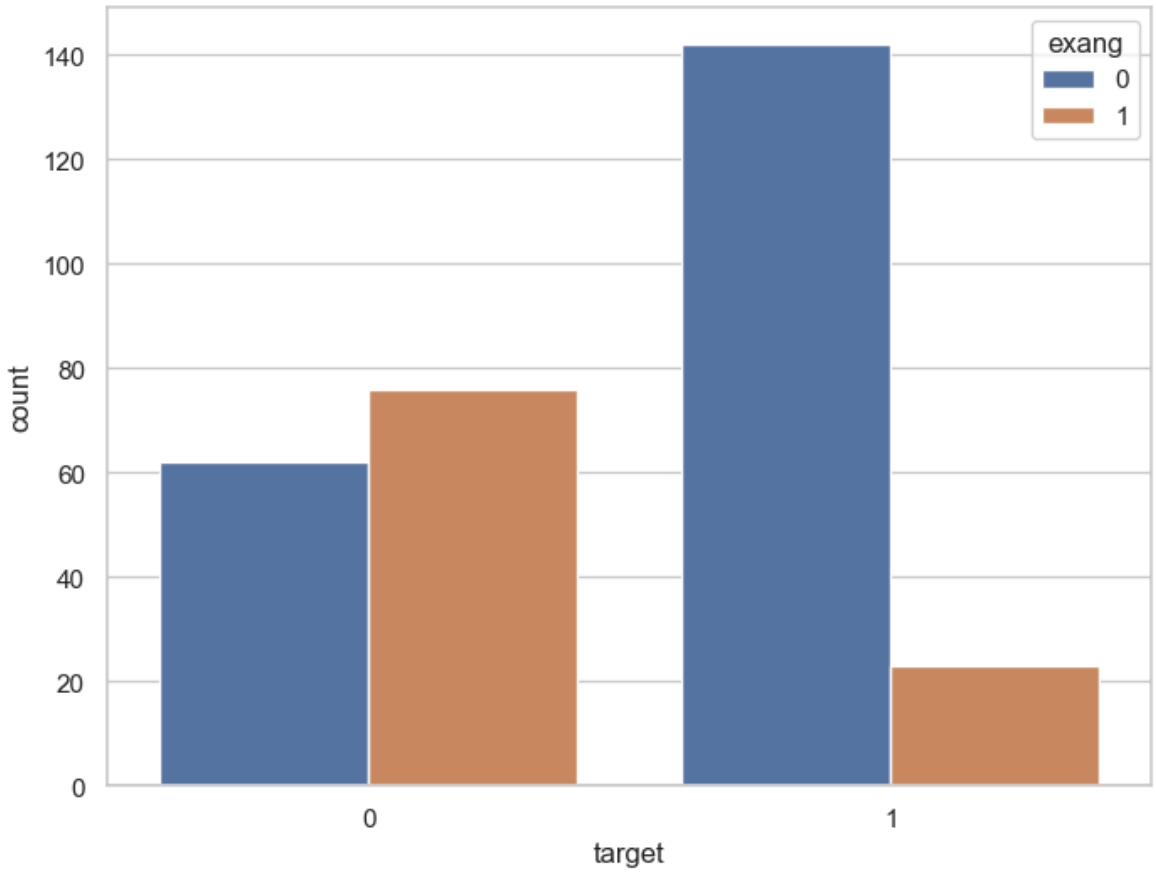
In [239…

```
#  isualize the `target` values distribution wrt `fbs (fasting blood sugar)` and

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



In [241…

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



In [243…

```
#  df.corr()--Estimate correlation coefficients
```

```
correlation = df.corr()
```

In [245…
```
correlation['target'].sort_values(ascending=False)
```

Out[245…
```
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
```

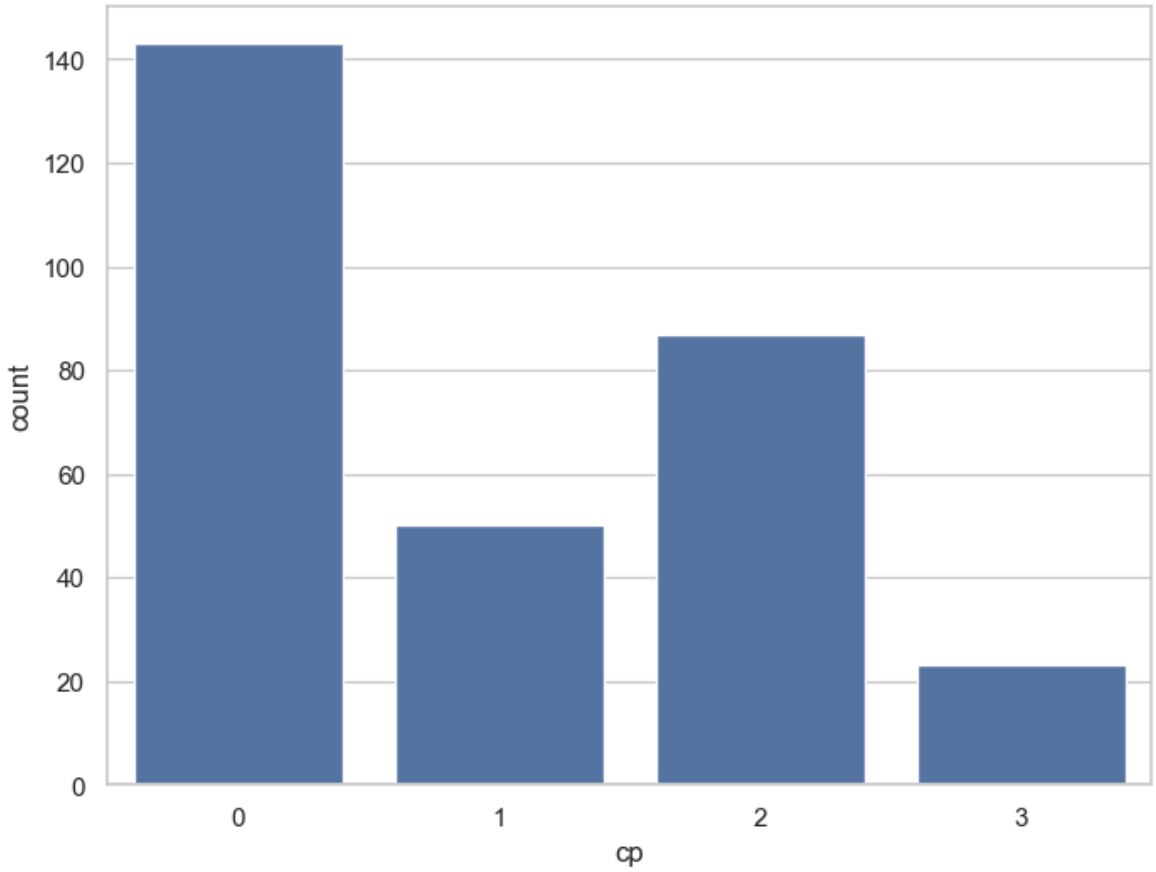
In [247…
```
# Analysis of `target` and `cp` variable
    #  Explore `cp` variable

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

Out[247…
```
4
```

In [249…
```
# Frequency Distribution

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

Out[249…
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

In [251…
```
# frequency distribution of `cp` variable

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



In [253…

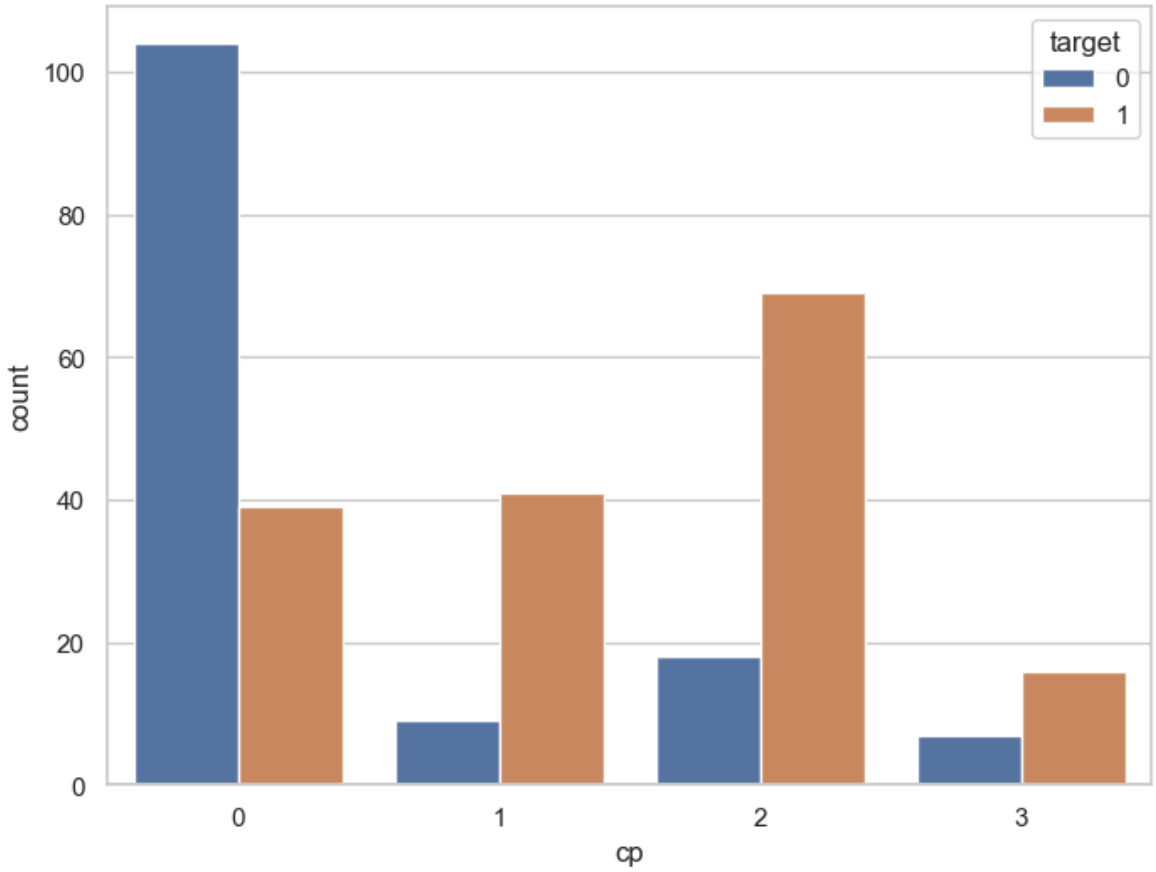
```
# Frequency distribution of `target` variable wrt `cp

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

Out[253…

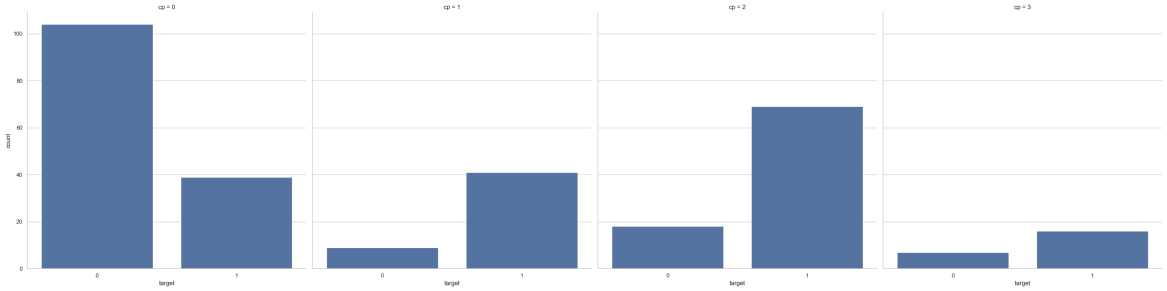
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: count, dtype: int64
```

In [255…

```
#  visualize 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()
```



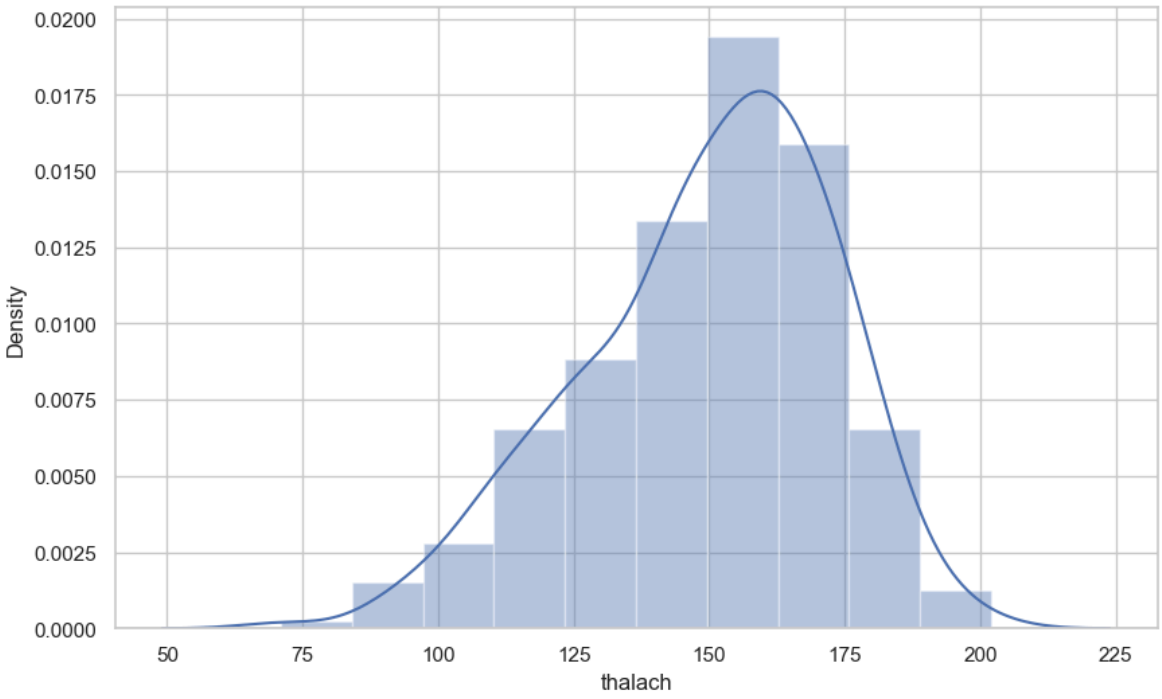
In [257…
```
#  Alternatively visualize the same information

ax = sns.catplot(x="target", col="cp", data=df, kind="count", height=8, aspect=1
```



In [259…
```
#  Analysis of `target` and `thalach` variable
    ##  Explore `thalach` variable
df['thalach'].nunique()
```

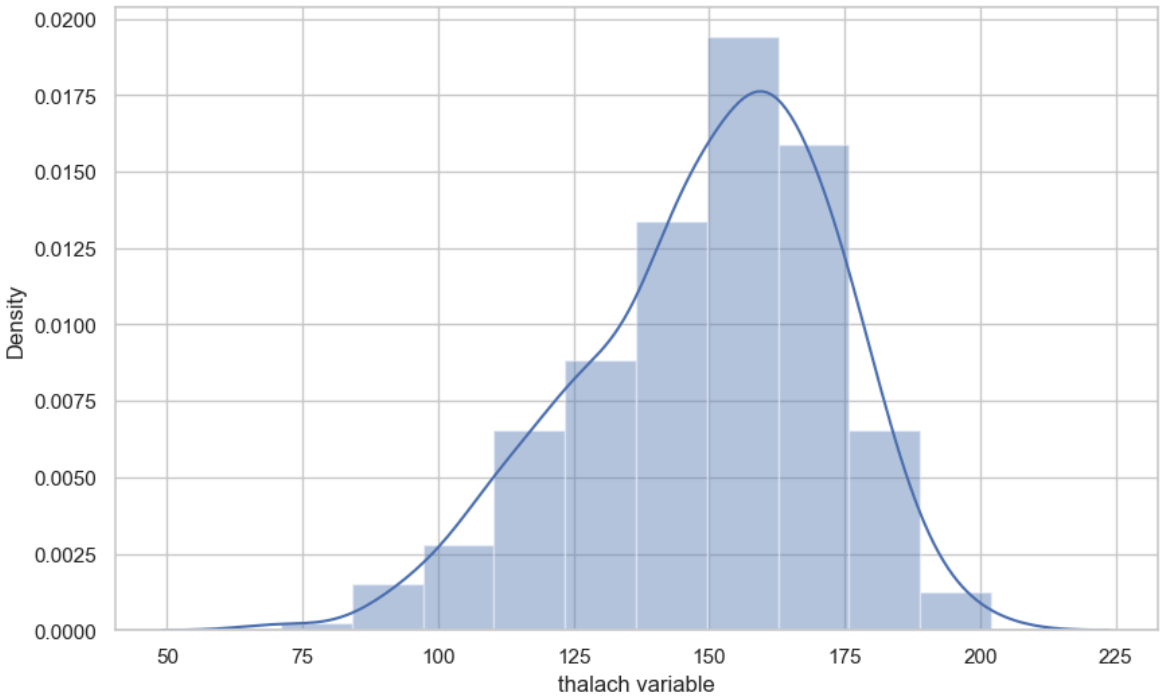
Out[259… 91

In [261…
```
#  frequency distribution of `thalach` variable

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



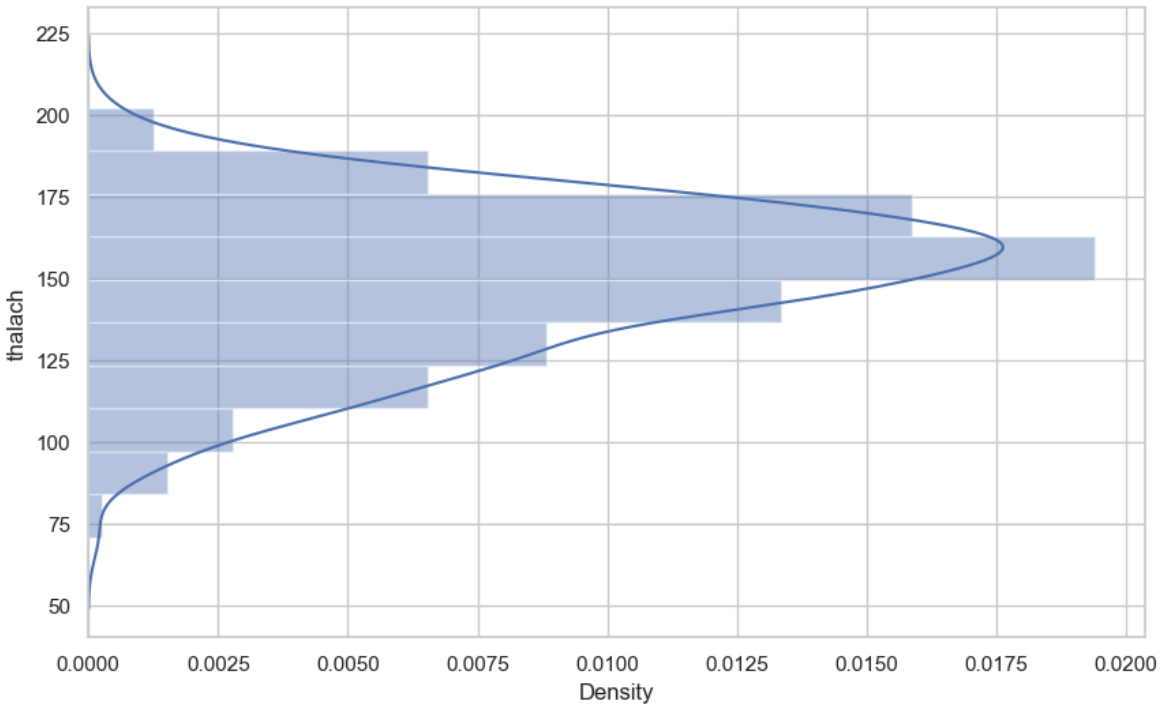
In [263…

```
#  the `thalach` variable is slightly negatively skewed

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



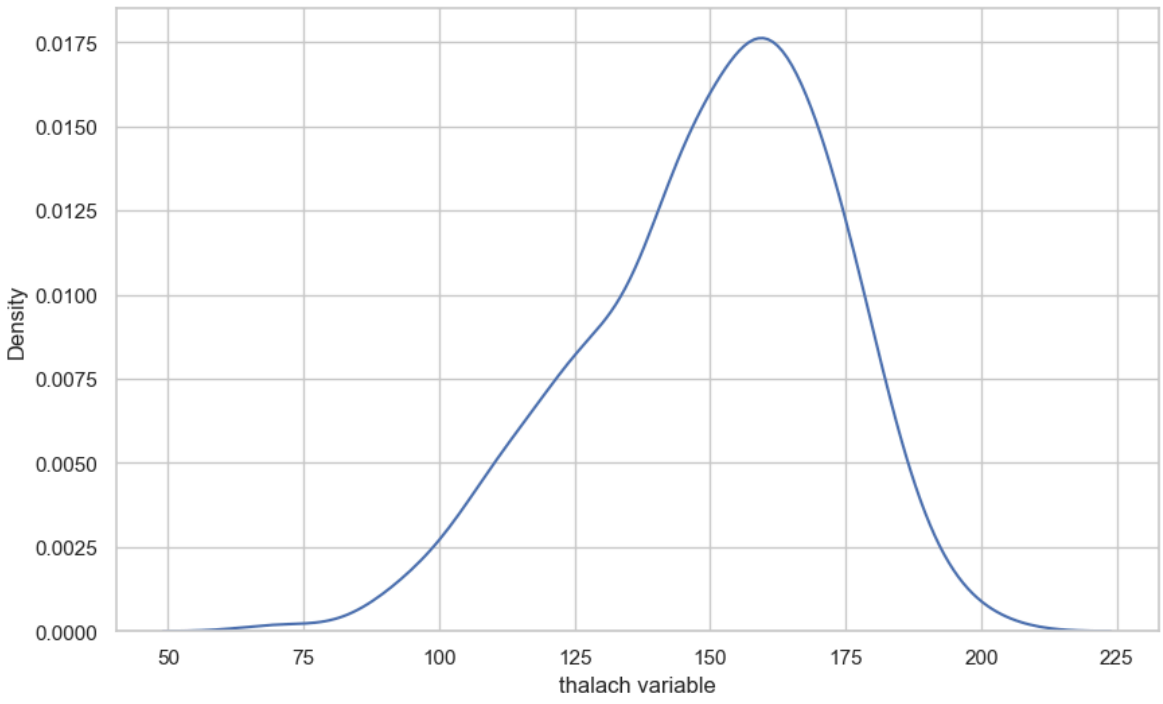
In [265…

```
#  plot the distribution on the vertical axis

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



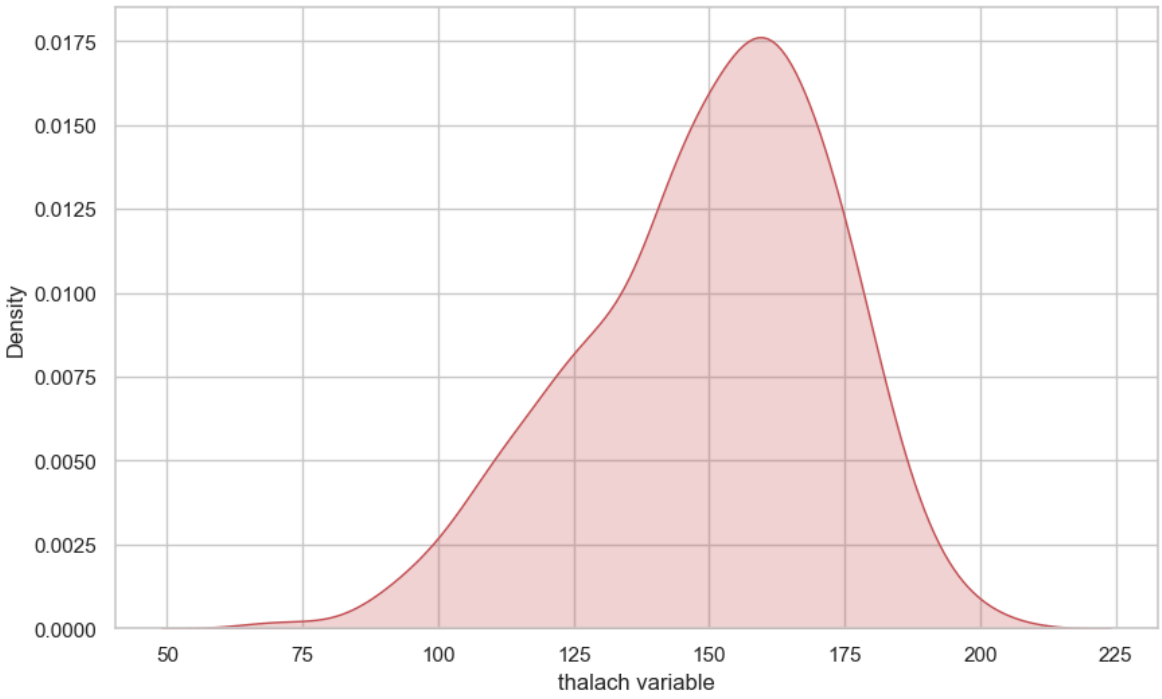
In [267…

```
#  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()
```



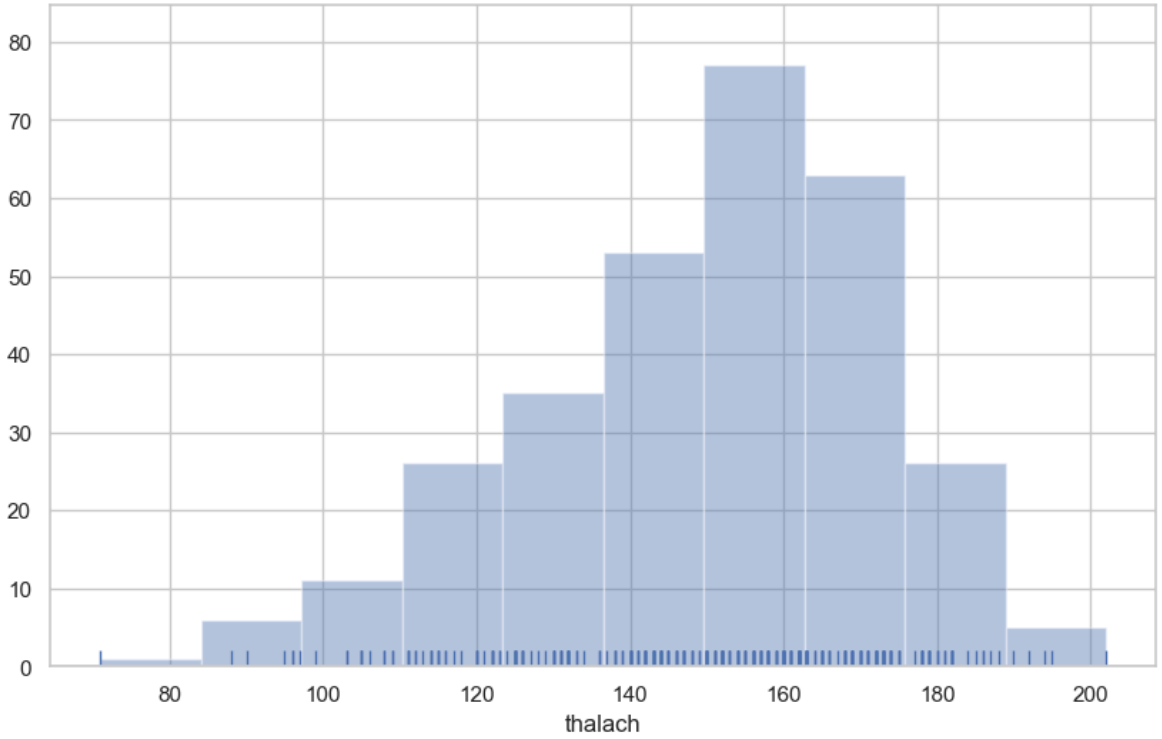
In [269…

```
#  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()
```



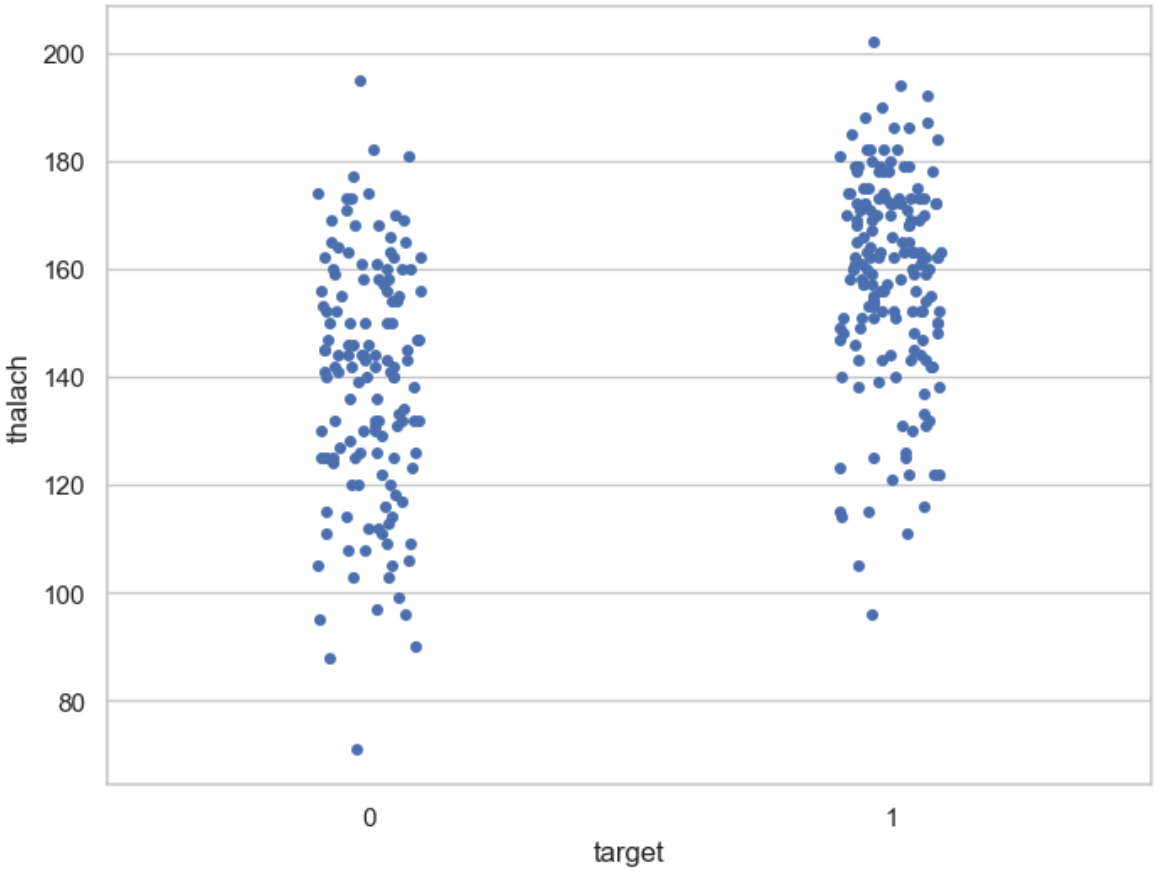
In [271…

```
#  Histogram

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



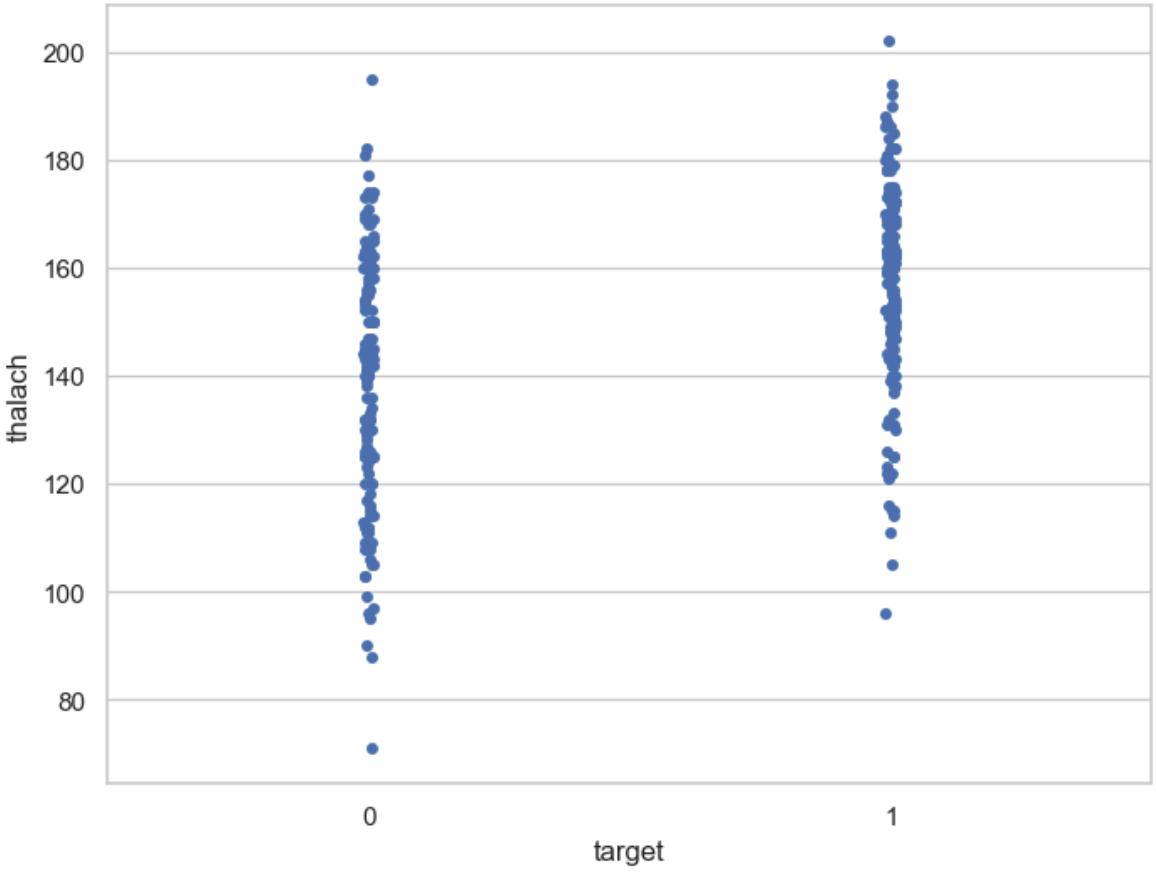
In [273…

```
#  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()
```



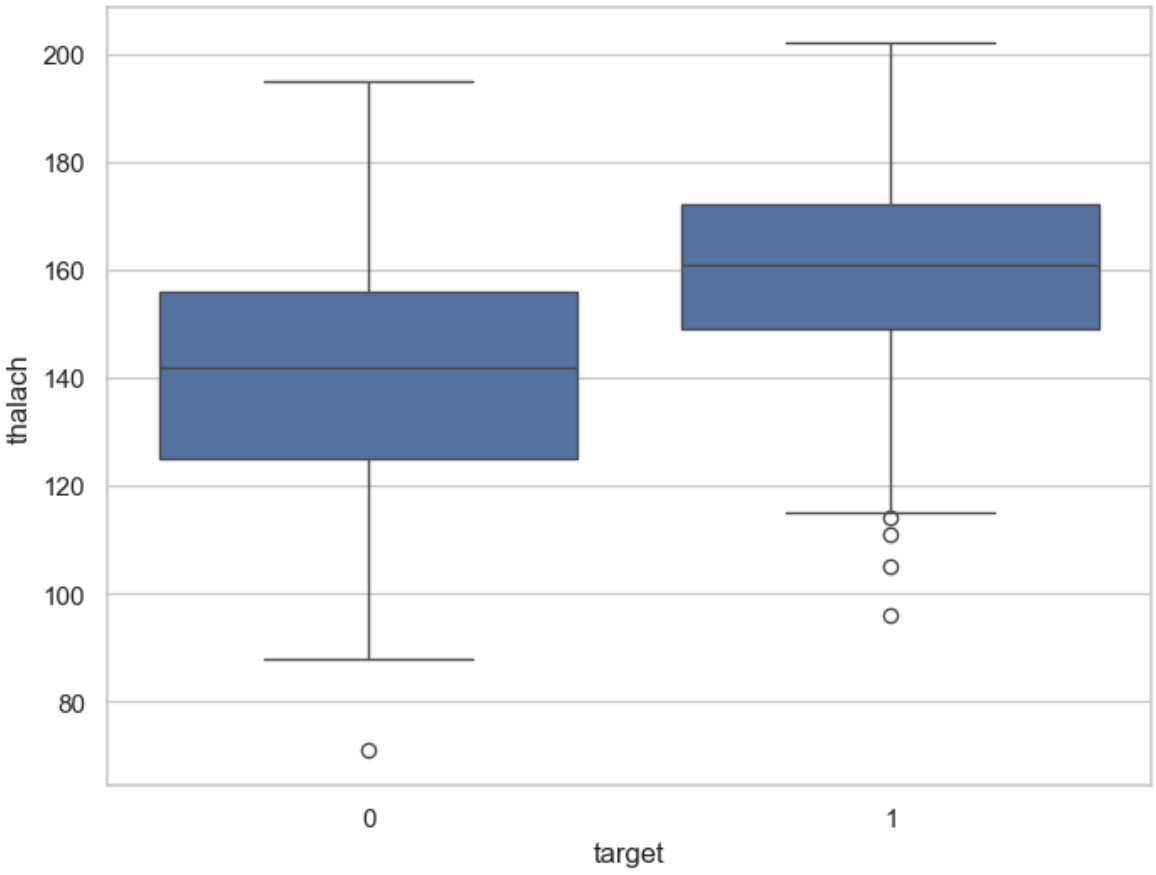
In [275…

```
#  add 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()
```



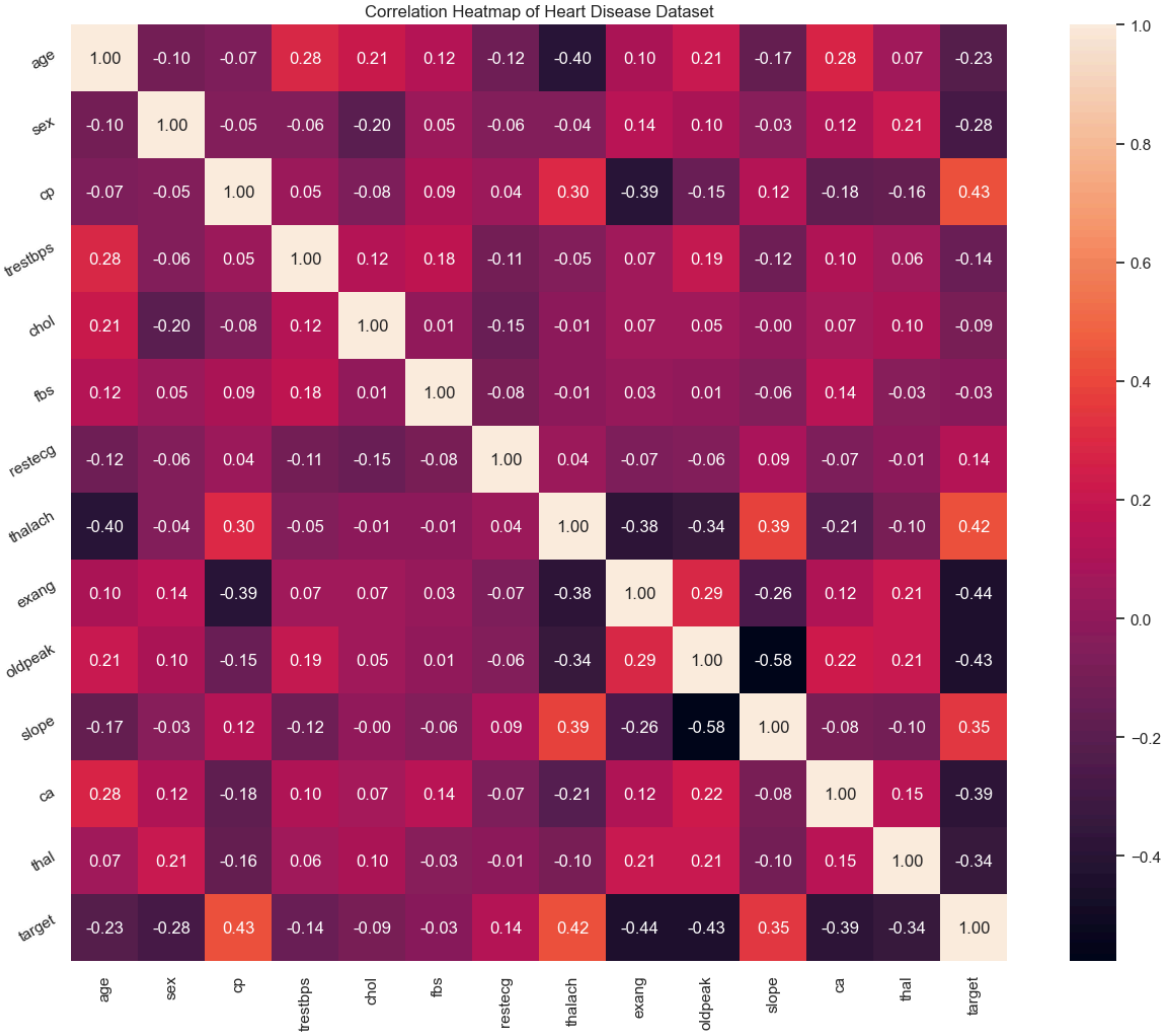
In [277…

```
#  thalach` variable wrt `target` with boxplot

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



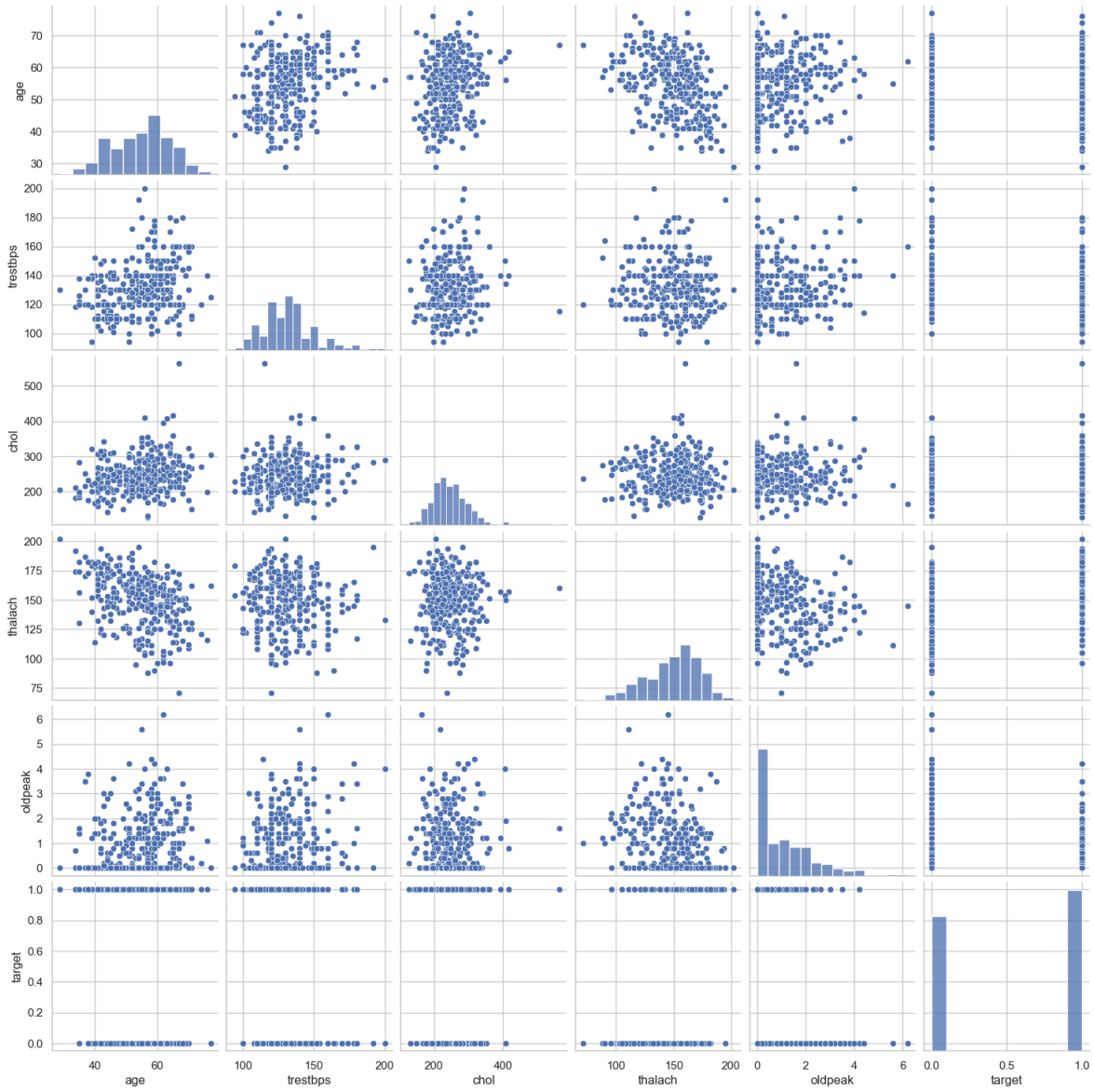
In [279…

```
#  Head 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 [281…

```
#  Pair plot

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



In [283…
```
#  Check the number of unique values in `age` variable

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

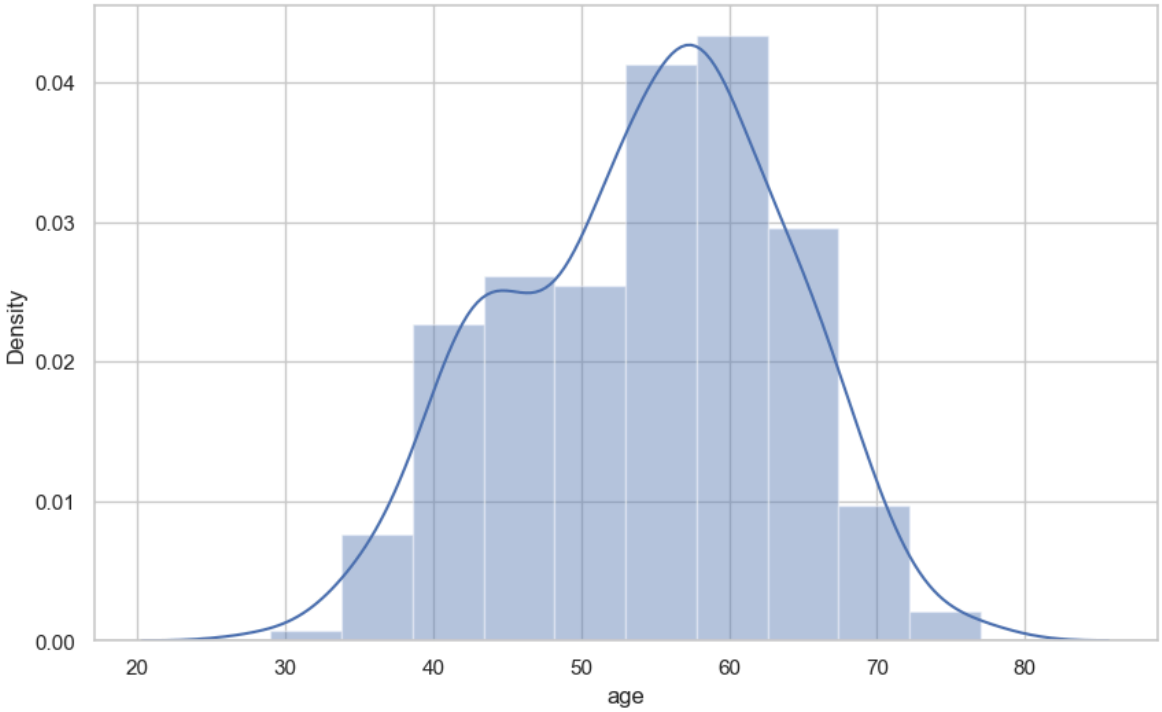
Out[283…
```
41
```

In [285…
```
#  View statistical summary of `age` variable

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

Out[285…
```
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
```

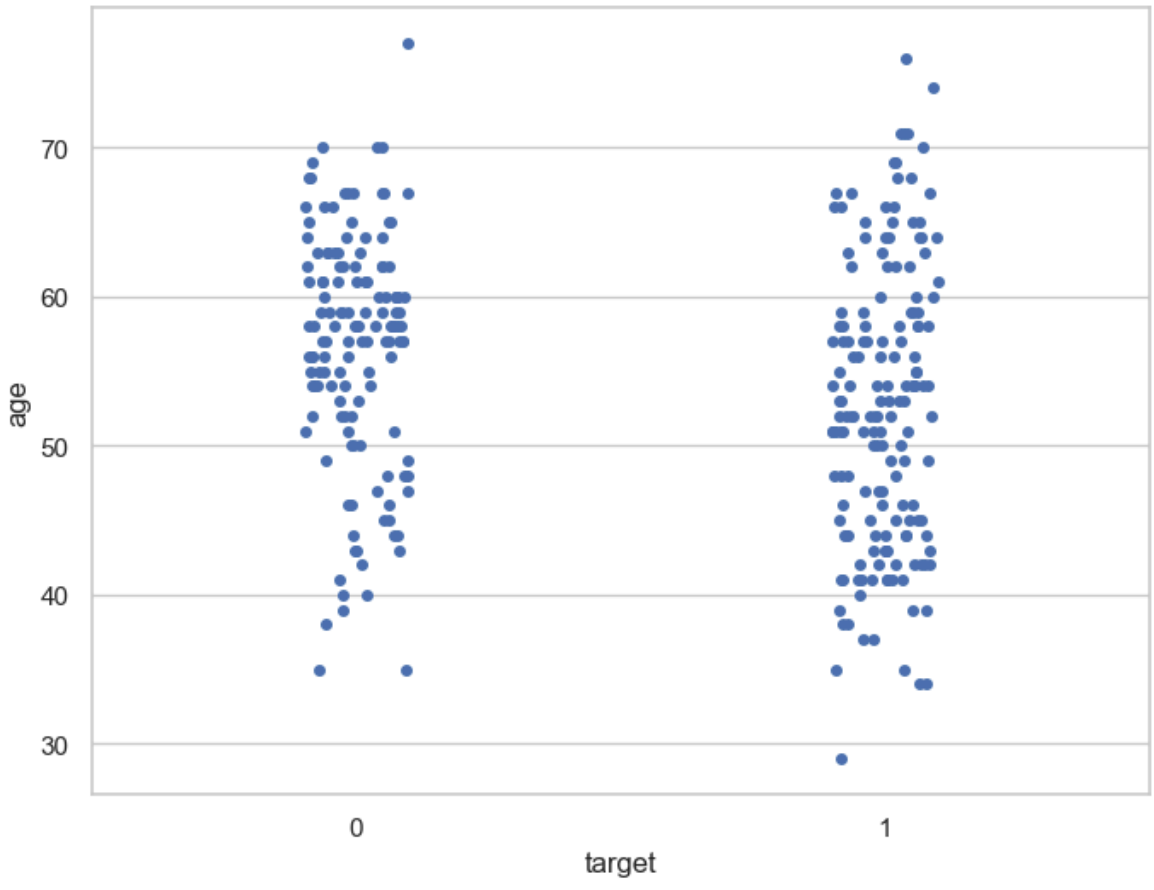
In [287…
```
#  Plot the distribution of `age` variable

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

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



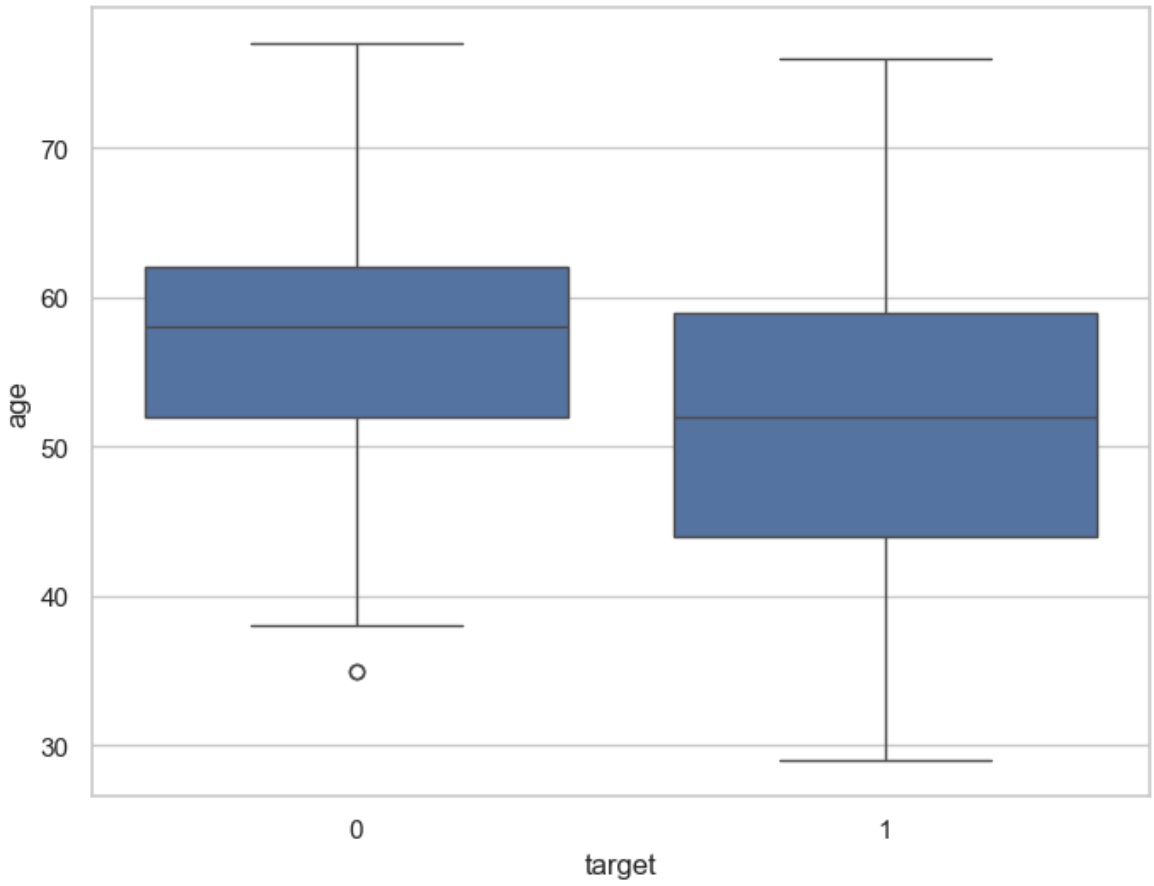
In [289…

```
#  Visualize frequency distribution of `age` variable wrt `target`

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



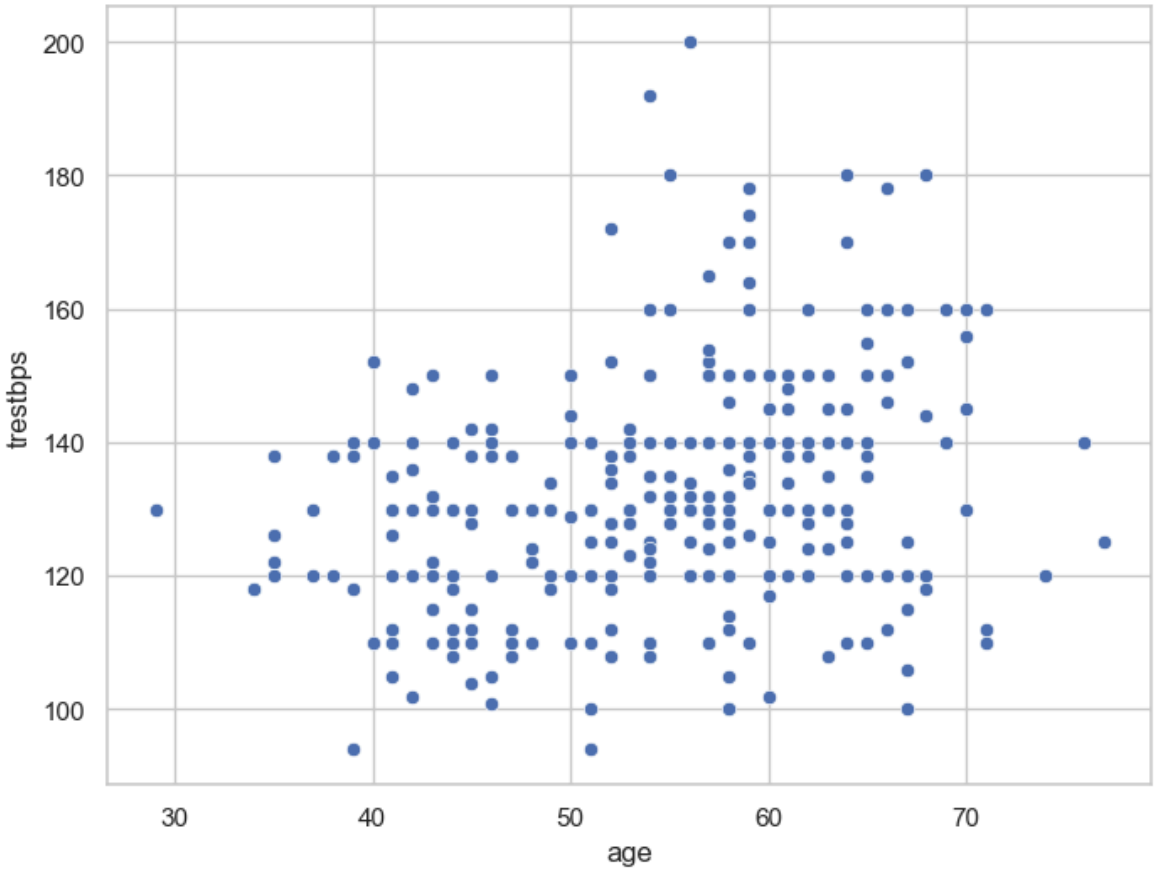
In [291…

```
# Visualize 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()
```



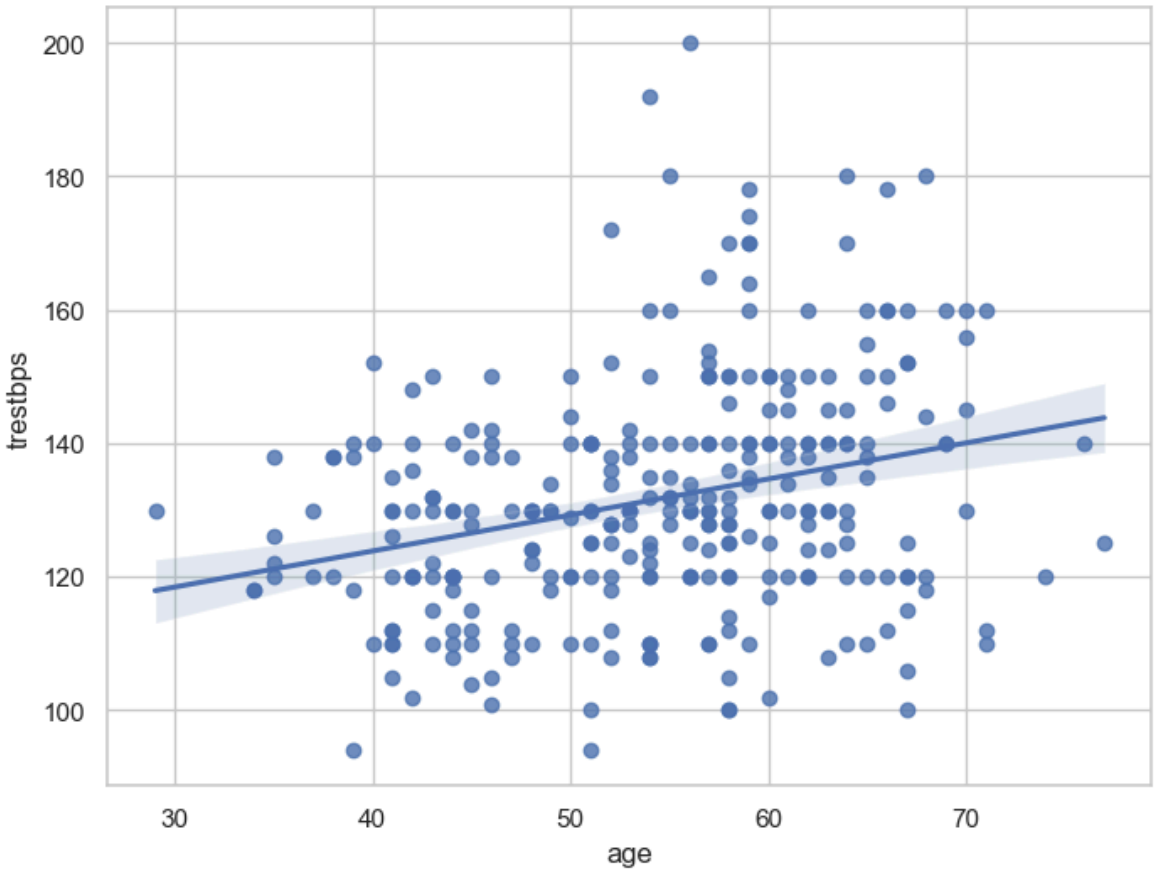
In [293…

```
#  Analyze `age` and `trestbps` variable

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



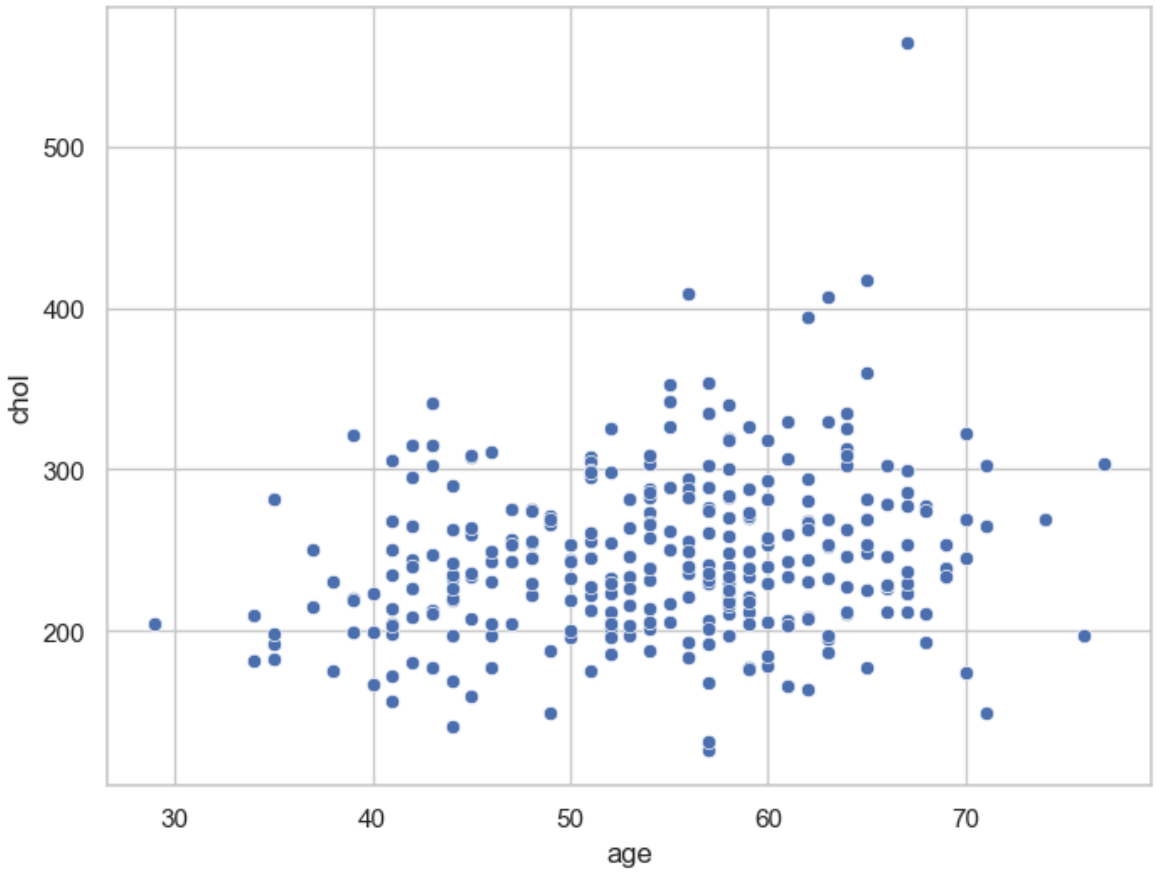
In [295…

```
# scatter plot shows that there is no correlation between `age` and `trestbps` v

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



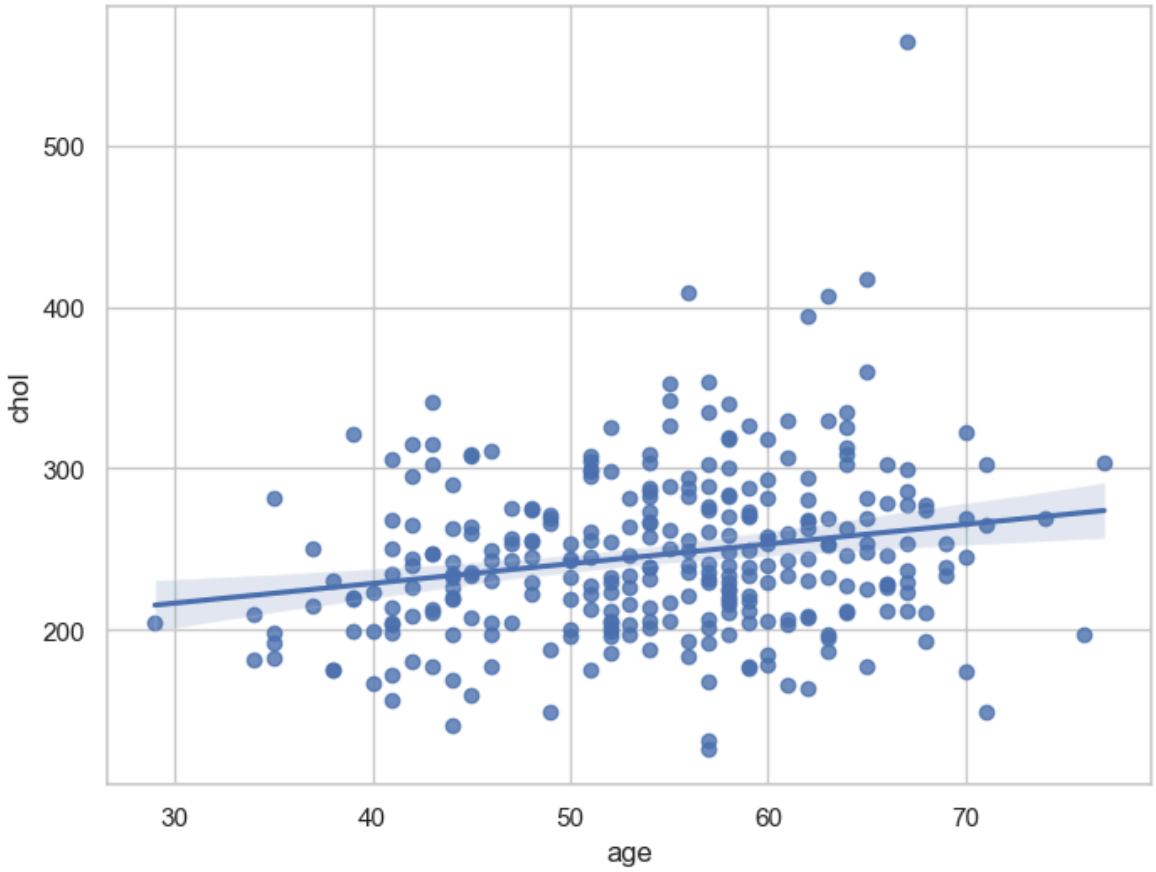
In [297…

```
# Analyze `age` and `chol` variable

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



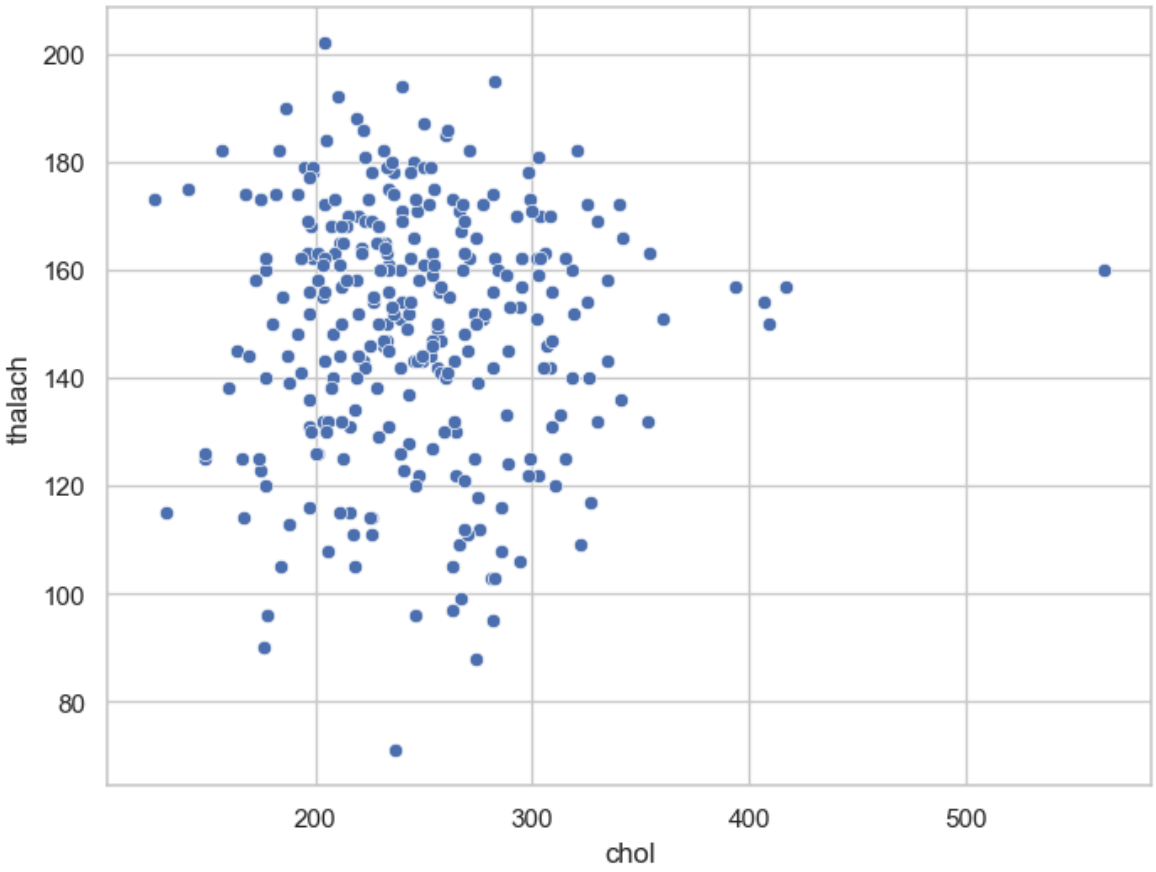
In [299…

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

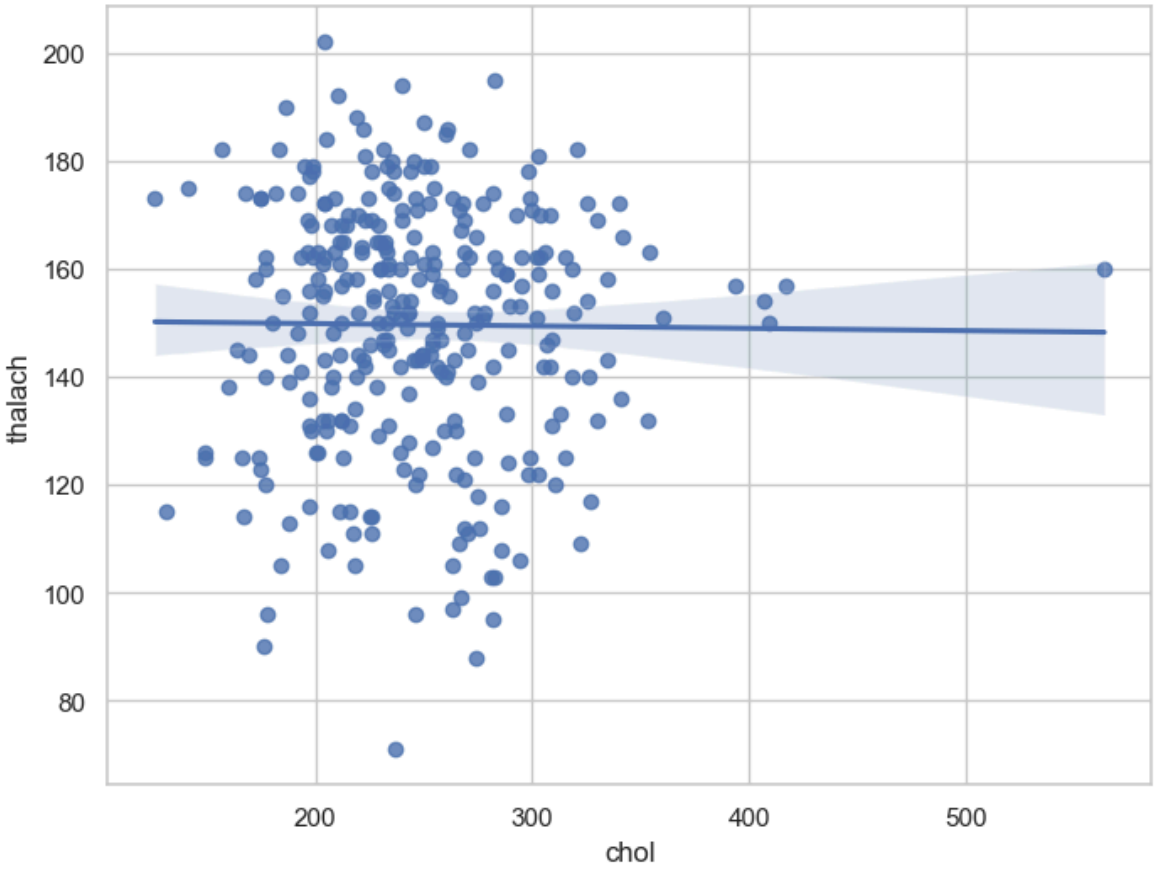


In [301…

```
# Analyze `chol` and `thalach` variable

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



In [303…]
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="chol", y="thalach", data=df)
plt.show()
```



In [305…]
```
# check for missing values

df.isnull().sum()
```

Out[305…]
```
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 [307…]
```
#assert that there are no missing values in the dataframe

assert pd.notnull(df).all().all()
```

In [309…]
```
#assert all values are greater than or equal to 0

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

In [311…

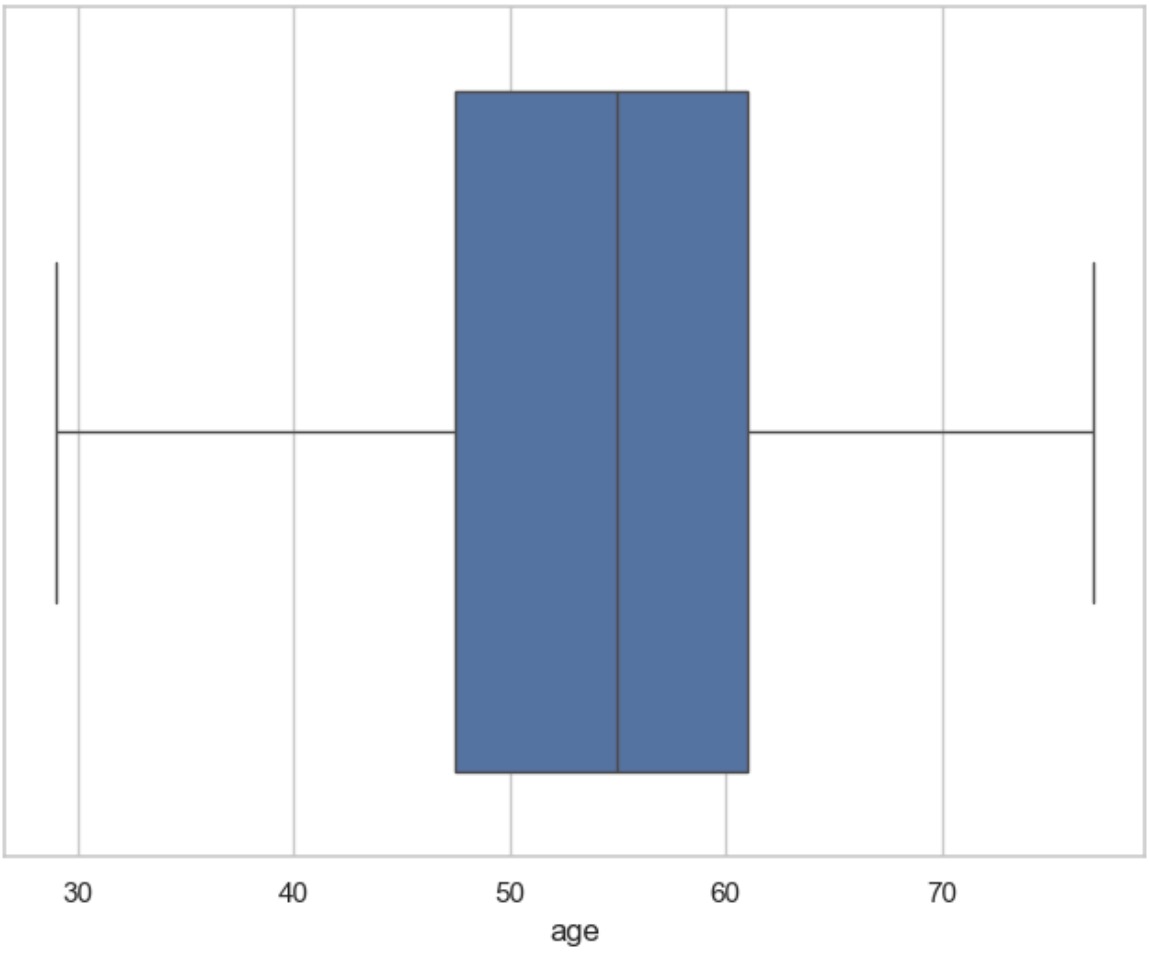
```
#  Outlier detection

## I will make boxplots to visualise outliers in the continuous numerical variab

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

Out[311…

```
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 [313…

```
#  Box-plot of `age` variable

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



In [315…

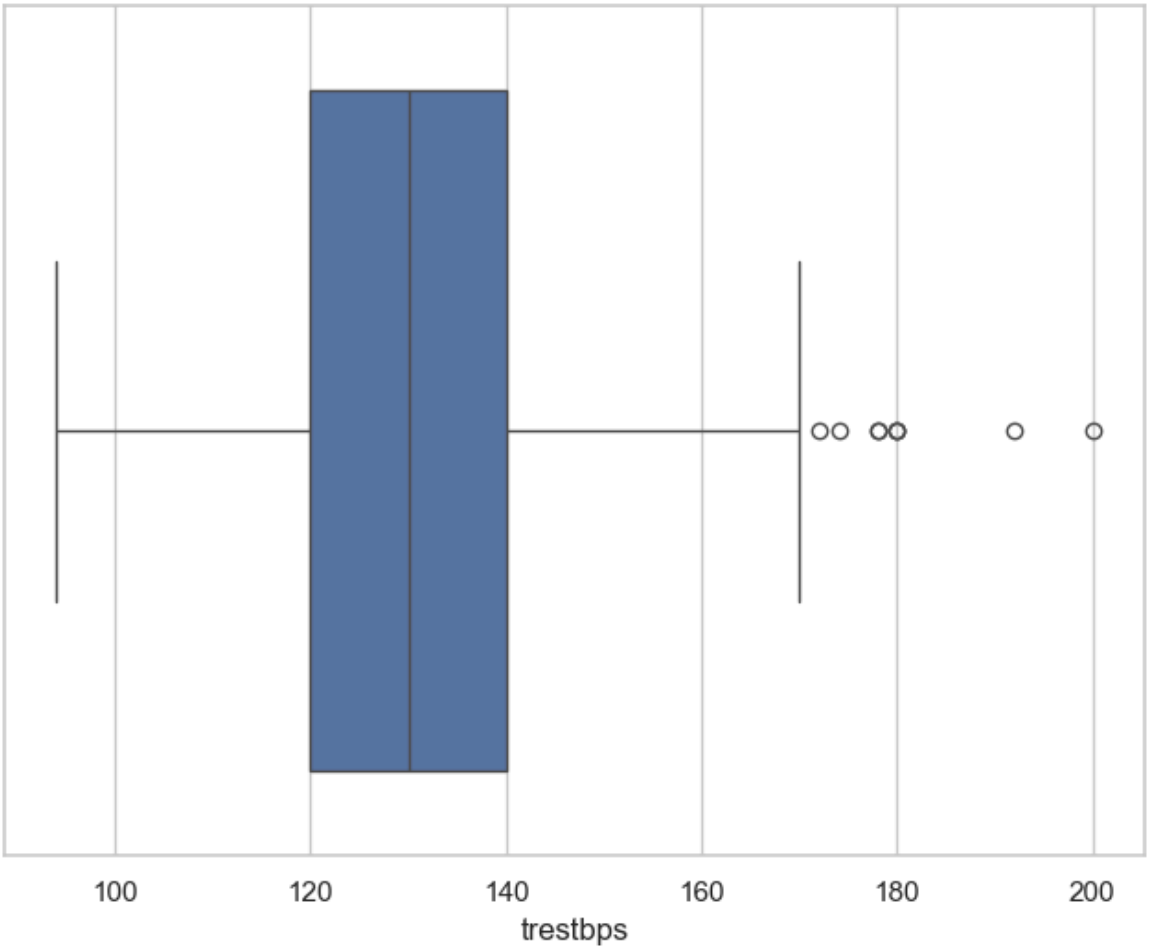
```
# `trestbps` variable

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

Out[315…

```
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 [317…

```
# Box-plot of `trestbps` variable

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



In [319…

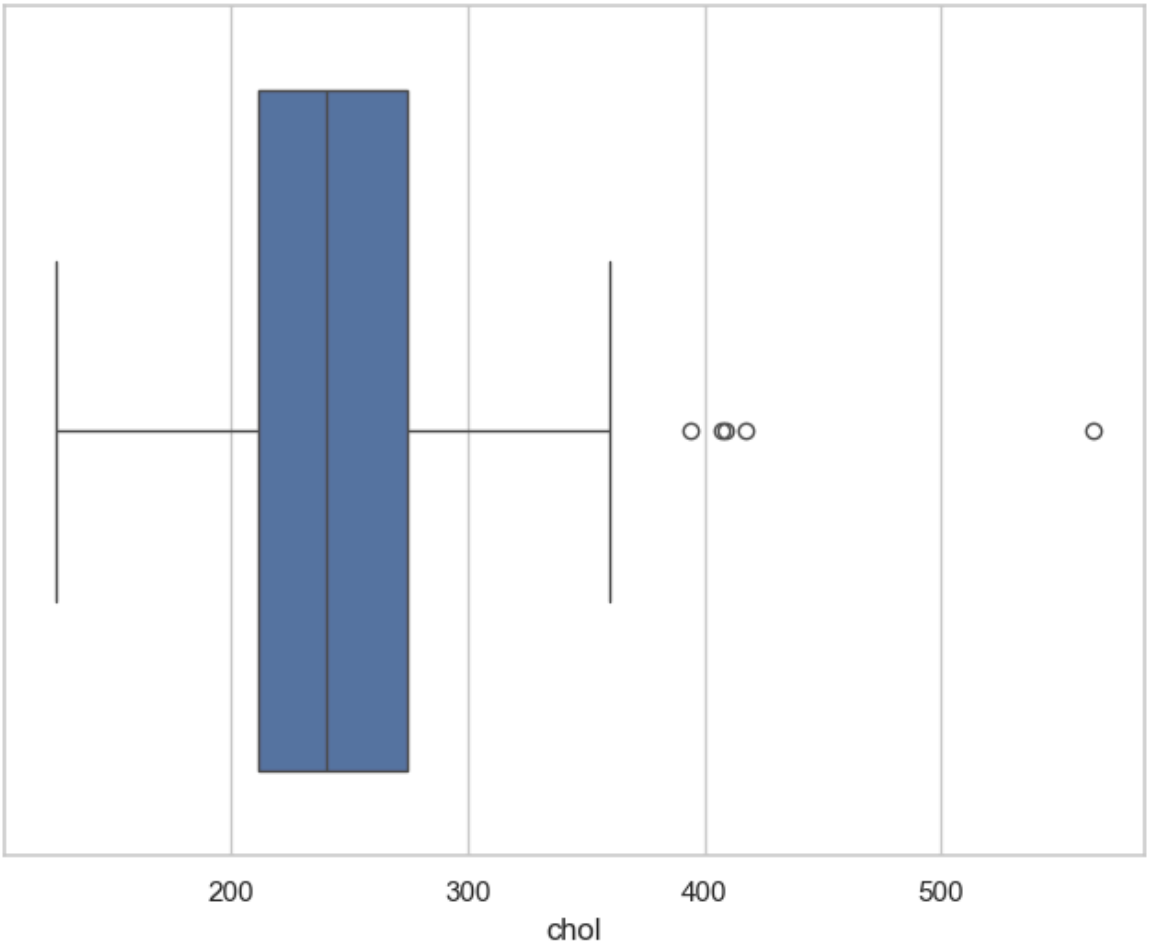
```
#  `chol` variable

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

Out[319…

```
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 [321…

```
#  Box-plot of `chol` variable

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



In [323…

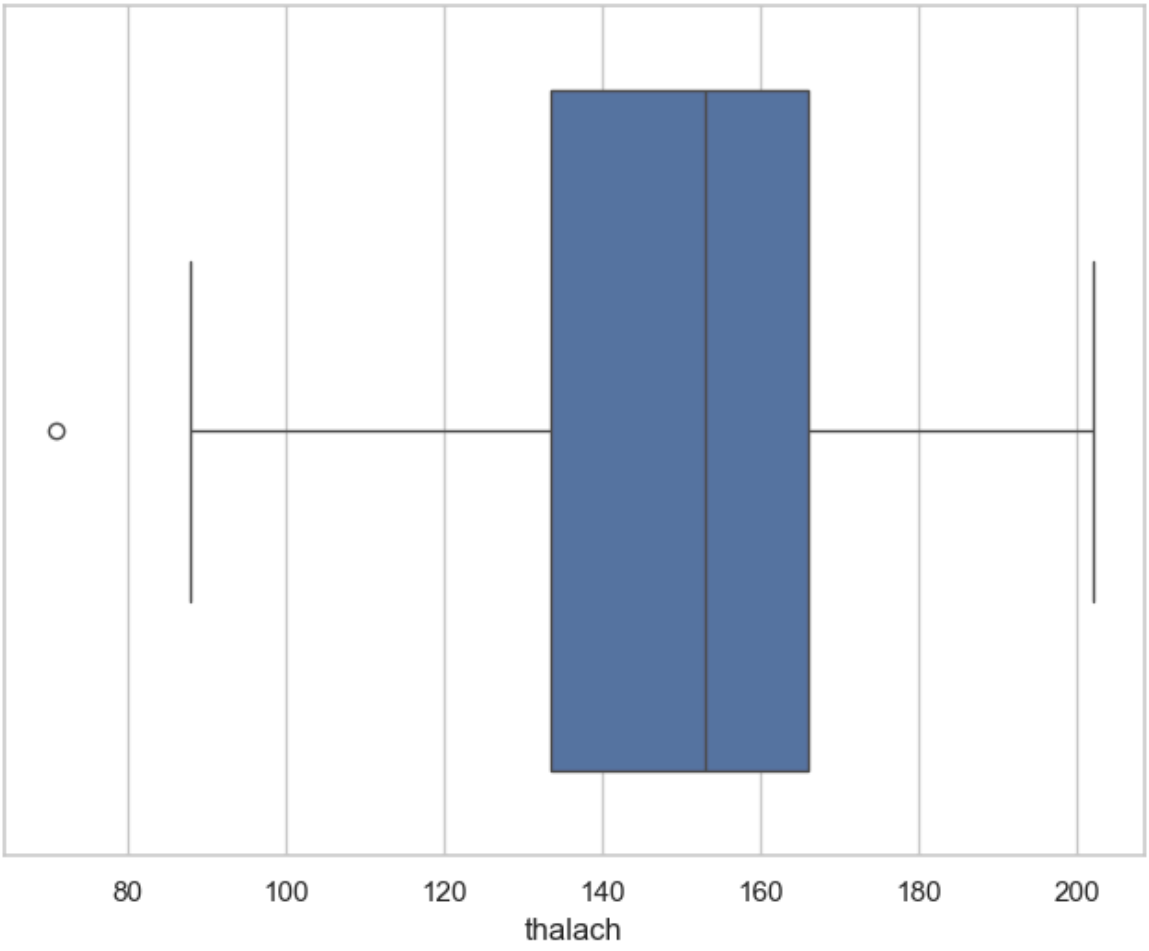
```
# `thalach` variable

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

Out[323…

```
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 [325…

```
# Box-plot of `thalach` variable

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



In [327…

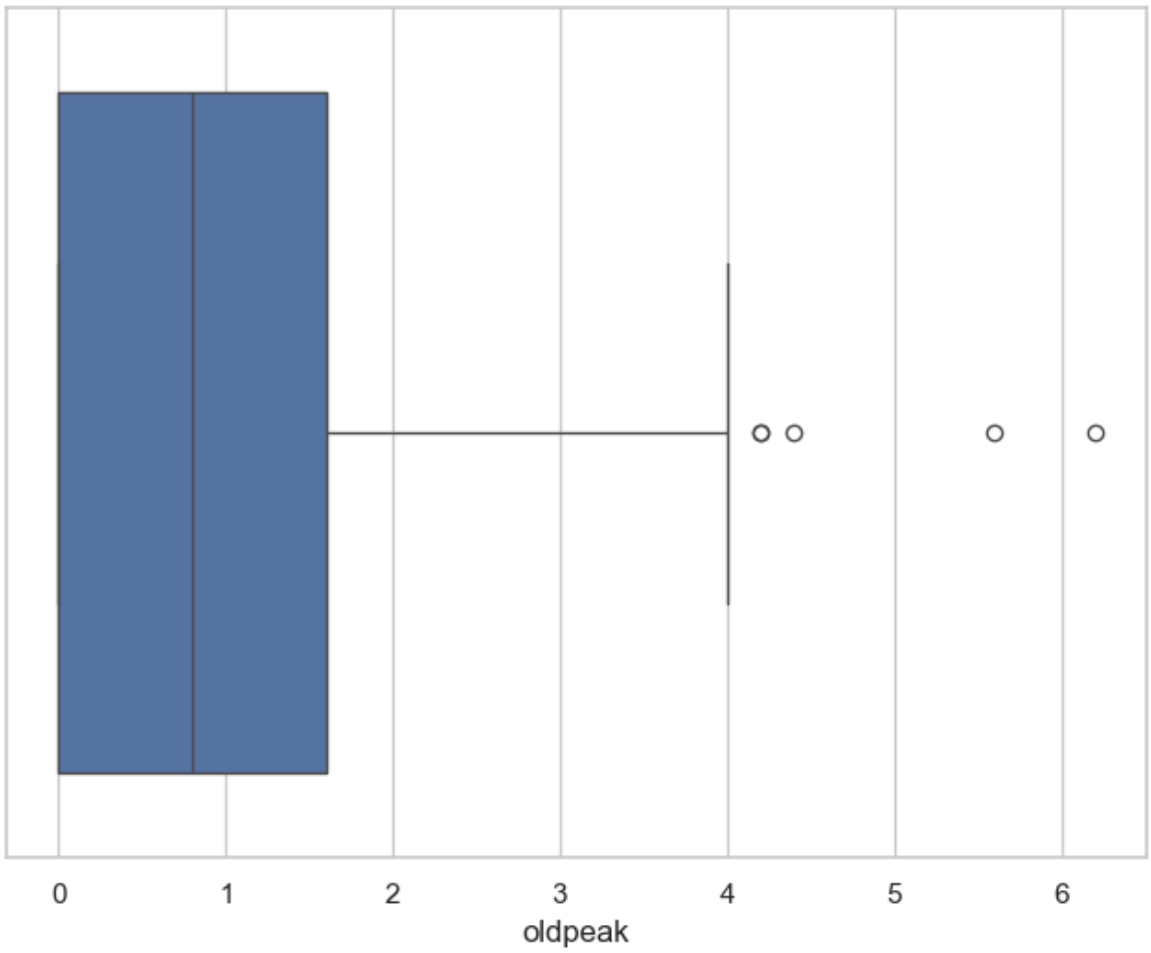
```
#  Oldpeak variable

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

Out[327…

```
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 [329…

```
#  Box-plot of `oldpeak` variable

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



In [ ]: