Extensive Analysis + Visualization with Python

Analysis Based on Heart disease or Cardiovascular disease (CVD)

In this section what we will learn

- import data into python
- Dataframe via panda
- exploring datasets: head() tail() info() describe()
- Univariate Analysis
- Bivariate Analysis
- Multivaraite Analysis
- Dealing with missing values
- Outlier Detection

In [1]:
```
# Import dataset
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

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

In [2]:
```
df  = pd.read_csv(r"C:\Users\Hp\Desktop\NAYAN\DATA SCIENCE\CSV_FILES\heart.csv
df
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	[illegible]
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	[illegible]
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	[illegible]
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	[illegible]
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	[illegible]

303 rows × 14 columns

In [3]:
```
# first always check the shape of the dataset
print('The shape of the daatset : ', df.shape)
```

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

Now, we can see that the dataset contains 303 instances and 14 variables.

In [4]:
```
# Now, we can preview of our dataset
df.head()
```

Out[4]:

	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 [5]:
```
# to get the summary of our data
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 [6]:
```
# Check the data types of columns
df.dtypes
```

Out[6]:
```
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 [7]:
```
# statistical properties of dataset
df.describe()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202

In [8]:
```
# If we want to view the statistical properties of character variables, we sho
# df.describe(include=['object'])
# If we want to view the statistical properties of all the variables, we shoul
# df.describe(include='all')
```

In [9]:
```
# view all your columns name
df.columns
```

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

In [10]:
```
df
```

Out[10]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

Univariate analysis

Analysis of `target` feature variable

- Our feature variable of interest is `target` .
- It refers to the presence of heart disease in the patient.
- It is integer valued as it contains two integers 0 and 1 - (0 stands for absence of heart disease and 1 for presence of heart disease).
- So, in this section, I will analyze the `target` variable.

Target is a dependent Variable (because this is our feature variable and it refers to the presence of heart disease in the patient)

TARGET: It is integer valued as it contains two integers 0 and 1 - (0 stands for absence of heart disease and 1 for presence of heart disease).

Check the number of unique values in `target` variable

In [11]:
```
# check the number of unique value
df['target'].nunique()  # In pandas, the nunique() function is used to count t
# as we can see there are 2 unique values in the target varaible.
```

Out[11]: 2

In [12]:
```
# view which are unique values
df['target'].unique()
# So, the unique values are 1 and 0. (1 stands for presence of heart disease a
```

Out[12]:
```
array([1, 0], dtype=int64)
```

Frequency distribution of target variable

In [13]:
```
df['target'].value_counts()
```

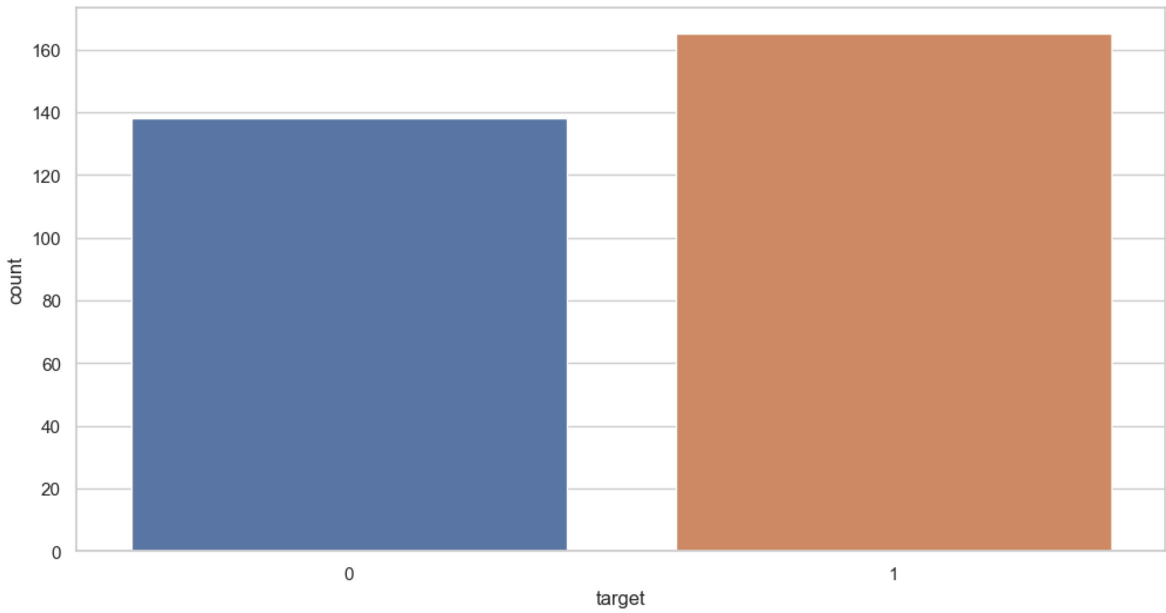
Out[13]:
```
1    165
0    138
Name: target, dtype: int64
```

Comment

- 1 stands for presence of heart disease. So, there are 165 patients suffering from heart disease.
- Similarly, 0 stands for absence of heart disease. So, there are 138 patients who do not have any heart disease.
- We can visualize this information below.

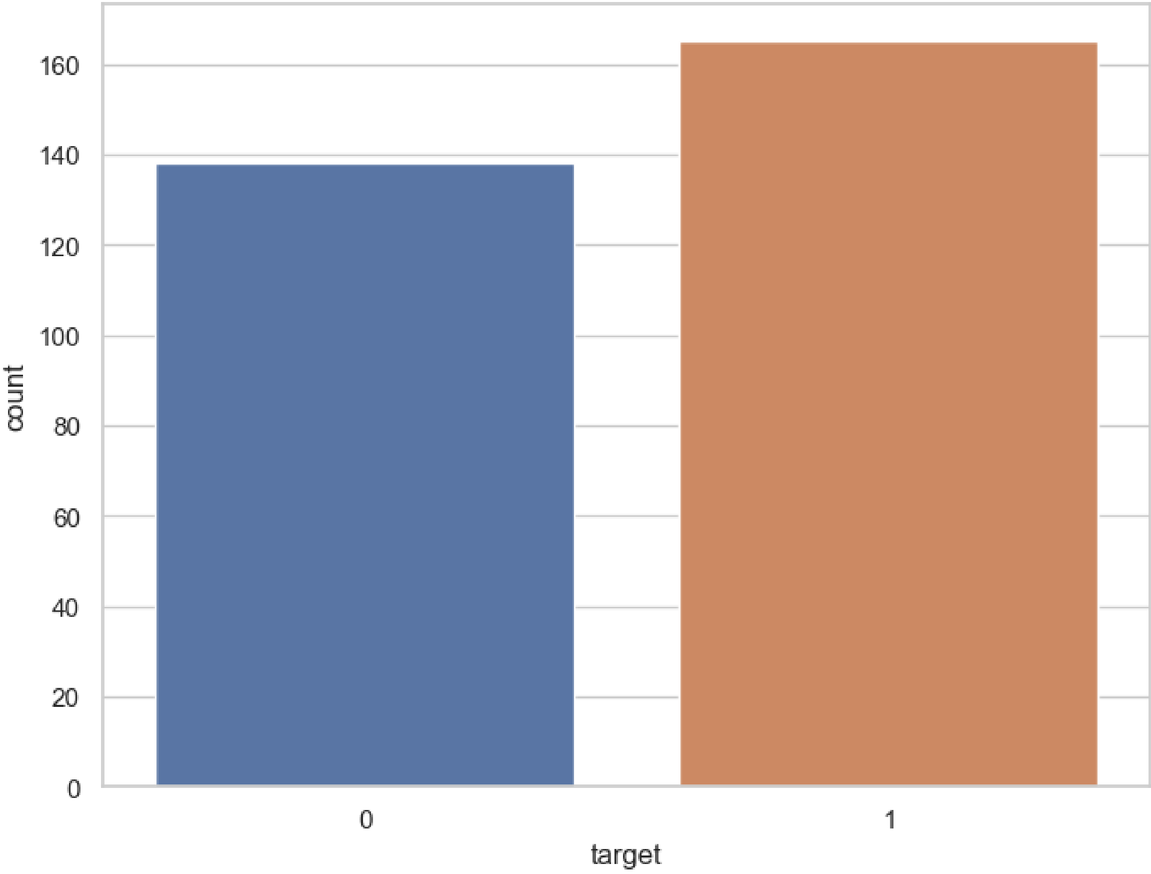
Visualize frequency distribution of target variable

In [14]:
```
y = plt.subplots(figsize = (12, 6))
f = sns.countplot(x="target", data=df)
# f = sns.countplot(x=df["target"])
plt.show()
```

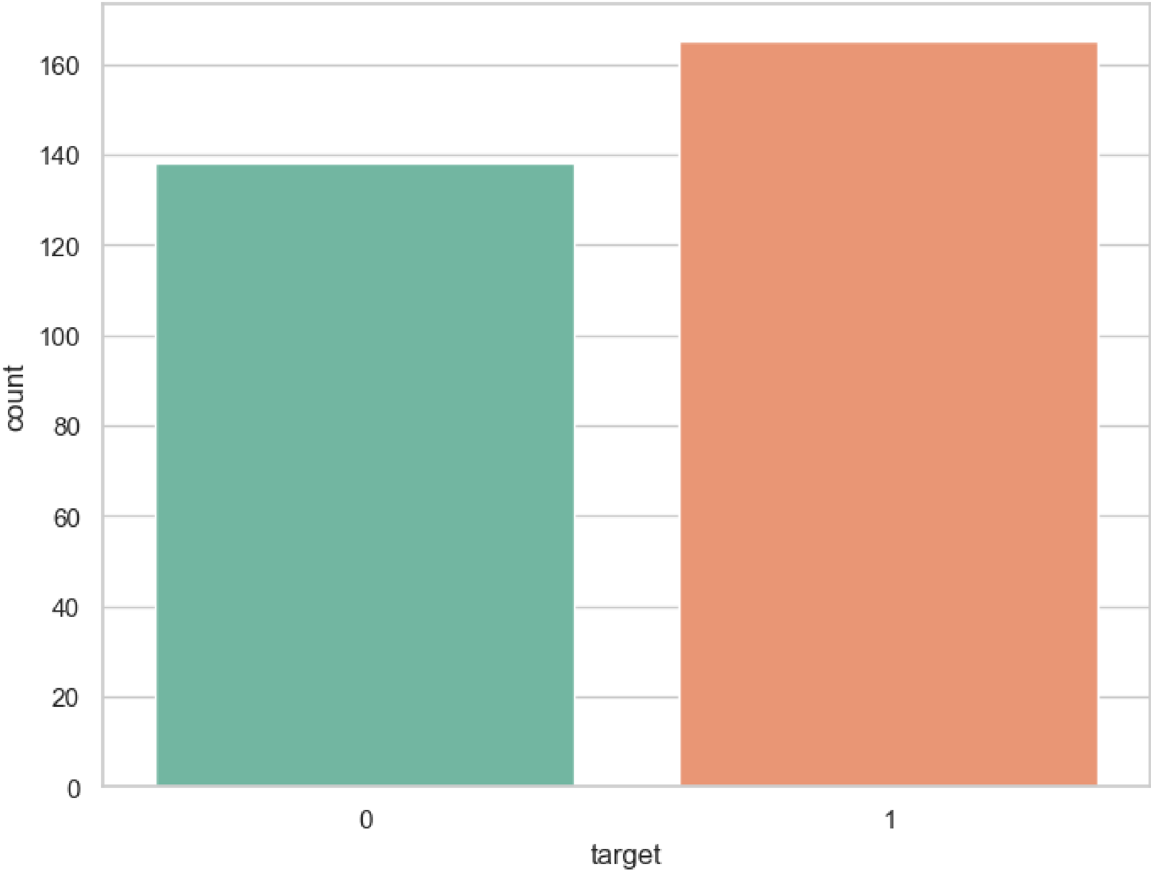


In [15]:

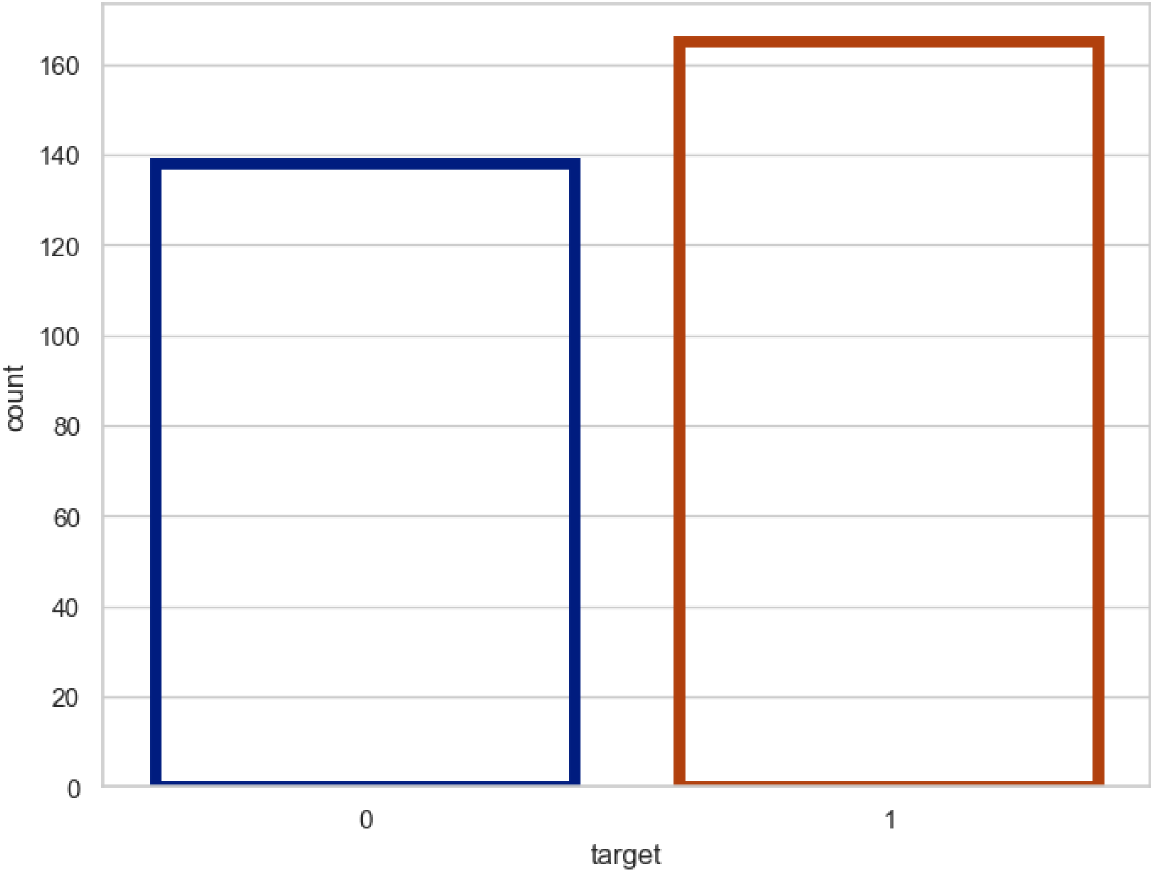
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df)
# ax = sns.countplot(x=df["target"])  # both syntax are same
plt.show()
```



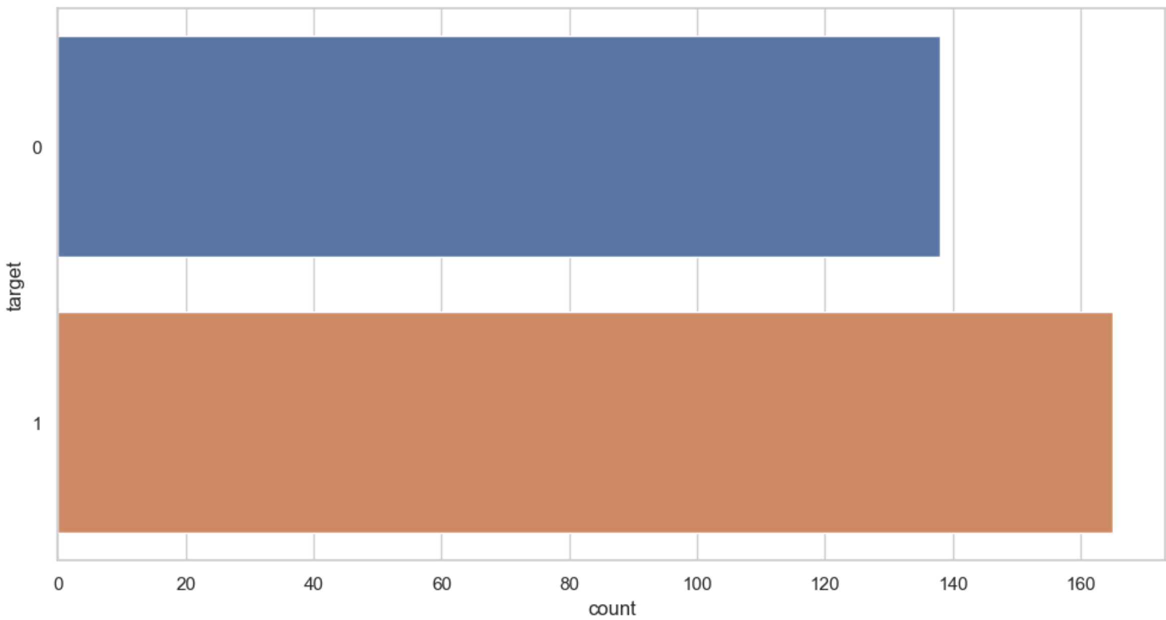
In [16]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, palette="Set2")
plt.show()
```



In [17]:
```
f, ax = plt.subplots(figsize =(8, 6))
ax = sns.countplot(x='target', data=df, facecolor=(0, 0, 0, 0), linewidth=5, e
```



In [18]:
```
y = plt.subplots(figsize = (12, 6))
f = sns.countplot(y="target", data=df)
plt.show()
```



Interpretation

- The above plot confirms the findings that -
  - There are 165 patients suffering from heart disease, and
  - There are 138 patients who do not have any heart disease.

Frequency distribution of target variable wrt sex¶

In [19]:
```
df
```

Out[19]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

In [20]:
```
df.groupby('sex')['target'].value_counts()
```

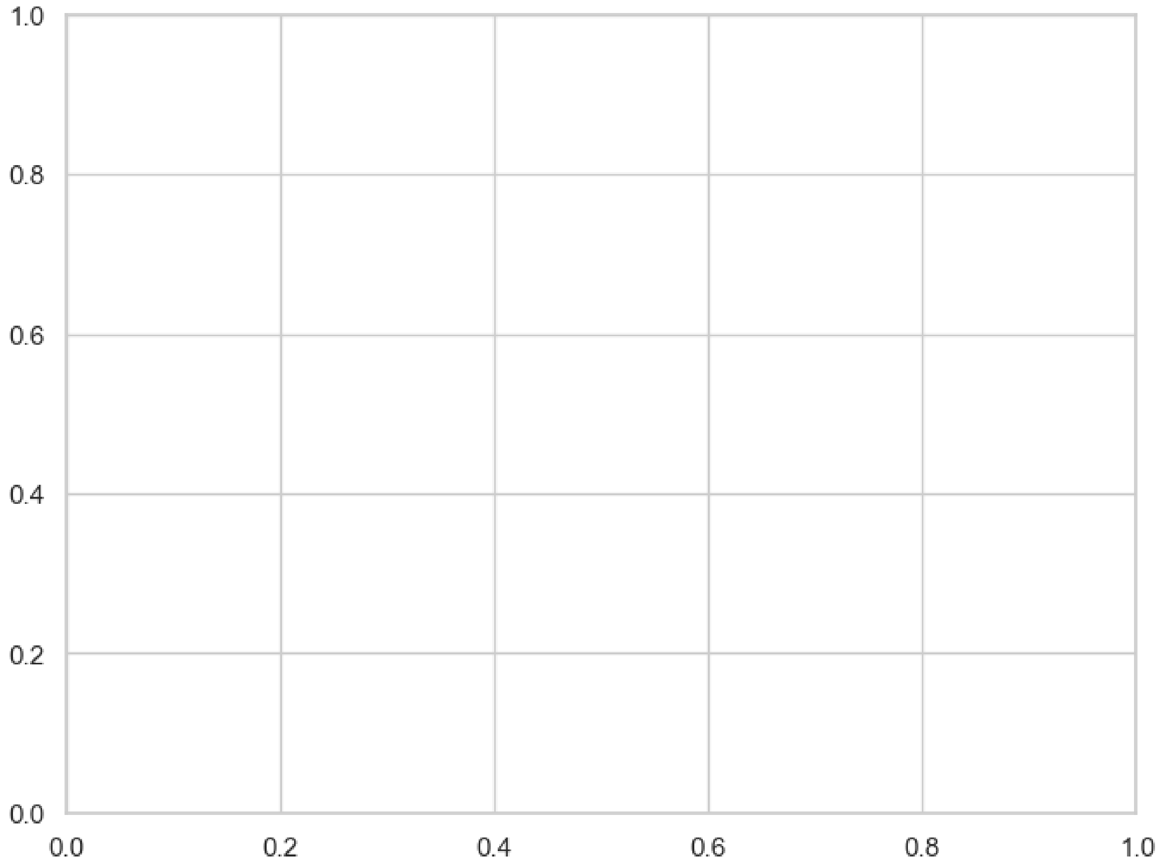
Out[20]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: target, dtype: int64
```

Comment

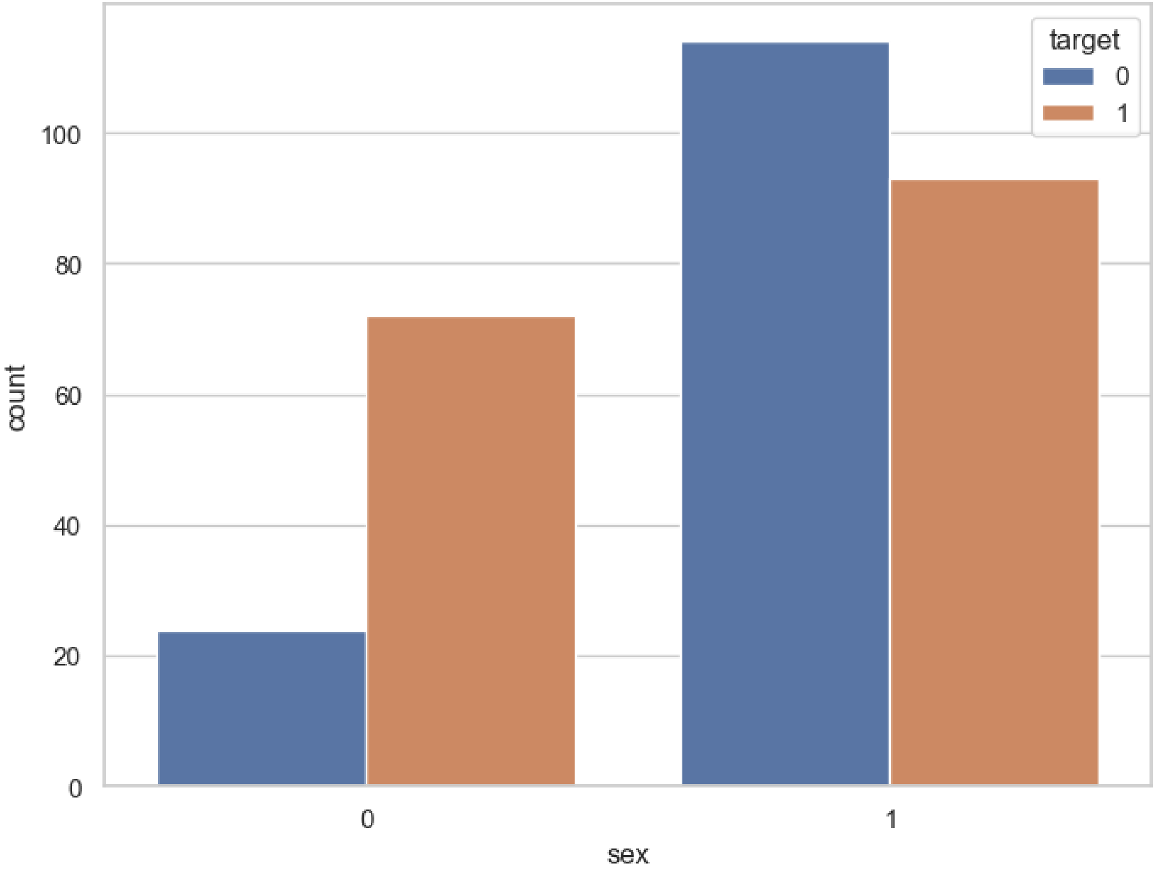
- `sex` variable contains two integer values 1 and 0 : (1 = male; 0 = female).
- `target` variable also contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- So, out of 96 females - 72 have heart disease and 24 do not have heart disease.
- Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.
- We can visualize this information below.

We can visualize the value counts of the `sex` variable wrt `target` as follows -

In [21]:
```
f, ax = plt.subplots(figsize=(8, 6))
```



In [22]:
```
# now, we can visualzie the value counts of the sex varaible wrt target (1 and
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x='sex', hue='target', data=df)
# plt.show() or ax both are same thing
```

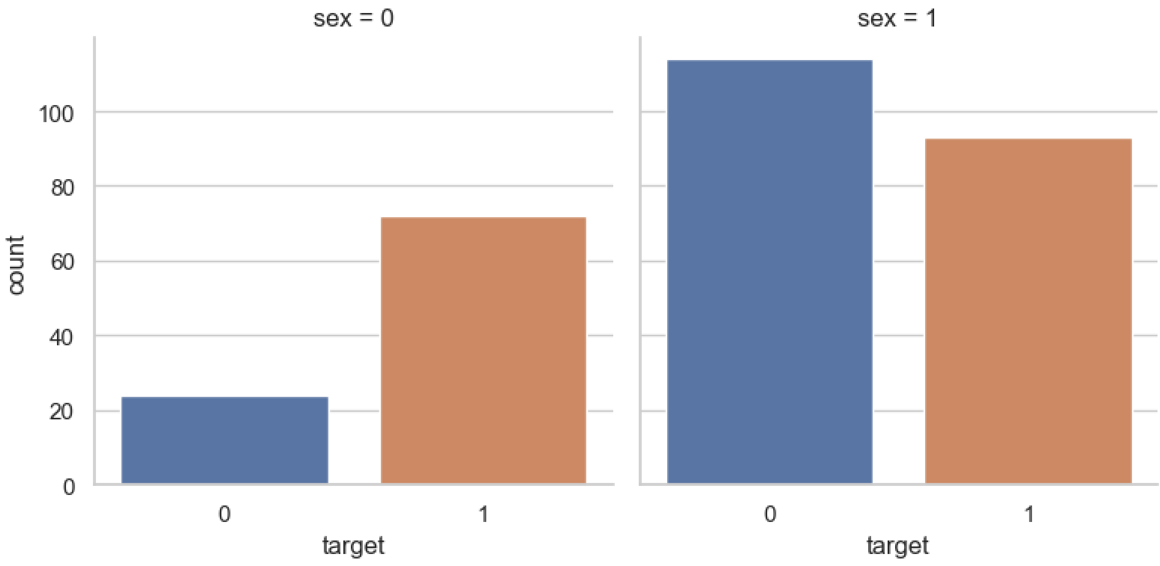


Interpretation

- The above plot confirms our findings that -
  - Out of 96 females - 72 have heart disease and 24 do not have heart disease.
  - Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

Alternatively, we can visualize the same information as follows :

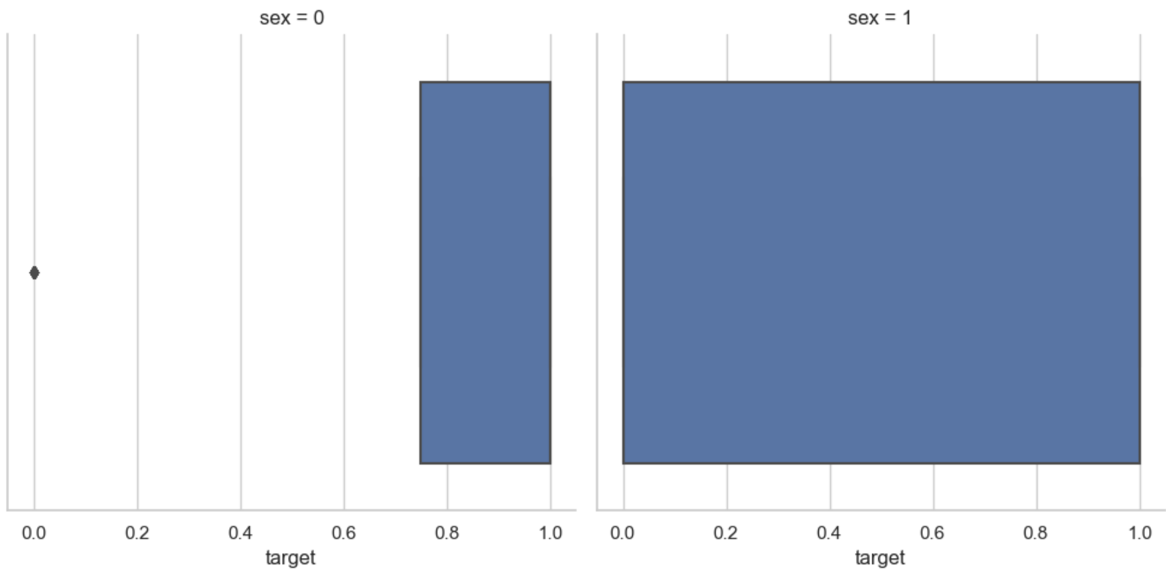
In [23]:
```
ax = sns.catplot(x = 'target', col = 'sex', data=df, kind = 'count', height=4,
```



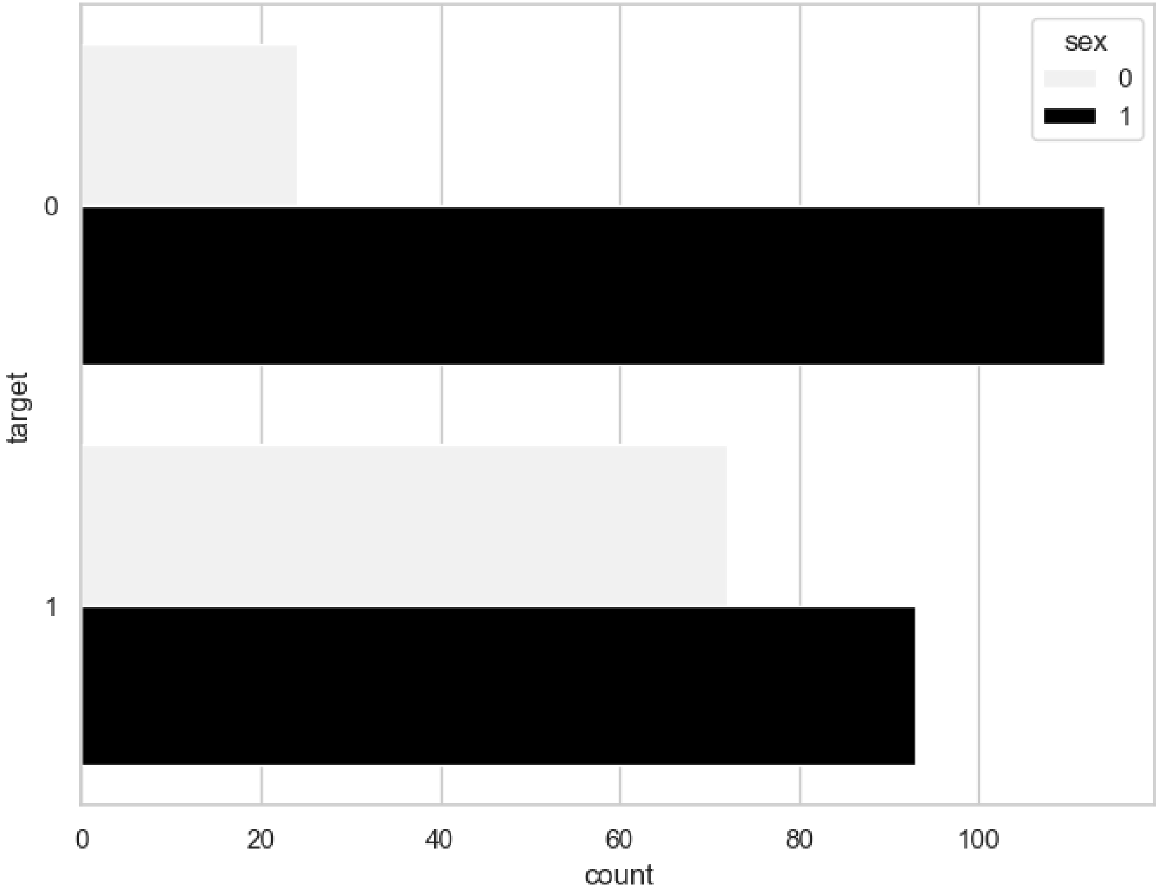
Comment

- The above plot segregate the values of target variable and plot on two different columns labelled as (sex = 0, sex = 1).
- I think it is more convinient way of interpret the plots.

In [24]:
```
ax = sns.catplot(x="target", col="sex", data=df, kind="box", height=5, aspect=
# axes-level plotting function. Options are: "strip", "swarm", "box", "violin'
```



In [25]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(y='target', hue='sex', data=df, color='black')
plt.show()
```



Frequency distribution of target variable wrt fbs(fasting bool sugar)

In [26]:
```
df
```

Out[26]:

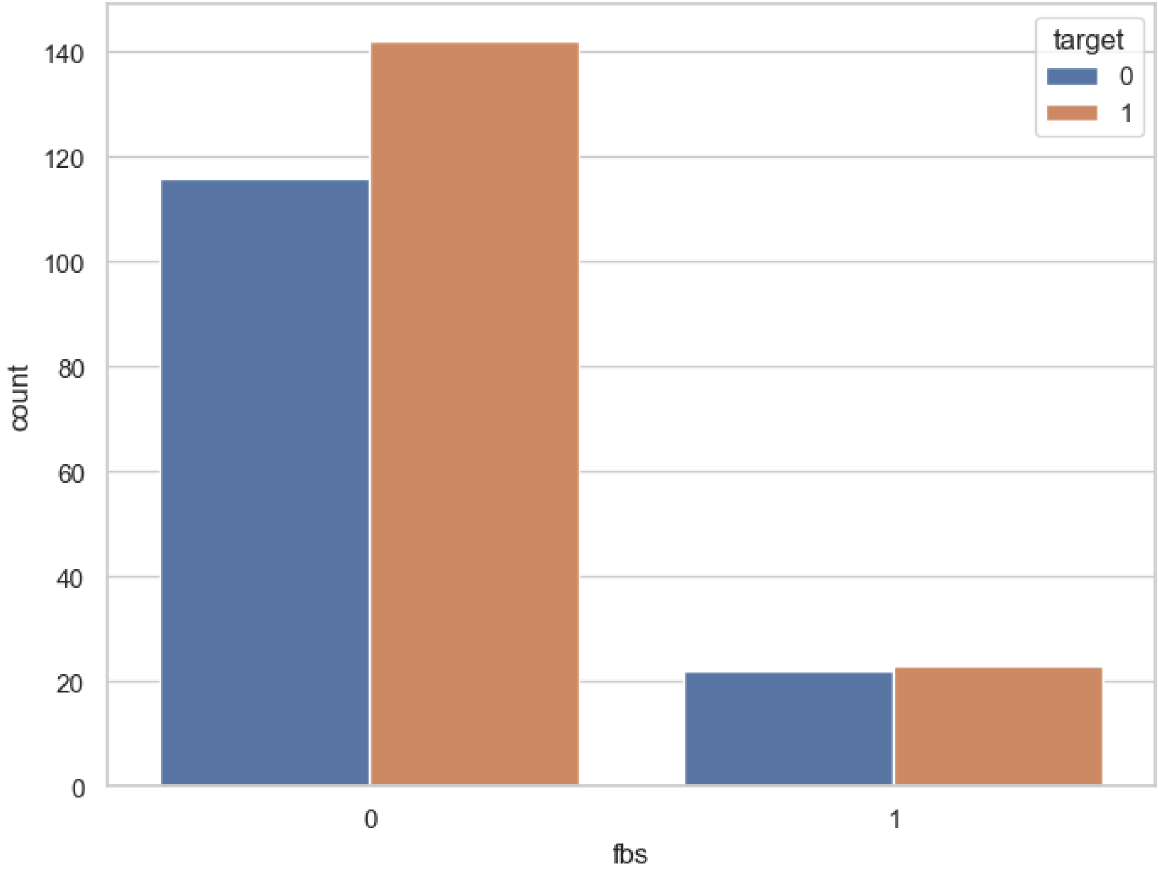
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...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

303 rows × 14 columns

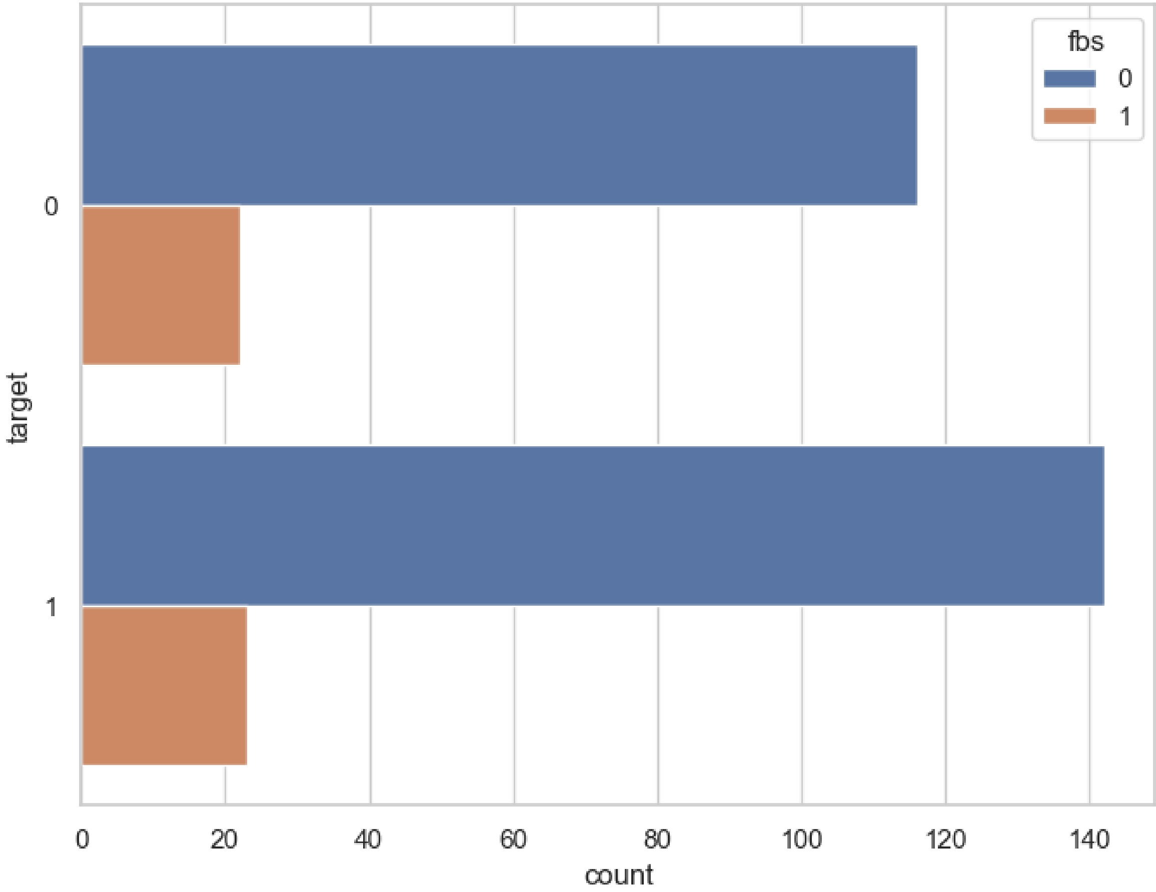
In [27]:
```
df.groupby('fbs')['target'].value_counts()
```

Out[27]:
```
fbs  target
0    1         142
     0         116
1    1          23
     0          22
Name: target, dtype: int64
```

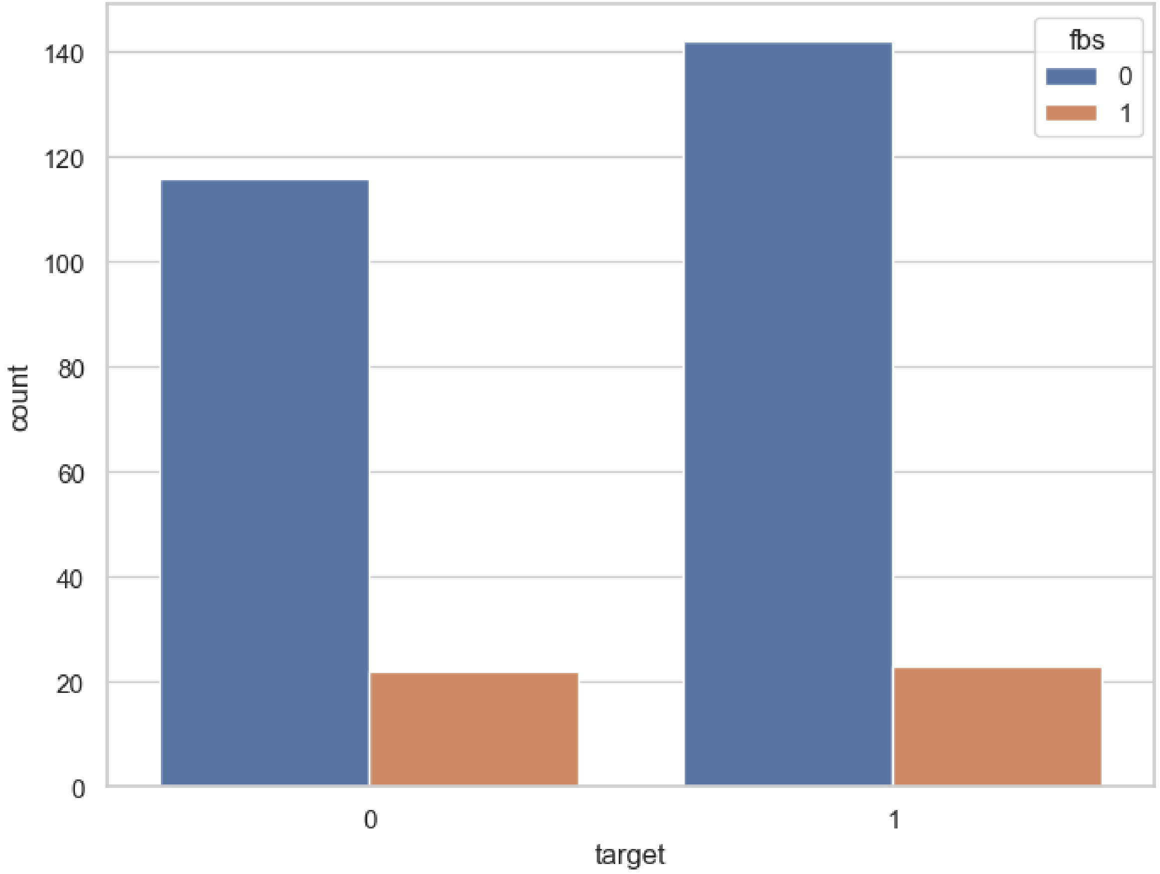
In [28]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x='fbs', hue='target', data=df)
plt.show()
```



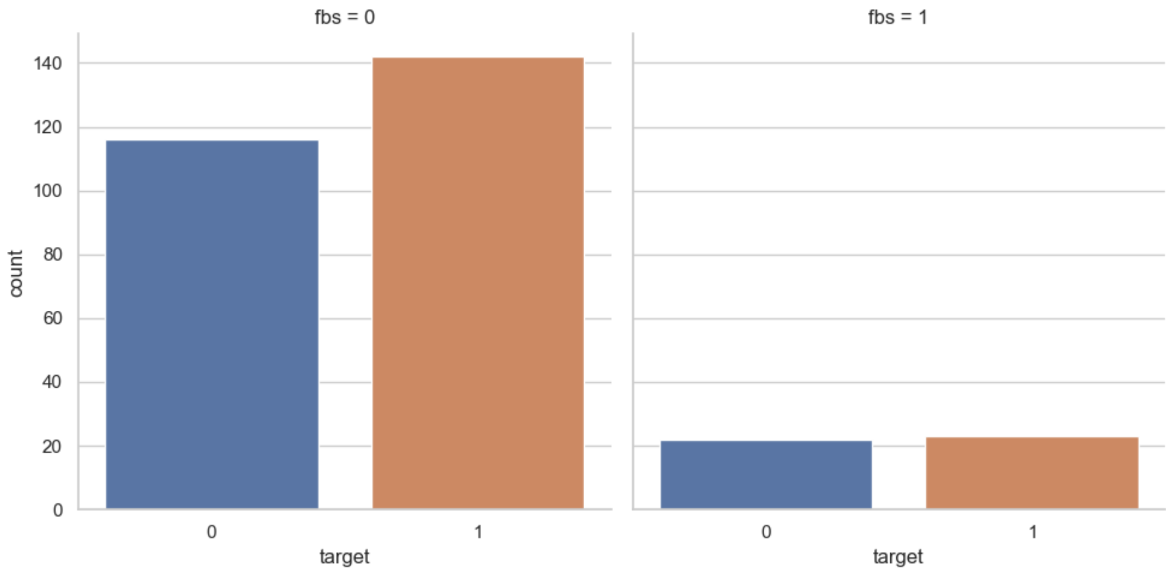
In [29]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(y= 'target',hue = 'fbs', data=df)
plt.show()
```



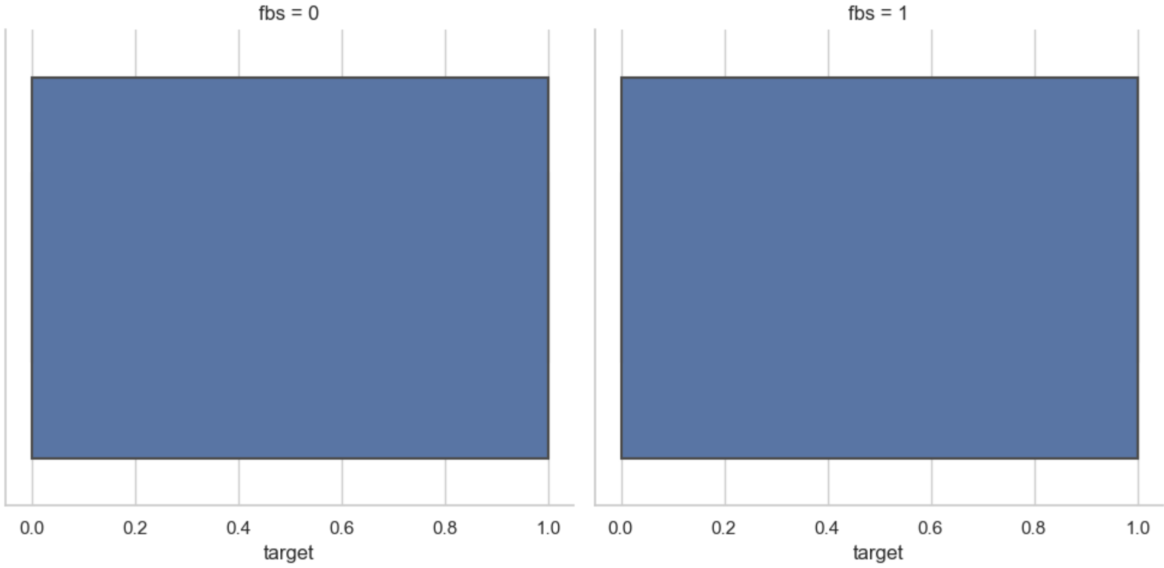
In [30]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x = 'target',hue = 'fbs', data=df)
plt.show()
```



In [31]:
```
ax = sns.catplot(x='target', col = 'fbs', data=df, kind='count', height = 5, a
```



In [32]:
```
ax = sns.catplot(x='target', col = 'fbs', data=df, kind='box', height = 5, asp
```



Frequency distribution of target variable wrt exang (exercise induced angina)

In [33]:
```
df
```

Out[33]:

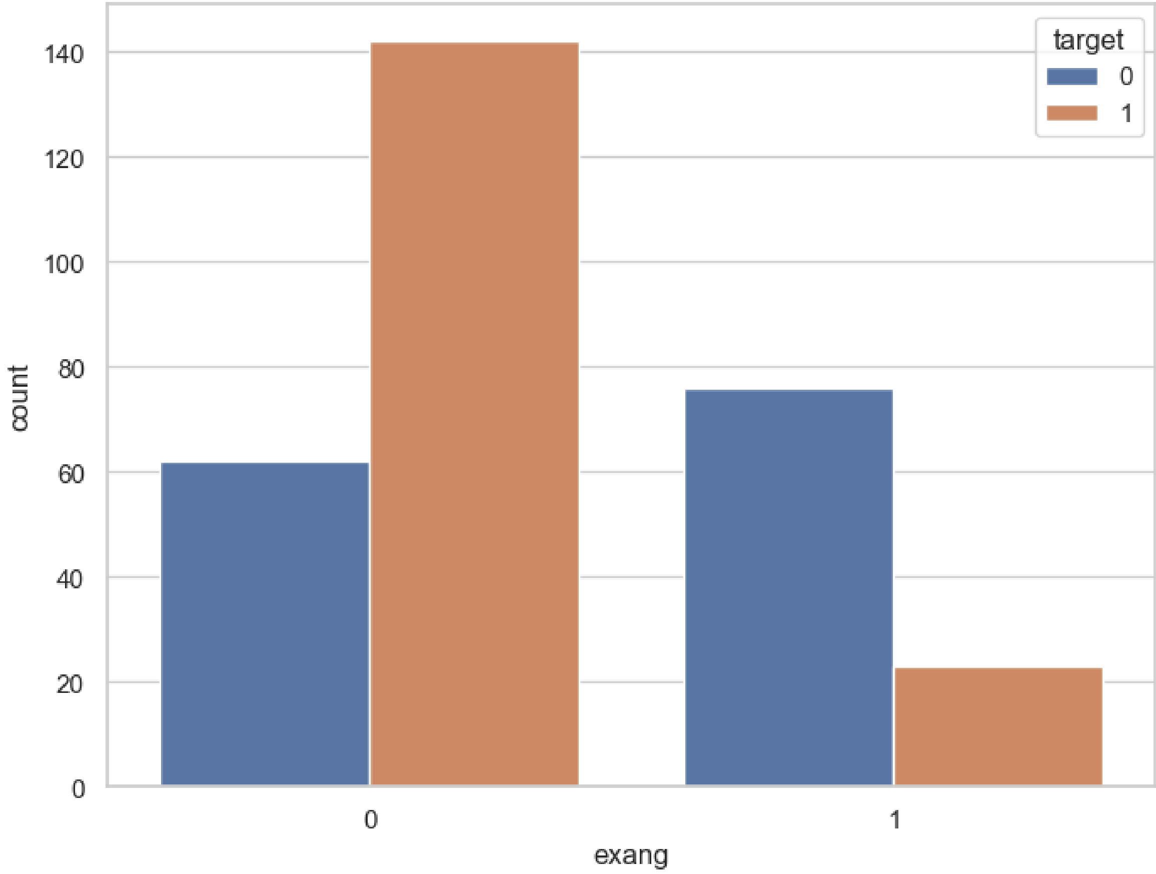
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	
...	...	...	...	...	...	...	...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

303 rows × 14 columns

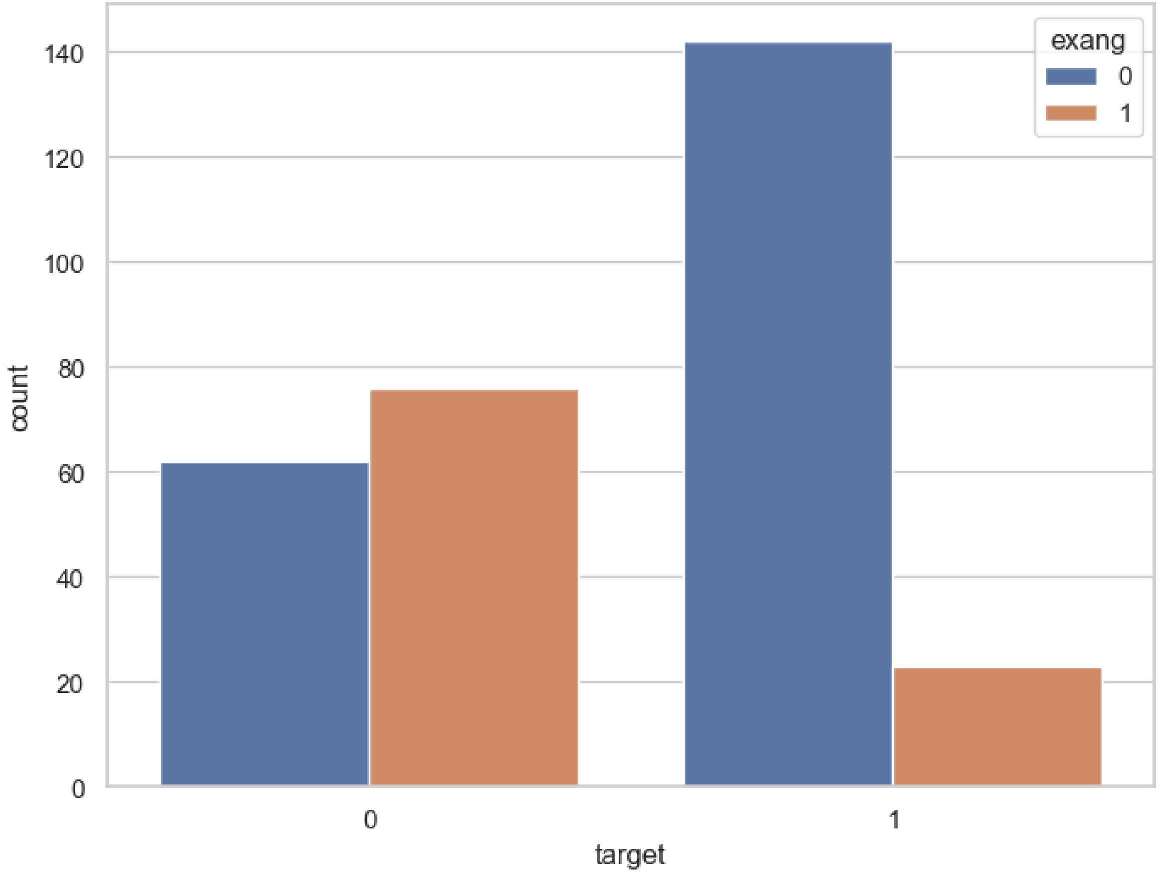
In [34]:
```
df.groupby('exang')['target'].value_counts()
```

Out[34]:
```
exang  target
0      1         142
       0          62
1      0          76
       1          23
Name: target, dtype: int64
```

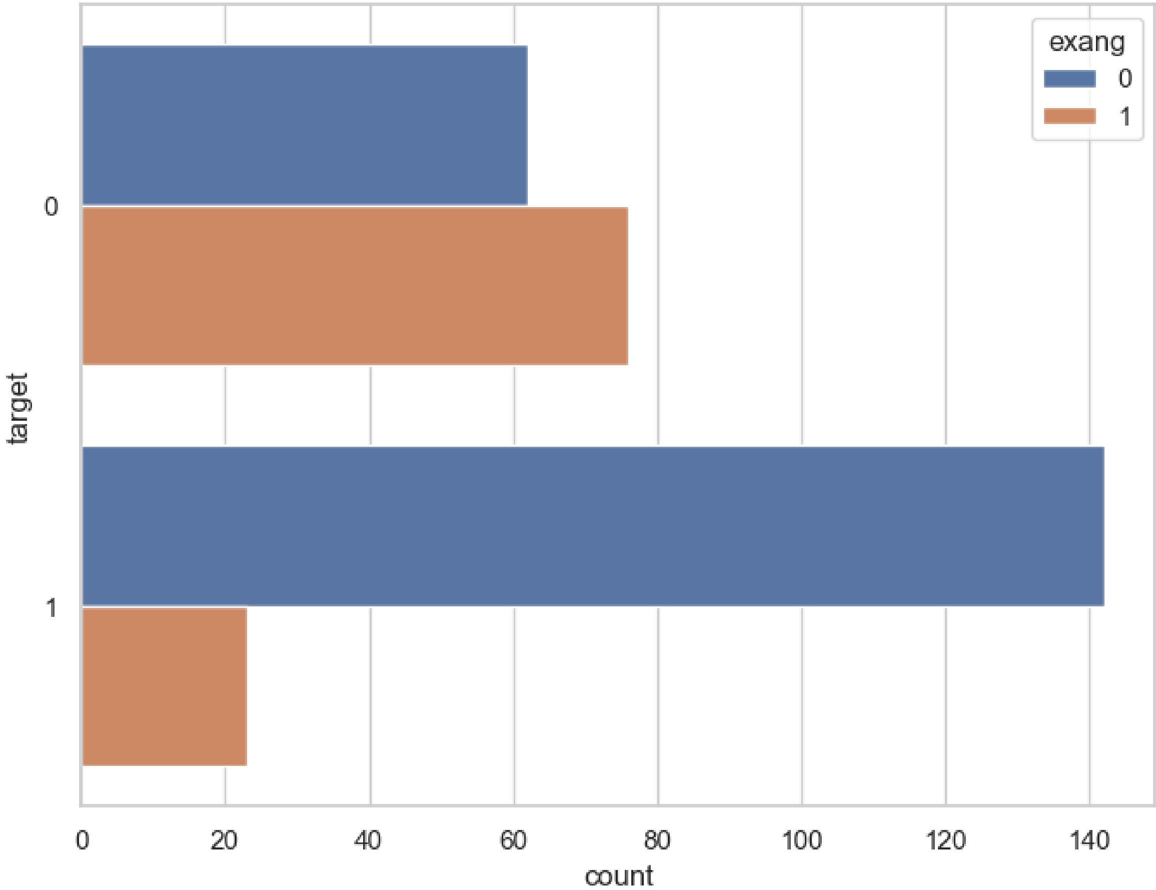
In [35]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x='exang', hue='target', data=df)
plt.show()
```



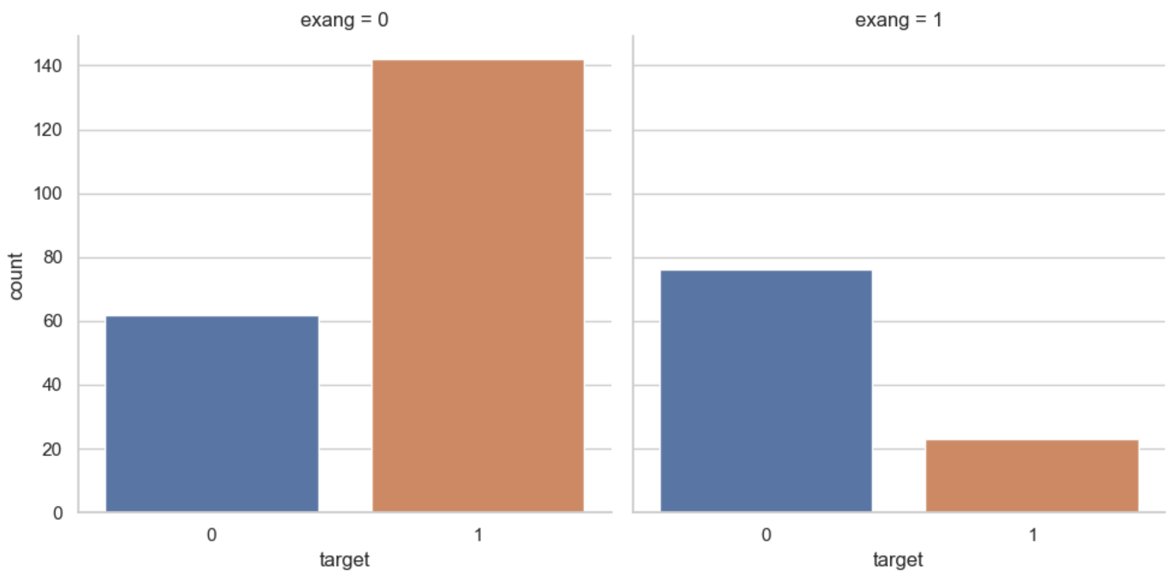
In [36]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x = 'target',hue = 'exang', data=df)
plt.show()
```



In [37]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(y= 'target',hue = 'exang', data=df)
plt.show()
```



In [38]:
```
ax = sns.catplot(x='target', col = 'exang', data=df, kind='count', height = 5,
```



Findings of Univariate Analysis

Findings of univariate analysis are as follows:-

- `Our feature variable of interest is `target`.`
- It refers to the presence of heart disease in the patient.

- It is integer valued as it contains two integers 0 and 1 - (0 stands for absence of heart disease and 1 for presence of heart disease).
- `1` stands for presence of heart disease. So, there are 165 patients suffering from heart disease.
- Similarly, `0` stands for absence of heart disease. So, there are 138 patients who do not have any heart disease.
- There are 165 patients suffering from heart disease, and
- There are 138 patients who do not have any heart disease.
- Out of 96 females - 72 have heart disease and 24 do not have heart disease.
- Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

Bivariate analysis

In [39]:
```
df
```

Out[39]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	(
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	(
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	(
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	(
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	(

303 rows × 14 columns

In [40]:
```
correlation = df.corr()
```

In [41]:
```
correlation['target'].sort_values(ascending=False)
```

Out[41]:
```
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
```

Interpretation of correlation coefficient

- The correlation coefficient ranges from -1 to +1.
- When it is close to +1, this signifies that there is a strong positive correlation. So, we can see that there is no variable which has strong positive correlation with `target` variable.
- When it is clsoe to -1, it means that there is a strong negative correlation. So, we can see that there is no variable which has strong negative correlation with `target` variable.
- When it is close to 0, it means that there is no correlation. So, there is no correlation between `target` and `fbs` .
- We can see that the `cp` and `thalach` variables are mildly positively correlated with `target` variable. So, I will analyze the interaction between these features and `target` variable.

Analysis of `target` and `cp` variable

In [42]:
```
# cp -- chest pain
'''Firsty, I will check number of unique values in cp variable'''
```

Out[42]:
```
'Firsty, I will check number of unique values in cp variable'
```

In [43]:
```
df['cp'].nunique()
```

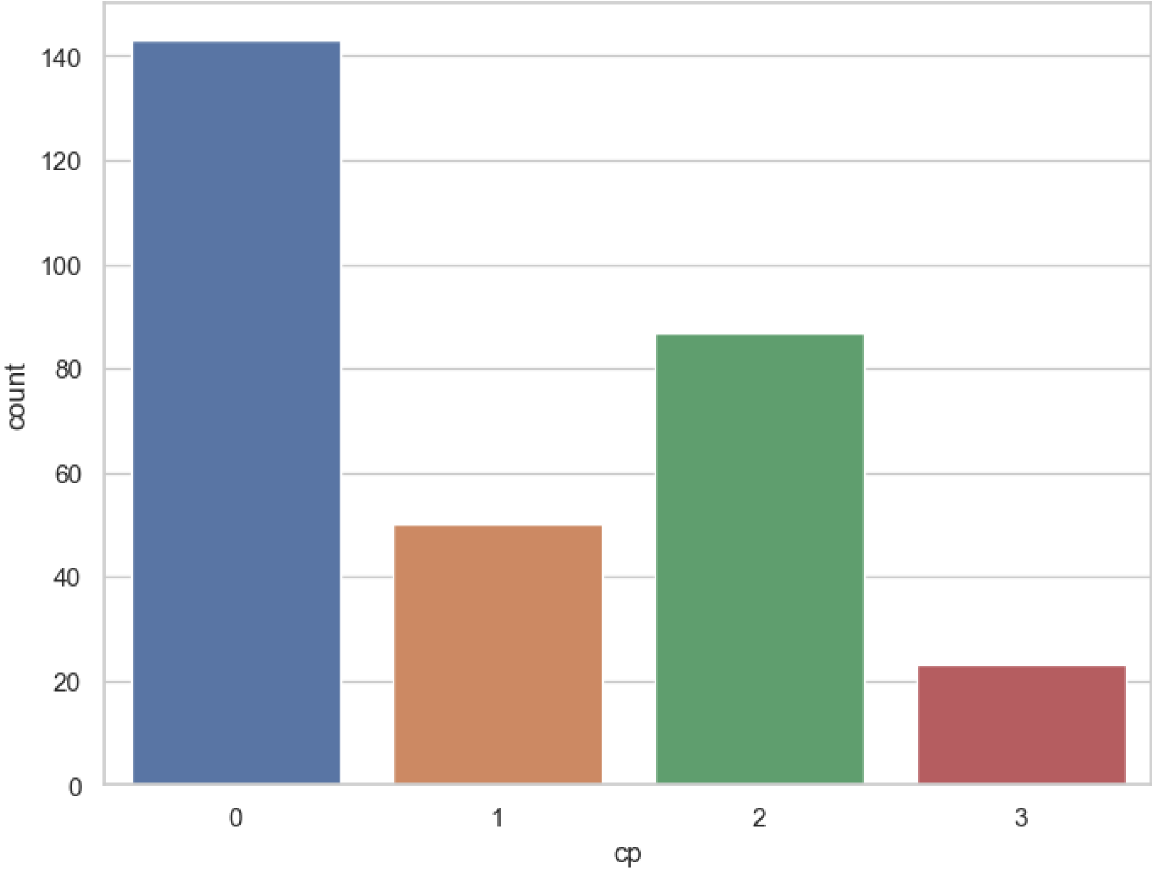
Out[43]:
```
4
```

In [44]:
```
df['cp'].value_counts()
```

Out[44]:
```
0    143
2     87
1     50
3     23
Name: cp, dtype: int64
```

Visualize the frequence distribution of `cp` varaible

In [45]:
```
f, ax = plt.subplots(figsize = (8, 6))
ax = sns.countplot(x = 'cp', data = df)
plt.show()
```



Frequency distribution of target variable wrt cp

In [46]:
```
df.groupby('cp')['target'].value_counts()
```

Out[46]:
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: target, dtype: int64
```

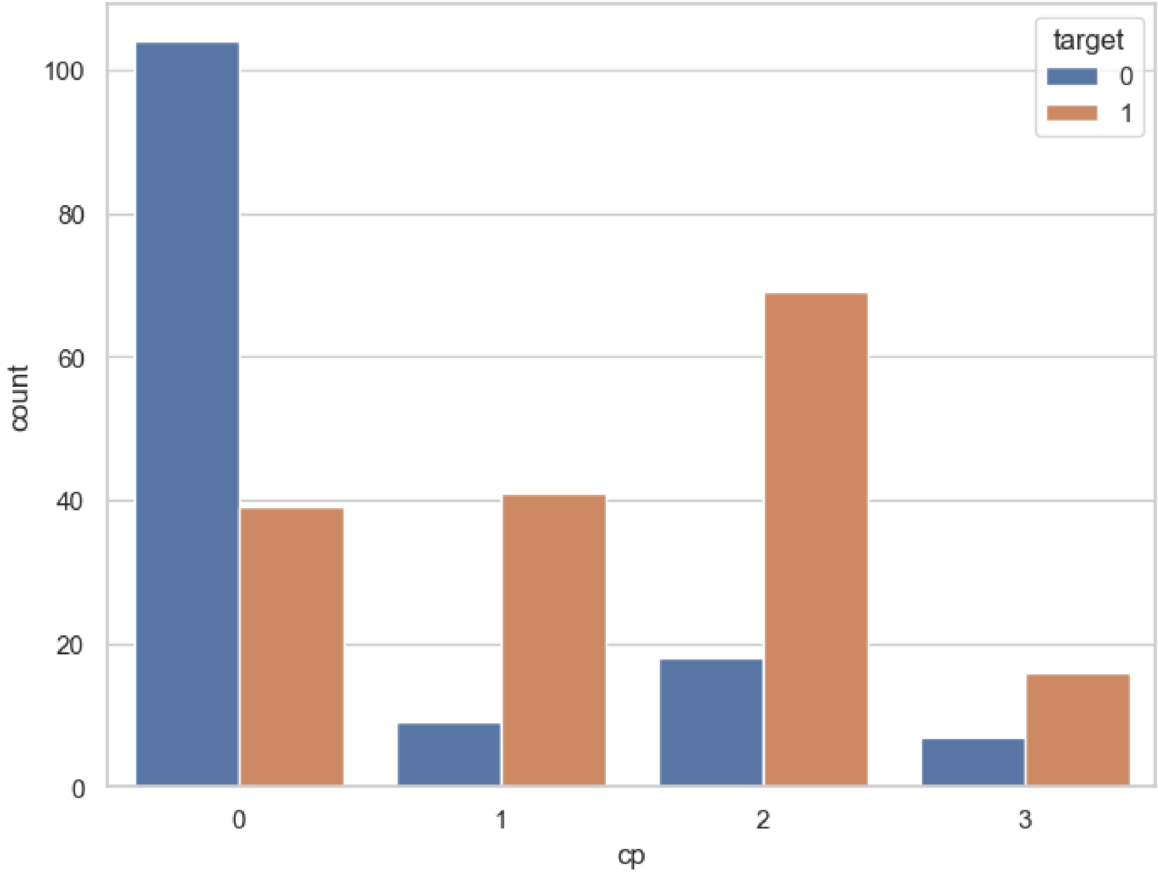
Comment

- `cp` variable contains four integer values 0, 1, 2 and 3.
- `target` variable contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- So, the above analysis gives `target` variable values categorized into presence and absence of heart disease and groupby `cp` variable values.

- We can visualize this information below.

We can visualize the value counts of the `cp` variable wrt `target` as follows -

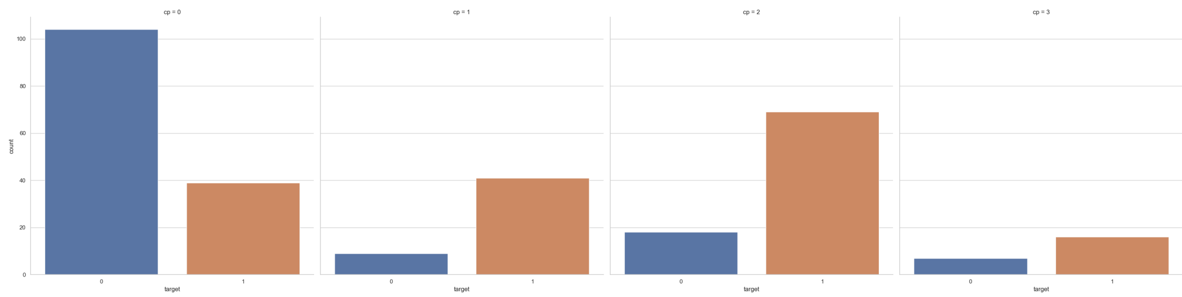
In [47]:
```
f, ax = plt.subplots(figsize = (8, 6))
ax = sns.countplot(x = 'cp', hue = 'target', data=df)
```



Interpretation

- We can see that the values of `target` variable are plotted wrt `cp` .
- `target` variable contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- The above plot confirms our above findings,

In [48]:
```
ax = sns.catplot(x = 'target', col = 'cp', data=df, kind= 'count', height = 8,
```



Analysis of `target` and `thalach` variable

Explore `thalach` variable

- `thalach` stands for maximum heart rate achieved.
- I will check number of unique values in `thalach` variable as follows :

In [49]:
```
df['thalach'].nunique()
```

Out[49]:
```
91
```

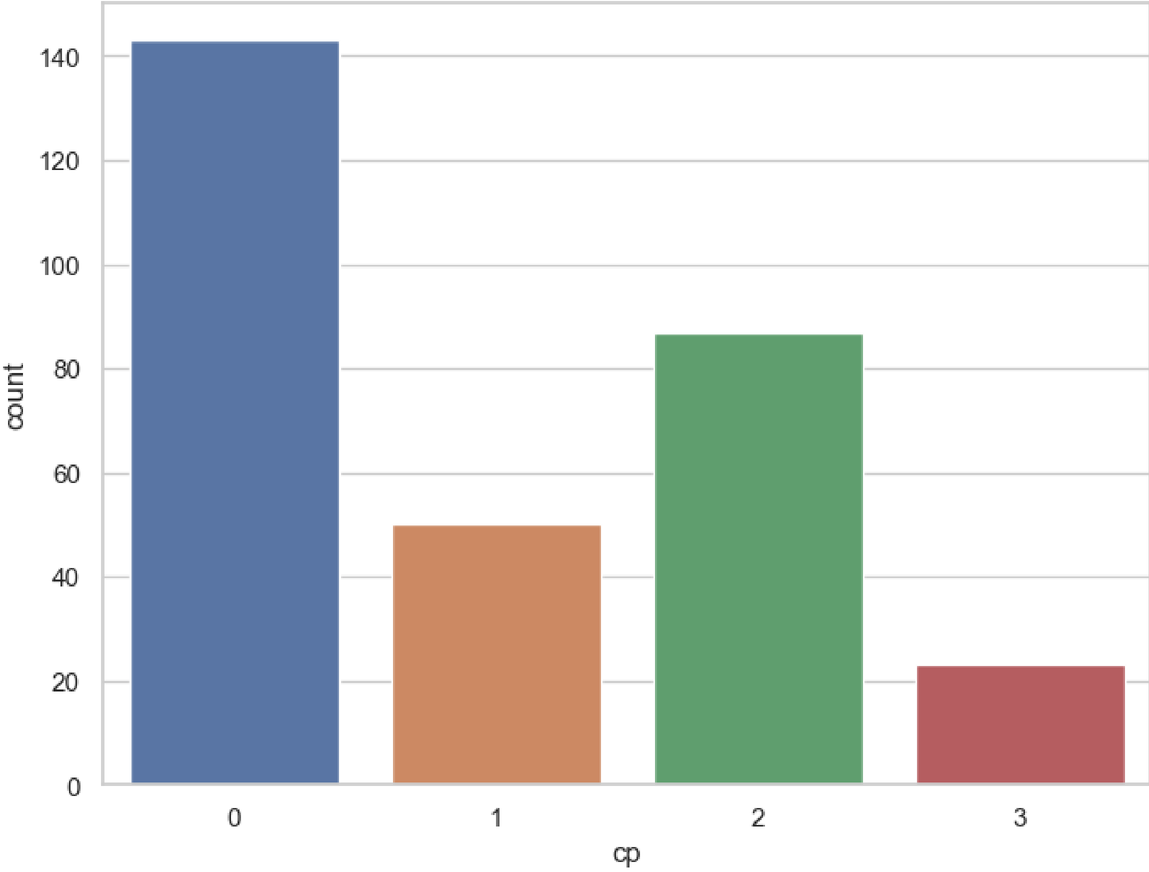
- So, number of unique values in `thalach` variable is 91. Hence, it is numerical variable.
- I will visualize its frequency distribution of values as follows :

In [50]:
```
df['thalach'].value_counts()
```

Out[50]:
```
162    11
160     9
163     9
152     8
173     8
       ..
202     1
184     1
121     1
192     1
90      1
Name: thalach, Length: 91, dtype: int64
```

Visualize the frequence distribution of `thalach` varaible

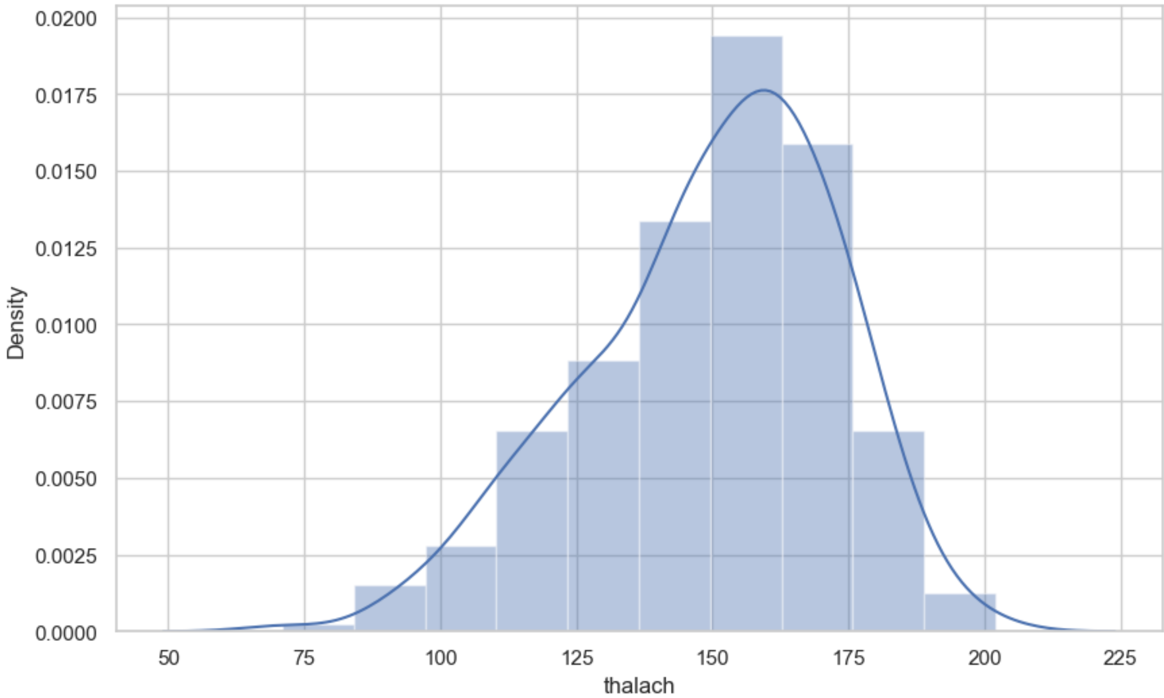
In [51]:
```
f, ax = plt.subplots(figsize = (8, 6))
ax = sns.countplot(x = 'cp', data = df)
plt.show()
```



In [52]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [53]:
```
# f, ax = plt.subplots(figsize=(10, 6))
# ax = sns.distplot(x = 'thalach', data = df, bins= 10)  # error because distp
# plt.show()
```

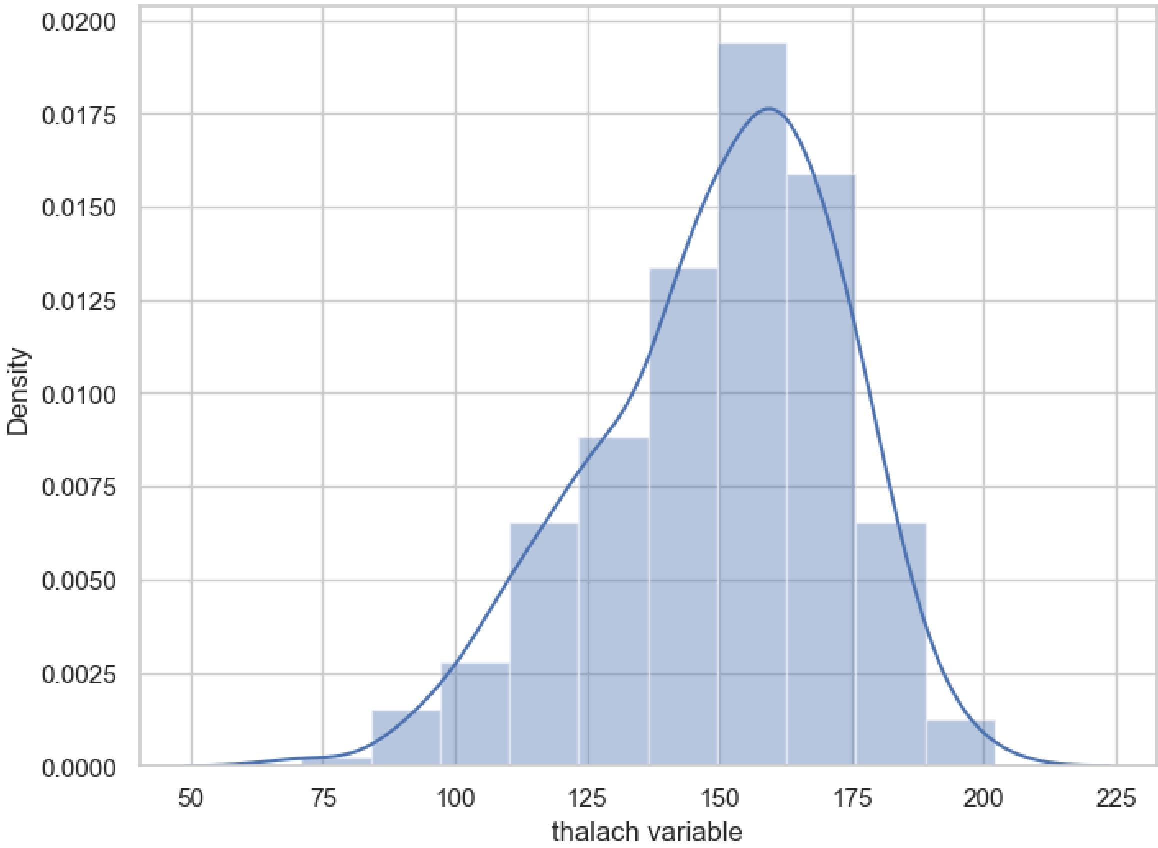
In [54]:
```
f, ax = plt.subplots(figsize=(10, 6))
x = df['thalach']
ax = sns.distplot(x, bins= 10)
plt.show()
```



Comment

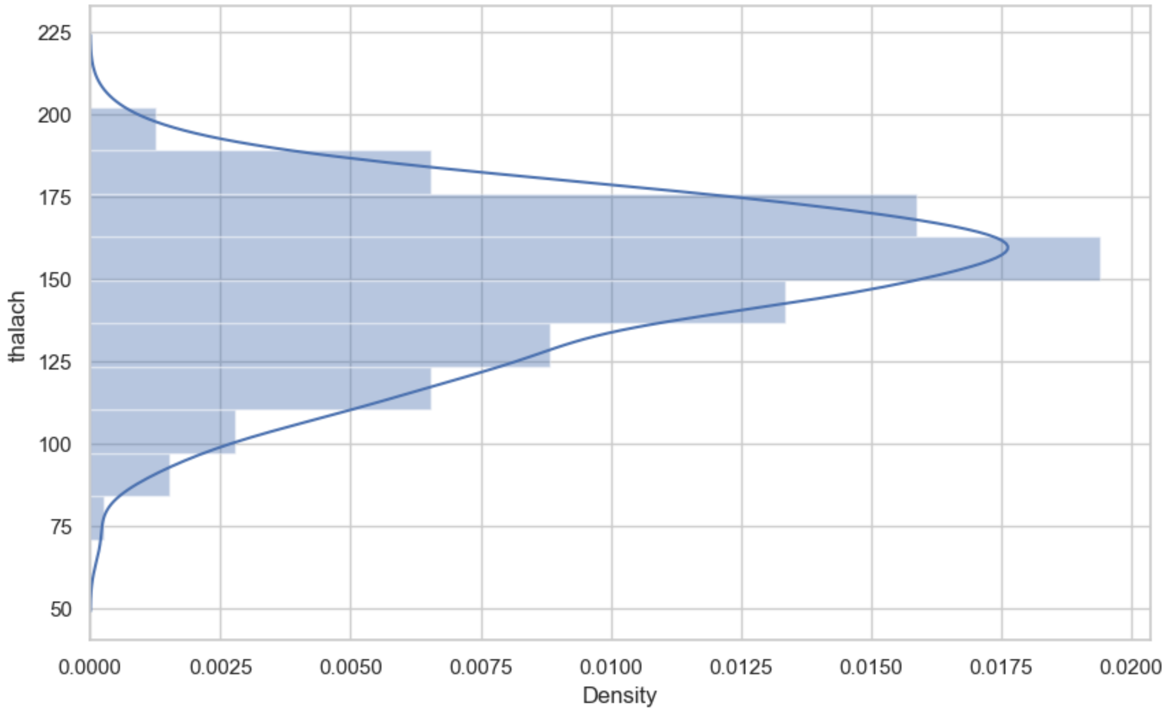
- We can see that the `thalach` variable is slightly negatively skewed.

In [55]:
```
f, ax = plt.subplots(figsize = (8, 6))
x = df['thalach']
x = pd.Series(x, name='thalach variable')
ax = sns.distplot(x, bins= 10)
plt.show()
```



We can plot the distribution on the vertical axis as follows:-

In [56]:
```
f, ax = plt.subplots(figsize=(10, 6))
x = df['thalach']
ax = sns.distplot(x, bins = 10, vertical= True)
plt.show()
```

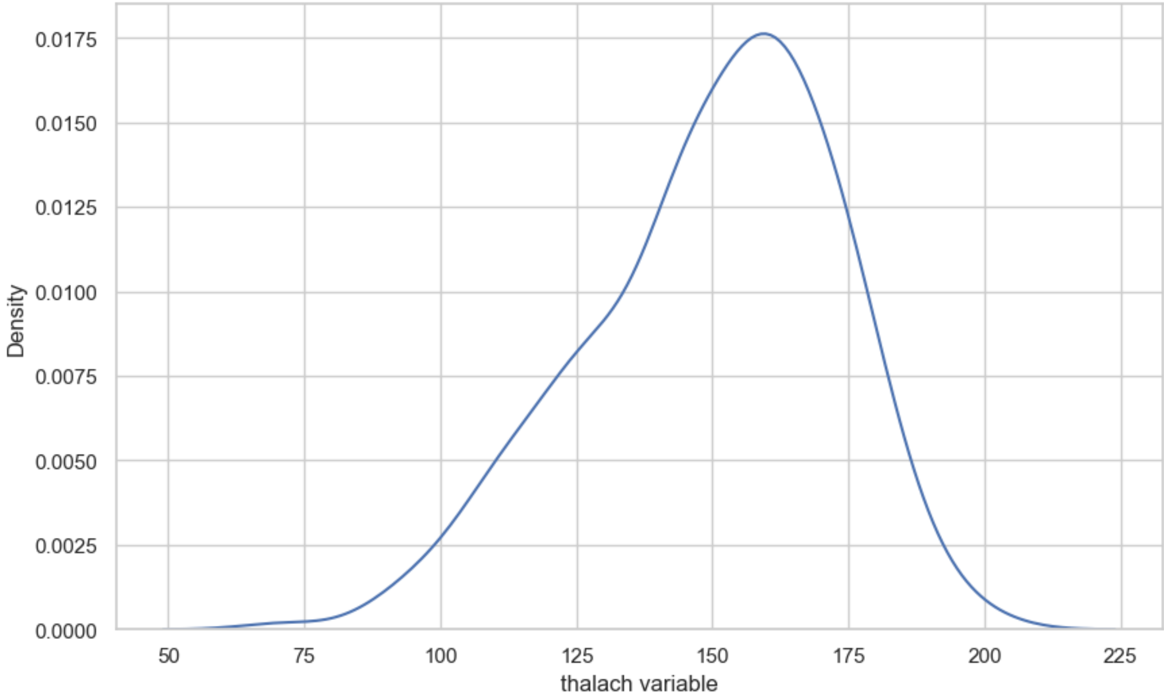


Seaborn Kernel Density Estimation (KDE) Plot

The kernel density estimate (KDE) plot is a useful tool for plotting the shape of a distribution.

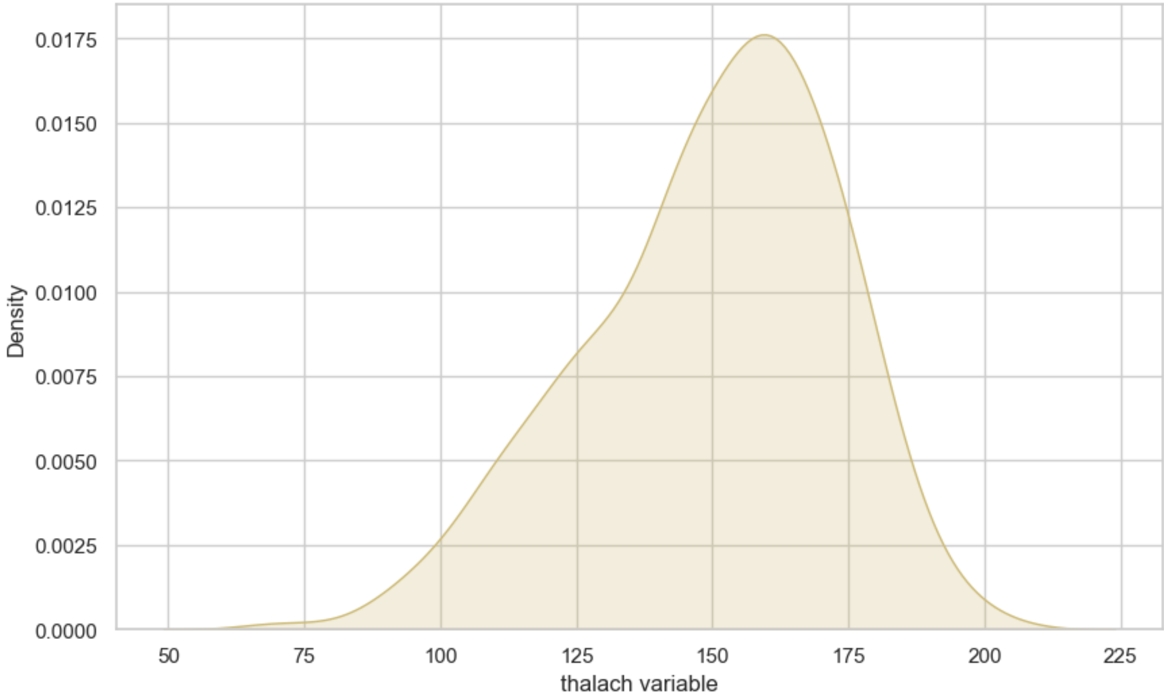
- The KDE plot plots the density of observations on one axis with height along the other axis.
- We can plot a KDE plot as follows :

In [57]:
```
f, ax = plt.subplots(figsize=(10, 6))
x = df['thalach']
x = pd.Series(x, name= 'thalach variable')
ax = sns.kdeplot(x)
plt.show()
```



We can shade under the density curve and use a different color as follows:

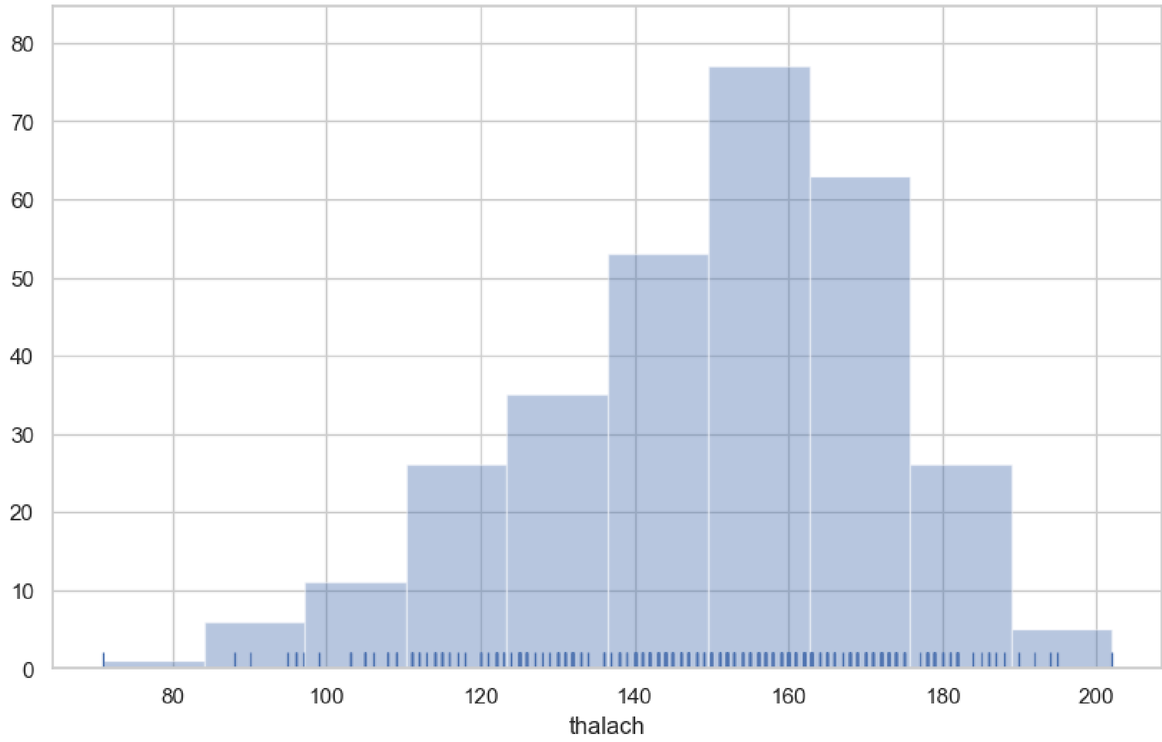
In [58]:
```
f, ax = plt.subplots(figsize=(10, 6))
x = df['thalach']
x = pd.Series(x, name= 'thalach variable')
ax = sns.kdeplot(x, shade=True, color='y')
plt.show()
```



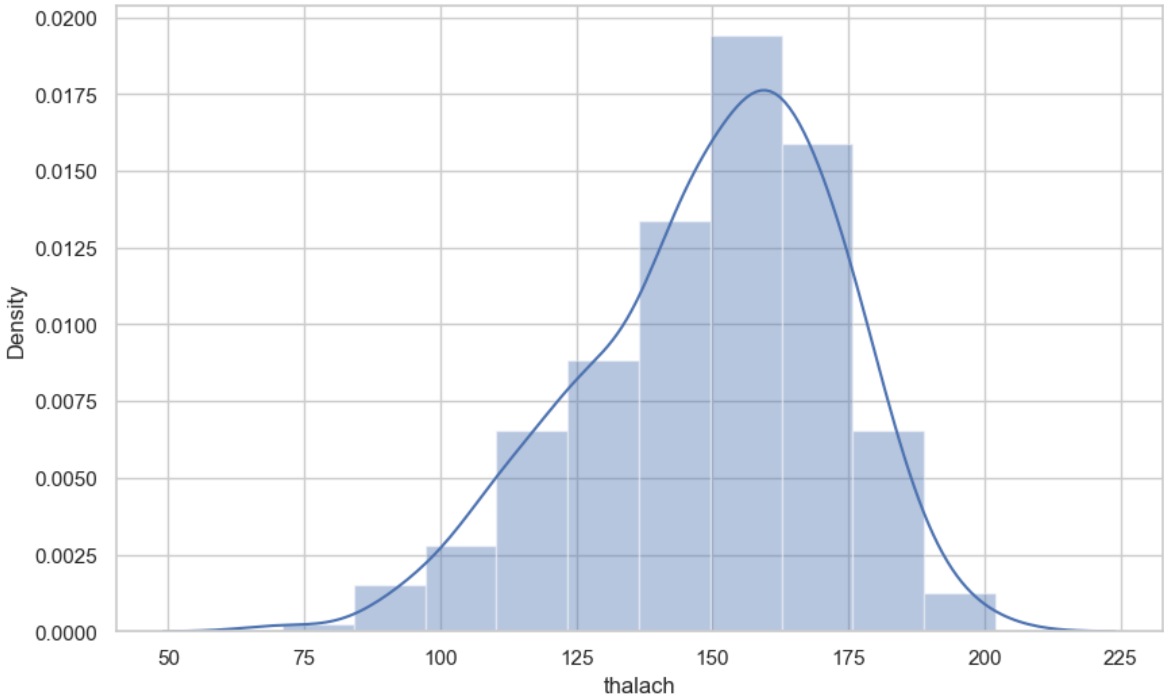
Histogram

- A histogram represents the distribution of data by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.

In [59]:
```
f, ax = plt.subplots(figsize= (10, 6))
x = df['thalach']
ax = sns.distplot(x, kde=False, rug = True, bins = 10)
plt.show()
```



In [60]:
```
f, ax = plt.subplots(figsize= (10, 6))
x = df['thalach']
ax = sns.distplot(x, kde=True, rug = False, bins = 10)
plt.show()
```

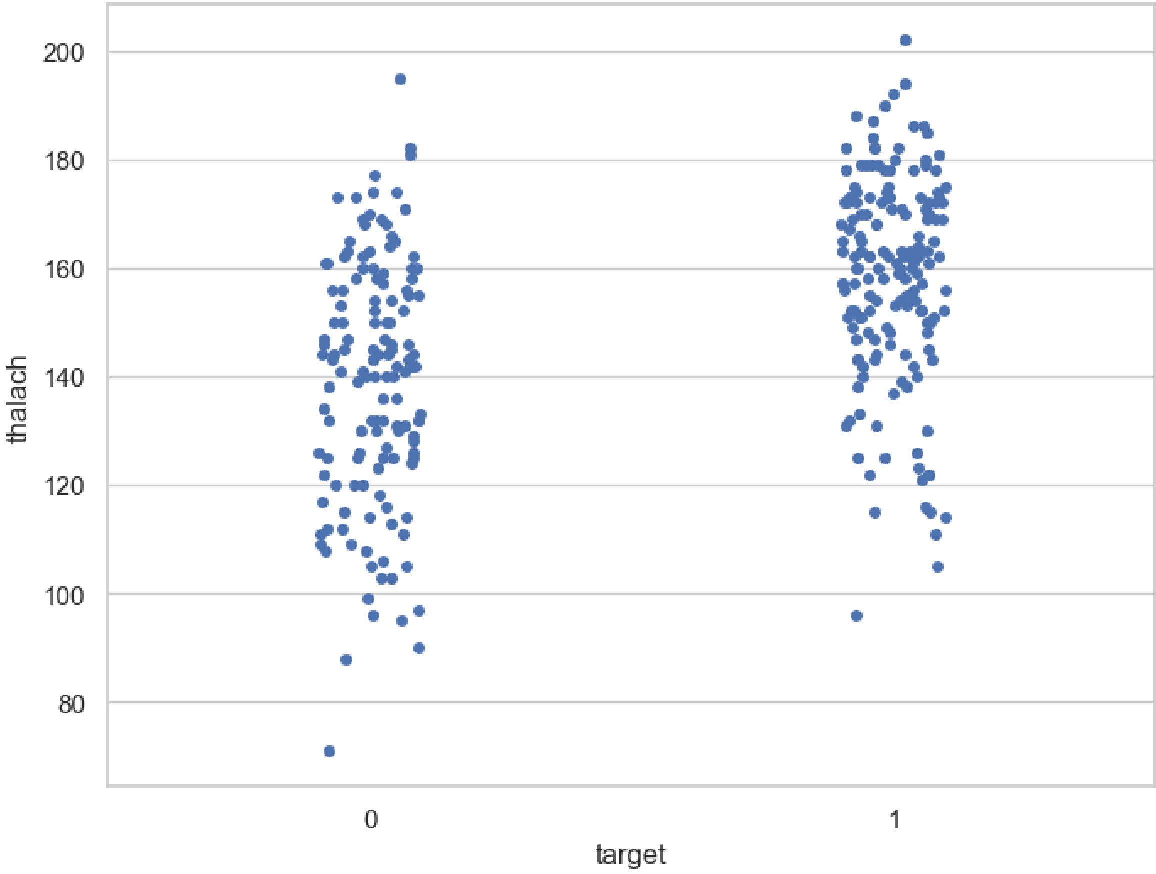


In [61]:
```
"""
x: The variable to plot the distribution of.
kde: Whether to plot a kernel density estimate (KDE) curve along with the hist
rug: Whether to plot a rug plot, which shows the individual data points.
bins: The number of bins to use for the histogram.
"""
```

Out[61]:
```
'\nx: The variable to plot the distribution of.\nkde: Whether to plot a kern
el density estimate (KDE) curve along with the histogram.\nrug: Whether to p
lot a rug plot, which shows the individual data points.\nbins: The number of
bins to use for the histogram.\n'
```

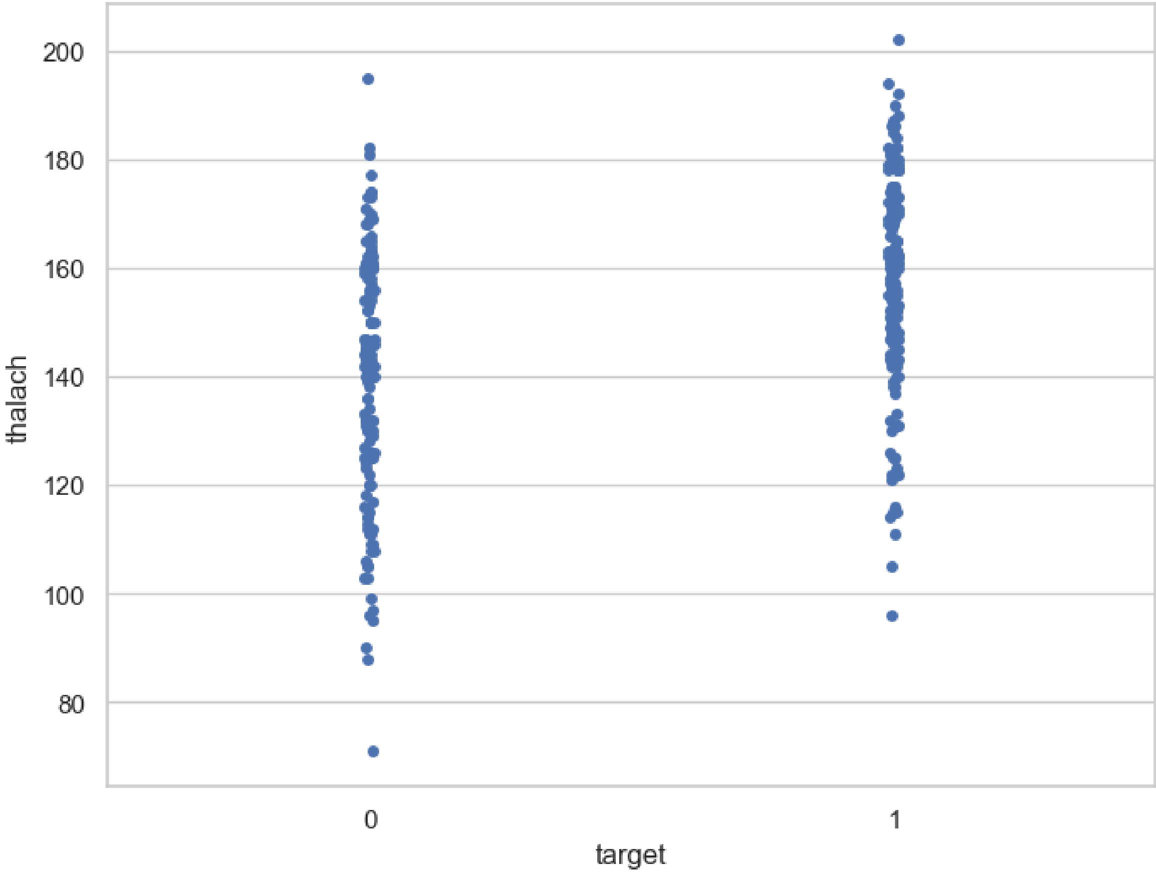
Visualize the frequence distribution of `thalach` varaible wrt `target`

In [62]:
```
f, ax = plt.subplots(figsize = (8, 6))
sns.stripplot(x='target', y = 'thalach', data = df)
plt.show()
```



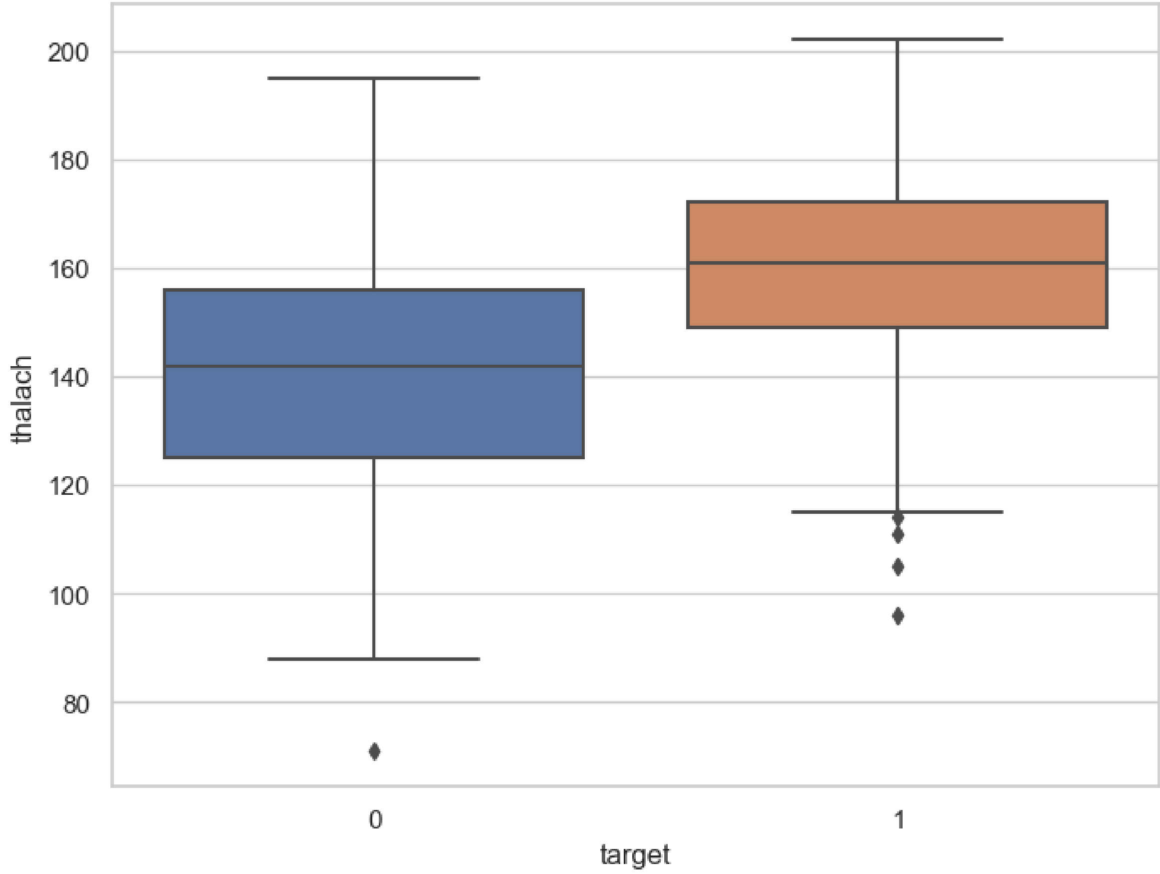
We can add jitter to bring out the distribution of values as follows :

In [63]:
```
f, ax = plt.subplots(figsize = (8, 6))
sns.stripplot(x='target', y = 'thalach', data = df, jitter= 0.01)
plt.show()
```



Visualize distribution of `thalach` variable wrt `target` with boxplot

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



Interpretation

The above boxplot confirms our finding that people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

Findings of Bivariate Analysis

Findings of Bivariate Analysis are as follows –

- There is no variable which has strong positive correlation with `target` variable.
- There is no variable which has strong negative correlation with `target` variable.
- There is no correlation between `target` and `fbs` .
- The `cp` and `thalach` variables are mildly positively correlated with `target` variable.
- We can see that the `thalach` variable is slightly negatively skewed.
- The people suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

Multivariate analysis

The objective of the multivariate analysis is to discover patterns and relationships in the

Discover patterns and relationships

- An important step in EDA is to discover patterns and relationships between variables in the dataset.
- I will use `heat map` and `pair plot` to discover the patterns and relationships in the dataset.

HEAT MAP

In [65]:
```
df
```

Out[65]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targe
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	

303 rows × 14 columns

In [66]:
```
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 [67]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [68]:

```
plt.figure(figsize=(16, 12))
plt.title('Correlation Heatmap of Heart Disease Dataset')
a = sns.heatmap(correlation, annot=True)

# a = sns.heatmap(correlation, annot=True, fmt='.2f', square=True, linecolor='
# a.set_xticklabels(a.get_xticklabels())
# a.set_yticklabels(a.get_yticklabels())
plt.show()
```

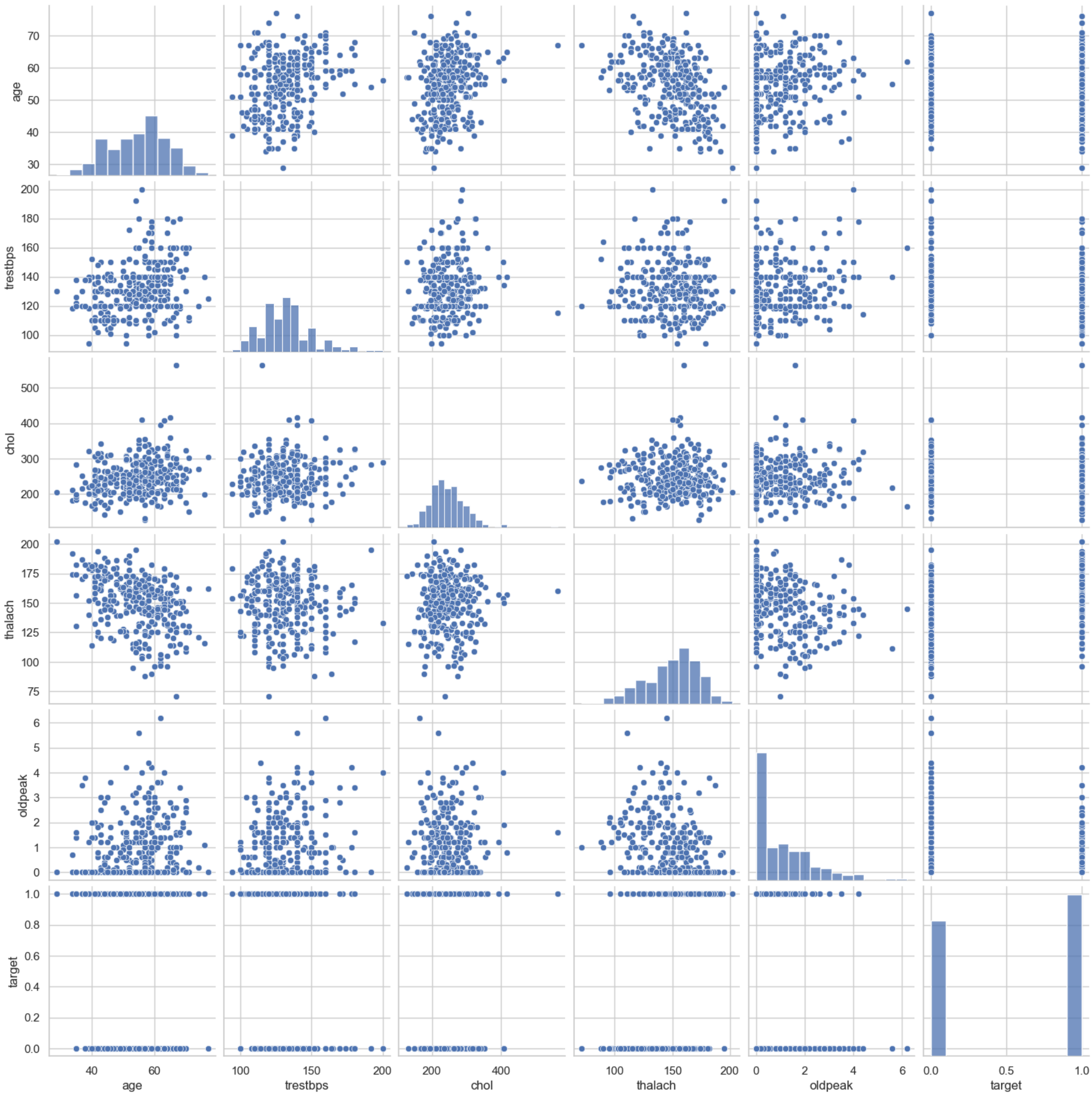
Interpretation

From the above correlation heat map, we can conclude that :-

- `target` and `cp` variable are mildly positively correlated (correlation coefficient = 0.43).
- `target` and `thalach` variable are also mildly positively correlated (correlation coefficient = 0.42).
- `target` and `slope` variable are weakly positively correlated (correlation coefficient = 0.35).
- `target` and `exang` variable are mildly negatively correlated (correlation coefficient = -0.44).
- `target` and `oldpeak` variable are also mildly negatively correlated (correlation coefficient = -0.43).
- `target` and `ca` variable are weakly negatively correlated (correlation coefficient = -0.39).
- `target` and `thal` variable are also waekly negatively correlated (correlation coefficient = -0.34).

Pair Plot

In [69]:
```
num_var = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target']
sns.pairplot(df[num_var], kind= 'scatter', diag_kind='hist')
plt.show()
```



astype()

In [70]:
```
movies.Film = movies.Film.astype('category')
movies.Film
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[70], line 1
----> 1 movies.Film = movies.Film.astype('category')
      2 movies.Film

NameError: name 'movies' is not defined
```

In [ ]:
```
df.target = df.target.astype('category')
df.target
```

In [ ]:
```
df.info()
```

In [ ]:
```
df.target = df.target.astype('int')
df.target
```

In [ ]:
```
df.info()
```

Analysis of `age` and other variables

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

view statistical summary of `age` varaible

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

Plot the distribution of `age` variable

By assigning f and ax explicitly, you can easily modify various properties of the figure and the subplot if needed, making your code more customizable and readable when working with complex plots or multiple subplots.

Here's an explanation of their roles:

f (often named fig):

- This variable represents the Matplotlib figure, which is essentially the canvas or container for your plots.
- You can use the figure (f) to set the overall properties of the plot, such as the figure size, title, and additional customization that applies to the entire figure.

ax (often named axes):

- This variable represents the Axes object, which is the specific plot or subplot within the figure.
- You use the ax variable to customize properties specific to the individual plot, such as labels, colors, legends, and more.
- In your code, you are creating a histogram (using sns.distplot) and customizing its appearance by modifying the ax object.

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

In [ ]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x='target', y = 'age', data=df)
plt.show()
```

In [ ]:
```
sns.stripplot(x='target', y = 'age', data=df)
plt.show()
```

Interpretation

- We can see that the people suffering from heart disease (target = 1) and people who are not suffering from heart disease (target = 0) have comparable ages.

Visualize distribution of `age` variable wrt `target` with boxplot

IN SUBPPLOT WE PLOT MULTIPLE DIAGRAM THEN WHY WE ARE PLOTTING ONE DIAGRAM HERE

In [ ]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x ='target', y='age', data=df)
plt.show()
```

Analyze `age` and `trestbps` variable

I will plot a scatterplot to visualize the relationship between `age` and `trestbps` variable.

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

Interpretation

- The above scatter plot shows that there is no correlation between `age` and `trestbps` variable.

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

Interpretation

- The above line shows that linear regression model is not good fit to the data.

Analyze `age` and `chol` variable

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

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

Interpretation

- The above plot confirms that there is a slighly positive correlation between `age` and `chol` variables.

Analyze `chol` and `thalach` variable

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

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

Interpretation

- The above plot shows that there is no correlation between `chol` and `thalach` variable.

Dealing with missing values

- ```
  In Pandas missing data is represented by two values:
  ```
  - ```
    **None**: None is a Python singleton object that is often used
    for missing data in Python code.
    ```
  - ```
    **NaN** : NaN (an acronym for Not a Number), is a special floa
    ting-point value recognized by all systems that use the standa
    rd IEEE floating-point representation.
    ```
- There are different methods in place on how to detect missing values.
    ```

Pandas isnull() and notnull() functions

- Pandas offers two functions to test for missing data - `isnull()` and `notnull()` . These are simple functions that return a boolean value indicating whether the passed in argument value is in fact missing data.
- Below, I will list some useful commands to deal with missing values.

Useful commands to detect missing values

- `**df.isnull()**`

The above command checks whether each cell in a dataframe contains missing values or not. If the cell contains missing value, it returns True otherwise it returns False.

- `**df.isnull().sum()**`

The above command returns total number of missing values in each column in the dataframe.

- `**df.isnull().sum().sum()**`

It returns total number of missing values in the dataframe.

- `**df.isnull().mean()**`

It returns percentage of missing values in each column in the dataframe.

- `**df.isnull().any()**`

It checks which column has null values and which has not. The columns which has null values returns TRUE and FALSE otherwise.

- `**df.isnull().any().any()**`

It returns a boolean value indicating whether the dataframe has missing values or not. If dataframe contains missing values it returns TRUE and FALSE otherwise.

- `**df.isnull().values.any()**`

It checks whether a particular column has missing values or not. If the column contains missing values, then it returns TRUE otherwise FALSE.

- `**df.isnull().values.sum()**`

In [73]:
```
# CHECK FOR MISSING VALUES
df.isnull().sum()
```

Out[73]:
```
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
```

Check with ASSERT statement

- We must confirm that our dataset has no missing values.
- We can write an **assert statement** to verify this.
- We can use an assert statement to programmatically check that no missing, unexpected 0 or negative values are present.
- This gives us confidence that our code is running properly.
- **Assert statement** will return nothing if the value being tested is true and will throw an AssertionError if the value is false.
- **Asserts**
  - assert 1 == 1 (return Nothing if the value is True)
  - assert 1 == 2 (return AssertionError if the value is False)

In [77]:
```
#assert that there are no missing values in the dataframe
assert pd.notnull(df).all().all()
```

In [78]:
```
#assert all values are greater than or equal to 0
assert (df >= 0).all().all()
```

Interpretation

- The above two commands do not throw any error. Hence, it is confirmed that there are no missing or negative values in the dataset.
- All the values are greater than or equal to zero.

Outlier Detection

I will make boxplots to visualise outliers in the continuous numerical variables : -

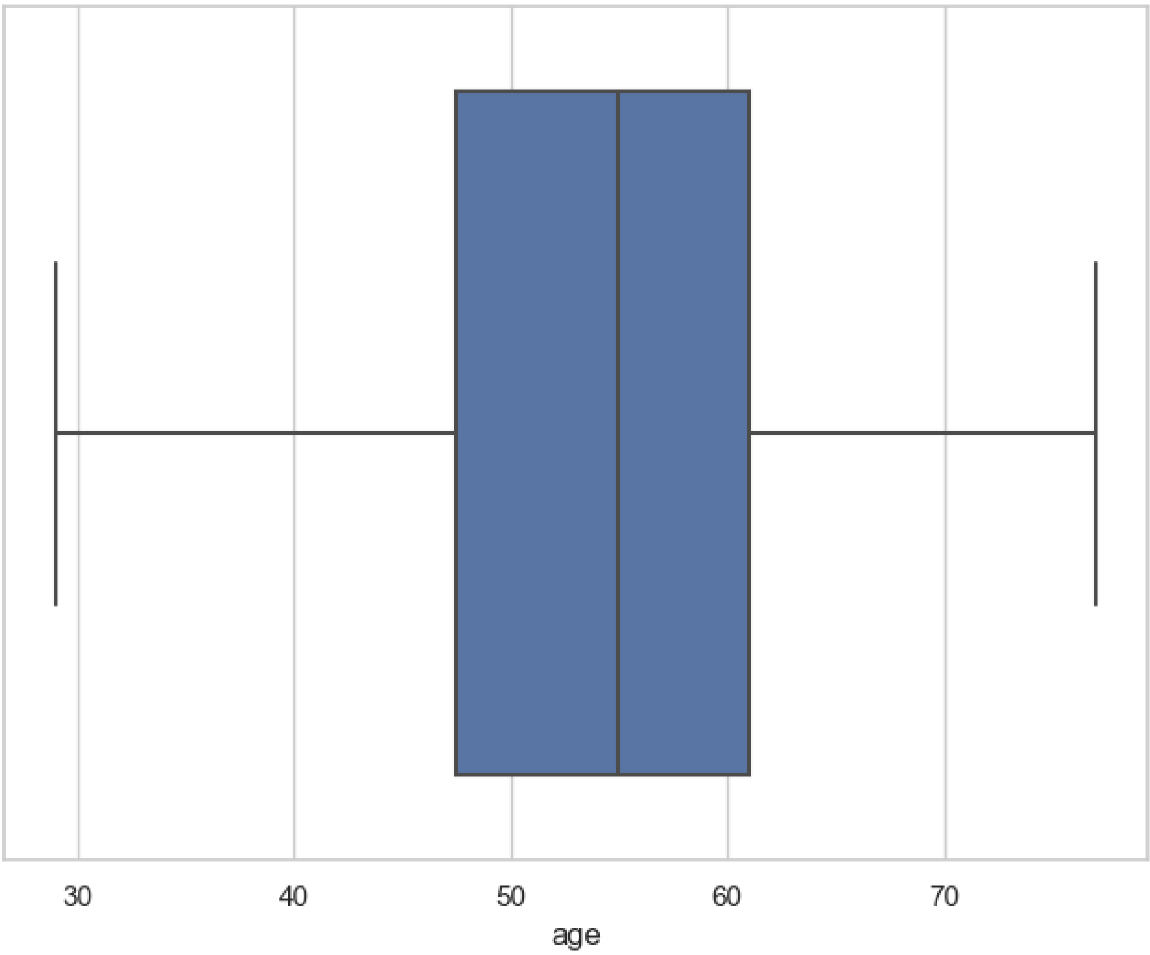
`age` , `trestbps` , `chol` , `thalach` and `oldpeak` variables.

age varaible

In [79]:
```
df['age'].describe()
```

Out[79]:
```
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 [81]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df['age'])
plt.show()
```

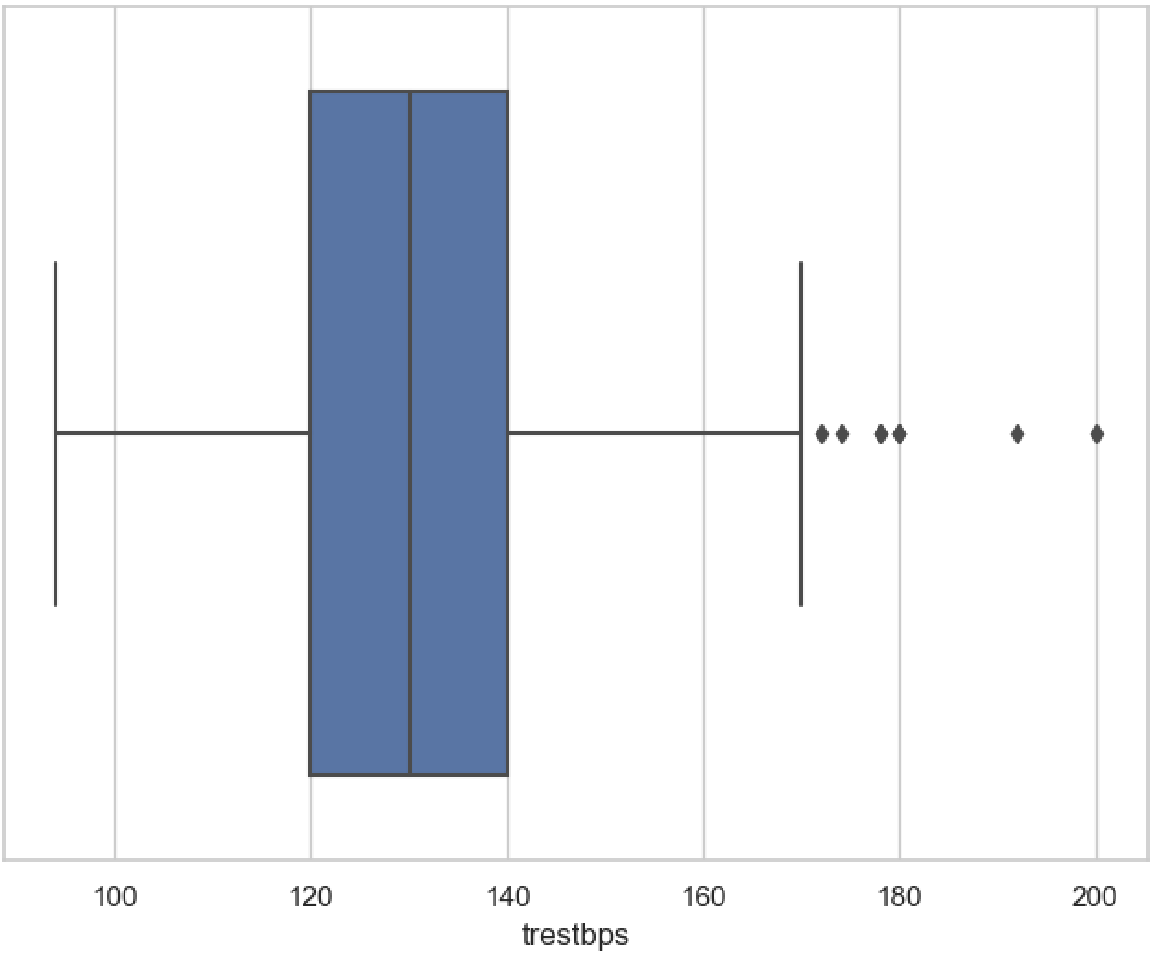


trestbps varaible

In [82]:
```
df['trestbps'].describe()
```

Out[82]:
```
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 [83]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["trestbps"])
plt.show()
```

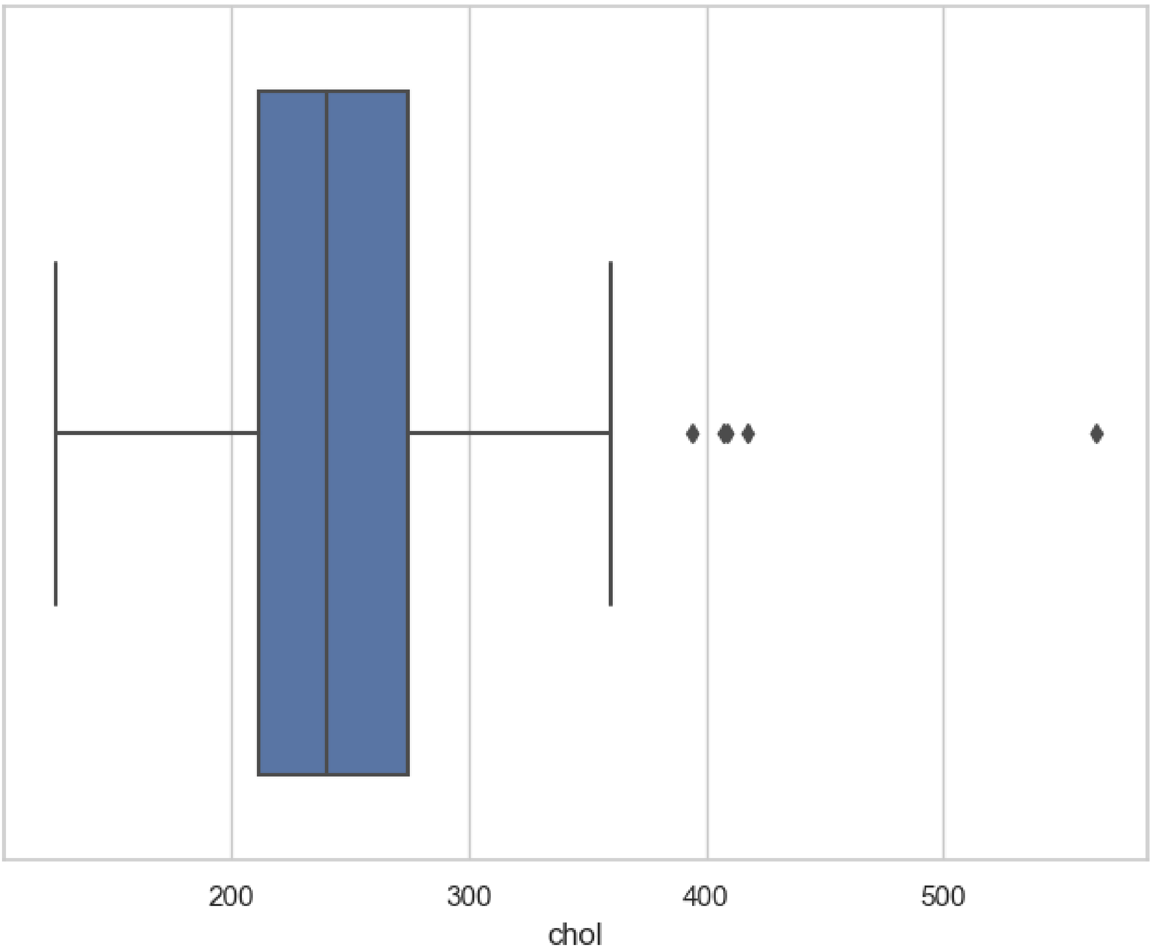


chol varaible

In [84]:
```
df['chol'].describe()
```

Out[84]:
```
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
```

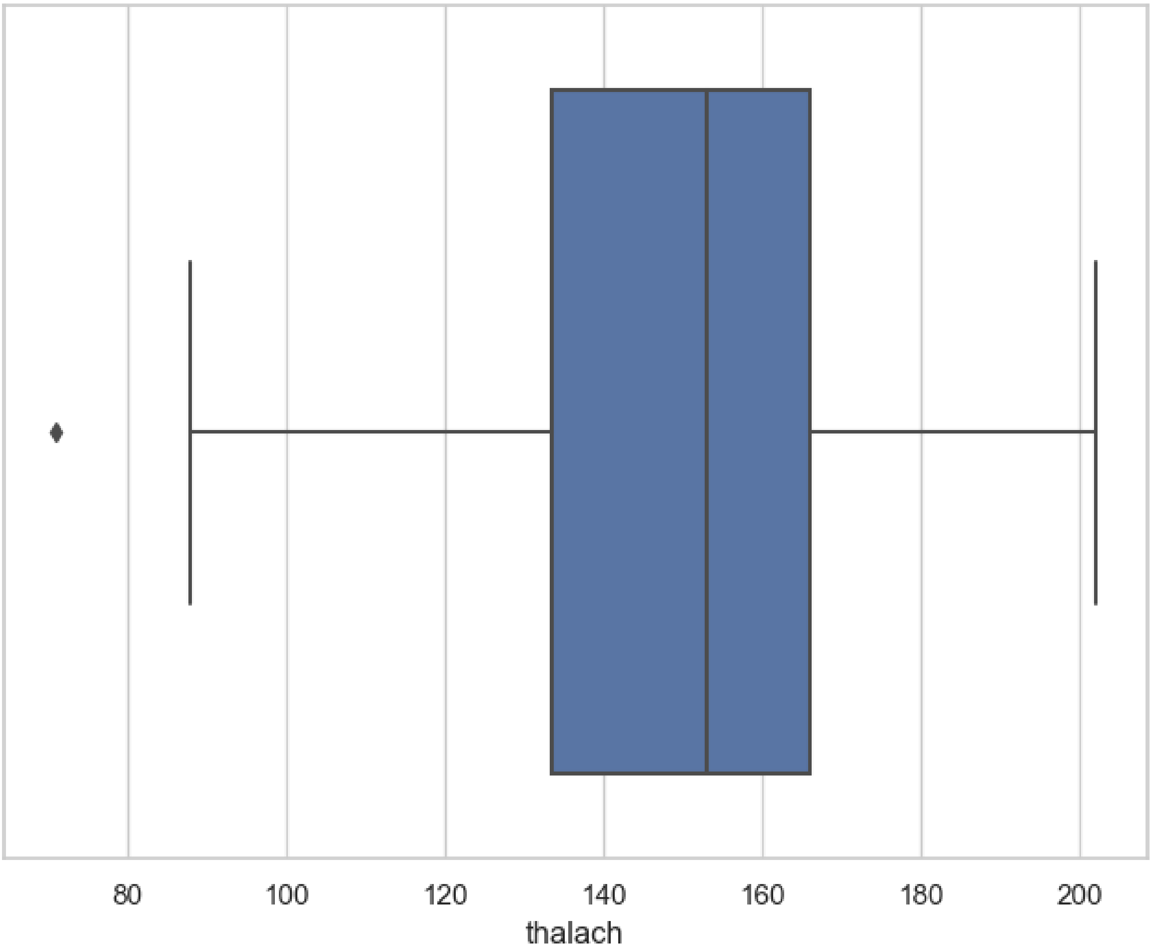
In [85]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["chol"])
plt.show()
```



thalach varaible

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

In [86]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["thalach"])
plt.show()
```

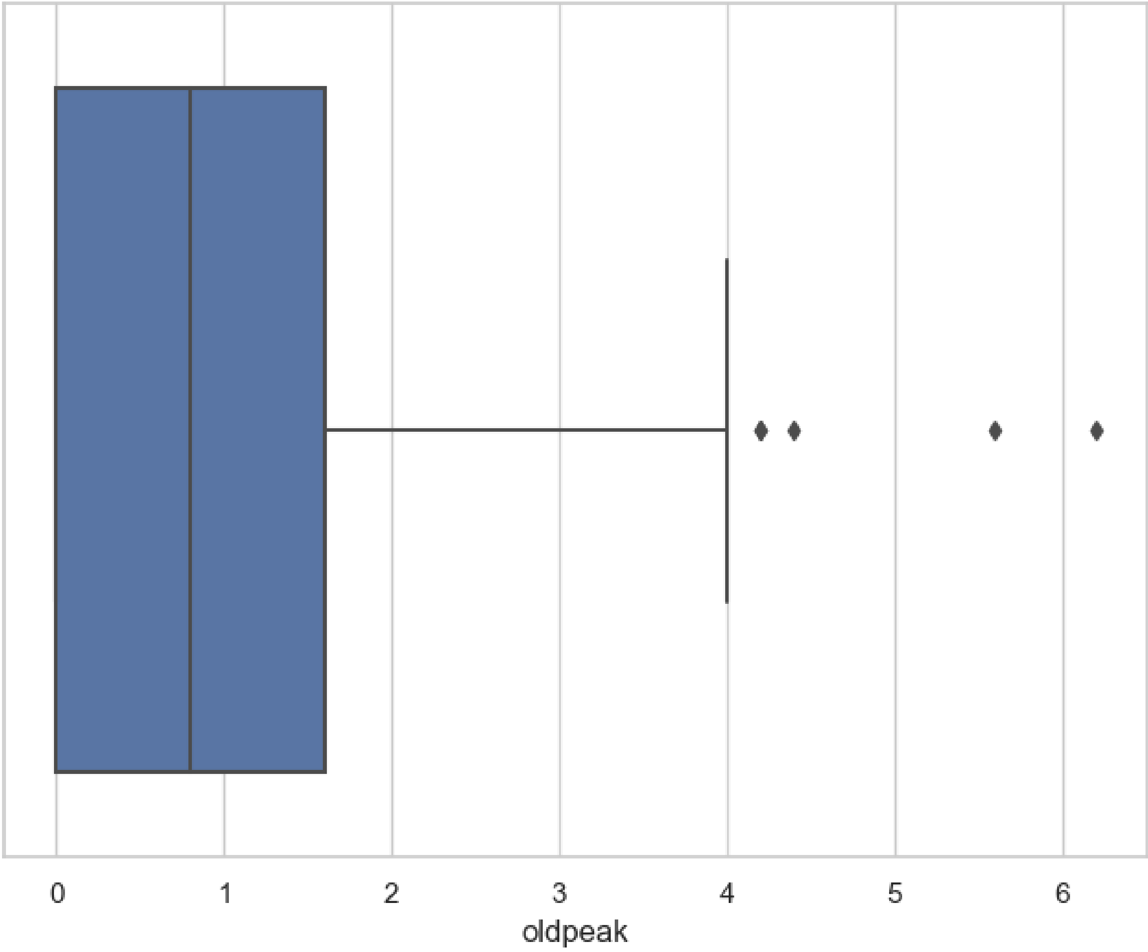


oldpeak varaible

In [87]:
```
df['oldpeak'].describe()
```

Out[87]:
```
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 [88]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["oldpeak"])
plt.show()
```



Findings

- The `age` variable does not contain any outlier.
- `trestbps` variable contains outliers to the right side.
- `chol` variable also contains outliers to the right side.
- `thalach` variable contains a single outlier to the left side.
- `oldpeak` variable contains outliers to the right side.
- Those variables containing outliers needs further investigation.

Conclusion

In [ ]:
```

```