EDA Project | Heart Disease UCI | Data from Kaggle.com

Import Libraries

In [13]:
```
import pandas as pd
import seaborn as sns #imports the Seaborn library
import matplotlib.pyplot as plt # imports the Matplotlib library
import scipy.stats as st #imports the SciPy library
%matplotlib inline
sns.set(style="darkgrid") #dark background with horizontal and vertical grid lines t
```

In [14]:
```
import warnings
warnings.filterwarnings('ignore') # ignore warnings
```

In [17]:
```
hd = pd.read_csv(r'C:\Users\Me\OneDrive\Data Science\0504\5th - Seaborn, Eda practic
```

In [20]:
```
print('The shape of the dataset : ', hd.shape) # print the dataset shape
```

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

In [21]:
```
hd.head() # preview dataset
```

Out[21]:

	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 [23]:
```
hd.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 [24]:
```
hd.dtypes
```

Out[24]:
```
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 [26]:
```
hd.describe() # statistical properties of dataset
```

Out[26]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalac
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64686
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.90516
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.50000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.00000

In [43]:
```
hd.describe(include=[object'='])
```

```
  File "<ipython-input-43-b3848279cbc5>", line 1
    hd.describe(include=[object'='])
                                ^
SyntaxError: invalid syntax
```

In [42]:
```
hd.describe(include='all')
```

Out[42]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalac
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64686
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.90516
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.50000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.00000

In [44]:
```
hd.columns #columns name
```

Out[44]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [46]:
```
hd['target'].nunique() #number of unique values in target variable
```

Out[46]:
```
2
```

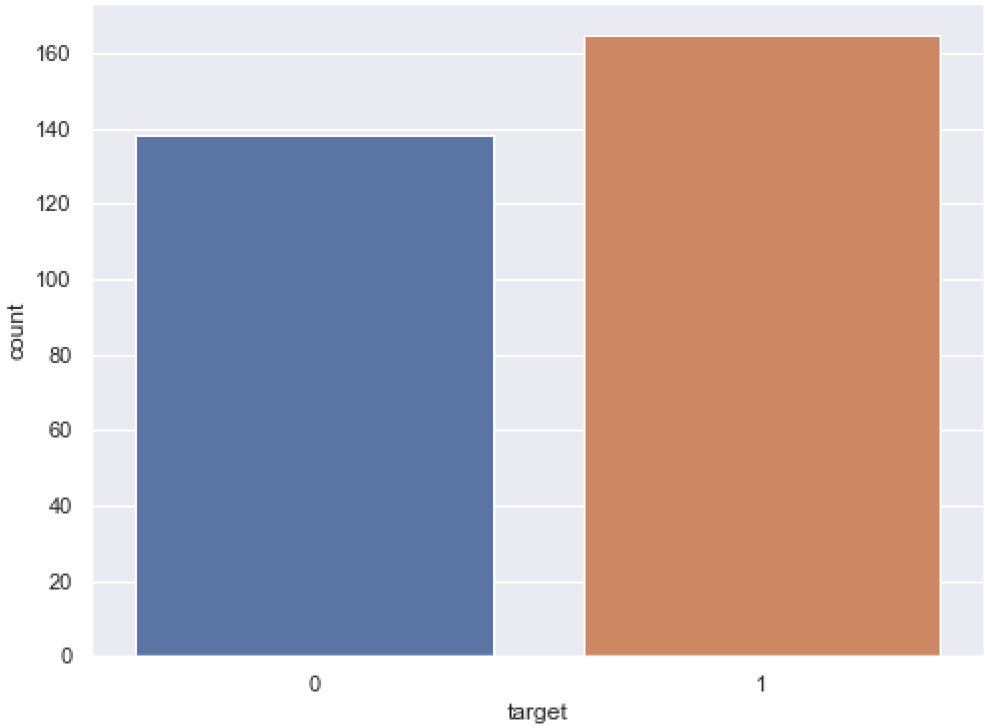
In [48]:
```
hd['target'].unique() #unique values in target variable
```

Out[48]:
```
array([1, 0], dtype=int64)
```

In [50]:
```
hd['target'].value_counts() #Frequency distribution of target variable
```

Out[50]:
```
1    165
0    138
Name: target, dtype: int64
```

In [51]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=hd)
plt.show()
```



Interpretation The above plot confirms the findings that -

There are 165 patients suffering from heart disease, and

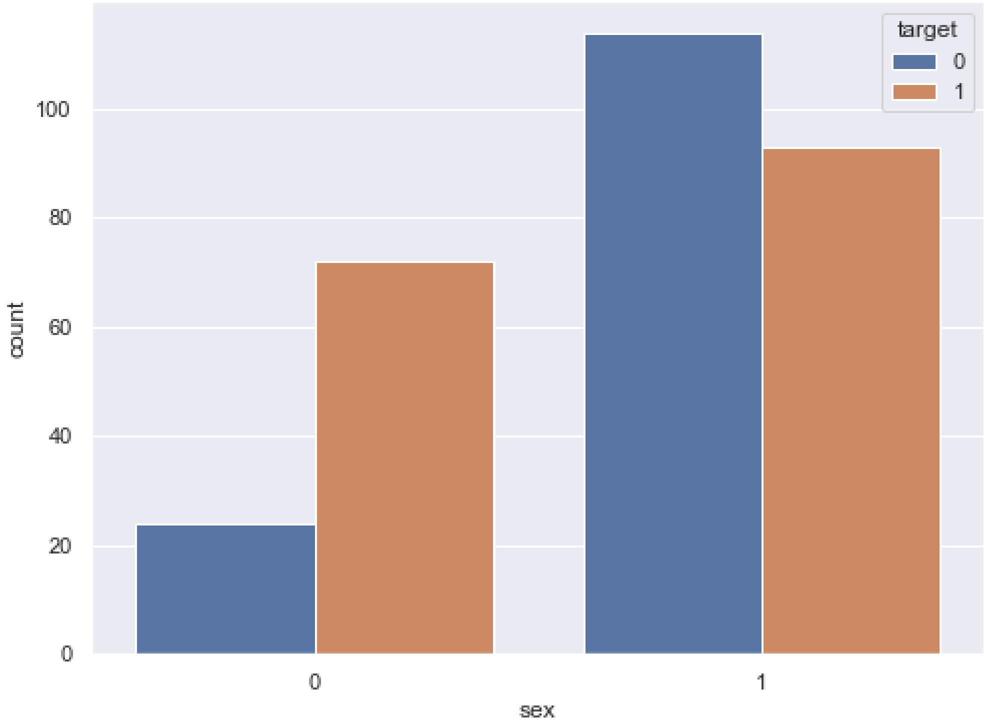
There are 138 patients who do not have any heart disease.

In [53]:
```
hd.groupby('sex')['target'].value_counts()
```

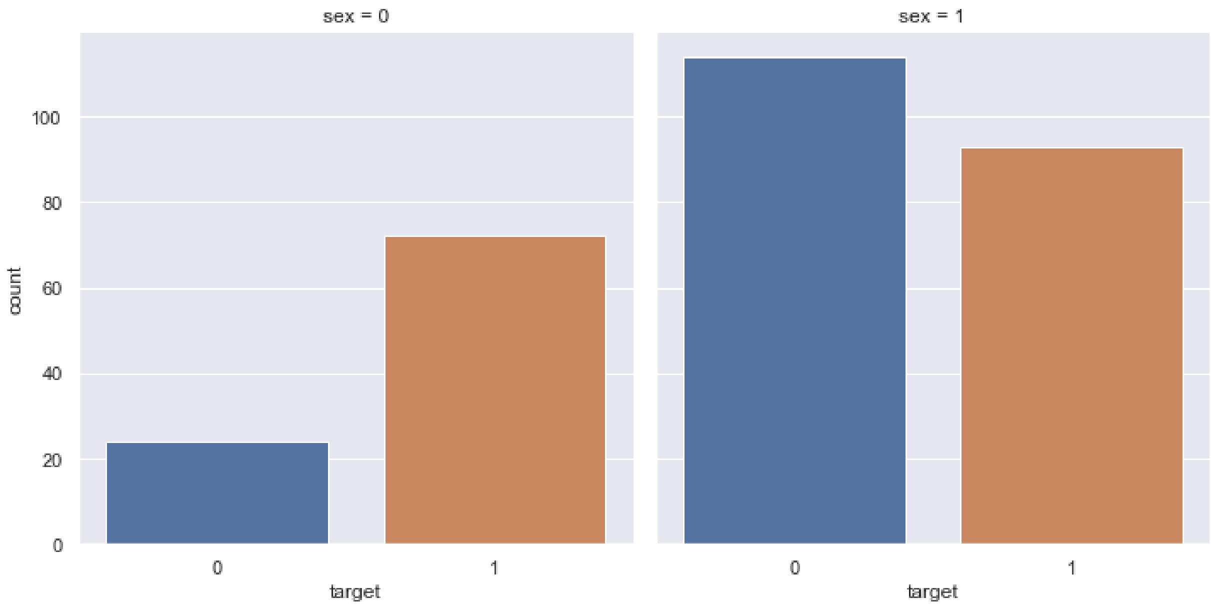
Out[53]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: target, dtype: int64
```

In [59]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="sex", hue="target", data=hd)
```

```
plt.show()
```



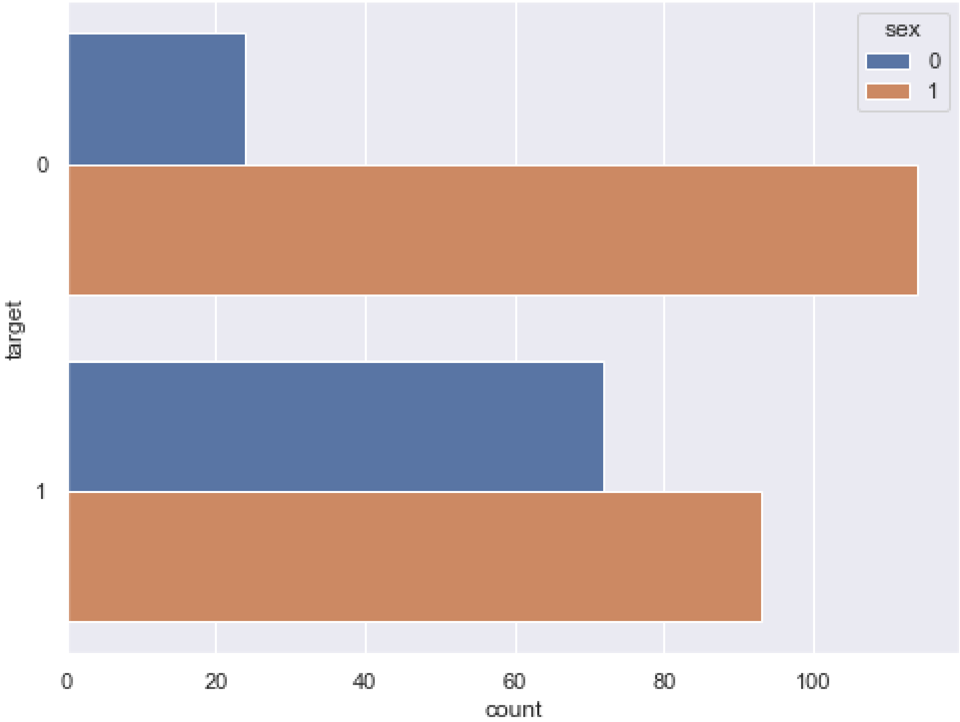
In [61]:
```
ax = sns.catplot(x="target", col="sex", data=hd, kind="count", height=5, aspect=1)
```



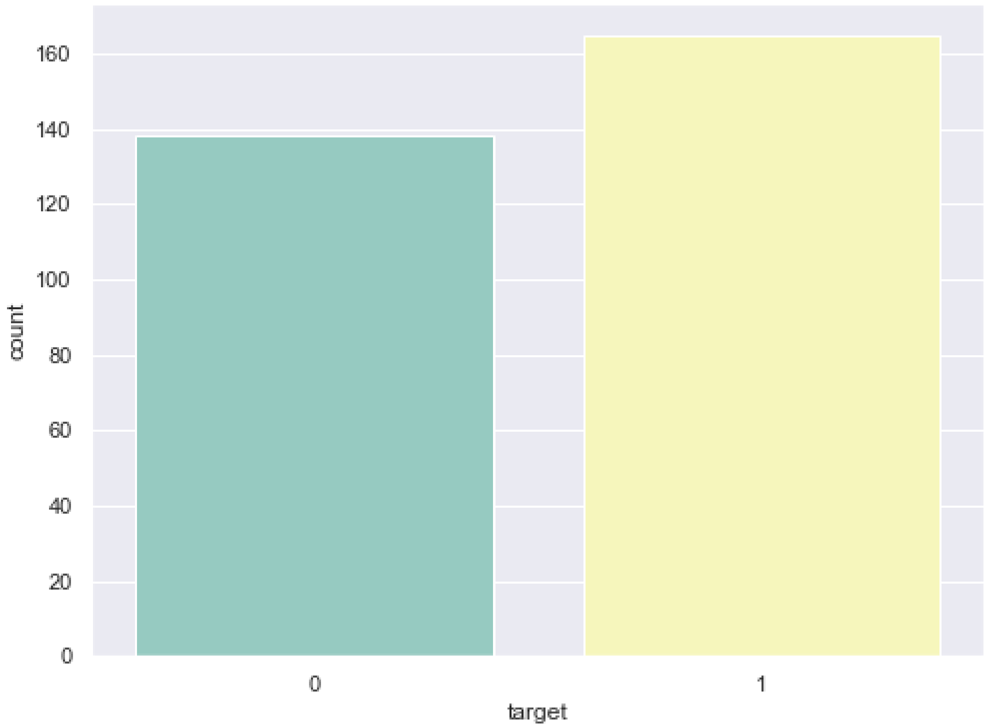
The above plot segregate the values of target variable and plot on two different columns labelled as (sex = 0, sex = 1).

I think it is more convinient way of interpret the plots.

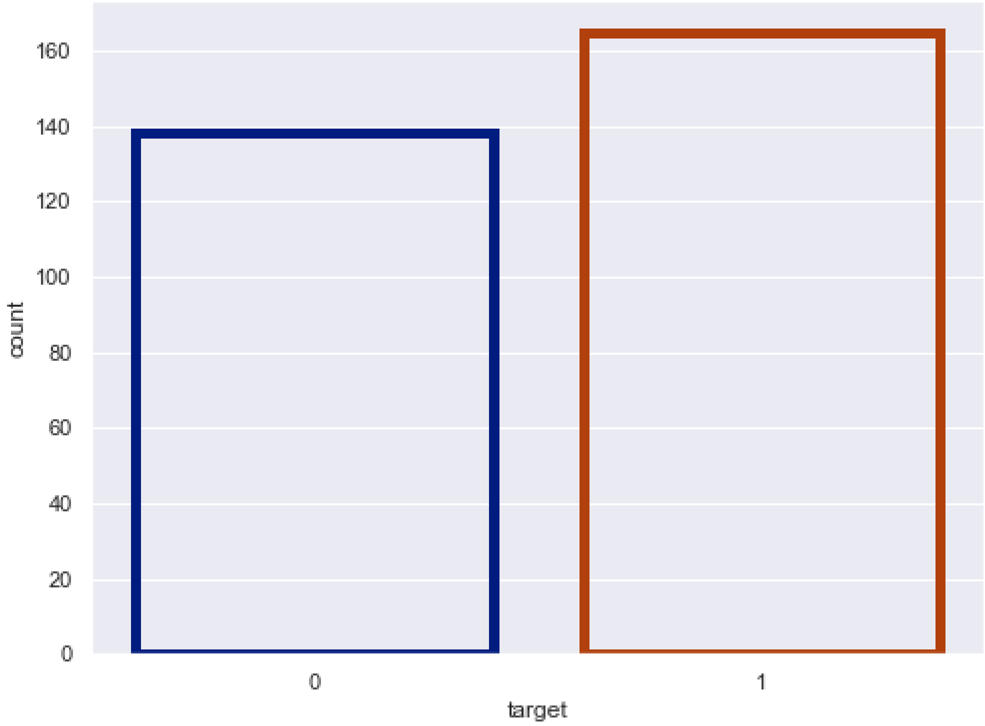
In [63]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(y="target", hue="sex", data=hd)
plt.show()   # horizontal plot
```



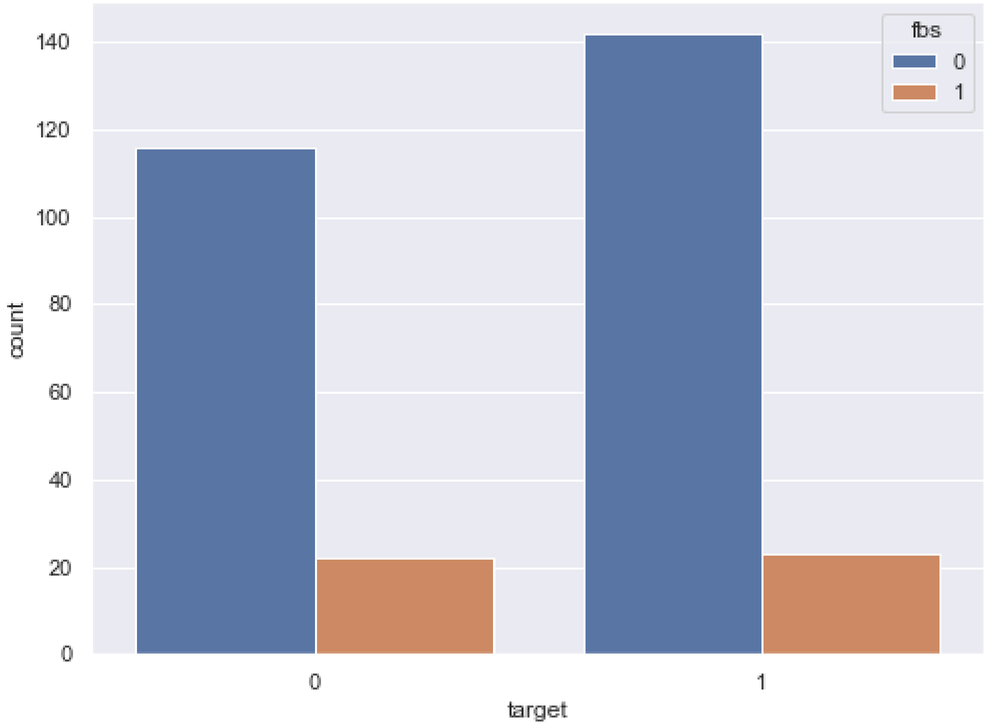
In [64]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=hd, palette="Set3") #adding colour
plt.show()
```



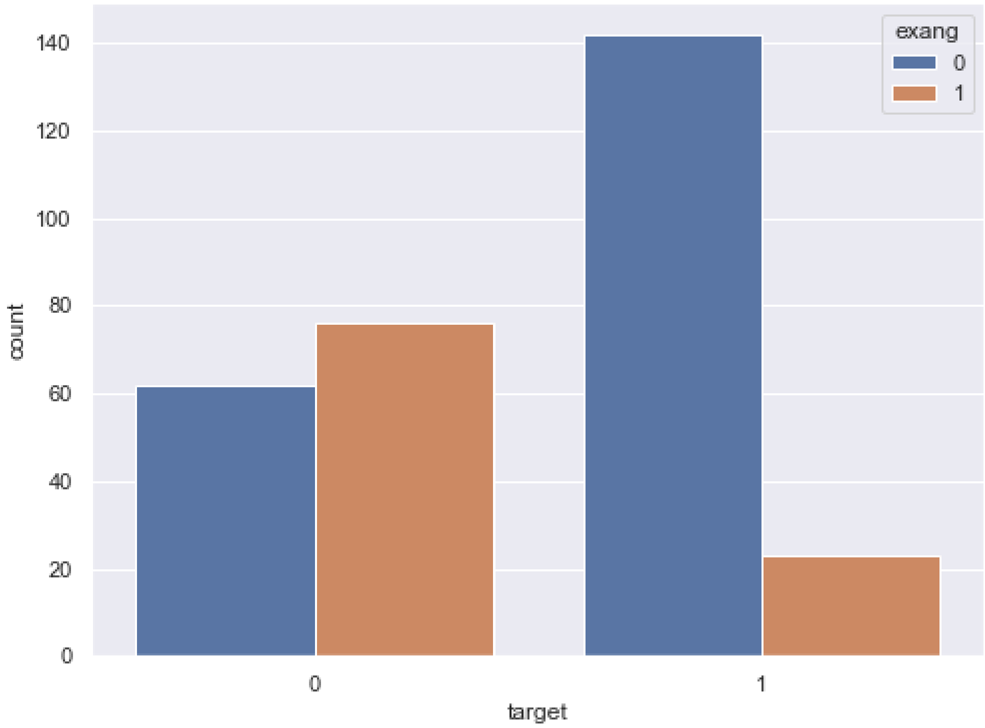
In [65]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=hd, facecolor=(0, 0, 0, 0), linewidth=5, edgecol
plt.show()
```



In [66]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", hue="fbs", data=hd)
plt.show()
```



In [67]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", hue="exang", data=hd)
plt.show()
```



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

Estimate correlation coefficients

In [68]:
```
correlation = hd.corr()
```

In [69]:
```
correlation['target'].sort_values(ascending=False)
```

Out[69]:
```
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 [71]:

```
hd['cp'].nunique() #check number of unique values in cp variable
```

Out[71]:

```
4
```

In [73]:

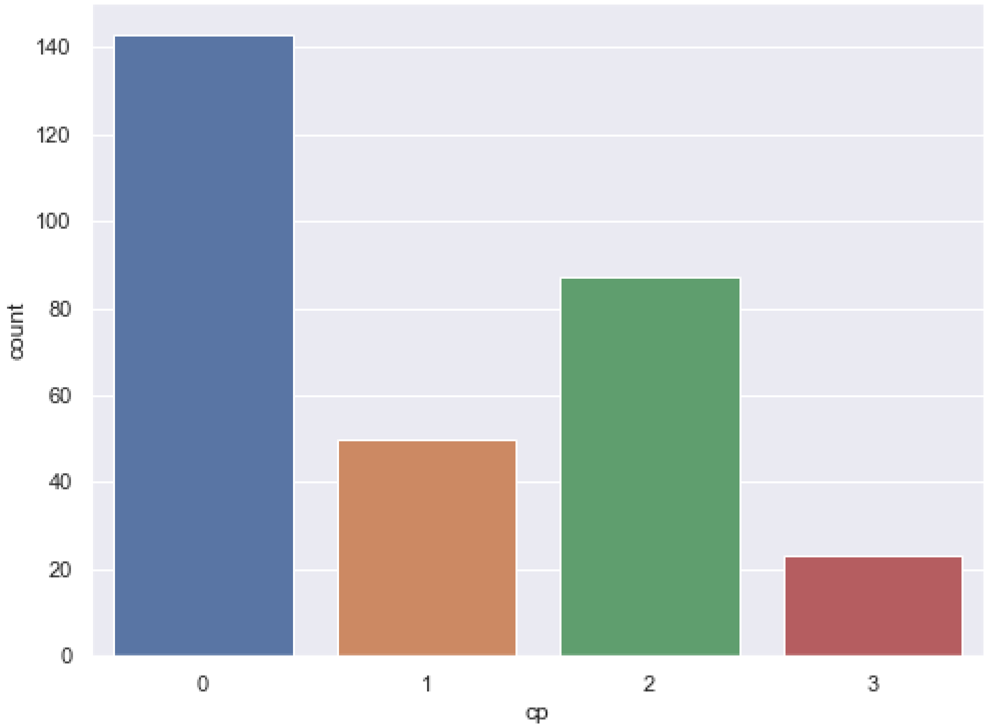
```
hd['cp'].value_counts() #frequency distribution
```

Out[73]:

```
0    143
2     87
1     50
3     23
Name: cp, dtype: int64
```

In [76]:

```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="cp", data=hd) #Frequency distribution of cp
plt.show()
```



In [79]:

```
hd.groupby('cp')['target'].value_counts()  #frequency distribution of target wrt cp
```

Out[79]:

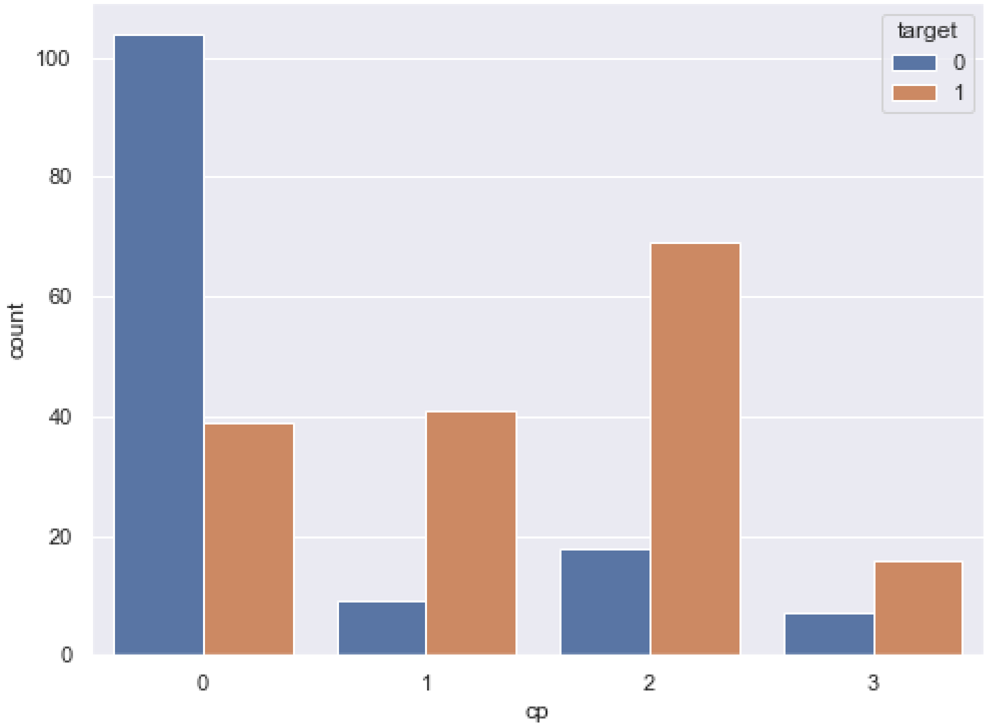
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: target, dtype: int64
```

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.

In [80]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="cp", hue="target", data=hd)
plt.show()
```

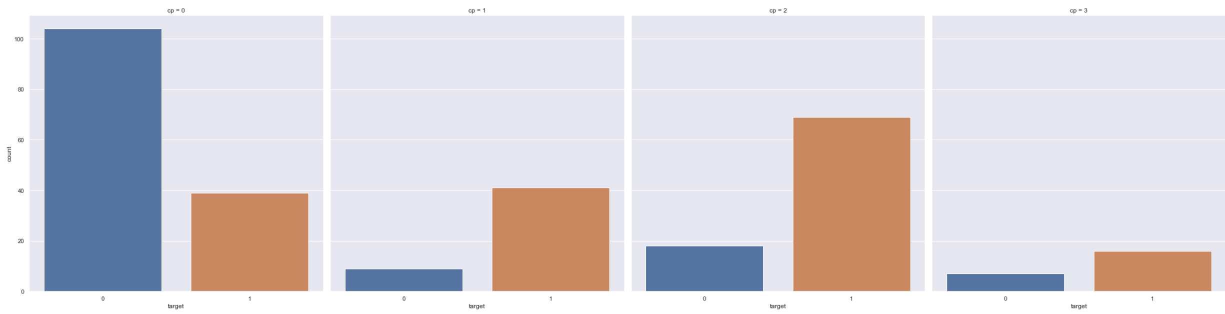


Interpretation We can see that the values of target variable are plotted wrt cp.

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 [83]:
```
ax = sns.catplot(x="target", col="cp", data=hd, kind="count", height=8, aspect=1)
```



Analysis of `target` and `thalach` variable

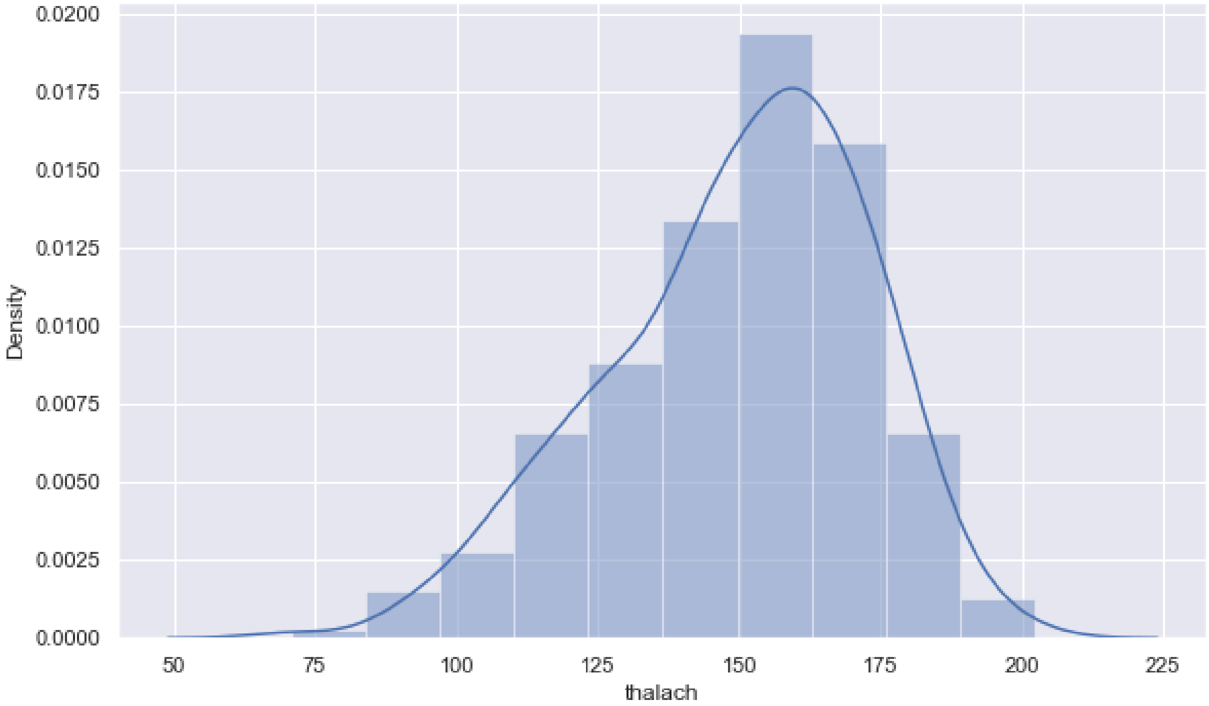
In [85]:
```
hd['thalach'].nunique()
```

Out[85]: 91

Number of unique values in thalach variable is 91. Hence, it is numerical variable.

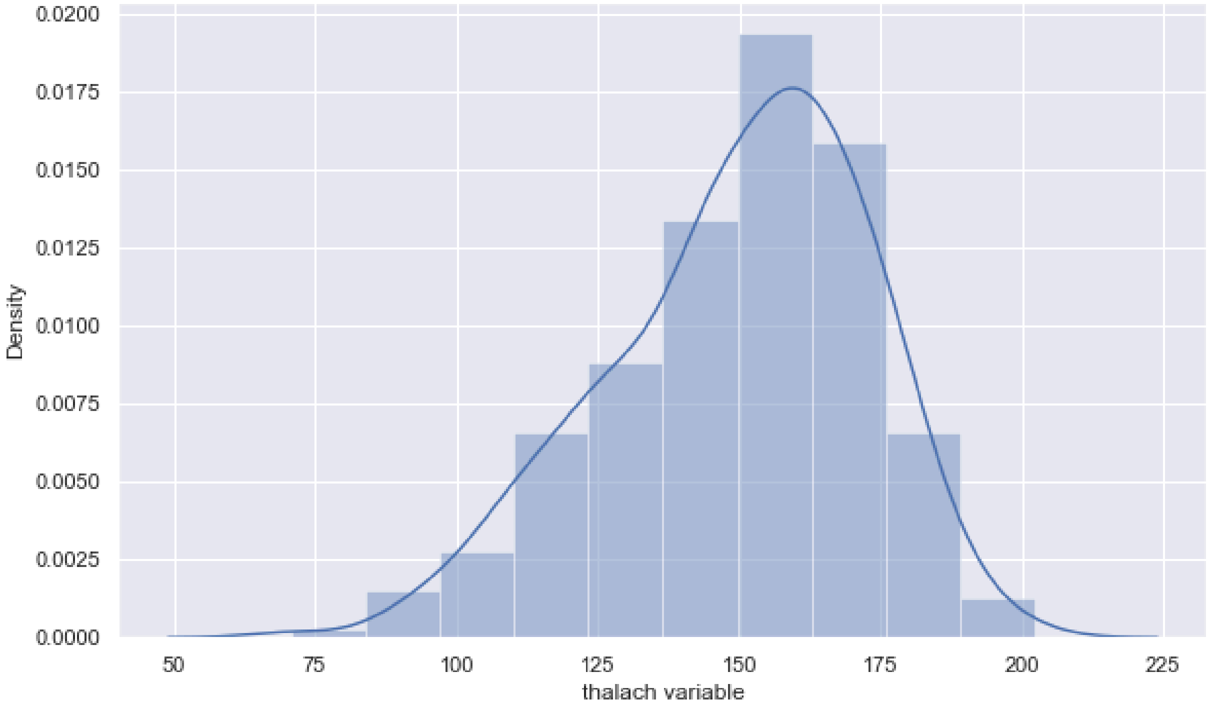
Visualize the frequency distribution of `thalach` variable

In [86]:
```
f, ax = plt.subplots(figsize=(10,6))
x = hd['thalach']
ax = sns.distplot(x, bins=10)
plt.show()
```



The thalach variable is slightly negatively skewed

In [87]:
```
f, ax = plt.subplots(figsize=(10,6))
x = hd['thalach']
x = pd.Series(x, name="thalach variable") #using panda series
ax = sns.distplot(x, bins=10)
plt.show()
```

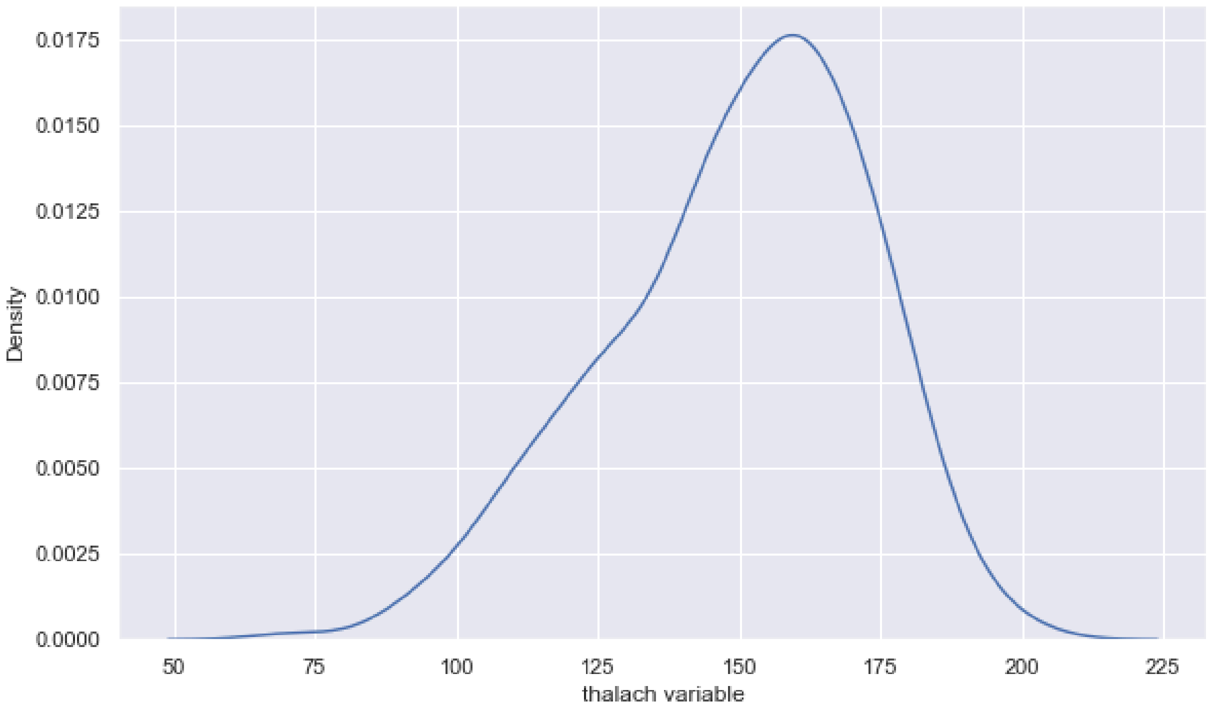


Seaborn Kernel Density Estimation (KDE) Plot

KDE plot is a useful tool for plotting the shape of a distribution.

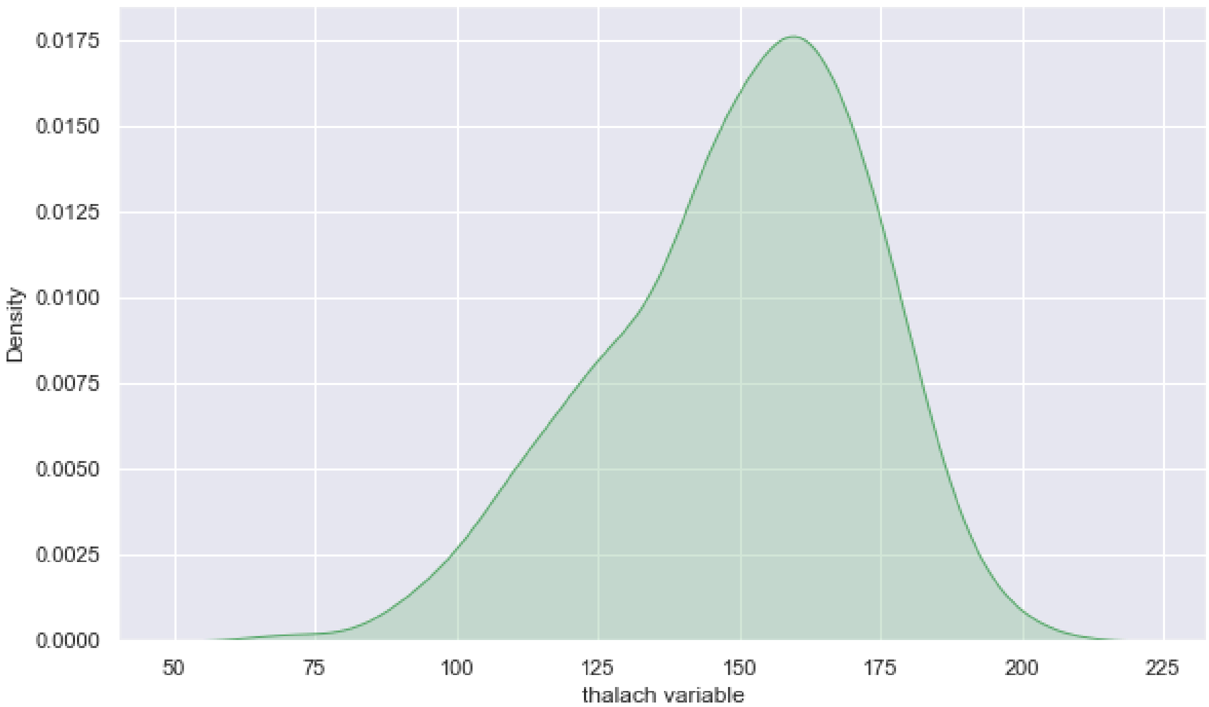
The KDE plot plots the density of observations on one axis with height along the other axis.

In [88]:
```
f, ax = plt.subplots(figsize=(10,6))
x = hd['thalach']
x = pd.Series(x, name="thalach variable")
ax = sns.kdeplot(x)
plt.show()
```



In [89]:
```
f, ax = plt.subplots(figsize=(10,6))
x = hd['thalach']
x = pd.Series(x, name="thalach variable")
```

```
ax = sns.kdeplot(x, shade=True, color='g')
plt.show()
```

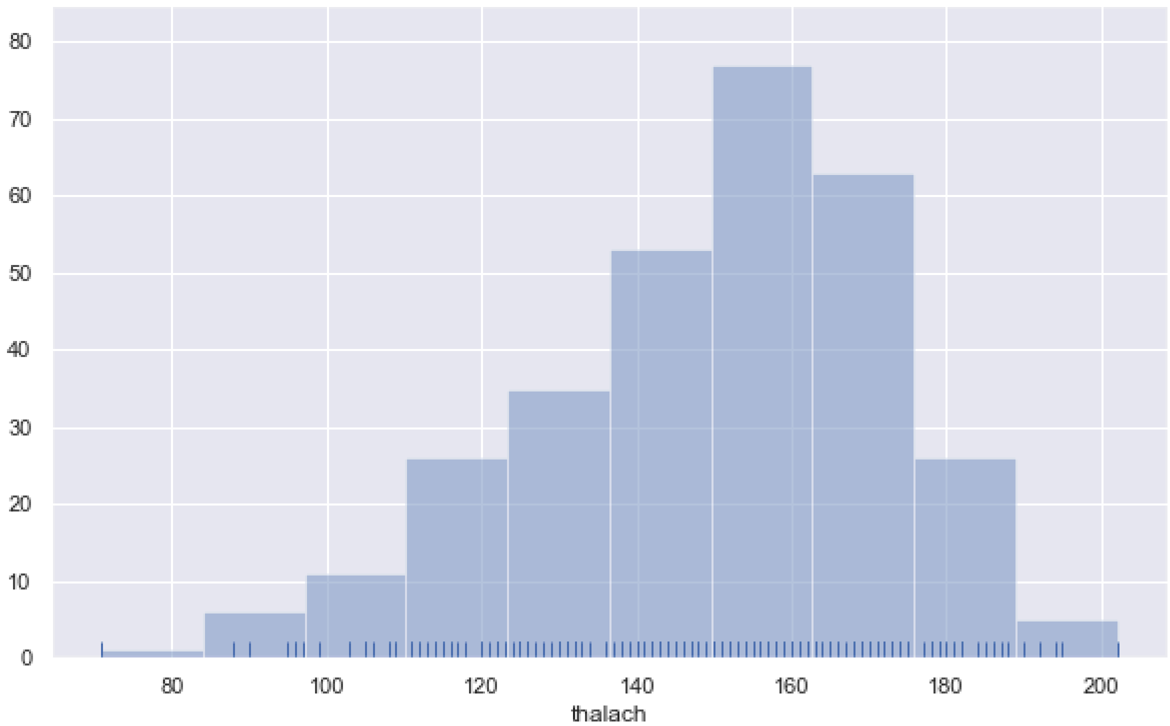


Histogram

- 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.

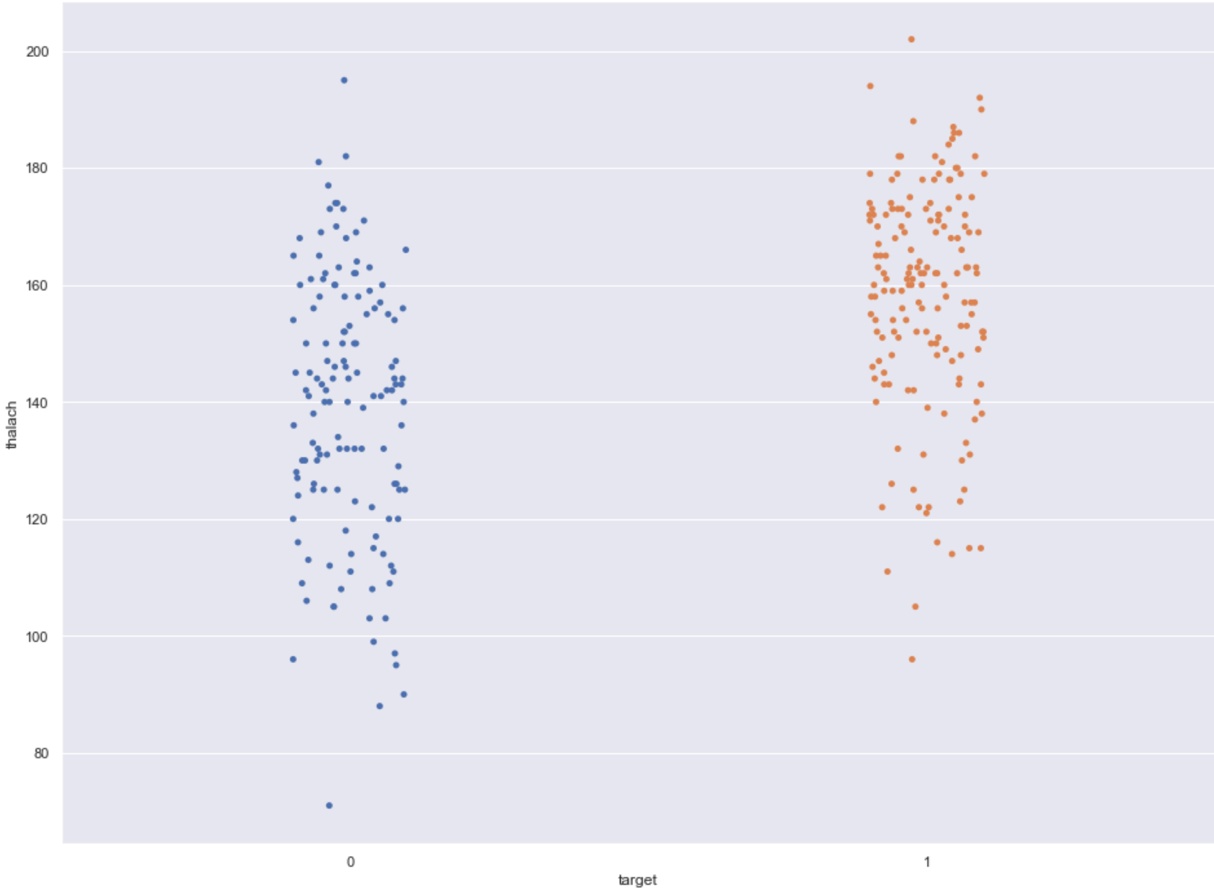
In [90]:

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



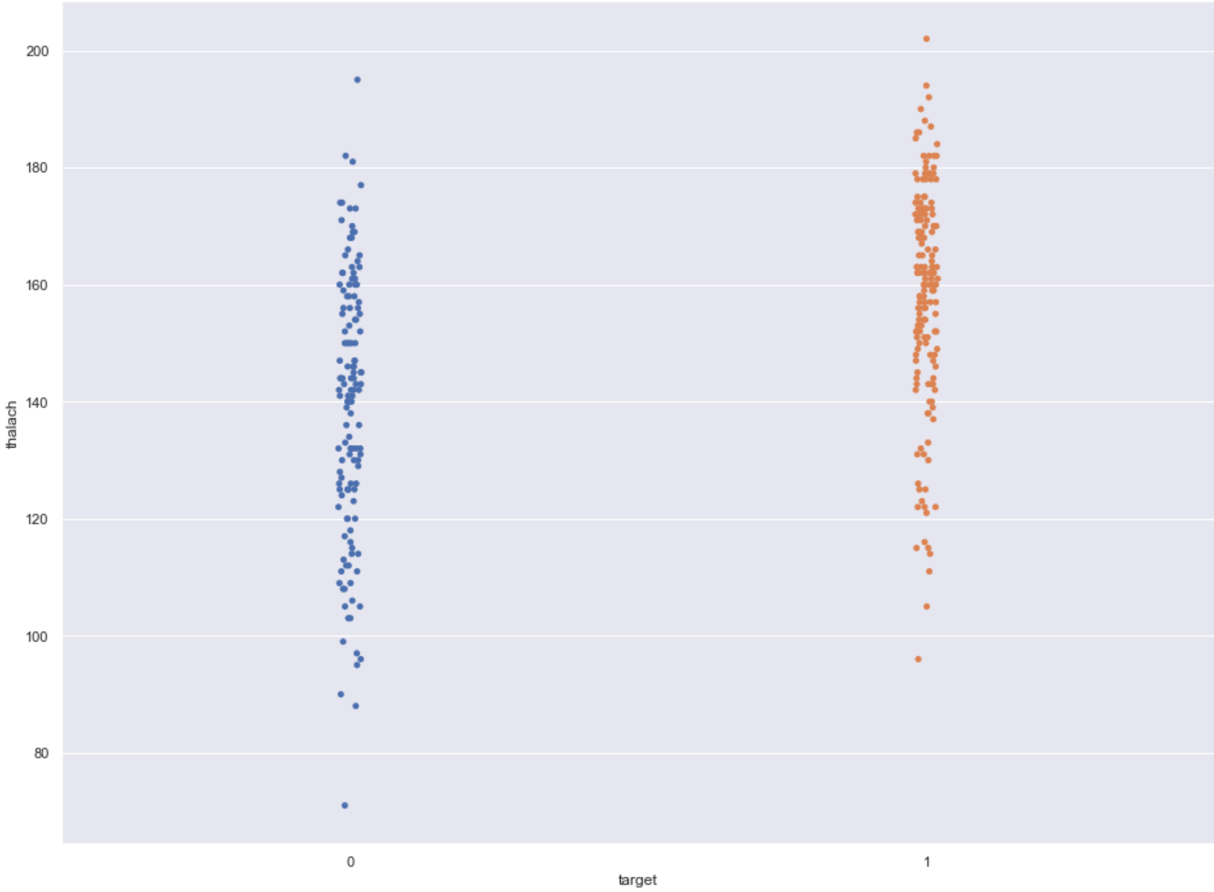
frequency distribution of thalach variable wrt target

In [92]:
```
f, ax = plt.subplots(figsize=(16, 12))
sns.stripplot(x="target", y="thalach", data=hd)
plt.show()
```

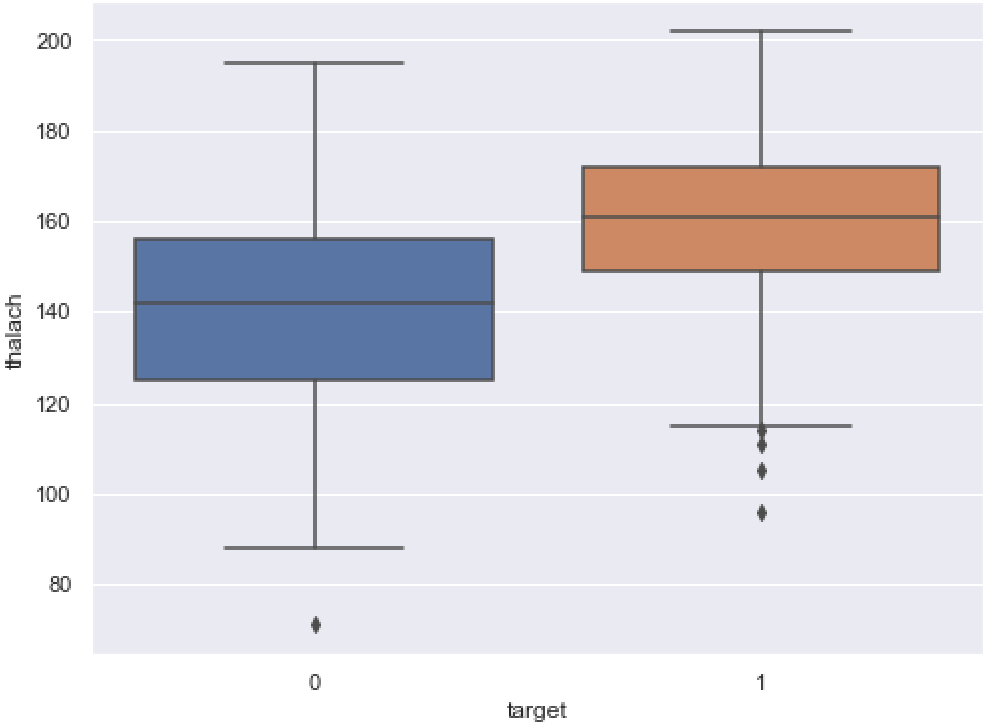


People suffering from heart disease (target = 1) have relatively higher heart rate (thalach) as compared to people who are not suffering from heart disease (target = 0).

In [95]:
```
f, ax = plt.subplots(figsize=(16, 12))
sns.stripplot(x="target", y="thalach", data=hd, jitter = 0.02) #jitter to bring out
plt.show()
```



In [97]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="target", y="thalach", data=hd) #Visualize distribution of thalach var
plt.show()
```



The above boxplot confirms the 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 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.

One 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).

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

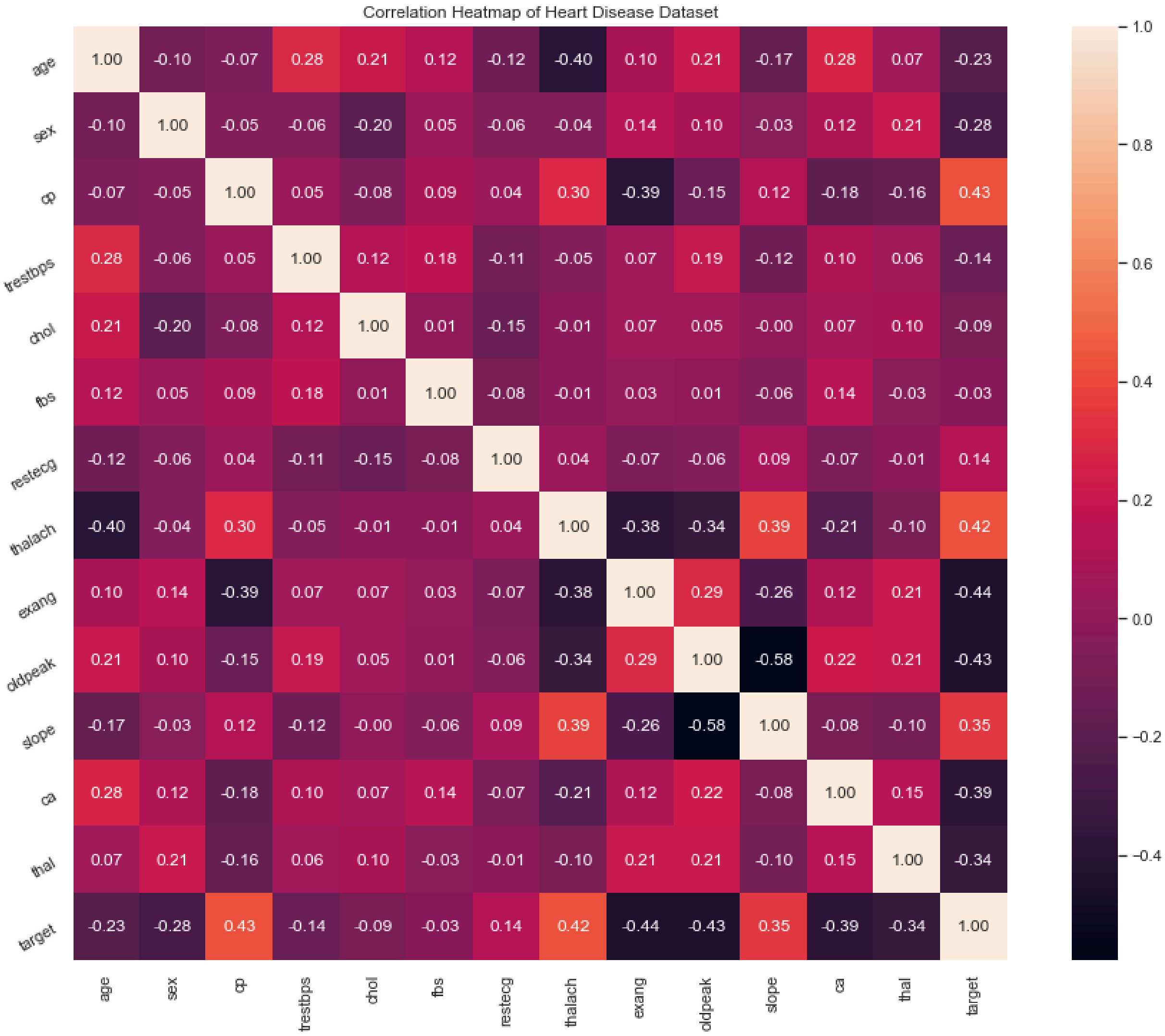
The objective of the multivariate analysis is to discover patterns and relationships in the dataset.

Discover patterns and relationships An important step in EDA is to discover patterns and relationships between variables in the dataset.

Heat map and pair plot is used to discover the patterns and relationships in the dataset.

In [103…

```
plt.figure(figsize=(16,12))
plt.title('Correlation Heatmap of Heart Disease Dataset')
a = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='white')
a.set_xticklabels(a.get_xticklabels(), rotation=90)
a.set_yticklabels(a.get_yticklabels(), rotation=30)
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 [106…
```
num_var = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target' ]
sns.pairplot(hd[num_var], kind='scatter', diag_kind='hist')
plt.show()
```

Comment

- I have defined a variable `num_var` . Here `age` , `trestbps` , `chol`` , ``thalach`` and ``oldpeak` are numerical variables and `target` is the categorical variable.

Analysis of `age` and other variables

In [108…
```
hd['age'].nunique() #number of unique values in `age` variable
```

Out[108… 41

View statistical summary of `age` variable

In [110…
```
hd['age'].describe()
```

Out[110…
```
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
```

Interpretation

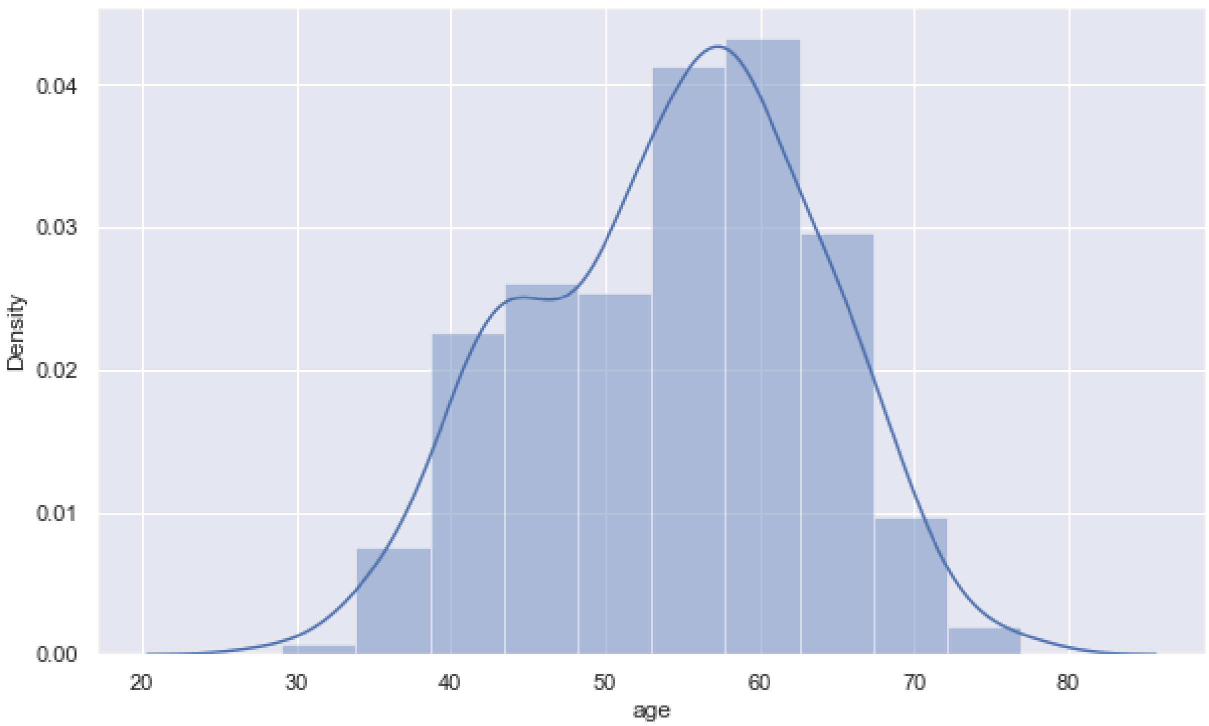
- The mean value of the `age` variable is 54.37 years.
- The minimum and maximum values of `age` are 29 and 77 years.

Plot the distribution of `age` variable

Plot the distribution of `age` variable to view the statistical properties.

In [112...

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



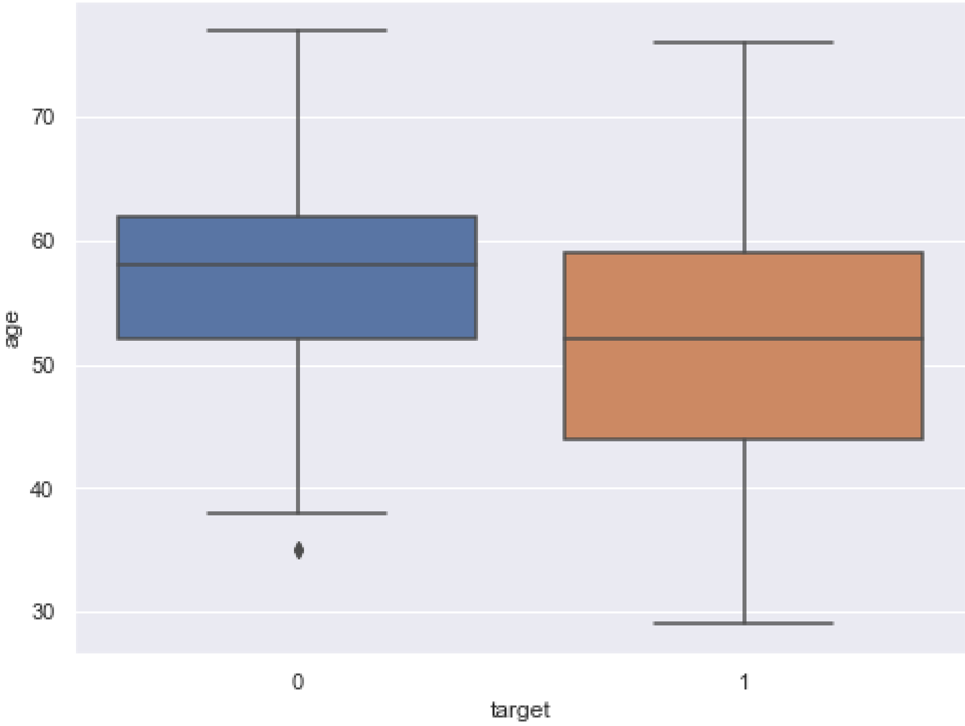
Interpretation

- The `age` variable distribution is approximately normal.

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

In [114...

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



Interpretation

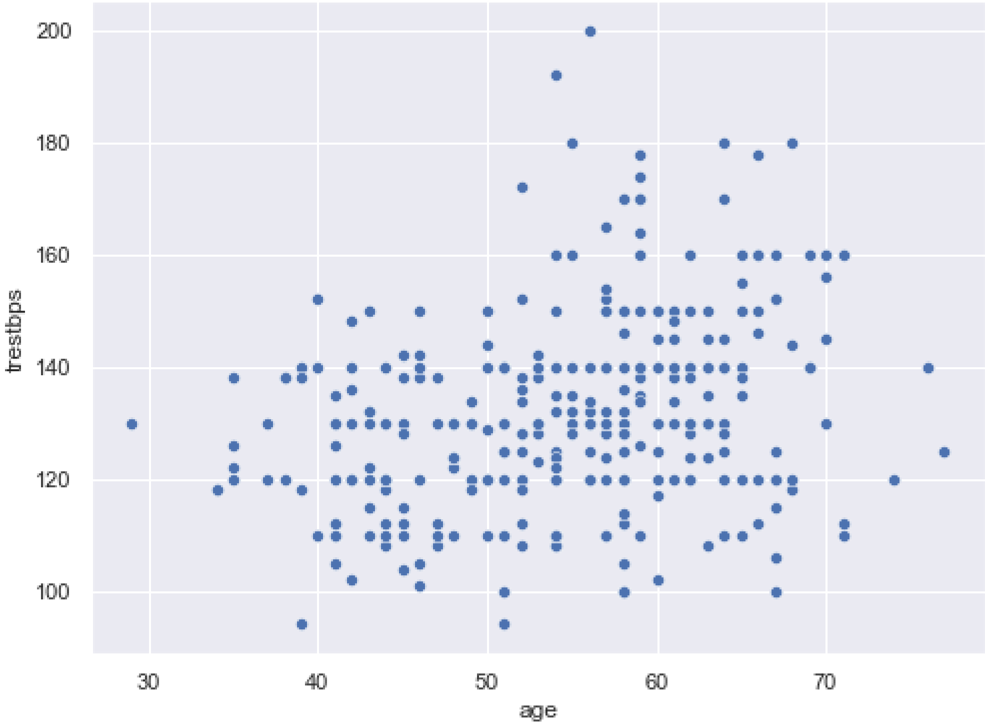
- The above boxplot tells two different things :
    - The mean age of the people who have heart disease is less than the mean age of the people who do not have heart disease.
    - The dispersion or spread of age of the people who have heart disease is greater than the dispersion or spread of age of the people who do not have heart disease.

Analyze `age` and `trestbps` variable

Plot a scatterplot to visualize the relationship between `age` and `trestbps` variable.

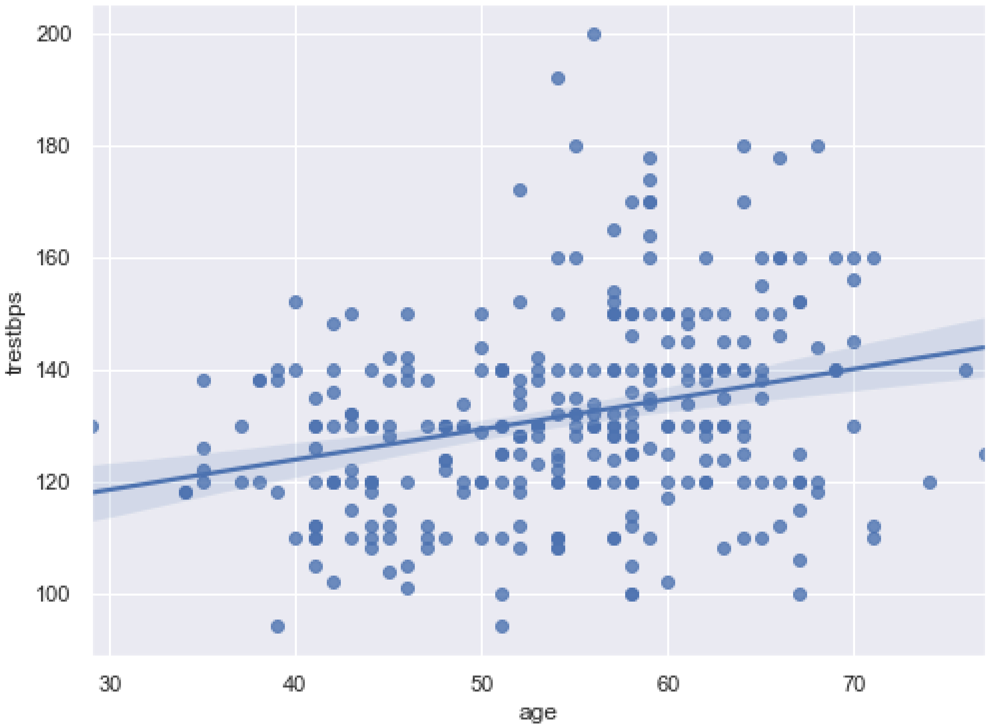
In [118…

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



In [119…

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



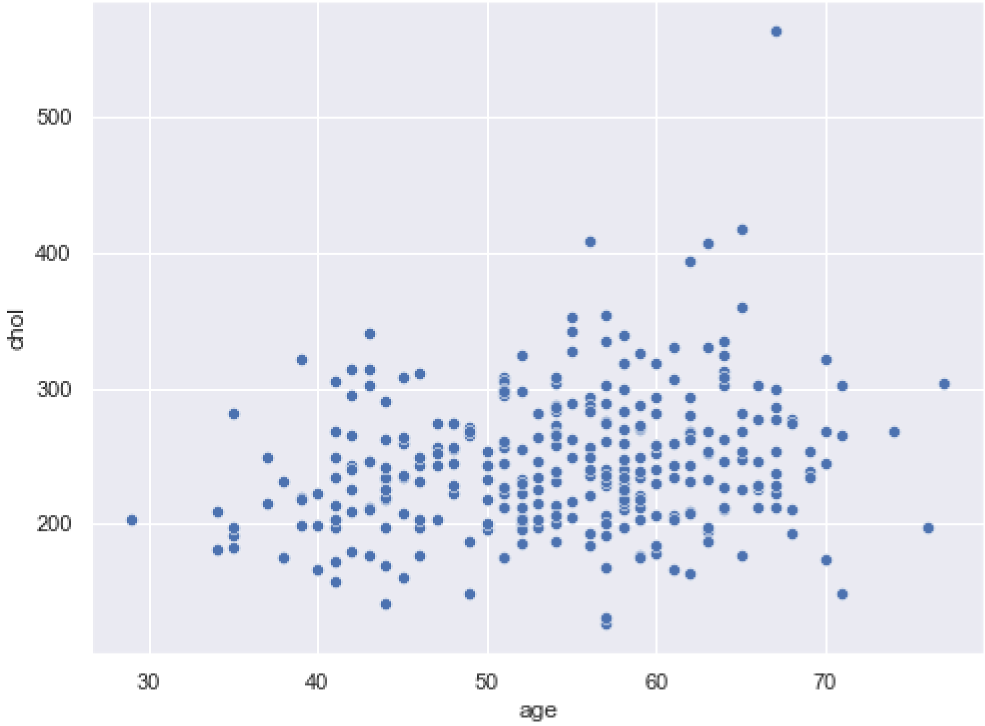
Interpretation

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

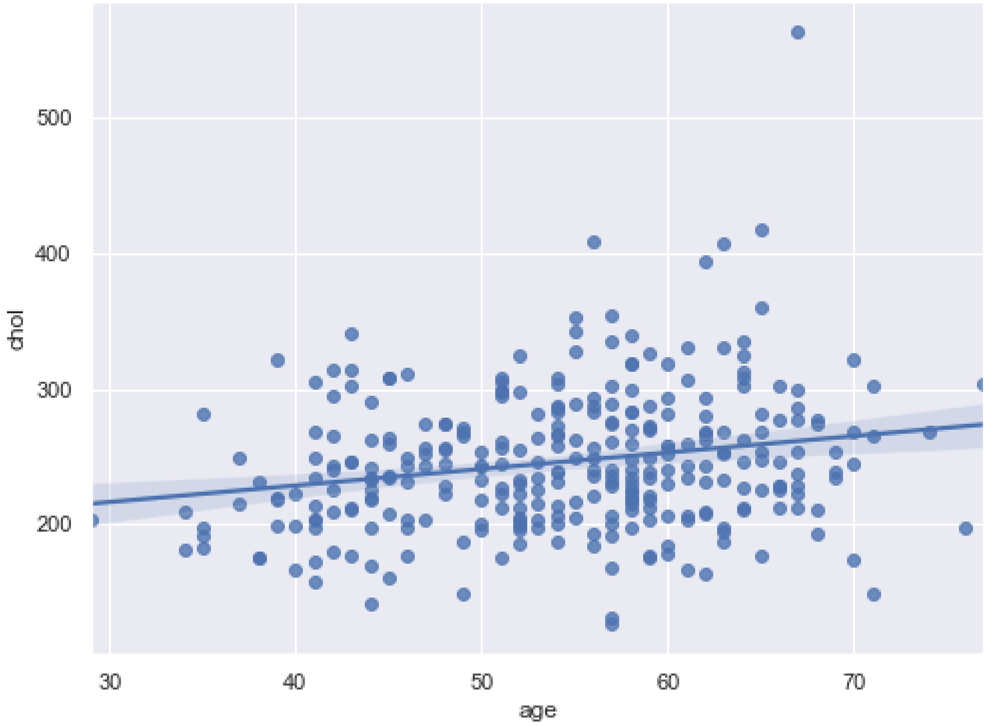
Analyze `age` and `chol` variable

In [121…

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



In [122…
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="chol", data=hd)
plt.show()
```

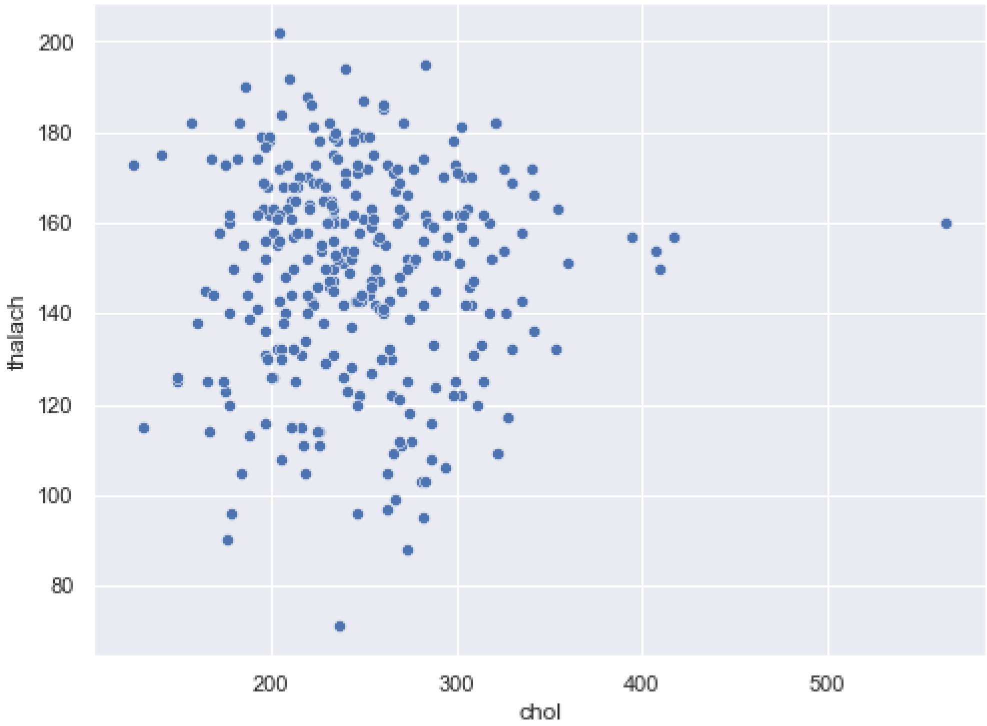


Interpretation

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

Analyze `chol` and `thalach` variable

In [123…
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="chol", y = "thalach", data=hd)
plt.show()
```



In [124...
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="chol", y="thalach", data=hd)
plt.show()
```

Interpretation

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

Dealing with missing values

In [126...
```
hd.isnull().sum() # check for missing values
```

Out[126...
```
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
```

There are no missing values in the dataset.

In [127…
```
assert pd.notnull(hd).all().all() #assert that there are no missing values in the da
```

In [128…
```
assert (hd >= 0).all().all() #assert all values are greater than or equal to 0
```

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

Visualise outliers in the continuous numerical variables : -

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

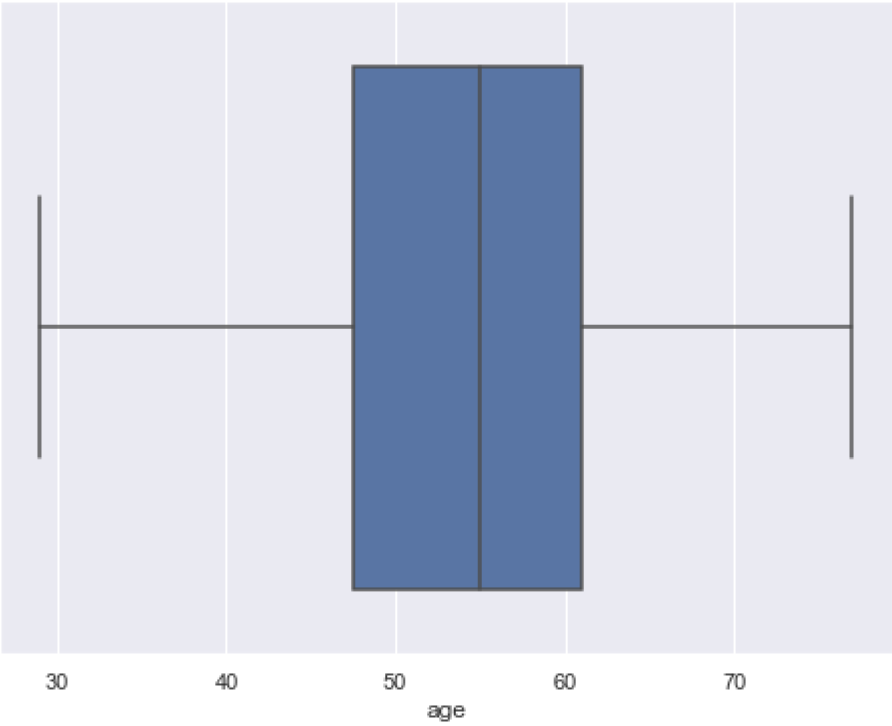
`age` variable

In [130…
```
hd['age'].describe()
```

Out[130…
```
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
```

Box-plot of `age` variable

In [133…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=hd["age"])
plt.show()
```



`trestbps` variable

In [134...

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

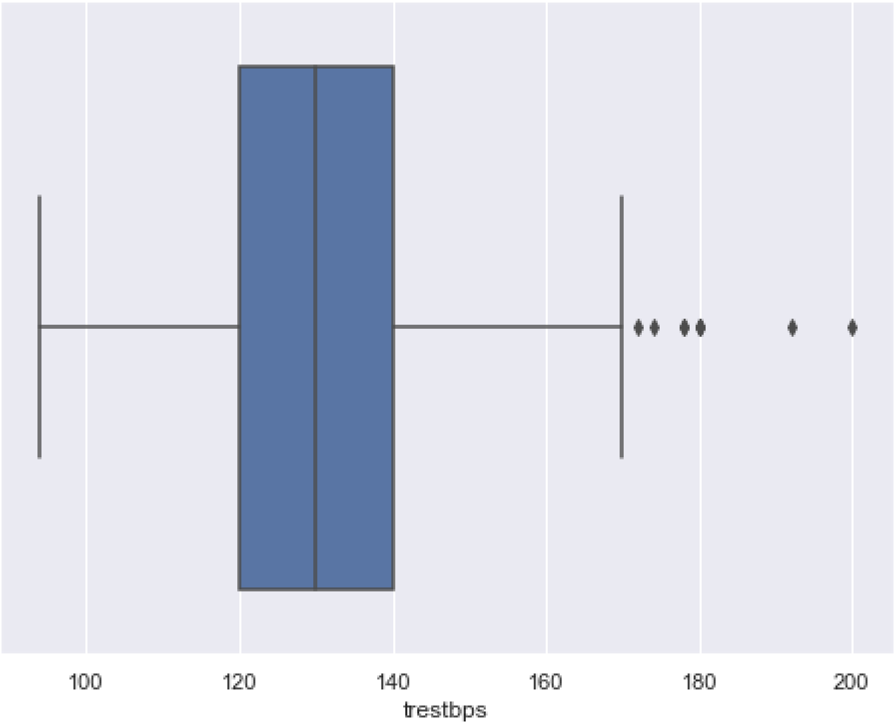
Out[134...

```
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
```

Box-plot of `trestbps` variable

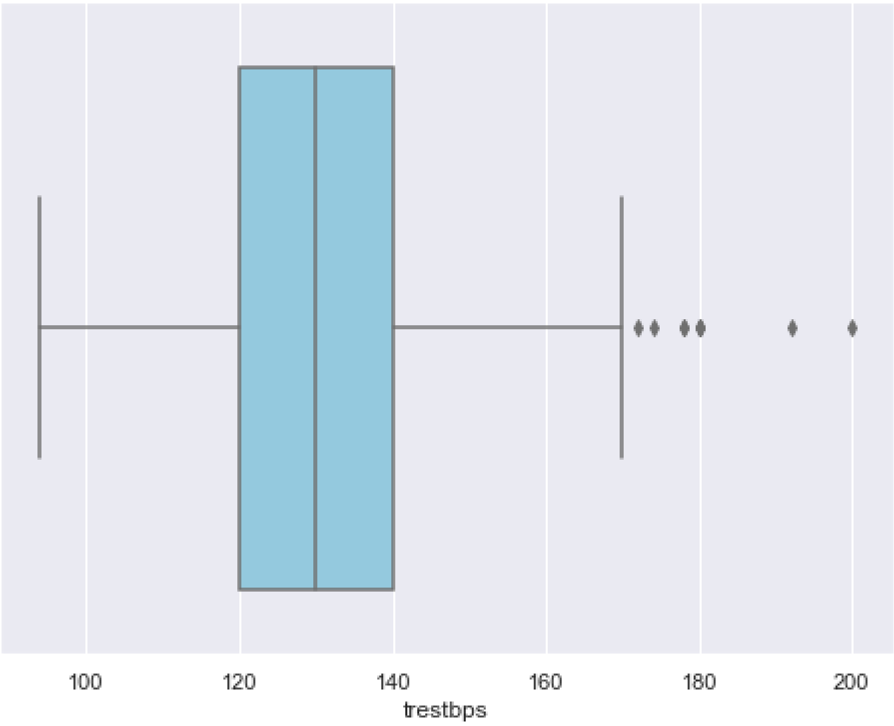
In [135...

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



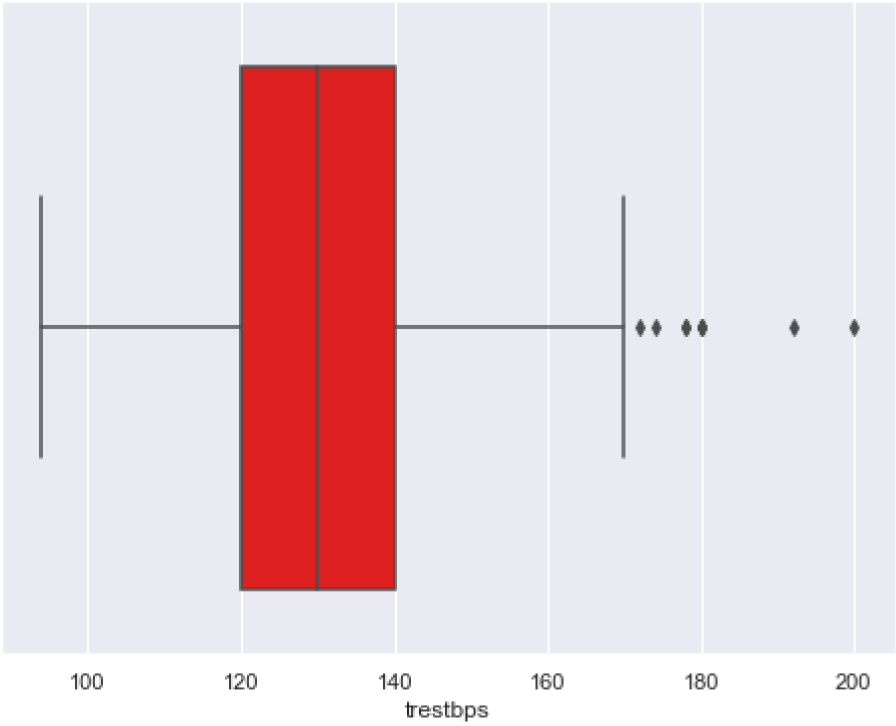
In [136…

```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=hd["trestbps"], color="skyblue")  # You can choose any color you like
plt.show()
```



In [138…

```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=hd["trestbps"], color="red")  # You can choose any color you like
plt.show()
```



`chol` variable

In [141...

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

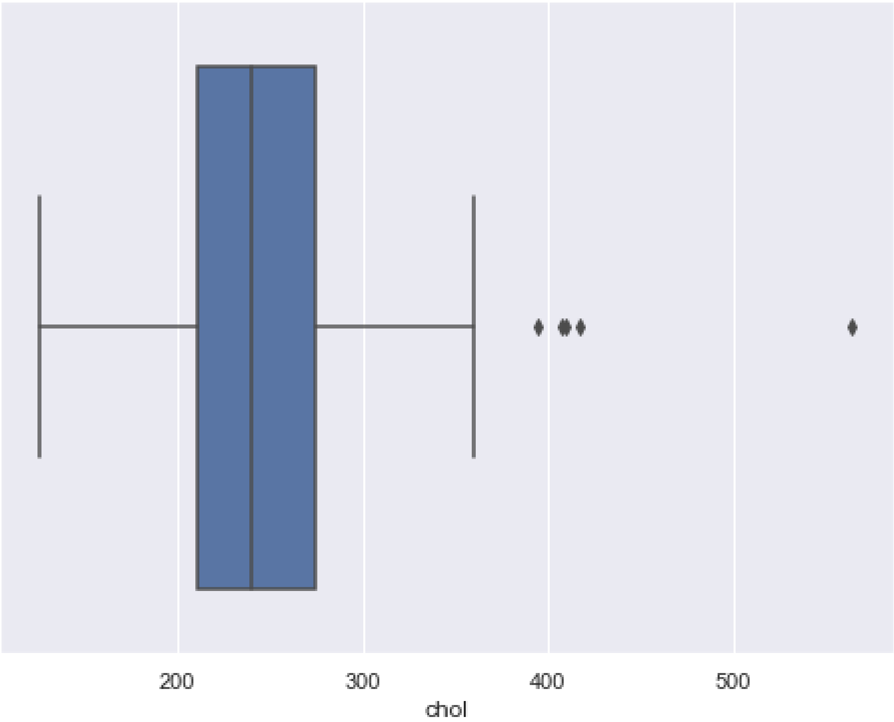
Out[141...

```
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
```

Box-plot of `chol` variable

In [142...

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



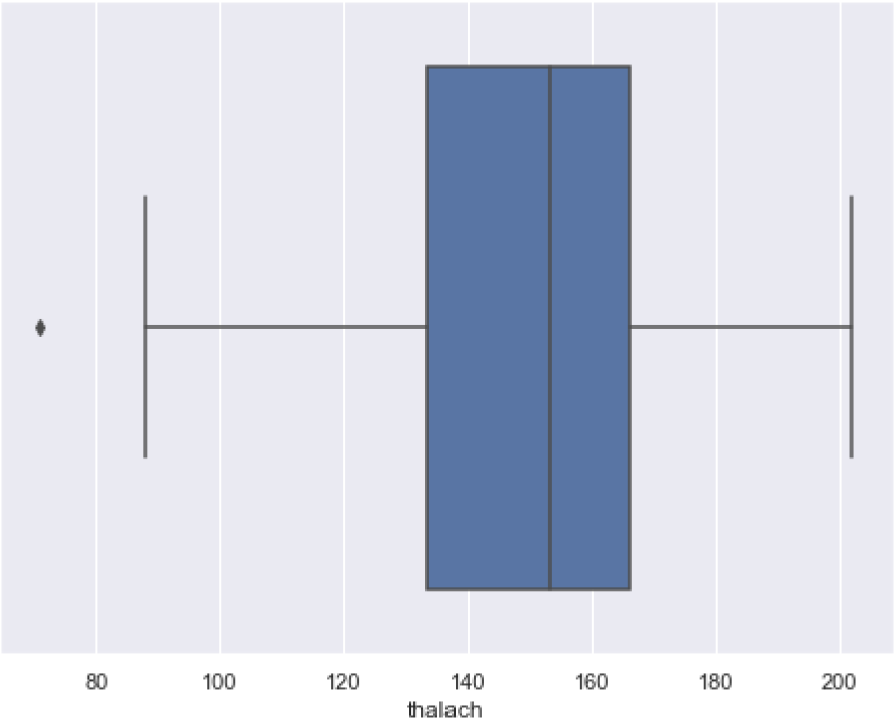
`thalach` variable

In [143…
```
hd['thalach'].describe()
```

Out[143…
```
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
```

Box-plot of `thalach` variable

In [144…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=hd["thalach"])
plt.show()
```



`oldpeak` variable

In [145…

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

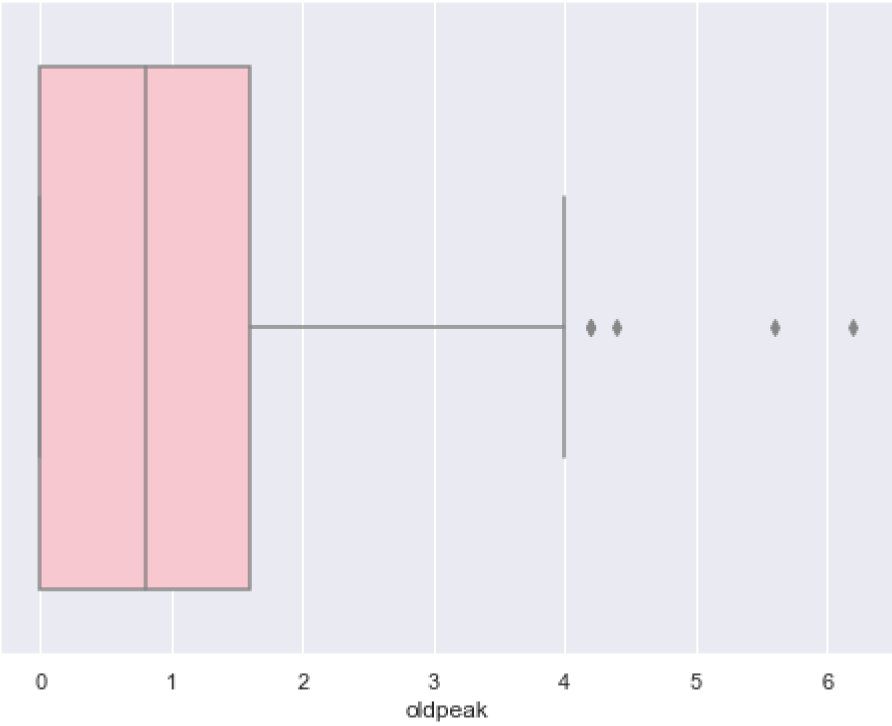
Out[145…

```
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
```

Box-plot of `oldpeak` variable

In [155…

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



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.

```
In [ ]:
```