Heart Patients Analysis

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

sns.set(style="whitegrid")

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

In [3]:
```
df=pd.read_csv(r'C:\Users\Affan\OneDrive\Desktop\FSDS Course NIT\Prakash Sir Sen
```

In [4]:
```
df
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	t
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 [5]:
```
df.head()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2

In [6]:
```
df.tail()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	th
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	

In [7]:
```
df.isna().count()
```

Out[7]:
```
age         303
sex         303
cp          303
trestbps    303
chol        303
fbs         303
restecg     303
thalach     303
exang       303
oldpeak     303
slope       303
ca          303
thal        303
target      303
dtype: int64
```

In [8]:
```
df.isna()
```

Out[8]:

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

303 rows × 14 columns

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

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

In [10]:
```
df.shape
```

Out[10]:
```
(303, 14)
```

In [12]:
```
df.describe()
```

Out[12]:

	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 [13]:
```
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
```

----Description---- - from the above info we get to know thar are 14 total columns out of which 1 col is of float dtype and rest are int dtype

Univariate Analysis

In [14]:
```
df['target'].unique()
```

Out[14]:
```
array([1, 0], dtype=int64)
```

In [16]:
```
df['target'].nunique()
```

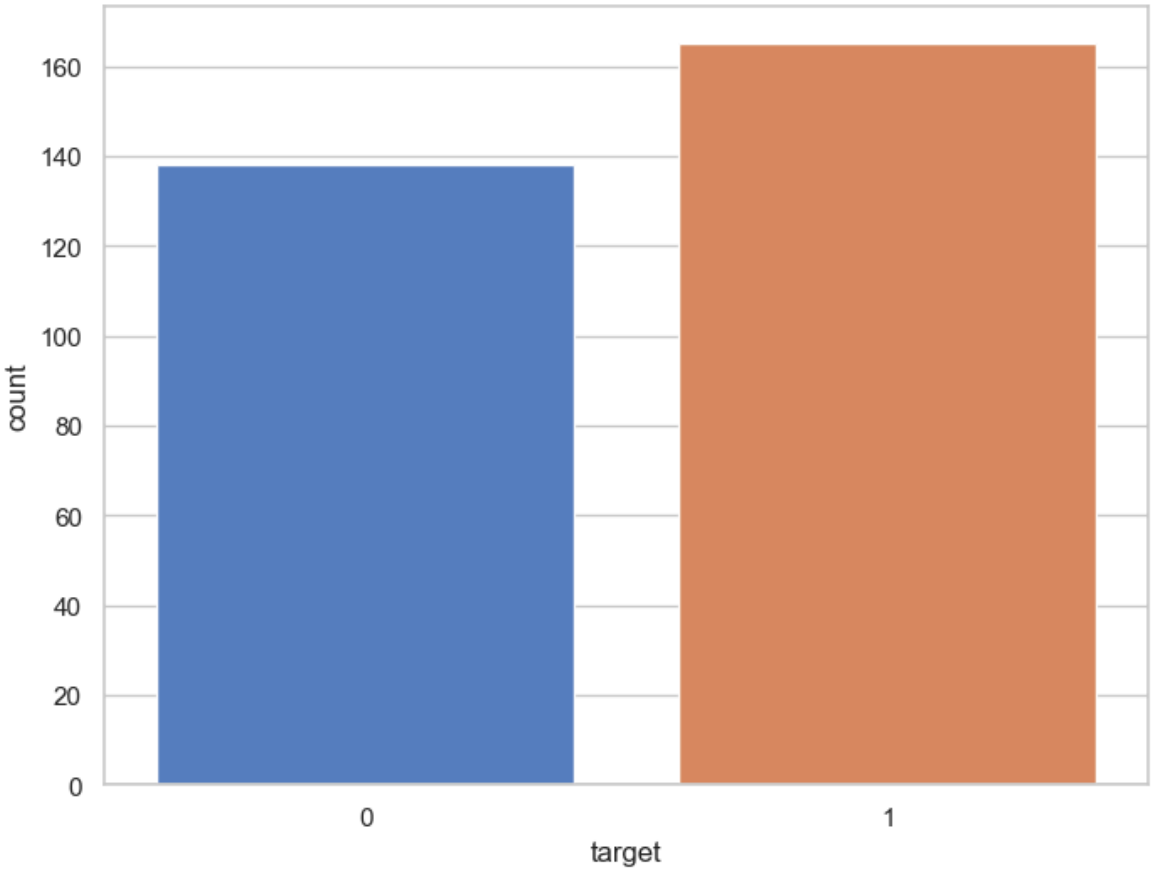
Out[16]:
```
2
```

In [17]:
```
df['target'].value_counts()
```

Out[17]:
```
target
1    165
0    138
Name: count, dtype: int64
```

In [18]:
```
#here 1 stands for presence of heart disease and 0 for non heart patience
```

In [23]:
```
#visualiziton of target variable
f,ax=plt.subplots(figsize=(8,6))
sns.countplot(x=df['target'],palette='muted')
plt.show()
```



----the above graph shows patience with and without heart disease---- 165 with heart 138 without heart---freq distr of target var with sex---

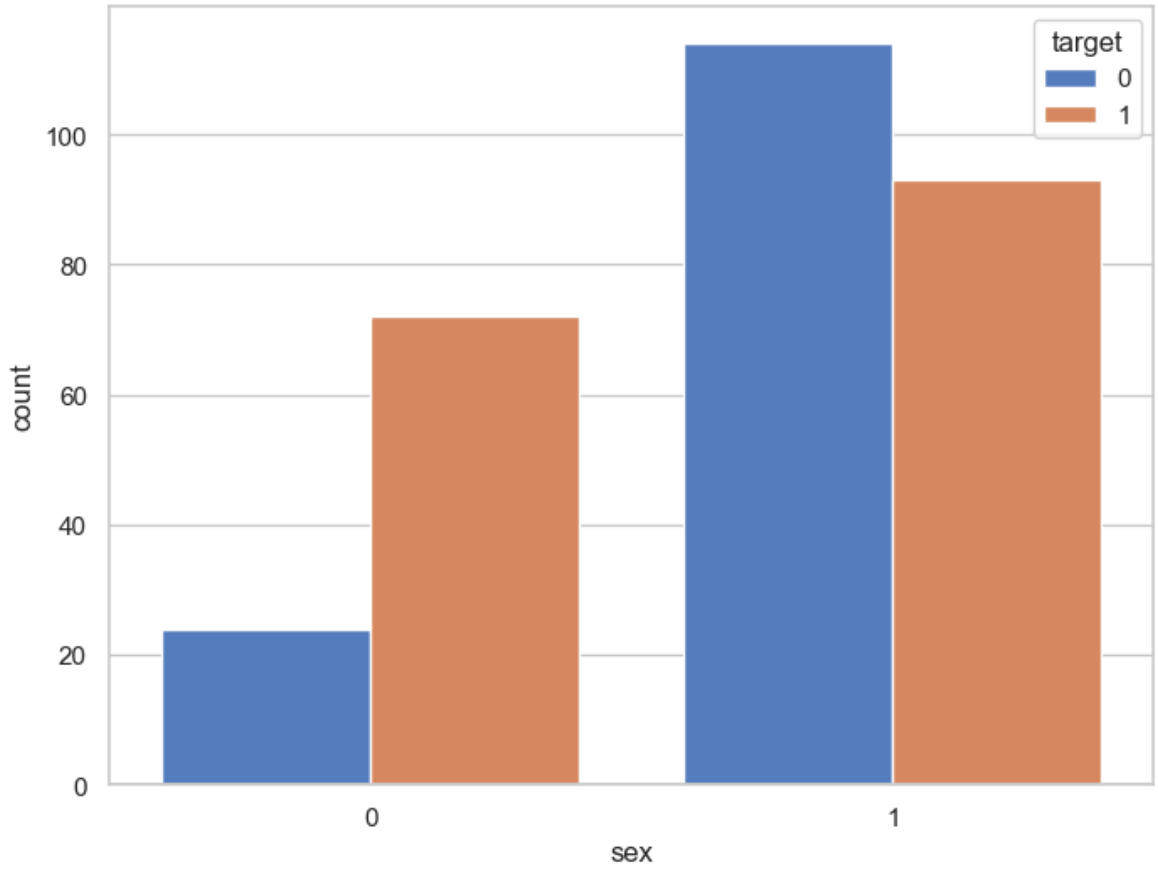
In [29]:
```
df.groupby('sex')['target'].value_counts()
```

Out[29]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

Comment

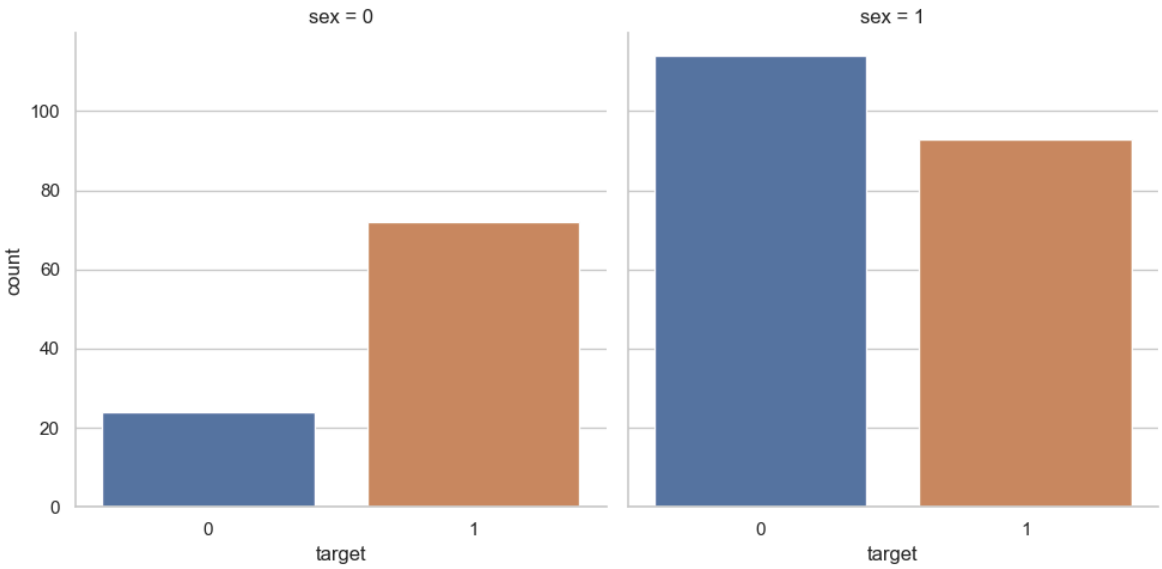
- `sex` variable contains two integer values 1 and 0 : (1 = male; 0 = female).
- `target` variable also contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)
- So, out of 96 females - 72 have heart disease and 24 do not have heart disease.
- Similarly, out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

In [31]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.countplot(data=df,x='sex',hue='target',palette='muted')
plt.show()
```



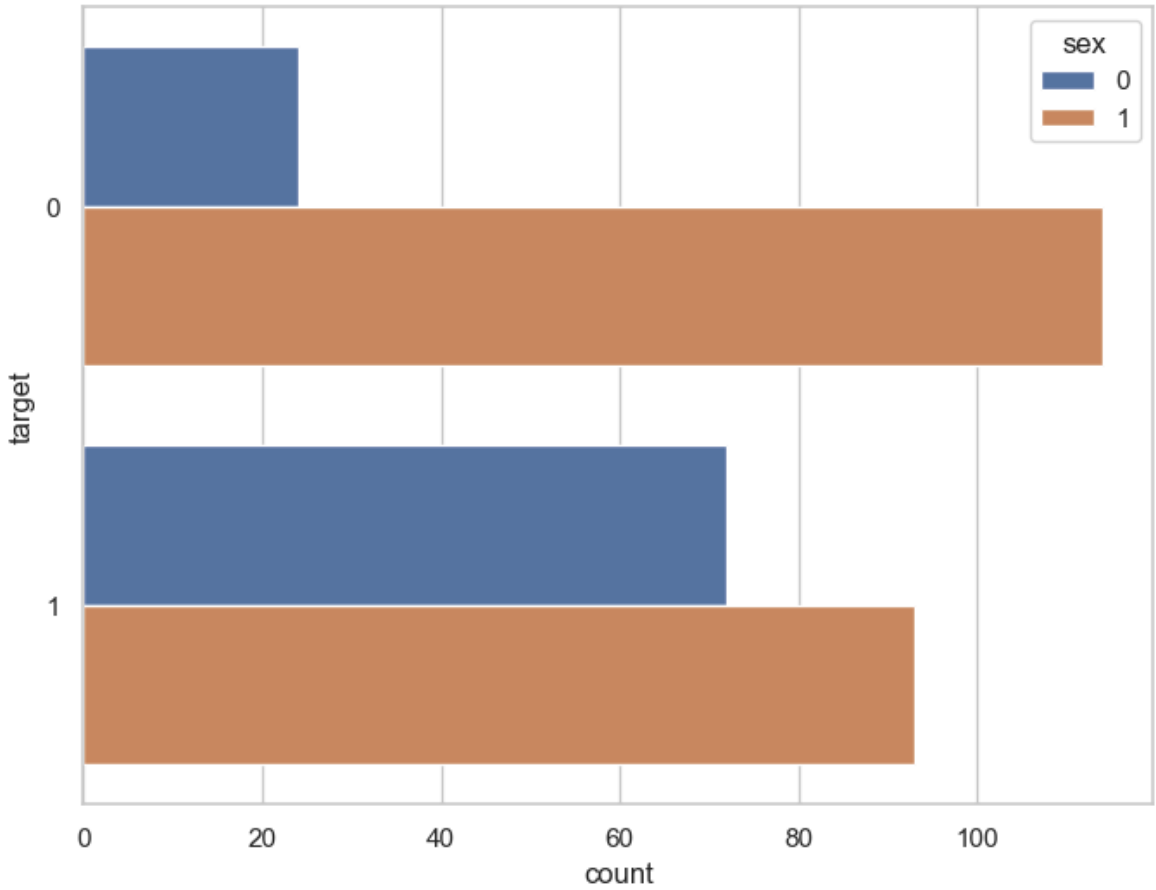
- Out of 96 females - 72 have heart disease and 24 do not have heart disease. - Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

In [36]:
```
sns.catplot(data=df,x='target',col='sex',kind='count',height=5,aspect=1,palette=
plt.show()
```

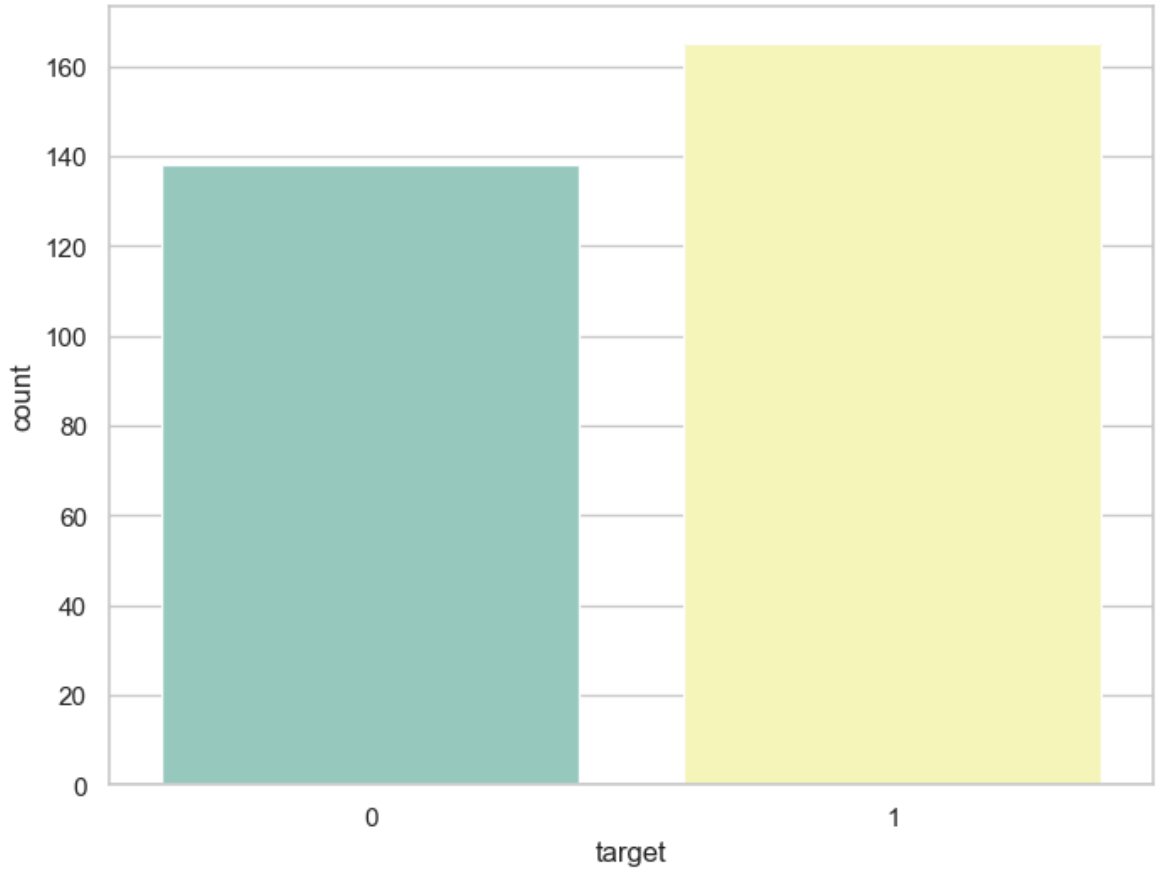


---the above graphs shows the target var for both sex in diff col graph---

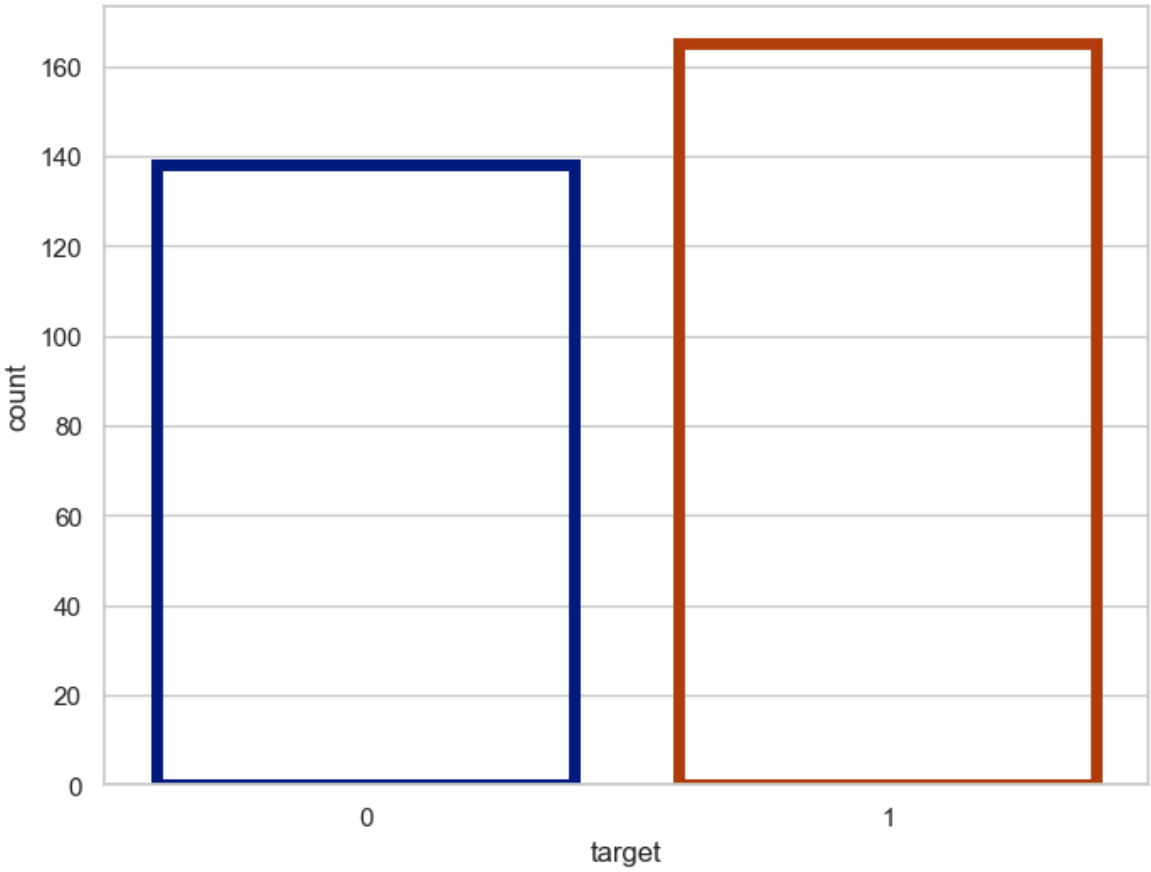
In [39]:
```
#the above graph but in horizontal
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(y='target',data=df,hue='sex')
plt.show()
```



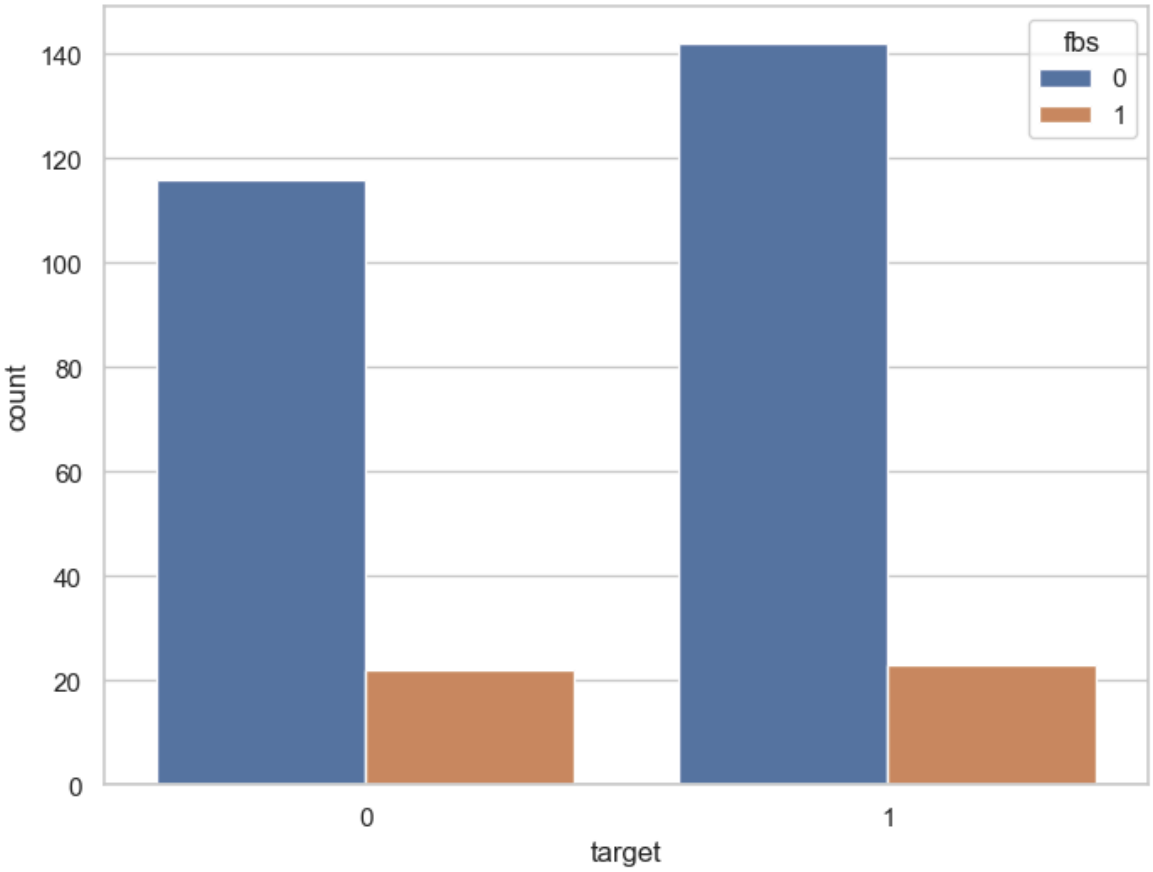
In [40]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, palette="Set3")
plt.show()
```



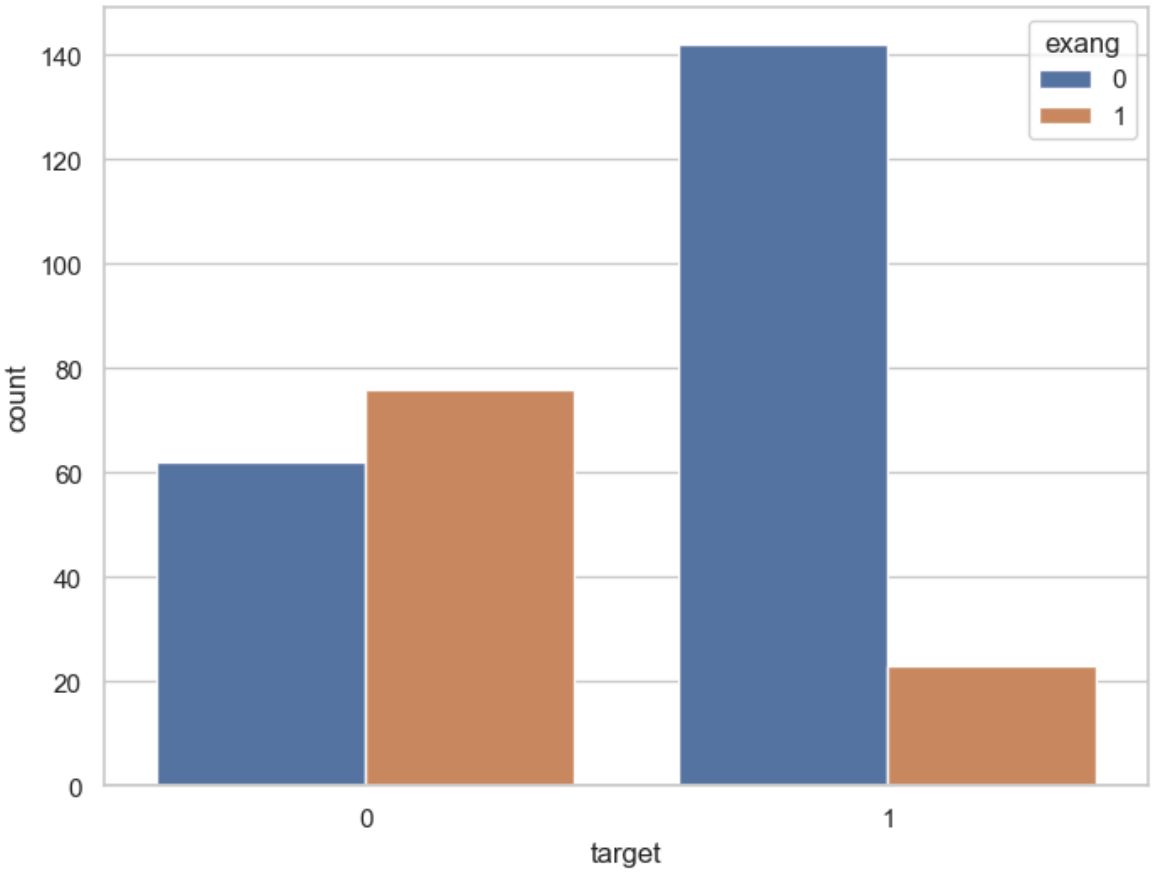
In [41]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, facecolor=(0,0,0,0),linewidth=5,edgecolo
plt.show()
```



In [43]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", hue="fbs", data=df) #fbs ia col which is(fasting
plt.show()
```



In [44]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", hue="exang", data=df)
plt.show()
```



In [45]:
```
correlation=df.corr()
```

In [47]:
```
#checking how each col correlates with target var
correlation['target'].sort_values(ascending=False)
```

Out[47]:
```
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

'cp' is a col or var stands for 'chest pain type'

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

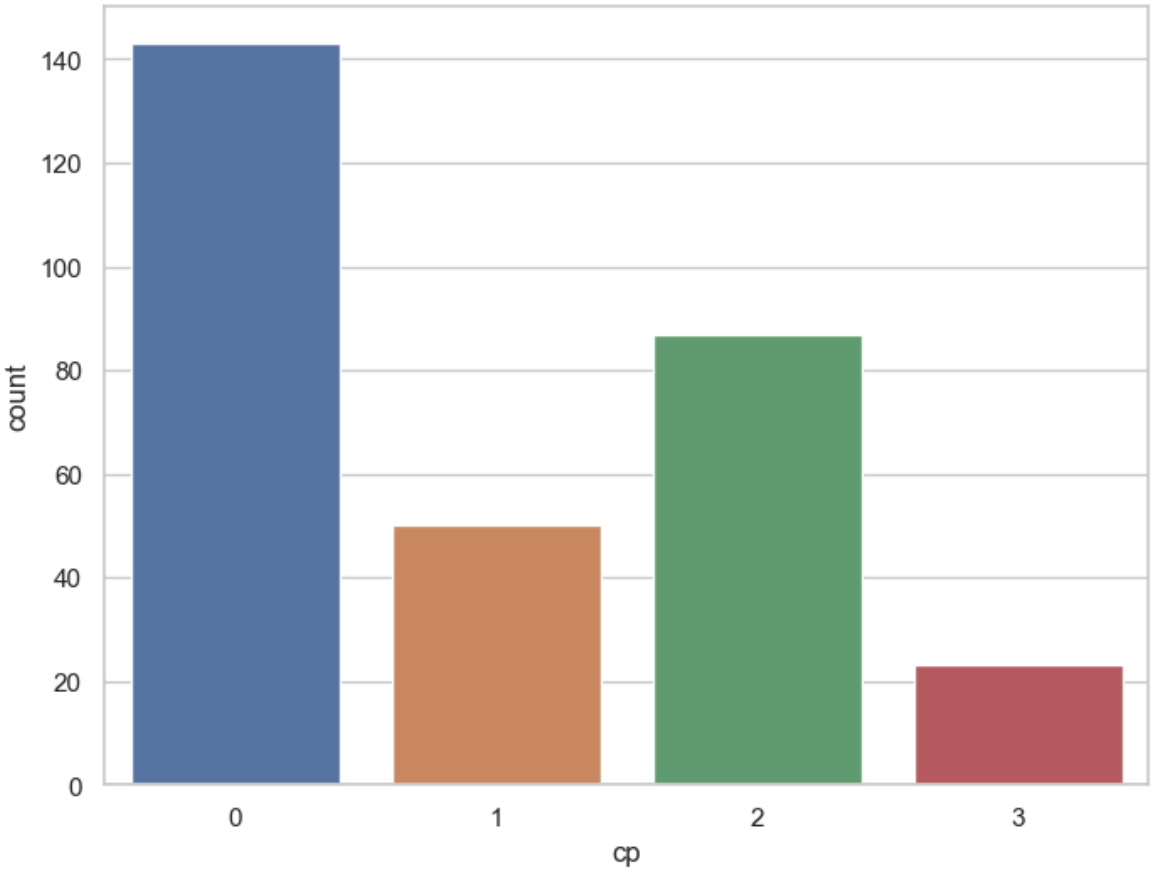
Out[49]:
```
4
```

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

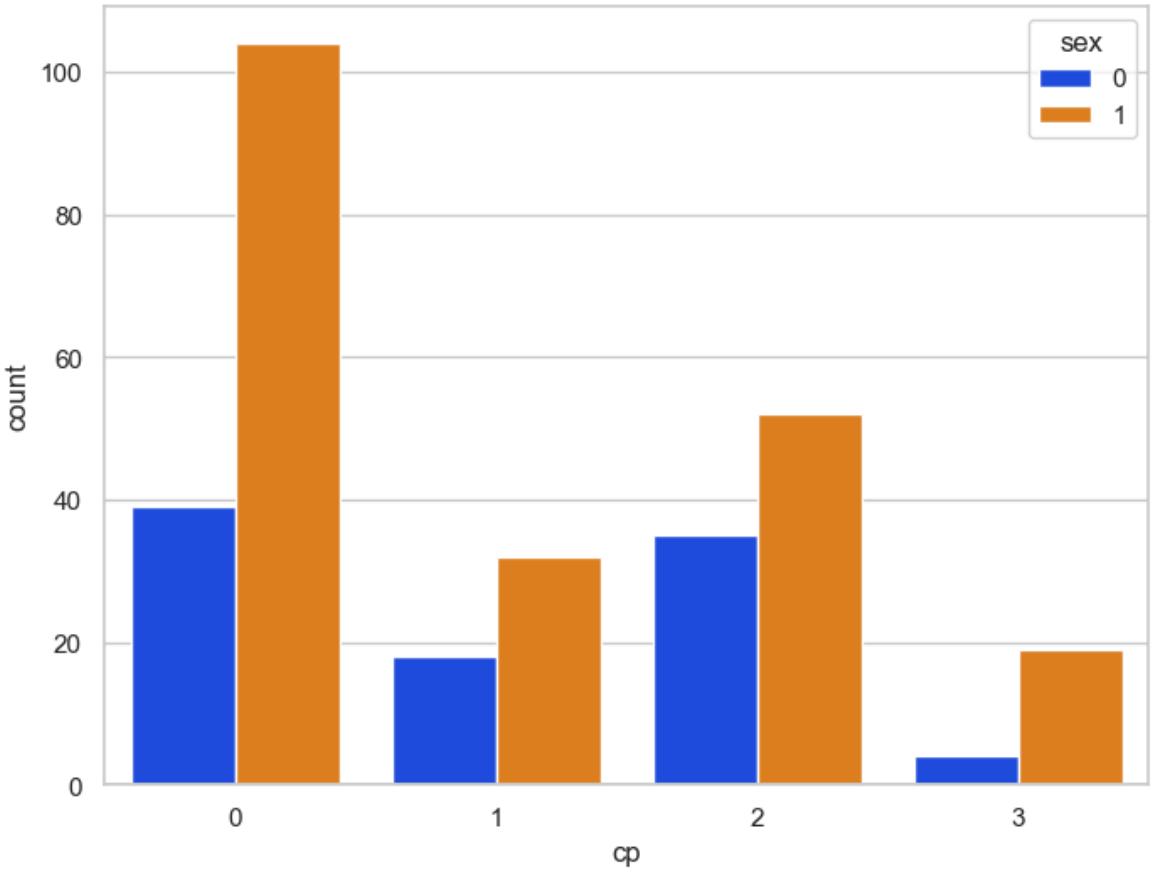
Out[50]:
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

it can be seen that cp is categorical and contaains 4 types of values 0,1,2,3

In [52]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.countplot(x='cp',data=df,palette='deep')
plt.show()
```



In [54]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.countplot(x='cp',hue='sex',data=df,palette='bright')
plt.show()
```

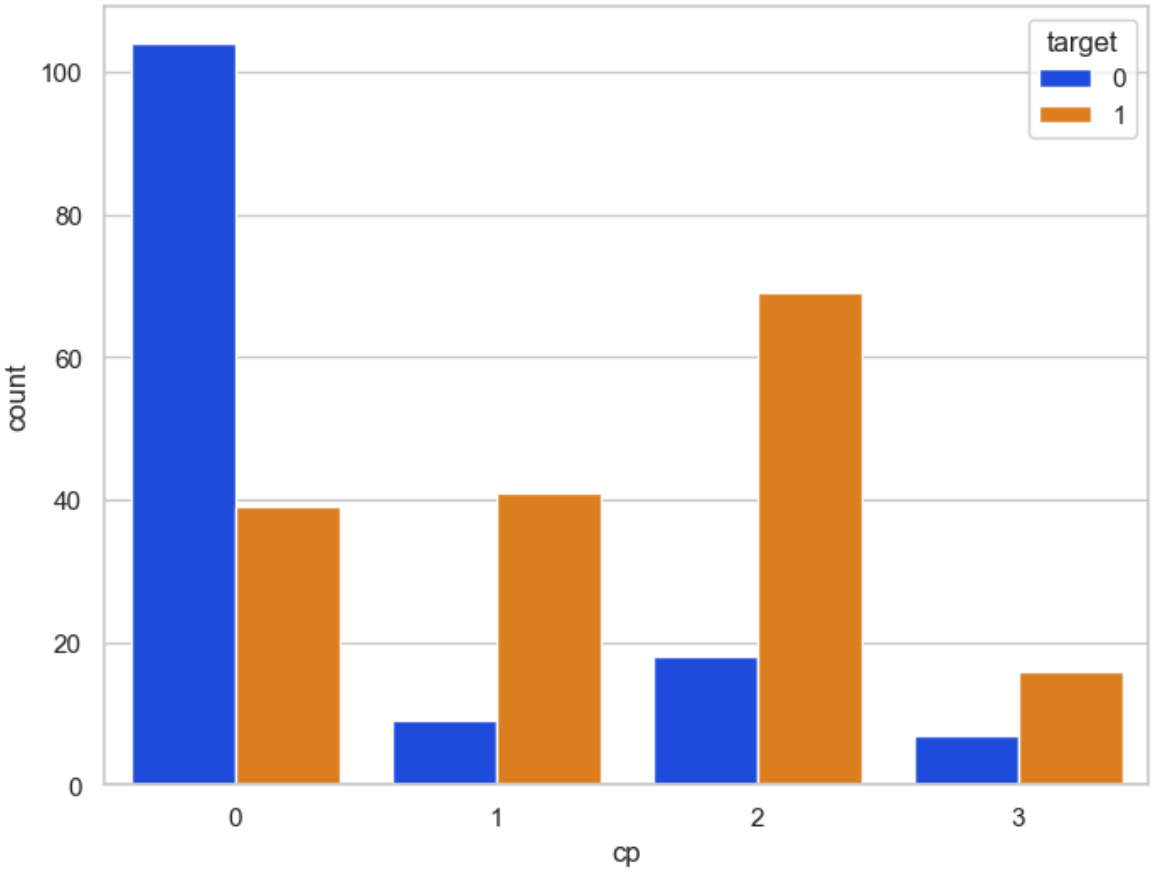


In [55]:
```
df.groupby('cp')['target'].value_counts()
```

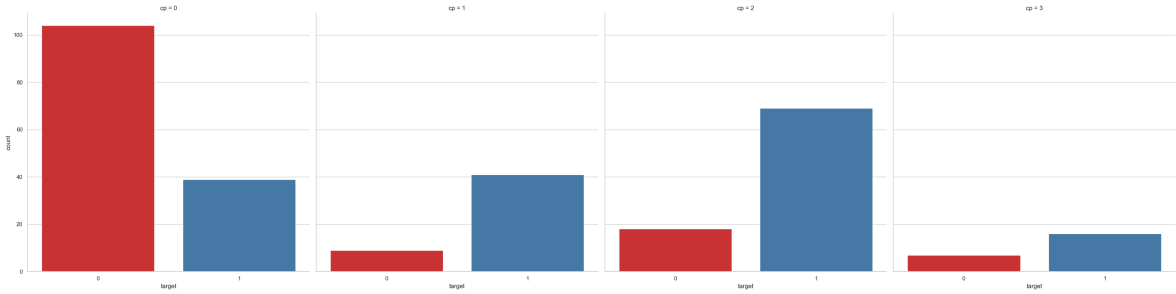
Out[55]:
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: count, dtype: int64
```

In [56]:
```
###comment
#the above we have grouped target var based on each cp
```

In [57]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.countplot(x='cp',hue='target',data=df,palette='bright')
plt.show()
```



In [59]:
```
ax = sns.catplot(x='target',col='cp',data=df,kind='count',height=8,aspect=1,pale
plt.show()
```



analysis of taget and thalach var --thalach stands for min heart disease rate achieve --belwo we check no. of unquiue val in thalach

In [60]:
```
df['thalach'].unique()
```

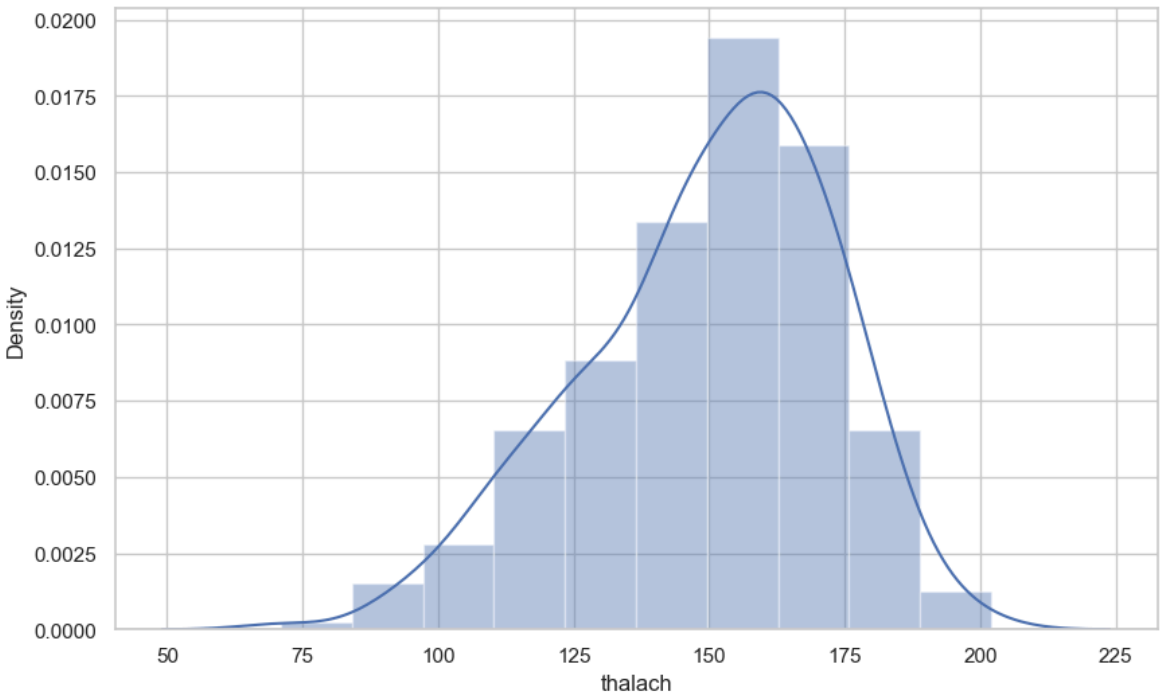
Out[60]:
```
array([150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 171,
       144, 158, 114, 151, 161, 179, 137, 157, 123, 152, 168, 140, 188,
       125, 170, 165, 142, 180, 143, 182, 156, 115, 149, 146, 175, 186,
       185, 159, 130, 190, 132, 147, 154, 202, 166, 164, 184, 122, 169,
       138, 111, 145, 194, 131, 133, 155, 167, 192, 121,  96, 126, 105,
       181, 116, 108, 129, 120, 112, 128, 109, 113,  99, 177, 141, 136,
        97, 127, 103, 124,  88, 195, 106,  95, 117,  71, 118, 134,  90],
      dtype=int64)
```

In [61]:
```
df['thalach'].nunique()
```

Out[61]:
```
91
```

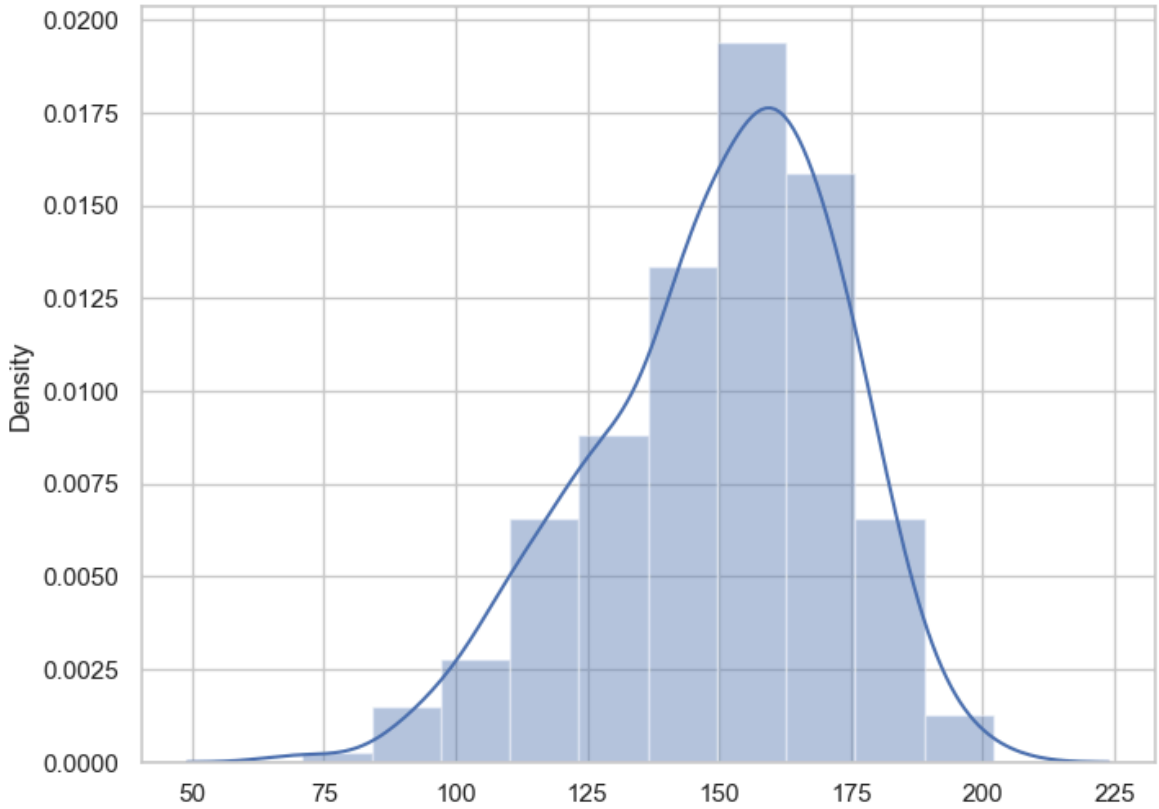
In [62]:
```
#visualization with 'thalach' var
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
```

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

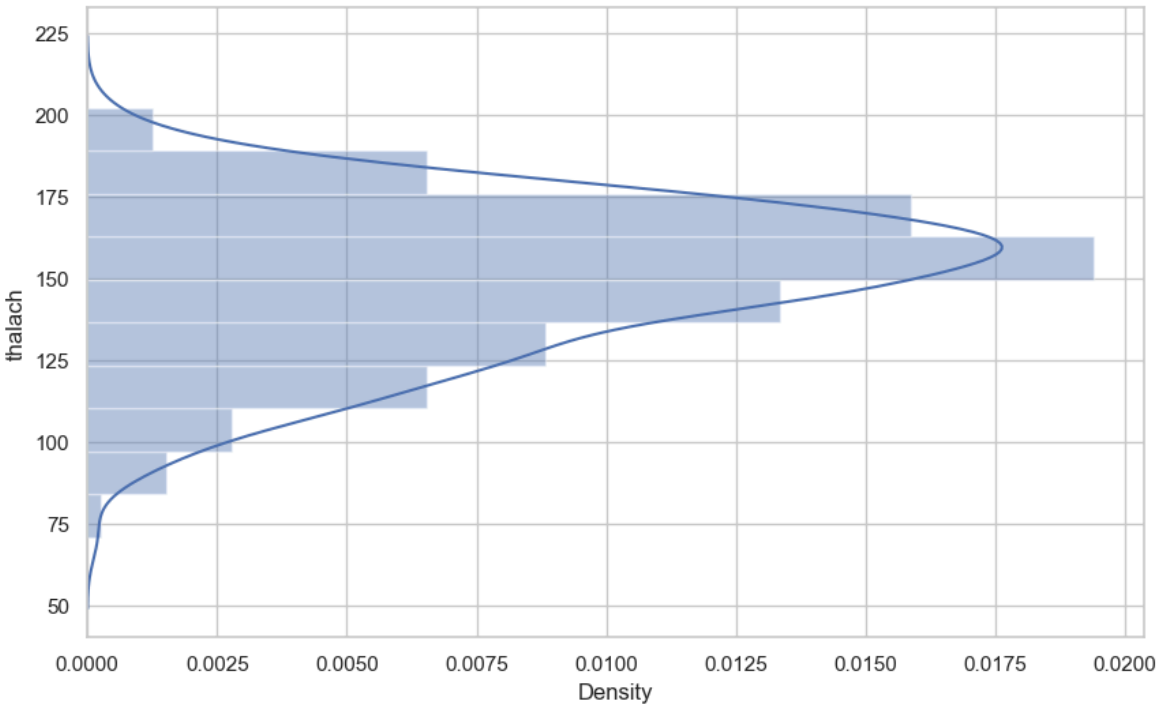


---slightly -ve skewed is thalach var---

In [63]:
```
f,ax=plt.subplots(figsize=(8,6))
x=pd.Series(x,name='thalach var')
sns.distplot(x=df['thalach'],bins=10)
plt.show()
```

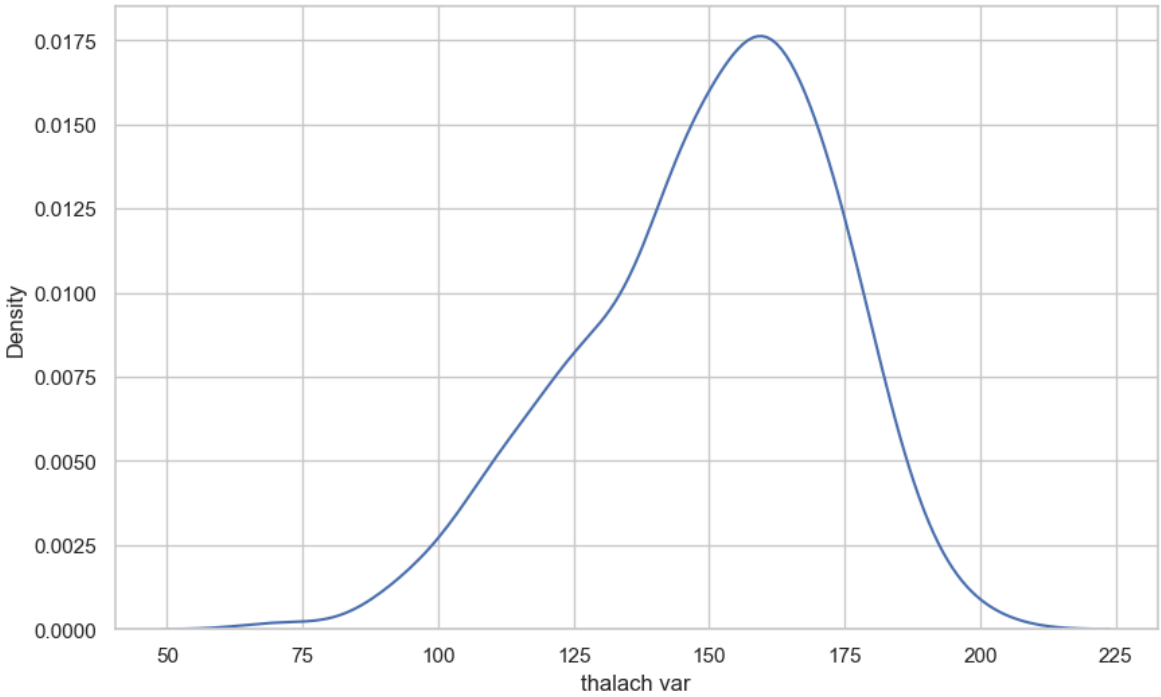


In [64]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.distplot(x, bins=10, vertical=True)
plt.show()
```

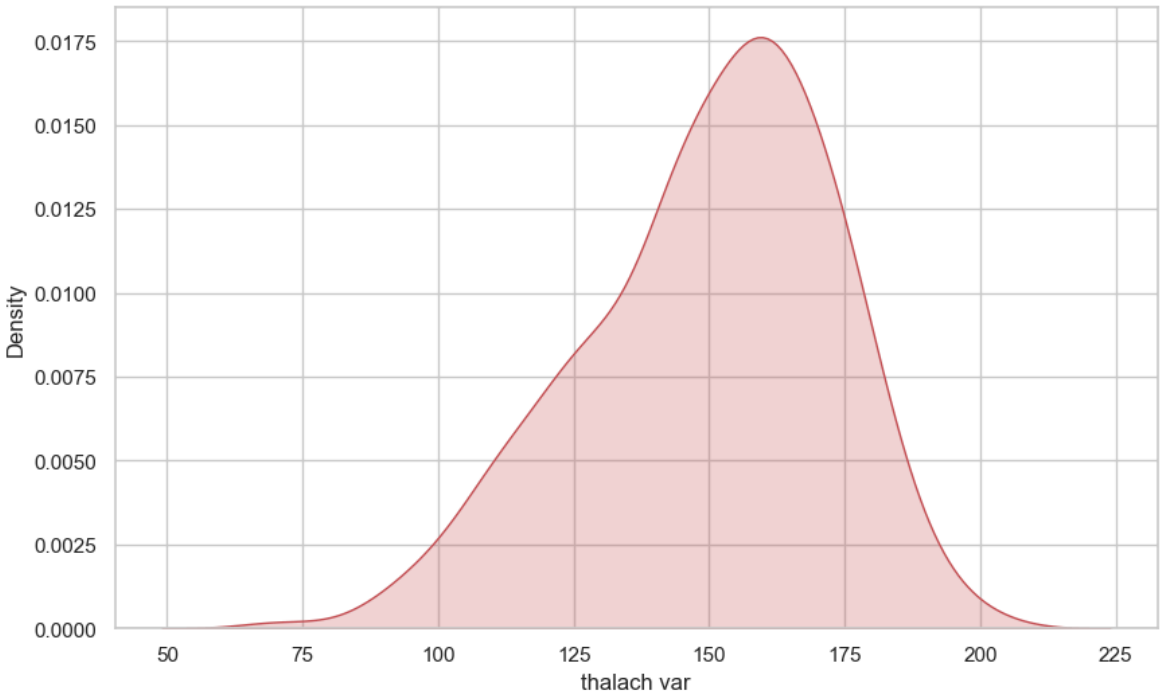


kde plot

In [72]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x=pd.Series(x,name='thalach var')
ax = sns.kdeplot(x)
plt.show()
```

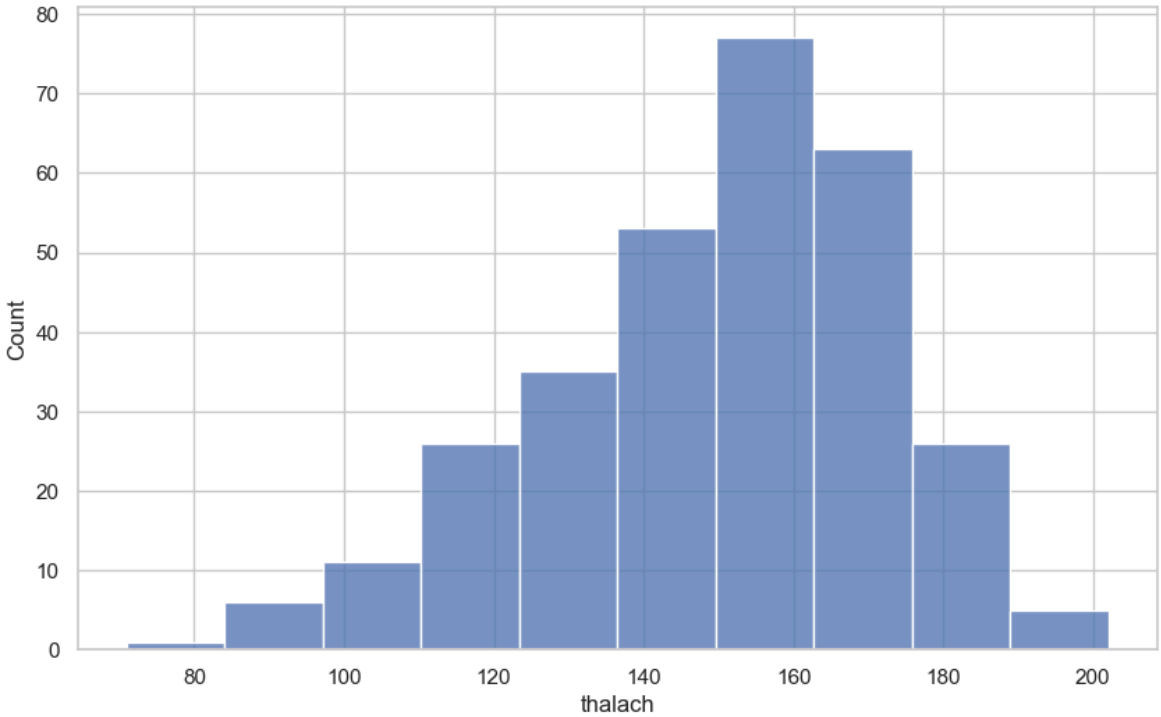


In [69]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
x=pd.Series(x,name='thalach var')
ax = sns.kdeplot(x,shade=True,color='r')
plt.show()
```



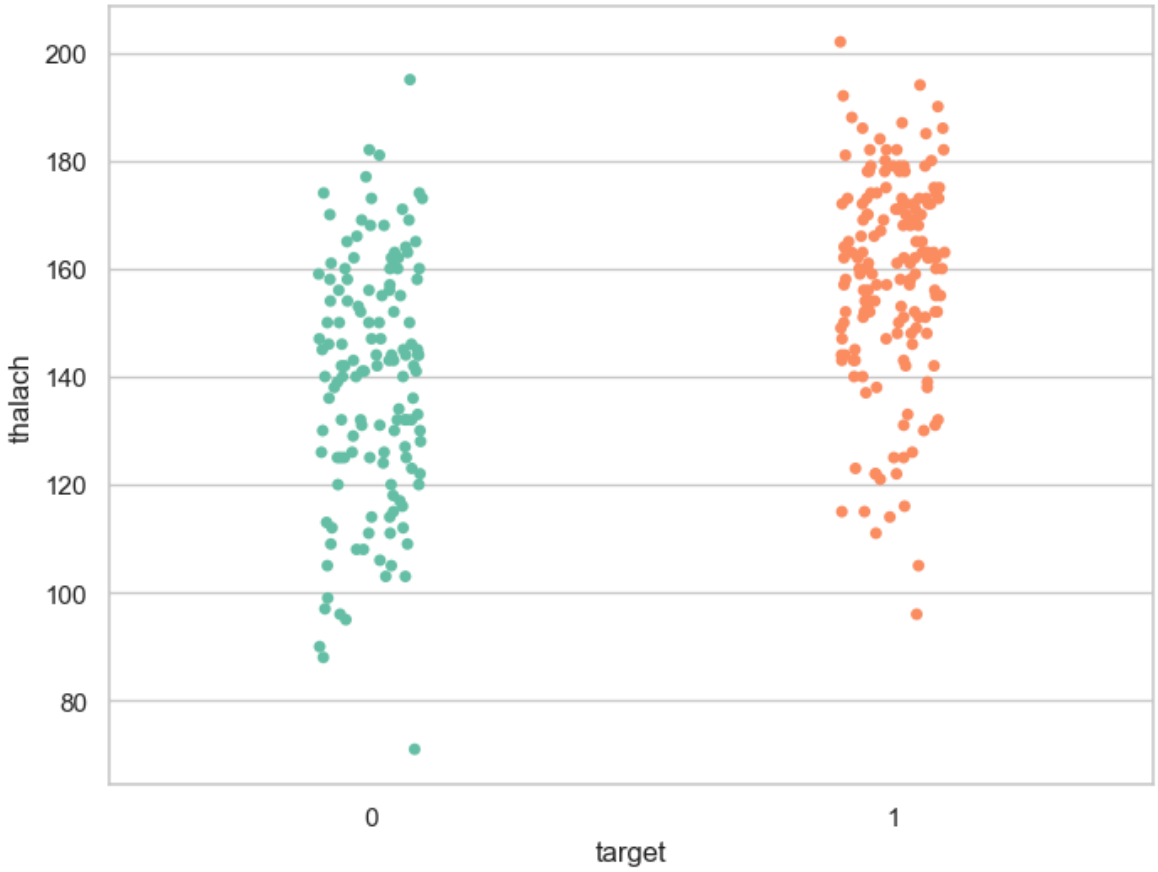
Histogram

In [77]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['thalach']
ax = sns.histplot(x,kde=False,bins=10)
plt.show()
```

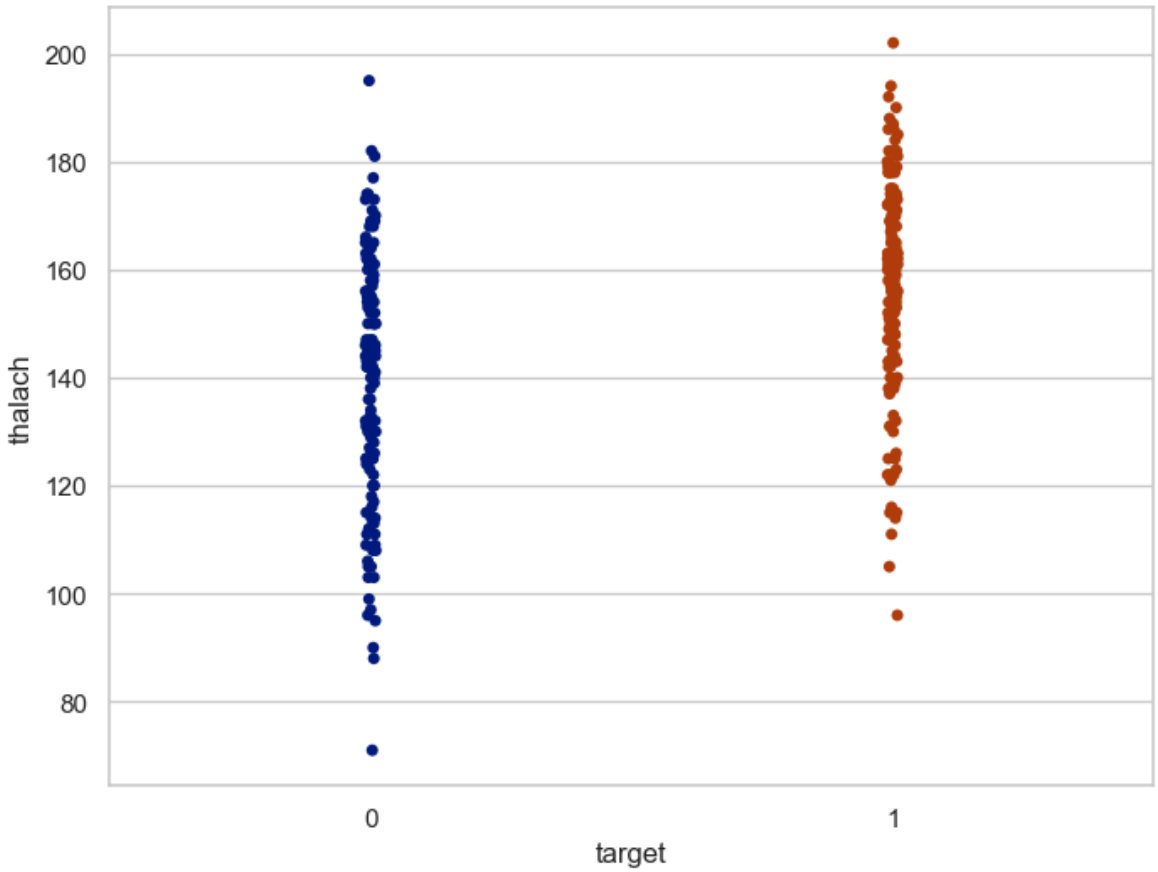


----visulize freq distr of thalach with target----

In [79]:
```
f,ax=plt.subplots(figsize=(8,6))
sns.stripplot(x='target',y='thalach',data=df,palette='Set2')
plt.show()
```

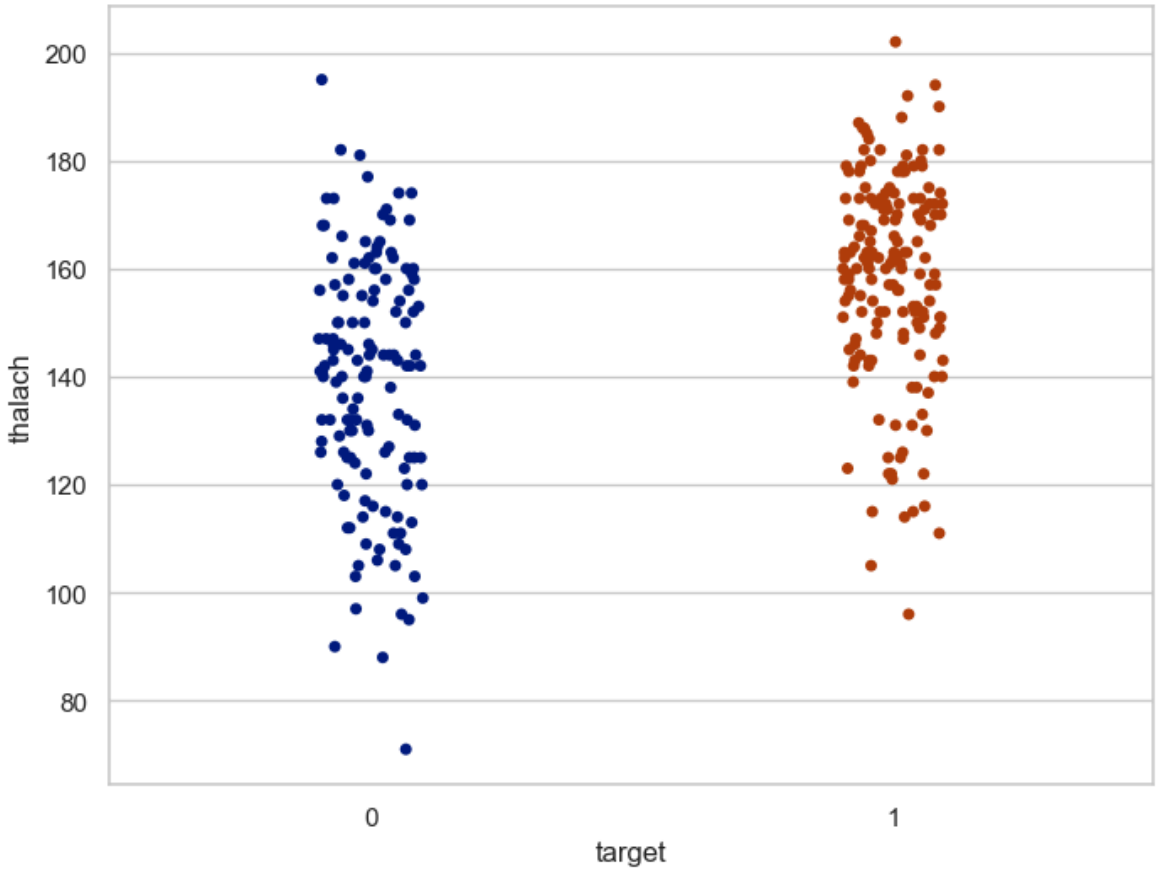


In [81]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="thalach", data=df, jitter = 0.01,palette='dark')
plt.show()
```

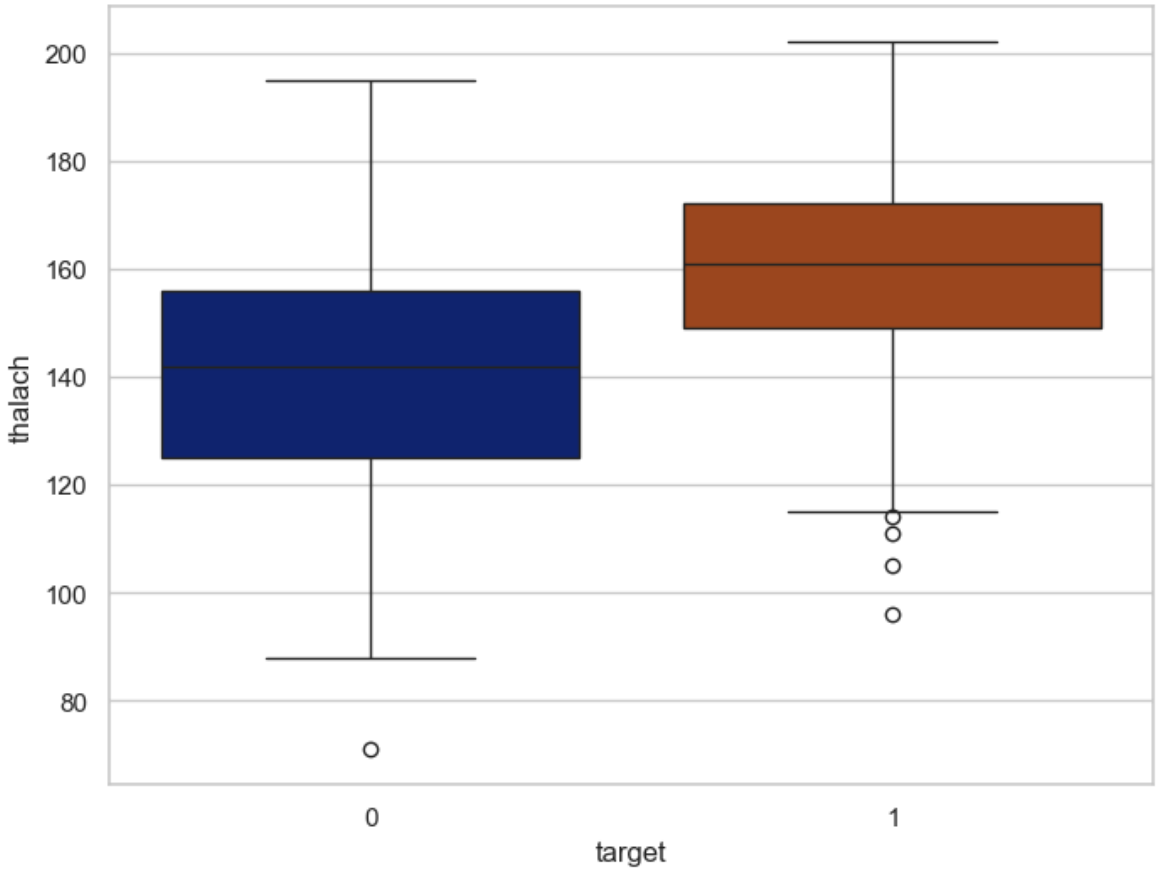


In [82]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="thalach", data=df,palette='dark')
```

```
plt.show()
```



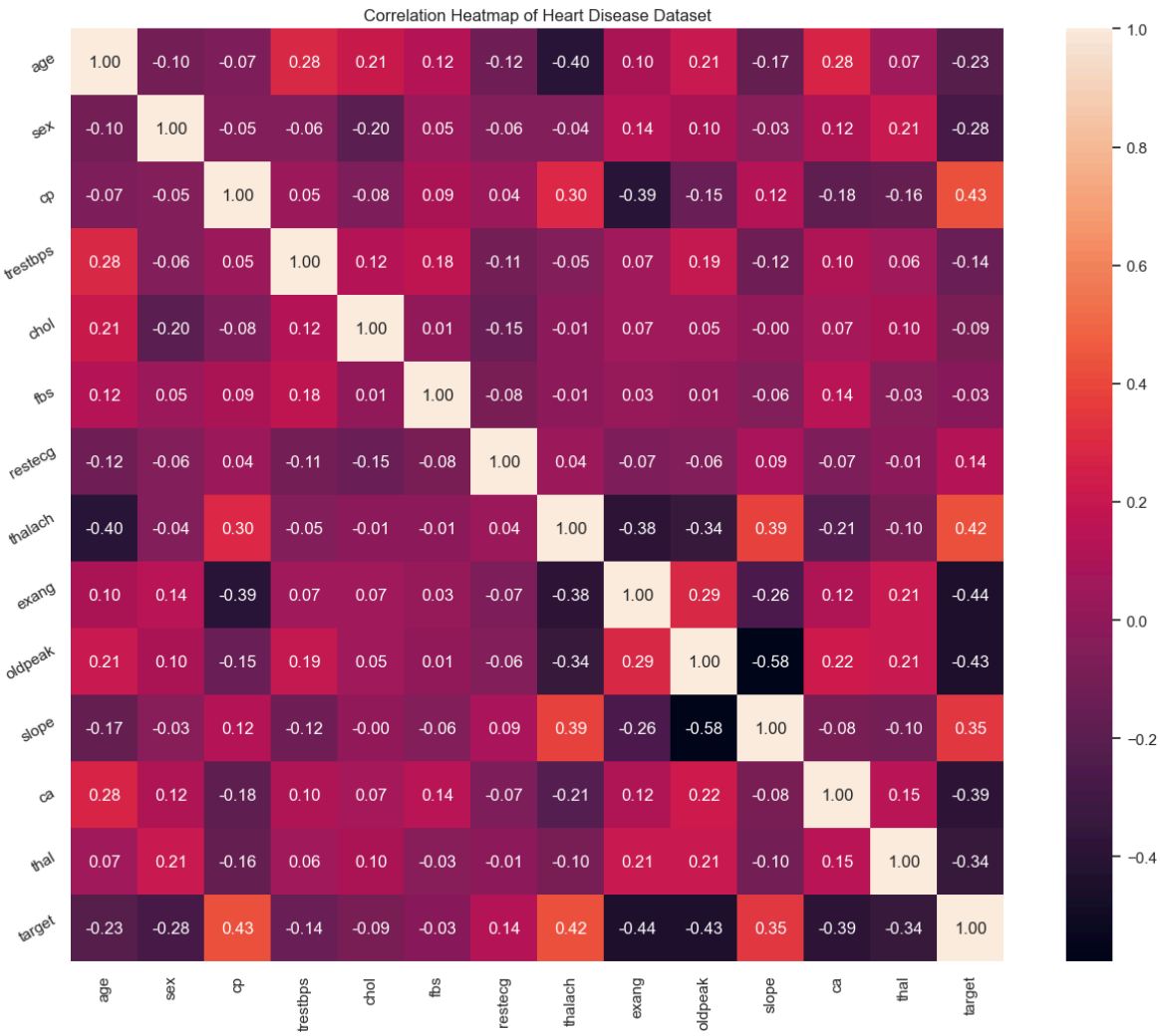
In [83]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="target", y="thalach", data=df,palette='dark')
plt.show()
```



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).

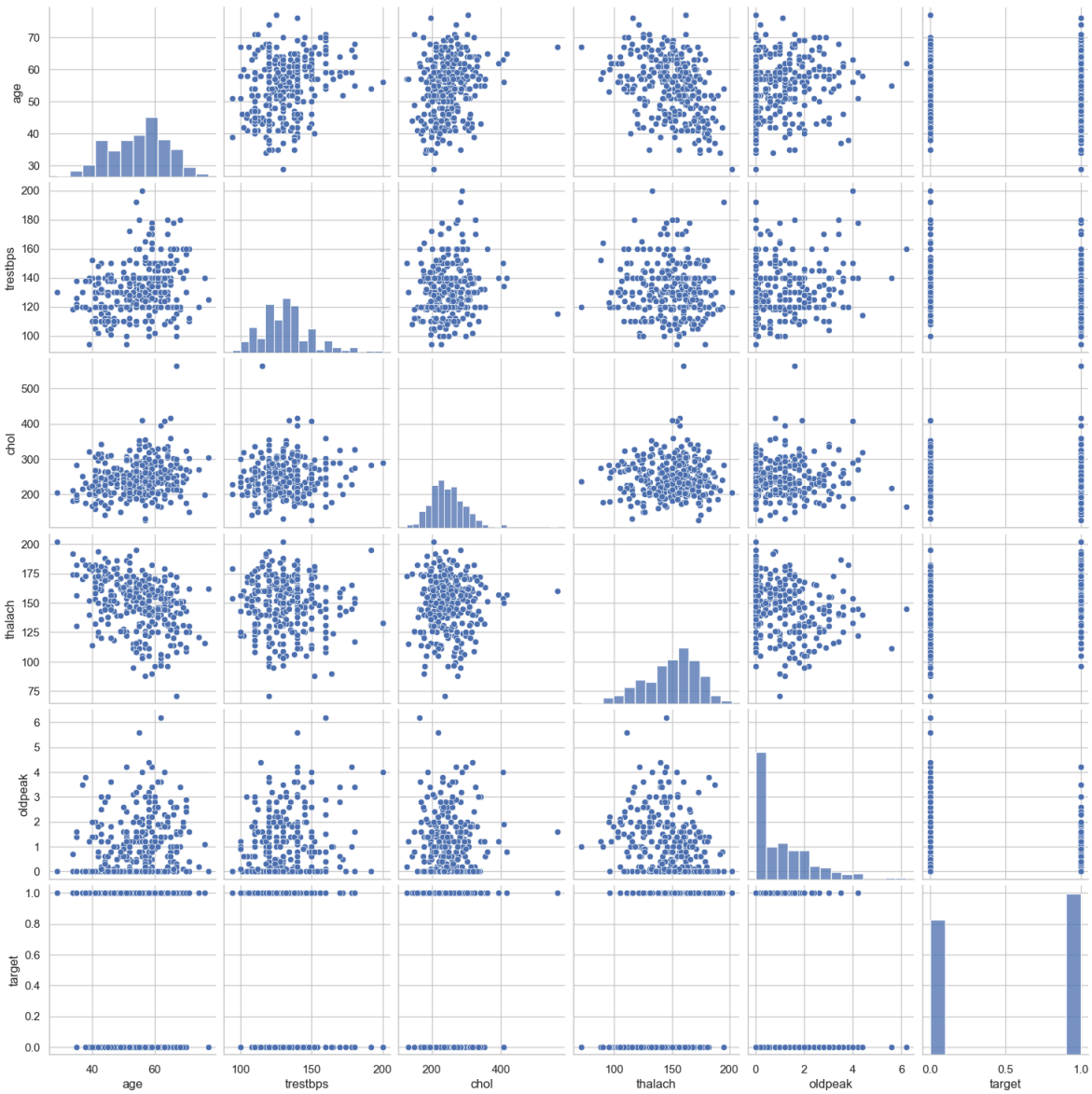
Multivariate Analysis

In [86]:
```
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()
```



Pair Plot

In [88]:
```
num_var=['age','trestbps','chol','thalach','oldpeak','target']
sns.pairplot(df[num_var],kind='scatter',diag_kind='hist',palette='dark')
plt.show()
```



In [ ]:

In [89]:
```
df['age'].nunique()
```

Out[89]: 41

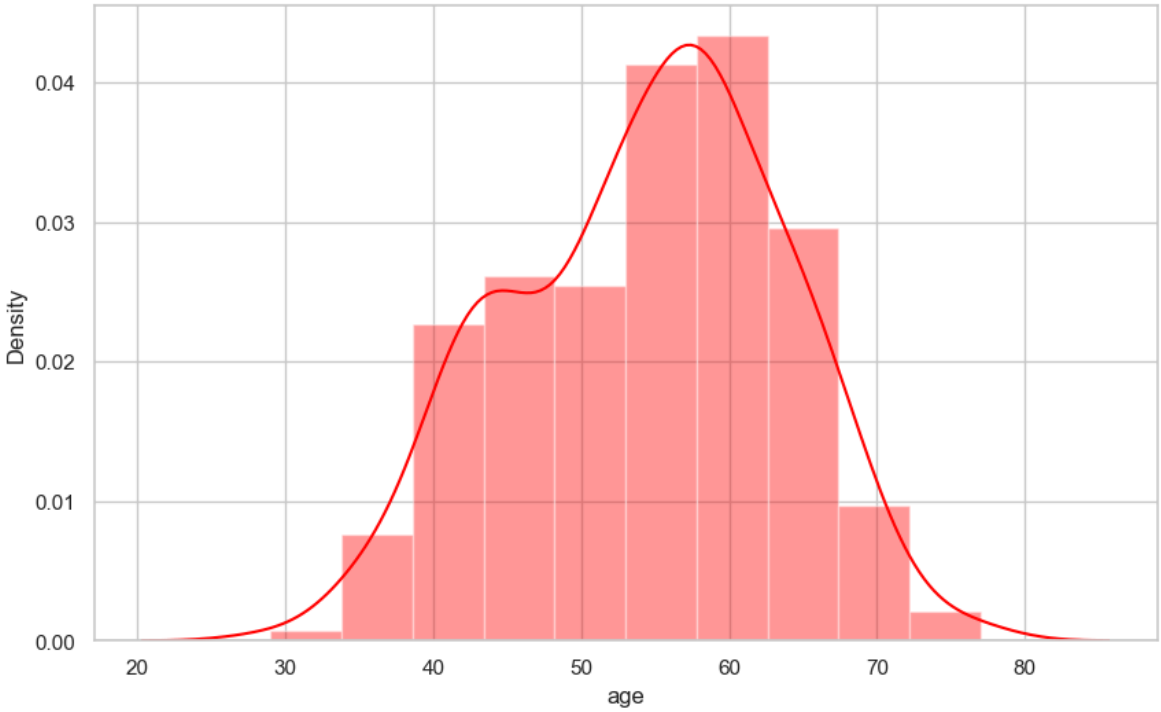
In [90]:
```
df['age'].describe()
```

Out[90]:
```
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
```

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

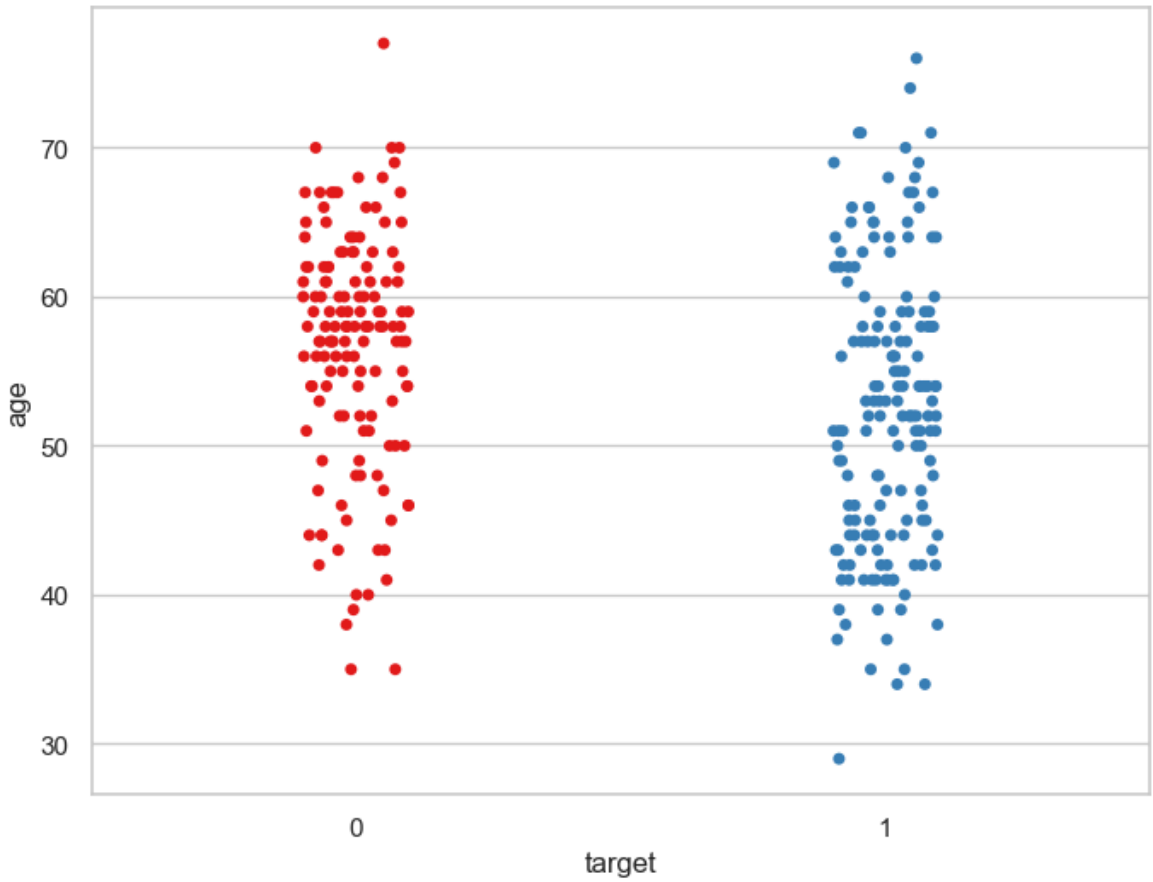
In [96]:
```
f, ax = plt.subplots(figsize=(10,6))
x = df['age']
```

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



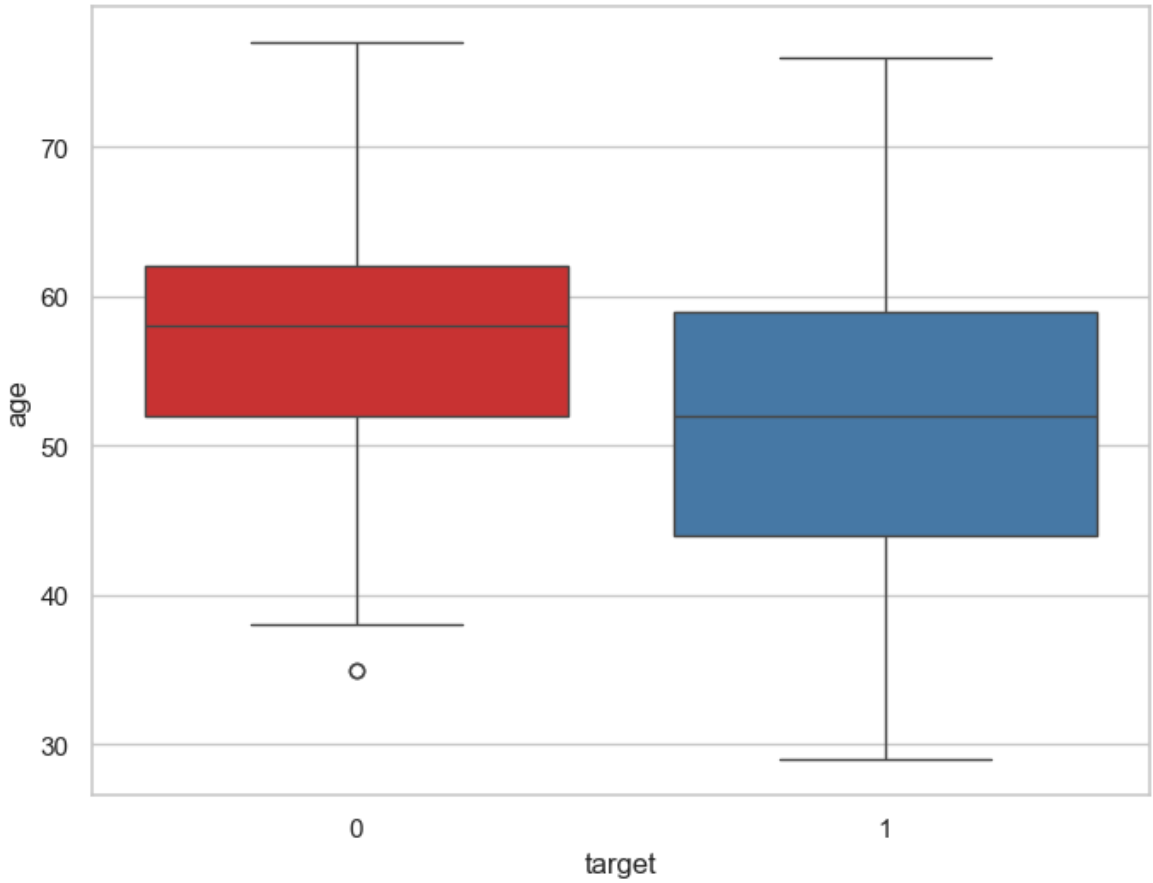
In [101…

```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="age", data=df,palette='Set1')
plt.show()
```



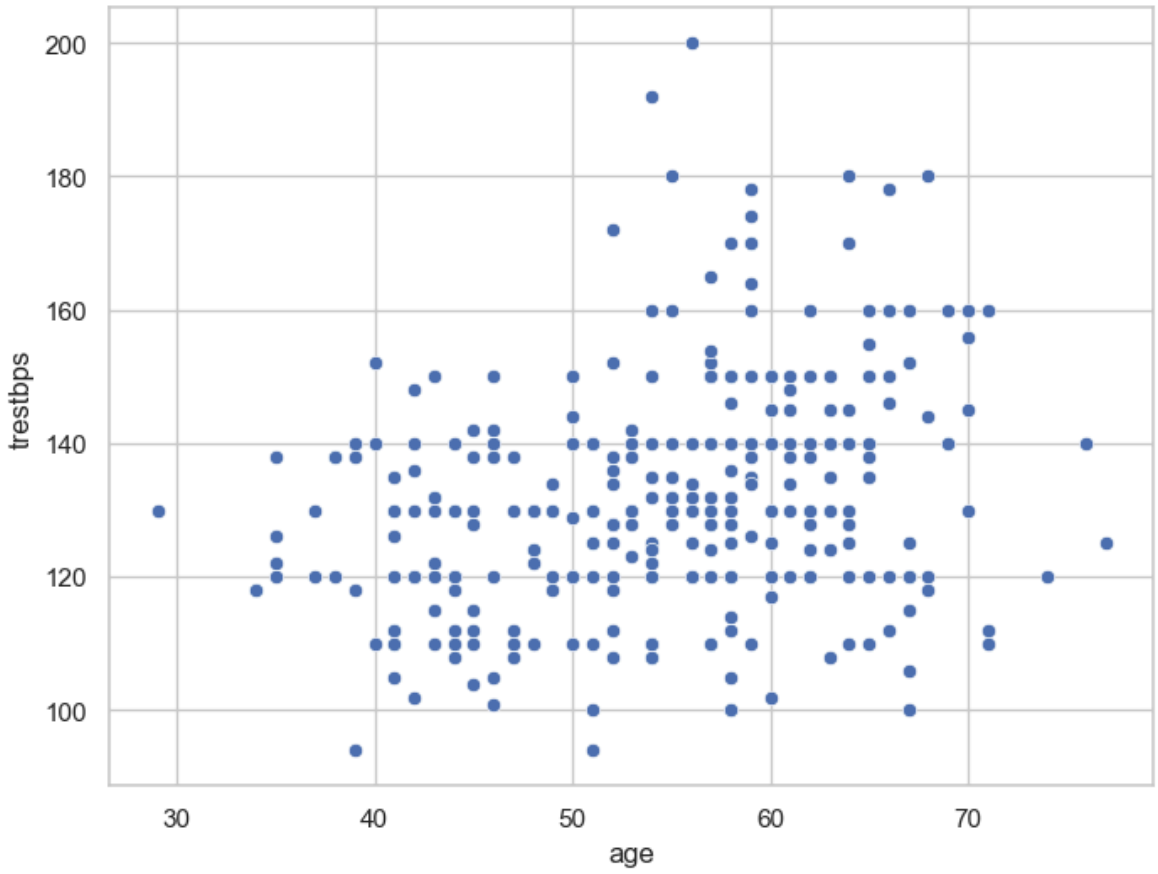
In [102…

```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x="target", y="age", data=df,palette='Set1')
plt.show()
```



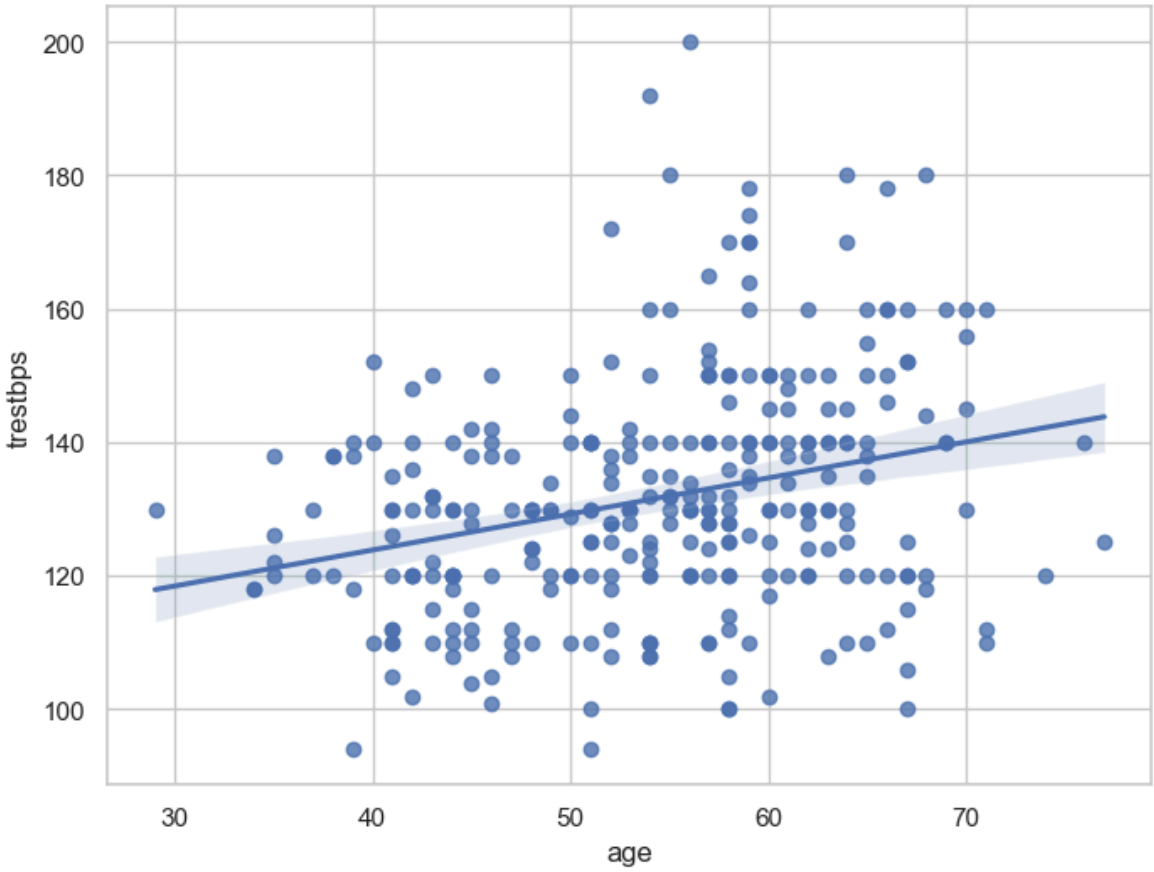
In [108…

```
f, ax = plt.subplots(figsize=(8, 6))
ax=sns.scatterplot(x="age", y="trestbps", data=df,palette='Set2')
plt.show()
```



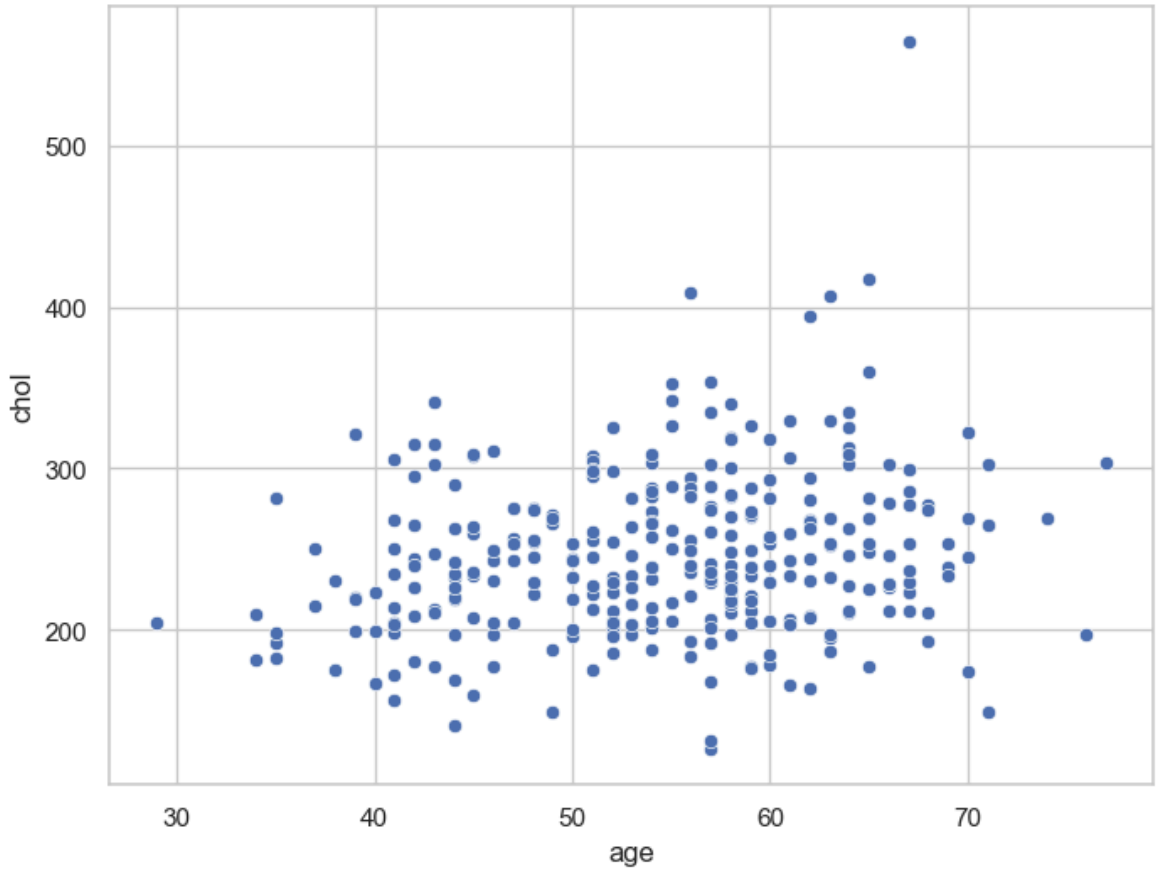
In [109…

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



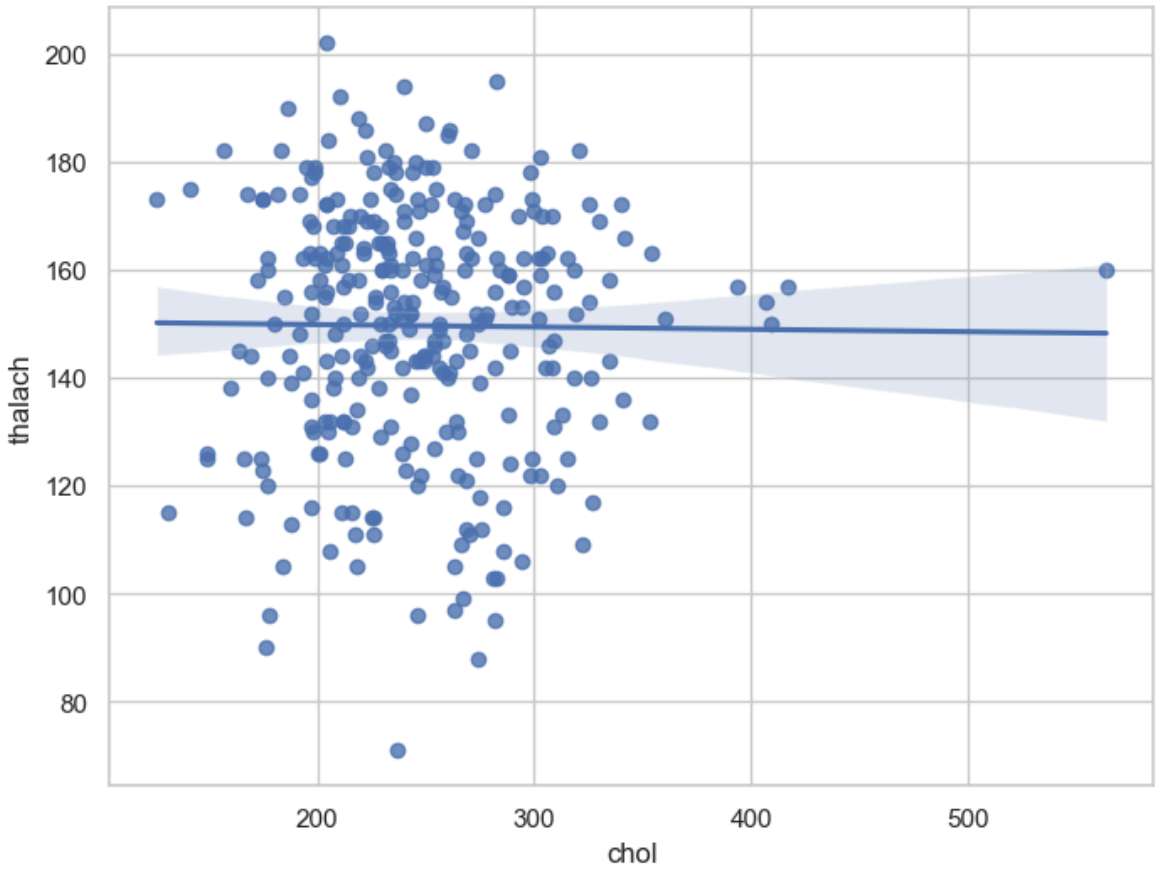
In [111…

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



In [112…

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



Dealing with missing values

In [113…

```
df.isnull()
```

Out[113…

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

303 rows × 14 columns

In [114…

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

Out[114…

```
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 [115…

```
df.isnull().count()
```

Out[115…

```
age         303
sex         303
cp          303
trestbps    303
chol        303
fbs         303
restecg     303
thalach     303
exang       303
oldpeak     303
slope       303
ca          303
thal        303
target      303
dtype: int64
```

In [116…

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

Out[116…

```
0
```

In [117…

```
df.isnull().mean()
```

Out[117…

```
age         0.0
sex         0.0
cp          0.0
trestbps    0.0
chol        0.0
fbs         0.0
restecg     0.0
thalach     0.0
exang       0.0
oldpeak     0.0
slope       0.0
ca          0.0
thal        0.0
target      0.0
dtype: float64
```

In [118…

```
df.isnull().any()
```

Out[118…

```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

In [119…

```
df.isnull().any().any()
```

Out[119…

```
False
```

In [120…
```
df.isnull().values.any()
```
Out[120…
```
False
```
In [121…
```
df.isnull().values.sum()
```
Out[121…
```
0
```

Check with Assery 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 [122…
```
assert pd.notnull(df).all().all()
```
In [123…
```
assert (df>=0).all().all()
```
the above two commands do not throw any error hence there is no missing values or -ve in the dataset

Outlier Detection

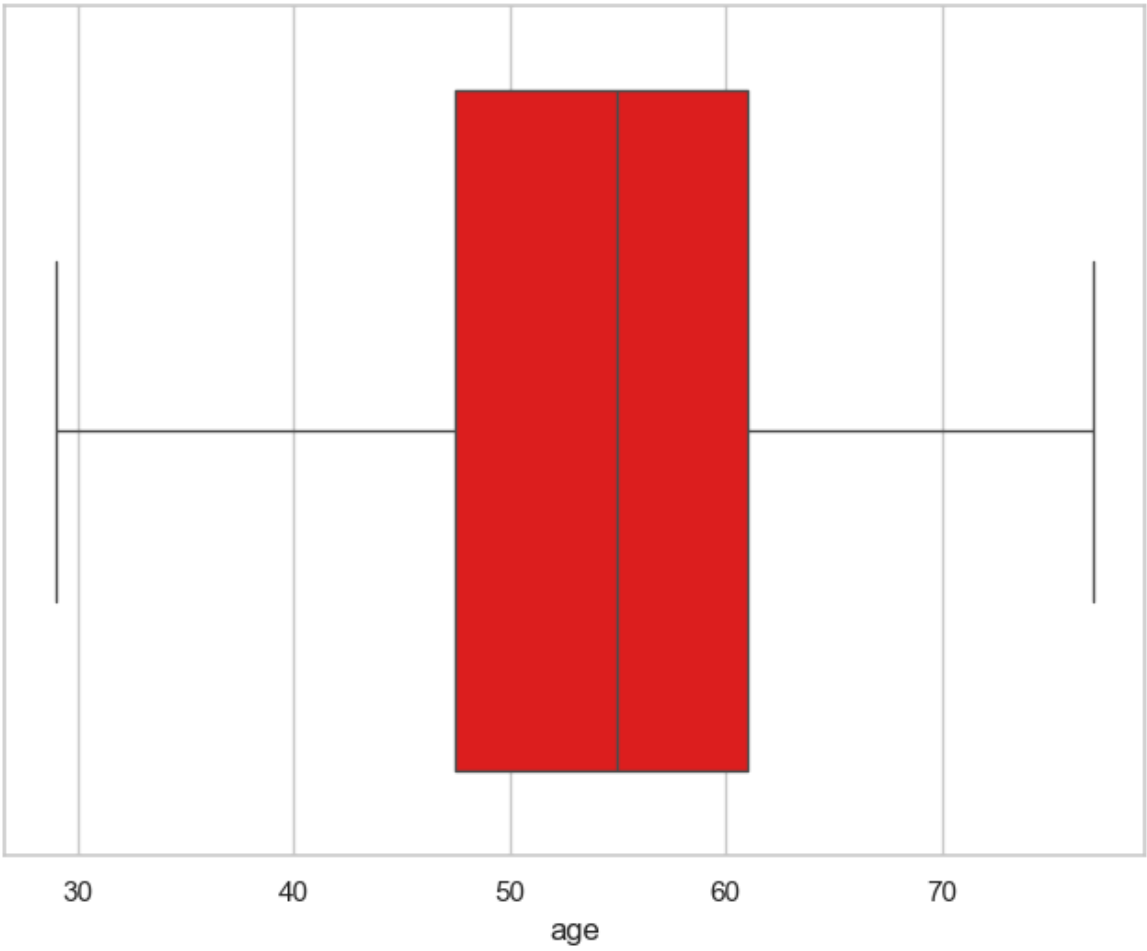
I will make boxplots to visualise outliers in the continuous numerical variables : -

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

In [125…
```
df['age'].describe()
```
Out[125…
```
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 [127…

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



In [128…

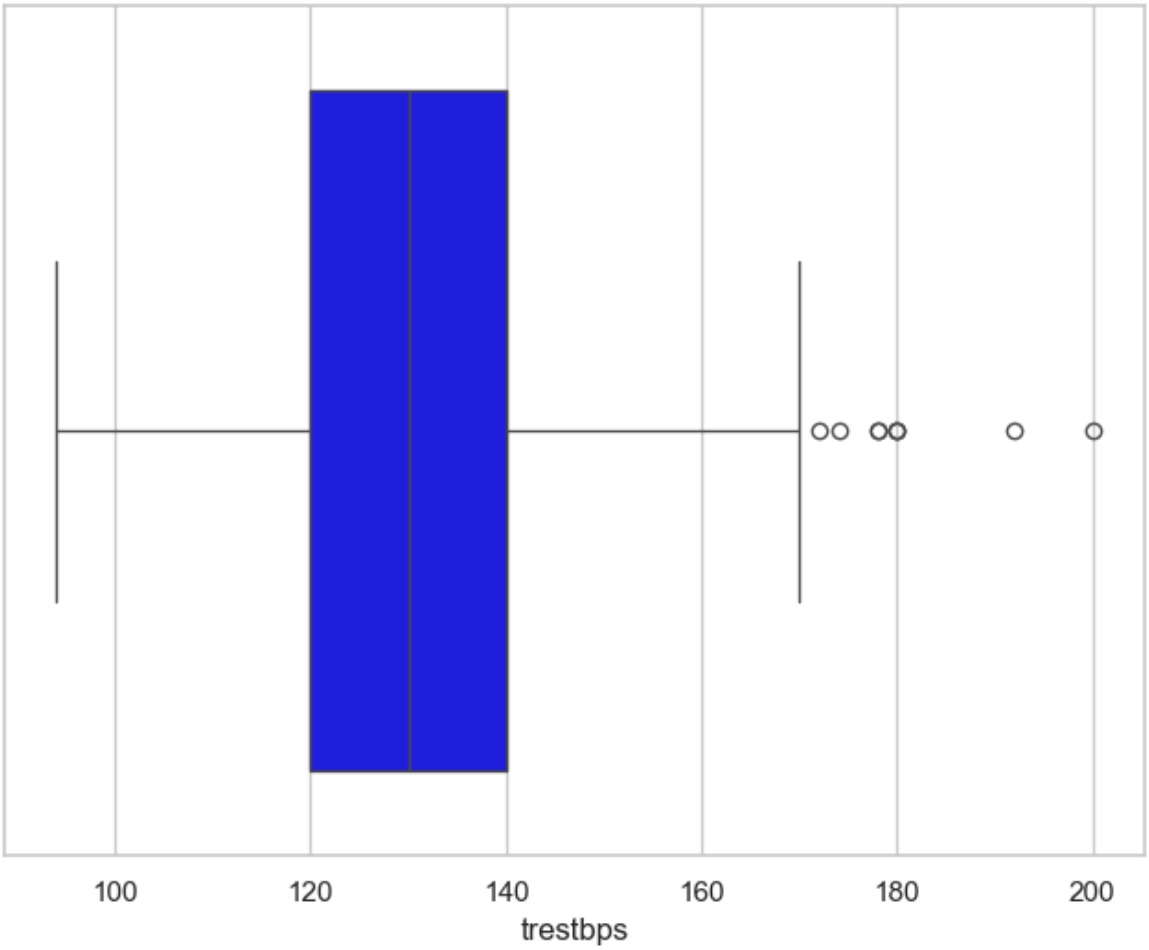
```
df['trestbps'].describe()
```

Out[128…

```
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 [129…

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



In [130…

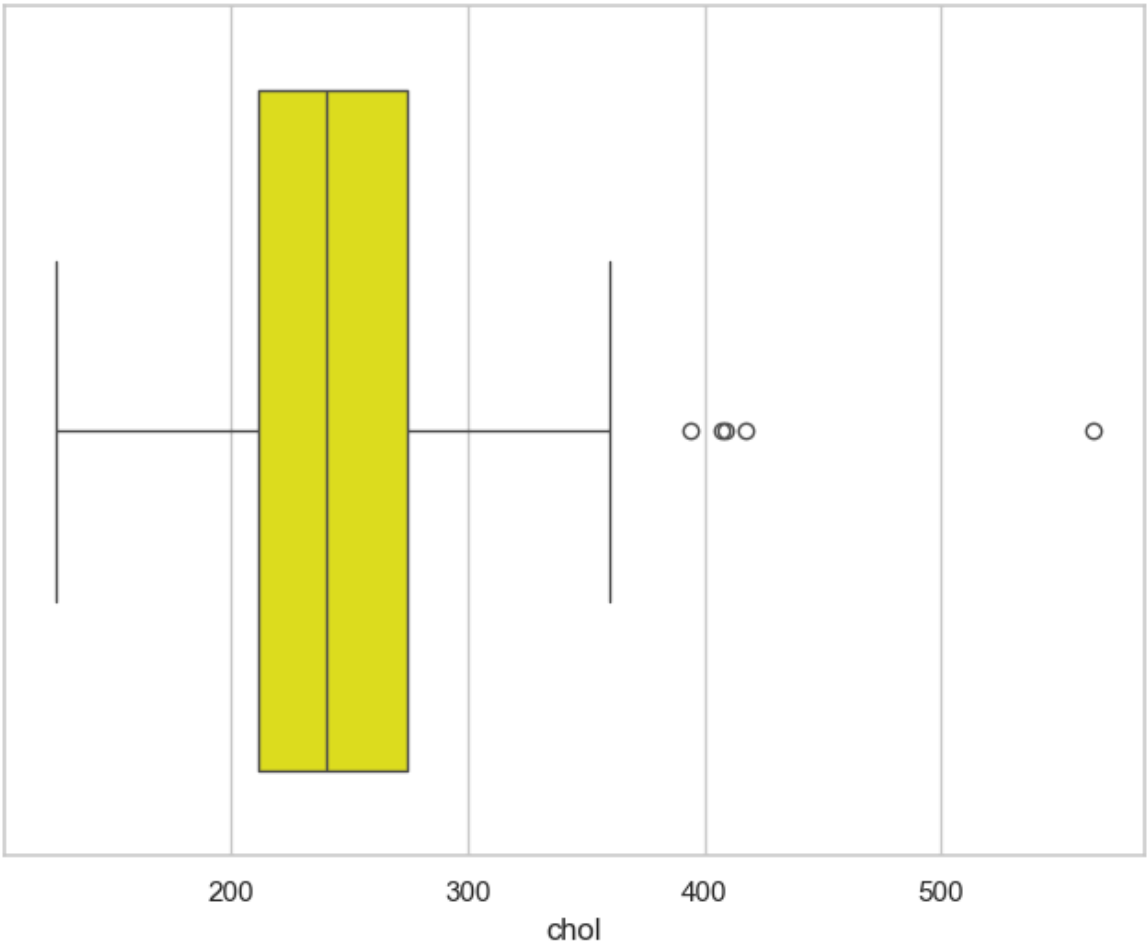
```
df['chol'].describe()
```

Out[130…

```
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 [131…

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



In [132…

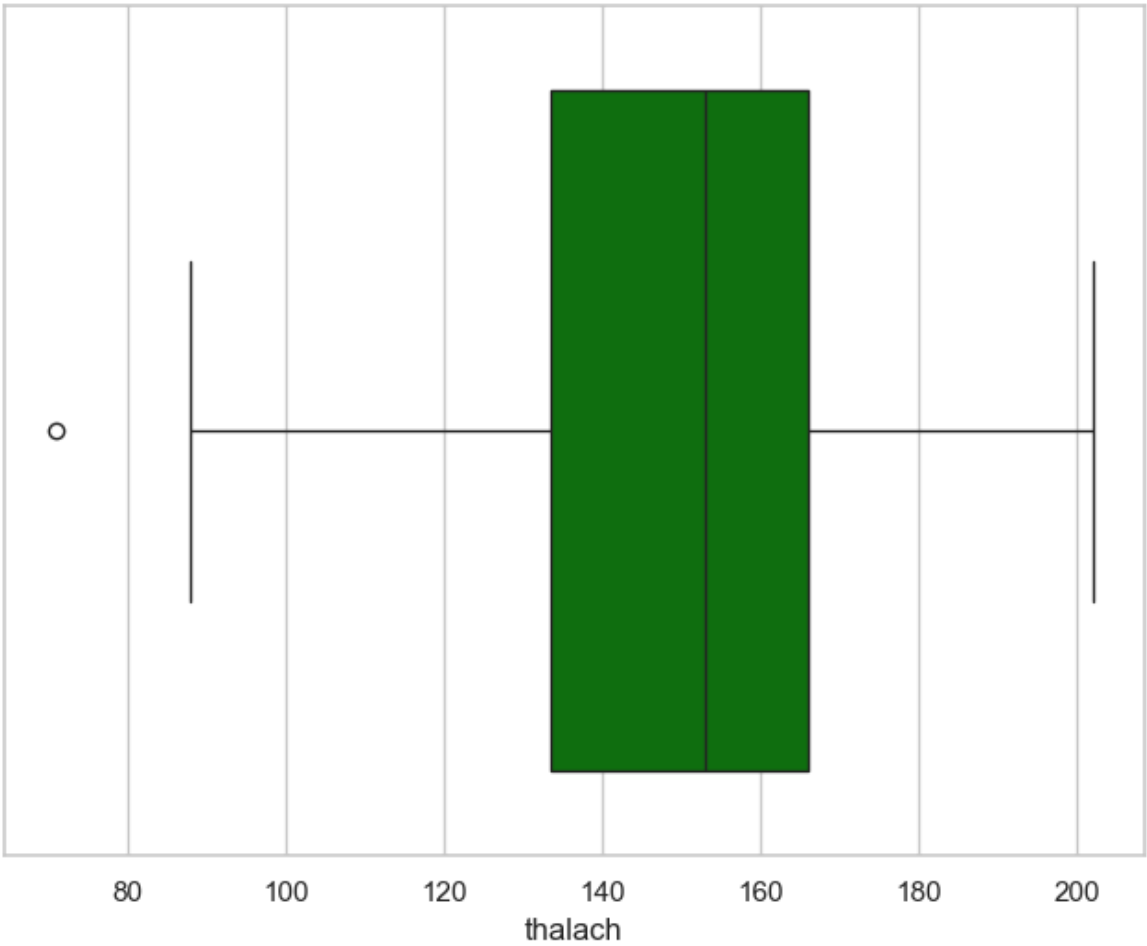
```
df['thalach'].describe()
```

Out[132…

```
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 [133…

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



In [134…

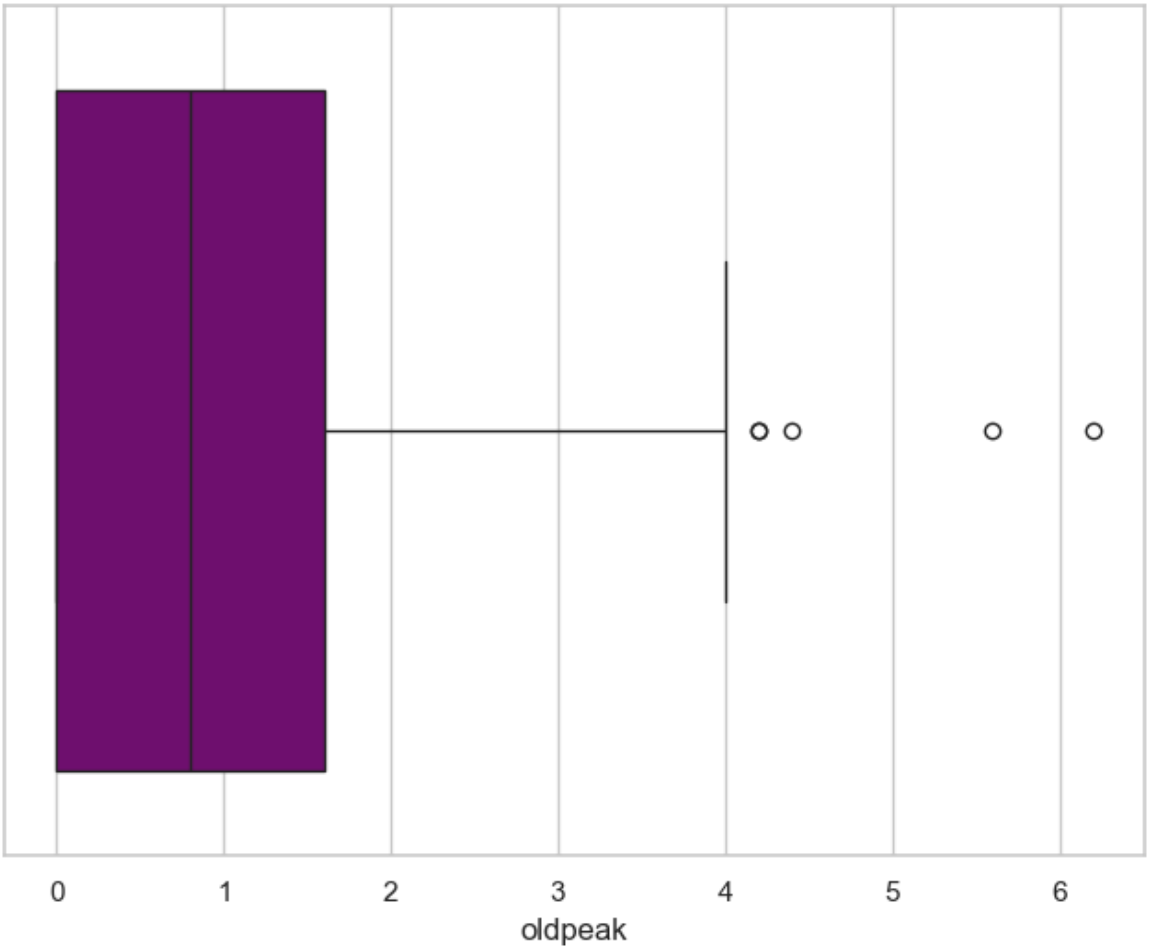
```
df['oldpeak'].describe()
```

Out[134…

```
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 [135…

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

we have finally completed our analysis on Heart Patients Analysis

In [ ]: