In [15]:
```
import numpy as np
```

In [16]:
```
import pandas as pd
```

In [17]:
```
import os
for dirname,_,filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname,filename))
```

In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
%matplotlib inline
sns.set(style="whitegrid")
```

In [19]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [20]:
```
df=pd.read_csv(r"D:\Data Science with AI\Data Science With AI\24th, 25th-july- A
```

In [21]:
```
print('The shape of the dataset:',df.shape)
```

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

In [22]:
```
df.shape
```

Out[22]: (303, 14)

In [23]:
```
df.head()
```

Out[23]:

| | 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 [24]:
```
df.tail()
```

Out[24]:

| | 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 [25]:
```
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 [26]:
```
df.dtypes
```

Out[26]:
```
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 [27]:
```
df.describe()
```

Out[27]:

| | 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 [28]:
```
df.describe(include=['object'])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[28], line 1
----> 1 df.describe(include=['object'])

File D:\New folder\Lib\site-packages\pandas\core\generic.py:11976, in NDFrame.des
cribe(self, percentiles, include, exclude)
  11734 @final
  11735 def describe(
  11736     self,
   (...)
  11739     exclude=None,
  11740 ) -> Self:
  11741     """
  11742     Generate descriptive statistics.
  11743
   (...)
  11974     max            NaN      3.0
  11975     """
> 11976     return describe_ndframe(
  11977         obj=self,
  11978         include=include,
  11979         exclude=exclude,
  11980         percentiles=percentiles,
  11981     ).__finalize__(self, method="describe")

File D:\New folder\Lib\site-packages\pandas\core\methods\describe.py:97, in descr
ibe_ndframe(obj, include, exclude, percentiles)
     90 else:
     91     describer = DataFrameDescriber(
     92         obj=cast("DataFrame", obj),
     93         include=include,
     94         exclude=exclude,
     95     )
---> 97 result = describer.describe(percentiles=percentiles)
     98 return cast(NDFrameT, result)

File D:\New folder\Lib\site-packages\pandas\core\methods\describe.py:175, in Data
FrameDescriber.describe(self, percentiles)
    172     ldesc.append(describe_func(series, percentiles))
    174 col_names = reorder_columns(ldesc)
--> 175 d = concat(
    176     [x.reindex(col_names, copy=False) for x in ldesc],
    177     axis=1,
    178     sort=False,
    179 )
    180 d.columns = data.columns.copy()
    181 return d

File D:\New folder\Lib\site-packages\pandas\core\reshape\concat.py:382, in concat
(objs, axis, join, ignore_index, keys, levels, names, verify_integrity, sort, cop
y)
    379 elif copy and using_copy_on_write():
    380     copy = False
--> 382 op = _Concatenator(
    383     objs,
    384     axis=axis,
    385     ignore_index=ignore_index,
    386     join=join,
    387     keys=keys,
```

```
    388     levels=levels,
    389     names=names,
    390     verify_integrity=verify_integrity,
    391     copy=copy,
    392     sort=sort,
    393 )
    395 return op.get_result()

File D:\New folder\Lib\site-packages\pandas\core\reshape\concat.py:445, in _Concatenator.__init__(self, objs, axis, join, keys, levels, names, ignore_index, verify_integrity, copy, sort)
    442 self.verify_integrity = verify_integrity
    443 self.copy = copy
--> 445 objs, keys = self._clean_keys_and_objs(objs, keys)
    447 # figure out what our result ndim is going to be
    448 ndims = self._get_ndims(objs)

File D:\New folder\Lib\site-packages\pandas\core\reshape\concat.py:507, in _Concatenator._clean_keys_and_objs(self, objs, keys)
    504     objs_list = list(objs)
    506 if len(objs_list) == 0:
--> 507     raise ValueError("No objects to concatenate")
    509 if keys is None:
    510     objs_list = list(com.not_none(*objs_list))

ValueError: No objects to concatenate
```

In [29]:
```
df.describe(include='all')
```

Out[29]:

| | age | sex | cp | trestbps | chol | fbs | reste |
|---|---|---|---|---|---|---|---|
| **count** | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000000 | 303.000( |
| **mean** | 54.366337 | 0.683168 | 0.966997 | 131.623762 | 246.264026 | 0.148515 | 0.528( |
| **std** | 9.082101 | 0.466011 | 1.032052 | 17.538143 | 51.830751 | 0.356198 | 0.525( |
| **min** | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.000000 | 0.000000 | 0.000( |
| **25%** | 47.500000 | 0.000000 | 0.000000 | 120.000000 | 211.000000 | 0.000000 | 0.000( |
| **50%** | 55.000000 | 1.000000 | 1.000000 | 130.000000 | 240.000000 | 0.000000 | 1.000( |
| **75%** | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 274.500000 | 0.000000 | 1.000( |
| **max** | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.000000 | 1.000000 | 2.000( |

In [30]:
```
df.columns
```

Out[30]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [31]:
```
df.head(1)
```

Out[31]:

| | 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 |

In [32]:
```
df.nunique['target']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[32], line 1
----> 1 df.nunique['target']

TypeError: 'method' object is not subscriptable
```

In [33]:
```
df['target'].nunique()
```

Out[33]:
```
2
```

In [34]:
```
df.['target'].unique()
```

```
  Cell In[34], line 1
    df.['target'].unique()
       ^
SyntaxError: invalid syntax
```

In [35]:
```
df['target'].unique()
```

Out[35]:
```
array([1, 0])
```

In [36]:
```
df['target'].value_counts()
```

Out[36]:
```
target
1    165
0    138
Name: count, dtype: int64
```

In [37]:
```
sns.bar['target']
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[37], line 1
----> 1 sns.bar['target']

AttributeError: module 'seaborn' has no attribute 'bar'
```

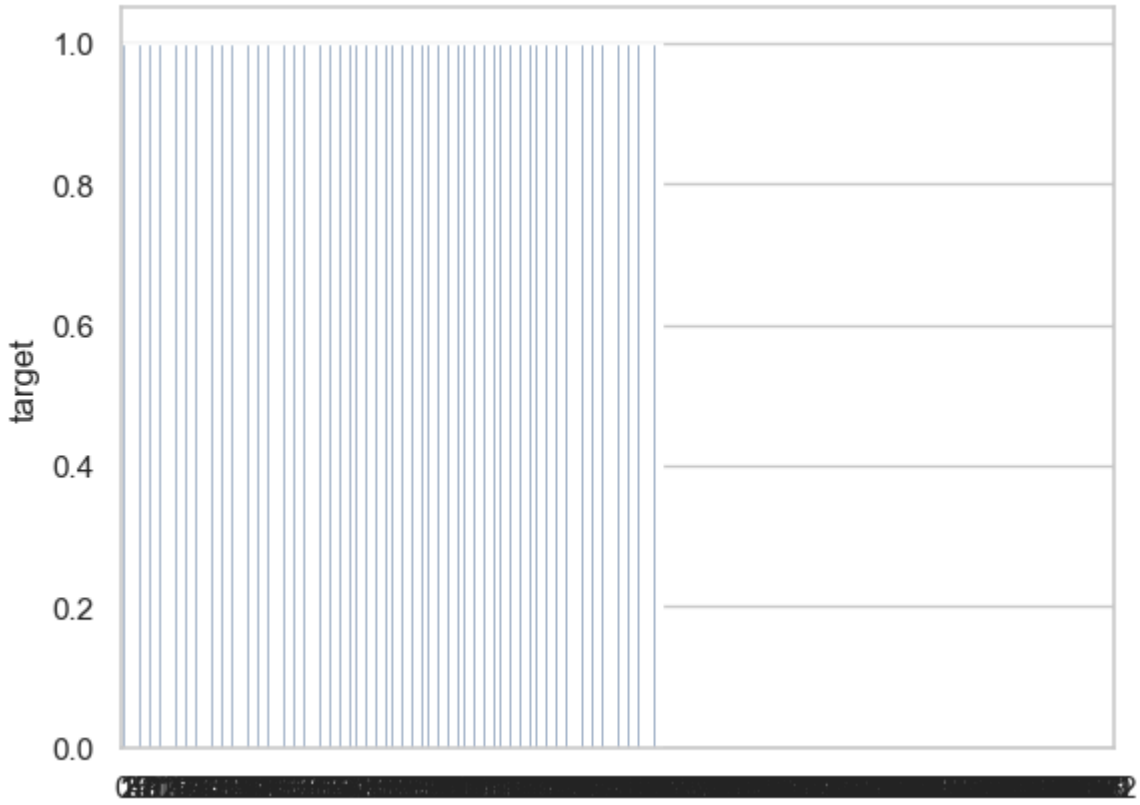
In [38]:
```
sns.barplot['target']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[38], line 1
----> 1 sns.barplot['target']

TypeError: 'function' object is not subscriptable
```

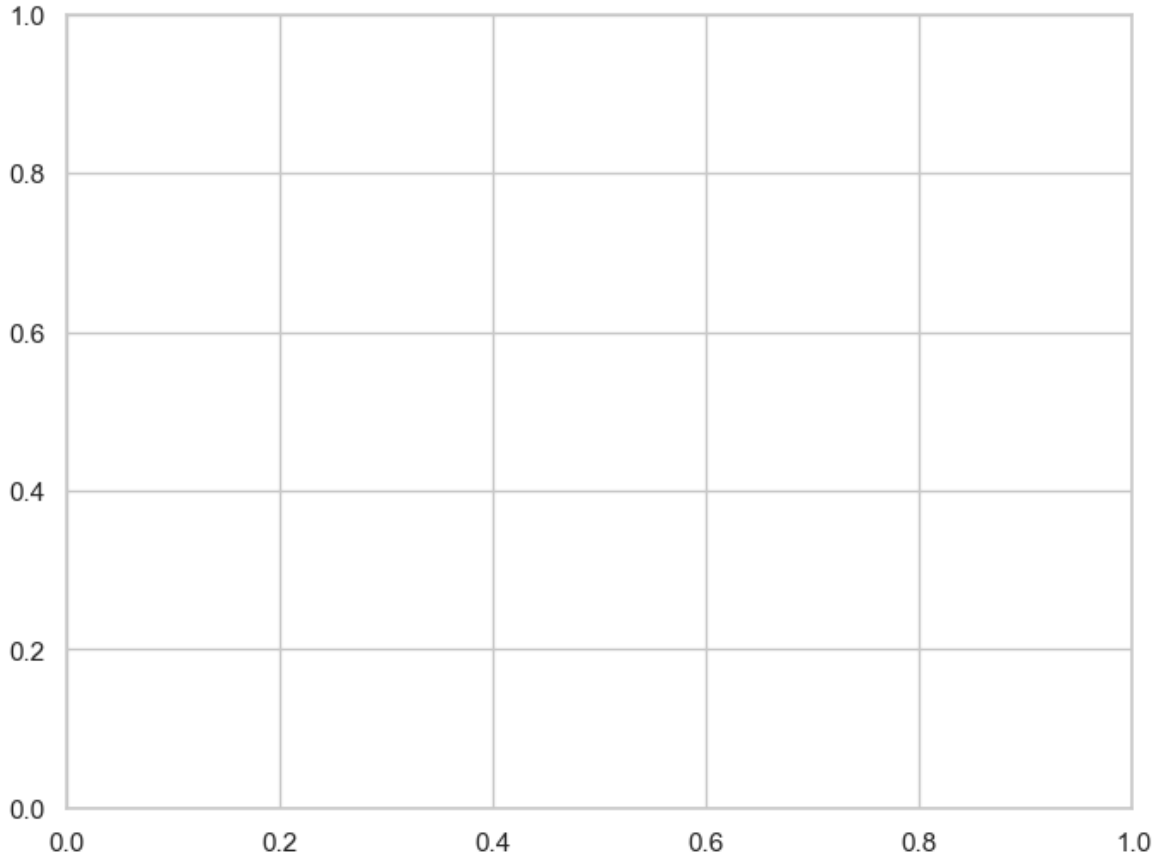
In [39]:
```
sns.barplot(df['target'])
```

Out[39]:
```
<Axes: ylabel='target'>
```

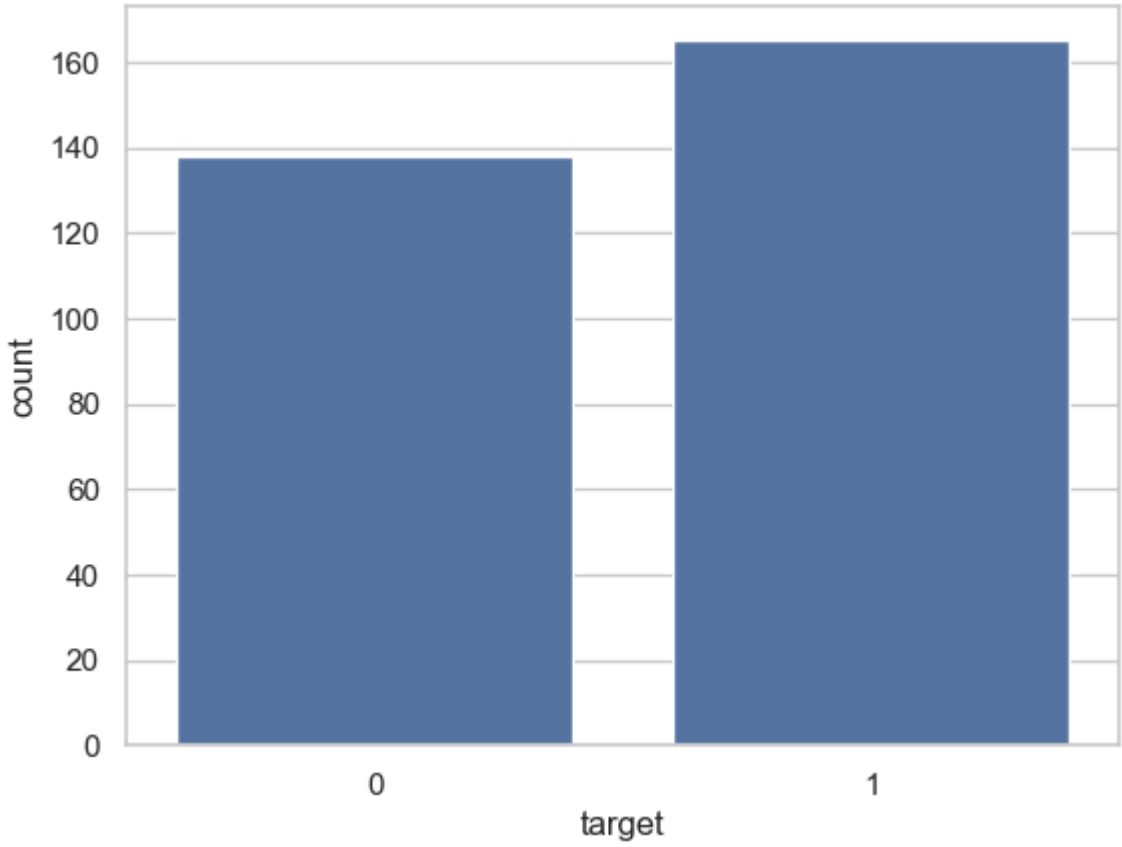
In [40]:
```
plt.show()
```



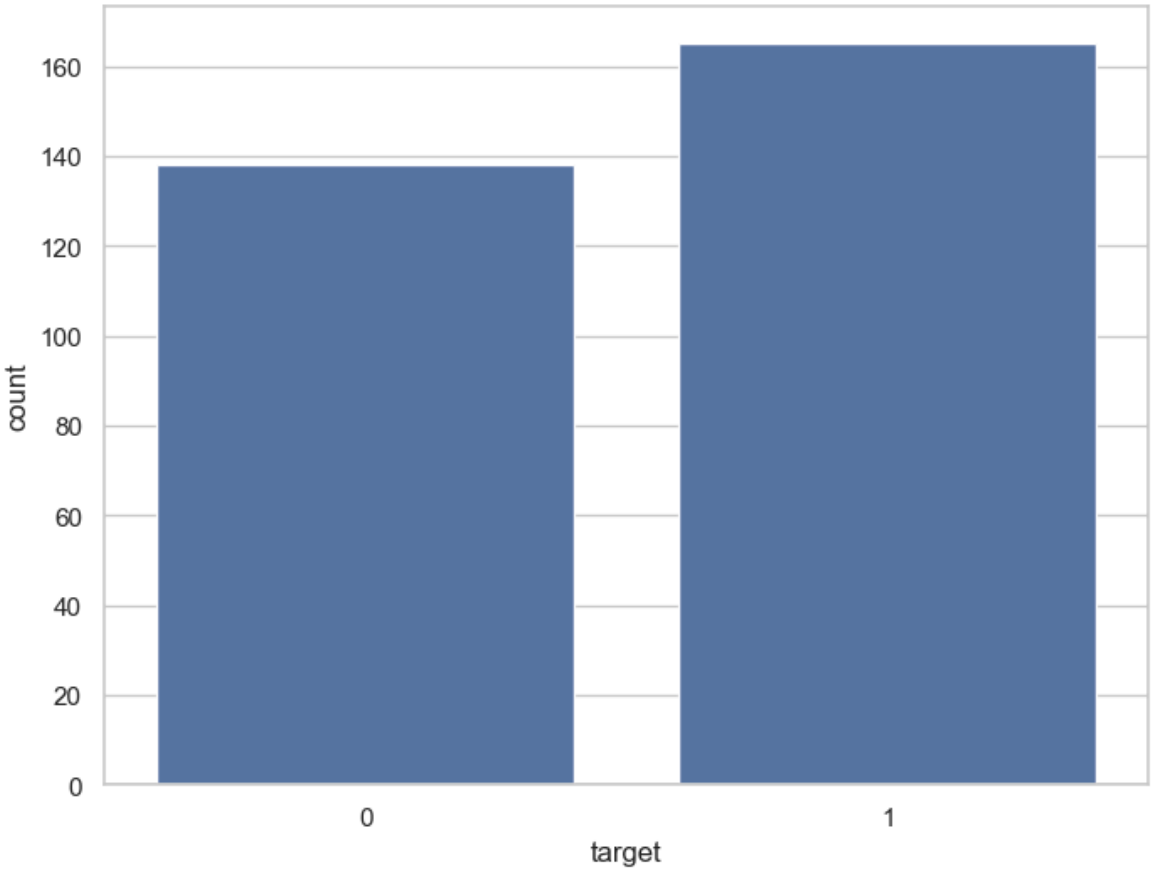

In [41]:
```
f=plt.subplots(figsize=(8,6))
plt.show()
```




In [42]:
```
ax=sns.countplot(x='target',data=df)
plt.show()
```



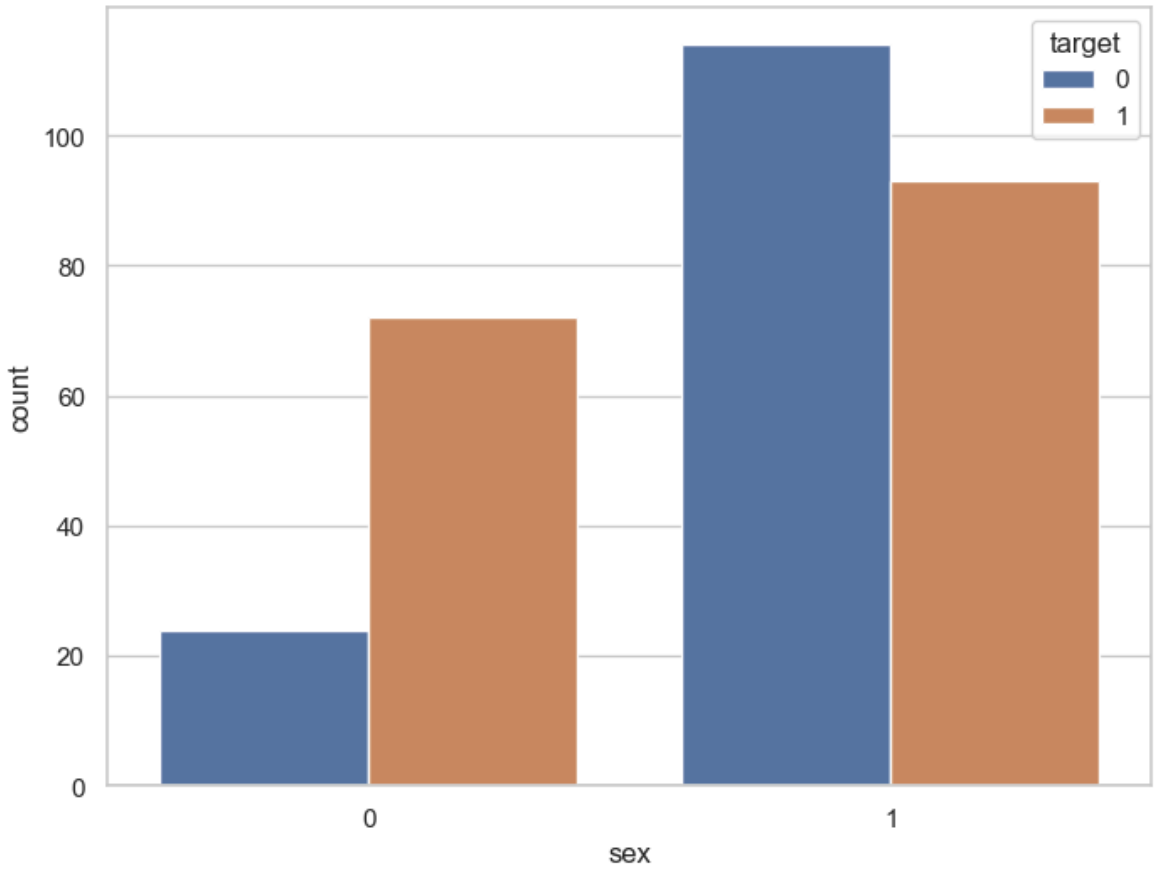

In [43]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x='target',data=df)
plt.show()
```



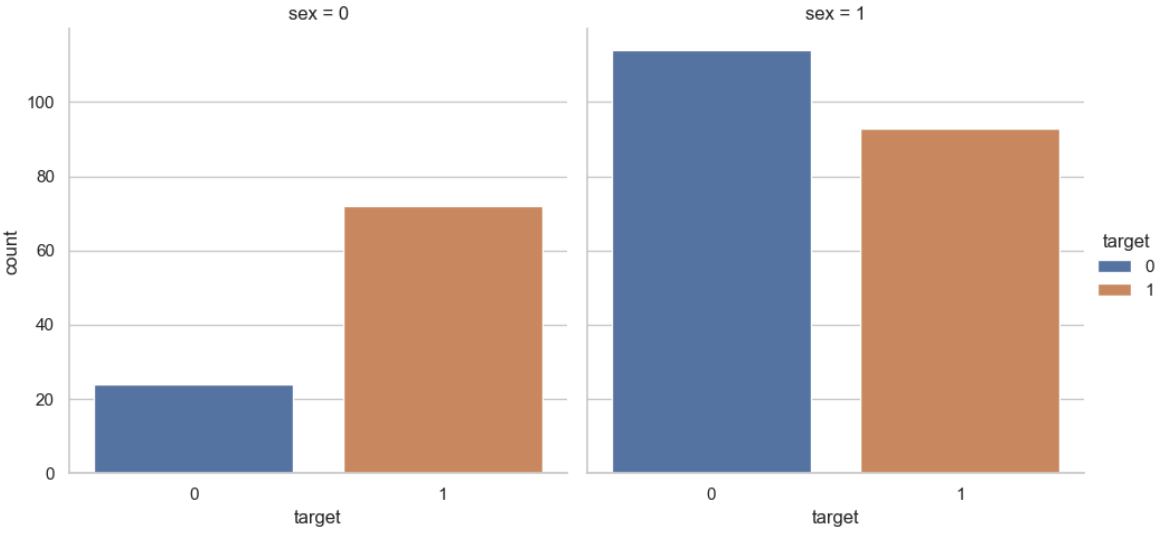

In [44]:
```
df.groupby('sex')['target'].value_counts()
```

Out[44]:
```
sex  target
0    1         72
     0         24
1    0        114
     1         93
Name: count, dtype: int64
```

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

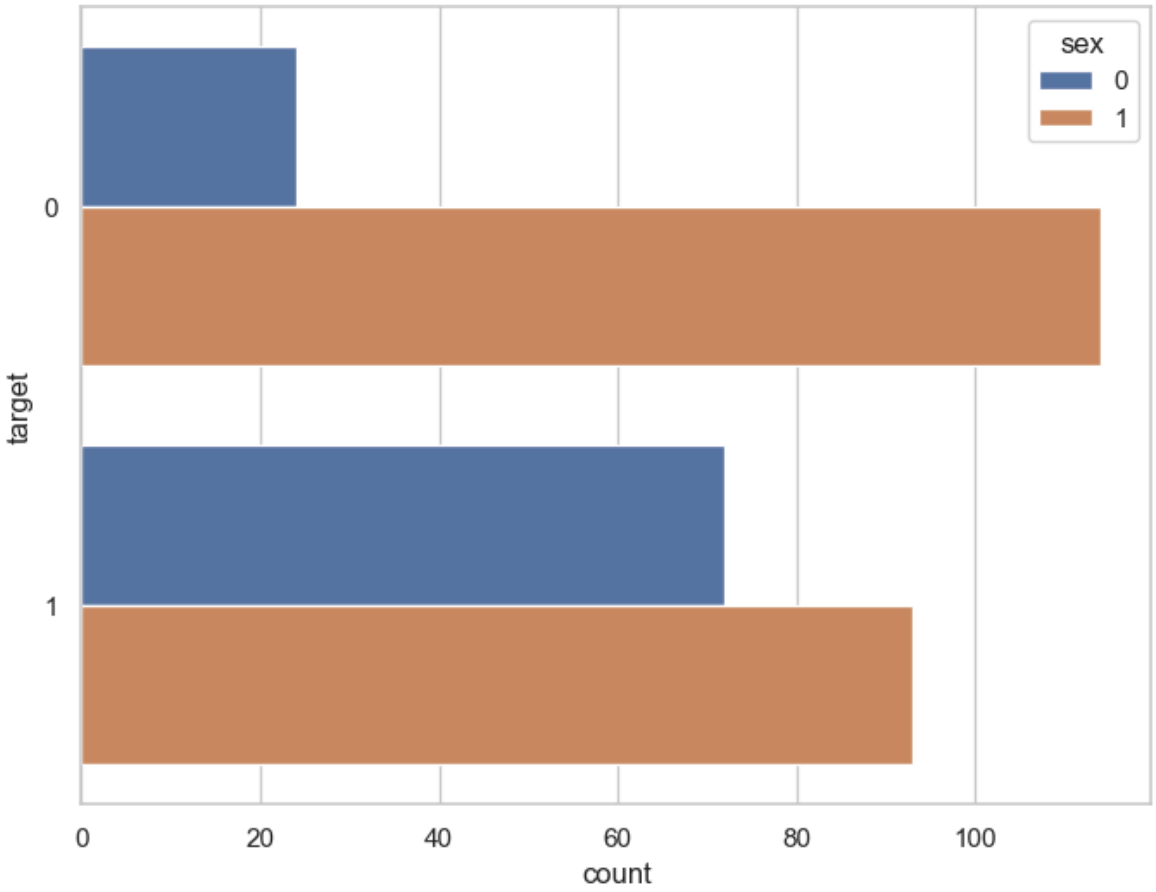



In [46]:
```
ax=sns.catplot(data=df,x='target',col='sex',kind='count',height=5,aspect=1,hue='
plt.show()
```

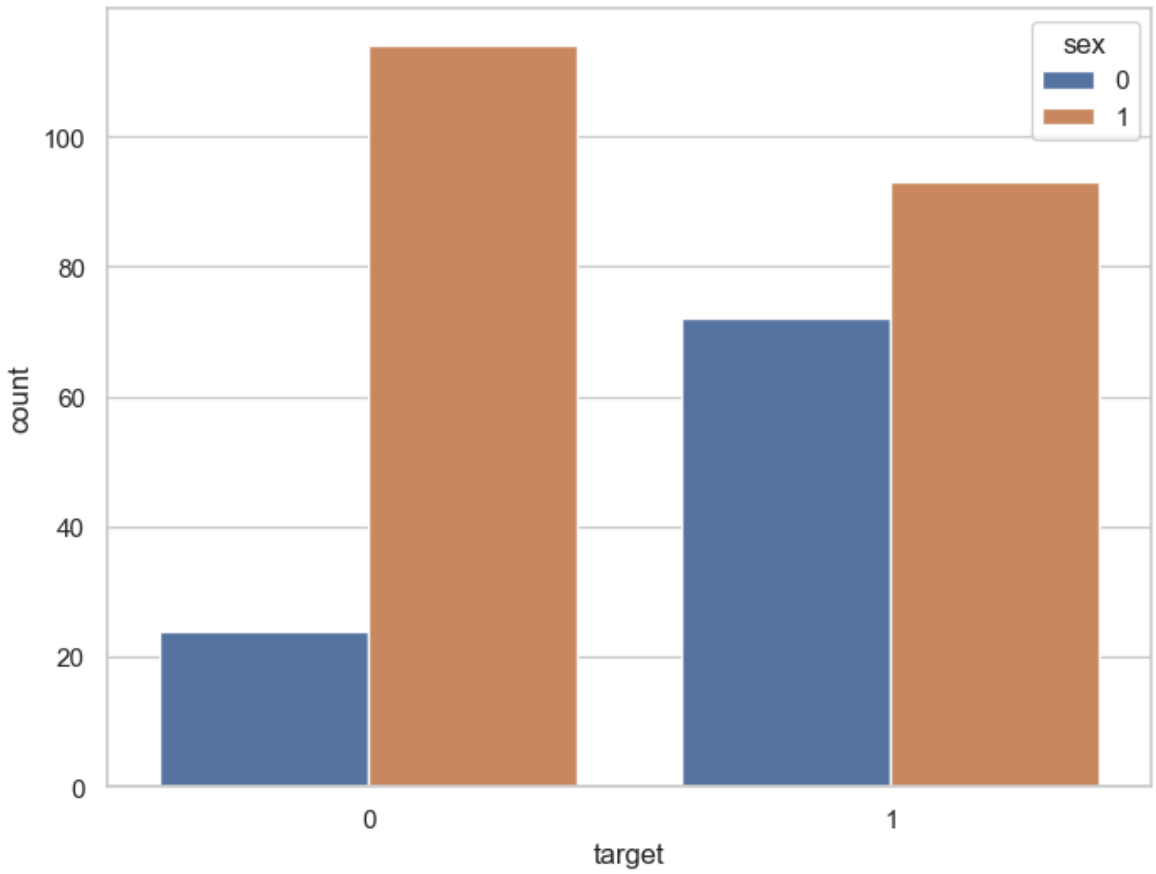



In [47]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(y='target',hue='sex',data=df)
```

```
plt.show()
```

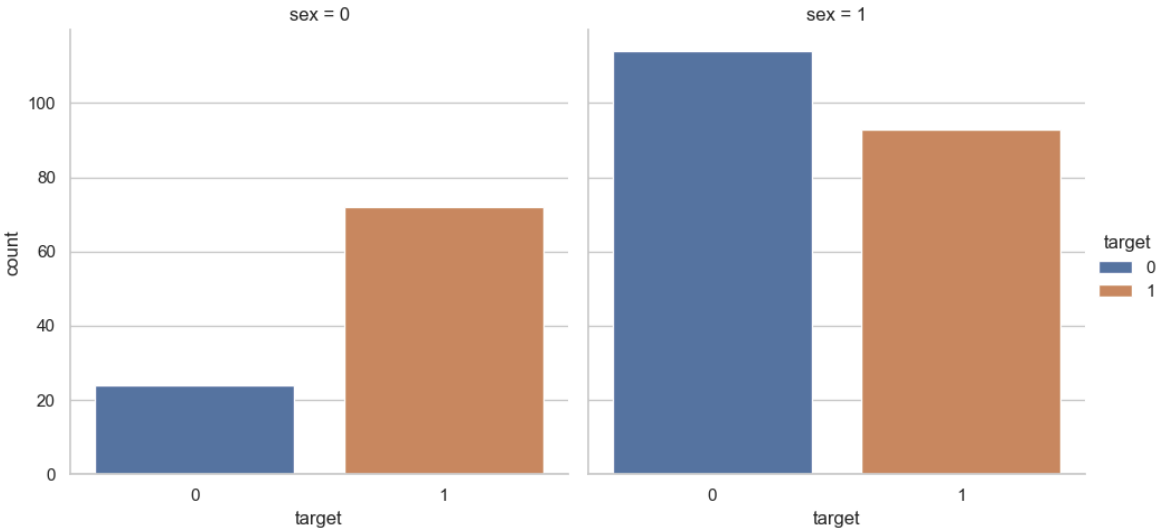



In [48]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='target',hue='sex')
plt.show()
```

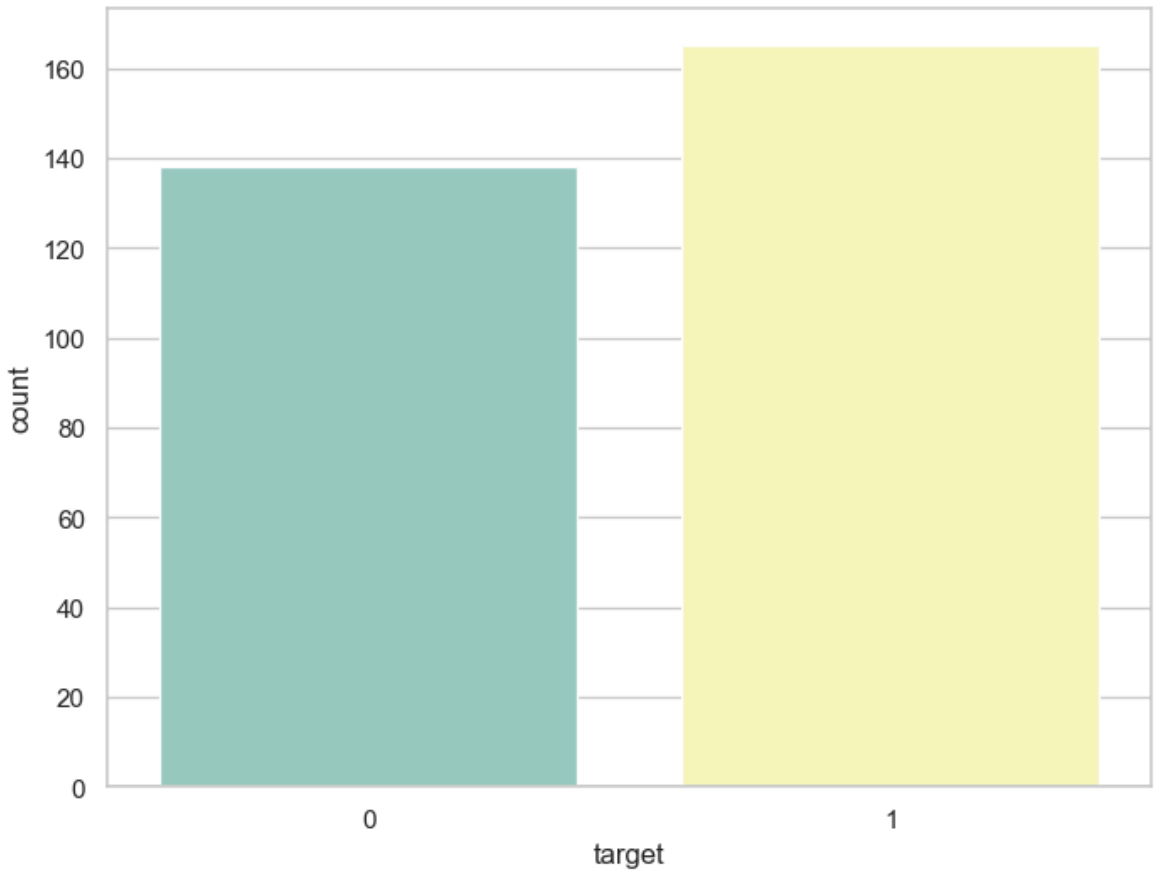


In [49]:
```
ax=sns.catplot(data=df,x='target',col='sex',kind='count',hue='target')
```

In [50]:
```
plt.show()
```




In [51]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",data=df,palette="Set3")
plt.show()
```

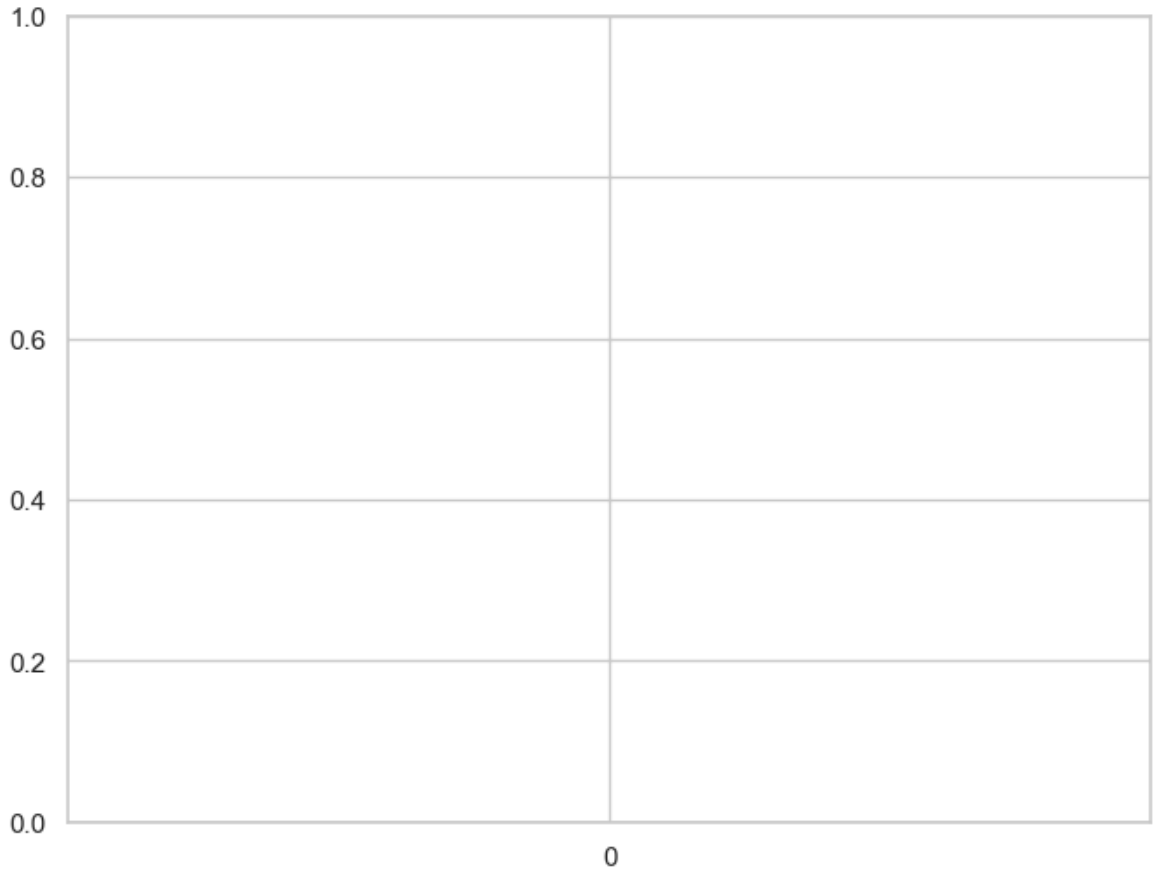


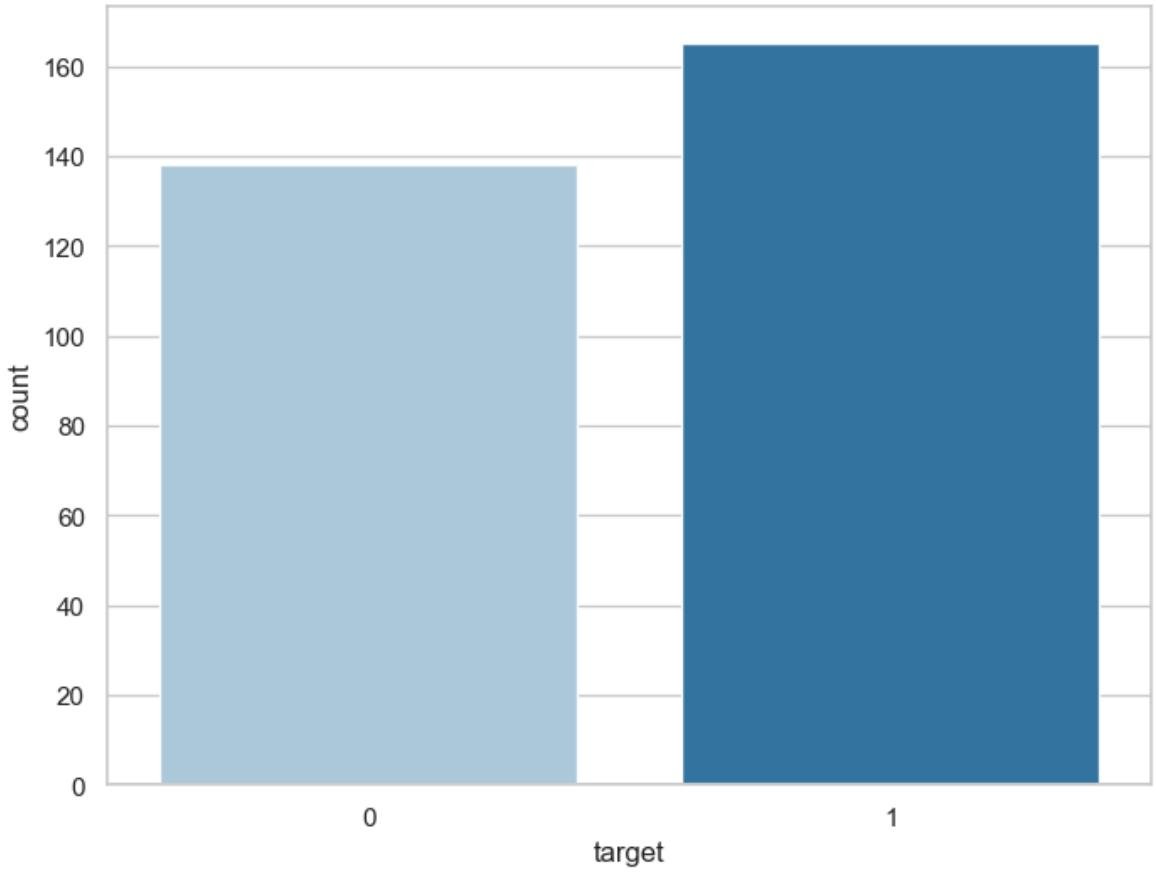

In [52]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",data=df,palette="paired")
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File D:\New folder\Lib\site-packages\seaborn\palettes.py:235, in color_palette(palette, n_colors, desat, as_cmap)
    233 try:
    234     # Perhaps a named matplotlib colormap?
--> 235     palette = mpl_palette(palette, n_colors, as_cmap=as_cmap)
    236 except (ValueError, KeyError):  # Error class changed in mpl36

File D:\New folder\Lib\site-packages\seaborn\palettes.py:406, in mpl_palette(name, n_colors, as_cmap)
    405 else:
--> 406     cmap = get_colormap(name)
    408 if name in MPL_QUAL_PALS:

File D:\New folder\Lib\site-packages\seaborn\_compat.py:62, in get_colormap(name)
     61 try:
---> 62     return mpl.colormaps[name]
     63 except AttributeError:

File D:\New folder\Lib\site-packages\matplotlib\cm.py:98, in ColormapRegistry.__getitem__(self, item)
     97 except KeyError:
---> 98     raise KeyError(f"{item!r} is not a known colormap name") from None

KeyError: "'paired' is not a known colormap name"

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
Cell In[52], line 2
      1 f,ax=plt.subplots(figsize=(8,6))
----> 2 ax=sns.countplot(x="target",data=df,palette="paired")
      3 plt.show()

File D:\New folder\Lib\site-packages\seaborn\categorical.py:2660, in countplot(data, x, y, hue, order, hue_order, orient, color, palette, saturation, fill, hue_norm, stat, width, dodge, gap, log_scale, native_scale, formatter, legend, ax, **kwargs)
   2657 palette, hue_order = p._hue_backcompat(color, palette, hue_order)
   2659 saturation = saturation if fill else 1
-> 2660 p.map_hue(palette=palette, order=hue_order, norm=hue_norm, saturation=saturation)
   2661 color = _default_color(ax.bar, hue, color, kwargs, saturation)
   2663 count_axis = {"x": "y", "y": "x"}[p.orient]

File D:\New folder\Lib\site-packages\seaborn\_base.py:838, in VectorPlotter.map_hue(self, palette, order, norm, saturation)
    837 def map_hue(self, palette=None, order=None, norm=None, saturation=1):
--> 838     mapping = HueMapping(self, palette, order, norm, saturation)
    839     self._hue_map = mapping

File D:\New folder\Lib\site-packages\seaborn\_base.py:150, in HueMapping.__init__(self, plotter, palette, order, norm, saturation)
    147 elif map_type == "categorical":
    149     cmap = norm = None
--> 150     levels, lookup_table = self.categorical_mapping(
    151         data, palette, order,
    152     )
    154 # --- Option 3: datetime mapping
```

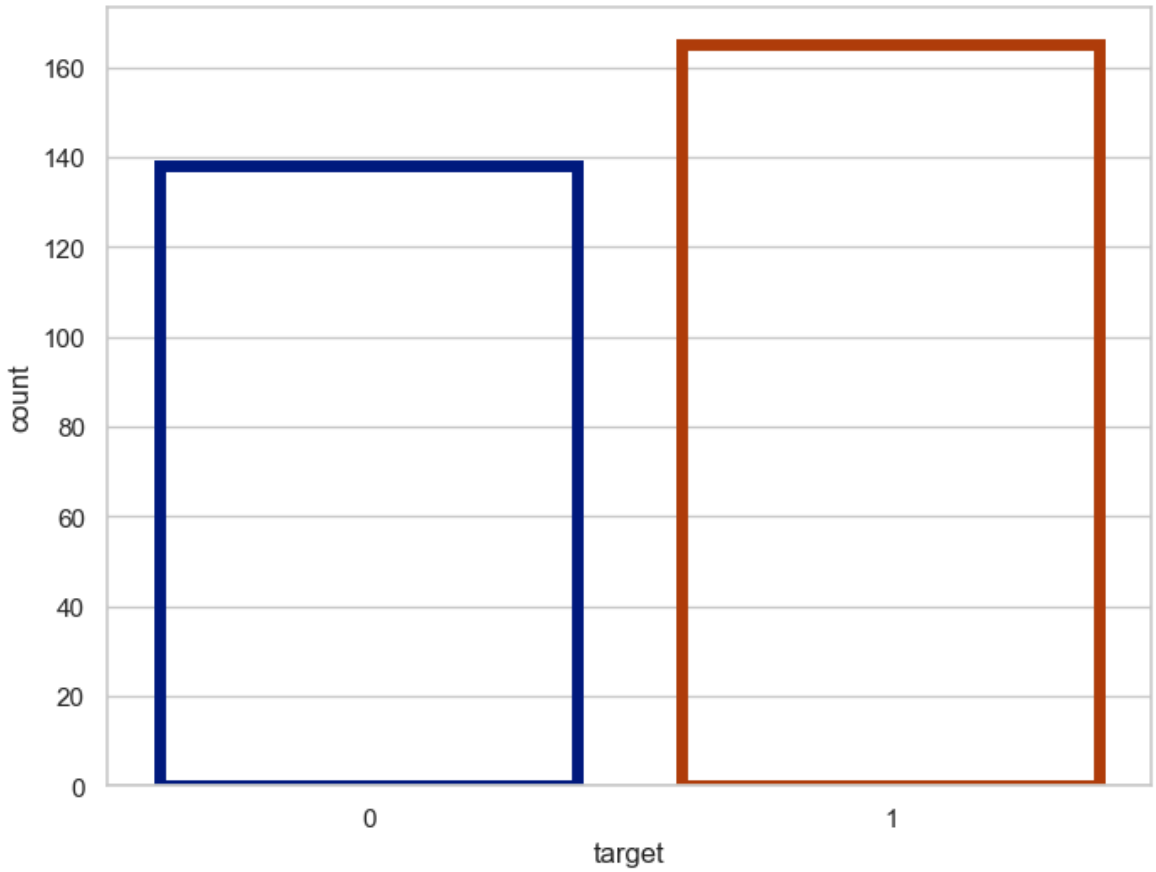
```
    155
    156 else:
    157     # TODO this needs actual implementation
    158     cmap = norm = None

File D:\New folder\Lib\site-packages\seaborn\_base.py:248, in HueMapping.categori
cal_mapping(self, data, palette, order)
    246         colors = self._check_list_length(levels, palette, "palette")
    247     else:
--> 248         colors = color_palette(palette, n_colors)
    250     lookup_table = dict(zip(levels, colors))
    252 return levels, lookup_table

File D:\New folder\Lib\site-packages\seaborn\palettes.py:237, in color_palette(pa
lette, n_colors, desat, as_cmap)
    235             palette = mpl_palette(palette, n_colors, as_cmap=as_cmap)
    236         except (ValueError, KeyError):  # Error class changed in mpl36
--> 237             raise ValueError(f"{palette!r} is not a valid palette name")
    239 if desat is not None:
    240     palette = [desaturate(c, desat) for c in palette]

ValueError: 'paired' is not a valid palette name
```

In [53]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",data=df,palette="Paired")
plt.show()
```



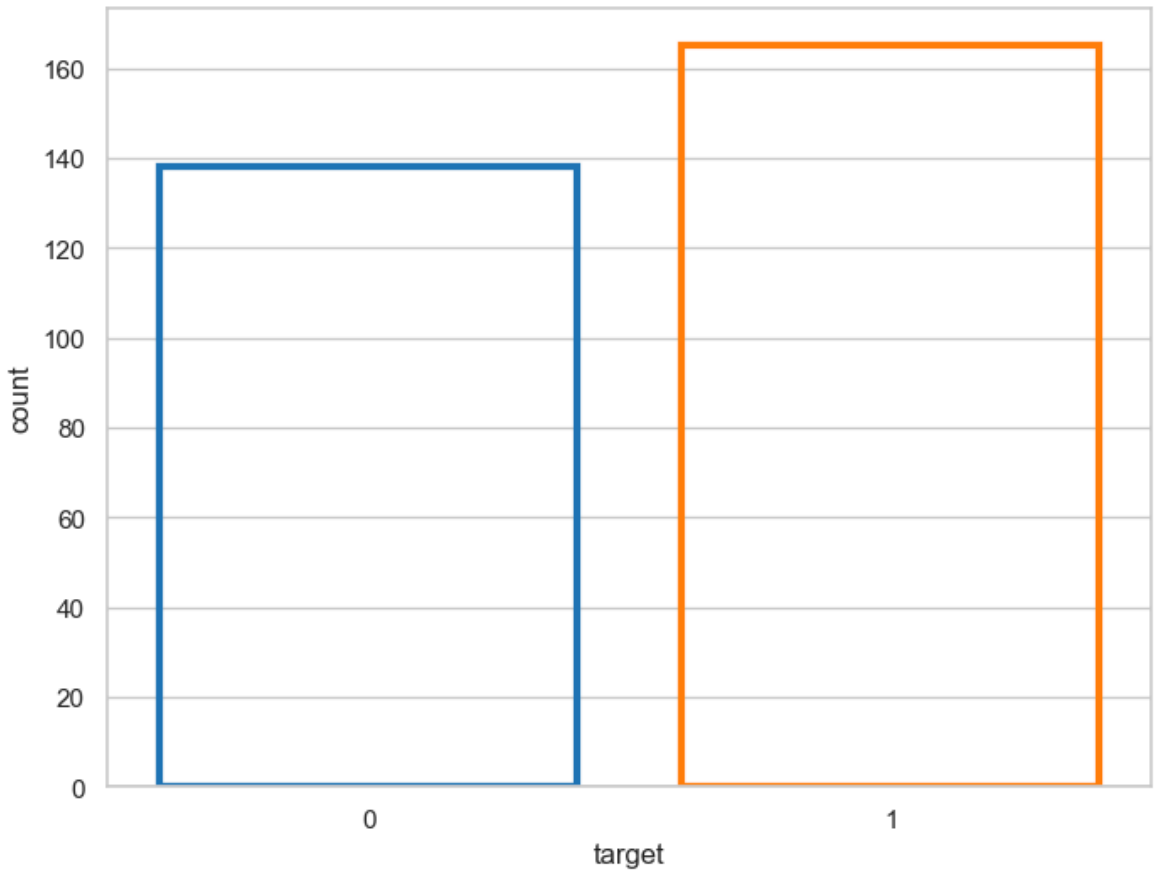



In [56]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x="target",data=df,facecolor=(0,0,0,0),linewidth=5,edgecolor=sn
plt.show()
```

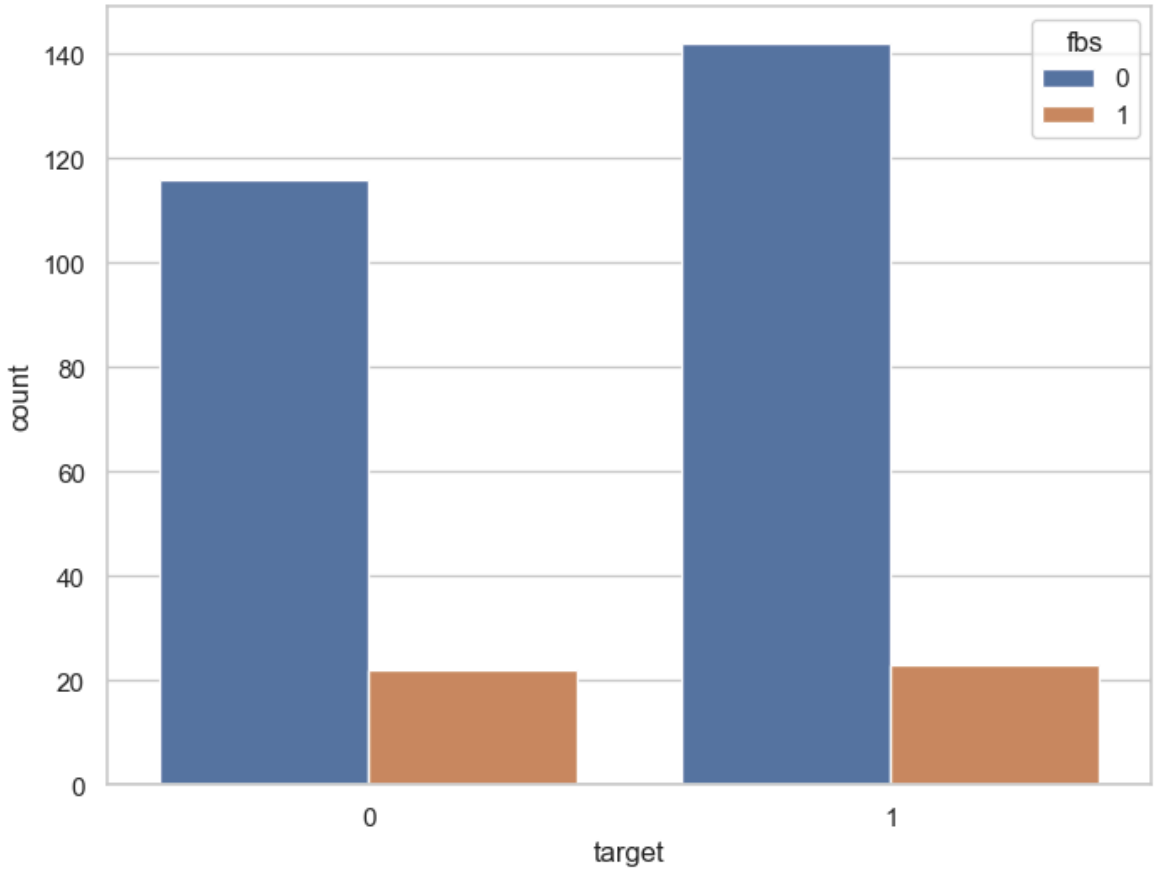



In [57]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x='target',data=df,facecolor=(0,0,0,0),linewidth=3,edgecolor=sn
```

```
plt.show()
```




In [58]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='target',hue='fbs')
plt.show()
```



In [59]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='target',hue='exang',palette('Set3'))
plt.show()
```

```
  Cell In[59], line 2
    ax=sns.countplot(data=df,x='target',hue='exang',palette('Set3'))
                                                                   ^
SyntaxError: positional argument follows keyword argument
```

In [60]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='target',hue='exang',palette('Set3')
plt.show()
```

```
  Cell In[60], line 2
    ax=sns.countplot(data=df,x='target',hue='exang',palette('Set3')
                     ^
SyntaxError: '(' was never closed
```

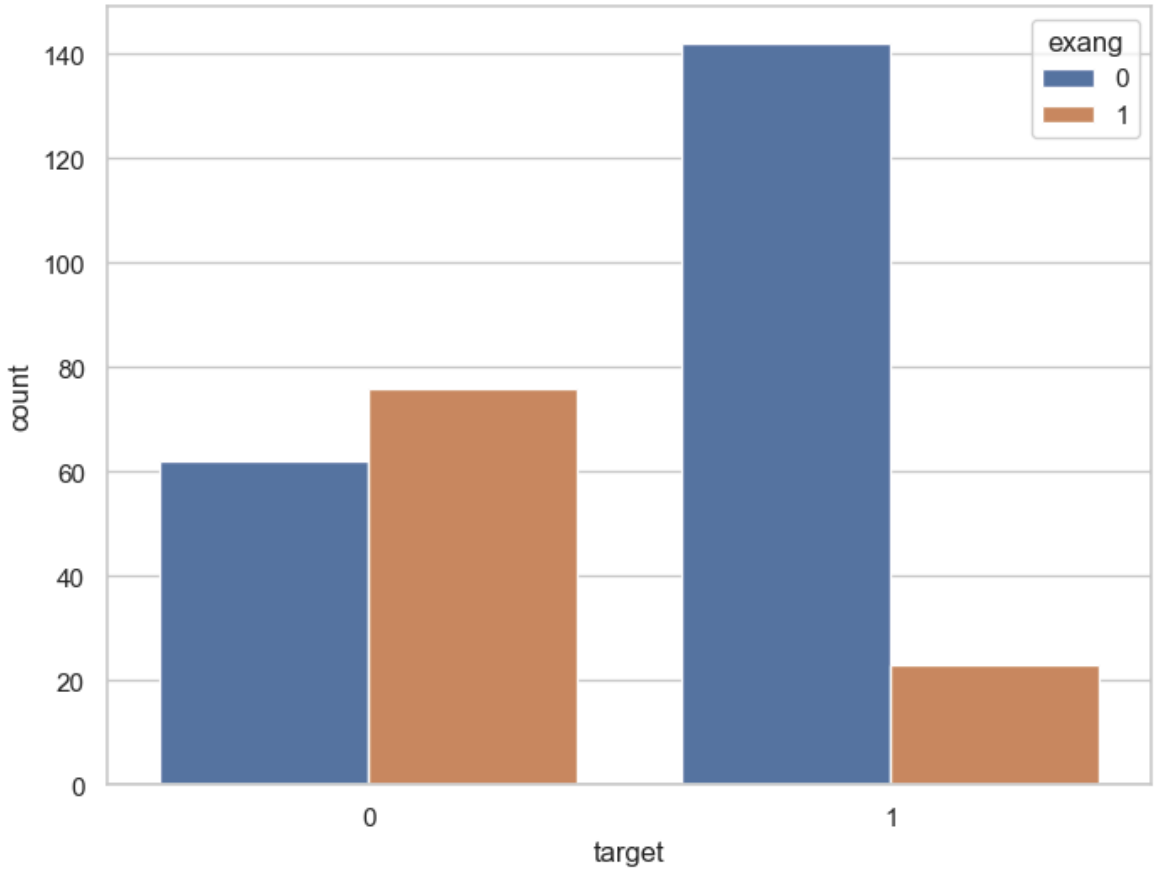
In [61]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='target',hue='exang',palette('Set3'))
plt.show()
```

```
  Cell In[61], line 2
    ax=sns.countplot(data=df,x='target',hue='exang',palette('Set3'))
                                                                   ^
SyntaxError: positional argument follows keyword argument
```

In [62]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='target',hue='exang',sns.color_palette('Set3'))
plt.show()
```

```
  Cell In[62], line 2
    ax=sns.countplot(data=df,x='target',hue='exang',sns.color_palette('Set3'))
                                                                             ^
SyntaxError: positional argument follows keyword argument
```

In [63]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='target',hue='exang')
plt.show()
```




In [64]:
```
correlation = df.corr()
```

In [65]:
```
correlation['target'].sort_values(ascending=False)
```

Out[65]:
```
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 [66]:
```
df['cp'].nunique()
```

Out[66]:
```
4
```

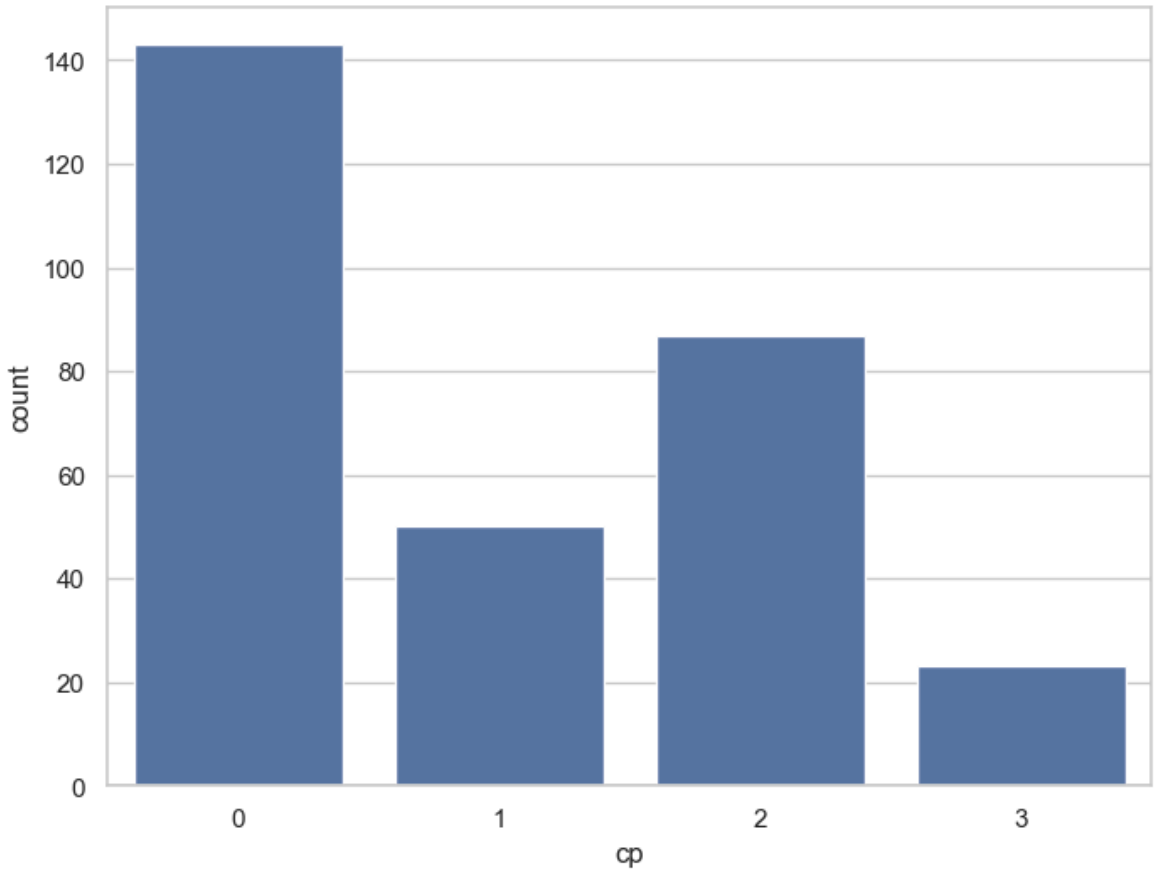
In [67]:
```
df['cp'].unique()
```

Out[67]:
```
array([3, 2, 1, 0])
```

In [68]:
```
df['cp'].value_counts()
```

Out[68]:
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

In [69]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='cp')
plt.show()
```



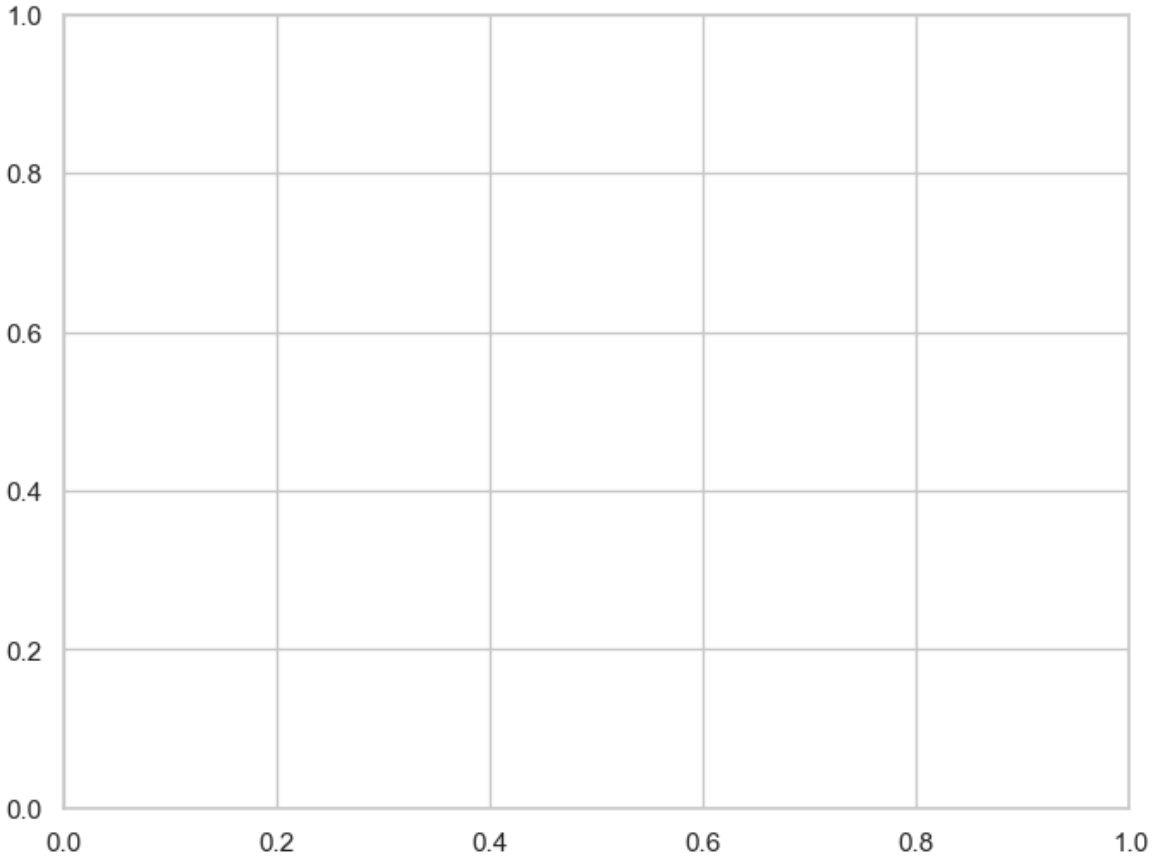

In [70]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,hue='cp')
plt.show()
```

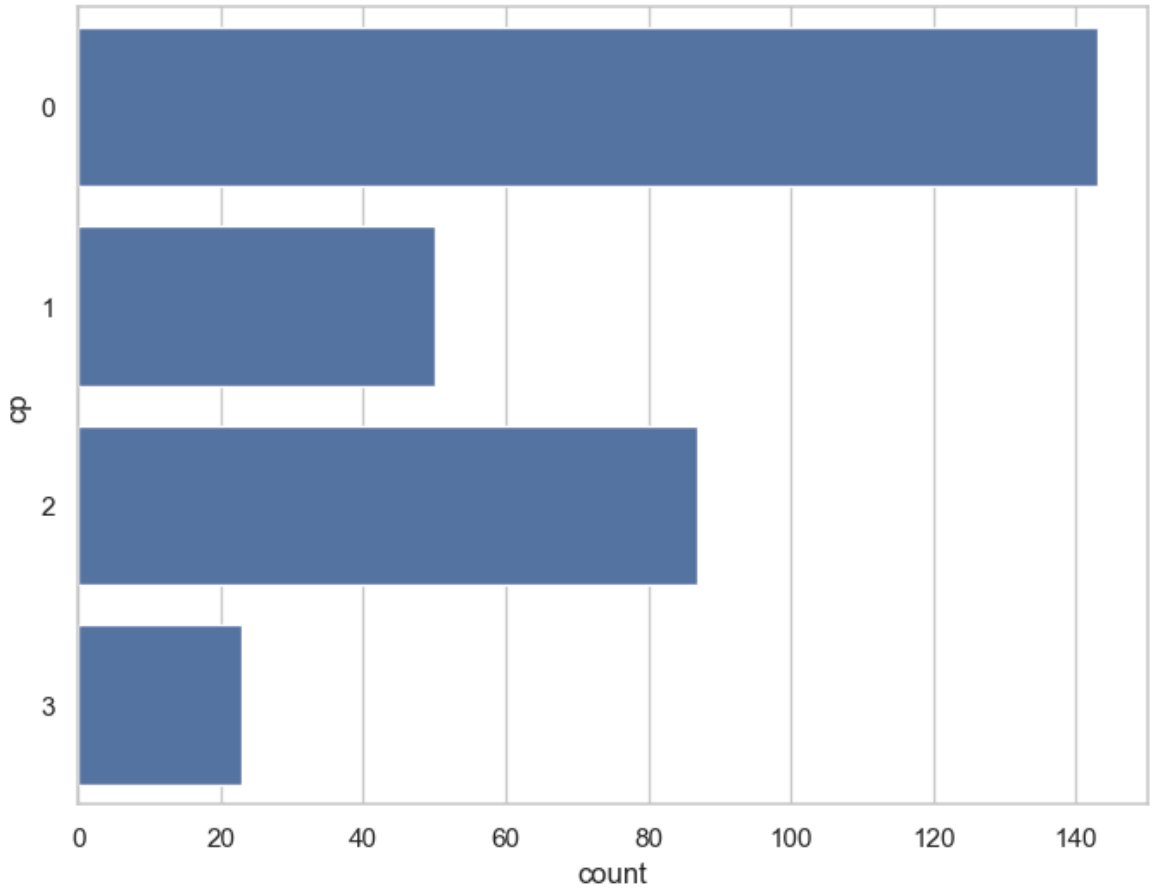
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[70], line 2
      1 f,ax=plt.subplots(figsize=(8,6))
----> 2 ax=sns.countplot(data=df,hue='cp')
      3 plt.show()

File D:\New folder\Lib\site-packages\seaborn\categorical.py:2631, in countplot(data, x, y, hue, order, hue_order, orient, color, palette, saturation, fill, hue_norm, stat, width, dodge, gap, log_scale, native_scale, formatter, legend, ax, **kwargs)
   2628 elif x is not None and y is not None:
   2629     raise TypeError("Cannot pass values for both `x` and `y`.")
-> 2631 p = _CategoricalAggPlotter(
   2632     data=data,
   2633     variables=dict(x=x, y=y, hue=hue),
   2634     order=order,
   2635     orient=orient,
   2636     color=color,
   2637     legend=legend,
   2638 )
   2640 if ax is None:
   2641     ax = plt.gca()

File D:\New folder\Lib\site-packages\seaborn\categorical.py:67, in _CategoricalPlotter.__init__(self, data, variables, order, orient, require_numeric, color, legend)
     56 def __init__(
     57     self,
     58     data=None,
   (...)
     64     legend="auto",
     65 ):
---> 67     super().__init__(data=data, variables=variables)
     69     # This method takes care of some bookkeeping that is necessary because the
     70     # original categorical plots (prior to the 2021 refactor) had some rules that
     71     # don't fit exactly into VectorPlotter logic. It may be wise to have a second
   (...)
     76     # default VectorPlotter rules. If we do decide to make orient part of the
     77     # _base variable assignment, we'll want to figure out how to express that.
     78     if self.input_format == "wide" and orient in ["h", "y"]:

File D:\New folder\Lib\site-packages\seaborn\_base.py:634, in VectorPlotter.__init__(self, data, variables)
    629 # var_ordered is relevant only for categorical axis variables, and may
    630 # be better handled by an internal axis information object that tracks
    631 # such information and is set up by the scale_* methods. The analogous
    632 # information for numeric axes would be information about log scales.
    633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
    636 # TODO Lots of tests assume that these are called to initialize the
    637 # mappings to default values on class initialization. I'd prefer to
    638 # move away from that and only have a mapping when explicitly called.
    639 for var in ["hue", "size", "style"]:
```

```
File D:\New folder\Lib\site-packages\seaborn\_base.py:673, in VectorPlotter.assig
n_variables(self, data, variables)
    671 if x is None and y is None:
    672     self.input_format = "wide"
--> 673     frame, names = self._assign_variables_wideform(data, **variables)
    674 else:
    675     # When dealing with long-form input, use the newer PlotData
    676     # object (internal but introduced for the objects interface)
    677     # to centralize / standardize data consumption logic.
    678     self.input_format = "long"

File D:\New folder\Lib\site-packages\seaborn\_base.py:723, in VectorPlotter._assi
gn_variables_wideform(self, data, **kwargs)
    721     err = f"The following variable{s} cannot be assigned with wide-form d
ata: "
    722     err += ", ".join(f"`{v}`" for v in assigned)
--> 723     raise ValueError(err)
    725 # Determine if the data object actually has any data in it
    726 empty = data is None or not len(data)

ValueError: The following variable cannot be assigned with wide-form data: `hue`
```

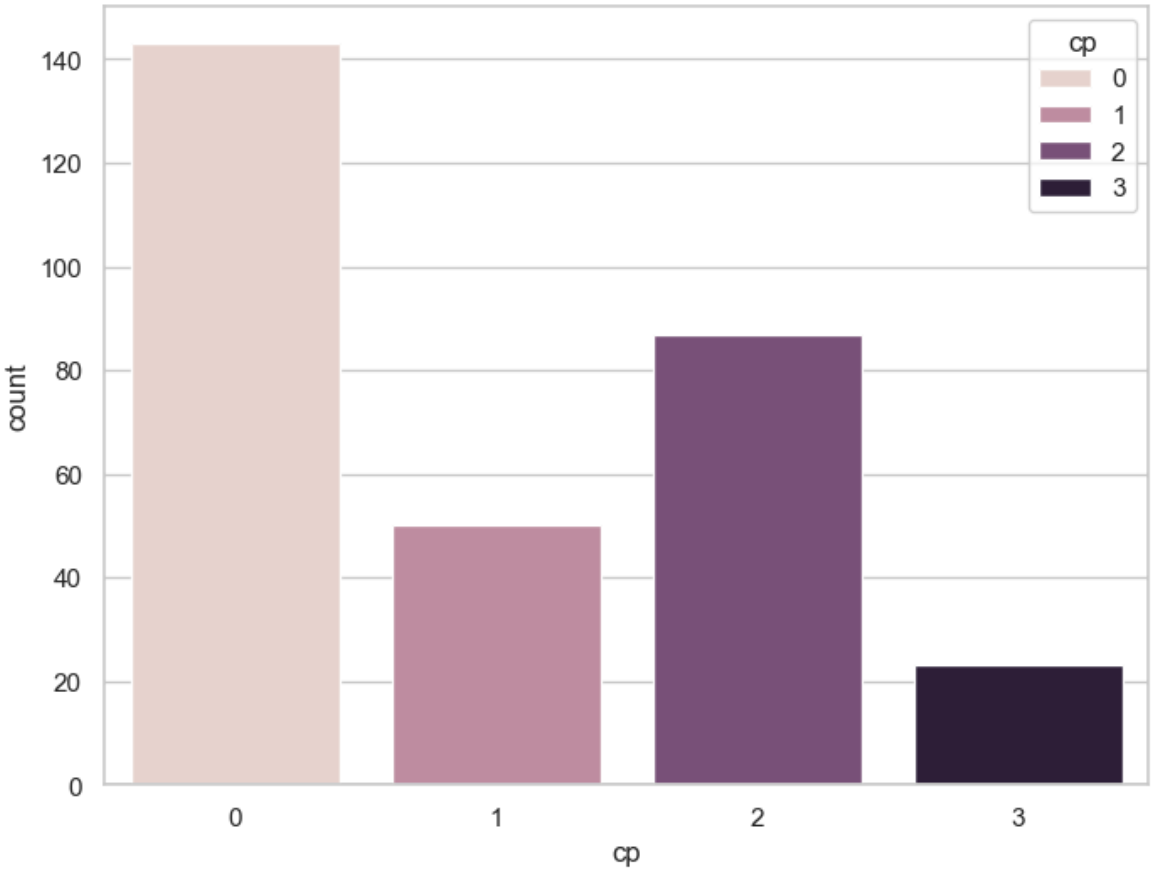
In [ ]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='cp')
plt.show()
```

In [71]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,y='cp')
plt.show()
```

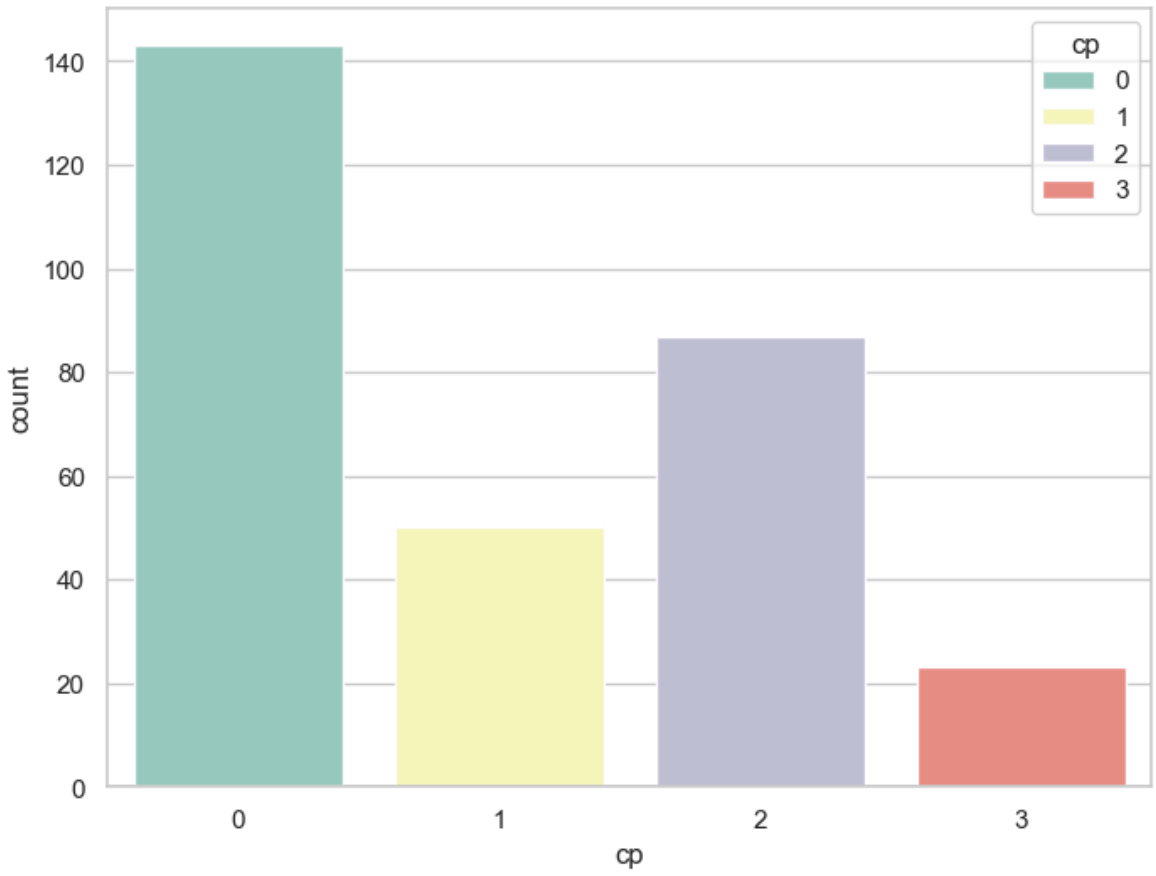





In [72]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='cp',hue='cp')
plt.show()
```



In [73]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(data=df,x='cp',hue='cp',palette='Set3')
plt.show()
```



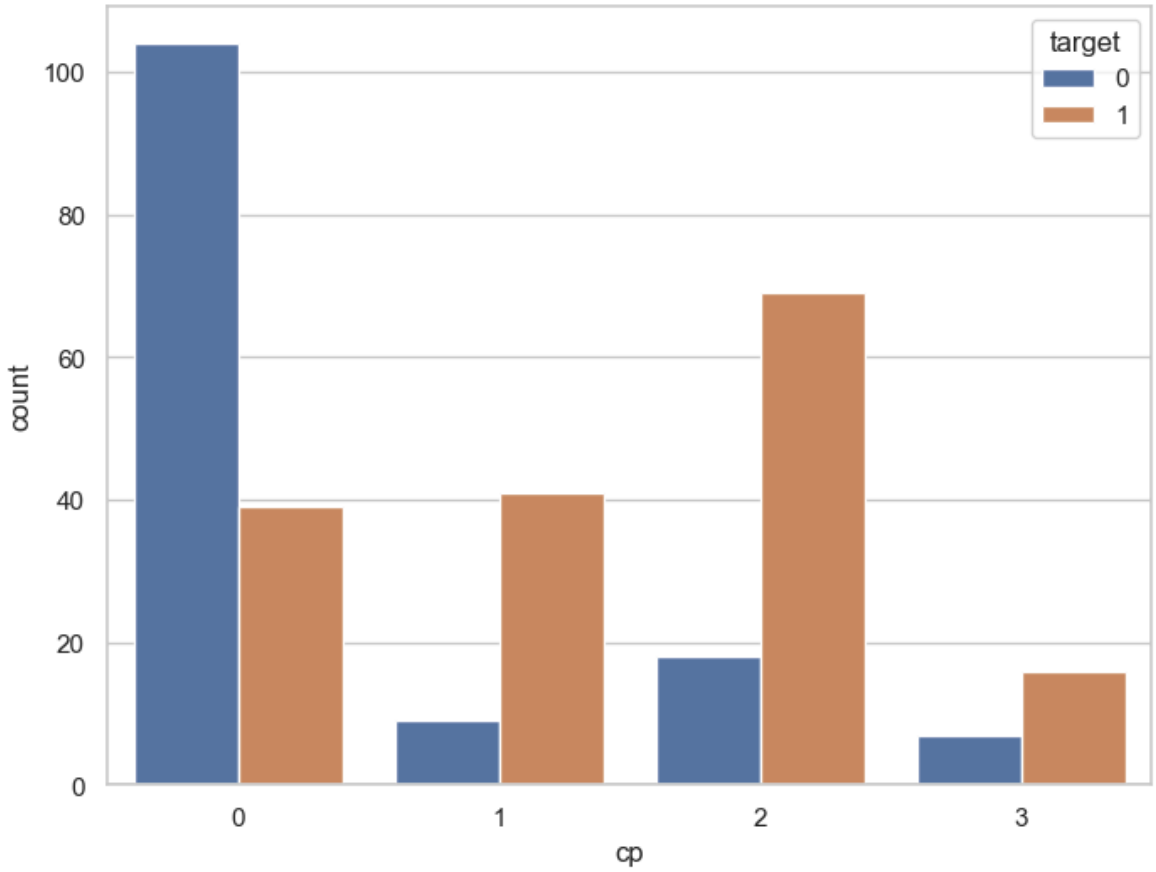

In [74]:
```
df['cp'].value_counts()
```

Out[74]:
```
cp
0    143
2     87
1     50
3     23
Name: count, dtype: int64
```

In [75]:
```
df.groupby('cp')['target'].value_counts()
```

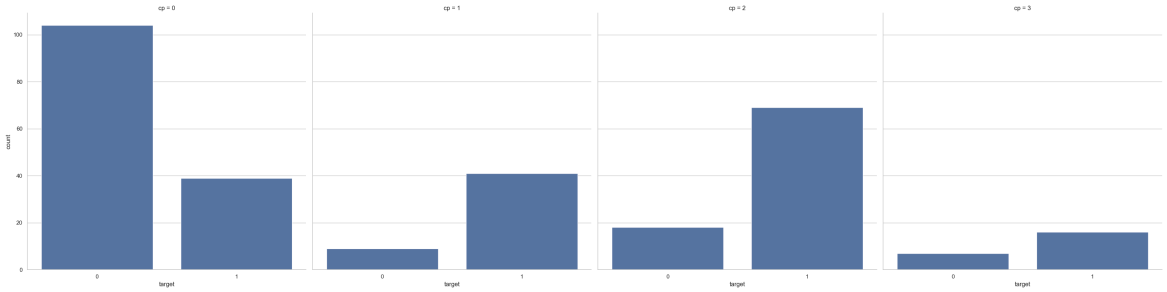
Out[75]:
```
cp  target
0   0         104
    1          39
1   1          41
    0           9
2   1          69
    0          18
3   1          16
    0           7
Name: count, dtype: int64
```

In [76]:
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.countplot(x='cp',hue='target',data=df)
plt.show()
```

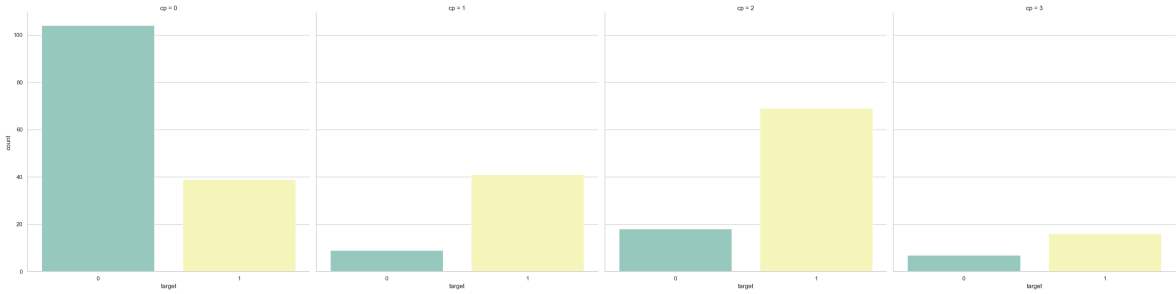



In [77]:
```
ax=sns.catplot(x="target",col='cp',data=df,kind='count',height=8,aspect=1)
```

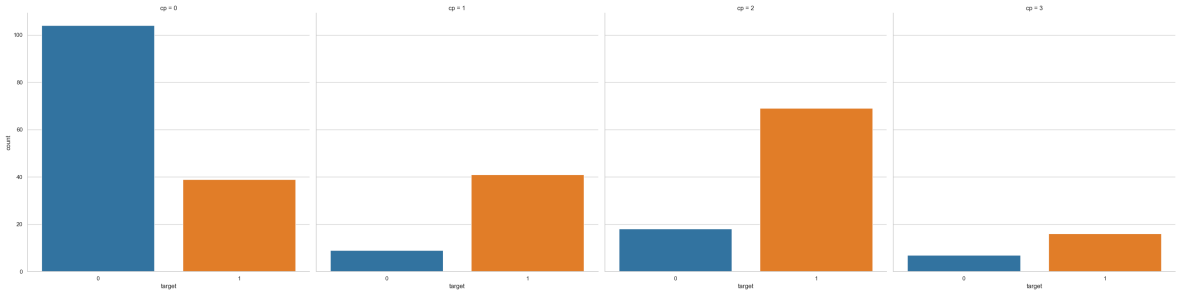
In [78]:
```
plt.show()
```



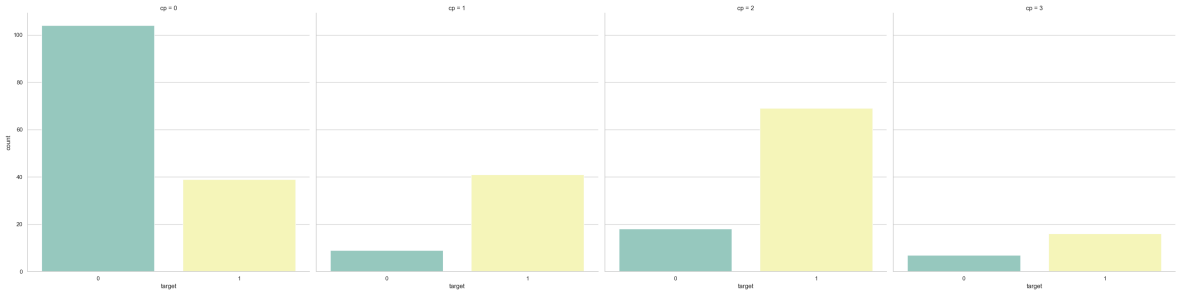
In [79]:
```
ax=sns.catplot(x="target",col='cp',data=df,kind='count',height=8,aspect=1,palett
plt.show()
```



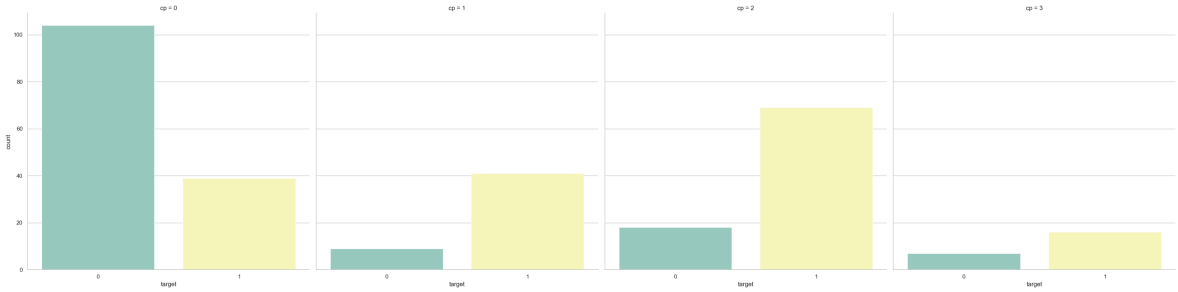
In [80]:
```
ax=sns.catplot(x="target",col='cp',data=df,kind='count',height=8,aspect=1,palett
plt.show()
```



In [81]:
```
ax=sns.catplot(x="target",col='cp',data=df,kind='count',height=8,palette='Set3')
plt.show()
```



In [82]:
```
ax=sns.catplot(x="target",col='cp',data=df,kind='count',aspect=1,height=8,palett
plt.show()
```



In [83]:
```
df['thalach'].nunique()
```

Out[83]: 91

In [84]:
```
df['thalach'].unique()
```

Out[84]:
```
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])
```

In [85]:
```
df['thalach'].value_count()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_40912\213323956.py in ?()
----> 1 df['thalach'].value_count()

D:\New folder\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   6295             and name not in self._accessors
   6296             and self._info_axis._can_hold_identifiers_and_holds_name(nam
e)
   6297         ):
   6298             return self[name]
-> 6299         return object.__getattribute__(self, name)

AttributeError: 'Series' object has no attribute 'value_count'
```

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

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[86], line 1
----> 1 f,ax=plt.suplots(figsize=(10,6))
      2 ax=sns.distplot(data=df,x='thalach',bins=10)
      3 plt.show()

AttributeError: module 'matplotlib.pyplot' has no attribute 'suplots'
```

In [87]:
```
f,ax=plt.subplots(figsize=(10,6))
ax=sns.distplot(data=df,x='thalach',bins=10)
plt.show()
```

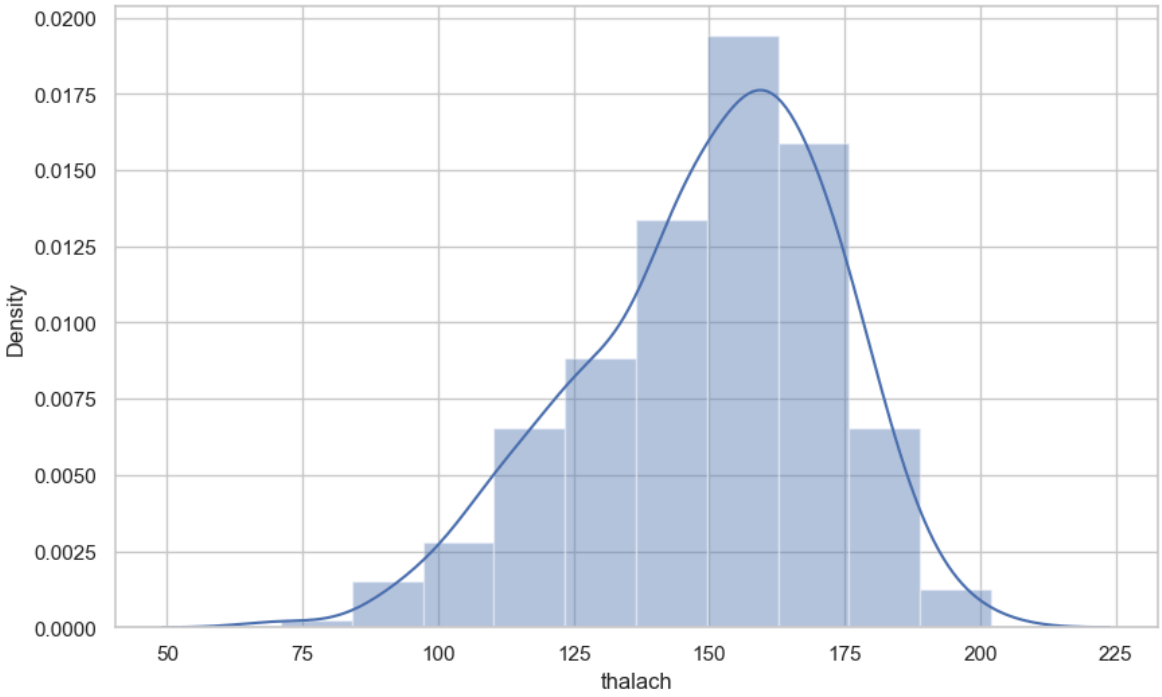
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[87], line 2
      1 f,ax=plt.subplots(figsize=(10,6))
----> 2 ax=sns.distplot(data=df,x='thalach',bins=10)
      3 plt.show()

TypeError: distplot() got an unexpected keyword argument 'data'
```

In [88]:
```
f,ax=plt.subplots(figsize=(10,6))
ax=sns.distplot(x='thalach',bins=10)
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[88], line 2
      1 f,ax=plt.subplots(figsize=(10,6))
----> 2 ax=sns.distplot(x='thalach',bins=10)
      3 plt.show()

File D:\New folder\Lib\site-packages\seaborn\distributions.py:2443, in distplot
(a, bins, hist, kde, rug, fit, hist_kws, kde_kws, rug_kws, fit_kws, color, vertic
al, norm_hist, axlabel, label, ax, x)
   2440     a = x
   2442 # Make a a 1-d float array
-> 2443 a = np.asarray(a, float)
   2444 if a.ndim > 1:
   2445     a = a.squeeze()

ValueError: could not convert string to float: 'thalach'
```

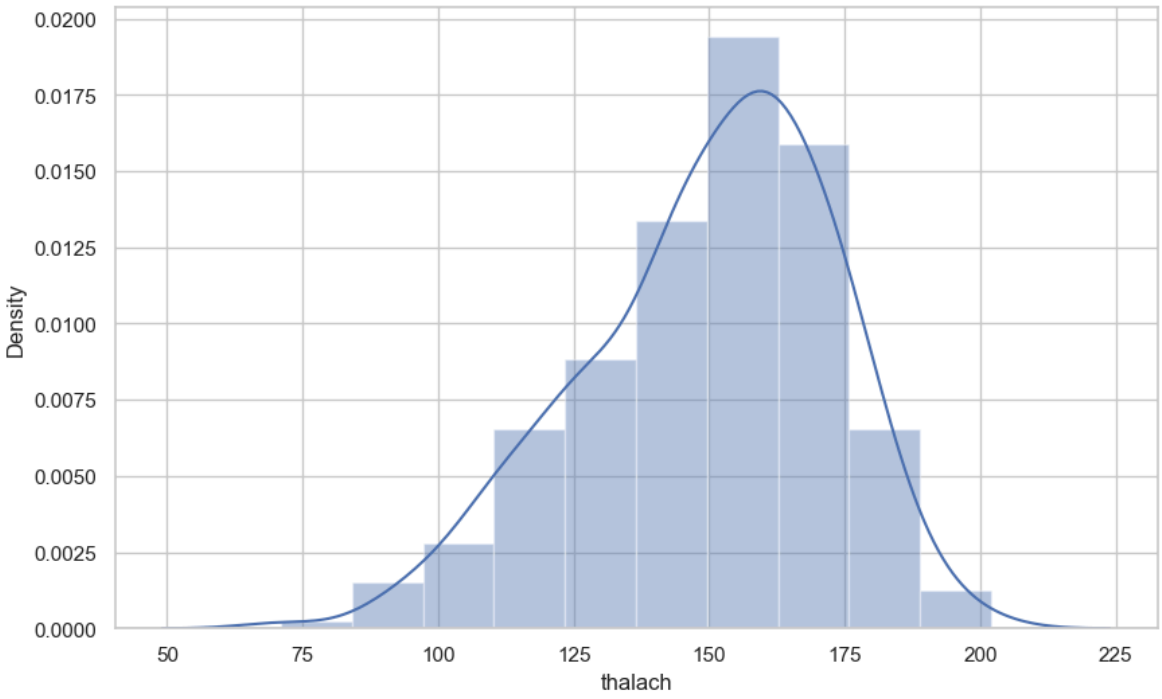
In [112…

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

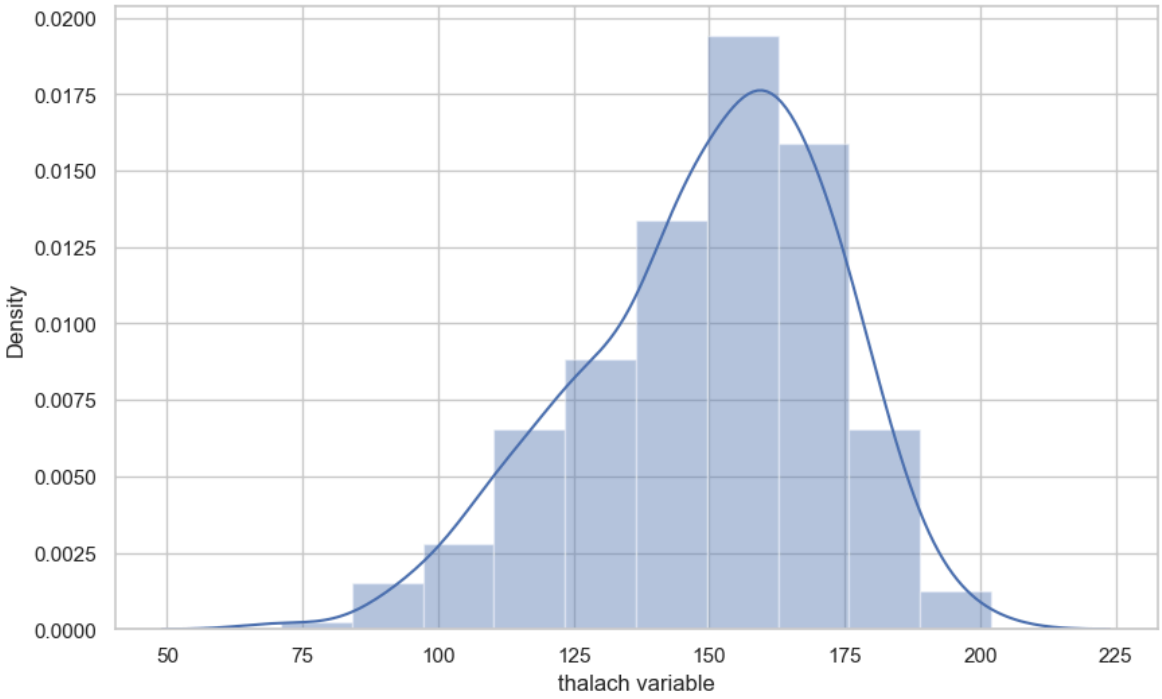



In [90]:

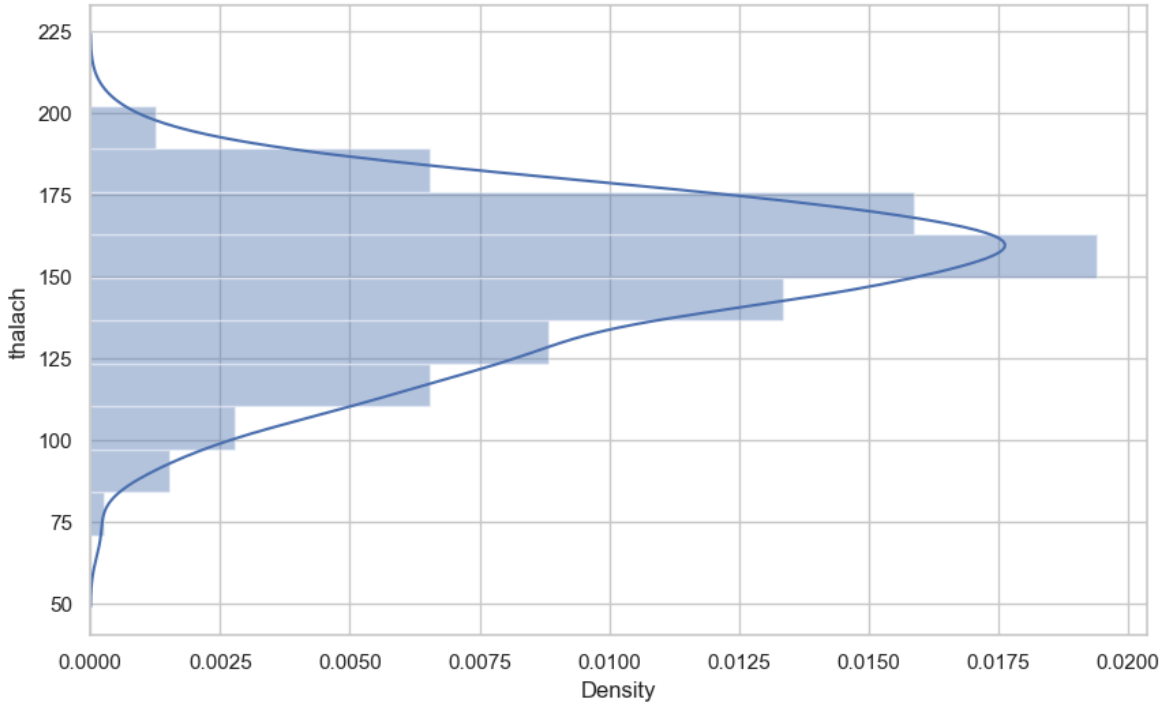
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
ax=sns.distplot(x,bins=10)
plt.show()
```



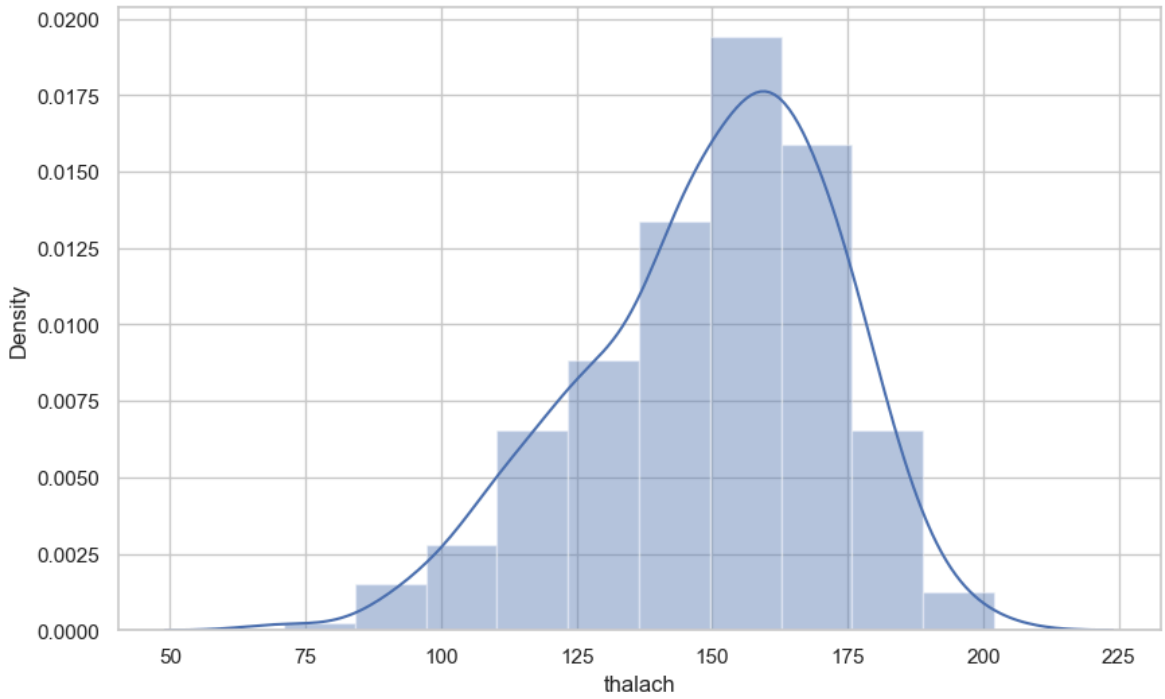

In [91]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name="thalach variable")
ax=sns.distplot(x,bins=10)
plt.show()
```



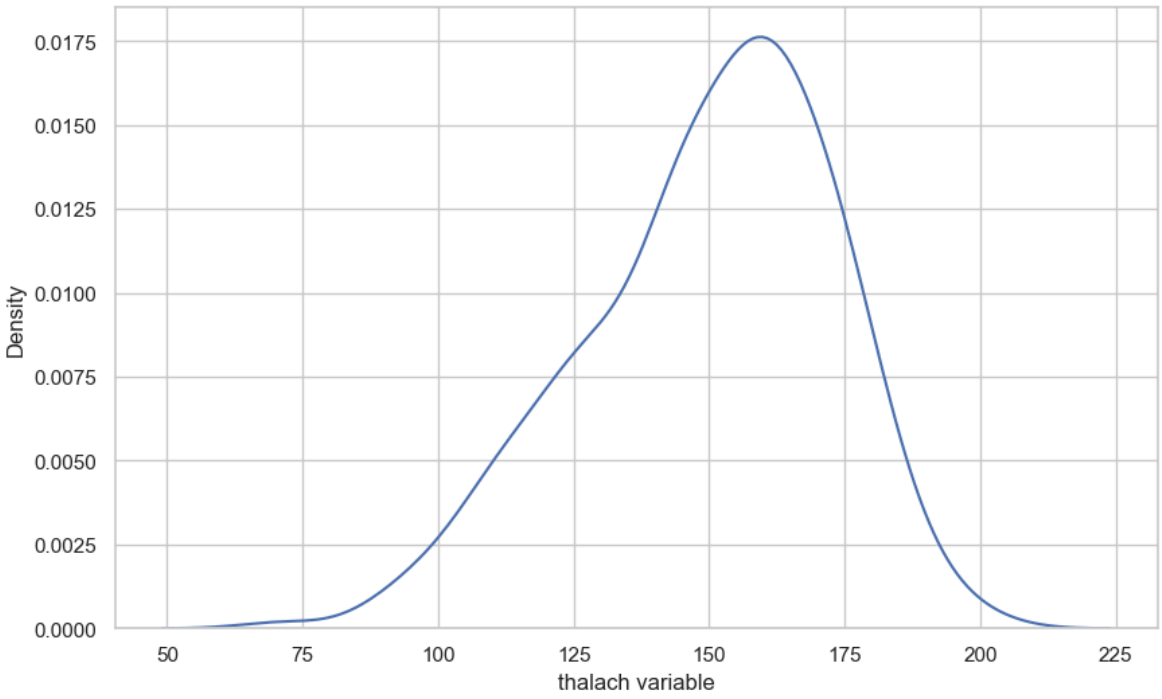

In [92]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
ax=sns.distplot(x,bins=10,vertical=True)
plt.show()
```



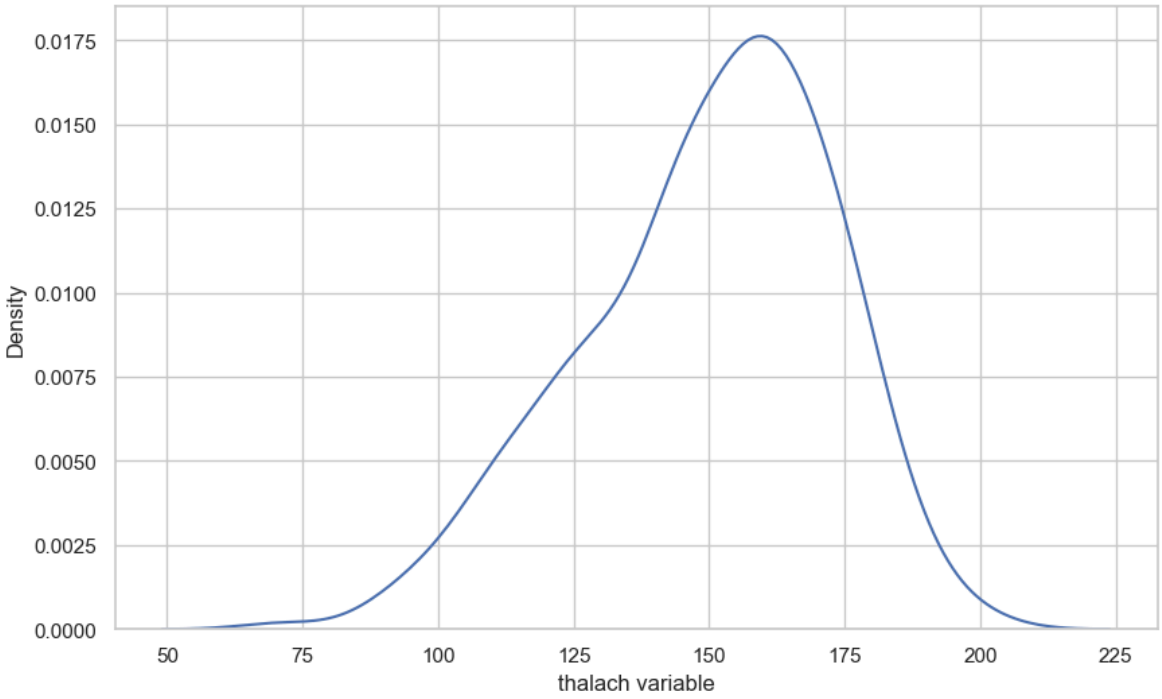

In [93]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
ax=sns.distplot(x,bins=10)
plt.show()
```



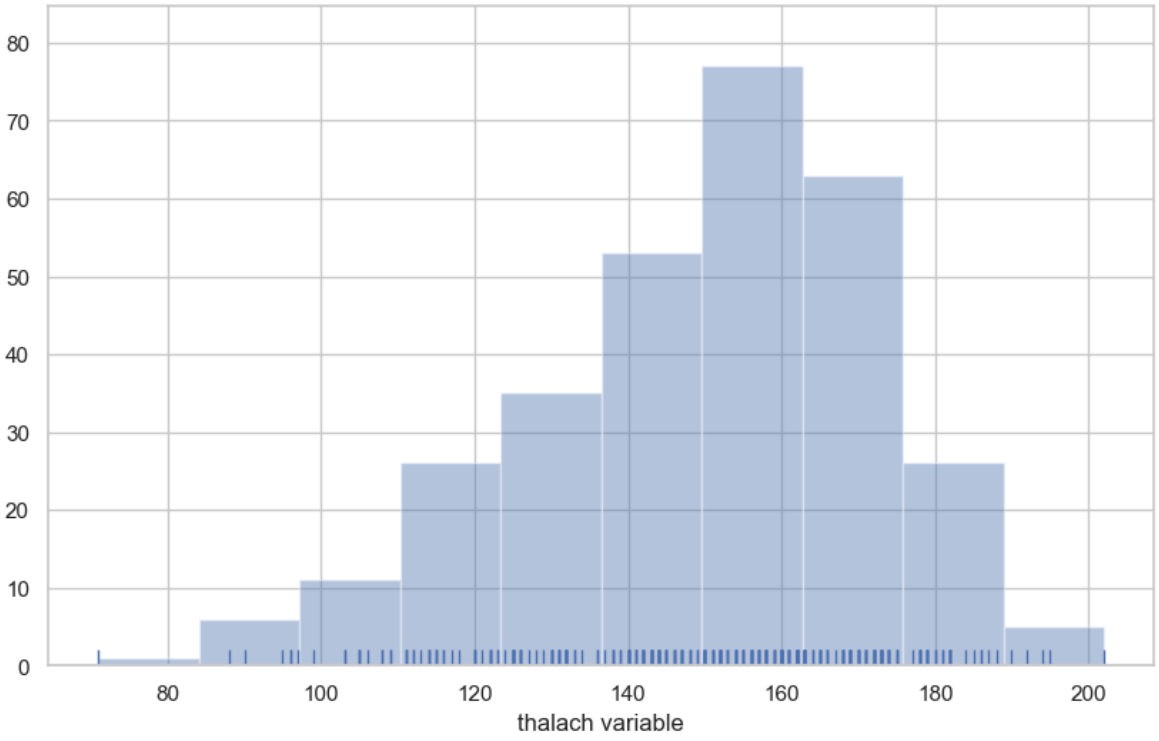

In [94]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name='thalach variable')
ax=sns.kdeplot(x)
plt.show()
```



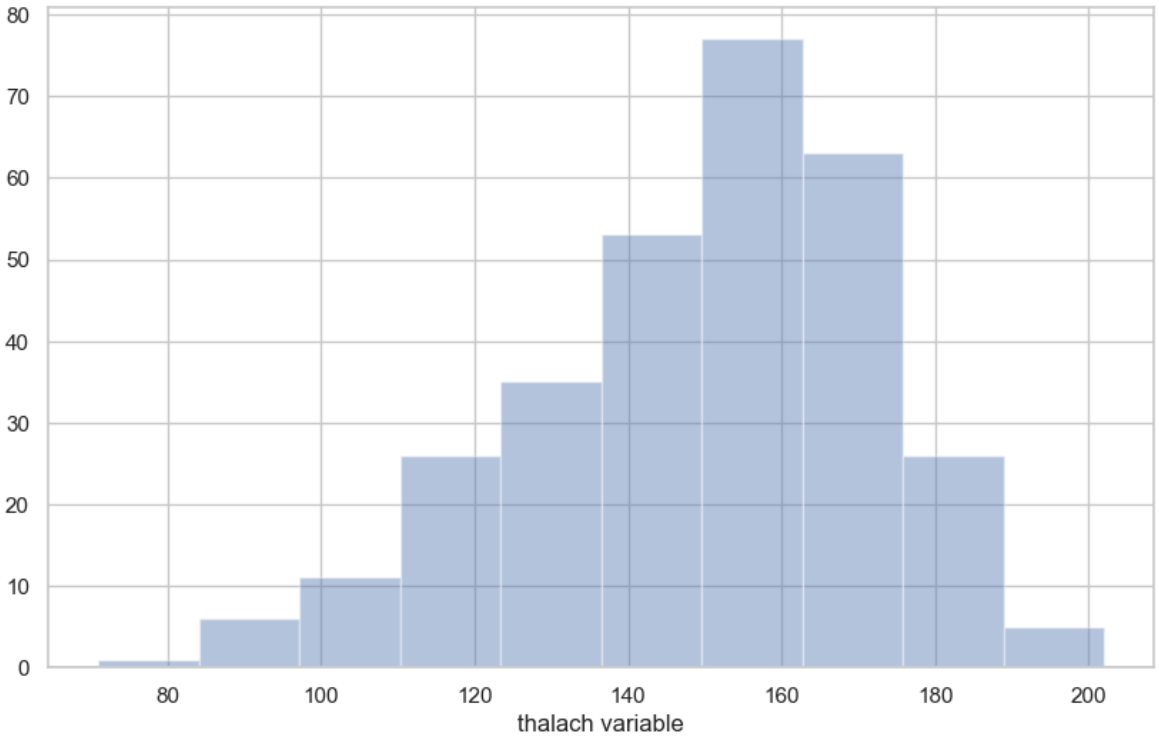

In [95]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name="thalach variable")
ax=sns.kdeplot(x)
plt.show()
```




In [96]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name='thalach variable')
ax=sns.distplot(x,kde=False,rug=True,bins=10)
plt.show()
```



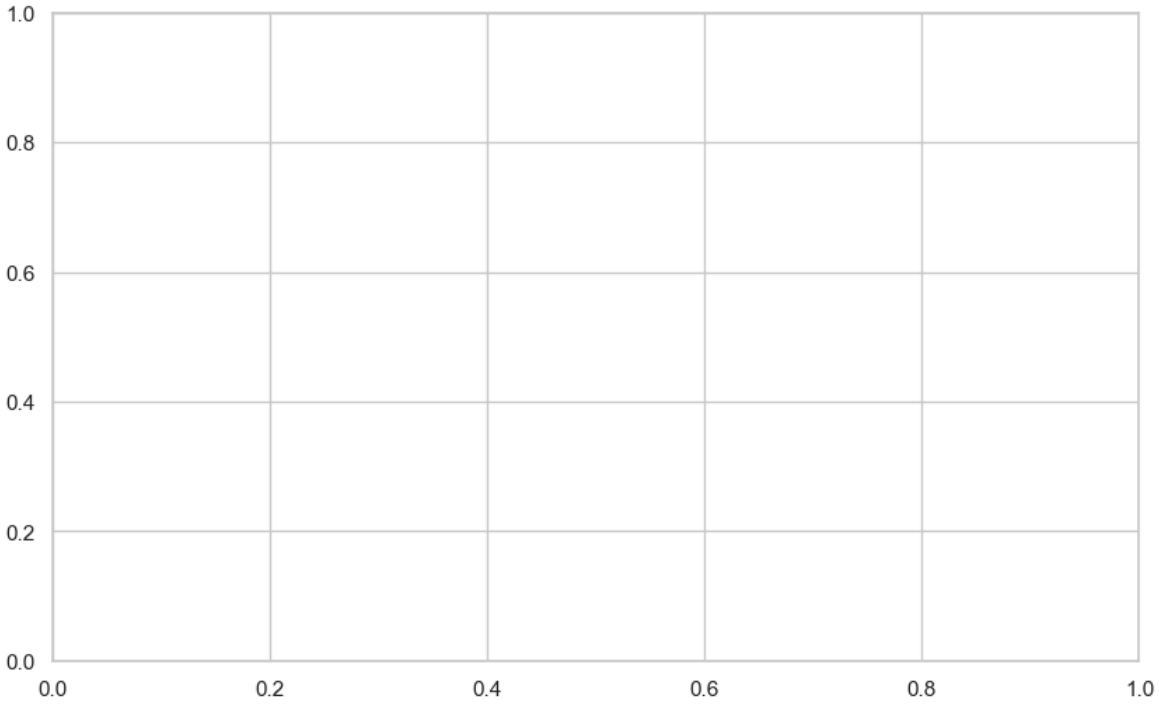

In [97]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name='thalach variable')
ax=sns.distplot(x,kde=False,rug=False,bins=10)
plt.show()
```

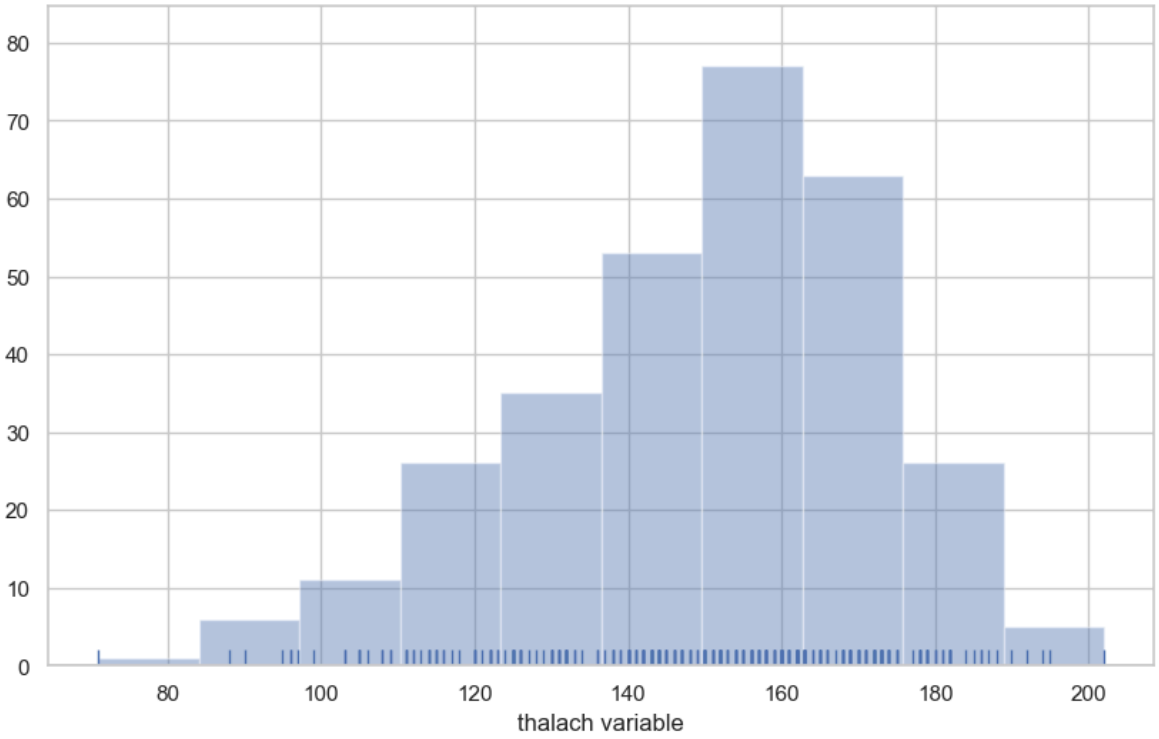



In [98]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name='thalach variable')
ax=sns.distplot(x,kde=False,reg=True,bins=10)
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[98], line 4
      2 x=df['thalach']
      3 x=pd.Series(x,name='thalach variable')
----> 4 ax=sns.distplot(x,kde=False,reg=True,bins=10)
      5 plt.show()

TypeError: distplot() got an unexpected keyword argument 'reg'. Did you mean 'ru
g'?
```

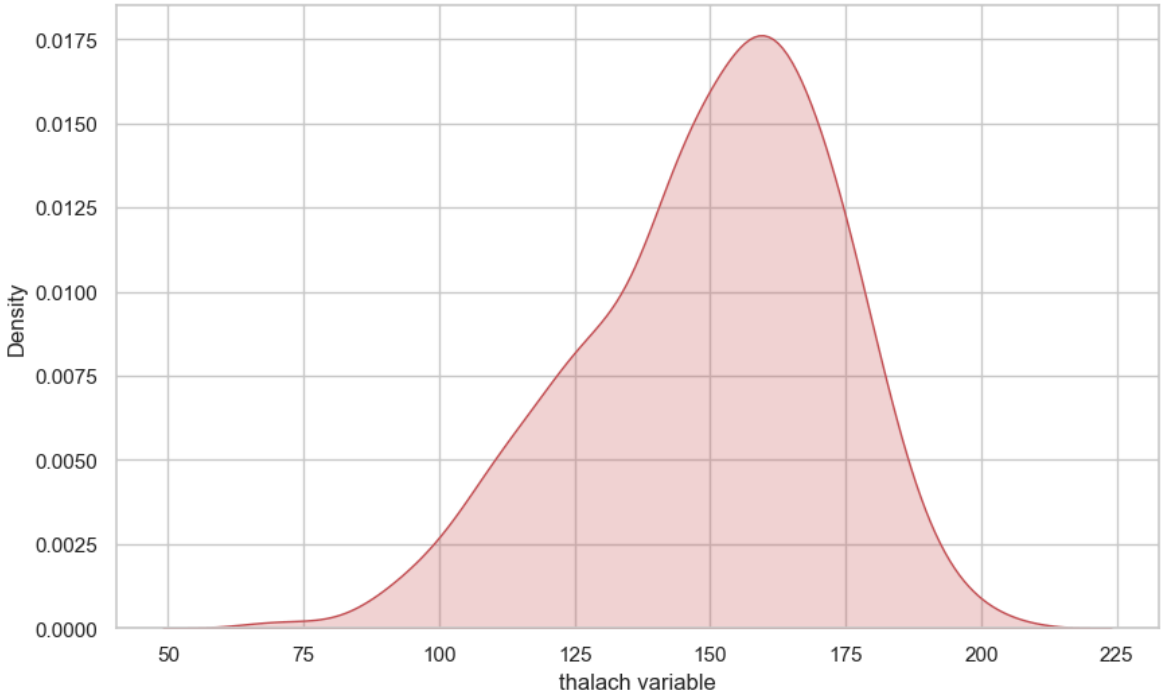
In [99]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name='thalach variable')
ax=sns.distplot(x,kde=False,rug=True,bins=10)
plt.show()
```





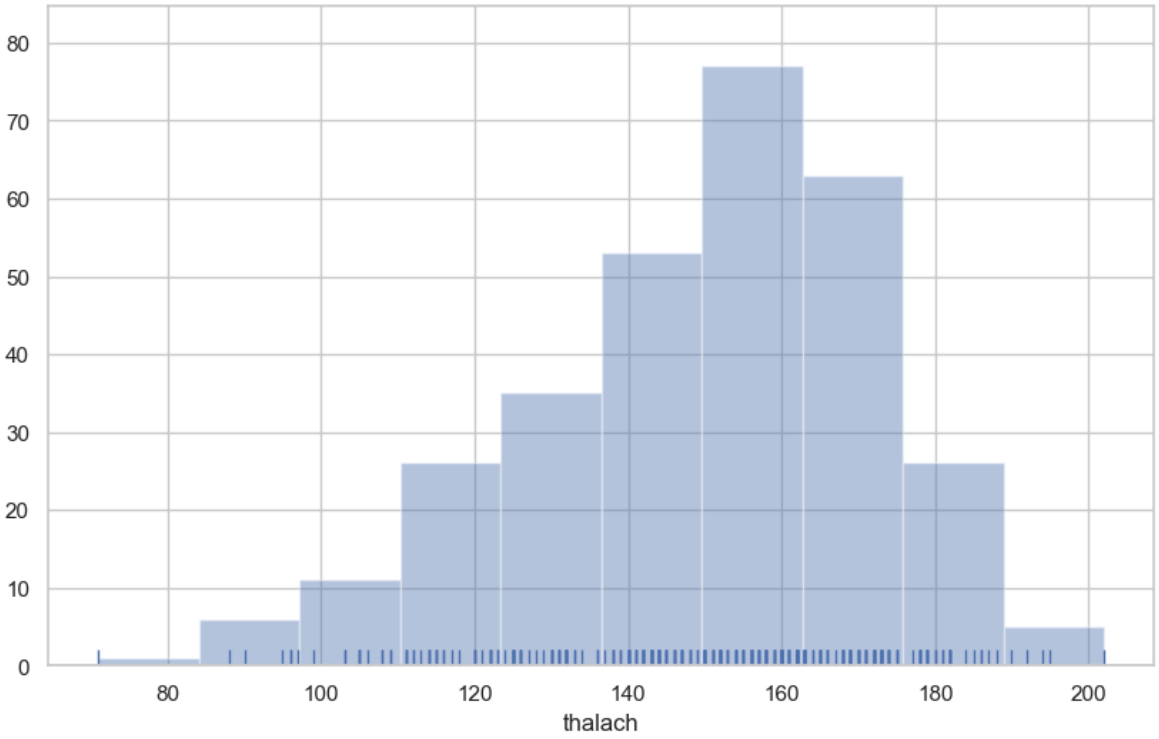

In [100…

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

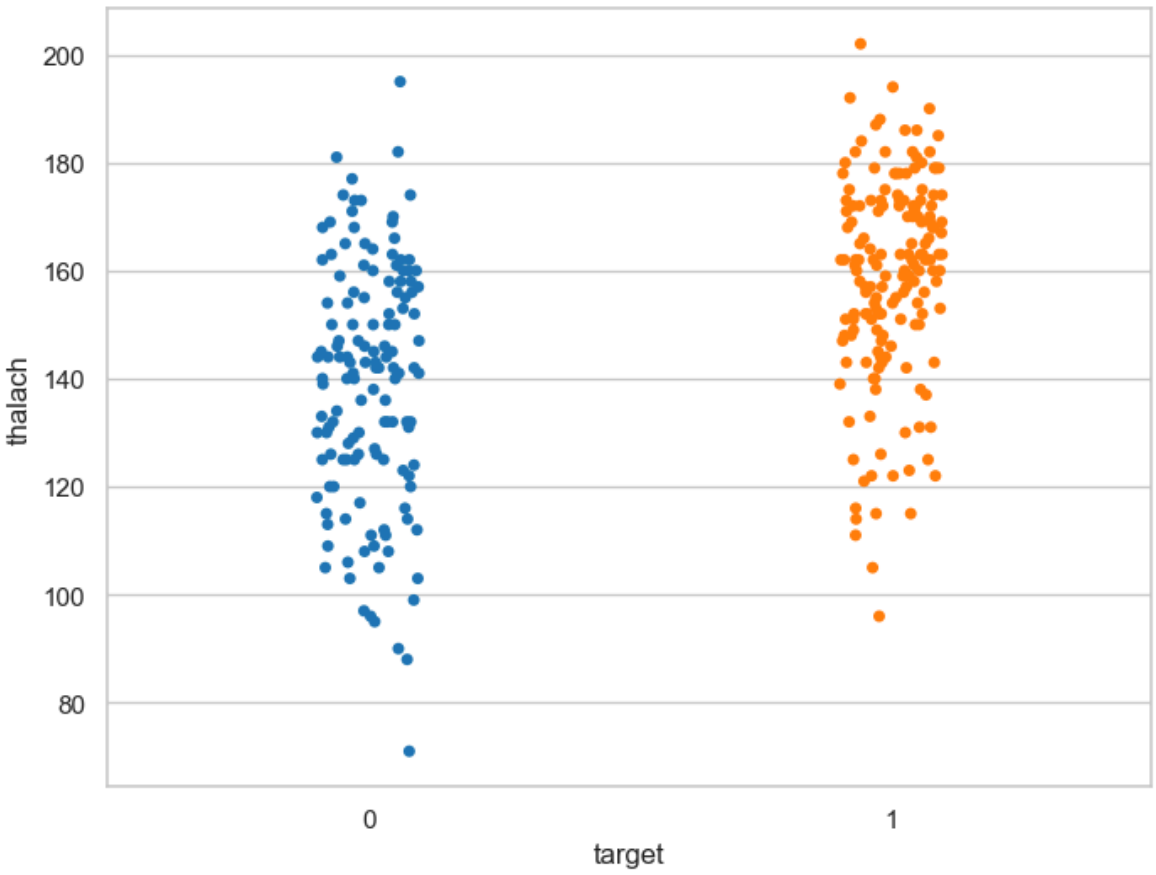



In [101…

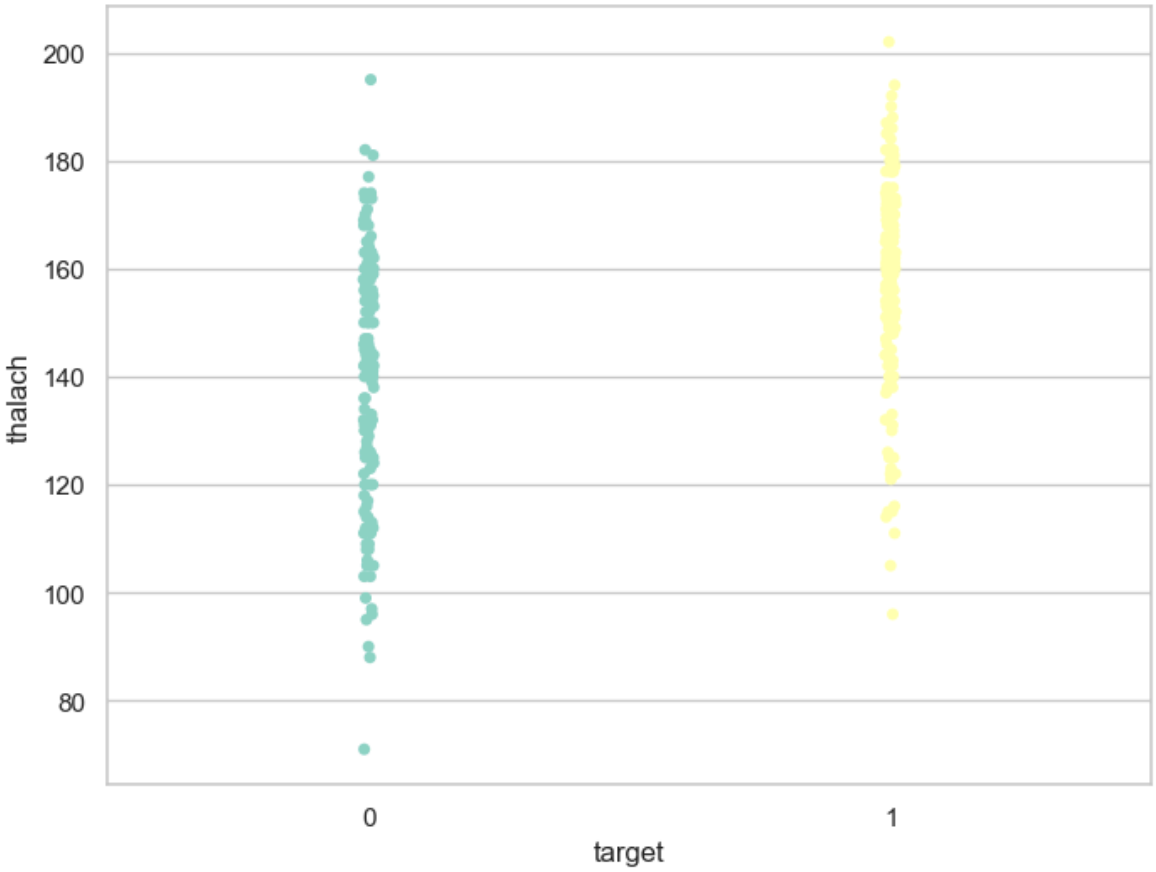
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
ax=sns.distplot(x,kde=False,rug=True,bins=10) #kde=kernal density estimation
plt.show()
```



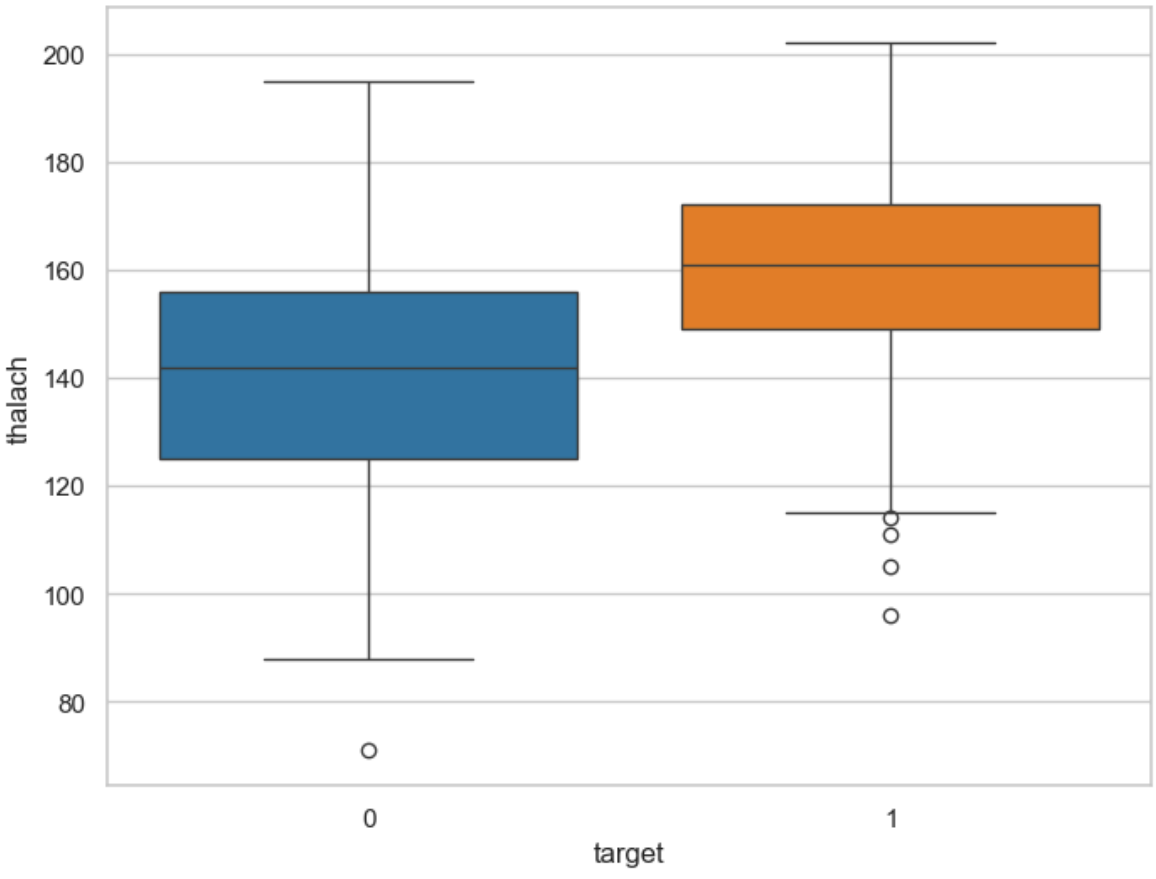

In [103…
```
f,ax=plt.subplots(figsize=(8,6))
sns.stripplot(data=df,x='target',y='thalach',palette='tab10')
plt.show()
```




In [105…
```
f,ax=plt.subplots(figsize=(8,6))
sns.stripplot(x='target',y='thalach',data=df,jitter=0.01,palette='Set3')
plt.show()
```

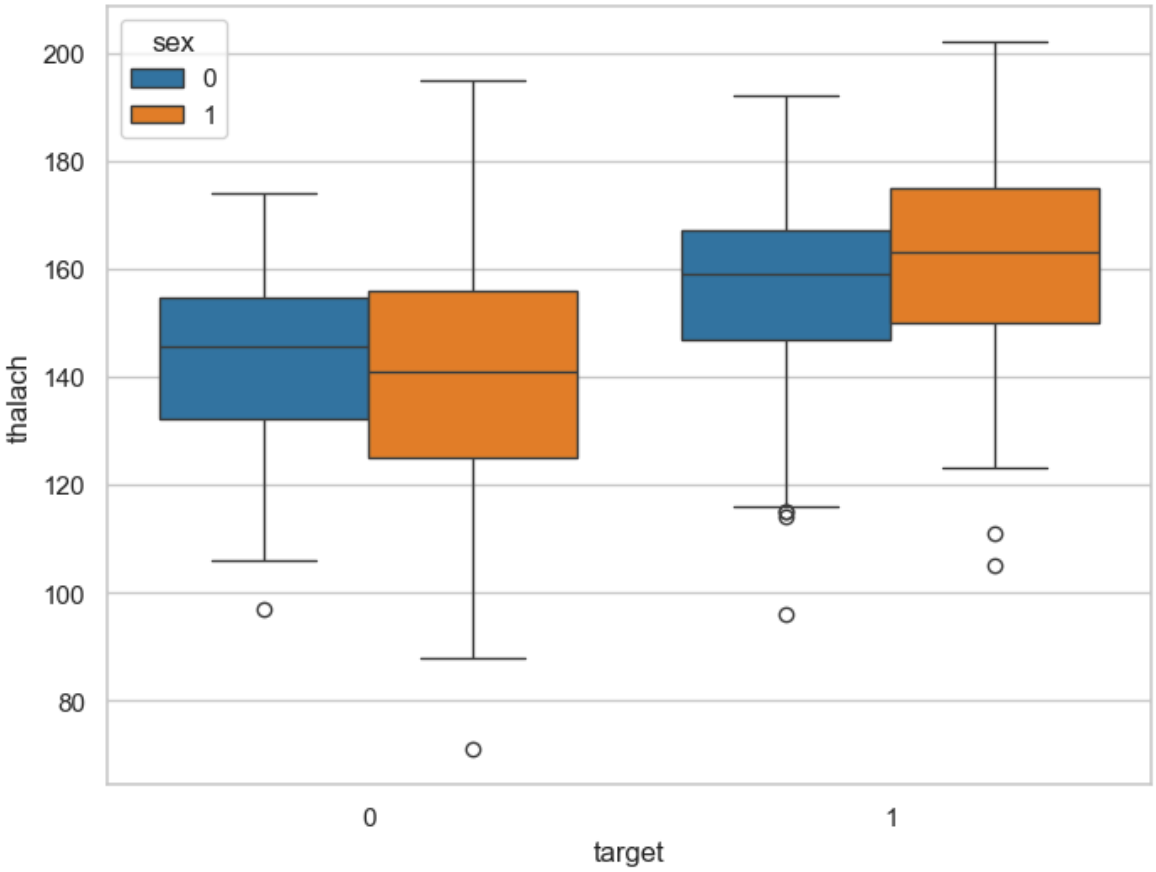



In [107…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x='target',y='thalach',data=df,palette='tab10')
plt.show()
```




In [108…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x='target',y='thalach',data=df,palette='tab10',hue='sex')
```

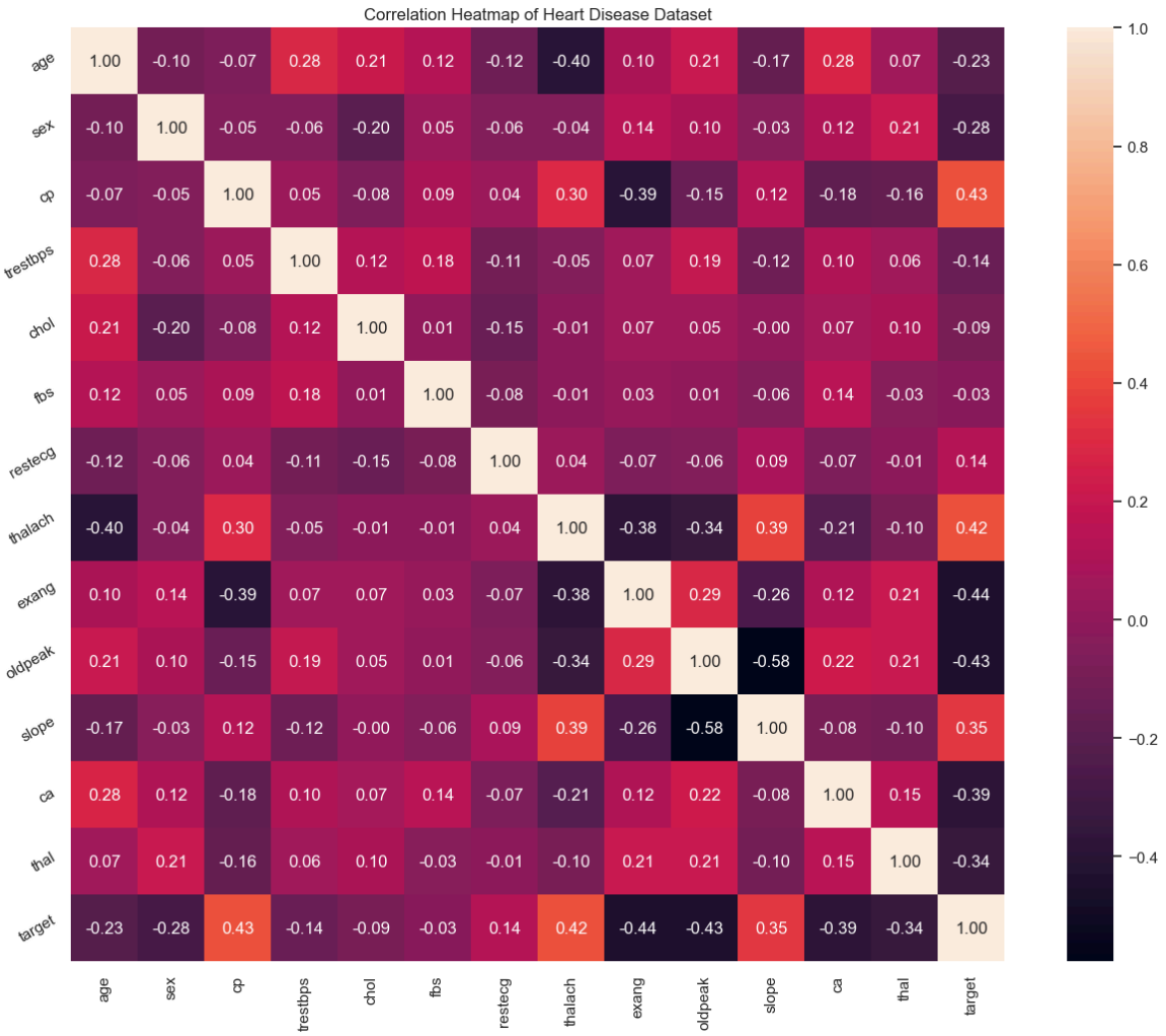
```
plt.show()
```




# Heat Map

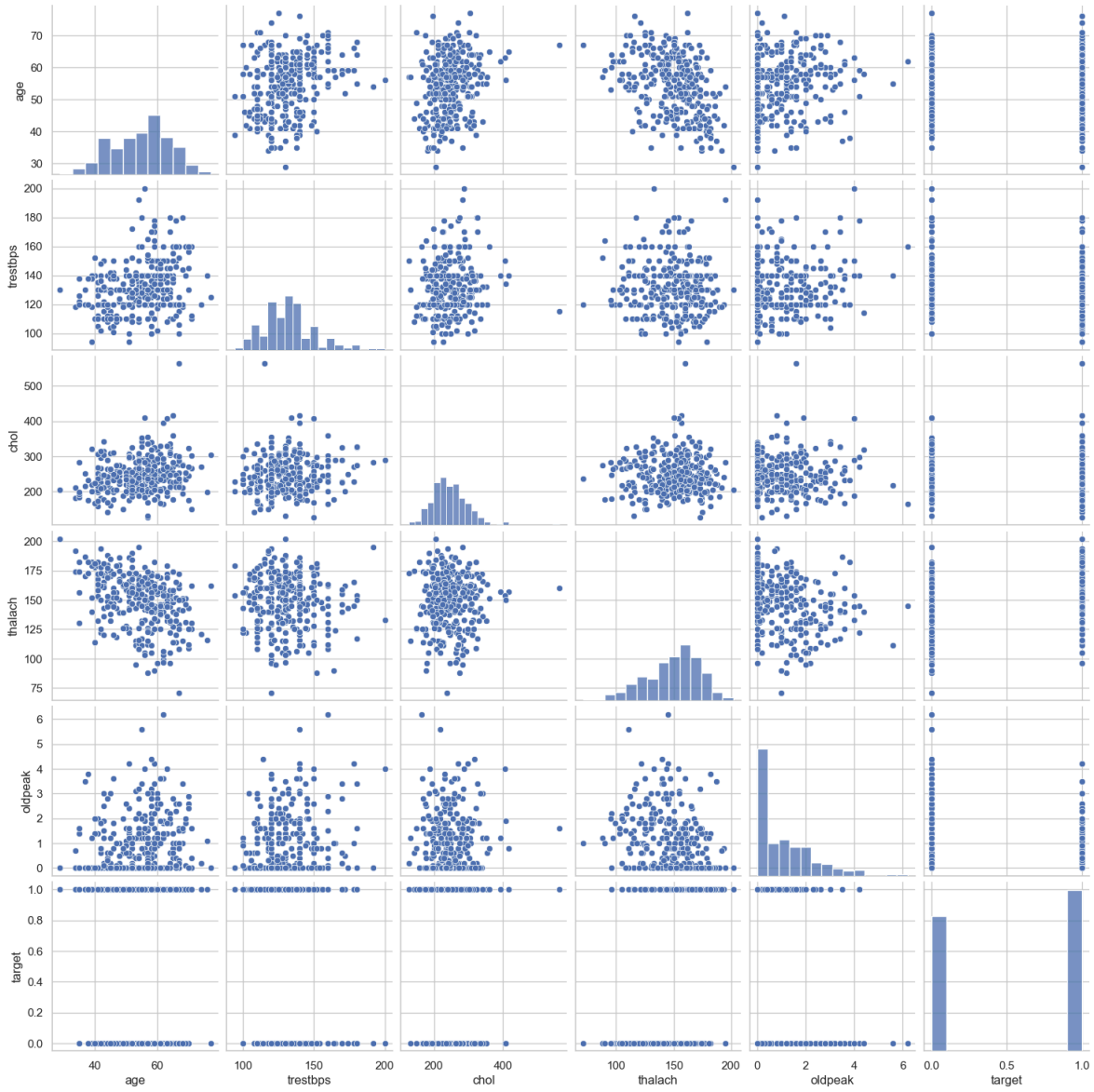
In [111…

```
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()
```




In [113…

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




In [114…
```
df['age'].nunique()
```

Out[114…
```
41
```

In [115…
```
df['age'].unique()
```

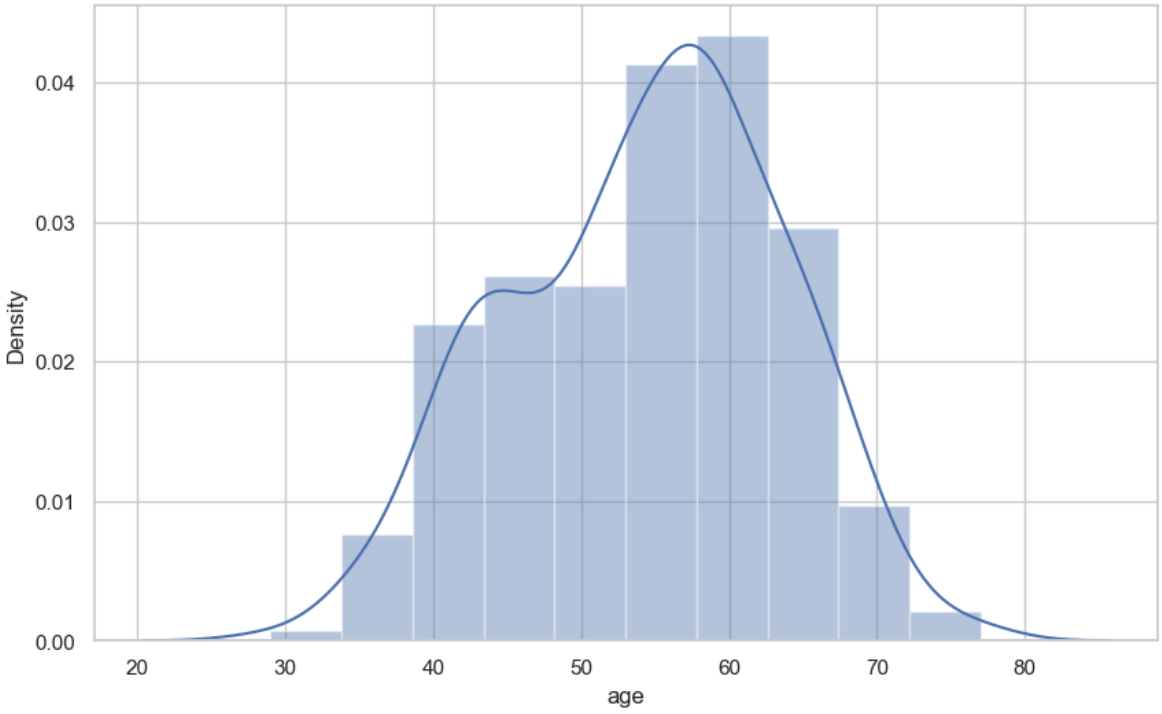
Out[115…
```
array([63, 37, 41, 56, 57, 44, 52, 54, 48, 49, 64, 58, 50, 66, 43, 69, 59,
       42, 61, 40, 71, 51, 65, 53, 46, 45, 39, 47, 62, 34, 35, 29, 55, 60,
       67, 68, 74, 76, 70, 38, 77])
```

In [116…
```
df['age'].describe()
```

Out[116…
```
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
```

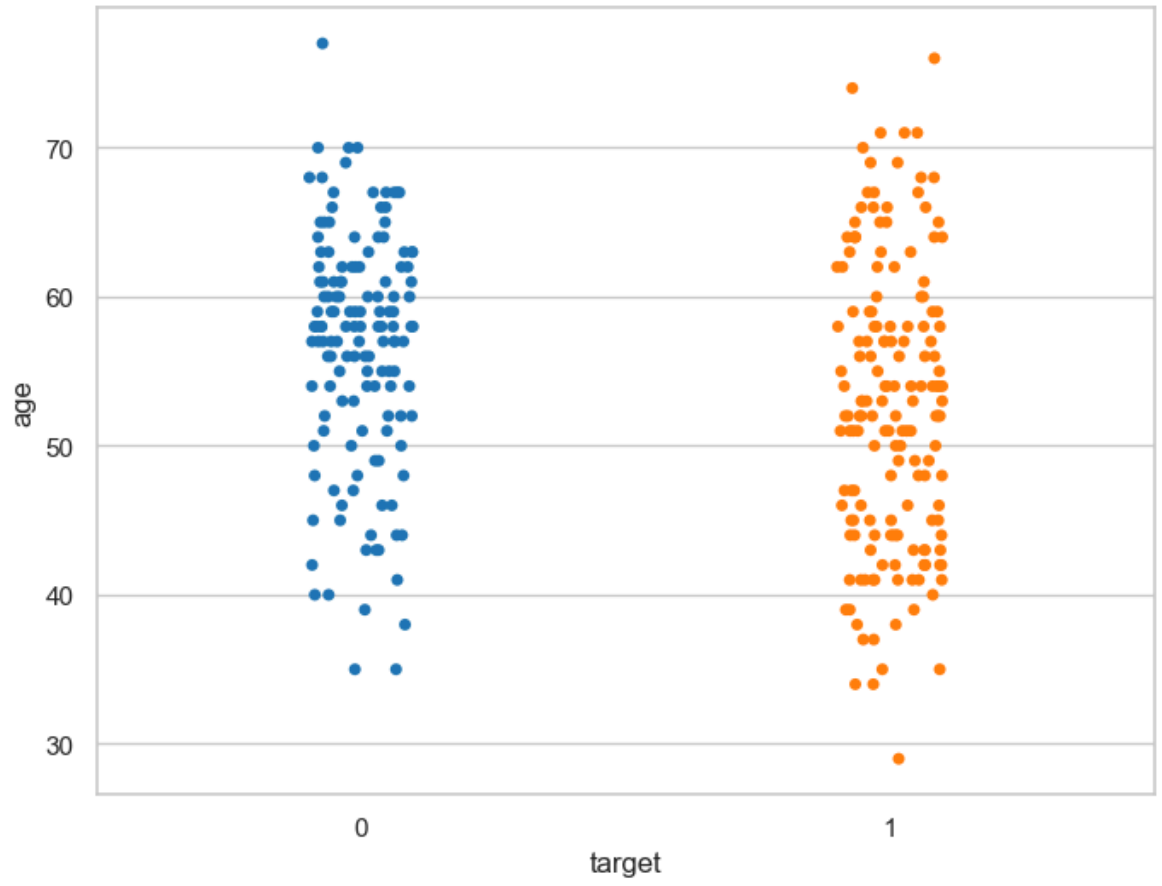
In [117…
```
f,ax=plt.subplots(figsize=(10,6))
x=df['age']
```

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



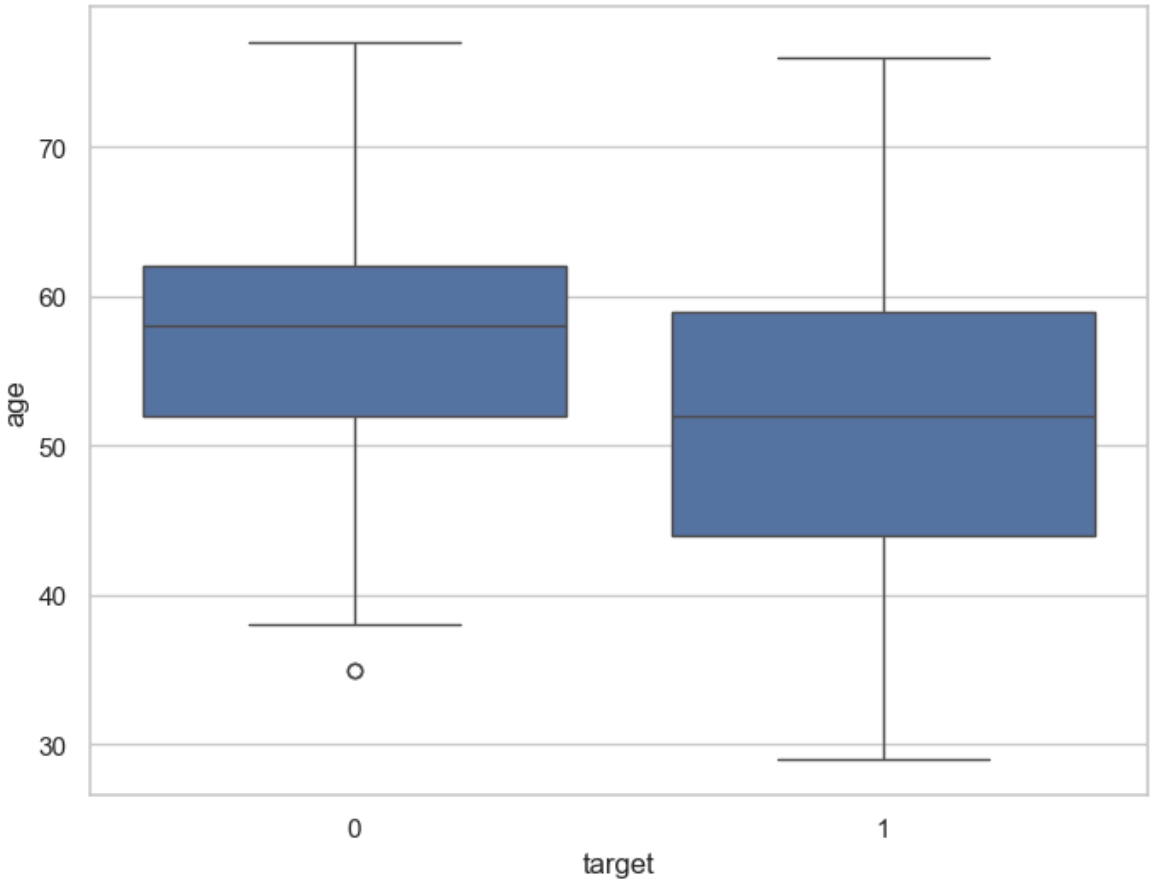

In [119…

```
f,ax=plt.subplots(figsize=(8,6))
sns.stripplot(x='target',y='age',data=df,palette='tab10')
plt.show()
```



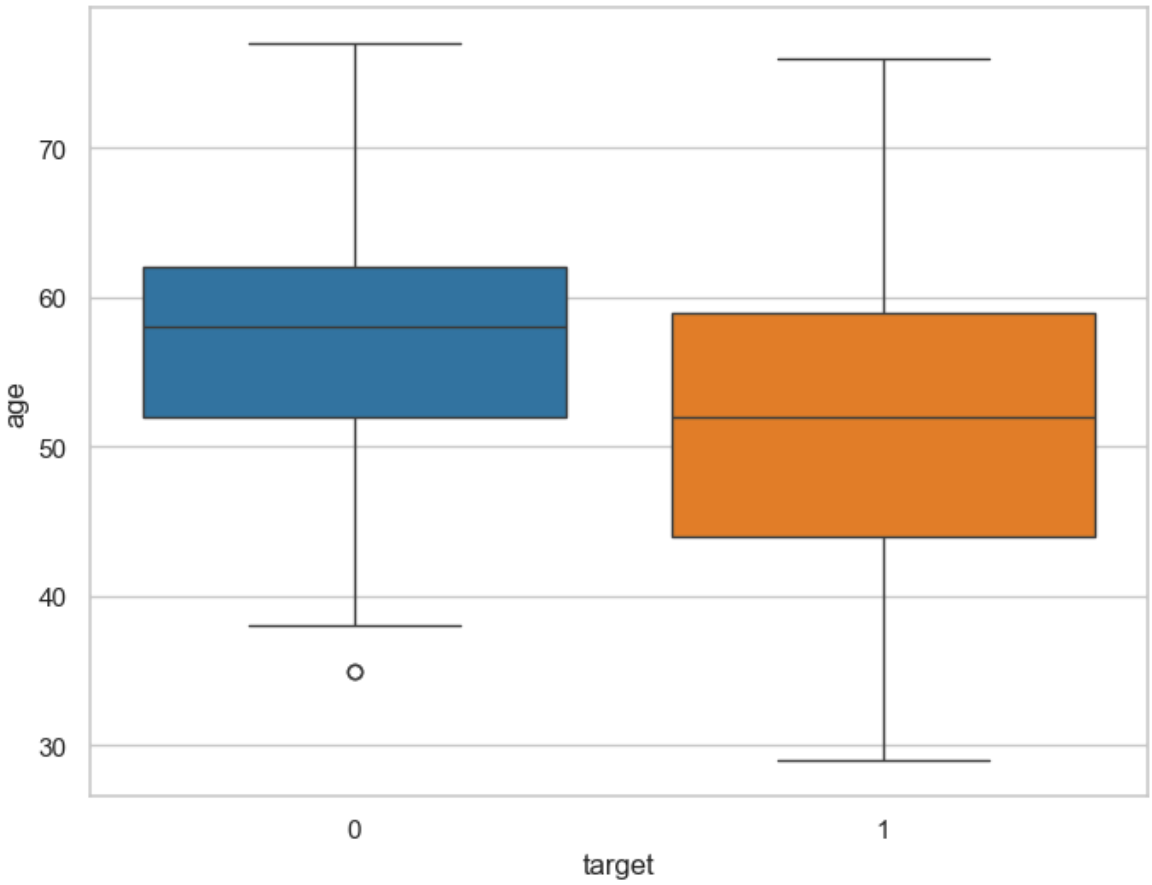

In [120…

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

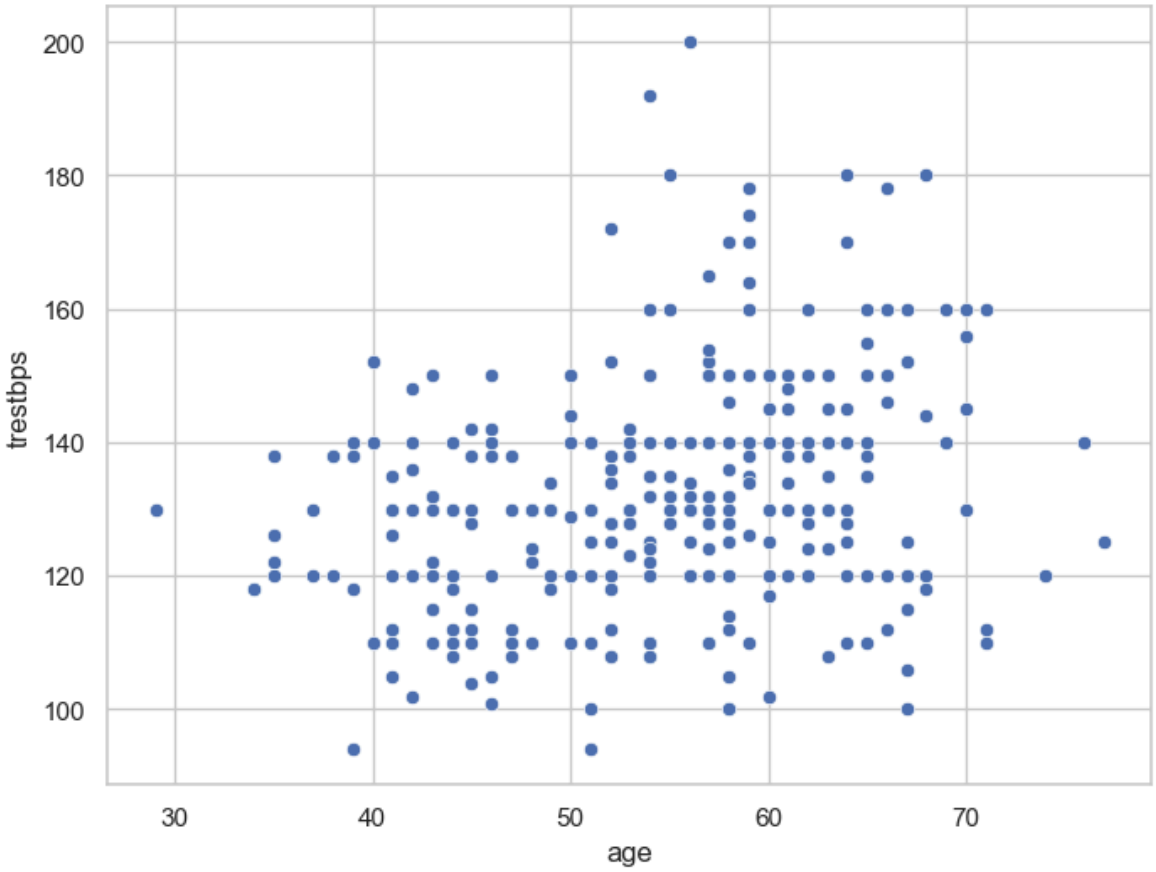



In [121…

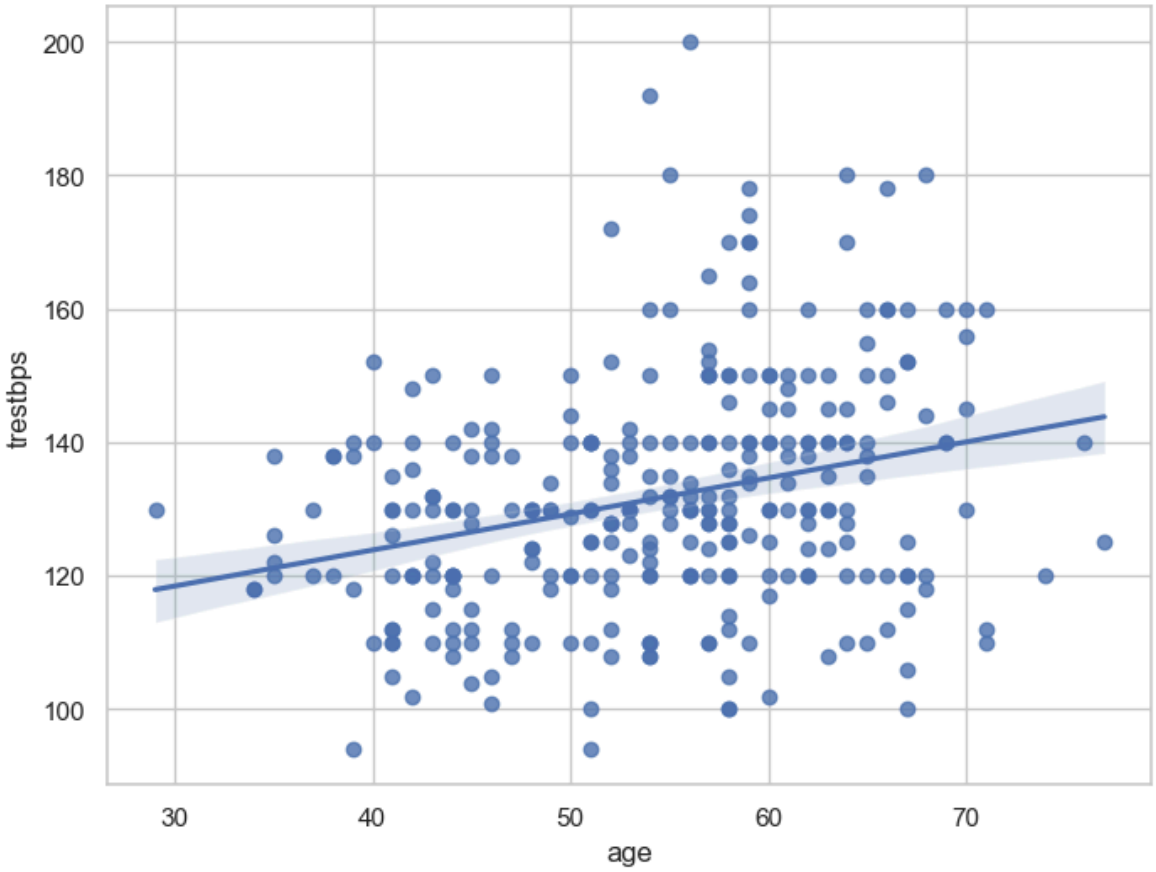
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(data=df,x='target',y='age',palette='tab10')
plt.show()
```



In [122…
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.scatterplot(x='age',y='trestbps',data=df)
plt.show()
```

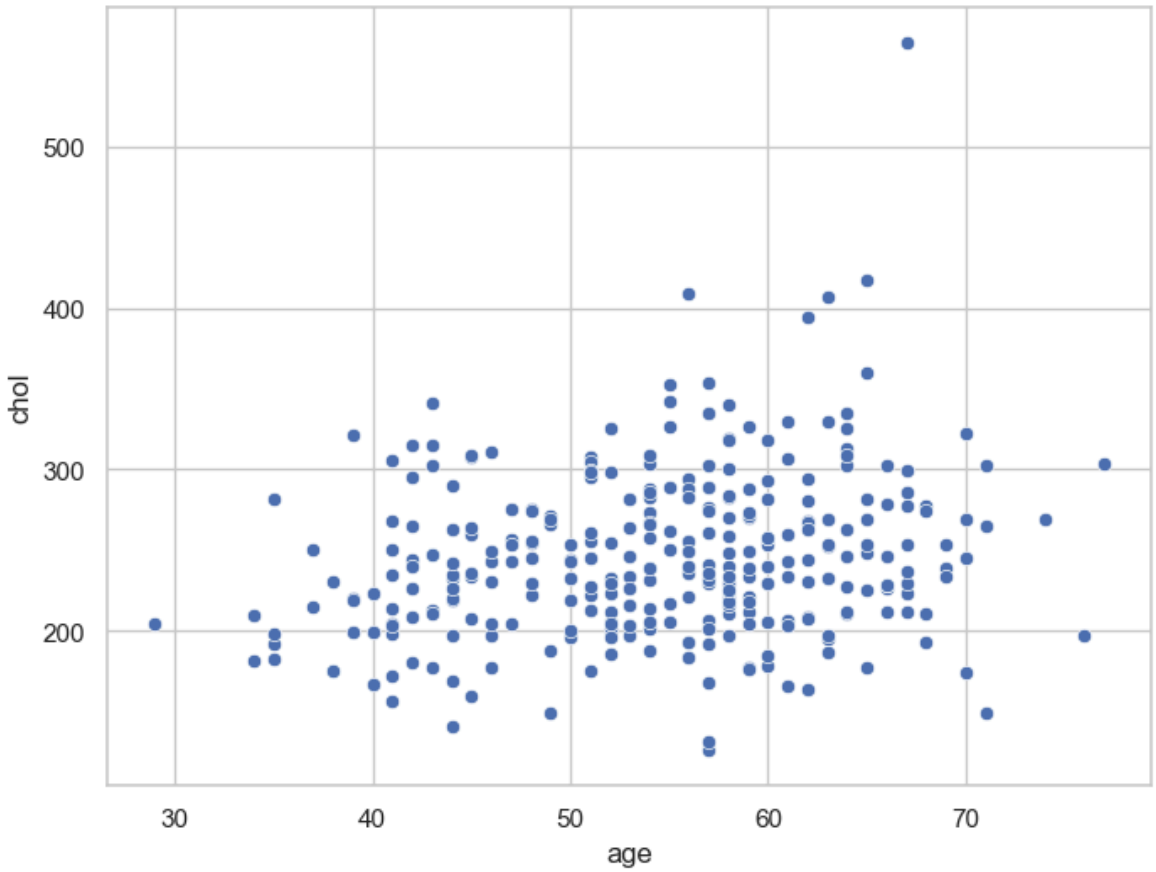



In [123…
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.regplot(x='age',y='trestbps',data=df)
plt.show()
```




In [124…

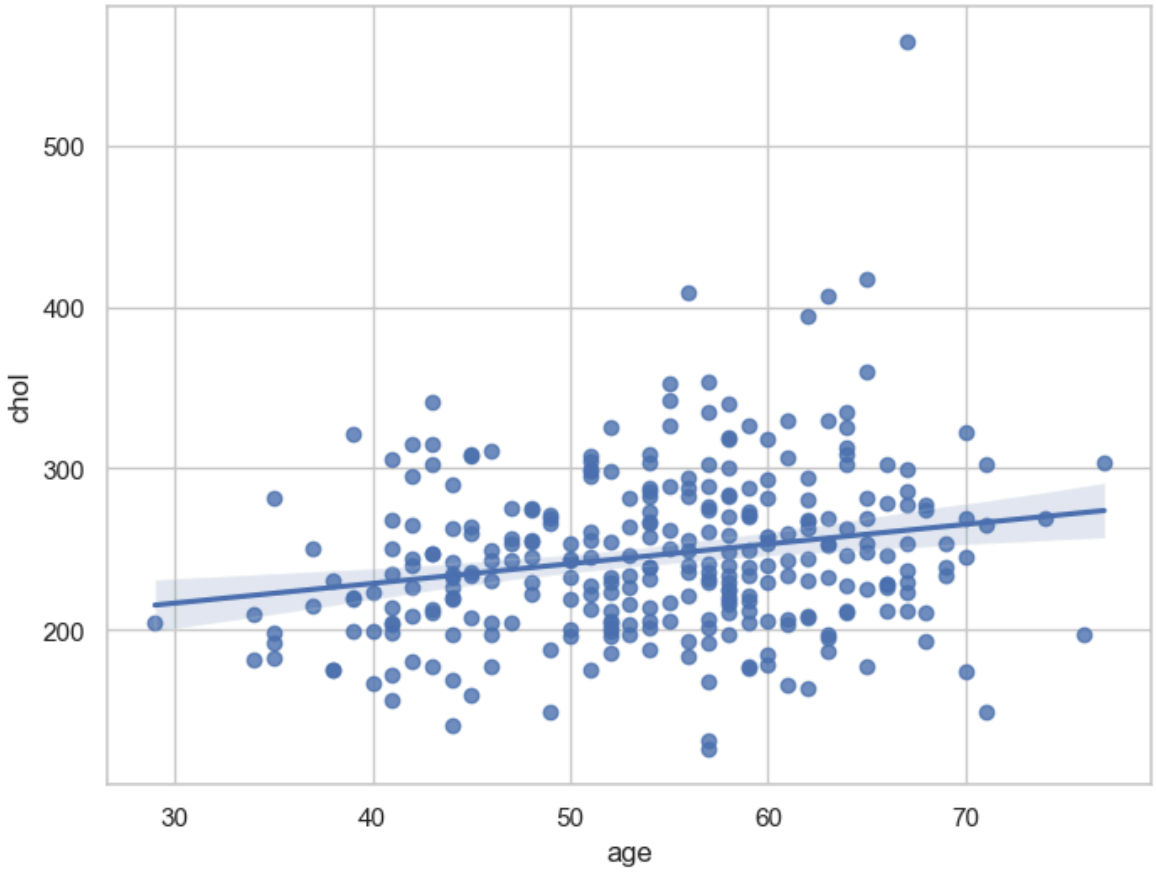
```
f,ax=plt.subplots(figsize=(8,6))
ax=sns.scatterplot(x='age',y='chol',data=df)
plt.show()
```




In [125…

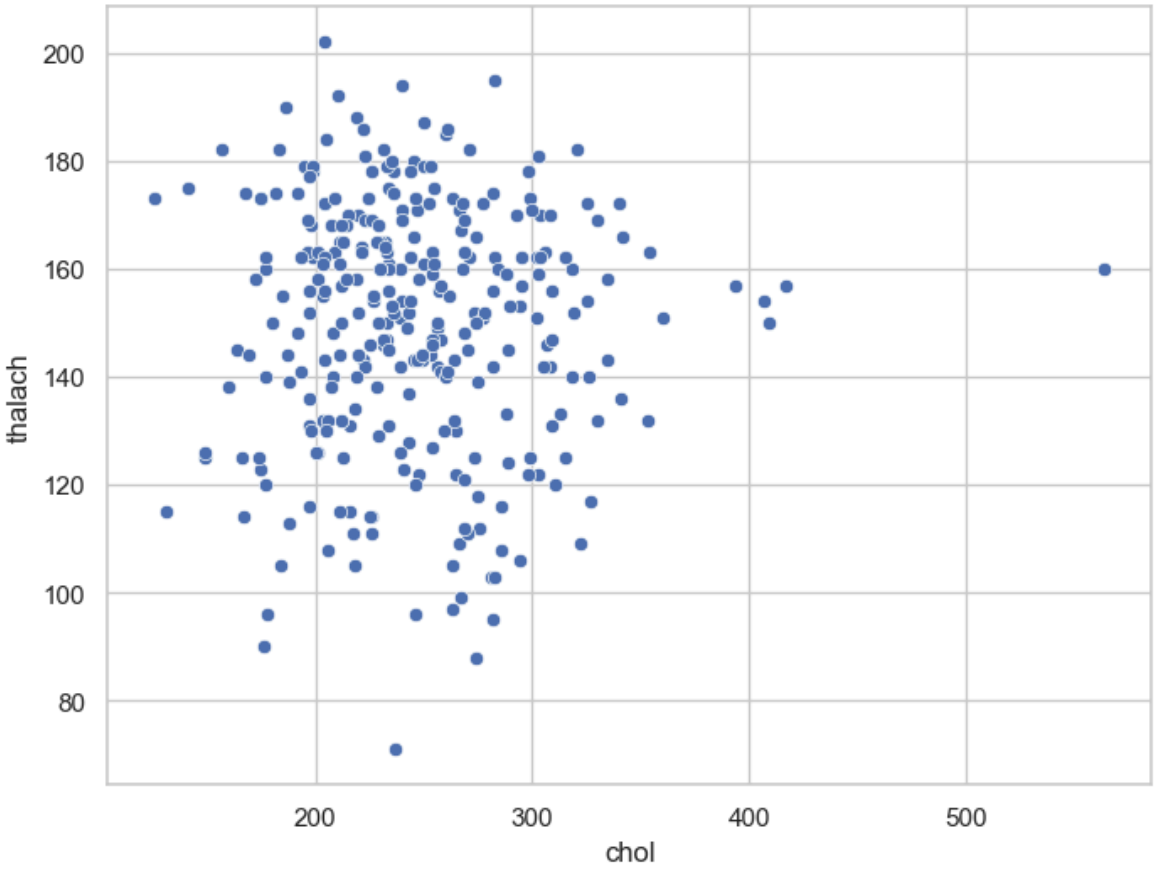
```
f,ax=plt.subplots(figsize=(8,6))
sns.regplot(x='age',y='chol',data=df)
```

```
plt.show()
```

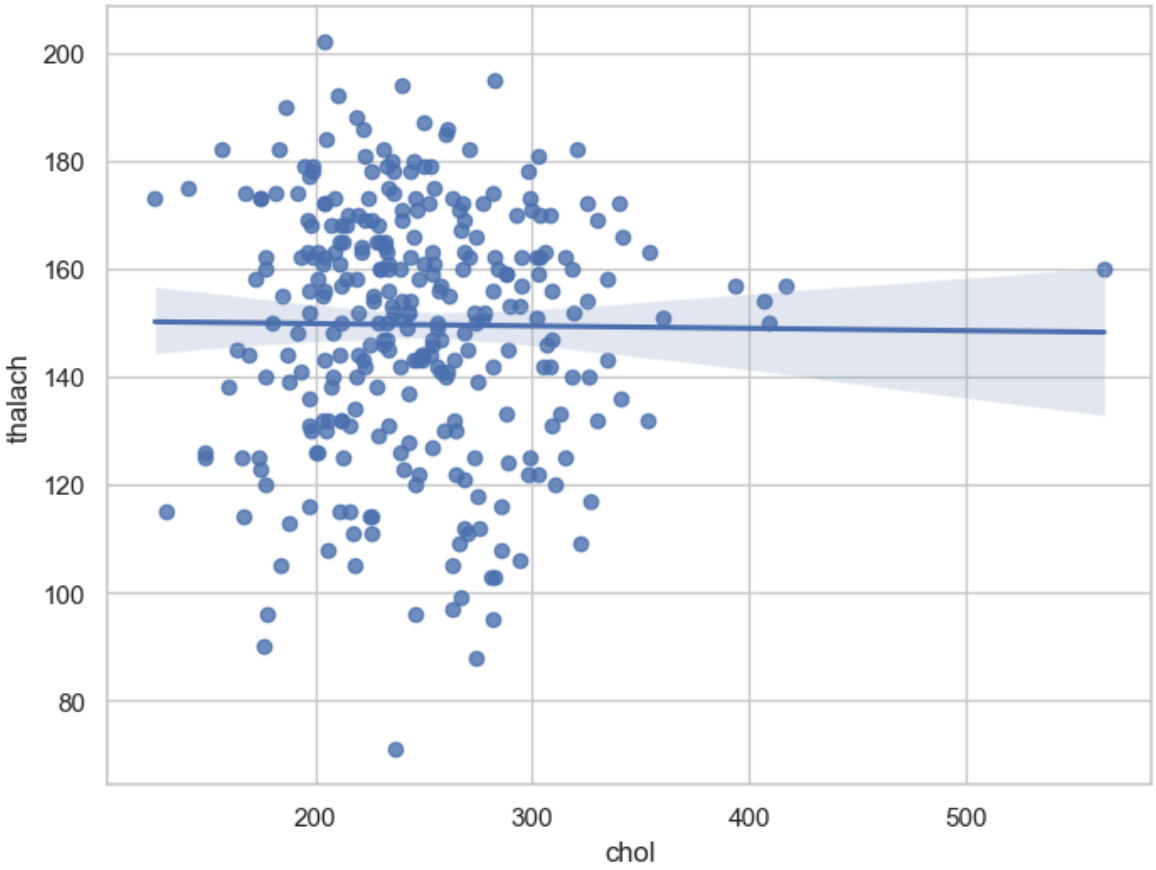



In [126…

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



In [127…
```
f,ax=plt.subplots(figsize=(8,6))
sns.regplot(data=df,x='chol',y='thalach')
plt.show()
```




In [128…
```
df.isnull()
```

Out[128…

| | 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 [129…
```
df.isnull().sum()
```

Out[129…
```
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 [130…
```
assert pd.notnull(df).all().all()
```

In [131…
```
assert pd.isnull(df).all().all()
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[131], line 1
----> 1 assert pd.isnull(df).all().all()

AssertionError:
```

In [132…
```
assert pd.notnull(df).all().all()
```

In [133…
```
assert(df>=0).all().all()
```

In [134…
```
assert(df<=0).all().all()
```

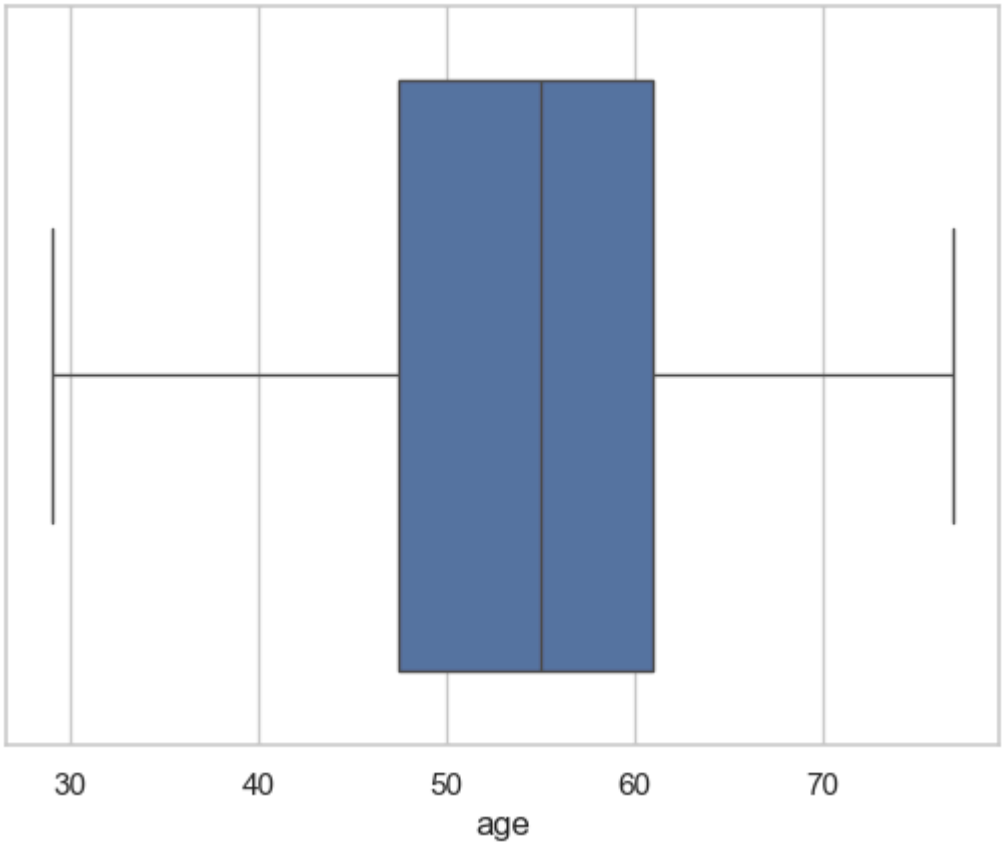
```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
Cell In[134], line 1
----> 1 assert(df<=0).all().all()

AssertionError:
```

In [135…
```
assert(df>=0).all().all()
```

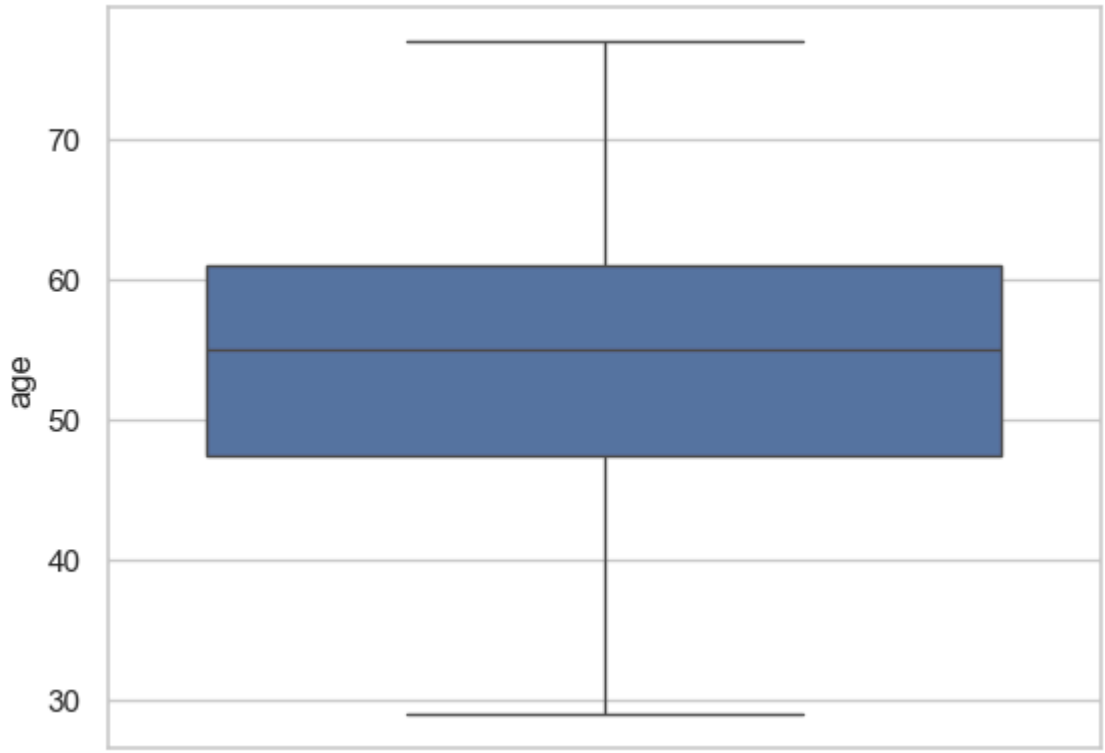
In [136…
```
df['age'].describe()
```

Out[136…
```
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
```

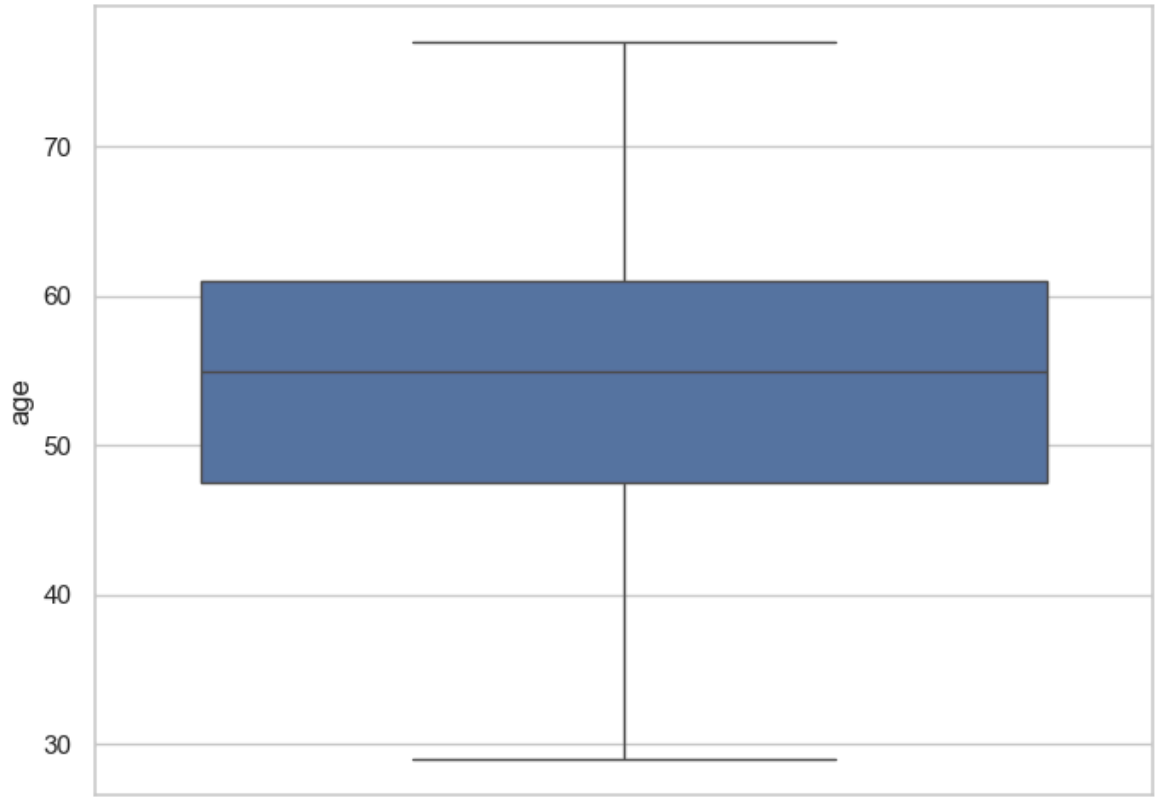
In [137…
```
sns.boxplot(x=df['age'])
plt.show()
```



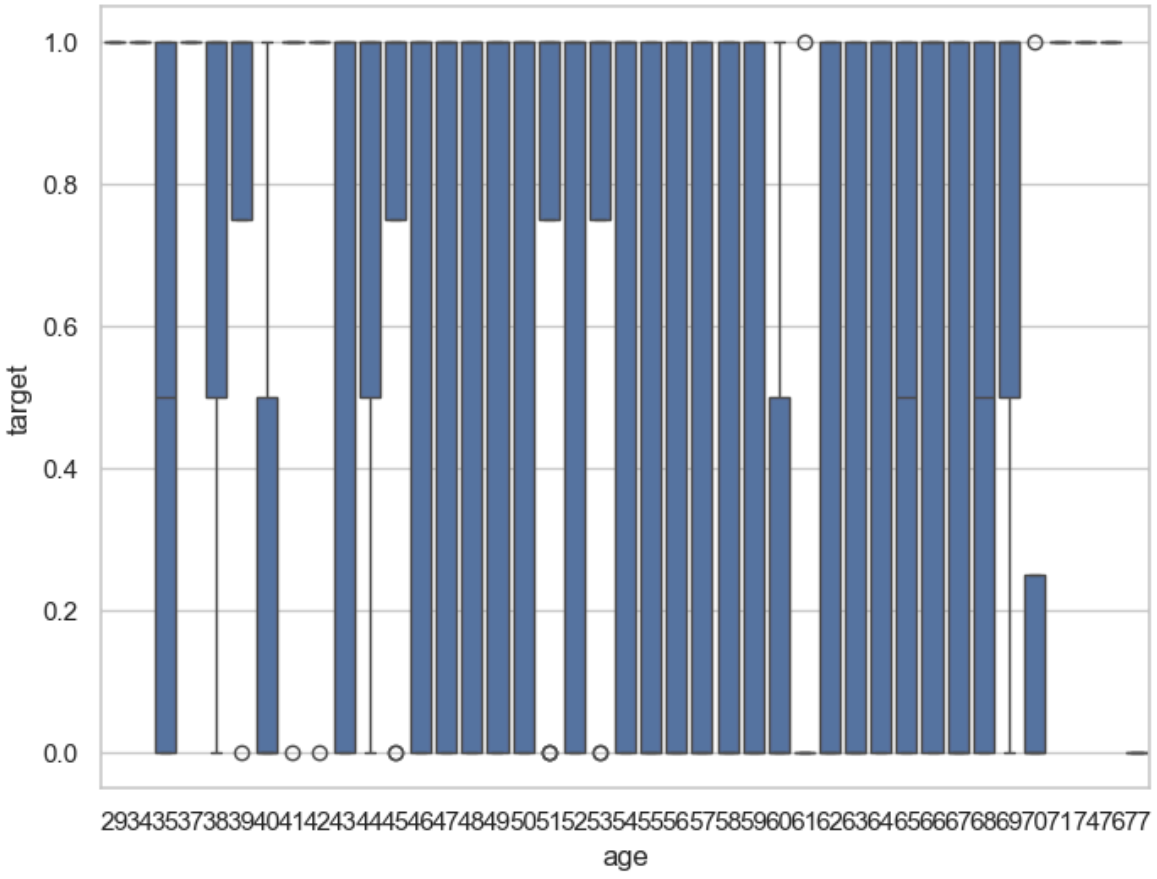

In [138…
```
sns.boxplot(df['age'])
plt.show()
```




In [141…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(y=df['age'])
plt.show()
```



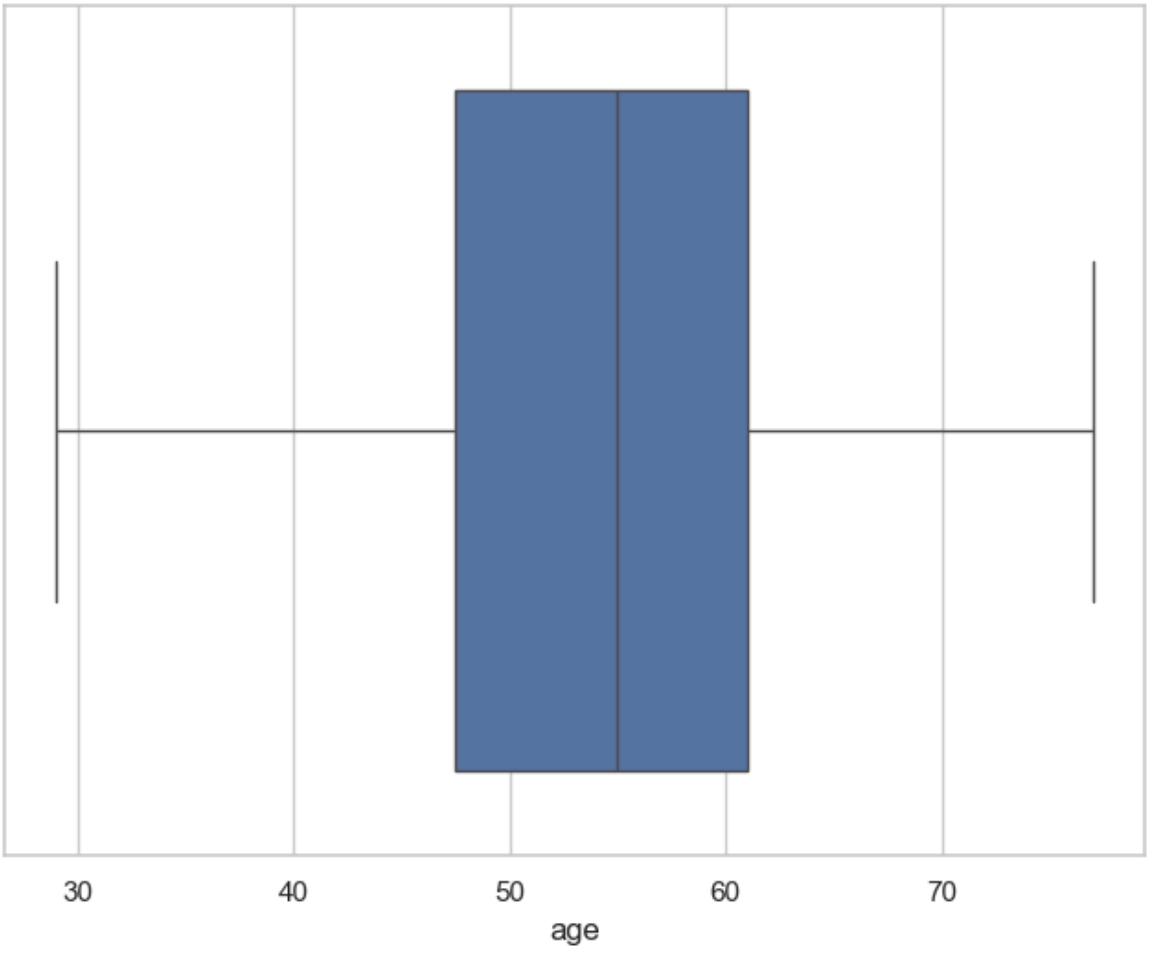

In [142…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=df['age'],y=df['target'])
plt.show()
```




In [143…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(df['age'],df['target'])
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[143], line 2
      1 f,ax=plt.subplots(figsize=(8,6))
----> 2 sns.boxplot(df['age'],df['target'])
      3 plt.show()

TypeError: boxplot() takes from 0 to 1 positional arguments but 2 were given
```

In [146…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=df['age'])
plt.show()
```



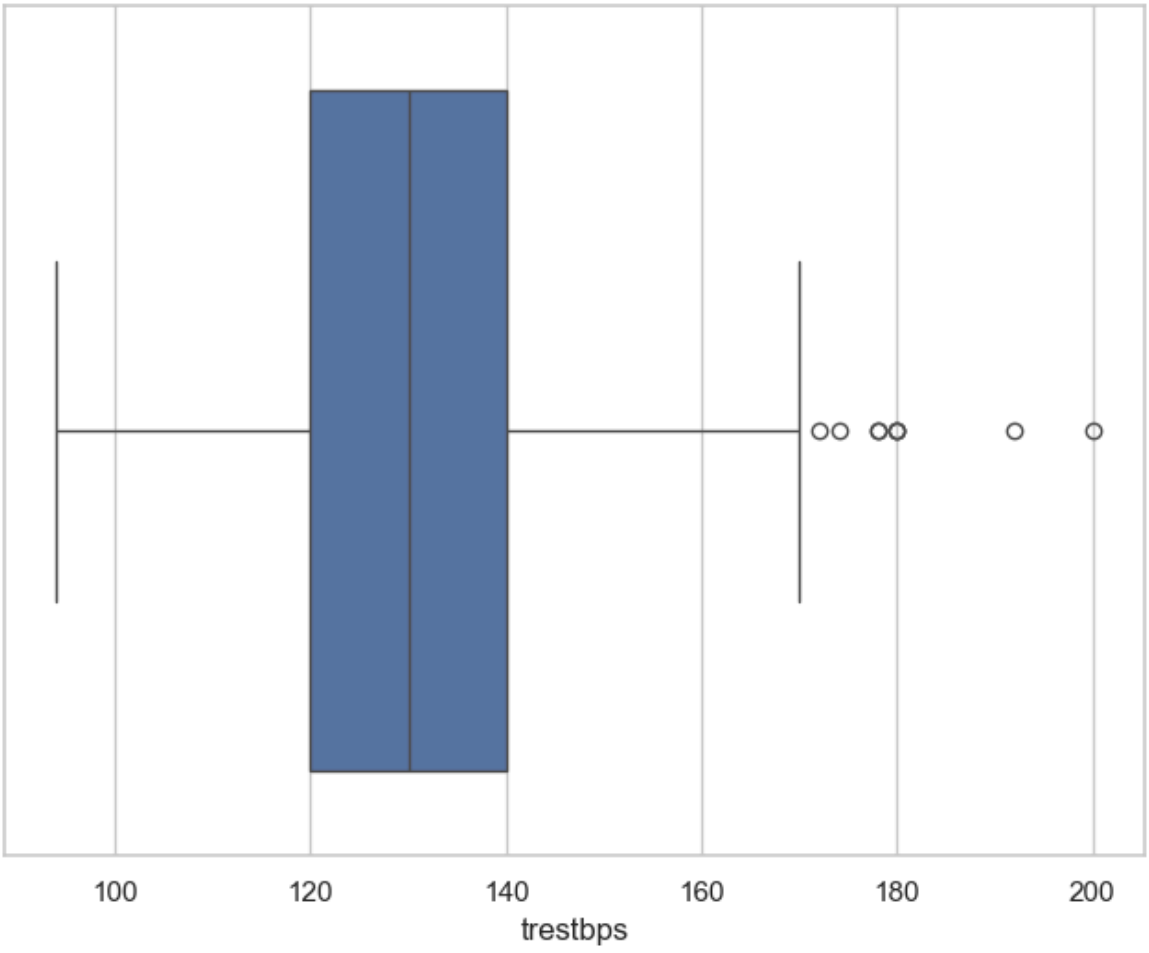

In [147…
```
df['age'].describe()
```

Out[147…
```
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 [148…
```
df['trestbps'].describe()
```

Out[148…
```
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
```

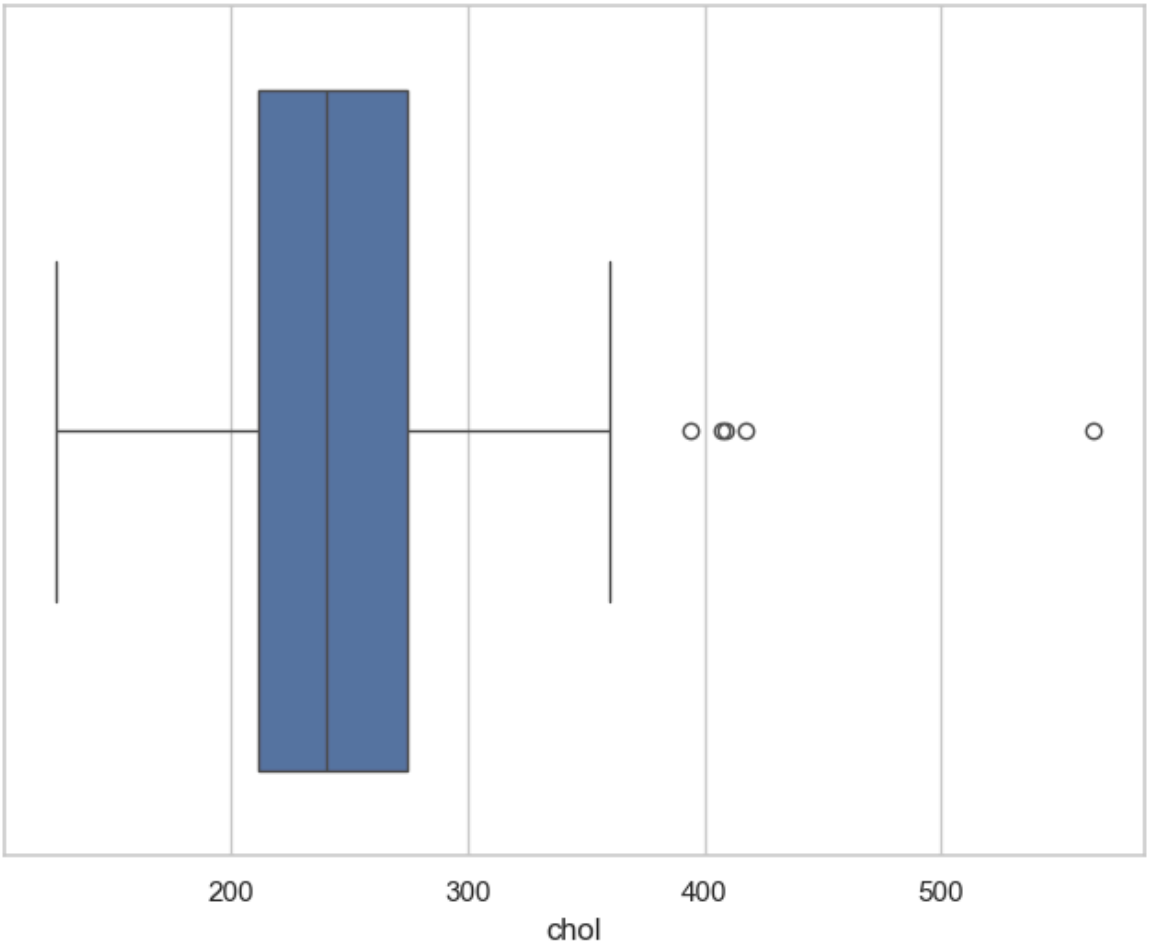
In [149…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=df['trestbps'])
plt.show()
```




In [150…
```
df['chol'].describe()
```

Out[150…
```
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
```

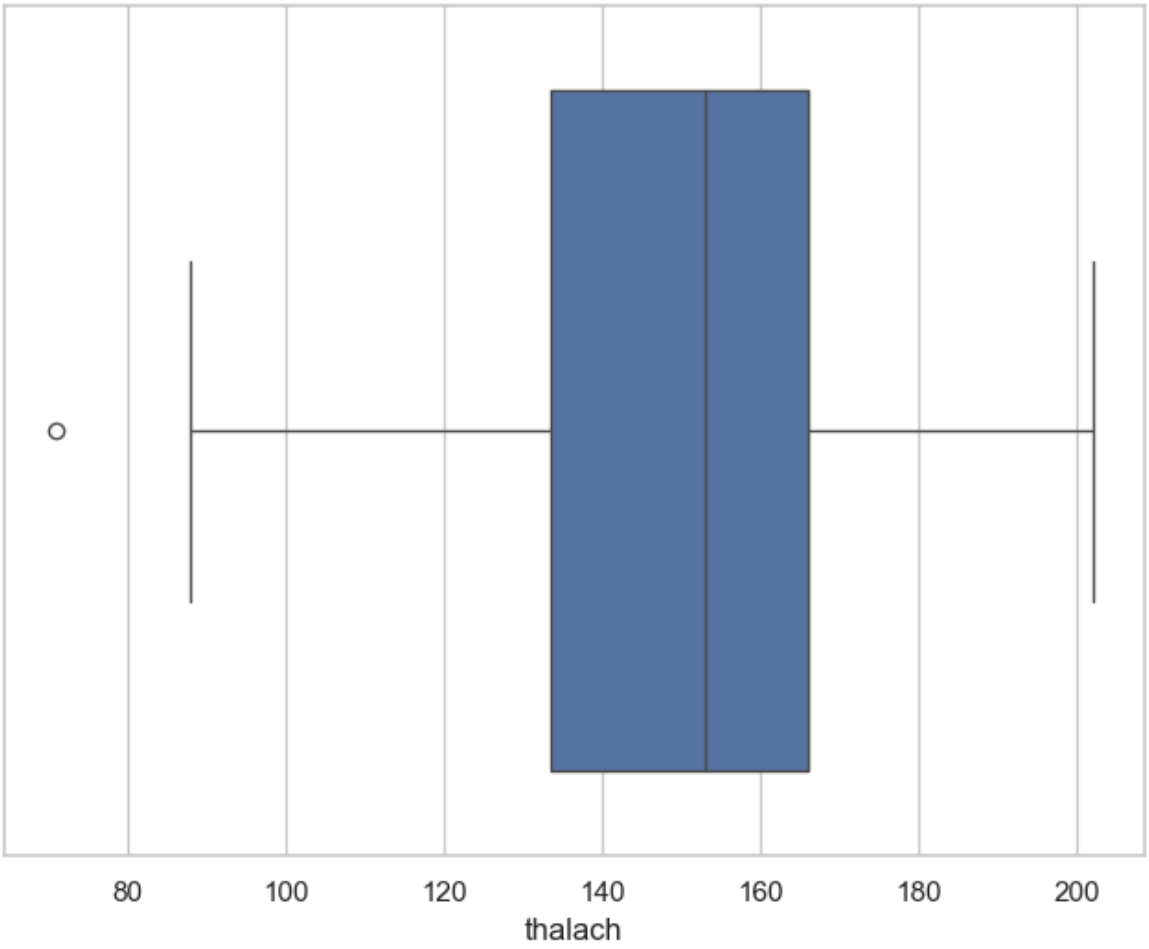
In [151…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=df['chol'])
plt.show()
```




In [152…
```
df['thalach'].describe()
```

Out[152…
```
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 [153…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=df['thalach'])
plt.show()
```




In [154…
```
df
```

Out[154…

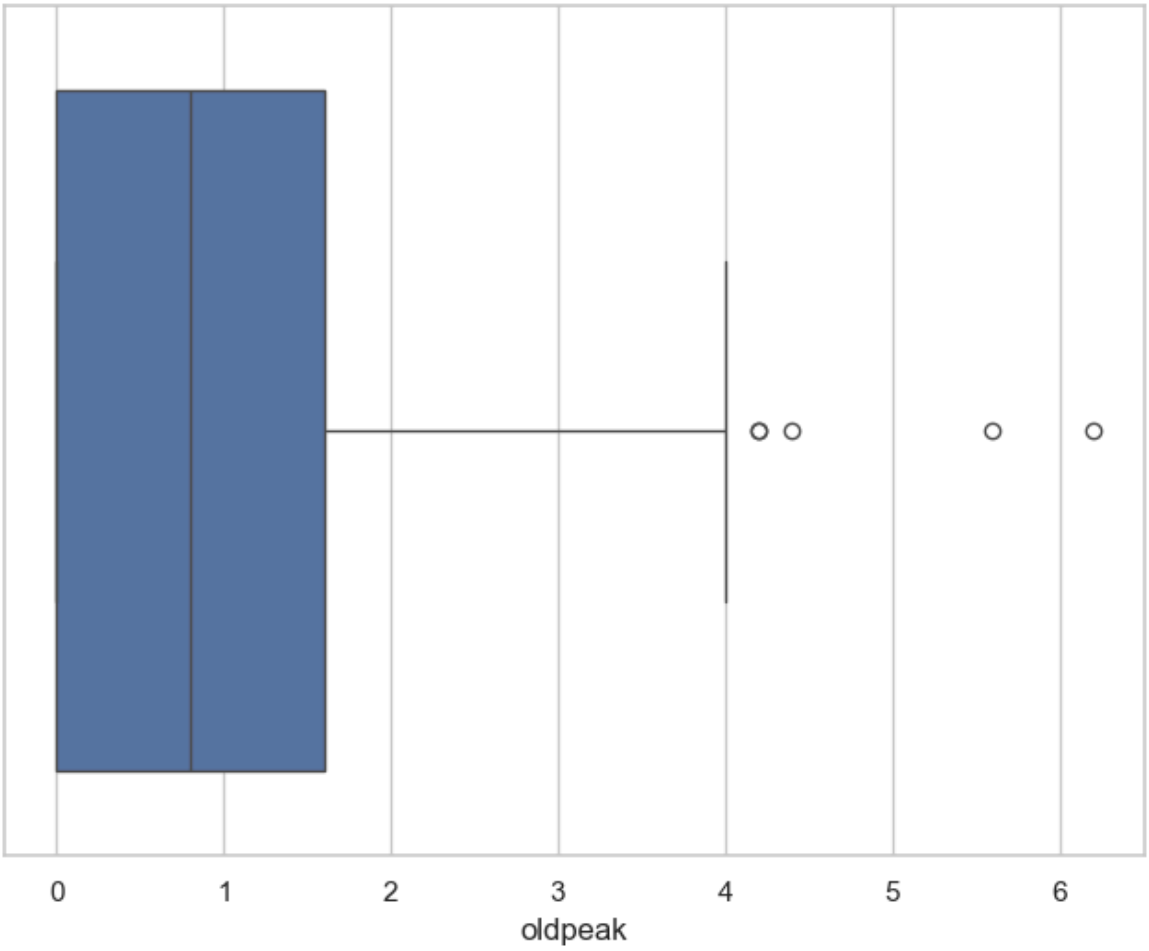
| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | th |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | |
| **2**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

303 rows × 14 columns

In [155…
```
df['oldpeak'].describe()
```

Out[155…
```
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 [156…
```
f,ax=plt.subplots(figsize=(8,6))
sns.boxplot(x=df['oldpeak'])
plt.show()
```




In [ ]: