HEART DISEASE PROJECT1

1.import libraries

In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
%matplotlib inline
sns.set(style="whitegrid")
```

In [4]:
```
# ignore warnings
import warnings
warnings.filterwarnings("ignore")
```

2.import dataset

In [6]:
```
df=pd.read_csv(r"C:\Users\megha\Downloads\2nd- Seaborn, Eda Practicle\2nd- Seaborn,
```

In [7]:
```
df
```

Out[7]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 14 columns

3. Exploratory data analysis

In [9]:
```
print('the shape of the dataset:', df.shape)
```

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

4.preview the dataset

In [11]:
```
df.head()
```

Out[11]:

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

5.summary of dataset

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
```

checking the datatypes

In [15]:
```
df.dtypes
```

Out[15]:
```
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
```

statistical properties of dataset

In [17]:
```
df.describe()
```

Out[17]:

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

view column names

In [19]:
```
df.columns
```

Out[19]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

6.Univariate analysis

In [21]:
```
df['target'].nunique()
```

Out[21]:
```
2
```

In [22]:
```
df['target'].unique()
```

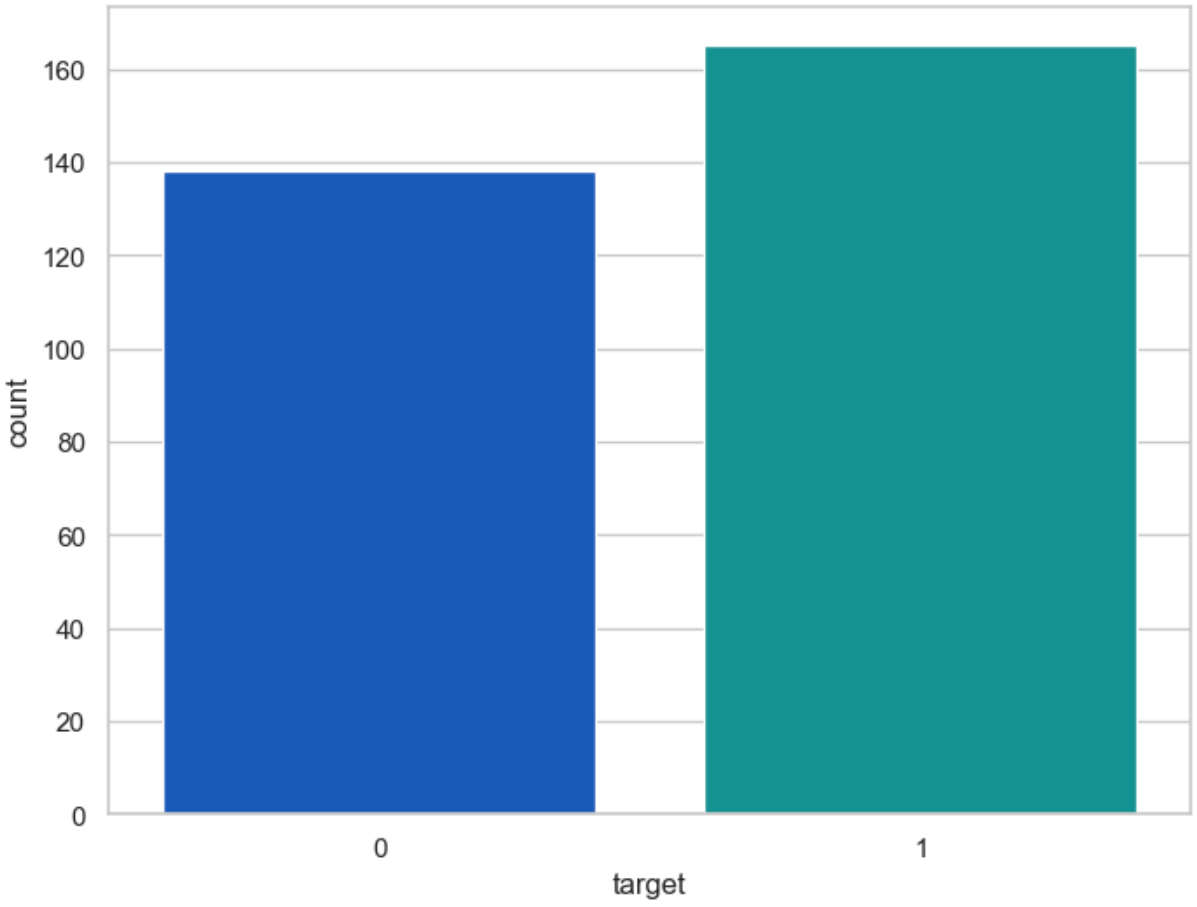
Out[22]:
```
array([1, 0], dtype=int64)
```

In [23]:
```
df['target'].value_counts()
```

Out[23]:
```
target
1    165
0    138
Name: count, dtype: int64
```

visualize frequency distribution of target variable

In [25]:
```
f,ax = plt.subplots(figsize=(8,6))
ax =sns.countplot(x="target",data=df,palette='winter')
plt.show()
```

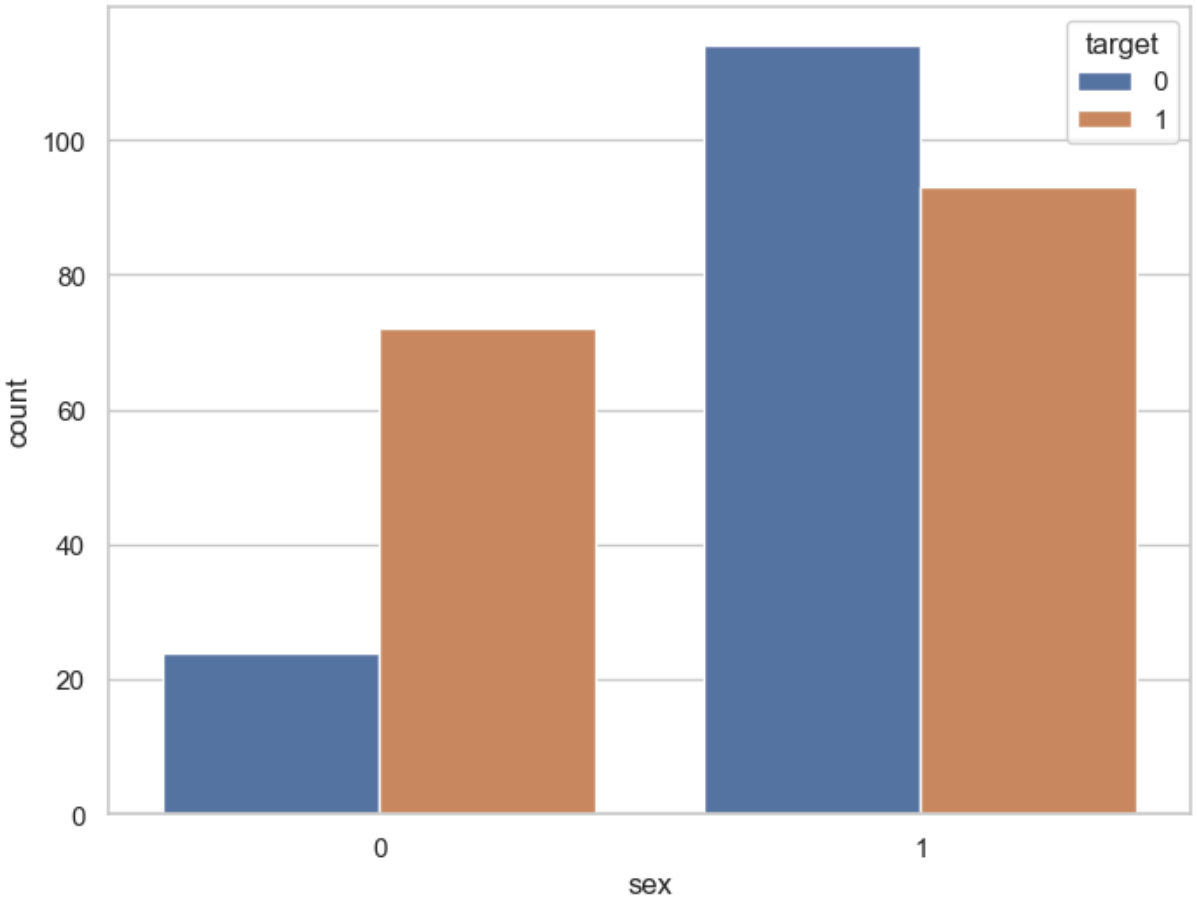


frequency distribution of target variable wrt sex

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

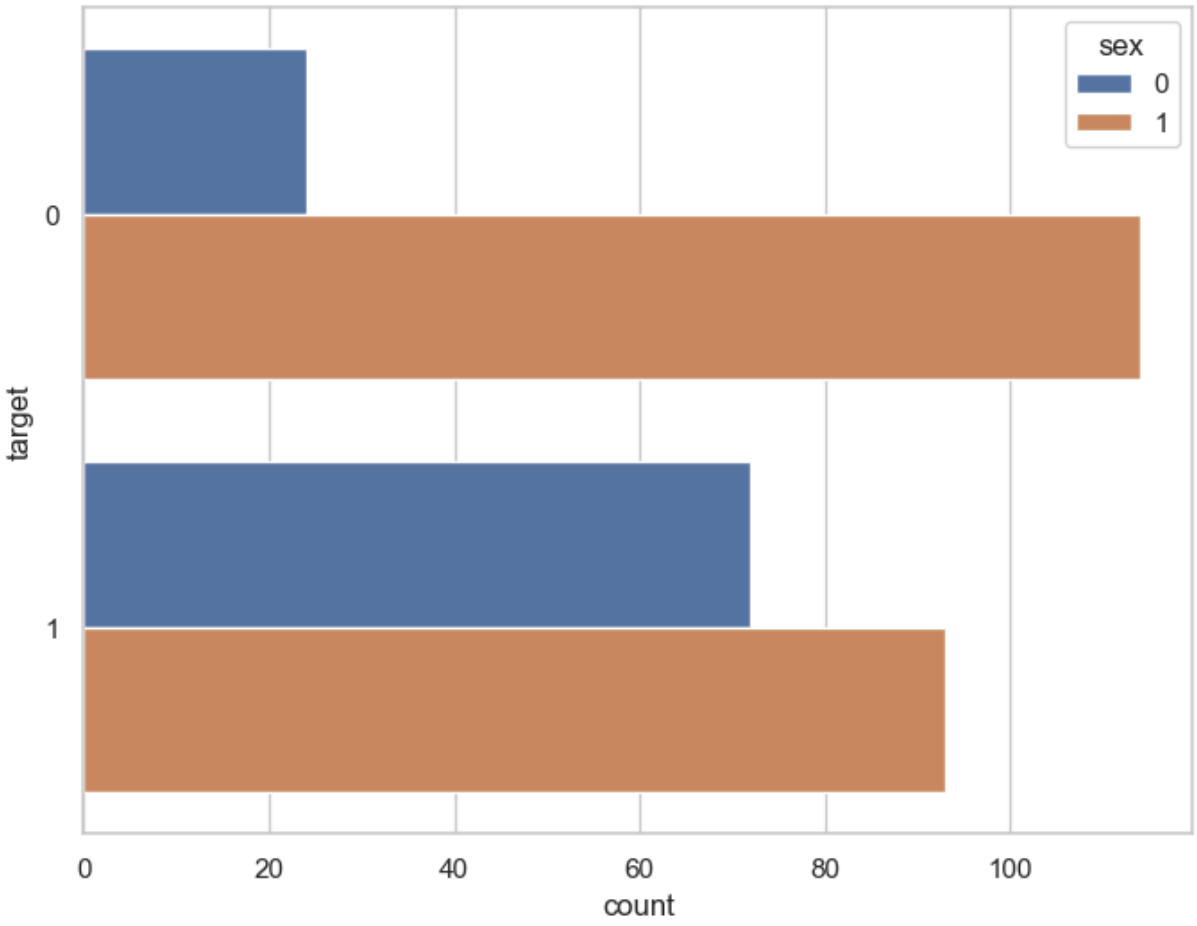
Out[27]:
```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

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



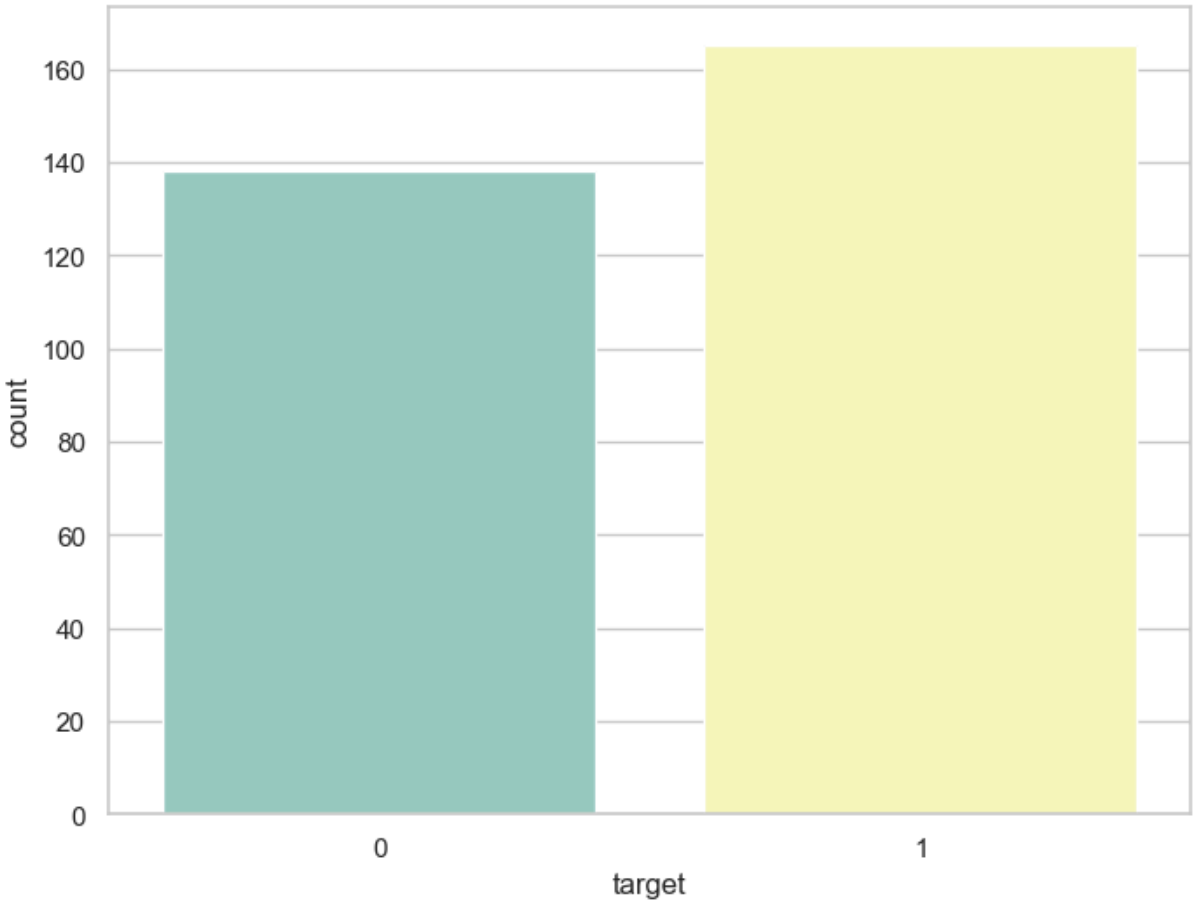
we can plot the bars horizontally as follows:

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



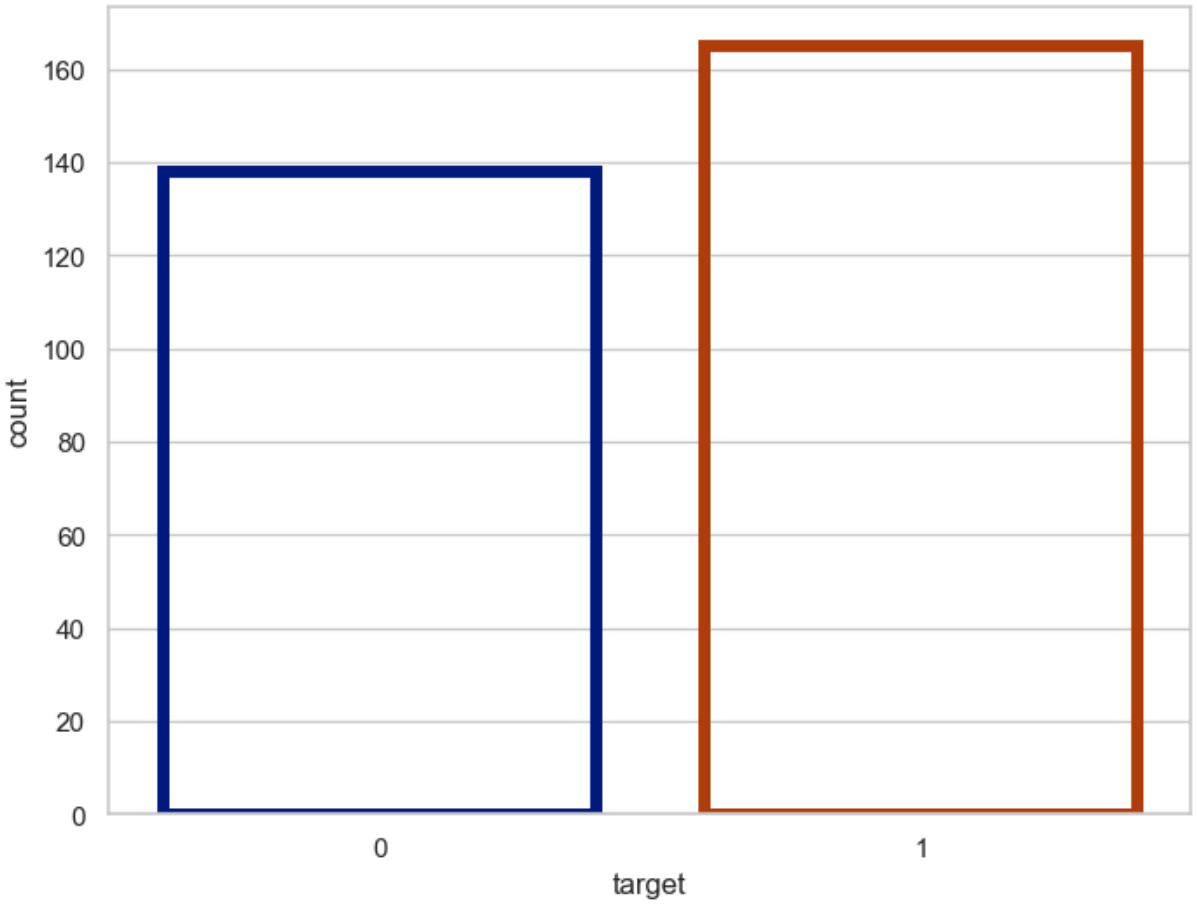
we can use different colour palette as follows:

In [32]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, palette="Set3")
plt.show()
```

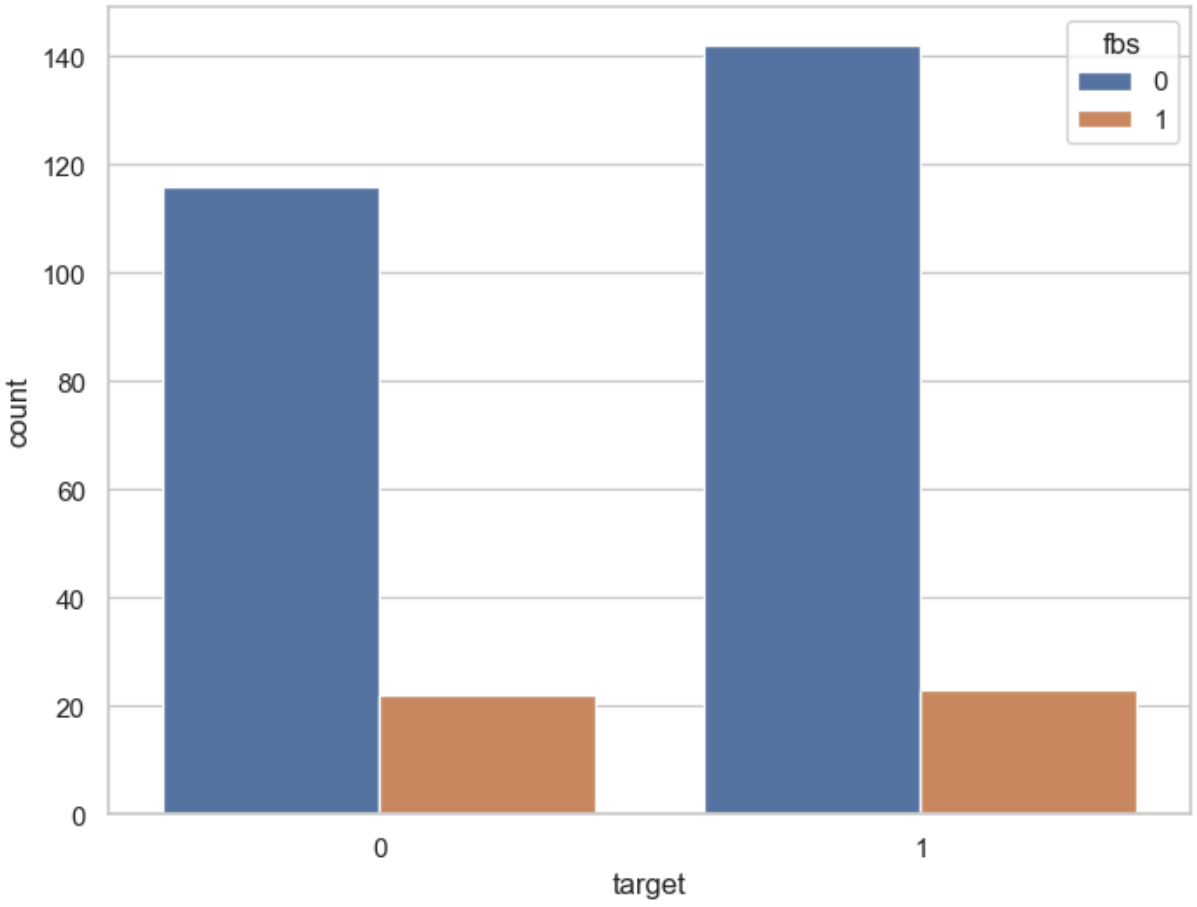


plt.bar keywords argumets for a different look

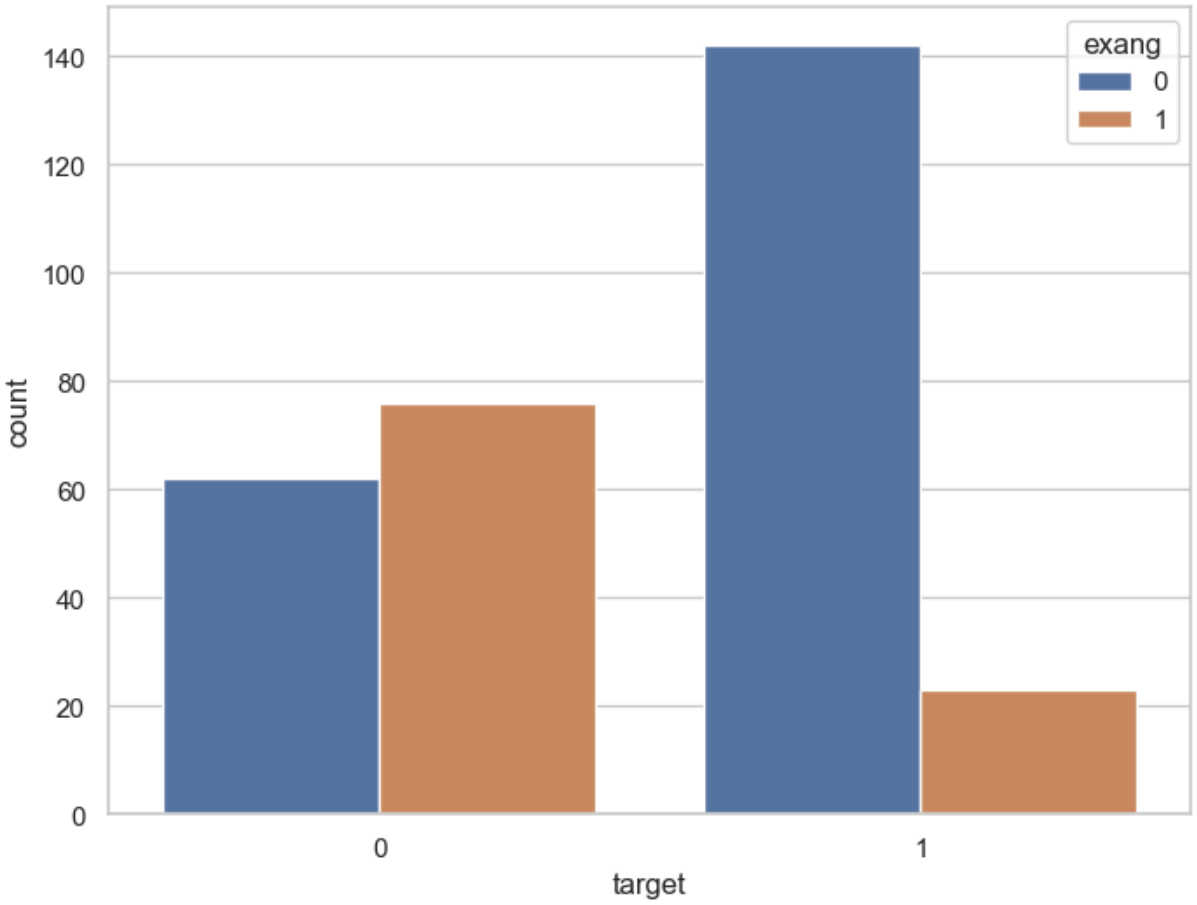
In [34]:
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.countplot(x="target", data=df, facecolor=(0,0,0,0),linewidth=5,edgecolor=s
plt.show()
```



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



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


exang
0
1
140
120
100
80
60
40
20
0
count
0
1
target

Findings of Univariate Analysis

Findings of univariate analysis are as follows:-

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

8.Bivariate Analysis

Estimate correaltion coefficients

In [39]:
```
correlation=df.corr()
```

In [40]:
```
correlation
```

Out[40]:

	age	sex	cp	trestbps	chol	fbs	restecg	thala
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421

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

Out[41]:
```
target      1.000000
cp          0.433798
thalach     0.421741
slope       0.345877
restecg     0.137230
fbs        -0.028046
chol       -0.085239
trestbps   -0.144931
age        -0.225439
sex        -0.280937
thal       -0.344029
ca         -0.391724
oldpeak    -0.430696
exang      -0.436757
Name: target, dtype: float64
```

analyses of target and cp variables

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

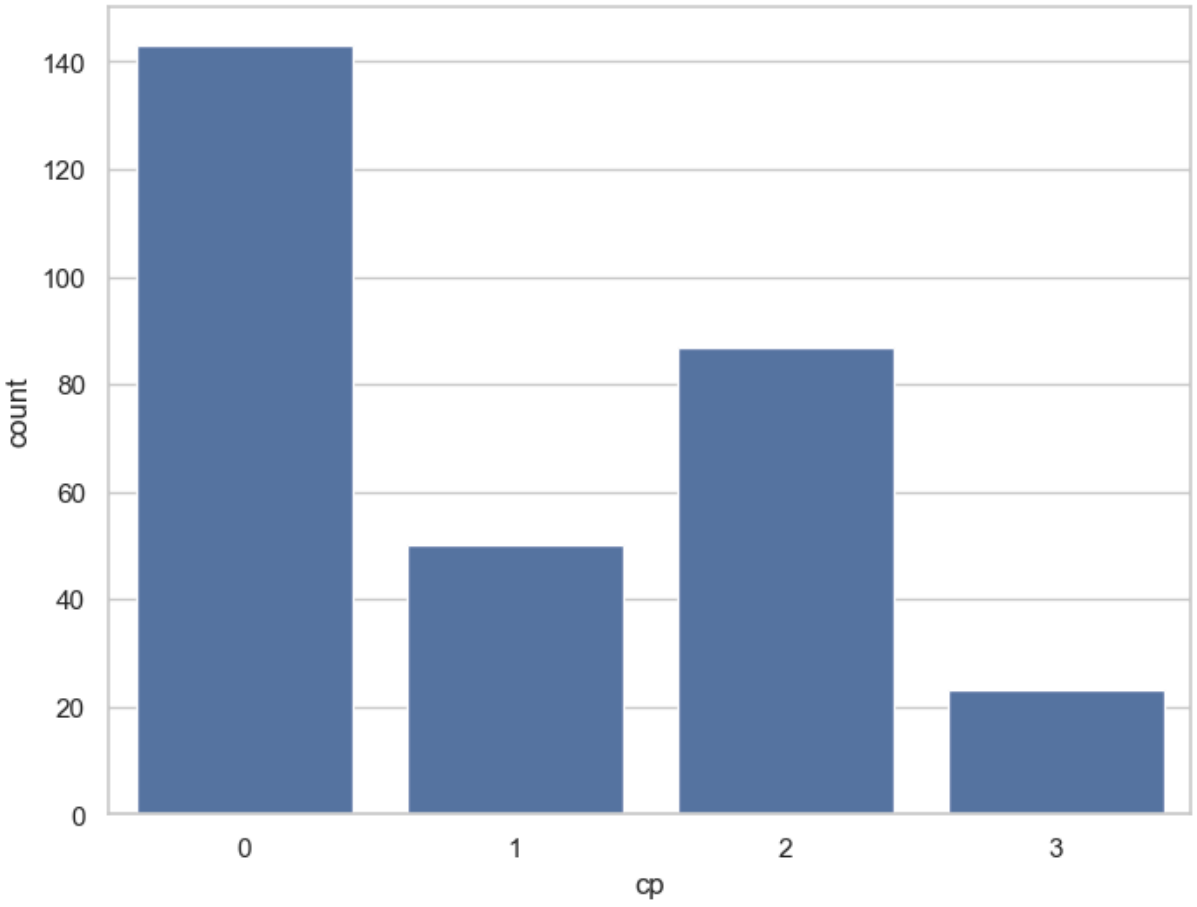
Out[43]:
```
4
```

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

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

visualize the frequency distribbution of cp variable

In [46]:
```
f,ax = plt.subplots(figsize=(8,6))
ax=sns.countplot(x="cp",data=df)
plt.show()
```



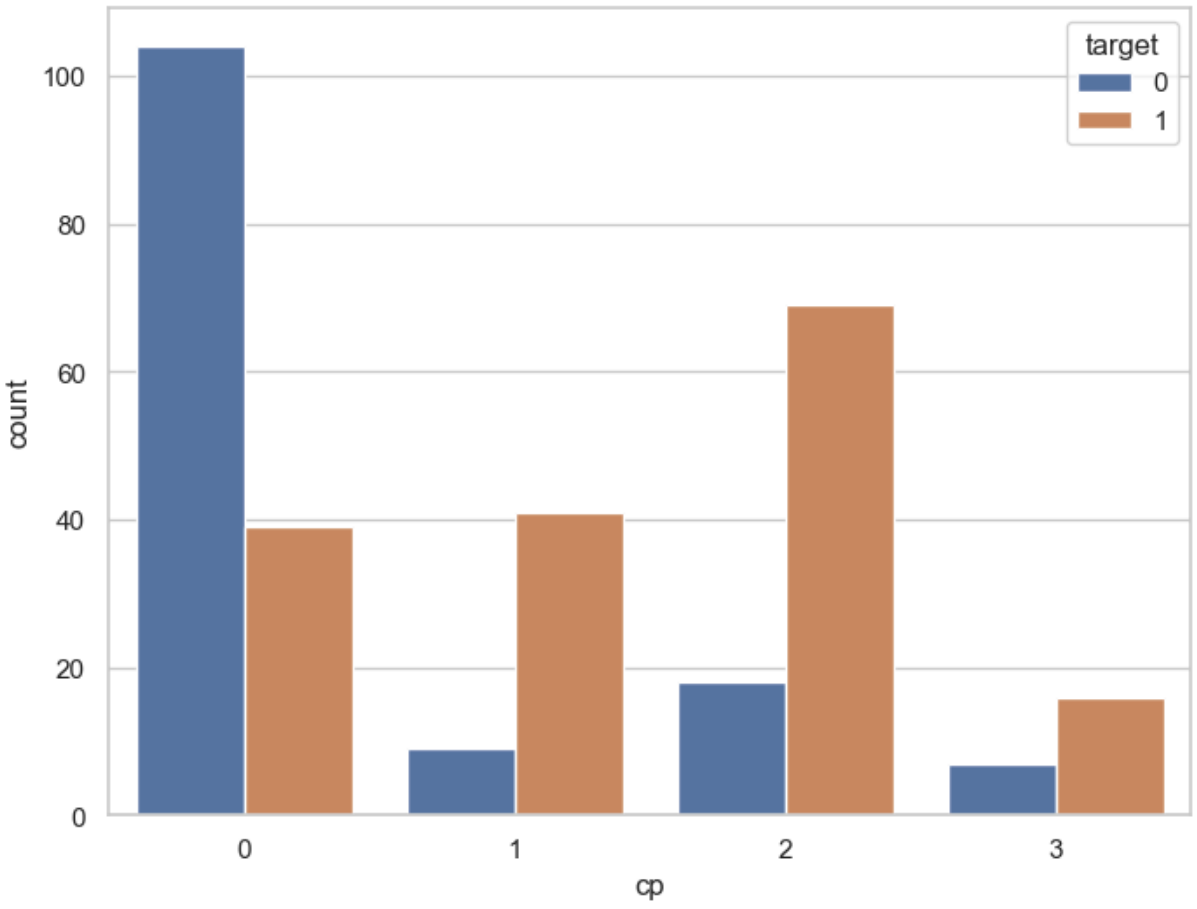
frequency distribution of target variable wrt cp

In [48]:
```
df.groupby('cp')['target'].value_counts()
```

Out[48]:
```
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
```

we can visualise the value counts of cp variable wrt target as folloows

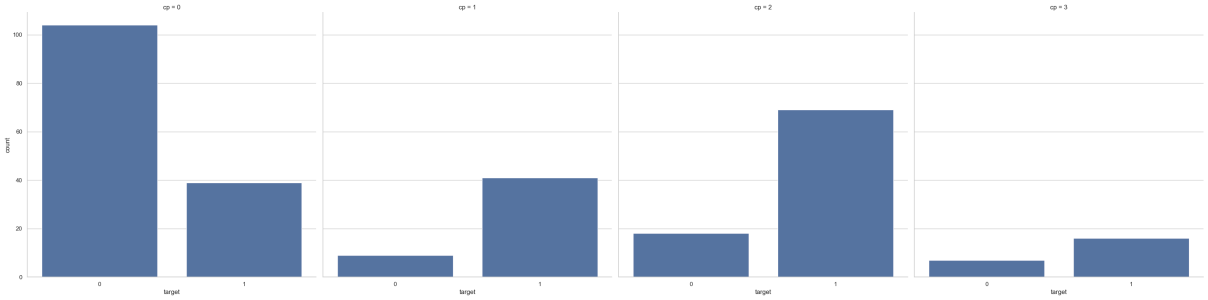
In [50]:
```
f,ax =plt.subplots(figsize=(8,6))
ax=sns.countplot(x="cp",hue="target",data=df)
plt.show()
```



alternatively, we can visualise the same information as follows

In [52]:
```
ax=sns.catplot(x="target",col="cp",data=df, kind="count", height=8, aspect=1)
```

In [53]:
```
plt.show()
```



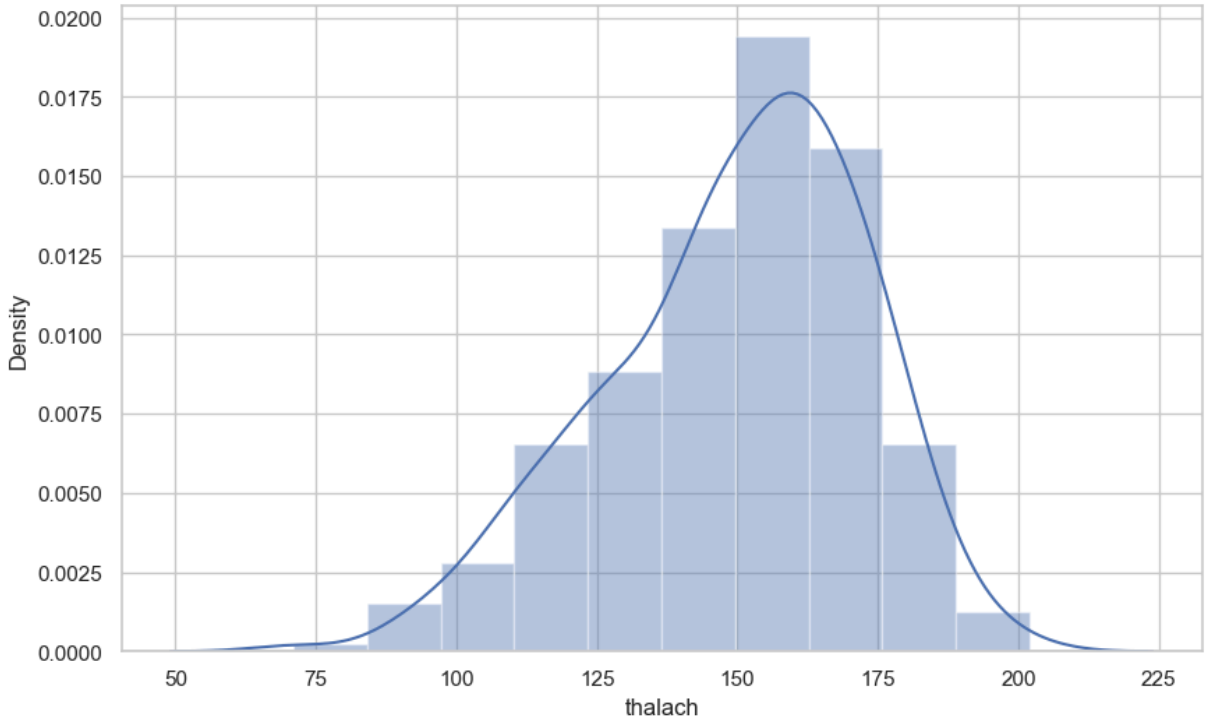
analysis of target and thalach variable

In [55]:
```
df['thalach'].nunique()
```

Out[55]: 91

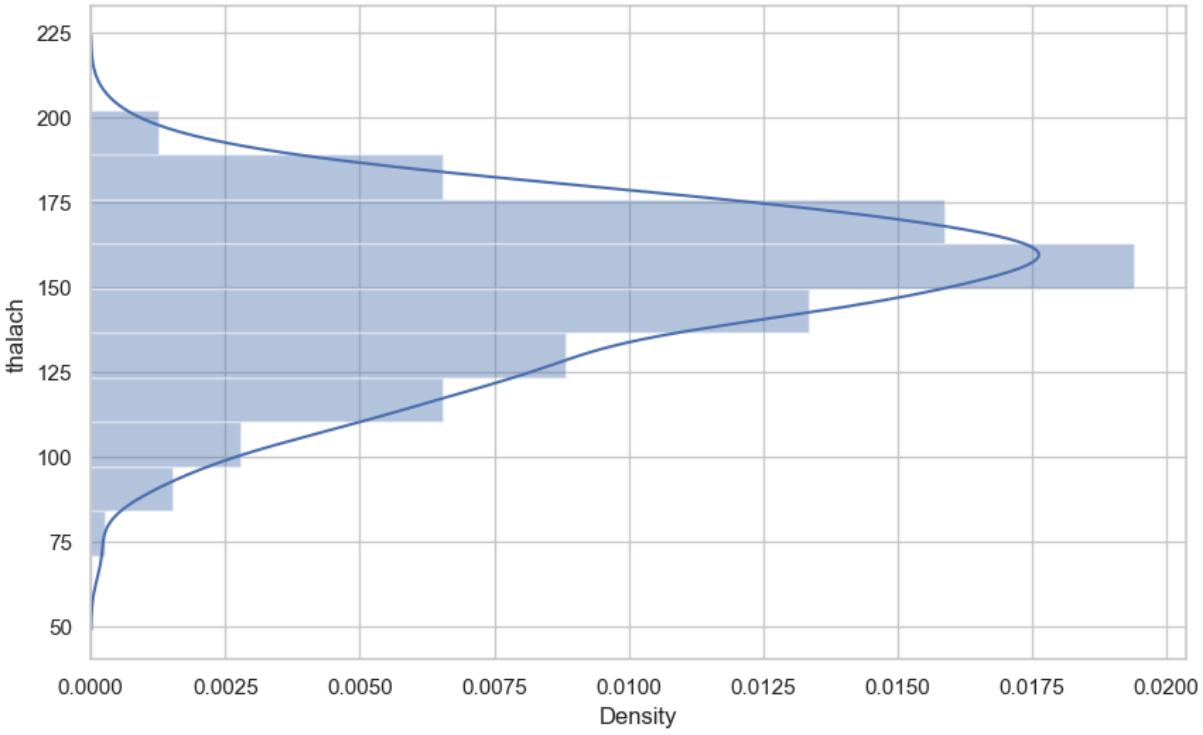
distribution of thalach variable

In [57]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
ax=sns.distplot(x,bins=10)
plt.show()
```



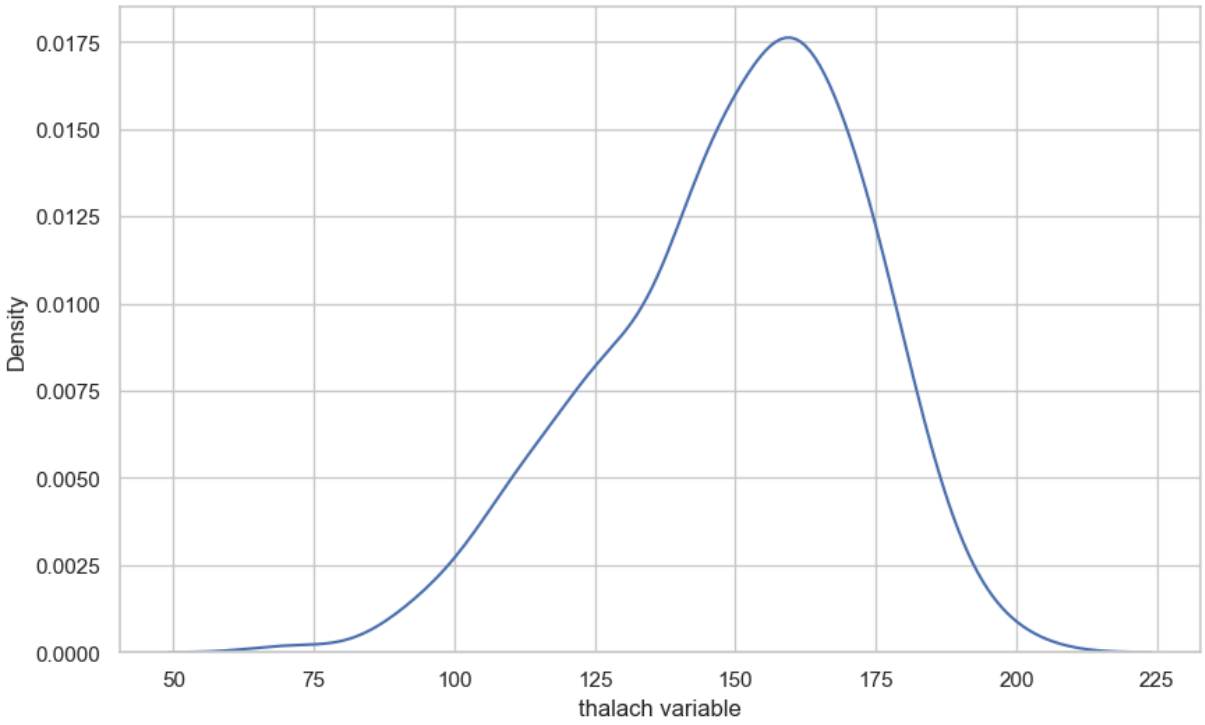
plot the distribution in vertical

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

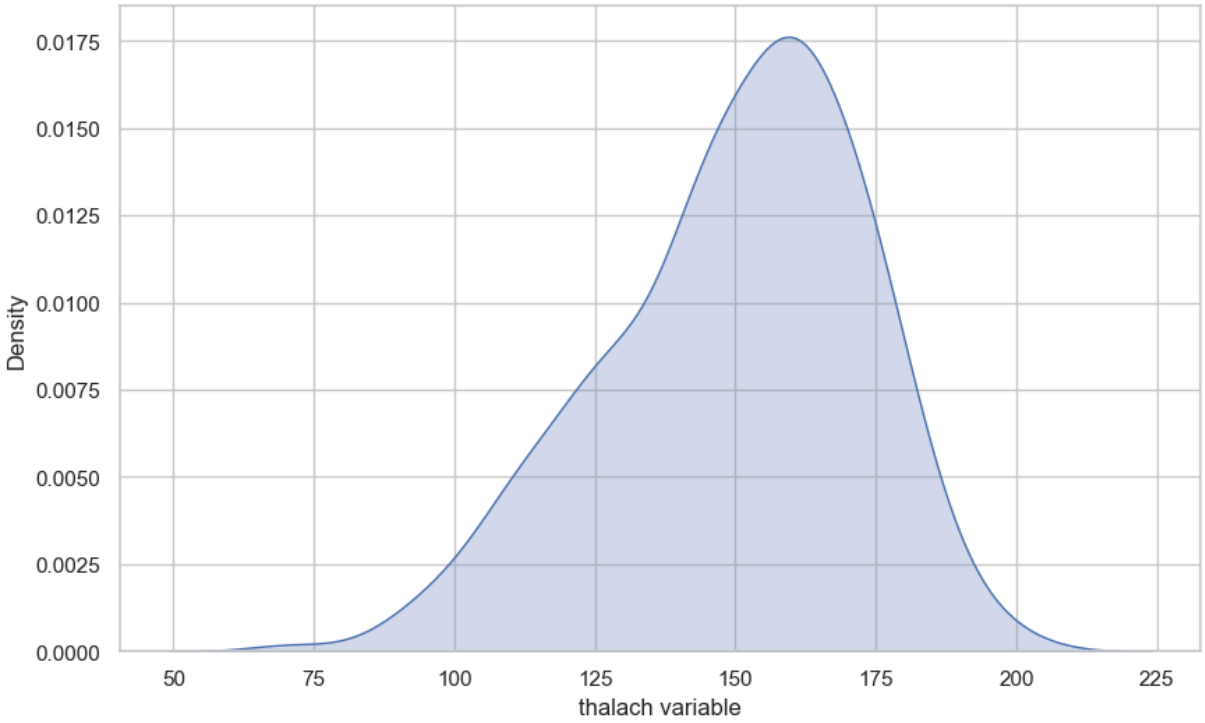


seaborn kernel density distribution(KDE) plot

In [96]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name="thalach variable")
ax=sns.kdeplot(x)
plt.show()
```

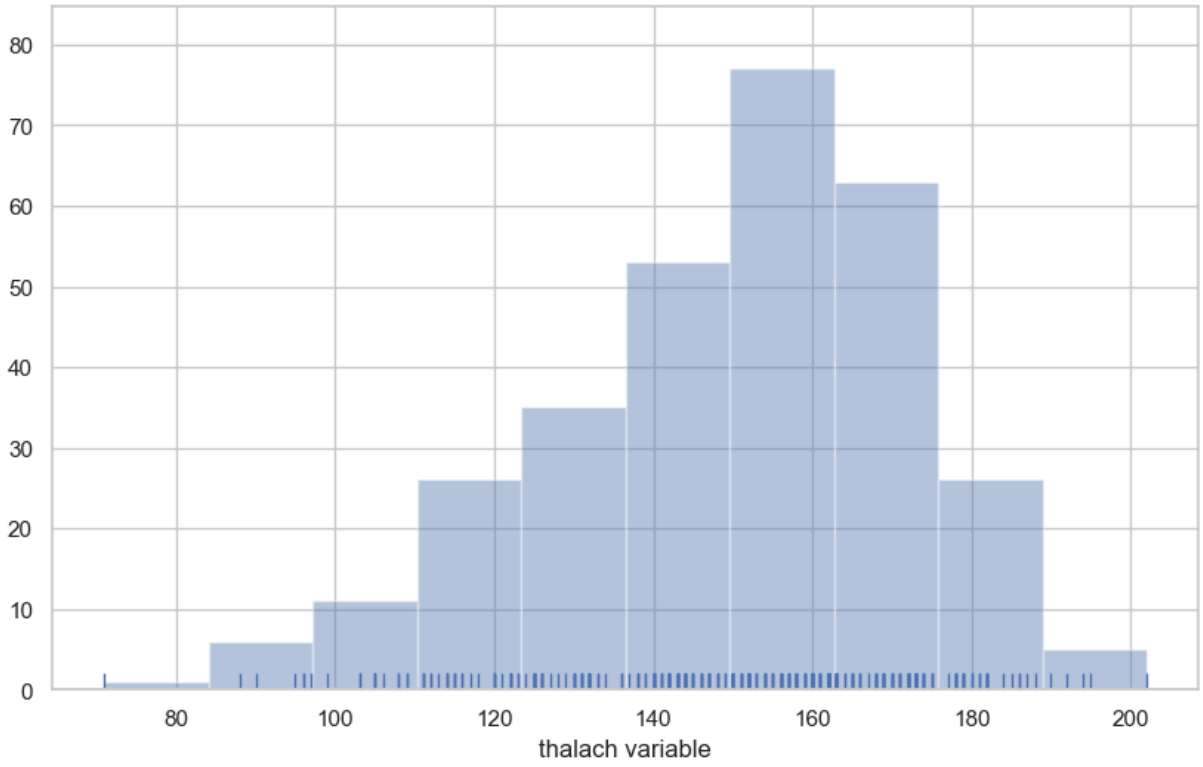


In [98]:
```
f,ax=plt.subplots(figsize=(10,6))
x=df['thalach']
x=pd.Series(x,name="thalach variable")
ax=sns.kdeplot(x,shade=True,color="b")
plt.show()
```



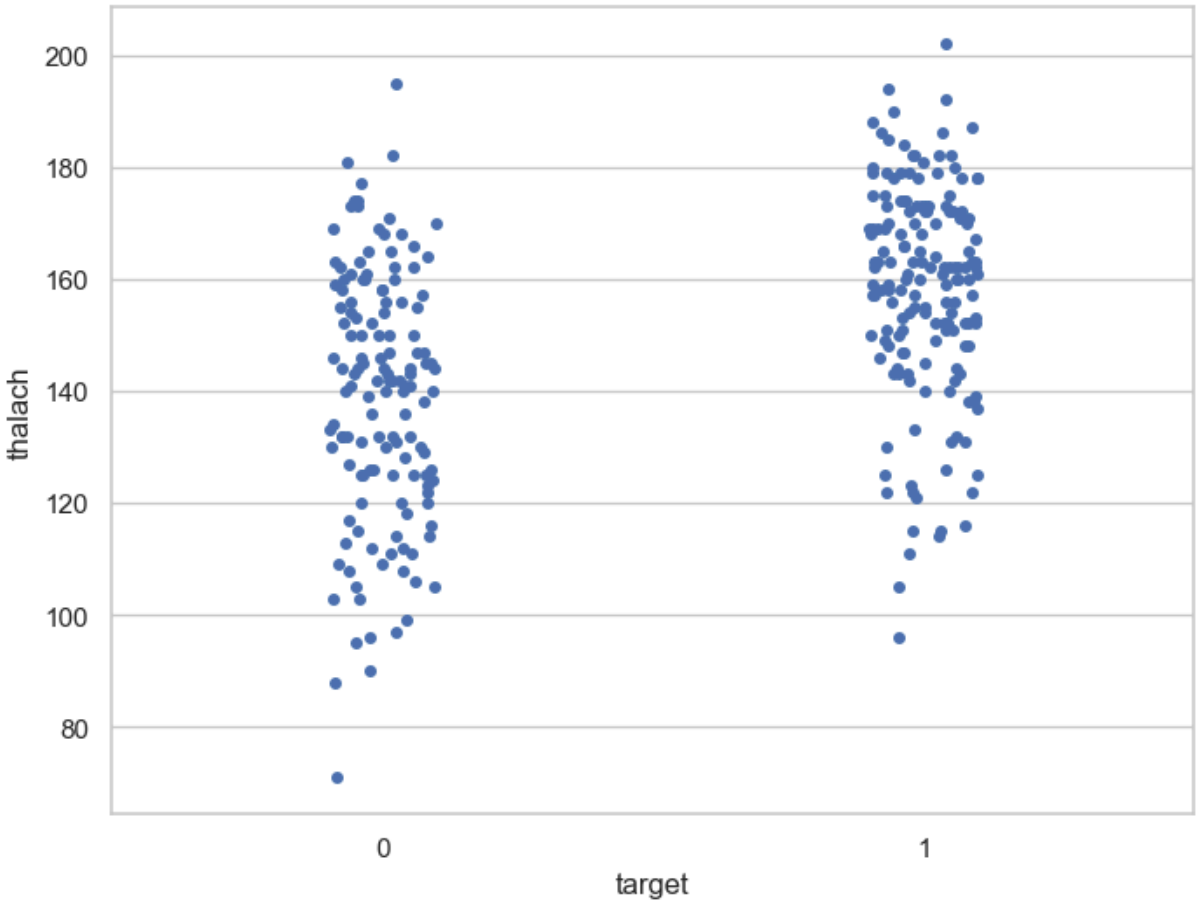
Histogram

In [101…
```
f,ax =plt.subplots(figsize=(10,6))
x-df['thalach']
ax=sns.distplot(x,kde=False,rug=True,bins=10)
plt.show()
```



visualise frequency distribution of thalach variable wrt to target

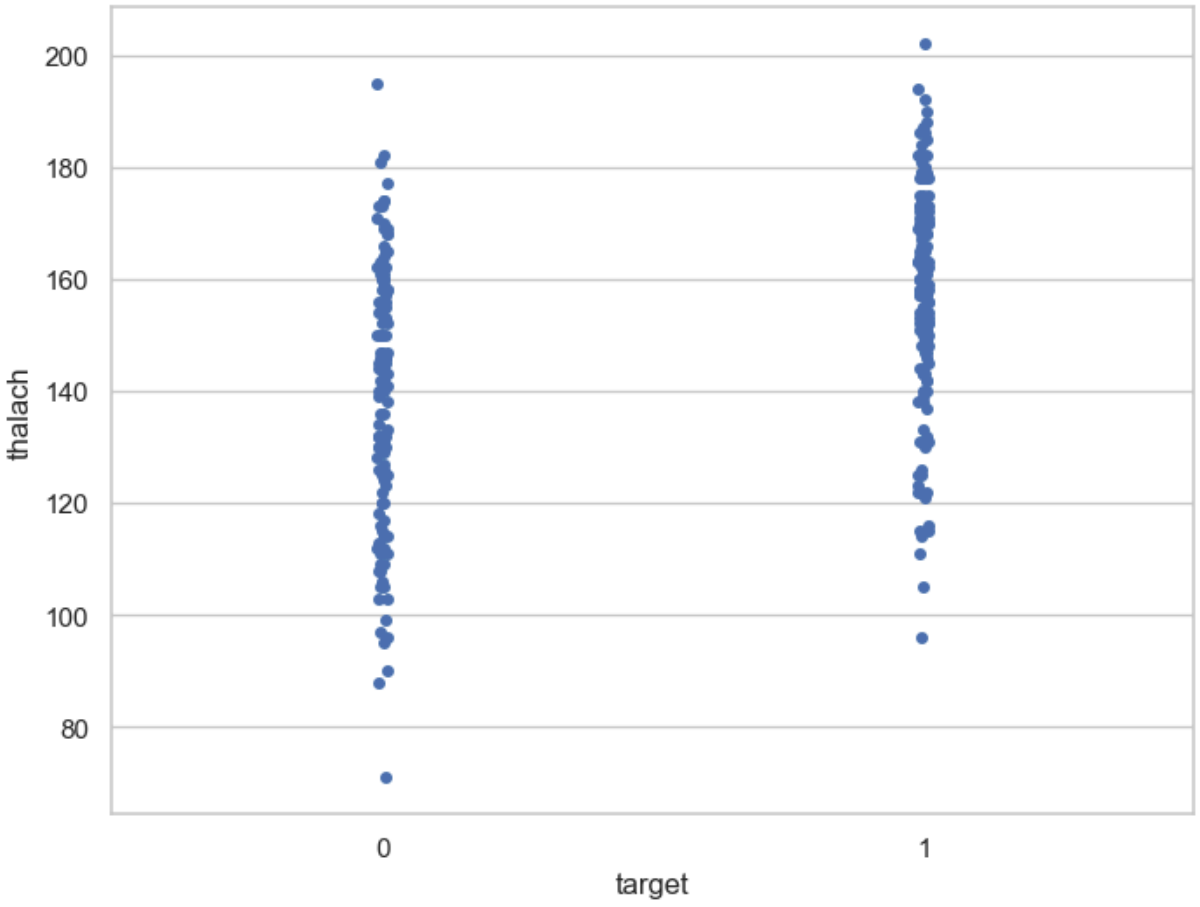
In [104…
```
f,ax  = plt.subplots(figsize=(8,6))
sns.stripplot(x="target",y="thalach",data=df)
plt.show()
```



interpretation

In [108…

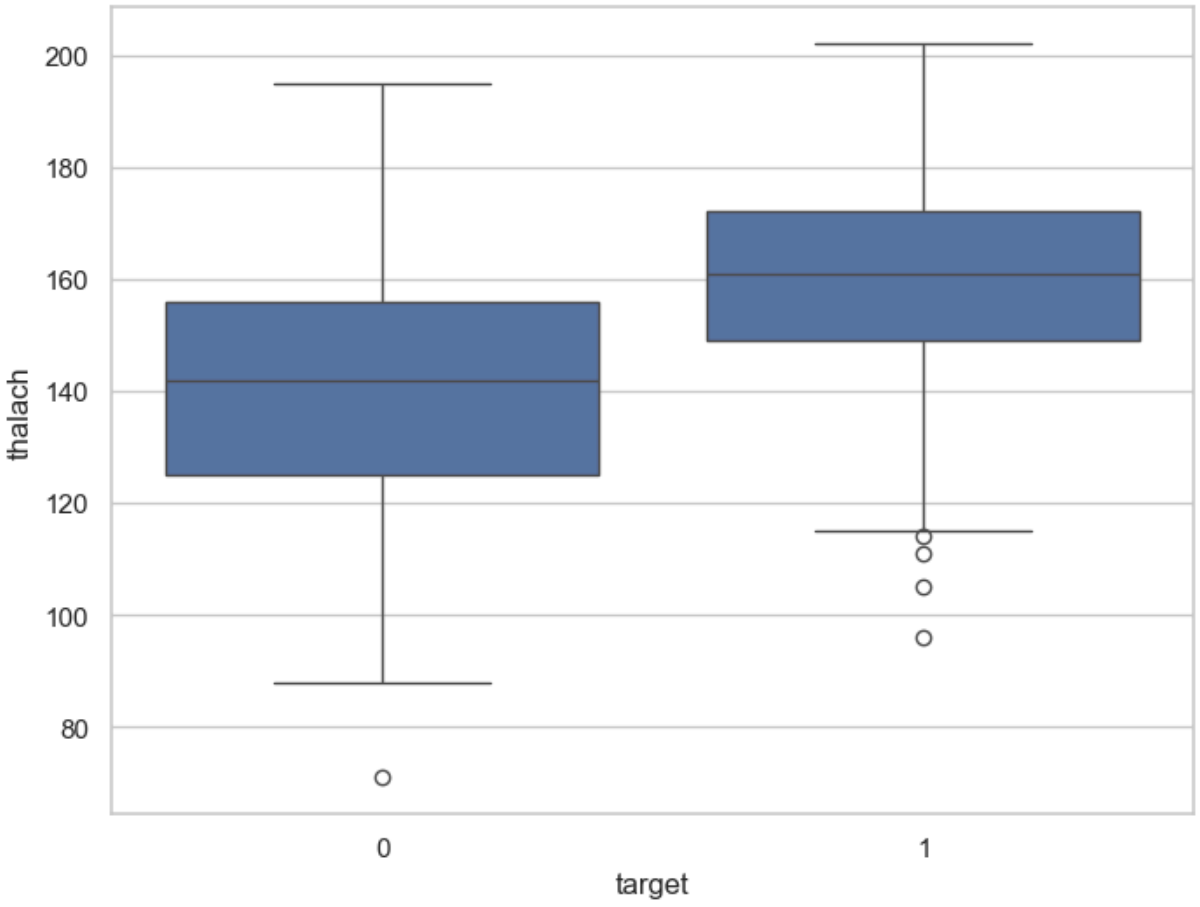
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="thalach", data=df, jitter = 0.01)
plt.show()
```



visualising wth box plot

In [111…

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



Findings of Bivariate Analysis

Findings of Bivariate Analysis are as follows –

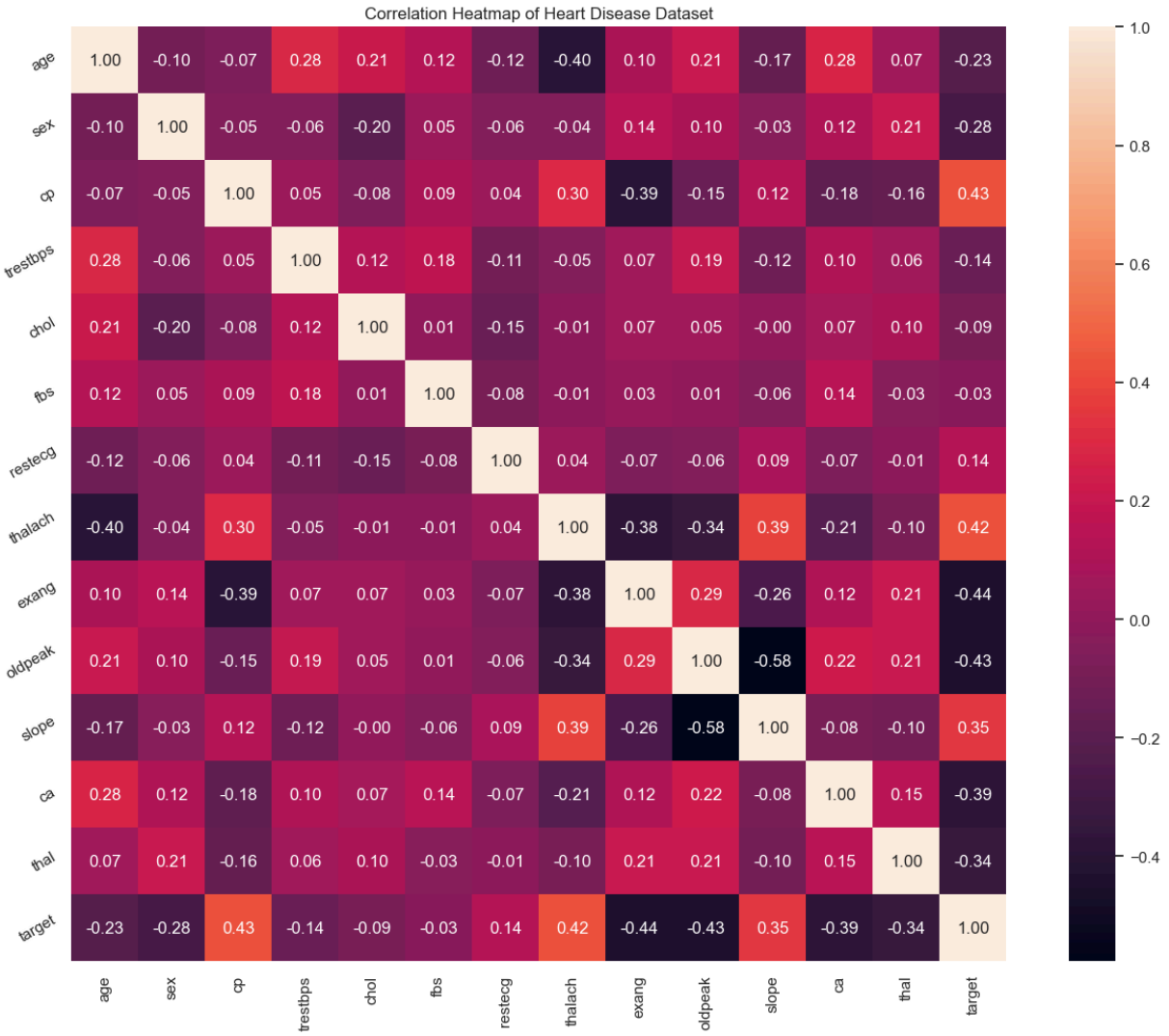
- There is no variable which has strong positive correlation with `target` variable.
- There is no variable which has strong negative correlation with `target` variable.
- There is no correlation between `target` and `fbs`.
- The `cp` and `thalach` variables are mildly positively correlated with `target`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

8. MULTIVARIATE ANALYSIS

Heat Map

In [117…

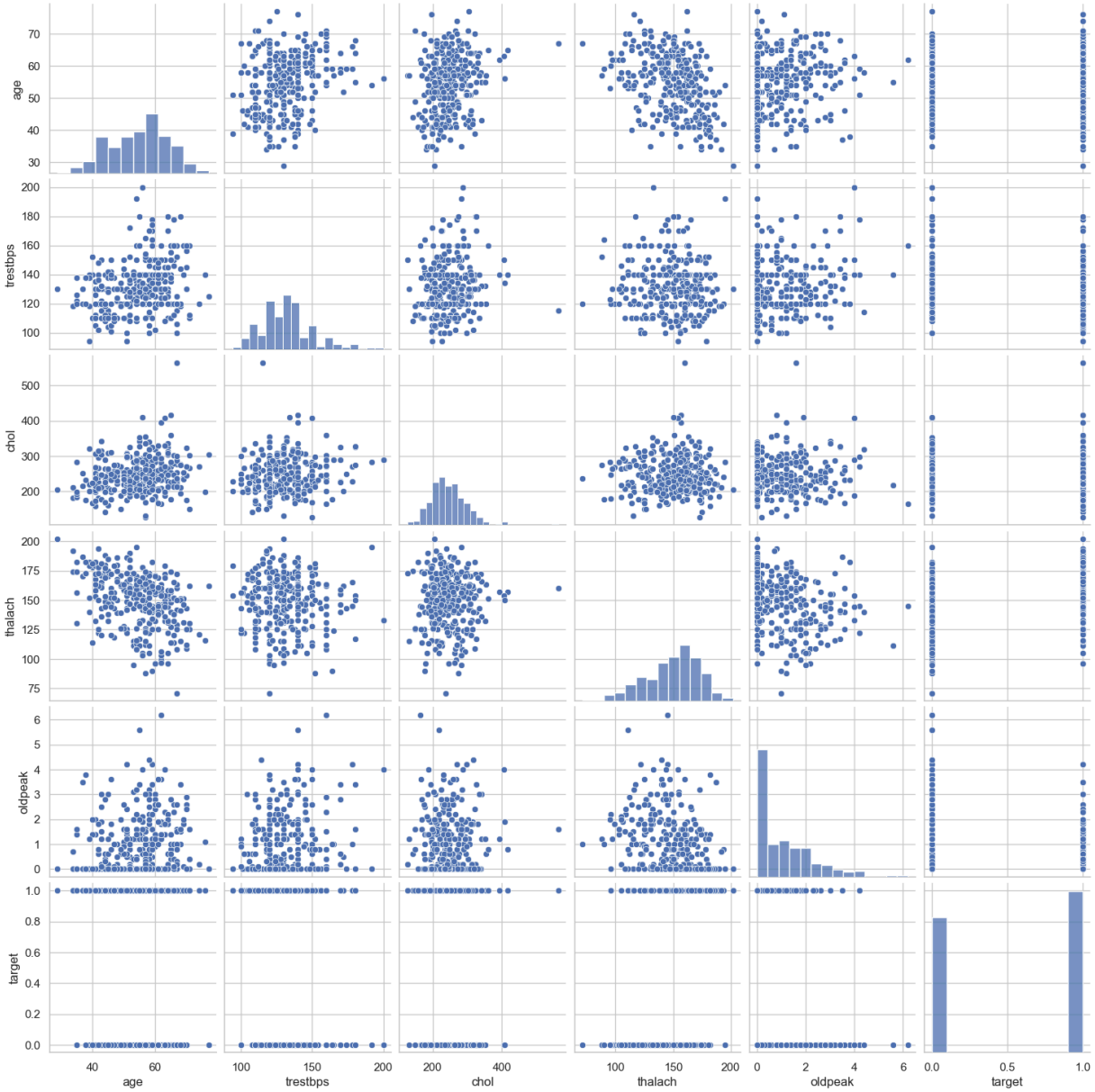
```
plt.figure(figsize=(16,12))
plt.title('Correlation Heatmap of Heart Disease Dataset')
a = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='white')
a.set_xticklabels(a.get_xticklabels(), rotation=90)
a.set_yticklabels(a.get_yticklabels(), rotation=30)
plt.show()
```



pair plot

In [120…

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



analysis of age and other variables

In [123…
```
df['age'].nunique()
```

Out[123…
```
41
```

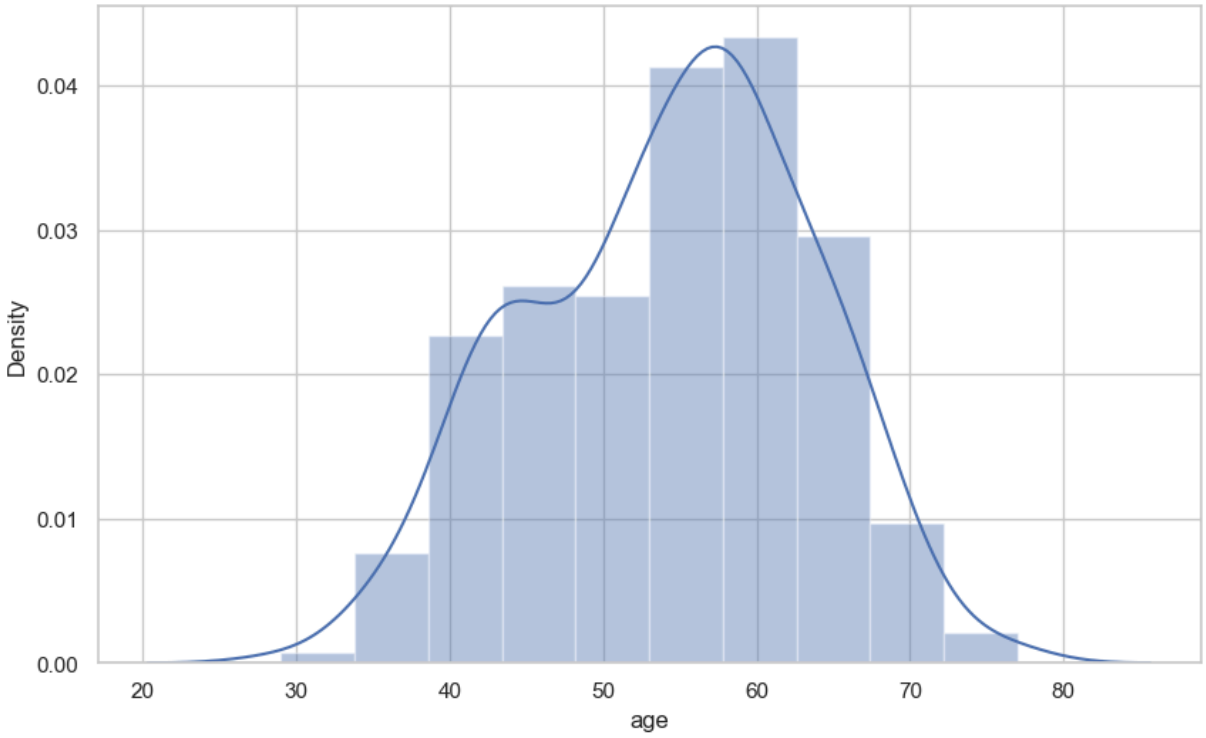
statistical summary of age variable

In [126…
```
df['age'].describe()
```

Out[126…
```
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
```

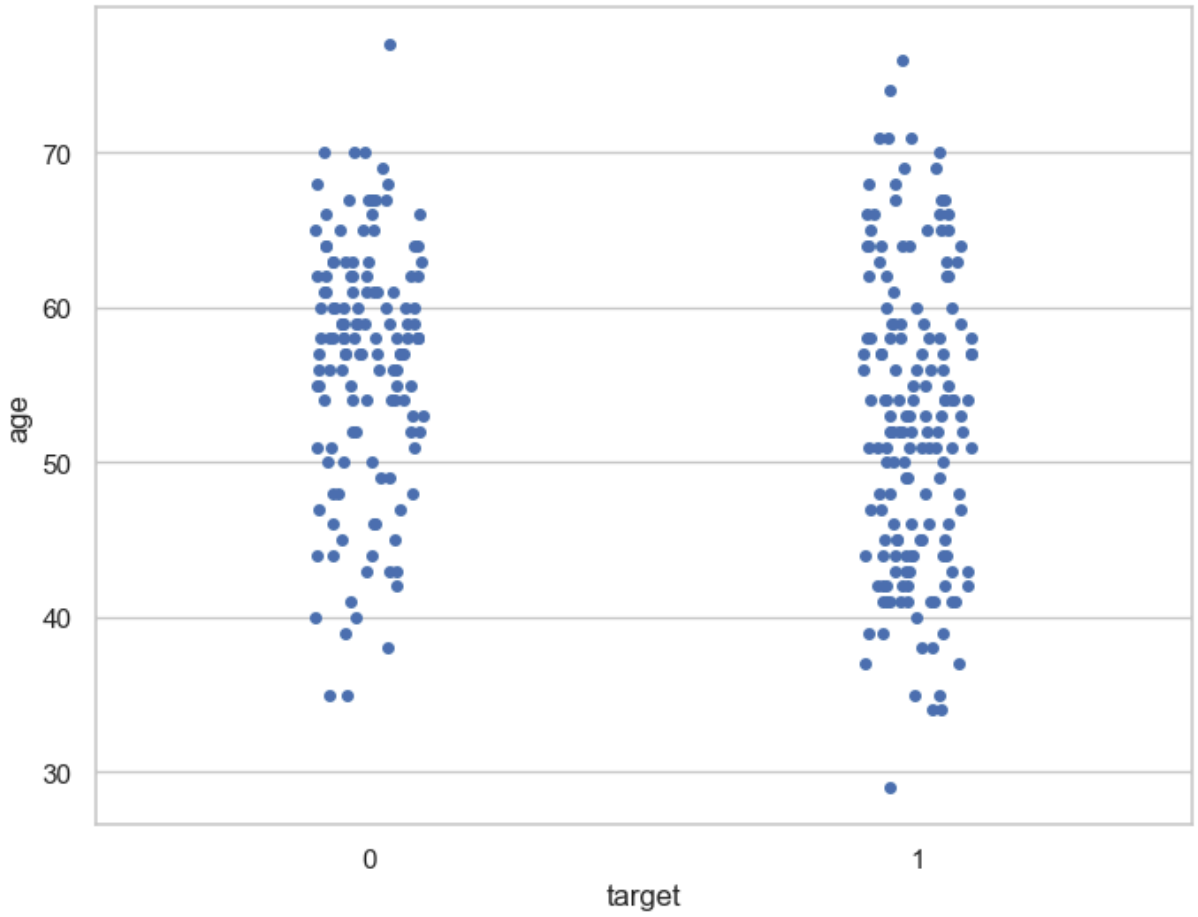
plot the distribution of age variable

In [129…
```
f, ax = plt.subplots(figsize=(10,6))
x = df['age']
ax = sns.distplot(x, bins=10)
plt.show()
```



visualise frequency distribution of age variable wrt target

In [132…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="age", data=df)
plt.show()
```

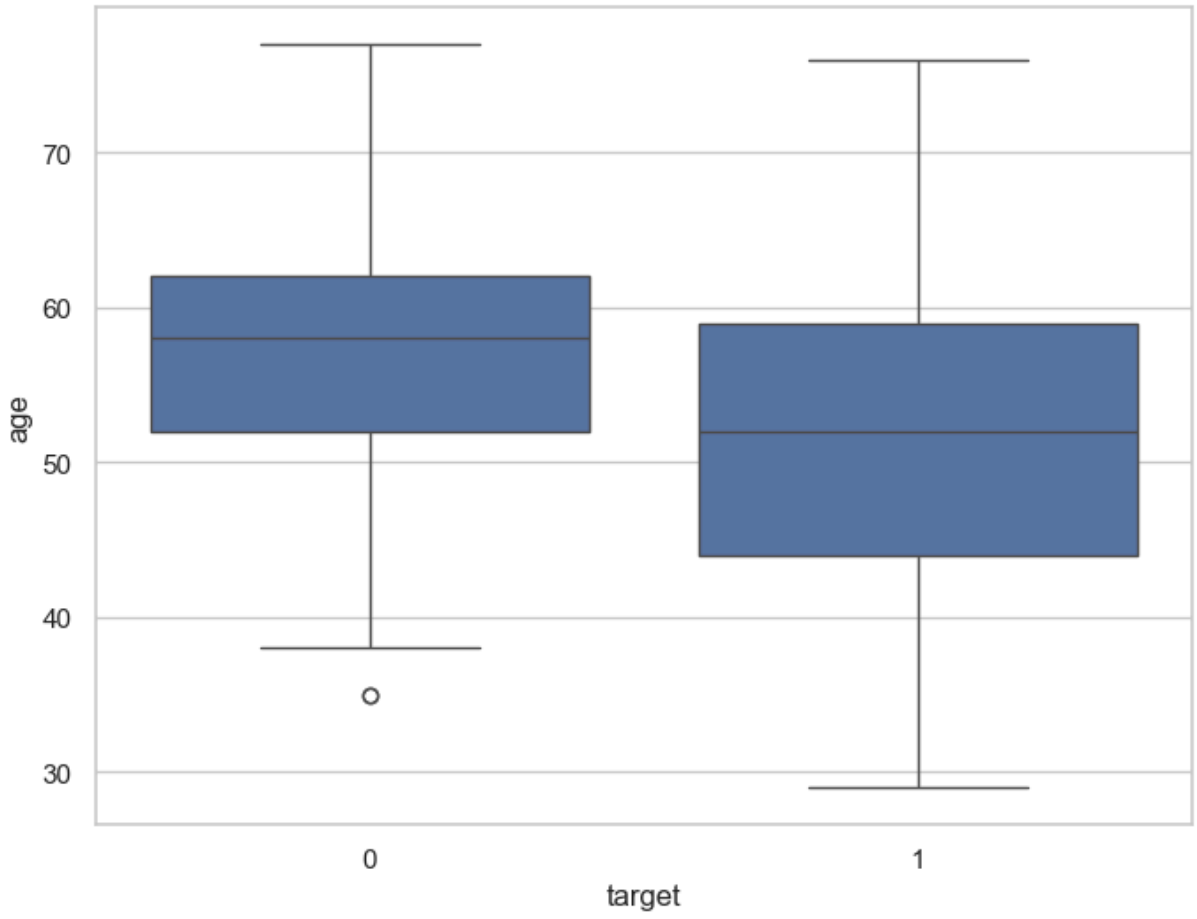


visualise distribution of age variable wrt target wth boxplot

In [139…

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

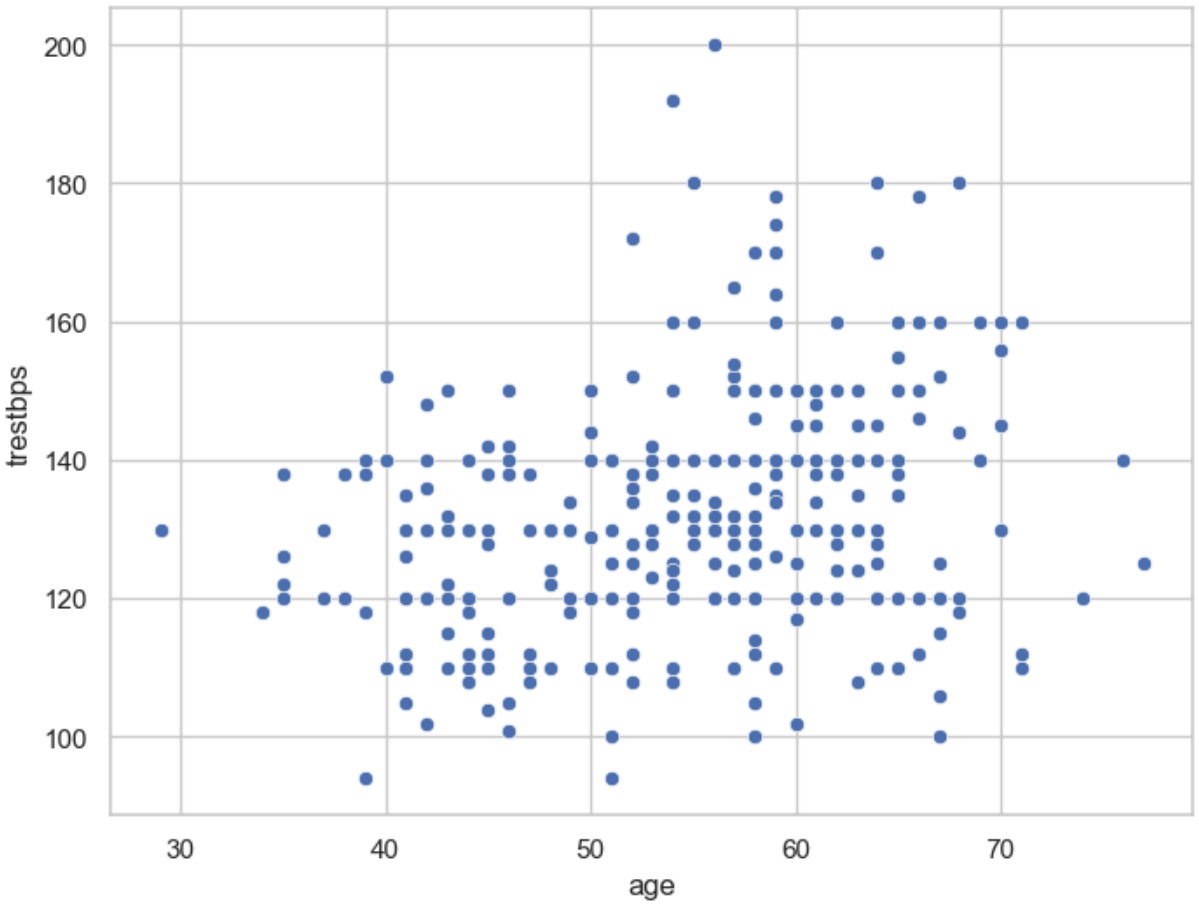

1.0
0.8
0.6
0.4
0.2
0.0
0


70
60
50
40
30
age
0
1
target

Analyse the age and trestbps variable

In [142…

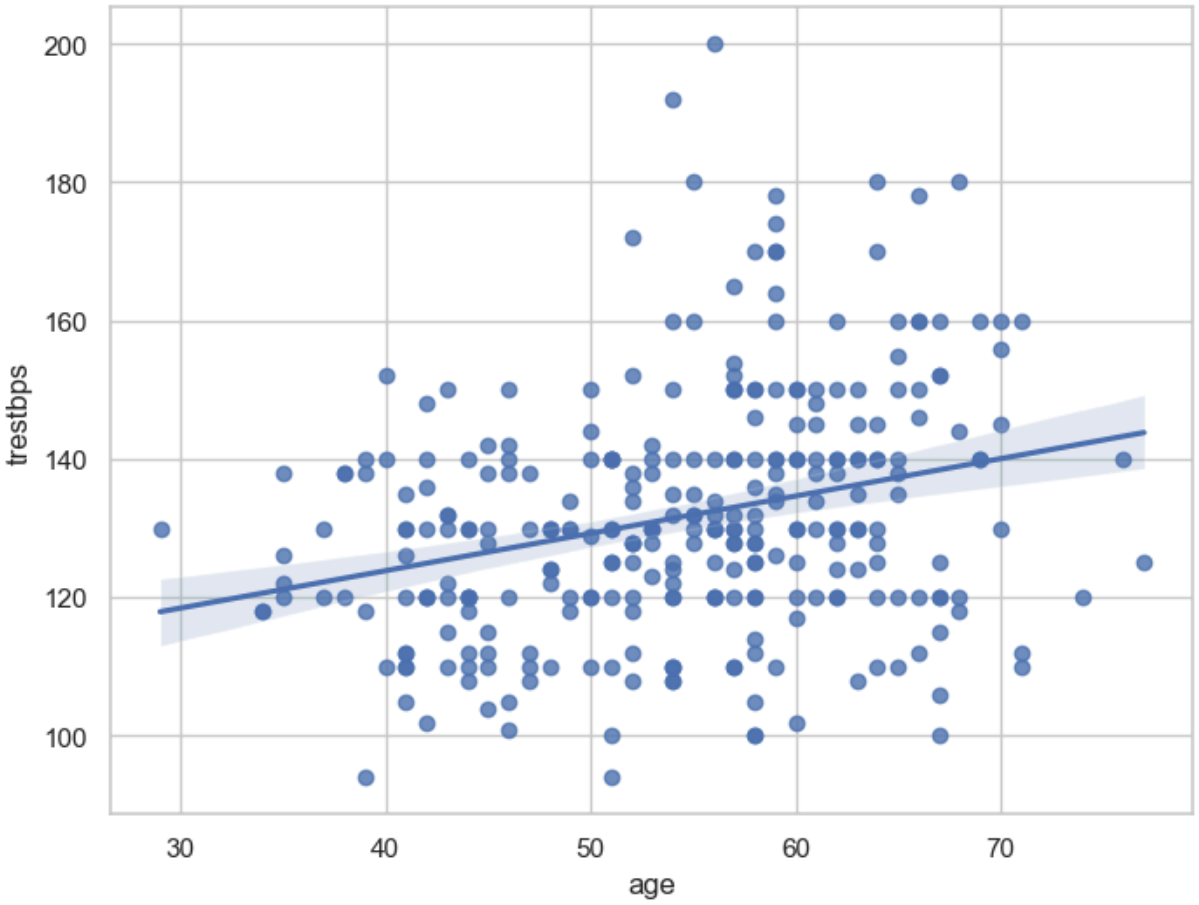
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.scatterplot(x="age", y="trestbps", data=df)
plt.show()
```



interpretation

In [147…

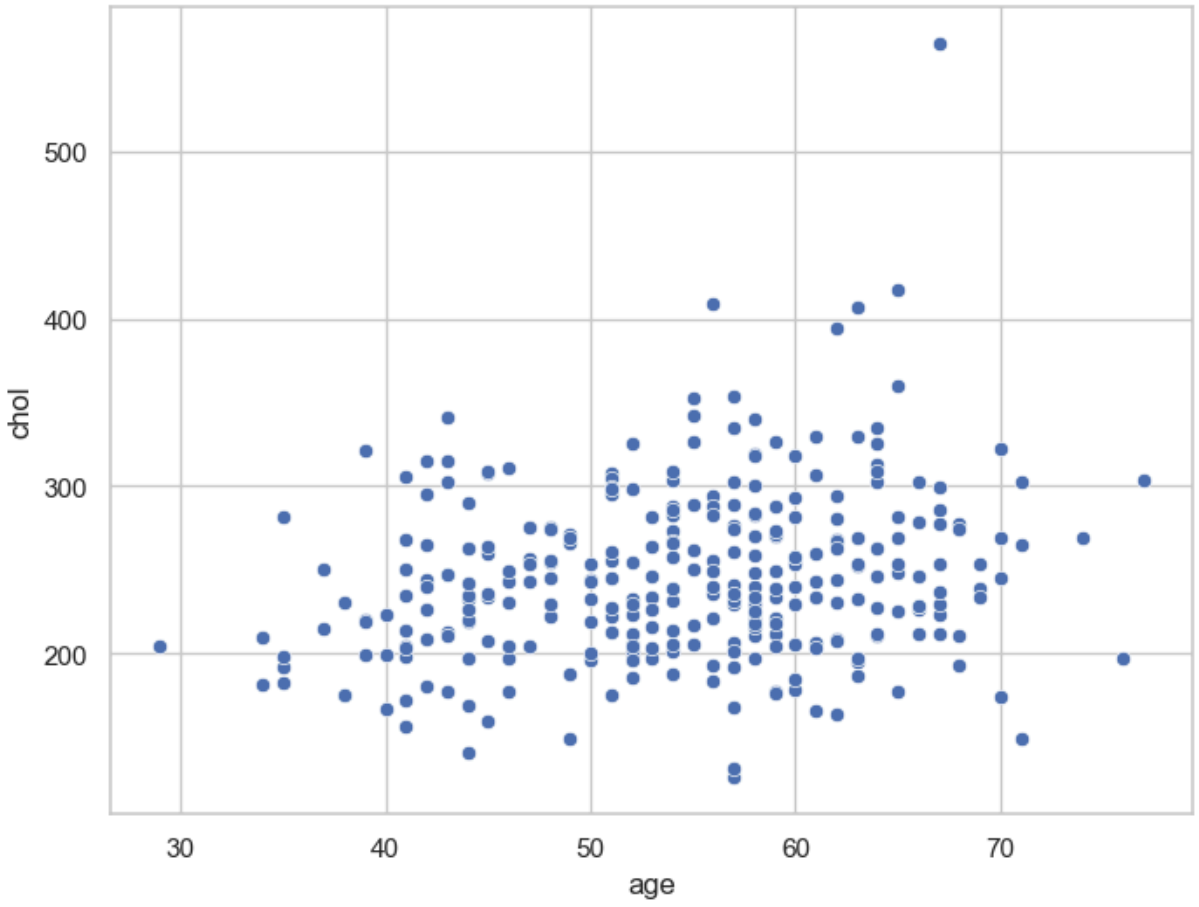
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="trestbps", data=df)
plt.show()
```



Analyse age and chol variable

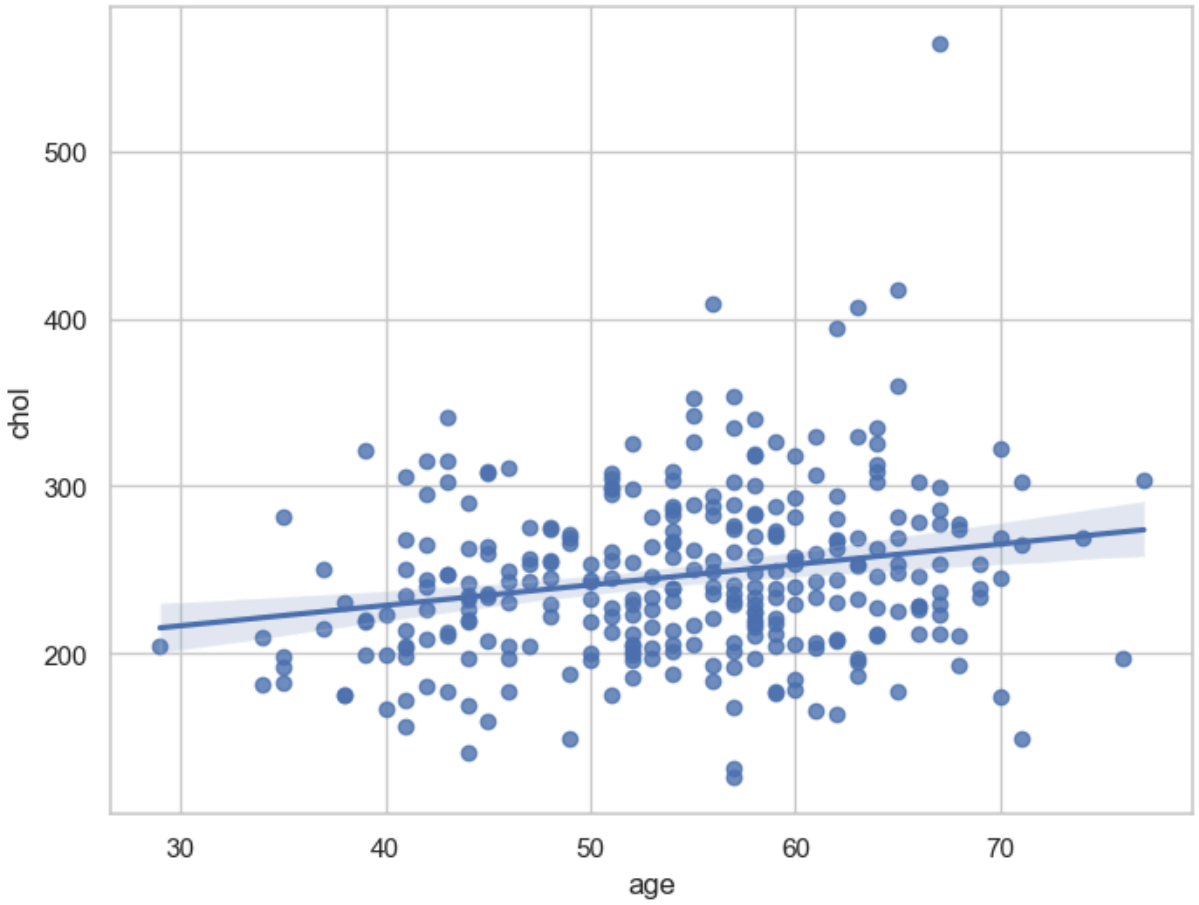
In [150…

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



In [152…

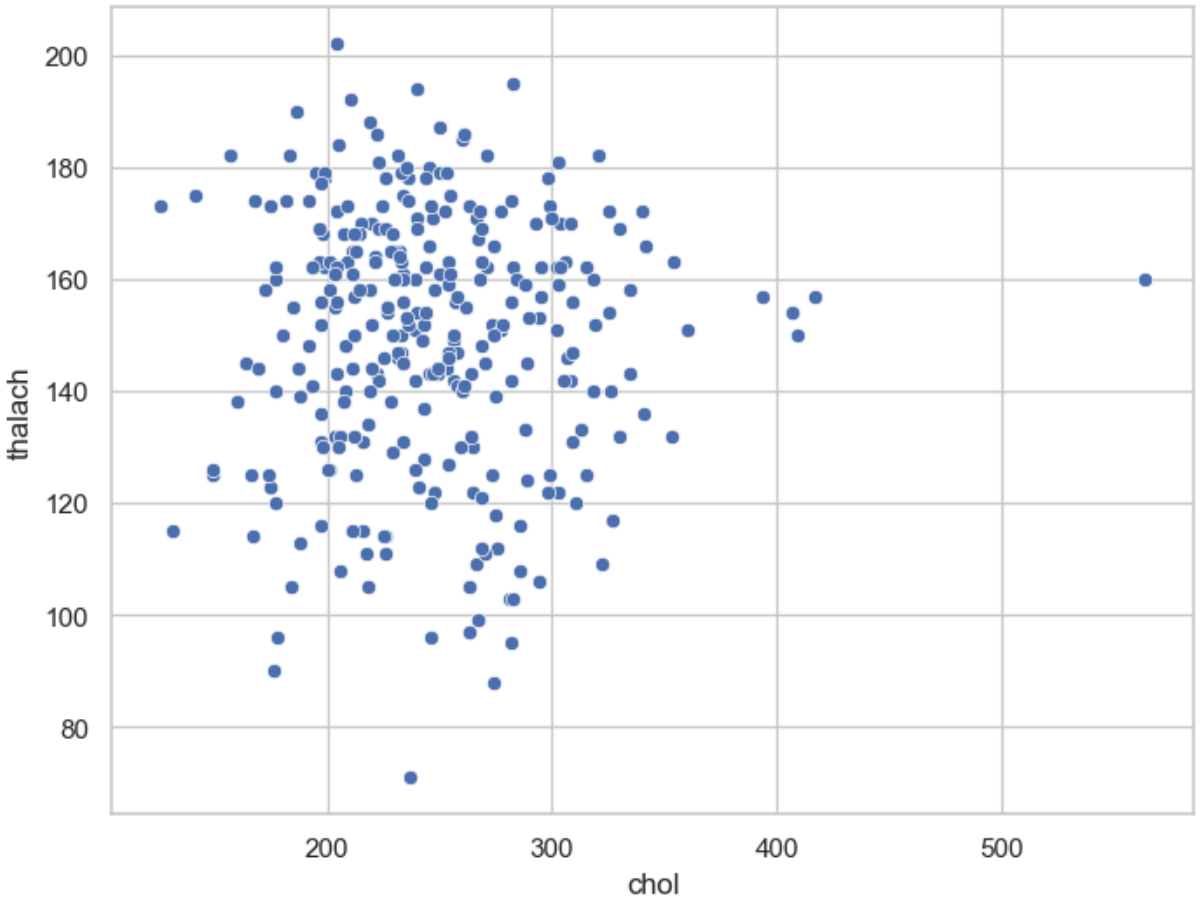
```
f, ax = plt.subplots(figsize=(8, 6))
ax = sns.regplot(x="age", y="chol", data=df)
plt.show()
```



analyse chol and thalach variable

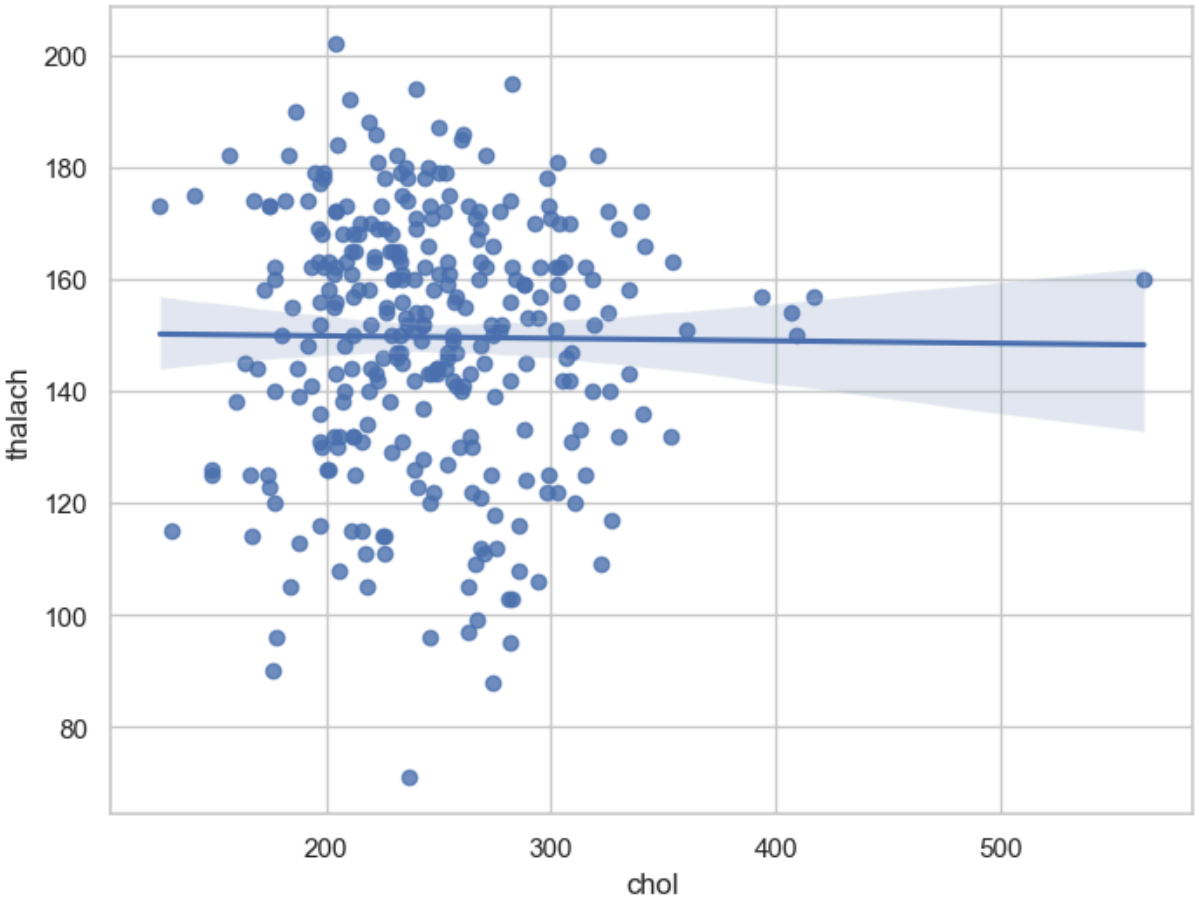
In [155…

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



In [157…

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



10.dealing with ,missing values

10. Dealing with missing values

Back to Table of Contents

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

Pandas isnull() and notnull() functions

- Pandas offers two functions to test for missing data - `isnull()` and `notnull()`. These are simple functions that return a boolean value indicating whether the passed in argument value is in fact missing data.

- Below, I will list some useful commands to deal with missing values.

Useful commands to detect missing values

- **df.isnull()**

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

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

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

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

It returns total number of missing values in the dataframe.

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

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

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

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

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

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

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

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

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

It returns the total number of missing values in the dataframe.

In [162…
```
df.isnull().sum()
```

Out[162…
```
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
```

11.check with ASSERT statement

11. Check with ASSERT statement

Back to Table of Contents

- 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 [168…
```
assert pd.notnull(df).all().all()
```

In [170…
```
assert (df >= 0).all().all()
```

12. Outlier detection

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

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

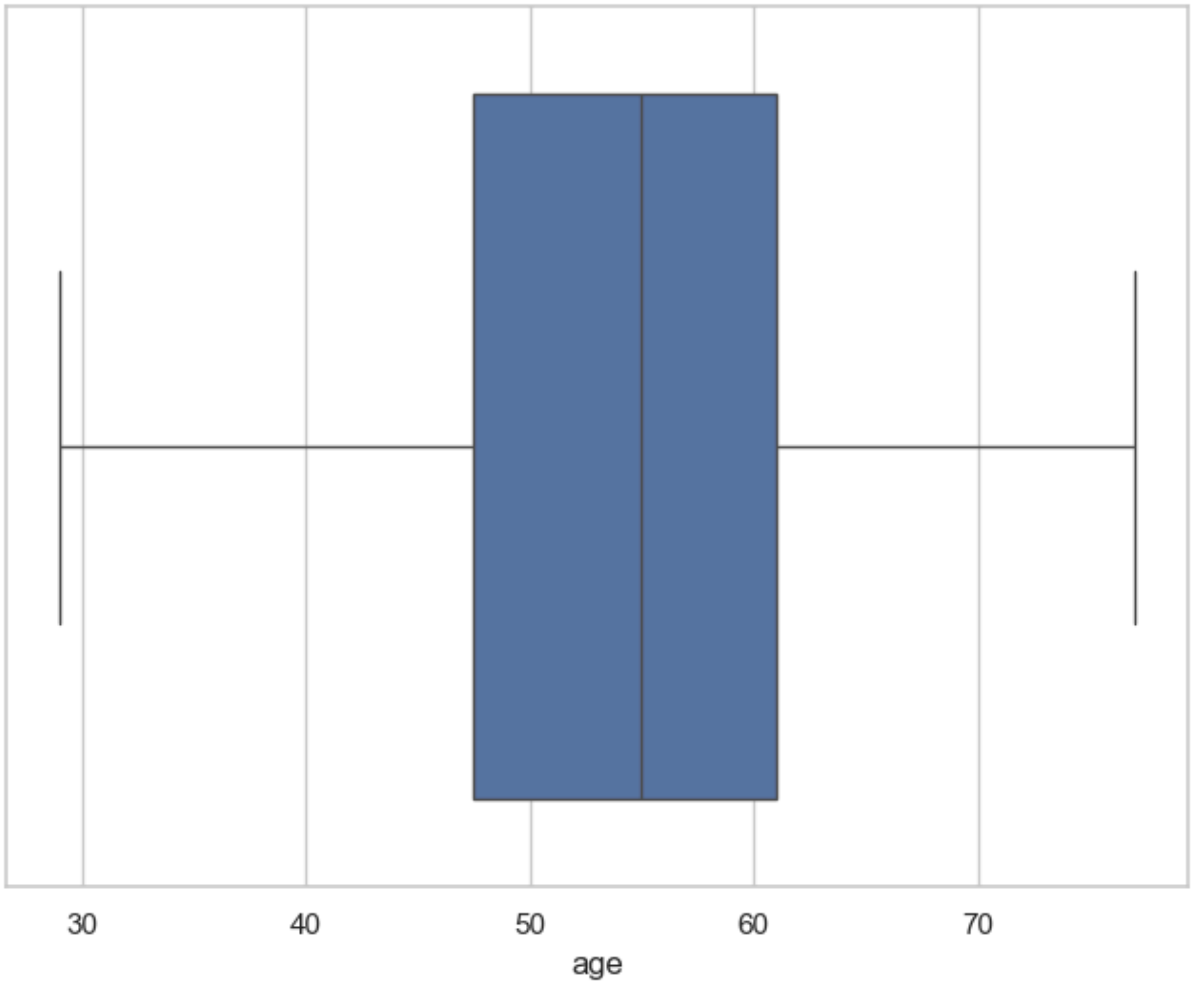
age variable

In [175…
```
df['age'].describe()
```

Out[175…
```
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
```

Boxplot of age variable

In [178…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["age"])
plt.show()
```



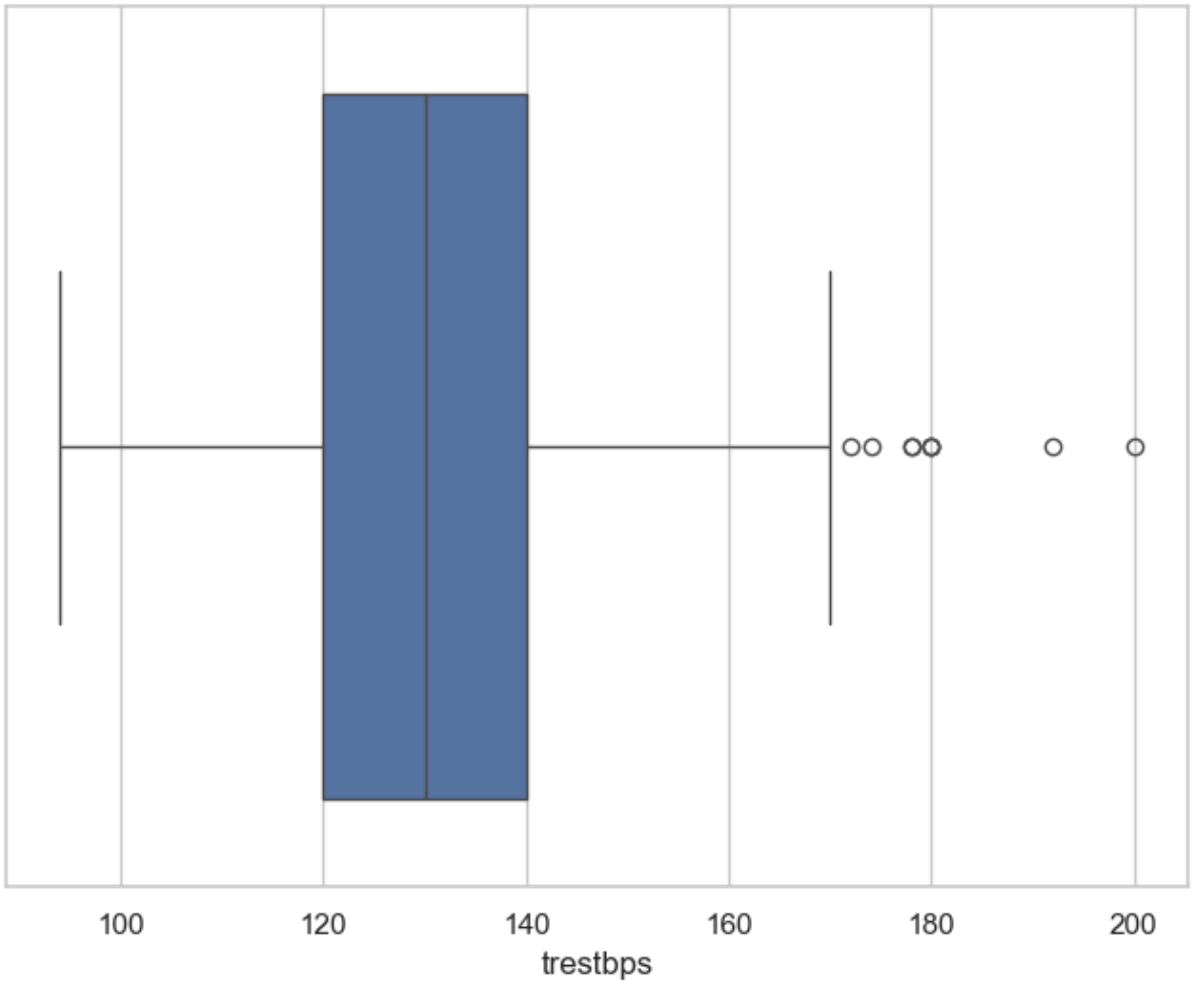
trestbps variable

In [181…
```
df['trestbps'].describe()
```

Out[181…
```
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 [184…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["trestbps"])
plt.show()
```



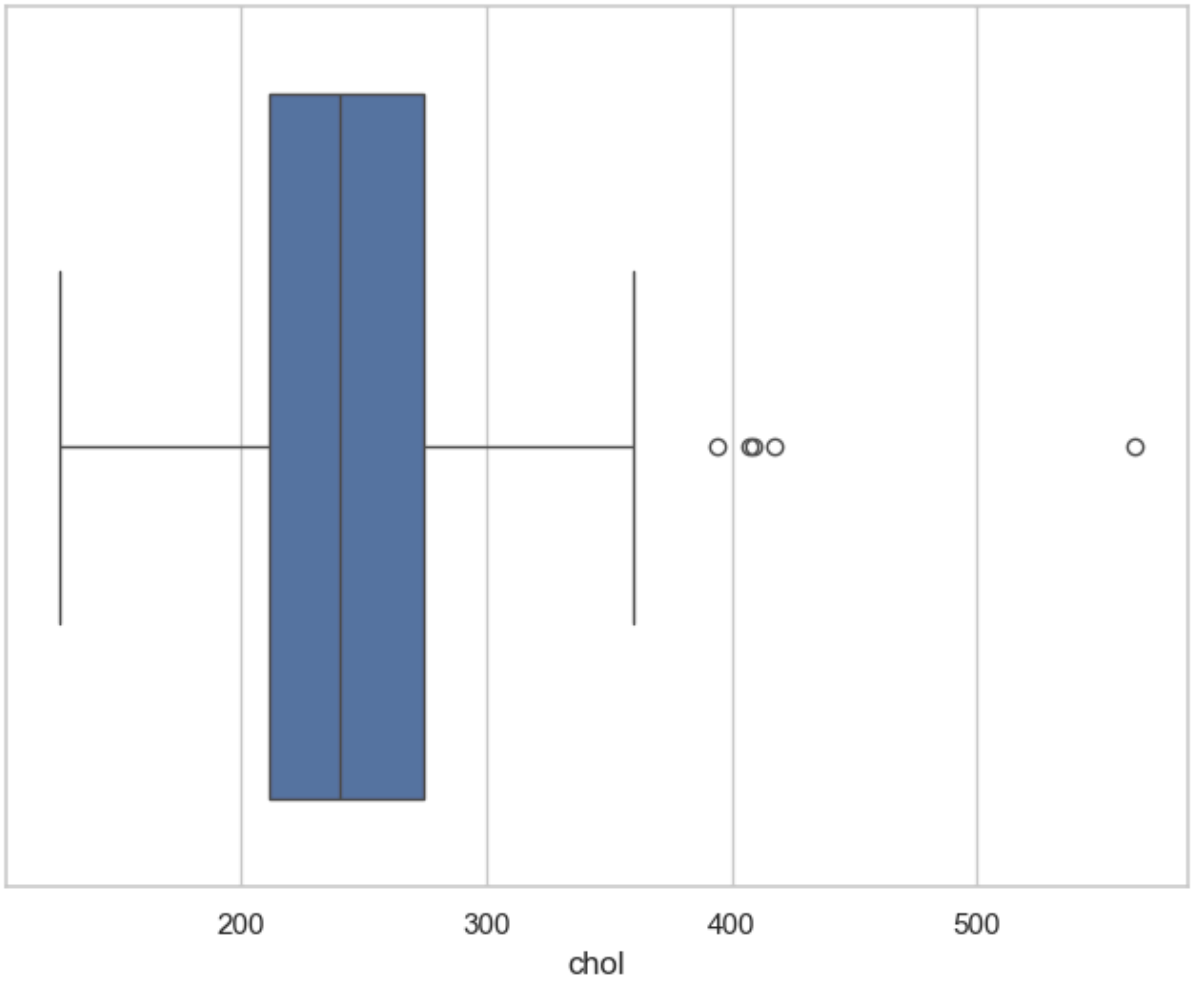
chol variable

In [187…
```
df['chol'].describe()
```

Out[187…
```
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 [190…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["chol"])
plt.show()
```



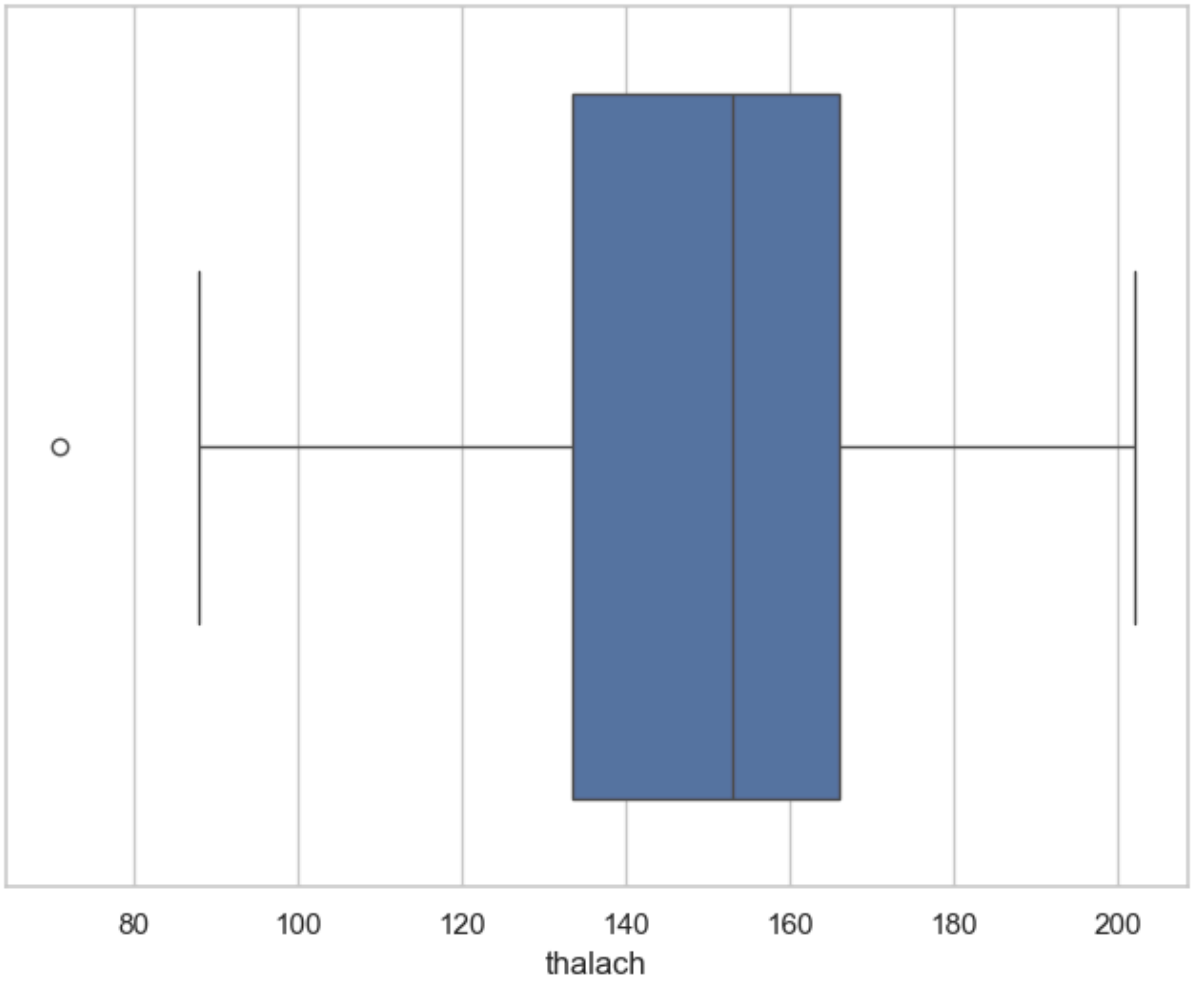
thalach variable

In [193…
```
df['thalach'].describe()
```

Out[193…
```
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
```

boxplot of thalach variable

In [196…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["thalach"])
plt.show()
```



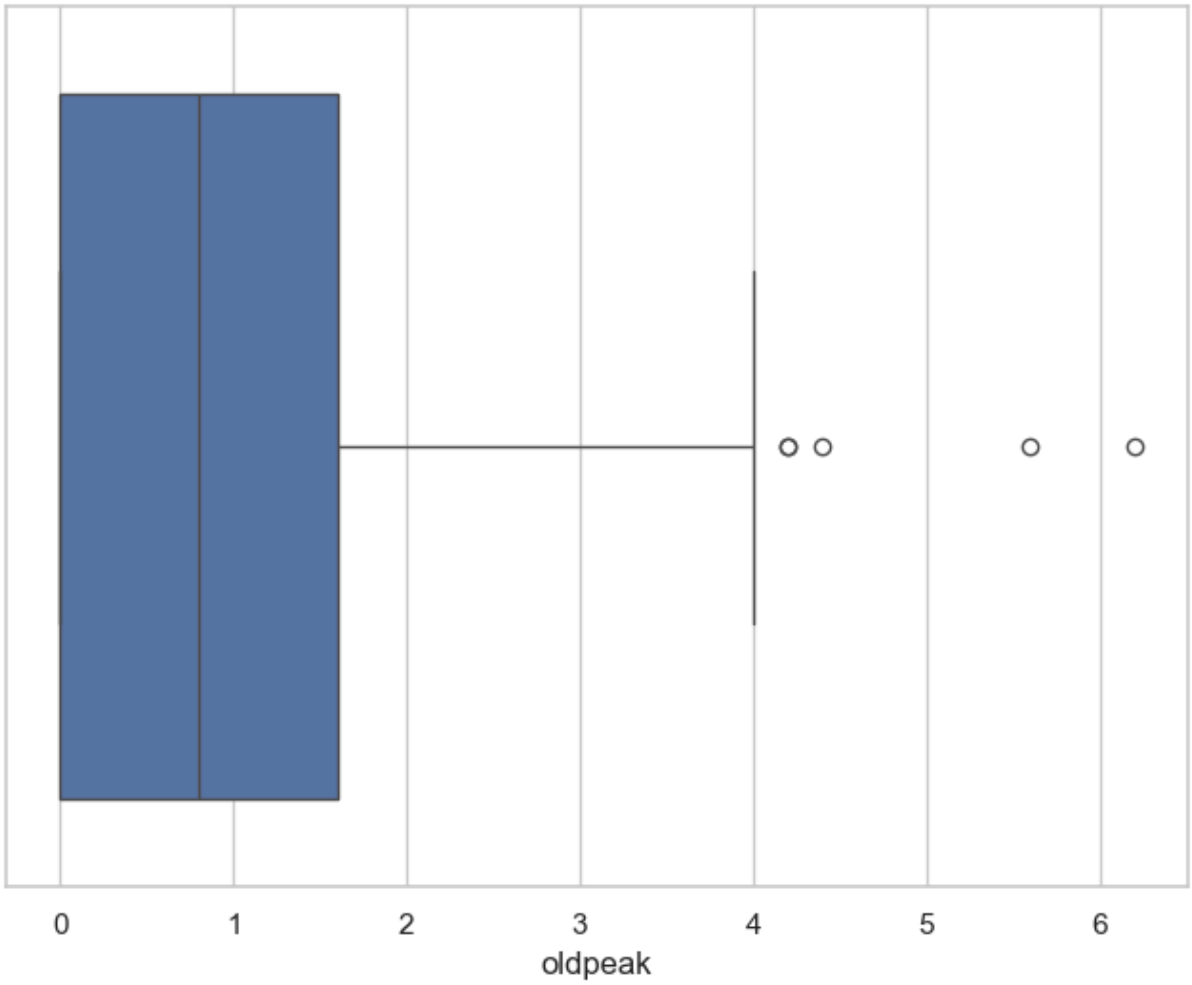
Old peak variable

In [199…
```
df['oldpeak'].describe()
```

Out[199…
```
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
```

boxplot of old peak varaiable

In [202…
```
f, ax = plt.subplots(figsize=(8, 6))
sns.boxplot(x=df["oldpeak"])
plt.show()
```



13.CONCLUSION

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.

So, friends, our EDA journey has come to an end.

In this kernel, we have explored the heart disease dataset. In this kernel, we have implemented many of the strategies presented in the book **Think Stats - Exploratory Data Analysis in Python by Allen B Downey** . The feature variable of interest is `target`

variable. We have analyzed it alone and check its interaction with other variables. We have also discussed how to detect missing data and outliers.

I hope you like this kernel on EDA journey.

Thanks

14. References

Back to Table of Contents

The following references are used to create this kernel

- Think Stats - Exploratory Data Analysis in Python by Allen B Downey
- Seaborn API reference
- My other kernel

In [ ]: