Heart disease or Cardiovascular disease (CVD)

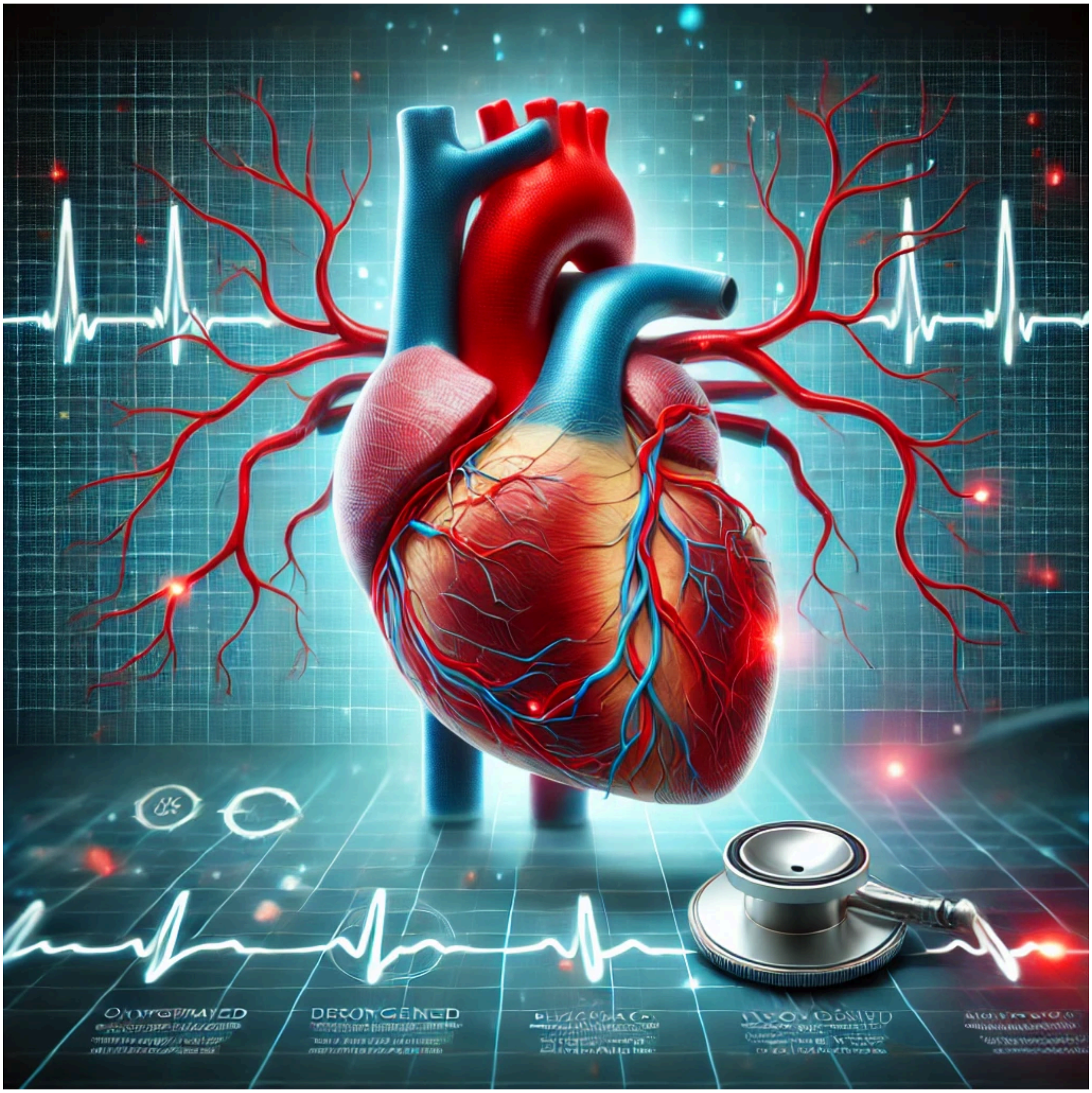
* EDA

* Extensive Analysis + Visualization with Python

In [60]:

```
from IPython.display import Image

# Display the image
img = Image(r"C:\Users\admin\Downloads\Heart.png")  # Replace with your actual imag
display(img)
```



In [3]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, csv file I/O (e.g. pd.read_csv)
```

- We can see that the input folder contains one input file named heart.csv.

In [4]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st
%matplotlib inline

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

In [5]:

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

I have imported the libraries. The next step is to import the datasets.

import dataset

- I will import the dataset with the usual pandas read_csv() function which is used to import CSV (Comma Separated Value) files.

In [7]:

```
df = pd.read_csv(r"C:\DataScience Classnotes\9th- september - seaborn - Seaborn, Ed
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

Exploratory Data Analysis

The scene has been set up. Now let the actual fun begin.

- Check shape of the dataset
-

It is a good idea to first check the shape of the dataset.

In [10]:

```
# print The shape
print('The Shape of the dataset',df.shape)
```

```
The Shape of the dataset (303, 14)
```

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

# Preview the dataset

In [12]:

```
# preview the dataset
df.head()
```

Out[12]:

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

#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
```

#Dataset description

- The dataset contains several columns which are as follows-
- age : age in years
- sex : 1 male & 0 female
- cp : chest pain type
- trestbps : resting blood pressure (in mm Hg on admission to the hospital)
- chol : serum cholestrol in mg/dl
- fbs : (fast blood sugar > 120 mg/dl)(1 =true; 0 =false)
- restecg : resting electrocardiographic results
- thalach : maximum heart rate achieved
- exang : exercise induced angina (1 = yes; 0 = no)
- oldpeak : ST depression induced by exercise relative to rest
- slope : the slope of the peak exercise ST segment
- ca : number of major vessels (0-3) colored by flourosopy
- thal : 3 = normal; 6 = fixed defect; 7 = reversable defect
- target : 1 or 0

#Checks the datatypes of columns

In [16]:

```
df.dtypes
```

```
Out[16]: 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
```

#Important points about dataset

- `sex` is a character variable. Its data type should be object. But it is encoded as (1 = male; 0 = female). So, its data type is given as `int64`.
- Same is the case with several other variables - `fbs`, exang and `target`.
- `fbs` (fasting blood sugar) should be a character variable as it contains only 0 and 1 as values (1 = true; 0 = false). As it contains only 0 and 1 as values, so its data type is given as int64.
- `exang` (exercise induced angina) should also be a character variable as it contains only 0 and 1 as values (1 = yes; 0 = no). It also contains only 0 and 1 as values, so its data type is given as int64.
- `target` should also be a character variable. But, it also contains 0 and 1 as values. So, its data type is given as int64.

#Statistical properties of dataset

```
In [19]: # statistical properties of dataset
         df.describe()
```

Out[19]:

	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

In [20]:

```
# view columns name
df.columns
```

Out[20]:

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

#Univeriate Analysis

Analysis of target feature veriables

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

#Check the number of unique values in target variable

In [24]:

```
df['target'].nunique()
```

Out[24]:

```
2
```

We can see that there are 2 unique values in the target variable

In [26]:

```
#view the unique values in target veriable
df['target'].unique()
```

Out[26]:

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

Comment

- So, the unique values are 1 and 0. (1 stands for presence of heart disease and 0 for absence of heart disease).

#Frequency distribution of `target` variable

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

Out[29]:
```
target
1    165
0    138
Name: count, dtype: int64
```

Comment

- `1` stands for presence of heart disease. So, there are 165 patients suffering from heart disease.

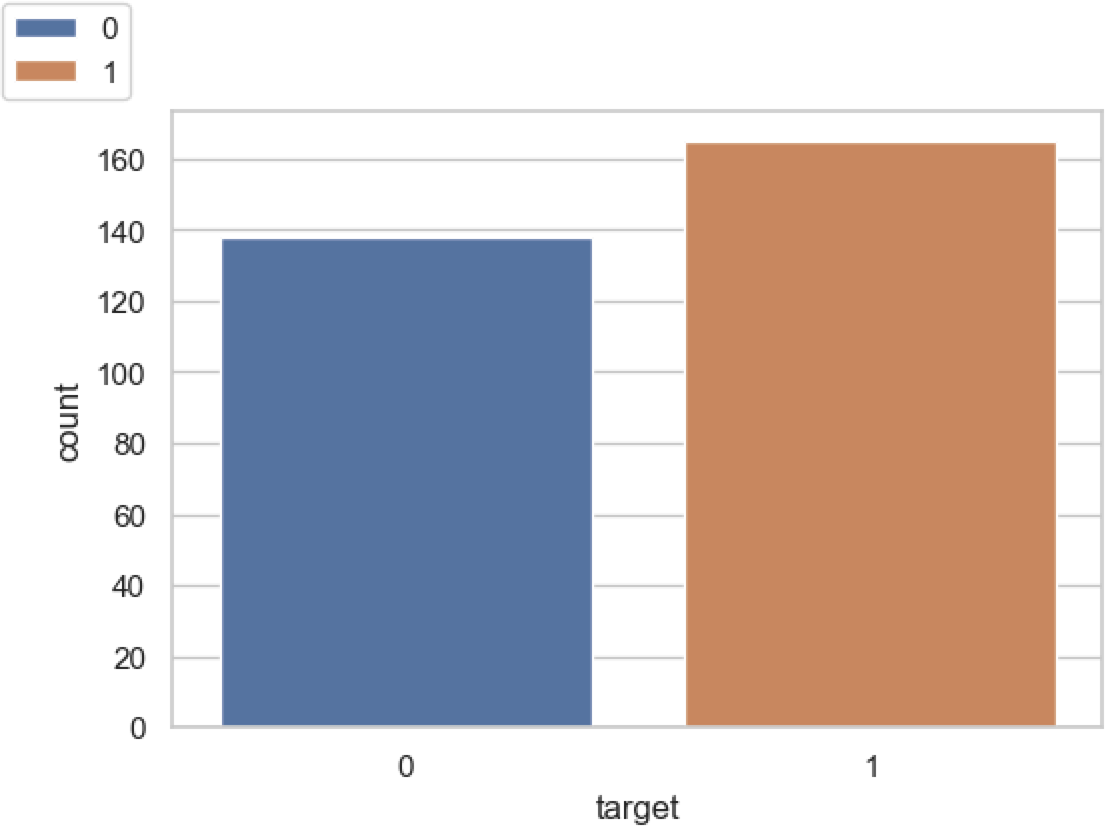
- Similarly, `0` stands for absence of heart disease. So, there are 138 patients who do not have any heart disease.

- We can visualize this information below.

Visualize frequency distribution of `target` variable

In [32]:
```
f, ax = plt.subplots(figsize=(6,4))

ax = sns.countplot(x = 'target', data=df ,hue='target')
plt.legend(loc = 'upper left',bbox_to_anchor=(-0.2, 1.2) )
plt.show()
```



Interpretation

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

Frequency distribution of `target` variable wrt `sex`

In [35]:
```
df.groupby('sex')['target'].value_counts()
```

Out[35]:
```
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.

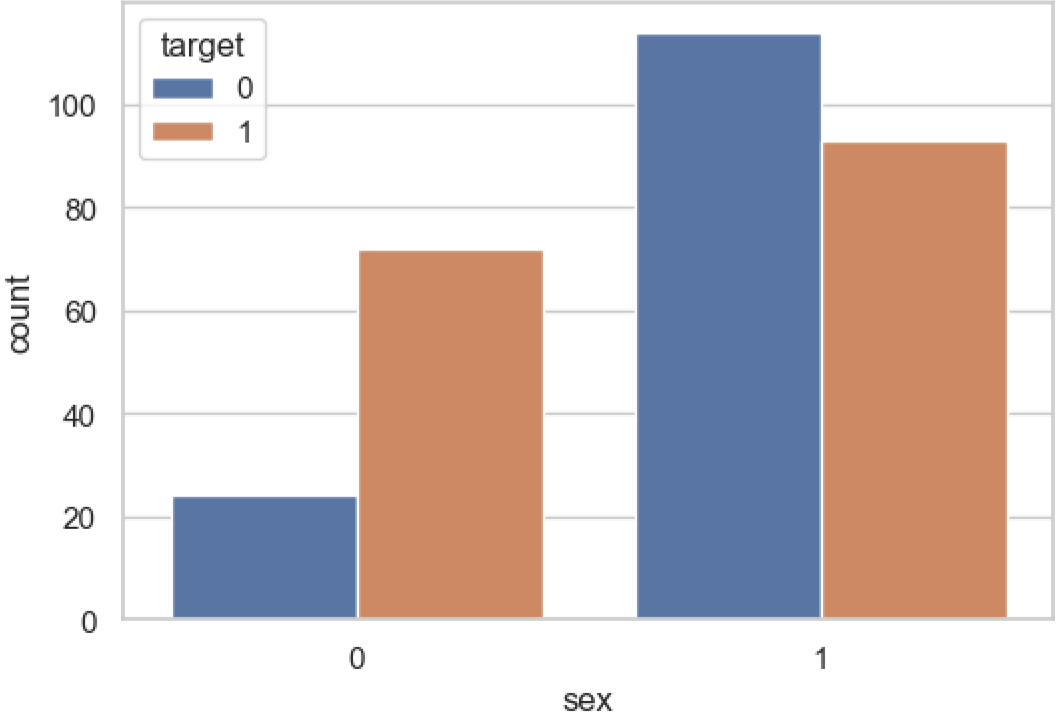
- Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

- We can visualize this information below.

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

In [38]:
```
#A countplot in Seaborn is a bar plot that shows the number of occurrences of each
#It helps visualize the distribution of categorical data.
f, ax=plt.subplots(figsize=(6,4))

ax = sns.countplot(x='sex', hue='target', data=df)
plt.show()
```



Interpretation

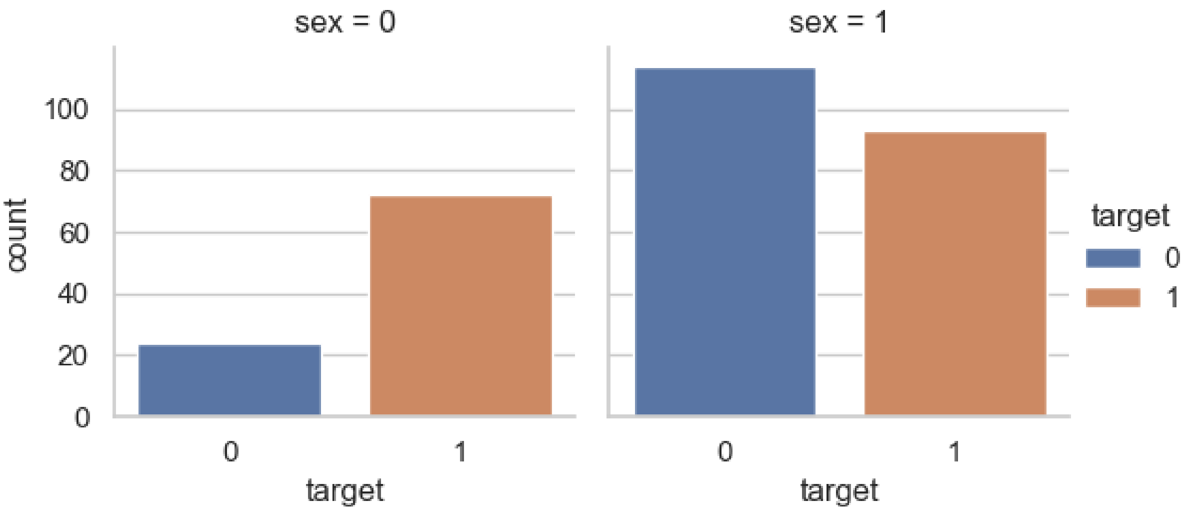
- We can see that the values of `target` variable are plotted wrt `sex` : (1 = male; 0 = female).

- `target` variable also contains two integer values 1 and 0 : (1 = Presence of heart disease; 0 = Absence of heart disease)

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

Alternatively, we can visualize the same information as follows :

```
In [41]: ax = sns.catplot(x='target',col='sex', data = df, kind='count',height=3, aspect=1,h
         # col='sex': Creates separate subplots for each unique value in the 'sex' column.
         # kind='count': Specifies that it's a count plot.
```

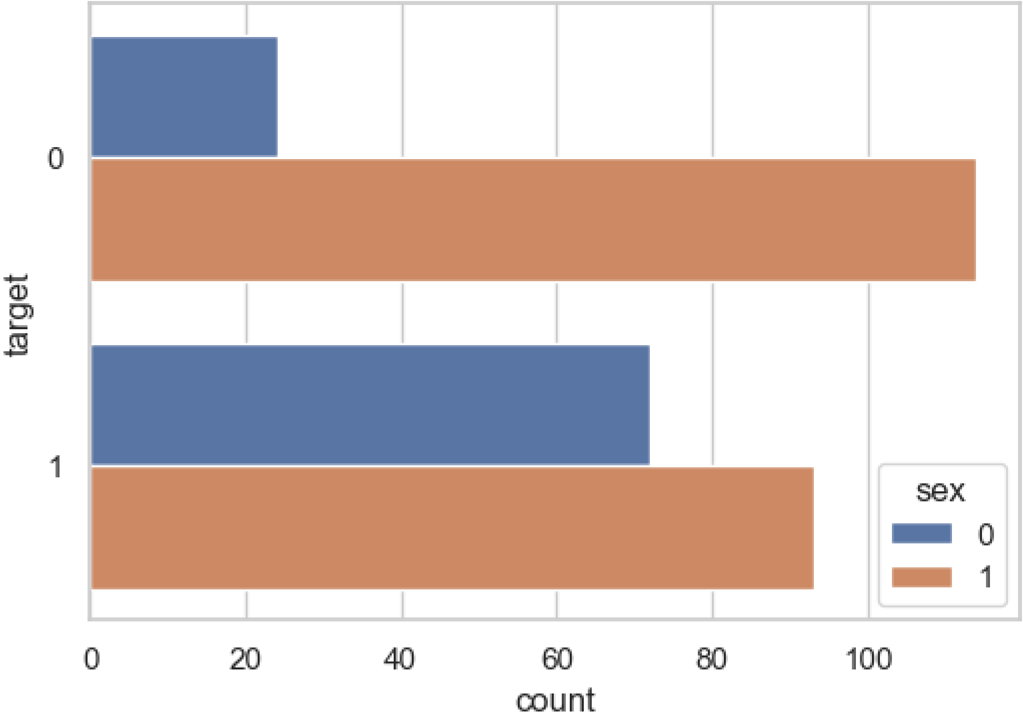


Comment

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

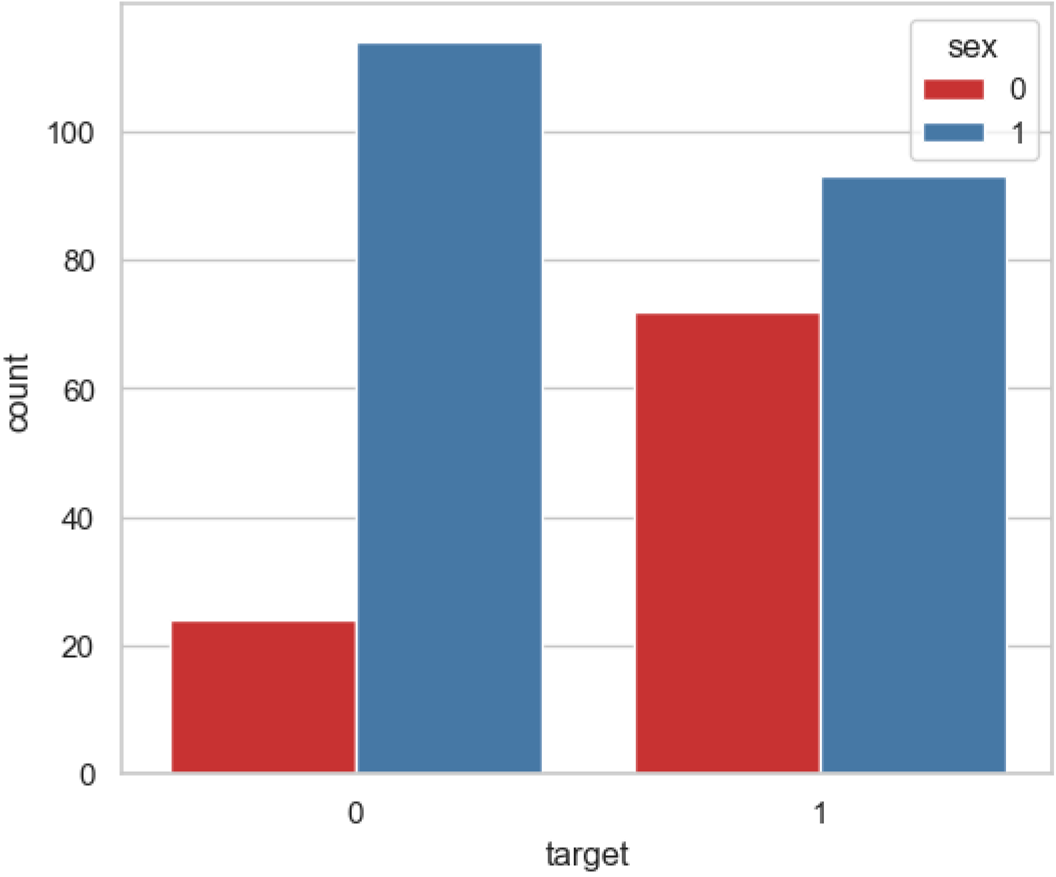
We can plot the bars horizontally as follows :

```
In [44]: f, ax= plt.subplots(figsize=(6,4))
         ax = sns.countplot(y='target', hue='sex',data=df)
         plt.show()
```



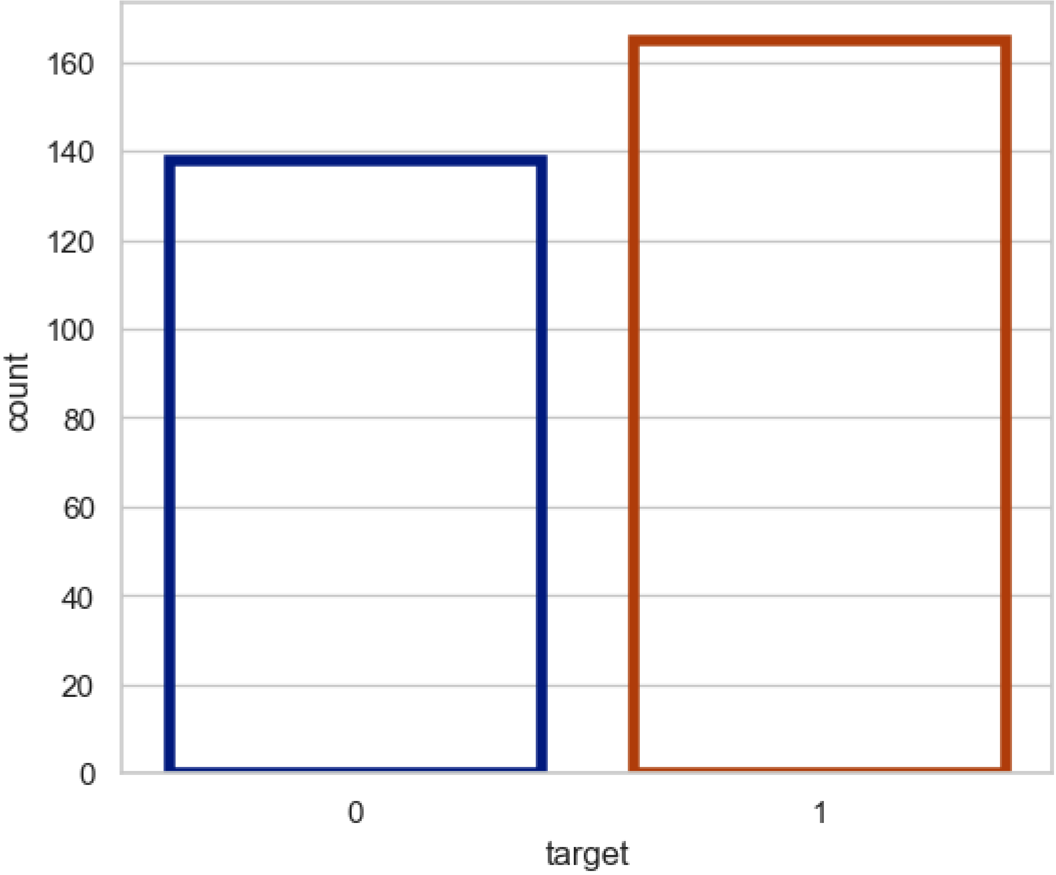
We can use a different color palette as follows :

```
In [46]: f,ax = plt.subplots(figsize=(6,5))
         ax = sns.countplot(x='target', hue='sex', data=df, palette="Set1")
         plt.show()
```



We can use `plt.bar` keyword arguments for a different look :

In [48]:
```
f, ax= plt.subplots(figsize=(6,5))
ax = sns.countplot(x='target', data = df, facecolor=(0, 0, 0, 0), linewidth=4, edge
plt.show()
```



Comment

- I have visualize the `target` values distribution wrt `sex`.
- We can follow the same principles and visualize the `target` values distribution wrt `fbs (fasting blood sugar)` and `exang (exercise induced angina)`.

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

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

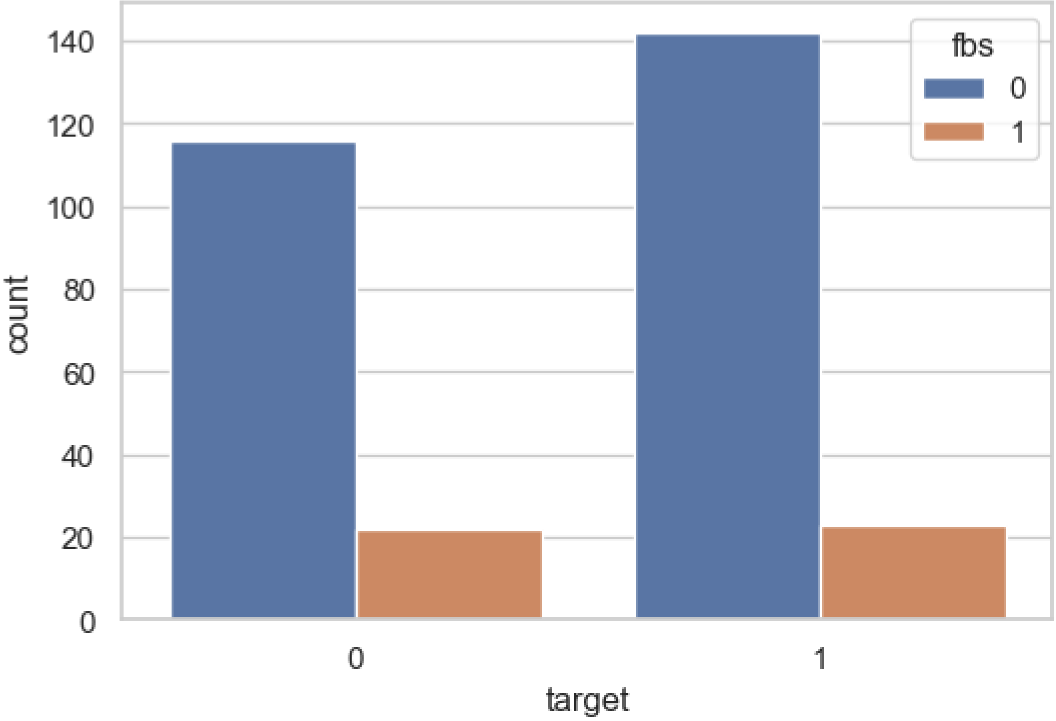
In [51]:
```
df.groupby('sex')['target'].value_counts()
```

Out[51]:

```
sex  target
0    1          72
     0          24
1    0         114
     1          93
Name: count, dtype: int64
```

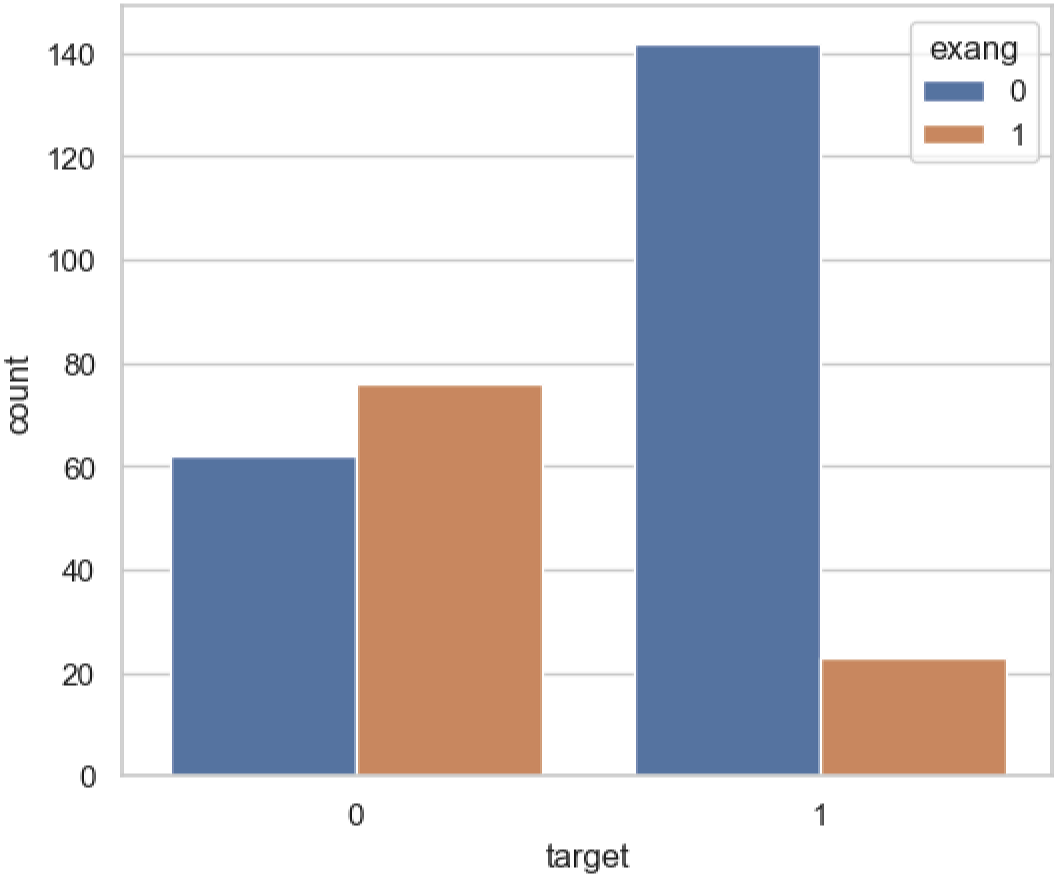
In [52]:

```
f,ax =plt.subplots(figsize=(6,4))
ax = sns.countplot(x='target',hue='fbs',data=df)
plt.show()
```



In [53]:

```
f,ax = plt.subplots(figsize=(6,5))
ax = sns.countplot(x='target',hue='exang',data=df)
plt.show()
```



```
In [ ]:
```

#Findings of Univariate Analysis

Findings of univariate analysis are as follows:-

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

- Similarly, out of 207 males - 93 have heart disease and 114 do not have heart disease.

#Bivariate Analysis

Estimate correlation coefficients

Our dataset is very small. So, I will compute the standard correlation coefficient (also called Pearson's r) between every pair of attributes. I will compute it using the `df.corr()` method as follows:-

```
In [57]: correlation = df.corr()
```

The target variable is `target`. So, we should check how each attribute correlates with the `target` variable. We can do it as follows:-

```
In [59]: correlation['target'].sort_values(ascending=False)
```

```
Out[59]: 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`.