Project B: Heart Disease Analysis Using Univariate/Bivariate Technique

In [1]:
```
import pandas as pd
import numpy as np
filepath = 'C:/Users/User/Downloads/heart.cvs'
```

In [2]:
```
df = pd.read_csv('C:/Users/User/Downloads/heart.csv')
```

In [3]:
```
df
```

Out[3]:

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

303 rows × 14 columns

In [4]:
```
df.shape
```

Out[4]:
```
(303, 14)
```

Data set has 14 columns and 303 rows

In [5]:
```
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 [6]:
```
df.isnull().sum()
```

Out[6]:
```
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
```

Inference: The inference we can derive from the above output is:

Out of 14 features, we have 13 int types and only one with the float data types.

Fortunately, this dataset doesn't hold any missing values.

In [7]:
```
df.describe().T
```

Out[7]:

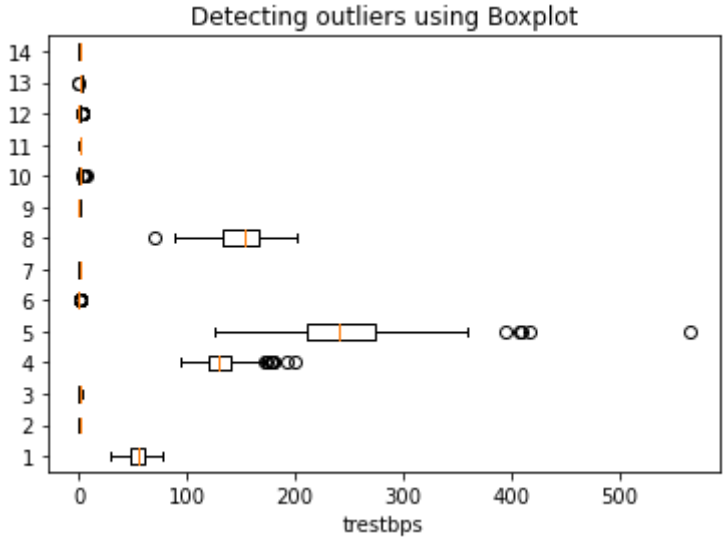
	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
sex	303.0	0.683168	0.466011	0.0	0.0	1.0	1.0	1.0
cp	303.0	0.966997	1.032052	0.0	0.0	1.0	2.0	3.0
trestbps	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
fbs	303.0	0.148515	0.356198	0.0	0.0	0.0	0.0	1.0
restecg	303.0	0.528053	0.525860	0.0	0.0	1.0	1.0	2.0
thalach	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
exang	303.0	0.326733	0.469794	0.0	0.0	0.0	1.0	1.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2
slope	303.0	1.399340	0.616226	0.0	1.0	1.0	2.0	2.0
ca	303.0	0.729373	1.022606	0.0	0.0	0.0	1.0	4.0
thal	303.0	2.313531	0.612277	0.0	2.0	2.0	3.0	3.0
target	303.0	0.544554	0.498835	0.0	0.0	1.0	1.0	1.0

To Check for Outlier

In [8]:
```
#create a box plot

import matplotlib.pyplot as plt
plt.boxplot(df, vert=False)
plt.title("Detecting outliers using Boxplot")
plt.xlabel('trestbps')
```

Out[8]:
```
Text(0.5, 0, 'trestbps')
```



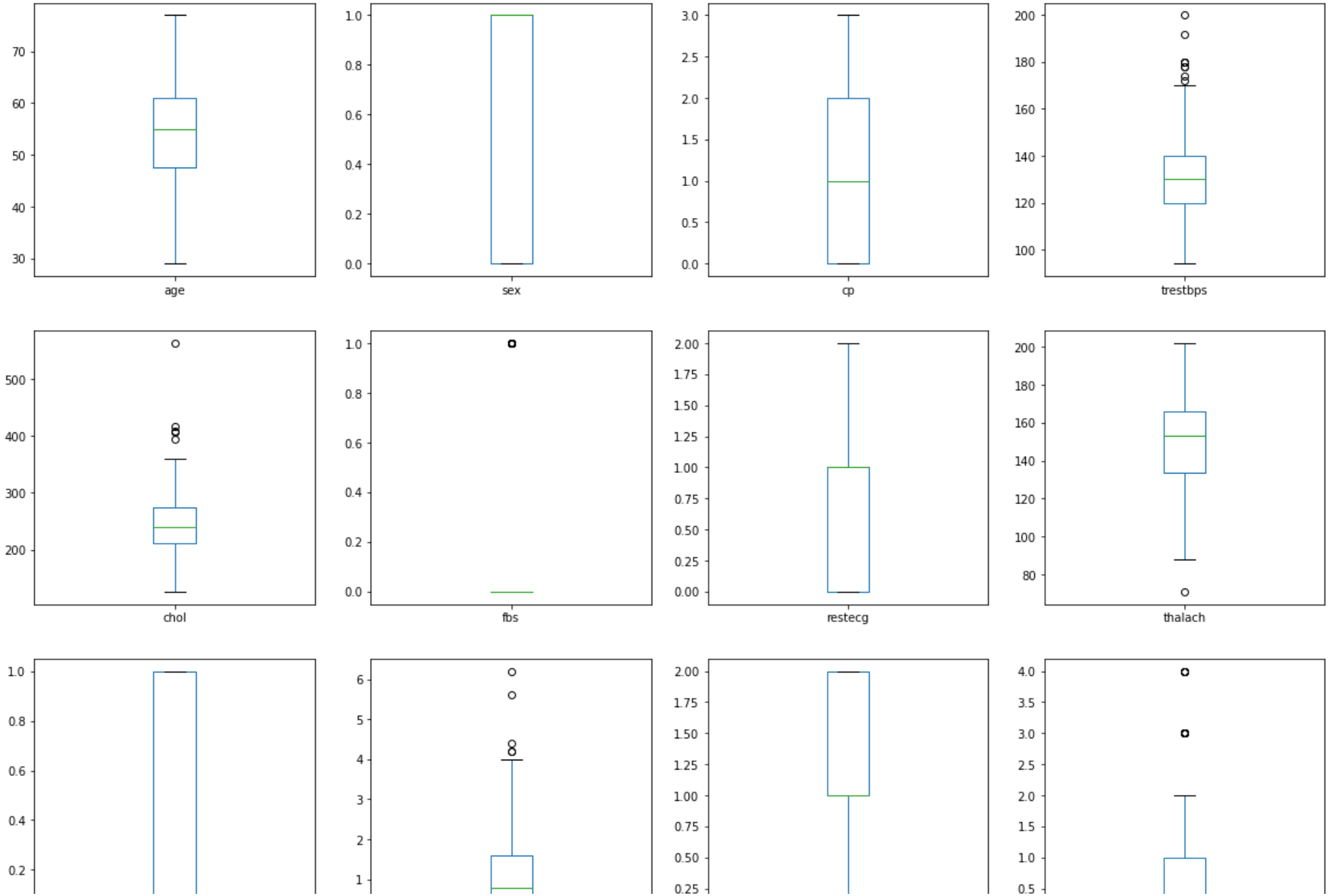
In [9]:
```
df.describe()[['trestbps','chol','thalach','thalach','oldpeak','ca','thal']]
```

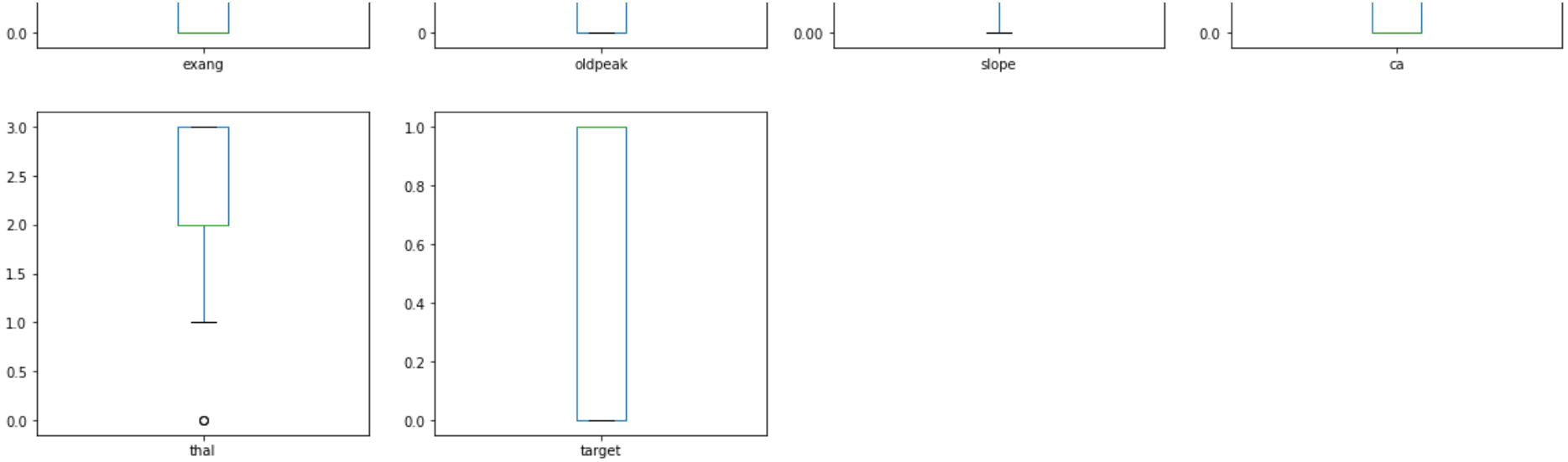
Out[9]:

	trestbps	chol	thalach	thalach	oldpeak	ca	thal
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	131.623762	246.264026	149.646865	149.646865	1.039604	0.729373	2.313531
std	17.538143	51.830751	22.905161	22.905161	1.161075	1.022606	0.612277
min	94.000000	126.000000	71.000000	71.000000	0.000000	0.000000	0.000000
25%	120.000000	211.000000	133.500000	133.500000	0.000000	0.000000	2.000000
50%	130.000000	240.000000	153.000000	153.000000	0.800000	0.000000	2.000000
75%	140.000000	274.500000	166.000000	166.000000	1.600000	1.000000	3.000000
max	200.000000	564.000000	202.000000	202.000000	6.200000	4.000000	3.000000

In [10]:
```
# Box and Whisker Plots
from matplotlib import pyplot
from pandas import read_csv
```

```
data = df
data.plot(kind='box', subplots=True, layout=(4,4), sharex=False, sharey=False)
plt.gcf().set_size_inches(20,20)
pyplot.show()
```





The following are Features with Outliers: trestbps, chol, thalach, thalach, oldpeak, ca, thal. contains Outliers

In [11]:
```
df
```

Out[11]:

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

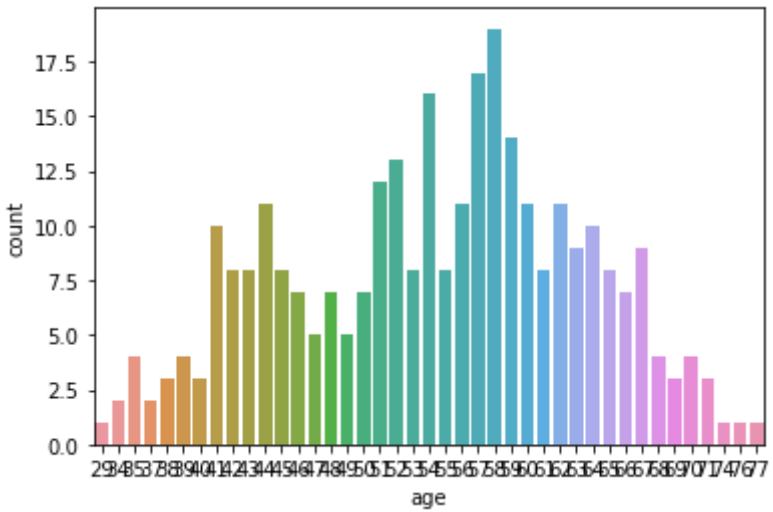
303 rows × 14 columns

Univariate Analysis

In [12]:
```
import seaborn as sns
import matplotlib.pyplot as plt
# plot count plot for the age column
sns.countplot(df.age)
```

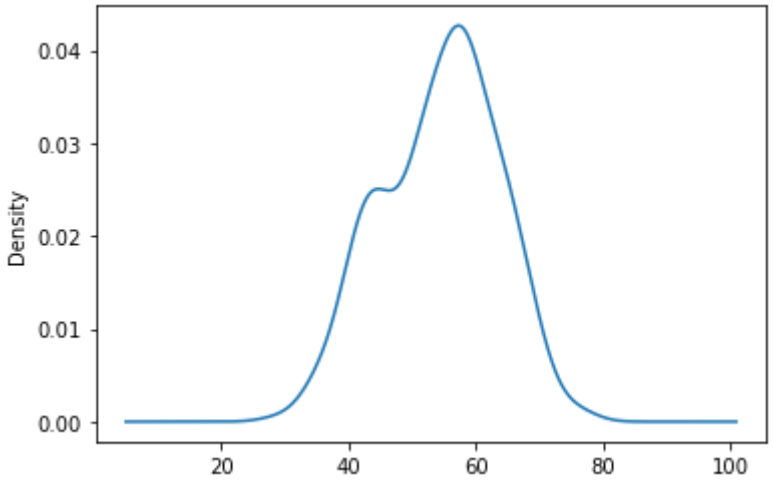
```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

Out[12]: <AxesSubplot:xlabel='age', ylabel='count'>



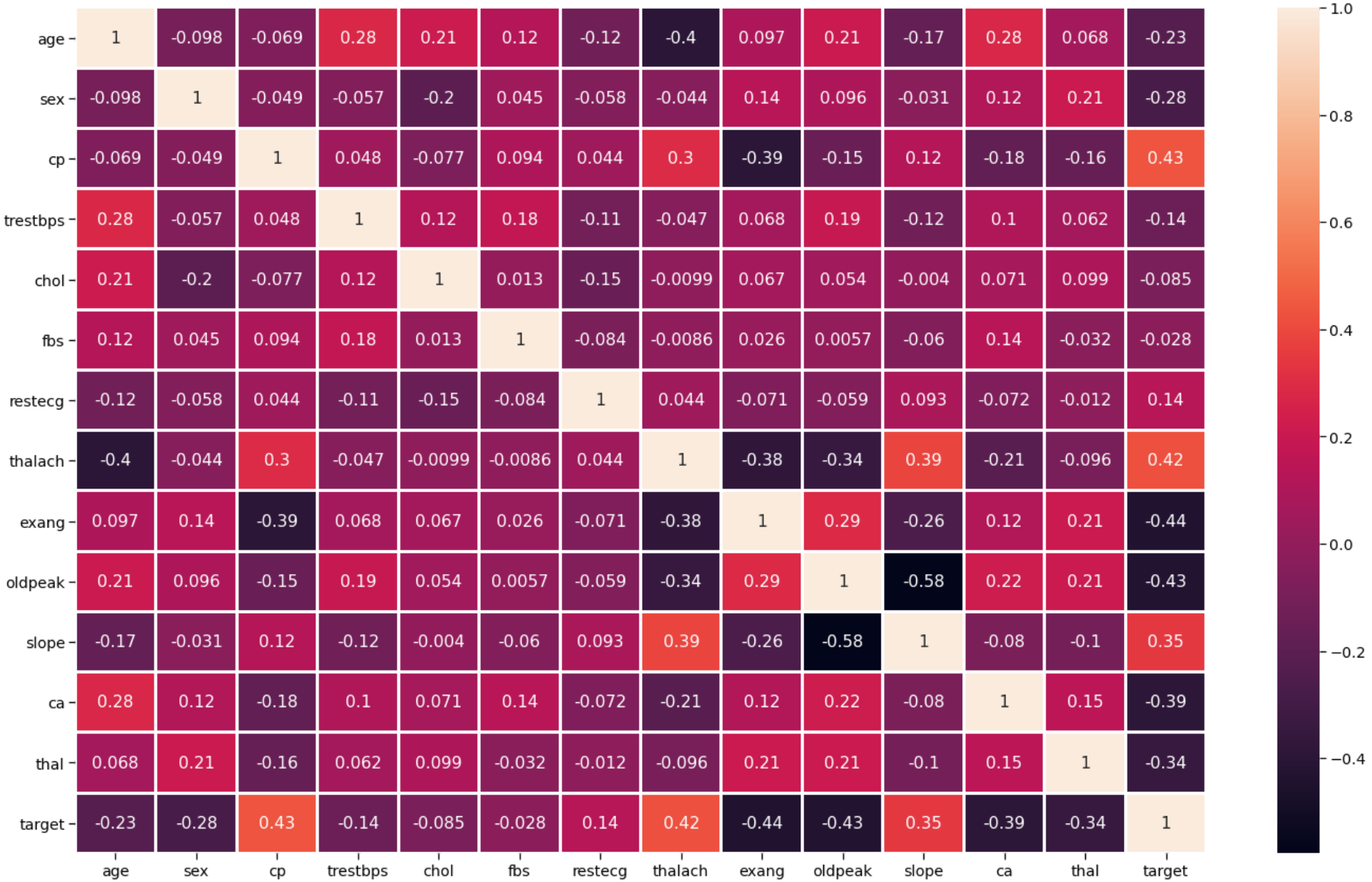
In [13]:
```
df.age.plot(kind='density')
```

Out[13]: <AxesSubplot:ylabel='Density'>



In [14]:

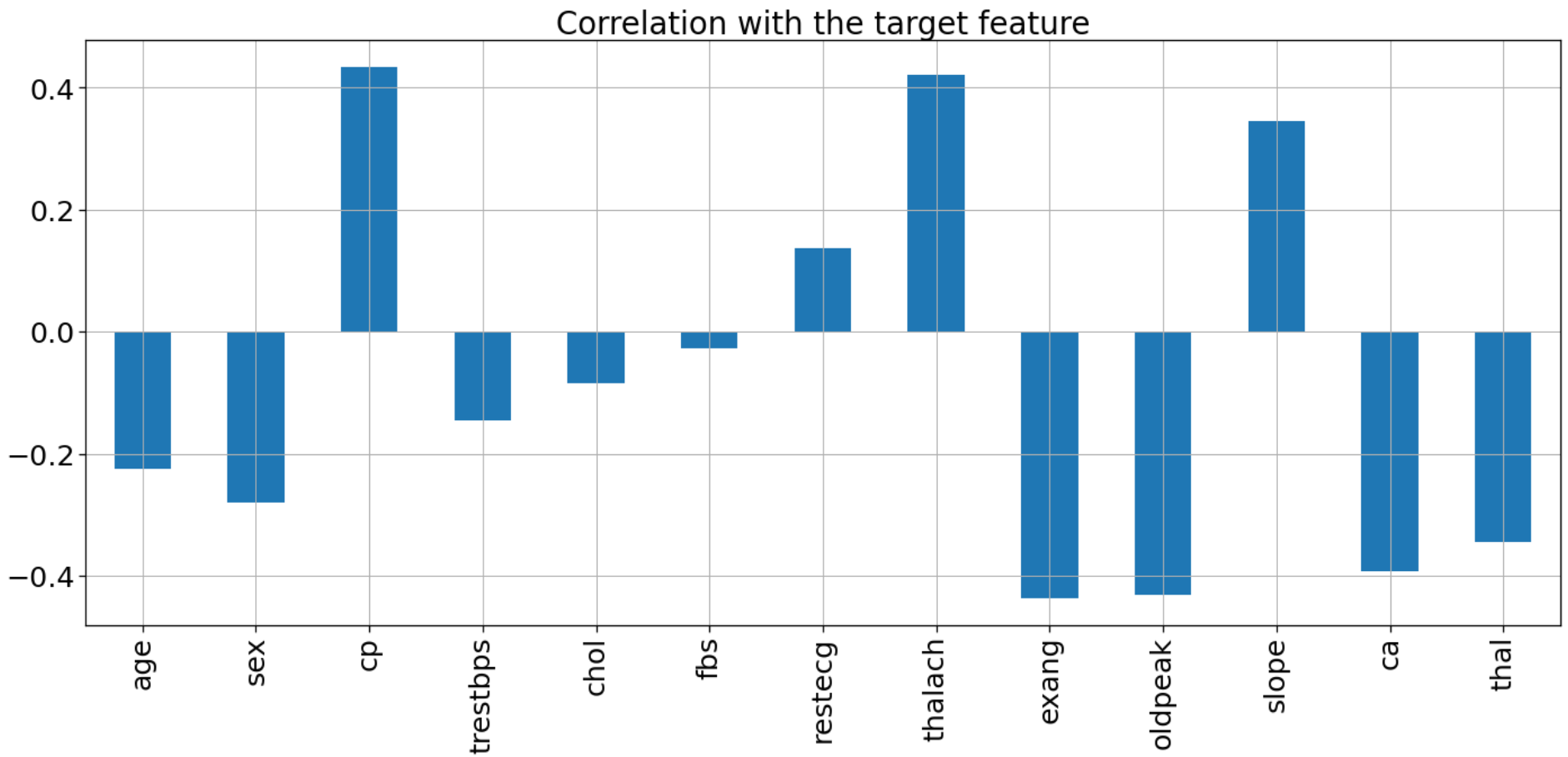
```
plt.figure(figsize=(20,12))
sns.set_context('notebook',font_scale = 1.3)
sns.heatmap(data.corr(),annot=True,linewidth =2)
plt.tight_layout()
```



By far we have checked the correlation between the features but it is also a good practice to check the correlation of the target variable.

In [15]:
```
sns.set_context('notebook',font_scale = 2.3)
data.drop('target', axis=1).corrwith(data.target).plot(kind='bar', grid=True, figsize=(20, 10),
```

```
                                                    title="Correlation with the target feature")
plt.tight_layout()
```



Inference: Insights from the above graph are:

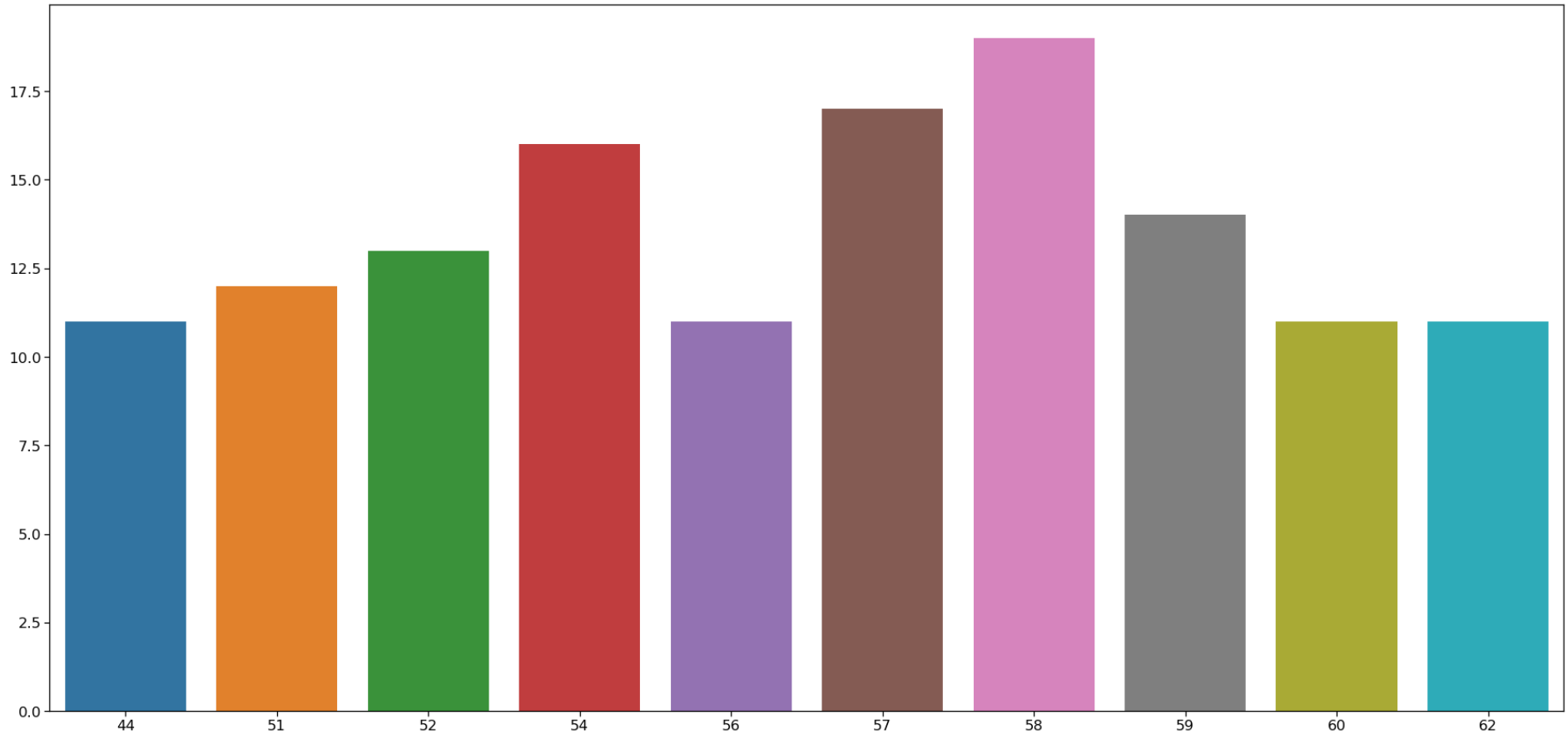
Four feature(“cp”, “restecg”, “thalach”, “slope”) are positively correlated with the target feature. Other features are negatively correlated with the target feature.

It is important to make the analysis of the individual features which comprises both univariate and bivariate analysis.

AGE ANALYSIS

Here i will be checking the 10 ages and their counts.

In [16]:
```
plt.figure(figsize=(25,12))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=data.age.value_counts()[:10].index,y=data.age.value_counts()[:10].values)
plt.tight_layout()
```



Inference: Here we can see that the 58 age column has the highest frequency.

In [17]:
```
minAge=min(data.age)
maxAge=max(data.age)
meanAge=data.age.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)
```

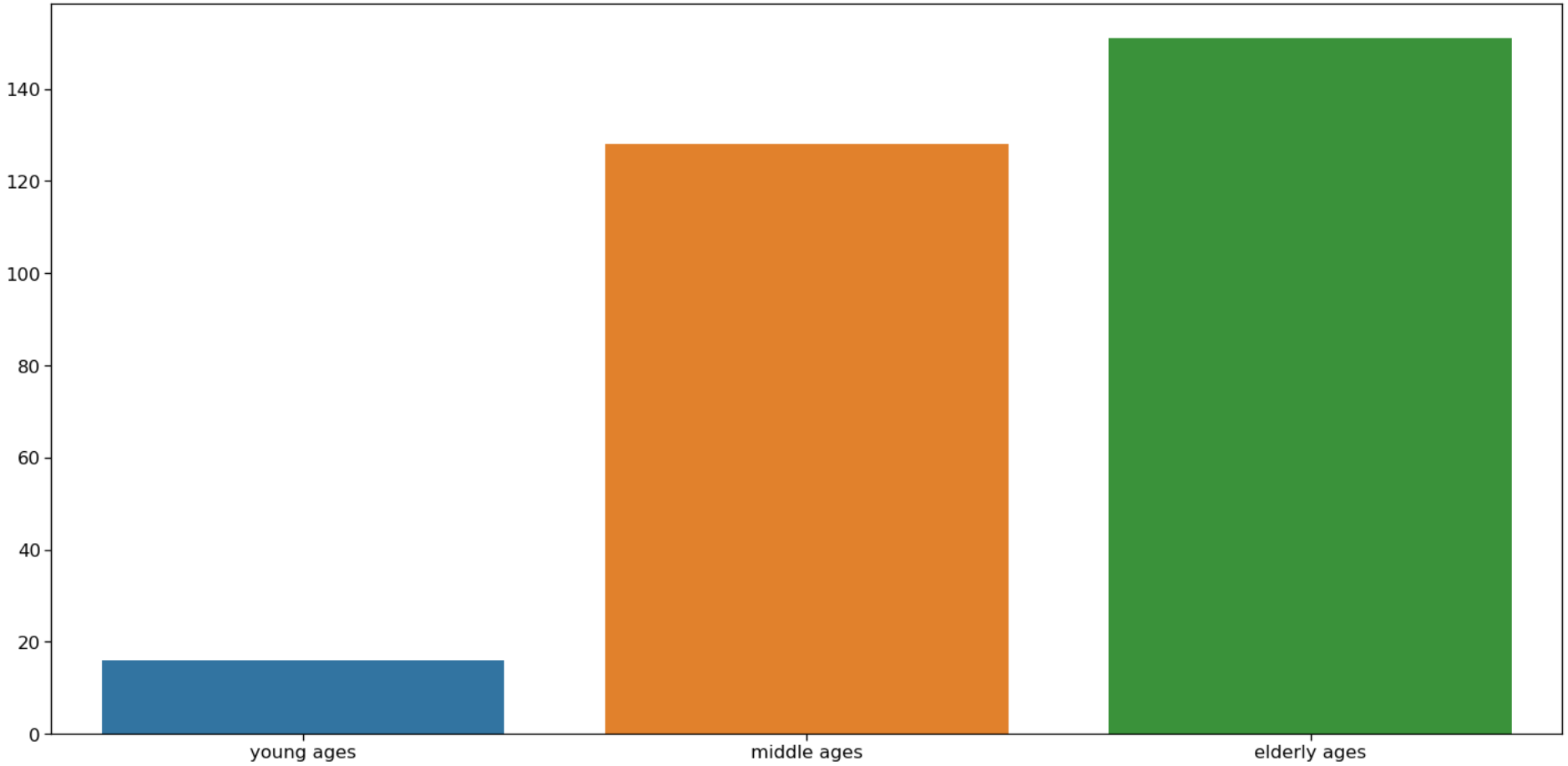
```
Min Age : 29
Max Age : 77
Mean Age : 54.366336633663366
```

We should divide the Age feature into three parts – "Young", "Middle" and "Elder"

In [18]:

```
Young = data[(data.age>=29)&(data.age<40)]
Middle = data[(data.age>=40)&(data.age<55)]
Elder = data[(data.age>55)]

plt.figure(figsize=(20,10))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),len(Middle),len(Elder)])
plt.tight_layout()
```

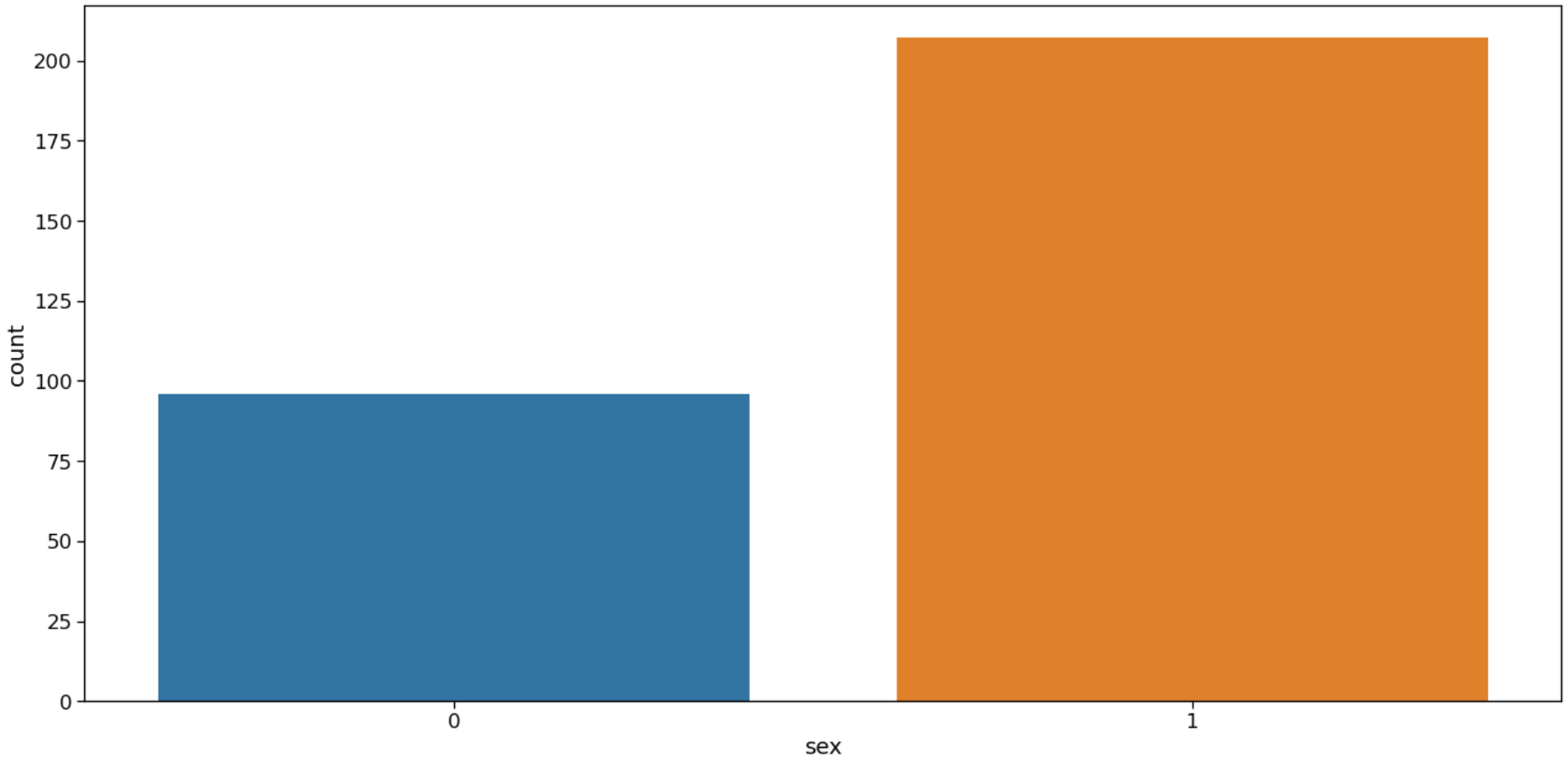


Inference: Here we can see that elder people are the most affected by heart disease and young ones are the least affected.

Sex Feature Analysis

In [19]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['sex'])
plt.tight_layout()
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

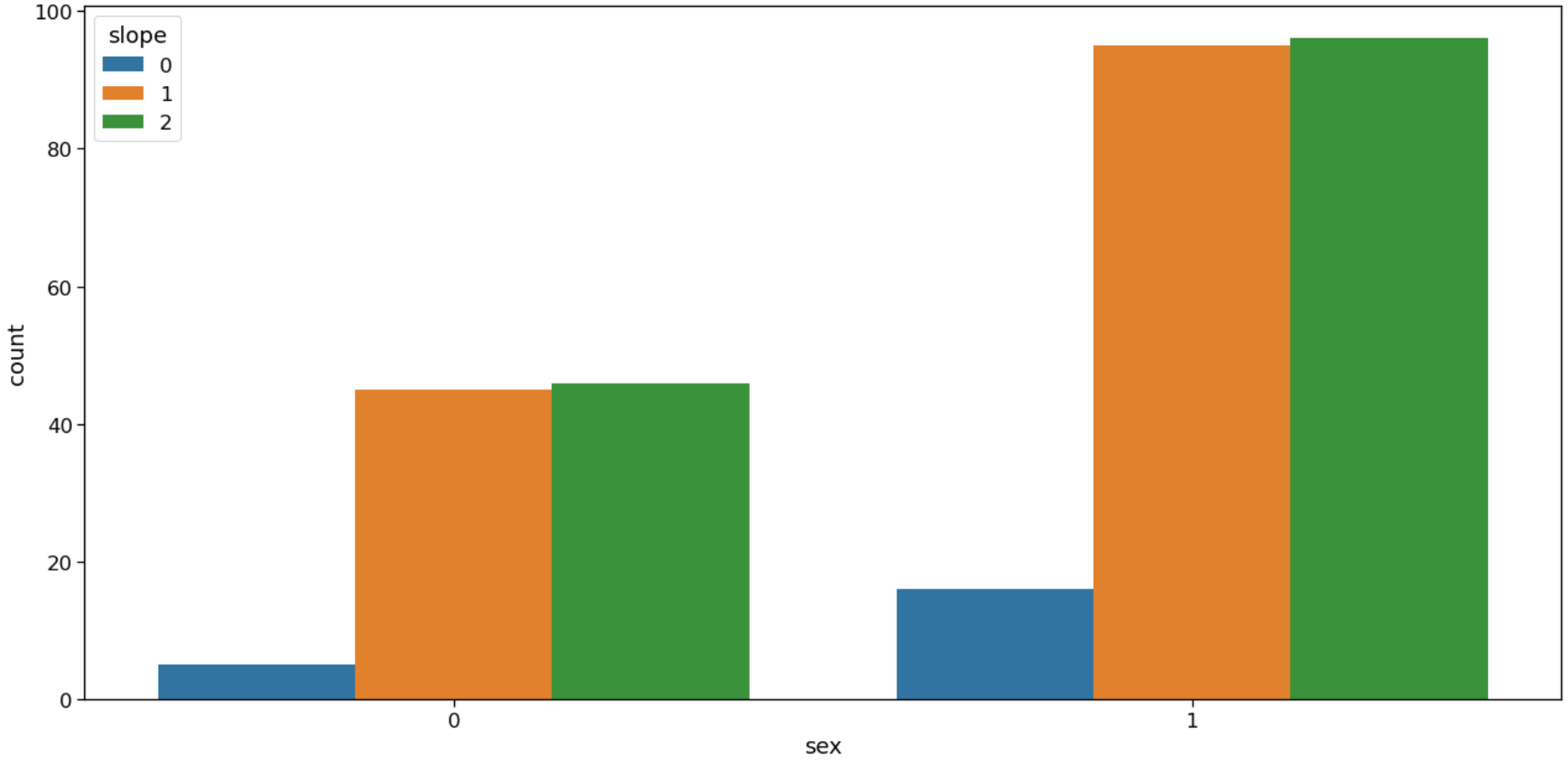


Inference: Here it is clearly visible that, Ratio of Male to Female is approx 2:1.

To plot the relation between sex and slope.

In [20]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['sex'],hue=data["slope"])
plt.tight_layout()
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

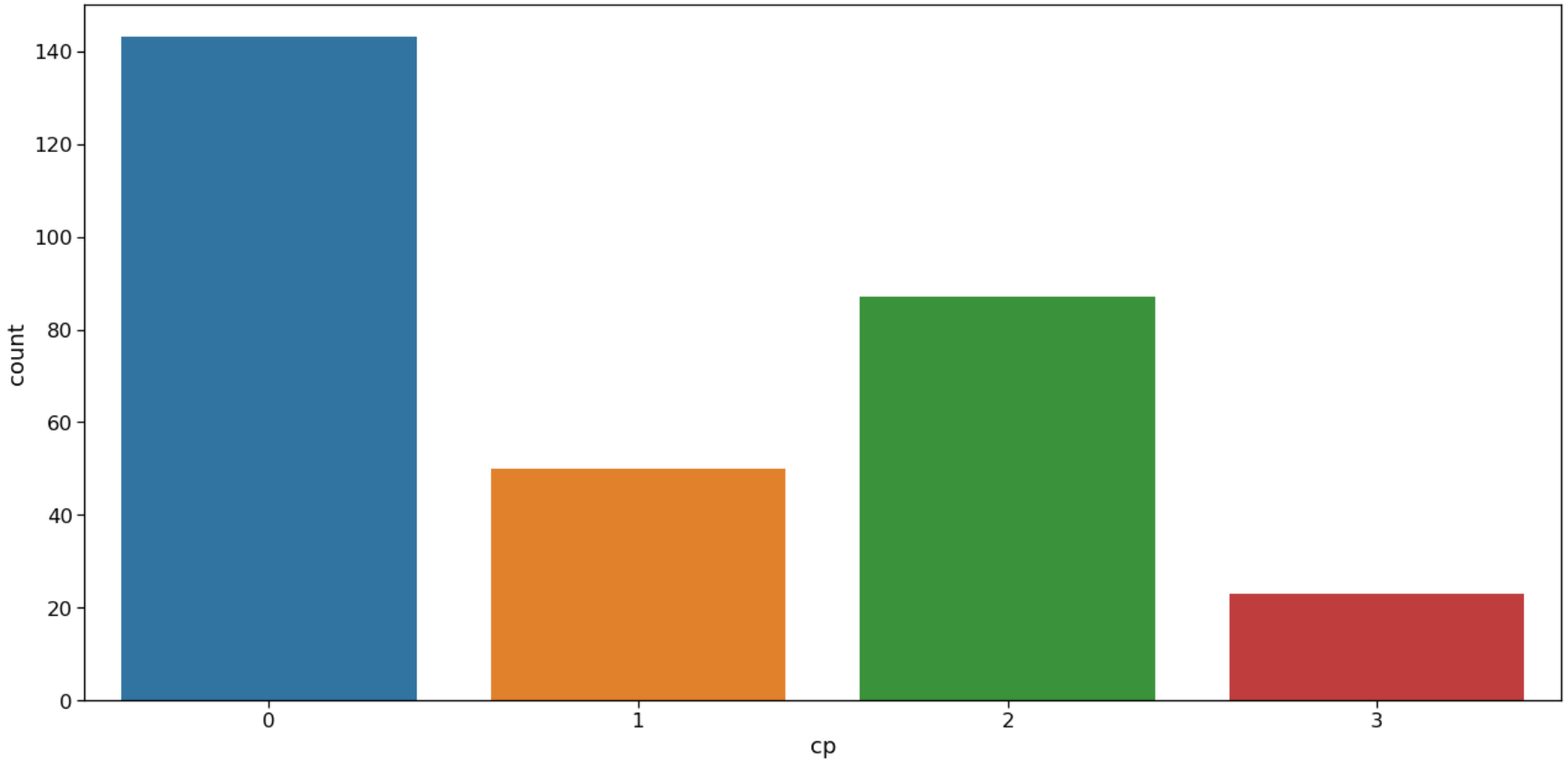


Inference: Here it is clearly visible that the slope value is higher in the case of males(1).

Chest Pain Type("cp") Analysis

In [21]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['cp'])
plt.tight_layout()
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```



Inference: As seen, there are 4 types of chest pain:

status at least;

condition slightly distressed;

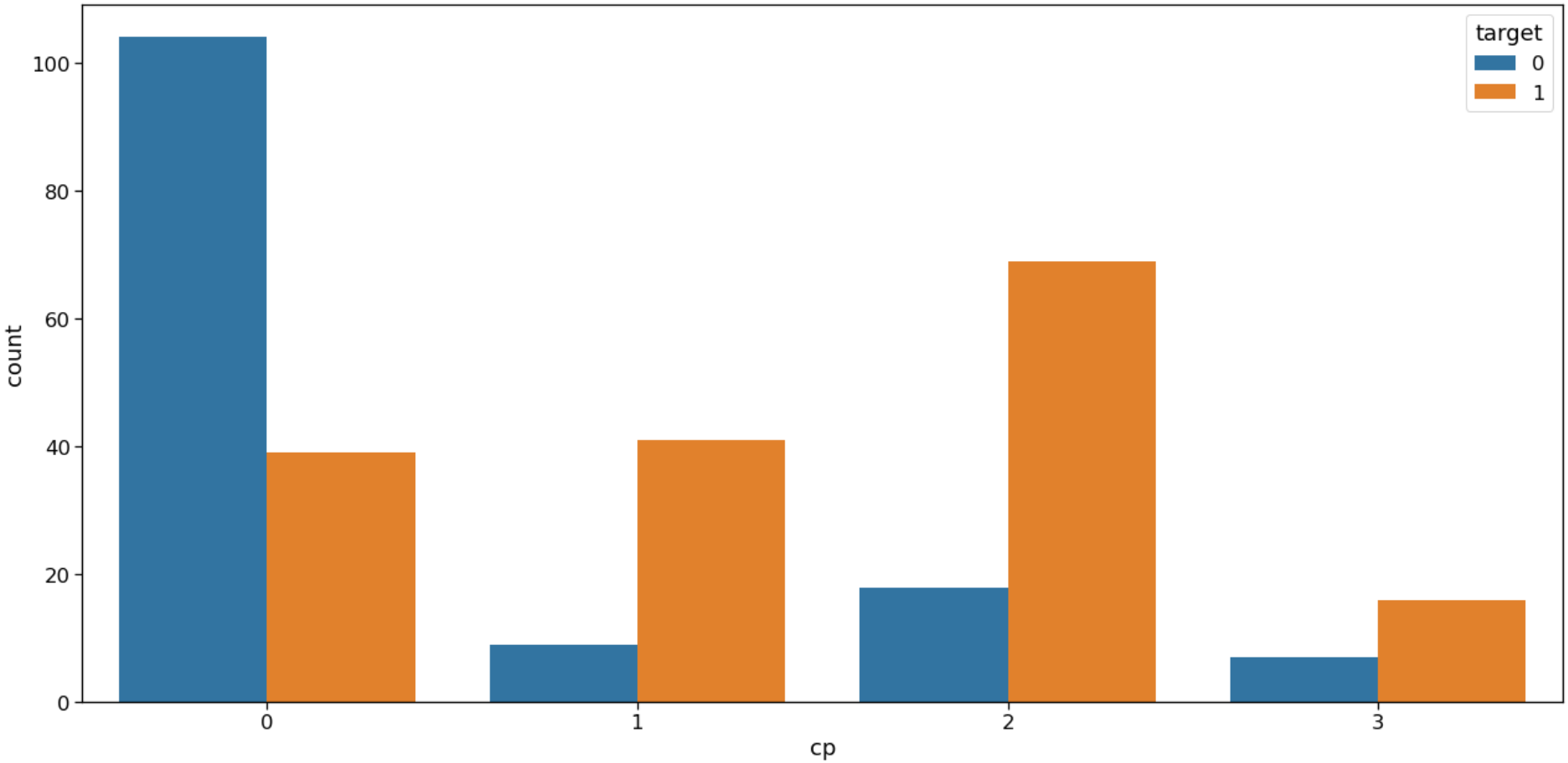
condition medium problem;

condition too bad

Analyzing cp vs target column

In [22]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['cp'],hue=data["target"])
plt.tight_layout()
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```



Inference: From the above graph we can make some inferences:

People having the least chest pain are not likely to have heart disease.

People having severe chest pain are likely to have heart disease.

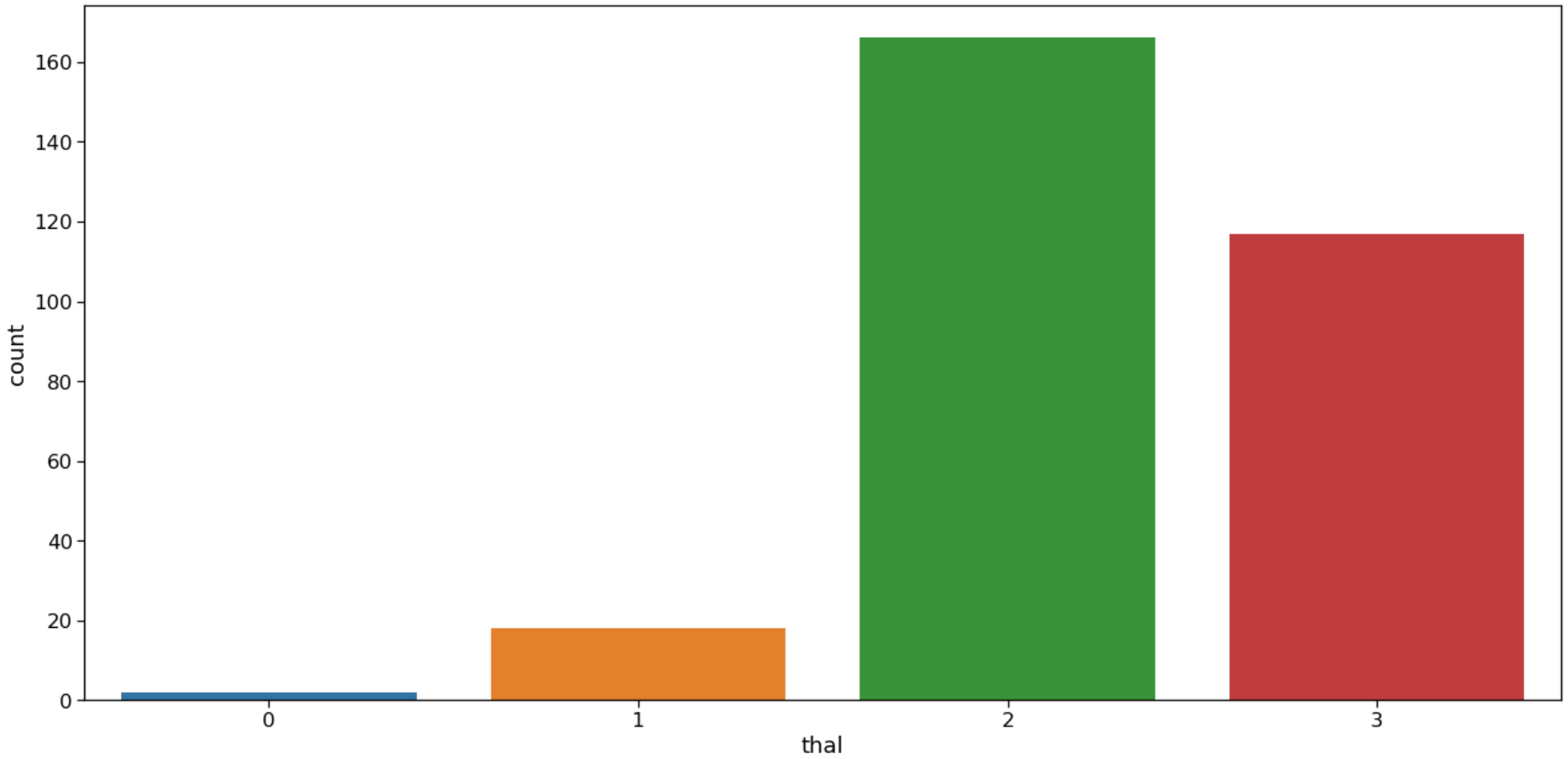
Elderly people are more likely to have chest pain.

Thal Analysis

In [23]:
```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
```

```
sns.countplot(data['thal'])
plt.tight_layout()
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

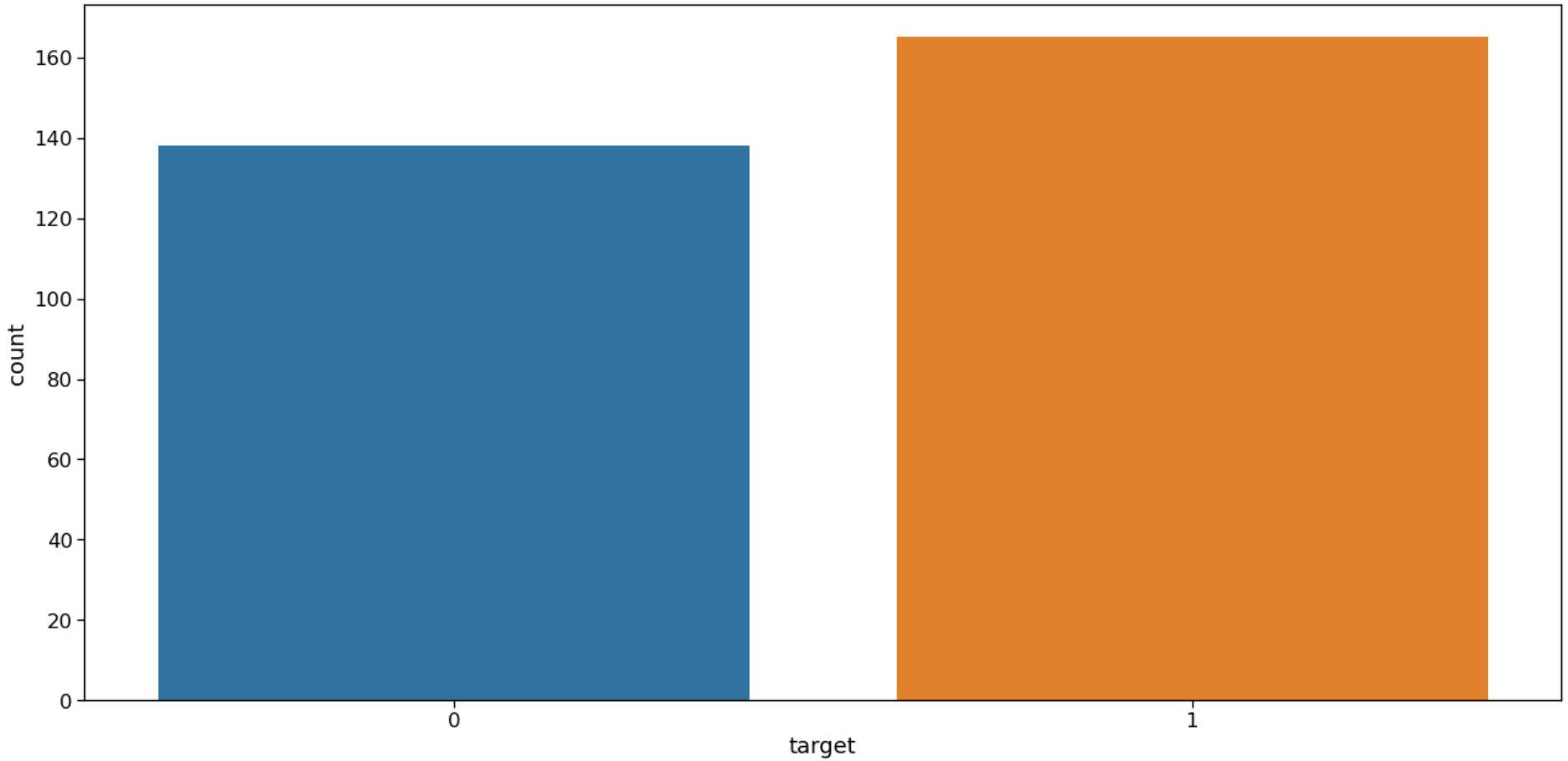


To check inbalance in Target

In [24]:

```
plt.figure(figsize=(18,9))
sns.set_context('notebook',font_scale = 1.5)
sns.countplot(data['target'])
plt.tight_layout()
```

```
C:\Users\User\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```



Inference: The ratio between 1 and 0 is much less than 1.5 which indicates that the target feature is not imbalanced. So for a balanced dataset, we can use accuracy_score as evaluation metrics.

In [ ]:

In [ ]: