In [1]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
df = pd.read_excel('C:/Users/User/Desktop/Data Science Notes/Project/PC DS - Mach
```

1. Preliminary analysis:

a. Perform preliminary data inspection and report the findings on the structure of the data, missing values, duplicates, etc.

In [3]:
```
df.head()
```

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

In [4]:
```
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 [5]:
```
# Structure of Data
df.describe().T
```

Out[5]:

	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

In [6]:
```
# Missing Values
df.isna().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
```

In [7]:
```
# Finding Duplicate Values
duplicate = df.iloc[1:,:][df.duplicated(keep='last')]
print('Duplicate Rows : ')
duplicate
```

```
Duplicate Rows :

C:\Users\User\AppData\Local\Temp\ipykernel_3752\234062564.py:2: UserWarning: Bo
olean Series key will be reindexed to match DataFrame index.
  duplicate = df.iloc[1:,:][df.duplicated(keep='last')]
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
163	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

In [8]:
```
# Finding Duplicate Values based on age
duplicate_age = df.iloc[1:,:][df.duplicated('age',keep='last')]
print('Duplicate Rows : ')
duplicate_age
```

```
Duplicate Rows :

C:\Users\User\AppData\Local\Temp\ipykernel_3752\4261517242.py:2: UserWarning: B
oolean Series key will be reindexed to match DataFrame index.
  duplicate_age = df.iloc[1:,:][df.duplicated('age',keep='last')]
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
288	57	1	0	110	335	0	1	143	1	3.0	1	1	3	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3	0
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0

261 rows × 14 columns

b. Based on these findings, remove duplicates (if any) and treat missing values using an appropriate strategy

In [9]:
```
# Removing Duplicate Values
df_sorted = df.drop_duplicates(keep='last')
```

In [10]:
```
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 [11]:
```
df_sorted.info()
```

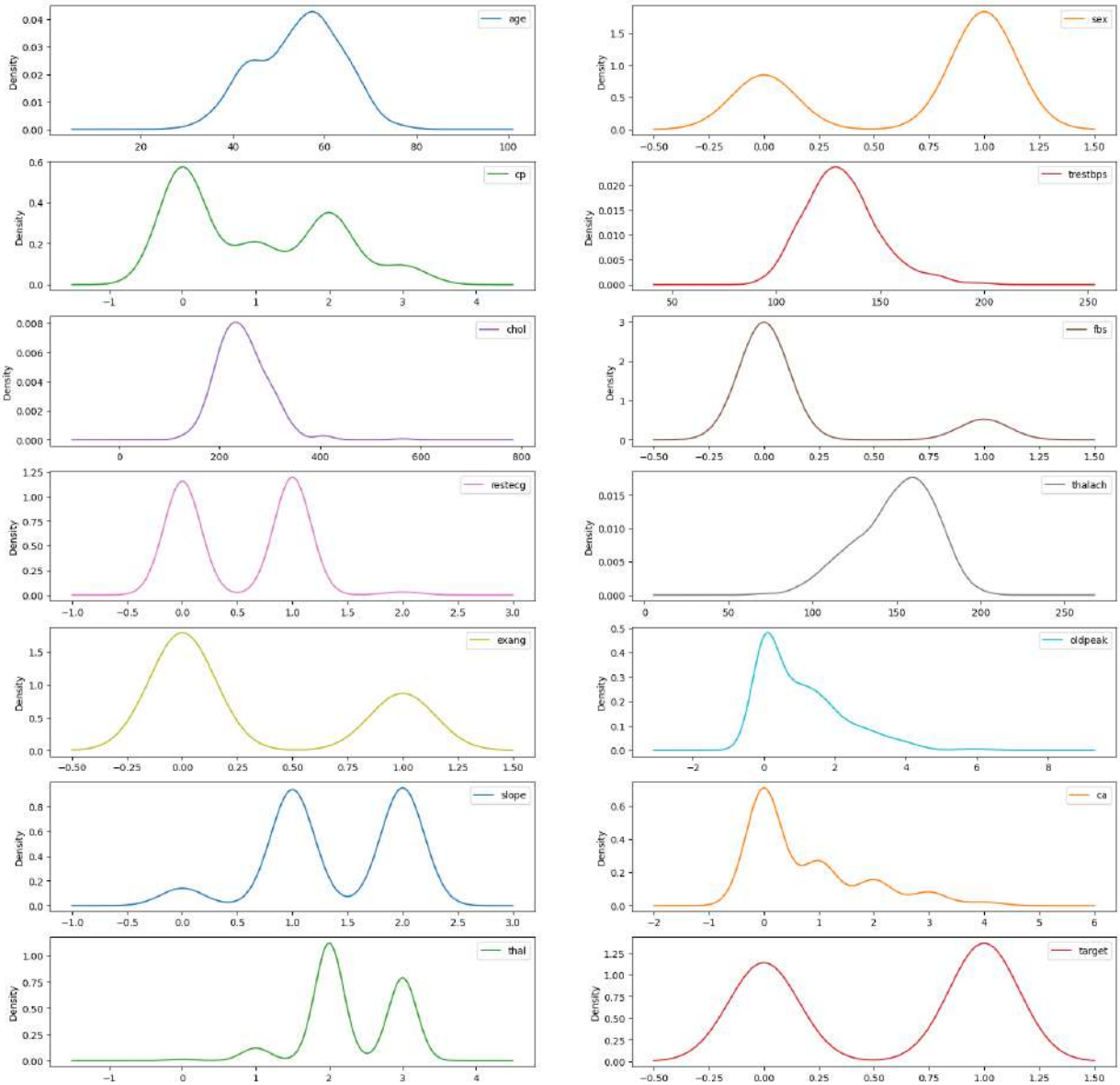
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 302 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       302 non-null    int64  
 1   sex       302 non-null    int64  
 2   cp        302 non-null    int64  
 3   trestbps  302 non-null    int64  
 4   chol      302 non-null    int64  
 5   fbs       302 non-null    int64  
 6   restecg   302 non-null    int64  
 7   thalach   302 non-null    int64  
 8   exang     302 non-null    int64  
 9   oldpeak   302 non-null    float64
 10  slope     302 non-null    int64  
 11  ca        302 non-null    int64  
 12  thal      302 non-null    int64  
 13  target    302 non-null    int64  
```

In [12]:
```
# Missing Value Treatment
df_sorted.isna().sum()
```

Out[12]:
```
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 [19]:
```
# Checking the distribution of dataframe
df.plot(kind='density', subplots=True, layout=(7,2), sharex=False, sharey=False,
```

Out[19]:
```
array([[<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>],
       [<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>]],
      dtype=object)
```



2. Prepare a report about the data explaining the distribution of the disease and the related factors using the steps listed below:

a. Get a preliminary statistical summary of the data and explore the measures of central tendencies and spread of the data

In [20]:
```
# Summary of Data
df_sorted.describe().T
```

Out[20]:

	count	mean	std	min	25%	50%	75%	max
age	302.0	54.420530	9.047970	29.0	48.00	55.5	61.00	77.0
sex	302.0	0.682119	0.466426	0.0	0.00	1.0	1.00	1.0
cp	302.0	0.963576	1.032044	0.0	0.00	1.0	2.00	3.0
trestbps	302.0	131.602649	17.563394	94.0	120.00	130.0	140.00	200.0
chol	302.0	246.500000	51.753489	126.0	211.00	240.5	274.75	564.0
fbs	302.0	0.149007	0.356686	0.0	0.00	0.0	0.00	1.0
restecg	302.0	0.526490	0.526027	0.0	0.00	1.0	1.00	2.0
thalach	302.0	149.569536	22.903527	71.0	133.25	152.5	166.00	202.0
exang	302.0	0.327815	0.470196	0.0	0.00	0.0	1.00	1.0
oldpeak	302.0	1.043046	1.161452	0.0	0.00	0.8	1.60	6.2
slope	302.0	1.397351	0.616274	0.0	1.00	1.0	2.00	2.0
ca	302.0	0.718543	1.006748	0.0	0.00	0.0	1.00	4.0
thal	302.0	2.314570	0.613026	0.0	2.00	2.0	3.00	3.0
target	302.0	0.543046	0.498970	0.0	0.00	1.0	1.00	1.0

In [21]:
```
# Measures of Central Tendancies
print('Mean : ', df_sorted.mean())
print('Median : ', df_sorted.median())
print('Mode : ', df_sorted.mode())
```

```
Mean :  age          54.420530
sex           0.682119
cp            0.963576
trestbps    131.602649
chol        246.500000
fbs           0.149007
restecg       0.526490
thalach     149.569536
exang         0.327815
oldpeak       1.043046
slope         1.397351
ca            0.718543
thal          2.314570
target        0.543046
dtype: float64
Median :  age          55.5
sex           1.0
cp            1.0
trestbps    130.0
chol        240.5
fbs           0.0
restecg       1.0
thalach     152.5
exang         0.0
oldpeak       0.8
slope         1.0
ca            0.0
thal          2.0
target        1.0
dtype: float64
Mode :       age  sex   cp  trestbps  chol  fbs  restecg  thalach  exang  oldpea
k  \
0  58.0  1.0  0.0     120.0   197  0.0      1.0    162.0    0.0      0.0
1   NaN  NaN  NaN       NaN   204  NaN      NaN      NaN    NaN      NaN
2   NaN  NaN  NaN       NaN   234  NaN      NaN      NaN    NaN      NaN

   slope   ca  thal  target
0    2.0  0.0   2.0     1.0
1    NaN  NaN   NaN     NaN
2    NaN  NaN   NaN     NaN
```

In [22]:
```
male = df_sorted[df_sorted['sex']==1]
print('Male Patients : ', len(male))
```

```
Male Patients :  206
```

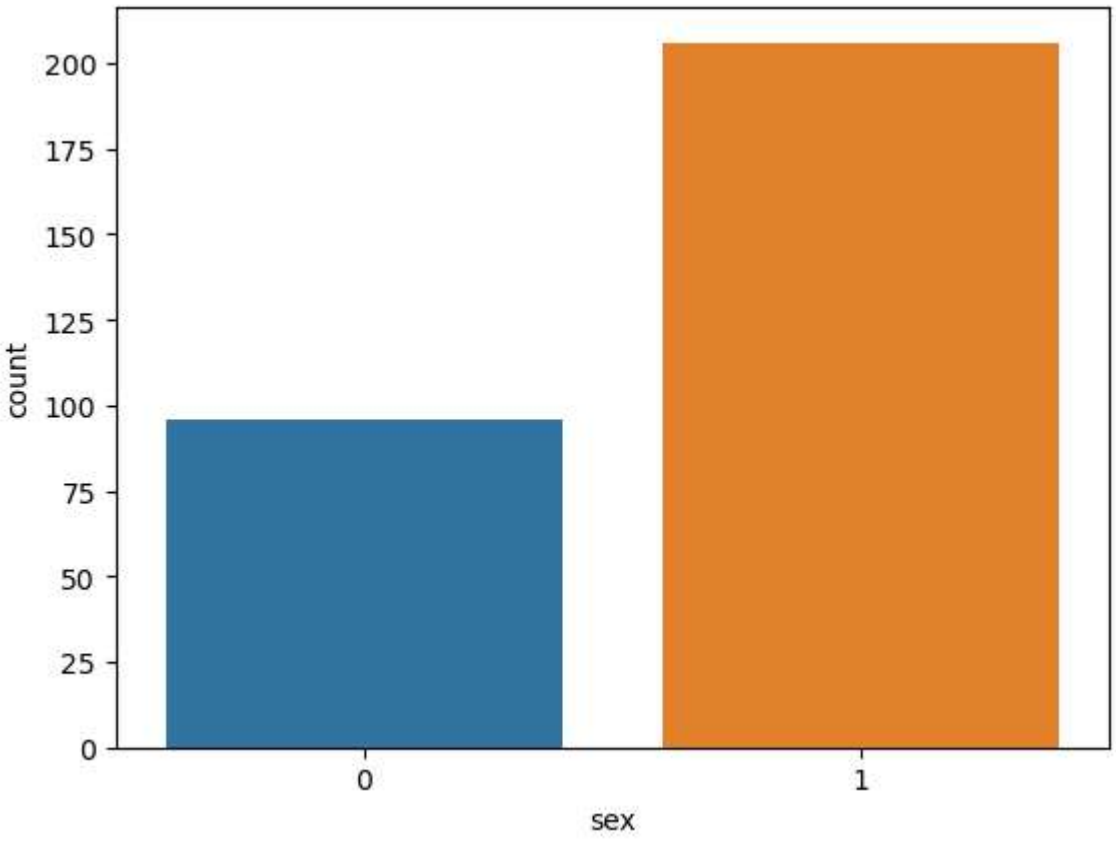
In [23]:
```
female = df_sorted[df_sorted['sex']==0]
print('Male Patients : ', len(female))
```

```
Male Patients :  96
```

b. Identify the data variables which are categorical and describe and explore these variables using the appropriate tools, such as count plot

In [24]:
```
# Count Plot based on Sex
sns.countplot(x='sex', data=df_sorted)
```

Out[24]:
```
<AxesSubplot:xlabel='sex', ylabel='count'>
```



c. Study the occurrence of CVD across the Age category

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]:
```
# We will see the sumary of how many patients suffered from Heart Attack and how
ha = df[df['target']==1]
nha = df[df['target']==0]
print('Number of patients sufered from Heart Attack : ', len(ha))
print('Number of patients not sufered from Heart Attack : ', len(nha))
sns.countplot(x='target', data=df)
```

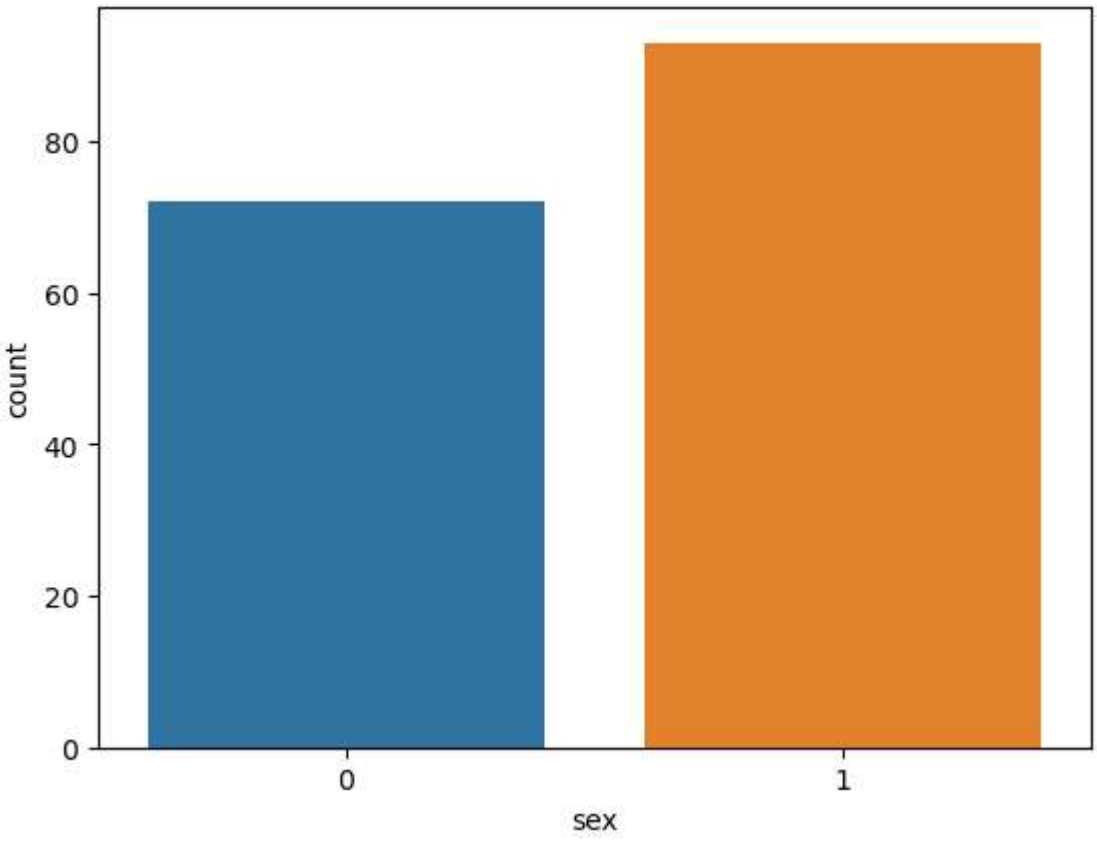
```
Number of patients sufered from Heart Attack :  165
Number of patients not sufered from Heart Attack :  138
```

Out[26]:
```
<AxesSubplot:xlabel='target', ylabel='count'>
```

In [27]:
```
# Occurance of CVD across to Sex category
ha_male = ha[ha['sex']==1]
ha_female = ha[ha['sex']==0]
print('Number of males suffered from heart attack : ', len(ha_male))
print('Number of females suffered from heart attack : ', len(ha_female))
sns.countplot(x='sex', data=ha)
```

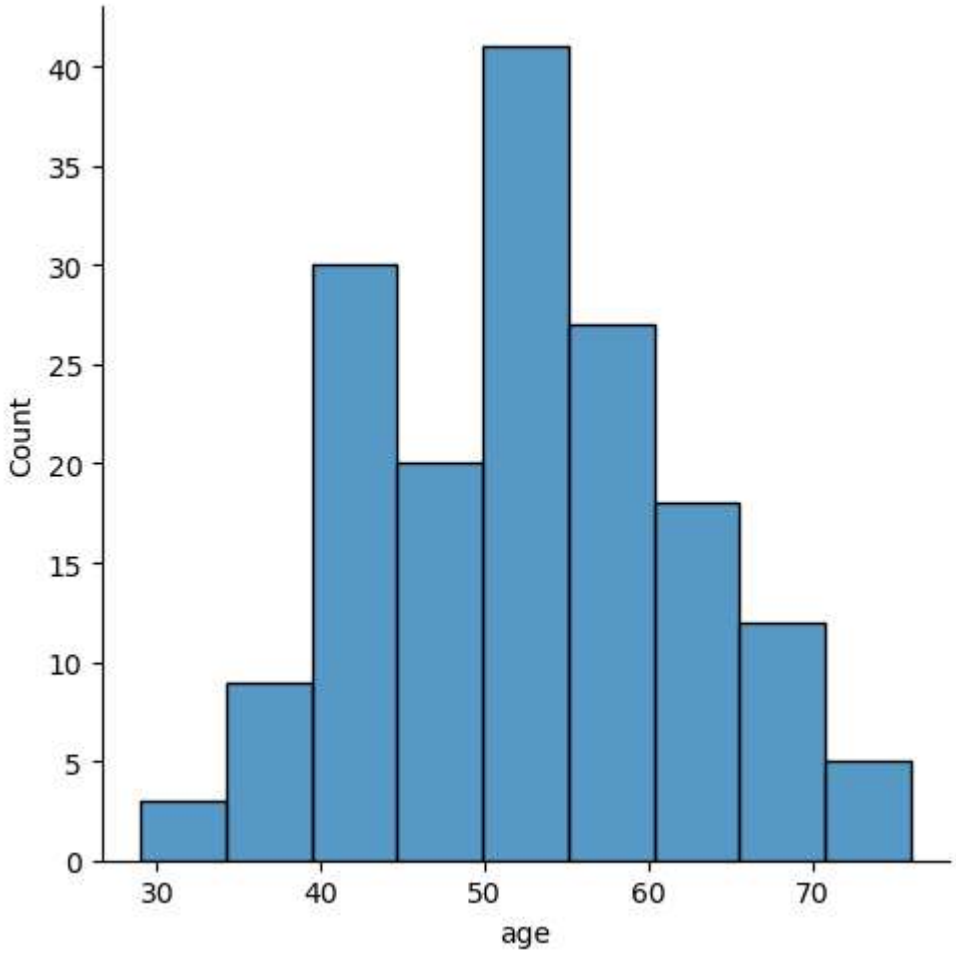
```
Number of males suffered from heart attack :  93
Number of females suffered from heart attack :  72
```

Out[27]: <AxesSubplot:xlabel='sex', ylabel='count'>



In [28]:
```
# Occurances of CVD according to Age category
ha_age = ha[['age','target']]
ha_age
sns.displot(x='age', data=ha_age)
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x30a0d3df0>



d. Study the composition of all patients with respect to the Sex category

In [29]:
```
male = df[df['sex']==1]
print('Number of Male Patients : ', len(male))
print('Male Paient Composition : ')
male
```

```
Number of Male Patients :  207
Male Paient Composition :
```

Out[29]:

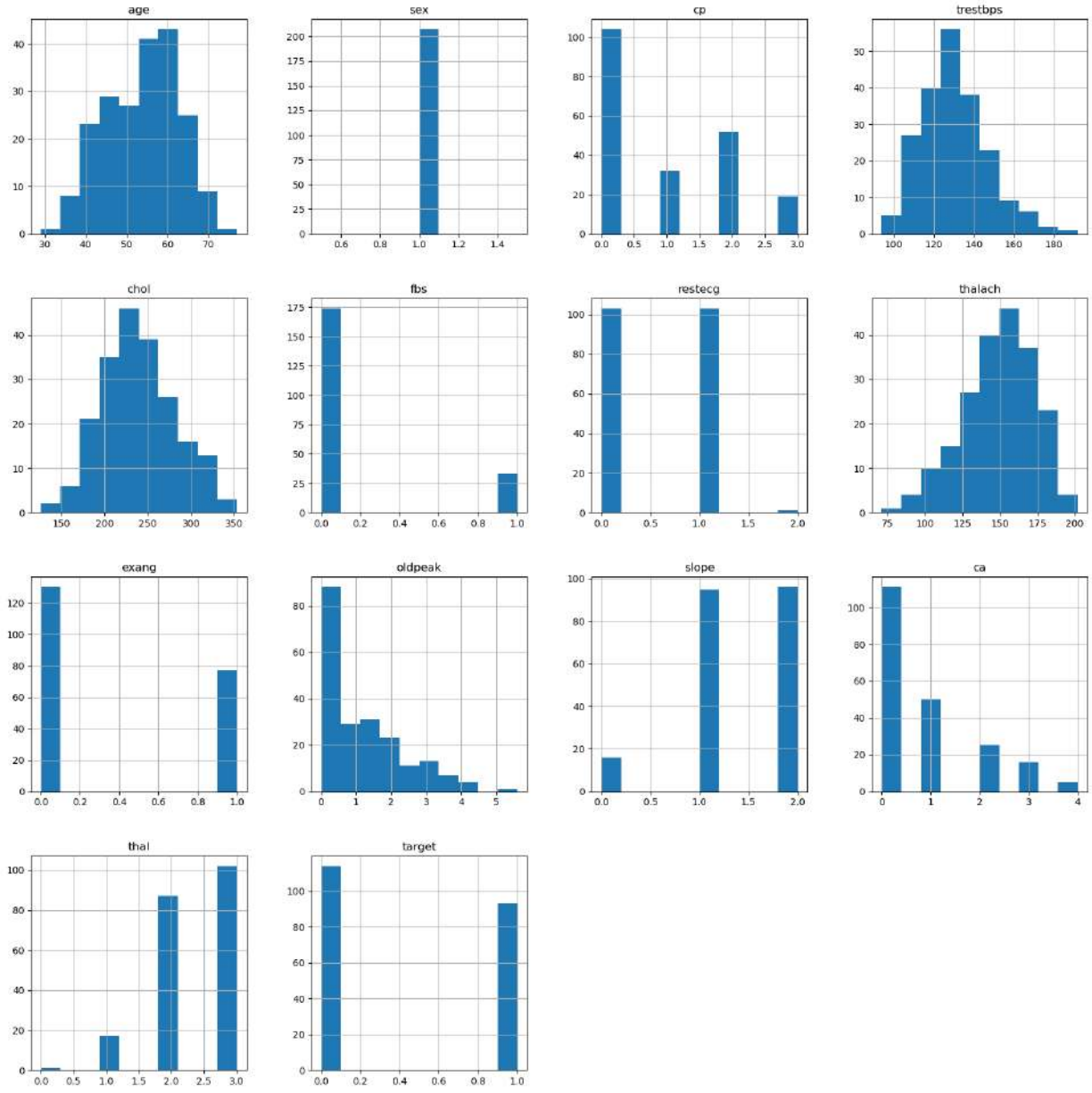
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
295	63	1	0	140	187	0	0	144	1	4.0	2	2	3	
297	59	1	0	164	176	1	0	90	0	1.0	1	2	1	
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	

In [33]:
```
male.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 207 entries, 0 to 301
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       207 non-null    int64
 1   sex       207 non-null    int64
 2   cp        207 non-null    int64
 3   trestbps  207 non-null    int64
 4   chol      207 non-null    int64
 5   fbs       207 non-null    int64
 6   restecg   207 non-null    int64
 7   thalach   207 non-null    int64
 8   exang     207 non-null    int64
 9   oldpeak   207 non-null    float64
 10  slope     207 non-null    int64
 11  ca        207 non-null    int64
 12  thal      207 non-null    int64
 13  target    207 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 24.3 KB
```

In [35]:
```
# Plotting the distribution of male patients
male.hist(figsize=(20,20))
```

Out[35]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'cp'}>,
        <AxesSubplot:title={'center':'trestbps'}>],
       [<AxesSubplot:title={'center':'chol'}>,
        <AxesSubplot:title={'center':'fbs'}>,
        <AxesSubplot:title={'center':'restecg'}>,
        <AxesSubplot:title={'center':'thalach'}>],
       [<AxesSubplot:title={'center':'exang'}>,
        <AxesSubplot:title={'center':'oldpeak'}>,
        <AxesSubplot:title={'center':'slope'}>,
        <AxesSubplot:title={'center':'ca'}>],
       [<AxesSubplot:title={'center':'thal'}>,
        <AxesSubplot:title={'center':'target'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



In [36]:
```
female = df[df['sex']==0]
print('Number of Female Patients : ', len(female))
print('Female Patient Composition : ')
female
```

```
Number of Female Patients :  96
Female Patient Composition :
```

Out[36]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
14	58	0	3	150	283	1	0	162	0	1.0	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
289	55	0	0	128	205	0	2	130	1	2.0	1	1	3	0
292	58	0	0	170	225	1	0	146	1	2.8	1	2	1	0
296	63	0	0	124	197	0	1	136	1	0.0	1	0	2	0
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

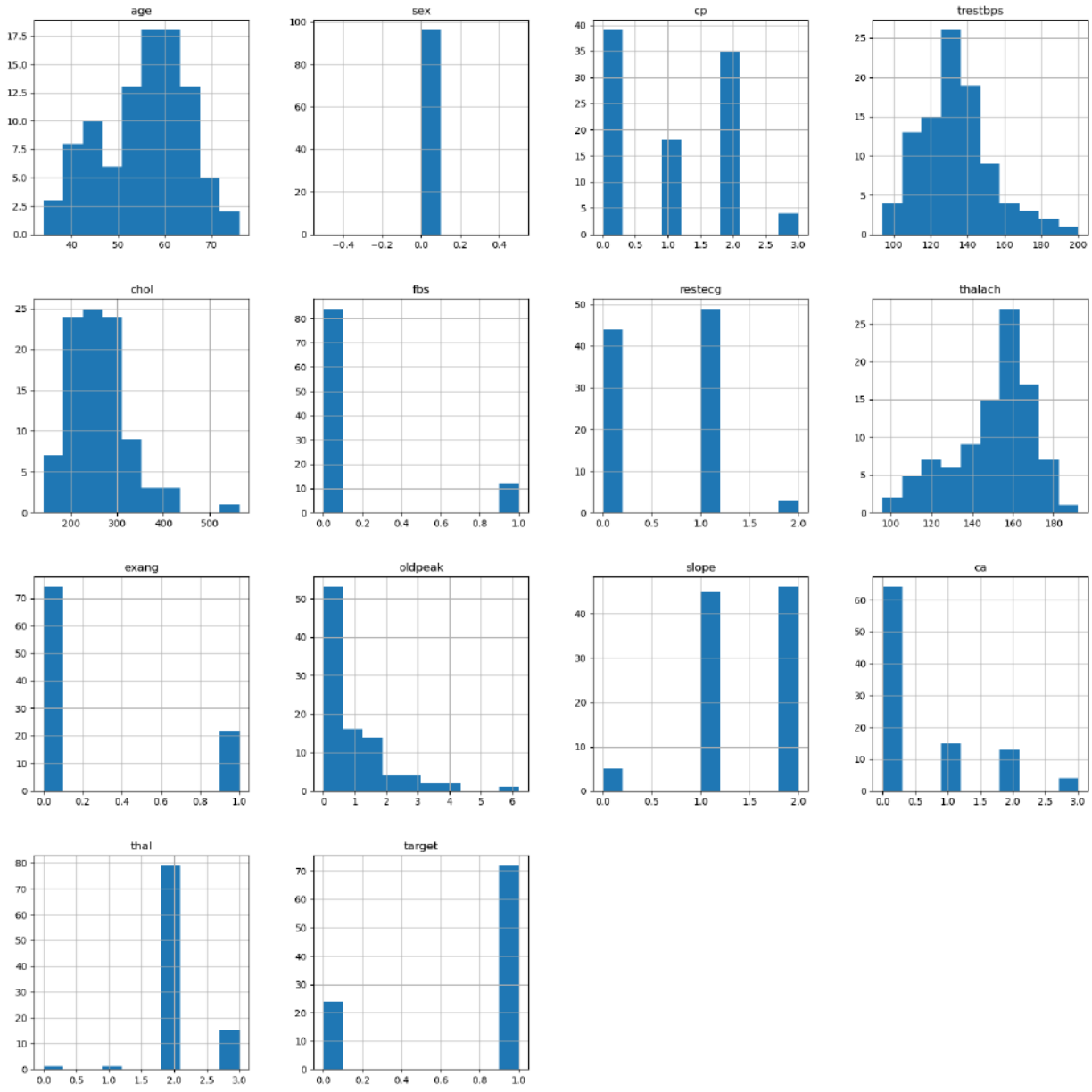
96 rows × 14 columns

In [37]:
```
female.info()
```

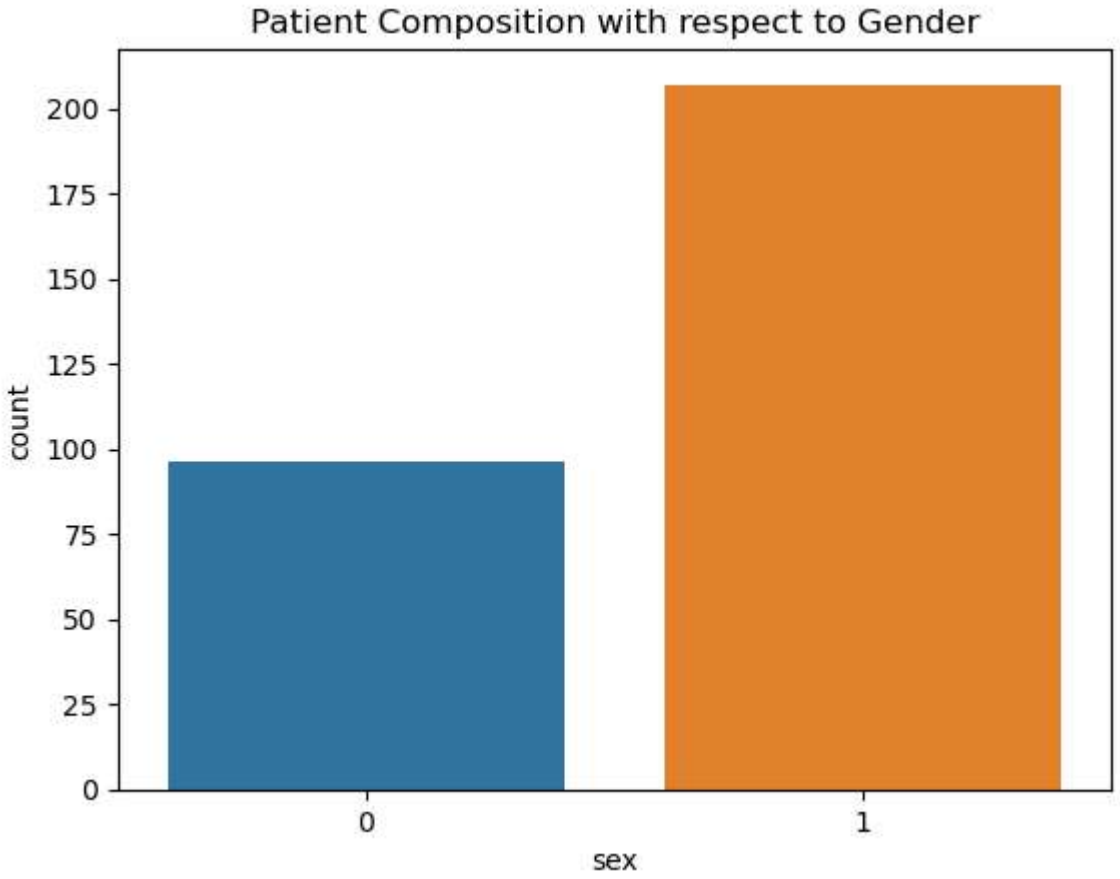
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 96 entries, 2 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       96 non-null     int64  
 1   sex       96 non-null     int64  
 2   cp        96 non-null     int64  
 3   trestbps  96 non-null     int64  
 4   chol      96 non-null     int64  
 5   fbs       96 non-null     int64  
 6   restecg   96 non-null     int64  
 7   thalach   96 non-null     int64  
 8   exang     96 non-null     int64  
 9   oldpeak   96 non-null     float64
 10  slope     96 non-null     int64  
 11  ca        96 non-null     int64  
 12  thal      96 non-null     int64  
 13  target    96 non-null     int64  
dtypes: float64(1), int64(13)
memory usage: 11.2 KB
```

In [38]:
```
# Plotting the distribution of male patients
female.hist(figsize=(20,20))
```

Out[38]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'cp'}>,
        <AxesSubplot:title={'center':'trestbps'}>],
       [<AxesSubplot:title={'center':'chol'}>,
        <AxesSubplot:title={'center':'fbs'}>,
        <AxesSubplot:title={'center':'restecg'}>,
        <AxesSubplot:title={'center':'thalach'}>],
       [<AxesSubplot:title={'center':'exang'}>,
        <AxesSubplot:title={'center':'oldpeak'}>,
        <AxesSubplot:title={'center':'slope'}>,
        <AxesSubplot:title={'center':'ca'}>],
       [<AxesSubplot:title={'center':'thal'}>,
        <AxesSubplot:title={'center':'target'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



In [39]:
```
sns.countplot(x='sex', data=df)
plt.title('Patient Composition with respect to Gender')
plt.show()
```



e. Study if one can detect heart attacks based on anomalies in the resting blood pressure (trestbps) of a patient

In [40]:
```
df['trestbps'].head(10)
```

Out[40]:
```
0    145
1    130
2    130
3    120
4    120
5    140
6    140
7    120
8    172
9    150
Name: trestbps, dtype: int64
```

In [42]:
```
df['trestbps'].describe()
```

Out[42]:
```
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 [43]:
```
# Detection of Heart Attack based on trestbps
df_trestbpstar = df[['trestbps','target']]
df_trestbpstar
```

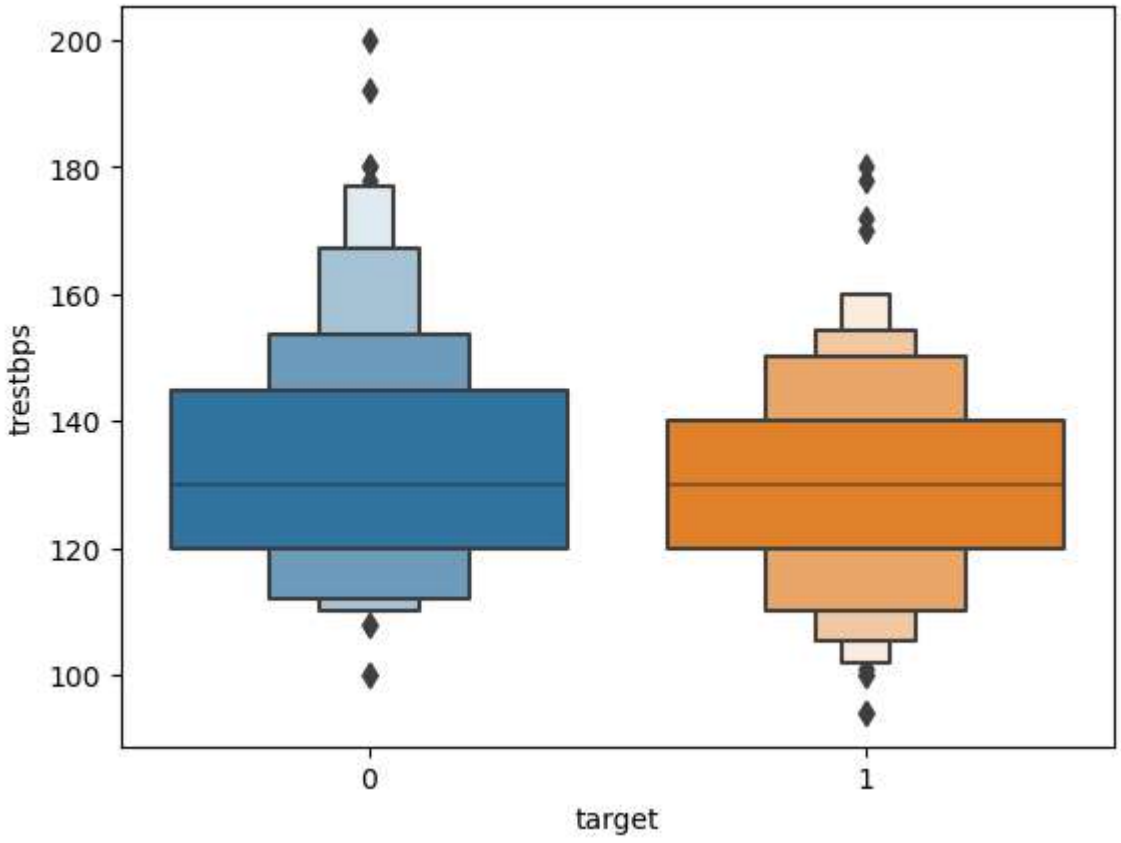
Out[43]:

	trestbps	target
0	145	1
1	130	1
2	130	1
3	120	1
4	120	1
...	...	...
298	140	0
299	110	0
300	144	0
301	130	0
302	130	0

303 rows × 2 columns

In [47]:
```
# Detection of Heart Attack based on trestbps
sns.boxenplot(x='target', y='trestbps', data=df_trestbpstar)
```

Out[47]: <AxesSubplot:xlabel='target', ylabel='trestbps'>



f. Describe the relationship between cholesterol levels and a target variable

In [49]:
```
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
```

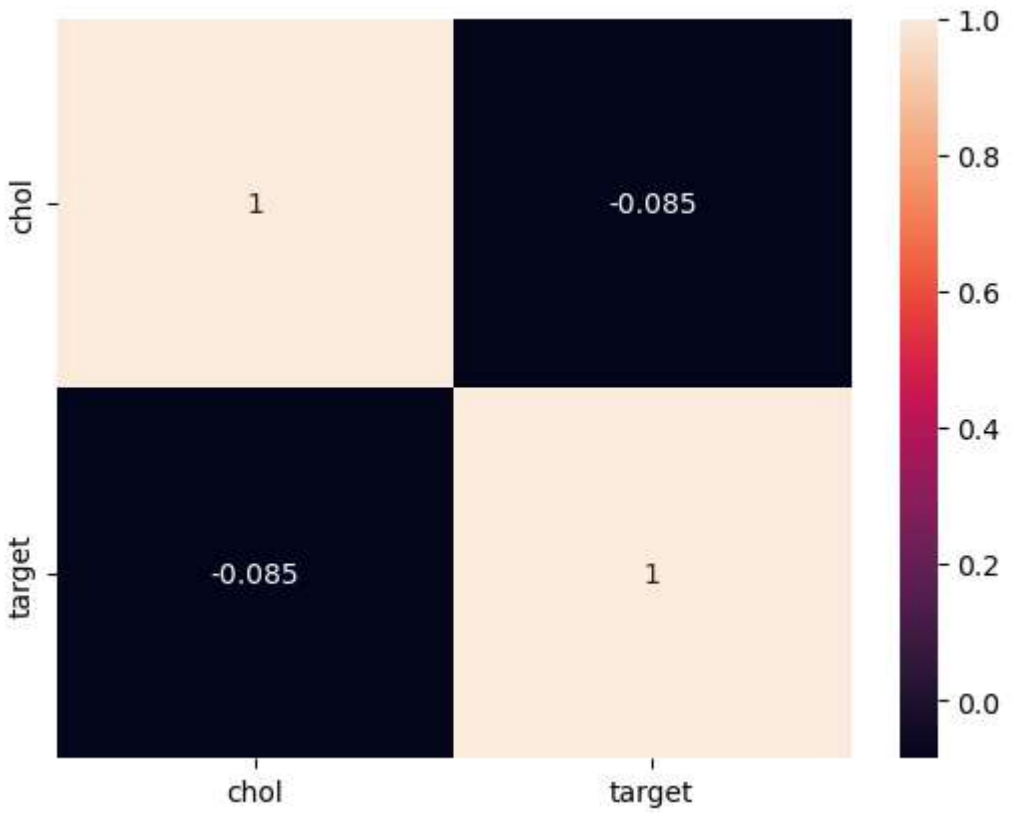
In [50]:
```
df_choltar = df[['chol','target']]
df_choltar.head()
```

Out[50]:

	chol	target
0	233	1
1	250	1
2	204	1
3	236	1
4	354	1

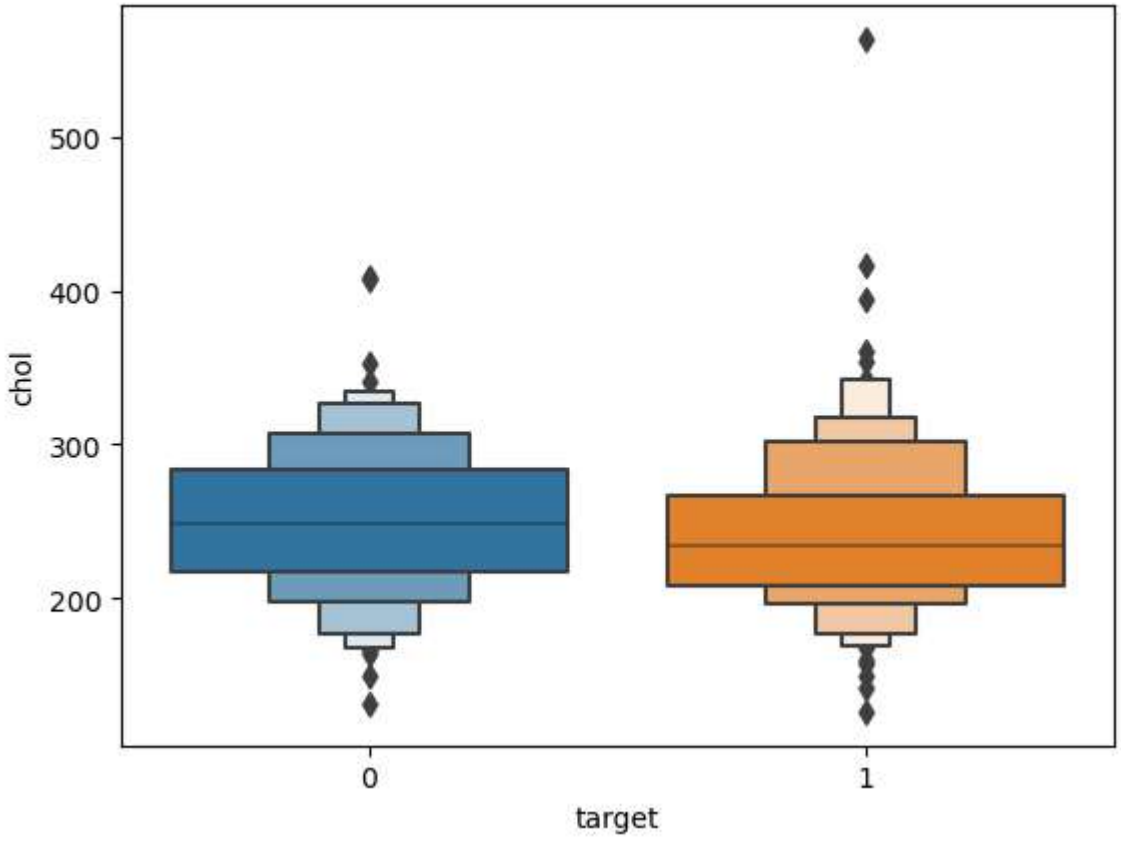
In [52]:
```
# Correlation Between cholestrol level & target
sns.heatmap(df_choltar.corr(), annot=True)
```

Out[52]: <AxesSubplot:>



In [53]:
```
sns.boxenplot(x='target', y='chol', data=df_choltar)
```

Out[53]:
```
<AxesSubplot:xlabel='target', ylabel='chol'>
```

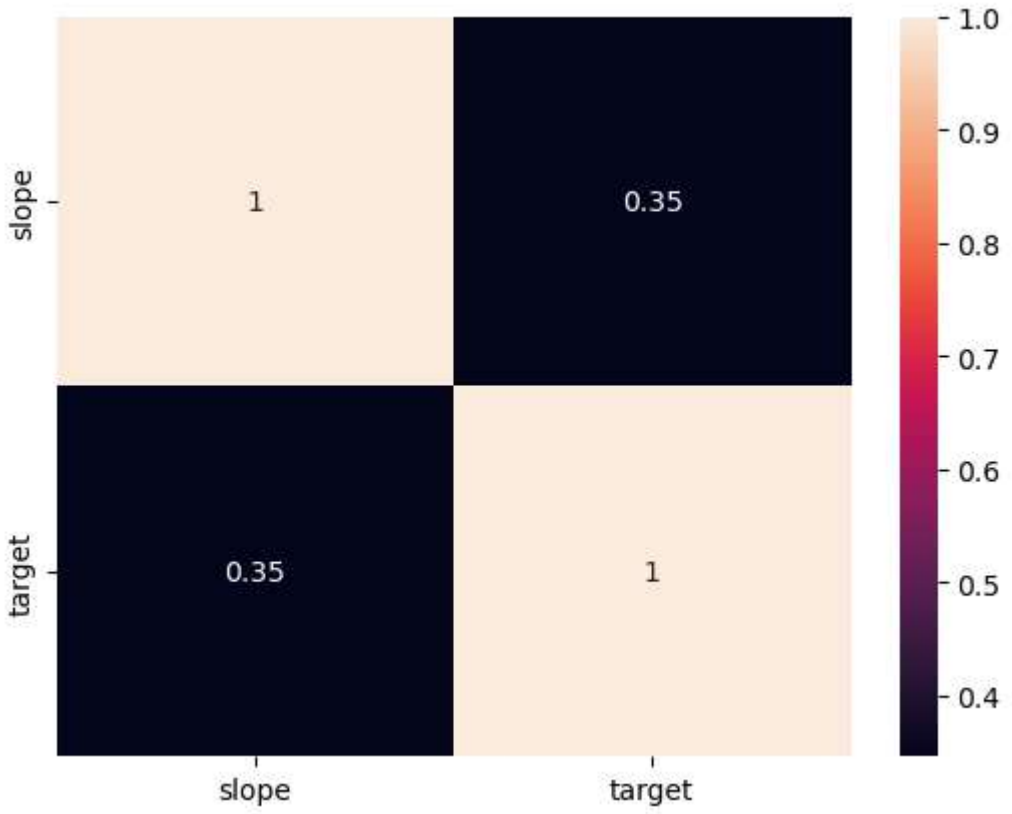


g. State what relationship exists between peak exercising and the occurrence of a heart attack

In [54]:
```
df_slopetar = df[['slope','target']]
```

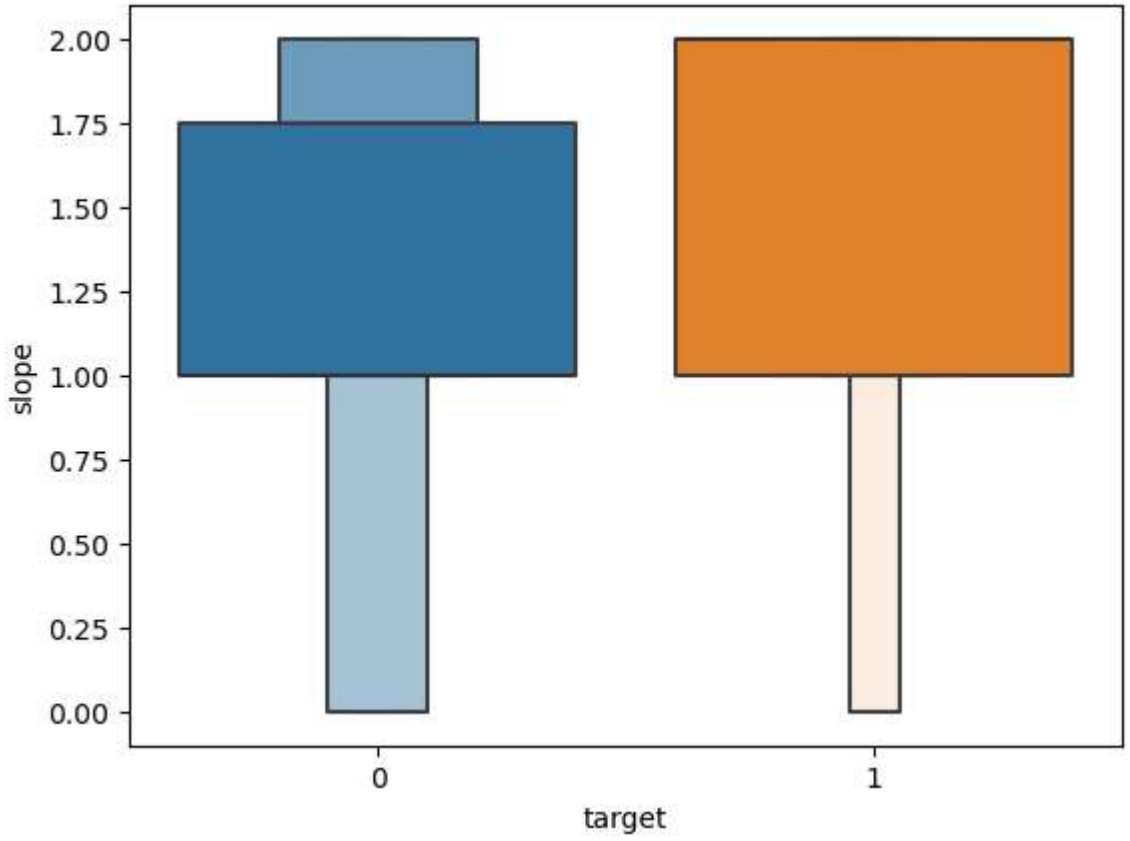
In [55]:
```
sns.heatmap(df_slopetar.corr(), annot=True)
```

Out[55]: <AxesSubplot:>



In [56]:
```
sns.boxenplot(x='target', y='slope', data=df_slopetar)
```

Out[56]:
```
<AxesSubplot:xlabel='target', ylabel='slope'>
```



h. Check if thalassemia is a major cause of CVD

In [57]:
```
df[['target','thalach']]
```

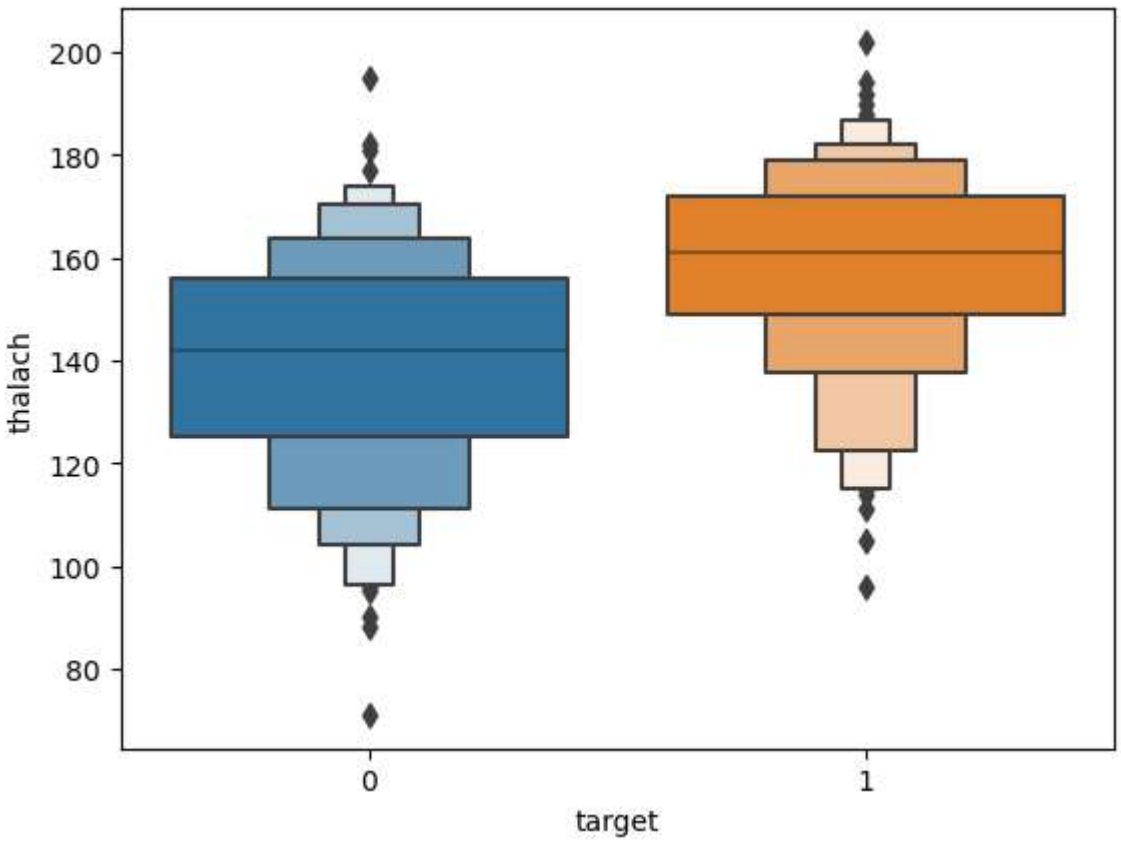
Out[57]:

	target	thalach
0	1	150
1	1	187
2	1	172
3	1	178
4	1	163
...	...	...
298	0	123
299	0	132
300	0	141
301	0	115
302	0	174

303 rows × 2 columns

In [58]:
```
sns.boxenplot(x='target', y='thalach', data=df)
```

Out[58]:
```
<AxesSubplot:xlabel='target', ylabel='thalach'>
```



i. List how the other factors determine the occurrence of CVD

In [64]:
```
# ha = df[df['target']==1] : Occurance of heart attack
ha
```

Out[64]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
160	56	1	1	120	240	0	1	169	0	0.0	0	0	2	1
161	55	0	1	132	342	0	1	166	0	1.2	2	0	2	1
162	41	1	1	120	157	0	1	182	0	0.0	2	0	2	1
163	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

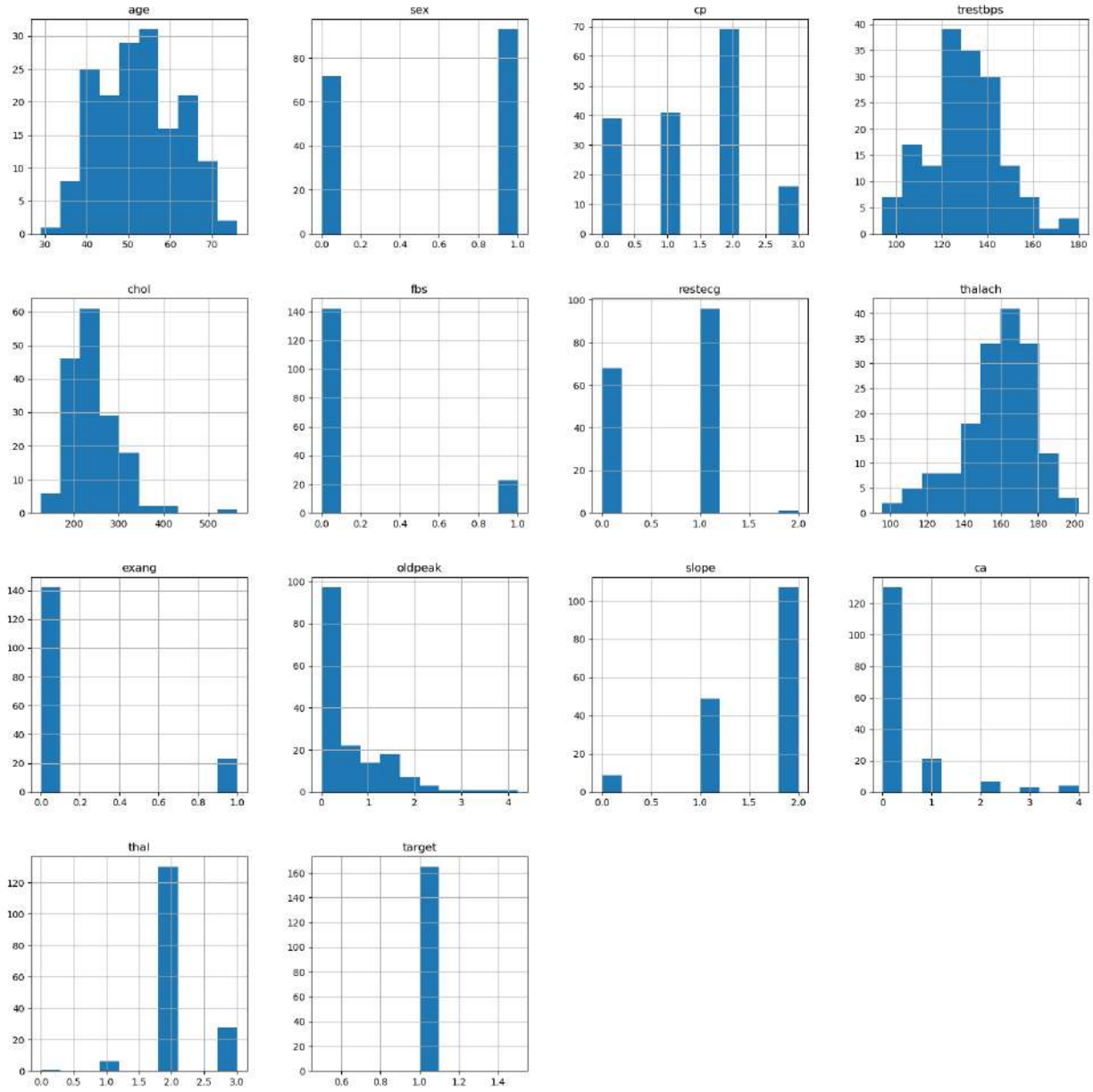
165 rows × 14 columns

In [65]:
```
ha.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 165 entries, 0 to 164
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       165 non-null    int64  
 1   sex       165 non-null    int64  
 2   cp        165 non-null    int64  
 3   trestbps  165 non-null    int64  
 4   chol      165 non-null    int64  
 5   fbs       165 non-null    int64  
 6   restecg   165 non-null    int64  
 7   thalach   165 non-null    int64  
 8   exang     165 non-null    int64  
 9   oldpeak   165 non-null    float64
 10  slope     165 non-null    int64  
 11  ca        165 non-null    int64  
 12  thal      165 non-null    int64  
 13  target    165 non-null    int64  
dtypes: float64(1), int64(13)
```

In [66]:
```
# Relation of other factors with respect to occurance of CVD
ha.hist(figsize=(20,20))
```

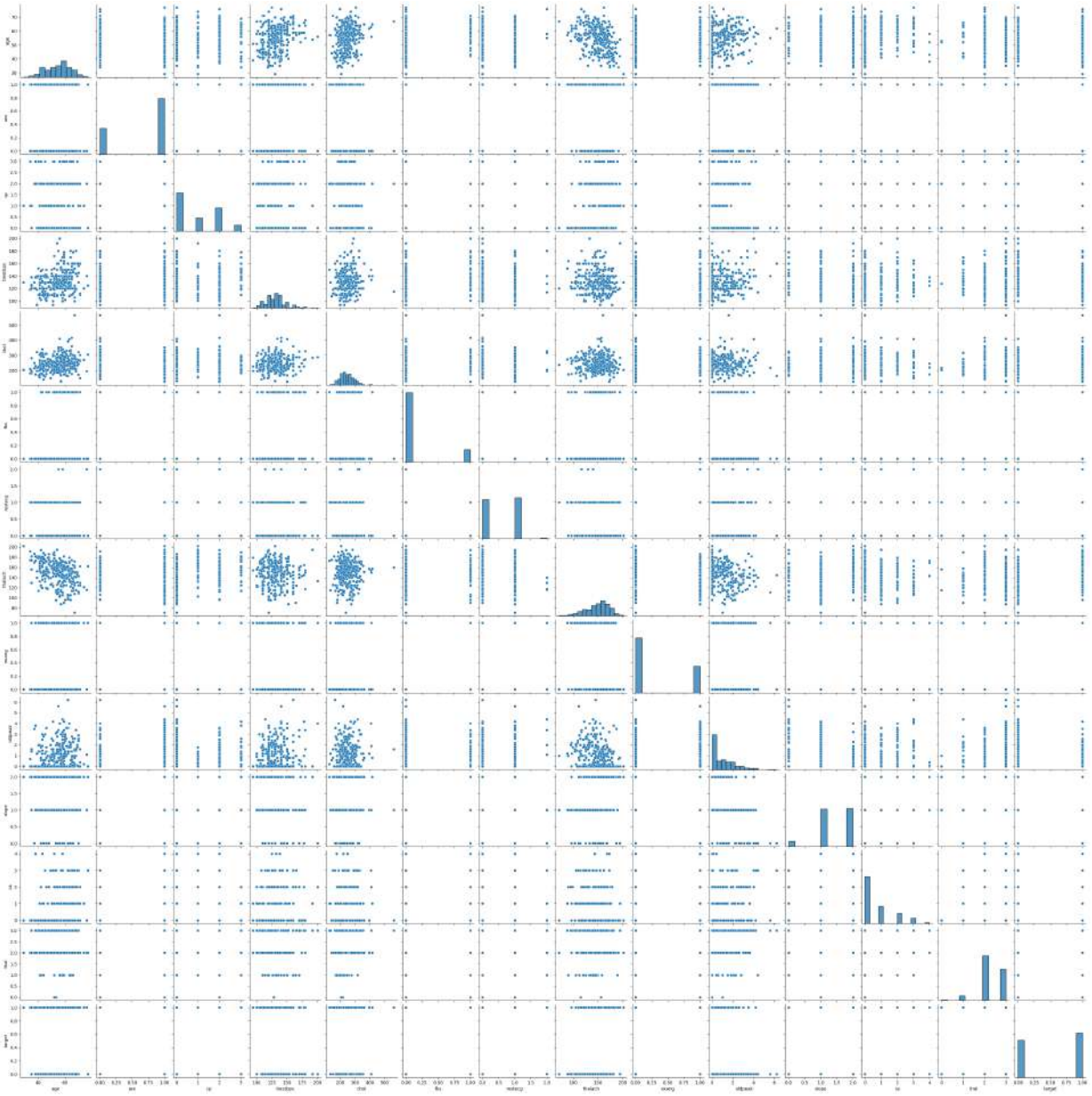
Out[66]:
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'cp'}>,
        <AxesSubplot:title={'center':'trestbps'}>],
       [<AxesSubplot:title={'center':'chol'}>,
        <AxesSubplot:title={'center':'fbs'}>,
        <AxesSubplot:title={'center':'restecg'}>,
        <AxesSubplot:title={'center':'thalach'}>],
       [<AxesSubplot:title={'center':'exang'}>,
        <AxesSubplot:title={'center':'oldpeak'}>,
        <AxesSubplot:title={'center':'slope'}>,
        <AxesSubplot:title={'center':'ca'}>],
       [<AxesSubplot:title={'center':'thal'}>,
        <AxesSubplot:title={'center':'target'}>, <AxesSubplot:>,
        <AxesSubplot:>]], dtype=object)
```



j. Use a pair plot to understand the relationship between all the given variables

In [68]:
```
sns.pairplot(df)
```

Out[68]:
```
<seaborn.axisgrid.PairGrid at 0x30a1005b0>
```



3. Build a baseline model to predict the risk of a heart attack using a logistic regression and random forest and explore the results while

using correlation analysis and logistic regression (leveraging standard error and p-values from statsmodels) for feature selection

In [69]:
```
df.columns
```

Out[69]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [75]:
```
from sklearn.model_selection import train_test_split
x = df[['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exa
y = df[['target']]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_
```

In [102]:
```
# Correlation Analysis
plt.figure(figsize=(16,8))
sns.heatmap(df.corr(), annot=True)
```

Out[102]:
```
<AxesSubplot:>
```

In [86]:
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. P
lease change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\User\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:81
4: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[86]:
```
LogisticRegression()
```

In [95]:
```
# Prediction using Logistic Regression
y_pred_log = logreg.predict(x_test)
```

In [96]:
```
y_pred_log
```

Out[96]:
```
array([0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 1], dtype=int64)
```

In [97]:
```
from sklearn.metrics import accuracy_score, r2_score, precision_score, confusion_
print(classification_report(y_test, y_pred_log))
```

```
              precision    recall  f1-score   support

           0       0.79      0.79      0.79        43
           1       0.81      0.81      0.81        48

    accuracy                           0.80        91
   macro avg       0.80      0.80      0.80        91
weighted avg       0.80      0.80      0.80        91
```

In [103]:
```
# Prediction Using Random Forest
from sklearn.ensemble import RandomForestClassifier
rfcl = RandomForestClassifier(n_estimators=100)
rfcl.fit(x_train, y_train)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_3752\1460127512.py:4: DataConversion
Warning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples,), for example using ravel().
  rfcl.fit(x_train, y_train)
```

Out[103]:
```
RandomForestClassifier()
```

In [105]:
```
pred_rf = rfcl.predict(x_test)
print(classification_report(y_test, pred_rf))
```

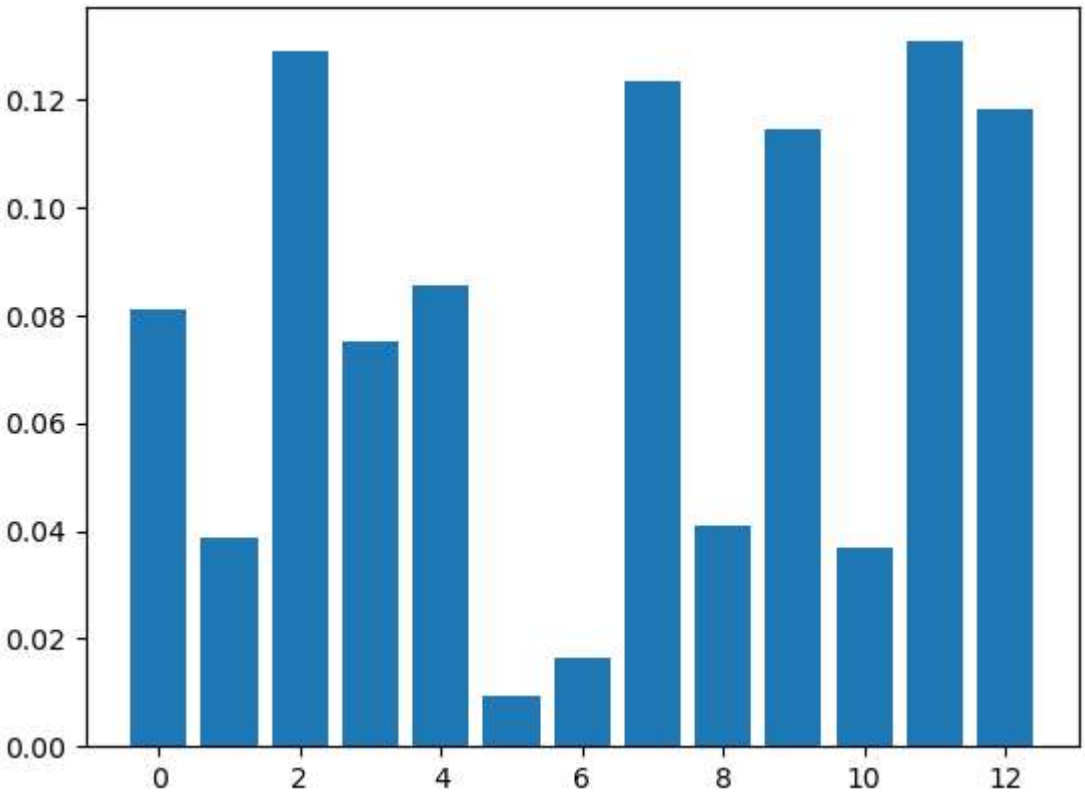
```
              precision    recall  f1-score   support

           0       0.79      0.72      0.76        43
           1       0.77      0.83      0.80        48

    accuracy                           0.78        91
   macro avg       0.78      0.78      0.78        91
weighted avg       0.78      0.78      0.78        91
```

In [136]:
```
# Feature Importance
importance = rfcl.feature_importances_
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 0.08104
Feature: 1, Score: 0.03887
Feature: 2, Score: 0.12908
Feature: 3, Score: 0.07512
Feature: 4, Score: 0.08562
Feature: 5, Score: 0.00939
Feature: 6, Score: 0.01655
Feature: 7, Score: 0.12320
Feature: 8, Score: 0.04104
Feature: 9, Score: 0.11429
Feature: 10, Score: 0.03704
Feature: 11, Score: 0.13067
Feature: 12, Score: 0.11809
```

In [137]:
```
plt.bar([x for x in range(len(importance))], importance)
```

Out[137]:
```
<BarContainer object of 13 artists>
```



In [108]:
```
print('Accuracy Score using Logistic Regression: ', accuracy_score(y_test, y_pred
print('Accuracy Score using Random Forest : ', accuracy_score(y_test, pred_rf))
```

```
Accuracy Score using Logistic Regression:  0.8021978021978022
Accuracy Score using Random Forest :  0.7802197802197802
```

In [115]:
```
df.columns
```

Out[115]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [116]:
```
# Logistic Regression using statsmodel
import statsmodels
import statsmodels.formula.api as smf

model = smf.logit('target ~ age + sex + cp + trestbps + chol+ fbs + restecg + tha
```

```
Optimization terminated successfully.
         Current function value: 0.348904
         Iterations 7
```

In [117]:
```
model.summary()
```

Out[117]:

Logit Regression Results

Dep. Variable:	target	**No. Observations:**	303
Model:	Logit	**Df Residuals:**	289
Method:	MLE	**Df Model:**	13
Date:	Sat, 25 Mar 2023	**Pseudo R-squ.:**	0.4937
Time:	14:13:59	**Log-Likelihood:**	-105.72
converged:	True	**LL-Null:**	-208.82
Covariance Type:	nonrobust	**LLR p-value:**	7.262e-37

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	3.4505	2.571	1.342	0.180	-1.590	8.490
age	-0.0049	0.023	-0.212	0.832	-0.050	0.041
sex	-1.7582	0.469	-3.751	0.000	-2.677	-0.839
cp	0.8599	0.185	4.638	0.000	0.496	1.223
trestbps	-0.0195	0.010	-1.884	0.060	-0.040	0.001
chol	-0.0046	0.004	-1.224	0.221	-0.012	0.003
fbs	0.0349	0.529	0.066	0.947	-1.003	1.073
restecg	0.4663	0.348	1.339	0.181	-0.216	1.149
thalach	0.0232	0.010	2.219	0.026	0.003	0.044
exang	-0.9800	0.410	-2.391	0.017	-1.783	-0.177
oldpeak	-0.5403	0.214	-2.526	0.012	-0.959	-0.121
slope	0.5793	0.350	1.656	0.098	-0.106	1.265
ca	-0.7733	0.191	-4.051	0.000	-1.147	-0.399
thal	-0.9004	0.290	-3.104	0.002	-1.469	-0.332

In [118]:
```
print(model.params)
```

```
Intercept    3.450472
age         -0.004908
sex         -1.758181
cp           0.859851
trestbps    -0.019477
chol        -0.004630
fbs          0.034888
restecg      0.466282
thalach      0.023211
exang       -0.979981
oldpeak     -0.540274
slope        0.579288
ca          -0.773349
thal        -0.900432
dtype: float64
```

In [124]:
```
cov = model.cov_params()
std_err = np.sqrt(np.diag(cov))
print('Standard Errors of Model : ')
print(std_err)
```

```
Standard Errors of Model :
[2.57148004 0.02317542 0.46877422 0.18539712 0.01033861 0.00378222
 0.52946527 0.34826937 0.01045996 0.40978427 0.21384914 0.34980667
 0.1908849  0.29009834]
```

In [125]:
```
print('p-values of Model :')
print(model.pvalues)
```

```
p-values of Model :
Intercept    0.179653
age          0.832266
sex          0.000176
cp           0.000004
trestbps     0.059582
chol         0.220873
fbs          0.947464
restecg      0.180618
thalach      0.026485
exang        0.016782
oldpeak      0.011523
slope        0.097717
ca           0.000051
thal         0.001910
dtype: float64
```

In [ ]:
```

```