In [1]:

```
#step 1:Import The Libraries

import pandas as pd
import matplotlib
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

#step 2: Import the dataset
data=pd.read_csv('heart.csv')
data

#In this dataset columns are:

#age

#sex

#cp(chest pain) of 4 types:
#a value 0: typical angina
#b value 1: atypical angina
#c value 2: non-anginal pain
#d value3: asymptomatic

#trestbps:resting blood pressure
#chol: serum cholestoral in mg/dl
#fbs: fasting blood sugar>120mg/dl (true=1,false=0)

#restecg: resting electrocardiographic results
#1 value 0: normal
#2 value 1: having ST-T wave abnormality(T wave inversions and ST elevation or depression o
#3 value 2: showing probable or define left ventricular hypertrophy by esters criteria

#thalach: maximum heart rate achieved
#exang: exercise induced angina(True=1,False=0)
#oldpeak: ST depression induced by exercise relative to rest

#slope: the slope of the peak exercise ST segment
#1 value 1: upsloping
#2 value 2:flat
#3 value 3:downsloping

#ca:number of major vessels(0-3) colored by flurosopy
#thal:3=normal,
#     6=fixed defect,
#     7=reversable defect.

#target:0=less chance of heart attack
#       1=more chance of heart attack
```

Out[1]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tai
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

1025 rows × 14 columns

In [2]:

```
#step 3: display top 5 rows of the dataset
data.head()
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [3]:

```
#step 4:check the last 5 rows of the dataset
data.tail()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	

In [4]:

```
#step 5: Find shape of our dataset(number of rows and number of columns)
data.shape
```

Out[4]:

```
(1025, 14)
```

In [5]:

```
print("number of rows",data.shape[0])
print("number of columns",data.shape[1])
```

```
number of rows 1025
number of columns 14
```

In [6]:

```
#step 6: Get information about our dataset like total no. of rows, total no. of columns,dat
#requirement

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

In [7]:

```
#step 7: check the null values in the dataset
data.isnull().sum()
```

Out[7]:

```
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 [8]:

```
#step 8: check for duplicate data and drop them
data_dup=data.duplicated().any()
print(data_dup)
```

```
True
```

In [9]:

```
data=data.drop_duplicates()
```

In [10]:

```
data.shape
```

Out[10]:

```
(302, 14)
```

In [11]:

```
#step 9: get overall statistics about the dataset
data.describe()
```

Out[11]:

	age	sex	cp	trestbps	chol	fbs	restecg	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202

In [12]:

```
#step 10: draw correlation matrix
data.corr()
```

Out[12]:

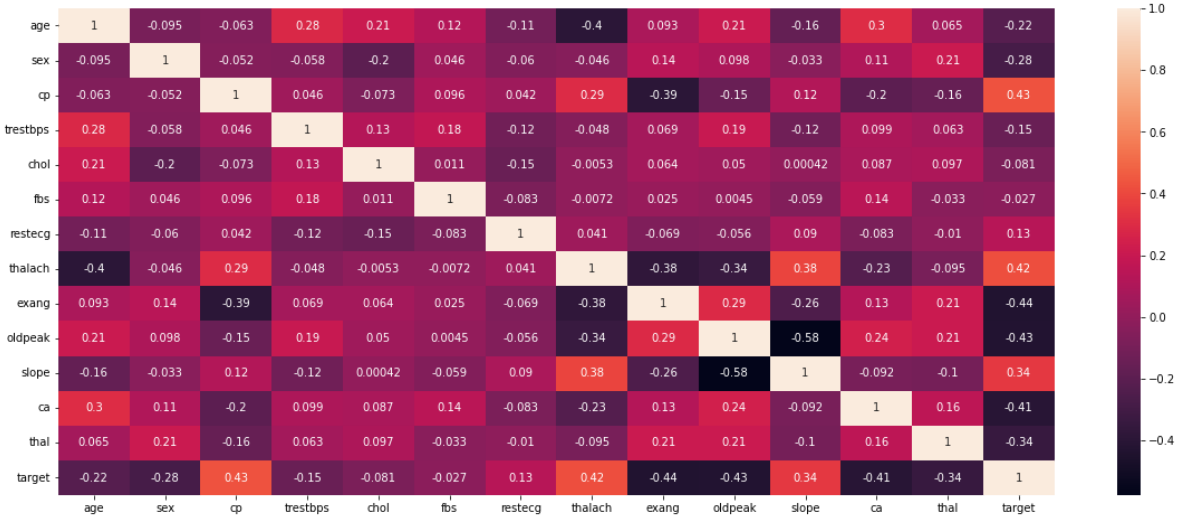
	age	sex	cp	trestbps	chol	fbs	restecg	thalach
age	1.000000	-0.094962	-0.063107	0.283121	0.207216	0.119492	-0.111590	-0.395235
sex	-0.094962	1.000000	-0.051740	-0.057647	-0.195571	0.046022	-0.060351	-0.046439
cp	-0.063107	-0.051740	1.000000	0.046486	-0.072682	0.096018	0.041561	0.293367
trestbps	0.283121	-0.057647	0.046486	1.000000	0.125256	0.178125	-0.115367	-0.048023
chol	0.207216	-0.195571	-0.072682	0.125256	1.000000	0.011428	-0.147602	-0.005308
fbs	0.119492	0.046022	0.096018	0.178125	0.011428	1.000000	-0.083081	-0.007169
restecg	-0.111590	-0.060351	0.041561	-0.115367	-0.147602	-0.083081	1.000000	0.041210
thalach	-0.395235	-0.046439	0.293367	-0.048023	-0.005308	-0.007169	0.041210	1.000000
exang	0.093216	0.143460	-0.392937	0.068526	0.064099	0.024729	-0.068807	-0.377411
oldpeak	0.206040	0.098322	-0.146692	0.194600	0.050086	0.004514	-0.056251	-0.342201
slope	-0.164124	-0.032990	0.116854	-0.122873	0.000417	-0.058654	0.090402	0.384754
ca	0.302261	0.113060	-0.195356	0.099248	0.086878	0.144935	-0.083112	-0.228311
thal	0.065317	0.211452	-0.160370	0.062870	0.096810	-0.032752	-0.010473	-0.094910
target	-0.221476	-0.283609	0.432080	-0.146269	-0.081437	-0.026826	0.134874	0.419955

In [13]:

```
plt.figure(figsize=(20,8))
sns.heatmap(data.corr(),annot=True)
```

Out[13]:

```
<AxesSubplot:>
```



In [14]:

```
#step 11: how many people have heart disease, and how many don't have heart disease in this
data['target'].value_counts()
```

Out[14]:

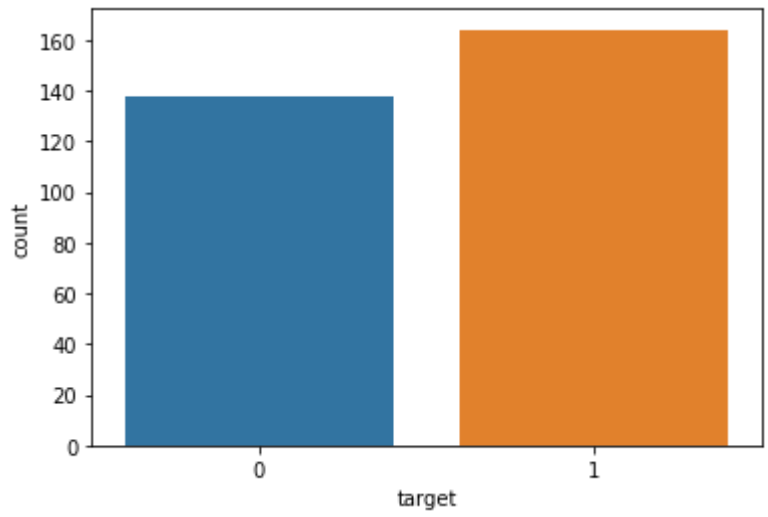
```
1    164
0    138
Name: target, dtype: int64
```

In [15]:

```
sns.countplot(data['target'])
```

Out[15]:

```
<AxesSubplot:xlabel='target', ylabel='count'>
```



In [16]:

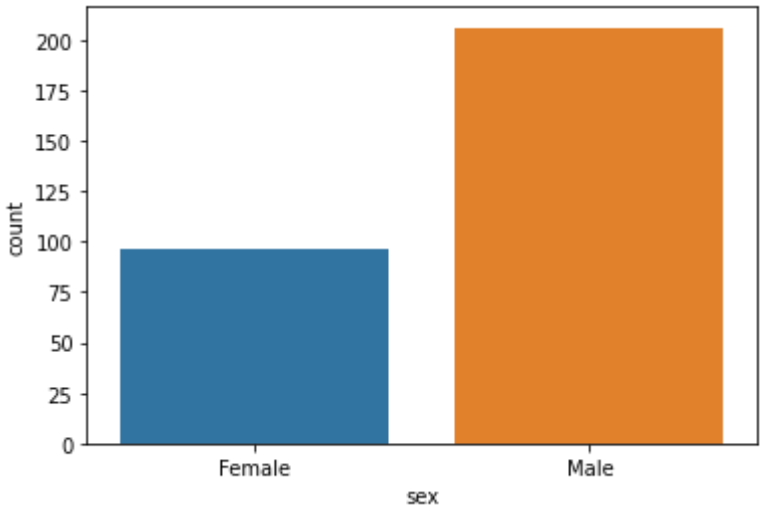
```
#step 12: find count of male and female in this dataset
data['sex'].value_counts()
```

Out[16]:

```
1    206
0     96
Name: sex, dtype: int64
```

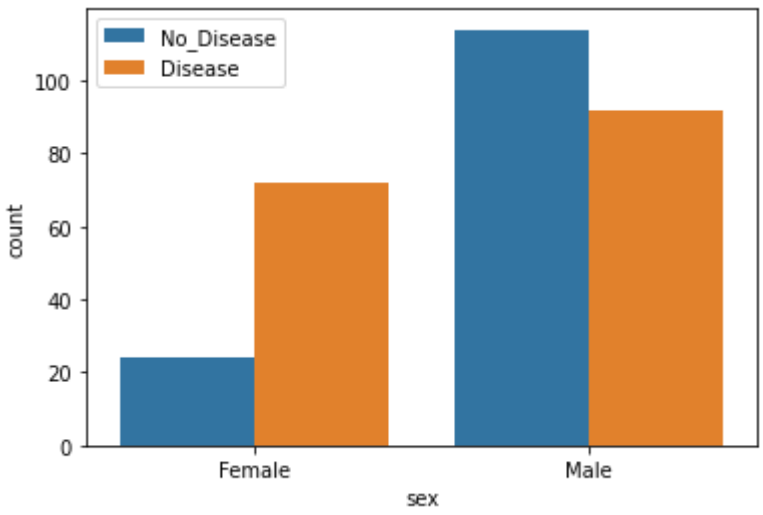
In [17]:

```
sns.countplot(data['sex'])
plt.xticks([0,1],['Female','Male'])
plt.show()
```



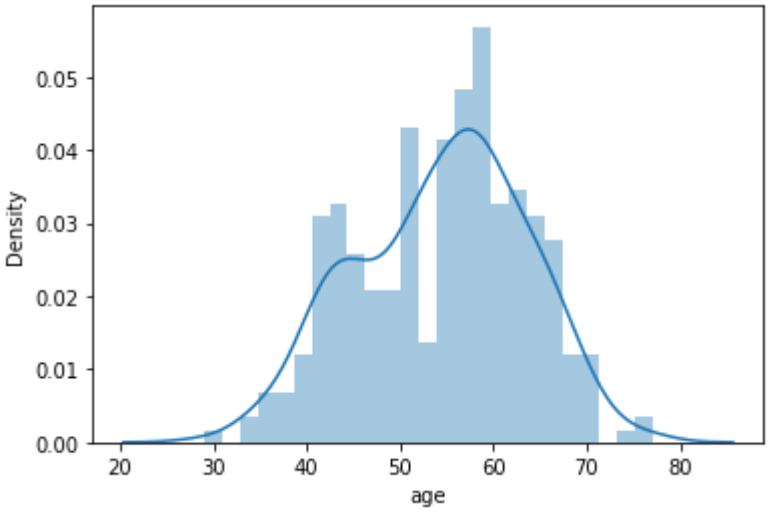
In [18]:

```
#step 13: find gender distribution according to the target variable
sns.countplot(x='sex',hue="target",data=data)
plt.xticks([1,0],['Male','Female'])
plt.legend(labels=['No_Disease','Disease'])
plt.show()
```



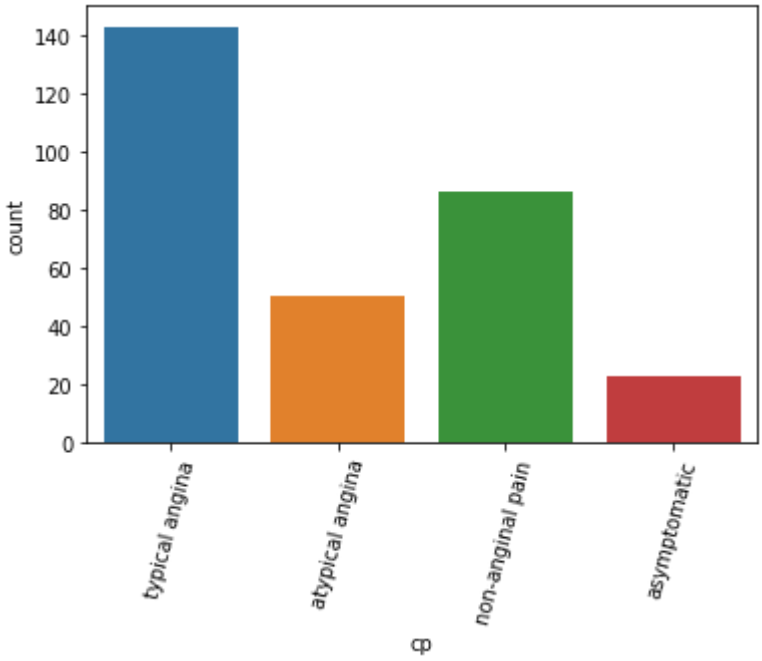
In [19]:

```
#step 14: check age distribution in dataset
sns.distplot(data['age'],bins=25)
plt.show()
```



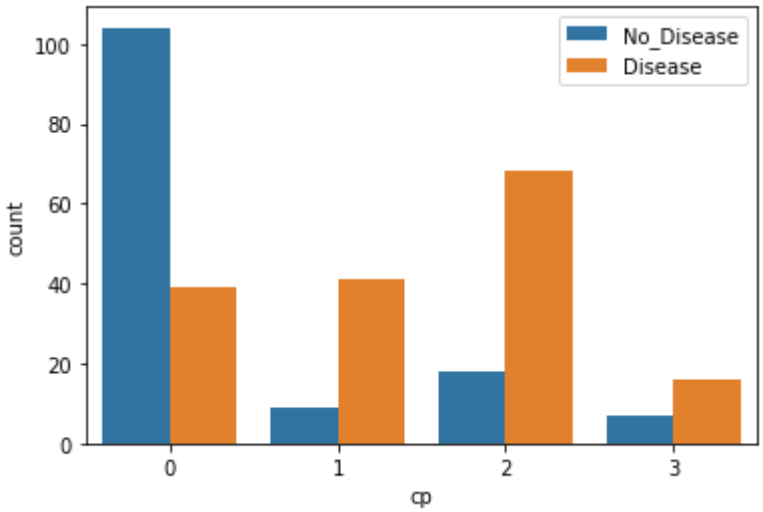
In [20]:

```
#step 15: check chest pain type

#cp(chest pain) of 4 types:
#a value 0: typical angina
#b value 1: atypical angina
#c value 2: non-anginal pain
#d value3: asymptomatic

sns.countplot(data['cp'])
plt.xticks([0,1,2,3],["typical angina","atypical angina","non-anginal pain","asymptomatic"]
plt.xticks(rotation=75)
plt.show()
```



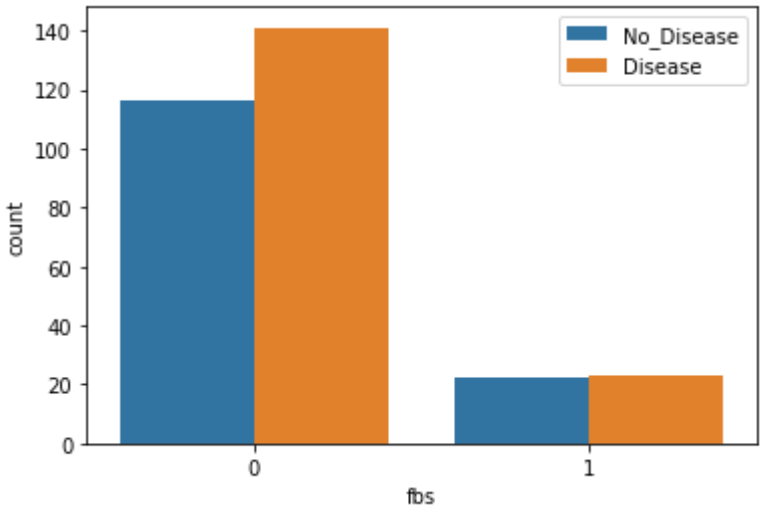
In [21]:

```
#step 16: show the chest pain distribution as per target variable
sns.countplot(x="cp",hue="target",data=data)
plt.legend(labels=['No_Disease','Disease'])
plt.show()
```



In [22]:

```
#step 17: show fasting blood sugar distribution according to target variable
sns.countplot(x="fbs",hue="target",data=data)
plt.legend(labels=['No_Disease','Disease'])
plt.show()
```

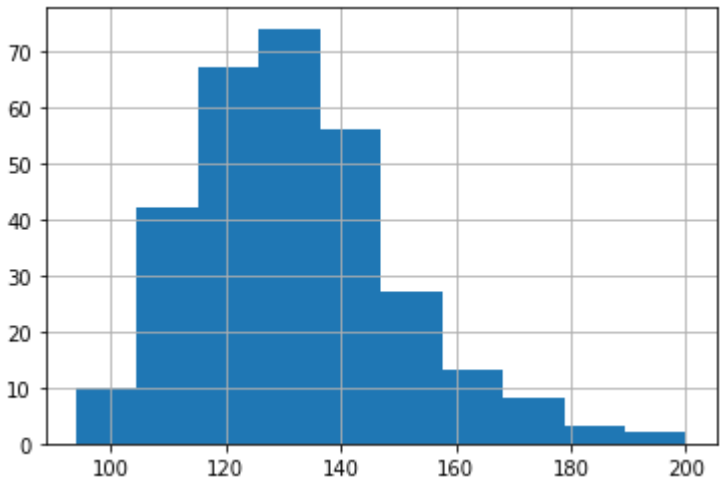


In [23]:

```
#step 18: check resting blood pressure distribution
data['trestbps'].hist()
```

Out[23]:

<AxesSubplot:>

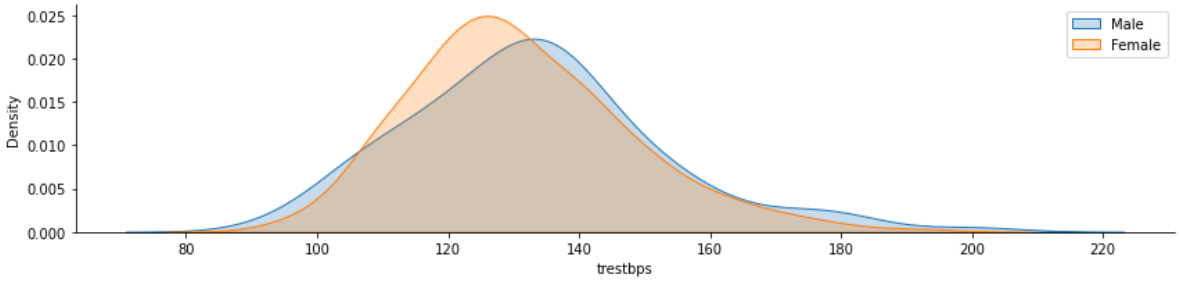


In [24]:

```
#step 19: compare resting blood pressure as per sex column
g=sns.FacetGrid(data,hue="sex",aspect=4)
g.map(sns.kdeplot,'trestbps',shade=True)
plt.legend(labels=['Male','Female'])
```

Out[24]:

```
<matplotlib.legend.Legend at 0x1fc14d71480>
```

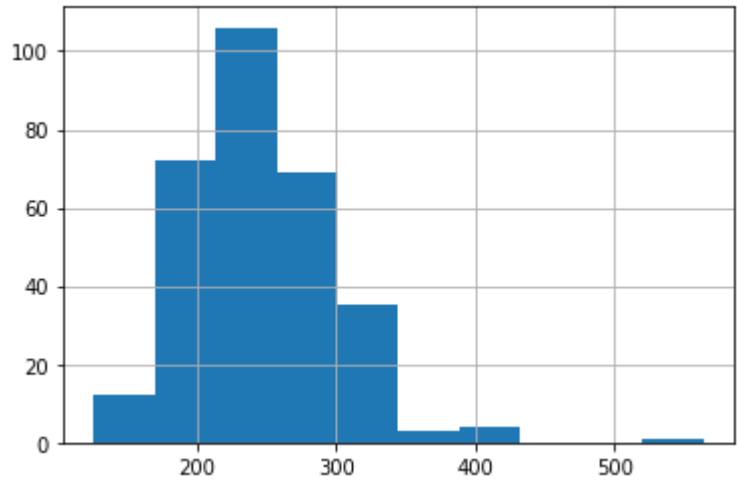


In [25]:

```
#step 20: show distribution of serum cholesterol
data['chol'].hist()
```

Out[25]:

```
<AxesSubplot:>
```



In [26]:

```
#step 21: plot continous variables
data.columns
```

Out[26]:

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

In [27]:

```
cate_val=[]
cont_val=[]

for column in data.columns:
    if data[column].nunique() <=10:
        cate_val.append(column)
    else:
        cont_val.append(column)
```

In [28]:

```
cate_val
```

Out[28]:

```
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
```

In [29]:

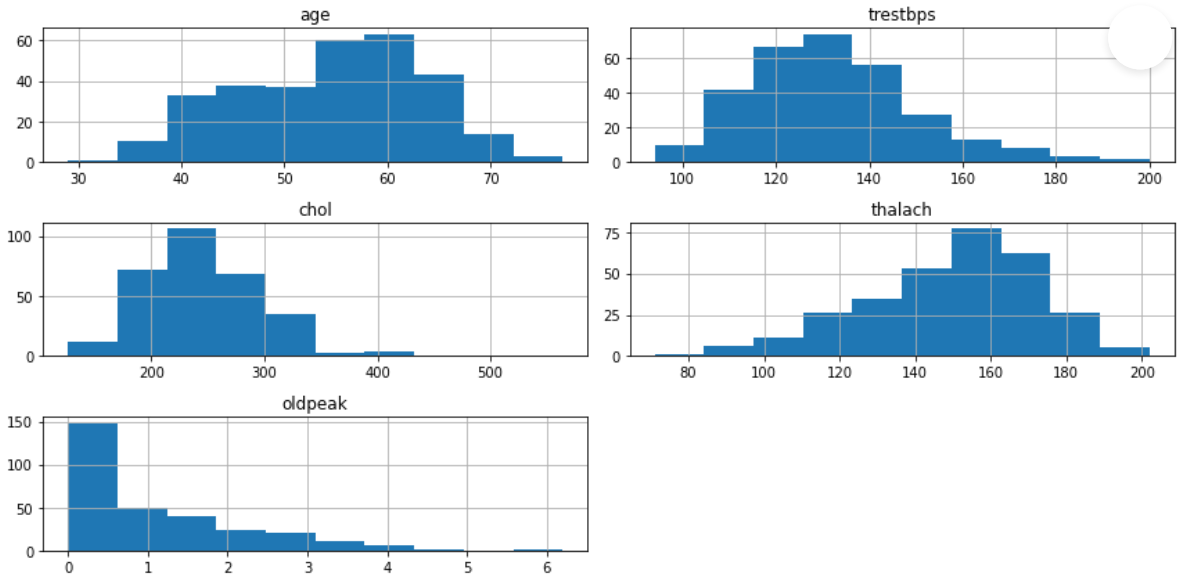
```
cont_val
```

Out[29]:

```
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

In [30]:

```
data.hist(cont_val,figsize=(12,6))
plt.tight_layout()
plt.show()
```



In [ ]:

In [ ]:

```
```