In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [7]:
```
heart = pd.read_csv('/Users/chijiokeifedili/Downloads/heart.csv')
heart.head()
```

Out[7]:

	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 [8]:
```
heart.isnull()
```

Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...
298	False	False	False	False	False	False	False	False	False	False	False	False
299	False	False	False	False	False	False	False	False	False	False	False	False
300	False	False	False	False	False	False	False	False	False	False	False	False
301	False	False	False	False	False	False	False	False	False	False	False	False
302	False	False	False	False	False	False	False	False	False	False	False	False

303 rows × 14 columns

In [15]:
```
heart.isnull().sum()
```

Out[15]:
```
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 [21]:
```
heart.isnull().sum().sum()
```

Out[21]:
```
0
```

We observe that the sum of the total missing values is 0.This concludes that there are no missing values to handle.

In [9]:
```
heart.columns
```

Out[9]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thala
ch',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

In [10]:
```
heart.dtypes
```

Out[10]:
```
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
```

In [11]:
```
heart.dtypes.value_counts()
```

Out[11]:
```
int64      13
float64     1
dtype: int64
```

In [12]:
```
heart.describe()
```

Out[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	t
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.

We can observe that the resting blood pressure has a minimum value of 126 and a maximum value of 200

In [13]:
```
heart['fbs'].unique()
```

Out[13]:
```
array([1, 0])
```

There are two values for the fbs. If fasting blood sugar > 120mg/dl then = 1 (true), if not it is allocated 0(false)

In [14]:
```
heart['exang'].unique()
```

Out[14]:
```
array([0, 1])
```

Exercise induced angina :

1 = yes

0 = no

In [14]:
```
heart['restecg'].unique()
```

Out[14]:
```
array([0, 1, 2])
```

Resting ECG

0 = normal

1 = having ST-T wave abnormality

2 = left ventricular hyperthrophy

In [15]:
```
heart['sex'].unique()
```

Out[15]:
```
array([1, 0])
```

Sex

1 = male

0 = female

In [16]:
```
heart['slope'].unique()
```

Out[16]:
```
array([0, 2, 1])
```

We observe that target, exang, restecg,fbs,slope and sex are categorical variables

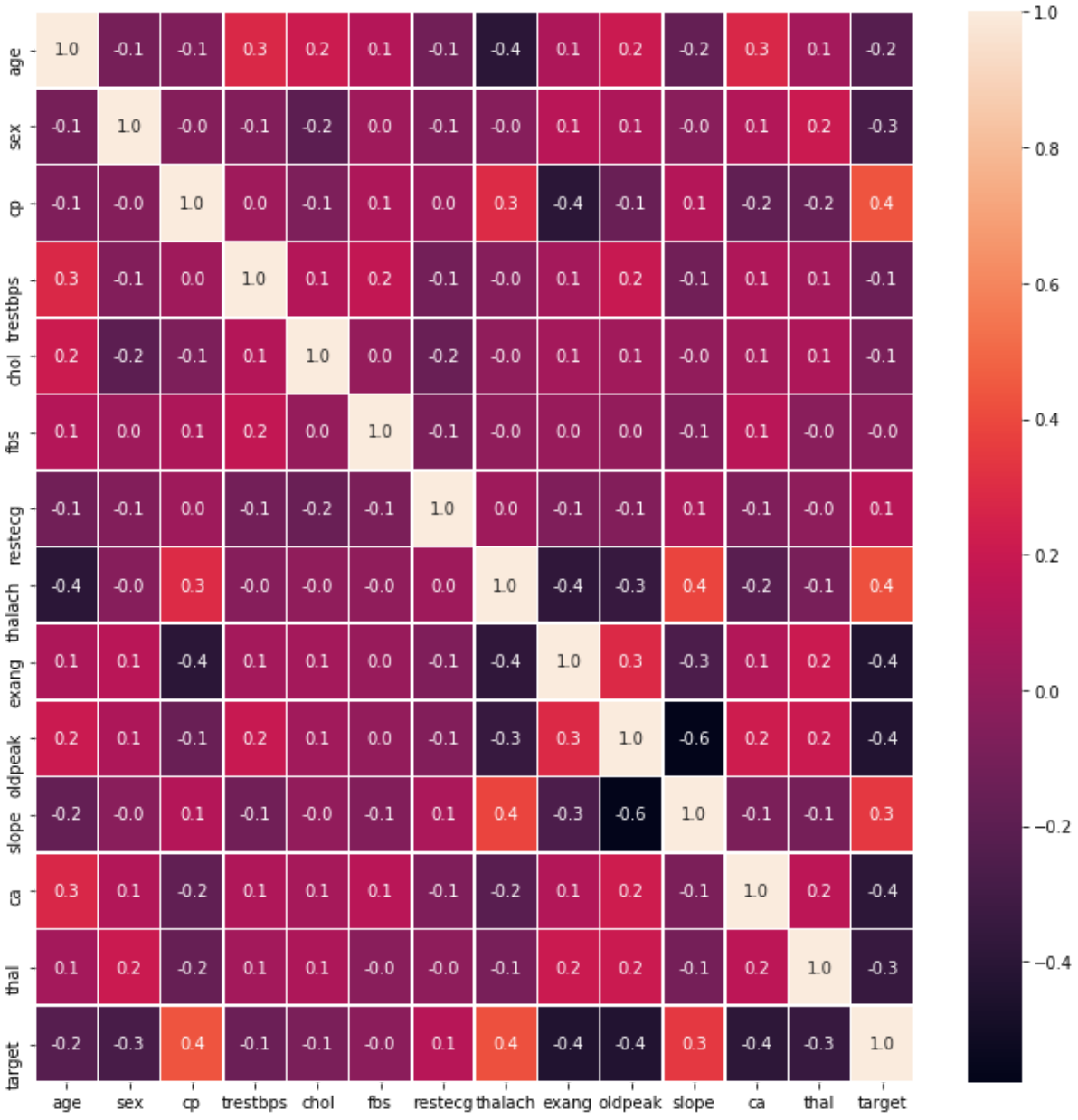
In [20]:
```
heart['age'].value_counts()
```

Out[20]:
```
58    19
57    17
54    16
59    14
52    13
51    12
62    11
44    11
60    11
56    11
64    10
41    10
63     9
67     9
55     8
45     8
42     8
53     8
61     8
65     8
43     8
66     7
50     7
48     7
46     7
49     5
47     5
39     4
35     4
68     4
70     4
40     3
71     3
69     3
38     3
34     2
37     2
77     1
76     1
74     1
29     1
Name: age, dtype: int64
```

In [22]:
```
heart.shape
```

Out[22]:
```
(303, 14)
```

In [24]:
```
f,ax = plt.subplots(figsize=(12,12))
sns.heatmap(heart.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```

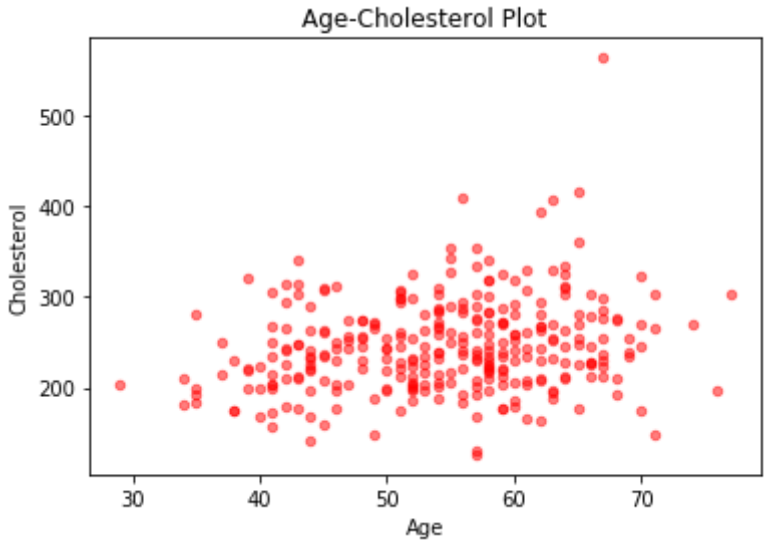


Scatter Plot(The aim is to show how much one variable is affected by the other)

Visualization with scatter plot, relation of age and cholesterol

In [26]:
```
heart.plot(kind = 'scatter',x = 'age', y = 'chol',alpha = 0.5, color =
'red')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.title('Age-Cholesterol Plot')
```

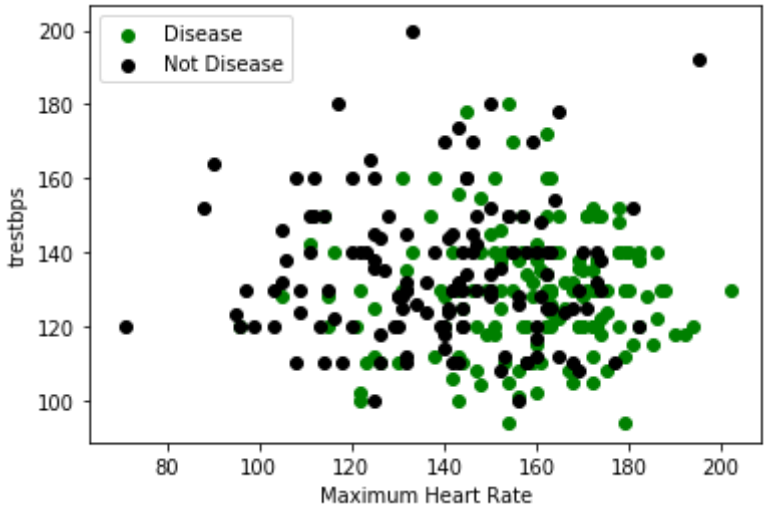
Out[26]: Text(0.5, 1.0, 'Age-Cholesterol Plot')



In [27]:
```
heart.plot(kind = 'scatter',x = 'age', y = 'thalach',alpha = 0.5, color
= 'red')
plt.xlabel('Age')
plt.ylabel('Cholesterol')
plt.title('Age-Cholesterol Plot')
```

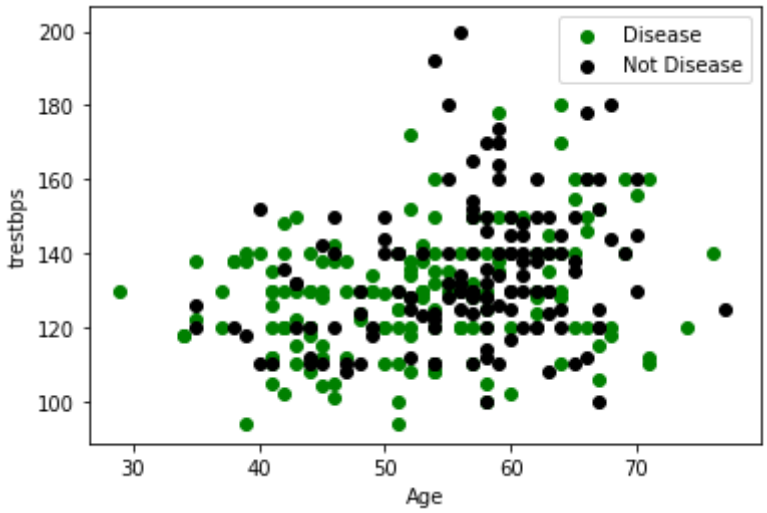
Out[27]: Text(0.5, 1.0, 'Age-Cholesterol Plot')

In [28]:
```
#lets see the relationship between maximum heart rate and resting bp
plt.scatter(x=heart.thalach[heart.target==1], y=heart.trestbps[(heart.ta
rget==1)], c="green")
plt.scatter(x=heart.thalach[heart.target==0], y=heart.trestbps[(heart.ta
rget==0)], c = 'black')
plt.legend(["Disease", "Not Disease"])
plt.xlabel("Maximum Heart Rate")
plt.ylabel("trestbps")
plt.show()
```



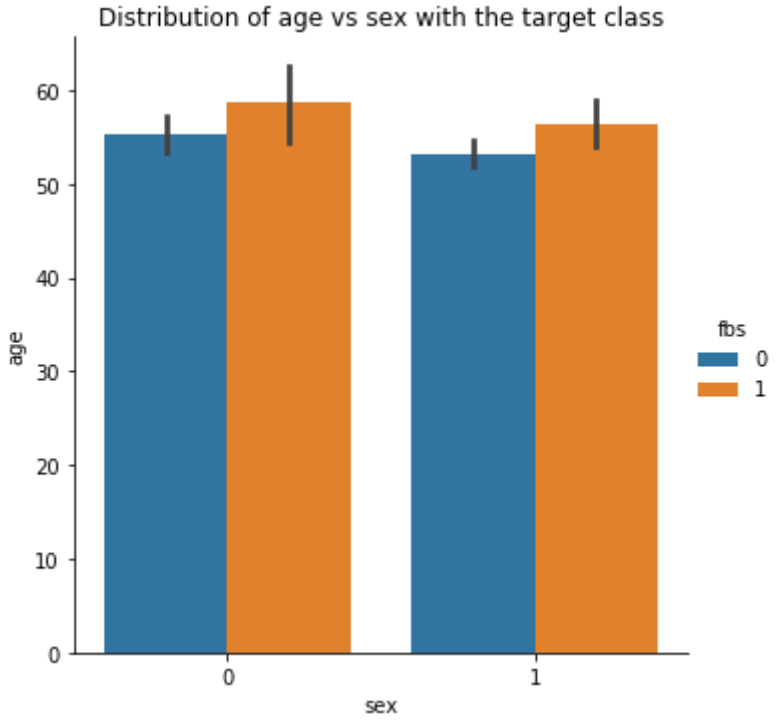
As the max heart rate increased, people started having heart disease.

In [29]:
```
plt.scatter(x=heart.age[heart.target==1], y=heart.trestbps[(heart.target
==1)], c="green")
plt.scatter(x=heart.age[heart.target==0], y=heart.trestbps[(heart.target
==0)], c = 'black')
plt.legend(["Disease", "Not Disease"])
plt.xlabel("Age")
plt.ylabel("trestbps")
plt.show()
```



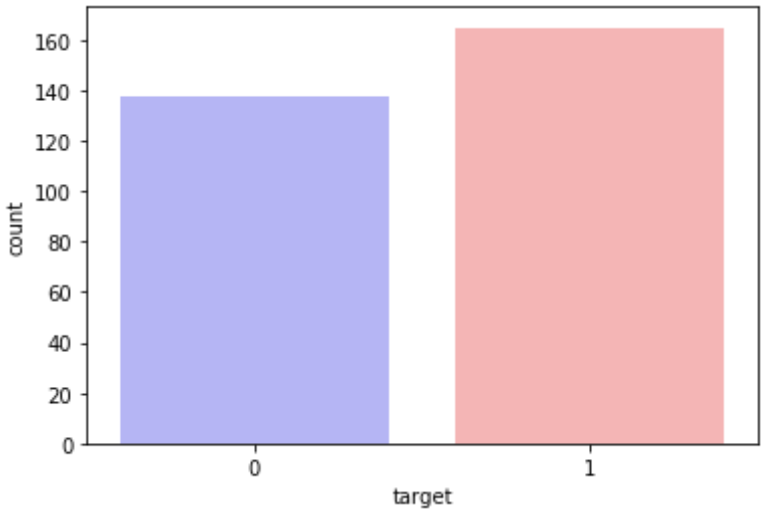
Barplot

In [30]:
```
# barplot of age vs sex with hue = target
sns.catplot(kind = 'bar', data = heart, y = 'age', x = 'sex', hue = 'fbs')
plt.title('Distribution of age vs sex with the target class')
plt.show()
#(1 = male; 0 = female)
```



For both males and females in the study, there were more diabetics in both groups than normal patients

In [31]:
```
sns.countplot(x="target", data=heart, palette="bwr")
plt.show()
```

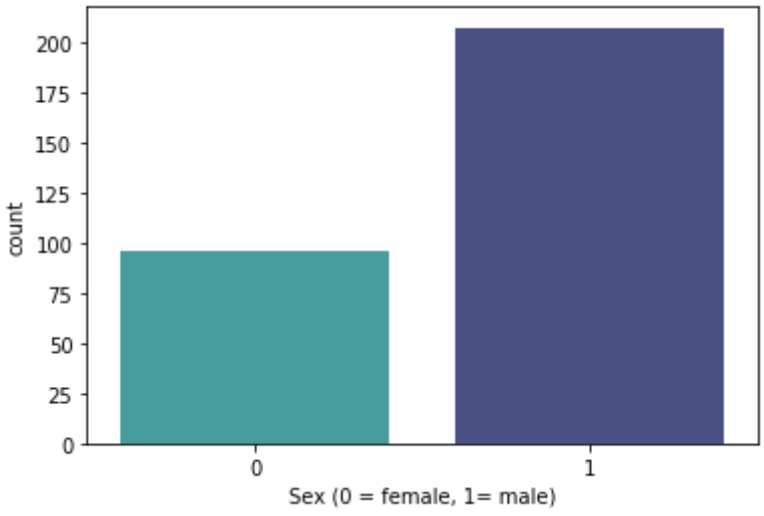


There were more individuals with heart disease than normal individuals

In [33]:
```
countNoDisease = len(heart[heart.target == 0])
countHaveDisease = len(heart[heart.target == 1])
print("Percentage of Patients Haven't Heart Disease: {:.2f}%".format((co
untNoDisease / (len(heart.target))*100)))
print("Percentage of Patients Have Heart Disease: {:.2f}%".format((count
HaveDisease / (len(heart.target))*100)))
```

```
Percentage of Patients Haven't Heart Disease: 45.54%
Percentage of Patients Have Heart Disease: 54.46%
```

In [34]:
```
sns.countplot(x='sex', data=heart, palette="mako_r")
plt.xlabel("Sex (0 = female, 1= male)")
plt.show()
```



In [35]:
```
countFemale = len(heart[heart.sex == 0])
countMale = len(heart[heart.sex == 1])
print("Percentage of Female Patients: {:.2f}%".format((countFemale / (le
n(heart.sex))*100)))
print("Percentage of Male Patients: {:.2f}%".format((countMale / (len(he
art.sex))*100)))
```

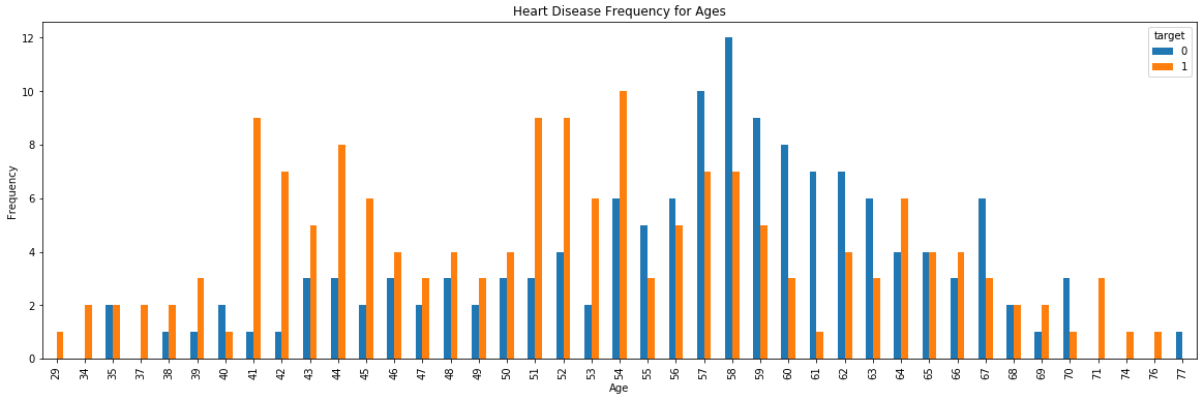
```
Percentage of Female Patients: 31.68%
Percentage of Male Patients: 68.32%
```

In [36]:
```
heart.groupby('target').mean()
```

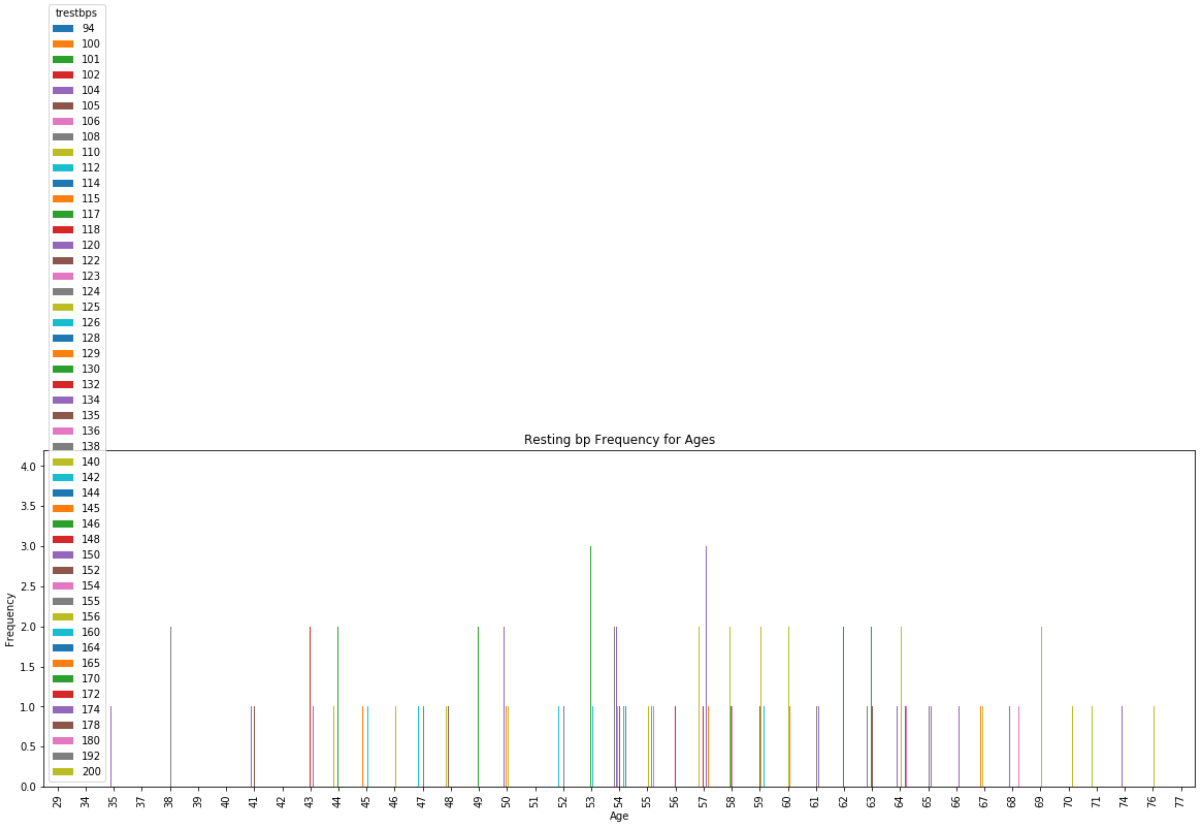
Out[36]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0

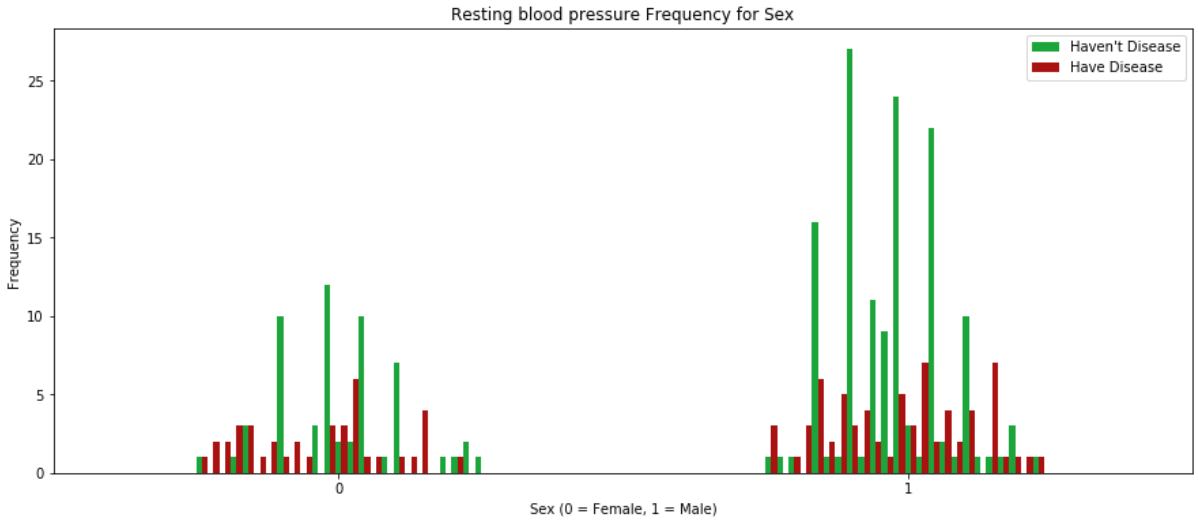
In [37]:
```
pd.crosstab(heart.age,heart.target).plot(kind="bar",figsize=(20,6))
plt.title('Heart Disease Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('heartDiseaseAndAges.png')
plt.show()
```



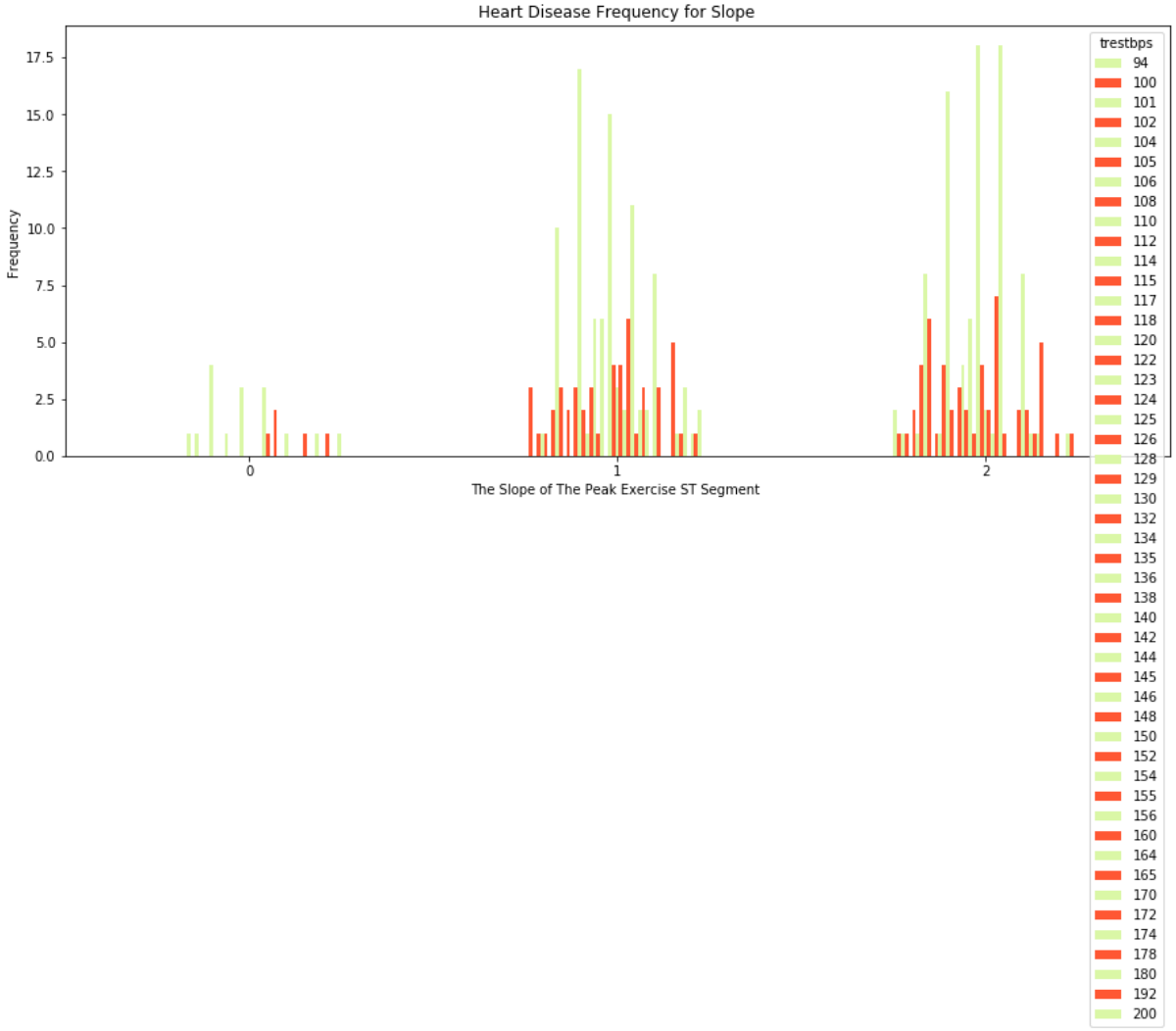
In [39]:
```
pd.crosstab(heart.age,heart.trestbps).plot(kind="bar",figsize=(20,6))
plt.title('Resting bp Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.savefig('RestingbpAndAges.png')
plt.show()
```



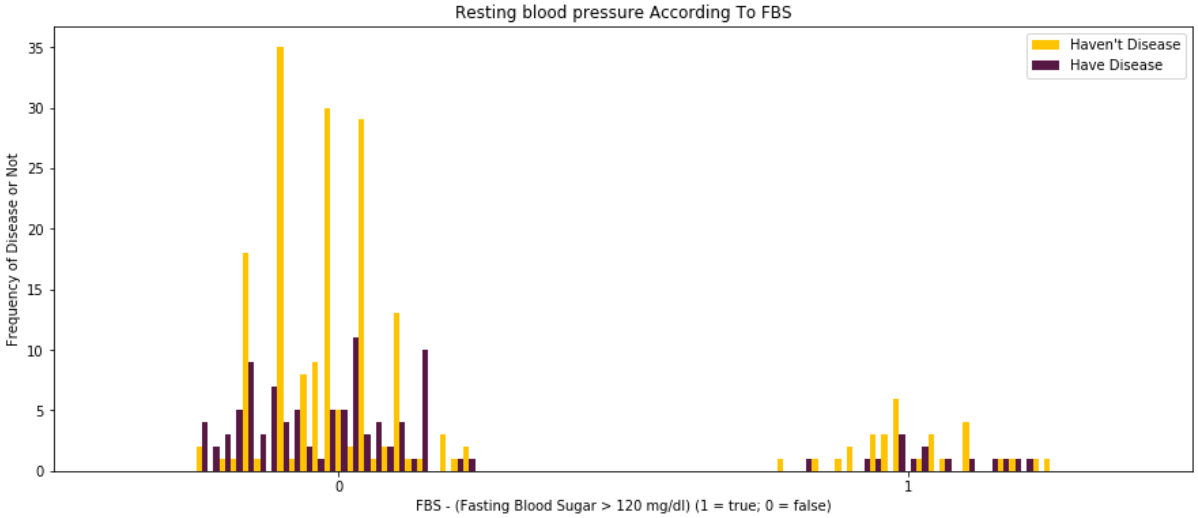
In [40]:
```
pd.crosstab(heart.sex,heart.trestbps).plot(kind="bar",figsize=(15,6),col
or=['#1CA53B','#AA1111' ])
plt.title('Resting blood pressure Frequency for Sex')
plt.xlabel('Sex (0 = Female, 1 = Male)')
plt.xticks(rotation=0)
plt.legend(["Haven't Disease", "Have Disease"])
plt.ylabel('Frequency')
plt.show()
```



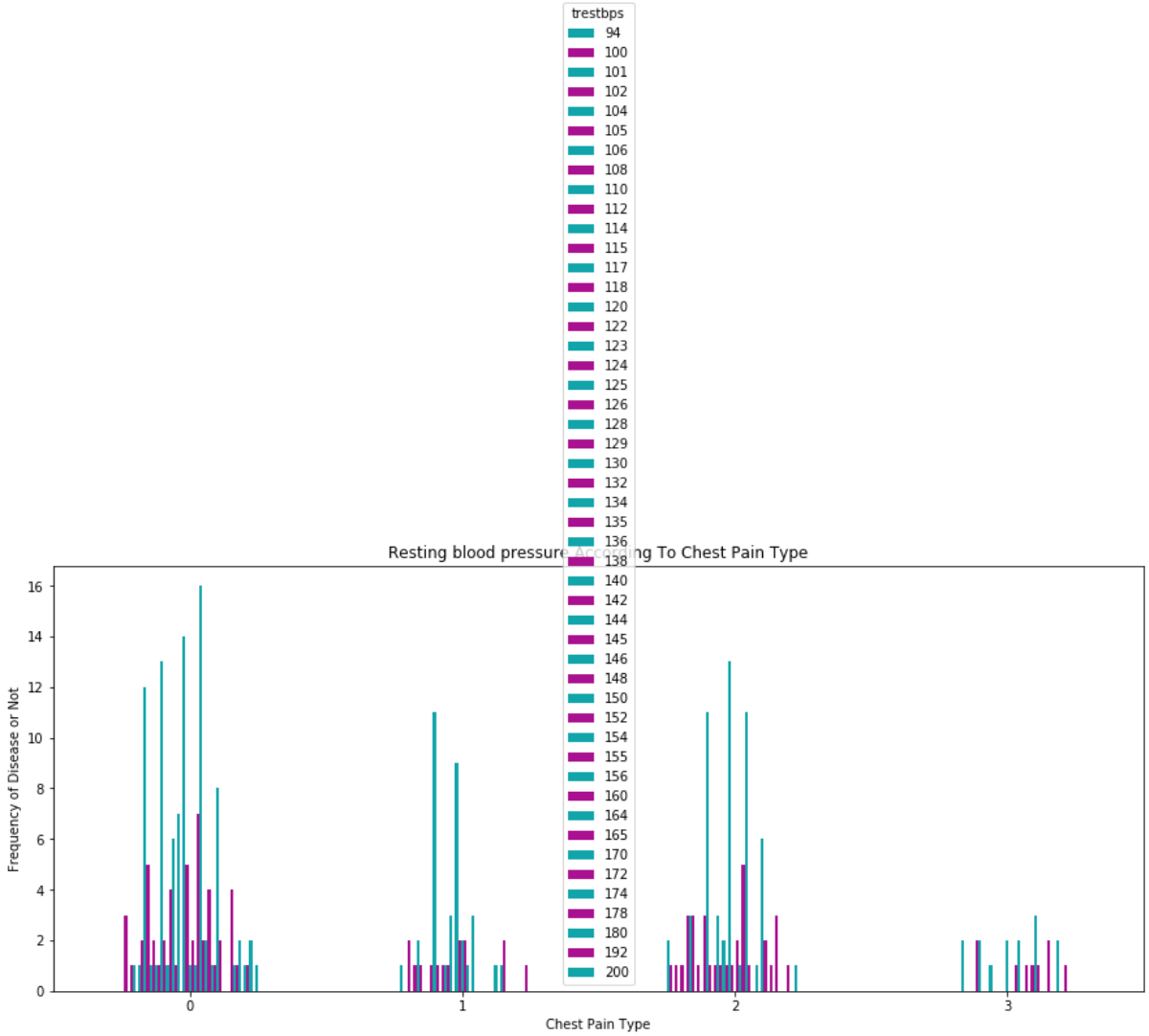
In [41]:
```
pd.crosstab(heart.slope,heart.trestbps).plot(kind="bar",figsize=(15,6),c
olor=['#DAF7A6','#FF5733' ])
plt.title('Heart Disease Frequency for Slope')
plt.xlabel('The Slope of The Peak Exercise ST Segment ')
plt.xticks(rotation = 0)
plt.ylabel('Frequency')
plt.show()
```



In [42]:
```
pd.crosstab(heart.fbs,heart.trestbps).plot(kind="bar",figsize=(15,6),col
or=['#FFC300','#581845' ])
plt.title('Resting blood pressure According To FBS')
plt.xlabel('FBS - (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = fals
e)')
plt.xticks(rotation = 0)
plt.legend(["Haven't Disease", "Have Disease"])
plt.ylabel('Frequency of Disease or Not')
plt.show()
```



In [43]:
```
pd.crosstab(heart.cp,heart.trestbps).plot(kind="bar",figsize=(15,6),colo
r=['#11A5AA','#AA1190' ])
plt.title('Resting blood pressure According To Chest Pain Type')
plt.xlabel('Chest Pain Type')
plt.xticks(rotation = 0)
plt.ylabel('Frequency of Disease or Not')
plt.show()
```



Feature engineering

In [45]:
```
heart['trest_band']=0
heart.loc[heart["trestbps"]<=130,'trest_band']=0
heart.loc[(heart["trestbps"]>130)&(heart["trestbps"]<=300),"trest_band"]
=1
```

In [46]:
```
heart.head()
```

Out[46]:

	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 [47]:
```
heart.drop(['trestbps', 'target'], axis=1, inplace=True)
```

In [48]:
```
heart
```

Out[48]:

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

303 rows × 13 columns

In [50]:
```
a = pd.get_dummies(heart['cp'], prefix = "cp")
b = pd.get_dummies(heart['thal'], prefix = "thal")
c = pd.get_dummies(heart['slope'], prefix = "slope")
```

In [51]:
```
frames = [heart, a, b, c]
heart = pd.concat(frames, axis = 1)
heart.head()
```

Out[51]:

	age	sex	cp	chol	fbs	restecg	thalach	exang	oldpeak	slope	...	cp_1	cp_2	cp_3	thal
0	63	1	3	233	1	0	150	0	2.3	0	...	0	0	1	
1	37	1	2	250	0	1	187	0	3.5	0	...	0	1	0	
2	41	0	1	204	0	0	172	0	1.4	2	...	1	0	0	
3	56	1	1	236	0	1	178	0	0.8	2	...	1	0	0	
4	57	0	0	354	0	1	163	1	0.6	2	...	0	0	0	

5 rows × 24 columns

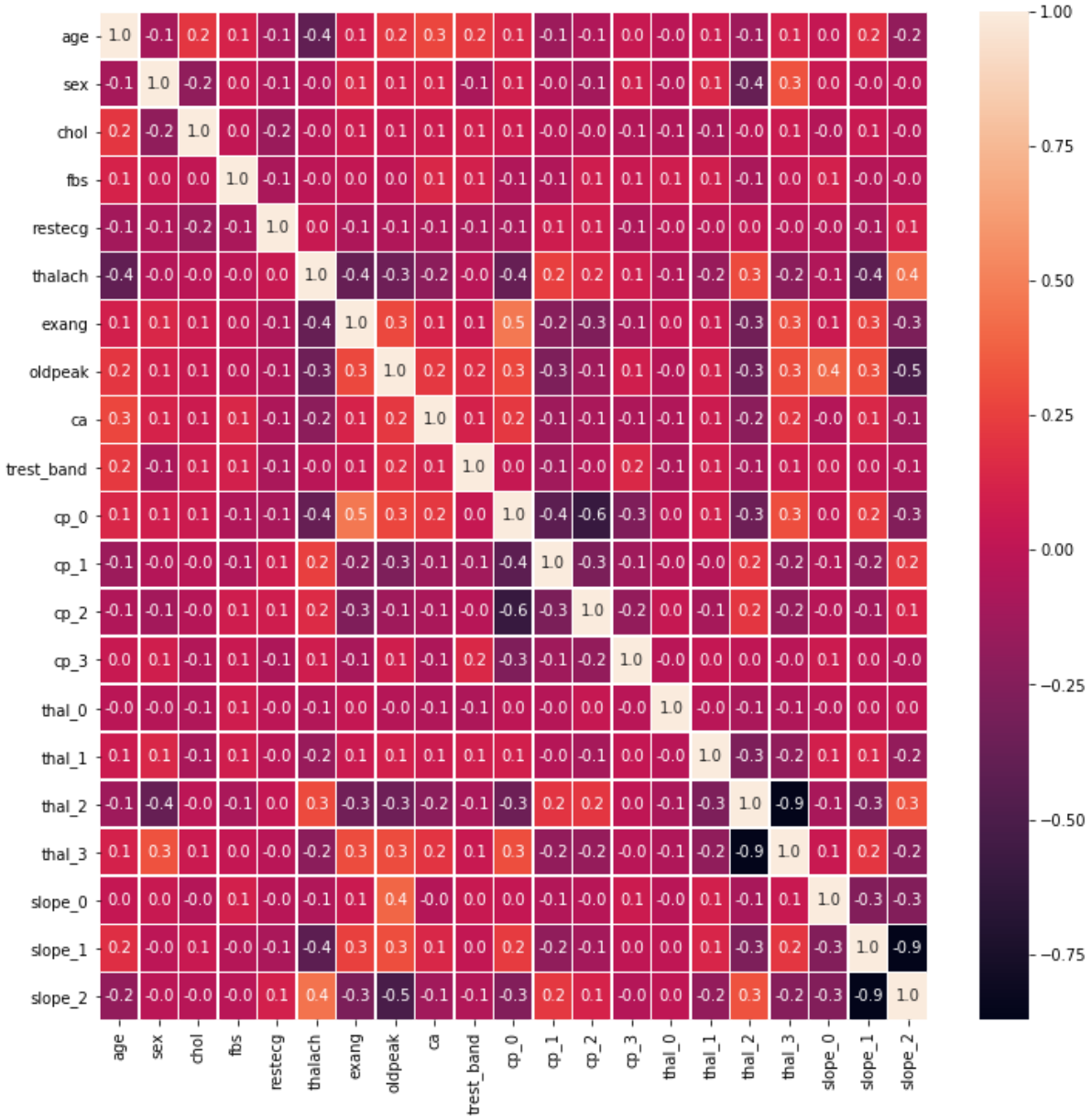
In [52]:
```
heart = heart.drop(columns = ['cp', 'thal', 'slope'])
heart.head()
```

Out[52]:

	age	sex	chol	fbs	restecg	thalach	exang	oldpeak	ca	trest_band	...	cp_1	cp_2	cp_3
0	63	1	233	1	0	150	0	2.3	0	1	...	0	0	1
1	37	1	250	0	1	187	0	3.5	0	0	...	0	1	0
2	41	0	204	0	0	172	0	1.4	0	0	...	1	0	0
3	56	1	236	0	1	178	0	0.8	0	0	...	1	0	0
4	57	0	354	0	1	163	1	0.6	0	0	...	0	0	0

5 rows × 21 columns

In [53]:
```
f,ax = plt.subplots(figsize=(12,12))
sns.heatmap(heart.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```



In [54]:
```
from sklearn import linear_model
from sklearn.feature_selection import RFE
```

In [55]:
```
X = heart.iloc[:,:-1].values
```

In [56]:
```
X
```

Out[56]:
```
array([[ 63.,   1., 233., ...,   0.,   1.,   0.],
       [ 37.,   1., 250., ...,   0.,   1.,   0.],
       [ 41.,   0., 204., ...,   0.,   0.,   0.],
       ...,
       [ 68.,   1., 193., ...,   1.,   0.,   1.],
       [ 57.,   1., 131., ...,   1.,   0.,   1.],
       [ 57.,   0., 236., ...,   0.,   0.,   1.]])
```

In [57]:
```
X = heart.iloc[:,:-1].values
Y = heart.iloc[:,-1].values
lr = linear_model.LinearRegression()
model_h = lr.fit(X,Y)
```

In [58]:
```
print('intercept: \n', model_h.intercept_)
print('coefficients: \n', model_h.coef_)
```

```
intercept:
 1.0000000000000082
coefficients:
 [-1.18220858e-16 -1.66046657e-16  7.67579066e-18  2.00487987e-16
 -4.50105056e-17 -2.33622336e-17  1.76250145e-16 -1.52069496e-16
 -2.22217300e-17  1.21037538e-16  6.60108616e-17  1.56781457e-16
 -1.56645948e-17  5.96141675e-16  5.99698023e-17 -3.46428888e-16
  2.31841058e-16 -2.13954259e-16 -1.00000000e+00 -1.00000000e+00]
```

In [59]:
```
print("R^2 = " + str(model_h.score(X,Y)))
```

```
R^2 = 1.0
```

In [60]:
```
rfe = RFE(lr, n_features_to_select=3)
```

In [61]:
```
model_h = rfe.fit(X,Y)
```

In [62]:
```
print(rfe.support_)
print(rfe.ranking_)
```

```
[False False False  True False False False False False False False Fals
e
 False False False False False False  True  True]
[ 2 13 18  1 12 16 11  5 15 10 17  4  8  3  9  7  6 14  1  1]
```

In [63]:
```
print('Features sorted by their rank:')
print(sorted(zip(map(lambda x: round(x, 4), rfe.ranking_), heart.columns
)))
```

```
Features sorted by their rank:
[(1, 'fbs'), (1, 'slope_0'), (1, 'slope_1'), (2, 'age'), (3, 'cp_3'),
(4, 'cp_1'), (5, 'oldpeak'), (6, 'thal_2'), (7, 'thal_1'), (8, 'cp_2'),
(9, 'thal_0'), (10, 'trest_band'), (11, 'exang'), (12, 'restecg'), (13,
'sex'), (14, 'thal_3'), (15, 'ca'), (16, 'thalach'), (17, 'cp_0'), (18,
'chol')]
```

In [64]:
```
feature_col = ['fbs', 'slope_0', 'slope_1']
A = heart[feature_col]
B = heart['trest_band']
```

In [65]:
```
from sklearn.model_selection import train_test_split
ATrain, ATest, BTrain, BTest = train_test_split(A, B, test_size=0.3, ran
dom_state=0)
```

In [66]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeClassifier as DT
classifier = DT(criterion='entropy', random_state=0)
classifier.fit(ATrain,BTrain)
BPred = classifier.predict(ATest)
mse = mean_squared_error(BTest,BPred)
r = r2_score(BTest,BPred)
mae = mean_absolute_error(BTest,BPred)
accuracy = accuracy_score(BTest,BPred)
print("Decision Tree Classifier :")
print("Mean Squared Error:",mse)
print("R score:",r)
print("Mean Absolute Error:",mae)
print("Accuracy = ", accuracy)
```

```
Decision Tree Classifier :
Mean Squared Error: 0.43956043956043955
R score: -0.8073485600794441
Mean Absolute Error: 0.43956043956043955
Accuracy =  0.5604395604395604
```

In [67]:
```
from sklearn.linear_model import Perceptron
classifier = Perceptron(tol=1e-3, random_state=0)
classifier.fit(ATrain,BTrain)
BPred = classifier.predict(ATest)
mse = mean_squared_error(BTest,BPred)
r = r2_score(BTest,BPred)
mae = mean_absolute_error(BTest,BPred)
accuracy = accuracy_score(BTest,BPred)
print("Perceptron :")
print("Accuracy = ", accuracy)
print("Mean Squared Error:",mse)
print("R score:",r)
print("Mean Absolute Error:",mae)
```

```
Perceptron :
Accuracy =  0.5714285714285714
Mean Squared Error: 0.42857142857142855
R score: -0.7621648460774582
Mean Absolute Error: 0.42857142857142855
```

In [68]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski'
)
classifier.fit(ATrain,BTrain)
BPred = classifier.predict(ATest)
mse = mean_squared_error(BTest,BPred)
r = r2_score(BTest,BPred)
mae = mean_absolute_error(BTest,BPred)
accuracy = accuracy_score(BTest,BPred)
print("K Nearest Neighbors :")
print("Accuracy = ", accuracy)
print("Mean Squared Error:",mse)
print("R score:",r)
print("Mean Absolute Error:",mae)
```

```
K Nearest Neighbors :
Accuracy =  0.46153846153846156
Mean Squared Error: 0.5384615384615384
R score: -1.214001986097319
Mean Absolute Error: 0.5384615384615384
```

In [69]:
```
from sklearn.svm import SVC
classifier = SVC(kernel='linear',random_state=0)
from sklearn.svm import SVC
classifier = SVC(kernel='linear',random_state=0)
classifier.fit(ATrain,BTrain)
BPred = classifier.predict(ATest)
mse = mean_squared_error(BTest,BPred)
r = r2_score(BTest,BPred)
mae = mean_absolute_error(BTest,BPred)
accuracy = accuracy_score(BTest,BPred)
print("Support Vector Machine :")
print("Accuracy = ", accuracy)
print("Mean Squared Error:",mse)
print("R score:",r)
print("Mean Absolute Error:",mae)
print("Support Vector Machine :")
print("Accuracy = ", accuracy)
print("Mean Squared Error:",mse)
print("R score:",r)
print("Mean Absolute Error:",mae)
```

```
Support Vector Machine :
Accuracy =  0.5714285714285714
Mean Squared Error: 0.42857142857142855
R score: -0.7621648460774582
Mean Absolute Error: 0.42857142857142855
Support Vector Machine :
Accuracy =  0.5714285714285714
Mean Squared Error: 0.42857142857142855
R score: -0.7621648460774582
Mean Absolute Error: 0.42857142857142855
```

In [70]:
```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(ATrain,BTrain)
BPred = classifier.predict(ATest)
mse = mean_squared_error(BTest,BPred)
r = r2_score(BTest,BPred)
mae = mean_absolute_error(BTest,BPred)
accuracy = accuracy_score(BTest,BPred)
print("Gaussian Naive Bayes :")
print("Accuracy = ", accuracy)
print("Mean Squared Error:",mse)
print("R score:",r)
print("Mean Absolute Error:",mae)
```

```
Gaussian Naive Bayes :
Accuracy =  0.5824175824175825
Mean Squared Error: 0.4175824175824176
R score: -0.716981132075472
Mean Absolute Error: 0.4175824175824176
```

In [71]:
```
from sklearn.ensemble import RandomForestClassifier as RF
classifier = RF(n_estimators=10, criterion='entropy', random_state=0)
classifier.fit(ATrain,BTrain)
BPred = classifier.predict(ATest)
mse = mean_squared_error(BTest,BPred)
r = r2_score(BTest,BPred)
mae = mean_absolute_error(BTest,BPred)
accuracy = accuracy_score(BTest,BPred)
print("Random Forest Classifier :")
print("Accuracy = ", accuracy)
print("Mean Squared Error:",mse)
print("R score:",r)
print("Mean Absolute Error:",mae)
```

```
Random Forest Classifier :
Accuracy =  0.5714285714285714
Mean Squared Error: 0.42857142857142855
R score: -0.7621648460774582
Mean Absolute Error: 0.42857142857142855
```

In [72]:
```
from sklearn import svm # support vector Machine
from sklearn import metrics # accuracy measure
model=svm.SVC(kernel="linear",C=0.1,gamma=0.1)
model.fit(A,B)
prediction1=model.predict(A)
print("Accuracy for linear SVM is",metrics.accuracy_score(prediction1,B
))
```

```
Accuracy for linear SVM is 0.5643564356435643
```

In [73]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

In [74]:
```
logmodel = LogisticRegression(solver="liblinear")
logmodel.fit(A,B)
```

Out[74]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, v
erbose=0,
                   warm_start=False)
```

In [75]:
```
predictions = logmodel.predict(A)
```

In [76]:
```
print(classification_report(B,predictions))
```

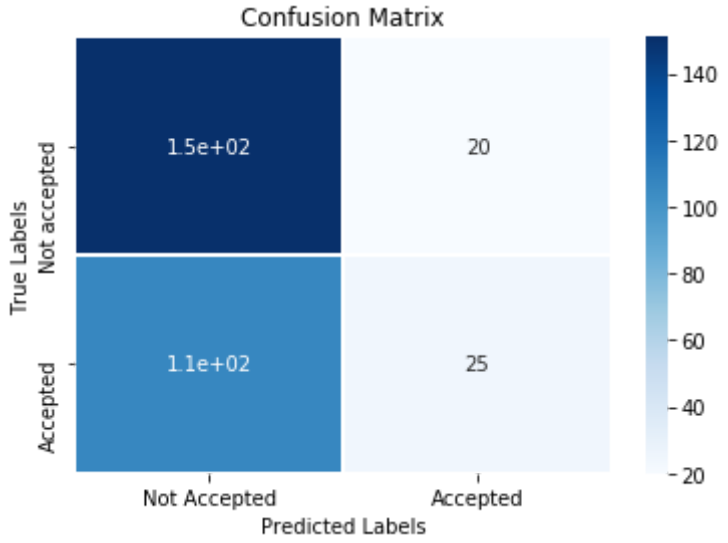
```
              precision    recall  f1-score   support

           0       0.59      0.88      0.70       171
           1       0.56      0.19      0.28       132

    accuracy                           0.58       303
   macro avg       0.57      0.54      0.49       303
weighted avg       0.57      0.58      0.52       303
```

In [77]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(B, predictions)
print(cm)
ax = plt.subplot()
sns.heatmap(cm, annot=True, ax = ax, linewidths=1.2, cmap="Blues");
ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['Not Accepted', 'Accepted']);ax.yaxis.set_tickl
abels(["Not accepted", 'Accepted']);
```

```
[[151  20]
 [107  25]]
```



In [ ]: