Heart Disease

September 4, 2021

[22]:
```
import pandas as pd
GP = pd.read_csv('data.csv')
GP.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
```

[23]:
```
GP.isnull().sum()
```

[23]:
```
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
```

[ ]:

[24]:
```
print('First DUPLICATE count:\t{}'.format(GP.duplicated().sum()))
GP.drop_duplicates(inplace = True)  #  drop duplitcates
print('Second DUPLICATE count:\t{}'.format(GP.duplicated().sum()))
```

```
First DUPLICATE count:  1
Second DUPLICATE count: 0
```

[25]:
```
GP.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
dtypes: float64(1), int64(13)
memory usage: 35.4 KB
```

[26]:
```
GP.describe()
```

[26]:

	age	sex	cp	trestbps	chol	fbs	\
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	

```
50%     55.50000    1.000000    1.000000  130.000000  240.500000    0.000000
75%     61.00000    1.000000    2.000000  140.000000  274.750000    0.000000
max     77.00000    1.000000    3.000000  200.000000  564.000000    1.000000

          restecg     thalach       exang     oldpeak       slope          ca  \
count  302.000000  302.000000  302.000000  302.000000  302.000000  302.000000
mean     0.526490  149.569536    0.327815    1.043046    1.397351    0.718543
std      0.526027   22.903527    0.470196    1.161452    0.616274    1.006748
min      0.000000   71.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000  133.250000    0.000000    0.000000    1.000000    0.000000
50%      1.000000  152.500000    0.000000    0.800000    1.000000    0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000    1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000    4.000000

             thal      target
count  302.000000  302.000000
mean     2.314570    0.543046
std      0.613026    0.498970
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

[27]:
```
for i in GP.columns:
    print(i,len(GP[i].unique()))
```

```
age 41
sex 2
cp 4
trestbps 49
chol 152
fbs 2
restecg 3
thalach 91
exang 2
oldpeak 40
slope 3
ca 5
thal 4
target 2
```

[28]:
```
GP = GP[GP.ca != 4]## removing number of major vessels more than 3
GP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 298 entries, 0 to 302
Data columns (total 14 columns):
```

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       298 non-null    int64
 1   sex       298 non-null    int64
 2   cp        298 non-null    int64
 3   trestbps  298 non-null    int64
 4   chol      298 non-null    int64
 5   fbs       298 non-null    int64
 6   restecg   298 non-null    int64
 7   thalach   298 non-null    int64
 8   exang     298 non-null    int64
 9   oldpeak   298 non-null    float64
 10  slope     298 non-null    int64
 11  ca        298 non-null    int64
 12  thal      298 non-null    int64
 13  target    298 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 34.9 KB
```

[29]:
```
GP=GP[GP.thal != 0] ## removing thalasemia values with zero;
GP.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 296 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       296 non-null    int64
 1   sex       296 non-null    int64
 2   cp        296 non-null    int64
 3   trestbps  296 non-null    int64
 4   chol      296 non-null    int64
 5   fbs       296 non-null    int64
 6   restecg   296 non-null    int64
 7   thalach   296 non-null    int64
 8   exang     296 non-null    int64
 9   oldpeak   296 non-null    float64
 10  slope     296 non-null    int64
 11  ca        296 non-null    int64
 12  thal      296 non-null    int64
 13  target    296 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 34.7 KB
```

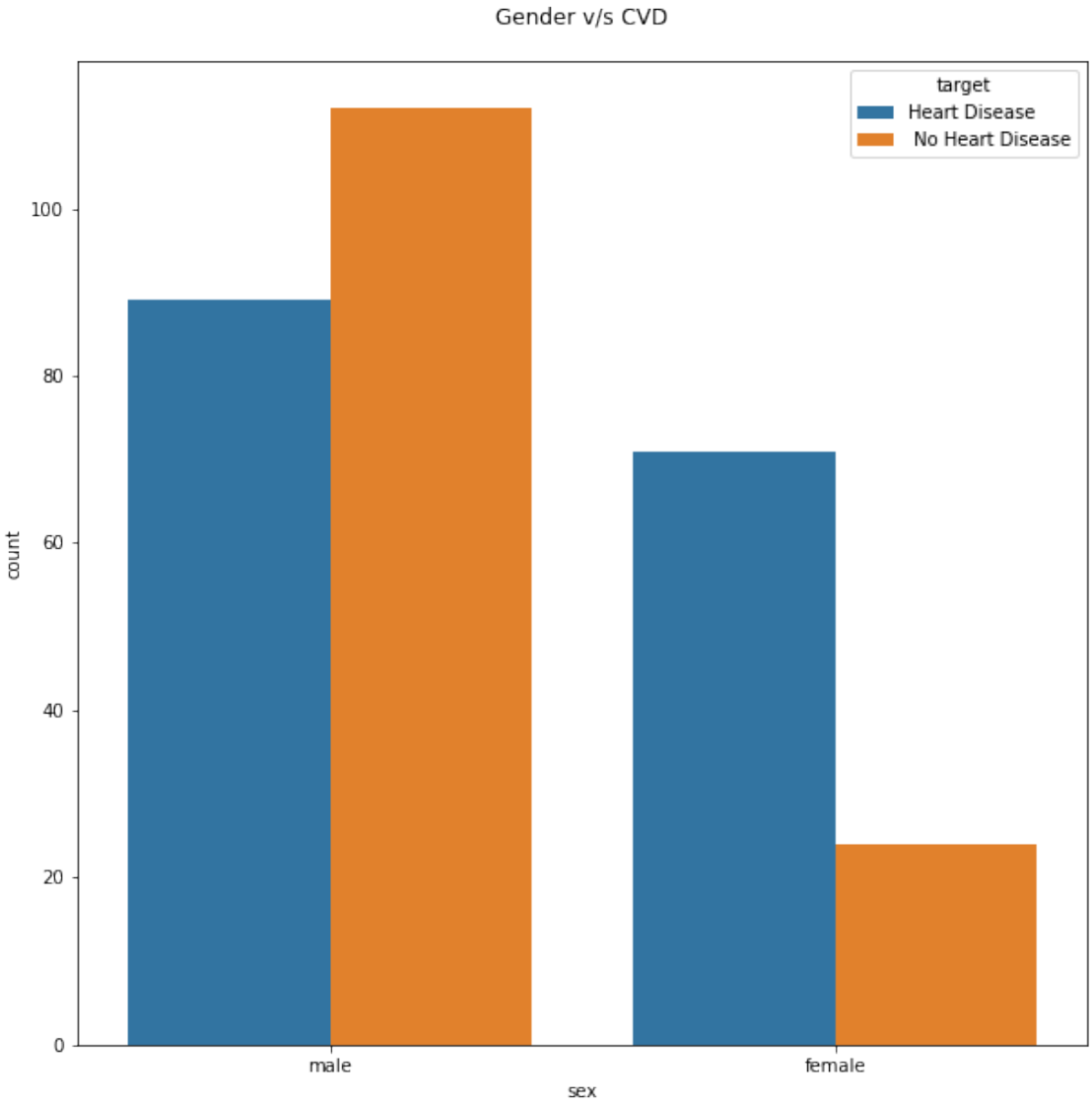
[39]:
```
GP3=GP.copy()
def gender(sex):
    if sex == 0:
```

```
        return 'female'
    else:
        return 'male'
GP3['sex'] = GP3['sex'].apply(gender)

def disease(t):
    if t == 0:
        return ' No Heart Disease'
    else:
        return 'Heart Disease'
GP3['target'] = GP3['target'].apply(disease)
```

[40]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[41]:
```
plt.figure(figsize=(10,10))
sns.countplot( x='sex',hue='target', data= GP3)
plt.title('Gender v/s CVD\n')
```

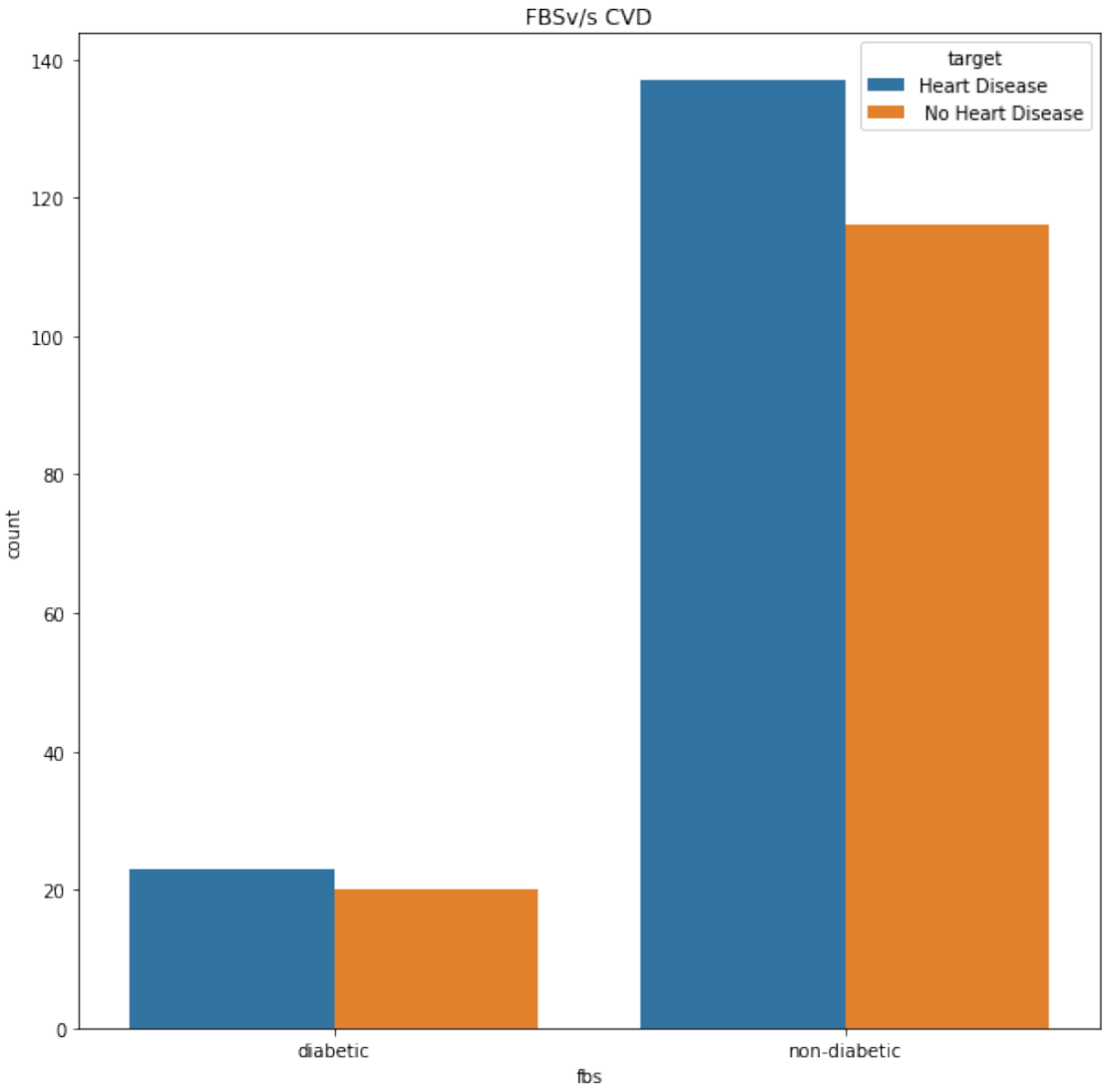
[41]:
```
Text(0.5, 1.0, 'Gender v/s CVD\n')
```



[42]:
```
def diabetes(fbs):
    if fbs == 0:
        return 'non-diabetic'
    else:
        return 'diabetic'
GP3['fbs'] = GP3['fbs'].apply(diabetes)
```

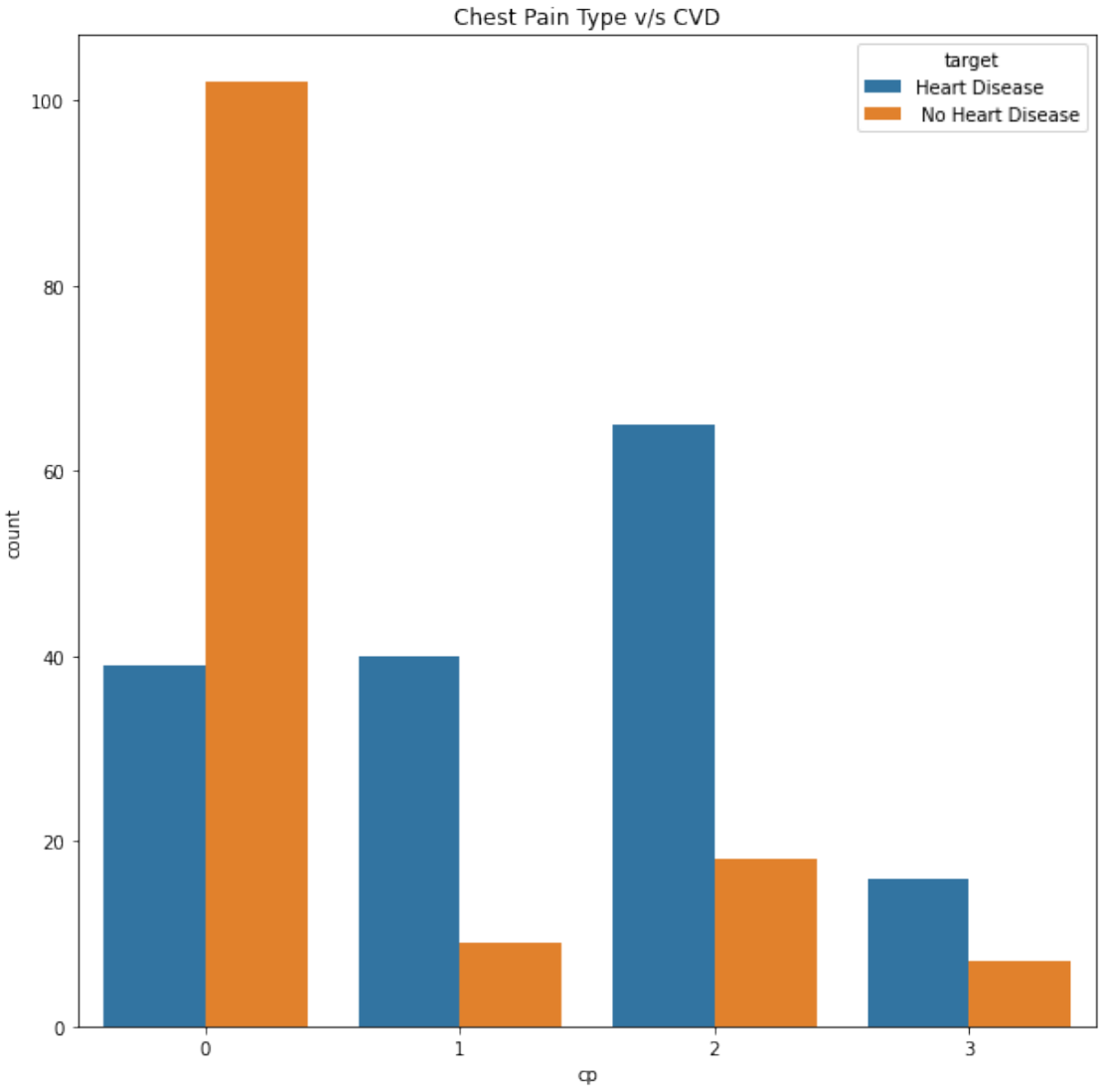
[43]:
```
plt.figure(figsize=(10,10))
sns.countplot(data= GP3, x='fbs',hue='target')
plt.title('FBSv/s CVD')
```

[43]:
```
Text(0.5, 1.0, 'FBSv/s CVD')
```



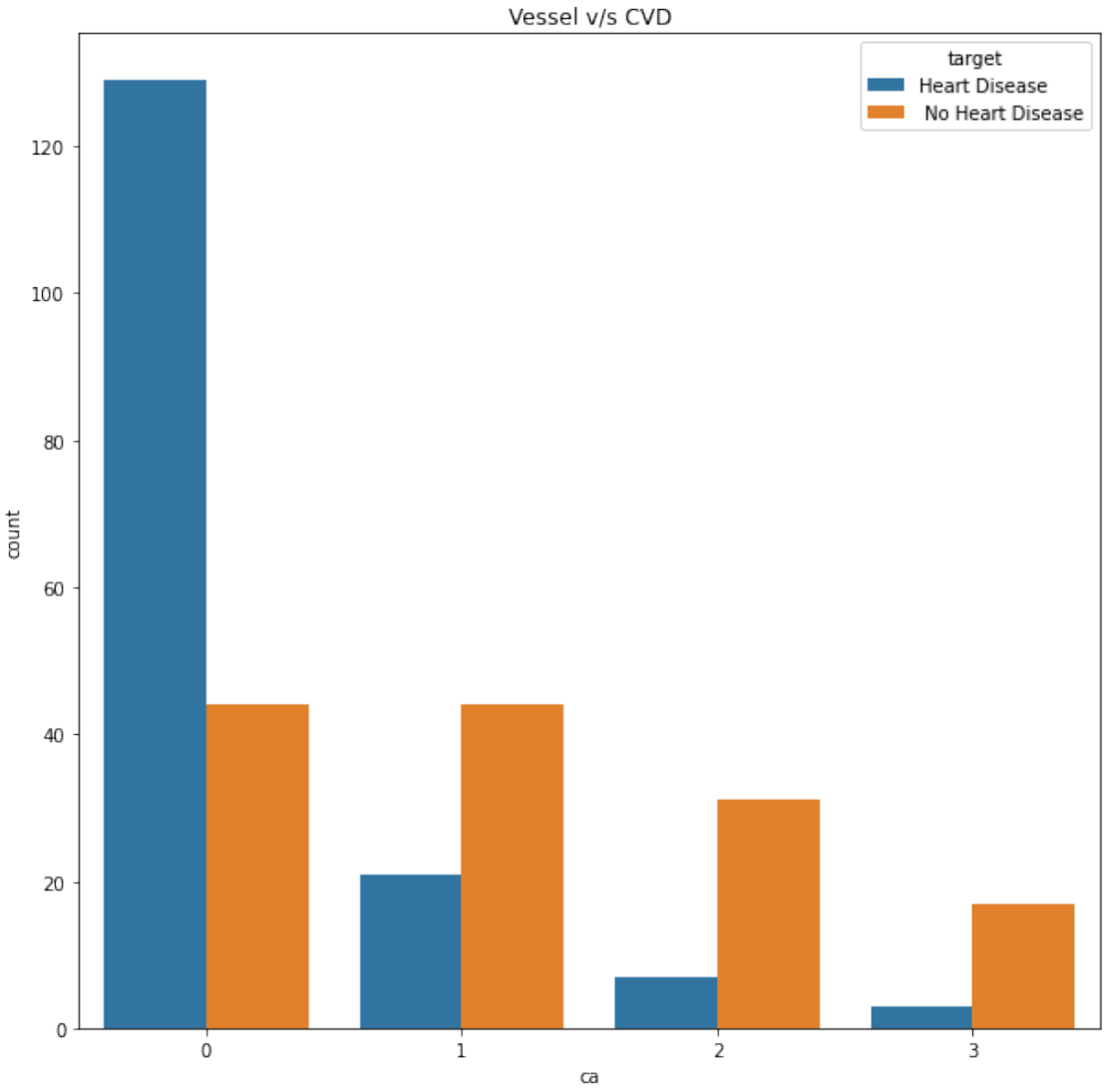
[44]:
```
plt.figure(figsize=(10,10))
sns.countplot(data= GP3, x='cp',hue='target')
plt.title('Chest Pain Type v/s CVD')
```

[44]: Text(0.5, 1.0, 'Chest Pain Type v/s CVD')



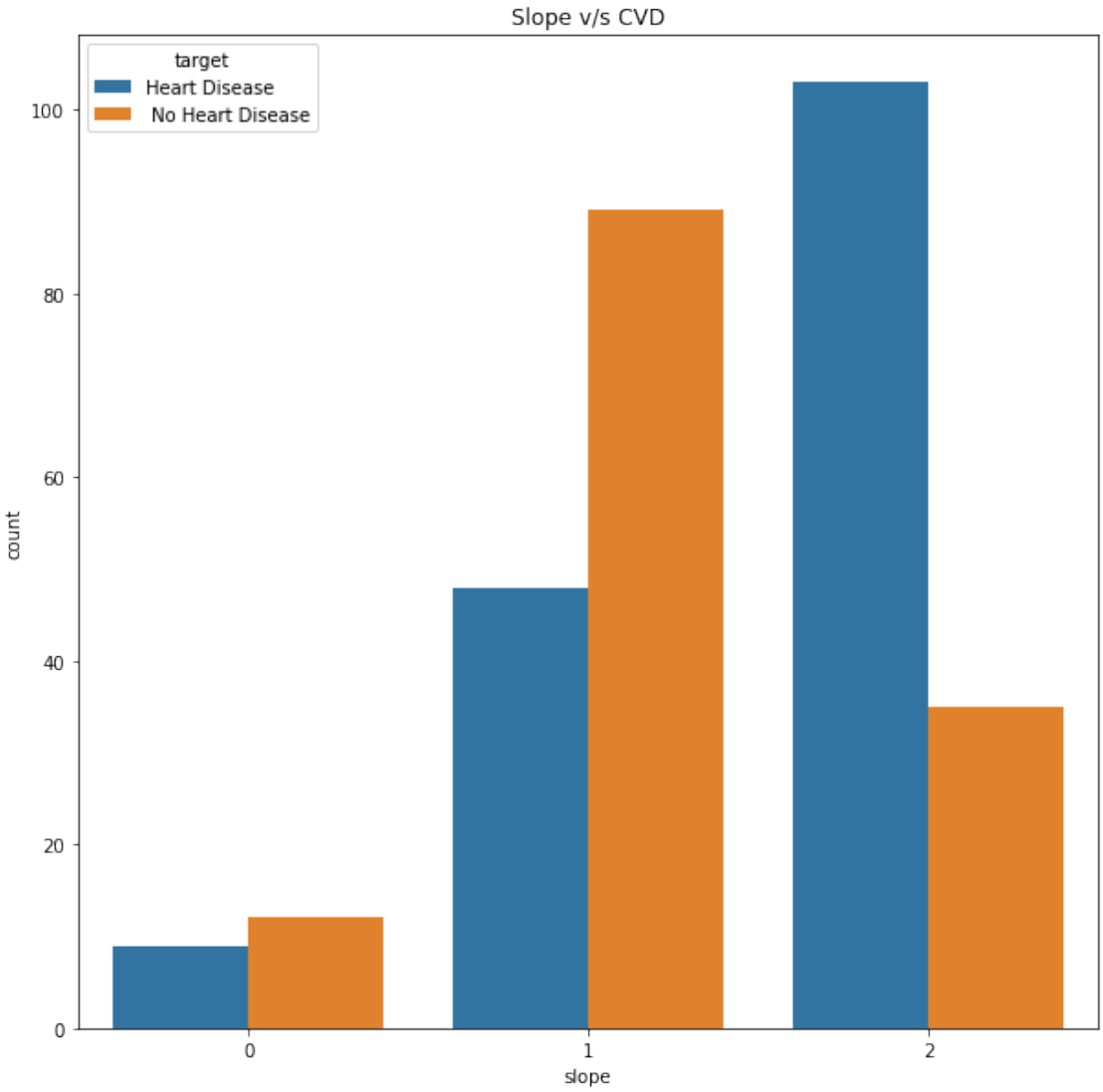
[45]:
```
plt.figure(figsize=(10,10))
sns.countplot(data= GP3, x='ca',hue='target')
plt.title('Vessel v/s CVD')
```

[45]: Text(0.5, 1.0, 'Vessel v/s CVD')



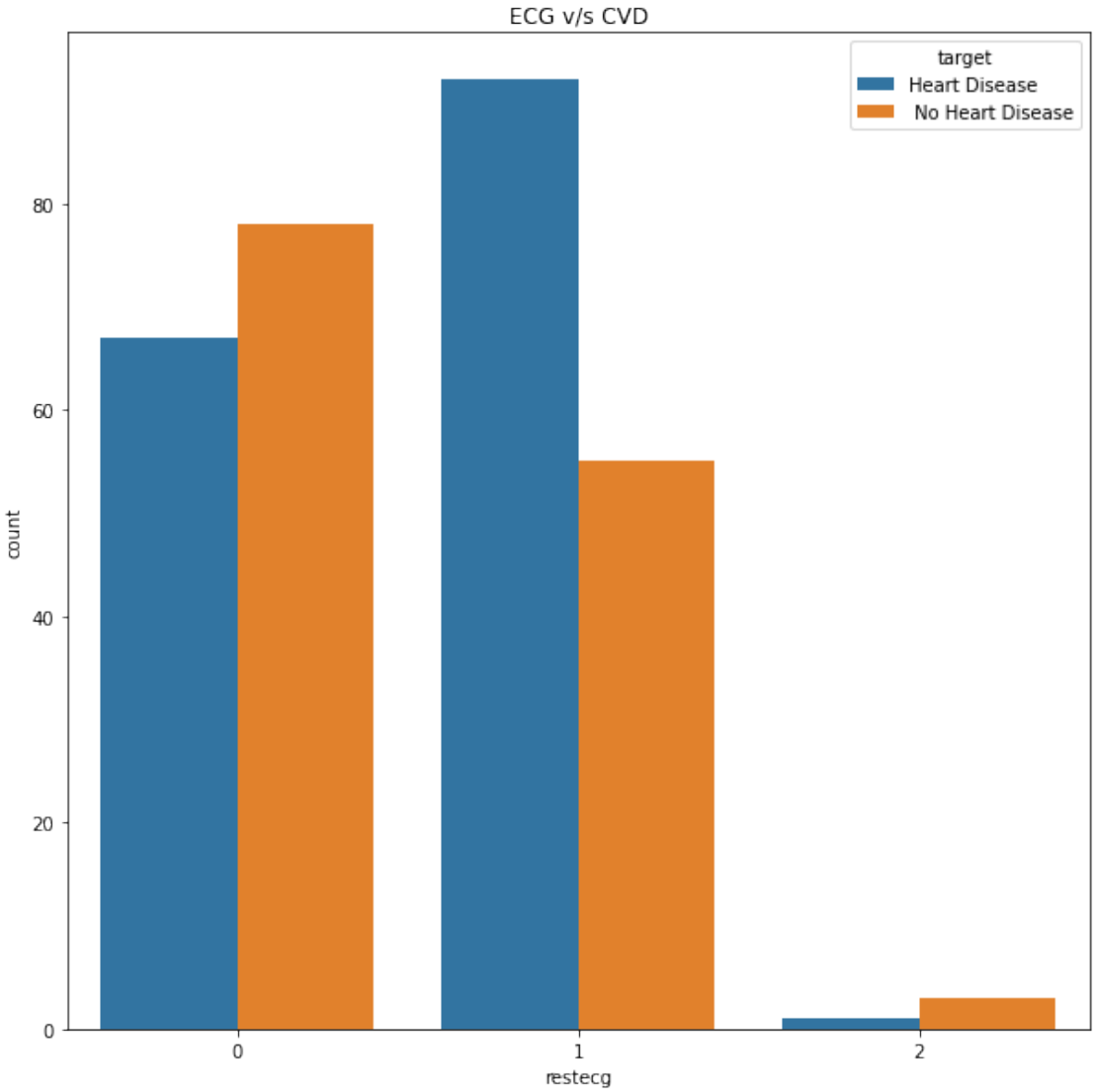
[46]:
```
plt.figure(figsize=(10,10))
sns.countplot(data= GP3, x='slope',hue='target')
plt.title('Slope v/s CVD')
```

[46]: Text(0.5, 1.0, 'Slope v/s CVD')



[47]:
```
plt.figure(figsize=(10,10))
sns.countplot(data= GP3, x='restecg',hue='target')
plt.title('ECG v/s CVD')
```

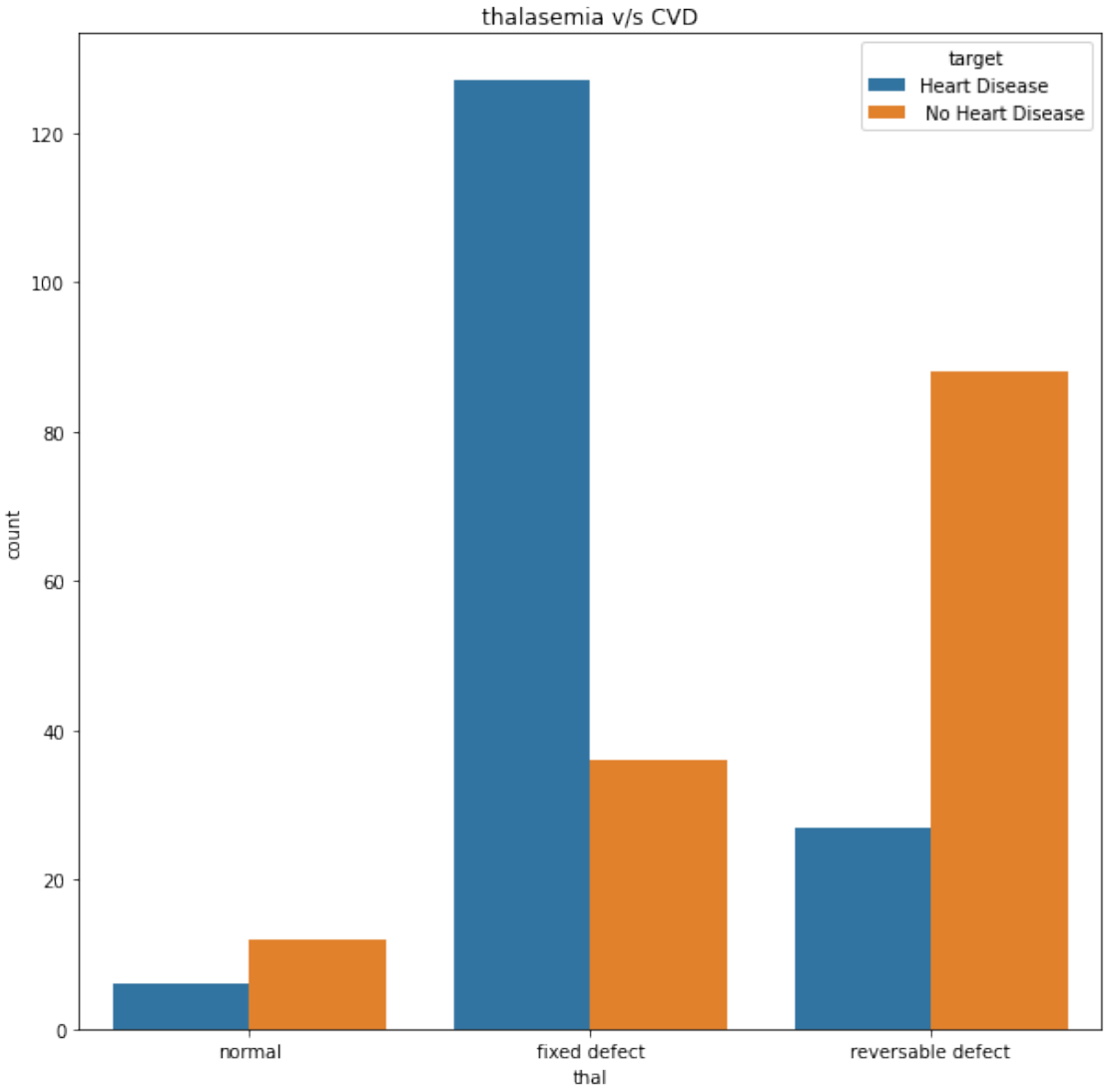
[47]: Text(0.5, 1.0, 'ECG v/s CVD')



[48]:
```
def thalasemia(thal):
    if thal == 1:
        return 'normal'
    elif thal == 2:
        return 'fixed defect'
    else:
        return 'reversable defect'

GP3['thal'] = GP3['thal'].apply(thalasemia)
```

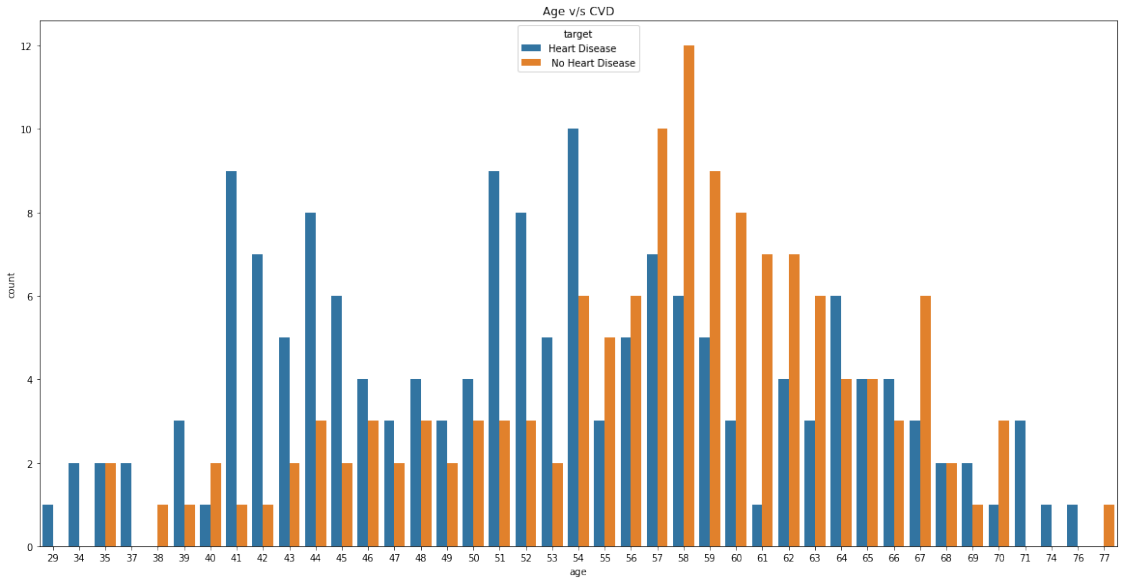
[49]:
```
plt.figure(figsize=(10,10))
sns.countplot(data= GP3, x='thal',hue='target')
plt.title('thalasemia v/s CVD')
```

[49]: Text(0.5, 1.0, 'thalasemia v/s CVD')



[56]:
```
plt.figure(figsize=(20,10))
sns.countplot(data= GP3, x='age',hue='target')
plt.title('Age v/s CVD')
```

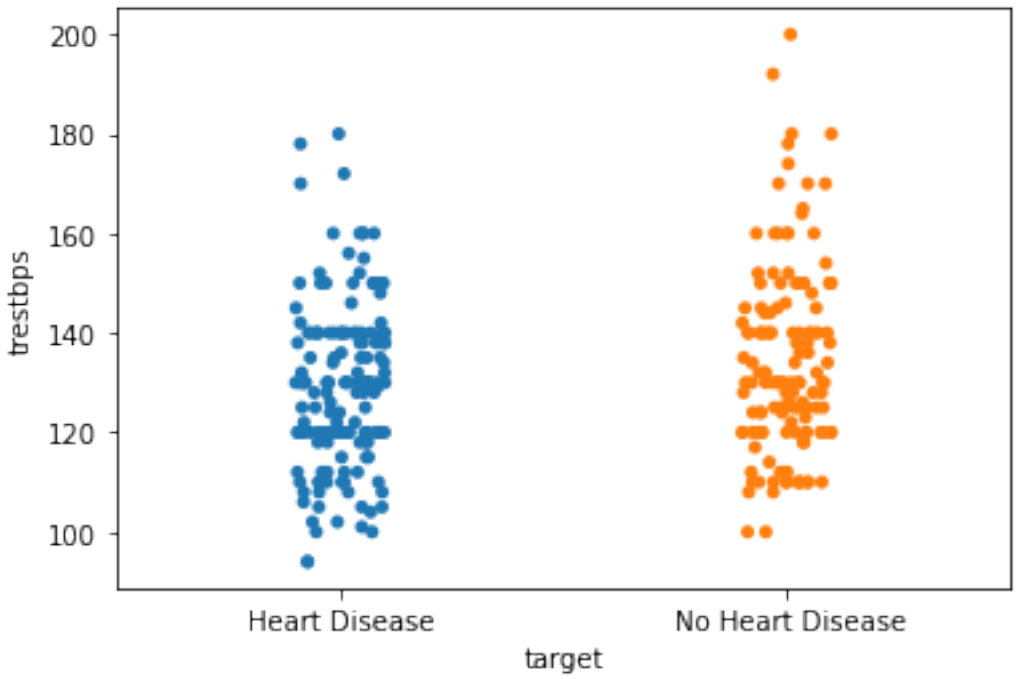
[56]: Text(0.5, 1.0, 'Age v/s CVD')



[55]:
```
sns.catplot(data=GP3, x='sex', y='age',  hue='target')
```

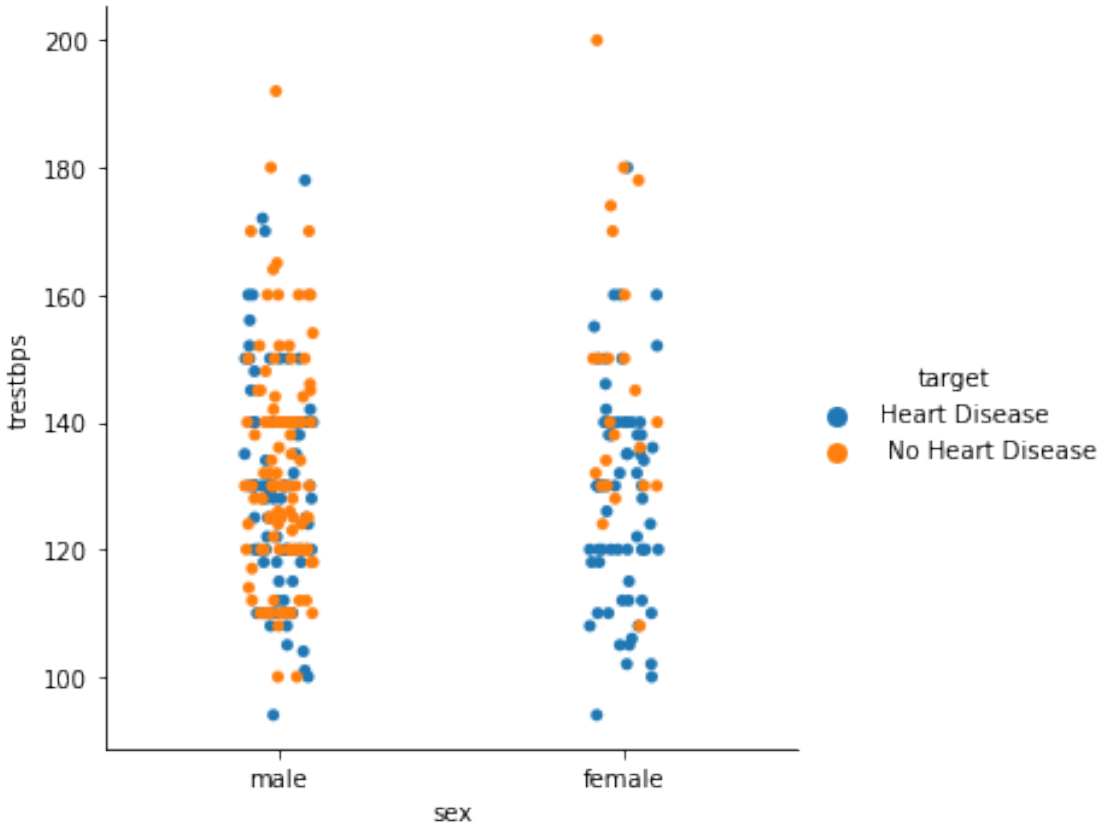
[55]: <seaborn.axisgrid.FacetGrid at 0x7f8464f15850>

[87]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="trestbps", data=GP3)
plt.show()
```



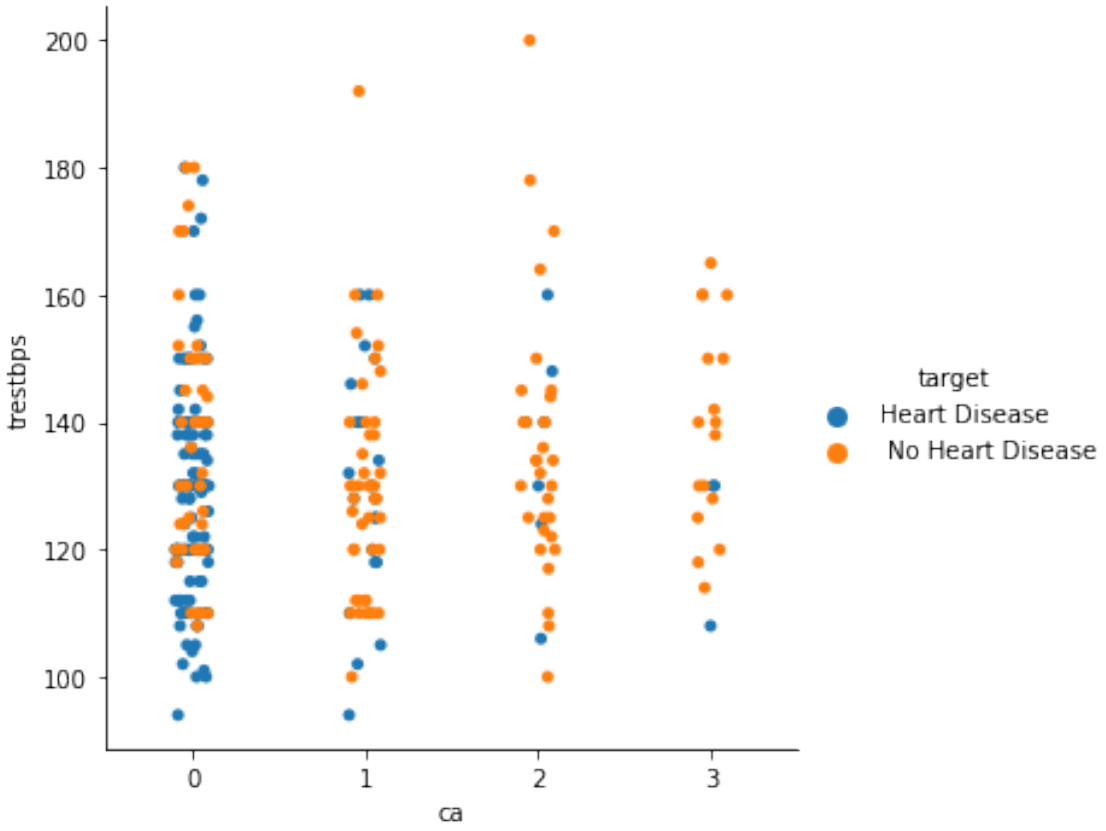
[73]:
```
sns.catplot(data=GP3, x='sex', y='trestbps',  hue='target')
```

[73]:
```
<seaborn.axisgrid.FacetGrid at 0x7f8463c46450>
```

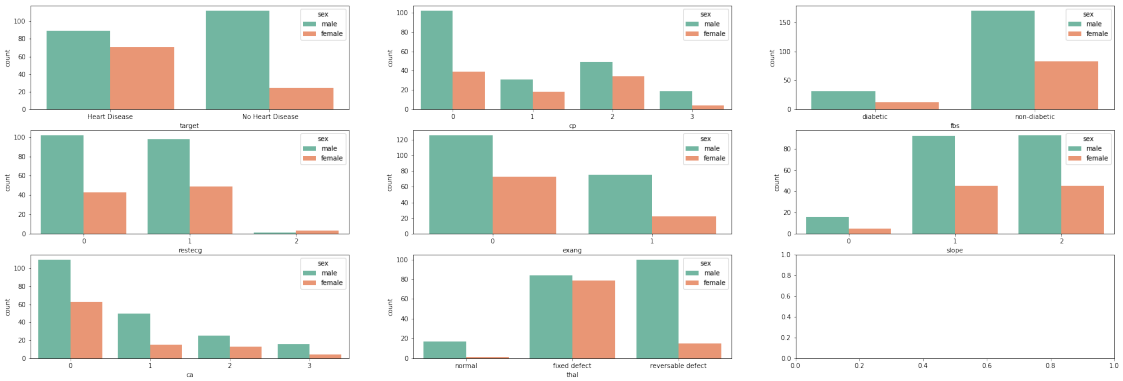


[146]:
```
sns.catplot(data=GP3, x='ca', y='trestbps',  hue='target')
```

[146]: <seaborn.axisgrid.FacetGrid at 0x7f846382ab50>

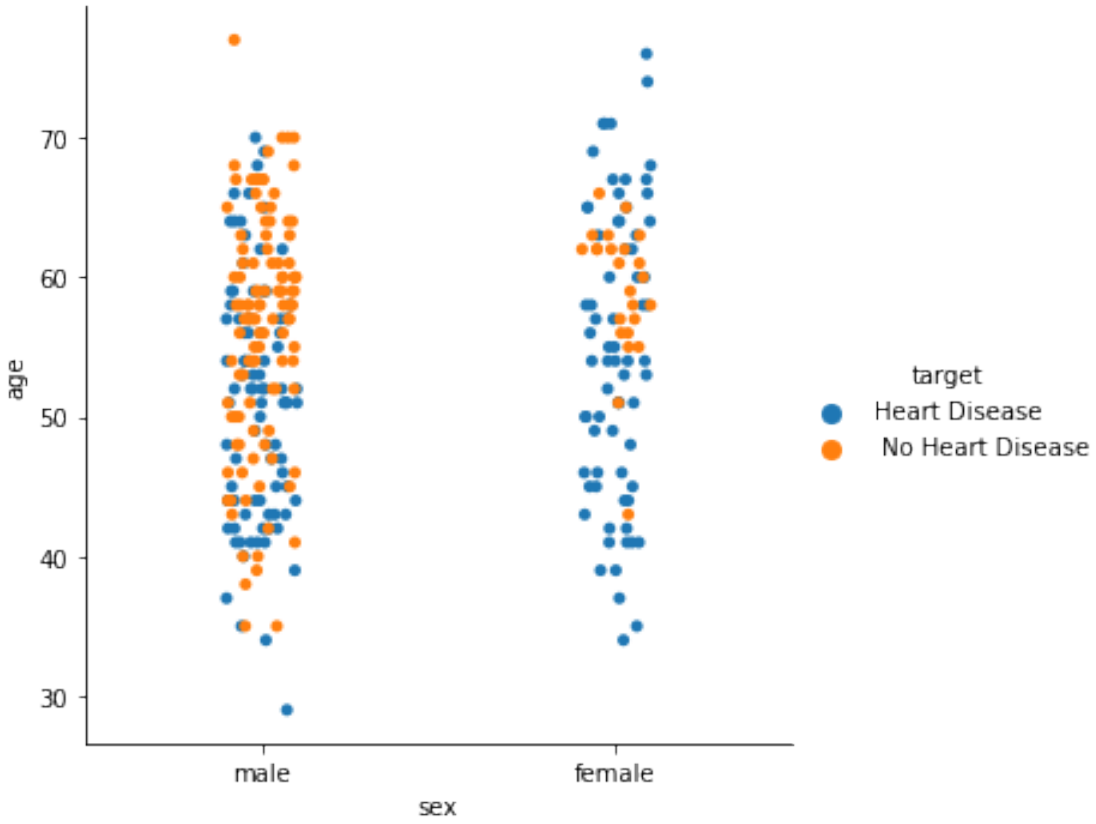


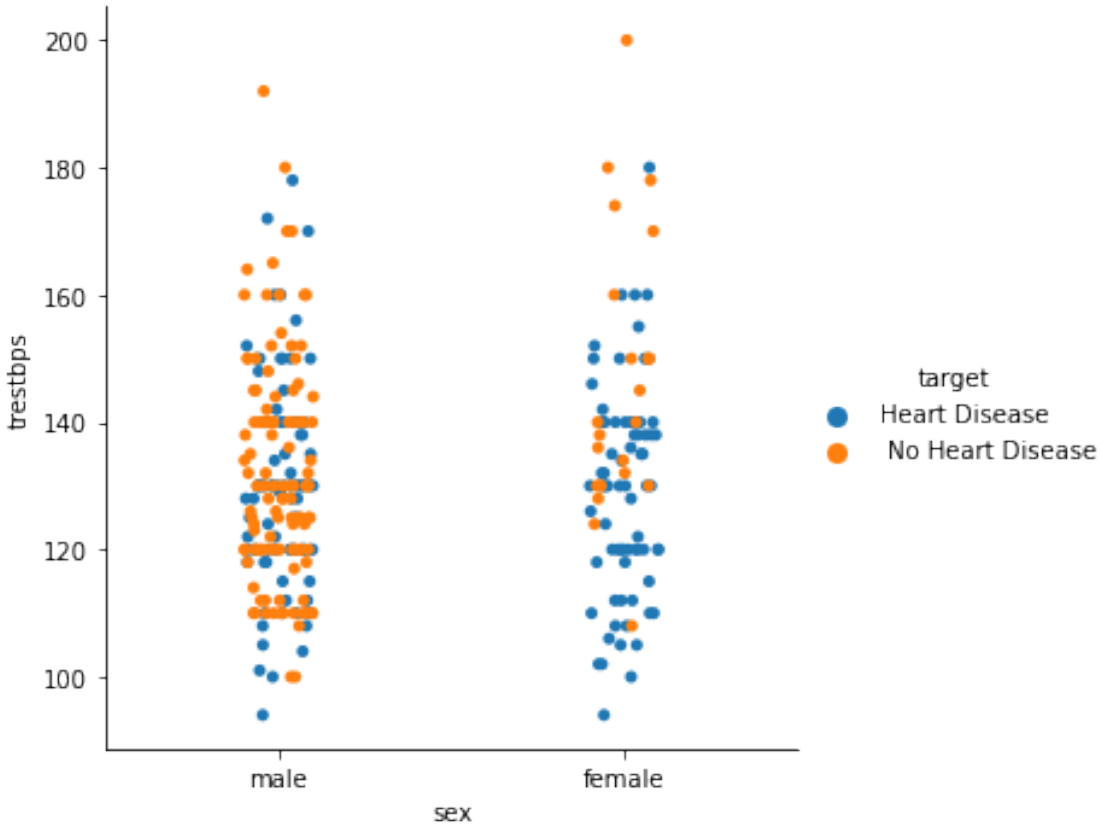
[59]:
```
# for plotting, group categorical features in cat_feat
# to create dist in 8 feature, 9th is gender,
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(30,10))
cat_feat = ['target', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']

for idx, feature in enumerate(cat_feat):
    ax = axes[int(idx/3), idx%3]
    if feature != 'sex':
        sns.countplot(x=feature, hue='sex', data=GP3, ax=ax, palette='Set2')
```

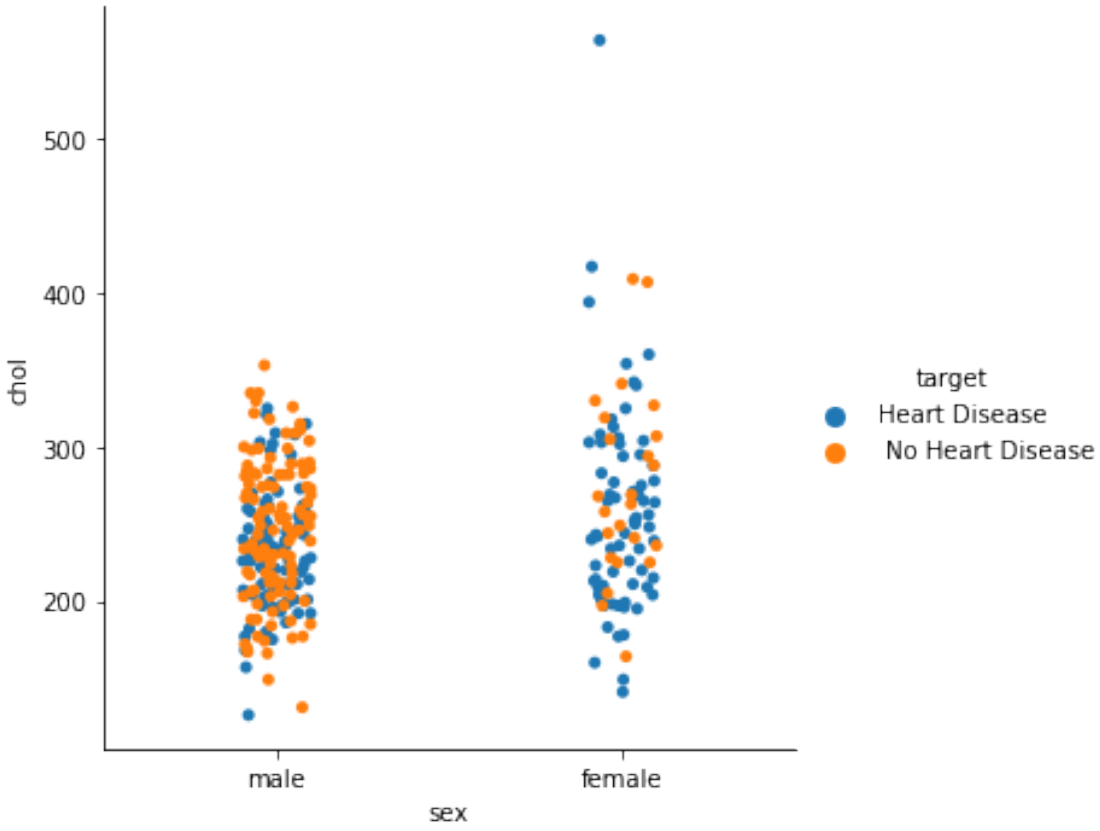


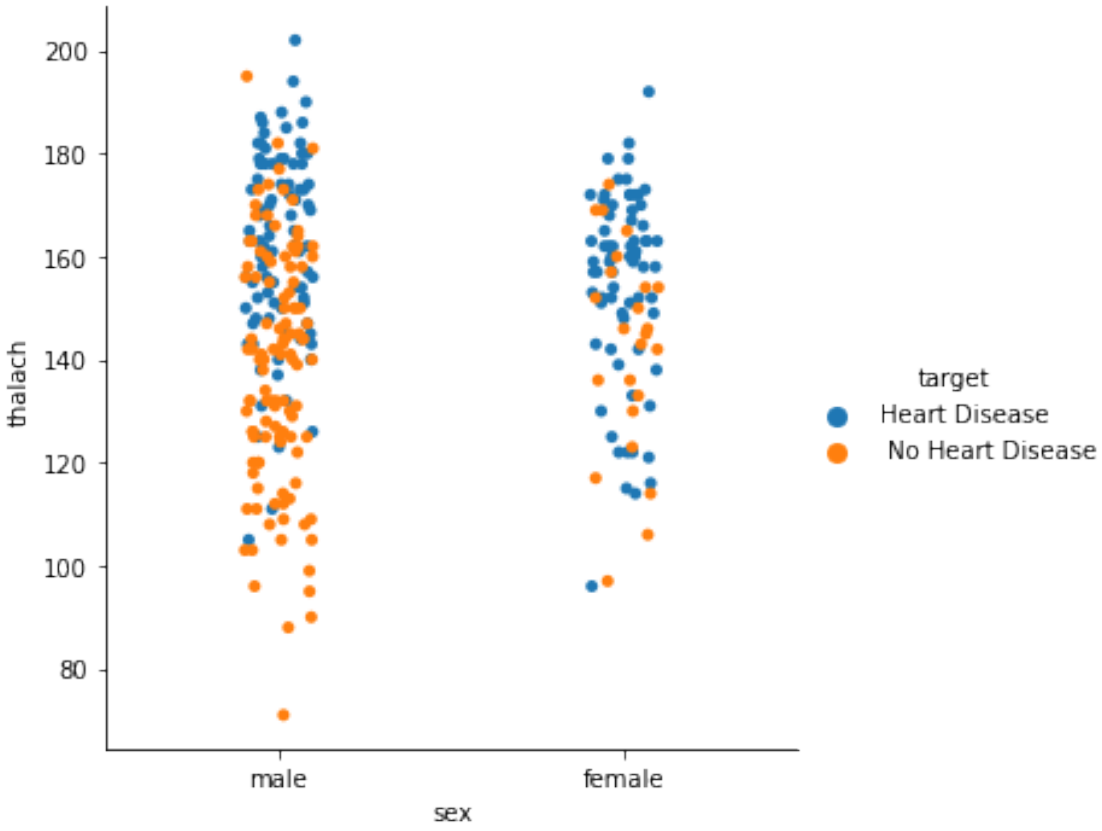
[75]:
```
sns.catplot(data=GP3, x='sex', y='age',  hue='target')
sns.catplot(data=GP3, x='sex', y='trestbps',  hue='target')
sns.catplot(data=GP3, x='sex', y='chol',  hue='target')
sns.catplot(data=GP3, x='sex', y='thalach',  hue='target')
sns.catplot(data=GP3, x='sex', y='oldpeak',  hue='target')
```

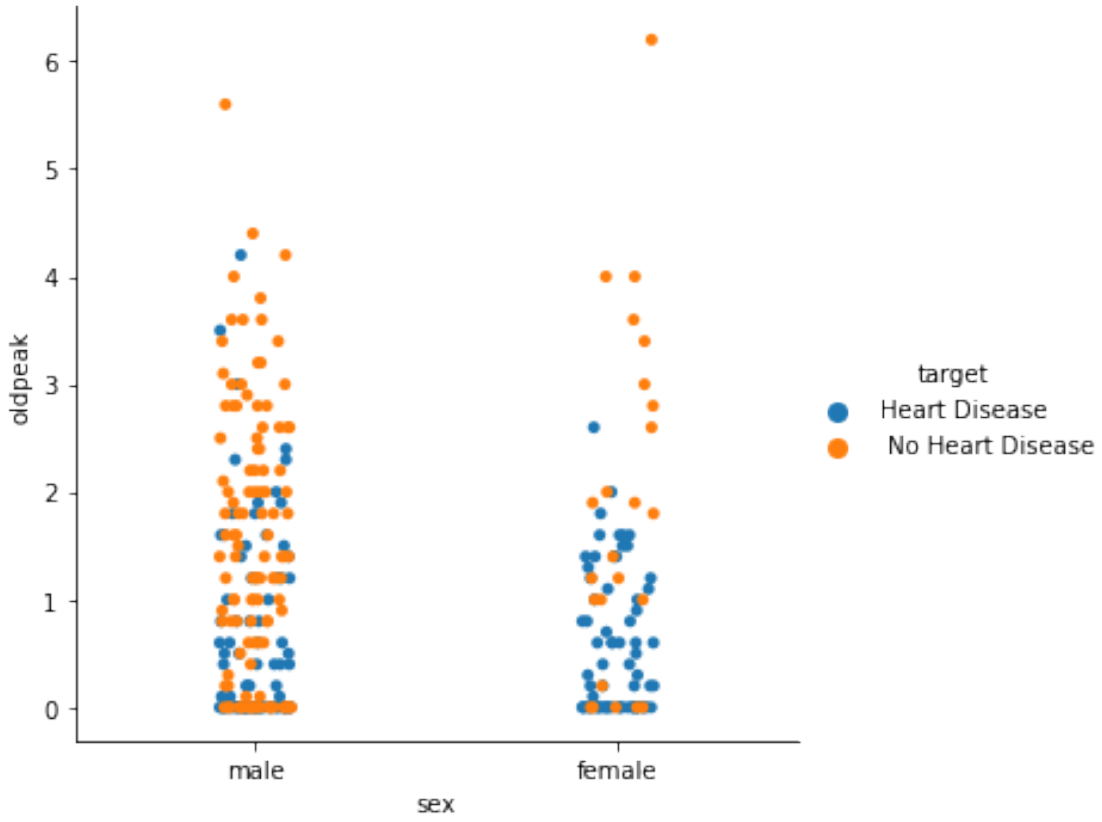
[75]:
```
<seaborn.axisgrid.FacetGrid at 0x7f84645b3490>
```



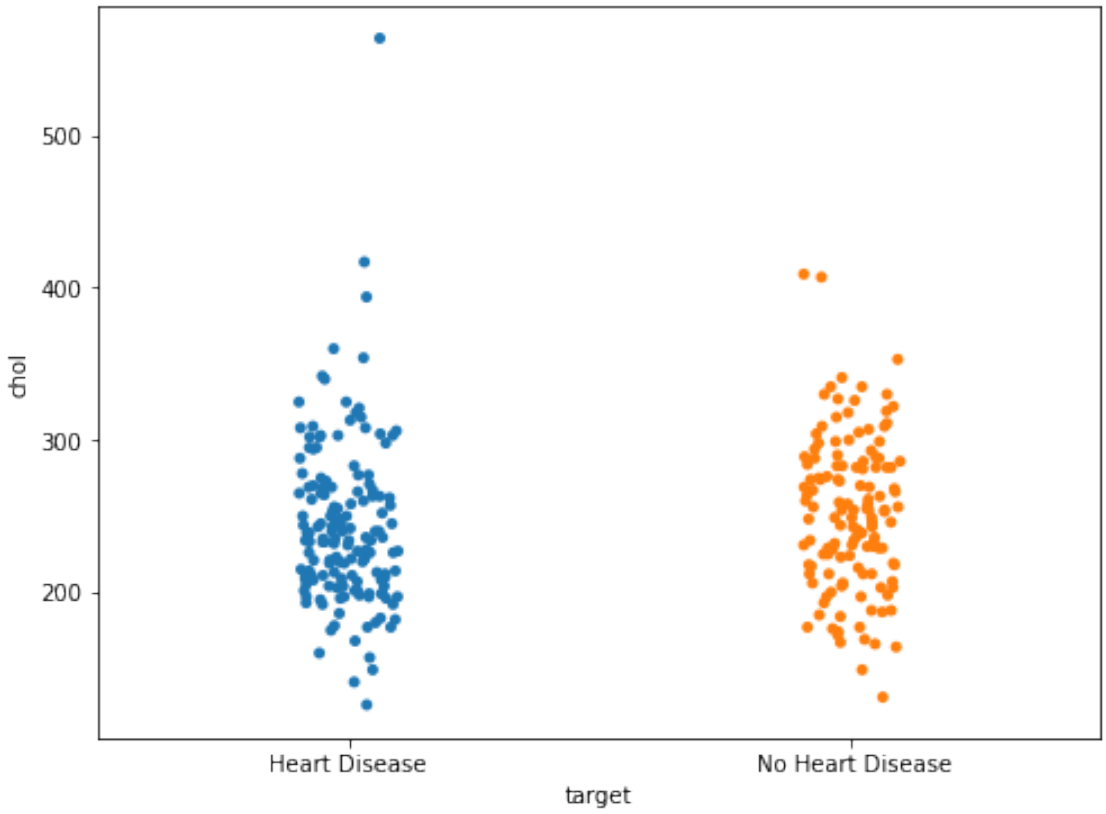

200
180
160
140
120
100
trestbps
male
female
sex
target
Heart Disease
No Heart Disease


500
400
300
200
chol
male
female
sex
target
Heart Disease
No Heart Disease


200
180
160
140
120
100
80
thalach
male
female
sex
target
Heart Disease
No Heart Disease

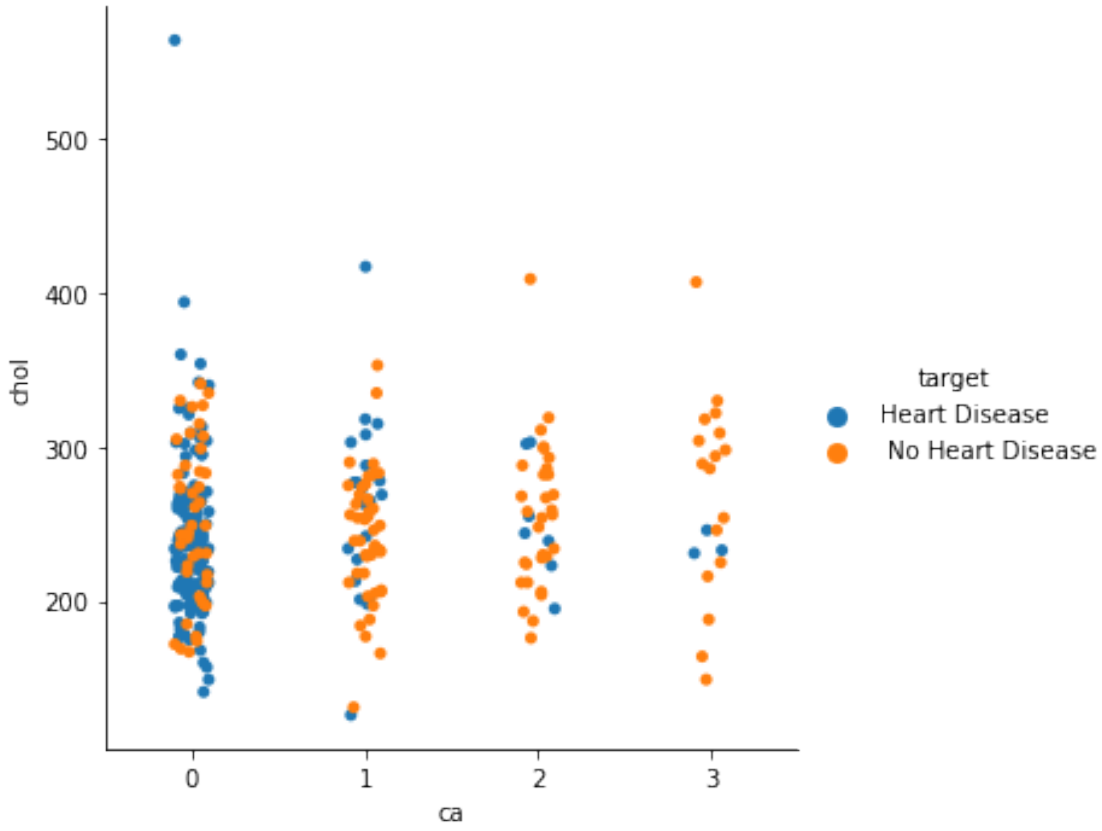


[88]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="target", y="chol", data=GP3)
plt.show()
```

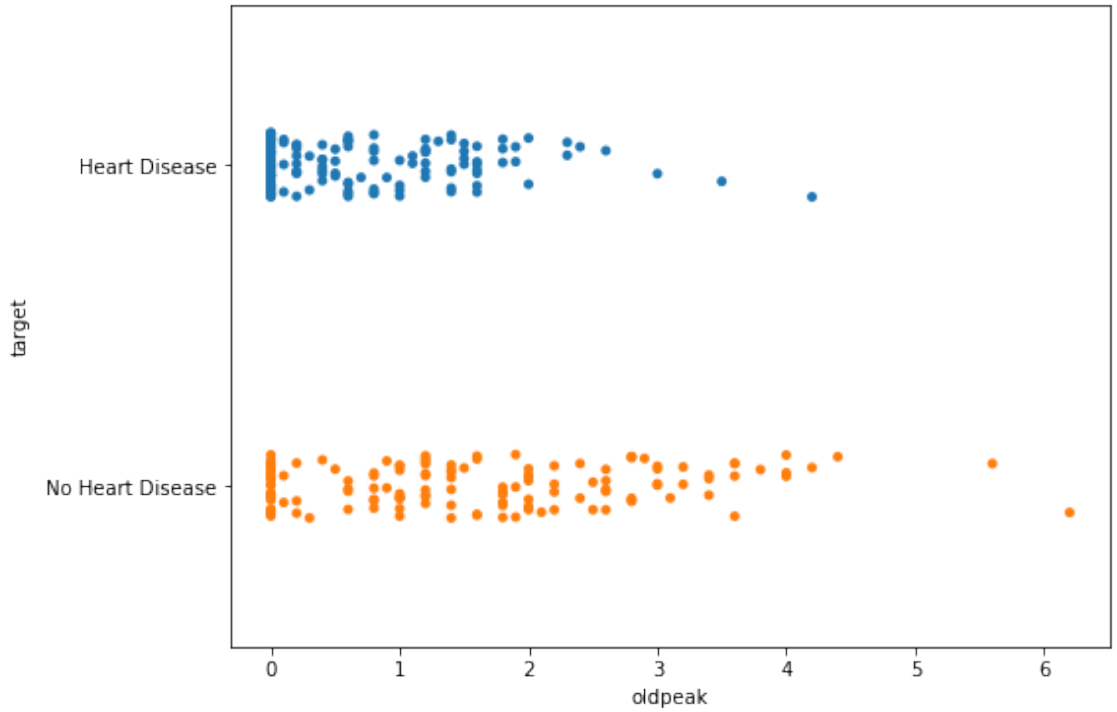


[85]:
```
sns.catplot(data=GP3, x='ca', y='chol',  hue='target')
```

[85]: <seaborn.axisgrid.FacetGrid at 0x7f84635f6710>

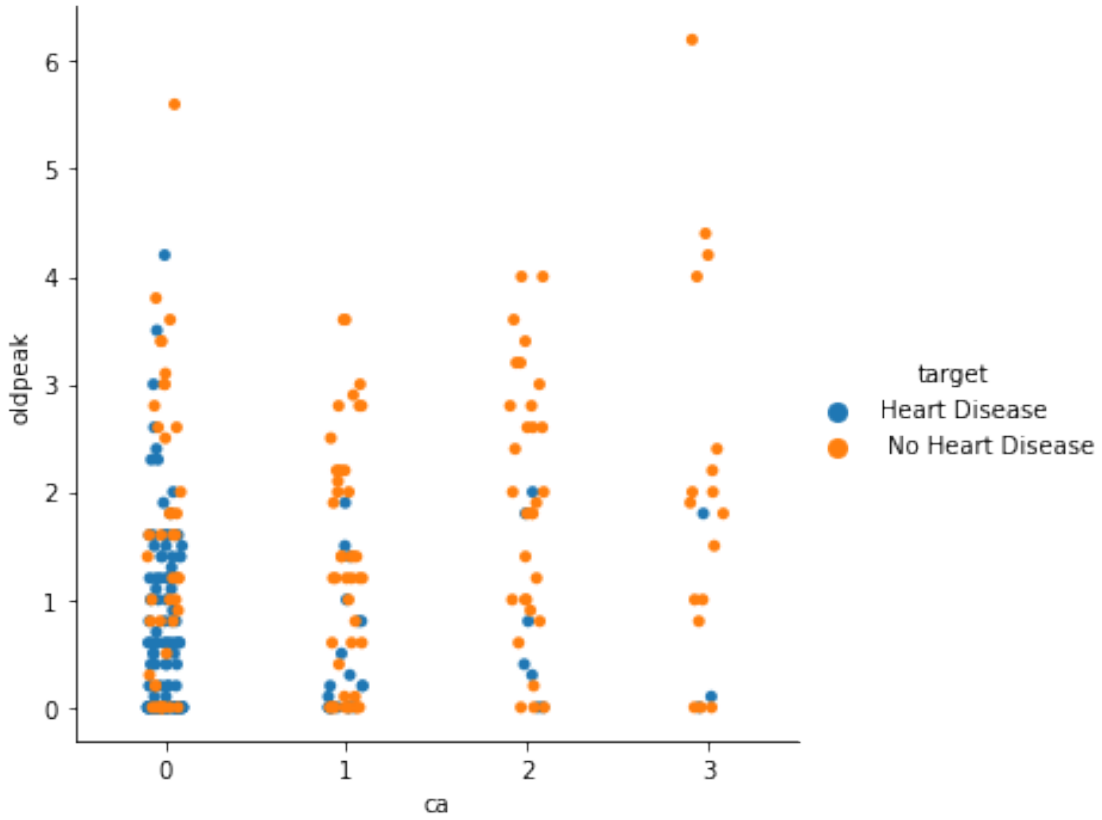


[90]:
```
f, ax = plt.subplots(figsize=(8, 6))
sns.stripplot(x="oldpeak", y="target", data=GP3)
plt.show()
```



[91]:
```
sns.catplot(data=GP3, x='ca', y='oldpeak',  hue='target')
```

[91]: <seaborn.axisgrid.FacetGrid at 0x7f84632aef90>

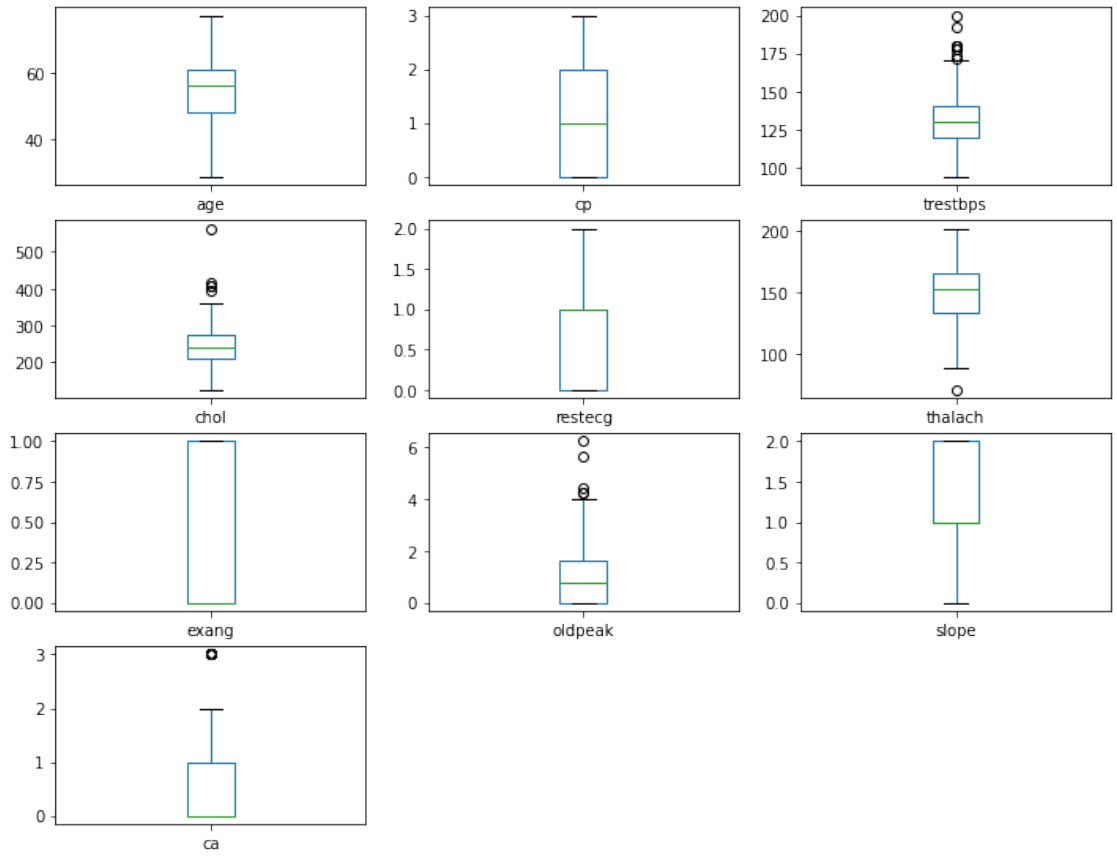


[95]:
```
# define continuous variable & plot
continous_features = ['age', 'chol', 'thalach', 'oldpeak','trestbps']
sns.pairplot(GP3[continous_features + ['target']], hue='target')
```

[95]: <seaborn.axisgrid.PairGrid at 0x7f84606c2ed0>

[114]:
```
#### This pair plot is not very conclusive, may be because of some outliers in␣
↪the data of bp, chol etc... Let's check for outliers
```

[116]:
```
GP3.plot(kind='box', subplots=True, layout=(5,3), figsize=(12,12))
plt.show()
```



[ ]:
```
#### outliers is seen for trestbps, cholesterol, oldpeak....
```

[106]:
```
GP6=GP.copy()
def gender(sex):
    if sex == 0:
        return 'female'
    else:
        return 'male'
GP6['sex'] = GP6['sex'].apply(gender)

def disease(t):
    if t == 0:
        return ' No Heart Disease'
    else:
        return 'Heart Disease'
GP6['target'] = GP6['target'].apply(disease)
```

[111]:
```
inp4=pd.get_dummies(GP6,columns=['sex', 'cp', 'fbs', 'restecg', 'exang',␣
↪'slope', 'ca', 'thal'])
```

[112]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
columns_to_scale = ['age','trestbps','chol','thalach','oldpeak']
inp4[columns_to_scale] = StandardScaler.fit_transform(inp4[columns_to_scale])
```

[113]:
```
# define continuous variable & plot
continous_features = ['age', 'chol', 'thalach', 'oldpeak','trestbps']
sns.pairplot(inp4[continous_features + ['target']], hue='target')
```

[113]: <seaborn.axisgrid.PairGrid at 0x7f8458380950>

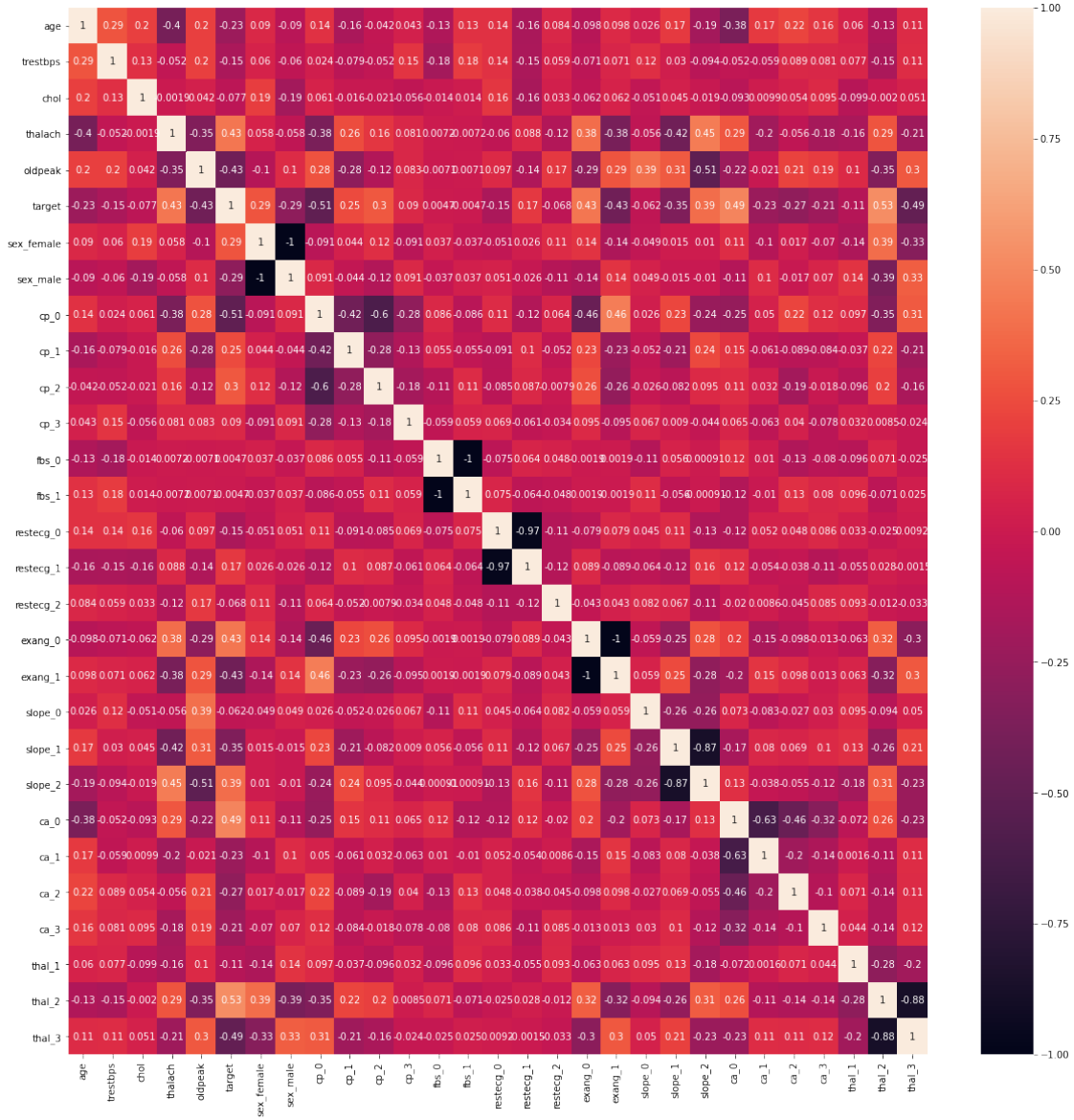
[121]:
```
GP7=GP.copy()
def gender(sex):
    if sex == 0:
        return 'female'
```

```
    else:
        return 'male'
GP7['sex'] = GP7['sex'].apply(gender)
```

[130]:
```
inp5=pd.get_dummies(GP7,columns=['sex', 'cp', 'fbs', 'restecg', 'exang',␣
↪'slope', 'ca', 'thal'])
```

[132]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
StandardScaler = StandardScaler()
columns_to_scale = ['age','trestbps','chol','thalach','oldpeak']
inp5[columns_to_scale] = StandardScaler.fit_transform(inp5[columns_to_scale])
```

[147]:
```
#visualize the correlation
Cor=inp5.corr()
plt.figure(figsize=(20,20))
sns.heatmap(inp5.corr(), annot=True)
plt.show()
```



[ ]:

[144]:
```
cor = dataCorr['target'].sort_values(ascending=False)
cor = cor.drop(['target'])
cor.to_frame()
```

[144]:

	target
thal_2	0.530032
ca_0	0.488146
thalach	0.426655
exang_0	0.425085

```
slope_2    0.386007
cp_2       0.303870
sex_0      0.285322
cp_1       0.246481
restecg_1  0.170030
cp_3       0.090342
fbs_0      0.004680
fbs_1     -0.004680
slope_0   -0.062087
restecg_2 -0.068235
chol      -0.076541
thal_1    -0.105799
trestbps  -0.148922
restecg_0 -0.154302
ca_3      -0.210955
age       -0.225453
ca_1      -0.231473
ca_2      -0.274408
sex_1     -0.285322
slope_1   -0.354225
exang_1   -0.425085
oldpeak   -0.428804
thal_3    -0.489046
cp_0      -0.505149
```

[149]:
```
X= inp5.drop(['target'], axis=1)
y= inp5['target']
```

[150]:
```
X_train, X_test,y_train, y_test=train_test_split(X,y,test_size=0.
↪3,random_state=1)
```

[151]:
```
print('X_train-', X_train.size)
print('X_test-',X_test.size)
print('y_train-', y_train.size)
print('y_test-', y_test.size)
```

```
X_train- 5796
X_test- 2492
y_train- 207
y_test- 89
```

[153]:
```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
LR
```

[153]:
```
LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
```

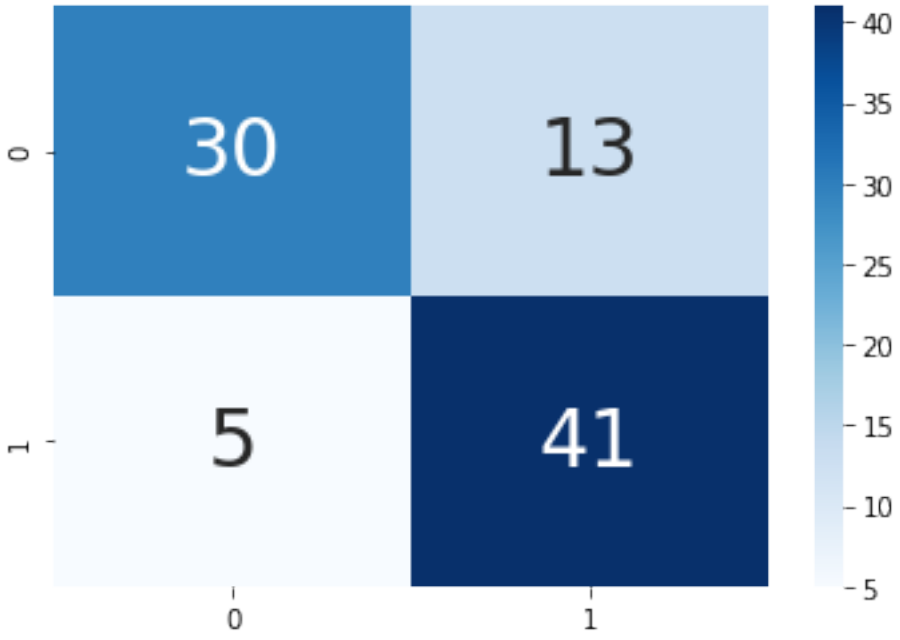
```
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

[154]:
```
y_predict = LR.predict(X_test)
y_predict[0:10]
```

[154]:
```
array([0, 0, 1, 0, 0, 1, 1, 1, 1, 0])
```

[155]:
```
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(y_test, y_predict)

sns.heatmap(conf_matrix, annot=True,cmap='Blues',annot_kws={"size": 30})


print('True Positive:\t{}'.format(conf_matrix[0,0]))
print('True Negative:\t{}'.format(conf_matrix[0,1]))
print('False Positive:\t{}'.format(conf_matrix[1,0]))
print('False Negative:\t{}'.format(conf_matrix[1,1]))
```

```
True Positive:  30
True Negative:  13
False Positive: 5
False Negative: 41
```



[156]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.86      0.70      0.77        43
           1       0.76      0.89      0.82        46

    accuracy                           0.80        89
   macro avg       0.81      0.79      0.79        89
weighted avg       0.81      0.80      0.80        89
```

[ ]:
```

```