Importing Libraries

In [78]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

Importing Data Set

In [79]:
```
dataset=pd.read_csv("/content/heart.csv")
```

In [80]:
```
dataset.shape
```

Out[80]: (1025, 14)

In [81]:
```
dataset.head(5)
```

Out[81]:

	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

1. age
2. sex (1=male; 0=female)
3. chest pain type(4 values)
4. resting blood pressure
5. serum cholestrol in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect (thallium heart scan or stress test)
14. target (0 = no heart disease; 1 = heart disease)

In [82]:
```
dataset.describe()
```

Out[82]:

	age	sex	cp	trestbps	chol	fbs	restec
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.0000C
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.52975
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.52787
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.0000C
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.0000C
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.0000C
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.0000C
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.0000C

Checking the null values in the Dataset

In [83]:
```
dataset.isnull().sum()#there are no null values in the dataset
```

Out[83]:
```
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 [84]:
```
dataset.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 [85]:
```
print(dataset.corr()["target"].abs().sort_values(ascending=False))
```

```
target      1.000000
oldpeak     0.438441
exang       0.438029
cp          0.434854
thalach     0.422895
ca          0.382085
slope       0.345512
thal        0.337838
sex         0.279501
age         0.229324
trestbps    0.138772
restecg     0.134468
chol        0.099966
fbs         0.041164
Name: target, dtype: float64
```

From the above information it shows that most columns are moderately correlated with target, but 'fbs' is very weakly correlated.

Finding the corelation between the variables

In [86]:
```
dataset.corr()
```

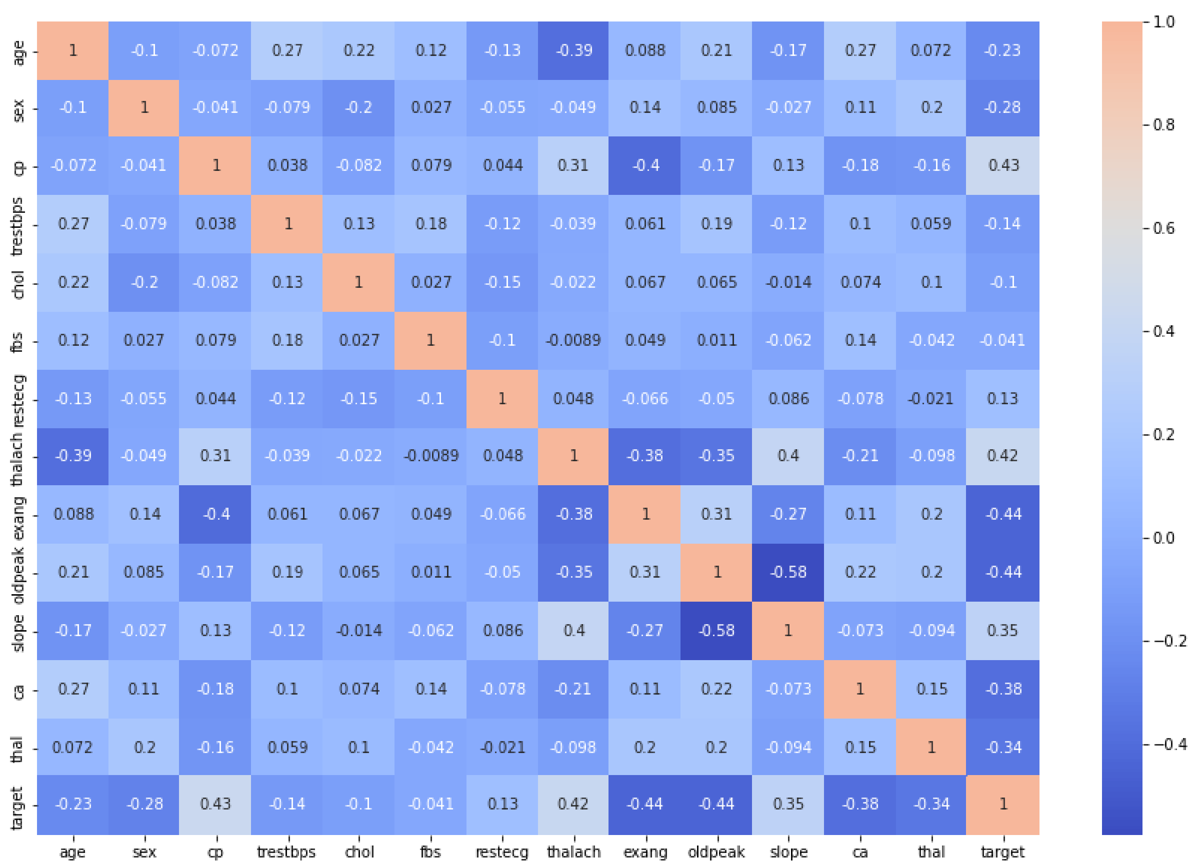
Out[86]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
age	1.000000	-0.103240	-0.071966	0.271121	0.219823	0.121243	-0.132696	-0.390227
sex	-0.103240	1.000000	-0.041119	-0.078974	-0.198258	0.027200	-0.055117	-0.049365
cp	-0.071966	-0.041119	1.000000	0.038177	-0.081641	0.079294	0.043581	0.306839
trestbps	0.271121	-0.078974	0.038177	1.000000	0.127977	0.181767	-0.123794	-0.039264
chol	0.219823	-0.198258	-0.081641	0.127977	1.000000	0.026917	-0.147410	-0.021772
fbs	0.121243	0.027200	0.079294	0.181767	0.026917	1.000000	-0.104051	-0.008866
restecg	-0.132696	-0.055117	0.043581	-0.123794	-0.147410	-0.104051	1.000000	0.048411
thalach	-0.390227	-0.049365	0.306839	-0.039264	-0.021772	-0.008866	0.048411	1.000000
exang	0.088163	0.139157	-0.401513	0.061197	0.067382	0.049261	-0.065606	-0.380281
oldpeak	0.208137	0.084687	-0.174733	0.187434	0.064880	0.010859	-0.050114	-0.349796
slope	-0.169105	-0.026666	0.131633	-0.120445	-0.014248	-0.061902	0.086086	0.395308
ca	0.271551	0.111729	-0.176206	0.104554	0.074259	0.137156	-0.078072	-0.207888
thal	0.072297	0.198424	-0.163341	0.059276	0.100244	-0.042177	-0.020504	-0.098068
target	-0.229324	-0.279501	0.434854	-0.138772	-0.099966	-0.041164	0.134468	0.422895

Representing the coorelation variables in heatmap

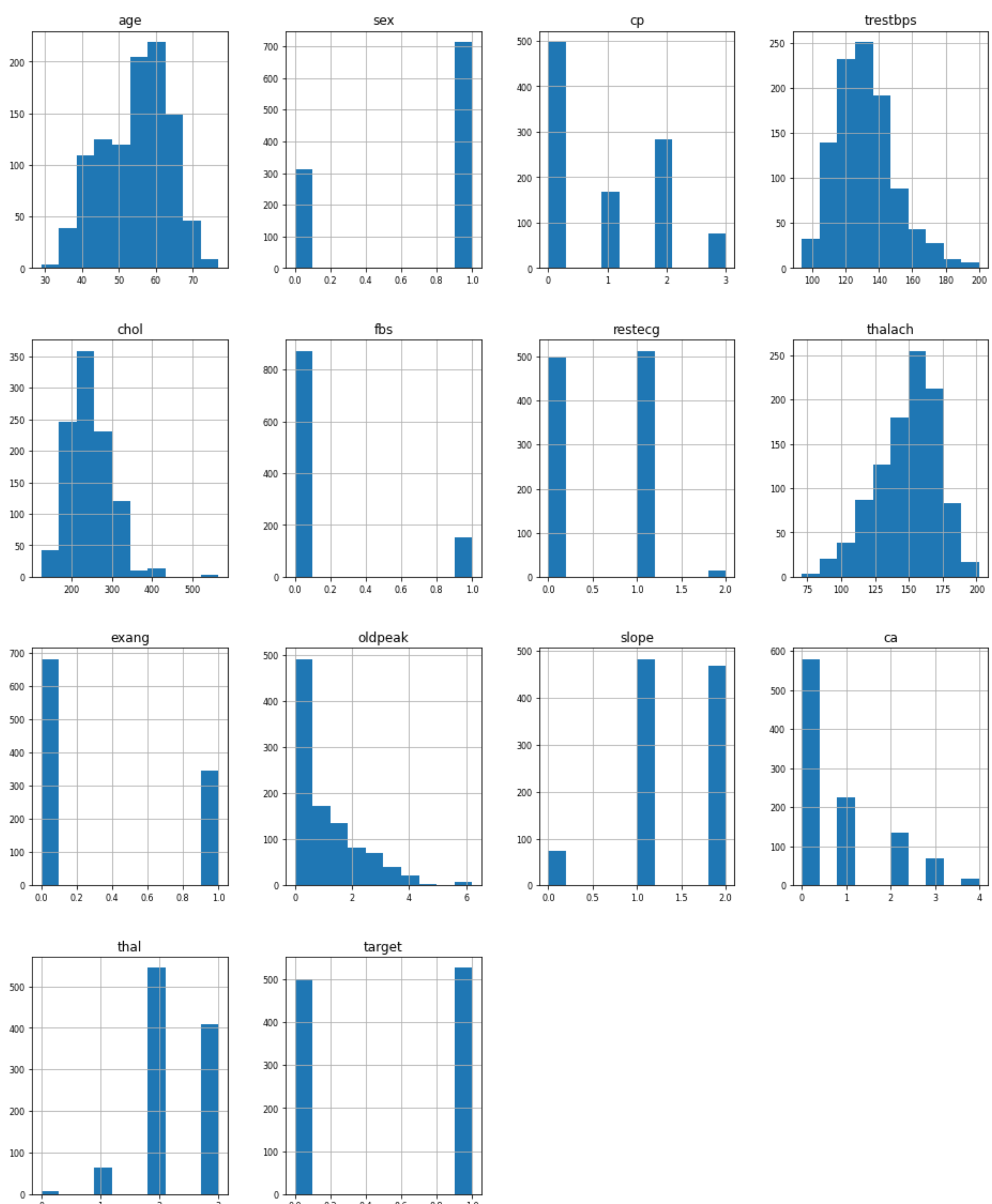
In [87]:
```
f, ax = plt.subplots(figsize=(15, 10))
sns.heatmap(dataset.corr(),annot=True,cmap='coolwarm',center=0.6)
```

Out[87]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2233fec90>



In [88]:
```
dataset.hist(figsize=(16, 20), xlabelsize=8, ylabelsize=8)
```

Out[88]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fd2231a1b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd2231738d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd223136ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd2230f8510>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd2230aeb10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd223072150>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd2230297d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd222fdfcd0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd222fdfd10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd222fa42d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd222f91bd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd222f55110>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7fd222f0c610>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd222ec4b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd222e7afd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fd222e3c550>]],
      dtype=object)
```



Analysing the target variable

In [89]:
```
dataset["target"].unique()
```

Out[89]:
```
array([0, 1])
```

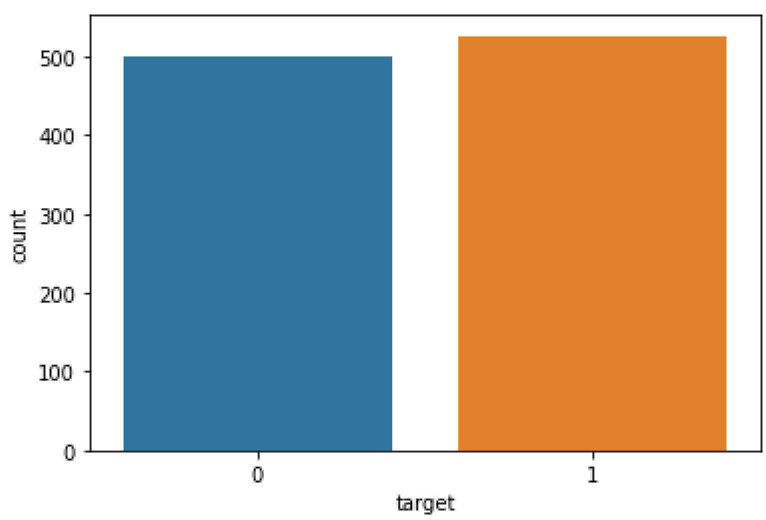
Clearly, this is a classification problem, with the target variable having values '0' and '1'

In [90]:
```
dataset["target"].describe()
```

Out[90]:
```
count    1025.000000
mean        0.513171
std         0.500070
min         0.000000
25%         0.000000
50%         1.000000
75%         1.000000
max         1.000000
Name: target, dtype: float64
```

In [91]:
```
y = dataset["target"]

sns.countplot(y)


target_temp = dataset.target.value_counts()

print(target_temp)
```

```
1    526
0    499
Name: target, dtype: int64

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



We will analyse all other features

Analysing 'sex' featutre

In [92]:
```
dataset['sex'].value_counts()
```

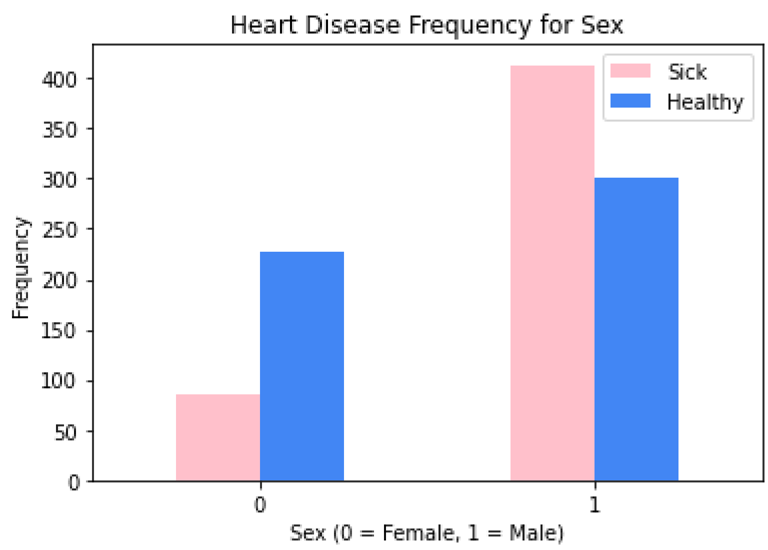
Out[92]:
```
1    713
0    312
Name: sex, dtype: int64
```

- Sex feature have 2 unique features
- Sex (1= male; 0=female)

In [93]:
```
dataset.groupby(['sex', 'target'])['sex'].count()
```

Out[93]:
```
sex  target
0    0          86
     1         226
1    0         413
     1         300
Name: sex, dtype: int64
```

In [94]:
```
pd.crosstab(dataset.sex,dataset.target).plot(kind="bar",color=['pink','#4286f
4' ])
plt.title('Heart Disease Frequency for Sex')
plt.xlabel('Sex (0 = Female, 1 = Male)')
plt.xticks(rotation=0)
plt.legend(["Sick", "Healthy"])
plt.ylabel('Frequency')
plt.show()
```



It look's like many females are suffering more from the heartdisease.

Analysing Chest Pain Type Feature

In [95]:
```
dataset["cp"].unique()
```

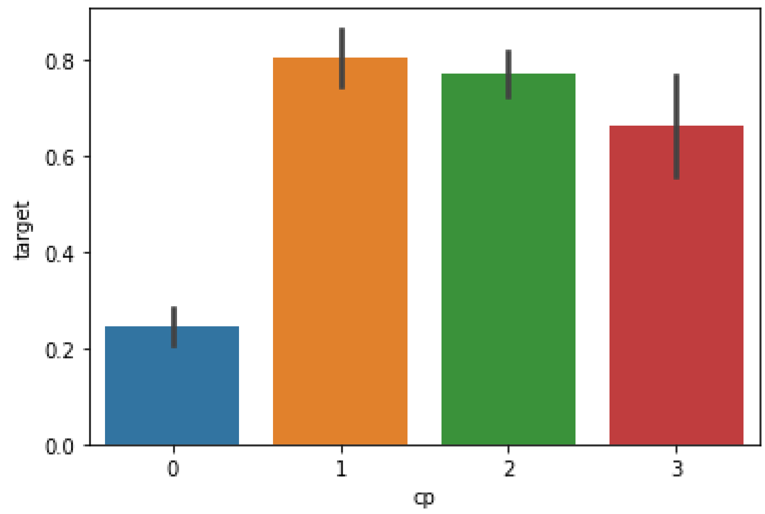
Out[95]:
```
array([0, 1, 2, 3])
```

We can see that chest pain feature have 0 to 3 values.

In [96]:
```
sns.barplot(dataset["cp"],y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[96]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd222984750>



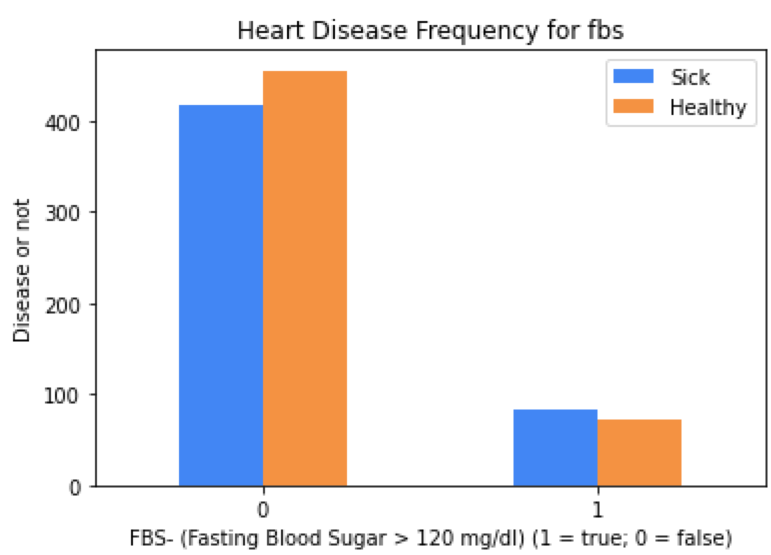
From the above barplot we can say persons with chest pain '0' typical angina are less likely to have heart problems.

Analysing 'Fasting Blood Sugar' feature fbs

In [97]:
```
dataset['fbs'].unique()
```

Out[97]: array([0, 1])

In [98]:
```
pd.crosstab(dataset.fbs,dataset.target).plot(kind="bar",color=['#4286f4','#f49
242'])
plt.title('Heart Disease Frequency for fbs')
plt.xlabel('FBS- (Fasting Blood Sugar > 120 mg/dl) (1 = true; 0 = false)')
plt.xticks(rotation=0)
plt.legend(["Sick", "Healthy"])
plt.ylabel('Disease or not')
plt.show()
```



Analysing the restecg feature

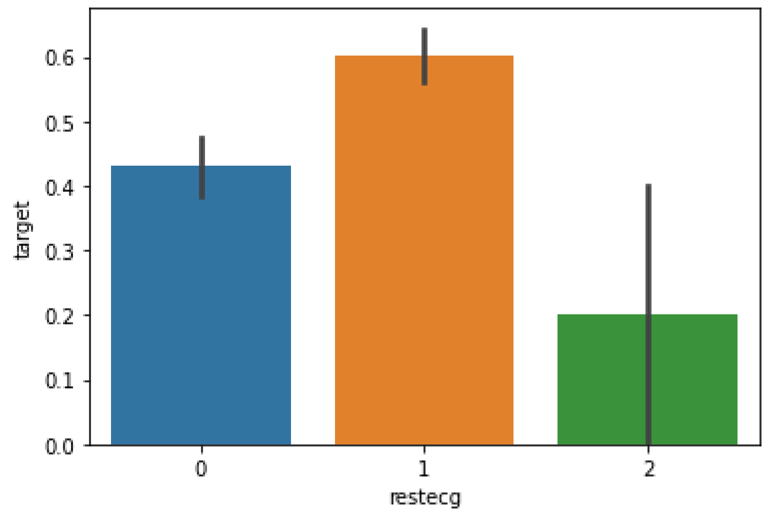
In [99]:
```
dataset['restecg'].unique()
```

Out[99]: array([1, 0, 2])

In [100]:
```
sns.barplot(dataset["restecg"],y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[100]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2228c8fd0>



From the above information we can say that the people with restecg '1' and '0' are having more chances of heart disease than with restecg '2'

Analysing the 'exang' feature

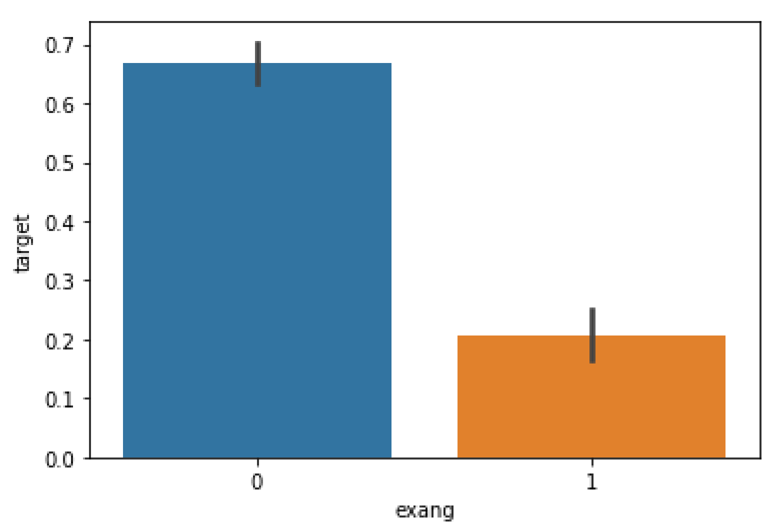
In [101]:
```
dataset['exang'].unique()
```

Out[101]: array([0, 1])

In [102]:
```
sns.barplot(dataset["exang"],y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[102]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd222854590>



People having exang=1 means Exercise includes angina are having less chances heart problems.(1 = yes; 0 = no)

Analysing the slope of the peak exercise ST segment (Value 1: upsloping, Value 2: flat, Value 3: downsloping)

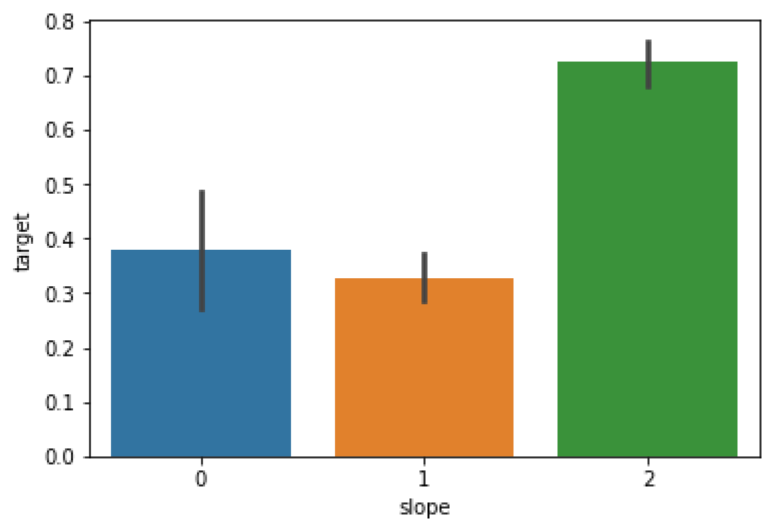
In [103]:
```
dataset["slope"].unique()
```

Out[103]:
```
array([2, 0, 1])
```

In [104]:
```
sns.barplot(dataset["slope"],y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[104]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2227d2d10>
```



from the above information we conclude that slope2 causes much pain than the slope 0 and slope 1.

Analysing the 'ca' feature Number of major vessels (0-3) colored by flourosopy

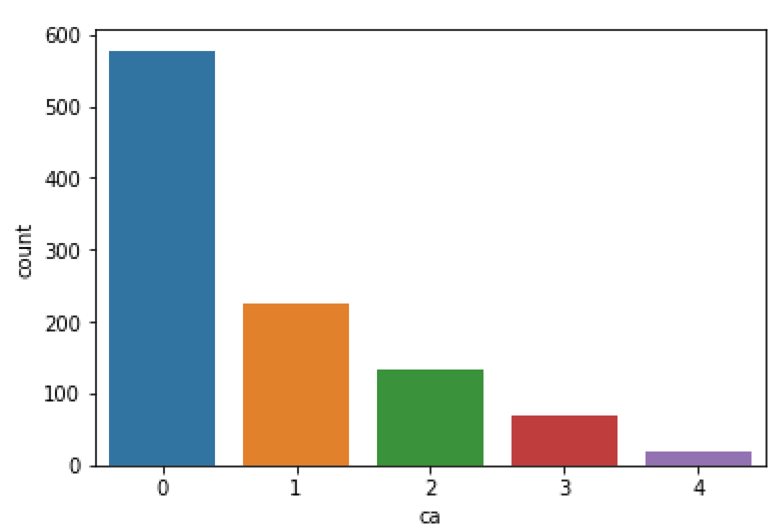
In [105]:
```
dataset["ca"].unique()
```

Out[105]:
```
array([2, 0, 1, 3, 4])
```

In [106]:
```
sns.countplot(dataset["ca"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the o
nly valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

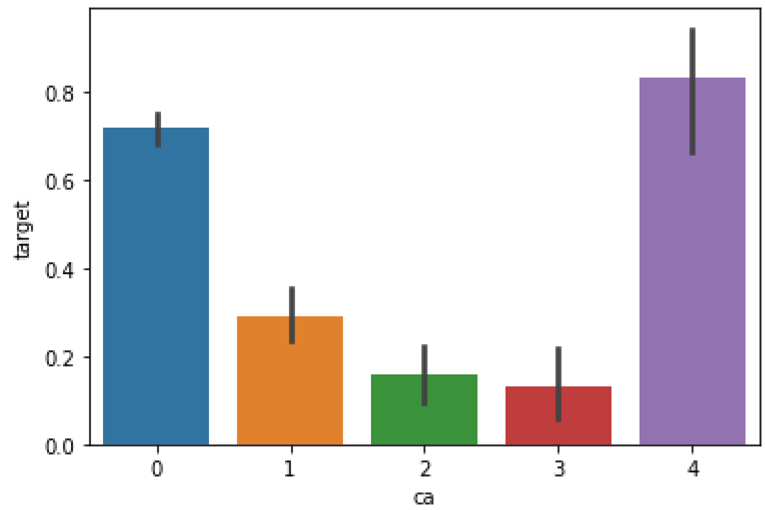
Out[106]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd222747890>
```



In [107]:
```
sns.barplot(dataset["ca"],y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[107]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2226a8390>



ca=4 has large number of heart patients

Analysing the 'thal' feature

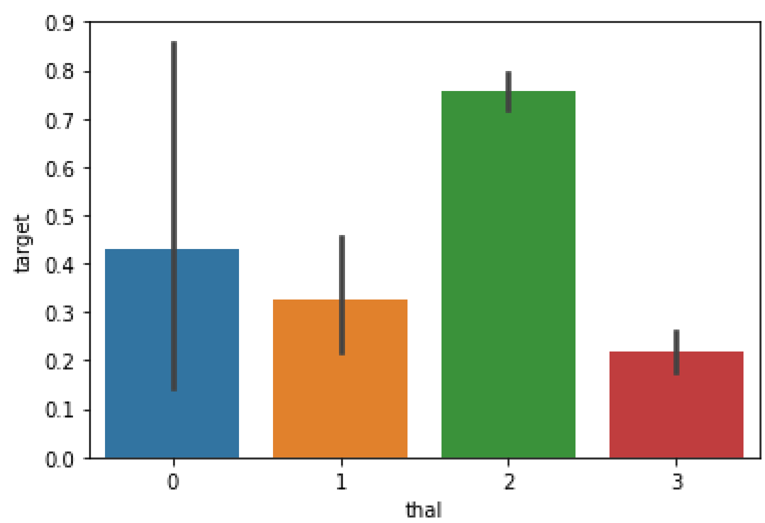
In [108]:
```
dataset["thal"].unique()
```

Out[108]: array([3, 2, 1, 0])

In [109]:
```
sns.barplot(dataset["thal"],y)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, th
e only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

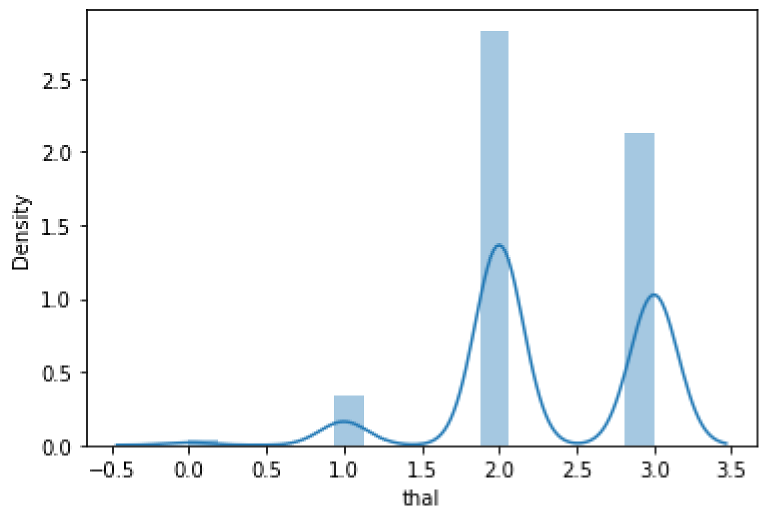
Out[109]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2226a2550>



In [110]:
```
sns.distplot(dataset["thal"])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level functi
on with similar flexibility) or `histplot` (an axes-level function for histog
rams).
  warnings.warn(msg, FutureWarning)
```

Out[110]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd222610b90>



Splitting the dataset to Train and Test

In [111]:
```
from sklearn.model_selection import train_test_split

X = dataset.drop("target",axis=1)
Y = dataset["target"]

x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.20,random_sta
te=0)
```

In [112]:
```
x_train.shape
```

Out[112]: (820, 13)

In [113]:
```
x_test.shape
```

Out[113]: (205, 13)

In [114]:
```
y_train.shape
```

Out[114]: (820,)

In [115]:
```
y_test.shape
```

Out[115]: (205,)

Building Models

Logistic Regression

In [116]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

lr = LogisticRegression()

lr.fit(x_train,y_train)

Y_pred_lr = lr.predict(x_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818:
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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

In [117]:
```
score_lr = round(accuracy_score(Y_pred_lr,y_test)*100,2)

print("The accuracy score achieved using Logistic Regression is: "+str(score_l
r)+" %")
```

```
The accuracy score achieved using Logistic Regression is: 86.34 %
```

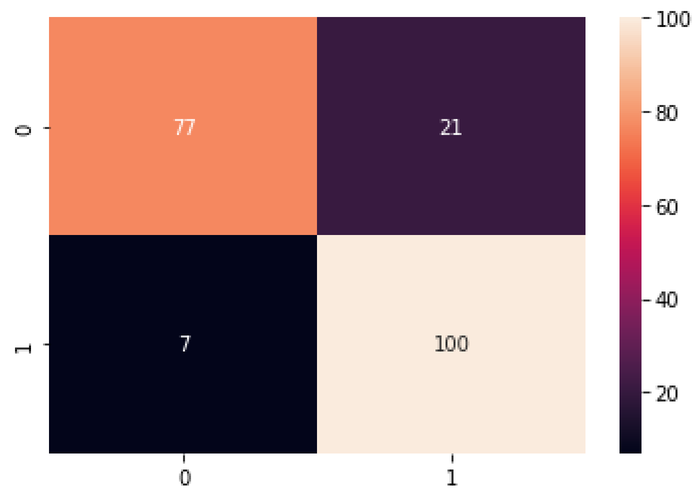
Confusion Matrix

In [118]:
```
from sklearn.metrics import confusion_matrix
```

In [119]:
```
matrix= confusion_matrix(y_test, Y_pred_lr)
```

In [120]:
```
sns.heatmap(matrix,annot = True, fmt = "d")
```

Out[120]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd2224e7c10>
```



In [121]:
```
from sklearn.metrics import precision_score
precision = precision_score(y_test, Y_pred_lr)
print("Precision: ",precision)
```

```
Precision:  0.8264462809917356
```

In [122]:
```
from sklearn.metrics import recall_score
recall = recall_score(y_test, Y_pred_lr)
print("Recall is: ",recall)
```

```
Recall is:  0.9345794392523364
```

SVM

In [123]:
```
from sklearn import svm

sv = svm.SVC(kernel='linear')

sv.fit(x_train, y_train)

Y_pred_svm = sv.predict(x_test)
```

In [124]:
```
score_svm = round(accuracy_score(Y_pred_svm,y_test)*100,2)

print("The accuracy score achieved using Linear SVM is: "+str(score_svm)+" %")
```

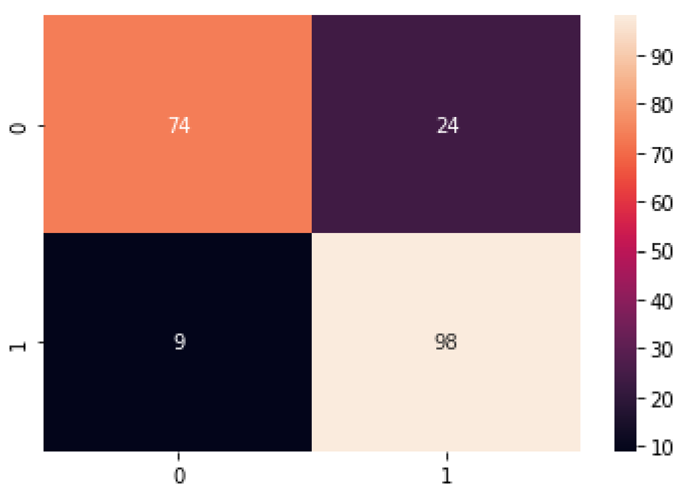
```
The accuracy score achieved using Linear SVM is: 83.9 %
```

Confusion Matrix for SVM

In [125]:
```
matrix= confusion_matrix(y_test, Y_pred_svm)
```

In [126]:
```
sns.heatmap(matrix,annot = True, fmt = "d")
```

Out[126]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd22248de50>
```



In [127]:
```
from sklearn.metrics import recall_score
recall = recall_score(y_test, Y_pred_svm)
print("Recall is: ",recall)
```

```
Recall is:  0.9158878504672897
```

In [128]:
```
from sklearn.metrics import precision_score
precision = precision_score(y_test, Y_pred_svm)
print("Precision: ",precision)
```

```
Precision:  0.8032786885245902
```

K Nearest Neighbors

In [129]:
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(x_train,y_train)
Y_pred_knn=knn.predict(x_test)
```

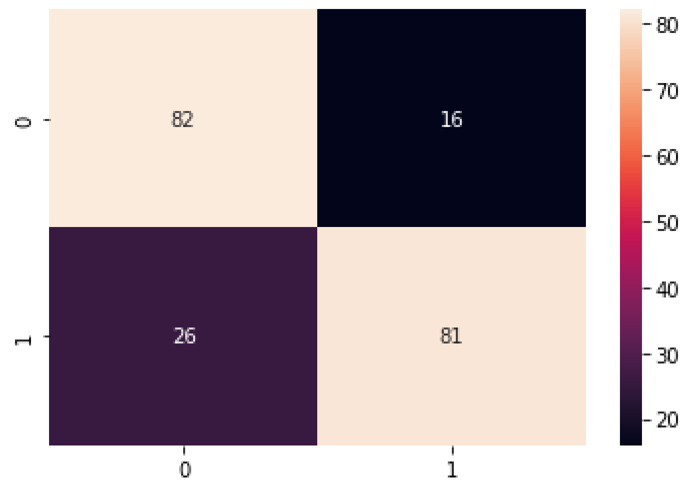
In [130]:
```
score_knn = round(accuracy_score(Y_pred_knn,y_test)*100,2)

print("The accuracy score achieved using KNN is: "+str(score_knn)+" %")
```

```
The accuracy score achieved using KNN is: 79.51 %
```

Confusion Matrix for KNN

In [131]:
```
matrix= confusion_matrix(y_test, Y_pred_knn)
sns.heatmap(matrix,annot = True, fmt = "d")
```

Out[131]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd2223b6550>



In [132]:
```
from sklearn.metrics import recall_score
recall = recall_score(y_test, Y_pred_knn)
print("Recall is: ",recall)
```

```
Recall is:  0.7570093457943925
```

In [133]:
```
from sklearn.metrics import precision_score
precision = precision_score(y_test, Y_pred_knn)
print("Precision: ",precision)
```

```
Precision:  0.8350515463917526
```

In [134]:
```
scores = [score_lr,score_svm,score_knn,]
algorithms = ["Logistic Regression","Support Vector Machine","K-Nearest Neighb
ors"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(score
s[i])+" %")
```

```
The accuracy score achieved using Logistic Regression is: 86.34 %
The accuracy score achieved using Support Vector Machine is: 83.9 %
The accuracy score achieved using K-Nearest Neighbors is: 79.51 %
```