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

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df=pd.read_csv('heart.csv')
df.head()
```

Out[2]:

	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 [3]:
```
df.exang.value_counts()
```

Out[3]:
```
0    204
1     99
Name: exang, dtype: int64
```

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]:
```
df.describe()
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	tha
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.90
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.00
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.50
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.00
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.00
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.00

In [6]:
```
df.corr()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.0
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.1
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.3
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.0
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.0
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.0
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.0
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.3
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.0
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.2
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.2
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.1
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.2
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.4

In [7]:
```
df.cov()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thala
age	82.484558	-0.416661	-0.643499	44.495902	100.585076	0.392433	-0.555013	-82.9033
sex	-0.416661	0.217166	-0.023736	-0.463970	-4.780309	0.007475	-0.014261	-0.4698
cp	-0.643499	-0.023736	1.065132	0.861714	-4.113774	0.034719	0.024108	6.9916
trestbps	44.495902	-0.463970	0.861714	307.586453	111.967215	1.109042	-1.052324	-18.7591
chol	100.585076	-4.780309	-4.113774	111.967215	2686.426748	0.245427	-4.116703	-11.8004
fbs	0.392433	0.007475	0.034719	1.109042	0.245427	0.126877	-0.015769	-0.0698
restecg	-0.555013	-0.014261	0.024108	-1.052324	-4.116703	-0.015769	0.276528	0.5314
thalach	-82.903318	-0.469871	6.991618	-18.759131	-11.800494	-0.069897	0.531462	524.6464
exang	0.413022	0.031014	-0.191168	0.557111	1.631991	0.004295	-0.017474	-4.0762
oldpeak	2.214583	0.051993	-0.178821	3.934486	3.246794	0.002377	-0.035883	-9.1535
slope	-0.944791	-0.008819	0.076137	-1.312832	-0.128964	-0.013147	0.030151	5.4593
ca	2.566356	0.056357	-0.191080	1.818373	3.737252	0.050259	-0.038741	-4.9932
thal	0.378139	0.059930	-0.102201	0.668022	3.135488	-0.006983	-0.003858	-1.3524
target	-1.021343	-0.065307	0.223330	-1.267950	-2.203855	-0.004983	0.035998	4.8187

In [8]:
```
df.age.value_counts()
```

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

In [9]:
```
df.sex.value_counts()
```

Out[9]:
```
1    207
0     96
Name: sex, dtype: int64
```

In [10]:
```
df.cp.value_counts()
```

Out[10]:
```
0    143
2     87
1     50
3     23
Name: cp, dtype: int64
```

In [11]:
```
pd.crosstab(df.sex,df.cp).T.head()
```

Out[11]:

sex	0	1
cp		
0	39	104
1	18	32
2	35	52
3	4	19

In [12]:
```
pd.crosstab(df.sex,df.age).T.head()
```

Out[12]:

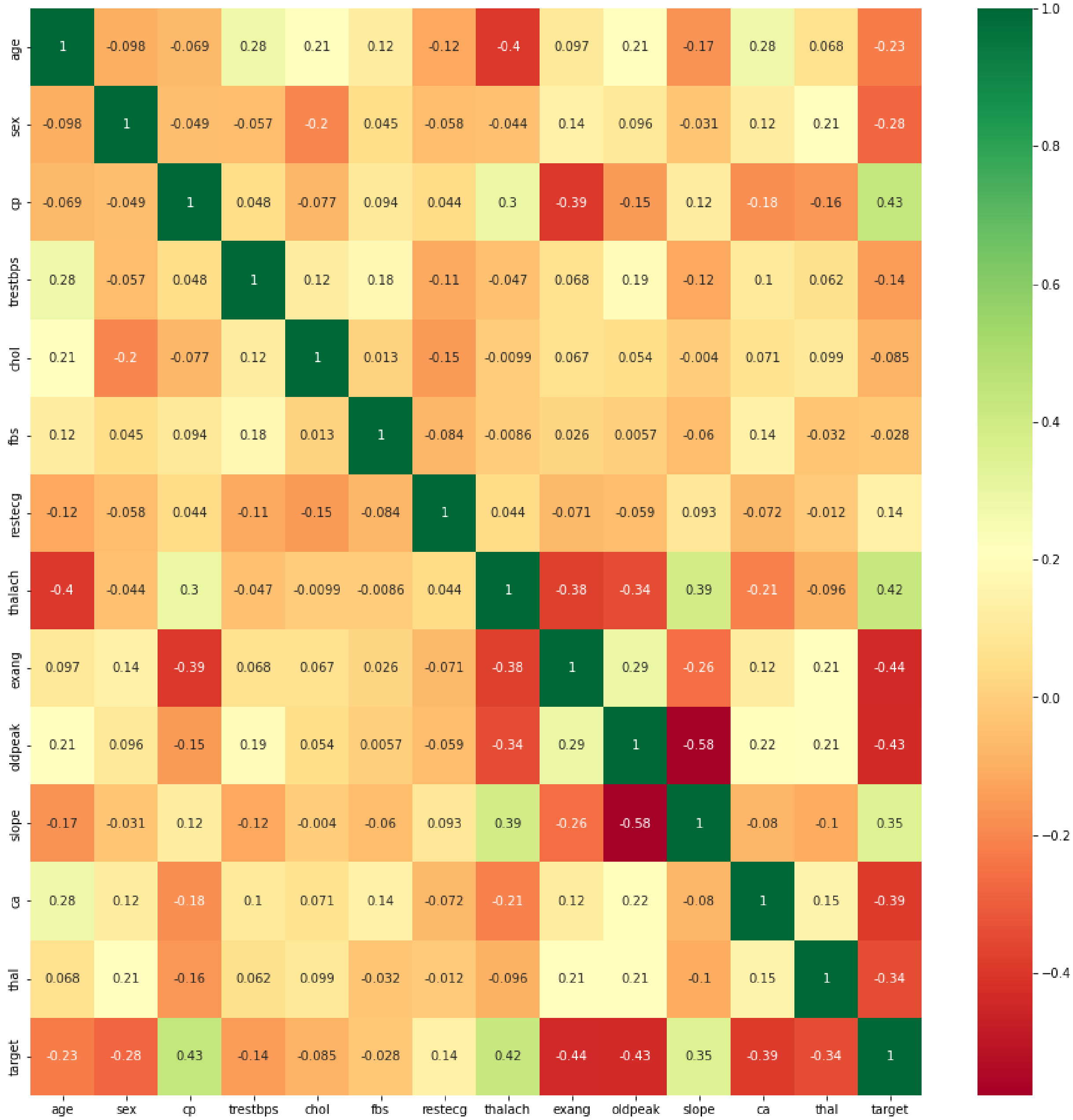
sex	0	1
age		
29	0	1
34	1	1
35	1	3
37	1	1
38	0	3

Data Visualization

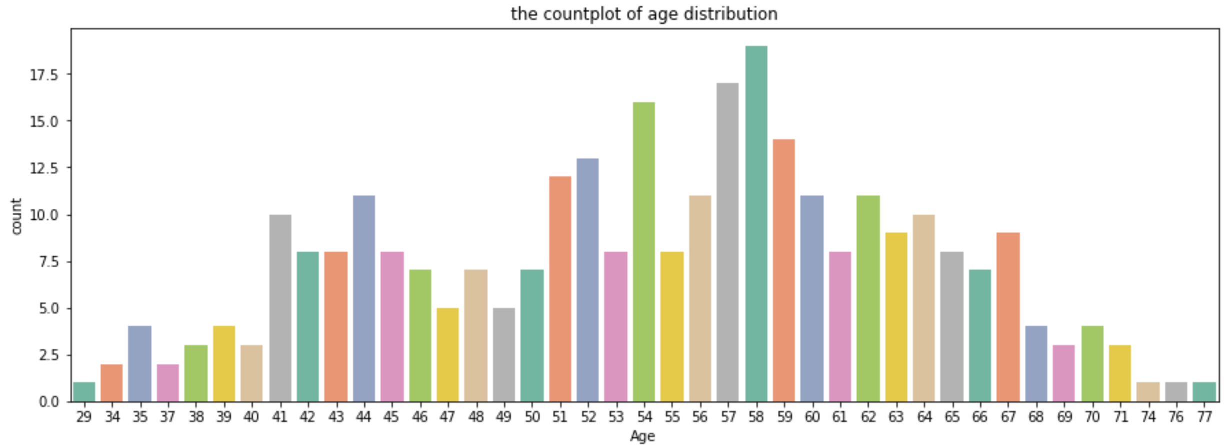
In [13]:
```
sns.pairplot(df)
```

Out[13]: <seaborn.axisgrid.PairGrid at 0x22f5b5449a0>

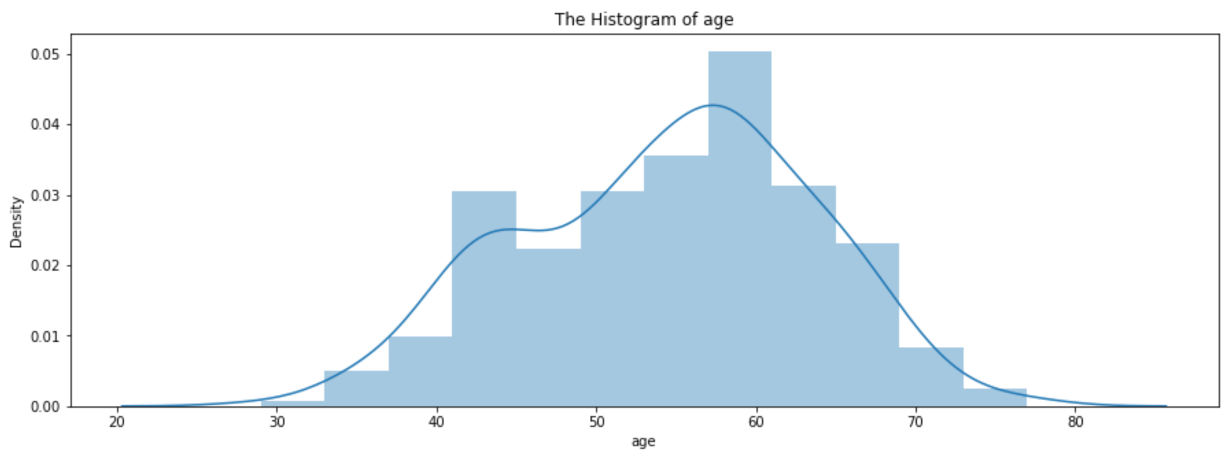
In [14]:
```
plt.figure(figsize=(16,16))
sns.heatmap(df.corr(),annot=True,cmap="RdYlGn")
plt.show()
```



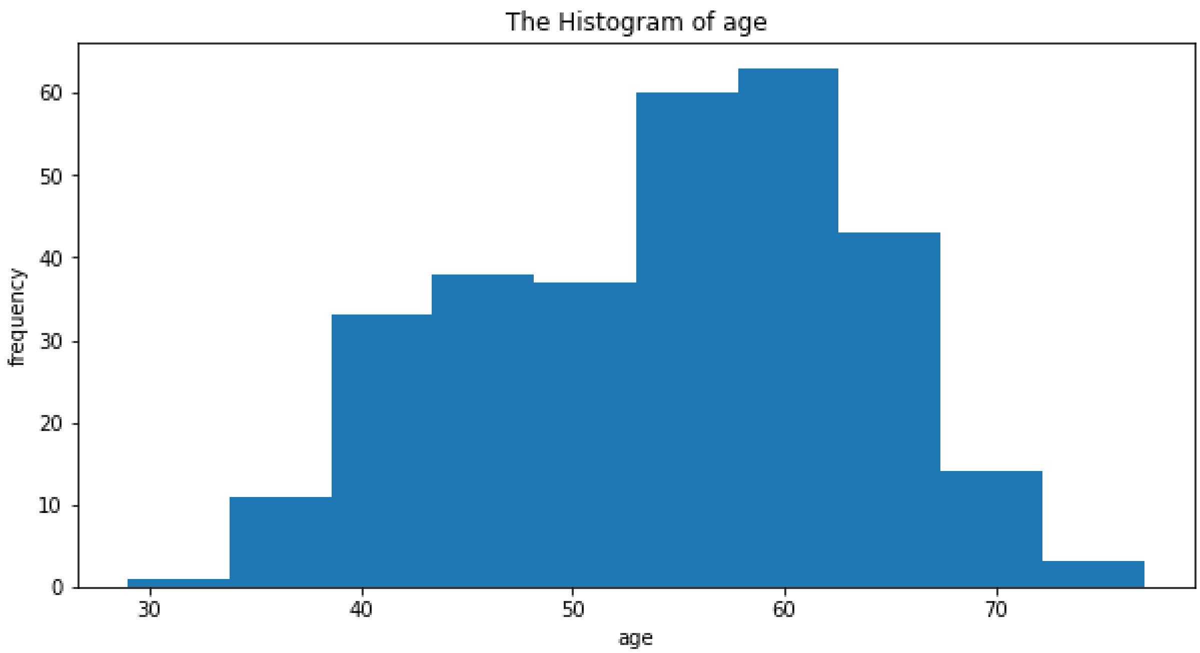
In [15]:
```
plt.figure(figsize=(15,5))
sns.countplot(df.age,palette='Set2')
plt.title('the countplot of age distribution')
plt.xlabel('Age')
plt.ylabel('count')
plt.show()
```



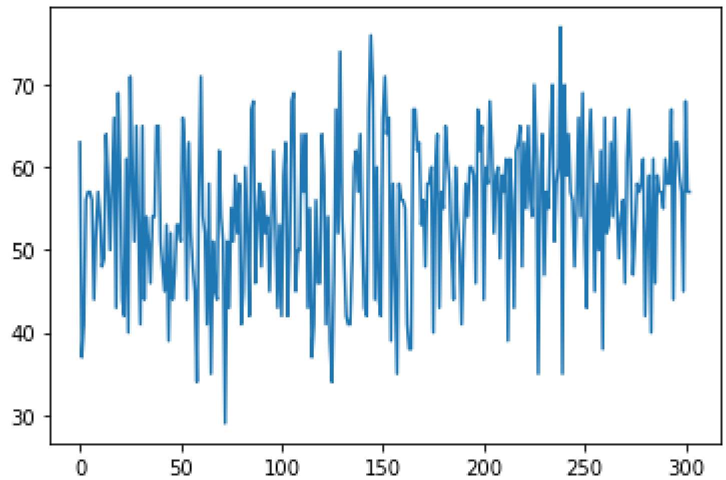
In [16]:
```
plt.figure(figsize=(15,5))
sns.distplot(df.age)
plt.title('The Histogram of age')
plt.show()
```



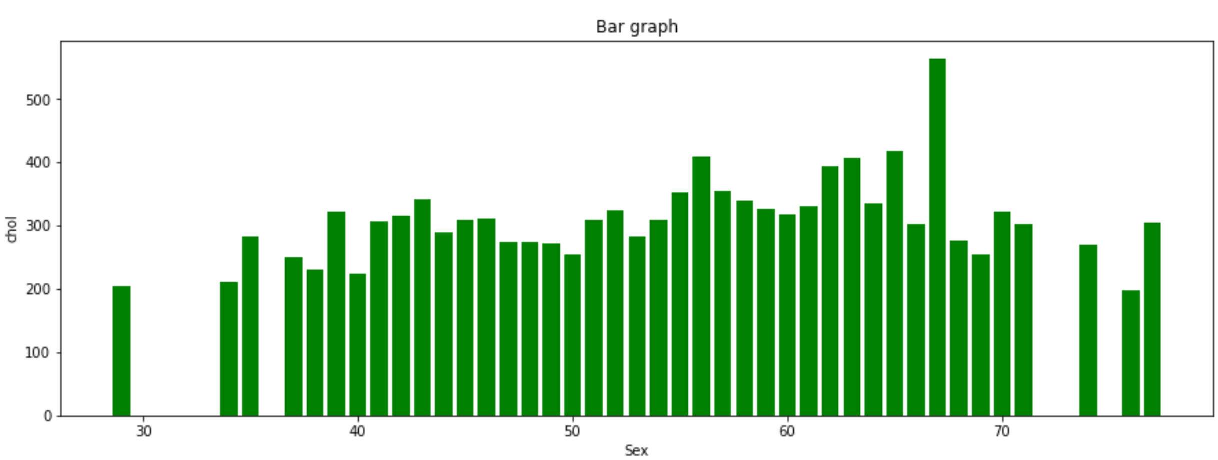
In [17]:
```
plt.figure(figsize=(10,5))
plt.hist(df.age)
plt.title('The Histogram of age')
plt.xlabel('age')
plt.ylabel('frequency')
plt.show()
```



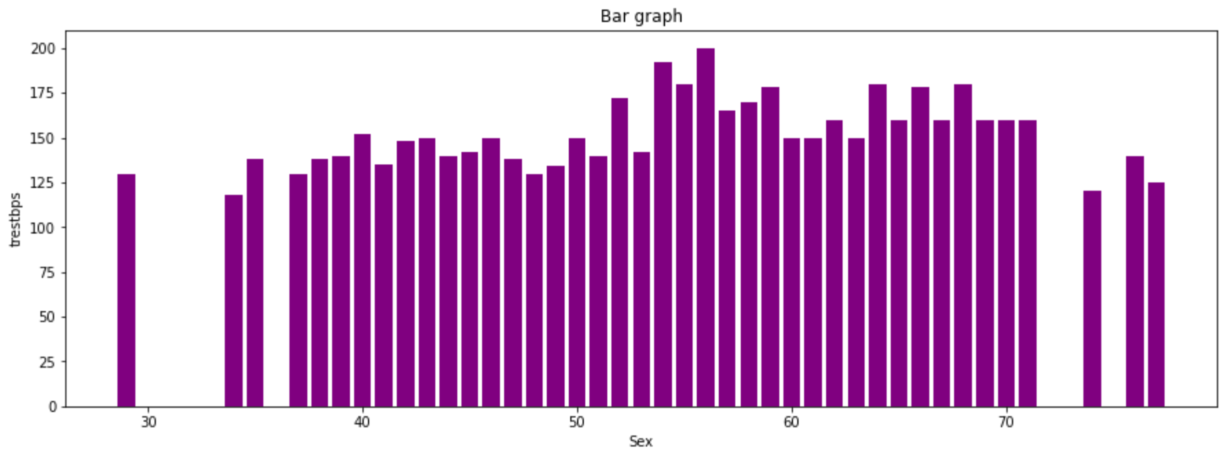
In [18]:
```
plt.plot(df.age)
plt.show()
```



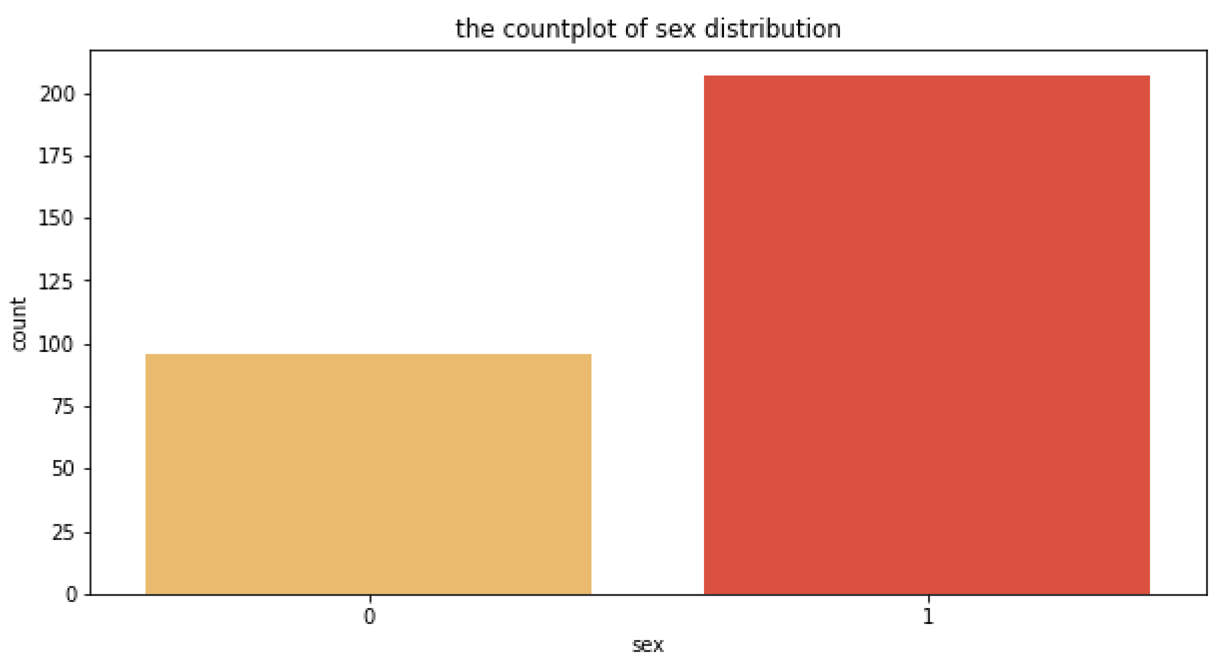
In [19]:
```
plt.figure(figsize=(15,5))
plt.bar(df.age,df.chol,color='green')
plt.xlabel('Sex')
plt.ylabel('chol')
plt.title('Bar graph')
plt.show()
```



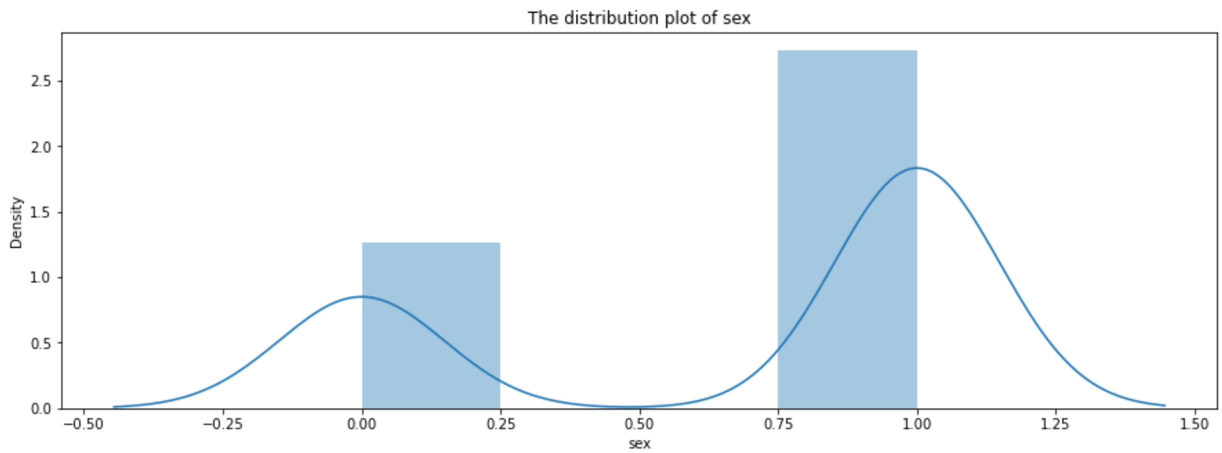
In [20]:
```
plt.figure(figsize=(15,5))
plt.bar(df.age,df.trestbps,color='purple')
plt.xlabel('Sex')
plt.ylabel('trestbps')
plt.title('Bar graph')
plt.show()
```



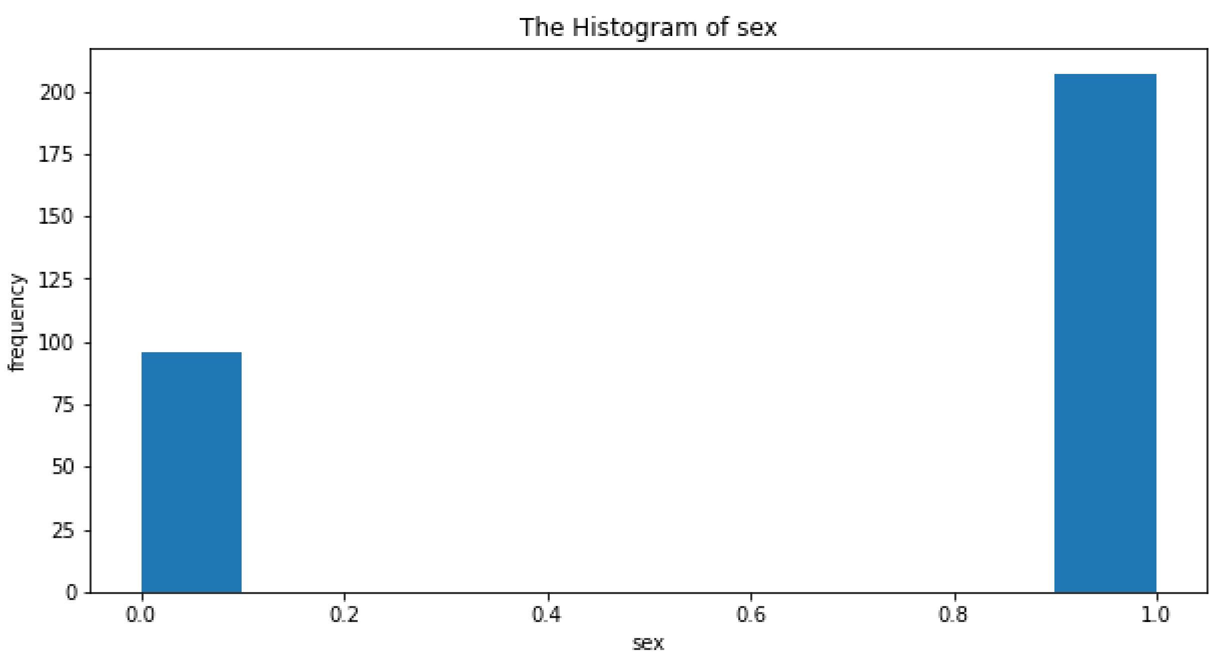
In [21]:
```
plt.figure(figsize=(10,5))
sns.countplot(df.sex,palette='YlOrRd')
plt.title('the countplot of sex distribution')
plt.show()
```



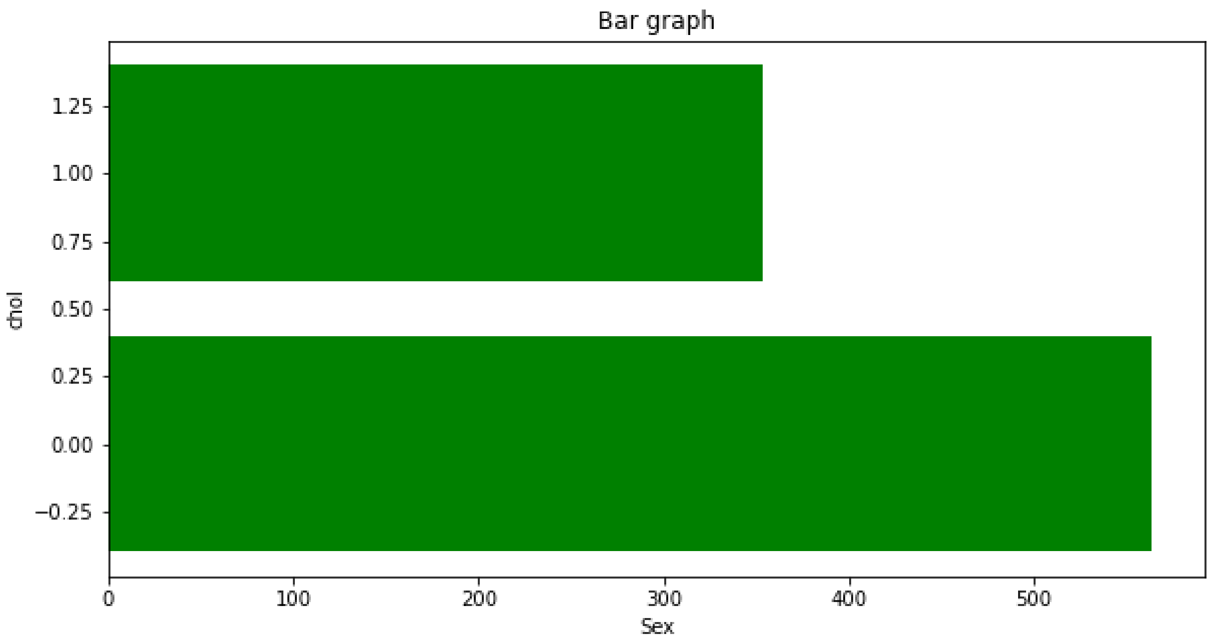
In [22]:
```
plt.figure(figsize=(15,5))
sns.distplot(df.sex)
plt.title('The distribution plot of sex')
plt.show()
```



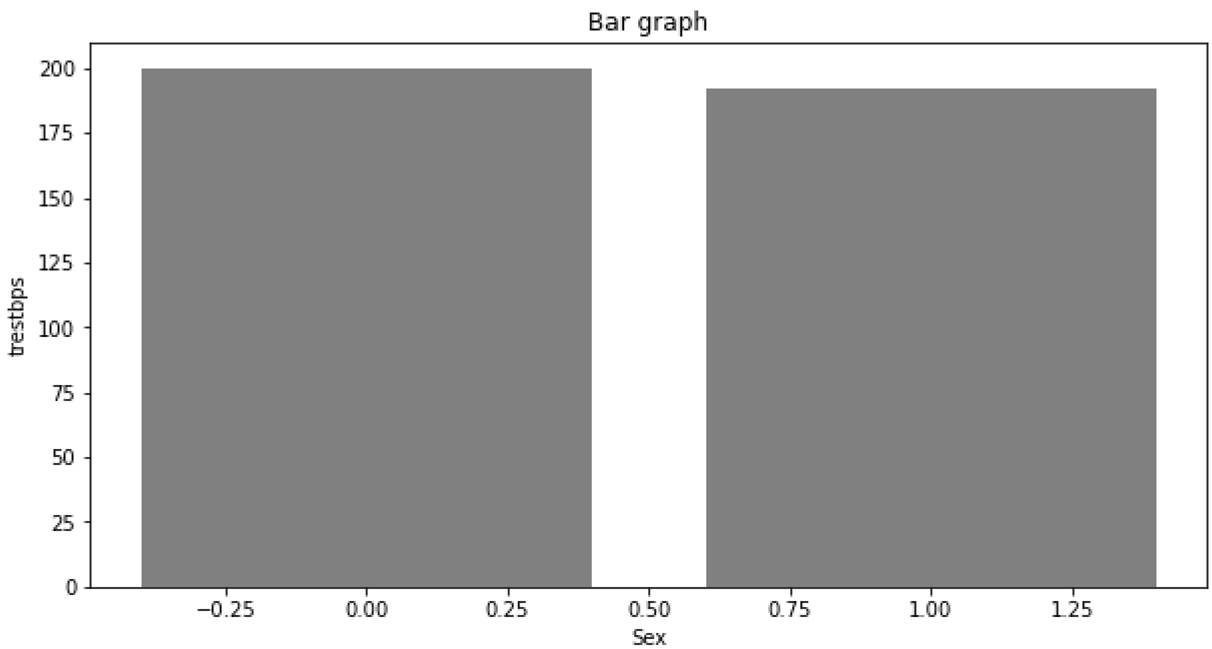
In [23]:
```
plt.figure(figsize=(10,5))
plt.hist(df.sex)
plt.title('The Histogram of sex')
plt.xlabel('sex')
plt.ylabel('frequency')
plt.show()
```



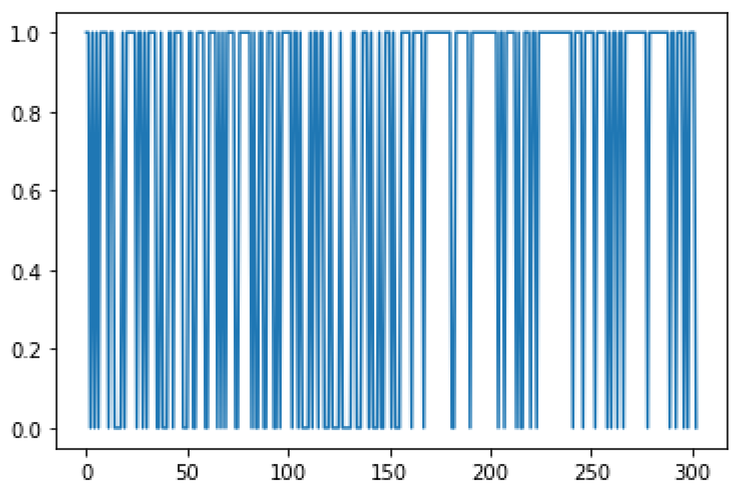
In [24]:
```
plt.figure(figsize=(10,5))
plt.barh(df.sex,df.chol,color='green')
plt.xlabel('Sex')
plt.ylabel('chol')
plt.title('Bar graph')
plt.show()
```



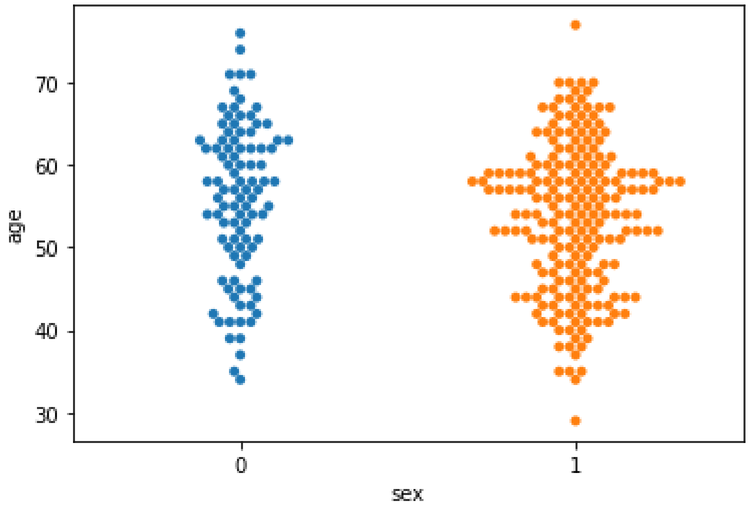
In [25]:
```
plt.figure(figsize=(10,5))
plt.bar(df.sex,df.trestbps,color='grey')
plt.xlabel('Sex')
plt.ylabel('trestbps')
plt.title('Bar graph')
plt.show()
```



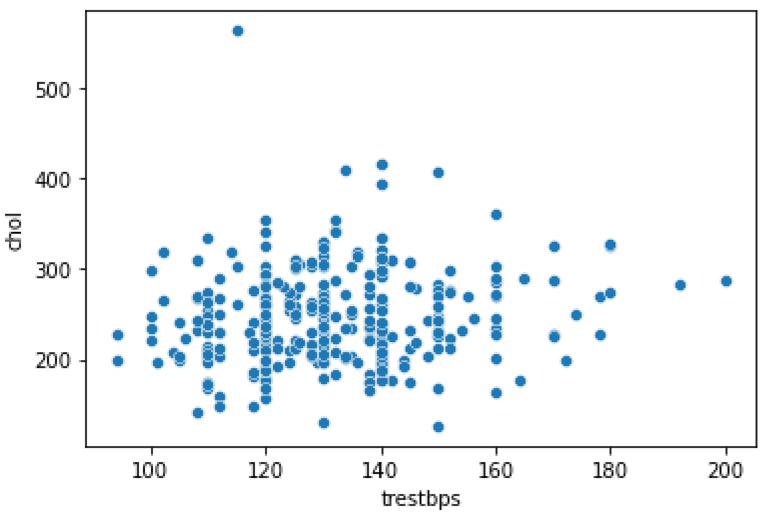
In [26]:
```
plt.plot(df.sex)
plt.show()
```



In [27]:
```
sns.swarmplot(df.sex,df.age)
plt.xlabel('sex')
plt.ylabel('age')
plt.show()
```



In [28]:
```
sns.scatterplot(df.trestbps,df.chol)
plt.show()
```



In [1]:
```
gender_size = df.groupby('sex').size()
plt.pie(gender_size, startangle=150, explode=[0.005,0],autopct='% 1.1f %%',shadow
plt.title('Pie chart for Employee sex')
plt.legend(title="sex")
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_8036/4286359145.py in <module>
----> 1 gender_size = df.groupby('insured_sex').size()
      2 plt.pie(gender_size, startangle=150, explode=[0.005,0],autopct='% 1.1f
 %%',shadow=False, labels=['Female','Male'])
      3 plt.title('Pie chart for Employee sex')
      4 plt.legend(title="sex")
      5 plt.show()

NameError: name 'df' is not defined
```

Feature engineering

In [30]:
```
df['age_category'] = ['youth' if 29<=age<=35
                              else 'adult'
                              if 36<=age<=45
                              else 'old'
                              if 46<=age<=77
                              else None
 for age in list(df.age.values)]
```

In [31]:
```
df.age_category.value_counts()
```

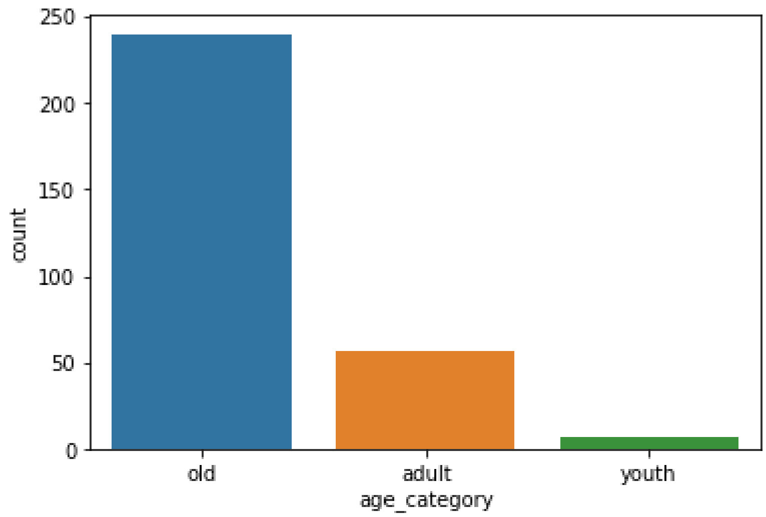
Out[31]:
```
old      239
adult     57
youth      7
Name: age_category, dtype: int64
```

In [32]:
```
pd.crosstab(df['target'], df['age_category'])
```

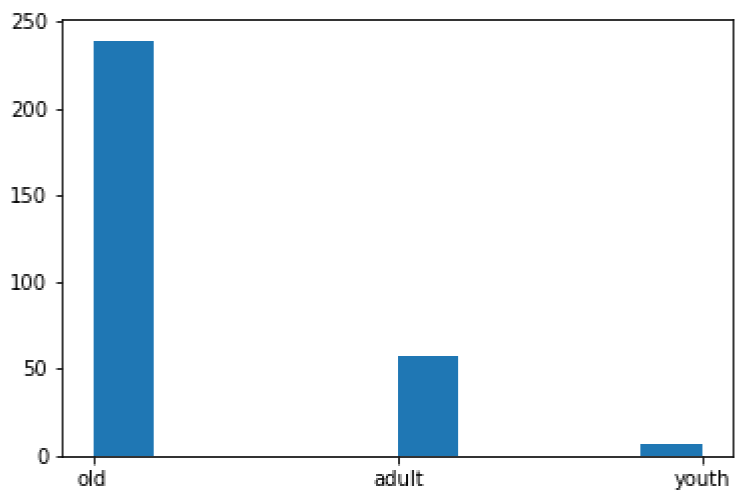
Out[32]:

age_category	adult	old	youth
target			
0	14	122	2
1	43	117	5

In [33]:
```
sns.countplot(df.age_category)
plt.show()
```

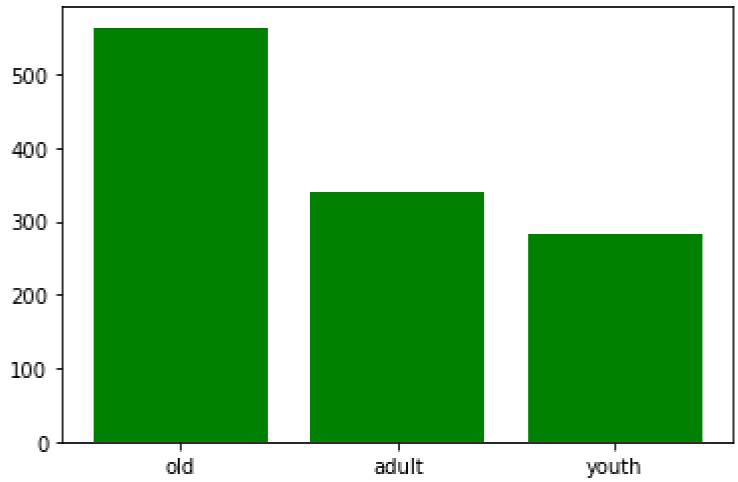


In [34]:
```
plt.hist(df.age_category)
plt.show()
```



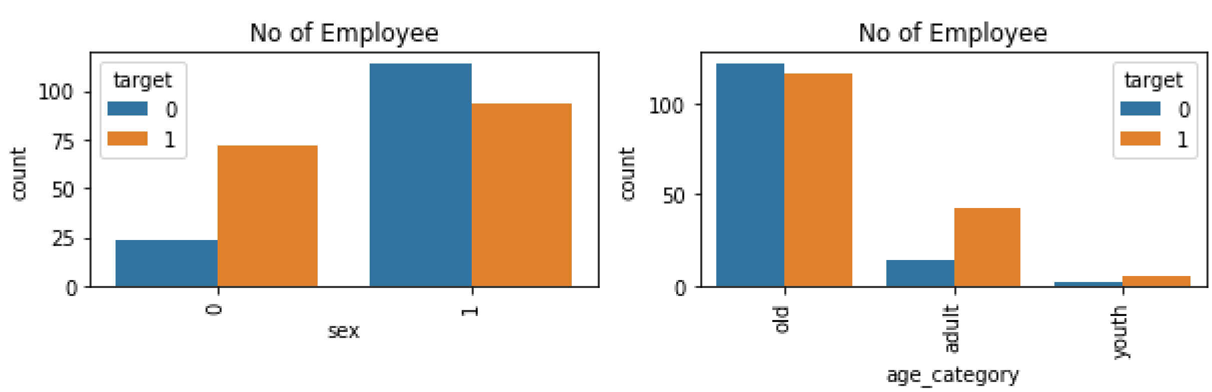
In [35]:
```
plt.bar(df.age_category,df.chol,color='green')
```

Out[35]: <BarContainer object of 303 artists>

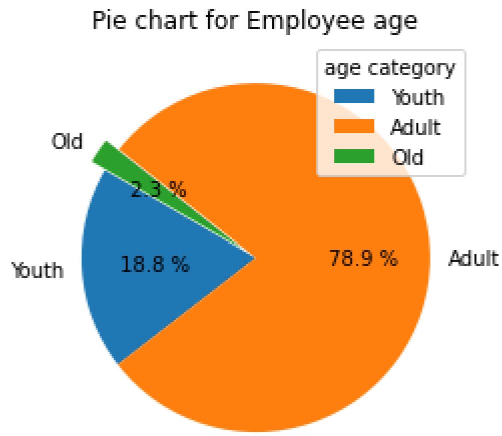


In [36]:
```
features= ['sex', 'age_category']
fig= plt.subplots(figsize= (10,15))

for i,j in enumerate(features):
    plt.subplot(4,2, i+1)
    plt.subplots_adjust(hspace=1.0)
    sns.countplot(x=j, data=df , hue= "target")
    plt.xticks(rotation= 90)
    plt.title("No of Employee")
```



In [37]:
```
age_size = df.groupby('age_category').size()
plt.pie(age_size, startangle=150, explode=[0,0,0.09],autopct='% 1.1f %%',shadow=F
plt.title('Pie chart for Employee age')
plt.legend(title="age category")
plt.show()
```



Data preprocessing

In [38]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df.age_category=le.fit_transform(df.age_category)
```

In [39]:
```
df.age_category.value_counts()
```

Out[39]:
```
1    239
0     57
2      7
Name: age_category, dtype: int64
```

In [40]:
```
dataset = pd.get_dummies(df, columns = ['sex', 'cp','fbs','restecg','exang', 'slo
dataset.head()
```

Out[40]:

	age	trestbps	chol	thalach	oldpeak	target	age_category	sex_0	sex_1	cp_0	...	slope_2
0	63	145	233	150	2.3	1	1	0	1	0	...	0
1	37	130	250	187	3.5	1	0	0	1	0	...	0
2	41	130	204	172	1.4	1	0	1	0	0	...	1
3	56	120	236	178	0.8	1	1	0	1	0	...	1
4	57	120	354	163	0.6	1	1	1	0	1	...	1

5 rows × 32 columns

Model Building

In [41]:
```
x=df.drop('target',axis=1)
y=df.target
```

In [42]:
```
x.head()
```

Out[42]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	age_categ
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	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

In [43]:
```
y.head()
```

Out[43]:
```
0    1
1    1
2    1
3    1
4    1
Name: target, dtype: int64
```

In [44]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x=scaler.fit_transform(x)
```

In [45]:
```
from sklearn.model_selection import train_test_split,cross_val_score
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

In [46]:
```
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score,
```

In [47]:
```
# Fitting Naive Bayes to the Training set
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(x_train, y_train)
```

Out[47]:
```
GaussianNB()
```

In [48]:
```
nb_pred=nb_classifier.predict(x_test)
nb_pred
```

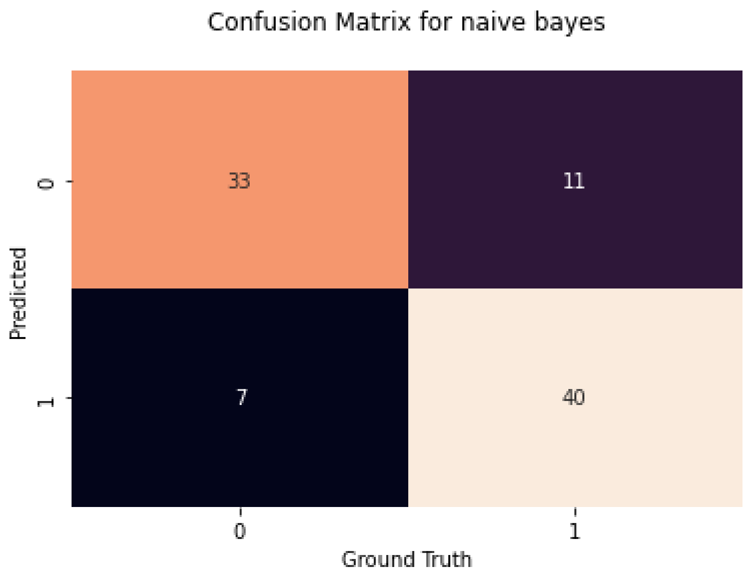
Out[48]:
```
array([0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0], dtype=int64)
```

In [49]:
```
print('Score: ',nb_classifier.score(x_test,y_test))
print('acuracy score: ',accuracy_score(y_test,nb_pred))
print('precision score: ',precision_score(y_test,nb_pred))
print('recall score: ',recall_score(y_test,nb_pred))
```

```
Score:  0.8021978021978022
acuracy score:  0.8021978021978022
precision score:  0.7843137254901961
recall score:  0.851063829787234
```

In [50]:
```
def display_confusion_matrix(test,pred,model_name =''):
    confmatrix= confusion_matrix(test, pred)
    ax = plt.subplot()
    sns.heatmap(confmatrix, annot=True, ax=ax, cbar=False)
    plt.title('Confusion Matrix for '+str(model_name) + '\n')
    ax.set_xlabel('Ground Truth')
    ax.set_ylabel('Predicted')
    ax.xaxis.set_ticklabels(['0','1'])
    ax.yaxis.set_ticklabels(['0','1'])
    plt.show()
```

In [51]:
```
display_confusion_matrix(y_test,nb_pred, 'naive bayes')
```



In [52]:
```
pd.Series(y_test).value_counts()
```

Out[52]:
```
1    47
0    44
Name: target, dtype: int64
```

In [53]:
```
# Fitting Logistic Regression to the Training set
from sklearn.svm import SVC
sv_classifier = SVC()
sv_classifier.fit(x_train, y_train)
```

Out[53]:
```
SVC()
```

In [54]:
```
sv_pred=sv_classifier.predict(x_test)
```

In [55]:
```
print('Score: ',sv_classifier.score(x_test,y_test))
print('accuracy score: ',accuracy_score(y_test,sv_pred))
print('precision score: ',precision_score(y_test,sv_pred))
print('recall score: ',recall_score(y_test,sv_pred))
print('f1 score: ',f1_score(y_test,sv_pred))
print('\n')
print('classification report: ',classification_report(y_test,sv_pred))
```

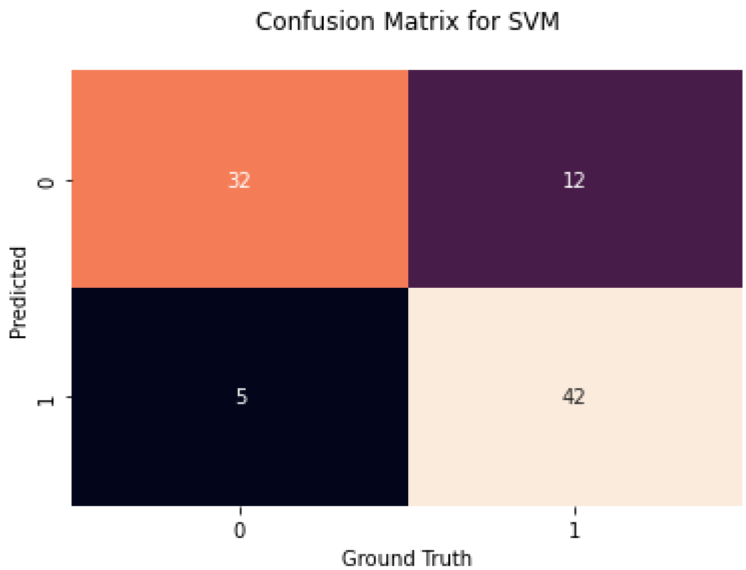
```
Score:  0.8131868131868132
accuracy score:  0.8131868131868132
precision score:  0.7777777777777778
recall score:  0.8936170212765957
f1 score:  0.8316831683168316


classification report:                precision    recall  f1-score   support

           0       0.86      0.73      0.79        44
           1       0.78      0.89      0.83        47

    accuracy                           0.81        91
   macro avg       0.82      0.81      0.81        91
weighted avg       0.82      0.81      0.81        91
```

In [56]:
```
display_confusion_matrix(y_test,sv_pred, 'SVM')
```



In [57]:
```
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
rfc_classifier = RandomForestClassifier()
rfc_classifier.fit(x_train, y_train)
```

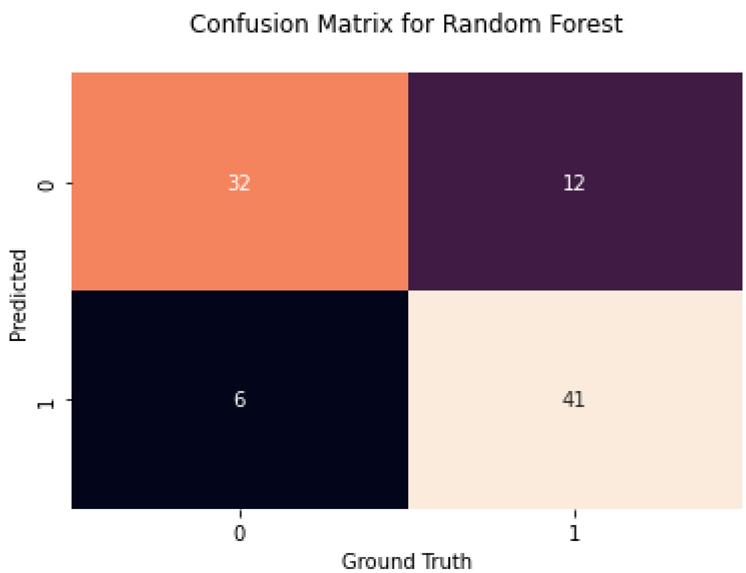
Out[57]: RandomForestClassifier()

In [58]:
```
rfc_pred=rfc_classifier.predict(x_test)
```

In [59]:
```
print('Score: ',rfc_classifier.score(x_test,y_test))
print('accuracy score: ',accuracy_score(y_test,rfc_pred))
print('precision score: ',precision_score(y_test,rfc_pred))
print('recall score: ',recall_score(y_test,rfc_pred))
```

```
Score:  0.8021978021978022
accuracy score:  0.8021978021978022
precision score:  0.7735849056603774
recall score:  0.8723404255319149
```

In [60]:
```
display_confusion_matrix(y_test,rfc_pred, 'Random Forest')
```



In [61]:
```
from sklearn.neighbors import KNeighborsClassifier

knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(x_train, y_train)
```

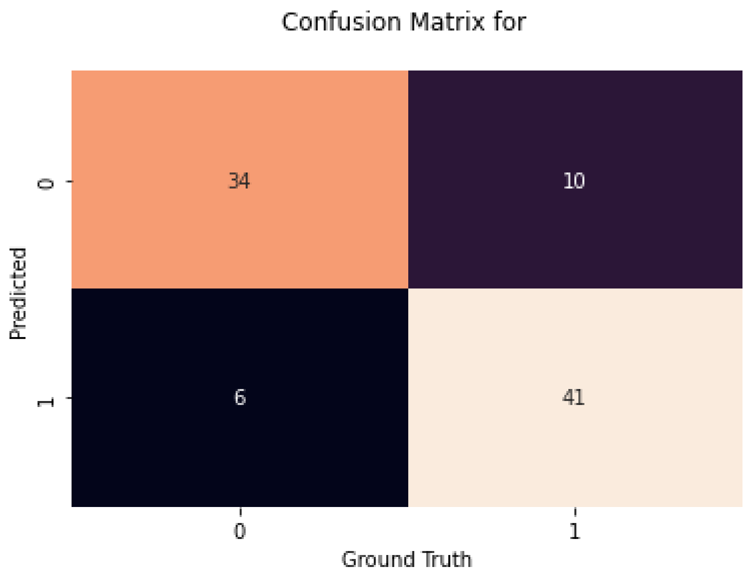
Out[61]: KNeighborsClassifier()

In [62]:
```
knn_pred=knn_classifier.predict(x_test)
```

In [63]:
```
print('Score: ',rfc_classifier.score(x_test,y_test))
print('accuracy score: ',accuracy_score(y_test,knn_pred))
print('precision score: ',precision_score(y_test,knn_pred))
print('recall score: ',recall_score(y_test,knn_pred))
```

```
Score:  0.8021978021978022
accuracy score:  0.8241758241758241
precision score:  0.803921568627451
recall score:  0.8723404255319149
```

In [64]:
```
display_confusion_matrix(y_test,knn_pred)
```



In [65]:
```
from xgboost import XGBClassifier
xgb=XGBClassifier()
xgb.fit(x_train,y_train)
```

```
[14:38:39] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the defa
ult evaluation metric used with the objective 'binary:logistic' was changed fro
m 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the
old behavior.
```

Out[65]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=4,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [66]:
```
xgb_pred=xgb.predict(x_test)
```

In [67]:
```
print('Score: ',xgb.score(x_test,y_test))
print('accuracy score: ',accuracy_score(y_test,xgb_pred))
print('precision score: ',precision_score(y_test,xgb_pred))
print('recall score: ',recall_score(y_test,xgb_pred))
```

```
Score:  0.8021978021978022
accuracy score:  0.8021978021978022
precision score:  0.7735849056603774
recall score:  0.8723404255319149
```

In [68]:
```
from catboost import CatBoostClassifier
cat=CatBoostClassifier()
cat.fit(x_train,y_train)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_9816/4008524781.py in <module>
----> 1 from catboost import CatBoostClassifier
      2 cat=CatBoostClassifier()
      3 cat.fit(x_train,y_train)

ModuleNotFoundError: No module named 'catboost'
```

In [76]:
```
cat_pred=cat.predict(x_test)
```

In [77]:
```
print('Score: ',cat.score(x_test,y_test))
print('accuracy score: ',accuracy_score(y_test,cat_pred))
print('precision score: ',precision_score(y_test,cat_pred))
print('recall score: ',recall_score(y_test,cat_pred))
```

```
Score:  0.8241758241758241
accuracy score:  0.8241758241758241
precision score:  0.7924528301886793
recall score:  0.8936170212765957
```

In [69]:
```
import pickle
```

In [72]:
```

pkl_model=open('rfheart.pkl','wb')
pickle.dump(rfc_classifier,pkl_model)
```

In [73]:
```
model=pickle.load(open('heart.pkl','rb'))
print(model.predict([[63,1,3,145,233,1,0,150,0,2.3,0,0,1,1]]))
```

```
[0]
```

In [74]:
```
sv_model=open('svheart.pkl','wb')
pickle.dump(sv_classifier,sv_model)
```

In [75]:
```
model=pickle.load(open('svheart.pkl','rb'))
print(model.predict([[63,1,3,145,233,1,0,150,0,2.3,0,0,1,1]]))
```

```
[1]
```

In [70]:
```
xg_model=open('xheart.pkl','wb')
pickle.dump(xgb,xg_model)
```

In [75]:
```
kn_model=open('knheart.pkl','wb')
pickle.dump(knn_classifier,kn_model)
```

In [ ]: