In [1]:
```
# importing required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Loading the dataset
df=pd.read_excel('data.xlsx')
```

Preliminery data inspection

In [3]:
```
# Printing the top 5 rows
df.head()
```

Out[3]:

	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 [4]:
```
# checking the dimension of the dataset
df.shape
```

Out[4]: (303, 14)

In [5]:
```
# understandong the dataset and checking the null values for each column
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
```

we have 14 variables in total and thirteen of them are integer data type and one float datatype. from the above information it is clear that there are no missing values, hence no action need to be taken regarding the missing values.

In [51]:
```
#cheching missing values
df.isnull().sum()
```

Out[51]:
```
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 [6]:
```
duplicates= df[df.duplicated()]
duplicates
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

In [7]:
```
df.drop_duplicates(inplace=True)
```

In [8]:
```
df.shape
```

Out[8]: (302, 14)

the duplicate row is been removed and now we have 302 rows

Preliminery statistical summary

In [9]:
```
df.describe()
```

Out[9]:

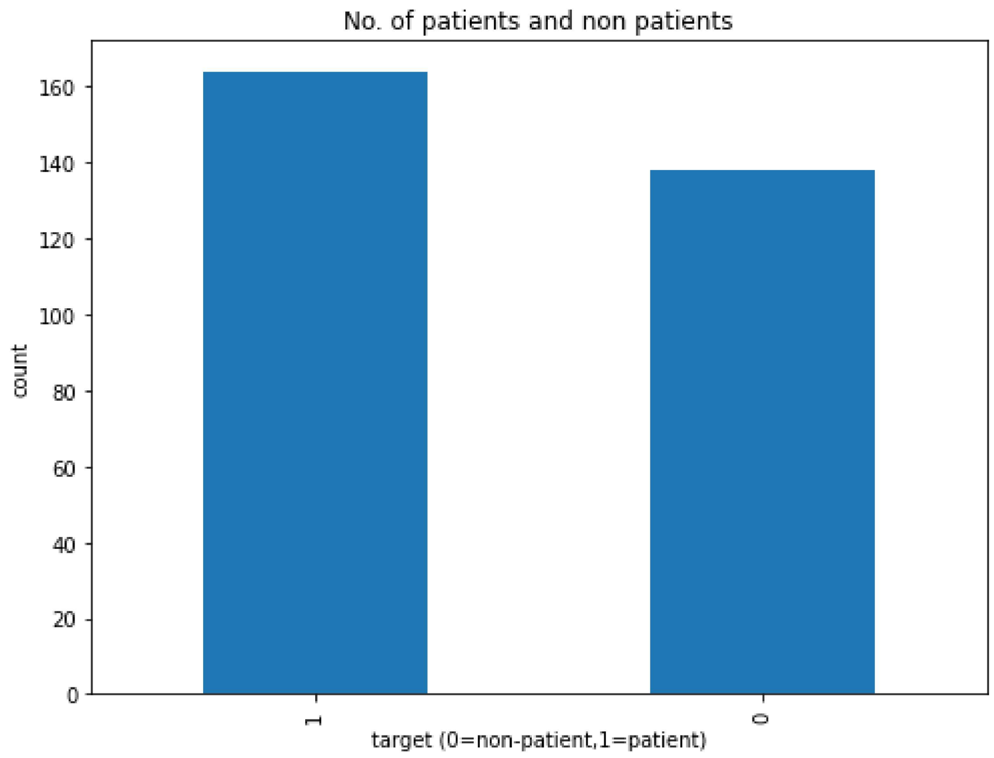
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.0(
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.7
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.0(
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.0(
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.0(
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.0(
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.0(
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.0(

The mean age is 54.42 where the minimum age is 29 and themaximum is 77. The statistcal summary of all the 14 variables can be analysed using the above table.

EDA - Exploratory Data Analysis

number of patients and non patients

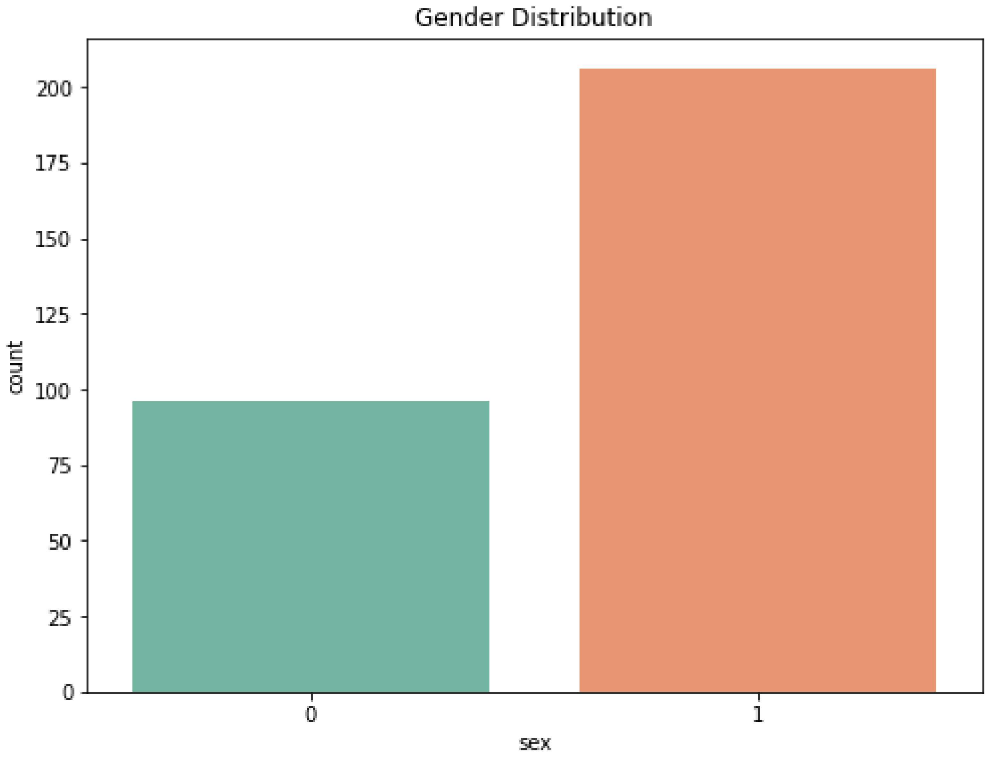
In [10]:
```
plt.figure(figsize=(8,6))
df['target'].value_counts().plot(kind='bar',title=("No. of patients and non patients"))
plt.xlabel("target (0=non-patient,1=patient)")
plt.ylabel("count")
plt.show()
```



This bar graph clearly shows the number of patients and non patients in our study. the number of patients is higher than the non patients

Gender distribution

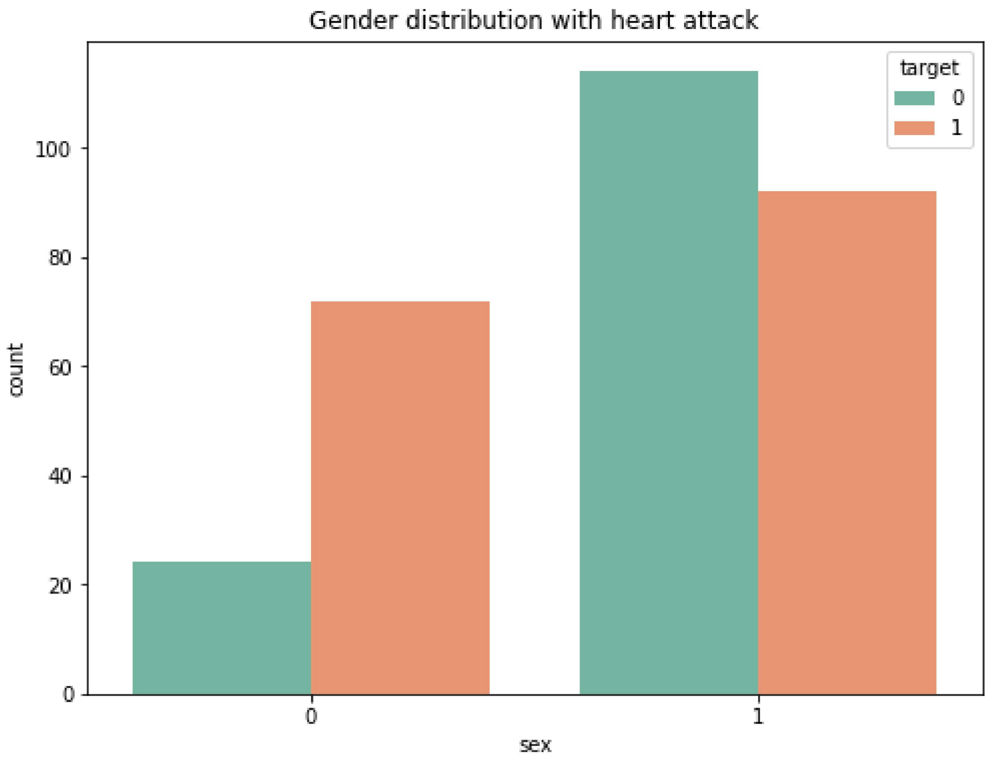
In [11]:
```
plt.figure(figsize=(8,6))
plt.title("Gender Distribution")
sns.countplot(x=df['sex'],palette='Set2')
plt.show()
```



This bargraph indicates that the number of males in our study is higher ompared to the females.The number of males is more than 200 where the number of womens is nearly 100.

Gender with heart attack

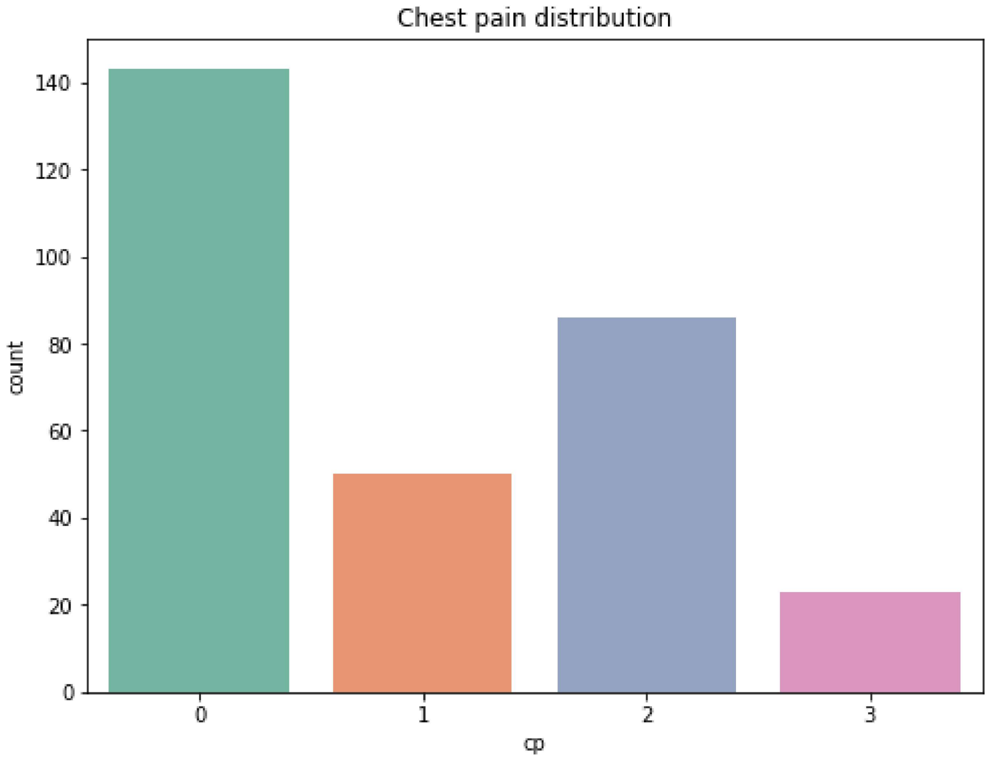
In [14]:
```
plt.figure(figsize=(8,6))
sns.countplot(x=df["sex"],hue="target",data=df,palette="Set2")
plt.title("Gender distribution with heart attack")
plt.show()
```



The graph shows that the women have a higher ratio of heart attack in comparison whith the males,

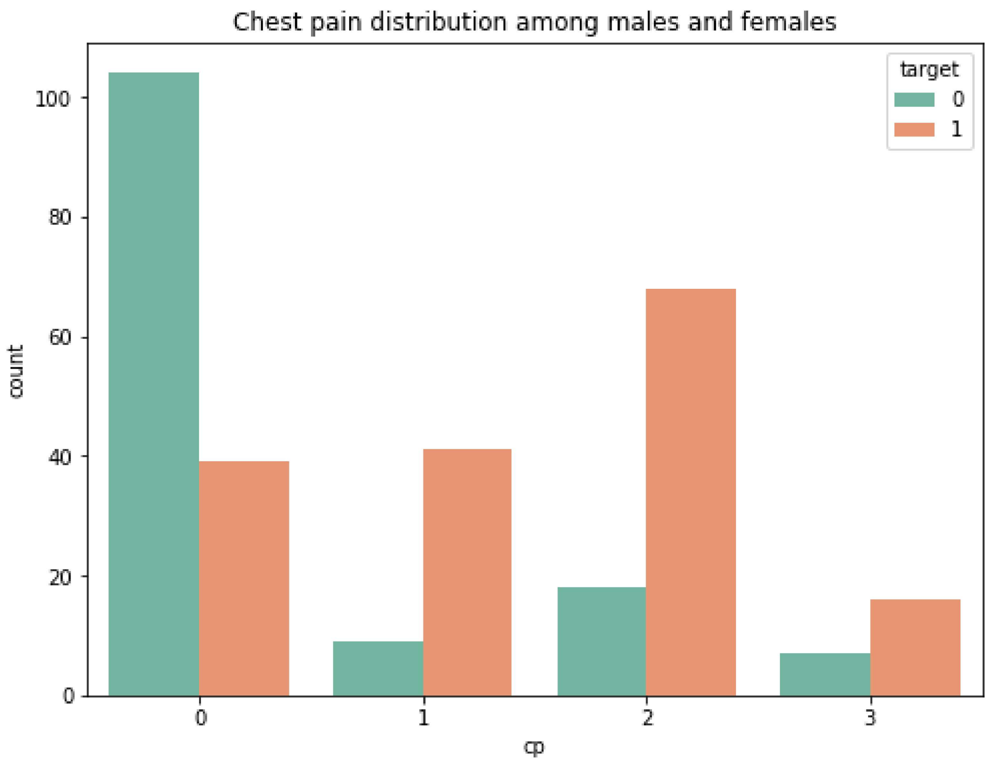
Chest Pain Distribution

In [15]:
```
plt.figure(figsize=(8,6))
sns.countplot(x=df['cp'],palette="Set2")
plt.title("Chest pain distribution")
plt.show()
```



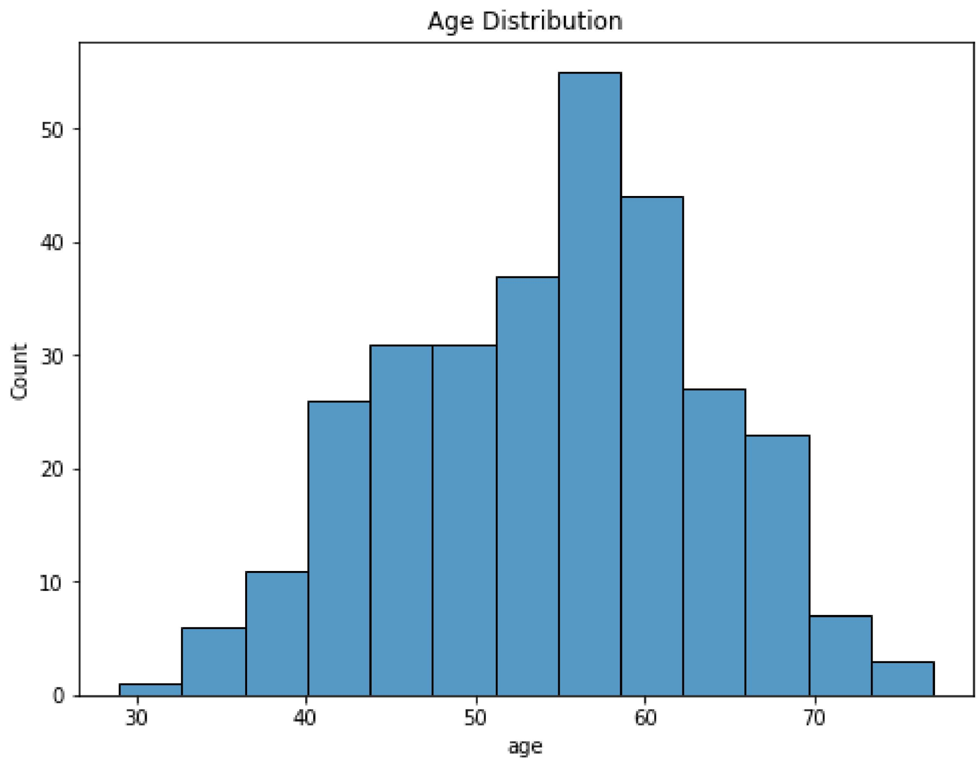
Chest Pain distribution among males and females

In [17]:
```
plt.figure(figsize=(8,6))
sns.countplot(x=df['cp'],hue="target",palette="Set2",data=df)
plt.title("Chest pain distribution among males and females")
plt.show()
```



Distribution of age

In [19]:
```
plt.figure(figsize=(8,6))
sns.histplot(x=df["age"])
plt.title("Age Distribution")
plt.show()
```



The above graph represents the age distribution where we can identify that people aged between 50 and 60 are higher in our study

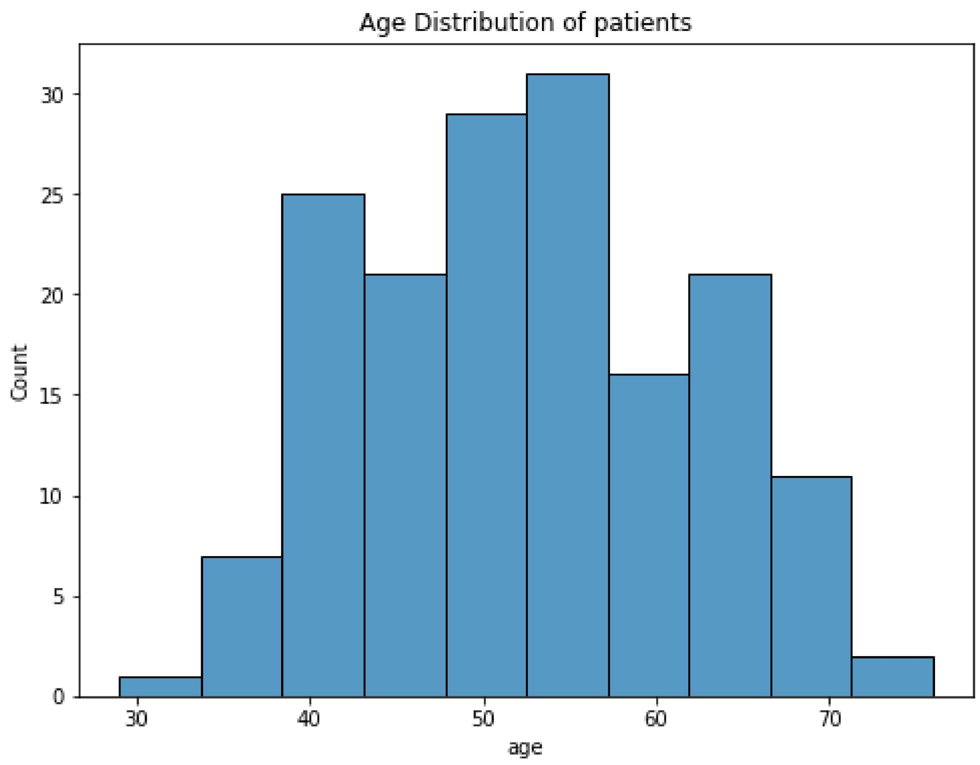
Lets find out which age group have more patients with

In [22]:
```
patients=df[df["target"]==1]
```

In [23]:
```
patients.shape
```

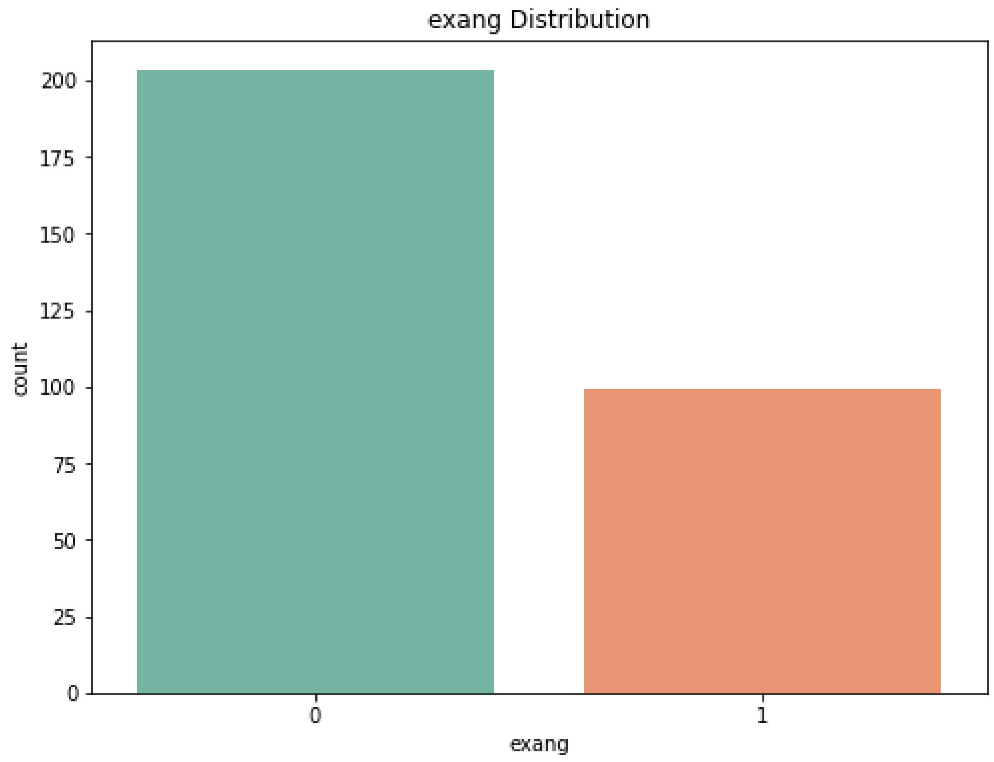
Out[23]:
```
(164, 14)
```

In [25]:
```
plt.figure(figsize=(8,6))
sns.histplot(x=patients["age"])
plt.title("Age Distribution of patients")
plt.show()
```



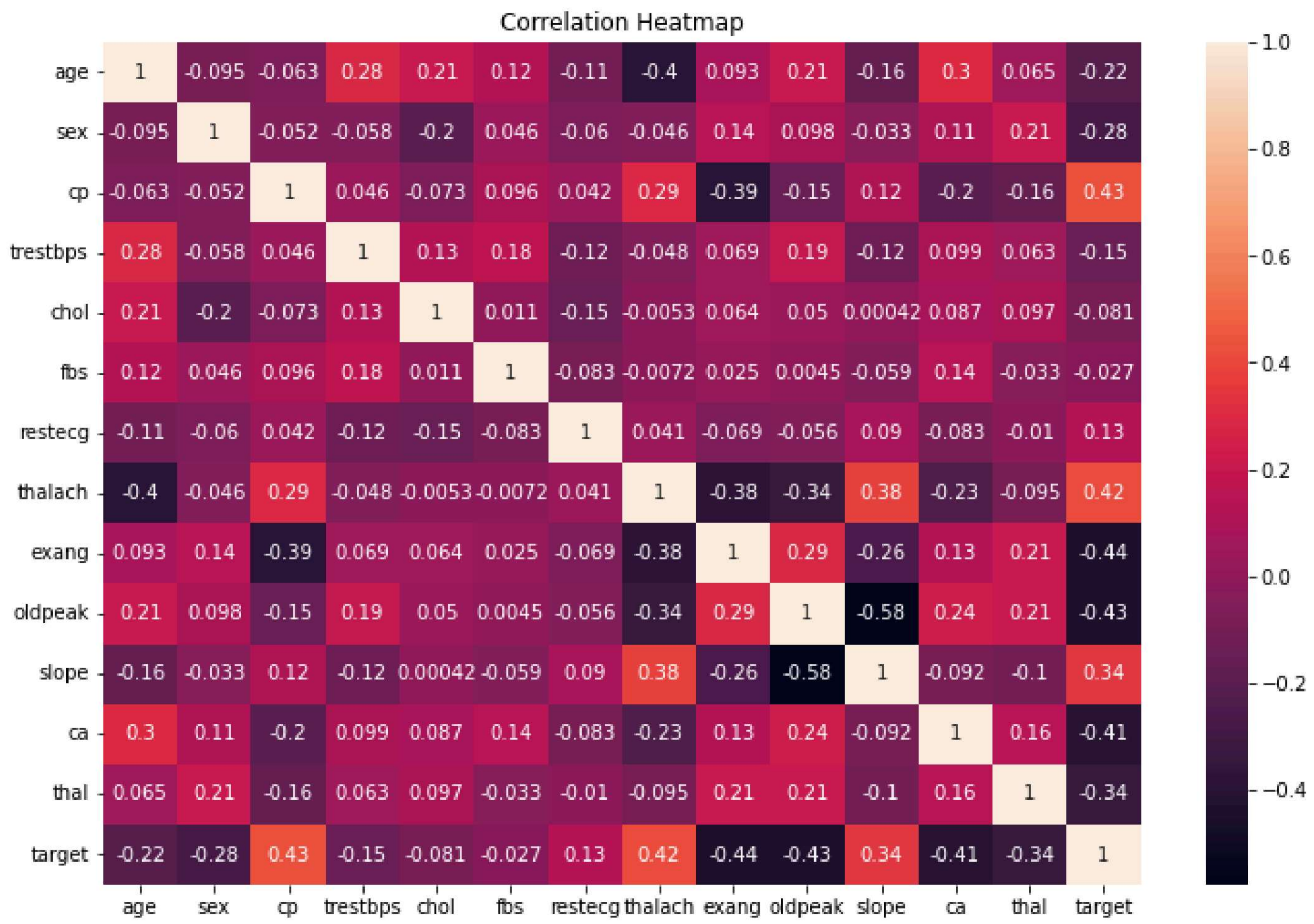
there are more patients with people aged between 45 and 55

In [27]:
```python
plt.figure(figsize=(8,6))
sns.countplot(x=df['exang'],palette='Set2')
plt.title('exang Distribution')
plt.show()
```



creating a correlation metrics

In [28]:
```python
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True)
plt.title('Correlation Heatmap')
plt.show()
```



The above heatmap shows the degree of relationship between all the variables in the dataset.where 1 indicates a perfect possitive correlation and -1 indicates a perfect negative correlation. Chest pain and target have a correlation of 0.43.

Pairplot for continuous variable

In [35]:
```python
df.columns
```

Out[35]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

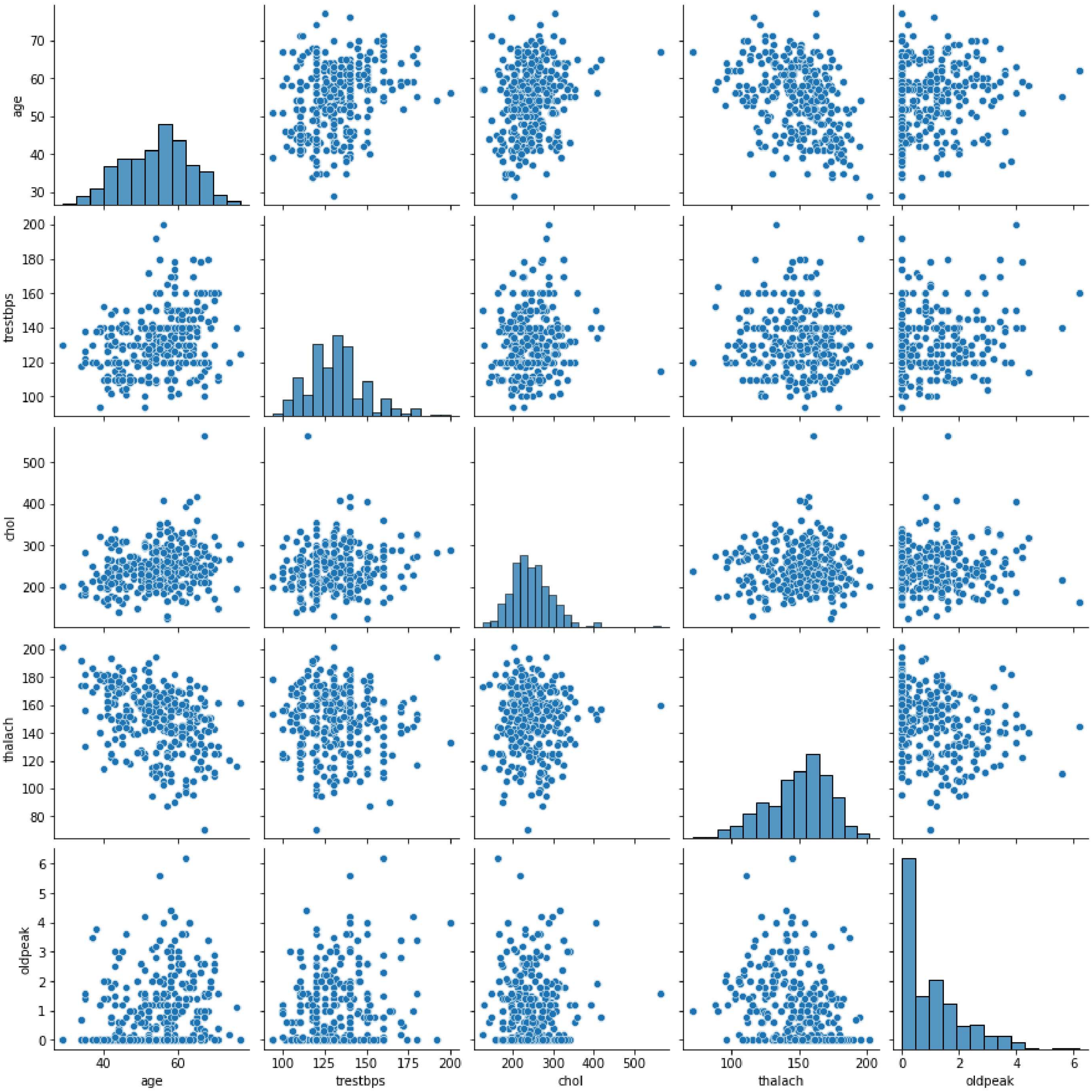
In [43]:
```
df1 = df[['age', 'trestbps', 'chol', 'thalach', 'oldpeak']]
df1.head()
```

Out[43]:

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

In [49]:
```
plt.figure(figsize=(12,8))
sns.pairplot(data=df1)
plt.show()
```

<Figure size 864x576 with 0 Axes>



In [ ]:

Building a machine learning model

Train test split

In [79]:
```
from sklearn.model_selection import train_test_split
X=df.drop('target',axis=1)
y=df['target']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

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

Out[80]:
```
(241, 13)
```

In [81]:
```
X_test.shape
```

Out[81]:
```
(61, 13)
```

The data is divided into 80% for training and 20 % for testing

In [82]:
```
X_train
```

Out[82]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
132	42	1	1	120	295	0	1	162	0	0.0	2	0	2
203	68	1	2	180	274	1	0	150	1	1.6	1	0	3
197	67	1	0	125	254	1	1	163	0	0.2	1	2	3
75	55	0	1	135	250	0	0	161	0	1.4	1	0	2
177	64	1	2	140	335	0	1	158	0	0.0	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
189	41	1	0	110	172	0	0	158	0	0.0	2	0	3
71	51	1	2	94	227	0	1	154	1	0.0	2	1	3
106	69	1	3	160	234	1	0	131	0	0.1	1	1	2
271	61	1	3	134	234	0	1	145	0	2.6	1	2	2
102	63	0	1	140	195	0	1	179	0	0.0	2	2	2

241 rows × 13 columns

applying a standard scaler to make all the feature in a similar sclae

In [83]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)  # Learning the scaling parameters and applying to X_train
X_test = sc.transform(X_test)        # Applying the scaling parameters learned from X_train to X_test
```

In [84]:
```
X_train
```

Out[84]:
```
array([[-1.350641  ,  0.73145871,  0.        , ...,  0.96543644,
        -0.68348955, -0.54576155],
       [ 1.48742624,  0.73145871,  0.9664929 , ..., -0.68470669,
        -0.68348955,  1.14050171],
       [ 1.37826981,  0.73145871, -0.9664929 , ..., -0.68470669,
         1.35010281,  1.14050171],
       ...,
       [ 1.59658267,  0.73145871,  1.9329858 , ..., -0.68470669,
         0.33330663, -0.54576155],
       [ 0.72333121,  0.73145871,  1.9329858 , ..., -0.68470669,
         1.35010281, -0.54576155],
       [ 0.94164408, -1.36713116,  0.        , ...,  0.96543644,
         1.35010281, -0.54576155]])
```

In [85]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[85]:
```
LogisticRegression()
```

In [86]:
```
# Model evaluation
y_pred = model.predict(X_test)
```

In [76]:
```
y_pred
```

Out[76]:
```
array([0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1], dtype=int64)
```

In [88]:
```
y_test.values
```

Out[88]:
```
array([0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

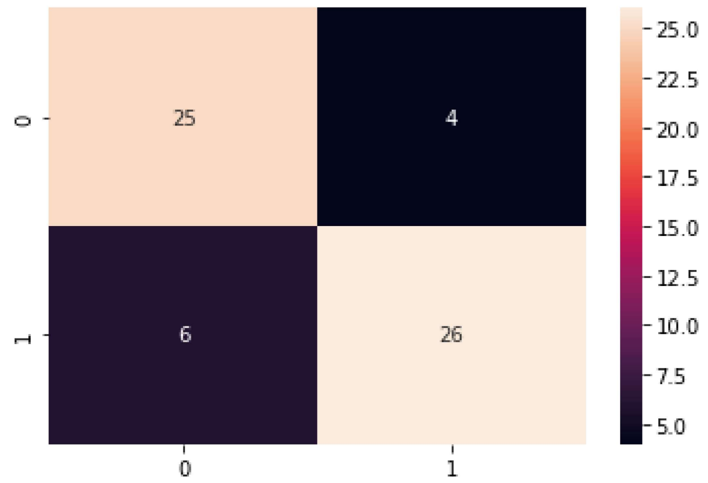
Calculating the accuracy of the model

In [89]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
```

```
[[25  4]
 [ 6 26]]
```

In [90]:
```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d")
```

Out[90]:
```
<AxesSubplot:>
```



In [92]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_pred))
```

```
0.8360655737704918
```

The accuracy score of this model is 0.83 which indicates that the model predicts the value with an 83% accuracy

In [ ]: