In [5]:
```python
import pandas as pd
import numpy  as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Step-1:: Reading Data

In [6]:
```python
data=pd.read_csv('C:/Users/malleswari/Desktop/heart_attack.csv')
data.head()
```

Out[6]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
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

Step-2:: Shape of Original Data

Before deleting null values and duplicate values the size of the original data frame is

In [7]:
```python
data.shape
```

Out[7]:
```
(303, 14)
```

Step-3:: Handle NULL values and DUPLICATE values in the Dataset

1.checking whether the dataset contain null values or not .

If it contains null values then simply remove it or modify it with the mean/median in the case of numerical values. In case of categorical values replace null values with the most frequently occuring value

In [8]:
```python
data.isna().sum()
```

Out[8]:
```
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

Here there is no null values present in this dataset

2.checking duplicate values in dataset

If it contains duplicate values then simply remove that duplicate values from the dataset

In [9]:
```python
data.duplicated()
```

Out[9]:
```
0      False
1      False
2      False
3      False
4      False
       ...
298    False
299    False
300    False
301    False
302    False
Length: 303, dtype: bool
```

In [10]:
```
data.duplicated().sum()
```
Out[10]:
```
1
```

Here one duplicate data is present. So simply remove that duplicate data from the dataset

Deleting duplicate values

In [11]:
```
data=data.drop_duplicates()
```

In [12]:
```
data.duplicated().sum()
```
Out[12]:
```
0
```

After deleting Null Values and Duplicated values the size of data set is

In [13]:
```
data.shape
```
Out[13]:
```
(302, 14)
```

Step-4:: Knowing Type of Data present in the dataset

In [14]:
```
data.dtypes
```
Out[14]:
```
age           int64
sex           int64
cp            int64
trtbps        int64
chol          int64
fbs           int64
restecg       int64
thalachh      int64
exng          int64
oldpeak     float64
slp           int64
caa           int64
thall         int64
output        int64
dtype: object
```

Here there is no categorical type of data and all 13 columns are int type and one column if of float type.

In [15]:
```
data.head()
```
Out[15]:

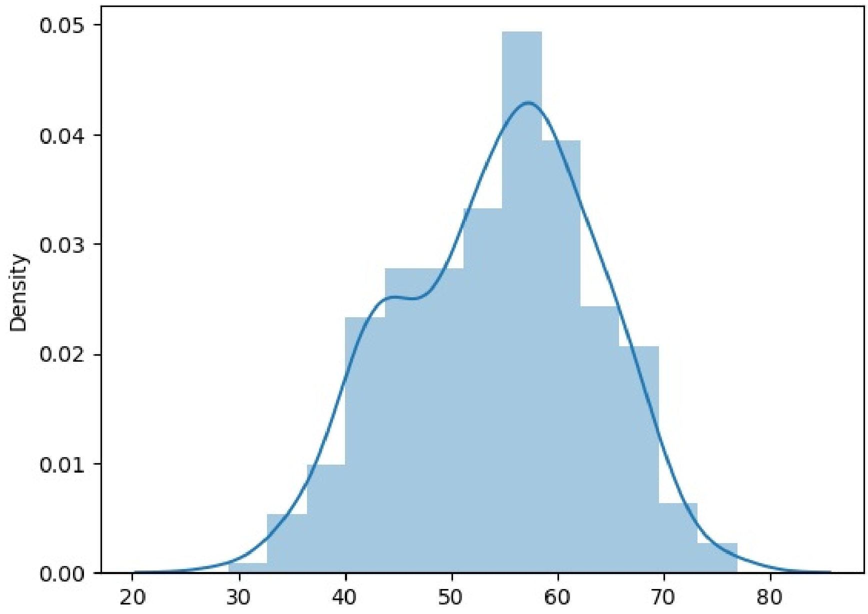
	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Step-5:::Knowing relationship between Variables by Using EDA

Age Analysis

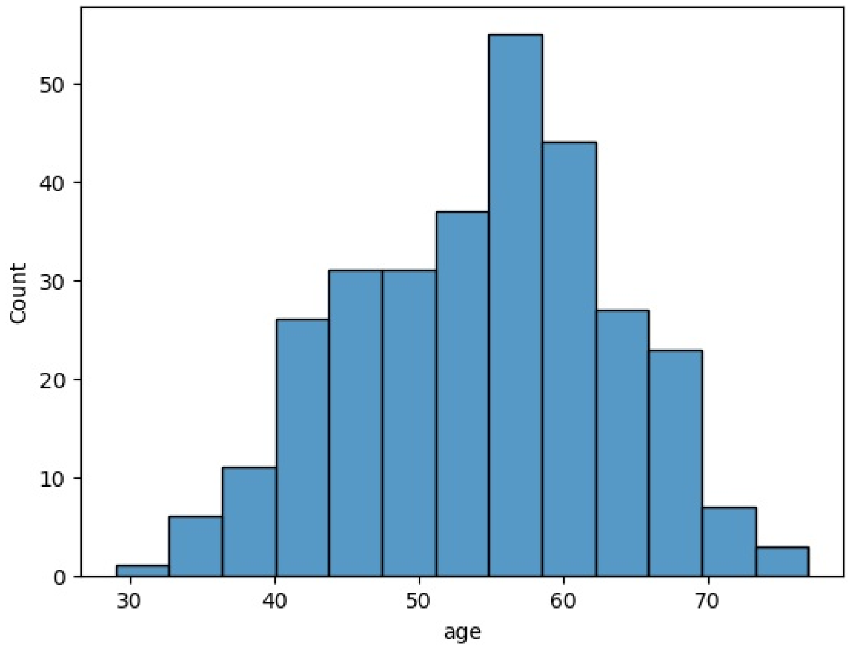
In [16]:
```
sns.distplot(x=data["age"])
```

```
C:\Users\malleswari\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a d
eprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a f
igure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```
Out[16]:
```
<AxesSubplot:ylabel='Density'>
```



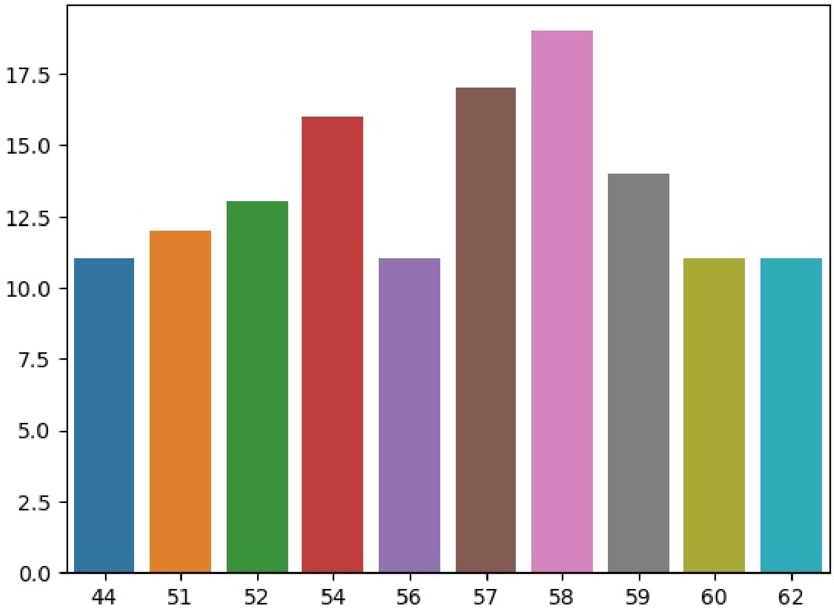
In [17]:
```
sns.histplot(x=data["age"])
```

Out[17]: <AxesSubplot:xlabel='age', ylabel='Count'>



In [18]:
```
sns.barplot(x=data.age.value_counts()[:10].index,y=data.age.value_counts()[:10].values)
```

Out[18]: <AxesSubplot:>

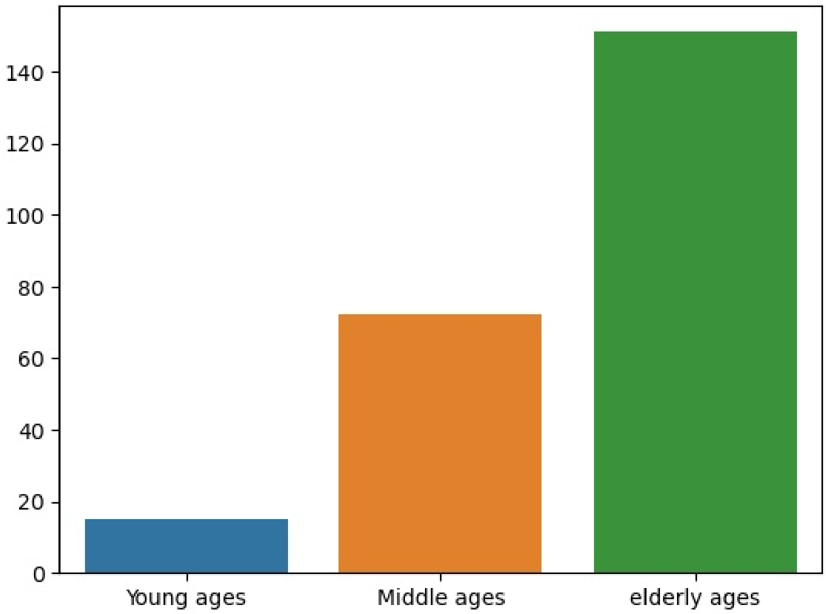


By this graph we can say that the majority of the peaople will affected the heart attack at the age 58 years

In [19]:
```
Young=data[(data.age>=29) & (data.age<40)]
middle=data[(data.age>=40) & (data.age<50)]
Elder=data[(data.age>55)]

sns.barplot(x=['Young ages','Middle ages','elderly ages'],y=[len(Young),len(middle),len(Elder)])
```

Out[19]:
```
<AxesSubplot:>
```

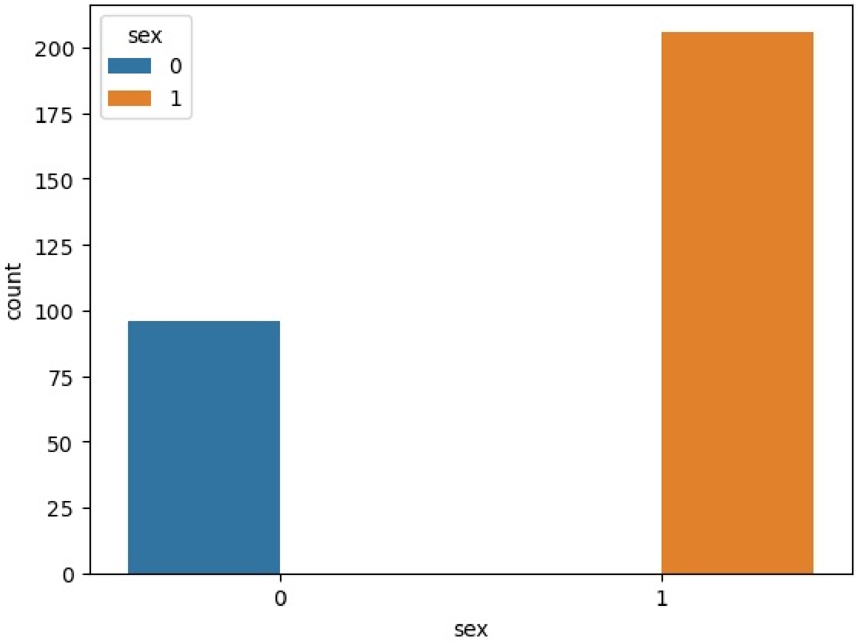


* By this plot we can conclude that elder peaople are the most affected by heart disease and young ones are the least affected

Gender Analysis

In [20]:
```
sns.countplot(x="sex",hue="sex",data=data)
```

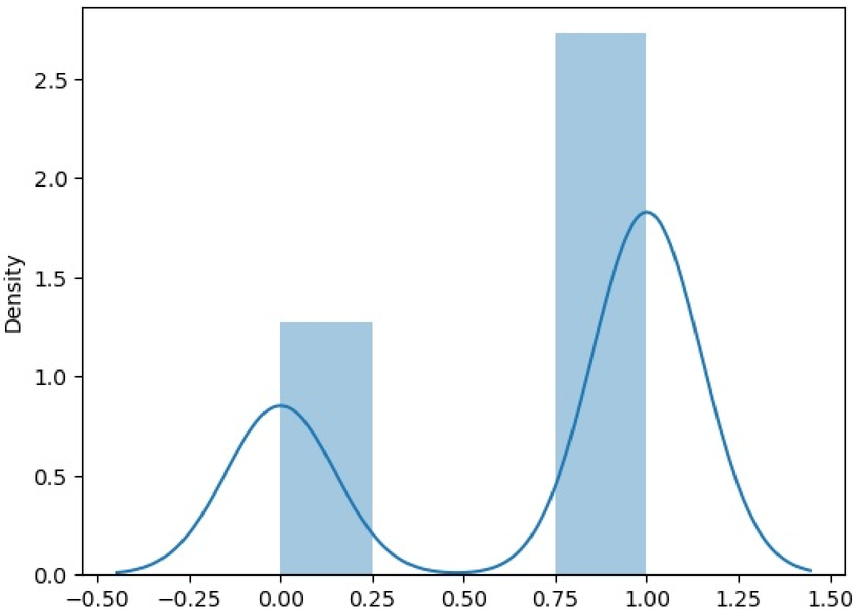
Out[20]: <AxesSubplot:xlabel='sex', ylabel='count'>



In [21]:
```
sns.distplot(x=data["sex"])
```

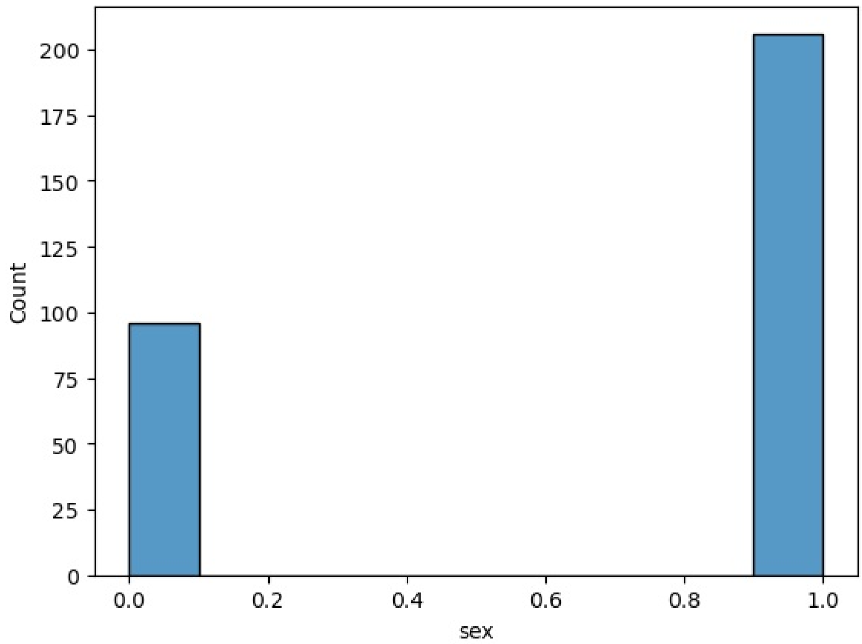
```
C:\Users\malleswari\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a d
eprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a f
igure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[21]: <AxesSubplot:ylabel='Density'>



In [22]:
```
sns.histplot(x=data["sex"])
```

Out[22]: <AxesSubplot:xlabel='sex', ylabel='Count'>



by this plot we can say that most of the mens get heart attack than womens

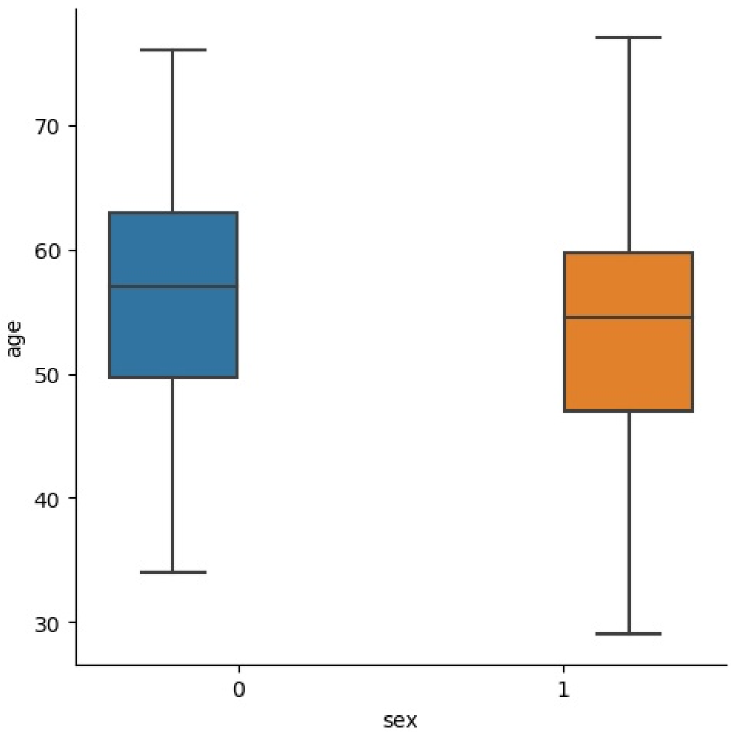
In [ ]:

In [ ]:

"To know the relation between Age and Gender of the human beings who get heart attack"

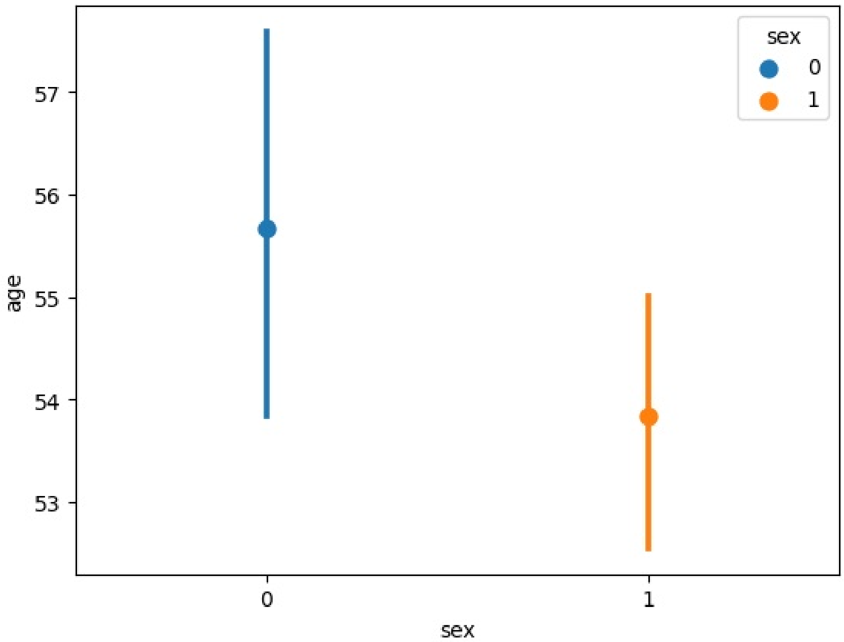
In [23]:
```
sns.catplot(x="sex",y="age",hue="sex",data=data,kind="box")
```

Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x21c4bbc6c10>
```



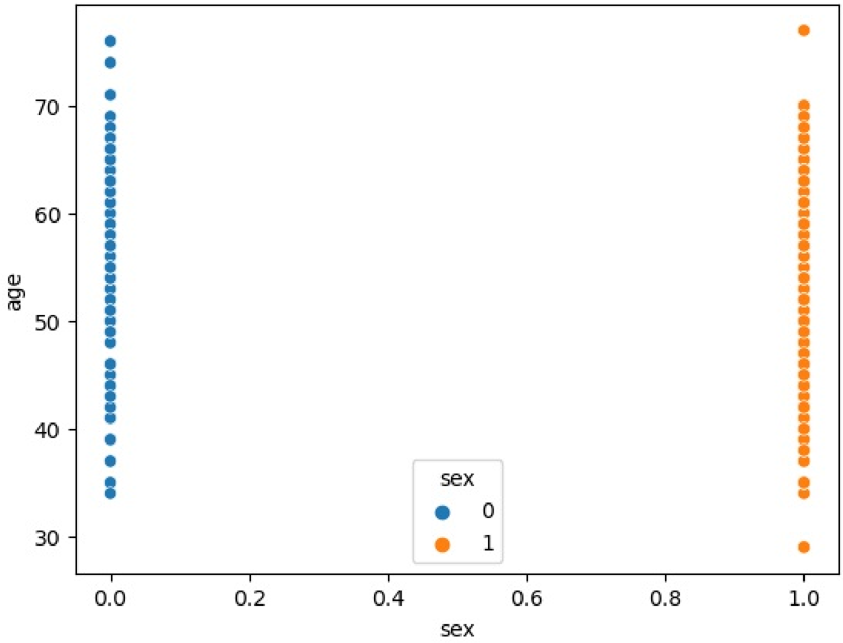
In [24]:
```
sns.pointplot(x="sex",y="age",data=data,hue="sex")
```

Out[24]: <AxesSubplot:xlabel='sex', ylabel='age'>



In [25]:
```
sns.scatterplot(x="sex",y="age",data=data,hue="sex")
```

Out[25]: <AxesSubplot:xlabel='sex', ylabel='age'>



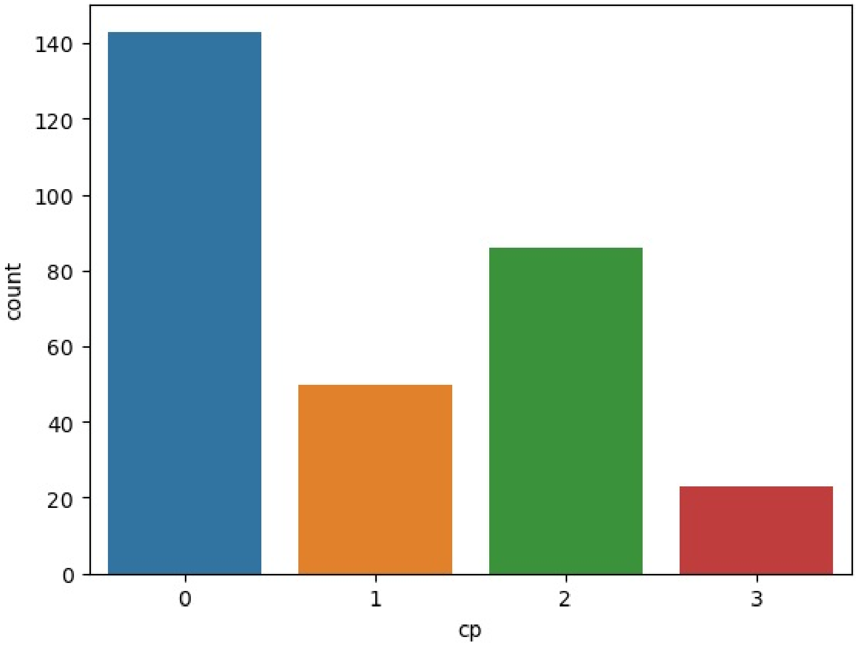
by this we can conclude that

As compare to the womens, mens get heart attack at younger age than womens

Chest pain analysis

In [26]:
```
sns.countplot(x=data["cp"])
```

Out[26]: <AxesSubplot:xlabel='cp', ylabel='count'>



1.Least Chest pain

2.Slightly distressed

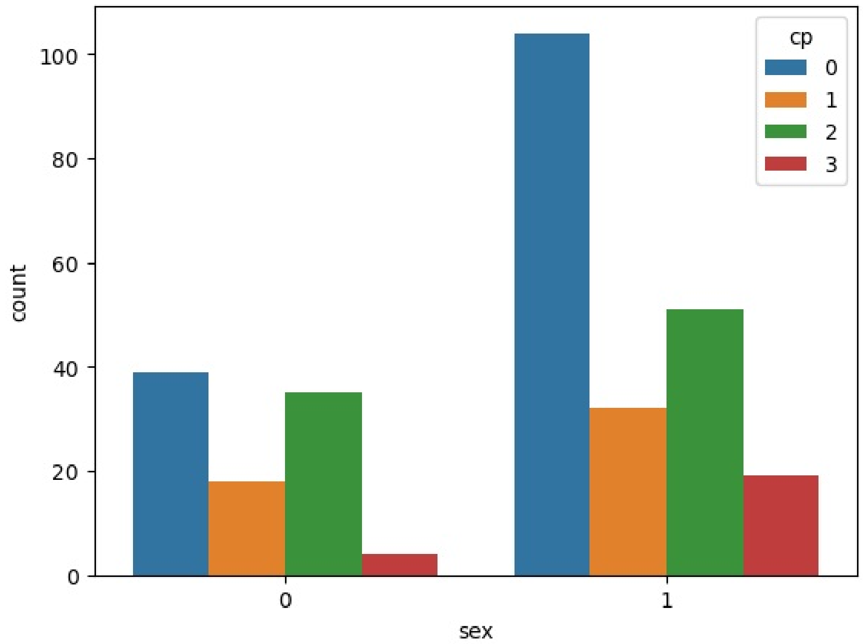
3.meadium distresses

4.high distressed

"To know the relation between Gender and Chest pain"

In [27]:
```
sns.countplot(x="sex",data=data,hue="cp")
```

Out[27]: <AxesSubplot:xlabel='sex', ylabel='count'>



By using above plot we say that

-As compare to the all types of chest pains more no.of womens and mens get "typical angina" type of chest pain

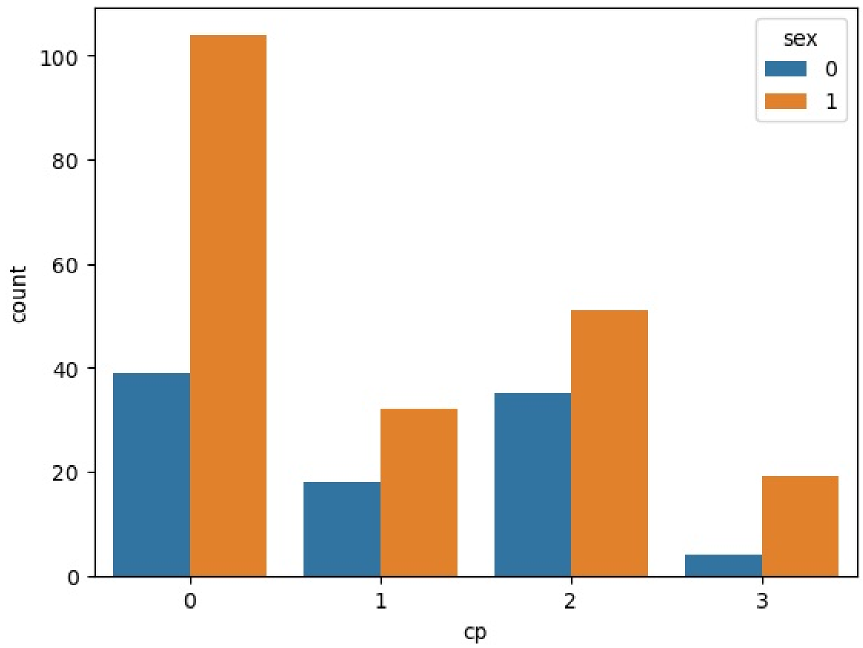
-As compare to the all types of chest pains less no.of womens and mens get "asymptomatic" type of chest pain

In [ ]:

In [ ]:

In [28]:
```
sns.countplot(x="cp",data=data,hue="sex")
```

Out[28]: <AxesSubplot:xlabel='cp', ylabel='count'>



By this plot we can conclude that

0(typical angina)

----There is more number (above 100 members) of mens get "Typical angina" type of chest pain than womens(between 35 and 40 members are there)

1(atypical angina)

----There is more number (between 30 and 40) of mens get "Atypical angina" type of chest pain than womens(between 15 and 20 members are there)

2(non-anginal pain)

----There is more number (near too 50) of mens get "Non-Anginal pain" type of chest pain than womens(between 35 and 40 members are there)

3(asymptomatic)

----There is more number (near to 20) of mens get "Asymptomatic" type of chest pain than womens(4 to 5 members)

As compare to both mens and womens ---mens are higher than womens who got heart attack

In [ ]:

In [ ]:

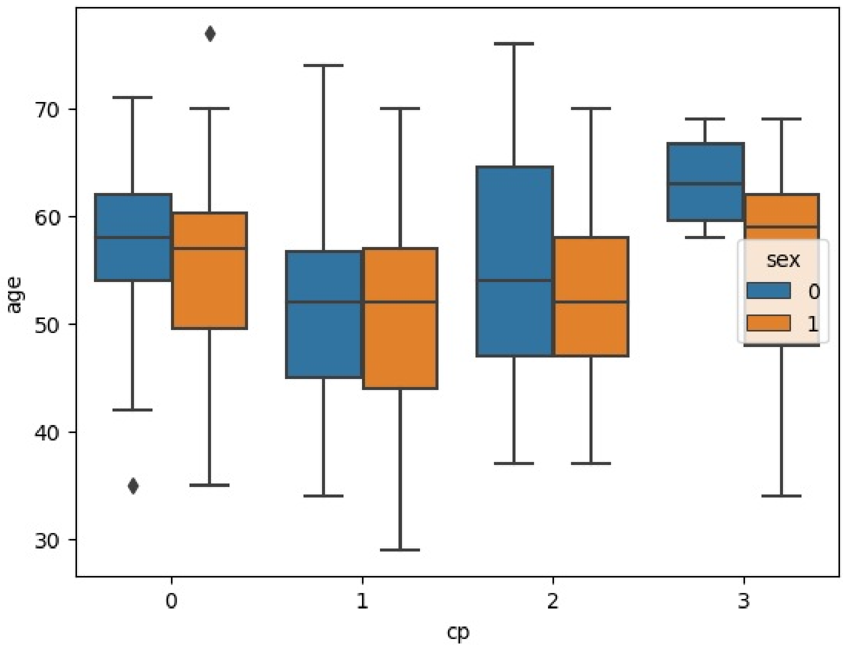
In [ ]:

In [ ]:

"" To Know the relation between Type of chest pain and Age Based on Gender ""

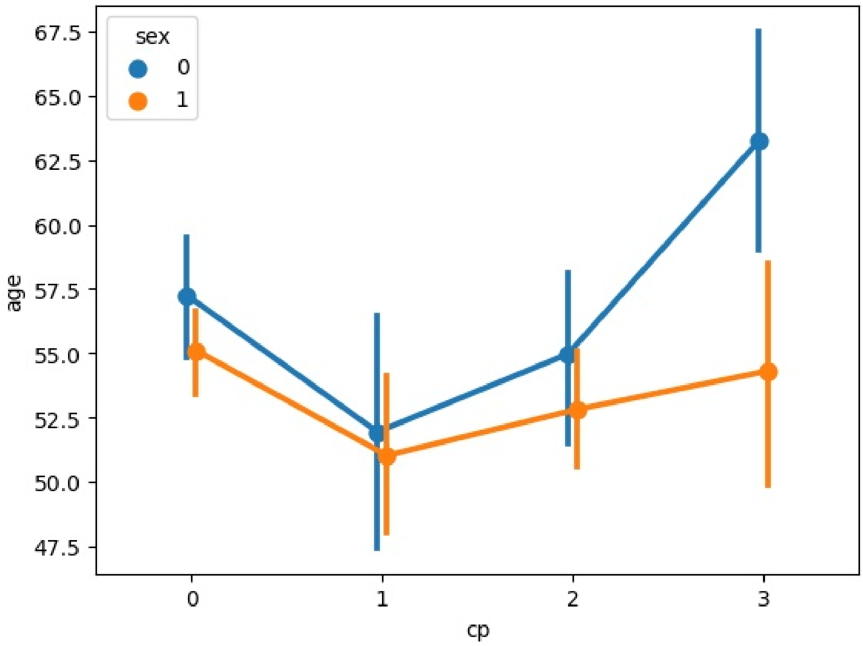
In [29]:
```
sns.boxplot(x="cp",y="age",data=data,hue="sex")
```

Out[29]:
```
<AxesSubplot:xlabel='cp', ylabel='age'>
```



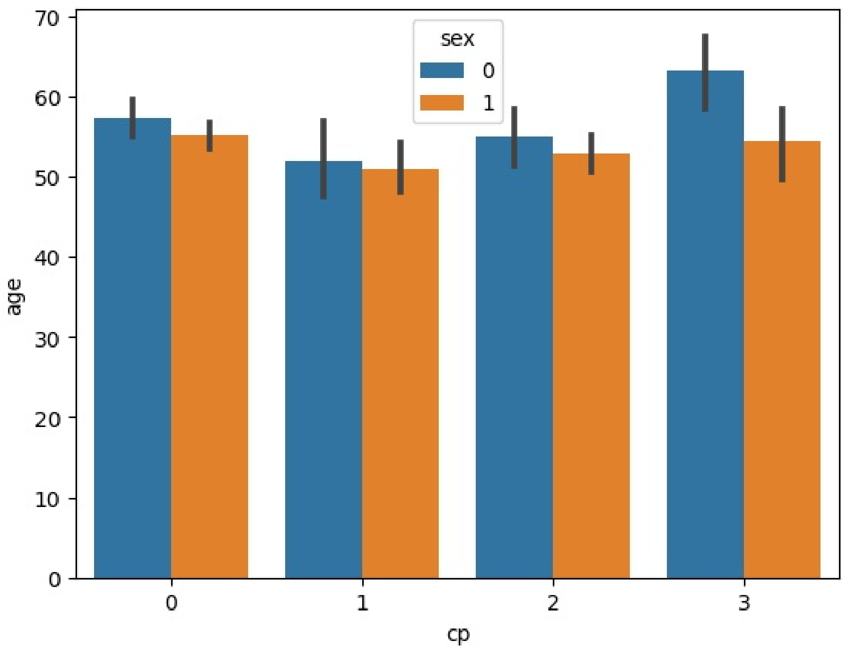
In [30]:
```
sns.pointplot(x="cp",y="age",data=data,hue="sex",dodge=True)
```

Out[30]:
```
<AxesSubplot:xlabel='cp', ylabel='age'>
```




```
In [31]: sns.barplot(x="cp",y="age",data=data,hue="sex",dodge=True)
```

Out[31]: <AxesSubplot:xlabel='cp', ylabel='age'>



By this plot we can conclude that

0(typical angina)

----Age limit for womens(near to 57) who got "Typical angina" type of chest pain is higher than mens(near to 55)

1(atypical angina)

----Age limit for womens(near to 52) who got "ATypical angina" type of chest pain is higher than mens(near to 50)

2(non-anginal pain)

----Age limit for womens(near to 55) who got "Non-anginal pain" type of chest pain is higher than mens(below 52)

3(asymptomatic)

----Age limit for womens(above 60) who got "Asymptomatic" type of chest pain is higher than mens(near to 52)

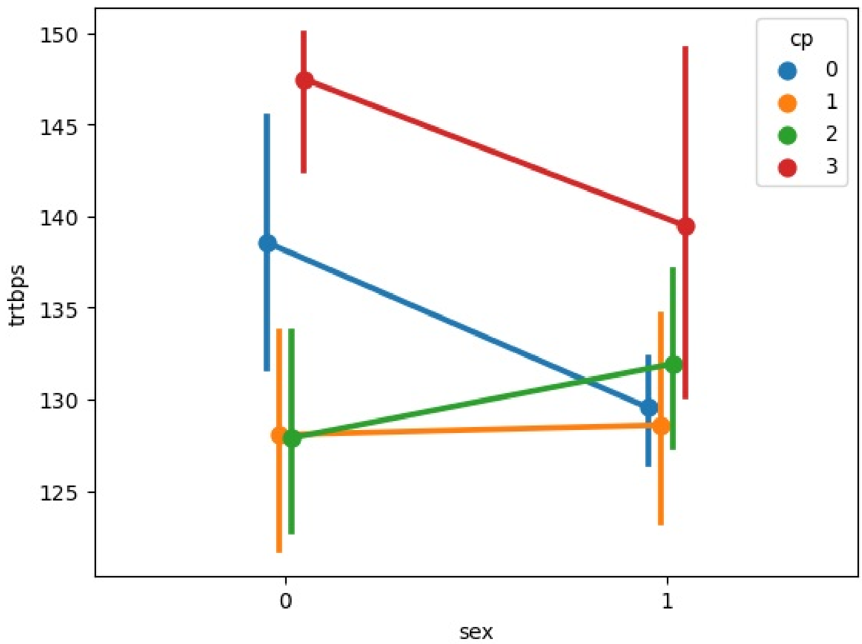
In [ ]:

In [ ]:

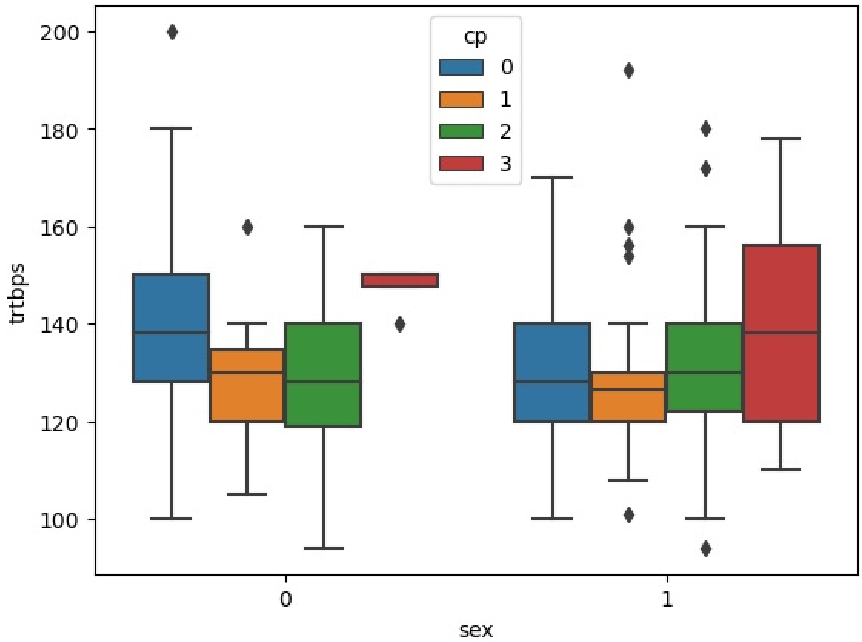
In [ ]:

Relation between Age and trtbps(resting blood pressure) based on Type of chest pain

In [32]:
```
sns.pointplot(x="sex",y="trtbps",data=data,hue="cp",dodge=True)
```
Out[32]:
```
<AxesSubplot:xlabel='sex', ylabel='trtbps'>
```

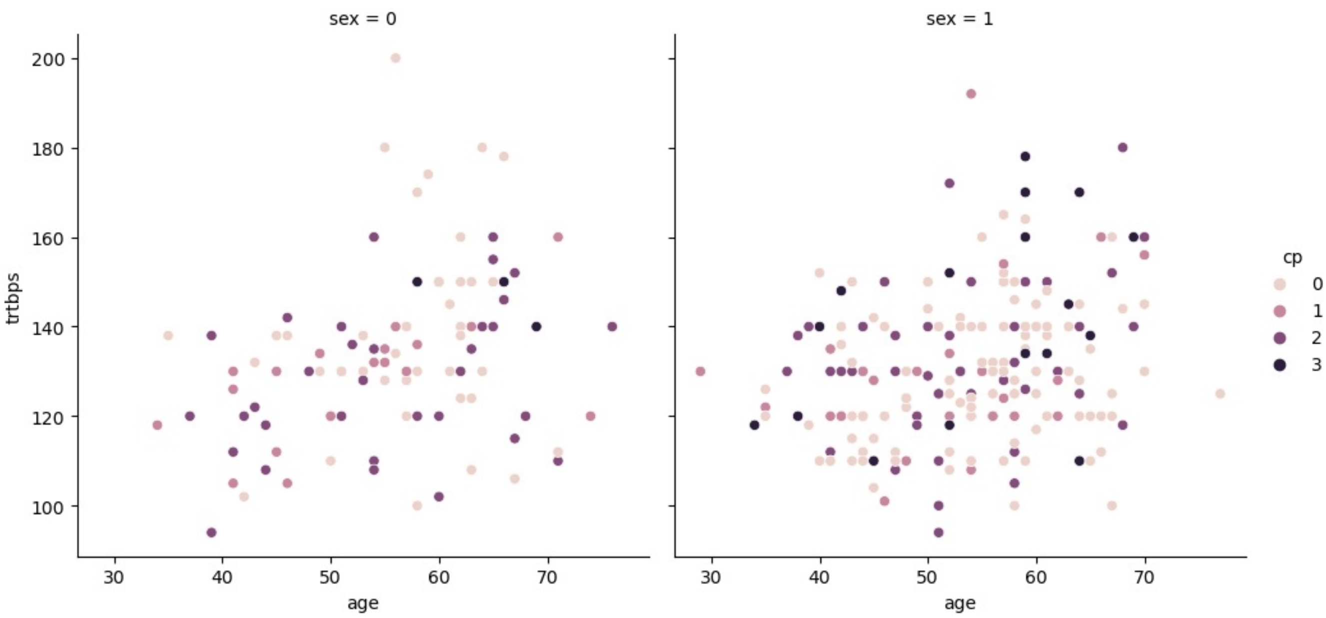


In [33]:
```
sns.boxplot(x="sex",y="trtbps",data=data,hue="cp")
```
Out[33]:
```
<AxesSubplot:xlabel='sex', ylabel='trtbps'>
```



In [34]:
```
sns.relplot(data=data,x="age",y="trtbps",hue="cp",col="sex",kind="scatter")
```

Out[34]: <seaborn.axisgrid.FacetGrid at 0x21c4e1a29a0>



By this plot we can conclude that

0(typical angina)

----hear blood pressure is between 126 mm hg and 146 mm hg(near) for who got Heart attack with "typical angina" type of chest pain

```
Here most of the womens at age above 60 years will get trtbps near to 137 mm hg
most of the mens at age between 50 and 60 years will get trtbps near to 130 mm hg
```

1(atypical angina)

----hear blood pressure is between 122 mm hg and 136 mm hg(near) for who got Heart attack with "atypical angina" type of chest pain

```
Here most of the womens at age near to 40 years will get trtbps near to 129 mm hg
most of the mens at age near to 50 years will get trtbps near to 129.5 mm hg
```

2(non-anginal pain)

----hear blood pressure is between 121.5 mm hg and 136.5 mm hg(near) for who got Heart attack with "non-anginal pain" type of chest pain

```
Here most of the womens at age between 50 and 60 years will get trtbps near to 129 mm
hg

most of the mens at age between 40 and 50 years will get trtbps near to 132.5 mm hg
```

3(asymptomatic)

----hear blood pressure is between 129 mm hg and 143 mm hg(near) for who got Heart attack with "Asymptomatic" type of chest pain

```
Here most of the womens at age between 55 and 65 years will get trtbps near to 147 mm
hg

most of the mens at age near to 60 years will get trtbps near to 137 mm hg
```

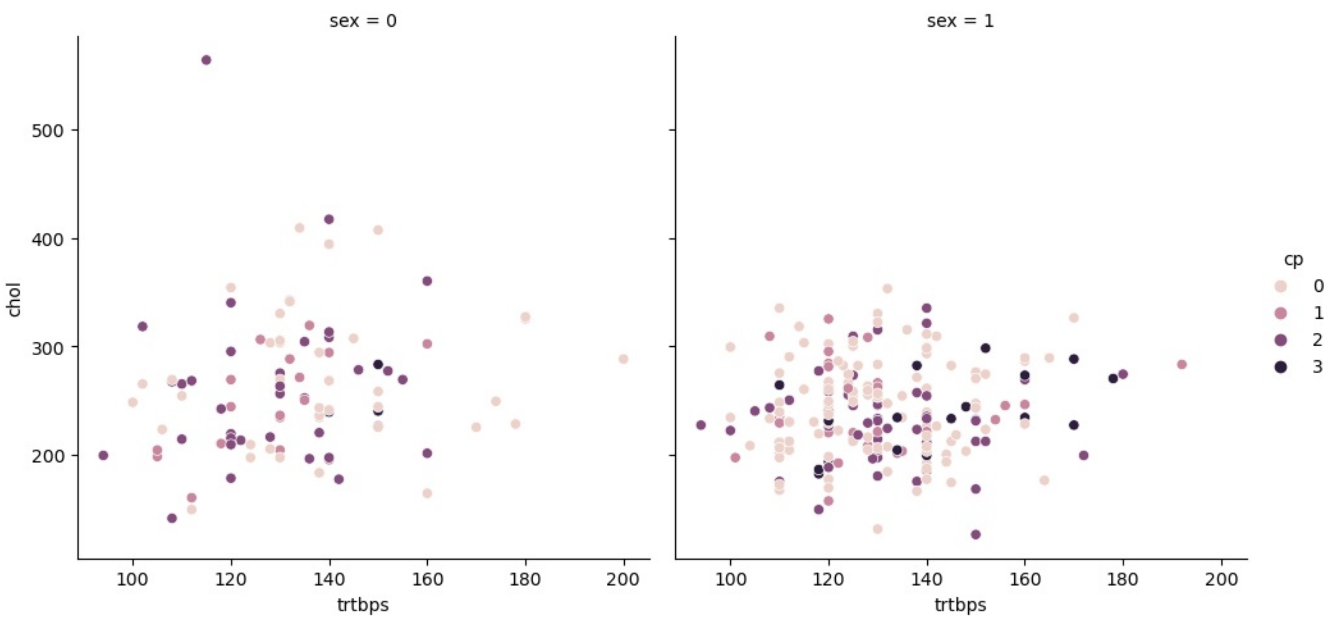
In [ ]:

In [ ]:

Relation between Blood Pressure(trtbps) and Cholastrol(chol) based on type of chest pain(cp)

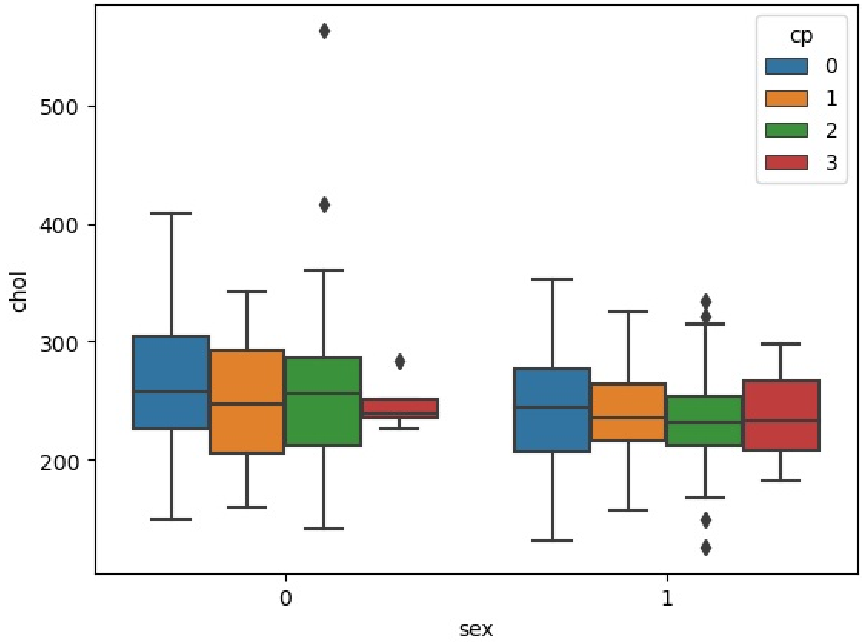
In [35]:
```
sns.relplot(y="chol",x="trtbps",col="sex",kind="scatter",data=data,hue="cp")
```

Out[35]:
```
<seaborn.axisgrid.FacetGrid at 0x21c4e271760>
```



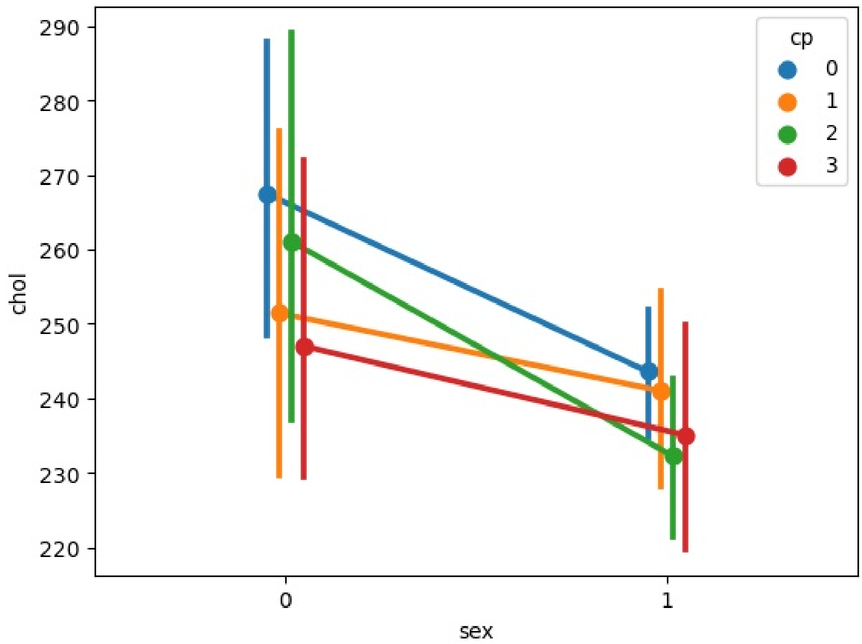
In [36]:
```
sns.boxplot(x="sex",y="chol",data=data,hue="cp")
```

Out[36]:
```
<AxesSubplot:xlabel='sex', ylabel='chol'>
```



In [37]:
```
sns.pointplot(x="sex",y="chol",dodge=True,data=data,hue="cp")
```

Out[37]: <AxesSubplot:xlabel='sex', ylabel='chol'>



By this plot we can conclude that

0(typical angina)

----hear cholestrol is between 238 mg/dl to 288 mg/dl for who got Heart attack with "typical angina" type of chest pain

```
        Here most of the womens will have cholestrol is about 268 mg/dl with near to 145 trtbps
        most of the mens womens will have cholestrol is about 248 mg/dl with trtbps between 120 mm hg to 140 mm hg
```

1(atypical angina)

----hear cholestrol is between 231 mm hg and 273 mm hg(near) for who got Heart attack with "atypical angina" type of chest pain

```
        Here most of the womens will have cholestrol is about 251 mg/dl with near to 140
trtbps
        most of the mens womens will have cholestrol is about 240 mg/dl with trtbps is about
to 120 mm hg
```

2(non-anginal pain)

----hear cholestrol is between 230 mm hg and 290 mm hg(near) for who got Heart attack with "atypical angina" type of chest pain

```
        Here most of the womens will have cholestrol is about 261 mg/dl with near to 125
trtbps
        most of the mens womens will have cholestrol is about 231 mg/dl with trtbps between
125 mm hg to 140 mm hg
```

3(asymptomatic)

----hear cholestrol is between 230 mm hg and 271 mm hg(near) for who got Heart attack with "atypical angina" type of chest pain

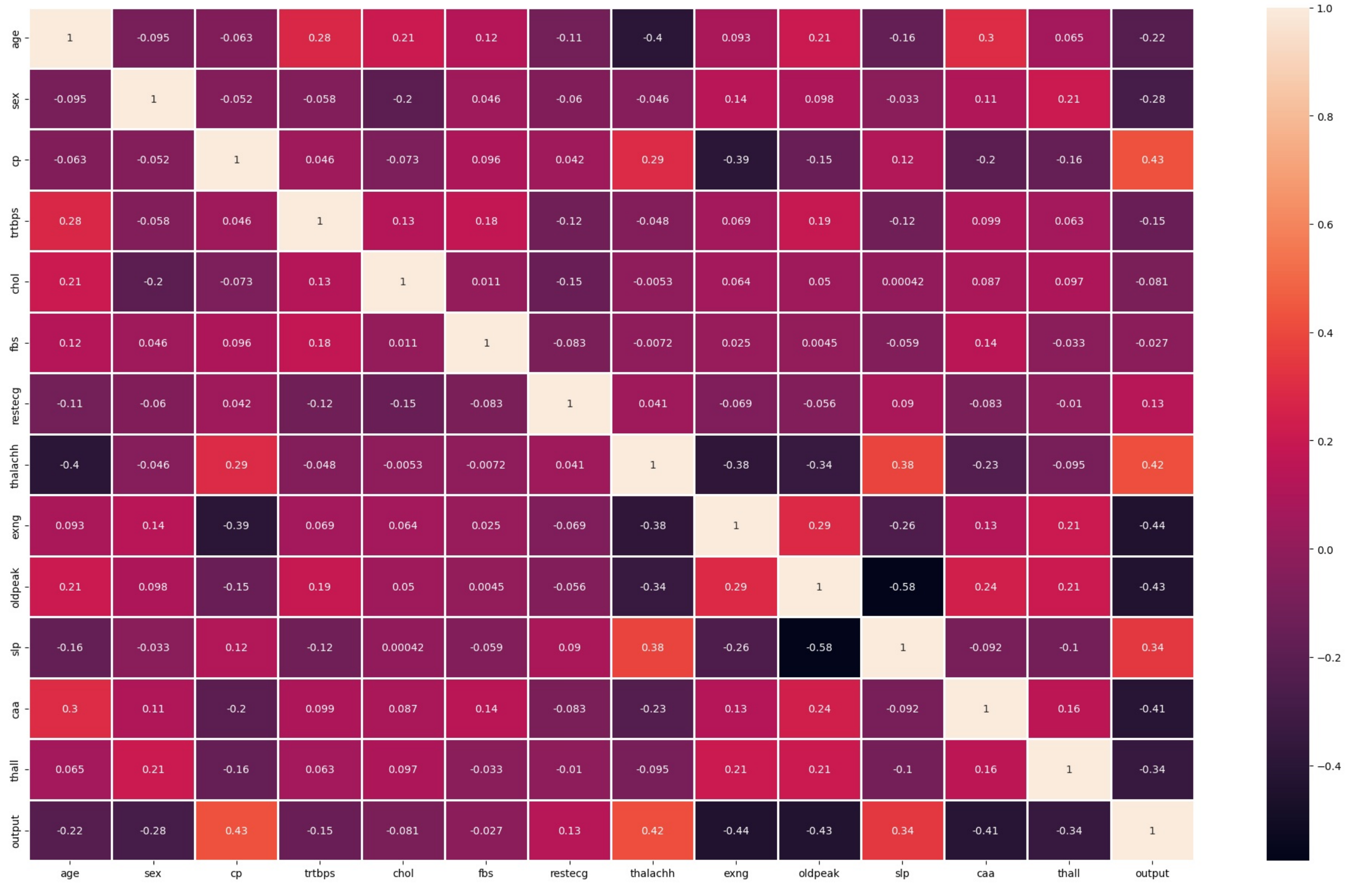
```
        Here most of the womens will have cholestrol is about 248 mg/dl with near to 145
trtbps
        most of the mens womens will have cholestrol is about 239 mg/dl with trtbps between
160 trtbps
```

In [ ]:

Correlation between variables:

In [38]:

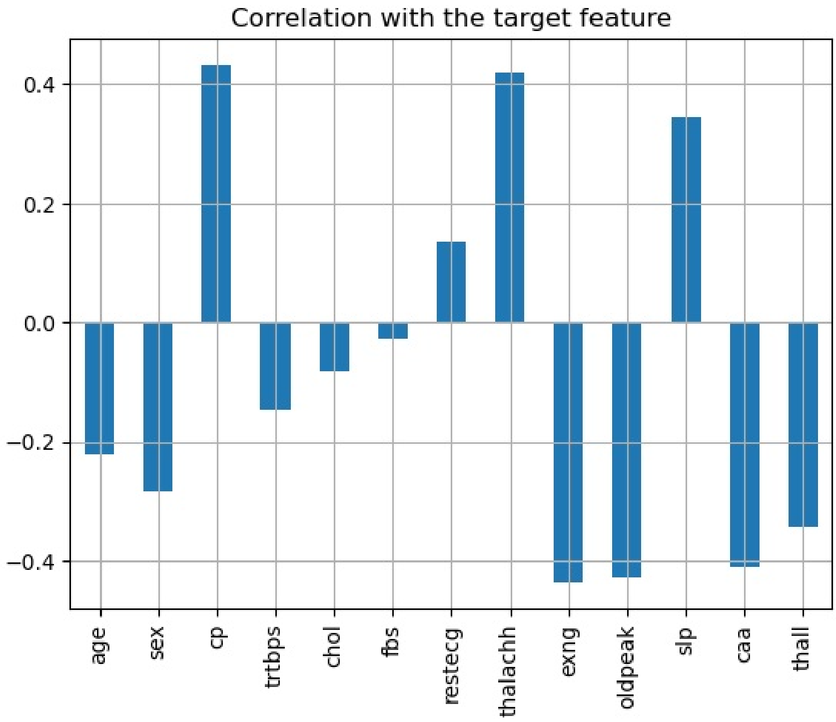
```
plt.figure(figsize=(20,12))
sns.heatmap(data.corr(),annot=True,linewidth=2)
plt.tight_layout()
```



In [39]:

```
data.drop('output',axis=1).corrwith(data.output).plot(kind="bar",grid="True",title="Correlation with the target
```

Out[39]: <AxesSubplot:title={'center':'Correlation with the target feature'}>



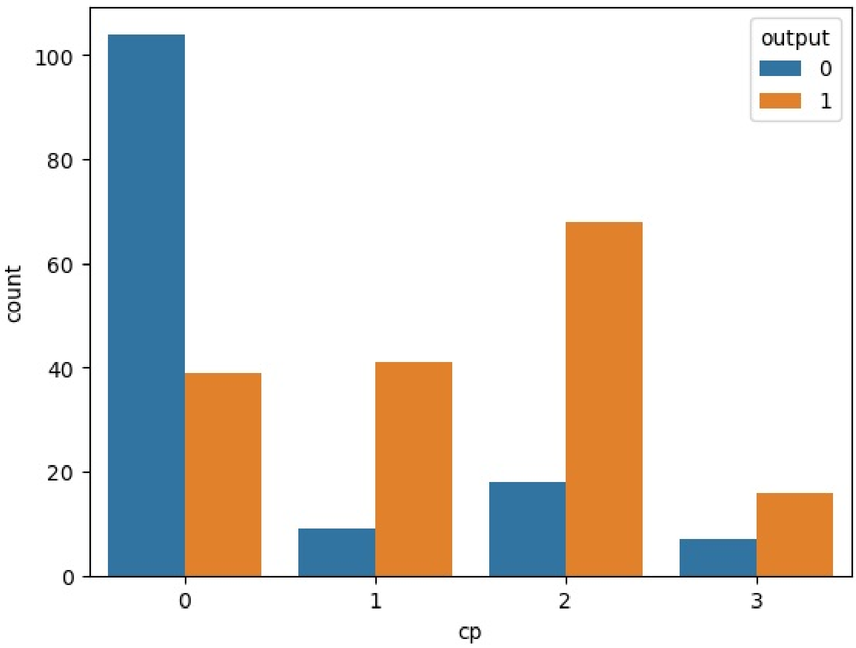
From above graph we conclude that---

- Four features(cp , restecg , thalachh, slp) are positively correlated with the target feature.
- Other features are negatively correlated with the target feature

To find the relation between chest pain(cp) and Heart Attack(output) based on age

In [40]:
```
sns.countplot(x=data["cp"],hue=data["output"])
```

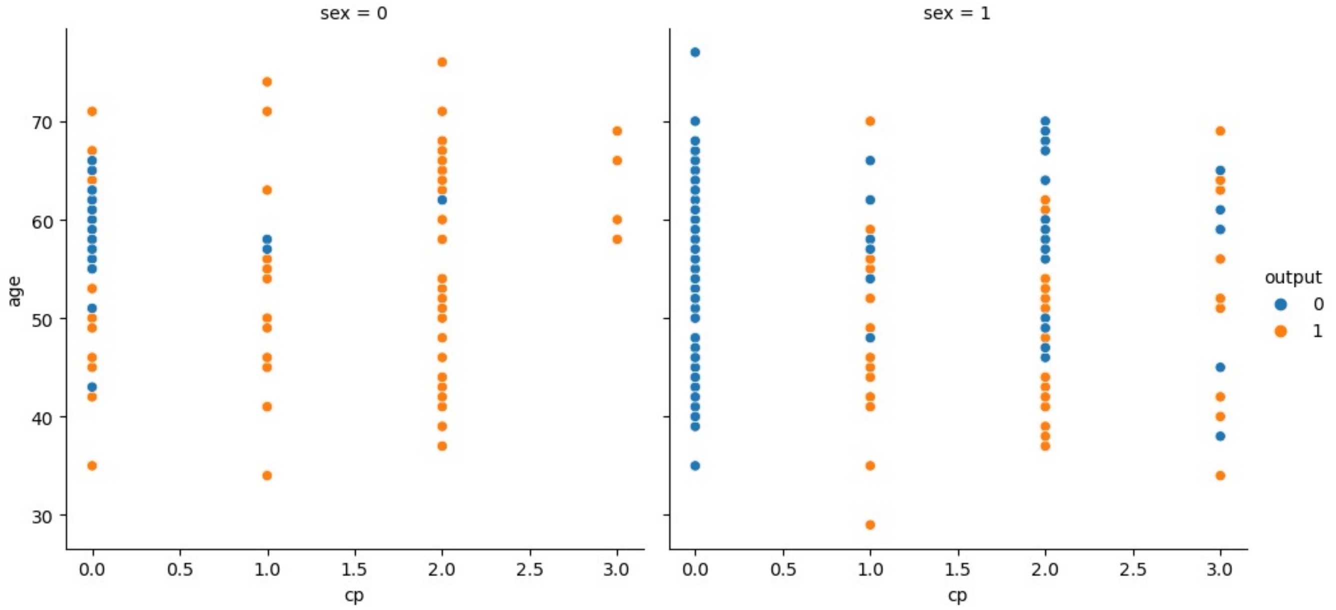
Out[40]:
```
<AxesSubplot:xlabel='cp', ylabel='count'>
```



- people have least chest pain are not likely to have heart disease
- people having severe chest pain are likely to have heart disease

In [41]:
```
sns.relplot(x="cp",y="age",col="sex",hue="output",data=data,kind="scatter")
```

Out[41]: <seaborn.axisgrid.FacetGrid at 0x21c4f69a6d0>



Final Conclusion

if lower the chest pain may cause the lower chances of heart attack

higher chest pain may cause the higher chances of heart attack

In [ ]:

In [ ]:

Apply Ml Algorithm to Our Dataset

In [42]:
```
data
```

Out[42]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

302 rows × 14 columns

Here there is no categorical valuesso this data is perfect to apply the Ml Model

removing target variable(Splitting)

In [43]:
```
data['output'].value_counts()
```

Out[43]:
```
1    164
0    138
Name: output, dtype: int64
```

In [44]:
```
x=data.drop(columns='output',axis=1)
y=data['output']
```

In [45]:
```
x
```

Out[45]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

302 rows × 13 columns

In [46]:

```
y
```

Out[46]:

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: output, Length: 302, dtype: int64
```

In [47]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y, random_state=40)
```

In [48]:

```
print(x.shape, x_train.shape, x_test.shape)
```

```
(302, 13) (241, 13) (61, 13)
```

In [ ]:

Splitting the data

In [57]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y, random_state=40)
print(x.shape, x_train.shape, x_test.shape)
```

```
(302, 13) (241, 13) (61, 13)
```

In [ ]:

Checking Overfitting and Underfitting

In [58]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
```

```
C:\Users\malleswari\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: 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-regression
  n_iter_i = _check_optimize_result(
```

Out[58]:

```
LogisticRegression()
```

In [59]:

```
from sklearn import metrics
print('training score: ',lr.score(x_train,y_train))
print('test score: ',lr.score(x_test,y_test))
```

```
training score:  0.8630705394190872
test score:  0.8852459016393442
```

In [ ]:

```
'''By using above results
training score and test score will be almost equal so it the neither underfitting or a overfitting'''
```

AUC-ROC curve

In [60]:
```
from sklearn.svm import SVC
model_svc=SVC(kernel='rbf', random_state=4)
model_svc.fit(x_train,y_train)

y_pred_svc=model_svc.decision_function(x_test)
```

In [62]:
```
from sklearn.linear_model import LogisticRegression
model_logistic = LogisticRegression()
model_logistic.fit(x_train,y_train)

y_pred_logistic=model_logistic.decision_function(x_test)
```

```
C:\Users\malleswari\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfg
s 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-regression
  n_iter_i = _check_optimize_result(
```

In [63]:
```
# Finding Threshold values for auc-roc curve

import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
logistic_fpr, logistic_tpr, threshold = roc_curve(y_test, y_pred_logistic)
auc_logistic = auc(logistic_fpr, logistic_tpr)

svm_fpr, svm_tpr, threshold= roc_curve(y_test, y_pred_svc)
auc_svm=auc(svm_fpr, svm_tpr)

threshold
```

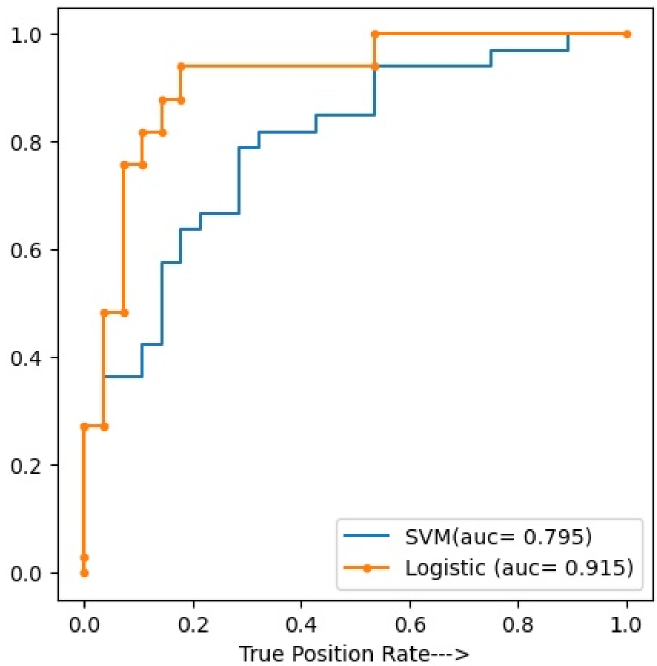
Out[63]:
```
array([ 2.31335938,  1.31335938,  0.9930326 ,  0.94370555,  0.799706  ,
        0.76903291,  0.75336001,  0.6961576 ,  0.60359861,  0.59640652,
        0.55430142,  0.53870105,  0.53186133,  0.47617372,  0.35022069,
        0.34117302,  0.33439401,  0.25932213,  0.23789842,  0.11636066,
       -0.09327361, -0.27358315, -0.28783171, -0.6115265 , -0.74430734,
       -0.95107628])
```

In [64]:
```
plt.figure(figsize=(5, 5), dpi=100)
plt.plot(svm_fpr, svm_tpr, linestyle='-', label='SVM(auc= %0.3f)' % auc_svm)
plt.plot(logistic_fpr, logistic_tpr, marker='.',label='Logistic (auc= %0.3f)' % auc_logistic)

plt.xlabel("False Position Rate--->")
plt.xlabel("True Position Rate--->")

plt.legend()

plt.show()
```



In [ ]:
```
''' here by seeing this plot we conclude that
orange line denotes ---logistic curve
blue line denotes---svm curve
and the AUC of Logistic is Greater than the AUC of SVM i.e, larger area means better module
```

```
so obviously we choose Logistic Regression
'''
```

In [ ]:

Model Training

In [66]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train, y_train)
```

```
C:\Users\malleswari\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfg
s 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-regression
  n_iter_i = _check_optimize_result(
```

Out[66]:
```
LogisticRegression()
```

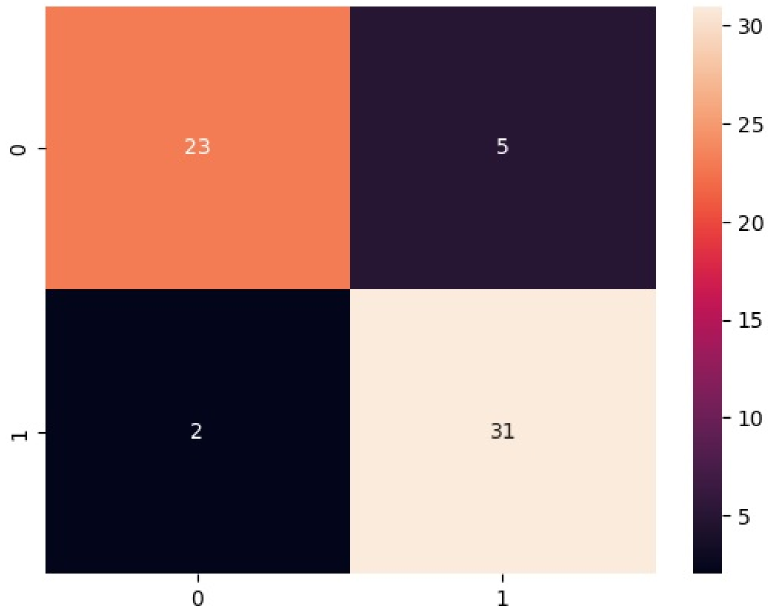
In [67]:
```
# confusion matrix
```

In [68]:
```
y_pred=model.predict(x_test)
from sklearn.metrics import confusion_matrix
con=confusion_matrix(y_test,y_pred)
print(con)
```

```
[[23  5]
 [ 2 31]]
```

In [69]:
```
label=[1,0]
sns.heatmap(con,label=label,annot=True)
```

Out[69]:
```
<AxesSubplot:>
```



In [ ]:
```
'''By using above Confusion_Matrix we conclude that our model the model is
performing better at predicting the negative class (people who do not have a heart attack) than the
positive class (people who do have a heart attack).'''
```

In [ ]:

In [74]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
x_train_prediction=model.predict(x_train)
training_data_accuracy=accuracy_score(x_train_prediction,y_train)
roc_loc_score1=roc_auc_score(x_train_prediction,y_train)
```

In [75]:
```
print("Accuracy on training data:", training_data_accuracy)

print("roc_loc_score1 is",roc_loc_score1)
```

```
Accuracy on training data: 0.8630705394190872
roc_loc_score1 is 0.8704397981254507
```

In [76]:
```
import sklearn.metrics as metrics
```

```
print(metrics.classification_report(x_train_prediction,y_train))
```

```
              precision    recall  f1-score   support

           0       0.78      0.91      0.84        95
           1       0.93      0.84      0.88       146

    accuracy                           0.86       241
   macro avg       0.86      0.87      0.86       241
weighted avg       0.87      0.86      0.86       241
```

In [ ]:

In [78]:
```
x_test_prediction=model.predict(x_test)
test_data_accuracy=accuracy_score(x_test_prediction,y_test)
roc_loc_score2=roc_auc_score(x_test_prediction,y_test)
```

In [79]:
```
print("Accuracy on testing data:", test_data_accuracy)
print("roc_loc_score2 is",roc_loc_score2)
```

```
Accuracy on testing data: 0.8852459016393442
roc_loc_score2 is 0.8905555555555555
```

In [80]:
```
print(metrics.classification_report(x_test_prediction,y_test))
```

```
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        25
           1       0.94      0.86      0.90        36

    accuracy                           0.89        61
   macro avg       0.88      0.89      0.88        61
weighted avg       0.89      0.89      0.89        61
```

In [ ]:
```
'''BY using above results we can say that our model will work 86% accurately in the Hear Attack dataset and it
an excellent model beacause of the AUC score is near to 1'''
```

In [ ]:

Predicting system

In [82]:
```
input_data=(62,0,0,140,268,0,0,160,0,3.6,0,2,2)

# change the input data into numpy array
input_data_array=np.array(input_data)

#reshaping the numpy array as we are prediction for only on instance
input_data_reshape=input_data_array.reshape(1,-1)

prediction=model.predict(input_data_reshape)
print(prediction)

if(prediction[0]==0):
    print("The person does not have a Heart Disease")
else:
    print("The person has Heart Disease")
```

```
[0]
The person does not have a Heart Disease
C:\Users\malleswari\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature
names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [83]:
```
input_data=(63,1,3,145,233,1,0 ,150 ,0,2.3,0,0,1)

# change the input data into numpy array
input_data_array=np.array(input_data)

#reshaping the numpy array as we are prediction for only on instance
input_data_reshape=input_data_array.reshape(1,-1)

prediction=model.predict(input_data_reshape)
print(prediction)

if(prediction[0]==0):
    print("The person does not have a Heart Disease")
else:
    print("The person has Heart Disease")
```

```
[1]
The person has Heart Disease
C:\Users\malleswari\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature
names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```