Importing Libraries

```
#importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
np.set_printoptions(suppress=True)
import sklearn
```

Loading dataset

upload file into google colab session

```
from google.colab import files
file = files.upload()
```

Choose Files Heart Disease data.csv

- **Heart Disease data.csv**(text/csv) - 38114 bytes, last modified: 3/11/2024 - 100% done

```
Saving Heart Disease data.csv to Heart Disease data (1).csv
```

```
#Loading dataset

data = pd.read_csv("Heart Disease data.csv", sep=",")
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	52	1	0	125	212	0	1	168	0	1.0	2	2	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	

Next steps: Generate code with data View recommended plots

```
data.shape
```

```
(1025, 14)
```

```
data.size
```

```
14350
```

Data Exploration

```
# Create Dataframe
## 1. Display Top 5 Rows of The Datasets

data.head()                # head()-shows default 5 rows
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	52	1	0	125	212	0	1	168	0	1.0	2	2	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	

Next steps: Generate code with data View recommended plots

```
## 2. Check the Last 5 Rows of The Datasets

data.tail()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
1020	59	1	1	140	221	0	1	164	1	0.0	2	0
1021	60	1	0	125	258	0	0	141	1	2.8	1	1
1022	47	1	0	110	275	0	0	118	1	1.0	1	1
1023	50	0	0	110	254	0	0	159	0	0.0	2	0
1024	54	1	0	120	188	0	1	113	0	1.4	1	1

```
## 3. Find Shape of our Dataset (No. of Rows and Columns)
##  shape = not methos, its is attribute of pandas dataframe

print(data.shape)        #ans is python tuple at index (0, 1) i.e (1025, 14)

print("Number of Rows", data.shape[0])
print("Number of Columns", data.shape[1])
```

```
(1025, 14)
Number of Rows 1025
Number of Columns 14
```

```
## 4. Information about Database

print(data.info())    #about no.of rows & columns, datatype of each columns, memory
```

```
<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
None
```

ETL- Extract-Transform-Load

5. Check Null Values in the Dataset

```
## 5. Check Null Values in the Dataset

print(data.isnull())           ## ans as a Boolean values - True or False...Here False in table
data.isnull().sum()      # 0 values to all columns
```

```
        age    sex     cp  trestbps   chol    fbs  restecg  thalach  exang  \
0     False  False  False     False  False  False    False    False  False
1     False  False  False     False  False  False    False    False  False
2     False  False  False     False  False  False    False    False  False
3     False  False  False     False  False  False    False    False  False
4     False  False  False     False  False  False    False    False  False
...     ...    ...    ...       ...    ...    ...      ...      ...    ...
1020  False  False  False     False  False  False    False    False  False
1021  False  False  False     False  False  False    False    False  False
1022  False  False  False     False  False  False    False    False  False
1023  False  False  False     False  False  False    False    False  False
1024  False  False  False     False  False  False    False    False  False

      oldpeak  slope     ca   thal  target
0       False  False  False  False   False
```

```
1       False  False  False  False   False
2       False  False  False  False   False
3       False  False  False  False   False
4       False  False  False  False   False
...       ...    ...    ...    ...     ...
1020    False  False  False  False   False
1021    False  False  False  False   False
1022    False  False  False  False   False
1023    False  False  False  False   False
1024    False  False  False  False   False

[1025 rows x 14 columns]
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
```

6. Check Duplicate Data and it exists,then Drop them

```
## 6. Check Duplicate Data

data_dup = data.duplicated().any()
print(data_dup)        ## ans is True..i.e some duolicates are there
```

```
True
```

```
##Check and it exists,
data[data.duplicated()==True]
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
15	34	0	1	118	210	0	1	192	0	0.7	2	0
31	50	0	1	120	244	0	1	162	0	1.1	2	0
43	46	1	0	120	249	0	0	144	0	0.8	2	0
55	55	1	0	140	217	0	1	111	1	5.6	0	0
61	66	0	2	146	278	0	0	152	0	0.0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0
1021	60	1	0	125	258	0	0	141	1	2.8	1	1
1022	47	1	0	110	275	0	0	118	1	1.0	1	1
1023	50	0	0	110	254	0	0	159	0	0.0	2	0
1024	54	1	0	120	188	0	1	113	0	1.4	1	1

723 rows × 14 columns

```
# then Drop them

data = data.drop_duplicates()
print(data)

print(data.shape)          ## To show no. of rows and columns respectively
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      52    1   0       125   212    0        1      168      0      1.0
1      53    1   0       140   203    1        0      155      1      3.1
2      70    1   0       145   174    0        1      125      1      2.6
3      61    1   0       148   203    0        1      161      0      0.0
4      62    0   0       138   294    1        1      106      0      1.9
..    ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
723    68    0   2       120   211    0        0      115      0      1.5
733    44    0   2       108   141    0        1      175      0      0.6
739    52    1   0       128   255    0        1      161      1      0.0
843    59    1   3       160   273    0        0      125      0      0.0
878    54    1   0       120   188    0        1      113      0      1.4
```

```
     slope  ca  thal  target
0        2   2     3       0
1        0   0     3       0
2        0   0     3       0
3        2   1     3       0
4        1   3     2       0
..     ...  ..   ...     ...
723      1   0     2       1
733      1   0     2       1
739      2   1     3       0
843      2   0     2       0
878      1   1     3       0

[302 rows x 14 columns]
(302, 14)
```

```
## Again Check if still their is any duplicate data
data[data.duplicated()==True]     ## Ans- No
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal

EDA Started

7. Get Overall Statistics about the Dataset

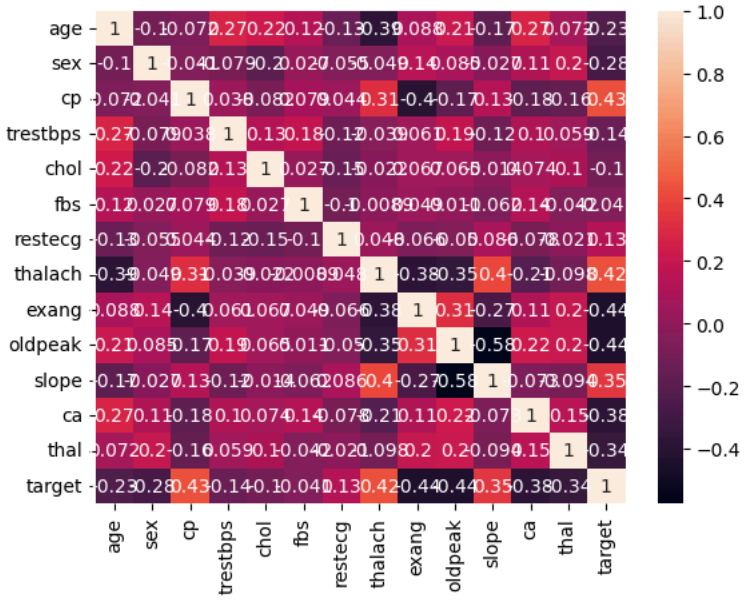
```
data.describe()      ## as it has int datatype, it is showing Numerical statistics i.e -Count,mean,std,min,% - Y axis
data.describe().T                  ## T - Column name , i.e sex, age ,cp etc- verticolarly i.e Y axis
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000

8. Draw Co-relation matrix

```
# print(data.corr())

# Plot heatmap with correlation values
print(sns.heatmap(data.corr(),annot=True))      ## ans- Axes(0.125,0.11;0.62x0.77)
plt.figure(figsize =(17,6))
```

```
Axes(0.125,0.11;0.62x0.77)
<Figure size 1700x600 with 0 Axes>
```

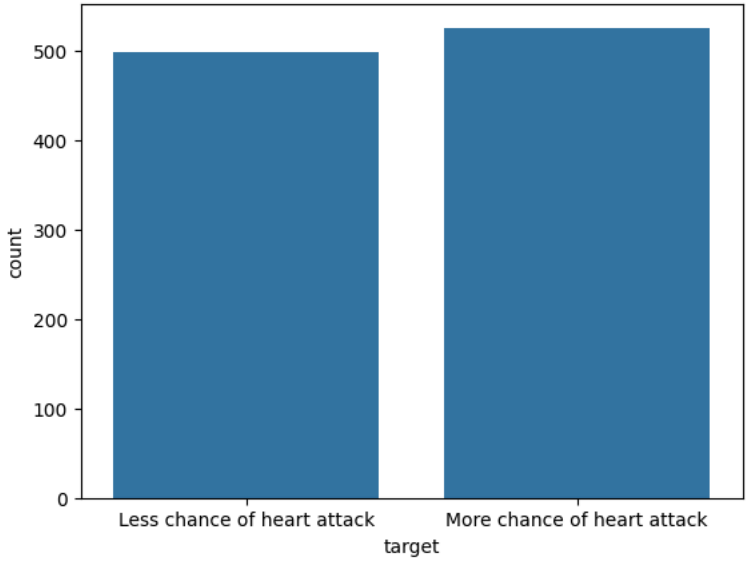



```
<Figure size 1700x600 with 0 Axes>
```

9. How Many & How Don't People have Heart Disease in this Dataset

```
##

print(data.columns)   ## shows column names

print(data["target"].value_counts())         # value_counts() - count of unique values in descending order..Ans- target 1  164 ,0  130

print(sns.countplot(x="target",data=data))   ##Bar graph  ##no visualiaztion shown only in sns         #countplot to check how many peo
plt.xticks(([0,1]),['Less chance of heart attack', 'More chance of heart attack'])
print(plt.show())
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
1    526
0    499
Name: target, dtype: int64
Axes(0.125,0.11;0.775x0.77)
```

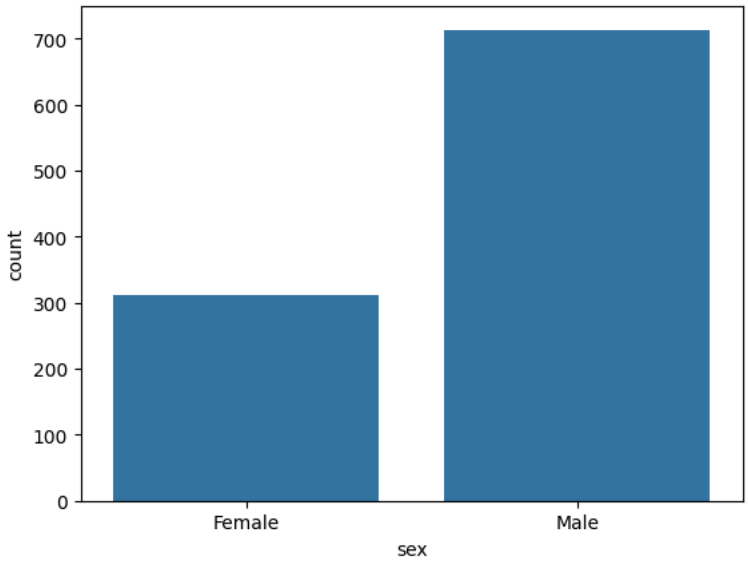



```
None
```

10. Find count of Male and Female in this Dataset

```
#

print(data.columns)   ## shows column names

    #interested to find No. of Males and Female
print(data["sex"].value_counts())         # value_counts() - count of unique values in descending order..Ans- target 1  164 ,0  130
print(sns.countplot(x = "sex", data = data))     # two inputs x is column which count require and data is dataset
plt.xticks(([0,1]),['Female', 'Male'])           ##Bar graph        #countplot to check how many people have heart disease
print(plt.show())
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
1    713
0    312
Name: sex, dtype: int64
Axes(0.125,0.11;0.775x0.77)
```

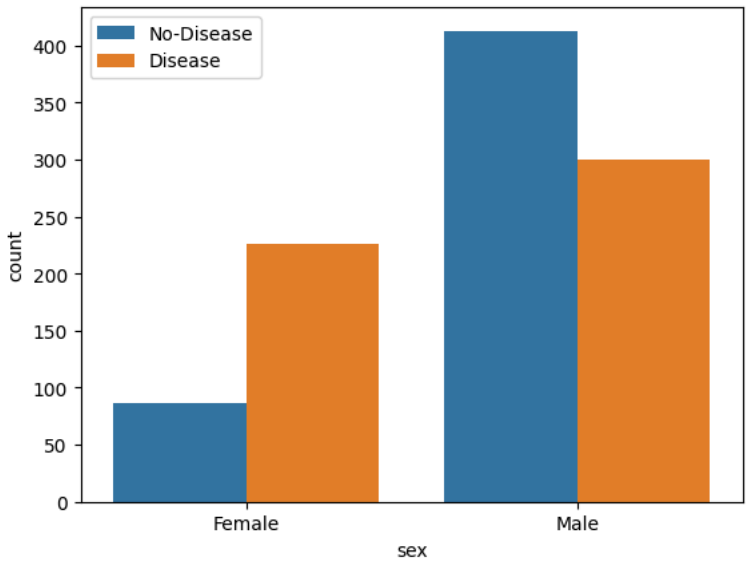



```
None
```

11. Find Gender Distribution According to the Target Variable

```
print(data.columns)

print(sns.countplot(x='sex', hue='target',data=data))

plt.xticks(([1,0]),['Male','Female'])
print(plt.legend(labels=['No-Disease', 'Disease']))
plt.show()             ## Bar Chart (0,1)
```

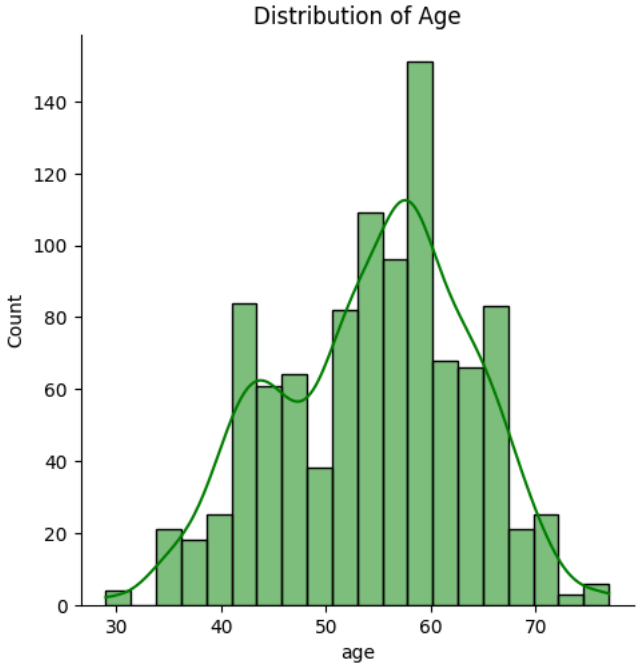
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
Axes(0.125,0.11;0.775x0.77)
Legend
```



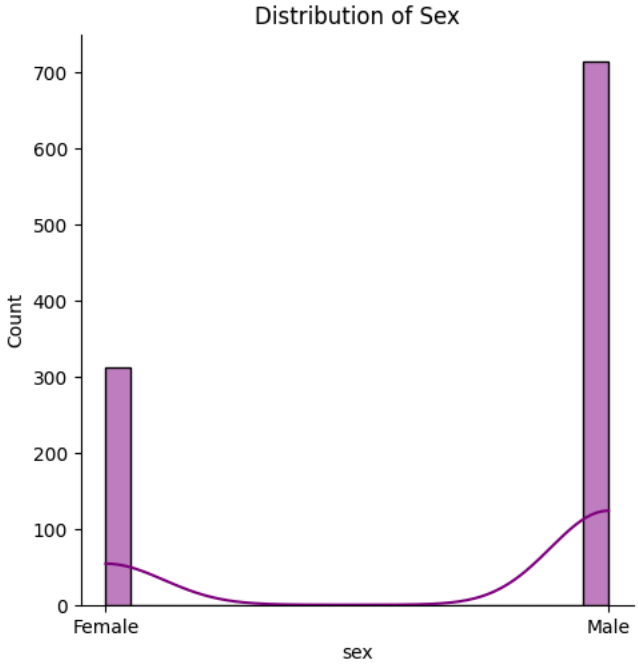
12. Check Age and Sex Distibution in the Dataset

```
##

print(sns.displot(data['age'],bins=20,color='green',kde=True))        ##ans- between 50-60 & displot is used to check  Age Distibution
plt.title("Distribution of Age")                              ## Histogram , line chart added to it due to -kde=True
print(plt.show())


print(sns.displot(data['sex'],bins=20,color='purple',kde=True))
plt.xticks(([0,1]),['Female', 'Male'])           ##Bar graph  ##no visualiazation shown only in sns         #countplot to check how many
plt.title("Distribution of Sex")
print(plt.show())
```

```
<seaborn.axisgrid.FacetGrid object at 0x7f12257bff70>
```




```
None
<seaborn.axisgrid.FacetGrid object at 0x7f12210d5a20>
```

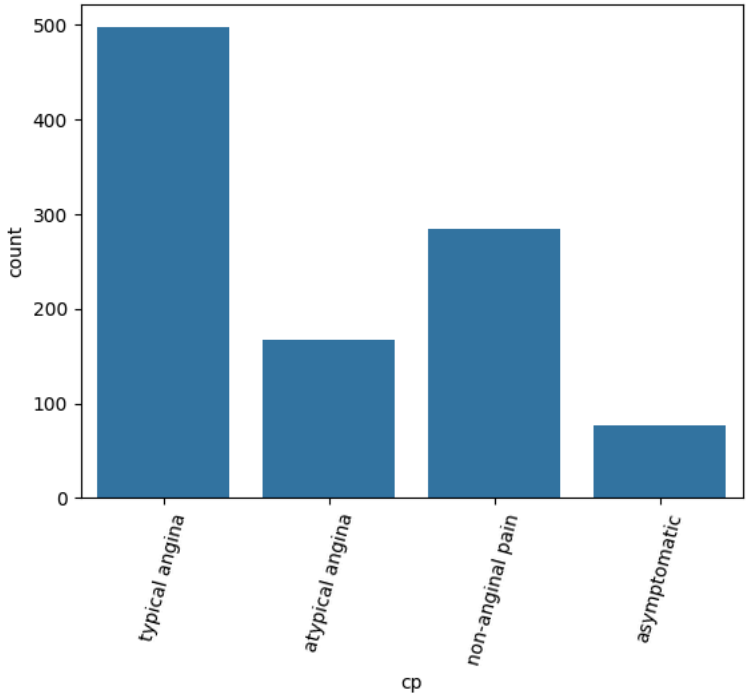



```
None
```

13. Check Chest Pain Type

```
##...(4 values) --To find which chest pain type is common (i.e Value)

print(sns.countplot(x="cp",data = data))

plt.xticks(([0,1,2,3]),['typical angina','atypical angina', 'non-anginal pain','asymptomatic'])   ##here Value 0 i.e typical angina is c
plt.xticks(rotation=75)             ##to change rotation of labels of X-axis
print(plt.show())                  ## graph -Bar Chart
```

```
Axes(0.125,0.11;0.775x0.77)
```

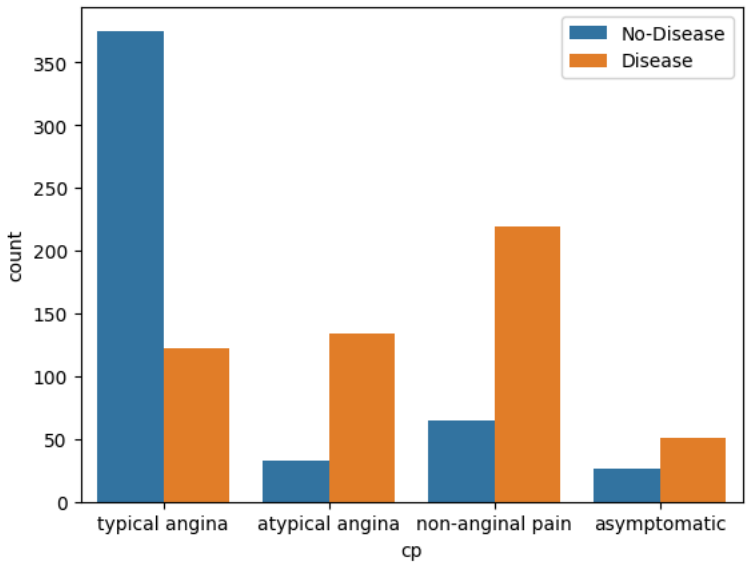



```
None
```

14. Show The Chest Pain Distribution As Per Target Variable

```
##
##( Healthy people having Chest Pain due to stress etc , & varies as per gender)
print(data.columns)

print(sns.countplot(x='cp', hue='target',data=data))
plt.legend(labels=['No-Disease', 'Disease'])
plt.xticks(([0,1,2,3]),['typical angina','atypical angina', 'non-anginal pain','asymptomatic'])
plt.show()
```

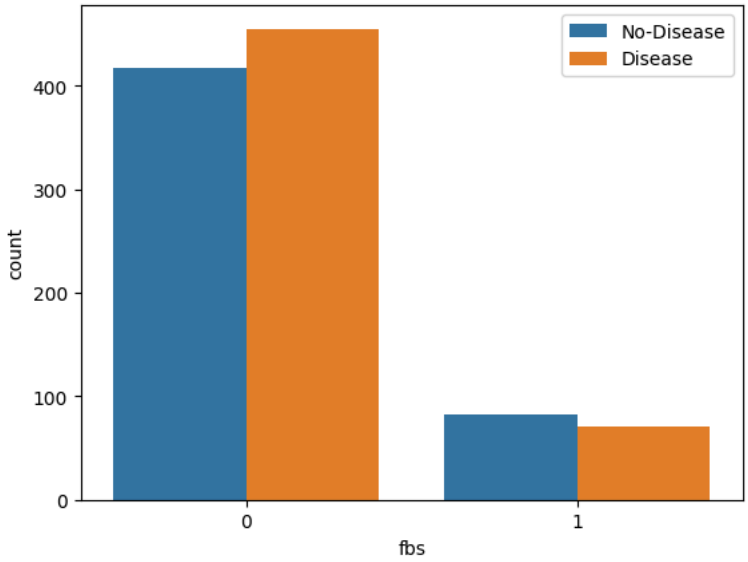
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
Axes(0.125,0.11;0.775x0.77)
```



15. Show Fasting Blood Sugar Distribution According To Target Variable

```
##

print(sns.countplot(x='fbs', hue='target',data=data))    #fbs -fasting blood sugar > 120 mg/dl = disease
print(plt.legend(labels=['No-Disease', 'Disease']))     ## @ Bar chart
print(plt.show())
```

```
Axes(0.125,0.11;0.775x0.77)
Legend
```

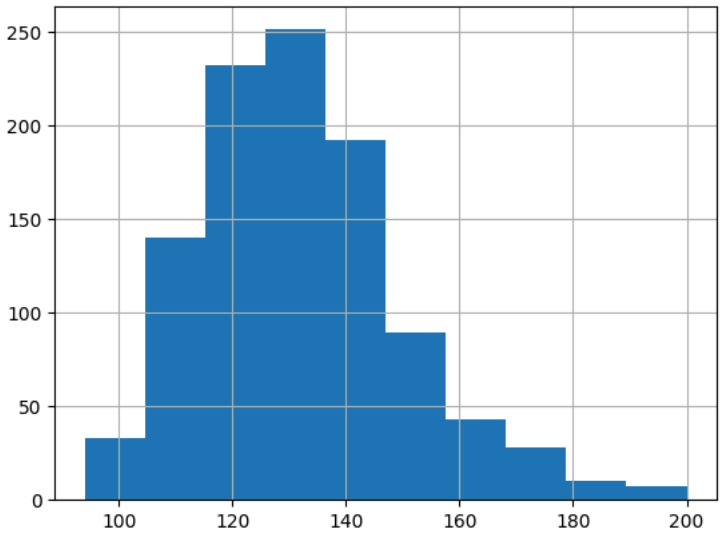



```
None
```

16. Compare Resting Blood Presure As Per Sex column

```
##
print(data.columns)

print(data['trestbps'].hist())    ## @  histogram  ## it show that bp of people is between 120 and 130
print(plt.show())
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
Axes(0.125,0.11;0.775x0.77)
```

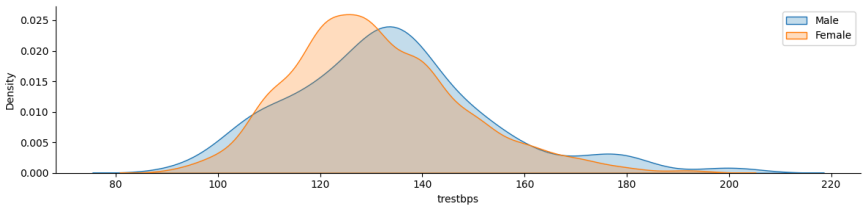



```
None
```

17. compare Resting Blood Presure As Per Sex column

```
##

## facetgrid class  = useful when want to visualize distribution of variable OR  relationship between multiple variables separately
## within sunset of your dataset

g=sns.FacetGrid(data, hue="sex", aspect=4)
print(g.map(sns.kdeplot,'trestbps', fill = True))    ## @ line graph
print(plt.legend(labels=['Male','Female']))      ## females having less resting bp i.e 120 than male 140
print(plt.show())
```

```
<seaborn.axisgrid.FacetGrid object at 0x7f121a8a6c50>
Legend
```

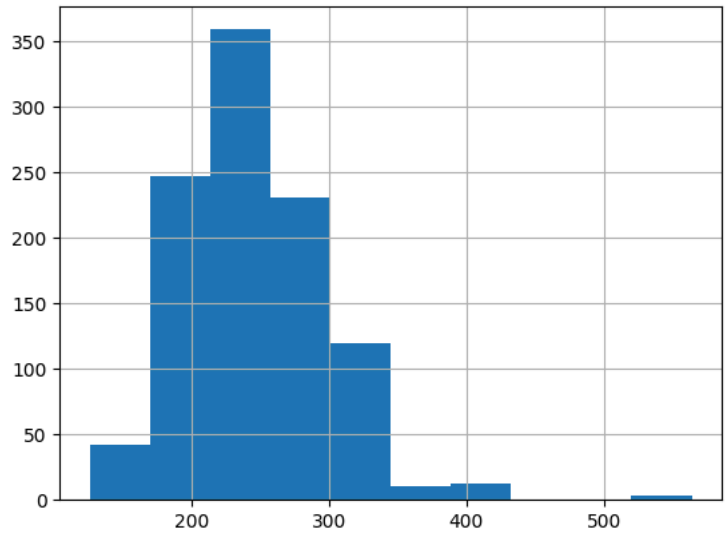



```
None
```

18. Show distribution of Serum Cholesterol (serum cholestoral in mg/dl)

```
##

print(data.columns)

print(data['chol'].hist())        ## @ histogram  ## healthy chol is <200
print(plt.show())
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
Axes(0.125,0.11;0.775x0.77)
```



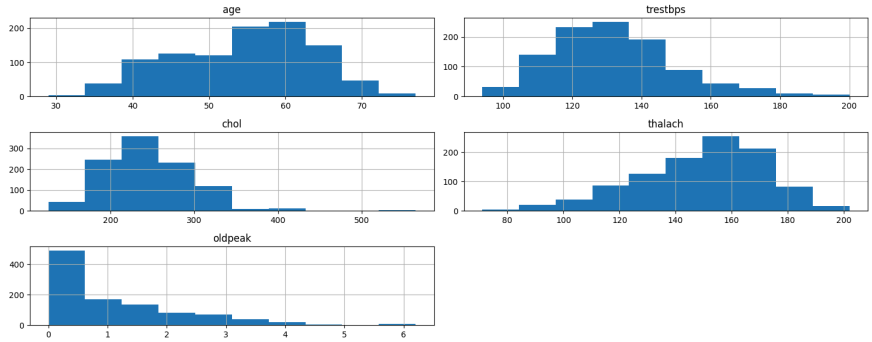

```
None
```

19. Plot Continuous Variables i.e columns having continuous values and categorical values

```
##

print(data.columns)

cate_val=[]     ##categorical values
cont_val=[]      ##continuous values

for column in data.columns:
```

```
        if data[column].nunique() <=10:
            cate_val.append(column)
        else:
            cont_val.append(column)

    print(cate_val)
    print(cont_val)

    ## we need continuous variable, so we will plot cont_val
    print(data.hist(cont_val,figsize=(15,6)))   ## to change size
    print(plt.tight_layout())    ## to avoid overlapping   ## @ multiple histogram
    print(plt.show())
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
[[<Axes: title={'center': 'age'}> <Axes: title={'center': 'trestbps'}>]
 [<Axes: title={'center': 'chol'}> <Axes: title={'center': 'thalach'}>]
 [<Axes: title={'center': 'oldpeak'}> <Axes: >]]
None
```




```
None
```