In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
df = pd.read_csv('/content/heart.csv')
```

In [ ]:
```
df
```

Out[ ]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG
0	40	M	ATA	140	289	0	Normal
1	49	F	NAP	160	180	0	Normal
2	37	M	ATA	130	283	0	ST
3	48	F	ASY	138	214	0	Normal
4	54	M	NAP	150	195	0	Normal
...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal
914	68	M	ASY	144	193	1	Normal
915	57	M	ASY	130	131	0	Normal
916	57	F	ATA	130	236	0	LVH
917	38	M	NAP	138	175	0	Normal

918 rows × 12 columns

EDA

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

Out[ ]:
```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingB
S',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

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

Out[ ]:
```
(918, 12)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 918 entries, 0 to 917
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Age             918 non-null    int64  
 1   Sex             918 non-null    object 
 2   ChestPainType   918 non-null    object 
 3   RestingBP       918 non-null    int64  
 4   Cholesterol     918 non-null    int64  
 5   FastingBS       918 non-null    int64  
 6   RestingECG      918 non-null    object 
 7   MaxHR           918 non-null    int64  
 8   ExerciseAngina  918 non-null    object 
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object 
 11  HeartDisease    918 non-null    int64  
dtypes: float64(1), int64(6), object(5)
memory usage: 86.2+ KB
```

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

Out[ ]:

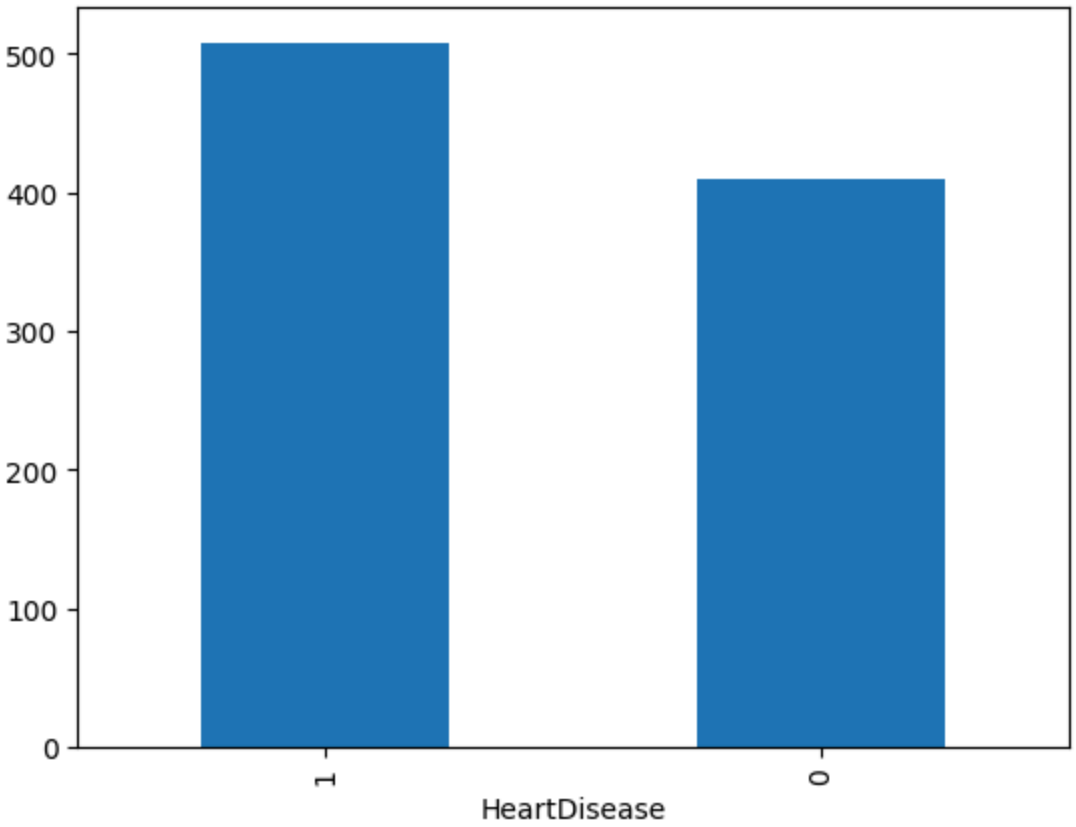
	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
np.int64(0)
```

In [ ]:
```
df['HeartDisease'].value_counts().plot(kind="bar")
```

Out[ ]:
```
<Axes: xlabel='HeartDisease'>
```



In [ ]:
```
df.isnull().sum()
```

Out[ ]:

	0
Age	0
Sex	0
ChestPainType	0
RestingBP	0
Cholesterol	0
FastingBS	0
RestingECG	0
MaxHR	0
ExerciseAngina	0
Oldpeak	0
ST_Slope	0
HeartDisease	0

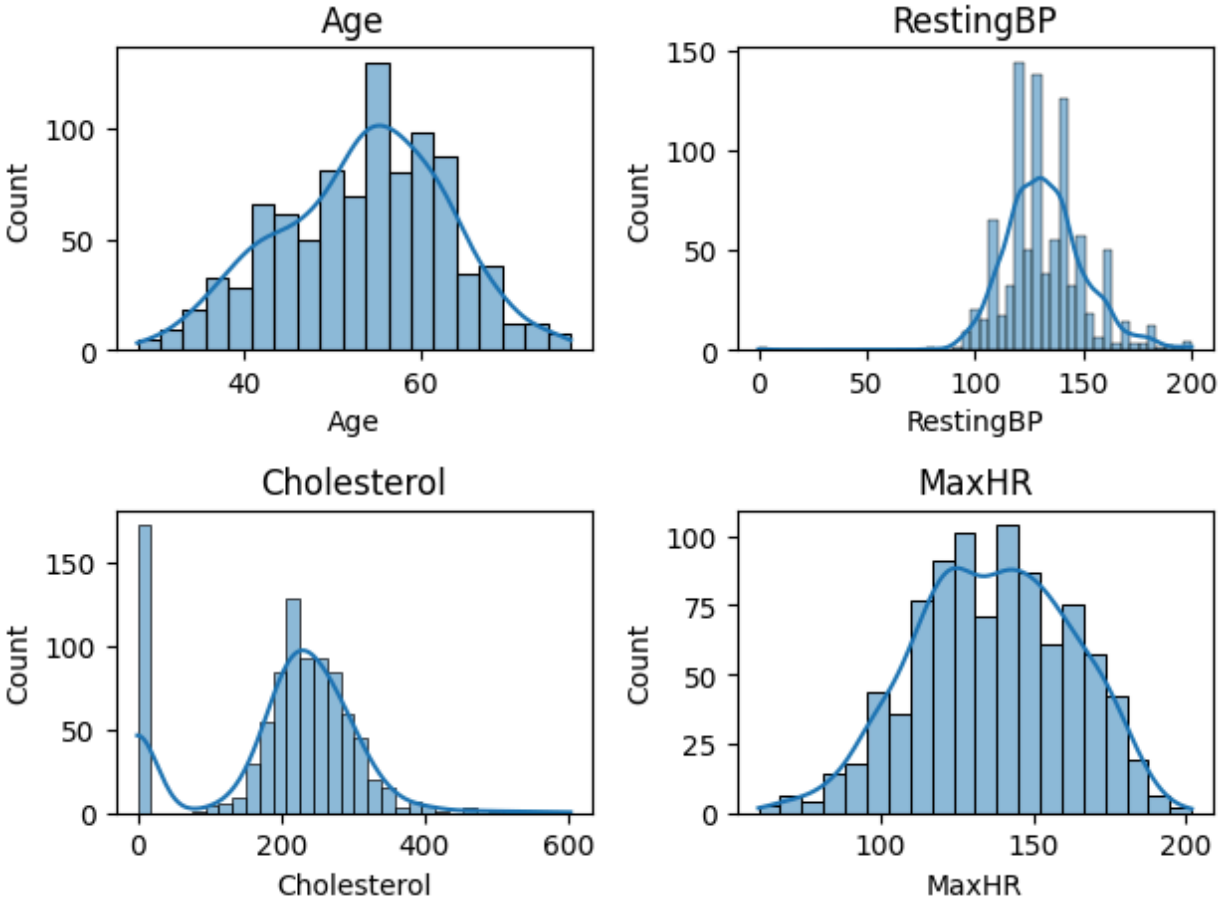
dtype: int64

The data is cleaned till now but is wronged

In [ ]:
```
def plotting(var,num):
    plt.subplot(2,2,num)
    sns.histplot(df[var],kde=True)
    plt.title(var)

plotting('Age',1)
plotting('MaxHR',4)
plotting('Cholesterol',3)
plotting('RestingBP',2)

plt.tight_layout()
```



In [ ]:
```
df['Cholesterol'].value_counts() #Here 172 person has 0 cholestrol that's not
```

Out[ ]:

Cholesterol	count
0	172
254	11
220	10
223	10
204	9
...	...
353	1
278	1
157	1
176	1
131	1

222 rows × 1 columns

dtype: int64

In [ ]:
```
ch_mean = df.loc[df['Cholesterol'] != 0,'Cholesterol'].mean()
df['Cholesterol'] = df['Cholesterol'].replace(0,ch_mean)
df['Cholesterol'] = df['Cholesterol'].round(2)
```

In [ ]:
```
df['Cholesterol'].value_counts()
```

Out[ ]:

Cholesterol	count
244.64	172
254.00	11
220.00	10
223.00	10
204.00	9
...	...
353.00	1
278.00	1
157.00	1
176.00	1
131.00	1

222 rows × 1 columns

dtype: int64

In [ ]:
```
resting_bp_mean = df.loc[df['RestingBP'] != 0, 'RestingBP'].mean()
df['RestingBP'] = df['RestingBP'].replace(0,ch_mean)
df['RestingBP'] = df['RestingBP'].round(2)
```

In [ ]:
```
df['RestingBP'].value_counts()
```

Out[ ]:

RestingBP	count
120.0	132
130.0	118
140.0	107
110.0	58
150.0	55
...	...
101.0	1
117.0	1
192.0	1
129.0	1
164.0	1

67 rows × 1 columns

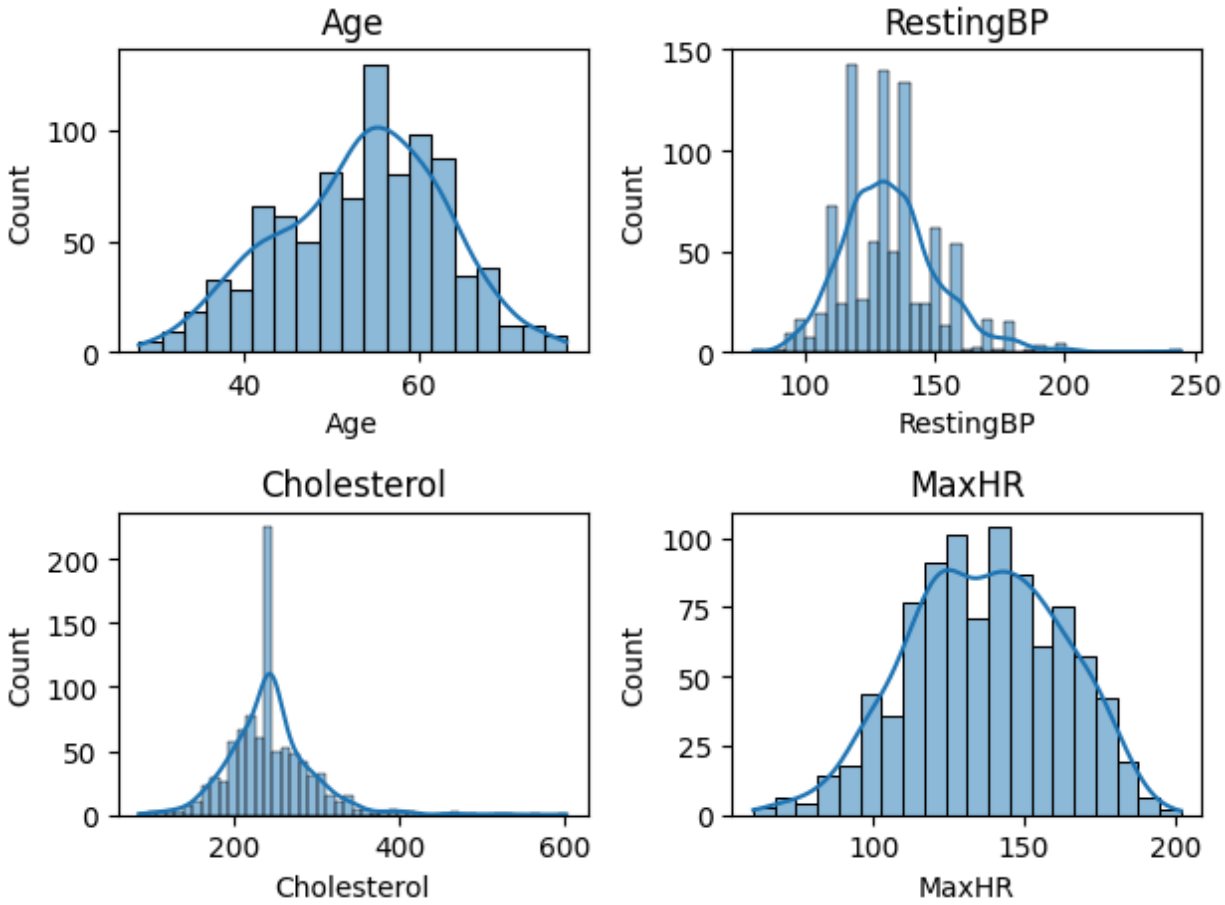
dtype: int64

In [ ]:

```
def plotting(var,num):
    plt.subplot(2,2,num)
    sns.histplot(df[var],kde=True)
    plt.title(var)

plotting('Age',1)
plotting('MaxHR',4)
plotting('Cholesterol',3)
plotting('RestingBP',2)

plt.tight_layout()
```



In [ ]:
```
pip install sheryanalysis==0.1.0
```

```
Collecting sheryanalysis==0.1.0
  Downloading sheryanalysis-0.1.0-py3-none-any.whl.metadata (574 bytes)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.12/dist-
packages (from sheryanalysis==0.1.0) (2.2.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.12/dist-
packages (from sheryanalysis==0.1.0) (2.0.2)
Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.1
2/dist-packages (from sheryanalysis==0.1.0) (1.6.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python
3.12/dist-packages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-p
ackages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dis
t-packages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-p
ackages (from scikit-learn>=0.22.0->sheryanalysis==0.1.0) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn>=0.22.0->sheryanalysis==0.1.0) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.1
2/dist-packages (from scikit-learn>=0.22.0->sheryanalysis==0.1.0) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packa
ges (from python-dateutil>=2.8.2->pandas>=1.0.0->sheryanalysis==0.1.0) (1.17.0)
Downloading sheryanalysis-0.1.0-py3-none-any.whl (10 kB)
Installing collected packages: sheryanalysis
Successfully installed sheryanalysis-0.1.0
```

In [ ]:
```
import sheryanalysis as sh
sh.analyze(df)
```

```
📊 Basic Analysis Report
INFO:sheryanalysis:
📊 Basic Analysis Report
------------------------------------------------------------
INFO:sheryanalysis:------------------------------------------------------------
📏 Shape: (918, 12)
INFO:sheryanalysis:📏 Shape: (918, 12)
📋 Columns: ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'Fastin
gBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDis
ease']
INFO:sheryanalysis:📋 Columns: ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Ch
olesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'S
T_Slope', 'HeartDisease']
✅ No null values found
INFO:sheryanalysis:
✅ No null values found
🔠 Categorical Columns: ['Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'Ex
erciseAngina', 'ST_Slope', 'HeartDisease']
INFO:sheryanalysis:
🔠 Categorical Columns: ['Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'Ex
erciseAngina', 'ST_Slope', 'HeartDisease']
🔢 Numerical Columns: ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

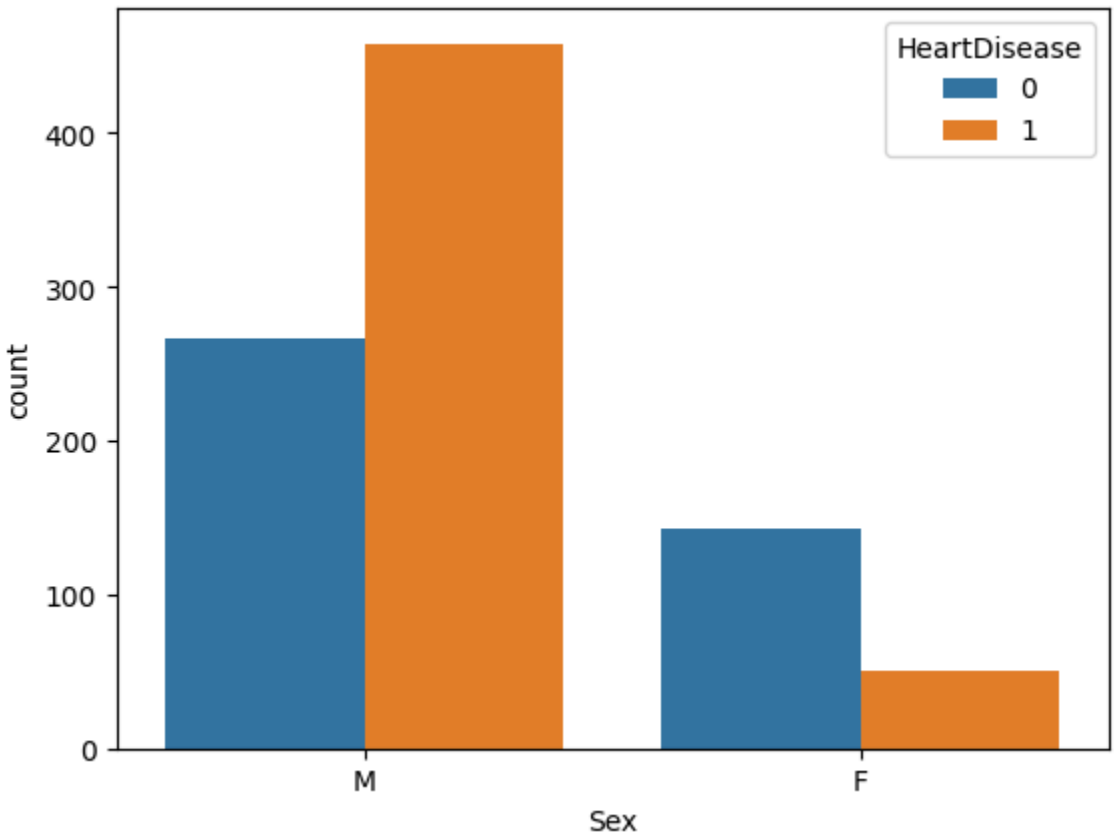
```
INFO:sheryanalysis:
🔷 Numerical Columns: ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

Out[ ]:

```
{'shape': (918, 12),
 'columns': ['Age',
  'Sex',
  'ChestPainType',
  'RestingBP',
  'Cholesterol',
  'FastingBS',
  'RestingECG',
  'MaxHR',
  'ExerciseAngina',
  'Oldpeak',
  'ST_Slope',
  'HeartDisease'],
 'dtypes': {'Age': dtype('int64'),
  'Sex': dtype('O'),
  'ChestPainType': dtype('O'),
  'RestingBP': dtype('float64'),
  'Cholesterol': dtype('float64'),
  'FastingBS': dtype('int64'),
  'RestingECG': dtype('O'),
  'MaxHR': dtype('int64'),
  'ExerciseAngina': dtype('O'),
  'Oldpeak': dtype('float64'),
  'ST_Slope': dtype('O'),
  'HeartDisease': dtype('int64')},
 'null_counts': {'Age': 0,
  'Sex': 0,
  'ChestPainType': 0,
  'RestingBP': 0,
  'Cholesterol': 0,
  'FastingBS': 0,
  'RestingECG': 0,
  'MaxHR': 0,
  'ExerciseAngina': 0,
  'Oldpeak': 0,
  'ST_Slope': 0,
  'HeartDisease': 0},
 'total_rows': 918,
 'column_types': {'categorical': ['Sex',
   'ChestPainType',
   'FastingBS',
   'RestingECG',
   'ExerciseAngina',
   'ST_Slope',
   'HeartDisease'],
  'numerical': ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak'],
  'datetime': [],
  'text': []}}
```

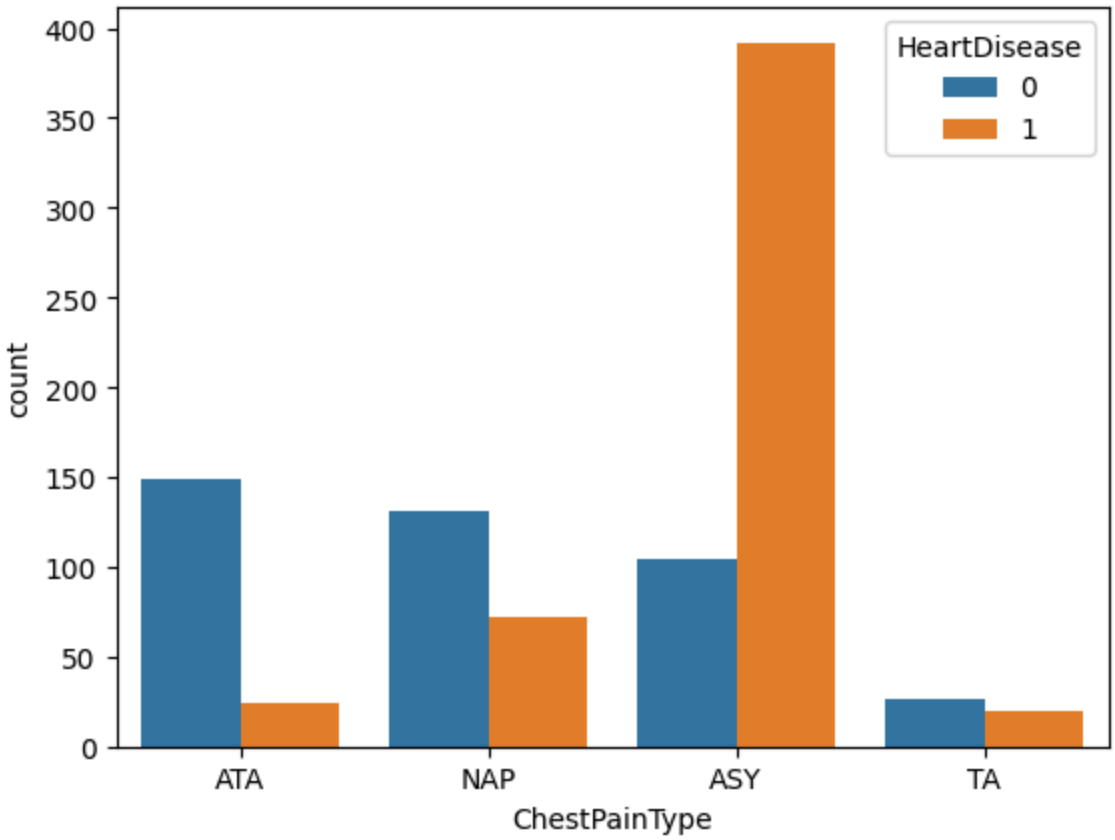

In [ ]:
```
sns.countplot(x= df['Sex'],hue = df['HeartDisease'])
```

Out[ ]: <Axes: xlabel='Sex', ylabel='count'>



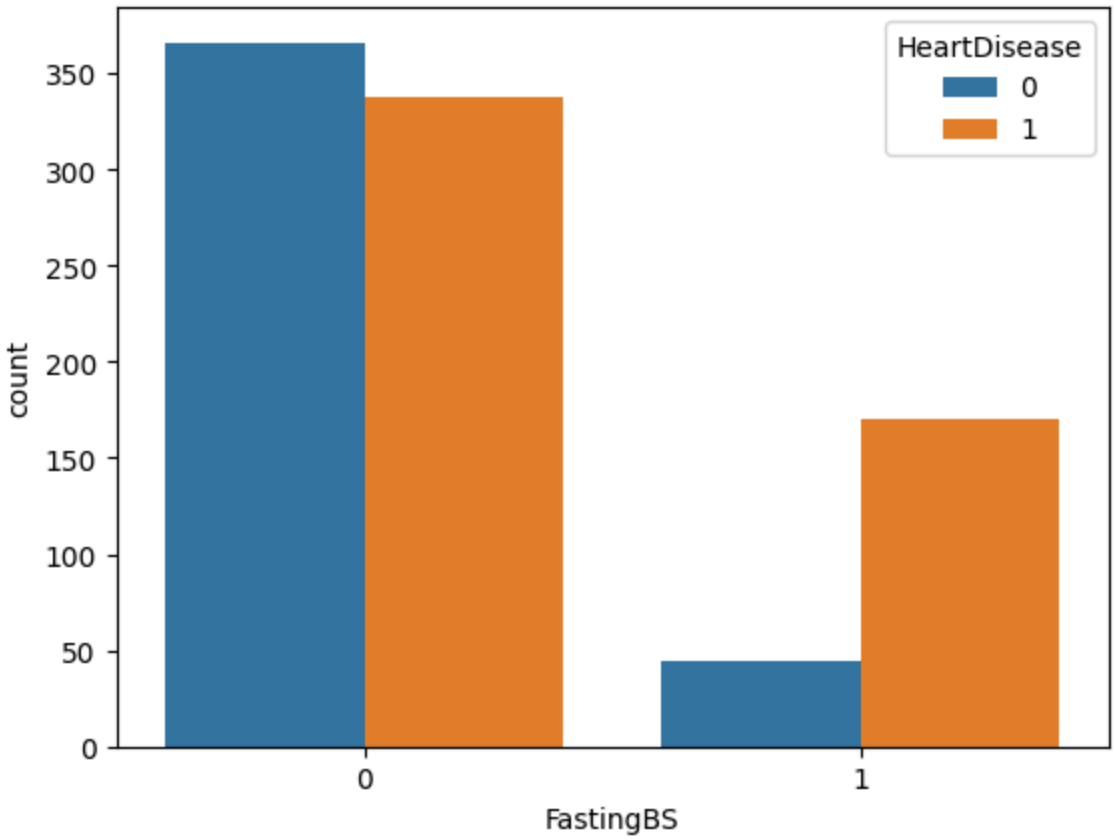
In [ ]:
```
sns.countplot(x= df['ChestPainType'],hue = df['HeartDisease'])
```

Out[ ]: <Axes: xlabel='ChestPainType', ylabel='count'>



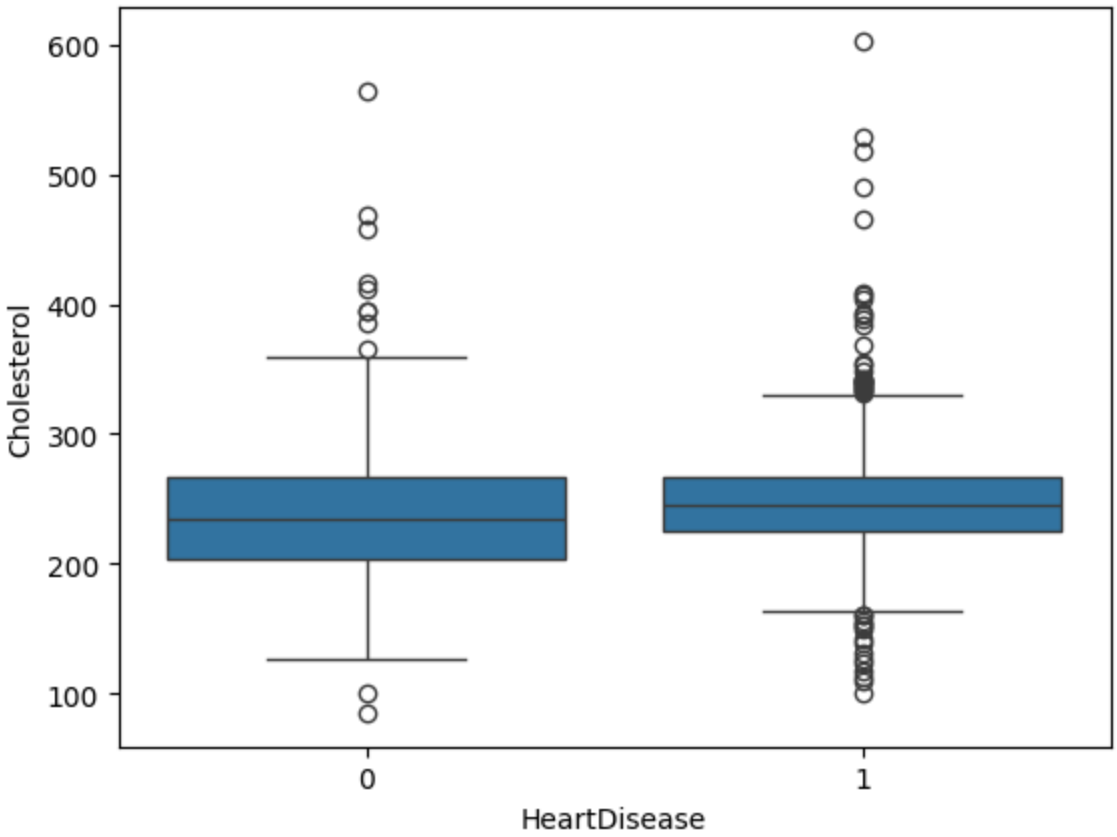
In [ ]:
```
sns.countplot(x= df['FastingBS'],hue = df['HeartDisease'])
```

Out[ ]:
```
<Axes: xlabel='FastingBS', ylabel='count'>
```



In [ ]:
```
sns.boxplot(y = 'Cholesterol', x = 'HeartDisease', data = df)
```

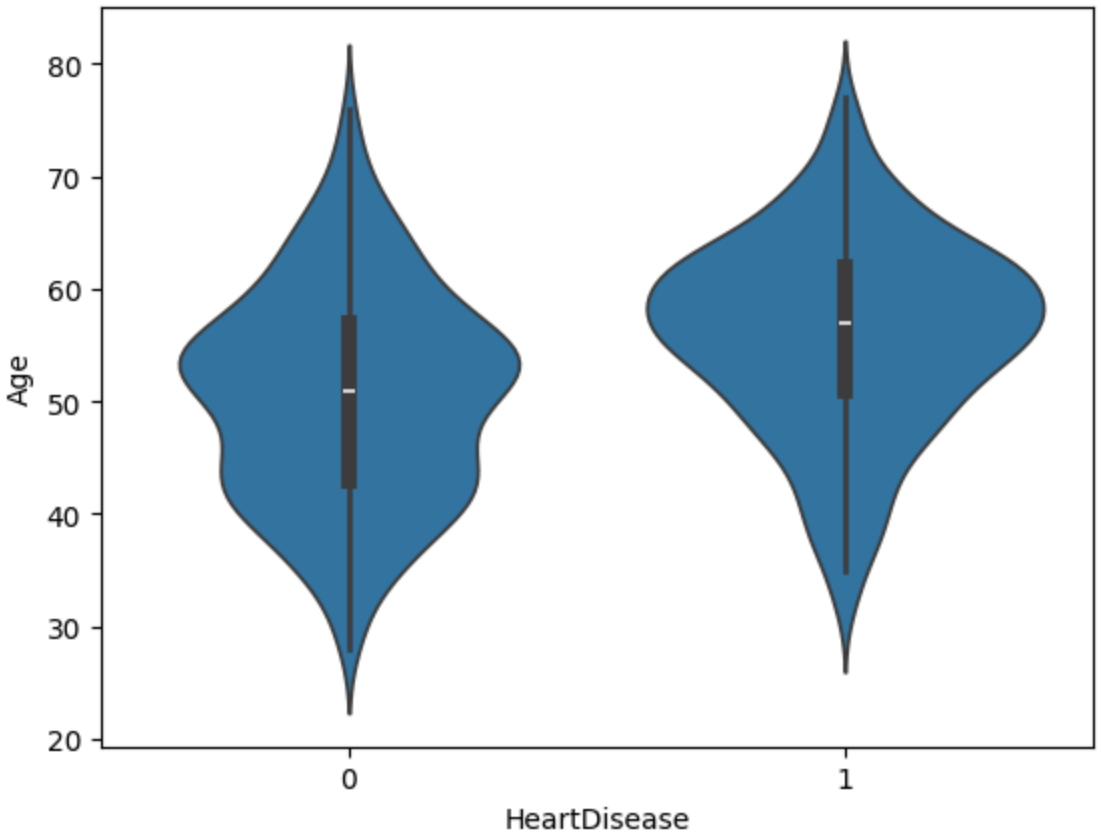
Out[ ]:
```
<Axes: xlabel='HeartDisease', ylabel='Cholesterol'>
```



In [ ]:
```
#A Violin Plot is a data visualization that combines a box plot and a KDE (Ker
# It shows the distribution, probability density, and summary statistics of th

sns.violinplot(x = 'HeartDisease', y = 'Age', data = df)
```

Out[ ]:
```
<Axes: xlabel='HeartDisease', ylabel='Age'>
```

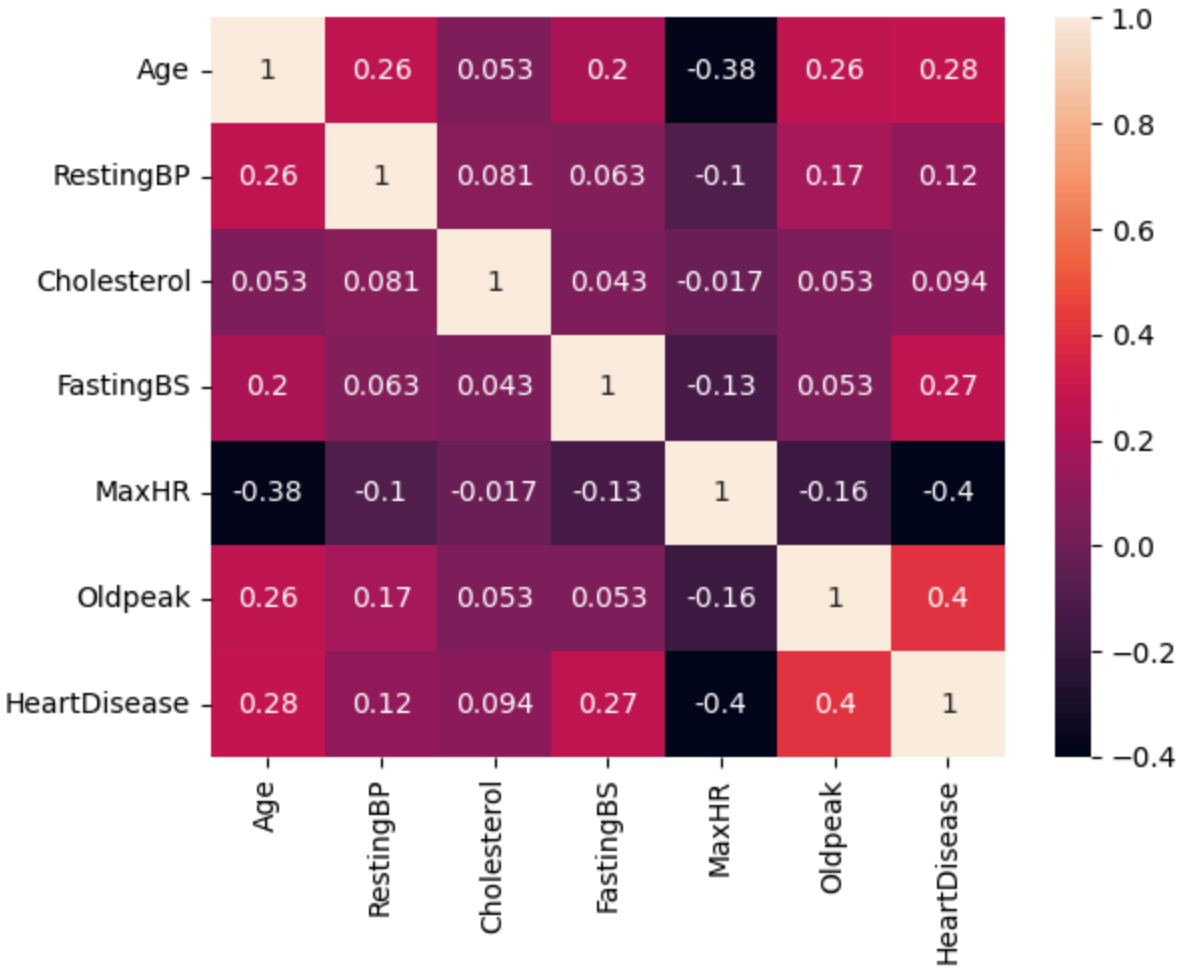


A **Heatmap** is a 2D colored representation of data where values are shown using color intensity. It's most commonly used for:

<> Correlation matrices <> Confusion matrices <> Any data in a matrix format (rows × columns)

In [ ]:
```
sns.heatmap(df.corr(numeric_only=True), annot = True)
```

Out[ ]:
```
<Axes: >
```



In [ ]:
```
#Data Preprocessing

df_encode = pd.get_dummies(df,drop_first=True)
df_encode
```

Out[ ]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	S
0	40	140.0	289.0	0	172	0.0	0	
1	49	160.0	180.0	0	156	1.0	1	
2	37	130.0	283.0	0	98	0.0	0	
3	48	138.0	214.0	0	108	1.5	1	
4	54	150.0	195.0	0	122	0.0	0	
...	...	...	...	...	...	...	...	
913	45	110.0	264.0	0	132	1.2	1	
914	68	144.0	193.0	1	141	3.4	1	
915	57	130.0	131.0	0	115	1.2	1	
916	57	130.0	236.0	0	174	0.0	1	
917	38	138.0	175.0	0	173	0.0	0	

918 rows × 16 columns

In [ ]:
```
df_encode.columns
```

Out[ ]:
```
Index(['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak',
       'HeartDisease', 'Sex_M', 'ChestPainType_ATA', 'ChestPainType_NAP',
       'ChestPainType_TA', 'RestingECG_Normal', 'RestingECG_ST',
       'ExerciseAngina_Y', 'ST_Slope_Flat', 'ST_Slope_Up'],
      dtype='object')
```

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:
```
X = df_encode.drop('HeartDisease',axis = 1)
y = df_encode['HeartDisease']
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, test_size=0.2, random_state=42)
```

In [ ]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [ ]:
```
models = {
```

```
    "Logistic Regression": LogisticRegression(),
    "KNN": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
    "Decision Tree": DecisionTreeClassifier(),
    "SVM (RBF Kernel)": SVC(probability=True)
}
```

In [ ]:
```
results = []
```

In [ ]:
```
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    results.append({
        'Model': name,
        'Accuracy': round(acc, 4),
        'F1 Score': round(f1, 4)
    })
```

In [ ]:
```
results
```

Out[ ]:
```
[{'Model': 'Logistic Regression', 'Accuracy': 0.8913, 'F1 Score': 0.9029},
 {'Model': 'KNN', 'Accuracy': 0.8913, 'F1 Score': 0.9029},
 {'Model': 'Naive Bayes', 'Accuracy': 0.875, 'F1 Score': 0.8844},
 {'Model': 'Decision Tree', 'Accuracy': 0.7554, 'F1 Score': 0.7783},
 {'Model': 'SVM (RBF Kernel)', 'Accuracy': 0.8587, 'F1 Score': 0.875}]
```

In [ ]:
```
import joblib
joblib.dump(models['KNN'],'KNN_heart.pkl')
joblib.dump(scaler,'scaler.pkl')
joblib.dump(X.columns.tolist(),'columns.pkl')
```

Out[ ]:
```
['columns.pkl']
```