In [7]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [9]:
```
df = pd.read_csv(r"C:\Users\ASUS\Downloads\heart.csv")
```

In [11]:
```
df.head()
```

Out[11]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	Exer
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	
4	54	M	NAP	150	195	0	Normal	122	

In [12]:
```
df.columns
```

Out[12]:
```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

In [13]:
```
df.shape
```

Out[13]:
```
(918, 12)
```

In [14]:
```
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 [15]:
```
df.describe()
```

Out[15]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisea
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000	918.0000
mean	53.510893	132.396514	198.799564	0.233115	136.809368	0.887364	0.5533
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.4974
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.0000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.0000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.0000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.0000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.0000

In [16]:
```
df.isnull().sum()
```

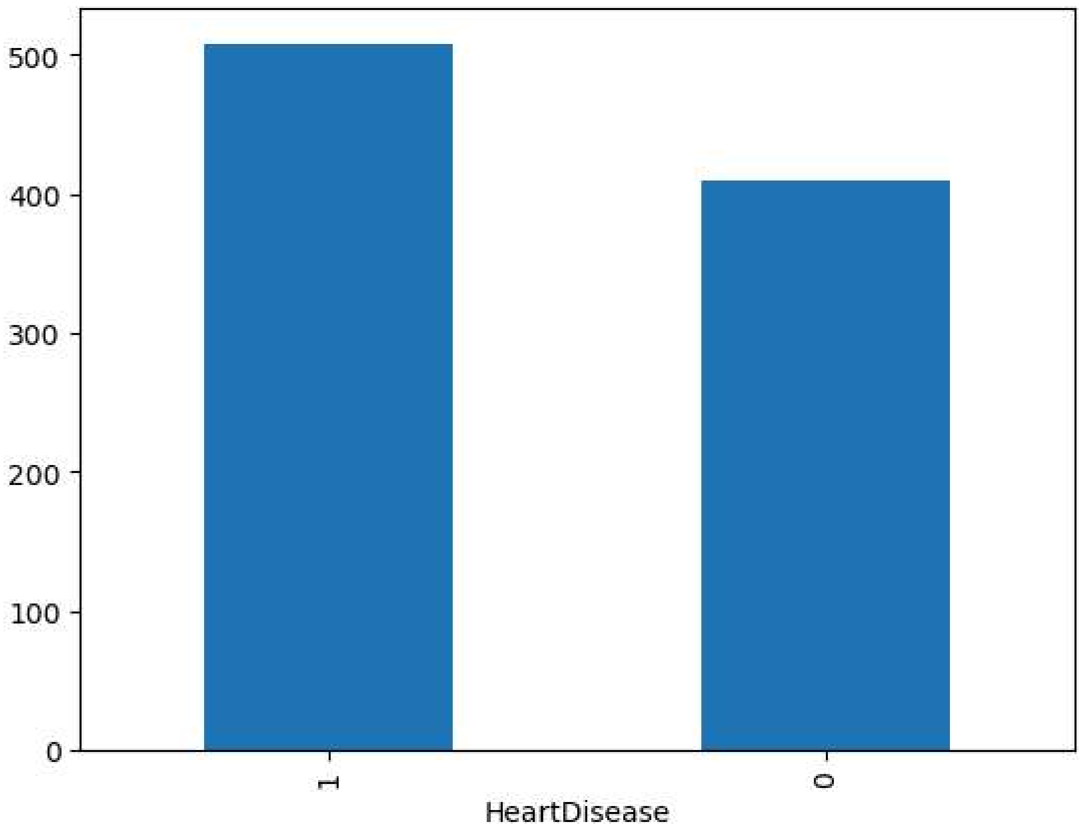
Out[16]:
```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
```

In [17]:
```python
df.duplicated().sum()
```

Out[17]:
```
0
```

In [18]:
```python
df['HeartDisease'].value_counts().plot(kind = 'bar')
```

Out[18]:
```
<Axes: xlabel='HeartDisease'>
```

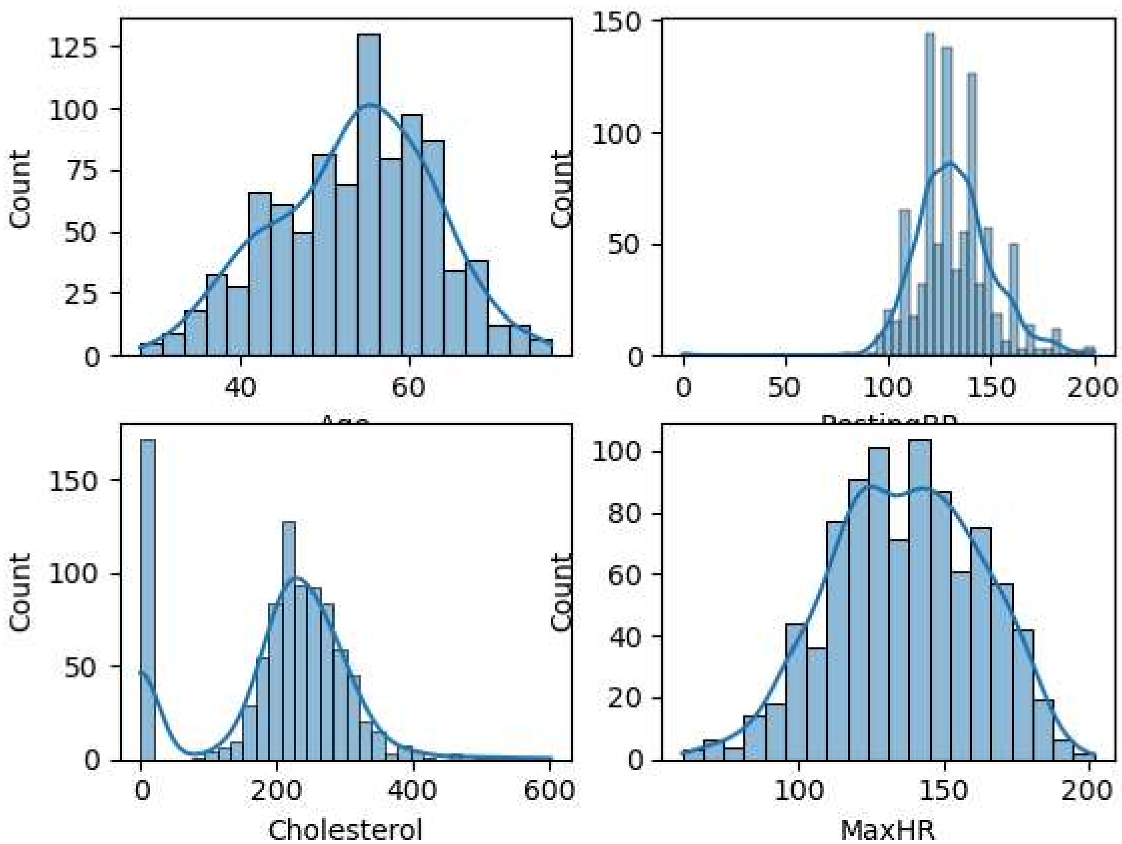


In [19]:
```python
df.isnull().sum()
```

Out[19]:
```
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
```

In [20]:
```python
def plotting(var,num):
    plt.subplot(2,2,num)
    sns.histplot(df[var],kde = True)

plotting('Age',1,)
```

```python
plotting('RestingBP',2)
plotting('Cholesterol',3)
plotting('MaxHR',4)
```



In [21]:
```python
df['Cholesterol'].value_counts()
```

Out[21]:
```
Cholesterol
0      172
254     11
223     10
220     10
230      9
      ...
392      1
316      1
153      1
466      1
131      1
Name: count, Length: 222, dtype: int64
```

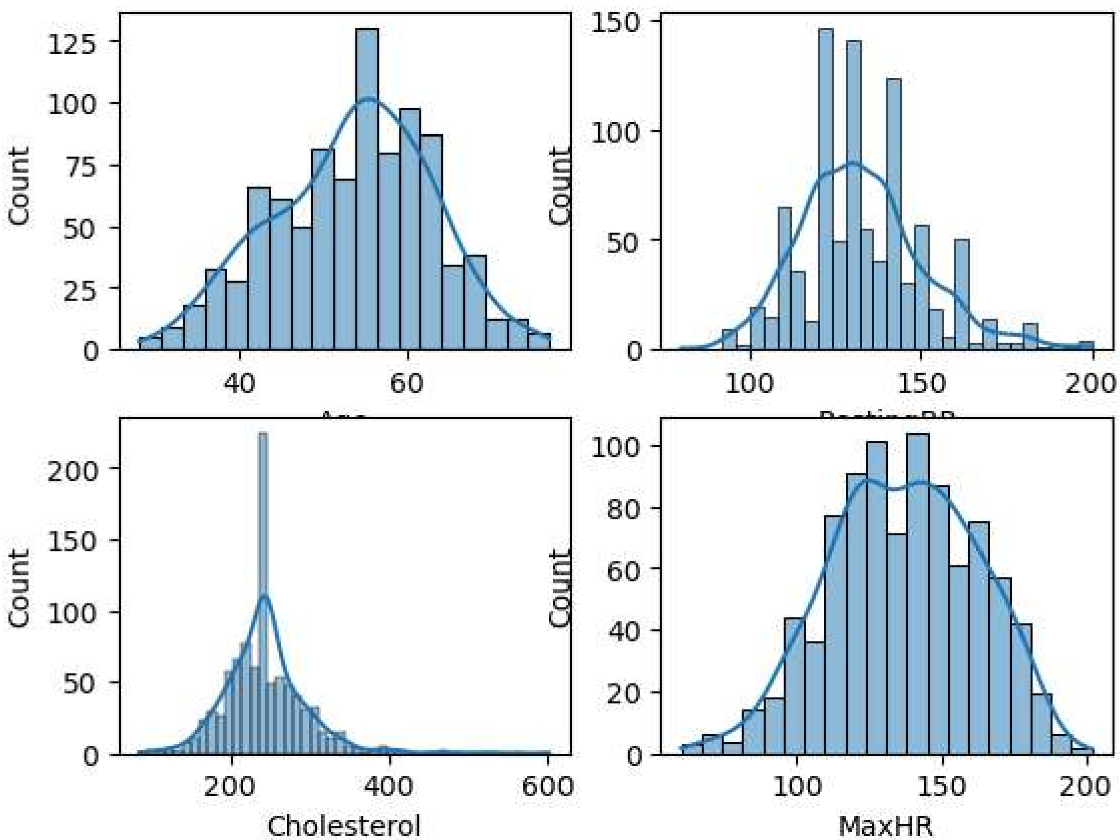
In [22]:
```python
# To fill mean value in cholestrol where 0 occurs
ch_mean = df.loc[df['Cholesterol'] != 0, 'Cholesterol'].mean()
ch_mean
```

Out[22]:
```
244.6353887399464
```

In [23]:
```python
df['Cholesterol'] = df['Cholesterol'].replace(0,ch_mean)
df['Cholesterol'] = df['Cholesterol'].round(2)
```

In [24]:
```python
# To fill mean value in RestingBP where 0 occurs
ch_mean = df.loc[df['RestingBP'] != 0, 'RestingBP'].mean()

df['RestingBP'] = df['RestingBP'].replace(0,ch_mean)
df['RestingBP'] = df['RestingBP'].round(2)
```

In [25]:
```python
def plotting(var,num):
    plt.subplot(2,2,num)
    sns.histplot(df[var],kde = True)

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



In [26]:
```python
pip install sheryanalysis==0.1.0
```

```
Requirement already satisfied: sheryanalysis==0.1.0 in c:\users\asus\anaconda3\lib\site-packages (0.1.0)
Requirement already satisfied: pandas>=1.0.0 in c:\users\asus\anaconda3\lib\site-packages (from sheryanalysis==0.1.0) (2.2.2)
Requirement already satisfied: numpy>=1.18.0 in c:\users\asus\anaconda3\lib\site-packages (from sheryanalysis==0.1.0) (1.26.4)
Requirement already satisfied: scikit-learn>=0.22.0 in c:\users\asus\anaconda3\lib\site-packages (from sheryanalysis==0.1.0) (1.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\asus\anaconda3\lib\site-packages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\asus\anaconda3\lib\site-packages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\asus\anaconda3\lib\site-packages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2023.3)
Requirement already satisfied: scipy>=1.6.0 in c:\users\asus\anaconda3\lib\site-packages (from scikit-learn>=0.22.0->sheryanalysis==0.1.0) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\asus\anaconda3\lib\site-packages (from scikit-learn>=0.22.0->sheryanalysis==0.1.0) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\asus\anaconda3\lib\site-packages (from scikit-learn>=0.22.0->sheryanalysis==0.1.0) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\users\asus\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.0.0->sheryanalysis==0.1.0) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [34]:
```
import sheryanalysis as sh
```

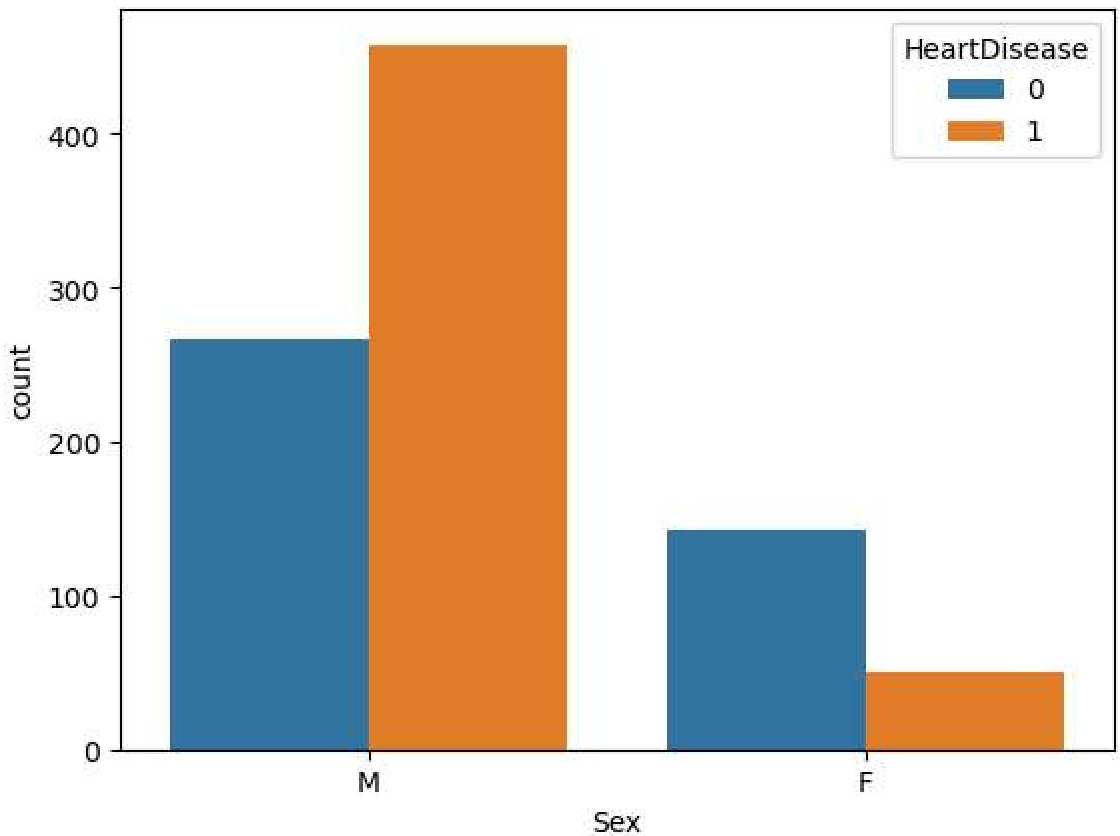
In [37]:
```
sh.analyze(df)
```

```
🔍 Basic Analysis Report
------------------------------------------------------------
📏 Shape: (918, 12)
🧱 Columns: ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']

✅ No null values found

🔠 Categorical Columns: ['Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope', 'HeartDisease']

🔢 Numerical Columns: ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

Out[37]:
```
{'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': []}}
```

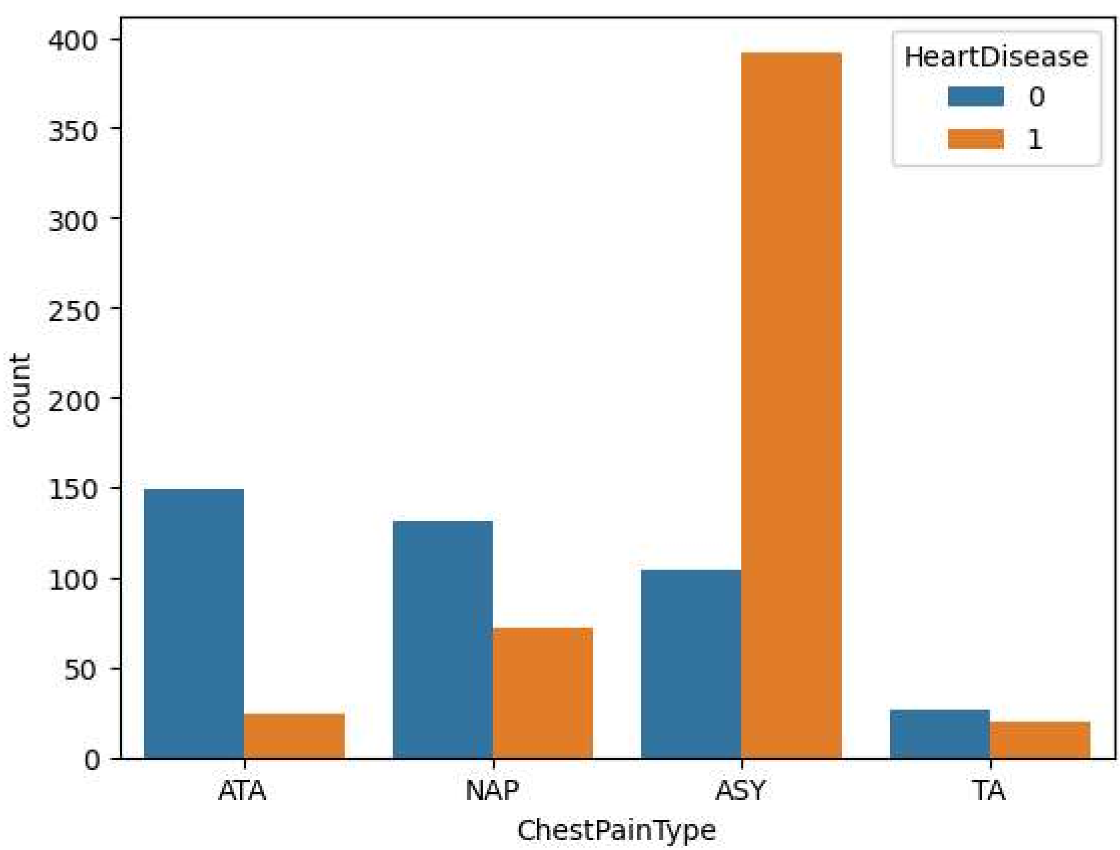
In [38]:
```
sns.countplot(x = df['Sex'],hue = df['HeartDisease'])
```

Out[38]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



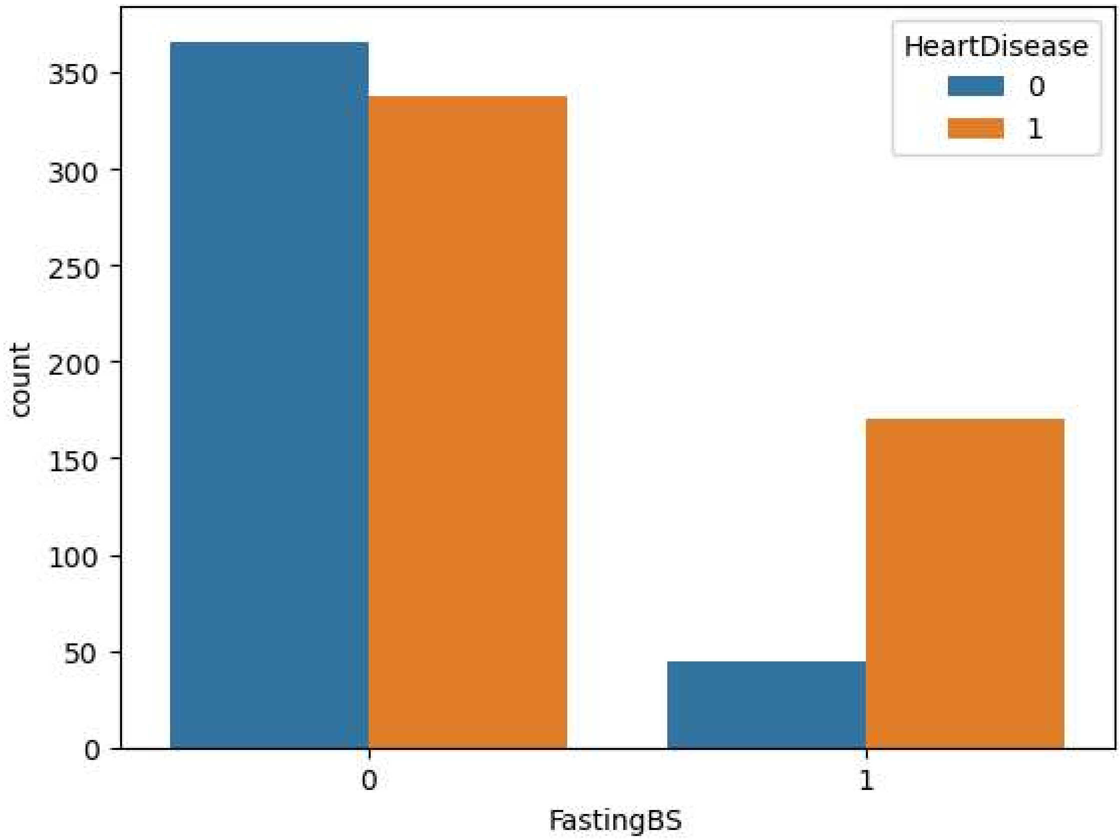
In [39]:
```
sns.countplot(x = df['ChestPainType'],hue = df['HeartDisease'])
```

Out[39]: <Axes: xlabel='ChestPainType', ylabel='count'>



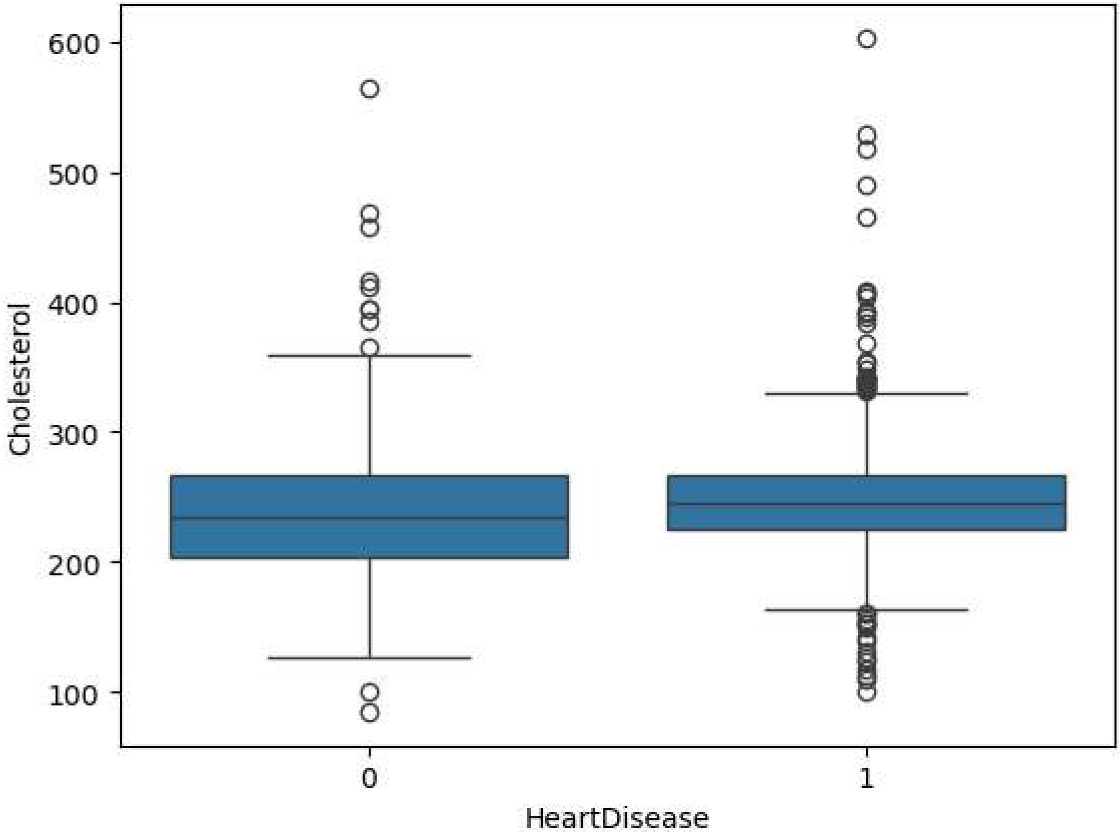
In [40]:
```
sns.countplot(x = df['FastingBS'],hue = df['HeartDisease'])
```

Out[40]: <Axes: xlabel='FastingBS', ylabel='count'>



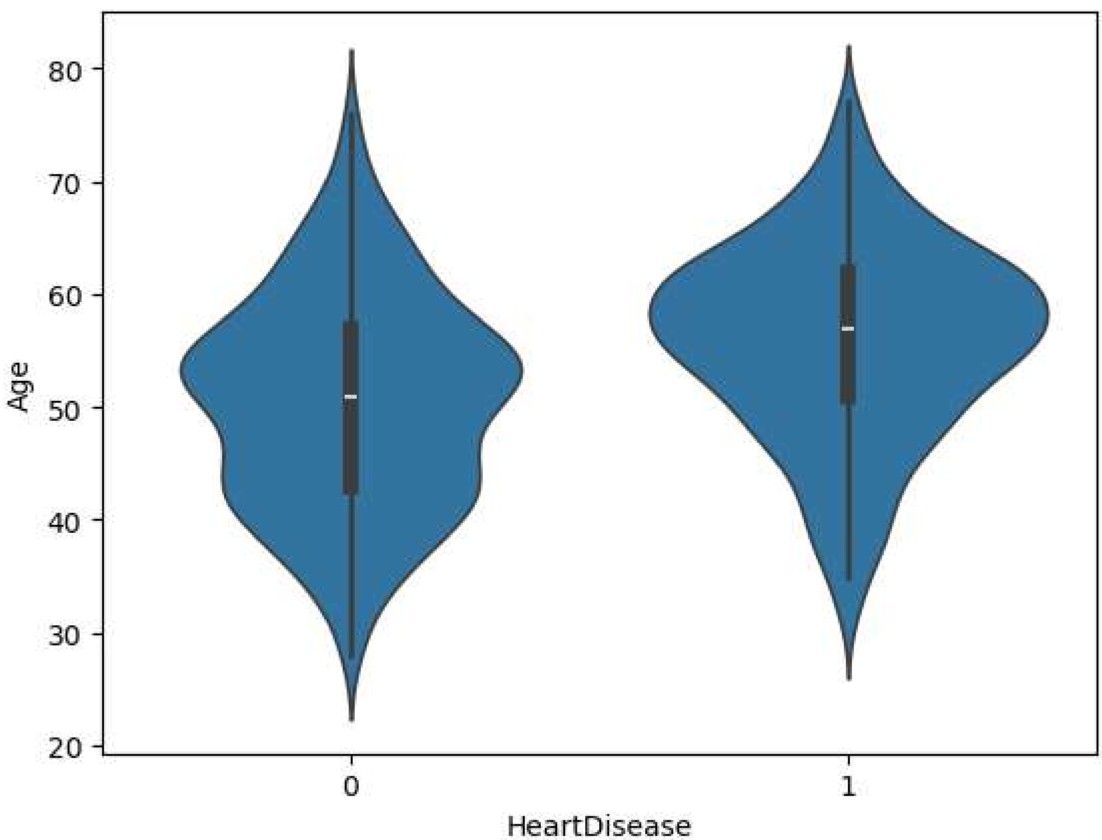
In [41]:
```
sns.boxplot(x = 'HeartDisease',y = 'Cholesterol', data = df)
```

Out[41]: <Axes: xlabel='HeartDisease', ylabel='Cholesterol'>



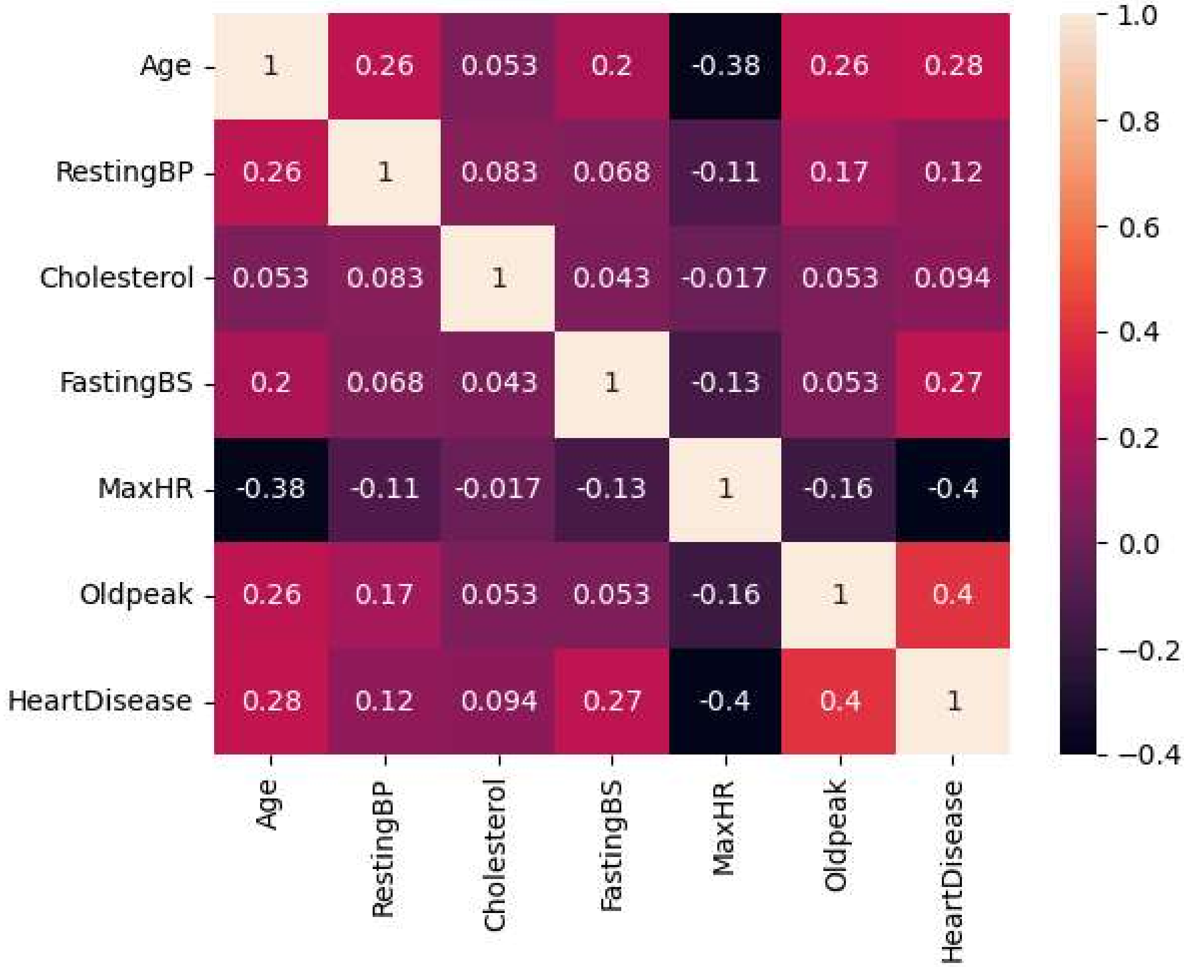
In [47]:
```
sns.violinplot(x = 'HeartDisease', y = 'Age', data = df)
```

Out[47]: <Axes: xlabel='HeartDisease', ylabel='Age'>



```
In [48]: sns.heatmap(df.corr(numeric_only = True), annot = True)
```

Out[48]: <Axes: >



Data preprocessing and cleaning

```
In [50]: df_encode = pd.get_dummies(df, drop_first = True)
```

```
In [51]: df_encode
```

Out[51]:

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

918 rows × 16 columns

In [53]:
```
df_encode = df_encode.astype(int)
```

In [54]:
```
from sklearn.preprocessing import StandardScaler
```

In [55]:
```
numerical_cols = ['Age', 'RestingBP','Cholesterol', 'MaxHR','Oldpeak']
```

In [56]:
```
scaler = StandardScaler()
df_encode[numerical_cols] = scaler.fit_transform(df_encode[numerical_cols])
```

In [57]:
```
df_encode
```

Out[57]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	Sex_
0	-1.433140	0.414885	0.834754	0	1.382928	-0.727592	0	
1	-0.478484	1.527224	-1.210675	0	0.754157	0.282891	1	
2	-1.751359	-0.141284	0.722161	0	-1.525138	-0.727592	0	
3	-0.584556	0.303651	-0.572651	0	-1.132156	0.282891	1	
4	0.051881	0.971054	-0.929194	0	-0.581981	-0.727592	0	
...	...	...	...	...	...	...	...	
913	-0.902775	-1.253622	0.365619	0	-0.188999	0.282891	1	
914	1.536902	0.637353	-0.966725	1	0.164684	2.303858	1	
915	0.370100	-0.141284	-2.130180	0	-0.857069	0.282891	1	
916	0.370100	-0.141284	-0.159813	0	1.461525	-0.727592	1	
917	-1.645286	0.303651	-1.304502	0	1.422226	-0.727592	0	

918 rows × 16 columns

Feature Engineering and extraction

In [59]:

```python
def age_group(age):
    if age < 40:
        return 'Young'
    elif age < 60:
        return 'Middle'
    else:
        return 'Senior'
df['AgeGroup'] = df['Age'].apply(age_group)

# show the count of each group and the updated dataFrame head
df['AgeGroup'].value_counts(), df[['Age', 'AgeGroup']].head()
```

Out[59]:

```
(AgeGroup
 Middle    585
 Senior    253
 Young      80
 Name: count, dtype: int64,
    Age AgeGroup
 0   40   Middle
 1   49   Middle
 2   37    Young
 3   48   Middle
 4   54   Middle)
```

In [60]:
```python
# Step 2: Create Chol_per_Age feature
df['Chol_per_Age'] = df['Cholesterol'] / df['Age']

# Display basic statistics and head of the new feature
df['Chol_per_Age'].describe(), df[['Cholesterol', 'Age', 'Chol_per_Age']].head()
```

Out[60]:
```
(count    918.000000
 mean       4.719937
 std        1.370476
 min        1.486842
 25%        3.877520
 50%        4.450490
 75%        5.317643
 max       16.531250
 Name: Chol_per_Age, dtype: float64,
    Cholesterol  Age  Chol_per_Age
 0        289.0   40      7.225000
 1        180.0   49      3.673469
 2        283.0   37      7.648649
 3        214.0   48      4.458333
 4        195.0   54      3.611111)
```

In [61]:
```python
df_encode
```

Out[61]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	Sex_
0	-1.433140	0.414885	0.834754	0	1.382928	-0.727592	0	
1	-0.478484	1.527224	-1.210675	0	0.754157	0.282891	1	
2	-1.751359	-0.141284	0.722161	0	-1.525138	-0.727592	0	
3	-0.584556	0.303651	-0.572651	0	-1.132156	0.282891	1	
4	0.051881	0.971054	-0.929194	0	-0.581981	-0.727592	0	
...	...	...	...	...	...	...	...	
913	-0.902775	-1.253622	0.365619	0	-0.188999	0.282891	1	
914	1.536902	0.637353	-0.966725	1	0.164684	2.303858	1	
915	0.370100	-0.141284	-2.130180	0	-0.857069	0.282891	1	
916	0.370100	-0.141284	-0.159813	0	1.461525	-0.727592	1	
917	-1.645286	0.303651	-1.304502	0	1.422226	-0.727592	0	

918 rows × 16 columns

In [81]:
```python
df_encode.columns
```

Out[81]:
```
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 [133…
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

In [135…
```
X = df_encode.drop('HeartDisease',axis = 1)
y = df_encode['HeartDisease']
```

In [137…
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.20, random_state
```

In [139…
```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(X_train)
x_test_scaled = scaler.fit_transform(X_test)
```

In [153…
```
models = {
    'logistic Regression': LogisticRegression(),
    'KNN' : KNeighborsClassifier(n_neighbors = 5),
    'Naive Bayes' : GaussianNB(),
    'Decision Tree' : DecisionTreeClassifier(),
    'SVM' : SVC(kernel = 'rbf')
}
```

In [155…
```
result = []
```

In [157…
```
for name,model in models.items():
    model.fit(x_train_scaled,y_train)
    y_pred = model.predict(x_test_scaled)
    accuracy = accuracy_score(y_test,y_pred)
    f1 = f1_score(y_test,y_pred)
    result.append({
    'model': name,
    'Accuracy': round(accuracy,4),
    'f1_score': round(f1,4)
    })
```

In [159…
```
result
```

Out[159…
```
[{'model': 'logistic Regression', 'Accuracy': 0.8696, 'f1_score': 0.8857},
 {'model': 'KNN', 'Accuracy': 0.8641, 'f1_score': 0.8815},
 {'model': 'Naive Bayes', 'Accuracy': 0.8533, 'f1_score': 0.8683},
 {'model': 'Decision Tree', 'Accuracy': 0.7826, 'f1_score': 0.8077},
 {'model': 'SVM', 'Accuracy': 0.8478, 'f1_score': 0.8679}]
```

In [161…

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

Out[161…

```
['columns.pkl']
```

Now we go to deploy model in VS code Heart Disease folder in desktop

In [ ]:

```python

```