In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('/content/heart.csv')
```

In [3]:
```
df.head()
```

Out[3]:

| | 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 |

## EDA

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingB
S',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

In [5]:
```
df.shape
```

Out[5]:
```
(918, 12)
```

In [6]:
```
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 [7]:
```
df.describe()
```

Out[7]:

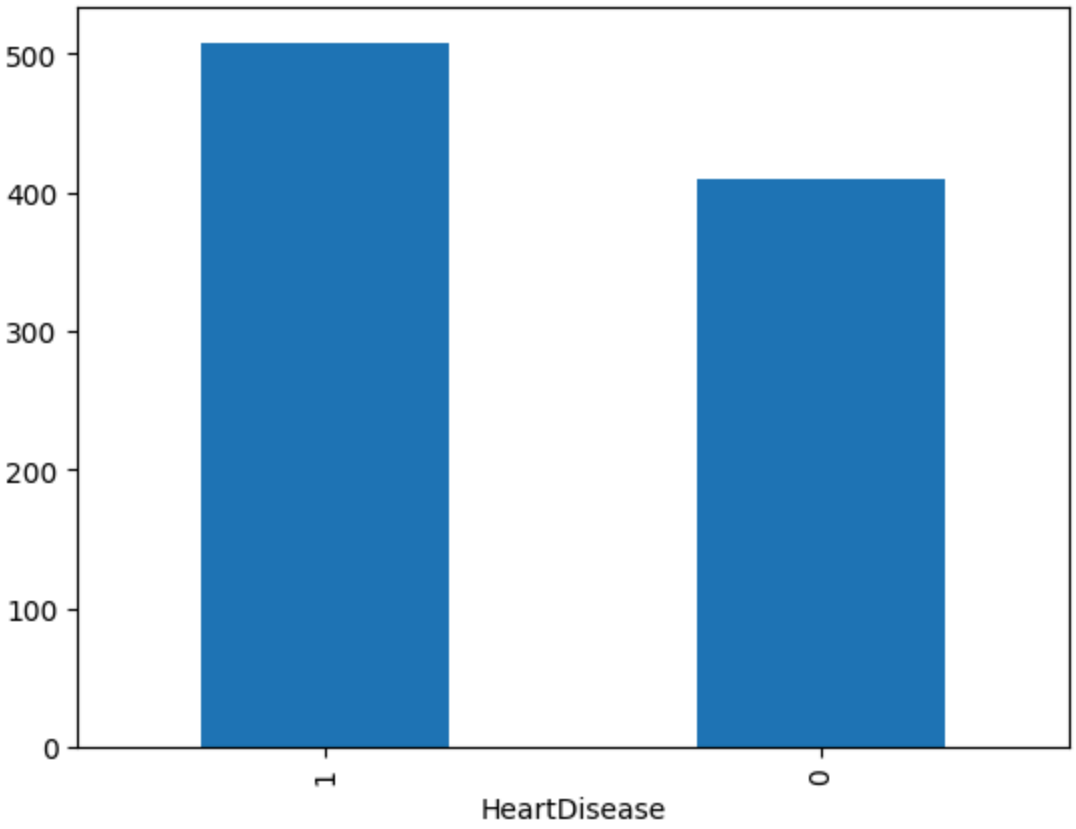
| | Age | RestingBP | Cholesterol | FastingBS | MaxHR | Oldpeak |
|---|---|---|---|---|---|---|
| **count** | 918.000000 | 918.000000 | 918.000000 | 918.000000 | 918.000000 | 918.000000 |
| **mean** | 53.510893 | 132.396514 | 198.799564 | 0.233115 | 136.809368 | 0.887364 |
| **std** | 9.432617 | 18.514154 | 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 [8]:
```
df.duplicated().sum()
```

Out[8]:
```
np.int64(0)
```

In [9]:
```
df['HeartDisease'].value_counts().plot(kind = 'bar')
```

Out[9]:
```
<Axes: xlabel='HeartDisease'>
```




In [10]:
```
df.isnull().sum()
```

Out[10]:

| | 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 |

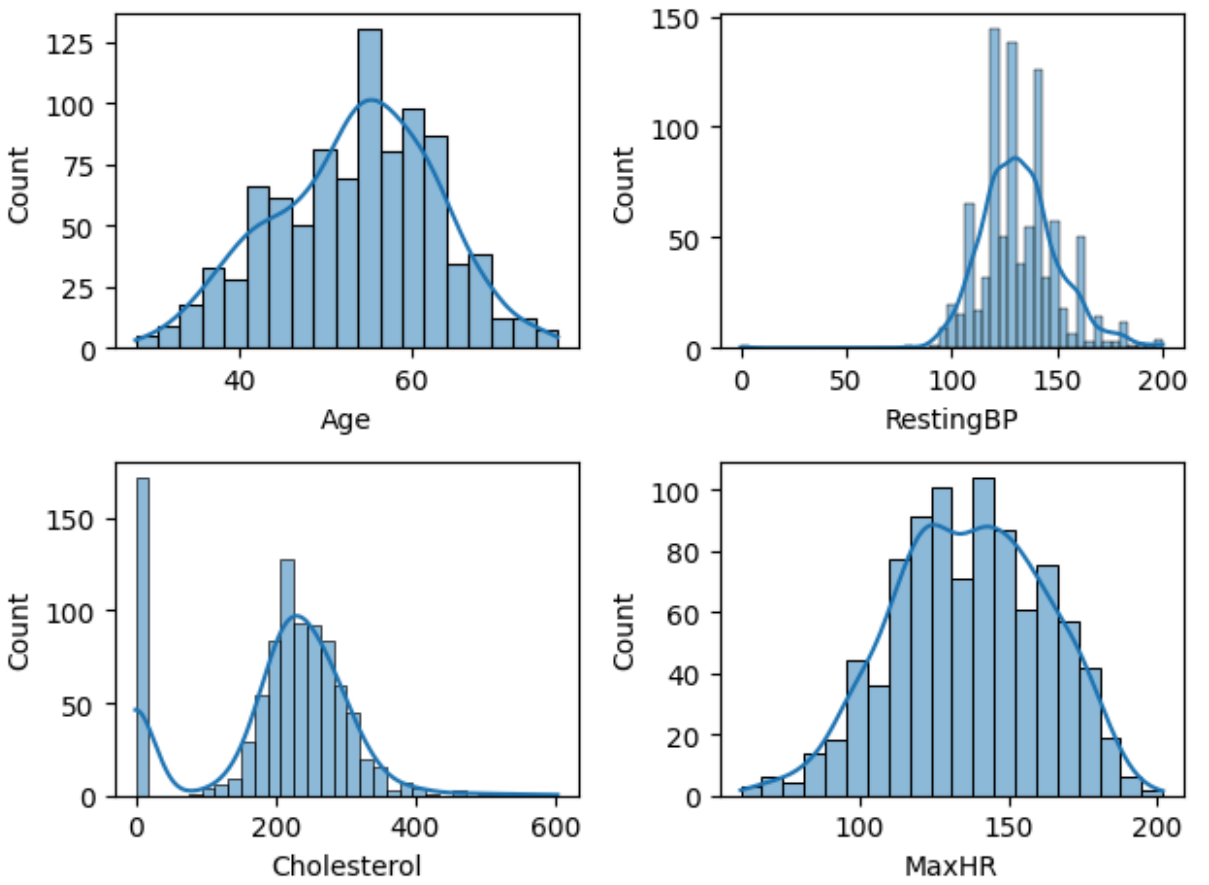
**dtype:** int64

In [11]:
```
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)

plt.tight_layout()
```




In [12]:
```
df['Cholesterol'].value_counts()
```

Out[12]:

| 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 [13]:
```
ch_mean = df.loc[df['Cholesterol'] != 0, 'Cholesterol'].mean()
```

In [14]:
```
ch_mean
```

Out[14]:
```
np.float64(244.6353887399464)
```

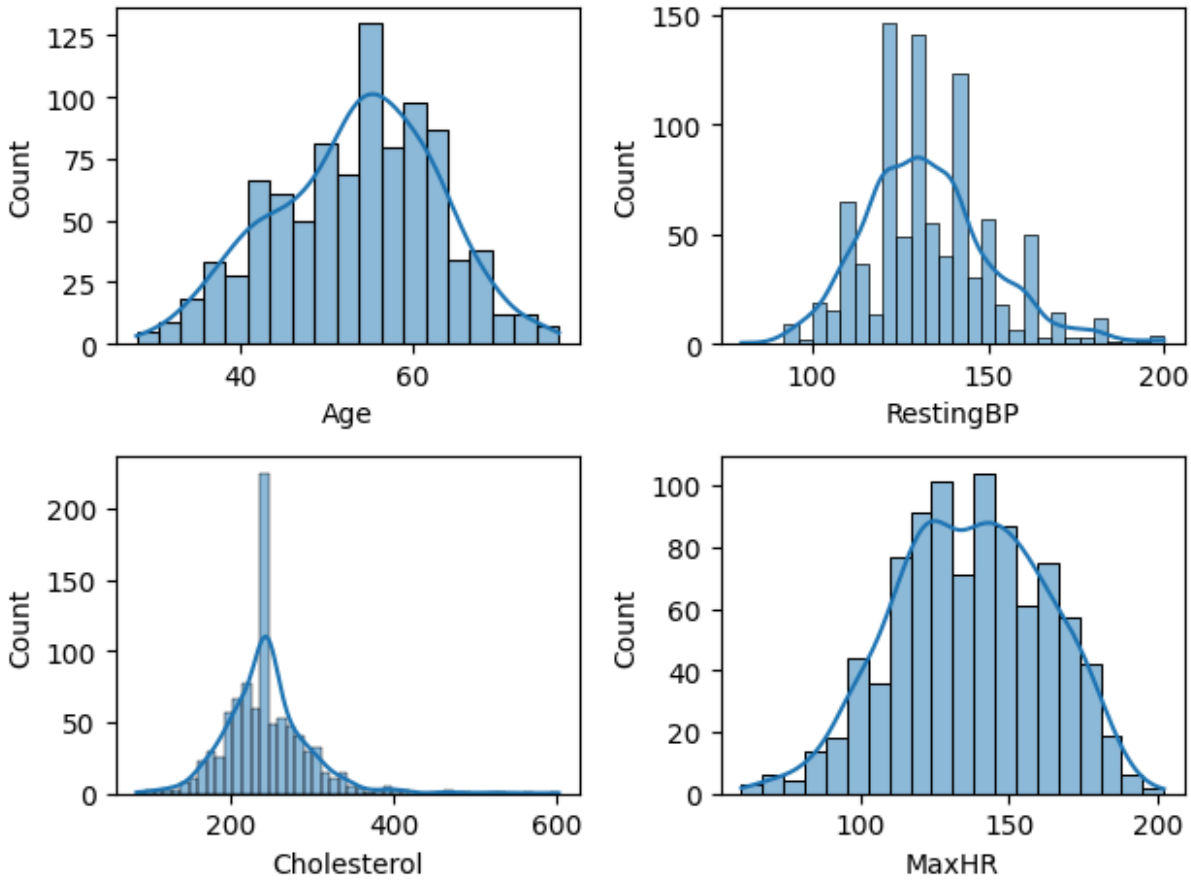
In [15]:
```
df['Cholesterol'] = df['Cholesterol'].replace(0,ch_mean)
df['Cholesterol'] = df['Cholesterol'].round(2)
```

In [16]:
```
resting_bp_mean = df.loc[df['RestingBP'] != 0, 'RestingBP'].mean()
df['RestingBP'] = df['RestingBP'].replace(0,resting_bp_mean)
df['RestingBP'] = df['RestingBP'].round(2)
```

In [17]:
```
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)

plt.tight_layout()
```




In [18]:
```
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/python3.11/dist-packages (from sheryanalysis==0.1.0) (2.2.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.11/dist-packages (from sheryanalysis==0.1.0) (2.0.2)
Requirement already satisfied: scikit-learn>=0.22.0 in /usr/local/lib/python3.11/dist-packages (from sheryanalysis==0.1.0) (1.6.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/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/python3.11/dist-packages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.0->sheryanalysis==0.1.0) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=0.22.0->sheryanalysis==0.1.0) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/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/python3.11/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/python3.11/dist-packages (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 [19]:
```
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', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']
INFO:sheryanalysis:🧱 Columns: ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope', 'HeartDisease']
✅ No null values found
INFO:sheryanalysis:
✅ No null values found
🔠 Categorical Columns: ['Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope', 'HeartDisease']
INFO:sheryanalysis:
🔠 Categorical Columns: ['Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope', 'HeartDisease']
🔢 Numerical Columns: ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

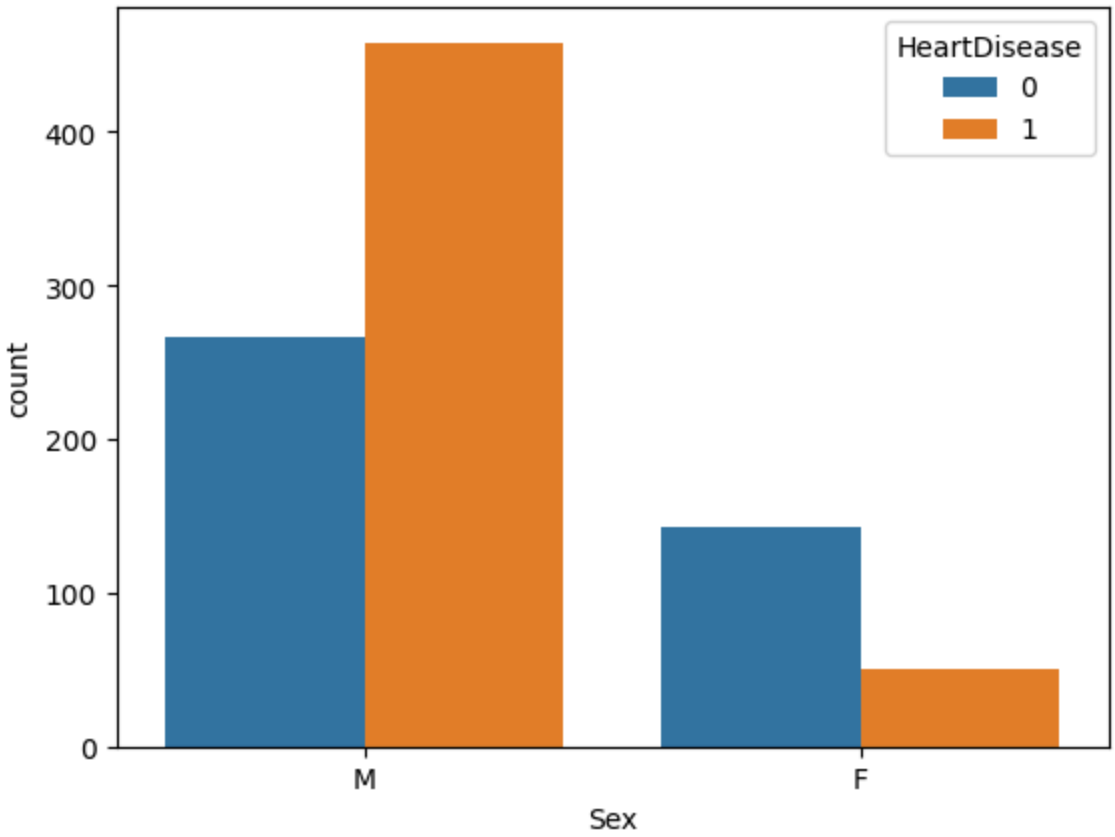
```
INFO:sheryanalysis:
🔢 Numerical Columns: ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpea
k']
```

Out[19]:

```
{'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': []}}
```

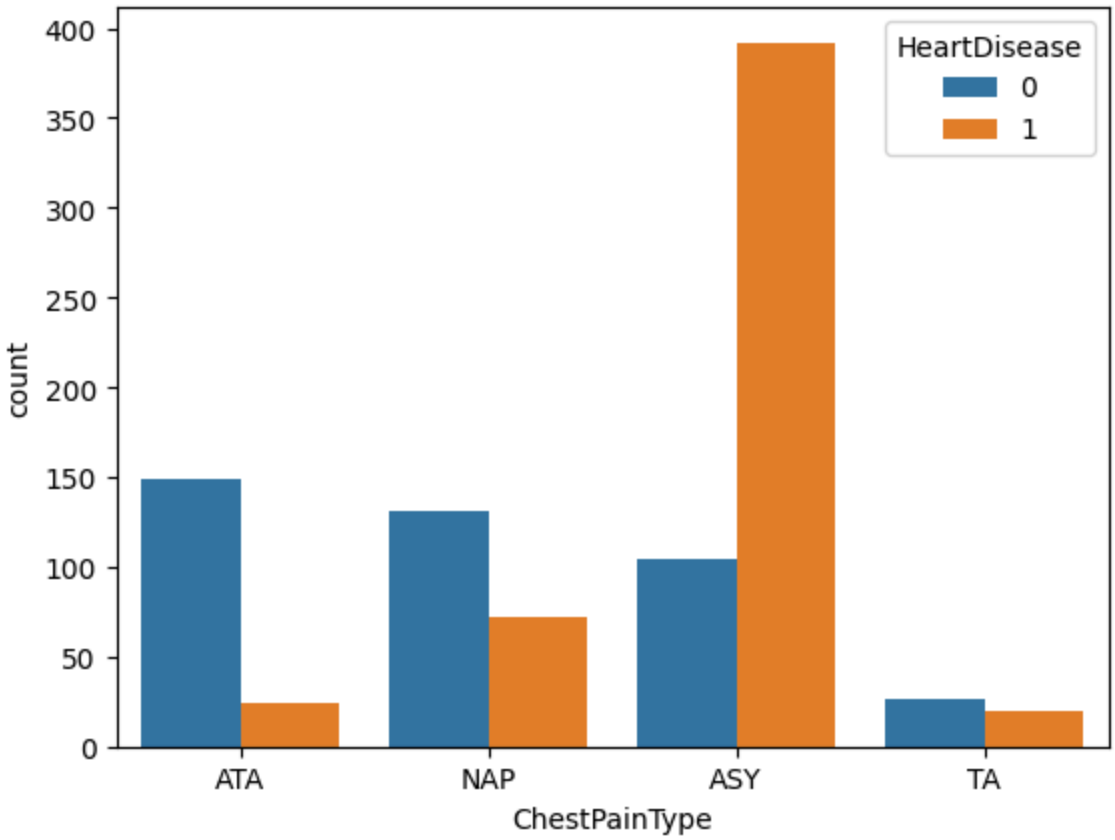

In [20]:
```
sns.countplot(x = df['Sex'], hue = df['HeartDisease'])
```

Out[20]:
```
<Axes: xlabel='Sex', ylabel='count'>
```



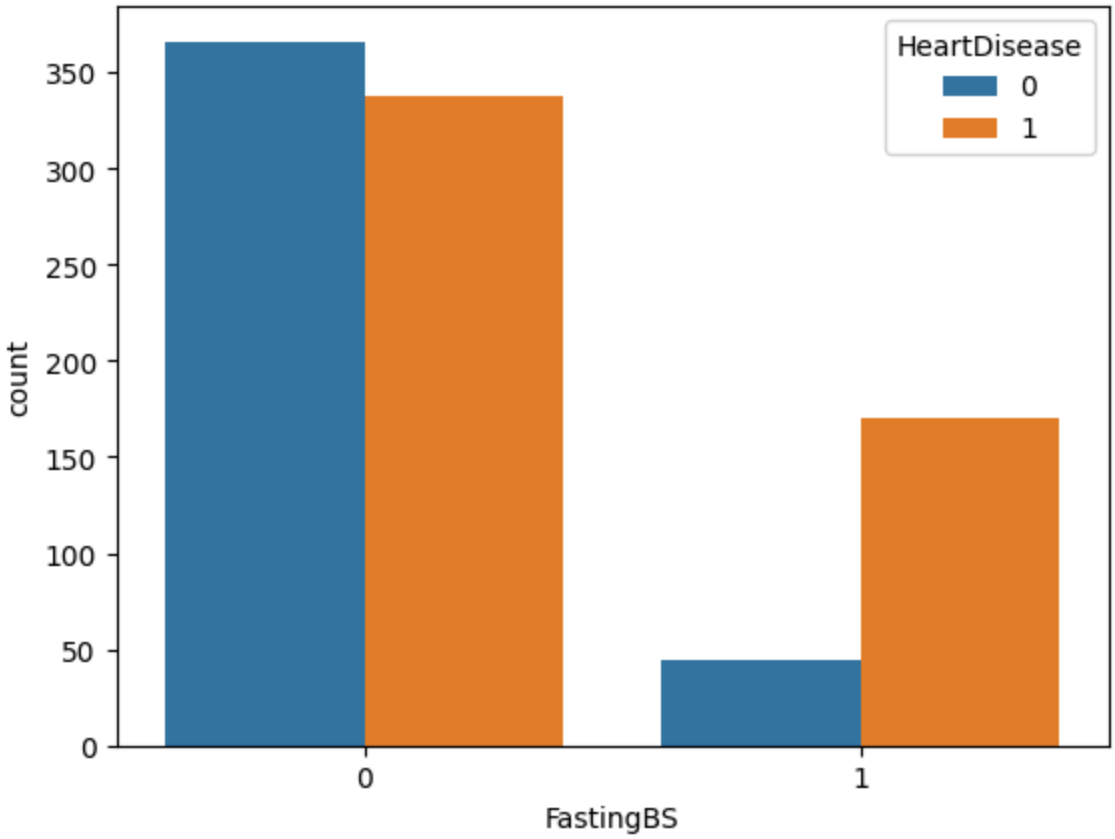

In [21]:
```
sns.countplot(x = df['ChestPainType'], hue = df['HeartDisease'])
```

Out[21]:
```
<Axes: xlabel='ChestPainType', ylabel='count'>
```



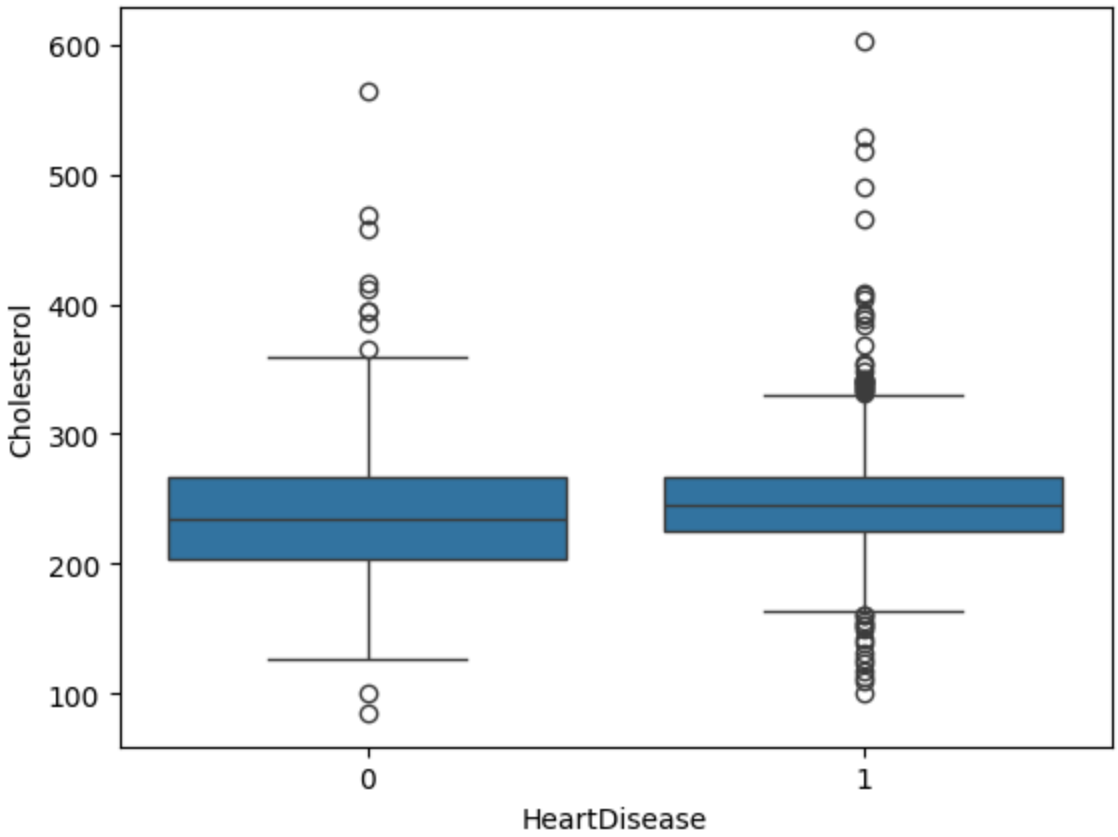
In [22]:
```
sns.countplot(x = df['FastingBS'], hue = df['HeartDisease'])
```

Out[22]:
```
<Axes: xlabel='FastingBS', ylabel='count'>
```



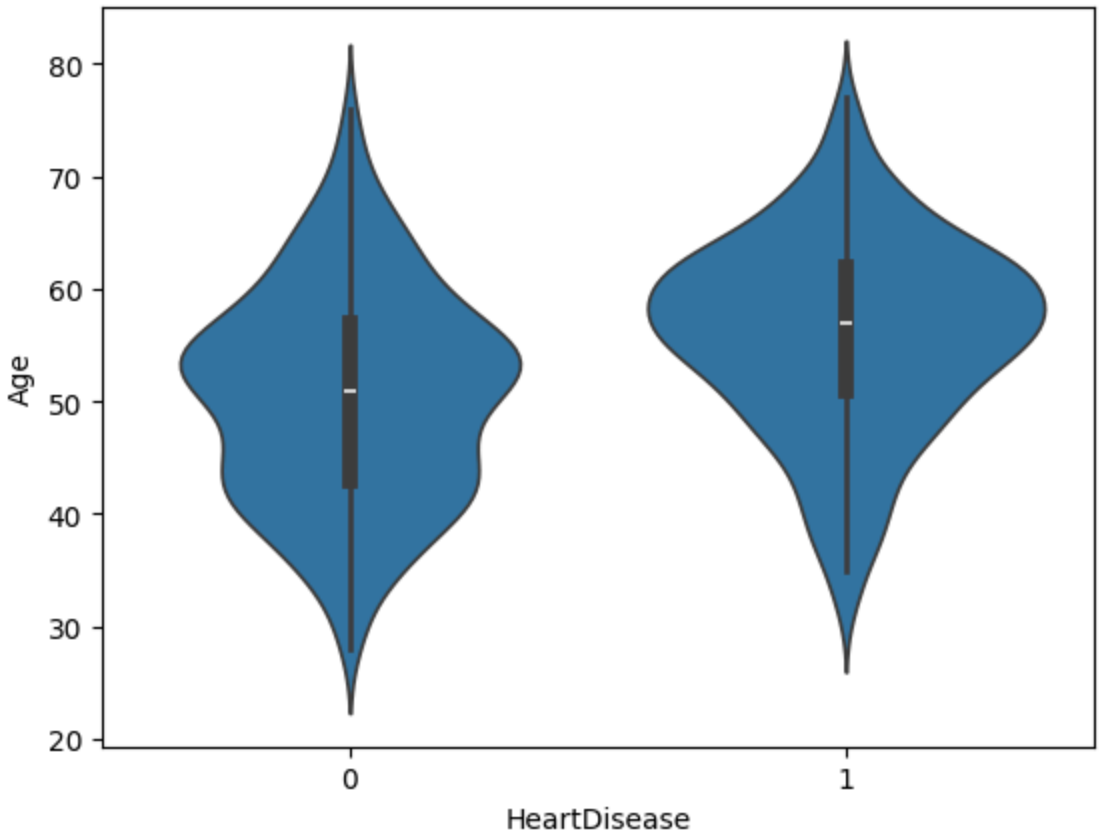

In [23]:
```
sns.boxplot(x = 'HeartDisease', y = 'Cholesterol', data = df)
```

Out[23]:
```
<Axes: xlabel='HeartDisease', ylabel='Cholesterol'>
```



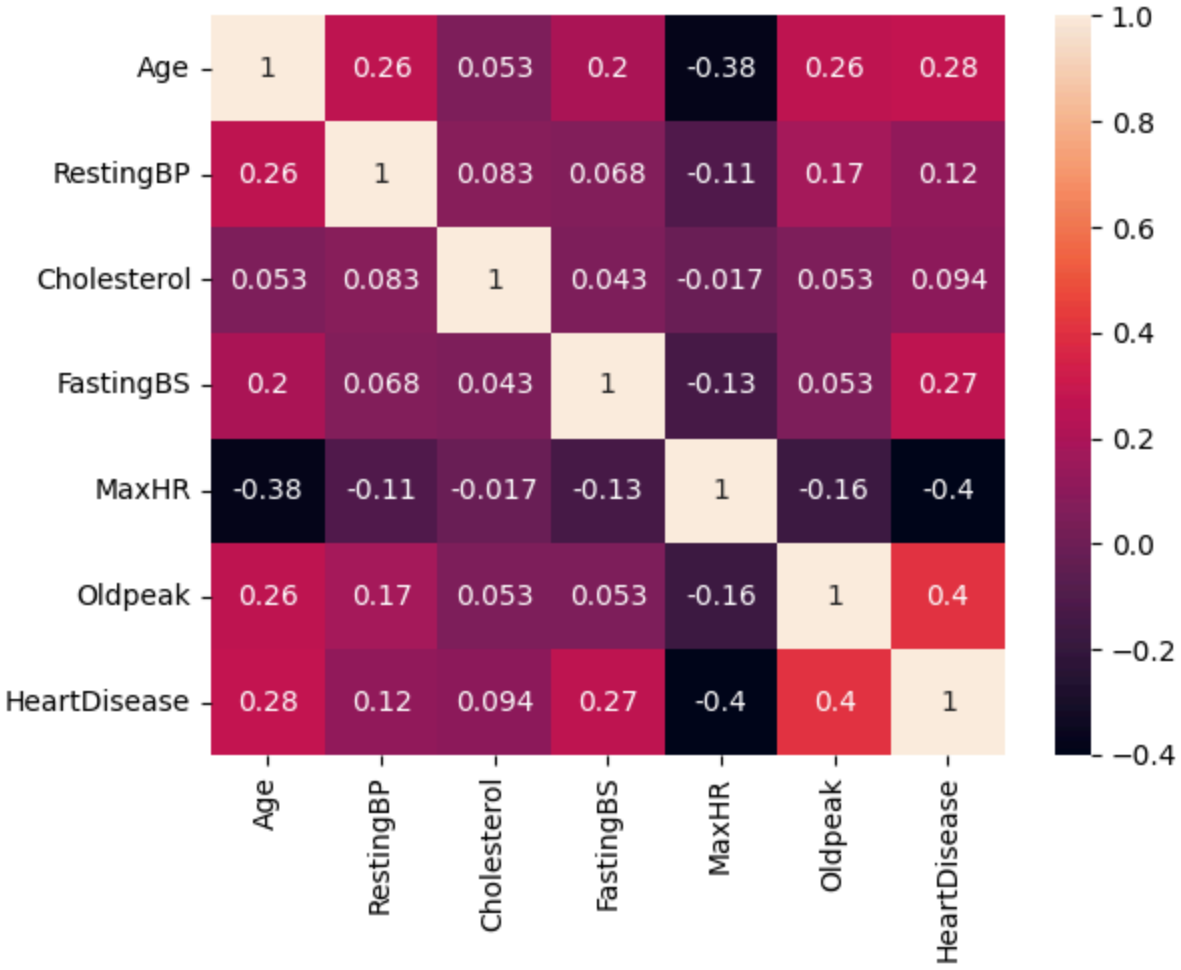

In [24]:
```
sns.violinplot(x = 'HeartDisease', y = 'Age', data = df)
```

Out[24]: <Axes: xlabel='HeartDisease', ylabel='Age'>



In [25]:
```
sns.heatmap(df.corr(numeric_only = True), annot =True)
```

Out[25]: <Axes: >




**Data Preprocessing and cleaning**

In [26]:
```
df_encode = pd.get_dummies(df, drop_first = True)
df_encode
```

Out[26]:

| | Age | RestingBP | Cholesterol | FastingBS | MaxHR | Oldpeak | HeartDisease |
|---|---|---|---|---|---|---|---|
| **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 [27]:
```
df_encode = df_encode.astype(int)
```

In [28]:
```
df_encode
```

Out[28]:

| | Age | RestingBP | Cholesterol | FastingBS | MaxHR | Oldpeak | HeartDisease |
|---|---|---|---|---|---|---|---|
| **0** | 40 | 140 | 289 | 0 | 172 | 0 | 0 |
| **1** | 49 | 160 | 180 | 0 | 156 | 1 | 1 |
| **2** | 37 | 130 | 283 | 0 | 98 | 0 | 0 |
| **3** | 48 | 138 | 214 | 0 | 108 | 1 | 1 |
| **4** | 54 | 150 | 195 | 0 | 122 | 0 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **913** | 45 | 110 | 264 | 0 | 132 | 1 | 1 |
| **914** | 68 | 144 | 193 | 1 | 141 | 3 | 1 |
| **915** | 57 | 130 | 131 | 0 | 115 | 1 | 1 |
| **916** | 57 | 130 | 236 | 0 | 174 | 0 | 1 |
| **917** | 38 | 138 | 175 | 0 | 173 | 0 | 0 |

918 rows × 16 columns

*standard scaling*

In [29]:
```
from sklearn.preprocessing import StandardScaler
numerical_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

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

df_encode.head()
```

Out[29]:

| | Age | RestingBP | Cholesterol | FastingBS | MaxHR | Oldpeak | HeartDis |
|---|---|---|---|---|---|---|---|
| **0** | -1.433140 | 0.414885 | 0.834754 | 0 | 1.382928 | -0.727592 | |
| **1** | -0.478484 | 1.527224 | -1.210675 | 0 | 0.754157 | 0.282891 | |
| **2** | -1.751359 | -0.141284 | 0.722161 | 0 | -1.525138 | -0.727592 | |
| **3** | -0.584556 | 0.303651 | -0.572651 | 0 | -1.132156 | 0.282891 | |
| **4** | 0.051881 | 0.971054 | -0.929194 | 0 | -0.581981 | -0.727592 | |

In [30]:
```
df_encode
```

Out[30]:

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

918 rows × 16 columns

In [31]:
```
df_encode.columns
```

Out[31]:
```
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 [32]:
```
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 [35]:
```
X = df_encode.drop('HeartDisease', axis = 1)
y = df_encode['HeartDisease']
```

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, ra
```

In [45]:
```
scaler = StandardScaler()
X_train_scale = scaler.fit_transform(X_train)
X_test_scale = scaler.fit_transform(X_test)
```

In [50]:
```
models = {
    "Logistic Regression": LogisticRegression(),
    "Naive Bayes": GaussianNB(),
    "Decision Tree": DecisionTreeClassifier(),
    "SVM": SVC(probability = True),
    "KNN": KNeighborsClassifier()
}
```

In [51]:
```
result =[]
```

In [52]:
```
for name, model in models.items():
  model.fit(X_train_scale, y_train)
  y_pred = model.predict(X_test_scale)
  accuracy = accuracy_score(y_test, y_pred)
  f1 = f1_score(y_test, y_pred)
  result.append({
      'model': name,
      'Accuracy': round(accuracy,4),
      'f1': round(f1,4)
      })
```

In [54]:
```
result
```

Out[54]:
```
[{'model': 'Logistic Regression', 'Accuracy': 0.8713, 'f1': 0.887},
 {'model': 'Naive Bayes', 'Accuracy': 0.8581, 'f1': 0.8746},
 {'model': 'Decision Tree', 'Accuracy': 0.7426, 'f1': 0.7651},
 {'model': 'SVM', 'Accuracy': 0.8614, 'f1': 0.88},
 {'model': 'KNN', 'Accuracy': 0.8449, 'f1': 0.863}]
```

In [55]:
```
import joblib
joblib.dump(models['Logistic Regression'], 'LR_Heart.pkl')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(X.columns.tolist(), 'Columns.pkl')
```

Out[55]:
```
['Columns.pkl']
```