In [4]:
```
## Importing Libraries

import numpy as np  # Linear algebra operations
import pandas as pd  # Data processing and analysis
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [46]:
```
## Upload dataset

df = pd.read_csv('/Users/serenaygoler/heart disease.csv')

df.head() # Displays the first 5 rows.
```

Out[46]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR
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 [48]:
```
df.tail() # Display the last 5 rows.
```

Out[48]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxH
913	45	M	TA	110	264	0	Normal	13
914	68	M	ASY	144	193	1	Normal	14
915	57	M	ASY	130	131	0	Normal	1
916	57	F	ATA	130	236	0	LVH	17
917	38	M	NAP	138	175	0	Normal	17

In [50]:
```
df.info() # Prints name and type of variables, number of observations, and c
```

```
<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 [52]:
```
df.shape # Displays the number of rows and columns in the dataset.
```

Out[52]:
```
(918, 12)
```

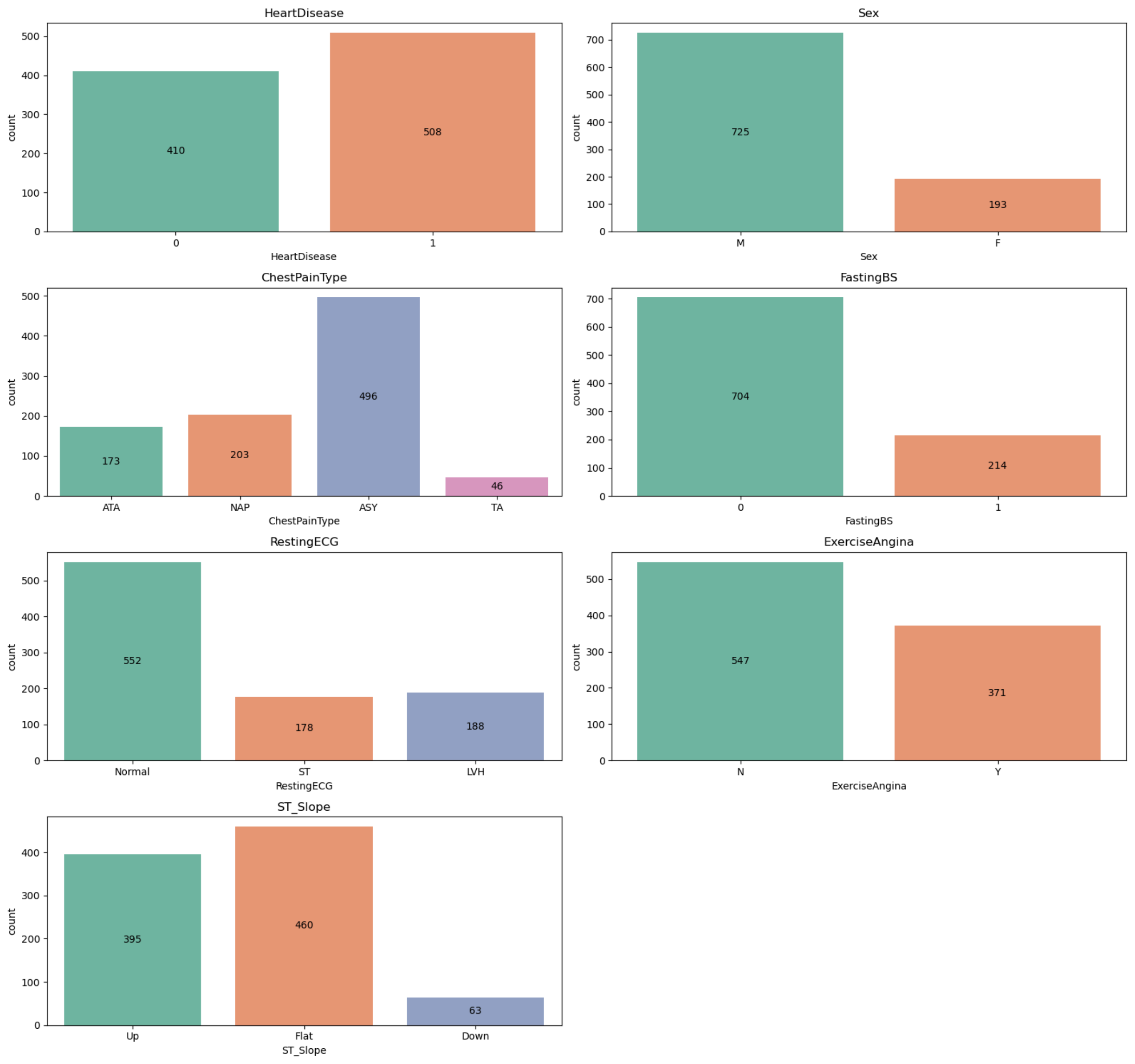
In [54]:
```
df.isna().sum() # Counts missing values in each column.
```

Out[54]:
```
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 [56]:
```
df.duplicated().sum() # Counts the number of duplicate rows.
```

Out[56]:
```
0
```

In [58]:
```
## Each subplot shows the frequency of categories for a specific column.

categorical_cols = ["HeartDisease", "Sex", "ChestPainType", "FastingBS", "Re
                    "ExerciseAngina", "ST_Slope"]

fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(16, 15))
axes = axes.flatten()

for ax, col in zip(axes, categorical_cols):
    sns.countplot(x=df[col], hue=df[col], ax=ax, palette="Set2", legend=Fals
    ax.set_title(col)
    for container in ax.containers:
        ax.bar_label(container, label_type="center")

for ax in axes[len(categorical_cols):]:
    fig.delaxes(ax)

plt.tight_layout()
plt.show()
```



In [60]:
```
## Provides summary statistics for numeric columns, rounded to 2 decimals ar
df.describe().round(2).T
```

Out[60]:

	count	mean	std	min	25%	50%	75%	max
Age	918.0	53.51	9.43	28.0	47.00	54.0	60.0	77.0
RestingBP	918.0	132.40	18.51	0.0	120.00	130.0	140.0	200.0
Cholesterol	918.0	198.80	109.38	0.0	173.25	223.0	267.0	603.0
FastingBS	918.0	0.23	0.42	0.0	0.00	0.0	0.0	1.0
MaxHR	918.0	136.81	25.46	60.0	120.00	138.0	156.0	202.0
Oldpeak	918.0	0.89	1.07	-2.6	0.00	0.6	1.5	6.2
HeartDisease	918.0	0.55	0.50	0.0	0.00	1.0	1.0	1.0

In [62]:

```python
# Count how many Cholesterol values are zero
chol_zero_count = (df["Cholesterol"] == 0).sum()

# Count how many RestingBP values are zero
bp_zero_count = (df["RestingBP"] == 0).sum()

print(f"Number of Cholesterol values equal to 0: {chol_zero_count}")
print(f"Number of RestingBP values equal to 0: {bp_zero_count}")
```

```
Number of Cholesterol values equal to 0: 172
Number of RestingBP values equal to 0: 1
```

In [64]:

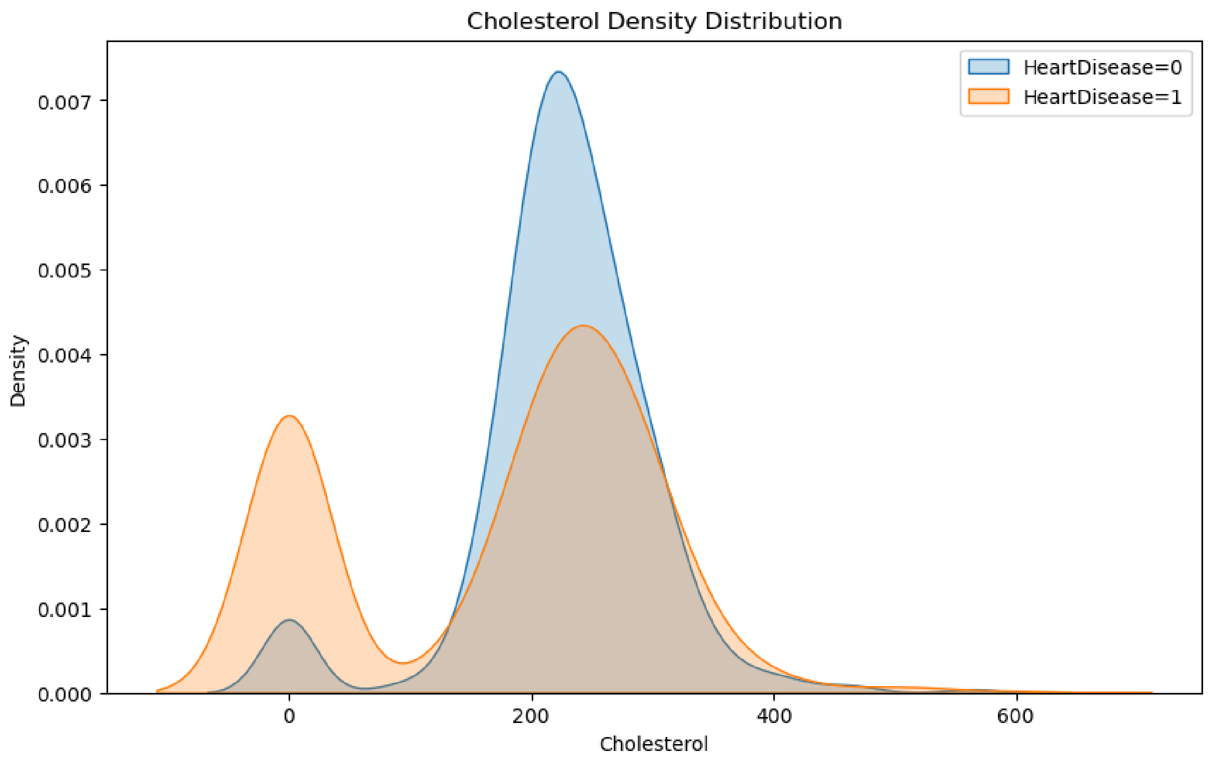
```python
## Filters out rows where Cholesterol equals zero and returns summary statis

print(df[df["Cholesterol"] != 0]["Cholesterol"].describe())
```

```
count    746.000000
mean     244.635389
std       59.153524
min       85.000000
25%      207.250000
50%      237.000000
75%      275.000000
max      603.000000
Name: Cholesterol, dtype: float64
```

In [66]:

```python
# With zeros included
print("=== With Zero values Included ===")
print(df.groupby("HeartDisease")["Cholesterol"].describe())

# Zeros removed
print("\n===  With zero values removed ===")
print(df[df["Cholesterol"] != 0].groupby("HeartDisease")["Cholesterol"].desc
=== With Zero values Included ===
              count        mean         std  min     25%    50%     75%    m
ax
HeartDisease
0             410.0  227.121951   74.634659  0.0  197.25  227.0  266.75   56
4.0
1             508.0  175.940945  126.391398  0.0    0.00  217.0  267.00   60
3.0

===  With zero values removed ===
              count        mean        std    min    25%    50%     75%    m
ax
HeartDisease
0             390.0  238.769231  55.394617   85.0  203.0  231.5  269.00   56
4.0
1             356.0  251.061798  62.462713  100.0  212.0  246.0  283.25   60
3.0
```

In [68]:
```
# Plot the cholesterol distribution for HeartDisease=0 and HeartDisease=1 us
plt.figure(figsize=(10,6))
sns.kdeplot(df[df["HeartDisease"]==0]["Cholesterol"], label="HeartDisease=0"
sns.kdeplot(df[df["HeartDisease"]==1]["Cholesterol"], label="HeartDisease=1"
plt.legend()
plt.title("Cholesterol Density Distribution")
plt.show()
```

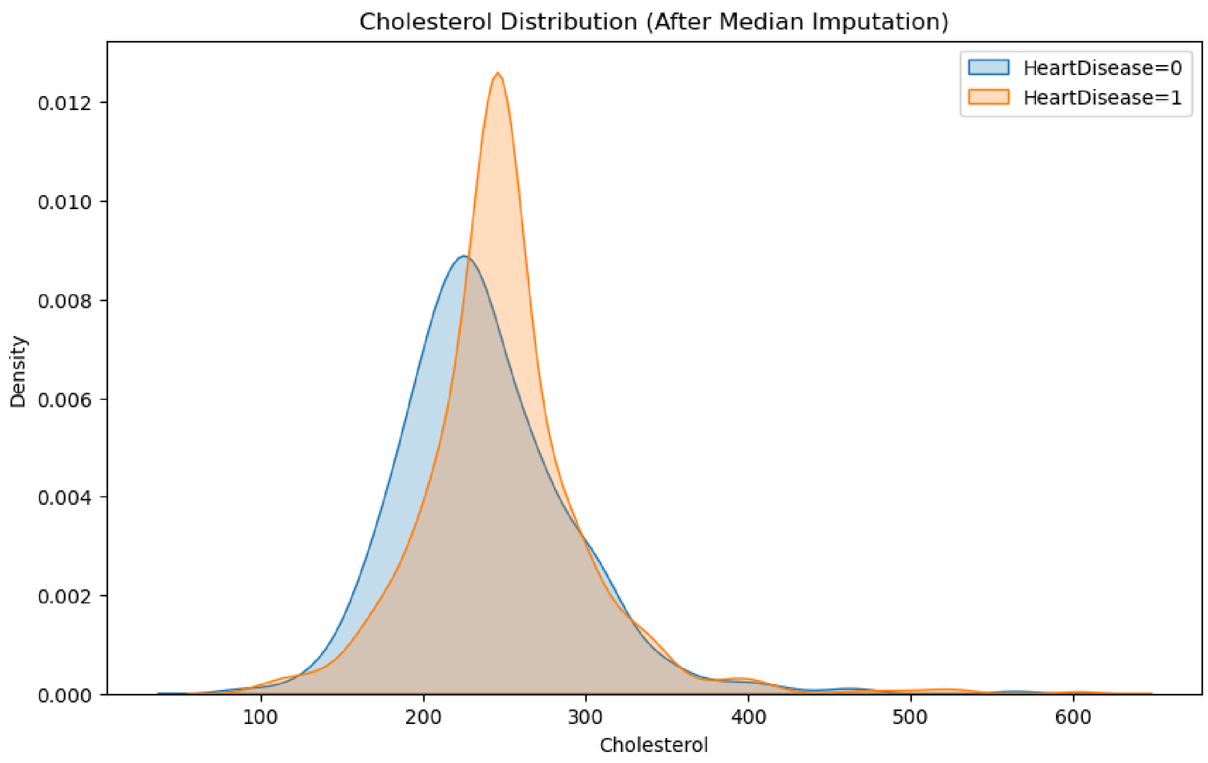


In [70]:
```
# This block cleans the dataset by:
# 1. Removing rows where RestingBP = 0 (unrealistic values).
# 2. Calculating group-wise medians of Cholesterol (by HeartDisease) excludi
# 3. Replacing Cholesterol values of zero with the corresponding group media
# 4. Checking that no zero values remain.
# 5. Displaying summary statistics of Cholesterol by HeartDisease after clea
```

```
df_clean = df.copy()
df_clean = df_clean[df_clean["RestingBP"] != 0].copy()

medians = (
    df_clean[df_clean["Cholesterol"] != 0]
    .groupby("HeartDisease")["Cholesterol"]
    .median()
)

mask_zero = df_clean["Cholesterol"] == 0
df_clean["Cholesterol"] = df_clean["Cholesterol"].astype(float)
df_clean.loc[mask_zero, "Cholesterol"] = (
    df_clean.loc[mask_zero, "HeartDisease"].map(medians)
)

print("Remaining zeros:", (df_clean["Cholesterol"] == 0).sum())
print(df_clean.groupby("HeartDisease")["Cholesterol"].describe())
```

```
Remaining zeros: 0
              count        mean        std    min    25%    50%     75%    m
ax
HeartDisease
0             410.0  238.414634  54.045994   85.0  204.0  231.5  266.75   56
4.0
1             507.0  249.554241  52.370323  100.0  225.0  246.0  267.00   60
3.0
```

In [72]:
```
# KDE plot — distribution comparison after median imputation
plt.figure(figsize=(10,6))
sns.kdeplot(df_clean[df_clean["HeartDisease"]==0]["Cholesterol"], label="Hea
sns.kdeplot(df_clean[df_clean["HeartDisease"]==1]["Cholesterol"], label="Hea
plt.title("Cholesterol Distribution (After Median Imputation)")
plt.xlabel("Cholesterol")
plt.ylabel("Density")
plt.legend()
plt.show()
```



In [74]:
```
## Provides summary statistics for numeric columns for clean data, rounded t
df_clean.describe().round(2).T
```

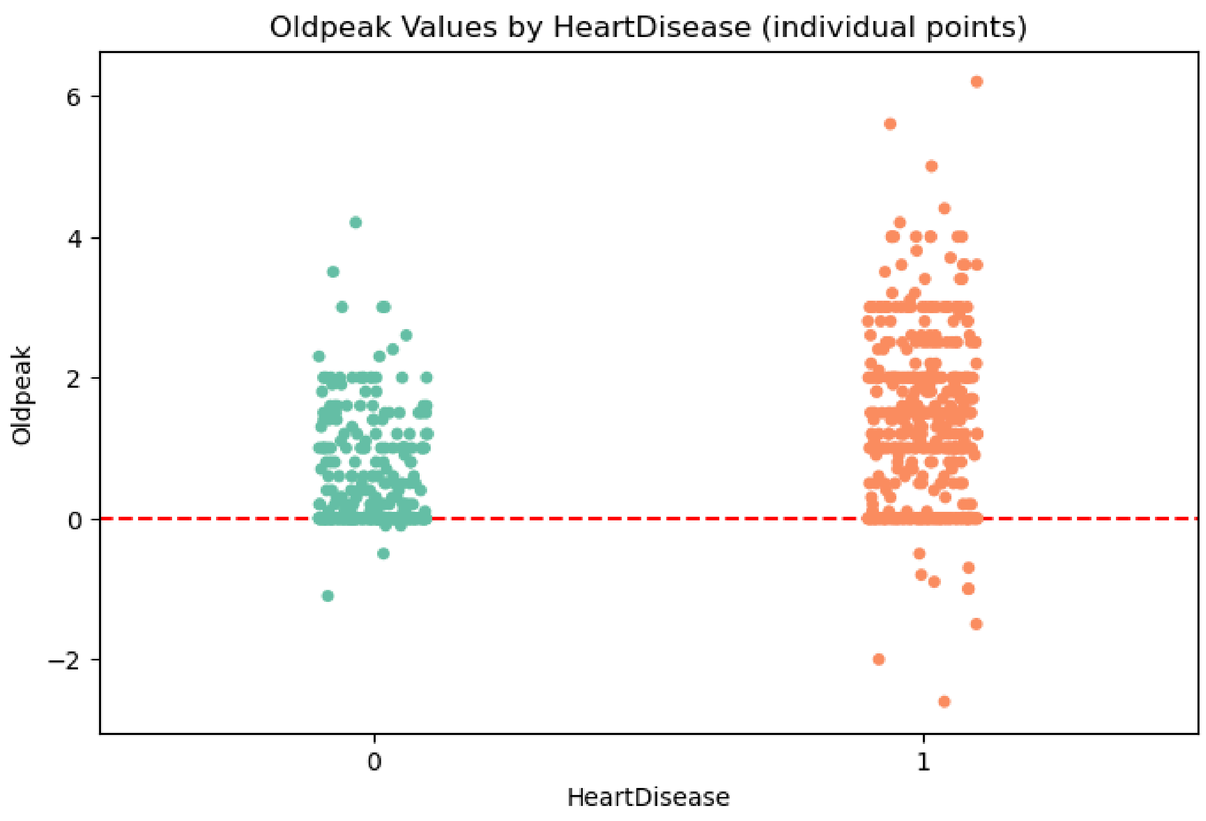
Out[74]:

	count	mean	std	min	25%	50%	75%	max
Age	917.0	53.51	9.44	28.0	47.0	54.0	60.0	77.0
RestingBP	917.0	132.54	18.00	80.0	120.0	130.0	140.0	200.0
Cholesterol	917.0	244.57	53.39	85.0	214.0	246.0	267.0	603.0
FastingBS	917.0	0.23	0.42	0.0	0.0	0.0	0.0	1.0
MaxHR	917.0	136.79	25.47	60.0	120.0	138.0	156.0	202.0
Oldpeak	917.0	0.89	1.07	-2.6	0.0	0.6	1.5	6.2
HeartDisease	917.0	0.55	0.50	0.0	0.0	1.0	1.0	1.0

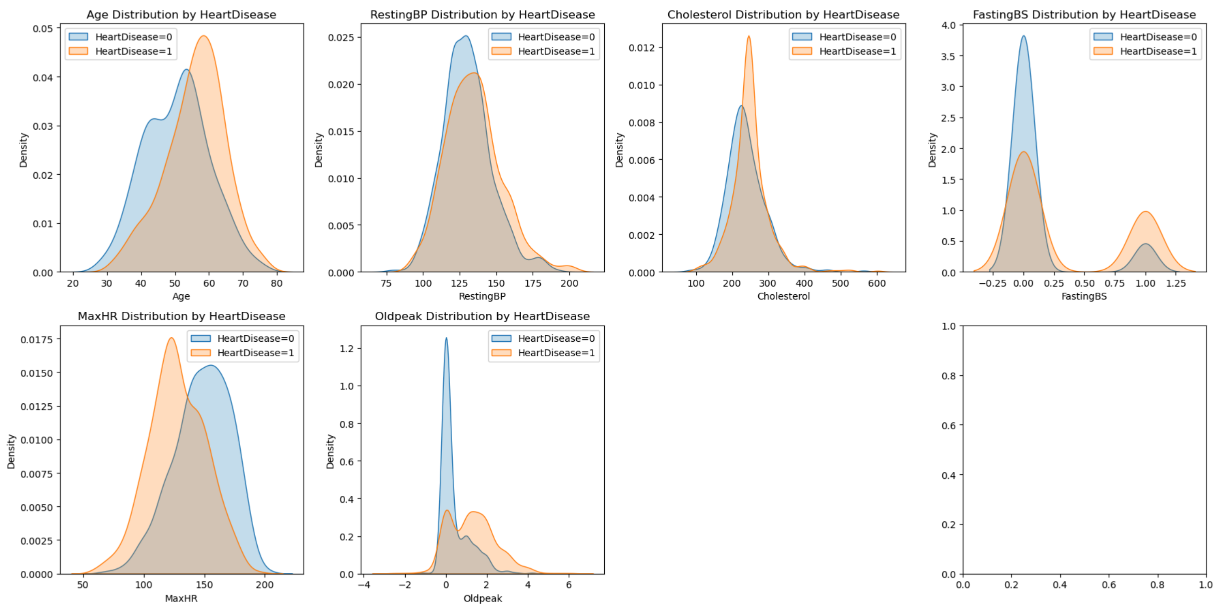
In [76]:
```
# Count negative Oldpeak values

neg_oldpeak_count = (df["Oldpeak"] < 0).sum()
print(f"Number of negative Oldpeak values: {neg_oldpeak_count}")
```

```
Number of negative Oldpeak values: 13
```

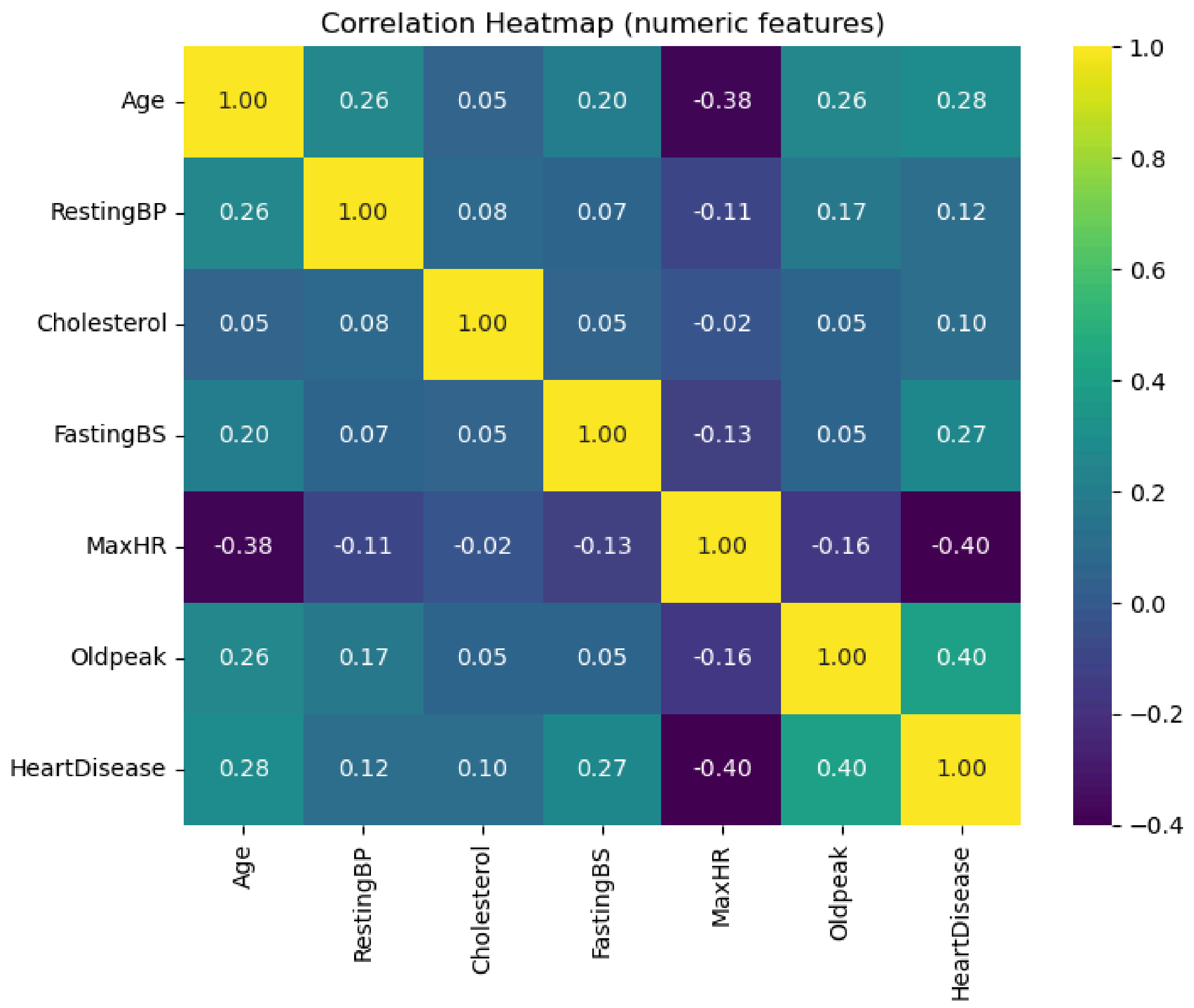
In [78]:
```
# Stripplot showing distribution of Oldpeak values by HeartDisease, with ref

plt.figure(figsize=(8,5))
sns.stripplot(x="HeartDisease", y="Oldpeak", data=df, jitter=True, palette="
plt.axhline(0, color="red", linestyle="--")
plt.title("Oldpeak Values by HeartDisease (individual points)")
plt.show()
```



In [82]:
```
# Select only numeric columns (excluding the target if needed)
num_cols = df_clean.select_dtypes(include="number").columns.drop("HeartDisea

# Distributions of numerical features by HeartDisease status
fig, axes = plt.subplots(2, 4, figsize=(18, 9))
axes = axes.flatten()
for i, col in enumerate(num_cols):
    ax = axes[i]
    sns.kdeplot(df_clean[df_clean["HeartDisease"]==0][col].dropna(), label="
    sns.kdeplot(df_clean[df_clean["HeartDisease"]==1][col].dropna(), label="
    ax.set_title(f"{col} Distribution by HeartDisease"); ax.legend()

# Remove empty subplot (if number of cols < 8)
fig.delaxes(axes[len(num_cols)])
plt.tight_layout()
plt.show()
```

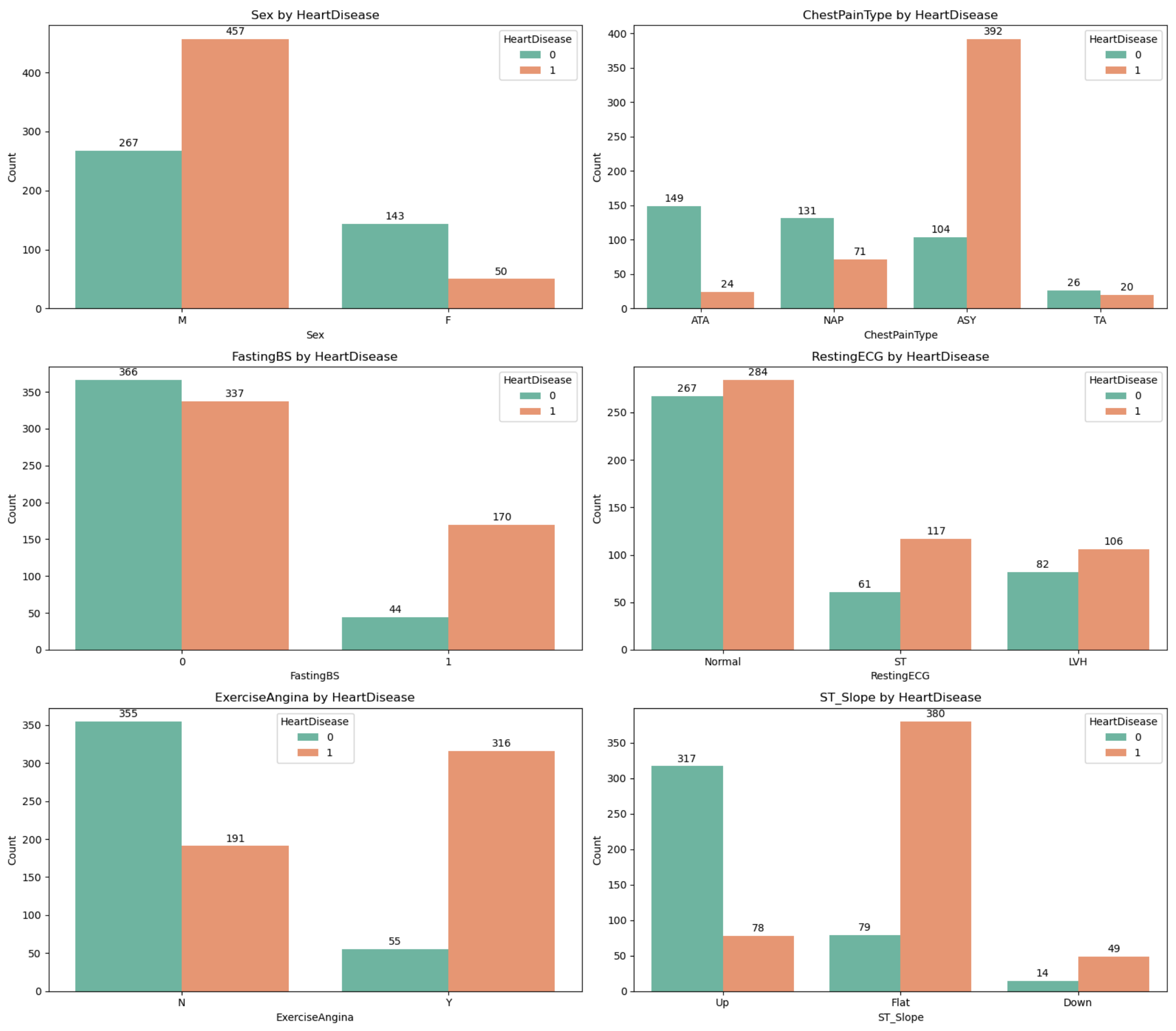


```
In [83]: # Select only numerical columns and to check correlation
         num_cols = df_clean.select_dtypes(include=[np.number]).columns

         plt.figure(figsize=(8,6))
         sns.heatmap(df_clean[num_cols].corr(), annot=True, cmap="viridis", fmt=".2f"
         plt.title("Correlation Heatmap (numeric features)")
         plt.show()
```



	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
Age	1.00	0.26	0.05	0.20	-0.38	0.26	0.28
RestingBP	0.26	1.00	0.08	0.07	-0.11	0.17	0.12
Cholesterol	0.05	0.08	1.00	0.05	-0.02	0.05	0.10
FastingBS	0.20	0.07	0.05	1.00	-0.13	0.05	0.27
MaxHR	-0.38	-0.11	-0.02	-0.13	1.00	-0.16	-0.40
Oldpeak	0.26	0.17	0.05	0.05	-0.16	1.00	0.40
HeartDisease	0.28	0.12	0.10	0.27	-0.40	0.40	1.00

In [84]:
```python
# Distribution of categorical variables by the target variable
cat_cols = ["Sex", "ChestPainType", "FastingBS", "RestingECG", "ExerciseAngi
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(16, 14))
axes = axes.flatten()

for ax, col in zip(axes, cat_cols):
    g = sns.countplot(data=df_clean, x=col, hue="HeartDisease", palette="Set
    ax.set_title(f"{col} by HeartDisease")
    ax.set_xlabel(col); ax.set_ylabel("Count")
    # label name
    for c in g.containers:
        g.bar_label(c, padding=2, fmt="%.0f")

# Remove extra axes
for ax in axes[len(cat_cols):]:
    fig.delaxes(ax)

plt.tight_layout()
plt.show()
```

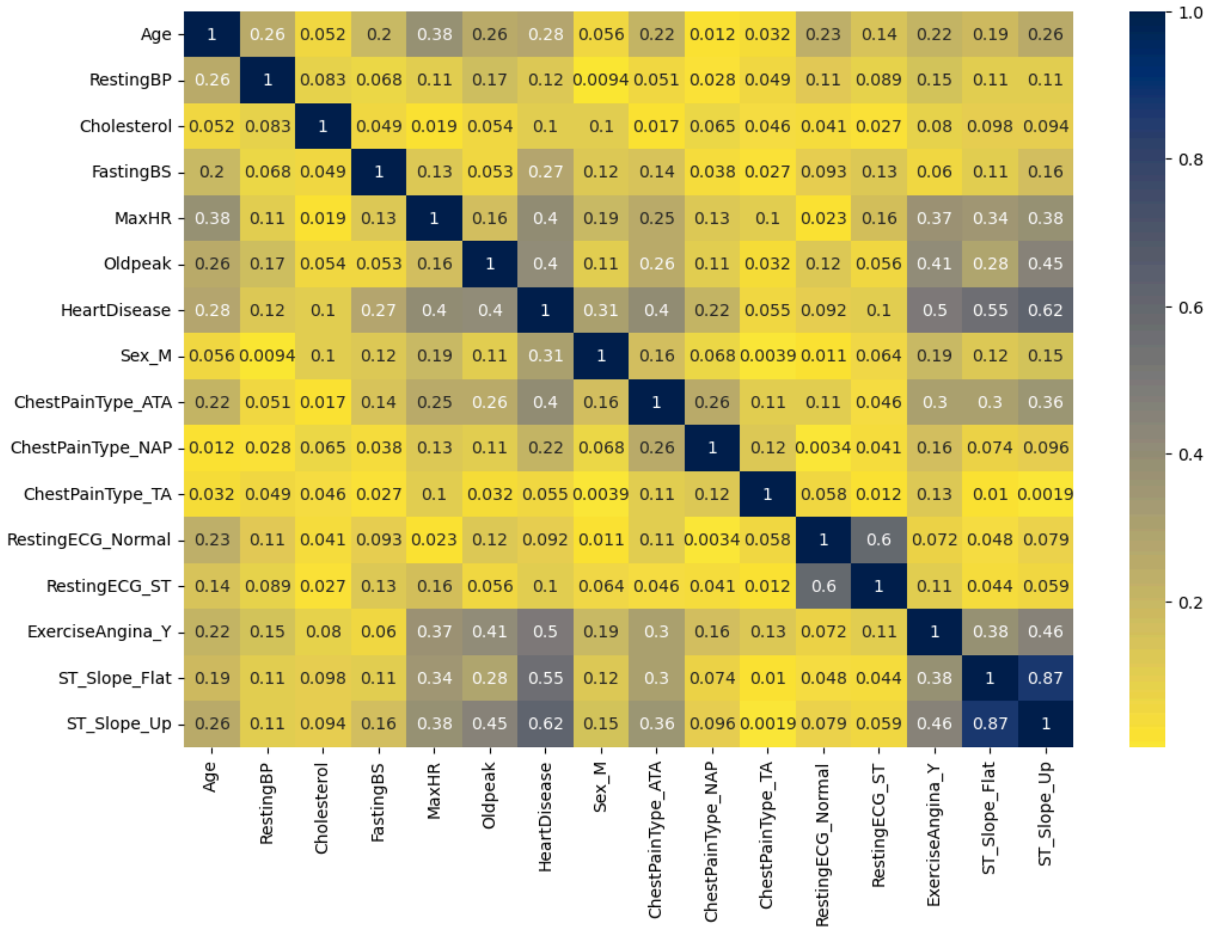


In [88]:
```
# One-hot encoding was applied to transform categorical variables into dummy
DUMMY = pd.get_dummies(df_clean, drop_first=True)
DUMMY.head()
```

Out[88]:

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	Sex_M
0	40	140	289.0	0	172	0.0	0	True
1	49	160	180.0	0	156	1.0	1	False
2	37	130	283.0	0	98	0.0	0	True
3	48	138	214.0	0	108	1.5	1	False
4	54	150	195.0	0	122	0.0	0	True

In [90]:
```
## Compute absolute pairwise correlations (after one-hot encoding) and visua
correlations = abs(DUMMY.corr())
plt.figure(figsize=(12,8))
sns.heatmap(correlations, annot=True, cmap="cividis_r")
plt.show()
```


Age 1 0.26 0.052 0.2 0.38 0.26 0.28 0.056 0.22 0.012 0.032 0.23 0.14 0.22 0.19 0.26
RestingBP 0.26 1 0.083 0.068 0.11 0.17 0.12 0.0094 0.051 0.028 0.049 0.11 0.089 0.15 0.11 0.11
Cholesterol 0.052 0.083 1 0.049 0.019 0.054 0.1 0.1 0.017 0.065 0.046 0.041 0.027 0.08 0.098 0.094
FastingBS 0.2 0.068 0.049 1 0.13 0.053 0.27 0.12 0.14 0.038 0.027 0.093 0.13 0.06 0.11 0.16
MaxHR 0.38 0.11 0.019 0.13 1 0.16 0.4 0.19 0.25 0.13 0.1 0.023 0.16 0.37 0.34 0.38
Oldpeak 0.26 0.17 0.054 0.053 0.16 1 0.4 0.11 0.26 0.11 0.032 0.12 0.056 0.41 0.28 0.45
HeartDisease 0.28 0.12 0.1 0.27 0.4 0.4 1 0.31 0.4 0.22 0.055 0.092 0.1 0.5 0.55 0.62
Sex_M 0.056 0.0094 0.1 0.12 0.19 0.11 0.31 1 0.16 0.068 0.0039 0.011 0.064 0.19 0.12 0.15
ChestPainType_ATA 0.22 0.051 0.017 0.14 0.25 0.26 0.4 0.16 1 0.26 0.11 0.11 0.046 0.3 0.3 0.36
ChestPainType_NAP 0.012 0.028 0.065 0.038 0.13 0.11 0.22 0.068 0.26 1 0.12 0.0034 0.041 0.16 0.074 0.096
ChestPainType_TA 0.032 0.049 0.046 0.027 0.1 0.032 0.055 0.0039 0.11 0.12 1 0.058 0.012 0.13 0.01 0.0019
RestingECG_Normal 0.23 0.11 0.041 0.093 0.023 0.12 0.092 0.011 0.11 0.0034 0.058 1 0.6 0.072 0.048 0.079
RestingECG_ST 0.14 0.089 0.027 0.13 0.16 0.056 0.1 0.064 0.046 0.041 0.012 0.6 1 0.11 0.044 0.059
ExerciseAngina_Y 0.22 0.15 0.08 0.06 0.37 0.41 0.5 0.19 0.3 0.16 0.13 0.072 0.11 1 0.38 0.46
ST_Slope_Flat 0.19 0.11 0.098 0.11 0.34 0.28 0.55 0.12 0.3 0.074 0.01 0.048 0.044 0.38 1 0.87
ST_Slope_Up 0.26 0.11 0.094 0.16 0.38 0.45 0.62 0.15 0.36 0.096 0.0019 0.079 0.059 0.46 0.87 1
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
1.0
0.8
0.6
0.4
0.2

```
In [ ]:
```

```
In [ ]:
```