

daa set link <https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction>

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/heart.csv')
```

```
df.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

```
print("Dataset Info:")
df.info()
```

```
print("\nSummary Statistics:")
df.describe()
```

```
Dataset 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

Summary Statistics:
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease
count	918.000000	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	0.553377
std	9.432617	18.514154	109.384145	0.423046	25.460334	1.066570	0.497414
min	28.000000	0.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	0.000000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	0.000000	138.000000	0.600000	1.000000
75%	60.000000	140.000000	267.000000	0.000000	156.000000	1.500000	1.000000
max	77.000000	200.000000	603.000000	1.000000	202.000000	6.200000	1.000000

```
print("Missing Values per Column:")
print(df.isnull().sum())
```

```
num_cols = df.select_dtypes(include=['float64', 'int64']).columns
cat_cols = df.select_dtypes(exclude=['float64', 'int64']).columns
```

```
for col in num_cols:
    df[col].fillna(df[col].median(), inplace=True)
```

```
for col in cat_cols:
    df[col].fillna(df[col].mode()[0], inplace=True)
```

```
print("\nMissing Values After Cleaning:")
print(df.isnull().sum())
```

```
Missing Values per Column:
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

Missing Values After Cleaning:
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
/tmp/ipython-input-1003229514.py:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method(col: value, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df[col].fillna(df[col].median(), inplace=True)
/tmp/ipython-input-1003229514.py:11: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
```

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method(col: value, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

```
df[col].fillna(df[col].mode()[0], inplace=True)
```

```
def handle_outliers(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df[col] = np.where(df[col] < lower, lower, np.where(df[col] > upper, upper, df[col]))
```

```
for col in num_cols:
    handle_outliers(df, col)

print("Outliers handled successfully.")
```

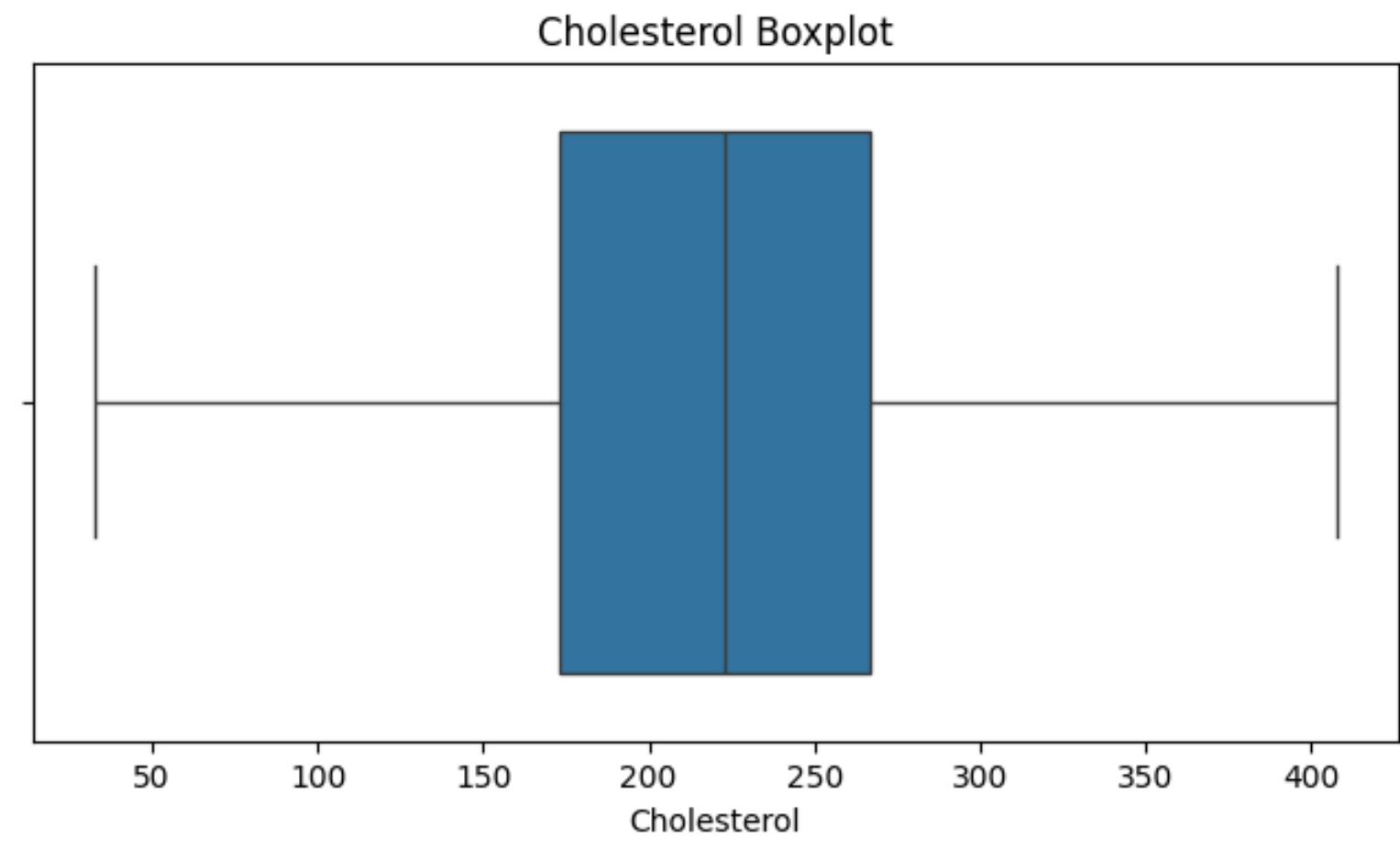
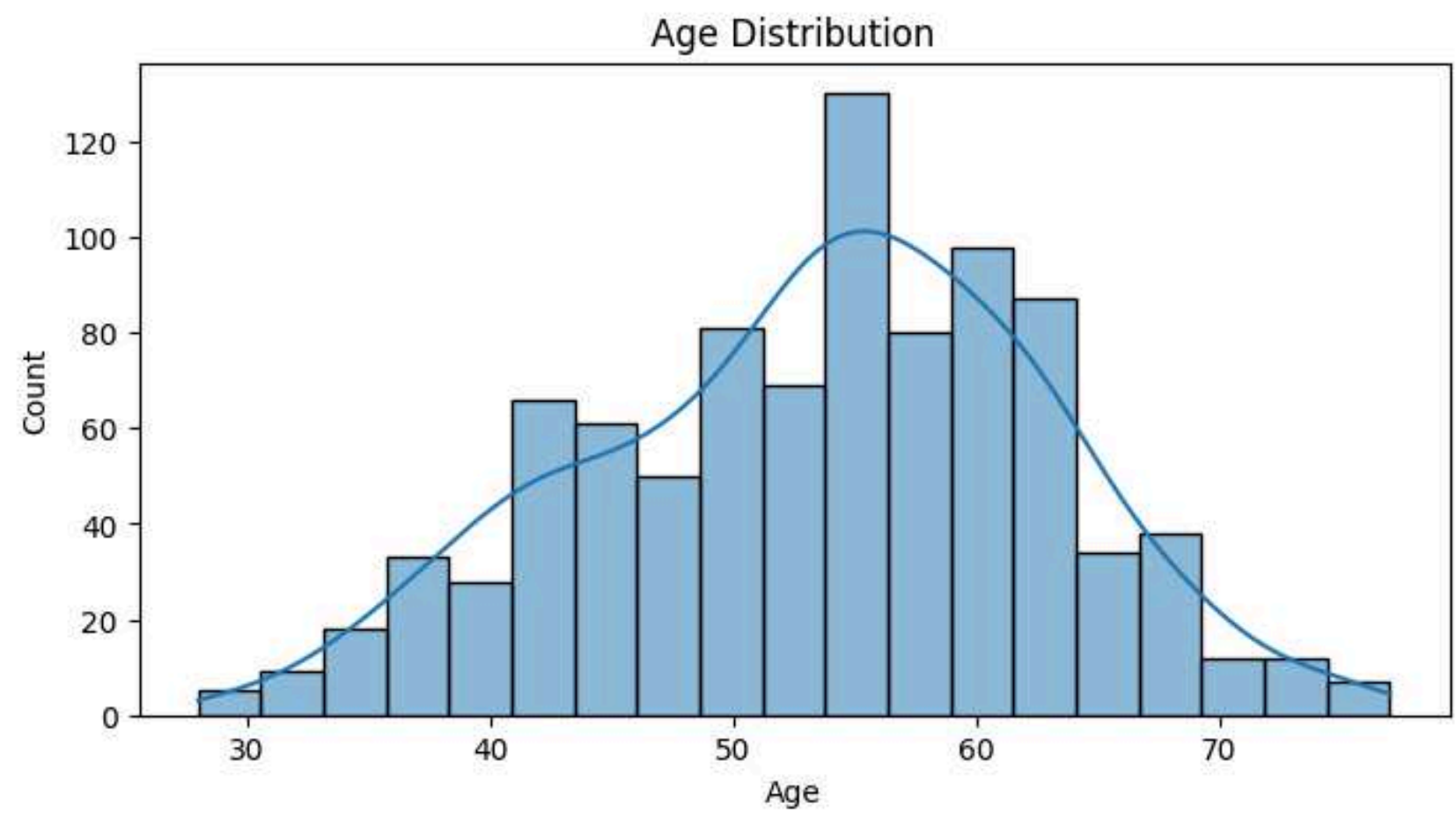
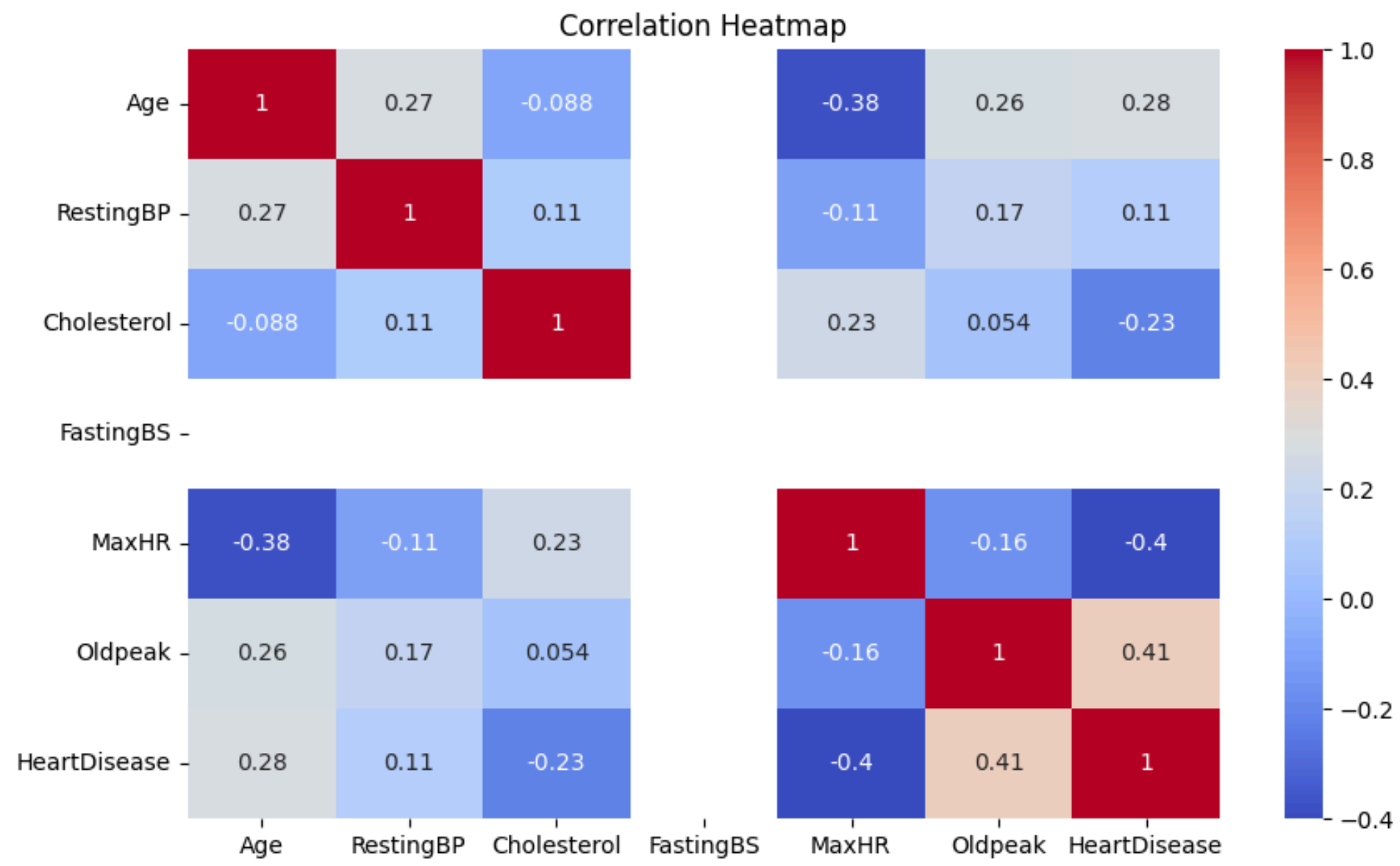
Outliers handled successfully.

```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()

plt.figure(figsize=(8,4))
sns.histplot(df['Age'], kde=True)
plt.title('Age Distribution')
plt.show()

plt.figure(figsize=(8,4))
sns.boxplot(x=df['Cholesterol'])
plt.title('Cholesterol Boxplot')
plt.show()

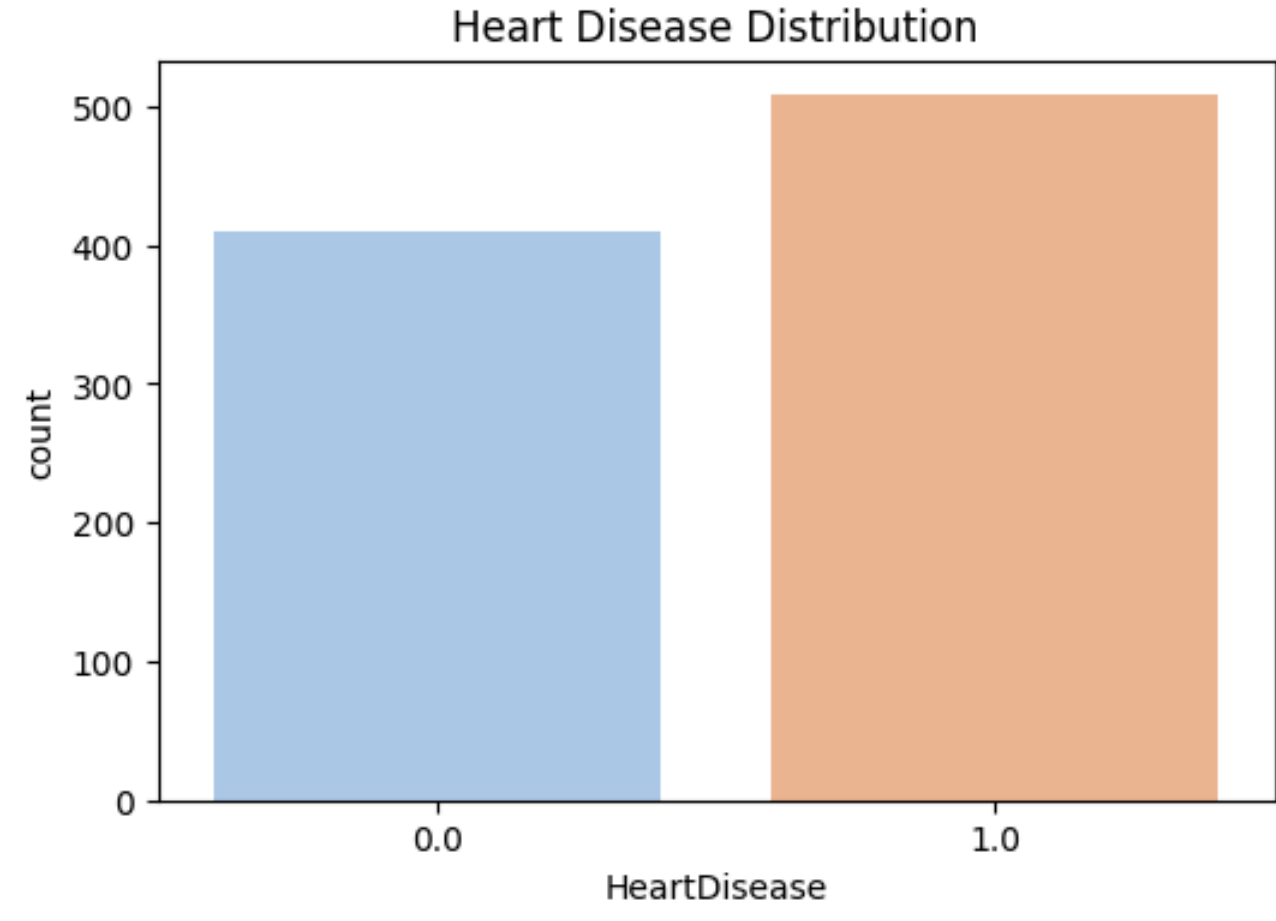
plt.figure(figsize=(6,4))
sns.countplot(x='HeartDisease', data=df, palette='pastel')
plt.title('Heart Disease Distribution')
plt.show()
```



/tmp/ipython-input-123469692.py:21: FutureWarning:

Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.

```
sns.countplot(x='HeartDisease', data=df, palette='pastel')
```



```
df.to_csv('/content/cleaned_heart_dataset.csv', index=False)
print("Cleaned dataset saved successfully.")
```

Cleaned dataset saved successfully.

