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

```
df = pd.read_csv('heart_disease_dataset.csv')
```

```
df
```

	Age	Gender	Cholesterol	Blood Pressure	Heart Rate	Smoking	Alcohol Intake	Exercise Hours	Family History	Diabetes	Obesity	Stress Level	Blood Sugar	Exercise Induced Angina	Cl
0	75	Female	228	119	66	Current	Heavy	1	No	No	Yes	8	119	Yes	
1	48	Male	204	165	62	Current	NaN	5	No	No	No	9	70	Yes	
2	53	Male	234	91	67	Never	Heavy	3	Yes	No	Yes	5	196	Yes	
3	69	Female	192	90	72	Current	NaN	4	No	Yes	No	7	107	Yes	N
4	62	Female	172	163	93	Never	NaN	6	No	Yes	No	2	183	Yes	Asy
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	56	Female	269	111	86	Never	Heavy	5	No	Yes	Yes	10	120	No	N
996	78	Female	334	145	76	Never	NaN	6	No	No	No	10	196	Yes	
997	79	Male	151	179	81	Never	Moderate	4	Yes	No	Yes	8	189	Yes	Asy
998	60	Female	326	151	68	Former	NaN	8	Yes	Yes	No	5	174	Yes	
999	53	Male	226	116	82	Current	NaN	6	No	No	Yes	5	161	Yes	Asy

1000 rows × 16 columns

Next steps: Generate code with df New interactive sheet

```
df.head(3)
```

	Age	Gender	Cholesterol	Blood Pressure	Heart Rate	Smoking	Alcohol Intake	Exercise Hours	Family History	Diabetes	Obesity	Stress Level	Blood Sugar	Exercise Induced Angina	Ches Pai Typ
0	75	Female	228	119	66	Current	Heavy	1	No	No	Yes	8	119	Yes	Atypica Angin
1	48	Male	204	165	62	Current	NaN	5	No	No	No	9	70	Yes	Typica Angin
2	53	Male	234	91	67	Never	Heavy	3	Yes	No	Yes	5	196	Yes	Atypica Angin

Next steps: Generate code with df New interactive sheet

```
df.shape
```

```
(1000, 16)
```

```
df.size
```

```
16000
```

```
df.describe()
```

	Age	Cholesterol	Blood Pressure	Heart Rate	Exercise Hours	Stress Level	Blood Sugar	Heart Disease
count	1000.000000	1000.000000	1000.0000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	52.293000	249.939000	135.2810	79.204000	4.529000	5.646000	134.941000	0.392000
std	15.727126	57.914673	26.3883	11.486092	2.934241	2.831024	36.699624	0.488441
min	25.000000	150.000000	90.0000	60.000000	0.000000	1.000000	70.000000	0.000000
25%	39.000000	200.000000	112.7500	70.000000	2.000000	3.000000	104.000000	0.000000
50%	52.000000	248.000000	136.0000	79.000000	4.500000	6.000000	135.000000	0.000000
75%	66.000000	299.000000	159.0000	89.000000	7.000000	8.000000	167.000000	1.000000
max	79.000000	349.000000	179.0000	99.000000	9.000000	10.000000	199.000000	1.000000

```
df.dtypes
```

	0
Age	int64
Gender	object
Cholesterol	int64
Blood Pressure	int64
Heart Rate	int64
Smoking	object
Alcohol Intake	object
Exercise Hours	int64
Family History	object
Diabetes	object
Obesity	object
Stress Level	int64
Blood Sugar	int64
Exercise Induced Angina	object
Chest Pain Type	object
Heart Disease	int64

dtype: object

```
df.columns
```

```
Index(['Age', 'Gender', 'Cholesterol', 'Blood Pressure', 'Heart Rate',
       'Smoking', 'Alcohol Intake', 'Exercise Hours', 'Family History',
       'Diabetes', 'Obesity', 'Stress Level', 'Blood Sugar',
       'Exercise Induced Angina', 'Chest Pain Type', 'Heart Disease'],
      dtype='object')
```

```
df.index
```

```
RangeIndex(start=0, stop=1000, step=1)
```

```
df.isnull().sum()
```

	0
Age	0
Gender	0
Cholesterol	0
Blood Pressure	0
Heart Rate	0
Smoking	0
Alcohol Intake	340
Exercise Hours	0
Family History	0
Diabetes	0
Obesity	0
Stress Level	0
Blood Sugar	0
Exercise Induced Angina	0
Chest Pain Type	0
Heart Disease	0

dtype: int64

```
df.drop('Alcohol Intake',axis=1)
df
```

	Age	Gender	Cholesterol	Blood Pressure	Heart Rate	Smoking	Alcohol Intake	Exercise Hours	Family History	Diabetes	Obesity	Stress Level	Blood Sugar	Exercise Induced Angina	Cl
0	75	Female	228	119	66	Current	Heavy	1	No	No	Yes	8	119	Yes	
1	48	Male	204	165	62	Current	NaN	5	No	No	No	9	70	Yes	
2	53	Male	234	91	67	Never	Heavy	3	Yes	No	Yes	5	196	Yes	
3	69	Female	192	90	72	Current	NaN	4	No	Yes	No	7	107	Yes	N
4	62	Female	172	163	93	Never	NaN	6	No	Yes	No	2	183	Yes	Asy
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	56	Female	269	111	86	Never	Heavy	5	No	Yes	Yes	10	120	No	N
996	78	Female	334	145	76	Never	NaN	6	No	No	No	10	196	Yes	
997	79	Male	151	179	81	Never	Moderate	4	Yes	No	Yes	8	189	Yes	Asy
998	60	Female	326	151	68	Former	NaN	8	Yes	Yes	No	5	174	Yes	
999	53	Male	226	116	82	Current	NaN	6	No	No	Yes	5	161	Yes	Asy

1000 rows × 16 columns

Next steps: Generate code with df New interactive sheet

```
df.head(2)
```

	Age	Gender	Cholesterol	Blood Pressure	Heart Rate	Smoking	Alcohol Intake	Exercise Hours	Family History	Diabetes	Obesity	Stress Level	Blood Sugar	Exercise Induced Angina	Ches Pai Typ
0	75	Female	228	119	66	Current	Heavy	1	No	No	Yes	8	119	Yes	Atypica Angin
1	48	Male	204	165	62	Current	NaN	5	No	No	No	9	70	Yes	Typica Angin

Next steps: Generate code with df New interactive sheet

```
df.nunique()
```

	0
Age	55
Gender	2
Cholesterol	200
Blood Pressure	90
Heart Rate	40
Smoking	3
Alcohol Intake	2
Exercise Hours	10
Family History	2
Diabetes	2
Obesity	2
Stress Level	10
Blood Sugar	130
Exercise Induced Angina	2
Chest Pain Type	4
Heart Disease	2

dtype: int64

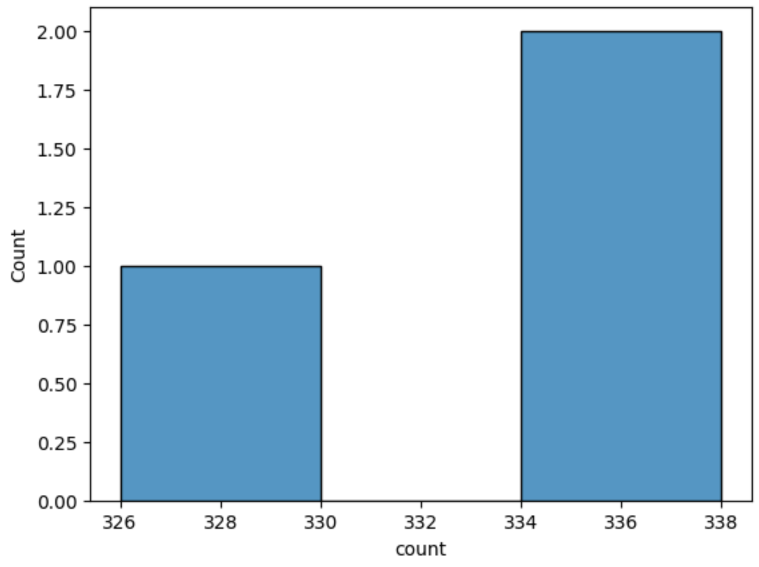
```
a=df['Smoking'].value_counts()
a
```

Smoking	count
Never	338
Current	336
Former	326

dtype: int64

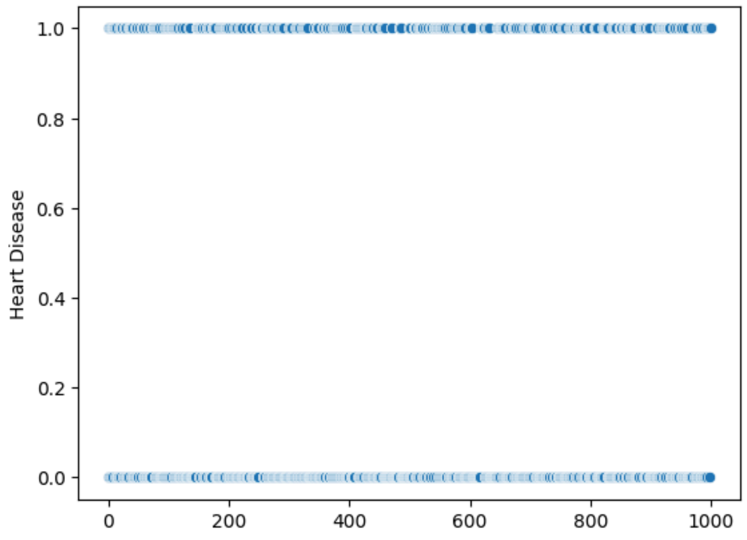
```
sns.histplot(a)
```

```
<Axes: xlabel='count', ylabel='Count'>
```



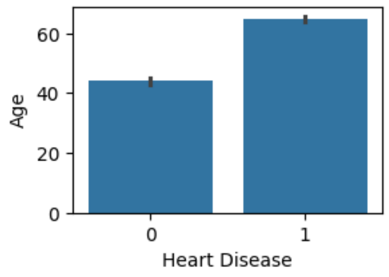
```
sns.scatterplot(df['Heart Disease'])
```

```
<Axes: ylabel='Heart Disease'>
```



```
plt.figure(figsize=(3, 2))
sns.barplot(x='Heart Disease',y='Age',data=df)
```

```
<Axes: xlabel='Heart Disease', ylabel='Age'>
```



encoding

```
df.columns
```

```
Index(['Age', 'Gender', 'Cholesterol', 'Blood Pressure', 'Heart Rate',
       'Smoking', 'Alcohol Intake', 'Exercise Hours', 'Family History',
       'Diabetes', 'Obesity', 'Stress Level', 'Blood Sugar',
       'Exercise Induced Angina', 'Chest Pain Type', 'Heart Disease'],
      dtype='object')
```

```
df.head(1)
```

	Age	Gender	Cholesterol	Blood Pressure	Heart Rate	Smoking	Alcohol Intake	Exercise Hours	Family History	Diabetes	Obesity	Stress Level	Blood Sugar	Exercise Induced Angina	Ches Pai Typ
0	75	Female	228	119	66	Current	Heavy	1	No	No	Yes	8	119	Yes	Atypica Angin

Next steps: Generate code with df New interactive sheet

```
from sklearn.preprocessing import LabelEncoder
Le=LabelEncoder
```

```
a=('Gender','Smoking','Family History','Diabetes','Obesity','Exercise Induced Angina','Chest Pain Type')
for i in a:
    df[i]=Le().fit_transform(df[i])
```

```
df
```

	Age	Gender	Cholesterol	Blood Pressure	Heart Rate	Smoking	Alcohol Intake	Exercise Hours	Family History	Diabetes	Obesity	Stress Level	Blood Sugar	Exercise Induced Angina	Che Pa Ty
0	75	0	228	119	66	0	Heavy	1	0	0	1	8	119	1	
1	48	1	204	165	62	0	NaN	5	0	0	0	9	70	1	
2	53	1	234	91	67	2	Heavy	3	1	0	1	5	196	1	
3	69	0	192	90	72	0	NaN	4	0	1	0	7	107	1	
4	62	0	172	163	93	2	NaN	6	0	1	0	2	183	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	56	0	269	111	86	2	Heavy	5	0	1	1	10	120	0	
996	78	0	334	145	76	2	NaN	6	0	0	0	10	196	1	
997	79	1	151	179	81	2	Moderate	4	1	0	1	8	189	1	
998	60	0	326	151	68	1	NaN	8	1	1	0	5	174	1	
999	53	1	226	116	82	0	NaN	6	0	0	1	5	161	1	

1000 rows × 16 columns

Next steps: Generate code with df New interactive sheet

```
x=df.drop(['Heart Disease', 'Alcohol Intake'],axis=1)
y=df['Heart Disease']
```

```
x
```

	Age	Gender	Cholesterol	Blood Pressure	Heart Rate	Smoking	Exercise Hours	Family History	Diabetes	Obesity	Stress Level	Blood Sugar	Exercise Induced Angina	Chest Pain Type
0	75	0	228	119	66	0	1	0	0	1	8	119	1	1
1	48	1	204	165	62	0	5	0	0	0	9	70	1	3
2	53	1	234	91	67	2	3	1	0	1	5	196	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	56	0	269	111	86	2	5	0	1	1	10	120	0	2
996	78	0	334	145	76	2	6	0	0	0	10	196	1	3
998	60	0	326	151	68	1	8	1	1	0	5	174	1	1
999	53	1	226	116	82	0	6	0	0	1	5	161	1	0

1000 rows × 14 columns

```
Start coding or generate with AI.
```

train test split

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

scaling

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

implementation of model

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```