Double-click (or enter) to edit

Objective: To load the heart disease dataset and understand its basic structure. Tasks:

1. Import Pandas and NumPy libraries.
2. Load the dataset using pandas read_csv().
3. Display first five and last five rows.
4. Find number of rows and columns.
5. Display column name

```
import pandas as pd
import numpy as np
```

```
!unzip -o /content/heart_disease.csv.zip -d /content/

df = pd.read_csv('/content/Heart_Disease_Prediction.csv')
display(df.head())
```

```
unzip:  cannot find or open /content/heart_disease.csv.zip, /content/heart_disea
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	S c
0	70	1	4	130	322	0	2	109	0	2.4	
1	67	0	3	115	564	0	2	160	0	1.6	
2	57	1	2	124	261	0	0	141	0	0.3	
3	64	1	4	128	263	0	0	105	1	0.2	
4	74	0	2	120	269	0	2	121	1	0.2	

```
display(df.tail())
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression
265	52	1	3	172	199	1	0	162	0	0.5
266	44	1	2	120	263	0	0	173	0	0.0
267	56	0	2	140	294	0	2	153	0	1.3
268	57	1	4	140	192	0	0	148	0	0.4
269	67	1	4	160	286	0	2	108	1	1.5

```
display(df.head(5))
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	S c
0	70	1	4	130	322	0	2	109	0	2.4	
1	67	0	3	115	564	0	2	160	0	1.6	
2	57	1	2	124	261	0	0	141	0	0.3	
3	64	1	4	128	263	0	0	105	1	0.2	
4	74	0	2	120	269	0	2	121	1	0.2	

```
num_rows, num_cols = df.shape
print(f"Number of rows: {num_rows}")
print(f"Number of columns: {num_cols}")
```

```
Number of rows: 270
Number of columns: 14
```

```
print("Column Names:")
for col in df.columns:
    print(col)
```

```
Column Names:
Age
Sex
Chest pain type
BP
Cholesterol
FBS over 120
```

```
EKG results
Max HR
Exercise angina
ST depression
Slope of ST
Number of vessels fluro
Thallium
Heart Disease
```

Objective: To identify numerical and categorical variables and their data types. Tasks:

1. Display dataset information using info().
2. Identify numerical and categorical features.
3. Display data types of all columns.
4. Check for missing values.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Age                     270 non-null    int64  
 1   Sex                     270 non-null    int64  
 2   Chest pain type         270 non-null    int64  
 3   BP                      270 non-null    int64  
 4   Cholesterol             270 non-null    int64  
 5   FBS over 120            270 non-null    int64  
 6   EKG results             270 non-null    int64  
 7   Max HR                  270 non-null    int64  
 8   Exercise angina         270 non-null    int64  
 9   ST depression           270 non-null    float64
 10  Slope of ST             270 non-null    int64  
 11  Number of vessels fluro  270 non-null    int64  
 12  Thallium                270 non-null    int64  
 13  Heart Disease           270 non-null    object 
dtypes: float64(1), int64(12), object(1)
memory usage: 29.7+ KB
```

```
numerical_features = df.select_dtypes(include=['int64', 'float64'])
categorical_features = df.select_dtypes(include=['object'])

print("Numerical Features:")
print(numerical_features.columns)

print("\nCategorical Features:")
print(categorical_features.columns)
```

```
Numerical Features:
Index(['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over 120',
```

```
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium'],
      dtype='object')

Categorical Features:
Index(['Heart Disease'], dtype='object')
```

```
print("Data Types of All Columns:")
print(df.dtypes)
```

```
Data Types of All Columns:
Age                         int64
Sex                         int64
Chest pain type             int64
BP                          int64
Cholesterol                 int64
FBS over 120                int64
EKG results                 int64
Max HR                      int64
Exercise angina             int64
ST depression             float64
Slope of ST                 int64
Number of vessels fluro     int64
Thallium                    int64
Heart Disease              object
dtype: object
```

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

```
Age                       0
Sex                       0
Chest pain type           0
BP                        0
Cholesterol               0
FBS over 120              0
EKG results               0
Max HR                    0
Exercise angina           0
ST depression             0
Slope of ST               0
Number of vessels fluro   0
Thallium                  0
Heart Disease             0
dtype: int64
```

Objective: To compute basic descriptive statistics using Pandas and NumPy. Tasks:

1. Display statistical summary using describe().
2. Calculate mean, median, and standard deviation of age.
3. Calculate minimum and maximum cholesterol values.

Double-click (or enter) to edit

```
display(df.describe())
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	27
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	

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

```
age_mean = df['Age'].mean()
age_median = df['Age'].median()
age_std = df['Age'].std()

print(f"Mean Age: {age_mean:.2f}")
print(f"Median Age: {age_median:.2f}")
print(f"Standard Deviation of Age: {age_std:.2f}")
```

```
Mean Age: 54.43
Median Age: 55.00
Standard Deviation of Age: 9.11
```

```
cholesterol_min = df['Cholesterol'].min()
cholesterol_max = df['Cholesterol'].max()

print(f"Minimum Cholesterol: {cholesterol_min}")
print(f"Maximum Cholesterol: {cholesterol_max}")
```

```
Minimum Cholesterol: 126
Maximum Cholesterol: 564
```

Interpretation of Distributions

Age Distribution: The histogram for 'Age' (which you have just plotted) shows a distribution that is somewhat symmetrical, possibly with a slight left-skew. The bulk of the data appears to be concentrated in the middle age ranges, generally between 45 and 65 years. The mean age (54.43) and median age (55.00) are very close, which confirms a relatively symmetrical distribution. This suggests that the heart disease dataset primarily consists of middle-aged to older individuals.

Cholesterol Distribution: The histogram for 'Cholesterol' (which you also plotted) likely indicates a distribution that is moderately right-skewed. This means that while most patients have cholesterol levels within a typical range, there is a tail extending towards higher cholesterol values, indicating a presence of some patients with significantly elevated cholesterol. The mean cholesterol (249.66) being slightly higher than the median (245.00) supports this right-skewness.

Objective: To analyze distributions and frequency of variables. Tasks:

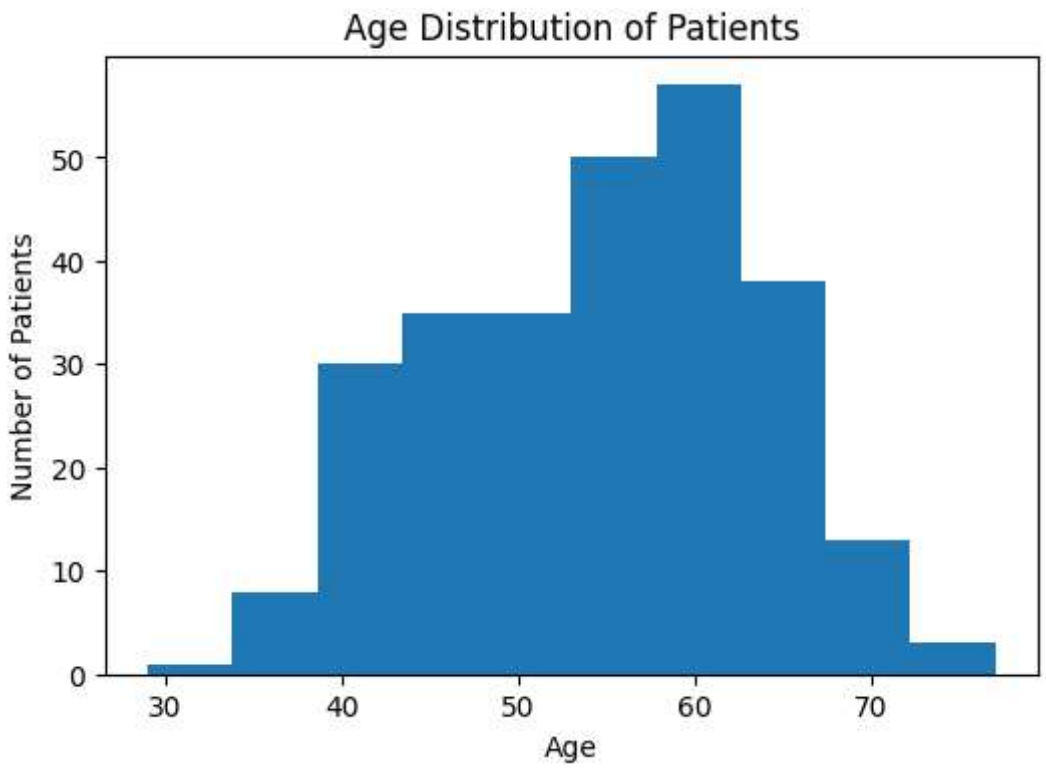
1. Count number of patients with and without heart disease.
2. Plot histogram for age.
3. Plot histogram for cholesterol levels.
4. Interpret distributions.

```
heart_disease_counts = df['Heart Disease'].value_counts()
print("Number of patients with and without heart disease:")
print(heart_disease_counts)
```

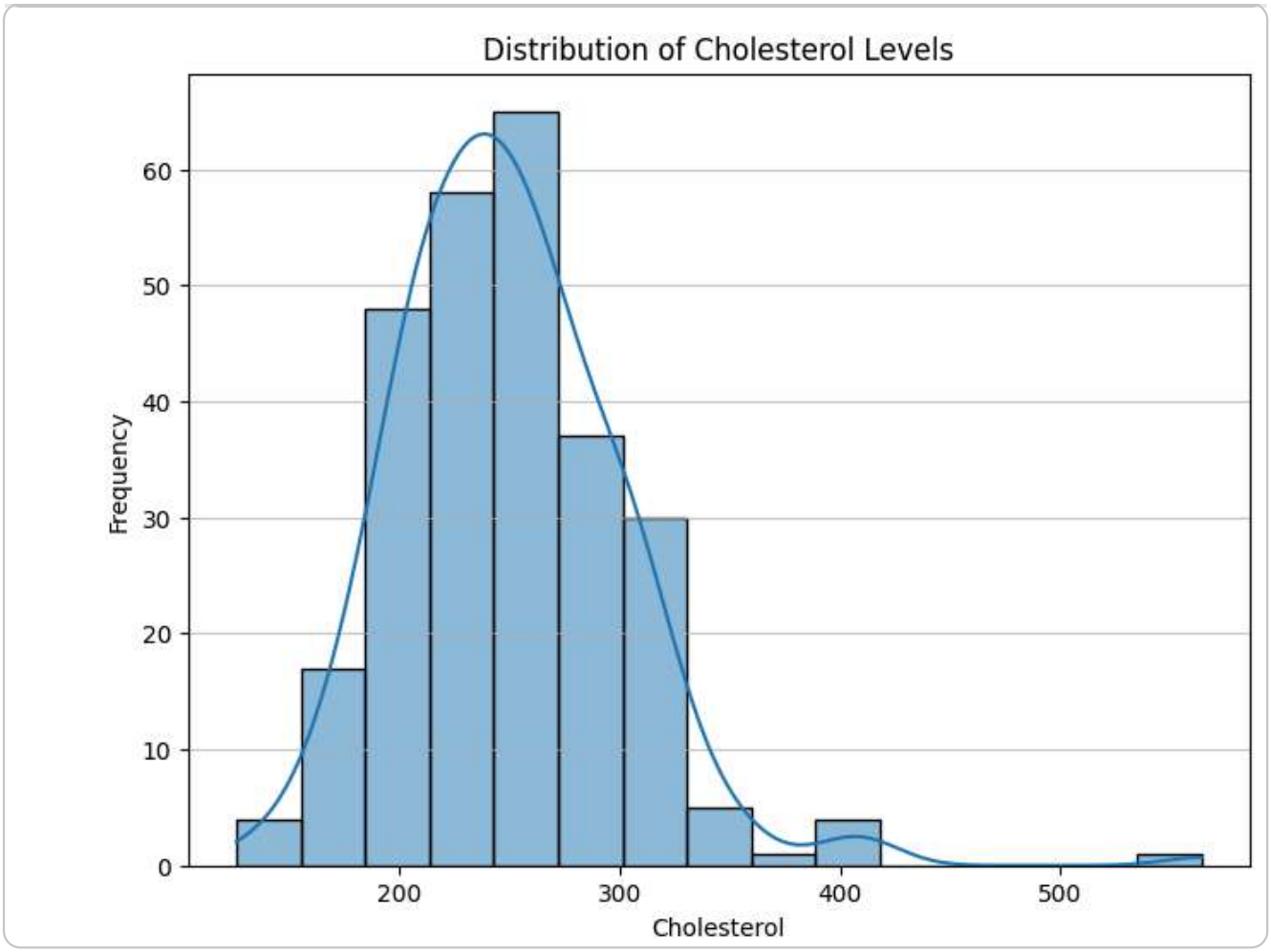
```
Number of patients with and without heart disease:
Heart Disease
Absence     150
Presence    120
Name: count, dtype: int64
```

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

# Plot histogram for age
plt.figure(figsize=(6,4))
plt.hist(df['Age'], bins=10) # Corrected column name to 'Age'
plt.xlabel("Age")
plt.ylabel("Number of Patients")
plt.title("Age Distribution of Patients")
plt.show()
```




```
plt.figure(figsize=(8, 6))
sns.histplot(df['Cholesterol'], kde=True, bins=15)
plt.title('Distribution of Cholesterol Levels')
plt.xlabel('Cholesterol')
plt.ylabel('Frequency')
plt.grid(axis='y', alpha=0.75)
plt.show()
```



Objective: To study relationships between variables using basic exploration. Tasks:

1. Compute correlation matrix.
2. Identify highly correlated variables.
3. Plot scatter plot between age and maximum heart rate.
4. Interpret relationships.

```
correlation_matrix = df.corr(numeric_only=True)
display(correlation_matrix)
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	res
Age	1.000000	-0.094401	0.096920	0.273053	0.220056	0.123458	0.12
Sex	-0.094401	1.000000	0.034636	-0.062693	-0.201647	0.042140	0.03
Chest pain type	0.096920	0.034636	1.000000	-0.043196	0.090465	-0.098537	0.07
BP	0.273053	-0.062693	-0.043196	1.000000	0.173019	0.155681	0.11
Cholesterol	0.220056	-0.201647	0.090465	0.173019	1.000000	0.025186	0.16
FBS over 120	0.123458	0.042140	-0.098537	0.155681	0.025186	1.000000	0.05
EKG results	0.128171	0.039253	0.074325	0.116157	0.167652	0.053499	1.00
Max HR	-0.402215	-0.076101	-0.317682	-0.039136	-0.018739	0.022494	-0.07
Exercise angina	0.098297	0.180022	0.353160	0.082793	0.078243	-0.004107	0.09
ST depression	0.194234	0.097412	0.167244	0.222800	0.027709	-0.025538	0.12
Slope of ST	0.159774	0.050545	0.136900	0.142472	-0.005755	0.044076	0.16
Number of vessels fluro	0.356081	0.086830	0.225890	0.085697	0.126541	0.123774	0.11
Thallium	0.106100	0.391046	0.262659	0.132045	0.028836	0.049237	0.00

Next steps: Generate code with `correlation_matrix` New interactive sheet

```
# Set a correlation threshold
correlation_threshold = 0.5

print(f"Highly correlated variable pairs (absolute correlation > {correlation_

already_checked = set()
for i in range(len(correlation_matrix.columns)):
    for j in range(i + 1, len(correlation_matrix.columns)):
        col1 = correlation_matrix.columns[i]
        col2 = correlation_matrix.columns[j]
        correlation_value = correlation_matrix.iloc[i, j]
```

```
Highly correlated variable pairs (absolute correlation > 0.5):
- ST depression and Slope of ST: 0.61
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Max HR', data=df)
plt.title('Scatter Plot of Age vs. Maximum Heart Rate')
plt.xlabel('Age')
plt.ylabel('Maximum Heart Rate')
plt.grid(True)
plt.show()
```