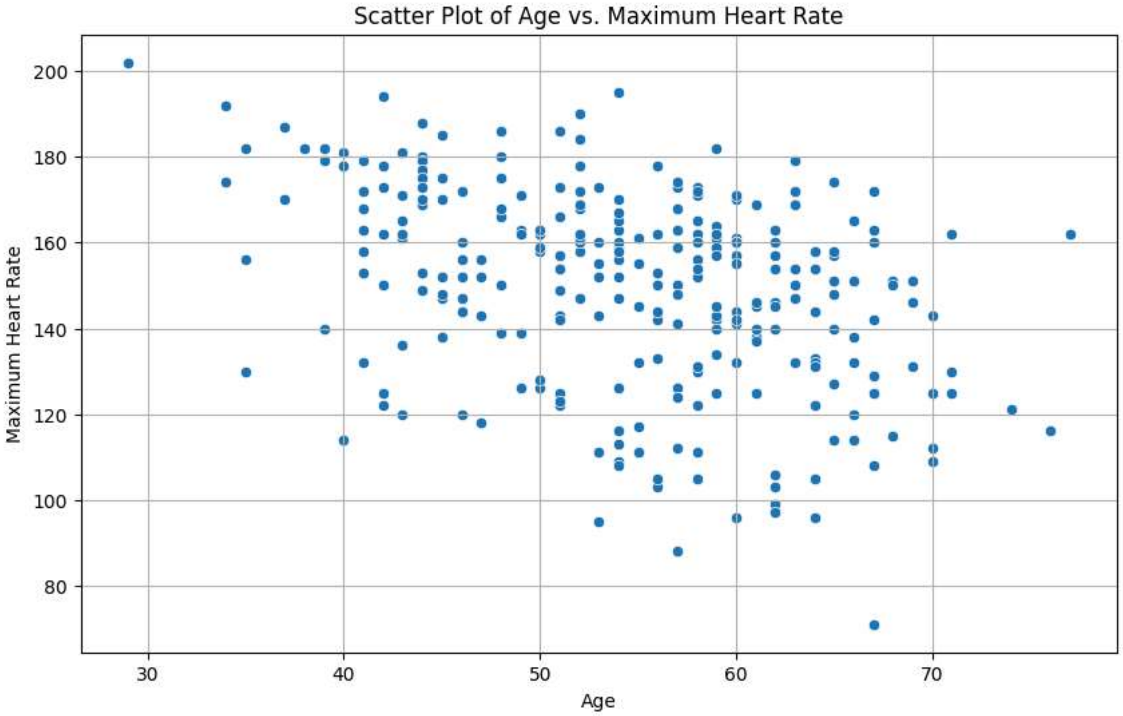
```
Start coding or generate with AI.
```

```
import pandas as pd
df = pd.read_csv('/content/Heart_Disease_Prediction.csv')

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
```

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

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()
```



```
import numpy as np

# Set a correlation threshold
correlation_threshold = 0.7

highly_correlated_pairs = []

# Iterate through the upper triangle of the correlation matrix
for i in range(len(correlation_matrix.columns)):
    for j in range(i + 1, len(correlation_matrix.columns)):
        feature1 = correlation_matrix.columns[i]
        feature2 = correlation_matrix.columns[j]
        correlation_value = correlation_matrix.iloc[i, j]

        if abs(correlation_value) > correlation_threshold:
            highly_correlated_pairs.append((feature1, feature2, correlation_value))

if highly_correlated_pairs:
    print(f"Highly correlated variable pairs (absolute correlation > {correlation_threshold}):")
    for f1, f2, corr_val in highly_correlated_pairs:
        print(f"- {f1} and {f2}: {corr_val:.2f}")
elif not highly_correlated_pairs and correlation_threshold == 0.7:
    # If no pairs found with 0.7, try a lower threshold of 0.5 and print a message to the user.
    correlation_threshold = 0.5
    for i in range(len(correlation_matrix.columns)):
        for j in range(i + 1, len(correlation_matrix.columns)):
            feature1 = correlation_matrix.columns[i]
            feature2 = correlation_matrix.columns[j]
            correlation_value = correlation_matrix.iloc[i, j]

            if abs(correlation_value) > correlation_threshold:
                highly_correlated_pairs.append((feature1, feature2, correlation_value))
    if highly_correlated_pairs:
        print(f"No highly correlated pairs found with threshold 0.7, trying with threshold 0.5:")
        print(f"Highly correlated variable pairs (absolute correlation > {correlation_threshold}):")
        for f1, f2, corr_val in highly_correlated_pairs:
            print(f"- {f1} and {f2}: {corr_val:.2f}")
    else:
        print(f"No highly correlated variable pairs found with absolute correlation > {correlation_threshold}.")
else:
    print(f"No highly correlated variable pairs found with absolute correlation > {correlation_threshold}.")
```

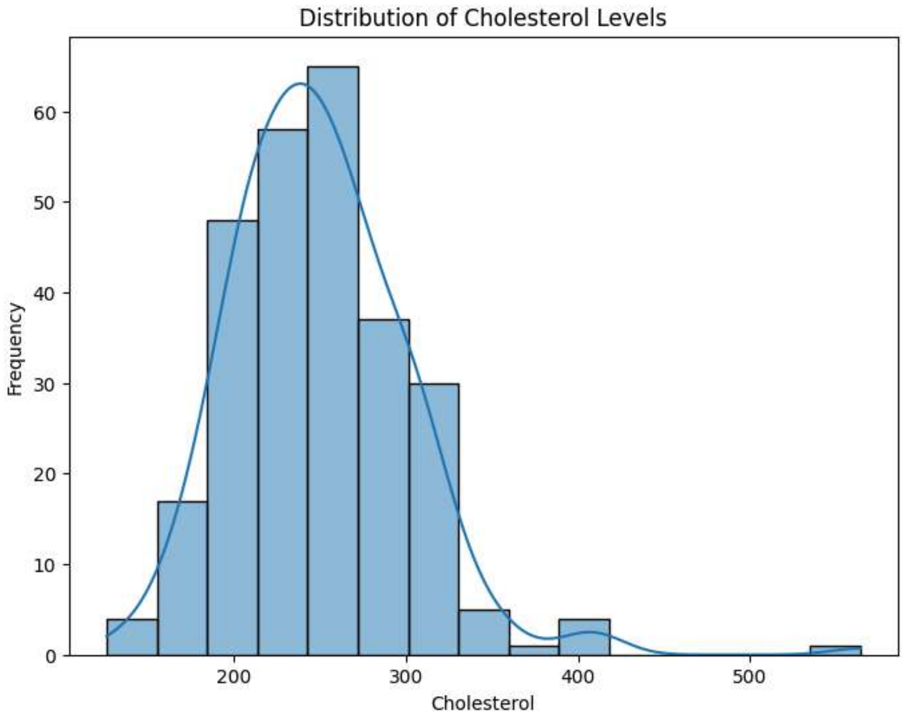
```
No highly correlated pairs found with threshold 0.7, trying with threshold 0.5:
Highly correlated variable pairs (absolute correlation > 0.5):
- ST depression and Slope of ST: 0.61
```

```
numerical_features = df.select_dtypes(include=['int64', 'float64']).columns
correlation_matrix = df[numerical_features].corr()
print('Correlation Matrix:')
display(correlation_matrix)
```

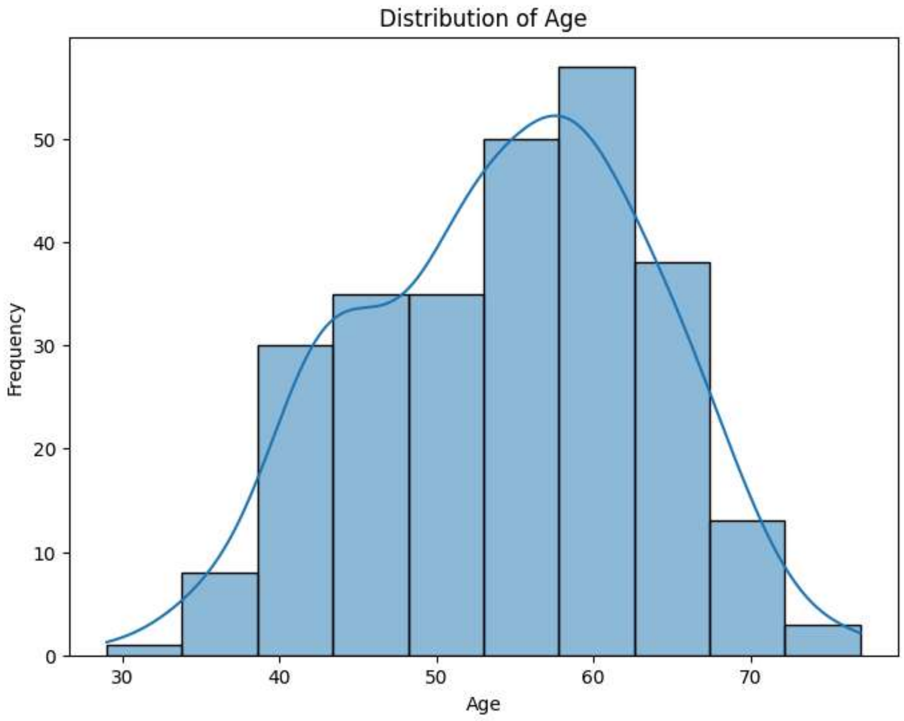
```
Correlation Matrix:
```

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST
Age	1.000000	-0.094401	0.096920	0.273053	0.220056	0.123458	0.128171	-0.402215	0.098297	0.194234	0.159774
Sex	-0.094401	1.000000	0.034636	-0.062693	-0.201647	0.042140	0.039253	-0.076101	0.180022	0.097412	0.050545
Chest pain type	0.096920	0.034636	1.000000	-0.043196	0.090465	-0.098537	0.074325	-0.317682	0.353160	0.167244	0.136900
BP	0.273053	-0.062693	-0.043196	1.000000	0.173019	0.155681	0.116157	-0.039136	0.082793	0.222800	0.142472
Cholesterol	0.220056	-0.201647	0.090465	0.173019	1.000000	0.025186	0.167652	-0.018739	0.078243	0.027709	-0.005755
FBS over 120	0.123458	0.042140	-0.098537	0.155681	0.025186	1.000000	0.053499	0.022494	-0.004107	-0.025538	0.044076
EKG results	0.128171	0.039253	0.074325	0.116157	0.167652	0.053499	1.000000	-0.074628	0.095098	0.120034	0.160614
Max HR	-0.402215	-0.076101	-0.317682	-0.039136	-0.018739	0.022494	-0.074628	1.000000	-0.380719	-0.349045	-0.386847
Exercise angina	0.098297	0.180022	0.353160	0.082793	0.078243	-0.004107	0.095098	-0.380719	1.000000	0.274672	0.255908
ST depression	0.194234	0.097412	0.167244	0.222800	0.027709	-0.025538	0.120034	-0.349045	0.274672	1.000000	0.609712
Slope of ST	0.159774	0.050545	0.136900	0.142472	-0.005755	0.044076	0.160614	-0.386847	0.255908	0.609712	1.000000
Number of											

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

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.show()
```



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

plt.figure(figsize=(8, 6))
sns.histplot(df['Age'], kde=True, bins=10)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



```
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
```

```
min_cholesterol = df['Cholesterol'].min()
max_cholesterol = df['Cholesterol'].max()

print(f"Minimum Cholesterol: {min_cholesterol}")
print(f"Maximum Cholesterol: {max_cholesterol}")
```

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

```
print('Missing values per column:')
print(df.isnull().sum())
```

```
Missing values per column:
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
```

```
print(df.dtypes)
```

```
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
```

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

print(f"Numerical features: {numerical_features}")
print(f"Categorical features: {categorical_features}")
```

```
Numerical features: ['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over 120', 'EKG results', 'Max HR', 'Exercise an
Categorical features: ['Heart Disease']
```

```
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
```

```
print('Column names:')
print(df.columns)
```

```
Column names:
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', 'Heart Disease'],
      dtype='object')
```

```
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('First 5 rows:')
display(df.head())

print('\nLast 5 rows:')
display(df.tail())
```

```
First 5 rows:
```

	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
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence
4	74	0	2	120	269	0	2	121	1	0.2	1	1	3	Absence

```
Last 5 rows:
```

	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
265	52	1	3	172	199	1	0	162	0	0.5	1	0	7	Absence
266	44	1	2	120	263	0	0	173	0	0.0	1	0	7	Absence
267	56	0	2	140	294	0	2	153	0	1.3	2	0	3	Absence
268	57	1	4	140	192	0	0	148	0	0.4	2	0	6	Absence

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

	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
0	70	1	4	130	322	0	2	109	0	2.4	2	3	3	Presence
1	67	0	3	115	564	0	2	160	0	1.6	2	0	7	Absence
2	57	1	2	124	261	0	0	141	0	0.3	1	0	7	Presence
3	64	1	4	128	263	0	0	105	1	0.2	2	1	7	Absence

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