Heart Disease Analysis:

The project focused on analyzing a medical dataset to extract insights about patient characteristics and assess their likelihood of having heart disease. This involved data cleaning, preprocessing, visualization using Python, and further exploration in Power BI.

Tasks Involved:

- **Dataset Overview: The dataset included patient information such as medical history and indicators that could predict heart disease.**
- **Data Cleaning and Preprocessing:**
  - **Pandas: Used for cleaning the dataset by managing missing values, normalizing data, and removing irrelevant columns.**
  - **Saved the cleaned dataset into a new CSV file for streamlined access and further analysis.**
- **Data Visualization with Python:**
  - **Bar Plot: Visualized the mean values of key medical indicators (cholesterol, blood pressure, heart rate, and oldpeak) by gender, revealing gender-specific trends.**
  - **Correlation Heatmap: Displayed the relationships between features like age, cholesterol, and the presence of heart disease, highlighting strong connections.**

- **Histograms: Showed the distribution of critical variables (age, cholesterol, heart rate), revealing patterns and trends within the dataset.**
- **Pair Plot: Illustrated feature relationships with heart disease presence, highlighting trends between patients with and without the condition.**

- **Visualization with Power BI:**
  - **Imported the cleaned CSV file into Power BI.**
  - **Created interactive dashboards and reports to present the patient data and provide insights into heart disease risk factors.**
  - **Developed dynamic visualizations to explore patient data and identify risk factors associated with heart disease.**

Data Import and Setup: The project began by importing essential libraries for data analysis and visualization:

- **Pandas for data manipulation**
- **NumPy for numerical operations**
- **SciPy for statistical functions**
- **Seaborn and Matplotlib for visualization The dataset, loaded from a CSV file, contained various medical data relevant to heart disease analysis.**

```
In [2]: import pandas as pd
        import numpy as np
        from scipy.stats import zscore
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [4]: data = pd.read_csv("D:/heart_cleveland_upload.csv")
        data.head()
```

Out[4]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	0
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	0
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	0
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	1
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	0

Data Overview and Inspection:

I used the data.info() function to inspect the dataset, which provided an overview of the data types for each column and helped identify any null values. This step was crucial in ensuring that all fields were correctly populated, while also flagging any potential data cleaning requirements before further analysis.

```
In [5]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 297 entries, 0 to 296
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        297 non-null    int64
 1   sex        297 non-null    int64
 2   cp         297 non-null    int64
 3   trestbps   297 non-null    int64
 4   chol       297 non-null    int64
 5   fbs        297 non-null    int64
 6   restecg    297 non-null    int64
 7   thalach    297 non-null    int64
 8   exang      297 non-null    int64
 9   oldpeak    297 non-null    float64
 10  slope      297 non-null    int64
 11  ca         297 non-null    int64
 12  thal       297 non-null    int64
 13  condition  297 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 32.6 KB
```

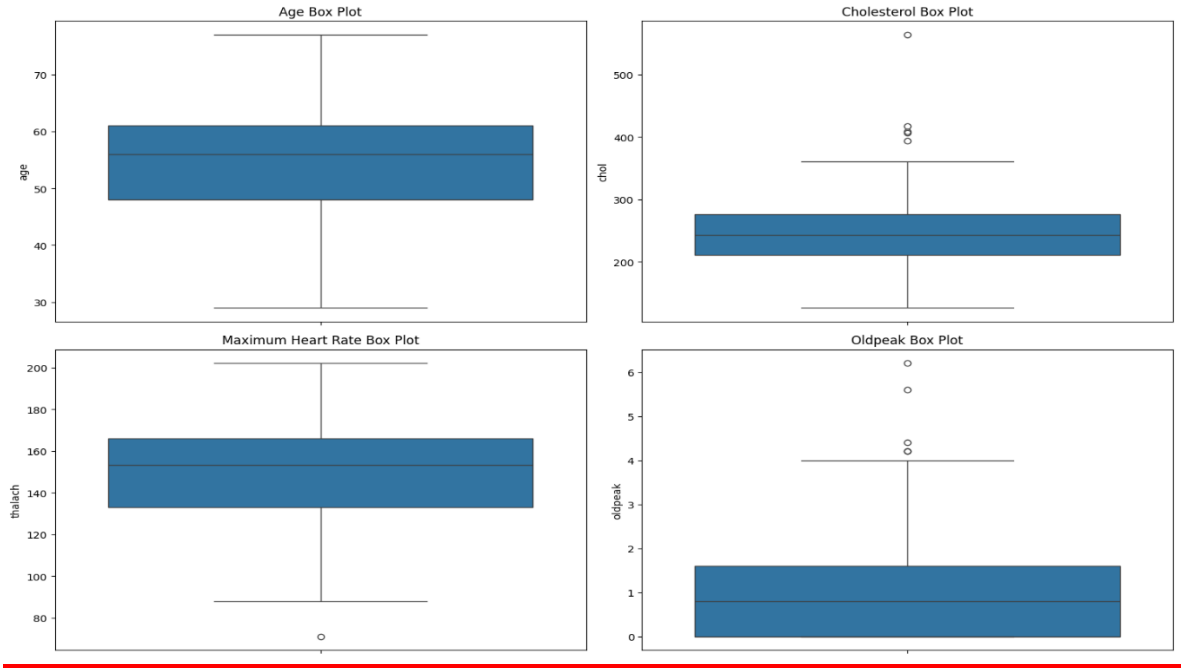
Outlier Detection:

I implemented a function to detect outliers using the Interquartile Range (IQR) method. This approach calculates the IQR by subtracting the first quartile (Q1) from the third quartile (Q3). Outliers are identified if they fall below Q1−1.5×IQRQ1 - 1.5 \times IQRQ1−1.5×IQR or above Q3+1.5×IQRQ3 + 1.5 \times IQRQ3+1.5×IQR for the specified columns. This method helped pinpoint data points that significantly deviated from the typical range, ensuring the dataset's integrity before analysis.

```
]: def detect_outliers(data, columns):
       for column in columns:
           Q1 = data[column].quantile(0.25)
           Q3 = data[column].quantile(0.75)
           IQR = Q3 - Q1
           lower_bound = Q1 - 1.5 * IQR
           upper_bound = Q3 + 1.5 * IQR
           outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]
           print(f"Outliers in {column}:")
           print(outliers)
           print(f"Number of outliers in {column}: {outliers.shape[0]}\n")
```

```
]: numeric_columns = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
   detect_outliers(data , numeric_columns)
```

I used the boxplot to check the outliers in the data then I found that there is some outliers in the data.



Outlier Treatment:

To handle outliers in the medical dataset, I opted to replace them rather than removing them, given the sensitivity of the data. I implemented a function, winsorize_outliers, which addresses extreme values by adjusting them to the nearest value within a specified z-score threshold. This function calculates z-scores to identify outliers and then replaces them with values close to the median within a defined range. This approach helps preserve the dataset's integrity while reducing the influence of extreme values.

```python
def winsorize_outliers(data, column, z_thresh=3):
    """
    Replace values with the nearest within the z-score threshold.
    """
    z_scores = np.abs(zscore(data[column]))
    median = data[column].median()
    std_dev = data[column].std()

    # Find outliers
    outliers = z_scores > z_thresh
    # Replace outliers with the nearest value within the z-score threshold
    data.loc[outliers, column] = np.clip(
        data.loc[outliers, column],
        median - z_thresh * std_dev,
        median + z_thresh * std_dev
    )

    return data
```

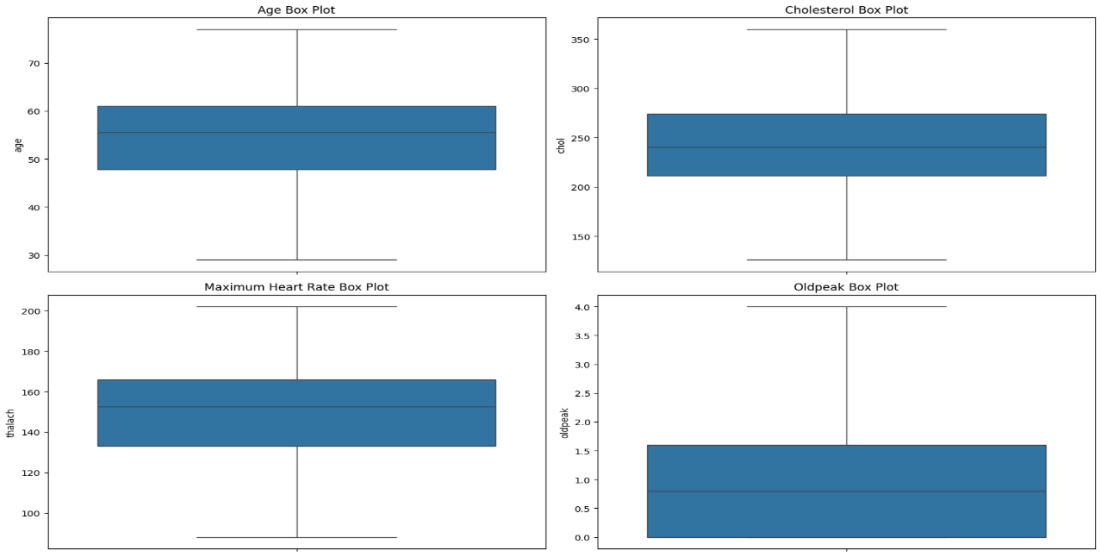
Outlier Replacement and Data Cleaning:

I applied outlier treatment to several key columns in the dataset to ensure data accuracy while maintaining the integrity of the medical data. The treatment involved both Winsorization and replacement of outliers using the Interquartile Range (IQR) method. This combined approach helped manage extreme values effectively.

```
columns_to_treat = ['trestbps', 'chol', 'thalach', 'oldpeak']

# Apply Winsorization to specified columns
for col in columns_to_treat:
    if col in ['trestbps', 'chol']:
        data = winsorize_outliers(data, col)
    else:
        q1 = data[col].quantile(0.25)
        q3 = data[col].quantile(0.75)
        iqr = q3 - q1
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr
        median = data[col].median()
        data[col] = np.where(data[col] < lower_bound, median, data[col])
        data[col] = np.where(data[col] > upper_bound, median, data[col])

# Identify and drop rows with outliers in the 'chol' column
q1_chol = data['chol'].quantile(0.25)
q3_chol = data['chol'].quantile(0.75)
iqr_chol = q3_chol - q1_chol
lower_bound_chol = q1_chol - 1.5 * iqr_chol
upper_bound_chol = q3_chol + 1.5 * iqr_chol
data_cleaned = data[~((data['chol'] < lower_bound_chol) | (data['chol'] > upper_bound_chol))]
```

Outlier Verification with Boxplots:

To confirm that the outliers were effectively replaced and validate the cleaning process, I used boxplots for visual inspection. These plots offered a clear view of the data distribution and helped verify that the outliers had been addressed properly.



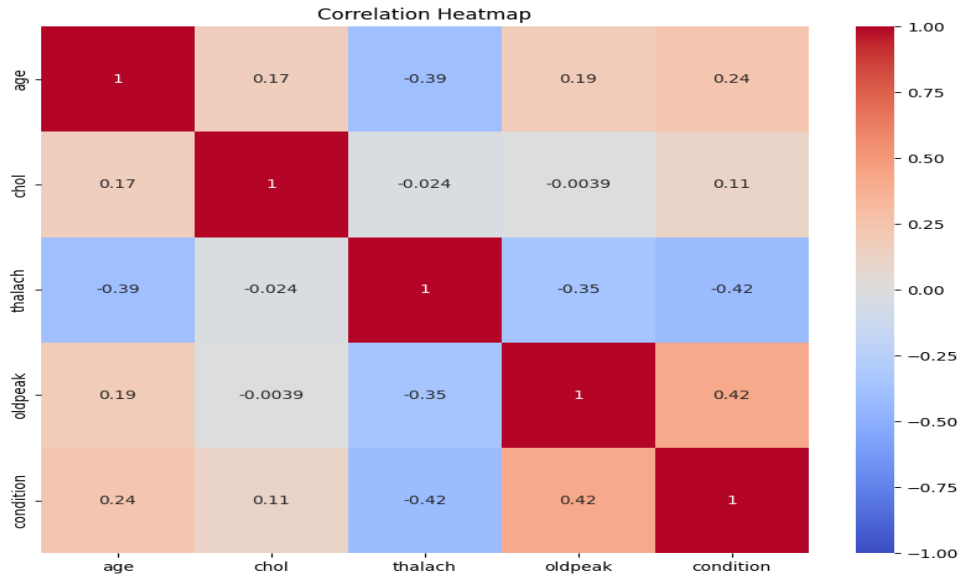
Data Export for Visualization:

After handling outliers and cleaning the dataset, I saved the processed data to a new CSV file. This cleaned dataset was then utilized for visualization in both Python libraries and Power BI to uncover trends and insights.

```
data_cleaned.to_csv('The_treated_dataset.csv', index=False)
```

Correlation Analysis:

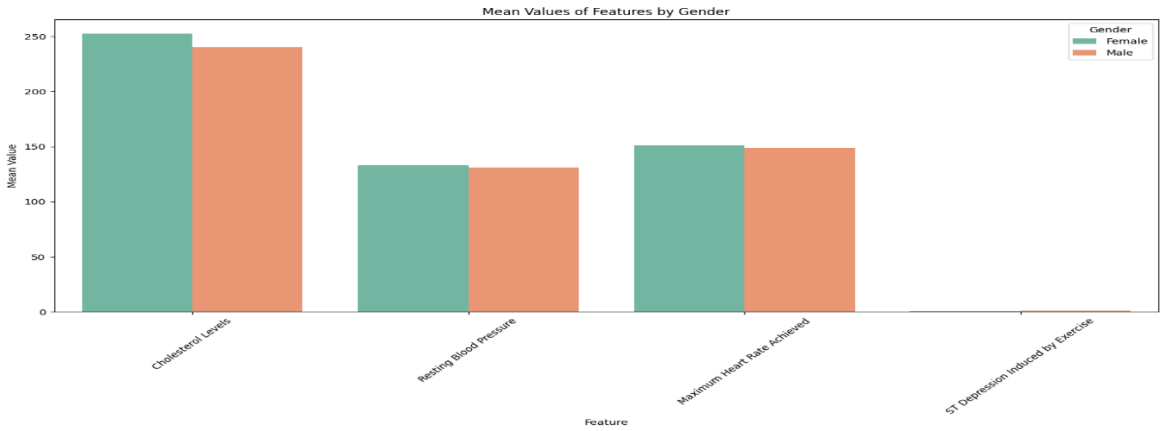
To explore the relationships between different features in the dataset and aid in visualization, I performed a correlation analysis. This analysis helped identify how various features are related to one another and to the target variable, providing valuable insights for data interpretation and visualization.



Bar Plot Analysis: Gender Comparison

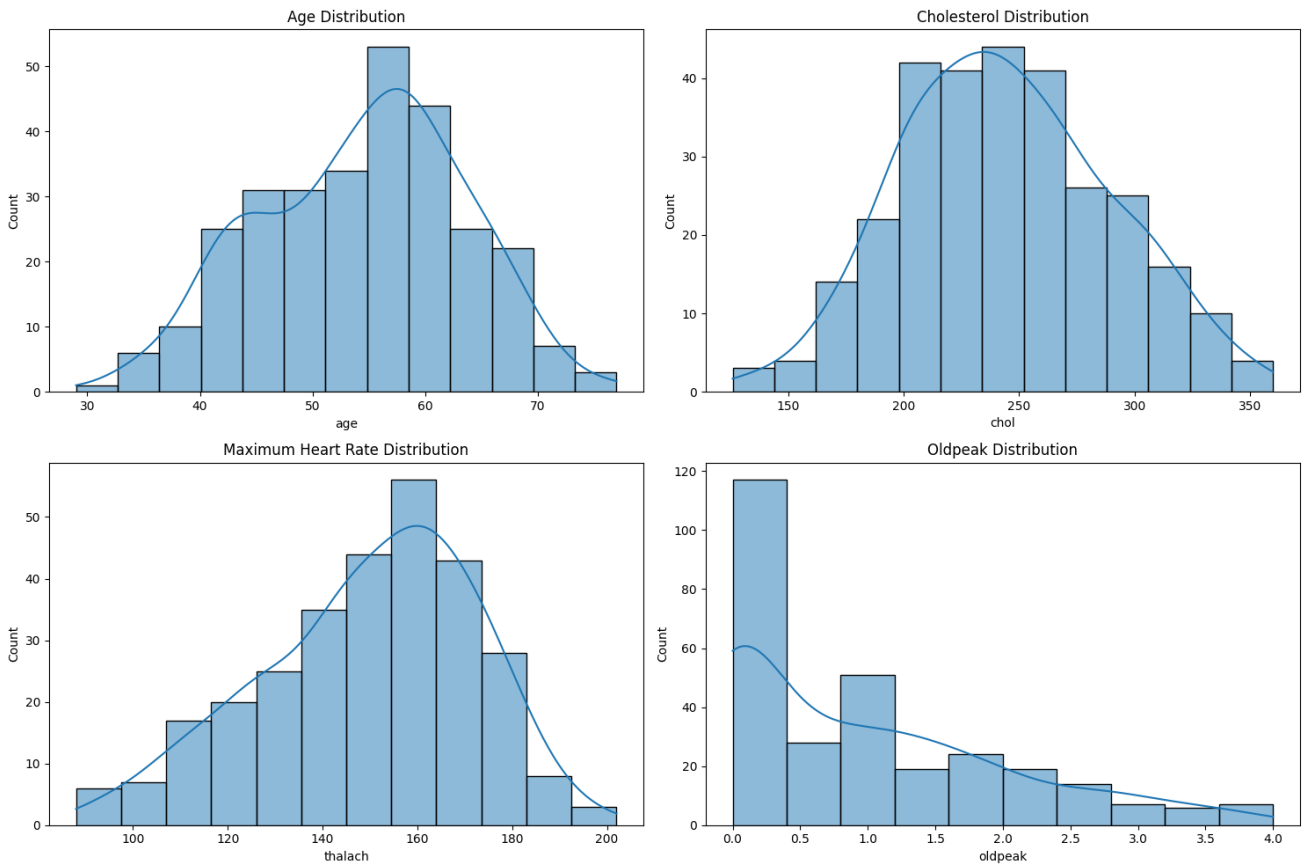
- **Purpose: The bar plot was designed to compare the mean values of various medical features between males and females. This visual representation helps in understanding gender-based differences in key medical metrics.**
- **Plot Description:**
- **Y-Axis: Displays the mean values of the selected medical features.**
- **X-Axis: Lists the features being compared.**
- **Colors: Different colors are used to represent males and females, providing a clear visual distinction between the genders.**
- **Legend: Includes a legend to identify which color corresponds to which gender.**
- **X-Axis Labels: Labels are rotated for better readability.**

```
mean_values = data_cleaned.groupby(['sex']).mean().reset_index()
mean_values = pd.melt(mean_values, id_vars=['sex'], value_vars=['chol', 'trestbps', 'thalach', 'oldpeak'],
                      var_name='feature', value_name='mean_value')

mean_values['sex'] = mean_values['sex'].map({0: 'Female', 1: 'Male'})

mean_values['feature'] = mean_values['feature'].map(feature_names)

plt.figure(figsize=(14, 8))
sns.barplot(x='feature', y='mean_value', hue='sex', data=mean_values, palette='Set2')


plt.title('Mean Values of Features by Gender')
plt.xlabel('Feature')
plt.ylabel('Mean Value')
plt.legend(title='Gender')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Histogram Analysis: Frequency Distributions

Purpose: Histograms were used to visualize the distribution and frequency of key medical features in the dataset. This approach helps in understanding the spread and central tendencies of various variables.



Pair Plot Analysis: Relationships and Effects on Heart Disease

Purpose: The pair plot visualizes the relationships between key medical features and their association with heart disease. It is used to identify patterns and correlations between features and the target variable (heart disease).

Analysis:

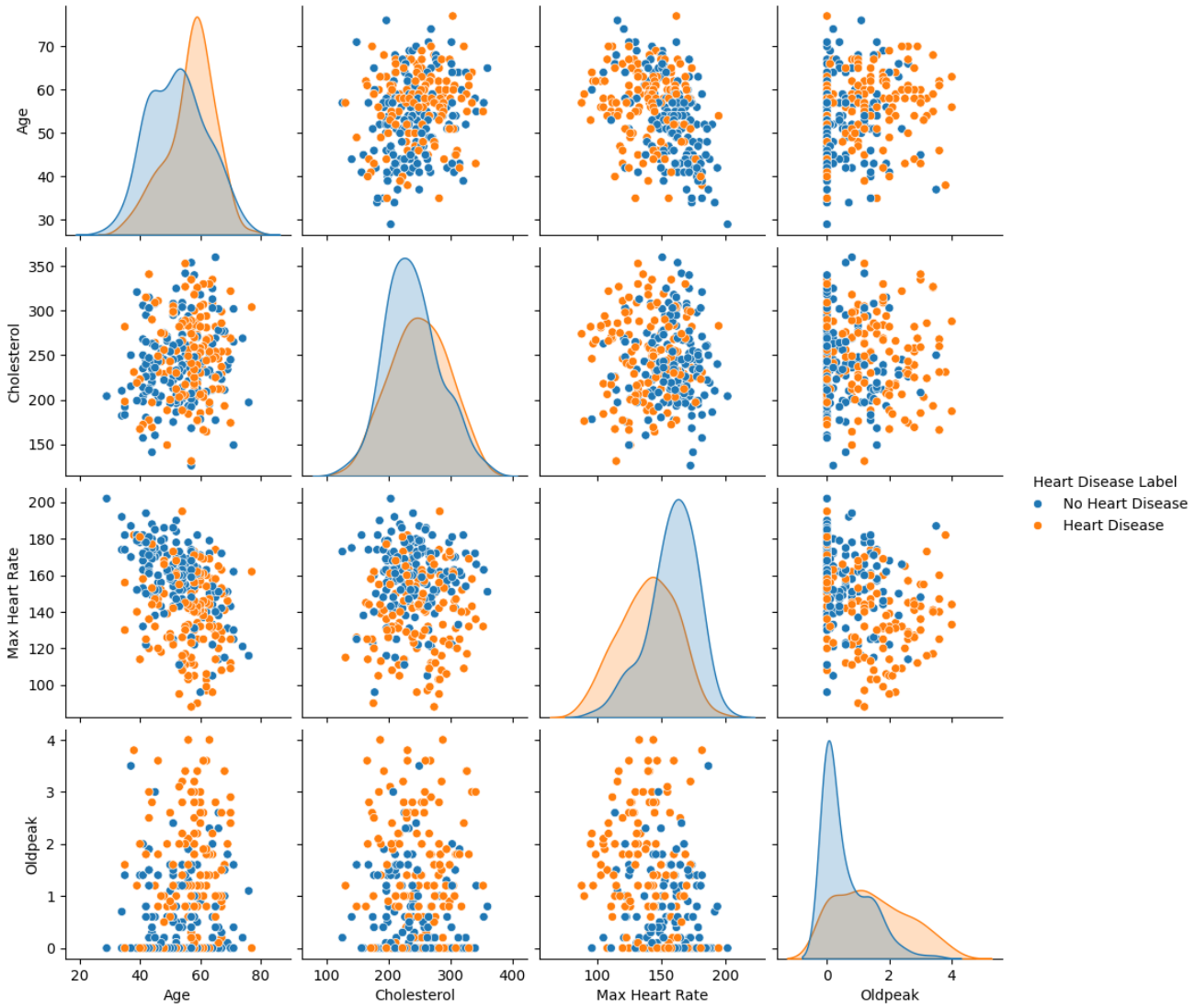
- **Scatter Plots: Display relationships between pairs of features, revealing correlations or patterns related to heart disease.**
- **Histograms: Illustrate the distribution of each feature, helping to understand how features vary among patients with different heart disease statuses.**

Insights:

- **Reveals clusters or trends, such as whether patients with heart disease have higher cholesterol levels or lower maximum heart rates.**

- **Highlights relationships between features that may influence heart disease risk.**
- **Provides a comprehensive view of how key medical features relate to heart disease, aiding in the identification of significant patterns and correlations.**

```
data_cleaned.rename(columns={
    'age': 'Age',
    'chol': 'Cholesterol',
    'thalach': 'Max Heart Rate',
    'oldpeak': 'Oldpeak',
    'condition': 'Heart Disease Present'
}, inplace=True)

# Map condition values to descriptive labels
data_cleaned['Heart Disease Label'] = data_cleaned['Heart Disease Present'].map({0: 'No Heart Disease', 1: 'Heart Disease'})

# Generate the pair plot with renamed columns and updated condition labels
sns.pairplot(data_cleaned[['Age', 'Cholesterol', 'Max Heart Rate', 'Oldpeak', 'Heart Disease Label']], hue='Heart Disease Label'

plt.show()
```


Age
Cholesterol
Max Heart Rate
Oldpeak
Heart Disease Label
No Heart Disease
Heart Disease

Final Visualization and Power BI Integration

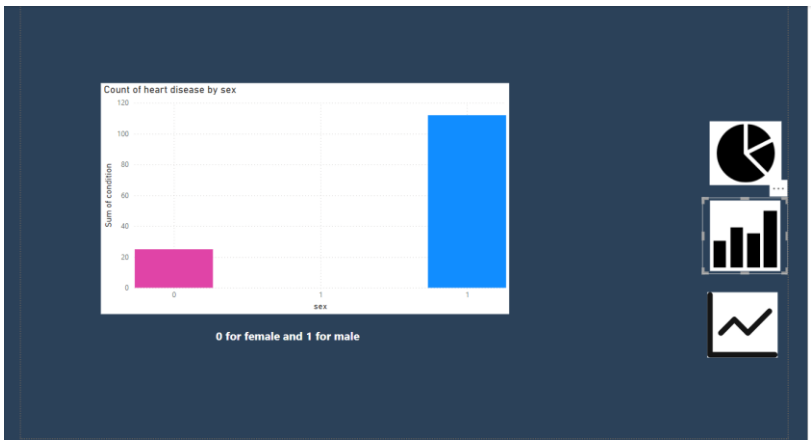
Purpose: To enhance data visualization and presentation, Power BI was utilized to create interactive and insightful dashboards. This tool facilitates comprehensive analysis and visualization of the cleaned dataset.

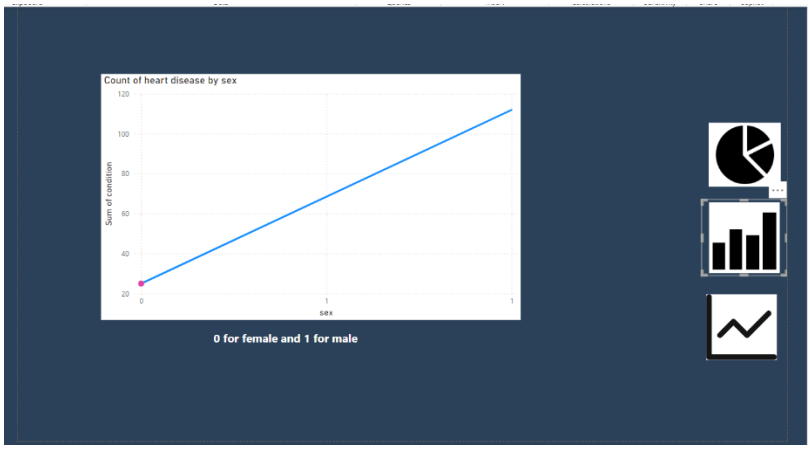
Tasks Involved:

1. **Data Import and Preparation:**
    - **Imported the cleaned dataset into Power BI for advanced analysis and visualization.**
2. **Creating Visualizations:**
    - **Developed a range of visualizations, including bar charts, pie charts, and line plots, to represent key features and their relationships.**
3. **Using Bookmarks:**
    - **Implemented bookmarks to create a dynamic report with interactive elements, allowing users to save specific views and easily navigate between different visualizations.**


Sum of exang by cp
Sum of exang
cp
Sum of condition by age
Sum of condition
age
Sum of chol by age
Sum of chol
age
Average of condition by restecg
0.37 (22.47%)
0.75 (45.04%)
0.54 (32.49%)
restecg
Sum of condition by restecg and ca
Sum of condition
ca
1 in restecg refere to Presence of abnormal changes in the ST-T wave.
and 2 in ca refer to Presence of narrowing in two major blood vessels
Filters
Page 1
Page 2


Count of heart disease by sex
25 (18.25%)
112 (81.75%)
Sex
0 for female and 1 for male


Count of heart disease by sex
Sum of condition
sex
0 for female and 1 for male



5. Key Learnings

Summary of Important Points:

- **Data Transformation Techniques: Acquired skills in converting raw data into actionable insights by focusing on key metrics.**
- **Effective Report Design: Developed the ability to design reports that emphasize crucial facts for decision-making.**
- **Predictive Analytics: Gained a solid understanding of predictive modeling to forecast future trends.**

Skills Acquired:

- **Proficiency in Data Analytics Tools: Gained hands-on experience with Power BI and other analytics tools for effective data visualization and analysis.**

- **Data Visualization Techniques: Developed skills in creating compelling and informative visualizations to communicate data clearly.**
- **Predictive Modeling: Learned how to build and interpret predictive models to make data-driven decisions in real-world applications.**