Team Alpha

HEALTH DATA ANALYSIS 2024

ARTIFICIAL INTELLIGENCE

DATE :
24 August 2024

PRESENTED TO :
Sir Muhammad Rizwan

PATIENT HEALTH DATA ANALYSIS AND VISUALIZATION

OVERVIEW

In this project, we aim to analyze and visualize patient health data using Python. The tools used include NumPy and pandas for data manipulation, and Matplotlib for data visualization. Our goal is to uncover insights into health trends such as blood pressure levels, cholesterol, and other vital statistics, which can aid in medical decision-making.

OBJECTIVES

- **Utilize Libraries:** Learn to effectively use Matplotlib, NumPy, and pandas in the context of medical data.
- **Data Handling:** Load, clean, and manipulate patient health data to ensure accuracy.
- **Statistical Analysis:** Conduct basic statistical analyses using NumPy.
- **Data Visualization:** Create visualizations to interpret health trends and patterns.
- **Comprehensive Analysis:** Combine visualizations to offer a holistic view of patient health data.

DATASET

The dataset comprises health records collected from a medical center, featuring several key columns. Each entry is uniquely identified by a **Patient ID**. The dataset includes the **Age** and **Gender** of the patient, as well as their **Blood Pressure (BP)**, which records both systolic and diastolic readings. **Cholesterol** levels are also noted, along with the patient's **Heart Rate** measured at rest. The **Body Mass Index (BMI)** of each patient is calculated and recorded. Additionally, there is an indicator for **Diabetes**, specifying whether the patient has diabetes (Yes/No).

Corvit System Muzaffarabad

Team Members

Name	CNIC
Syed Mansoor ul Hassan Bukhari	82601-0343458-9
Syed Taqi Haider	82203-9180312-5

1. Installation and Setup

Objective:

- Install the required libraries.

```
# !pip install , pandas, matplotlib
```

Installing Libraries

Installing `NumPy`, `Pandas`, and `Matplotlib` is an important step as these libraries are essential tools for data manipulation and data visualization. Since I already have these libraries installed, I have commented out the installation code. If you need to install these libraries, you can uncomment the codabovetlib

---

2. Import Libraries

Objective:

- Import `Matplotlib`, `NumPy`, and `pandas`

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

Importing Required Libraries

We import the necessary libraries using the `import` keyword and give them aliases with the `as` keyword for convenience. The libraries imported are:

- **NumPy**: For numerical operations
- **Pandas**: For data manipulation
- **Matplotlib**: For data visualization

---

3. Loading and Exploring Dataset

Objective:

- Load the patient health data and examine its structure.

Load Dataset

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

Analyze rows and columns

```
df.shape

(100000, 8)
```

Check for null values

```
df.isna().sum()
Patient ID                    0
Age                        5000
Gender                     5000
Blood Pressure (BP)        5000
Cholesterol                5000
Heart Rate                 5000
Body Mass Index (BMI)      5000
Diabetes                   5000
dtype: int64
```

Display in table format

```
df.head(3)
   Patient ID   Age  Gender Blood Pressure (BP)  Cholesterol  Heart
Rate  \

0           1  69.0  Female              169/113        170.0
83.0
1           2  96.0     NaN                  NaN        129.0
65.0
2           3   5.0    Male               116/61        167.0
98.0
   Body Mass Index (BMI) Diabetes
0                  39.7       No
1                  22.2       No
2                  28.0       No
```

Check for datatypes of attributes

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999 Data
columns (total 8 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
0   Patient ID             100000 non-null  int64
1   Age                    95000 non-null   float64
```

```
2   Gender                  95000 non-null   object  3
    Blood Pressure (BP)     95000 non-null   object
4   Cholesterol             95000 non-null   float64
5   Heart Rate              95000 non-null   float64
6   Body Mass Index (BMI)  95000 non-null   float64  7
    Diabetes                95000 non-null   object
dtypes: float64(4), int64(1), object(3) memory
usage: 6.1+ MB
```

Check max, min, mean of data

```
df.describe()
          Patient ID            Age   Cholesterol    Heart Rate  \
count  100000.000000  95000.000000  95000.000000  95000.000000
mean    50000.500000     50.543389    199.432674     84.509758
std     28867.657797     28.863072     57.719658     20.201143
min         1.000000      1.000000    100.000000     50.000000
25%     25000.750000     26.000000    149.000000     67.000000
50%     50000.500000     50.000000    199.000000     85.000000
75%     75000.250000     76.000000    250.000000    102.000000
max    100000.000000    100.000000    299.000000    119.000000

       Body Mass Index (BMI)
count           95000.000000
mean               27.491998
std                 7.227360
min                15.000000
25%                21.200000
50%                27.500000
75%                33.800000
max                40.000000
```

Data Loading and Initial Exploration

We start by loading the data into a DataFrame `df` from a CSV file and performing some initial operations to understand the dataset.

Missing Values

The dataset contains 100,000 rows, and the following columns have missing values:

- **Age**: 5,000 missing values
- **Gender**: 5,000 missing values
- **Blood Pressure (BP)**: 5,000 missing values
- **Cholesterol**: 5,000 missing values
- **Heart Rate**: 5,000 missing values
- **Body Mass Index (BMI)**: 5,000 missing values
- **Diabetes**: 5,000 missing values

Dataset Shape

- **Number of Rows**: 100,000
- **Number of Columns**: 8 Data Format

The dataset contains the following columns with their respective data types:

- **Patient ID**: `int64`
- **Age**: `float64`
- **Gender**: `object`
- **Blood Pressure (BP)**: `object`
- **Cholesterol**: `float64`
- **Heart Rate**: `float64`
- **Body Mass Index (BMI)**: `f`7 Diabetes 95000 non-null object 5000

---

4. Data Cleaning and Manipulation

Objective

- Handle missing values, categorize data, and calculate new metrics (e.g., BMI categories).

Handle missing values

We fill `Age` with median and `Cholesterol` with mean and after that we drop all null values

---

Fill Missing Values

```
fill_values = {
    'Age': df['Age'].median(),
    'Cholesterol': df['Cholesterol'].mean()
}

df = df.fillna(value=fill_values)
```

Drop Null Values

```
df.dropna(inplace=True)
```

Organize Blood Pressure

```
# Split the 'Blood Pressure (BP)' column into 'Systolic' and
'Diastolic'
df[['Systolic',      'Diastolic']]      =      df['Blood      Pressure
(BP)'].str.split('/', expand=True)

# Convert these new columns to numeric type
```

```
df['Systolic']    =    pd.to_numeric(df['Systolic'],    errors='coerce')
df['Diastolic'] = pd.to_numeric(df['Diastolic'], errors='coerce')

#    Drop    the    original    'Blood    Pressure    (BP)'    column
df.drop(columns=['Blood Pressure (BP)'], inplace=True)
```

Calculate New Metrics

Define BMI categories

```
def categorize_bmi(bmi):
    if bmi < 18.5:
        return 'Underweight'
    elif 18.5 <= bmi < 24.9:
        return 'Normal weight'
    elif 25 <= bmi < 29.9:
        return 'Overweight'
    else:
        return 'Obesity'
```

```
df['BMI Category'] = df['Body Mass Index (BMI)'].apply(categorize_bmi)
```

Blood Pressure Categories

```
def categorize_blood_pressure(systolic, diastolic):
    if systolic < 120 and diastolic < 80:
        return 'Normal'
    elif 120 <= systolic < 130 and diastolic < 80:
        return 'Elevated'
    elif 130 <= systolic < 140 or 80 <= diastolic < 90:
        return 'Hypertension Stage 1'
    else:
        return 'Hypertension Stage 2'
```

```
# Apply  function to create a new column for Blood Pressure category
df['BP Category'] = df.apply(
    lambda row: categorize_blood_pressure(row['Systolic'], row['Diastolic']),
    axis=1
)
```

Age Categories

```python
def age_categories(age):
    if age < 13:
        return 'Child'
    elif 13 <= age < 20:
        return 'Teenager'
    elif 20 <= age < 40:
        return 'Adult'
    elif 40 <= age < 60:
        return 'Middle-aged'
    else:
        return 'Senior'


# Apply age_categories function to create a new column for Age Category
df['Age Category'] = df['Age'].apply(age_categories)
```

Cholesterol Categories

```python
def categorize_cholesterol(cholesterol):
    if cholesterol < 200:
        return 'Desirable'
    elif 200 <= cholesterol < 240:
        return 'Borderline High'
    else:
        return 'High'

# Apply function to create a new column for Cholesterol Category
df['Cholesterol Category'] = df['Cholesterol'].apply(categorize_cholesterol)
```

Data Cleaning and Transformation

Filling Missing Values

We handle missing values as follows:

- **Age**: Filled with the median value.
- **Cholesterol:** Filled with the mean value.

After filling in missing values, we drop all remaining `null` values, resulting in a new data shape of **77,381 rows** and **8 columns**.

Splitting Blood Pressure

We split the `Blood Pressure` column into two separate columns:

- **Systolic**
- **Diastolic**

The original `Blood Pressure` column is then removed, and we now have **9 columns**.

Data Categorization

We categorize our data by creating new metrics for:

- **BMI**: `BMI Category`
- **Age**: `Age Category`
- **Cholesterol**: `Cholesterol Category`
- **Blood Pressure**: `BP Category`

After these data set have `13` columns and `77381` rows.

---

5. Visualizing Blood Pressure Trends

Objective

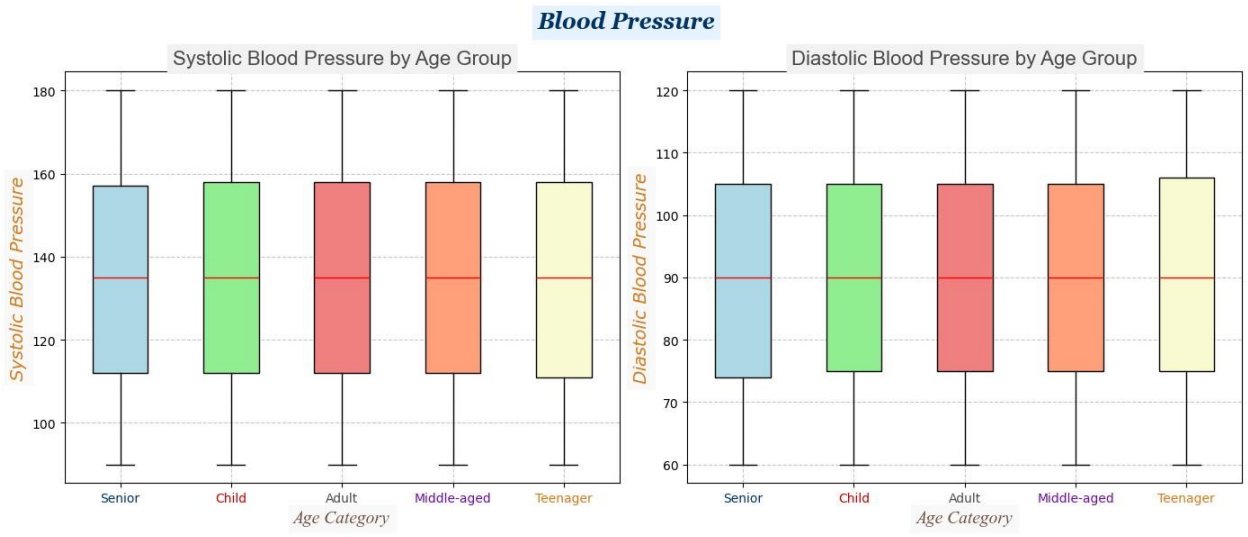
- Plot the distribution of systolic and diastolic blood pressure across different age groups

First we define font_dict for fonts and colors

```
title_font = {
    'family': 'Georgia',
    'color': '#003366',
    'weight': 'bold',
    'size': 18,
    'style': 'italic',
    'backgroundcolor': '#e6f2ff'
}

subtitle_font      =      {
'family': 'Arial',
    'color':          '#4a4a4a',
'weight': 'normal',
    'size': 16,
    'style': 'normal',
    'backgroundcolor': '#f2f2f2'
}

xlabel_font = {
    'family': 'Times New Roman',
    'color':          '#6e4c41',
'weight': 'normal',
    'size': 14,
    'style': 'italic',
    'backgroundcolor': '#f9f9f9'
```

```
}

ylabel_font = {
    'family': 'Verdana',
    'color':          '#d87a1a',
'weight': 'normal',
    'size': 14,
    'style': 'italic',
    'backgroundcolor': '#f9f9f9'
}

tick_font = {
    'family': 'DejaVu Sans',
    'color':          '#333333',
'weight': 'normal',
    'size': 12,
    'style': 'normal',
    'backgroundcolor': 'none'
}

label_colors = ['#003366', '#cc0000', '#4a4a4a', '#6a0dad', '#d87a1a',
'#6e4c41',      '#f08080',      '#90ee90']

box_colors = ['#add8e6', '#f08080']

# Define colors for each histogram
colors = ['skyblue', 'lightcoral', 'lightgreen', 'lightgoldenrodyellow',
'lightpink', 'lightsteelblue']
```

```python
# Define the age categories
age_categories = df['Age Category'].unique()

# Plotting
plt.figure(figsize=(14, 6))
plt.suptitle('Blood Pressure', **title_font)

# Define color map
box_colors = ['lightblue', 'lightgreen', 'lightcoral', 'lightsalmon', 'lightgoldenrodyellow']

# Systolic Blood Pressure Box Plot
plt.subplot(1, 2, 1)
data_to_plot = [df[df['Age Category'] == category]['Systolic'] for category in age_categories]
box = plt.boxplot(data_to_plot, labels=age_categories, patch_artist=True,
                  boxprops=dict(facecolor='lightblue', color='black'),
                  whiskerprops=dict(color='black'),
                  capprops=dict(color='black'),
                  medianprops=dict(color='red'))
for patch, color in zip(box['boxes'], box_colors):
    patch.set_facecolor(color)

# Customize x-axis labels' colors
for tick, color in zip(plt.gca().get_xticklabels(), label_colors):
    tick.set_color(color)


plt.title("Systolic Blood Pressure by Age Group", **subtitle_font)
plt.xlabel('Age Category', **xlabel_font)
plt.ylabel('Systolic Blood Pressure', **ylabel_font)
plt.grid(True, linestyle='--', alpha=0.7)

# Diastolic Blood Pressure Box Plot
plt.subplot(1, 2, 2)
data_to_plot = [df[df['Age Category'] == category]['Diastolic'] for category in age_categories]
box = plt.boxplot(data_to_plot, labels=age_categories, patch_artist=True,
                  boxprops=dict(facecolor='lightgreen', color='black'),
                  whiskerprops=dict(color='black'),
                  capprops=dict(color='black'),
                  medianprops=dict(color='red'))
for patch, color in zip(box['boxes'], box_colors):
    patch.set_facecolor(color)

plt.title("Diastolic Blood Pressure by Age Group", **subtitle_font)
plt.xlabel('Age Category', **xlabel_font)
plt.ylabel('Diastolic Blood Pressure', **ylabel_font)

# Customize the x-axis labels' colors
for tick, color in zip(plt.gca().get_xticklabels(), label_colors):
    tick.set_color(color)


plt.grid(True, linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```



Box Plots for Blood Pressure Readings

Box plots are a valuable tool for visualizing the distribution of blood pressure readings across different age groups. We use box polt to show the distribution of blood pressure readings across different age groups, revealing variations and outliers in the data.

---

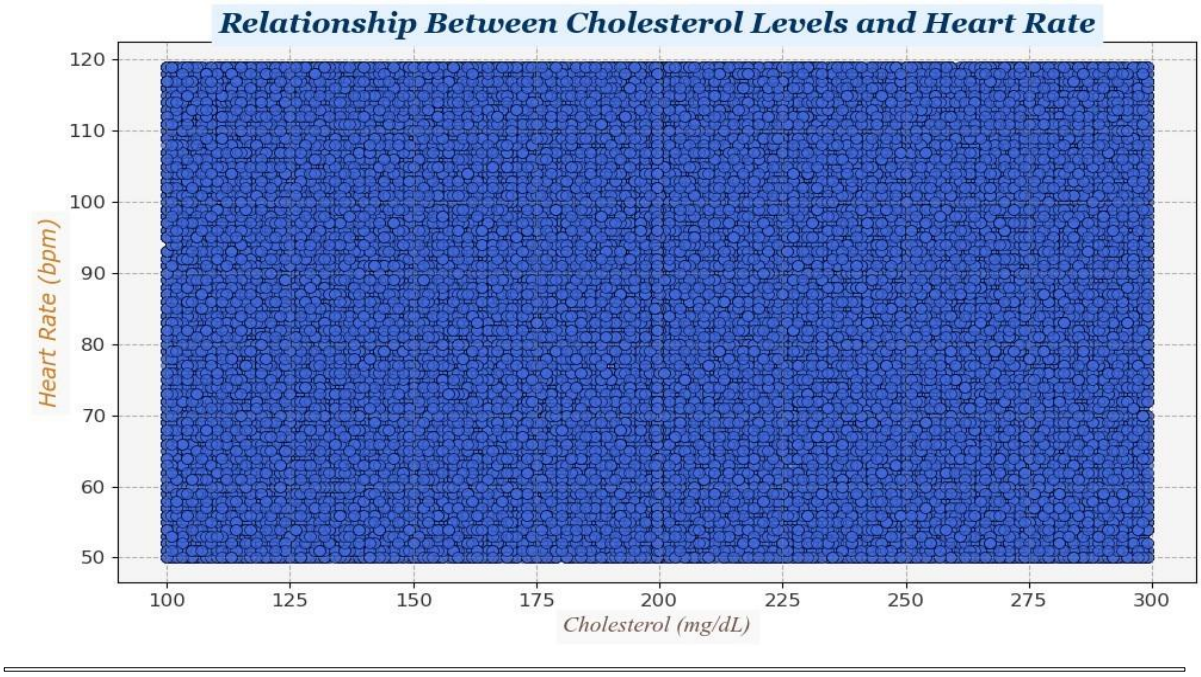
6. Cholesterol and Heart Rate Analysis:

Objective

- Create scatter plots to analyze the relationship between cholesterol levels and heart rate.

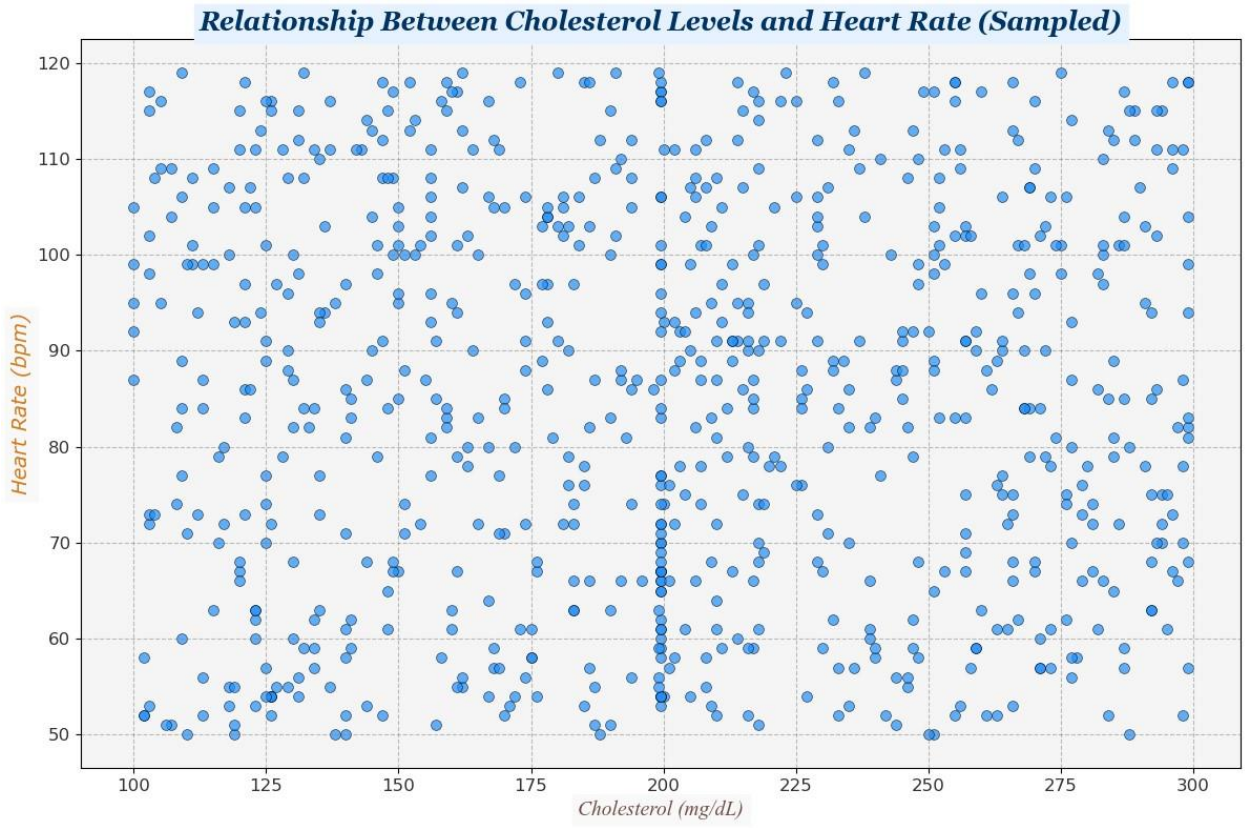
```
#         Create         scatter         plot
plt.figure(figsize=(10,6))

# Scatter plot for Cholesterol vs. Heart Rate
plt.scatter(df['Cholesterol'],  df['Heart  Rate'],  color='royalblue',
alpha=0.7, s=50, edgecolor='black', linewidth=0.5)

# Add titles and labels
plt.title('Relationship Between Cholesterol Levels and Heart Rate',
**title_font)
plt.xlabel('Cholesterol          (mg/dL)',          **xlabel_font)
plt.ylabel('Heart Rate (bpm)', **ylabel_font)

# Customize tick labels
plt.xticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])
plt.yticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])
```

```
# Show grid with customized style
plt.grid(True, linestyle='--', color='gray', alpha=0.6)

#     Add     a     background     color     to     the     plot     area
plt.gca().set_facecolor('whitesmoke')


# Display the plot plt.tight_layout()
plt.show()
```



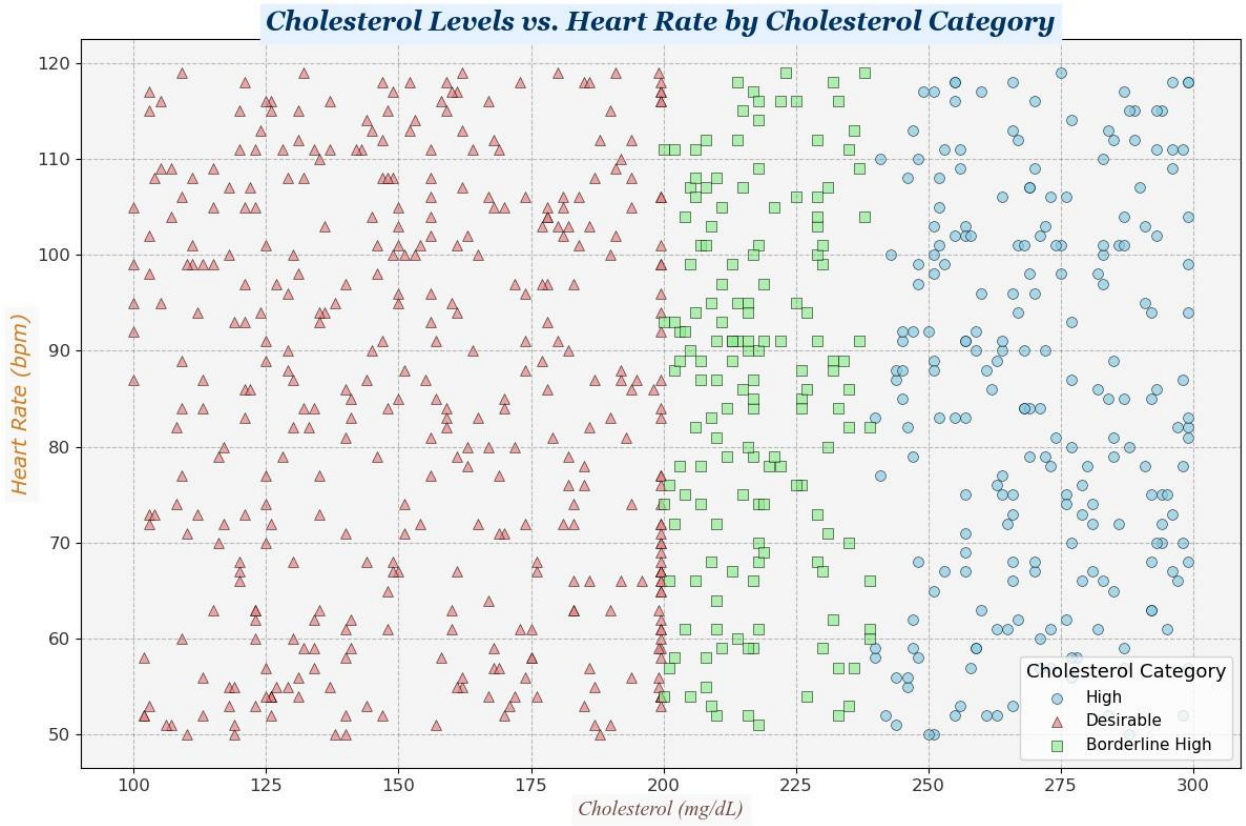
Sample 1% Data For Visualizing

```
# Sample 1% of the data for visualization
sampled_df = df.sample(frac=0.01, random_state=1) # 1% of data, frac is used for fraction

# Create scatter plot for sampled data
plt.figure(figsize=(12, 8))

# Scatter plot with improved style
plt.scatter(
    sampled_df['Cholesterol'],
    sampled_df['Heart Rate'],
    color='dodgerblue',  # A vibrant blue
    alpha=0.7,           # Transparency
    s=50,                # Larger marker size for better visibility
    edgecolor='black',   # Marker border color
    linewidth=0.5        # Border width
)

# Add titles and labels
plt.title('Relationship Between Cholesterol Levels and Heart Rate (Sampled)', **title_font)
plt.xlabel('Cholesterol (mg/dL)', **xlabel_font)
plt.ylabel('Heart Rate (bpm)', **ylabel_font)

# Customize tick labels
plt.xticks(fontsize=tick_font['size'], family=tick_font['family'], color=tick_font['color'])
plt.yticks(fontsize=tick_font['size'], family=tick_font['family'], color=tick_font['color'])

# Show grid with customized style
plt.grid(True, linestyle='--', color='gray', alpha=0.5)

# Add a background color to the plot area
plt.gca().set_facecolor('whitesmoke')

# Display the plot
plt.tight_layout()
plt.show()
```


Relationship Between Cholesterol Levels and Heart Rate (Sampled)
Heart Rate (bpm)
120
110
100
90
80
70
60
50
100
125
150
175
200
225
250
275
300
Cholesterol (mg/dL)

```
# Create scatter plot with categories

plt.figure(figsize=(12, 8))

markers = ['o', '^', 's', 'D', 'x']  # Circle, triangle, square, diamond,
x

categories = sampled_df['Cholesterol Category'].unique()

for category, color, marker in zip(categories, colors, markers):
    subset = sampled_df[sampled_df['Cholesterol Category'] ==
                    category]     plt.scatter(
     subset['Cholesterol'],
      subset['Heart Rate'],
         label=category,
    alpha=0.7,          s=50,
          color=color,
         marker=marker,
       edgecolor='black',
          linewidth=0.5
                                    )

plt.title('Cholesterol Levels vs. Heart Rate by Cholesterol Category',
**title_font)
```

```
plt.xlabel('Cholesterol          (mg/dL)',          **xlabel_font)
plt.ylabel('Heart Rate (bpm)', **ylabel_font)

# Customize tick labels
plt.xticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])
plt.yticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])

# Add legend
plt.legend(title='Cholesterol      Category',      title_fontsize='13',
fontsize='11')

# Show grid with customized style
plt.grid(True, linestyle='--', color='gray', alpha=0.5)

#     Add     a     background     color     to     the     plot     area
plt.gca().set_facecolor('whitesmoke')

# Display the plot plt.tight_layout()
plt.show()
```



Scatter Plots for Cholesterol Levels and Heart Rate

Scatter plots are an effective way to illustrate the relationship between cholesterol levels and heart rate. Given that our dataset contains over 70,000 rows, the initial visualization became too dense and challenging to interpret due to the high volume of data points.

To enhance clarity and make the visualization more insightful, I implemented the following strategy:

- **Sampling**: I randomly sampled 1% of the data to reduce clutter and improve the plot's readability. This approach allows us to focus on a manageable subset of the data, making patterns more discernible.
- **Visualization**: Using the sampled data, I created scatter plots with the following enhancements:
  - **Cholesterol vs. Heart Rate**: This plot demonstrates the relationship between cholesterol levels and heart rate. Markers have been styled with different colors and shapes for better distinction, and their size and edge coloring have been adjusted to improve visibility.
  - **Cholesterol Category vs. Heart Rate**: This plot further breaks down the relationship by categorizing cholesterol levels into distinct groups. Each category is represented with unique colors and markers, facilitating comparison across different cholesterol levels.

For Better Understanding Applied:

1. **Marker Customization**:
   - Used vibrant colors and distinct markers to differentiate between categories and make the plot visually engaging.
   - Increased marker size and added borders to enhance clarity and highlight indivih2lesterol categories.
2. **Grid and Background**:
   - Customized the grid lines to be dashed and in a soft gray color, ensuring they aid readability without distracting from the data.
   - Set a light background color to contrast with the markers, making the plot easier to interpret.

By applying these modifications, the scatter plots now provide a clearer visualization of the relationship between cholesterol levels and heart rate, allowing for better identification of trends and patterns without being overlapping data.

---

7. Gender-Based Health Comparisons:

Conclusions:

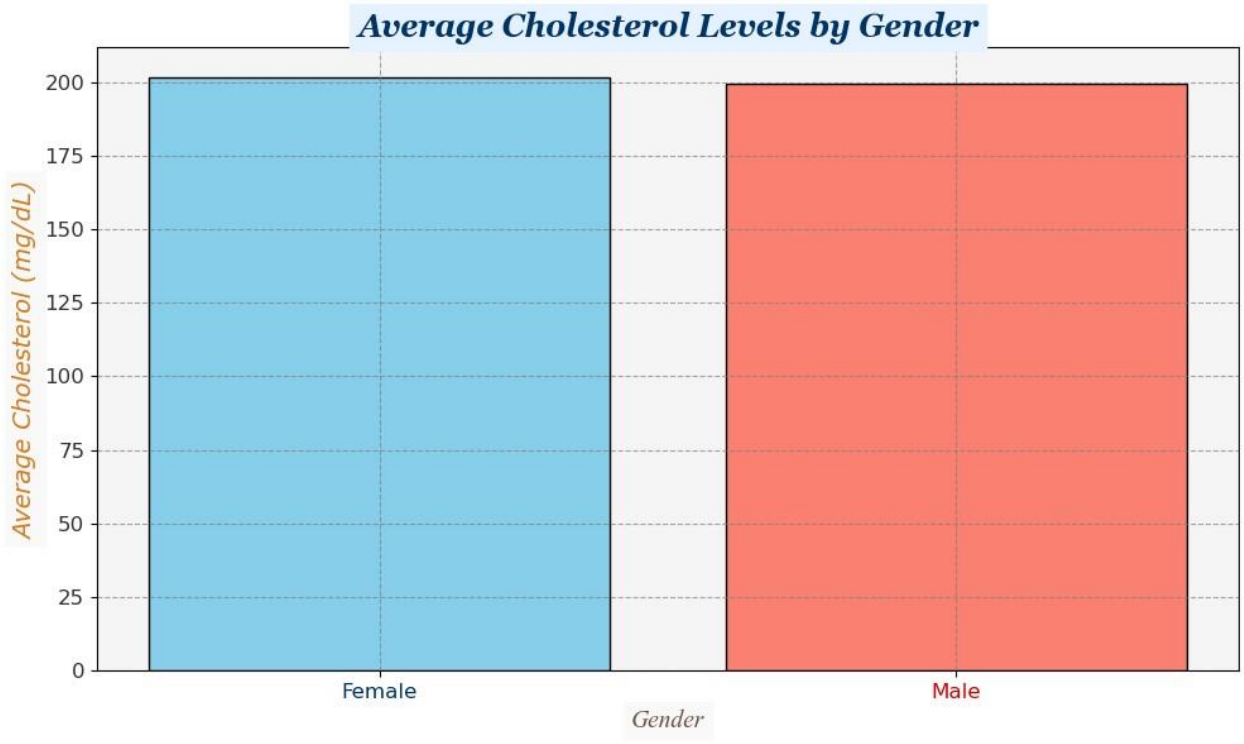
- Generate bar charts to compare average blood pressure, cholesterol, and BMI between male and female patients

---

```
# Calculate average metrics by gender
averages = sampled_df.groupby('Gender').agg({
    'Cholesterol': 'mean',
    'Body Mass Index (BMI)': 'mean',
    'Systolic':              'mean',
'Diastolic': 'mean'
}).reset_index()
```

Average Cholesterol

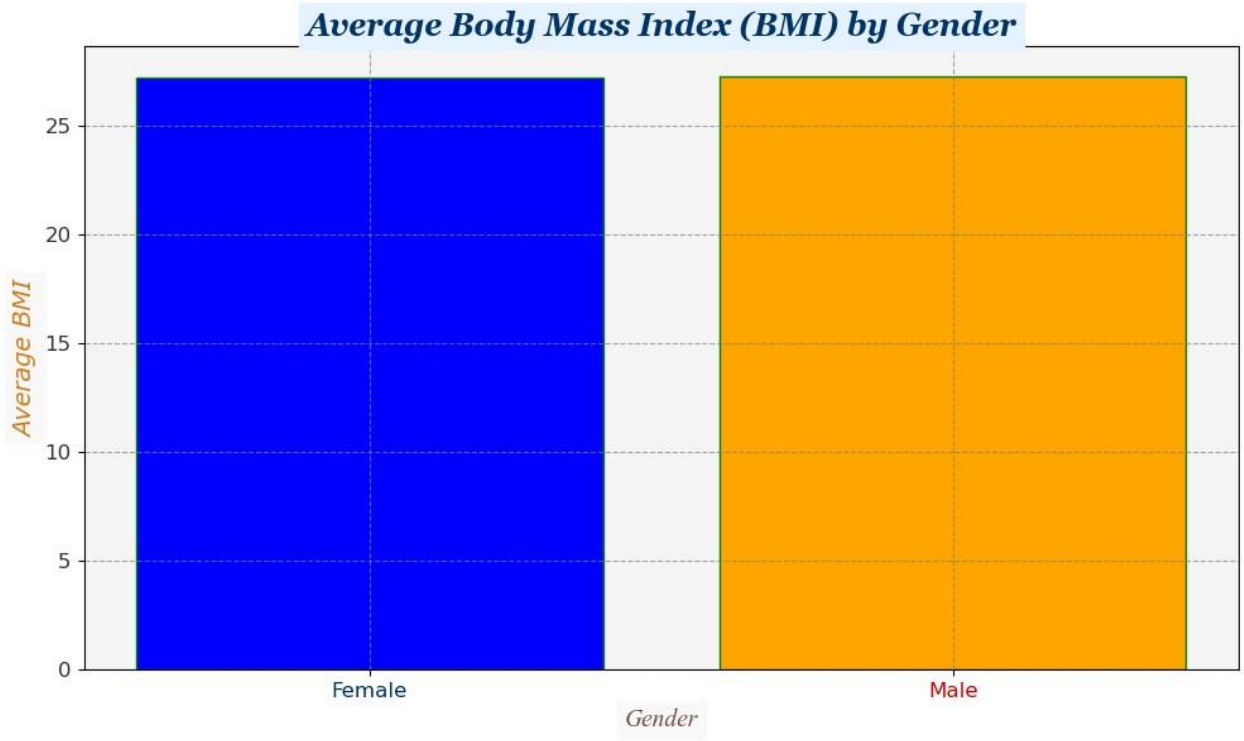
```
plt.figure(figsize=(10, 6))

# Bar plot with distinct colors for genders
plt.bar(
    averages['Gender'],
averages['Cholesterol'],     color=['skyblue',
'salmon'],
    edgecolor='black'              # Border color for bars
)

# Add titles and labels
plt.title('Average Cholesterol Levels by Gender', **title_font)
plt.xlabel('Gender', **xlabel_font)
plt.ylabel('Average Cholesterol (mg/dL)', **ylabel_font)

# Customize tick labels
plt.xticks(fontsize=tick_font['size'], family=tick_font['family'],
color=tick_font['color'])
plt.yticks(fontsize=tick_font['size'], family=tick_font['family'],
color=tick_font['color'])

# Customize the x-axis labels' colors for tick, color in
zip(plt.gca().get_xticklabels(),                    label_colors):
tick.set_color(color)



# Show grid with customized style
plt.grid(True, linestyle='--', color='gray', alpha=0.7)

# Add a background color to the plot area
plt.gca().set_facecolor('whitesmoke')

# Display the plot plt.tight_layout()
plt.show()
```



Average Body Mass Index

```
plt.figure(figsize=(10, 6))

# Bar plot with distinct colors for genders
plt.bar(
    averages['Gender'],
    averages['Body Mass Index (BMI)'],
color=['blue',              'orange'],
edgecolor='green'  )

# Add titles and labels
plt.title('Average Body Mass Index (BMI) by Gender', **title_font)
plt.xlabel('Gender',   **xlabel_font)   plt.ylabel('Average   BMI',
**ylabel_font)

# Customize tick labels
plt.xticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])
plt.yticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])

#  Customize  the  x-axis  labels'  colors  for  tick,  color  in
zip(plt.gca().get_xticklabels(),                    label_colors):
tick.set_color(color)
```

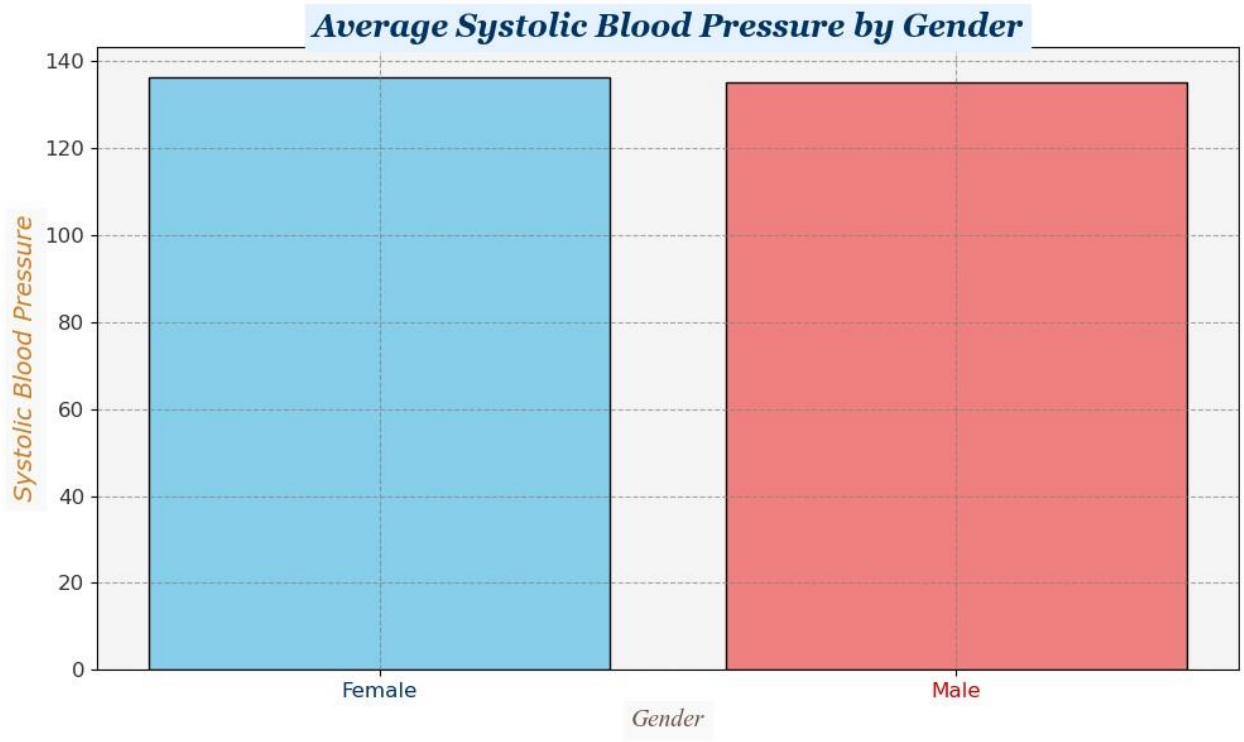
```
# Show grid with customized style
plt.grid(True, linestyle='--', color='gray', alpha=0.7)

# Add a background color to the plot area
plt.gca().set_facecolor('whitesmoke')


# Display the plot plt.tight_layout()
plt.show()
```



Average Systolic

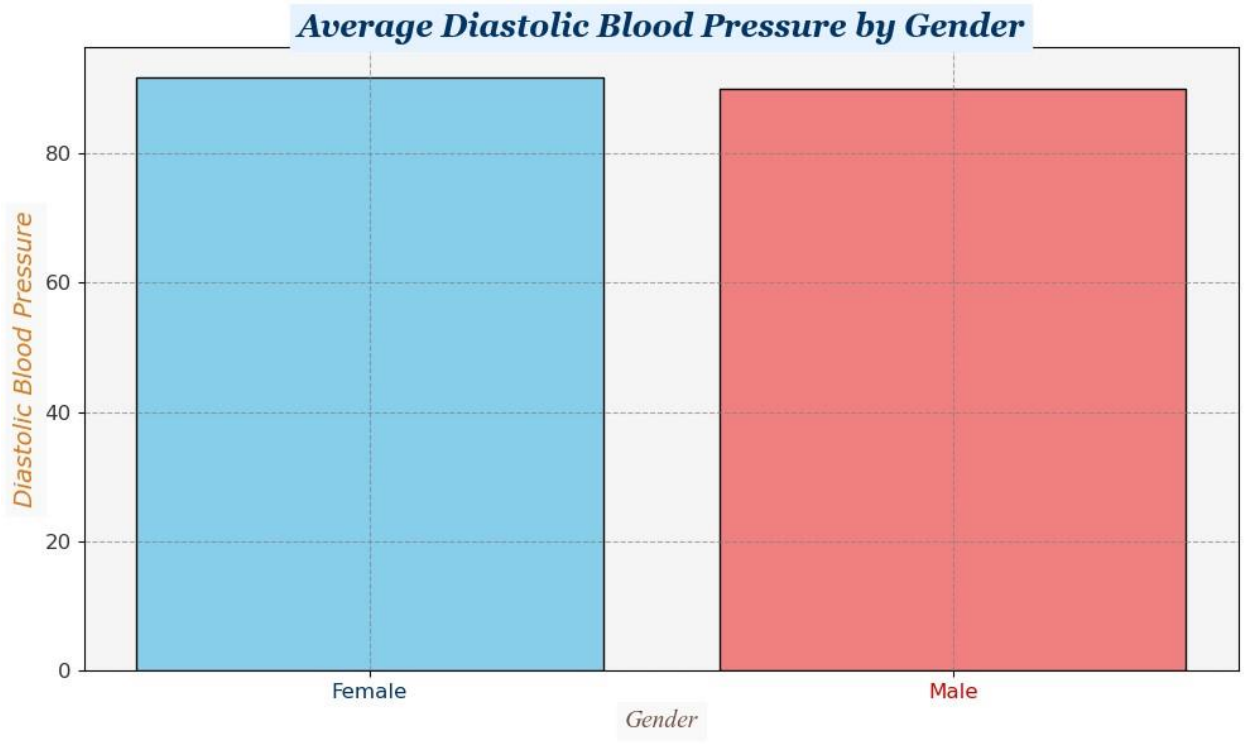
```
plt.figure(figsize=(10, 6))

# Bar plot with distinct colors for genders
plt.bar(
    averages['Gender'],
averages['Systolic'],
    color=['skyblue',                    'lightcoral'],
edgecolor='black'
)

# Add titles and labels
plt.title('Average Systolic Blood Pressure by Gender', **title_font)
plt.xlabel('Gender', **xlabel_font)
```

```
plt.ylabel('Systolic Blood Pressure', **ylabel_font)


# Customize tick labels
plt.xticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])
plt.yticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])

# Customize the x-axis labels' colors for tick, color in
zip(plt.gca().get_xticklabels(),                    label_colors):
tick.set_color(color)

# Show grid with customized style
plt.grid(True, linestyle='--', color='gray', alpha=0.7)


#    Add    a    background    color    to    the    plot    area
plt.gca().set_facecolor('whitesmoke')

# Display the plot plt.tight_layout()
plt.show()
```



Average Diastolic

```
plt.figure(figsize=(10, 6))

# Bar plot with distinct colors for genders
plt.bar(
    averages['Gender'],
averages['Diastolic'],
    color=['skyblue',                    'lightcoral'],
edgecolor='black'
)

# Add titles and labels
plt.title('Average Diastolic Blood Pressure by Gender', **title_font)
plt.xlabel('Gender', **xlabel_font)
plt.ylabel('Diastolic Blood Pressure', **ylabel_font)

# Customize tick labels
plt.xticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])
plt.yticks(fontsize=tick_font['size'],      family=tick_font['family'],
color=tick_font['color'])

#  Customize  the  x-axis  labels'  colors  for  tick,  color  in
zip(plt.gca().get_xticklabels(),                    label_colors):
tick.set_color(color)

# Show grid with customized style
plt.grid(True, linestyle='--', color='gray', alpha=0.7)


#    Add    a    background    color    to    the    plot    area
plt.gca().set_facecolor('whitesmoke')

# Display the plot plt.tight_layout()
plt.show()
```



Bar Charts for Comparing Health Metrics by Gender

Bar charts are employed to compare various health metrics between genders, providing insights into potential gender-based differences. Specifically, these charts are used to:

- **Compare Average Levels**: Bar charts illustrate the average cholesterol and BMI levels for different genders.
- **Analyze Blood Pressure**: They also compare average diastolic and systolic blood pressure readings between genders.

By visualizing these metrics, bar charts highlight potential health differences between genders, allowing for a clearer understanding of how these factors may vary.

---

8. Diabetes Impact on Health Metrics:

Objective:

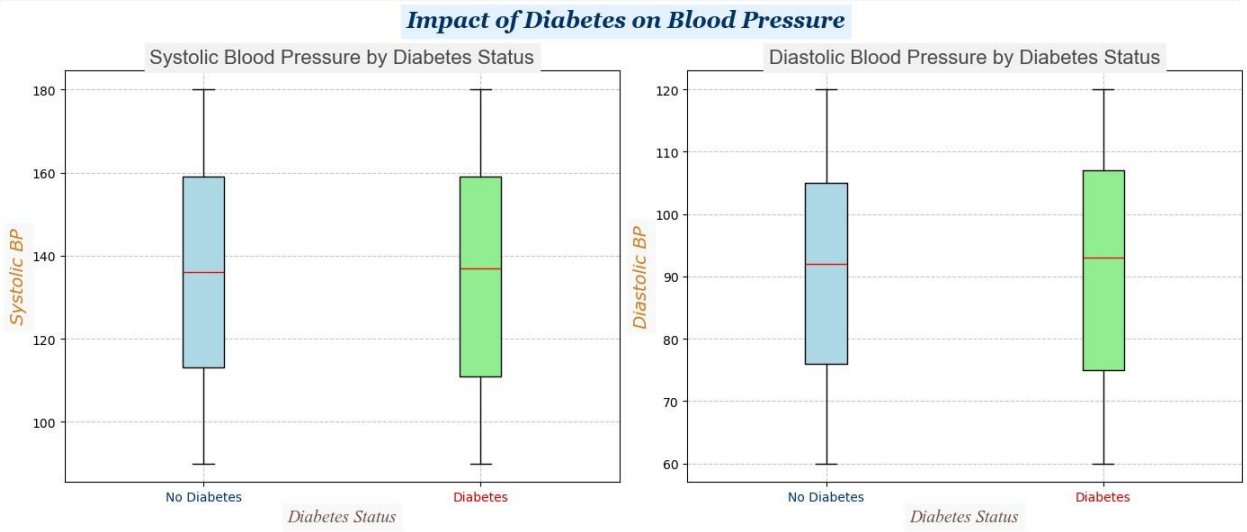
- Use box plots to compare health metrics (BP, Cholesterol, BMI) between diabetic and non-diabetic patients.

Plot For BP

```
sampled_df['Diabetes'] = sampled_df['Diabetes'].astype('category') ## Convert Diabetes into Category type b/c we only use .cat command with category
```

```
# Prepare the data for plotting
data_to_plot_systolic = [sampled_df[sampled_df['Diabetes'] ==
category]['Systolic'] for category in
sampled_df['Diabetes'].cat.categories]
data_to_plot_diastolic = [sampled_df[sampled_df['Diabetes'] ==
category]['Diastolic'] for category in
sampled_df['Diabetes'].cat.categories]

# Create a new figure for the box plots plt.figure(figsize=(14,
6))

# Sup Title
plt.suptitle('Impact of Diabetes on Blood Pressure', **title_font)

# Systolic Blood Pressure Box Plot plt.subplot(1,
2, 1)
box_systolic = plt.boxplot(data_to_plot_systolic, patch_artist=True,
boxprops=dict(facecolor='lightblue', color='black'),
whiskerprops=dict(color='black'),
capprops=dict(color='black'),
medianprops=dict(color='red'))

# Apply colors to the boxes for patch, color in
zip(box_systolic['boxes'], box_colors):
patch.set_facecolor(color)

# Set x-axis labels to 'No Diabetes' and 'Diabetes' labels
= ['No Diabetes', 'Diabetes'] plt.xticks(ticks=[1, 2],
labels=labels)

# Customize the x-axis labels' colors for tick, color in
zip(plt.gca().get_xticklabels(), label_colors):
tick.set_color(color)

plt.title('Systolic Blood Pressure by Diabetes Status',
**subtitle_font)
plt.xlabel('Diabetes Status', **xlabel_font)
plt.ylabel('Systolic BP', **ylabel_font) plt.grid(True,
linestyle='--', alpha=0.7)

# Diastolic Blood Pressure Box Plot plt.subplot(1,
2, 2)
box_diastolic = plt.boxplot(data_to_plot_diastolic, patch_artist=True,
boxprops=dict(facecolor='lightgreen', color='black'),
whiskerprops=dict(color='black'),
capprops=dict(color='black'),
medianprops=dict(color='red'))
```

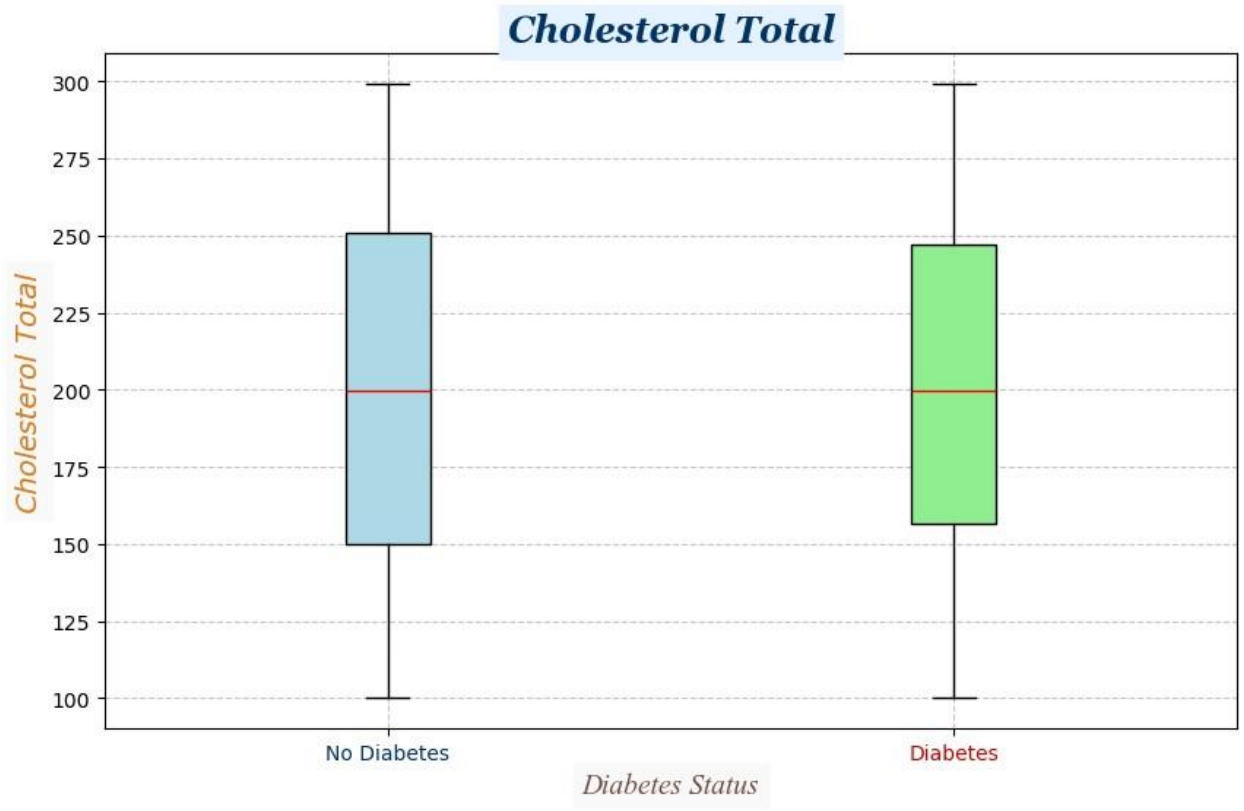
```
# Apply colors to the boxes for patch, color in
zip(box_diastolic['boxes'], box_colors):
patch.set_facecolor(color)

# Set x-axis labels to 'No Diabetes' and 'Diabetes' labels
= ['No Diabetes', 'Diabetes'] plt.xticks(ticks=[1, 2],
labels=labels)

# Customize the x-axis labels' colors for tick, color in
zip(plt.gca().get_xticklabels(), label_colors):
tick.set_color(color)

plt.title('Diastolic Blood Pressure by Diabetes Status',
**subtitle_font)
plt.xlabel('Diabetes Status', **xlabel_font)
plt.ylabel('Diastolic BP', **ylabel_font)
plt.grid(True, linestyle='--', alpha=0.7)


#                 layout
plt.tight_layout()
plt.show()
```



Plot for Cholesterol

```
# Prepare the data for plotting
data_to_plot_cholesterol = [sampled_df[sampled_df['Diabetes'] ==
category]['Cholesterol'] for category in
sampled_df['Diabetes'].cat.categories]

# Create a new figure plt.figure(figsize=(10,
6))
```

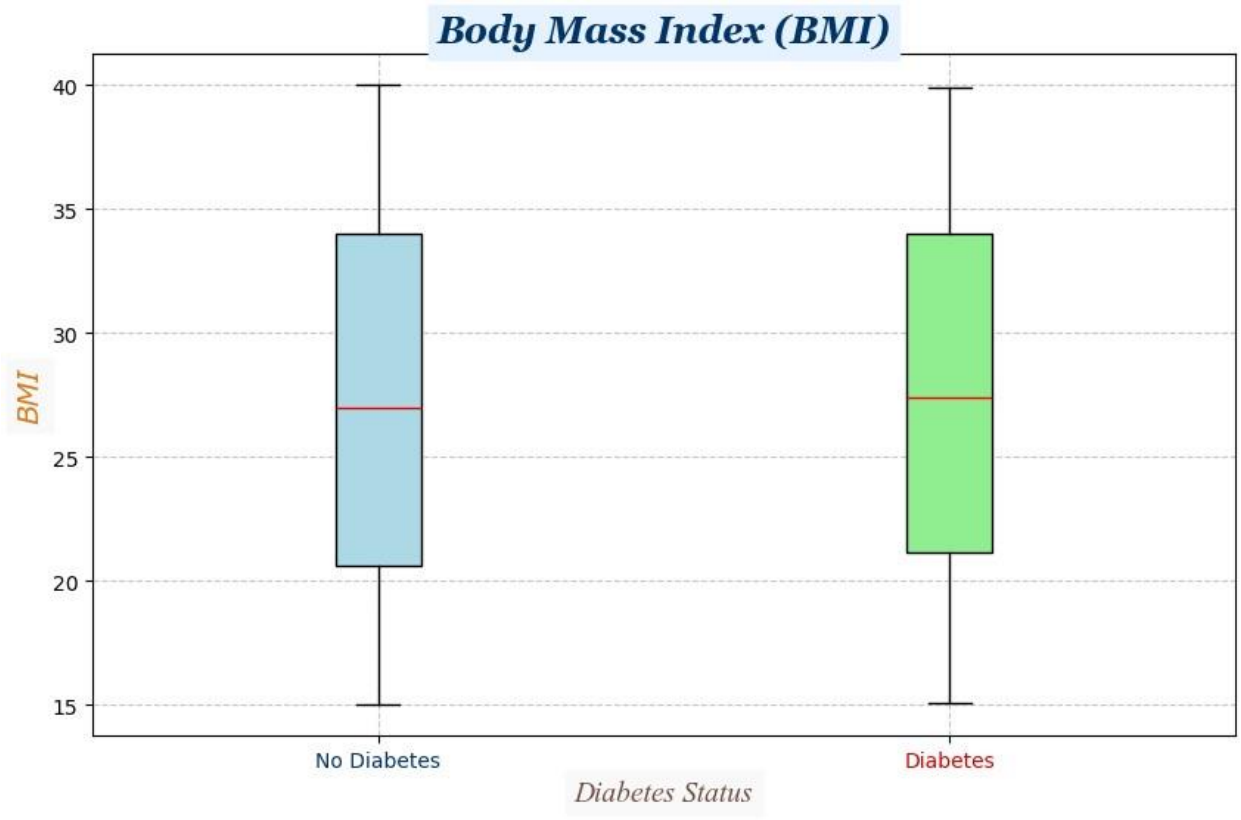
```
# Plot for Cholesterol
box_cholesterol          =          plt.boxplot(data_to_plot_cholesterol,
patch_artist=True,
                                boxprops=dict(facecolor='lightblue',
color='black'),
                                whiskerprops=dict(color='black'),
capprops=dict(color='black'),
medianprops=dict(color='red'))

#   Apply   colors   to   the   boxes   for   patch,   color   in
zip(box_cholesterol['boxes'],                       box_colors):
patch.set_facecolor(color)

# Set x-axis labels to 'No Diabetes' and 'Diabetes' labels
= ['No Diabetes', 'Diabetes'] plt.xticks(ticks=[1, 2],
labels=labels)

#  Customize  the  x-axis  labels'  colors  for  tick,  color  in
zip(plt.gca().get_xticklabels(),                      label_colors):
tick.set_color(color)

# Add titles and labels
plt.title('Cholesterol Total', **title_font)
plt.suptitle('')  # Suppress the default title to make the plot cleaner
plt.xlabel('Diabetes              Status',              **xlabel_font)
plt.ylabel('Cholesterol Total', **ylabel_font)

# Customize the grid
plt.grid(True, linestyle='--', alpha=0.7)

# Display the plot
plt.show()
```


Cholesterol Total
300
275
250
225
200
175
150
125
100
Cholesterol Total
No Diabetes
Diabetes
Diabetes Status

Plot For BMI

```
# Prepare the data for plotting
data_to_plot_bmi = [sampled_df[sampled_df['Diabetes'] == category]
['Body Mass Index (BMI)'] for category in
sampled_df['Diabetes'].cat.categories]

# Create a new figure plt.figure(figsize=(10,
6))

# Plot for Body Mass Index (BMI)
box_bmi = plt.boxplot(data_to_plot_bmi, patch_artist=True,
boxprops=dict(facecolor='lightblue', color='black'),
                      whiskerprops=dict(color='black'),
capprops=dict(color='black'),
medianprops=dict(color='red'))

# Apply colors to the boxes for patch, color in
zip(box_bmi['boxes'], box_colors):
patch.set_facecolor(color)

# Set x-axis labels to 'No Diabetes' and 'Diabetes' labels
= ['No Diabetes', 'Diabetes'] plt.xticks(ticks=[1, 2],
labels=labels)
```

```
# Customize the x-axis labels' colors for tick, color in
zip(plt.gca().get_xticklabels(), label_colors):
tick.set_color(color)

# Add titles and labels
plt.title('Body Mass Index (BMI)', **title_font)
plt.suptitle('')  # Suppress the default title to make the plot cleaner
plt.xlabel('Diabetes Status', **xlabel_font)
plt.ylabel('BMI', **ylabel_font)

# Customize the grid
plt.grid(True, linestyle='--', alpha=0.7)


# Display the plot
plt.show()
```



Box Plots for Analyzing the Impact of Diabetes on Health Metrics

Box plots are used to reveal how diabetes affects various health metrics by comparing their distributions between diabetic and non-diabetic patients. Specifically, these plots are used to:

- **Compare Blood Pressure**: Box plots show the distribution of blood pressure readings for diabetic versus non-diabetic patients.
- **Analyze Cholesterol Levels**: They illustrate how cholesterol levels differ between the two groups.
- **Evaluate BMI**: Box plots also display variations in BMI between diabetic and non-diabetic individuals.

By visualizing these distributions, box plots help in understanding how diabetes may influence these key health metrics.

---

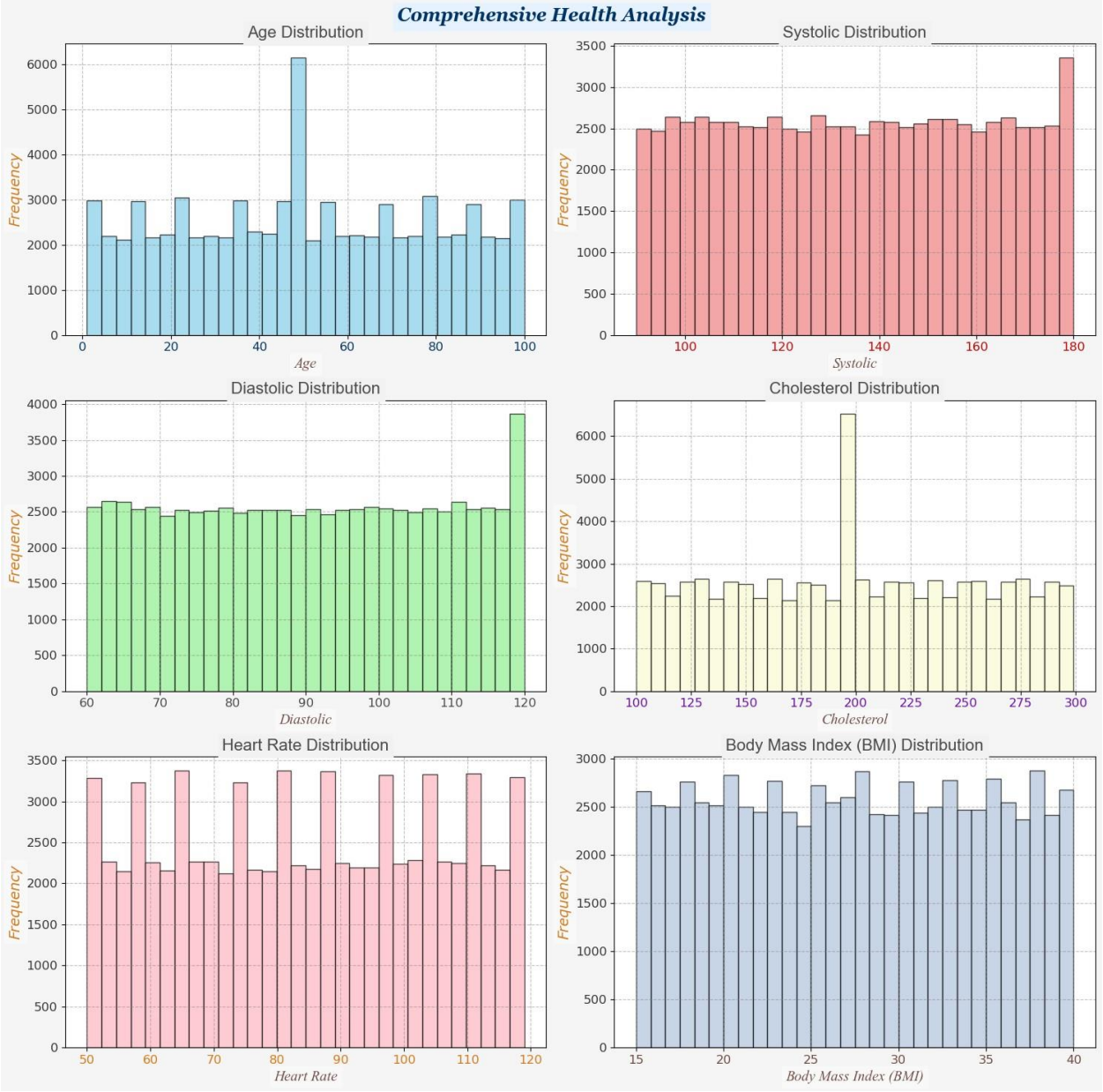
9. Subplots for Comprehensive Health Analysis:

Objective

- Create a figure with multiple subplots to provide an overview of key health indicators

---

```
# Define the key health indicators
columns = ['Age', 'Systolic', 'Diastolic', 'Cholesterol', 'Heart Rate',
'Body Mass Index (BMI)']

# Create the figure and axes
fig, axes = plt.subplots(3, 2, figsize=(15, 15))

# Flatten axes array for easy iteration axes
= axes.flatten()


# Super Title
plt.suptitle("Comprehensive Health Analysis", **title_font)

# Plot histograms for each health indicator with different colors for
i, column in enumerate(columns):    axes[i].hist(df[column].dropna(),
bins=30, color=colors[i], edgecolor='black', alpha=0.7)
    axes[i].set_title(f'{column}    Distribution',    **subtitle_font)
axes[i].set_xlabel(column,                                **xlabel_font)
axes[i].set_ylabel('Frequency', **ylabel_font)
    axes[i].grid(True,   linestyle='--',   color='gray',   alpha=0.5)
axes[i].tick_params(axis='both',          labelsize=tick_font['size'],
labelcolor=tick_font['color'])
    # Customize the x-axis labels' colors
    for tick, color in zip(axes[i].get_xticklabels(), [label_colors[i
%      len(label_colors)]]      *      len(axes[i].get_xticklabels())):
tick.set_color(color)

# Remove empty subplot (if the number of columns is less than 6)
if len(columns) < 6:     fig.delaxes(axes[len(columns)])

#          Adjust          layout
plt.tight_layout()



# Set a light background color for the entire
figure fig.patch.set_facecolor('whitesmoke')

# Display the plot plt.show()
```



Explanation:

Subplots provide a consolidated view of various health indicators, facilitating a comprehensive analysis of the data.

---

10. Conclusion

The analysis reveals key trends and patterns in patient health data:

- Blood pressure trends show variations across different age groups.
- Cholesterol levels and heart rate exhibit correlations that may inform cardiovascular health.
- Gender-based comparisons highlight differences in average cholesterol and BMI levels.
- Diabetes status significantly impacts various health metrics, suggesting targeted treatment approaches.

These insights are valuable for improving patient care and developing effective treatment strategies.
