```
import pymysql
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Connect to MySQL using PyMySQL
db_connection = pymysql.connect(
    host="127.0.0.1",
    user="root",
    password="AnIsH@123#",  # Replace with your MySQL password
    database="healthcare"
)

cursor = db_connection.cursor()
```

```
# Fetch patient details and health metrics
query = """
SELECT p.name, p.age, p.gender, p.weight, p.height, h.date_of_record, h.blood_pressure, h.heart_rate, h.cholesterol
FROM patients p
JOIN health_metrics h ON p.patient_id = h.patient_id;
"""
cursor.execute(query)
data = cursor.fetchall()

print(data)
```

```
(('John Doe', 45, 'Male', Decimal('78.50'), Decimal('175.00'), datetime.date(2024, 10, 1), '130/85', 78, Decimal('5.60')), ('Jane Sm
```

```
# Convert the fetched data into a Pandas DataFrame
df = pd.DataFrame(data, columns=['name', 'age', 'gender', 'weight', 'height', 'date_of_record', 'blood_pressure', 'heart_rate', 'cholest
print(df)
# Close the database connection
cursor.close()
db_connection.close()

# Data Preprocessing and Visualization (Assumed done before)
```

```
    16   Patient 17   58    Male  76.91  172.78     2024-10-14         130/85
    17   Patient 18   45    Male  69.68  153.37     2024-10-15         120/78
    18   Patient 19   55  Female  88.85  152.01     2024-10-16         140/88
    19   Patient 20   35  Female  90.67  175.96     2024-10-17         115/75
    20   Patient 21   30  Female  83.85  161.29     2024-10-18         130/84
    21   Patient 22   32    Male  76.01  179.83     2024-10-19         135/89
    22   Patient 23   53  Female  61.12  166.11     2024-10-20         120/80
    23   Patient 24   48    Male  67.05  170.78     2024-10-21         145/93
    24   Patient 25   58  Female  66.73  151.74     2024-10-22         130/87
    25   Patient 26   53  Female  76.35  173.54     2024-10-23         120/76
    26   Patient 27   50    Male  82.88  163.85     2024-10-24         140/90
    27   Patient 28   47  Female  64.49  174.82     2024-10-25         110/70
    28   Patient 29   53  Female  67.12  160.33     2024-10-26         150/95
    29   Patient 30   35  Female  98.92  154.29     2024-10-27         125/80
    30   Patient 31   53  Female  73.88  152.89     2024-10-28         135/88
    31   Patient 32   43  Female  69.27  162.85     2024-10-29         140/89
    32   Patient 33   43  Female  75.60  153.38     2024-10-30         120/77
    33   Patient 34   41  Female  99.16  179.93     2024-10-31         145/92
    34   Patient 35   31    Male  66.91  152.02     2024-11-01         130/85
    35   Patient 36   54  Female  99.93  159.56     2024-11-02         150/96

        heart_rate cholesterol
    0           78        5.60
```

```
18          79        6.00
19          65        5.40
20          78        6.30
21          80        5.80
22          70        5.20
23          84        6.40
24          77        6.10
25          73        5.60
26          81        6.50
27          67        4.70
28          85        6.90
29          74        5.90
30          79        6.30
31          81        6.60
32          70        5.50
33          82        6.80
34          75        5.40
35          88        7.10
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   name            36 non-null     object
 1   age             36 non-null     int64 
 2   gender          36 non-null     object
 3   weight          36 non-null     object
 4   height          36 non-null     object
 5   date_of_record  36 non-null     object
 6   blood_pressure  36 non-null     object
 7   heart_rate      36 non-null     int64 
 8   cholesterol     36 non-null     object
dtypes: int64(2), object(7)
memory usage: 2.7+ KB
```

```
df['date_of_record'] = pd.to_datetime(df['date_of_record'])
df.info()
```

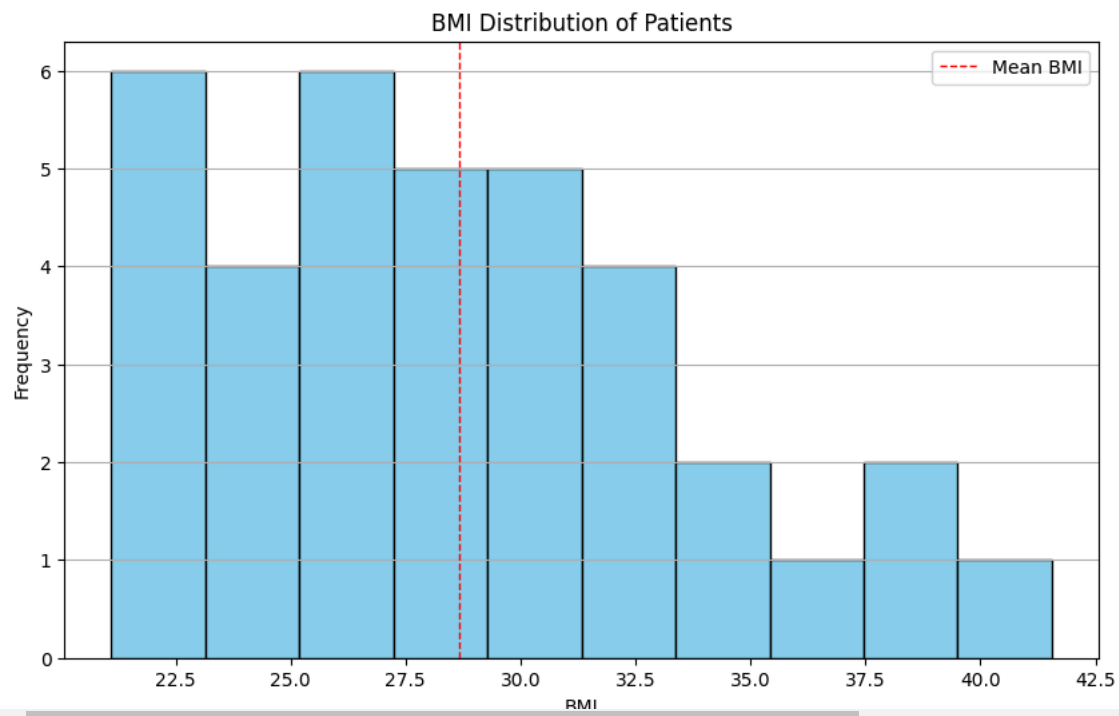
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   name            36 non-null     object
 1   age             36 non-null     int64 
 2   gender          36 non-null     object
 3   weight          36 non-null     object
 4   height          36 non-null     object
 5   date_of_record  36 non-null     datetime64[ns]
 6   blood_pressure  36 non-null     object
 7   heart_rate      36 non-null     int64 
 8   cholesterol     36 non-null     object
dtypes: datetime64[ns](1), int64(2), object(6)
memory usage: 2.7+ KB
```

```
# Convert relevant columns to float
df['weight'] = df['weight'].astype(float)
df['height'] = df['height'].astype(float)
df['cholesterol'] = df['cholesterol'].astype(float)

# Calculate BMI for each patient
df['bmi'] = df['weight'] / ((df['height'] / 100) ** 2)
print(df.bmi)
```

```
0     25.632653
1     23.948577
2     28.242187
3     37.043735
4     29.081080
5     21.851449
6     24.809263
7     29.208367
8     32.491885
9     22.909343
10    22.935522
11    29.403128
12    24.519338
13    25.738354
14    33.645813
15    33.616488
16    25.762967
17    29.622882
18    38.451485
```

```
19    29.284358
20    32.232069
21    23.504252
22    22.150921
23    22.989248
24    28.981507
25    25.351862
26    30.871437
27    21.101345
28    26.110931
29    41.553590
30    31.605941
31    26.119789
32    32.135446
33    30.628756
34    28.952734
35    39.250738
Name: bmi, dtype: float64
```

```
# Plotting the BMI distribution
plt.figure(figsize=(10, 6))
plt.hist(df['bmi'], bins=10, color='skyblue', edgecolor='black')
plt.title('BMI Distribution of Patients')
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.grid(axis='y')
plt.axvline(df['bmi'].mean(), color='red', linestyle='dashed', linewidth=1, label='Mean BMI')
plt.legend()
plt.show()
```

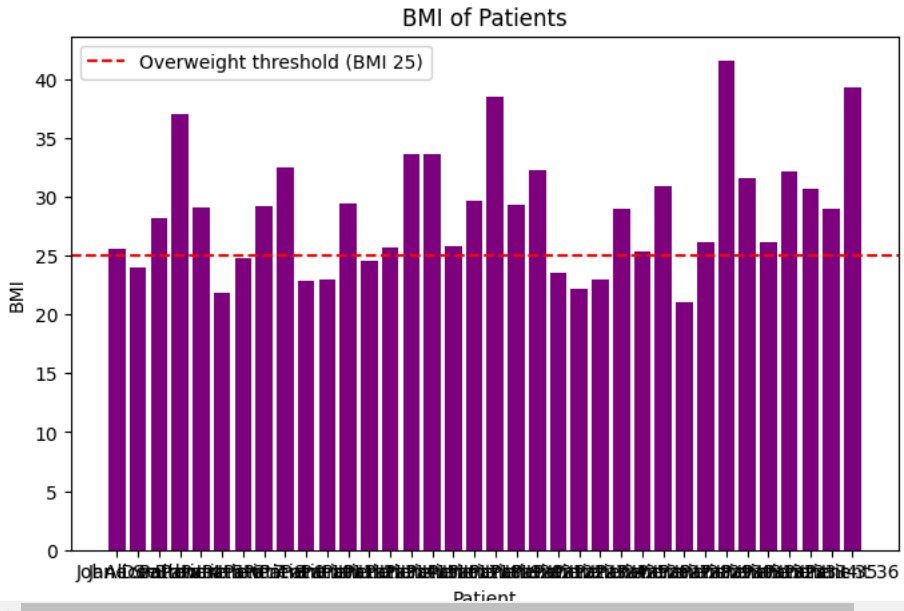



```
# Calculate BMI
df['bmi'] = df['weight'] / ((df['height'] / 100) ** 2)

# Print and plot BMI
print("BMI for each patient:\n", df[['name', 'weight', 'height', 'bmi']])
```

```
BMI for each patient:
             name weight  height                          bmi
0      John Doe  78.50  175.00  25.632653061224489795918367735
1    Jane Smith  65.20  165.00  23.948576675849403122130394867
2   Alice Brown  72.30  160.00                    28.2421875
3     Patient 4  84.05  150.63  37.043734732323361137906666888
4     Patient 5  72.92  158.35  29.081079561299232723313288911
5     Patient 6  64.36  171.62  21.851448960888907708809573944
6     Patient 7  69.93  167.89  24.809263399346397064345724890
7     Patient 8  71.20  156.13  29.208366823645365892206136620
8     Patient 9  97.20  172.96  32.491885318780962226191016130
9    Patient 10  72.79  178.25  22.909343410003363698415914420
10   Patient 11  73.94  179.55  22.935521916893474442317309720
11   Patient 12  83.73  168.75  29.403127572016460905349794240
12   Patient 13  69.36  168.19  24.519338162683765431226559990
13   Patient 14  67.04  161.39  25.738353564898838115665189340
14   Patient 15  77.49  151.76  33.645812868249926568849241470
```

```
15   Patient 16  89.02  162.73  33.616488111769906248705088 31
16   Patient 17  76.91  172.78  25.762966691066729654568829 50
17   Patient 18  69.68  153.37  29.622881880514277886838797 18
18   Patient 19  88.85  152.01  38.451484749013786495311444 07
19   Patient 20  90.67  175.96  29.284358112966735242944352 65
20   Patient 21  83.85  161.29  32.232068928146784311424658 56
21   Patient 22  76.01  179.83  23.504252499384725709717954 57
22   Patient 23  61.12  166.11  22.150921045770931621023646 47
23   Patient 24  67.05  170.78  22.989247916748983374655920 73
24   Patient 25  66.73  151.74  28.981507062341154370646661 03
25   Patient 26  76.35  173.54  25.351861591679324445507470 16
26   Patient 27  82.88  163.85  30.871437392637434094854242 06
27   Patient 28  64.49  174.82  21.101345341156041703561573 35
28   Patient 29  67.12  160.33  26.110931334789992895313616 50
29   Patient 30  98.92  154.29  41.553589926247040687706704 38
30   Patient 31  73.88  152.89  31.605941028798449628487093 46
31   Patient 32  69.27  162.85  26.119788850899915714944378 56
32   Patient 33  75.60  153.38  32.135445769046359903288290 94
33   Patient 34  99.16  179.93  30.628756005260127451764195 81
34   Patient 35  66.91  152.02  28.952733543930033233462917 77
35   Patient 36  99.93  159.56  39.250738477918814217161873 01
```

```
plt.figure(figsize=(8, 5))
plt.bar(df['name'], df['bmi'], color='purple')
plt.title('BMI of Patients')
plt.xlabel('Patient')
plt.ylabel('BMI')
plt.axhline(y=25, color='r', linestyle='--', label='Overweight threshold (BMI 25)')
plt.legend()
plt.show()
```

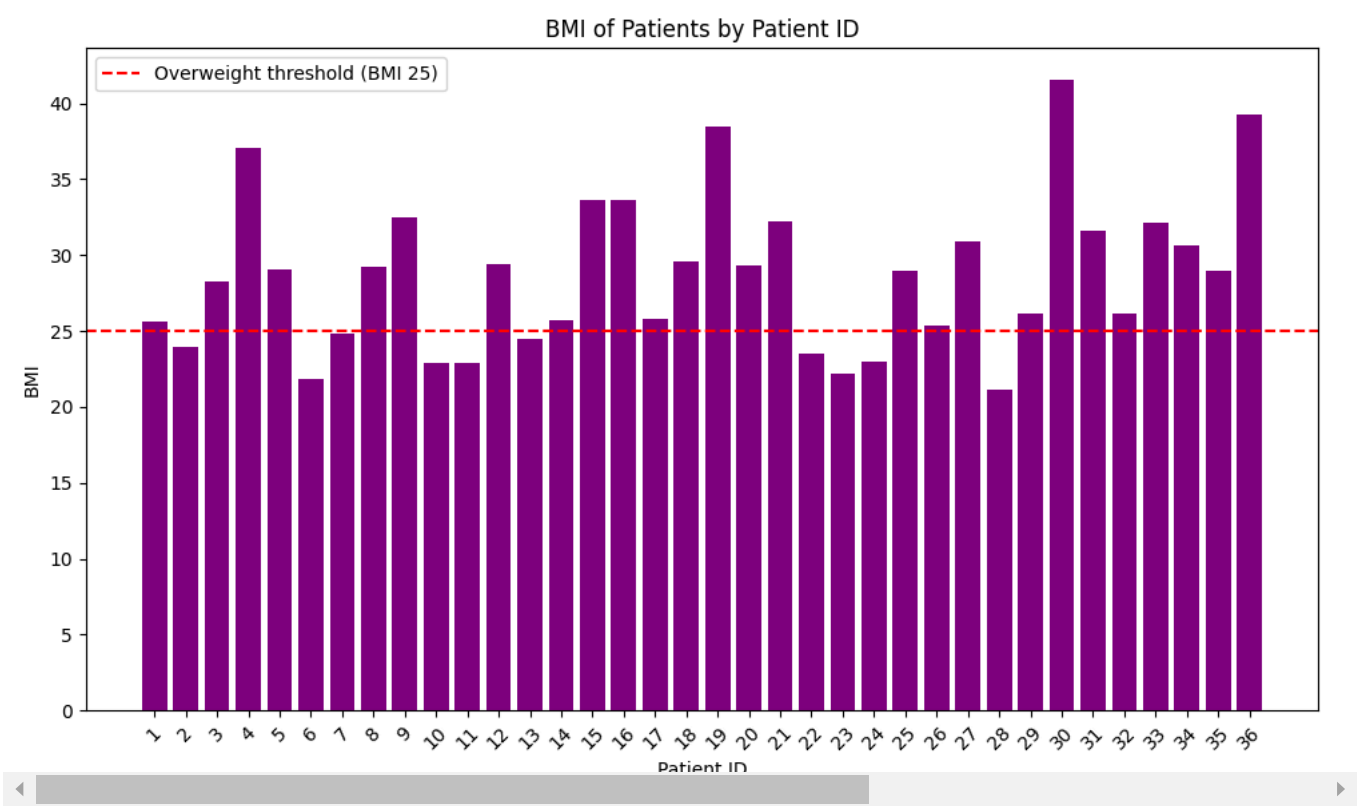



```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Connect to the MySQL database
db_connection = pymysql.connect(
    host="127.0.0.1",
    user="root",
    password="AnIsH@123#",  # Replace with your MySQL password
    database="healthcare"
)

cursor = db_connection.cursor()

# Step 2: Fetch data from MySQL
query = """
SELECT p.patient_id, p.name, p.weight, p.height
FROM patients p;
"""
cursor.execute(query)
data = cursor.fetchall()

# Check the data fetched
print("Data Fetched from Database:\n", data)
```

```
# Step 3: Convert the data into a Pandas DataFrame
# Ensure that we only create a DataFrame with the columns we are fetching
df = pd.DataFrame(data, columns=['patient_id', 'name', 'weight', 'height'])
cursor.close()
db_connection.close()
# Check if the DataFrame has the expected columns
print("DataFrame Columns: ", df.columns)

# Calculate BMI for each patient
df['bmi'] = df['weight'] / ((df['height'] / 100) ** 2)

# Print BMI for each patient
print("BMI for each patient:\n", df[['patient_id', 'name', 'weight', 'height', 'bmi']])
```

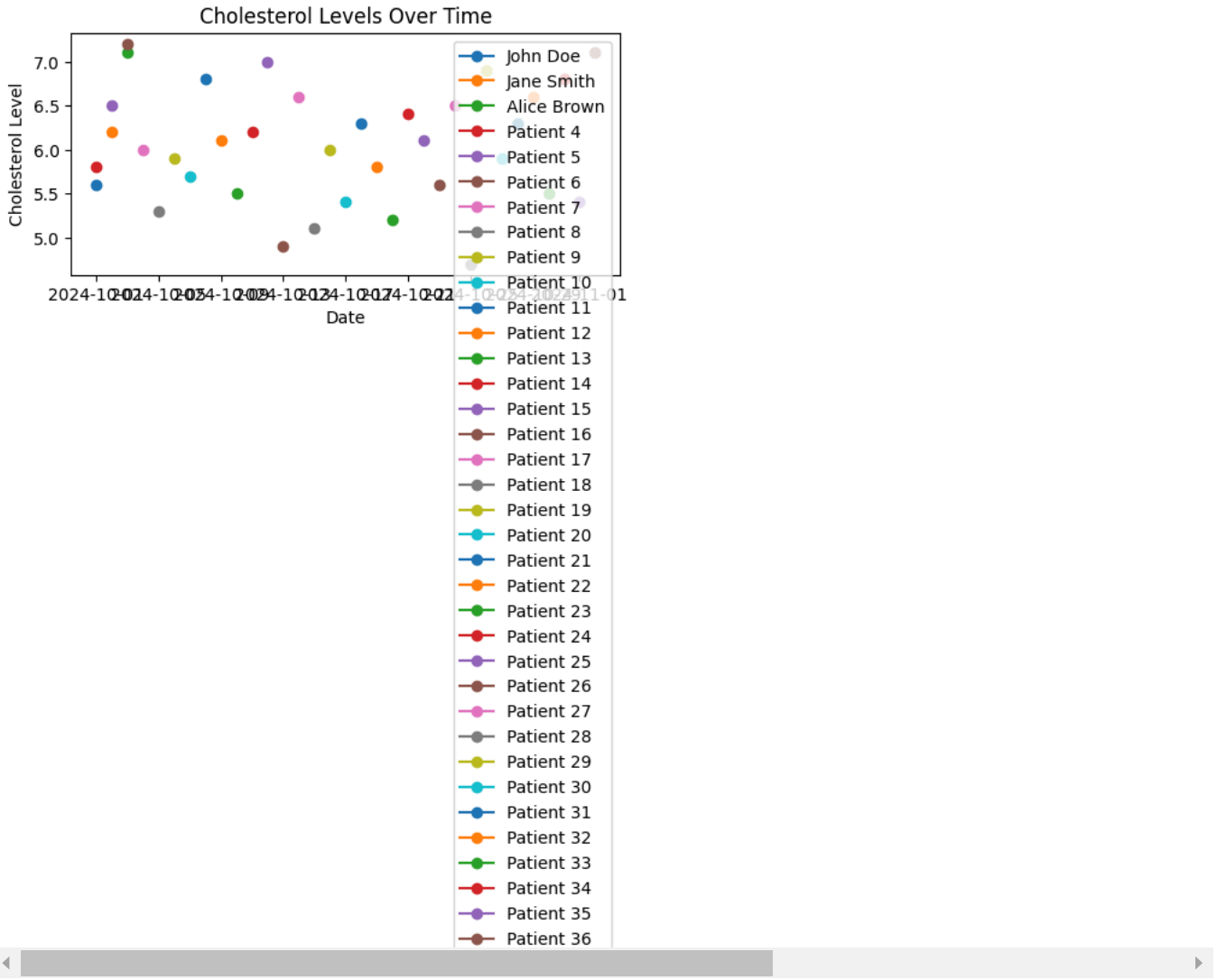
```
Data Fetched from Database:
 ((1, 'John Doe', Decimal('78.50'), Decimal('175.00')), (2, 'Jane Smith', Decimal('65.20'), Decimal('165.00')), (3, 'Alice Brown', D
DataFrame Columns:  Index(['patient_id', 'name', 'weight', 'height'], dtype='object')
BMI for each patient:
     patient_id         name weight  height                           bmi
0             1     John Doe  78.50  175.00  25.632653061224489795918367
1             2   Jane Smith  65.20  165.00  23.948576675849403122130394
2             3  Alice Brown  72.30  160.00                     28.2421875
3             4    Patient 4  84.05  150.63  37.043734732323361137906666688
4             5    Patient 5  72.92  158.35  29.081079561299232723313288 91
5             6    Patient 6  64.36  171.62  21.851448960888907708809573 94
6             7    Patient 7  69.93  167.89  24.809263399346397064345724 89
7             8    Patient 8  71.20  156.13  29.208366823645365892206136 62
8             9    Patient 9  97.20  172.96  32.491885318780962226191016 13
9            10   Patient 10  72.79  178.25  22.909343410003363698415914 42
10           11   Patient 11  73.94  179.55  22.935521916893474442317309 72
11           12   Patient 12  83.73  168.75  29.403127572016460905349794 24
12           13   Patient 13  69.36  168.19  24.519338162683765431226559 99
13           14   Patient 14  67.04  161.39  25.738353564898838115665189 34
14           15   Patient 15  77.49  151.76  33.645812868249926568849241 47
15           16   Patient 16  89.02  162.73  33.616488111769906248705088 31
16           17   Patient 17  76.91  172.78  25.762966691066729654568829 50
17           18   Patient 18  69.68  153.37  29.622881880514277886838797 18
18           19   Patient 19  88.85  152.01  38.451484749013786495311444 07
19           20   Patient 20  90.67  175.96  29.284358112966735242944352 65
20           21   Patient 21  83.85  161.29  32.232068928146784311424658 56
21           22   Patient 22  76.01  179.83  23.504252499384725709717954 57
22           23   Patient 23  61.12  166.11  22.150921045770931621023646 47
23           24   Patient 24  67.05  170.78  22.989247916748983374655920 73
24           25   Patient 25  66.73  151.74  28.981507062341154370646661 03
25           26   Patient 26  76.35  173.54  25.351861591679324445507470 16
26           27   Patient 27  82.88  163.85  30.871437392637434094854242 06
27           28   Patient 28  64.49  174.82  21.101345341156041703561573 35
28           29   Patient 29  67.12  160.33  26.110931334789992895313616 50
29           30   Patient 30  98.92  154.29  41.553589926247040687706704 38
30           31   Patient 31  73.88  152.89  31.605941028798449628487093 46
31           32   Patient 32  69.27  162.85  26.119788850899915714944378 56
32           33   Patient 33  75.60  153.38  32.135445769046359903288290 94
33           34   Patient 34  99.16  179.93  30.628756005260127451764195 81
34           35   Patient 35  66.91  152.02  28.952733543930033233462917 77
35           36   Patient 36  99.93  159.56  39.250738477918814217161873 01
```

```
# Plotting BMI with patient_id on the x-axis
plt.figure(figsize=(10, 6))
plt.bar(df['patient_id'].astype(str), df['bmi'], color='purple')  # Convert patient_id to string for plotting
plt.title('BMI of Patients by Patient ID')
plt.xlabel('Patient ID')
plt.ylabel('BMI')
plt.axhline(y=25, color='r', linestyle='--', label='Overweight threshold (BMI 25)')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.legend()
plt.tight_layout()
plt.show()
```



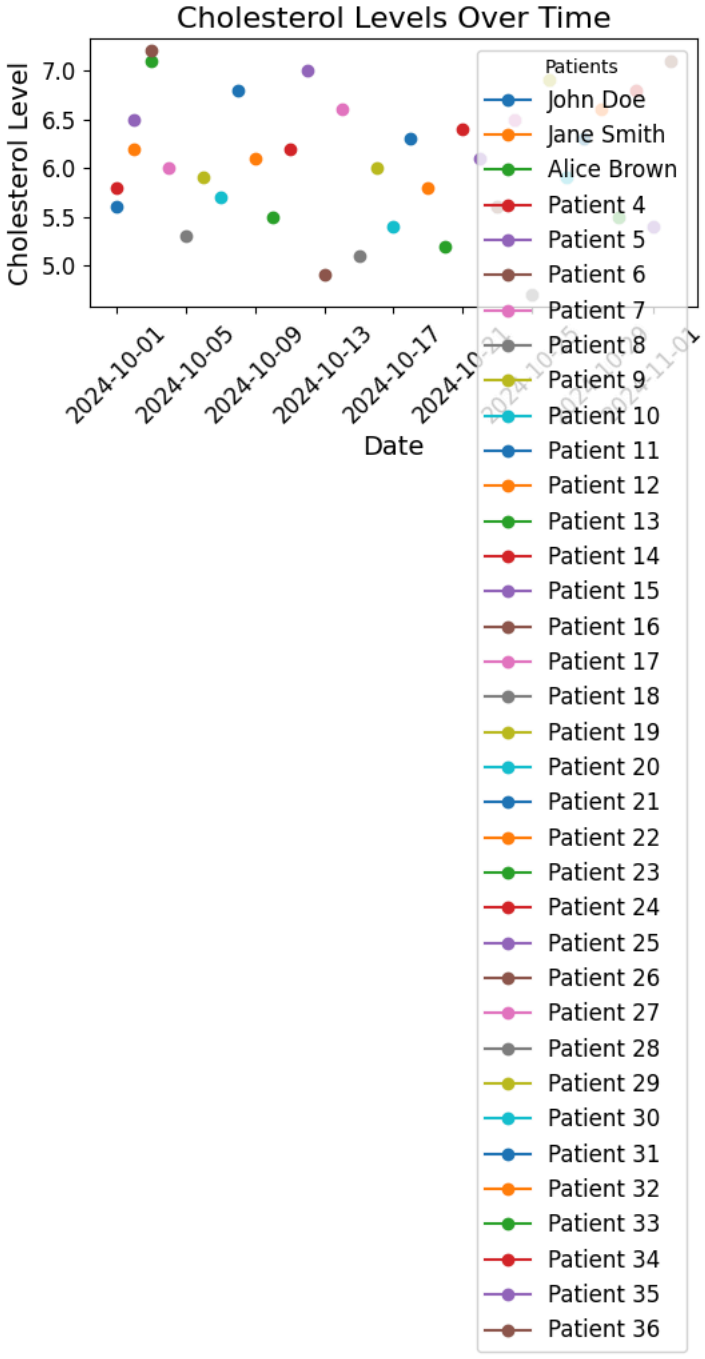

```
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 2)
for name in df['name'].unique():
    patient_data = df[df['name'] == name]
    plt.plot(patient_data['date_of_record'], patient_data['cholesterol'], marker='o', label=name)

plt.title('Cholesterol Levels Over Time')
plt.xlabel('Date')
plt.ylabel('Cholesterol Level')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x1d539d58aa0>
```



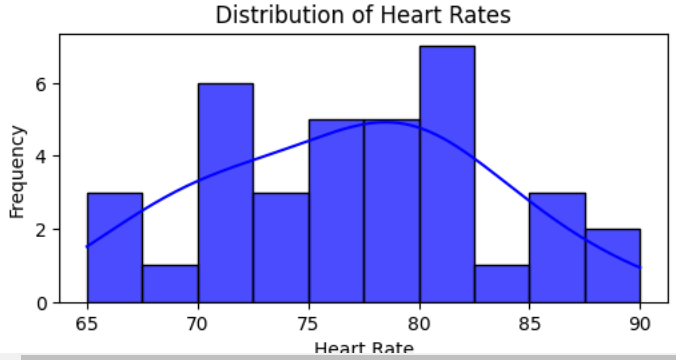
Start coding or generate with AI.

```
C:\Users\anish rane\AppData\Local\Temp\ipykernel_16872\4146644537.py:25: UserWarning: Tight layout not applied. tight_layout can
  plt.tight_layout()  # Automatically adjust subplot parameters for a clean layout
```



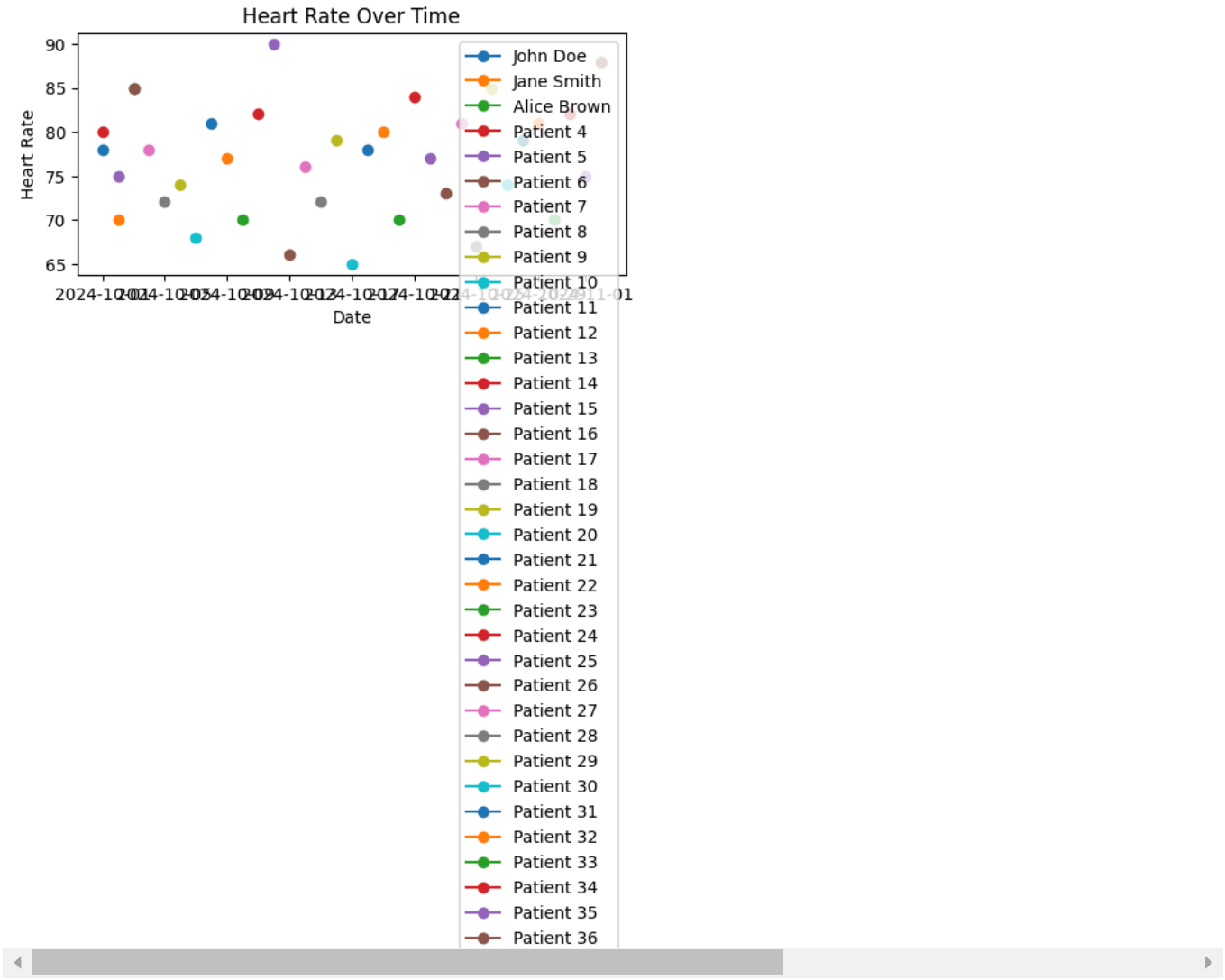

```
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 3)
sns.histplot(df['heart_rate'], bins=10, kde=True, color='blue', alpha=0.7)
plt.title('Distribution of Heart Rates')
plt.xlabel('Heart Rate')
plt.ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```

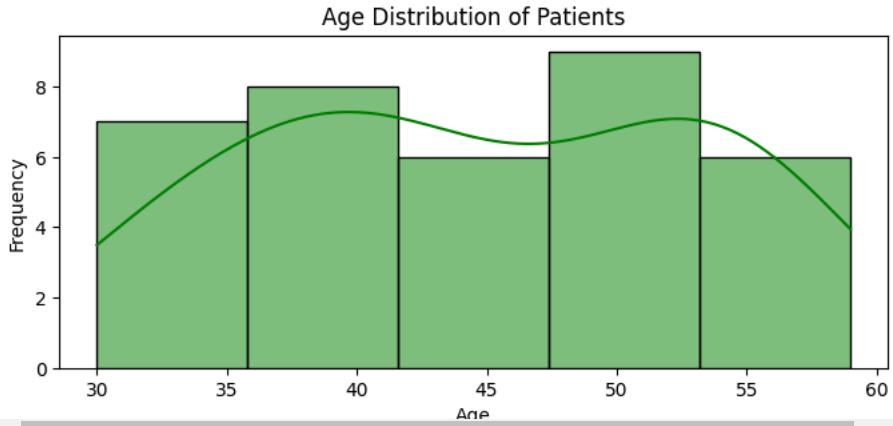


```
# Visualization 4: Heart Rate Over Time
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 4)
for name in df['name'].unique():
    patient_data = df[df['name'] == name]
    plt.plot(patient_data['date_of_record'], patient_data['heart_rate'], marker='o', label=name)

plt.title('Heart Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Heart Rate')
plt.legend()
```

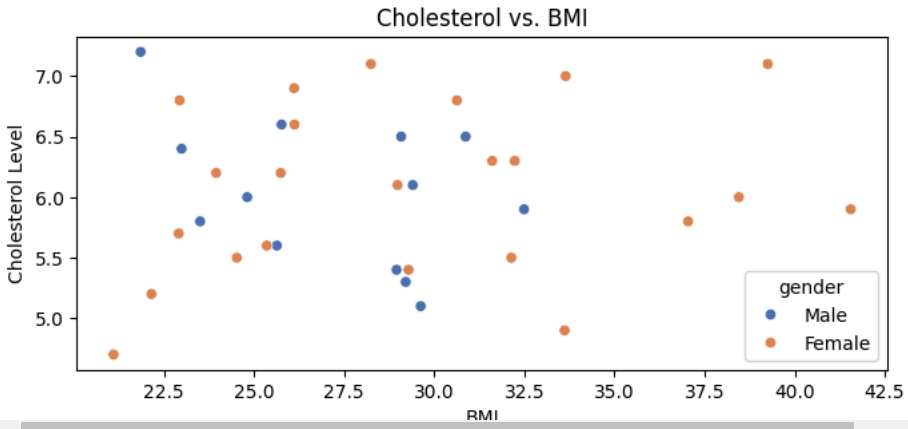
```
<matplotlib.legend.Legend at 0x1d53aba7500>
```




```
# Visualization 5: Age Distribution
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 5)
sns.histplot(df['age'], bins=5, color='green', kde=True)
plt.title('Age Distribution of Patients')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

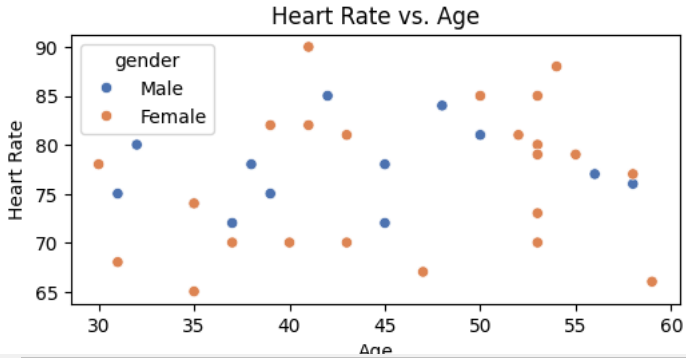



```
# Visualization 6: Cholesterol vs. BMI
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 6)
sns.scatterplot(x='bmi', y='cholesterol', data=df, hue='gender', palette='deep')
plt.title('Cholesterol vs. BMI')
plt.xlabel('BMI')
plt.ylabel('Cholesterol Level')
plt.tight_layout()
plt.show()
```




```
# Visualization 7: Heart Rate vs. Age
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 7)
sns.scatterplot(x='age', y='heart_rate', data=df, hue='gender', palette='deep')
plt.title('Heart Rate vs. Age')
plt.xlabel('Age')
plt.ylabel('Heart Rate')
```

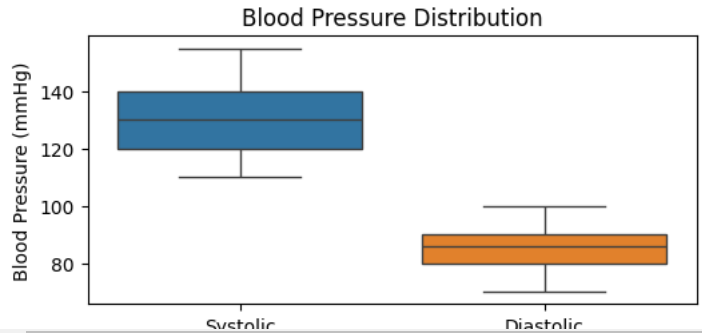
```
Text(0, 0.5, 'Heart Rate')
```




```
# Visualization 8: Blood Pressure Analysis
# Convert blood pressure to separate systolic and diastolic columns for analysis
plt.figure(figsize=(20, 12))
df[['systolic', 'diastolic']] = df['blood_pressure'].str.split('/', expand=True).astype(int)
plt.subplot(4, 3, 8)
sns.boxplot(data=df[['systolic', 'diastolic']])
plt.title('Blood Pressure Distribution')
```

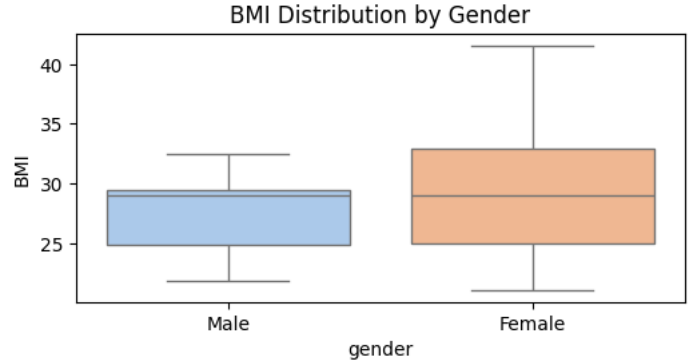
```
plt.ylabel('Blood Pressure (mmHg)')
plt.xticks([0, 1], ['Systolic', 'Diastolic'])
```

```
([<matplotlib.axis.XTick at 0x1d53c4258b0>,
  <matplotlib.axis.XTick at 0x1d53c3d5c40>],
 [Text(0, 0, 'Systolic'), Text(1, 0, 'Diastolic')])
```



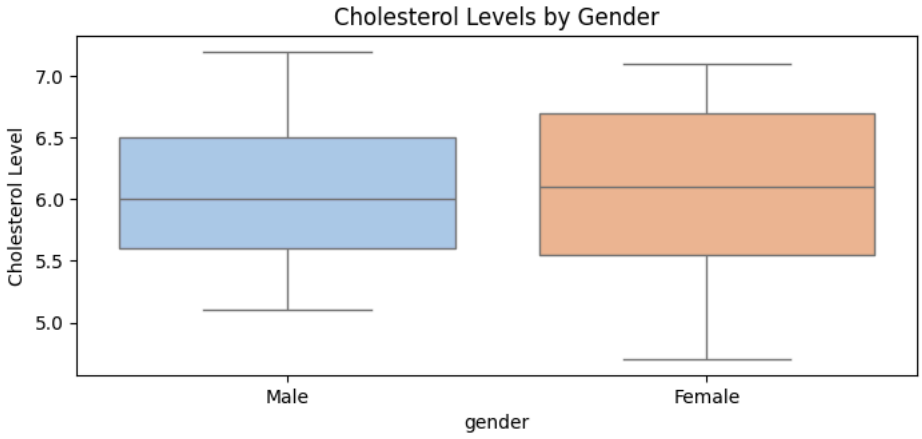

```
# Visualization 9: BMI Distribution by Gender
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 9)
sns.boxplot(x='gender', y='bmi', data=df, palette='pastel')
plt.title('BMI Distribution by Gender')
plt.ylabel('BMI')
```

```
C:\Users\anish rane\AppData\Local\Temp\ipykernel_16872\961257994.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.boxplot(x='gender', y='bmi', data=df, palette='pastel')
Text(0, 0.5, 'BMI')
```

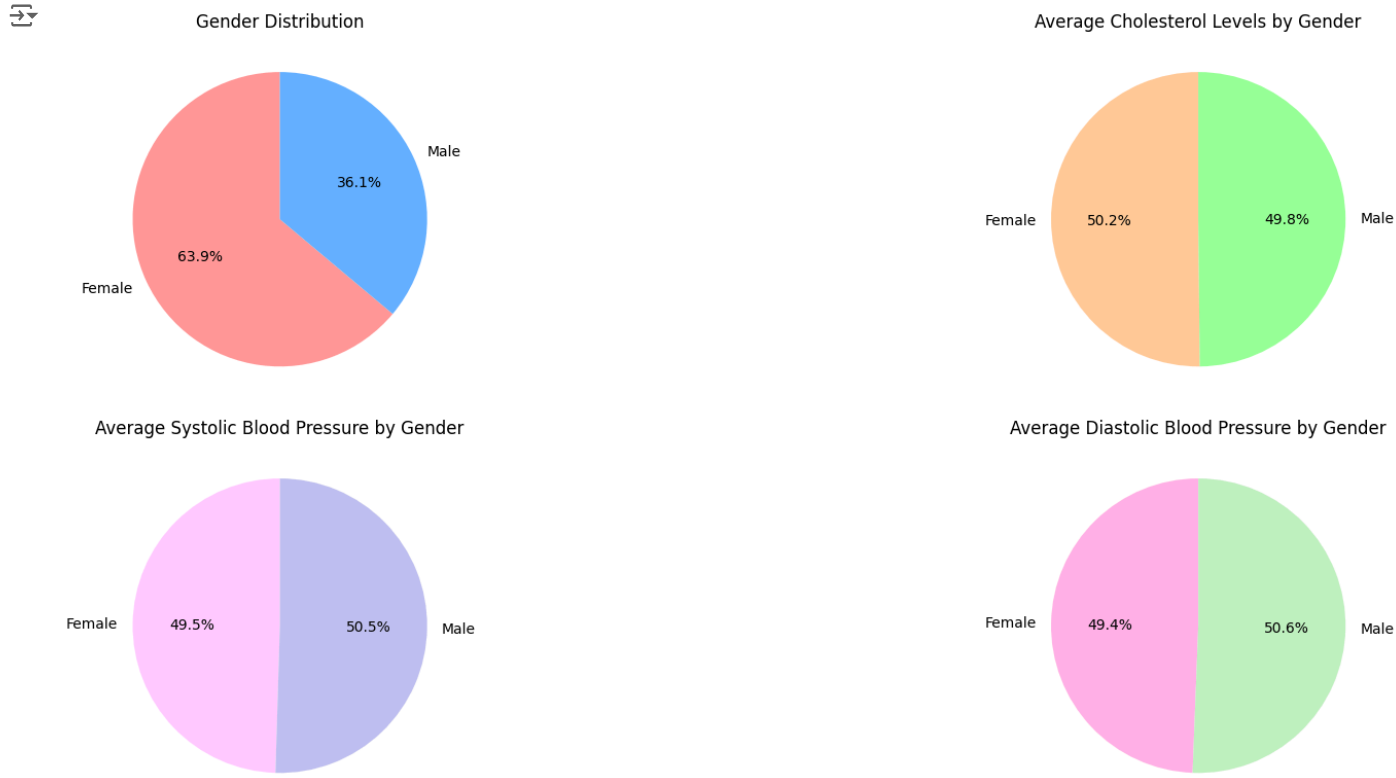



```
# Visualization 10: Cholesterol Levels by Gender
plt.figure(figsize=(20, 12))
plt.subplot(4, 3, 10)
sns.boxplot(x='gender', y='cholesterol', data=df, palette='pastel')
plt.title('Cholesterol Levels by Gender')
plt.ylabel('Cholesterol Level')

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

```
C:\Users\anish rane\AppData\Local\Temp\ipykernel_16872\1925240386.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.boxplot(x='gender', y='cholesterol', data=df, palette='pastel')
```




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

# Assuming you have already fetched your DataFrame `df` from the database as shown earlier

# 1. Count the number of males and females
gender_counts = df['gender'].value_counts()

# 2. Calculate average cholesterol and blood pressure by gender
# Convert blood pressure from string to two separate columns for systolic and diastolic
df[['systolic', 'diastolic']] = df['blood_pressure'].str.split('/', expand=True).astype(int)

average_cholesterol = df.groupby('gender')['cholesterol'].mean()
average_blood_pressure = df.groupby('gender')[['systolic', 'diastolic']].mean()

# 3. Plotting
plt.figure(figsize=(18, 12))

# Pie chart for Gender Distribution
plt.subplot(3, 2, 1)
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90, colors=['#ff9999','#66b3ff'])
plt.title('Gender Distribution')

# Pie chart for Average Cholesterol Levels
plt.subplot(3, 2, 2)
plt.pie(average_cholesterol, labels=average_cholesterol.index, autopct='%1.1f%%', startangle=90, colors=['#ffcc99','#99ff99'])
plt.title('Average Cholesterol Levels by Gender')

# Pie chart for Average Blood Pressure (Systolic)
plt.subplot(3, 2, 3)
plt.pie(average_blood_pressure['systolic'], labels=average_blood_pressure.index, autopct='%1.1f%%', startangle=90, colors=['#ffccff','#
plt.title('Average Systolic Blood Pressure by Gender')

# Pie chart for Average Blood Pressure (Diastolic)
plt.subplot(3, 2, 4)
plt.pie(average_blood_pressure['diastolic'], labels=average_blood_pressure.index, autopct='%1.1f%%', startangle=90, colors=['#ffb3e6','#
plt.title('Average Diastolic Blood Pressure by Gender')

plt.tight_layout()  # Automatically adjust subplot parameters for a clean layout
plt.show()
```




```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Convert categorical gender column to numeric (0 for Female, 1 for Male)
le = LabelEncoder()
df['gender'] = le.fit_transform(df['gender'])  # 0 for Female, 1 for Male

# Step 1: Categorize cholesterol levels
# Example: Normal < 200, Borderline 200-239, High >= 240
def categorize_cholesterol(value):
    if value < 200:
        return 'Normal'
    elif 200 <= value < 240:
        return 'Borderline'
    else:
        return 'High'

df['cholesterol_category'] = df['cholesterol'].apply(categorize_cholesterol)

# Convert cholesterol categories to numerical labels
le = LabelEncoder()
df['cholesterol_category'] = le.fit_transform(df['cholesterol_category'])

# Step 2: Define the features (X) and target (y)
X = df[['age', 'bmi', 'gender', 'heart_rate', 'systolic', 'diastolic']]
y = df['cholesterol_category']

# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 5: Train the model (RandomForestClassifier for classification)
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 6: Make predictions on the test data
```

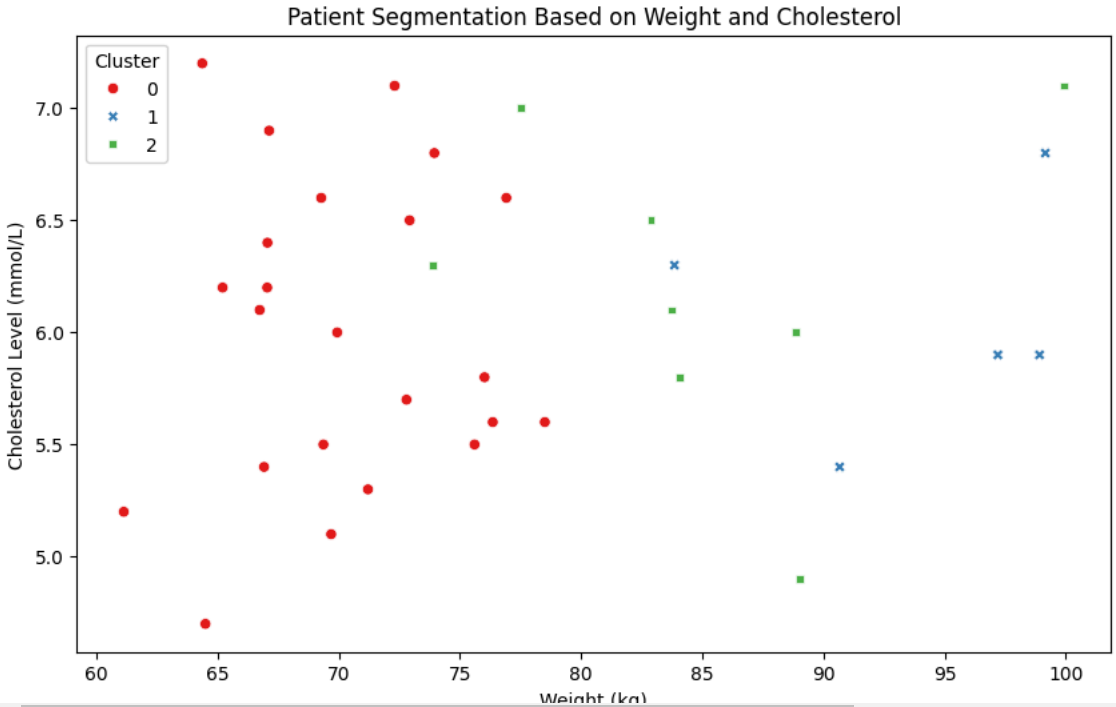
```
y_pred = model.predict(X_test)

# Step 7: Calculate accuracy
accuracy = accuracy_score(y_test, y_pred) * 100
print(f'Accuracy: {accuracy:.2f}%')

# Step 8: Classification report for detailed metrics
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
Accuracy: 100.00%
                  precision    recall  f1-score   support

          Normal       1.00      1.00      1.00         8

        accuracy                           1.00         8
       macro avg       1.00      1.00      1.00         8
    weighted avg       1.00      1.00      1.00         8
```

```
# Step 1: Find Alice's data in the DataFrame
alice_data = df[df['name'] == 'Alice Brown']

if alice_data.empty:
    print("Alice's data not found in the dataset.")
else:
    # Step 2: Select relevant features for prediction (same as used for training)
    X_alice = alice_data[['age', 'bmi', 'gender', 'heart_rate', 'systolic', 'diastolic']]

    # Step 3: Apply the same preprocessing (scaling)
    X_alice_scaled = scaler.transform(X_alice)

    # Step 4: Make the prediction
    alice_prediction = model.predict(X_alice_scaled)

    # Step 5: Convert the numeric prediction back to the category (Normal, Borderline, High)
    predicted_category = le.inverse_transform(alice_prediction)

    print(f"Alice's predicted cholesterol category: {predicted_category[0]}")
```

```
Alice's predicted cholesterol category: Normal
```

```
import mysql.connector
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns



# Calculate BMI for each patient
df['bmi'] = df['weight'] / ((df['height'] / 100) ** 2)

# Step 4: Prepare data for clustering
features = df[['age', 'weight', 'cholesterol', 'bmi']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Step 5: Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['cluster'] = kmeans.fit_predict(scaled_features)

# Step 6: Visualize Segmentation
plt.figure(figsize=(10, 6))
sns.scatterplot(x='weight', y='cholesterol', hue='cluster', data=df, palette='Set1', style='cluster')
plt.title('Patient Segmentation Based on Weight and Cholesterol')
plt.xlabel('Weight (kg)')
plt.ylabel('Cholesterol Level (mmol/L)')
plt.legend(title='Cluster')
plt.show()
```




```
# Convert 'date_of_record' to datetime
df['date_of_record'] = pd.to_datetime(df['date_of_record'])

# Step 4: Plot Cholesterol Levels Over Time for Each Patient
plt.figure(figsize=(12, 6))
for name in df['name'].unique():
    patient_data = df[df['name'] == name]
    plt.plot(patient_data['date_of_record'], patient_data['cholesterol'], marker='o', label=name)

plt.title('Cholesterol Levels Over Time for Each Patient')
plt.xlabel('Date')
plt.ylabel('Cholesterol Level (mmol/L)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


```
import numpy as np
import pandas as pd
import mysql.connector
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Assuming you have already fetched your DataFrame `df` from the database

# Define conditions based on thresholds
def label_conditions(row):
    conditions = []
    if row['systolic'] > 140 or row['diastolic'] > 90:
        conditions.append('Hypertension')
    if row['cholesterol'] > 6.2:
        conditions.append('Hyperlipidemia')
    if row['bmi'] > 30:
        conditions.append('Obesity')
    return ', '.join(conditions) if conditions else 'Healthy'

# Apply function to label conditions
df['conditions'] = df.apply(label_conditions, axis=1)

# Visualize conditions
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='conditions', palette='viridis')
```