xnjmzbjat

December 17, 2024

1 Health Monitoring and Analysis Project

Welcome to the Health Monitoring and Analysis project! This project aims to explore and analyze health data to uncover patterns, correlations, and insights related to various health metrics. Below is a structured roadmap that outlines the key steps taken in the analysis process, and explains why each step is important.

1.1 Table of Contents

1.2 1. Data Understanding and Preparation

1.2.1 Dataset Overview

- **Explanation**: The first step is understanding the dataset's structure, its columns, and the type of health metrics available for analysis.
- **Impact of Skipping**: Without this understanding, you may miss crucial variables or misinterpret the data, leading to inaccurate analysis.

1.2.2 Data Cleaning and Imputation

- **Explanation**: Handling missing values and incorrect data entries ensures that the dataset is accurate and ready for analysis.
- **Why It's Necessary**: Missing or erroneous values could distort any analysis, leading to unreliable results.
  - Missing values in columns such as Body Temperature and Oxygen Saturation are filled using their median values.
  - Incorrect or poorly formatted data (such as Blood Pressure in a single string format) is split and converted for consistency.
- **Impact of Skipping**: If missing or incorrect data is left unaddressed, it could lead to misleading conclusions or errors during statistical analysis.

1.2.3 Feature Engineering

- **Explanation**: Creating new features (such as categorizing Age, Blood Pressure, Heart Rate, and Oxygen Saturation) allows for more meaningful analysis.
- **Why It's Necessary**: Derived features help to categorize patients into different groups (e.g., young, middle-aged, senior), which can provide better insights into trends and patterns.
- **Impact of Skipping**: Without feature engineering, key patterns related to age groups, blood pressure categories, etc., would not be visible.

1.3 2. Exploratory Data Analysis (EDA)

1.3.1 Distribution Analysis

- **Explanation**: Understanding the distribution of key health metrics (Age, Heart Rate, Blood Pressure, etc.) is crucial for understanding the dataset's characteristics.
- **Why It's Necessary**: This step helps to uncover outliers, identify skewness, and get an overview of the general distribution of variables.
- **Impact of Skipping**: Missing distribution analysis can lead to misinterpretation of data, especially if certain variables are skewed or contain outliers.

1.3.2 Relationship and Correlation Analysis

- **Explanation**: Understanding how variables relate to each other helps in identifying key correlations and insights.
- **Why It's Necessary**: Correlation analysis identifies which metrics (e.g., Age, Heart Rate, Blood Pressure) are most strongly related, which could be useful for predictive modeling or understanding underlying health trends.
- **Impact of Skipping**: Without this analysis, hidden relationships between variables could go unnoticed, leading to incomplete insights.

1.4 3. Visualizations

1.4.1 Univariate Analysis

- **Explanation**: Visualizing the distribution of individual metrics, such as Age, Heart Rate, Blood Pressure, and Oxygen Saturation, provides insight into the frequency and range of these health metrics.
- **Why It's Necessary**: This type of analysis helps to quickly spot patterns, distributions, and outliers in individual variables.
- **Impact of Skipping**: Without these visualizations, it would be hard to get an overview of how individual health metrics behave across the population.

1.4.2 Bivariate Analysis

- **Explanation**: Visualizing the relationship between two health metrics at a time (e.g., Heart Rate vs. Blood Pressure) helps understand their interactions.
- **Why It's Necessary**: Bivariate plots reveal how two variables influence each other, which can be essential for understanding complex health relationships.
- **Impact of Skipping**: Missing these insights might obscure potential interactions between health metrics.

1.4.3 Health Category Analysis

- **Explanation**: Exploring the distribution of various health categories, such as Blood Pressure Categories, Heart Rate Categories, and Oxygen Saturation Categories, based on demographic features like Age and Gender.
- **Why It's Necessary**: This analysis can highlight health disparities and help identify which categories are more prevalent in certain demographics.
- **Impact of Skipping**: Without this step, we could miss important demographic-based trends that are essential for targeted health interventions.

1.5 4. Statistical Analysis

1.5.1 Correlation Heatmap

- **Explanation**: Visualizing the correlation between different health metrics using a heatmap allows for an understanding of how different health parameters influence each other.
- **Why It's Necessary**: Correlation analysis helps determine which variables should be used together in models and provides insights into the underlying health patterns.
- **Impact of Skipping**: Without this, it would be difficult to see strong interrelationships between variables, potentially leading to missed patterns.

1.5.2 Statistical Comparison Across Groups

- **Explanation**: Performing statistical tests (like boxplots) to compare health metrics across different groups (e.g., Gender, Sleep Quality, Stress Level) helps in understanding differences within the population.
- **Why It's Necessary**: These comparisons highlight important health disparities, such as gender-based differences in Heart Rate or Oxygen Saturation.

- **Impact of Skipping**: Without these comparisons, potential health differences could be overlooked, which is critical for understanding population health.

1.6 5. Conclusion and Insights

- **Explanation**: The final step is to summarize the findings and draw conclusions from the exploratory and statistical analyses.
- **Why It's Necessary**: Drawing insights from the data enables the formulation of actionable recommendations, which can inform health policies, interventions, or further research.
- **Impact of Skipping**: Without this step, the entire analysis would remain abstract and not lead to any useful conclusions or recommendations.

This roadmap ensures that all necessary steps for a comprehensive health analysis are covered, with each stage being clearly explained for context. Now, let's dive into the code implementation that follows the outlined steps.

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
```

[2]:
```
# Load the dataset
health_data = pd.read_csv('healthmonitoring.csv')
print(health_data.head())
```

```
   PatientID  Age  Gender  HeartRate BloodPressure  RespiratoryRate  \
0          1   69    Male  60.993428        130/85               15
1          2   32    Male  98.723471        120/80               23
2          3   78  Female  82.295377        130/85               13
3          4   38  Female  80.000000        111/78               19
4          5   41    Male  87.531693        120/80               14

   BodyTemperature ActivityLevel  OxygenSaturation SleepQuality StressLevel  \
0        98.885236       resting              95.0    excellent         low
1        98.281883       walking              97.0         good        high
2        98.820286       resting              98.0         fair        high
3        98.412594       running              98.0         poor    moderate
4        99.369871       resting              98.0         good         low

                   Timestamp
0  2024-04-26 17:28:55.286711
1  2024-04-26 17:23:55.286722
2  2024-04-26 17:18:55.286726
3  2024-04-26 17:13:55.286728
4  2024-04-26 17:08:55.286731
```

[3]:
```
print(health_data.dtypes)
```

```
PatientID             int64
Age                   int64
Gender               object
HeartRate           float64
BloodPressure        object
RespiratoryRate       int64
BodyTemperature     float64
ActivityLevel        object
OxygenSaturation    float64
SleepQuality         object
StressLevel          object
Timestamp            object
dtype: object
```

[4]:
```
health_data.isnull().sum()
```

[4]:
```
PatientID             0
Age                   0
Gender                0
HeartRate             0
BloodPressure         0
RespiratoryRate       0
BodyTemperature      18
ActivityLevel         0
OxygenSaturation    163
SleepQuality          0
StressLevel           0
Timestamp             0
dtype: int64
```

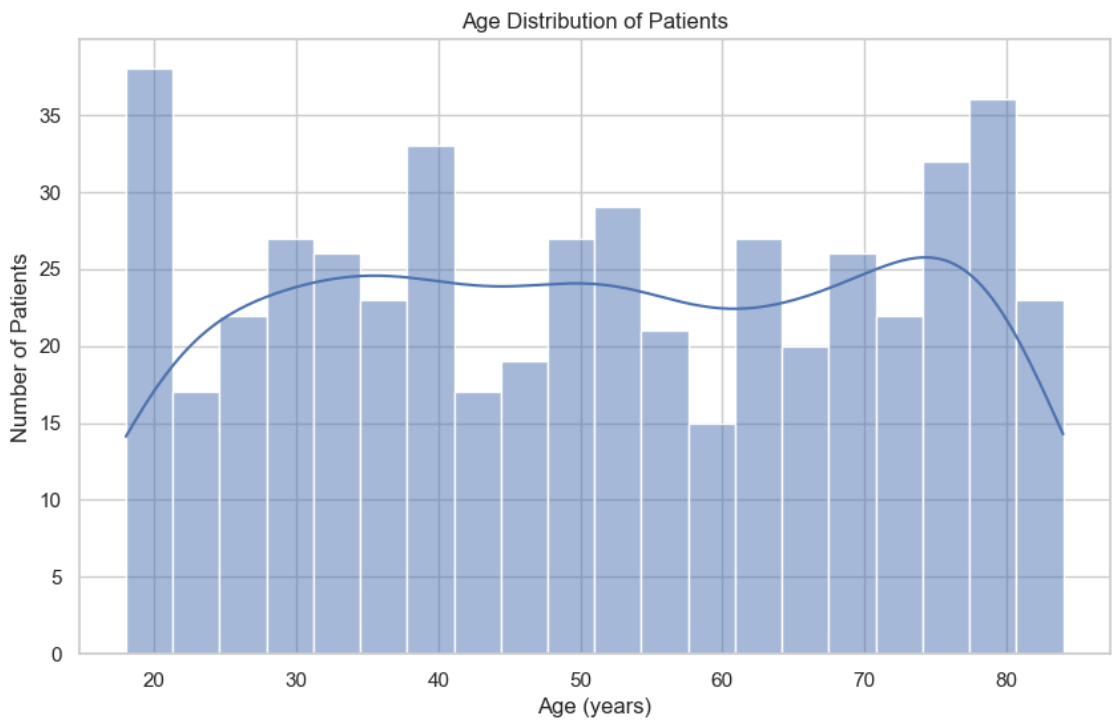
[5]:
```
# Fill missing values
median_body_temp = health_data['BodyTemperature'].median()
median_oxygen_sat = health_data['OxygenSaturation'].median()
health_data['BodyTemperature'] = health_data['BodyTemperature'].
  ↪fillna(median_body_temp)
health_data['OxygenSaturation'] = health_data['OxygenSaturation'].
  ↪fillna(median_oxygen_sat)
```

[6]:
```
# Data Preparation for Blood Pressure
health_data[['SystolicBP', 'DiastolicBP']] = health_data['BloodPressure'].str.
  ↪split('/', expand=True).astype(int)

# function to categorize Age
def age_group(age):
    if age <= 35:
        return 'Young'
    elif age <= 55:
        return 'Middle-aged'
    else:
        return 'Senior'

# function to categorize Blood Pressure
def bp_category(systolic, diastolic):
    if systolic < 120 and diastolic < 80:
        return 'Normal'
    elif 120 <= systolic < 140 or 80 <= diastolic < 90:
        return 'Elevated'
    elif 140 <= systolic < 160 or 90 <= diastolic < 100:
        return 'Hypertension Stage 1'
    else:
        return 'Hypertension Stage 2'

# function to categorize Heart Rate
def hr_category(hr):
    if hr < 60:
        return 'Low'
    elif hr <= 100:
        return 'Normal'
    else:
        return 'High'

# function to categorize Oxygen Saturation
def oxy_category(oxy):
    if oxy < 94:
        return 'Low'
    else:
        return 'Normal'

# applying categorizations
health_data['AgeGroup'] = health_data['Age'].apply(age_group)
health_data['BPCategory'] = health_data.apply(lambda x:␣
 ↪bp_category(x['SystolicBP'], x['DiastolicBP']), axis=1)
health_data['HRCategory'] = health_data['HeartRate'].apply(hr_category)
health_data['OxyCategory'] = health_data['OxygenSaturation'].apply(oxy_category)

print(health_data[['Age', 'AgeGroup', 'SystolicBP', 'DiastolicBP',␣
 ↪'BPCategory', 'HeartRate', 'HRCategory', 'OxygenSaturation', 'OxyCategory']].
 ↪head())
```

```
   Age     AgeGroup  SystolicBP  DiastolicBP BPCategory  HeartRate HRCategory  \
0   69       Senior         130           85   Elevated  60.993428     Normal
1   32        Young         120           80   Elevated  98.723471     Normal
```

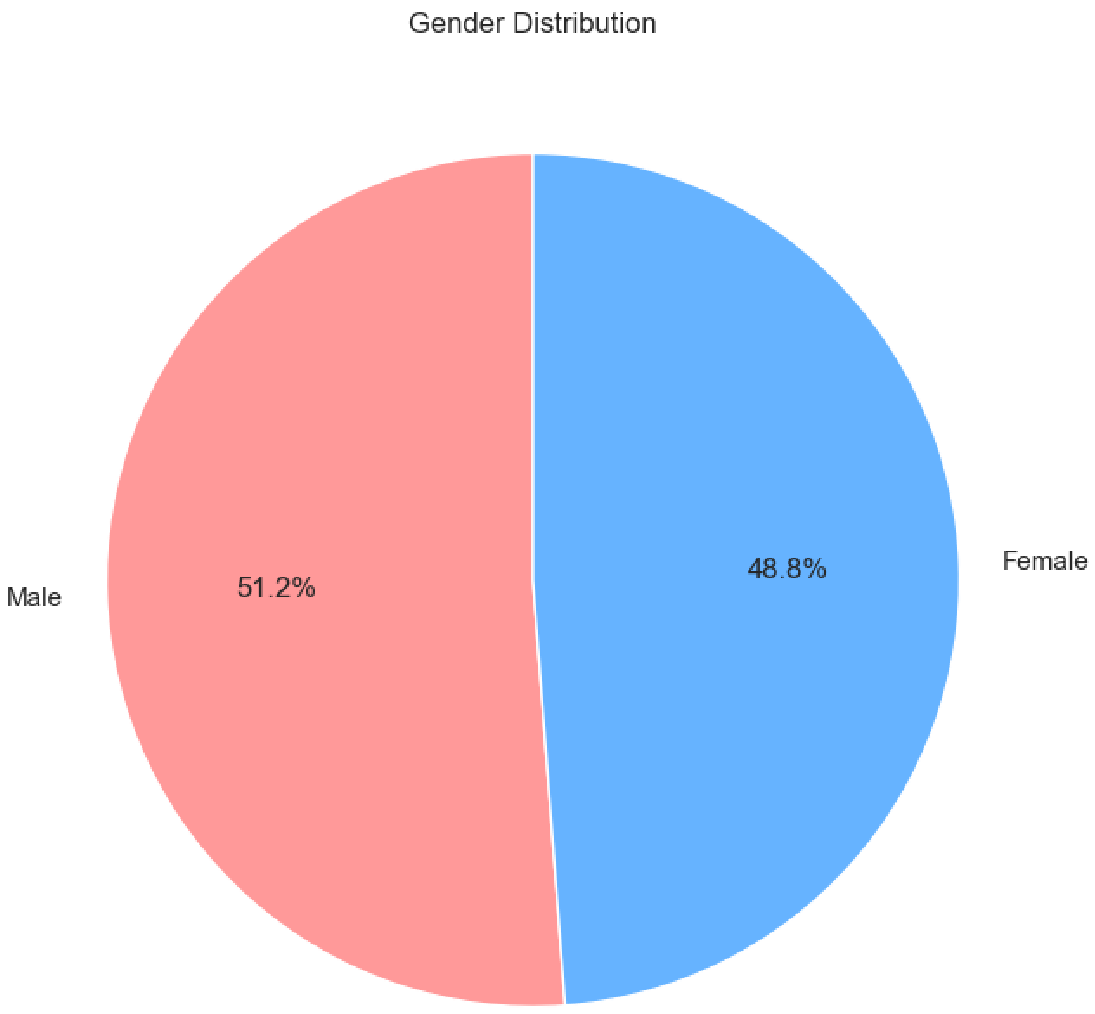
```
2   78       Senior          130            85   Elevated  82.295377     Normal
3   38  Middle-aged          111            78     Normal  80.000000     Normal
4   41  Middle-aged          120            80   Elevated  87.531693     Normal

   OxygenSaturation OxyCategory
0              95.0      Normal
1              97.0      Normal
2              98.0      Normal
3              98.0      Normal
4              98.0      Normal
```

[7]:
```
# 1. Bar Chart for Age Distribution
plt.figure(figsize=(10, 6))
sns.histplot(health_data['Age'], bins=20, kde=True)
plt.title('Age Distribution of Patients')
plt.xlabel('Age (years)')
plt.ylabel('Number of Patients')
plt.show()
```



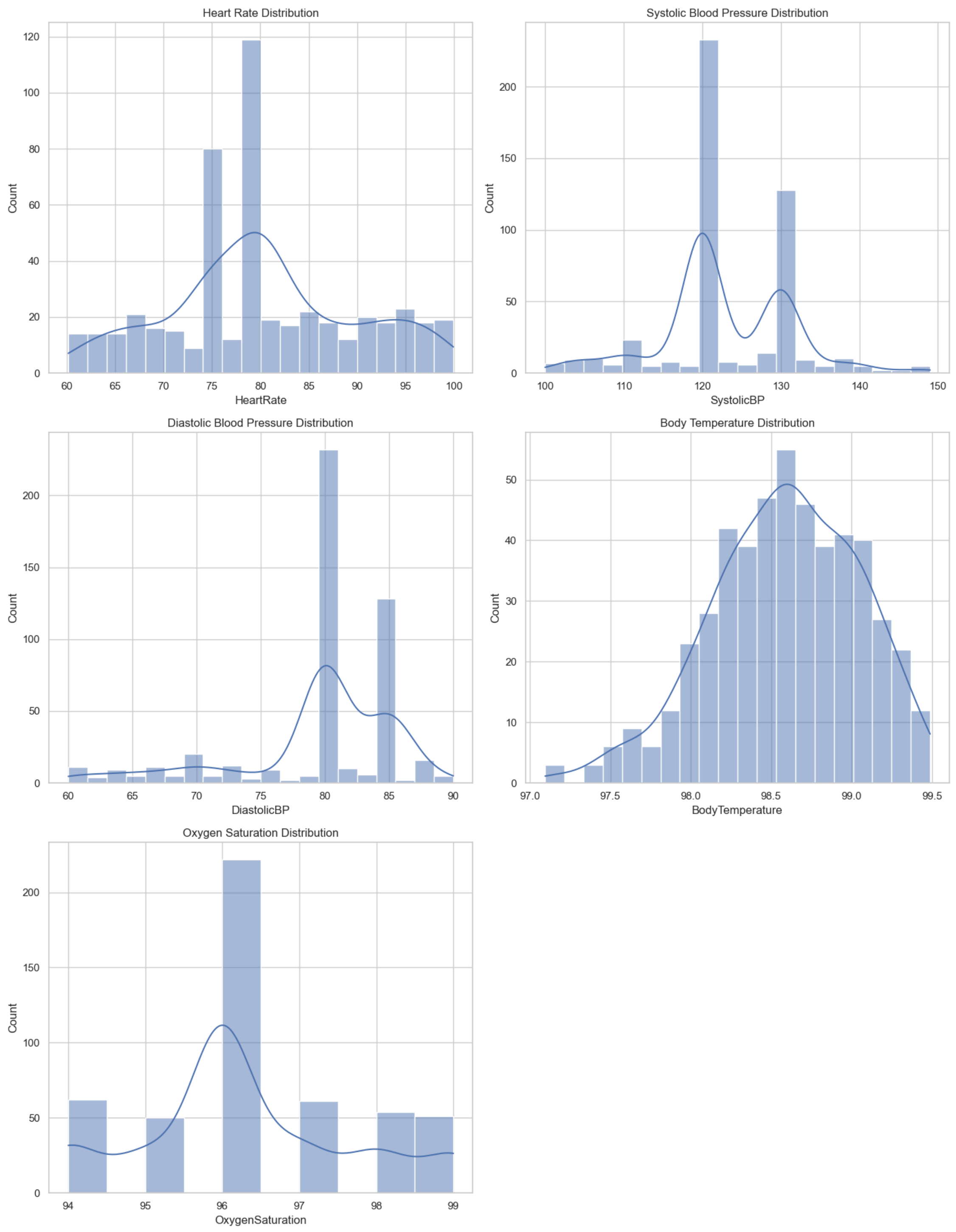
[8]:
```
# 2. Pie Chart for Gender Distribution
gender_counts = health_data['Gender'].value_counts()
plt.figure(figsize=(8, 8))
gender_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90,␣
 ↪colors=['#ff9999','#66b3ff'])
```

```
plt.title('Gender Distribution')
plt.ylabel('')
plt.show()
```



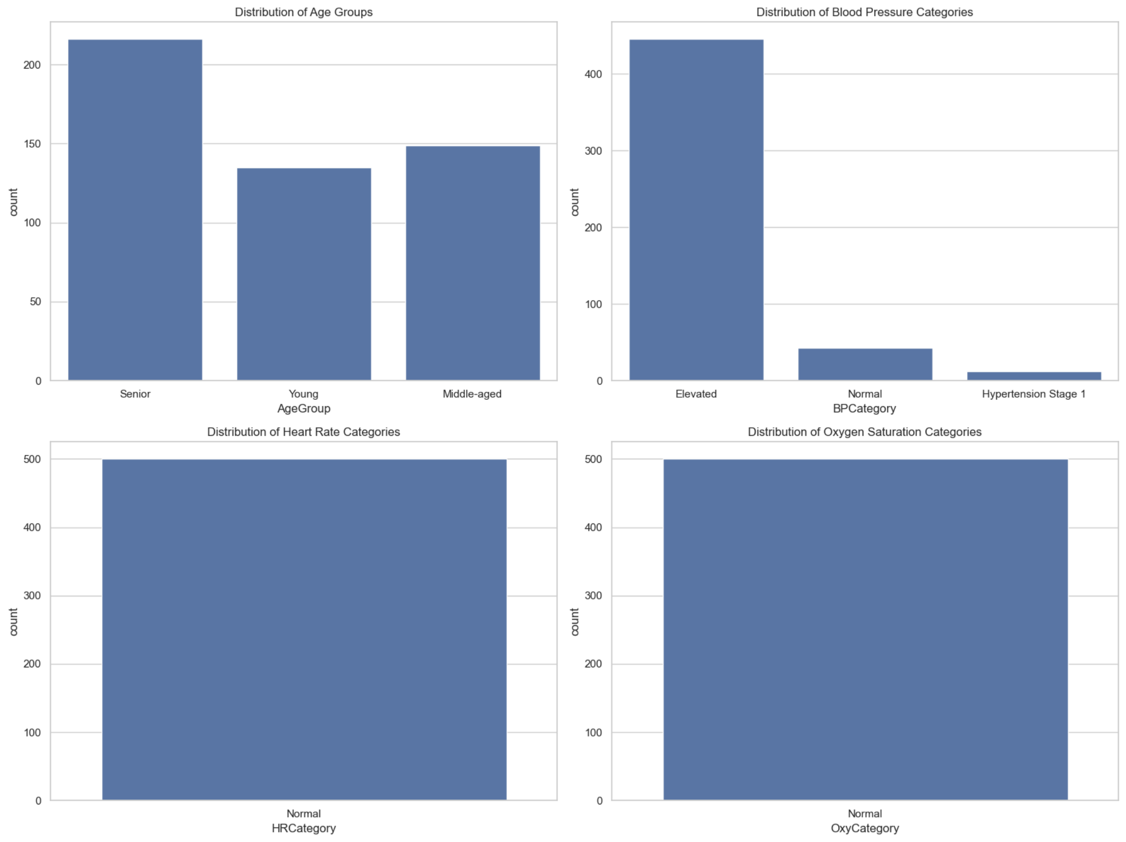
[9]:
```
# 3. Histograms for Key Metrics
fig, axes = plt.subplots(3, 2, figsize=(14, 18))

# Heart Rate Distribution
sns.histplot(health_data['HeartRate'], bins=20, kde=True, ax=axes[0, 0])
axes[0, 0].set_title('Heart Rate Distribution')

# Systolic Blood Pressure Distribution
```

```python
sns.histplot(health_data['SystolicBP'], bins=20, kde=True, ax=axes[0, 1])
axes[0, 1].set_title('Systolic Blood Pressure Distribution')

# Diastolic Blood Pressure Distribution
sns.histplot(health_data['DiastolicBP'], bins=20, kde=True, ax=axes[1, 0])
axes[1, 0].set_title('Diastolic Blood Pressure Distribution')

# Body Temperature Distribution
sns.histplot(health_data['BodyTemperature'], bins=20, kde=True, ax=axes[1, 1])
axes[1, 1].set_title('Body Temperature Distribution')

# Oxygen Saturation Distribution
sns.histplot(health_data['OxygenSaturation'], bins=10, kde=True, ax=axes[2, 0])
axes[2, 0].set_title('Oxygen Saturation Distribution')

# Remove unused subplot
fig.delaxes(axes[2, 1])

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



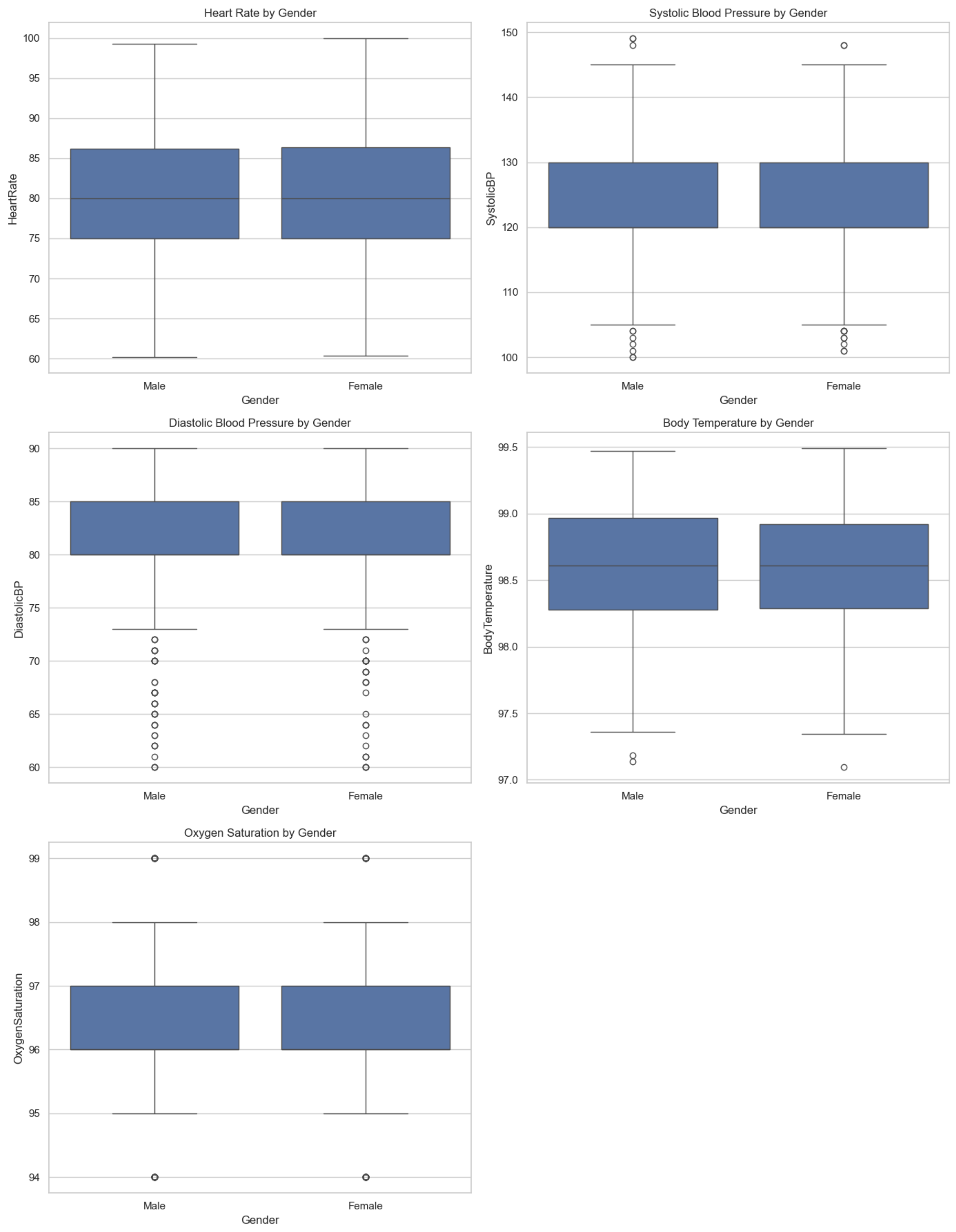
```
[10]: # 4. Count Plots for Health Categories
fig, axes = plt.subplots(2, 2, figsize=(16, 12))
sns.countplot(x='AgeGroup', data=health_data, ax=axes[0, 0])
axes[0, 0].set_title('Distribution of Age Groups')
```

```
sns.countplot(x='BPCategory', data=health_data, ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Blood Pressure Categories')

sns.countplot(x='HRCategory', data=health_data, ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Heart Rate Categories')

sns.countplot(x='OxyCategory', data=health_data, ax=axes[1, 1])
axes[1, 1].set_title('Distribution of Oxygen Saturation Categories')

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



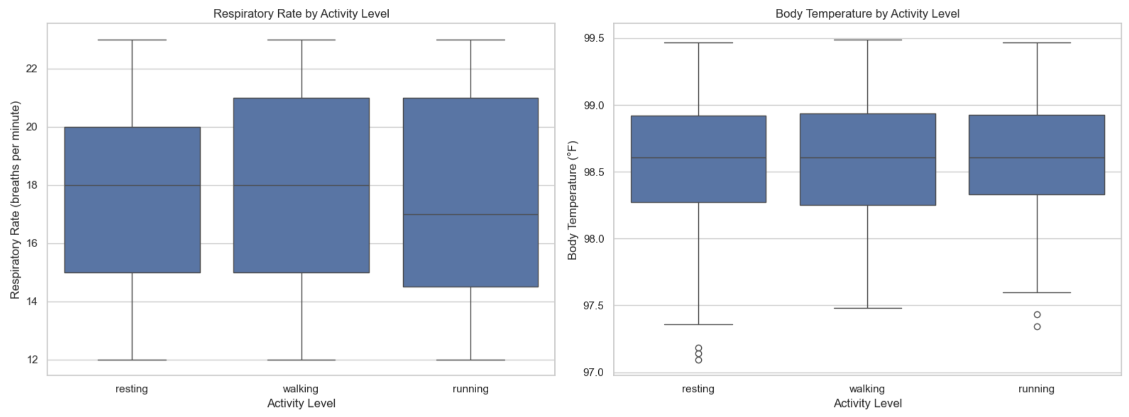
[11]:
```
# 5. Boxplots for each metric grouped by Gender
fig, axes = plt.subplots(3, 2, figsize=(14, 18))

# Boxplot for Heart Rate by Gender
sns.boxplot(x='Gender', y='HeartRate', data=health_data, ax=axes[0, 0])
axes[0, 0].set_title('Heart Rate by Gender')

# Boxplot for Systolic Blood Pressure by Gender
```

```python
sns.boxplot(x='Gender', y='SystolicBP', data=health_data, ax=axes[0, 1])
axes[0, 1].set_title('Systolic Blood Pressure by Gender')

# Boxplot for Diastolic Blood Pressure by Gender
sns.boxplot(x='Gender', y='DiastolicBP', data=health_data, ax=axes[1, 0])
axes[1, 0].set_title('Diastolic Blood Pressure by Gender')

# Boxplot for Body Temperature by Gender
sns.boxplot(x='Gender', y='BodyTemperature', data=health_data, ax=axes[1, 1])
axes[1, 1].set_title('Body Temperature by Gender')

# Boxplot for Oxygen Saturation by Gender
sns.boxplot(x='Gender', y='OxygenSaturation', data=health_data, ax=axes[2, 0])
axes[2, 0].set_title('Oxygen Saturation by Gender')

# Remove unused subplot
fig.delaxes(axes[2, 1])

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

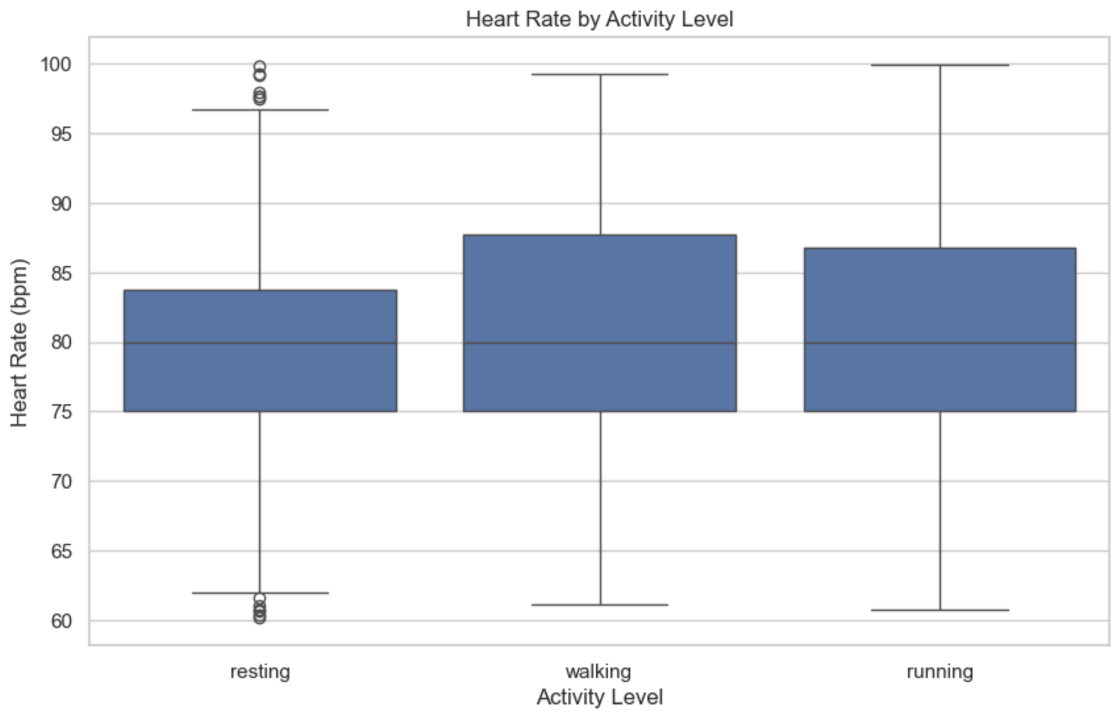


[12]:
```
# 6. Boxplots of Respiratory Rate and Body Temperature by Activity Level
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
sns.boxplot(x='ActivityLevel', y='RespiratoryRate', data=health_data,␣
  ↪ax=axes[0])
```

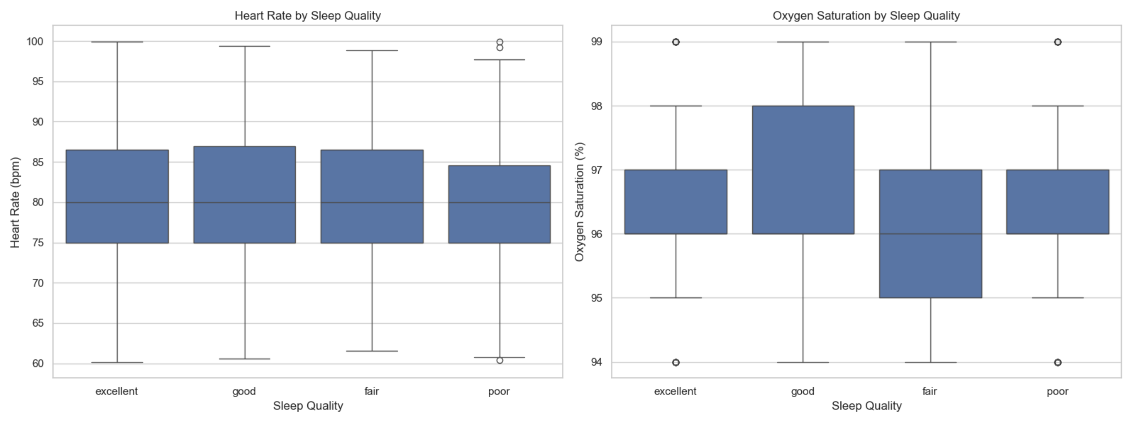
```
axes[0].set_title('Respiratory Rate by Activity Level')
axes[0].set_ylabel('Respiratory Rate (breaths per minute)')
axes[0].set_xlabel('Activity Level')

sns.boxplot(x='ActivityLevel', y='BodyTemperature', data=health_data,␣
  ↪ax=axes[1])
axes[1].set_title('Body Temperature by Activity Level')
axes[1].set_ylabel('Body Temperature (°F)')
axes[1].set_xlabel('Activity Level')

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



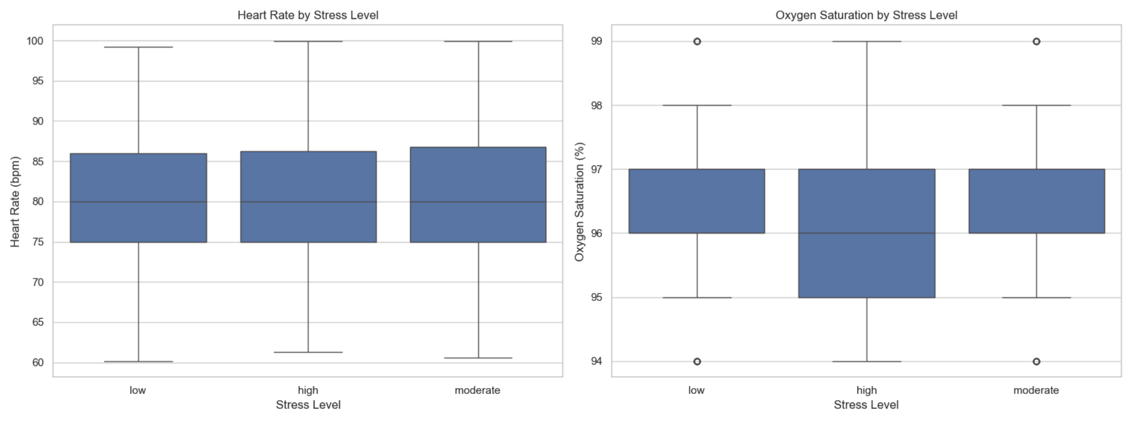
[13]:
```
# 7. Boxplot of Heart Rate by Activity Level
plt.figure(figsize=(10, 6))
sns.boxplot(x='ActivityLevel', y='HeartRate', data=health_data)
plt.title('Heart Rate by Activity Level')
plt.ylabel('Heart Rate (bpm)')
plt.xlabel('Activity Level')
plt.show()
```



[14]:
```
# 8. Boxplots for Heart Rate and Oxygen Saturation by Sleep Quality
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
sns.boxplot(x='SleepQuality', y='HeartRate', data=health_data, ax=axes[0])
axes[0].set_title('Heart Rate by Sleep Quality')
axes[0].set_ylabel('Heart Rate (bpm)')
axes[0].set_xlabel('Sleep Quality')

sns.boxplot(x='SleepQuality', y='OxygenSaturation', data=health_data,␣
  ↪ax=axes[1])
axes[1].set_title('Oxygen Saturation by Sleep Quality')
axes[1].set_ylabel('Oxygen Saturation (%)')
axes[1].set_xlabel('Sleep Quality')

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

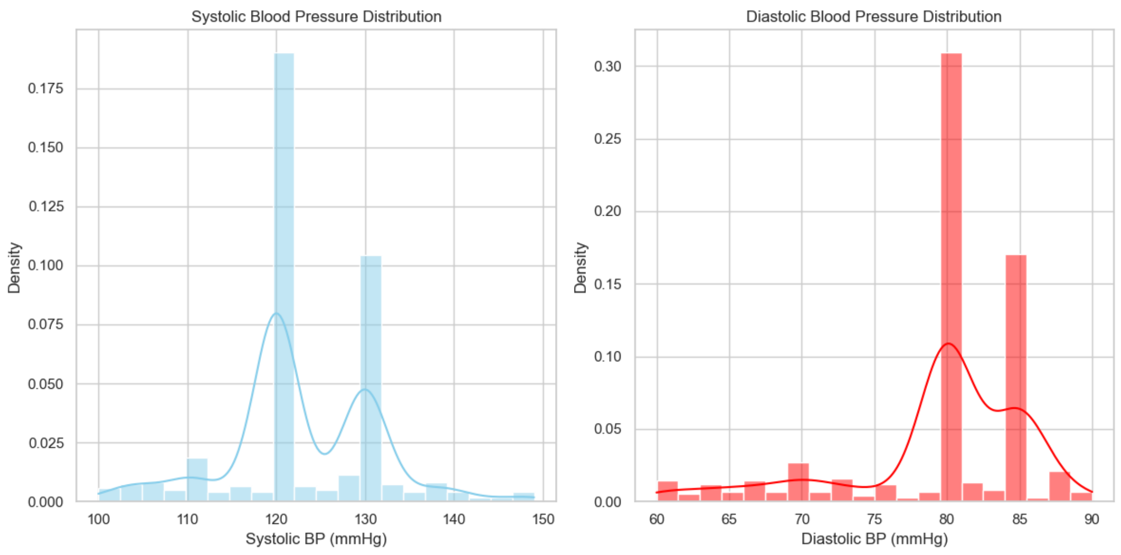


[15]:
```
# 9. Boxplots for Heart Rate and Oxygen Saturation by Stress Level
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
sns.boxplot(x='StressLevel', y='HeartRate', data=health_data, ax=axes[0])
axes[0].set_title('Heart Rate by Stress Level')
axes[0].set_ylabel('Heart Rate (bpm)')
axes[0].set_xlabel('Stress Level')

sns.boxplot(x='StressLevel', y='OxygenSaturation', data=health_data, ax=axes[1])
axes[1].set_title('Oxygen Saturation by Stress Level')
axes[1].set_ylabel('Oxygen Saturation (%)')
axes[1].set_xlabel('Stress Level')

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

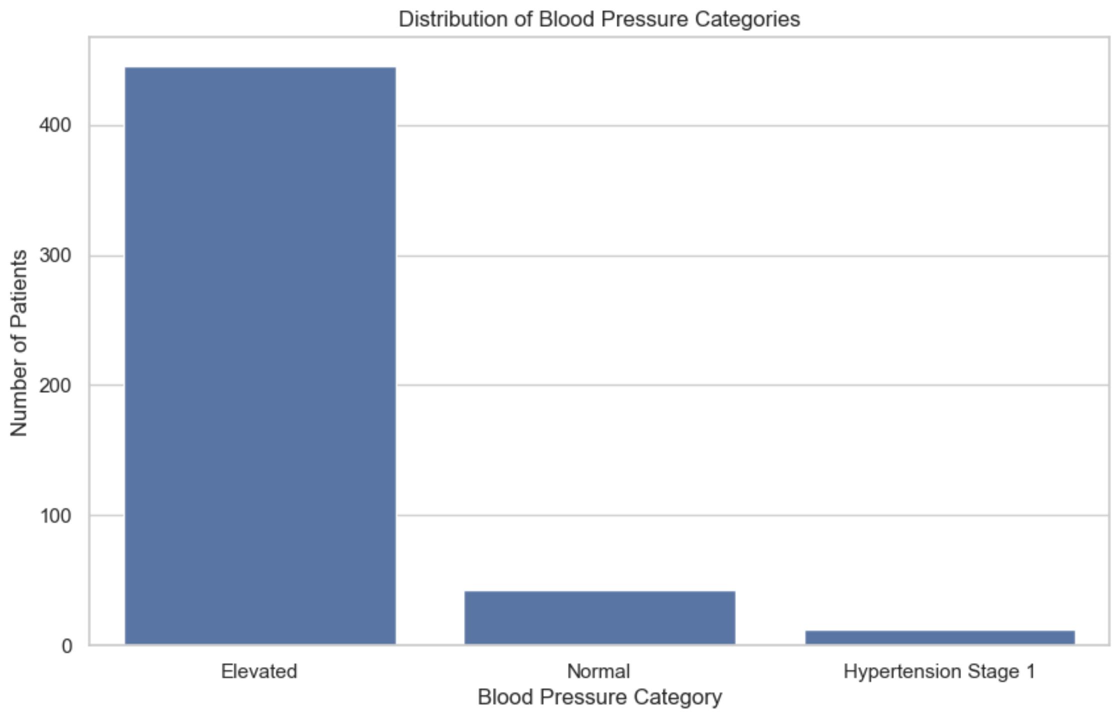


[16]:
```
# 10. Dual Histogram for Systolic and Diastolic Blood Pressure
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
```

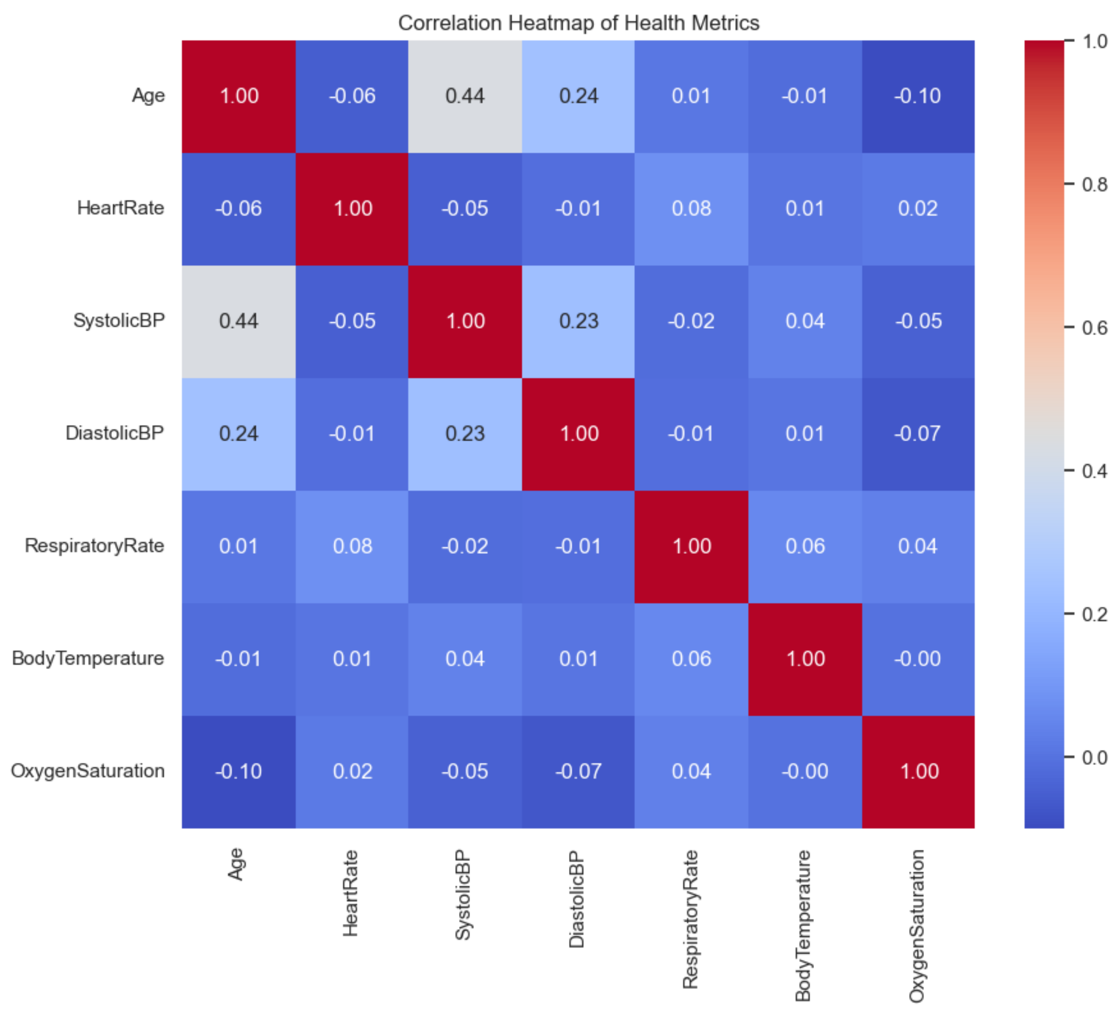
```
# Histogram for Systolic Blood Pressure
sns.histplot(health_data['SystolicBP'], color="skyblue", kde=True,␣
 ↪stat='density', bins=20, ax=axes[0])
axes[0].set_title('Systolic Blood Pressure Distribution')
axes[0].set_xlabel('Systolic BP (mmHg)')
axes[0].set_ylabel('Density')

# Histogram for Diastolic Blood Pressure
sns.histplot(health_data['DiastolicBP'], color="red", kde=True, stat='density',␣
 ↪bins=20, ax=axes[1])
axes[1].set_title('Diastolic Blood Pressure Distribution')
axes[1].set_xlabel('Diastolic BP (mmHg)')
axes[1].set_ylabel('Density')

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



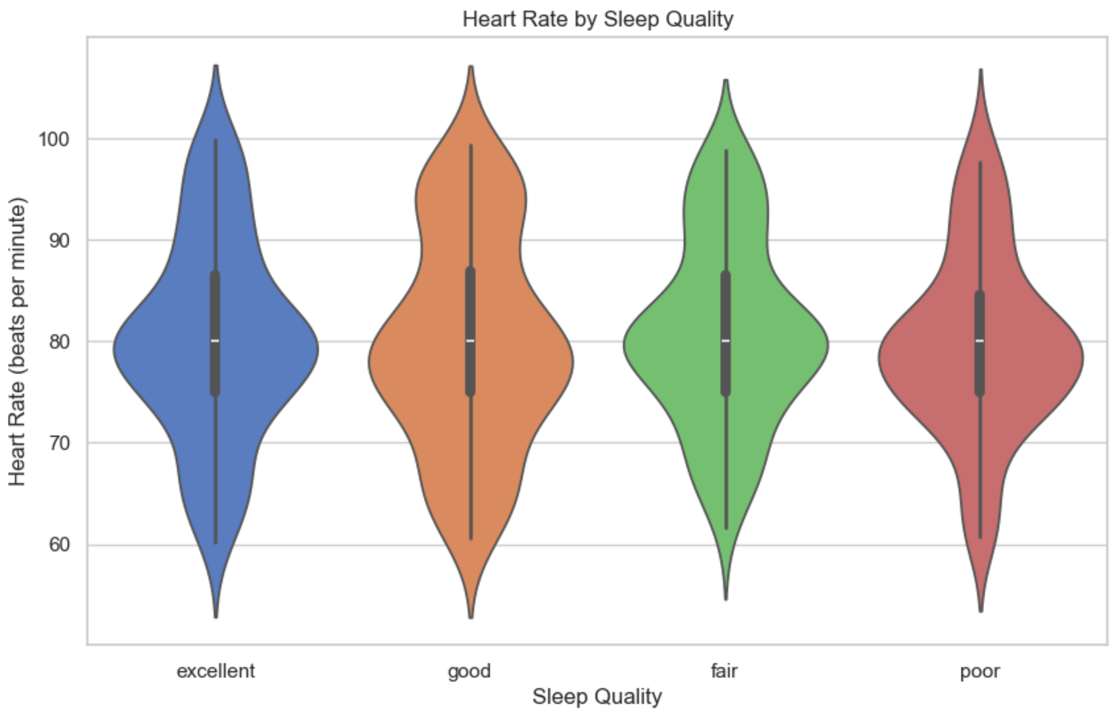
[17]:
```
# 11. Boxplot of Blood Pressure Categories
plt.figure(figsize=(10, 6))
sns.countplot(x='BPCategory', data=health_data)
plt.title('Distribution of Blood Pressure Categories')
plt.xlabel('Blood Pressure Category')
plt.ylabel('Number of Patients')
plt.show()
```

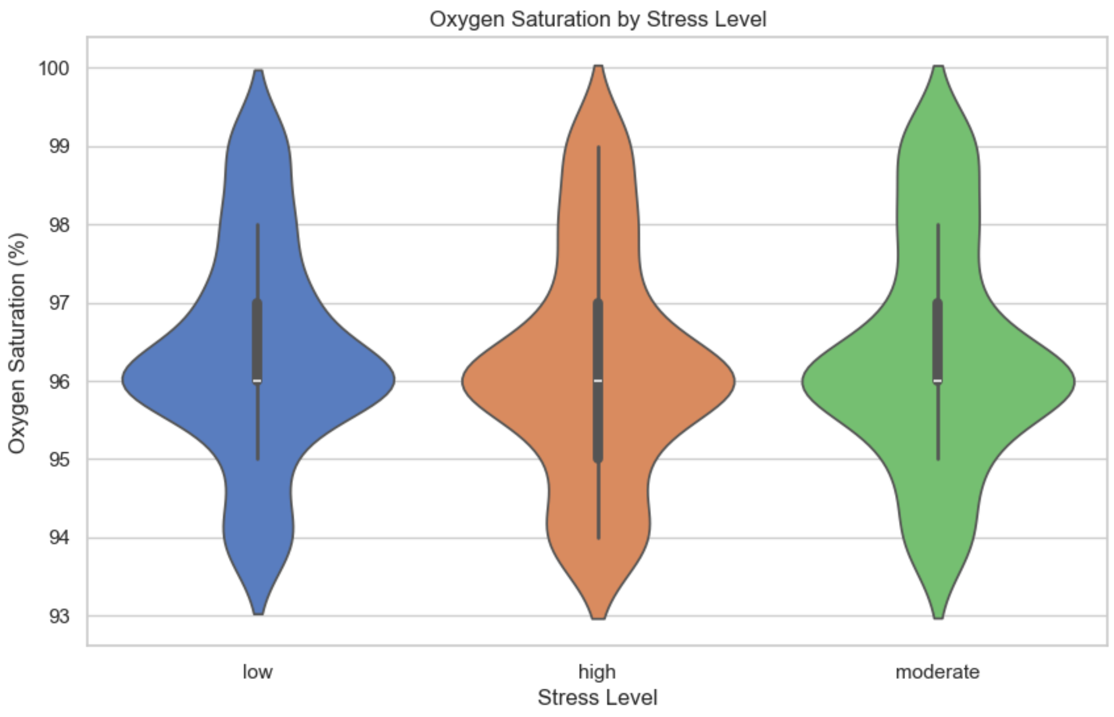


```
[18]: # 12. Correlation Heatmap
correlation_matrix = health_data[['Age', 'HeartRate', 'SystolicBP',␣
 ↪'DiastolicBP', 'RespiratoryRate', 'BodyTemperature', 'OxygenSaturation']].
 ↪corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Heatmap of Health Metrics')
plt.show()
```

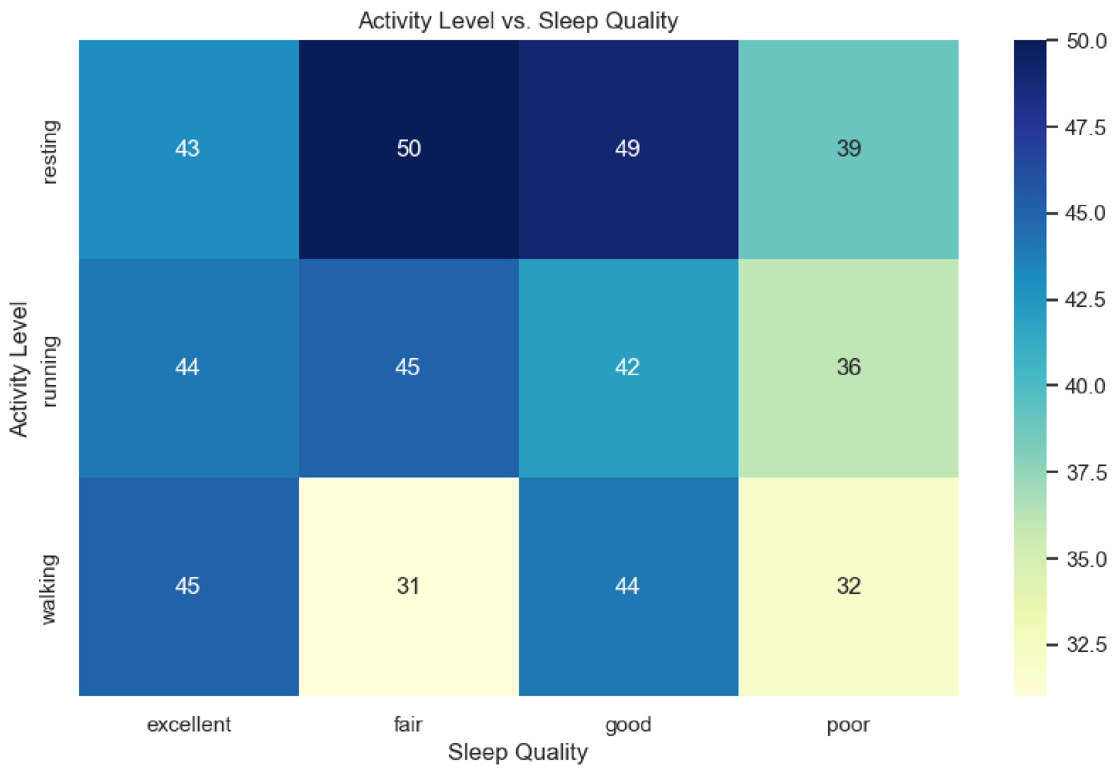


[19]:
```
# 13. Violin Plot for Heart Rate by Sleep Quality
plt.figure(figsize=(10, 6))
sns.violinplot(x='SleepQuality', y='HeartRate', data=health_data,␣
↪hue='SleepQuality', palette='muted', legend=False)
plt.title('Heart Rate by Sleep Quality')
plt.xlabel('Sleep Quality')
plt.ylabel('Heart Rate (beats per minute)')
plt.show()

# 14. Violin Plot for Oxygen Saturation by Stress Level
plt.figure(figsize=(10, 6))
sns.violinplot(x='StressLevel', y='OxygenSaturation', data=health_data,␣
↪hue='StressLevel', palette='muted', legend=False)
plt.title('Oxygen Saturation by Stress Level')
plt.xlabel('Stress Level')
plt.ylabel('Oxygen Saturation (%)')
plt.show()
```

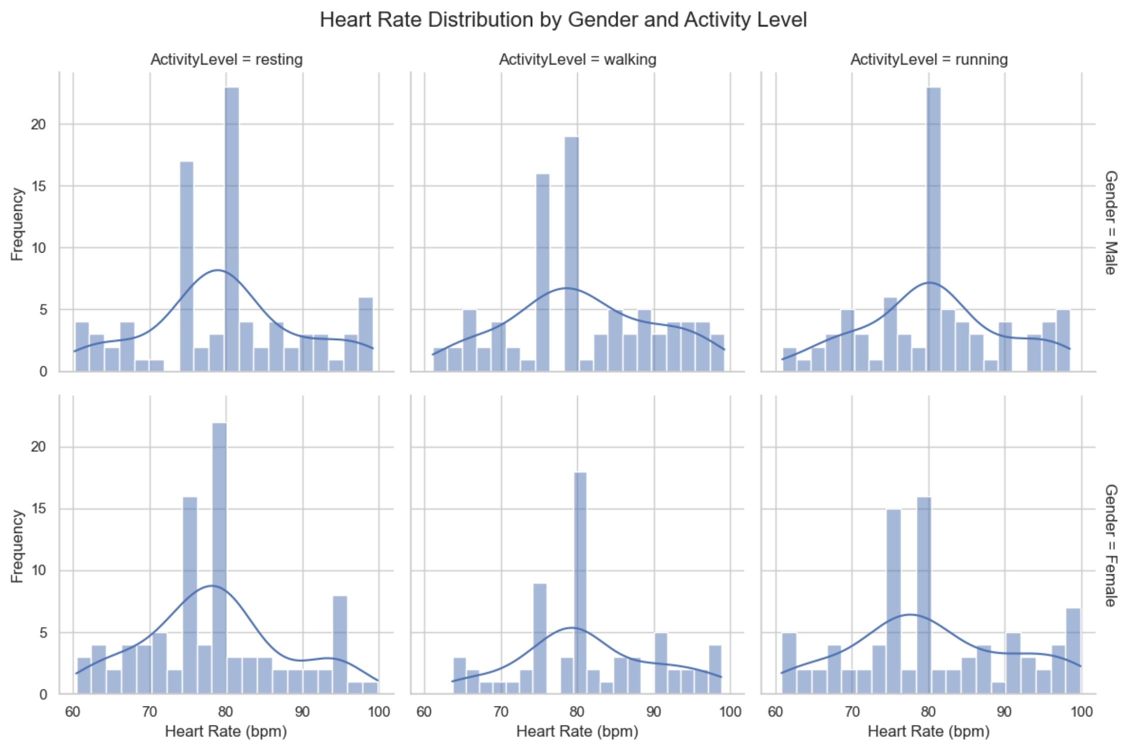

Heart Rate by Sleep Quality
Heart Rate (beats per minute)
100
90
80
70
60
excellent
good
fair
poor
Sleep Quality


Oxygen Saturation by Stress Level
Oxygen Saturation (%)
100
99
98
97
96
95
94
93
low
high
moderate
Stress Level

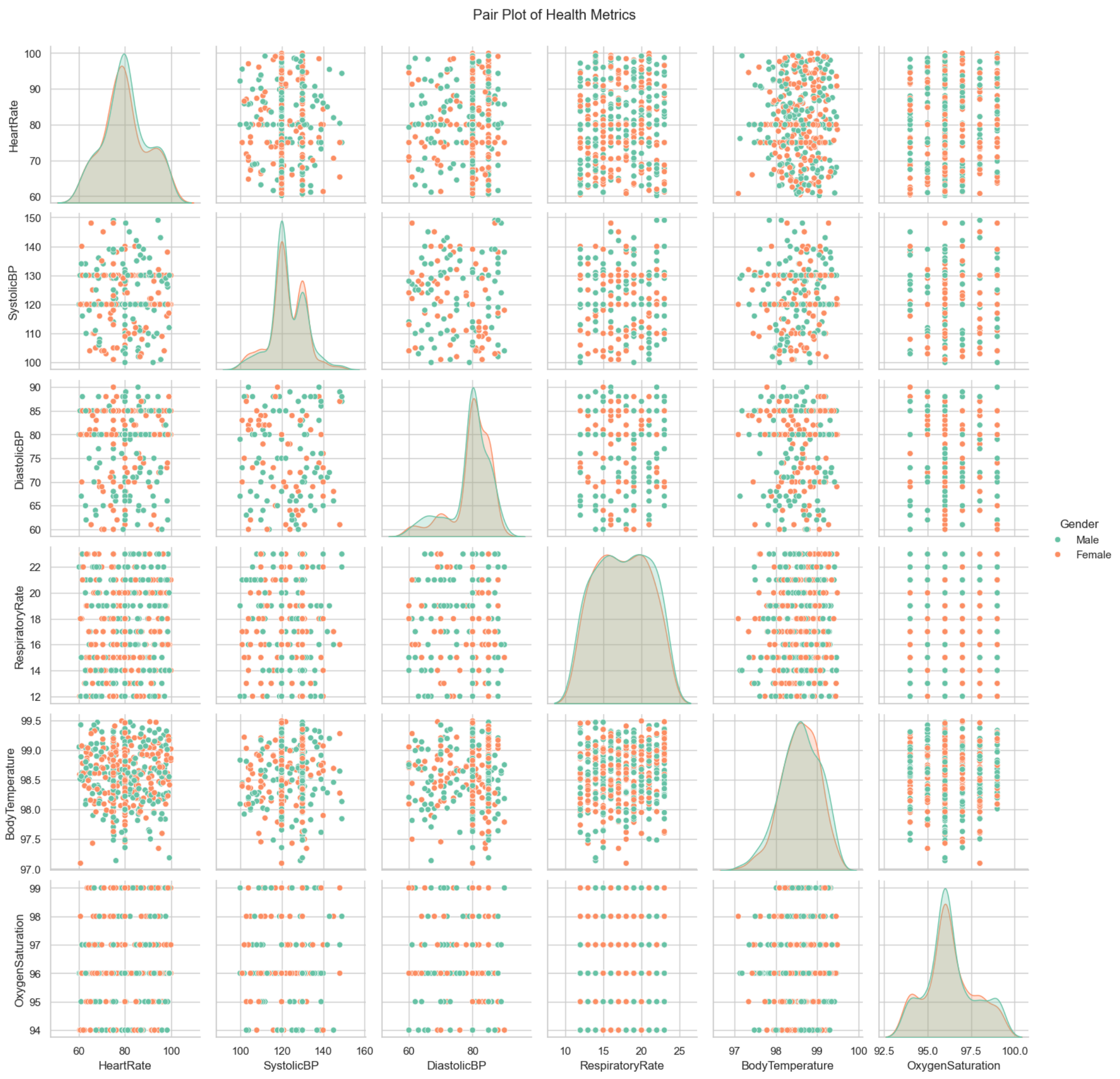
[20]:
```
# 15. Heatmap of Activity Level vs. Sleep Quality
activity_sleep_counts = health_data.groupby(['ActivityLevel', 'SleepQuality']).
  ↪size().unstack()
plt.figure(figsize=(10, 6))
sns.heatmap(activity_sleep_counts, annot=True, fmt="d", cmap='YlGnBu',␣
  ↪cbar=True)
plt.title('Activity Level vs. Sleep Quality')
plt.xlabel('Sleep Quality')
plt.ylabel('Activity Level')
plt.show()
```



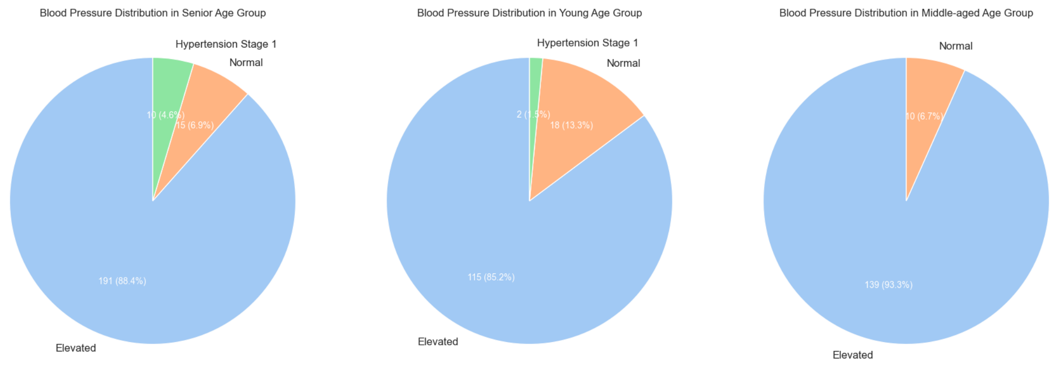
[21]:
```
# 16. Faceted Density Plots for Heart Rate by Gender and Activity Level
g = sns.FacetGrid(health_data, row='Gender', col='ActivityLevel',␣
  ↪margin_titles=True, height=4)
g.map(sns.histplot, 'HeartRate', bins=20, kde=True)
# Set titles and labels
g.fig.subplots_adjust(top=0.9)
g.fig.suptitle('Heart Rate Distribution by Gender and Activity Level',␣
  ↪fontsize=16)
# Set x and y axis labels for the entire grid
g.set_axis_labels('Heart Rate (bpm)', 'Frequency')
plt.show()
```



```
[22]: # 17. Pair Plot of Continuous Variables
continuous_vars = ['HeartRate', 'SystolicBP', 'DiastolicBP', 'RespiratoryRate',␣
  ↪'BodyTemperature', 'OxygenSaturation']

# Make sure the 'Gender' column is present and correctly spelled
sns.pairplot(health_data[continuous_vars + ['Gender']], diag_kind='kde',␣
  ↪markers='o', hue='Gender', palette='Set2')
plt.suptitle('Pair Plot of Health Metrics', y=1.02)
plt.show()
```



[23]:
```
# 18. Pie charts for each age group with counts and percentages
age_groups = health_data['AgeGroup'].unique()

# Set up the figure for pie charts
fig, axes = plt.subplots(1, len(age_groups), figsize=(18, 6))

for i, age_group in enumerate(age_groups):
    # Filter the data for the current age group
    group_data = health_data[health_data['AgeGroup'] == age_group]
    bp_counts = group_data['BPCategory'].value_counts()

    # Function to format the labels
    def func(pct, allval):
        absolute = int(round(pct / 100. * sum(allval)))
        return f"{absolute} ({pct:.1f}%)"

    # Create a pie chart for the blood pressure categories in this age group
    wedges, texts, autotexts = axes[i].pie(bp_counts, labels=bp_counts.index,␣
↪autopct=lambda pct: func(pct, bp_counts), startangle=90, colors=sns.
↪color_palette("pastel"))

    axes[i].set_title(f'Blood Pressure Distribution in {age_group} Age Group')

    # Improve the appearance of the texts
    for text in texts:
        text.set_fontsize(12)
    for autotext in autotexts:
        autotext.set_color('white')
        autotext.set_fontsize(10)

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


Blood Pressure Distribution in Senior Age Group
Hypertension Stage 1
Normal
10 (4.6%)
15 (6.9%)
191 (88.4%)
Elevated
Blood Pressure Distribution in Young Age Group
Hypertension Stage 1
Normal
2 (1.5%)
18 (13.3%)
115 (85.2%)
Elevated
Blood Pressure Distribution in Middle-aged Age Group
Normal
10 (6.7%)
139 (93.3%)
Elevated