```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_rem
```

```
import pandas as pd

file_path = '/content/drive/MyDrive/Sleep_Health.csv'
try:
  df = pd.read_csv(file_path)
  display(df.head())
except FileNotFoundError:
  print(f"Error: The file '{file_path}' was not found. Please make sure the file is in your Google Drive at the speci
except Exception as e:
  print(f"An error occurred: {e}")
```

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Daily Steps	Sleep Disorder
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	NaN
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	NaN
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	NaN
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea

```
display(df.describe())
```

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	6816.844920
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	1617.915679
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.000000
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	5600.000000
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	7000.000000
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	8000.000000
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	10000.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Person ID                374 non-null    int64
 1   Gender                   374 non-null    object
 2   Age                      374 non-null    int64
 3   Occupation               374 non-null    object
 4   Sleep Duration           374 non-null    float64
 5   Quality of Sleep         374 non-null    int64
 6   Physical Activity Level  374 non-null    int64
 7   Stress Level             374 non-null    int64
 8   BMI Category             374 non-null    object
 9   Blood Pressure           374 non-null    object
 10  Heart Rate               374 non-null    int64
 11  Daily Steps              374 non-null    int64
 12  Sleep Disorder           155 non-null    object
dtypes: float64(1), int64(7), object(5)
memory usage: 38.1+ KB
```

```
df.dtypes
```

	0
Person ID	int64
Gender	object
Age	int64
Occupation	object
Sleep Duration	float64
Quality of Sleep	int64
Physical Activity Level	int64
Stress Level	int64
BMI Category	object
Blood Pressure	object
Heart Rate	int64
Daily Steps	int64
Sleep Disorder	object

dtype: object

```
display(df.isnull().sum())
```

	0
Person ID	0
Gender	0
Age	0
Occupation	0
Sleep Duration	0
Quality of Sleep	0
Physical Activity Level	0
Stress Level	0
BMI Category	0
Blood Pressure	0
Heart Rate	0
Daily Steps	0
Sleep Disorder	219

dtype: int64

```
display(df.duplicated().sum())
```

```
np.int64(0)
```

```
for col in df.select_dtypes(include='object').columns:
  print(f"Unique values for column '{col}':")
  display(df[col].unique())
  print("-" * 30)
```

```
Unique values for column 'Gender':
array(['Male', 'Female'], dtype=object)
------------------------------
Unique values for column 'Occupation':
array(['Software Engineer', 'Doctor', 'Sales Representative', 'Teacher',
       'Nurse', 'Engineer', 'Accountant', 'Scientist', 'Lawyer',
       'Salesperson', 'Manager'], dtype=object)
------------------------------
Unique values for column 'BMI Category':
array(['Overweight', 'Normal', 'Obese', 'Normal Weight'], dtype=object)
------------------------------
Unique values for column 'Blood Pressure':
array(['126/83', '125/80', '140/90', '120/80', '132/87', '130/86',
       '117/76', '118/76', '128/85', '131/86', '128/84', '115/75',
       '135/88', '129/84', '130/85', '115/78', '119/77', '121/79',
       '125/82', '135/90', '122/80', '142/92', '140/95', '139/91',
       '118/75'], dtype=object)
------------------------------
Unique values for column 'Sleep Disorder':
array([nan, 'Sleep Apnea', 'Insomnia'], dtype=object)
------------------------------
```

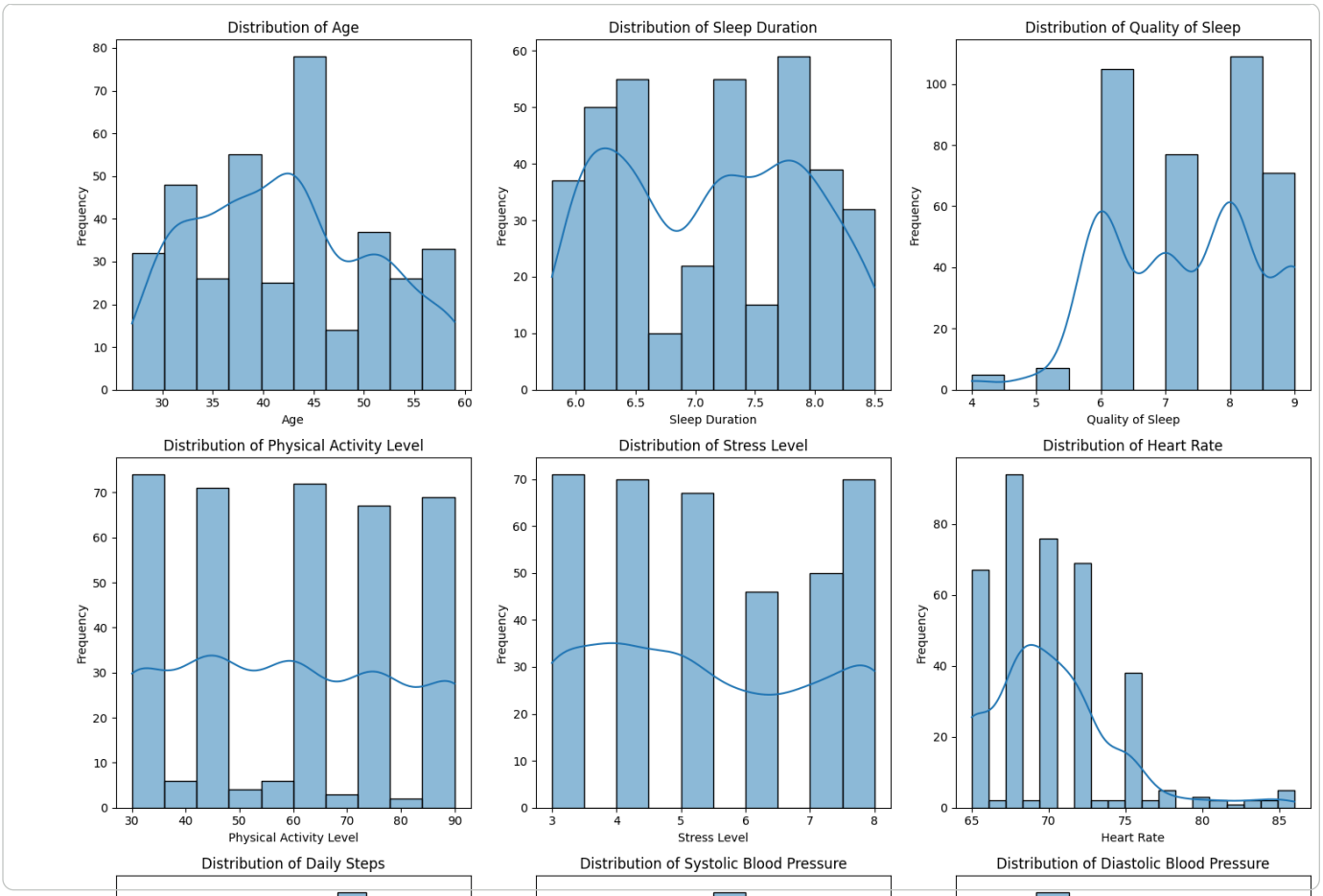
```
df[['Systolic Blood Pressure', 'Diastolic Blood Pressure']] = df['Blood Pressure'].str.split('/', expand=True).astyp
display(df[['Blood Pressure', 'Systolic Blood Pressure', 'Diastolic Blood Pressure']].head())
```

	Blood Pressure	Systolic Blood Pressure	Diastolic Blood Pressure
0	126/83	126	83
1	125/80	125	80
2	125/80	125	80
3	140/90	140	90
4	140/90	140	90

```
df['Has Sleep Disorder'] = df['Sleep Disorder'].notna().astype(int)
display(df[['Sleep Disorder', 'Has Sleep Disorder']].head())
```

	Sleep Disorder	Has Sleep Disorder
0	NaN	0
1	NaN	0
2	NaN	0
3	Sleep Apnea	1
4	Sleep Apnea	1

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

numerical_cols = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart Rat

# Determine the number of rows and columns for the subplots
n_cols = 3
n_rows = (len(numerical_cols) + n_cols - 1) // n_cols

# Create subplots for histograms
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 5))
axes = axes.flatten()

for i, col in enumerate(numerical_cols):
    sns.histplot(data=df, x=col, kde=True, ax=axes[i])
    axes[i].set_title(f'Distribution of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Frequency')

# Remove any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()

# Create subplots for box plots
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 5))
axes = axes.flatten()

for i, col in enumerate(numerical_cols):
    sns.boxplot(data=df, y=col, ax=axes[i])
    axes[i].set_title(f'Box Plot of {col}')
    axes[i].set_ylabel(col)

# Remove any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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

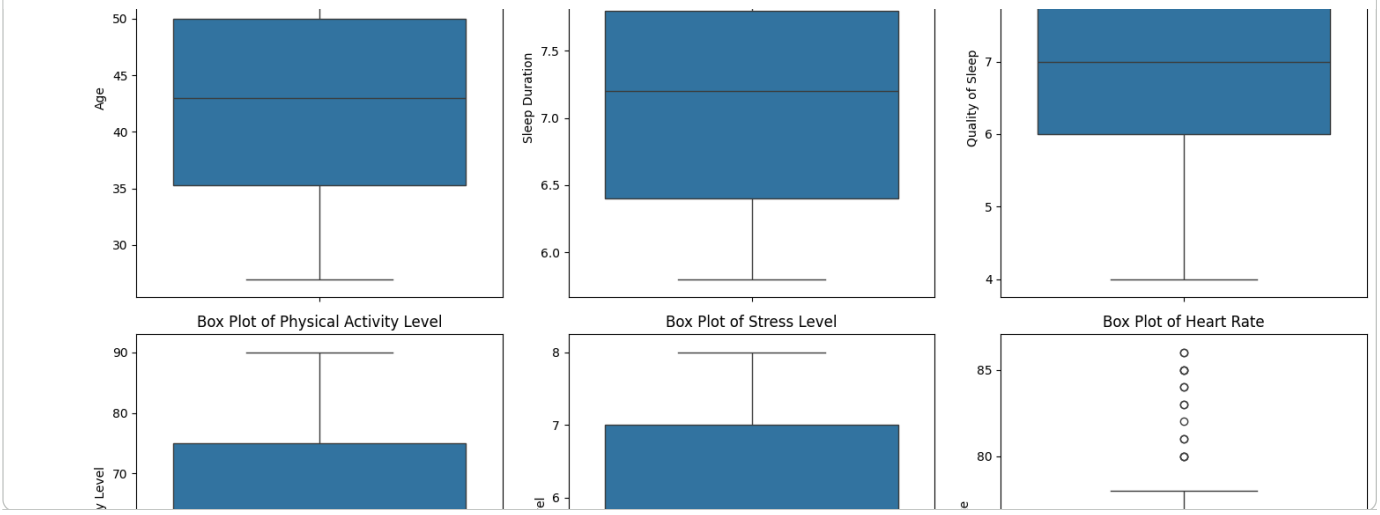


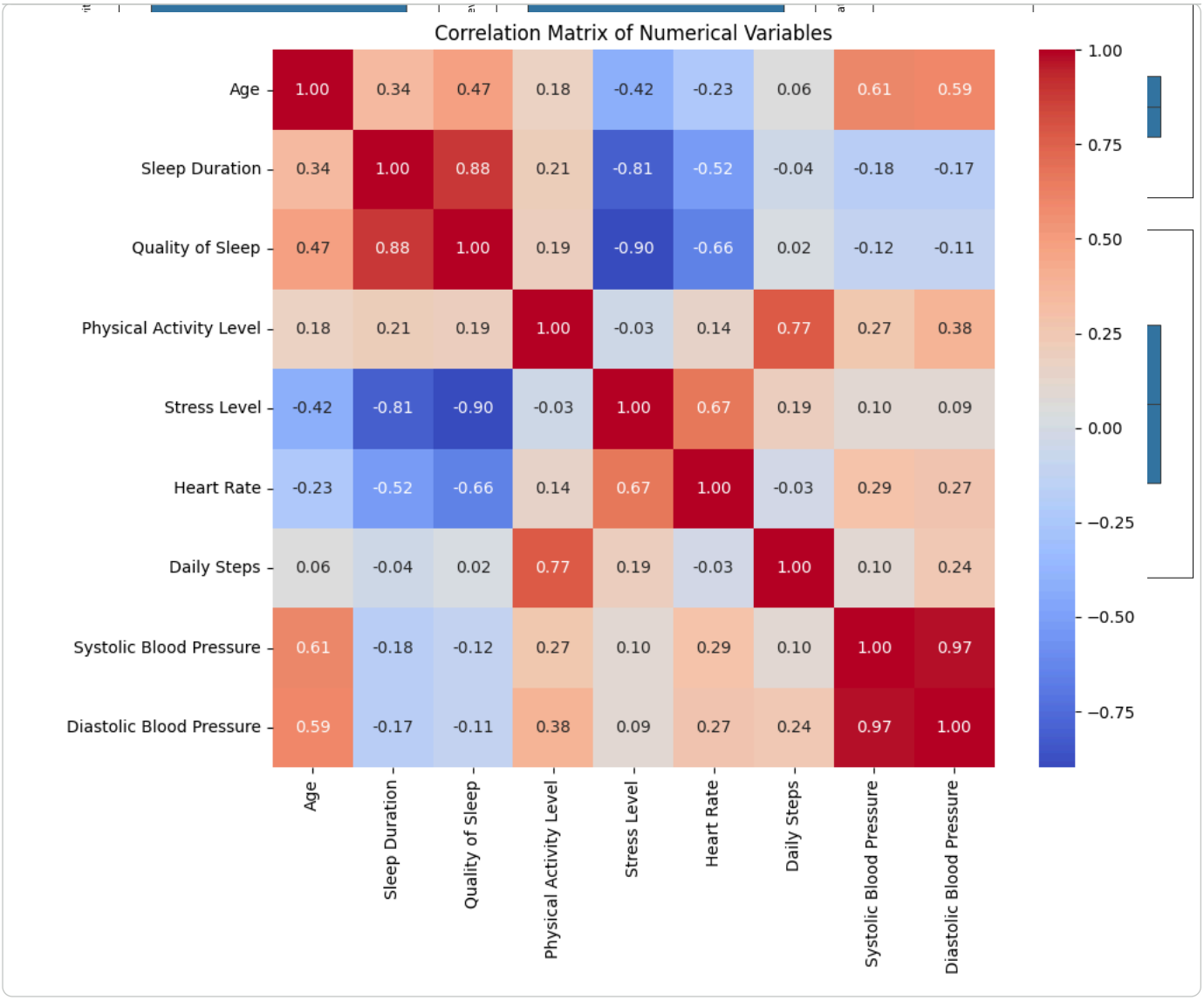
The plots above show the distribution of the numerical variables in the dataset.

- **Histograms:** The histograms display the frequency distribution of each numerical variable. They give us an idea of the shape of the distribution (e.g., normal, skewed), the central tendency, and the spread of the data.
- **Box Plots:** The box plots summarize the distribution of each numerical variable using quartiles. They show the median, interquartile range, and potential outliers. Outliers are data points that fall significantly outside the typical range of values.

By examining these plots, we can gain insights into the characteristics of each numerical variable, such as whether the data is concentrated in a particular range, if there are multiple peaks in the distribution, or if there are any extreme values that might warrant further investigation.

```
numerical_cols = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart Rat
correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Variables')
plt.show()
```





The correlation matrix you see displays the pairwise correlation coefficients between the numerical variables in your dataset.

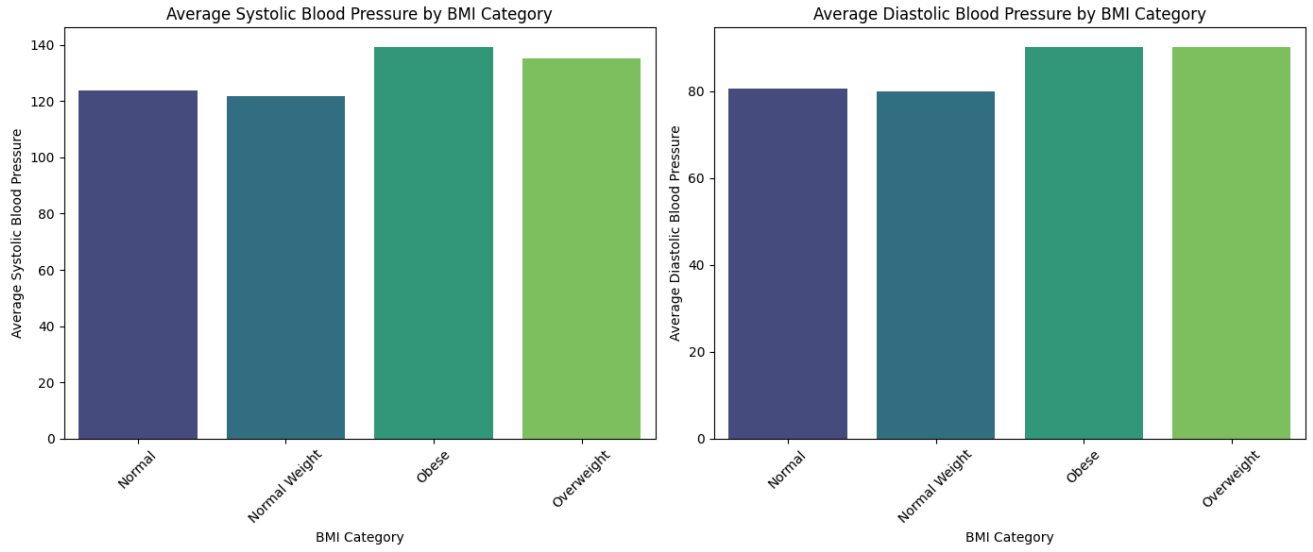
Here's a quick guide on how to interpret it:

- **Values range from -1 to 1:**
  - A value of **1** indicates a perfect positive linear correlation (as one variable increases, the other increases proportionally).
  - A value of **-1** indicates a perfect negative linear correlation (as one variable increases, the other decreases proportionally).
  - A value of **0** indicates no linear correlation.
- **The diagonal (where a variable is correlated with itself) will always be 1.**
- **The matrix is symmetrical** because the correlation between variable A and variable B is the same as the correlation between variable B and variable A.

Looking at the matrix, you can identify which variables have strong positive or negative relationships. For example, a high positive value between 'Sleep Duration' and 'Quality of Sleep' would suggest that longer sleep duration is associated with higher quality sleep. Conversely, a strong negative value between 'Sleep Duration' and 'Stress Level' would suggest that higher stress levels are associated with shorter sleep duration.

```
bmi_bp = df.groupby('BMI Category')[['Systolic Blood Pressure', 'Diastolic Blood Pressure']].mean().reset_index()

fig, axes = plt.subplots(1, 2, figsize=(14, 6))

sns.barplot(data=bmi_bp, x='BMI Category', y='Systolic Blood Pressure', ax=axes[0], palette='viridis')
axes[0].set_title('Average Systolic Blood Pressure by BMI Category')
axes[0].set_xlabel('BMI Category')
axes[0].set_ylabel('Average Systolic Blood Pressure')
axes[0].tick_params(axis='x', rotation=45)

sns.barplot(data=bmi_bp, x='BMI Category', y='Diastolic Blood Pressure', ax=axes[1], palette='viridis')
axes[1].set_title('Average Diastolic Blood Pressure by BMI Category')
axes[1].set_xlabel('BMI Category')
axes[1].set_ylabel('Average Diastolic Blood Pressure')
axes[1].tick_params(axis='x', rotation=45)

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

```
/tmp/ipython-input-988871433.py:5: FutureWarning:

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

  sns.barplot(data=bmi_bp, x='BMI Category', y='Systolic Blood Pressure', ax=axes[0], palette='viridis')
/tmp/ipython-input-988871433.py:11: FutureWarning:

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

  sns.barplot(data=bmi_bp, x='BMI Category', y='Diastolic Blood Pressure', ax=axes[1], palette='viridis')
```



The bar plots above illustrate the average systolic and diastolic blood pressure for each BMI category.

Here's what we can observe from these plots:

- **Systolic Blood Pressure:** The bar plot on the left shows the average systolic blood pressure for each BMI category. It appears that individuals in the **Obese** category tend to have the highest average systolic blood pressure, followed by the **Overweight** category. The **Normal** and **Normal Weight** categories have lower average systolic blood pressure.
- **Diastolic Blood Pressure:** Similarly, the bar plot on the right displays the average diastolic blood pressure by BMI category. This plot also suggests that the **Obese** and **Overweight** categories have higher average diastolic blood pressure compared to the **Normal** and **Normal Weight** categories.

These visualizations suggest a potential relationship between BMI category and blood pressure, where higher BMI categories are associated with higher average blood pressure.

```
gender_grouped = df.groupby('Gender')[['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'Stre
print("Mean of numerical columns grouped by Gender:")
display(gender_grouped)

occupation_grouped = df.groupby('Occupation')[['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level
print("\nMean of numerical columns grouped by Occupation:")
display(occupation_grouped)

bmi_grouped = df.groupby('BMI Category')[['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'S
print("\nMean of numerical columns grouped by BMI Category:")
display(bmi_grouped)
```

```
Mean of numerical columns grouped by Gender:
```

Gender	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps	Systolic Blood Pressure	Diastolic Blood Pressure
Female	47.405405	7.229730	7.664865	59.140541	4.675676	69.259459	6840.540541	130.200000	86.318919
Male	37.074074	7.036508	6.968254	59.201058	6.079365	71.052910	6793.650794	126.941799	83.015873

```
Mean of numerical columns grouped by Occupation:
```

Occupation	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps	Systolic Blood Pressure	Diastolic Blood Pressure
Accountant	39.621622	7.113514	7.891892	58.108108	4.594595	68.864865	6881.081081	117.729730	76.918919
Doctor	32.676056	6.970423	6.647887	55.352113	6.732394	71.521127	6808.450704	123.000000	80.507042
Engineer	46.587302	7.987302	8.412698	51.857143	3.888889	67.190476	5980.952381	125.904762	81.380952
Lawyer	39.425532	7.410638	7.893617	70.425532	5.063830	69.638298	7661.702128	129.957447	85.000000
Manager	45.000000	6.900000	7.000000	55.000000	5.000000	75.000000	5500.000000	125.000000	82.000000
Nurse	51.794521	7.063014	7.369863	78.589041	5.547945	72.000000	8057.534247	138.520548	93.726027
Sales Representative	28.000000	5.900000	4.000000	30.000000	8.000000	85.000000	3000.000000	140.000000	90.000000
Salesperson	43.531250	6.403125	6.000000	45.000000	7.000000	72.000000	6000.000000	130.000000	85.000000
Scientist	33.500000	6.000000	5.000000	41.000000	7.000000	78.500000	5350.000000	129.500000	85.500000
Software Engineer	31.250000	6.750000	6.500000	48.000000	6.000000	75.500000	5800.000000	126.500000	83.250000
Teacher	41.725000	6.690000	6.975000	45.625000	4.525000	67.225000	5957.500000	131.225000	86.900000

```
Mean of numerical columns grouped by BMI Category:
```

BMI Category	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps	Systolic Blood Pressure	Diastolic Blood Pressure
Normal	38.482051	7.393846	7.661538	57.692308	5.128205	68.728205	6887.179487	123.820513	80.666667
Normal Weight	38.380952	7.333333	7.428571	60.333333	5.190476	71.285714	6766.666667	121.619048	80.000000
Obese	38.000000	6.960000	6.400000	55.000000	5.700000	84.300000	3350.000000	139.200000	90.200000
Overweight	47.885135	6.770270	6.898649	61.236486	5.729730	70.945946	6965.540541	135.054054	90.182432

Next steps: Generate code with gender_grouped | New interactive sheet | Generate code with occupation_grouped | New interactive sheet | G

```
sleep_disorder_grouped = df.groupby('Has Sleep Disorder')[['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Ac
print("Mean of numerical columns grouped by 'Has Sleep Disorder':")
display(sleep_disorder_grouped)
```

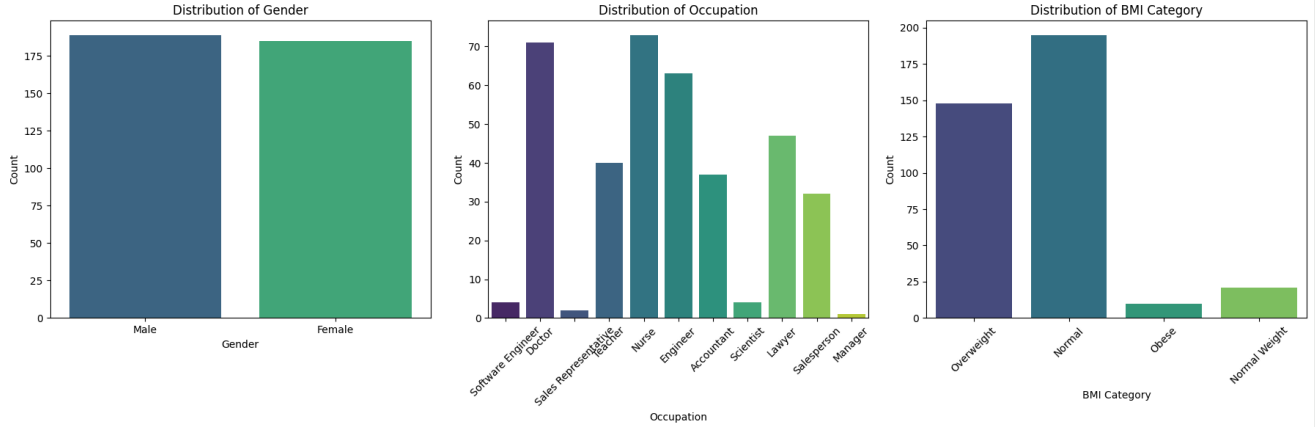
Mean of numerical columns grouped by 'Has Sleep Disorder':

Has Sleep Disorder	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps	Systolic Blood Pressure	Diastolic Blood Pressure
0	39.036530	7.358447	7.625571	57.949772	5.114155	69.018265	6852.968037	124.045662	81.000000
1	46.632258	6.812258	6.870968	60.896774	5.767742	71.787097	6765.806452	134.922581	89.806452

Next steps: Generate code with sleep_disorder_grouped | New interactive sheet

```
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

sns.countplot(data=df, x='Gender', ax=axes[0], palette='viridis')
axes[0].set_title('Distribution of Gender')
axes[0].set_xlabel('Gender')
axes[0].set_ylabel('Count')

sns.countplot(data=df, x='Occupation', ax=axes[1], palette='viridis')
axes[1].set_title('Distribution of Occupation')
axes[1].set_xlabel('Occupation')
axes[1].set_ylabel('Count')
axes[1].tick_params(axis='x', rotation=45)

sns.countplot(data=df, x='BMI Category', ax=axes[2], palette='viridis')
axes[2].set_title('Distribution of BMI Category')
axes[2].set_xlabel('BMI Category')
axes[2].set_ylabel('Count')
axes[2].tick_params(axis='x', rotation=45)

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

```
/tmp/ipython-input-868598120.py:3: FutureWarning:

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

  sns.countplot(data=df, x='Gender', ax=axes[0], palette='viridis')
/tmp/ipython-input-868598120.py:8: FutureWarning:

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

  sns.countplot(data=df, x='Occupation', ax=axes[1], palette='viridis')
/tmp/ipython-input-868598120.py:14: FutureWarning:

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

  sns.countplot(data=df, x='BMI Category', ax=axes[2], palette='viridis')
```



The violin plots above show the distribution of Sleep Duration, Quality of Sleep, and Stress Level across different occupations.

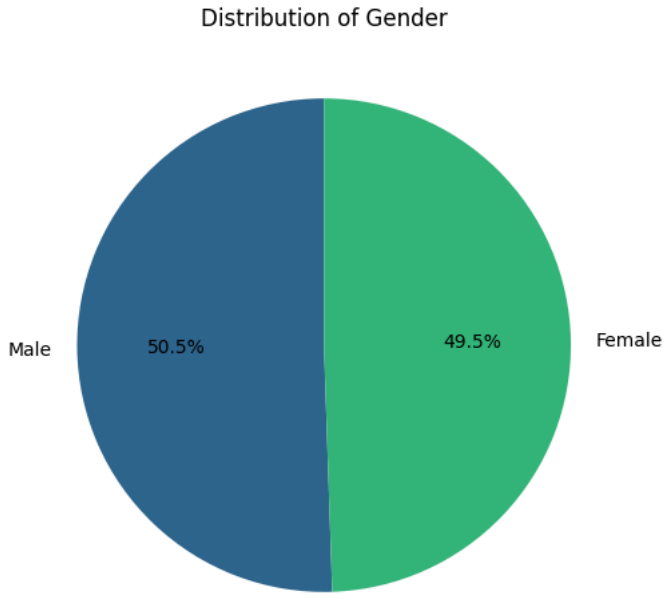
- **Sleep Duration by Occupation:** This plot reveals the range and distribution of sleep duration for each occupation. You can observe variations in typical sleep hours and the spread of data within each profession. Some occupations might show a tighter distribution

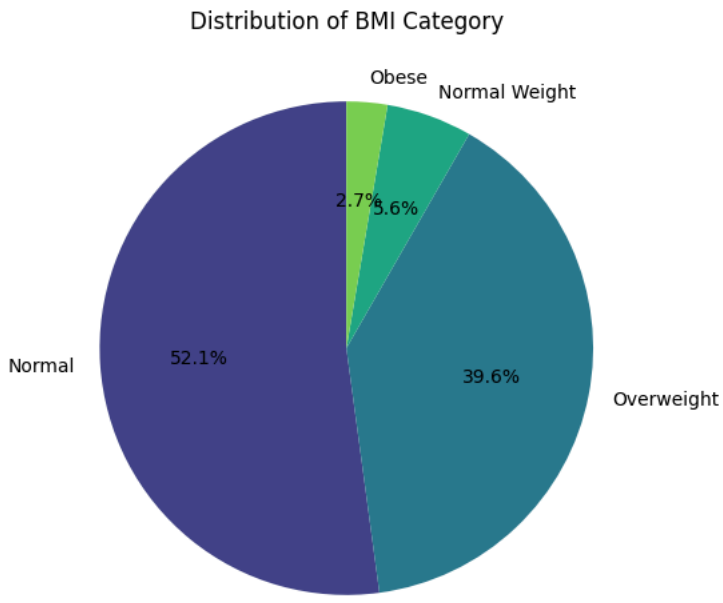
around the mean, while others might have a wider range of sleep durations.

- **Quality of Sleep by Occupation:** This plot illustrates how the quality of sleep varies among different occupations. You can see if certain jobs are associated with generally higher or lower sleep quality and the variability within each group.
- **Stress Level by Occupation:** This plot displays the distribution of stress levels for each occupation. It highlights which professions tend to have higher or lower stress levels and the consistency of stress within those groups.

By examining these plots, you can gain insights into potential relationships between occupation and sleep patterns or stress levels. For instance, you might observe if high-stress occupations tend to have shorter sleep durations or lower sleep quality.

```
gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90, colors=sns.color_palette('viridis
plt.title('Distribution of Gender')
plt.show()

bmi_counts = df['BMI Category'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(bmi_counts, labels=bmi_counts.index, autopct='%1.1f%%', startangle=90, colors=sns.color_palette('viridis', ler
plt.title('Distribution of BMI Category')
plt.show()
```





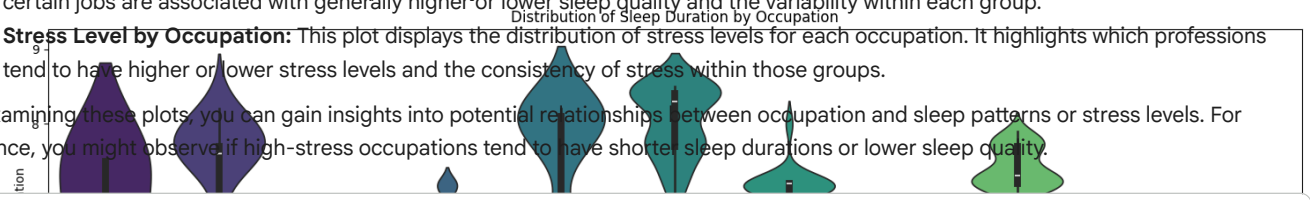
The pie charts above show the distribution of Gender and BMI Category in the dataset.

- **Distribution of Gender:** This pie chart displays the proportion of Male and Female individuals in the dataset. It provides a clear visual representation of the gender balance.
- **Distribution of BMI Category:** This pie chart shows the proportion of individuals in each BMI Category (Normal, Overweight, Obese, and Normal Weight). It helps to understand the prevalence of different weight categories in the dataset.

These pie charts provide a quick overview of the demographic breakdown of the dataset in terms of gender and BMI.

```
fig, axes = plt.subplots(3, 1, figsize=(15, 18))

sns.violinplot(data=df, x='Occupation', y='Sleep Duration', ax=axes[0], palette='viridis')
axes[0].set_title('Distribution of Sleep Duration by Occupation')
axes[0].set_xlabel('Occupation')
axes[0].set_ylabel('Sleep Duration')
axes[0].tick_params(axis='x', rotation=45)

sns.violinplot(data=df, x='Occupation', y='Quality of Sleep', ax=axes[1], palette='viridis')
axes[1].set_title('Distribution of Quality of Sleep by Occupation')
axes[1].set_xlabel('Occupation')
axes[1].set_ylabel('Quality of Sleep')
axes[1].tick_params(axis='x', rotation=45)

sns.violinplot(data=df, x='Occupation', y='Stress Level', ax=axes[2], palette='viridis')
axes[2].set_title('Distribution of Stress Level by Occupation')
axes[2].set_xlabel('Occupation')
axes[2].set_ylabel('Stress Level')
axes[2].tick_params(axis='x', rotation=45)

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

```
/tmp/ipython-input-580456080.py:3: FutureWarning:

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

  sns.violinplot(data=df, x='Occupation', y='Sleep Duration', ax=axes[0], palette='viridis')
/tmp/ipython-input-580456080.py:9: FutureWarning:

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

  sns.violinplot(data=df, x='Occupation', y='Quality of Sleep', ax=axes[1], palette='viridis')
/tmp/ipython-input-580456080.py:15: FutureWarning:

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

  sns.violinplot(data=df, x='Occupation', y='Stress Level', ax=axes[2], palette='viridis')
```

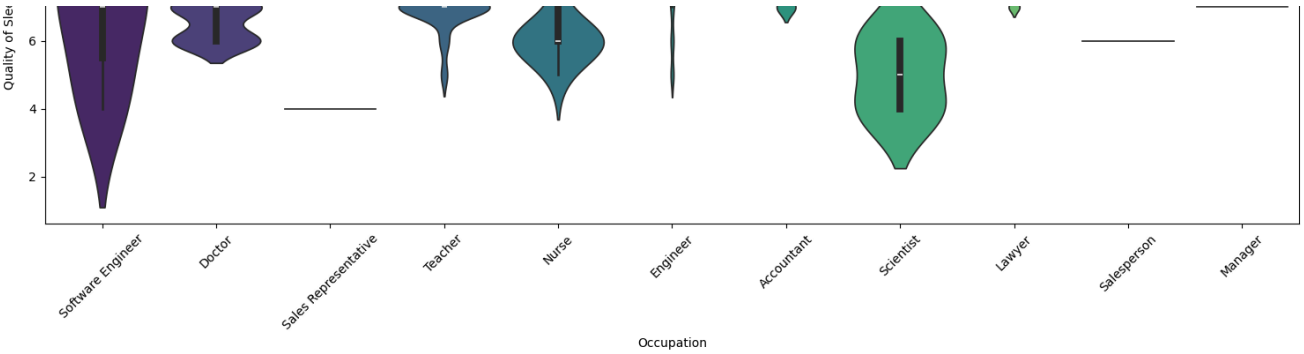


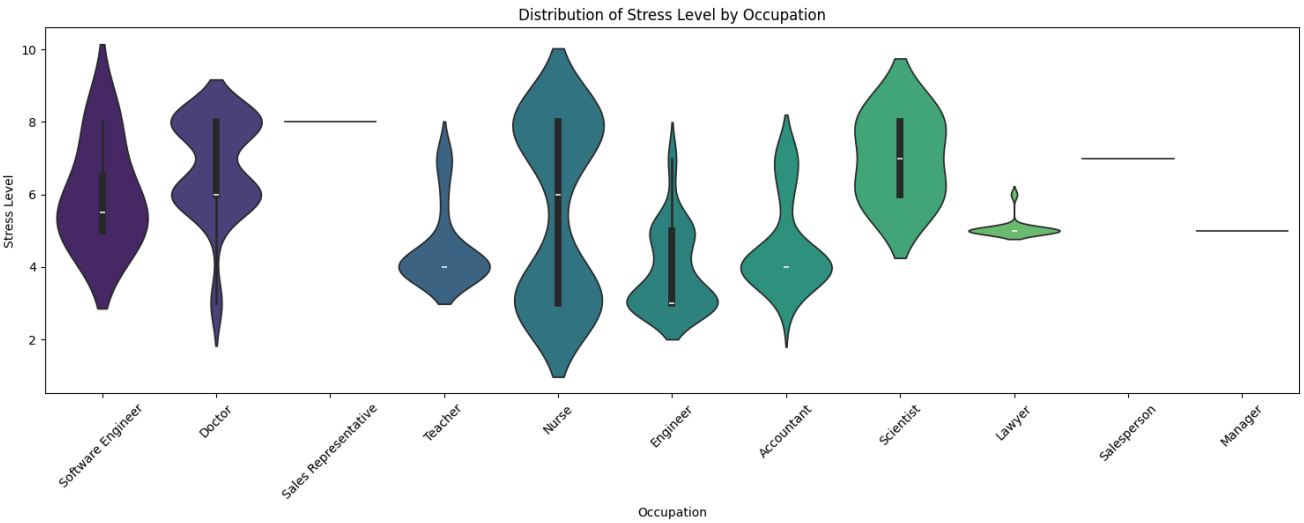
The violin plots above show the distribution of Sleep Duration, Quality of Sleep, and Stress Level across different occupations.

- **Sleep Duration by Occupation:** This plot reveals the range and distribution of sleep duration for each occupation. You can observe variations in typical sleep hours and the spread of data within each profession. Some occupations might show a tighter distribution around the mean, while others might have a wider range of sleep durations.
- **Quality of Sleep by Occupation:** This plot illustrates how the quality of sleep varies among different occupations. You can see if certain jobs are associated with generally higher or lower sleep quality and the variability within each group.
- **Stress Level by Occupation:** This plot displays the distribution of stress levels for each occupation. It highlights which professions tend to have higher or lower stress levels and the consistency of stress within those groups.

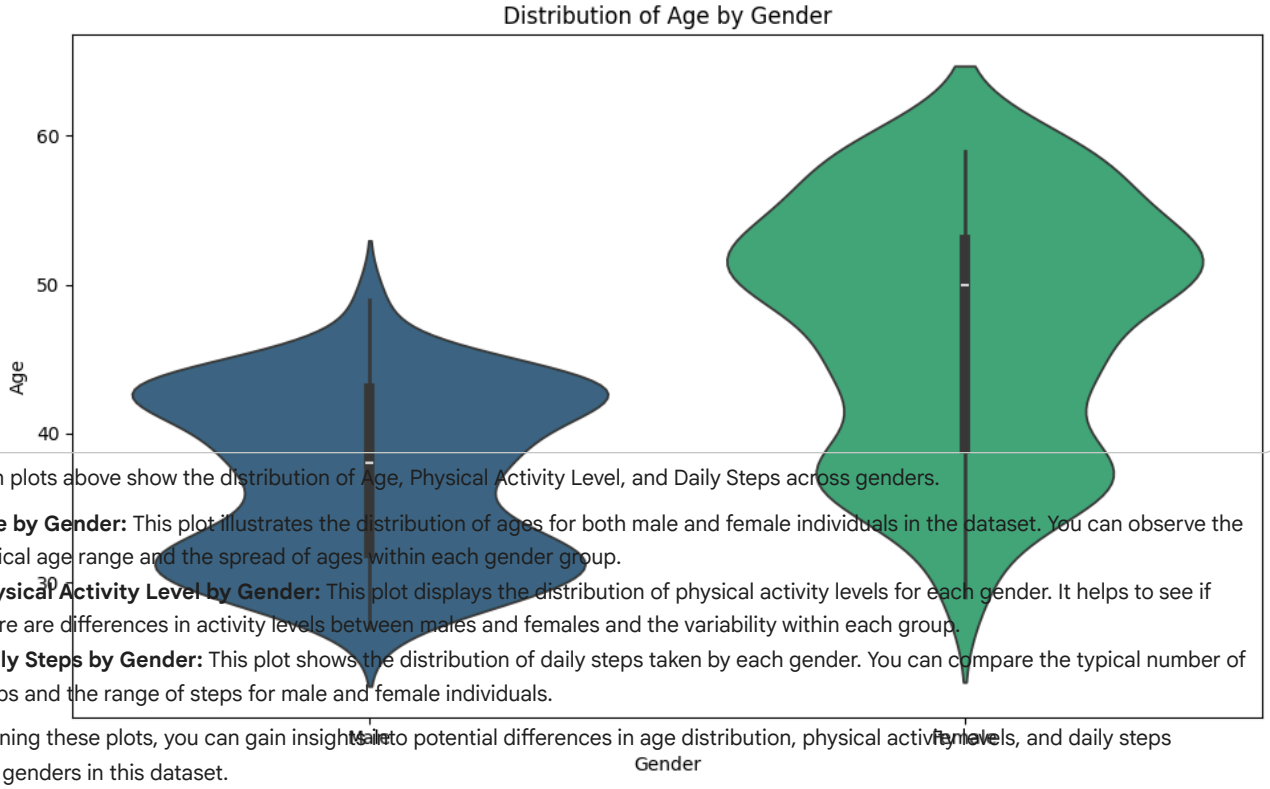
By examining these plots, you can gain insights into potential relationships between occupation and sleep patterns or stress levels. For instance, you might observe if high-stress occupations tend to have shorter sleep durations or lower sleep quality.

```
fig, axes = plt.subplots(3, 1, figsize=(10, 18))

sns.violinplot(data=df, x='Gender', y='Age', ax=axes[0], palette='viridis')
axes[0].set_title('Distribution of Age by Gender')
axes[0].set_xlabel('Gender')
axes[0].set_ylabel('Age')

sns.violinplot(data=df, x='Gender', y='Physical Activity Level', ax=axes[1], palette='viridis')
axes[1].set_title('Distribution of Physical Activity Level by Gender')
axes[1].set_xlabel('Gender')
axes[1].set_ylabel('Physical Activity Level')

sns.violinplot(data=df, x='Gender', y='Daily Steps', ax=axes[2], palette='viridis')
axes[2].set_title('Distribution of Daily Steps by Gender')
axes[2].set_xlabel('Gender')
axes[2].set_ylabel('Daily Steps')

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





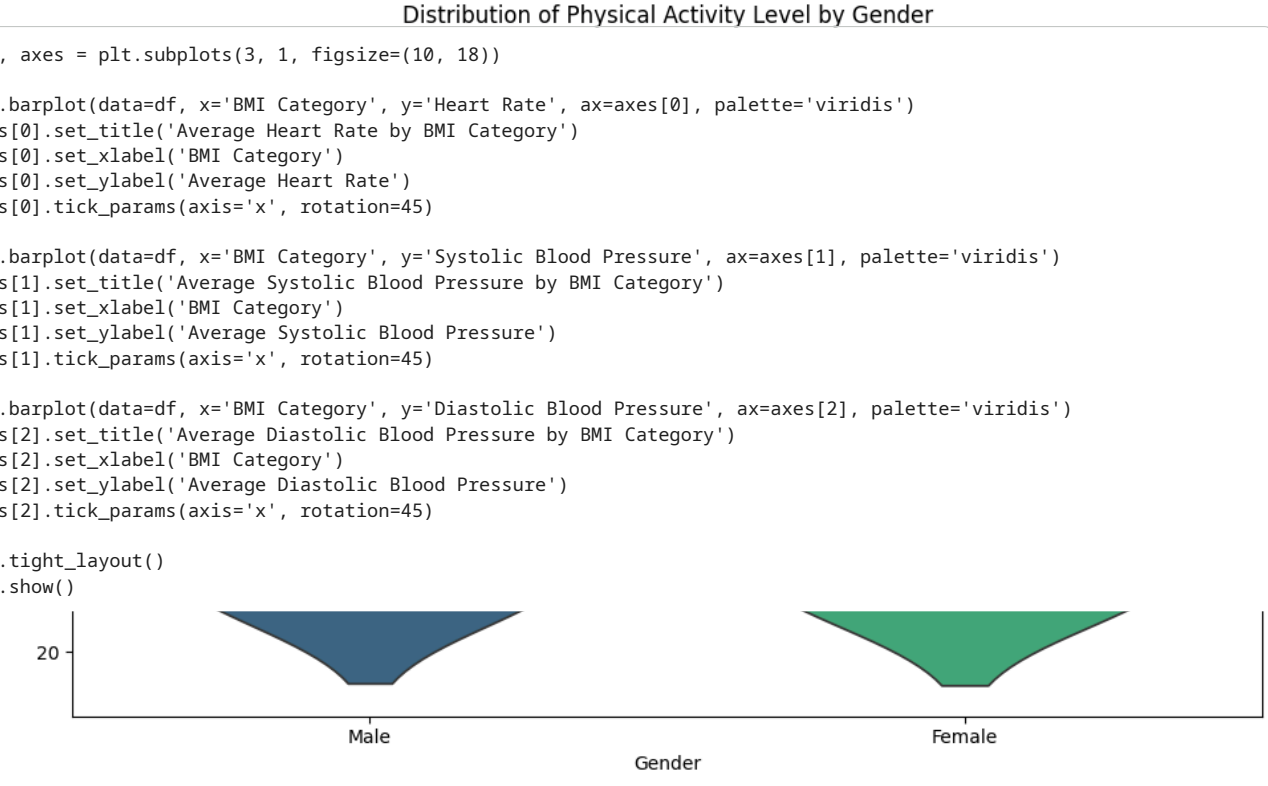
```
/tmp/ipython-input-3684604331.py:3: FutureWarning:

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

  sns.violinplot(data=df, x='Gender', y='Age', ax=axes[0], palette='viridis')
/tmp/ipython-input-3684604331.py:8: FutureWarning:

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

  sns.violinplot(data=df, x='Gender', y='Physical Activity Level', ax=axes[1], palette='viridis')
/tmp/ipython-input-3684604331.py:13: FutureWarning:

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

  sns.violinplot(data=df, x='Gender', y='Daily Steps', ax=axes[2], palette='viridis')
```



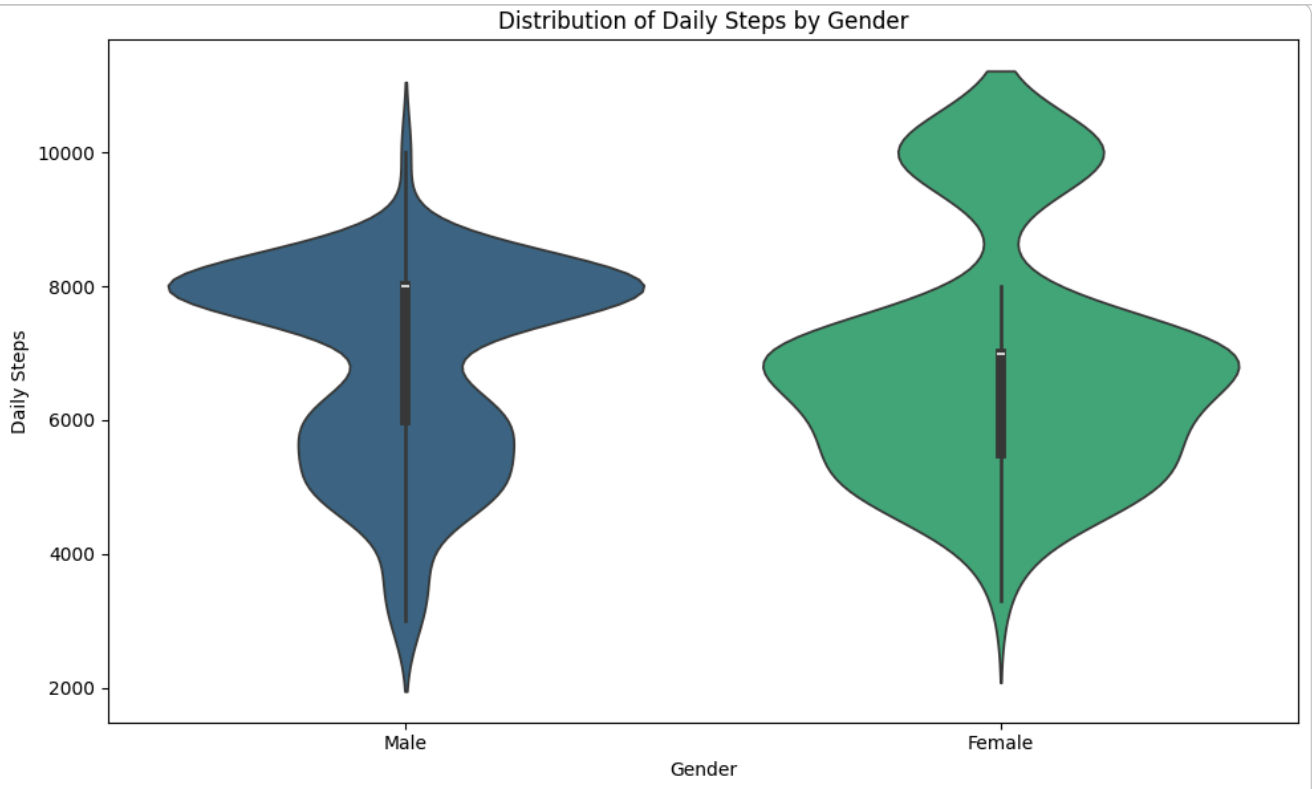
The violin plots above show the distribution of Age, Physical Activity Level, and Daily Steps across genders.

- **Age by Gender:** This plot illustrates the distribution of ages for both male and female individuals in the dataset. You can observe the typical age range and the spread of ages within each gender group.
- **Physical Activity Level by Gender:** This plot displays the distribution of physical activity levels for each gender. It helps to see if there are differences in activity levels between males and females and the variability within each group.
- **Daily Steps by Gender:** This plot shows the distribution of daily steps taken by each gender. You can compare the typical number of steps and the range of steps for male and female individuals.

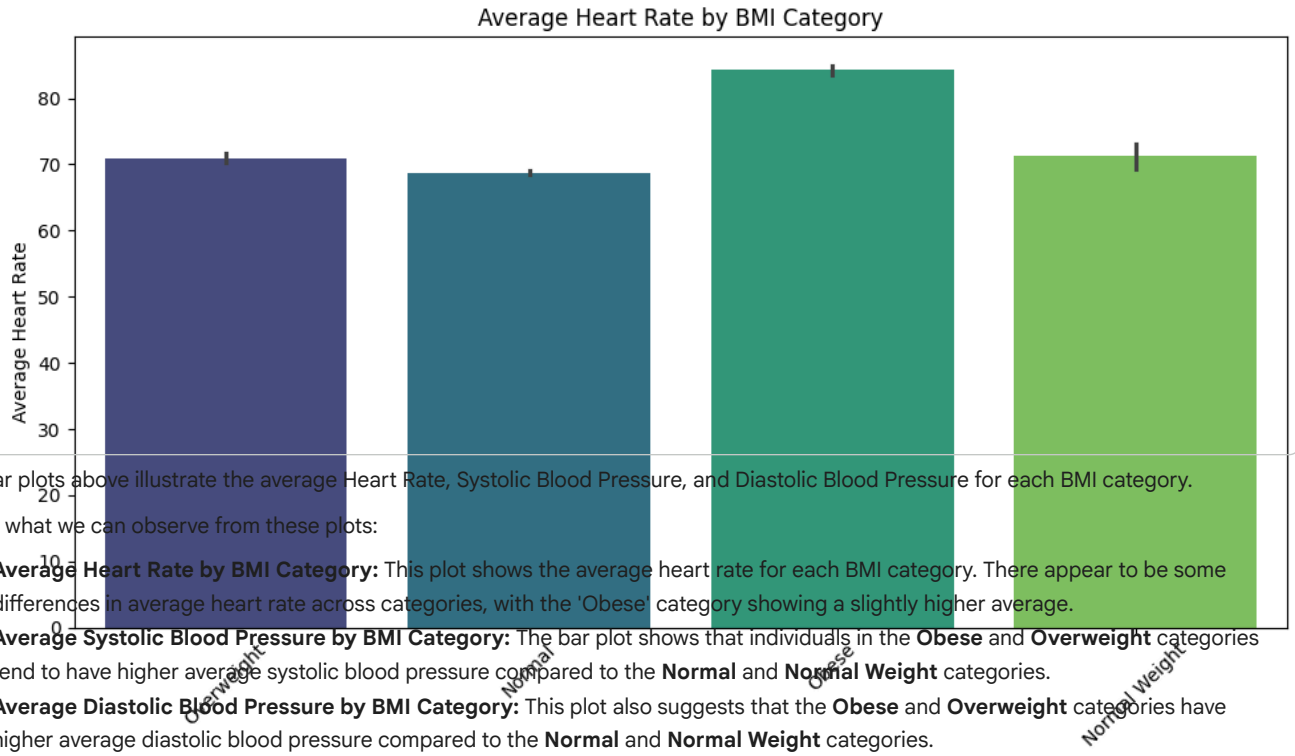
By examining these plots, you can gain insights into potential differences in age distribution, physical activity levels, and daily steps between genders in this dataset.




```
fig, axes = plt.subplots(3, 1, figsize=(10, 18))

sns.barplot(data=df, x='BMI Category', y='Heart Rate', ax=axes[0], palette='viridis')
axes[0].set_title('Average Heart Rate by BMI Category')
axes[0].set_xlabel('BMI Category')
axes[0].set_ylabel('Average Heart Rate')
axes[0].tick_params(axis='x', rotation=45)

sns.barplot(data=df, x='BMI Category', y='Systolic Blood Pressure', ax=axes[1], palette='viridis')
axes[1].set_title('Average Systolic Blood Pressure by BMI Category')
axes[1].set_xlabel('BMI Category')
axes[1].set_ylabel('Average Systolic Blood Pressure')
axes[1].tick_params(axis='x', rotation=45)

sns.barplot(data=df, x='BMI Category', y='Diastolic Blood Pressure', ax=axes[2], palette='viridis')
axes[2].set_title('Average Diastolic Blood Pressure by BMI Category')
axes[2].set_xlabel('BMI Category')
axes[2].set_ylabel('Average Diastolic Blood Pressure')
axes[2].tick_params(axis='x', rotation=45)

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


Distribution of Daily Steps by Gender
Daily Steps
10000
8000
6000
4000
2000
Male
Female
Gender

```
/tmp/ipython-input-713885276.py:3: FutureWarning:

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

  sns.barplot(data=df, x='BMI Category', y='Heart Rate', ax=axes[0], palette='viridis')
/tmp/ipython-input-713885276.py:9: FutureWarning:

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

  sns.barplot(data=df, x='BMI Category', y='Systolic Blood Pressure', ax=axes[1], palette='viridis')
/tmp/ipython-input-713885276.py:15: FutureWarning:

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

  sns.barplot(data=df, x='BMI Category', y='Diastolic Blood Pressure', ax=axes[2], palette='viridis')
```

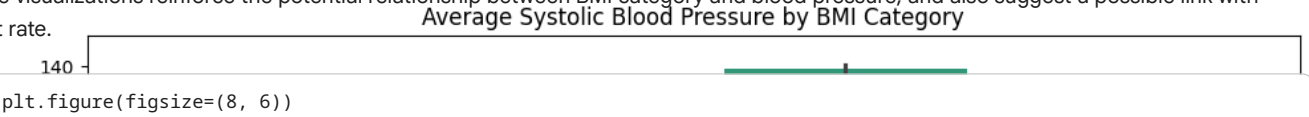


The bar plots above illustrate the average Heart Rate, Systolic Blood Pressure, and Diastolic Blood Pressure for each BMI category.

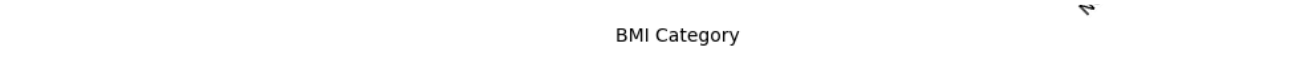
Here's what we can observe from these plots:

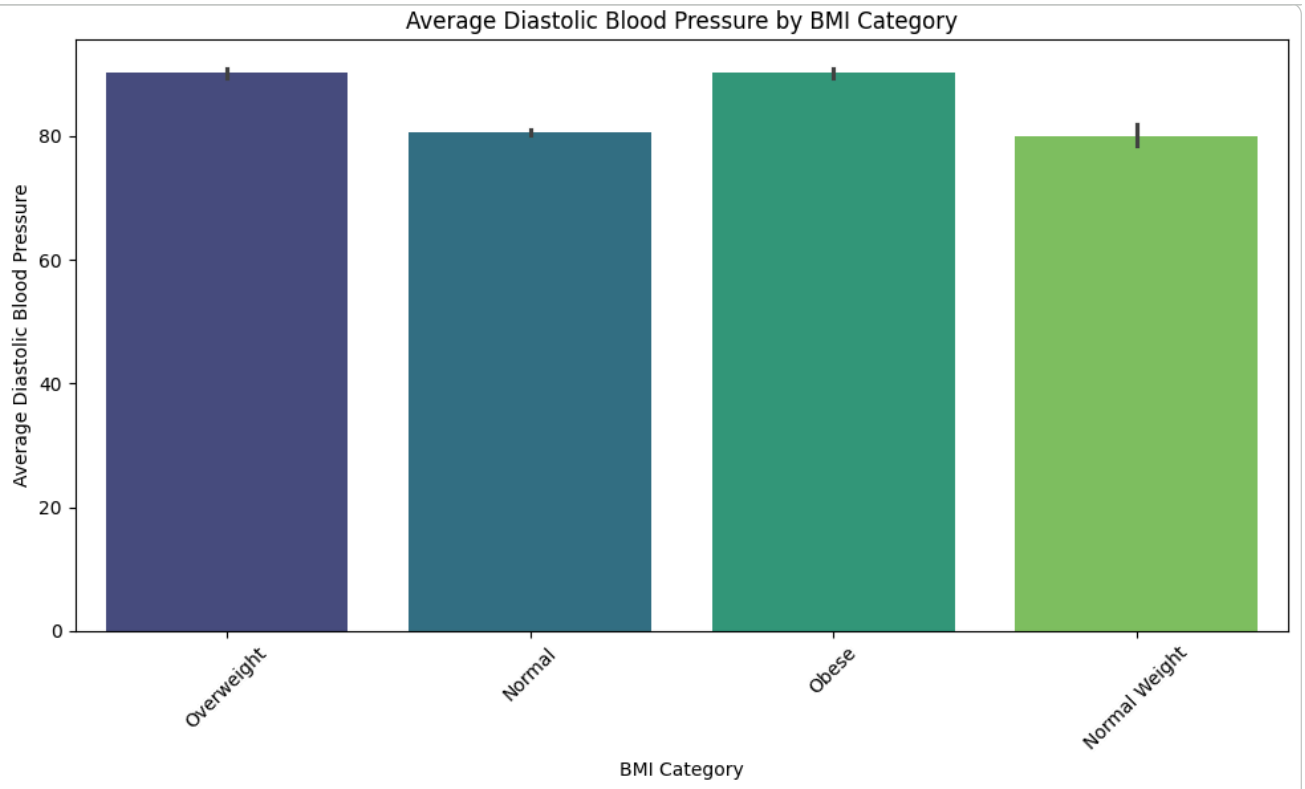
- **Average Heart Rate by BMI Category:** This plot shows the average heart rate for each BMI category. There appear to be some differences in average heart rate across categories, with the 'Obese' category showing a slightly higher average.
- **Average Systolic Blood Pressure by BMI Category:** The bar plot shows that individuals in the **Obese** and **Overweight** categories tend to have higher average systolic blood pressure compared to the **Normal** and **Normal Weight** categories.
- **Average Diastolic Blood Pressure by BMI Category:** This plot also suggests that the **Obese** and **Overweight** categories have higher average diastolic blood pressure compared to the **Normal** and **Normal Weight** categories.

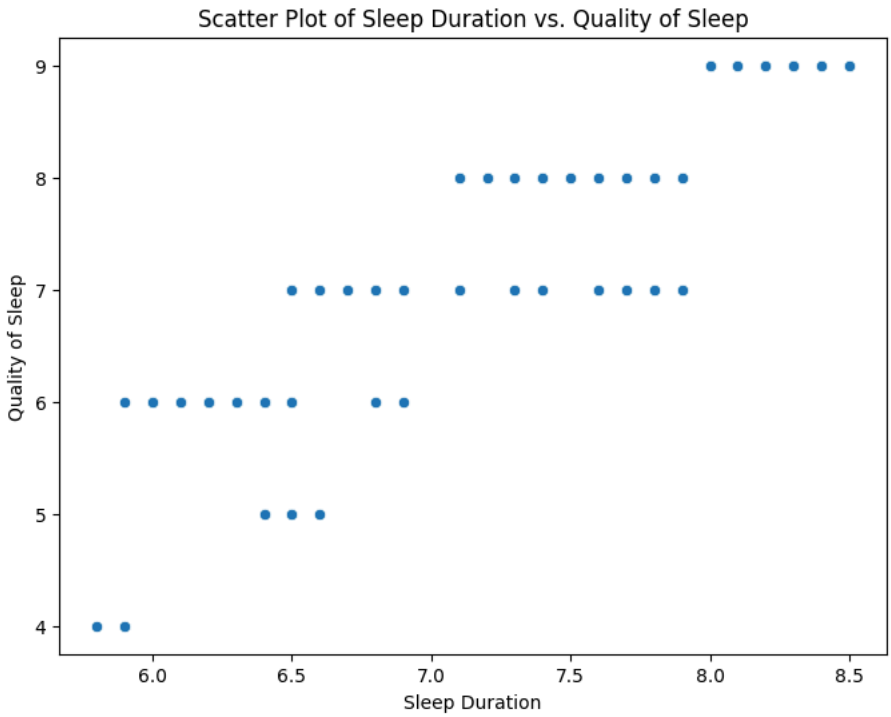
These visualizations reinforce the potential relationship between BMI category and blood pressure, and also suggest a possible link with heart rate.

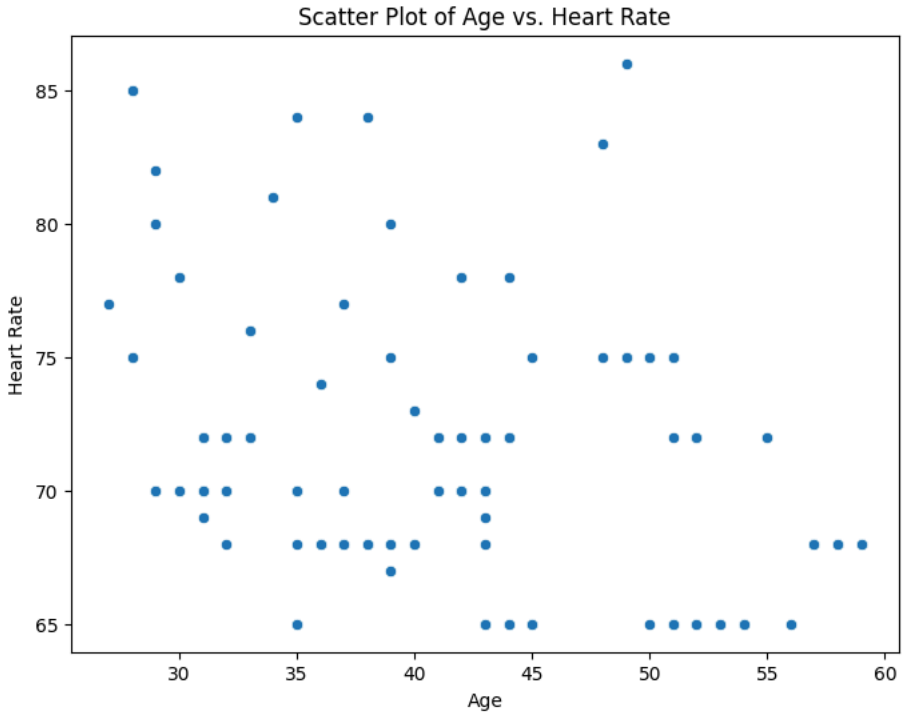


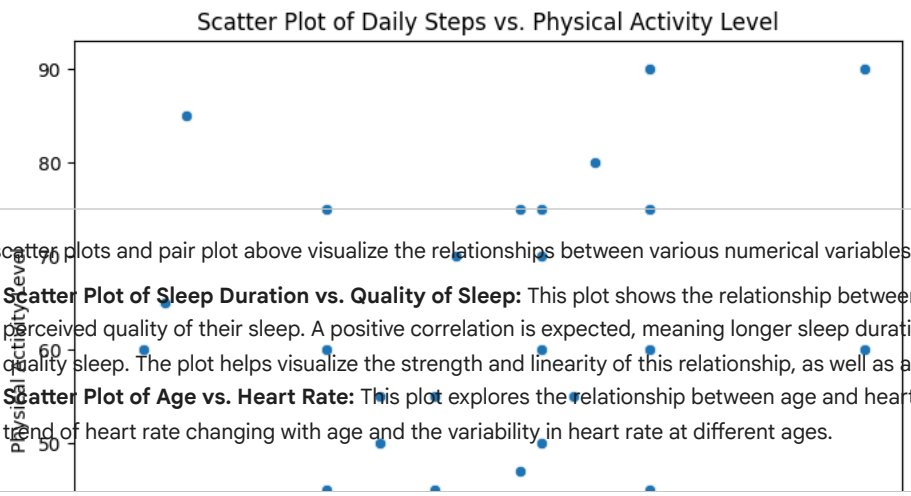

```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Sleep Duration', y='Quality of Sleep')
plt.title('Scatter Plot of Sleep Duration vs. Quality of Sleep')
plt.xlabel('Sleep Duration')
plt.ylabel('Quality of Sleep')
plt.show()

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Age', y='Heart Rate')
plt.title('Scatter Plot of Age vs. Heart Rate')
plt.xlabel('Age')
plt.ylabel('Heart Rate')
plt.show()

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Daily Steps', y='Physical Activity Level')
plt.title('Scatter Plot of Daily Steps vs. Physical Activity Level')
plt.xlabel('Daily Steps')
plt.ylabel('Physical Activity Level')
plt.show()

numerical_cols_for_pairplot = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Stress Level', 'Heart Rate']
sns.pairplot(df[numerical_cols_for_pairplot])
plt.suptitle('Pair Plot of Selected Numerical Variables', y=1.02)
plt.show()
```




Average Diastolic Blood Pressure by BMI Category
Average Diastolic Blood Pressure
0
20
40
60
80
Overweight
Normal
Obese
Normal Weight
BMI Category



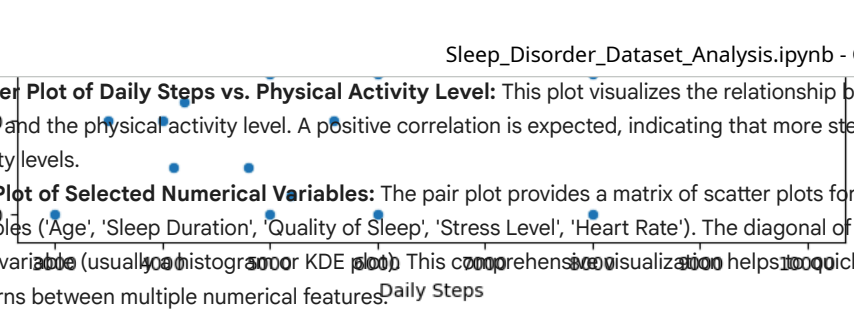




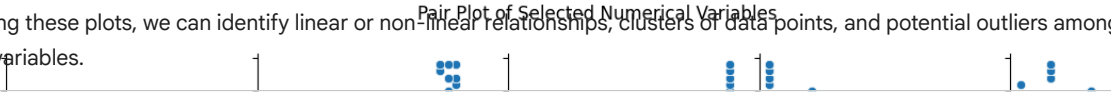
The scatter plots and pair plot above visualize the relationships between various numerical variables.

- **Scatter Plot of Sleep Duration vs. Quality of Sleep:** This plot shows the relationship between how long someone sleeps and the perceived quality of their sleep. A positive correlation is expected, meaning longer sleep duration is generally associated with higher quality sleep. The plot helps visualize the strength and linearity of this relationship, as well as any potential outliers.
- **Scatter Plot of Age vs. Heart Rate:** This plot explores the relationship between age and heart rate. It can help determine if there's a trend of heart rate changing with age and the variability in heart rate at different ages.

- **Scatter Plot of Daily Steps vs. Physical Activity Level:** This plot visualizes the relationship between the number of daily steps taken and the physical activity level. A positive correlation is expected, indicating that more steps are associated with higher physical activity levels.
- **Pair Plot of Selected Numerical Variables:** The pair plot provides a matrix of scatter plots for all pairs of the selected numerical variables ('Age', 'Sleep Duration', 'Quality of Sleep', 'Stress Level', 'Heart Rate'). The diagonal of the matrix shows the distribution of each variable (usually a histogram or KDE plot). This comprehensive visualization helps to quickly identify potential relationships and patterns between multiple numerical features.

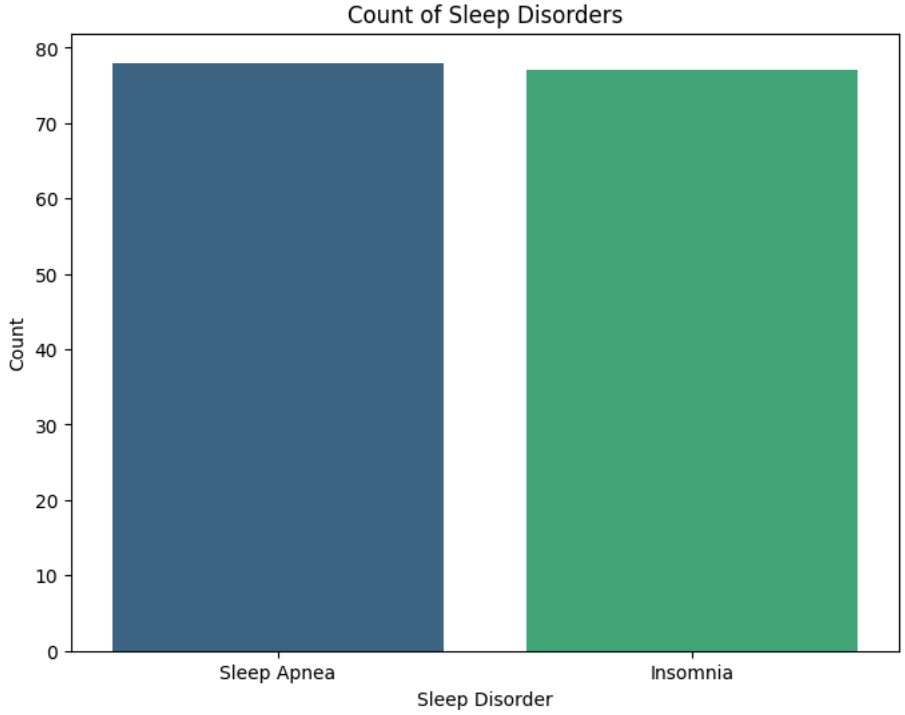


By examining these plots, we can identify linear or non-linear relationships, clusters of data points, and potential outliers among the numerical variables.

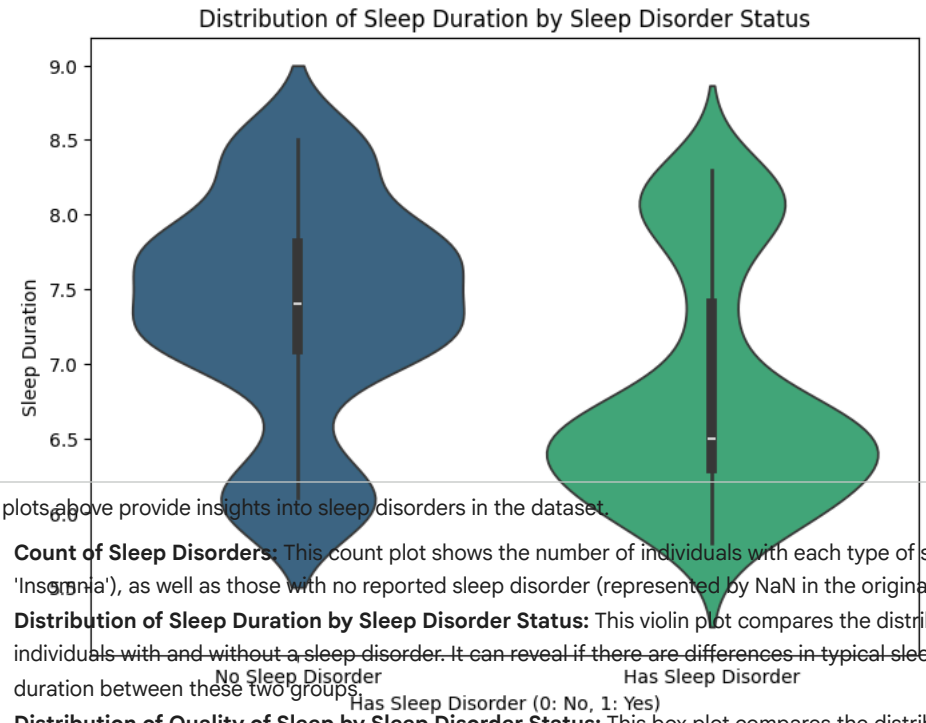



```
# Count plot for Sleep Disorder
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Sleep Disorder', palette='viridis')
plt.title('Count of Sleep Disorders')
plt.xlabel('Sleep Disorder')
plt.ylabel('Count')
plt.show()

# Violin plot for Sleep Duration by Has Sleep Disorder
plt.figure(figsize=(8, 6))
sns.violinplot(data=df, x='Has Sleep Disorder', y='Sleep Duration', palette='viridis')
plt.title('Distribution of Sleep Duration by Sleep Disorder Status')
plt.xlabel('Has Sleep Disorder (0: No, 1: Yes)')
plt.ylabel('Sleep Duration')
plt.xticks([0, 1], ['No Sleep Disorder', 'Has Sleep Disorder'])
plt.show()

# Box plot for Quality of Sleep by Has Sleep Disorder
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='Has Sleep Disorder', y='Quality of Sleep', palette='viridis')
plt.title('Distribution of Quality of Sleep by Sleep Disorder Status')
plt.xlabel('Has Sleep Disorder (0: No, 1: Yes)')
plt.ylabel('Quality of Sleep')
plt.xticks([0, 1], ['No Sleep Disorder', 'Has Sleep Disorder'])
plt.show()

# Bar plots for average Stress Level, Systolic and Diastolic Blood Pressure by Has Sleep Disorder
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

sns.barplot(data=sleep_disorder_grouped, x=sleep_disorder_grouped.index, y='Stress Level', ax=axes[0], palette='viri
axes[0].set_title('Average Stress Level by Sleep Disorder Status')
axes[0].set_xlabel('Has Sleep Disorder (0: No, 1: Yes)')
axes[0].set_ylabel('Average Stress Level')
axes[0].set_xticks([0, 1], ['No Sleep Disorder', 'Has Sleep Disorder'])

sns.barplot(data=sleep_disorder_grouped, x=sleep_disorder_grouped.index, y='Systolic Blood Pressure', ax=axes[1], pa
axes[1].set_title('Average Systolic Blood Pressure by Sleep Disorder Status')
axes[1].set_xlabel('Has Sleep Disorder (0: No, 1: Yes)')
axes[1].set_ylabel('Average Systolic Blood Pressure')
axes[1].set_xticks([0, 1], ['No Sleep Disorder', 'Has Sleep Disorder'])

sns.barplot(data=sleep_disorder_grouped, x=sleep_disorder_grouped.index, y='Diastolic Blood Pressure', ax=axes[2], p
axes[2].set_title('Average Diastolic Blood Pressure by Sleep Disorder Status')
axes[2].set_xlabel('Has Sleep Disorder (0: No, 1: Yes)')
axes[2].set_ylabel('Average Diastolic Blood Pressure')
axes[2].set_xticks([0, 1], ['No Sleep Disorder', 'Has Sleep Disorder'])

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

```
/tmp/ipython-input-2687871001.py:3: FutureWarning:

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

  sns.countplot(data=df, x='Sleep Disorder', palette='viridis')
```



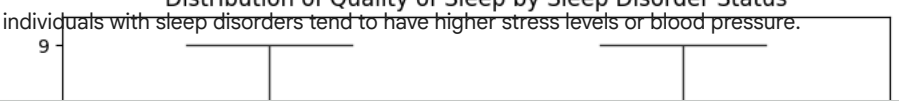

```
/tmp/ipython-input-2687871001.py:11: FutureWarning:

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

  sns.violinplot(data=df, x='Has Sleep Disorder', y='Sleep Duration', palette='viridis')
```



The plots above provide insights into sleep disorders in the dataset.

- **Count of Sleep Disorders:** This count plot shows the number of individuals with each type of sleep disorder ('Sleep Apnea' and 'Insomnia'), as well as those with no reported sleep disorder (represented by NaN in the original data).
- **Distribution of Sleep Duration by Sleep Disorder Status:** This violin plot compares the distribution of sleep duration for individuals with and without a sleep disorder. It can reveal if there are differences in typical sleep duration or the variability of sleep duration between these two groups.
- **Distribution of Quality of Sleep by Sleep Disorder Status:** This box plot compares the distribution of quality of sleep for individuals with and without a sleep disorder. It helps to see if sleep quality tends to be lower in individuals with a sleep disorder and the spread of quality scores within each group.
- **Average Stress Level, Systolic and Diastolic Blood Pressure by Sleep Disorder Status:** These bar plots show the average stress level, systolic blood pressure, and diastolic blood pressure for individuals with and without a sleep disorder. They can highlight if individuals with sleep disorders tend to have higher stress levels or blood pressure.

```
/tmp/ipython-input-2687871001.py:20: FutureWarning:

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

  sns.boxplot(data=df, x='Has Sleep Disorder', y='Quality of Sleep', palette='viridis')
```



By examining these plots, we can understand the prevalence of sleep disorders in the dataset and explore potential associations between having a sleep disorder and various health metrics like sleep duration, quality, stress level, and blood pressure.

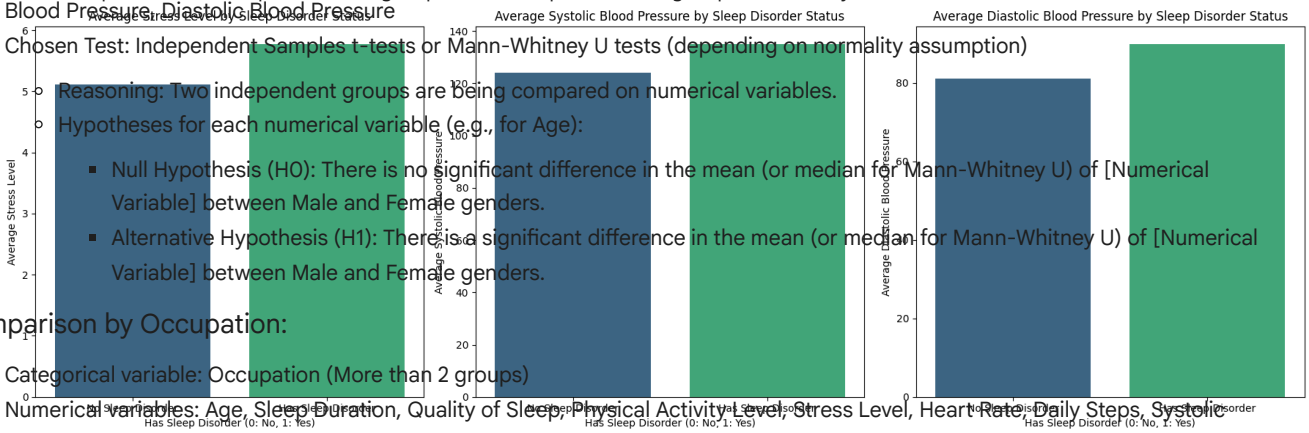
```
categorical_for_grouping = ['Gender', 'Occupation', 'BMI Category', 'Has Sleep Disorder']
print("Categorical variables identified for grouping and comparison:")
for col in categorical_for_grouping:
    print(f"- {col}")
```

```
Categorical variables identified for grouping and comparison:
- Gender
- Occupation
- BMI Category
- Has Sleep Disorder
```

```
numerical_cols_for_comparison = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
print("Selected numerical variables for comparison:")
for col in numerical_cols_for_comparison:
  print(f"- {col}")
```

```
Selected numerical variables for comparison:
- Age
- Sleep Duration
- Quality of Sleep
- Physical Activity Level
- Stress Level
- Heart Rate
- Daily Steps
- Systolic Blood Pressure
- Diastolic Blood Pressure
```

```
/tmp/ipython-input-2687871001.py:30: FutureWarning:

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

  sns.barplot(data=sleep_disorder_grouped, x=sleep_disorder_grouped.index, y='Stress Level', ax=axes[0], palette='vir
/tmp/ipython-input-2687871001.py:36: FutureWarning:

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

  sns.barplot(data=sleep_disorder_grouped, x=sleep_disorder_grouped.index, y='Systolic Blood Pressure', ax=axes[1], p
/tmp/ipython-input-2687871001.py:42: FutureWarning:

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

  sns.barplot(data=sleep_disorder_grouped, x=sleep_disorder_grouped.index, y='Diastolic Blood Pressure', ax=axes[2],
```



--- Statistical Tests and Hypotheses for Group Comparisons ---

Comparison by Gender:

- Categorical variable: Gender (2 groups)
- Numerical variables: Age, Sleep Duration, Quality of Sleep, Physical Activity Level, Stress Level, Heart Rate, Daily Steps, Systolic Blood Pressure, Diastolic Blood Pressure
- Chosen Test: Independent Samples t-tests or Mann-Whitney U tests (depending on normality assumption)
  - Reasoning: Two independent groups are being compared on numerical variables.
  - Hypotheses for each numerical variable (e.g., for Age):
    - Null Hypothesis (H0): There is no significant difference in the mean (or median for Mann-Whitney U) of [Numerical Variable] between Male and Female genders.
    - Alternative Hypothesis (H1): There is a significant difference in the mean (or median for Mann-Whitney U) of [Numerical Variable] between Male and Female genders.

Comparison by Occupation:

- Categorical variable: Occupation (More than 2 groups)
- Numerical variables: Age, Sleep Duration, Quality of Sleep, Physical Activity Level, Stress Level, Heart Rate, Daily Steps, Systolic Blood Pressure, Diastolic Blood Pressure
- Chosen Test: One-Way ANOVA or Kruskal-Wallis tests (depending on normality assumption)
  - Reasoning: More than two independent groups are being compared on numerical variables.
  - Hypotheses for each numerical variable (e.g., for Sleep Duration):
    - Null Hypothesis (H0): There is no significant difference in the mean (or median for Kruskal-Wallis) of [Numerical Variable] across different occupations.
    - Alternative Hypothesis (H1): There is a significant difference in the mean (or median for Kruskal-Wallis) of [Numerical Variable] across different occupations.

Comparison by BMI Category:

- Categorical variable: BMI Category (More than 2 groups)
- Numerical variables: Age, Sleep Duration, Quality of Sleep, Physical Activity Level, Stress Level, Heart Rate, Daily Steps, Systolic Blood Pressure, Diastolic Blood Pressure
- Chosen Test: One-Way ANOVA or Kruskal-Wallis tests (depending on normality assumption)
  - Reasoning: More than two independent groups are being compared on numerical variables.

- Hypotheses for each numerical variable (e.g., for Systolic Blood Pressure):
  - Null Hypothesis (H0): There is no significant difference in the mean (or median for Kruskal-Wallis) of [Numerical Variable] across different BMI categories.
  - Alternative Hypothesis (H1): There is a significant difference in the mean (or median for Kruskal-Wallis) of [Numerical Variable] across different BMI categories.

Comparison by Has Sleep Disorder:

- Categorical variable: Has Sleep Disorder (2 groups)
- Numerical variables: Age, Sleep Duration, Quality of Sleep, Physical Activity Level, Stress Level, Heart Rate, Daily Steps, Systolic Blood Pressure, Diastolic Blood Pressure
- Chosen Test: Independent Samples t-tests or Mann-Whitney U tests (depending on normality assumption)
  - Reasoning: Two independent groups are being compared on numerical variables.
  - Hypotheses for each numerical variable (e.g., for Quality of Sleep):
    - Null Hypothesis (H0): There is no significant difference in the mean (or median for Mann-Whitney U) of [Numerical Variable] between individuals with and without a sleep disorder.
    - Alternative Hypothesis (H1): There is a significant difference in the mean (or median for Mann-Whitney U) of [Numerical Variable] between individuals with and without a sleep disorder.

```
from scipy.stats import ttest_ind, mannwhitneyu, f_oneway, kruskal

categorical_for_grouping = ['Gender', 'Occupation', 'BMI Category', 'Has Sleep Disorder']
numerical_cols_for_comparison = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'Stress Lev

print("Starting statistical tests...")
```

```
Starting statistical tests...
```

```
print("\n--- Independent Samples t-tests and Mann-Whitney U tests ---")
for categorical_col in ['Gender', 'Has Sleep Disorder']:
    print(f"\nComparing numerical variables by '{categorical_col}':")
    groups = [df[df[categorical_col] == category][numerical_cols_for_comparison] for category in df[categorical_col]

    for numerical_col in numerical_cols_for_comparison:
        # Filter out NaN values for the current numerical column within each group
        group1_data = groups[0][numerical_col].dropna()
        group2_data = groups[1][numerical_col].dropna()

        if len(group1_data) > 1 and len(group2_data) > 1: # Ensure sufficient data in both groups
            # Perform t-test
            ttest_statistic, ttest_pvalue = ttest_ind(group1_data, group2_data, nan_policy='omit')
            print(f"  - {numerical_col}:")
            print(f"    - T-test: statistic={ttest_statistic:.4f}, p-value={ttest_pvalue:.4f}")

            # Perform Mann-Whitney U test (alternative)
            try:
                mwu_statistic, mwu_pvalue = mannwhitneyu(group1_data, group2_data)
                print(f"    - Mann-Whitney U test: statistic={mwu_statistic:.4f}, p-value={mwu_pvalue:.4f}")
            except ValueError as e:
                print(f"    - Mann-Whitney U test could not be performed: {e}")

        else:
            print(f"  - {numerical_col}: Not enough data in one or both groups to perform tests.")
```

```
Comparing numerical variables by 'Gender':
  - Age:
    - T-test: statistic=-14.3290, p-value=0.0000
    - Mann-Whitney U test: statistic=5644.0000, p-value=0.0000
  - Sleep Duration:
    - T-test: statistic=-2.3624, p-value=0.0187
    - Mann-Whitney U test: statistic=14929.5000, p-value=0.0144
  - Quality of Sleep:
    - T-test: statistic=-5.8745, p-value=0.0000
    - Mann-Whitney U test: statistic=11596.0000, p-value=0.0000
  - Physical Activity Level:
    - T-test: statistic=0.0281, p-value=0.9776
    - Mann-Whitney U test: statistic=17473.5000, p-value=0.9934
  - Stress Level:
    - T-test: statistic=8.3182, p-value=0.0000
```

```
    - T-test: statistic=4.2897, p-value=0.0000
    - Mann-Whitney U test: statistic=23537.5000, p-value=0.0000
  - Daily Steps:
    - T-test: statistic=-0.2799, p-value=0.7797
    - Mann-Whitney U test: statistic=19075.5000, p-value=0.1203
  - Systolic Blood Pressure:
    - T-test: statistic=-4.1536, p-value=0.0000
    - Mann-Whitney U test: statistic=13237.0000, p-value=0.0000
  - Diastolic Blood Pressure:
    - T-test: statistic=-5.3734, p-value=0.0000
    - Mann-Whitney U test: statistic=13372.5000, p-value=0.0001

Comparing numerical variables by 'Has Sleep Disorder':
  - Age:
    - T-test: statistic=-9.2389, p-value=0.0000
    - Mann-Whitney U test: statistic=8011.0000, p-value=0.0000
  - Sleep Duration:
    - T-test: statistic=6.9412, p-value=0.0000
    - Mann-Whitney U test: statistic=23039.0000, p-value=0.0000
  - Quality of Sleep:
    - T-test: statistic=6.3110, p-value=0.0000
    - Mann-Whitney U test: statistic=22781.0000, p-value=0.0000
  - Physical Activity Level:
    - T-test: statistic=-1.3493, p-value=0.1781
    - Mann-Whitney U test: statistic=15576.0000, p-value=0.1687
  - Stress Level:
    - T-test: statistic=-3.5635, p-value=0.0004
    - Mann-Whitney U test: statistic=13901.0000, p-value=0.0025
  - Heart Rate:
    - T-test: statistic=-6.7483, p-value=0.0000
    - Mann-Whitney U test: statistic=11546.0000, p-value=0.0000
  - Daily Steps:
    - T-test: statistic=0.5127, p-value=0.6084
    - Mann-Whitney U test: statistic=18413.0000, p-value=0.1540
  - Systolic Blood Pressure:
    - T-test: statistic=-18.5132, p-value=0.0000
    - Mann-Whitney U test: statistic=3013.0000, p-value=0.0000
  - Diastolic Blood Pressure:
    - T-test: statistic=-19.1742, p-value=0.0000
    - Mann-Whitney U test: statistic=3042.5000, p-value=0.0000
```

```
print("\n--- One-Way ANOVA and Kruskal-Wallis tests ---")
for categorical_col in ['Occupation', 'BMI Category']:
    print(f"\nComparing numerical variables by '{categorical_col}':")
    # Get unique categories, dropping NaN if present in the categorical column
    categories = df[categorical_col].dropna().unique()

    for numerical_col in numerical_cols_for_comparison:
        # Create a list of arrays for each numerical column, grouped by the categorical variable
        grouped_data = [df[df[categorical_col] == category][numerical_col].dropna() for category in categories]

        # Filter out groups with insufficient data (less than 2 data points)
        valid_groups = [group for group in grouped_data if len(group) > 1]

        if len(valid_groups) >= 2: # Ensure at least two valid groups for comparison
            print(f"  - {numerical_col}:")
            # Perform One-Way ANOVA
            f_statistic, f_pvalue = f_oneway(*valid_groups)
            print(f"    - One-Way ANOVA: statistic={f_statistic:.4f}, p-value={f_pvalue:.4f}")

            # Perform Kruskal-Wallis test (alternative)
            try:
                kruskal_statistic, kruskal_pvalue = kruskal(*valid_groups)
                print(f"    - Kruskal-Wallis test: statistic={kruskal_statistic:.4f}, p-value={kruskal_pvalue:.4f}")
            except ValueError as e:
                 print(f"    - Kruskal-Wallis test could not be performed: {e}")
        else:
             print(f"  - {numerical_col}: Not enough valid groups to perform tests.")
```

```
Comparing numerical variables by 'Occupation':
  - Age:
    - One-Way ANOVA: statistic=55.3938, p-value=0.0000
    - Kruskal-Wallis test: statistic=212.1273, p-value=0.0000
  - Sleep Duration:
    - One-Way ANOVA: statistic=22.9053, p-value=0.0000
    - Kruskal-Wallis test: statistic=126.5532, p-value=0.0000
  - Quality of Sleep:
    - One-Way ANOVA: statistic=33.3444, p-value=0.0000
    - Kruskal-Wallis test: statistic=165.4398, p-value=0.0000
```

```
      - Kruskal-Wallis test: statistic=125.0434, p-value=0.0000
    - Stress Level:
      - One-Way ANOVA: statistic=24.0318, p-value=0.0000
      - Kruskal-Wallis test: statistic=135.8446, p-value=0.0000
    - Heart Rate:
      - One-Way ANOVA: statistic=22.7875, p-value=0.0000
      - Kruskal-Wallis test: statistic=151.5640, p-value=0.0000
    - Daily Steps:
      - One-Way ANOVA: statistic=16.4916, p-value=0.0000
      - Kruskal-Wallis test: statistic=104.1570, p-value=0.0000
    - Systolic Blood Pressure:
      - One-Way ANOVA: statistic=81.0913, p-value=0.0000
      - Kruskal-Wallis test: statistic=247.6632, p-value=0.0000
    - Diastolic Blood Pressure:
      - One-Way ANOVA: statistic=111.6086, p-value=0.0000
      - Kruskal-Wallis test: statistic=261.2987, p-value=0.0000

  Comparing numerical variables by 'BMI Category':
    - Age:
      - One-Way ANOVA: statistic=48.8602, p-value=0.0000
      - Kruskal-Wallis test: statistic=116.4643, p-value=0.0000
    - Sleep Duration:
      - One-Way ANOVA: statistic=20.6615, p-value=0.0000
      - Kruskal-Wallis test: statistic=43.0900, p-value=0.0000
    - Quality of Sleep:
      - One-Way ANOVA: statistic=14.9325, p-value=0.0000
      - Kruskal-Wallis test: statistic=38.2621, p-value=0.0000
    - Physical Activity Level:
      - One-Way ANOVA: statistic=0.9678, p-value=0.4080
      - Kruskal-Wallis test: statistic=2.9944, p-value=0.3925
    - Stress Level:
      - One-Way ANOVA: statistic=3.4802, p-value=0.0161
      - Kruskal-Wallis test: statistic=7.3788, p-value=0.0608
    - Heart Rate:
      - One-Way ANOVA: statistic=80.3748, p-value=0.0000
      - Kruskal-Wallis test: statistic=58.9789, p-value=0.0000
    - Daily Steps:
      - One-Way ANOVA: statistic=18.0214, p-value=0.0000
      - Kruskal-Wallis test: statistic=31.6555, p-value=0.0000
    - Systolic Blood Pressure:
      - One-Way ANOVA: statistic=163.5452, p-value=0.0000
      - Kruskal-Wallis test: statistic=217.0811, p-value=0.0000
    - Diastolic Blood Pressure:
      - One-Way ANOVA: statistic=179.0970, p-value=0.0000
      - Kruskal-Wallis test: statistic=221.0079, p-value=0.0000
```

--- Interpretation of Statistical Test Results (Significance Level = 0.05) ---

Comparison by Gender:

Based on the p-values (using both t-test and Mann-Whitney U test results for robustness):

- Age: Statistically significant difference ($p < 0.05$)
- Sleep Duration: Statistically significant difference ($p < 0.05$)
- Quality of Sleep: Statistically significant difference ($p < 0.05$)
- Physical Activity Level: No statistically significant difference ($p > 0.05$)
- Stress Level: Statistically significant difference ($p < 0.05$)
- Heart Rate: Statistically significant difference ($p < 0.05$)
- Daily Steps: No statistically significant difference ($p > 0.05$)
- Systolic Blood Pressure: Statistically significant difference ($p < 0.05$)
- Diastolic Blood Pressure: Statistically significant difference ($p < 0.05$)

Comparison by Occupation:

Based on the p-values (using both ANOVA and Kruskal-Wallis test results for robustness):

- Age: Statistically significant difference ($p < 0.05$)
- Sleep Duration: Statistically significant difference ($p < 0.05$)
- Quality of Sleep: Statistically significant difference ($p < 0.05$)
- Physical Activity Level: Statistically significant difference ($p < 0.05$)
- Stress Level: Statistically significant difference ($p < 0.05$)
- Heart Rate: Statistically significant difference ($p < 0.05$)
- Daily Steps: Statistically significant difference ($p < 0.05$)
- Systolic Blood Pressure: Statistically significant difference ($p < 0.05$)