In [ ]:
```
# DSC 530
# Gracie Inman
# Final Project
# 08/12/23
```

In [57]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import statsmodels.api as sm

NUM_BINS = 20
# Step 1: Load the dataset
def load_dataset(filename):
    return pd.read_csv(filename)

# Step 2: Describe the 5 variables
# Heart rate is the patient's heart rate.
## Sleep duration is how long the patient slept
## Sleep quality quantifies the quality of the sleep
## Physical activity level quantifies how active the patient is.
## Age is how old the subject is.
## Stress Level is how stressed the subject feels quantified.

# Step 3: Create histograms and identify outliers (Chapter 2)
def plot_histogram_with_outliers(data, column):
    plt.figure(figsize=(8, 6))
    plt.hist(data[column], bins=NUM_BINS, edgecolor='k', alpha=0.7)
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.title(f'Histogram of {column}')
    plt.grid(True)
    plt.show()

# Plot histograms for each variable and identify outliers
variables = ['Gender', 'Age', 'Sleep Duration', 'Quality of Sleep', 'Physical A
for var in variables:
    plot_histogram_with_outliers(data, var)

# Step 4: Calculate descriptive characteristics (Chapter 2)
def calculate_descriptive_stats(data, column):
    mean = data[column].mean()
    mode = data[column].mode()[0]
    spread = data[column].std()
    tails = data[column].skew()
    return mean, mode, spread, tails

# Step 5: Compare scenarios using PMF (Chapter 3)
def plot_pmf(data, column, scenario_column, scenario_value):
    filtered_data = data[data[scenario_column] == scenario_value]
    pmf = filtered_data[column].value_counts(normalize=True).sort_index()

    plt.bar(pmf.index, pmf.values, label='Data')
    plt.xlabel(column)
    plt.ylabel('PMF')
    plt.title(f'PMF of {column} for {scenario_column} = {scenario_value}')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

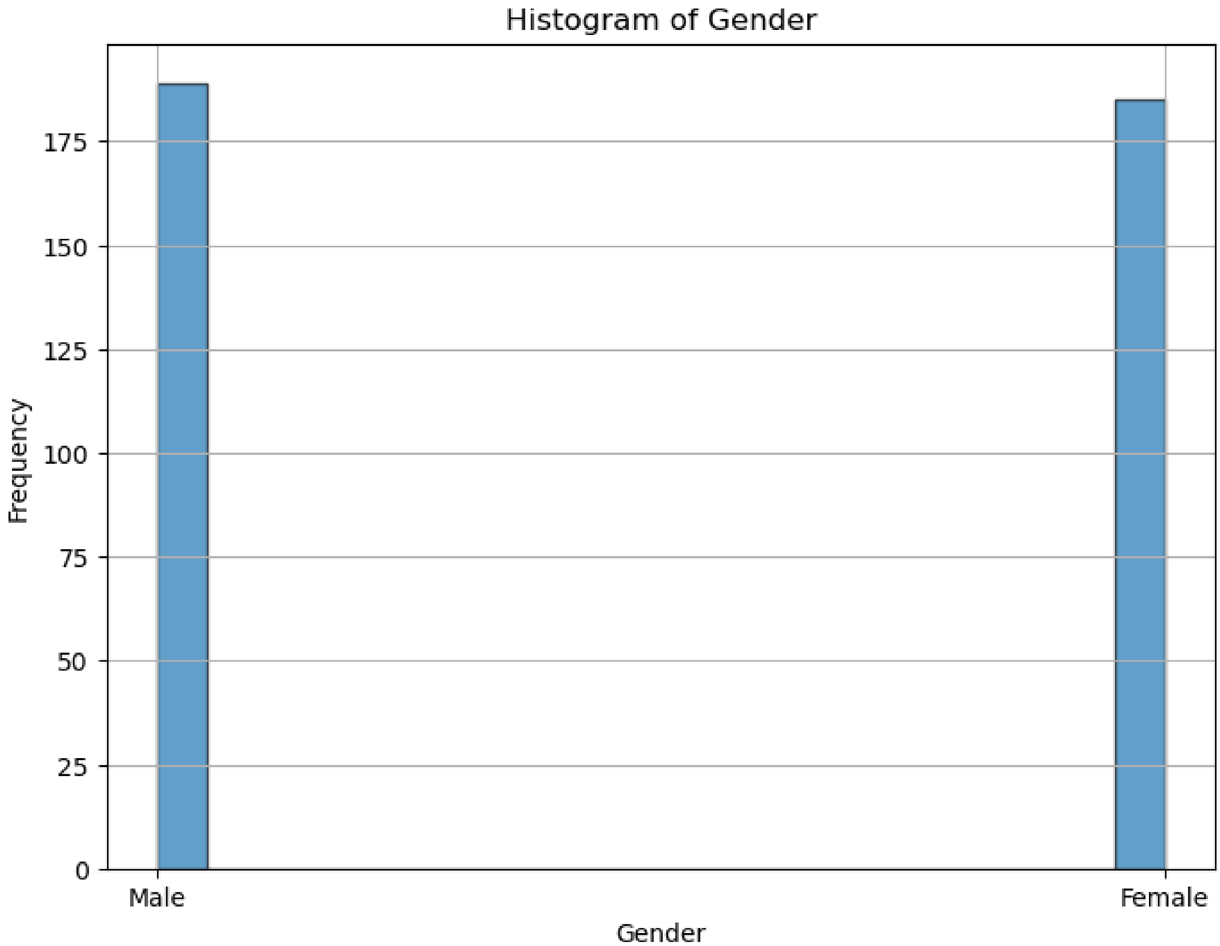
```
    # Add a normal distribution line for comparison
    mean = filtered_data[column].mean()
    std = filtered_data[column].std()
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)

    # Plot a normal distribution line with a custom scaling factor
    norm_pdf = stats.norm.pdf(x, mean, std)
    scaling_factor = np.max(pmf.values) / np.max(norm_pdf) * 0.8
    plt.plot(x, norm_pdf * scaling_factor, 'r', label='Normal Distribution')
    plt.legend()

    plt.show()


# Step 6: Create a CDF (Chapter 4)
def plot_cdf(data, column):
    plt.figure(figsize=(8, 6))
    sorted_data = np.sort(data[column])
    cdf = np.arange(1, len(sorted_data) + 1) / len(sorted_data)
    plt.plot(sorted_data, cdf)
    plt.xlabel(column)
    plt.ylabel('CDF')
    plt.title(f'CDF of {column}')
    plt.grid(True)

    # Add a normal distribution line for comparison
    mean = data[column].mean()
    std = data[column].std()
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 20)
    cdf_norm = stats.norm.cdf(x, mean, std)
    plt.plot(x, cdf_norm, 'r', label='Normal Distribution')
    plt.legend()

    plt.show()

# Step 7: Plot an analytical distribution and provide analysis (Chapter 5)
def plot_analytical_distribution(data, column, distribution=stats.norm):
    plt.figure(figsize=(8, 6))

    # Plot the histogram or KDE of the data
    plt.hist(data[column], bins=20, density=True, alpha=0.7, label='Data')

    # Plot the analytical distribution
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    params = distribution.fit(data[column])
    pdf = distribution.pdf(x, *params)
    plt.plot(x, pdf, 'r', label='Analytical Distribution')

    plt.xlabel(column)
    plt.ylabel('Density')
    plt.title(f'Analytical Distribution vs. Data Distribution of {column}')
    plt.legend()
    plt.grid(True)

    # Adjust y-axis limits for better visibility
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

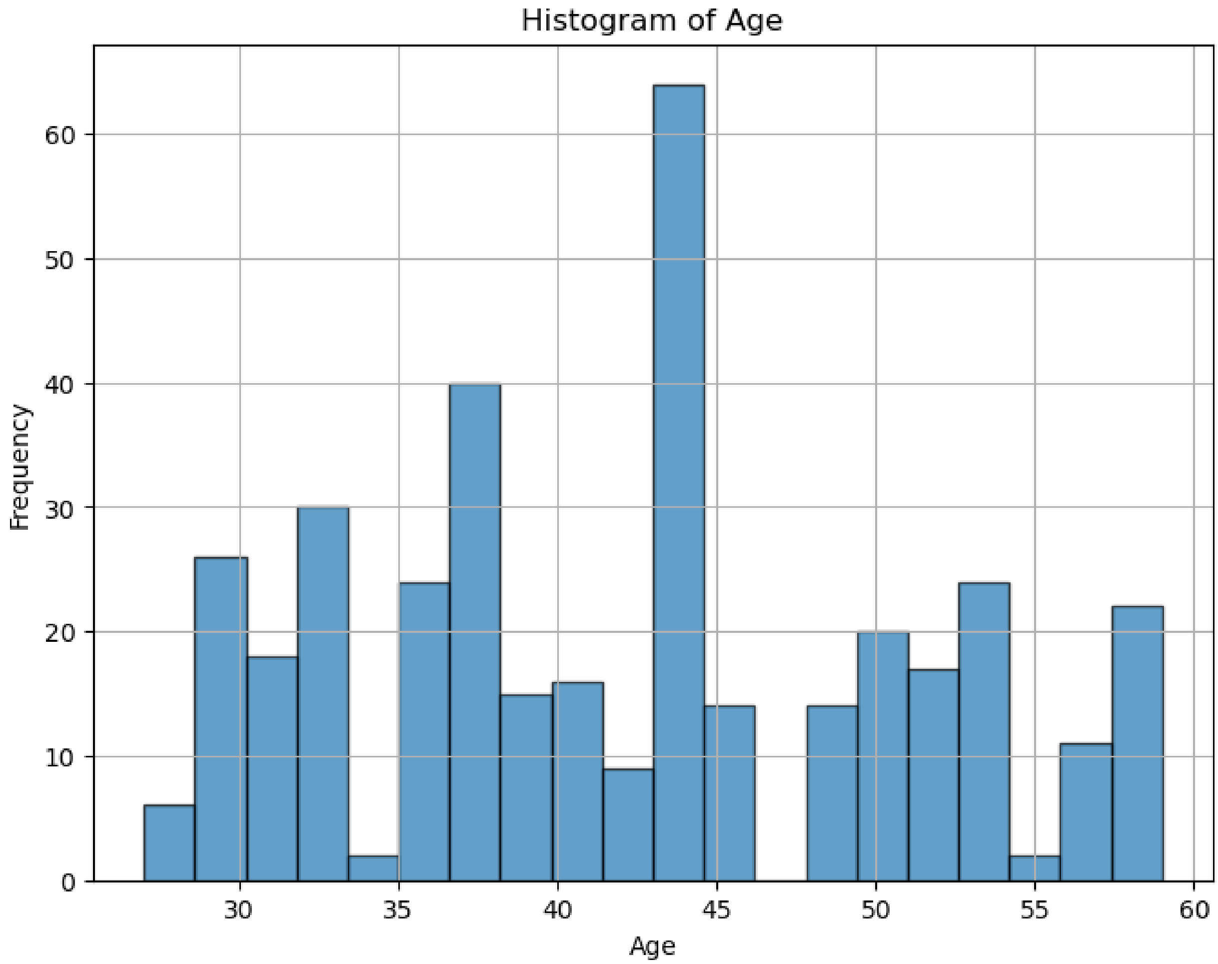
```
    plt.show()

# Step 8: Create scatter plots and analyze correlation and causation (Chapter 7
def plot_scatter(data, x_column, y_column):
    plt.figure(figsize=(8, 6))
    plt.scatter(data[x_column], data[y_column], alpha=0.5)

    # Calculate and plot the trend line (linear regression line)
    slope, intercept, r_value, p_value, std_err = stats.linregress(data[x_colum
    trend_line_x = np.array([min(data[x_column]), max(data[x_column])])
    trend_line_y = slope * trend_line_x + intercept
    plt.plot(trend_line_x, trend_line_y, color='r', label='Trend Line')

    plt.xlabel(x_column)
    plt.ylabel(y_column)
    plt.title(f'Scatter Plot of {x_column} vs {y_column}')
    plt.legend()
    plt.grid(True)
    plt.show()



# Step 9: Conduct a hypothesis test (Chapter 9)
def conduct_hypothesis_test(data, variable1, variable2):
    group1 = data[data[variable2] == 'Male'][variable1]
    group2 = data[data[variable2] == 'Female'][variable1]

    # Perform a hypothesis test (t-test)
    t_stat, p_value = stats.ttest_ind(group1, group2)

    # Provide interpretation of results
    if p_value < 0.05:
        result = "Reject null hypothesis: There is a significant difference bet
    else:
        result = "Fail to reject null hypothesis: There is no significant diffe

    print(f"T-Stat: {t_stat}")
    print(f"P-Value: {p_value}")
    print(result)
# ...

# Step 10: Conduct a regression analysis (Chapter 10 & 11)
def conduct_regression_analysis(data, dependent_variable, explanatory_variables
    X = sm.add_constant(data[explanatory_variables])
    y = data[dependent_variable]
    model = sm.OLS(y, X).fit()

    # Print regression summary and interpret results
    print(model.summary())

def analyze_variable(column_data):
    if column_data.dtype.kind in "biufc":
        # If the data is numerical, perform statistical analysis
        stats = {
            "Mean": column_data.mean(),
            "Median": column_data.median(),
            "Standard Deviation": column_data.std(),
            "Minimum": column_data.min(),
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js max(),
        }
```

```
    else:
        # If the data is non-numerical, count occurrences of each value
        counts = column_data.value_counts()
        stats = {value: count for value, count in counts.items()}
    return stats

def correlation_matrix(data_frame):
    correlation_matrix = data_frame.corr()
    return correlation_matrix

def main():
    # Call the functions to perform the analyses
    data = load_dataset('Sleep_health_and_lifestyle_dataset.csv')

    # Calculate and display descriptive statistics
    variables_to_analyze = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Heart
    for variable in variables_to_analyze:
        stats = calculate_descriptive_stats(data, variable)
        print(f"Descriptive Statistics for {variable}:")
        print(f"  Mean: {stats[0]:.2f}")
        print(f"  Mode: {stats[1]:.2f}")
        print(f"  Standard Deviation: {stats[2]:.2f}")
        print(f"  Skewness: {stats[3]:.2f}")
        print()
    plot_analytical_distribution(data, 'Quality of Sleep')
    plot_histogram_with_outliers(data, 'Physical Activity Level')
    plot_pmf(data, 'Sleep Duration', 'Gender', 'Male')
    plot_cdf(data, 'Quality of Sleep')
    plot_analytical_distribution(data, 'Physical Activity Level')
    plot_scatter(data, 'Heart Rate', 'Age')
    plot_scatter(data, 'Quality of Sleep', 'Heart Rate')
    conduct_hypothesis_test(data, 'Quality of Sleep', 'Gender')
    conduct_regression_analysis(data, 'Heart Rate', ['Sleep Duration', 'Age'])

if __name__ == "__main__":
    main()
```

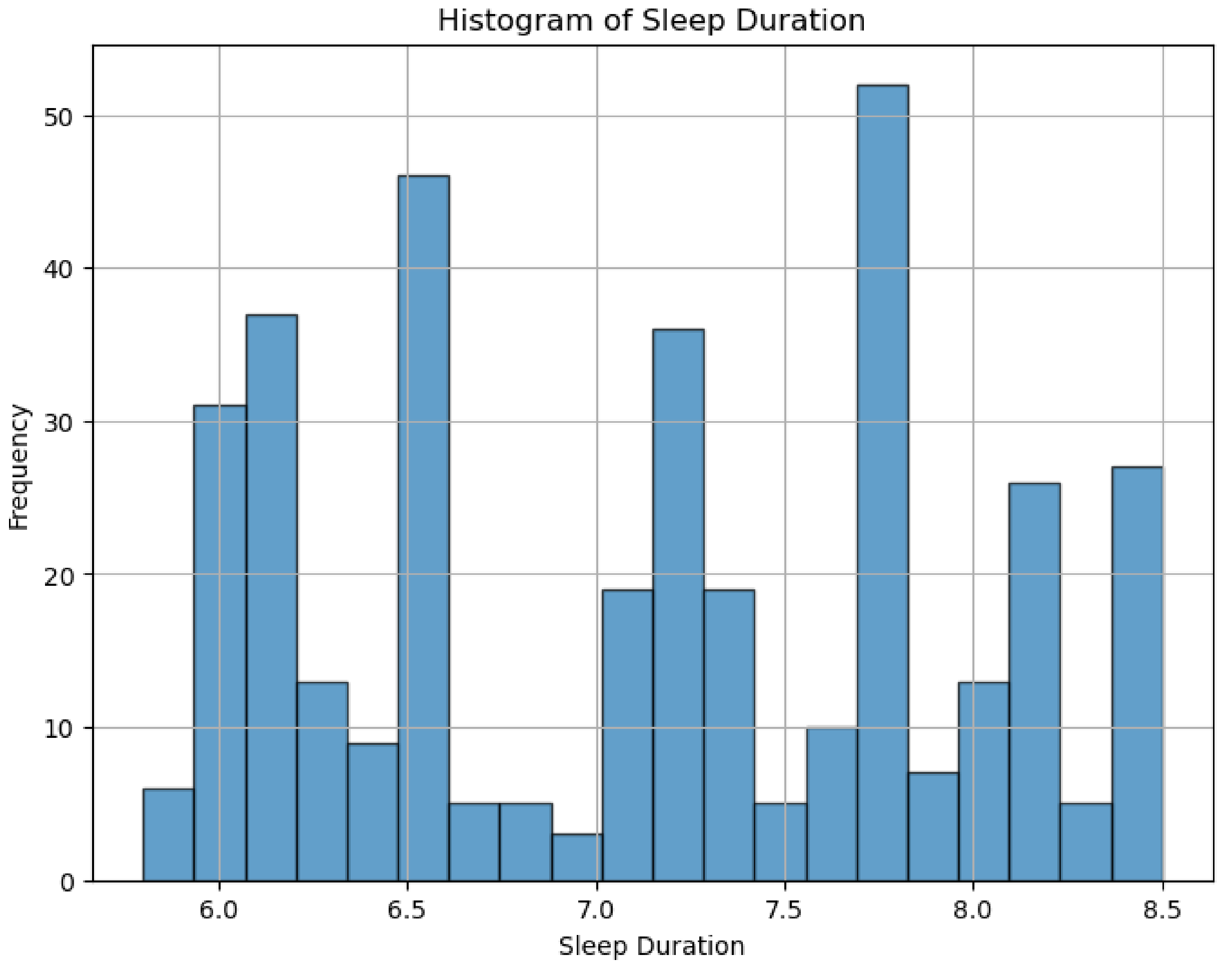
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



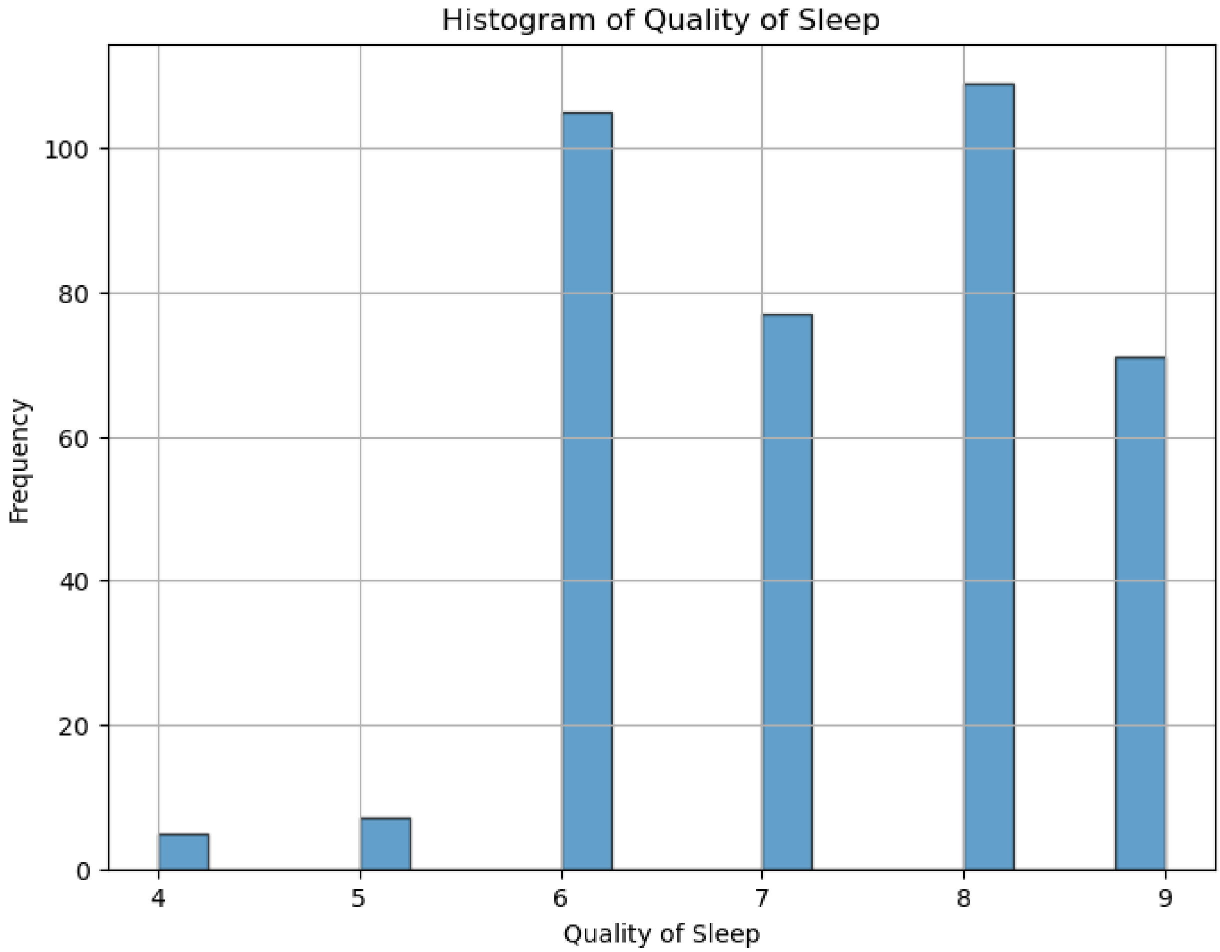
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



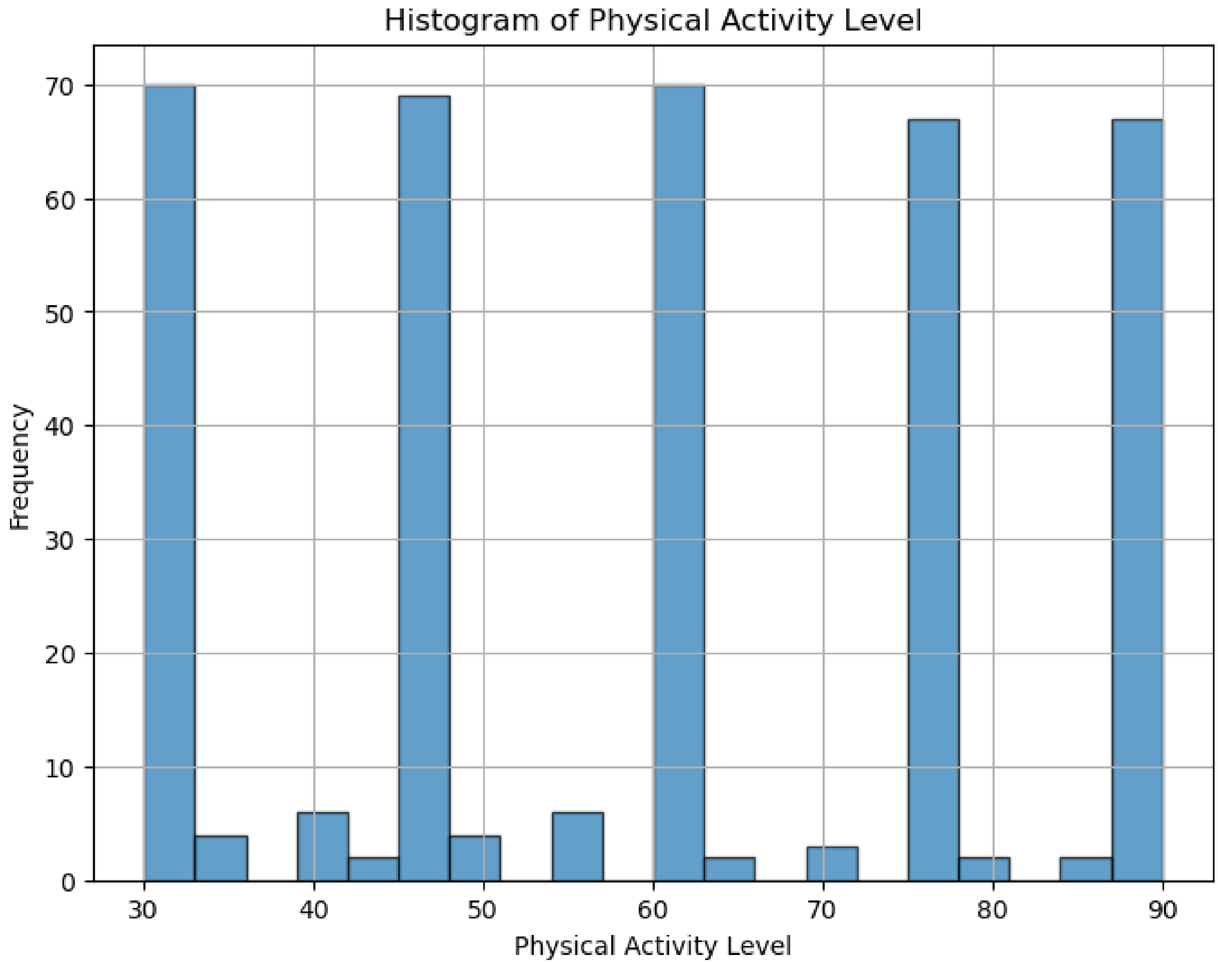
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



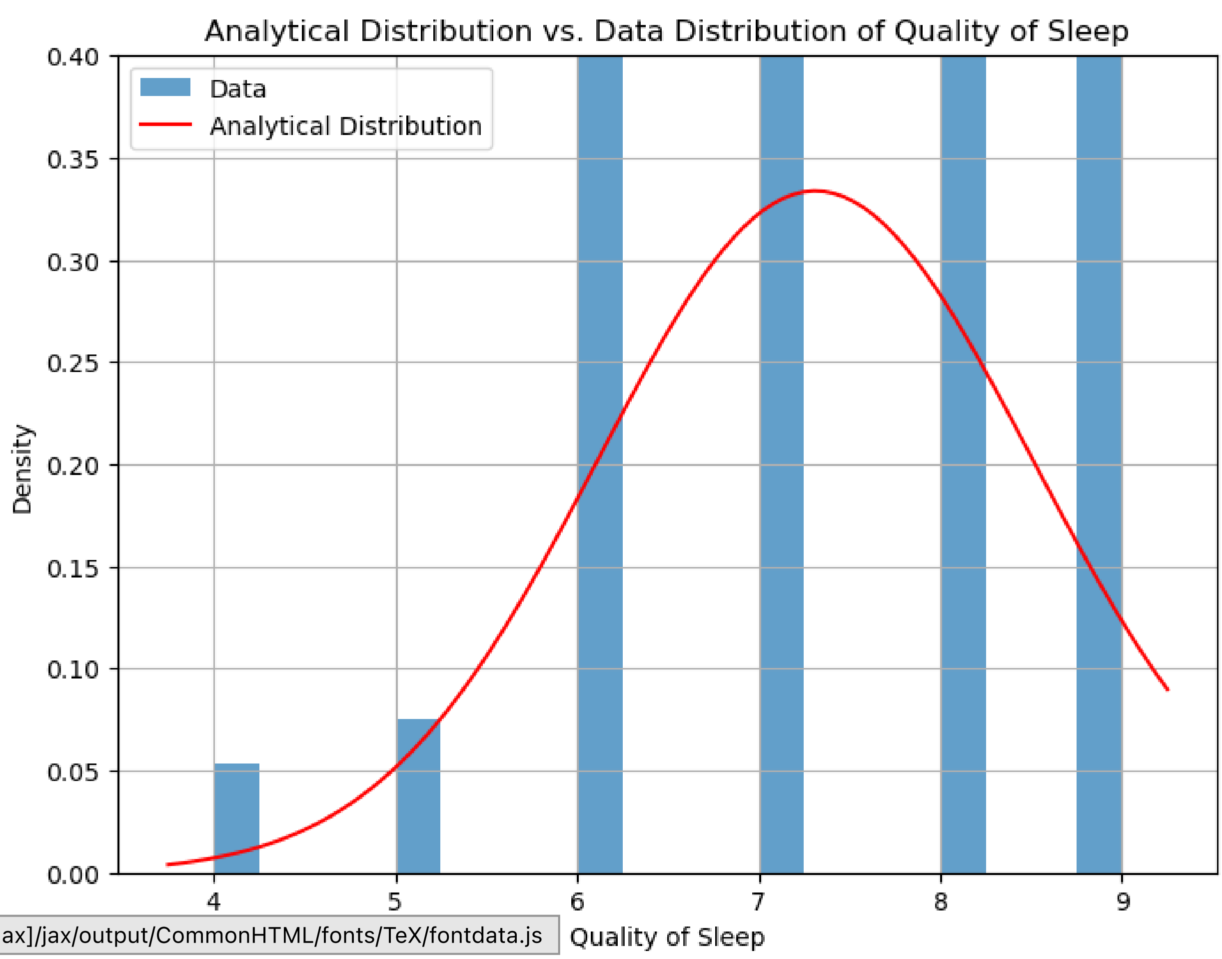
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



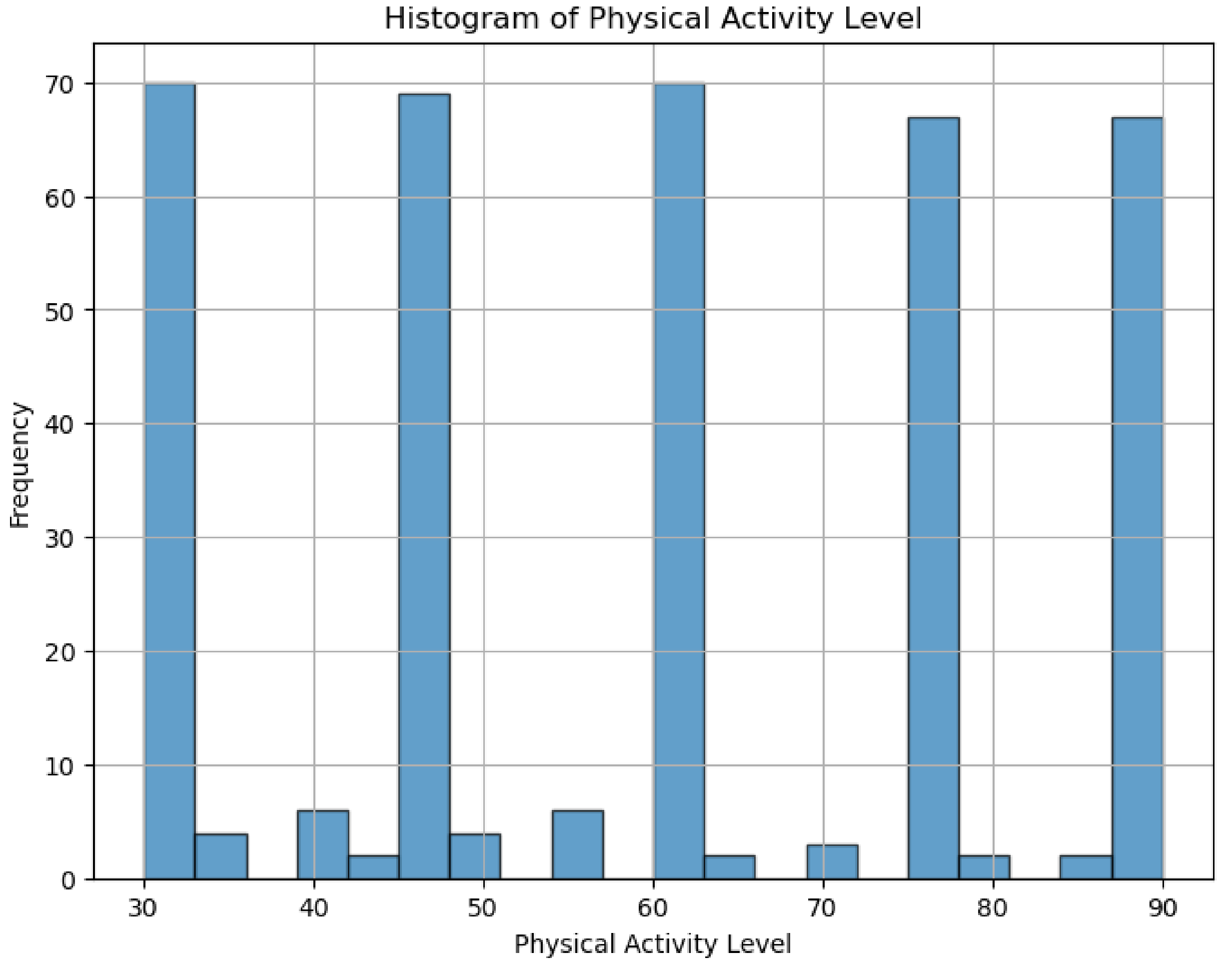
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

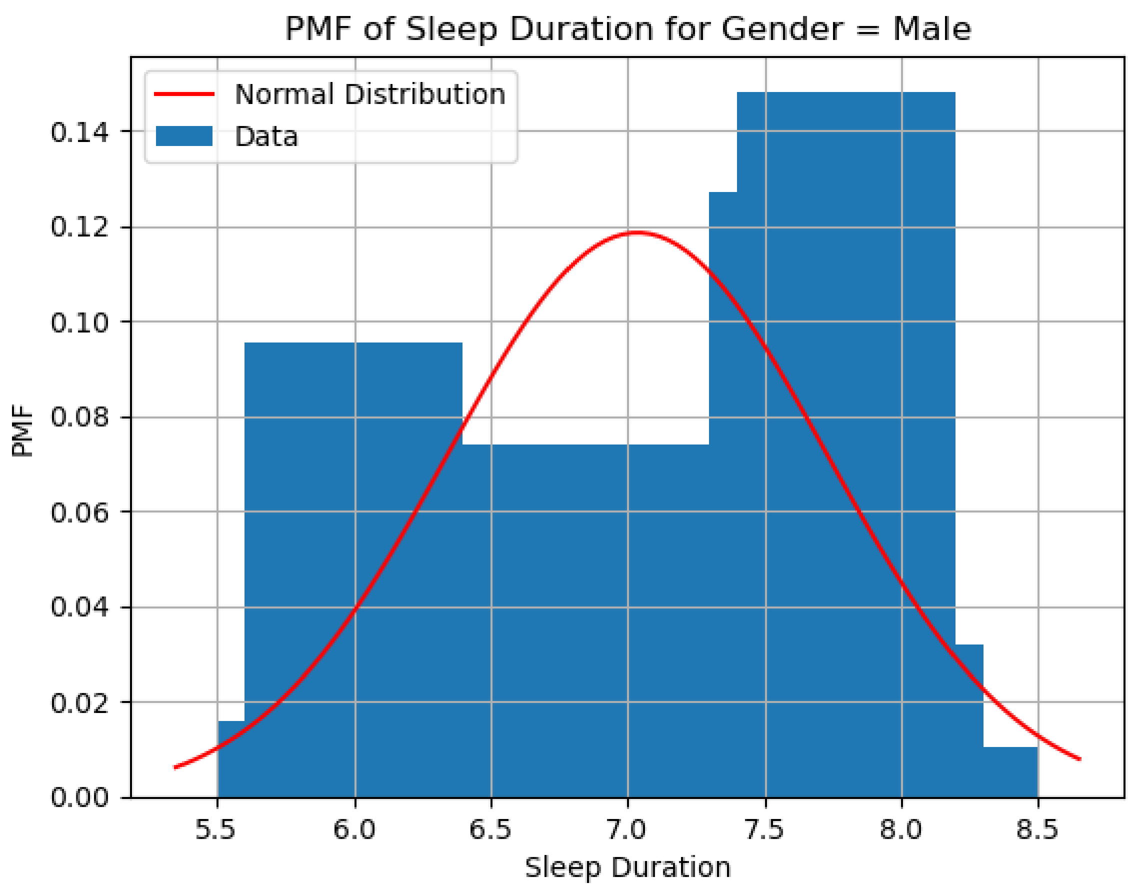

Histogram of Heart Rate
Frequency
Heart Rate
0
20
40
60
80
65
70
75
80
85

```
Descriptive Statistics for Age:
  Mean: 42.18
  Mode: 43.00
  Standard Deviation: 8.67
  Skewness: 0.26

Descriptive Statistics for Sleep Duration:
  Mean: 7.13
  Mode: 7.20
  Standard Deviation: 0.80
  Skewness: 0.04

Descriptive Statistics for Quality of Sleep:
  Mean: 7.31
  Mode: 8.00
  Standard Deviation: 1.20
  Skewness: -0.21

Descriptive Statistics for Heart Rate:
  Mean: 70.17
  Mode: 68.00
  Standard Deviation: 4.14
  Skewness: 1.22

Descriptive Statistics for Physical Activity Level:
  Mean: 59.17
  Mode: 60.00
  Standard Deviation: 20.83
  Skewness: 0.07
```

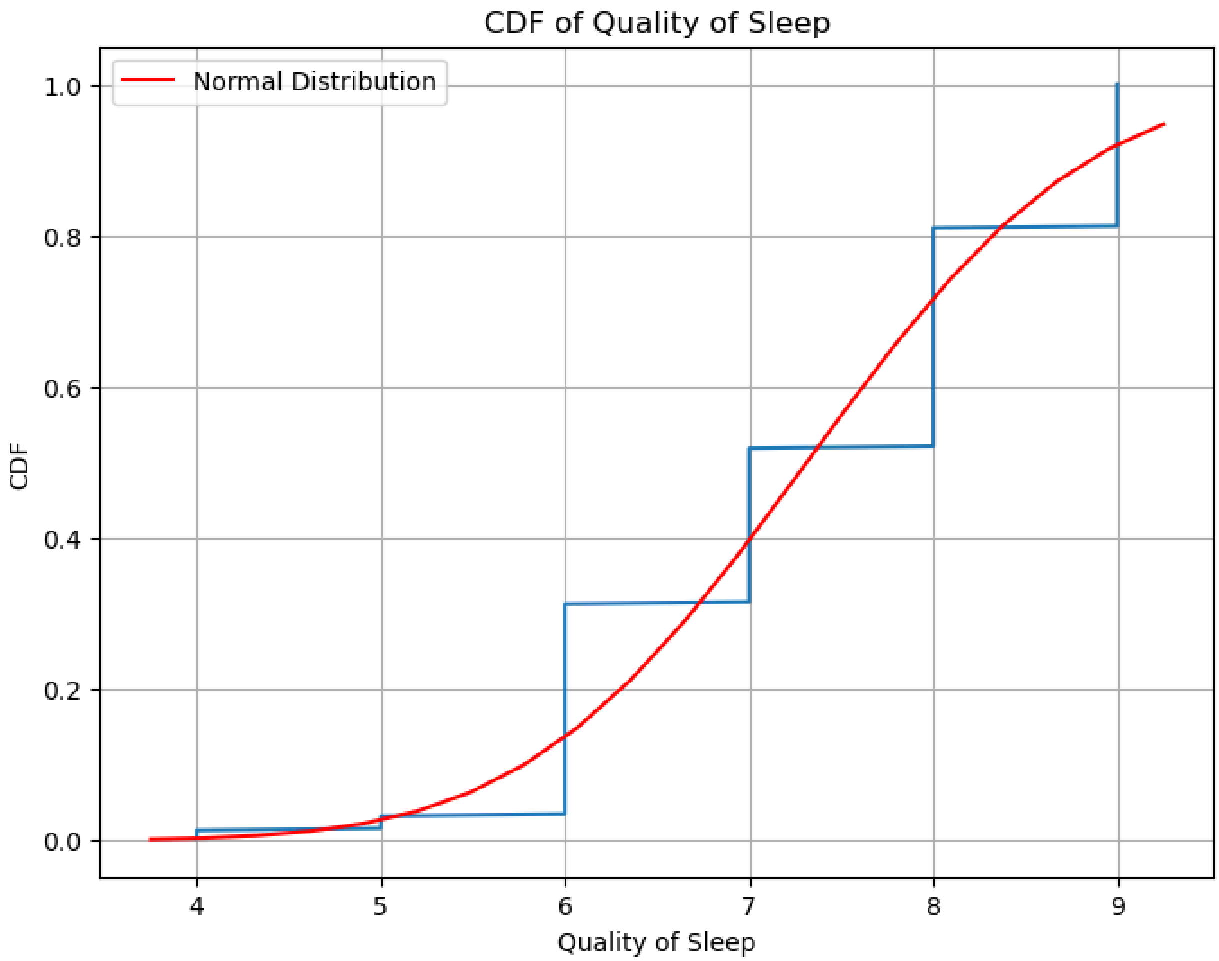


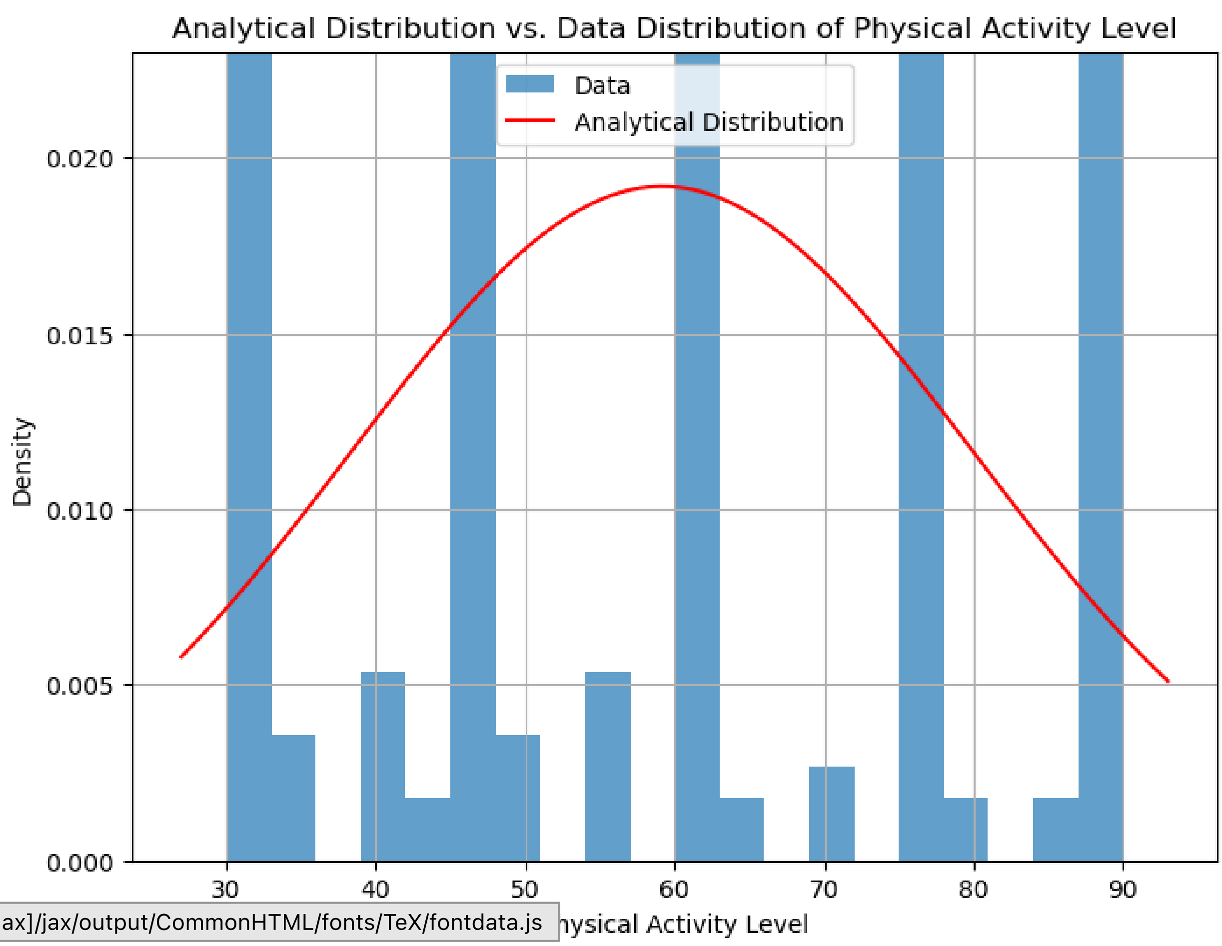
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



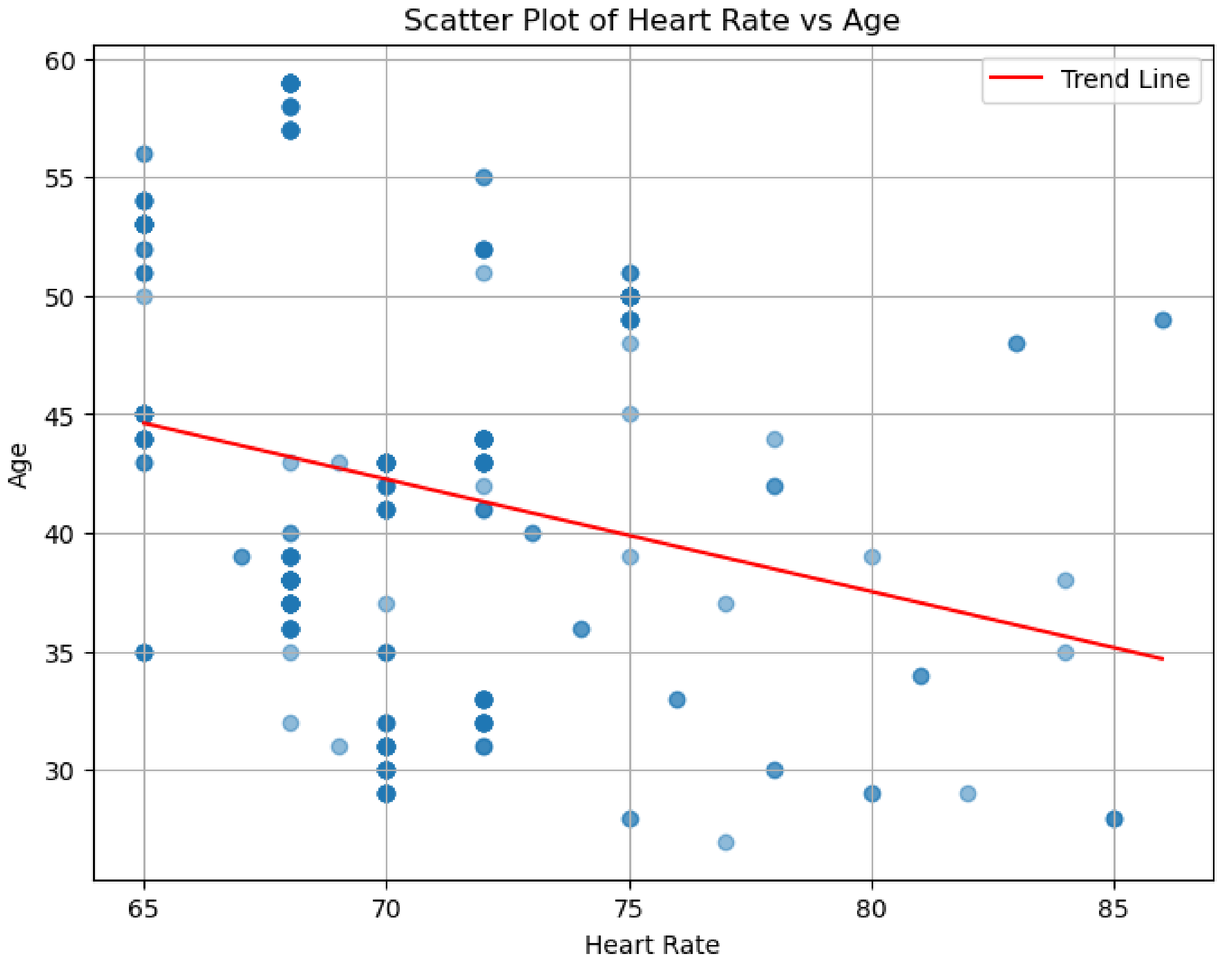


Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

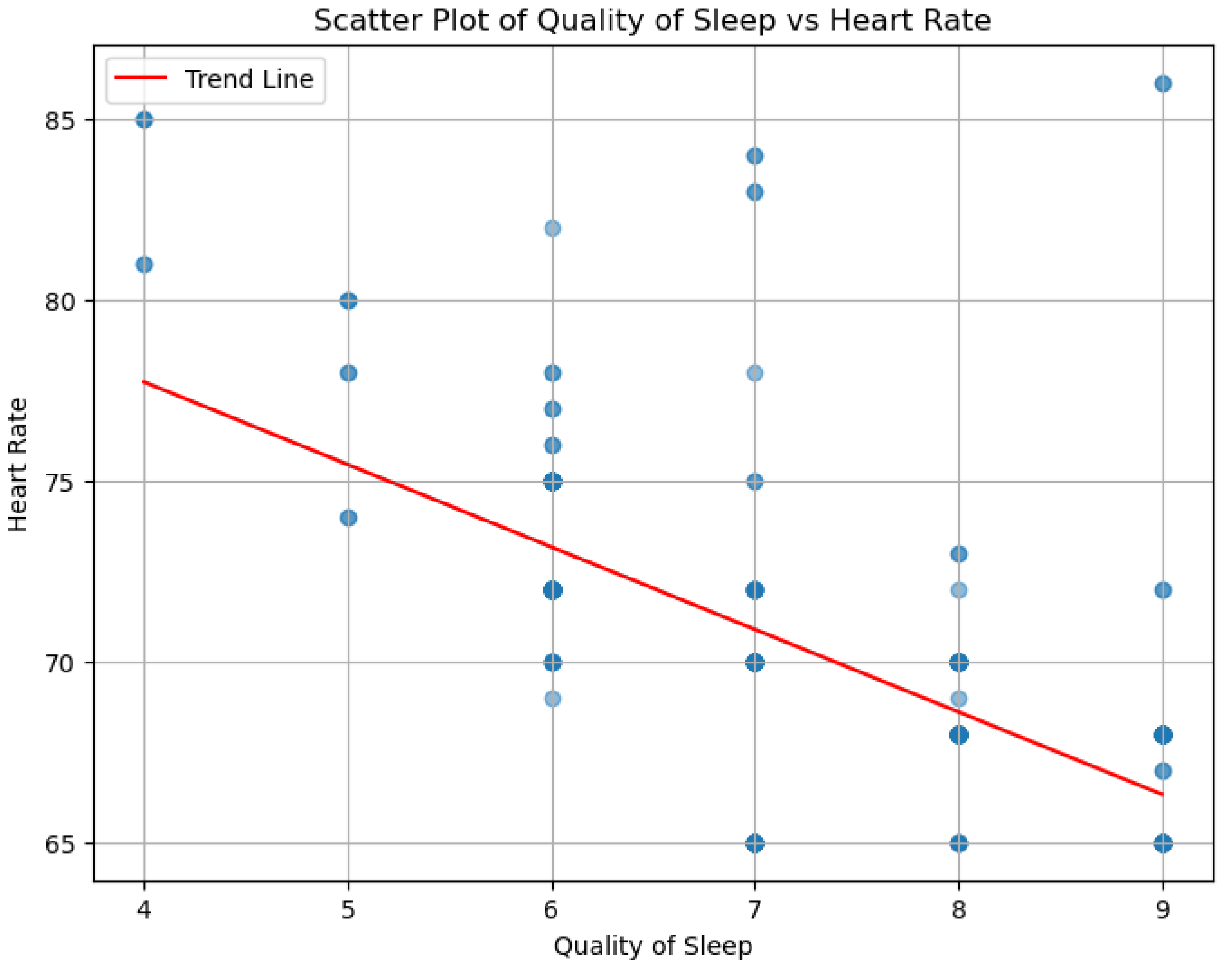




Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
T-Stat: -5.874547760454642
P-Value: 9.416446532689304e-09
Reject null hypothesis: There is a significant difference between the groups.
                            OLS Regression Results
==============================================================================
Dep. Variable:             Heart Rate   R-squared:                       0.269
Model:                            OLS   Adj. R-squared:                  0.265
Method:                 Least Squares   F-statistic:                     68.36
Date:                Sat, 12 Aug 2023   Prob (F-statistic):           5.30e-26
Time:                        20:52:06   Log-Likelihood:                -1002.5
No. Observations:                 374   AIC:                             2011.
Df Residuals:                     371   BIC:                             2023.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
====
                     coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
const             89.7075      1.691     53.039      0.000      86.382      9
3.033
Sleep Duration    -2.5877      0.246    -10.530      0.000      -3.071      -
2.104
Age               -0.0257      0.023     -1.142      0.254      -0.070
0.019
==============================================================================
Omnibus:                      136.960   Durbin-Watson:                   1.076
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              706.619
Skew:                           1.473   Prob(JB):                    3.63e-154
Kurtosis:                       9.055   Cond. No.                         406.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correct
ly specified.
```

In [ ]:

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js