DA_wellnes_tech_company

September 4, 2024

Data Analysis report

Table of Contents

- Introduction
- Business task
- Data Overview
- Data Preparation
- Descriptive and Correlation Analysis
- Recommendations
- Conclusion
- Data Usage Note

1 Introduction

Welcome to the Smart Device company analysis case study! In this report, we delve into the smart device usage data to uncover insights into user behavior and device performance. Our goal is to understand how different user segments interact with the devices and identify opportunities for improvement and innovation. This analysis will provide actionable recommendations to enhance user experience, optimize product features, and guide strategic decisions for product development and marketing.

2 Business task

A high-tech wellness company specializing in health-focused smart products is looking to leverage insights from smart device usage data to enhance its marketing strategy. As a junior data analyst on the marketing analytics team, my role involves analyzing consumer data from various smart devices, including those not produced by the company. The objective is to uncover trends in consumer usage of these devices. These insights will then be applied to one of the company's products to provide high-level recommendations for refining its marketing approach. The analysis should focus on identifying relevant usage trends, understanding their implications for customers, and recommending strategies that could strengthen the company's market positioning and customer engagement.

3 Data Overview

Data Source: The data used for this analysis spans from April 12, 2016, to May 12, 2016. It consists of three CSV files, each containing different types of data: Activity, Sleep, and Weight.

Python: For data cleaning and analysis, Python was utilized to process and integrate these datasets, ensuring accurate insights and comprehensive understanding of user behavior and device performance.

4 Data Preparation

4.1 Data Cleaning

This section details the cleaning and preprocessing steps applied to the **Daily Activity**, **Sleep Day**, and **Weight Log Info** datasets covering the period from April 12, 2016, to May 12, 2016. These steps were essential to ensure the integrity, consistency, and readiness of the data for further analysis.

1. **Loading and Inspecting the Data**:
   - The datasets were loaded into the environment using Pandas. A preliminary inspection was conducted to identify duplicates and missing values.
   - **Duplicate Handling**:
     - The **Sleep Day** dataset contained 3 duplicate rows, which were removed to avoid redundant data entries.
   - **Missing Values**:
     - The **Weight Log Info** dataset had 65 missing values in the 'Fat' column, which represented 97% of the column's data. This column was dropped due to insufficient information.
2. **Feature Engineering**:
   - New columns were introduced in the **Daily Activity** dataset:
     - **active_minutes**: Created by summing `VeryActiveMinutes`, `FairlyActiveMinutes`, and `LightlyActiveMinutes`, providing a comprehensive view of total active minutes.
     - **active_distance**: Calculated by summing `VeryActiveDistance`, `ModeratelyActiveDistance`, and `LightActiveDistance`, offering a complete measure of distance covered during active periods.
   - A new column was introduced in the **Sleep Day** dataset:
     - **NoSleepBedMin** was calculated as the difference between TotalTimeInBed and TotalMinutesAsleep, providing insights into the time participants spent awake in bed.
3. **Date Standardization**:
   - Date columns across all datasets were converted to datetime objects for consistent handling. Dates were then converted back to a date-only format to simplify merging and temporal analysis.
   - New columns were added to extract the day of the week from these dates, ordered from Monday to Sunday to ensure coherent weekly patterns.
4. **Merging Datasets**:
   - **Merging daily_activity and sleep_day**:
     - The Daily Activity and Sleep Day datasets were merged using Pandas' pd.merge function. The merge was performed on Id and date columns: SleepDay from Sleep

Day and ActivityDate from Daily Activity, using an inner join. This merge aligned activity and sleep data by participant and date.

- **Merging daily_activity and weight_info**:
  - The Daily Activity and Weight Log Info datasets were similarly merged on Id and date columns: Date from Weight Log Info and ActivityDate from Daily Activity, also using an inner join. This merge aligned weight data with daily activity metrics.

5. **Final Data Structure**:

- After the initial cleaning and transformation, the datasets were re-inspected to verify changes. The data structures were confirmed to align with the analysis objectives, and the new columns were checked for accuracy.
- The resulting cleaned datasets now serve as a reliable foundation for subsequent analysis, ensuring that insights are based on accurate and well-prepared data.

Importing necessary libraries

[ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings("ignore", category=FutureWarning)
```

Loading Datasets

[ ]:
```
# Load the dataset
daily_activity = pd.read_csv('dailyActivity_4_12_5_12.csv')
sleep_day = pd.read_csv('sleepDay_4_12_5_12.csv')
weight_info = pd.read_csv('weightLogInfo_4_12_5_12.csv')
```

Identifying and removing duplicate data entries

[ ]:
```
# Check for duplicates
print('daily_activity duplicated rows:', daily_activity.duplicated().sum(),
 ↪'\n')
### daily_activity duplicated rows: 0
print('sleep_day duplicated rows:', sleep_day.duplicated().sum(), '\n')
### sleep_day duplicated rows: 3
print('weight_info duplicated rows:', weight_info.duplicated().sum(), '\n')
### weight_info duplicated rows: 0

#Dropping duplicated rows

sleep_day = sleep_day.drop_duplicates()
print('sleep_day duplicated rows after dropping duplicates:', sleep_day.
 ↪duplicated().sum(), '\n')
```

```
daily_activity duplicated rows: 0

sleep_day duplicated rows: 3
```

```
weight_info duplicated rows: 0

sleep_day duplicated rows after dropping duplicates: 0
```

Identifying and handling missing values

```
[ ]: # Check for missing values
print('daily_activity missing values', '\n' , daily_activity.isnull().sum(),␣
 ↪'\n')
### no missing values in daily_activity
print('sleep_day missing values', '\n', sleep_day.isnull().sum(), '\n')
### no missing values in sleep_day
print('weight_info missing values', '\n', weight_info.isnull().sum(), '\n')
### 65 missing values in column 'Fat' in weight_info

#-----------------------------------
# Check for total rows
print('\nTotal rows in daily_activity:', len(daily_activity))
print('\nTotal rows in sleep_day:', len(sleep_day))
print('\nTotal rows in weight_info:', len(weight_info))
print("\nIt's better to remove the 'Fat' column. There are 65 missing values␣
 ↪out of 67.")
weight_info = weight_info.drop('Fat', axis=1)
print("Missing values in 'weight_info' without the 'Fat' column:\n",␣
 ↪weight_info.isnull().sum(), "\n")
```

```
daily_activity missing values
 Id                          0
ActivityDate                0
TotalSteps                  0
TotalDistance               0
TrackerDistance             0
LoggedActivitiesDistance    0
VeryActiveDistance          0
ModeratelyActiveDistance    0
LightActiveDistance         0
SedentaryActiveDistance     0
VeryActiveMinutes           0
FairlyActiveMinutes         0
LightlyActiveMinutes        0
SedentaryMinutes            0
Calories                    0
dtype: int64

sleep_day missing values
 Id                    0
```

```
SleepDay              0
TotalSleepRecords     0
TotalMinutesAsleep    0
TotalTimeInBed        0
dtype: int64

weight_info missing values
 Id                  0
Date               0
WeightKg           0
WeightPounds       0
Fat               65
BMI                0
IsManualReport     0
LogId              0
dtype: int64


Total rows in daily_activity: 940

Total rows in sleep_day: 410

Total rows in weight_info: 67

It's better to remove the 'Fat' column. There are 65 missing values out of 67.
Missing values in 'weight_info' without the 'Fat' column:
 Id                 0
Date              0
WeightKg          0
WeightPounds      0
BMI               0
IsManualReport    0
LogId             0
dtype: int64
```

Analysing data structure

[ ]:
```
# Analysing data structure

print(daily_activity.head())
print(daily_activity.columns.tolist(), '\n')

print(sleep_day.head())
print(sleep_day.columns.tolist(), '\n')

print(weight_info.head())
print(weight_info.columns.tolist(), '\n')
```

```
           Id ActivityDate  TotalSteps  TotalDistance  TrackerDistance  \
0  1503960366    4/12/2016       13162           8.50             8.50
1  1503960366    4/13/2016       10735           6.97             6.97
2  1503960366    4/14/2016       10460           6.74             6.74
3  1503960366    4/15/2016        9762           6.28             6.28
4  1503960366    4/16/2016       12669           8.16             8.16

   LoggedActivitiesDistance  VeryActiveDistance  ModeratelyActiveDistance  \
0                       0.0                1.88                      0.55
1                       0.0                1.57                      0.69
2                       0.0                2.44                      0.40
3                       0.0                2.14                      1.26
4                       0.0                2.71                      0.41

   LightActiveDistance  SedentaryActiveDistance  VeryActiveMinutes  \
0                 6.06                      0.0                 25
1                 4.71                      0.0                 21
2                 3.91                      0.0                 30
3                 2.83                      0.0                 29
4                 5.04                      0.0                 36

   FairlyActiveMinutes  LightlyActiveMinutes  SedentaryMinutes  Calories
0                   13                   328               728      1985
1                   19                   217               776      1797
2                   11                   181              1218      1776
3                   34                   209               726      1745
4                   10                   221               773      1863
['Id', 'ActivityDate', 'TotalSteps', 'TotalDistance', 'TrackerDistance',
'LoggedActivitiesDistance', 'VeryActiveDistance', 'ModeratelyActiveDistance',
'LightActiveDistance', 'SedentaryActiveDistance', 'VeryActiveMinutes',
'FairlyActiveMinutes', 'LightlyActiveMinutes', 'SedentaryMinutes', 'Calories']

           Id              SleepDay  TotalSleepRecords  TotalMinutesAsleep  \
0  1503960366  4/12/2016 12:00:00 AM                  1                 327
1  1503960366  4/13/2016 12:00:00 AM                  2                 384
2  1503960366  4/15/2016 12:00:00 AM                  1                 412
3  1503960366  4/16/2016 12:00:00 AM                  2                 340
4  1503960366  4/17/2016 12:00:00 AM                  1                 700

   TotalTimeInBed
0             346
1             407
2             442
3             367
4             712
['Id', 'SleepDay', 'TotalSleepRecords', 'TotalMinutesAsleep', 'TotalTimeInBed']

           Id                   Date    WeightKg  WeightPounds        BMI  \
```

```
0  1503960366   5/2/2016 11:59:59 PM   52.599998    115.963147  22.650000
1  1503960366   5/3/2016 11:59:59 PM   52.599998    115.963147  22.650000
2  1927972279   4/13/2016 1:08:52 AM  133.500000    294.317120  47.540001
3  2873212765  4/21/2016 11:59:59 PM   56.700001    125.002104  21.450001
4  2873212765  5/12/2016 11:59:59 PM   57.299999    126.324875  21.690001

   IsManualReport          LogId
0            True  1462233599000
1            True  1462319999000
2           False  1460509732000
3            True  1461283199000
4            True  1463097599000
['Id', 'Date', 'WeightKg', 'WeightPounds', 'BMI', 'IsManualReport', 'LogId']
```

How many unique participants does the data frames have?

```
[ ]: #How many unique participants does this data has?
print('\ndayly_activity Number of participants:',daily_activity['Id'].nunique())
print('\nsleep_day Number of participants:',sleep_day['Id'].nunique())
print('\nweight_info Number of participants:',weight_info['Id'].nunique())
```

```
dayly_activity Number of participants: 33

sleep_day Number of participants: 24

weight_info Number of participants: 8
```

Creating columns

Let's create a column called "Active minutes" summing VeryActiveMinutes, FairlyActiveMinutes and LightlyActiveMinutes; a column called "Active distance" summing VeryActiveDistance, ModeratelyActiveDistance and LightActiveDistance and a column called "NoSleepBedMin" as the difference between TotalTimeInBed and TotalMinutesAsleep, providing insights into the time participants spent awake in bed.

```
[ ]: #Let's create a column called "Active minutes" summing VeryActiveMinutes,␣
     ↪FairlyActiveMinutes and LightlyActiveMinutes
daily_activity['active_minutes'] = (daily_activity['VeryActiveMinutes'] +
                                    daily_activity['FairlyActiveMinutes'] +
                                    daily_activity['LightlyActiveMinutes'])

#Let's create a column called "Active distance" summing VeryActiveDistance,␣
↪ModeratelyActiveDistance and LightActiveDistance
daily_activity['active_distance'] = (daily_activity['VeryActiveDistance'] +
                                    daily_activity['ModeratelyActiveDistance'] +
                                    daily_activity['LightActiveDistance'])
```

```
#-----------------------------------
# Display the updated DataFrame to check the new column
print(daily_activity[['VeryActiveMinutes', 'FairlyActiveMinutes',␣
↪'LightlyActiveMinutes', 'active_minutes']].head())

# Display the updated DataFrame to check the new column
print(daily_activity[['VeryActiveDistance', 'ModeratelyActiveDistance',␣
↪'LightActiveDistance', 'active_distance']].head())

#-----------------------------------
#Let's create a column called "NoSleepBedMin" as the difference between␣
↪TotalTimeInBed and TotalMinutesAsleep,
# providing insights into the time participants spent awake in bed.

sleep_day['NoSleepBedMin'] = sleep_day['TotalTimeInBed'] -␣
↪sleep_day['TotalMinutesAsleep']

# Display the updated DataFrame to check the new column
print(sleep_day.head())
```

```
   VeryActiveMinutes  FairlyActiveMinutes  LightlyActiveMinutes  \
0                 25                   13                   328
1                 21                   19                   217
2                 30                   11                   181
3                 29                   34                   209
4                 36                   10                   221

   active_minutes
0             366
1             257
2             222
3             272
4             267
   VeryActiveDistance  ModeratelyActiveDistance  LightActiveDistance  \
0                1.88                      0.55                 6.06
1                1.57                      0.69                 4.71
2                2.44                      0.40                 3.91
3                2.14                      1.26                 2.83
4                2.71                      0.41                 5.04

   active_distance
0             8.49
1             6.97
2             6.75
3             6.23
4             8.16
           Id                SleepDay  TotalSleepRecords  TotalMinutesAsleep  \
```

```
0  1503960366  4/12/2016 12:00:00 AM                   1                 327
1  1503960366  4/13/2016 12:00:00 AM                   2                 384
2  1503960366  4/15/2016 12:00:00 AM                   1                 412
3  1503960366  4/16/2016 12:00:00 AM                   2                 340
4  1503960366  4/17/2016 12:00:00 AM                   1                 700

   TotalTimeInBed  NoSleepBedMin
0             346             19
1             407             23
2             442             30
3             367             27
4             712             12
```

Transforming date columns and creating day of the week column

[ ]:
```
# Convert date columns
daily_activity['ActivityDate'] = pd.to_datetime(daily_activity['ActivityDate'])
sleep_day['SleepDay'] = pd.to_datetime(sleep_day['SleepDay'])
weight_info['Date'] = pd.to_datetime(weight_info['Date'])

# Extract the day of the week
daily_activity['day_of_week'] = daily_activity['ActivityDate'].dt.day_name()
sleep_day['day_of_week_s'] = sleep_day['SleepDay'].dt.day_name()
weight_info['day_of_week_w'] = weight_info['Date'].dt.day_name()

#Changing the dates to date to merge with Weight_info later
daily_activity['ActivityDate'] = pd.to_datetime(daily_activity['ActivityDate']).
  ↪dt.date
sleep_day['SleepDay'] = pd.to_datetime(sleep_day['SleepDay']).dt.date
weight_info['Date'] = pd.to_datetime(weight_info['Date']).dt.date

# Reorder days of the week (from Monday to Sunday)
days_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
  ↪'Saturday', 'Sunday']
daily_activity['day_of_week'] = pd.Categorical(daily_activity['day_of_week'],␣
  ↪categories=days_order, ordered=True)
sleep_day['day_of_week_s'] = pd.Categorical(sleep_day['day_of_week_s'],␣
  ↪categories=days_order, ordered=True)
weight_info['day_of_week_w'] = pd.Categorical(weight_info['day_of_week_w'],␣
  ↪categories=days_order, ordered=True)
```

```
C:\Users\Fran\AppData\Local\Temp\ipykernel_10352\1982826622.py:3: UserWarning:
Could not infer format, so each element will be parsed individually, falling
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  sleep_day['SleepDay'] = pd.to_datetime(sleep_day['SleepDay'])
C:\Users\Fran\AppData\Local\Temp\ipykernel_10352\1982826622.py:4: UserWarning:
Could not infer format, so each element will be parsed individually, falling
```

```
back to `dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  weight_info['Date'] = pd.to_datetime(weight_info['Date'])
```

Merging data

```
[ ]: # Merging daily_activity and sleep_day
day_act_sleep = pd.merge(sleep_day, daily_activity,
                                       left_on=['Id', 'SleepDay'],
                                       right_on=['Id', 'ActivityDate'],
                                       how='inner')

# Check if the merge was successful and the expected columns exist
print(day_act_sleep.head())  # Ensure 'TotalSteps', 'TotalMinutesAsleep', and␣
 ↪'Calories' exist

# Merging daily_activity and weight_info
day_act_wei = pd.merge(weight_info, daily_activity,
                                       left_on=['Id', 'Date'],
                                       right_on=['Id', 'ActivityDate'],
                                       how='inner')

print(day_act_wei.head())
```

```
           Id    SleepDay  TotalSleepRecords  TotalMinutesAsleep  \
0  1503960366  2016-04-12                  1                 327
1  1503960366  2016-04-13                  2                 384
2  1503960366  2016-04-15                  1                 412
3  1503960366  2016-04-16                  2                 340
4  1503960366  2016-04-17                  1                 700

   TotalTimeInBed  NoSleepBedMin day_of_week_s ActivityDate  TotalSteps  \
0             346             19       Tuesday   2016-04-12       13162
1             407             23     Wednesday   2016-04-13       10735
2             442             30        Friday   2016-04-15        9762
3             367             27      Saturday   2016-04-16       12669
4             712             12        Sunday   2016-04-17        9705

   TotalDistance  …  LightActiveDistance  SedentaryActiveDistance  \
0           8.50  …                 6.06                      0.0
1           6.97  …                 4.71                      0.0
2           6.28  …                 2.83                      0.0
3           8.16  …                 5.04                      0.0
4           6.48  …                 2.51                      0.0

   VeryActiveMinutes  FairlyActiveMinutes  LightlyActiveMinutes  \
0                 25                   13                   328
1                 21                   19                   217
```

```
2                  29                   34                    209
3                  36                   10                    221
4                  38                   20                    164

   SedentaryMinutes  Calories  active_minutes  active_distance  day_of_week
0               728      1985             366             8.49      Tuesday
1               776      1797             257             6.97    Wednesday
2               726      1745             272             6.23       Friday
3               773      1863             267             8.16     Saturday
4               539      1728             222             6.48       Sunday

[5 rows x 24 columns]
           Id        Date    WeightKg  WeightPounds        BMI  \
0  1503960366  2016-05-02   52.599998    115.963147  22.650000
1  1503960366  2016-05-03   52.599998    115.963147  22.650000
2  1927972279  2016-04-13  133.500000    294.317120  47.540001
3  2873212765  2016-04-21   56.700001    125.002104  21.450001
4  2873212765  2016-05-12   57.299999    126.324875  21.690001

   IsManualReport          LogId day_of_week_w ActivityDate  TotalSteps  …  \
0            True  1462233599000        Monday   2016-05-02       14727  …
1            True  1462319999000       Tuesday   2016-05-03       15103  …
2           False  1460509732000     Wednesday   2016-04-13         356  …
3            True  1461283199000      Thursday   2016-04-21        8859  …
4            True  1463097599000      Thursday   2016-05-12        7566  …

   LightActiveDistance  SedentaryActiveDistance  VeryActiveMinutes  \
0                 5.92                     0.00                 41
1                 4.88                     0.00                 50
2                 0.25                     0.00                  0
3                 5.47                     0.01                  2
4                 5.11                     0.00                  0

   FairlyActiveMinutes  LightlyActiveMinutes  SedentaryMinutes  Calories  \
0                   15                   277               798      2004
1                   24                   254               816      1990
2                    0                    32               986      2151
3                   10                   371              1057      1970
4                    0                   268               720      1431

   active_minutes  active_distance  day_of_week
0             333             9.70       Monday
1             328             9.66      Tuesday
2              32             0.25    Wednesday
3             383             5.97     Thursday
4             268             5.11     Thursday

[5 rows x 25 columns]
```

```
           Id        Date    WeightKg  WeightPounds        BMI  \
0  1503960366  2016-05-02   52.599998    115.963147  22.650000
1  1503960366  2016-05-03   52.599998    115.963147  22.650000
2  1927972279  2016-04-13  133.500000    294.317120  47.540001
3  2873212765  2016-04-21   56.700001    125.002104  21.450001
4  2873212765  2016-05-12   57.299999    126.324875  21.690001

   IsManualReport          LogId day_of_week_w ActivityDate  TotalSteps  …  \
0            True  1462233599000        Monday   2016-05-02       14727  …
1            True  1462319999000       Tuesday   2016-05-03       15103  …
2           False  1460509732000     Wednesday   2016-04-13         356  …
3            True  1461283199000      Thursday   2016-04-21        8859  …
4            True  1463097599000      Thursday   2016-05-12        7566  …

   LightActiveDistance  SedentaryActiveDistance  VeryActiveMinutes  \
0                 5.92                     0.00                 41
1                 4.88                     0.00                 50
2                 0.25                     0.00                  0
3                 5.47                     0.01                  2
4                 5.11                     0.00                  0

   FairlyActiveMinutes  LightlyActiveMinutes  SedentaryMinutes  Calories  \
0                   15                   277               798      2004
1                   24                   254               816      1990
2                    0                    32               986      2151
3                   10                   371              1057      1970
4                    0                   268               720      1431

   active_minutes  active_distance  day_of_week
0             333             9.70       Monday
1             328             9.66      Tuesday
2              32             0.25    Wednesday
3             383             5.97     Thursday
4             268             5.11     Thursday

[5 rows x 25 columns]
```

5 Descriptive and Correlation Analysis

5.1 Descriptive Statistics

Descriptive statistics were computed for various variables across the datasets, providing a comprehensive summary of the data. This analysis aimed to capture the central tendency, dispersion, and overall distribution of key metrics related to daily activities, sleep, and weight information.

1. **Daily Activity**:
    - Descriptive statistics were calculated for the following variables: `TotalSteps`, `VeryActiveMinutes`, `FairlyActiveMinutes`, `LightlyActiveMinutes`,

`active_minutes`, `SedentaryMinutes`, and `Calories`. The summary statistics include measures such as mean, standard deviation, minimum, and maximum values, which help in understanding the overall activity patterns and energy expenditure of the subjects.
   - **Summary Results**: These statistics provide insights into the average number of steps taken, the duration of various activity levels, and the total calories burned. The data also reveals the extent of sedentary behavior among individuals.
2. **Sleep Data**:
   - For the sleep data, summary statistics were computed for `TotalSleepRecords`, `TotalMinutesAsleep`, and `TotalTimeInBed`. These metrics give a snapshot of sleep patterns, including the total number of sleep records, average minutes asleep, and the total time spent in bed.
   - **Summary Results**: The descriptive statistics provide a clear view of average sleep duration and patterns, which are crucial for understanding sleep quality and behavior.
3. **Weight Information**:
   - Descriptive statistics were obtained for `WeightKg`, `WeightPounds`, and `BMI`. This analysis helps in understanding the distribution of body weight and body mass index across the dataset.
   - **Summary Results**: These statistics offer insights into the average body weight and BMI, which are essential for analyzing health and fitness levels.

5.1.1 Activity Analysis by Day of the Week

To gain insights into daily activity patterns, the data was aggregated by day of the week. The following metrics were analyzed:

1. **Total Steps**:
   - The average number of steps taken per day of the week was calculated. This analysis helps identify which days see the most or least physical activity.
   - **Results**: The average total steps varied by day, providing insights into daily activity trends.
2. **Sedentary Minutes**:
   - The average minutes spent in a sedentary state were calculated by day of the week. This metric highlights the amount of time individuals spend being inactive.
   - **Results**: Differences in sedentary behavior were observed across different days, revealing potential patterns in inactivity.
3. **Calories Burned**:
   - The average calories burned per day of the week were computed. This analysis indicates how daily physical activity correlates with energy expenditure.
   - **Results**: Variations in calorie expenditure across the week were noted, providing insights into energy consumption patterns.

5.1.2 Sleep Analysis by Day of the Week

The sleep data was also analyzed by day of the week to understand sleep patterns better:

1. **Total Sleep Records**:
   - The average number of sleep records per day was calculated. This metric helps to determine the consistency of sleep recording across different days.

- **Results**: Variations in sleep records were observed, indicating potential inconsistencies or patterns in sleep recording.

2. **Total Minutes Asleep**:
   - The average minutes asleep per day of the week were computed. This analysis helps in understanding the amount of sleep individuals get on average.
   - **Results**: Patterns in sleep duration were identified, highlighting any variations in sleep quality.
3. **Total Time in Bed**:
   - The average time spent in bed per day was calculated. This metric provides insight into the total time allocated for sleep and rest.
   - **Results**: Differences in time spent in bed were noted, offering a view of sleep habits.

5.2 Correlation Analysis

Correlation analysis was performed to investigate the relationships between various metrics. The following correlations were computed and visualized:

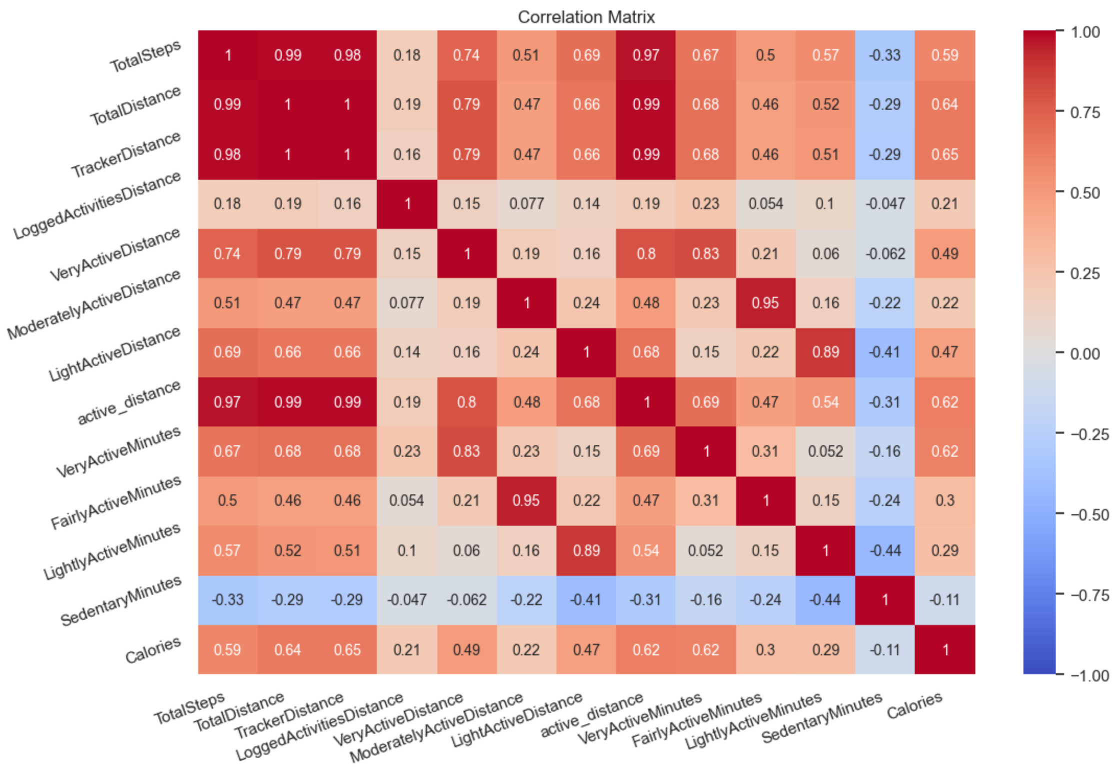
1. **Correlation Matrix**:
   - A correlation matrix was created for variables related to daily activity, including `TotalSteps`, `TotalDistance`, `VeryActiveDistance`, `ModeratelyActiveDistance`, `LightActiveDistance`, `Calories`, and others. This matrix helps identify how different activity metrics are interrelated.
   - **Visualization**: A heatmap was generated to visually represent these correlations, with key observations such as high correlations between `TotalSteps` and `TotalDistance`, as well as between `VeryActiveDistance` and `VeryActiveMinutes`.
2. **Activity and Sleep Correlations**:
   - The correlations between `VeryActiveMinutes` and `NoSleepBedMin` (a new variable representing the difference between total time in bed and minutes asleep) and between `TotalMinutesAsleep` and `Calories` were examined.
   - **Visualization**: Scatter plots with regression lines were used to visualize these correlations, providing insights into how active minutes and sleep metrics relate to calories.
3. **Weight and Activity Correlations**:
   - Correlations between `BMI` and `TotalSteps`, as well as between `BMI` and `Calories`, were analyzed to understand the relationship between body metrics and activity levels.
   - **Visualization**: Regression plots were created to visualize these relationships, showing how body mass index correlates with physical activity and caloric expenditure.

5.2.1 Summary

The descriptive and correlation analyses provided a comprehensive understanding of the dataset, revealing patterns and relationships between various metrics. Descriptive statistics offered insights into daily activity, sleep, and weight information, while correlation analysis highlighted significant relationships between these variables. Visualizations such as heatmaps and scatter plots enhanced the interpretation of these analyses, offering valuable insights into activity patterns, sleep behavior, and body metrics.

Correlation Matrix

```
[ ]: # Calculate the correlation matrix
correlation_matrix = daily_activity[['TotalSteps', 'TotalDistance',␣
 ↪'TrackerDistance', 'LoggedActivitiesDistance',
                                                        ␣
 ↪'VeryActiveDistance','ModeratelyActiveDistance', 'LightActiveDistance',
                                     'active_distance', 'VeryActiveMinutes',␣
 ↪'FairlyActiveMinutes',
                                     'LightlyActiveMinutes',␣
 ↪'SedentaryMinutes', 'Calories' ]].corr()


# Plot the heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1,␣
 ↪annot_kws={"size": 10})
plt.xticks(rotation=20, ha='right')  # Rotate x labels
plt.yticks(rotation=20)  # Rotate y labels
plt.title('Correlation Matrix')
plt.show()


#Total steps is highly correlated with TotalDistance, TrackerDistance and␣
 ↪active_distance.
#VeryActiveDistance is highly correlated with active_distance and␣
 ↪VeryActiveMinutes.
#ModeratelyActiveMinutes is highly correlated with FairlyActiveMinutes.
#LightActiveDistance is highly correlated with LightlyActiveMinutes.
```



Descriptive Summaries

```
[ ]: #Summaries
summary_stats_da = daily_activity[['TotalSteps', 'VeryActiveMinutes', 'FairlyActiveMinutes',
                                   'LightlyActiveMinutes', 'active_minutes', 'SedentaryMinutes',
                                   'Calories']].describe()
print(summary_stats_da)

summary_stats_sd = sleep_day[['TotalSleepRecords', 'TotalMinutesAsleep', 'TotalTimeInBed']].describe()
print(summary_stats_sd)

summary_stats_wi = weight_info[['WeightKg', 'WeightPounds', 'BMI']].describe()
print(summary_stats_wi)



average_steps_by_day = daily_activity.groupby('day_of_week')['TotalSteps'].mean().reset_index()
```

```
average_sedentarymin_by_day = daily_activity.
 ↪groupby('day_of_week')['SedentaryMinutes'].mean().reset_index()
average_calories_by_day = daily_activity.groupby('day_of_week')['Calories'].
 ↪mean().reset_index()

print(average_steps_by_day)
print(average_sedentarymin_by_day)
print(average_calories_by_day)

average_sleep_rec_by_day = sleep_day.
 ↪groupby('day_of_week_s')['TotalSleepRecords'].mean().reset_index()
average_min_aspleep_by_day = sleep_day.
 ↪groupby('day_of_week_s')['TotalMinutesAsleep'].mean().reset_index()
average_time_bed_by_day = sleep_day.groupby('day_of_week_s')['TotalTimeInBed'].
 ↪mean().reset_index()

print(average_sleep_rec_by_day)
print(average_min_aspleep_by_day)
print(average_time_bed_by_day)
```

```
        TotalSteps  VeryActiveMinutes  FairlyActiveMinutes  \
count   940.000000         940.000000           940.000000
mean   7637.910638          21.164894            13.564894
std    5087.150742          32.844803            19.987404
min       0.000000           0.000000             0.000000
25%    3789.750000           0.000000             0.000000
50%    7405.500000           4.000000             6.000000
75%   10727.000000          32.000000            19.000000
max   36019.000000         210.000000           143.000000

       LightlyActiveMinutes  active_minutes  SedentaryMinutes     Calories
count            940.000000      940.000000        940.000000   940.000000
mean             192.812766      227.542553        991.210638  2303.609574
std              109.174700      121.776307        301.267437   718.166862
min                0.000000        0.000000          0.000000     0.000000
25%              127.000000      146.750000        729.750000  1828.500000
50%              199.000000      247.000000       1057.500000  2134.000000
75%              264.000000      317.250000       1229.500000  2793.250000
max              518.000000      552.000000       1440.000000  4900.000000
       TotalSleepRecords  TotalMinutesAsleep  TotalTimeInBed
count         410.000000          410.000000      410.000000
mean            1.119512          419.173171      458.482927
std             0.346636          118.635918      127.455140
min             1.000000           58.000000       61.000000
25%             1.000000          361.000000      403.750000
50%             1.000000          432.500000      463.000000
75%             1.000000          490.000000      526.000000
```

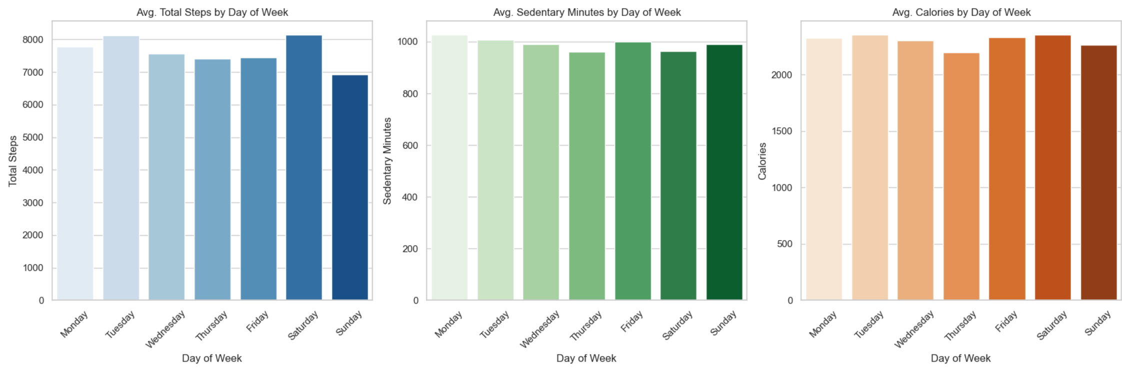
```
max             3.000000          796.000000      961.000000
          WeightKg  WeightPounds        BMI
count   67.000000     67.000000  67.000000
mean    72.035821    158.811801  25.185224
std     13.923206     30.695415   3.066963
min     52.599998    115.963147  21.450001
25%     61.400002    135.363832  23.959999
50%     62.500000    137.788914  24.389999
75%     85.049999    187.503152  25.559999
max    133.500000    294.317120  47.540001
  day_of_week   TotalSteps
0      Monday  7780.866667
1     Tuesday  8125.006579
2   Wednesday  7559.373333
3    Thursday  7405.836735
4      Friday  7448.230159
5    Saturday  8152.975806
6      Sunday  6933.231405
  day_of_week  SedentaryMinutes
0      Monday       1027.941667
1     Tuesday       1007.361842
2   Wednesday        989.480000
3    Thursday        961.993197
4      Friday       1000.309524
5    Saturday        964.282258
6      Sunday        990.256198
  day_of_week     Calories
0      Monday  2324.208333
1     Tuesday  2356.013158
2   Wednesday  2302.620000
3    Thursday  2199.571429
4      Friday  2331.785714
5    Saturday  2354.967742
6      Sunday  2263.000000
  day_of_week_s  TotalSleepRecords
0        Monday           1.108696
1       Tuesday           1.107692
2     Wednesday           1.151515
3      Thursday           1.031250
4        Friday           1.070175
5      Saturday           1.192982
6        Sunday           1.181818
  day_of_week_s  TotalMinutesAsleep
0        Monday           419.500000
1       Tuesday           404.538462
2     Wednesday           434.681818
3      Thursday           401.296875
4        Friday           405.421053
```

```
5      Saturday          419.070175
6        Sunday          452.745455
  day_of_week_s  TotalTimeInBed
0        Monday      457.347826
1       Tuesday      443.292308
2     Wednesday      470.030303
3      Thursday      434.875000
4        Friday      445.052632
5      Saturday      459.842105
6        Sunday      503.509091
```

Activity during the week plots

[ ]:
```
#Activity_plot
# Set the plot style
sns.set(style="whitegrid")

# Create a figure with 3 subplots in one row
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True)

# Plot 1: Day of Week vs Total Steps
sns.barplot(x='day_of_week', y='TotalSteps', data=average_steps_by_day,␣
↪ax=axes[0], palette="Blues", errorbar = None)
axes[0].set_title('Avg. Total Steps by Day of Week')
axes[0].set_xlabel('Day of Week')
axes[0].set_ylabel('Total Steps')
axes[0].tick_params(axis='x', rotation=45)  # Rotate x-axis labels


# Plot 2: Day of Week vs Sedentary Minutes
sns.barplot(x='day_of_week', y='SedentaryMinutes',␣
↪data=average_sedentarymin_by_day, ax=axes[1], palette="Greens", errorbar =␣
↪None)
axes[1].set_title('Avg. Sedentary Minutes by Day of Week')
axes[1].set_xlabel('Day of Week')
axes[1].set_ylabel('Sedentary Minutes')
axes[1].tick_params(axis='x', rotation=45)  # Rotate x-axis labels

# Plot 3: Day of Week vs Calories
sns.barplot(x='day_of_week', y='Calories', data=average_calories_by_day,␣
↪ax=axes[2], palette="Oranges", errorbar = None)
axes[2].set_title('Avg. Calories by Day of Week')
axes[2].set_xlabel('Day of Week')
axes[2].set_ylabel('Calories')
axes[2].tick_params(axis='x', rotation=45)  # Rotate x-axis labels

# Adjust the layout for better spacing
plt.tight_layout()
```

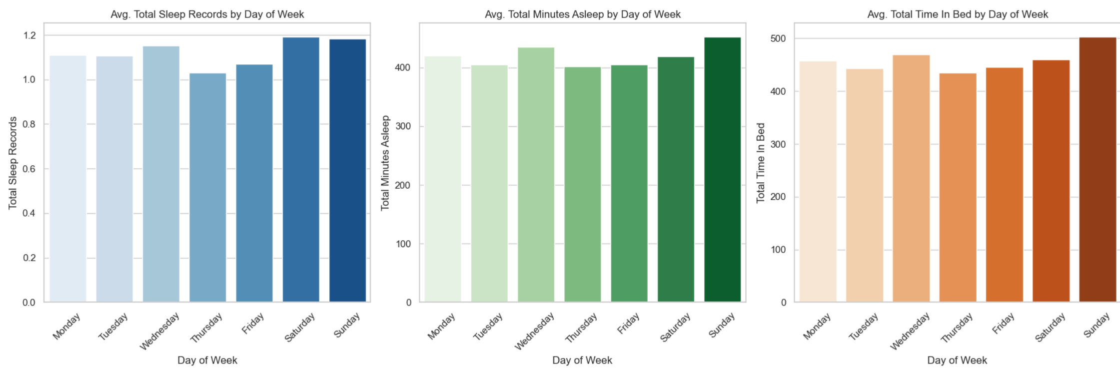
```
# Show the plots
plt.show()
```



Sleep During Week Plots

```
[ ]: #Sleep_plot
# Set the plot style
sns.set(style="whitegrid")

# Create a figure with 3 subplots in one row
fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True)

# Plot 1: Day of Week vs Total Sleep Records
sns.barplot(x='day_of_week_s', y='TotalSleepRecords', data=sleep_day,␣
 ↪ax=axes[0], palette="Blues", errorbar = None)
axes[0].set_title('Avg. Total Sleep Records by Day of Week')
axes[0].set_xlabel('Day of Week')
axes[0].set_ylabel('Total Sleep Records')
axes[0].tick_params(axis='x', rotation=45)  # Rotate x-axis labels

# Plot 2: Day of Week vs Total Minutes Asleep
sns.barplot(x='day_of_week_s', y='TotalMinutesAsleep', data=sleep_day,␣
 ↪ax=axes[1], palette="Greens", errorbar = None)
axes[1].set_title('Avg. Total Minutes Asleep by Day of Week')
axes[1].set_xlabel('Day of Week')
axes[1].set_ylabel('Total Minutes Asleep')
axes[1].tick_params(axis='x', rotation=45)  # Rotate x-axis labels

# Plot 3: Day of Week vs Total Time In Bed
sns.barplot(x='day_of_week_s', y='TotalTimeInBed', data=sleep_day, ax=axes[2],␣
 ↪palette="Oranges", errorbar = None)
axes[2].set_title('Avg. Total Time In Bed by Day of Week')
axes[2].set_xlabel('Day of Week')
```

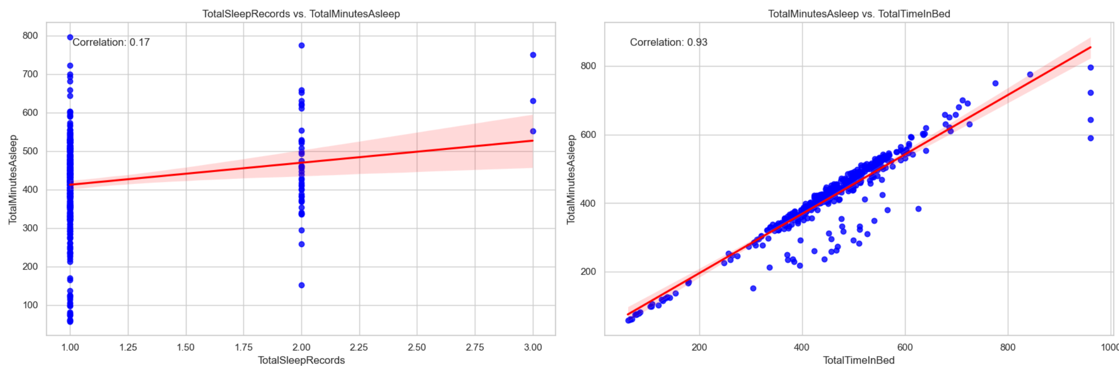
```
axes[2].set_ylabel('Total Time In Bed')
axes[2].tick_params(axis='x', rotation=45)  # Rotate x-axis labels

# Adjust the layout for better spacing
plt.tight_layout()

# Show the plots
plt.show()
```



Sleep Correlations

```
[ ]: correlation_1 = sleep_day['TotalSleepRecords'].
      ↪corr(sleep_day['TotalMinutesAsleep'])
     correlation_2 = sleep_day['TotalMinutesAsleep'].
      ↪corr(sleep_day['TotalTimeInBed'])


     # Create a figure with 3 subplots in one row
     fig, axes = plt.subplots(1, 2, figsize=(18, 6))

     # Plot 1: Calories vs. Total Steps
     sns.regplot(x='TotalSleepRecords', y='TotalMinutesAsleep', data=sleep_day,␣
      ↪scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=axes[0])
     axes[0].set_title('TotalSleepRecords vs. TotalMinutesAsleep')
     axes[0].set_xlabel('TotalSleepRecords')
     axes[0].set_ylabel('TotalMinutesAsleep')
     axes[0].text(0.05, 0.95, f'Correlation: {correlation_1:.2f}', transform=axes[0].
      ↪transAxes, fontsize=12, verticalalignment='top')
     axes[0].grid(True)

     # Plot 2: Calories vs. Very Active Minutes
     sns.regplot(x='TotalTimeInBed', y='TotalMinutesAsleep', data=sleep_day,␣
      ↪scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=axes[1])
     axes[1].set_title('TotalMinutesAsleep vs. TotalTimeInBed')
```

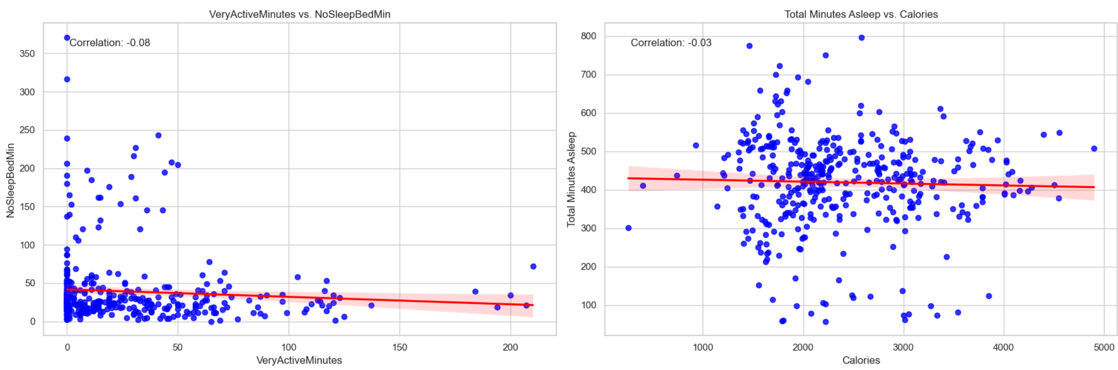
```
axes[1].set_xlabel('TotalTimeInBed')
axes[1].set_ylabel('TotalMinutesAsleep')
axes[1].text(0.05, 0.95, f'Correlation: {correlation_2:.2f}', transform=axes[1].
    ↪transAxes, fontsize=12, verticalalignment='top')
axes[1].grid(True)

# Adjust layout for better spacing
plt.tight_layout()

# Show the combined plot
plt.show()
```



Activity vs Sleep correlations

[ ]:
```
# Calculate correlations
correlation_3 = day_act_sleep['VeryActiveMinutes'].
    ↪corr(day_act_sleep['NoSleepBedMin'])
correlation_4 = day_act_sleep['TotalMinutesAsleep'].
    ↪corr(day_act_sleep['Calories'])




# Create a figure with 2 subplots in one row
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1: Total Steps vs. Total Minutes Asleep
sns.regplot(x='VeryActiveMinutes', y='NoSleepBedMin', data=day_act_sleep,␣
    ↪scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=axes[0])
axes[0].set_title('VeryActiveMinutes vs. NoSleepBedMin')
axes[0].set_xlabel('VeryActiveMinutes')
axes[0].set_ylabel('NoSleepBedMin')
axes[0].text(0.05, 0.95, f'Correlation: {correlation_3:.2f}', transform=axes[0].
    ↪transAxes, fontsize=12, verticalalignment='top')
axes[0].grid(True)
```

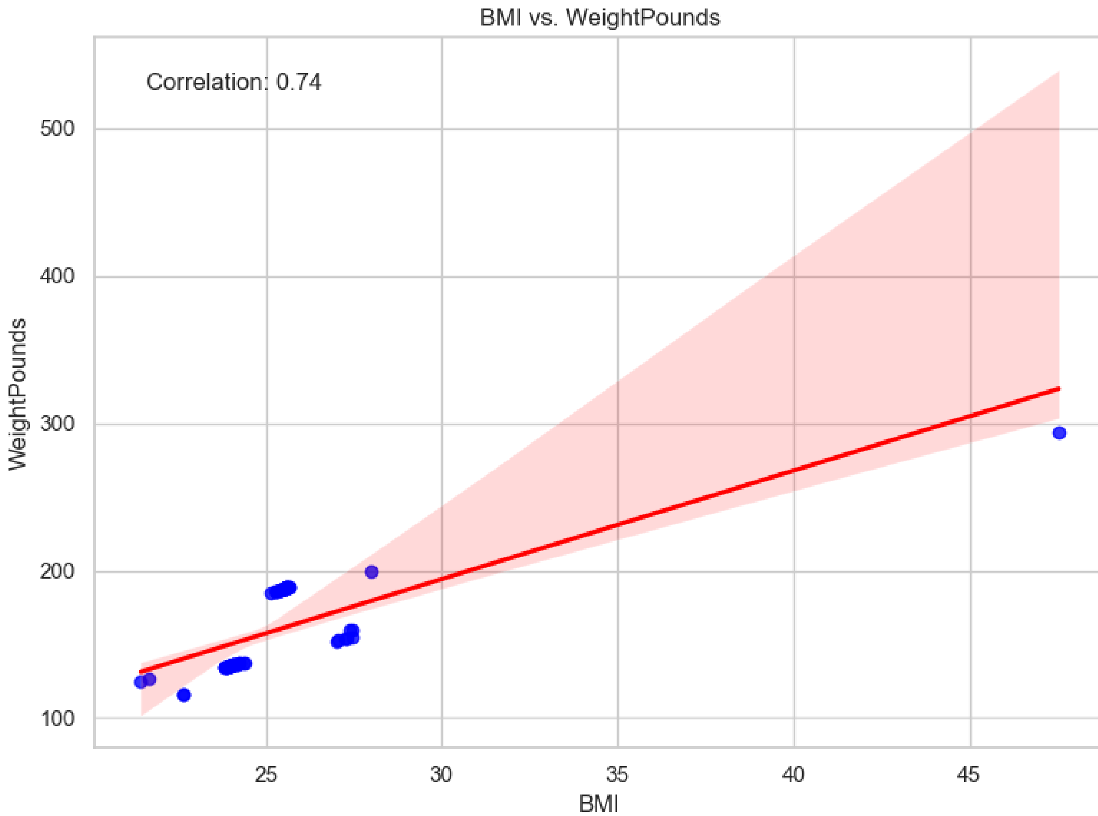
```
# Plot 2: Calories vs. Total Minutes Asleep
sns.regplot(x='Calories', y='TotalMinutesAsleep', data=day_act_sleep,␣
↪scatter_kws={'color': 'blue'}, line_kws={'color': 'red'}, ax=axes[1])
axes[1].set_title('Total Minutes Asleep vs. Calories')
axes[1].set_xlabel('Calories')
axes[1].set_ylabel('Total Minutes Asleep')
axes[1].text(0.05, 0.95, f'Correlation: {correlation_4:.2f}', transform=axes[1].
↪transAxes, fontsize=12, verticalalignment='top')
axes[1].grid(True)

# Adjust layout for better spacing
plt.tight_layout()

# Show the combined plot
plt.show()
```



Weight Correlations

[ ]:
```
correlation_5 = weight_info['BMI'].corr(weight_info['WeightPounds'])

# Plot: BMI vs. WeightKg
plt.figure(figsize=(8, 6))  # Set the figure size
sns.regplot(x='BMI', y='WeightPounds', data=weight_info, scatter_kws={'color':␣
↪'blue'}, line_kws={'color': 'red'})
plt.title('BMI vs. WeightPounds')
plt.xlabel('BMI')
plt.ylabel('WeightPounds')
plt.text(0.05, 0.95, f'Correlation: {correlation_5:.2f}', transform=plt.gca().
↪transAxes, fontsize=12, verticalalignment='top')
plt.grid(True)

# Show the plot
plt.tight_layout()
```

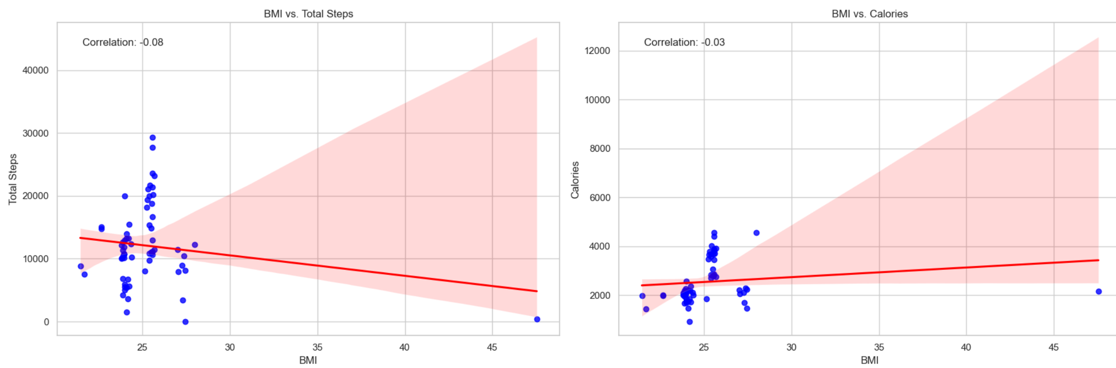
```
plt.show()
```



Activity vs BMI correlation

```
[ ]: # Create a figure with 2 subplots in one row
fig, axes = plt.subplots(1, 2, figsize=(18, 6))

# Plot 1: Total Steps vs. Total Minutes Asleep
sns.regplot(x='BMI', y='TotalSteps', data=day_act_wei, scatter_kws={'color':␣
 ↪'blue'}, line_kws={'color': 'red'}, ax=axes[0])
axes[0].set_title('BMI vs. Total Steps')
axes[0].set_xlabel('BMI')
axes[0].set_ylabel('Total Steps')
axes[0].text(0.05, 0.95, f'Correlation: {correlation_3:.2f}', transform=axes[0].
 ↪transAxes, fontsize=12, verticalalignment='top')
axes[0].grid(True)

# Plot 2: Calories vs. Total Minutes Asleep
sns.regplot(x='BMI', y='Calories', data=day_act_wei, scatter_kws={'color':␣
 ↪'blue'}, line_kws={'color': 'red'}, ax=axes[1])
axes[1].set_title('BMI vs. Calories')
```

```python
axes[1].set_xlabel('BMI')
axes[1].set_ylabel('Calories')
axes[1].text(0.05, 0.95, f'Correlation: {correlation_4:.2f}', transform=axes[1].
↪transAxes, fontsize=12, verticalalignment='top')
axes[1].grid(True)

# Adjust layout for better spacing
plt.tight_layout()

# Show the combined plot
plt.show()
```



6 Recommendations

Based on the analysis, the following actions are recommended:

6.1 Optimize Activity Based on Day of the Week

- **Boost Weekend and Low-Activity Day Engagement:** Increase activity on Sundays and Thursdays with motivational programs or challenges. Utilize high-activity days (Saturdays and Tuesdays) for promotional campaigns to maintain engagement throughout the week.

6.2 Enhance Sleep Quality

- **Improve Sleep Consistency:** Address low TotalSleepRecords on Thursdays by promoting regular sleep habits and using sleep tracking tools. Highlight the benefits of consistent sleep patterns.

6.3 Adjust Caloric Intake and Activity Balance

- **Personalize Caloric Goals:** Align caloric intake recommendations with daily activity levels. Develop strategies to reduce sedentary behavior and manage calorie expenditure effectively.

6.4 Promote Active Minutes

- **Encourage Active Distances:** Utilize data on VeryActiveDistance and active_minutes to promote goals and challenges. Focus on increasing both very active and moderately active minutes through targeted programs.

6.5 Refine Weight Management

- **Tailor Weight Management Plans:** Use BMI and weight data to create personalized weight management strategies. Align caloric intake recommendations with BMI goals.

6.6 Enhance Wellness Programs

- **Leverage Data for Personalization:** Use data insights to tailor wellness programs and adjust strategies based on user feedback and data trends.

7 Conclusion

Significant insights into user activity patterns, sleep quality, and caloric expenditure have been revealed through this analysis. Targeted strategies can be developed by leveraging these insights to enhance user engagement, optimize health and wellness programs, and improve overall user satisfaction. By implementing these recommendations, activities can be better aligned with user needs and preferences, ultimately leading to improved outcomes and engagement.

8 Data Usage Note

Data Licence Link: The dataset is in the public domain (CC0: Public Domain). More information can be found on Kaggle here.

Data Source: The data used in this analysis is sourced from the FitBit Fitness Tracker dataset available on Kaggle. This dataset includes personal fitness tracker data from thirty Fitbit users, encompassing minute-level outputs for physical activity, heart rate, and sleep monitoring.

Purpose: This report is intended for educational and portfolio demonstration purposes only. It is not intended for commercial use.

Data Usage: The data has been utilized to demonstrate data visualization and analytical skills. It is not sold or distributed as a standalone product.

Affiliation: This work is independent and is not affiliated with, endorsed by, or sponsored by Fitbit or Kaggle.

Trademark Notice: No logos or trademarks of Fitbit are used in this analysis. All trademarks and logos are the property of their respective owners.

Data Access: The data was accessed through Kaggle's dataset repository and made available through Mobius.