Case Study - (Bella Beat) How Can a Wellness Technology Company Play It Smart?

This is a capstone project for Google Data Analytics Professional Certificate Course

Bellabeat is a high-tech manufacturer of beautifully-designed health-focused smart products for women since 2013. Inspiring and empowering women with knowledge about their own health and habits, Bellabeat has grown rapidly and quickly positioned itself as a tech-driven wellness company for females.

The co-founder and Chief Creative Officer, Urška Sršen is confident that an analysis of non-Bellebeat consumer data (ie. FitBit fitness tracker usage data) would reveal more opportunities for growth.

I will be using 6 phases for the analysis such as Ask, Prepare, Process, Analyse, Share and Act.

Phase 1: ASK

Business Task Summary:

Analyzing Smart Device Fitness Data:

The junior data analyst is tasked with analyzing smart device data related to Bellabeat's products. This analysis aims to uncover insights into consumer behavior and usage patterns of smart wellness devices like the Leaf, Time, and Spring. The goal is to identify trends and patterns in how consumers interact with these devices through the Bellabeat app. This data will provide valuable insights that can inform Bellabeat's marketing strategies.

Developing Marketing Strategy Recommendations:

Based on the analysis of smart device data, the data analyst must formulate high-level recommendations for Bellabeat's marketing strategy. These recommendations should align with Bellabeat's mission of empowering women with health-focused technology and capitalize on the identified consumer trends. The aim is to propose strategic initiatives that enhance customer engagement, improve product relevance, and drive growth in the smart device market.

Key Stakeholders:

Urška Sršen: As Bellabeat's cofounder and Chief Creative Officer, Sršen drives the company's vision and growth strategy. She has prioritized the analysis of smart device data to unlock new growth opportunities.

Sando Mur: As a mathematician and cofounder, Mur plays a crucial role in strategic decision-making. He will likely be involved in evaluating and implementing the marketing strategy recommendations derived from the data analysis.

Bellabeat Marketing Analytics Team: This team, including the junior data analyst, is responsible for collecting, analyzing, and interpreting data to support Bellabeat's marketing initiatives. Their insights will shape how Bellabeat engages with its customers and markets its products effectively.

Bellabeat Customers: Ultimately, the insights gained from the data analysis will impact Bellabeat's customers. The goal is to enhance customer satisfaction by tailoring marketing efforts to better meet their health and wellness needs.

Phase 2: Prepare

Determine the credibility of the data

It is stored in CSV files and organized in long format. The dataset provides minute-level outputs for physical activity, heart rate, and sleep monitoring from 30 FitBit users. There are notable concerns about its reliability, originality, and comprehensiveness. The data, collected via Amazon Mechanical Turk, lacks representativeness and may not reflect current trends, being eight years old. Its use is cautioned due to potential biases and limitations in relevance, suggesting that strategic decisions based on this dataset should be approached cautiously.

The following files are selected for analysis using Python as downloaded from kaggle:

- dailyActivity_merged.csv
- sleepDay_merged.csv
- heartrate_seconds_merged.csv

Import neccessary libraries

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Load the dataset

In [ ]:
```
daily_activity_df = pd.read_csv('data/dailyActivity_merged.csv')
sleep_day_df = pd.read_csv('data/sleepDay_merged.csv')
heart_rate_df = pd.read_csv('data/heartrate_seconds_merged.csv')
```

Phase 3: Process

Data cleaning

Handling missing value

In [ ]:
```
# Change column name to lower case for ease of use
daily_activity_df.columns = daily_activity_df.columns.str.lower()
sleep_day_df.columns = sleep_day_df.columns.str.lower()
heart_rate_df.columns = heart_rate_df.columns.str.lower()
```

In [ ]:
```
# Store dataframes in a dictionary
df_dict = {
    'Daily Activity': daily_activity_df,
    'Sleep Day': sleep_day_df,
    'Heart Rate': heart_rate_df
}
```

In [ ]:
```
# Check for missing values
for name, df in df_dict.items():
    print(f"{name} Missing Values:")
    print(df.isnull().sum())
```

```
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
Heart Rate Missing Values:
id       0
time     0
value    0
dtype: int64
```

Remove duplicates

In [ ]:
```
# Drop duplicate and reset index
for name, df in df_dict.items():
    df.drop_duplicates(inplace=True)
    df.reset_index(inplace=True, drop=True)
```

Handling outliers

It is common for some highly active individuals to be present in the dataset, so I chose to remove them to avoid skewing the data too much.

In [ ]:
```
def remove_outliers(df):
    """
    Remove outliers from a DataFrame based on the IQR method.
    """
    # Select numeric column
    numeric_cols = df.select_dtypes(include=['number']).columns
    df_numeric = df[numeric_cols]

    # Calculate Q1 and Q3
    Q1 = df_numeric.quantile(0.25)
    Q3 = df_numeric.quantile(0.75)
    IQR = Q3 - Q1

    # Remove outlier
    condition = ~((df_numeric < (Q1 - 1.5 * IQR)) | (df_numeric > (Q3 + 1.5 * IQR))).any(axis=1)
    df = df[condition]

    return df
```

In [ ]:
```
remove_outliers(daily_activity_df) # Test function
```

Out[ ]:

	id	activitydate	totalsteps	totaldistance	trackerdistance	loggedactivitiesdistance	veryactivedistance	moderatelyactivedistance	lightactivedistance	sedentaryactivedistance	veryacti
0	1503960366	4/12/2016	13162	8.50	8.50	0.0	1.88	0.55	6.06	0.0	
1	1503960366	4/13/2016	10735	6.97	6.97	0.0	1.57	0.69	4.71	0.0	
2	1503960366	4/14/2016	10460	6.74	6.74	0.0	2.44	0.40	3.91	0.0	
3	1503960366	4/15/2016	9762	6.28	6.28	0.0	2.14	1.26	2.83	0.0	
4	1503960366	4/16/2016	12669	8.16	8.16	0.0	2.71	0.41	5.04	0.0	
...	...	...	...	...	...	...	...	...	...	...	
928	8877689391	5/1/2016	10930	8.32	8.32	0.0	3.13	0.57	4.57	0.0	
929	8877689391	5/2/2016	4790	3.64	3.64	0.0	0.00	0.00	3.56	0.0	
935	8877689391	5/8/2016	10686	8.11	8.11	0.0	1.08	0.20	6.80	0.0	
937	8877689391	5/10/2016	10733	8.15	8.15	0.0	1.35	0.46	6.28	0.0	
939	8877689391	5/12/2016	8064	6.12	6.12	0.0	1.82	0.04	4.25	0.0	

671 rows × 15 columns

In [ ]:
```
# Apply function to all dataframe
for name, df in df_dict.items():
    print(f"{name} removed outliers:")
    remove_outliers(df)
    print(df)
```

```
Daily Activity removed outliers:
             id activitydate  totalsteps  totaldistance  trackerdistance  \
0    1503960366    4/12/2016       13162       8.500000         8.500000   
1    1503960366    4/13/2016       10735       6.970000         6.970000   
2    1503960366    4/14/2016       10460       6.740000         6.740000   
3    1503960366    4/15/2016        9762       6.280000         6.280000   
4    1503960366    4/16/2016       12669       8.160000         8.160000   
..          ...          ...         ...            ...              ...   
935  8877689391     5/8/2016       10686       8.110000         8.110000   
936  8877689391     5/9/2016       20226      18.250000        18.250000   
937  8877689391    5/10/2016       10733       8.150000         8.150000   
938  8877689391    5/11/2016       21420      19.559999        19.559999   
939  8877689391    5/12/2016        8064       6.120000         6.120000   

     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   
..                        ...                 ...                       ...   
935                       0.0                1.08                      0.20   
936                       0.0               11.10                      0.80   
937                       0.0                1.35                      0.46   
938                       0.0               13.22                      0.41   
939                       0.0                1.82                      0.04   

     lightactivedistance  sedentaryactivedistance  veryactiveminutes  \
0                   6.06                     0.00                 25   
1                   4.71                     0.00                 21   
2                   3.91                     0.00                 30   
3                   2.83                     0.00                 29   
4                   5.04                     0.00                 36   
..                   ...                      ...                ...   
935                 6.80                     0.00                 17   
936                 6.24                     0.05                 73   
937                 6.28                     0.00                 18   
938                 5.89                     0.00                 88   
939                 4.25                     0.00                 23   

     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  
..                   ...                   ...               ...       ...  
935                    4                   245              1174      2847  
936                   19                   217              1131      3710  
937                   11                   224              1187      2832  
938                   12                   213              1127      3832  
939                    1                   137               770      1849  

[940 rows x 15 columns]
Sleep Day removed outliers:
             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   
..          ...                    ...                ...                 ...   
405  8792009665  4/30/2016 12:00:00 AM                  1                 343   
406  8792009665   5/1/2016 12:00:00 AM                  1                 503   
407  8792009665   5/2/2016 12:00:00 AM                  1                 415   
408  8792009665   5/3/2016 12:00:00 AM                  1                 516   
409  8792009665   5/4/2016 12:00:00 AM                  1                 439   

     totaltimeinbed  
0               346  
1               407  
2               442  
3               367  
4               712  
..              ...  
405             360  
406             527  
407             423  
408             545  
409             463  

[410 rows x 5 columns]
Heart Rate removed outliers:
                 id                  time  value
0        2022484408  4/12/2016 7:21:00 AM     97
1        2022484408  4/12/2016 7:21:05 AM    102
2        2022484408  4/12/2016 7:21:10 AM    105
3        2022484408  4/12/2016 7:21:20 AM    103
4        2022484408  4/12/2016 7:21:25 AM    101
...             ...                   ...    ...
2483653  8877689391  5/12/2016 2:43:53 PM     57
2483654  8877689391  5/12/2016 2:43:58 PM     56
2483655  8877689391  5/12/2016 2:44:03 PM     55
2483656  8877689391  5/12/2016 2:44:18 PM     55
2483657  8877689391  5/12/2016 2:44:28 PM     56

[2483658 rows x 3 columns]
```

Data summary

In [ ]:
```python
# Data summary
for name, df in df_dict.items():
    print(f"Data summary for {name}:")
    print(f"{name} unique values: {df.id.nunique()}")
    print(df.info())
    print(df.describe())
    print('\n ********************************** \n')
```

```
Data summary for Daily Activity:
Daily Activity unique values: 33
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   id                        940 non-null    int64  
 1   activitydate              940 non-null    object 
 2   totalsteps                940 non-null    int64  
 3   totaldistance             940 non-null    float64
 4   trackerdistance           940 non-null    float64
 5   loggedactivitiesdistance  940 non-null    float64
 6   veryactivedistance        940 non-null    float64
 7   moderatelyactivedistance  940 non-null    float64
 8   lightactivedistance       940 non-null    float64
 9   sedentaryactivedistance   940 non-null    float64
 10  veryactiveminutes         940 non-null    int64  
 11  fairlyactiveminutes       940 non-null    int64  
 12  lightlyactiveminutes      940 non-null    int64  
 13  sedentaryminutes          940 non-null    int64  
 14  calories                  940 non-null    int64  
dtypes: float64(7), int64(7), object(1)
memory usage: 110.3+ KB
None
                 id    totalsteps  totaldistance  trackerdistance  \
count  9.400000e+02    940.000000     940.000000       940.000000   
mean   4.855407e+09   7637.910638       5.489702         5.475351   
std    2.424805e+09   5087.150742       3.924606         3.907276   
min    1.503960e+09      0.000000       0.000000         0.000000   
25%    2.320127e+09   3789.750000       2.620000         2.620000   
50%    4.445115e+09   7405.500000       5.245000         5.245000   
75%    6.962181e+09  10727.000000       7.712500         7.710000   
max    8.877689e+09  36019.000000      28.030001        28.030001   

       loggedactivitiesdistance  veryactivedistance  moderatelyactivedistance  \
count                940.000000          940.000000                940.000000   
mean                   0.108171            1.502681                  0.567543   
std                    0.619897            2.658941                  0.883580   
min                    0.000000            0.000000                  0.000000   
25%                    0.000000            0.000000                  0.000000   
50%                    0.000000            0.210000                  0.240000   
75%                    0.000000            2.052500                  0.800000   
max                    4.942142           21.920000                  6.480000   

       lightactivedistance  sedentaryactivedistance  veryactiveminutes  \
count           940.000000               940.000000         940.000000   
mean              3.340819                 0.001606          21.164894   
std               2.040655                 0.007346          32.844803   
min               0.000000                 0.000000           0.000000   
25%               1.945000                 0.000000           0.000000   
50%               3.365000                 0.000000           4.000000   
75%               4.782500                 0.000000          32.000000   
max              10.710000                 0.110000         210.000000   

       fairlyactiveminutes  lightlyactiveminutes  sedentaryminutes  \
count           940.000000            940.000000        940.000000   
mean             13.564894            192.812766        991.210638   
std              19.987404            109.174700        301.267437   
min               0.000000              0.000000          0.000000   
25%               0.000000            127.000000        729.750000   
50%               6.000000            199.000000       1057.500000   
75%              19.000000            264.000000       1229.500000   
max             143.000000            518.000000       1440.000000   

          calories  
count   940.000000  
mean   2303.609574  
std     718.166862  
min       0.000000  
25%    1828.500000  
50%    2134.000000  
75%    2793.250000  
max    4900.000000  

 ***********************************

Data summary for Sleep Day:
Sleep Day unique values: 24
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 410 entries, 0 to 409
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   id                  410 non-null    int64 
 1   sleepday            410 non-null    object
 2   totalsleeprecords   410 non-null    int64 
 3   totalminutesasleep  410 non-null    int64 
 4   totaltimeinbed      410 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 16.1+ KB
None
                 id  totalsleeprecords  totalminutesasleep  totaltimeinbed
count  4.100000e+02         410.000000          410.000000      410.000000
mean   4.994963e+09           1.119512          419.173171      458.482927
std    2.060863e+09           0.346636          118.635918      127.455140
min    1.503960e+09           1.000000           58.000000       61.000000
25%    3.977334e+09           1.000000          361.000000      403.750000
50%    4.702922e+09           1.000000          432.500000      463.000000
75%    6.962181e+09           1.000000          490.000000      526.000000
max    8.792010e+09           3.000000          796.000000      961.000000

 ***********************************

Data summary for Heart Rate:
Heart Rate unique values: 14
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2483658 entries, 0 to 2483657
Data columns (total 3 columns):
 #   Column  Dtype 
---  ------  ----- 
 0   id      int64 
 1   time    object
 2   value   int64 
dtypes: int64(2), object(1)
memory usage: 56.8+ MB
None
                 id         value
count  2.483658e+06  2.483658e+06
mean   5.513765e+09  7.732842e+01
std    1.950224e+09  1.940450e+01
min    2.022484e+09  3.600000e+01
25%    4.388162e+09  6.300000e+01
50%    5.553957e+09  7.300000e+01
75%    6.962181e+09  8.800000e+01
max    8.877689e+09  2.030000e+02
```

****

Feature engineering

Daily Activity

In [ ]:
```python
# Create weekday and total active minutes column for further analysis
daily_activity_df.rename(columns={'activitydate':'date'}, inplace=True)
daily_activity_df.date = pd.to_datetime(daily_activity_df.date)
daily_activity_df['weekday'] = daily_activity_df.date.dt.day_name()
daily_activity_df['total_active_minutes']=daily_activity_df.fairlyactiveminutes + daily_activity_df.lightlyactiveminutes + daily_activity_df.veryactiveminutes

# daily_activity_df.to_csv('daily_activity_update.csv', index=False) # enable if needed
daily_activity_df.head()
```

Out[ ]:

	id	date	totalsteps	totaldistance	trackerdistance	loggedactivitiesdistance	veryactivedistance	moderatelyactivedistance	lightactivedistance	sedentaryactivedistance	veryactiveminute
0	1503960366	2016-04-12	13162	8.50	8.50	0.0	1.88	0.55	6.06	0.0	2
1	1503960366	2016-04-13	10735	6.97	6.97	0.0	1.57	0.69	4.71	0.0	2
2	1503960366	2016-04-14	10460	6.74	6.74	0.0	2.44	0.40	3.91	0.0	3
3	1503960366	2016-04-15	9762	6.28	6.28	0.0	2.14	1.26	2.83	0.0	2
4	1503960366	2016-04-16	12669	8.16	8.16	0.0	2.71	0.41	5.04	0.0	3

Heart Rate

In [ ]:
```python
# Heart rate
heart_rate_df.time = pd.to_datetime(heart_rate_df.time)
heart_rate_df['weekday'] = heart_rate_df.time.dt.day_name()
heart_rate_df['date'] = heart_rate_df.time.dt.date
heart_rate_df['hour'] = heart_rate_df.time.dt.time
```

In [ ]:
```python
# Create weekday, date, time, session column for further analysis
def session(time):
    hour = time.hour
    if 5 <= hour < 12:
        return 'morning'
    if 12 <= hour < 17:
        return 'afternoon'
    if 17 <= hour < 21:
        return 'evening'
    else:
        return 'night'

heart_rate_df['session'] = heart_rate_df.time.apply(session)
heart_rate_df.drop(columns='time', inplace=True)
heart_rate_df = heart_rate_df.rename(columns={'hour':'time'})

# heart_rate_df.to_csv('hear_rate_updated.csv', index=False) # enable if needed
heart_rate_df.head()
```

Out[ ]:

	id	value	weekday	date	time	session
0	2022484408	97	Tuesday	2016-04-12	07:21:00	morning
1	2022484408	102	Tuesday	2016-04-12	07:21:05	morning
2	2022484408	105	Tuesday	2016-04-12	07:21:10	morning
3	2022484408	103	Tuesday	2016-04-12	07:21:20	morning
4	2022484408	101	Tuesday	2016-04-12	07:21:25	morning

In [ ]:
```python
# Create heart rate mean dataframe
heart_rate_mean = heart_rate_df.groupby(['id', 'weekday','date'])['value'].mean()
heart_rate_mean = heart_rate_mean.reset_index()
heart_rate_mean = heart_rate_mean.rename(columns={'value': 'avgheartrate'})
heart_rate_mean.head()
```

Out[ ]:

	id	weekday	date	avgheartrate
0	2022484408	Friday	2016-04-15	80.437382
1	2022484408	Friday	2016-04-22	80.125444
2	2022484408	Friday	2016-04-29	83.412873
3	2022484408	Friday	2016-05-06	81.722098
4	2022484408	Monday	2016-04-18	82.712829

Sleep

In [ ]:
```python
# Create weekday, date, time column for further analysis
sleep_day_df['sleepday'] = pd.to_datetime(sleep_day_df.sleepday)
sleep_day_df['weekday'] = sleep_day_df.sleepday.dt.day_name()
sleep_day_df['date'] = sleep_day_df.sleepday.dt.date
sleep_day_df['time'] = sleep_day_df.sleepday.dt.time
sleep_day_df = sleep_day_df.drop('sleepday', axis=1)

# sleep_day_df.to_csv('sleep_day_updated.csv', index=False) # enable if needed
sleep_day_df.head()
```

Out[ ]:

	id	totalsleeprecords	totalminutesasleep	totaltimeinbed	weekday	date	time
0	1503960366	1	327	346	Tuesday	2016-04-12	00:00:00
1	1503960366	2	384	407	Wednesday	2016-04-13	00:00:00
2	1503960366	1	412	442	Friday	2016-04-15	00:00:00
3	1503960366	2	340	367	Saturday	2016-04-16	00:00:00
4	1503960366	1	700	712	Sunday	2016-04-17	00:00:00

In [ ]:
```python
# Create sleep mean dataframe
sleep_mean = sleep_day_df.groupby(['id', 'weekday','date'])[['totalminutesasleep','totaltimeinbed']].mean()
```

```
sleep_mean = sleep_mean.reset_index()
sleep_mean.head()
```

Out[ ]:

	id	weekday	date	totalminutesasleep	totaltimeinbed
0	1503960366	Friday	2016-04-15	412.0	442.0
1	1503960366	Friday	2016-04-29	341.0	354.0
2	1503960366	Friday	2016-05-06	334.0	367.0
3	1503960366	Monday	2016-04-25	277.0	323.0
4	1503960366	Monday	2016-05-02	277.0	309.0

Merge data

In [ ]:

```
# Concat data on the same id, weekday and date
df_all = pd.concat([daily_activity_df.set_index(['id', 'weekday', 'date']),
                       heart_rate_mean.set_index(['id', 'weekday', 'date']),
                       sleep_mean.set_index(['id', 'weekday', 'date'])],
                      axis=1, join='outer')
df_all.head()
```

Out[ ]:

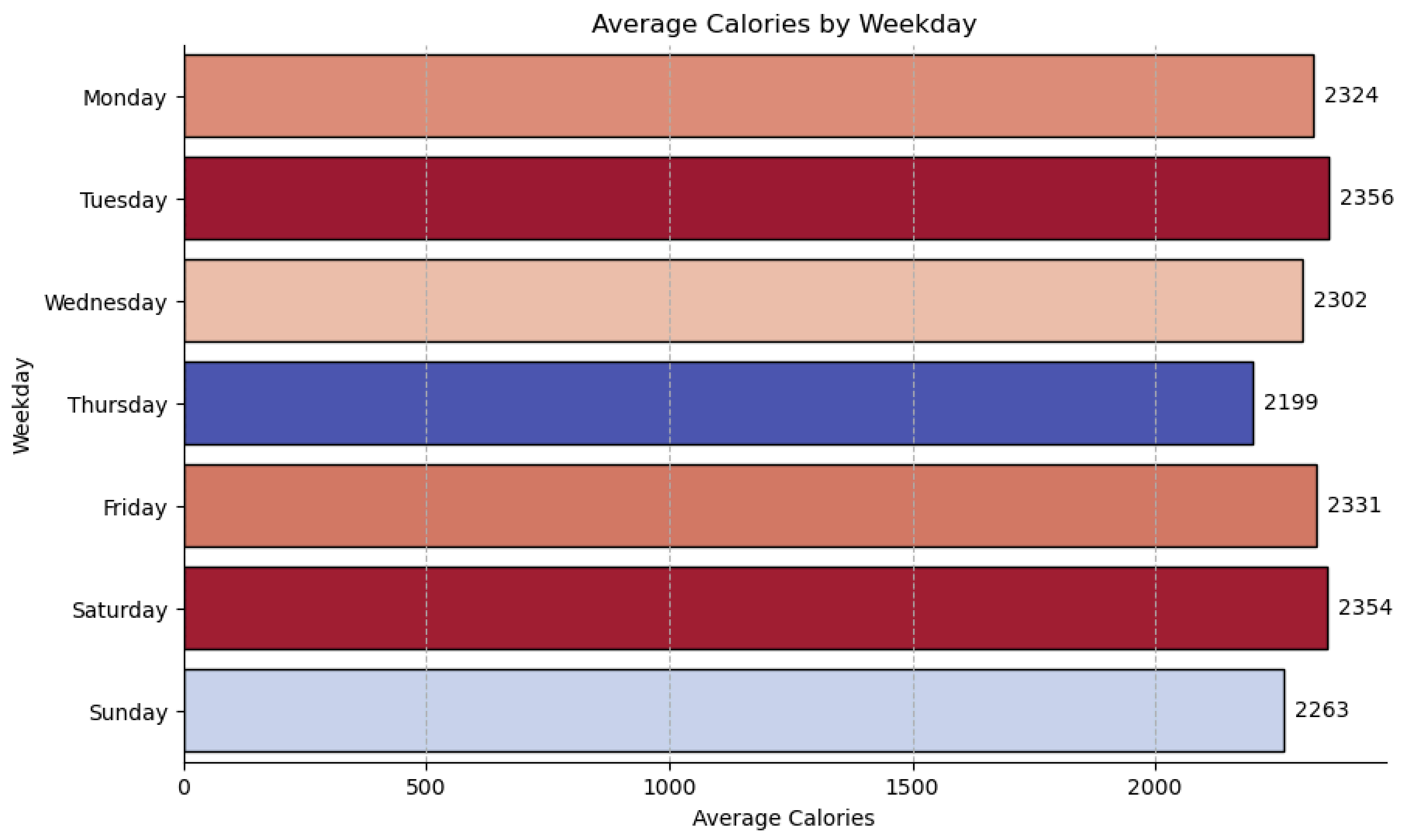
id	weekday	date	totalsteps	totaldistance	trackerdistance	loggedactivitiesdistance	veryactivedistance	moderatelyactivedistance	lightactivedistance	sedentaryactivedistance	verya
1503960366	Tuesday	2016-04-12	13162	8.50	8.50	0.0	1.88	0.55	6.06	0.0	
	Wednesday	2016-04-13	10735	6.97	6.97	0.0	1.57	0.69	4.71	0.0	
	Thursday	2016-04-14	10460	6.74	6.74	0.0	2.44	0.40	3.91	0.0	
	Friday	2016-04-15	9762	6.28	6.28	0.0	2.14	1.26	2.83	0.0	
	Saturday	2016-04-16	12669	8.16	8.16	0.0	2.71	0.41	5.04	0.0	

Phase 4: Analyze

Activity

In [ ]:

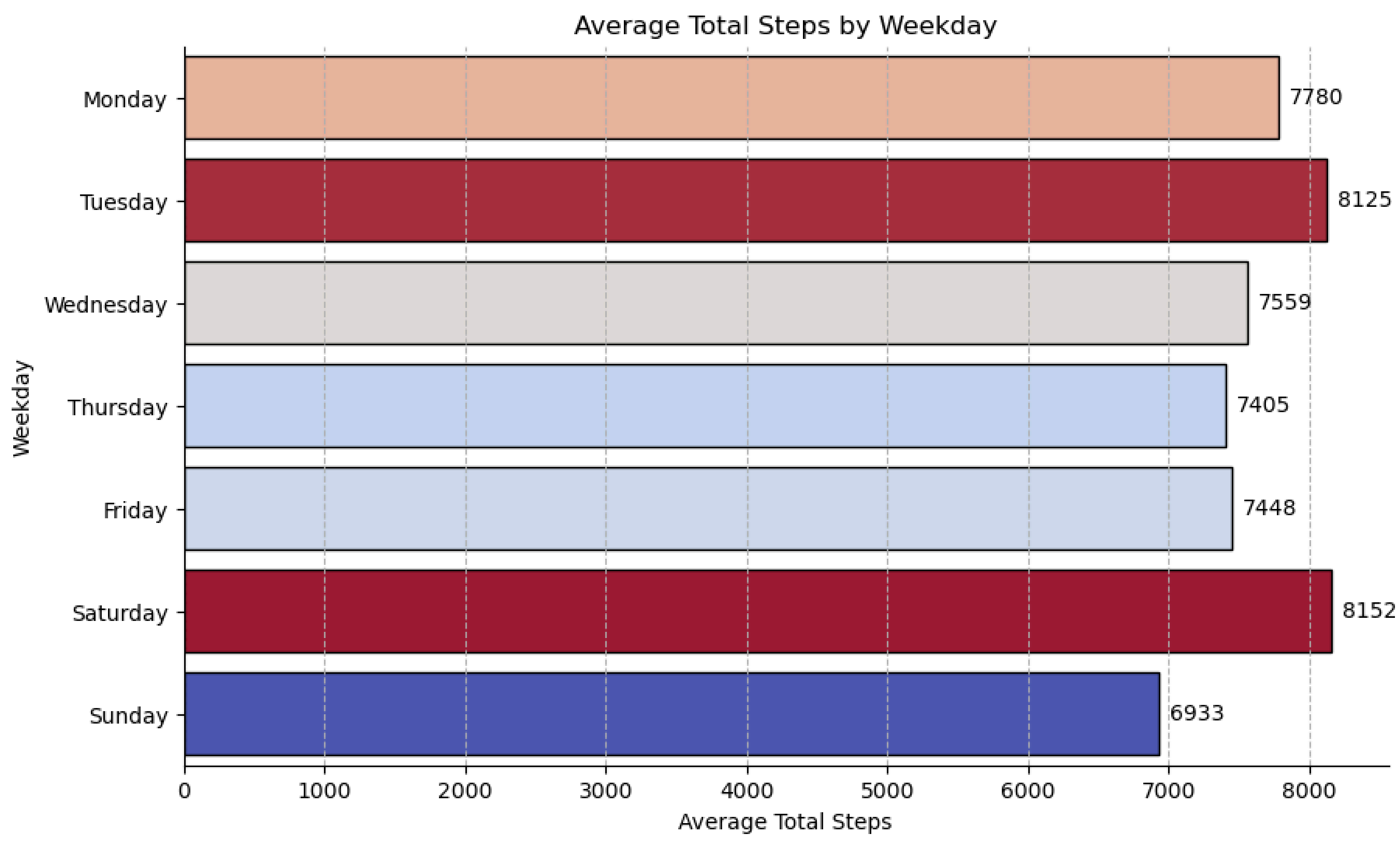
```
# Calculate the average calories burned each weekday
avg_calories = daily_activity_df.groupby('weekday')['calories'].mean()

# Create weekday oder for the bar plot
weekday_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
# Plot bar graph, title, labels
plt.figure(figsize=(10, 6), dpi=100)
ax = sns.barplot(y='weekday', x='calories', data=pd.DataFrame(avg_calories).reset_index(), order=weekday_order, hue='calories', edgecolor="black", palette="coolwarm", legend=
plt.ylabel('Weekday')
plt.xlabel('Average Calories')
plt.title('Average Calories by Weekday')
# Annotate bar value
for p in ax.patches:
    ax.annotate(str(int(p.get_width())), (p.get_width(), p.get_y() + p.get_height() / 2.),
                ha='left', va='center', color='black', xytext=(5, 0), textcoords='offset points')
# Add grid lines and remove spines
ax.grid(axis='x', linestyle='--', alpha=0.9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.yaxis.grid(False)
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')
plt.savefig('avg_cal_burned_by_weekday.png')
plt.show()
```



In [ ]:

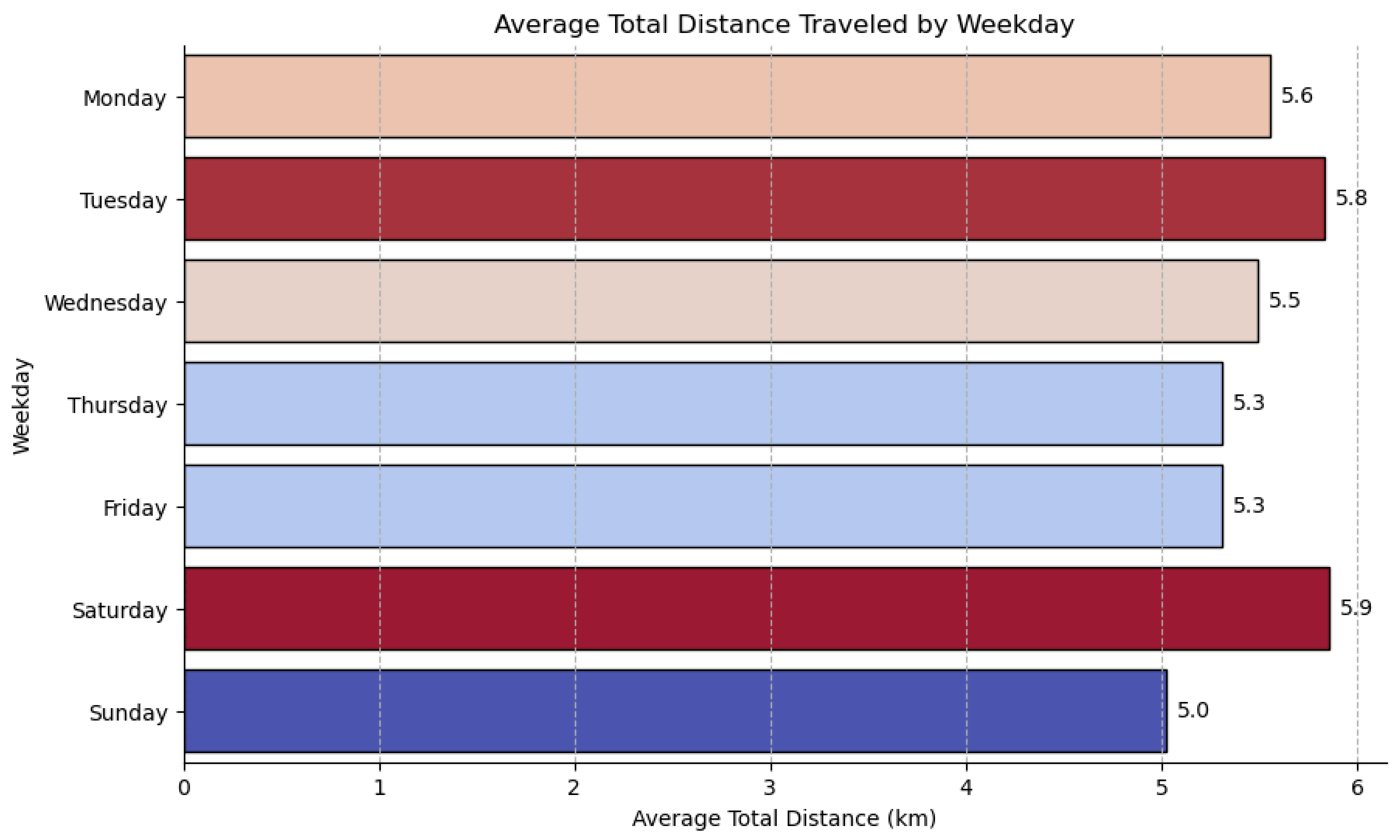
```
# Calculate the average steps each weekday
avg_step_by_weekday = daily_activity_df.groupby('weekday')['totalsteps'].mean()

# Plot bar graph, title, labels
plt.figure(figsize=(10, 6), dpi=100)
ax = sns.barplot(y='weekday', x='totalsteps', data=pd.DataFrame(avg_step_by_weekday).reset_index(), order=weekday_order, hue='totalsteps', edgecolor="black", palette="coolwarn
plt.ylabel('Weekday')
plt.xlabel('Average Total Steps')
plt.title('Average Total Steps by Weekday')
# Annotate bar value
for p in ax.patches:
    ax.annotate(str(int(p.get_width())), (p.get_width(), p.get_y() + p.get_height() / 2.),
                ha='left', va='center', color='black', xytext=(5, 0), textcoords='offset points')
# Add grid lines and remove spines
```

```python
ax.grid(axis='x', linestyle='--', alpha=0.9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.yaxis.grid(False)
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')
plt.savefig('avg_total_steps_by_weekday.png')
plt.show()
```



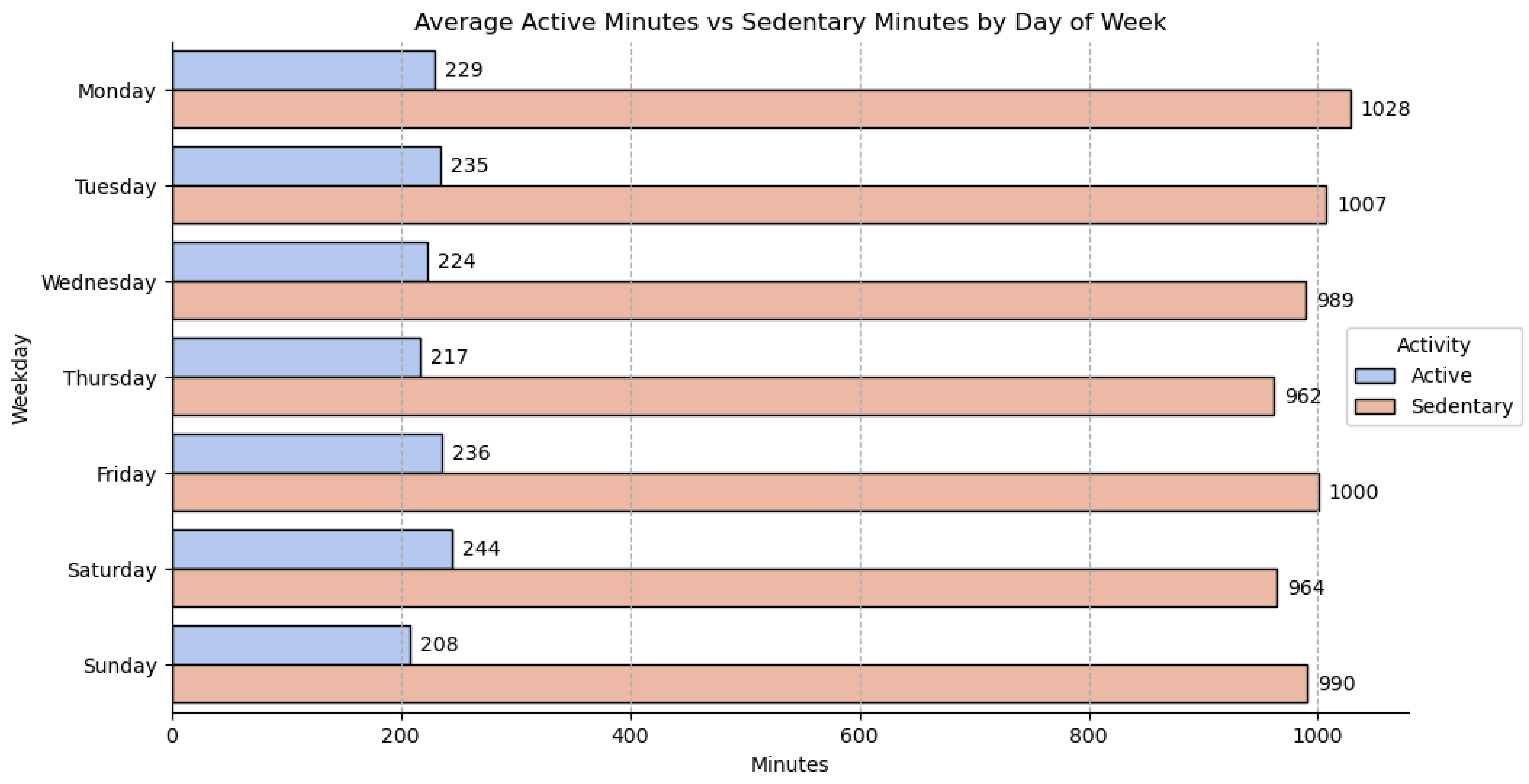
In [ ]:
```python
# Scatter plot calories and step taken to see the correlation
plt.figure(figsize=(10, 6), dpi=100)
ax = sns.scatterplot(x='totalsteps', y='calories', data=daily_activity_df, hue='calories', palette='coolwarm', edgecolor="black", legend=False)
plt.xlabel("Steps Taken")
plt.ylabel("Calories Burned")
plt.title("Calories Burned for Steps Taken")

# Add grid lines and remove spines
plt.grid(axis='both', linestyle='--', alpha=0.9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')
plt.savefig('cal_burned_for_steps.png')
plt.show()
```

In [ ]:
```python
# Calculate the average total distance travel each weekday
avg_total_dist_by_day = daily_activity_df.groupby('weekday')['totaldistance'].mean()

# Plot bar graph, title, labels
plt.figure(figsize=(10, 6), dpi=100)
ax = sns.barplot(y='weekday', x='totaldistance', data=pd.DataFrame(avg_total_dist_by_day).reset_index(), order=weekday_order, hue='totaldistance', edgecolor="black", palette='
plt.ylabel('Weekday')
plt.xlabel('Average Total Distance (km)')
plt.title('Average Total Distance Traveled by Weekday')
# Annotate bar value
for p in ax.patches:
    ax.annotate('{:.1f}'.format(p.get_width()), (p.get_width(), p.get_y() + p.get_height() / 2.), ha='left', va='center', color='black', xytext=(5, 0), textcoords='offset poi
# Add grid lines and remove spines
ax.grid(axis='x', linestyle='--', alpha=0.9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.yaxis.grid(False)
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')
plt.savefig('avg_total_distance_by_weekday.png')
plt.show()
```



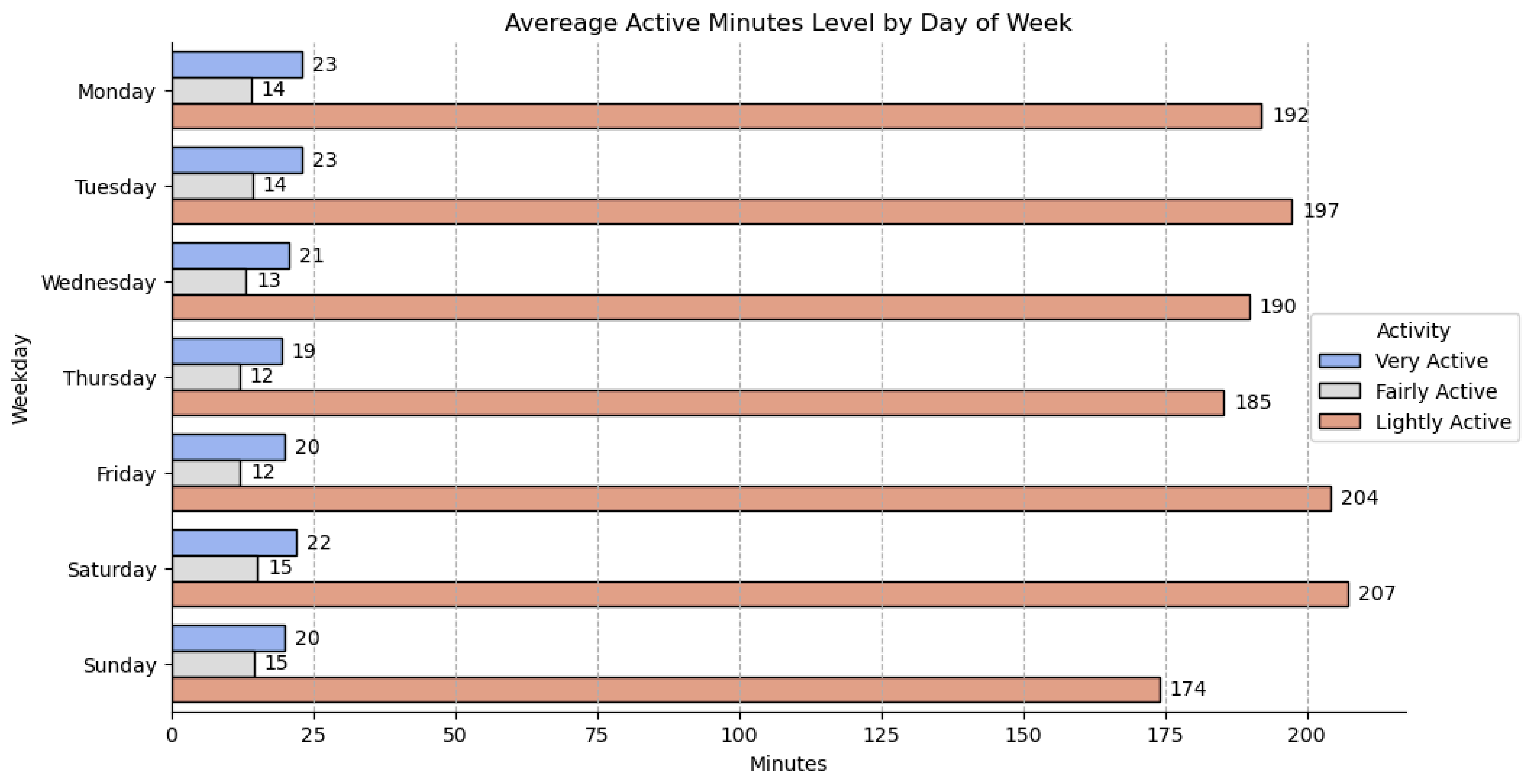
In [ ]:

```python
# average activity by weekday
avg_activity_by_day = daily_activity_df.groupby('weekday')[['total_active_minutes', 'sedentaryminutes']].mean()
# Plot bar graph, title, labels
plt.figure(figsize=(11, 6), dpi=100)
ax = sns.barplot(y='weekday', x='value', hue='variable', data=avg_activity_by_day.reset_index().melt(id_vars='weekday'), order=weekday_order, palette="coolwarm", edgecolor="b
plt.ylabel('Weekday')
plt.xlabel('Minutes')
plt.title('Average Active Minutes vs Sedentary Minutes by Day of Week')

# Annotate bar values
for p in ax.patches:
    if p.get_width() > 0:
        ax.annotate('{:.0f}'.format(p.get_width()), (p.get_width(), p.get_y() + p.get_height() / 2.),
                    ha='left', va='center', color='black', xytext=(5, 0), textcoords='offset points')

# Add grid lines and remove spines
ax.grid(axis='x', linestyle='--', alpha=0.9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.yaxis.grid(False)
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')

# Show legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, ['Active', 'Sedentary'], title='Activity', loc='center right', bbox_to_anchor=(1.1, 0.5))
plt.savefig('avg_active_sedentary_by_weekday.png')
plt.show()
```

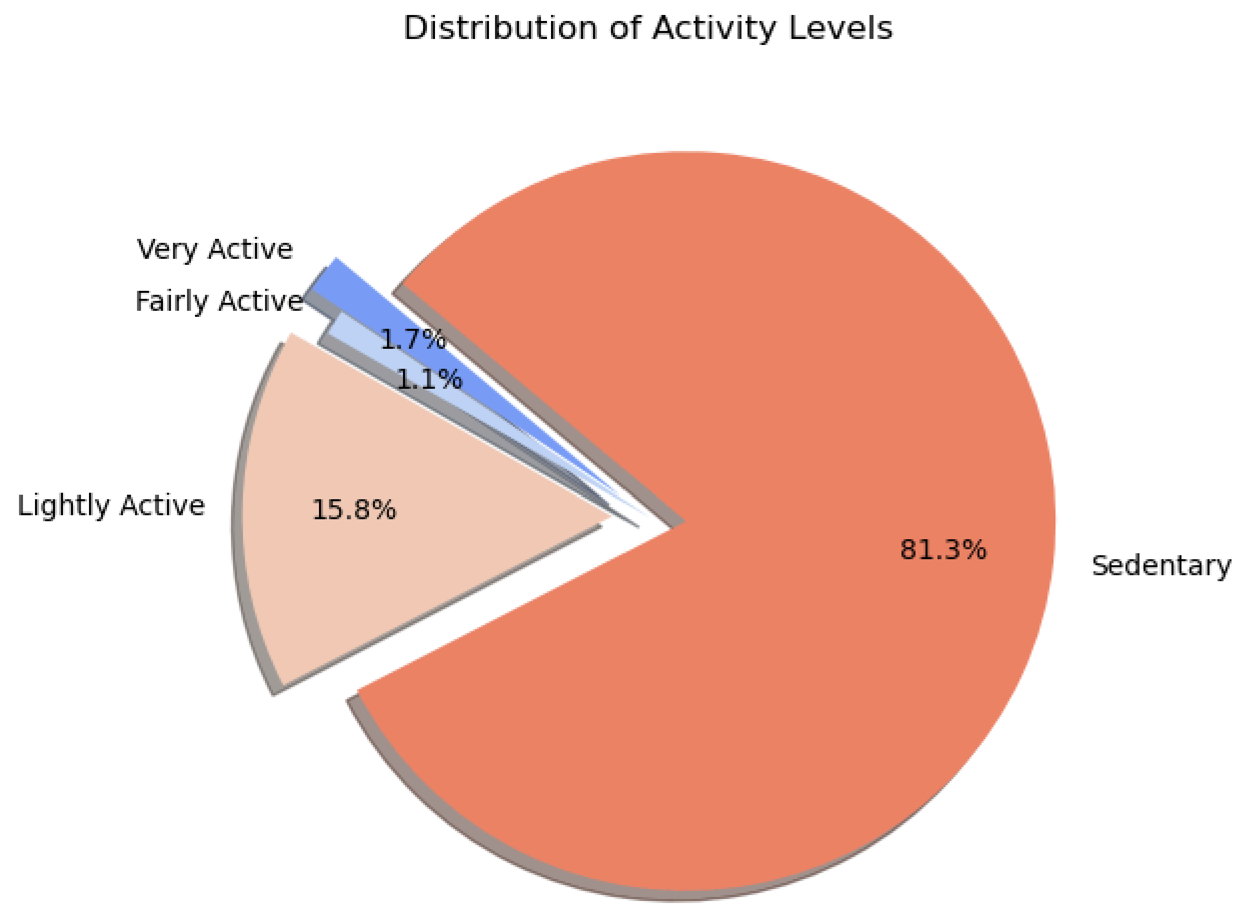


In [ ]:

```python
# average activity minutes by level and weekday
avg_active_minutes_by_day = daily_activity_df.groupby('weekday')[['veryactiveminutes', 'fairlyactiveminutes', 'lightlyactiveminutes']].mean()

# Plot bar graph, title, labels
plt.figure(figsize=(11, 6), dpi=100)
ax = sns.barplot(y='weekday', x='value', hue='variable', data=avg_active_minutes_by_day.reset_index().melt(id_vars='weekday'), order=weekday_order, palette="coolwarm", edgeco
plt.ylabel('Weekday')
plt.xlabel('Minutes')
plt.title('Avereage Active Minutes Level by Day of Week')

# Annotate bar values
for p in ax.patches:
    if p.get_width() > 0:
        ax.annotate('{:.0f}'.format(p.get_width()), (p.get_width(), p.get_y() + p.get_height() / 2.),
                    ha='left', va='center', color='black', xytext=(5, 0), textcoords='offset points')

# Add grid lines and remove spines
ax.grid(axis='x', linestyle='--', alpha=0.9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.yaxis.grid(False)
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')

# Show legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, ['Very Active', 'Fairly Active', 'Lightly Active'], title='Activity', loc='center right', bbox_to_anchor=(1.1, 0.5))
plt.savefig('avg_active_level_by_weekday.png')
plt.show()
```

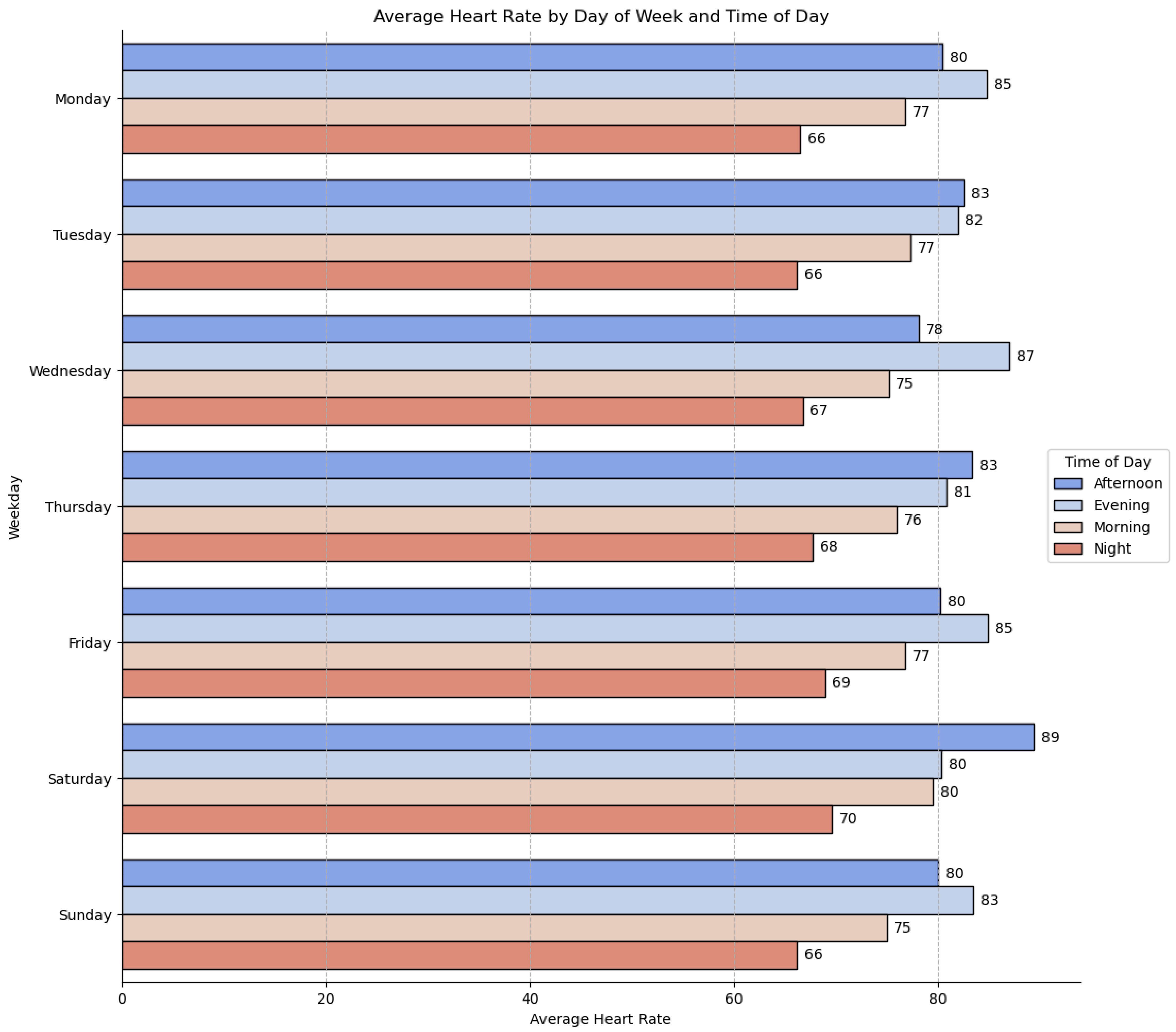


In [ ]:
```python
# Total activity minutes by level
very_active_mins = daily_activity_df['veryactiveminutes'].sum()
fairly_active_mins = daily_activity_df['fairlyactiveminutes'].sum()
lightly_active_mins = daily_activity_df['lightlyactiveminutes'].sum()
sedentary_mins = daily_activity_df['sedentaryminutes'].sum()

# Plot and annotate value
plt.figure(figsize=(6, 6), dpi=100)
labels = ['Very Active', 'Fairly Active', 'Lightly Active', 'Sedentary']
sizes = [very_active_mins, fairly_active_mins, lightly_active_mins, sedentary_mins]
explode = [0.1, 0, 0.1, 0.1]
colors = plt.cm.coolwarm([0.2, 0.4, 0.6, 0.8])
plt.pie(sizes, labels=labels, autopct="%1.1f%%", pctdistance=0.7, explode=explode, shadow=True, startangle=140, colors=colors)

# Show legend
plt.title('Distribution of Activity Levels')
plt.savefig('distribution_active_level.png')
plt.show()
```



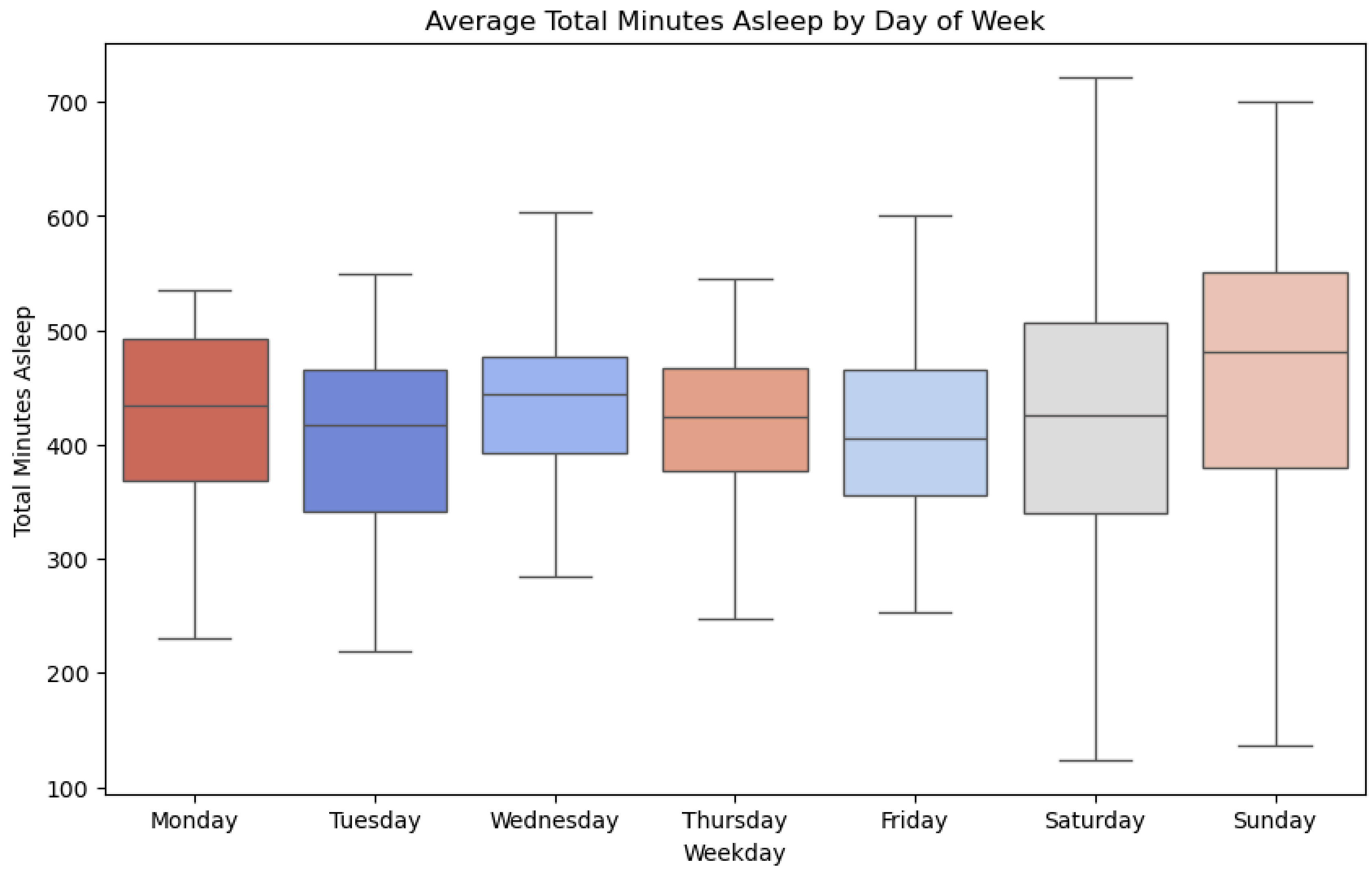
Heart Rate

In [ ]:
```python
# Average heart rate group by day of week
avg_heartrate = heart_rate_df.groupby(["weekday", "session"])["value"].mean()
avg_heartrate = avg_heartrate.reset_index()
avg_heartrate_melted = avg_heartrate.melt(id_vars=['weekday', 'session'], var_name='time_of_day', value_name='average_heartrate')

# Plot bar graph, title, labels
plt.figure(figsize=(12, 12), dpi=100)
ax = sns.barplot(y='weekday', x='average_heartrate', hue='session', data=avg_heartrate_melted, order=weekday_order, palette="coolwarm", edgecolor="black")
plt.ylabel('Weekday')
plt.xlabel('Average Heart Rate')
plt.title('Average Heart Rate by Day of Week and Time of Day')

# Annotate bar values
for p in ax.patches:
    if p.get_width() > 0:
        ax.annotate('{:.0f}'.format(p.get_width()), (p.get_width(), p.get_y() + p.get_height() / 2.),
                    ha='left', va='center', color='black', xytext=(5, 0), textcoords='offset points')

# Add grid lines and remove spines
ax.grid(axis='x', linestyle='--', alpha=0.9)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.yaxis.grid(False)
ax.spines['left'].set_color('black')
ax.spines['bottom'].set_color('black')

# Show legend
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, ['Afternoon', 'Evening', 'Morning', 'Night'], title='Time of Day', loc='center right', bbox_to_anchor=(1.1, 0.5))
plt.savefig('avg_heart_rate_by_weekday.png')
plt.show()
```



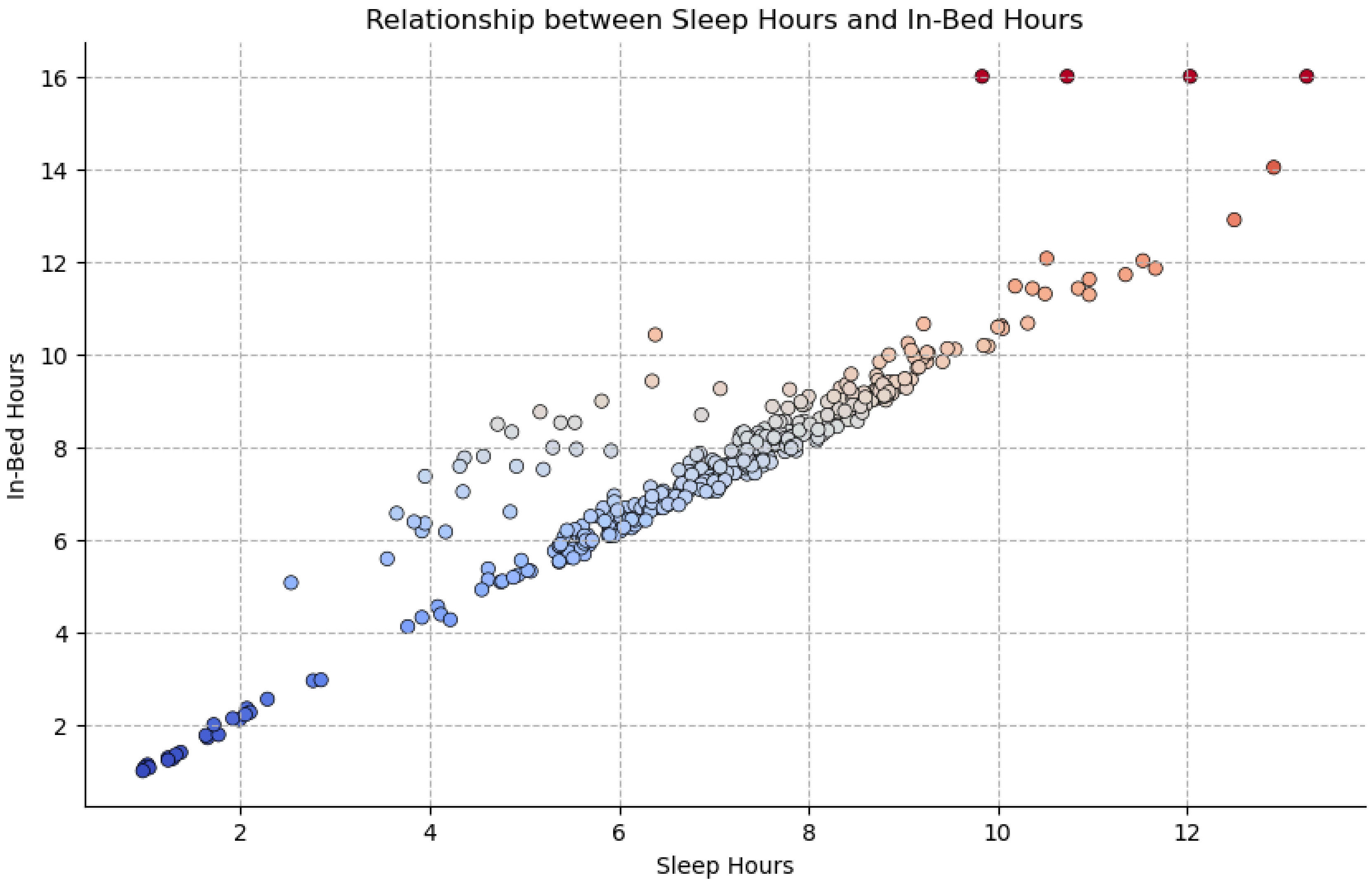
Sleep

In [ ]:
```
# Average sleep minutes group by day of week
sleep_grouped = sleep_day_df.groupby('weekday')['totalminutesasleep'].mean()
# Plot boxplot
fig, ax = plt.subplots(figsize=(10, 6), dpi=100)
sns.boxplot(x='weekday', y='totalminutesasleep', data=sleep_day_df, order=weekday_order, showfliers=False, ax=ax, palette="coolwarm", hue='weekday', legend= False)

# Add title and labels
plt.title('Average Total Minutes Asleep by Day of Week')
plt.xlabel('Weekday')
plt.ylabel('Total Minutes Asleep')

# Show plot
plt.savefig('avg_total_asleep_by_weekday.png')
plt.show()
```



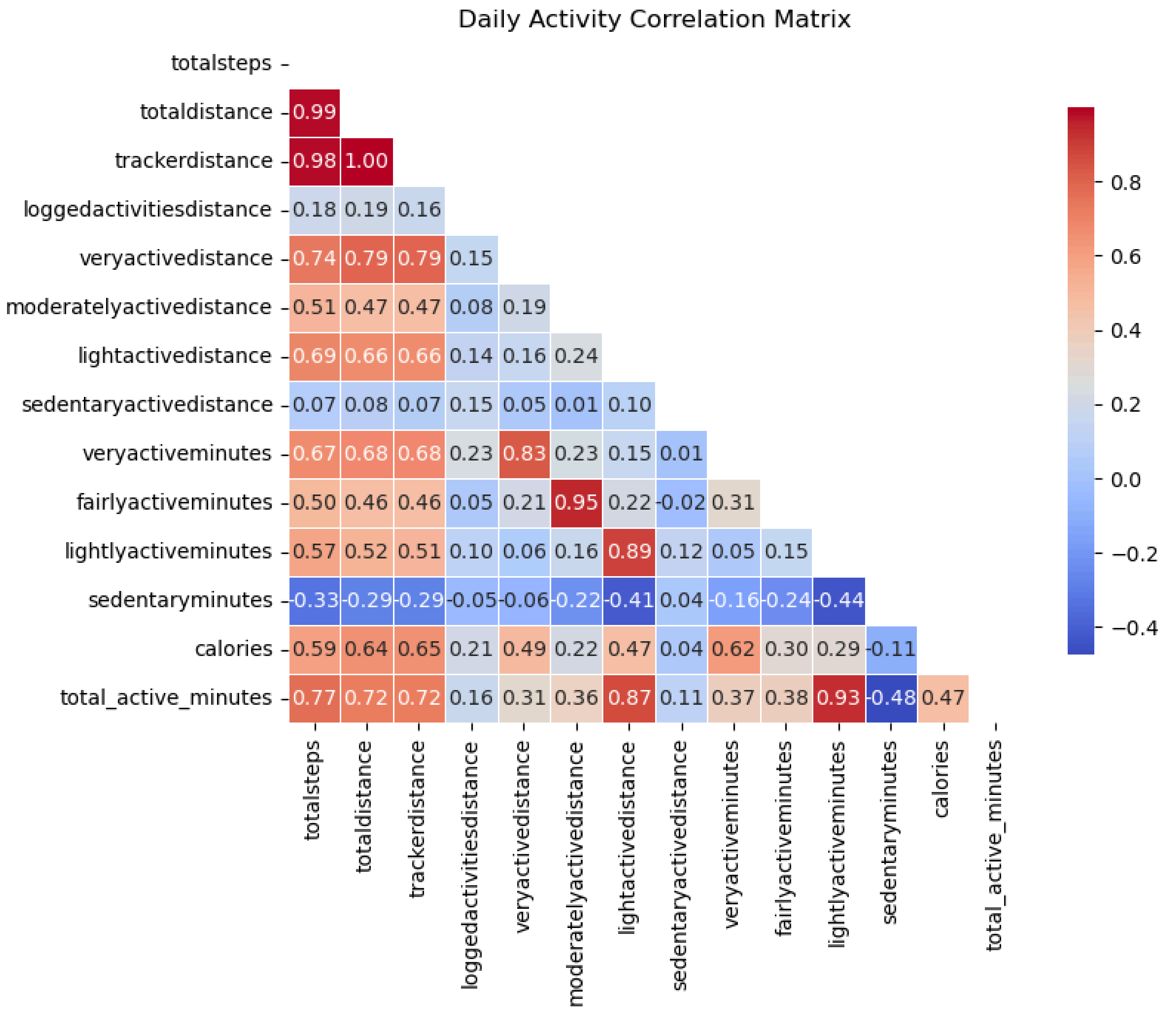
In [ ]:
```
# Create sleep hours and inbed hours column
sleep_day_df['sleephours'] = sleep_day_df.totalminutesasleep /60
sleep_day_df['inbedhours'] =sleep_day_df.totaltimeinbed /60

# Scatter plot
plt.figure(figsize=(10, 6), dpi=100)
ax = sns.scatterplot(x='sleephours', y='inbedhours', data=sleep_day_df, hue='inbedhours', legend=False , palette='coolwarm', edgecolor='black')
plt.xlabel("Sleep Hours")
plt.ylabel("In-Bed Hours")
plt.title("Relationship between Sleep Hours and In-Bed Hours")

# Add grid lines and remove spines
plt.grid(axis='both', linestyle='--', alpha=0.9)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
```

```
plt.gca().spines['left'].set_color('black')
plt.gca().spines['bottom'].set_color('black')
plt.savefig('sleep_and_inbed.png')
plt.show()
```

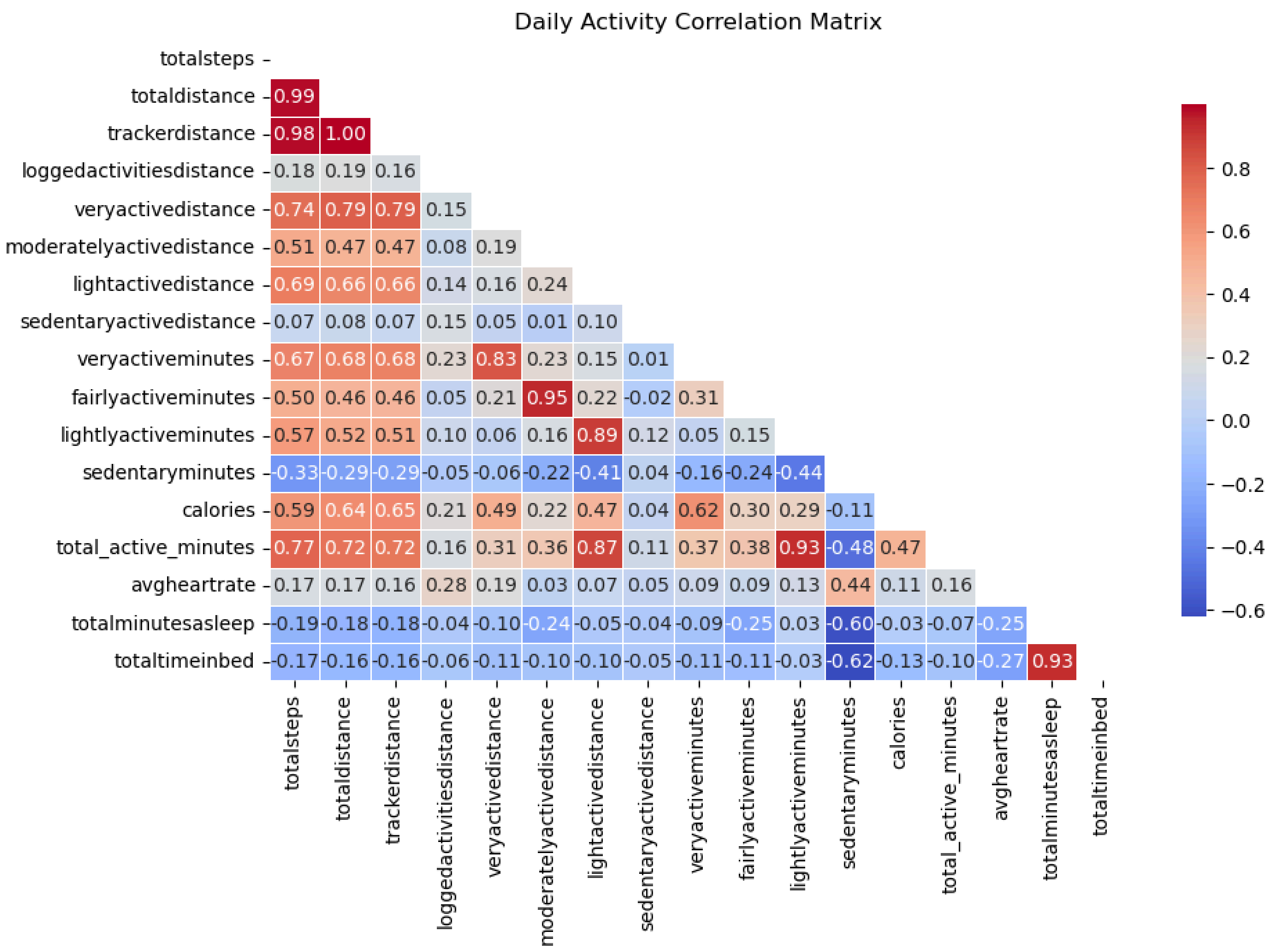


Correlation

In [ ]:
```
# Correlation between activity
corr = daily_activity_df.drop('id',axis=1).corr(numeric_only=True)

# Create a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
plt.figure(figsize=(8, 6), dpi=100)
plt.title("Daily Activity Correlation Matrix")

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap='coolwarm', annot=True, fmt='.2f', linewidths=0.5, cbar_kws={"shrink": .8})

# Display the heatmap
plt.show()
```



In [ ]:
```
# Correlation between activity, heart rate, and sleep
corr = df_all.corr(numeric_only=True)

# Create a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
plt.figure(figsize=(10, 6), dpi=100)
plt.title("Daily Activity Correlation Matrix")

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap='coolwarm', annot=True, fmt='.2f', linewidths=0.5, cbar_kws={"shrink": .8})

# Display the heatmap
plt.show()
```



Phase 5: Share

Activity

Fitbit trackers measure physical activity in terms of "active minutes," which are the minutes during when you have spent at least 10 minutes in an activity that burns three times as many calories as you do at rest.

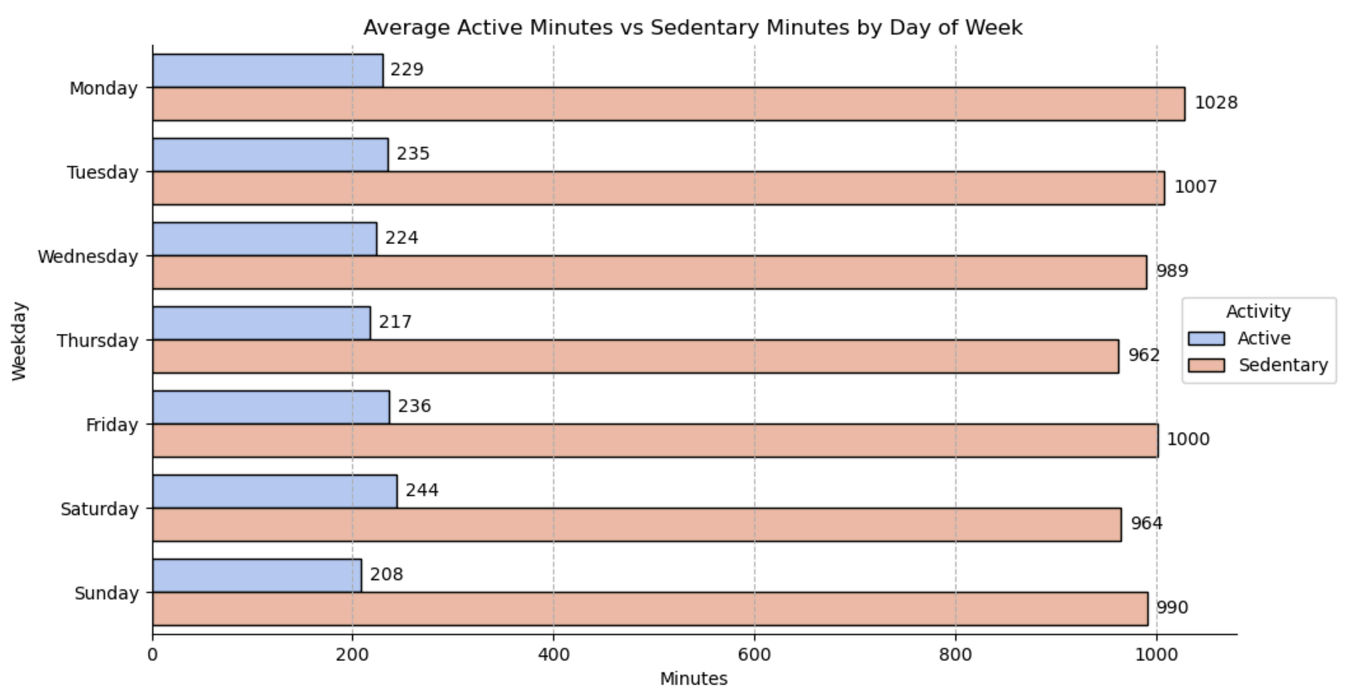
Fitbit defines three levels of physical activity:

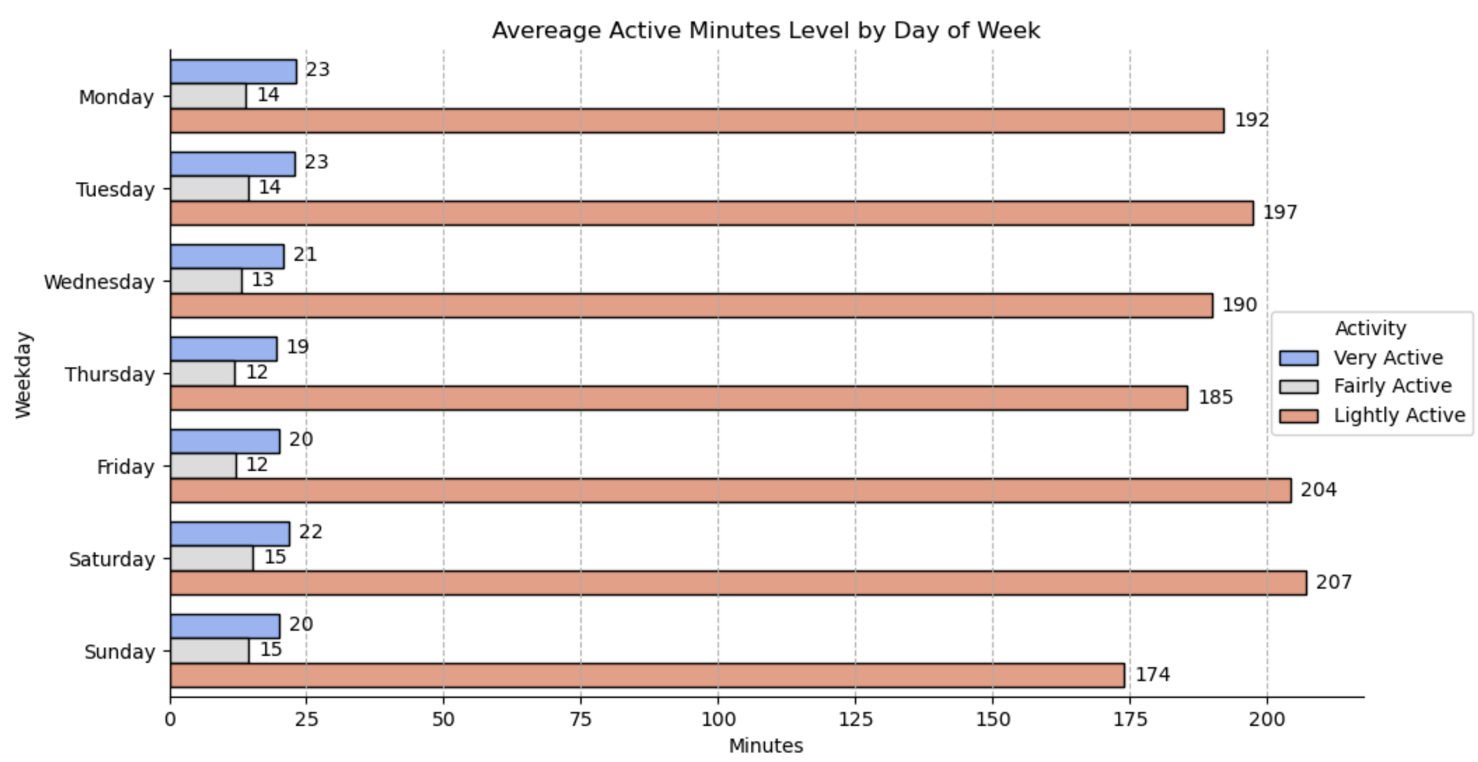
Lightly Active Minutes: These minutes are logged when you engage in light physical activities that are more intense than just resting but not as intense as brisk walking. This can include activities like slow walking, light household chores, and casual movement.

Fairly Active Minutes: - These minutes are recorded when you participate in activities that are moderately intense. Examples include brisk walking, gardening, or playing with kids. These activities are more vigorous than light activities but not as intense as running or high-intensity workouts.

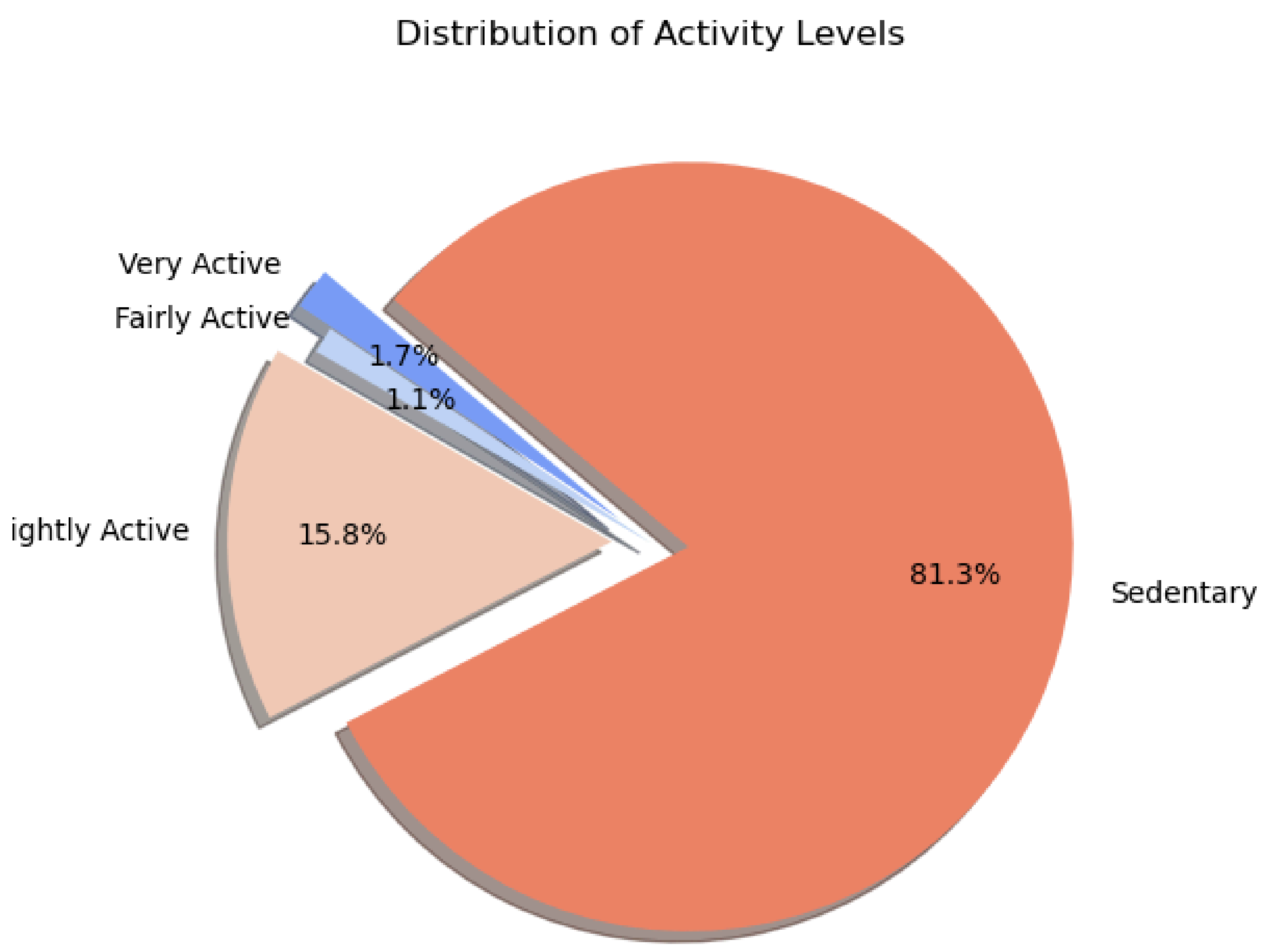
Very Active Minutes: These are the minutes you spend engaging in activities that are vigorous and significantly raise your heart rate. Examples include running, high-intensity interval training (HIIT), and aerobics.

Fitbit uses sensors to track movements and heart rate, then employs proprietary algorithms to analyze this data and classify activities. The more active you are, the more active minutes you will accumulate.





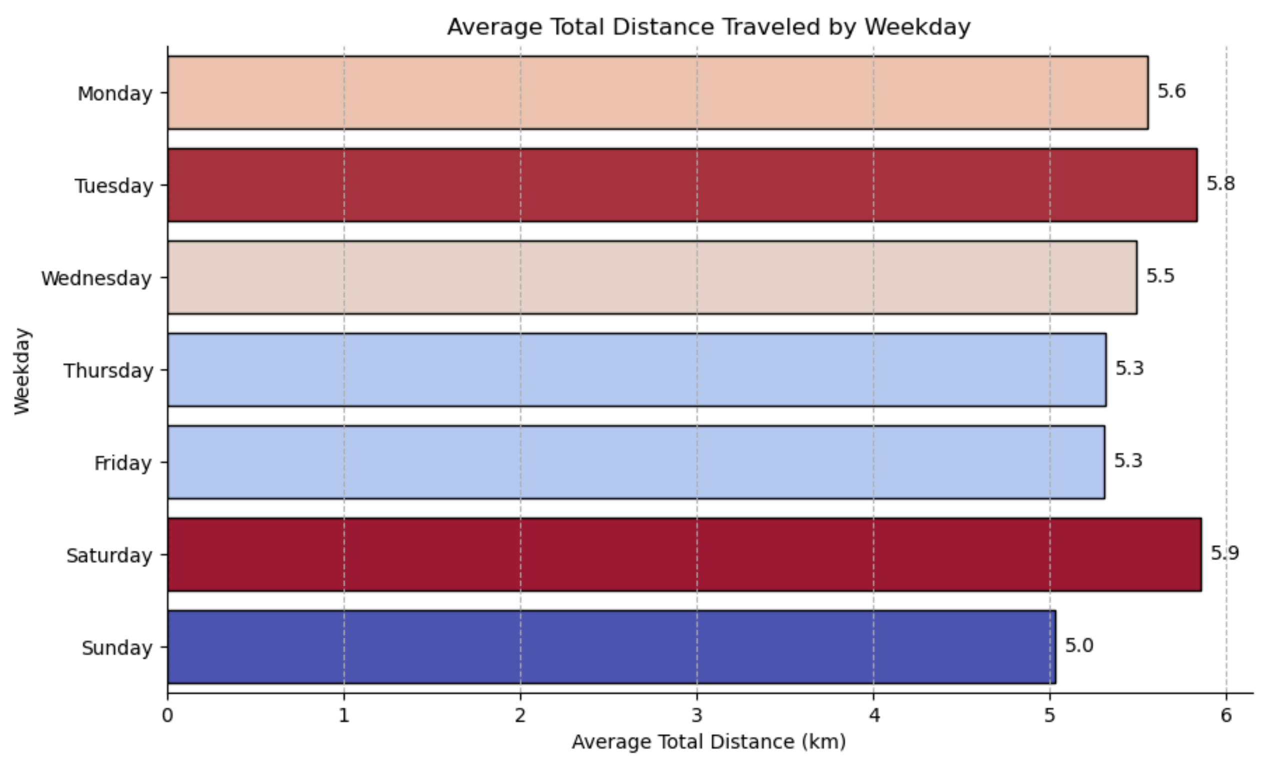
On average, Saturday, Tuesday, and Friday are the most active days, with Monday, Wednesday, Thursday, and Sunday following. People usually get around 20 minutes of very active and 15 minutes of fairly active time each day. However, lightly active minutes vary, with Saturday, Friday, and Tuesday leading, followed by Monday, Wednesday, Thursday, and Sunday.

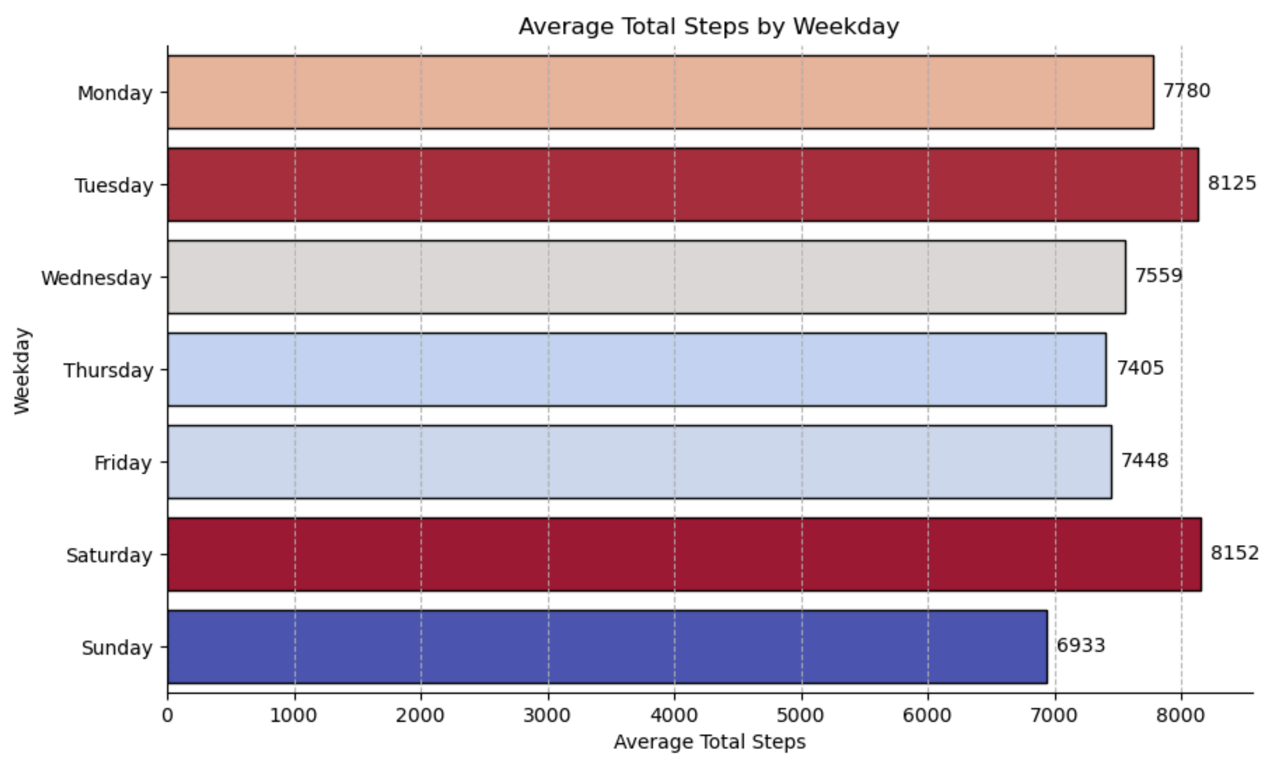


Sedentary minutes account for a significant 81.3% of recorded time, indicating extensive periods of inactivity. Conversely, lightly active minutes, comprising 15.8%. However lightly active minutes significantly differentiate the most active day from the least active one. This suggests that those who engage in more light activities tend to be more active overall.

Analysis of total distance traveled and steps taken by day shows Saturday, Tuesday, and Friday as the most active days. Outdoor activities are more common on Saturday and Tuesday, contrasting with indoor activities dominating on Friday. This trend is reflected in their respective averages for distance and steps, with Saturday and Tuesday consistently ranking highest, while Friday consistently ranks lower.

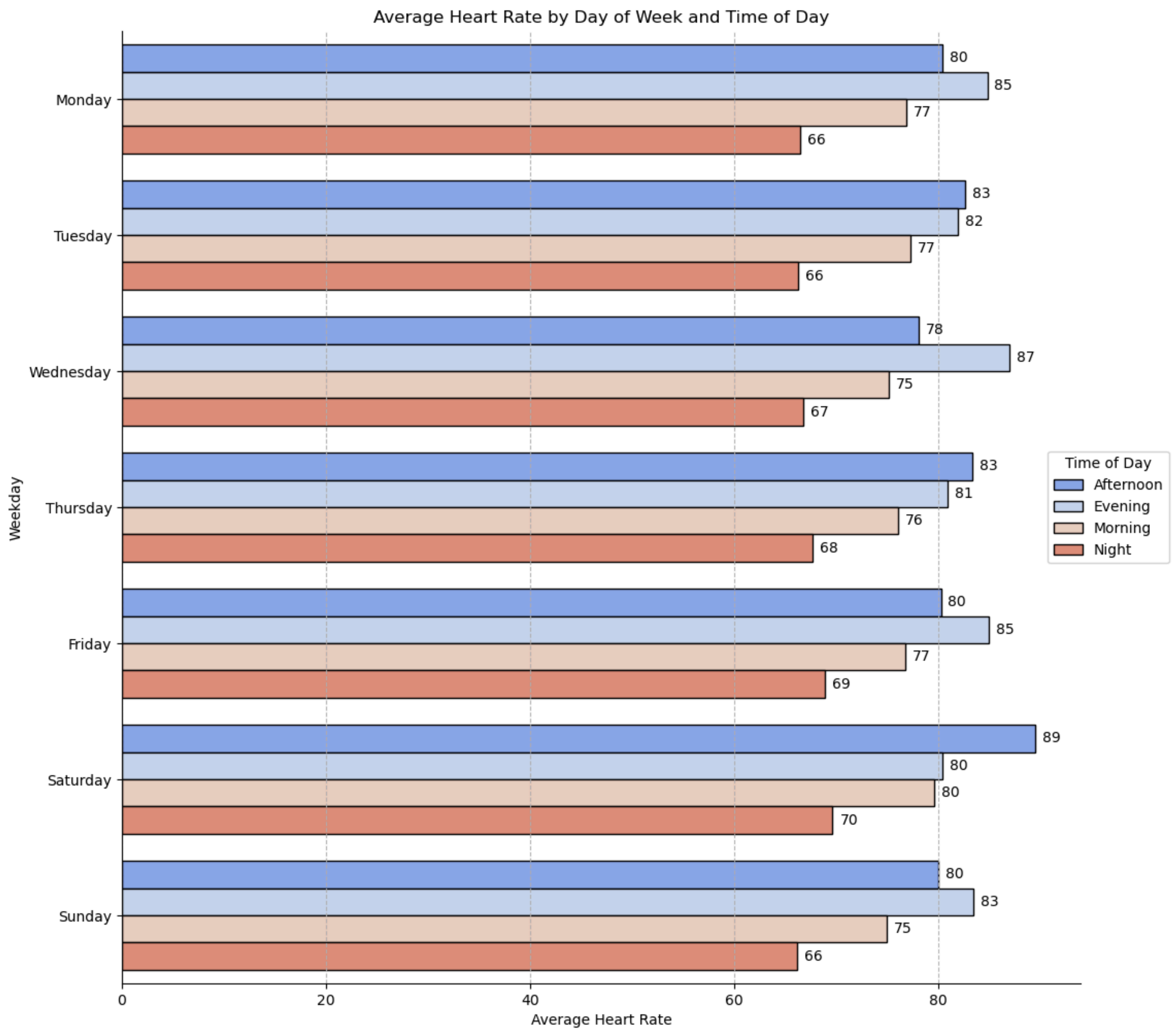
Given that most active minutes involve light activities, such as outdoor jogging on Saturday and Tuesday, these insights suggest strategies to promote physical activity. Emphasizing outdoor exercises could leverage these days to encourage healthier lifestyles and increased activity levels.





In terms of calorie expenditure, Tuesday, Saturday, and Friday stand out as the top three days when people tend to burn the most calories, with Monday, Wednesday, Sunday, and Thursday following in that order. This aligns with the distribution of active minutes across the week, suggesting a correlation between physical activity and calorie burn. Understanding these patterns could inform strategies to promote increased physical activity and calorie expenditure, contributing to better health outcomes.

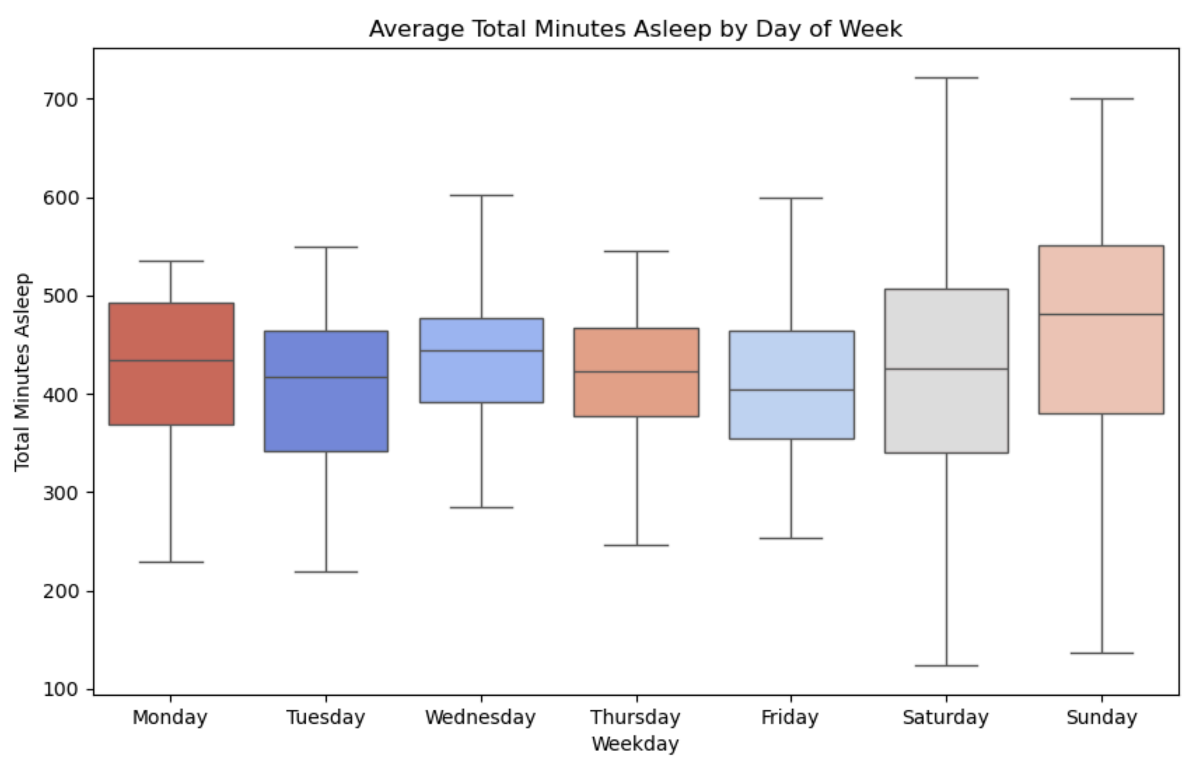
Heart Rate



Analyzing average heart rates by day of the week and time of day reveals patterns in high-intensity exercise. On the most active days, high-intensity workouts are common in the afternoon on Saturday, and in the evening or afternoon on Tuesday. On Friday, these exercises are more often performed in the evening.

These insights can benefit fitness product manufacturers and wellness professionals by enabling them to create personalized recommendations that fit users' schedules and natural tendencies. For example, they might suggest high-intensity workouts during specific times to maximize benefits. Additionally, these patterns can guide the design and marketing of fitness products to better cater to consumer needs, ultimately promoting improved health and wellness outcomes.

Sleep



The box plot analysis reveals that people generally get less total sleep time on their most active days, with Saturday, Friday, and Tuesday ranking among the lowest four. Conversely, on the least active days like Sunday, Wednesday, and Monday, average total sleep minutes tend to be higher.

These findings offer insights for developing personalized recommendations to enhance sleep habits and overall wellness. For instance, on more active days, incorporating relaxation and stress-reducing activities before bedtime could aid in winding down. Conversely, on less active days, engaging in physically demanding activities may promote better sleep quality. Moreover, this information can guide fitness product manufacturers and wellness professionals in designing products that include sleep tracking and offer tailored advice to optimize both sleep and overall health.

Summary:

- **Activity Levels**: Saturday, Tuesday, and Friday are the most active days, while Sunday, Wednesday, and Monday are the least active. However sedentary minutes account for a significant 81.3% of recorded time, indicating extensive periods of inactivity.
- **Activity Breakdown**: Daily, people typically accumulate about 20 minutes of very active time and 15 minutes of fairly active time.

- **Light Activity**: Lightly active minutes vary significantly and are crucial in distinguishing the most active day from the least active.
- **Sedentary Time**: Sedentary minutes constitute a significant 81.3% of recorded time, highlighting prolonged periods of inactivity.
- **Calorie Burn**: Tuesday, Saturday, and Friday are the top days for calorie expenditure, followed by Monday, Wednesday, Sunday, and Thursday.
- **Distance and Steps**: Saturday, Tuesday, and Friday lead in total distance traveled and steps taken.
- **Activity Types**: Outdoor activities are common on Saturday and Tuesday, while Friday sees more indoor activities.
- **High-Intensity Exercise**: Saturday afternoons and Tuesday afternoons or evenings are peak times for high-intensity exercise, with Friday evenings also being popular.
- **Sleep Patterns**: People tend to get less total sleep on their most active days, particularly on Saturday, Friday, and Tuesday.

Phase 6: Act

Recommendation for Bellabeat

Based on the insights provided, Bellabeat can develop a marketing strategy for one of their products as follows:

Product: Bellabeat fitness tracker

Target audience: People who are interested in improving their physical activity levels and overall wellness

Marketing Strategy:

1. **Promote Outdoor Activities**: Highlight the health benefits of outdoor exercises like jogging and hiking through social media campaigns. Use images of people engaging in these activities to encourage outdoor fitness.
2. **Emphasize Light Activities**: Showcase how integrating light activities throughout the day can enhance overall physical activity levels and lead to better health outcomes.
3. **Personalized Workout Recommendations**: Provide tailored exercise suggestions based on users' high-intensity exercise patterns. For example, recommend specific exercises for afternoon sessions based on user data.
4. **Enhance Sleep Habits**: Offer personalized tips to improve sleep quality, especially on days with higher activity levels. Recommend relaxation techniques to help users unwind before bedtime.
5. **Highlight Calorie Burn**: Emphasize how physical activity on specific days can increase calorie burn and improve overall health. Illustrate the correlation between activity levels and fitness goals.
6. **Utilize Social Media Influencers**: Collaborate with health and wellness influencers to promote Bellabeat's fitness tracker. Leverage their credibility to endorse product features and benefits effectively.