How Can a Wellness Technology Company Play It Smart?

A case study on Bellbeat

Introduction

About Company

Bellabeat, a high-tech company that manufactures health-focused smart products. Sršen used her background as an artist to develop beautifully designed technology that informs and inspires women around the world.

Collecting data on activity, sleep, stress, and reproductive health has allowed Bellabeat to empower women with knowledge about their own health and habits. Since it was founded in 2013, Bellabeat has grown rapidly and quickly positioned itself as a tech-driven wellness company for women.

Objective

Bellabeat, believes that analyzing smart device fitness data could help unlock new growth opportunities for the company. You have been asked to focus on one of Bellabeat's products and analyze smart device data to gain insight into how consumers are using their smart devices.

The insights you discover will then help guide marketing strategy for the company. You will present your analysis to the Bellabeat executive team along with your high-level recommendations for Bellabeat's marketing strategy.

About Data

Source

[Download Dataset]

Description

This dataset generated by respondents to a distributed survey via Amazon Mechanical Turk between 03.12.2016-05.12.2016. Thirty eligible Fitbit users consented to the submission of personal tracker data, including minute-level output for physical activity, heart rate, and sleep monitoring.

Data Preparation and Processing

In this analysis, we will concentrate specifically on the Bellabeat Time product. By examining data related to its usage and user behavior, we aim to uncover valuable insights that can inform and enhance Bellabeat's marketing strategy.

Time - This wellness watch combines the timeless look of a classic timepiece with smart technology to track user activity, sleep, and stress. The Time watch connects to the Bellabeat app to provide you with insights into your daily wellness.

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing Datasets

```
import os
os.listdir()
```

```
['.ipynb_checkpoints', 'Bellabeat Case Study.ipynb']
```

```
daily_activity = pd.read_csv(r"C:\Users\Mohit Yadav\Downloads\Fitbit Users Data\dailyActi
```

```
daily_activity.head()
```

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivi
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	

```
daily_activity.describe()
```

	Id	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDis
count	9.400000e+02	940.000000	940.000000	940.000000	940.0
mean	4.855407e+09	7637.910638	5.489702	5.475351	0.1
std	2.424805e+09	5087.150742	3.924606	3.907276	0.6
min	1.503960e+09	0.000000	0.000000	0.000000	0.0
25%	2.320127e+09	3789.750000	2.620000	2.620000	0.0
50%	4.445115e+09	7405.500000	5.245000	5.245000	0.0
75%	6.962181e+09	10727.000000	7.712500	7.710000	0.0
max	8.877689e+09	36019.000000	28.030001	28.030001	4.9

```
daily_activity.isnull().sum()
```

```
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
```

```
daily_activity.info()
```

```
<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
```

ActivityDate attribute has object datatype, it should be in datetime data type.

```
daily_activity["ActivityDate"] = pd.to_datetime(daily_activity["ActivityDate"])
daily_activity["Id"] = (daily_activity["Id"]).astype(str)
```

```
daily_activity["ActivityDate"].dtypes
```

```
dtype('<M8[ns]')
```

```
daily_activity.dtypes
```

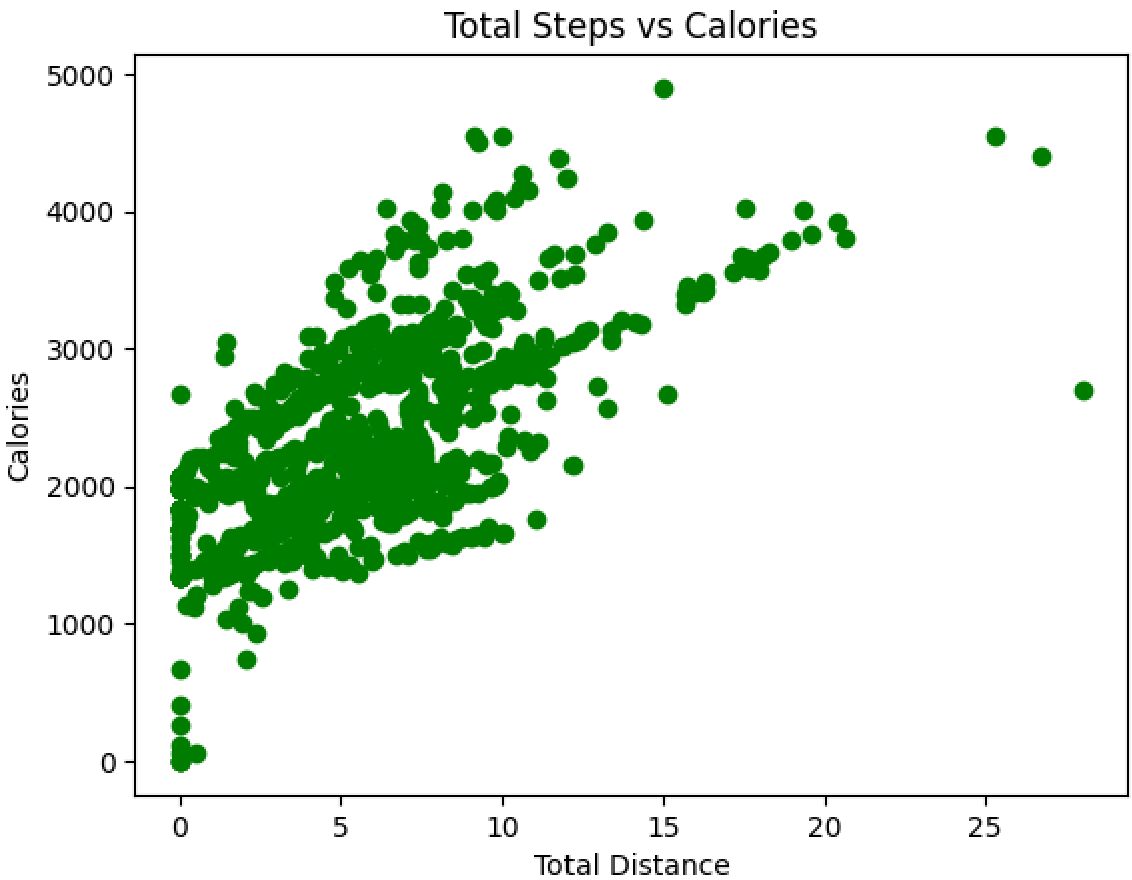
```
Id                                  object
ActivityDate                datetime64[ns]
TotalSteps                           int64
TotalDistance                      float64
TrackerDistance                    float64
LoggedActivitiesDistance           float64
VeryActiveDistance                 float64
ModeratelyActiveDistance           float64
LightActiveDistance                float64
SedentaryActiveDistance            float64
VeryActiveMinutes                    int64
FairlyActiveMinutes                  int64
LightlyActiveMinutes                 int64
SedentaryMinutes                     int64
Calories                             int64
dtype: object
```

```
daily_activity.duplicated().sum()
```

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

Enusing dataset doesn't have any duplicte value

```
 scatter = plt.scatter(daily_activity['TotalDistance'], daily_activity['Calories'], c= "g

plt.xlabel('Total Distance')
plt.ylabel('Calories')
plt.title('Total Steps vs Calories')

plt.show()
```



Ensuring Data Integrity - The scatter plot shows a positive correlation between the total distance and the calories burned. This means that, in general, the more distance someone travels, the more calories they burn.This shows that the data is accurate.

```
(daily_activity["TotalDistance"] != daily_activity["TrackerDistance"]).sum()
```

```
np.int64(15)
```

```
daily_activity[daily_activity["TotalDistance"] != daily_activity["TrackerDistance"]]
```

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActi
689	6962181067	2016-04-21	11835	9.71	7.88	
693	6962181067	2016-04-25	13239	9.27	9.08	
707	6962181067	2016-05-09	12342	8.72	8.68	
711	7007744171	2016-04-12	14172	10.29	9.48	
712	7007744171	2016-04-13	12862	9.65	8.60	
713	7007744171	2016-04-14	11179	8.24	7.48	
717	7007744171	2016-04-18	14816	10.98	9.91	
718	7007744171	2016-04-19	14194	10.48	9.50	
719	7007744171	2016-04-20	15566	11.31	10.41	
724	7007744171	2016-04-25	18229	13.34	12.20	
726	7007744171	2016-04-27	13541	10.22	9.06	
728	7007744171	2016-04-29	20067	14.30	13.42	
731	7007744171	2016-05-02	13041	9.18	8.72	
732	7007744171	2016-05-03	14510	10.87	9.71	
734	7007744171	2016-05-05	15010	11.10	10.04	

Checking total distance is not always equal to tracker distance

```
daily_activity.columns = daily_activity.columns.str.lower()
```

```
daily_activity.columns
```

```
Index(['id', 'activitydate', 'totalsteps', 'totaldistance', 'trackerdistance',
       'loggedactivitiesdistance', 'veryactivedistance',
       'moderatelyactivedistance', 'lightactivedistance',
       'sedentaryactivedistance', 'veryactiveminutes', 'fairlyactiveminutes',
       'lightlyactiveminutes', 'sedentaryminutes', 'calories'],
      dtype='object')
```

```
daily_activity["activitydate"].min()
```

```
Timestamp('2016-04-12 00:00:00')
```

```
daily_activity["activitydate"].max()
```

```
Timestamp('2016-05-12 00:00:00')
```

```
daily_activity["weekday"] = daily_activity["activitydate"].dt.day_name()
daily_activity["day"] = daily_activity["activitydate"].dt.dayofweek
```

```
daily_activity.head()
```

	id	activitydate	totalsteps	totaldistance	trackerdistance	loggedactivi
0	1503960366	2016-04-12	13162	8.50	8.50	
1	1503960366	2016-04-13	10735	6.97	6.97	
2	1503960366	2016-04-14	10460	6.74	6.74	
3	1503960366	2016-04-15	9762	6.28	6.28	
4	1503960366	2016-04-16	12669	8.16	8.16	

```
sleep = pd.read_csv(r"C:\Users\Mohit Yadav\Downloads\Fitbit Users Data\sleepDay_merged.cs
sleep.head()
```

	Id	SleepDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed
0	1503960366	4/12/2016 12:00:00 AM	1	327	346
1	1503960366	4/13/2016 12:00:00 AM	2	384	407
2	1503960366	4/15/2016 12:00:00 AM	1	412	442
		4/16/2016			

Start coding or generate with AI.

```
sleep.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 413 entries, 0 to 412
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Id                  413 non-null    int64
 1   SleepDay            413 non-null    object
 2   TotalSleepRecords   413 non-null    int64
 3   TotalMinutesAsleep  413 non-null    int64
 4   TotalTimeInBed      413 non-null    int64
dtypes: int64(4), object(1)
memory usage: 16.3+ KB
```

```
sleep.isnull().sum()
```

```
Id                    0
SleepDay              0
TotalSleepRecords     0
```

```
TotalMinutesAsleep    0
TotalTimeInBed        0
dtype: int64
```

```
sleep.nunique()
```

```
Id                     24
SleepDay               31
TotalSleepRecords       3
TotalMinutesAsleep    256
TotalTimeInBed        242
dtype: int64
```

```
sleep["Id"].value_counts()
```

```
Id
8378563200    32
5553957443    31
6962181067    31
3977333714    28
4445114986    28
2026352035    28
4702921684    28
4319703577    26
5577150313    26
1503960366    25
7086361926    24
4388161847    24
6117666160    18
2347167796    15
8792009665    15
4020332650     8
4558609924     5
1927972279     5
1644430081     4
1844505072     3
8053475328     3
6775888955     3
7007744171     2
2320127002     1
Name: count, dtype: int64
```

```
def convert(column):
    result = ""
    for index,i in enumerate(column):
        if i.isupper() and index != 0:
            result += "_"
            result += i.lower()
        else:
            result += i.lower()
    return result
```

```
columns = list(sleep.columns)
converted_columns = [convert(i) for i in columns]
sleep.columns = converted_columns
```

```
sleep.head()
```

	id	sleep_day	total_sleep_records	total_minutes_asleep	total_time_in_b
0	1503960366	4/12/2016 12:00:00 AM	1	327	3
1	1503960366	4/13/2016 12:00:00 AM	2	384	4
		4/15/2016			

```
# Convert 'sleep_day' to datetime
sleep['sleep_day'] = pd.to_datetime(sleep['sleep_day'], format='%d/%m/%Y %I:%M:%S %p')

# Extract hour from 'sleep_day' and create 'Time' column
sleep['Time'] = sleep['sleep_day'].dt.strftime('%H')

# Drop the 'Time' column
sleep.drop(['Time'], axis=1, inplace=True)
```

```
sleep["id"] = sleep["id"].astype(str)
```

```
sleep["sleep_day"].dtype
```

```
dtype('<M8[ns]')
```

```
sleep.head()
```

	id	sleep_day	total_sleep_records	total_minutes_asleep	total_time_in_b
0	1503960366	2016-04-12	1	327	3
1	1503960366	2016-04-13	2	384	4
2	1503960366	2016-04-15	1	412	4

```
sleep[sleep.duplicated()]
```

	id	sleep_day	total_sleep_records	total_minutes_asleep	total_time_in_
161	4388161847	5/5/2016 12:00:00 AM	1	471	
		5/7/2016			

```
sleep.drop_duplicates(inplace = True)
```

```
sleep.duplicated().sum()
```

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

```
calorie_time = pd.read_csv(r"C:\Users\Mohit Yadav\Downloads\Fitbit Users Data\hourlyCalor
```

```
calorie_time.head()
```

	Id	ActivityHour	Calories
0	1503960366	4/12/2016 12:00:00 AM	81
1	1503960366	4/12/2016 1:00:00 AM	61
2	1503960366	4/12/2016 2:00:00 AM	59
3	1503960366	4/12/2016 3:00:00 AM	47
4	1503960366	4/12/2016 4:00:00 AM	48

```
calorie_time.columns = [ convert(i) for i in list(calorie_time.columns)]
```

```
calorie_time.head()
```

	id	activity_hour	calories
0	1503960366	4/12/2016 12:00:00 AM	81
1	1503960366	4/12/2016 1:00:00 AM	61
2	1503960366	4/12/2016 2:00:00 AM	59
3	1503960366	4/12/2016 3:00:00 AM	47
4	1503960366	4/12/2016 4:00:00 AM	48

```
calorie_time.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22099 entries, 0 to 22098
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
```

```
---  ------         --------------  -----
 0   id             22099 non-null  int64
 1   activity_hour  22099 non-null  object
 2   calories       22099 non-null  int64
dtypes: int64(2), object(1)
memory usage: 518.1+ KB
```

```
calorie_time.duplicated().sum()
```

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

```
calorie_time.head()
```

	id	activity_hour	calories
0	1503960366	4/12/2016 12:00:00 AM	81
1	1503960366	4/12/2016 1:00:00 AM	61
2	1503960366	4/12/2016 2:00:00 AM	59
3	1503960366	4/12/2016 3:00:00 AM	47
4	1503960366	4/12/2016 4:00:00 AM	48

```
# Convert 'activity_hour' to datetime
calorie_time['activity_hour'] = pd.to_datetime(calorie_time['activity_hour'], format='%m/

# Extract hour from 'activity_hour' and create 'hour' column
calorie_time['hour'] = calorie_time['activity_hour'].dt.strftime('%H')
```

```
calorie_time.tail()
```

	id	activity_hour	calories	hour
22094	8877689391	2016-05-12 10:00:00	126	10
22095	8877689391	2016-05-12 11:00:00	192	11
22096	8877689391	2016-05-12 12:00:00	321	12
22097	8877689391	2016-05-12 13:00:00	101	13
22098	8877689391	2016-05-12 14:00:00	113	14

```
calorie_time.dtypes
```

```
id                        int64
activity_hour    datetime64[ns]
calories                  int64
hour                     object
dtype: object
```

Analysis

Physically Active Users based on steps taken

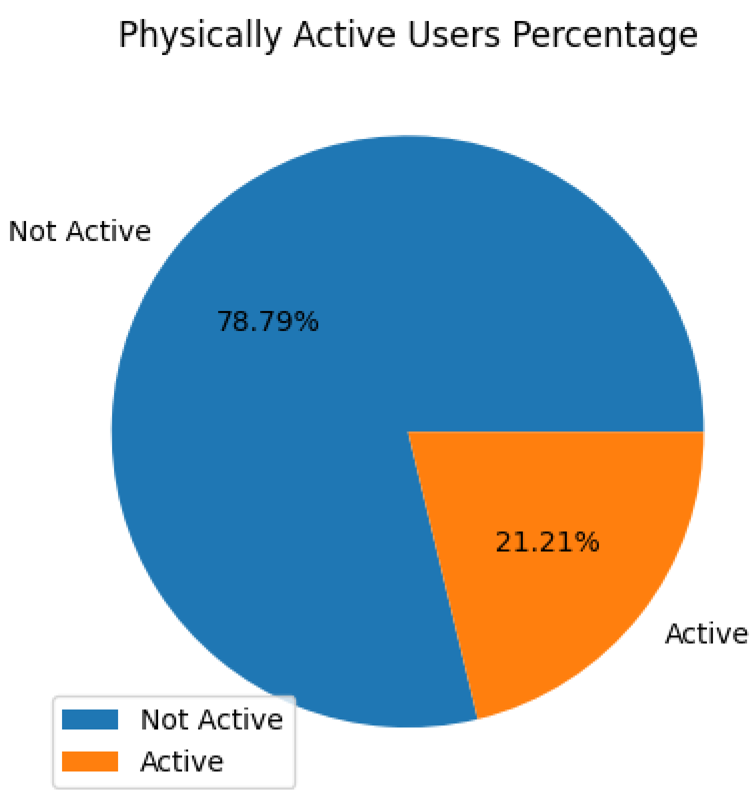
I am considering a user who took more than or equal to 10000 steps a day as a physically active user

```
avg_steps = daily_activity.groupby(["id"])["totalsteps"].mean().to_frame("avg_daily_steps
```

```
avg_steps["is_active"] = np.where(avg_steps["avg_daily_steps"] >= 10000,"Active" , "Not A
physically_active_users = avg_steps["is_active"].value_counts().reset_index()
physically_active_users
```

	is_active	count
0	Not Active	26
1	Active	7

```
plt.pie(physically_active_users["count"], labels = physically_active_users["is_active"],a
plt.title("Physically Active Users Percentage")
plt.legend(physically_active_users["is_active"]);
```



Based on the analysis of one month's data, we observed that 22% of users are physically active within the month that means there average total steps are greater than 10000. This indicates a strong level of engagement among this segment, highlighting their commitment to their health.

We can only calculate average total steps of the days when users logged in, we do not know about the days when the user did not logged in his/her device.

Type of active users based on No. of days device is logged in

- High — users who use their device for 21–31 days
- Medium — users who use their device for 10–20 days
- Low — users who use their device for 1–10 days

```
user_active_days = daily_activity.groupby('id')['activitydate'].nunique().reset_index(nam
user_active_days.head()
```

	id	total_active_days
0	1503960366	31
1	1624580081	31
2	1644430081	30
3	1844505072	31
4	1927972279	31

```
def using_frequency(active_days):
    if active_days >= 20:
        return "High"
    elif active_days >=11:
        return "Medium"
    else:
        return "low"
```

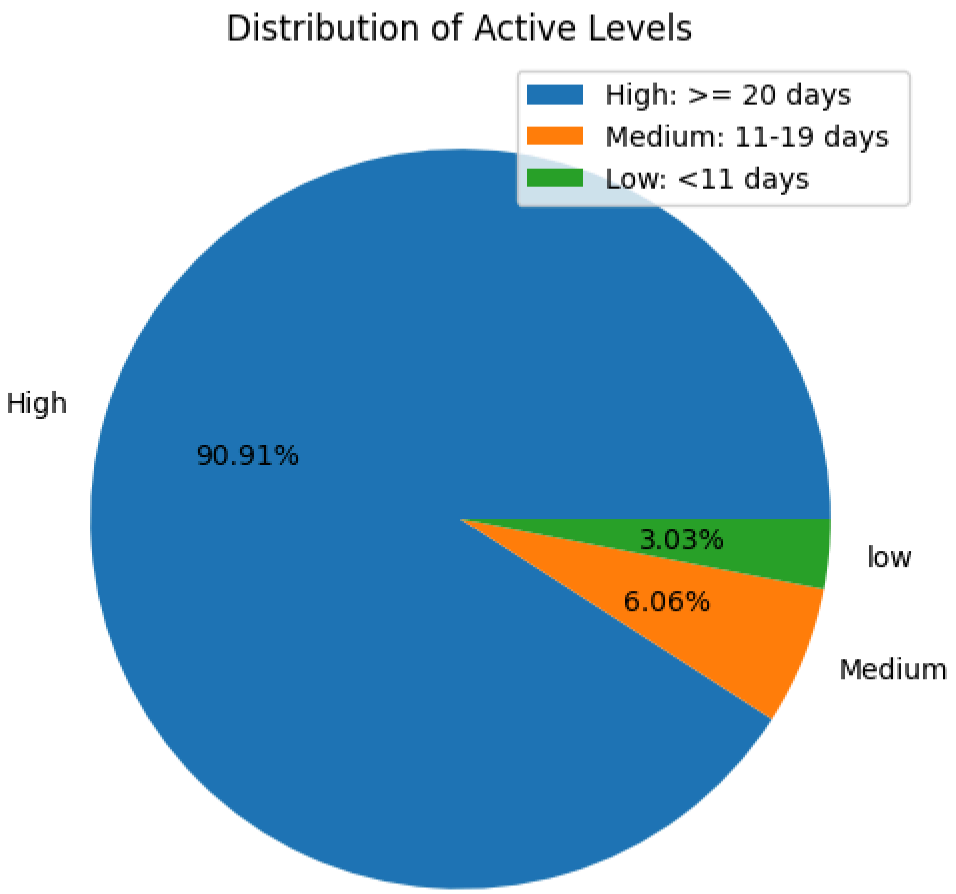
```
user_active_days["active_level"] = user_active_days["total_active_days"].apply(using_freq
user_active_days.head()
```

	id	total_active_days	active_level
0	1503960366	31	High
1	1624580081	31	High
2	1644430081	30	High
3	1844505072	31	High
4	1927972279	31	High

```
active_level_count = user_active_days["active_level"].value_counts().reset_index()
active_level_count
```

	active_level	count
0	High	30
1	Medium	2
2	low	1

```
plt.figure(figsize = (6,6))
plt.pie(
    active_level_count['count'],
    labels=active_level_count['active_level'],
    autopct='%1.2f%%'
)

# Create a legend with conditions
conditions = {
    'High': '>= 20 days',
    'Medium': '11-19 days ',
    'Low': '<11 days'
}

# Add legend to the plot
plt.legend(
    loc='upper right',
    labels=[f"{level}: {cond}" for level, cond in conditions.items()]
)

plt.title('Distribution of Active Levels')
plt.show();
```



High Engagement: 90% of users are active for more than 20 days out of the month.

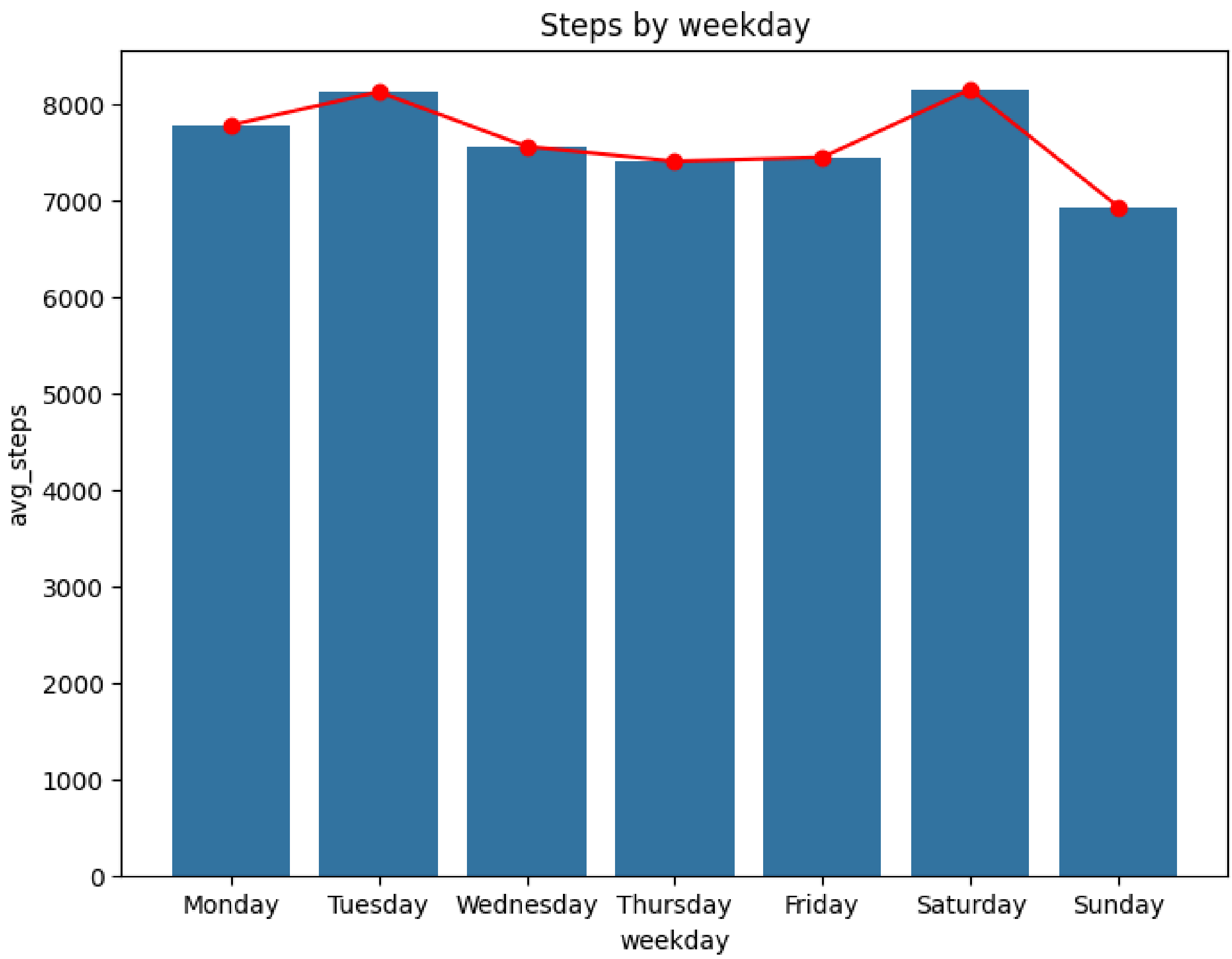
User Segment: This suggests that most users find the fitness tracker valuable and are consistently using it.

Average steps by each weekday

```
weekday_steps_count = daily_activity.groupby(["weekday","day"])["totalsteps"].mean().rese

weekday_steps_count
```

	weekday	day	avg_steps
1	Monday	0	7780.866667
5	Tuesday	1	8125.006579
6	Wednesday	2	7559.373333
4	Thursday	3	7405.836735
0	Friday	4	7448.230159
2	Saturday	5	8152.975806
3	Sunday	6	6933.231405

```
plt.figure(figsize=(8,6))
sns.barplot(x = "weekday", y = "avg_steps", data = weekday_steps_count)
plt.plot(weekday_steps_count["weekday"],weekday_steps_count["avg_steps"],"r-o")
plt.title("Steps by weekday");
```



The chart shows the average number of steps taken each day of the week. The highest average steps were taken on Tuesdays, and the lowest average steps were taken on Sundays.

Effect on sleep of the users who take >= 10000 average steps

```
sleep.head()
```

	id	sleep_day	total_sleep_records	total_minutes_asleep	total_time_in_b
0	1503960366	2016-04-12	1	327	3
1	1503960366	2016-04-13	2	384	4
2	1503960366	2016-04-15	1	412	4

```
sleep["sleep_ratio_on_bed"] = round((sleep["total_minutes_asleep"])*100/sleep["total_time
```

```
sleep.head()
```

	id	sleep_day	total_sleep_records	total_minutes_asleep	total_time_in_b
0	1503960366	2016-04-12	1	327	3
1	1503960366	2016-04-13	2	384	4
2	1503960366	2016-04-15	1	412	4

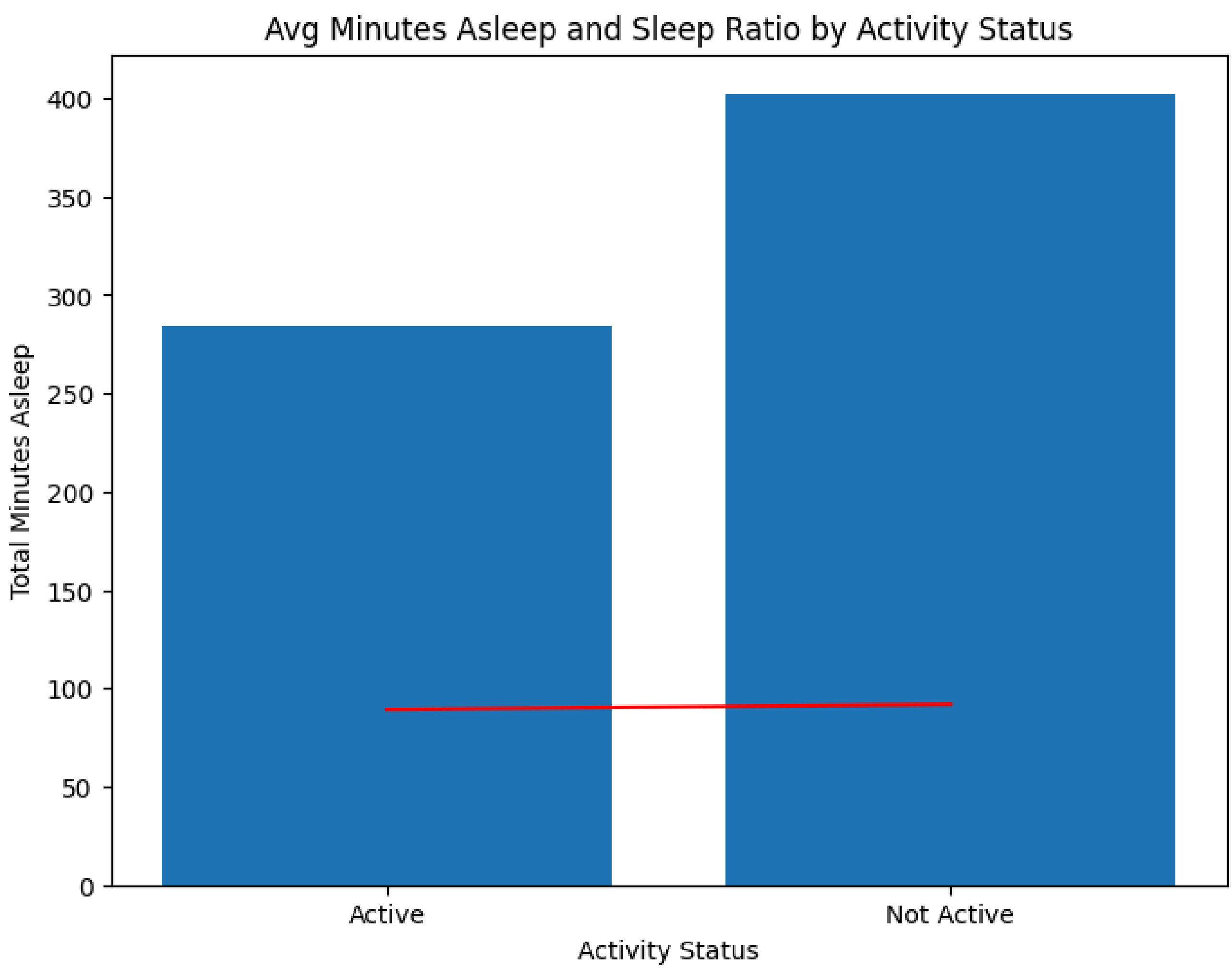
```
avg_sleep = sleep.groupby("id")[["total_minutes_asleep","sleep_ratio_on_bed"]].mean().res
```

```
merged_df= avg_steps.merge(avg_sleep, on = "id")
```

```
effect_on_sleep = merged_df.groupby("is_active")[["total_minutes_asleep","sleep_ratio_on_
effect_on_sleep
```

	is_active	total_minutes_asleep	sleep_ratio_on_bed
0	Active	283.919354	89.185884
1	Not Active	402.058762	91.860436

```
fig , ax = plt.subplots(figsize = (8,6))
ax.bar(effect_on_sleep["is_active"],effect_on_sleep["total_minutes_asleep"])
ax.set_xlabel('Activity Status')
ax.set_ylabel('Total Minutes Asleep')
ax.plot(effect_on_sleep["is_active"],effect_on_sleep["sleep_ratio_on_bed"],c="red")
plt.title('Avg Minutes Asleep and Sleep Ratio by Activity Status')
plt.show()
```



Bar representing avg sleep minutes of active and non active users and line showing sleep ration while on bed

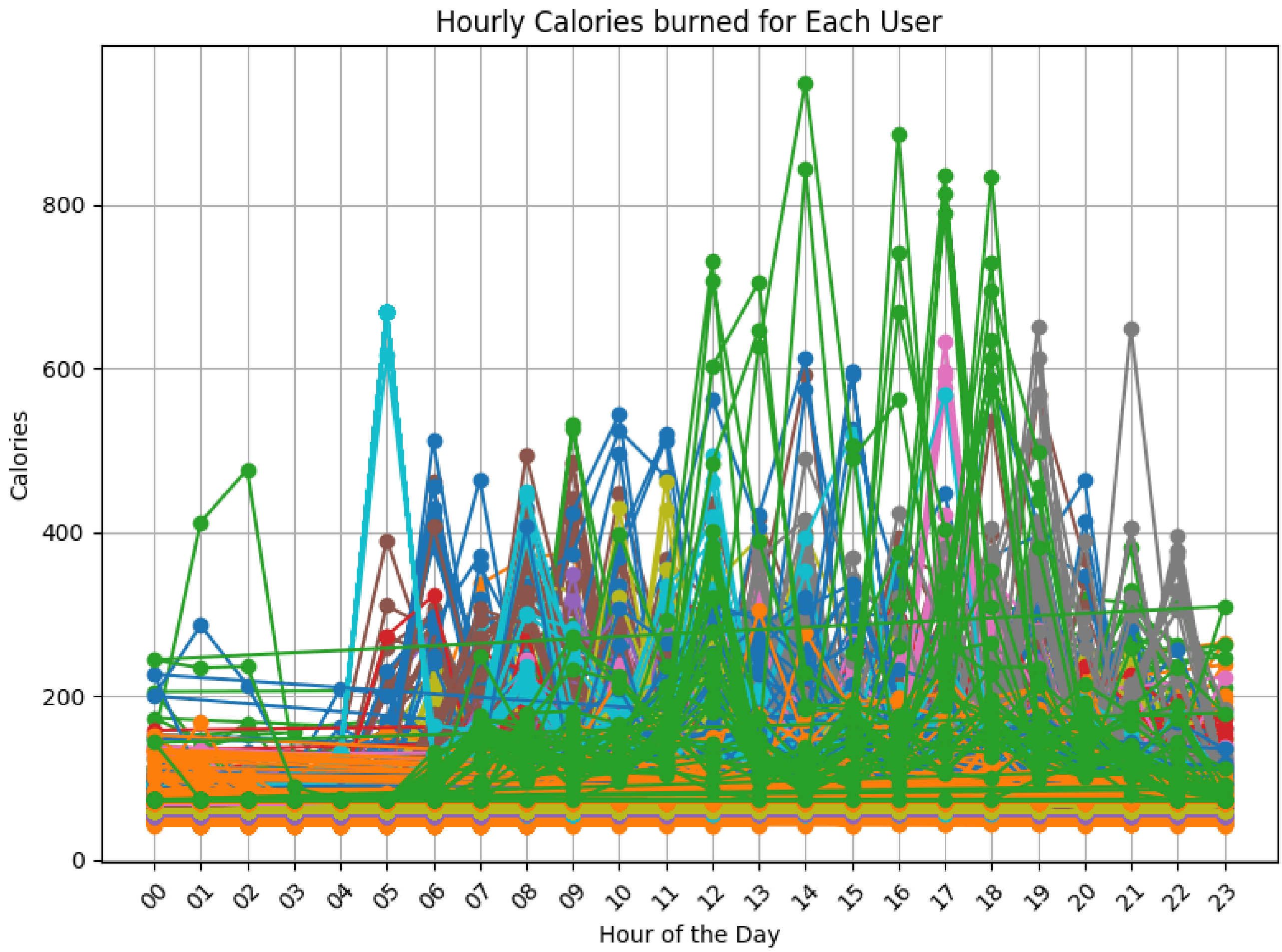
People who are physically active sleep less than the non active people, but the sleep ratio while on bed for active and non active is nearly same that shows that active people probably getting less time to sleep.

Calories trend during whole day

```
calorie_time.head()
```

	id	activity_hour	calories	hour
0	1503960366	2016-04-12 00:00:00	81	00
1	1503960366	2016-04-12 01:00:00	61	01
2	1503960366	2016-04-12 02:00:00	59	02
3	1503960366	2016-04-12 03:00:00	47	03
4	1503960366	2016-04-12 04:00:00	48	04

```
plt.figure(figsize = (8,6))
for user_id, user_data in calorie_time.groupby('id'):
    plt.plot(user_data['hour'], user_data['calories'], marker='o')

plt.xlabel('Hour of the Day')
plt.ylabel('Calories')
plt.title('Hourly Calories burned for Each User')

plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

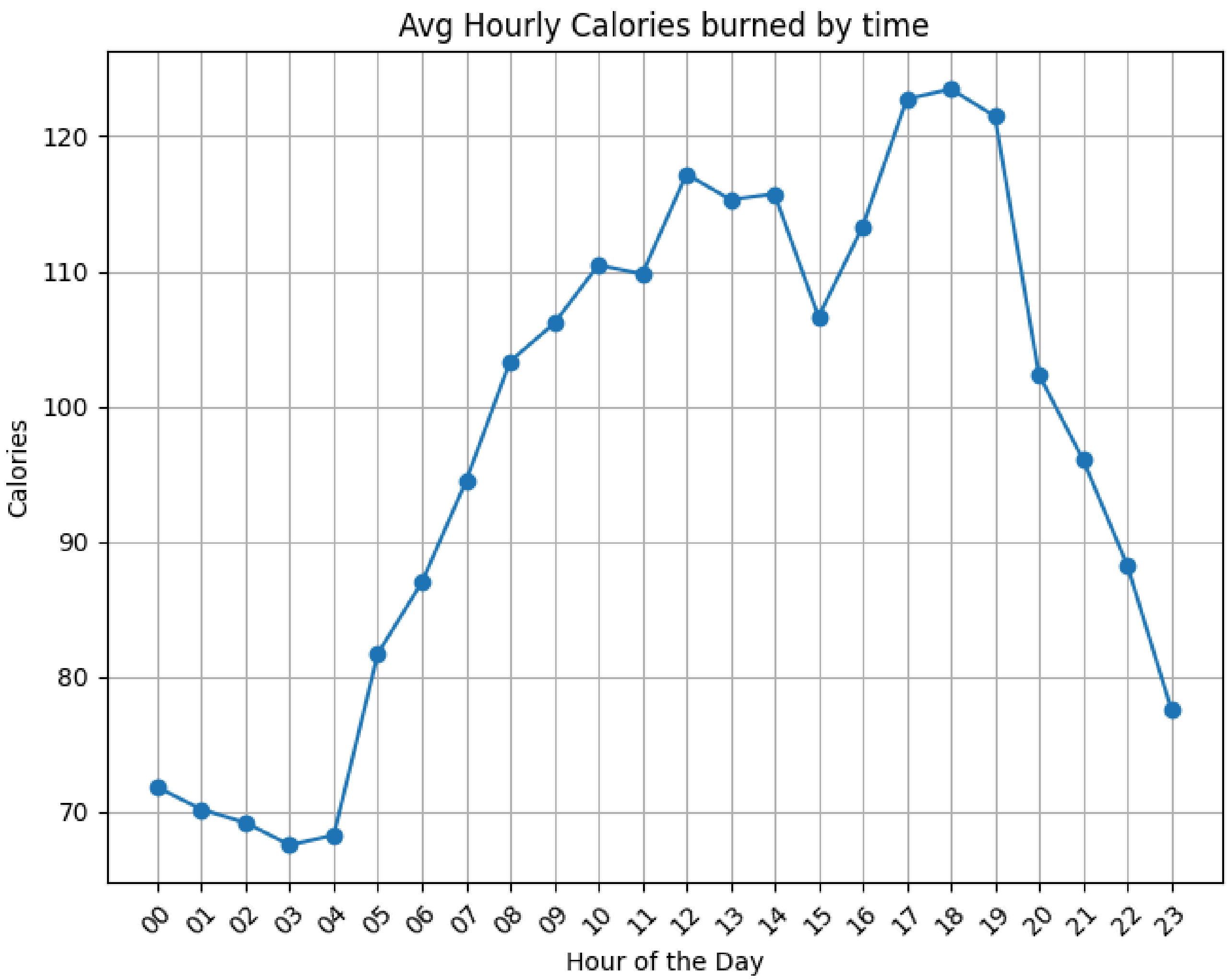


The graph shows the hourly calories burned by each user. The X axis represents the hours of the day and the Y axis represents calories burned. Each line represents a different user. We can see that some users burn more calories than others, and that the amount of calories burned varies throughout the day. For example, many users burn the most calories between 14 and 17. The lines also indicate the different activity levels of the users.

```
calorie_time.head()
```

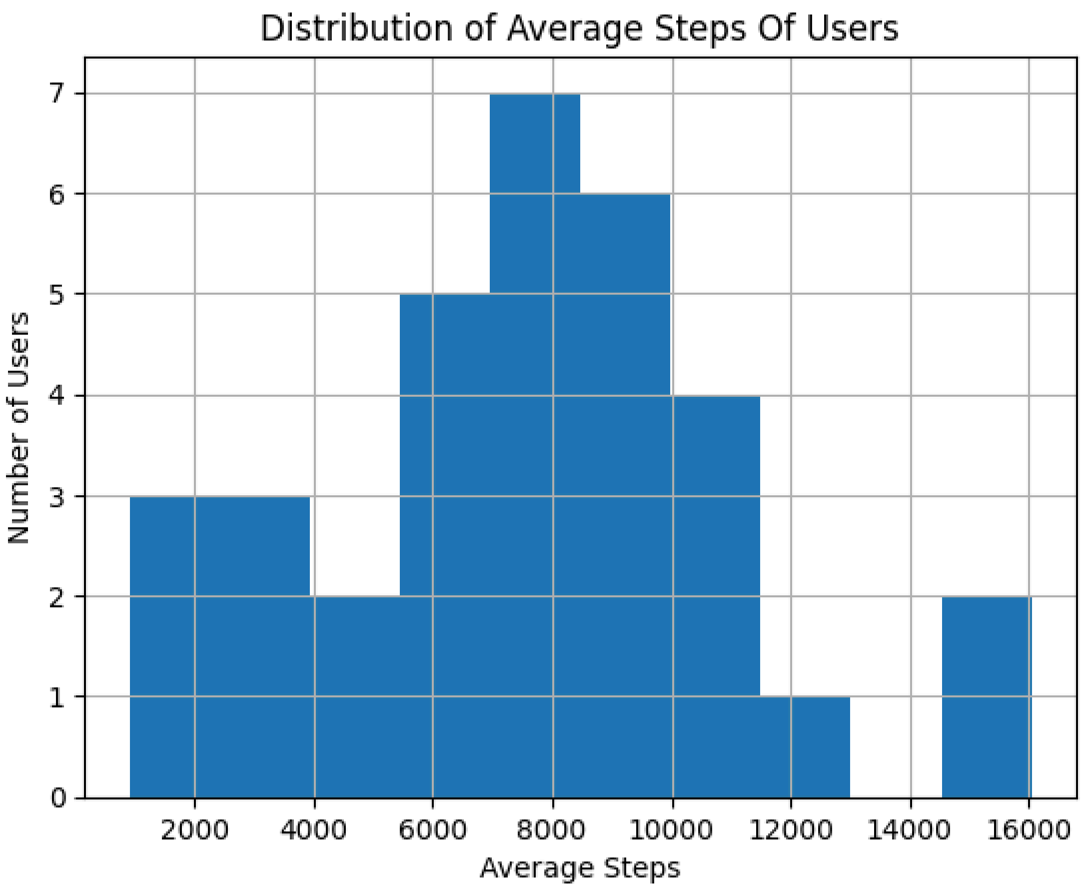
	id	activity_hour	calories	hour
0	1503960366	2016-04-12 00:00:00	81	00
1	1503960366	2016-04-12 01:00:00	61	01
2	1503960366	2016-04-12 02:00:00	59	02
3	1503960366	2016-04-12 03:00:00	47	03
4	1503960366	2016-04-12 04:00:00	48	04

```
grouped_hrs = calorie_time.groupby("hour")["calories"].mean().reset_index()
plt.figure(figsize = (8,6))

plt.plot(grouped_hrs['hour'],grouped_hrs['calories'], marker='o')

plt.xlabel('Hour of the Day')
plt.ylabel('Calories')
plt.title('Avg Hourly Calories burned by time')
plt.grid(True)
plt.xticks(rotation=45)

plt.show()
```



Distribution of Avg Steps

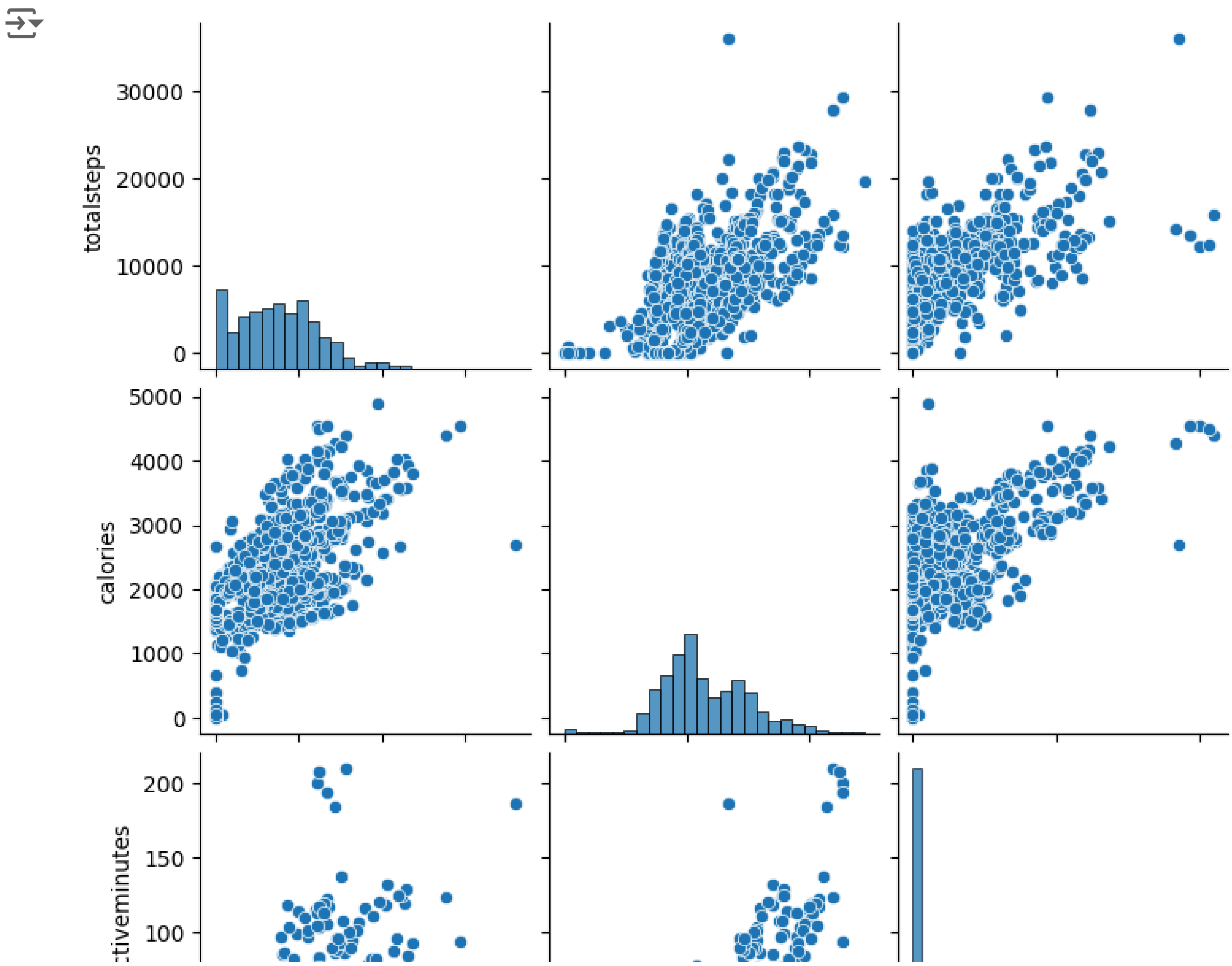
```
plt.hist(avg_steps["avg_daily_steps"],bins = 10)
plt.xlabel("Average Steps")
plt.ylabel("Number of Users")
plt.title("Distribution of Average Steps Of Users")
plt.grid(True)  # Add grid lines for easier reading
plt.show();
```



Most users take between 6,000 and 10,000 steps on average. This is the largest bar in the histogram, indicating that this range contains the most users.

Correlation between variables

```
columns_to_plot = ["totalsteps", "calories", "veryactiveminutes"]
pairplot_data = daily_activity[columns_to_plot]

# Create the pair plot
sns.pairplot(pairplot_data)

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



The scatterplots show a positive correlation between total steps and calories burned, which indicates that the more steps someone takes, the more calories they tend to burn. There's also a positive correlation between total steps and very active minutes, which indicates that people who are more active tend to take more steps.

Act

90% of users logged in device for more than 20 days out of the month that shows that most users find fitness tracker valuable and are consistently using it.