Tanner Lilienthal
Bellabeat Case Study

How are consumers using non-Bellabeat smart devices?

Understanding trends in other smart devices in order to make suggestions to stakeholders about one or more Bellabeat products (the app, Leaf, Time watch, and Spring water bottle) or other smart device marketspace that they might penetrate.

FitBit Fitness Tracker Data
This public Kaggle.com dataset includes data from 30 consenting FitBit users who agreed to provide minute-level output for physical activity, heart rate, and sleep monitoring. It includes information about daily activity, steps, and heart rate.

The provided data contained 18 separate .csv files which were downloaded and stored in a file on my computer named “fitbit_data”.

The data was then skimmed for headers manually using Excel. Doing this allowed me to understand the data at a surface level and even remove some files due to their contents being duplicates. The 10 files that remained are named:

- daily_activity
- day_sleep
- heartrate_seconds
- hourly_calories
- hourly_intensities
- hourly_steps
- minute_calories
- minute_intensities
- minute_steps
- weight_log

Out of personal preference, data was kept in a narrow format where necessary. Being that the sample size for this dataset is only 30 users of FitBit who opted-in, this is not a truly random sample and does inherently create some bias. The data was gathered in 2016 and there is no participant identifiable information (including but not limited to gender, age, race, occupation, and location). All of these factors make the data unreliable and should not lead to any meaningful business recommendations. I will still be able to analyze the data for trends and answer the business task.

For cleaning the data I will be using SQL, specifically PostgreSQL which I use through pgAdmin 4. I manually created tables, imported the data from the 10 .csv files, and ensured things were imported properly with the correct formats. In an effort to maintain clarity of this document, all queries will be presented in a separate document.

First step I took in the cleaning process was to check for duplicates. I already did this manually by skimming the 18 different files as mentioned above but not within the many rows of each table I want to ensure there are none hiding. On the other document you can see after the first 10 “CREATE TABLE” queries what I used to verify the integrity of, clean, and increase my familiarity of the data with.

Below are the discoveries I made about the data during this process, in no particular order:

1. 33 distinct id’s exist, 3 more than expected
2. day_sleep, heartrate_seconds, and weight_log had less than 33 distinct id’s with weight log having only 8
3. The dates reflect one month of user data, specifically from April 12, 2016 to May 12, 2016

There are now no more duplicates, nulls or extreme outliers.

I then decided to add a column for day_of_week onto each table and during this process I decided to lump together the three different hourly_* tables to reduce the amount of times I would have to input the same query, plus having the data spread over the different tables was needlessly expensive in terms of space. I then renamed the table for more clarity. I attempted to try the same queries for the minute_* tables but was taking multiple hours to conclude so I aborted and decided to leave them separated.

After all the cleaning was complete, at least enough to make me content, I downloaded the new and updated tables as .csv files. I then transitioned over to Python and Tableau for further analysis.

Three trends were discovered in my analysis. I will provide the full python script at my github while here I will only show the relevant functions used to further understand each relationship. I created standard description tables as my first action. The code and tables are shown below:

```
def print_out_summary_statistics():
   tables = [daily_activity, day_sleep, heartrate_seconds, hourly_activity,
             minute_calories, minute_intensities, minute_steps, weight_log]
   for table in tables:
       numeric_data_types = ['int64', 'float64']
       numeric_columns = []
       for column in table:
           if table[column].dtype in numeric_data_types:
               numeric_columns.append(column)
       print(f'==========={table.name}===========')
       print(table.describe())
```

Figure 1a - daily_activity_description

	id	total steps	total distance	tracker distance	logged activite distance	very active distance	moderately active distance
count	940	940	940	940	940	940	940
mean	4855407369	7637.911	5.490	5.475	0.108	1.503	0.568
std	2424805476	5087.151	3.925	3.907	0.620	2.659	0.884
min	1503960366	0	0	0	0	0	0
25%	2320127002	3789.75	2.62	2.62	0	0	0
50%	4445114986	7405.5	5.245	5.245	0	0.21	0.24
75%	6962181067	10727	7.7125	7.71	0	2.0525	0.8
max	8877689391	36019	28.03	28.03	4.942142	21.92	6.48

Figure 1b - daily_activity_description (cont.)

	logged activity distance	very active distance	moderately active distance	light active distance	sedentary active distance
count	940	940	940	940	940
mean	0.1082	1.5027	0.5675	3.3408	0.0016
std	0.6199	2.6589	0.8836	2.0407	0.0073
min	0	0	0	0	0
25%	0	0	0	1.945	0
50%	0	0.21	0.24	3.365	0
75%	0	2.0525	0.8	4.7825	0
max	4.942142	21.92	6.48	10.71	0.11

Figure 1c - daily_activity_description (cont. again)

	very active minutes	fairy active minutes	lightly active minutes	sedentary active minutes	calories
count	940	940	940	940	940
mean	21.165	13.565	192.813	991.211	2303.610
std	32.845	19.987	109.175	301.267	718.167
min	0	0	0	0	0
25%	0	0	127	729.75	1828.5
50%	4	6	199	1057.5	2134
75%	32	19	264	1229.5	2793.25
max	210	143	518	1440	4900

Figure 2 - day_sleep_description

	id	total sleep records	total minutes asleep	total time in bed
count	413	413	413	413
mean	5000979403	1.1186	419.4673	458.6392
std	2060360174	0.3455	118.3447	127.1016
min	1503960366	1	58	61
25%	3977333714	1	361	403
50%	4702921684	1	433	463
75%	6962181067	1	490	526
max	8792009665	3	796	961

Figure 3 - heartrate_seconds_description

	id	heartrate
count	2483658	2483658
mean	5513764629	77.328
std	1950223761	19.404
min	2022484408	36
25%	4388161847	63
50%	5553957443	73
75%	6962181067	88
max	8877689391	203

Figure 4 - hourly_activity_description

	id	calories	total intensity	average intensity	step total
count	22099	22099	22099	22099	22099
mean	4848235270	97.387	12.035	0.201	320.166
std	2422500401	60.703	21.133	0.352	690.384
min	1503960366	42	0	0	0
25%	2320127002	63	0	0	0
50%	4445114986	83	3	0.05	40
75%	6962181067	108	16	0.26666668	357
max	8877689391	948	180	3	10554

Figure 5 - minute_calories_description

	id	calories
count	1325580	1325580
mean	4847897692	1.623
std	2422313222	1.410
min	1503960366	0
25%	2320127002	0.9357
50%	4445114986	1.218
75%	6962181067	1.433
max	8877689391	19.75

Figure 6 - minute_intensity_description

	id	intensity
count	1325580	1325580
mean	4847897692	0.201
std	2422313222	0.519
min	1503960366	0
25%	2320127002	0
50%	4445114986	0
75%	6962181067	0
max	8877689391	3

Figure 7 - minute_steps_description

	id	steps
count	1325580	1325580
mean	4847897692	5.336
std	2422313222	18.128
min	1503960366	0
25%	2320127002	0
50%	4445114986	0
75%	6962181067	0
max	8877689391	220

Figure 8 - weight_log_description

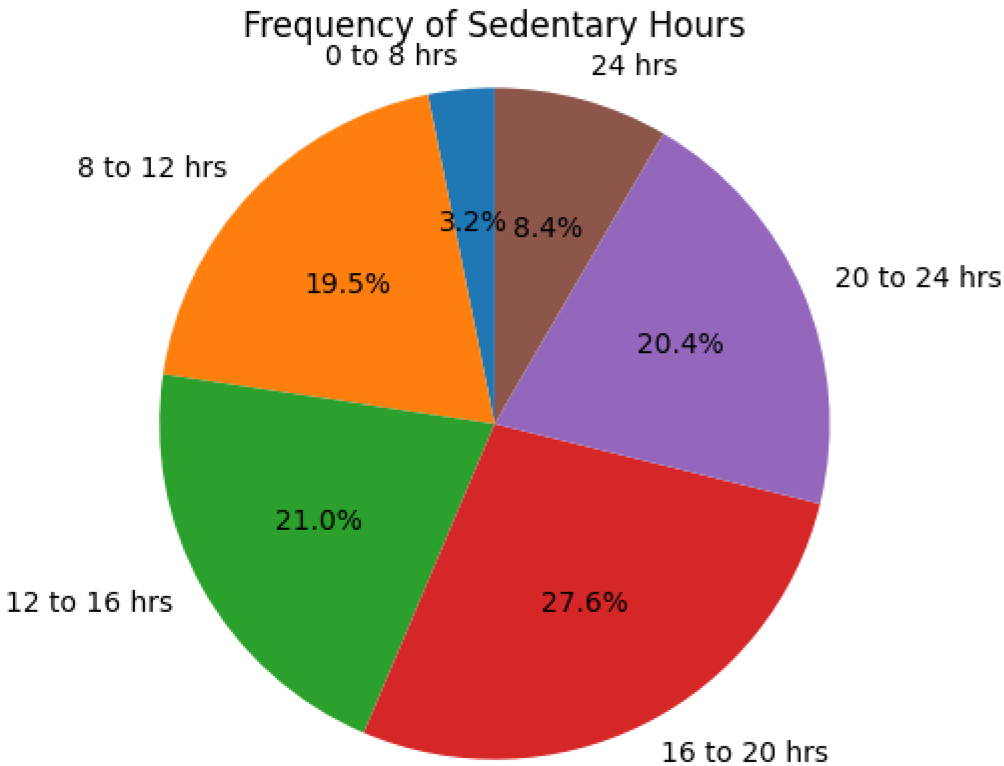
	id	weight kg	weight lb	bmi
count	67	67	67	67
mean	7009282135	72.036	158.812	25.185
std	1950321944	13.923	30.695	3.067
min	1503960366	52.6	115.96315	21.45
25%	6962181067	61.4	135.36383	23.96
50%	6962181067	62.5	137.78891	24.39
75%	8877689391	85.05	187.50316	25.56
max	8877689391	133.5	294.3171	47.54

First I noticed that the amount of time spent sedentary was quite high. I did a count of specific ranges of hours a day spent sedentary resulting in the pie chart in Figure 9 below. As you can see 56.4% of users are sedentary more than three-fourths of the day. While it is tough to determine the cause of this, the consumers who are spending 20+ hours a day sedentary are likely not taking their devices with them unless doing non-sedentary activities. Potential for the customer acquisition team to educate about the importance of wearing

```python
def sedentary_hours_visual():
    # sectors for 0-8hr, 8-12, 12-16, 16-20, 20-24, and 24
    zero_to_eight = daily_activity[daily_activity['sedentary_minutes'] < 480]
    eight_to_twelve = daily_activity[(daily_activity['sedentary_minutes'] >= 480) &
                                     (daily_activity['sedentary_minutes'] < 720)]
    twelve_to_sixteen = daily_activity[(daily_activity['sedentary_minutes'] >= 720) &
                                     (daily_activity['sedentary_minutes'] < 960)]
    sixteen_to_twenty = daily_activity[(daily_activity['sedentary_minutes'] >= 960) &
                                     (daily_activity['sedentary_minutes'] < 1200)]
    twenty_to_twentyfour = daily_activity[(daily_activity['sedentary_minutes'] >= 1200) &
                                     (daily_activity['sedentary_minutes'] < 1440)]
    twentyfour = daily_activity[daily_activity['sedentary_minutes'] == 1440]

    # make pie showing consecutive hours sedentary
    labels = ['0 to 8 hrs', '8 to 12 hrs', '12 to 16 hrs',
              '16 to 20 hrs', '20 to 24 hrs', '24 hrs']
    sizes = [zero_to_eight['id'].count(), eight_to_twelve['id'].count(),
             twelve_to_sixteen['id'].count(), sixteen_to_twenty['id'].count(),
             twenty_to_twentyfour['id'].count(), twentyfour['id'].count()]
    fig1, pie_chart = plt.subplots()
    pie_chart.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
    plt.title('Frequency of Sedentary Hours')
    pie_chart.axis('equal')
    plt.show()
```

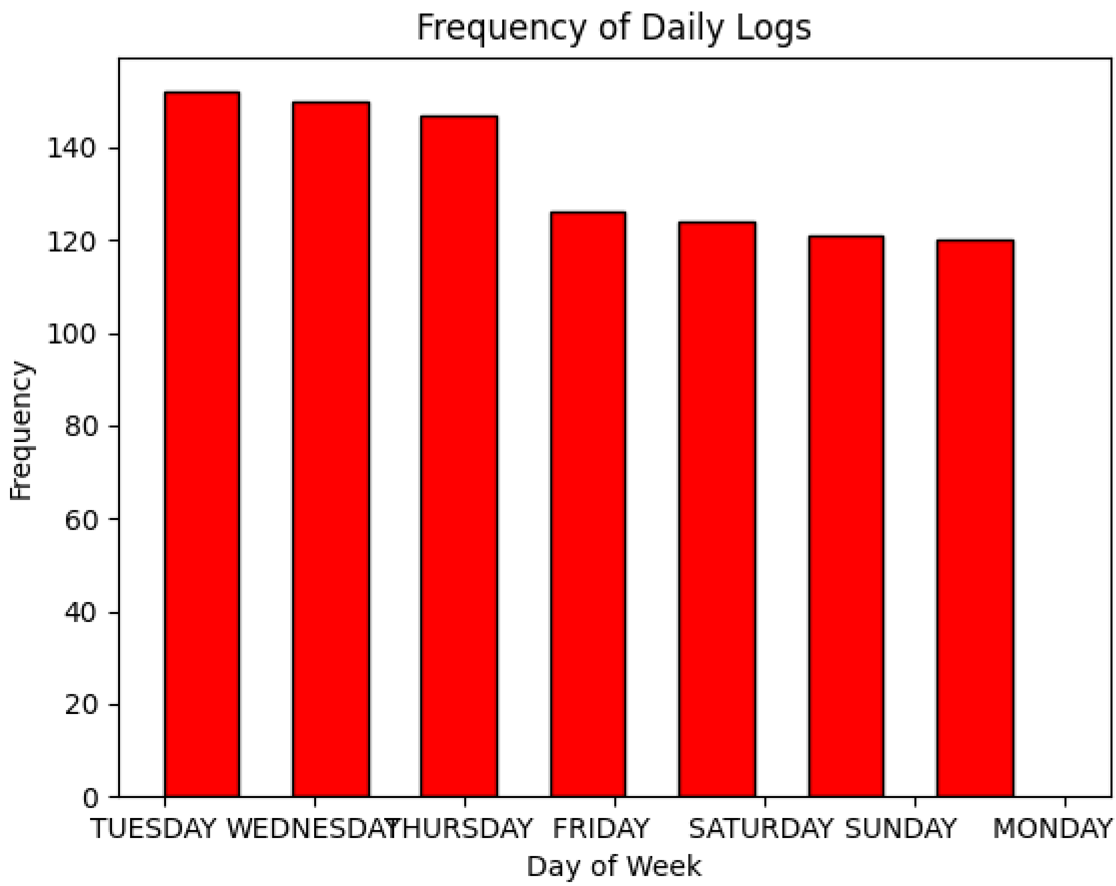
Figure 9 - Frequency of Sedentary Hours



This led me to think that potentially device users were not manually logging events consistently. The histogram below in Figure 10 shows that users are least likely to log events through their device on Friday, Saturday, Sunday, and Monday by about 20%. Potentially users are just less active on those days, or more than likely have set up weekend routines that do not involve consistent logging of their activities. A suggestion for improvement in this aspect would be to push out reminders to users through the app on these days or whatever day the user is least likely to log.

```
def logged_events_visual():
   plt.hist(daily_activity['day_of_week'], bins=7, width=0.5, edgecolor='black',
color='red')
   plt.title('Frequency of Daily Logs')
   plt.xlabel('Day of Week')
   plt.ylabel('Frequency')
   plt.show()
```

Figure 10 - Frequency of Daily Logs



The last thing I looked at for this capstone case study was the cumulative minutes users spent on different intensity levels of activities. I believed that finding a trend here would help the marketing team better understand their consumers. For example, Figure 11 shows that the most common activity is a light one like going for a walk. This could be used in ads, showcasing a person or person(s) wearing the device out for a walk. On the other end of the spectrum the team could also formulate strategies to penetrate the higher level intensity consumers. It is also worth noting that the level of intensity does not necessarily correlate to activity minutes being that fairly active minutes were cumulatively lower than very active minutes.

```
def type_of_activity_visual():
   # make a line graph that has separate lines for each type of activity
   sorted_dates = daily_activity['activity_date'].sort_values()
   plt.plot(sorted_dates, daily_activity['very_active_minutes'].cumsum(),
color='blue')
   plt.plot(sorted_dates, daily_activity['fairly_active_minutes'].cumsum(),
color='red')
   plt.plot(sorted_dates, daily_activity['lightly_active_minutes'].cumsum(),
color='green')
```

```
    plt.title('Cumulative User Minutes Over Time (30 days)')
    plt.xticks([0, 15, 30], ['Day 1', 'Day 15', 'Day 30'])
    plt.xlabel('Time')
    plt.ylabel('Total Minutes')
    plt.legend(['Very Active Minutes', 'Fairly Active Minutes', 'Lightly Active Minutes'])
    plt.show()
```

Figure 11 - Cumulative User Minutes Over Time (30 days)