All data is my personal health data, I downloaded this data from the health app. My data is as follows:

Active Energy

397	15 Eyl 2023	>
49,8	14 Eyl 2023	>
295	13 Eyl 2023	>
254	12 Eyl 2023	>
555	11 Eyl 2023	>
489	10 Eyl 2023	>
95,6	9 Eyl 2023	>
106	8 Eyl 2023	>
76,1	7 Eyl 2023	>
63	6 Eyl 2023	>
149	5 Eyl 2023	>
57,2	4 Eyl 2023	>
46,4	3 Eyl 2023	>
93	2 Eyl 2023	>
70,6	1 Eyl 2023	>

Heart Rate Variability

21 – 59	15 Eyl 2023	>
43 – 49	13 Eyl 2023	>
28 – 48	12 Eyl 2023	>
52 – 94	11 Eyl 2023	>
21 – 36	10 Eyl 2023	>

Heart rate

43 – 142	15 Eyl 2023	>
72 – 115	14 Eyl 2023	>
46 – 109	13 Eyl 2023	>
56 – 111	12 Eyl 2023	>
55 – 115	11 Eyl 2023	>
53 – 119	10 Eyl 2023	>

Daily Steps

7.294	15 Eyl 2023
7.296	14 Eyl 2023
4.189	13 Eyl 2023
4.074	12 Eyl 2023
7.984	11 Eyl 2023
9.519	10 Eyl 2023
3.368	9 Eyl 2023
4.106	8 Eyl 2023
2.356	7 Eyl 2023
2.049	6 Eyl 2023
4.859	5 Eyl 2023
2.060	4 Eyl 2023
1.770	3 Eyl 2023
2.843	2 Eyl 2023
1.968	1 Eyl 2023

Standing Time

7 sa.	15 Eyl 2023 Cum
1 sa.	14 Eyl 2023 Per
7 sa.	13 Eyl 2023 Çar
5 sa.	12 Eyl 2023 Sal
12 sa.	11 Eyl 2023 Pzt
6 sa.	10 Eyl 2023 Paz

Rest Energy

2.148	15 Eyl 2023
1.952	14 Eyl 2023
2.115	13 Eyl 2023
2.085	12 Eyl 2023
2.228	11 Eyl 2023
2.080	10 Eyl 2023

1,8 – 5,4	15 Eyl 2023
2,8 – 5,3	14 Eyl 2023
2,3 – 5,5	13 Eyl 2023
2,3 – 4,8	12 Eyl 2023
2,2 – 4,7	11 Eyl 2023
2,3 – 5,1	10 Eyl 2023
3,2 – 5,8	9 Eyl 2023
2,4 – 5,1	8 Eyl 2023
3,1 – 5,1	7 Eyl 2023
3,2 – 4,5	6 Eyl 2023
2,8 – 5,4	5 Eyl 2023
3 – 4,7	4 Eyl 2023
2,8 – 4,2	3 Eyl 2023
3,1 – 4,8	2 Eyl 2023
3,3 – 4,6	1 Eyl 2023

Walk Speed

5,3 km	15 Eyl 2023
5,3 km	14 Eyl 2023
3 km	13 Eyl 2023
2,9 km	12 Eyl 2023
5,8 km	11 Eyl 2023
6,9 km	10 Eyl 2023
2,3 km	9 Eyl 2023
2,9 km	8 Eyl 2023
1,7 km	7 Eyl 2023
1,4 km	6 Eyl 2023
3,4 km	5 Eyl 2023
1,5 km	4 Eyl 2023
1,3 km	3 Eyl 2023
2,1 km	2 Eyl 2023
1,4 km	1 Eyl 2023

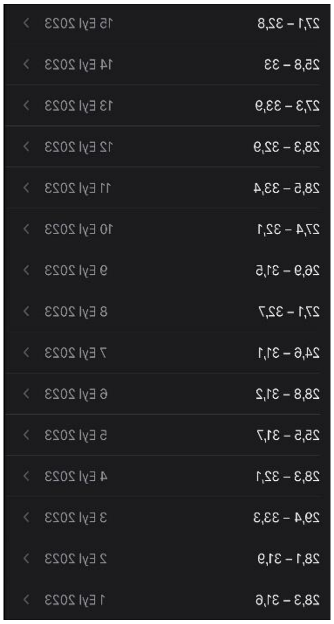
Distance Walk and Run

15 Eyl 2023	6
14 Eyl 2023	15
13 Eyl 2023	4
12 Eyl 2023	5
11 Eyl 2023	4
10 Eyl 2023	11
9 Eyl 2023	4
8 Eyl 2023	4
7 Eyl 2023	3
6 Eyl 2023	3
5 Eyl 2023	1
4 Eyl 2023	3
3 Eyl 2023	1
2 Eyl 2023	5

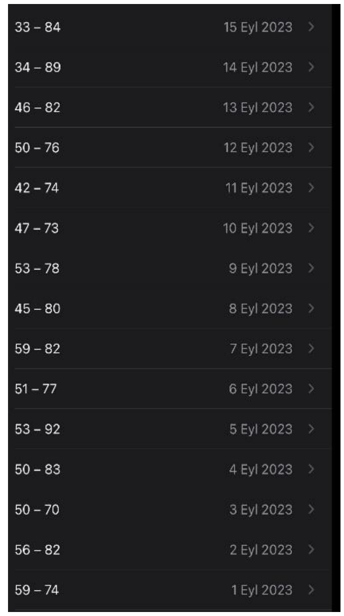
Floor Ascended

0 – 20	15 Eyl 2023
0 – 17	14 Eyl 2023
0 – 2	13 Eyl 2023
0 – 25	12 Eyl 2023
0 – 27	11 Eyl 2023
0 – 4	10 Eyl 2023
0 – 18	9 Eyl 2023
0 – 13	8 Eyl 2023
0 – 6	7 Eyl 2023
0 – 0	6 Eyl 2023
0 – 4	5 Eyl 2023
0 – 5	4 Eyl 2023
0 – 0	3 Eyl 2023
0 – 29	2 Eyl 2023
0 – 1	1 Eyl 2023

Gait Asymmetry



Double Support Time



Walking Step Length

Then, in order to use this data with Python, I compiled the data into Excel and created a table for each variable, allowing me to read this data from Python.

A	B	C	D	E	F	G	H	I	J	K	L	M
Days:	Daily Steps	Standing Time	Rest Energy	Walk Speed	Distance Walk and Run	Floor Ascended	Gait Asymmetry	Double Support Time	Walking step Length:	Heart rate	Heart Rate Variability	Active Energy
1-Sep	1968	NON	NON	3,3-4,6	1,4 km	NON	0-1	28,3 - 31,6	59 - 74	NON	NON	70,6
2-Sep	2843	NON	NON	3,1-4,8	2,1 km	5	0-29	28,1 - 31,9	56 - 82	NON	NON	93
3-Sep	1770	NON	NON	2,8-4,2	1,3 km	1	0-0	29,4 - 33,3	50 - 70	NON	NON	46,4
4-Sep	2060	NON	NON	3-4,7	1,5 km	3	0-5	28,3 - 32,1	50 - 83	NON	NON	57,2
5-Sep	4859	NON	NON	2,8-5,4	3,4 km	1	0-4	25,5 - 31,7	53 - 92	NON	NON	149
6-Sep	2049	NON	NON	3,2-4,5	1,4 km	3	0-0	28,8 - 31,2	51 - 77	NON	NON	63
7-Sep	2356	NON	NON	3,1-5,1	1,7 km	3	0-6	24,6 - 31,1	59 - 82	NON	NON	76,1
8-Sep	4106	NON	NON	2,4-5,1	2,9 km	4	0-13	27,1 - 32,7	45 - 80	NON	NON	106
9-Sep	3368	NON	NON	3,2-5,8	2,3 km	4	0-18	26,9 - 31,5	53 - 78	NON	NON	95,6
10-Sep	9519	6	2080	2,3-5,1	6,9 km	11	0-4	27,4 - 32,1	47 - 73	53 - 119	NON	489
11-Sep	7984	12	2228	2,2-4,7	5,8 km	4	0-27	28,5 - 33,4	42 - 74	55 - 115	21 - 36	555
12-Sep	4074	5	2085	2,3-4,8	2,9 km	5	0-25	28,3 - 33,4	50 - 76	56 - 111	52 - 94	254
13-Sep	4189	7	2115	2,3-5,5	3 km	4	0-2	27,3 - 33,9	46 - 82	46 - 109	28 - 48	295
14-Sep	7296	1	1952	2,8-5,3	5,3 km	15	0-17	25,8 - 33	34 - 89	72 - 115	43 - 49	49,8
15-Sep	7294	7	2148	1,8-5,4	5,3 km	6	0-20	27,1 - 32,8	33 - 85	43 - 142	21 - 59	397

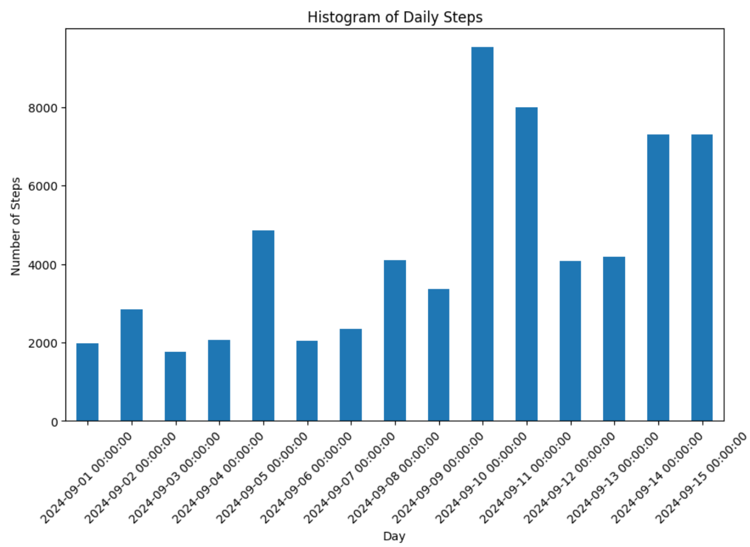
First of all, I imported the data into colab and started reading the variable with the pandas library. First, I started analyzing the data and checked the first 5 lines by making a .head from the data. Then, by .describing the data, I obtained count, mean, unique and similar properties for both numeric and non-numeric data.

```
Descriptive Statistics for Numeric Data:
       Daily Steps
count    15.000000
mean   4382.333333
std    2503.654976
min    1770.000000
25%    2208.000000
50%    4074.000000
75%    6076.500000
max    9519.000000

Descriptive Statistics for Non-Numeric Data:
       Standing Time Rest Energy Walk Speed Distance Walk and Run  \
count             15          15         15                    15
unique             6           7         15                    12
top              NON         NON    3,3-4,6                1,4 km
freq               9           9          1                     2

        Floor Ascended Gait Asymmetry Double Support Time  \
count               15             15                  15
unique               8             13                  15
top                  4            0-0        28,3 - 31,6
freq                 4              2                   1

       Walking step Length: Heart rate Heart Rate Variability Active Energy
count                    15         15                     15            15
unique                   15          7                      6            15
top                 59 - 74        NON                    NON          70,6
freq                      1          9                     10             1
```

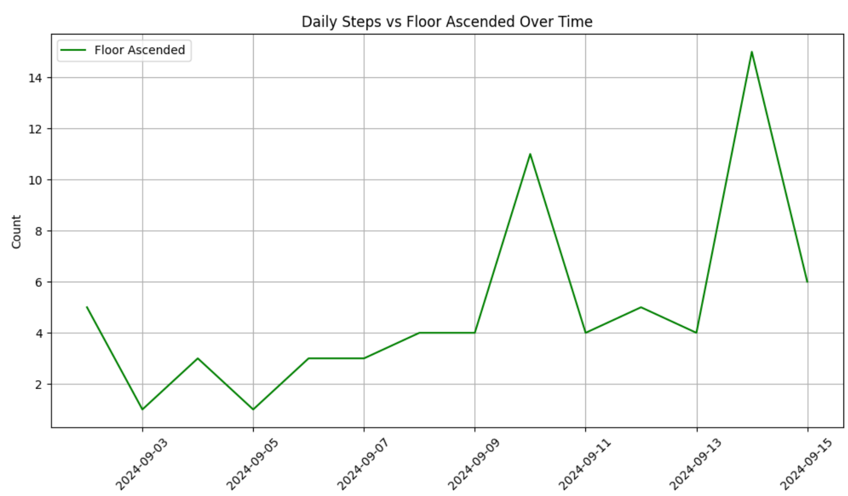
Later, since I wanted to see HealthData.xlsx in a tabular form on the code, I tabulated the data using tabulate.

Days:	Daily Steps	Standing Time	Rest Energy	Walk Speed	Distance Walk and Run	Floor Ascended	Gait Asymmetry	Double Support Time	Walking step Length:	Heart rate
2024-09-01 00:00:00	1968	NON	NON	3,3-4,6	1,4 km	NON	0-1	28,3 - 31,6	59 - 74	NON
2024-09-02 00:00:00	2843	NON	NON	3,1-4,8	2,1 km	5	0-29	28,1 - 31,9	56 - 82	NON
2024-09-03 00:00:00	1770	NON	NON	2,8-4,2	1,3 km	1	0-0	29,4 - 33,3	50 - 70	NON
2024-09-04 00:00:00	2060	NON	NON	3-4,7	1,5 km	3	0-5	28,3 - 32,1	50 - 83	NON
2024-09-05 00:00:00	4859	NON	NON	2,8-5,4	3,4 km	1	0-4	25,5 - 31,7	53 - 92	NON
2024-09-06 00:00:00	2049	NON	NON	3,2-4,5	1,4 km	3	0-0	28,8 - 31,2	51 - 77	NON
2024-09-07 00:00:00	2356	NON	NON	3,1-5,1	1,7 km	3	0-6	24,6 - 31,1	59 - 82	NON
2024-09-08 00:00:00	4106	NON	NON	2,4-5,1	2,9 km	4	0-13	27,1 - 32,7	45 - 80	NON
2024-09-09 00:00:00	3368	NON	NON	3,2-5,8	2,3 km	4	0-18	26,9 - 31,5	53 - 78	NON
2024-09-10 00:00:00	9519	6	2080	2,3-5,1	6,9 km	11	0-4	27,4 - 32,1	47 - 73	53 - 119
2024-09-11 00:00:00	7984	12	2228	2,2-4,7	5,8 km	4	0-27	28,5 - 33,4	42 - 74	55 - 115
2024-09-12 00:00:00	4074	5	2085	2,3-4,8	2,9 km	5	0-25	28,3 - 33,4	50 - 76	56 - 111
2024-09-13 00:00:00	4189	7	2115	2,3-5,5	3 km	4	0-2	27,3 - 33,9	46 - 82	46 - 109
2024-09-14 00:00:00	7296	1	1952	2,8-5,3	5,3 km	15	0-17	25,8 - 33	34 - 89	72 - 115
2024-09-15 00:00:00	7294	7	2148	1,8-5,4	5,3 km	6	0-20	27,1 - 32,8	33 - 85	43 - 142

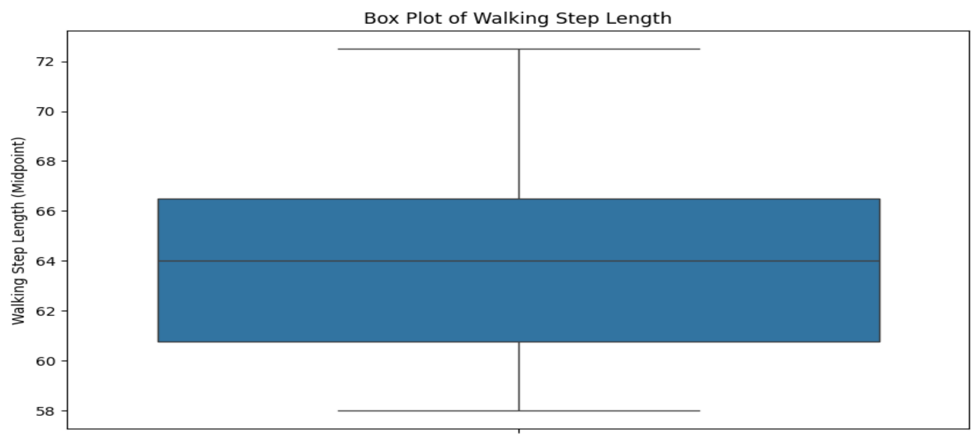
Next, we used Python's matplotlib module to generate a bar chart depiction of a time-series dataset. To prepare the data for time-series analysis, a particular column must first be converted to a datetime format. The data is then arranged chronologically by setting this column as DataFrame index. A bar chart is produced once DataFrame is configured in this way. The plot is made with bars that reflect values of a specific column in the dataset, corresponding to daily measurements, and the scale of the figure is specified.



Then, a line chart was prepared for the floor ascended variable, changing according to each day. This made it easier for us to analyze the data in the line chart.

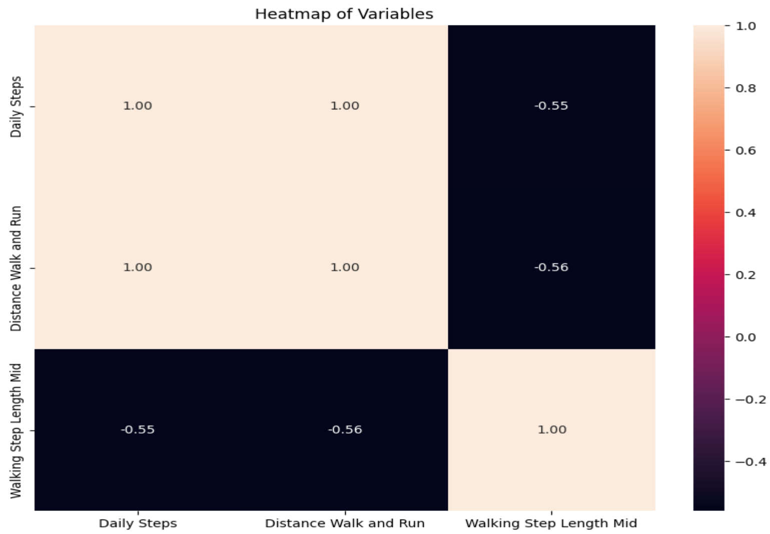


Then I prepared a box plot for Walking Step Length. This box plot shows the maximum and minimum values and is prepared with the average of each day's step length range.



Then I created heatmap thats color intensity denotes degree of connection between several variables in the dataset that in this instance seem to be associated with physical activity, including "Daily Steps," "Distance Walk and Run," and "Walking Step Length Mid."

The correlation between two variables represented by the row label and the column label is represented by each square on the heatmap. A complete positive correlation, or one in which both variables grow proportionately as one increases, is indicated by correlation value of 1.00. As would be predicted, the diagonal line of squares, where the labels for the row and column correspond to the same variable, displays a perfect correlation of 1.00. The dark color of the other off diagonal squares indicates negative correlations which point to inverse link between those variable pairs. For instance, a drop in "Walking Step Length Mid" may be associated with increase in "Daily Steps," and vice versa. A perfect negative correlation is shown by a scale on the right side of heatmap that goes from -1.0 to 1.0, with 0 denoting no correlation and 1.0 denoting a perfect positive correlation. Quickly grasping the pairwise correlations between several variables is made easier with the aid of this visualization, which is useful for a variety of tasks in data analysis, including recognizing underlying patterns in the data and feature selection in machine learning.



In order to predict target variable from the characteristics, a linear regression model is created and fitted to the training set. The model is used to generate predictions on the test set after training. The Mean Squared Error (MSE) metric, that calculates average of the squares of the errors between the anticipated and actual values, is used to quantify the accuracy of these predictions. A numerical representation of the model's predictive ability is given by the resulting MSE, where lower values indicate more accurate predictions.

```
#linear regression

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.impute import SimpleImputer

file_path = 'healthdata.xlsx'
data = pd.read_excel(file_path)

data['Active Energy'] = pd.to_numeric(data['Active Energy'], errors='coerce')
data.dropna(subset=['Active Energy'], inplace=True)

data['Daily Steps'] = pd.to_numeric(data['Daily Steps'], errors='coerce')
data['Distance Walk and Run'] = data['Distance Walk and Run'].str.replace(' km', '').str.replace(',', '.').astype(float)
data['Floor Ascended'] = pd.to_numeric(data['Floor Ascended'], errors='coerce')
data['Gait Asymmetry'] = pd.to_numeric(data['Gait Asymmetry'], errors='coerce')
```

```
[23]
features = data[['Daily Steps', 'Distance Walk and Run', 'Floor Ascended', 'Gait Asymmetry']]

imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(features)

X_train, X_test, y_train, y_test = train_test_split(X_imputed, data['Active Energy'], test_size=0.2, random_state=42)
```

```
[24]
model = LinearRegression()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 15064.411639435984
```

I then used RandomForestRegressor to reduce the accuracy rate and MSE value, thus obtaining a lower MSE value.

```
#RandomForestRegressor

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.impute import SimpleImputer

file_path = 'healthdata.xlsx'
data = pd.read_excel(file_path)

data['Active Energy'] = pd.to_numeric(data['Active Energy'], errors='coerce')
data.dropna(subset=['Active Energy'], inplace=True)

data['Daily Steps'] = pd.to_numeric(data['Daily Steps'], errors='coerce')
data['Distance Walk and Run'] = data['Distance Walk and Run'].str.replace(' km', '').str.replace(',', '.').astype(float)
data['Floor Ascended'] = pd.to_numeric(data['Floor Ascended'], errors='coerce')
data['Gait Asymmetry'] = pd.to_numeric(data['Gait Asymmetry'], errors='coerce')

features = data[['Daily Steps', 'Distance Walk and Run', 'Floor Ascended', 'Gait Asymmetry']]

imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(features)

X_train, X_test, y_train, y_test = train_test_split(X_imputed, data['Active Energy'], test_size=0.2, random_state=42)

model = RandomForestRegressor(random_state=42)
model.fit(X_train, y_train)

predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 11715.8714
```