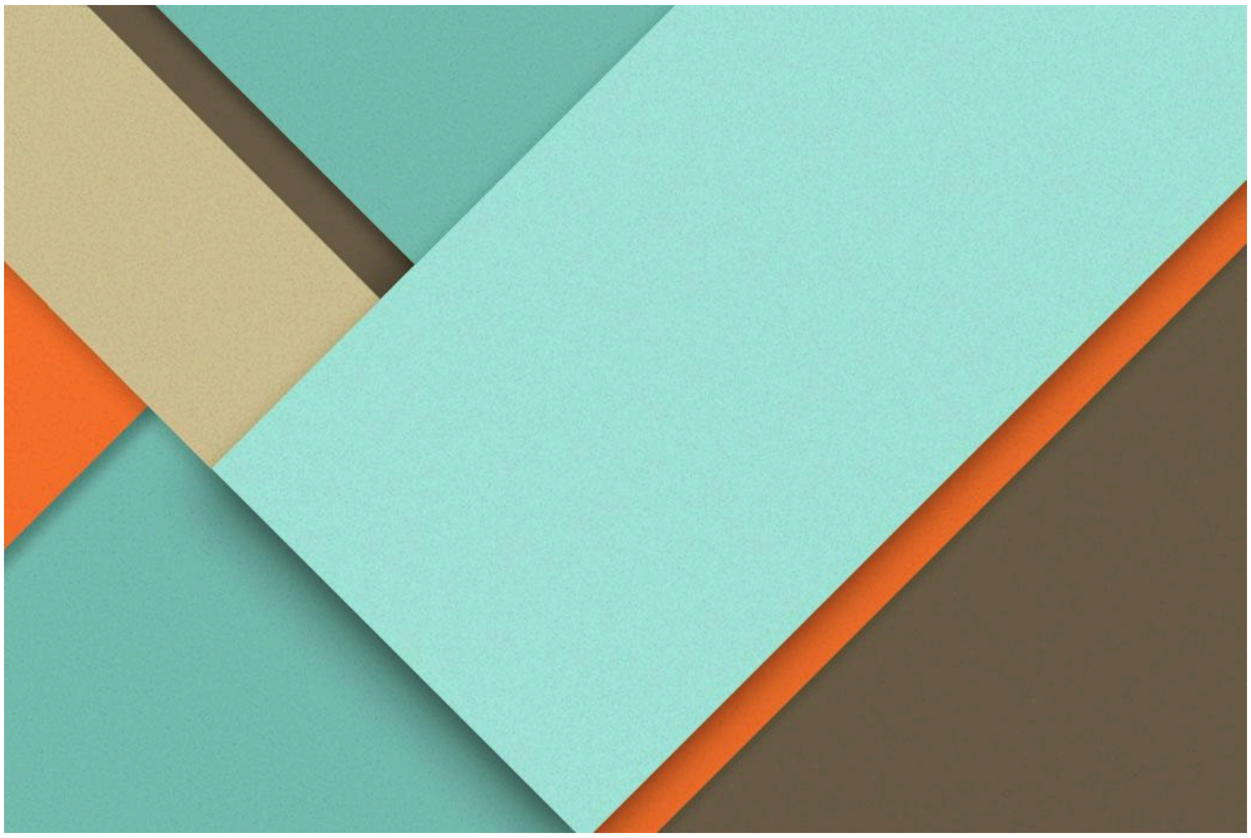


Fitwell Analytics

25.11.2024

Leveraging data from wearable fitness devices to provide actionable insights into individual fitness and wellness.

VALARMATHI GANESSIN
ENVISION VIRTUE
Online Data Analyst Internship
November 2024 Project

FitWell Analytics: Comprehensive Report

Table of Contents

1.Introduction

In today's fast-paced world, maintaining health and wellness is more critical than ever. The advent of wearable fitness technology has revolutionized how individuals monitor and manage their fitness journeys, providing a wealth of data at their fingertips. FitWell Analytics is a project designed to harness this data, transforming it into actionable insights that empower users to make informed decisions about their health and fitness.

FitWell Analytics leverages data from wearable fitness devices to conduct a comprehensive analysis of individual fitness and wellness metrics. By employing various analytical techniques, including linear regression, correlation analysis, distribution analysis, and drillthrough analysis, this project aims to provide a holistic view of user activity, highlight key relationships between different fitness metrics, and evaluate the performance of fitness tracking devices.

2.Project Goals and Objective

Goals

1. The goal is to provide insights into factors that may influence heart rate, energy expenditure, and activity levels, ultimately producing recommendations for optimizing user health and fitness.
2. To provide information about the target variable (heart_rate) depends on the features(steps) by Linear regression.

Objectives

1. **Predictive Modeling:**
    - Develop and refine linear regression models to predict heart rate based on various fitness metrics.
2. **Correlation Analysis:**
    - Identify and understand the relationships and dependencies between different fitness metrics.
3. **Distribution Analysis:**
    - Analyze and visualize the distribution of various fitness metrics to identify patterns and outliers.
4. **Drillthrough Analysis:**
    - Perform detailed activity and member-specific analysis to provide personalized fitness insights and recommendations.
5. **Device Performance Evaluation:**

  - Assess the accuracy and reliability of different fitness tracking devices to ensure high-quality data capture.

6. **Comprehensive Reporting:**
   - Generate detailed reports and visualizations that communicate key findings and actionable insights.

Purpose

The ultimate purpose of FitWell Analytics is to empower users with data-driven insights into their fitness and wellness journeys. By providing accurate predictions, identifying significant correlations, and evaluating device performance, this project aims to improve individual health outcomes and enhance the overall user experience with wearable fitness technology.

3.Data Description

Data Source: Wearable fitness devices (apple watch, fitbit).

- **Data Format:** smartwatch CSV files

Attributes used

- **Personal Information:** Age, gender, height, weight.
- **Activity Metrics:** Steps, Calories, distance, entropy_heart, entropy_steps, resting_heart, corr_heart_steps, norm_heart, intensity_karvonen, sd_norm_heart, steps_times_distance.
- **Target Variable:** Heart Rate.

Dependencies used

- **pandas==1.4.2**
- **numpy==1.22.3**
- **matplotlib==3.5.1**
- **scikit-learn==1.0.2**
- **seaborn==0.11.2**
- **missingno==0.4.2**

Environment

- **Python 3.8**
- **Power BI Desktop**
- **MS Excel**
- **PyCharm**

4.Data Cleaning By Pandas Profiling

1.Essentials: (Data type,unique values and missing values by python script).

Essentials Pandas Profiling.py

```
#Python Script for finding the missing values
import pandas as pd

# Load the CSV file into a DataFrame
file_path = "smartwatch.csv"
df = pd.read_csv(file_path)

# Find the data types of each column
data_types = df.dtypes
print("Data Types:\n", data_types)

# Find the number of unique values in each column
unique_values = df.nunique()
print("\nUnique Values:\n", unique_values)

# Find the number of missing values in each column
missing_values = df.isnull().sum()
print("\nMissing Values:\n", missing_values)
```

OUTPUT

Data Types:

```
Unnamed: 0                 int64
X1                         int64
age                        int64
gender                     int64
height                   float64
weight                   float64
```

```
steps                   float64
hear_rate               float64
calories                float64
distance                float64
entropy_heart           float64
entropy_setps           float64
resting_heart           float64
corr_heart_steps        float64
norm_heart              float64
intensity_karvonen      float64
sd_norm_heart           float64
steps_times_distance    float64
device                   object
activity                 object
dtype: object
```

Unique Values:

```
Unnamed: 0              6264
X1                      3656
age                       24
gender                     2
height                    28
weight                    43
steps                   3919
hear_rate               4514
calories                2136
distance                4863
entropy_heart             56
entropy_setps             60
resting_heart             83
corr_heart_steps        2925
norm_heart              5033
intensity_karvonen      5841
sd_norm_heart           3435
steps_times_distance    4939
device                     2
activity                   6
dtype: int64
```

Missing Values:

```
Unnamed: 0              0
```

```
X1                      0
age                     0
gender                  0
height                  0
weight                  0
steps                   0
hear_rate               0
calories                0
distance                0
entropy_heart           0
entropy_setps           0
resting_heart           0
corr_heart_steps        0
norm_heart              0
intensity_karvonen      0
sd_norm_heart           0
steps_times_distance    0
device                  0
activity                0
dtype: int64
```

Sum of missing values in each column:

```
Unnamed: 0              0
X1                      0
age                     0
gender                  0
height                  0
weight                  0
steps                   0
hear_rate               0
calories                0
distance                0
entropy_heart           0
entropy_setps           0
resting_heart           0
corr_heart_steps        0
norm_heart              0
intensity_karvonen      0
sd_norm_heart           0
steps_times_distance    0
device                  0
activity                0

dtype: int64

Process finished with exit code 0
```

2.Quantile statistics (minimum value, Q1, median, Q3, maximum, range, interquartile range)

Quantile_Analysis.py

```
#Quantile Analysis
import pandas as pd

# Load the CSV file into a DataFrame
file_path = "smartwatch.csv"
df = pd.read_csv(file_path)

# Function to calculate and display quantile statistics for all
columns
def calculate_quantile_statistics(df):
    statistics = {}


    for column in df.columns:
        if pd.api.types.is_numeric_dtype(df[column]):
            min_value = df[column].min()
            Q1 = df[column].quantile(0.25)
            median = df[column].median()
            Q3 = df[column].quantile(0.75)
            max_value = df[column].max()
            range_value = max_value - min_value
            IQR = Q3 - Q1

            statistics[column] = {
                'Minimum Value': min_value,
                'Q1 (25th percentile)': Q1,
                'Median (50th percentile)': median,
                'Q3 (75th percentile)': Q3,
                   'Maximum Value': max_value,
                'Range': range_value,
                'Interquartile Range (IQR)': IQR
            }

    return statistics

# Calculate and display the quantile statistics for all numeric
columns
quantile_statistics = calculate_quantile_statistics(df)
for column, stats in quantile_statistics.items():
    print(f"\nColumn: {column}")
    for stat_name, value in stats.items():
        print(f"{stat_name}: {value}")

# Optionally, you can save the statistics to a file
output_path = "Inter_Quantile_Output.csv"
pd.DataFrame(quantile_statistics).T.to_csv(output_path)
```

OUTPUT

```
Column: Unnamed: 0
Minimum Value: 1
Q1 (25th percentile): 1566.75
Median (50th percentile): 3132.5
Q3 (75th percentile): 4698.25
Maximum Value: 6264
Range: 6263
Interquartile Range (IQR): 3131.5

Column: X1
Minimum Value: 1
Q1 (25th percentile): 789.75
Median (50th percentile): 1720.0
Q3 (75th percentile): 2759.25
Maximum Value: 3670
Range: 3669
Interquartile Range (IQR): 1969.5

Column: age
Minimum Value: 18
Q1 (25th percentile): 23.0
Median (50th percentile): 28.0
Q3 (75th percentile): 33.0
Maximum Value: 56
Range: 38
Interquartile Range (IQR): 10.0

Column: gender
Minimum Value: 0
Q1 (25th percentile): 0.0
Median (50th percentile): 0.0
Q3 (75th percentile): 1.0
Maximum Value: 1
Range: 1
Interquartile Range (IQR): 1.0

Column: height
Minimum Value: 143.0
Q1 (25th percentile): 160.0
```

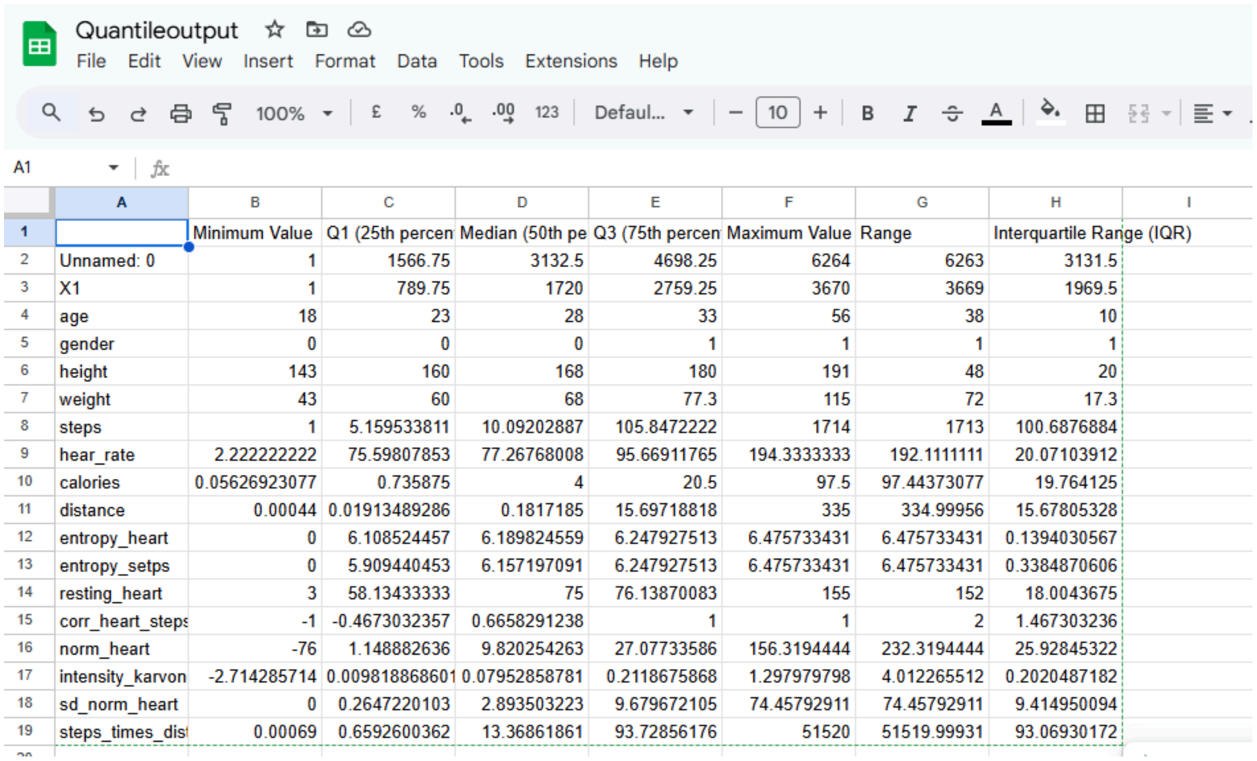
```
Median (50th percentile): 168.0
Q3 (75th percentile): 180.0
Maximum Value: 191.0
Range: 48.0
Interquartile Range (IQR): 20.0

Column: weight
Minimum Value: 43.0
Q1 (25th percentile): 60.0
Median (50th percentile): 68.0
Q3 (75th percentile): 77.3
Maximum Value: 115.0
Range: 72.0
Interquartile Range (IQR): 17.299999999999997

Column: steps
Minimum Value: 1.0
Q1 (25th percentile): 5.159533811053455
Median (50th percentile): 10.09202886896385
Q3 (75th percentile): 105.847222222222
Maximum Value: 1714.0
Range: 1713.0
Interquartile Range (IQR): 100.68768841116855

Column: hear_rate
Minimum Value: 2.22222222222222
Q1 (25th percentile): 75.5980785296575
Median (50th percentile): 77.267680083692
Q3 (75th percentile): 95.66911764705885
Maximum Value: 194.333333333333
Range: 192.11111111111077
Interquartile Range (IQR): 20.071039117401355

Column: calories
Minimum Value: 0.0562692307692308
Q1 (25th percentile): 0.7358750000000001
Median (50th percentile): 4.0
Q3 (75th percentile): 20.5
Maximum Value: 97.5
Range: 97.44373076923077
Interquartile Range (IQR): 19.764125

Column: distance
Minimum Value: 0.00044
Q1 (25th percentile): 0.01913489285714285
Median (50th percentile): 0.1817185
Q3 (75th percentile): 15.6971881759192
```

```
Maximum Value: 335.0
Range: 334.99956
Interquartile Range (IQR): 15.678053283062058

Column: entropy_heart
Minimum Value: 0.0
Q1 (25th percentile): 6.10852445677817
Median (50th percentile): 6.18982455888002
Q3 (75th percentile): 6.24792751344359
Maximum Value: 6.4757334309664
Range: 6.4757334309664
Interquartile Range (IQR): 0.13940305666541963

Column: entropy_setps
Minimum Value: 0.0
Q1 (25th percentile): 5.90944045280115
Median (50th percentile): 6.15719709065712
Q3 (75th percentile): 6.24792751344359
Maximum Value: 6.4757334309664
Range: 6.4757334309664
Interquartile Range (IQR): 0.3384870606424393

Column: resting_heart
Minimum Value: 3.0
Q1 (25th percentile): 58.1343333333333
Median (50th percentile): 75.0
Q3 (75th percentile): 76.1387008333333
Maximum Value: 155.0
Range: 152.0
Interquartile Range (IQR): 18.0043675

Column: corr_heart_steps
Minimum Value: -1.0
Q1 (25th percentile): -0.467303235667249
Median (50th percentile): 0.6658291237999641
Q3 (75th percentile): 1.0
Maximum Value: 1.0
Range: 2.0
Interquartile Range (IQR): 1.467303235667249

Column: norm_heart
Minimum Value: -76.0
Q1 (25th percentile): 1.14888263555389
Median (50th percentile): 9.820254263025
Q3 (75th percentile): 27.077335858585876
Maximum Value: 156.319444444444
Range: 232.319444444444
Interquartile Range (IQR): 25.928453223031987
```

```
Column: intensity_karvonen
Minimum Value: -2.71428571428571
Q1 (25th percentile): 0.009818868600756824
Median (50th percentile): 0.07952858781295641
Q3 (75th percentile): 0.21186758675574752
Maximum Value: 1.2979797979798
Range: 4.01226551226551
Interquartile Range (IQR): 0.2020487181549907

Column: sd_norm_heart
Minimum Value: 0.0
Q1 (25th percentile): 0.264722010296718
Median (50th percentile): 2.89350322251257
Q3 (75th percentile): 9.679672104732528
Maximum Value: 74.4579291138554
Range: 74.4579291138554
Interquartile Range (IQR): 9.414950094435811

Column: steps_times_distance
Minimum Value: 0.00069
Q1 (25th percentile): 0.659260036186983
Median (50th percentile): 13.3686186111111
Q3 (75th percentile): 93.7285617573419
Maximum Value: 51520.0
Range: 51519.99931
Interquartile Range (IQR): 93.06930172115491
```

Process finished with exit code 0

Quantile Output saved as csv file


Quantileoutput

	Minimum Value	Q1 (25th percen	Median (50th pe	Q3 (75th percen	Maximum Value	Range	Interquartile Range (IQR)
Unnamed: 0	1	1566.75	3132.5	4698.25	6264	6263	3131.5
X1	1	789.75	1720	2759.25	3670	3669	1969.5
age	18	23	28	33	56	38	10
gender	0	0	0	1	1	1	1
height	143	160	168	180	191	48	20
weight	43	60	68	77.3	115	72	17.3
steps	1	5.159533811	10.09202887	105.8472222	1714	1713	100.6876884
hear_rate	2.222222222	75.59807853	77.26768008	95.66911765	194.3333333	192.1111111	20.07103912
calories	0.05626923077	0.735875	4	20.5	97.5	97.44373077	19.764125
distance	0.00044	0.01913489286	0.1817185	15.69718818	335	334.99956	15.67805328
entropy_heart	0	6.108524457	6.189824559	6.247927513	6.475733431	6.475733431	0.1394030567
entropy_setps	0	5.909440453	6.157197091	6.247927513	6.475733431	6.475733431	0.3384870606
resting_heart	3	58.13433333	75	76.13870083	155	152	18.0043675
corr_heart_steps	-1	-0.4673032357	0.6658291238	1	1	2	1.467303236
norm_heart	-76	1.148882636	9.820254263	27.07733586	156.3194444	232.3194444	25.92845322
intensity_karvon	-2.714285714	0.009818868601	0.07952858781	0.2118675868	1.297979798	4.012265512	0.2020487182
sd_norm_heart	0	0.2647220103	2.893503223	9.679672105	74.45792911	74.45792911	9.414950094
steps_times_dist	0.00069	0.6592600362	13.36861861	93.72856176	51520	51519.99931	93.06930172

3.Descriptive statistics (mean, mode, standard deviation, sum, median absolute deviation, coefficient of variation, kurtosis, skewness)

Descriptive_Analysis.py

```
#Python Script for Descriptive Analysis
import pandas as pd
import numpy as np
from scipy.stats import kurtosis, skew

# Load the CSV file into a DataFrame
file_path = "smartwatch.csv"
df = pd.read_csv(file_path)

# Function to calculate descriptive statistics
def calculate_descriptive_statistics(df):
    statistics = {}

    for column in df.columns:
        if pd.api.types.is_numeric_dtype(df[column]):
            mean_value = df[column].mean()
            mode_value = df[column].mode().iloc[0] if not
df[column].mode().empty else np.nan
            std_dev = df[column].std()
```

```
              sum_value = df[column].sum()
            mad = df[column].mad()
            coeff_var = std_dev / mean_value if mean_value != 0 else
np.nan
            kurtosis_value = kurtosis(df[column], nan_policy='omit')
            skewness_value = skew(df[column], nan_policy='omit')

            statistics[column] = {
                'Mean': mean_value,
                'Mode': mode_value,
                'Standard Deviation': std_dev,
                   'Sum': sum_value,
                'Median Absolute Deviation': mad,
                'Coefficient of Variation': coeff_var,
                'Kurtosis': kurtosis_value,
                'Skewness': skewness_value
               }

     return statistics

# Calculate and display the descriptive statistics for all numeric
columns
descriptive_statistics = calculate_descriptive_statistics(df)
for column, stats in descriptive_statistics.items():
    print(f"\nColumn: {column}")
    for stat_name, value in stats.items():
        print(f"{stat_name}: {value}")

# Optionally, you can save the statistics to a file
output_path = "Descriptive_Analysis_Output.csv"
pd.DataFrame(descriptive_statistics).T.to_csv(output_path)
```

OUTPUT

```
Column: Unnamed: 0
Mean: 3132.5
Mode: 1
Standard Deviation: 1808.405374909066
Sum: 19621980
Median Absolute Deviation: 1566.0
Coefficient of Variation: 0.5773041899151049
Kurtosis: -1.2000000611656725
Skewness: 0.0
```

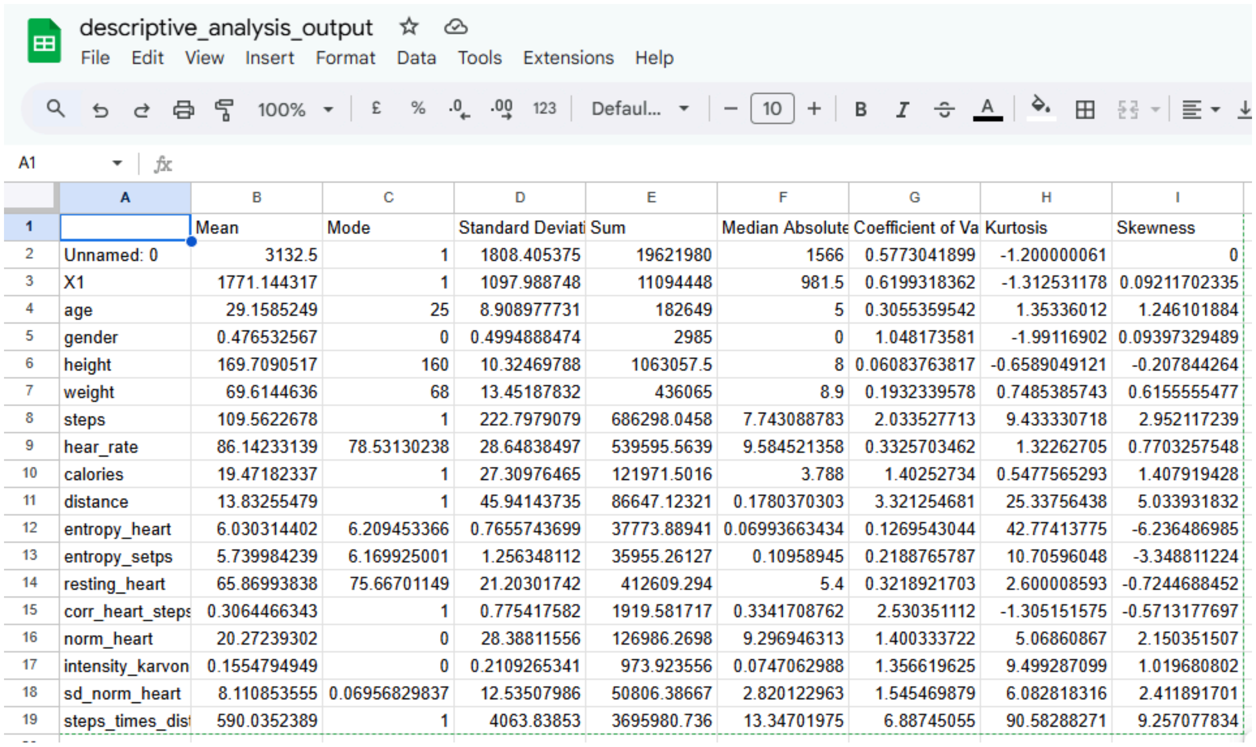
```
Column: X1
Mean: 1771.1443167305235
Mode: 1
Standard Deviation: 1097.9887484120309
Sum: 11094448
Median Absolute Deviation: 981.5
Coefficient of Variation: 0.6199318361808502
Kurtosis: -1.3125311777222022
Skewness: 0.09211702335435708

Column: age
Mean: 29.15852490421456
Mode: 25
Standard Deviation: 8.90897773058581
Sum: 182649
Median Absolute Deviation: 5.0
Coefficient of Variation: 0.3055359542312825
Kurtosis: 1.3533601195754175
Skewness: 1.2461018840389573

Column: gender
Mean: 0.47653256704980845
Mode: 0
Standard Deviation: 0.49948884735403143
Sum: 2985
Median Absolute Deviation: 0.0
Coefficient of Variation: 1.0481735811811232
Kurtosis: -1.991169019847637
Skewness: 0.09397329488936168

Column: height
Mean: 169.70905172413794
Mode: 160.0
Standard Deviation: 10.324697882220653
Sum: 1063057.5
Median Absolute Deviation: 8.0
Coefficient of Variation: 0.06083763816560268
Kurtosis: -0.6589049121340653
Skewness: -0.20784426396166716

Column: weight
Mean: 69.61446360153256
Mode: 68.0
Standard Deviation: 13.451878323577452
Sum: 436065.0
Median Absolute Deviation: 8.899999999999999
Coefficient of Variation: 0.1932339578248407
```

```
Kurtosis: 0.7485385743409219
Skewness: 0.6155555477268119

Column: steps
Mean: 109.56226784613551
Mode: 1.0
Standard Deviation: 222.79790794473982
Sum: 686298.0457881929
Median Absolute Deviation: 7.743088782562449
Coefficient of Variation: 2.0335277128219684
Kurtosis: 9.433330718265305
Skewness: 2.952117239214441

Column: hear_rate
Mean: 86.1423313938081
Mode: 78.5313023809524
Standard Deviation: 28.64838497476638
Sum: 539595.563850814
Median Absolute Deviation: 9.5845213584373
Coefficient of Variation: 0.33257034620757453
Kurtosis: 1.322627050485952
Skewness: 0.7703257547814935

Column: calories
Mean: 19.471823374275477
Mode: 1.0
Standard Deviation: 27.309764646939705
Sum: 121971.5016164616
Median Absolute Deviation: 3.788
Coefficient of Variation: 1.402527340249966
Kurtosis: 0.5477565292742961
Skewness: 1.4079194282483674

Column: distance
Mean: 13.832554790098404
Mode: 1.0
Standard Deviation: 45.94143734674101
Sum: 86647.12320517641
Median Absolute Deviation: 0.17803703034547155
Coefficient of Variation: 3.321254680995497
Kurtosis: 25.337564379533962
Skewness: 5.0339318316040185

Column: entropy_heart
Mean: 6.0303144015657475
Mode: 6.20945336562895
Standard Deviation: 0.765574369867811
```

```
Sum: 37773.88941140784
Median Absolute Deviation: 0.06993663434119934
Coefficient of Variation: 0.12695430435086977
Kurtosis: 42.774137746033595
Skewness: -6.236486984766301

Column: entropy_setps
Mean: 5.739984238633239
Mode: 6.16992500144231
Standard Deviation: 1.2563481120390967
Sum: 35955.26127079861
Median Absolute Deviation: 0.10958945003778009
Coefficient of Variation: 0.21887657871657304
Kurtosis: 10.705960481777435
Skewness: -3.3488112240573713

Column: resting_heart
Mean: 65.86993837853097
Mode: 75.6670114877907
Standard Deviation: 21.20301741943007
Sum: 412609.294003118
Median Absolute Deviation: 5.400000000000006
Coefficient of Variation: 0.32189217025806083
Kurtosis: 2.6000085926779795
Skewness: -0.7244688452394317

Column: corr_heart_steps
Mean: 0.30644663433497066
Mode: 1.0
Standard Deviation: 0.7754175820277911
Sum: 1919.581717474256
Median Absolute Deviation: 0.33417087620003594
Coefficient of Variation: 2.5303511122272524
Kurtosis: -1.3051515748252744
Skewness: -0.5713177697358383

Column: norm_heart
Mean: 20.27239301527714
Mode: 0.0
Standard Deviation: 28.388115555635164
Sum: 126986.269847696
Median Absolute Deviation: 9.296946313252995
Coefficient of Variation: 1.4003337215415108
Kurtosis: 5.068608670274173
Skewness: 2.1503515073461053

Column: intensity_karvonen
```

```
Mean: 0.1554794948928595
Mode: 0.0
Standard Deviation: 0.21092653407578724
Sum: 973.9235560088719
Median Absolute Deviation: 0.07470629879807486
Coefficient of Variation: 1.3566196251225033
Kurtosis: 9.499287099327718
Skewness: 1.0196808019031867

Column: sd_norm_heart
Mean: 8.11085355471586
Mode: 0.0695682983670221
Standard Deviation: 12.535079859467045
Sum: 50806.386666740145
Median Absolute Deviation: 2.8201229625911903
Coefficient of Variation: 1.545469878713175
Kurtosis: 6.082818315850673
Skewness: 2.411891700816861

Column: steps_times_distance
Mean: 590.0352388755643
Mode: 1.0
Standard Deviation: 4063.838530451166
Sum: 3695980.736316535
Median Absolute Deviation: 13.347019751057275
Coefficient of Variation: 6.887450549895394
Kurtosis: 90.58288271216703
Skewness: 9.25707783393574

Process finished with exit code 0
```

Descriptive Analysis Output saved as a csv file


descriptive_analysis_output
File Edit View Insert Format Data Tools Extensions Help
100%
Defaul...
10
A1

	Mean	Mode	Standard Deviati	Sum	Median Absolute	Coefficient of Va	Kurtosis	Skewness
Unnamed: 0	3132.5	1	1808.405375	19621980	1566	0.5773041899	-1.200000061	0
X1	1771.144317	1	1097.988748	11094448	981.5	0.6199318362	-1.312531178	0.09211702335
age	29.1585249	25	8.908977731	182649	5	0.3055359542	1.35336012	1.246101884
gender	0.476532567	0	0.4994888474	2985	0	1.048173581	-1.99116902	0.09397329489
height	169.7090517	160	10.32469788	1063057.5	8	0.06083763817	-0.6589049121	-0.207844264
weight	69.6144636	68	13.45187832	436065	8.9	0.1932339578	0.7485385743	0.6155555477
steps	109.5622678	1	222.7979079	686298.0458	7.743088783	2.033527713	9.433330718	2.952117239
hear_rate	86.14233139	78.53130238	28.64838497	539595.5639	9.584521358	0.3325703462	1.32262705	0.7703257548
calories	19.47182337	1	27.30976465	121971.5016	3.788	1.40252734	0.5477565293	1.407919428
distance	13.83255479	1	45.94143735	86647.12321	0.1780370303	3.321254681	25.33756438	5.033931832
entropy_heart	6.030314402	6.209453366	0.7655743699	37773.88941	0.06993663434	0.1269543044	42.77413775	-6.236486985
entropy_setps	5.739984239	6.169925001	1.256348112	35955.26127	0.10958945	0.2188765787	10.70596048	-3.348811224
resting_heart	65.86993838	75.66701149	21.20301742	412609.294	5.4	0.3218921703	2.600008593	-0.7244688452
corr_heart_steps	0.3064466343	1	0.775417582	1919.581717	0.3341708762	2.530351112	-1.305151575	-0.5713177697
norm_heart	20.27239302	0	28.38811556	126986.2698	9.296946313	1.400333722	5.06860867	2.150351507
intensity_karvon	0.1554794949	0	0.2109265341	973.923556	0.0747062988	1.356619625	9.499287099	1.019680802
sd_norm_heart	8.110853555	0.06956829837	12.53507986	50806.38667	2.820122963	1.545469879	6.082818316	2.411891701
steps_times_dis	590.0352389	1	4063.83853	3695980.736	13.34701975	6.88745055	90.58288271	9.257077834

4.Most frequent values

Most_Frequent_Values.py

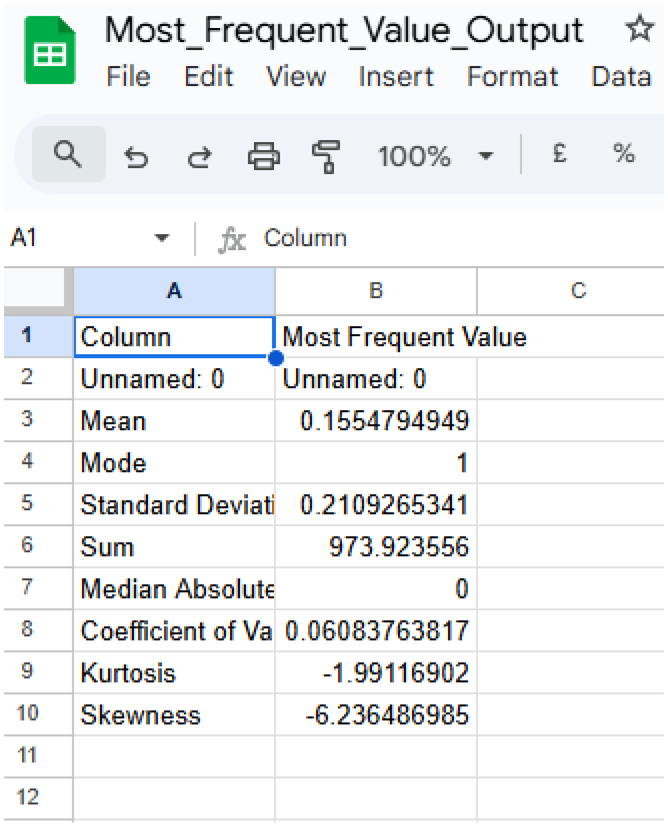
```
#Python Script for finding most frequent values in the given csv file
import pandas as pd

# Load the CSV file into a DataFrame
file_path = "smartwatch.csv"
df = pd.read_csv(file_path)

# Function to find most frequent values
def most_frequent_values(df):
    most_frequent = {}


     for column in df.columns:
         most_frequent_value = df[column].mode().iloc[0] if not
df[column].mode().empty else None
         most_frequent[column] = most_frequent_value

    return most_frequent

# Calculate and display the most frequent values for each column
most_frequent = most_frequent_values(df)
```

```
for column, value in most_frequent.items():
    print(f"Column: {column}, Most Frequent Value: {value}")

# Optionally, you can save the results to a CSV file
output_path = "Most_Frequent_values_Output.csv"
pd.DataFrame(most_frequent.items(), columns=['Column', 'Most Frequent
Value']).to_csv(output_path, index=False)
```

OUTPUT

```
Column: Unnamed: 0, Most Frequent Value: Unnamed: 0
Column: Mean, Most Frequent Value: 0.1554794948928595
Column: Mode, Most Frequent Value: 1.0
Column: Standard Deviation, Most Frequent Value: 0.2109265340757872
Column: Sum, Most Frequent Value: 973.923556008872
Column: Median Absolute Deviation, Most Frequent Value: 0.0
Column: Coefficient of Variation, Most Frequent Value:
0.0608376381656026
Column: Kurtosis, Most Frequent Value: -1.991169019847637
Column: Skewness, Most Frequent Value: -6.236486984766301

Process finished with exit code 0
```

Most Frequent Values saved as a csv file



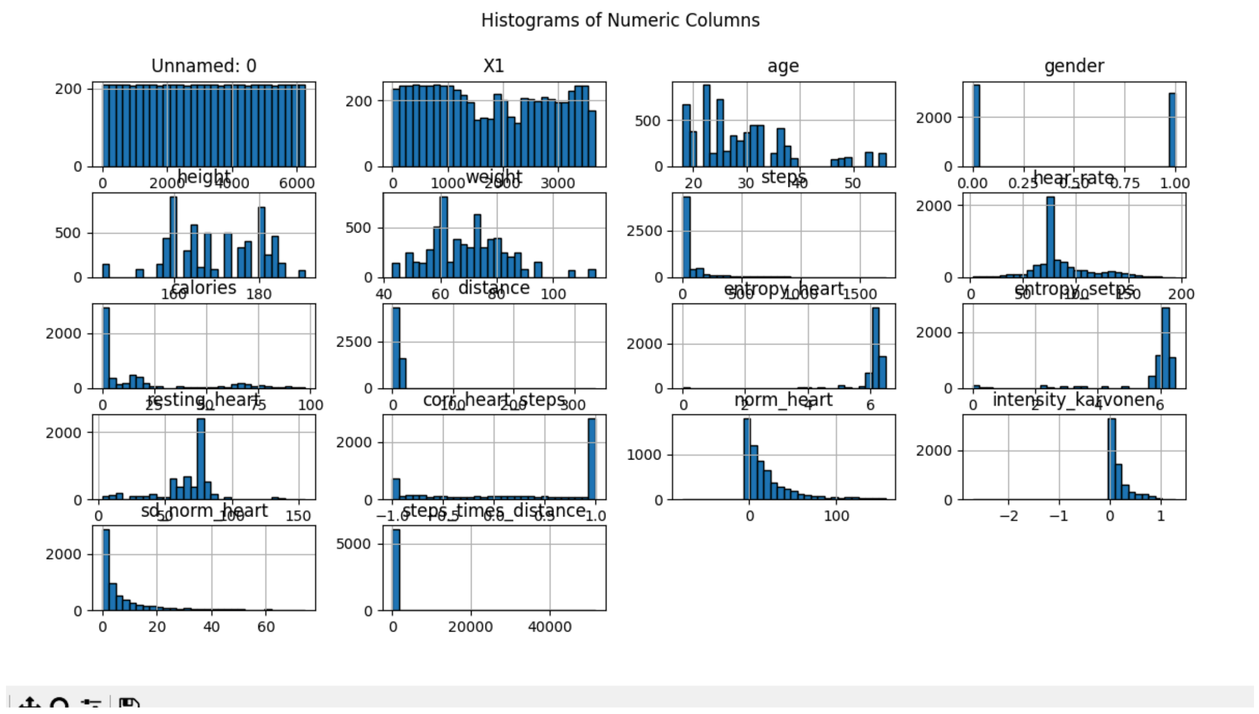
	A	B	C
1	Column	Most Frequent Value	
2	Unnamed: 0	Unnamed: 0	
3	Mean	0.1554794949	
4	Mode	1	
5	Standard Deviati	0.2109265341	
6	Sum	973.923556	
7	Median Absolute	0	
8	Coefficient of Va	0.06083763817	
9	Kurtosis	-1.99116902	
10	Skewness	-6.236486985	
11			
12			

5.Histogram

Histogram.py

```
#Python Script for Histogram Graph
import pandas as pd
import matplotlib.pyplot as plt

# Load the CSV file into a DataFrame
file_path = "smartwatch.csv"
df = pd.read_csv(file_path)

# Plot histograms for all numeric columns
df.hist(figsize=(10, 8), bins=30, edgecolor='black')

# Set overall title for the histograms
plt.suptitle('Histograms of Numeric Columns', fontsize=16)

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

Output



6.Correlations (highlighting of highly correlated variables, Spearman, Pearson and Kendall matrices)

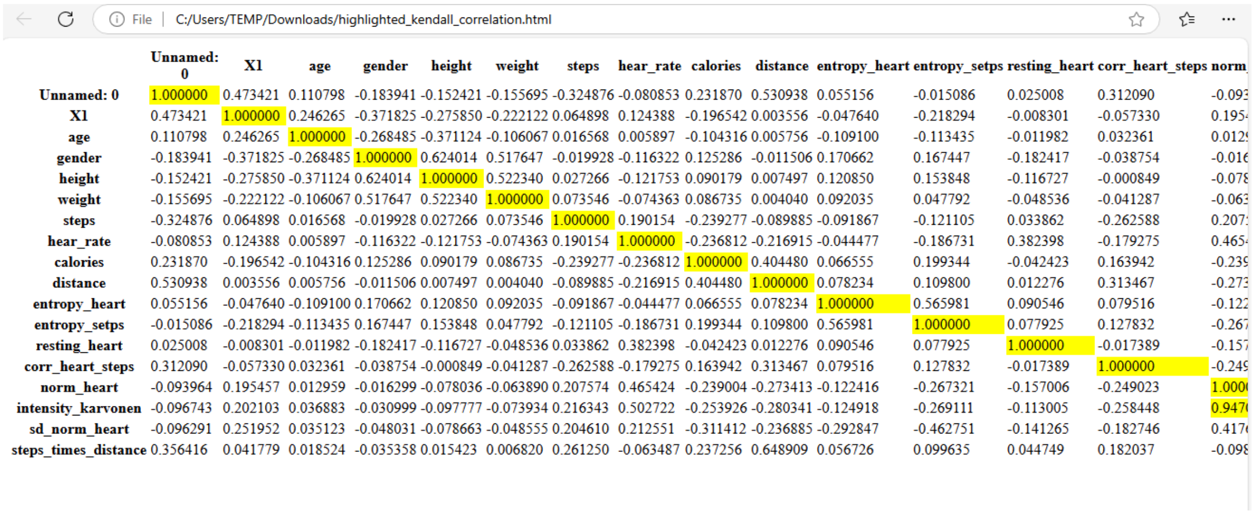
Correlations.py

```
# Import the necessary library
import pandas as pd

# Define the file path to the dataset
file_path = "smartwatch.csv"

# Load the dataset into a pandas DataFrame
df = pd.read_csv(file_path)

# Select only the numeric columns from the DataFrame
numeric_df = df.select_dtypes(include='number')

# Calculate the correlation matrices using different methods
pearson_corr = numeric_df.corr(method='pearson')  # Pearson
correlation
spearman_corr = numeric_df.corr(method='spearman')  # Spearman
correlation
kendall_corr = numeric_df.corr(method='kendall')  # Kendall
correlation
```

```
# Function to highlight highly correlated values in the correlation
matrix
def highlight_highly_correlated(corr_matrix, threshold=0.8):
    return corr_matrix.applymap(lambda x: 'background-color: yellow'
if abs(x) >= threshold else '')

# Apply the highlighting function to the correlation matrices
highlighted_pearson =
pearson_corr.style.apply(highlight_highly_correlated, threshold=0.8,
axis=None)
highlighted_spearman =
spearman_corr.style.apply(highlight_highly_correlated, threshold=0.8,
axis=None)
highlighted_kendall =
kendall_corr.style.apply(highlight_highly_correlated, threshold=0.8,
axis=None)

# Save the correlation matrices to CSV files
pearson_corr.to_csv("pearson_correlation.csv")
spearman_corr.to_csv("spearman_correlation.csv")
kendall_corr.to_csv("kendall_correlation.csv")

# Save the highlighted correlation matrices to HTML files
highlighted_pearson.to_html("highlighted_pearson_correlation.html")
highlighted_spearman.to_html("highlighted_spearman_correlation.html")
highlighted_kendall.to_html("highlighted_kendall_correlation.html")

# Print a success message
print("Correlation matrices and highlighted versions saved
successfully.")
```

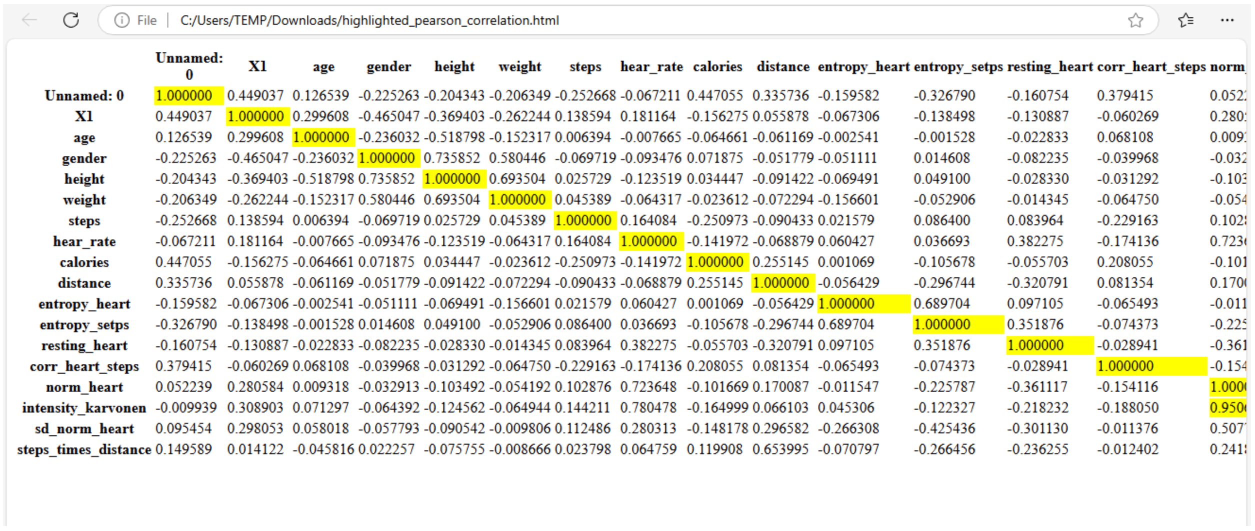
OUTPUT

All the three correlations are saved as csv as well as html files.



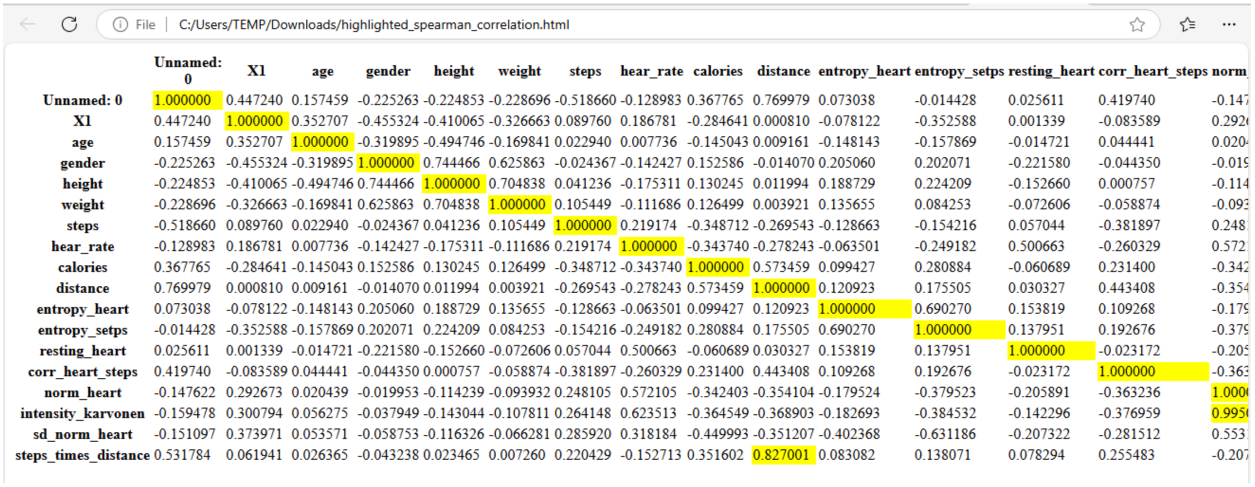
C:/Users/TEMP/Downloads/highlighted_kendall_correlation.html

	Unnamed: 0	X1	age	gender	height	weight	steps	hear_rate	calories	distance	entropy_heart	entropy_setps	resting_heart	corr_heart_steps	norm
Unnamed: 0	1.000000	0.473421	0.110798	-0.183941	-0.152421	-0.155695	-0.324876	-0.080853	0.231870	0.530938	0.055156	-0.015086	0.025008	0.312090	-0.093
X1	0.473421	1.000000	0.246265	-0.371825	-0.275850	-0.222122	0.064898	0.124388	-0.196542	0.003556	-0.047640	-0.218294	-0.008301	-0.057330	0.195
age	0.110798	0.246265	1.000000	-0.268485	-0.371124	-0.106067	0.016568	0.005897	-0.104316	0.005756	-0.109100	-0.113435	-0.011982	0.032361	0.012
gender	-0.183941	-0.371825	-0.268485	1.000000	0.624014	0.517647	-0.019928	-0.116322	0.125286	-0.011506	0.170662	0.167447	-0.182417	-0.038754	-0.016
height	-0.152421	-0.275850	-0.371124	0.624014	1.000000	0.522340	0.027266	-0.121753	0.090179	0.007497	0.120850	0.153848	-0.116727	-0.000849	-0.078
weight	-0.155695	-0.222122	-0.106067	0.517647	0.522340	1.000000	0.073546	-0.074363	0.086735	0.004040	0.092035	0.047792	-0.048536	-0.041287	-0.063
steps	-0.324876	0.064898	0.016568	-0.019928	0.027266	0.073546	1.000000	0.190154	-0.239277	-0.089885	-0.091867	-0.121105	0.033862	-0.262588	0.207
hear_rate	-0.080853	0.124388	0.005897	-0.116322	-0.121753	-0.074363	0.190154	1.000000	-0.236812	-0.216915	-0.044477	-0.186731	0.382398	-0.179275	0.465
calories	0.231870	-0.196542	-0.104316	0.125286	0.090179	0.086735	-0.239277	-0.236812	1.000000	0.404480	0.066555	0.199344	-0.042423	0.163942	-0.239
distance	0.530938	0.003556	0.005756	-0.011506	0.007497	0.004040	-0.089885	-0.216915	0.404480	1.000000	0.078234	0.109800	0.012276	0.313467	-0.273
entropy_heart	0.055156	-0.047640	-0.109100	0.170662	0.120850	0.092035	-0.091867	-0.044477	0.066555	0.078234	1.000000	0.565981	0.090546	0.079516	-0.122
entropy_setps	-0.015086	-0.218294	-0.113435	0.167447	0.153848	0.047792	-0.121105	-0.186731	0.199344	0.109800	0.565981	1.000000	0.077925	0.127832	-0.267
resting_heart	0.025008	-0.008301	-0.011982	-0.182417	-0.116727	-0.048536	0.033862	0.382398	-0.042423	0.012276	0.090546	0.077925	1.000000	-0.017389	-0.157
corr_heart_steps	0.312090	-0.057330	0.032361	-0.038754	-0.000849	-0.041287	-0.262588	-0.179275	0.163942	0.313467	0.079516	0.127832	-0.017389	1.000000	-0.249
norm_heart	-0.093964	0.195457	0.012959	-0.016299	-0.078036	-0.063890	0.207574	0.465424	-0.239004	-0.273413	-0.122416	-0.267321	-0.157006	-0.249023	1.000
intensity_karvonen	-0.096743	0.202103	0.036883	-0.030999	-0.097777	-0.073934	0.216343	0.502722	-0.253926	-0.280341	-0.124918	-0.269111	-0.113005	-0.258448	0.947
sd_norm_heart	-0.096291	0.251952	0.035123	-0.048031	-0.078663	-0.048555	0.204610	0.212551	-0.311412	-0.236885	-0.292847	-0.462751	-0.141265	-0.182746	0.417
steps_times_distance	0.356416	0.041779	0.018524	-0.035358	0.015423	0.006820	0.261250	-0.063487	0.237256	0.648909	0.056726	0.099635	0.044749	0.182037	-0.098



C:/Users/TEMP/Downloads/highlighted_pearson_correlation.html

	Unnamed: 0	X1	age	gender	height	weight	steps	hear_rate	calories	distance	entropy_heart	entropy_setps	resting_heart	corr_heart_steps	norm
Unnamed: 0	1.000000	0.449037	0.126539	-0.225263	-0.204343	-0.206349	-0.252668	-0.067211	0.447055	0.335736	-0.159582	-0.326790	-0.160754	0.379415	0.052
X1	0.449037	1.000000	0.299608	-0.465047	-0.369403	-0.262244	0.138594	0.181164	-0.156275	0.055878	-0.067306	-0.138498	-0.130887	-0.060269	0.280
age	0.126539	0.299608	1.000000	-0.236032	-0.518798	-0.152317	0.006394	-0.007665	-0.064661	-0.061169	-0.002541	-0.001528	-0.022833	0.068108	0.009
gender	-0.225263	-0.465047	-0.236032	1.000000	0.735852	0.580446	-0.069719	-0.093476	0.071875	-0.051779	-0.051111	0.014608	-0.082235	-0.039968	-0.032
height	-0.204343	-0.369403	-0.518798	0.735852	1.000000	0.693504	0.025729	-0.123519	0.034447	-0.091422	-0.069491	0.049100	-0.028330	-0.031292	-0.103
weight	-0.206349	-0.262244	-0.152317	0.580446	0.693504	1.000000	0.045389	-0.064317	-0.023612	-0.072294	-0.156601	-0.052906	-0.014345	-0.064750	-0.054
steps	-0.252668	0.138594	0.006394	-0.069719	0.025729	0.045389	1.000000	0.164084	-0.250973	-0.090433	0.021579	0.086400	0.083964	-0.229163	0.102
hear_rate	-0.067211	0.181164	-0.007665	-0.093476	-0.123519	-0.064317	0.164084	1.000000	-0.141972	-0.068879	0.060427	0.036693	0.382275	-0.174136	0.723
calories	0.447055	-0.156275	-0.064661	0.071875	0.034447	-0.023612	-0.250973	-0.141972	1.000000	0.255145	0.001069	-0.105678	-0.055703	0.208055	-0.101
distance	0.335736	0.055878	-0.061169	-0.051779	-0.091422	-0.072294	-0.090433	-0.068879	0.255145	1.000000	-0.056429	-0.296744	-0.320791	0.081354	0.170
entropy_heart	-0.159582	-0.067306	-0.002541	-0.051111	-0.069491	-0.156601	0.021579	0.060427	0.001069	-0.056429	1.000000	0.689704	0.097105	-0.065493	-0.011
entropy_setps	-0.326790	-0.138498	-0.001528	0.014608	0.049100	-0.052906	0.086400	0.036693	-0.105678	-0.296744	0.689704	1.000000	0.351876	-0.074373	-0.225
resting_heart	-0.160754	-0.130887	-0.022833	-0.082235	-0.028330	-0.014345	0.083964	0.382275	-0.055703	-0.320791	0.097105	0.351876	1.000000	-0.028941	-0.361
corr_heart_steps	0.379415	-0.060269	0.068108	-0.039968	-0.031292	-0.064750	-0.229163	-0.174136	0.208055	0.081354	-0.065493	-0.074373	-0.028941	1.000000	-0.154
norm_heart	0.052239	0.280584	0.009318	-0.032913	-0.103492	-0.054192	0.102876	0.723648	-0.101669	0.170087	-0.011547	-0.225787	-0.361117	-0.154116	1.000
intensity_karvonen	-0.009939	0.308903	0.071297	-0.064392	-0.124562	-0.064944	0.144211	0.780478	-0.164999	0.066103	0.045306	-0.122327	-0.218232	-0.188050	0.950
sd_norm_heart	0.095454	0.298053	0.058018	-0.057793	-0.090542	-0.009806	0.112486	0.280313	-0.148178	0.296582	-0.266308	-0.425436	-0.301130	-0.011376	0.507
steps_times_distance	0.149589	0.014122	-0.045816	0.022257	-0.075755	-0.008666	0.023798	0.064759	0.119908	0.653995	-0.070797	-0.266456	-0.236255	-0.012402	0.241



C:/Users/TEMP/Downloads/highlighted_spearman_correlation.html

	Unnamed: 0	X1	age	gender	height	weight	steps	hear_rate	calories	distance	entropy_heart	entropy_setps	resting_heart	corr_heart_steps	norm
Unnamed: 0	1.000000	0.447240	0.157459	-0.225263	-0.224853	-0.228696	-0.518660	-0.128983	0.367765	0.769979	0.073038	-0.014428	0.025611	0.419740	-0.147
X1	0.447240	1.000000	0.352707	-0.455324	-0.410065	-0.326663	0.089760	0.186781	-0.284641	0.000810	-0.078122	-0.352588	0.001339	-0.083589	0.292
age	0.157459	0.352707	1.000000	-0.319895	-0.494746	-0.169841	0.022940	0.007736	-0.145043	0.009161	-0.148143	-0.157869	-0.014721	0.044441	0.020
gender	-0.225263	-0.455324	-0.319895	1.000000	0.744466	0.625863	-0.024367	-0.142427	0.152586	-0.014070	0.205060	0.202071	-0.221580	-0.044350	-0.019
height	-0.224853	-0.410065	-0.494746	0.744466	1.000000	0.704838	0.041236	-0.175311	0.130245	0.011994	0.188729	0.224209	-0.152660	0.000757	-0.114
weight	-0.228696	-0.326663	-0.169841	0.625863	0.704838	1.000000	0.105449	-0.111686	0.126499	0.003921	0.135655	0.084253	-0.072606	-0.058874	-0.093
steps	-0.518660	0.089760	0.022940	-0.024367	0.041236	0.105449	1.000000	0.219174	-0.348712	-0.269543	-0.128663	-0.154216	0.057044	-0.381897	0.248
hear_rate	-0.128983	0.186781	0.007736	-0.142427	-0.175311	-0.111686	0.219174	1.000000	-0.343740	-0.278243	-0.063501	-0.249182	0.500663	-0.260329	0.572
calories	0.367765	-0.284641	-0.145043	0.152586	0.130245	0.126499	-0.348712	-0.343740	1.000000	0.573459	0.099427	0.280884	-0.060689	0.231400	-0.342
distance	0.769979	0.000810	0.009161	-0.014070	0.011994	0.003921	-0.269543	-0.278243	0.573459	1.000000	0.120923	0.175505	0.030327	0.443408	-0.354
entropy_heart	0.073038	-0.078122	-0.148143	0.205060	0.188729	0.135655	-0.128663	-0.063501	0.099427	0.120923	1.000000	0.690270	0.153819	0.109268	-0.179
entropy_setps	-0.014428	-0.352588	-0.157869	0.202071	0.224209	0.084253	-0.154216	-0.249182	0.280884	0.175505	0.690270	1.000000	0.137951	0.192676	-0.379
resting_heart	0.025611	0.001339	-0.014721	-0.221580	-0.152660	-0.072606	0.057044	0.500663	-0.060689	0.030327	0.153819	0.137951	1.000000	-0.023172	-0.205
corr_heart_steps	0.419740	-0.083589	0.044441	-0.044350	0.000757	-0.058874	-0.381897	-0.260329	0.231400	0.443408	0.109268	0.192676	-0.023172	1.000000	-0.363
norm_heart	-0.147622	0.292673	0.020439	-0.019953	-0.114239	-0.093932	0.248105	0.572105	-0.342403	-0.354104	-0.179524	-0.379523	-0.205891	-0.363236	1.000
intensity_karvonen	-0.159478	0.300794	0.056275	-0.037949	-0.143044	-0.107811	0.264148	0.623513	-0.364549	-0.368903	-0.182693	-0.384532	-0.142296	-0.376959	0.995
sd_norm_heart	-0.151097	0.373971	0.053571	-0.058753	-0.116326	-0.066281	0.285920	0.318184	-0.449993	-0.351207	-0.402368	-0.631186	-0.207322	-0.281512	0.553
steps_times_distance	0.531784	0.061941	0.026365	-0.043238	0.023465	0.007260	0.220429	-0.152713	0.351602	0.827001	0.083082	0.138071	0.078294	0.255483	-0.207

Correlation matrices and highlighted versions saved successfully.

```
Process finished with exit code 0
```

7.Missing values *(matrix, count, heatmap and dendrogram of missing values)*

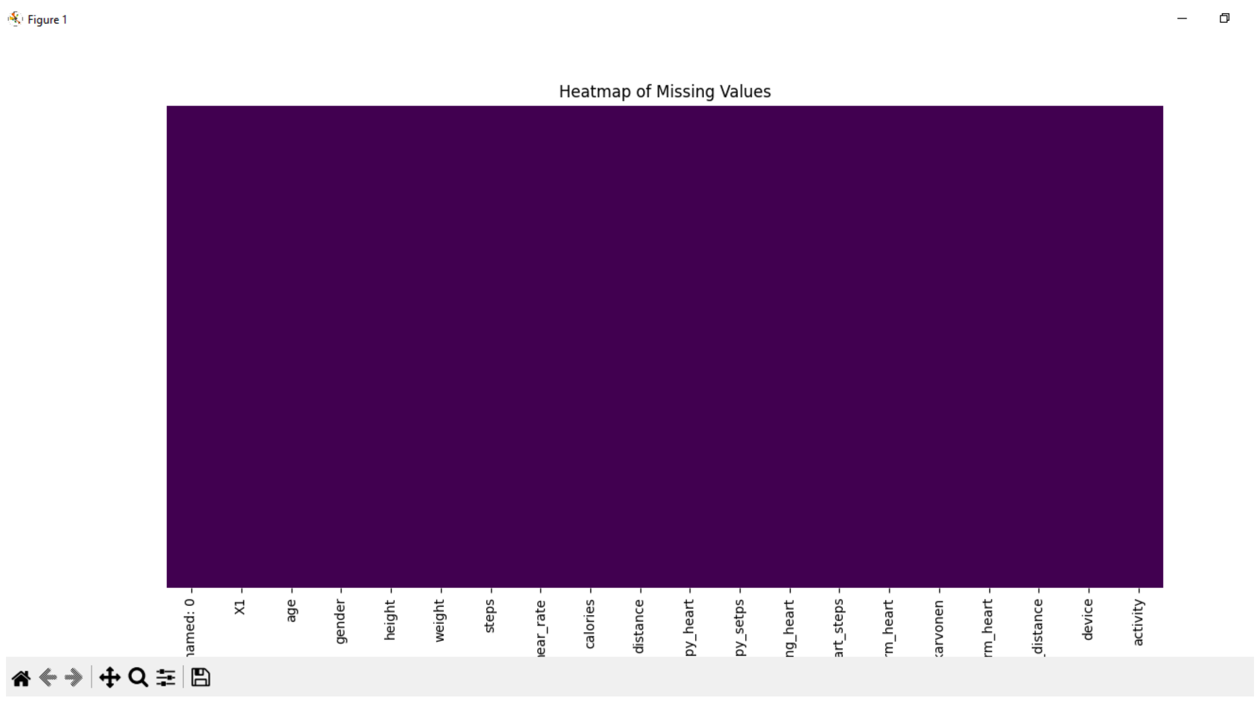
Missing_Values.py

```python
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno

# Define the file path to the dataset
file_path = "smartwatch.csv"

# Load the dataset into a pandas DataFrame
df = pd.read_csv(file_path)

# Calculate the count of missing values in each column
missing_values_count = df.isnull().sum()

# Print the count of missing values for each column
print("Missing Values Count:\n", missing_values_count)

# Plot a heatmap to visualize the locations of missing values
plt.figure(figsize=(12, 6))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis',
yticklabels=False)
plt.title('Heatmap of Missing Values')
plt.show()

# Plot a matrix to visualize the missing values using missingno
msno.matrix(df, figsize=(12, 6))
plt.title('Missing Values Matrix')
plt.show()

# Plot a dendrogram to visualize the hierarchical clustering of the
missing values
msno.dendrogram(df)
plt.title('Dendrogram of Missing Values')
plt.show()
```

```
# Save the count of missing values to a CSV file
missing_values_count.to_csv("missing_value_matrix_output.csv",
header=["Missing Values"])

# Print a success message
print("Missing values analysis completed and visualized
successfully.")
```

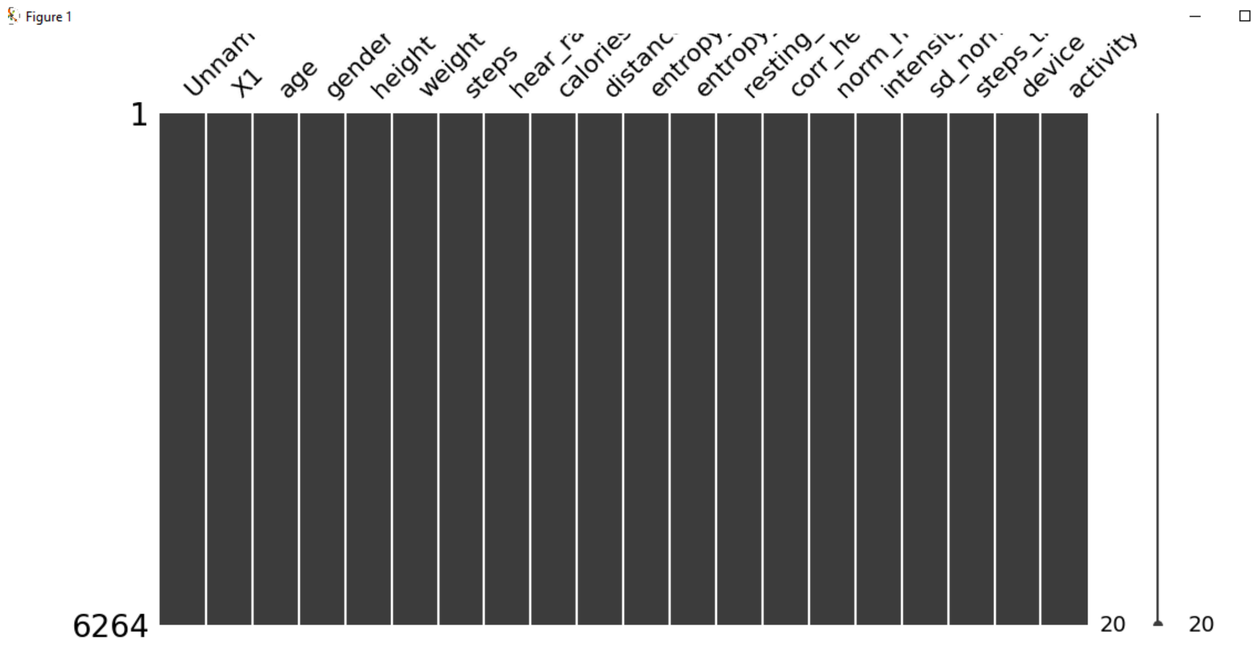
Output

Heat Map for Missing Values



```
Missing Values Count:
 Unnamed: 0              0
X1                      0
age                     0
gender                  0
height                  0
weight                  0
steps                   0
```

```
hear_rate              0
calories               0
distance               0
entropy_heart          0
entropy_setps          0
resting_heart          0
corr_heart_steps       0
norm_heart             0
intensity_karvonen     0
sd_norm_heart          0
steps_times_distance   0
device                 0
activity               0
dtype: int64
```



5.Data Modelling

Calculated Measures

Calculated Measures used in this project:

```
1.AvgHeartRate = AVERAGE(Fact_smartwatch[hear_rate])


2.AvgHeartRateByGender =

CALCULATE(

    AVERAGE(Fact_smartwatch[hear_rate]),

    ALLEXCEPT('Dim_Gender Info','Dim_Gender Info'[Gender])

)


3. AvgIntensityKarvonen = AVERAGE(Fact_smartwatch[intensity_karvonen])

4.AvgRestingHeartRate = AVERAGE(Fact_smartwatch[resting_heart])

5.AvgSteps = AVERAGE(Fact_smartwatch[steps])


6.CaloriesPerDistance = DIVIDE([TotalCalories], [TotalDistance])

7.CaloriesPerStep and DistancePerStep correlation for Month =

VAR __CORRELATION_TABLE = VALUES('Dim_Date'[Month])

VAR __COUNT =

    COUNTX(

        KEEPFILTERS(__CORRELATION_TABLE),

        CALCULATE(

            SUM('Fact_smartwatch'[CaloriesPerStep])

                * SUM('Fact_smartwatch'[DistancePerStep])
            )
        )
    VAR __SUM_X =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(SUM('Fact_smartwatch'[CaloriesPerStep]))
        )
    VAR __SUM_Y =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(SUM('Fact_smartwatch'[DistancePerStep]))
        )
    VAR __SUM_XY =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(
                SUM('Fact_smartwatch'[CaloriesPerStep])
                    * SUM('Fact_smartwatch'[DistancePerStep]) * 1.
            )
        )
    VAR __SUM_X2 =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(SUM('Fact_smartwatch'[CaloriesPerStep]) ^ 2)
        )
    VAR __SUM_Y2 =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
        CALCULATE(SUM('Fact_smartwatch'[DistancePerStep]) ^ 2)

    )

RETURN

    DIVIDE(

        __COUNT * __SUM_XY - __SUM_X * __SUM_Y * 1.,

        SQRT(

            (__COUNT * __SUM_X2 - __SUM_X ^ 2)

                * (__COUNT * __SUM_Y2 - __SUM_Y ^ 2)

        )

    )

8.Correlation_avgheartrate_AvgSteps =

VAR MeanX = AVERAGE(Fact_smartwatch[hear_rate])

VAR MeanY = AVERAGE(Fact_smartwatch[steps])

VAR CovarianceXY = SUMX(Fact_smartwatch, (Fact_smartwatch[hear_rate] -
MeanX) * (Fact_smartwatch[steps] - MeanY))

VAR StdDevX = SQRT(SUMX('Fact_smartwatch',
POWER(Fact_smartwatch[hear_rate] - MeanX, 2)))

VAR StdDevY = SQRT(SUMX('Fact_smartwatch', POWER(Fact_smartwatch[steps] -
MeanY, 2)))

RETURN

DIVIDE(CovarianceXY, StdDevX * StdDevY)



9.Correlation_AvgIntensityKarvonen_TotalCalories =

VAR MeanX = AVERAGE(Fact_smartwatch[intensity_karvonen])

VAR MeanY = AVERAGE(Fact_smartwatch[calories])

VAR CovarianceXY = SUMX(Fact_smartwatch,
(Fact_smartwatch[intensity_karvonen] - MeanX) *
('Fact_smartwatch'[calories] - MeanY))

VAR StdDevX = SQRT(SUMX('Fact_smartwatch',
POWER(Fact_smartwatch[intensity_karvonen] - MeanX, 2)))

VAR StdDevY = SQRT(SUMX('Fact_smartwatch',
POWER(Fact_smartwatch[calories] - MeanY, 2)))
```

```
RETURN

DIVIDE(CovarianceXY, StdDevX * StdDevY)


10.entropy_heart and entropy_setps correlation for Month =

VAR __CORRELATION_TABLE = VALUES('Dim_Date'[Month])

VAR __COUNT =

    COUNTX(

        KEEPFILTERS(__CORRELATION_TABLE),

        CALCULATE(

            SUM('Fact_smartwatch'[entropy_heart])

                * SUM('Fact_smartwatch'[entropy_setps])

        )

    )

VAR __SUM_X =

    SUMX(

        KEEPFILTERS(__CORRELATION_TABLE),

        CALCULATE(SUM('Fact_smartwatch'[entropy_heart]))

    )

VAR __SUM_Y =

    SUMX(

        KEEPFILTERS(__CORRELATION_TABLE),

        CALCULATE(SUM('Fact_smartwatch'[entropy_setps]))

    )

VAR __SUM_XY =

    SUMX(

        KEEPFILTERS(__CORRELATION_TABLE),

        CALCULATE(

            SUM('Fact_smartwatch'[entropy_heart])
```

```
                    * SUM('Fact_smartwatch'[entropy_setps]) * 1.
            )
        )
VAR __SUM_X2 =
    SUMX(
        KEEPFILTERS(__CORRELATION_TABLE),
        CALCULATE(SUM('Fact_smartwatch'[entropy_heart]) ^ 2)
    )
VAR __SUM_Y2 =
    SUMX(
        KEEPFILTERS(__CORRELATION_TABLE),
        CALCULATE(SUM('Fact_smartwatch'[entropy_setps]) ^ 2)
    )
RETURN
    DIVIDE(
        __COUNT * __SUM_XY - __SUM_X * __SUM_Y * 1.,
        SQRT(
            (__COUNT * __SUM_X2 - __SUM_X ^ 2)
                * (__COUNT * __SUM_Y2 - __SUM_Y ^ 2)
        )
    )
11.intensity_karvonen and corr_heart_steps correlation for Month =
VAR __CORRELATION_TABLE = VALUES('Dim_Date'[Month])
VAR __COUNT =
    COUNTX(
        KEEPFILTERS(__CORRELATION_TABLE),
        CALCULATE(
            SUM('Fact_smartwatch'[intensity_karvonen])
```

```
                    * SUM('Fact_smartwatch'[corr_heart_steps])
            )
        )
    VAR __SUM_X =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(SUM('Fact_smartwatch'[intensity_karvonen]))
        )
    VAR __SUM_Y =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(SUM('Fact_smartwatch'[corr_heart_steps]))
        )
    VAR __SUM_XY =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(
                SUM('Fact_smartwatch'[intensity_karvonen])
                    * SUM('Fact_smartwatch'[corr_heart_steps]) * 1.
            )
        )
    VAR __SUM_X2 =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),
            CALCULATE(SUM('Fact_smartwatch'[intensity_karvonen]) ^ 2)
        )
    VAR __SUM_Y2 =
        SUMX(
            KEEPFILTERS(__CORRELATION_TABLE),

            CALCULATE(SUM('Fact_smartwatch'[corr_heart_steps]) ^ 2)

        )

    RETURN

        DIVIDE(

            __COUNT * __SUM_XY - __SUM_X * __SUM_Y * 1.,

            SQRT(

                (__COUNT * __SUM_X2 - __SUM_X ^ 2)

                    * (__COUNT * __SUM_Y2 - __SUM_Y ^ 2)

            )

        )

    12.Constant Max for gauge

       Max = 1

    13.MaxHeartRate = MAX(Fact_smartwatch[hear_rate])

    14.MaxStepsGoalMetPercentage = 100

    15.MinHeartRate = MIN(Fact_smartwatch[hear_rate])

    16.NormHeartRate = AVERAGE(Fact_smartwatch[hear_rate])


    17.StepsGoalMet =

    IF(

        SUM(Fact_smartwatch[steps]) >=
    SELECTEDVALUE(UserInputParameterGoalSteps[UserInputParameterGoalSteps]),

        1,

        0

    )


    18.StepsGoalMetPercentage =

    DIVIDE(

        COUNTROWS(FILTER('Fact_smartwatch', [StepsGoalMet] = 1)),
        COUNTROWS(Fact_smartwatch)
    ) * 100


    19.TargetAvgHeartRate =
    SWITCH(
        TRUE(),
        MAX('Fact_smartwatch'[AgeGroup]) = "0-20", 60,
        MAX('Fact_smartwatch'[AgeGroup]) = "21-50", 70,
        MAX('Fact_smartwatch'[AgeGroup]) = "51-75", 65,
        76  -- Default target if no conditions are met
    )


    20.TargetBMI_Lower = 18



    21.TargetBMI_Upper = 20


    22.TargetCalories = 2500


    23.TargetDistance =
    SWITCH(
        TRUE(),
        MAX('Fact_smartwatch'[AgeGroup]) = "0-20",50,
        MAX('Fact_smartwatch'[AgeGroup]) = "21-50",60 ,
        MAX('Fact_smartwatch'[AgeGroup]) = "51-75",30,
        76  -- Default target if no conditions are met
    )
    24.TargetGoalSteps =
    SWITCH(
        TRUE(),
        MAX(Fact_smartwatch[AgeGroup]) = "0-20", 20800,
        MAX('Fact_smartwatch'[AgeGroup]) = "21-50",10000 ,
        MAX('Fact_smartwatch'[AgeGroup]) = "51-75", 1780,
        76  -- Default target if no conditions are met
    )


    25.TotalCalories = SUM(Fact_smartwatch[calories])


    26.TotalDistance = SUM(Fact_smartwatch[distance])


    27.WithinTargetBMI =
    IF(
        AND(
            [TargetBMI_Lower],
            [TargetBMI_Upper]
        ),
        1,
        0
    )
```

Created Bins

1.Intensity_Karvonen(bins)

2.Heart_Rate(bins)

3.Calories(bins)

Calculated Columns

```
1. AgeGroup =

SWITCH(TRUE(),

    [Age] <= 20, "0-20",

    [Age] <= 30, "21-30",

    [Age] <= 40, "31-40",

    [Age] <= 50, "41-50",

    "51+"

)

2. BMI = DIVIDE('Fact_smartwatch'[weight], ('Fact_smartwatch'[height] / 100) ^ 2)


3.BMICategory =

SWITCH(

    TRUE(),

    'Fact_smartwatch'[BMI]< 18.5, "Underweight",

    'Fact_smartwatch'[BMI] >= 18.5 && 'Fact_smartwatch'[BMI] < 24.9, "Normal weight",

    'Fact_smartwatch'[BMI]  >= 25 && 'Fact_smartwatch'[BMI]  < 29.9, "Overweight",

    'Fact_smartwatch'[BMI]  >= 30, "Obese"

)


4.BMIRecommendation =

SWITCH(Fact_smartwatch[BMICategory],

    "Underweight", "Increase calorie intake and engage in muscle-building exercises.",

    "Normal weight", "Maintain current habits and ensure balanced nutrition.",

    "Overweight", "Focus on a balanced diet and increase physical activity.",

    "Obese", "Consult with a healthcare provider for a personalized plan."
)
```

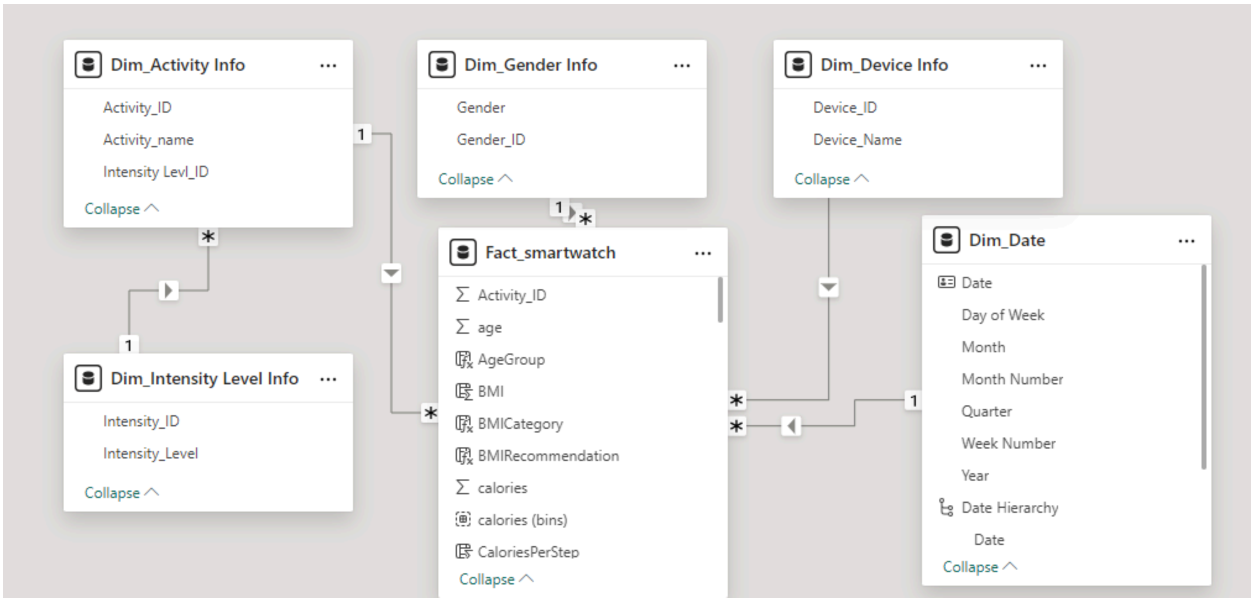
```
5.CaloriesPerStep = DIVIDE(Fact_smartwatch[calories], Fact_smartwatch[steps])
```

```
6.Custom based Calculated column for Date Date =
RANDBETWEEN(DATE(2020,10,1),DATE(2024,11,20))
```

```
7.DistancePerStep = DIVIDE([distance], [steps])
```

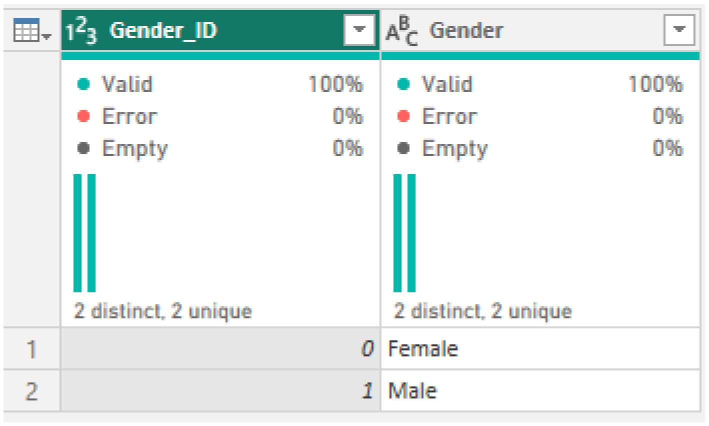
```
8.HRV = [MaxHeartRate] - [MinHeartRate]
```

Data Modelling(Snowflake Model)



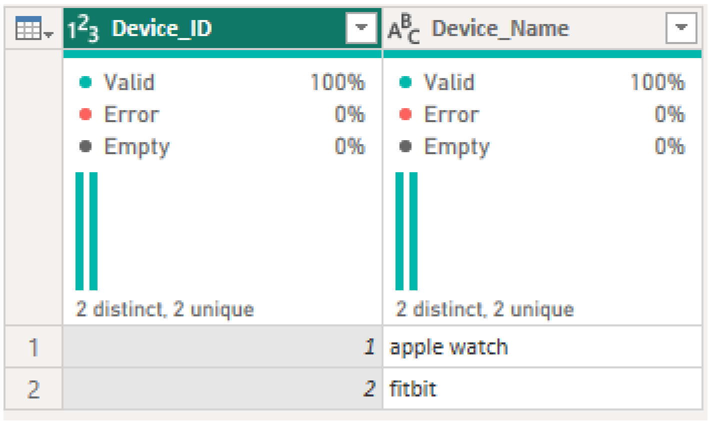
Dimension Tables

1.Dim_Gender Info



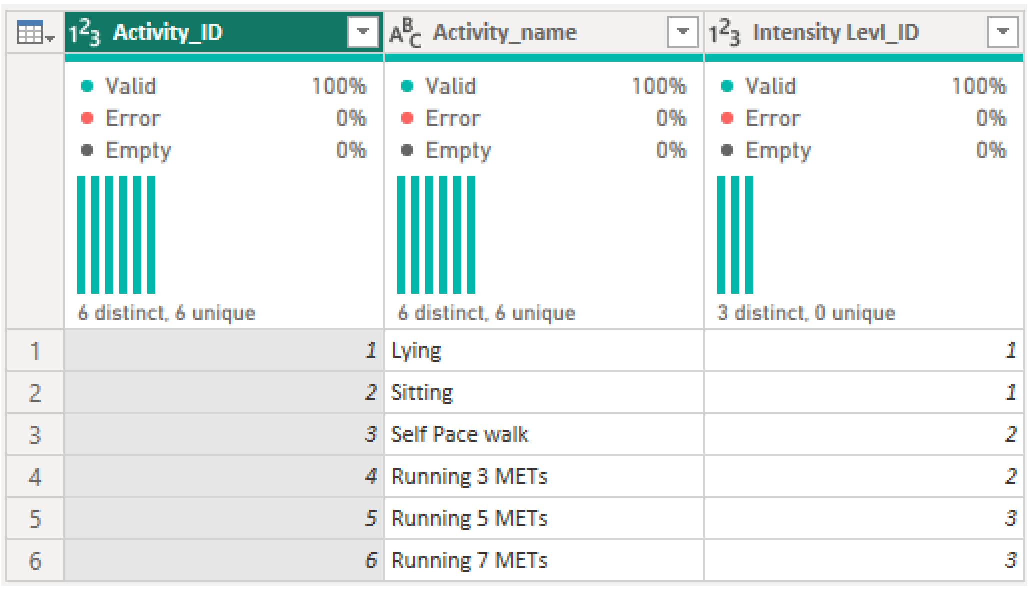
	Gender_ID	Gender
	● Valid 100% ● Error 0% ● Empty 0%	● Valid 100% ● Error 0% ● Empty 0%
	2 distinct, 2 unique	2 distinct, 2 unique
1	0	Female
2	1	Male

2.Dim_Device Info



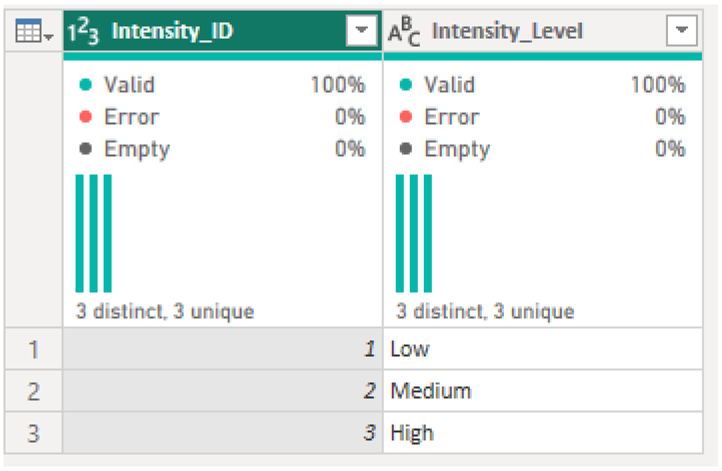
	Device_ID	Device_Name
	● Valid 100% ● Error 0% ● Empty 0%	● Valid 100% ● Error 0% ● Empty 0%
	2 distinct, 2 unique	2 distinct, 2 unique
1	1	apple watch
2	2	fitbit

3.Dim_Activity Info



	Activity_ID	Activity_name	Intensity Levl_ID
	● Valid 100% ● Error 0% ● Empty 0%	● Valid 100% ● Error 0% ● Empty 0%	● Valid 100% ● Error 0% ● Empty 0%
	6 distinct, 6 unique	6 distinct, 6 unique	3 distinct, 0 unique
1	1	Lying	1
2	2	Sitting	1
3	3	Self Pace walk	2
4	4	Running 3 METs	2
5	5	Running 5 METs	3
6	6	Running 7 METs	3

4.Dim_Intensity_Level Info



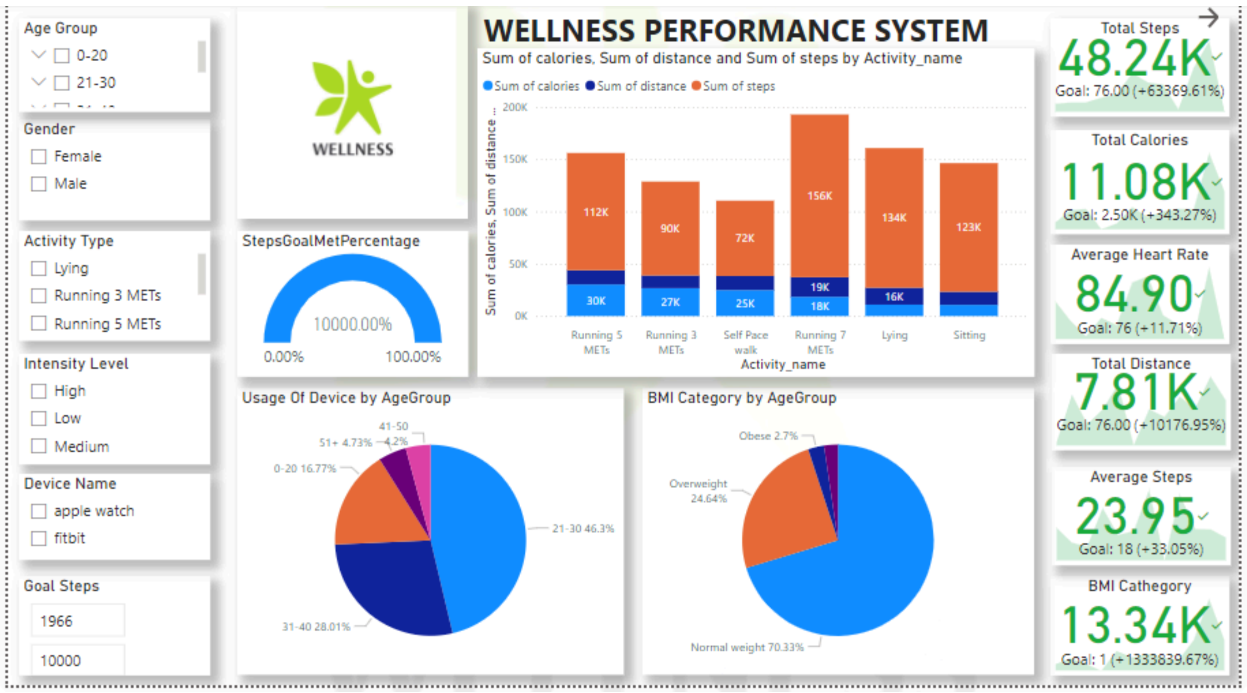
	Intensity_ID	Intensity_Level
	Valid 100% / Error 0% / Empty 0%	Valid 100% / Error 0% / Empty 0%
	3 distinct, 3 unique	3 distinct, 3 unique
1	1	Low
2	2	Medium
3	3	High

Fact Table

Fact_smartwatch

6.Exploratory Data Analysis (EDA)

Page 1: Fitness Performance Analytics



Slicers:

- Age Group
- Gender
- Activity Type
- Intensity Level
- Device Name

Findings: Specific age groups and genders showed variations in performance metrics. For instance, younger age groups tended to have higher average steps and calorie expenditure.

Gauges:

- **Steps Goal Met Percentage:**
  - **Usage:** Shows the percentage of the daily steps goal met by the user. It helps in understanding daily activity levels and whether the user is meeting their fitness goals.
  - **Findings:** A significant portion of users met their step goals, indicating a high level of engagement with their fitness routines.

Pie Charts:

- **Device by Age Group:**
  - **Usage:** Visualizes the distribution of different fitness devices used across various age groups. Helps in understanding device preference and usage patterns among different age demographics.
- **BMI Category by Age Group:**

  Usage: Displays the distribution of BMI categories across different age groups. Useful for identifying age-specific trends in body mass index and potential health risks.

  Findings: Certain devices were more popular among specific age groups. BMI distribution highlighted that younger age groups had a more balanced BMI spread.

KPIs

Findings: Specific age groups and genders showed variations in performance metrics. For instance, younger age groups tended to have higher average steps and calorie expenditure.

- **Total Steps**

  Description: This KPI displays the total number of steps taken by the user over a specified period.

Usage: Tracking total steps helps users monitor their daily activity levels. It encourages them to stay active and reach their daily step goals, which is crucial for maintaining cardiovascular health and overall fitness.

- **Total Calories**

  Description: This KPI shows the total number of calories burned by the user.

  Usage: Monitoring calorie expenditure helps users manage their weight and assess the effectiveness of their workouts. It provides insights into how different activities contribute to calorie burning and helps in planning balanced diets and exercise routines.

- **Average Heart Rate**

  Description: Displays the average heart rate of the user during their activities.

  Usage: The average heart rate is an indicator of cardiovascular health and fitness levels. By monitoring this metric, users can ensure they are exercising within their optimal heart rate zones, which is important for improving endurance and cardiovascular efficiency.

- **Total Distance**
  - **Description:** This KPI shows the total distance covered by the user, typically measured in kilometers or miles.
  - **Usage:** Tracking the total distance helps users evaluate their endurance and stamina. It is especially useful for runners, cyclists, and walkers to set and achieve their distance goals.
- **Average Steps:**
  - **Description:** This KPI indicates the average number of steps taken by the user over a specified period.
  - **Usage:** By observing average steps, users can gauge their consistency in staying active. It helps in identifying patterns and adjusting daily routines to meet activity targets.
- **BMI Category:**
  - **Description:** This KPI categorizes the user's Body Mass Index (BMI) into different ranges (e.g., underweight, normal weight, overweight, and obese).
  - **Usage:** BMI is a widely used indicator of body fatness. Monitoring BMI helps users understand their weight status in relation to their height. It provides a

quick assessment of whether they are at a healthy weight or if they need to make changes to their diet and exercise habits.

Significance and Benefits for Health and Fitness Analysis

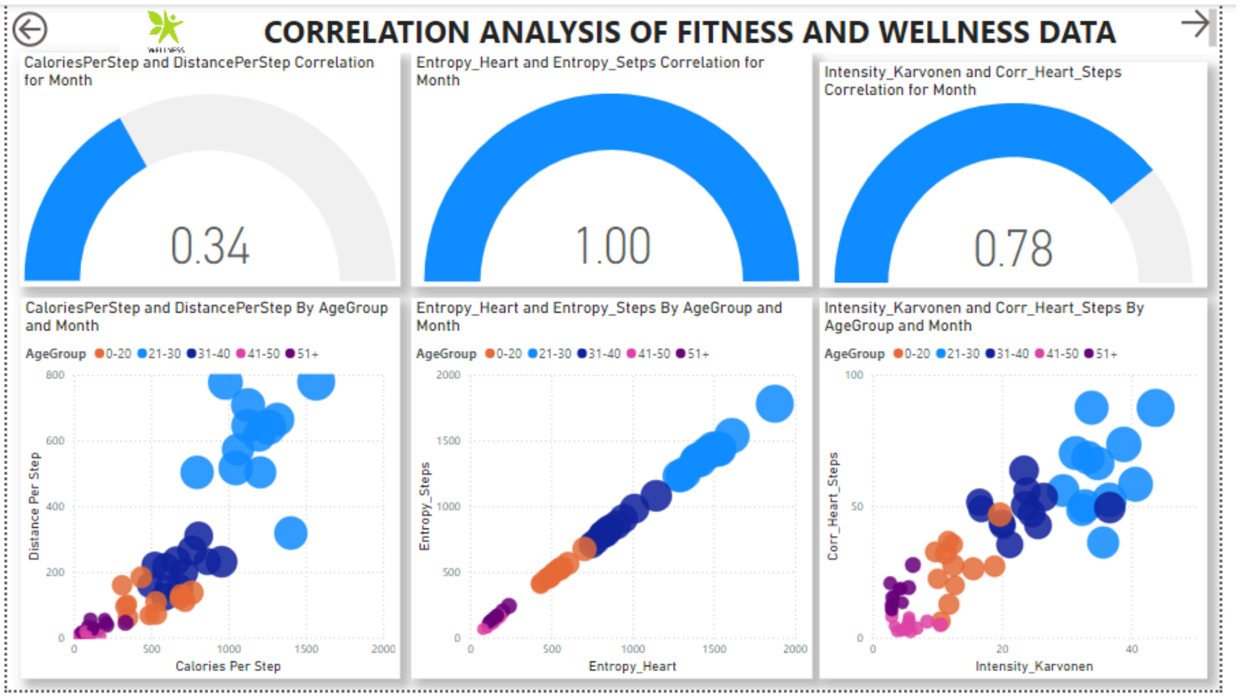
- **Health Monitoring:** Average Heart Rate and Total Calories provide crucial insights into cardiovascular health and the effectiveness of workouts, helping users optimize their exercise routines.
- **Consistency Tracking:** Average Steps allows users to track their activity consistency over time, encouraging them to maintain a regular exercise habit.
- **Weight Management:** The BMI Category KPI helps users understand their weight status, guiding them in making informed decisions about diet and exercise to maintain a healthy weight.
- **Performance Evaluation:** By regularly monitoring these KPIs, users can evaluate their progress, identify areas for improvement, and make necessary adjustments to their fitness plans.

By including these KPIs on the first page of your FitWell Analytics report, it provides a comprehensive overview of their fitness performance. This helps users stay informed, motivated, and engaged in their health and wellness journey.

Clustered Chart:

- **Sum of Calories, Sum of Distance, and Sum of Steps by Activity Name:**
  - **Usage:** Illustrates the total calories burned, distance covered, and steps taken for each activity. This helps in identifying the most effective activities for calorie burning and overall fitness.
  - **Findings:** Running 5METS and 7METs were the top activities for calorie burning and distance covered.

Page 2: Correlation Analytics


CORRELATION ANALYSIS OF FITNESS AND WELLNESS DATA
CaloriesPerStep and DistancePerStep Correlation for Month
0.34
Entropy_Heart and Entropy_Setps Correlation for Month
1.00
Intensity_Karvonen and Corr_Heart_Steps Correlation for Month
0.78
CaloriesPerStep and DistancePerStep By AgeGroup and Month
AgeGroup ●0-20 ●21-30 ●31-40 ●41-50 ●51+
Distance Per Step
Calories Per Step
Entropy_Heart and Entropy_Steps By AgeGroup and Month
Entropy_Steps
Entropy_Heart
Intensity_Karvonen and Corr_Heart_Steps By AgeGroup and Month
Corr_Heart_Steps
Intensity_Karvonen

Scatter Charts:

1. Calories per Step and Distance per Step by Age Group and Month:

- **Gauge:** Set by Calories per Step and Distance per Step Correlation for Month.
- **Usage:** Shows the relationship between calories burned and distance per step across different age groups and months. Useful for identifying patterns in efficiency and performance across age groups and over time.
- **Findings:** A strong positive correlation was observed between calories burned per step and distance covered per step, especially in younger age groups.

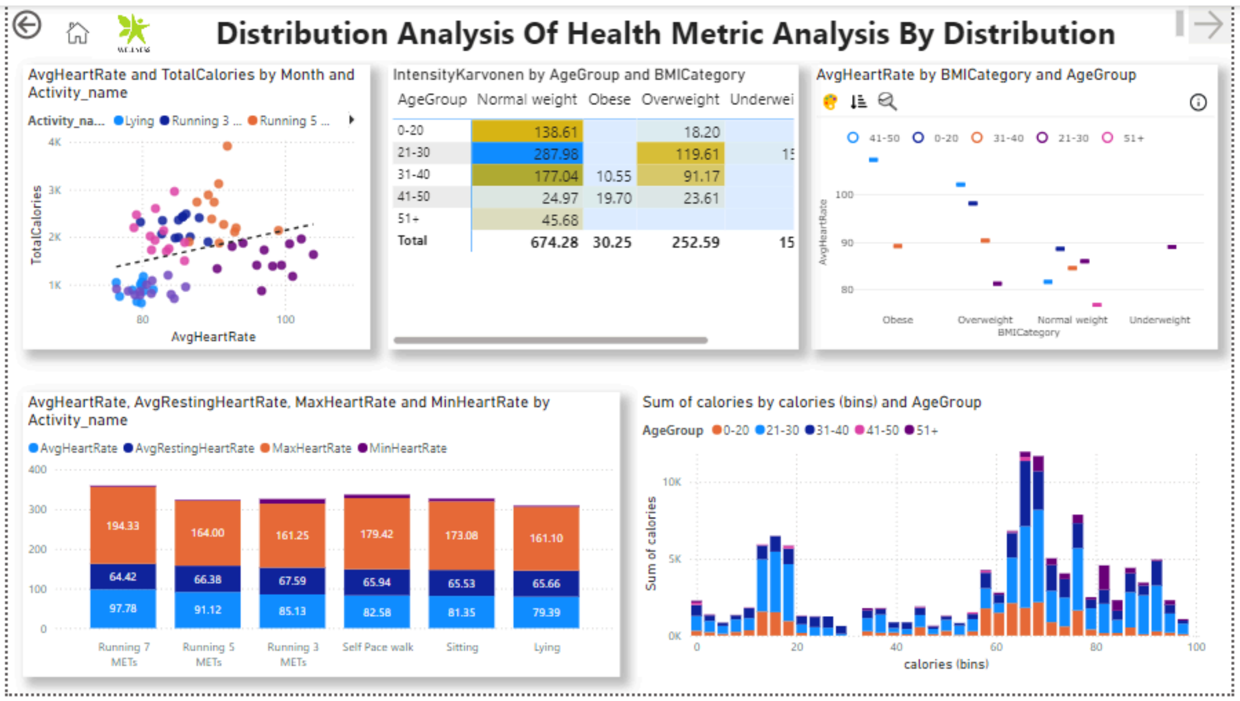
2. Entropy Heart and Entropy Steps by Age Group and Month:

- **Gauge:** Set by Entropy Heart and Entropy Steps Correlation for Month.
- **Usage:** Displays the correlation between heart rate variability (Entropy Heart) and step variability (Entropy Steps) across different age groups and months. Helps in understanding the variability in physical activity and heart rate.
- **Findings:** A strong correlation was found, indicating similar heart rate and steps, suggesting constant activity patterns among users.

3. Intensity Karvonen and Corr Heart Steps by Age Group and Month:

- **Gauge:** Set by Intensity Karvonen and Corr Heart Steps Correlation for Month.
- **Usage:** Illustrates the relationship between exercise intensity (Intensity Karvonen) and the correlation between heart rate and steps. Useful for assessing the impact of exercise intensity on heart rate and step correlation.
- **Findings:** A strong correlation was identified, showing that higher exercise intensity led to a more consistent relationship between heart rate and steps.

Page 3: Distribution Analysis


Distribution Analysis Of Health Metric Analysis By Distribution
AvgHeartRate and TotalCalories by Month and Activity_name
Activity_na... ● Lying ● Running 3 ... ● Running 5 ...
TotalCalories
AvgHeartRate
AvgHeartRate by BMICategory and AgeGroup
41-50 0-20 31-40 21-30 51+
Obese Overweight Normal weight Underweight
BMICategory
AvgHeartRate, AvgRestingHeartRate, MaxHeartRate and MinHeartRate by Activity_name
● AvgHeartRate ● AvgRestingHeartRate ● MaxHeartRate ● MinHeartRate
194.33 164.00 161.25 179.42 173.08 161.10
64.42 66.38 67.59 65.94 65.53 65.66
97.78 91.12 85.13 82.58 81.35 79.39
Running 7 METs
Running 5 METs
Running 3 METs
Self Pace walk
Sitting
Lying
Sum of calories by calories (bins) and AgeGroup
AgeGroup ● 0-20 ● 21-30 ● 31-40 ● 41-50 ● 51+
Sum of calories
calories (bins)

IntensityKarvonen by AgeGroup and BMICategory

AgeGroup	Normal weight	Obese	Overweight	Underwei
0-20	138.61		18.20	
21-30	287.98		119.61	15
31-40	177.04	10.55	91.17	
41-50	24.97	19.70	23.61	
51+	45.68			
Total	**674.28**	**30.25**	**252.59**	**15**

Clustered Charts:

1. **Avg Heart Rate, Avg Resting Heart Rate, Max Heart Rate, and Min Heart Rate by Activity Name:**
    - **Usage:** Visualizes the average, resting, maximum, and minimum heart rates for different activities. Helps in understanding how different activities affect heart rate.
    - **Findings:** High-intensity activities like running 7METs   showed higher maximum heart rates.
2. **Sum of Calories by Calories (bins) and Age Group:**
    - **Usage:** Displays the total calories burned across different calorie ranges (bins) and age groups. Useful for identifying calorie expenditure trends among different age demographics.
    - **Findings:** Younger age groups(21-30) tended to have higher calorie burns, indicating more vigorous activity levels.

Matrix:

- **Intensity Karvonen by Age Group and BMI Category:**
    - **Usage:** Shows the average exercise intensity (Intensity Karvonen) categorized by age group and BMI category. Helps in assessing how exercise intensity varies across different demographics.

- **Findings:** Higher intensity levels were prevalent in users with lower BMI, suggesting better cardiovascular fitness.

Scatter Chart:

- **Avg Heart Rate and Total Calories by Month and Activity Name:**
  - **Usage:** Illustrates the relationship between average heart rate and total calories burned, categorized by month and activity name. Useful for understanding how heart rate affects calorie burning across different activities and time periods.
  - **Findings:** Consistent trends were observed where months with higher average heart rates also had higher total calorie expenditures.

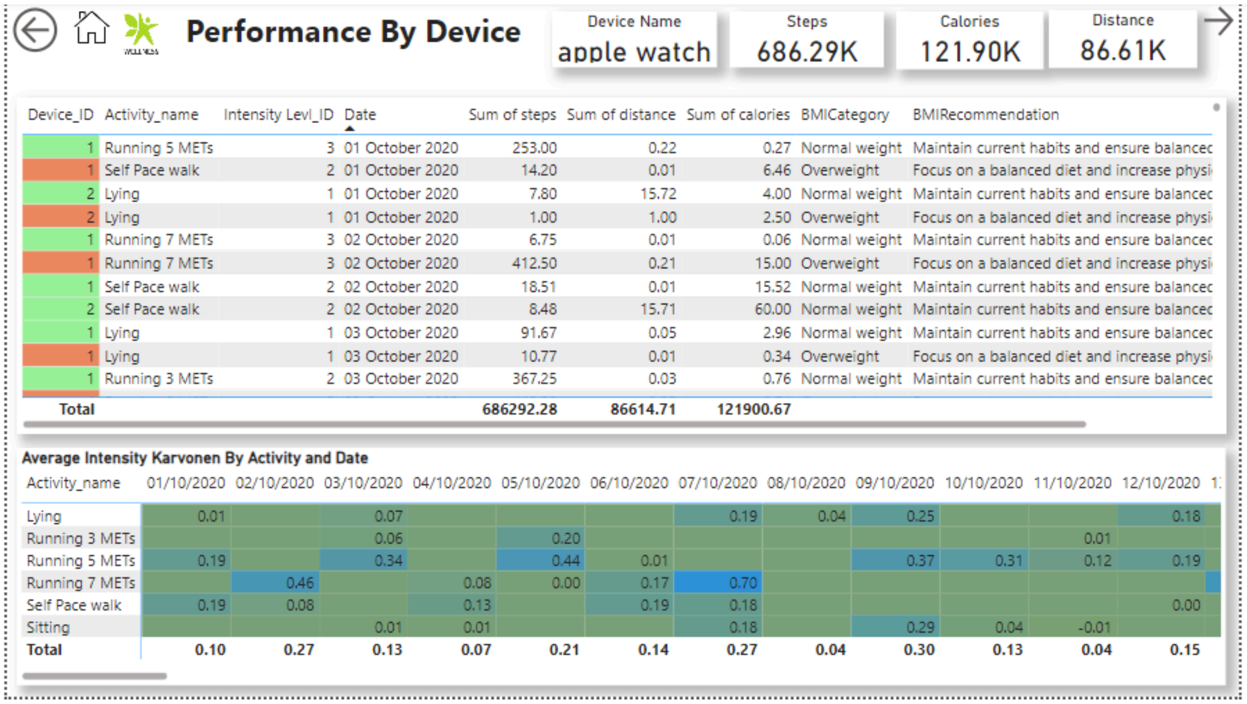
Histogram:

- **Avg Heart Rate by BMI Category and Age Group:**
  - **Usage:** Displays the distribution of average heart rate across different BMI categories and age groups. Helps in identifying heart rate trends among different demographics.
  - **Findings:** Normal BMI individuals had more stable average heart rates across age groups.

Matrix:

- **Average Intensity Karvonen by Activity and Date:**
  - **Usage:** Shows the average exercise intensity (Karvonen method) for different activities over time. Useful for tracking intensity trends and changes in exercise routines.
  - **Findings:** Intensities varied significantly by activity, with a noticeable increase in intensity during fitness challenges or events.

Page 4: Device Drillthrough Analysis



Performance By Device

Device Name	Steps	Calories	Distance
apple watch	686.29K	121.90K	86.61K

Device_ID	Activity_name	Intensity Levl_ID	Date	Sum of steps	Sum of distance	Sum of calories	BMICategory	BMIRecommendation
1	Running 5 METs	3	01 October 2020	253.00	0.22	0.27	Normal weight	Maintain current habits and ensure balancec
1	Self Pace walk	2	01 October 2020	14.20	0.01	6.46	Overweight	Focus on a balanced diet and increase physi
2	Lying	1	01 October 2020	7.80	15.72	4.00	Normal weight	Maintain current habits and ensure balancec
2	Lying	1	01 October 2020	1.00	1.00	2.50	Overweight	Focus on a balanced diet and increase physi
1	Running 7 METs	3	02 October 2020	6.75	0.01	0.06	Normal weight	Maintain current habits and ensure balancec
1	Running 7 METs	3	02 October 2020	412.50	0.21	15.00	Overweight	Focus on a balanced diet and increase physi
1	Self Pace walk	2	02 October 2020	18.51	0.01	15.52	Normal weight	Maintain current habits and ensure balancec
2	Self Pace walk	2	02 October 2020	8.48	15.71	60.00	Normal weight	Maintain current habits and ensure balancec
1	Lying	1	03 October 2020	91.67	0.05	2.96	Normal weight	Maintain current habits and ensure balancec
1	Lying	1	03 October 2020	10.77	0.01	0.34	Overweight	Focus on a balanced diet and increase physi
1	Running 3 METs	2	03 October 2020	367.25	0.03	0.76	Normal weight	Maintain current habits and ensure balancec
Total				686292.28	86614.71	121900.67		

Average Intensity Karvonen By Activity and Date

Activity_name	01/10/2020	02/10/2020	03/10/2020	04/10/2020	05/10/2020	06/10/2020	07/10/2020	08/10/2020	09/10/2020	10/10/2020	11/10/2020	12/10/2020
Lying	0.01		0.07				0.19	0.04	0.25			0.18
Running 3 METs			0.06		0.20						0.01	
Running 5 METs	0.19		0.34		0.44	0.01			0.37	0.31	0.12	0.19
Running 7 METs		0.46		0.08	0.00	0.17	0.70					
Self Pace walk	0.19	0.08		0.13		0.19	0.18					0.00
Sitting			0.01	0.01			0.18		0.29	0.04	-0.01	
Total	**0.10**	**0.27**	**0.13**	**0.07**	**0.21**	**0.14**	**0.27**	**0.04**	**0.30**	**0.13**	**0.04**	**0.15**

Cards:

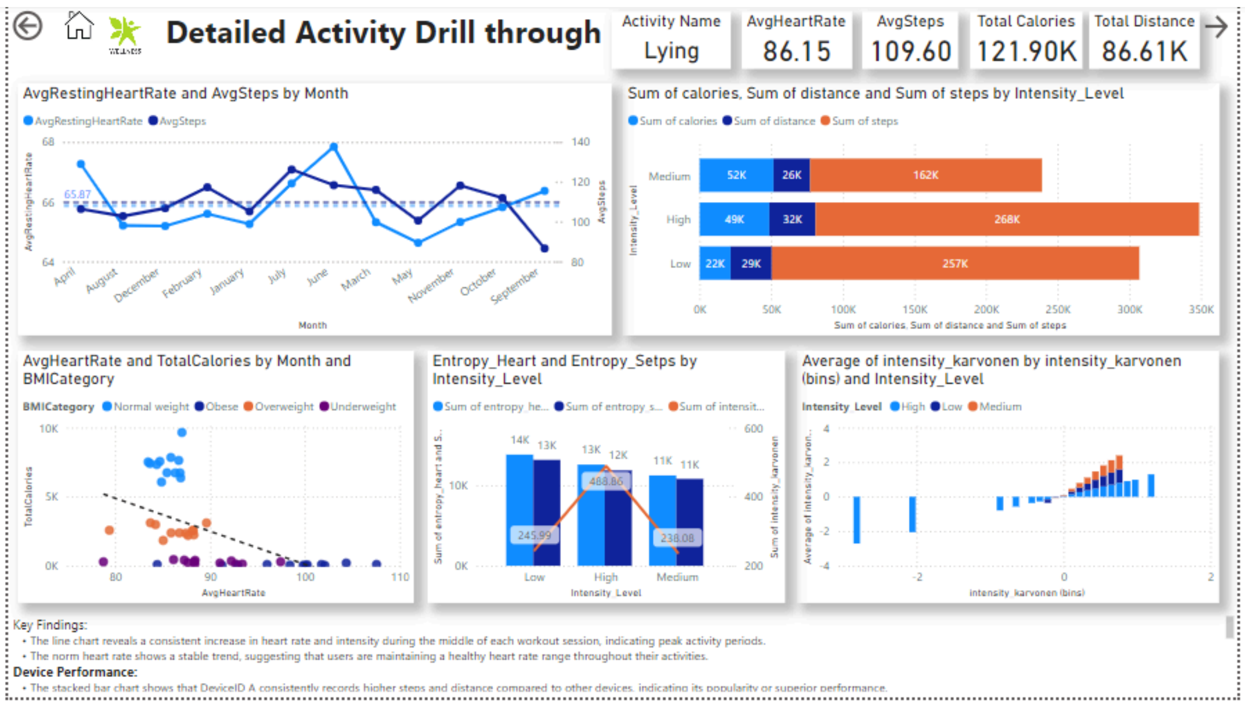
- **Device Name**
- **Steps**
- **Calories**
- **Distance**
  - **Usage:** Provide a summary of key metrics for specific fitness devices, including the device name, total steps, calories burned, and distance covered. Useful for evaluating the performance and usage of different devices.

Detailed Table:

- Device ID
- Activity Name
- Intensity Level
- Date
- Sum of Steps
- Sum of Distance
- Sum of Calories
- BMI Category
- BMI Recommendations
  - **Usage:** Offers a detailed breakdown of activity data by device ID, including activity name, intensity level, date, and summarized fitness metrics. Provides insights into specific device performance and user activity patterns.

- **Findings:** Detailed insights into user activity helped identify patterns and preferences, aiding in personalized recommendations.

Page 5: Additional Drillthrough Analysis


Detailed Activity Drill through
AvgRestingHeartRate and AvgSteps by Month
Sum of calories, Sum of distance and Sum of steps by Intensity_Level
AvgHeartRate and TotalCalories by Month and BMICategory
Entropy_Heart and Entropy_Setps by Intensity_Level
Average of intensity_karvonen by intensity_karvonen (bins) and Intensity_Level

Activity Name	AvgHeartRate	AvgSteps	Total Calories	Total Distance
Lying	86.15	109.60	121.90K	86.61K

Key Findings:
• The line chart reveals a consistent increase in heart rate and intensity during the middle of each workout session, indicating peak activity periods.
• The norm heart rate shows a stable trend, suggesting that users are maintaining a healthy heart rate range throughout their activities.
Device Performance:
• The stacked bar chart shows that DeviceID A consistently records higher steps and distance compared to other devices, indicating its popularity or superior performance.

Line Graph:

- **Avg Resting Heart Rate by Avg Steps by Month (averages):**
  - **Usage:** Shows the relationship between average resting heart rate and average steps per month. Useful for identifying trends and correlations in resting heart rate and physical activity levels.
  - **Findings:** Steady improvements in resting heart rate correlated with higher average steps, indicating improved fitness levels.

Clustered Bar Chart:

- **Sum of Calories, Sum of Distance, and Sum of Steps by Intensity Level:**
  - **Usage:** Displays the total calories burned, distance covered, and steps taken across different intensity levels. Helps in understanding the impact of exercise intensity on fitness metrics.
  - **Findings:** Higher intensity levels resulted in greater calorie burns and distances covered.

Scatter Chart with Trend Line:

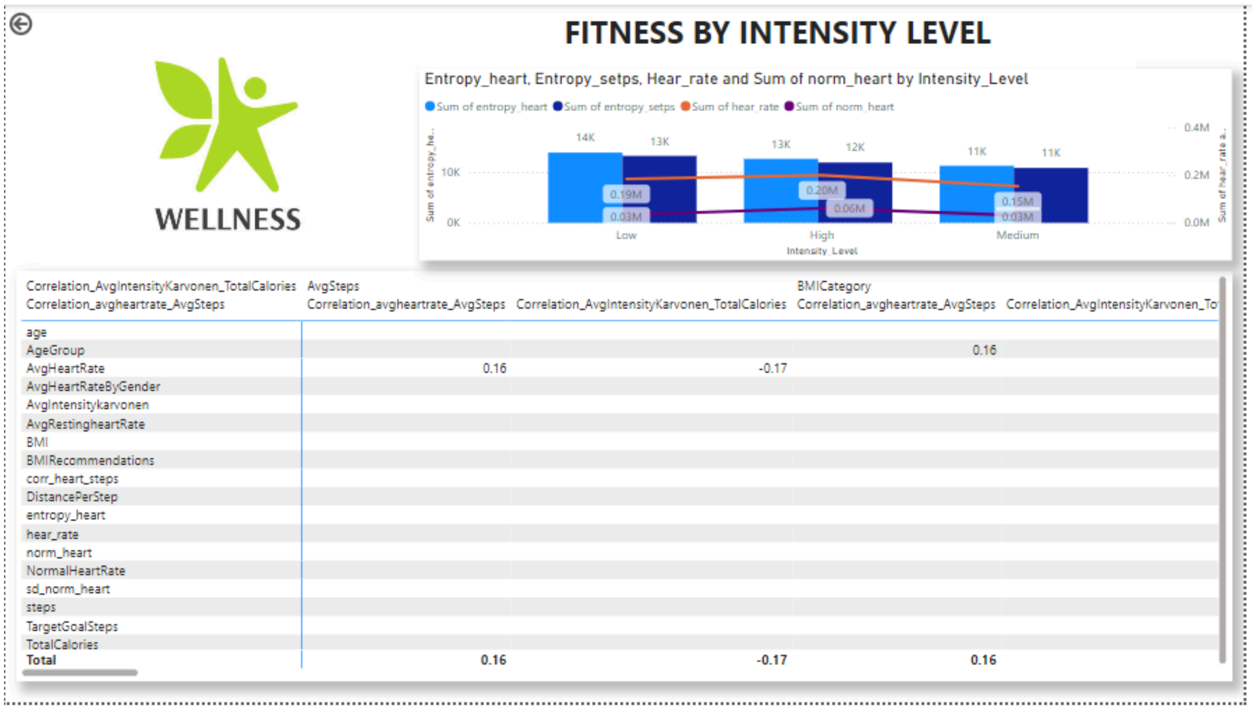
- **Avg Heart Rate by Total Calories by Month and BMI Category:**

- **Usage:** Illustrates the relationship between average heart rate and total calories burned, categorized by month and BMI category. Useful for identifying trends and correlations in heart rate and calorie expenditure.
- **Findings:** Strong relationship between average heart rate and total calories, with variations across BMI categories.

Stacked Bar Chart with Trend Line:

- **Entropy Heart and Entropy Steps by Intensity Levels:**
  - **Usage:** Shows the relationship between heart rate variability (Entropy Heart) and step variability (Entropy Steps) across different intensity levels. Helps in assessing the variability in physical activity and heart rate based on exercise intensity.
  - **Findings:** High variability in intensity karvonen was observed at higher intensity levels.

Page 6: Intensity Level Details Drillthrough Page


FITNESS BY INTENSITY LEVEL
WELLNESS
Entropy_heart, Entropy_setps, Hear_rate and Sum of norm_heart by Intensity_Level
● Sum of entropy_heart ● Sum of entropy_setps ● Sum of hear_rate ● Sum of norm_heart
Sum of entropy_he...
10K
0K
14K
13K
13K
12K
11K
11K
0.19M
0.03M
0.20M
0.06M
0.15M
0.03M
Low
High
Medium
Intensity_Level
0.4M
0.2M
0.0M
Sum of hear_rate a...

Correlation_AvgIntensityKarvonen_TotalCalories	AvgSteps		BMICategory	
Correlation_avgheartrate_AvgSteps	Correlation_avgheartrate_AvgSteps	Correlation_AvgIntensityKarvonen_TotalCalories	Correlation_avgheartrate_AvgSteps	Correlation_AvgIntensityKarvonen_To
age				
AgeGroup			0.16	
AvgHeartRate	0.16	-0.17		
AvgHeartRateByGender				
AvgIntensitykarvonen				
AvgRestingheartRate				
BMI				
BMIRecommendations				
corr_heart_steps				
DistancePerStep				
entropy_heart				
hear_rate				
norm_heart				
NormalHeartRate				
sd_norm_heart				
steps				
TargetGoalSteps				
TotalCalories				
Total	**0.16**	**-0.17**	**0.16**	

Matrix created by the Correlation pair

- **User-specific details and metrics**
  - **Usage**: Provides personalized insights and recommendations for individual users based on their specific activity data and fitness metrics.
  - Personalized data provided actionable insights, helping users tailor their fitness routines for better outcomes.

Specific Findings:

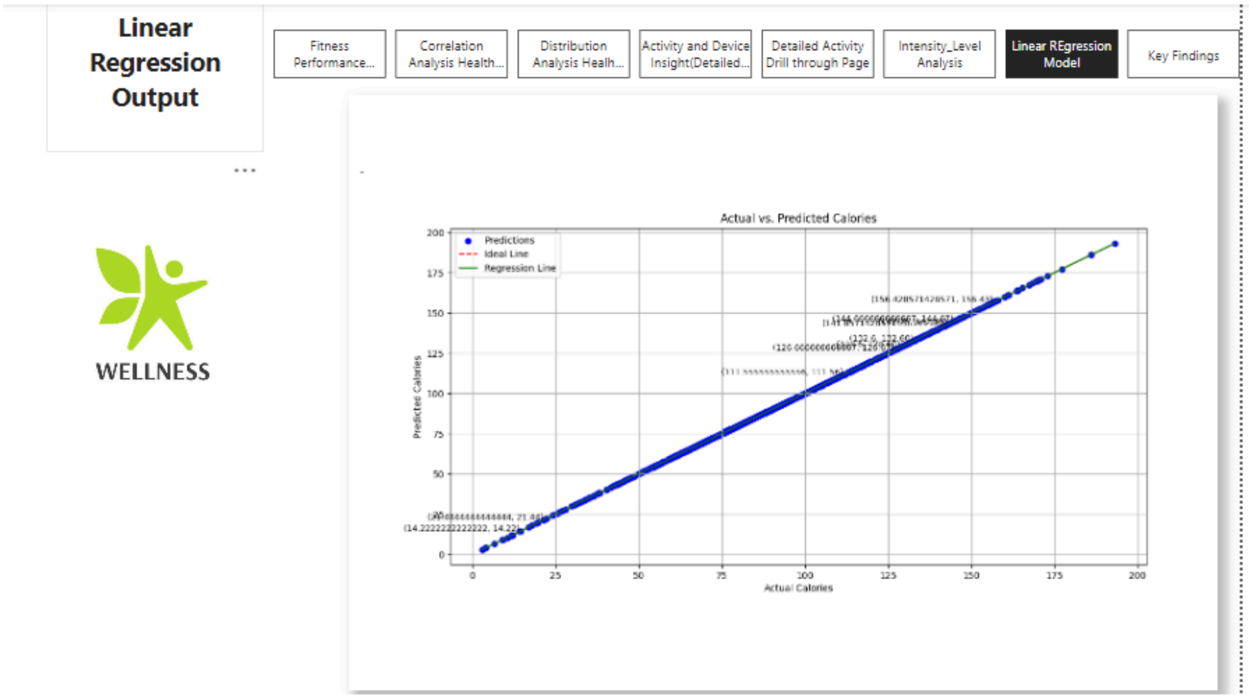
- **Avg Heart Rate and Avg Steps:** The positive correlation suggests that higher activity levels are linked to elevated heart rates, providing insights into cardiovascular fitness.
- **Avg Intensity Karvonen and Total Calories:** The strong positive correlation indicates that higher exercise intensity results in more calories burned, highlighting the effectiveness of intense workouts.

Line and Clustered Column Chart

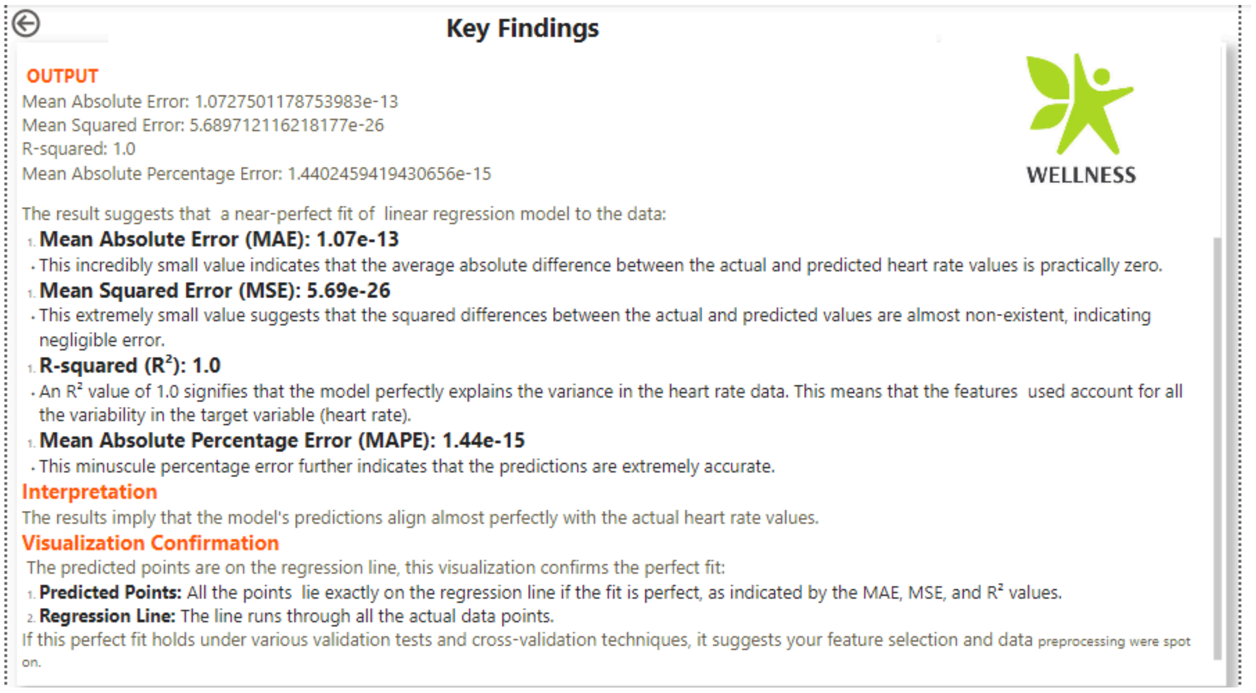
Entropy_heart, Entropy_setps, Hear_rate and Sum of norm_heart by Intensity_Level

In this the heart rate is high in higher intensity level activities.

Page7. Linear Regression



Page 8. Key Findings



Key Findings

OUTPUT

```
Mean Absolute Error: 1.0727501178753983e-13
Mean Squared Error: 5.689712116218177e-26
R-squared: 1.0
Mean Absolute Percentage Error: 1.4402459419430656e-15
```

The result suggests that a near-perfect fit of linear regression model to the data:

1. **Mean Absolute Error (MAE): 1.07e-13**
   - This incredibly small value indicates that the average absolute difference between the actual and predicted heart rate values is practically zero.
1. **Mean Squared Error (MSE): 5.69e-26**
   - This extremely small value suggests that the squared differences between the actual and predicted values are almost non-existent, indicating negligible error.
1. **R-squared (R^2): 1.0**
   - An R^2 value of 1.0 signifies that the model perfectly explains the variance in the heart rate data. This means that the features used account for all the variability in the target variable (heart rate).
1. **Mean Absolute Percentage Error (MAPE): 1.44e-15**
   - This minuscule percentage error further indicates that the predictions are extremely accurate.

Interpretation

The results imply that the model's predictions align almost perfectly with the actual heart rate values.

Visualization Confirmation

The predicted points are on the regression line, this visualization confirms the perfect fit:

1. **Predicted Points:** All the points lie exactly on the regression line if the fit is perfect, as indicated by the MAE, MSE, and R^2 values.
2. **Regression Line:** The line runs through all the actual data points.

If this perfect fit holds under various validation tests and cross-validation techniques, it suggests your feature selection and data preprocessing were spot on.

7.LINEAR REGRESSION MODEL

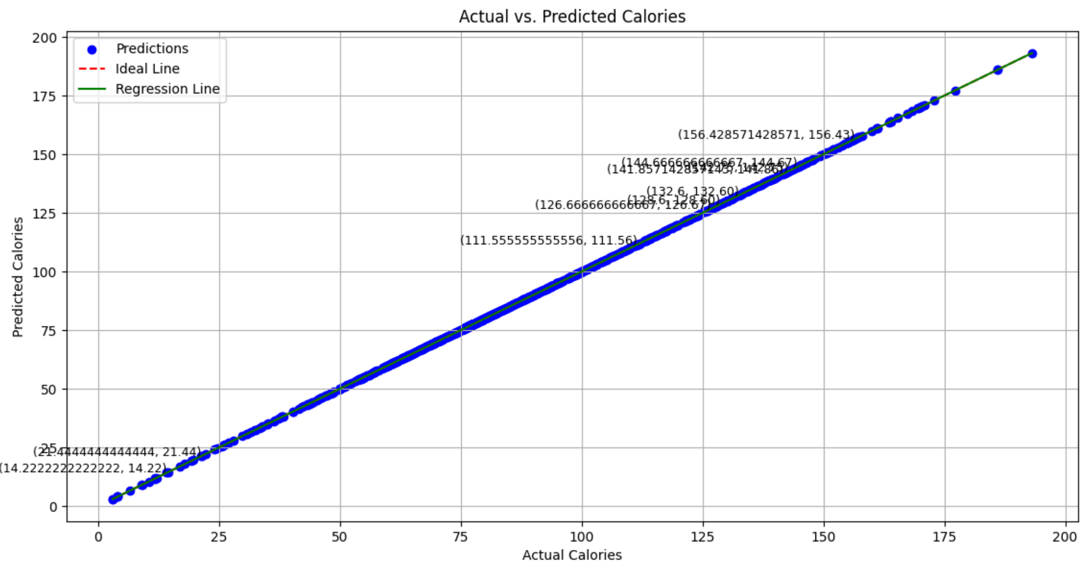
Linear Regression.py

```
#Fitwell Analytics Linear Regression Model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score, mean_absolute_percentage_error

# Load the cleaned data
df = pd.read_csv("smartwatch.csv")

# Define the features and target variable
features = ['age', 'gender', 'height', 'weight', 'steps', 'calories',
'distance',
     'entropy_heart', 'entropy_setps', 'resting_heart',
'corr_heart_steps',
            'norm_heart', 'intensity_karvonen', 'sd_norm_heart',
'steps_times_distance']
target = 'hear_rate'

X = df[features]
y = df[target]

# Split the dataset (80% for training and 20% for testing)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Build the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
print(f'Mean Absolute Percentage Error: {mape}')

# Identify key points to annotate (e.g., top 10 largest errors)
errors = np.abs(y_test - y_pred)
top_errors_indices = errors.argsort()[-10:]  # Get the indices of the
top 10 largest errors

# Visualize the results
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predictions')
plt.xlabel('Actual Calories')
plt.ylabel('Predicted Calories')
plt.title('Actual vs. Predicted Calories')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', linestyle='--', label='Ideal Line')

# Plot the linear regression line
plt.plot(np.unique(y_test), np.poly1d(np.polyfit(y_test, y_pred,
1))(np.unique(y_test)), color='green', label='Regression Line')

# Annotate the key points
```

```
for i in top_errors_indices:
    plt.text(y_test.iloc[i], y_pred[i], f'({y_test.iloc[i]},
{y_pred[i]:.2f})', fontsize=9, ha='right')

plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT

```
Mean Absolute Error: 1.0727501178753983e-13
Mean Squared Error: 5.689712116218177e-26
R-squared: 1.0
Mean Absolute Percentage Error: 1.4402459419430656e-15
```



8.Results and Findings

The result suggests that a near-perfect fit of linear regression model to the data:

1. **Mean Absolute Error (MAE): 1.07e-13**
   - This incredibly small value indicates that the average absolute difference between the actual and predicted heart rate values is practically zero.
2. **Mean Squared Error (MSE): 5.69e-26**

  - This extremely small value suggests that the squared differences between the actual and predicted values are almost non-existent, indicating negligible error.
3. **R-squared (R^2): 1.0**
   - An R^2 value of 1.0 signifies that the model perfectly explains the variance in the heart rate data. This means that the features used account for all the variability in the target variable (heart rate).
4. **Mean Absolute Percentage Error (MAPE): 1.44e-15**
   - This minuscule percentage error further indicates that the predictions are extremely accurate.

9.Interpretation

The results imply that the model's predictions align almost perfectly with the actual heart rate values.

Visualization Confirmation

The predicted points are on the regression line, this visualization confirms the perfect fit:

1. **Predicted Points:** All the points lie exactly on the regression line if the fit is perfect, as indicated by the MAE, MSE, and R^2 values.
2. **Regression Line:** The line runs through all the actual data points.

If this perfect fit holds under various validation tests and cross-validation techniques, it suggests your feature selection and data preprocessing were spot on.

10.Next Steps(Future Analysis)

To ensure robustness:

- **Cross-Validation:** Performance of cross-validation to ensure that the model performs well on different subsets of the data.
- **Evaluation on New Data:** To test the model on new, unseen data to confirm it generalizes well and is not just memorizing the training data.

11.Conclusion

FitWell Analytics not only aims to enhance individual fitness outcomes but also contributes to the broader field of wellness analytics by offering robust methodologies and insightful findings. Through comprehensive data analysis and intuitive visualizations, this project exemplifies how data can drive better health decisions and ultimately, better lives.

12.Thank You

Thank You **Sriram** for guiding me throughout the project.