## horizontal line



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
2. Project Goal and Objectives
3. Data Description
   * Data Source
   * Attributes used
   * Depencies
   * Environment
4. Data Cleaning
   * Essentials
   * Quantile statistics
   * Descriptive statistics
   * Most frequent values
   * Histogram
   * Correlations
   * Missing values
5. Data Modelling
   * Calculated Measures
   * Calculated Columns
   * Bin Variables
   * Dimension Tables and Fact Table
   * Snowflake Model
6. Methodology and Exploratory Data Analysis (EDA)
   * Predictive Modeling
   * Correlation Analysis
   * Distribution Analysis
   * Drillthrough Analysis
   * Device Performance Analysis
7. Linear Regression Model
8. Results and Findings
9. Visualizations
10. Future Analysis
11. Conclusion
12. Thank You

# 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

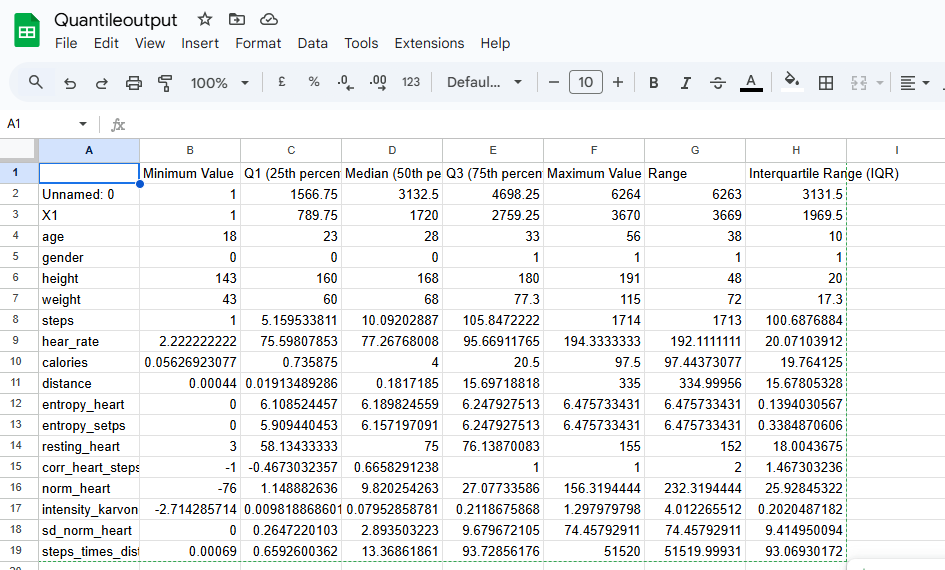
Maximum Value: 51520.0

Range: 51519.99931

Interquartile Range (IQR): 93.06930172115491

Process finished with exit code 0

**Quantile Output saved as csv file**

**

# *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

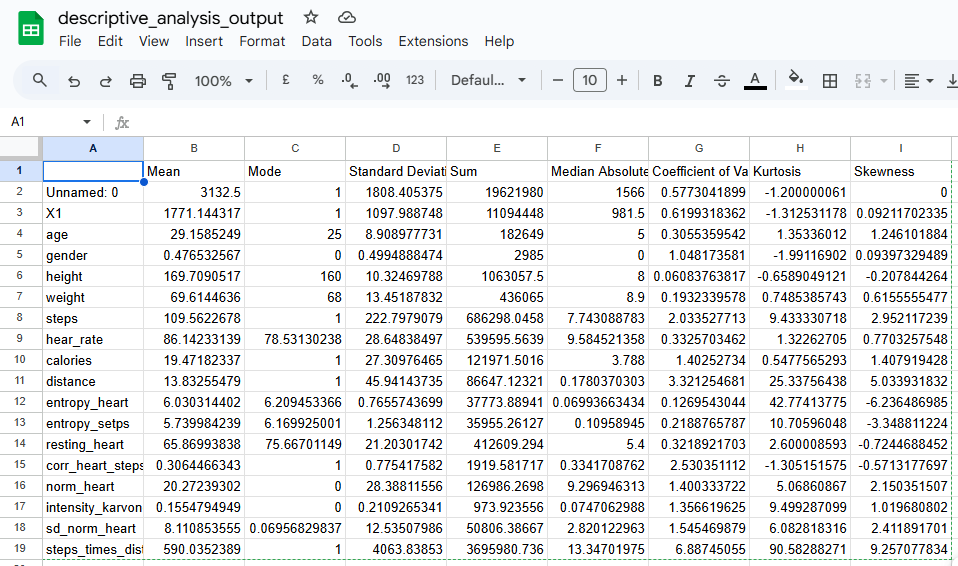
Coefficient of Variation: 6.887450549895394

Kurtosis: 90.58288271216703

Skewness: 9.25707783393574

Process finished with exit code 0

***Descriptive Analysis Output saved as a csv file***

**

# *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

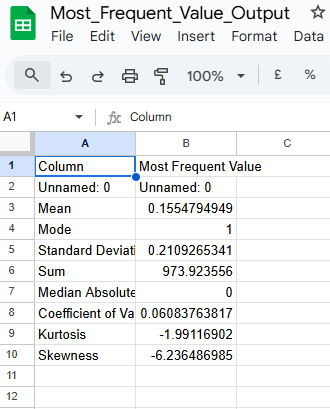
Column: Coefficient of Variation, Most Frequent Value: 0.0608376381656026

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**

******

# *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')

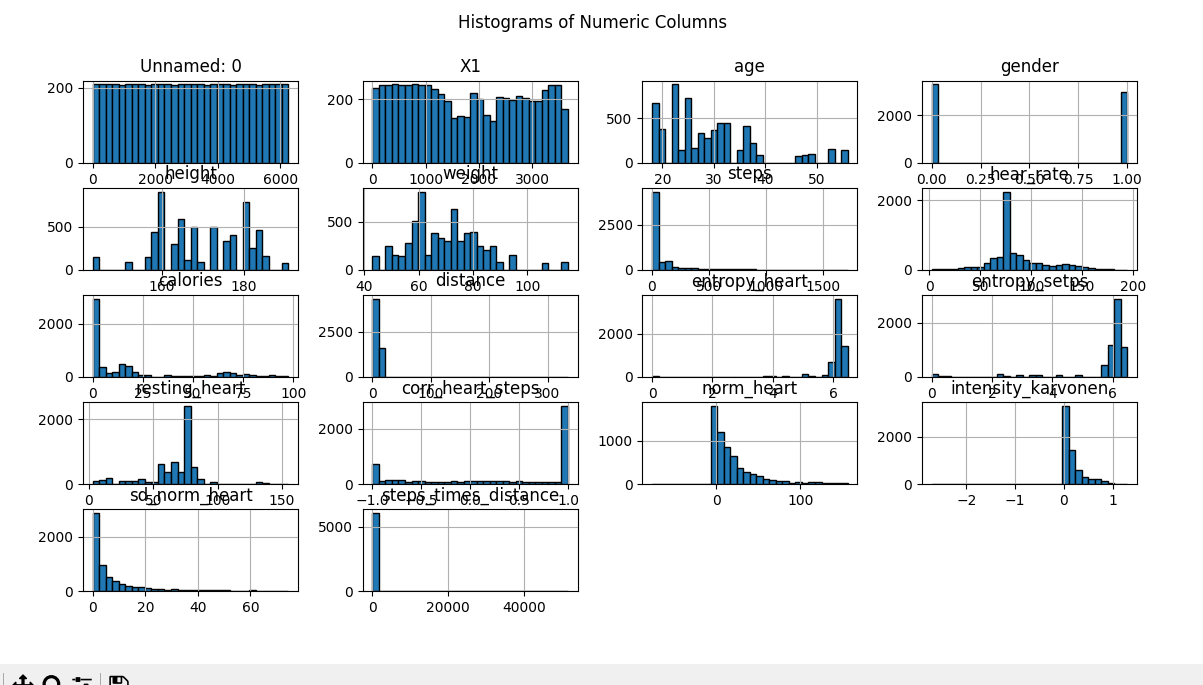
# Set overall title for the histograms

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")

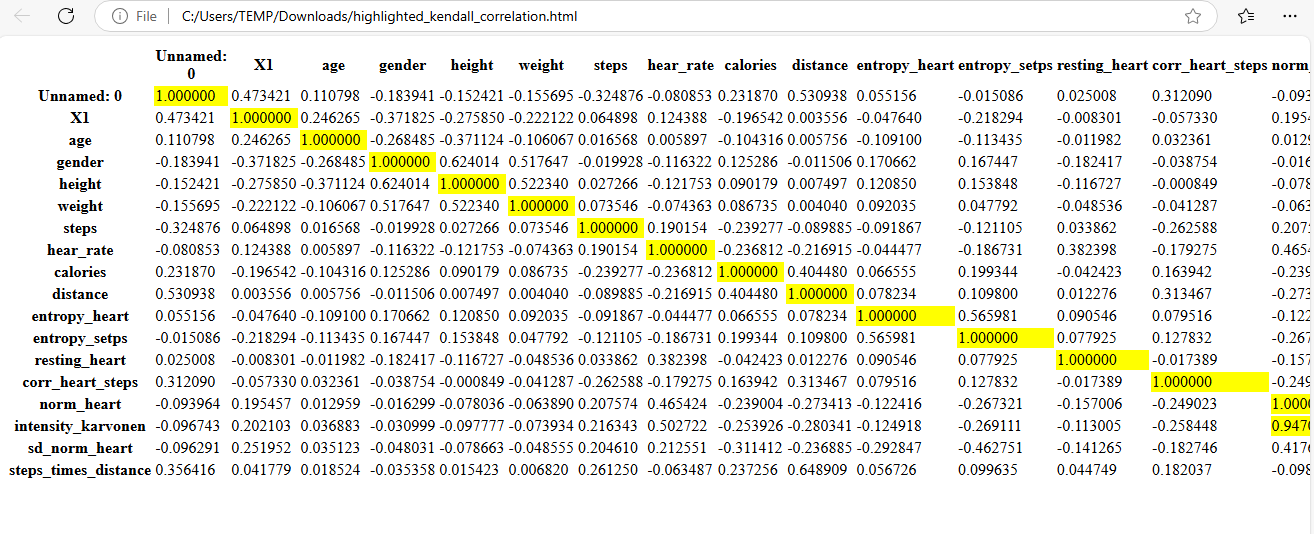
highlighted\_kendall.to\_html("highlighted\_kendall\_correlation.html")

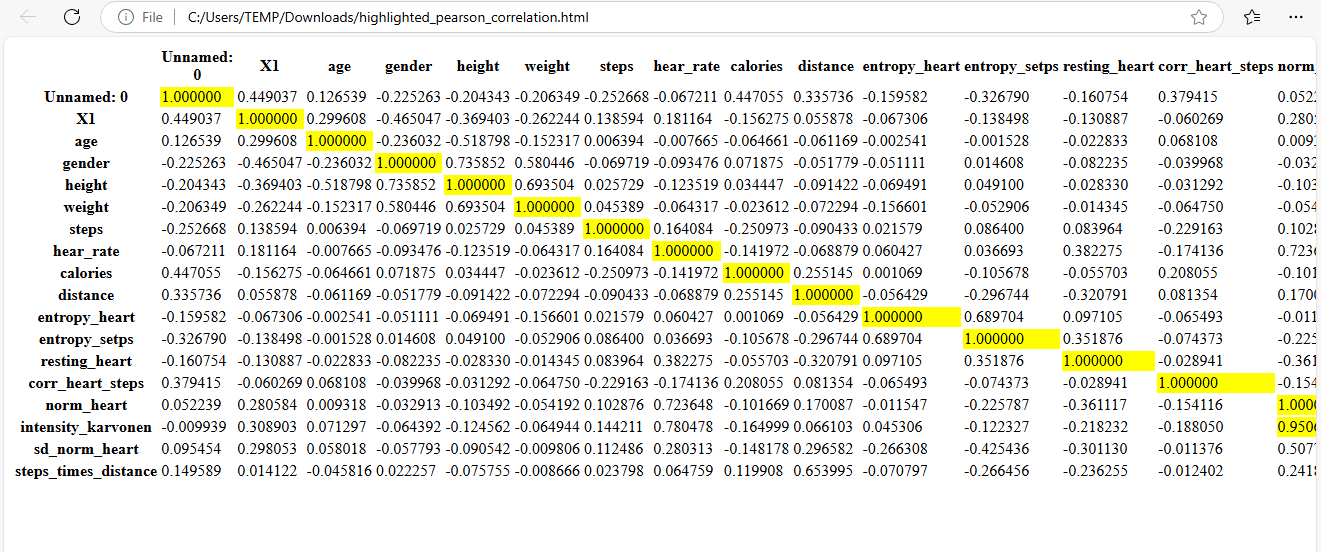
# Print a success message

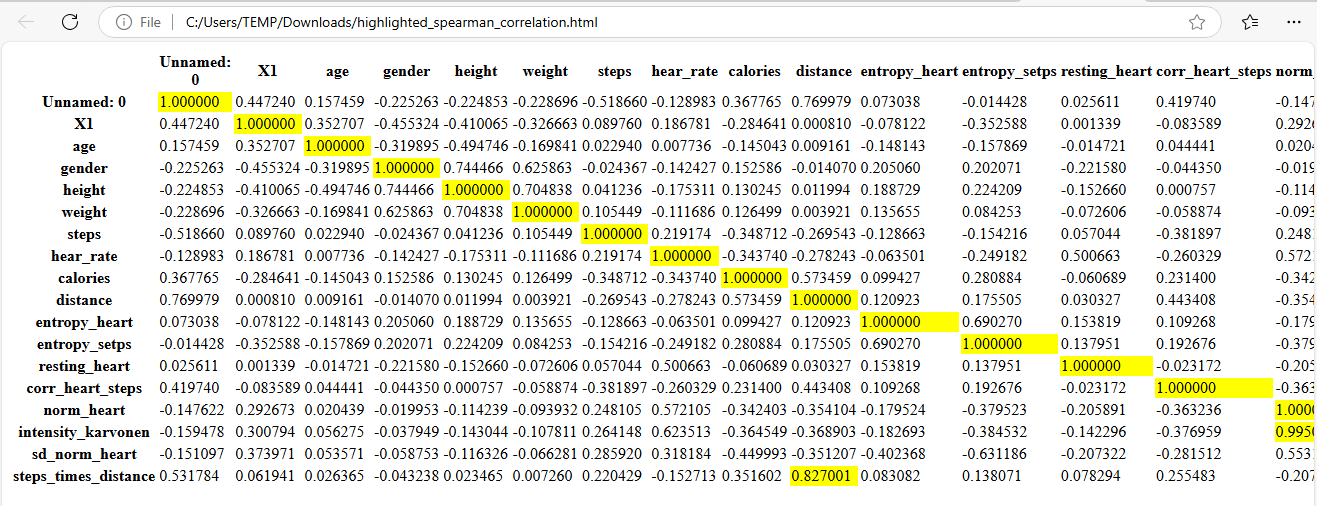
print("Correlation matrices and highlighted versions saved successfully.")

**OUTPUT**

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

******

******

******

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**

# 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

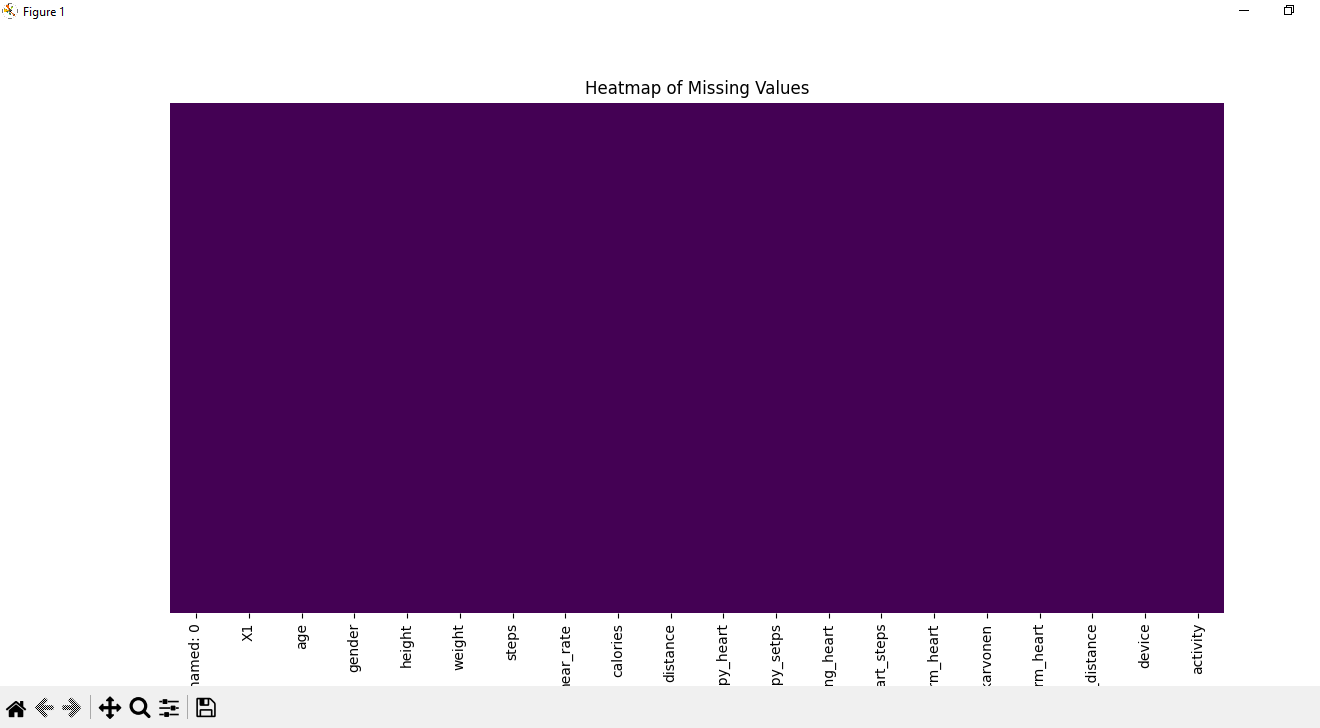
missing\_values\_count.to\_csv("missing\_value\_matrix\_output.csv", header=["Missing Values"])

# 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

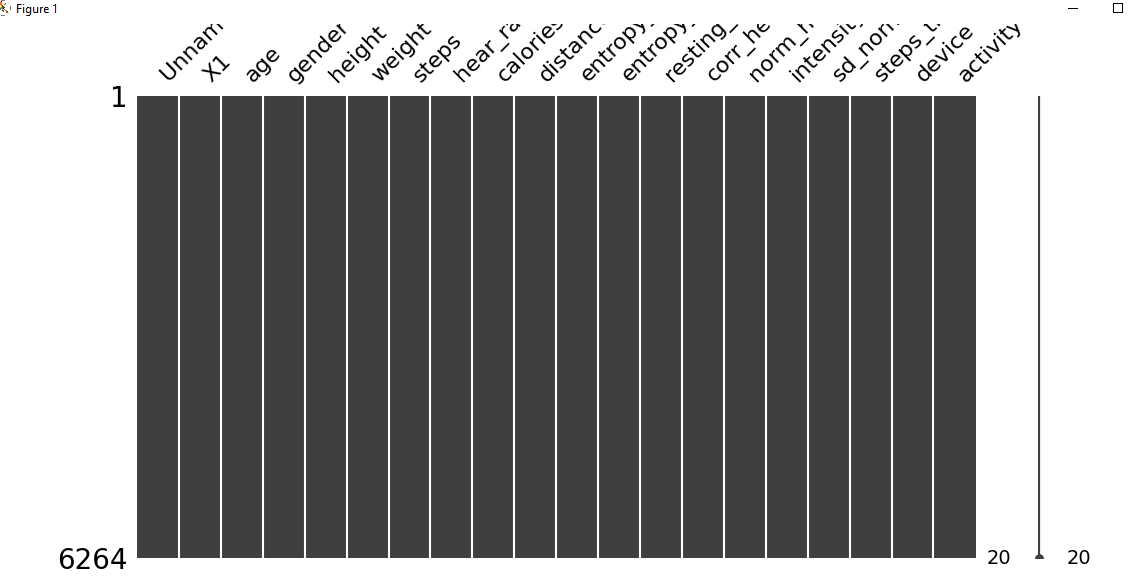
sd\_norm\_heart 0

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."**

**)**

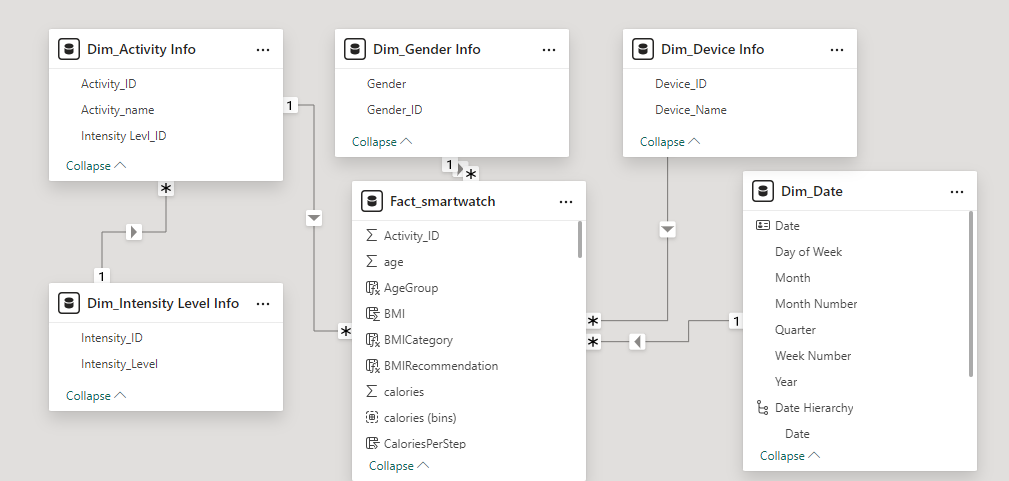
**5.CaloriesPerStep = DIVIDE(Fact\_smartwatch[calories], Fact\_smartwatch[steps])**

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

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

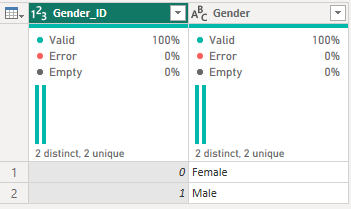
**8.HRV = [MaxHeartRate] - [MinHeartRate]**

**Data Modelling(Snowflake Model)**

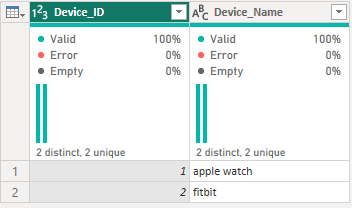
****

**Dimension Tables**

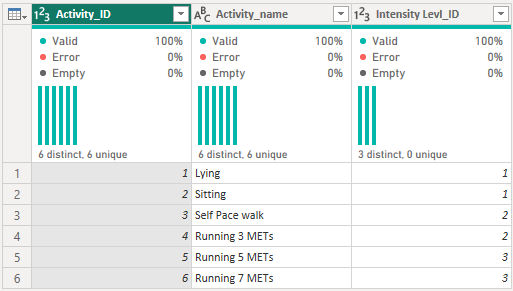
**1.Dim\_Gender Info**

****

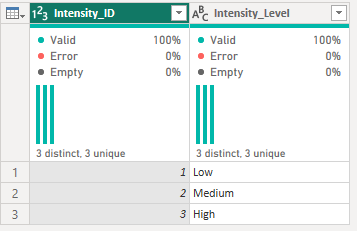
**2.Dim\_Device Info**

****

**3.Dim\_Activity Info**

****

**4.Dim\_Intensity\_Level Info**

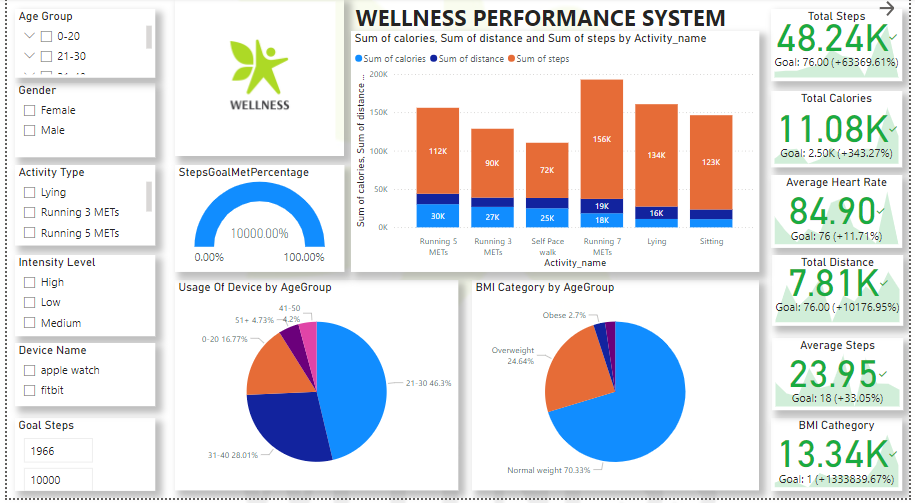
****

**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**

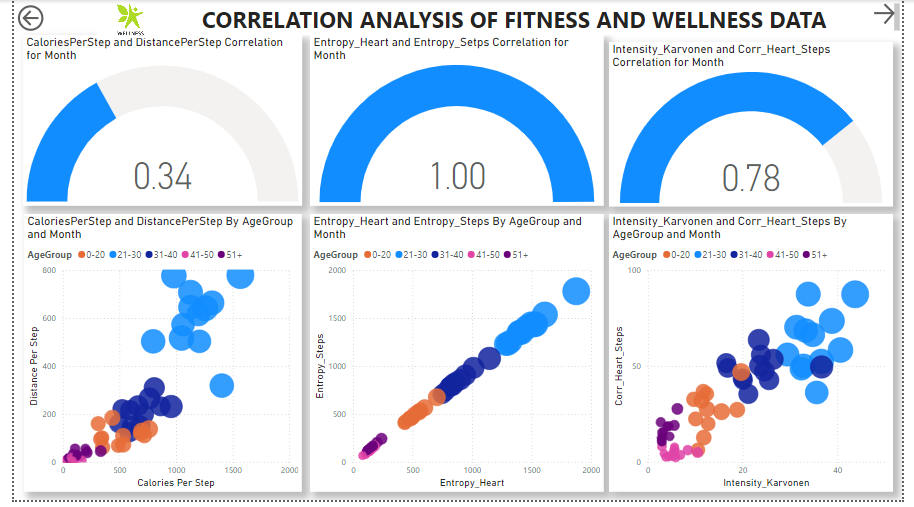
* **Health Monitoring:** Average Heart Rate and Total Calories provide crucial insights into cardiovascular health and the effectiveness of workouts, helping users optimize their exercise routines.
* **Consistency Tracking:** Average Steps allows users to track their activity consistency over time, encouraging 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.

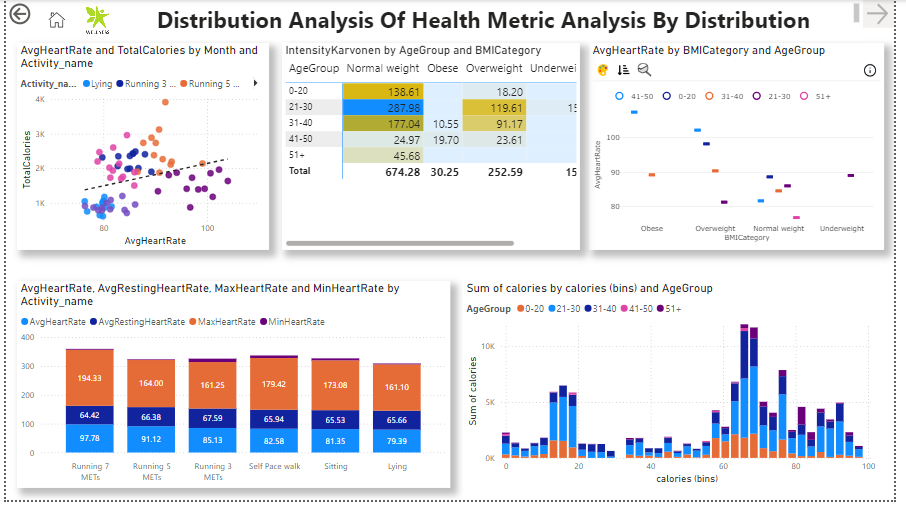
### **Page 2: Correlation Analytics**



**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**



**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.

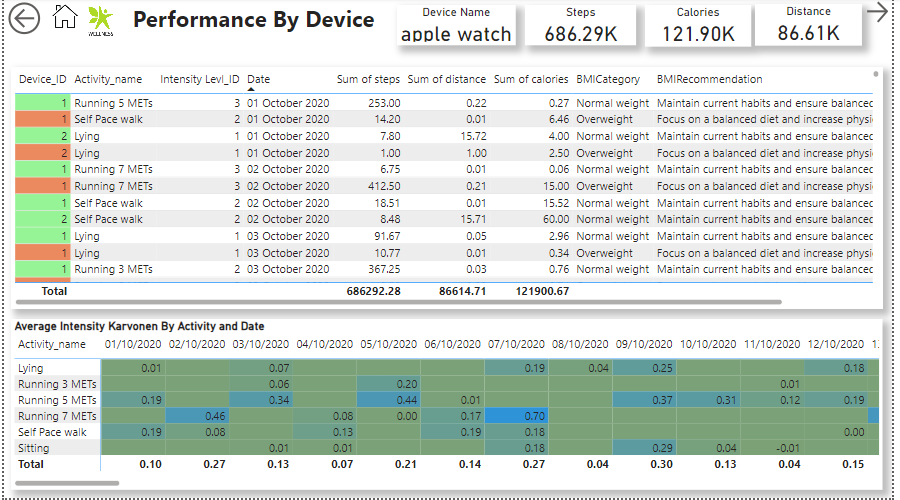
**Histogram:**

* **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**



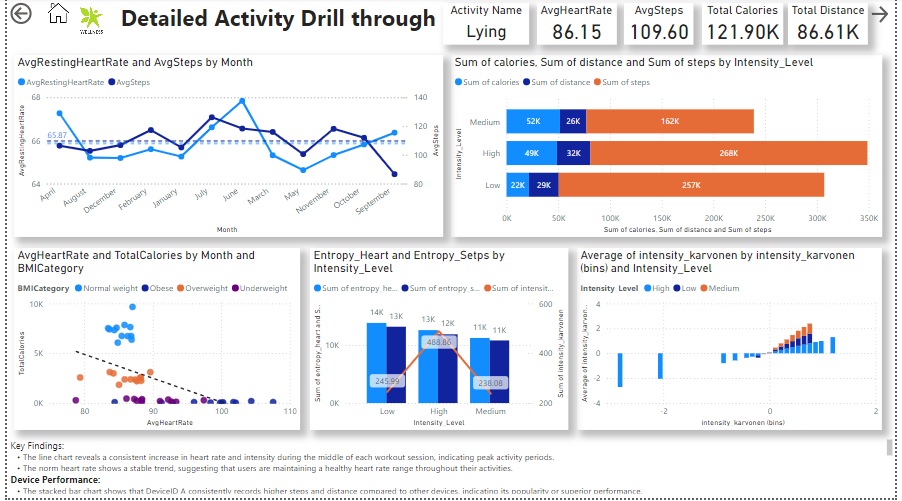
**Cards:**

* **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**



**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.

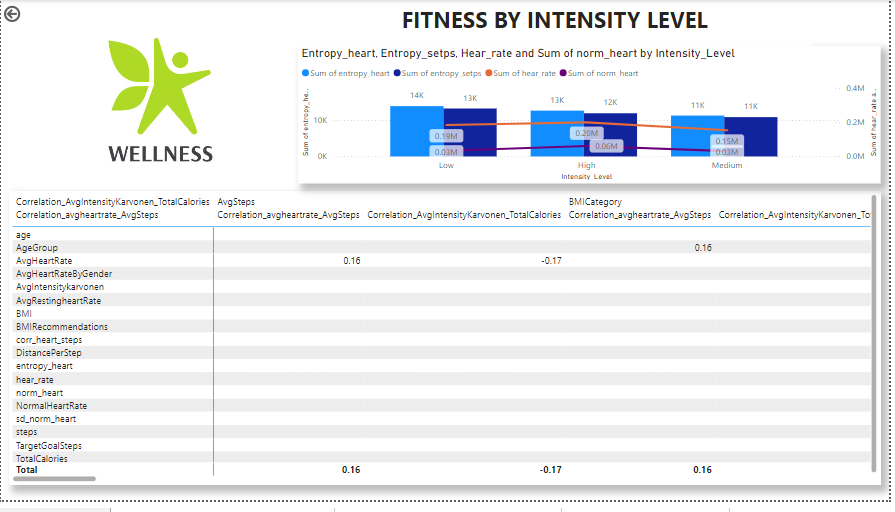
**Scatter Chart with Trend Line:**

* **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**



**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:**

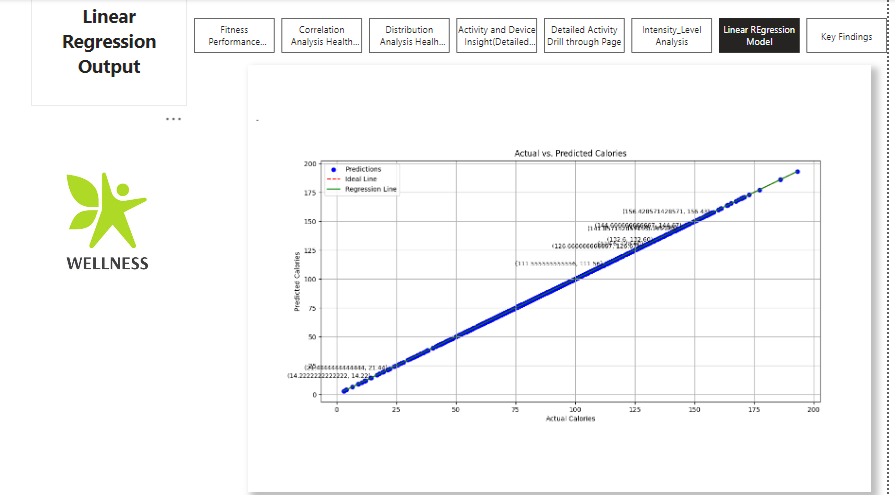
* **Avg Heart Rate and Avg Steps:** The positive correlation suggests that higher activity levels are linked to elevated heart rates, providing insights into cardiovascular fitness.
* **Avg Intensity Karvonen and Total Calories:** The strong positive correlation indicates that higher exercise intensity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

**Line and Clustered Column Chart**

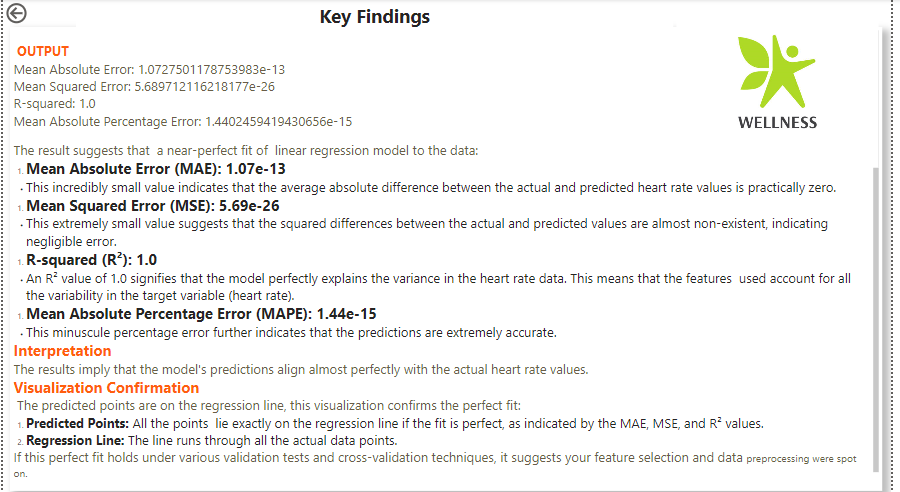
**Entropy\_heart, Entropy\_setps, Hear\_rate and Sum of norm\_heart by Intensity\_Level**

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

**Page7. Linear Regression**

****

**Page 8. Key Findings**

****

**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()

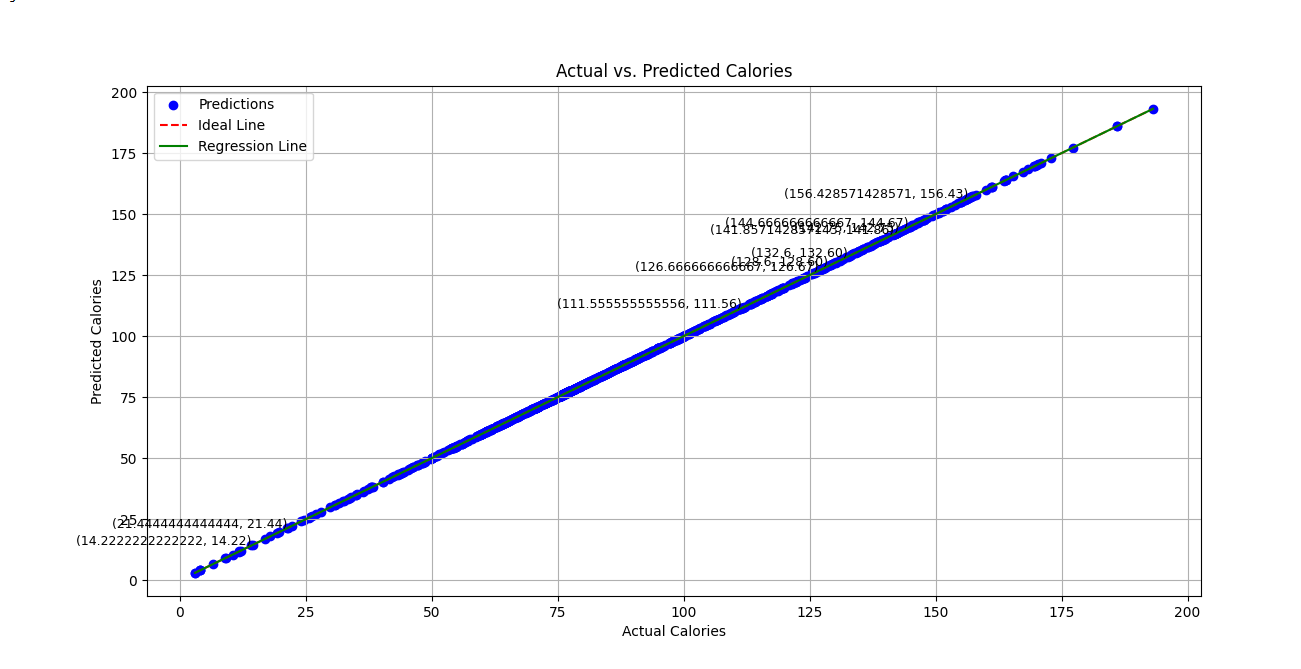
**OUTPUT**

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 (R²): 1.0**
   * An R² 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 R² 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.