

# Python **Cardio** Group Project

---

Liang Weitong	2230006111
Lu Rongde	2230033027
Tang Zhengli	2230033034
Wen han	2230006198

(sort by surname)



# Table Of Contents

---

01 **Prompt word optimization**

02 **GPT answer**

03 **GPT generation  
problem**

04 **Generate initial  
dataset**

05 **Data Set Cleaning  
(Preprocessing)**

06 **Algorithms and  
packages used**

07 **Solutions**



# Prompt word optimization

I would like to generate the problem similar to the one described below. The name and the set of questions should be different. The dataset should be generated using description of dataset. In your output the parts of the solution shouldn't be presented. The description of the problem:  
Project Description

This data set CardioGoodFitness.csv is for customers of the treadmill product(s) of a retail store called Cardio Good Fitness. It contains the following variables Product, Age, Gender, Education, MaritalStatus, Usage, Fitness, Income, and Miles.

Product: the model of the treadmill

Age: in number of years, of the customer

Gender: of the customer

Education: in number of years, of the customer

MaritalStatus: of the customer

Usage: average number of times the customer wants to use the treadmill per week

Fitness: self rated fitness score of the customer (5 - very fit, 1 - very unfit)

Income: of the customer

Miles: expected to run

To understand products, your project focuses on the following two parts:

Process and analyze the data

Create visualizations

Guidelines

You will read instructions and hints carefully and apply your best judgement to answer the questions.

For each question, you need to write Python codes to answer and you need to assign the value to a variable.

For Part 1, you need to store your answers in the answers dictionary

For Part 1, you need to store your answers in the answers dictionary with keys "1", "2", "3", ... "10" and corresponding values (the answer to each question).

```
# You will need to import these two packages.  
import pandas as pd  
import matplotlib.pyplot as plt  
# This is your dataset.  
data = pd.read_csv("CardioGoodFitness.csv")  
data.head()  
Product Age Gender Education MaritalStatus Usage Fitness  
Income Miles
```

0	TM195 18	Male	14	Single	3	4	29562	112
1	TM195 19	Male	15	Single	2	3	31836	75
2	TM195 19	Female	14	Partnered	4	3	30699	66
3	TM195 19	Male	12	Single	3	3	32973	85
4	TM195 20	Male	13	Partnered	4	2	35247	47

1. Questions (50 points)

Run preliminary data analysis on the dataset to answer the following questions:

How many records in the dataset?

How many unique products in the dataset?

What is the largest age difference in the dataset (maximum age - minimum age)?

What is ratio between 'Female' and 'Male' in the column 'Gender' (Female/Male)?

What is the median year of education of the customers in the dataset?

How many of the customers are not single?

How frequent do customers want to use the treadmill per week in average?

What is the percentage of customers self rated to be very fit (in float, e.g., 0.12)?

What is the highest income in the dataset?

How many customers expected to run farther than 150 miles (150 not included)?

Write all your answers in a dictionary named answers. This dictionary

Write all your answers in a dictionary named answers. This dictionary has keys "1", "2", "3", ... "10" and values a1, a2, a3, ... a10 respectively. (e.g., a1 is the variable storing the answer to question 1)

# 1. How many records in the dataset?

# 2. How many unique products in the dataset?

# 3. What is the largest age difference in the dataset (maximum age - minimum age)?

# 4. What is ratio between 'Female' and 'Male' in the column 'Gender' (Female/Male)?

# 5. What is the median year of education of the customers in the dataset?

# 6. How many of the customers are not single?

# 7. How frequent do customers want to use the treadmill per week in average?

# 8. What is the percentage of customers self rated to be very fit (in float, e.g., 0.12)?

# 9. What is the highest income in the dataset?

# 10. How many customers expected to run farther than 150 miles (150 not included)?

# Write all your answer in a dictionary named answers.

# Important: Your dictionary MUST be named 'answers'.

# This dictionary has keys "1", "2", "3", ... "10" and values a1, a2, a3, ... a10 respectively.

# (e.g., a1 is the variable storing the answer to question 1)

### BEGIN SOLUTION

answers = {"1": a1,

"2": a2,

"3": a3,

"4": a4,

"5": a5,

"6": a6,

"7": a7,

"8": a8,

"9": a9,

"10": a10}

### END SOLUTION

2. Visualization (50 points)

You should create plots using the dataset. Your python coding, visualization style, and the appearance of the plots will be manually graded.

# 1. Create a pie chart of products in the dataset.

average?

# 8. What is the percentage of customers self rated to be very fit (in float, e.g., 0.12)?

# 9. What is the highest income in the dataset?

# 10. How many customers expected to run farther than 150 miles (150 not included)?

# Write all your answer in a dictionary named answers.

# Important: Your dictionary MUST be named 'answers'.

# This dictionary has keys "1", "2", "3", ... "10" and values a1, a2, a3, ... a10 respectively.

# (e.g., a1 is the variable storing the answer to question 1)

### BEGIN SOLUTION

answers = {"1": a1,

"2": a2,

"3": a3,

"4": a4,

"5": a5,

"6": a6,

"7": a7,

"8": a8,

"9": a9,

"10": a10}

### END SOLUTION

2. Visualization (50 points)

You should create plots using the dataset. Your python coding, visualization style, and the appearance of the plots will be manually graded.

# 1. Create a pie chart of products in the dataset.

# 2. Create the distribution of self rated fitness score of the customers.

# 3. Draw horizontal barplots of MaritalStatus (on y-axis) and Income (on x-axis) with gender distinction.

Pay attention: You to generate the new project and corresponding dataset. The size of the dataset should be 1000; There need to be some outliers and missing values in the data set. The ten question types in the first part need to be as different as possible from the original question; The answer to each question is fixed, not subjective; In the drawing part, in addition to drawing the three kinds of pictures in the instance, it is also necessary to draw three more advanced pictures, which are heat map, bivariate distribution map (using jointplot()), and box plot. The problem context, data sets, and problems you provide need to be logical and related.

# GPT answer

## Problem Part 1 Data Analysis

### Part 1: Advanced Data Analysis

Answer the following advanced questions. Store answers in a dictionary named `answers`.

1. (Complex Ratio) What is the ratio of customers who report walking fewer than 5,000 steps daily to those walking more than 12,000 steps daily (rounded to 2 decimal places)?
2. (Nested Grouping) For each `ResidenceArea`, calculate the median `AnnualIncome`. Then return the difference between the highest and lowest median incomes across the three areas.
3. (Filtered Mean) Calculate the average `WorkoutHoursWeekly` of female customers who are `Married` and live in `Suburban` areas. Round to 2 decimal places.
4. (Custom Metric) Define a new metric called `ActivityEfficiency`, calculated as  $\text{ActivityEfficiency} = \frac{\text{DailySteps}}{\text{DeviceUsageHours}}$ . What is the average `ActivityEfficiency` of customers younger than 40? Round to 2 decimal places.
5. (Advanced Count) How many customers under the age of 30 have a `FitnessRating` of 4 or 5 and live in `Urban` areas?
6. (Weighted Sum) Compute a weighted sum of `FitnessRating`, weighted by `WorkoutHoursWeekly`, for all customers whose `DailySteps` exceed 10,000.
7. (Proportional Analysis) What percentage of customers live in `Urban` areas and walk fewer than 6,000 steps daily? Provide the answer as a float (e.g., `0.12`).
8. (Conditional Count) How many customers have an `AnnualIncome` in the top 5% but report `FitnessRating` below 3?
9. (Time-to-Fit Prediction) Assuming customers improve their fitness level by  $0.2 \times \text{WorkoutHoursWeekly}$  every week, calculate how many weeks it would take for a customer with a `FitnessRating` of 1 to reach a `FitnessRating` of 5, on average, for all customers with non-missing workout data. Round to the nearest integer.
10. (Correlation Analysis) What is the correlation coefficient between `DailySteps` and `WorkoutHoursWeekly` for `Rural` customers? Round to 3 decimal places.

Store all answers in the dictionary `answers` using the format:

```
answers = {"1": a1, "2": a2, ..., "10": a10}
```

### Challenging Project Description

#### Customer Wellness Tracker Insights

This dataset, `WellnessTracker.csv`, contains survey data of customers using advanced fitness trackers. The dataset includes realistic challenges like missing values, outliers, and multiple categorical and numerical features.

##### Variables

- `TrackerModel`: Model of the fitness tracker
- `Age`: Customer's age in years
- `Gender`: Gender of the customer
- `EducationYears`: Completed education years
- `MaritalStatus`: Customer's marital status (Single, Married, Divorced)
- `DailySteps`: Average daily steps taken
- `FitnessRating`: Self-rated fitness level (1 = very unfit, 5 = very fit)
- `AnnualIncome`: Annual income in USD
- `WorkoutHoursWeekly`: Average weekly workout hours
- `ResidenceArea`: Area of residence (Urban, Suburban, Rural)
- `DeviceUsageHours`: Daily average device usage in hours



给“ChatGPT”发送消息



ChatGPT 也可能犯错。请检查重要信息。

给“ChatGPT”发送消息



ChatGPT 也可能犯错。请检查重要信息。

# GPT answer

## Part 2 Visualizations

### Part 2: Advanced Visualizations

Create the following visualizations. Ensure proper styling, legends, and labels.

1. **Pie Chart:** Percentage distribution of customers by `ResidenceArea`.
2. **Histogram:** Distribution of `ActivityEfficiency`.
3. **Horizontal Bar Plot:** Show the average `AnnualIncome` for each `MaritalStatus`, differentiated by `Gender`.
4. **Heatmap:** Correlation matrix of all numeric variables (`Age`, `DailySteps`, `FitnessRating`, `AnnualIncome`, `WorkoutHoursWeekly`, `DeviceUsageHours`).
5. **Jointplot:** Relationship between `AnnualIncome` and `DailySteps` for Urban customers with KDE overlay.
6. **Box Plot:** Distribution of `DailySteps` by `TrackerModel`, categorized by `Gender`.
7. **Line Plot:** Average `WorkoutHoursWeekly` for customers grouped into age ranges (e.g., 18-25, 26-35, etc.).
8. **Scatter Plot with Regression Line:** Plot `DeviceUsageHours` vs. `DailySteps` with a regression line overlay.
9. **Strip Plot:** Visualize `FitnessRating` by `ResidenceArea`.



给“ChatGPT”发送消息



### Dataset Generation

Below is the Python code for generating the dataset:

python

```
import pandas as pd  
import numpy as np
```

```
# Set random seed for reproducibility  
np.random.seed(42)
```

```
# Dataset size  
n = 1000
```

```
# Generate data  
data = {
```

```
    "TrackerModel": np.random.choice(["Basic", "Advanced", "Pro"], size=n, p=[0.4, 0.4, 0.2]),  
    "Age": np.random.randint(18, 65, size=n),  
    "Gender": np.random.choice(["Male", "Female"], size=n),  
    "EducationYears": np.random.choice([10, 12, 14, 16, 18, np.nan], size=n, p=[0.1, 0.3, 0.2, 0.2, 0.15, 0.05]),  
    "MaritalStatus": np.random.choice(["Single", "Married", "Divorced"], size=n, p=[0.5, 0.3, 0.2]),  
    "DailySteps": np.clip(np.random.normal(loc=7500, scale=3000, size=n), 1000, 20000),  
    "FitnessRating": np.random.choice([1, 2, 3, 4, 5], size=n, p=[0.2, 0.3, 0.3, 0.15, 0.05]),  
    "AnnualIncome": np.random.randint(20000, 200000, size=n),  
    "WorkoutHoursWeekly": np.clip(np.random.normal(loc=3.5, scale=2, size=n), 0, 12),  
    "ResidenceArea": np.random.choice(["Urban", "Suburban", "Rural"], size=n, p=[0.5, 0.3, 0.2]),  
    "DeviceUsageHours": np.clip(np.random.normal(loc=2, scale=0.5, size=n), 0.5, 5)
```

}

```
# Add missing values
```

```
data["WorkoutHoursWeekly"][np.random.choice(n, size=30, replace=False)] = np.nan
```

```
# Add outliers to AnnualIncome
```

```
data["AnnualIncome"][np.random.choice(n, size=10, replace=False)] *= 5
```

```
# Create DataFrame  
df = pd.DataFrame(data)
```

```
# Save dataset
```

```
df.to_csv("WellnessTracker.csv", index=False)  
print("Dataset created: WellnessTracker.csv")
```

## Dataset Generation

ChatGPT 也可能犯错。请核查重要信息。

ChatGPT 也可能犯错。请核查重要信息。

# GPT generation problem



## Complex Ratio

What is the ratio of customers who report walking fewer than 5,000 steps daily to those walking more than 12,000 steps daily (rounded to 2 decimal places)

## Nested Grouping

For each ResidenceArea, calculate the median AnnualIncome. Then return the difference between the highest and lowest median incomes across the three areas.

## Filtered Mean

Calculate the average WorkoutHoursWeekly of female customers who are Married and live in Suburban areas. Round to 2 decimal places.

## Custom Metric

Define a new metric called ActivityEfficiency, calculated as  $\text{ActivityEfficiency} = \frac{\text{DailySteps}}{\text{DeviceUsageHours}}$ . What is the average ActivityEfficiency of customers younger than 40? Round to 2 decimal places.

1

2

3

4

# GPT generation problem



## Advanced Count

How many customers under the age of 30 have a FitnessRating of 4 or 5 and live in Urban areas?

## Weighted Sum

Compute a weighted sum of FitnessRating, weighted by WorkoutHoursWeekly, for all customers whose DailySteps exceed 10,000.

## Proportional Analysis

What percentage of customers live in Urban areas and walk fewer than 6,000 steps daily? Provide the answer as a float (e.g., 0.12).

## Conditional Count

How many customers have an AnnualIncome in the top 5% but report FitnessRating below 3?

5

6

7

8

# GPT generation problem



## Time-to-Fit Prediction

Assume each week a customer's fitness improves by 1.2 times the previous week's improvement. Calculate how many weeks it would take, on average, for a customer to go from a FitnessRating of 1 to 5, using data from customers with non-missing WorkoutHoursWeekly. Round to the nearest integer.

## Correlation Analysis

What is the correlation coefficient between DailySteps and WorkoutHoursWeekly for Rural customers? Round to 3 decimal places.

9

10

# GPT generation problem

## Part 2 Visualizations



### Pie Chart

Percentage distribution of customers by ResidenceArea.



### Heatmap

Correlation matrix of all numeric variables (Age, DailySteps, FitnessRating, AnnualIncome, WorkoutHoursWeekly, DeviceUsageHours).



### Histogram

Distribution of ActivityEfficiency.



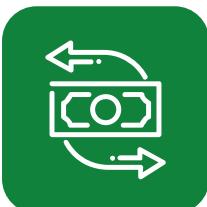
### Jointplot

Relationship between AnnualIncome and DailySteps for Urban customers with KDE overlay.



### Horizontal Bar Plot

Show the average AnnualIncome for each MaritalStatus, differentiated by Gender.



### Box Plot

Distribution of DailySteps by TrackerModel, categorized by Gender.



# Generate initial data set



```
import pandas as pd
import numpy as np

# Set random seed for reproducibility
np.random.seed(42)

# Dataset size
n = 1000

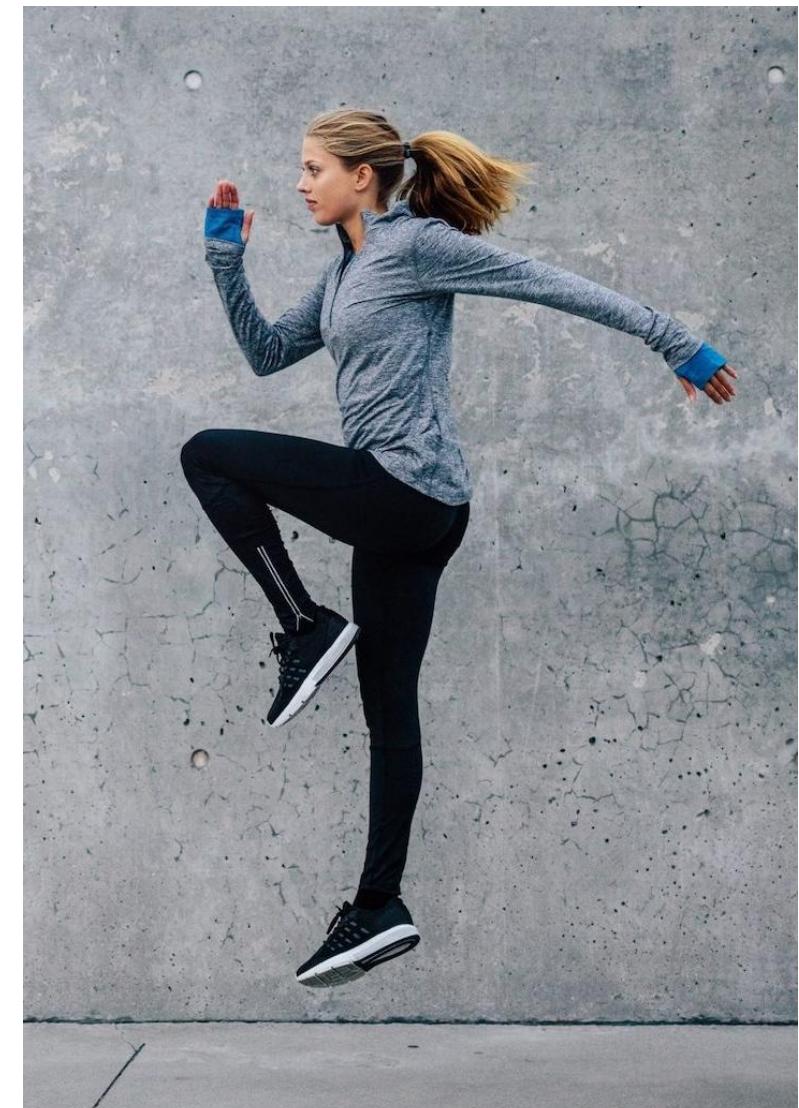
# Generate data
data = {
    "TrackerModel": np.random.choice(["Basic", "Advanced", "Pro"], size=n, p=[0.4, 0.4, 0.2]),
    "Age": np.random.randint(18, 65, size=n),
    "Gender": np.random.choice(["Male", "Female"], size=n),
    "EducationYears": np.random.choice([10, 12, 14, 16, 18, np.nan], size=n, p=[0.1, 0.3, 0.3, 0.2, 0.05, 0.05]),
    "MaritalStatus": np.random.choice(["Single", "Married", "Divorced"], size=n, p=[0.5, 0.4, 0.1]),
    "DailySteps": np.clip(np.random.normal(loc=7500, scale=3000, size=n), 1000, 20000),
    "FitnessRating": np.random.choice([1, 2, 3, 4, 5], size=n, p=[0.2, 0.3, 0.3, 0.15, 0.05]),
    "AnnualIncome": np.random.randint(20000, 200000, size=n),
    "WorkoutHoursWeekly": np.clip(np.random.normal(loc=3.5, scale=2, size=n), 0, 12),
    "ResidenceArea": np.random.choice(["Urban", "Suburban", "Rural"], size=n, p=[0.5, 0.3, 0.2]),
    "DeviceUsageHours": np.clip(np.random.normal(loc=2, scale=0.5, size=n), 0.5, 5)
}

# Add missing values
data["WorkoutHoursWeekly"][np.random.choice(n, size=30, replace=False)] = np.nan

# Add outliers to AnnualIncome
data["DailySteps"][np.random.choice(n, size=10, replace=False)] *= 10

# Create DataFrame
df = pd.DataFrame(data)

# Save dataset
df.to_csv("WellnessTracker.csv", index=False)
print("Dataset created: WellnessTracker.csv")
```



# Data Set Cleaning (Preprocessing)

```
import pandas as pd
import numpy as np
```

## Package

```
# Read the original dataset
df = pd.read_csv("WellnessTracker.csv")
```

```
# Handle missing values and outliers
```

```
# 1. Fill missing values in EducationYears with the median
education_median = df["EducationYears"].median()
df["EducationYears"] = df["EducationYears"].fillna(education_median)
```

```
# 2. Fill missing values in WorkoutHoursWeekly with the mean
workout_mean = df["WorkoutHoursWeekly"].mean()
df["WorkoutHoursWeekly"] = df["WorkoutHoursWeekly"].fillna(workout_mean)
```

```
# 3. Process outliers in DailySteps using the IQR method
Q1 = df["DailySteps"].quantile(0.25) # First quartile (25th percentile)
Q3 = df["DailySteps"].quantile(0.75) # Third quartile (75th percentile)
IQR = Q3 - Q1 # Interquartile range
lower_bound = Q1 - 1.5 * IQR # Lower bound
upper_bound = Q3 + 1.5 * IQR # Upper bound
```

```
# Replace outliers with the lower or upper bounds and convert to integers
df["DailySteps"] = np.where(
    df["DailySteps"] < lower_bound, lower_bound,
    np.where(df["DailySteps"] > upper_bound, upper_bound, df["DailySteps"])
).astype(int) # Ensure the values are integers
```

```
# 4. Round WorkoutHoursWeekly and DeviceUsageHours to two decimal places
df["WorkoutHoursWeekly"] = df["WorkoutHoursWeekly"].round(2)
df["DeviceUsageHours"] = df["DeviceUsageHours"].round(2)
```

```
# Save the cleaned dataset
df.to_csv("WellnessTracker_Cleaned.csv", index=False)
print("Cleaned dataset saved: WellnessTracker_Cleaned.csv")
```

## ● Handle Missing Values

### fillna() function:

#### ➤ EducationYears —— median

【 Avoid extreme values.】

#### ➤ WorkoutHoursWeekly —— mean

【The data is usually evenly distributed.】

## ● Manage Outliers

### ➤ Interquartile range (IQR) for DailySteps

By calculating the quartiles and defining a reasonable range, any values outside this range are replaced with the boundary values.

### ➤ round() function

Format certain columns to two decimal places, ensuring consistency.

# WellnessTracker.csv

**Before**

	A	B	C	D	E	F	G	H	I	J	K
1	TrackerModel	Age	Gender	EducationYears	MaritalStatus	DailySteps	FitnessRating	AnnualIncome	WorkoutHoursWeekly	ResidenceArea	DeviceUsageHours
2	Basic	64	Male		12 Single	1042.8493661724588	5	187567	3.802806095903465	Urban	1.1096871635462553
3	Pro	29	Female		16 Married	10129.366936508806	2	33703	4.096903627962879	Urban	2.751367163152794
4	Advanced	33	Male		14 Single	2815.520332674422	3	181887	1.9348028051628219	Urban	2.388114252362053
5	Advanced	41	Male		Single	12010.259485286997	3	180637	4.91406471922157	Urban	1.7580332775243834
6	Basic	36	Female		14 Divorced	6509.639718100649	3	137491	7.669425842462032	Urban	2.6185904001445044
7	Basic	25	Female		12 Single	6864.99905420647	1	78453	1.4841766091435877	Rural	1.8727619616773292
8	Basic	48	Male		14 Married	5616.798281746484	3	108478	2.8204819847049523	Urban	1.7709874084927217
9	Pro	38	Female		12 Single	6635.883378233771	1	53161	3.9632149483361836	Urban	2.2376674627137456
25	Basic	64	Male		18 Married	10684.1128946194	1	88274	1.8884006902315906	Suburban	1.816381499600357
26	Advanced	49	Female		16 Single	9096.01859646583	2	127853		Suburban	1.789321596114779
27	Advanced	27	Male		16 Single	8584.98318374105	2	24134	3.4112770889768766	Rural	2.310673369984551
28	Basic	33	Female		16 Married	12783.564374602121	2	111428	6.2569347109076165	Urban	2.4740714752360247
29	Advanced	24	Female		14 Married	7499.058982164676	2	59601	2.1569755860389086	Suburban	1.847022784234204
30	Advanced	34	Male		12 Single	3954.3776237375964	3	93839	3.39044327932235	Suburban	1.3817608567038255
31	Basic	40	Male		18 Divorced	8848.42173849692	2	35690	1.6551740303120934	Urban	2.6372201139244518
32	Advanced	43	Male		14 Divorced	13758.141148611341	2	21909	3.5147253628471344	Suburban	1.9523294544016587
33	Basic	38	Male		14 Single	4455.721520442444	2	111239	0.5429061260024022	Urban	2.865160907428584
691	Basic	26	Male		12 Single	4193.190778632434	3	60612	3.484622008557261	Suburban	2.9162806201780063
692	Basic	50	Male		12 Married	8542.684841410655	3	33450	2.7242484362750607	Suburban	1.2348604320351062
693	Advanced	58	Female		18 Divorced	7760.09309824319	5	29491	6.1121444077432585	Urban	1.9387820051263012
694	Advanced	42	Female		12 Married	85670.06845689687	2	137026	2.51687612196214	Urban	1.5457487442439044
695	Basic	30	Male		14 Single	8074.7321885822685	3	94804	0.6745484821988876	Urban	2.302053727719522
696	Basic	26	Male		12 Single	9018.722967917778	1	113293	2.466306233652343	Suburban	1.1914140549772276

	A	B	C	D	E	F	G	H	I	J	K
1	TrackerModel	Age	Gender	EducationYears	MaritalStatus	DailySteps	FitnessRating	AnnualIncome	WorkoutHoursWeekly	ResidenceArea	DeviceUsageHours
2	Basic	64	Male		12 Single	1042	5	187567		3.8 Urban	1.11
3	Pro	29	Female		16 Married					4.1 Urban	2.75
4	Advanced	33	Male		14 Single					1.93 Urban	2.39
5	Advanced	41	Male		14 Single					4.91 Urban	1.76
6	Basic	36	Female		14 Divorced					7.67 Urban	2.62
7	Basic	25	Female		12 Single					1.48 Rural	1.87
8	Basic	48	Male		14 Married	5616	3	108478		2.82 Urban	1.77
9	Pro	38	Female		12 Single	6635	1	53161		3.96 Urban	2.24
10	Advanced	34	Female		14 Married	11755	3	134296		5.07 Urban	1.47
25	Basic	64	Male								
26	Advanced	49	Female								
27	Advanced	27	Male								
28	Basic	33	Female								
29	Advanced	24	Female								
30	Advanced	34	Male		12 Single	3954	3	93889		3.89 Suburban	1.82
31	Basic	40	Male		18 Divorced	8848	2	35690		3.47 Suburban	1.79
32	Advanced	43	Male		14 Divorced	13758	2	21909		3.41 Rural	2.31
33	Basic	38	Male		14 Single	4455	2	111239		6.26 Urban	2.47
691	Basic	26	Male		12 Single	4193	3	60612		2.16 Suburban	1.85
692	Basic	50	Male		12 Married	8542				3.39 Suburban	1.38
693	Advanced	58	Female		18 Divorced	7760				1.66 Urban	2.64
694	Advanced	42	Female		12 Married	15541				3.51 Suburban	1.95
695	Basic	30	Male		14 Single	8074				0.54 Urban	2.87
696	Basic	26	Male		12 Single	9018					
697	Advanced	30	Female		10 Married	11841					

**fillna() function:**  
 ➤ **EducationYears -- median**  
 [ Avoid extreme values.]

**fillna() function:**  
 ➤ **WorkoutHoursWeekly -- mean**  
 [The data is usually evenly distributed.]

➤ **Interquartile range (IQR) for DailySteps**  
 By calculating the quartiles and defining a reasonable range, any values outside this range are replaced with the boundary values.

# Algorithms & packages used

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

# Load the dataset
data = pd.read_csv("WellnessTracker_Cleaned.csv")
```



## Pandas

- `pd.read_csv()`: Loads the dataset from a CSV file.
- `groupby()`: Groups data for aggregated calculations, such as median income.
- `fillna()`: Fills missing values with specified statistics (mean, median).
- `value_counts()`: Counts unique values for pie charts.

## NumPy

- `np.where()`: Used to conditionally replace values in an array (e.g., handling outliers).
- Array operations for calculations like averages and ratios.

## Matplotlib

- `plt.hist()`: Creates histograms to visualize data distributions.
- `plt.plot()`: Used for plotting lines and scatter plots.
- `plt.show()`: Displays the plots.

## Seaborn

- `sns.boxplot()`: Creates box plots for visualizing distributions across categories.
- `sns.heatmap()`: Displays correlation matrices as heatmaps.
- `sns.jointplot()`: Shows the relationship between two variables with density estimation.



# Solution 1

```
# 1. (Complex Ratio) What is the ratio of customers who report walking fewer than 5,000 steps daily  
# to those walking more than 12,000 steps daily (rounded to 2 decimal places)?  
# Calculate the number of customers walking less than 5,000 steps and those walking more than 12,000 steps,  
# then return the ratio rounded to two decimal places.  
# (Round to two decimal places)  
  
### BEGIN SOLUTION  
less_5000 = len(data[data["DailySteps"] < 5000]) # Rows with DailySteps < 5000  
more_12000 = len(data[data["DailySteps"] > 12000]) # Rows with DailySteps > 12000  
  
try:  
    a1 = round(less_5000 / more_12000, 2) # Calculate ratio  
except ZeroDivisionError:  
    a1 = None # Handle division by zero  
### END SOLUTION  
  
answers = {"1": a1}  
  
print(answers)  
{'1': 2.73}
```

## Complex Ratio

This code calculates the ratio of customers walking fewer than 5,000 steps to those walking more than 12,000.

- The dataset is filtered with  
`data["DailySteps"] < 5000` and  
`data["DailySteps"] > 12000`
- The `len()` function counts each group.
- The ratio is then calculated and rounded with `round()`
- Use `try...except` to handle the known exception type which is `ZeroDivisionError` (here we set the answer to be `None` if `more_12000` is 0)



# Solution 2

```
# 2. (Nested Grouping) For each ResidenceArea, calculate the median AnnualIncome.  
# Then return the difference between the highest and lowest median incomes across the three areas.  
# (No rounding needed)  
  
### BEGIN SOLUTION  
median_incomes = data.groupby("ResidenceArea", observed=True)[ "AnnualIncome" ].median()  
a2 = median_incomes.max() - median_incomes.min() # Difference  
### END SOLUTION  
  
# Store the answer in a dictionary  
answer = {"2": a2}  
  
print(answer)  
{'2': 7257.5}
```

## Nested Grouping

The code calculates the difference between the highest and lowest median incomes across residential areas.

- Grouping is done with `groupby ("ResidenceArea")`
- Medians are computed with `.median()`
- The difference is found using `.max()` and `.min()`



# Solution 3

```
# 3. (Filtered Mean) Calculate the average WorkoutHoursWeekly of female customers who are Married  
# and live in Suburban areas. Round to 2 decimal places.  
# (Round to two decimal places)  
  
### BEGIN SOLUTION  
filtered_data = data[(data["Gender"] == "Female") &  
                     (data["MaritalStatus"] == "Married") &  
                     (data["ResidenceArea"] == "Suburban")]  
a3 = round(filtered_data["WorkoutHoursWeekly"].mean(), 2) # Average hours  
### END SOLUTION  
  
# Store the answer in a dictionary  
answer = {"3": a3}  
  
print(answer)  
{'3': 3.39}
```

## Filtered Mean

This calculates the average workout hours for married female customers in suburban areas.

- Data is filtered using conditions combined with **&**
- The mean is calculated with **mean()** and rounded with **round ()**



# Solution 4

```
# 4. (Custom Metric) Define a new metric called ActivityEfficiency, calculated as
# ActivityEfficiency = DailySteps / DeviceUsageHours.
# What is the average ActivityEfficiency of customers younger than 40? Round to 2 decimal places.
# (Round to two decimal places)

### BEGIN SOLUTION
data["ActivityEfficiency"] = data["DailySteps"] / data["DeviceUsageHours"]
a4 = round(data[data["Age"] < 40]["ActivityEfficiency"].mean(), 2)
### END SOLUTION

# Store the answer in a dictionary
answer = {"4": a4}

print(answer)
{'4': 4039.18}
```

## Custom Metric

This code defines a new metric called `ActivityEfficiency`, which evaluates how efficiently customers are active relative to their device usage hours.

- Calculate `ActivityEfficiency` as the ratio of `DailySteps` to `DeviceUsageHours` and adds it as a new column in the dataset using  
`data["ActivityEfficiency"] = data["DailySteps"] / data[ "DeviceUsageHours"]`
- Filter Data to select customers whose age is below 40 using `data[data ["Age"] < 40]`
- Calculate the Average with `mean()`
- Round to two decimal places using `round()`.



# Solution 5

```
# 5. (Advanced Count) How many customers under the age of 30 have a FitnessRating of 4 or 5 and live in Urban areas?  
# (No rounding needed)  
  
### BEGIN SOLUTION  
a5 = len(data[(data["Age"] < 30) &  
             (data["FitnessRating"] >= 4) &  
             (data["ResidenceArea"] == "Urban")])  
### END SOLUTION  
  
# Store the answer in a dictionary  
answer = {"5": a5}  
  
print(answer)  
  
{'5': 26}
```

## Advanced Count

This code answers a straightforward question: how many customers under the age of 30 have a FitnessRating of 4 or 5 and live in urban areas?

- The code uses multiple conditions to narrow down the dataset:
  - ✓ `(data["Age"] < 30)`.
  - ✓ `(data["FitnessRating"] >= 4)`.
  - ✓ `(data["ResidenceArea"] == "Urban")`.These conditions are combined with `&`
- The filtered dataset is passed to the `len()` function, which counts the number of matching rows.



# Solution 6

```
# 6. (Weighted Sum) Compute a weighted sum of FitnessRating, weighted by WorkoutHoursWeekly,  
# for all customers whose DailySteps exceed 10,000.  
# (Round to two decimal places)  
  
### BEGIN SOLUTION  
high_steps = data[data["DailySteps"] > 10000]  
  
# Compute the weighted sum normalized by the total WorkoutHoursWeekly.  
numerator = (high_steps["FitnessRating"] * high_steps["WorkoutHoursWeekly"]).sum()  
denominator = high_steps["WorkoutHoursWeekly"].sum()  
  
# To avoid division by zero, check if the denominator is greater than zero.  
a6 = round(numerator / denominator, 2) if denominator > 0 else 0  
### END SOLUTION  
  
answers = {"6": a6}  
  
print(answers)  
{'6': 2.56}
```

## Weighted Sum

This code calculates a weighted sum of FitnessRating, weighted by WorkoutHoursWeekly, for customers who walk more than 10,000 steps daily.

- `data["DailySteps"] > 10000`
- Divide the sum of the product of FitnessRating and WorkoutHoursWeekly by the sum of WorkoutHoursWeekly to compute the normalized weighted sum. (here we use `.sum()` to calculate the sum.)
- We use `if...else` to set the solution to be 0 if denominator is 0.
- Round the result to two decimal places.



# Solution 7

```
# 7. (Proportional Analysis) What percentage of customers live in Urban areas and walk fewer than 6,000 steps daily?  
# Provide the answer as a float (e.g., 0.12).  
# (Round to four decimal places)  
  
### BEGIN SOLUTION  
urban_customers = data[data["ResidenceArea"] == "Urban"] # Filter urban customers  
  
try:  
    a7 = round(len(urban_customers[urban_customers["DailySteps"] < 6000]) / len(urban_customers), 4) # Calculate ratio  
except ZeroDivisionError:  
    a7 = None # Handle zero division  
### END SOLUTION  
answers = {"7": a7}  
  
print(answers)  
{'7': 0.3235}
```

## Proportional Analysis

This code calculates the percentage of customers living in urban areas who walk fewer than 6,000 steps daily.

- Filter Urban Customers:  
`[data["ResidenceArea"] == "Urban"]`
- The proportion is calculated by dividing the count of low-step customers by the total number of urban customers (`(len(urban_customers))`). Round the result.
- Use `try...except` to handle the known exception type which is `ZeroDivisionError` (here we set the answer to be `None` if `len(urban_customers)` is 0)



# Solution 8

```
# 8. (Conditional Count) How many customers have an AnnualIncome in the top 5% but report FitnessRating below 3?  
# (No rounding needed)  
  
### BEGIN SOLUTION  
income_threshold = data["AnnualIncome"].quantile(0.95)  
a8 = len(data[(data["AnnualIncome"] > income_threshold) & (data["FitnessRating"] < 3)])  
### END SOLUTION  
  
# Store the answer in a dictionary  
answer = {"8": a8}  
  
print(answer)  
{'8': 22}
```

## Conditional Count

This code identifies how many customers have an `AnnualIncome` in the top 5% while reporting a `FitnessRating` below 3.

- The `quantile(0.95)` function calculates the value that separates the top 5% of `AnnualIncome` values from the rest.
- Filter Customers:
  - ✓ `(data ["AnnualIncome"] > income_threshold)`
  - ✓ `data["FitnessRating"] < 3`
- The filtered dataset is passed to the `len()` function to count how many rows meet both criteria.



# Solution 9

```
# 9. (Time-to-Fit Prediction)
# Assuming a customer improves their fitness level by multiplying their current FitnessRating by 1.2 every week,
# calculate how many weeks it would take for a customer with a FitnessRating of 2 to reach or exceed a FitnessRating of 5.
# Round to the nearest integer.

### BEGIN SOLUTION
initial_value = 2 # Initial value
growth_rate = 1.2 # Growth rate (multiplier)
target_value = 5 # Target value
a9 = 0 # Step counter
current_value = initial_value # Current value
# Loop until the current value reaches or exceeds the target value
while current_value < target_value:
    current_value *= growth_rate # Increase current value by the growth rate
    a9 += 1 # Increment the step counter
### END SOLUTION

# Store the answer in a dictionary
answer = {"9": a9}

print(answer)

{'9': 6}
```

## Time-to-fit Prediction

This code estimates how many weeks it would take for a customer with a FitnessRating of 2 to improve to 5 or more, assuming their fitness level increases by 20% each week.

- Simulate Weekly Growth
- ✓ A **while loop** runs as long as the current fitness level ( `current_value` ) is less than the target ( 5 ).
- ✓ The current fitness level is multiplied by the growth rate ( `current_value *= growth_rate` ), simulating weekly improvement.
- ✓ The week counter is incremented each time the loop runs ( `a9 += 1` ).



# Solution 10

```
# 10. (Correlation Analysis) What is the correlation coefficient between DailySteps and WorkoutHoursWeekly  
# for Rural customers?  
# (Round to four decimal places)  
  
### BEGIN SOLUTION  
rural_data = data[data["ResidenceArea"] == "Rural"]  
a10 = round(rural_data["DailySteps"].corr(rural_data["WorkoutHoursWeekly"]), 4)  
### END SOLUTION  
  
# Store all answers in a dictionary  
answers = {"10": a10}  
  
print(answers)  
  
{'10': -0.0563}
```

## Correlation Analysis

This code calculates the correlation coefficient between DailySteps and WorkoutHoursWeekly for customers living in rural areas.

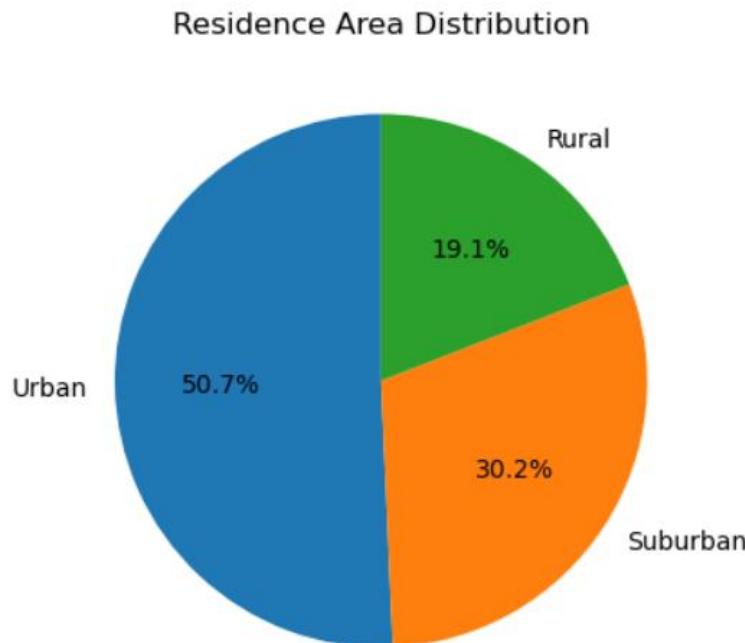
- Filter Rural Customers:  
**[data["ResidenceArea"] == "Rural"]**
- It calculates the correlation coefficient between the DailySteps and WorkoutHoursweekly columns of the filtered data using the **.corr()** method.
- Round the Result

# Visualization 1

```
# PART 2: Advanced Visualizations
```

```
# 1. Pie chart: Percentage distribution of customers by ResidenceArea.
```

```
### BEGIN SOLUTION
data["ResidenceArea"].value_counts().plot(kind='pie', autopct='%1.1f%%', startangle=90)
plt.title("Residence Area Distribution")
plt.ylabel("")
plt.show()
### END SOLUTION
```



## Pie chart

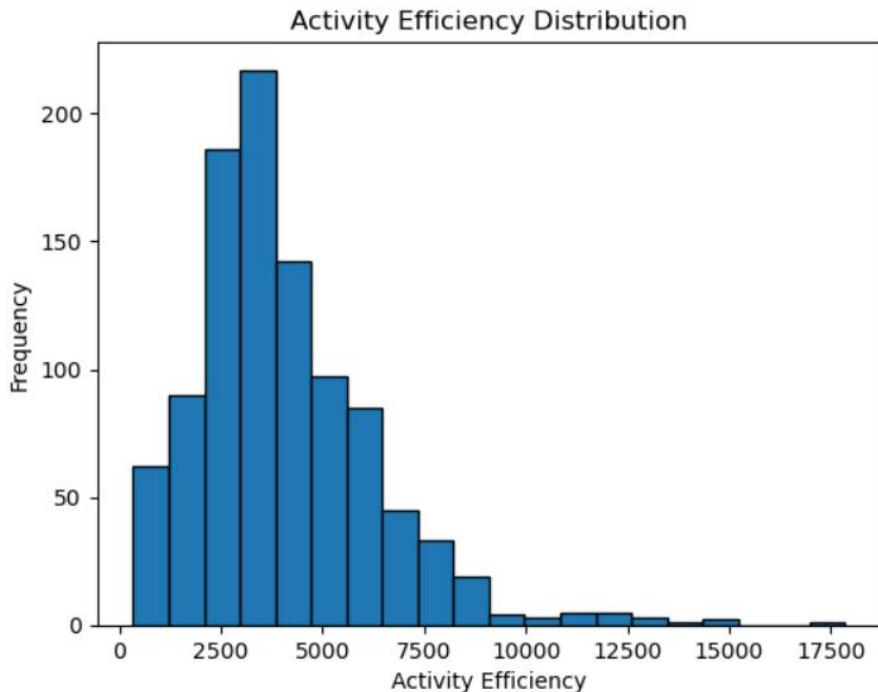
Percentage distribution of customers by ResidenceArea.

- **value\_counts()**: count the occurrences of each residence area in the "ResidenceArea" column.
- **plot(kind='pie')**: generate the pie chart.
  - ✓ `autopct='%1.1f%%'` is used to display the percentage of each category
  - ✓ `startangle=90` sets the starting angle
- **plt.title()** adds the title to the chart, which is "Residence Area Distribution"
- **plt.ylabel ("")** removes the y-axis label

# Visualization 2

```
# 2. Histogram: Distribution of ActivityEfficiency.
```

```
### BEGIN SOLUTION
plt.hist(data["ActivityEfficiency"].dropna(), bins=20, edgecolor='black')
plt.title("Activity Efficiency Distribution")
plt.xlabel("Activity Efficiency")
plt.ylabel("Frequency")
plt.show()
### END SOLUTION
```



## Histogram

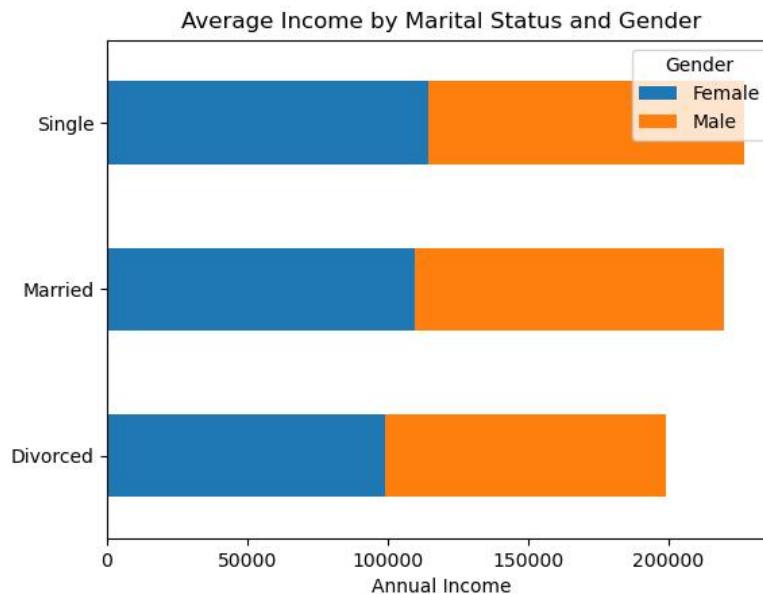
Distribution of ActivityEfficiency.

- Remove any missing values using **dropna()**
- **plt.hist()** is used to create the histogram.
  - ✓ The parameter **bins=20** divides the data into 20 bins
  - ✓ **edgecolor='black'** adds a black border around each bar.
- **plt.title()** adds title "Activity Efficiency Distribution"
- **plt.xlabel ("Activity Efficiency")**
- **plt.ylabel ("Frequency")**

# Visualization 3

```
# 3. Horizontal Bar Plot: Average AnnualIncome by MaritalStatus, differentiated by Gender.
```

```
### BEGIN SOLUTION
income_plot_data = data.groupby(["MaritalStatus", "Gender"], observed=True)[["AnnualIncome"]].mean().unstack()
income_plot_data.plot(kind="barh", stacked=True)
plt.title("Average Income by Marital Status and Gender")
plt.xlabel("Annual Income")
plt.ylabel("Marital Status")
plt.legend(title="Gender")
plt.show()
### END SOLUTION
```



## Horizontal Bar Plot

Show the average AnnualIncome for each MaritalStatus, differentiated by Gender.

- The data is grouped by "MaritalStatus" and "Gender" using **groupby()**, and the **mean()** "AnnualIncome" is calculated for each group.
- **unstack()** is used to turn the "Gender" level into columns, making the data easier to visualize.
- **plot (kind="barh", stacked=True)**: create a stacked horizontal bar plot
- **plt.title()** adds title "Average Income by Marital Status and Gender"
- **plt.xlabel ("Activity Efficiency")**
- **plt.ylabel ("Frequency")**
- **plt.legend()** adds a legend to indicate the gender distinction.

# Visualization 4



## Heatmap

Correlation matrix of all numeric variables (Age, DailySteps, FitnessRating, AnnualIncome, WorkoutHoursWeekly, DeviceUsageHours).  
(with determining statistical significance)

```
from scipy.stats import t

sns.set_theme(style="whitegrid", font_scale=1.2)

data = pd.read_csv("WellnessTracker_Cleaned.csv")

numeric_features = ["Age", "DailySteps", "FitnessRating", "AnnualIncome", "WorkoutHoursWeekly", "DeviceUsageHours"]
correlation_matrix = pd.DataFrame(index=numeric_features, columns=numeric_features, dtype=float)
t_value_matrix = pd.DataFrame(index=numeric_features, columns=numeric_features, dtype=float)
p_value_matrix = pd.DataFrame(index=numeric_features, columns=numeric_features, dtype=float)

n = len(data)

for var1 in numeric_features:
    for var2 in numeric_features:
        if var1 == var2:
            correlation_matrix.loc[var1, var2] = 1.0
            t_value_matrix.loc[var1, var2] = np.nan
            p_value_matrix.loc[var1, var2] = 0.0
        else:
            r = data[[var1, var2]].corr().iloc[0, 1]
            correlation_matrix.loc[var1, var2] = r
            t_val = (r * np.sqrt(n - 2)) / np.sqrt(1 - r**2)
            t_value_matrix.loc[var1, var2] = t_val
            p_val = 2 * (1 - t.cdf(abs(t_val), df=n - 2))
            p_value_matrix.loc[var1, var2] = p_val

def plot_heatmap(data_matrix, title, fmt, cmap):
    plt.figure(figsize=(10, 8))
    ax = sns.heatmap(
        data_matrix,
        annot=True,
        fmt=fmt,
        cmap=cmap,
        cbar_kws={"shrink":0.8},
        square=True,
        linewidths=.5
    )
    plt.title(title, fontsize=16, pad=15)
    plt.xticks(rotation=45, ha='right')
    plt.yticks(rotation=0)
    plt.tight_layout()
    plt.show()

plot_heatmap(correlation_matrix, "Correlation Matrix", ".2f", "RdBu_r")
plot_heatmap(t_value_matrix, "T-Value Matrix", ".2f", "inferno")
plot_heatmap(p_value_matrix, "P-Value Matrix (Significance of Correlations)", ".2e", "viridis")
```

# Visualization 4

- Initialize Matrices
- ✓ **correlation\_matrix**: Stores the Pearson correlation coefficients.
- ✓ **t\_value\_matrix**: Stores the t-values for the corresponding correlations.
- ✓ **p\_value\_matrix**: Stores the p-values to measure statistical significance.



## Heatmap

Correlation matrix of all numeric variables (Age, DailySteps, FitnessRating, AnnualIncome, WorkoutHoursWeekly, DeviceUsageHours).  
(with determining statistical significance)

### Correlation and Significance Calculations

- ✓ Correlation Coefficients (( r )): Calculated using `data[[var1, var2]].corr().iloc[0, 1]` for each pair of variables.

- ✓ t-Value: Calculated using the formula:

( n ): Sample size (number of rows in the dataset).

$$t = \frac{r \cdot \sqrt{n - 2}}{\sqrt{1 - r^2}}$$

- ✓ p-Value: Calculated using the t-distribution (`scipy.stats.t.cdf`), with degrees of freedom (( df = n - 2 )).

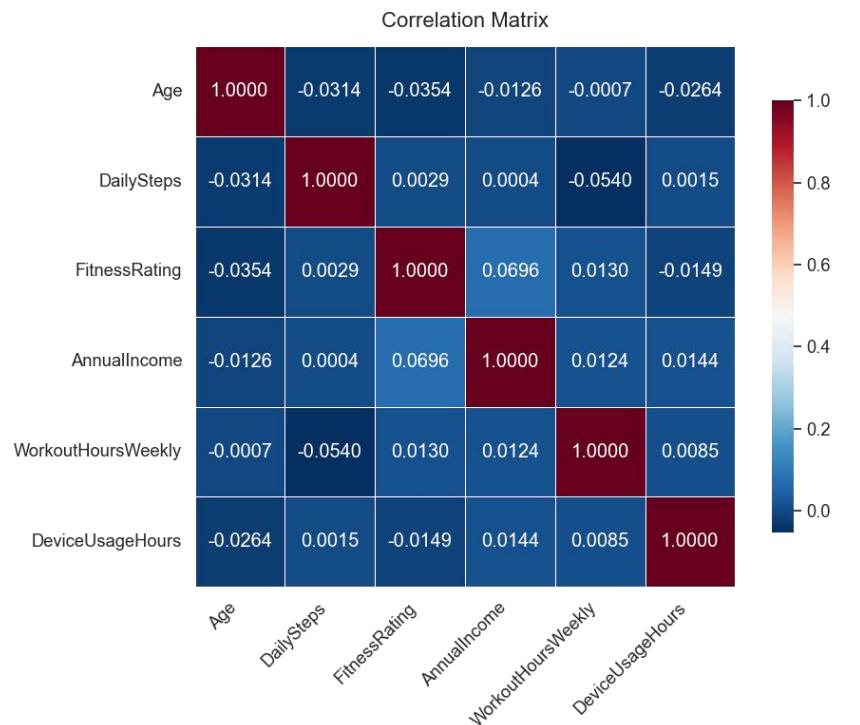
Multiplied by 2 for two-tailed significance testing.

### Visualization

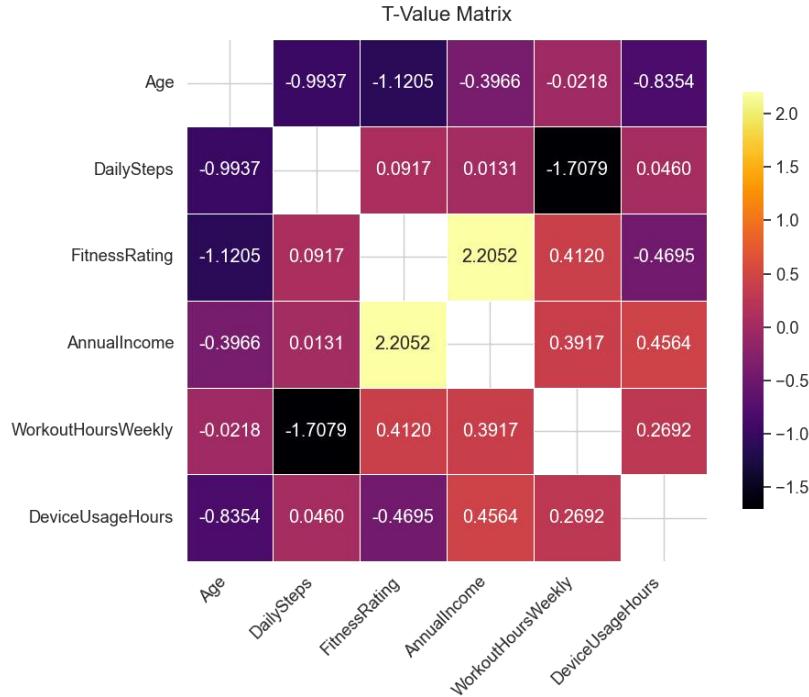
- ✓ Correlation Matrix Heatmap: Displays the strength and direction of correlations (range: -1 to 1). (red = positive, blue = negative).
- ✓ T-Value Matrix Heatmap: Displays the t-values, which are used to calculate p-values. Uses the "inferno" color scheme for better contrast.
- ✓ P-Value Matrix Heatmap: Uses the "viridis" color scheme to highlight small p-values (statistically significant).  
(smaller p-values = stronger evidence).

$$r = \frac{\sum_{i=1}^n (x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{X})^2 \sum_{i=1}^n (y_i - \bar{Y})^2}}$$

# Visualization 4

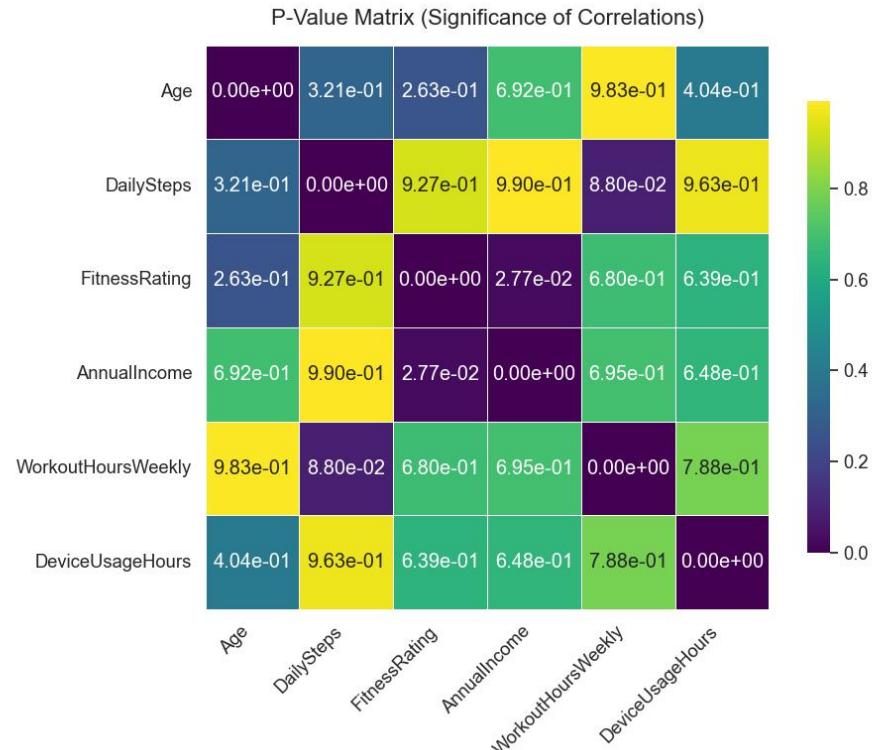


- **Correlation Matrix Heatmap**
- **Each cell in this heatmap shows the correlation coefficient ( $r$ ) between two variables:**
  - Positive values (closer to +1): Strong positive linear relationship.
  - Negative values (closer to -1): Strong negative linear relationship.
  - Values near 0: Weak or no linear relationship.



- **T-Value Matrix Heatmap**
- **Each cell in the T-Value Matrix Heatmap shows the t-value for the corresponding correlation:**
  - t-value  $> 0$ : Indicates a positive correlation between the two variables. The larger the t-value, the stronger the evidence for a positive relationship.
  - t-value  $< 0$ : Indicates a negative correlation between the two variables. The smaller (more negative) the t-value, the stronger the evidence for a negative relationship.
  - t-value close to 0: Indicates weak or no evidence of a linear relationship between the variables.

# Visualization 4



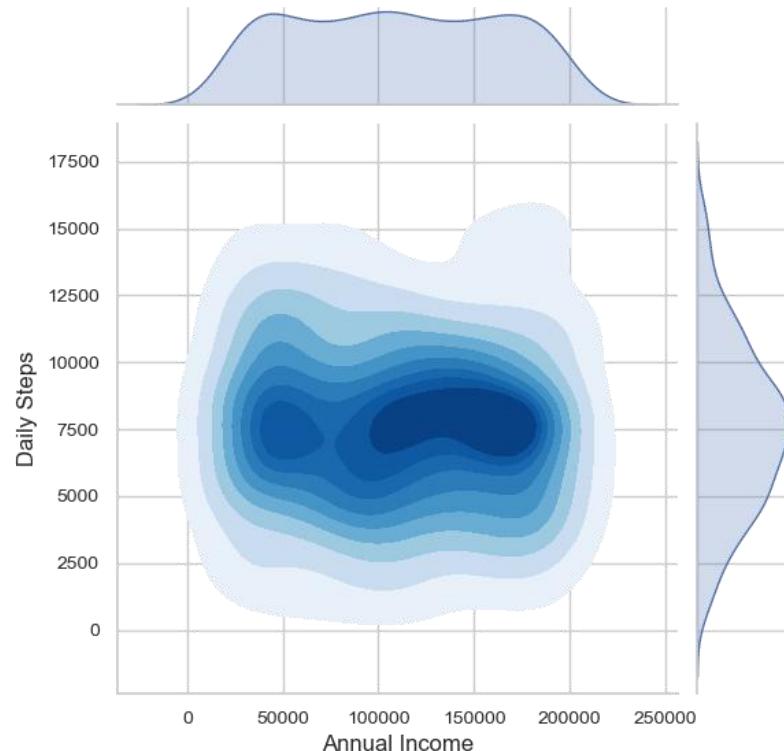
- **P-Value Matrix Heatmap**
- **Each cell in this heatmap shows the p-value for the corresponding correlation:**
  - p-value < 0.05: The correlation is statistically significant.
  - p-value ≥ 0.05: The correlation is not statistically significant (likely due to chance).

# Visualization 5

```
# 5. Jointplot: Relationship between AnnualIncome and DailySteps for Urban customers.
```

```
### BEGIN SOLUTION
```

```
urban_data = data[data["ResidenceArea"] == "Urban"]
sns.jointplot(data=urban_data, x="AnnualIncome", y="DailySteps", kind="kde", fill=True)
plt.suptitle("Income vs. Steps (Urban Customers)", y=1.02)
plt.show()
### END SOLUTION
```



## Jointplot

Relationship between AnnualIncome and DailySteps for Urban customers with KDE overlay.

- Filter the data to include only customers whose "ResidenceArea" is "Urban."
- **sns.jointplot()** is used to plot the relationship between annual income and daily steps
- ✓ **kind="kde"** specify a kernel density estimate plot and fill-True filling the interior of the plot.
- **plt. suptitle()**: The title "Income vs. Steps (Urban Customers)" is set
- ✓ **y=1.02** parameter moves the title slightly upward to avoid overlap with the plot.

$$\hat{f}(x, y) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2\pi h_x h_y} \exp \left( -\frac{(x - x_i)^2}{2h_x^2} - \frac{(y - y_i)^2}{2h_y^2} \right)$$

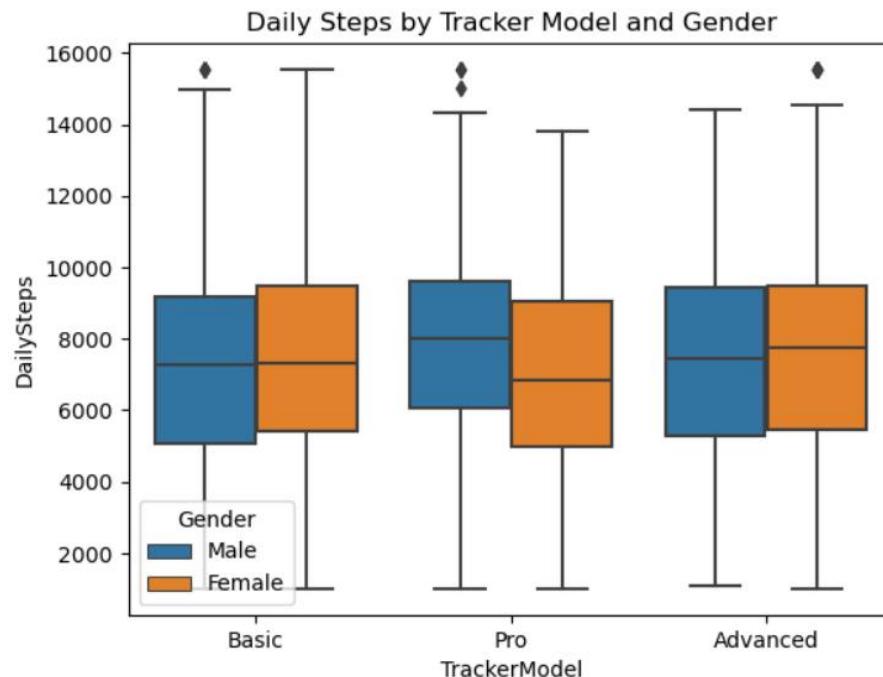
$$h_i = \sigma_i \cdot n^{-1/(n+4)}$$

# Visualization 6

```
# 6. Box Plot: DailySteps by TrackerModel, categorized by Gender.
```

```
### BEGIN SOLUTION
```

```
sns.boxplot(data=data, x="TrackerModel", y="DailySteps", hue="Gender")
plt.title("Daily Steps by Tracker Model and Gender")
plt.show()
### END SOLUTION
```



## Box Plot

Distribution of DailySteps by TrackerModel, categorized by Gender.

- **sns.boxplot()**

- ✓ **x="TrackerModel"** sets the tracker model on the x-axis

- ✓ **y="DailySteps"** sets daily steps on the y-axis

- ✓ **hue="Gender"** differentiates the data by gender.

- **plt.title()** adds the title "Daily Steps by Tracker Model and Gender."



Thank You