Code written for Supplementary Material

Table of Contents

[Case 1 2](#_Toc185116454)

[Temperature= 0 2](#_Toc185116455)

[Temperature= 0.1 5](#_Toc185116456)

[Temperature= 0.5 7](#_Toc185116457)

[Case 2 10](#_Toc185116458)

[Temperature= 0 10](#_Toc185116459)

[Temperature= .1 11](#_Toc185116460)

[Temperature= 0.5 13](#_Toc185116461)

[Case 3 17](#_Toc185116462)

[Temperature= 0 17](#_Toc185116463)

[Temperature= 0.1 19](#_Toc185116464)

[Temperature= 0.5 21](#_Toc185116465)

[Case 4 24](#_Toc185116466)

[Temperature= 0 24](#_Toc185116467)

[Temperature= 0.1 25](#_Toc185116468)

[Temperature= 0.5 28](#_Toc185116469)

[Case 5 31](#_Toc185116470)

[Temperature= 0 31](#_Toc185116471)

[Temperature= 0.1 33](#_Toc185116472)

[Temperature= 0.5 36](#_Toc185116473)

[Case 6 40](#_Toc185116474)

[Temperature= 0 40](#_Toc185116475)

[Temperature= 0.1 43](#_Toc185116476)

[Temperature= 0.5 45](#_Toc185116477)

[Case 7 48](#_Toc185116478)

[Temperature= 0 48](#_Toc185116479)

[Temperature= 0.1 51](#_Toc185116480)

[Temperature= 0.5 54](#_Toc185116481)

[Case 8 57](#_Toc185116482)

[Temperature= 0 57](#_Toc185116483)

[Temperature= 0.1 59](#_Toc185116484)

[Temperature= 0.5 62](#_Toc185116485)

[Case 9 65](#_Toc185116486)

[Temperature= 0 65](#_Toc185116487)

[Temperature= 0.1 67](#_Toc185116488)

[Temperature= 0.5 69](#_Toc185116489)

[Case 10 72](#_Toc185116490)

[Temperature= 0 72](#_Toc185116491)

[Temperature= 0.1 73](#_Toc185116492)

[Temperature= 0.5 76](#_Toc185116493)

# Case 1

## Temperature= 0

assistant > First, I will load and examine the data from the uploaded file to understand its structure and format. This will help in extracting the necessary information to calculate the requested metrics.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-M1681JERtMMSA3PyBJKJ5W'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset and its columns to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process the data to calculate the requested metrics. This includes:

1. Number of days the sensor is active

2. Percentage of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. Percentage of time above target glucose range (>250 mg/dL and >180 mg/dL)

7. Percentage of time in target glucose range (70-180 mg/dL)

8. Percentage of time below target glucose range (<70 mg/dL and <54 mg/dL)

Let's start by calculating these metrics.from datetime import datetime

# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the total possible readings (assuming 5-minute intervals throughout the day)

total\_possible\_readings = num\_days \* 24 \* 60 / 5

# Calculate the percentage of sensor data captured

percent\_data\_captured = (len(data) / total\_possible\_readings) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate glucose variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate time above target glucose range

time\_above\_250 = len(data[data['CGM'] > 250]) / len(data) \* 100

time\_above\_180 = len(data[data['CGM'] > 180]) / len(data) \* 100

# Calculate time in target glucose range (70-180 mg/dL)

time\_in\_range = len(data[(data['CGM'] >= 70) & (data['CGM'] <= 180)]) / len(data) \* 100

# Calculate time below target glucose range

time\_below\_70 = len(data[data['CGM'] < 70]) / len(data) \* 100

time\_below\_54 = len(data[data['CGM'] < 54]) / len(data) \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose, time\_above\_250, time\_above\_180, time\_in\_range, time\_below\_70, time\_below\_54]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*88.54%\*\*

3. Mean glucose concentration: \*\*128.56 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.39\*\*

5. Glucose variability (coefficient of variation): \*\*24.11%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*0.00%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*4.71%\*\*

8. % Time in range (70-180 mg/dL): \*\*95.01%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.28%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I'll examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-9AeXMfs53Fj3Te72UgRsm6'

# Attempt to open and read the file to determine its format (CSV, Excel, etc.)

import pandas as pd

try:

# Try reading as a CSV

data = pd.read\_csv(file\_path)

except Exception as e\_csv:

try:

# Try reading as an Excel file

data = pd.read\_excel(file\_path)

except Exception as e\_excel:

# Output errors if both attempts fail

error\_csv = str(e\_csv)

error\_excel = str(e\_excel)

data, error\_csv, error\_excel

assistant > It seems there was an issue with reading the file directly. I will attempt to read the file again using different methods to determine its format and content. Let's try reading it as both a CSV and an Excel file explicitly to see if we can access the data.# Attempt to read the file as both CSV and Excel to identify the format

try:

data\_csv = pd.read\_csv(file\_path)

format\_used = 'CSV'

except Exception as e\_csv:

try:

data\_excel = pd.read\_excel(file\_path)

format\_used = 'Excel'

except Exception as e\_excel:

format\_used = 'Unknown'

error\_csv = str(e\_csv)

error\_excel = str(e\_excel)

format\_used, error\_csv if format\_used == 'Unknown' else None, error\_excel if format\_used == 'Unknown' else None

assistant > The file is in CSV format. I will now examine the first few rows of the data to understand its structure and the type of information it contains. This will help in extracting the necessary data for the analysis.# Display the first few rows of the CSV data to understand its structure

data\_csv.head()

assistant > The data contains the following columns:

- `Unnamed: 0`: An index column.

- `CGM`: The glucose concentration in mg/dL.

- `time`: The time of day.

- `weekday`: The day of the week.

- `Time`: The full timestamp including date and time.

Next, I will process this data to calculate the requested metrics. This includes:

1. Number of days the sensor is active.

2. Percentage of sensor data captured.

3. Mean glucose concentration.

4. Glucose management indicator (GMI).

5. Glucose variability (coefficient of variation).

6. Percentage of time above target glucose range (>250 mg/dL and >180 mg/dL).

7. Percentage of time in target range (70-180 mg/dL).

8. Percentage of time below target glucose range (<70 mg/dL and <54 mg/dL).

Let's start by calculating these metrics.import numpy as np

# Convert 'Time' column to datetime format

data\_csv['Time'] = pd.to\_datetime(data\_csv['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data\_csv['Time'].dt.date.nunique()

# Calculate the percentage of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements\_per\_day = 24 \* 60 / 5

actual\_measurements\_per\_day = data\_csv.groupby(data\_csv['Time'].dt.date).size().mean()

percent\_data\_captured = (actual\_measurements\_per\_day / expected\_measurements\_per\_day) \* 100

# Calculate mean glucose concentration

mean\_glucose = data\_csv['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI = 3.31 + 0.02392 × [mean glucose in mg/dL]

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate glucose variability (coefficient of variation)

cv\_glucose = data\_csv['CGM'].std() / mean\_glucose \* 100

# Calculate time above range, in range, and below range

total\_measurements = len(data\_csv)

tar\_level\_2 = (data\_csv['CGM'] > 250).sum() / total\_measurements \* 100

tar\_level\_1 = (data\_csv['CGM'] > 180).sum() / total\_measurements \* 100

tir = ((data\_csv['CGM'] >= 70) & (data\_csv['CGM'] <= 180)).sum() / total\_measurements \* 100

tbr\_level\_1 = (data\_csv['CGM'] < 70).sum() / total\_measurements \* 100

tbr\_level\_2 = (data\_csv['CGM'] < 54).sum() / total\_measurements \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, tir, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics based on the CGM data provided:

1. \*\*Number of days the sensor is active:\*\* 14 days

2. \*\*% of sensor data captured:\*\* 88.54%

3. \*\*Mean glucose concentration:\*\* 128.56 mg/dL

4. \*\*Glucose management indicator (GMI):\*\* 6.39

5. \*\*Glucose variability (coefficient of variation):\*\* 24.11%

6. \*\*% Time above target glucose range (TAR) in level 2 (>250 mg/dL):\*\* 0.00%

7. \*\*% Time above target glucose range (TAR) in level 1 (>180 mg/dL):\*\* 4.71%

8. \*\*% Time in range (70-180 mg/dL):\*\* 95.01%

9. \*\*% Time below target glucose range (TBR) in level 1 (<70 mg/dL):\*\* 0.28%

10. \*\*% Time below target glucose range (TBR) in level 2 (<54 mg/dL):\*\* 0.00%

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.5

assistant > First, I will examine the contents of the uploaded file to determine the structure and format of the data. This will allow me to process it accurately to compute the requested metrics.

assistant > code\_interpreter

# Let's read the uploaded file and examine the first few rows to understand its structure and format.

import pandas as pd

# Load the data

file\_path = '/mnt/data/file-J3GVABHczhiZuHLXc7GHB1'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataframe

data.head()

assistant > The data consists of columns indicating the glucose concentration (CGM), time of the day, weekday, and a combined timestamp column (Time). I will now proceed to calculate the requested metrics based on this data.

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

Let's calculate these metrics.from datetime import datetime

# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements\_per\_day = 24 \* 60 / 5

total\_expected\_measurements = expected\_measurements\_per\_day \* num\_days

percent\_data\_captured = (len(data) / total\_expected\_measurements) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate Glucose Variability (Coefficient of Variation)

cv\_glucose = (data['CGM'].std() / mean\_glucose) \* 100

# Calculate % Time in different glucose ranges

total\_measurements = len(data)

tar\_level\_2 = (data['CGM'] > 250).sum() / total\_measurements \* 100

tar\_level\_1 = (data['CGM'] > 180).sum() / total\_measurements \* 100

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).sum() / total\_measurements \* 100

tbr\_level\_1 = (data['CGM'] < 70).sum() / total\_measurements \* 100

tbr\_level\_2 = (data['CGM'] < 54).sum() / total\_measurements \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*88.54%\*\*

3. Mean glucose concentration: \*\*128.56 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.39\*\*

5. Glucose variability (coefficient of variation): \*\*24.11%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*0.00%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*4.71%\*\*

8. % Time in range (70-180 mg/dL): \*\*95.01%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.28%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These metrics provide a comprehensive overview of the patient's glucose management over the analyzed period.

# Case 2

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help in extracting the necessary information to calculate the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-6RRVtBq28YeGYsaCPfAd1C'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of the day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process this data to calculate the requested metrics. Let's start by converting the `Time` column to a datetime format and then proceed with the calculations.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured

expected\_measurements\_per\_day = 24 \* 60 / 5 # 5-minute intervals in a day

actual\_measurements\_per\_day = data.groupby(data['Time'].dt.date).size()

percent\_captured = (actual\_measurements\_per\_day / expected\_measurements\_per\_day).mean() \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate Glucose Variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*88.81%\*\*

3. Mean glucose concentration: \*\*183.26 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.69\*\*

5. Glucose variability (coefficient of variation): \*\*29.22%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*10.47%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*48.81%\*\*

8. % Time in range (70-180 mg/dL): \*\*50.80%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.39%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= .1

assistant > First, I'll need to examine the contents of the uploaded file to understand its format and the data it contains. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-6yGpDFAsXr9sDdRMExbLth'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head()

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process this data to calculate the metrics requested. This includes:

1. Number of days the sensor is active

2. Percentage of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. Percentage of time above target glucose range (>250 mg/dL)

7. Percentage of time above target glucose range (>180 mg/dL)

8. Percentage of time in target glucose range (70-180 mg/dL)

9. Percentage of time below target glucose range (<70 mg/dL)

10. Percentage of time below target glucose range (<54 mg/dL)

Let's start by calculating these metrics.from datetime import datetime

# Convert 'Time' column to datetime format

data['Datetime'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Datetime'].dt.date.nunique()

# Calculate the total possible readings (assuming 5-minute intervals for 24 hours)

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = num\_days \* expected\_readings\_per\_day

# Calculate the percentage of sensor data captured

actual\_readings = len(data)

percent\_data\_captured = (actual\_readings / total\_expected\_readings) \* 100

# Calculate the mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate the glucose variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose

# Calculate the percentage of time in different glucose ranges

time\_above\_250 = (data['CGM'] > 250).sum() / actual\_readings \* 100

time\_above\_180 = (data['CGM'] > 180).sum() / actual\_readings \* 100

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).sum() / actual\_readings \* 100

time\_below\_70 = (data['CGM'] < 70).sum() / actual\_readings \* 100

time\_below\_54 = (data['CGM'] < 54).sum() / actual\_readings \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose,

time\_above\_250, time\_above\_180, time\_in\_range, time\_below\_70, time\_below\_54]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*88.81%\*\*

3. Mean glucose concentration: \*\*183.26 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.69\*\*

5. Glucose variability (coefficient of variation): \*\*29.22%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*10.47%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*48.81%\*\*

8. % Time in range (70-180 mg/dL): \*\*50.80%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.39%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.5

assistant > Let's start by examining the contents of the uploaded file to understand its structure and format. This will help us extract the necessary data to calculate the requested metrics.

assistant > code\_interpreter

# Let's first load the uploaded file to understand its structure and format.

import pandas as pd

# Load the data

file\_path = '/mnt/data/file-PgGqghEqGjJgM94cu6yVFP'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataframe and its column names to understand its structure.

data.head(), data.columns

assistant > The data contains the following columns:

- `Unnamed: 0`: An index column that we can ignore.

- `CGM`: Continuous Glucose Monitoring data (glucose concentrations in mg/dL).

- `time`: Time of the day in 5-minute intervals.

- `weekday`: Day of the week.

- `Time`: Combined date and time information.

We will use the `Time` and `CGM` columns to calculate the requested metrics. Let's start by cleaning and preparing the data.# Convert 'Time' column to datetime format for easier manipulation

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Drop the unnecessary columns

data\_clean = data[['Time', 'CGM']]

# Verify the changes and inspect the data types

data\_clean.head(), data\_clean.dtypes

assistant > The data is now clean with the `Time` column formatted as datetime and the `CGM` column containing glucose values. Next, I will calculate the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250)

7. % Time above target glucose range (TAR) in level 1 (>180)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70)

10. % Time below target glucose range (TBR) in level 2 (<54)

Let's proceed with these calculations.import numpy as np

# Total number of readings possible (assuming a reading every 5 minutes for 24 hours)

total\_readings\_per\_day = 24 \* 60 / 5

# 1. Number of days the sensor is active

num\_days = (data\_clean['Time'].max() - data\_clean['Time'].min()).days + 1

# 2. % of sensor data captured

actual\_readings = data\_clean.shape[0]

expected\_readings = total\_readings\_per\_day \* num\_days

percent\_captured = (actual\_readings / expected\_readings) \* 100

# 3. Mean glucose concentration

mean\_glucose = data\_clean['CGM'].mean()

# 4. Glucose management indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# 5. Glucose variability (coefficient of variation)

cv\_glucose = (data\_clean['CGM'].std() / mean\_glucose) \* 100

# 6. % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data\_clean['CGM'] > 250).mean() \* 100

# 7. % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data\_clean['CGM'] > 180).mean() \* 100

# 8. % Time in range (70-180 mg/dL)

time\_in\_range = ((data\_clean['CGM'] >= 70) & (data\_clean['CGM'] <= 180)).mean() \* 100

# 9. % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data\_clean['CGM'] < 70).mean() \* 100

# 10. % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data\_clean['CGM'] < 54).mean() \* 100

[num\_days, percent\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics based on the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*88.81%\*\*

3. Mean glucose concentration: \*\*183.26 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.69\*\*

5. Glucose variability (coefficient of variation): \*\*29.22%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*10.47%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*48.81%\*\*

8. % Time in range (70-180 mg/dL): \*\*50.80%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.39%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These metrics provide a comprehensive overview of the patient's glucose management over the period the sensor was active. If you need further analysis or have any other questions, feel free to ask!

# Case 3

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-Xix892ZcCytyQQYmUf3UbU'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of the day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process this data to calculate the requested metrics. I'll start by converting the `Time` column to a datetime format for easier manipulation and then proceed with the calculations.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured

# Assuming data should be captured every 5 minutes, calculate the expected number of readings per day

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

percent\_data\_captured = (len(data) / total\_expected\_readings) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* Mean glucose

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate Glucose Variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose

# Calculate % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose,

tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*86.06%\*\*

3. Mean glucose concentration: \*\*171.67 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.42\*\*

5. Glucose variability (coefficient of variation): \*\*37.59%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*8.62%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*34.12%\*\*

8. % Time in range (70-180 mg/dL): \*\*65.76%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.12%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I'll examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-VsyhpPSmWdjmE2qdWZ7AUo'

# Attempt to read the file with common formats to identify the data structure

def check\_file\_content(file\_path):

try:

import pandas as pd

# Try reading as CSV

data = pd.read\_csv(file\_path)

return data.head()

except Exception as e:

return str(e)

check\_file\_content(file\_path)

assistant > The file appears to be a CSV containing columns for CGM glucose readings, time, weekday, and a combined date-time column. I will now proceed to load the data and calculate the requested metrics.import pandas as pd

# Load the data

data = pd.read\_csv(file\_path)

# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Sort data by time to ensure it's in chronological order

data.sort\_values('Time', inplace=True)

# Reset index after sorting

data.reset\_index(drop=True, inplace=True)

data.head()

assistant > The data is now loaded and formatted correctly. I will proceed with calculating the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)import numpy as np

# Calculate the number of days the sensor is active

num\_days = (data['Time'].iloc[-1] - data['Time'].iloc[0]).days + 1

# Calculate % of sensor data captured

expected\_measurements\_per\_day = 24 \* 60 / 5 # 5-minute intervals in a day

total\_expected\_measurements = expected\_measurements\_per\_day \* num\_days

percent\_data\_captured = (len(data) / total\_expected\_measurements) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate glucose variability (coefficient of variation)

cv\_glucose = (data['CGM'].std() / mean\_glucose) \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*86.06%\*\*

3. Mean glucose concentration: \*\*171.67 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.42\*\*

5. Glucose variability (coefficient of variation): \*\*37.59%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*8.62%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*34.12%\*\*

8. % Time in range (70-180 mg/dL): \*\*65.76%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.12%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These metrics provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.5

assistant > First, I will begin by examining the contents of the uploaded file to understand its format and the data it contains. This will help us extract the necessary information to calculate the requested metrics.

assistant > code\_interpreter

# Let's read the uploaded file to understand its format and data structure.

file\_path = '/mnt/data/file-H7FLSQSQqrPt1Ui8Z8NfS6'

# Attempting to open and preview the file to determine its structure and content type

with open(file\_path, 'r') as file:

content = file.readlines()

content[:10] # Display the first 10 lines to get an idea of the data structure

assistant > The file appears to be a CSV with columns indicating:

- An index

- CGM (Continuous Glucose Monitoring) glucose values

- Time of the day

- Day of the week

- Full date and time

Next, I will load the data into a DataFrame for further analysis and processing to compute the requested metrics.import pandas as pd

# Load the data into a DataFrame

df = pd.read\_csv(file\_path)

# Parse the 'Time' column into datetime format for easier manipulation

df['Time'] = pd.to\_datetime(df['Time'], format='%m/%d/%y %H:%M')

# Display the first few rows of the DataFrame to confirm correct loading and parsing

df.head()

assistant > The data has been successfully loaded into a DataFrame. Now, I will proceed to calculate the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

Let's start by calculating these metrics.import numpy as np

# 1. Number of days the sensor is active

num\_days = df['Time'].dt.date.nunique()

# 2. % of sensor data captured

# Assuming data should be captured every 5 minutes, 288 readings per day

expected\_readings\_per\_day = 288

actual\_readings = df.shape[0]

expected\_readings = num\_days \* expected\_readings\_per\_day

percent\_data\_captured = (actual\_readings / expected\_readings) \* 100

# 3. Mean glucose concentration

mean\_glucose = df['CGM'].mean()

# 4. Glucose management indicator (GMI)

# GMI = 3.31 + 0.02392 × [Mean glucose in mg/dL]

gmi = 3.31 + 0.02392 \* mean\_glucose

# 5. Glucose variability (coefficient of variation)

cv\_glucose = (df['CGM'].std() / mean\_glucose) \* 100

# 6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

tar\_level\_2 = (df['CGM'] > 250).mean() \* 100

# 7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

tar\_level\_1 = (df['CGM'] > 180).mean() \* 100

# 8. % Time in range (70-180 mg/dL)

time\_in\_range = ((df['CGM'] >= 70) & (df['CGM'] <= 180)).mean() \* 100

# 9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

tbr\_level\_1 = (df['CGM'] < 70).mean() \* 100

# 10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

tbr\_level\_2 = (df['CGM'] < 54).mean() \* 100

# Compile all metrics into a list

metrics = [

num\_days,

percent\_data\_captured,

mean\_glucose,

gmi,

cv\_glucose,

tar\_level\_2,

tar\_level\_1,

time\_in\_range,

tbr\_level\_1,

tbr\_level\_2

]

metrics

assistant > Here are the computed metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*86.06%\*\*

3. Mean glucose concentration: \*\*171.67 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.42\*\*

5. Glucose variability (coefficient of variation): \*\*37.59%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*8.62%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*34.12%\*\*

8. % Time in range (70-180 mg/dL): \*\*65.76%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.12%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These metrics provide a comprehensive overview of the glucose management for the patient over the observed period.

# Case 4

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help in extracting the necessary information to compute the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-MscgemacGbgWUMo9PnQPSR'

# Attempt to open and read the file to determine its format (CSV, Excel, etc.)

import pandas as pd

try:

# Try reading as a CSV

data = pd.read\_csv(file\_path)

except Exception as e:

print("Not a CSV. Error:", e)

try:

# Try reading as an Excel file

data = pd.read\_excel(file\_path)

except Exception as e:

print("Not an Excel file. Error:", e)

data.head()

assistant > The data appears to be structured with columns indicating the time, glucose readings (CGM), and additional time-related information. I will now proceed to calculate the requested metrics based on this data. Let's start by parsing the time information and computing the metrics.# Parse the 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate the % of sensor data captured assuming a reading every 5 minutes

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

percent\_captured = (len(data) / total\_expected\_readings) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* mean glucose

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate Glucose Variability (Coefficient of Variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time in different ranges

total\_time = len(data)

time\_above\_250 = len(data[data['CGM'] > 250]) / total\_time \* 100

time\_above\_180 = len(data[data['CGM'] > 180]) / total\_time \* 100

time\_in\_range = len(data[(data['CGM'] >= 70) & (data['CGM'] <= 180)]) / total\_time \* 100

time\_below\_70 = len(data[data['CGM'] < 70]) / total\_time \* 100

time\_below\_54 = len(data[data['CGM'] < 54]) / total\_time \* 100

[num\_days, percent\_captured, mean\_glucose, gmi, cv\_glucose, time\_above\_250, time\_above\_180, time\_in\_range, time\_below\_70, time\_below\_54]

assistant > Here are the computed metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*90.33%\*\*

3. Mean glucose concentration: \*\*204.30 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*8.20\*\*

5. Glucose variability (coefficient of variation): \*\*26.82%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*18.29%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*67.44%\*\*

8. % Time in range (70-180 mg/dL): \*\*32.43%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.14%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to compute the requested metrics.

assistant > code\_interpreter

# Let's first check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-GRiAX9EgwK3cTQZ7eH5L6f'

# Attempt to read the file with common formats such as CSV or Excel to see the structure.

import pandas as pd

try:

data = pd.read\_csv(file\_path)

except Exception as e:

try:

data = pd.read\_excel(file\_path)

except Exception as e2:

error\_csv = str(e)

error\_excel = str(e2)

data = None, error\_csv, error\_excel

data

assistant > The data appears to be in a structured format with columns indicating the CGM (Continuous Glucose Monitoring) values, time, weekday, and a combined timestamp. I will now proceed to calculate the requested metrics based on this data.import numpy as np

# Parse the 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured

# Assuming data should be captured every 5 minutes, calculate total expected readings

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

percent\_data\_captured = (len(data) / total\_expected\_readings) \* 100

# Calculate the mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* Mean glucose in mg/dL

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate the Glucose Variability (Coefficient of Variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

# Compile all metrics into a list

metrics = [

num\_days,

percent\_data\_captured,

mean\_glucose,

gmi,

cv\_glucose,

tar\_level\_2,

tar\_level\_1,

time\_in\_range,

tbr\_level\_1,

tbr\_level\_2

]

metrics

assistant > Here are the computed metrics based on the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*90.33%\*\*

3. Mean glucose concentration: \*\*204.30 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*8.20\*\*

5. Glucose variability (coefficient of variation): \*\*26.82%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*18.29%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*67.44%\*\*

8. % Time in range (70-180 mg/dL): \*\*32.43%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.14%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.5

assistant > First, I will examine the contents of the uploaded file to understand its structure and determine how to extract the necessary data for the analysis. Let's start by loading the file and taking a look at its contents.

assistant > code\_interpreter

import pandas as pd

# Load the data to see the first few rows and understand its structure

file\_path = '/mnt/data/file-86YwavRtMaQYqhsbkdhoCj'

data = pd.read\_csv(file\_path)

data.head()

assistant > The dataset contains the following columns:

- `Unnamed: 0`: an index column

- `CGM`: the glucose concentration in mg/dL

- `time`: the time of day

- `weekday`: the day of the week

- `Time`: the date and time combined

Next, I'll process this data to extract the required metrics. I'll start by ensuring the `Time` column is in the appropriate datetime format and then proceed with the analysis.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days\_active = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements\_per\_day = 24 \* 60 / 5 # 288 measurements per day

total\_expected\_measurements = expected\_measurements\_per\_day \* num\_days\_active

percent\_data\_captured = (len(data) / total\_expected\_measurements) \* 100

# Calculate the mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate the Glucose Variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose

# Calculate % Time above target glucose range (TAR) in level 2 (>250)

tar\_level2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180)

tar\_level1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

tir = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level2 = (data['CGM'] < 54).mean() \* 100

metrics = [

num\_days\_active,

percent\_data\_captured,

mean\_glucose,

gmi,

cv\_glucose,

tar\_level2,

tar\_level1,

tir,

tbr\_level1,

tbr\_level2

]

metrics

assistant > Here are the computed metrics based on the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*90.33%\*\*

3. Mean glucose concentration: \*\*204.30 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*8.20\*\*

5. Glucose variability (coefficient of variation): \*\*26.82%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*18.29%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*67.44%\*\*

8. % Time in range (70-180 mg/dL): \*\*32.43%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.14%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values are presented as a list in the same order as requested.

# Case 5

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-F8nMztXu4fjezMaW28a4uw'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process this data to calculate the requested metrics. Let's start by converting the `Time` column to a datetime format and then proceed with the calculations.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured

# Assuming data should be captured every 5 minutes, calculate total expected readings

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

percent\_data\_captured = (len(data) / total\_expected\_readings) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* mean glucose

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate glucose variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*97.67%\*\*

3. Mean glucose concentration: \*\*121.24 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.21\*\*

5. Glucose variability (coefficient of variation): \*\*18.12%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*0.0%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*0.63%\*\*

8. % Time in range (70-180 mg/dL): \*\*99.37%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.0%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.0%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I will examine the contents of the uploaded file to understand the format and structure of the data. This will help in extracting the necessary information to calculate the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-5493jK2oHoeYPjhAXRTr6T'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of the day

- `weekday`: Day of the week

- `Time`: Combined date and time

Next, I will clean the data by removing unnecessary columns and converting the `Time` column to a datetime format for easier analysis. Then, I will proceed to calculate the requested metrics.# Remove unnecessary columns and convert 'Time' to datetime format

data.drop(columns=['Unnamed: 0', 'time', 'weekday'], inplace=True)

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Check the cleaned data

data.head(), data.dtypes

assistant > The data is now cleaned and ready for analysis. I will proceed with calculating the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

Let's start by calculating these metrics.import numpy as np

# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements = num\_days \* 24 \* 60 / 5

percent\_captured = (len(data) / expected\_measurements) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate glucose variability (coefficient of variation)

cv\_glucose = (data['CGM'].std() / mean\_glucose) \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*97.67%\*\*

3. Mean glucose concentration: \*\*121.24 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.21\*\*

5. Glucose variability (coefficient of variation): \*\*18.12%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*0.00%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*0.63%\*\*

8. % Time in range (70-180 mg/dL): \*\*99.37%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.00%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.5

assistant > First, I will load and examine the data from the uploaded CGM file to understand its structure and format. Then, I'll proceed with calculating the requested metrics. Let's start by loading the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-PeQz5PWfiHWMrYRAfp68ci'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataframe to understand its structure

data.head(), data.columns

assistant > The data contains columns for CGM glucose readings, time, weekday, and a timestamp. I will use the 'Time' column for datetime operations and 'CGM' for glucose level calculations.

Now, let's calculate the metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250)

7. % Time above target glucose range (TAR) in level 1 (>180)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70)

10. % Time below target glucose range (TBR) in level 2 (<54)

First, I'll convert the 'Time' column to datetime format and calculate the number of days the sensor is active.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

num\_days

assistant > The sensor has been active for 14 days.

Next, let's calculate the percentage of sensor data captured. This is typically calculated as the actual number of readings divided by the expected number of readings (assuming a reading every 5 minutes).# Calculate total number of expected readings (every 5 minutes over the number of days)

expected\_readings = num\_days \* 24 \* 60 / 5

# Calculate the percentage of sensor data captured

percent\_data\_captured = (len(data) / expected\_readings) \* 100

percent\_data\_captured

assistant > The percentage of sensor data captured is approximately 97.67%.

Now, let's calculate the mean glucose concentration.# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

mean\_glucose

assistant > The mean glucose concentration is approximately 121.24 mg/dL.

Next, we will calculate the Glucose Management Indicator (GMI). The GMI is calculated using the formula:

\[ \text{GMI} = 3.31 + 0.02392 \times \text{Mean Glucose} \]# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + 0.02392 \* mean\_glucose

gmi

assistant > The Glucose Management Indicator (GMI) is approximately 6.21.

Next, let's calculate the glucose variability, which is the coefficient of variation (CV). The CV is defined as the standard deviation of the glucose readings divided by the mean glucose concentration.# Calculate the standard deviation of glucose readings

std\_glucose = data['CGM'].std()

# Calculate the coefficient of variation (CV)

cv\_glucose = (std\_glucose / mean\_glucose) \* 100

cv\_glucose

assistant > The glucose variability, measured as the coefficient of variation (CV), is approximately 18.12%.

Now, we will calculate the percentages of time spent:

- Above target glucose range (TAR) in level 2 (>250 mg/dL)

- Above target glucose range (TAR) in level 1 (>180 mg/dL)

- In target range (70-180 mg/dL)

- Below target glucose range (TBR) in level 1 (<70 mg/dL)

- Below target glucose range (TBR) in level 2 (<54 mg/dL)

Let's start with these calculations.# Calculate percentage of time in different glucose ranges

total\_readings = len(data)

# Time above range (TAR) level 2 (>250 mg/dL)

tar\_level\_2 = (data['CGM'] > 250).sum() / total\_readings \* 100

# Time above range (TAR) level 1 (>180 mg/dL)

tar\_level\_1 = (data['CGM'] > 180).sum() / total\_readings \* 100

# Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).sum() / total\_readings \* 100

# Time below range (TBR) level 1 (<70 mg/dL)

tbr\_level\_1 = (data['CGM'] < 70).sum() / total\_readings \* 100

# Time below range (TBR) level 2 (<54 mg/dL)

tbr\_level\_2 = (data['CGM'] < 54).sum() / total\_readings \* 100

tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2

assistant > Here are the percentages of time spent in different glucose ranges:

- % Time above target glucose range (TAR) in level 2 (>250 mg/dL): 0.0%

- % Time above target glucose range (TAR) in level 1 (>180 mg/dL): 0.63%

- % Time in range (70-180 mg/dL): 99.37%

- % Time below target glucose range (TBR) in level 1 (<70 mg/dL): 0.0%

- % Time below target glucose range (TBR) in level 2 (<54 mg/dL): 0.0%

Now, I will compile all the calculated metrics into a list as requested.# Compile metrics into a list

metrics = [

num\_days, # Number of days the sensor is active

percent\_data\_captured, # % of sensor data captured

mean\_glucose, # Mean glucose concentration

gmi, # Glucose management indicator

cv\_glucose, # Glucose variability (coefficient of variation)

tar\_level\_2, # % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_1, # % Time above target glucose range (TAR) in level 1 (>180)

time\_in\_range, # % Time in range (70-180 mg/dL)

tbr\_level\_1, # % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_2 # % Time below target glucose range (TBR) in level 2 (<54)

]

metrics

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: 14 days

2. % of sensor data captured: 97.67%

3. Mean glucose concentration: 121.24 mg/dL

4. Glucose management indicator (GMI): 6.21

5. Glucose variability (coefficient of variation): 18.12%

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): 0.0%

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): 0.63%

8. % Time in range (70-180 mg/dL): 99.37%

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): 0.0%

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): 0.0%

These metrics provide a comprehensive overview of the glucose management for the patient over the 14 days of sensor activity. If you need further analysis or have additional questions, feel free to ask!

# Case 6

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-AfUmqpK5wtsu5S2procYbe'

# Attempting to read the file with common formats to identify the correct one.

def check\_file\_content(file\_path):

try:

# Try reading as a CSV

import pandas as pd

data = pd.read\_csv(file\_path)

return data.head()

except Exception as e\_csv:

try:

# Try reading as an Excel file

data = pd.read\_excel(file\_path)

return data.head()

except Exception as e\_excel:

# Return errors if both attempts fail

return str(e\_csv), str(e\_excel)

check\_file\_content(file\_path)

assistant > The file appears to be a CSV containing glucose monitoring data with columns for CGM values, time, weekday, and a combined date-time column. I will now proceed to analyze this data to compute the requested metrics.import pandas as pd

# Load the data

data = pd.read\_csv(file\_path)

# Convert 'Time' column to datetime format for easier manipulation

data['Time'] = pd.to\_datetime(data['Time'])

# Check the data types and look for any missing values

data.info(), data.describe()

output >

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 3726 entries, 0 to 3725

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Unnamed: 0 3726 non-null int64

1 CGM 3726 non-null int64

2 time 3726 non-null object

3 weekday 3726 non-null object

4 Time 3726 non-null datetime64[ns]

dtypes: datetime64[ns](1), int64(2), object(2)

memory usage: 145.7+ KB

Attempt 1 failed with error: 'EventHandler' object has no attribute 'outputs'

Attempting...

#######

6

0

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-UD7McHWsfJjYT28Aa5RWpi'

# Attempt to open and read the file to determine its format (CSV, Excel, etc.)

import pandas as pd

# Try reading the file with common formats

def try\_read\_file(file\_path):

try:

# Try reading as CSV

data = pd.read\_csv(file\_path)

return data, "csv"

except Exception as e\_csv:

try:

# Try reading as Excel

data = pd.read\_excel(file\_path)

return data, "excel"

except Exception as e\_excel:

return None, "unknown"

data, file\_type = try\_read\_file(file\_path)

data.head(), file\_type

assistant > The file is in CSV format and contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Glucose concentration in mg/dL

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process this data to calculate the requested metrics.import numpy as np

# Convert 'Time' column to datetime format for easier manipulation

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements\_per\_day = 24 \* 60 / 5 # 288 measurements per day

actual\_measurements = len(data)

expected\_measurements = num\_days \* expected\_measurements\_per\_day

percent\_data\_captured = (actual\_measurements / expected\_measurements) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI = 3.31 + 0.02392 × [mean glucose in mg/dL]

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate Glucose Variability (Coefficient of Variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time in different ranges

total\_time = len(data)

time\_above\_250 = len(data[data['CGM'] > 250]) / total\_time \* 100

time\_above\_180 = len(data[data['CGM'] > 180]) / total\_time \* 100

time\_in\_range = len(data[(data['CGM'] >= 70) & (data['CGM'] <= 180)]) / total\_time \* 100

time\_below\_70 = len(data[data['CGM'] < 70]) / total\_time \* 100

time\_below\_54 = len(data[data['CGM'] < 54]) / total\_time \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose,

time\_above\_250, time\_above\_180, time\_in\_range, time\_below\_70, time\_below\_54]

assistant > Here are the calculated metrics based on the CGM data provided:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*92.41%\*\*

3. Mean glucose concentration: \*\*195.55 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.99\*\*

5. Glucose variability (coefficient of variation): \*\*33.46%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*19.78%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*54.27%\*\*

8. % Time in range (70-180 mg/dL): \*\*45.20%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.54%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help in extracting the necessary information to compute the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-DBe9eB1ZRMHy6c8tivzD7W'

with open(file\_path, 'r') as file:

lines = file.readlines()

sample\_lines = lines[:10] # Display the first 10 lines to understand the structure

sample\_lines

assistant > The file appears to be a CSV with columns for index, CGM glucose readings, time of day, day of the week, and a timestamp combining date and time. I will now load this data into a DataFrame for further analysis and computation of the requested metrics.import pandas as pd

# Load the data into a DataFrame

df = pd.read\_csv(file\_path)

# Display the first few rows of the DataFrame to confirm correct loading and to understand the data better

df.head()

assistant > The data has been successfully loaded into a DataFrame. Next, I will process and analyze the data to compute the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250)

7. % Time above target glucose range (TAR) in level 1 (>180)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70)

10. % Time below target glucose range (TBR) in level 2 (<54)

Let's start by computing each of these metrics.from datetime import datetime

# Convert 'Time' column to datetime format for easier manipulation

df['DateTime'] = pd.to\_datetime(df['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = (df['DateTime'].max() - df['DateTime'].min()).days + 1

# Calculate the % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements\_per\_day = 24 \* 60 / 5

total\_expected\_measurements = expected\_measurements\_per\_day \* num\_days

percent\_data\_captured = (len(df) / total\_expected\_measurements) \* 100

# Calculate the mean glucose concentration

mean\_glucose = df['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate the Glucose Variability (coefficient of variation)

cv\_glucose = df['CGM'].std() / mean\_glucose \* 100

# Calculate % Time in various glucose ranges

total\_time = len(df)

time\_above\_250 = len(df[df['CGM'] > 250]) / total\_time \* 100

time\_above\_180 = len(df[df['CGM'] > 180]) / total\_time \* 100

time\_in\_range = len(df[(df['CGM'] >= 70) & (df['CGM'] <= 180)]) / total\_time \* 100

time\_below\_70 = len(df[df['CGM'] < 70]) / total\_time \* 100

time\_below\_54 = len(df[df['CGM'] < 54]) / total\_time \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose,

time\_above\_250, time\_above\_180, time\_in\_range, time\_below\_70, time\_below\_54]

assistant > Here are the computed metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*92.41%\*\*

3. Mean glucose concentration: \*\*195.55 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.99\*\*

5. Glucose variability (coefficient of variation): \*\*33.46%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*19.78%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*54.27%\*\*

8. % Time in range (70-180 mg/dL): \*\*45.20%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.54%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.5

assistant > First, I'll need to inspect the contents of the uploaded file to understand the format and structure of the data. This will help me extract the necessary information to compute the metrics you've requested. Let's start by examining the file.

assistant > code\_interpreter

import pandas as pd

# Load the uploaded file to see the first few rows and understand its structure

file\_path = '/mnt/data/file-NM46iqgtZyq4VfSshhmYYc'

data = pd.read\_csv(file\_path)

data.head()

assistant > The data includes columns for glucose concentration (`CGM`), time (`time`), weekday, and a combined timestamp (`Time`). The structure indicates that glucose readings are taken at 5-minute intervals.

Next, I will:

1. Convert the `Time` column to a datetime format for easier manipulation.

2. Calculate the metrics based on this data.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate % of sensor data captured (assuming 5-minute intervals throughout the days)

expected\_measurements = 288 \* num\_days # 288 measurements per day (24 hours \* 60 minutes / 5 minutes)

actual\_measurements = len(data)

percent\_captured = (actual\_measurements / expected\_measurements) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate Glucose Variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time in different ranges

time\_above\_250 = len(data[data['CGM'] > 250]) / actual\_measurements \* 100

time\_above\_180 = len(data[data['CGM'] > 180]) / actual\_measurements \* 100

time\_in\_range = len(data[(data['CGM'] >= 70) & (data['CGM'] <= 180)]) / actual\_measurements \* 100

time\_below\_70 = len(data[data['CGM'] < 70]) / actual\_measurements \* 100

time\_below\_54 = len(data[data['CGM'] < 54]) / actual\_measurements \* 100

[

num\_days,

percent\_captured,

mean\_glucose,

gmi,

cv\_glucose,

time\_above\_250,

time\_above\_180,

time\_in\_range,

time\_below\_70,

time\_below\_54

]

assistant > Here are the computed metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*92.41%\*\*

3. Mean glucose concentration: \*\*195.55 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*7.99\*\*

5. Glucose variability (coefficient of variation): \*\*33.46%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*19.78%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*54.27%\*\*

8. % Time in range (70-180 mg/dL): \*\*45.20%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.54%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose control and sensor performance for the patient over the specified period.

# Case 7

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its structure and format. This will help in extracting the necessary data to compute the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-NgTUY4VYfi2y9c7oFntv1o'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataframe and its column names to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Full timestamp including date and time

Next, I will process this data to compute the requested metrics. I'll start by converting the `Time` column to a datetime format for easier manipulation and calculation of the number of days the sensor was active.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days\_active = (data['Time'].max() - data['Time'].min()).days + 1

num\_days\_active

assistant > The sensor was active for \*\*14 days\*\*.

Next, I will calculate the percentage of sensor data captured. This involves determining the expected number of readings per day (assuming a reading every 5 minutes) and comparing it to the actual number of readings.# Calculate expected number of readings per day (288 readings per day, one every 5 minutes)

expected\_readings\_per\_day = 24 \* 60 / 5

# Total expected readings

total\_expected\_readings = expected\_readings\_per\_day \* num\_days\_active

# Actual number of readings

actual\_readings = len(data)

# Percentage of sensor data captured

percent\_data\_captured = (actual\_readings / total\_expected\_readings) \* 100

percent\_data\_captured

assistant > The percentage of sensor data captured is approximately \*\*95.49%\*\*.

Next, I will calculate the mean glucose concentration.# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

mean\_glucose

assistant > The mean glucose concentration is approximately \*\*237.82 mg/dL\*\*.

Now, I will calculate the Glucose Management Indicator (GMI). The GMI can be estimated using the formula:

\[ \text{GMI} = 3.31 + 0.02392 \times \text{Mean Glucose} \]# Calculate Glucose Management Indicator (GMI)

GMI = 3.31 + 0.02392 \* mean\_glucose

GMI

assistant > The Glucose Management Indicator (GMI) is approximately \*\*8.999\*\*.

Next, I will calculate the glucose variability, which is the coefficient of variation (CV). The CV is defined as the standard deviation divided by the mean, expressed as a percentage:

\[ \text{CV} = \left(\frac{\text{Standard Deviation}}{\text{Mean}}\right) \times 100 \]# Calculate glucose variability (coefficient of variation, CV)

glucose\_std\_dev = data['CGM'].std()

glucose\_cv = (glucose\_std\_dev / mean\_glucose) \* 100

glucose\_cv

assistant > The glucose variability (coefficient of variation) is approximately \*\*33.57%\*\*.

Next, I will calculate the percentage of time above the target glucose range (TAR) in level 2 (>250 mg/dL) and level 1 (>180 mg/dL), the percentage of time in range (70-180 mg/dL), and the percentage of time below the target glucose range (TBR) in level 1 (<70 mg/dL) and level 2 (<54 mg/dL).# Calculate percentages of time in various glucose ranges

time\_above\_250 = (data['CGM'] > 250).sum() / actual\_readings \* 100

time\_above\_180 = (data['CGM'] > 180).sum() / actual\_readings \* 100

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).sum() / actual\_readings \* 100

time\_below\_70 = (data['CGM'] < 70).sum() / actual\_readings \* 100

time\_below\_54 = (data['CGM'] < 54).sum() / actual\_readings \* 100

time\_above\_250, time\_above\_180, time\_in\_range, time\_below\_70, time\_below\_54

assistant > Here are the percentages of time spent in various glucose ranges:

- \*\*Time above target glucose range (TAR) in level 2 (>250 mg/dL)\*\*: 41.90%

- \*\*Time above target glucose range (TAR) in level 1 (>180 mg/dL)\*\*: 71.22%

- \*\*Time in range (70-180 mg/dL)\*\*: 28.44%

- \*\*Time below target glucose range (TBR) in level 1 (<70 mg/dL)\*\*: 0.34%

- \*\*Time below target glucose range (TBR) in level 2 (<54 mg/dL)\*\*: 0.08%

Finally, I will compile all these metrics into a list as requested.# Compile all metrics into a list

metrics\_list = [

num\_days\_active, # Number of days the sensor is active

percent\_data\_captured, # % of sensor data captured

mean\_glucose, # Mean glucose concentration

GMI, # Glucose management indicator

glucose\_cv, # Glucose variability (coefficient of variation)

time\_above\_250, # % Time above target glucose range (TAR) in level 2 (>250)

time\_above\_180, # % Time above target glucose range (TAR) in level 1 (>180)

time\_in\_range, # % Time in range (70-180 mg/dL)

time\_below\_70, # % Time below target glucose range (TBR) in level 1 (<70)

time\_below\_54 # % Time below target glucose range (TBR) in level 2 (<54)

]

metrics\_list

assistant > Here are the requested metrics for the CGM data analysis:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*95.49%\*\*

3. Mean glucose concentration: \*\*237.82 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*8.999\*\*

5. Glucose variability (coefficient of variation): \*\*33.57%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*41.90%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*71.22%\*\*

8. % Time in range (70-180 mg/dL): \*\*28.44%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.34%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.08%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I'll need to examine the contents of the uploaded file to understand the data format and structure. Let's start by loading the data and taking a look at the first few rows.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-FxR4i99HuV78t8hivdZxWD'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Combined date and time

Next, I will clean the data by removing unnecessary columns and converting the `Time` column to a datetime format for easier analysis. Then, I will proceed to calculate the requested metrics.# Remove unnecessary columns and convert 'Time' to datetime format

data.drop(columns=['Unnamed: 0', 'time', 'weekday'], inplace=True)

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Check the cleaned data

data.head(), data.dtypes

assistant > The data is now cleaned and the `Time` column is in datetime format. Let's proceed with the calculations for the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

I will start by calculating the number of days the sensor is active and the percentage of sensor data captured.# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate the expected number of readings (assuming a reading every 5 minutes)

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

# Calculate the percentage of sensor data captured

percent\_data\_captured = (len(data) / total\_expected\_readings) \* 100

num\_days, percent\_data\_captured

assistant > The sensor was active for 14 days, and approximately 95.49% of the sensor data was captured.

Next, I will calculate the mean glucose concentration and the Glucose Management Indicator (GMI). The GMI is calculated using the formula:

\[ \text{GMI} = 3.31 + 0.02392 \times \text{Mean Glucose} \]# Calculate the mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

gmi = 3.31 + 0.02392 \* mean\_glucose

mean\_glucose, gmi

assistant > The mean glucose concentration is approximately 237.82 mg/dL, and the Glucose Management Indicator (GMI) is approximately 8.999.

Next, I will calculate the glucose variability, which is the coefficient of variation (CV). The CV is defined as the standard deviation divided by the mean, expressed as a percentage:

\[ \text{CV} = \left(\frac{\text{Standard Deviation}}{\text{Mean}}\right) \times 100 \]# Calculate the standard deviation of glucose levels

std\_dev\_glucose = data['CGM'].std()

# Calculate the coefficient of variation (CV)

cv\_glucose = (std\_dev\_glucose / mean\_glucose) \* 100

cv\_glucose

assistant > The glucose variability, measured as the coefficient of variation (CV), is approximately 33.57%.

Now, I will calculate the percentage of time spent in various glucose ranges:

- Above 250 mg/dL (TAR level 2)

- Above 180 mg/dL (TAR level 1)

- Between 70 and 180 mg/dL (TIR)

- Below 70 mg/dL (TBR level 1)

- Below 54 mg/dL (TBR level 2)# Calculate the percentage of time in various glucose ranges

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

tir = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

tar\_level\_2, tar\_level\_1, tir, tbr\_level\_1, tbr\_level\_2

assistant > Here are the percentages of time spent in various glucose ranges:

- % Time above target glucose range (TAR) in level 2 (>250 mg/dL): 41.90%

- % Time above target glucose range (TAR) in level 1 (>180 mg/dL): 71.22%

- % Time in range (70-180 mg/dL): 28.44%

- % Time below target glucose range (TBR) in level 1 (<70 mg/dL): 0.34%

- % Time below target glucose range (TBR) in level 2 (<54 mg/dL): 0.08%

Now, I will compile all the calculated metrics into a list as requested.# Compile all metrics into a list

metrics\_list = [

num\_days, # Number of days the sensor is active

percent\_data\_captured, # % of sensor data captured

mean\_glucose, # Mean glucose concentration

gmi, # Glucose management indicator

cv\_glucose, # Glucose variability (coefficient of variation)

tar\_level\_2, # % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_1, # % Time above target glucose range (TAR) in level 1 (>180)

tir, # % Time in range (70-180 mg/dL)

tbr\_level\_1, # % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_2 # % Time below target glucose range (TBR) in level 2 (<54)

]

metrics\_list

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*95.49%\*\*

3. Mean glucose concentration: \*\*237.82 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*8.999\*\*

5. Glucose variability (coefficient of variation): \*\*33.57%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*41.90%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*71.22%\*\*

8. % Time in range (70-180 mg/dL): \*\*28.44%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.34%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.08%\*\*

These metrics provide a comprehensive overview of the glucose management for the individual with type 1 diabetes over the period the sensor was active.

## Temperature= 0.5

assistant > First, I'll examine the contents of the uploaded file to understand its format and the data it contains. This will help in extracting the necessary information to calculate the requested metrics.

assistant > code\_interpreter

# Let's check the contents of the uploaded file to understand its format and structure

file\_path = '/mnt/data/file-TsC3L65hgbR7tUFWFhBmgX'

# Attempt to open and read the file to determine its structure

with open(file\_path, 'r') as file:

sample\_lines = [next(file) for \_ in range(10)]

sample\_lines

assistant > The file appears to be a CSV format with the following columns:

- An unnamed index column

- CGM: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- time: Time of day in HH:MM:SS format

- weekday: Day of the week

- Time: Date and time in M/D/YY H:M format

Next, I'll load the data into a DataFrame for further analysis and calculation of the metrics.import pandas as pd

# Load the data into a DataFrame

df = pd.read\_csv(file\_path)

df.head()

assistant > Now that the data is loaded into a DataFrame, I will proceed with the calculations for the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250)

7. % Time above target glucose range (TAR) in level 1 (>180)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70)

10. % Time below target glucose range (TBR) in level 2 (<54)

First, I'll preprocess the data to ensure the date and time columns are properly formatted for analysis.# Convert 'Time' column to datetime format for easier manipulation

df['Datetime'] = pd.to\_datetime(df['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = (df['Datetime'].max() - df['Datetime'].min()).days + 1

# Calculate the % of sensor data captured

expected\_measurements = 288 \* num\_days # 288 readings per day (every 5 minutes)

actual\_measurements = len(df)

percent\_data\_captured = (actual\_measurements / expected\_measurements) \* 100

# Calculate mean glucose concentration

mean\_glucose = df['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate glucose variability (coefficient of variation)

cv\_glucose = df['CGM'].std() / mean\_glucose \* 100

# Calculate % Time in various ranges

total\_time = len(df)

time\_above\_250 = len(df[df['CGM'] > 250]) / total\_time \* 100

time\_above\_180 = len(df[df['CGM'] > 180]) / total\_time \* 100

time\_in\_range = len(df[(df['CGM'] >= 70) & (df['CGM'] <= 180)]) / total\_time \* 100

time\_below\_70 = len(df[df['CGM'] < 70]) / total\_time \* 100

time\_below\_54 = len(df[df['CGM'] < 54]) / total\_time \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose,

time\_above\_250, time\_above\_180, time\_in\_range, time\_below\_70, time\_below\_54]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*95.49%\*\*

3. Mean glucose concentration: \*\*237.82 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*8.999\*\*

5. Glucose variability (coefficient of variation): \*\*33.57%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*41.90%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*71.22%\*\*

8. % Time in range (70-180 mg/dL): \*\*28.44%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*0.34%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.08%\*\*

These metrics give a comprehensive overview of the glucose control status for the patient over the recorded period.

# Case 8

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-67BNqk5JKHGadKyMqr6w7j'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of the day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will clean the data by removing unnecessary columns and converting the `Time` column to a datetime format for easier analysis. Then, I will proceed to calculate the requested metrics.# Remove unnecessary columns and convert 'Time' to datetime format

data.drop(columns=['Unnamed: 0', 'time', 'weekday'], inplace=True)

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Check the cleaned data and data types

data.head(), data.dtypes

assistant > The data is now cleaned and ready for analysis. I will proceed with calculating the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

Let's start by calculating these metrics.import numpy as np

# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate the % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements = 288 \* num\_days # 288 measurements per day (every 5 minutes in 24 hours)

actual\_measurements = len(data)

percent\_captured = (actual\_measurements / expected\_measurements) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate glucose variability (coefficient of variation)

cv\_glucose = (data['CGM'].std() / mean\_glucose) \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*91.54%\*\*

3. Mean glucose concentration: \*\*112.59 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.00\*\*

5. Glucose variability (coefficient of variation): \*\*20.57%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*0.00%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*0.92%\*\*

8. % Time in range (70-180 mg/dL): \*\*97.78%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*1.30%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

. assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-Nw1Tku1n4DYghJiLvAxc73'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column that we can ignore.

- `CGM`: The glucose concentration in mg/dL.

- `time`: The time of day.

- `weekday`: The day of the week.

- `Time`: The full date and time.

Next, I will clean the data by removing unnecessary columns and converting the `Time` column into a proper datetime format for further analysis.# Drop the unnecessary column

data.drop(columns=['Unnamed: 0', 'time', 'weekday'], inplace=True)

# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Check the data types and preview the cleaned data

data.dtypes, data.head()

assistant > The data is now cleaned with the `Time` column in datetime format and the `CGM` column representing glucose concentrations. Let's proceed to calculate the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

I will start by calculating the number of days the sensor is active and the percentage of sensor data captured.# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate the expected number of readings (assuming readings every 5 minutes)

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

# Calculate the percentage of sensor data captured

percent\_data\_captured = (len(data) / total\_expected\_readings) \* 100

num\_days, percent\_data\_captured

assistant > The sensor was active for 14 days, and 91.54% of the sensor data was captured.

Next, I will calculate the mean glucose concentration and the Glucose Management Indicator (GMI). The GMI is calculated using the formula:

\[ \text{GMI} = 3.31 + 0.02392 \times \text{Mean Glucose} \]# Calculate the mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

gmi = 3.31 + 0.02392 \* mean\_glucose

mean\_glucose, gmi

assistant > The mean glucose concentration is approximately 112.59 mg/dL, and the Glucose Management Indicator (GMI) is approximately 6.00.

Next, I will calculate the glucose variability, which is the coefficient of variation (CV). The CV is defined as the ratio of the standard deviation to the mean, expressed as a percentage:

\[ \text{CV} = \left(\frac{\text{Standard Deviation}}{\text{Mean}}\right) \times 100 \]# Calculate the standard deviation of glucose concentrations

std\_dev\_glucose = data['CGM'].std()

# Calculate the coefficient of variation (CV)

cv\_glucose = (std\_dev\_glucose / mean\_glucose) \* 100

cv\_glucose

assistant > The glucose variability, measured as the coefficient of variation (CV), is approximately 20.57%.

Now, I will calculate the percentages of time spent in various glucose ranges:

- % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

- % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

- % Time in range (70-180 mg/dL)

- % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

- % Time below target glucose range (TBR) in level 2 (<54 mg/dL)# Calculate the percentages of time in various glucose ranges

total\_readings = len(data)

tar\_level\_2 = (data['CGM'] > 250).sum() / total\_readings \* 100

tar\_level\_1 = (data['CGM'] > 180).sum() / total\_readings \* 100

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).sum() / total\_readings \* 100

tbr\_level\_1 = (data['CGM'] < 70).sum() / total\_readings \* 100

tbr\_level\_2 = (data['CGM'] < 54).sum() / total\_readings \* 100

tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2

assistant > Here are the percentages of time spent in various glucose ranges:

- % Time above target glucose range (TAR) in level 2 (>250 mg/dL): 0.00%

- % Time above target glucose range (TAR) in level 1 (>180 mg/dL): 0.92%

- % Time in range (70-180 mg/dL): 97.78%

- % Time below target glucose range (TBR) in level 1 (<70 mg/dL): 1.30%

- % Time below target glucose range (TBR) in level 2 (<54 mg/dL): 0.00%

Now, I will compile all these metrics into a single list as requested.# Compile all metrics into a list

metrics = [

num\_days, # Number of days the sensor is active

percent\_data\_captured, # % of sensor data captured

mean\_glucose, # Mean glucose concentration

gmi, # Glucose management indicator

cv\_glucose, # Glucose variability (coefficient of variation)

tar\_level\_2, # % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_1, # % Time above target glucose range (TAR) in level 1 (>180)

time\_in\_range, # % Time in range (70-180 mg/dL)

tbr\_level\_1, # % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_2 # % Time below target glucose range (TBR) in level 2 (<54)

]

metrics

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*91.54%\*\*

3. Mean glucose concentration: \*\*112.59 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.00\*\*

5. Glucose variability (coefficient of variation): \*\*20.57%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*0.00%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*0.92%\*\*

8. % Time in range (70-180 mg/dL): \*\*97.78%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*1.30%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.00%\*\*

These metrics provide a comprehensive overview of the glucose management for the patient over the period the sensor was active. If you need further analysis or have additional questions, feel free to ask!

## Temperature= 0.5

assistant > First, I'll review the contents of the uploaded file to understand its structure and format. This will help me extract the necessary data to calculate the requested metrics. Let's start by loading the data from the file.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file to see its structure

file\_path = '/mnt/data/file-8EKSdobnhn5wAcg48ppS9v'

data = pd.read\_csv(file\_path)

data.head()

assistant > The data includes columns for glucose concentration (`CGM`), time of day (`time`), day of the week (`weekday`), and a combined date-time column (`Time`). I will use these columns to compute the requested metrics.

Let's start by processing the data:

1. Convert the `Time` column to a datetime format.

2. Calculate the number of days the sensor was active.

3. Compute the percentage of sensor data captured, assuming the sensor should ideally capture data every 5 minutes.

4. Calculate the mean glucose concentration.

5. Determine the Glucose Management Indicator (GMI).

6. Calculate the glucose variability (coefficient of variation).

7. Calculate the percentage of time above target glucose range (>250 mg/dL and >180 mg/dL).

8. Calculate the percentage of time in the target range (70-180 mg/dL).

9. Calculate the percentage of time below target glucose range (<70 mg/dL and <54 mg/dL).

Let's proceed with these calculations.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor was active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate the percentage of sensor data captured

expected\_measurements = num\_days \* 24 \* 60 / 5 # 5-minute intervals per day

actual\_measurements = len(data)

sensor\_data\_captured = (actual\_measurements / expected\_measurements) \* 100

# Calculate the mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

gmi = 3.31 + (0.02392 \* mean\_glucose)

# Calculate the glucose variability (coefficient of variation)

glucose\_cv = data['CGM'].std() / mean\_glucose \* 100

# Calculate the percentage of time in different glucose ranges

tar\_level2 = (data['CGM'] > 250).mean() \* 100

tar\_level1 = (data['CGM'] > 180).mean() \* 100

tir = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

tbr\_level1 = (data['CGM'] < 70).mean() \* 100

tbr\_level2 = (data['CGM'] < 54).mean() \* 100

[num\_days, sensor\_data\_captured, mean\_glucose, gmi, glucose\_cv, tar\_level2, tar\_level1, tir, tbr\_level1, tbr\_level2]

assistant > Here are the calculated metrics for the CGM data:

1. \*\*Number of days the sensor is active:\*\* 14 days

2. \*\*% of sensor data captured:\*\* 91.54%

3. \*\*Mean glucose concentration:\*\* 112.59 mg/dL

4. \*\*Glucose management indicator (GMI):\*\* 6.00

5. \*\*Glucose variability (coefficient of variation):\*\* 20.57%

6. \*\*% Time above target glucose range (TAR) in level 2 (>250 mg/dL):\*\* 0.00%

7. \*\*% Time above target glucose range (TAR) in level 1 (>180 mg/dL):\*\* 0.92%

8. \*\*% Time in range (70-180 mg/dL):\*\* 97.78%

9. \*\*% Time below target glucose range (TBR) in level 1 (<70 mg/dL):\*\* 1.30%

10. \*\*% Time below target glucose range (TBR) in level 2 (<54 mg/dL):\*\* 0.00%

These values provide a comprehensive overview of the glucose control for this patient over the period the sensor was active.

# Case 9

## Temperature= 0

assistant > First, I will examine the contents of the uploaded file to understand its format and structure. This will help me extract the necessary data to compute the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-B5yFDSoZEtJZsaGeDYBgWQ'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process this data to compute the requested metrics. I'll start by converting the `Time` column to a datetime format for easier manipulation and then proceed with the calculations.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured

# Assuming data should be captured every 5 minutes, calculate total expected readings

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

percent\_data\_captured = (len(data) / total\_expected\_readings) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* mean glucose

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate glucose variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the computed metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*99.16%\*\*

3. Mean glucose concentration: \*\*148.84 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.87\*\*

5. Glucose variability (coefficient of variation): \*\*38.07%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*6.05%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*21.59%\*\*

8. % Time in range (70-180 mg/dL): \*\*76.41%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*2.00%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.25%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I'll examine the contents of the uploaded file to understand its format and the data it contains. This will help in extracting the necessary information to compute the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-My2YCDEqnBkdV4PFavADhp'

# Attempt to open and read the file to determine its format (CSV, Excel, etc.)

import pandas as pd

# Try reading the file with common formats

def try\_read\_file(file\_path):

try:

# Try reading as CSV

data = pd.read\_csv(file\_path)

return data, "csv"

except Exception as e\_csv:

try:

# Try reading as Excel

data = pd.read\_excel(file\_path)

return data, "excel"

except Exception as e\_excel:

return None, "unknown"

data, file\_type = try\_read\_file(file\_path)

file\_type, data.head() if data is not None else "Data could not be read"

assistant > The file is in CSV format and contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Glucose concentration in mg/dL

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Date and time combined

Next, I will process this data to compute the requested metrics.import numpy as np

# Convert 'Time' column to datetime format for easier manipulation

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements\_per\_day = 24 \* 60 / 5

actual\_measurements\_per\_day = data.groupby(data['Time'].dt.date).size().mean()

percent\_data\_captured = (actual\_measurements\_per\_day / expected\_measurements\_per\_day) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI = 3.31 + 0.02392 × mean glucose in mg/dL

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate glucose variability (coefficient of variation)

cv\_glucose = (data['CGM'].std() / mean\_glucose) \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

# Compile all metrics into a list

metrics = [

num\_days,

percent\_data\_captured,

mean\_glucose,

gmi,

cv\_glucose,

tar\_level\_2,

tar\_level\_1,

time\_in\_range,

tbr\_level\_1,

tbr\_level\_2

]

metrics

assistant > Here are the computed metrics based on the CGM data provided:

1. \*\*Number of days the sensor is active:\*\* 14 days

2. \*\*% of sensor data captured:\*\* 99.16%

3. \*\*Mean glucose concentration:\*\* 148.84 mg/dL

4. \*\*Glucose management indicator (GMI):\*\* 6.87

5. \*\*Glucose variability (coefficient of variation):\*\* 38.07%

6. \*\*% Time above target glucose range (TAR) in level 2 (>250 mg/dL):\*\* 6.05%

7. \*\*% Time above target glucose range (TAR) in level 1 (>180 mg/dL):\*\* 21.59%

8. \*\*% Time in range (70-180 mg/dL):\*\* 76.41%

9. \*\*% Time below target glucose range (TBR) in level 1 (<70 mg/dL):\*\* 2.00%

10. \*\*% Time below target glucose range (TBR) in level 2 (<54 mg/dL):\*\* 0.25%

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.Success

## Temperature= 0.5

assistant > First, I'll need to examine the contents of the uploaded file to understand the data format and structure. I will then proceed to calculate the metrics as requested. Let's start by loading and inspecting the data file.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-3Ry4ZkxoZSpabjSN7CkFnh'

data = pd.read\_csv(file\_path)

# Display the first few rows of the data to understand its structure

data.head(), data.columns

assistant > The data contains the following columns:

- `Unnamed: 0`: An index column, likely not necessary for analysis.

- `CGM`: Continuous Glucose Monitoring values in mg/dL.

- `time`: Time of the day.

- `weekday`: Day of the week.

- `Time`: Combined date and time.

Next, I'll calculate the requested metrics based on the `CGM` and `Time` columns. I'll start by parsing the `Time` column to datetime format for better manipulation and calculation of the metrics.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Check the datatype to confirm conversion and view the range of dates

data['Time'].dtype, data['Time'].min(), data['Time'].max()

assistant > The `Time` column has been successfully converted to datetime format, covering the period from January 8, 2023, to January 21, 2023.

Now, I will proceed with the calculation of the requested metrics:

1. Number of days the sensor is active

2. Percentage of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. Percentage of time above target glucose range (>250 mg/dL)

7. Percentage of time above target glucose range (>180 mg/dL)

8. Percentage of time in target range (70-180 mg/dL)

9. Percentage of time below target glucose range (<70 mg/dL)

10. Percentage of time below target glucose range (<54 mg/dL)

Let's start by calculating these metrics.import numpy as np

# 1. Number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# 2. % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements = 24 \* 60 / 5 \* num\_days

percent\_data\_captured = (len(data) / expected\_measurements) \* 100

# 3. Mean glucose concentration

mean\_glucose = data['CGM'].mean()

# 4. Glucose management indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* mean\_glucose

gmi = 3.31 + 0.02392 \* mean\_glucose

# 5. Glucose variability (coefficient of variation)

cv\_glucose = (data['CGM'].std() / mean\_glucose) \* 100

# 6. % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data['CGM'] > 250).sum() / len(data) \* 100

# 7. % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data['CGM'] > 180).sum() / len(data) \* 100

# 8. % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).sum() / len(data) \* 100

# 9. % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data['CGM'] < 70).sum() / len(data) \* 100

# 10. % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data['CGM'] < 54).sum() / len(data) \* 100

# Collect all metrics in a list

metrics = [

num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose,

tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2

]

metrics

assistant > Here are the calculated metrics for the CGM data:

1. \*\*Number of days the sensor is active\*\*: 14 days

2. \*\*Percentage of sensor data captured\*\*: 99.16%

3. \*\*Mean glucose concentration\*\*: 148.84 mg/dL

4. \*\*Glucose management indicator (GMI)\*\*: 6.87

5. \*\*Glucose variability (coefficient of variation)\*\*: 38.07%

6. \*\*Percentage of time above target glucose range (TAR) in level 2 (>250 mg/dL)\*\*: 6.05%

7. \*\*Percentage of time above target glucose range (TAR) in level 1 (>180 mg/dL)\*\*: 21.59%

8. \*\*Percentage of time in target range (70-180 mg/dL)\*\*: 76.41%

9. \*\*Percentage of time below target glucose range (TBR) in level 1 (<70 mg/dL)\*\*: 2.00%

10. \*\*Percentage of time below target glucose range (TBR) in level 2 (<54 mg/dL)\*\*: 0.25%

These metrics provide a comprehensive overview of the glucose control and sensor performance for the patient over the 14-day period.

# Case 10

## Temperature= 0

assistant > First, I'll need to examine the contents of the uploaded file to understand the format and structure of the data. This will help me extract the necessary information to calculate the requested metrics. Let's start by loading and previewing the data.

assistant > code\_interpreter

import pandas as pd

# Load the data from the uploaded file

file\_path = '/mnt/data/file-FdHfRAu7KdZZDnNNHx2r9r'

data = pd.read\_csv(file\_path)

# Display the first few rows of the dataset to understand its structure

data.head(), data.columns

assistant > The dataset contains the following columns:

- `Unnamed: 0`: An index column

- `CGM`: Continuous Glucose Monitoring data (glucose concentration in mg/dL)

- `time`: Time of day

- `weekday`: Day of the week

- `Time`: Full timestamp including date and time

Next, I will process the data to calculate the requested metrics. I'll start by converting the `Time` column to a datetime format for easier manipulation and then proceed with the calculations.# Convert 'Time' column to datetime format

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = data['Time'].dt.date.nunique()

# Calculate the % of sensor data captured

# Assuming data should be captured every 5 minutes, calculate the expected number of readings per day

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

percent\_data\_captured = (len(data) / total\_expected\_readings) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* mean glucose

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate glucose variability (coefficient of variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

[num\_days, percent\_data\_captured, mean\_glucose, gmi, cv\_glucose, tar\_level\_2, tar\_level\_1, time\_in\_range, tbr\_level\_1, tbr\_level\_2]

assistant > Here are the calculated metrics for the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*97.37%\*\*

3. Mean glucose concentration: \*\*140.25 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.66\*\*

5. Glucose variability (coefficient of variation): \*\*27.13%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*1.22%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*12.40%\*\*

8. % Time in range (70-180 mg/dL): \*\*84.51%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*3.08%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.76%\*\*

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.1

assistant > First, I will examine the contents of the uploaded file to understand its format and the data it contains. This will help me extract the necessary information to calculate the requested metrics.

assistant > code\_interpreter

# Let's check the content of the uploaded file to understand its structure and format.

file\_path = '/mnt/data/file-5axL6LyfaDAfnY33n63hyT'

# Attempt to read the file assuming it might be a common data file format like CSV or Excel.

import pandas as pd

try:

# Try reading as CSV

data = pd.read\_csv(file\_path)

except Exception as e\_csv:

try:

# Try reading as Excel

data = pd.read\_excel(file\_path)

except Exception as e\_excel:

# Output errors if both attempts fail

data = None

errors = {"csv\_error": str(e\_csv), "excel\_error": str(e\_excel)}

data.head() if data is not None else errors

assistant > The data appears to be structured with columns indicating the glucose measurement (`CGM`), time of the day (`time`), day of the week (`weekday`), and a combined date and time column (`Time`). I will now proceed to calculate the requested metrics based on this data.from datetime import datetime

# Convert 'Time' column to datetime format for easier manipulation

data['Time'] = pd.to\_datetime(data['Time'], format='%m/%d/%y %H:%M')

# Calculate the number of days the sensor is active

num\_days = (data['Time'].max() - data['Time'].min()).days + 1

# Calculate the % of sensor data captured

# Assuming data should be captured every 5 minutes, calculate total expected readings

expected\_readings\_per\_day = 24 \* 60 / 5

total\_expected\_readings = expected\_readings\_per\_day \* num\_days

percent\_captured = (len(data) / total\_expected\_readings) \* 100

# Calculate mean glucose concentration

mean\_glucose = data['CGM'].mean()

# Calculate Glucose Management Indicator (GMI)

# GMI formula: 3.31 + 0.02392 \* Mean glucose in mg/dL

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate Glucose Variability (Coefficient of Variation)

cv\_glucose = data['CGM'].std() / mean\_glucose \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

tar\_level\_2 = (data['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

tar\_level\_1 = (data['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((data['CGM'] >= 70) & (data['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

tbr\_level\_1 = (data['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

tbr\_level\_2 = (data['CGM'] < 54).mean() \* 100

# Compile all metrics into a list

metrics = [

num\_days,

percent\_captured,

mean\_glucose,

gmi,

cv\_glucose,

tar\_level\_2,

tar\_level\_1,

time\_in\_range,

tbr\_level\_1,

tbr\_level\_2

]

metrics

assistant > Here are the calculated metrics based on the CGM data provided:

1. \*\*Number of days the sensor is active:\*\* 14 days

2. \*\*% of sensor data captured:\*\* 97.37%

3. \*\*Mean glucose concentration:\*\* 140.25 mg/dL

4. \*\*Glucose Management Indicator (GMI):\*\* 6.66

5. \*\*Glucose variability (coefficient of variation):\*\* 27.13%

6. \*\*% Time above target glucose range (TAR) in level 2 (>250 mg/dL):\*\* 1.22%

7. \*\*% Time above target glucose range (TAR) in level 1 (>180 mg/dL):\*\* 12.40%

8. \*\*% Time in range (70-180 mg/dL):\*\* 84.51%

9. \*\*% Time below target glucose range (TBR) in level 1 (<70 mg/dL):\*\* 3.08%

10. \*\*% Time below target glucose range (TBR) in level 2 (<54 mg/dL):\*\* 0.76%

These values provide a comprehensive overview of the glucose management for the patient over the period the sensor was active.

## Temperature= 0.5

assistant > First, I will examine the contents of the uploaded file to understand its format and structure. This will help in extracting and analyzing the CGM data to compute the requested metrics.

assistant > code\_interpreter

# Let's first examine the file to understand its format and structure.

file\_path = '/mnt/data/file-QB84kNV8PRnhVA4wFu1ERu'

# Checking the first few lines to understand the structure of the data.

with open(file\_path, 'r') as file:

lines = [next(file) for \_ in range(5)]

lines

assistant > The file appears to be a CSV with columns indicating the index, CGM glucose readings, time of day, day of the week, and date and time. The columns are labeled as:

- An unnamed index column

- CGM (glucose concentration in mg/dL)

- Time of day (formatted as HH:MM:SS)

- Weekday

- Time (formatted as MM/DD/YY HH:MM)

Next, I will load the data into a DataFrame for further analysis and computation of the requested metrics.import pandas as pd

# Load the data into a DataFrame

df = pd.read\_csv(file\_path)

# Parse the 'Time' column to datetime format for easier manipulation

df['Time'] = pd.to\_datetime(df['Time'], format='%m/%d/%y %H:%M')

# Check the first few rows of the DataFrame to confirm correct loading and parsing

df.head()

assistant > The data has been successfully loaded into a DataFrame. Now, I will calculate the requested metrics:

1. Number of days the sensor is active

2. % of sensor data captured

3. Mean glucose concentration

4. Glucose management indicator (GMI)

5. Glucose variability (coefficient of variation)

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

8. % Time in range (70-180 mg/dL)

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL)import numpy as np

# Calculate the number of days the sensor is active

num\_days = df['Time'].dt.date.nunique()

# Calculate the % of sensor data captured (assuming data should be captured every 5 minutes)

expected\_measurements\_per\_day = 24 \* 60 / 5

actual\_measurements = df.groupby(df['Time'].dt.date).size()

percent\_captured = (actual\_measurements / expected\_measurements\_per\_day).mean() \* 100

# Calculate the mean glucose concentration

mean\_glucose = df['CGM'].mean()

# Calculate the Glucose Management Indicator (GMI)

# GMI = 3.31 + 0.02392 \* mean\_glucose

gmi = 3.31 + 0.02392 \* mean\_glucose

# Calculate the glucose variability (coefficient of variation)

cv\_glucose = df['CGM'].std() / mean\_glucose \* 100

# Calculate % Time above target glucose range (TAR) in level 2 (>250 mg/dL)

tar\_level2 = (df['CGM'] > 250).mean() \* 100

# Calculate % Time above target glucose range (TAR) in level 1 (>180 mg/dL)

tar\_level1 = (df['CGM'] > 180).mean() \* 100

# Calculate % Time in range (70-180 mg/dL)

time\_in\_range = ((df['CGM'] >= 70) & (df['CGM'] <= 180)).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 1 (<70 mg/dL)

tbr\_level1 = (df['CGM'] < 70).mean() \* 100

# Calculate % Time below target glucose range (TBR) in level 2 (<54 mg/dL)

tbr\_level2 = (df['CGM'] < 54).mean() \* 100

[

num\_days,

percent\_captured,

mean\_glucose,

gmi,

cv\_glucose,

tar\_level2,

tar\_level1,

time\_in\_range,

tbr\_level1,

tbr\_level2

]

assistant > Here are the computed metrics based on the CGM data:

1. Number of days the sensor is active: \*\*14 days\*\*

2. % of sensor data captured: \*\*97.37%\*\*

3. Mean glucose concentration: \*\*140.25 mg/dL\*\*

4. Glucose management indicator (GMI): \*\*6.66\*\*

5. Glucose variability (coefficient of variation): \*\*27.13%\*\*

6. % Time above target glucose range (TAR) in level 2 (>250 mg/dL): \*\*1.22%\*\*

7. % Time above target glucose range (TAR) in level 1 (>180 mg/dL): \*\*12.40%\*\*

8. % Time in range (70-180 mg/dL): \*\*84.51%\*\*

9. % Time below target glucose range (TBR) in level 1 (<70 mg/dL): \*\*3.08%\*\*

10. % Time below target glucose range (TBR) in level 2 (<54 mg/dL): \*\*0.76%\*\*

These values provide a comprehensive overview of the glucose management and sensor performance over the period analyzed.