1. We need to have openAI workbench

Generate telemetry data for biking

1. Simple prompt to generate 10 records

A screenshot of a computer

AI-generated content may be incorrect.

1. Generate prompt to 1000 records

A screenshot of a white text

AI-generated content may be incorrect.

1. Generate code for generating the sample dataset

Please write some python code that will generate a synthetic dataset of 1000 records that are similar to this example

[

{

"CustomerID": 100401,

"StartTime": "022224 07:00:00",

"EndTime": "022224 07:45:00",

"AvgHR": 148,

"MaxHR": 167,

"AvgMPH": 14,

"MaxMPH": 24,

"AvgOxygen": 94,

"ElevationGain": 760,

"KCal": 465,

"Weight": 180,

"Age": 32

}

]

StartTime should be between 05:00:00 and 23:00:00.

EndTime is usually between 10 and 90 minutes later with most sessions around 30 minutes.

AvgHR is usually between 135 and 172.

MaxHR is never more than 180 or less than 135.

AvgMPH is usually around 10 to 20 MPH.

MaxMPH is never more than 50 MPH.

AvgOxygen is usually between 94 and 99 with the occasional outlier closer to 90.

ElevationGain will be between 0 and 4000 feet.

KCal is highly dependent on the StartTime and EndTime as well as AvgMPH and ElevationGain as well as the weight. A 30 minute session for a 200LB person with 500 feet of elevation gain is usually around 380, higher durations, speeds, or elevation gains should relate to higher KCal numbers.

Ages will range from 16 to 76 with ages between 18 and 38 having the highest MPH and ElevationGain

Python code generated

A screenshot of a computer code

AI-generated content may be incorrect.

import random

import datetime

import json

import os

def generate\_time():

    # Generate a random time between 05:00:00 to 23:00:00

    start\_hour = random.randint(5, 22)

    start\_minute = random.randint(0, 59)

    start\_time = datetime.time(start\_hour, start\_minute)

    # Duration between 10 to 90 minutes, most around 30 minutes

    duration = int(random.triangular(10, 90, 30))

    end\_time = (datetime.datetime.combine(datetime.date.today(), start\_time) + datetime.timedelta(minutes=duration)).time()

    return (start\_time.strftime("%H:%M:%S"), end\_time.strftime("%H:%M:%S"))

def calculate\_kcal(start\_hour, duration, avg\_mph, elevation\_gain, weight):

    # A simplified calculation for KCal

    # Base calorie consumption

    kcal = 0.2 \* weight \* (duration/60) \* ((avg\_mph/5) + (elevation\_gain/500))

    return int(kcal)

def generate\_record(customer\_id):

    start\_time, end\_time = generate\_time()

    avg\_hr = random.randint(135, 172)

    max\_hr = random.randint(max(avg\_hr, 135), 180)

    avg\_mph = random.randint(10, 20)

    max\_mph = random.randint(avg\_mph, 50)

    avg\_oxygen = random.choice([random.randint(94, 99)] + [90] \* 5)  # more likely to be 94-99

    elevation\_gain = random.randint(0, 4000)

    weight = random.randint(100, 250)  # Assuming a weight range

    age = random.randint(16, 76)

    end\_hour = int(end\_time.split(':')[0])

    start\_hour = int(start\_time.split(':')[0])

    duration = (end\_hour\*60 + int(end\_time.split(':')[1])) - (start\_hour\*60 + int(start\_time.split(':')[1]))

    # Calculate KCal with a simplified model

    kcal = calculate\_kcal(start\_hour, duration, avg\_mph, elevation\_gain, weight)

    return {

        "CustomerID": customer\_id,

        "StartTime": "022224 " + start\_time,

        "EndTime": "022224 " + end\_time,

        "AvgHR": avg\_hr,

        "MaxHR": max\_hr,

        "AvgMPH": avg\_mph,

        "MaxMPH": max\_mph,

        "AvgOxygen": avg\_oxygen,

        "ElevationGain": elevation\_gain,

        "KCal": kcal,

        "Weight": weight,

        "Age": age

    }

# Generate 1000 records

data = [generate\_record(100401 + i) for i in range(1000)]

# Ensure the output directory exists

output\_dir = "output"

os.makedirs(output\_dir, exist\_ok=True)

# Output as JSON string to a file

output\_file\_path = os.path.join(output\_dir, "generated\_data.json")

with open(output\_file\_path, 'w') as f:

    json.dump(data, f, indent=2)

print(f"Data has been written to {output\_file\_path}")