t-patient-data-simulation-s2152880

May 1, 2025

1 0. Setup and Configuration

To prepare the runtime environment for AI-assisted patient simulation and analysis by:

- Installing required dependencies
- Initializing credentials for Azure OpenAI and Hugging Face
- Validating API keys via secure secrets.json loading

Step 1: Install Core Dependencies

Install the essential Python libraries needed for GenAI data generation and EDA.

[3]:
```
# Install Core Dependencies
!pip install -q openai numpy pandas
!pip install -q ydata-profiling
!pip install -q transformers
!pip install tqdm
```

```
  Preparing metadata (setup.py) … done
                         400.1/400.1 kB
9.7 MB/s eta 0:00:00
                         296.5/296.5 kB
24.2 MB/s eta 0:00:00
                         687.8/687.8 kB
31.1 MB/s eta 0:00:00
                         105.4/105.4 kB
10.0 MB/s eta 0:00:00
                         43.2/43.2 kB
3.5 MB/s eta 0:00:00
                         4.5/4.5 MB
107.5 MB/s eta 0:00:00
  Building wheel for htmlmin (setup.py) … done
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(4.67.1)
```

Step 2: Set Working Directory

All generated data files (e.g., raw outputs, datasets, summaries) will be saved to this path for clarity and version control.

```
[4]: # Set Working Directory
my_file_path = "/content/drive/MyDrive/UM Data Science Course Information/
↪WQD7005/Assignment Project/"
```

Step 3: Setup Hugging Face Token

plan to access transformer models (e.g., MiniLM, BERT-tiny), login to Hugging Face Hub is required.

```
[5]: # Setup Hugging Face Token
from huggingface_hub import notebook_login
notebook_login()
```

```
VBox(children=(HTML(value='<center> <img\nsrc=https://huggingface.co/front/
↪assets/huggingface_logo-noborder.sv…
```

Step 4: Securely Load Azure API Credentials

Using secrets.json avoids hardcoding sensitive information. This supports secure API usage and easier sharing of my notebook.

```
[ ]: # Securely Load Azure API Credentials
# Azure endpoint and keys
import json

# Load secrets.json after upload
with open(my_file_path+"secrets.json", "r") as f:
    secrets = json.load(f)

endpoint = secrets["AZURE_ENDPOINT"]
subscription_key = secrets["AZURE_KEY"]
```

Step 5: Configure Azure OpenAI Client and Define GPT Prompt Wrapper

- **api_version**: ensures compatibility with latest GPT-4o deployment and
- **deployment**: name matches my Azure OpenAI Studio setting.
- **model_prompt**: This utility function encapsulates the entire GPT-4o interaction, allowing clean and reusable prompting for various sections (data simulation, summarization, classification, etc.)

```
[ ]: # Import Supporting Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, timedelta
from openai import AzureOpenAI
import random
import time
```

```python
# Configure Azure OpenAI Client
api_version = "2024-12-01-preview"
deployment = "gpt-4o"
client = AzureOpenAI(
    api_version=api_version,
    azure_endpoint=endpoint,
    api_key=subscription_key,
)
# Prompt execution wrapper for reuse
def model_prompt(prompt, system_prompt="Act as a professional clinicians.",␣
 ↪temperature=0.7, max_tokens=4096):
    response = client.chat.completions.create(
        model=deployment,
        messages=[
            {"role": "system", "content": system_prompt},
            {"role": "user", "content": prompt}
        ],
        max_tokens=max_tokens,
        temperature=temperature,
    )
    return response.choices[0].message.content
```

Step 6: Single Sample Data Generation via GPT (Validation Prompt)

To verify the response structure of GPT-4o by generating a realistic single-patient daily monitoring record, ensuring the output conforms to expected JSON schema for later batch generation.

[ ]:
```python
# Single Sample Data Generation via model
data_prompt = """
Generate a single, realistic patient monitoring record for one randomly␣
 ↪selected adult patient.

Provide the following fields:
- oxygen_saturation (in %)
- heart_rate (in bpm)
- temperature (in °C)
- blood_pressure (systolic/diastolic, e.g. "120/80")
- weight (in kg)
- blood_glucose (in mg/dL)

At the end, include a brief clinical_note (1-2 sentences, max 30 words)␣
 ↪summarizing the patient status based on the values above. Use professional␣
 ↪clinical tone with realistic variation (e.g. stable, recovering, mild␣
 ↪concerns).

Output as a valid JSON object with keys:
oxygen_saturation, heart_rate, temperature, blood_pressure, weight,␣
 ↪blood_glucose, clinical_note.
```

```
Constraints:
- Only output one JSON object.
- No markdown or explanation.
- Include realistic variation across different health conditions (e.g. fatigue,␣
↪post-op, dietary changes, stress).
- Ensure all fields are complete, no missing values.
"""

print(model_prompt(data_prompt))
```


```
{
  "oxygen_saturation": 95,
  "heart_rate": 88,
  "temperature": 37.5,
  "blood_pressure": "130/85",
  "weight": 72,
  "blood_glucose": 145,
  "clinical_note": "Patient exhibits mild hyperglycemia and elevated blood
pressure; overall vitals suggest moderate stress or dietary changes. Monitoring
recommended for further stabilization."
}
```


2 1. Dataset Simulation using GenAI

This section outlines the complete process of generating synthetic inpatient monitoring data using a GenAI-powered approach. It simulates a realistic clinical setting where multiple patients are monitored daily for up to 30 days. Vital signs and clinical notes are generated based on assigned medical scenarios, ensuring medically coherent trends suitable for downstream exploratory analysis and machine learning modeling.

1: Prompt Engineering with Clinical Context

The prompt defines the required fields (vital signs + clinical note), instructs model on how to simulate recovery trends, controls for missing value logic, and constrains the output to valid JSON without extra formatting.

2: Generate and Save Raw Outputs

To generate 30-day clinical data per patient using model and safely store each raw response as .txt for later parsing.

3: Parse and Build the Final Dataset

To transform model-generated .txt files into a clean and structured DataFrame, validating JSON format and extracting fields.

4: Final Export to CSV

To save the simulated and parsed data in a structured format suitable for downstream analytics and visualization.

Step 1: Configuration and Library Setup

Import essential libraries (e.g., pandas, numpy, json, re, glob) and define basic configuration. This ensures the simulation pipeline runs in a clean and reproducible environment.

```
[ ]: import pandas as pd
import numpy as np
import json
import time
import re
import os
import glob

# Configuration
num_patients = 500
start_date_str = "2025-01-01"
raw_output_dir = os.path.join(my_file_path, "data")
os.makedirs(raw_output_dir, exist_ok=True)
```

Step 2: Prompt Engineering with Clinical Context

Design a dynamic prompt that tells model GPT-4o: 1. To act as a clinical simulation engine

2. To choose one of 4 scenarios (e.g., post-surgery, infection, chronic illness, or acute deterioration)
3. To generate daily vitals and notes for each patient with 10–30 days of monitoring
4. To follow professional tone and simulate realistic recovery or decline
5. To occasionally omit 1 field (e.g., temperature, weight, blood_glucose) on 1–3 days

This prompt ensures data is diverse, realistic, and medically grounded.

```
[ ]: # === Prompt Generator 1 ===
def generate_patient_prompt(patient_id, start_date):
    return f"""
You are a clinical simulation engine.

Generate a valid JSON array of **daily hospital monitoring records** for␣
 ↪patient {patient_id}, starting from {start_date} (YYYY-MM-DD).

---

SCENARIO & DURATION INSTRUCTIONS:
- Randomly choose ONE of the following clinical scenarios for the patient:
  1. **Post-surgery recovery** → 18-30 days
  2. **Moderate infection (e.g. pneumonia)** → 15-25 days
  3. **Chronic illness flare-up (e.g. diabetes complications, hypertension␣
 ↪crisis)** → 20-30 days
```

```
 4. **Acute deterioration (e.g. sepsis, cardiac failure)** → 10-15 days (with␣
↪possible death or ICU transfer)

- Based on the chosen scenario, simulate a realistic number of monitoring days␣
↪accordingly.
- Most patients should remain under monitoring for **more than 20 days**.
- Only a small number of cases should stop early due to rapid recovery or␣
↪deterioration.
- If the patient recovers or deteriorates, you may stop early (but never before␣
↪day 10).

---

EACH RECORD MUST INCLUDE:
- date
- oxygen_saturation (in %)
- heart_rate (in bpm)
- temperature (in °C)
- blood_pressure (format: "systolic/diastolic")
- weight (in kg)
- blood_glucose (in mg/dL)
- clinical_note: a brief clinical_note (2-3 sentences, max 35 words)␣
↪summarizing the patient status based on the values above. Use professional␣
↪clinical tone with realistic variation

---

MISSING DATA RULES:
- On no more than 3 random days, omit 1 of: temperature, weight, blood_glucose.
- Do not use nulls or placeholders. Simply omit the key entirely.

---

OUTPUT RULES:
- Return only a **valid JSON array** of N records (N = 10 to 30 depending on␣
↪scenario).
- No markdown, No explanation, No code block.
- Just pure JSON.
"""
```

[ ]:
```
# === Prompt Generator 2 ===
# Monitor each patient's 30-day record of prompts
def generate_patient_prompt_30days(patient_id, start_date):
    return f"""
Generate a JSON array of 30 daily monitoring records for patient {patient_id},␣
↪starting from {start_date} (YYYY-MM-DD).
```

```
Each record must include:
- date
- oxygen_saturation (%)
- heart_rate (bpm)
- temperature (°C)
- blood_pressure ("systolic/diastolic")
- weight (kg)
- blood_glucose (mg/dL)
- clinical_note (2-3 sentences in professional tone, max 35 words)

Instructions:
- Choose a realistic medical scenario (e.g. infection, post-surgery, pregnancy,␣
↪anxiety, fatigue)
- Simulate realistic changes over 30 days: early-stage symptoms → improvement →␣
↪stabilization or recovery
- Use only professional clinical language, describing daily changes, recovery,␣
↪and trends
- Avoid device/system mentions

Missing Data Instructions:
- Omit at most 1 field per day, no more than 3 total days
- Only temperature, weight, and blood_glucose may be missing
- Do not use null/empty values-omit keys entirely

Output only a valid JSON array of 30 objects. No markdown, no code block, no␣
↪explanation.
"""
```

Step 3: Generate and Save Raw Output

Loop over all patient IDs and:

1. Send the prompt to GPT-4o
2. Receive the response (a JSON array)
3. Save the raw output to a .txt file for traceability and debugging

Each patient gets one raw output file (data_output_Pxxxx.txt) stored safely.

[ ]:
```python
# === Phase 1: Generation and Save Raw Output ===
for pid in range(1, num_patients + 1):
    patient_id = f"P{pid:04d}"
    prompt = generate_patient_prompt(patient_id, start_date_str)
    raw_output = model_prompt(prompt)

    with open(f"{raw_output_dir}/data_output_{patient_id}.txt", "w") as f:
        f.write(raw_output)
```

```python
    print(f"Saved output for patient id: {patient_id}.")
    time.sleep(0.25)
```

```
Saved output for patient id: P0001.
Saved output for patient id: P0002.
Saved output for patient id: P0003.
Saved output for patient id: P0004.
Saved output for patient id: P0005.
Saved output for patient id: P0006.
Saved output for patient id: P0007.
Saved output for patient id: P0008.
Saved output for patient id: P0009.
Saved output for patient id: P0010.
Saved output for patient id: P0011.
Saved output for patient id: P0012.
Saved output for patient id: P0013.
Saved output for patient id: P0014.
Saved output for patient id: P0015.
Saved output for patient id: P0016.
Saved output for patient id: P0017.
Saved output for patient id: P0018.
Saved output for patient id: P0019.
Saved output for patient id: P0020.
Saved output for patient id: P0021.
Saved output for patient id: P0022.
Saved output for patient id: P0023.
Saved output for patient id: P0024.
Saved output for patient id: P0025.
Saved output for patient id: P0026.
Saved output for patient id: P0027.
Saved output for patient id: P0028.
Saved output for patient id: P0029.
Saved output for patient id: P0030.
Saved output for patient id: P0031.
Saved output for patient id: P0032.
Saved output for patient id: P0033.
Saved output for patient id: P0034.
Saved output for patient id: P0035.
Saved output for patient id: P0036.
Saved output for patient id: P0037.
Saved output for patient id: P0038.
Saved output for patient id: P0039.
Saved output for patient id: P0040.
Saved output for patient id: P0041.
Saved output for patient id: P0042.
Saved output for patient id: P0043.
Saved output for patient id: P0044.
```

```
Saved output for patient id: P0045.
Saved output for patient id: P0046.
Saved output for patient id: P0047.
Saved output for patient id: P0048.
Saved output for patient id: P0049.
Saved output for patient id: P0050.
Saved output for patient id: P0051.
Saved output for patient id: P0052.
Saved output for patient id: P0053.
Saved output for patient id: P0054.
Saved output for patient id: P0055.
Saved output for patient id: P0056.
Saved output for patient id: P0057.
Saved output for patient id: P0058.
Saved output for patient id: P0059.
Saved output for patient id: P0060.
Saved output for patient id: P0061.
Saved output for patient id: P0062.
Saved output for patient id: P0063.
Saved output for patient id: P0064.
Saved output for patient id: P0065.
Saved output for patient id: P0066.
Saved output for patient id: P0067.
Saved output for patient id: P0068.
Saved output for patient id: P0069.
Saved output for patient id: P0070.
Saved output for patient id: P0071.
Saved output for patient id: P0072.
Saved output for patient id: P0073.
Saved output for patient id: P0074.
Saved output for patient id: P0075.
Saved output for patient id: P0076.
Saved output for patient id: P0077.
Saved output for patient id: P0078.
Saved output for patient id: P0079.
Saved output for patient id: P0080.
Saved output for patient id: P0081.
Saved output for patient id: P0082.
Saved output for patient id: P0083.
Saved output for patient id: P0084.
Saved output for patient id: P0085.
Saved output for patient id: P0086.
Saved output for patient id: P0087.
Saved output for patient id: P0088.
Saved output for patient id: P0089.
Saved output for patient id: P0090.
Saved output for patient id: P0091.
Saved output for patient id: P0092.
```

```
Saved output for patient id: P0093.
Saved output for patient id: P0094.
Saved output for patient id: P0095.
Saved output for patient id: P0096.
Saved output for patient id: P0097.
Saved output for patient id: P0098.
Saved output for patient id: P0099.
Saved output for patient id: P0100.
Saved output for patient id: P0101.
Saved output for patient id: P0102.
Saved output for patient id: P0103.
Saved output for patient id: P0104.
Saved output for patient id: P0105.
Saved output for patient id: P0106.
Saved output for patient id: P0107.
Saved output for patient id: P0108.
Saved output for patient id: P0109.
Saved output for patient id: P0110.
Saved output for patient id: P0111.
Saved output for patient id: P0112.
Saved output for patient id: P0113.
Saved output for patient id: P0114.
Saved output for patient id: P0115.
Saved output for patient id: P0116.
Saved output for patient id: P0117.
Saved output for patient id: P0118.
Saved output for patient id: P0119.
Saved output for patient id: P0120.
Saved output for patient id: P0121.
Saved output for patient id: P0122.
Saved output for patient id: P0123.
Saved output for patient id: P0124.
Saved output for patient id: P0125.
Saved output for patient id: P0126.
Saved output for patient id: P0127.
Saved output for patient id: P0128.
Saved output for patient id: P0129.
Saved output for patient id: P0130.
Saved output for patient id: P0131.
Saved output for patient id: P0132.
Saved output for patient id: P0133.
Saved output for patient id: P0134.
Saved output for patient id: P0135.
Saved output for patient id: P0136.
Saved output for patient id: P0137.
Saved output for patient id: P0138.
Saved output for patient id: P0139.
Saved output for patient id: P0140.
```

```
Saved output for patient id: P0141.
Saved output for patient id: P0142.
Saved output for patient id: P0143.
Saved output for patient id: P0144.
Saved output for patient id: P0145.
Saved output for patient id: P0146.
Saved output for patient id: P0147.
Saved output for patient id: P0148.
Saved output for patient id: P0149.
Saved output for patient id: P0150.
Saved output for patient id: P0151.
Saved output for patient id: P0152.
Saved output for patient id: P0153.
Saved output for patient id: P0154.
Saved output for patient id: P0155.
Saved output for patient id: P0156.
Saved output for patient id: P0157.
Saved output for patient id: P0158.
Saved output for patient id: P0159.
Saved output for patient id: P0160.
Saved output for patient id: P0161.
Saved output for patient id: P0162.
Saved output for patient id: P0163.
Saved output for patient id: P0164.
Saved output for patient id: P0165.
Saved output for patient id: P0166.
Saved output for patient id: P0167.
Saved output for patient id: P0168.
Saved output for patient id: P0169.
Saved output for patient id: P0170.
Saved output for patient id: P0171.
Saved output for patient id: P0172.
Saved output for patient id: P0173.
Saved output for patient id: P0174.
Saved output for patient id: P0175.
Saved output for patient id: P0176.
Saved output for patient id: P0177.
Saved output for patient id: P0178.
Saved output for patient id: P0179.
Saved output for patient id: P0180.
Saved output for patient id: P0181.
Saved output for patient id: P0182.
Saved output for patient id: P0183.
Saved output for patient id: P0184.
Saved output for patient id: P0185.
Saved output for patient id: P0186.
Saved output for patient id: P0187.
Saved output for patient id: P0188.
```

```
Saved output for patient id: P0189.
Saved output for patient id: P0190.
Saved output for patient id: P0191.
Saved output for patient id: P0192.
Saved output for patient id: P0193.
Saved output for patient id: P0194.
Saved output for patient id: P0195.
Saved output for patient id: P0196.
Saved output for patient id: P0197.
Saved output for patient id: P0198.
Saved output for patient id: P0199.
Saved output for patient id: P0200.
Saved output for patient id: P0201.
Saved output for patient id: P0202.
Saved output for patient id: P0203.
Saved output for patient id: P0204.
Saved output for patient id: P0205.
Saved output for patient id: P0206.
Saved output for patient id: P0207.
Saved output for patient id: P0208.
Saved output for patient id: P0209.
Saved output for patient id: P0210.
Saved output for patient id: P0211.
Saved output for patient id: P0212.
Saved output for patient id: P0213.
Saved output for patient id: P0214.
Saved output for patient id: P0215.
Saved output for patient id: P0216.
Saved output for patient id: P0217.
Saved output for patient id: P0218.
Saved output for patient id: P0219.
Saved output for patient id: P0220.
Saved output for patient id: P0221.
Saved output for patient id: P0222.
Saved output for patient id: P0223.
Saved output for patient id: P0224.
Saved output for patient id: P0225.
Saved output for patient id: P0226.
Saved output for patient id: P0227.
Saved output for patient id: P0228.
Saved output for patient id: P0229.
Saved output for patient id: P0230.
Saved output for patient id: P0231.
Saved output for patient id: P0232.
Saved output for patient id: P0233.
Saved output for patient id: P0234.
Saved output for patient id: P0235.
Saved output for patient id: P0236.
```

```
Saved output for patient id: P0237.
Saved output for patient id: P0238.
Saved output for patient id: P0239.
Saved output for patient id: P0240.
Saved output for patient id: P0241.
Saved output for patient id: P0242.
Saved output for patient id: P0243.
Saved output for patient id: P0244.
Saved output for patient id: P0245.
Saved output for patient id: P0246.
Saved output for patient id: P0247.
Saved output for patient id: P0248.
Saved output for patient id: P0249.
Saved output for patient id: P0250.
Saved output for patient id: P0251.
Saved output for patient id: P0252.
Saved output for patient id: P0253.
Saved output for patient id: P0254.
Saved output for patient id: P0255.
Saved output for patient id: P0256.
Saved output for patient id: P0257.
Saved output for patient id: P0258.
Saved output for patient id: P0259.
Saved output for patient id: P0260.
Saved output for patient id: P0261.
Saved output for patient id: P0262.
Saved output for patient id: P0263.
Saved output for patient id: P0264.
Saved output for patient id: P0265.
Saved output for patient id: P0266.
Saved output for patient id: P0267.
Saved output for patient id: P0268.
Saved output for patient id: P0269.
Saved output for patient id: P0270.
Saved output for patient id: P0271.
Saved output for patient id: P0272.
Saved output for patient id: P0273.
Saved output for patient id: P0274.
Saved output for patient id: P0275.
Saved output for patient id: P0276.
Saved output for patient id: P0277.
Saved output for patient id: P0278.
Saved output for patient id: P0279.
Saved output for patient id: P0280.
Saved output for patient id: P0281.
Saved output for patient id: P0282.
Saved output for patient id: P0283.
Saved output for patient id: P0284.
```

```
Saved output for patient id: P0285.
Saved output for patient id: P0286.
Saved output for patient id: P0287.
Saved output for patient id: P0288.
Saved output for patient id: P0289.
Saved output for patient id: P0290.
Saved output for patient id: P0291.
Saved output for patient id: P0292.
Saved output for patient id: P0293.
Saved output for patient id: P0294.
Saved output for patient id: P0295.
Saved output for patient id: P0296.
Saved output for patient id: P0297.
Saved output for patient id: P0298.
Saved output for patient id: P0299.
Saved output for patient id: P0300.
Saved output for patient id: P0301.
Saved output for patient id: P0302.
Saved output for patient id: P0303.
Saved output for patient id: P0304.
Saved output for patient id: P0305.
Saved output for patient id: P0306.
Saved output for patient id: P0307.
Saved output for patient id: P0308.
Saved output for patient id: P0309.
Saved output for patient id: P0310.
Saved output for patient id: P0311.
Saved output for patient id: P0312.
Saved output for patient id: P0313.
Saved output for patient id: P0314.
Saved output for patient id: P0315.
Saved output for patient id: P0316.
Saved output for patient id: P0317.
Saved output for patient id: P0318.
Saved output for patient id: P0319.
Saved output for patient id: P0320.
Saved output for patient id: P0321.
Saved output for patient id: P0322.
Saved output for patient id: P0323.
Saved output for patient id: P0324.
Saved output for patient id: P0325.
Saved output for patient id: P0326.
Saved output for patient id: P0327.
Saved output for patient id: P0328.
Saved output for patient id: P0329.
Saved output for patient id: P0330.
Saved output for patient id: P0331.
Saved output for patient id: P0332.
```

```
Saved output for patient id: P0333.
Saved output for patient id: P0334.
Saved output for patient id: P0335.
Saved output for patient id: P0336.
Saved output for patient id: P0337.
Saved output for patient id: P0338.
Saved output for patient id: P0339.
Saved output for patient id: P0340.
Saved output for patient id: P0341.
Saved output for patient id: P0342.
Saved output for patient id: P0343.
Saved output for patient id: P0344.
Saved output for patient id: P0345.
Saved output for patient id: P0346.
Saved output for patient id: P0347.
Saved output for patient id: P0348.
Saved output for patient id: P0349.
Saved output for patient id: P0350.
Saved output for patient id: P0351.
Saved output for patient id: P0352.
Saved output for patient id: P0353.
Saved output for patient id: P0354.
Saved output for patient id: P0355.
Saved output for patient id: P0356.
Saved output for patient id: P0357.
Saved output for patient id: P0358.
Saved output for patient id: P0359.
Saved output for patient id: P0360.
Saved output for patient id: P0361.
Saved output for patient id: P0362.
Saved output for patient id: P0363.
Saved output for patient id: P0364.
Saved output for patient id: P0365.
Saved output for patient id: P0366.
Saved output for patient id: P0367.
Saved output for patient id: P0368.
Saved output for patient id: P0369.
Saved output for patient id: P0370.
Saved output for patient id: P0371.
Saved output for patient id: P0372.
Saved output for patient id: P0373.
Saved output for patient id: P0374.
Saved output for patient id: P0375.
Saved output for patient id: P0376.
Saved output for patient id: P0377.
Saved output for patient id: P0378.
Saved output for patient id: P0379.
Saved output for patient id: P0380.
```

```
Saved output for patient id: P0381.
Saved output for patient id: P0382.
Saved output for patient id: P0383.
Saved output for patient id: P0384.
Saved output for patient id: P0385.
Saved output for patient id: P0386.
Saved output for patient id: P0387.
Saved output for patient id: P0388.
Saved output for patient id: P0389.
Saved output for patient id: P0390.
Saved output for patient id: P0391.
Saved output for patient id: P0392.
Saved output for patient id: P0393.
Saved output for patient id: P0394.
Saved output for patient id: P0395.
Saved output for patient id: P0396.
Saved output for patient id: P0397.
Saved output for patient id: P0398.
Saved output for patient id: P0399.
Saved output for patient id: P0400.
Saved output for patient id: P0401.
Saved output for patient id: P0402.
Saved output for patient id: P0403.
Saved output for patient id: P0404.
Saved output for patient id: P0405.
Saved output for patient id: P0406.
Saved output for patient id: P0407.
Saved output for patient id: P0408.
Saved output for patient id: P0409.
Saved output for patient id: P0410.
Saved output for patient id: P0411.
Saved output for patient id: P0412.
Saved output for patient id: P0413.
Saved output for patient id: P0414.
Saved output for patient id: P0415.
Saved output for patient id: P0416.
Saved output for patient id: P0417.
Saved output for patient id: P0418.
Saved output for patient id: P0419.
Saved output for patient id: P0420.
Saved output for patient id: P0421.
Saved output for patient id: P0422.
Saved output for patient id: P0423.
Saved output for patient id: P0424.
Saved output for patient id: P0425.
Saved output for patient id: P0426.
Saved output for patient id: P0427.
Saved output for patient id: P0428.
```

```
Saved output for patient id: P0429.
Saved output for patient id: P0430.
Saved output for patient id: P0431.
Saved output for patient id: P0432.
Saved output for patient id: P0433.
Saved output for patient id: P0434.
Saved output for patient id: P0435.
Saved output for patient id: P0436.
Saved output for patient id: P0437.
Saved output for patient id: P0438.
Saved output for patient id: P0439.
Saved output for patient id: P0440.
Saved output for patient id: P0441.
Saved output for patient id: P0442.
Saved output for patient id: P0443.
Saved output for patient id: P0444.
Saved output for patient id: P0445.
Saved output for patient id: P0446.
Saved output for patient id: P0447.
Saved output for patient id: P0448.
Saved output for patient id: P0449.
Saved output for patient id: P0450.
Saved output for patient id: P0451.
Saved output for patient id: P0452.
Saved output for patient id: P0453.
Saved output for patient id: P0454.
Saved output for patient id: P0455.
Saved output for patient id: P0456.
Saved output for patient id: P0457.
Saved output for patient id: P0458.
Saved output for patient id: P0459.
Saved output for patient id: P0460.
Saved output for patient id: P0461.
Saved output for patient id: P0462.
Saved output for patient id: P0463.
Saved output for patient id: P0464.
Saved output for patient id: P0465.
Saved output for patient id: P0466.
Saved output for patient id: P0467.
Saved output for patient id: P0468.
Saved output for patient id: P0469.
Saved output for patient id: P0470.
Saved output for patient id: P0471.
Saved output for patient id: P0472.
Saved output for patient id: P0473.
Saved output for patient id: P0474.
Saved output for patient id: P0475.
Saved output for patient id: P0476.
```

```
Saved output for patient id: P0477.
Saved output for patient id: P0478.
Saved output for patient id: P0479.
Saved output for patient id: P0480.
Saved output for patient id: P0481.
Saved output for patient id: P0482.
Saved output for patient id: P0483.
Saved output for patient id: P0484.
Saved output for patient id: P0485.
Saved output for patient id: P0486.
Saved output for patient id: P0487.
Saved output for patient id: P0488.
Saved output for patient id: P0489.
Saved output for patient id: P0490.
Saved output for patient id: P0491.
Saved output for patient id: P0492.
Saved output for patient id: P0493.
Saved output for patient id: P0494.
Saved output for patient id: P0495.
Saved output for patient id: P0496.
Saved output for patient id: P0497.
Saved output for patient id: P0498.
Saved output for patient id: P0499.
Saved output for patient id: P0500.
```

Special Handling: Regenerating Patient P0264 Monitoring Records

During the JSON-to-DataFrame parsing phase, it was discovered that some patients' monitoring data had formatting issues that prevented successful conversion. This ensures the dataset remains complete and all patients are properly processed for subsequent analysis.

[ ]:
```python
# Define Patient ID
pid = 264
patient_id = f"P{pid:04d}"

# Generate Prompt
prompt = generate_patient_prompt(patient_id, start_date_str)

# Call GenAI model
raw_output = model_prompt(prompt)

# Save output
with open(f"{raw_output_dir}/data_output_{patient_id}.txt", "w") as f:
    f.write(raw_output)

print(f"Saved new output for patient id: {patient_id}.")
```

```
Saved new output for patient id: P0264.
```

Step 4: Parse and Build the Final Dataset

Loop over all patient IDs and:

1. Validate and parse each .txt file using json.loads
2. Extract daily records into a pandas.DataFrame

This ensures that malformed or empty outputs are skipped, and clean, structured data is retained.

[ ]:
```python
# === Utility Functions ===
def parse_json_from_prompt(raw_output):
    clean_text = re.sub(r"```json|```", "", raw_output).strip()
    return json.loads(clean_text)

def is_valid_json(text):
    try:
        json.loads(text)
        return True
    except json.JSONDecodeError as e:
        print("JSON error:", e)
        return False
# === Phase 2: Parse Raw Output into DataFrame ===
records = []

for file_path in sorted(glob.glob(f"{raw_output_dir}/data_output_P*.txt")):
    patient_id = os.path.basename(file_path).split("_")[-1].split(".")[0]

    try:
        with open(file_path, "r") as f:
            raw_output = f.read()

        # Parse JSON
        daily_data = json.loads(raw_output)
        monitoring_days = len(daily_data)

        for i, day in enumerate(daily_data):
            records.append({
                "patient_id": patient_id,
                "timestamp": day.get("date", np.nan),
                "oxygen_saturation": day.get("oxygen_saturation", np.nan),
                "heart_rate": day.get("heart_rate", np.nan),
                "temperature": day.get("temperature", np.nan),
                "blood_pressure": day.get("blood_pressure", np.nan),
                "weight": day.get("weight", np.nan),
                "blood_glucose": day.get("blood_glucose", np.nan),
                "clinical_note": day.get("clinical_note", ""),
            })

    except json.JSONDecodeError as err:
```

```python
        print(f"JSON Decode Error for {patient_id}: {err}")
    except Exception as err:
        print(f"Unexpected error for {patient_id}: {err}")
```

Step 5: Export Final CSV

Save the final compiled dataframe into generate_patient_dataset.csv for use in later sections:

1. Exploratory Data Analysis
2. SLM-based preprocessing

The exported dataset contains patient_id, timestamp, all vital signs, and clinical notes, ready for AI-driven analysis.

[ ]:
```python
# Save parsed dataset
df = pd.DataFrame(records)
df.to_csv(my_file_path+"generate_patient_dataset.csv", index=False)
df.head()
```

[ ]:
```
  patient_id   timestamp  oxygen_saturation  heart_rate  temperature  \
0      P0001  2025-01-01                 96          82         37.2
1      P0001  2025-01-02                 95          85         37.4
2      P0001  2025-01-03                 94          88         37.8
3      P0001  2025-01-04                 95          84         37.6
4      P0001  2025-01-05                 96          81         37.3

  blood_pressure  weight  blood_glucose  \
0         128/82    70.5          112.0
1         130/84    70.3          115.0
2         132/86    70.2            NaN
3         130/85    70.1          120.0
4         126/82    70.1          108.0

                                       clinical_note
0  Patient stable post-surgery. Vitals within nor…
1  Mildly elevated heart rate and temperature. Mo…
2  Temperature trending upward. Possible low-grad…
3  Temperature stabilizing. Patient reports impro…
4  Patient showing signs of steady recovery. Vita…
```

3 2. Exploratory Data Analysis (EDA) Enhanced by LLMs

This section performs structured analysis on the GenAI-simulated patient dataset through both statistical exploration and large language model (LLM) interpretation. The aim is to uncover clinical trends, detect missing patterns, and summarize patient recovery trajectories across variable monitoring durations.

Step 1: Load and Preview the Dataset

Load the final structured dataset (generate_patient_dataset.csv) , convert the timestamp to datetime format, and perform a quick inspection of the dataset shape and features.

[6]:
```python
import pandas as pd

# Load the dataset
df = pd.read_csv(my_file_path + "generate_patient_dataset.csv")
df["timestamp"] = pd.to_datetime(df["timestamp"])

# Preview structure
df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6501 entries, 0 to 6500
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   patient_id         6501 non-null   object
 1   timestamp          6501 non-null   datetime64[ns]
 2   oxygen_saturation  6501 non-null   int64
 3   heart_rate         6501 non-null   int64
 4   temperature        5832 non-null   float64
 5   blood_pressure     6494 non-null   object
 6   weight             6110 non-null   float64
 7   blood_glucose      5653 non-null   float64
 8   clinical_note      6501 non-null   object
dtypes: datetime64[ns](1), float64(3), int64(2), object(3)
memory usage: 457.2+ KB
```

[6]:
```
  patient_id  timestamp  oxygen_saturation  heart_rate  temperature  \
0      P0001 2025-01-01                 96          82         37.2
1      P0001 2025-01-02                 95          85         37.4
2      P0001 2025-01-03                 94          88         37.8
3      P0001 2025-01-04                 95          84         37.6
4      P0001 2025-01-05                 96          81         37.3

  blood_pressure  weight  blood_glucose  \
0         128/82    70.5          112.0
1         130/84    70.3          115.0
2         132/86    70.2            NaN
3         130/85    70.1          120.0
4         126/82    70.1          108.0

                                       clinical_note
0  Patient stable post-surgery. Vitals within nor…
1  Mildly elevated heart rate and temperature. Mo…
2  Temperature trending upward. Possible low-grad…
```

```
3  Temperature stabilizing. Patient reports impro…
4  Patient showing signs of steady recovery. Vita…
```

Step 2: Automated Full EDA with ydata-profiling

Using ydata-profiling to a comprehensive and professional-grade EDA report covering data types, statistics, missing values, correlations, interactions, distributions, and more.

[7]:
```python
from ydata_profiling import ProfileReport
from IPython.core.display import display, HTML

# Generate profiling report
profile = ProfileReport(df, title="GenAI Patient Dataset Exploratory Data␣
 ↪Analysis (EDA)", explorative=True)
profile.to_file(my_file_path+"eda_patient_report.html")

# Display in notebook
with open(my_file_path+"eda_patient_report.html", "r") as f:
    display(HTML(f.read()))
```

```
Output hidden; open in https://colab.research.google.com to view.
```

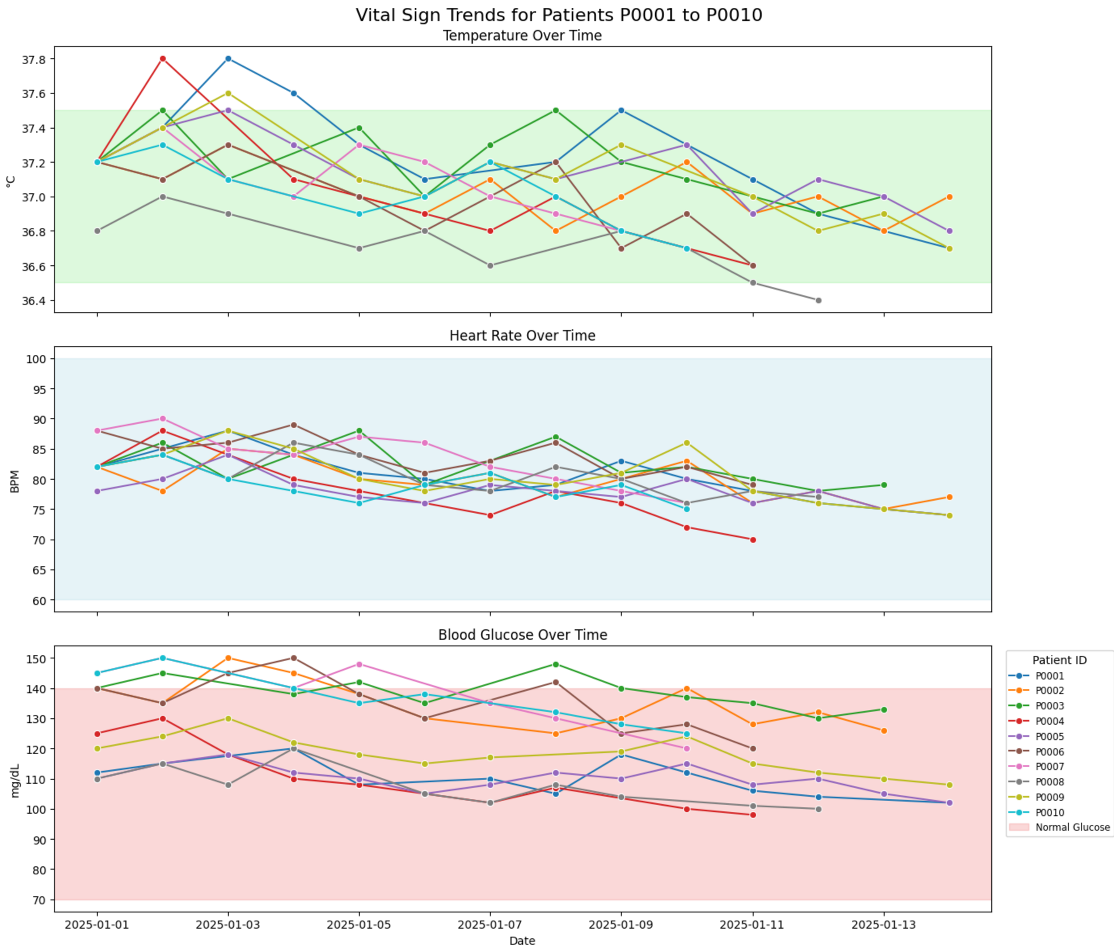
Step 3: Time Series Trend Visualization with Patient Range Control

This step visualizes temporal trends in core health indicators — temperature, heart rate, and blood glucose — for a custom-selected range of patients. Given the large dataset (500 patients), plotting all records together would be visually overwhelming. Therefore, I segment the visualizations by patient index range for clarity.

Clinical Reference Bands: 1. Temperature(°C): 36.5 ~ 37.5 2. Heart Rate (bpm): 60 – 100 3. Blood Glucose: 70 – 140 mg/dL

[8]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# === Define patient range (P0100 to P0120 means index 99 to 120) ===
start_index = 1
end_index = 10

# Get the exact patient IDs (e.g., 'P0100' to 'P0120')
patient_ids = df['patient_id'].unique()
selected_ids = patient_ids[start_index - 1 : end_index]
subset_df = df[df["patient_id"].isin(selected_ids)]

# === Plot time series with clinical reference zones ===
fig, axes = plt.subplots(3, 1, figsize=(14, 12), sharex=True)

# Temperature
```

```
sns.lineplot(ax=axes[0], data=subset_df, x="timestamp", y="temperature", hue="patient_id", marker="o")
axes[0].axhspan(36.5, 37.5, color='lightgreen', alpha=0.3, label='Normal Temp')
axes[0].set_title("Temperature Over Time")
axes[0].set_ylabel("°C")
axes[0].legend().remove()

# Heart Rate
sns.lineplot(ax=axes[1], data=subset_df, x="timestamp", y="heart_rate", hue="patient_id", marker="o")
axes[1].axhspan(60, 100, color='lightblue', alpha=0.3, label='Normal HR')
axes[1].set_title("Heart Rate Over Time")
axes[1].set_ylabel("BPM")
axes[1].legend().remove()

# Blood Glucose
sns.lineplot(ax=axes[2], data=subset_df, x="timestamp", y="blood_glucose", hue="patient_id", marker="o")
axes[2].axhspan(70, 140, color='lightcoral', alpha=0.3, label='Normal Glucose')
axes[2].set_title("Blood Glucose Over Time")
axes[2].set_ylabel("mg/dL")
axes[2].set_xlabel("Date")
axes[2].legend(bbox_to_anchor=(1.01, 1), loc="upper left", fontsize="small", title="Patient ID")

plt.suptitle(f"Vital Sign Trends for Patients P{start_index:04d} to P{end_index:04d}", fontsize=16)
plt.tight_layout()
plt.show()
```

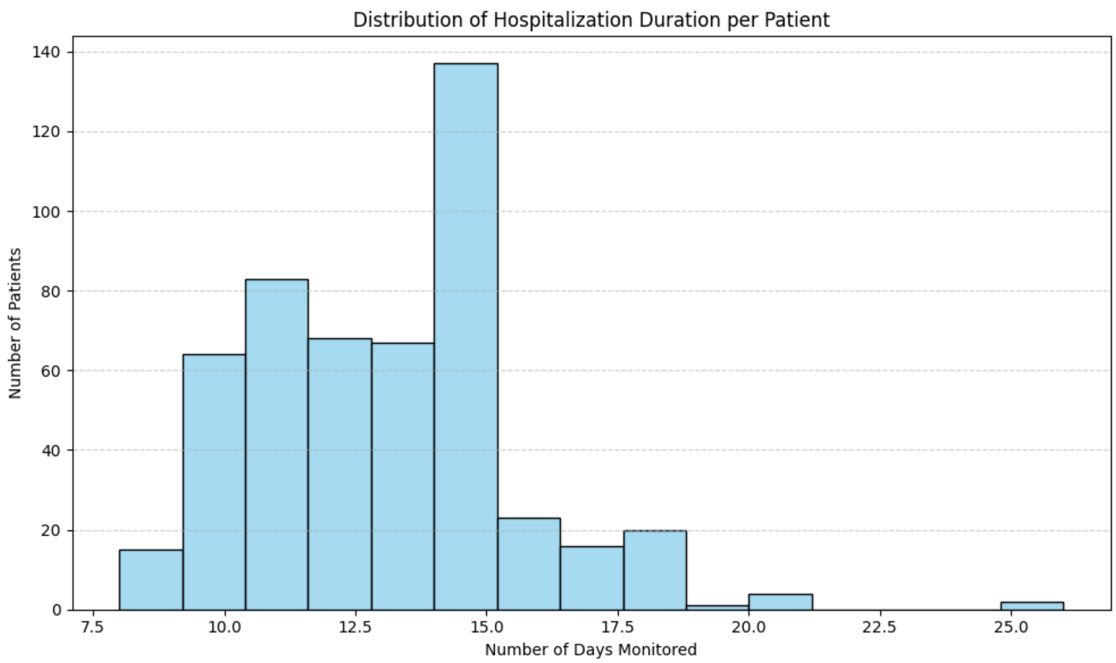


Step 4: Patient Hospitalization Duration Distribution

This step calculates the number of days each patient remained under daily monitoring by counting the number of records per patient ID. The result is visualized as a histogram to show the distribution of hospitalization durations across all patients. This helps identify how many patients were monitored for short, typical, or full durations (up to 30 days), reflecting real-world discharge or deterioration scenarios.

[9]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Count hospitalization duration per patient
duration_df = df.groupby("patient_id").size().reset_index(name="days_monitored")

# Plot histogram or barplot
plt.figure(figsize=(10, 6))
sns.histplot(duration_df["days_monitored"], bins=15, kde=False, color="skyblue")
plt.title("Distribution of Hospitalization Duration per Patient")
```

```
plt.xlabel("Number of Days Monitored")
plt.ylabel("Number of Patients")
plt.grid(axis="y", linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```



[9]:
```
# Display the top 10 patients with the longest monitoring durations
duration_df.sort_values("days_monitored", ascending=False).head(10)
```

[9]:

	patient_id	days_monitored
281	P0282	26
401	P0402	25
312	P0313	20
89	P0090	20
233	P0234	20
227	P0228	20
81	P0082	19
327	P0328	18
74	P0075	18
97	P0098	18

Step 5: Clinical Note Summarization using LLMs (e.g., GPT-4o)

This step leverages Large Language Models (LLMs) to generate concise, high-level summaries of each patient's clinical progression across the monitoring period:

1. Aggregate Clinical Notes Daily **clinical_note** entries are grouped by patient and concate-

nated into a single text block, providing a comprehensive view of the patient's condition evolution.

2. Design Structured Summarization Prompt A structured prompt is crafted to guide the LLM in summarizing key clinical trends, such as recovery, deterioration, or stabilization, while maintaining a professional medical tone.

3. LLM Invocation for Each Patient The structured prompt is sent to GPT-4o (or an equivalent LLM) on a per-patient basis, ensuring that each summary captures trend patterns, notable events, and the final clinical outcome.

4. Export Summarized Results The generated summaries are consolidated into a CSV file, enabling further analysis, visualization, or integration into downstream modeling tasks.

[ ]:
```python
# Group and prepare all_notes
patient_notes = df.groupby("patient_id")["clinical_note"].apply(lambda x: "\n".
 ↪join(x)).reset_index()
patient_notes.columns = ["patient_id", "all_notes"]

# Define system prompt
system_prompt = "You are a senior clinical analyst. Summarize patient trend␣
 ↪professionally."

# Generate prompts
def generate_summary_prompt(pid, notes):
    return f"""
Patient ID: {pid}
{notes}
Summarize in 2-3 sentences:
- Overall trend
- Turning points
- Final outcome
"""
summary_prompts = patient_notes.apply(lambda row:␣
 ↪generate_summary_prompt(row["patient_id"], row["all_notes"]), axis=1)
```

[ ]:
```python
from tqdm import tqdm

summaries = []

for idx, prompt in tqdm(summary_prompts.items(), total=len(summary_prompts),␣
 ↪desc="Generating Summaries"):
    pid = patient_notes.loc[idx, "patient_id"]
    output = model_prompt(prompt, system_prompt=system_prompt)
    summaries.append({
        "patient_id": pid,
        "summary": output
    })
```

```
Generating Summaries: 100%|     | 500/500 [45:34<00:00,  5.47s/it]
```

[ ]:
```python
# Save the summary dataset
summary_df = pd.DataFrame(summaries)
summary_df.to_csv(my_file_path+"patient_summary.csv", index=False)
summary_df.head()
```

[ ]:
```
  patient_id                                            summary
0      P0001  The patient demonstrated a steady recovery pos…
1      P0002  The patient demonstrated a stable and progress…
2      P0003  The patient demonstrated a steady recovery pos…
3      P0004  The patient demonstrated a steady and progress…
4      P0005  The patient demonstrated a stable recovery tra…
```

4 3. Advanced Data Preprocessing with SLMs / LLMs

This section demonstrates the use of advanced preprocessing techniques to prepare a patient monitoring dataset for analysis or modeling. The process includes intelligent missing value handling, feature normalization, categorical feature encoding, application of Small Language Models (SLMs) for text embedding or enhancement

Step 1: Categorical Feature Encoding

The composite blood_pressure string (e.g., "120/80") is split into two separate numerical columns: systolic_bp and diastolic_bp. This improves model interpretability and allows for numerical analysis or binning of these values.

[10]:
```python
df_cleaned = df.copy()
df_cleaned[["systolic_bp", "diastolic_bp"]] = df["blood_pressure"].str.split("/
↪", expand=True).astype(float)
df_cleaned = df_cleaned.drop(columns=["blood_pressure"])
```

[11]:
```python
# Preview
df_cleaned.info()
df_cleaned.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6501 entries, 0 to 6500
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   patient_id         6501 non-null   object
 1   timestamp          6501 non-null   datetime64[ns]
 2   oxygen_saturation  6501 non-null   int64
 3   heart_rate         6501 non-null   int64
 4   temperature        5832 non-null   float64
 5   weight             6110 non-null   float64
 6   blood_glucose      5653 non-null   float64
```

```
 7   clinical_note      6501 non-null   object
 8   systolic_bp        6494 non-null   float64
 9   diastolic_bp       6494 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(2), object(2)
memory usage: 508.0+ KB
```

[11]:

```
  patient_id  timestamp  oxygen_saturation  heart_rate  temperature  weight  \
0      P0001 2025-01-01                 96          82         37.2    70.5
1      P0001 2025-01-02                 95          85         37.4    70.3
2      P0001 2025-01-03                 94          88         37.8    70.2
3      P0001 2025-01-04                 95          84         37.6    70.1
4      P0001 2025-01-05                 96          81         37.3    70.1

   blood_glucose                                      clinical_note  \
0          112.0  Patient stable post-surgery. Vitals within nor…
1          115.0  Mildly elevated heart rate and temperature. Mo…
2            NaN  Temperature trending upward. Possible low-grad…
3          120.0  Temperature stabilizing. Patient reports impro…
4          108.0  Patient showing signs of steady recovery. Vita…

   systolic_bp  diastolic_bp
0        128.0          82.0
1        130.0          84.0
2        132.0          86.0
3        130.0          85.0
4        126.0          82.0
```

Step 2: Intelligent Per-Patient Missing Value Handling

To preserve individual variation in clinical patterns, missing values in key physiological metrics — including temperature, weight, blood glucose, and blood pressure (systolic and diastolic) — are imputed separately for each patient using their own median values. This patient-centric approach ensures that imputation aligns with personal baselines and avoids distortion from population-level statistics, maintaining the integrity of temporal health trends for downstream analysis.

[12]:

```python
fill_cols = ["temperature", "weight", "blood_glucose", "systolic_bp",␣
 ↪"diastolic_bp"]
for col in fill_cols:
    df_cleaned[col] = df_cleaned.groupby("patient_id")[col].transform(lambda x:␣
 ↪x.fillna(x.median()))
```

[13]:

```python
# Preview
df_cleaned.info()
df_cleaned.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6501 entries, 0 to 6500
Data columns (total 10 columns):
```

```
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   patient_id          6501 non-null   object
 1   timestamp           6501 non-null   datetime64[ns]
 2   oxygen_saturation   6501 non-null   int64
 3   heart_rate          6501 non-null   int64
 4   temperature         6501 non-null   float64
 5   weight              6501 non-null   float64
 6   blood_glucose       6501 non-null   float64
 7   clinical_note       6501 non-null   object
 8   systolic_bp         6501 non-null   float64
 9   diastolic_bp        6501 non-null   float64
dtypes: datetime64[ns](1), float64(5), int64(2), object(2)
memory usage: 508.0+ KB
```

[13]:
```
  patient_id  timestamp  oxygen_saturation  heart_rate  temperature  weight  \
0      P0001 2025-01-01                 96          82         37.2    70.5
1      P0001 2025-01-02                 95          85         37.4    70.3
2      P0001 2025-01-03                 94          88         37.8    70.2
3      P0001 2025-01-04                 95          84         37.6    70.1
4      P0001 2025-01-05                 96          81         37.3    70.1

   blood_glucose                                       clinical_note  \
0          112.0  Patient stable post-surgery. Vitals within nor…
1          115.0  Mildly elevated heart rate and temperature. Mo…
2          110.0  Temperature trending upward. Possible low-grad…
3          120.0  Temperature stabilizing. Patient reports impro…
4          108.0  Patient showing signs of steady recovery. Vita…

   systolic_bp  diastolic_bp
0        128.0          82.0
1        130.0          84.0
2        132.0          86.0
3        130.0          85.0
4        126.0          82.0
```

Step 3: Numerical Feature Normalization

To ensure fair comparisons between patients and to prevent scale dominance during model training, we applied Z-score normalization to key numerical features. This transformation centers values around zero and scales them based on standard deviation, enabling balanced feature contributions across machine learning models such as Neural Networks and gradient-boosted trees. The normalized features include:

- temperature
- heart_rate
- blood_glucose
- oxygen_saturation
- systolic_bp

- diastolic_bp
- weight

Each original column is transformed into a new column with _zscore suffix (e.g., temperature_zscore), while the original columns will be dropped to reduce redundancy and improve clarity in downstream modeling.

[14]:
```python
from sklearn.preprocessing import StandardScaler

norm_cols = ["temperature", "heart_rate", "blood_glucose", "oxygen_saturation", 
 ↪"systolic_bp", "diastolic_bp", "weight"]
scaler = StandardScaler()
df_cleaned[[f"{col}_zscore" for col in norm_cols]] = scaler.
 ↪fit_transform(df_cleaned[norm_cols])
df_cleaned.drop(columns=norm_cols, inplace=True)
```

[15]:
```python
# Preview
df_cleaned.info()
df_cleaned.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6501 entries, 0 to 6500
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   patient_id                6501 non-null   object
 1   timestamp                 6501 non-null   datetime64[ns]
 2   clinical_note             6501 non-null   object
 3   temperature_zscore        6501 non-null   float64
 4   heart_rate_zscore         6501 non-null   float64
 5   blood_glucose_zscore      6501 non-null   float64
 6   oxygen_saturation_zscore  6501 non-null   float64
 7   systolic_bp_zscore        6501 non-null   float64
 8   diastolic_bp_zscore       6501 non-null   float64
 9   weight_zscore             6501 non-null   float64
dtypes: datetime64[ns](1), float64(7), object(2)
memory usage: 508.0+ KB
```

[15]:
```
  patient_id  timestamp                                      clinical_note  \
0      P0001 2025-01-01  Patient stable post-surgery. Vitals within nor…
1      P0001 2025-01-02  Mildly elevated heart rate and temperature. Mo…
2      P0001 2025-01-03  Temperature trending upward. Possible low-grad…
3      P0001 2025-01-04  Temperature stabilizing. Patient reports impro…
4      P0001 2025-01-05  Patient showing signs of steady recovery. Vita…

   temperature_zscore  heart_rate_zscore  blood_glucose_zscore  \
0            0.360442           0.256585             -0.617148
1            1.089682           0.881403             -0.438365
```

```
2           2.548162          1.506221           -0.736337
3           1.818922          0.673130           -0.140392
4           0.725062          0.048312           -0.855527

   oxygen_saturation_zscore  systolic_bp_zscore  diastolic_bp_zscore  \
0                  0.029323            1.034085             0.690185
1                 -0.785334            1.423315             1.208257
2                 -1.599992            1.812545             1.726329
3                 -0.785334            1.423315             1.467293
4                  0.029323            0.644855             0.690185

   weight_zscore
0      -0.633036
1      -0.734106
2      -0.784640
3      -0.835175
4      -0.835175
```

Step 4: LLM-Based Clinical Note Classification using Hugging Face

To automate the labeling of patient clinical status, we applied a zero-shot classification pipeline using the `facebook/bart-large-mnli` model via Hugging Face Transformers. Each clinical note is classified into one of the predefined categories: **Stable**, **Recovering**, **Deteriorating**, or **Critical**.

This approach leverages a language model's semantic understanding to map raw free-text medical notes into structured health status labels, without requiring manual annotation or rule-based logic.

The output (`note_status`) is then label-encoded into `note_status_encoded` for model training.

Label Definitions: * Stable – Patient remained within normal clinical limits. * Recovering – Signs of gradual improvement observed. * Deteriorating – Abnormality or worsening symptoms detected. * Critical – Urgent care required due to severe condition.

This method supports downstream supervised model training, enabling the development of predictive models based on both structured data and LLM-generated labels.

[16]:
```python
import torch
# Device Detection (GPU if available)
device = 0 if torch.cuda.is_available() else -1
print("Using device:", "GPU" if device == 0 else "CPU")
```

```
Using device: GPU
```

[17]:
```python
from transformers import pipeline

# Load zero-shot classifier
classifier = pipeline(
    "zero-shot-classification",
    model="facebook/bart-large-mnli",
    device=device
```

```
)
# Define medical status labels
labels = ["Stable", "Recovering", "Deteriorating", "Critical"]

# Initialize empty column for note_status
df_cleaned["note_status"] = None
```

```
Device set to use cuda:0
```

[18]:
```
from tqdm import tqdm
# Start classification
print("Starting clinical note classification with Hugging Face zero-shot model..
 ↪.\n")

for idx, text in tqdm(enumerate(df_cleaned["clinical_note"]),␣
 ↪total=len(df_cleaned), desc="Classifying Notes"):
    result = classifier(text, candidate_labels=labels)
    df_cleaned.loc[idx, "note_status"] = result["labels"][0]  # Choose the most␣
 ↪likely label
```

```
Starting clinical note classification with Hugging Face zero-shot model…


Classifying Notes:   0%|          | 9/6501 [00:01<12:50,  8.43it/s]You seem to
be using the pipelines sequentially on GPU. In order to maximize efficiency
please use a dataset
Classifying Notes: 100%|     | 6501/6501 [08:54<00:00, 12.17it/s]
```

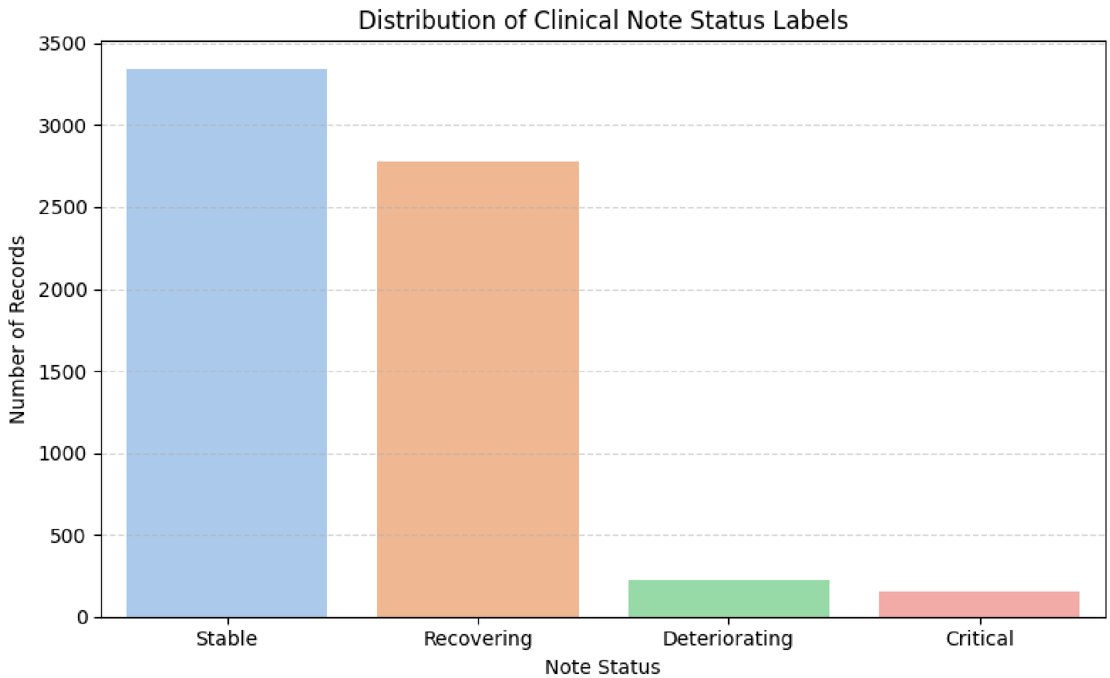
[19]:
```
# Preview sample
df_cleaned[["patient_id", "timestamp", "clinical_note", "note_status"]].head()
```

[19]:
```
  patient_id  timestamp                                      clinical_note  \
0      P0001 2025-01-01  Patient stable post-surgery. Vitals within nor…
1      P0001 2025-01-02  Mildly elevated heart rate and temperature. Mo…
2      P0001 2025-01-03  Temperature trending upward. Possible low-grad…
3      P0001 2025-01-04  Temperature stabilizing. Patient reports impro…
4      P0001 2025-01-05  Patient showing signs of steady recovery. Vita…

  note_status
0      Stable
1  Recovering
2  Recovering
3      Stable
4  Recovering
```

[24]:
```
# Distribution of classified note_status labels.
plt.figure(figsize=(8, 5))
sns.countplot(
```

```
    data=df_cleaned,
    x="note_status",
    palette="pastel",
    order=df_cleaned["note_status"].value_counts().index
)
plt.title("Distribution of Clinical Note Status Labels")
plt.xlabel("Note Status")
plt.ylabel("Number of Records")
plt.grid(axis="y", linestyle="--", alpha=0.5)
plt.tight_layout()
plt.show()

df_cleaned["note_status"].value_counts()
```

```
<ipython-input-24-72fad9dac576>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(
```



[24]:
```
note_status
Stable           3346
Recovering       2775
```

```
Deteriorating     224
Critical          156
Name: count, dtype: int64
```

[25]:
```python
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df_cleaned["note_status_encoded"] = label_encoder.
    ↪fit_transform(df_cleaned["note_status"])
label_mapping = dict(zip(label_encoder.classes_, label_encoder.
    ↪transform(label_encoder.classes_)))
label_mapping
```

[25]:
```
{'Critical': np.int64(0),
 'Deteriorating': np.int64(1),
 'Recovering': np.int64(2),
 'Stable': np.int64(3)}
```

[26]:
```python
# Preview
df_cleaned.info()
df_cleaned.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6501 entries, 0 to 6500
Data columns (total 12 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   patient_id               6501 non-null   object
 1   timestamp                6501 non-null   datetime64[ns]
 2   clinical_note            6501 non-null   object
 3   temperature_zscore       6501 non-null   float64
 4   heart_rate_zscore        6501 non-null   float64
 5   blood_glucose_zscore     6501 non-null   float64
 6   oxygen_saturation_zscore 6501 non-null   float64
 7   systolic_bp_zscore       6501 non-null   float64
 8   diastolic_bp_zscore      6501 non-null   float64
 9   weight_zscore            6501 non-null   float64
 10  note_status              6501 non-null   object
 11  note_status_encoded      6501 non-null   int64
dtypes: datetime64[ns](1), float64(7), int64(1), object(3)
memory usage: 609.6+ KB
```

[26]:
```
  patient_id  timestamp                                      clinical_note  \
0      P0001 2025-01-01  Patient stable post-surgery. Vitals within nor…
1      P0001 2025-01-02  Mildly elevated heart rate and temperature. Mo…
2      P0001 2025-01-03  Temperature trending upward. Possible low-grad…
3      P0001 2025-01-04  Temperature stabilizing. Patient reports impro…
```

```
4      P0001 2025-01-05  Patient showing signs of steady recovery. Vita…

   temperature_zscore  heart_rate_zscore  blood_glucose_zscore  \
0            0.360442           0.256585             -0.617148
1            1.089682           0.881403             -0.438365
2            2.548162           1.506221             -0.736337
3            1.818922           0.673130             -0.140392
4            0.725062           0.048312             -0.855527

   oxygen_saturation_zscore  systolic_bp_zscore  diastolic_bp_zscore  \
0                  0.029323            1.034085             0.690185
1                 -0.785334            1.423315             1.208257
2                 -1.599992            1.812545             1.726329
3                 -0.785334            1.423315             1.467293
4                  0.029323            0.644855             0.690185

   weight_zscore note_status  note_status_encoded
0      -0.633036      Stable                    3
1      -0.734106  Recovering                    2
2      -0.784640  Recovering                    2
3      -0.835175      Stable                    3
4      -0.835175  Recovering                    2
```

[27]:
```python
# Save classified results
df_cleaned.to_csv(my_file_path + "preprocessing_generate_patient_dataset.csv",
 index=False)
```