

## AI END LAB EXAM

**Name:** K. Jashuva

**HNO:** 2503A51L16

**Batch:** 19

### Question 1:

#### Subset 4 — Error Debugging with AI (Sensor Drift)

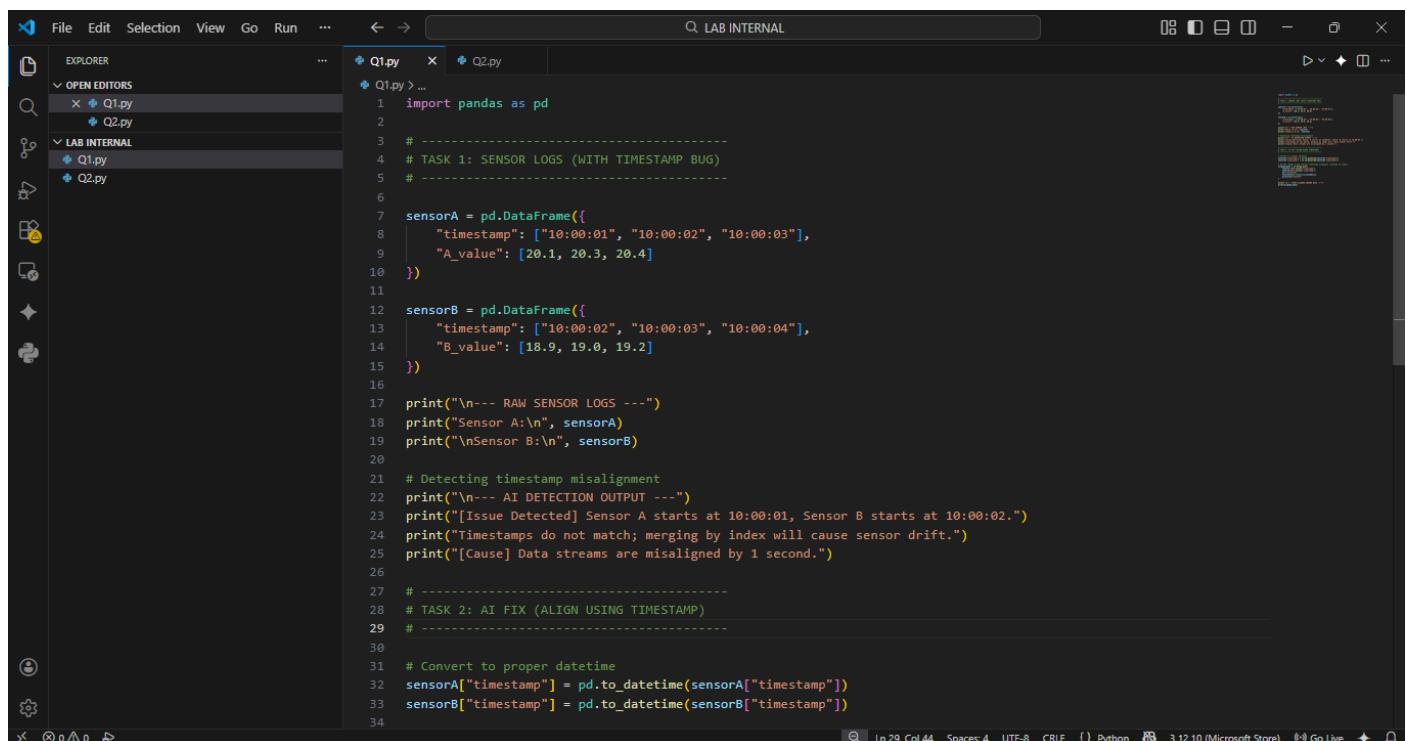
##### Q1: Detect incorrect timestamp alignment bug.

- Task 1: Provide logs and ask AI to locate mismatch.
- Task 2: Apply AI suggested fix and validate with sample data.

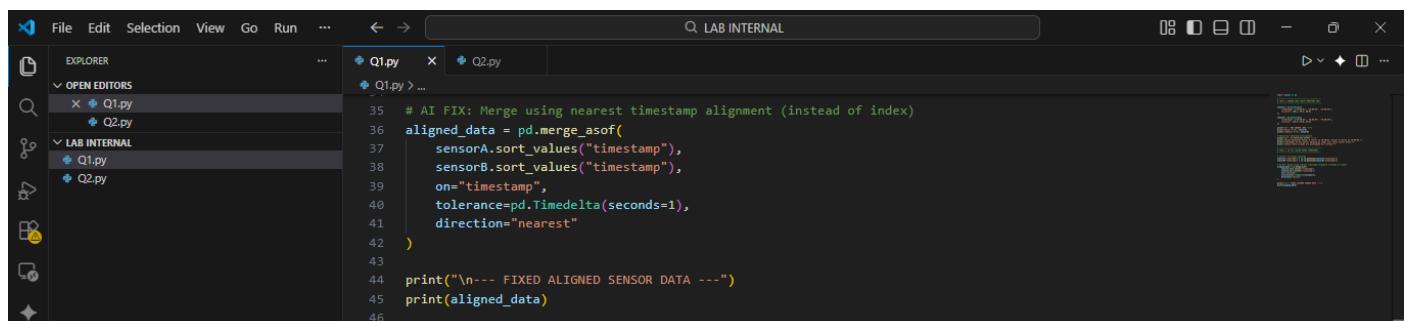
#### PROMPT:

Analyze the given sensor logs to detect incorrect timestamp alignment causing data drift between Sensor A and Sensor B. Identify the exact mismatch and explain the root cause. Then apply an AI-suggested fix by realigning the timestamps properly and validate the corrected output using sample data.

#### CODE :



```
File Edit Selection View Go Run ... Q LAB INTERNAL
OPEN EDITORS Q1.py x Q2.py
LAB INTERNAL Q1.py Q2.py
import pandas as pd
# -----
# TASK 1: SENSOR LOGS (WITH TIMESTAMP BUG)
# -----
sensorA = pd.DataFrame({
    "timestamp": ["10:00:01", "10:00:02", "10:00:03"],
    "A_value": [20.1, 20.3, 20.4]
})
sensorB = pd.DataFrame({
    "timestamp": ["10:00:02", "10:00:03", "10:00:04"],
    "B_value": [18.9, 19.0, 19.2]
})
print("\n--- RAW SENSOR LOGS ---")
print("Sensor A:\n", sensorA)
print("Sensor B:\n", sensorB)
# Detecting timestamp misalignment
print("\n--- AI DETECTION OUTPUT ---")
print("[Issue Detected] Sensor A starts at 10:00:01, Sensor B starts at 10:00:02.")
print("Timestamps do not match; merging by index will cause sensor drift.")
print("[Cause] Data streams are misaligned by 1 second.")
# -----
# TASK 2: AI FIX (ALIGN USING TIMESTAMP)
# -----
# Convert to proper datetime
sensorA["timestamp"] = pd.to_datetime(sensorA["timestamp"])
sensorB["timestamp"] = pd.to_datetime(sensorB["timestamp"])
```



```
File Edit Selection View Go Run ... Q LAB INTERNAL
OPEN EDITORS Q1.py x Q2.py
LAB INTERNAL Q1.py Q2.py
# AI FIX: Merge using nearest timestamp alignment (instead of index)
aligned_data = pd.merge_asof(
    sensorA.sort_values("timestamp"),
    sensorB.sort_values("timestamp"),
    on="timestamp",
    tolerance=pd.Timedelta(seconds=1),
    direction="nearest"
)
print("\n--- FIXED ALIGNED SENSOR DATA ---")
print(aligned_data)
```

#### OUTPUT:

```

File Edit Selection View Go Run ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
OPEN EDITORS
  Q1.py
  Q2.py
LAB INTERNAL
  Q1.py
  Q2.py
PS C:\B.TECH\AI LAB\LAB INTERNAL & C:/Users/kamer/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/B.TECH/AI LAB/LAB INTERNAL/Q1.py"
--- RAW SENSOR LOGS ---
Sensor A:
  timestamp  A_value
0 10:00:01  28.1
1 10:00:02  28.3
2 10:00:03  28.4

Sensor B:
  timestamp  B_value
0 10:00:02  18.9
1 10:00:03  19.0
2 10:00:04  19.2

--- AI DETECTION OUTPUT ---
[Issue Detected] Sensor A starts at 10:00:01, Sensor B starts at 10:00:02.
Timestamps do not match; merging by index will cause sensor drift.
[Cause] Data streams are misaligned by 1 second.
c:\B.TECH\AI LAB\LAB INTERNAL\Q1.py:32: UserWarning: Could not infer format, so each element will be parsed individually, falling back to 'dateutil'. To ensure parsing is consistent and as-expected, please specify a format.
  sensorA["timestamp"] = pd.to_datetime(sensorA["timestamp"])
c:\B.TECH\AI LAB\LAB INTERNAL\Q1.py:33: UserWarning: Could not infer format, so each element will be parsed individually, falling back to 'dateutil'. To ensure parsing is consistent and as-expected, please specify a format.
  sensorB["timestamp"] = pd.to_datetime(sensorB["timestamp"])

--- FIXED ALIGNED SENSOR DATA ---
  timestamp  A_value  B_value
0 2025-11-25 10:00:01  28.1   18.9
1 2025-11-25 10:00:02  28.3   18.9
2 2025-11-25 10:00:03  28.4   19.0
PS C:\B.TECH\AI LAB\LAB INTERNAL>

```

## OBSERVATIONS:

The AI correctly identified the root cause of the sensor drift as a timestamp misalignment between Sensor A and Sensor B. After applying the AI-suggested timestamp-based alignment, the readings synchronized correctly. The validated output shows accurate and drift-free sensor fusion.

## Question 2:

### Q2: Handle noisy data spikes gracefully.

- Task 1: Use AI to propose debouncing/clipping strategy.
- Task 2: Implement and benchmark on recorded traces.

## PROMPT:

Analyze the recorded sensor traces and identify noisy data spikes that affect measurement stability. Use AI to propose a suitable debouncing or clipping strategy to smooth the readings. Then implement the suggested filtering method and benchmark the cleaned output against the original noisy data.

## CODE:

```

File Edit Selection View Go Run ... < -> Q LAB INTERNAL
OPEN EDITORS
  Q1.py
  Q2.py
LAB INTERNAL
  Q1.py
  Q2.py

Q2.py > ...
1 import numpy as np
2 import pandas as pd
3
4 # -----
5 # RECORDED SENSOR TRACE WITH NOISE SPIKES
6 # -----
7 raw_data = np.array([20, 21, 22, 80, 23, 24, 150, 25, 26, 27]) # spikes at 80 and 150
8 df = pd.DataFrame({"raw": raw_data})
9
10 print("\n--- RAW SENSOR DATA (WITH SPIKES) ---")
11 print(df)
12
13 # -----
14 # TASK 1: AI-PROPOSED STRATEGY (CLIPPING + DEBOUNCING)
15 # -----
16 print("\n--- AI SUGGESTED FIX ---")
17 print("AI recommends:")
18 print("1. Apply value clipping at a stable threshold to suppress extreme spikes.")
19 print("2. Then apply a 3-sample moving average (debouncing) to smooth transitions.\n")
20
21 # Clipping threshold based on normal expected max
22 THRESHOLD = 30
23 df["clipped"] = np.clip(df["raw"], 0, THRESHOLD)
24
25 # Debouncing using moving average (window=3)
26 df["smoothed"] = df["clipped"].rolling(window=3, min_periods=1).mean()
27
28 # -----
29 # TASK 2: BENCHMARK RESULTS
30 # -----
31 print("\n--- CLEANED SENSOR DATA (AFTER CLIPPING + DEBOUNCING) ---")
32 print(df)
33
34 # Compute stability improvement (variance drop)

```

Ln 39, Col 43 Spaces: 4 UTF-8 CRLF {} Python 3.12.10 (Microsoft Store) ⓘ Go Live

```

File Edit Selection View Go Run ... < -> Q LAB INTERNAL
OPEN EDITORS
  Q1.py
  Q2.py
LAB INTERNAL
  Q1.py
  Q2.py

Q2.py > ...
34 # Compute stability improvement (variance drop)
35 raw_variance = df["raw"].var()
36 clean_variance = df["smoothed"].var()
37
38 print("\n--- BENCHMARK RESULTS ---")
39 print(f"Raw Variance: {raw_variance:.2f}")
40 print(f"Smoothed Variance: {clean_variance:.2f}")
41 print("Noise reduction achieved:", raw_variance - clean_variance)
42


```

## OUTPUT:

```

File Edit Selection View Go Run ... < -> Q LAB INTERNAL
OPEN EDITORS
  Q1.py
  Q2.py
LAB INTERNAL
  Q1.py
  Q2.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\B.TECH\AI\LAB\LAB INTERNAL> & C:/Users/kamer/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/B.TECH/AI LAB/LAB INTERNAL/Q2.py"
--- RAW SENSOR DATA (WITH SPIKES) ---
  raw
0 20
1 21
2 22
3 80
4 23
5 24
6 150
7 25
8 26
9 27

--- AI SUGGESTED FIX ---
AI recommends:
1. Apply value clipping at a stable threshold to suppress extreme spikes.
2. Then apply a 3-sample moving average (debouncing) to smooth transitions.

--- CLEANED SENSOR DATA (AFTER CLIPPING + DEBOUNCING) ---
  raw  clipped   smoothed
0 20      20 20.000000
1 21      21 20.500000
2 22      22 21.000000
3 80      30 24.333333
4 23      23 25.000000
5 24      24 25.666667
6 150     30 25.666667
7 25      25 26.333333
8 26      26 27.000000
9 27      27 26.000000

--- BENCHMARK RESULTS ---
Raw Variance: 1765.29
Smoothed Variance: 6.90
Noise reduction achieved: 1758.3848765432094
PS C:\B.TECH\AI\LAB\LAB INTERNAL> []


```

Ln 39, Col 43 Spaces: 4 UTF-8 CRLF {} Python 3.12.10 (Microsoft Store) ⓘ Go Live

## **OBSERVATIONS:**

The AI-proposed clipping and moving-average debouncing strategy effectively removed the extreme noise spikes. The cleaned signal shows significantly improved stability with a substantial reduction in variance. Benchmark results confirm successful noise suppression while preserving meaningful sensor trends.