PART 4 — Open-Ended Optimization (10% weight)

Goal: Increase throughput from **10,000 → 100,000 data points per minute** (10× improvement) without changing business logic.

---

1) Debugging Process

- **Find bottlenecks first**: measure before optimizing.
- **Tools**:
  - `cProfile` or `py-spy` → where CPU time is spent.
  - `memory_profiler` → track object usage.
  - System tools (`htop` , `iostat` , `ping`) → check CPU, disk, and network.
- **Metrics**: throughput (items/sec), latency per stage, CPU %, memory usage, disk/network usage.

---

2) Optimization Strategy

Common bottlenecks and fixes:

1. **Database / File I/O**
   - Problem: too many small reads/writes.
   - Fix: batch operations, use faster formats (Parquet/Arrow), enable async I/O.
2. **CPU (Python overhead)**
   - Problem: slow loops and JSON parsing.
   - Fix: use NumPy/pandas for vectorized ops, faster libraries (`orjson`), or compile hot loops (Numba/Cython).
3. **Memory**

    - Problem: too many small objects causing garbage collection (GC).
    - Fix: process data in batches, reuse buffers, use arrays instead of lists/dicts.

4. **Network**
    - Problem: too many API calls or waiting on responses.
    - Fix: connection pooling, caching results, async requests, reduce request frequency.

5. **Concurrency**
    - Problem: only one worker at a time.
    - Fix: run tasks in parallel (multiprocessing for CPU, asyncio for I/O), use queues to balance load.

---

3) Implementation Plan

- **Step 1: Measure baseline** → run profiling and record current throughput.
- **Step 2: Quick wins** → batching I/O, switch to faster JSON parser, enable connection pooling.
- **Step 3: Optimize compute** → vectorize calculations, reduce object churn.
- **Step 4: Add concurrency** → multiprocessing for CPU-heavy tasks, asyncio for I/O tasks.
- **Step 5: Test** → compare outputs to the old system (to ensure correctness).
- **Step 6: Deploy carefully** → canary rollout (small % of data first), then full deployment.

---

4) Documentation Strategy

- **Before/After Metrics** → record throughput, latency, CPU/memory.
- **Change Log** → list each optimization and why it was done.
- **Runbook** → steps to profile again, tune batch sizes, or rollback changes.

- **Post-mortem Guide** → what slowed the system originally, what fixed it, and what to watch out for next time.