Retail Operational Intelligence - Solution Architecture

1. Executive Summary

This document outlines the technical approach to build an automated, config-driven data pipeline for the 'Retail Operational Intelligence & Personalization Suite'. The system will ingest raw inventory reports and restock logs, validate data integrity, reconcile discrepancies using fuzzy logic, and produce a high-quality 'Golden Record' of inventory.

2. Key Goals & Focus Areas

- Ingest daily inventory_snapshot.csv and restock_events.csv into a RAW layer.
- Validate inventory quantities (Negative stock, Mismatched Product IDs, Duplicates, Logical Max).
- Recompute Effective Stock Level = Snapshot + Restock - Damaged/Expired.
- Load cleaned data into a Curated Inventory Fact Table.
- Route invalid records to a Quarantine Table.

3. System Architecture Strategy

We implementation a 'Medallion Architecture' (Bronze -> Silver -> Gold):

A. Config-Driven Ingestion (The 'Additional Score' Goal)

To support *any* new inventory file without code changes, we abstract file definitions into a YAML Configuration system. A 'schema_config.yaml' will define file patterns and required columns. A generic reader factory will parse files based on this config.

B. Validation & Quarantine Logic

We implement a validation pipeline. Records failing checks (e.g., negative stock) are tagged and moved to a Quarantine dataset, while valid records proceed to the Silver layer.

4. Inventory Reconciliation Engine (Fuzzy Matching)

For records quarantined due to 'Mismatched Product ID', we run a recovery process:

1. Identify records where product_id does not exist in Master Data.

2. Use Levenshtein Distance (Fuzzy Matching) to compare names against the Master Catalog.

3. If Match Score > 90%: Auto-map and move to Silver Layer.

4. If Match Score < 90%: Keep in Quarantine for manual review.

5. Technical Stack Recommendation

Language: Python 3.10+

Data Processing: Polars (for performance) or Pandas

Database: DuckDB (Serverless SQL OLAP)

Fuzzy Matching: rapid_fuzz or thefuzz

Orchestration: Python script (main.py) with modular design.