# Retail Data Processing Hackathon Report

Implementation of End-to-End Business Logic

Data Engineering Team

December 19, 2025

## 1 Problem Statement

In modern retail environments, raw transactional data often contains noise and requires complex processing to derive value. This project establishes a robust data pipeline to handle high-volume sales data, ensuring quality through automated validation and providing actionable insights across customer loyalty and performance metrics.

### 1.1 Objective

The primary objective is the successful implementation of the six critical business use cases defined for this hackathon:

- **UC1:** Automated Data Ingestion and Quality Validation.
- **UC2:** Real-Time Promotion Effectiveness Analysis.
- **UC3:** Loyalty Point Calculation Engine.
- **UC4:** Customer Segmentation for Targeted Offers.
- **UC5:** Automated Loyalty Notification System.
- **UC6:** Inventory and Store Performance Correlation.

## 2 System Methodology

The solution utilizes a tiered architecture to process data from raw generation to final visualization.

### 2.1 Data Quality Pipeline (UC1)

Raw data is processed through a validation layer. Records with negative total amounts or missing primary identifiers (Customer IDs) are segregated into a *Quarantine* table.

### 2.2 Loyalty and Segmentation Engine (UC3, UC4)

Loyalty points are accrued using a defined logic: \$1 = 1 point, with a 50-point bonus for purchases exceeding \$100. Customers are segmented using Recency metrics to identify **At-Risk** groups (no purchase in 60+ days).

# 3 Technical Implementation Summary

## 3.1 Automated Quality Validation (data_gen.py)

The pipeline identifies and rejects records that do not meet data quality rules.

```
# Logic added to segregate "bad" data
bad_data = raw_data[
    (raw_data['total_amount'] < 0) |
    (raw_data['customer_id'].isna())
]

clean_data = raw_data[~raw_data.index.isin(bad_data.index)]
clean_data.to_csv('sales_header_clean.csv', index=False)
bad_data.to_csv('quarantine_header.csv', index=False)
```

Listing 1: Data Quality Segregation Logic (UC1)

## 3.2 Loyalty Point Engine (analysis.py)

This logic accurately accrues loyalty points for every eligible customer transaction.

```
# Logic added for loyalty calculation
def calculate_points(row):
    base_points = int(row['total_amount']) # $1 = 1 point
    bonus = 50 if row['total_amount'] > 100 else 0
    return base_points + bonus

df_sales['points_earned'] = df_sales.apply(calculate_points, axis=1)
```

Listing 2: Loyalty Point Calculation (UC3)

## 3.3 Customer Segmentation and Inventory Analysis (analysis.py)

Segmentation logic identifies high-value and at-risk customers, while inventory correlation estimates potential lost sales.

```
# Segmentation logic: Identify At-Risk customers
at_risk = customer_stats[customer_stats['days_since_last_purchase'] >
    60]

# Inventory correlation logic: Estimated Revenue Leakage
lost_sales = (avg_daily_sales * days_out_of_stock) * unit_price
```

Listing 3: Segmentation and Inventory Logic (UC4

# 4 Conclusion

This solution demonstrates a successful end-to-end cycle from data ingestion to customer engagement. By automating complex calculations and quality checks, the system empowers data-driven decisions that enhance customer loyalty and operational efficiency.