# Document Info

**Author : Ankur-ipynb**

**Version: 1.0**

**Project Status: V1 completed**

**Last Updated: May 28, 2025, 05:01 PM IST**

# Project Title

**Inventory Management Dashboard with Real-Time Insights**

# Project Scope

This project develops an Inventory Management Dashboard to monitor and manage inventory data across multiple store locations. It ingests data using Apache Pulsar, processes it with Apache Spark, detects anomalies using rule-based and ML methods, and provides predictive insights using ML models (ARIMA, XGBoost, Isolation Forest, Prophet). The dashboard is built with Streamlit and Plotly for interactive visualizations. The project aims to provide actionable insights for inventory managers through real-time metrics, anomaly detection, and forecasting.

# Learning Outcomes

* Data Ingestion: Implement near real-time data ingestion with Apache Pulsar.
* Distributed Processing: Use Apache Spark for scalable data processing.
* Anomaly Detection: Develop rule-based and ML-based anomaly detection.
* Machine Learning: Apply time-series forecasting models (ARIMA, Prophet).
* Visualization: Create an interactive dashboard with Streamlit and Plotly.
* Project Management: Plan, execute, and document a complex data project.

# Index

1. Document Metadata

2. Project Title

3. Project Scope

4. Planning

- Objectives

- Resources Required

- Key Stakeholders

5. Execution

- Startup Steps

- Progress Tracking

- Timeline

- Deviations from the Plan

6. Code Explanations

7. Troubleshooting Steps

8. Desired Outputs

9. Dependencies and Installations

10. Effort Tracking

11. Appendices

- Appendix 1: Dependency Setup

- Appendix 2: Cleanup Steps

- Appendix 3: Performance Enhancements

# Planning

## Objectives

* Ingest inventory data using Apache Pulsar and store it in SQLite.
* Process data with Apache Spark to compute metrics (e.g., total stock value).
* Detect numerical and text-based anomalies in inventory data.
* Generate ML predictions for stock levels using ARIMA, XGBoost, and Prophet.
* Build an interactive Streamlit dashboard to visualize metrics, anomalies, and predictions.
* Document the project lifecycle, including setup, execution, and troubleshooting.

## Resources Required

**Hardware:**

* Windows 11 PC, 16GB RAM, 500GB SSD.

**Software:**

* Python 3.8+ (`inventory\_venv`).
* Jupyter Notebook (for development).
* Apache Spark (for distributed processing).
* Apache Pulsar (for data ingestion).

**Libraries:**

* `pandas`, `pyspark`, `pulsar-client`, `streamlit`, `plotly`, `scikit-learn`, `prophet`, `xgboost`.

**Scripts:**

* `run\_system.py`: Orchestrates the entire system workflow.
* `ingest.py`: Generates and ingests data.
* `process\_inventory.py`: Processes data with Spark.
* `app.py`: Streamlit dashboard.
* `anomaly\_detection.py`: Detects numerical anomalies.
* ML scripts: `arima\_model.py`, `xgboost\_model.py`, etc.

**Files:**

* `D:\inventory\_project\data\`: SQLite databases (`inventory.db`, `inventory\_metrics.db`, `ml\_metrics.db`).

**Storage:**

* `D:\inventory\_project\`.

## Key Stakeholders

* Developer: Implements scripts, tests functionality, documents findings.
* Mentor: Provides guidance on Spark, Pulsar, and ML model implementation.
* End User: Inventory manager using the dashboard for decision-making.

# Execution

## Startup Steps

┌──(inventory\_venv)──[D:\inventory\_project]

│ 1. \*\*Activate Virtual Environment\*\*:

│ Ensure the virtual environment is activated with all dependencies installed.

└─$

┌──(inventory\_venv)──[D:\inventory\_project]

│ 2. \*\*Verify Databases\*\*:

│ Ensure `inventory.db`, `inventory\_metrics.db`, `ml\_metrics.db` exist in `D:\inventory\_project\data\`.

│ Databases are automatically checked by `run\_system.py`.

└─$

┌──(inventory\_venv)──[D:\inventory\_project]

│ 3. \*\*Start Pulsar and Spark\*\*:

│ Ensure Pulsar is running at `pulsar://<address>:<port>`.

│ Pulsar runs better on Vbox or WSL

└─$

┌──(inventory\_venv)──[D:\inventory\_project]

│ 4. \*\*Run the Entire System\*\*:

│ Execute `app.py` to start the entire workflow:

│ - Launches the Streamlit dashboard (`app.py`) at `http://localhost:<port>`.

│ - From the dashboard, users can ingest data, process metrics, run ML models, and detect anomalies by interacting with the UI.

└─$

## Progress Tracking

**Daily Logs:**

* Record script executions, errors (e.g., `Broken pipe` in Spark), and outputs in `logs/`.

**Metrics:**

* Dataset rows: ~100 records per ingestion run. Reduced to 20 for test case batch-based process.
* Dashboard: All visualizations (metrics, predictions, anomalies) rendered.
* Anomalies: Detected and stored in `ml\_metrics.db`.

**Reporting:**

* Logs: Check `ingest.log`, `process\_inventory.log` for execution details.
* Dashboard: Visuals rendered in Streamlit (e.g., stock value by store).

**Checkpoints:**

* Day 1: Data ingestion and processing scripts completed.
* Day 2: Dashboard and anomaly detection implemented.
* Day 3: ML models and documentation finalized.

## Timeline

**Day 1 (Ingestion and Processing):**

* Develop `ingest.py` and `process\_inventory.py`.
* Verify data in SQLite databases.

**Day 2 (Dashboard and Anomalies):**

* Build Streamlit dashboard with `app.py`.
* Implement anomaly detection with `anomaly\_detection.py`.

**Day 3 (ML Models and Documentation):**

* Run ML models (`arima\_model.py`, `xgboost\_model.py`).
* Document project in `docs/` (e.g., `README.md`, `DEMO.md`).

## Deviations from the Plan

**Pulsar Connection Issue:**

* Issue: Failed to connect to Pulsar (`pulsar://<address>:<port>`).

WSL : ip addr show eth0 | grep 'inet ' | awk '{print $2}' | cut -d'/' -f1

* Fix: Added retries in `ingest.py` and fallback to SQLite.
* Reason: Network instability.

**Spark Broken Pipe Error:**

* Issue: `Broken pipe` error during Spark DataFrame conversion to Pandas.
* Fix: Added retries and disabled Arrow optimization.
* Reason: Compatibility issue with PySpark on Windows.

# Code Explanations

**`app.py` - Streamlit Dashboard**

┌──(inventory\_venv)──[D:\inventory\_project]

│ import streamlit as st

│ import sqlite3

│ import pandas as pd

│ import plotly.express as px

│ import subprocess

│

│ # Set up logging

│ logging.basicConfig(

│ filename='D:\inventory\_project\logs\dashboard.log',

│ level=logging.INFO,

│ format='%(asctime)s - %(levelname)s - %(message)s'

│ )

│

│ # Set page config

│ st.set\_page\_config(page\_title='Inventory Dashboard', layout='wide')

│

│ # Title

│ st.title('Inventory Dashboard')

│ current\_time = datetime.now().strftime('%I:%M %p IST, %B %d, %Y')

│ st.markdown(f'\*\*Last Updated:\*\* {current\_time}')

│

│ # Sidebar navigation

│ st.sidebar.title('Inventory Management Dashboard')

│ section = st.sidebar.radio(

│ 'Select Section',

│ [

│ 'Ingest More Data',

│ 'Inventory Metrics',

│ 'Machine Learning Models',

│ 'Anomaly Detection and Notifications'

│ ]

│ )

└─$

Purpose: Serves as the main interface for the Inventory Management Dashboard, built with Streamlit. It allows users to ingest data, process metrics, run ML models, detect anomalies, and visualize results through an interactive UI.

**Key Areas Where Other Scripts Are Executed:**

* `ingest\_data()`: Executes `ingest.py` to ingest data from Pulsar when the 'Ingest More Data' button is clicked.
* `process\_metrics()`: Runs `process\_inventory.py` to process data with Spark when the 'Process Metrics' button is clicked.
* `run\_ml\_model()`: Executes ML scripts (e.g., `arima\_model.py`) based on user selection in the 'Machine Learning Models' section.
* `view\_ml\_metrics()`: Runs `view\_ml\_metrics.py` to process ML metrics for visualization.
* `view\_metrics()`: Executes `post\_process\_metrics.py` to post-process metrics for visualization.
* `view\_anomalies\_and\_notifications()`: Runs `anomaly\_detection.py`, `text\_anomaly\_detection.py`, and `query.py` for anomaly detection and notification generation.

**Console Output:**

* Streamlit server starts at `http://localhost:<port>`.
* Logs execution details in `dashboard.log` (e.g., 'Ingestion: Complete!', 'ARIMA Model: Complete!').

**Key Lines:**

* `st.set\_page\_config(...)`: Configures the Streamlit page with a wide layout.
* `st.sidebar.radio(...)`: Creates a sidebar for navigating between the four main sections.

**`ingest.py` - Data Generation and Ingestion**

┌──(inventory\_venv)──[D:\inventory\_project]

│ import pandas as pd

│ import numpy as np

│ from pulsar import Client

│

│ # Generate synthetic data

│ PRODUCTS = ['P001', 'P002', 'P003', 'P004', 'P005']

│ LOCATIONS = ['NYC', 'LA', 'Chicago', 'Houston', 'Miami']

│ CATEGORIES = ['Electronics', 'Clothing', 'Books', 'Toys', 'Food']

│

│ def generate\_inventory\_data(num\_records=10):

│ data = []

│ for \_ in range(num\_records):

│ record = {

│ 'product\_id': np.random.choice(PRODUCTS),

│ 'quantity': np.random.randint(1, 100),

│ 'price': round(np.random.uniform(10, 500), 2),

│ 'event\_timestamp': datetime.now().strftime('%Y-%m-%dT%H:%M:%SZ'),

│ 'store\_location': np.random.choice(LOCATIONS),

│ 'category': np.random.choice(CATEGORIES)

│ }

│ data.append(record)

│ return pd.DataFrame(data)

└─$

Purpose: Generates synthetic inventory data with product IDs, quantities, prices, timestamps, locations, and categories for testing.

**Console Output:**

* Generated 10 records with columns: product\_id, quantity, price, event\_timestamp, store\_location, category.

**Key Lines:**

* `PRODUCTS = [...]`: Defines sample product IDs.
* `np.random.choice(PRODUCTS)`: Randomly selects a product ID.
* `pd.DataFrame(data)`: Converts data to a Pandas DataFrame.

**`process\_inventory.py` - Data Processing with Spark**

┌──(inventory\_venv)──[D:\inventory\_project]

│ from pyspark.sql import SparkSession

│ from pyspark.sql.functions import col, sum, avg

│

│ # Initialize Spark session

│ spark = SparkSession.builder \

│ .appName('InventoryProcessing') \

│ .master('local[2]') \

│ .getOrCreate()

│

│ # Compute metrics

│ base\_data = df.withColumn('stock\_value', col('quantity') \* col('price'))

│ windowed\_data = base\_data.groupBy('store\_location', 'category') \

│ .agg(sum('quantity').alias('total\_quantity'), avg('price').alias('avg\_price'))

└─$

Purpose: Processes inventory data using Spark to compute metrics like total quantity and average price per store location and category.

**Console Output:**

* Spark DataFrame created with ~100 rows.
* Metrics computed: total\_quantity, avg\_price.

**Key Lines:**

* `SparkSession.builder ...`: Initializes a Spark session in local mode.
* `df.withColumn('stock\_value', ...)`: Computes stock value as quantity \* price.
* `groupBy(...).agg(...)`: Aggregates data by store location and category.

# Troubleshooting Steps

**Error: `Pulsar Connection Failed`**

* Cause: Unable to connect to Pulsar at `pulsar://172.27.235.96:6650`.

┌──(inventory\_venv)──[D:\inventory\_project]

│ Add retries in `ingest.py`:

│ for attempt in range(1, 4):

│ try:

│ client = Client(PULSAR\_SERVICE\_URL)

│ producer = client.create\_producer(PULSAR\_TOPIC)

│ break

│ except Exception as e:

│ if attempt == 3:

│ raise

└─$

* Verify:

┌──(inventory\_venv)──[D:\inventory\_project]

│ Ensure Pulsar is running:

│ docker ps | findstr pulsar

└─$

**Error: `Broken Pipe` in Spark**

* Cause: Spark DataFrame to Pandas conversion failed.

┌──(inventory\_venv)──[D:\inventory\_project]

│ Disable Arrow optimization:

│ spark.conf.set('spark.sql.execution.arrow.pyspark.enabled', 'false')

└─$

* Verify:

┌──(inventory\_venv)──[D:\inventory\_project]

│ Check Spark logs:

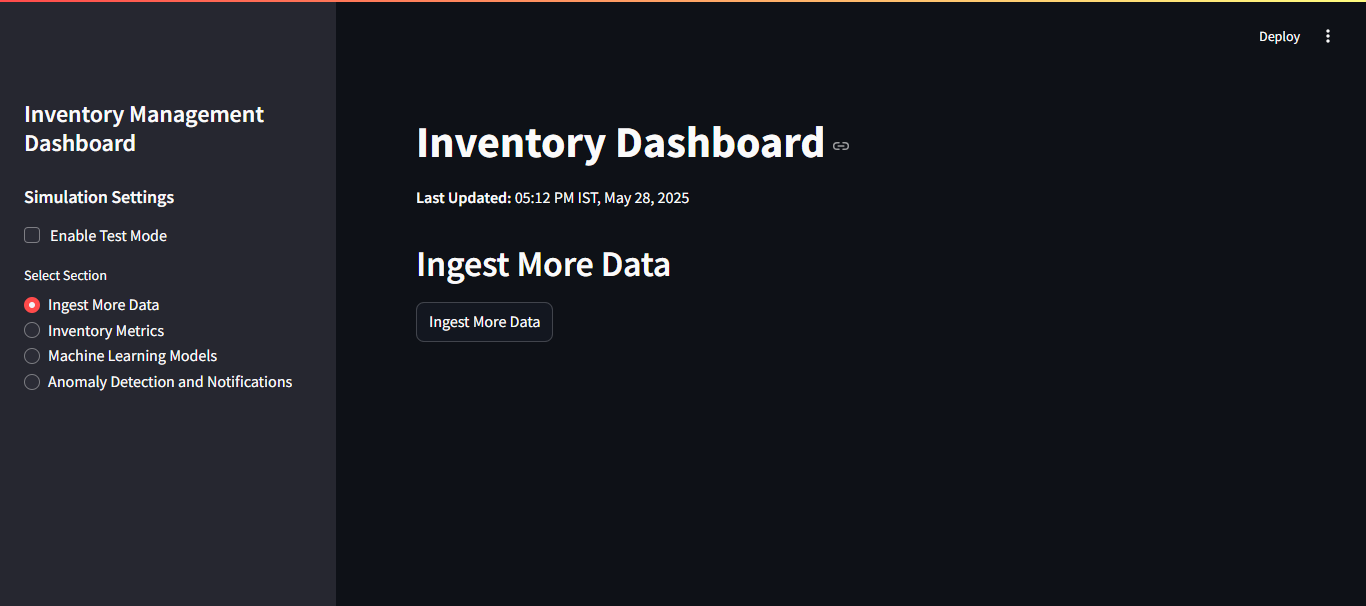
│ type D:\inventory\_project\spark-events\\*.log

└─$

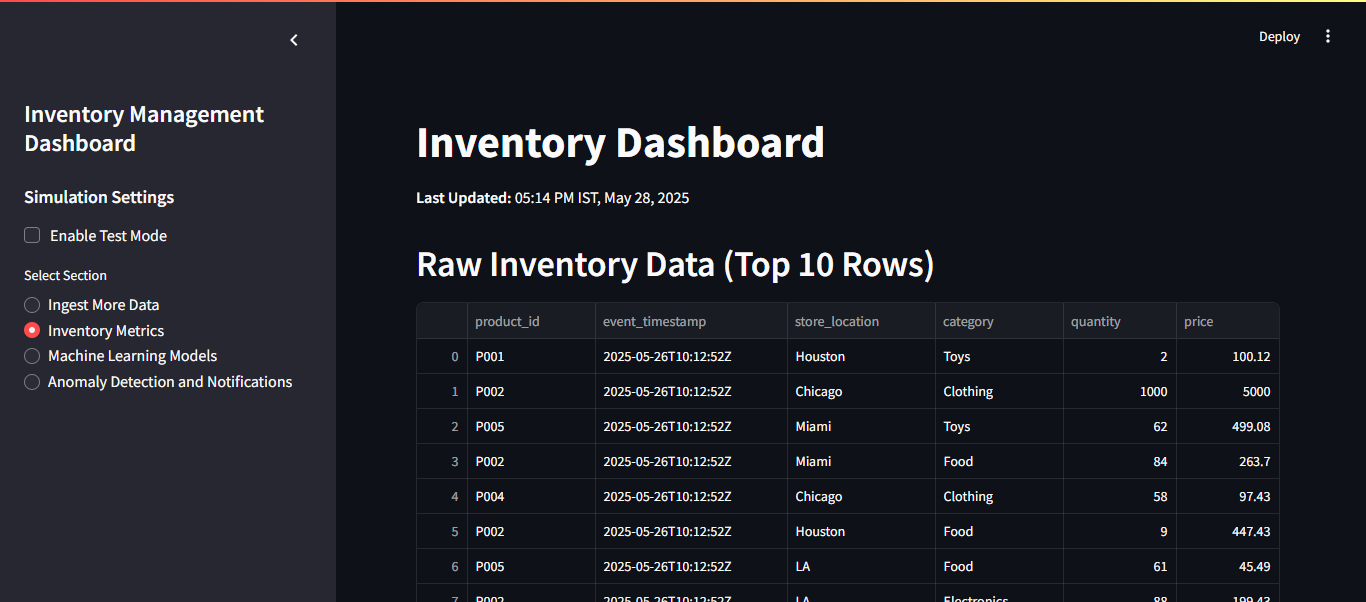
# Desired Outputs

**Streamlit Dashboard:**

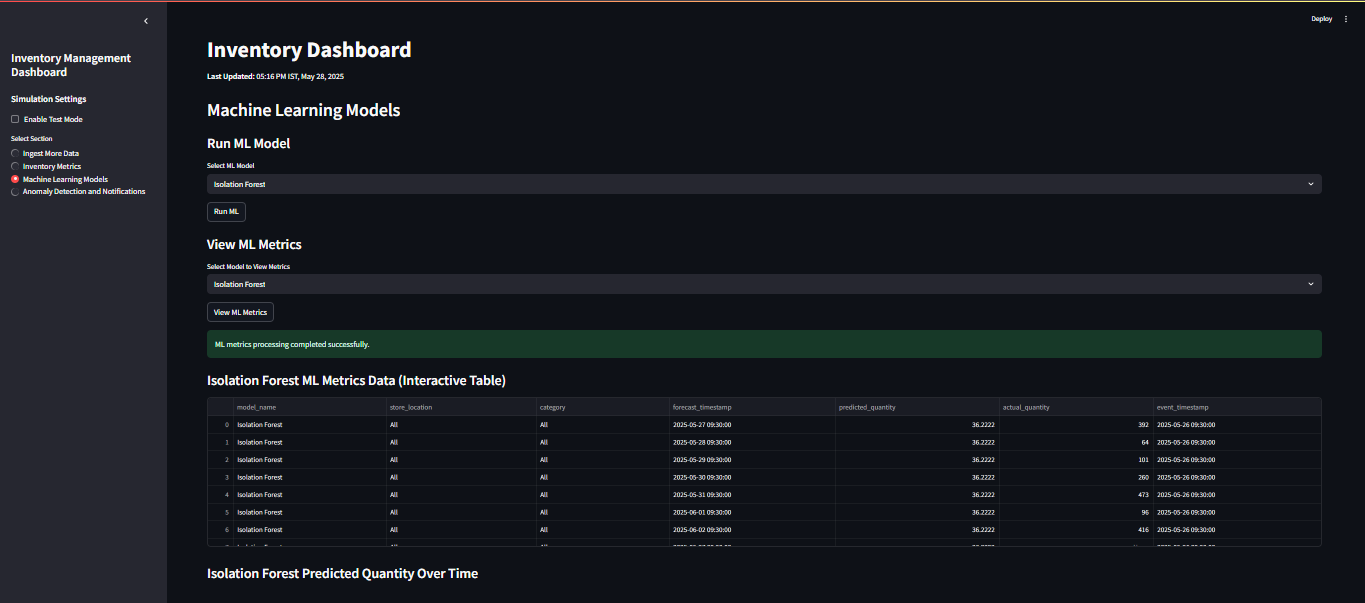
* Interactive dashboard at `http://localhost:8501`.
* Sections: Ingest More Data, Inventory Metrics, Machine Learning Models, Anomaly Detection and Notifications.



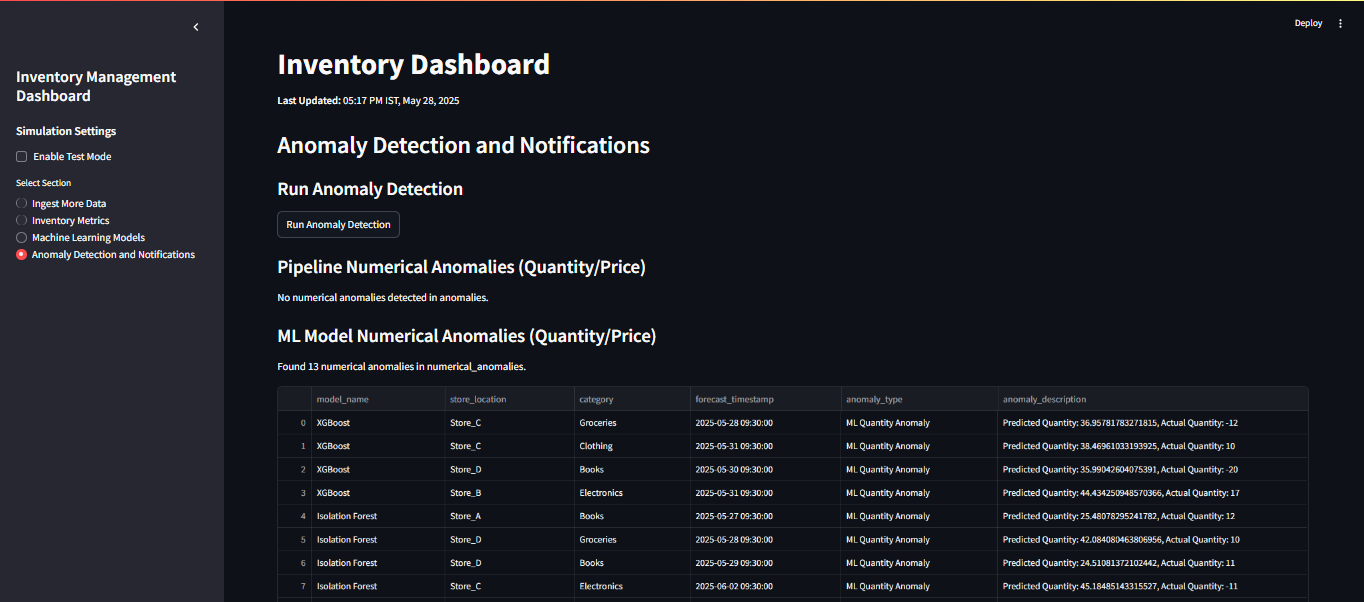
* [Screenshot: Ingest More Data Section (Version 1.0)]



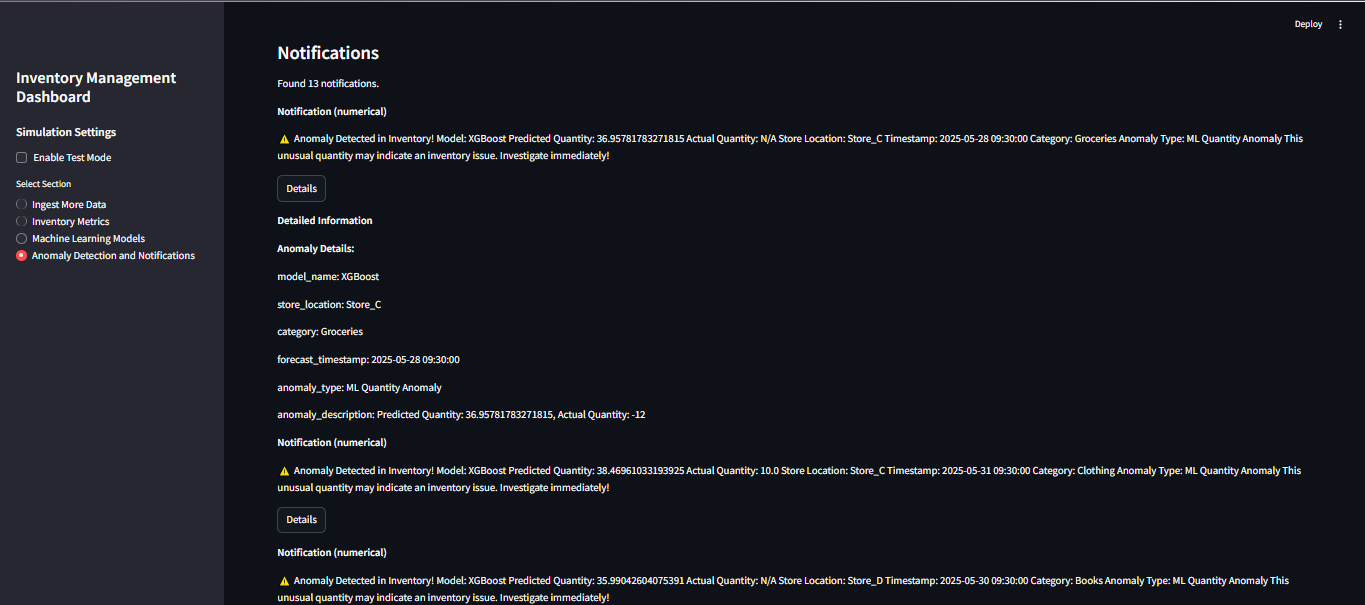
* [Screenshot: Inventory Metrics Section (Version 1.0)]



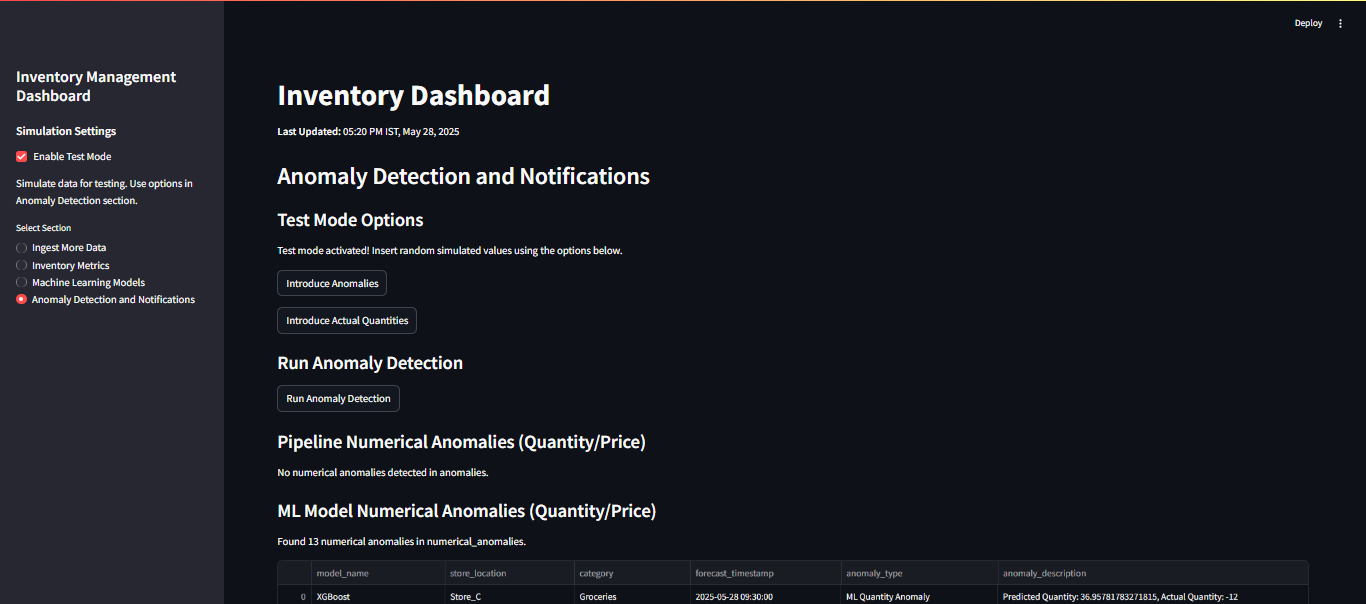
* [Screenshot: Machine Learning Models Section (Version 1.0)]



* [Screenshot: Anomaly Detection (Version 1.0)]



* [Screenshot: Notifications Section (Version 1.0)]



* [Screenshot: Test Mode with sim values (Version 1.0)]

**SQLite Databases:**

* `inventory.db`: Raw inventory data (~100 records per run).
* `inventory\_metrics.db`: Processed metrics (e.g., total stock value).
* `ml\_metrics.db`: ML predictions and anomalies.

**Logs:**

* `ingest.log`, `process\_inventory.log`: Execution details and errors.

# Dependencies and Installations

**Dependencies**

Python: 3.8+.

**Libraries:**

* `pandas`, `pyspark`, `pulsar-client`, `streamlit`, `plotly`, `scikit-learn`, `prophet`, `xgboost`.

**Scripts:**

* `run\_system.py`, `ingest.py`, `process\_inventory.py`, `app.py`, `anomaly\_detection.py`, etc.

**Tools:**

* Jupyter Notebook (for development).
* Apache Spark and Pulsar.

**Installation Commands**

┌──(inventory\_venv)──[D:\inventory\_project]

│ D:\>python -m venv D:\inventory\_venv

│ D:\>D:\inventory\_venv\Scripts\activate

│ (inventory\_venv) D:\>pip install pandas pyspark pulsar-client streamlit plotly scikit-learn prophet xgboost

└─$

# Effort Tracking

The table below summarizes the upskilling hours invested in the project from April 3 to May 28, 2025.

|  |  |  |
| --- | --- | --- |
| **Week** | **Date Range (2025)** | **Hours** |
| Week 1 | April 3–5 | 7.0 |
| Week 2 | April 6–12 | 11.0 |
| Week 3 | April 13–19 | 11.0 |
| Week 4 | April 20–26 | 9.0 |
| Week 5 | April 27–May 3 | 9.0 |
| Week 6 | May 4–10 | 11.0 |
| Week 7 | May 11–17 | 13.5 |
| Week 8 | May 18–24 | 27.5 |
| Week 9 | May 25–28 | 32.5 |
| Total | April 3–May 28 | 141.5 |

# Appendices

## Appendix 1: Dependency Setup

┌──(inventory\_venv)──[D:\inventory\_project]

│ Virtual Environment:

│ D:\>python -m venv D:\inventory\_venv

│ D:\>D:\inventory\_venv\Scripts\activate

│ Libraries:

│ (inventory\_venv) D:\>pip install pandas pyspark pulsar-client streamlit plotly scikit-learn prophet xgboost

└─$

## Appendix 2: Cleanup Steps

┌──(inventory\_venv)──[D:\inventory\_project]

│ Stop Streamlit:

│ Ctrl+C in terminal (if run manually).

│ Deactivate Environment:

│ (inventory\_venv) D:\>deactivate

│ Verify:

│ D:\>tasklist | findstr 'python streamlit'

└─$

## Appendix 3: Performance Enhancements

┌──(inventory\_venv)──[D:\inventory\_project]

│ Spark Optimization:

│ spark.conf.set('spark.sql.shuffle.partitions', '1') # Reduce partitions for local mode

│ Logging:

│ logging.basicConfig(filename='D:/inventory\_project'logs\project.log')

│ Backup Databases:

│ D:\>copy D:\inventory\_project\data\\*.db D:\inventory\_project\data\backup\

└─$