SmartStock Inventory Optimization for Retail Stores

Smart Stock Inventory System – Overview

Introduction

A Smart Stock Inventory System is a digital solution designed to help retail stores track their products automatically. It replaces manual stock counting with technology that updates inventory in real time, reduces errors, and ensures that important items are never out of stock.

Main Objective

- To monitor stock levels accurately and automatically
- To alert store owners about low stock, overstock, or expired items
- To make daily operations faster, smarter, and more organized

Python basics

1. Python Casting

- Casting = data type conversion
- Common conversions:
    - int() → integer
    - float() → floating number
    - str() → string
- Example: float(5) = 5.0

---

2. Function Initialization (def)

- Syntax:
- def function_name():

- # code
- Function can have parameters and return values**.**

---

3. f-string

- f-string =Used to insert variables directly into a string using {}.
- Fastest & cleanest string formatting.
- name = "Mannat"
- print(f"Hello {name}")

---

4. NumPy

- NumPy = Numerical Python (fast math library).

5. Pandas

- Pandas = data analysis and data cleaning library.
- DataFrame .
- Important operations:
  - df.head()
  - df.describe()

---

6. Matplotlib

- graph/visualization library.
- Graph types:
  - Line → plt.plot()
  - Bar → plt.bar()
  - Pie → plt.pie()
  - Scatter → plt.scatter()
- Basic functions:
  - plt.title(), plt.xlabel(), plt.ylabel(), plt.show()

<u>AI MODELS</u>

AI models are computer programs that **learn patterns from data** and then use that learning to **predict, understand, or generate** things on their own.

Company	**Famous Model Family**
Google	**Gemini**
OpenAI	**GPT** (Generative Pre-trained Transformer)
Groq	**Llama**

What is an LLM?

LLM = Large Language Model

It is an AI model trained on huge amounts of text to understand and generate human-like language.

Examples of LLMs

These are all LLMs:

OpenAI

- GPT-5
- GPT-4o
- GPT-3.5

Google

- Gemini
- Gemma

Meta

- Llama 3
- Llama 3.1

LLMs are trained using large datasets.

Chat Completion means using an AI model (LLM) to generate the next response in a conversation.
This is how GPT, Gemini, Groq (Llama), Claude, etc. work.

What is Chat Completion?

It is an API style where you send:

- messages (user + assistant history)
- The model replies with the next message

What is PostgreSQL (Postgres)?

PostgreSQL is an **open-source relational database management system (RDBMS)**.

In simple words:
It stores your data in **tables** (rows & columns)
Helps you **query**, **insert**, **update**, **delete**, and **manage data**
Uses **SQL** language

1. Type of Database

- **PostgreSQL** → *Object–Relational Database* (supports advanced data types, complex queries).
- **MySQL** → *Relational Database* (simple structure, very fast for basic queries).

---

2. Speed

- **PostgreSQL** → Slower for simple reads but *faster for complex queries*.
- **MySQL** → Very *fast for simple read-heavy operations*.

Why do we need Vector Databases?

Normal databases (**MySQL, PostgreSQL**) can't search:

- similar meaning sentences
- similar images

- related documents

They only match *exact words*, not *meaning*.

Vector DBs search using **similarity**, like:

- “Find documents similar to this one”
- “Find images that look like this picture”
- “Find product recommendations”

3 Main Types of Machine Learning

Supervised Learning

Learning with labeled data
The model is given **input + correct output**, and it learns the mapping.

Used for:

- Predicting prices
- Classifying emails (spam/not spam)
- Face recognition

Unsupervised Learning

Learning without labeled data
The model finds **patterns and groups** on its own.

Used for:

- Market segmentation
- Anomaly detection
- Recommendation systems

Reinforcement Learning (RL)

Learning by trial and error
The model gets **rewards** for good actions and **punishments** for bad actions.

Semi-Supervised Learning

Some data → labeled
Most data → unlabeled

Used when labeling is expensive.
Examples:

- Google Photos
- Medical images

1. API (Application Programming Interface)

Definition:
An API is a set of rules that allows two software applications to communicate and exchange data.

Simple meaning:
API works like a **messenger** between two programs. One program sends a request, API forwards it, and returns the response.

Real-life example:
A weather app requests today's temperature using a weather API and displays the result.

Benefits:

- Easy communication between software
- Reuse features without rewriting code
- Saves development time
- Secure, fast, structured communication

---

2. API Request Methods

GET Method

- Used to **fetch data** from the server.
- No request body/payload.
- Example: Getting user details.

POST Method

- Used to **send or create data** on the server.
- Contains payload (JSON data).

- Example: Creating a new user.

---

3. Payload

Definition:
Payload is the **actual data** sent inside an API request or response.

Example:
POST Request Payload:

---

4. Endpoint

Definition:
Endpoint is a **specific URL path** of an API where requests are made.

Examples:

- /api/users – GET all users
- /api/users/1 – GET a user by ID
- /api/users – POST to create a user

---

5. FastAPI vs Flask

Flask

- Lightweight web framework
- Beginner-friendly
- Manually handle validation
- Good for small/medium projects

Example:

```
from flask import Flask, jsonify


app = Flask(__name__)


@app.route("/hello")
def hello():
    return jsonify({"message": "Hello World"})
```

FastAPI

- Modern & high-performance

Example:

```
from fastapi import FastAPI


app = FastAPI()


@app.get("/hello")
def hello():
    return {"message": "Hello World"}
```

---

6. Python :

Create a virtual environment

```
python -m venv venv
```

or

```
python -m venv tenv
```

Activate environment

Windows:

```
venv\Scripts\activate
```

Deactivate

```
deactivate
```

---

7. requirements.txt

Create requirements file

requirements.txt

Install from requirements

```
pip install -r requirements.txt
```

Purpose:
Helps install the same packages on any system.

---

8. Git Commands

git add

- Adds files to staging area.

Add all files:

```
git add .
```

Add specific file:

```
git add filename.py
```

---

git commit

Saves staged changes:

```
git commit -m "message"
```

---

git push

Sends your commits to remote repository:

```
git push
```

---

git branch

View branches:

```
git branch
```

View all branches (local + remote):

```
git branch --all
```

Switch to another branch

```
git checkout branch_name
```

Or newer way:

```
git switch branch_name
```

Create and switch to a new branch

```
git checkout -b new_branch
```

or

```
git switch -c new_branch
```

---

git pull

Fetches latest changes and merges them into your branch:

```
git pull
```

---

git fetch --all

Downloads all changes from remote **without merging**:

```
git fetch –all
```

Database

A **database** is an organized collection of data that allows easy **storage, retrieval, updating, and management** of information. Databases help keep data structured and accessible for applications like banking systems, websites, social media platforms, and school management systems.

Why Do We Use a Database?

- To store large amounts of data safely
- To avoid mistakes and inconsistency
- To access data quickly
- To allow multiple users to work at the same time
- To keep data organized and structured

Types of Databases

1. Relational Databases (SQL)

Data is stored in **tables** with rows and columns. Uses **SQL language**.
Examples: MySQL, PostgreSQL, Oracle.

2. Non-Relational Databases (NoSQL)

Data stored in **documents, key-value pairs, graphs, or wide columns**.
Examples: MongoDB, Firebase, Cassandra.

Important Database Terms

- **Table:** Collection of rows and columns
- **Row (Record):** One entry of data
- **Column (Field):** Attribute of the data
- **Primary Key:** Unique identifier for each record
- **Foreign Key:** Connects two tables
- **Query:** Command to retrieve or update data
- **Index:** Speeds up searching

Normalization

Normalization is a database process used to **organize data, reduce redundancy, and improve data consistency**.
It breaks a large table into smaller linked tables so that data becomes clean and easy to manage.

Why Normalization Is Important

- Removes duplicate data
- Saves storage
- Ensures consistency
- Avoids update/delete anomalies

- Improves performance
- Makes database structure logical

Normal Forms (1NF, 2NF, 3NF)

First Normal Form (1NF)

Rules:

- No repeating columns
- Values must be **atomic** (single values only)
- Each table must have a **primary key**

Meaning:
Break multi-valued and repeating data into separate rows.

Second Normal Form (2NF)

Rules:

- Table must be in **1NF**
- No **partial dependency**
  (A non-key attribute should not depend on part of a composite key)

Meaning:
Non-key attributes must depend on the **whole primary key**.

Third Normal Form (3NF)

Rules:

- Table must be in **2NF**
- No **transitive dependency**
  (Non-key attribute should not depend on another non-key attribute)

Meaning:
Every column must depend **only** on the primary key.

Summary Table

Normal Form	**What It Fixes**	**Meaning**
1NF	Repeating / multi-valued data	Only single values per cell
2NF	Partial dependency	Non-key depends on whole primary key
3NF	Transitive dependency	Non-key depends only on primary key