# **Assignment: Python Programming for GUI Development**

Name : B. Harishwar Reddy

Registration Number : 192311338

Department : CSE

Date Of Submission : 26-08-2024

**Problem 2: Inventory Management System Optimization**

**Scenario:**

You have been hired by a retail company to optimize their inventory management system.  The company wants to minimize stockouts and overstock situations while maximizing  inventory turnover and profitability.

**Tasks:**

1. **Model the inventory system**: Define the structure of the inventory system, including  products, warehouses, and current stock levels.

2. **Implement an inventory tracking application**: Develop a Python application that  tracks inventory levels in real-time and alerts when stock levels fall below a certain  threshold.

3. **Optimize inventory ordering**: Implement algorithms to calculate optimal reorder  points and quantities based on historical sales data, lead times, and demand forecasts.

4. **Generate reports**: Provide reports on inventory turnover rates, stockout occurrences,  and cost implications of overstock situations.

5. **User interaction**: Allow users to input product IDs or names to view current stock  levels, reorder recommendations, and historical data.

**Deliverables:**

• **Data Flow Diagram**: Illustrate how data flows within the inventory management  system, from input (e.g., sales data, inventory adjustments) to output (e.g., reorder  alerts, reports).

• **Pseudocode and Implementation**: Provide pseudocode and actual code  demonstrating how inventory levels are tracked, reorder points are calculated, and  reports are generated.

• **Documentation**: Explain the algorithms used for reorder optimization, how historical  data influences decisions, and any assumptions made (e.g., constant lead times).

• **User Interface**: Develop a user-friendly interface for accessing inventory  information, viewing reports, and receiving alerts.

• **Assumptions and Improvements**: Discuss assumptions about demand patterns,  supplier reliability, and potential improvements for the inventory management  system's efficiency and accuracy.

**Solution:**

**Inventory Management System Optimization**

**1.Flow Diagram:**

Inventory Tracking

Report

Generation

User Interface

Reorder Optimisation

<

**2.Implementation:**

import pandas as pd

import numpy as np

# Sample data for demonstration

products = pd.DataFrame({

'product\_id': [1, 2, 3],

'product\_name': ['Widget A', 'Widget B', 'Widget C'],

'stock\_level': [100, 50, 20],

'reorder\_point': [30, 20, 15],

'order\_quantity': [70, 60, 50]

})

sales\_history = pd.DataFrame({

'product\_id': [1, 1, 2, 2, 3, 3, 3],

'quantity\_sold': [10, 15, 25, 30, 5, 10, 15]

})

def track\_inventory(product\_id):

"""Track current stock level and alert if below reorder point."""

product = products[products['product\_id'] == product\_id]

if product.empty:

return "Product not found"

stock\_level = product['stock\_level'].values[0]

reorder\_point = product['reorder\_point'].values[0]

if stock\_level < reorder\_point:

return f"Alert: Stock level of product {product\_id} is below reorder point. Current stock: {stock\_level}, Reorder Point: {reorder\_point}."

else:

return f"Stock level of product {product\_id} is sufficient. Current stock: {stock\_level}."

def optimize\_reorder(product\_id):

"""Calculate optimal reorder quantity based on sales history."""

product = products[products['product\_id'] == product\_id]

if product.empty:

return "Product not found"

average\_sales = sales\_history[sales\_history['product\_id'] == product\_id]['quantity\_sold'].mean()

lead\_time = 5 # assumed lead time in days

reorder\_point = average\_sales \* lead\_time

order\_quantity = reorder\_point - product['stock\_level'].values[0]

if order\_quantity < 0:

order\_quantity = 0

return f"Recommended reorder quantity for product {product\_id}: {order\_quantity}. Reorder Point: {reorder\_point}."

def generate\_report():

"""Generate a report on inventory turnover and stockouts."""

report = []

for \_, row in products.iterrows():

product\_id = row['product\_id']

stock\_level = row['stock\_level']

reorder\_point = row['reorder\_point']

turnover\_rate = sales\_history[sales\_history['product\_id'] == product\_id]['quantity\_sold'].sum() / stock\_level if stock\_level > 0 else 0

stockout = stock\_level < reorder\_point

report.append({

'Product ID': product\_id,

'Stock Level': stock\_level,

'Reorder Point': reorder\_point,

'Turnover Rate': turnover\_rate,

'Stockout': stockout

})

return pd.DataFrame(report)

# Example Usage

print(track\_inventory(2))

print(optimize\_reorder(3))

print(generate\_report())

**3.Display The Information:**

Stock level of product 2 is sufficient. Current stock: 50.

Recommended reorder quantity for product 3: 30.0. Reorder Point: 50.0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product ID | Stock Level | Reorder Point | Turnover Rate | Stockout |
| 1 | 100 | 30 | 0.25 | False |
| 2 | 50 | 20 | 1.10 | False |
| 3 | 20 | 15 | 1.50 | False |

**4.User-Input:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**5.Documentation:**

1. **track\_inventory(product\_id):**
   * Checks if the stock level is below the reorder point and returns an alert if needed.
2. **optimize\_reorder(product\_id):**
   * Calculates the optimal reorder quantity based on historical sales data and a fixed lead time.
3. **generate\_report():**
   * Creates a report showing the stock level, reorder point, turnover rate, and stockout status for each product.

**Assumptions and Improvements**

1. **Assumptions:**
   * Lead times are constant.
   * Historical sales data is accurate and stable.
   * Product data and sales history are up-to-date.
2. **Improvements:**
   * Implement dynamic lead times based on supplier performance.
   * Incorporate machine learning for more accurate demand forecasting.
   * Enhance the user interface for better interaction and data visualization.

This code provides a foundation for an inventory management system, allowing you to track stock levels, optimize reorder points, and generate reports. Further enhancements can include integrating with real-time databases and implementing more advanced algorithms for optimization and forecasting.