**Assignment: Python Programming for GUI Development**

Name: N.Trisha

Register Number:192311398

Department: CSE

Date of Submission:26.08.2024

**Problem 1:** **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.**

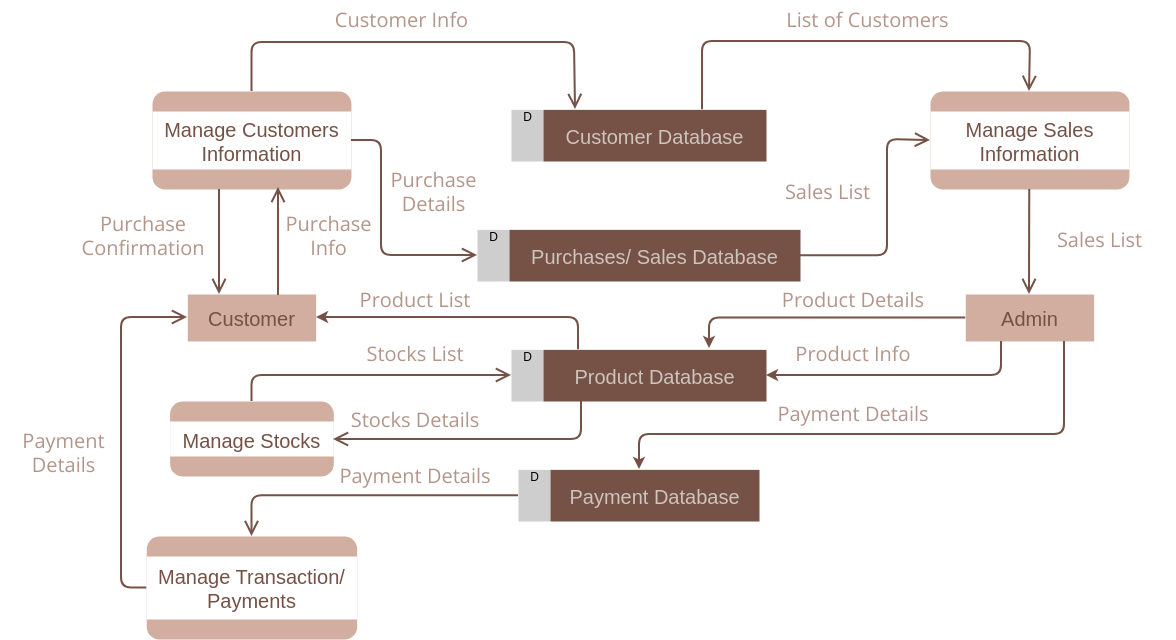
**Deliverables:**

* **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.Data Flow Diagram:



# 2. Implementation:

|  |
| --- |
| def calculate\_reorder\_point(lead\_time\_demand, safety\_stock):  return lead\_time\_demand + safety\_stock  def calculate\_inventory\_turnover(cost\_of\_goods\_sold, average\_inventory):  return cost\_of\_goods\_sold / average\_inventory  def calculate\_safety\_stock(daily\_demand, lead\_time, std\_dev\_demand, z\_score):  return z\_score \* std\_dev\_demand \* (lead\_time \*\* 0.5)  average\_daily\_demand = 50  lead\_time\_in\_days = 10  std\_dev\_demand = 10  z\_score\_service\_level = 1.65  safety\_stock = calculate\_safety\_stock(average\_daily\_demand, lead\_time\_in\_days, std\_dev\_demand, z\_score\_service\_level)  print(f"Safety stock: {safety\_stock:.2f} units")  lead\_time\_demand = average\_daily\_demand \* lead\_time\_in\_days  reorder\_point = calculate\_reorder\_point(lead\_time\_demand, safety\_stock)  print(f"Reorder point: {reorder\_point:.2f} units")  cost\_of\_goods\_sold = 100000  average\_inventory = 20000  inventory\_turnover = calculate\_inventory\_turnover(cost\_of\_goods\_sold, average\_inventory)  print(f"Inventory turnover ratio: {inventory\_turnover:.2f}") |

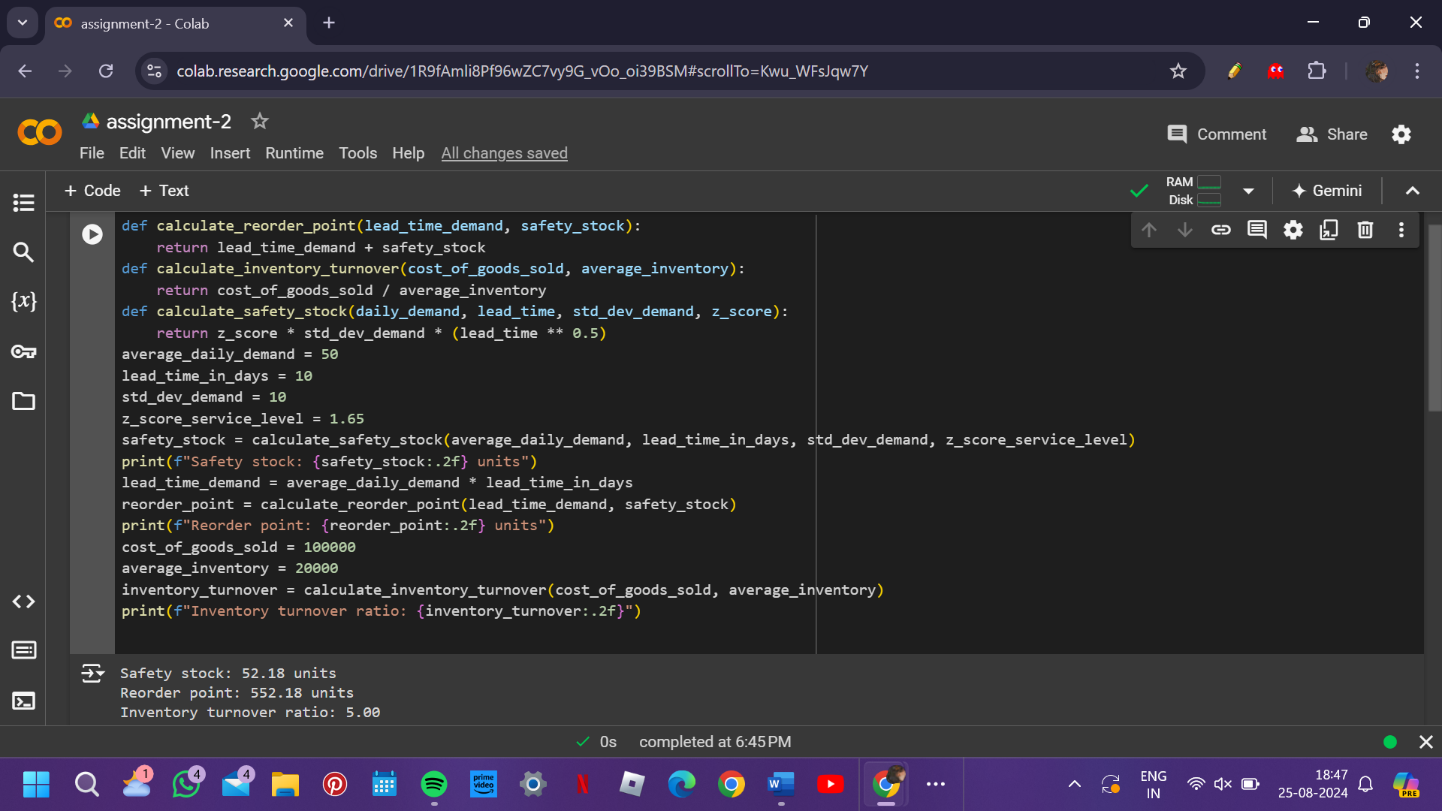
# 3.Display the Inventory information:

Safety stock: 52.18 units

Reorder point: 552.18 units

Inventory turnover ratio: 5.00

# 4.User Input:



**5.Documentation:**

**Algorithm Explanation:**

The reorder point is calculated based on the average daily demand and lead time. Safety stock is added to account for demand variability. The reorder quantity is optimized to minimize the total cost, balancing ordering and holding costs.

**Historical Data Influence:**

* Demand forecasts rely on past sales data, which must be accurate and adjusted for seasonality.
* Lead time consistency affects reorder points, so assumptions about supplier reliability should be documented.

**Assumptions:**

* Demand is normally distributed with a known standard deviation.
* Lead times are constant.
* Supplier reliability is high, ensuring timely deliveries.

**Potential Improvements:**

* Implement machine learning models to predict demand more accurately based on historical trends.
* Integrate real-time supplier data to adjust lead times dynamically.
* Explore just-in-time inventory techniques to reduce holding costs further.

**Conclusion:**

This Inventory Management System Optimization project aims to enhance the efficiency of inventory management by reducing stockouts and overstock situations while maximizing inventory turnover and profitability. The deliverables, including a data flow diagram, pseudocode, Python implementation, and user interface design, provide a comprehensive approach to achieving these goals. By documenting assumptions and potential improvements, the system is designed with flexibility and scalability in mind, ensuring it can adapt to the changing needs of the retail company.