Assignment –1

Problem 2: Inventory Management System Optimization

1. Data Flow Diagram

Data Flow Steps:

1. User Input: Users input product details, sales data, or requests for reports.

2. Inventory Tracking: The system updates inventory levels in real-time based on sales data or manual adjustments.

3. Reorder Point Calculation: The system calculates optimal reorder points and quantities based on historical data, lead times, and demand forecasts.

4. Alerts: The system generates alerts when stock levels fall below predefined thresholds.

5. Reporting: The system generates reports on inventory turnover, stockouts, and overstock costs.

6. User Output: Users receive inventory alerts and can view reports or current stock levels.

Data Flow Diagram (DFD):

[User] ---> (Input Data / Request Reports) ---> [Inventory Management System]

[Inventory Management System] ---> (Update Inventory) ---> [Inventory Database]

[Inventory Database] ---> (Check Levels) ---> [Inventory Management System]

[Inventory Management System] ---> (Generate Alerts / Reports) ---> [User]

2. Pseudocode for Inventory Management System

Here’s a high-level pseudocode outline:

BEGIN

PROMPT user for action:

1. View current stock levels

2. View reorder recommendations

3. View historical data

4. Generate reports

IF user selects "View current stock levels":

PROMPT user to input product ID or name

DISPLAY current stock levels for the selected product

IF user selects "View reorder recommendations":

CALCULATE optimal reorder point and quantity based on historical sales, lead times, and demand forecasts

DISPLAY reorder recommendations

IF user selects "View historical data":

PROMPT user to input product ID or name

DISPLAY historical sales and inventory data for the selected product

IF user selects "Generate reports":

GENERATE inventory turnover, stockout, and overstock cost reports

DISPLAY reports to the user

MONITOR inventory levels in real-time

IF inventory level for any product falls below threshold:

ALERT user to reorder

END

```

### 3. Python Implementation

#### Inventory Management System

```python

import Json

# Sample inventory and sales data (in practice, this would be in a database)

inventory = {

'101': {'name': 'Product A', 'stock': 50, 'reorder\_point': 20, 'lead\_time': 5},

'102': {'name': 'Product B', 'stock': 10, 'reorder\_point': 15, 'lead\_time': 3},

}

sales\_history = {

'101': [5, 10, 4, 6],

# Example sales data for Product A

'102': [2, 3, 5, 1],

# Example sales data for Product B

}

def view\_stock\_levels():

for product\_id, details in inventory.items():

print(f"Product ID: {product\_id}, Name: {details['name']}, Stock: {details['stock']}")

def calculate\_reorder(product\_id):

details = inventory[product\_id]

avg\_sales = sum(sales\_history[product\_id]) / len(sales\_history[product\_id])

reorder\_quantity = avg\_sales \* details['lead\_time']

return reorder\_quantity

def reorder\_recommendations():

for product\_id, details in inventory.items():

if details['stock'] < details['reorder\_point']:

reorder\_quantity = calculate\_reorder(product\_id)

print(f"Reorder Recommendation for {details['name']}: {reorder\_quantity} units")

def generate\_reports():

turnover\_rate = {}

stockouts = {}

overstock\_costs = {}

for product\_id, details in inventory.items():

avg\_sales = sum(sales\_history[product\_id]) / len(sales\_history[product\_id])

turnover\_rate[product\_id] = avg\_sales / details['stock']

stockouts[product\_id] = len([sale for sale in sales\_history[product\_id] if sale > details['stock']])

overstock\_costs[product\_id] = details['stock'] - avg\_sales

print("Inventory Turnover Rates:", json.dumps(turnover\_rate, indent=4))

print("Stockouts:", json.dumps(stockouts, indent=4))

print("Overstock Costs:", json.dumps(overstock\_costs, indent=4))

def main ():

while True:

print("\nInventory Management System")

print("1. View Current Stock Levels")

print("2. View Reorder Recommendations")

print("3. View Historical Data")

print("4. Generate Reports")

print("5. Exit")

choice = input ("Select an option: ")

if choice == '1':

view\_stock\_levels()

elif choice == '2':

reorder\_recommendations()

elif choice == '3':

product\_id = input ("Enter Product ID: ")

print (f"Historical Data for Product {product\_id}: {sales\_history[product\_id]}")

elif choice == '4':

generate\_reports()

elif choice == '5':

break

else:

print ("Invalid option. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main ()

4. Documentation

Algorithms:

1. Reorder Point Calculation:

- The reorder point is calculated based on historical sales data and lead times. The average daily sales rate is calculated, and then multiplied by the lead time to determine the reorder quantity.

2. Inventory Turnover:

- Inventory turnover is calculated by dividing the average sales by the current stock level. A higher turnover rate indicates efficient inventory management.

3. Stockout Occurrences:

- Stockout occurrences are identified by checking historical sales data against available stock. If sales exceed available stock, a stockout is recorded.

4. Overstock Costs:

- Overstock costs are estimated by subtracting average sales from current stock levels. This helps identify products that are overstocked.

Historical Data Influence:

- Historical sales data is critical for forecasting demand, calculating reorder points, and identifying stockout and overstock situations.

Assumptions:

- Constant lead times are assumed, though this may vary in real-world scenarios.

- Sales patterns are consistent and predictable based on historical data.

5. User Interface

- The command-line interface (CLI) allows users to interact with the system, view stock levels, get reorder recommendations, and generate reports.

- For a more user-friendly experience, this CLI can be upgraded to a graphical user interface (GUI) using tools like Tkinter or a web interface using Flask or Django.

6. Assumptions and Potential Improvements

Assumptions:

- The system assumes constant lead times for suppliers and stable demand patterns.

- The system assumes that historical sales data is a reliable predictor of future demand.

Potential Improvements:

Dynamic Lead Times: Incorporate dynamic lead times based on supplier performance data.

Advanced Forecasting: Implement more advanced demand forecasting algorithms using machine learning.

Integration with POS: Integrate the system with a Point-of-Sale (POS) system for real-time sales data.

Supplier Reliability: Incorporate supplier reliability metrics to adjust reorder points dynamically.

User Interface: Develop a more sophisticated user interface for better user experience.

## Output:

Inventory Management System

1. View Current Stock Levels

2. View Reorder Recommendations

3. View Historical Data

4. Generate Reports

5. Exit

Select an option: 1

Product ID: 101, Name: Product A, Stock: 50

Product ID: 102, Name: Product B, Stock: 10

Inventory Management System

1. View Current Stock Levels

2. View Reorder Recommendations

3. View Historical Data

4. Generate Reports

5. Exit

Select an option: 2

Reorder Recommendation for Product B: 6.25 units

Inventory Management System

1. View Current Stock Levels

2. View Reorder Recommendations

3. View Historical Data

4. Generate Reports

5. Exit

Select an option: 3

Enter Product ID: 101

Historical Data for Product 101: [5, 10, 4, 6]

Inventory Management System

1. View Current Stock Levels

2. View Reorder Recommendations

3. View Historical Data

4. Generate Reports

5. Exit

Select an option: 4

Inventory Turnover Rates: {

"101": 0.25,

"102": 0.55

}

Stockouts: {

"101": 0,

"102": 0

}

Overstock Costs: {

"101": 46.25,

"102": 4.25

}

Inventory Management System

1. View Current Stock Levels

2. View Reorder Recommendations

3. View Historical Data

4. Generate Reports

5. Exit

Select an option: 5

Exiting the system...