**Exercise 1: Inventory Management System**

**1. Understand the Problem:**

o Explain why data structures and algorithms are essential in handling large inventories.

o Discuss the types of data structures suitable for this problem.

*Significance of Data Structures and Algorithms:*

The data types that we use and the approach of numerical algorithms to call are important when handling large scale inventories due to:

Efficiency : They permit to quickly access, insert, delete and update data. Inefficient operations can dramatically slow down large inventories.

Performance: The use of appropriate data structures also results in improved performance without a compromise with scalability.

Memory : Using memory in a moderate way is important not to consume too much of the memory, which even critical as it causes overconsumption OOM depending on the size depends with large systems.

Complex Operations : Advanced algorithms can handle complex queries or operations like searching for products with some criteria and creating reports.

*Suitable Data Structures:*

For an inventory management system, the following data structures are suitable:

**ArrayList (Dynamic Array)**: Useful for maintaining an ordered collection of products. However, it has limitations in terms of insertion and deletion, as these operations can be costly.

**HashMap (Hash Table)**: Ideal for quick access, insertion, and deletion of products by their unique identifiers. It provides average O(1) time complexity for these operations, making it highly efficient for inventory management.

**Linked List**: Useful for situations where frequent insertions and deletions are required, but it has a higher access time compared to arrays or hash tables.

**Binary Search Tree (BST):** Useful for maintaining sorted data, but it can degrade to O(n) time complexity for operations in the worst case. Balanced trees like AVL or Red-Black Trees can maintain O(log n) complexity for operations.

**4. Analysis:**

o Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.

o Discuss how you can optimize these operations.

*Time Complexity Analysis for Hashmap :*

* **Add Product**: O(1) average time complexity for insertion.
* **Update Product**: O(1) average time complexity for access and update.
* **Delete Product**: O(1) average time complexity for deletion.

*Optimization :*

* **Efficient Indexing**: Use hash tables for quick access and updates.
* **Batch Processing**: Implement batch processing for large-scale updates or deletions to reduce overhead.
* **Concurrency Control**: Implement locks or other concurrency control mechanisms if the system will handle multiple simultaneous operations.
* **Caching**: Utilize caching for frequently accessed data to reduce access time