

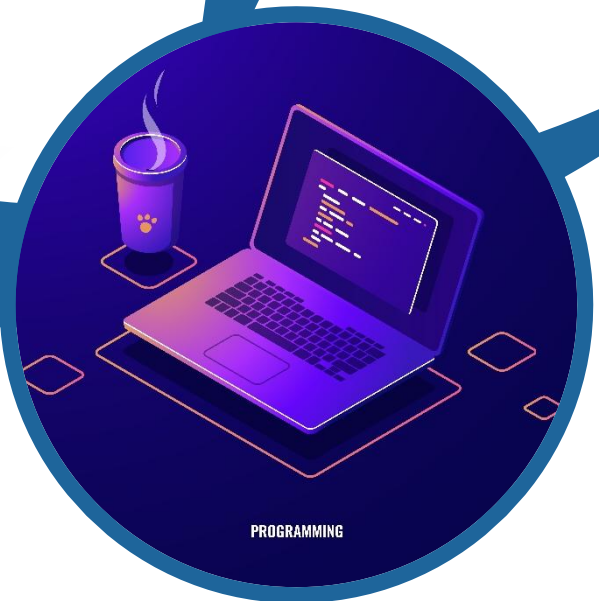


# UNIVERSITY OF BARISHAL

## EDGE: BU-CSE DIGITAL SKILLS

### Lab Report on Inventory Management of Walton Wholesale Store by Using Basic Programming with Python

Date of Submission: February 16, 2024



**Submitted To**  
Md. Rashid Al Asif  
Assistant Professor  
Department of Computer  
Science & Engineering  
University of Barishal

**Submitted By**  
Md Julker Nayem  
Roll : 03-008-04  
Batch: Python 8  
Department of Marketing  
University of Barishal

## Table of Contents

Topic Name	Page Number
Abstract	2
<b>Executive Summary</b>	3
<b>1. Introduction</b>	4
1.1 Overview of Walton's Inventory System	4
1.2 Importance of Inventory Management in Wholesale & Manufacturing	4
1.3 Role of Python Code in Automating Inventory Tasks	5
<b>2. Objectives of the Inventory Management System</b>	5
2.1 Efficient Tracking of Stock Levels	5
2.2 Automating Stock Replenishment	6
2.3 Reducing Inventory Holding Costs	6
2.4 Enhancing Supply Chain Visibility	7
<b>3. Tools &amp; Technologies Used</b>	7
3.1 Programming Language: Python	7
3.2 Libraries Used	8
3.3 Program's Functionality	9
<b>4. Features of the Python-based Inventory System</b>	13
4.1 Stock Tracking: Automatically Updates Inventory Levels	13
4.2 Order Management: Generates Purchase Orders When Stock is Low	13
4.3 Sales Integration: Updates Inventory Based on Sales Data	13
4.4 Supplier Management: Keeps Track of Suppliers and Restocking Schedules	14
4.5 Reporting & Analytics: Generates Reports on Stock Movement, Demand Forecasting, and Sales Trends	14
<b>5. Challenges of Inventory Management System</b>	14
<b>6. Conclusion</b>	15

## **Abstract**

The report explores the design, implementation, and analysis of an automated inventory management system developed for Walton, a major player in the wholesale electronics and home appliance sector. The system is built using Python, a versatile and user-friendly programming language, and aims to address the inefficiencies of manual inventory management. By automating key tasks such as stock tracking, product additions, and order management, the system reduces human errors, optimizes stock levels, and ensures real-time availability of inventory data.

The automation of inventory processes in wholesale is crucial, especially for large-scale businesses like Walton, which deal with a diverse range of products and high sales volumes. The Python-based system is designed to integrate seamlessly with Walton's existing operations, providing scalability for future growth. The implementation highlights Python's capability to handle inventory-related tasks efficiently while leaving room for potential integration with external databases, APIs, and additional functionalities such as demand forecasting, sales data integration, and supplier management. This project demonstrates how technology-driven inventory automation can reduce operational costs, improve decision-making, and support a more agile business model.

## Executive Summary

Effective inventory management is one of the cornerstones of operational success in the wholesale and manufacturing industries. Walton, a prominent name in Bangladesh's electronics and home appliance market, has a vast inventory system that requires constant tracking, updating, and replenishment. Managing inventory at such a scale manually leads to numerous challenges such as human errors, delays in restocking, and overstocking, which in turn inflate costs and affect customer satisfaction. To address these challenges, this report examines the development of an automated inventory management system using Python to streamline Walton's stock management processes.

The primary goal of this Python-based system is to improve the efficiency of Walton's inventory operations by automating repetitive tasks and minimizing human intervention. The system performs essential inventory tasks such as adding new products, updating stock levels, displaying product information, and editing or deleting product entries. By eliminating the need for manual data entry, the system reduces errors, improves accuracy, and saves time, allowing staff to focus on more strategic tasks. One of the system's key features is real-time stock monitoring, which ensures that Walton's wholesale stores and warehouses are consistently updated with current stock levels. This minimizes the risk of both stockouts (running out of products) and overstocking, which can lead to excessive holding costs. The automation of stock replenishment is also a vital component, allowing the system to monitor predefined stock thresholds and notify staff or even trigger reorders when stock levels fall below critical levels.

The system is highly scalable and can be expanded to integrate advanced features such as supplier management, where information about suppliers and restocking schedules can be tracked automatically, and sales data integration, where inventory is automatically updated based on sales transactions. Additionally, using Python allows for the potential incorporation of data analytics tools, providing Walton with reports on stock movement, sales trends, and demand forecasting, thus helping them make informed decisions about future inventory needs.

In its current form, the system significantly enhances Walton's inventory control by providing a centralized, automated solution for stock management. However, as Walton continues to grow, the system's flexibility allows for further development, ensuring it will remain relevant and capable of handling increasingly complex inventory challenges. This project underscores the importance of automation in modern wholesale operations, particularly in a competitive, fast-moving market.

# **1. Introduction**

## **1.1 Overview of Walton's Inventory System**

Walton, a leading brand in electronics and home appliances, operates a vast supply chain that requires efficient inventory management. The company's inventory system is designed to ensure smooth stock tracking, real-time updates, and optimal product availability across its wholesale stores and warehouses. By leveraging technology, Walton minimizes the risks of overstocking and stockouts, leading to improved operational efficiency and cost savings.

The system integrates automated stock monitoring, demand forecasting, and supplier coordination to maintain a seamless supply chain. Real-time stock monitoring helps track product levels across multiple locations, while automated replenishment ensures that stock is restocked at the right time to prevent shortages. Additionally, the use of data-driven insights allows Walton to optimize inventory based on sales trends and customer demand. Warehouse optimization further enhances storage efficiency, reducing unnecessary holding costs. By aligning stock levels with distribution and sales, Walton ensures a consistent and efficient inventory management process that enhances productivity and improves customer satisfaction.

## **1.2 Importance of Inventory Management in Wholesale & Manufacturing**

Inventory management plays a crucial role in both the wholesale and manufacturing sectors, as it directly impacts profitability, efficiency, and customer experience. In the wholesale industry, proper inventory management helps prevent stockouts and overstocking, ensuring that customers always find the products they need without businesses holding excessive unsold stock. This leads to better cash flow management, as resources are not unnecessarily tied up in inventory. A well-organized inventory system also enhances customer satisfaction by making sure that products are available when needed, strengthening brand loyalty. Additionally, it helps reduce storage and holding costs while enabling data-driven decision-making for forecasting future demand.

In manufacturing, inventory management is essential for optimizing production planning. Having the right raw materials available at the right time ensures that production runs smoothly without unnecessary downtime. Proper inventory control also reduces waste and losses by preventing excess stock that could expire or become obsolete. Furthermore, it enhances overall supply chain efficiency by streamlining coordination between suppliers, production units, and distribution channels. A well-managed inventory system minimizes production delays, improves cost efficiency, and ensures that manufacturing operations run seamlessly. Both

wholesale and manufacturing industries rely on efficient inventory management to maintain operational success, reduce costs, and remain competitive in their respective markets.

### **1.3 Role of Python Code in Automating Inventory Tasks**

Python plays a significant role in automating inventory management tasks. The uploaded Python code is a perfect example of how automation can simplify the otherwise cumbersome and error-prone manual processes. By using Python to automate stock tracking, replenishment, and record management, businesses like Walton can streamline operations, reduce human errors, and improve decision-making.

In the context of the provided Python code, key inventory management tasks such as adding new products, displaying product information, searching for existing products, editing product details, and deleting products are all automated. The use of Python eliminates the need for manual data entry and provides an organized and systematic approach to inventory control. Additionally, Python's ability to integrate with databases or external APIs can further enhance the scope of automation by enabling real-time stock updates and synchronization across various platforms.

Python also enables the easy scaling of the inventory management system, allowing businesses like Walton to handle a growing number of products and data with minimal manual intervention. The simplicity and efficiency of Python make it a perfect choice for small businesses as well as large corporations aiming to automate their inventory systems.

## **2. Objectives of the Inventory Management System**

The primary objective of any automated inventory management system is to optimize the processes of stock management and product tracking. With Python-based automation, Walton's inventory management system seeks to meet several important goals, ensuring both efficiency and accuracy in its operations.

### **2.1 Efficient Tracking of Stock Levels**

One of the fundamental objectives of the inventory management system is the efficient tracking of stock levels. Using the Python code provided, stock data is easily recorded and maintained in a structured format. Each product's details, such as total stock, number of orders, and

shortages, are captured and stored in a text file (or CSV format, in the enhanced version). This ensures that the inventory data is always up to date and that stock levels are monitored in real-time.

The Python script allows for continuous tracking and provides an organized record of product availability. This automatic tracking ensures that no product is overstocked or understocked, helping Walton make informed decisions about reordering products. Furthermore, the ability to search for specific products and view their stock levels directly from the system makes it easier to manage and monitor product movement.

## **2.2 Automating Stock Replenishment**

Automating stock replenishment is another key objective of the system. By continuously monitoring stock levels and automatically updating product information in real-time, Walton can implement a process where stock is replenished based on predefined thresholds. For example, when the total stock of a product reaches a set minimum level, the system can trigger a stock replenishment action, either by notifying relevant staff or directly placing orders with suppliers.

In the case of the Python code, though this specific functionality isn't explicitly built in, the system can be easily extended to include stock-level triggers. This ensures that products are reordered in time, preventing stockouts and ensuring uninterrupted sales.

## **2.3 Reducing Inventory Holding Costs**

Inventory holding costs are a significant concern for businesses like Walton, as they involve storage fees, insurance, and the risk of product deterioration. The efficient inventory management system built using Python helps reduce holding costs by providing accurate data on stock levels and allowing for better control of stock.

With real-time tracking of stock levels and automated stock management, Walton can avoid overstocking, which leads to higher holding costs. By automating the process of product ordering and stock monitoring, the system ensures that inventory levels are maintained at optimal levels, which in turn reduces unnecessary expenditures related to excess stock storage.

Furthermore, by automating tasks like adding new products and updating stock levels, Walton can reduce the need for physical stock audits, further lowering the overall cost of inventory management.

## **2.4 Enhancing Supply Chain Visibility**

Supply chain visibility is crucial for smooth business operations, especially in the wholesale and manufacturing sectors. By implementing an automated inventory management system using Python, Walton gains enhanced visibility into the entire supply chain process. Real-time access to inventory data allows for better communication and coordination between different parts of the supply chain, ensuring that stock movements, from warehouse to wholesale shelves, are efficiently tracked and managed.

The Python code in the report provides a simple yet effective way to manage product data and track stock movements. By using a CSV or text-based file format to store inventory information, the system can integrate with other supply chain management tools, such as those used for order fulfillment, logistics, and supplier management. This integration ensures that Walton can have a clear picture of its entire supply chain, leading to more informed decision-making and proactive measures to address potential supply chain disruptions.

## **3. Tools & Technologies Used**

### **3.1 Programming Language: Python**

Python is the primary programming language used for the development of the automated inventory management system. Python is well-known for its simplicity, readability, and vast ecosystem of libraries, which make it ideal for developing automation systems. The uploaded code leverages Python to automate various inventory management tasks, such as adding products, editing product details, tracking stock levels, and deleting products. Python's extensive standard library and third-party packages provide functionality that can be easily extended to meet specific business requirements.

Python's versatility is crucial for businesses like Walton, as it allows for quick prototyping, easy integration with other systems, and effective automation of complex processes. It is also highly scalable, making it suitable for both small businesses and large organizations.



## 3.2 Libraries Used

### 1. os Module

The **os** module in Python is part of the standard library and provides a way to interact with the operating system. It is essential for handling tasks related to the file system, directories, and the environment in which the Python script is running.

`os.path.exists(FILE_NAME)`: This function checks if a file (in this case, "products.txt") exists in the system. It returns True if the file exists and False if it doesn't. This is important for ensuring that the program doesn't attempt to open a non-existing file, which would raise an error.

### 2. input() Function

The **input()** function is used to gather user input in Python. It reads a line of text entered by the user and returns it as a string. This function is interactive, allowing users to provide data dynamically, making the program more flexible and user-driven.

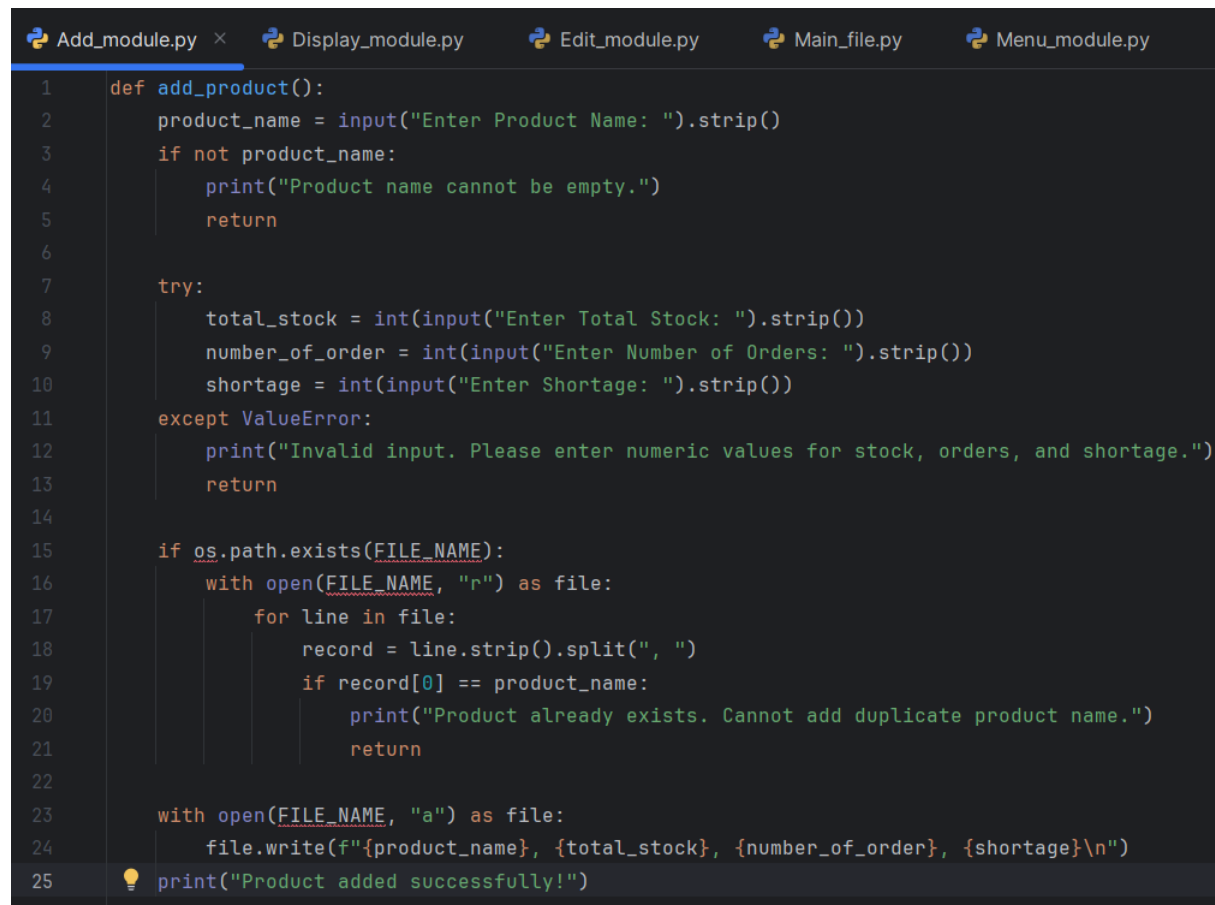
`input()` function is used in multiple parts of the program, such as when adding a new product, searching for a product, editing product details, or deleting a product. Each time a user needs to provide information (like product name, stock, number of orders, or shortage), `input()` is used to prompt them for the necessary input. In the `add_product()` function, for example, it is used to prompt the user for details like the product name and stock information. The program reads the response from the user and processes it further.

### 3. int() Function

The **int()** function is a built-in Python function that converts a string or another numeric type into an integer. It is essential for validating user input, particularly when dealing with numerical data. The `int()` function is used in the code to ensure that the values entered for total stock, number of orders, and shortage are numeric. For example, in the `add_product()` function, `int()` is used to convert the user's input for `total_stock`, `number_of_order`, and `shortage` into integers. This guarantees that the values are suitable for numerical operations (e.g., adding to inventory, calculating shortages). The `int()` function plays a critical role in data validation, ensuring that the system accepts only valid numerical inputs for inventory-related operations.

### 3.3 Program's Functionality:

#### Add\_product():

The image shows a code editor with five tabs: 'Add\_module.py', 'Display\_module.py', 'Edit\_module.py', 'Main\_file.py', and 'Menu\_module.py'. The 'Add\_module.py' tab is active, displaying the following Python code:

```
1 def add_product():
2     product_name = input("Enter Product Name: ").strip()
3     if not product_name:
4         print("Product name cannot be empty.")
5         return
6
7     try:
8         total_stock = int(input("Enter Total Stock: ").strip())
9         number_of_order = int(input("Enter Number of Orders: ").strip())
10        shortage = int(input("Enter Shortage: ").strip())
11    except ValueError:
12        print("Invalid input. Please enter numeric values for stock, orders, and shortage.")
13        return
14
15    if os.path.exists(FILE_NAME):
16        with open(FILE_NAME, "r") as file:
17            for line in file:
18                record = line.strip().split(", ")
19                if record[0] == product_name:
20                    print("Product already exists. Cannot add duplicate product name.")
21                    return
22
23    with open(FILE_NAME, "a") as file:
24        file.write(f"{product_name}, {total_stock}, {number_of_order}, {shortage}\n")
25    print("Product added successfully!")
```

This function is responsible for adding a new product to the inventory file. It first checks if the product name is unique by verifying the file for existing products. Then, it gathers the necessary details (product name, total stock, number of orders, shortage) from the user. If the input is valid, it writes the product information to the file. If invalid data is entered (e.g., non-numeric stock values), the function will handle the exception and notify the user.

## Display\_products():

```
Display_module.py x Edit_module.py Main_file.py Menu_module.py products.txt Search_module.py Delete_module.py
1 def display_products():
2     if not os.path.exists(FILE_NAME) or os.path.getsize(FILE_NAME) == 0:
3         print("No records found.")
4         return
5
6     print("\nProduct Records:")
7     with open(FILE_NAME, "r") as file:
8         for line in file:
9             try:
10                 product_name, total_stock, number_of_order, shortage = line.strip().split(", ")
11                 print(f"Product: {product_name}, Total Stock: {total_stock}, Orders: {number_of_order}, Shortage: {shortage}")
12             except ValueError:
13                 print(f"Malformed line skipped: {line.strip()}")
```

This function reads all products stored in the inventory file and displays them to the user. It ensures that the file is not empty or malformed before attempting to display any records. If the file is empty, it notifies the user that no records are available.

## Search\_product():

```
Main_file.py products.txt Search_module.py x
1 def search_product():
2     product_name = input("Enter Product Name to search: ").strip()
3
4     if not os.path.exists(FILE_NAME) or os.path.getsize(FILE_NAME) == 0:
5         print("No records found.")
6         return
7
8     with open(FILE_NAME, "r") as file:
9         for line in file:
10             record = line.strip().split(", ")
11             if record[0] == product_name:
12                 print(f"Record Found: Product: {record[0]}, Total Stock: {record[1]}, Orders: {record[2]}, Shortage: {record[3]}")
13                 return
14     print("Record not found.")
```

The function prompts the user to enter the product name to search for and displays the product details if it is found. It reads the file line by line, looking for a matching product name. If no match is found, it informs the user that the record does not exist.

## Edit\_product():

This function allows the user to modify the details of an existing product. It displays the current information and allows the user to enter new values (or leave the fields blank to retain the existing ones). The modified product details are then written back to the file, ensuring that the changes are reflected in the inventory.

```
Edit_module.py × Main_file.py Menu_module.py products.txt Search_module.py
1 def edit_product():
2
3
4
5
6
7
8
9
10     with open(FILE_NAME, "r") as file:
11         for line in file:
12             record = line.strip().split(", ")
13             if record[0] == product_name:
14                 found = True
15                 print("Enter new details (leave blank to keep current value):")
16                 new_total_stock = input(f"New Total Stock (current: {record[1]}): ").strip() or record[1]
17                 new_number_of_order = input(f"New Number of Orders (current: {record[2]}): ").strip() or record[2]
18                 new_shortage = input(f"New Shortage (current: {record[3]}): ").strip() or record[3]
19                 updated_records.append(f"{product_name}, {new_total_stock}, {new_number_of_order}, {new_shortage}\n")
20             else:
21                 updated_records.append(line)
22
23     if found:
24         with open(FILE_NAME, "w") as file:
25             file.writelines(updated_records)
26         print("Product updated successfully!")
27     else:
28         print("Product not found.")
```

## Delete\_product():

```
Edit_module.py Main_file.py Menu_module.py products.txt Search_module.py
1 def delete_product():
2     product_name = input("Enter Product Name to delete: ").strip()
3
4     if not os.path.exists(FILE_NAME) or os.path.getsize(FILE_NAME) == 0:
5         print("No records found.")
6         return
7
8     updated_records = []
9     found = False
10
11     with open(FILE_NAME, "r") as file:
12         for line in file:
13             record = line.strip().split(", ")
14             if record[0] == product_name:
15                 found = True
16                 print(f"Product '{product_name}' deleted.")
17             else:
18                 updated_records.append(line)
19
20     if found:
21         with open(FILE_NAME, "w") as file:
22             file.writelines(updated_records)
23     else:
24         print("Product not found.")
```

This function enables the user to delete a product from the inventory. It searches for the product by name and, if found, removes the product record from the file. The updated inventory is then saved.

### Main\_menu():

```
Main_file.py  Menu_module.py ×  products.txt  Search_module.py

1  def main_menu(): 1 usage
2      while True:
3          print("\nElectronic Product Inventory Management:")
4          print("1. Add Product")
5          print("2. Display All Products")
6          print("3. Search Product")
7          print("4. Edit Product")
8          print("5. Delete Product")
9          print("6. Exit")
10
11         choice = input("Enter your choice: ").strip()
12
13         if choice == "1":
14             add_product()
15         elif choice == "2":
16             display_products()
17         elif choice == "3":
18             search_product()
19         elif choice == "4":
20             edit_product()
21         elif choice == "5":
22             delete_product()
23         elif choice == "6":
24             print("Exiting the program. Goodbye!")
25             break
26         else:
27             print("Invalid choice. Please try again.")
28
29  if __name__ == "__main__":
30      main_menu()
```

This function serves as the interface for the user to interact with the system. It provides a menu with different options for managing the inventory (add, display, search, edit, delete, or exit). The user selects the desired action, and the appropriate function is executed

## **4. Features of the Python-based Inventory System**

The Python-based inventory management system, as demonstrated in the uploaded code, provides a range of features that ensure efficient tracking, management, and automation of stock levels. Below are the key features and how they align with the system's capabilities.

### **4.1 Stock Tracking: Automatically Updates Inventory Levels**

One of the primary functions of the inventory system is to track stock levels in real-time. As demonstrated in the Python code, product information (including stock levels, orders, and shortages) is updated in a structured file, ensuring that stock levels are continuously monitored. The system automatically updates inventory data whenever products are added or edited, and this data is stored in a consistent format for easy retrieval. In the future, integration with databases or visualization tools can further enhance the stock tracking process. The system ensures that decision-makers are always aware of current stock levels, enabling timely decisions related to restocking or discontinuing products.

### **4.2 Order Management: Generates Purchase Orders When Stock is Low**

While the uploaded code does not include automatic purchase order generation, the system can be extended to do so. With Python, it is possible to create rules or triggers that automatically generate purchase orders when stock levels fall below a predefined threshold. The system could generate a report of low-stock products and send purchase orders to suppliers directly or trigger notifications to inventory managers for action. This automation of order management would help Walton avoid stockouts and ensure that inventory levels remain optimal, reducing the manual work involved in ordering products and improving the efficiency of the restocking process.

### **4.3 Sales Integration: Updates Inventory Based on Sales Data**

Integrating the inventory management system with sales data would allow the system to automatically update stock levels based on sales transactions. Whenever a sale is made, the system can deduct the sold quantity from the total stock and update inventory records accordingly. With Python, the system can be easily connected to a point-of-sale (POS) system or an e-commerce platform's API to receive real-time sales data. This ensures that inventory levels are always accurate and up to date, preventing overstocking or understocking issues. Additionally, it enables better forecasting of future inventory needs based on historical sales trends.

#### **4.4 Supplier Management: Keeps Track of Suppliers and Restocking Schedules**

The uploaded code could be extended to include supplier management functionality. By tracking information about suppliers, including contact details, delivery schedules, and lead times, the inventory management system can automatically update restocking schedules and notify inventory managers when it's time to reorder products. Python can easily be used to integrate supplier data into the inventory system, enabling Walton to manage supplier relationships and ensure timely replenishment of products. For example, automated emails or notifications can be triggered when stock reaches a reorder point, and the system can keep track of the expected delivery dates from suppliers.

#### **4.5 Reporting & Analytics: Generates Reports on Stock Movement, Demand Forecasting, and Sales Trends**

Reporting and analytics are key components of a robust inventory management system. With Python, tools like matplotlib, seaborn, and pandas can be integrated to generate insightful reports on stock movement, demand forecasting, and sales trends. For example, Walton can use the system to generate reports that visualize sales trends over time, identify slow-moving products, and forecast future demand based on historical data. Moreover, the system can be enhanced to generate automatic reports on stock movement, indicating when products are selling faster or slower than expected, enabling better decisions around restocking. This level of visibility helps Walton optimize inventory levels and make more accurate predictions regarding future product needs.

### **5. Challenges of Inventory Management System**

Here are five key challenges associated with implementing the Python-based automated inventory management system for Walton:

1. **System Integration:** Integrating the Python-based inventory system with Walton's existing platforms (e.g., sales systems, supplier management tools) is a challenge. Seamlessly connecting these systems for real-time data synchronization requires compatibility and may demand additional customization.

2. **Data Accuracy and Real-Time Updates:** Ensuring that inventory data remains accurate and up-to-date in real-time is critical. Any delays or discrepancies in updating stock levels across multiple locations can lead to stockouts, overstocking, or operational inefficiencies.

3. **User Training and Adoption:** Transitioning from manual processes to an automated system requires training employees on how to use the new system effectively. There may also be resistance to change from staff used to traditional methods, requiring time and resources for proper training.

4. **Security and Data Privacy:** Protecting sensitive business information, such as inventory levels and supplier details, is essential. Implementing security measures to prevent unauthorized access, data breaches, and ensuring safe data transfers poses an ongoing challenge.

## **6. Conclusion**

The automation of Walton's inventory management system using Python has significantly enhanced the company's operational efficiency. By replacing manual processes with an automated system, Walton has minimized human error, improved stock accuracy, and gained real-time insights into inventory levels. This transformation ensures seamless stock management, preventing both stockouts and overstocking, which ultimately enhances customer satisfaction and reduces operational costs.

A key advantage of the Python-based system is its ability to continuously update and track inventory. Routine tasks such as stock replenishment are automated, reducing human oversight and ensuring timely product reordering. Future integration with supplier data will further streamline coordination between Walton and its vendors, optimizing inventory flow. Scalability is another crucial feature of the system. As Walton grows, the system can easily adapt to handle more extensive inventory needs. By integrating sales data and supplier management, the system automates product tracking and reordering, freeing staff from manual checks.

Additionally, the incorporation of reporting and analytics tools allows Walton to analyze sales trends, stock movement, and demand forecasting. This data-driven approach supports strategic decision-making, ensuring agility in a competitive market. Walton's investment in automation underscores its commitment to operational excellence and long-term success in the wholesale industry.