Medical Inventory Management using Salesforce

Aim:

To develop a cloud-based Medical Inventory Management System using Salesforce that efficiently manages suppliers, purchase orders, and stock, while automating data updates, reporting, and dashboards for better decision-making.

Objectives:

1. To understand and implement Salesforce CRM features for inventory management.
2. To create and manage custom objects such as Suppliers, Purchase Orders, and Order Items.
3. To establish relationships between objects for linked data flow.
4. To implement Apex Triggers for automatic total calculations.
5. To generate reports and dashboards for real-time data visualization.
6. To enhance business process efficiency in the medical inventory domain.

Overview:

The Smart Intrnz – Medical Inventory Management project focuses on building a Salesforce-based application to streamline the medical inventory process.
The system allows users to manage supplier information, create purchase orders, track order items, and monitor total purchase costs using automation and dashboards.

Key Salesforce Components Used:

Custom Objects:

1. Supplier__c
2. Purchase_Order__c
3. Order_Item__c

Relationships:

- Lookup relationship between Supplier and Purchase Order.
- Master-Detail relationship between Purchase Order and Order Item.

Automation:

- Apex Trigger to calculate total order amount automatically.

Reports:

- Summary Report: Purchase Orders based on Suppliers

Dashboards:

- Visual chart showing total order cost grouped by supplier.

Procedure:

Step 1: Create Custom Objects

Navigate to Setup → Object Manager → New Object

Create the following:

Supplier (fields: Supplier Name, Supplier Email, Supplier Contact)

Purchase Order (fields: Purchase Order ID, Total Order Cost, Supplier ID)

Order Item (fields: Item Name, Quantity, Amount, Purchase Order ID)

Step 2: Establish Relationships

Create a Lookup Relationship from Purchase Order → Supplier.

Create a Master-Detail Relationship from Order Item → Purchase Order.

Step 3: Create Apex Trigger

In Developer Console:

```
trigger CalculateTotalAmountTrigger on Order_Item__c (after insert,
after update, after delete, after undelete) {
   CalculateTotalAmountHandler.calculateTotal(Trigger.new,
Trigger.old, Trigger.isInsert, Trigger.isUpdate, Trigger.isDelete,
Trigger.isUndelete);
}
```

Create handler class:

```
public class CalculateTotalAmountHandler {
   public static void calculateTotal(List<Order_Item__c> newItems,
List<Order_Item__c> oldItems, Boolean isInsert, Boolean isUpdate,
Boolean isDelete, Boolean isUndelete) {
      Set<Id> parentIds = new Set<Id>();
      if (isInsert || isUpdate || isUndelete)
         for (Order_Item__c ordItem : newItems)
            parentIds.add(ordItem.Purchase_Order_Id__c);
      if (isUpdate || isDelete)
         for (Order_Item__c ordItem : oldItems)
            parentIds.add(ordItem.Purchase_Order_Id__c);

      if (!parentIds.isEmpty()) {
         List<AggregateResult> aggrList = [
            SELECT Purchase_Order_Id__c, SUM(Amount__c)
totalAmount
            FROM Order_Item__c
            WHERE Purchase_Order_Id__c IN :parentIds
            GROUP BY Purchase_Order_Id__c
         ];
         Map<Id, Decimal> purchaseToUpdateMap = new Map<Id,
Decimal>();
```

```
            for (AggregateResult aggr : aggrList) {
                purchaseToUpdateMap.put((Id)aggr.get('Purchase_Order_I
d__c'), (Decimal)aggr.get('totalAmount'));
            }
            List<Purchase_Order__c> purchaseToUpdate = new
List<Purchase_Order__c>();
            for (Id purchaseOrderId : purchaseToUpdateMap.keySet())
                purchaseToUpdate.add(new Purchase_Order__c(Id =
purchaseOrderId, Total_Order_Cost__c =
purchaseToUpdateMap.get(purchaseOrderId)));
            update purchaseToUpdate;
        }
    }
}
```

Step 4: Create Report

App Launcher → Medical Inventory Management App

Go to Reports → New Report

Select report type Purchase Orders → Start Report

Set filters and group rows by:

Supplier ID

Purchase Order ID

Add columns:

Order Count

Total Order Cost

Save as: Purchase Orders based on Suppliers

Step 5: Create Dashboard

Go to Dashboards → New Dashboard

Name: Purchase Orders Dashboard

Add a Chart Component

Source: Purchase Orders based on Suppliers Report

Display: Donut Chart

Show total order cost grouped by supplier

Format values as ₹ Currency

Save and Run Dashboard.

Screenshot Descriptions

(Add your actual screenshots here — below are sample captions)

Screenshot 1: Supplier Object creation screen.

Screenshot 2: Relationship setup between Purchase Order and Order Item.

Screenshot 3: Apex Trigger in Developer Console.

Screenshot 4: Summary Report view.

Screenshot 5: Final Dashboard (Donut chart with ₹ in center).

Output:

The system successfully displays a summary of purchase orders grouped by suppliers, showing the total order cost in ₹ currency. The dashboard dynamically updates when order items are inserted, updated, or deleted.

Result:

The Smart Intrnz Medical Inventory Management project was successfully implemented using Salesforce. All modules, including data relationships, triggers, reports, and dashboards, functioned as expected.

Conclusion:

This project demonstrates the use of Salesforce CRM for building a real-time cloud-based medical inventory system. It automates total cost calculations and provides insightful visual reports, enhancing data accuracy and operational efficiency for healthcare inventory management.