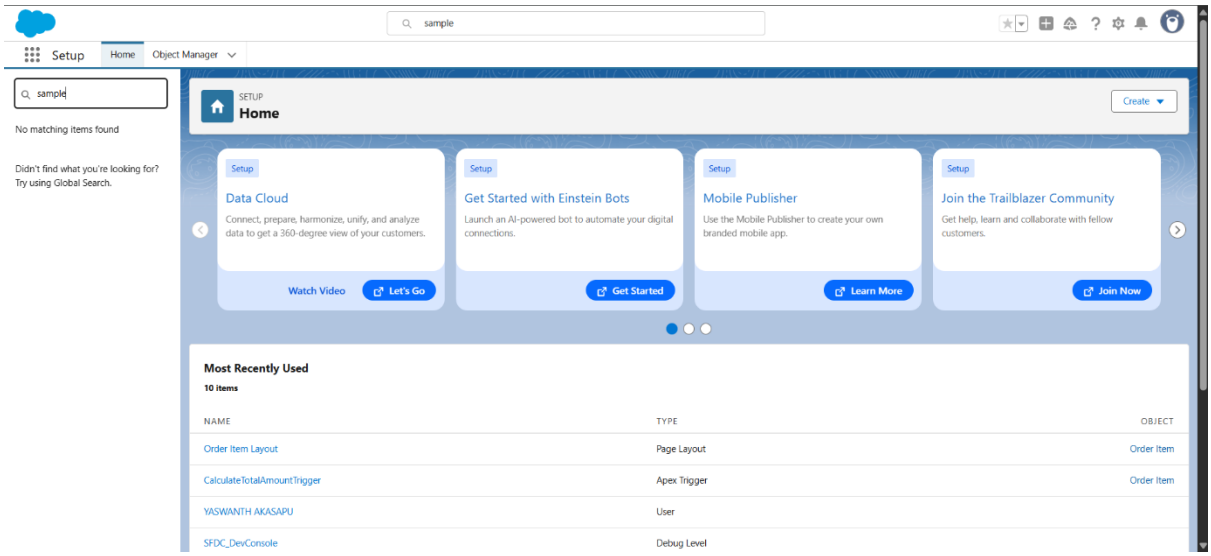
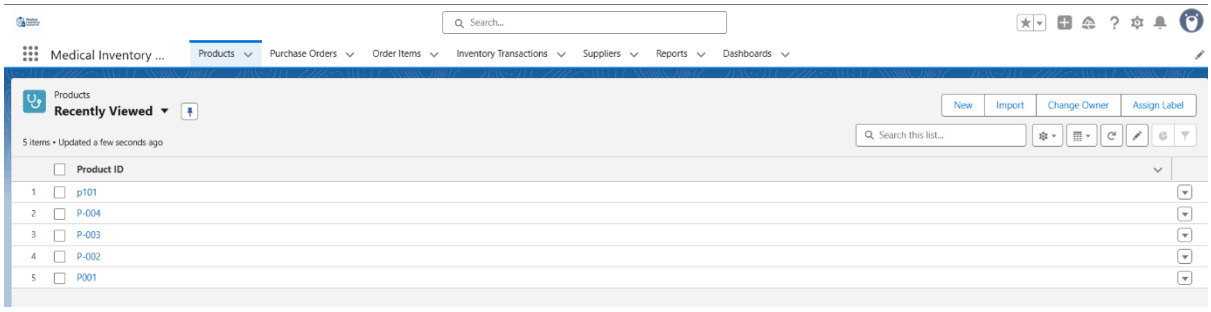
1. Login & App Launcher :

1: Salesforce homepage after login

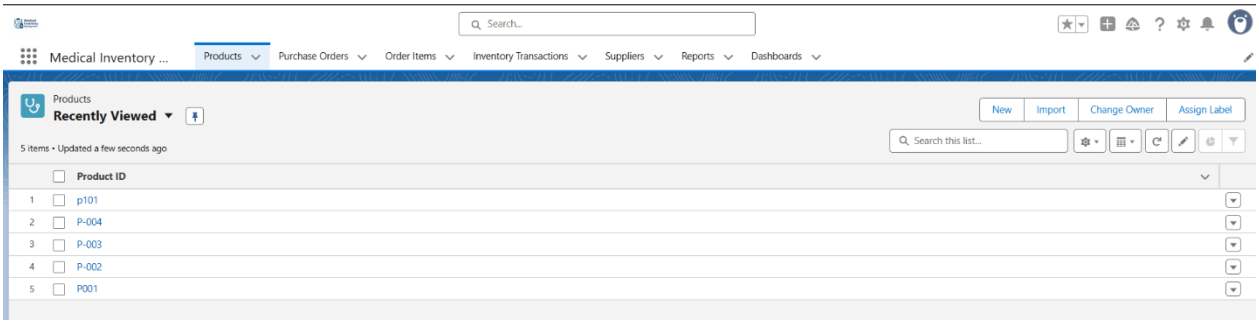


2: App Launcher > Medical Inventory Management app selected.

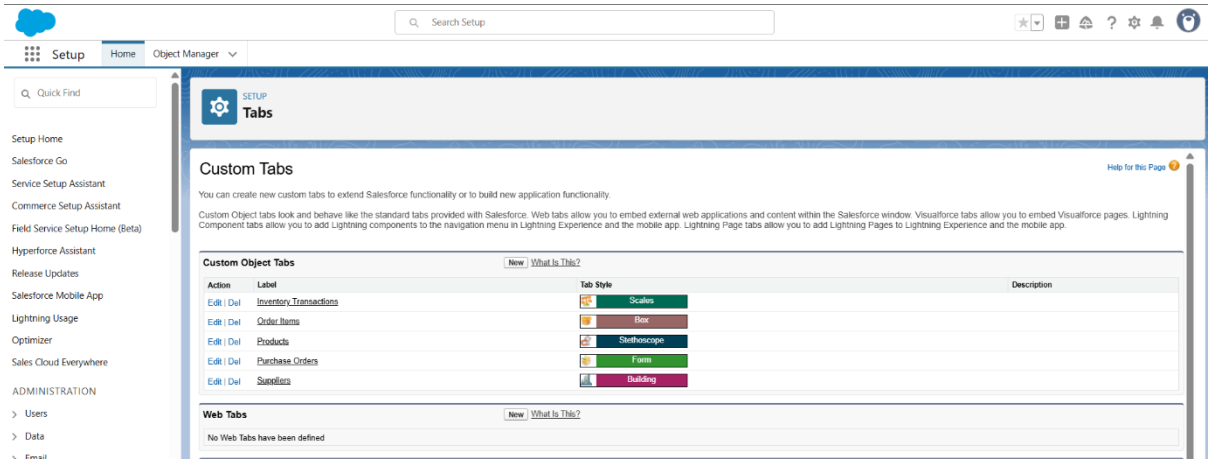


2. Key Tabs and Navigation

1: Home screen of the Medical Inventory Management app



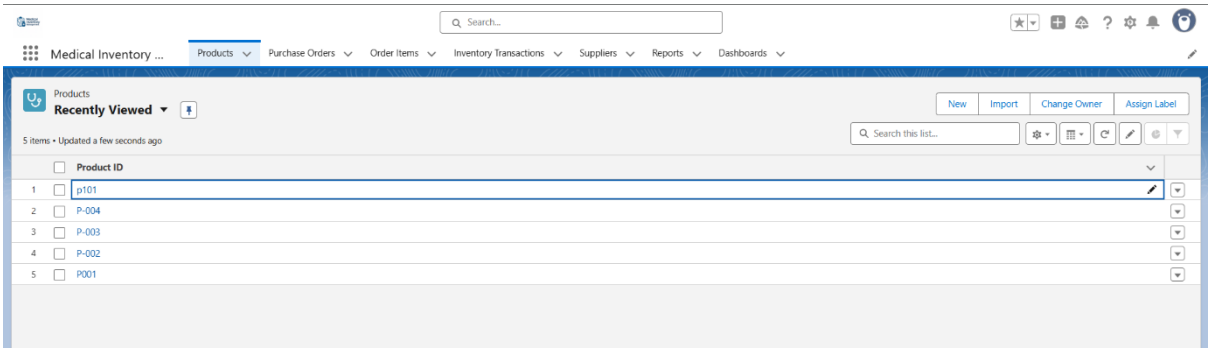
2: Tabs for Products, Suppliers, Purchase Orders, Order Items, Inventory Transactions



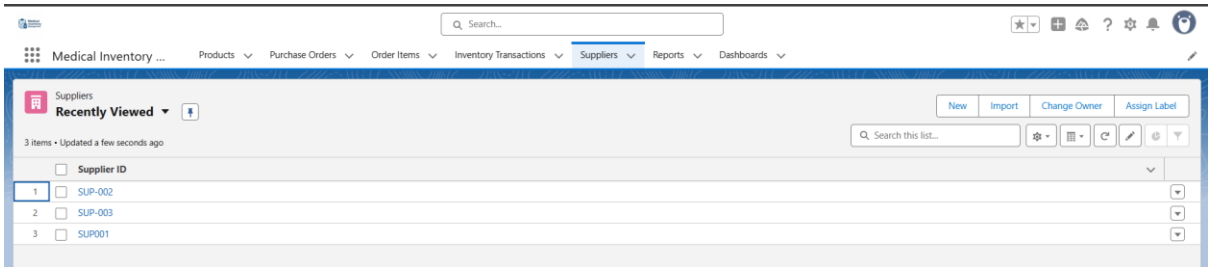
3. Objects & Records

Each object's list view with some records:

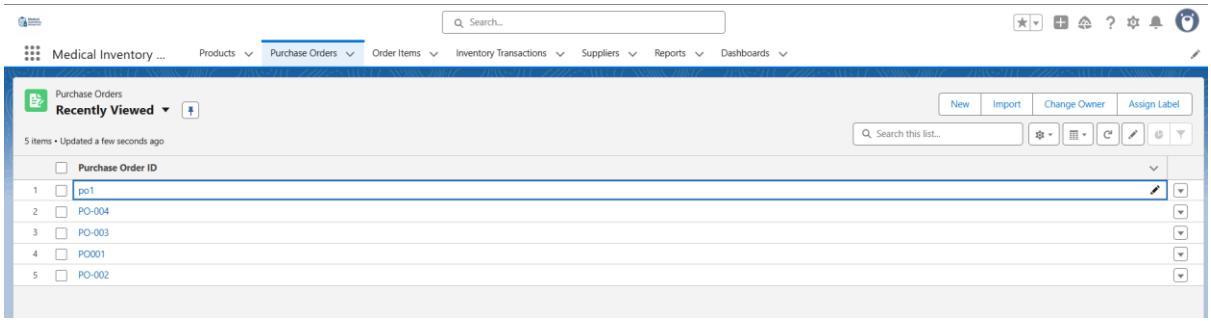
1: Products list showing 3+ products with Unit Price, Stock Levels.



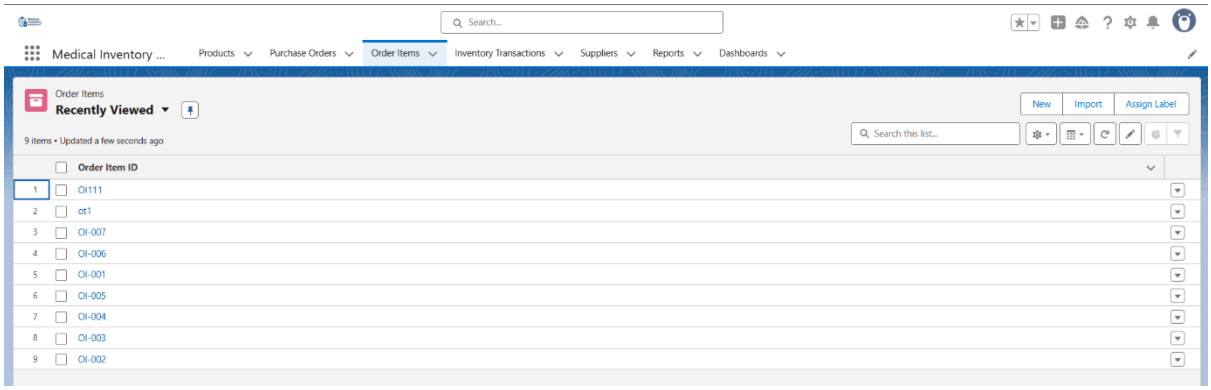
2. Suppliers list with Supplier Name, Address, Contact Person



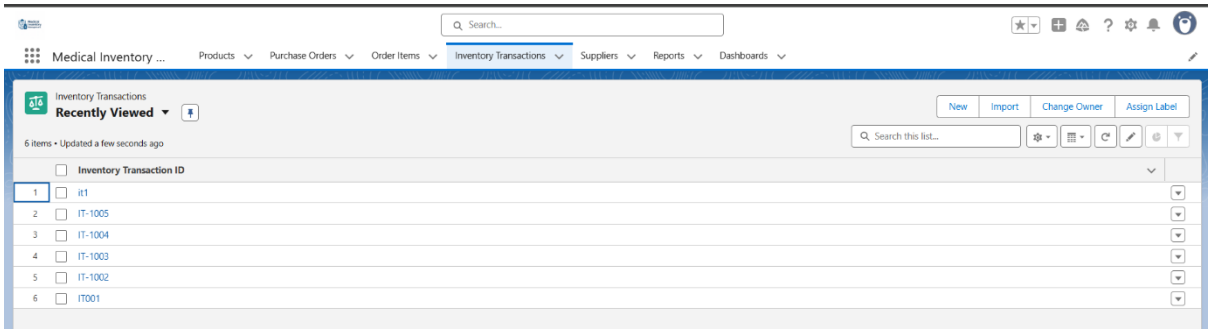
3. Purchase Orders list showing PO ID, Supplier, Total Cost



4. Order Items list showing Product, Quantity, Amount

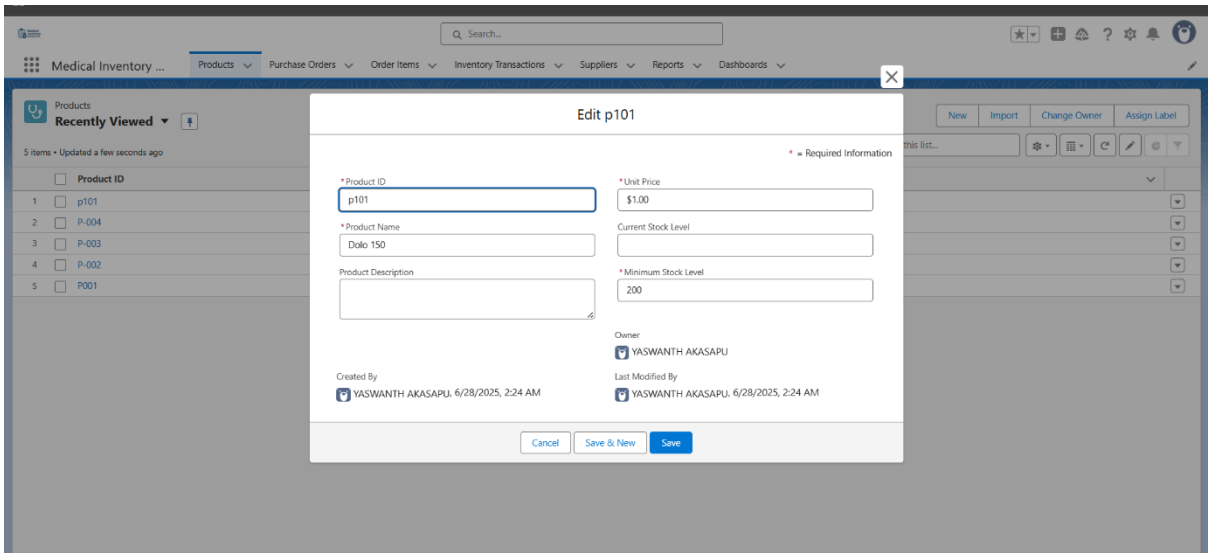


5. Inventory Transactions list with Type (Issue, Receipt), Date

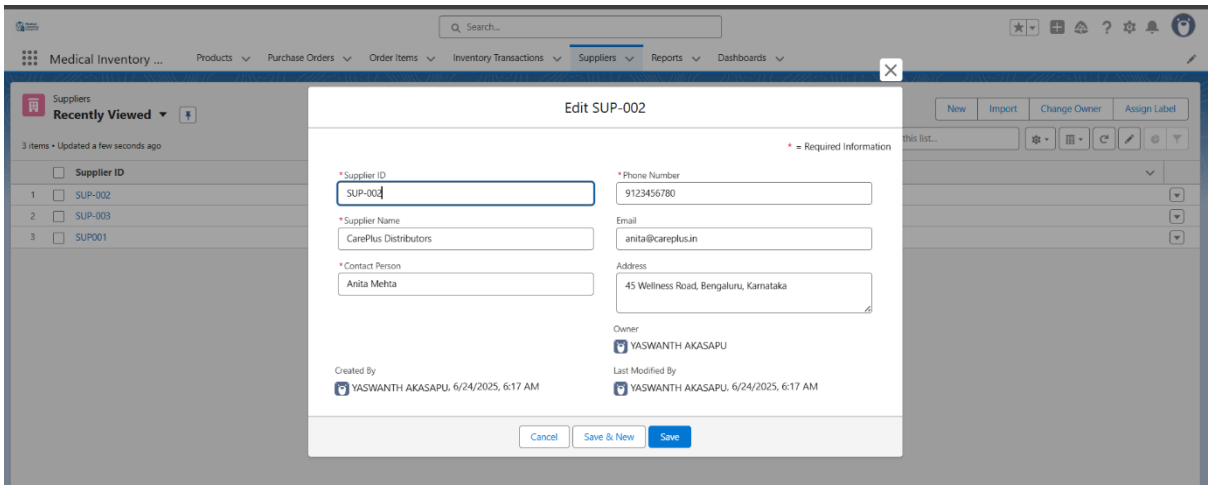


4. Detailed Record View (Record Page Layout)

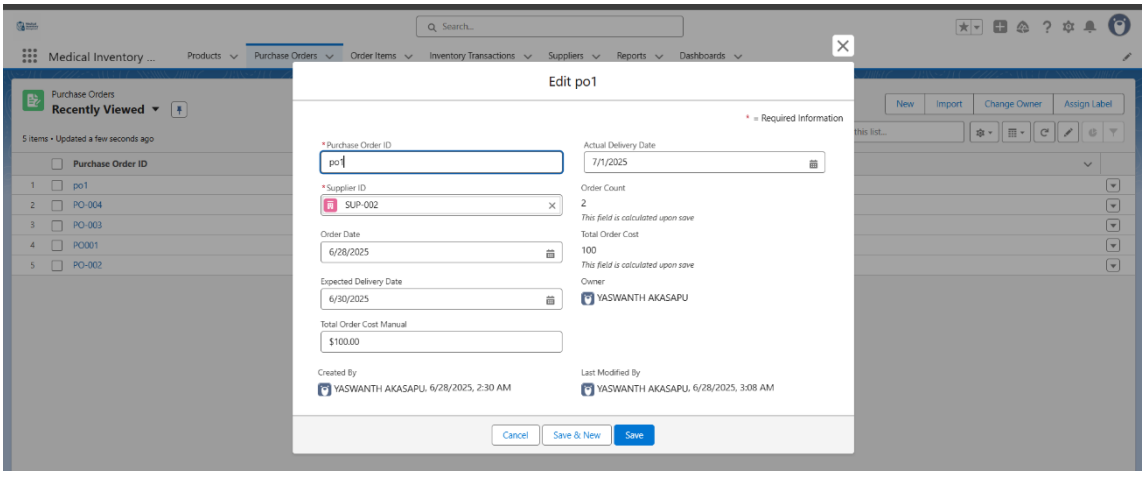
1. One Product record – show all fields like Product Name, Stock Level, Min Stock, Unit Price



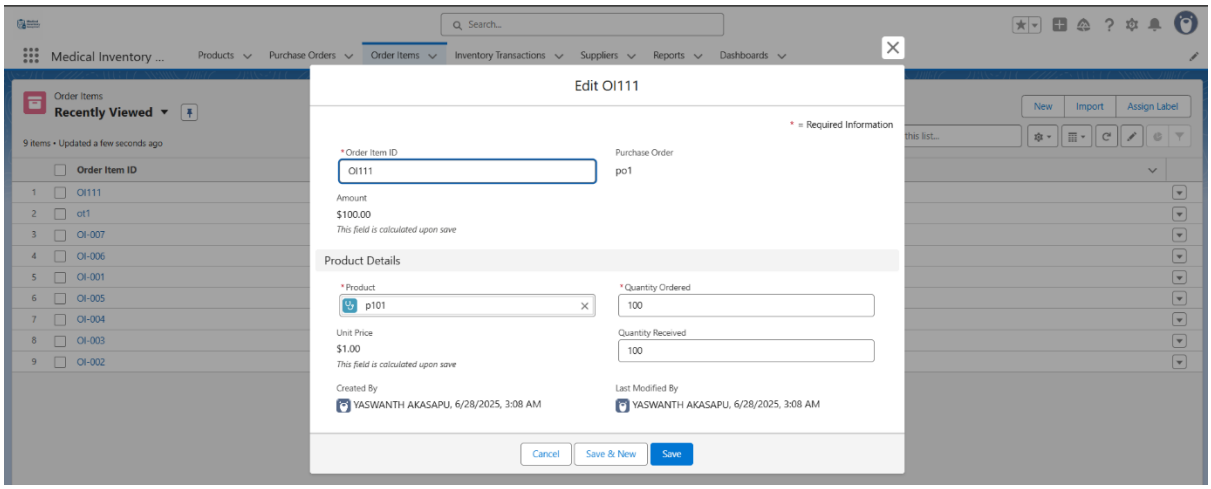
2. One Supplier record – show all fields like Name, Contact, Email



3.One Purchase Order record – show Actual & Expected Delivery Dates, Total Cost

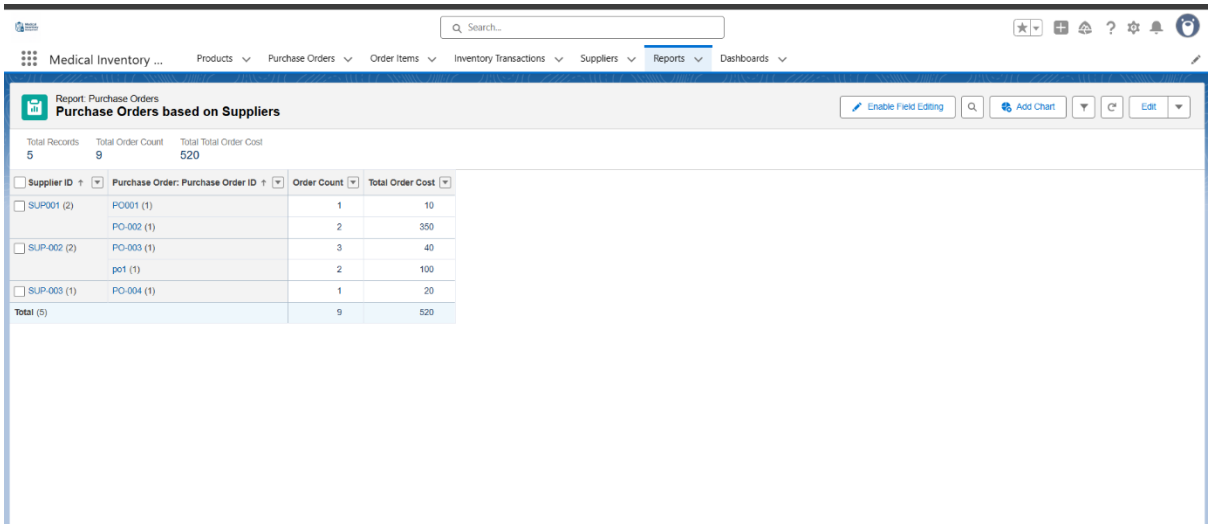


4. One Order Item – Product, Quantity Ordered/Received, Amount

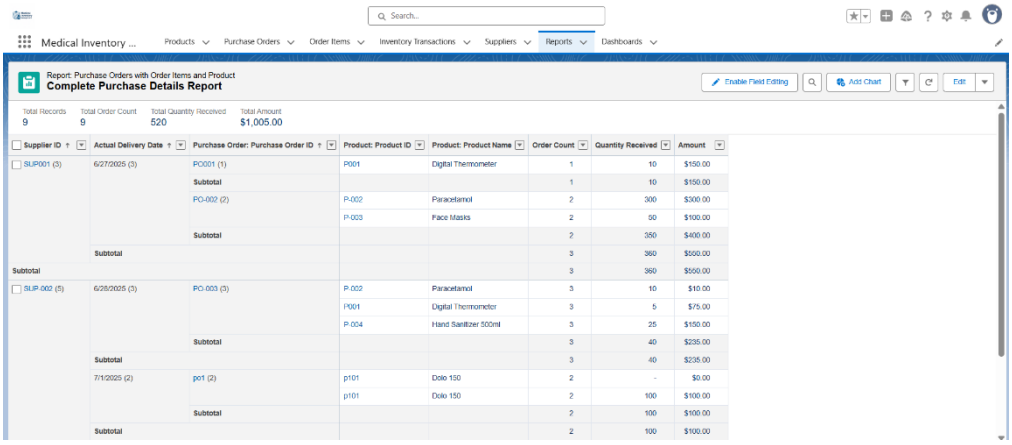


5. Reports

1. "Purchase Orders based on Suppliers" report – summary format

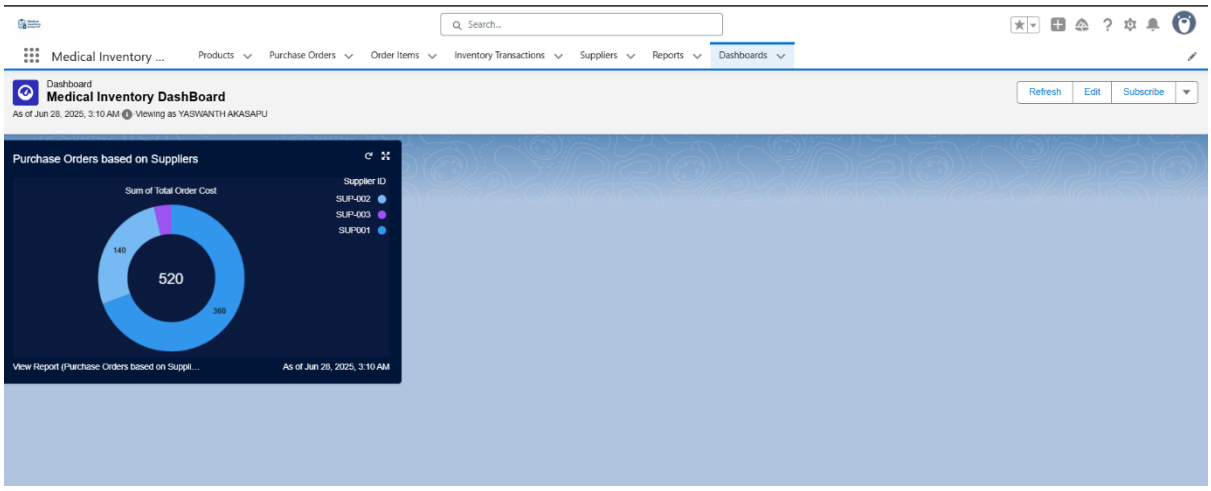


2. "Complete Purchase Details Report" – grouped by Supplier ID, showing Product, Quantity, Amount

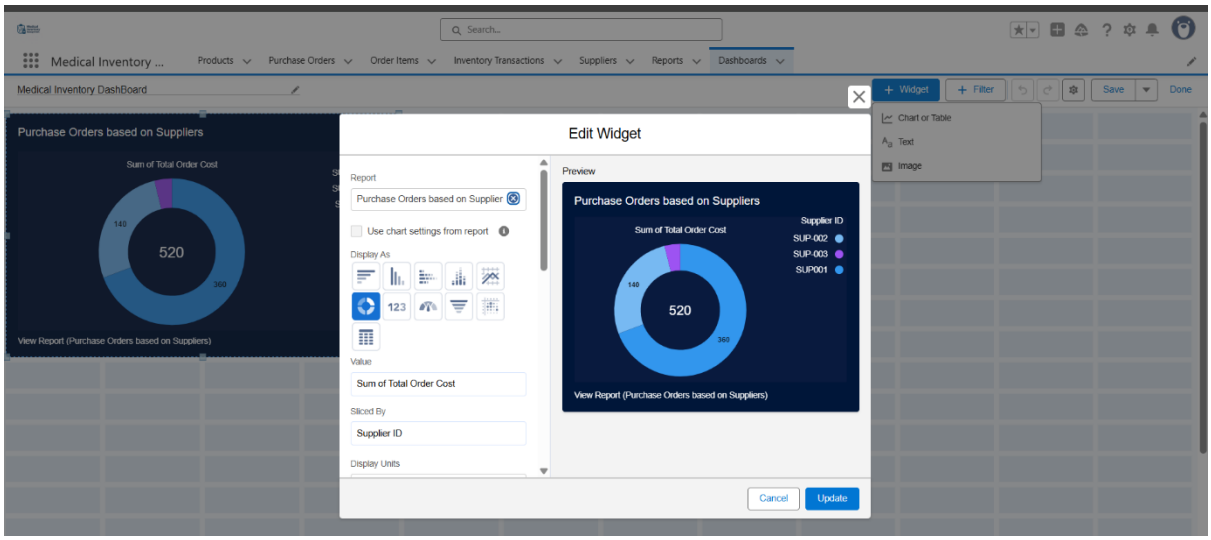


6. Dashboard

1. “Medical Inventory Dashboard” – show donut chart or bar graph.

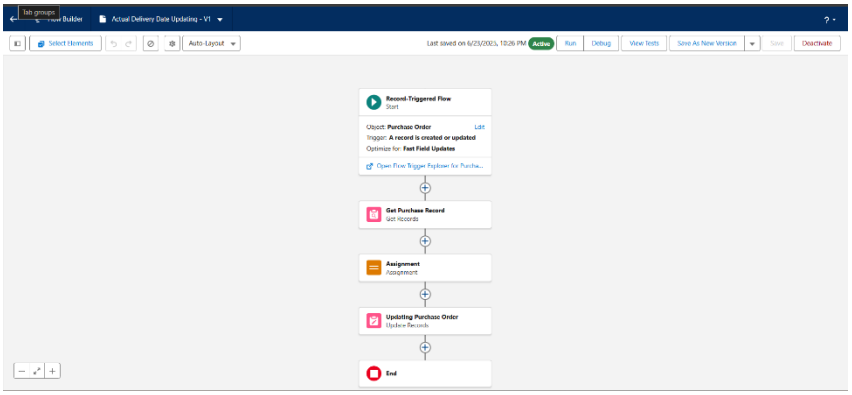


2. Click into widget to show underlying report



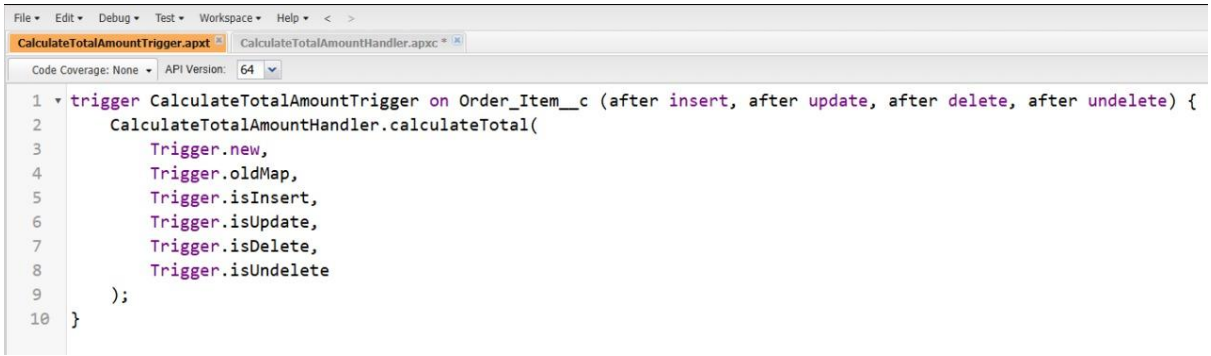
7. Automation/Flow & Trigger

1. Open Flow Builder – show Actual Delivery Date Flow



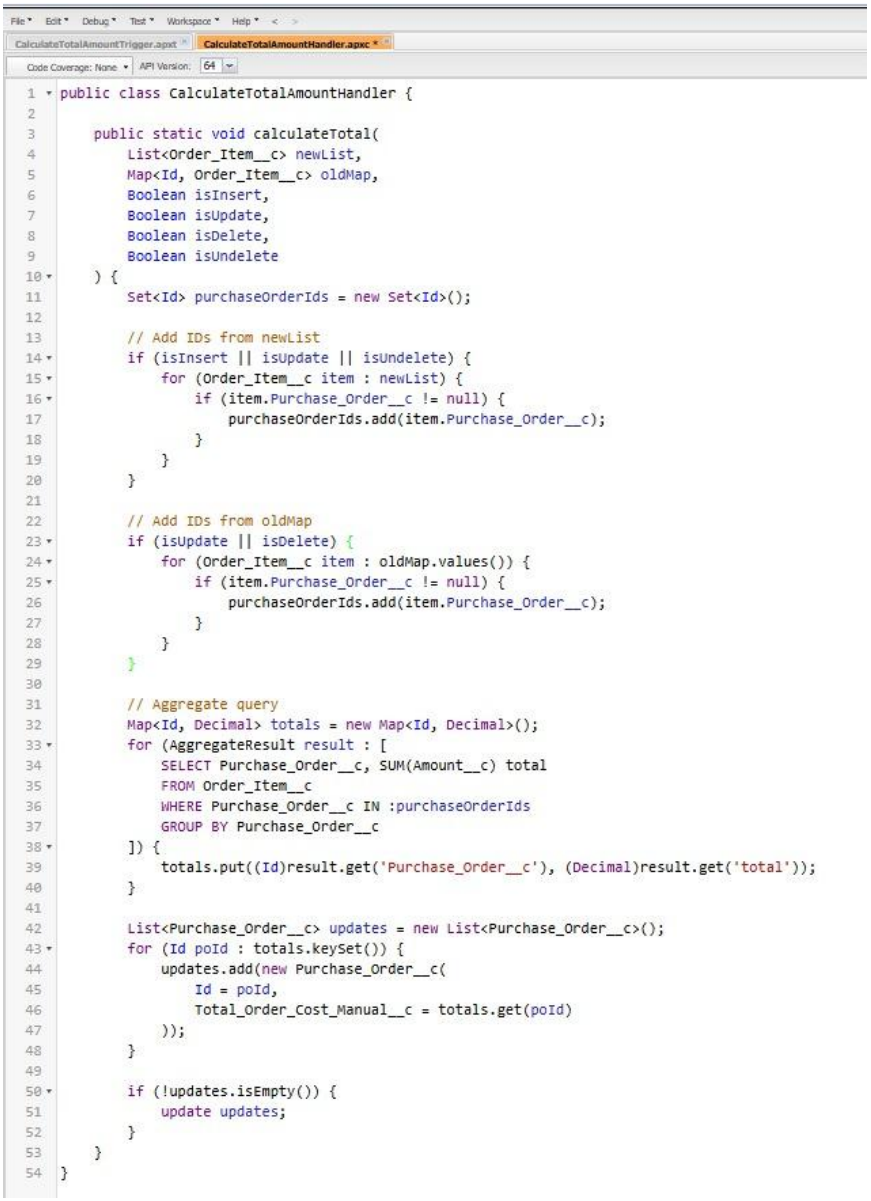
2. Developer Console > Apex Trigger and Apex Class (trigger + handler)

2.1 Apex trigger



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

2.2 Apex class



```
public class CalculateTotalAmountHandler {

    public static void calculateTotal(
        List<Order_Item__c> newList,
        Map<Id, Order_Item__c> oldMap,
        Boolean isInsert,
        Boolean isUpdate,
        Boolean isDelete,
        Boolean isUndelete
    ) {
        Set<Id> purchaseOrderIds = new Set<Id>();

        // Add IDs from newList
        if (isInsert || isUpdate || isUndelete) {
            for (Order_Item__c item : newList) {
                if (item.Purchase_Order__c != null) {
                    purchaseOrderIds.add(item.Purchase_Order__c);
                }
            }
        }

        // Add IDs from oldMap
        if (isUpdate || isDelete) {
            for (Order_Item__c item : oldMap.values()) {
                if (item.Purchase_Order__c != null) {
                    purchaseOrderIds.add(item.Purchase_Order__c);
                }
            }
        }

        // Aggregate query
        Map<Id, Decimal> totals = new Map<Id, Decimal>();
        for (AggregateResult result : [
            SELECT Purchase_Order__c, SUM(Amount__c) total
            FROM Order_Item__c
            WHERE Purchase_Order__c IN :purchaseOrderIds
            GROUP BY Purchase_Order__c
        ]) {
            totals.put((Id)result.get('Purchase_Order__c'), (Decimal)result.get('total'));
        }

        List<Purchase_Order__c> updates = new List<Purchase_Order__c>();
        for (Id poId : totals.keySet()) {
            updates.add(new Purchase_Order__c(
                Id = poId,
                Total_Order_Cost_Manual__c = totals.get(poId)
            ));
        }

        if (!updates.isEmpty()) {
            update updates;
        }
    }
}
```