PROJECT TITLE:
Medical Inventory Management

College Name: Nilgiri College of Arts and Science

College Code: Bruac

TEAM ID: NM2025TMID26670

TEAM MEMBERS:

Team Leader Name: SANTHRA.S

Email: santhrasatheesh2005@gmail.com

Team Member 1: SREEHARI.V

Email: sreehari.vengalil@gmail.com

Team Member 2: SIVAPRAKASH M

Email: praveendotz3@gmail.com

TeamMember 3: SENIN NIZAR P

Email: seninnizarpottayil@gmail.com

1.INTRODUCTION

1.1 Project Overview

The **Medical Inventory Management System (MIMS)** is a Salesforce-based application designed to manage medical supplies, track stock levels, manage orders, and automate alerts for expiry or low stock. The system helps healthcare providers maintain inventory accuracy, automate workflows, and make data-driven decisions

1.2 Purpose

- Efficient inventory tracking for medical supplies.
- Automation of alerts for low stock and item expiry.
- Minimize manual data entry errors.
- Generate analytics reports for decision-making.
- Streamline order processing and supplier management.

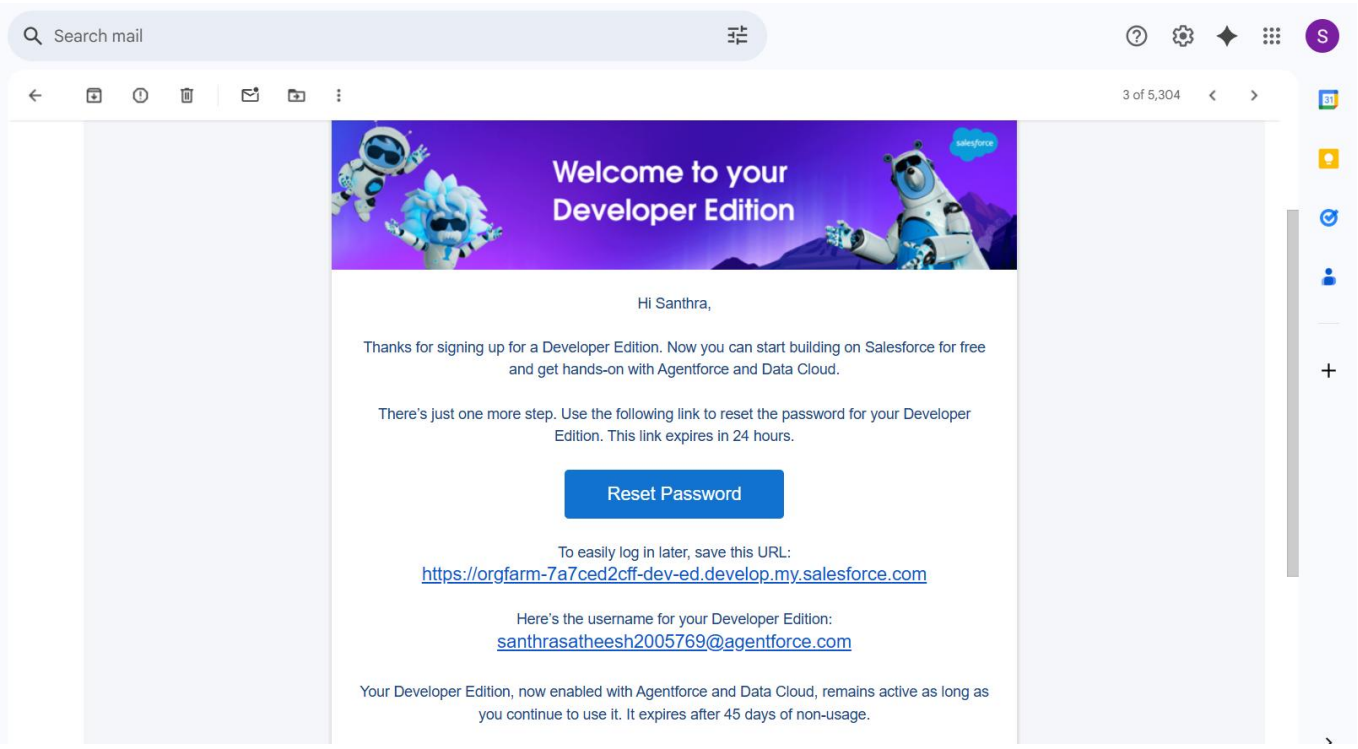
.

DEVELOPMENT PHASE

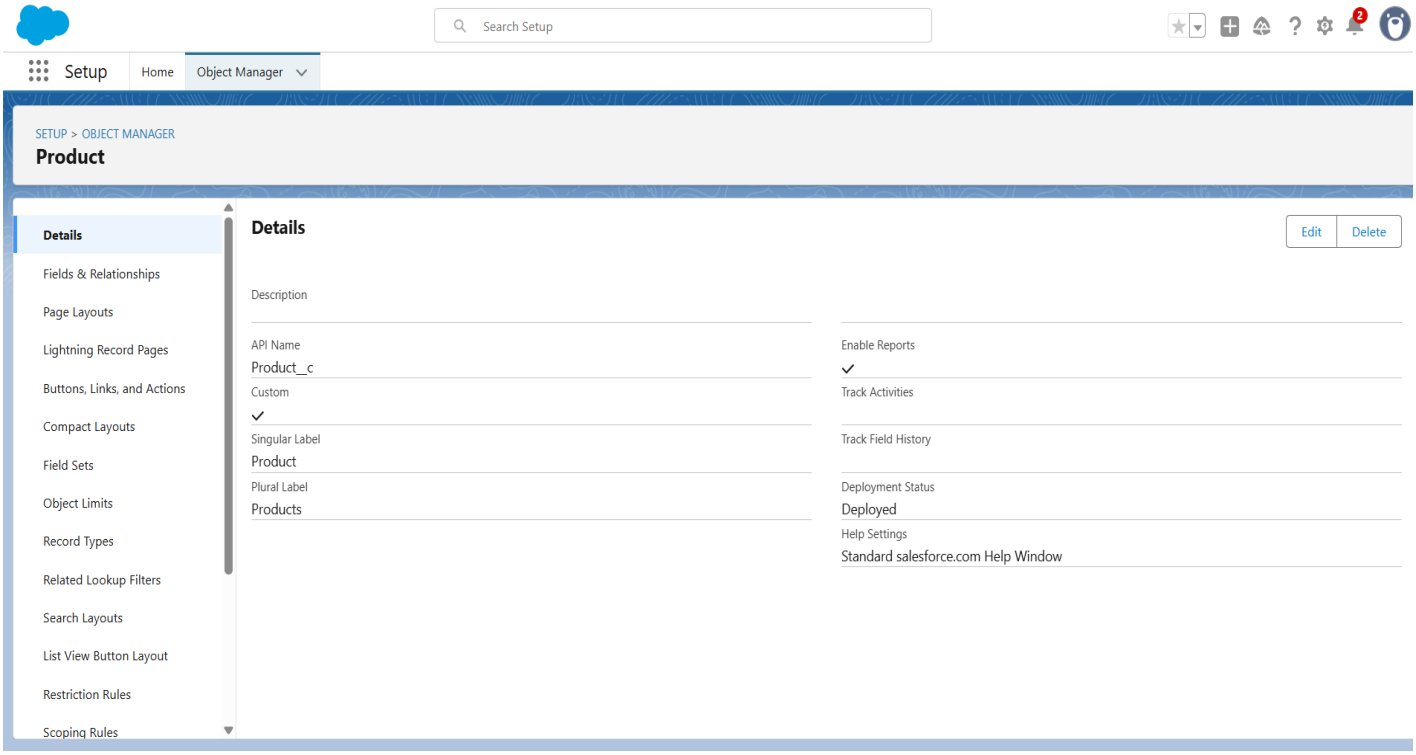
Creating Developer Account:

By using this URL - **https://www.salesforce.com/form/developer-signup/?d=pb**

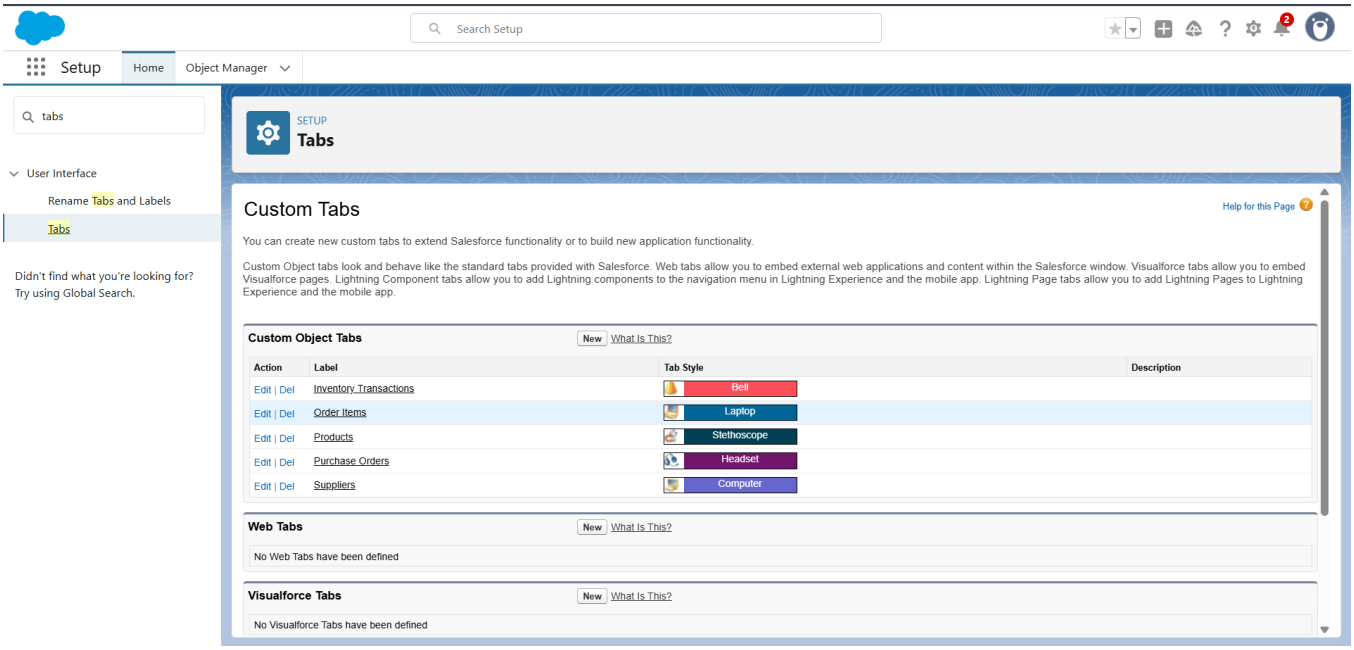
Created a developer account



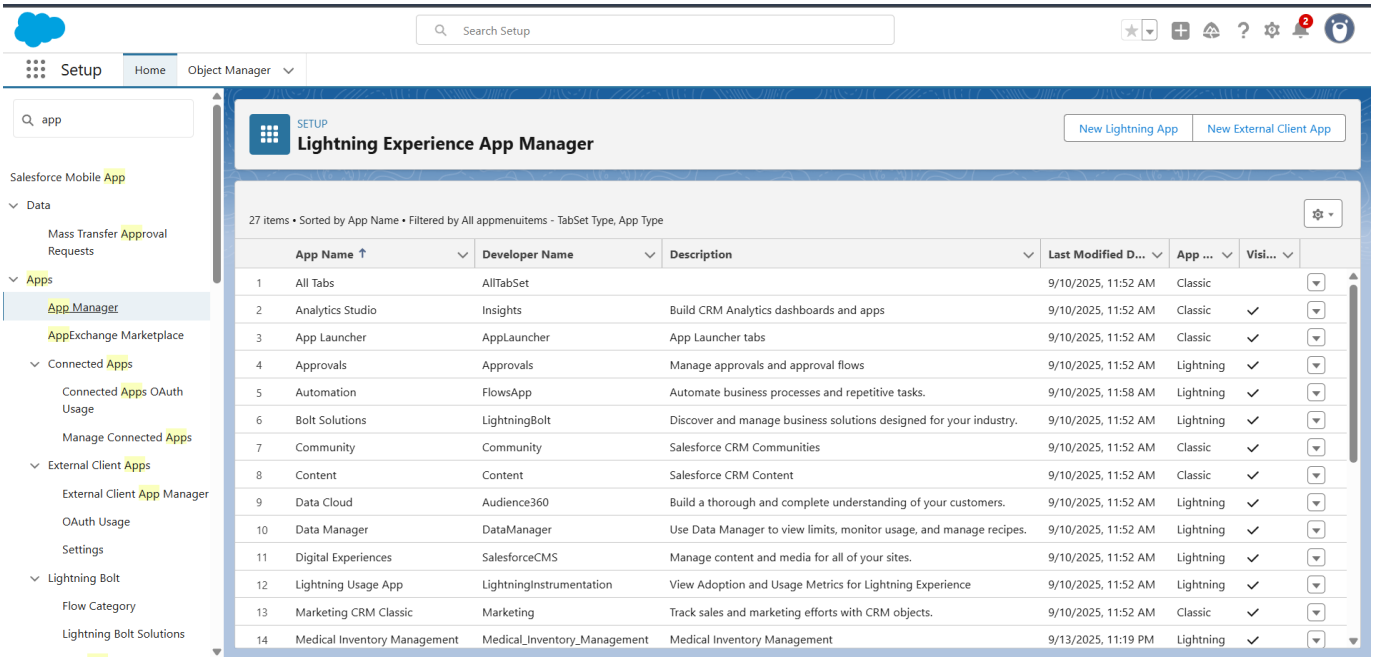
Created a product object



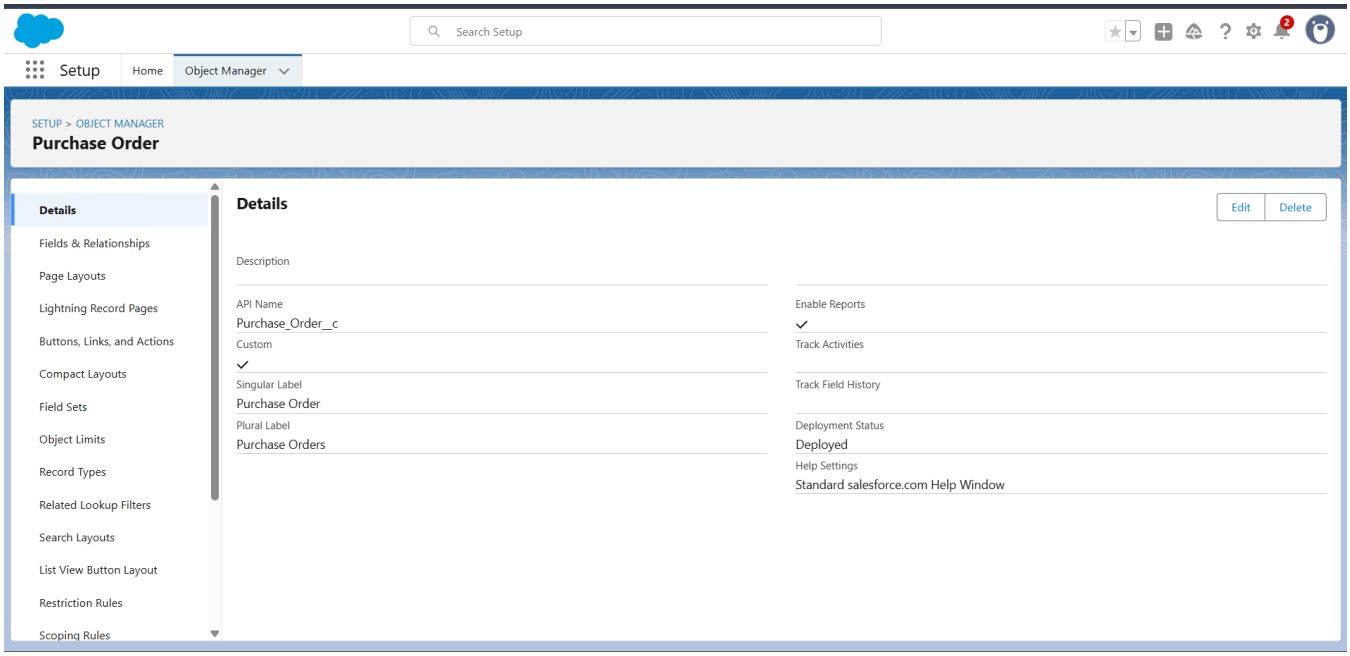
Created a tab for product object



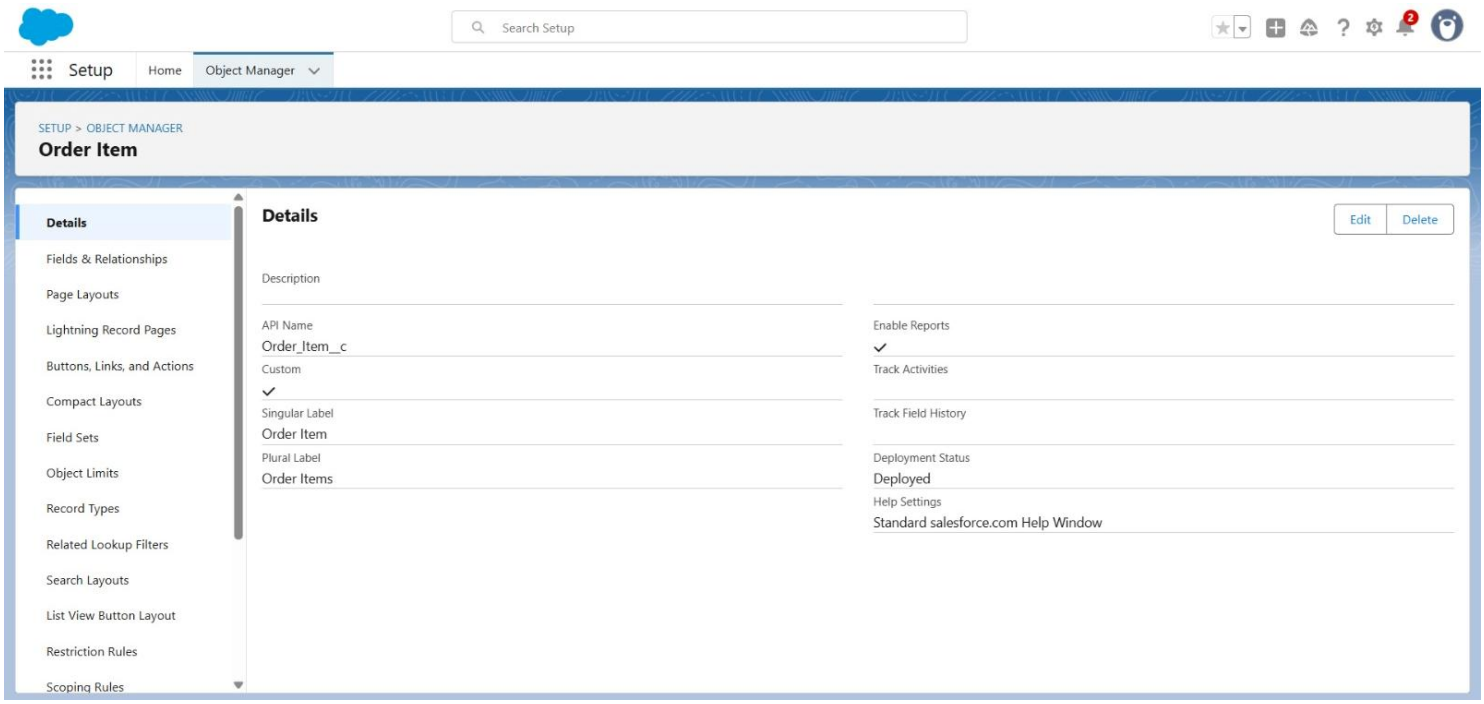
Created a lighting app for medical inventory management



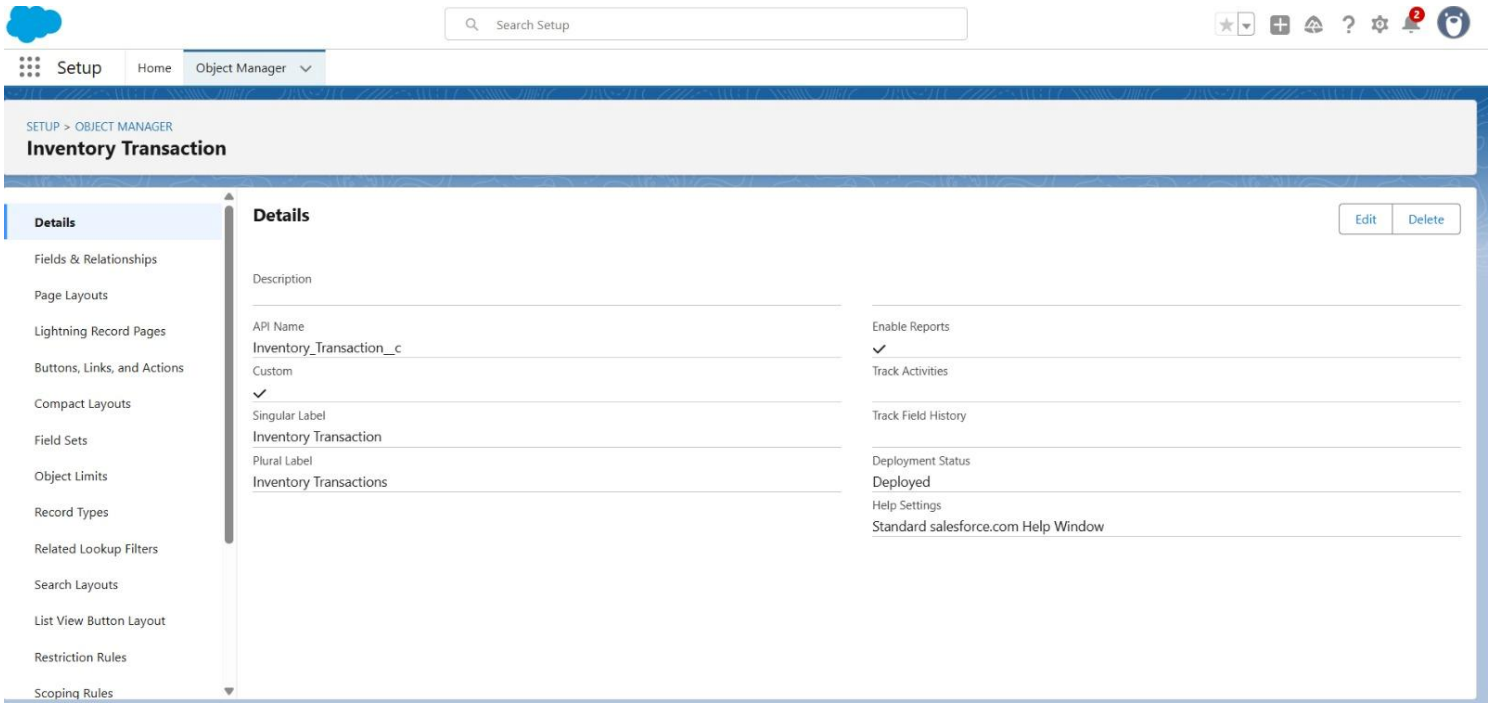
Creating a purchase order object



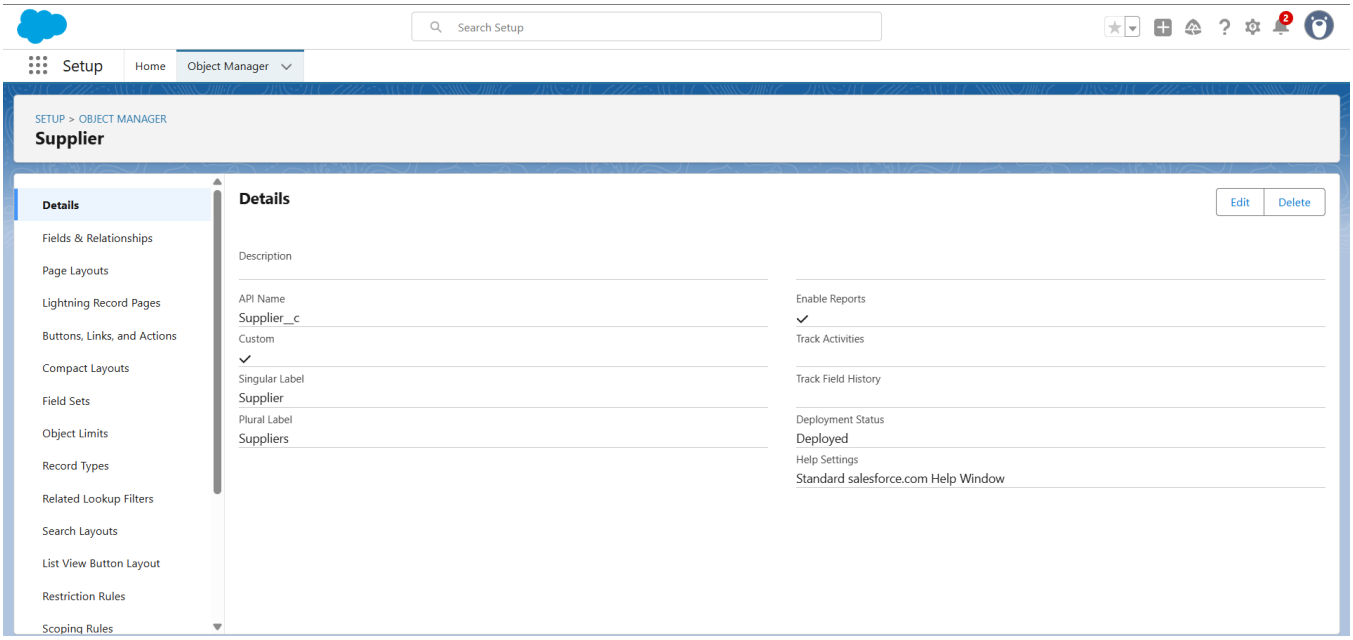
Order item object



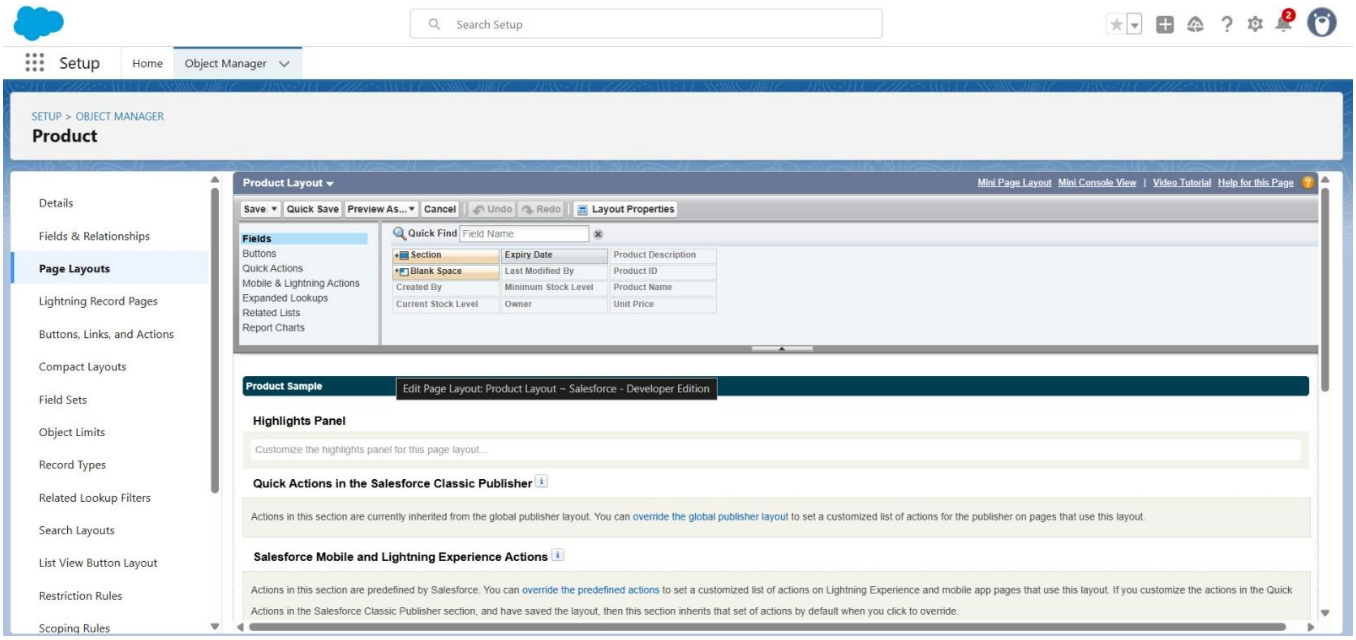
Inventory transaction object



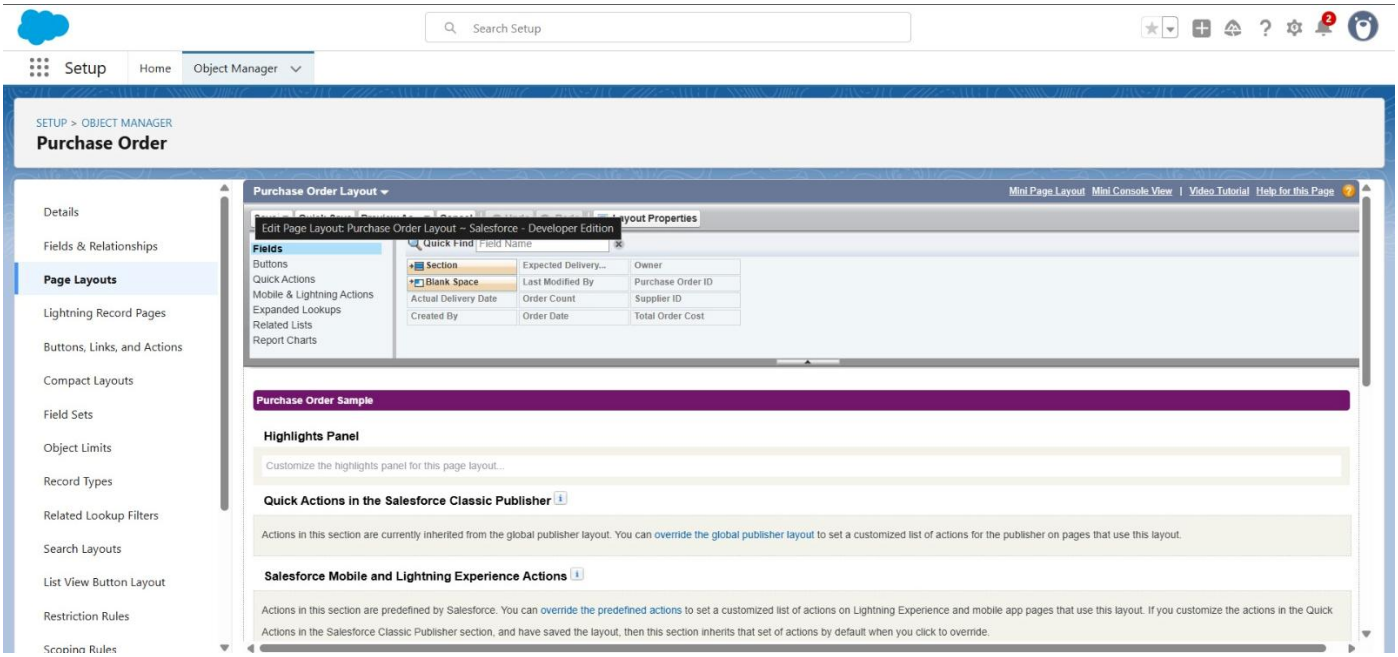
Supplier object



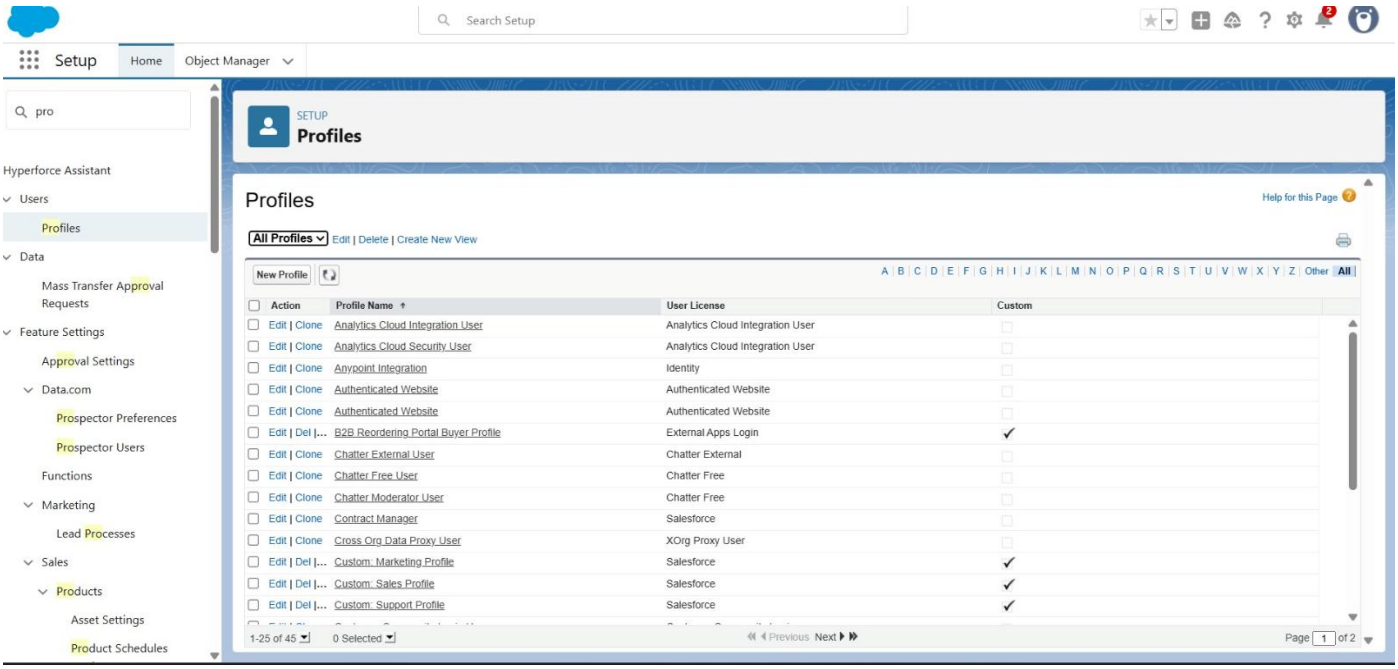
Page layouts of product object



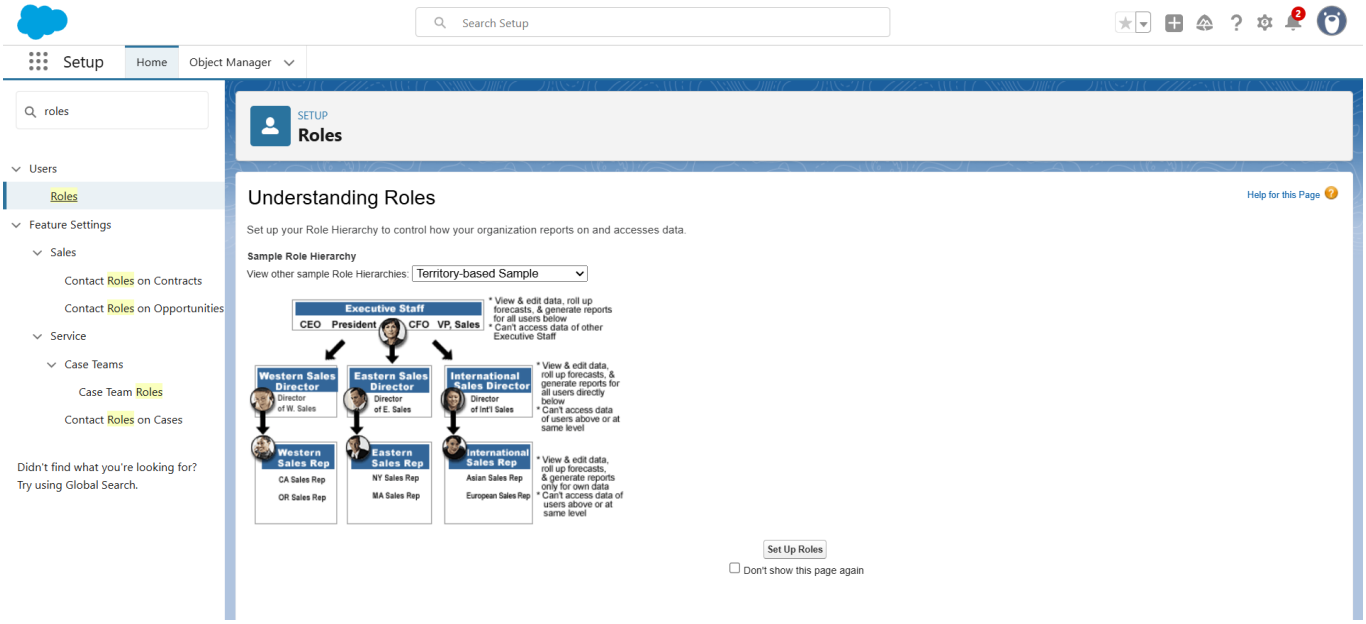
For purchase order



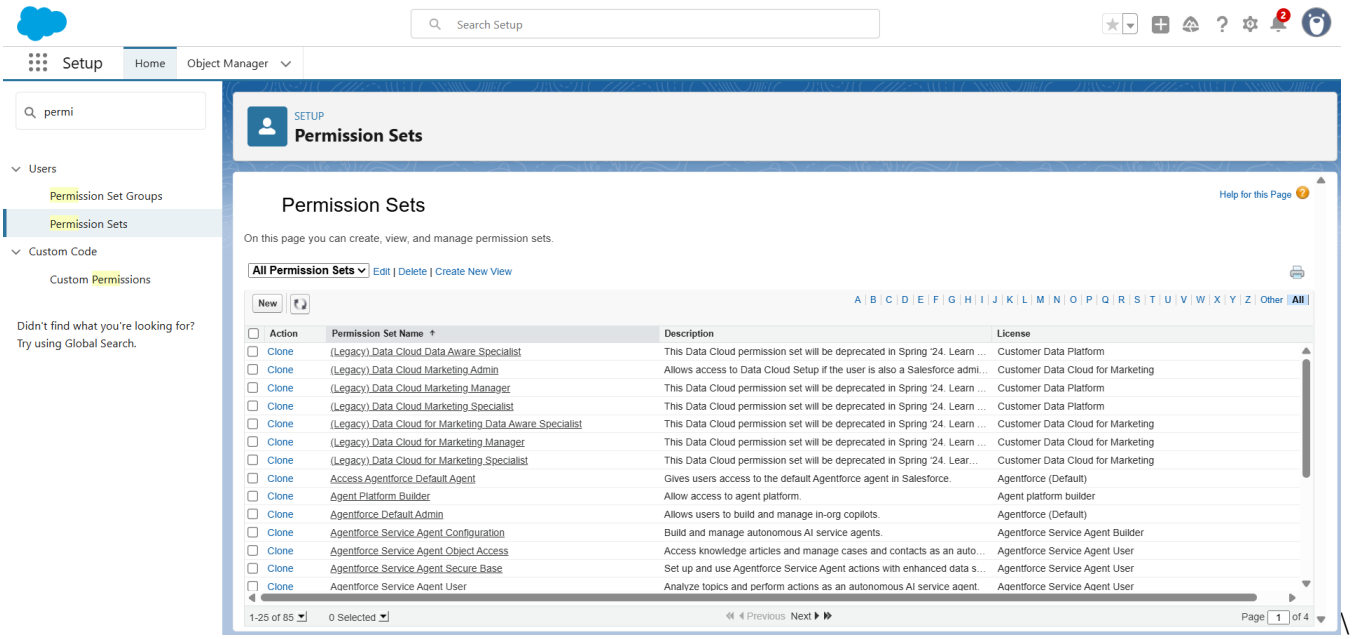
Created inventory manager and purchase manager profile



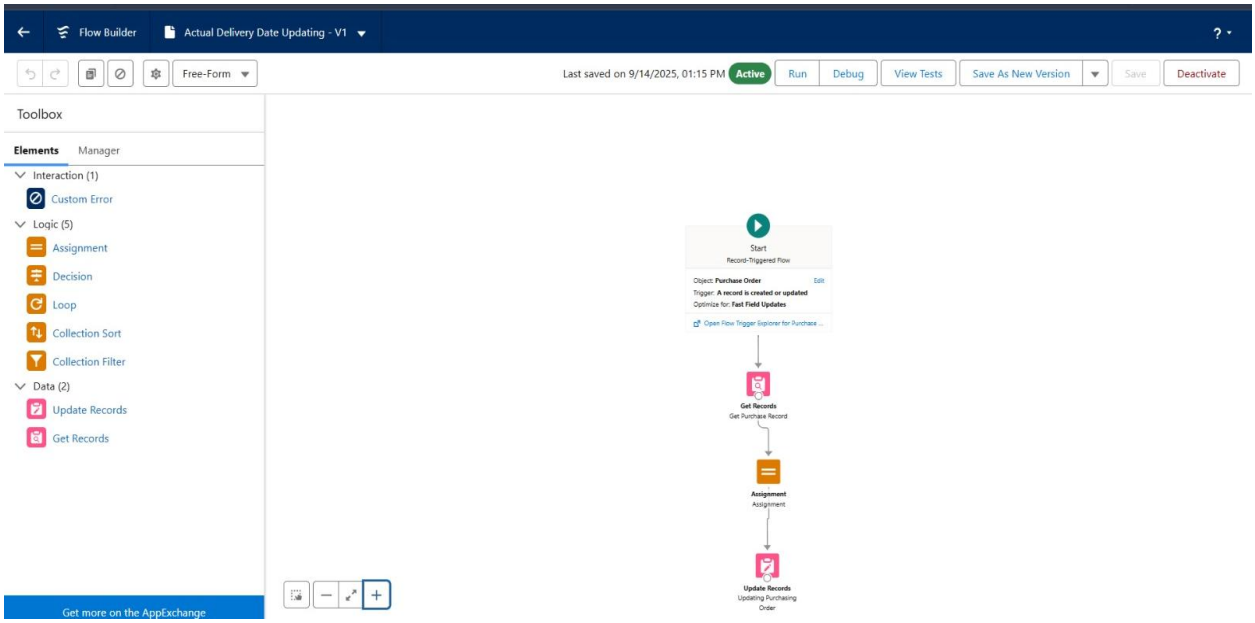
Created Purchase manager roles



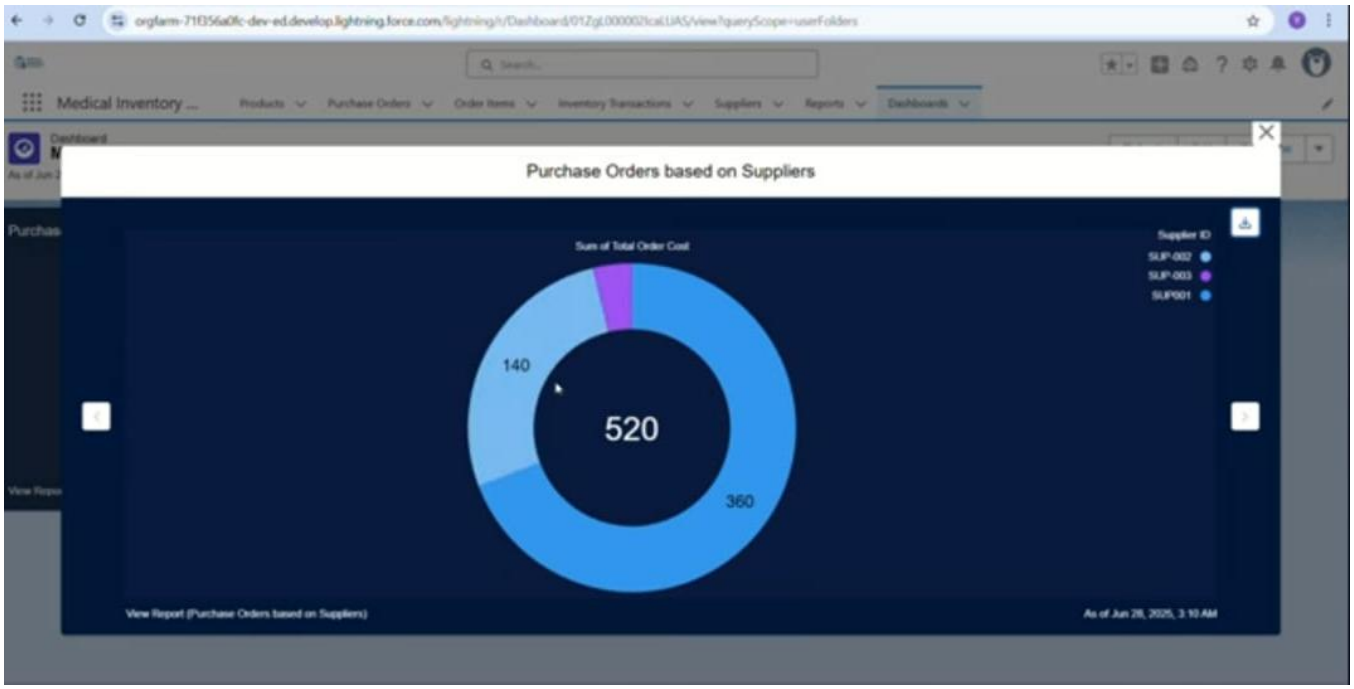
Created a permission sets



Created flow to update the actual delivery date



Created and viewed the dashboard



ADVANTAGES & DISADVANTAGES

Advantages

- Real-time tracking of medical inventory.
- Automation reduces human error.
- Streamlined supplier and order management.
- Automated expiry and low stock alerts.

Disadvantages

- Requires Salesforce knowledge for customization.
- Initial setup and configuration can be time-consuming.

CONCLUSION

The **Medical Inventory Management System** automates and simplifies inventory management in hospitals or clinics. It helps track stock levels, manage supplier relationships, and reduce manual errors. The system was successfully tested to validate all automated features and meets the project's objectives of improving accuracy, efficiency, and patient safety.

APPENDIX

1.Apex Trigger Code

```
trigger CalculateTotalAmountTrigger on Order_Item__c
(after insert, after update, after delete, after undelete) {

    // Call the handler class to handle the logic
    CalculateTotalAmountHandler.calculateTotal(
        Trigger.new,
        Trigger.old,
        Trigger.isInsert,
        Trigger.isUpdate,
        Trigger.isDelete,
        Trigger.isUndelete
    );
}
```

2. Handler Class

```
public class CalculateTotalAmountHandler {

  // Method to calculate the total amount for Purchase Orders based on related Order Items
  public static void calculateTotal(
    List<Order_Item__c> newItems,
    List<Order_Item__c> oldItems,
    Boolean isInsert,
    Boolean isUpdate,
    Boolean isDelete,
    Boolean isUndelete
  ) {

    // Collect Purchase Order IDs affected by changes
    Set<Id> parentIds = new Set<Id>();

    // For insert, update, and undelete scenarios
    if (isInsert || isUpdate || isUndelete) {
      for (Order_Item__c ordItem : newItems) {
        parentIds.add(ordItem.Purchase_Order_Id__c);
      }
    }
```

```
    // For update and delete scenarios
    if (isUpdate || isDelete) {
        for (Order_Item__c ordItem : oldItems) {
            parentIds.add(ordItem.Purchase_Order_Id__c);
        }
    }

    // Map to store Purchase Order total amounts
    Map<Id, Decimal> purchaseToUpdateMap = new Map<Id, Decimal>();

    if (!parentIds.isEmpty()) {
        // Aggregate query to sum Amount__c for each Purchase Order
        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
        ];

        for (AggregateResult aggr : aggrList) {
            Id purchaseOrderId = (Id)aggr.get('Purchase_Order_Id__c');
            Decimal totalAmount = (Decimal)aggr.get('totalAmount');
            purchaseToUpdateMap.put(purchaseOrderId, totalAmount);
        }

        // Prepare Purchase Order updates
        List<Purchase_Order__c> purchaseToUpdate = new List<Purchase_Order__c>();
        for (Id purchaseOrderId : purchaseToUpdateMap.keySet()) {
            Purchase_Order__c purchaseOrder = new Purchase_Order__c(
                Id = purchaseOrderId,
                Total_Order_cost__c = purchaseToUpdateMap.get(purchaseOrderId)
            );
            purchaseToUpdate.add(purchaseOrder);
        }

        // Update Purchase Orders
        if (!purchaseToUpdate.isEmpty()) {
            update purchaseToUpdate;
        }
    }
  }
}
```