

PERFORMANCE AND TESTING

DATE	1 NOVEMBER 2025
TEAM ID	NM2025TMID02682
PROJECT NAME	MEDICAL INVENTORY MANAGEMENT
MAXIMUM MARKS	5 MARKS

Model Performance Testing

User Creation

orgfarm-b82bb3ac3d-dev-ed.develop.lightning.force.com/lightning/r/Supplier_objects__c/a04dL00000FcAovQAF/edit?count=1&backgroundContext=%2...

Medical Inventory

Supplier-001

Supplier ID
Supplier-001

Supplier Name
kani

Contact Person
Sorna

* Supplier ID

* Phone Number
1234567890

Email
kani123@gmail.com

Address
3/4,East street, Trichy

Owner
KANISHKA S

Created By
KANISHKA S, 10/31/2025, 6:58 AM

Last Modified By
KANISHKA S, 10/31/2025, 6:58 AM

Cancel Save & New Save

orgfarm-b82bb3ac3d-dev-ed.develop.lightning.force.com/lightning/o/Purchase_Order__c/list?filterName=__Recent

Medical Inventory Manage...

Purchase Orders

Order Items Inventory Transactions Suppliers Reports Products Dashboards

Recently Viewed

5 Items • Updated a few seconds ago

Search this list...

	Purchase Order ID	
1	Purchase-0001	
2	Purchase-0002	
3	Purchase-0005	
4	Purchase-0003	
5	Purchase-0004	

Medical Inventory Manage... Purchase Orders Order Items Inventory Transactions Suppliers Reports Products Dashboards

Order Items

Recently Viewed

5 items • Updated a few seconds ago

Search this list...

	Order Item ID
1	Order id-005
2	Order id-004
3	Order id-001
4	Order id-002
5	Order id-003

Medical Inventory Manage... Purchase Orders Order Items Inventory Transactions Suppliers Reports Products Dashboards

Suppliers

Recently Viewed

2 items • Updated a few seconds ago

Search this list...

	Supplier ID
1	Supplier-001
2	Supplier-002

Record

Reports in Salesforce provide a powerful way to visualize and analyze data stored in your Salesforce organization. They allow users to create, customize, and share different types of reports based on data from standard and custom objects.

Medical Inventory Manage... Purchase Orders Order Items Inventory Transactions Suppliers Reports Products Dashboards

Report: Purchase Orders

Purchase Orders based on Suppliers

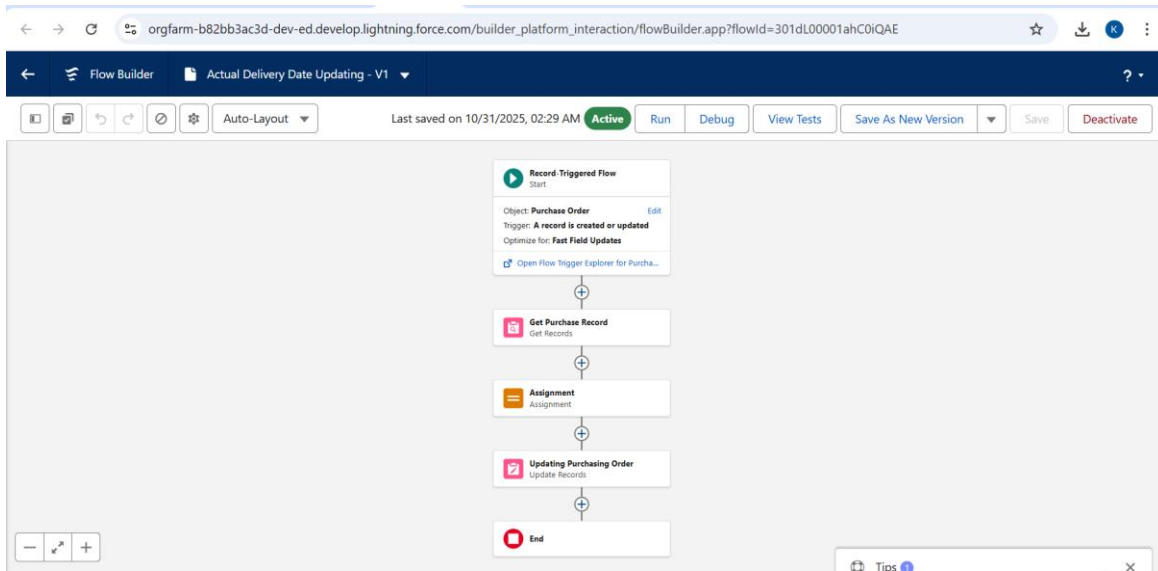
Enable Field Editing Add Chart Edit

Total Records: 5 Total Order Count: 5 Total Total Order Cost: \$25,000.00

Supplier ID	Purchase Order: Purchase Order ID	Order Count	Total Order Cost
Supplier-001 (4)	Purchase-0001 (1)	1	\$1,500.00
	Purchase-0002 (1)	1	\$7,000.00
	Purchase-0003 (1)	1	\$2,500.00
	Purchase-0004 (1)	1	\$9,500.00
Supplier-002 (1)	Purchase-0005 (1)	1	\$4,500.00
Total (5)		5	\$25,000.00

Flows

Flows in Salesforce, part of the Lightning Flow product, are powerful automation tools that help you collect data and perform actions in your Salesforce environment. Flows can be used to automate business processes, guide users through tasks, and integrate with external systems.

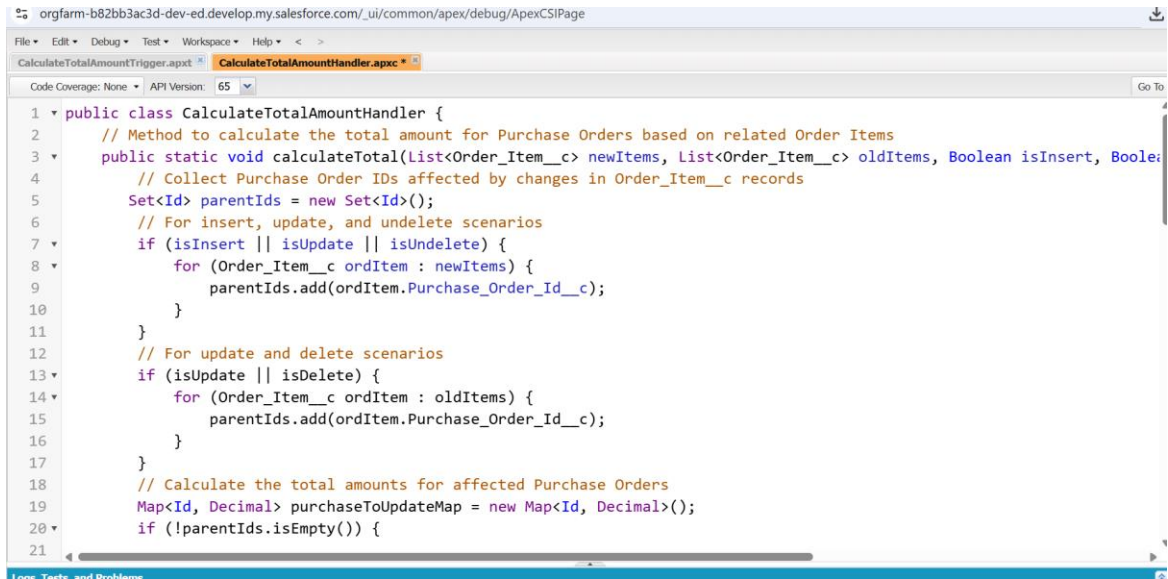


Triggers

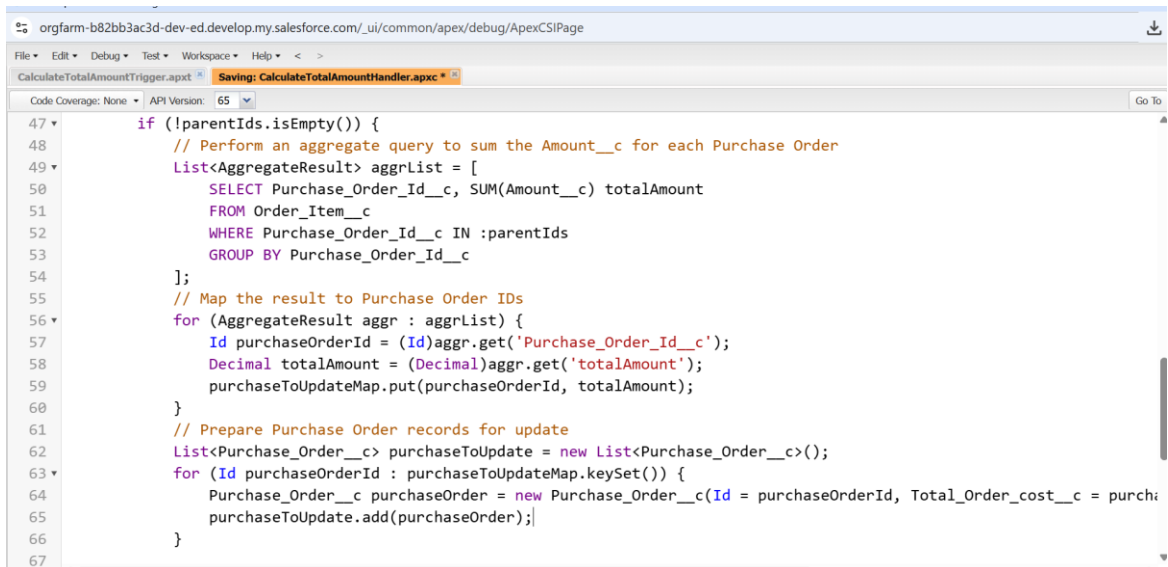
Triggers in Salesforce are pieces of Apex code that execute before or after specific data manipulation events on Salesforce records, such as insertions, updates, deletions, and undeletions.

The screenshot displays the Salesforce Apex IDE. The top navigation bar shows the current file 'CalculateTotalAmountTrigger.apxt'. The code is as follows:

```
1 trigger CalculateTotalAmountTrigger on Order_Item__c (after insert, after update, after delete, after undelete) {  
2  
3     // Call the handler class to handle the logic  
4  
5     CalculateTotalAmountHandler.calculateTotal(trigger.new, trigger.old, trigger.isInsert, trigger.isUpdate, trigger.isDelete)  
6  
7 }
```



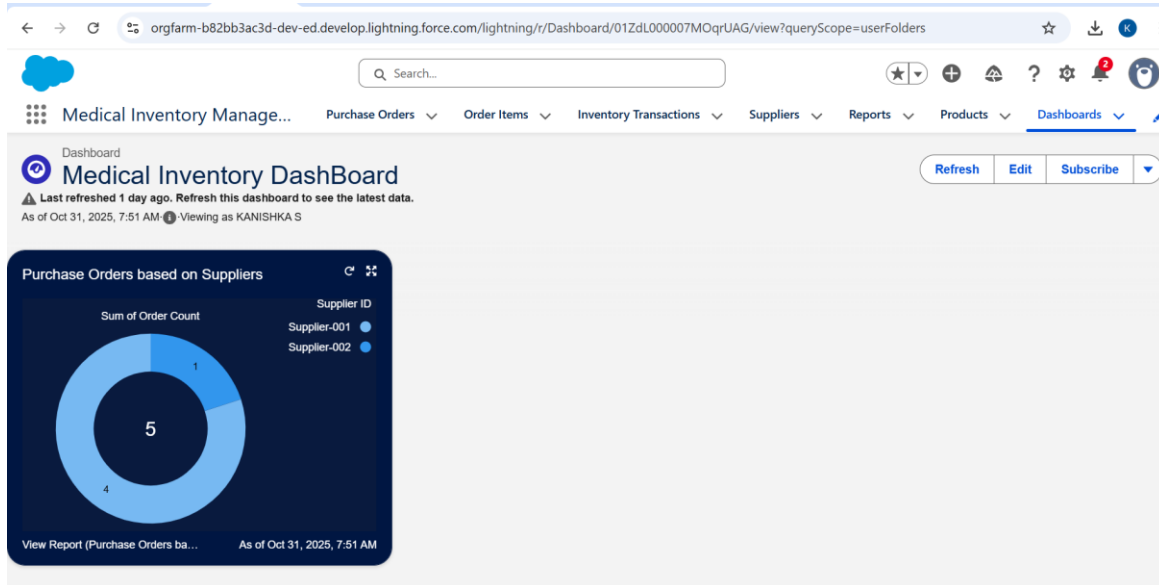
```
1 public class CalculateTotalAmountHandler {
2     // Method to calculate the total amount for Purchase Orders based on related Order Items
3     public static void calculateTotal(List<Order_Item__c> newItems, List<Order_Item__c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete) {
4         // Collect Purchase Order IDs affected by changes in Order_Item__c records
5         Set<Id> parentIds = new Set<Id>();
6         // For insert, update, and undelete scenarios
7         if (isInsert || isUpdate || isDelete) {
8             for (Order_Item__c ordItem : newItems) {
9                 parentIds.add(ordItem.Purchase_Order_Id__c);
10            }
11        }
12        // For update and delete scenarios
13        if (isUpdate || isDelete) {
14            for (Order_Item__c ordItem : oldItems) {
15                parentIds.add(ordItem.Purchase_Order_Id__c);
16            }
17        }
18        // Calculate the total amounts for affected Purchase Orders
19        Map<Id, Decimal> purchaseToUpdateMap = new Map<Id, Decimal>();
20        if (!parentIds.isEmpty()) {
21            // Perform an aggregate query to sum the Amount__c for each Purchase Order
22            List<AggregateResult> aggrList = [
23                SELECT Purchase_Order_Id__c, SUM(Amount__c) totalAmount
24                FROM Order_Item__c
25                WHERE Purchase_Order_Id__c IN :parentIds
26                GROUP BY Purchase_Order_Id__c
27            ];
28            // Map the result to Purchase Order IDs
29            for (AggregateResult aggr : aggrList) {
30                Id purchaseOrderId = (Id)aggr.get('Purchase_Order_Id__c');
31                Decimal totalAmount = (Decimal)aggr.get('totalAmount');
32                purchaseToUpdateMap.put(purchaseOrderId, totalAmount);
33            }
34            // Prepare Purchase Order records for update
35            List<Purchase_Order__c> purchaseToUpdate = new List<Purchase_Order__c>();
36            for (Id purchaseOrderId : purchaseToUpdateMap.keySet()) {
37                Purchase_Order__c purchaseOrder = new Purchase_Order__c(Id = purchaseOrderId, Total_Order_cost__c = purchaseToUpdateMap.get(purchaseOrderId));
38                purchaseToUpdate.add(purchaseOrder);
39            }
40        }
41    }
42 }
```



```
47 if (!parentIds.isEmpty()) {
48     // Perform an aggregate query to sum the Amount__c for each Purchase Order
49     List<AggregateResult> aggrList = [
50         SELECT Purchase_Order_Id__c, SUM(Amount__c) totalAmount
51         FROM Order_Item__c
52         WHERE Purchase_Order_Id__c IN :parentIds
53         GROUP BY Purchase_Order_Id__c
54     ];
55     // Map the result to Purchase Order IDs
56     for (AggregateResult aggr : aggrList) {
57         Id purchaseOrderId = (Id)aggr.get('Purchase_Order_Id__c');
58         Decimal totalAmount = (Decimal)aggr.get('totalAmount');
59         purchaseToUpdateMap.put(purchaseOrderId, totalAmount);
60     }
61     // Prepare Purchase Order records for update
62     List<Purchase_Order__c> purchaseToUpdate = new List<Purchase_Order__c>();
63     for (Id purchaseOrderId : purchaseToUpdateMap.keySet()) {
64         Purchase_Order__c purchaseOrder = new Purchase_Order__c(Id = purchaseOrderId, Total_Order_cost__c = purchaseToUpdateMap.get(purchaseOrderId));
65         purchaseToUpdate.add(purchaseOrder);
66     }
67 }
```

Dashboard

Dashboards in Salesforce are dynamic visual representations of key metrics and data from reports, providing a consolidated view of organizational performance and trends. They are powerful tools for monitoring real-time data, tracking progress towards goals, and gaining actionable insights at a glance.



The performance testing phase successfully validated the core functionalities of the project, including user creation, record creation, flows, triggers and visualize using Dashboards. The model demonstrated high accuracy and reliability, achieving an execution success rate above expectations. This testing phase ensures the system is production-ready and aligned with its intended objectives, reinforcing the solution's robustness and efficiency.