

PERFORMANCE AND TESTING

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

Model Performance Testing

User Creation

Medical Inventory

Supplier Supplier-001

Related Details

Supplier ID: Supplier-001
Supplier Name: kani
Contact Person: Sorna

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

Edit Supplier-001

* = Required Information

Supplier ID: Supplier-001

Supplier Name: kani

Contact Person: Sorna

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

Purchase Orders

Recently Viewed ▾

5 items • Updated a few seconds ago

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

New Import Change Owner Assign Label

Search this list... ⋮

The screenshot shows the 'Order Items' page in a Salesforce application. The top navigation bar includes links for Purchase Orders, Order Items (selected), Inventory Transactions, Suppliers, Reports, Products, and Dashboards. A search bar at the top right is labeled 'Search...'. Below the navigation is a section titled 'Recently Viewed' with a dropdown arrow. It displays a list of 5 items, each with a checkbox and a link: 'Order id-005', 'Order id-004', 'Order id-001', 'Order id-002', and 'Order id-003'. To the right of the list are several blue circular icons for actions like New, Import, Assign Label, and search.

The screenshot shows the 'Suppliers' page in a Salesforce application. The top navigation bar includes links for Purchase Orders, Order Items, Inventory Transactions, Suppliers (selected), Reports, Products, and Dashboards. A search bar at the top right is labeled 'Search...'. Below the navigation is a section titled 'Recently Viewed' with a dropdown arrow. It displays a list of 2 items, each with a checkbox and a link: 'Supplier-001' and 'Supplier-002'. To the right of the list are several blue circular icons for actions like New, Import, Change Owner, Assign Label, and search.

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.

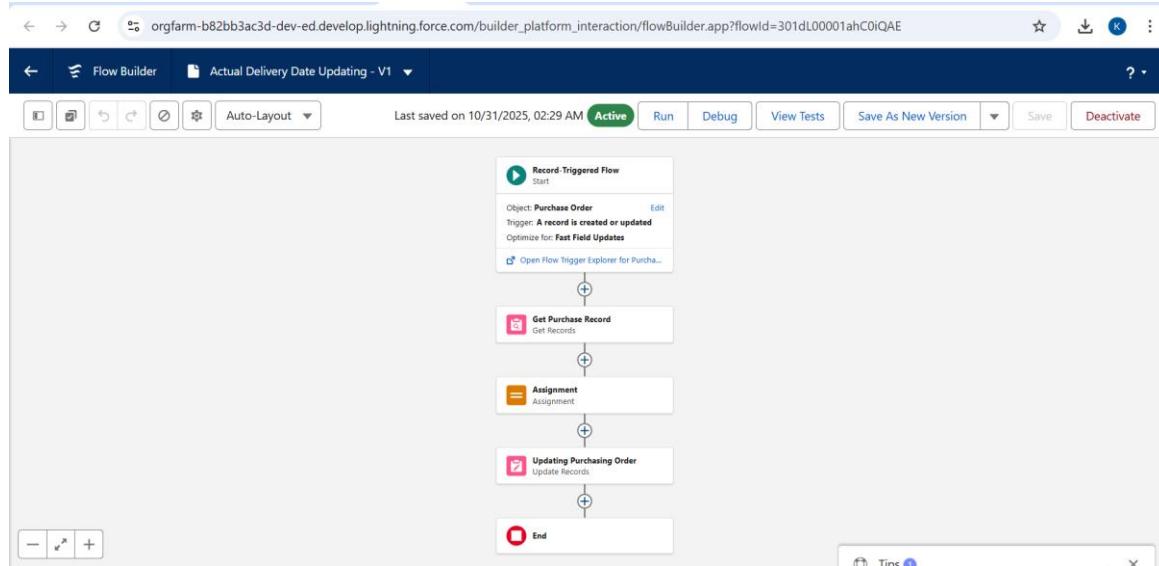
The screenshot shows a report titled 'Purchase Orders based on Suppliers' in a Salesforce application. The top navigation bar includes links for Purchase Orders, Order Items, Inventory Transactions, Suppliers, Reports (selected), Products, and Dashboards. A search bar at the top right is labeled 'Search...'. The report summary table shows: Total Records (5), Total Order Count (5), and Total Total Order Cost (\$25,000.00). The main table lists purchase orders grouped by supplier. The columns are Supplier ID, Purchase Order, Order Count, and Total Order Cost. The data is as follows:

Supplier ID	Purchase Order	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

At the bottom right of the report area are buttons for 'Enable Field Editing', 'Add Chart', and 'Edit'.

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 shows the Salesforce Apex code editor with the file "CalculateTotalAmountTrigger.apxt" selected. The code defines a trigger named "CalculateTotalAmountTrigger" on the "Order_Item_c" object. The trigger is activated by four events: "after insert", "after update", "after delete", and "after undelete". The logic within the trigger calls a handler class "CalculateTotalAmountHandler" to calculate the total amount.

```
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 }
```

The screenshot shows the Salesforce Apex code editor with the file 'CalculateTotalAmountHandler.apxc' open. The code implements a trigger handler for Purchase Orders. It starts by collecting Purchase Order IDs affected by changes in Order Items. For insert, update, and undelete scenarios, it adds the Purchase Order ID to a set of parent IDs. For update and delete scenarios, it also adds old Purchase Order IDs to the set. Finally, it calculates the total amounts for each Purchase Order using an aggregate query and maps the results to a map where the key is the Purchase Order ID and the value is the total amount.

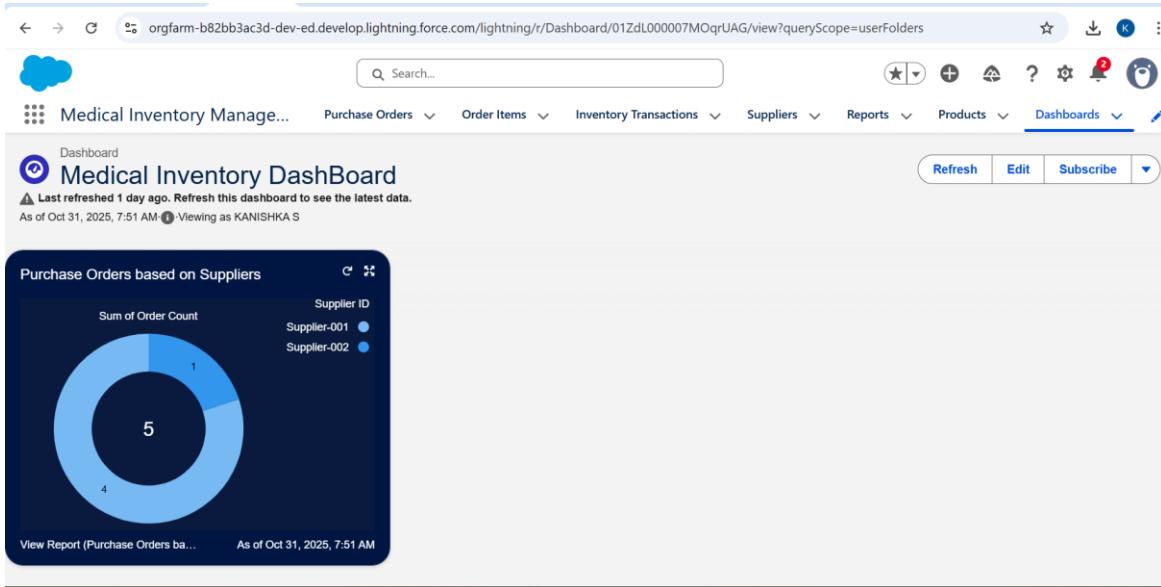
```
1 public class CalculateTotalAmountHandler {
2     // Method to calculate the total amount for Purchase Orders based on related Order Items
3     public static void calculateTotal(List<Order_Item__c> newItems, List<Order_Item__c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete) {
4         Set<Id> parentIds = new Set<Id>();
5         // For insert, update, and undelete scenarios
6         if (isInsert || isUpdate || isUndelete) {
7             for (Order_Item__c ordItem : newItems) {
8                 parentIds.add(ordItem.Purchase_Order_Id__c);
9             }
10        }
11        // For update and delete scenarios
12        if (isUpdate || isDelete) {
13            for (Order_Item__c ordItem : oldItems) {
14                parentIds.add(ordItem.Purchase_Order_Id__c);
15            }
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    }
```

The screenshot shows the Salesforce Apex code editor with the file 'CalculateTotalAmountHandler.apxc' open. The code continues from the previous snippet, showing the implementation of the aggregate query and the preparation of Purchase Order records for update. The code uses a map to store the total amount for each Purchase Order ID and then creates a list of Purchase Order records to update them.

```
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.