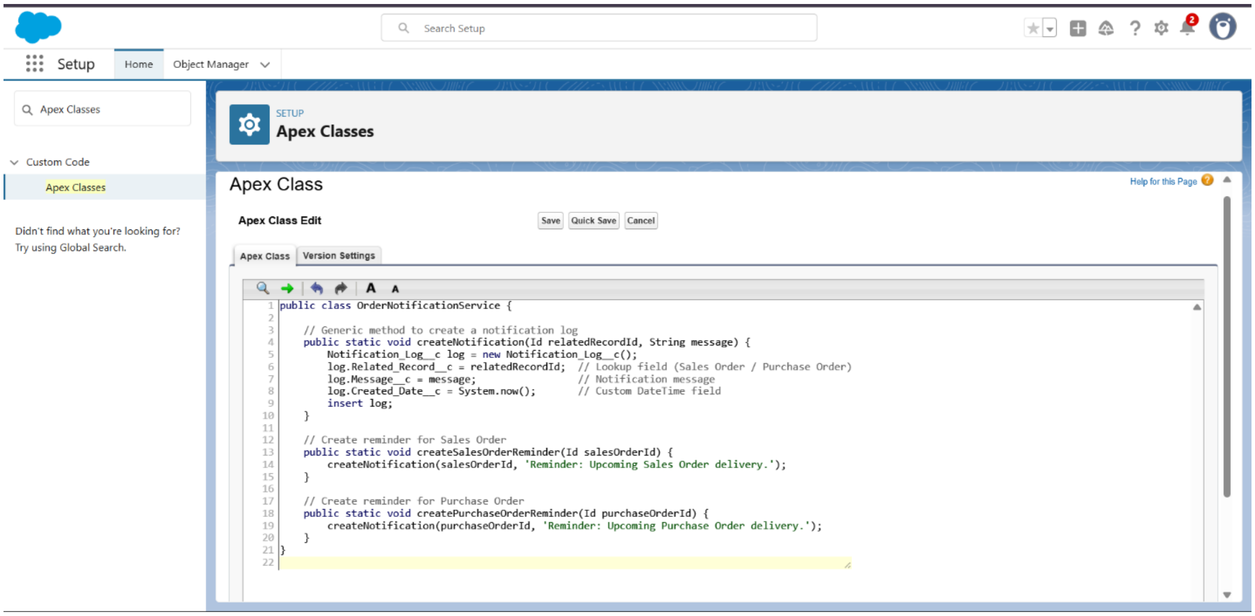
Phase 5: Apex Programming (Developer) – Manufacturing & Supply Chain Project

In this phase, Apex programming concepts were implemented to extend automation and add advanced logic to the Manufacturing & Supply Chain CRM project. Apex was used to handle scenarios that couldn't be fully achieved with declarative tools like flows and process automation.

1. Apex Classes

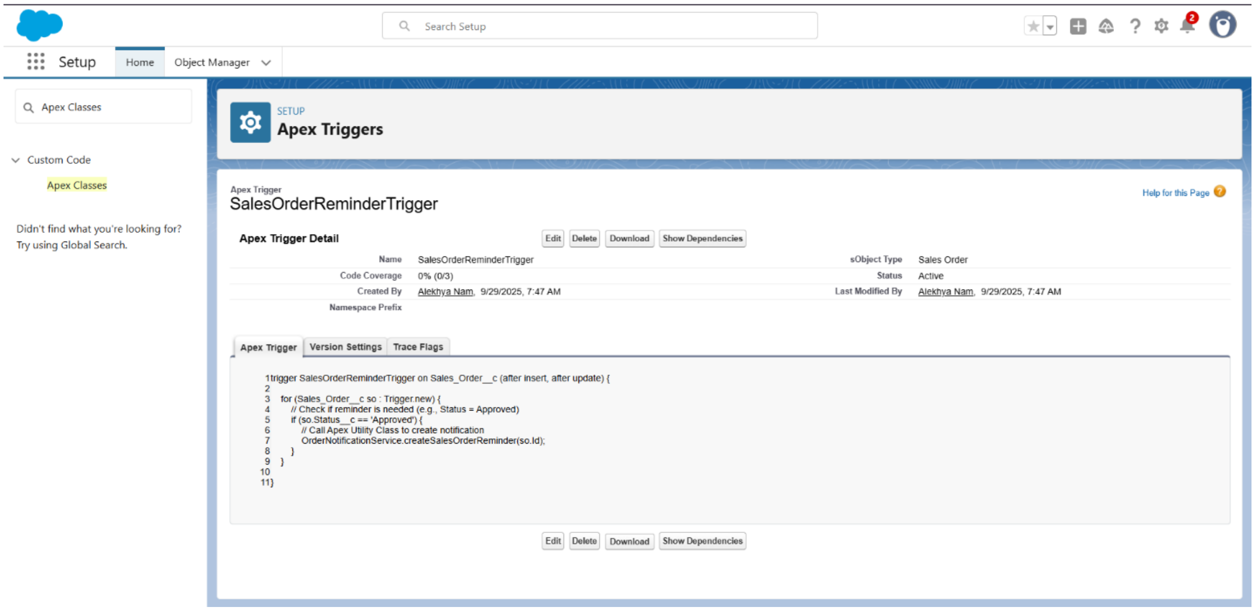
A utility class **OrderNotificationService** was created to handle automatic reminder creation by inserting **Notification_Log__c** records when a Sales Order or Purchase Order required follow-up.



```
public class OrderNotificationService {

    // Generic method to create a notification log
    public static void createNotification(Id relatedRecordId, String message) {
        Notification_Log__c log = new Notification_Log__c();
        log.Related_Record__c = relatedRecordId;  // Lookup field (Sales Order / Purchase Order)
        log.Message__c = message;                 // Notification message
        log.Created_Date__c = System.now();       // Custom DateTime field
        insert log;
    }

    // Create reminder for Sales Order
    public static void createSalesOrderReminder(Id salesOrderId) {
        createNotification(salesOrderId, 'Reminder: Upcoming Sales Order delivery.');
    }

    // Create reminder for Purchase Order
    public static void createPurchaseOrderReminder(Id purchaseOrderId) {
        createNotification(purchaseOrderId, 'Reminder: Upcoming Purchase Order delivery.');
    }
}
```

2. Apex Triggers

An Apex Trigger on **Production_Schedule__c** was developed to prevent overlapping schedules. It ensures that no two production batches are assigned to the same machine at the same time.

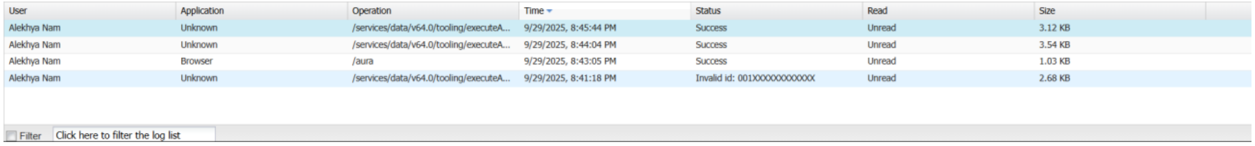


Apex Trigger
SalesOrderReminderTrigger

Apex Trigger Detail

Name	SalesOrderReminderTrigger	sObject Type	Sales Order
Code Coverage	0% (0/3)	Status	Active
Created By	Alekhya Nam, 9/29/2025, 7:47 AM	Last Modified By	Alekhya Nam, 9/29/2025, 7:47 AM
Namespace Prefix			

```
trigger SalesOrderReminderTrigger on Sales_Order__c (after insert, after update) {

  for (Sales_Order__c so : Trigger.new) {
      // Check if reminder is needed (e.g., Status = Approved)
      if (so.Status__c == 'Approved') {
          // Call Apex Utility Class to create notification
          OrderNotificationService.createSalesOrderReminder(so.Id);
      }
  }

}
```

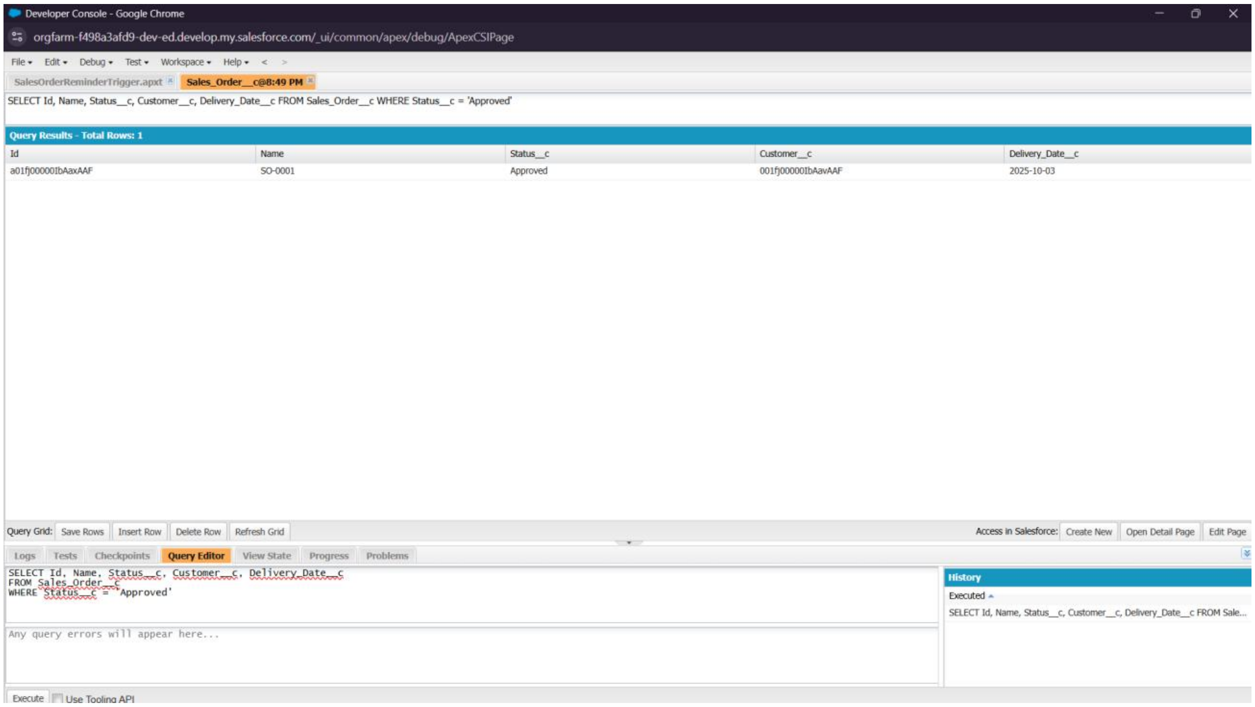


User	Application	Operation	Time	Status	Read	Size
Alekhya Nam	Unknown	/services/data/v64.0/tooling/executeA...	9/29/2025, 8:45:44 PM	Success	Unread	3.12 KB
Alekhya Nam	Unknown	/services/data/v64.0/tooling/executeA...	9/29/2025, 8:44:04 PM	Success	Unread	3.54 KB
Alekhya Nam	Browser	/aura	9/29/2025, 8:43:05 PM	Success	Unread	1.03 KB
Alekhya Nam	Unknown	/services/data/v64.0/tooling/executeA...	9/29/2025, 8:41:18 PM	Invalid id: 001XXXXXXXXXXXX	Unread	2.68 KB

3. SOQL and SOSL Queries

- **SOQL Example:** Fetch pending Sales Orders for a particular Customer.
- **SOSL Example:** Search for a Supplier by name.

```
List<List<Account>> results =
  [FIND 'Steel*' IN ALL FIELDS RETURNING Account(Id, Name, Phone, Email)];
System.debug(results);
```


Developer Console - Google Chrome
orgfarm-f498a3afd9-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File ▾
Edit ▾
Debug ▾
Test ▾
Workspace ▾
Help ▾
SalesOrderReminderTrigger.apxt
Sales_Order__c@8:49 PM

```
SELECT Id, Name, Status__c, Customer__c, Delivery_Date__c FROM Sales_Order__c WHERE Status__c = 'Approved'
```

Query Results - Total Rows: 1

Id	Name	Status__c	Customer__c	Delivery_Date__c
a01fj00000IbAaxAAF	SO-0001	Approved	001fj00000IbAavAAF	2025-10-03

```
SELECT Id, Name, Status__c, Customer__c, Delivery_Date__c
FROM Sales_Order__c
WHERE Status__c = 'Approved'
```

```
Any query errors will appear here...
```

SELECT Id, Name, Status__c, Customer__c, Delivery_Date__c FROM Sale...

---

4. Batch Apex

A batch class **OrderBatchProcessor** was created to automatically update old pending Purchase Orders to “Closed” after 90 days.

---

5. Scheduled Apex

A scheduled class **DailyOrderReminderScheduler** was created to send daily reminders at 9 AM for upcoming deliveries and pending production schedules.

---

6. Test Classes

A test class **ManufacturingTests** was developed to cover triggers, batch apex, and the notification service. It creates test data including Customers, Suppliers, and Production Schedules. It verifies that:

- Double booking of machines is prevented
- Batch jobs correctly close old Purchase Orders
- Notification logs are created as expected

7. Summary

In Phase 5, Apex programming was applied to enforce complex business logic not achievable by declarative tools alone. This included preventing machine double-booking in production, automatically closing overdue Purchase Orders, sending daily reminders for deliveries, and ensuring robust testing with Apex test classes. These features strengthened the Manufacturing & Supply Chain CRM and made the system more reliable and efficient for end users