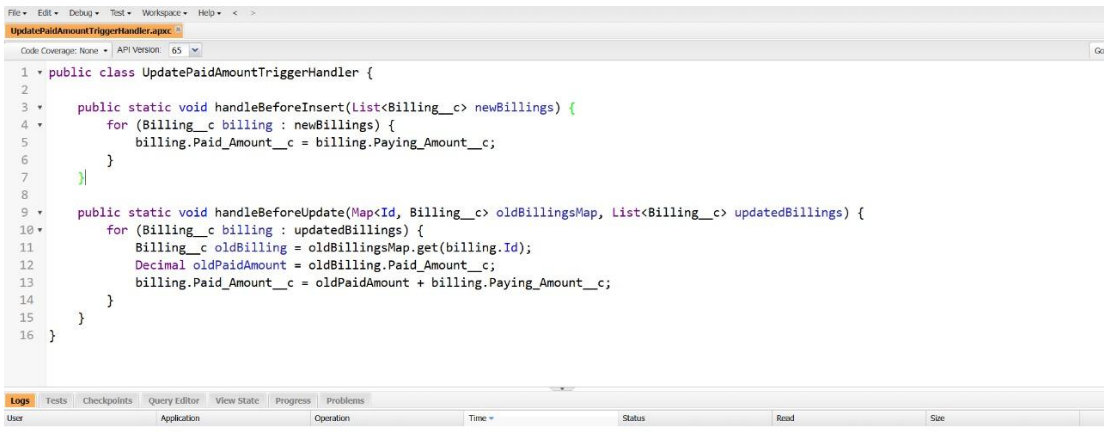
APEX TRIGGERS AND CLASSES

Date	05 November 2025
Team Id	NM2025TMID01192
Project Name	CRM Application for Jewel Management

Where declarative automation cannot handle complex conditions, Apex Triggers and Apex Classes were developed.

Examples:

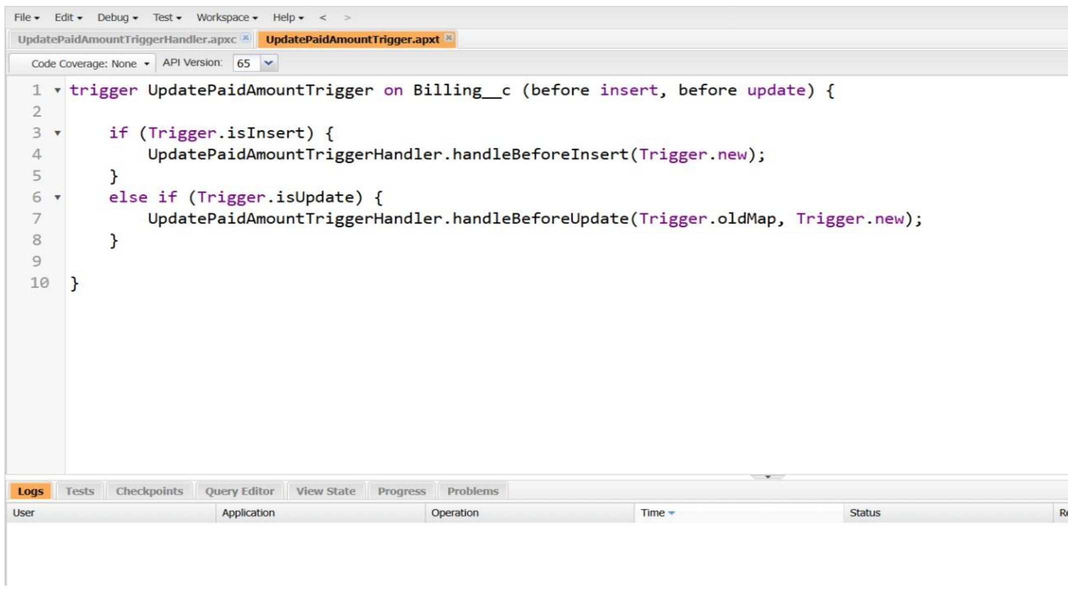
- **Trigger on Order:** Automatically calculate and update total order value based on selected jewelry items.
- **Trigger on Payment:** Update the Payment Status field in the related Order record when full payment is received.
- **Trigger on Jewelry Item:** Automatically change item status to "Out of Stock" when stock quantity reaches zero.


File Edit Debug Test Workspace Help
UpdatePaidAmountTriggerHandler.apxc
Code Coverage: None
API Version: 65
Logs Tests Checkpoints Query Editor View State Progress Problems
User Application Operation Time Status Read Size

```
public class UpdatePaidAmountTriggerHandler {

    public static void handleBeforeInsert(List<Billing__c> newBillings) {
        for (Billing__c billing : newBillings) {
            billing.Paid_Amount__c = billing.Paying_Amount__c;
        }
    }

    public static void handleBeforeUpdate(Map<Id, Billing__c> oldBillingsMap, List<Billing__c> updatedBillings) {
        for (Billing__c billing : updatedBillings) {
            Billing__c oldBilling = oldBillingsMap.get(billing.Id);
            Decimal oldPaidAmount = oldBilling.Paid_Amount__c;
            billing.Paid_Amount__c = oldPaidAmount + billing.Paying_Amount__c;
        }
    }
}
```


File Edit Debug Test Workspace Help < >
UpdatePaidAmountTriggerHandler.apxc
UpdatePaidAmountTrigger.apxt
Code Coverage: None
API Version: 65
Logs Tests Checkpoints Query Editor View State Progress Problems
User Application Operation Time Status

```
trigger UpdatePaidAmountTrigger on Billing__c (before insert, before update) {

    if (Trigger.isInsert) {
        UpdatePaidAmountTriggerHandler.handleBeforeInsert(Trigger.new);
    }
    else if (Trigger.isUpdate) {
        UpdatePaidAmountTriggerHandler.handleBeforeUpdate(Trigger.oldMap, Trigger.new);
    }

}
```

Apex Classes were used to implement backend logic for:

- Generating invoices.
- Sending scheduled payment reminders.
- Running daily maintenance batch jobs.

This custom logic provides the system with flexibility and enhances automation accuracy.