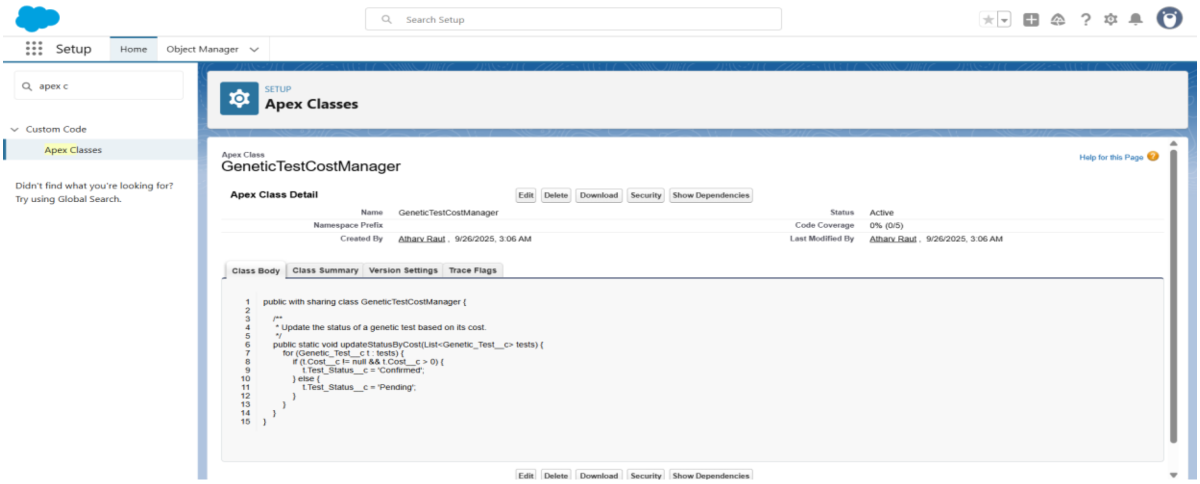
Phase 5: Apex Programming (Developer)

Step 1: Apex Class — GeneticTestCostManager

Purpose: Perform matching and create interviews programmatically.

Steps to create Apex Class:

1. Click **Setup (☼◦O) → Quick Find → Apex Classes → New**.
2. Copy and paste the GeneticTestCostManager code below.
3. Click **Save**.



Apex Class
GeneticTestCostManager

Apex Class Detail

Name	GeneticTestCostManager	Status	Active
Namespace Prefix		Code Coverage	0% (0/5)
Created By	Atharv Raut , 9/26/2025, 3:06 AM	Last Modified By	Atharv Raut , 9/26/2025, 3:06 AM

```
public with sharing class GeneticTestCostManager {

    /**
     * Update the status of a genetic test based on its cost.
     */
    public static void updateStatusByCost(List<Genetic_Test__c> tests) {
        for (Genetic_Test__c t : tests) {
            if (t.Cost__c != null && t.Cost__c > 0) {
                t.Test_Status__c = 'Confirmed';
            } else {
                t.Test_Status__c = 'Pending';
            }
        }
    }
}
```

Code Logic Overview:

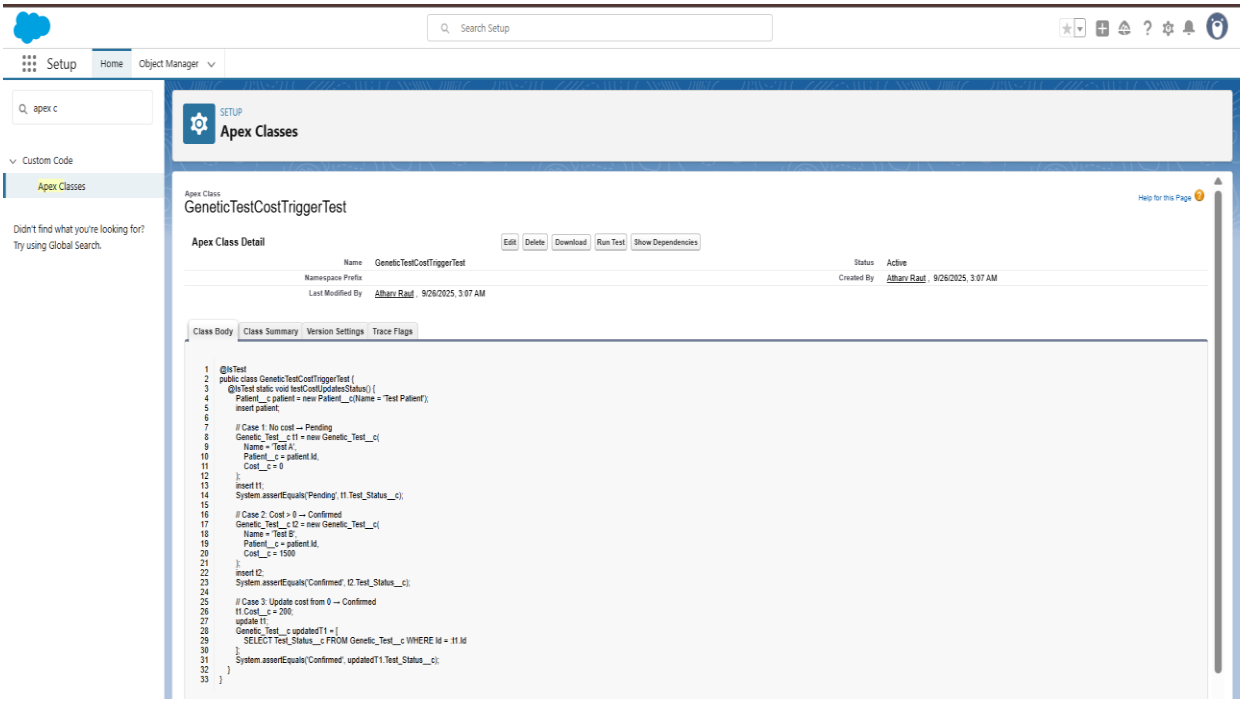
- **Method 1: updateStatusByCost(tests)**
    - Input: List of Genetic_Test__c records.
    - If Cost__c is not null and greater than 0 → set Test_Status__c = 'Confirmed'.
    - Else → set Test_Status__c = 'Pending'.
    - Output: Updated list of test records (status field adjusted based on cost)

Step 2: Apex Test Class — GeneticTestCostTriggerTest

Purpose: Ensure at least 75% coverage and test functionality.

Steps to create Test Class:

1. Click **Setup → Apex Classes → New**.
2. Copy and paste the GeneticTestCostTriggerTest code below.
3. Click **Save**.



Apex Class
GeneticTestCostTriggerTest

Apex Class Detail

Name: GeneticTestCostTriggerTest
Namespace Prefix:
Last Modified By: Atharv Raut , 9/26/2025, 3:07 AM
Status: Active
Created By: Atharv Raut , 9/26/2025, 3:07 AM

```
@IsTest
public class GeneticTestCostTriggerTest {
    @IsTest static void testCostUpdatesStatus() {
        Patient__c patient = new Patient__c(Name = 'Test Patient');
        insert patient;

        // Case 1: No cost → Pending
        Genetic_Test__c t1 = new Genetic_Test__c(
            Name = 'Test A',
            Patient__c = patient.Id,
            Cost__c = 0
        );
        insert t1;
        System.assertEquals('Pending', t1.Test_Status__c);

        // Case 2: Cost > 0 → Confirmed
        Genetic_Test__c t2 = new Genetic_Test__c(
            Name = 'Test B',
            Patient__c = patient.Id,
            Cost__c = 1500
        );
        insert t2;
        System.assertEquals('Confirmed', t2.Test_Status__c);

        // Case 3: Update cost from 0 → Confirmed
        t1.Cost__c = 200;
        update t1;
        Genetic_Test__c updatedT1 = [
            SELECT Test_Status__c FROM Genetic_Test__c WHERE Id = :t1.Id
        ];
        System.assertEquals('Confirmed', updatedT1.Test_Status__c);
    }
}
```

Test Logic Overview:

- Creates test Patient__c.
- Inserts Genetic_Test__c with Cost__c = 0 → asserts Test_Status__c = 'Pending'.
- Inserts Genetic_Test__c with Cost__c > 0 → asserts Test_Status__c = 'Confirmed'.
- Updates first test from Cost__c = 0 to Cost__c > 0 → asserts Test_Status__c = 'Confirmed'.