Phase 5: Apex Programming (Developer)

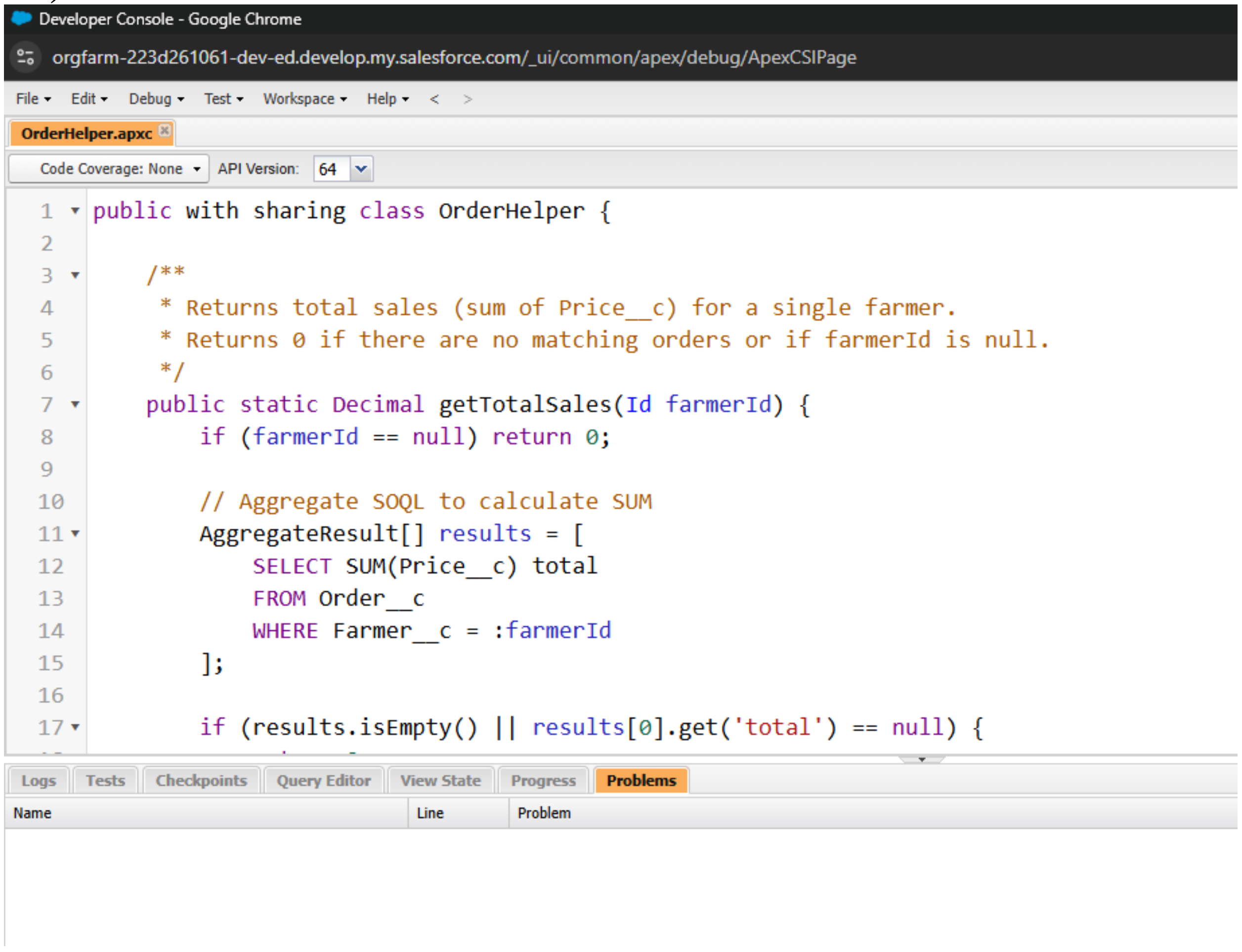
Goal: The goal of Phase 5 is to extend Salesforce functionality using Apex programming. While declarative automation (Flows, Workflows) covered most business needs, certain processes in GreenHarvest require custom business logic, bulk data processing, and asynchronous execution that only Apex can handle.

Apex Classes & Objects

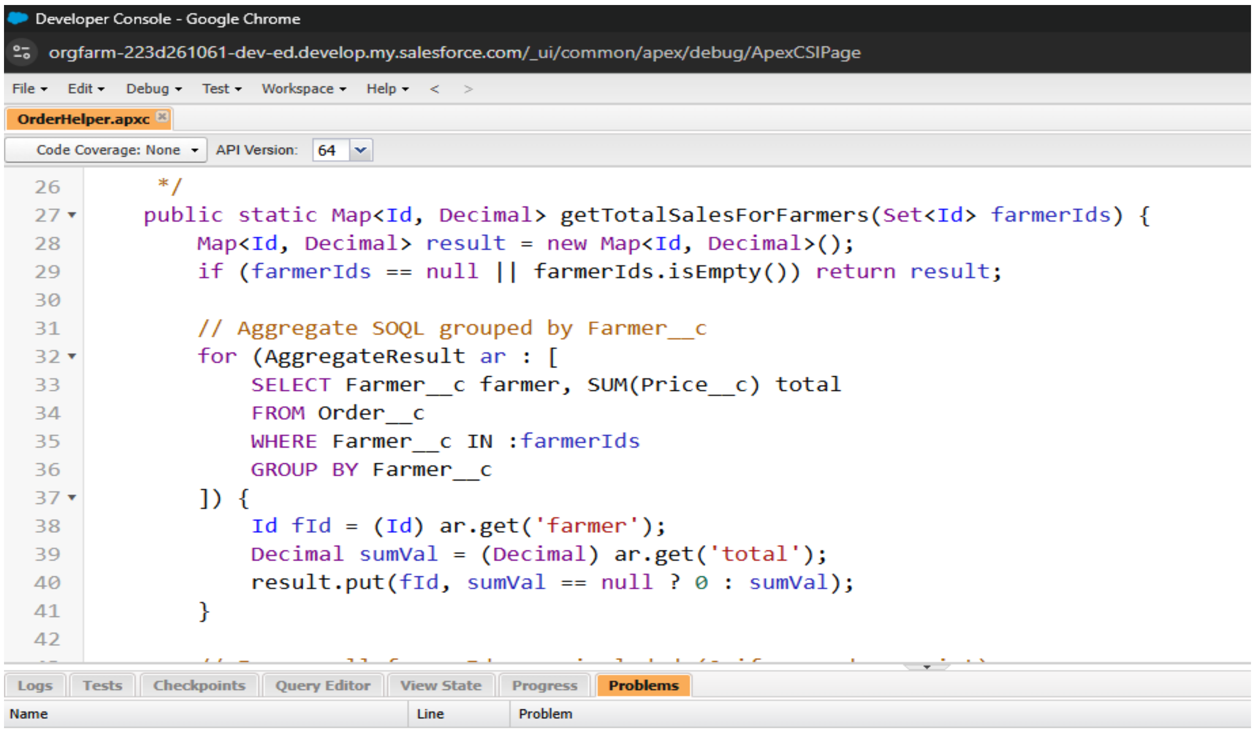
Purpose: Encapsulate reusable business logic (e.g., calculating Farmer's total income).

Implementation Steps:

1. Created an `OrderHelper` Apex class.
2. Implemented `getTotalSales(farmerId)` for single farmer income.
3. Implemented `getTotalSalesForFarmers(Set<Id>)` for bulk farmer income (trigger-safe).

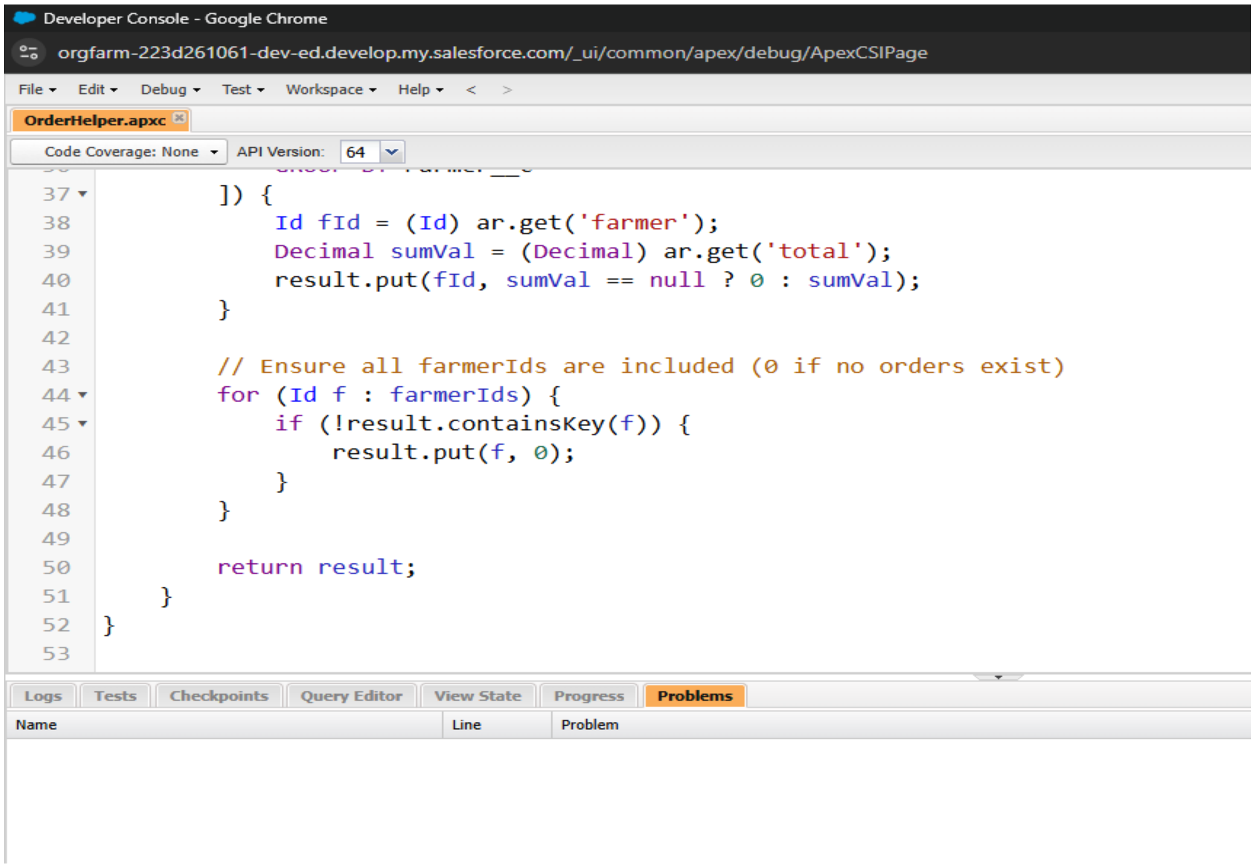

Developer Console - Google Chrome
orgfarm-223d261061-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
OrderHelper.apxc
Code Coverage: None
API Version: 64
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```apex
public with sharing class OrderHelper {

    /**
     * Returns total sales (sum of Price__c) for a single farmer.
     * Returns 0 if there are no matching orders or if farmerId is null.
     */
    public static Decimal getTotalSales(Id farmerId) {
        if (farmerId == null) return 0;

        // Aggregate SOQL to calculate SUM
        AggregateResult[] results = [
            SELECT SUM(Price__c) total
            FROM Order__c
            WHERE Farmer__c = :farmerId
        ];

        if (results.isEmpty() || results[0].get('total') == null) {
```



Developer Console - Google Chrome
orgfarm-223d261061-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
OrderHelper.apxc
Code Coverage: None
API Version: 64
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```
     */
    public static Map<Id, Decimal> getTotalSalesForFarmers(Set<Id> farmerIds) {
        Map<Id, Decimal> result = new Map<Id, Decimal>();
        if (farmerIds == null || farmerIds.isEmpty()) return result;

        // Aggregate SOQL grouped by Farmer__c
        for (AggregateResult ar : [
            SELECT Farmer__c farmer, SUM(Price__c) total
            FROM Order__c
            WHERE Farmer__c IN :farmerIds
            GROUP BY Farmer__c
        ]) {
            Id fId = (Id) ar.get('farmer');
            Decimal sumVal = (Decimal) ar.get('total');
            result.put(fId, sumVal == null ? 0 : sumVal);
        }

```



Developer Console - Google Chrome
orgfarm-223d261061-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
OrderHelper.apxc
Code Coverage: None
API Version: 64
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

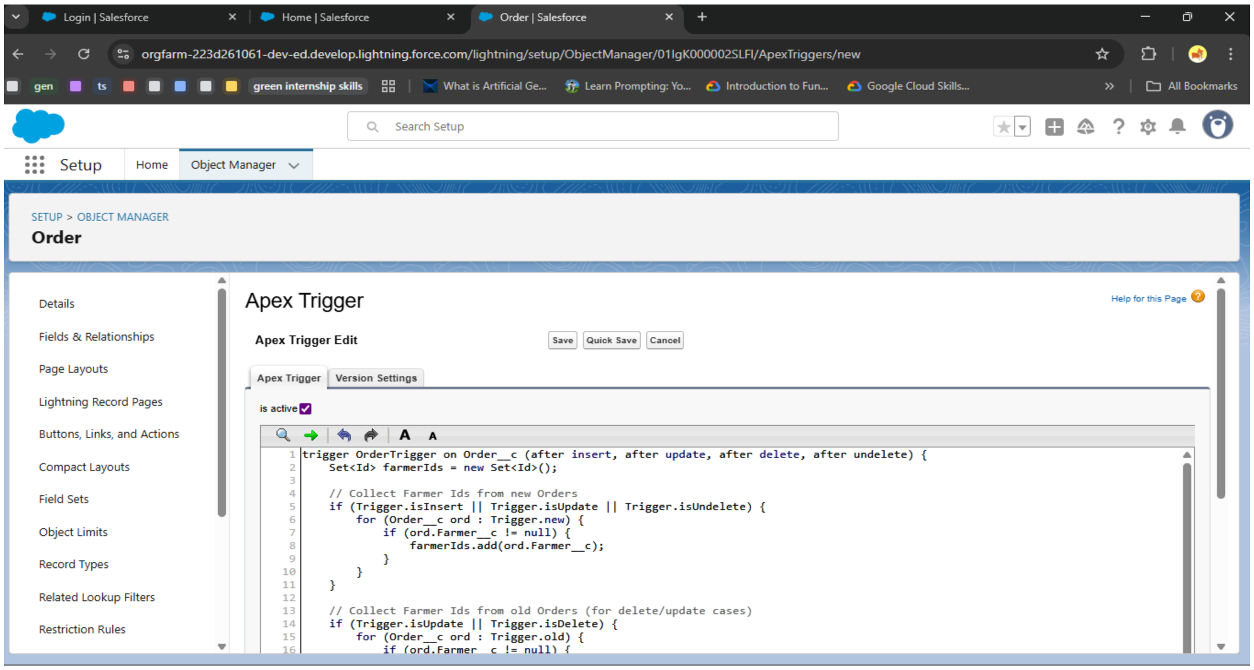
```
        ]) {
            Id fId = (Id) ar.get('farmer');
            Decimal sumVal = (Decimal) ar.get('total');
            result.put(fId, sumVal == null ? 0 : sumVal);
        }

        // Ensure all farmerIds are included (0 if no orders exist)
        for (Id f : farmerIds) {
            if (!result.containsKey(f)) {
                result.put(f, 0);
            }
        }

        return result;
    }
}

```

Apex Triggers & Trigger Design Pattern

Purpose: Automate recalculation of Farmer income whenever Orders change.

Implementation Steps:

1. Created `OrderTrigger` on `Order__c`.
2. Runs on **after insert, update, delete, undelete**.
3. Uses `OrderHelper` to calculate income.
4. Follows **one trigger per object** best practice.



orgfarm-223d261061-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01IgK000002SLFI/ApexTriggers/new

SETUP > OBJECT MANAGER

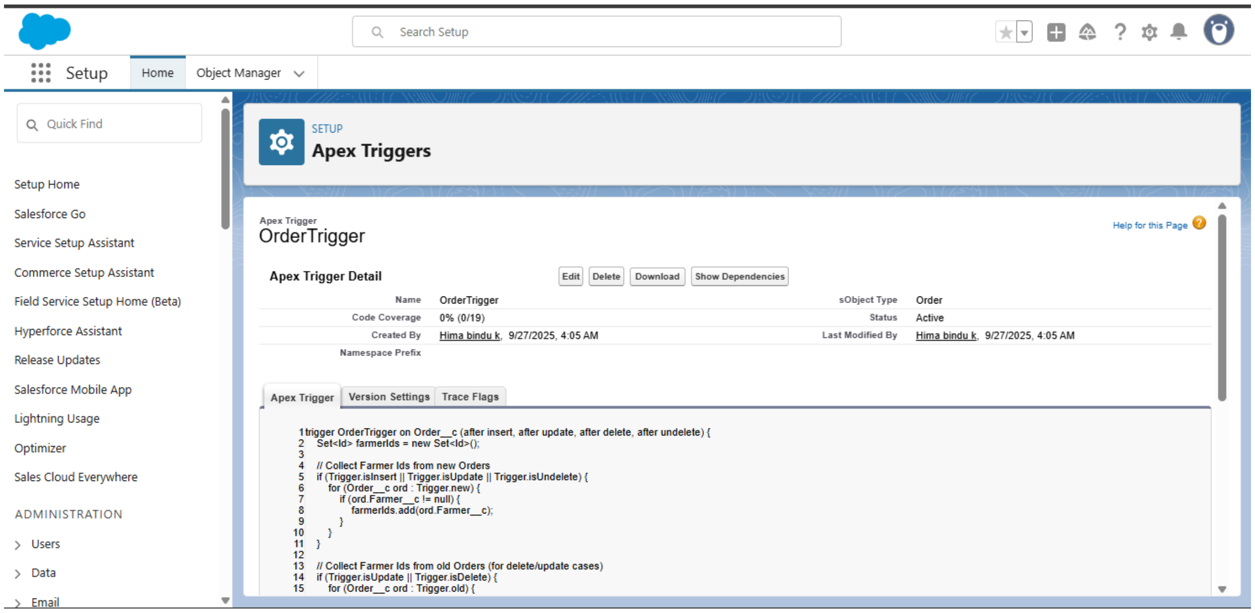
Order

Apex Trigger

Apex Trigger Edit

is active ✓

```
trigger OrderTrigger on Order__c (after insert, after update, after delete, after undelete) {
    Set<Id> farmerIds = new Set<Id>();

    // Collect Farmer Ids from new Orders
    if (Trigger.isInsert || Trigger.isUpdate || Trigger.isUndelete) {
        for (Order__c ord : Trigger.new) {
            if (ord.Farmer__c != null) {
                farmerIds.add(ord.Farmer__c);
            }
        }
    }

    // Collect Farmer Ids from old Orders (for delete/update cases)
    if (Trigger.isUpdate || Trigger.isDelete) {
        for (Order__c ord : Trigger.old) {
            if (ord.Farmer__c != null) {
```



SETUP

Apex Triggers

Apex Trigger

OrderTrigger

Apex Trigger Detail

Name	OrderTrigger	sObject Type	Order
Code Coverage	0% (0/19)	Status	Active
Created By	Hima bindu k, 9/27/2025, 4:05 AM	Last Modified By	Hima bindu k, 9/27/2025, 4:05 AM
Namespace Prefix			

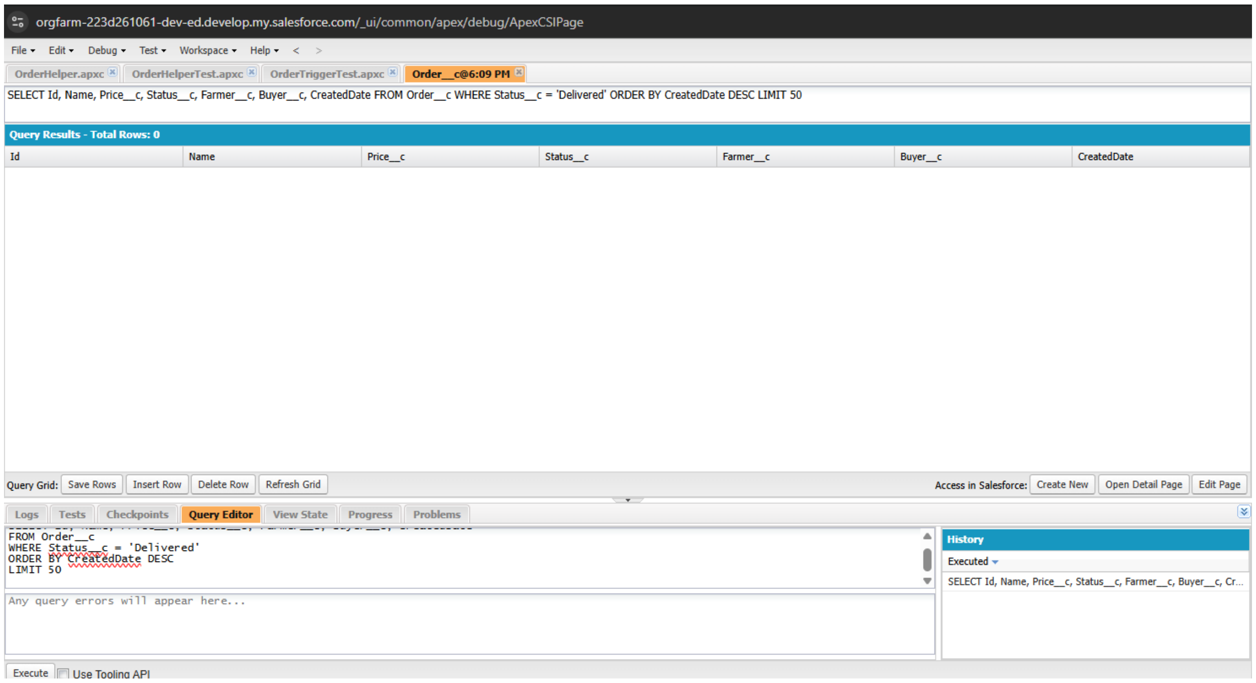
```
trigger OrderTrigger on Order__c (after insert, after update, after delete, after undelete) {
    Set<Id> farmerIds = new Set<Id>();

    // Collect Farmer Ids from new Orders
    if (Trigger.isInsert || Trigger.isUpdate || Trigger.isUndelete) {
        for (Order__c ord : Trigger.new) {
            if (ord.Farmer__c != null) {
                farmerIds.add(ord.Farmer__c);
            }
        }
    }

    // Collect Farmer Ids from old Orders (for delete/update cases)
    if (Trigger.isUpdate || Trigger.isDelete) {
        for (Order__c ord : Trigger.old) {
```

SOQL & SOSL Queries

Purpose: Retrieve data efficiently from Salesforce objects.

Implementation Steps:

- SOQL example: Fetch all delivered Orders.
- SOSL example: Search Crops by keyword.


orgfarm-223d261061-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
OrderHelper.apxc
OrderHelperTest.apxc
OrderTriggerTest.apxc
Order__c@6:09 PM
Query Grid: Save Rows Insert Row Delete Row Refresh Grid
Access in Salesforce: Create New Open Detail Page Edit Page
Logs Tests Checkpoints Query Editor View State Progress Problems
History
Executed
Execute
Use Tooling API

```
SELECT Id, Name, Price__c, Status__c, Farmer__c, Buyer__c, CreatedDate FROM Order__c WHERE Status__c = 'Delivered' ORDER BY CreatedDate DESC LIMIT 50
```

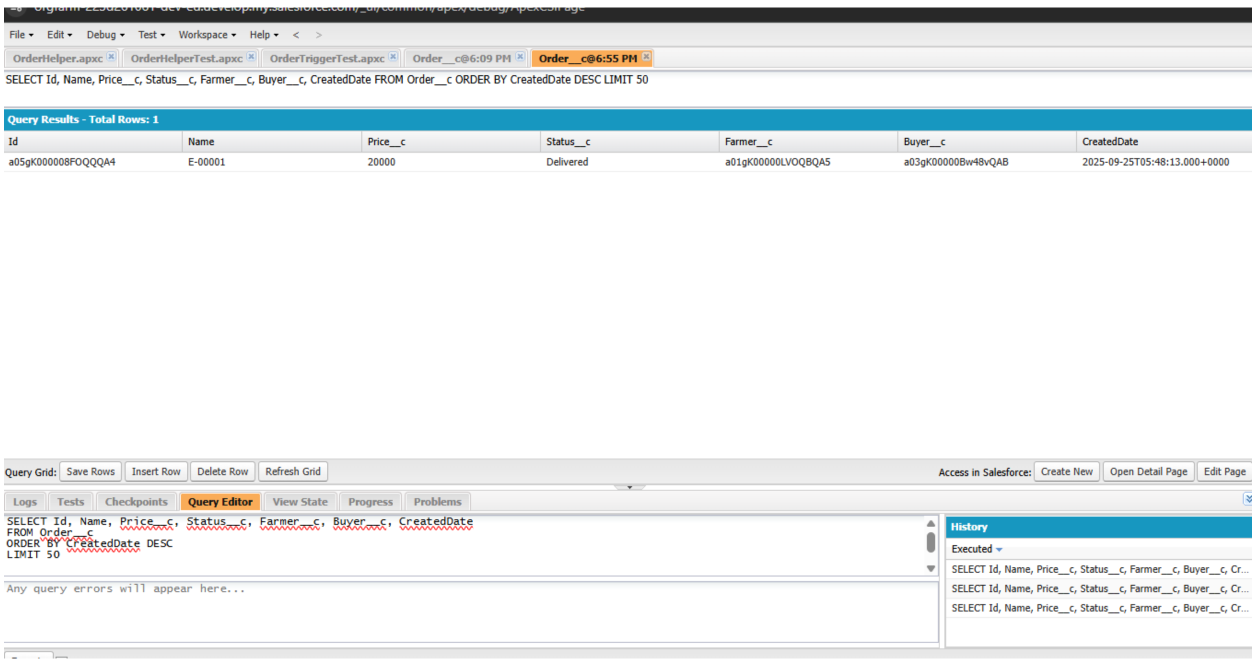
Query Results - Total Rows: 0

Id	Name	Price__c	Status__c	Farmer__c	Buyer__c	CreatedDate

```
FROM Order__c
WHERE Status__c = 'Delivered'
ORDER BY CreatedDate DESC
LIMIT 50
```

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

```
SELECT Id, Name, Price__c, Status__c, Farmer__c, Buyer__c, Cr...
```


File Edit Debug Test Workspace Help
OrderHelper.apxc
OrderHelperTest.apxc
OrderTriggerTest.apxc
Order__c@6:09 PM
Order__c@6:55 PM
Query Grid: Save Rows Insert Row Delete Row Refresh Grid
Access in Salesforce: Create New Open Detail Page Edit Page
Logs Tests Checkpoints Query Editor View State Progress Problems
History
Executed

```
SELECT Id, Name, Price__c, Status__c, Farmer__c, Buyer__c, CreatedDate FROM Order__c ORDER BY CreatedDate DESC LIMIT 50
```

Query Results - Total Rows: 1

Id	Name	Price__c	Status__c	Farmer__c	Buyer__c	CreatedDate
a05gK000008FOQQQA4	E-00001	20000	Delivered	a01gK00000LVOQBQA5	a03gK00000Bw48vQAB	2025-09-25T05:48:13.000+0000

```
SELECT Id, Name, Price__c, Status__c, Farmer__c, Buyer__c, CreatedDate
FROM Order__c
ORDER BY CreatedDate DESC
LIMIT 50
```

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

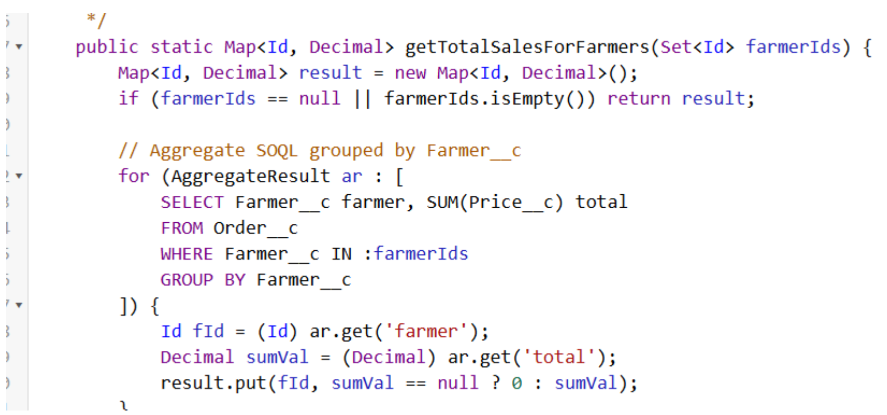
```
SELECT Id, Name, Price__c, Status__c, Farmer__c, Buyer__c, Cr...
SELECT Id, Name, Price__c, Status__c, Farmer__c, Buyer__c, Cr...
SELECT Id, Name, Price__c, Status__c, Farmer__c, Buyer__c, Cr...
```

Collections & Control Statements

Purpose: Store and process bulk data without hitting limits.

Implementation Steps:

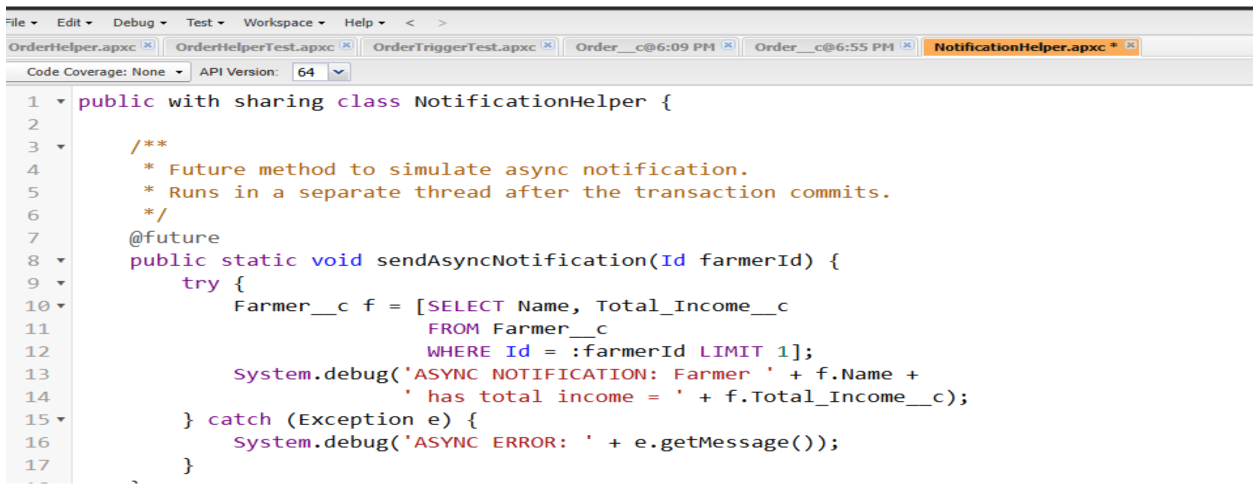
- **List** → store Orders.
- **Set** → collect unique Farmer Ids.
- **Map** → relate Farmer Id → Total Income.
- Used loops and IF statements for conditions.



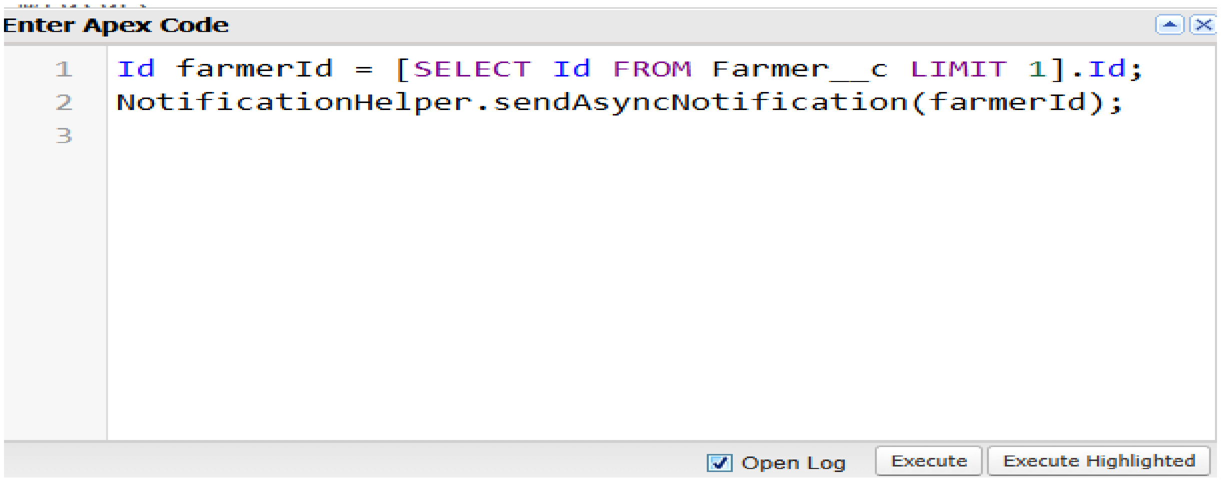
```
     */
    public static Map<Id, Decimal> getTotalSalesForFarmers(Set<Id> farmerIds) {
        Map<Id, Decimal> result = new Map<Id, Decimal>();
        if (farmerIds == null || farmerIds.isEmpty()) return result;

        // Aggregate SOQL grouped by Farmer__c
        for (AggregateResult ar : [
            SELECT Farmer__c farmer, SUM(Price__c) total
            FROM Order__c
            WHERE Farmer__c IN :farmerIds
            GROUP BY Farmer__c
        ]) {
            Id fId = (Id) ar.get('farmer');
            Decimal sumVal = (Decimal) ar.get('total');
            result.put(fId, sumVal == null ? 0 : sumVal);
```

Asynchronous Apex

- Developer Console → NotificationHelper class code.
- Screenshot of the **Future Method code**.
- Execute Anonymous → Run `NotificationHelper.sendAsyncNotification(farmerId).`
- Screenshot of the **Debug Log** output.



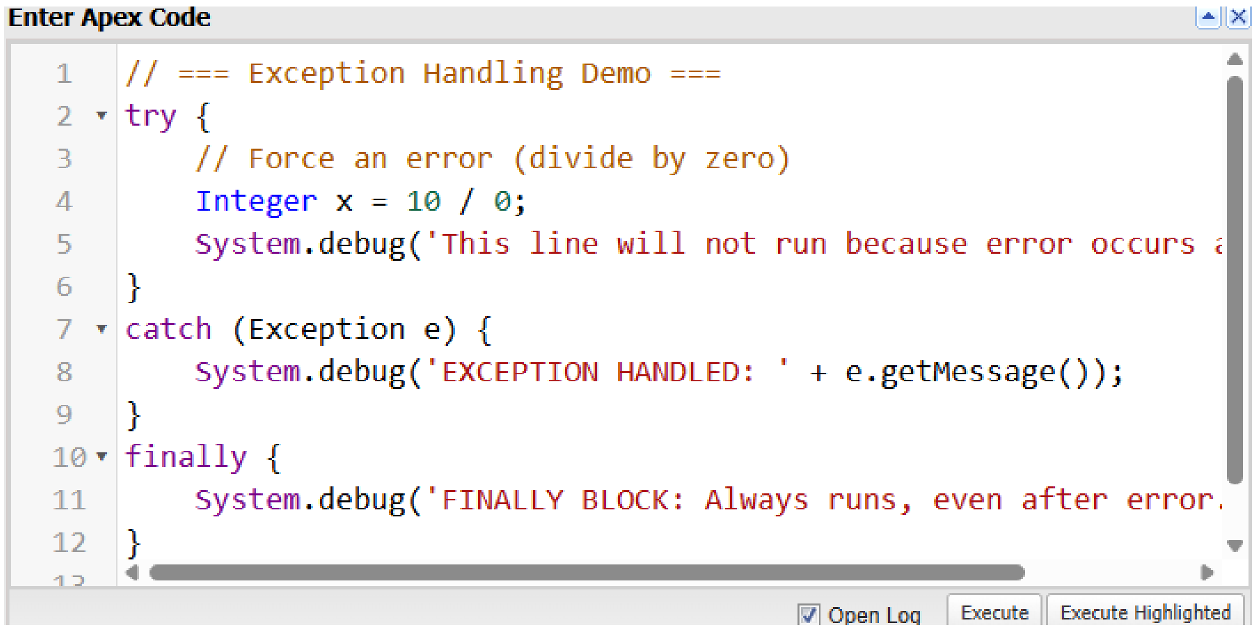
```
public with sharing class NotificationHelper {

    /**
     * Future method to simulate async notification.
     * Runs in a separate thread after the transaction commits.
     */
    @future
    public static void sendAsyncNotification(Id farmerId) {
        try {
            Farmer__c f = [SELECT Name, Total_Income__c
                           FROM Farmer__c
                           WHERE Id = :farmerId LIMIT 1];
            System.debug('ASYNC NOTIFICATION: Farmer ' + f.Name +
                         ' has total income = ' + f.Total_Income__c);
        } catch (Exception e) {
            System.debug('ASYNC ERROR: ' + e.getMessage());
        }
```


Enter Apex Code

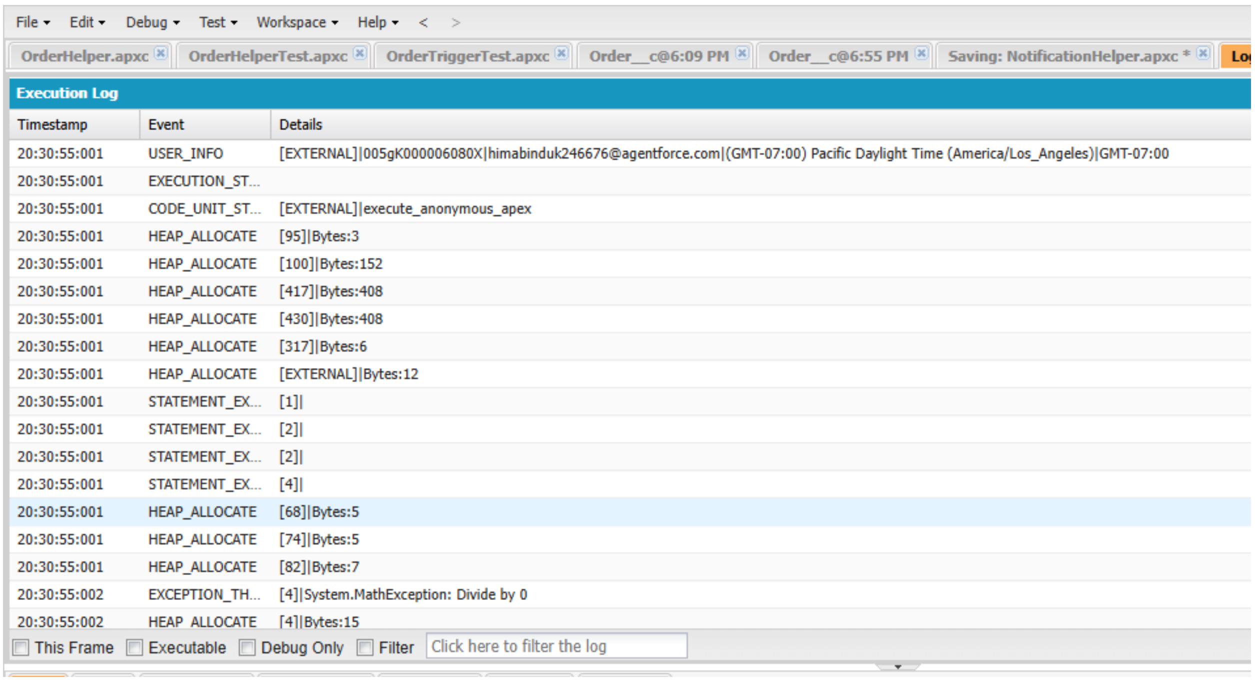
```
Id farmerId = [SELECT Id FROM Farmer__c LIMIT 1].Id;
NotificationHelper.sendAsyncNotification(farmerId);
```

Exception Handling

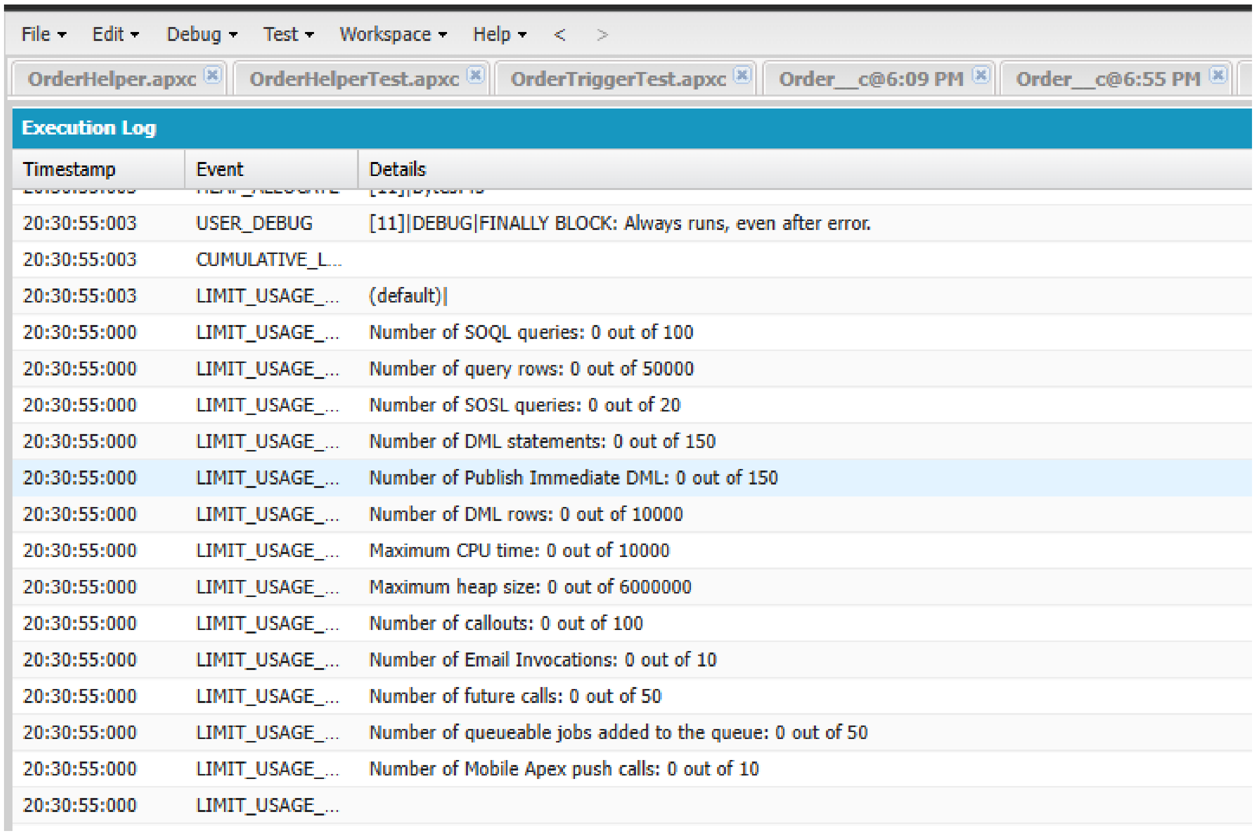
- **Purpose:** Ensure errors are caught gracefully without breaking execution.
- **Implementation Steps:**
    1. Added try-catch blocks in helper methods.
    2. Logged errors using `System.debug()`.
    3. Displayed user-friendly error messages when needed.


Enter Apex Code

```
// === Exception Handling Demo ===
try {
    // Force an error (divide by zero)
    Integer x = 10 / 0;
    System.debug('This line will not run because error occurs a
}
catch (Exception e) {
    System.debug('EXCEPTION HANDLED: ' + e.getMessage());
}
finally {
    System.debug('FINALLY BLOCK: Always runs, even after error.
}
```



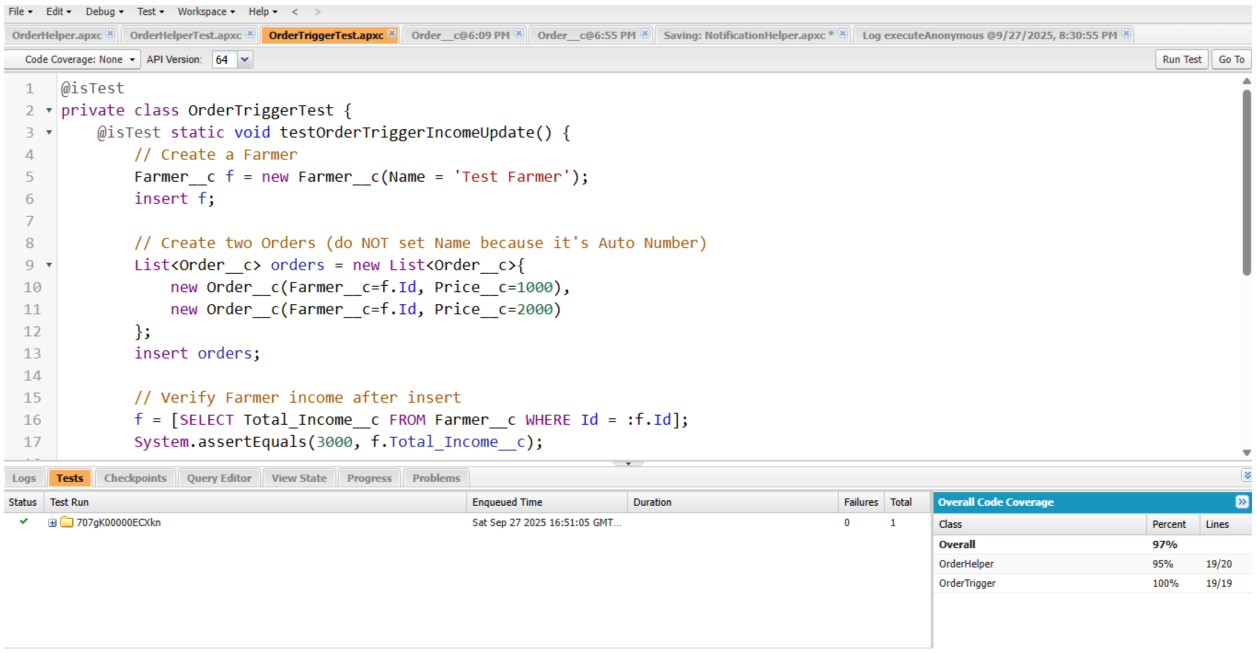
```
Timestamp      Event            Details
20:30:55:001   USER_INFO        [EXTERNAL]|005gK000006080X|himabinduk246676@agentforce.com|(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)|GMT-07:00
20:30:55:001   EXECUTION_ST...
20:30:55:001   CODE_UNIT_ST...  [EXTERNAL]|execute_anonymous_apex
20:30:55:001   HEAP_ALLOCATE    [95]|Bytes:3
20:30:55:001   HEAP_ALLOCATE    [100]|Bytes:152
20:30:55:001   HEAP_ALLOCATE    [417]|Bytes:408
20:30:55:001   HEAP_ALLOCATE    [430]|Bytes:408
20:30:55:001   HEAP_ALLOCATE    [317]|Bytes:6
20:30:55:001   HEAP_ALLOCATE    [EXTERNAL]|Bytes:12
20:30:55:001   STATEMENT_EX...  [1]|
20:30:55:001   STATEMENT_EX...  [2]|
20:30:55:001   STATEMENT_EX...  [2]|
20:30:55:001   STATEMENT_EX...  [4]|
20:30:55:001   HEAP_ALLOCATE    [68]|Bytes:5
20:30:55:001   HEAP_ALLOCATE    [74]|Bytes:5
20:30:55:001   HEAP_ALLOCATE    [82]|Bytes:7
20:30:55:002   EXCEPTION_TH...  [4]|System.MathException: Divide by 0
20:30:55:002   HEAP_ALLOCATE    [4]|Bytes:15
```



```
Timestamp      Event            Details
20:30:55:003   USER_DEBUG       [11]|DEBUG|FINALLY BLOCK: Always runs, even after error.
20:30:55:003   CUMULATIVE_L...
20:30:55:003   LIMIT_USAGE_...  (default)|
20:30:55:000   LIMIT_USAGE_...  Number of SOQL queries: 0 out of 100
20:30:55:000   LIMIT_USAGE_...  Number of query rows: 0 out of 50000
20:30:55:000   LIMIT_USAGE_...  Number of SOSL queries: 0 out of 20
20:30:55:000   LIMIT_USAGE_...  Number of DML statements: 0 out of 150
20:30:55:000   LIMIT_USAGE_...  Number of Publish Immediate DML: 0 out of 150
20:30:55:000   LIMIT_USAGE_...  Number of DML rows: 0 out of 10000
20:30:55:000   LIMIT_USAGE_...  Maximum CPU time: 0 out of 10000
20:30:55:000   LIMIT_USAGE_...  Maximum heap size: 0 out of 6000000
20:30:55:000   LIMIT_USAGE_...  Number of callouts: 0 out of 100
20:30:55:000   LIMIT_USAGE_...  Number of Email Invocations: 0 out of 10
20:30:55:000   LIMIT_USAGE_...  Number of future calls: 0 out of 50
20:30:55:000   LIMIT_USAGE_...  Number of queueable jobs added to the queue: 0 out of 50
20:30:55:000   LIMIT_USAGE_...  Number of Mobile Apex push calls: 0 out of 10
20:30:55:000   LIMIT_USAGE_...
```

Test Classes

- **Purpose:** Ensure Apex code works correctly and achieves required test coverage.
- **Implementation Steps:**
    1. Created `OrderTriggerTest` test class.
    2. Inserted sample Farmer + Order.
    3. Verified Farmer.Total_Income__c updates correctly.
    4. Achieved >75% test coverage.

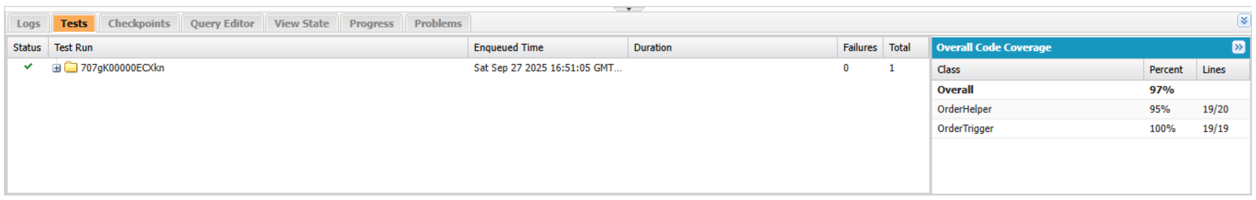


```
@isTest
private class OrderTriggerTest {
    @isTest static void testOrderTriggerIncomeUpdate() {
        // Create a Farmer
        Farmer__c f = new Farmer__c(Name = 'Test Farmer');
        insert f;

        // Create two Orders (do NOT set Name because it's Auto Number)
        List<Order__c> orders = new List<Order__c>{
            new Order__c(Farmer__c=f.Id, Price__c=1000),
            new Order__c(Farmer__c=f.Id, Price__c=2000)
        };
        insert orders;

        // Verify Farmer income after insert
        f = [SELECT Total_Income__c FROM Farmer__c WHERE Id = :f.Id];
        System.assertEquals(3000, f.Total_Income__c);
```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	707gK00000ECXkn	Sat Sep 27 2025 16:51:05 GMT...		0	1

Overall Code Coverage

Class	Percent	Lines
Overall	**97%**	
OrderHelper	95%	19/20
OrderTrigger	100%	19/19



Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	707gK00000ECXkn	Sat Sep 27 2025 16:51:05 GMT...		0	1

Overall Code Coverage

Class	Percent	Lines
Overall	**97%**	
OrderHelper	95%	19/20
OrderTrigger	100%	19/19

Outcome

Phase 5 successfully demonstrated how Apex extends Salesforce functionality beyond declarative automation. With Classes, Triggers, SOQL, Collections, and Asynchronous Apex, GreenHarvest now supports complex, scalable business logic. Test Classes ensure quality and maintainability.