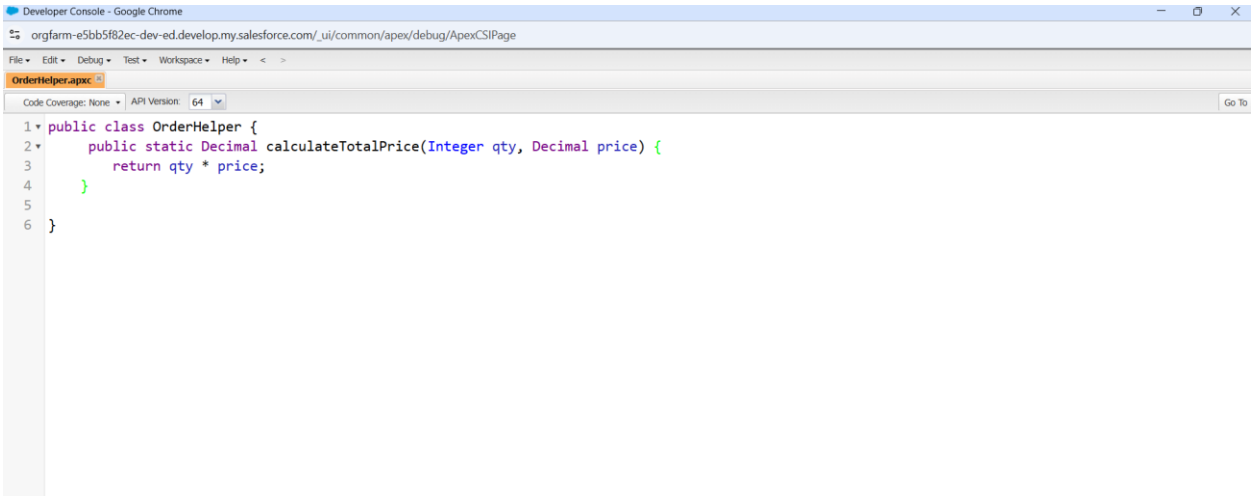
Phase 5 – Apex Programming (Complete Guide)

This guide explains all Apex concepts required for **Phase 5** of the Farmers Market & Supply Chain Project. It combines Classes, Triggers, SOQL, SOSL, Collections, Control Statements, Batch Apex, Queueable Apex, Scheduled Apex, Future Methods, Exception Handling, and Test Classes.

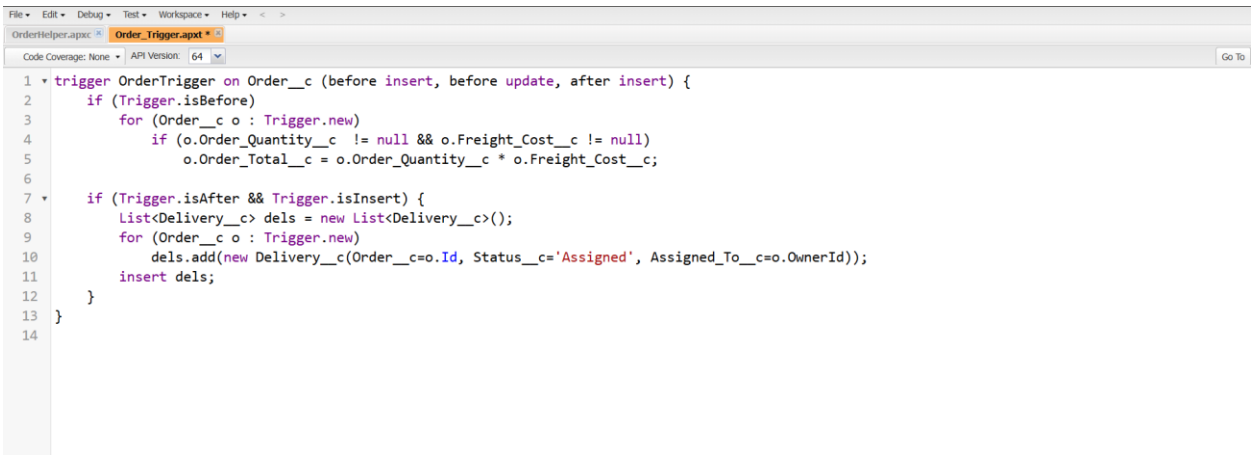
◇ 1) Classes & Objects

- Apex classes group reusable logic and methods.
-



◇ 2) Apex Triggers

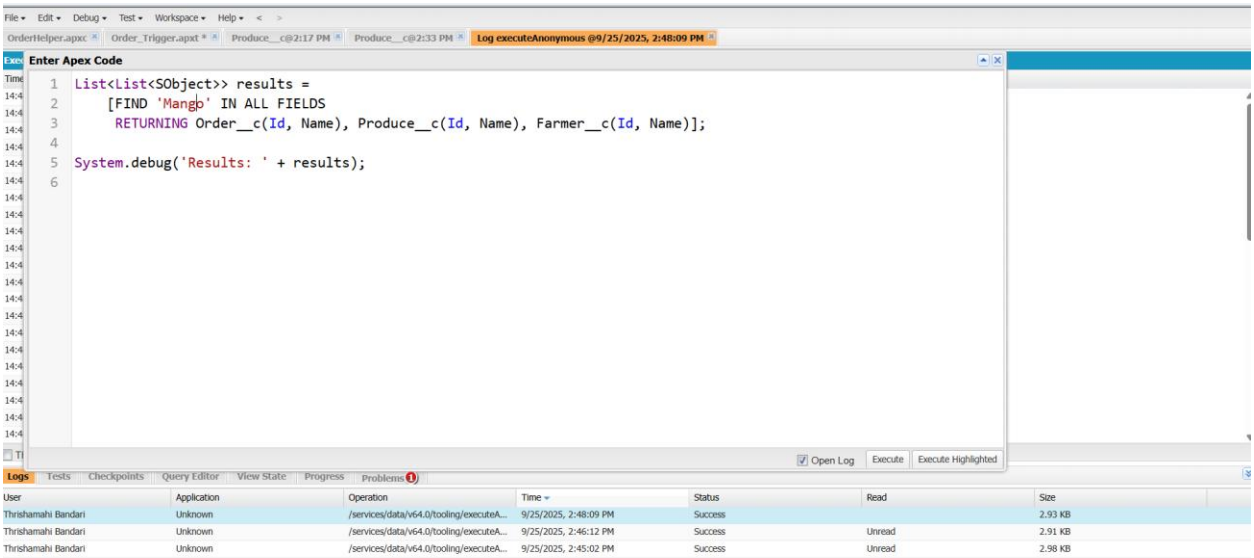
- Execute logic before/after DML events (insert, update, delete).



◇ 3) SOQL (Salesforce Object Query Language)

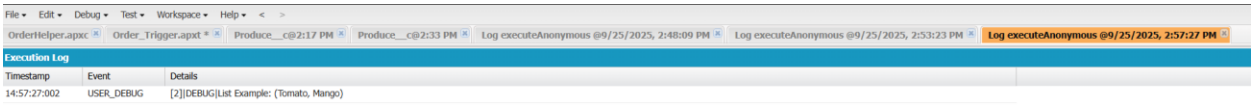
- Query data from Salesforce.

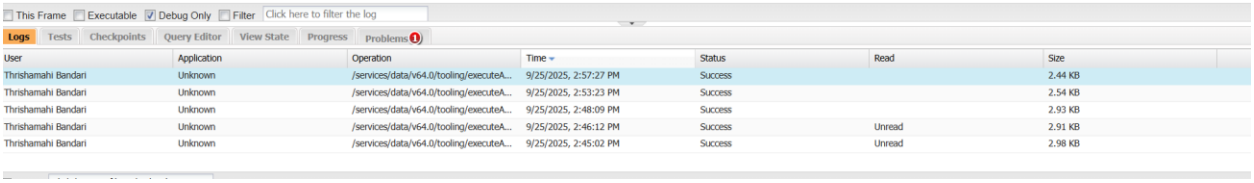
Example:



◇ 4) SOSL (Salesforce Object Search Language)

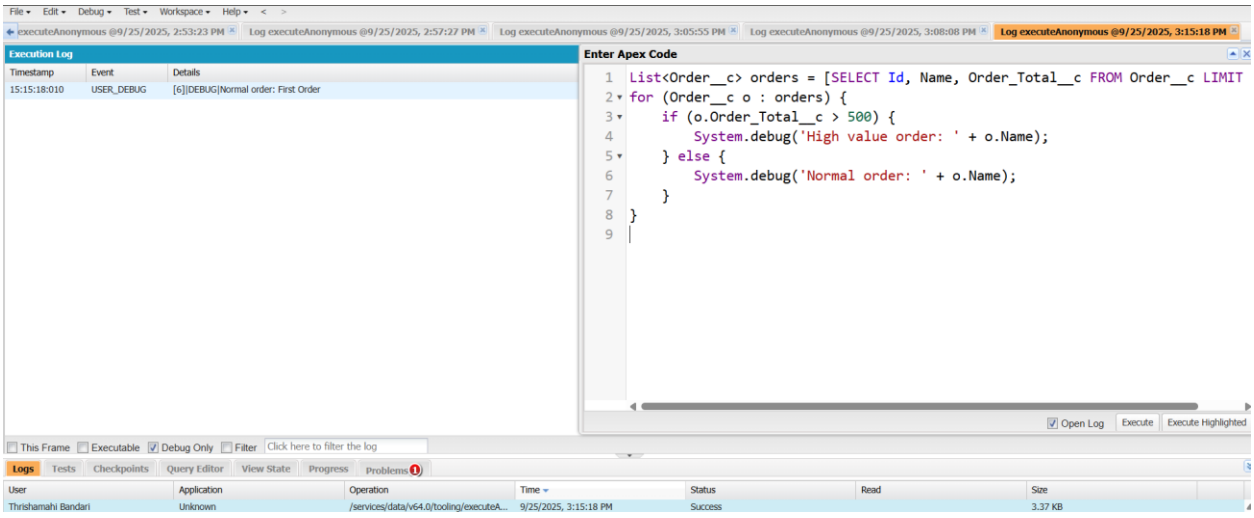
- Search text across multiple objects.



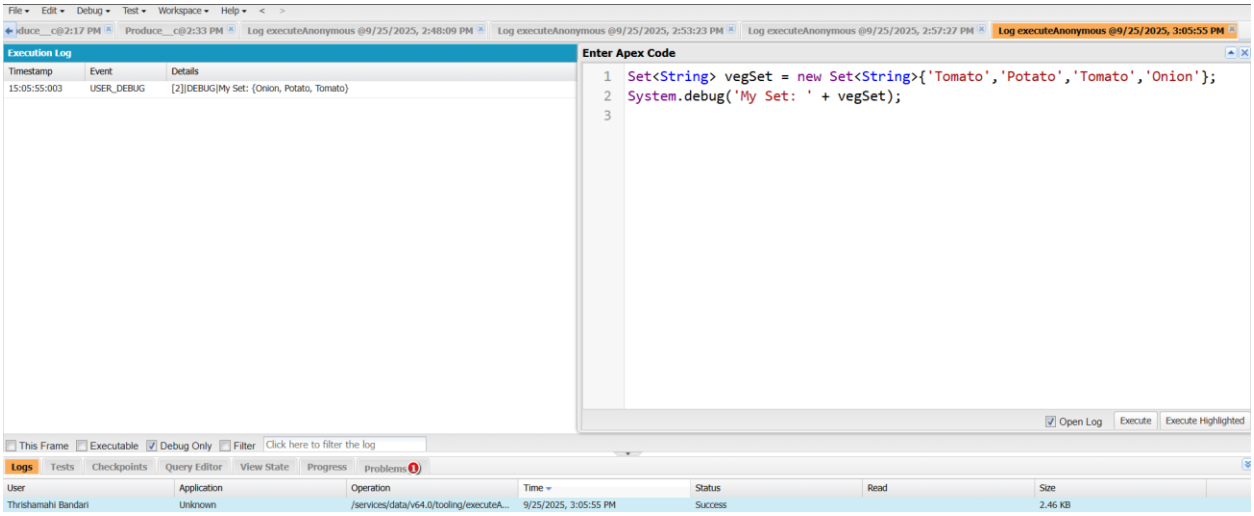


◇ 5) Collections: List, Set, Map

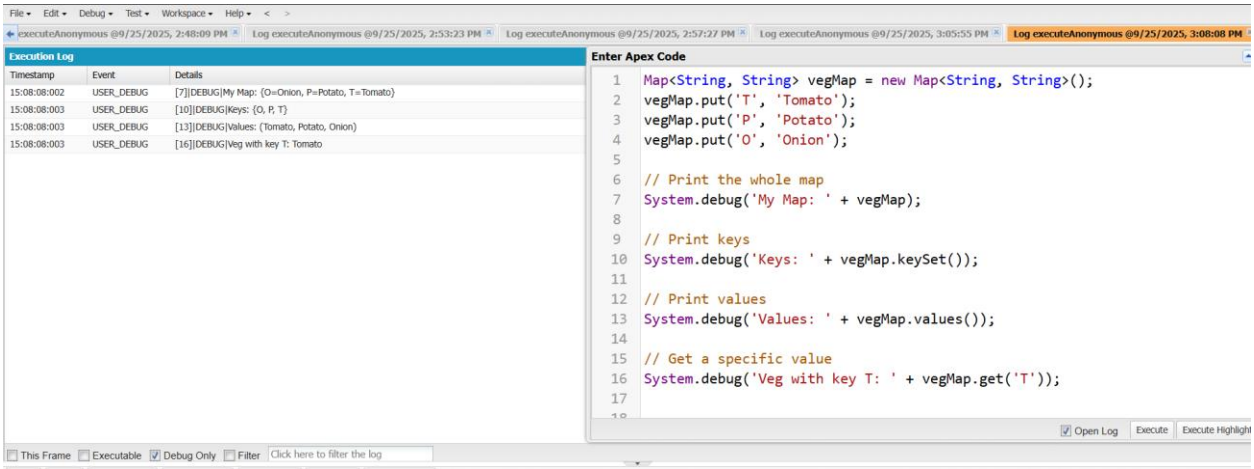
List:



Set:



Map:



◇ 6) Control Statements

- Core logic flow: if, else, for, while.

Example:

```
for (Order__c o : [SELECT Id, Order_Total__c FROM Order__c]) {
    if (o.Order_Total__c > 1000) {
        System.debug('High value order: ' + o.Name);
    }
}
```

◇ 7) Batch Apex

- Handles **large volumes of records** in chunks.
- Useful for stock updates or recalculations in the Farmers Market project.

◇ 8) Queueable Apex

- For **asynchronous background processing**.
- Similar to Batch but simpler, supports job chaining.
- **Use Case:** Send order details to logistics system.

◇ 9) Scheduled Apex

- Runs code at specific times (daily, weekly, etc.).
- **Use Case:** Nightly job to notify farmers of low stock.
- Configurable via **Setup → Scheduled Jobs**.

◇ 10) Future Methods

- Lightweight async operations, annotated with `@future`.
- Good for callouts or quick notifications.
- **Use Case:** Notify farmer by email after order confirmation.

◇ 11) Exception Handling

- Apex supports `try-catch-finally`.
- Prevents entire transactions from failing.
- **Use Case:** Catch errors when updating Produce stock.

◇ 12) Test Classes

- Required for deployment (≥75% coverage).
- Written with `@isTest`.
- Validate outcomes using `System.assert()`.
- **Use Case:** Ensure Order trigger calculates totals correctly and creates Delivery records.

Example:

```
@isTest
public class OrderTriggerTest {
    @isTest static void testOrderTotal() {
        Order__c o = new Order__c(Name='Test Order', Order_Quantity__c=10,
Unit_Price__c=5, Freight_Cost__c=5);
        insert o;
        System.assertEquals(55, o.Order_Total__c);
    }
}
```