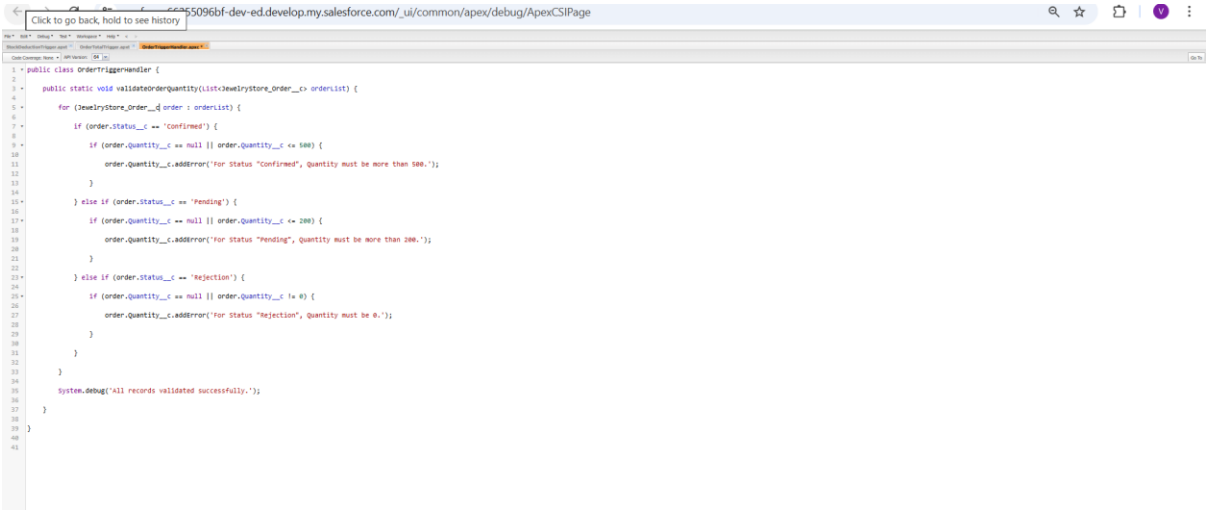
Phase 5: Apex Programming (Developer)

Project:Jewellery Store Management CRM

Step 1: Classes & Objects

- **Apex is object-oriented**: It allows the creation of **classes** (blueprints) and **objects** (instances) to organize and reuse business logic.

- In this project, a **Trigger Handler Class** approach was implemented to follow best practices.



Source Code:

```
public class OrderTriggerHandler {

   public static void validateOrderQuantity(List<JewelryStore_Order__c> orderList) {

      for (JewelryStore_Order__c order : orderList) {

         if (order.Status__c == 'Confirmed') {

            if (order.Quantity__c == null || order.Quantity__c <= 500) {

               order.Quantity__c.addError('For Status "Confirmed", Quantity must be more than
500.');

            }

         } else if (order.Status__c == 'Pending') {

            if (order.Quantity__c == null || order.Quantity__c <= 200) {

           order.Quantity__c.addError('For Status "Pending", Quantity must be more than
200.');
```

```
            }
        } else if (order.Status__c == 'Rejection') {
            if (order.Quantity__c == null || order.Quantity__c != 0) {
                order.Quantity__c.addError('For Status "Rejection", Quantity must be 0.');
            }
        }
    }
    System.debug('All records validated successfully.');
  }
}
```

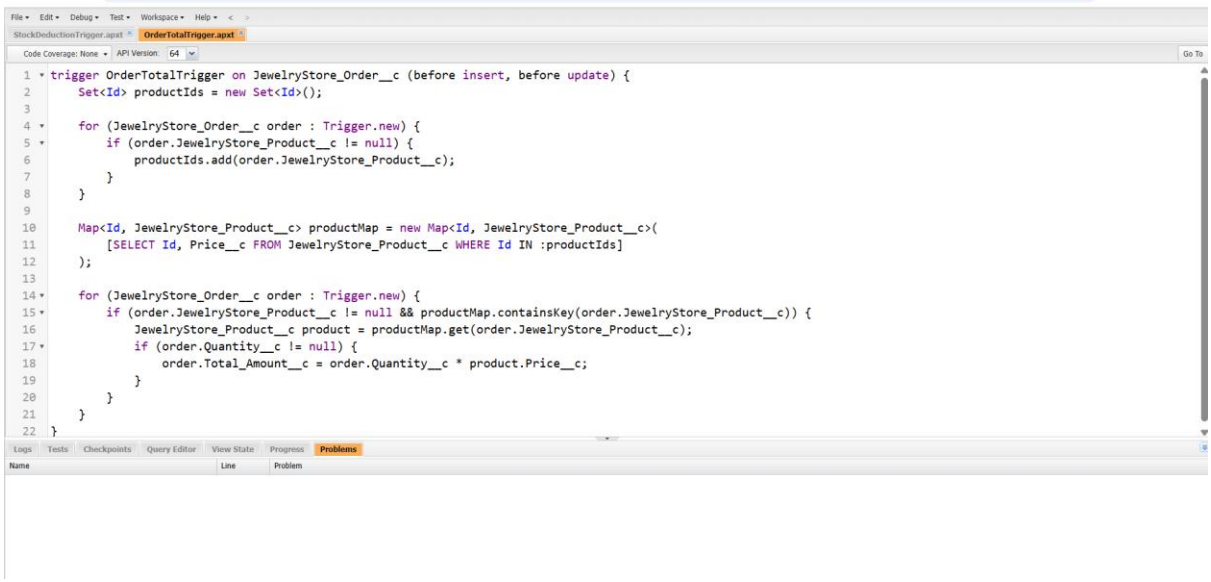
Step 2:Apex Triggers (before/after insert/update/delete)

In this project, two custom Apex triggers were implemented to automate critical processes in the Golden Era Enterprises CRM:

a) Order Total Trigger

Purpose:

- Automatically calculate the **Total Price** for each order.
- Formula: **Quantity × Product Price**



```
trigger OrderTotalTrigger on JewelryStore_Order__c (before insert, before update) {
    Set<Id> productIds = new Set<Id>();

    for (JewelryStore_Order__c order : Trigger.new) {
        if (order.JewelryStore_Product__c != null) {
            productIds.add(order.JewelryStore_Product__c);
        }
    }

    Map<Id, JewelryStore_Product__c> productMap = new Map<Id, JewelryStore_Product__c>(
        [SELECT Id, Price__c FROM JewelryStore_Product__c WHERE Id IN :productIds]
    );

    for (JewelryStore_Order__c order : Trigger.new) {
        if (order.JewelryStore_Product__c != null && productMap.containsKey(order.JewelryStore_Product__c)) {
            JewelryStore_Product__c product = productMap.get(order.JewelryStore_Product__c);
            if (order.Quantity__c != null) {
                order.Total_Amount__c = order.Quantity__c * product.Price__c;
            }
        }
    }
}
```

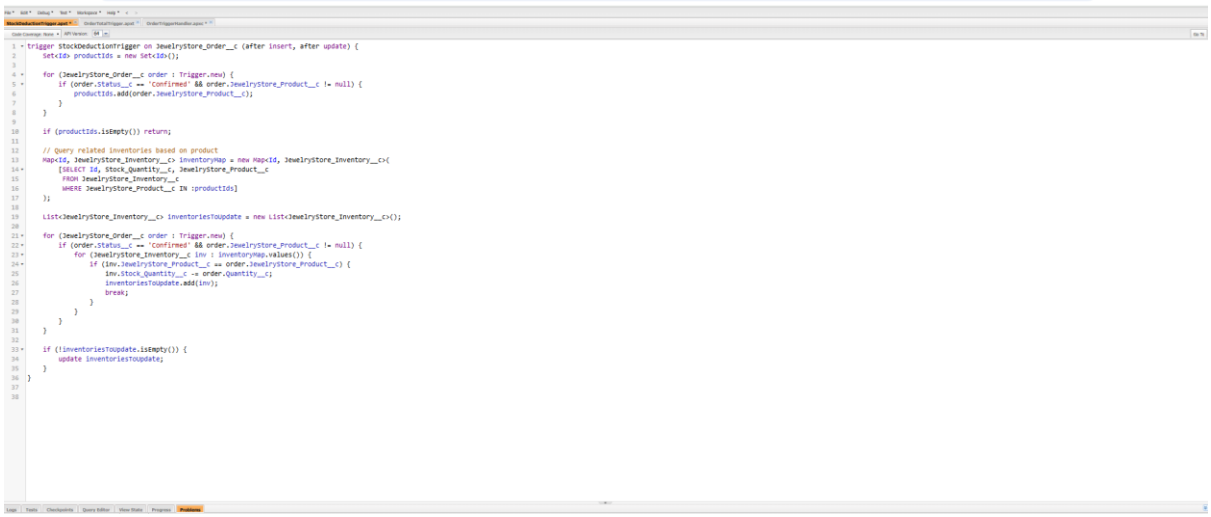
Source Code:

```
trigger OrderTotalTrigger on JewelryStore_Order__c (before insert, before update)
{

   Set<Id> productIds = new Set<Id>();

   for (JewelryStore_Order__c order : Trigger.new) {

      if (order.JewelryStore_Product__c != null) {

         productIds.add(order.JewelryStore_Product__c);

      }

   }

   Map<Id, JewelryStore_Product__c> productMap = new Map<Id,
JewelryStore_Product__c>(

      [SELECT Id, Price__c FROM JewelryStore_Product__c WHERE Id IN
:productIds]

   );

   for (JewelryStore_Order__c order : Trigger.new) {

      if (order.JewelryStore_Product__c != null &&
productMap.containsKey(order.JewelryStore_Product__c)) {

         JewelryStore_Product__c product =
productMap.get(order.JewelryStore_Product__c);

         if (order.Quantity__c != null) {

            order.Total_Amount__c = order.Quantity__c * product.Price__c;

         }

      }

   }

}
```

b) Stock Deduction Trigger

Purpose:

- Deduct stock from **Inventory** (or Product Stock Quantity) based on the order quantity.
- Example: If customer orders 2 rings, Inventory decreases by 2.



Source Code:

```
trigger StockDeductionTrigger on JewelryStore_Order__c (after insert, after update) {

    Set<Id> productIds = new Set<Id>();

    for (JewelryStore_Order__c order : Trigger.new) {

        if (order.Status__c == 'Confirmed' && order.JewelryStore_Product__c != null) {

            productIds.add(order.JewelryStore_Product__c);

        }

    }

    if (productIds.isEmpty()) return;

    // Query related inventories based on product

    Map<Id, JewelryStore_Inventory__c> inventoryMap = new Map<Id,
JewelryStore_Inventory__c>(
```

```
        [SELECT Id, Stock_Quantity__c, JewelryStore_Product__c
         FROM JewelryStore_Inventory__c
         WHERE JewelryStore_Product__c IN :productIds]
    );
    List<JewelryStore_Inventory__c> inventoriesToUpdate = new
List<JewelryStore_Inventory__c>();
    for (JewelryStore_Order__c order : Trigger.new) {
        if (order.Status__c == 'Confirmed' && order.JewelryStore_Product__c != null) {
            for (JewelryStore_Inventory__c inv : inventoryMap.values()) {
                if (inv.JewelryStore_Product__c == order.JewelryStore_Product__c) {
                    inv.Stock_Quantity__c -= order.Quantity__c;
                    inventoriesToUpdate.add(inv);
                    break;
                }
            }
        }
    }
    if (!inventoriesToUpdate.isEmpty()) {
        update inventoriesToUpdate;
    }
}
```