# PHASE 5 – Apex Programming (Developer)

In this phase, we extend beyond Salesforce’s point-and-click tools and use Apex programming to implement custom business logic for Deliveries, Orders, and Loyalty Points in the CRM. Apex ensures scalability, efficiency, and complex automation where declarative tools are insufficient.

## 1. Apex Classes & Objects

Apex classes were created to handle reusable logic.  
  
Example – Confirm Delivery

**public class DeliveryService {  
 public static void confirmDelivery(Id deliveryId) {  
 Delivery\_\_c d = [SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c WHERE Id = :deliveryId];  
 d.Delivery\_Status\_\_c = 'Confirmed';  
 update d;  
 }  
}**

## 2. Apex Triggers

Triggers automate processes when records are inserted, updated, or deleted.  
Example – Prevent Duplicate Delivery Numbers

trigger DeliveryTrigger on Delivery\_\_c (before insert, before update) {  
 for (Delivery\_\_c d : Trigger.new) {  
 Boolean exists = [  
 SELECT Id FROM Delivery\_\_c   
 WHERE Delivery\_Number\_\_c = :d.Delivery\_Number\_\_c   
 AND Id != :d.Id  
 LIMIT 1  
 ] != null;  
 if (exists) {  
 d.addError('This Delivery Number already exists.');  
 }  
 }  
}

## 3. Trigger Design Pattern

Best practice is to move logic into a handler class to keep triggers clean.

trigger DeliveryTrigger on Delivery\_\_c (before insert, before update) {  
 DeliveryHandler.beforeSave(Trigger.new, Trigger.oldMap);  
}  
  
public class DeliveryHandler {  
 public static void beforeSave(List<Delivery\_\_c> newList, Map<Id, Delivery\_\_c> oldMap) {  
 // custom validation logic here  
 }  
}

## 4. SOQL & SOSL

SOQL – Query Deliveries by Status

List<Delivery\_\_c> deliveries = [SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c WHERE Delivery\_Status\_\_c = 'Pending'];  
SOSL – Search Across Customers and OrdersList<List<SObject>> results =   
 [FIND 'Kavya\*' IN ALL FIELDS   
 RETURNING Customer\_\_c(Id, Name), Order\_\_c(Id, Status\_\_c)];

## 5. Collections (List, Set, Map)

List<String> orderIds = new List<String>();  
Set<Id> customerIds = new Set<Id>();  
Map<Id, Order\_\_c> orderMap = new Map<Id, Order\_\_c>([SELECT Id, Status\_\_c FROM Order\_\_c]);

## 6. Control Statements

for (Order\_\_c o : [SELECT Id, Status\_\_c FROM Order\_\_c]) {  
 if (o.Status\_\_c == 'Pending') {  
 o.Status\_\_c = 'Completed';  
 }  
}

## 7. Batch Apex

For large data processing (e.g., updating multiple deliveries).

global class UpdateDeliveryBatch implements Database.Batchable<SObject> {  
 global Database.QueryLocator start(Database.BatchableContext bc) {  
 return Database.getQueryLocator('SELECT Id, Delivery\_Status\_\_c FROM Delivery\_\_c');  
 }  
 global void execute(Database.BatchableContext bc, List<Delivery\_\_c> scope) {  
 for (Delivery\_\_c d : scope) {  
 d.Delivery\_Status\_\_c = 'Archived';  
 }  
 update scope;  
 }  
 global void finish(Database.BatchableContext bc) {}  
}

## 8. Queueable Apex

public class NotifyCustomerJob implements Queueable {  
 public void execute(QueueableContext context) {  
 // send email/notification about delivery update  
 }  
}

## 9. Scheduled Apex

global class DailyDeliveryReminder implements Schedulable {  
 global void execute(SchedulableContext sc) {  
 // send daily reminders for pending deliveries  
 }  
}

## 10. Future Methods

public class PaymentGateway {  
 @future(callout=true)  
 public static void processPayment(Id paymentId) {  
 // external API call for payment processing  
 }  
}

## 11. Exception Handling

try {  
 // risky logic  
} catch (Exception e) {  
 System.debug('Error: ' + e.getMessage());  
}

## 12. Test Classes

All Apex logic must be covered by test classes before deployment.

@isTest  
private class DeliveryTriggerTest {  
 @isTest static void testDuplicateDeliveryNumber() {  
 // insert deliveries and check addError  
 }  
}

## 13. Asynchronous Processing

Future → async callouts.  
  
Batch → large dataset processing (Deliveries, Orders).  
  
Queueable → notifications & chained jobs.  
  
Scheduled → reminders & daily updates.