# Phase 5 — Apex Programming (Developer) — Migrant Health CRM

## Summary

Apex classes, triggers and tests to: calculate patient age, prevent duplicates by Patient ID, auto-create tasks for case managers, and auto-create migrant records when appointments are completed.

## Prerequisites

Ensure these objects and fields exist before deploying code:

- Patient\_\_c: Patient\_ID\_\_c (Text), DOB\_\_c (Date), Age\_\_c (Number), Gender\_\_c (Picklist), Phone\_\_c (Phone), Email\_\_c (Email), Case\_Manager\_\_c (Lookup to User)  
- Appointment\_\_c: Patient\_\_c (Lookup), Appointment\_Date\_\_c (Date), Status\_\_c (Picklist: Scheduled, Completed)  
- Migrant\_Record\_Number\_\_c: Patient\_\_c (Lookup), Appointment\_\_c (Lookup), Status\_\_c (Picklist, must include 'Open'), Notes\_\_c (Long Text)

## Implementation Steps

1. Go to Setup → Apex Classes → New → Paste Apex Class code (below).

2. Go to Object Manager → Patient\_\_c → Triggers → New → Paste PatientTrigger code.

3. Go to Object Manager → Appointment\_\_c → Triggers → New → Paste AppointmentTrigger code.

4. Deploy handler classes and utility classes from below code sections.

5. Deploy the test class MigrantHealthTests.

6. Go to Setup → Apex Test Execution → Run All → Verify all tests pass.

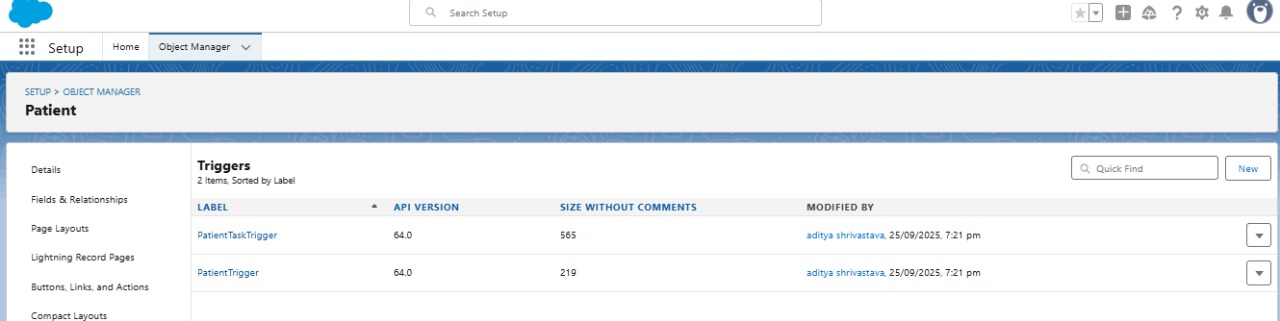
## Apex Code

### PatientUtils.cls

public class PatientUtils {  
 public static Integer calculateAge(Date dob) {  
 if (dob == null) return null;  
 Date today = Date.today();  
 Integer age = today.year() - dob.year();  
 if (today.month() < dob.month() || (today.month() == dob.month() && today.day() < dob.day())) {  
 age--;  
 }  
 return age;  
 }  
}

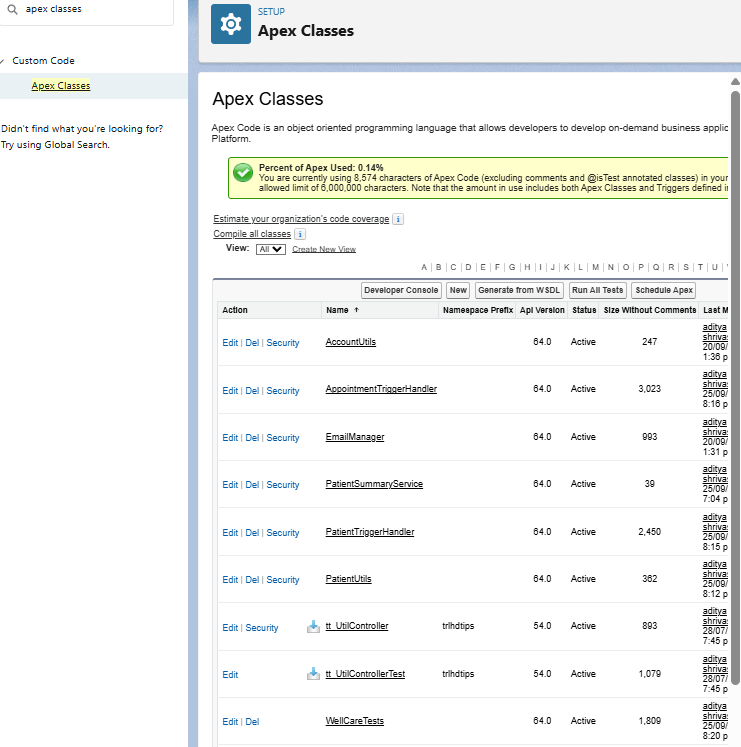
### PatientTrigger.trigger

trigger PatientTrigger on Patient\_\_c (before insert, before update, after insert) {  
 if (Trigger.isBefore) {  
 if (Trigger.isInsert) {  
 PatientTriggerHandler.beforeInsert(Trigger.new);  
 }  
 if (Trigger.isUpdate) {  
 PatientTriggerHandler.beforeUpdate(Trigger.new);  
 }  
 }  
 if (Trigger.isAfter && Trigger.isInsert) {  
 PatientTriggerHandler.afterInsert(Trigger.new);  
 }  
}



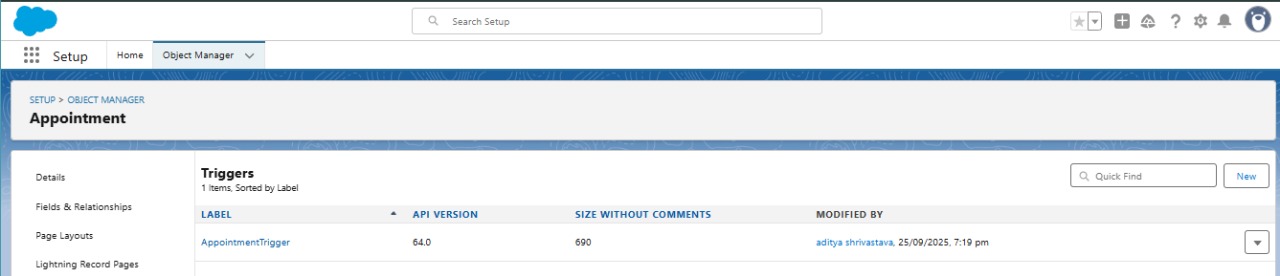
### PatientTriggerHandler.cls

public with sharing class PatientTriggerHandler {  
  
 public static void beforeInsert(List<Patient\_\_c> newPatients) {  
 calculateAgeForList(newPatients);  
 preventDuplicates(newPatients, null);  
 }  
  
 public static void beforeUpdate(List<Patient\_\_c> patients) {  
 calculateAgeForList(patients);  
 preventDuplicates(patients, null);  
 }  
  
 public static void afterInsert(List<Patient\_\_c> insertedPatients) {  
 List<Task> tasks = new List<Task>();  
 for (Patient\_\_c p : insertedPatients) {  
 if (p.Case\_Manager\_\_c != null) {  
 Task t = new Task();  
 t.OwnerId = p.Case\_Manager\_\_c;  
 t.WhatId = p.Id;  
 t.Subject = 'Follow up with new patient';  
 t.ActivityDate = Date.today().addDays(2);  
 t.Status = 'Not Started';  
 t.Priority = 'Normal';  
 tasks.add(t);  
 }  
 }  
 if (!tasks.isEmpty()) insert tasks;  
 }  
  
 private static void calculateAgeForList(List<Patient\_\_c> patients) {  
 for (Patient\_\_c p : patients) {  
 if (p.DOB\_\_c != null) {  
 p.Age\_\_c = PatientUtils.calculateAge(p.DOB\_\_c);  
 }  
 }  
 }  
  
 private static void preventDuplicates(List<Patient\_\_c> patients, Map<Id, Patient\_\_c> oldMap) {  
 Set<String> ids = new Set<String>();  
 for (Patient\_\_c p : patients) {  
 if (p.Patient\_ID\_\_c != null) {  
 ids.add(p.Patient\_ID\_\_c.trim().toLowerCase());  
 }  
 }  
 if (ids.isEmpty()) return;  
  
 Map<String, Patient\_\_c> existingByPid = new Map<String, Patient\_\_c>();  
 for (Patient\_\_c ex : [SELECT Id, Patient\_ID\_\_c FROM Patient\_\_c WHERE Patient\_ID\_\_c IN :ids]) {  
 existingByPid.put(ex.Patient\_ID\_\_c.trim().toLowerCase(), ex);  
 }  
  
 for (Patient\_\_c p : patients) {  
 if (p.Patient\_ID\_\_c == null) continue;  
 String key = p.Patient\_ID\_\_c.trim().toLowerCase();  
 if (existingByPid.containsKey(key)) {  
 Patient\_\_c existing = existingByPid.get(key);  
 if (p.Id == null || existing.Id != p.Id) {  
 p.addError('A Patient with this Patient ID already exists.');  
 }  
 }  
 }  
 }  
}



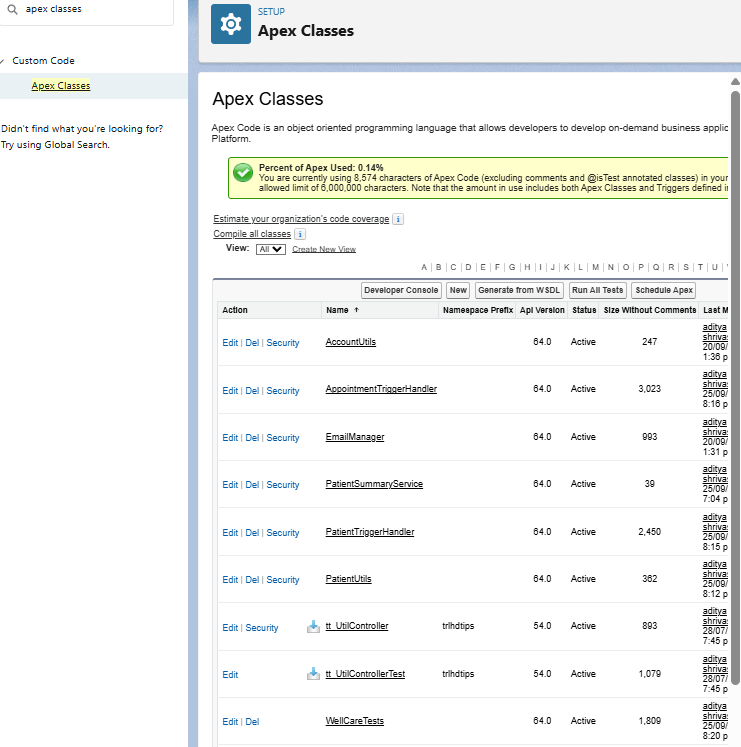
### AppointmentTrigger.trigger

trigger AppointmentTrigger on Appointment\_\_c (after insert, after update) {  
 if (Trigger.isAfter) {  
 if (Trigger.isInsert) {  
 AppointmentTriggerHandler.handleAfterInsert(Trigger.new);  
 }  
 if (Trigger.isUpdate) {  
 AppointmentTriggerHandler.handleAfterUpdate(Trigger.new, Trigger.oldMap);  
 }  
 }  
}



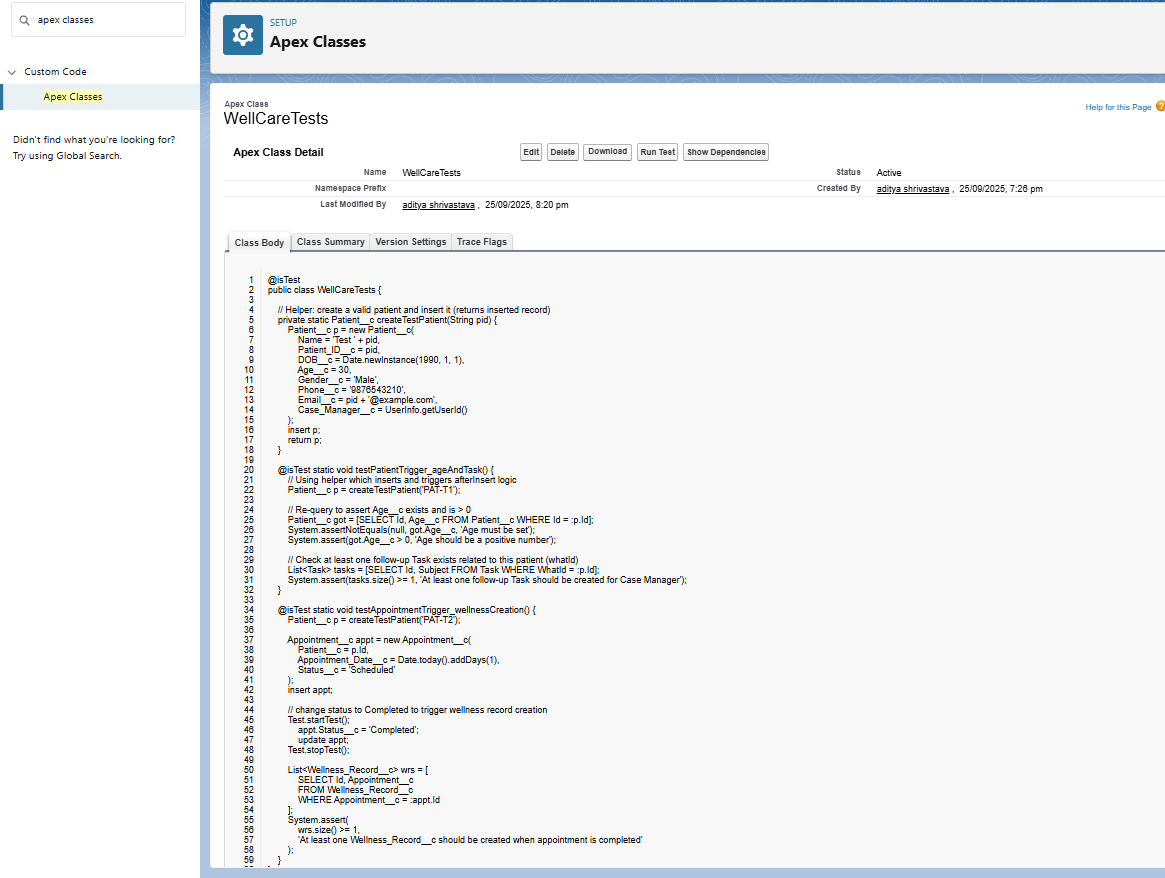
### AppointmentTriggerHandler.cls

public with sharing class AppointmentTriggerHandler {  
  
 public static void handleAfterInsert(List<Appointment\_\_c> newAppts) {  
 Set<Id> patientIds = new Set<Id>();  
 for (Appointment\_\_c a : newAppts) if (a.Patient\_\_c != null) patientIds.add(a.Patient\_\_c);  
 if (patientIds.isEmpty()) return;  
  
 Map<Id, Patient\_\_c> patients = new Map<Id, Patient\_\_c>(  
 [SELECT Id, Case\_Manager\_\_c FROM Patient\_\_c WHERE Id IN :patientIds]  
 );  
  
 List<Task> tasks = new List<Task>();  
 for (Appointment\_\_c a : newAppts) {  
 if (a.Patient\_\_c == null || a.Appointment\_Date\_\_c == null) continue;  
 Patient\_\_c p = patients.get(a.Patient\_\_c);  
 if (p != null && p.Case\_Manager\_\_c != null) {  
 Date due = a.Appointment\_Date\_\_c.addDays(-1);  
 Task t = new Task();  
 t.OwnerId = p.Case\_Manager\_\_c;  
 t.WhatId = a.Id;  
 t.Subject = 'Reminder: Appointment scheduled';  
 t.ActivityDate = due;  
 t.Status = 'Not Started';  
 t.Priority = 'Normal';  
 tasks.add(t);  
 }  
 }  
 if (!tasks.isEmpty()) insert tasks;  
 }  
  
 public static void handleAfterUpdate(List<Appointment\_\_c> newAppts, Map<Id, Appointment\_\_c> oldMap) {  
 List<Id> apptIds = new List<Id>();  
 for (Appointment\_\_c a : newAppts) {  
 Appointment\_\_c oldA = oldMap.get(a.Id);  
 if (oldA == null) continue;  
 if ((oldA.Status\_\_c == null || oldA.Status\_\_c != 'Completed') && a.Status\_\_c == 'Completed') {  
 apptIds.add(a.Id);  
 }  
 }  
 if (apptIds.isEmpty()) return;  
  
 Map<Id, Integer> existingCount = new Map<Id, Integer>();  
 for (AggregateResult ar : [  
 SELECT Appointment\_\_c apptId, COUNT(Id) cnt  
 FROM Migrant\_Record\_Number\_\_c  
 WHERE Appointment\_\_c IN :apptIds  
 GROUP BY Appointment\_\_c  
 ]) {  
 existingCount.put((Id)ar.get('apptId'), (Integer)ar.get('cnt'));  
 }  
  
 List<Migrant\_Record\_Number\_\_c> toInsert = new List<Migrant\_Record\_Number\_\_c>();  
 for (Appointment\_\_c a : newAppts) {  
 if (!apptIds.contains(a.Id)) continue;  
 Integer cnt = existingCount.containsKey(a.Id) ? existingCount.get(a.Id) : 0;  
 if (cnt == 0) {  
 Migrant\_Record\_Number\_\_c wr = new Migrant\_Record\_Number\_\_c(  
 Appointment\_\_c = a.Id,  
 Patient\_\_c = a.Patient\_\_c,  
 Status\_\_c = 'Open',  
 Notes\_\_c = 'Auto-created on Appointment completion.'  
 );  
 toInsert.add(wr);  
 }  
 }  
  
 if (!toInsert.isEmpty()) insert toInsert;  
 }  
}



### MigrantHealthTests.cls

@isTest  
public class MigrantHealthTests {  
  
 private static Patient\_\_c createTestPatient(String pid) {  
 Patient\_\_c p = new Patient\_\_c(  
 Name = 'Test ' + pid,  
 Patient\_ID\_\_c = pid,  
 DOB\_\_c = Date.newInstance(1990, 1, 1),  
 Age\_\_c = 30,  
 Gender\_\_c = 'Male',  
 Phone\_\_c = '9876543210',  
 Email\_\_c = pid + '@example.com',  
 Case\_Manager\_\_c = UserInfo.getUserId()  
 );  
 insert p;  
 return p;  
 }  
  
 @isTest static void testPatientTrigger() {  
 Patient\_\_c p = createTestPatient('PAT-T1');  
 Patient\_\_c got = [SELECT Id, Age\_\_c FROM Patient\_\_c WHERE Id = :p.Id];  
 System.assertNotEquals(null, got.Age\_\_c);  
 List<Task> tasks = [SELECT Id FROM Task WHERE WhatId = :p.Id];  
 System.assert(tasks.size() >= 1);  
 }  
  
 @isTest static void testAppointmentTrigger() {  
 Patient\_\_c p = createTestPatient('PAT-T2');  
 Appointment\_\_c appt = new Appointment\_\_c(  
 Patient\_\_c = p.Id,  
 Appointment\_Date\_\_c = Date.today().addDays(1),  
 Status\_\_c = 'Scheduled'  
 );  
 insert appt;  
 Test.startTest();  
 appt.Status\_\_c = 'Completed';  
 update appt;  
 Test.stopTest();  
 List<Migrant\_Record\_Number\_\_c> mrs = [SELECT Id FROM Migrant\_Record\_Number\_\_c WHERE Appointment\_\_c = :appt.Id];  
 System.assert(wrs.size() >= 1);  
 }  
}



**Test Checked.**  
  
