**Project : Hospital and clinic management CRM**

***Phase 5: Apex Programming (Developer)***

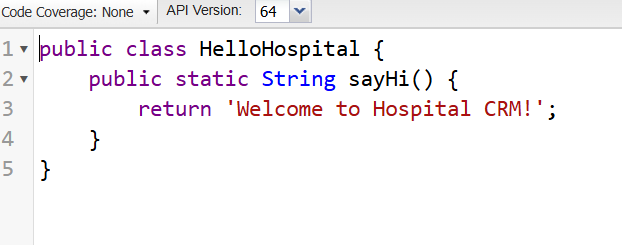
1. ***Classes & Objects + Apex Classes***

**Appex classes**

1. Create the patient helper class (hello Hospital)
2. In developer console ---> File ----->New ---> Apex Classs.

**Code :**

**public class HelloHospital {  
    public static String sayHi() {  
        return 'Welcome to Hospital CRM!';  
    }  
}**



1. Save & test using Developer Console → Execute Anonymous:

Apex Code :

public class AppointmentManager {

public String message;

// Constructor

public AppointmentManager() {

message = 'Welcome to Apex!';

}

public String getGreeting(String name) return 'Hello ' + name;

}

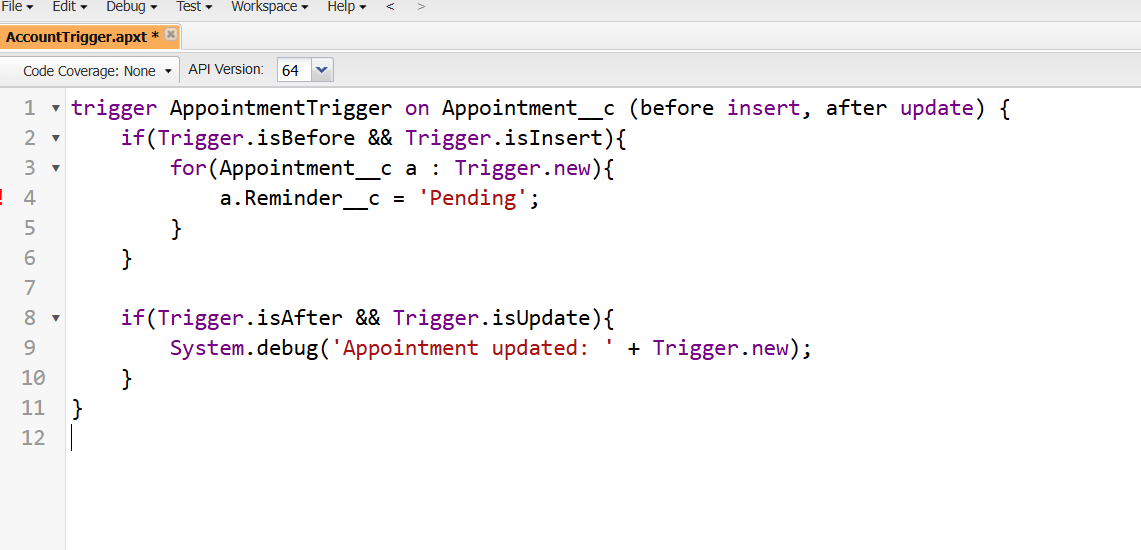
}



**Test:** Execute Anonymous Window me:

AppointmentManager am = new AppointmentManager();

System.debug(am.getGreeting('Vivek'));



1. ***Trigger design pattern***

In Developer console --> Apex triggers --> New --> Appiontment Trigger

**Code :**

trigger AppointmentTrigger on Appointment\_\_c (before insert) {

AppointmentTriggerHandler.handleBeforeInsert(Trigger.new);

}

public class AppointmentTriggerHandler {

public static void handleBeforeInsert(List<Appointment\_\_c> appointments){

for(Appointment\_\_c a : appointments){

a.Reminder\_\_c = 'Pending';

}

}

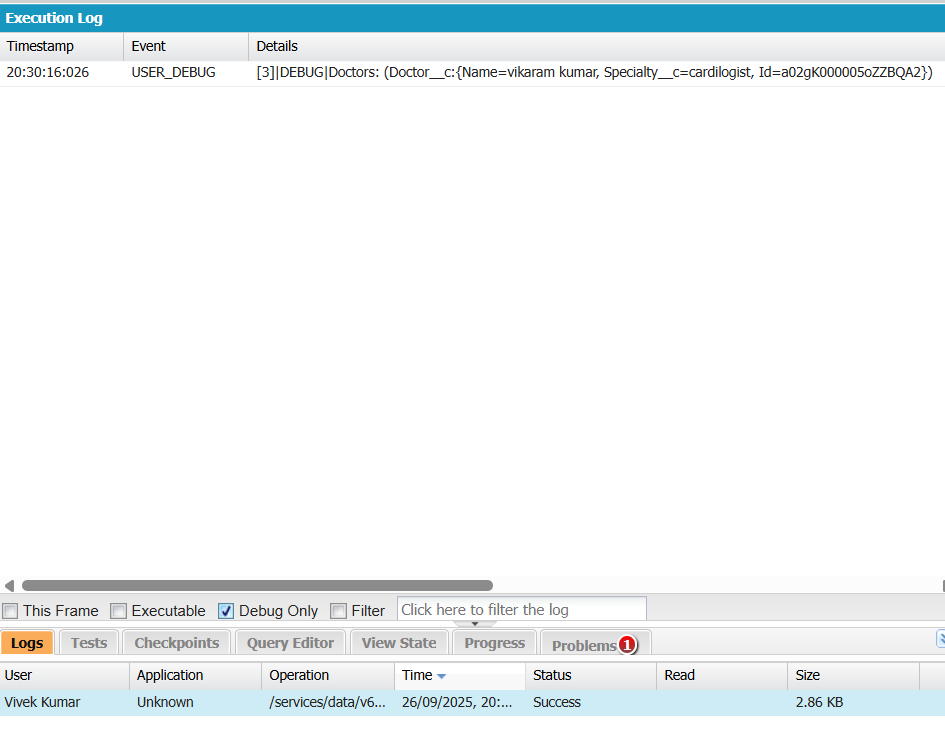
}

Run a SOQL Query

// SOQL Example: Get all Doctors and their Specialty

List<Doctor\_\_c> docs = [SELECT Name, Specialty\_\_c FROM Doctor\_\_c];

System.debug('Doctors: ' + docs);



**(Control+E)**

**SOSL** = search text across objects.

List<List<SObject>> results =

[FIND 'cardio' IN ALL FIELDS RETURNING Doctor\_\_c(Name)];

1. ***Test Classes***

In developer console---> File----->New--->Apex Classs.

**Name : HelloHospitalTest**

**Code :**

@isTest

public class HelloHospitalTest {

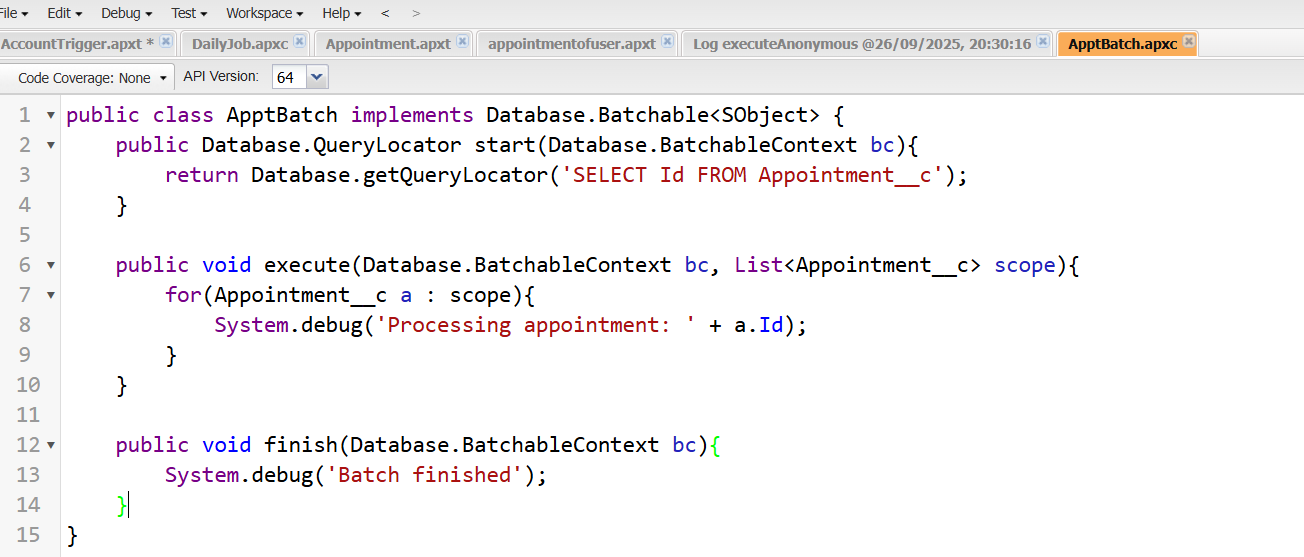
@isTest static void testSayHi(){

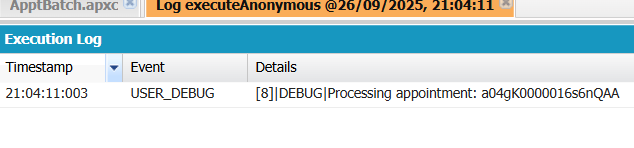
String msg = HelloHospital.sayHi();

System.assertEquals('Welcome to Hospital CRM!', msg);

}

}





1. ***SOQL & SOSL***
2. **Using SOQL in Helper Classes**

In developer console---> File----->New--->Apex Classs.

**Name :** AppointmentHelper

**Code :**

public with sharing class AppointmentHelper {

public static void assignDefaultDoctor(List<Appointment\_\_c> apptList) {

if (apptList == null || apptList.isEmpty()) return;

User defaultDoc = [

SELECT Id

FROM User

WHERE FirstName = 'Dr.' AND LastName = 'Ravi Kumar'

LIMIT 1

];

for (Appointment\_\_c a : apptList) {

if (a.Doctor\_\_c == null) { // only if not already assigned

a.Doctor\_\_c = defaultDoc.Id; // assign the queried Id

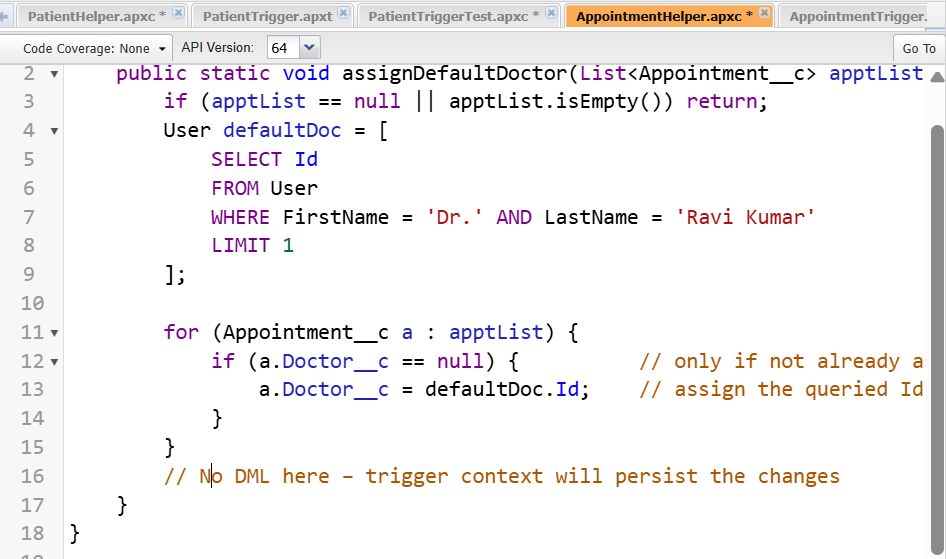
}

}

// No DML here – trigger context will persist the changes

}

}



**Apex Triggers**

**Code :**

trigger AppointmentTrigger on Appointment\_\_c (before insert, before update) {

List<Appointment\_\_c> toProcess = new List<Appointment\_\_c>();

for (Appointment\_\_c a : Trigger.new) {

if (Trigger.isInsert) {

toProcess.add(a);

} else if (Trigger.isUpdate &&

a.Status\_\_c == 'Confirmed' &&

a.Status\_\_c != Trigger.oldMap.get(a.Id).Status\_\_c) {

toProcess.add(a);

}

}

if (!toProcess.isEmpty()) {

AppointmentHelper.assignDefaultDoctor(toProcess);

}

}



1. **Query for some doctor records-**

**//** Query some Doctor records so we can build a Map

List<Doctor\_\_c> docs = [SELECT Id, Name, Specialty\_\_c FROM Doctor\_\_c LIMIT 10];

// --- Collections ---

List<String> names = new List<String>{'A','B'};

Set<String> uniq = new Set<String>{'A','A','B'}; // Duplicates removed

Map<Id, Doctor\_\_c> docMap = new Map<Id, Doctor\_\_c>(docs);

// --- Control Statements ---

for (String n : names) {

if (n.startsWith('A')) {

System.debug('Name starts with A: ' + n);

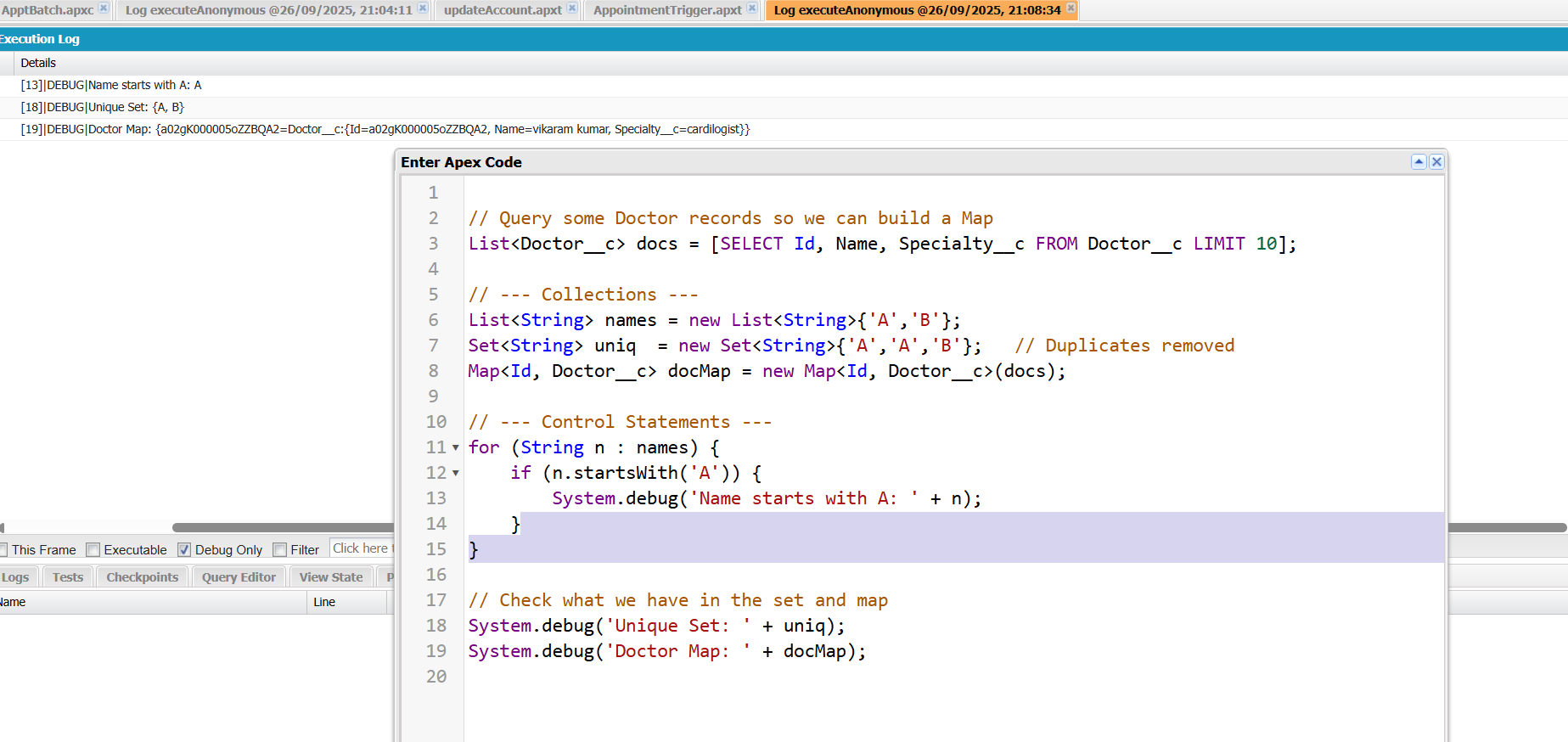
}

}

// Check what we have in the set and map

System.debug('Unique Set: ' + uniq);

System.debug('Doctor Map: ' + docMap);



1. ***Collections: List, Set, Map***

**List:** Ordered collection.

**Set:** Unique items, no duplicates.

**Map:** Key-value pair, fast lookup.

**Code :**

List<Patient\_\_c> patients = new List<Patient\_\_c>();

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

Map<Id, Patient\_\_c> patientMap = new Map<Id, Patient\_\_c>();

1. ***Batch Apex***

In developer console---> File----->New--->Apex Classs.

**Purpose**: Process large data asynchronously in batches.

**Name :** AppointmentReminderBatch

**Code :**

global class AppointmentReminderBatch implements Database.Batchable<SObject>, Database.Stateful {

// Optional: track total processed count

global Integer totalProcessed = 0;

// Query to select appointments to process

global Database.QueryLocator start(Database.BatchableContext bc) {

return Database.getQueryLocator([

SELECT Id, Name, Status\_\_c, Region\_\_c, Reminder\_Sent\_\_c

FROM Appointment\_\_c

WHERE Status\_\_c = 'Scheduled' AND Region\_\_c = 'East'

]);

}

// Process each batch of records

global void execute(Database.BatchableContext bc, List<Appointment\_\_c> scope) {

for (Appointment\_\_c appt : scope) {

appt.Reminder\_Sent\_\_c = true; // Update the reminder field

}

update scope; // Bulk-safe DML

totalProcessed += scope.size();

}

// Optional: finish method for post-processing

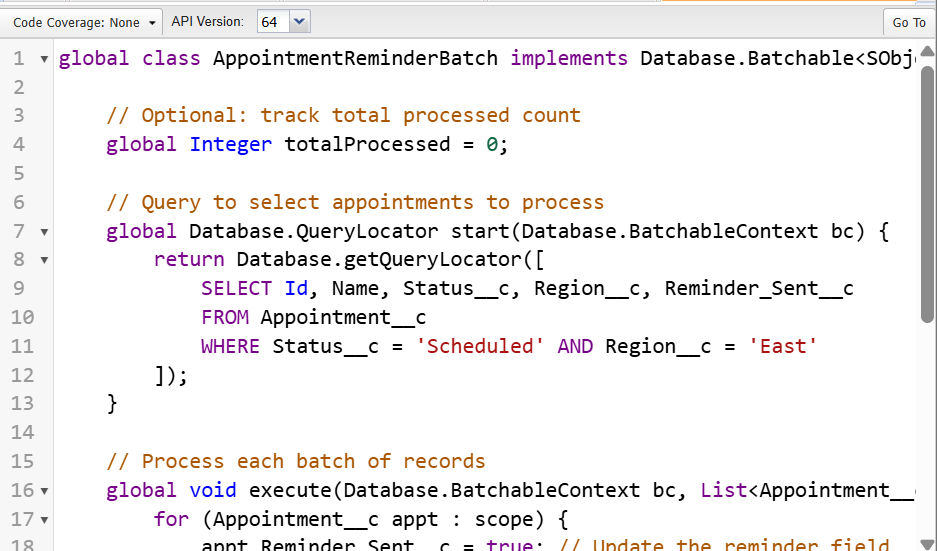
global void finish(Database.BatchableContext bc) {

System.debug('Total Appointments processed: ' + totalProcessed);

// You could also send an email notification to admin here if needed

}

}



1. ***Queueable Apex***

**Purpose:** Asynchronous processing, can chain jobs.

**Code :**

public class QueueJob implements Queueable {

public void execute(QueueableContext qc){

System.debug('Running in background');

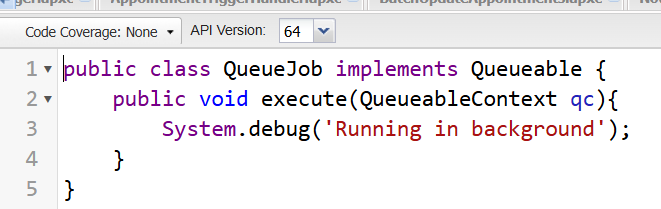
}

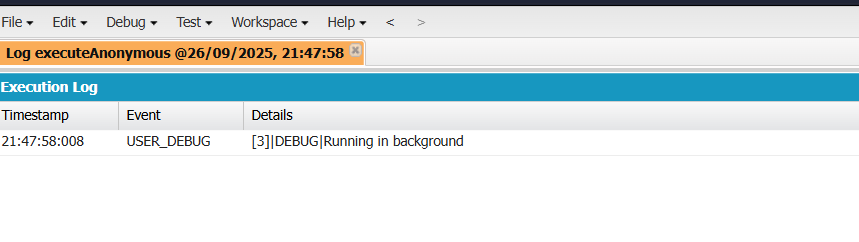
}

// Run in Execute Anonymous Window:

ID jobId = System.enqueueJob(new QueueJob());

System.debug('Job ID: ' + jobId);



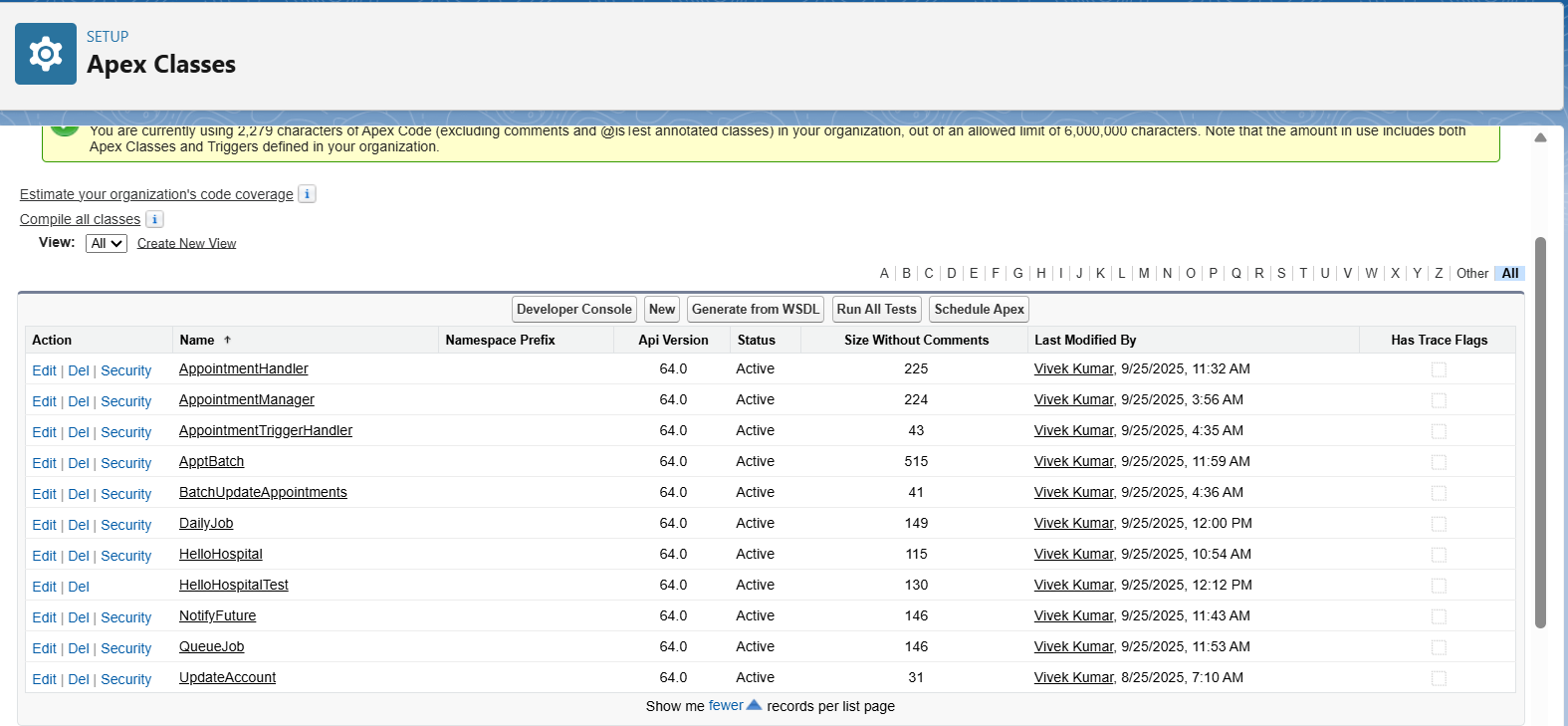
****

**Create the class in Apex Classes → New.**

**Run System.enqueueJob(new QueueJob()) in Execute Anonymous Window.**

**Check Debug Logs for Running in background.**

1. ***future method-***
2. Create the class in **Setup → Apex Classes → New**.
3. Save it.
4. Open **Execute Anonymous Window**, paste the run line, click **Execute**.
5. Check **Debug Logs** to see Sending mail to: test@example.com.

****