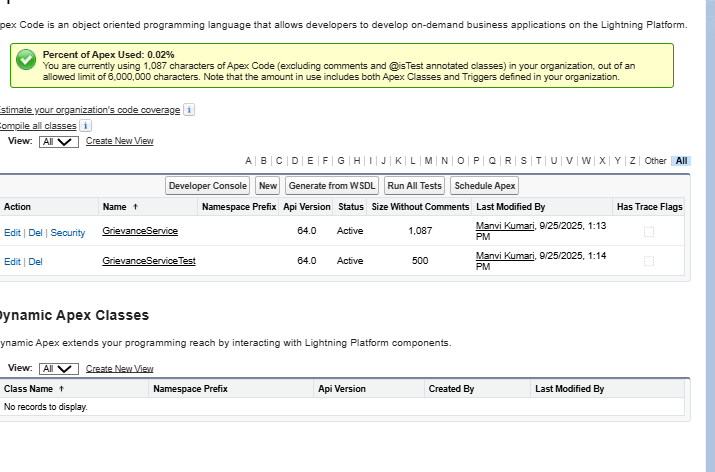
# Phase 5: Apex Programming

Classes & Objects  
Automate grievance handling using Apex reusable service class.

Reduce manual updates (status, escalation, assignment).

Ensure structured workflow → Citizen → Department → Escalation.





createGrievance

Purpose: Create a new grievance record.

Sets: Contact, Department, Status = New.

Returns: New grievance record Id.  
Used when a citizen submits a new complaint.

assignCase

Purpose: Assign an existing grievance to a Department and an Owner (Agent/Officer).

Updates the grievance record with new Department\_\_c and OwnerId.  
 Used when a case is routed to the right department/agent.

escalateCase

Purpose: Escalate a grievance to higher authority.

Creates a Complaint Escalation record (linked to case).

Updates grievance Status → "Escalated - {level}".  
Used when a complaint needs Level 1/Level 2 escalation.

Business Use in Grievease App

Citizens raise complaints (new grievances).

System/Admin assigns cases to departments.

Cases can be escalated automatically or manually.

Provides traceability (history of escalations, owners).

* **Apex Triggers**

Steps in Org to Create Apex Trigger

Go to Setup → Object Manager.

Search and open Grievance Case object.

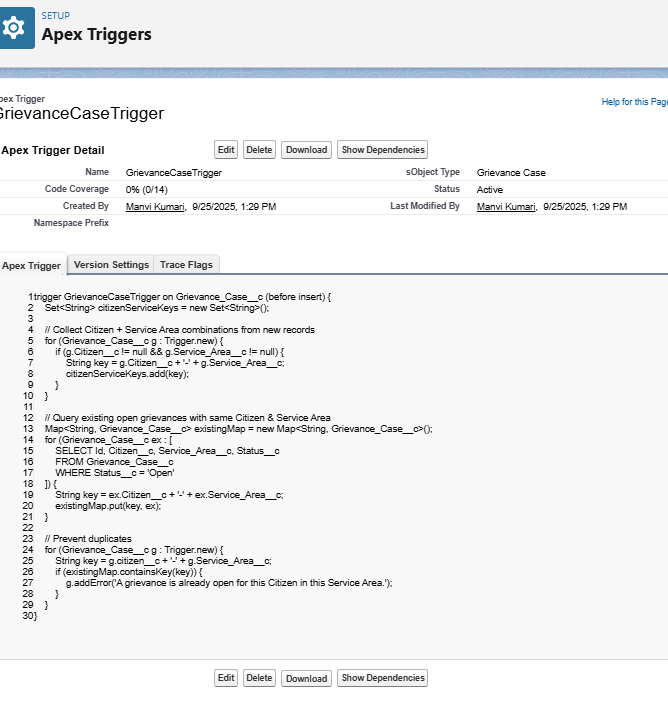
In the left menu, click Triggers → New.

Enter:

Trigger Name: GrievanceCaseTrigger

Object: Grievance\_Case\_\_c

Click Save.



What the code does:

Collects all new grievance records being inserted.

Checks if those citizens already have an Open grievance in the same service area.

If yes → shows an error and stops duplicate record creation.

## **Trigger Design Pattern**

## Trigger Design Pattern (Handler Class Approach)

Instead of putting all logic inside the trigger, we separate it into a **Trigger Handler Class**.  
This makes code **cleaner, reusable, and testable**.

**Create the Trigger**

* Go to **Setup → Object Manager → Grievance Case → Triggers → New Trigger**.
* Name: GrievanceCaseTrigger
* Event: *before insert* (for duplicate prevention) or *after insert* (for escalation).
* Keep trigger body minimal (just call handler).

CODE   
trigger GrievanceCaseTrigger on Grievance\_Case\_\_c (before insert, before update) {

GrievanceCaseTriggerHandler.run(Trigger.new, Trigger.oldMap, Trigger.isInsert, Trigger.isUpdate);

}

**Create the Handler Class**

Go to Setup → Apex Classes → New.

Name: GrievanceCaseTriggerHandler.

Add business logic here. Example:

* ***SOQL & SOSL***

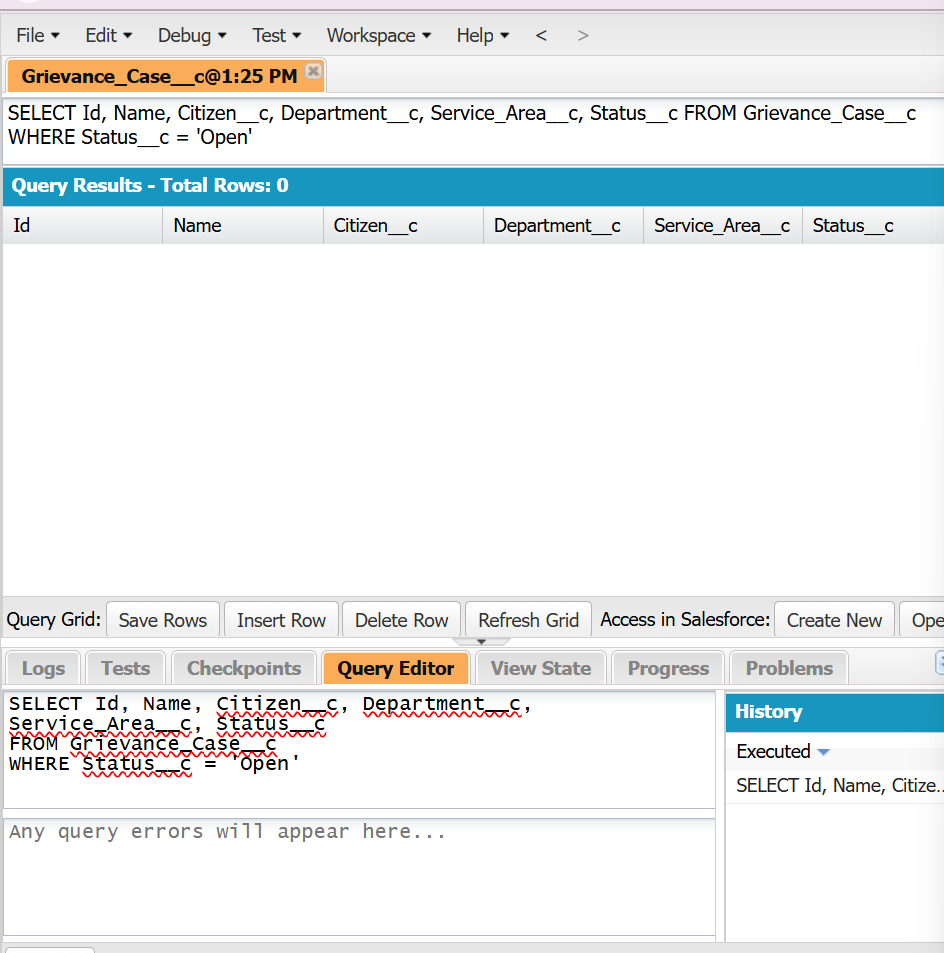
**Steps to Run in Org**

Log in to Salesforce.

Go to Developer Console (top right gear → Developer Console).

Open Query Editor and write code.

Click Execute → You’ll see all “Open” grievances.

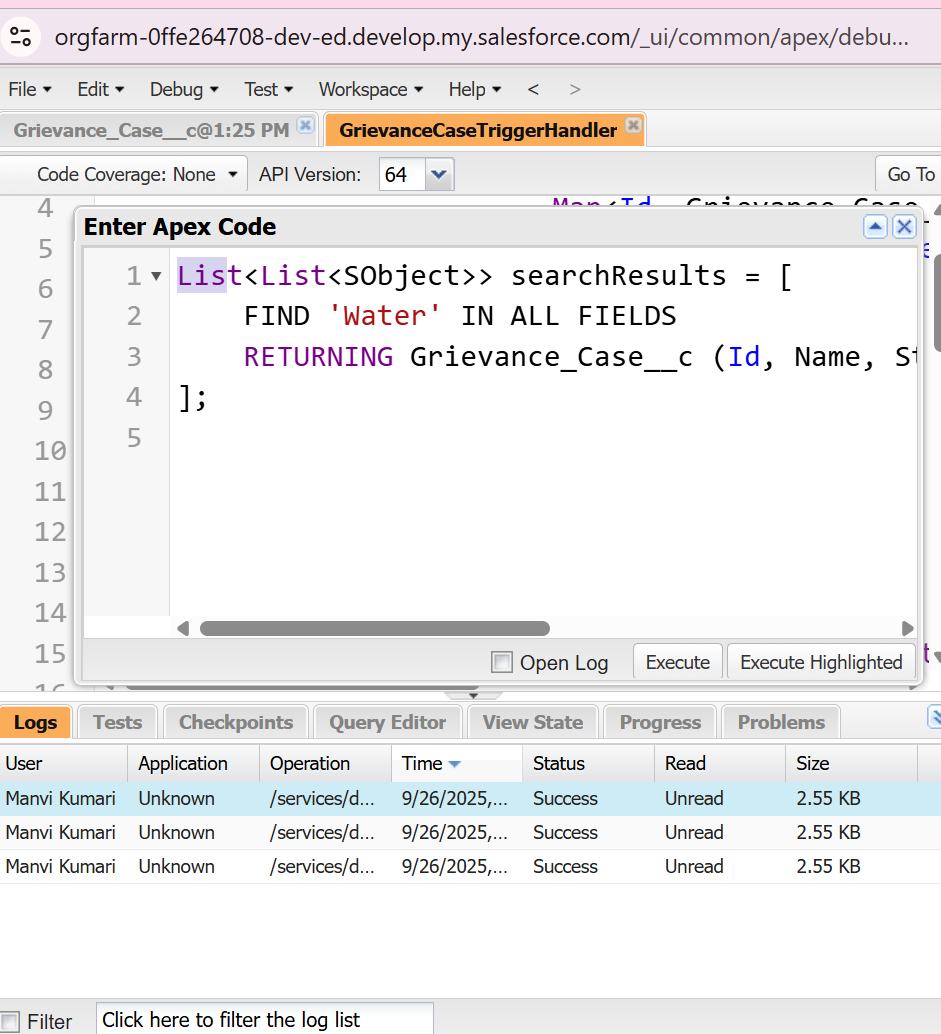
  
 **Steps to Run in Org**

Log in to Salesforce.

Go to Developer Console (top right gear → Developer Console).

Open Query Editor.

Click Execute → You’ll see all “Open” grievances.

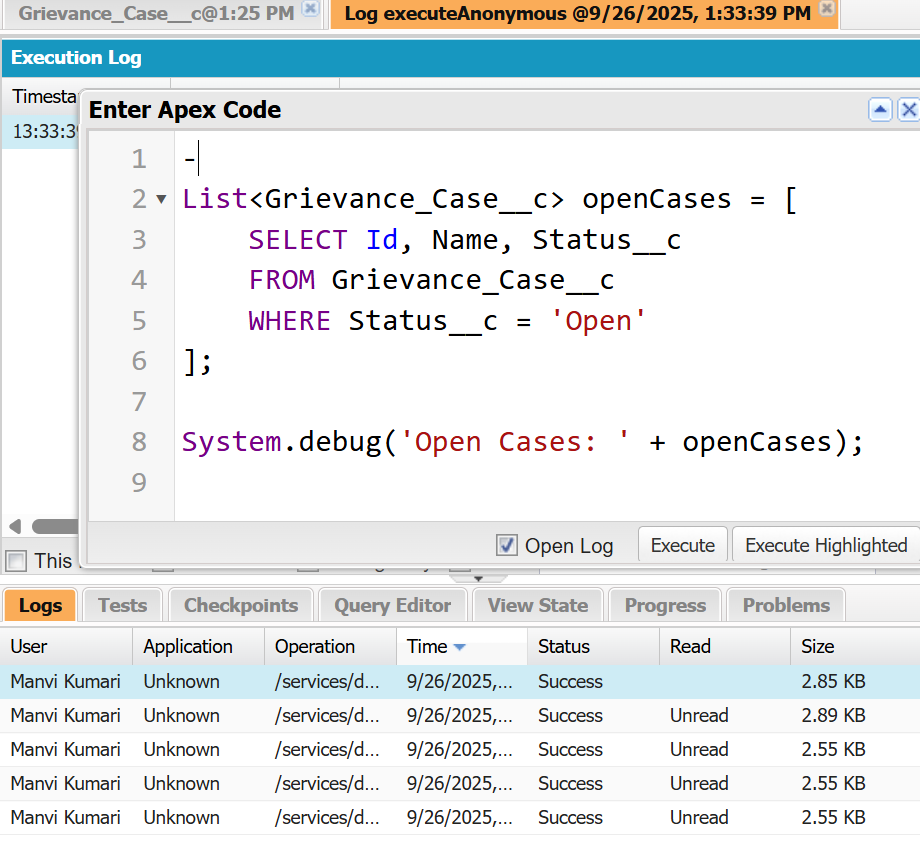


**Purpose**

* Finds grievances containing the keyword **“Water”** anywhere (title, description, etc.).
* Useful when citizens don’t know exact case ID but know a keyword.

## **Collections: List, Set, Map**

Open **Developer Console → Execute Anonymous Window (Ctrl+E)**.

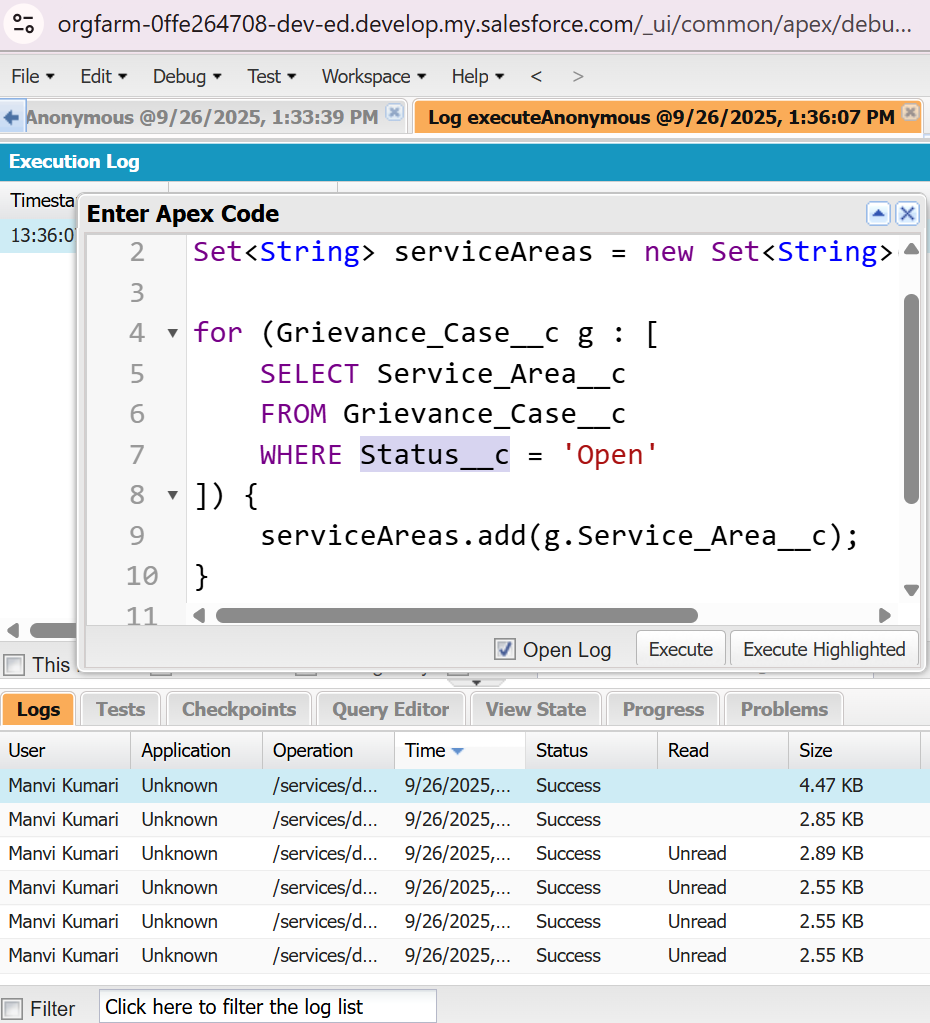


Purpose: To work with multiple grievance records together (like showing all open complaints).

SETS - Unordered collection, does **not allow duplicates**.  
 **Use in GrievEase**: Store unique Service Areas for complaints.

**Steps**:

1. In Execute Anonymous Window.

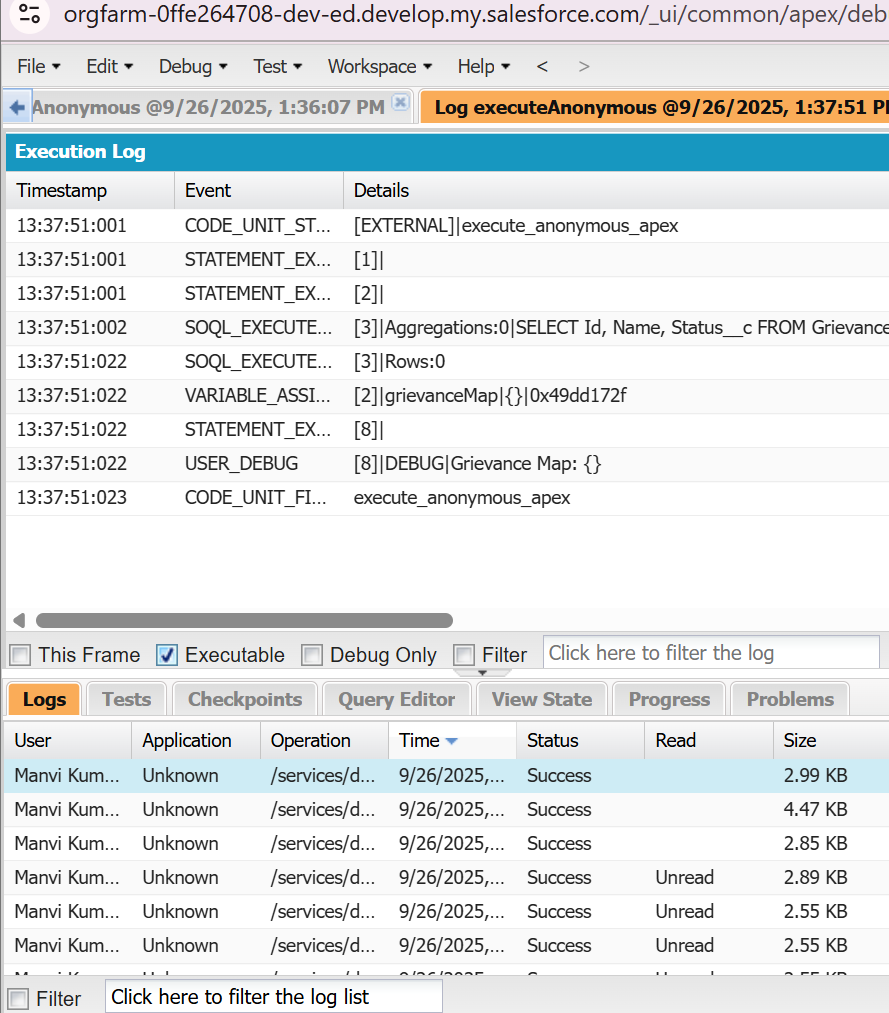


Avoid duplicate entries, like finding which **areas** have open complaints.

**MAPS** : Collection of key–value pairs.  
 **Use in GrievEase**: Map grievance **Id → grievance record** or Citizen → Case.

**Steps**:

1. In Execute Anonymous Window.

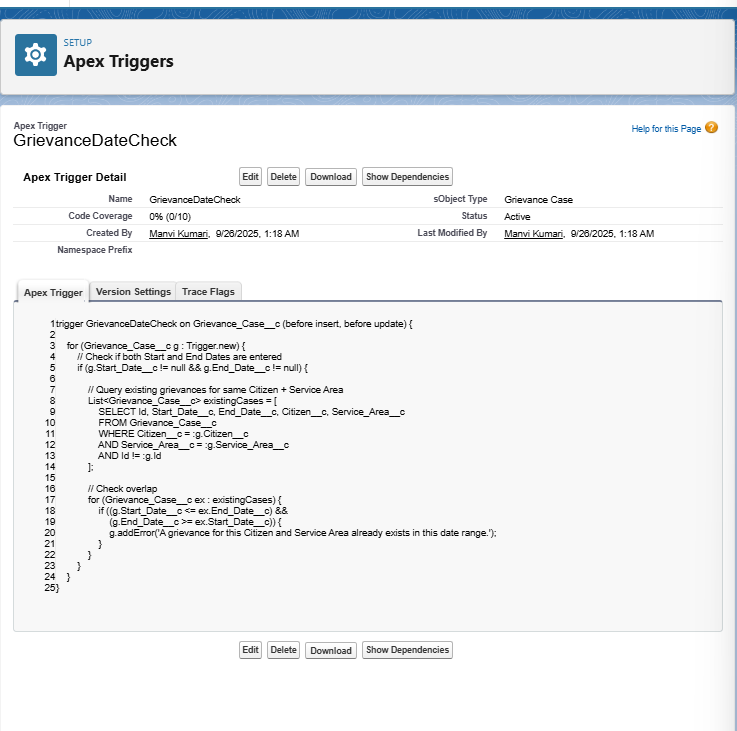
****

Quickly fetch a grievance record by Id or Citizen, useful in triggers and services.

## **Control Statements**

Go to Setup → Apex Triggers → New Trigger.

Select object → Grievance\_Case\_\_c.



**Purpose**

* + Uses an **IF condition** to check **date overlap**.
  + Prevents duplicate/overlapping grievances for the same **Citizen + Service Area**.
* **Why important**:
  + Ensures **data accuracy**.
  + Avoids **multiple active cases** for the same issue at the same time.

## **Batch Apex**

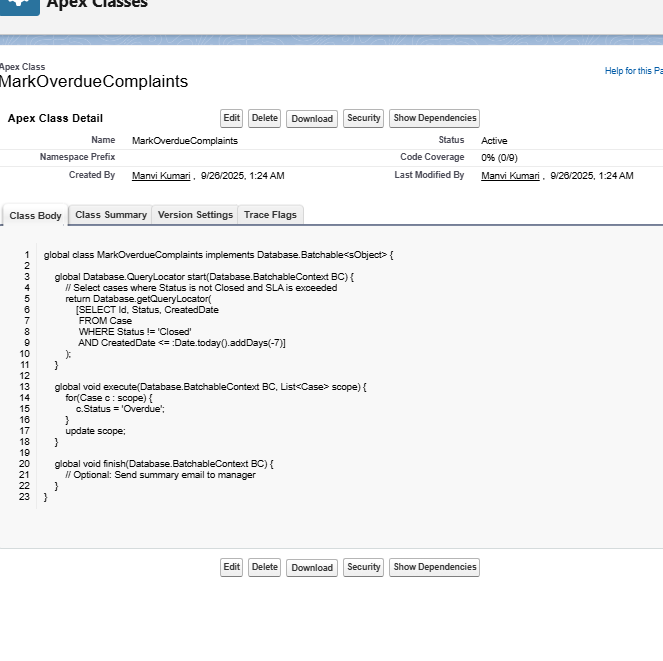
* Purpose in GrievEase

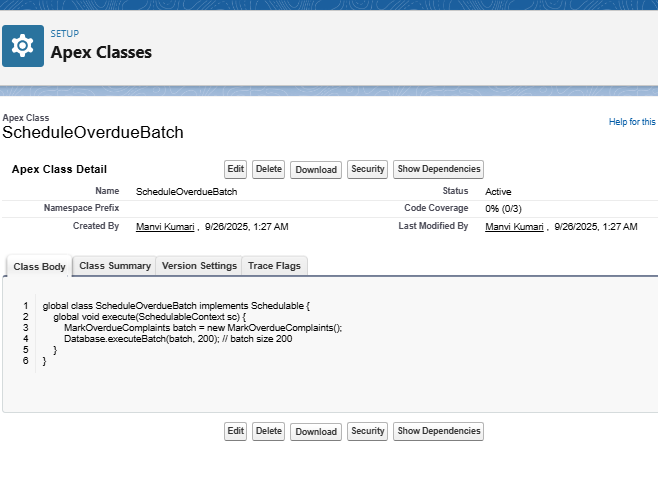
Automated monitoring of complaint resolution

Highlights overdue complaints for faster follow-up

Supports transparency and accountability

Scales to large number of complaints





Schedule via **Setup → Apex Classes → Schedule Apex**

Run nightly at, e.g., 2:00 AM

## **Queueable Apex**

Some **complaints** may require **bulk calculations or updates** that can’t be done immediately (e.g., applying SLA penalties, priority scoring, or overdue flags).

**Queueable Apex** allows asynchronous execution of such tasks.

Purpose in GrievEase:

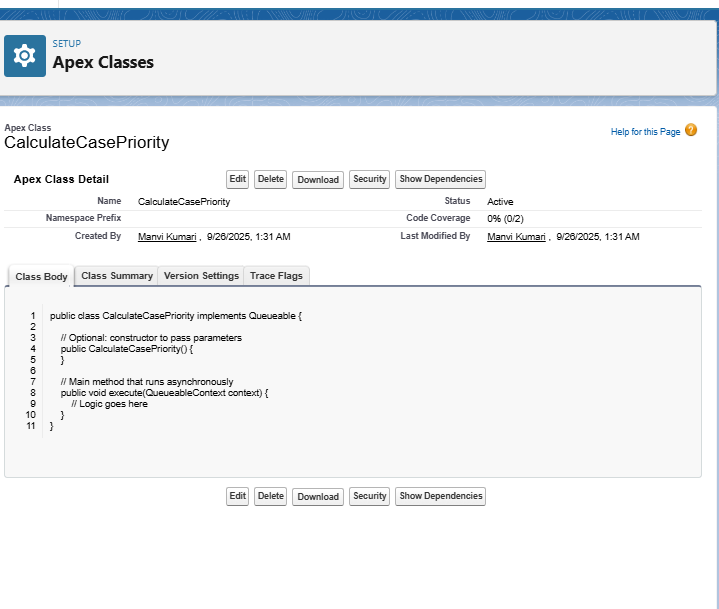
**Calculate SLA penalties or priority scores** for multiple complaints in the background.

Avoids **slowing down the UI** for users.

Supports **batch updates** efficiently.

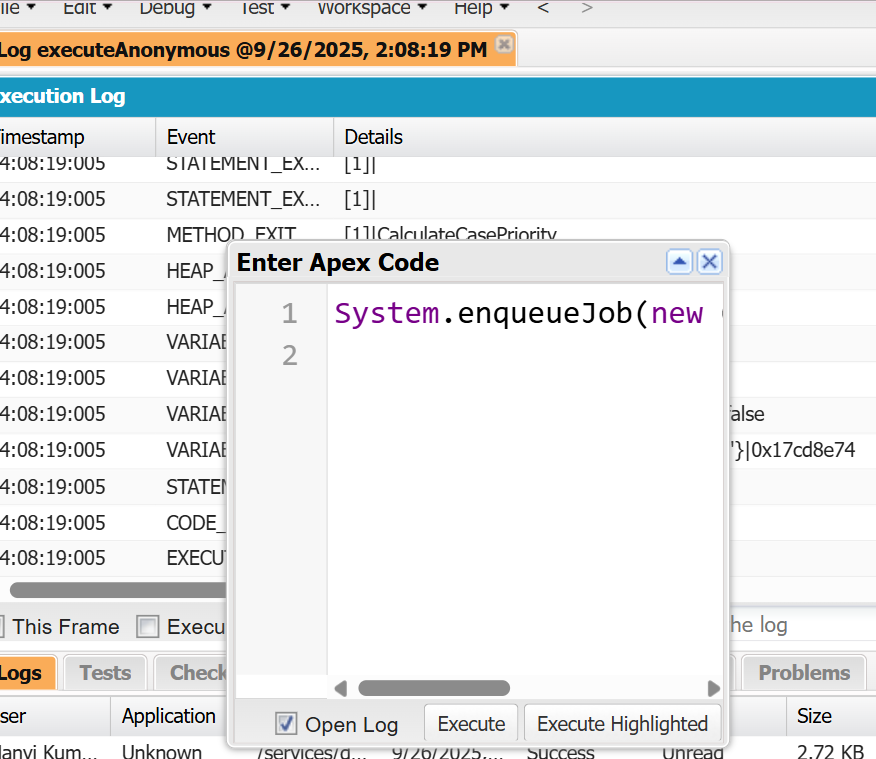
Go to **Setup → Apex Classes → New**.

Write a **Queueable class** that implements Queueable.



**Execute Anonymous Window (Developer Console)**

1. Open **Developer Console → Debug → Open Execute Anonymous Window**



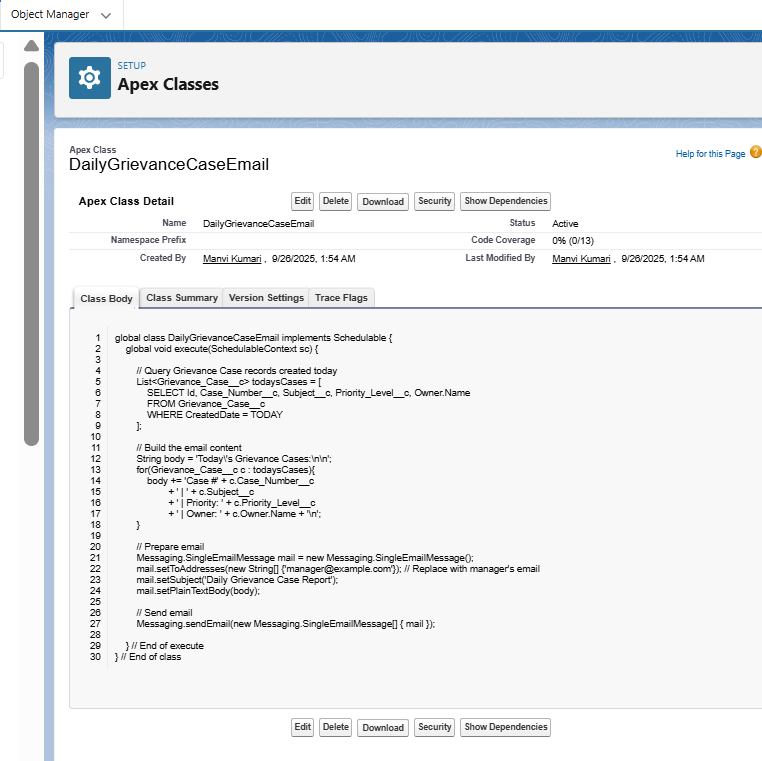
## **Scheduled Apex**

**Purpose**

* Automatically **send a daily email to the manager** with all complaints created or assigned today.
* Purpose in GrievEase:
  + Keep managers **updated on new complaints**
  + Enable **timely assignment and follow-up**
  + Improve **transparency and SLA compliance**

**Create the Apex Class**

1. Go to **Setup → Apex Classes → New**.
2. Write a **Schedulable class**:



Every morning, the manager receives an email with all today’s complaints.

Reduces manual reporting and ensures timely action.

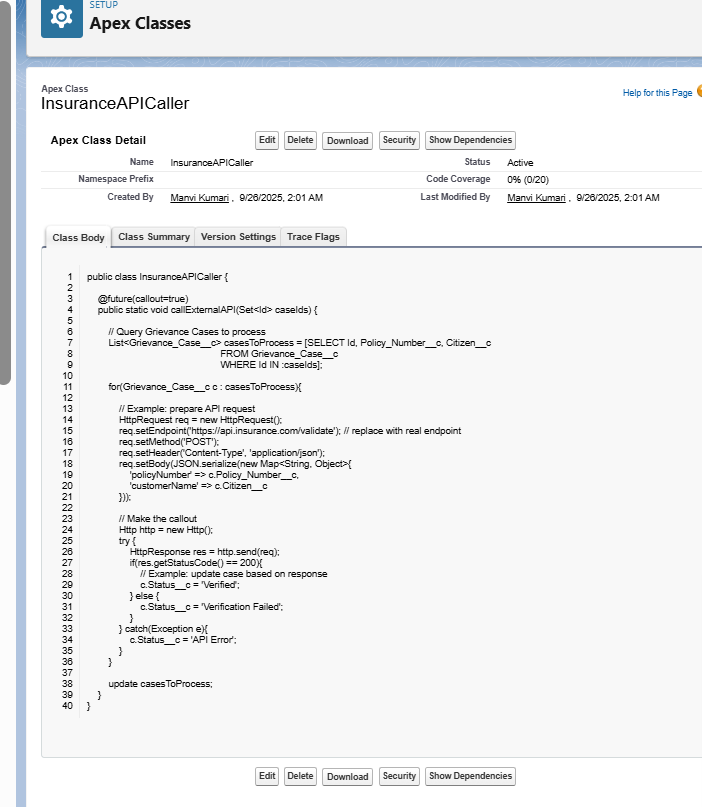
## **Future Methods**

Some complaints in GrievEase may require **verification or validation via an external system**, like insurance confirmation.

Future methods allow **asynchronous API calls** without blocking the main Salesforce transaction.

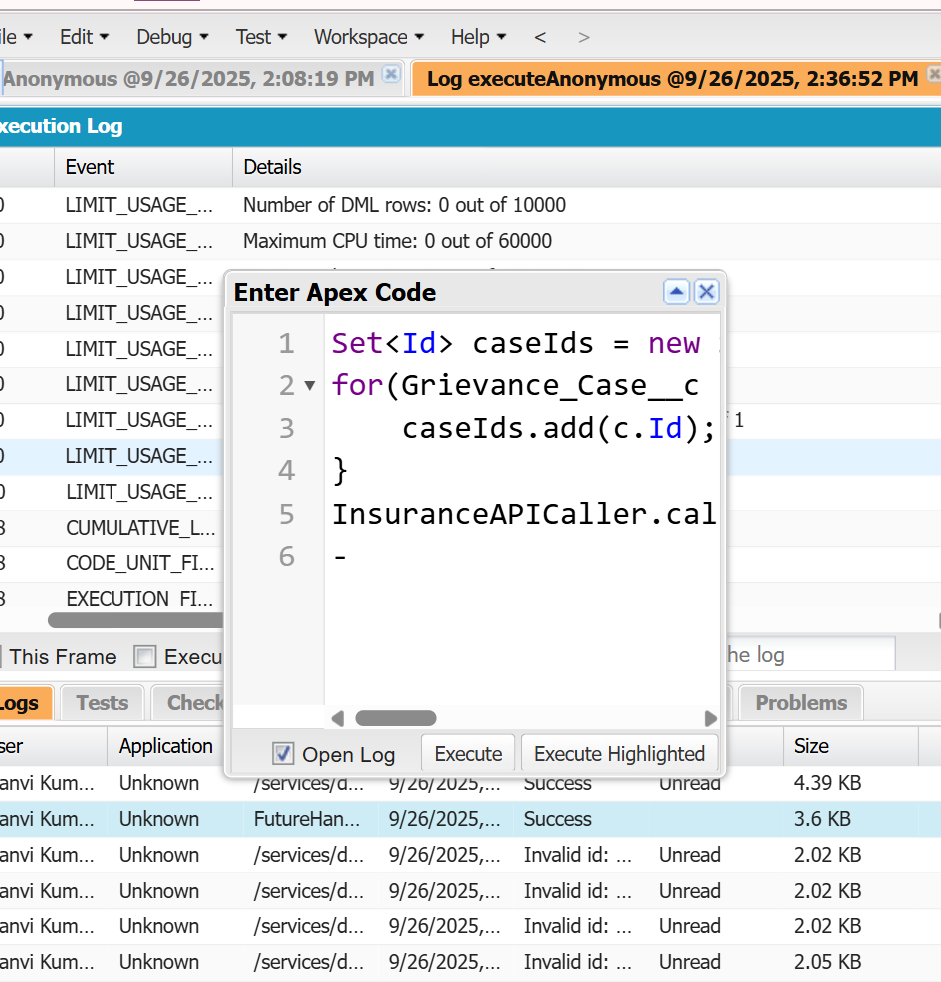
Purpose:

* **Call external insurance APIs** in the background
* Avoid UI delays for users
* Process multiple cases efficiently



**Key Takeaways**

| **Field** | **Use Case** |
| --- | --- |
| Id (15/18 char) | Apex operations, queries, callouts, triggers |
| Case\_ID\_\_c (CI-0001) | Display, emails, reports |



* **Exception Handling**

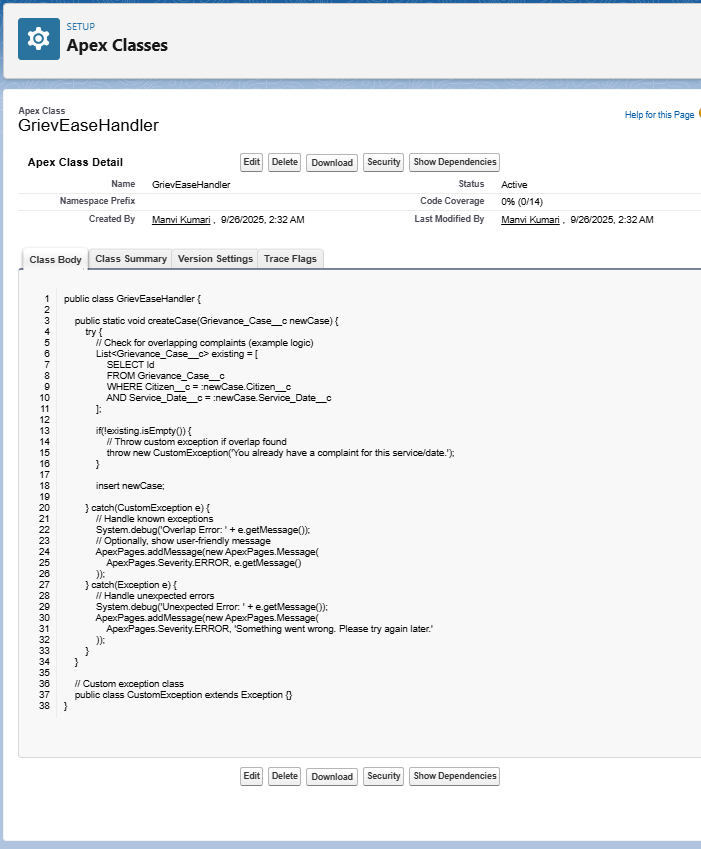
Purpose

Prevent duplicate or overlapping entries (e.g., same citizen submits same complaint twice, or overlapping service requests).

Handle the error gracefully so users see a friendly message.

Log the error if needed.

1. Using Try-Catch



## **Test Classes**

For **GrievEase Cases**, we need a **Test Class** to:

1. **Create test data**
2. **Insert a Grievance Case**
3. **Check that triggers, validation rules, or exception handling work**

