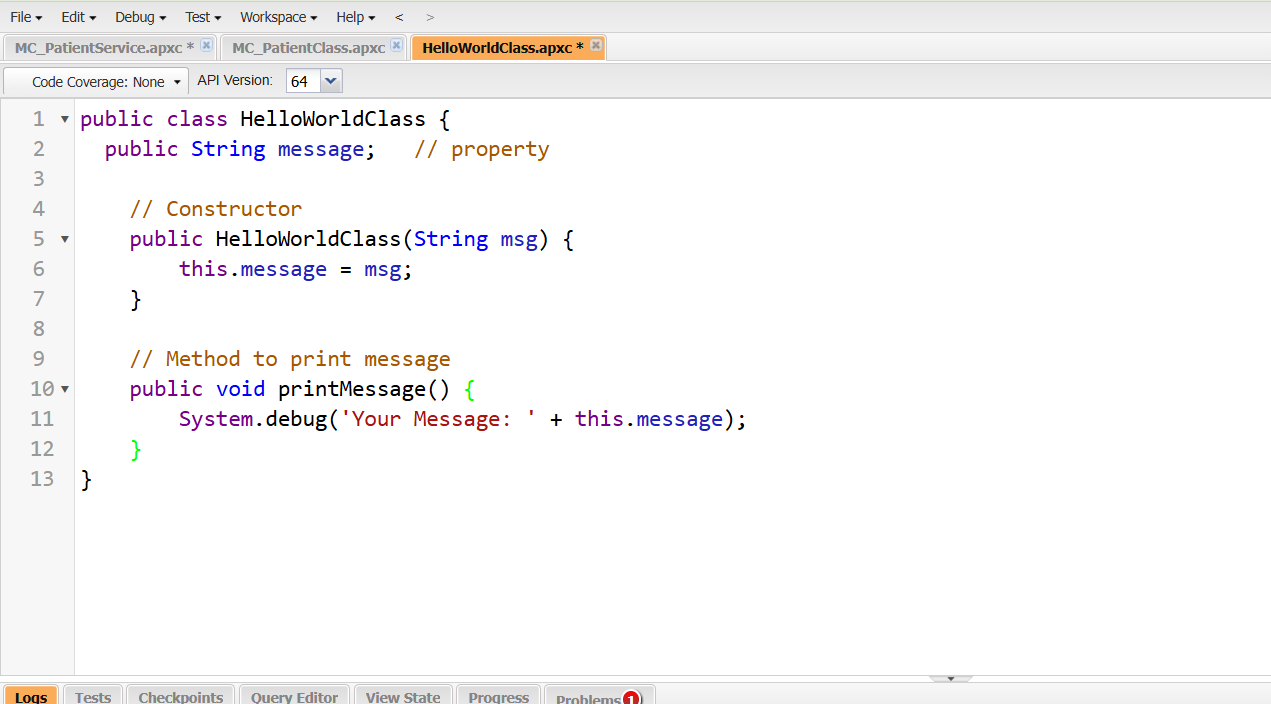
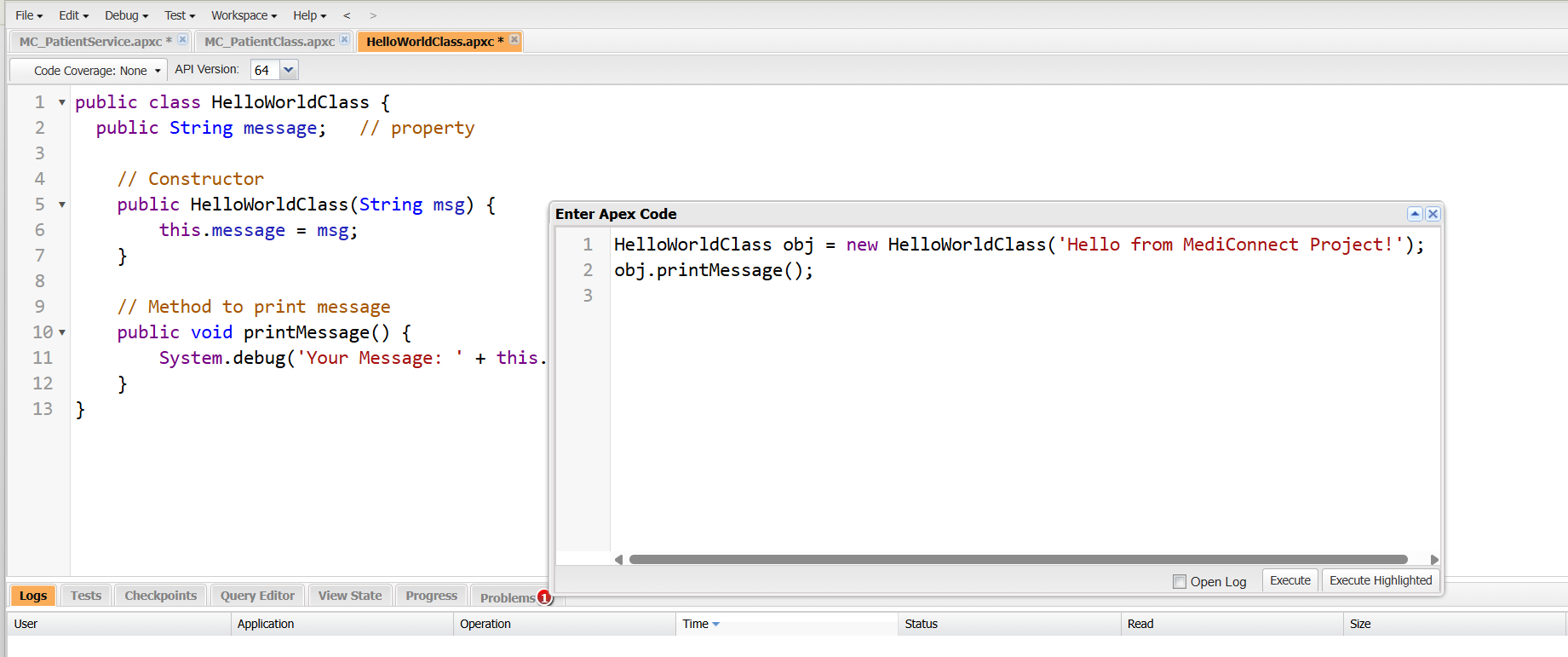
**Phase 5: Apex Programming (Developer)**

1. **Classes & Objects**

Class in Apex → A blueprint that defines variables (data) and methods (actions).  
Object in Apex → An instance of a class (real entity created from the blueprint).

* **Class = design/blueprint**
* **Object = actual usable copy**



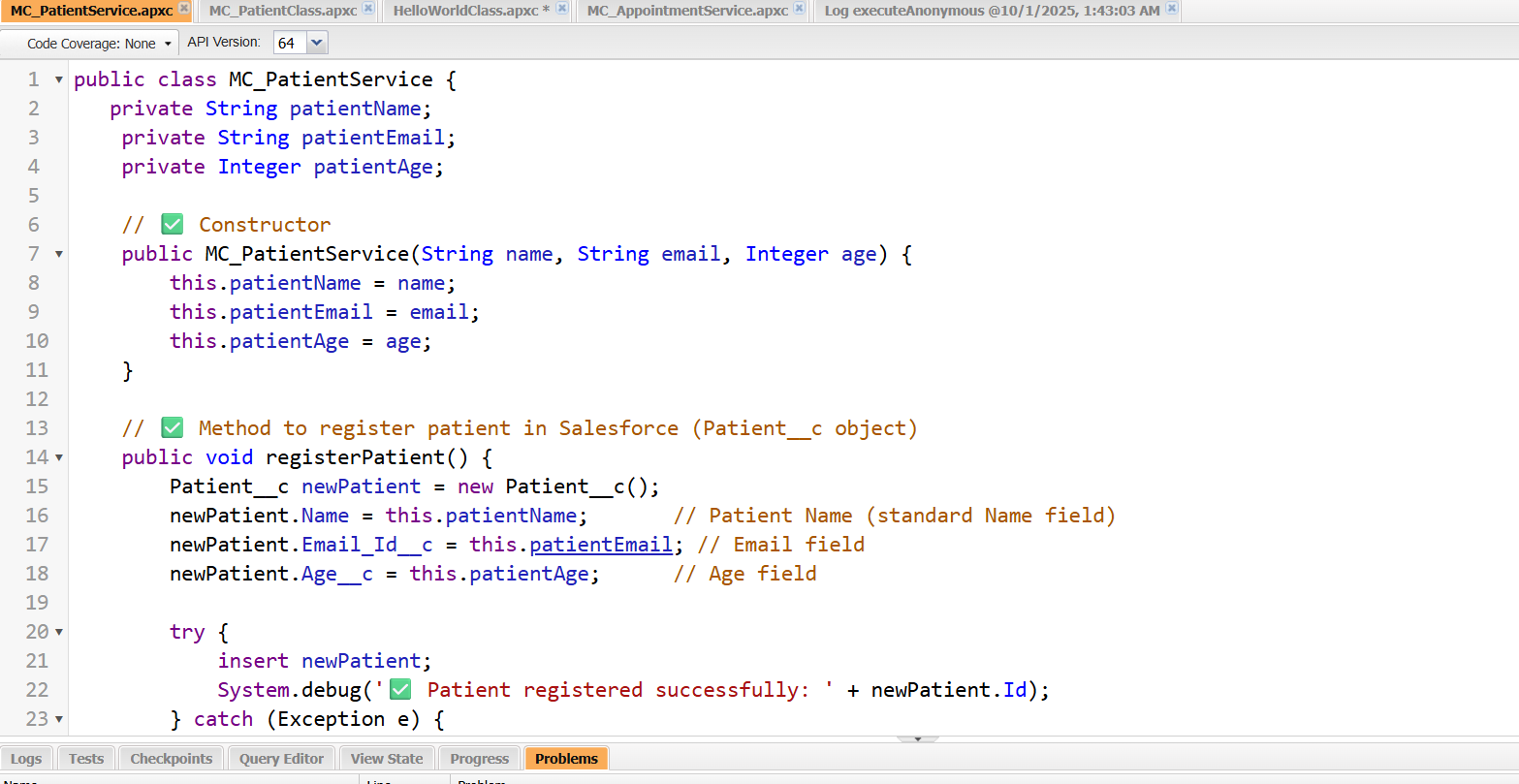


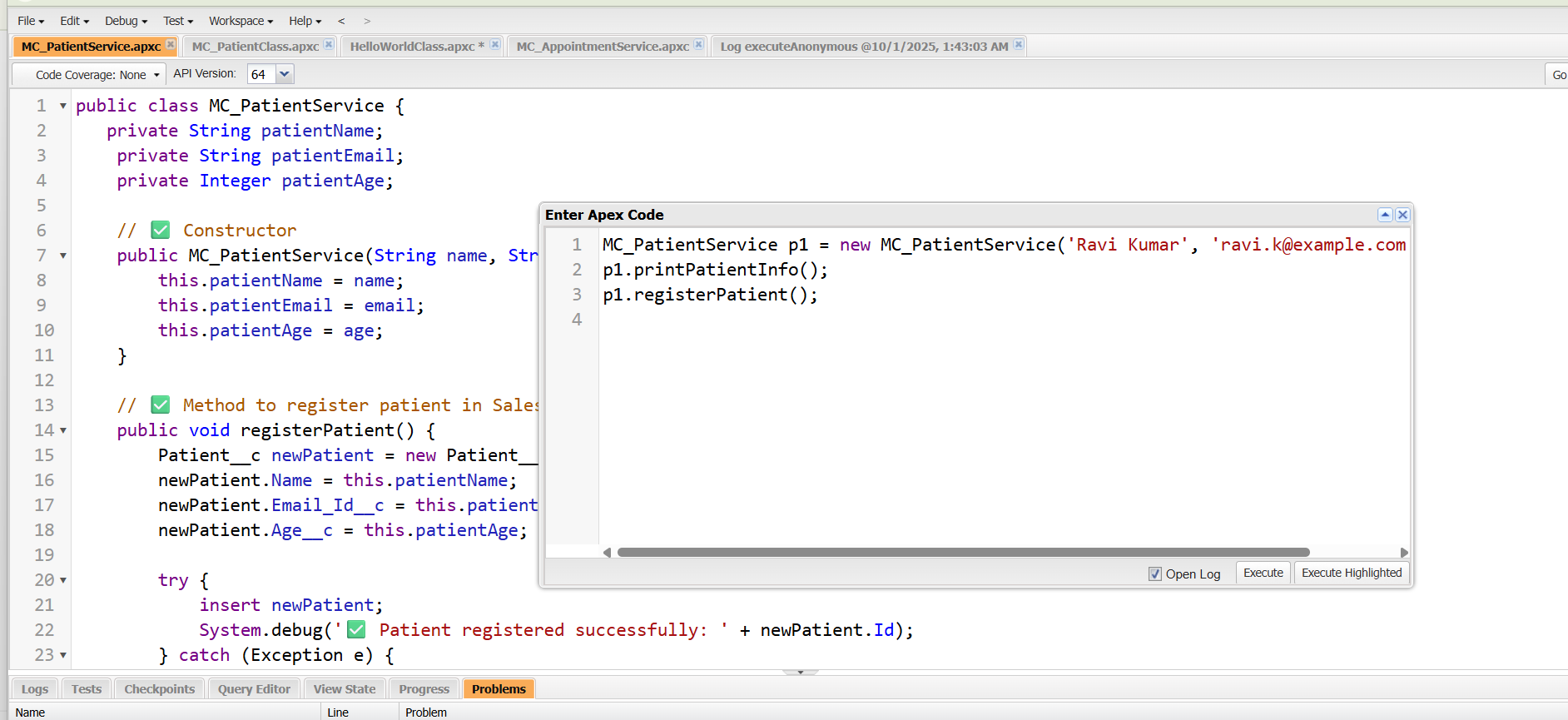
Example- MC\_PatientService Class

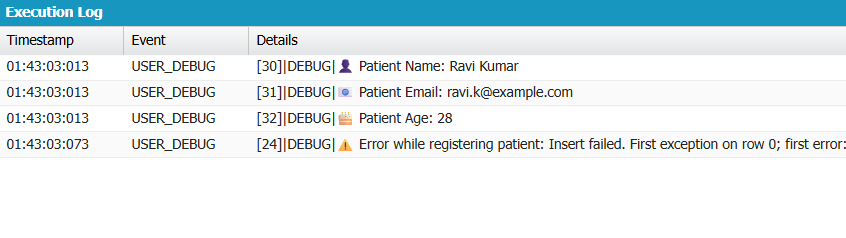
**Purpose:**  
This Apex class handles patient registration in the MediConnect project. It allows creating a new patient record in Salesforce and printing patient information to debug logs.

**Key Features:**

1. **Constructor:** Initializes patient details (Name, Email, Age).
2. **registerPatient() Method:** Inserts a new record into the Patient\_\_c object with the provided details.
3. **printPatientInfo() Method:** Prints the patient’s information in the debug logs for verification.







**Output:**

* Debug Log shows patient details.
* A new patient record is inserted into Salesforce Patient\_\_c object.

**Related Object:** Patient\_\_c with fields: Name, Email\_Id\_\_c, Age\_\_c

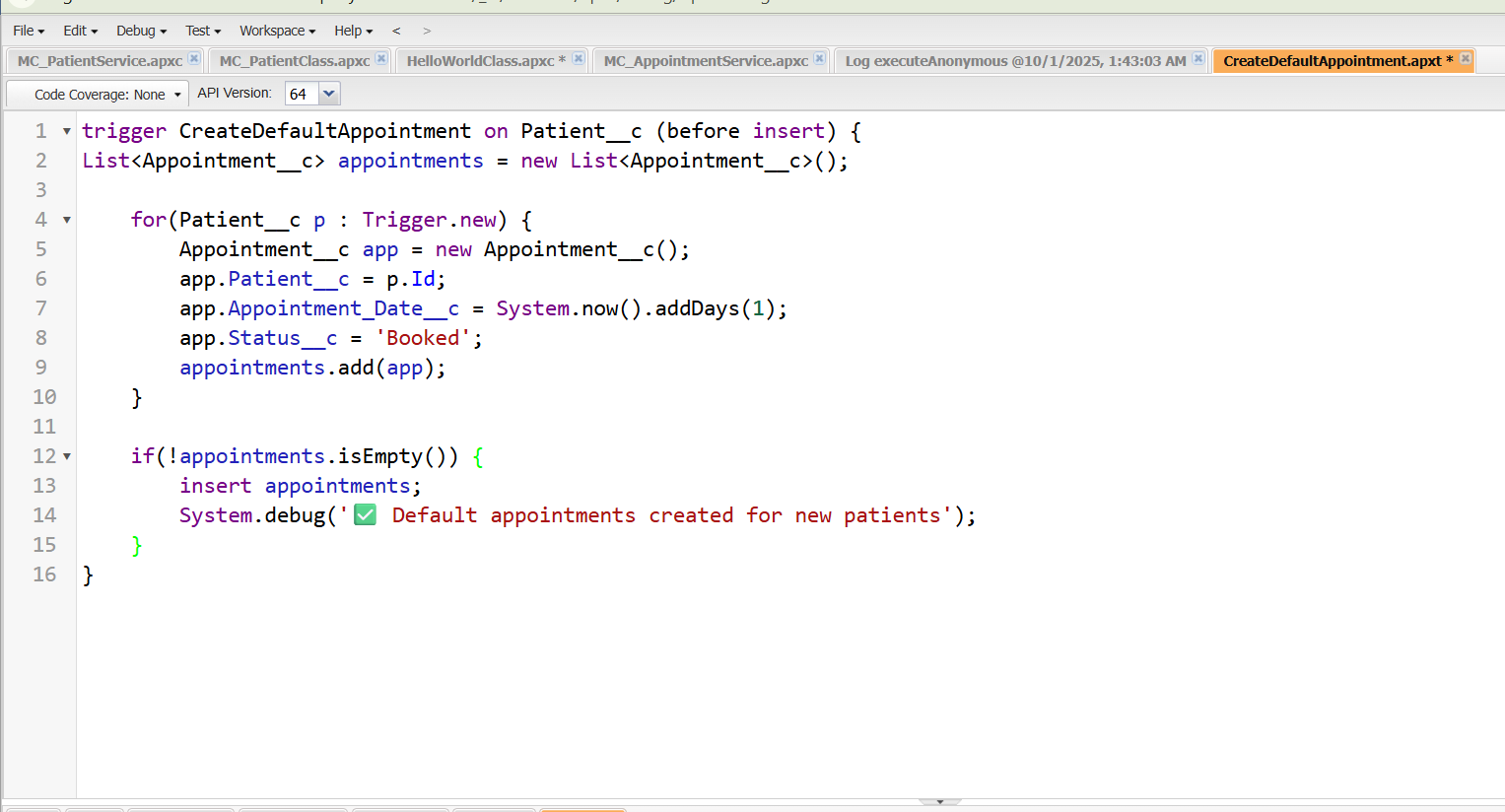
**2.Apex Triggers (before/after insert/update/delete)**

**Apex Trigger** → A piece of code that executes **automatically** before or after a record is **inserted, updated, deleted, or undeleted** in Salesforce.

* **Before Trigger:** Runs **before** the record is saved → commonly used for validation or modifying values.
* **After Trigger:** Runs **after** the record is saved → commonly used for creating related records, sending notifications, or updating other objects.

**Example-** Create DefaultAppointment Trigger

**Use Case:** Automatically create a default **Appointment\_\_c** when a new **Patient\_\_c** is inserted.

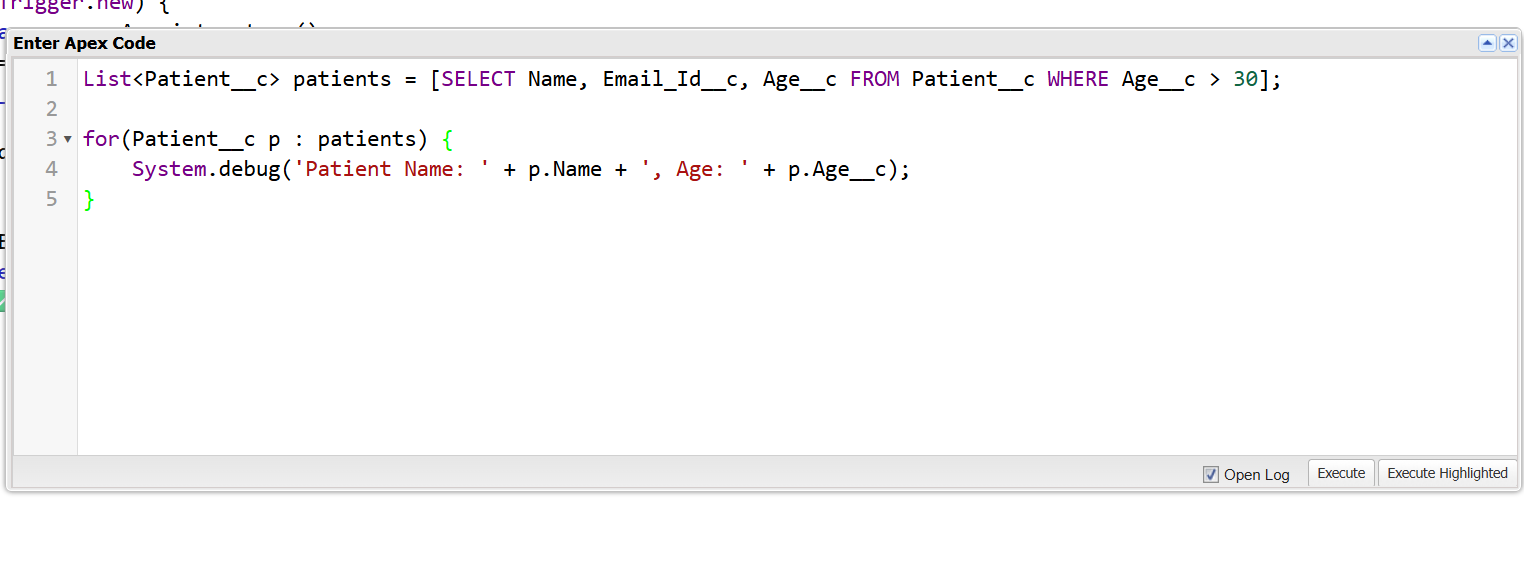


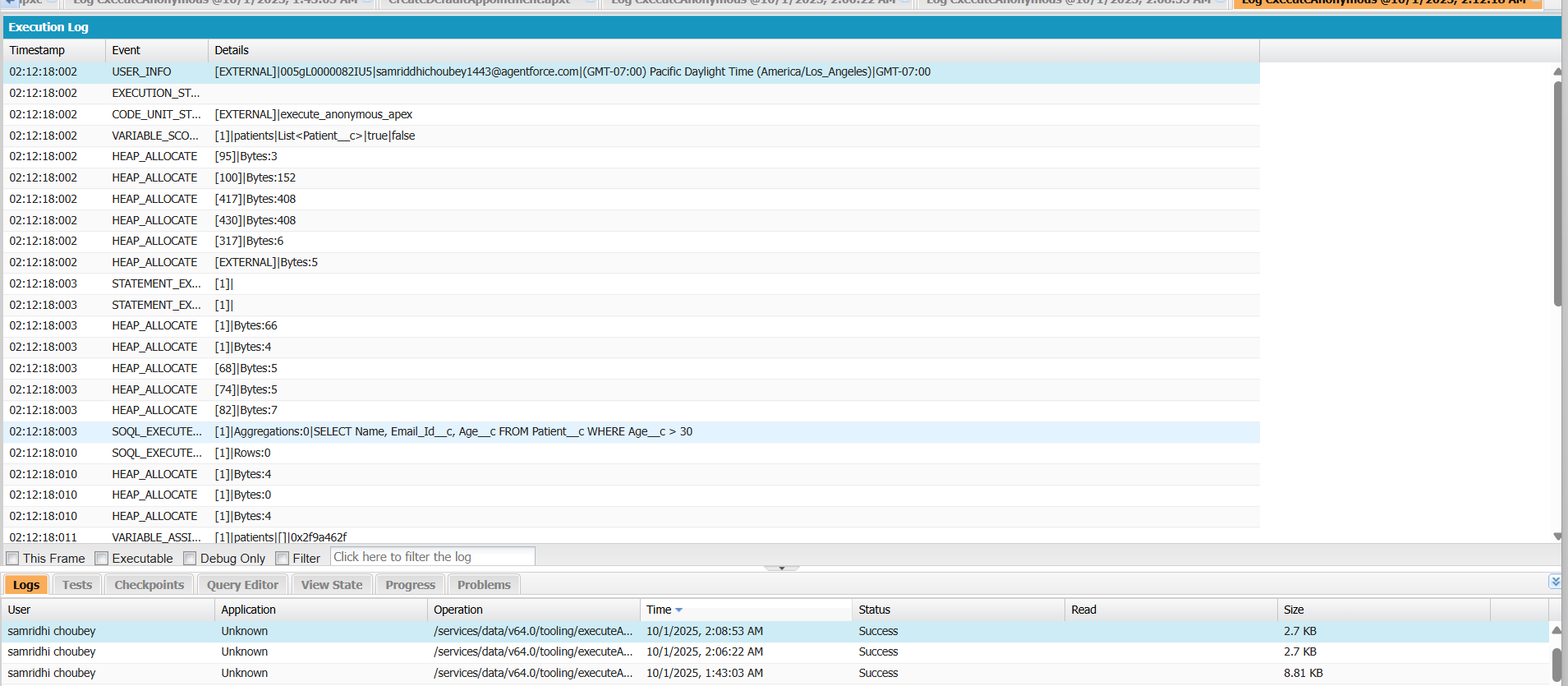
**3.SOQL & SOSL**

**1. SOQL (Salesforce Object Query Language)**

SOQL is used to query records from a single Salesforce object (like SQL in databases).

* Use Case: Get patient details, appointments, doctors, etc.

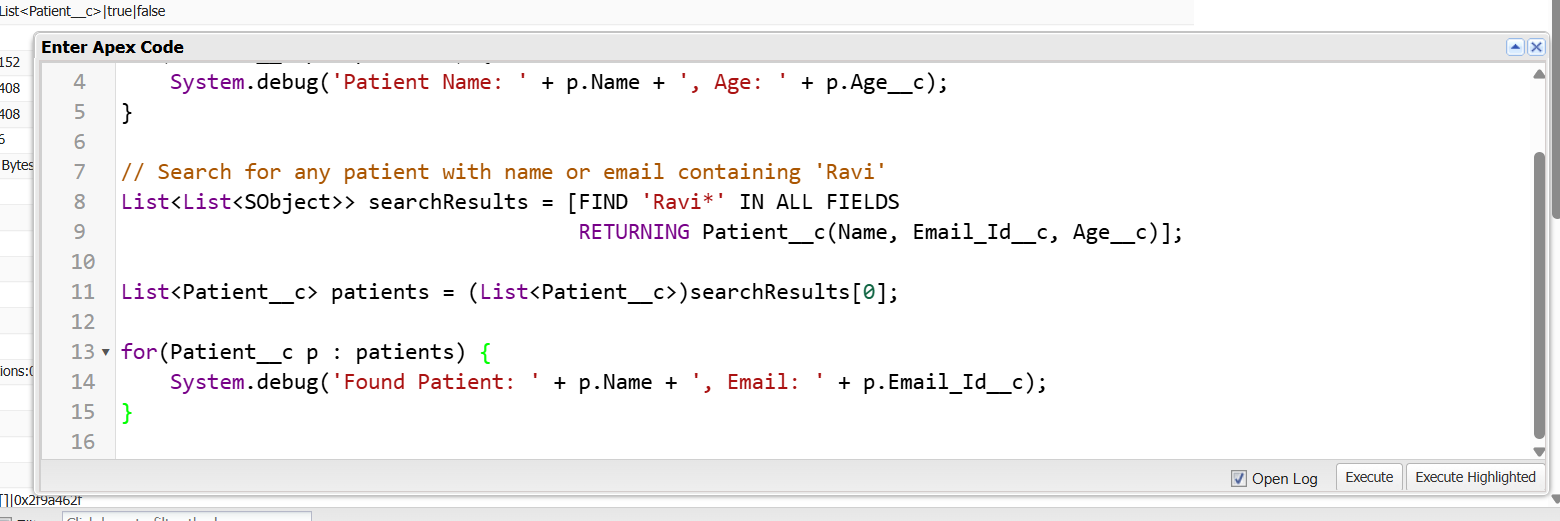




**2. SOSL (Salesforce Object Search Language)**

SOSL is used to **search across multiple objects** or fields at once.

* Useful for global search or when you don’t know which object/field contains the data.



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

**1. Collections in Apex**

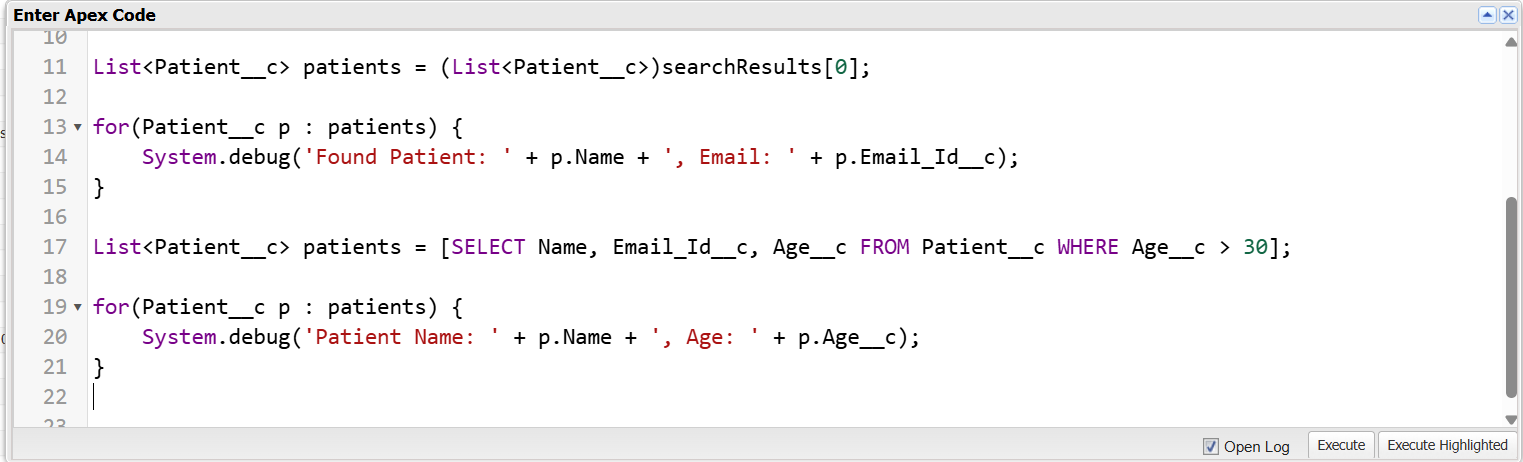
**Collections →** Special types of variables that can store multiple values. Apex me 3 main types hain:

|  | **Description** | **Syntax Example** |
| --- | --- | --- |
| **List:** | Ordered collection, -duplicates allowed | *List<Patient\_\_c> patients = new List<Patient\_\_c>();* |
| **Set:** | Unordered, unique -values only | *Set<String> emails = new Set<String>();* |
| **Map:** | Key-Value pair, -unique keys | *Map<Id, Patient\_\_c> patientMap = new Map<Id, Patient\_\_c>();* |

**MediConnect Examples:**

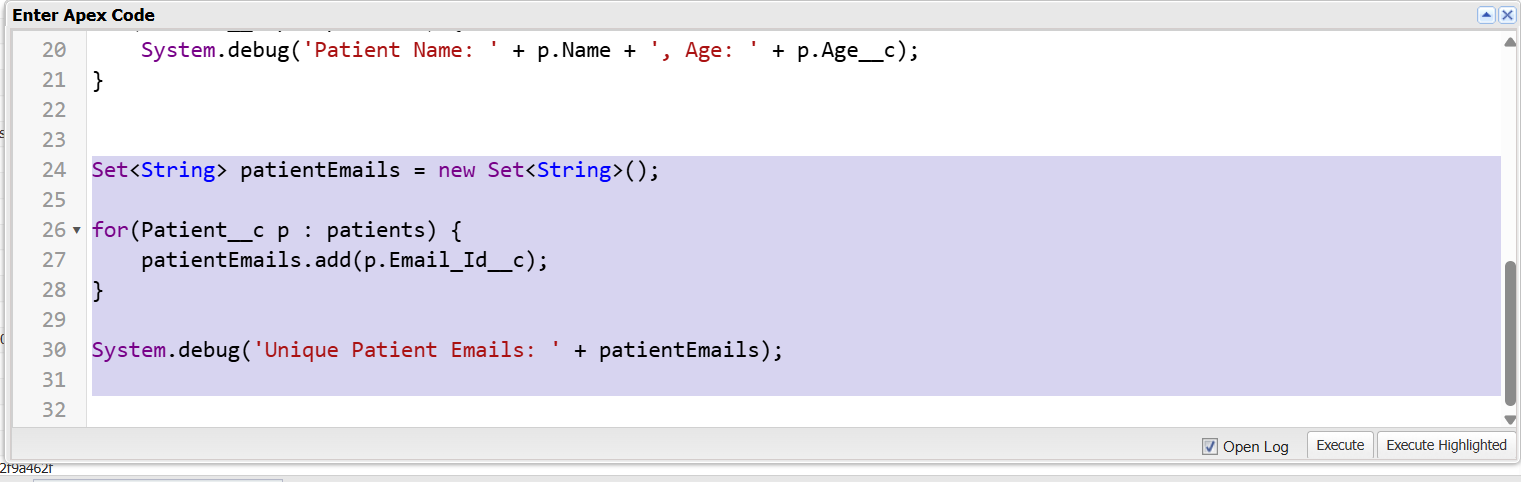
A. List Example

Get all patients above age 30:



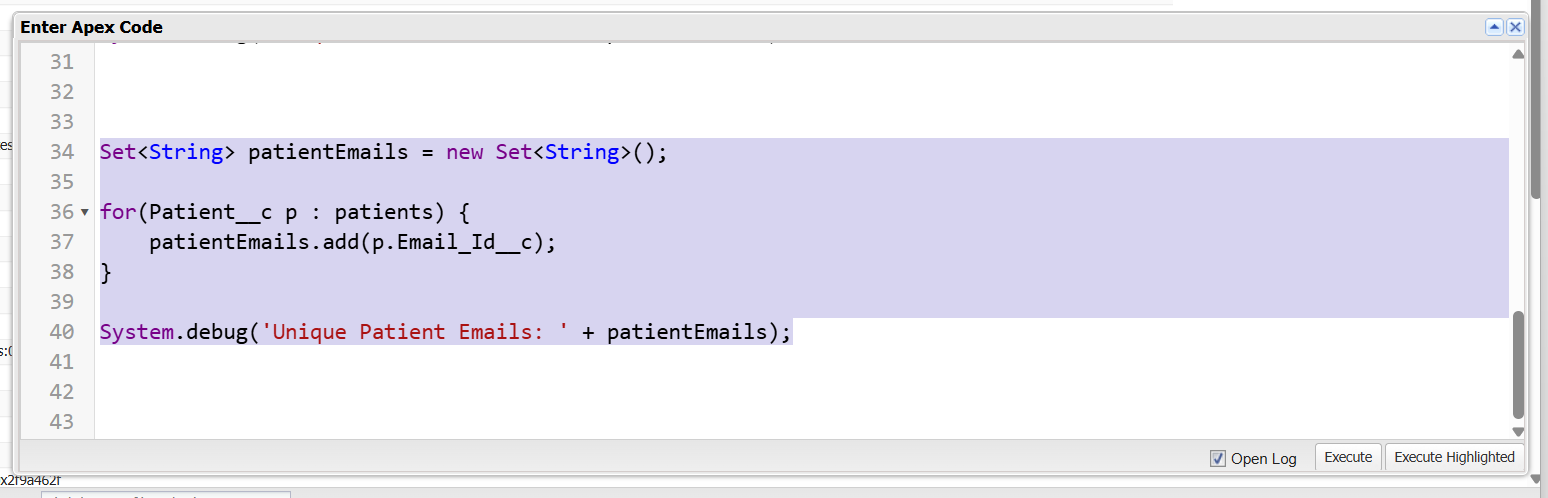
**B. Set Example**

Store unique emails of patients:



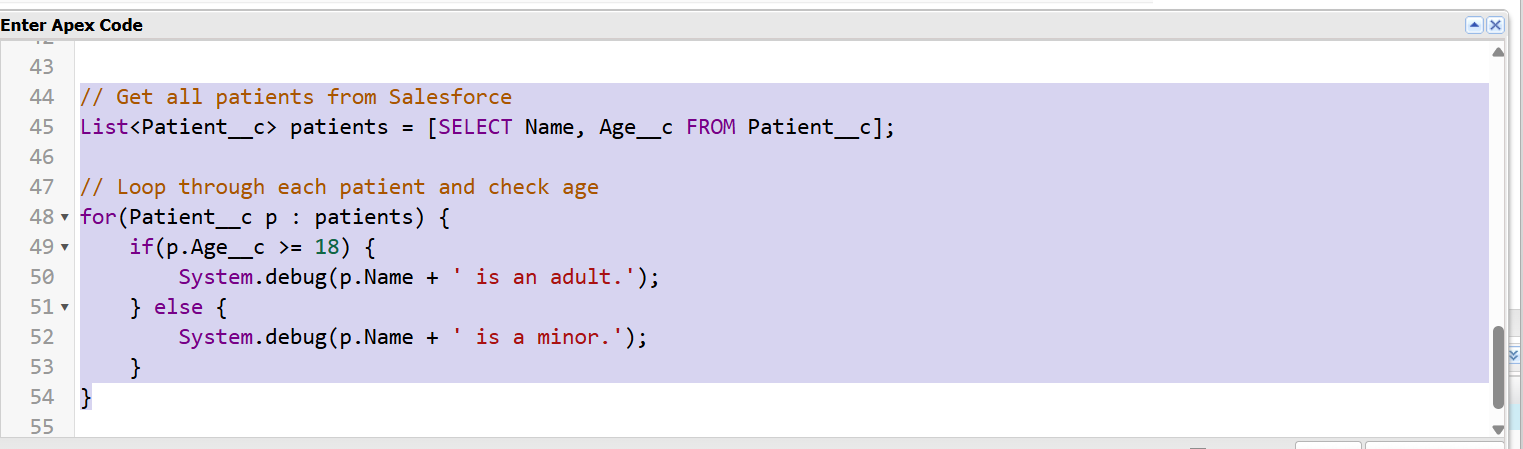
**B. Set Example**

Store unique emails of patients:



**5.Control Statements**

Control Statements in Apex:

* **If / Else:** Execute a block of code based on a condition. Example: Check if a patient is an adult (if(p.Age\_\_c >= 18) { ... } else { ... }).
* **For Loop**: Repeat a block of code for a known number of times. Example: Iterate through a list of patients (for(Patient\_\_c p : patients) { ... }).
* **While / Do-While Loop**: Repeat a block of code until a condition becomes false. Example: Loop through patients while a counter is less than the list size.
* **Switch / Case**: Execute code based on the value of a variable. Example: Handle different patient genders (switch on p.Sex\_\_c { when 'Male' {...} }).
* **Break / Continue**: Control loop execution. break exits the loop early, continue skips the current iteration. Example: Skip minors or stop after finding a specific patient.
* **Example**
* 

**6.Batch Apex**

**Batch Apex →** A way to process large volumes of records asynchronously in Salesforce.

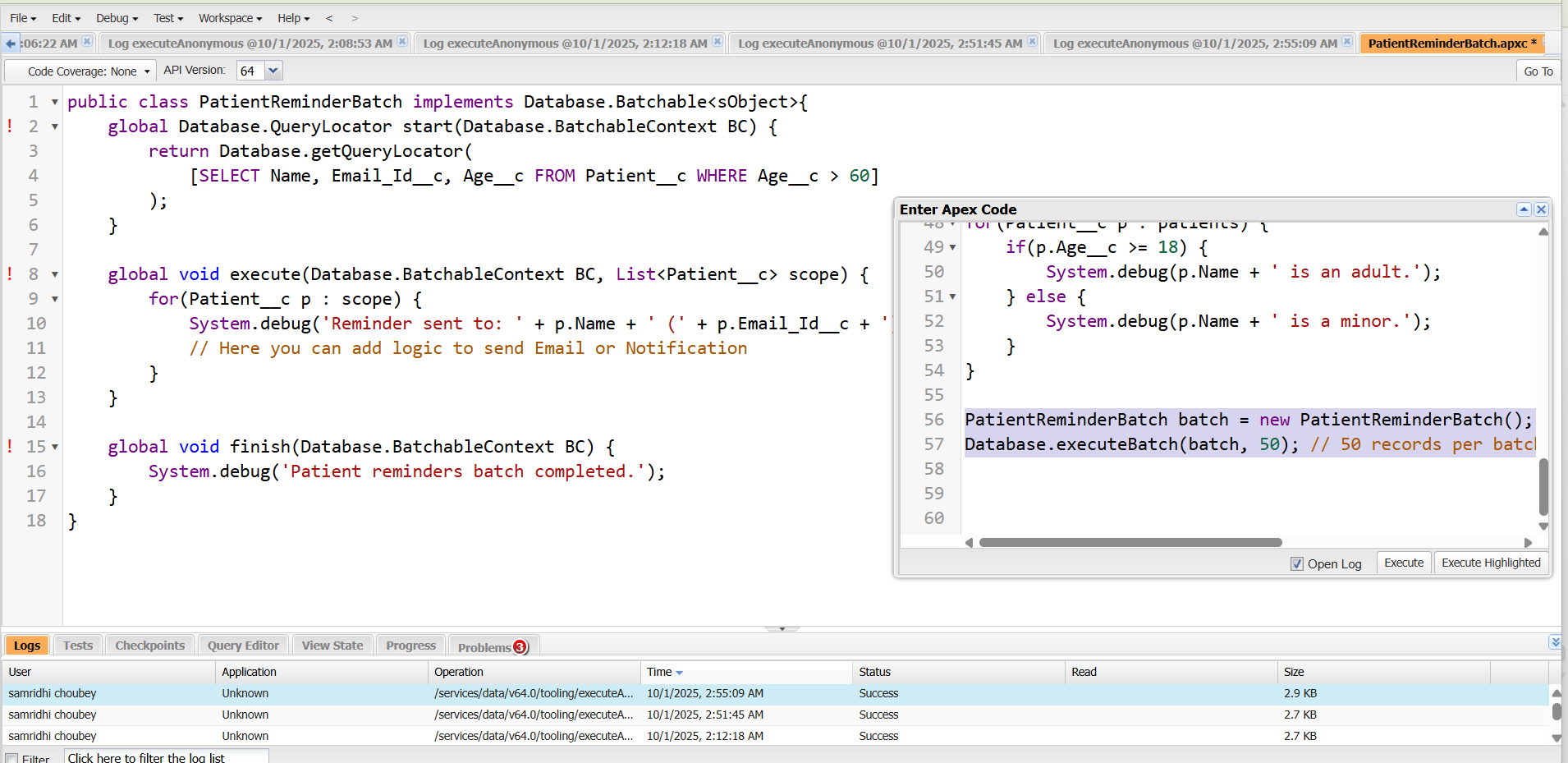
* Useful when you have thousands/millions of records and can’t process them in a single transaction.
* Batch Apex breaks records into smaller chunks (batches) and processes them separately.

**Key Points:**

1. Asynchronous → runs in background
2. Processes large datasets efficiently
3. Uses Database.Batchable interface

**Example:**

**Use Case:** Send reminder to all patients above 60 years for a checkup



**7.Queueable Apex**

Queueable Apex → A way to run asynchronous Apex code in Salesforce, similar to Batch Apex but lighter and more flexible.

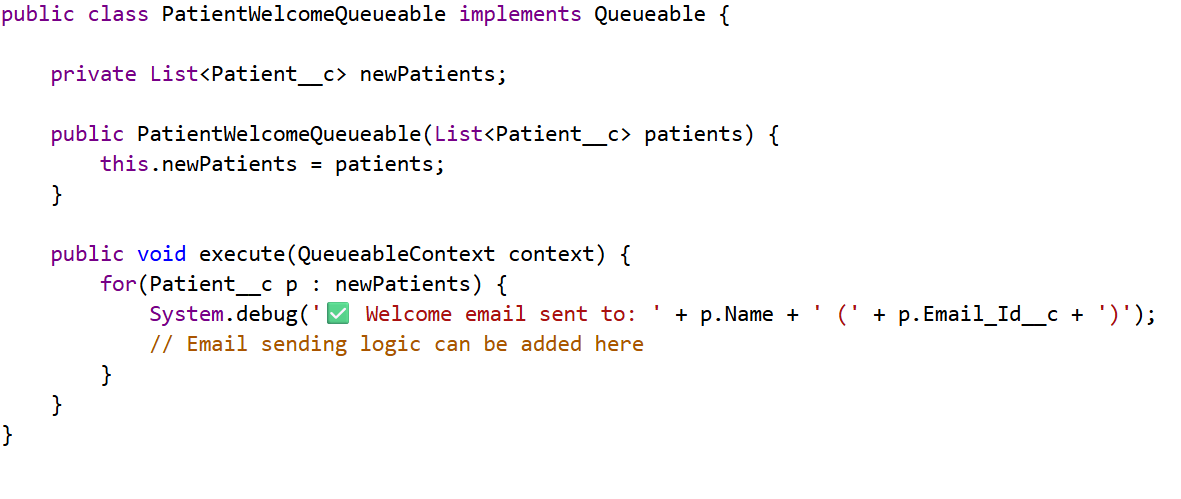
* Use for complex logic that can run in background
* Can chain jobs (one job calls another)
* Easier to implement than Batch Apex for smaller datasets

Key Points:

1. Asynchronous → runs in background
2. Implements Queueable interface
3. Can pass complex objects (like sObjects, custom classes)

Example:

Use Case: Send welcome email to newly registered patients

****

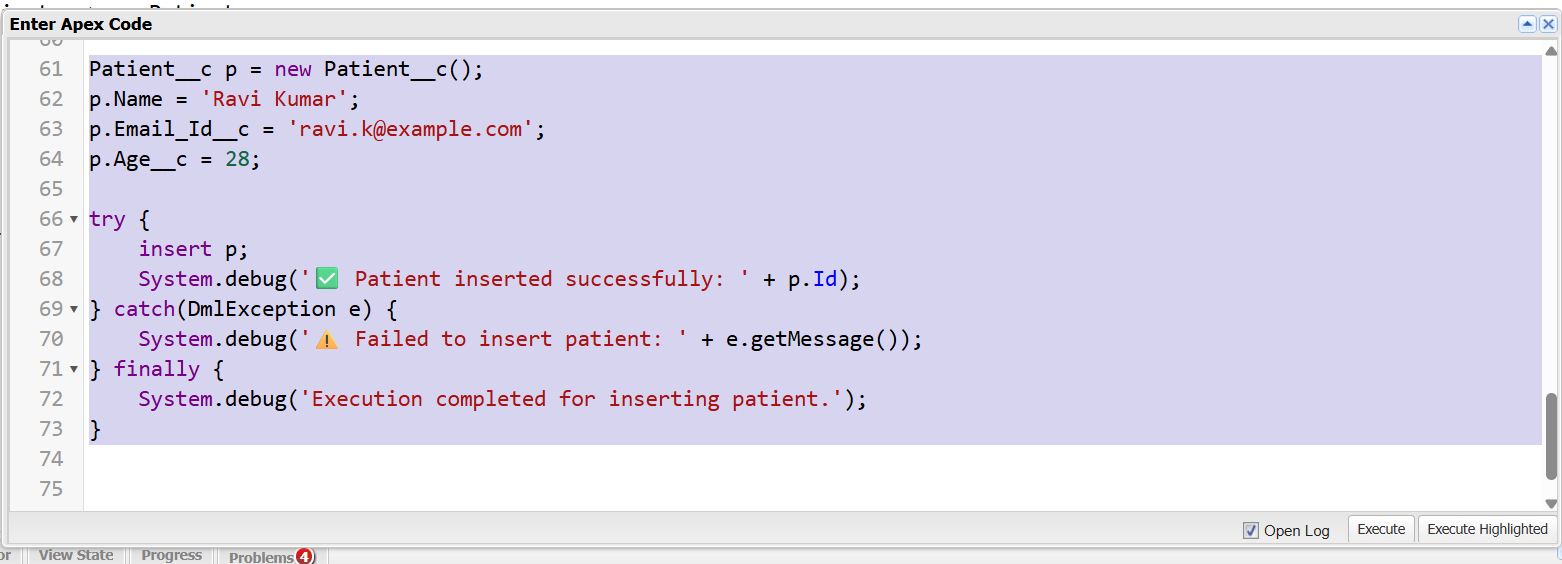
**8.Exception Handling**

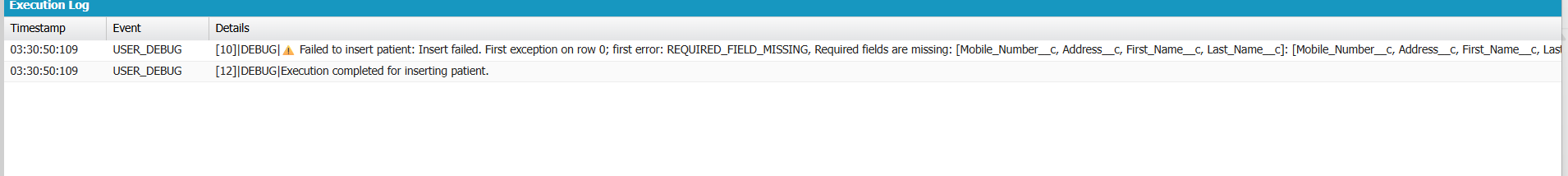
Exception Handling → A way to catch errors in Apex code and handle them gracefully instead of letting the transaction fail.

* Salesforce me errors = Exceptions
* Use try-catch-finally blocks to handle exceptions
* Helps maintain data integrity and avoid failed transactions

**Example:**

**Use Case:** Insert a patient and handle duplicate email error





**9.Test Classes**

Test Classes → Apex classes that verify the correctness of your code.

* Salesforce requires at least 75% code coverage to deploy Apex to production.
* Test classes simulate data and execute logic without affecting real records.
* Always use @isTest annotation for test classes and methods.

