

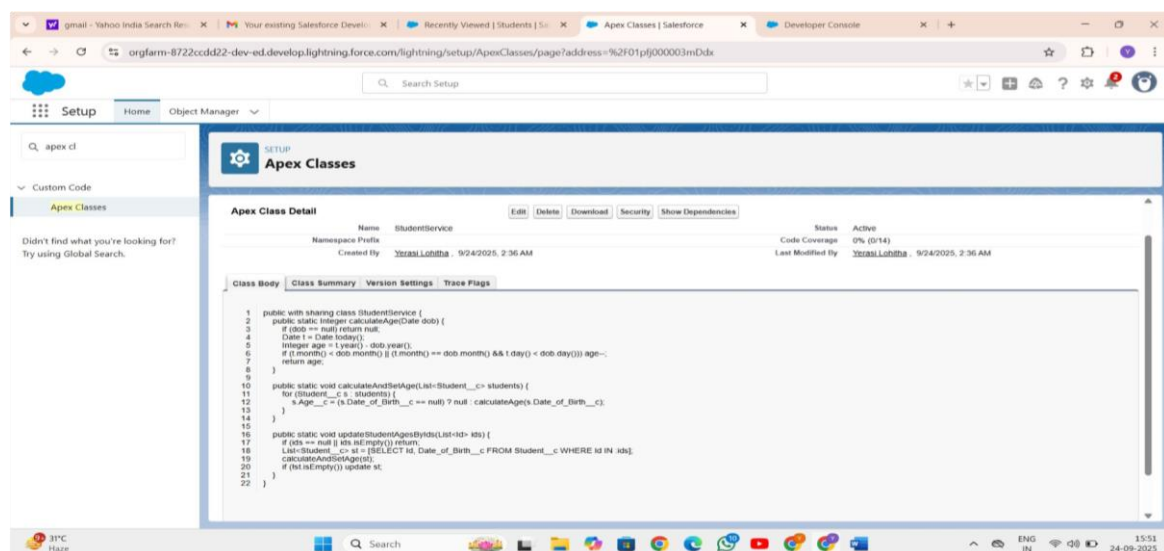
Phase 5: Apex Programming (Developer)

Goal of this Phase

The goal of Phase 5 was to implement **Apex programming features** in Salesforce to support the Student Management System. This included creating **classes, triggers, SOQL/SOSL queries, collections, asynchronous processing, exception handling, and test classes** to ensure the application is efficient, scalable, and reliable.

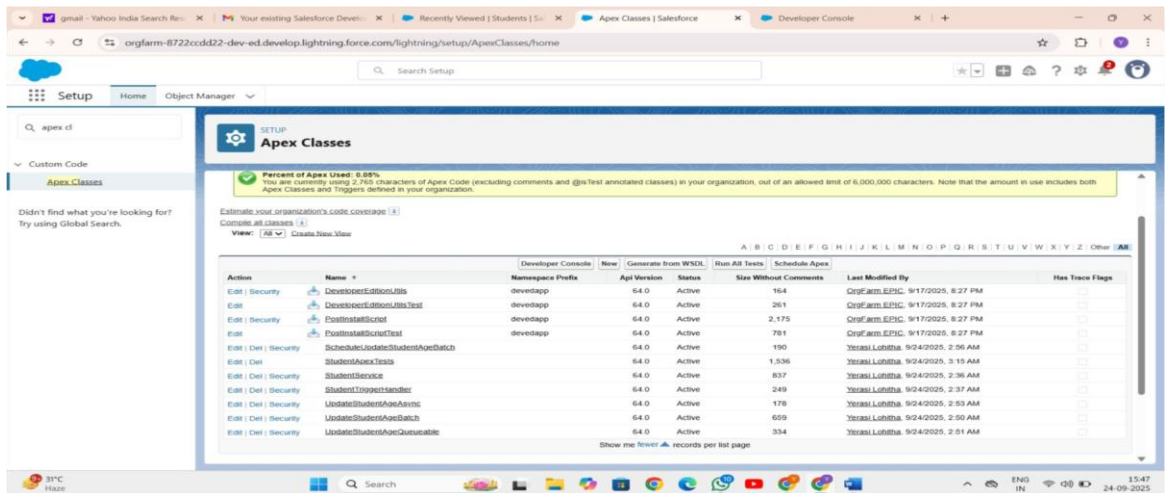
1. Classes & Objects

- Created **StudentService** Apex class to handle business logic.
- Added methods for calculating student age based on Date of Birth and updating records.



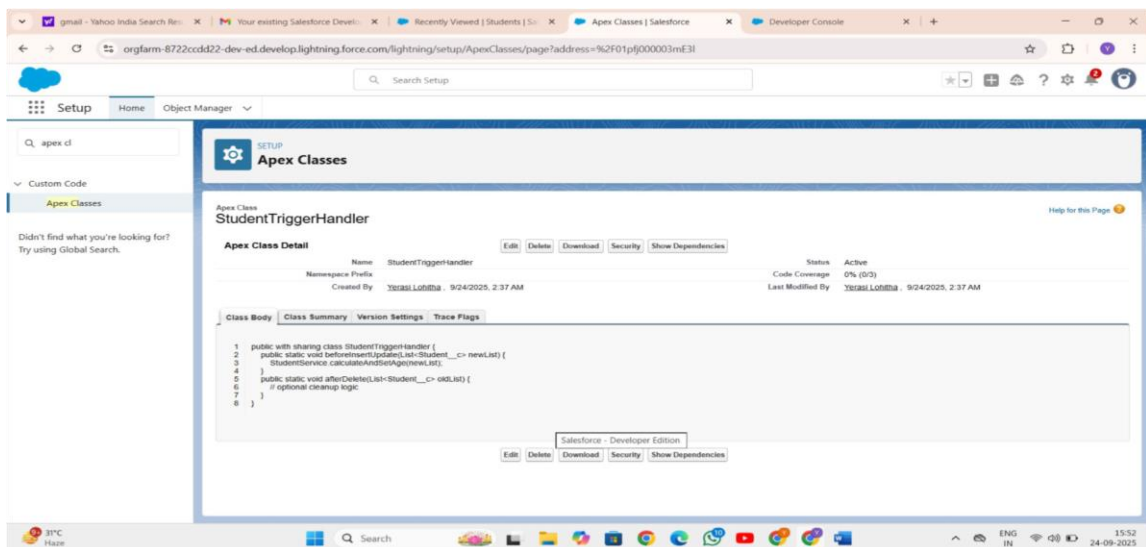
2. Apex Triggers (Before/After Insert/Update/Delete)

- Implemented a **StudentTrigger** to:
 - Calculate Age before insert/update when Date of Birth is set.
 - Prevent invalid updates using before update triggers.
 - Maintain consistency when student records are deleted.



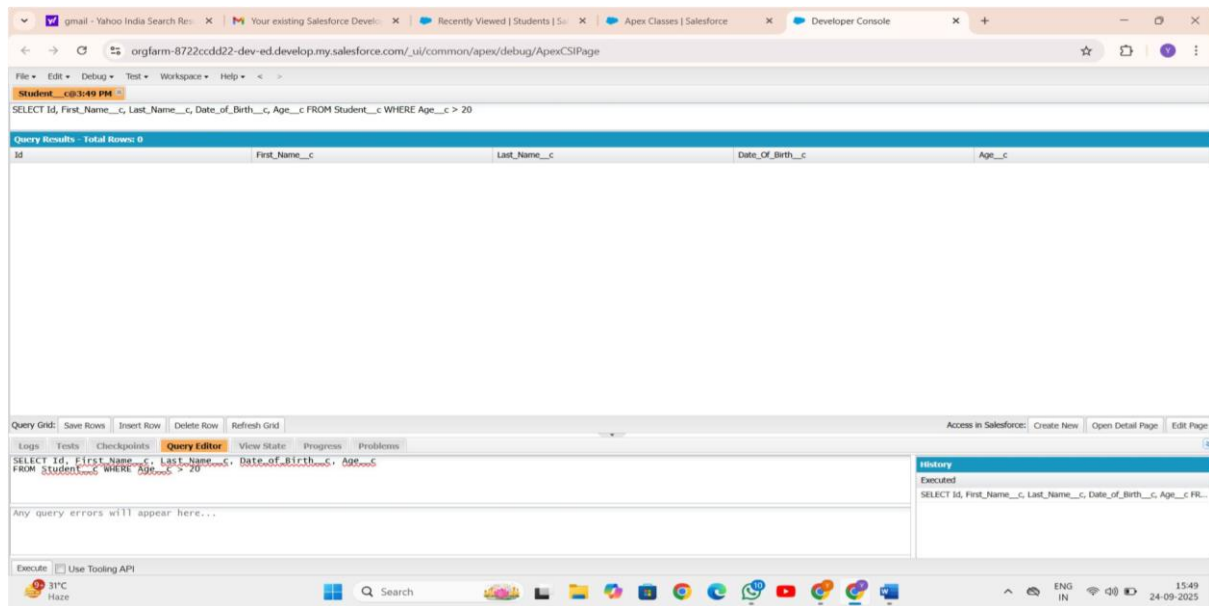
3. Trigger Design Pattern

- Applied **Handler Class Pattern**:
 - Trigger delegates logic to StudentTriggerHandler class.
 - Ensures clean separation of logic, reusable code, and easier maintenance.



4. SOQL & SOSL

- Used **SOQL queries** to fetch Student records with fields like Id, Date_of_Birth__c, and Age__c.
- Demonstrated **SOSL** for searching students by name across multiple fields.

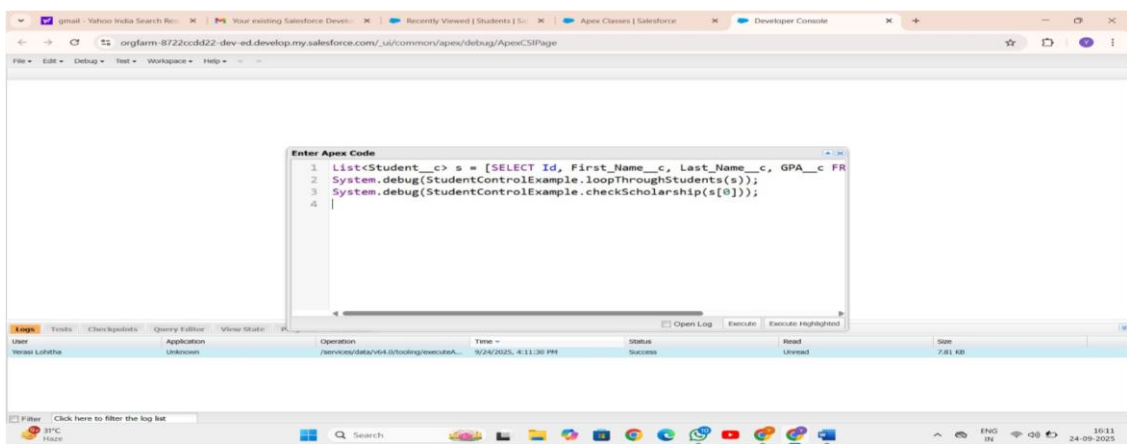
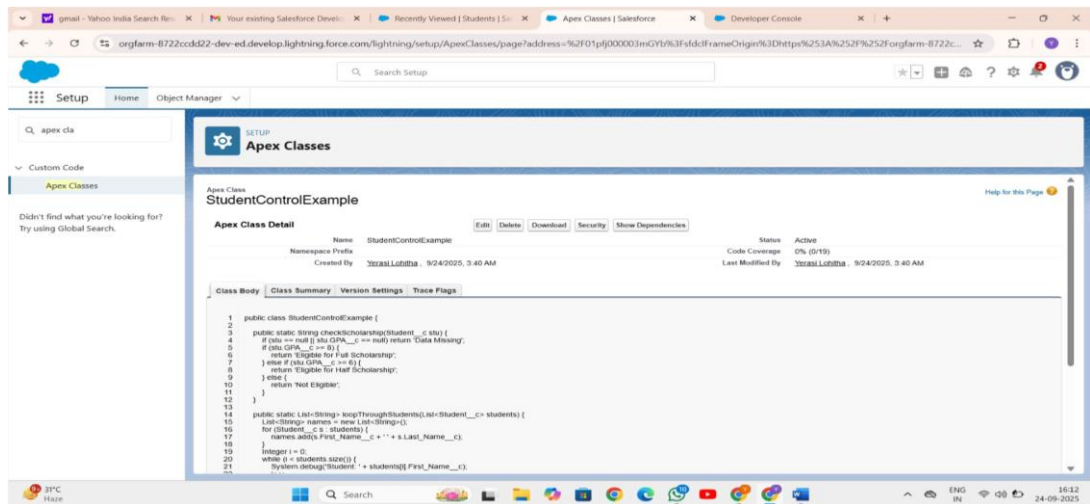


5. Collections (List, Set, Map)

- Used **List<Student__c>** for batch updates.
- Applied **Set<Id>** to handle unique record IDs.
- Implemented **Map<Id, Student__c>** to efficiently compare old and new trigger contexts.

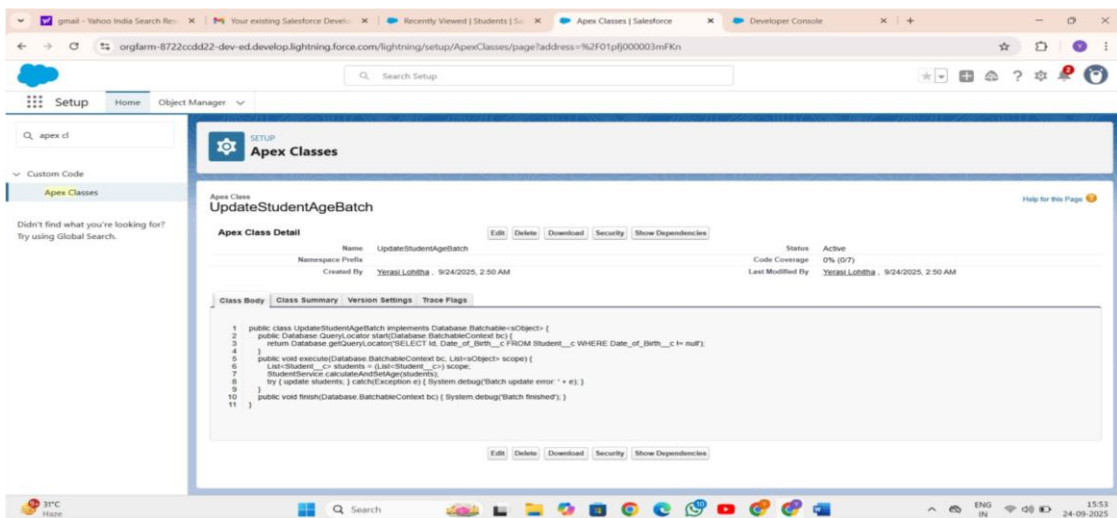
6. Control Statements

- Used **If-Else conditions** for null checks.
- Applied **For loops** to process multiple student records.
- Incorporated **Try-Catch blocks** for exception handling.



7. Batch Apex

- Created **UpdateStudentAgeBatch** class implementing **Database.Batchable<SObject>**.
- Processes students in batches to calculate and update Age.
- Scheduled and monitored using **Apex Jobs**.



8. Queueable Apex

- Developed **UpdateStudentAgeQueueable** class implementing Queueable.
- Allows background execution of student age updates.
- Can be chained for sequential execution.

The image displays two screenshots from the Salesforce Setup interface. The top screenshot shows the 'Apex Classes' page for the class 'UpdateStudentAgeQueueable'. The class is active, with 0% code coverage. The class body is visible, showing a public class that implements the Queueable interface. The bottom screenshot shows the 'Apex Jobs' page, which monitors the status of all Apex jobs. A green banner indicates that 0% of asynchronous Apex is used. Below the banner is a table with columns for Action, Submitted Date, Job Type, Status, Status Detail, Total Batches, Batches Processed, Failures, Submitted By, Completion Date, Apex Class, Apex Method, and Apex Job ID. The table shows four jobs: a Scheduled Apex job, a Future job, a Queueable job, and a Batch Apex job, all of which are completed.

Apex Class Detail

Name	UpdateStudentAgeQueueable
Namespace Prefix	
Created By	Yerasi Lobitha - 9/24/2025, 2:51 AM
Status	Active
Code Coverage	0% (0/4)
Last Modified By	Yerasi Lobitha - 9/24/2025, 2:51 AM

Class Body

```
1 public class UpdateStudentAgeQueueable implements Queueable {  
2     public void executeQueueableContext() {  
3         List<Student__c> students = [SELECT Id, Date_of_Birth__c FROM Student__c WHERE Date_of_Birth__c != null]  
4         StudentService.calculateAndSetAge(students);  
5         if (!students.isEmpty()) update students;  
6     }  
7 }
```

Apex Jobs

Click here to go to the new batch jobs page

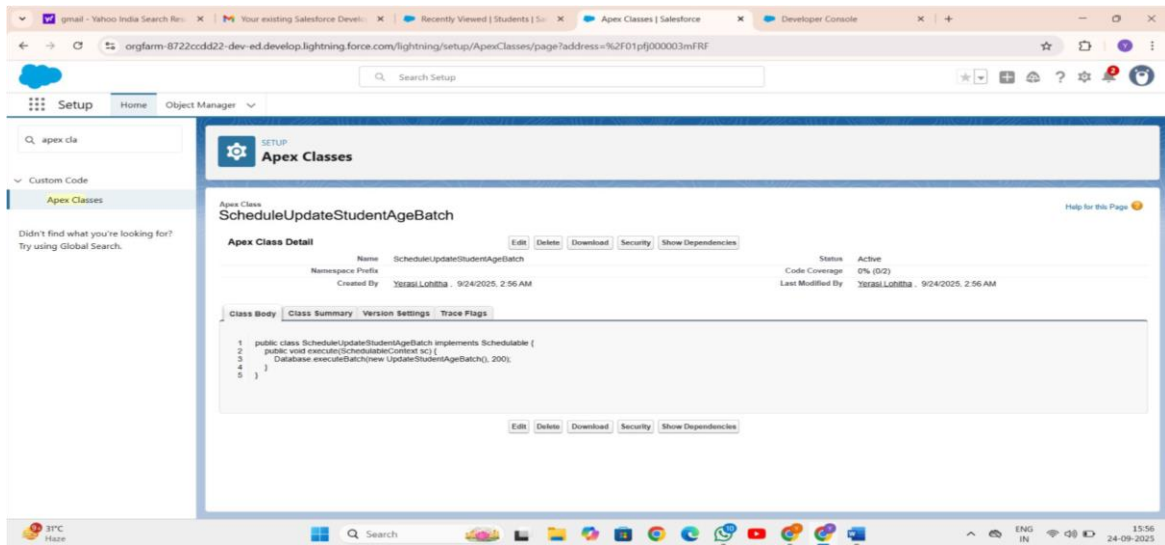
Monitor the status of all Apex jobs, and optionally, abort jobs that are in progress.

Percent of Asynchronous Apex Used: 0%
You have currently used 4 asynchronous Apex operations out of an allowed 24-hour organization limit of 250,000. To learn about how this limit is calculated and what contributes to it, see the [Lightning Platform Apex Limits](#) topic.

Action	Submitted Date	Job Type	Status	Status Detail	Total Batches	Batches Processed	Failures	Submitted By	Completion Date	Apex Class	Apex Method	Apex Job ID
	9/24/2025, 3:05 AM	Scheduled Apex	Queued		0	0	0	0	Lobitha_Yerasi	ScheduleUpdateStudentAgeBatch		707f900000vV40
	9/24/2025, 2:55 AM	Future	Completed		0	0	0	0	Lobitha_Yerasi	UpdateStudentAgeFuture	updateAgeFuture	707f900000vVZ04
	9/24/2025, 2:52 AM	Queueable	Completed		0	0	0	0	Lobitha_Yerasi	UpdateStudentAgeQueueable		707f900000vVYpZ
	9/24/2025, 2:51 AM	Batch Apex	Completed		1	1	0	0	Lobitha_Yerasi	UpdateStudentAgeBatch		707f900000vVYvU
	9/24/2025, 2:51 AM	Batch Apex	Completed		1	1	0	0	Lobitha_Yerasi	UpdateStudentAgeBatch		707f900000vVKpP

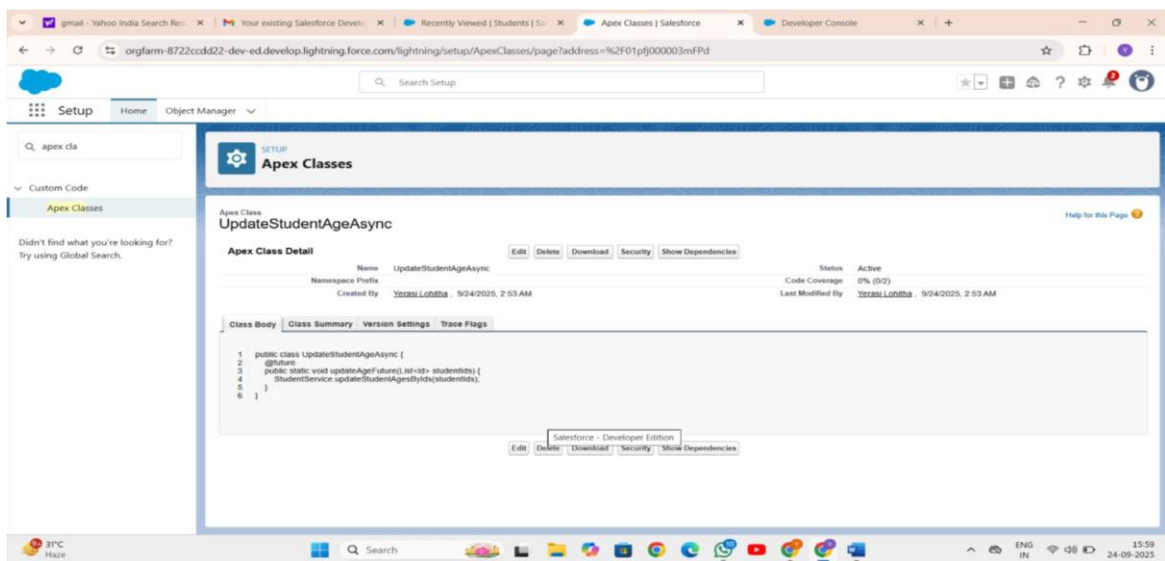
9. Scheduled Apex

- Created **ScheduleUpdateStudentAgeBatch** class implementing Schedulable.
- Scheduled batch jobs to run at predefined times.



10. Future Methods

- Implemented **UpdateStudentAgeFuture** class with **@future** annotation.
- Runs background updates asynchronously for lightweight processing.

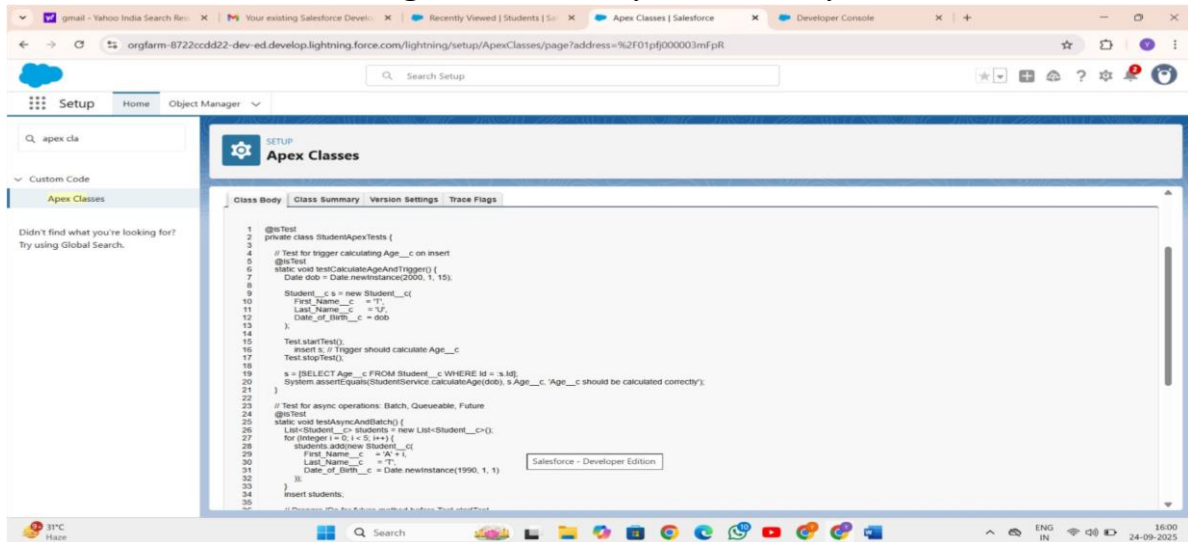


11.ExceptionHandling

- Wrapped logic in **try-catch** blocks.
- Handled null Date of Birth scenario to prevent runtime errors.
- Ensured no unhandled exceptions crash the system.

12. Test Classes

- Created **TestStudentService** and other test classes with **@isTest**.
- Verified Batch, Queueable, Future, and Exception Handling methods.
- Achieved **code coverage** and validated system reliability.



13. Asynchronous Processing

- Implemented and tested all four types: **Batch, Queueable, Scheduled, and Future Methods**.
- Verified job execution via **Apex Jobs** in Setup.
- Ensured scalability for handling large volumes of Student records.