

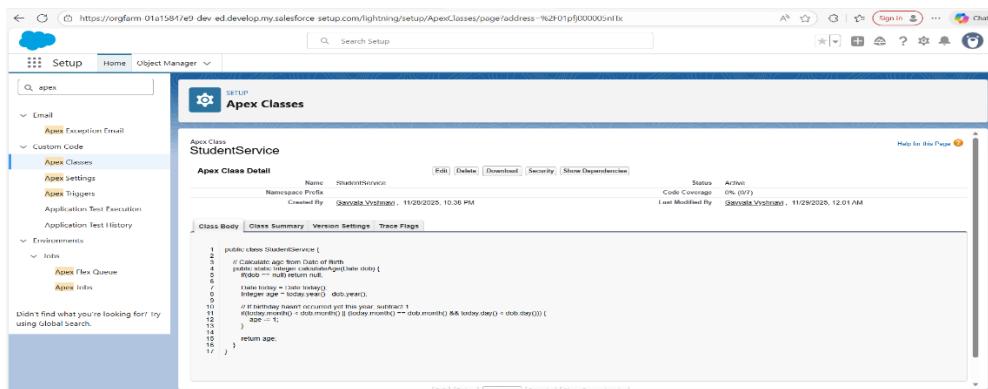
# Phase 5: Apex Programming (Developer)

## 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.

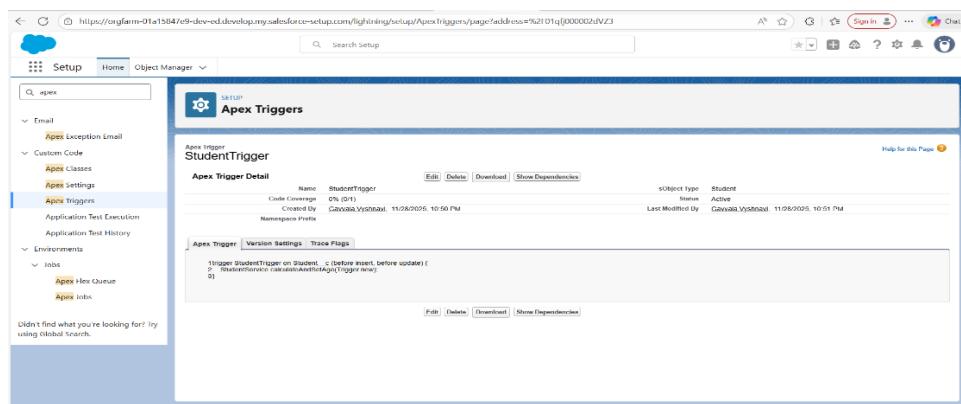
## 1. Classes & Objects

- 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)

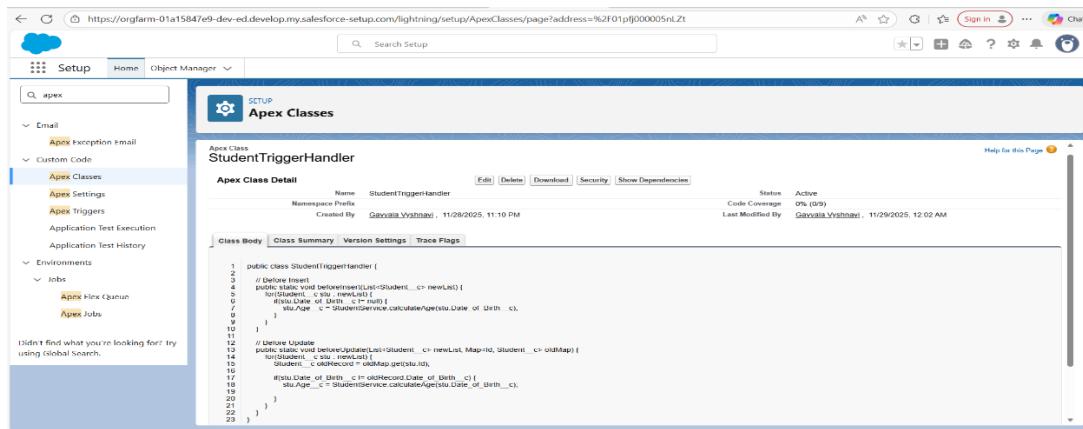
- Implemented a **StudentTrigger** to:
  - Calculate Age before insert/update when Date of Birth is set.
  - Prevent invalid updates using before update triggers.
  - Maintain consistency when student records are deleted.



### 3. Trigger Design Pattern

- Applied Handler Class Pattern:

- Trigger delegates logic to StudentTriggerHandler class.
- Ensures clean separation of logic, reusable code, and easier maintenance.

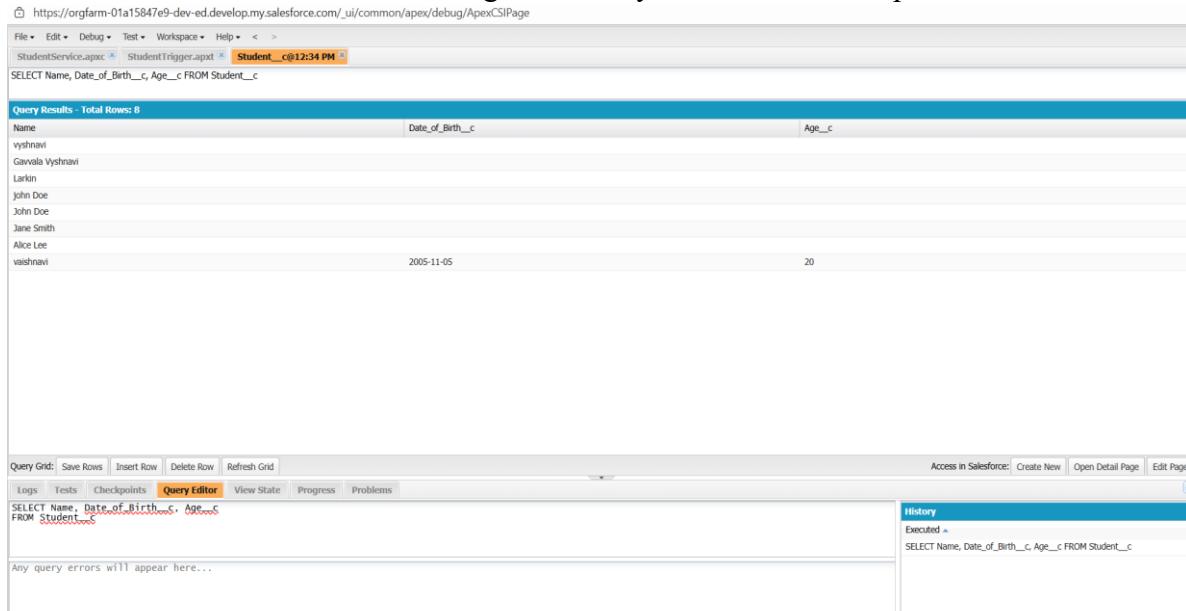


The screenshot shows the Salesforce Setup interface with the search bar set to 'apex'. The 'Apex Classes' section is selected in the left sidebar. A specific class, 'StudentTriggerHandler', is displayed in the main area. The class details show it was created by 'Gavala Vyshnavi' on '11/28/2025, 11:10 PM'. The status is 'Active' with 0% code coverage. The class body contains the following Apex code:

```
1 public class StudentTriggerHandler {
2     // Trigger Handler
3     public static void beforeInsert(List<Student__c> newList) {
4         for(Student__c stu : newList) {
5             stu.Date_of_Birth__c = null;
6             stu.Age__c = StudentService.calculateAge(stu.Date_of_Birth__c);
7         }
8     }
9 }
10 // Update
11 public static void beforeUpdate(List<Student__c> newList, Map<Id, Student__c> oldMap) {
12     for(Student__c stu : newList) {
13         if(stu.Date_of_Birth__c != null) {
14             stu.Age__c = StudentService.calculateAge(stu.Date_of_Birth__c);
15         }
16     }
17 }
18
19
20
21
22 }
```

### 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.



The screenshot shows the Salesforce Query Editor interface. At the top, the URL is https://orgfarm-01a15847e9-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSPage. The query being run is:

```
SELECT Name, Date_of_Birth__c, Age__c FROM Student__c
```

The 'Query Results - Total Rows: 8' section displays the following data:

Name	Date_of_Birth__c	Age__c
vyshnavi		
Gavala Vyshnavi		
Larkin		
John Doe		
John Doe		
Jane Smith		
Alice Lee		
vaishnavi	2005-11-05	20

Below the results, the 'Logs' tab shows the executed query:

```
SELECT Name, Date_of_Birth__c, Age__c
FROM Student__c
```

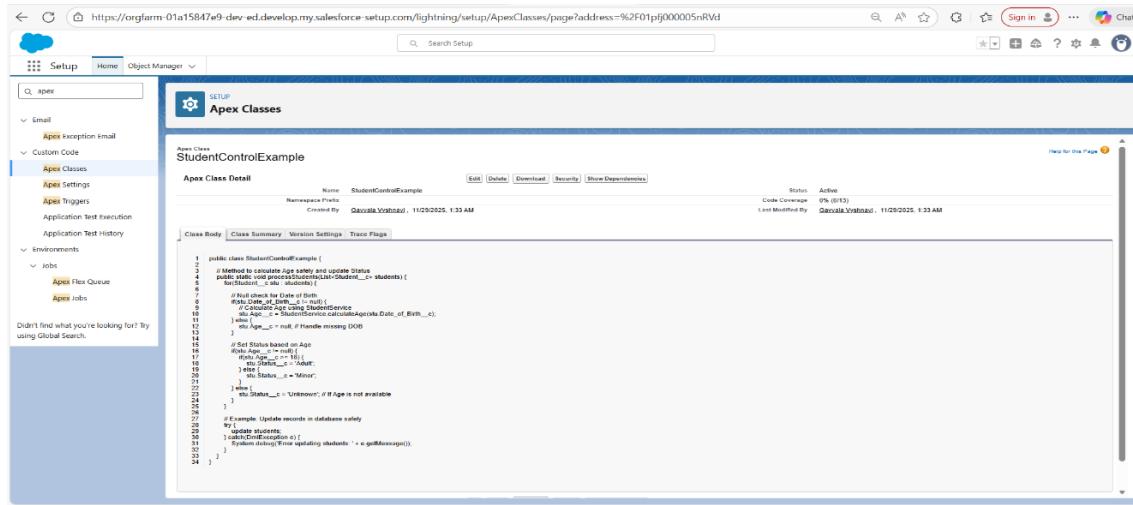
The 'History' tab shows the same query listed under 'Executed'.

## 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.



The screenshot shows the Salesforce Setup Apex Classes page. The URL is https://orgfarm-01a15847e9-dev-ed.develop.my.salesforce.com/lightning/setup/ApexClasses/page?address=%2F01p0j000005nRvD. The page title is "Apex Classes". On the left sidebar, under "Custom Code", "Apex Classes" is selected. The main content area displays the code for "StudentControlExample".

```
Apex Class Detail
Name: StudentControlExample
Namespace Prefix: 
Created By: Gavazla Vrashabha | 11/29/2025, 1:33 AM
Status: Active
Code Coverage: 9% (0/13)
Last Modified By: Gavazla Vrashabha | 11/29/2025, 1:33 AM

Class Body | Class Summary | Version Settings | Trace Flags

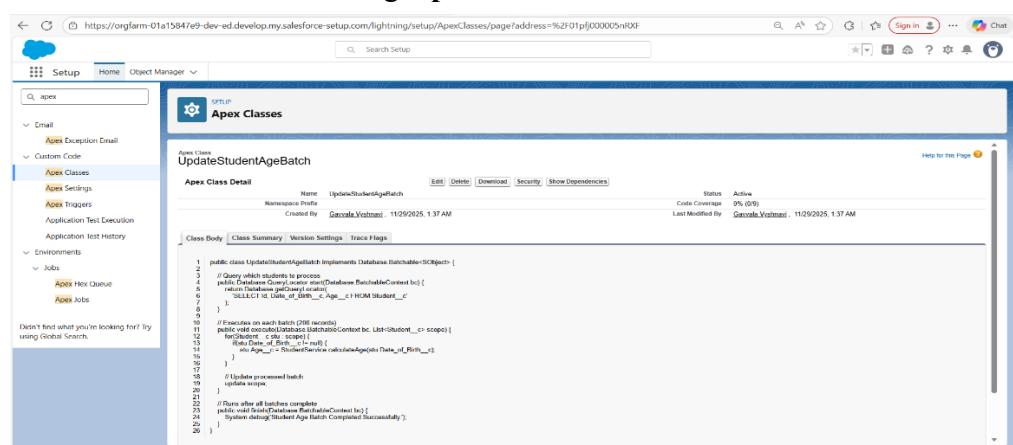
public class StudentControlExample {
    // Method to calculate Age batch and update Status
    public static void processStudents(List<Student__c> students) {
        for (Student__c student : students) {
            if (student.Date_of_Birth__c == null) {
                StudentService.calculateAge(student);
            } else {
                student.Age__c = StudentService.calculateAge(student.Date_of_Birth__c);
            }
        }
    }

    // Set Status based on Age
    public static void updateStatus(List<Student__c> students) {
        for (Student__c student : students) {
            if (student.Age__c < null) {
                student.Status__c = 'Missing';
            } else if (student.Age__c < 18) {
                student.Status__c = 'Minor';
            } else {
                student.Status__c = 'Unknown'; // If age is not available
            }
        }
    }

    // Example: Update records in database safely
    public static void updateStudents() {
        List<Student__c> students = new List<Student__c>();
        System.debug('Before updating students: ' + students);
        try {
            Database.update(students, false);
        } catch (DatabaseException e) {
            System.debug('Error updating students: ' + e.getMessage());
        }
    }
}
```

## 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**.



The screenshot shows the Salesforce Setup Apex Classes page. The URL is https://orgfarm-01a15847e9-dev-ed.develop.my.salesforce.com/lightning/setup/ApexClasses/page?address=%2F01p0j000005nRxF. The page title is "Apex Classes". On the left sidebar, under "Custom Code", "Apex Classes" is selected. The main content area displays the code for "UpdateStudentAgeBatch".

```
Apex Class Detail
Name: UpdateStudentAgeBatch
Namespace Prefix: 
Created By: Gavazla Vrashabha | 11/29/2025, 1:37 AM
Status: Active
Code Coverage: 0% (0/0)
Last Modified By: Gavazla Vrashabha | 11/29/2025, 1:37 AM

Class Body | Class Summary | Version Settings | Trace Flags

public class UpdateStudentAgeBatch implements Database.Batchable<Student__c> {
    // Query
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Date_of_Birth__c, Age__c FROM Student__c');
    }

    // Process
    public void execute(Database.BatchableContext bc, List<Student__c> scope) {
        for (Student__c student : scope) {
            if (student.Date_of_Birth__c != null) {
                student.Age__c = StudentService.calculateAge(student.Date_of_Birth__c);
            }
        }
    }

    // Finish
    public void finish(Database.BatchableContext bc) {
        System.debug('Update processed batch');
        update scope;
    }

    // Debug
    public void afterAll(Database.BatchableContext bc) {
        System.debug('After all batches complete');
        System.debug('Database.BatchableContext bc');
        System.debug('Scope');
    }
}
```

## 8. Queueable Apex

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

The screenshot shows the Salesforce Setup Apex Classes page. The URL is https://orgfarm-01a15847e9-dev-ed.develop.my.salesforce-setup.com/lightning/setup/ApexClasses/page?address=%2F01pfj000005nRyr. The page title is "Apex Classes". The left sidebar has a search bar and navigation links for Email, Custom Code, Apex Classes, Apex Settings, Apex Triggers, Application Test Execution, Application Test History, Environments, and Jobs. Under "Jobs", "Apex Flex Queue" and "Apex Jobs" are listed. The main content area shows the "Apex Class Detail" for "UpdateStudentAgeQueueable". The class body contains the following code:

```
1 public class UpdateStudentAgeQueueable implements Queueable {
2     public void execute(QueueableContext context) {
3         //Query all student records
4         List<Student__c> students = [
5             SELECT Id, Date_of_Birth__c, Age__c
6             FROM Student__c
7         ];
8
9         //Process each student
10        for(Student__c stu : students) {
11            if(stu.Date_of_Birth__c != null) {
12                stu.Age__c = StudentService.calculateAge(stu.Date_of_Birth__c);
13            }
14        }
15
16        //Update students
17        update students;
18
19        System.debug('Queueable Job Completed Successfully');
20    }
21 }
22 }
```

## 9. Scheduled Apex

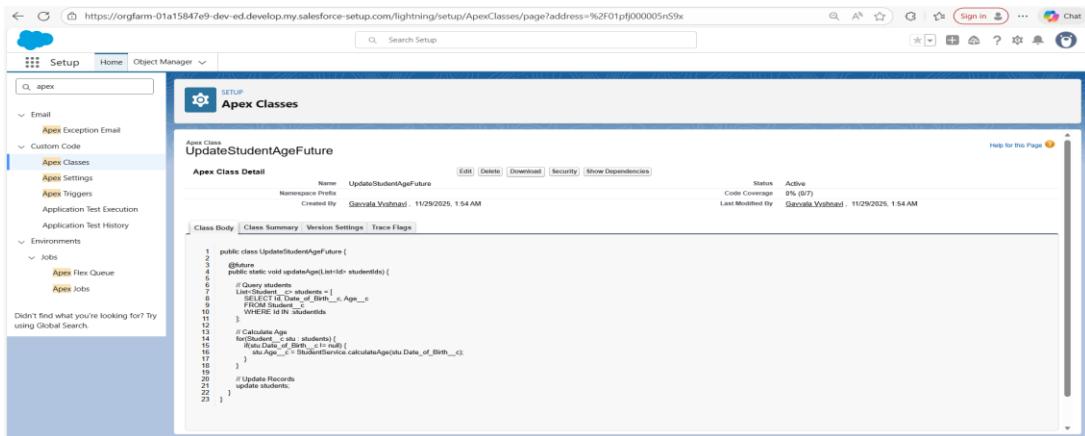
- Created **ScheduleUpdateStudentAgeBatch** class implementing Scheduledable.
- Scheduled batch jobs to run at predefined times.

The screenshot shows the Salesforce Setup Apex Classes page. The URL is https://orgfarm-01a15847e9-dev-ed.develop.my.salesforce-setup.com/lightning/setup/ApexClasses/page?address=%2F01pfj000005nS3V. The page title is "Apex Classes". The left sidebar is identical to the previous screenshot. The main content area shows the "Apex Class Detail" for "ScheduleUpdateStudentAgeBatch". The class body contains the following code:

```
1 public class ScheduleUpdateStudentAgeBatch implements Scheduledable {
2     public void execute(SchedulableContext sc) {
3         // Call your Batch Class
4         UpdateStudentAgeBatch b = new UpdateStudentAgeBatch();
5         Database.executeBatch(b, 200);
6     }
7 }
```

## 10. Future Methods

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



The screenshot shows the Salesforce Setup Apex Classes page. The search bar at the top has "apex" typed into it. On the left, there's a sidebar with sections like Email, Custom Code, Apex Classes, Apex Settings, Apex Triggers, Application Test Execution, Application Test History, Environments, and Jobs. The "Apex Classes" section is selected. In the main area, the title is "Apex Class Detail" for "UpdateStudentAgeFuture". The class body contains the following code:

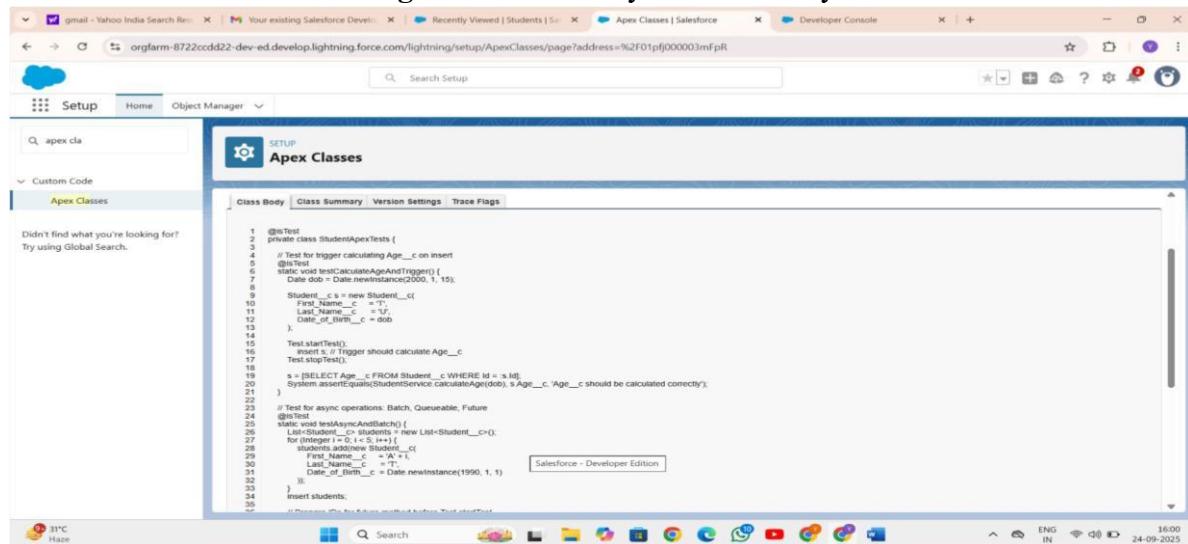
```
1 public class UpdateStudentAgeFuture {
2     @future
3     public static void updateAge(List<Student__c> students) {
4         List<Student__c> students = [
5             SELECT Age__c, Date_of_Birth__c FROM Student__c
6             WHERE Id IN :ids
7         ];
8
9         // Calculate Age
10        for(Student__c student : students) {
11            if(student.Date_of_Birth__c != null) {
12                student.Age__c = StudentService.calculateAge(student.Date_of_Birth__c);
13            }
14        }
15
16        // Update Records
17        update students;
18    }
19 }
20
21 }
```

## 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.



The screenshot shows the Salesforce Setup Apex Classes page. The search bar at the top has "apex cla" typed into it. On the left, there's a sidebar with sections like Email, Custom Code, and Apex Classes. The "Apex Classes" section is selected. In the main area, the title is "Class Body" for "StudentApexTests". The class body contains the following code:

```
1 @isTest
2 private class StudentApexTests {
3
4     // Test for trigger calculating Age__c on insert
5     @isTest
6     static void testCalculateAgeAndTrigger() {
7         Date dob = Date.newInstance(2000, 1, 15);
8
9         Student__c s = new Student__c();
10        s.Last_Name__c = 'V';
11        s.Date_of_Birth__c = dob;
12
13        Test.startTest();
14        insert s; // Trigger should calculate Age__c
15        Test.stopTest();
16
17        s = [SELECT Age__c FROM Student__c WHERE Id = :s.Id];
18        System.assertEqual(StudentService.calculateAge(dob), s.Age__c, 'Age__c should be calculated correctly');
19
20    }
21
22    // Test for async operations: Batch, Queueable, Future
23    @isTest
24    static void testAsyncOperations() {
25        List<Student__c> students = new List<Student__c>();
26
27        for(Integer i = 0; i < 1000; i++) {
28            students.add(new Student__c(
29                Last_Name__c = 'A' + i,
30                Last_Name__c = 'B' + i,
31                Date_of_Birth__c = Date.newInstance(1990, 1, 1)
32            ));
33        }
34        insert students;
35    }
36}
```

### **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.