

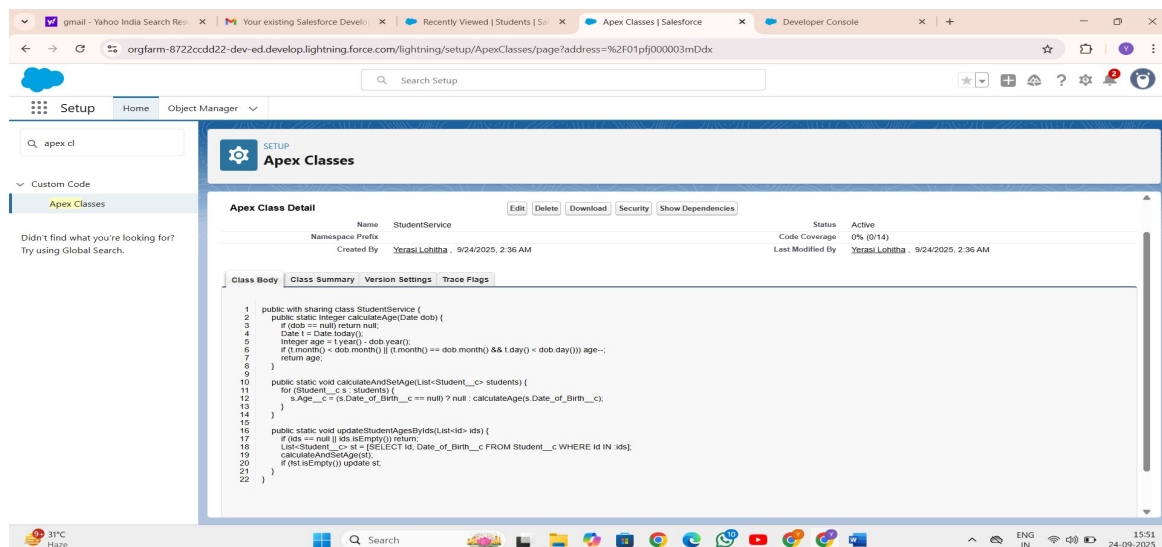
Phase 5: Apex Programming (Developer)

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.

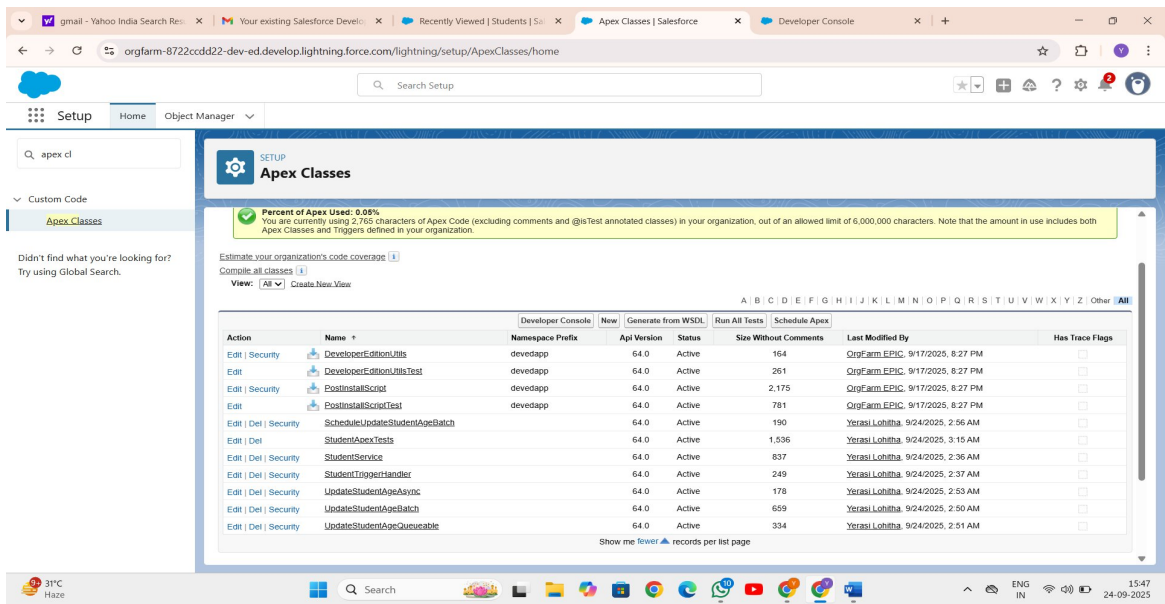
1. Classes & Objects

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



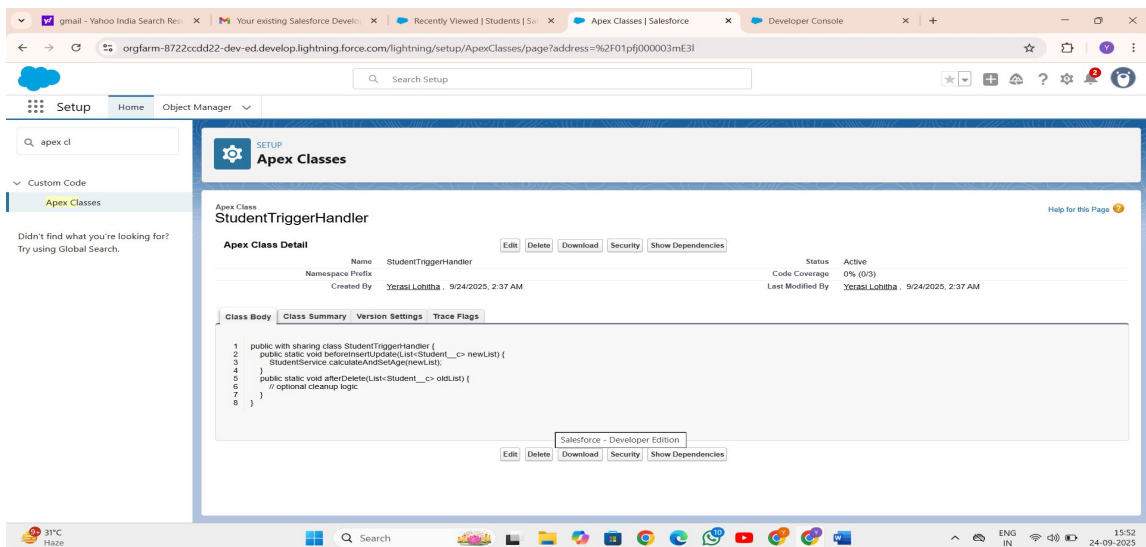
2. Apex Triggers (Before/After Insert/Update/Delete)

- 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.



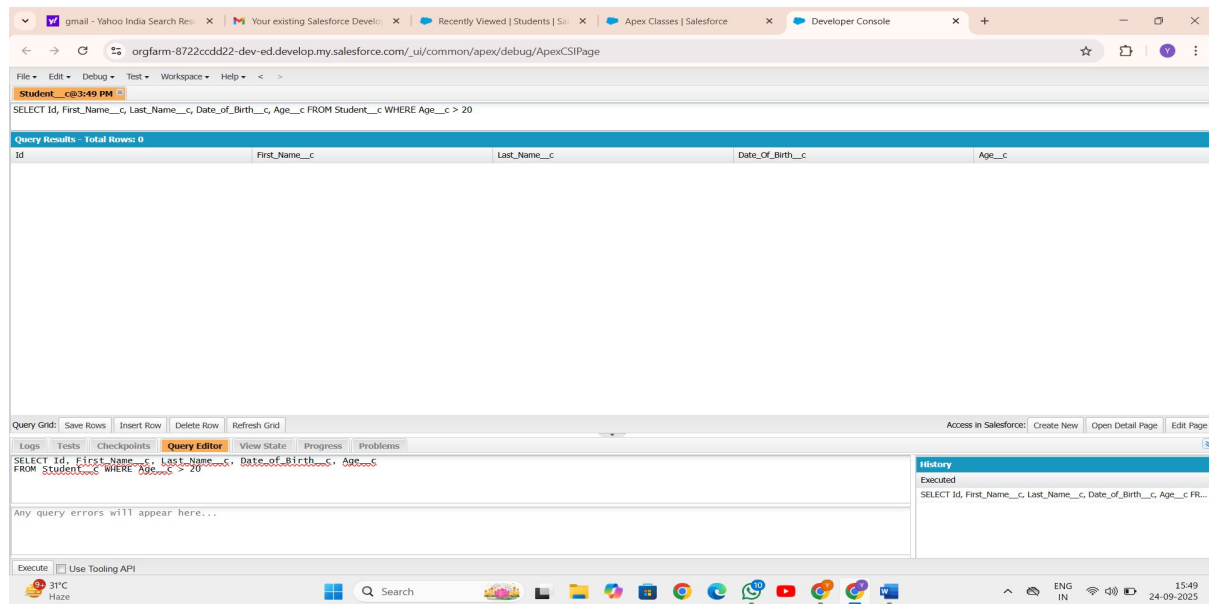
3. Trigger Design Pattern

- 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.

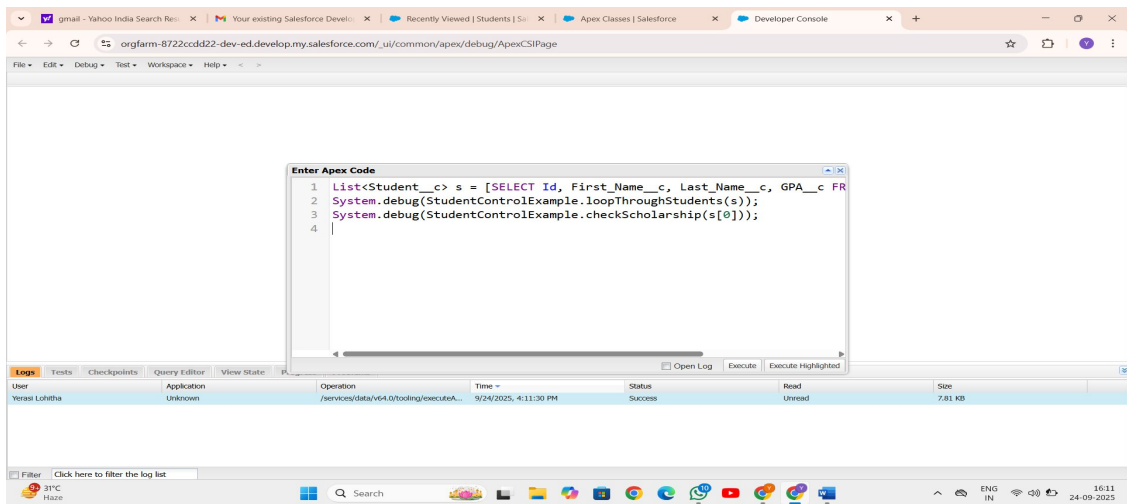
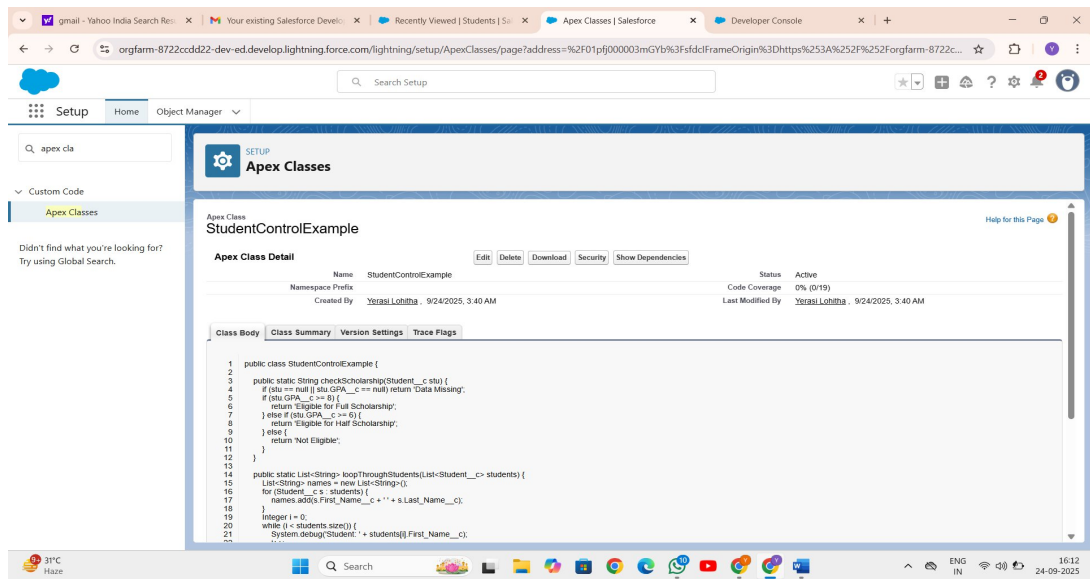


5. Collections (List, Set, Map)

- 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.

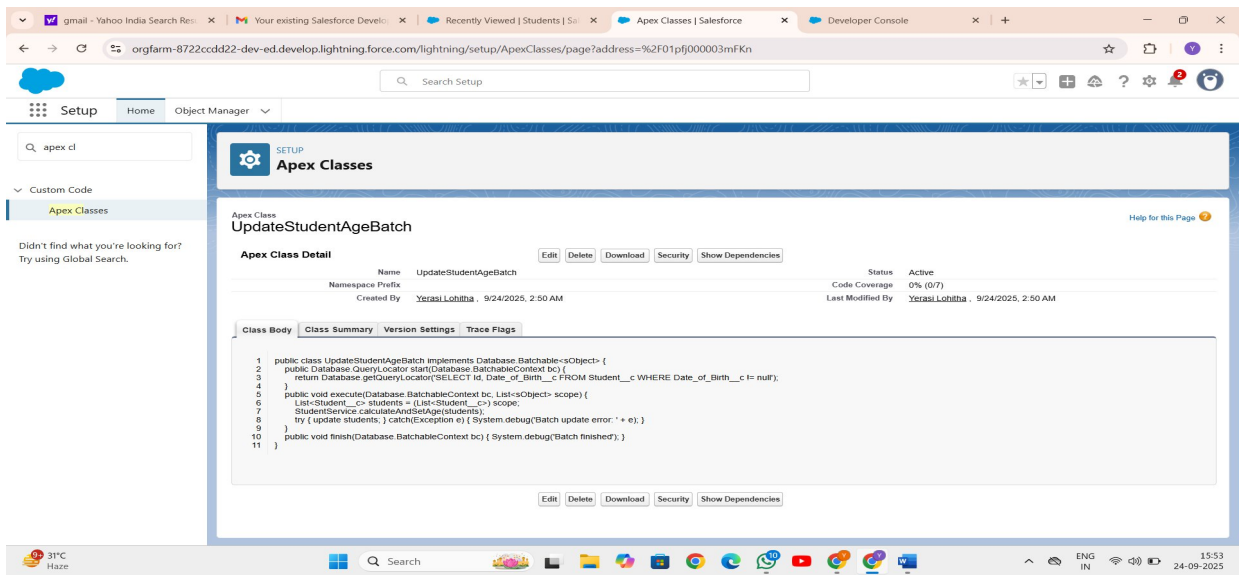
6. Control Statements

- 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**.



The screenshot shows the Salesforce Apex Classes page for the class **UpdateStudentAgeBatch**. The class is implemented as a **Database.Batchable<SObject>**. The code is as follows:

```

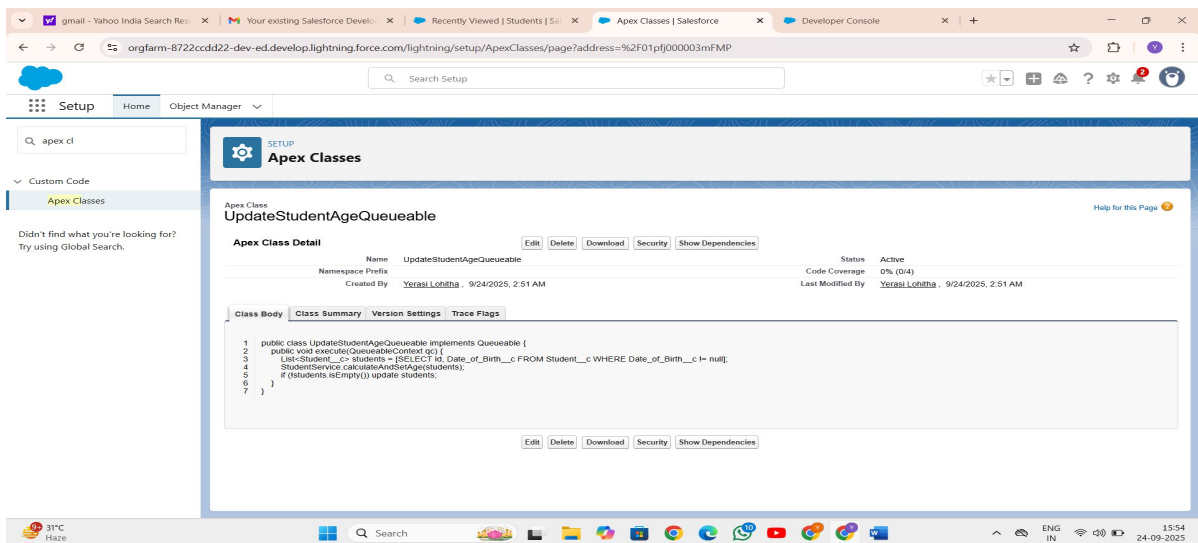
1 public class UpdateStudentAgeBatch implements Database.Batchable<SObject> {
2     public Database.QueryLocator start(Database.BatchableContext bc) {
3         return Database.getQueryLocator(SELECT Id, Date_of_Birth__c FROM Student__c WHERE Date_of_Birth__c != null);
4     }
5     public void execute(Database.BatchableContext bc, List<SObject> scope) {
6         List<Student__c> students = (List<Student__c>) scope;
7         StudentService.calculateAndSetAge(students);
8         try { update students; } catch (Exception e) { System.debug('Batch update error: ' + e); }
9     }
10    public void finish(Database.BatchableContext bc) { System.debug('Batch finished'); }
11 }

```

The class is created by Yerasi Lohitha on 9/24/2025 at 2:50 AM. The status is Active, and the code coverage is 0% (0/7).

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 Apex Classes page for the class **UpdateStudentAgeQueueable**. The class is implemented as a **Queueable**. The code is as follows:

```

1 public class UpdateStudentAgeQueueable implements Queueable {
2     public void execute(QueueableContext qc) {
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 }

```

The class is created by Yerasi Lohitha on 9/24/2025 at 2:51 AM. The status is Active, and the code coverage is 0% (0/4).

Apex Jobs

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, 3:00 AM	Scheduled Apex	Queued		0	0	0	Lobitha_Yerasi	9/24/2025, 2:55 AM	ScheduleUpdateStudentAgeBatch		707f900005V14S
	9/24/2025, 2:55 AM	Future	Completed		0	0	0	Lobitha_Yerasi	9/24/2025, 2:52 AM	UpdateStudentAgeAsync	updateAgeFuture	707f900005VZGd
	9/24/2025, 2:52 AM	Queueable	Completed		0	0	0	Lobitha_Yerasi	9/24/2025, 2:52 AM	UpdateStudentAgeQueueable		707f900005VpPz
	9/24/2025, 2:51 AM	Batch Apex	Completed		1	1	1	Lobitha_Yerasi	9/24/2025, 2:51 AM	UpdateStudentAgeBatch		707f900005VYhu
	9/24/2025, 2:51 AM	Batch Apex	Completed		1	1	1	Lobitha_Yerasi	9/24/2025, 2:51 AM	UpdateStudentAgeBatch		707f900005VKCP

9. Scheduled Apex

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

Apex Class

ScheduleUpdateStudentAgeBatch

Apex Class Detail

Name: ScheduleUpdateStudentAgeBatch
Namespace Prefix: Yerasi_Lobitha
Created By: Yerasi Lobitha, 9/24/2025, 2:56 AM
Status: Active
Code Coverage: 0% (0/2)
Last Modified By: Yerasi Lobitha, 9/24/2025, 2:56 AM

Class Body

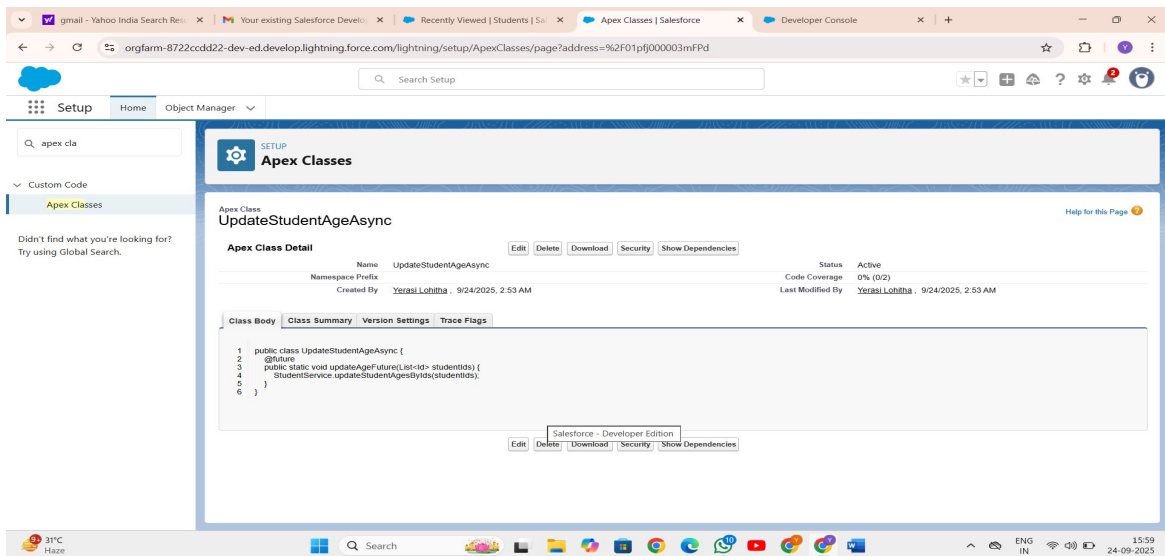
```

1 public class ScheduleUpdateStudentAgeBatch implements Schedulable {
2     public void execute(SchedulableContext sc) {
3         Database.executeBatch(new UpdateStudentAgeBatch(), 200);
4     }
5 }

```

10. Future Methods

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

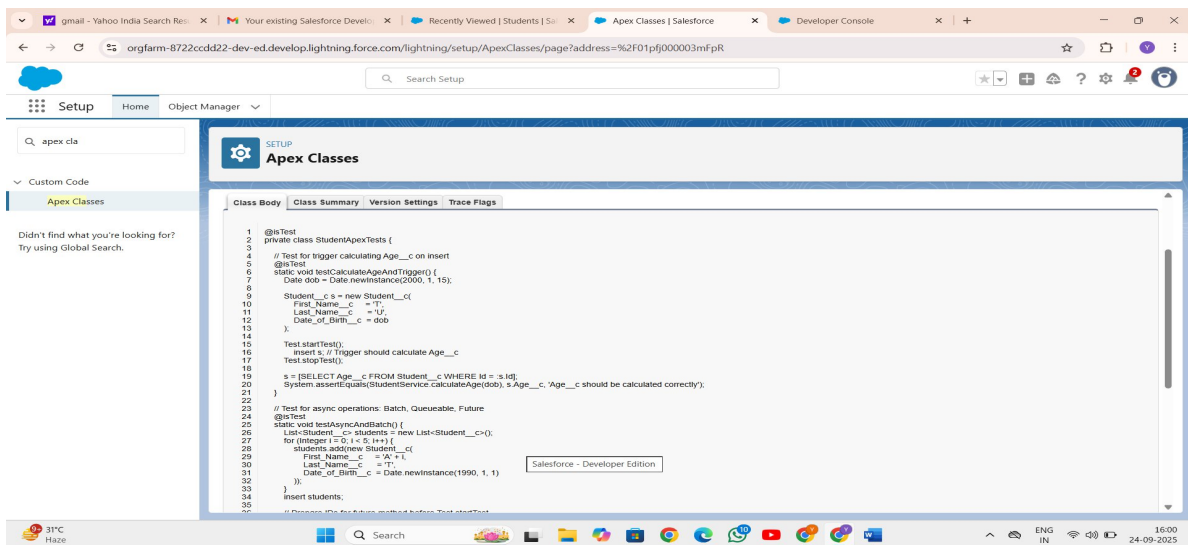


11. Exception Handling

- 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.