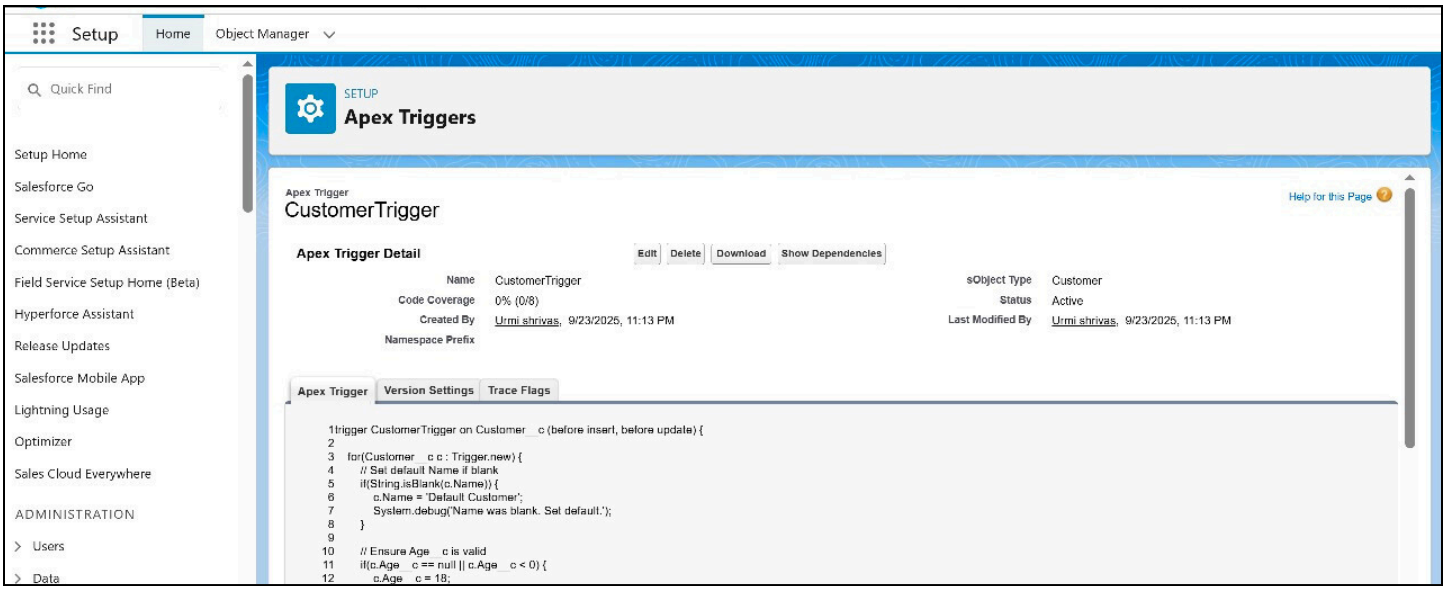
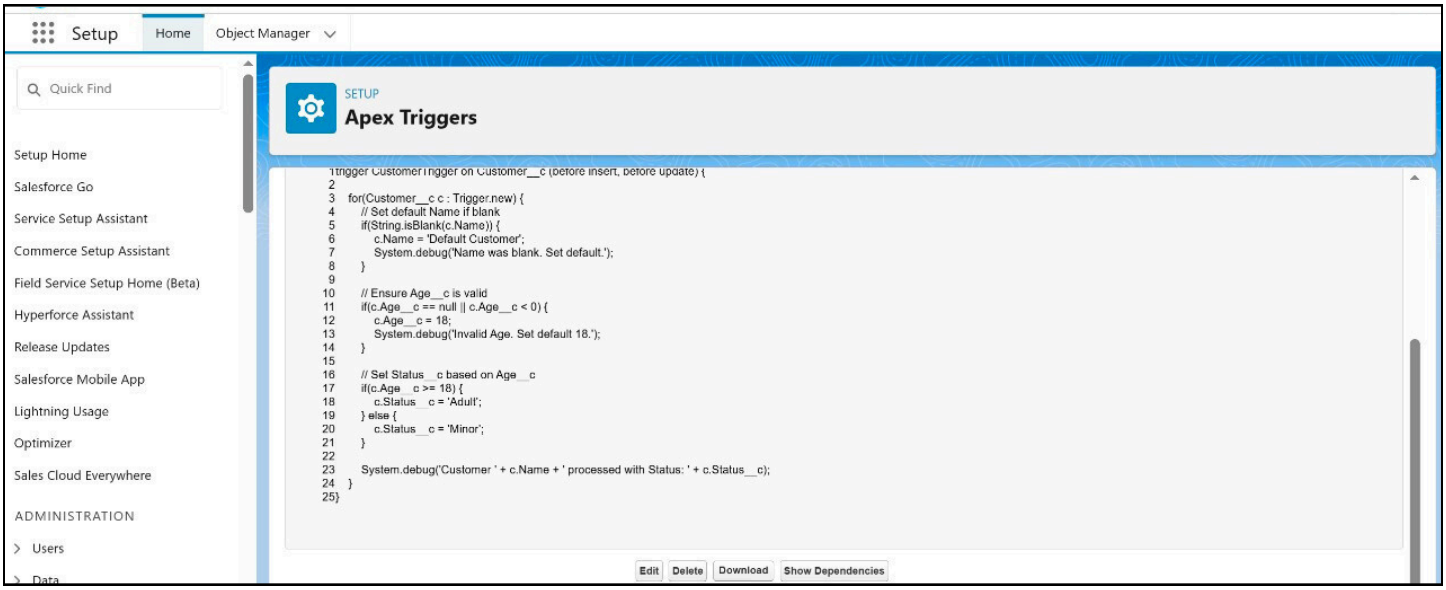
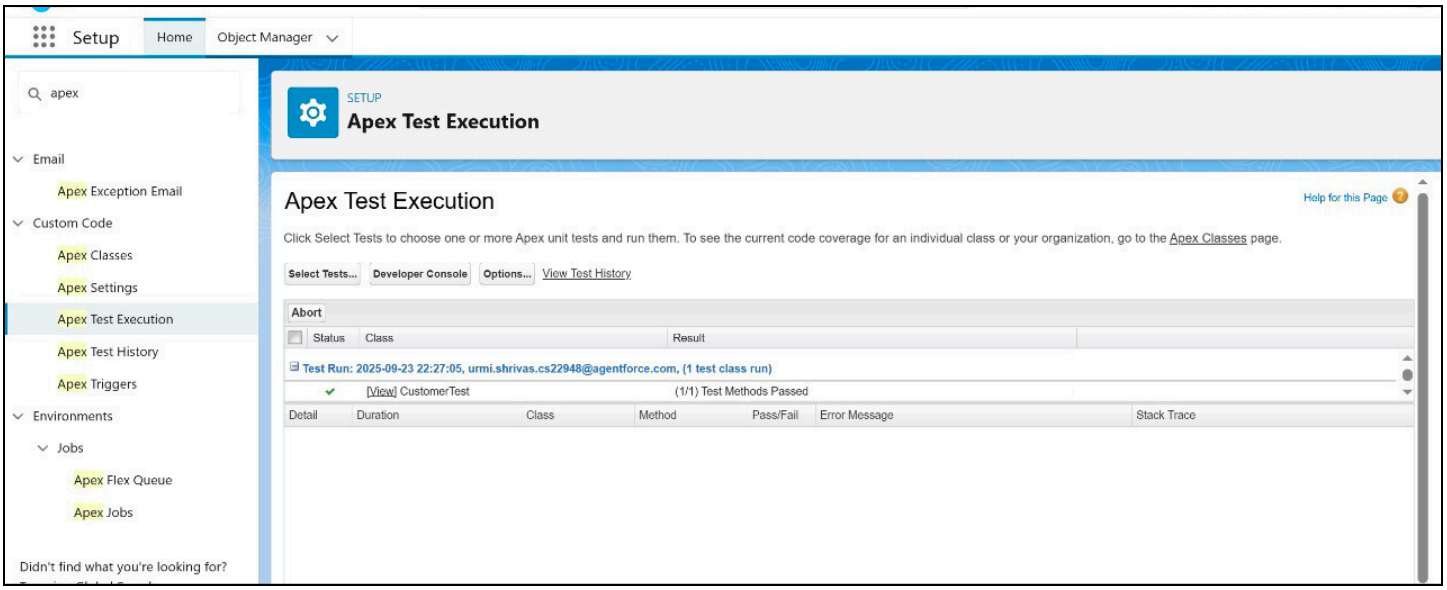
Phase 5: Apex Programming(Developers)

1. ·Classes & Objects

- Created **Apex Class Customer** with variables: name, age, email, phoneNumber, city, isActive.
- Added **constructor** to initialize variables.
- Implemented **methods**:
- displayInfo() → prints customer details.
- isAdult() → checks if age ≥ 18.
- Created **multiple Customer objects** and called their methods.
- Added **static test method (runTest())** to test class functionality.
- Used **System.debug()** to verify outputs.
- Executed the class via **Apex Test Execution** in the org.





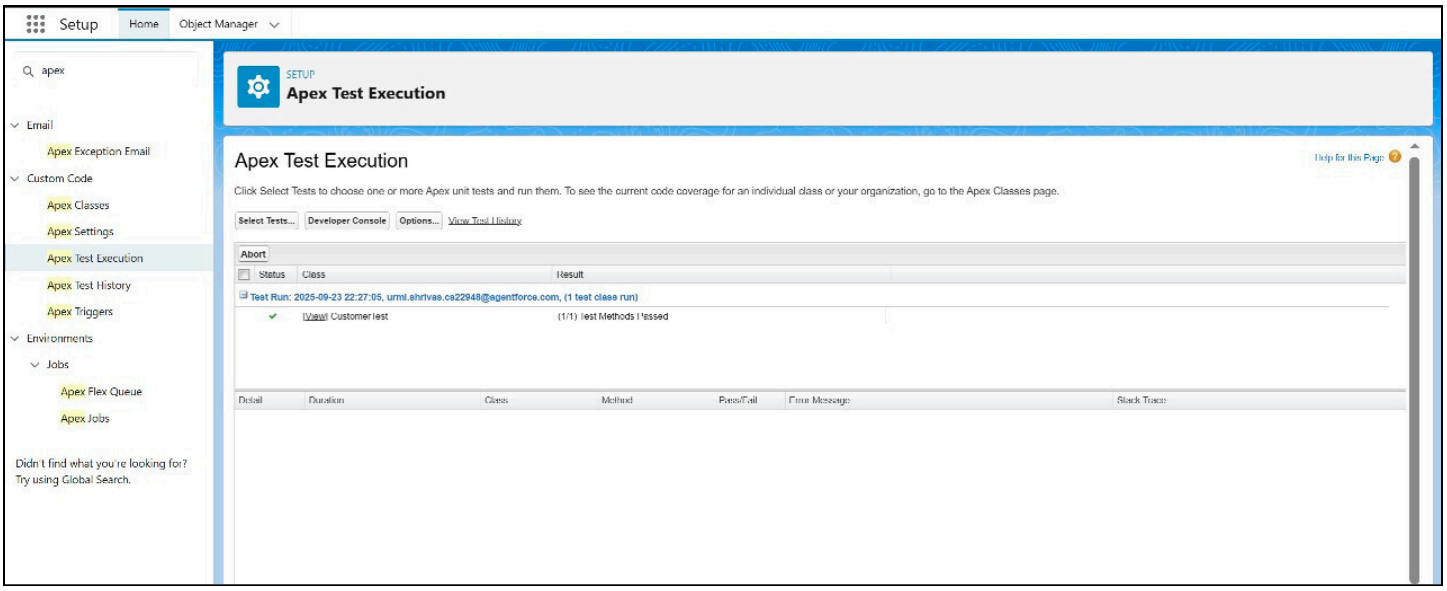


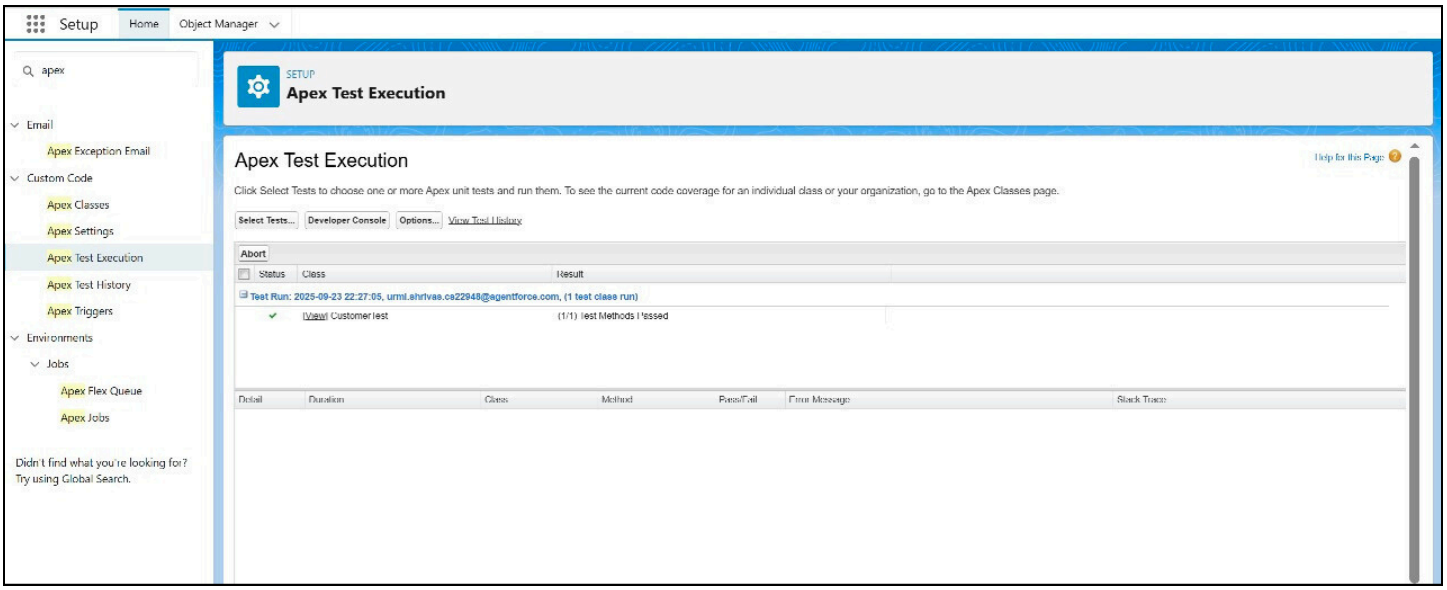
---

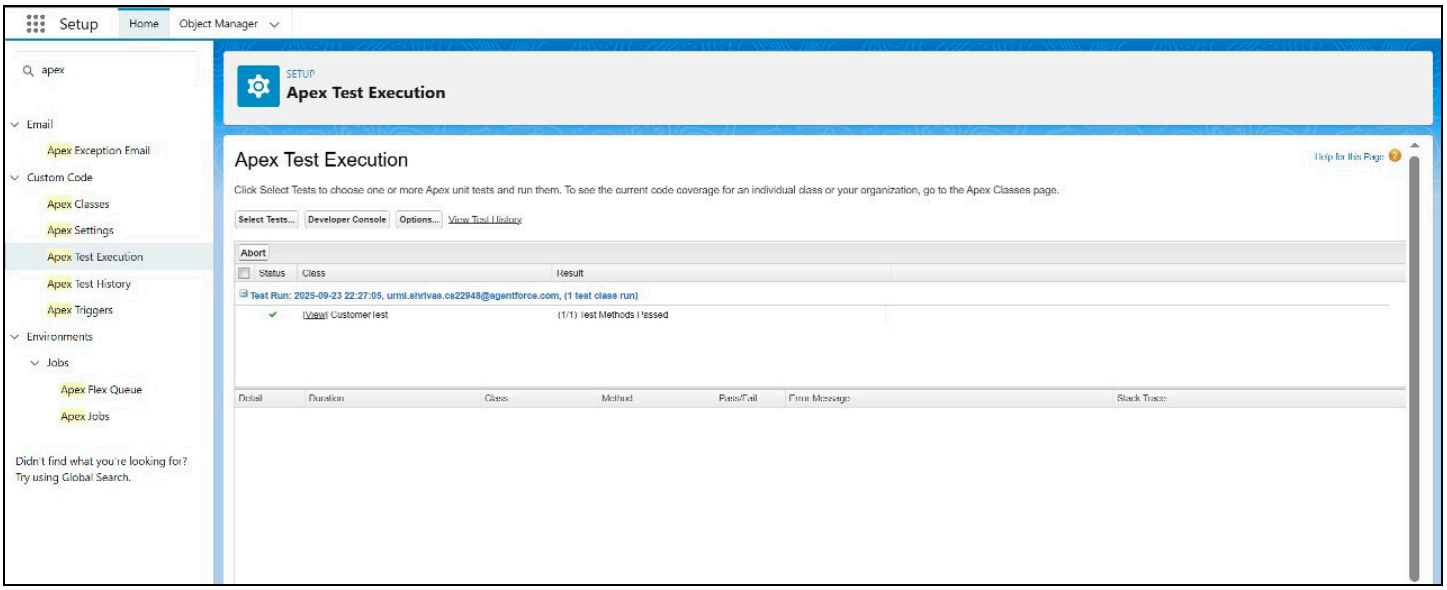
2. Apex Triggers(before/after insert/update/delete)

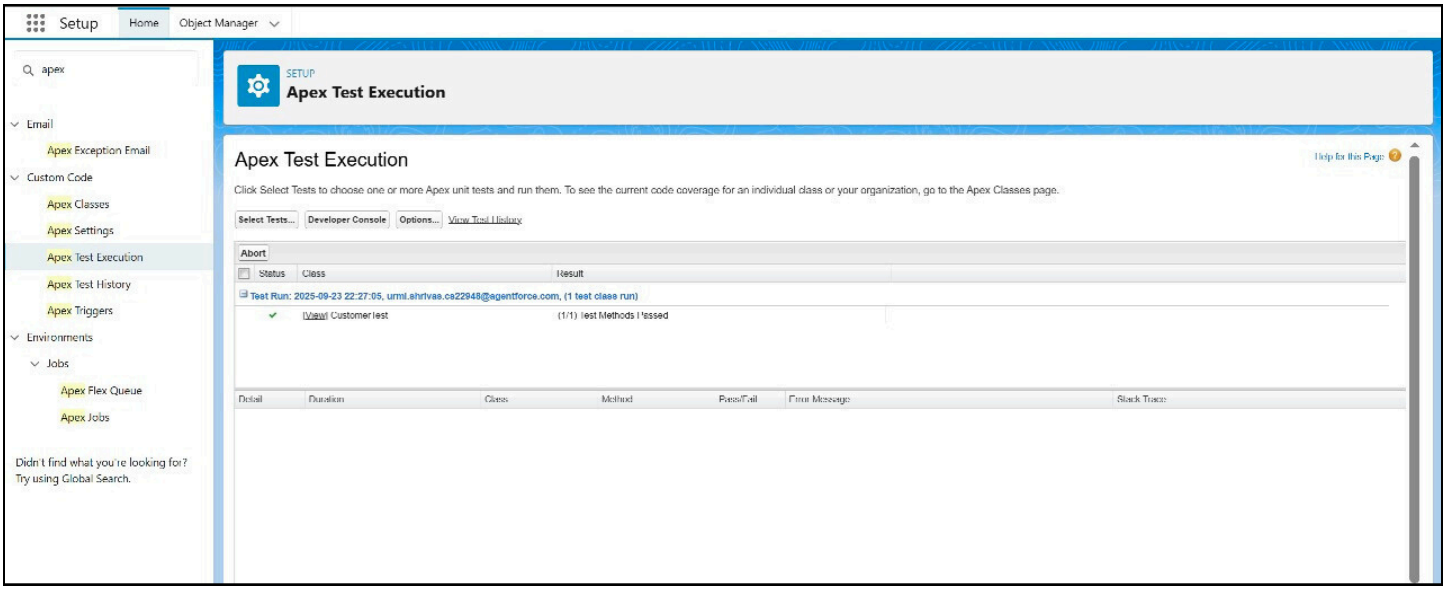
- Created **CustomerTrigger** on Customer__c (before insert/update).
- Validated **Name** (default if blank) and **Age__c** (default if null/negative).
- Set **Status__c** based on age: Adult or Minor.
- Created **CustomerTriggerTest** class with test records to verify logic.
- Ran test via **Apex Test Execution** and checked **debug logs**.

Outcome: Trigger works correctly; validations and status assignment are verified.




Setup
Home
Object Manager
apex
Email
Apex Exception Email
Custom Code
Apex Classes
Apex Settings
Apex Test Execution
Apex Test History
Apex Triggers
Environments
Jobs
Apex Flex Queue
Apex Jobs
Didn't find what you're looking for?
Try using Global Search.
SETUP
Apex Test Execution
Apex Test Execution
Help for this Page
Click Select Tests to choose one or more Apex unit tests and run them. To see the current code coverage for an individual class or your organization, go to the Apex Classes page.
Select Tests...
Developer Console
Options...
View Test History
Abort
Status
Class
Result
Test Run: 2025-09-23 22:27:05, urmi.shrivas.ca22948@agentforce.com, (1 test class run)
[View] Customer Test
(1/1) Test Methods Passed
Detail
Duration
Class
Method
Pass/Fail
Error Message
Stack Trace

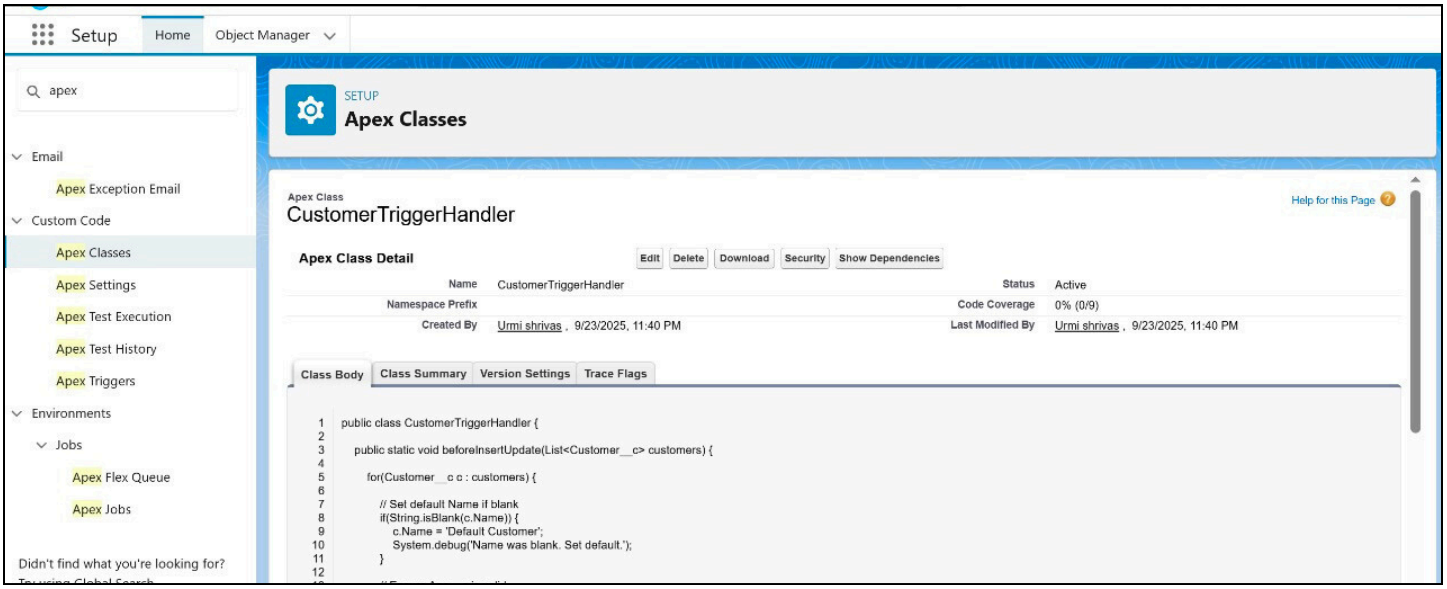

Setup
Home
Object Manager
apex
Email
Apex Exception Email
Custom Code
Apex Classes
Apex Settings
Apex Test Execution
Apex Test History
Apex Triggers
Environments
Jobs
Apex Flex Queue
Apex Jobs
Didn't find what you're looking for?
Try using Global Search.
SETUP
Apex Test Execution
Apex Test Execution
Help for this Page
Click Select Tests to choose one or more Apex unit tests and run them. To see the current code coverage for an individual class or your organization, go to the Apex Classes page.
Select Tests...
Developer Console
Options...
View Test History
Abort
Status
Class
Result
Test Run: 2025-09-23 22:27:05, urmi.shrivas.ca22948@agentforce.com, (1 test class run)
[View] Customer Test
(1/1) Test Methods Passed
Detail
Duration
Class
Method
Pass/Fail
Error Message
Stack Trace

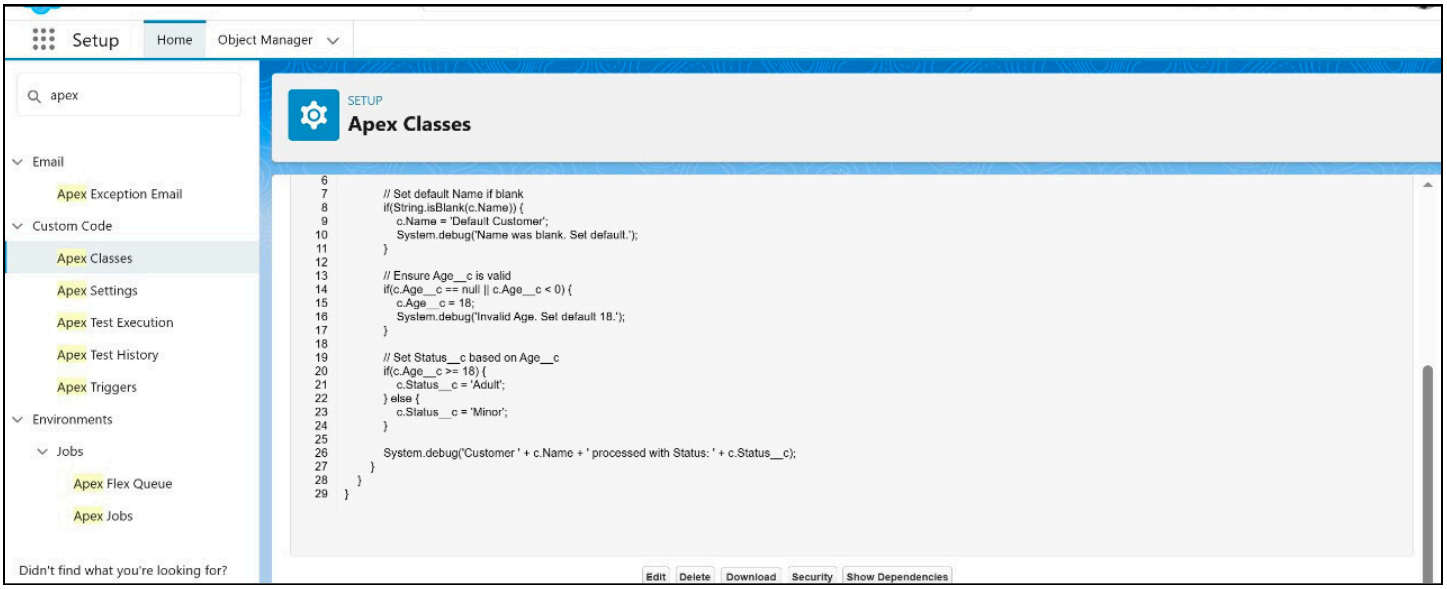

Setup
Home
Object Manager
apex
Email
Apex Exception Email
Custom Code
Apex Classes
Apex Settings
Apex Test Execution
Apex Test History
Apex Triggers
Environments
Jobs
Apex Flex Queue
Apex Jobs
Didn't find what you're looking for?
Try using Global Search.
SETUP
Apex Test Execution
Apex Test Execution
Help for this Page
Click Select Tests to choose one or more Apex unit tests and run them. To see the current code coverage for an individual class or your organization, go to the Apex Classes page.
Select Tests...
Developer Console
Options...
View Test History
Abort
Status
Class
Result
Test Run: 2025-09-23 22:27:05, urmi.shrivas.ca22948@agentforce.com, (1 test class run)
[View] Customer Test
(1/1) Test Methods Passed
Detail
Duration
Class
Method
Pass/Fail
Error Message
Stack Trace

3.Trigger Design Pattern

- Moved trigger logic to a **handler class CustomerTriggerHandler**.
  - beforeInsertUpdate(List<Customer__c> customers) handles validations and status assignment.
- Updated trigger (CustomerTriggerHandlerTrigger) to call the **handler class**, making the trigger **clean and maintainable**.
- Created a **test class CustomerTriggerHandlerTest** to verify the handler logic.
- Ran test via **Apex Test Execution** → debug logs confirmed correct execution.

Outcome: Trigger logic is now **organized, reusable, and follows Salesforce best practices**.



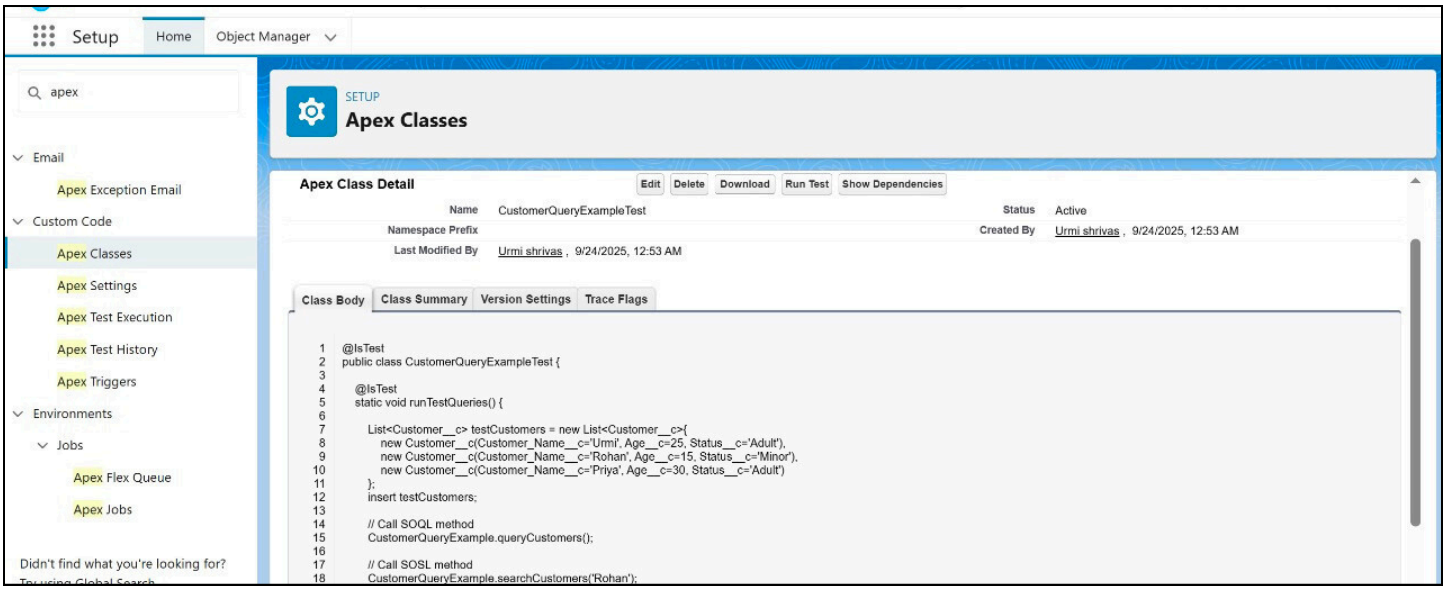


4· Apex Programming – SOQL & SOSL

- Created **Apex Class CustomerQueryExample** to implement:
  - **SOQL query** → retrieve adult customers (Age__c >= 18).
  - **SOSL search** → search customer records by name (Customer_Name__c).
- Created **separate Test Class CustomerQueryExampleTest** to:
  - Insert sample Customer__c records.
  - Call the main class methods for testing.
- Verified functionality using **Apex Test Execution**.
- Ensured **Customer__c object is searchable** for SOSL.

Outcome:

- Practiced **retrieving and searching records** in Apex.
- Learned proper **test class creation** and **execution of queries in Salesforce**.



5· Collections (List, Set, Map)

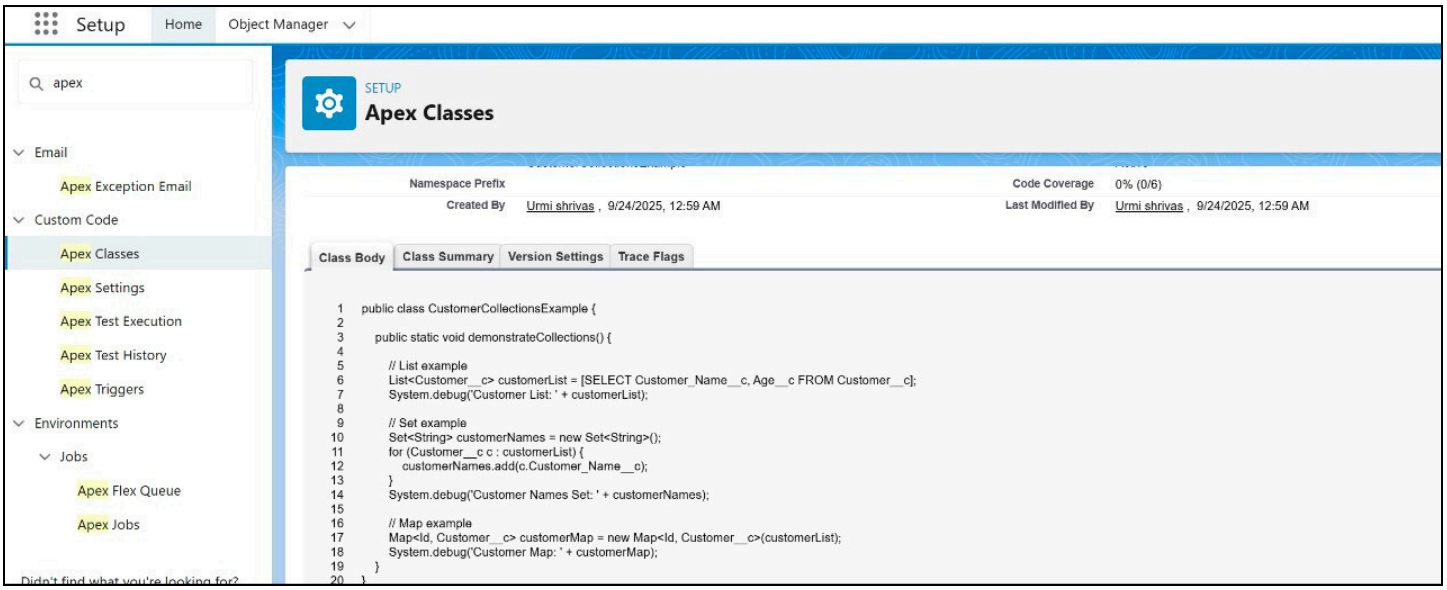
Created Apex Class CustomerCollectionsExample to demonstrate:

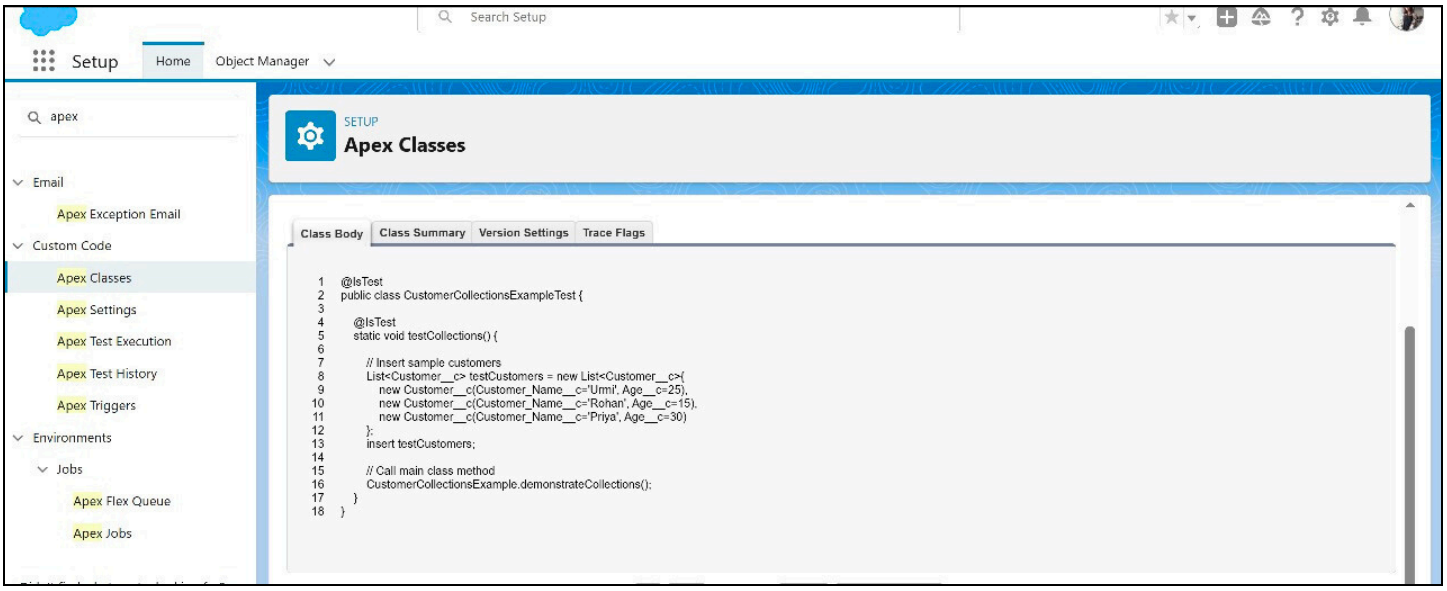
- **List → stored all Customer__c records.**
- **Set → stored unique customer names from the list.**
- **Map → stored Customer__c records with Id as key for quick access.**

- **Created separate Test Class CustomerCollectionsExampleTest to:**
  - **Insert sample Customer__c records.**
  - **Call the main class method for testing collections.**
- **Verified functionality using Apex Test Execution and Debug Logs.**

Outcome:

- **Practiced organizing multiple records using List, Set, and Map.**
- **Learned proper handling of duplicates, ordering, and key-value pairs in Apex collections.**



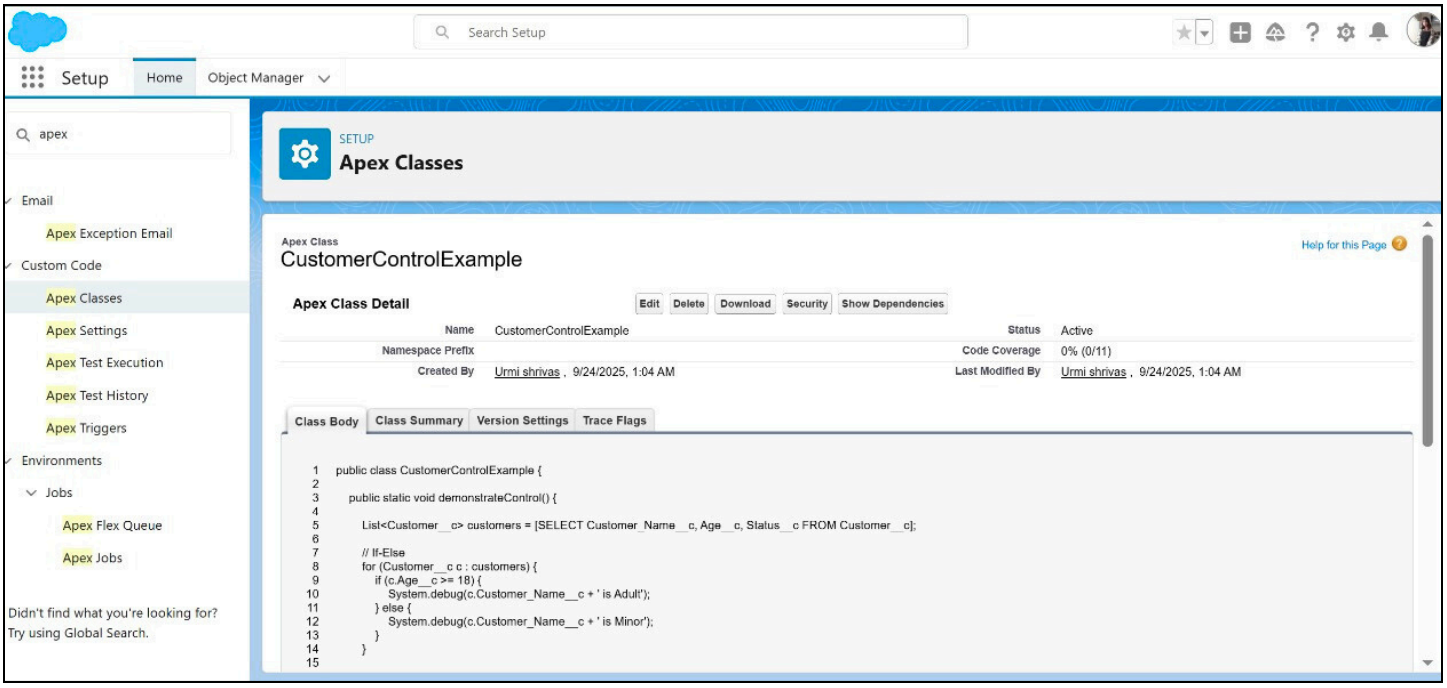


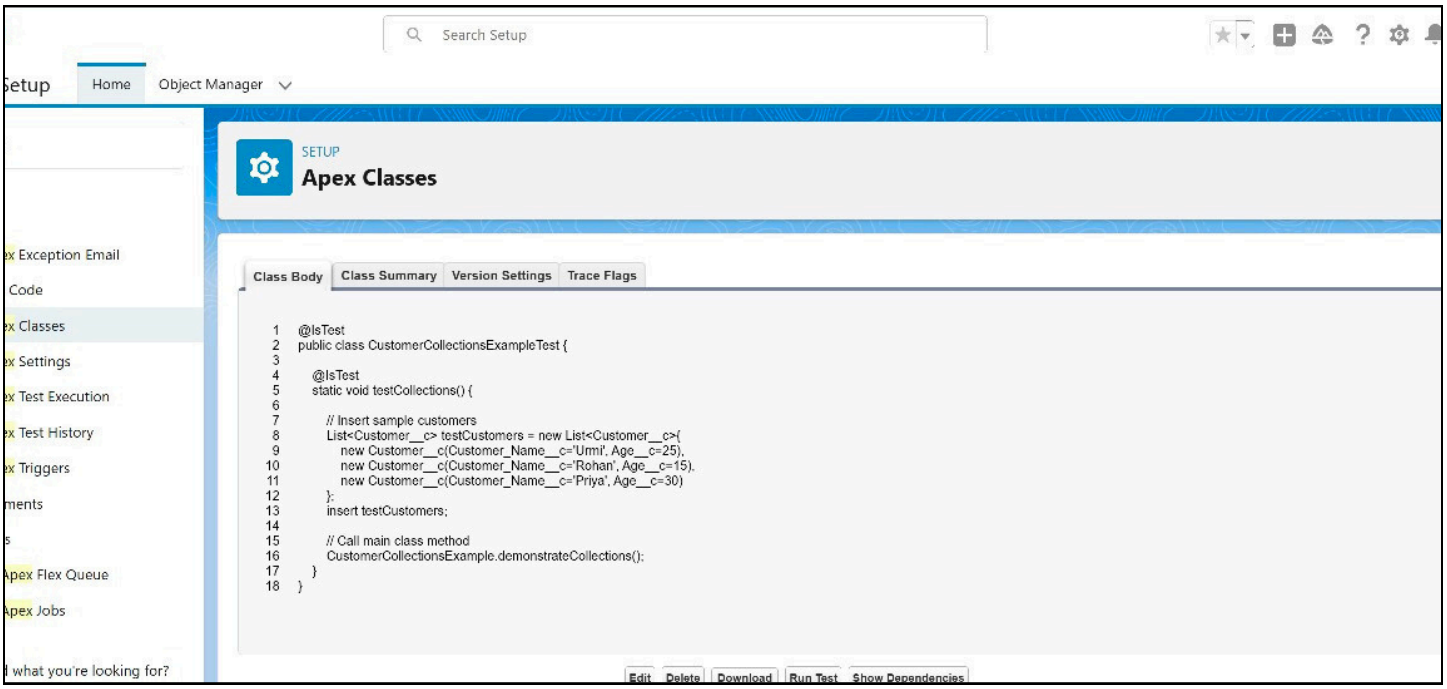
6· Control Statements

- **Created Apex Class CustomerControlExample to demonstrate:**
  - If-Else → check if customer is Adult or Minor.
  - For Loop → iterate over all Customer__c records.
  - While Loop → process customers until all are handled.
  - Switch Statement → handle different Status__c values.
- **Created Test Class CustomerControlExampleTest to:**
  - Insert sample Customer__c records.
  - Call the main class method to verify control statements.
- Verified results using **Apex Test Execution and Debug Logs.**

Outcome:

- Practiced using conditional logic and loops in Apex.
- Learned how to control code flow based on record data.



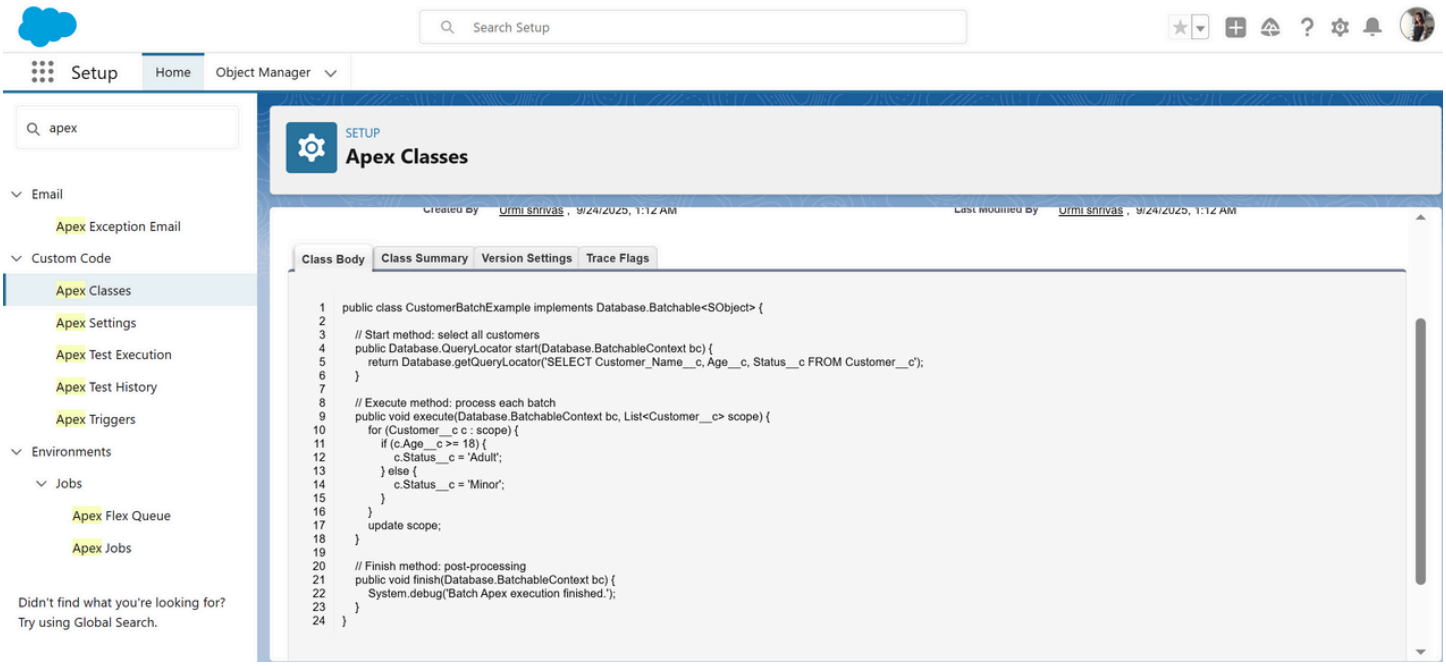


7. Batch Apex

- Created **Batch Apex Class CustomerBatchExample** to:
  - Process all Customer__c records in batches.
  - Update Status__c based on Age__c (Adult or Minor).
- Implemented **start()**, **execute()**, and **finish()** methods.
- Created **Test Class CustomerBatchExampleTest** to:
  - Insert sample Customer__c records.
  - Execute batch with Database.executeBatch.
- Verified batch execution using **Apex Test Execution** and **Debug Logs**.

Outcome:

- Practiced **processing large record sets asynchronously**.
- Learned how to implement **Database.Batchable** interface in Apex.



---

8. Queueable Apex

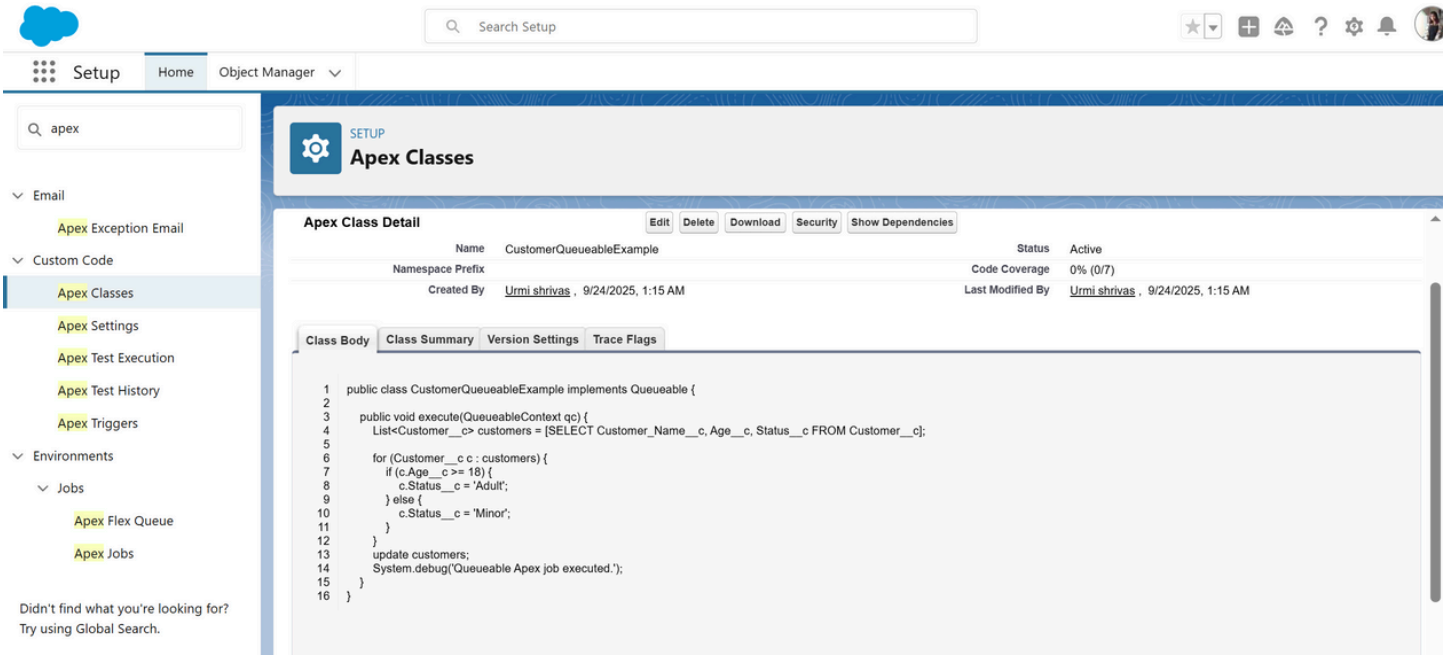
- Created **Queueable Apex Class CustomerQueueableExample** to:
  - Process Customer__c records asynchronously.
  - Update Status__c based on Age__c (Adult or Minor).
- Created **Test Class CustomerQueueableExampleTest** to:

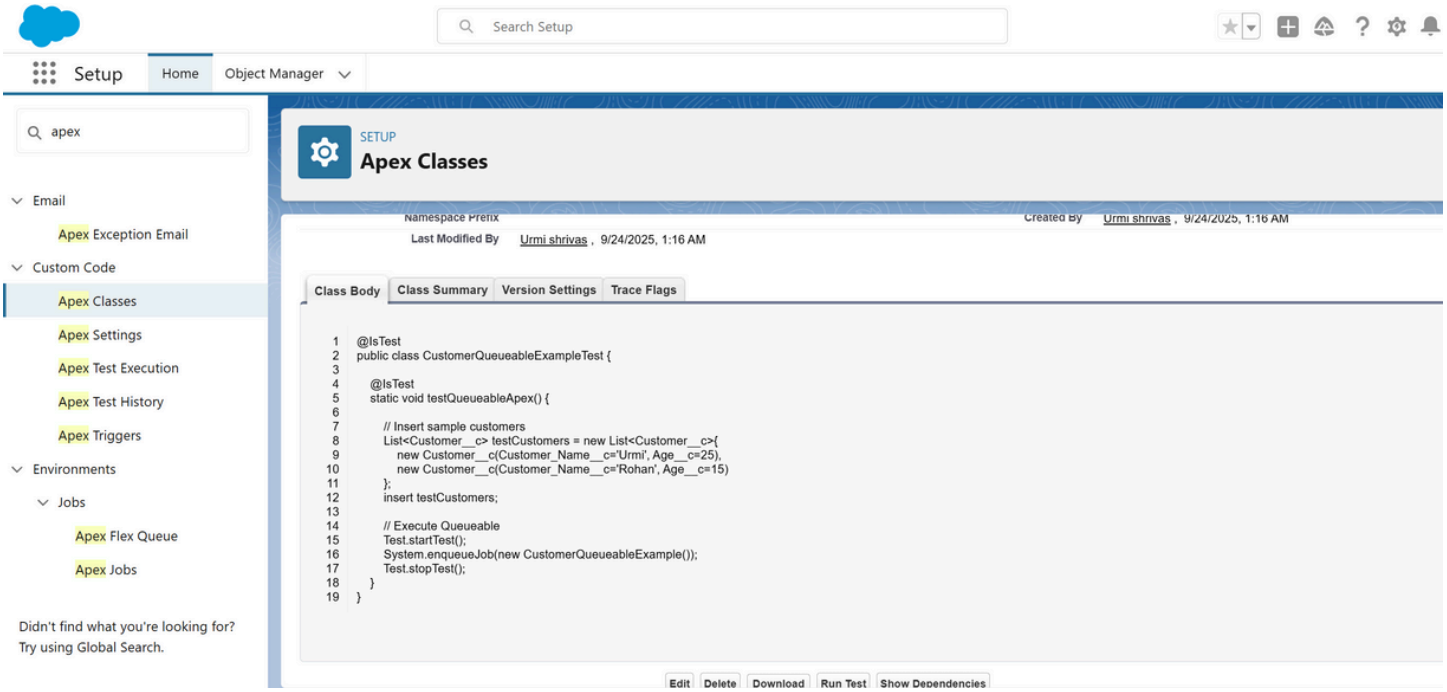
- Insert sample Customer__c records.
- Execute the queueable job using System.enqueueJob.

- Verified results using **Apex Test Execution** and **Debug Logs**.

Outcome:

- Practiced **asynchronous processing with Queueable Apex**.
- Learned how to **execute jobs and verify outcomes via test classes**



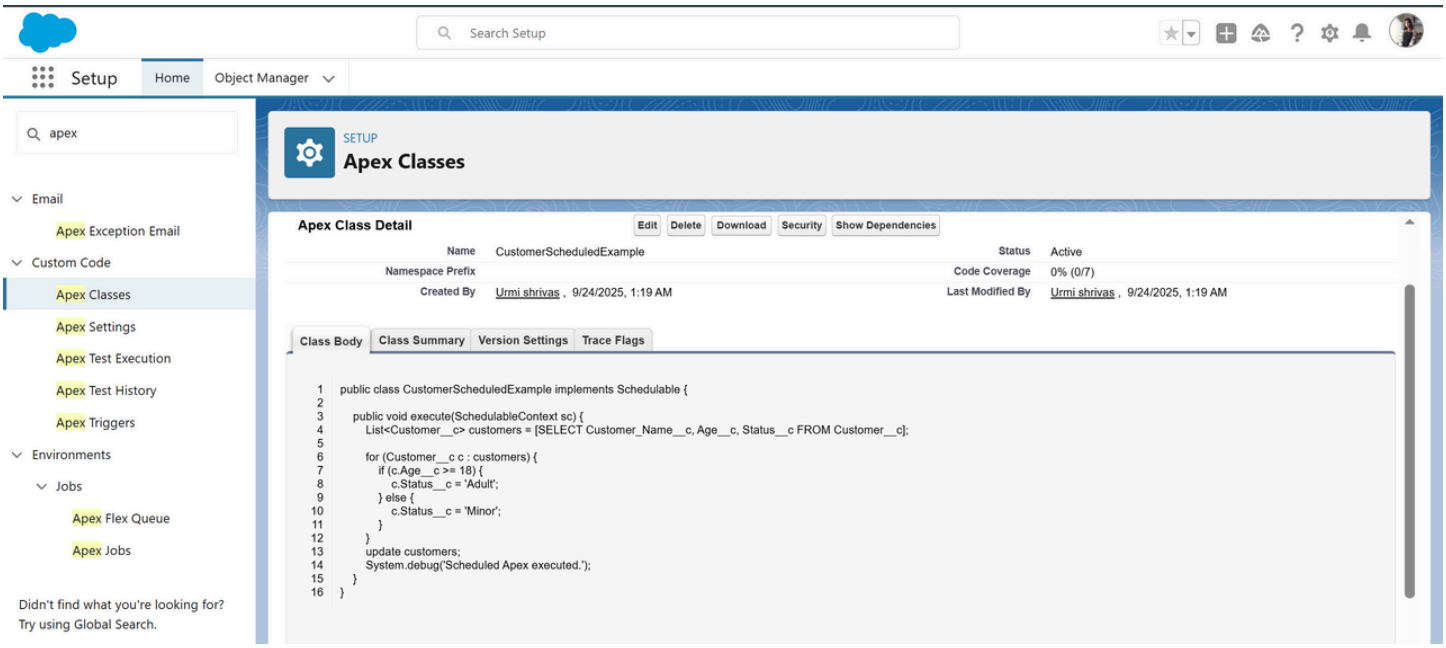


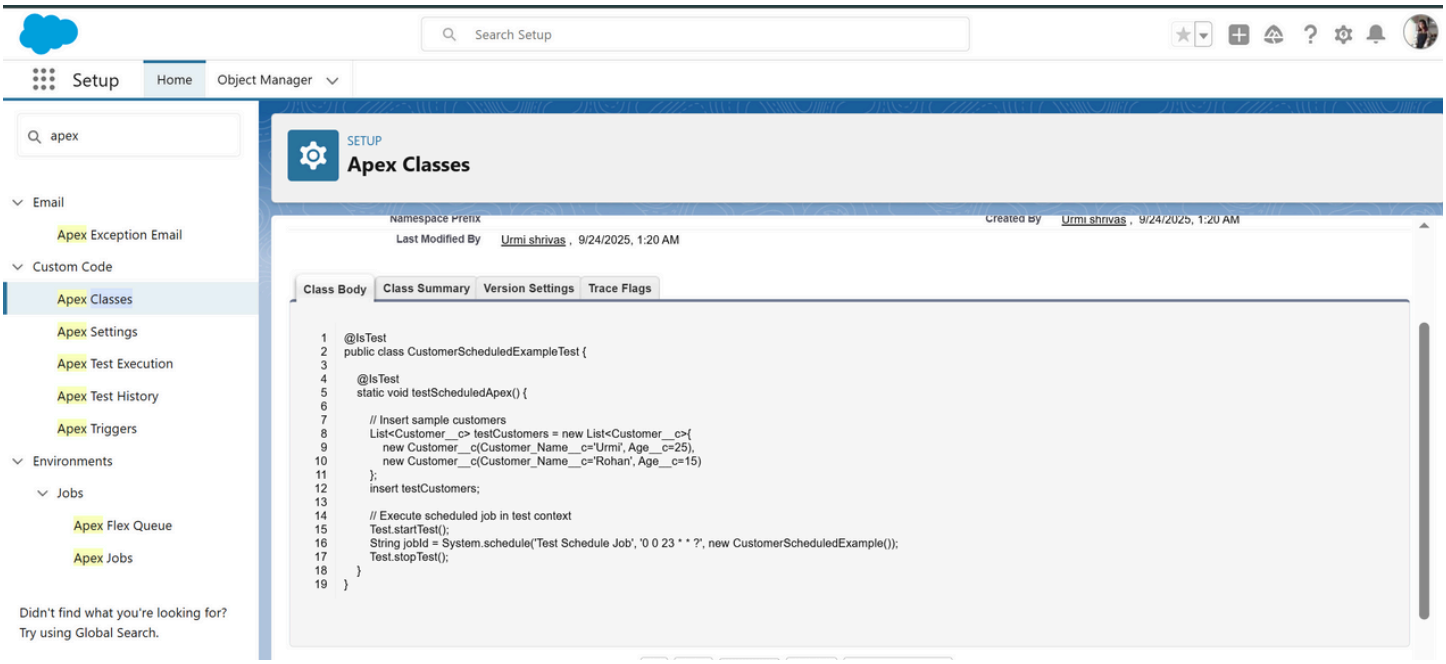
9. Scheduled Apex

- Created **Scheduled Apex Class CustomerScheduledExample** to:
  - Update Status__c of Customer__c records based on Age__c.
  - Run automatically at a scheduled time using CRON expression.
- Created **Test Class CustomerScheduledExampleTest** to:
  - Insert sample Customer__c records.
  - Schedule and execute the job in a test context using System.schedule.
- Verified results using **Apex Test Execution** and **Debug Logs**.

Outcome:

- Practiced **automating tasks with Scheduled Apex**.
- Learned how to **schedule jobs and test them properly** in Salesforce.





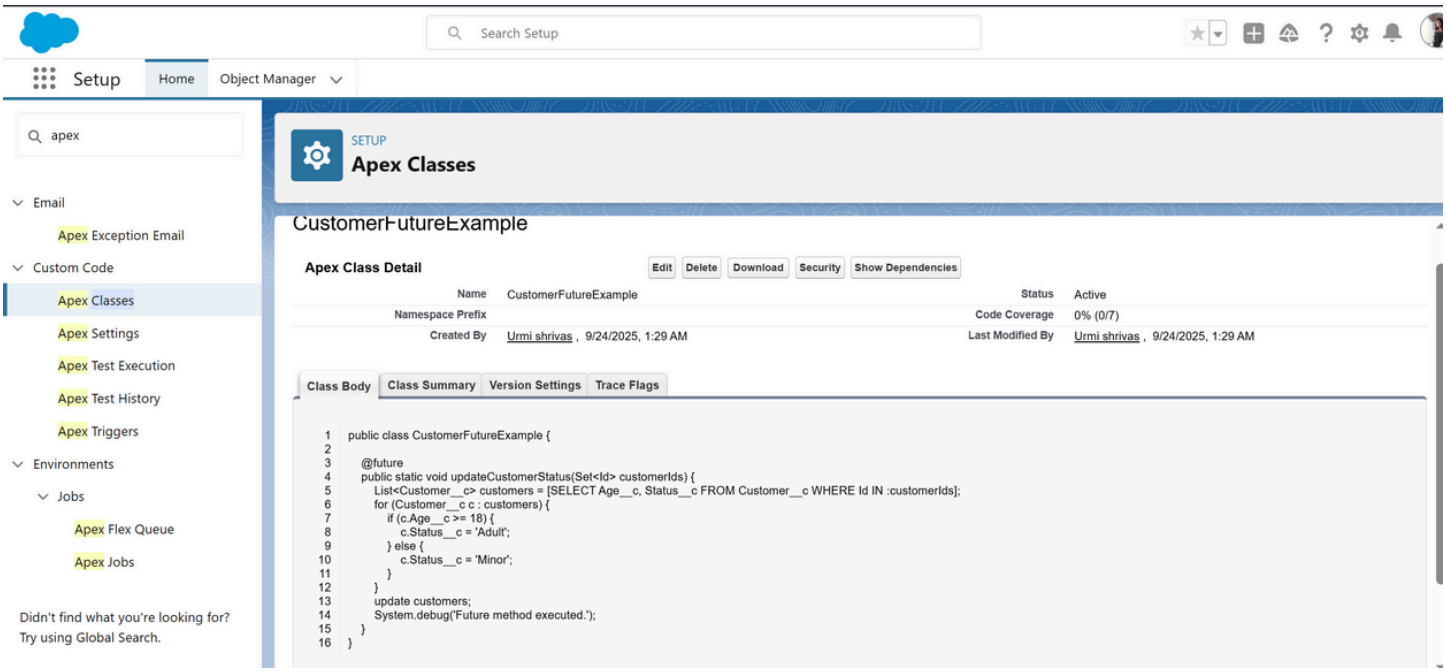
---

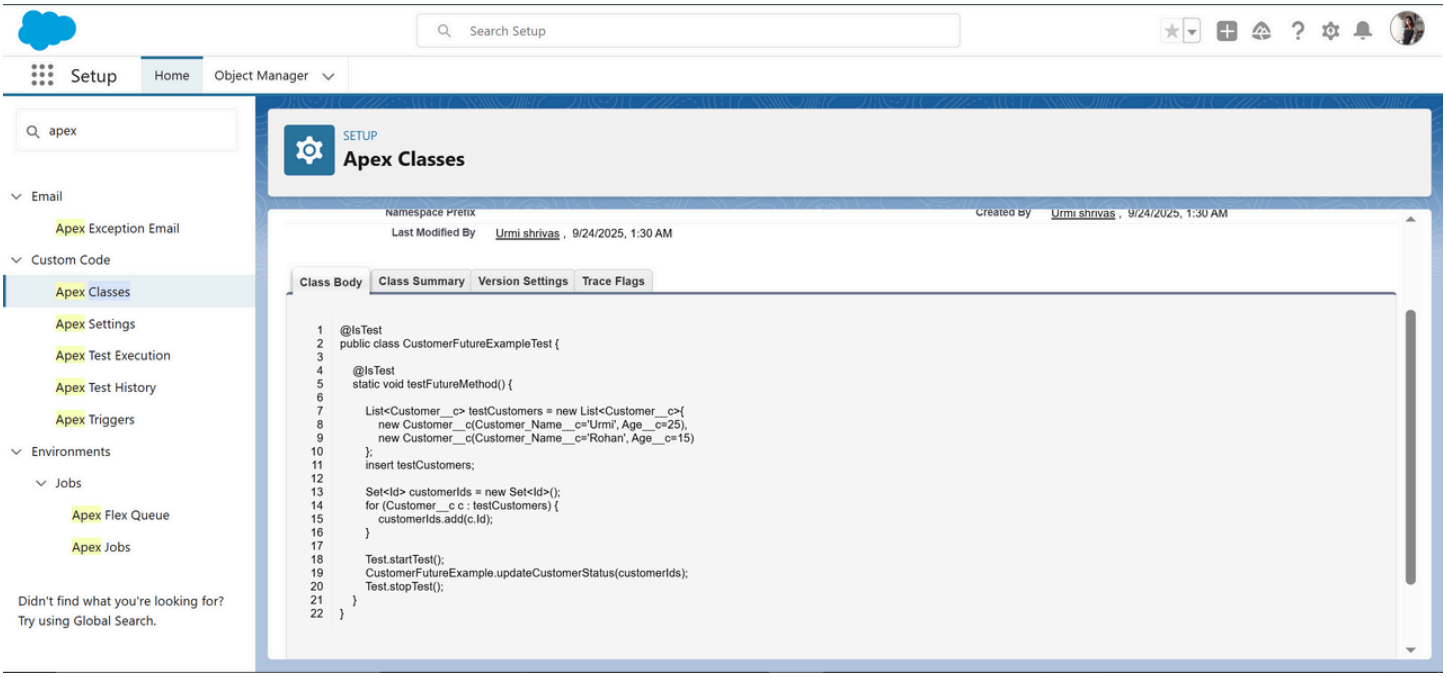
10. Future Methods

- Created **Future Method Class CustomerFutureExample** to:
  - Update Status__c of Customer__c records asynchronously.
  - Determine Adult or Minor based on Age__c.
- Created **Test Class CustomerFutureExampleTest** to:
  - Insert sample Customer__c records.
  - Call the future method in a test context.
- Verified results using **Apex Test Execution** and **Debug Logs**.

Outcome:

- Practiced **asynchronous processing with Future Methods**.
- Learned how to **run operations in the background without blocking users**.





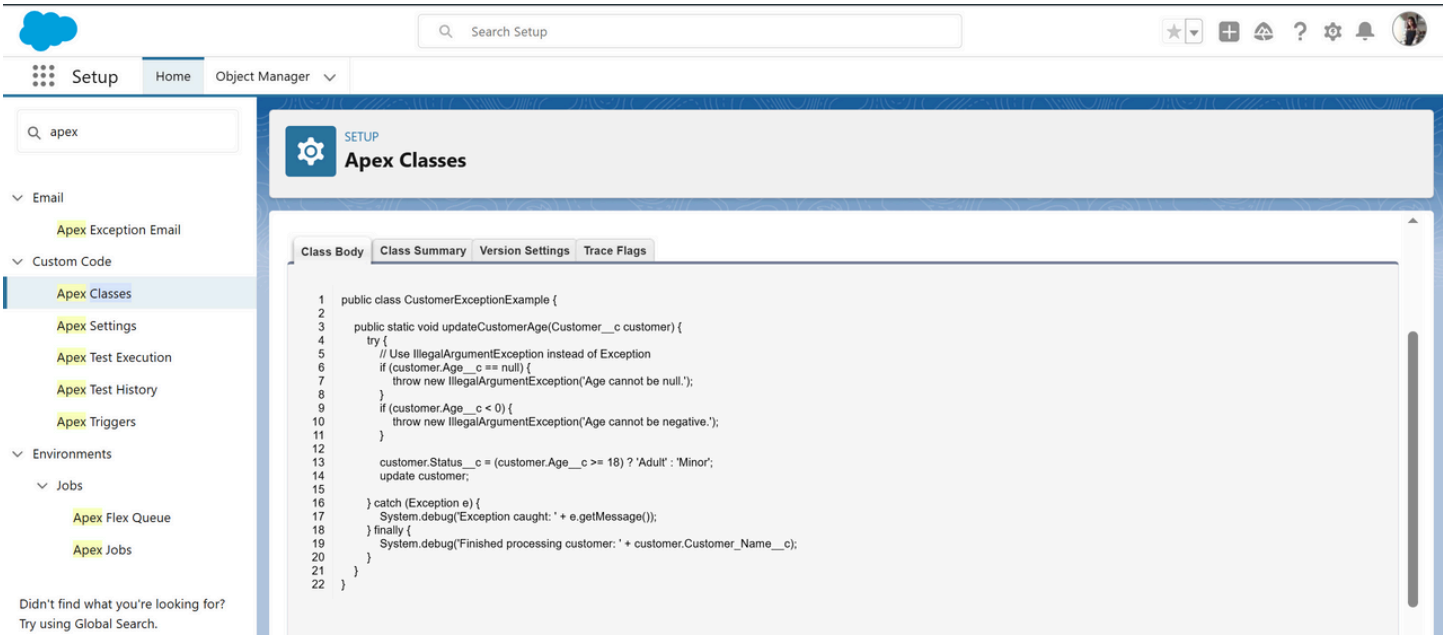
Exception Handling

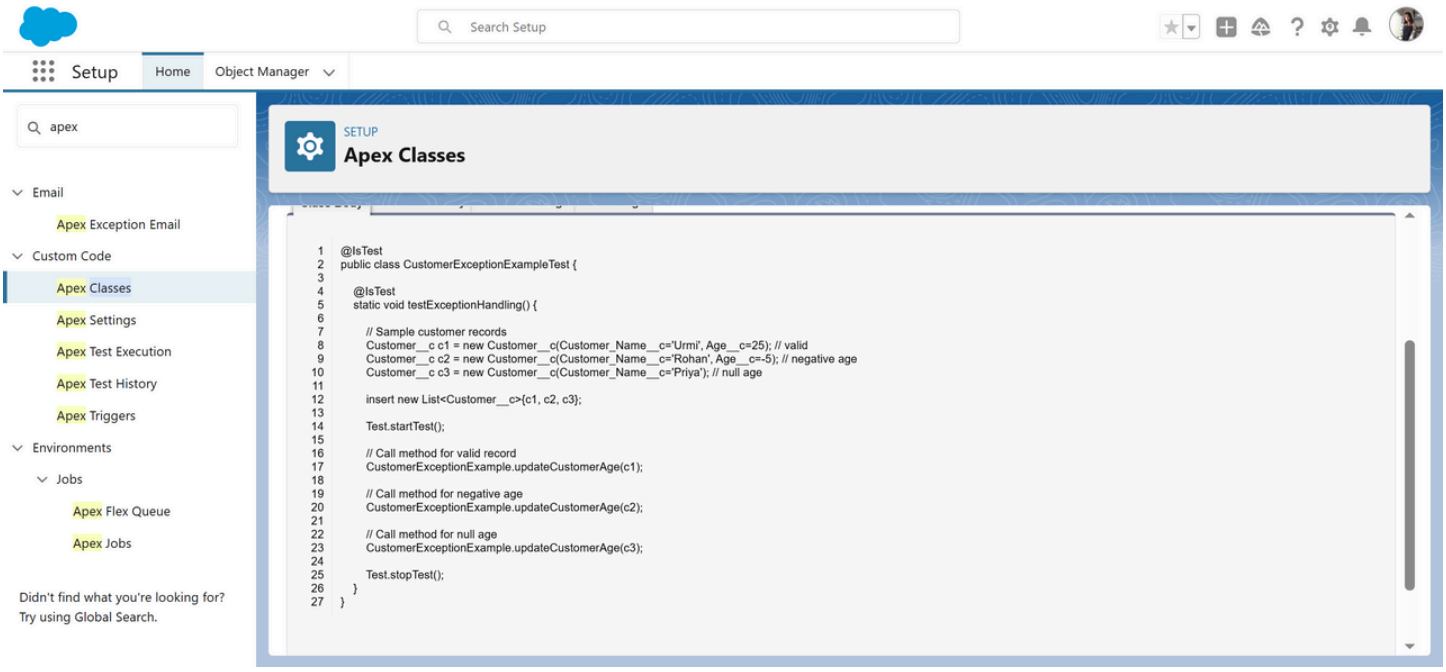
- Created **Apex Class CustomerExceptionExample** to:
  - Handle errors when Age__c is null or negative.
  - Update Status__c based on valid Age__c.
- Created **Test Class CustomerExceptionExampleTest** to:
  - Insert sample Customer__c records with valid and invalid ages.
  - Call the method to test exception handling.

- Verified results using **Apex Test Execution** and **Debug Logs**.

Outcome:

- Practiced **handling exceptions in Apex**.
- Learned how to **prevent runtime errors from breaking the code**.



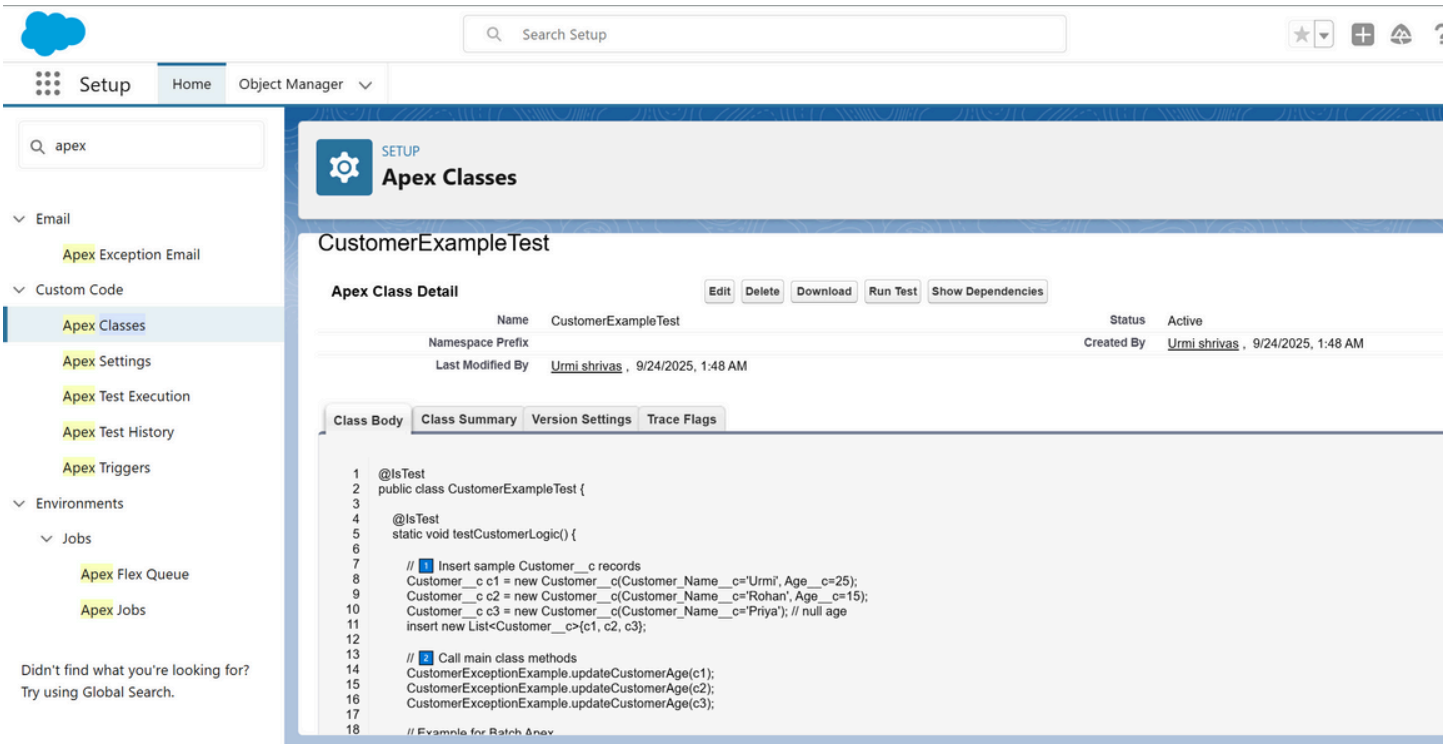


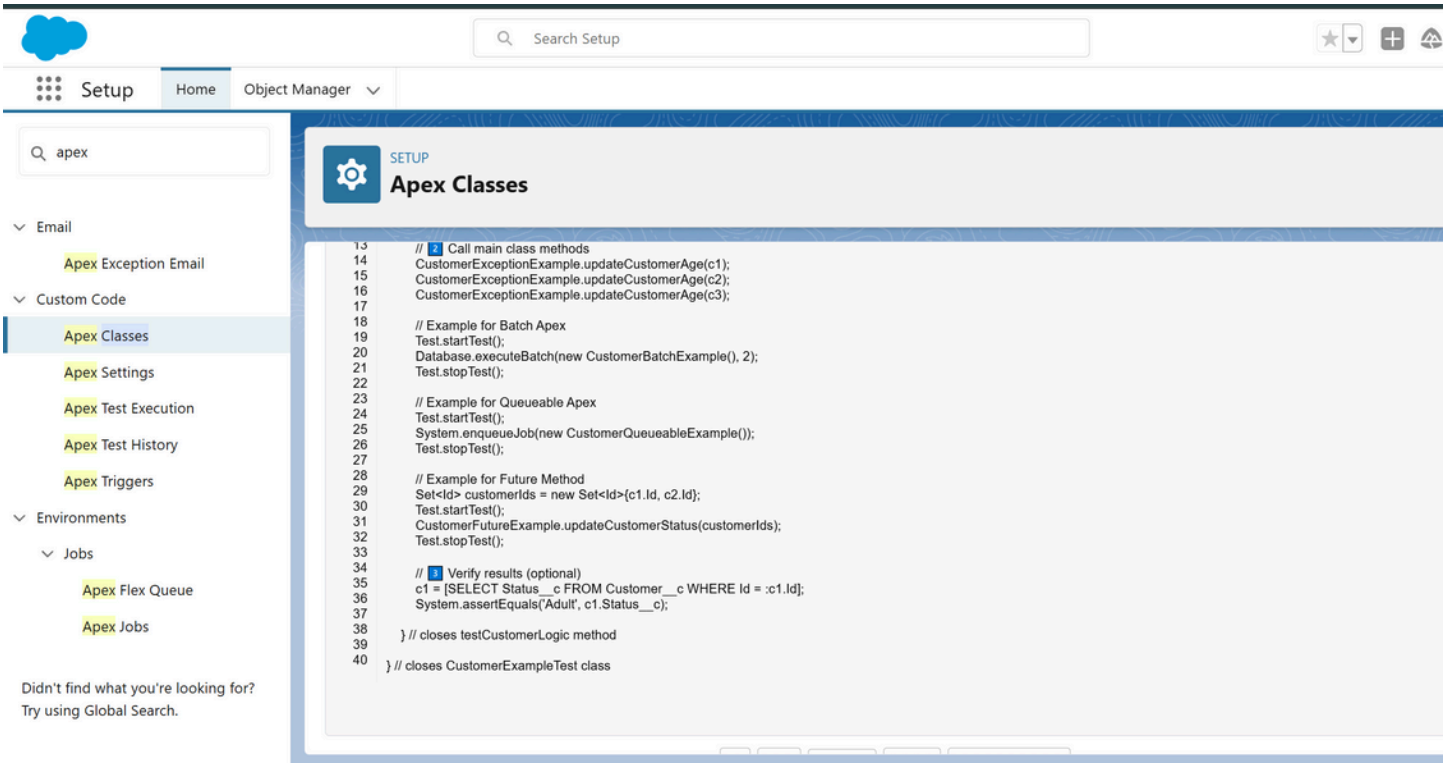
Test Classes

- Created **separate test classes** for each Apex class and trigger.
- Inserted **sample Customer__c records** with valid, invalid, and edge-case data.
- Used **Test.startTest() / Test.stopTest()** for asynchronous processing (Batch, Queueable, Scheduled, Future).
- Verified results using **System.assert**, **Apex Test Execution**, and **Debug Logs**.
- Ensured each test class covered its corresponding logic:
  - **Triggers** → CustomerTriggerTest
  - **SOQL & SOSL** → CustomerQueryExampleTest
  - **Collections** → CustomerCollectionsExampleTest
  - **Control Statements** → CustomerControlExampleTest
  - **Batch Apex** → CustomerBatchExampleTest
  - **Queueable Apex** → CustomerQueueableExampleTest
  - **Scheduled Apex** → CustomerScheduledExampleTest
  - **Future Methods** → CustomerFutureExampleTest
  - **Exception Handling** → CustomerExceptionExampleTest

Outcome:

- Achieved required **75%+ code coverage** for deployment.
- Learned to write effective **unit tests for both synchronous and asynchronous logic**.
- Verified **end-to-end execution of Apex code** safely in test context.





Asynchronous Processing

- **Batch Apex** → Large records ko chunks me process kiya (executeBatch).
- **Queueable Apex** → Background jobs run kiye (enqueueJob).
- **Scheduled Apex** → Jobs ko CRON expression ke sath schedule kiya.
- **Future Methods** → Async updates run kiye without blocking users.

Outcome: Asynchronous methods use karke background me process karna aur performance improve karna sikha.