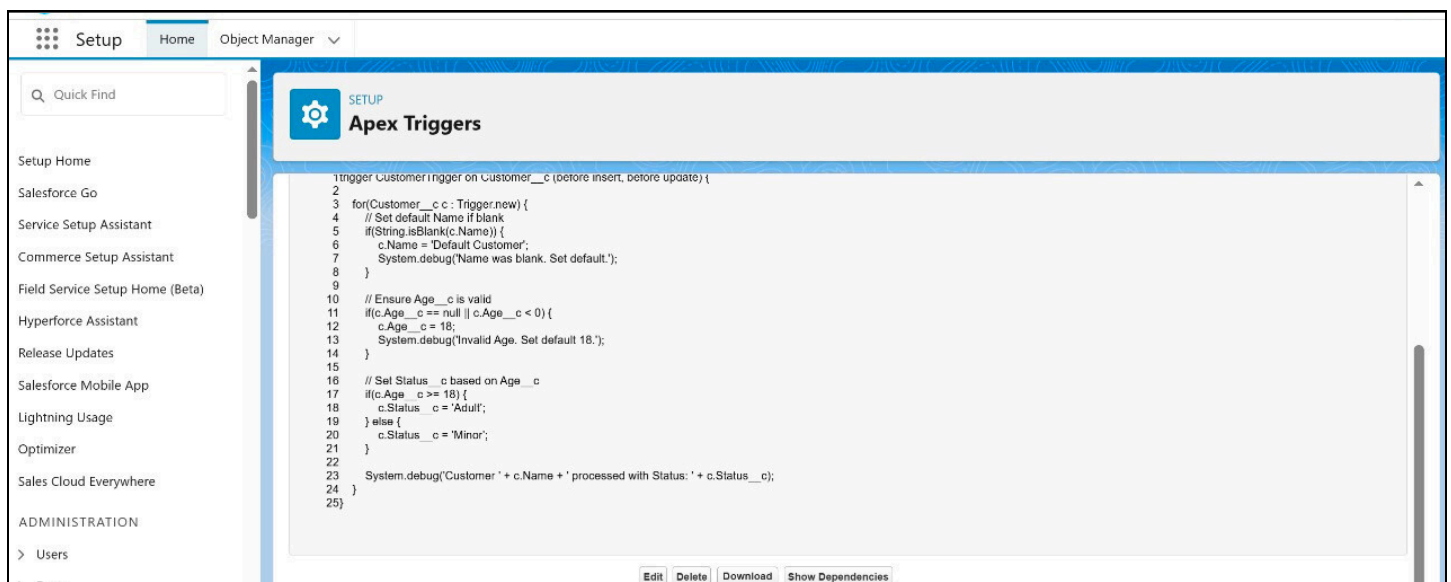
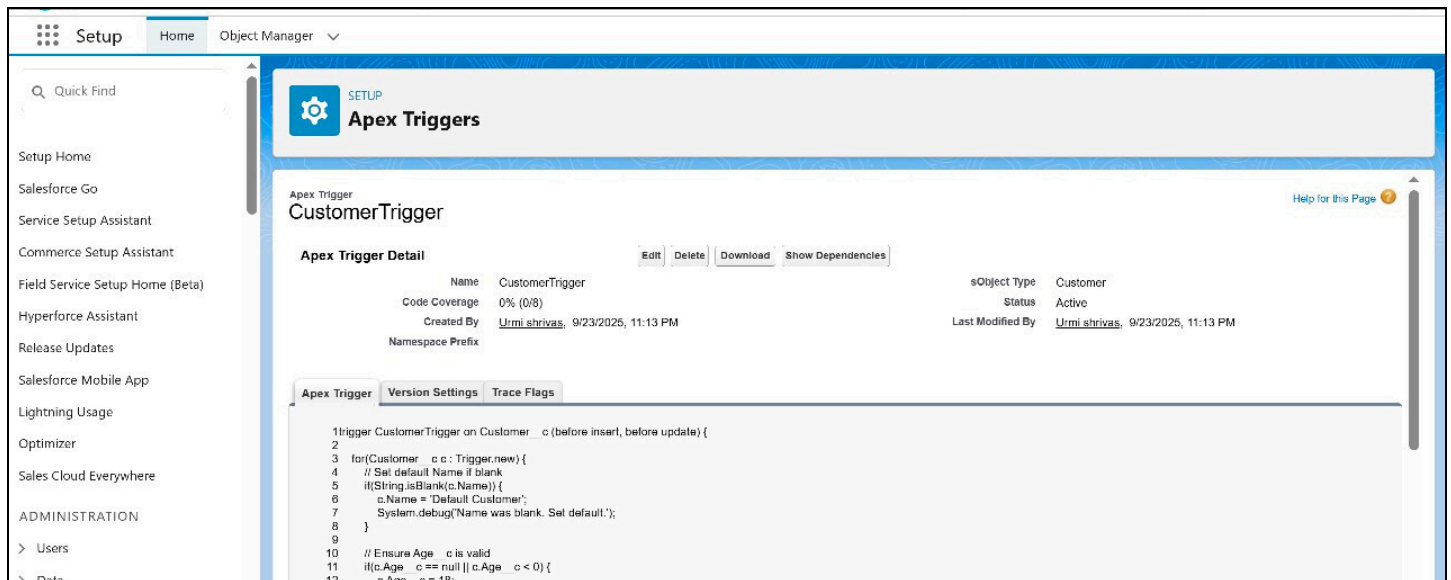
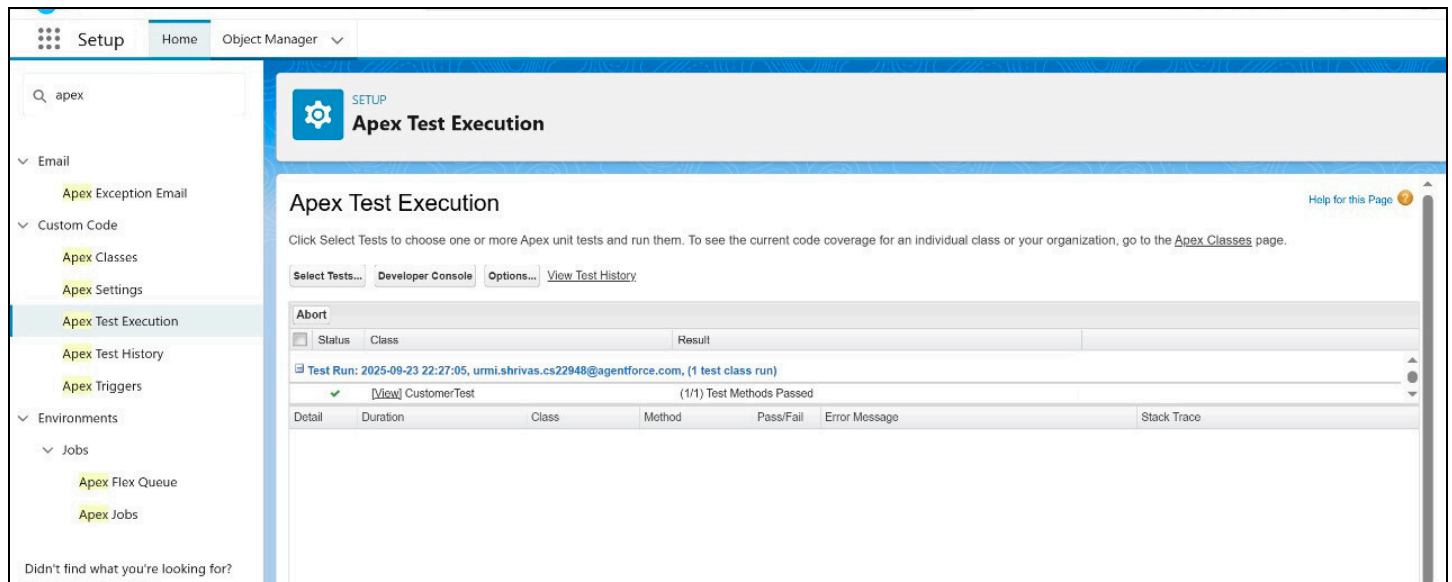


Phase 5: Apex Programming(Developers)

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.

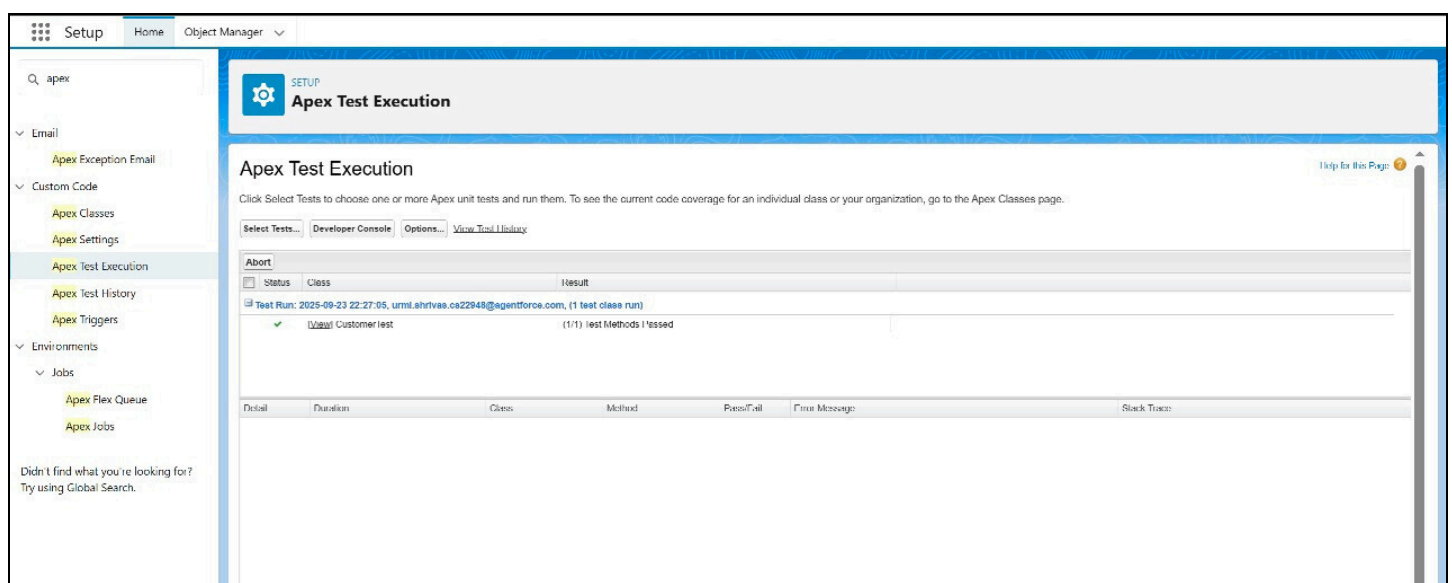




2. Apex Triggers(before/after insert/update/delete)

- 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

Custom Code

Environments

Jobs

Apex Exception Email

Apex Classes

Apex Settings

Apex Test Execution

Apex Test History

Apex Triggers

Apex Flex Queue

Apex Jobs

Didn't find what you're looking for?
Try using Global Search.

Apex Test Execution

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, urml.ahrivaa.cs22948@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

Custom Code

Environments

Jobs

Apex Exception Email

Apex Classes

Apex Settings

Apex Test Execution

Apex Test History

Apex Triggers

Apex Flex Queue

Apex Jobs

Didn't find what you're looking for?
Try using Global Search.

Apex Test Execution

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, urml.ahrivaa.cs22948@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

Custom Code

Environments

Jobs

Apex Exception Email

Apex Classes

Apex Settings

Apex Test Execution

Apex Test History

Apex Triggers

Apex Flex Queue

Apex Jobs

Didn't find what you're looking for?
Try using Global Search.

Apex Test Execution

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, urml.ahrivaa.cs22948@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**.

The screenshot shows the Salesforce Setup interface. On the left, the navigation menu is visible with 'Apex Classes' selected under 'Custom Code'. The main content area is titled 'Apex Classes' and displays the details for the 'CustomerTriggerHandler' class. The 'Apex Class Detail' section shows the class name, namespace prefix, and creation/modification dates. Below this, the 'Class Body' tab is active, showing the following code:

```
1 public class CustomerTriggerHandler {
2
3     public static void beforeInsertUpdate(List<Customer__c> customers) {
4
5         for(Customer__c c : customers) {
6
7             // Set default Name if blank
8             if(String.isBlank(c.Name)) {
9                 c.Name = 'Default Customer';
10                System.debug('Name was blank. Set default.');
```

This screenshot shows the same Salesforce Setup interface, but with more code visible in the 'Class Body' tab. The code continues from the previous screenshot:

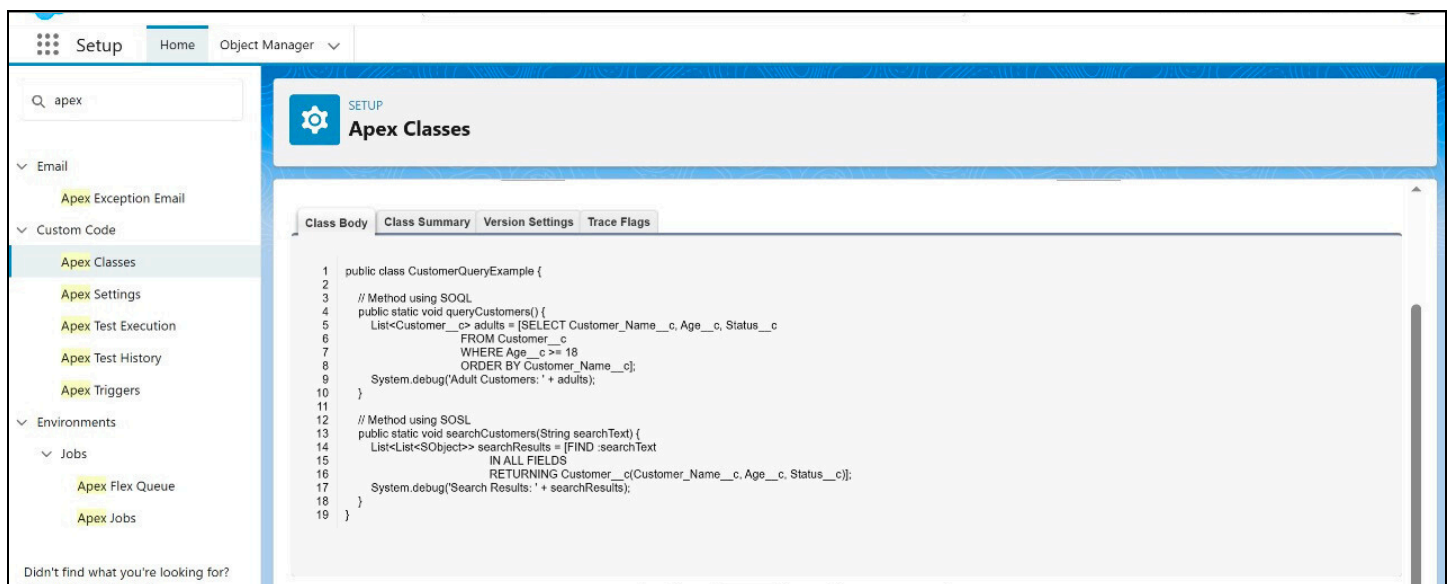
```
14        // Ensure Age__c is valid
15        if(c.Age__c == null || c.Age__c < 0) {
16            c.Age__c = 18;
17            System.debug('Invalid Age. Set default 18.');
```

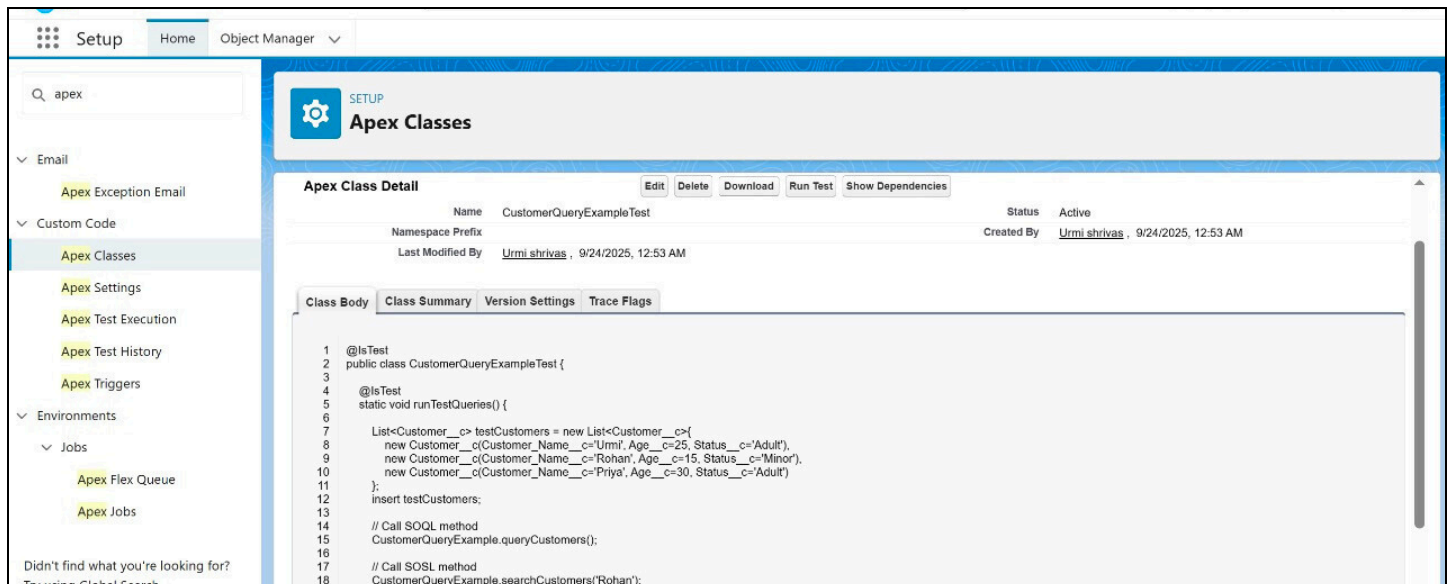
4- Apex Programming – SOQL & SOSL

- 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**.





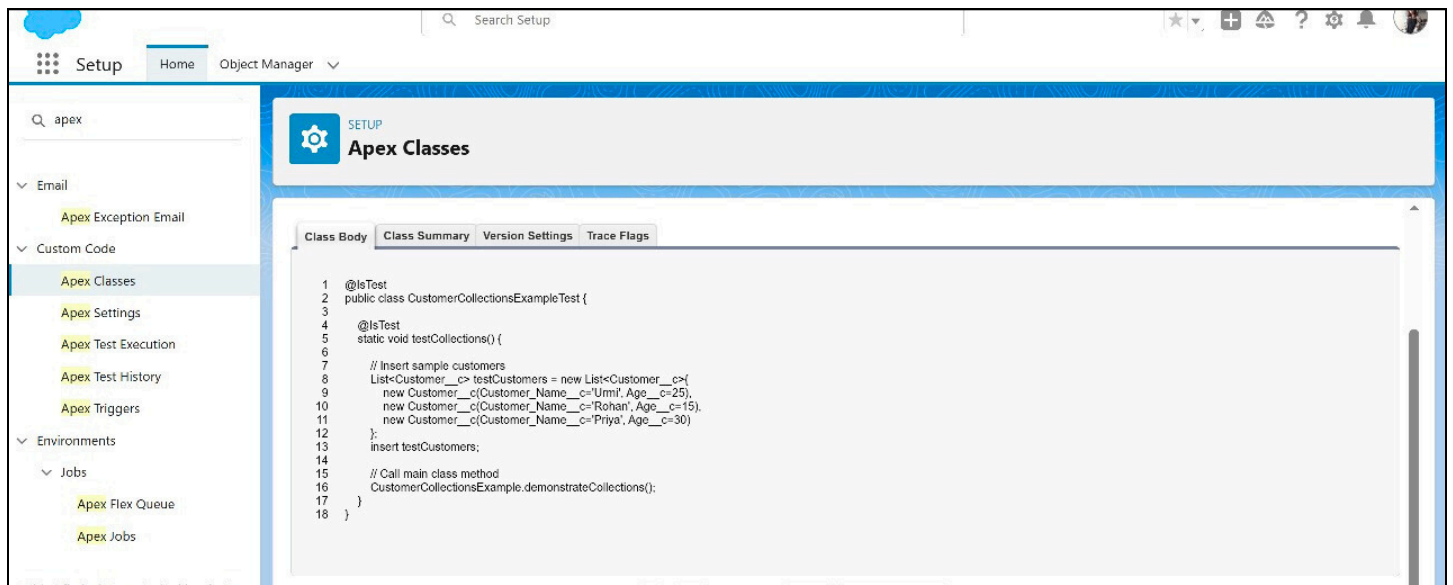
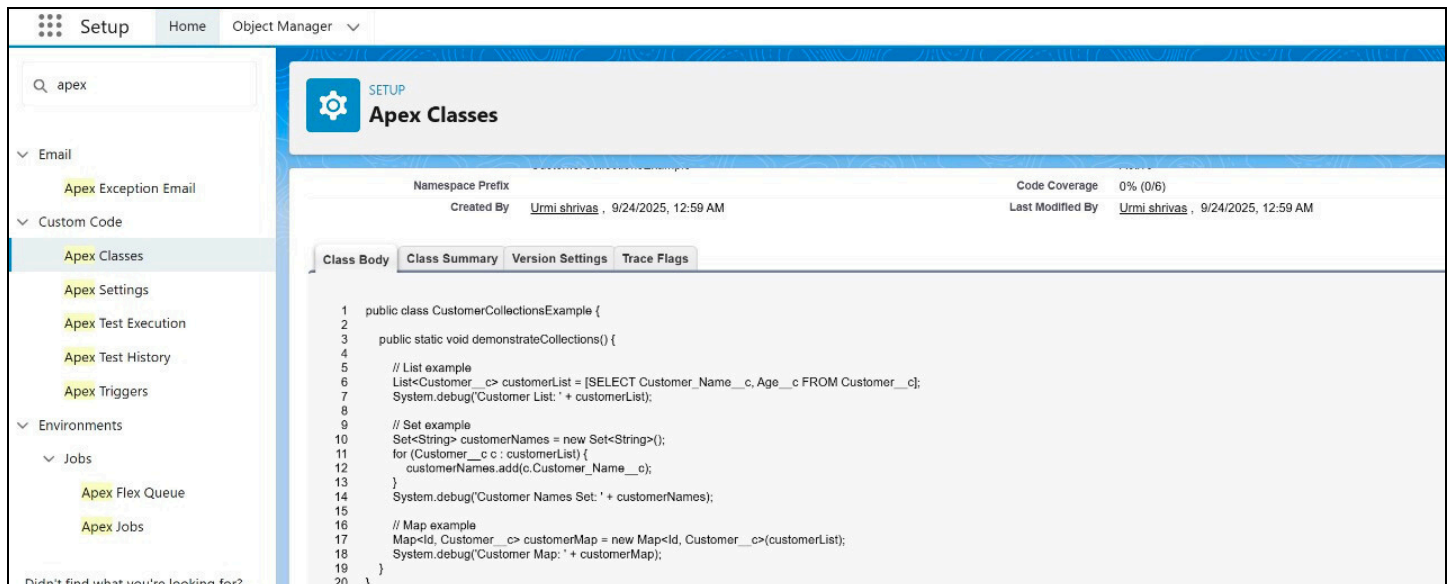
5. Collections (List, Set, Map)

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.

The screenshot shows the Salesforce Setup interface. The left sidebar contains a search bar and a list of setup categories: Email, Custom Code, Apex Classes (selected), Apex Settings, Apex Test Execution, Apex Test History, Apex Triggers, Environments, and Jobs. The main content area is titled 'Apex Classes' and displays the details for the 'CustomerControlExample' class. The class is in the 'CustomerControlExample' namespace, created by 'Urmi shrivas' on 9/24/2025 at 1:04 AM. The status is 'Active' and code coverage is 0% (0/11). The class body is shown in a code editor with the following code:

```
1 public class CustomerControlExample {
2
3     public static void demonstrateControl() {
4
5         List<Customer__c> customers = [SELECT Customer__Name__c, Age__c, Status__c FROM Customer__c];
6
7         // If-Else
8         for (Customer__c c : customers) {
9             if (c.Age__c >= 18) {
10                 System.debug(c.Customer__Name__c + ' is Adult');
11             } else {
12                 System.debug(c.Customer__Name__c + ' is Minor');
13             }
14         }
15     }
16 }
```

The screenshot shows the Salesforce Setup interface. The left sidebar contains a search bar and a list of setup categories: Email, Custom Code, Apex Classes (selected), Apex Settings, Apex Test Execution, Apex Test History, Apex Triggers, Environments, and Jobs. The main content area is titled 'Apex Classes' and displays the details for the 'CustomerCollectionsExampleTest' class. The class is in the 'CustomerCollectionsExampleTest' namespace, created by 'Urmi shrivas' on 9/24/2025 at 1:04 AM. The status is 'Active' and code coverage is 0% (0/11). The class body is shown in a code editor with the following code:

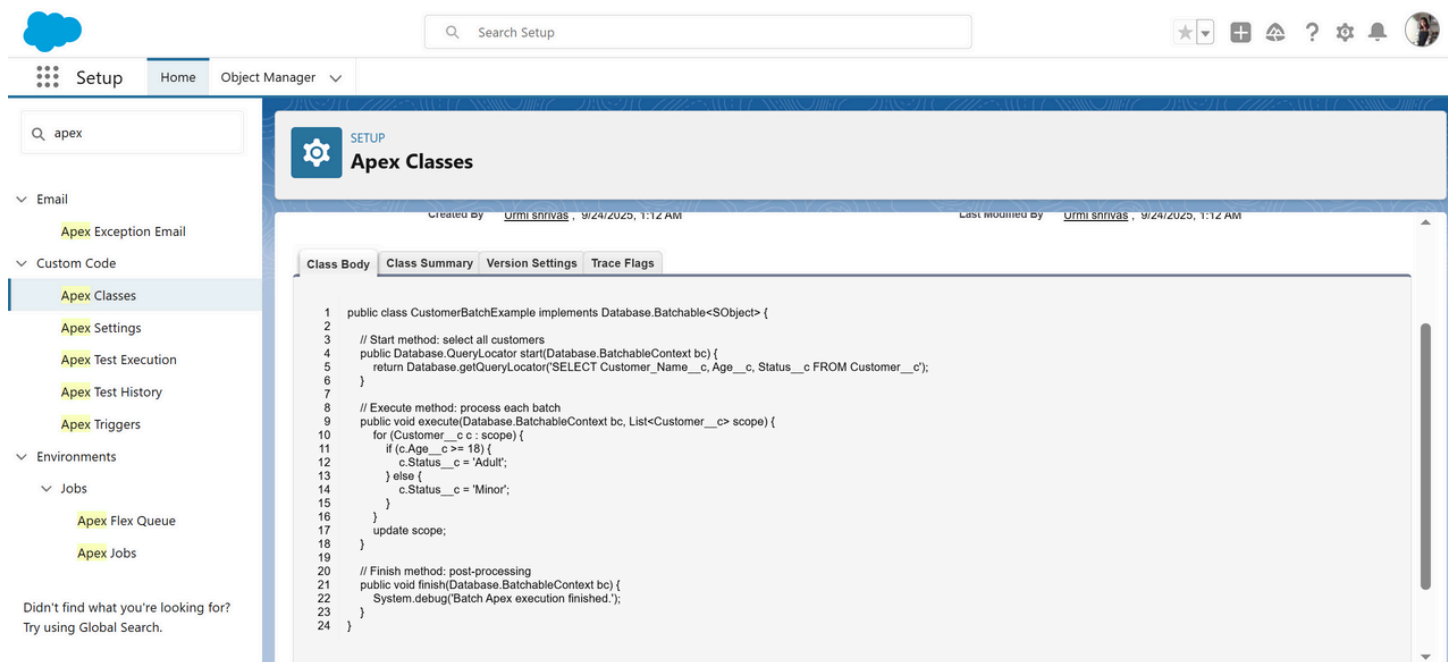
```
1 @IsTest
2 public class CustomerCollectionsExampleTest {
3
4     @IsTest
5     static void testCollections() {
6
7         // Insert sample customers
8         List<Customer__c> testCustomers = new List<Customer__c>{
9             new Customer__c(Customer__Name__c='Urmi', Age__c=25),
10             new Customer__c(Customer__Name__c='Rohan', Age__c=15),
11             new Customer__c(Customer__Name__c='Priya', Age__c=30)
12         };
13         insert testCustomers;
14
15         // Call main class method
16         CustomerCollectionsExample.demonstrateCollections();
17     }
18 }
```


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.



The screenshot shows the Salesforce Setup interface. On the left is a navigation menu with categories like Email, Custom Code, and Environments. Under Custom Code, 'Apex Classes' is selected. The main area displays the 'Apex Classes' page for a class named 'CustomerBatchExample'. The class is implemented as a Batch Apex class that implements the Database.Batchable<SObject> interface. The code includes a start method to select all customers, an execute method to process each batch (updating status based on age), and a finish method for post-processing. The class was created by 'Urmil Srinivas' on 9/24/2020 at 1:12 AM.

```
1 public class CustomerBatchExample implements Database.Batchable<SObject> {
2
3     // Start method: select all customers
4     public Database.QueryLocator start(Database.BatchableContext bc) {
5         return Database.getQueryLocator(SELECT Customer_Name__c, Age__c, Status__c FROM Customer__c);
6     }
7
8     // Execute method: process each batch
9     public void execute(Database.BatchableContext bc, List<Customer__c> scope) {
10         for (Customer__c c : scope) {
11             if (c.Age__c >= 18) {
12                 c.Status__c = 'Adult';
13             } else {
14                 c.Status__c = 'Minor';
15             }
16         }
17         update scope;
18     }
19
20     // Finish method: post-processing
21     public void finish(Database.BatchableContext bc) {
22         System.debug('Batch Apex execution finished.');
```

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**

The screenshot shows the Salesforce Setup interface. On the left, the navigation menu is open, showing 'Setup' > 'Custom Code' > 'Apex Classes'. The main content area displays the 'Apex Class Detail' for 'CustomerQueueableExample'. The class is active, created by 'Urmi shrivastava' on 9/24/2025 at 1:15 AM, and has 0% code coverage. The 'Class Body' tab is selected, showing the following Apex code:

```

1 public class CustomerQueueableExample implements Queueable {
2
3     public void execute(QueueableContext qc) {
4         List<Customer__c> customers = [SELECT Customer_Name__c, Age__c, Status__c FROM Customer__c];
5
6         for (Customer__c c : customers) {
7             if (c.Age__c >= 18) {
8                 c.Status__c = 'Adult';
9             } else {
10                 c.Status__c = 'Minor';
11             }
12         }
13         update customers;
14         System.debug('Queueable Apex job executed.');
```

The screenshot shows the Salesforce Setup interface. On the left, the navigation menu is open, showing 'Setup' > 'Custom Code' > 'Apex Classes'. The main content area displays the 'Apex Class Detail' for 'CustomerQueueableExampleTest'. The class is created by 'Urmi shrivastava' on 9/24/2025 at 1:16 AM. The 'Class Body' tab is selected, showing the following Apex code:

```

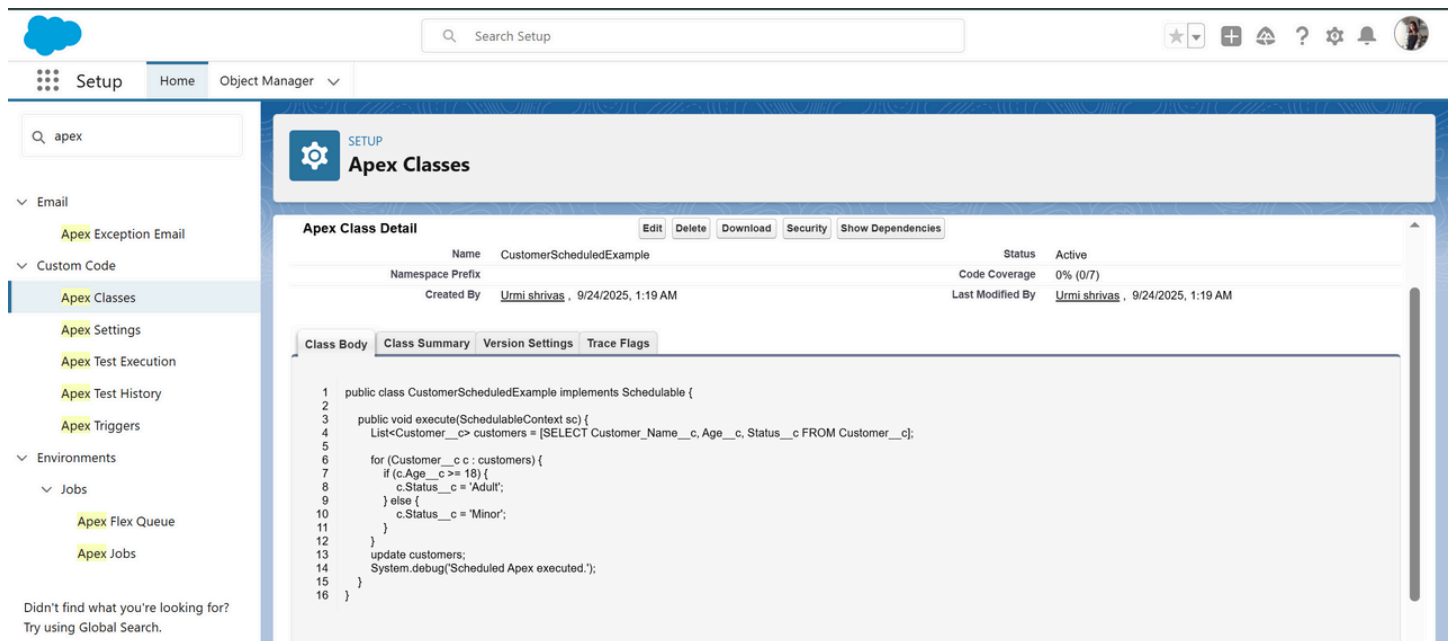
1 @IsTest
2 public class CustomerQueueableExampleTest {
3
4     @IsTest
5     static void testQueueableApex() {
6
7         // Insert sample customers
8         List<Customer__c> testCustomers = new List<Customer__c>{
9             new Customer__c(Customer_Name__c='Urmi', Age__c=25),
10            new Customer__c(Customer_Name__c='Rohan', Age__c=15)
11         };
12         insert testCustomers;
13
14         // Execute Queueable
15         Test.startTest();
16         System.enqueueJob(new CustomerQueueableExample());
17         Test.stopTest();
18     }
19 }
```

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.



The screenshot displays the Salesforce Setup interface. On the left, the navigation menu is visible with categories like Email, Custom Code, and Environments. Under Custom Code, 'Apex Classes' is selected. The main content area shows the 'Apex Class Detail' for 'CustomerScheduledExample'. It includes metadata such as Name, Namespace Prefix, Created By, and Status. Below this, the 'Class Body' tab is active, showing the Apex code. The code defines a class that implements the 'Schedulable' interface and contains an 'execute' method that queries and updates customer records based on their age.

Apex Class Detail

Name	CustomerScheduledExample	Status	Active
Namespace Prefix		Code Coverage	0% (0/7)
Created By	Urmi shrivas	Last Modified By	Urmi shrivas
	9/24/2025, 1:19 AM		9/24/2025, 1:19 AM

Class Body

```
1 public class CustomerScheduledExample implements Schedulable {
2
3     public void execute(SchedulableContext sc) {
4         List<Customer__c> customers = [SELECT Customer_Name__c, Age__c, Status__c FROM Customer__c];
5
6         for (Customer__c c : customers) {
7             if (c.Age__c >= 18) {
8                 c.Status__c = 'Adult';
9             } else {
10                 c.Status__c = 'Minor';
11             }
12         }
13         update customers;
14         System.debug('Scheduled Apex executed.');
```

The screenshot shows the Salesforce Setup interface. On the left is a navigation sidebar with a search bar containing 'apex'. Under the 'Custom Code' section, 'Apex Classes' is selected. The main content area is titled 'Apex Classes' and shows details for a class named 'CustomerScheduledExampleTest'. It includes the 'Namespace Prefix', 'Last Modified By' (Urmi shrivas), and 'Created By' (Urmi shrivas) information. Below this, there are tabs for 'Class Body', 'Class Summary', 'Version Settings', and 'Trace Flags'. The 'Class Body' tab is active, displaying the following Apex code:

```
1 @IsTest
2 public class CustomerScheduledExampleTest {
3
4     @IsTest
5     static void testScheduledApex() {
6
7         // Insert sample customers
8         List<Customer__c> testCustomers = new List<Customer__c>{
9             new Customer__c(Customer_Name__c='Urmi', Age__c=25),
10            new Customer__c(Customer_Name__c='Rohani', Age__c=15)
11        };
12        insert testCustomers;
13
14        // Execute scheduled job in test context
15        Test.startTest();
16        String jobId = System.schedule('Test Schedule Job', '0 0 23 * * ?', new CustomerScheduledExample());
17        Test.stopTest();
18    }
19 }
```

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**.

The screenshot shows the Salesforce Setup interface. On the left, the navigation menu includes Setup, Home, and Object Manager. The main content area is titled "Apex Classes" and displays details for the "CustomerFutureExample" class. The class is active, created by Urmi shrivas on 9/24/2025 at 1:29 AM, and has 0% code coverage (0/7). The class body is visible, showing a public class definition with a static method updateCustomerStatus that uses a SOQL query to update customer status based on age.

Apex Class Detail

Name	CustomerFutureExample	Status	Active
Namespace Prefix		Code Coverage	0% (0/7)
Created By	Urmi shrivas , 9/24/2025, 1:29 AM	Last Modified By	Urmi shrivas , 9/24/2025, 1:29 AM

Class Body

```

1 public class CustomerFutureExample {
2
3     @future
4     public static void updateCustomerStatus(Set<Id> customerIds) {
5         List<Customer__c> customers = [SELECT Age__c, Status__c FROM Customer__c WHERE Id IN :customerIds];
6         for (Customer__c c : customers) {
7             if (c.Age__c >= 18) {
8                 c.Status__c = 'Adult';
9             } else {
10                 c.Status__c = 'Minor';
11             }
12         }
13         update customers;
14         System.debug('Future method executed.');
```

The screenshot shows the Salesforce Setup interface. On the left, the navigation menu includes Setup, Home, and Object Manager. The main content area is titled "Apex Classes" and displays details for the "CustomerFutureExampleTest" class. The class is active, created by Urmi shrivas on 9/24/2025 at 1:30 AM, and has 0% code coverage (0/7). The class body is visible, showing a public class definition with a static method testFutureMethod that inserts sample customer records and calls the updateCustomerStatus method.

Apex Class Detail

Name	CustomerFutureExampleTest	Status	Active
Namespace Prefix		Code Coverage	0% (0/7)
Created By	Urmi shrivas , 9/24/2025, 1:30 AM	Last Modified By	Urmi shrivas , 9/24/2025, 1:30 AM

Class Body

```

1 @IsTest
2 public class CustomerFutureExampleTest {
3
4     @IsTest
5     static void testFutureMethod() {
6
7         List<Customer__c> testCustomers = new List<Customer__c>{
8             new Customer__c(Customer_Name__c='Urmi', Age__c=25),
9             new Customer__c(Customer_Name__c='Rohan', Age__c=15)
10        };
11        insert testCustomers;
12
13        Set<Id> customerIds = new Set<Id>();
14        for (Customer__c c : testCustomers) {
15            customerIds.add(c.Id);
16        }
17
18        Test.startTest();
19        CustomerFutureExample.updateCustomerStatus(customerIds);
20        Test.stopTest();
21    }
22 }
```

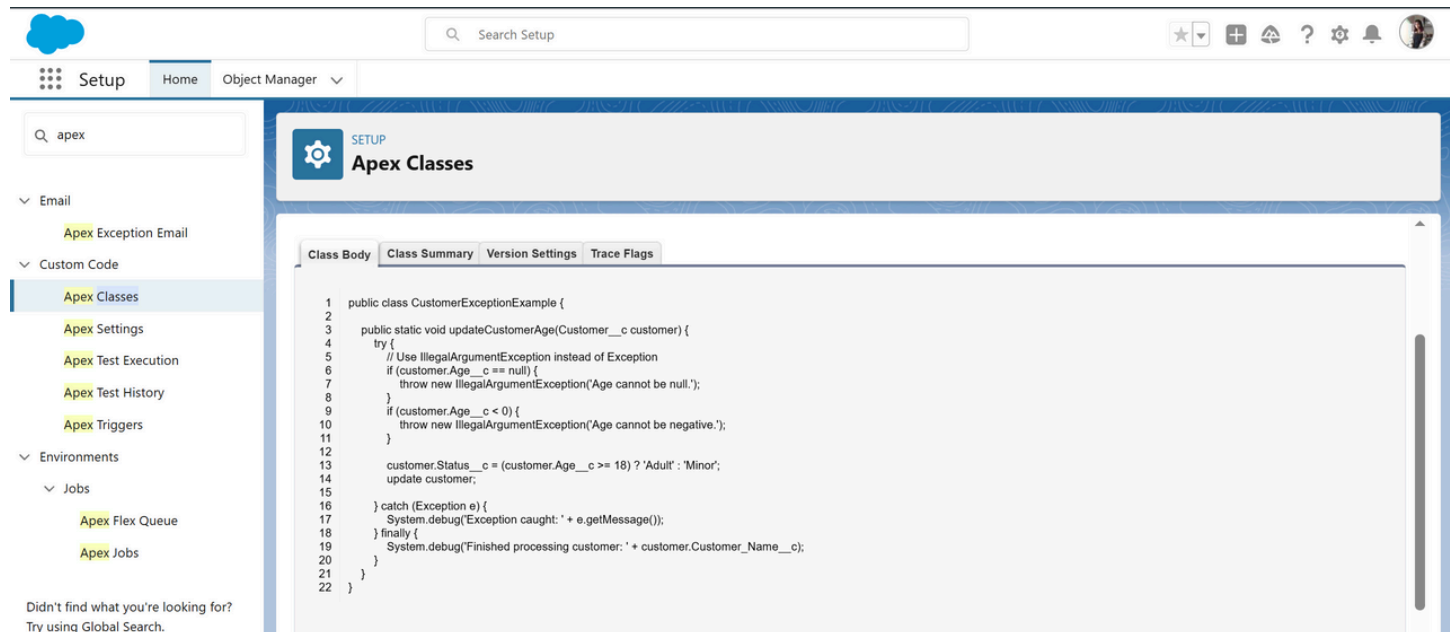
Exception Handling

- 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**.



Search Setup

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

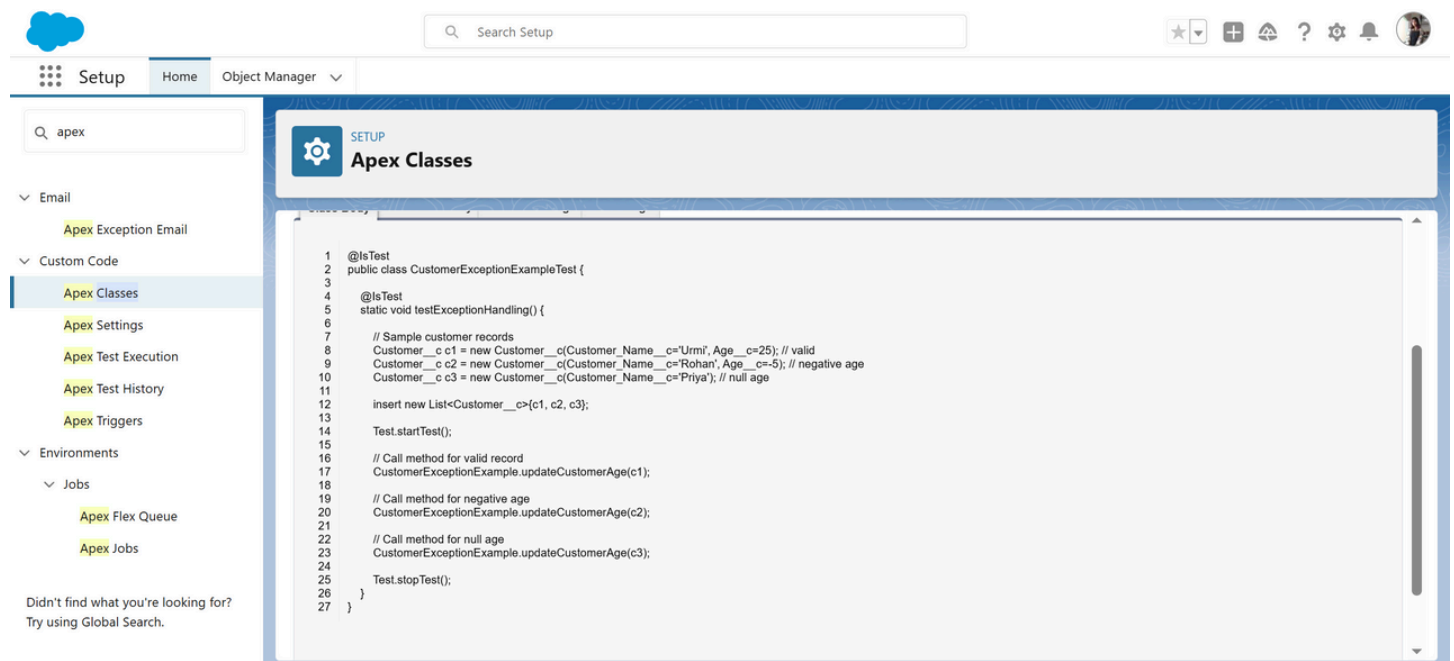
Didn't find what you're looking for? Try using Global Search.

SETUP Apex Classes

Class Body Class Summary Version Settings Trace Flags

```

1 public class CustomerExceptionExample {
2
3     public static void updateCustomerAge(Customer__c customer) {
4         try {
5             // Use IllegalArgumentException instead of Exception
6             if (customer.Age__c == null) {
7                 throw new IllegalArgumentException("Age cannot be null.");
8             }
9             if (customer.Age__c < 0) {
10                throw new IllegalArgumentException("Age cannot be negative.");
11            }
12
13            customer.Status__c = (customer.Age__c >= 18) ? 'Adult' : 'Minor';
14            update customer;
15        } catch (Exception e) {
16            System.debug("Exception caught: " + e.getMessage());
17        } finally {
18            System.debug("Finished processing customer: " + customer.Customer_Name__c);
19        }
20    }
21 }
22
  
```



Search Setup

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 Classes

Class Body Class Summary Version Settings Trace Flags

```

1 @IsTest
2 public class CustomerExceptionExampleTest {
3
4     @IsTest
5     static void testExceptionHandling() {
6
7         // Sample customer records
8         Customer__c c1 = new Customer__c(Customer_Name__c='Urmil', Age__c=25); // valid
9         Customer__c c2 = new Customer__c(Customer_Name__c='Rohan', Age__c=-5); // negative age
10        Customer__c c3 = new Customer__c(Customer_Name__c='Priya'); // null age
11
12        insert new List<Customer__c>{c1, c2, c3};
13
14        Test.startTest();
15
16        // Call method for valid record
17        CustomerExceptionExample.updateCustomerAge(c1);
18
19        // Call method for negative age
20        CustomerExceptionExample.updateCustomerAge(c2);
21
22        // Call method for null age
23        CustomerExceptionExample.updateCustomerAge(c3);
24
25        Test.stopTest();
26    }
27 }
  
```


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.

The screenshot shows the Salesforce Setup interface. On the left, the navigation menu includes Setup, Home, and Object Manager. The main content area is titled 'Apex Classes' and shows the details for the 'CustomerExampleTest' class. The class is active and was last modified by 'Urmi shrivastava' on 9/24/2025 at 1:48 AM. The class body is displayed, showing test logic for inserting sample records and calling main class methods.

Apex Class Detail

Name	CustomerExampleTest	Status	Active
Namespace Prefix		Created By	Urmi shrivastava
Last Modified By	Urmi shrivastava		9/24/2025, 1:48 AM

Class Body

```
1 @IsTest
2 public class CustomerExampleTest {
3
4     @IsTest
5     static void testCustomerLogic() {
6
7         // Insert sample Customer__c records
8         Customer__c c1 = new Customer__c(Customer_Name__c='Urmi', Age__c=25);
9         Customer__c c2 = new Customer__c(Customer_Name__c='Rohan', Age__c=15);
10        Customer__c c3 = new Customer__c(Customer_Name__c='Priya'); // null age
11        insert new List<Customer__c>{c1, c2, c3};
12
13        // Call main class methods
14        CustomerExceptionExample.updateCustomerAge(c1);
15        CustomerExceptionExample.updateCustomerAge(c2);
16        CustomerExceptionExample.updateCustomerAge(c3);
17
18        // Example for Batch Apex
```

The screenshot shows the Salesforce Setup interface. On the left, a sidebar contains a search bar with 'apex' entered and a list of categories: Email, Custom Code, Environments, and Jobs. Under Custom Code, 'Apex Classes' is selected. The main content area is titled 'Apex Classes' and displays a list of Apex classes. The first class, 'CustomerExceptionExample', is expanded, showing its source code. The code includes comments for calling main class methods, examples for Batch Apex, Queueable Apex, and Future Method, and a verification step. The code is as follows:

```
13 // Call main class methods
14 CustomerExceptionExample.updateCustomerAge(c1);
15 CustomerExceptionExample.updateCustomerAge(c2);
16 CustomerExceptionExample.updateCustomerAge(c3);
17
18 // Example for Batch Apex
19 Test.startTest();
20 Database.executeBatch(new CustomerBatchExample(), 2);
21 Test.stopTest();
22
23 // Example for Queueable Apex
24 Test.startTest();
25 System.enqueueJob(new CustomerQueueableExample());
26 Test.stopTest();
27
28 // Example for Future Method
29 Set<Id> customerIds = new Set<Id>{c1.Id, c2.Id};
30 Test.startTest();
31 CustomerFutureExample.updateCustomerStatus(customerIds);
32 Test.stopTest();
33
34 // Verify results (optional)
35 c1 = [SELECT Status__c FROM Customer__c WHERE Id = :c1.Id];
36 System.assertEquals('Adult', c1.Status__c);
37
38 } // closes testCustomerLogic method
39
40 } // closes CustomerExampleTest class
```

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.