

# Phase: 5 Apex Programming

---

## Part 1: Creating the Apex Files

First, we need to create each of the necessary Apex files in your Salesforce org.

Resource:- using the Developer Console.

### Step 1.1: Open the Developer Console

1. Log in to your Salesforce org.
2. Click the **Gear Icon** in the top-right corner.
3. Select **Developer Console** from the dropdown menu.

### Step 1.2: Create Each Apex File

For each of the six files listed below, you will follow the same process:

- In the Developer Console, click **File** → **New** → **Apex Class** (or **Apex Trigger** for the .trigger file).
- Enter the exact file name provided.
- Delete any default text in the editor window.
- Copy and paste the complete code for that file.
- Click **File** → **Save** or press Ctrl + S.

Create the files in this order:

1. **VolunteerTriggerHandler.apxc (Class)**

```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
VolunteerTrigger.apxt ▾ VolunteerTriggerHandler.apxt
Code Coverage: None ▾ API Version: 64 ▾
1 • public class VolunteerTriggerHandler {
2     private static Boolean isRunning = false;
3
4     public static void updateVolunteerTotals(List<volunteer__c> volunteers){
5         if(isRunning) return;
6         isRunning = true;
7
8         Set<Id> volunteerIds = new Set<Id>();
9         for(Volunteer__c v : volunteers){
10             if(v.Id != null){
11                 volunteerIds.add(v.Id);
12             }
13         }
14         if(volunteerIds.isEmpty()) {
15             isRunning = false;
16             return;
17         }
18
19         // 1. Aggregate Donations
20         Map<Id, Decimal> donationTotals = new Map<Id, Decimal>();
21         for(AggregateResult ar : [SELECT Linked_Volunteer__c.Vid, SUM(Amount__c) total FROM Donation__c WHERE Linked_Volunteer__c IN :volunteerIds GROUP BY Linked_Volunteer__c]){
22             donationTotals.put((Id)ar.get('Vid'), (Decimal)ar.get('total'));
23         }
24
25         // 2. Aggregate Hours using your confirmed field names
26         Map<Id, Decimal> hourTotals = new Map<Id, Decimal>();
27         for(AggregateResult ar : [SELECT Volunteer__c.Vid, SUM(Hours_Worked__c) totalHours FROM Volunteer_Task__c WHERE Volunteer__c IN :volunteerIds GROUP BY Volunteer__c]){
28             hourTotals.put((Id)ar.get('Vid'), (Decimal)ar.get('totalHours'));
29         }
30
31         // 3. Prepare updates
32         List<Volunteer__c> updates = new List<Volunteer__c>();
33         List<Volunteer__c> existingVolunteers = [SELECT Id, Total_Donations__c, Total_Hours__c FROM Volunteer__c WHERE Id IN :volunteerIds];
34
35         for(Volunteer__c existing : existingVolunteers){
36             Decimal newDonations = donationTotals.get(existing.Id) == null ? 0 : donationTotals.get(existing.Id);
37             Decimal newHours = hourTotals.get(existing.Id) == null ? 0 : hourTotals.get(existing.Id);
38
39             if(existing.Total_Donations__c != newDonations || existing.Total_Hours__c != newHours){
40                 Volunteer__c vupdate = new Volunteer__c(Id = existing.Id);
41                 vupdate.Total_Donations__c = newDonations;
42                 vupdate.Total_Hours__c = newHours;
43                 updates.add(vupdate);
44             }
45         }
46
47         if(!updates.isEmpty()){
48             update updates;
49         }
50
51         isRunning = false;
52     }
53 }

```

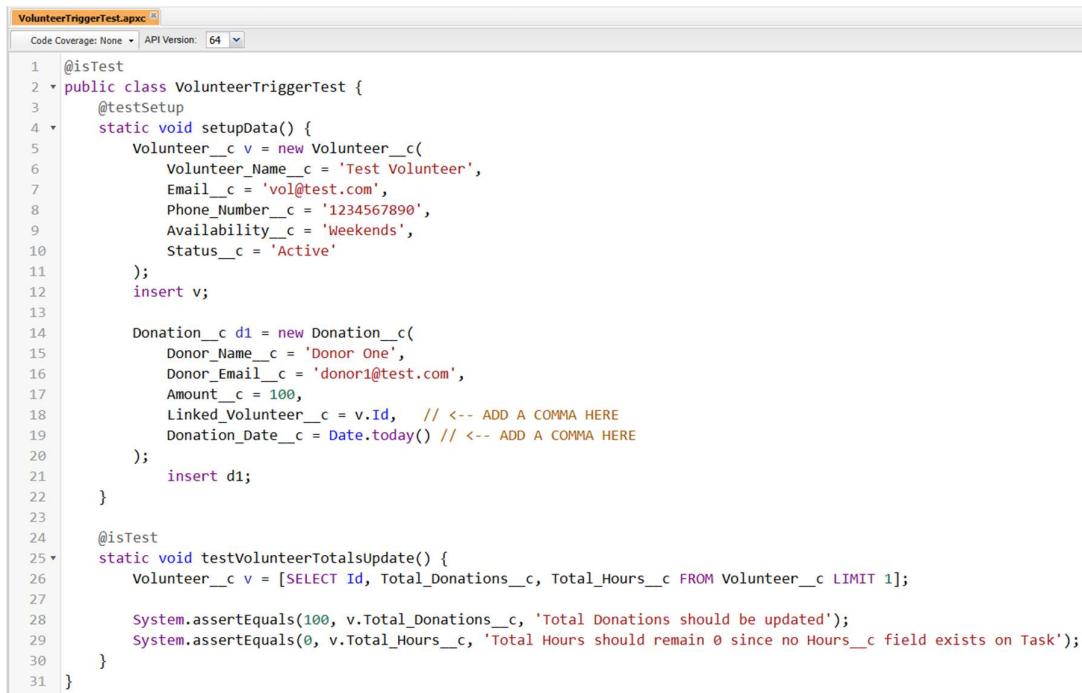
## 2. VolunteerTrigger.apxt (Trigger)

```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
VolunteerTrigger.apxt
Code Coverage: None ▾ API Version: 64 ▾
1 trigger VolunteerTrigger on Volunteer__c (after insert, after update) {
2     if(Trigger.isAfter){
3         VolunteerTriggerHandler.updateVolunteerTotals(Trigger.new);
4     }
5 }

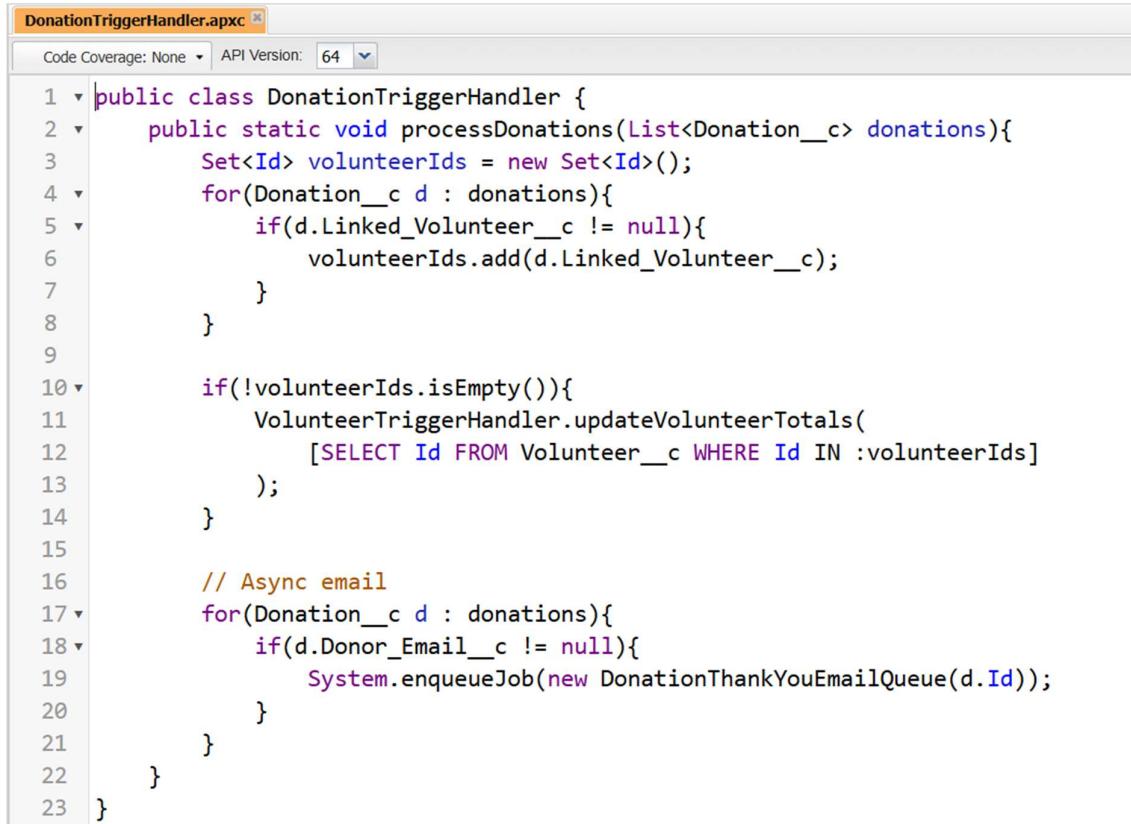
```

### 3. VolunteerTriggerTest.apxc (Class)



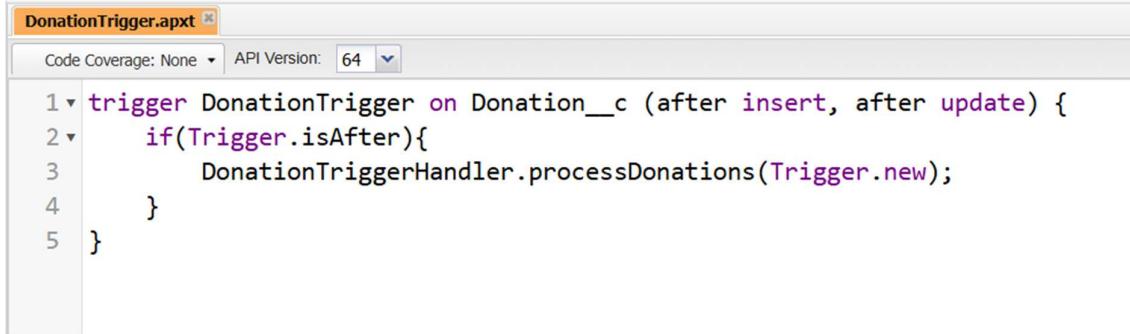
```
VolunteerTriggerTest.apxc
Code Coverage: None API Version: 64
1 @isTest
2 public class VolunteerTriggerTest {
3     @testSetup
4     static void setupData() {
5         Volunteer__c v = new Volunteer__c(
6             Volunteer_Name__c = 'Test Volunteer',
7             Email__c = 'vol@test.com',
8             Phone_Number__c = '1234567890',
9             Availability__c = 'Weekends',
10            Status__c = 'Active'
11        );
12        insert v;
13
14        Donation__c d1 = new Donation__c(
15            Donor_Name__c = 'Donor One',
16            Donor_Email__c = 'donor1@test.com',
17            Amount__c = 100,
18            Linked_Volunteer__c = v.Id, // <-- ADD A COMMA HERE
19            Donation_Date__c = Date.today() // <-- ADD A COMMA HERE
20        );
21        insert d1;
22    }
23
24    @isTest
25    static void testVolunteerTotalsUpdate() {
26        Volunteer__c v = [SELECT Id, Total_Donations__c, Total_Hours__c FROM Volunteer__c LIMIT 1];
27
28        System.assertEquals(100, v.Total_Donations__c, 'Total Donations should be updated');
29        System.assertEquals(0, v.Total_Hours__c, 'Total Hours should remain 0 since no Hours__c field exists on Task');
30    }
31 }
```

### 4. DonationTriggerHandler.apxc (Class)



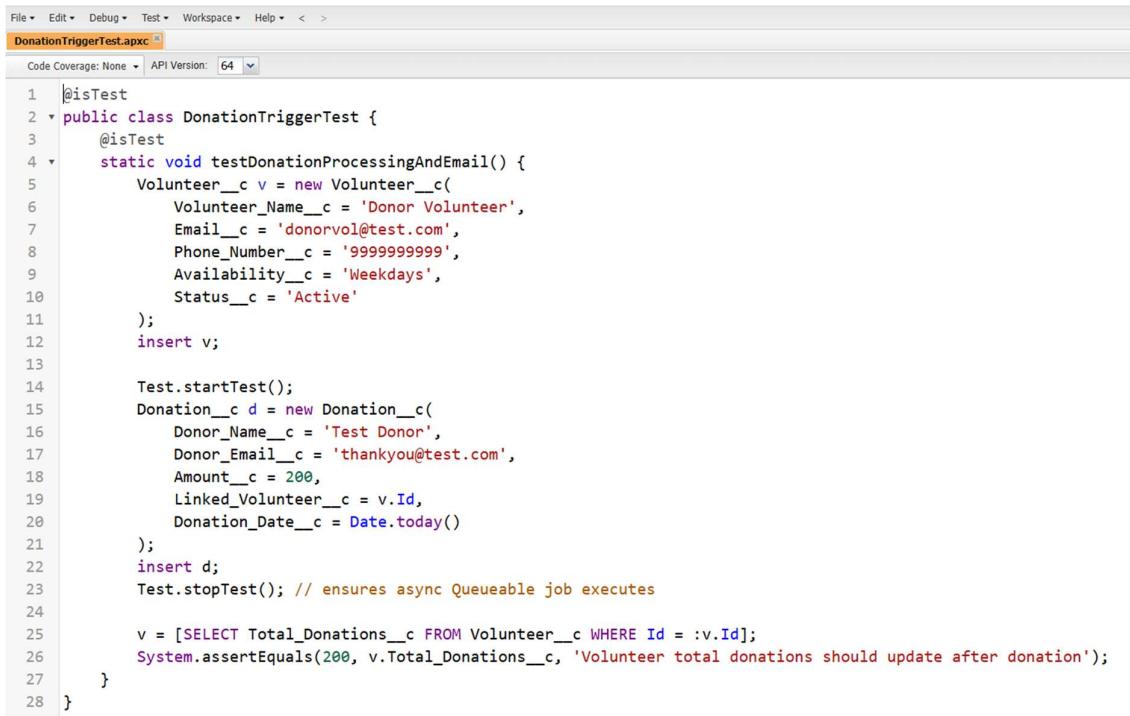
```
DonationTriggerHandler.apxc
Code Coverage: None API Version: 64
1 public class DonationTriggerHandler {
2     public static void processDonations(List<Donation__c> donations){
3         Set<Id> volunteerIds = new Set<Id>();
4         for(Donation__c d : donations){
5             if(d.Linked_Volunteer__c != null){
6                 volunteerIds.add(d.Linked_Volunteer__c);
7             }
8         }
9
10        if(!volunteerIds.isEmpty()){
11            VolunteerTriggerHandler.updateVolunteerTotals(
12                [SELECT Id FROM Volunteer__c WHERE Id IN :volunteerIds]
13            );
14        }
15
16        // Async email
17        for(Donation__c d : donations){
18            if(d.Donor_Email__c != null){
19                System.enqueueJob(new DonationThankYouEmailQueue(d.Id));
20            }
21        }
22    }
23 }
```

## 5. DonationTrigger.apxt (Trigger)



```
trigger DonationTrigger on Donation__c (after insert, after update) {
    if(Trigger.isAfter){
        DonationTriggerHandler.processDonations(Trigger.new);
    }
}
```

## 6. DonationTriggerTest.apxc (Class)

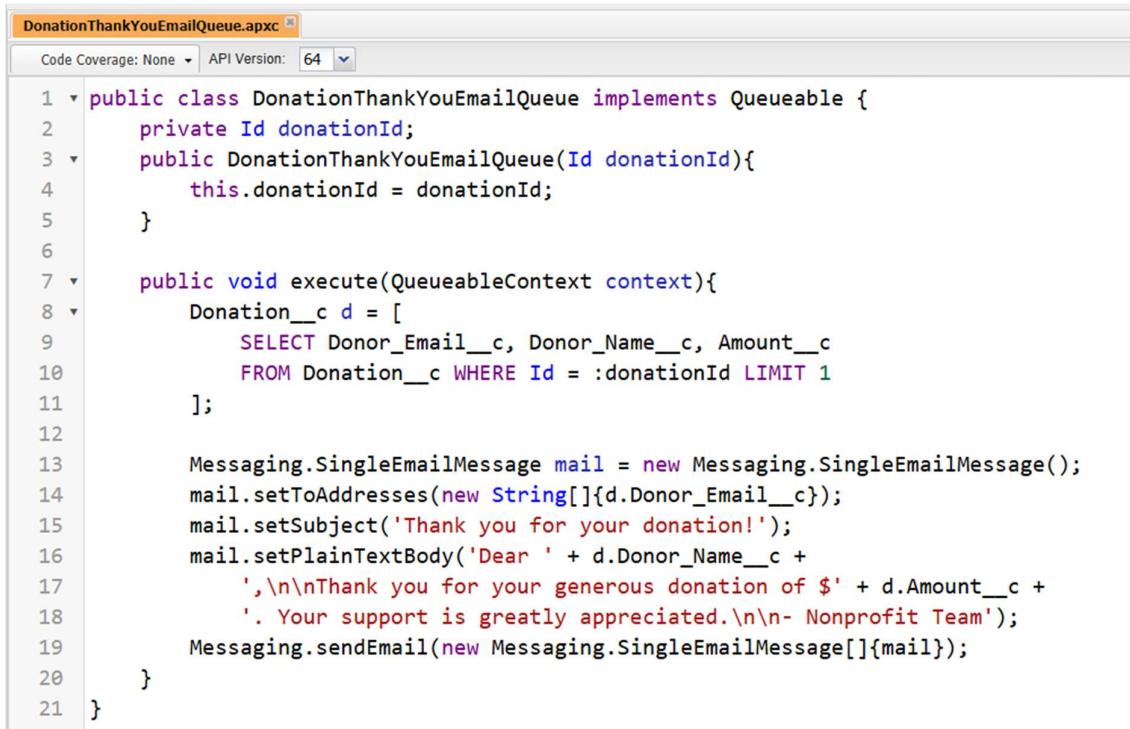


```
@isTest
public class DonationTriggerTest {
    @isTest
    static void testDonationProcessingAndEmail() {
        Volunteer__c v = new Volunteer__c(
            Volunteer_Name__c = 'Donor Volunteer',
            Email__c = 'donorvol@test.com',
            Phone_Number__c = '9999999999',
            Availability__c = 'Weekdays',
            Status__c = 'Active'
        );
        insert v;

        Test.startTest();
        Donation__c d = new Donation__c(
            Donor_Name__c = 'Test Donor',
            Donor_Email__c = 'thankyou@test.com',
            Amount__c = 200,
            Linked_Volunteer__c = v.Id,
            Donation_Date__c = Date.today()
        );
        insert d;
        Test.stopTest(); // ensures async Queueable job executes

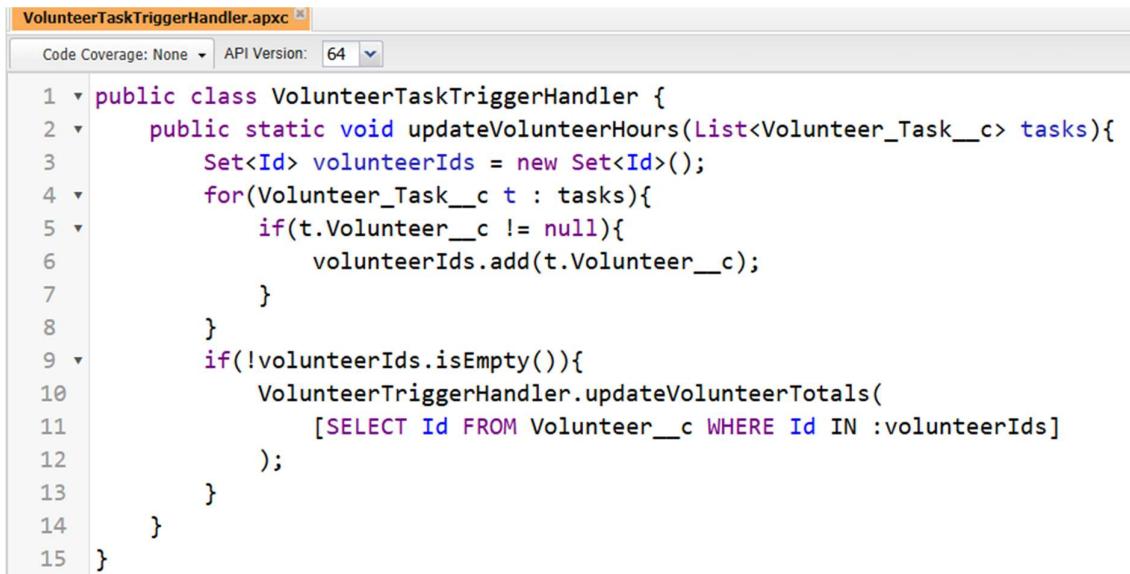
        v = [SELECT Total_Donations__c FROM Volunteer__c WHERE Id = :v.Id];
        System.assertEquals(200, v.Total_Donations__c, 'Volunteer total donations should update after donation');
    }
}
```

## 7. DonationThankYouEmailQueue.apxc (Class)



```
DonationThankYouEmailQueue.apxc
Code Coverage: None API Version: 64
1 public class DonationThankYouEmailQueue implements Queueable {
2     private Id donationId;
3     public DonationThankYouEmailQueue(Id donationId){
4         this.donationId = donationId;
5     }
6
7     public void execute(QueueableContext context){
8         Donation__c d = [
9             SELECT Donor_Email__c, Donor_Name__c, Amount__c
10            FROM Donation__c WHERE Id = :donationId LIMIT 1
11        ];
12
13        Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
14        mail.setToAddresses(new String[]{d.Donor_Email__c});
15        mail.setSubject('Thank you for your donation!');
16        mail.setPlainTextBody('Dear ' + d.Donor_Name__c +
17            ',\n\nThank you for your generous donation of $' + d.Amount__c +
18            '. Your support is greatly appreciated.\n\n- Nonprofit Team');
19        Messaging.sendEmail(new Messaging.SingleEmailMessage[]{mail});
20    }
21 }
```

## 8. VolunteerTaskTriggerHandler.apxc (Class)



```
VolunteerTaskTriggerHandler.apxc
Code Coverage: None API Version: 64
1 public class VolunteerTaskTriggerHandler {
2     public static void updateVolunteerHours(List<Volunteer_Task__c> tasks){
3         Set<Id> volunteerIds = new Set<Id>();
4         for(Volunteer_Task__c t : tasks){
5             if(t.Volunteer__c != null){
6                 volunteerIds.add(t.Volunteer__c);
7             }
8         }
9         if(!volunteerIds.isEmpty()){
10             VolunteerTriggerHandler.updateVolunteerTotals(
11                 [SELECT Id FROM Volunteer__c WHERE Id IN :volunteerIds]
12             );
13         }
14     }
15 }
```

## 9. VolunteerTaskTrigger (Trigger)

The screenshot shows the Salesforce code editor with the tab "VolunteerTaskTrigger.apxt" selected. The code is a trigger for the "Volunteer\_Task\_\_c" object:

```
trigger VolunteerTaskTrigger on Volunteer_Task__c (after insert, after update, after delete) {
    if(Trigger.isAfter){
        VolunteerTaskTriggerHandler.updateVolunteerHours(
            Trigger.isDelete ? Trigger.old : Trigger.new
        );
    }
}
```

## 10. VolunteerTaskTriggerTest (Class)

The screenshot shows the Salesforce code editor with the tab "VolunteerTaskTriggerTest.apxt" selected. The code is a test class for the VolunteerTaskTrigger:

```
trigger VolunteerTaskTrigger on Volunteer_Task__c (after insert, after update, after delete) {
    if(Trigger.isAfter){
        VolunteerTaskTriggerHandler.updateVolunteerHours(
            Trigger.isDelete ? Trigger.old : Trigger.new
        );
    }
}
```

## 11. VolunteerTotalBatch.apxc (Class)

The screenshot shows the Salesforce code editor with the tab "VolunteerTotalBatch.apxc" selected. The code is a batchable class for calculating volunteer totals:

```
public class VolunteerTotalsBatch implements Database.Batchable<SObject>, Database.Stateful {
    public Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id FROM Volunteer__c');
    }

    public void execute(Database.BatchableContext bc, List<Volunteer__c> scope) {
        VolunteerTriggerHandler.updateVolunteerTotals(scope);
    }

    public void finish(Database.BatchableContext bc) {
        System.debug('Volunteer totals recalculation completed.');
    }
}
```

## 12. VolunteerTotalsScheduler.cls (Class)

The screenshot shows the Salesforce code editor with the tab "VolunteerTotalsScheduler.apxc" selected. The code is a schedulable class for running the batchable class:

```
public class VolunteerTotalsScheduler implements Schedulable {
    public void execute(SchedulableContext sc) {
        Database.executeBatch(new VolunteerTotalsBatch(), 200);
    }
}
```

### 13. VolunteerTotalsScheduler.cls (Class)

```
VolunteerBatchScheduler/Test.apxc
Code Coverage: None API Version: 64
1 @isTest
2 * public class VolunteerBatchSchedulerTest {
3     @isTest
4     static void testBatchExecution() {
5         // Setup Volunteer with all required fields
6         Volunteer__c v = new Volunteer__c(
7             Volunteer_Name__c = 'Batch Volunteer',
8             Availability__c = 'Weekends',
9             Status__c = 'Active',
10            Email__c = 'vol1@test.com'
11        );
12        insert v;
13
14        // Setup Donation
15        Donation__c d = new Donation__c(
16            Donor_Name__c = 'Test Donor',
17            Linked_Volunteer__c = v.Id,
18            Amount__c = 500,
19            Donation_Date__c = Date.today(),
20            Donor_Email__c = 'donori@test.com'
21        );
22        insert d;
23
24        // Setup Volunteer Task using your confirmed field names
25        Volunteer_Task__c t = new Volunteer_Task__c(
26            Name = 'Test Task',
27            Volunteer__c = v.Id,
28            Hours_Worked__c = 8,
29            Priority__c = 'High',
30            Due_Date__c = Date.today().addDays(7)
31        );
32        insert t;
33
34        Test.startTest();
35        Database.executeBatch(new VolunteerTotalsBatch(), 1);
36        Test.stopTest();
37
38        // Re-query the volunteer to get the updated values
39        v = [SELECT Total_Donations__c, Total_Hours__c FROM Volunteer__c WHERE Id = :v.Id];
40
41        System.assertEquals(500, v.Total_Donations__c, 'Batch should recalc donations');
42        System.assertEquals(8, v.Total_Hours__c, 'Batch should recalc hours');
43    }
44
45    @isTest
46    static void testSchedulerExecution() {
47        Test.startTest();
48        String cron = '0 0 1 1 ? 2099';
49        System.schedule('Test Volunteer Totals Scheduler', cron, new VolunteerTotalsscheduler());
50        Test.stopTest();
51
52        System.assertEquals(1, [SELECT COUNT() FROM CronTrigger WHERE CronJobDetail.Name = 'Test Volunteer Totals Scheduler']);
53    }
54 }
```

After saving all 13 files without any errors, you are ready to run the tests.

## Part 2: Running Tests and Capturing Results

Now we will execute the test classes to verify all the logic works correctly and to check your code coverage.

### Step 2.1: Navigate to the Test Execution Tab

1. In the Developer Console, click on the **Test** tab in the menu bar.
2. Select **New Run**.

## Step 2.2: Run the Trigger Test

1. From the "Test Classes" list, select **VolunteerTriggerTest**.
2. Click the **Add Selected** button to move it to the right-hand column.
3. Click the **Run** button.

The test will execute, and the results will appear in the "Tests" tab.

The screenshot shows the Salesforce IDE interface. At the top, there's a header bar with tabs for 'Logs', 'Tests' (which is selected), 'Checkpoints', 'Query Editor', 'View State', 'Progress', and 'Problems'. Below the header, the main area displays the code for 'VolunteerTriggerTest.apxc'. The code includes a setup method that inserts a new 'Volunteer' record and a donation record. It also contains a test method 'testVolunteerTotalsUpdate()' that performs assertions on the volunteer's total donations and hours. At the bottom, the 'Tests' tab is active, showing a table with one row labeled 'Test Run'. The table has columns for 'Status', 'Test Run', 'Enqueued Time', and 'Duration'.

```
1 @isTest
2 public class VolunteerTriggerTest {
3     @testSetup
4     static void setupData() {
5         Volunteer__c v = new Volunteer__c(
6             Volunteer_Name__c = 'Test Volunteer',
7             Email__c = 'vol@test.com',
8             Phone_Number__c = '1234567890',
9             Availability__c = 'Weekends',
10            Status__c = 'Active'
11        );
12        insert v;
13
14        Donation__c d1 = new Donation__c(
15            Donor_Name__c = 'Donor One',
16            Donor_Email__c = 'donor1@test.com',
17            Amount__c = 100,
18            Linked_Volunteer__c = v.Id, // <-- ADD A COMMA HERE
19            Donation_Date__c = Date.today() // <-- ADD A COMMA HERE
20        );
21        insert d1;
22    }
23
24    @isTest
25    static void testVolunteerTotalsUpdate() {
26        Volunteer__c v = [SELECT Id, Total_Donations__c, Total_Hours__c FROM Volunteer__c LIMIT 1];
27
28        System.assertEquals(100, v.Total_Donations__c, 'Total Donations should be updated');
29        System.assertEquals(0, v.Total_Hours__c, 'Total Hours should remain 0 since no Hours__c field exists on Task');
30    }
31 }
```

Status	Test Run	Enqueued Time	Duration
✓	TestRun @ 5:13:46 pm		

## Step 2.3: Run the Trigger Test

1. From the "Test Classes" list, select **DonationTriggerTest**.
2. Click the **Add Selected** button to move it to the right-hand column.
3. Click the **Run** button.

The test will execute, and the results will appear in the "Tests" tab.

The screenshot shows the Salesforce IDE interface. The top part displays the code for 'DonationTriggerTest.apxc'. The code is a test class for a trigger. It includes logic to insert a new volunteer record and a donation record, then assert that the volunteer's total donations are updated correctly. The bottom part shows the 'Tests' tab of the test results table, which lists two successful test runs.

```

1  @isTest
2  public class DonationTriggerTest {
3      @isTest
4      static void testDonationProcessingAndEmail() {
5          Volunteer__c v = new Volunteer__c(
6              Volunteer_Name__c = 'Donor Volunteer',
7              Email__c = 'donorvol@test.com',
8              Phone_Number__c = '9999999999',
9              Availability__c = 'Weekdays',
10             Status__c = 'Active'
11         );
12         insert v;
13
14         Test.startTest();
15         Donation__c d = new Donation__c(
16             Donor_Name__c = 'Test Donor',
17             Donor_Email__c = 'thankyou@test.com',
18             Amount__c = 200,
19             Linked_Volunteer__c = v.Id,
20             Donation_Date__c = Date.today()
21         );
22         insert d;
23         Test.stopTest(); // ensures async Queueable job executes
24
25         v = [SELECT Total_Donations__c FROM Volunteer__c WHERE Id = :v.Id];
26         System.assertEquals(200, v.Total_Donations__c, 'Volunteer total donations should update after donation');
27     }
28 }

```

Logs	Tests	Checkpoints	Query Editor	View State	Progress	Problems
Status	Test Run					
✓	TestRun @ 5:13:46 pm					
✓	TestRun @ 5:17:17 pm					
					Enqueued Time	Duration

### Step 2.3: Run the Trigger Test

1. From the "Test Classes" list, select **VolunteerTaskTriggerTest**.
2. Click the **Add Selected** button to move it to the right-hand column.
3. Click the **Run** button.

The test will execute, and the results will appear in the "Tests" tab.

```

1  @isTest
2  public class VolunteerTaskTriggerTest {
3      @isTest
4      static void testTaskTriggerDoesNotBreak() {
5          Volunteer__c v = new Volunteer__c(
6              Volunteer_Name__c = 'Task Volunteer',
7              Email__c = 'taskvol@test.com',
8              Phone_Number__c = '8888888888',
9              Availability__c = 'Evenings',
10             Status__c = 'Active'
11         );
12         insert v;
13
14         Test.startTest();
15         Volunteer_Task__c t = new Volunteer_Task__c(
16             Name = 'Old Age Homes Food Supply',
17             Volunteer__c = v.Id,
18             Status__c = 'Not Started',
19             Priority__c = 'Medium',           // Provide a valid picklist value (e.g., 'Low', 'Medium', 'High')
20             Due_Date__c = Date.today().addDays(7) // Set a future due date
21         );
22         insert t;
23         update t;
24         delete t;
25         Test.stopTest();
26
27         v = [SELECT Total_Hours__c, Total_Donations__c FROM Volunteer__c WHERE Id = :v.Id];
28         System.assertEquals(0, v.Total_Hours__c, 'Volunteer hours should remain 0');
29     }
30 }

```

Status	Test Run	Enqueued Time	Duration
✓	TestRun @ 5:13:46 pm		
✓	TestRun @ 5:17:17 pm		
✓	TestRun @ 5:19:14 pm		

## Step 2.4: Run the Batch and Scheduler Test

1. Just like before, click **Test → New Run**.
2. This time, select **VolunteerBatchSchedulerTest** from the list.
3. Click **Add Selected**, then click **Run**.

```

1  @isTest
2  public class VolunteerBatchSchedulerTest {
3      @isTest
4      static void testBatchExecution() {
5          // Setup Volunteer with all required fields
6          Volunteer__c v = new Volunteer__c(
7              Volunteer_Name__c = 'Batch Volunteer',
8              Availability__c = 'Weekends',
9              Status__c = 'Active',
10             Email__c = 'vol1@test.com'
11         );
12         insert v;
13
14         // Setup Donation
15         Donation__c d = new Donation__c(
16             Donor_Name__c = 'Test Donor',
17             Linked_Volunteer__c = v.Id,
18             Amount__c = 500,
19             Donation_Date__c = Date.today(),
20             Donor_Email__c = 'donor1@test.com'
21         );
22         insert d;
23
24         // Setup Volunteer Task using your confirmed field names
25         Volunteer_Task__c t = new Volunteer_Task__c(
26             Name = 'Test Task',
27             Volunteer__c = v.Id,
28             Hours_Worked__c = 8,
29             Priority__c = 'High',
30             Due_Date__c = Date.today().addDays(7)
31         );
32         insert t;
33
34         Test.startTest();
35         Database.executeBatch(new VolunteerTotalsBatch(), 1);
36         Test.stopTest();
37
38         // Re-query the volunteer to get the updated values
39         v = [SELECT Total_Donations__c, Total_Hours__c FROM Volunteer__c WHERE Id = :v.Id];
40
41         System.assertEquals(500, v.Total_Donations__c, 'Batch should recalc donations');
42         System.assertEquals(8, v.Total_Hours__c, 'Batch should recalc hours');
43     }
44
45     @isTest
46     static void testschedulerExecution() {
47         Test.startTest();
48         String cron = '@ 0 0 1 1 ? 2099';
49         System.schedule('Test Volunteer Totals Scheduler', cron, new VolunteerTotalsScheduler());
50         Test.stopTest();
51
52         System.assertEquals(1, [SELECT COUNT() FROM CronTrigger WHERE CronJobDetail.Name = 'Test Volunteer Totals Scheduler']);
53     }
54 }

```

## Step 2.4: Check Overall Code Coverage

After all tests have passed, you can view your organization's overall Apex code coverage.

1. In the "Tests" tab of the Developer Console, look at the top-right corner. You will see "Overall Code Coverage".
2. Click the name of a class to see which lines were covered by your tests.

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage
						Class
✓	TestRun @ 5:13:46 pm			0	1	97%
✓	TestRun @ 5:13:17 pm			0	1	Overall
✓	TestRun @ 5:10:14 pm			0	1	DonationThankYouEmailQueue
✓	TestRun @ 5:26:44 pm			0	1	DonationTrigger
✓	7079K00000005J7	Thu Sep 25 2025 17:27:11 GM...		0	2	2/2
						VolunteerTaskTrigger
						100% 3/3
						VolunteerTaskTriggerHandler
						100% 8/8

## Part 3: Scheduling the Nightly Job

The final step is to run the script that officially schedules your batch job to run every night.

### Step 3.1: Run the Anonymous Apex Script

1. In the Developer Console, click **Debug** → **Open Execute Anonymous Window**.
2. Copy and paste the following script into the window:

## Apex

```
VolunteerTotalsScheduler job = new VolunteerTotalsScheduler();  
String cron = '0 0 2 * * ?'; // Runs daily at 2 AM  
System.schedule('Nightly Volunteer Totals Job', cron, job);
```

3. Click the **Execute** button at the bottom right.

### Step 3.2: Verify the Job is Scheduled

1. Go back to the main Salesforce window (not the Developer Console).
2. Click the **Gear Icon** (⚙) and select **Setup**.
3. In the "Quick Find" box on the left, type Scheduled Jobs.
4. Click on the **Scheduled Jobs** link under the "Jobs" heading.

You will see a list of all scheduled jobs in your org.

The All Scheduled Jobs page lists all of the jobs scheduled by your users. Multiple job types may display on this page. You can delete scheduled jobs if you have the permission to do so.

**Percentage of Scheduled Jobs Used: 1%**  
You have currently used 1 scheduled Apex jobs out of an allowed organization limit of 100 active or scheduled jobs. To learn about how this limit is calculated and what contributes to it see the [Lightning Platform Apex Limits](#) topic.

**View:** [All Scheduled Jobs](#) [Create New View](#)

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
<a href="#">Manage</a>   <a href="#">Del</a>   <a href="#">Pause Job</a>	Nightly Volunteer Totals Job	Koyi Venkata Likhith Sai	25/09/2025, 3:16 pm		26/09/2025, 2:00 am	Scheduled Apex	08egK00000C6BDD