Phase 5: Apex Programming (Developer)

Airline Management System

Salesforce-Based Passenger & Operations Management

Step 1: Log in to Salesforce Developer Org

1. Go to Salesforce Developer and sign up if you don't have an org.
2. Log in to your **Developer Edition** or **Sandbox**.
3. Switch to **Lightning Experience** for easier navigation.

Step 2: Create Custom Objects

Purpose: Define your Airline entities.

Steps:

1. Go to **Setup → Object Manager → Create → Custom Object**.
2. Create these objects:
   - Airline__c
   - Flight__c
   - Passenger__c
   - Booking__c
3. Add **custom fields**:
   - Flight__c: Name, Status (Picklist: Scheduled, Completed, Cancelled), DepartureDate (DateTime)
   - Airline__c: Name, TotalFlights (Number)
   - Passenger__c: Name, Email
   - Booking__c: Flight (Lookup to Flight__c), Passenger (Lookup to Passenger__c)

Step 3: Create Apex Classes

Purpose: Write logic to manage your objects.

Steps:

1. Go to **Setup → Apex Classes → New**.
2. Create a class like Airline:

- Click **Save**.
- Test the class using **Developer Console → Debug → Open Execute Anonymous Window**:

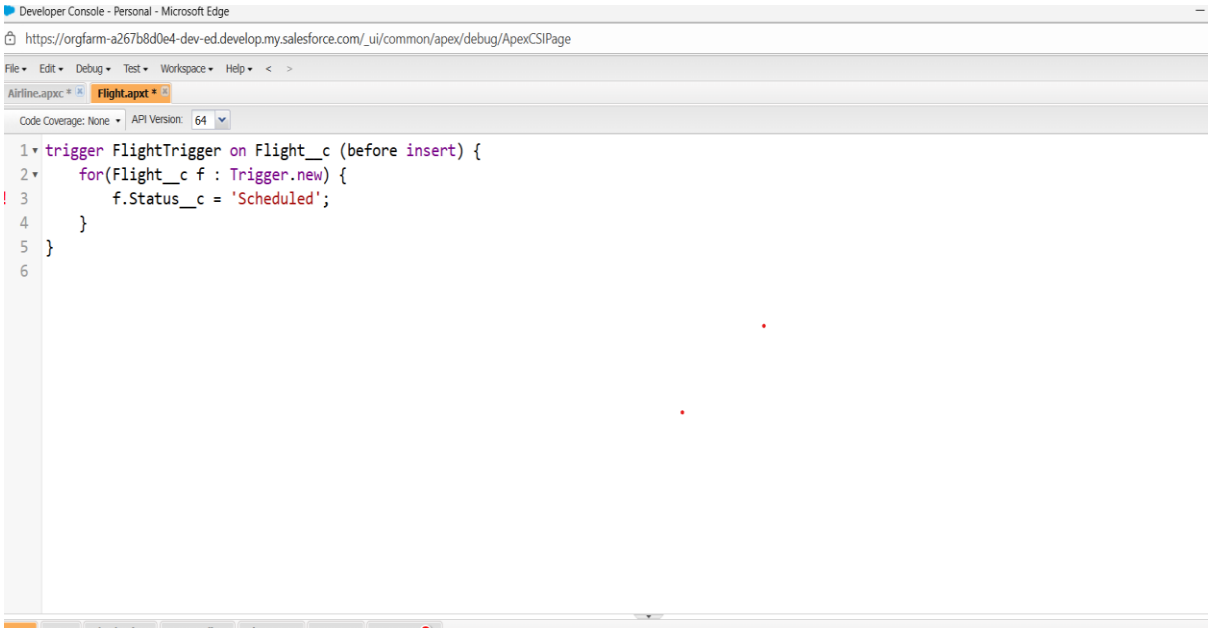
```
Airline a = new Airline('SkyAir', 100);
a.displayInfo();
```

Step 4: Create Apex Triggers

Purpose: Automate actions when records are created/updated/deleted.

Steps:

1. Go to **Setup → Object Manager → Flight__c → Triggers → New**.
2. Create a trigger:



3. Click **Save**.

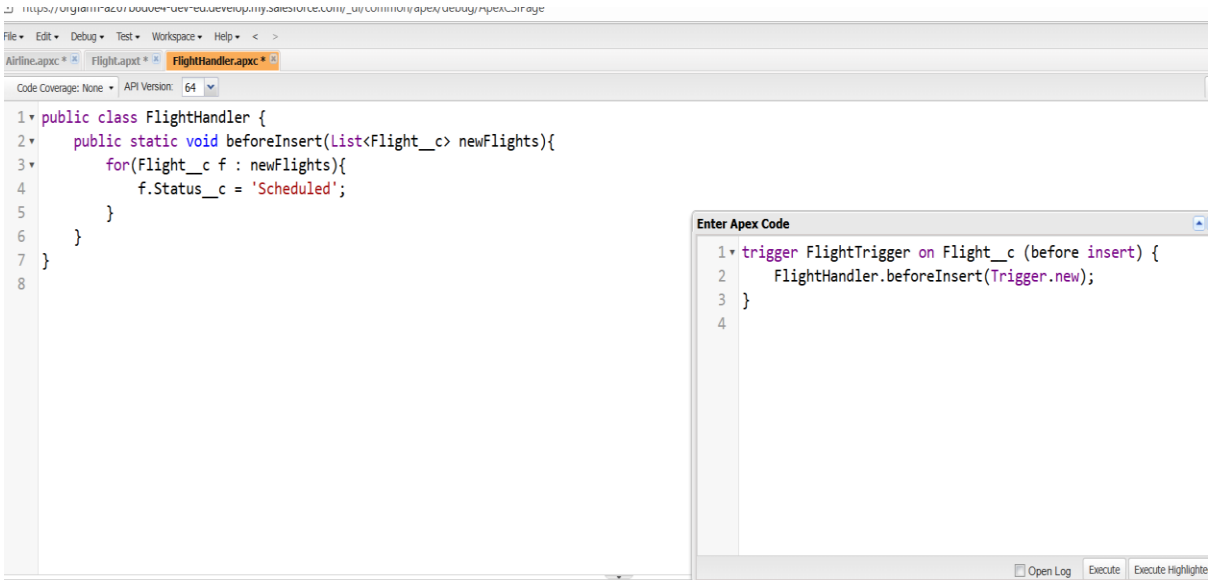
4. Test: Go to **Flight__c** → **New Record**, add a flight → **Save** → check Status is automatically "Scheduled".

Step 5: Implement Trigger Handler (Design Pattern)

Purpose: Keep triggers clean.

Steps:

1. Create **Apex Class** → **FlightHandler**:



2. Update trigger to delegate:

```
trigger FlightTrigger on Flight__c (before insert) {
FlightHandler.beforeInsert(Trigger.new);
}
```

Step 6: Use SOQL & SOSL

Purpose: Query records in Salesforce.

Steps:

1. Open **Developer Console** → **Query Editor**.
2. Run a SOQL query:

```
SELECT Name, Status__c FROM Flight__c WHERE Status__c='Scheduled'
```

3. Run a SOSL query in **Apex**:

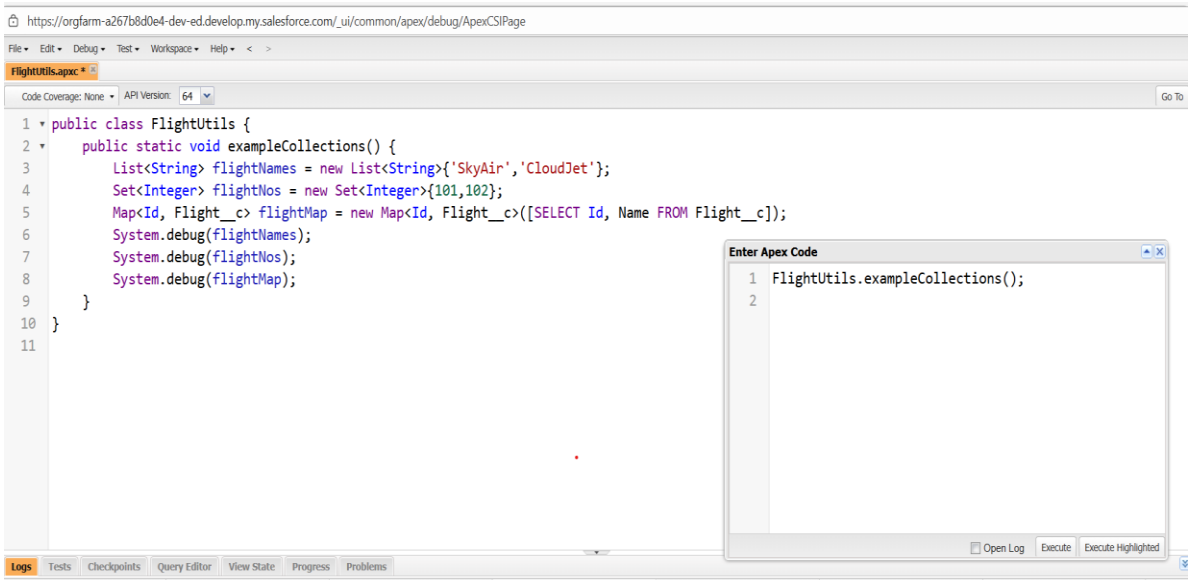
```
List<List<SObject>> results = [FIND 'SkyAir*' IN ALL FIELDS RETURNING Flight__c(Name, Status__c)];
```

Step 7: Use Collections

Purpose: Handle multiple records in code.

Steps:

1. Create a new Apex Class FlightUtils:



2. Execute in **Developer Console → Execute Anonymous**:

```
FlightUtils.exampleCollections();
```

Step 8: Use Control Statements

Purpose: Handle conditions and loops.

Steps:

1. In your Apex Class, use if-else:

```
for(Flight__c f : [SELECT Name, Status__c FROM Flight__c]){
    if(f.Status__c == 'Cancelled') System.debug(f.Name + ' is cancelled.');
    else System.debug(f.Name + ' is active.');
}
```

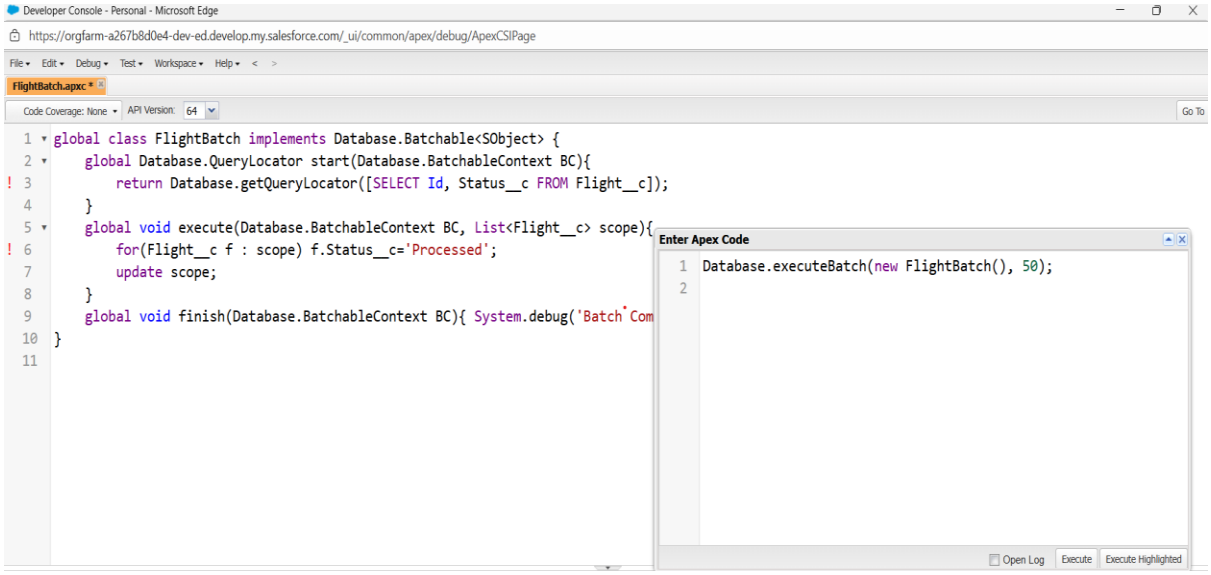
2. Execute in **Anonymous Window**.

Step 9: Implement Batch Apex

Purpose: Process large datasets.

Steps:

1. Create **Apex Class → FlightBatch**:



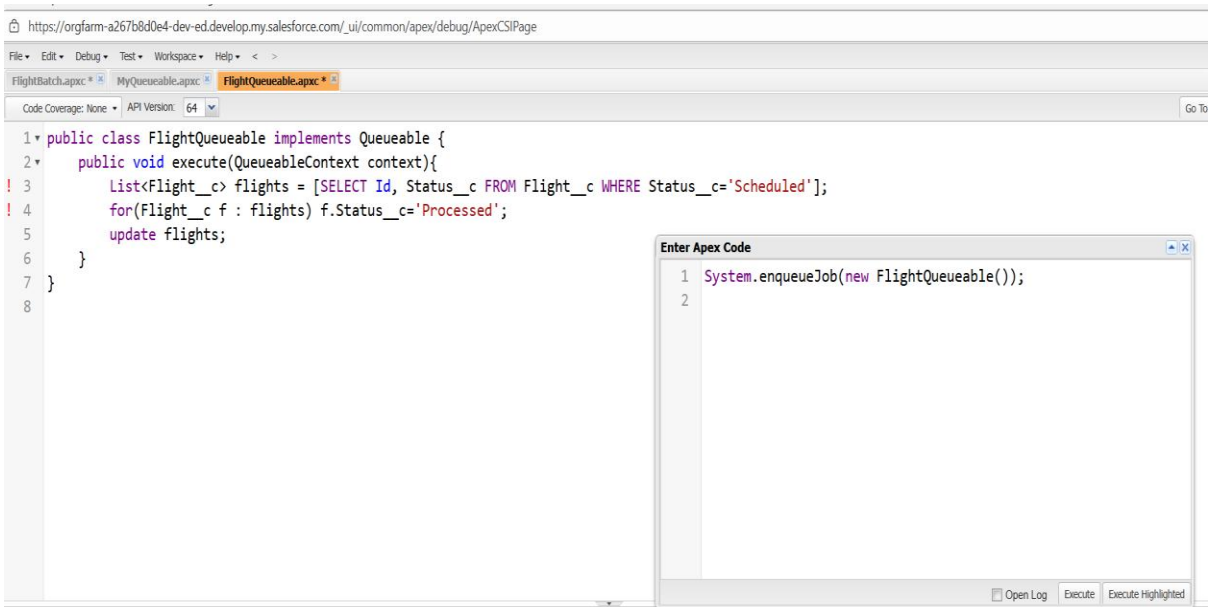
2. Run in **Execute Anonymous**:

```
Database.executeBatch(new FlightBatch(), 50);
```

Step 10: Implement Queueable Apex

Steps:

1. Create **Apex Class → FlightQueueable**:



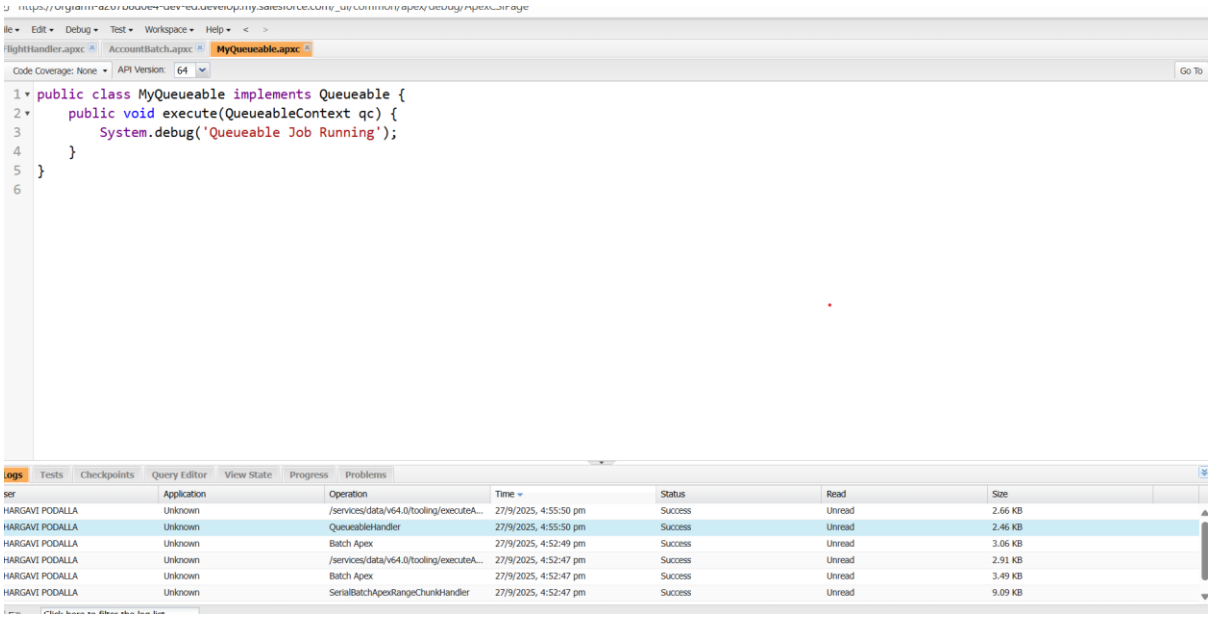
2. Enqueue job:

```
System.enqueueJob(new FlightQueueable());
```

Step 11: Implement Scheduled Apex

Steps:

1. Create **Apex Class → FlightScheduler**:



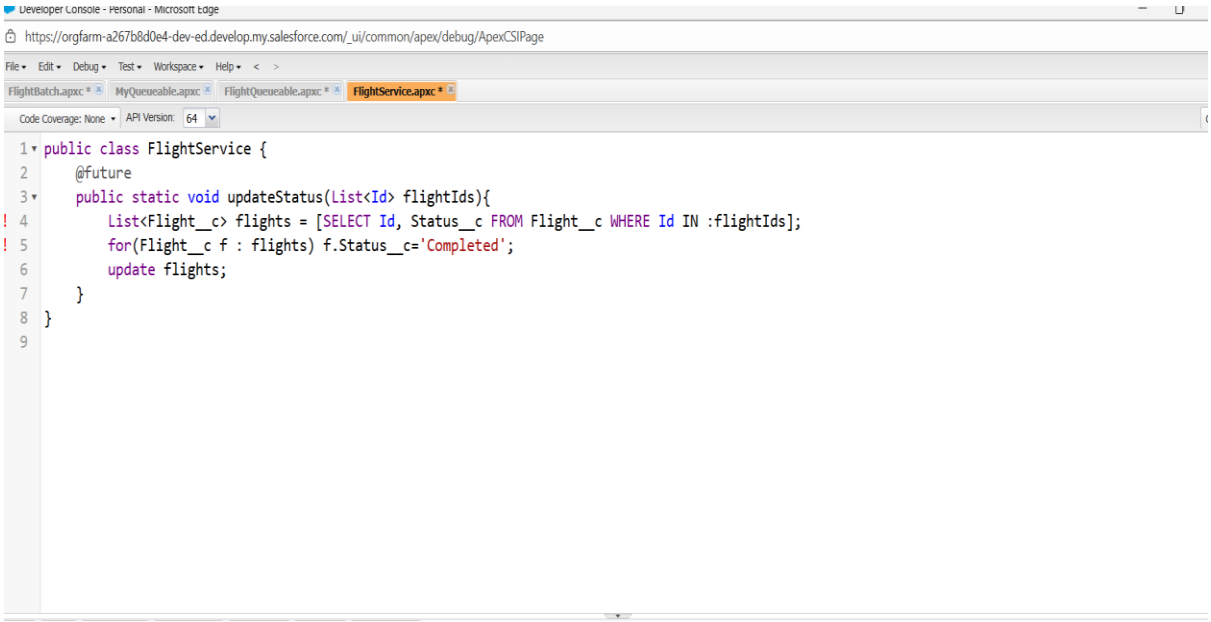
2. Schedule job in **Execute Anonymous**:

```
String cronExp = '0 0 12 * * ?'; // Daily 12 PM
System.schedule('DailyFlightJob', cronExp, new FlightScheduler());
```

Step 12: Use Future Methods

Steps:

1. Create **Apex Class → FlightService**:



2. Call asynchronously:

```
FlightService.updateStatus(new List<Id>{'a0F1t00000123AB'});
```

Step 13: Exception Handling

Steps:

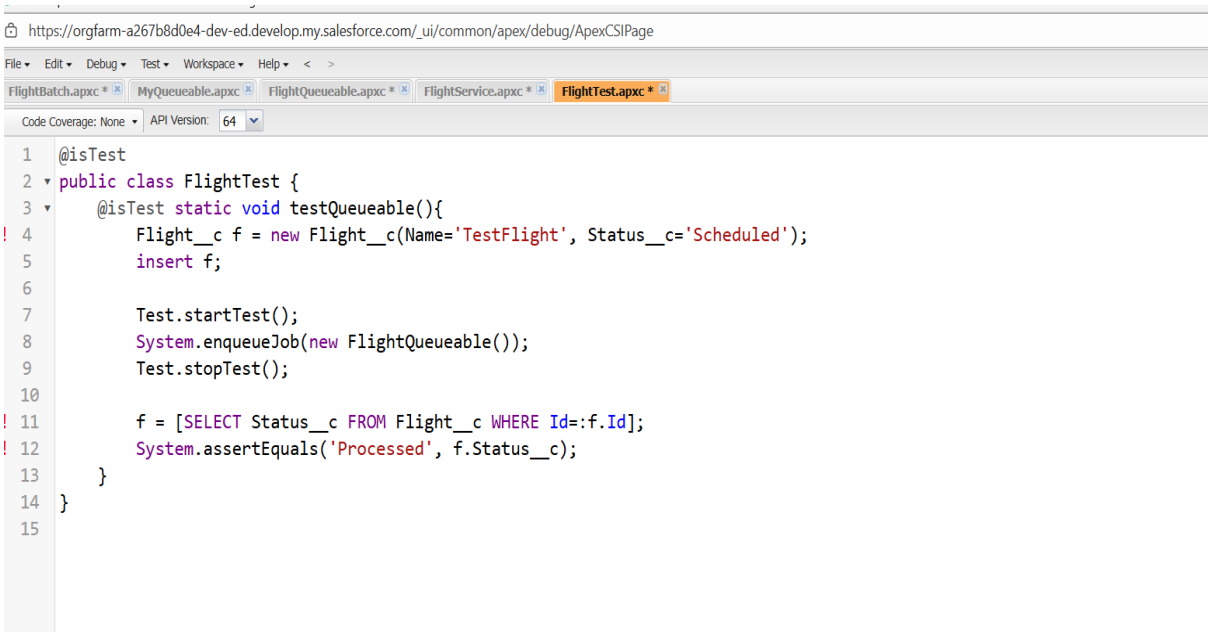
1. Wrap DML operations in **try-catch**:

```
try {
    Flight__c f = [SELECT Id FROM Flight__c WHERE Name='SkyAir' LIMIT 1];
    f.Status__c='Delayed';
    update f;
} catch(DmlException e) {
    System.debug('Error: ' + e.getMessage());
}
```

Step 14: Write Test Classes

Steps:

1. Create **Apex Class → FlightTest**:



2. Run tests in **Setup → Apex Test Execution → Run All Tests**.

Following this, you will have **fully implemented Apex logic, triggers, asynchronous processing, and test coverage in your Salesforce Org** for the Airline Management System.