Use Future Methods

Apex Class Name : AccountProcessor

Code :

```
public class AccountProcessor {
    @future

  public static void someFutureMethod(List<id> scope) {

   Account[] updates = new Account[] {};
       for (AggregateResult ar : [
              select AccountId a, count(Id) c
              from Contact
              where AccountId in :scope
              group by AccountId
              ]) {
          updates.add(new Account(
                Id = (Id) ar.get('a'),
                Number_of_Contacts__c = (Decimal) ar.get('c')
                ));
       }
       update updates;
   }

}
```

Apex Test Class Name : AccountProcessorTest

Code :

```
@IsTest
public class AccountProcessorTest {
  public static testmethod void TestAccountProcessorTest() {

Test.startTest();
     Account a = new Account();
        a.Name = 'Test Account';
        Insert a;

      Contact cont = New Contact();

      cont.FirstName ='Bob';
      cont.LastName ='Masters';
      cont.AccountId = a.Id;
      Insert cont;

    Test.stopTest() ;
    Contact ACC = [select AccountId from Contact where id = :a.id LIMIT 1];


        System.assert(Cont.AccountId != null);
        System.assertequals(cont.id, ACC.AccountId);

  }
}
```

Use Batch Apex

Apex Class Name : LeadProcessor

Code :

```
global class LeadProcessor implements    Database.Batchable<Sobject>
{
   global Database.QueryLocator start(Database.BatchableContext bc)
   {
      return Database.getQueryLocator([Select LeadSource From Lead ]);
   }

   global void execute(Database.BatchableContext bc, List<Lead> scope)
   {
         for (Lead Leads : scope)
         {
            Leads.LeadSource = 'Dreamforce';
         }
      update scope;
   }

   global void finish(Database.BatchableContext bc){   }
}
```

Apex Test Class Name : LeadProcessorTest

Code :

```
@isTest
public class LeadProcessorTest
{
   static testMethod void testMethod1()
   {
      List<Lead> lstLead = new List<Lead>();
      for(Integer i=0 ;i <200;i++)
      {
         Lead led = new Lead();
         led.FirstName ='FirstName';
         led.LastName ='LastName'+i;
         led.Company ='demo'+i;
         lstLead.add(led);
      }

      insert lstLead;

      Test.startTest();

         LeadProcessor obj = new LeadProcessor();
         DataBase.executeBatch(obj);

      Test.stopTest();
   }
}
```

Control Processes with Queueable Apex

Apex Class Name : AddPrimaryContact

Code :

```
public class AddPrimaryContact implements Queueable
{
   private Contact c;
   private String state;
   public  AddPrimaryContact(Contact c, String state)
   {
      this.c = c;
      this.state = state;
   }
   public void execute(QueueableContext context)
   {
       List<Account> ListAccount = [SELECT ID, Name ,(Select id,FirstName,LastName from
contacts ) FROM ACCOUNT WHERE BillingState = :state LIMIT 200];
       List<Contact> lstContact = new List<Contact>();
       for (Account acc:ListAccount)
       {
            Contact cont = c.clone(false,false,false,false);
            cont.AccountId =  acc.id;
            lstContact.add( cont );
       }

       if(lstContact.size() >0 )
       {
          insert lstContact;
       }

   }

}
```

Apex Test Class Name : AddPrimaryContactTest

Code :

```
@isTest
public class AddPrimaryContactTest {

    @isTest static void testQueueable(){
        //<-----@testSetup
        List<Account> accounts = new List<Account>();
        for (Integer i = 0; i < 50; i++){accounts.add(new Account(name = 'acc' + i, BillingState =
'NY')); }
        for (Integer i = 50; i < 100; i++){accounts.add(new Account(name = 'acc' + i, BillingState =
'CA')); }
        insert accounts;

            String strState = 'CA';
        Contact cont = new Contact(LastName = 'TstsName');
        AddPrimaryContact updater = new AddPrimaryContact(cont, strState);
        //<-----@testSetup

        //<-----@testExecution
        Test.startTest();
            System.enqueueJob(updater);
        Test.stopTest();
        //<-----@testExecution

        //<-----@testResult
        System.assertEquals(50, [select count() from Contact where accountID IN (SELECT id
FROM Account WHERE BillingState = :strState)]);
        //<-----@testResult
    }
}
```

Schedule Jobs Using the Apex Scheduler

Apex Class Name : DailyLeadProcessor

Code :

```
global class DailyLeadProcessor implements Schedulable {

   global void execute(SchedulableContext ctx) {
      List<Lead> lList = [Select Id, LeadSource from Lead where LeadSource = null];

      if(!lList.isEmpty()) {
                              for(Lead l: lList) {
                                       l.LeadSource = 'Dreamforce';
                              }
                              update lList;
                     }
   }
}
```

Apex Test Class Name : DailyLeadProcessorTest

Code :

```
@isTest
private class DailyLeadProcessorTest {
   public static String CRON_EXP = '0 0 0 15 3 ? 2022';
   static testmethod void testScheduledJob() {
      for (Integer i = 0; i < 200; i++) {
         Leads.add(new lead(
            name='Dream force'+i
         ));
      }
      insert Leads;
```

```
    }
        Test.startTest();
       String jobId = System.schedule('ScheduledApexTest',
          CRON_EXP,
          new DailyLeadProcessorTest());
}
```