Asynchronous Apex

Use Future Methods

AccountProcessor.apxc

```
public class AccountProcessor {

  @future
  public static void countContacts(List<Id> accountIds){

    List<Account> accList = [Select Id, Number_Of_Contacts__c, (Select Id from Contacts) from
Account where Id in :accountIds];

    For(Account acc : accList){

      acc.Number_Of_Contacts__c = acc.Contacts.size();
    }

    update accList;
  }
}
```

AccountProcessorTest.apxc

```
@isTest
public class AccountProcessorTest {

  public static testmethod void testAccountProcessor(){

    Account a = new Account();
    a.Name = 'Test Account';
    insert a;
```

```
        Contact con = new Contact();
        con.FirstName = 'Binary';
        con.LastName = 'Programming';
        con.AccountId = a.Id;

        insert con;

        List<Id> accListId = new List<Id>();
        accListId.add(a.Id);

        Test.startTest();
        AccountProcessor.countContacts(accListId);
        Test.stopTest();

        Account acc = [Select Number_Of_Contacts__c from Account where Id=: a.Id];
        System.assertEquals(Integer.valueOf(acc.Number_Of_Contacts__c),1);
    }
}
```

Use Batch Apex

LeadProcessor.apxc

```
global class LeadProcessor implements Database.Batchable<sObject> {
   global Integer count = 0;


   global Database.QueryLocator start(Database.BatchableContext bc){
      return Database.getQueryLocator('SELECT ID, LeadSource FROM Lead');
   }


   global void execute (Database.BatchableContext bc, List<Lead> L_list){
      List<lead> L_list_new = new List<lead>();


      for(lead L:L_list){
         L.leadsource = 'Dreamforce';
         L_list_new.add(L);
         count +=1;
      }
      update L_list_new;
   }


   global void finish(Database.BatchableContext bc){
      system.debug('count = '+ count);
   }


}
```

LeadProcessorTest.apxc

```
@isTest
public class LeadProcessorTest {

    @isTest
    public static void test(){
        List<lead> L_list = new List<lead>();

        for(Integer i=0; i<200;i++){
            Lead L = new lead();
            L.LastName = 'name' + i;
            L.Company = 'Company';
            L.Status = 'Random Status';
            L_list.add(L);
        }
        insert L_list;

        Test.startTest();
        LeadProcessor lp = new LeadProcessor();
        Id batchId = Database.executeBatch(lp);
        Test.stopTest();
    }
}
```

Control Processes with Queueable Apex

AddPrimaryContact.apxc

```
public class AddPrimaryContact implements Queueable{


  private Contact con;

  private String state;


  public AddPrimaryContact(Contact con, String state){

    this.con = con;

    this.state = state;

  }


  public void execute(QueueableContext context){

    List<Account> accounts = [Select Id, Name, (Select FirstName,LastName, Id from contacts)

                  from Account where BillingState = :state Limit 200];

    List<Contact> primaryContacts = new List<Contact>();


    for(Account acc:accounts){

      Contact c = con.clone();

      c.AccountId = acc.Id;

      primaryContacts.add(c);

    }


    if(primaryContacts.size() > 0){

      insert primaryContacts;
```

```
        }
    }


}
```

AddPrimaryContactTest.apxc

```
@isTest
public class AddPrimaryContactTest {

    static testmethod void testQueueable(){
        List<Account> testAccounts = new List<Account>();
        for(Integer i=0;i<50;i++){
            testAccounts.add(new Account(Name='Account '+i,BillingState='CA'));
        }
        for(Integer j=0;j<50;j++){
            testAccounts.add(new Account(Name='Account '+j,BillingState='NY'));
        }
        insert testAccounts;

        Contact testContact = new Contact(FirstName = 'John', LastName = 'Doe');
        insert testContact;

        AddPrimaryContact addit = new addPrimaryContact(testContact, 'CA');
        Test.startTest();

        system.enqueueJob(addit);

        Test.stopTest();


        System.assertEquals(50, [Select count() from Contact where accountId in (Select Id from
Account where BillingState='CA')]);

    }



}
```

Schedule Jobs Using the Apex Scheduler

DailyLeadProcessor.apxc

```
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;

  }

    }



}
```

DailyLeadProcessorTest.apxc

```
@isTest

public class DailyLeadProcessorTest {

//Seconds Minutes Hours Day_of_month Month Day_of_week optional_year

  public static String CRON_EXP = '0 0 0 2 4 ? 2023';



  static testmethod void testScheduledJob(){

    List<Lead> leads = new List<Lead>();



    for(Integer i = 0; i < 200; i++){

      Lead lead = new Lead(LastName = 'Test ' + i, LeadSource = '', Company = 'Test Company ' + i,
Status = 'Open - Not Contacted');

      leads.add(lead);

    }



    insert leads;



    Test.startTest();

    // Schedule the test job

    String jobId = System.schedule('Update LeadSource to DreamForce', CRON_EXP, new
DailyLeadProcessor());



    // Stopping the test will run the job synchronously

    Test.stopTest();

  }

}
```