Asynchronous-Apex

Use Future Methods

AccountProcessor Code

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

      List<Account> accountsToUpdate=new List<Account>();
      List<Account> accounts = [Select Id,Name, (Select Id from Contacts) from Account
Where Id in :accountIds];

      For(Account acc:accounts){
         List<Contact> contactList =acc.Contacts;
         acc.Number_Of_Contacts__c=contactList.size();
         accountsToUpdate.add(acc);
      }

      update accountsToUpdate;
   }

}
```

AccountProcessorTest Code

```
@Istest
private class AccountProcessorTest {
   @IsTest
```

```
    private static void testCountContacts(){
        Account newAccount = new Account(Name='Test Account');
        insert newAccount;

        Contact newContact1 = new Contact(FirstName='John',LastName='Doe',AccountId =
newAccount.Id);
        insert newContact1;

        Contact newContact2 = new Contact(FirstName='Jane',LastName='Doe',AccountId =
newAccount.Id);
        insert newContact2;

        List<Id> accountIds =new List<Id>();
        accountIds.add(newAccount.Id);

        Test.startTest();
        AccountProcessor.countContacts(accountIds);
        Test.stopTest();
    }

}
```

Use Batch Apex

Lead Processor

```
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);
    }

}
```

Lead Processor test

```
@isTest
public class LeadProcessorTest {
```

```
    @isTest
    public static void testit(){
        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 code

```
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 Code

```
@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 code

```
global class DailyLeadProcessor implements Schedulable{
   global void execute(SchedulableContext ctx)
   {
      List<Lead> leads = [SELECT Id, LeadSource FROM Lead WHERE LeadSource = ''];
      if(leads.size() > 0){
         List<Lead> newLeads = new List<Lead>();
         for(Lead lead : leads)
         {
            lead.LeadSource = 'DreamForce';
            newLeads.add(lead);
         }
         update newLeads;
      }
   }
}
```

DailyLeadProcessortest code

```
@isTest
private class DailyLeadProcessorTest{

   public static String CRON_EXP = '0 0 0 2 6 ? 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();

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

        Test.stopTest();
    }
}
```