ASYNCHRONUS APEX

Use Future Methods:-

Apex Class->

```
public class AccountProcessor {

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

      Map<Id,Integer> account_cno = new Map<Id,Integer>();
      List<account> account_lst_all = new List<account>([select id,
(select id from contacts) from account]);
      for(account a:account_lst_all) {
         account_cno.put(a.id,a.contacts.size()); //populate the map

      }

      List<account> account_lst = new List<account>(); // list of account
that we will upsert

      for(Id accountId : accountId_lst) {
         if(account_cno.containsKey(accountId)) {
            account acc = new account();
            acc.Id = accountId;
            acc.Number_of_Contacts__c = account_cno.get(accountId);
            account_lst.add(acc);
         }

      }
      upsert account_lst;
   }

}
```

Apex Test Class->

```
@isTest
public class AccountProcessorTest {

   @isTest
   public static void testFunc() {
      account acc = new account();
      acc.name = 'MATW INC';
      insert acc;

      contact con = new contact();
      con.lastname = 'Mann1';
      con.AccountId = acc.Id;
      insert con;
      contact con1 = new contact();
      con1.lastname = 'Mann2';
      con1.AccountId = acc.Id;
      insert con1;


      List<Id> acc_list = new List<Id>();
      acc_list.add(acc.Id);
      Test.startTest();
        AccountProcessor.countContacts(acc_list);
      Test.stopTest();
      List<account> acc1 = new List<account>([select
Number_of_Contacts__c from account where id = :acc.id]);
      system.assertEquals(2,acc1[0].Number_of_Contacts__c);
   }
}
```

Use Batch Apex:-

Apex Class->

```
global class LeadProcessor implements
Database.Batchable<sObject>, Database.Stateful {

   // instance member to retain state across transactions
   global Integer recordsProcessed = 0;

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

   global void execute(Database.BatchableContext bc, List<Lead>
scope){
      // process each batch of records
      List<Lead> leads = new List<Lead>();
      for (Lead lead : scope) {

            lead.LeadSource = 'Dreamforce';
            // increment the instance member counter
            recordsProcessed = recordsProcessed + 1;

      }
      update leads;
   }

   global void finish(Database.BatchableContext bc){
      System.debug(recordsProcessed + ' records processed. Shazam!');

   }
}
```

Apex Test Class->

```
@isTest
public class LeadProcessorTest {
 @testSetup
    static void setup() {
        List<Lead> leads = new List<Lead>();
        // insert 200 leads
        for (Integer i=0;i<200;i++) {
            leads.add(new Lead(LastName='Lead '+i,
                Company='Lead', Status='Open - Not Contacted'));
        }
        insert leads;
    }

    static testmethod void test() {
        Test.startTest();
        LeadProcessor lp = new LeadProcessor();
        Id batchId = Database.executeBatch(lp, 200);
        Test.stopTest();

        // after the testing stops, assert records were updated properly
        System.assertEquals(200, [select count() from lead where
LeadSource = 'Dreamforce']);
    }
}
```

Control Processes with Queueable Apex:-

Apex Class->

```
public class AddPrimaryContact implements Queueable {
   public contact c;
   public String state;

   public AddPrimaryContact(Contact c, String state) {
      this.c = c;
      this.state = state;
   }

   public void execute(QueueableContext qc) {
      system.debug('this.c = '+this.c+' this.state = '+this.state);
      List<Account> acc_lst = new List<account>([select id, name,
BillingState from account where account.BillingState = :this.state limit
200]);
      List<contact> c_lst = new List<contact>();
      for(account a: acc_lst) {
         contact c = new contact();
         c = this.c.clone(false, false, false, false);
         c.AccountId = a.Id;
         c_lst.add(c);
      }
      insert c_lst;
   }

}
```

Apex Test Class->

```
@IsTest
public class AddPrimaryContactTest {

    @IsTest
    public static void testing() {
        List<account> acc_lst = new List<account>();
        for (Integer i=0; i<50;i++) {
            account a = new
account(name=string.valueOf(i),billingstate='NY');
            system.debug('account a = '+a);
            acc_lst.add(a);
        }
        for (Integer i=0; i<50;i++) {
            account a = new
account(name=string.valueOf(50+i),billingstate='CA');
            system.debug('account a = '+a);
            acc_lst.add(a);
        }
        insert acc_lst;
        Test.startTest();
        contact c = new contact(lastname='alex');
        AddPrimaryContact apc = new AddPrimaryContact(c,'CA');
        system.debug('apc = '+apc);
        System.enqueueJob(apc);
        Test.stopTest();
        List<contact> c_lst = new List<contact>([select id from contact]);
        Integer size = c_lst.size();
        system.assertEquals(50, size);
    }
}
```

Schedule Jobs Using The Apex Scheduler:-

Apex Class->

```
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;
        }
    }
}
```

Apex Test Class->

```
@isTest
private class DailyLeadProcessorTest{
    //Seconds Minutes Hours Day_of_month Month Day_of_week
optional_year
    public static String CRON_EXP = '0 0 0 2 6 ? 2022';
    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();
    }
}
```