```
public class demo2 {
    public  static integer add(integer x,integer y) // user want to insert a value at run time
    {
       string s='Tantul'; //assigning  a value
        system.debug('Show me name'+s);
        integer result;
        result=x+y;
        system.debug('show me result ::'+ result);
        return result;
    }
    //@testvisible
     public  static integer sub(integer x,integer y) // user want to insert a value at run time and
return integer value
    {
       string s='Tantul'; //assigning  a value
        system.debug('Show me name'+s);
        integer result;
        result=x-y;
       // system.debug('show me result ::'+ result);
       return result;
    }



}
……..
@istest
public class testclass1
{
@istest static void test_add()
{
```

```
    integer result=demo2.add(10,20);
    system.assertEquals(80,result);
}

@istest static void test_sub()
{
    integer result=demo2.sub(60,20);
    system.assertEquals(40,result);

}

}
```

……..

Task1: Create a test class for leap year

Tak2:Create a test class for Positive number

Task3:Create a test class for Childclass.

……

```
trigger scenario1 on Account (before insert)
{


    for(account ac:trigger.new)
    {
        if(ac.type==null)
        {
            ac.adderror('Account type is required');
            ac.type.adderror('Please enter account type');
        }

    }
```

```
}
@istest
public class testclass2
{


    @istest static void text_trig()
    {
        account acc= new account(name='Infosys3',annualrevenue=899999,rating='Hot');
        insert acc;


    }


}
```

……

```
trigger scenario6 on Account (before insert)
{
    list<string> str1=new list<string>();
for(account a:trigger.new)
{
    str1.add(a.name);
}
```

```
    list<account> aclist=[select name from account where name in :str1]; // fetch the match record from database



    for(account a1:trigger.new)
    {3
        for(account a2:aclist)
        {
            if(a1.name==a2.name)
            {
                a1.adderror('duplicate record exists');
            }

        }
    }

}
@istest
public class testclass3
{


    @istest static void test_ins1()
    {
```

```
        account acc1=new account(name='Infosys',type='prospect',rating='hot');
        insert acc1;

        account acc2=new account(name='Infosys',type='prospect',rating='hot');
        insert acc2;

        system.assertEquals(acc1, acc2,'Same Account Name exists Try again');

    }
}
public class testdatafactory
{

    public static list<lead> createlead(integer num)
    {
        list<lead>ls=new list<lead>();
        for(integer i=1;i<num;i++)
        {
            lead le=new lead();
            le.firstname='testdata'+i;
            le.lastname='testdata'+i;
            le.company='xerox'+i;
            ls.add(le);
```

```
        }
        insert ls;
        return ls;

    }
}
public class displayleadmethod
{

    public static list<lead> displayleadmethod()
    {
        list<lead> ls=[select firstname,lastname,company from lead limit 20];
        //it will fail because of lead limit
        return ls;
    }
}
@istest
public class testclass4
{
@istest static void test_lead()
{
```

```
    list<lead> ls= new list<lead>();
    ls=testdatafactory.createlead(50);
    system.assertequals(ls,displayleadmethod.displayleadmethod());
}

}
```

Example on Batch Testing

```
global class batchexample3 implements database.batchable<sobject>
{

    global database.QueryLocator start(database.BatchableContext bcq)
    {

        //system.debug('----query of account'+query);
        return database.getQueryLocator('select leadsource,rating from lead');
    }

    global void execute(database.BatchableContext bcq,list<lead> scope)
    {
            //list<lead> cust=new list<lead>();
        system.debug('---scope'+scope.size());
        for(lead a:scope)
        {
            IF(a.leadsource=='Web')
            {
            a.rating='warm';
            }
```

```
            //cust.add(a);
        }
        update scope;

    }
    global void finish(database.BatchableContext bcq)
    {
        //nothing to write
    }


}
@istest
public class testclass4 {
    @istest
    public static void testm()
    {
        lead l=new lead();
        l.firstname='Salesforce';
        l.lastname='code';
        l.company='almamalearn';
        l.LeadSource='Web';
        insert l;
        test.startTest();
        batchexample3 ba= new batchexample3();
        id jobid=database.executeBatch(ba,5);
        test.stopTest();
        lead leads=[select rating from lead where id=:l.id];
        system.assertEquals('Warm',leads.rating);
```

```
  }


}
```

……

```
global class AccountUpdate implements
Database.Batchable {


     global Database.QueryLocator
start(Database.BatchableContext BC){
          return Database.getQueryLocator('Select
Id, Name From Account');
     }
     global void execute(Database.BatchableContext
BC, List  scope){
          for(Account a: scope){
               a.Name += ' Updated';
          }
          update scope;
     }
     global void finish(Database.BatchableContext
BC){}
```

Testclass

```
@isTest
private class accListountUpdate {
```

```
static testmethod void test() {

// Create test accounts to be updated
// by the batch job.

Account[] accList = new List();
for (Integer i=0;i<200;i++) {
Account m = new Account(Name = 'Account ' + i);
accList.add(m);
}
insert accList;

Test.startTest();
AccountUpdate c = new AccountUpdate();
Database.executeBatch(c);
Test.stopTest();

// Verify accounts updated
Account[] accUpdatedList = [SELECT Id,Name FROM
Account];
System.assert(accUpdatedList[0].Name.Contains('Upda
ted'));
}
}
```

In the code below, we are testing the above batch class by creating 200 test accounts to make sure that Account Name gets updated by batch apex correctly.

As best practice, you must execute the batch class between Test.StartTest() and Test.StopTest().

We can also load data from static resource to create Accounts using the following syntax.

List ls = Test.loadData(Account.sObjectType, 'myResource');

You must store the static resource (e.g. account.csv) under Salesforce static resources. The supported types of files are text/csv, application/vnd.ms-excel, application/octet-stream, text/plain.

.....