1.

trigger updateType on Account (before insert,before update) {

for(Account ac:Trigger.new){

if(ac.Type=='Prospect')

ac.Type\_\_c='Active';

else

ac.Type\_\_c='Inactive';

}

isTest private class TestUpdateType {

@isTest static void test1(){

Account ac=new Account();

ac.Name='saif test';

ac.type='Prospect';

insert ac;

}

}

@isTest private class TestUpdateType {

@isTest static void test1(){

List<Account> acl=new List<Account>();

Account ac=new Account();

ac.Name='saif test';

ac.type='Prospect';

acl.add(ac);

Account ac1=new Account();

ac1.Name='saif test';

acl.add(ac1);

insert acl;

}

}

2.

if(ac.Type\_\_c=='Active' && ac.Active\_\_c!='Yes')

ac.Active\_\_c.addError('Active field should be yes');

else if(ac.Type\_\_c=='Inactive' && ac.Active\_\_c!='No')

ac.Active\_\_c.addError('Active field should be No');

}

@isTest private class TestUpdateType {

@isTest static void test1(){

List<Account> acl=new List<Account>();

Account ac=new Account();

ac.Name='saif test';

ac.type='Prospect';

acl.add(ac);

Account ac1=new Account();

ac1.Name='saif test';

acl.add(ac1);

try{

insert acl;

}

catch(Exception e){

system.assertEquals(e.getMessage(), e.getMessage());

}

}

}

3.

trigger createContact on Account (after insert, after update){

// we have to insert conatcts based on the count on Account's No of contact field, so we need a list of contacts

List<contact> contlist=new List<contact>();

for(Account a:Trigger.new){

//contact number

integer i=0;

if(a.No\_of\_Contact\_\_c!=null){

// based on no of contact field we will create contact

if((trigger.isUpdate && a.No\_of\_Contact\_\_c != Trigger.oldMap.get(a.id).No\_of\_Contact\_\_c) || trigger.isInsert){

/\* here if account record is being inserted then we will need to create contact or

we are updating no of contact field on the account \*/

while(i<a.No\_of\_Contact\_\_c){

contact c=new contact();

c.LastName='Cont Of '+a.Name+'-'+i;

c.AccountId=a.id;

contlist.add(c);

i++;

}

}

}

}

if(contlist.size()>0)

insert contlist;

}

@isTest private class testCreateCont {

@isTest static void test1(){

List<Account> acl=new List<Account>();

Account ac=new Account();

ac.Name='saif test';

ac.No\_of\_Contact\_\_c=2;

acl.add(ac);

Account ac1=new Account();

ac1.name='yeye';

acl.add(ac1);

insert acl;

}

}

4.

public class TempratureClass {

Public Double fahrenToCelsius( Double f){

return ((f-32)/1.8);

}

Public Double fahrenToCelsiusWrongMethod( Double f){

return ((f-30)/1.8);

}

}

@isTest

public class TestTempratureClass {

@isTest static void testmthd1(){

TempratureClass tc= new TempratureClass();

Double c = tc.fahrenToCelsius(212);

System.assertEquals(100, c, 'Answer is correct');

}

@isTest static void testmthd2(){

TempratureClass tc= new TempratureClass();

Double c1 = tc.fahrenToCelsiusWrongMethod(212);

System.assertEquals(100, c1, 'Answer is correct');

}

}

5.

@isTest private class testCreateCont {

@testSetup static void testFill(){

// record creation

Account acc= new Account(Name=’saif’);

Insert acc;

}

@isTest static void test1(){

acc.Name= ‘saif changed’;

update acc;

}

@isTest static void test2(){

System.debug(acc.Name);

acc.Name= ‘saif changed 2’;

update acc;

}

}

6.

public class Mycontroller{

public Integer addMethod(Ineteger a, Integer b){

return a+b;

}

public Decimal FarenhietMethod(Decimal cel){

return ((9 \* cel )/ 5 +32);

}

}

@isTest private class MycontrollerTest{

@isTest static void test1(){

Mycontroller mcnt = new Mycontroller();

Integer sum = mcnt.addMethod(5,6);

Integer far = mcnt.FarenhietMethod(-40);

}

}

@isTest private class MycontrollerTest{

@isTest static void test1(){

Mycontroller mcnt = new Mycontroller();

Integer sum =mcnt.addMethod(5,6);

System.assertEquals(11, sum);

Integer far = mcnt.FarenhietMethod(-40);

System.assertEquals(-40, far);

}

}

7.

<apex:page standardController="Account" extensions="accControllerExtension">

<!----- some line of code-->

<!------>

</apex:page>

public class accControllerExtension{

public Account acc;

public accControllerExtension(ApexPages.StandardController controller) {

acc = (Account)controller.getrecord();

string accId= ApexPages.currentPage().getParameters().get('Id');

}

// some other method

public void methd1(){

//… ……..

}

}

@isTest

public class accControllerExtensionTest{

@isTest

public static void testmethd1(){

Account acc= new Account(name='saif');

insert acc;

// suppose page name is accCheck

PageReference pageRef = Page.accCheck;

Test.setCurrentPage(pageRef);

// parameter passing for page

pageRef.getParameters().put('Id', String.valueOf(acc.Id));

// standard controller

ApexPages.StandardController sc = new ApexPages.StandardController(acc);

// calling extension with standardcontroller

accControllerExtension accExt = new accControllerExtension(sc);

// calling method of extension

accExt.methd1();

}

}

8.

public class CalloutapiMap {

public static void chckMap(){

Http h = new Http();

HttpRequest req = new HttpRequest ();

String url ='https://maps.googleapis.com/maps/api/geocode/json?key=AIzaSyDCJfSJhXuKJlffbFfB57yOO\_iQK4kAmio&latlng=28.5810215,77.3152004&sensor=true';

req.setEndpoint(url);

req.setMethod('GET');

HttpResponse res = h.send(req);

if(res.getStatusCode()==200){

string str= res.getBody().split('"formatted\_address" : "')[1].split('",')[0];

system.debug('This is the address'+str);

}

}

}

#testclass

global class MockApiResponseAPI implements HttpCalloutMock {

protected integer statuscode;

protected String Body;

public DemoApiMockHttpResponseGenerator(integer code,String Body){

this.statuscode=code;

this.Body=body;

}

global HTTPResponse respond(HTTPRequest req) {

HttpResponse res = new HttpResponse();

res.setHeader('Content-Type', 'application/json');

res.setBody(Body);

res.setStatusCode(statuscode);

return res;

}

}

#mock

@isTest

public class CalloutapiMapTest{

public static testmethod void testCallout() {

string responsebdy='{"formatted\_address" :"vyapar market sector2 noida",jjj”}';

MockApiResponseAPI fakeResponse = new MockApiResponseAPI(200,responsebdy);

//using constructor we can decide what body/status code should come in fake response

Test.setMock(HttpCalloutMock.class, fakeResponse);

CalloutapiMap.chckMap();

}

}