Apex Specialist -SuperBadge

Apex Trigger:

1. Get Started With Apex Trigger

```
trigger AccountAddressTrigger on Account (before insert, before update)
{
  if(Trigger.isInsert)
  {
        for(Account a : Trigger.new)
        {
             If (a.Match_Billing_Address__c == true && a.BillingPostalCode!=Null)
        {
                    a.ShippingPostalCode = a.BillingPostalCode;
             }
         }
    }
}
```

2. Bulk Apex Triggers

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {

    List<Task> taskList = new List<Task>();

    for(Opportunity opp : Trigger.new) {

        if(Trigger.isInsert) {
            if(Opp.StageName == 'Closed Won') {
                taskList.add(new Task(Subject = 'Follow Up Test Task', WhatId =
opp.Id));
            }
        }

        if(Trigger.isUpdate)
        {
            if(Opp.StageName == 'Closed Won' && Opp.StageName !=
Trigger.oldMap.get(opp.Id).StageName)
            {
```

```
                taskList.add(new Task(Subject = 'Follow Up Test Task', WhatId =
    opp.Id));
            }
        }
    }

    if(taskList.size()>0) {
        insert taskList;
    }
}
```

Apex Testing:

1. Get Started with Apex Unit Tests

test class

```
@isTest
public class TestVerifyDate {
    @isTest static void testCheckDates()
    {
        Date D1=system.today();
        Date D2=system.today()+20;
        date D=VerifyDate.CheckDates(D1,D2);

    }
     @isTest static void testCheckDates1()
    {
        Date D1=system.today();
        Date D2=system.today()-20;
        date D=VerifyDate.CheckDates(D1,D2);

    }
     @isTest static void testCheckDates2()
    {
        Date D1=system.today();
        Date D2=system.today()+60;
        date D=VerifyDate.CheckDates(D1,D2);

    }

```

2. Test Apex Triggers

Class

```
trigger RestrictContactByName on Contact (before insert, before update) {
    //check contacts prior to insert or update for invalid data
    For (Contact c : Trigger.New) {
        if(c.LastName == 'INVALIDNAME') {   //invalidname is invalid
            c.AddError('The Last Name "'+c.LastName+'" is not allowed for DML');
        }
    }
}
```

Test Class

```
@isTest
private class TestRestrictContactByName {

    static testMethod void  metodoTest()
    {

        List<Contact> listContact= new List<Contact>();
        Contact c1 = new Contact(FirstName='Francesco', LastName='Riggio' , email='Test@test.com');
        Contact c2 = new Contact(FirstName='Francesco1', LastName = 'INVALIDNAME',email='Test@test.com');
        listContact.add(c1);
        listContact.add(c2);

        Test.startTest();
            try
            {
                insert listContact;
            }
            catch(Exception ee)
            {
            }

        Test.stopTest();

    }

}
```

3. Create Test Data for Apex Tests

```
public class RandomContactFactory{

    public static List<Contact> generateRandomContacts(integer n,string LastName){
    integer n1=n;
    List<contact> c1 = new list<contact>();
    list<contact> c2 =new list<contact>();
     c1 = [select FirstName from Contact Limit : n1];
     integer i=0;
     for(contact cnew : c1){
     contact cnew1 = new contact();
     cnew1.firstname = cnew.firstname + i;

     c2.add(cnew1);
     i++;
     }
    return c2;

    }
}
```

Asynchronous Apex:

1. Use Future Methods

class

```
public class AccountProcessor
{
  @future
  public static void countContacts(Set<id> setId)
  {
      List<Account> lstAccount = [select id,Number_of_Contacts__c , (select id from
contacts ) from account where id in :setId ];
      for( Account acc : lstAccount )
      {
          List<Contact> lstCont = acc.contacts ;

          acc.Number_of_Contacts__c = lstCont.size();
      }
```

```
13      update lstAccount;
14   }
15 }
```

Test Class

```
1  @IsTest
2  public class AccountProcessorTest {
3      public static testmethod void TestAccountProcessorTest()
4      {
5          Account a = new Account();
6          a.Name = 'Test Account';
7          Insert a;
8
9          Contact cont = New Contact();
10         cont.FirstName ='Bob';
11         cont.LastName ='Masters';
12         cont.AccountId = a.Id;
13         Insert cont;
14
15         set<Id> setAccId = new Set<ID>();
16         setAccId.add(a.id);
17
18         Test.startTest();
19             AccountProcessor.countContacts(setAccId);
20         Test.stopTest();
21
22         Account ACC = [select Number_of_Contacts__c from Account where id = :a.id
   LIMIT 1];
23         System.assertEquals ( Integer.valueOf(ACC.Number_of_Contacts__c) ,1);
24   }
25
26 }
```

2. Use Batch Apex

Class

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

```
    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!');

    }
}
```

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']);
    }
}
```

3. Control Processes with Queueable Apex

Class

```
public class AddPrimaryContact implements Queueable
{
    private Contact c;
    private String state;
    public  AddPrimaryContact(Contact c, String state)
    {
        this.c = c;
        this.state = state;
    }
    public void execute(QueueableContext context)
    {
         List<Account> ListAccount = [SELECT ID, Name ,(Select
id,FirstName,LastName from contacts ) FROM ACCOUNT WHERE BillingState = :state
LIMIT 200];
         List<Contact> lstContact = new List<Contact>();
         for (Account acc:ListAccount)
         {
                 Contact cont = c.clone(false,false,false,false);
                 cont.AccountId =  acc.id;
                 lstContact.add( cont );
         }

         if(lstContact.size() >0 )
         {
             insert lstContact;
         }
```

```
25
26     }
27
28 }
```

Test Class

```
1  @isTest
2  public class AddPrimaryContactTest
3  {
4       @isTest static void TestList()
5       {
6           List<Account> Teste = new List <Account>();
7           for(Integer i=0;i<50;i++)
8           {
9               Teste.add(new Account(BillingState = 'CA', name = 'Test'+i));
10          }
11          for(Integer j=0;j<50;j++)
12          {
13              Teste.add(new Account(BillingState = 'NY', name = 'Test'+j));
14          }
15          insert Teste;
16
17          Contact co = new Contact();
18          co.FirstName='demo';
19          co.LastName ='demo';
20          insert co;
21          String state = 'CA';
22
23           AddPrimaryContact apc = new AddPrimaryContact(co, state);
24           Test.startTest();
25             System.enqueueJob(apc);
26           Test.stopTest();
27       }
28  }
```

4. Schedule Jobs Using the Apex Scheduler

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;
        }
    }
}
```

test class

```
@isTest
private class DailyLeadProcessorTest {
  static testMethod void testDailyLeadProcessor() {
        String CRON_EXP = '0 0 1 * * ?';
        List<Lead> lList = new List<Lead>();
      for (Integer i = 0; i < 200; i++) {
              lList.add(new Lead(LastName='Dreamforce'+i, Company='Test1 Inc.',
Status='Open - Not Contacted'));
        }
        insert lList;

        Test.startTest();
        String jobId = System.schedule('DailyLeadProcessor', CRON_EXP, new
DailyLeadProcessor());
  }
}
```

Apex Integration Services:

1. Apex REST Call outs

Apex Class

```
1  public class AnimalLocator
2  {
3
4    public static String getAnimalNameById(Integer id)
5     {
6          Http http = new Http();
7          HttpRequest request = new HttpRequest();
8          request.setEndpoint('https://th-apex-http-

9          request.setMethod('GET');
10         HttpResponse response = http.send(request);
11           String strResp = '';
12            system.debug('******response '+response.getStatusCode());
13            system.debug('******response '+response.getBody());
14
15         if (response.getStatusCode() == 200)
16         {
17
18            Map<String, Object> results = (Map<String, Object>)
   JSON.deserializeUntyped(response.getBody());
19
20            Map<string,object> animals = (map<string,object>) results.get('animal');
21             System.debug('Received the following animals:' + animals );
22             strResp = string.valueof(animals.get('name'));
23             System.debug('strResp >>>>>>' + strResp );
24         }
25         return strResp ;
26    }
27
28 }
```

Apex Test Class

```
1  @isTest
2  private class AnimalLocatorTest{
3      @isTest static  void AnimalLocatorMock1() {
```

```
        Test.SetMock(HttpCallOutMock.class, new AnimalLocatorMock());
        string result=AnimalLocator.getAnimalNameById(3);
        string expectedResult='chicken';
        System.assertEquals(result, expectedResult);
    }
}
```

Apex Mock Test Class

```
@isTest
global class AnimalLocatorMock implements HttpCalloutMock {
    global HTTPResponse respond(HTTPRequest request) {
         HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');
        response.setBody('{"animal":{"id":1,"name":"chicken","eats":"chicken

        response.setStatusCode(200);
        return response;
    }
}
```

2. Apex SOAP Call outs

Apex service

```
public class ParkService {
    public class byCountryResponse {
        public String[] return_x;
        private String[] return_x_type_info = new
   String[]{'return','http://parks.services/',null,'0','-1','false'};
        private String[] apex_schema_type_info = new
   String[]{'http://parks.services/','false','false'};
        private String[] field_order_type_info = new String[]{'return_x'};
    }
    public class byCountry {
        public String arg0;
        private String[] arg0_type_info = new
   String[]{'arg0','http://parks.services/',null,'0','1','false'};
        private String[] apex_schema_type_info = new
   String[]{'http://parks.services/','false','false'};
```

```
        private String[] field_order_type_info = new String[]{'arg0'};
    }
    public class ParksImplPort {
        public String endpoint_x = 'https://th-apex-soap-

        public Map<String,String> inputHttpHeaders_x;
        public Map<String,String> outputHttpHeaders_x;
        public String clientCertName_x;
        public String clientCert_x;
        public String clientCertPasswd_x;
        public Integer timeout_x;
        private String[] ns_map_type_info = new String[]{'http://parks.services/',
  'ParkService'};
        public String[] byCountry(String arg0) {
            ParkService.byCountry request_x = new ParkService.byCountry();
            request_x.arg0 = arg0;
            ParkService.byCountryResponse response_x;
            Map<String, ParkService.byCountryResponse> response_map_x = new
  Map<String, ParkService.byCountryResponse>();
            response_map_x.put('response_x', response_x);
            WebServiceCallout.invoke(
              this,
              request_x,
              response_map_x,
              new String[]{endpoint_x,
              '',
              'http://parks.services/',
              'byCountry',
              'http://parks.services/',
              'byCountryResponse',
              'ParkService.byCountryResponse'}
            );
            response_x = response_map_x.get('response_x');
            return response_x.return_x;
        }
    }
}
```

Apex Class

```
public class ParkLocator {
    public static String[] country(String country){
```

```
        ParkService.ParksImplPort parks = new ParkService.ParksImplPort();
        String[] parksname = parks.byCountry(country);
        return parksname;
    }
}
```

Apex Test Class

```
@isTest
private class ParkLocatorTest{
    @isTest
    static void testParkLocator() {
        Test.setMock(WebServiceMock.class, new ParkServiceMock());
        String[] arrayOfParks = ParkLocator.country('India');

        System.assertEquals('Park1', arrayOfParks[0]);
    }
}
```

Apex Mock Test Class

```
@isTest
global class ParkServiceMock implements WebServiceMock {
    global void doInvoke(
           Object stub,
           Object request,
           Map<String, Object> response,
           String endpoint,
           String soapAction,
           String requestName,
           String responseNS,
           String responseName,
           String responseType) {
        ParkService.byCountryResponse response_x = new
ParkService.byCountryResponse();
        List<String> lstOfDummyParks = new List<String> {'Park1','Park2','Park3'};
        response_x.return_x = lstOfDummyParks;

        response.put('response_x', response_x);
    }
}

```

3. Apex Web Services

class

```
@RestResource(urlMapping='/Accounts/*/contacts')
global with sharing class AccountManager{
    @HttpGet
    global static Account getAccount(){
        RestRequest req = RestContext.request;
        String accId = req.requestURI.substringBetween('Accounts/', '/contacts');
        Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)
                       FROM Account WHERE Id = :accId];

        return acc;
    }
}
```

Test Class

```
@IsTest
private class AccountManagerTest{
    @isTest static void testAccountManager(){
        Id recordId = getTestAccountId();
        // Set up a test request
        RestRequest request = new RestRequest();
        request.requestUri =
            'https://ap5.salesforce.com/services/apexrest/Accounts/'+ recordId
+'/contacts';
        request.httpMethod = 'GET';
        RestContext.request = request;

        // Call the method to test
        Account  acc = AccountManager.getAccount();

        // Verify results
        System.assert(acc != null);
    }

```

```
    private static Id getTestAccountId(){
        Account acc = new Account(Name = 'TestAcc2');
        Insert acc;

        Contact con = new Contact(LastName = 'TestCont2', AccountId = acc.Id);
        Insert con;

        return acc.Id;
    }
}
```

Apex Specialist:

1. Automate record creation

Apex Trigger

```
trigger MaintenanceRequest on Case (before update, after update) {

    Map<Id,Case> validCaseMap = new Map<Id,Case>();

    if(Trigger.isUpdate  && Trigger.isAfter){
        for(Case caseHere: Trigger.new){
            if (caseHere.IsClosed && (caseHere.Type.equals('Repair') ||
caseHere.Type.equals('Routine Maintenance'))){
                validCaseMap.put(caseHere.Id, caseHere);
            }
        }

        if(!validCaseMap.values().isEmpty()){
 MaintenanceRequestHelper.createNewRequest(validCaseMap);
        }
    }

}
```

Apex class

```
public class MaintenanceRequestHelper {

    public static void createNewRequest(Map<Id, Case> validCaseMap){
```

```
        List<Case> newCases = new List<Case>();
        Map<Id, Integer> productMaintenanceCycleMap = new Map<Id, Integer>();
        Map<Id, Integer> workPartMaintenanceCycleMap = new Map<Id, Integer>();

        for (Product2 productHere : [select Id, Maintenance_Cycle__c from Product2])
{
            if (productHere.Maintenance_Cycle__c != null) {
                productMaintenanceCycleMap.put(productHere.Id,
Integer.valueOf(productHere.Maintenance_Cycle__c));
            }
        }

        for (Work_Part__c workPart : [select Id, Equipment__c,
Maintenance_Request__c from Work_Part__c where Maintenance_Request__c in
:validCaseMap.keySet()]) {
            if (workPart.Equipment__c != null) {

if(!workPartMaintenanceCycleMap.containsKey(workPart.Maintenance_Request__c)){

workPartMaintenanceCycleMap.put(workPart.Maintenance_Request__c,
productMaintenanceCycleMap.get(workPart.Equipment__c));
                }
                else if(productMaintenanceCycleMap.get(workPart.Equipment__c) <
workPartMaintenanceCycleMap.get(workPart.Maintenance_Request__c)){

workPartMaintenanceCycleMap.put(workPart.Maintenance_Request__c,
productMaintenanceCycleMap.get(workPart.Equipment__c));
                }
            }
        }

        for(Case caseHere: validCaseMap.values()){
            Case newCase = new Case();
            newCase.Vehicle__c = caseHere.Vehicle__c;
            newCase.Equipment__c = caseHere.Equipment__c;
            newCase.Type = 'Routine Maintenance';
            newCase.Subject = String.isBlank(caseHere.Subject) ? 'Routine


            newCase.Date_Reported__c = Date.today();
            newCase.Date_Due__c =
workPartMaintenanceCycleMap.containsKey(caseHere.Product__c) ?
Date.today().addDays(workPartMaintenanceCycleMap.get(caseHere.Product__c)) :
Date.today();
```

```
            newCase.Status = 'New';
            newCase.Product__c = caseHere.Product__c;
            newCase.AccountId = caseHere.AccountId;
            newCase.ContactId = caseHere.ContactId;
            newCase.AssetId = caseHere.AssetId;
            newCase.Origin = caseHere.Origin;
            newCase.Reason = caseHere.Reason;

            newCases.add(newCase);
        }

        if(newCases.size() > 0){
            insert newCases;
        }
    }

}
```

2. Synchronize Salesforce data with an external system

Apex Class

```
public with sharing class WarehouseCalloutService {

    private static final String WAREHOUSE_URL = 'https://th-superbadge-apex.herokuapp.com/equipment';

    @future(callout=true)
    public static void runWarehouseEquipmentSync(){
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);

        if (response.getStatusCode() == 200) {
            List<Object> results = (List<Object>) JSON.deserializeUntyped(response.getBody());
            List<Product2> equipmentList = new List<Product2>();

            for (Object record: results) {
                Map<String, Object> recordMap = (Map<String, Object>)record;
                Product2 equipment = new Product2();
```

```
                equipment.Name = (String)recordMap.get('name');
                equipment.Cost__c = (Decimal)recordMap.get('cost');
                equipment.ProductCode = (String)recordMap.get('_id');
                equipment.Current_Inventory__c = (Integer)recordMap.get('quantity');
                equipment.Maintenance_Cycle__c = (Integer)recordMap.get('maintenanceperiod');
                equipment.Replacement_Part__c = (Boolean)recordMap.get('replacement');
                equipment.Lifespan_Months__c = (Integer)recordMap.get('lifespan');
                equipment.Warehouse_SKU__c = (String)recordMap.get('sku');

                equipmentList.add(equipment);
            }

            if(equipmentList.size() > 0){
                upsert equipmentList;
            }
        }

    }
}
```

Anonymous Window

```
WarehouseCalloutService.runWarehouseEquipmentSync();
```

3. Schedule synchronization

Class

```
global with sharing class WarehouseSyncSchedule implements Schedulable{
    global void execute(SchedulableContext ctx){
        System.enqueueJob(new WarehouseCalloutService());
    }
}
```

Anonymous Window

```
System.schedule('WarehouseSyncScheduleTest', '0 0 1 * * ?', new
WarehouseSyncSchedule());
```

4. Test automation logic

Helper Test Class

```
@istest
public with sharing class MaintenanceRequestHelperTest {

    private static final string STATUS_NEW = 'New';
    private static final string WORKING = 'Working';
    private static final string CLOSED = 'Closed';
    private static final string REPAIR = 'Repair';
    private static final string REQUEST_ORIGIN = 'Web';
    private static final string REQUEST_TYPE = 'Routine Maintenance';
    private static final string REQUEST_SUBJECT = 'Testing subject';

    PRIVATE STATIC Vehicle__c createVehicle(){
        Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
        return Vehicle;
    }

    PRIVATE STATIC Product2 createEq(){
        product2 equipment = new product2(name = 'SuperEquipment',
                                          lifespan_months__C = 10,
                                          maintenance_cycle__C = 10,
                                          replacement_part__c = true);
        return equipment;
    }

    PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id equipmentId){
        case cs = new case(Type=REPAIR,
                           Status=STATUS_NEW,
                           Origin=REQUEST_ORIGIN,
                           Subject=REQUEST_SUBJECT,
                           Equipment__c=equipmentId,
                           Vehicle__c=vehicleId);
        return cs;
    }

    PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id equipmentId,id
```

```
requestId){
        Equipment_Maintenance_Item__c wp = new
Equipment_Maintenance_Item__c(Equipment__c = equipmentId,

Maintenance_Request__c = requestId);
        return wp;
    }


    @istest
    private static void testMaintenanceRequestPositive(){
        Vehicle__c vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        Product2 equipment = createEq();
        insert equipment;
        id equipmentId = equipment.Id;

        case somethingToUpdate = createMaintenanceRequest(vehicleId,equipmentId);
        insert somethingToUpdate;

        Equipment_Maintenance_Item__c workP =
createWorkPart(equipmentId,somethingToUpdate.id);
        insert workP;

        test.startTest();
        somethingToUpdate.status = CLOSED;
        update somethingToUpdate;
        test.stopTest();

        Case newReq = [Select id, subject, type, Equipment__c, Date_Reported__c,
Vehicle__c, Date_Due__c
                       from case
                       where status =:STATUS_NEW];

        Equipment_Maintenance_Item__c workPart = [select id
                                                  from Equipment_Maintenance_Item__c
                                                  where Maintenance_Request__c
=:newReq.Id];

        system.assert(workPart != null);
        system.assert(newReq.Subject != null);
```

```
        system.assertEquals(newReq.Type, REQUEST_TYPE);
        SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
        SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
        SYSTEM.assertEquals(newReq.Date_Reported__c, system.today());
    }

    @istest
    private static void testMaintenanceRequestNegative(){
        Vehicle__C vehicle = createVehicle();
        insert vehicle;
        id vehicleId = vehicle.Id;

        product2 equipment = createEq();
        insert equipment;
        id equipmentId = equipment.Id;

        case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);
        insert emptyReq;

        Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId,
  emptyReq.Id);
        insert workP;

        test.startTest();
        emptyReq.Status = WORKING;
        update emptyReq;
        test.stopTest();

         list<case> allRequest = [select id
                                  from case];

         Equipment_Maintenance_Item__c workPart = [select id from
  Equipment_Maintenance_Item__c where Maintenance_Request__c = :emptyReq.Id];

         system.assert(workPart != null);
         system.assert(allRequest.size() == 1);
     }

     @istest
     private static void testMaintenanceRequestBulk(){
         list<Vehicle__C> vehicleList = new list<Vehicle__C>();
         list<Product2> equipmentList = new list<Product2>();
         list<Equipment_Maintenance_Item__c> workPartList = new
```

```
   list<Equipment_Maintenance_Item__c>();
114         list<case> requestList = new list<case>();
115         list<id> oldRequestIds = new list<id>();
116 
117         for(integer i = 0; i < 300; i++){
118            vehicleList.add(createVehicle());
119             equipmentList.add(createEq());
120         }
121         insert vehicleList;
122         insert equipmentList;
123 
124         for(integer i = 0; i < 300; i++){
125             requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
   equipmentList.get(i).id));
126         }
127         insert requestList;
128 
129         for(integer i = 0; i < 300; i++){
130             workPartList.add(createWorkPart(equipmentList.get(i).id,
   requestList.get(i).id));
131         }
132         insert workPartList;
133 
134         test.startTest();
135         for(case req : requestList){
136             req.Status = CLOSED;
137             oldRequestIds.add(req.Id);
138         }
139         update requestList;
140         test.stopTest();
141 
142         list<case> allRequests = [select id
143                                   from case
144                                   where status =: STATUS_NEW];
145 
146         list<Equipment_Maintenance_Item__c> workParts = [select id
147                                                          from
   Equipment_Maintenance_Item__c
148                                                          where
   Maintenance_Request__c in: oldRequestIds];
149  system.assert(allRequests.size() == 300);
150     }
151 }
```

test Class

```
public with sharing class MaintenanceRequestHelper {
    public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case> nonUpdCaseMap) {
        Set<Id> validIds = new Set<Id>();


        For (Case c : updWorkOrders){
            if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
                if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
                    validIds.add(c.Id);


                }
            }
        }

        if (!validIds.isEmpty()){
            List<Case> newCases = new List<Case>();
            Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c, Equipment__r.Maintenance_Cycle__c,(SELECT Id,Equipment__c,Quantity__c FROM Equipment_Maintenance_Items__r)
                                                          FROM Case WHERE Id IN :validIds]);
            Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
            AggregateResult[] results = [SELECT Maintenance_Request__c, MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM Equipment_Maintenance_Item__c WHERE Maintenance_Request__c IN :ValidIds GROUP BY Maintenance_Request__c];

        for (AggregateResult ar : results){
            maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
        }

            for(Case cc : closedCasesM.values()){
                Case nc = new Case (
                    ParentId = cc.Id,
                Status = 'New',
                    Subject = 'Routine Maintenance',
```

```
                    Type = 'Routine Maintenance',
                    Vehicle__c = cc.Vehicle__c,
                    Equipment__c =cc.Equipment__c,
                    Origin = 'Web',
                    Date_Reported__c = Date.Today()

                );

                If (maintenanceCycles.containskey(cc.Id)){
                    nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
                }

                newCases.add(nc);
             }

            insert newCases;

            List<Equipment_Maintenance_Item__c> clonedWPs = new List<Equipment_Maintenance_Item__c>();
            for (Case nc : newCases){
                for (Equipment_Maintenance_Item__c wp : closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
                    Equipment_Maintenance_Item__c wpClone = wp.clone();
                    wpClone.Maintenance_Request__c = nc.Id;
                    ClonedWPs.add(wpClone);

                }
             }
             insert ClonedWPs;
        }
    }
}
```

Apex Trigger

```
trigger MaintenanceRequest on Case (before update, after update) {
    if(Trigger.isUpdate && Trigger.isAfter){
        MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);
    }
}
```

5. Test call out logic

Apex Class

```
public with sharing class WarehouseCalloutService {

private static final String WAREHOUSE_URL = 'https://th-superbadge-
        apex.herokupp.com/equipment';
    public static void runWarehouseEquipmentSync(){

        Http http = new Http();
        HttpRequest request = new HttpRequest();

        request.setEndpoint(WAREHOUSE_URL);
        request.setMethod('GET');
        HttpResponse response = http.send(request);


        List<Product2> warehouseEq = new List<Product2>();

        if (response.getStatusCode() == 200){
            List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
            System.debug(response.getBody());

            for (Object eq : jsonResponse){
                Map<String,Object> mapJson = (Map<String,Object>)eq;
                Product2 myEq = new Product2();
                myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
                myEq.Name = (String) mapJson.get('name');
                myEq.Maintenance_Cycle__c = (Integer)
mapJson.get('maintenanceperiod');
                myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
                myEq.Cost__c = (Decimal) mapJson.get('lifespan');
                myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
                myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
                warehouseEq.add(myEq);
            }

            if (warehouseEq.size() > 0){
                upsert warehouseEq;
                System.debug('Your equipment was synced with the warehouse one');
```

```
                System.debug(warehouseEq);
            }

        }
    }
}
```

Test Class

```
@isTest

private class WarehouseCalloutServiceTest {
    @isTest
    static void testWareHouseCallout(){
        Test.startTest();
        // implement mock callout test here
        Test.setMock(HTTPCalloutMock.class, new WarehouseCalloutServiceMock());
        WarehouseCalloutService.runWarehouseEquipmentSync();
        Test.stopTest();
        System.assertEquals(1, [SELECT count() FROM Product2]);
    }
}
```

Mock Test Class

```
@isTest
global class WarehouseCalloutServiceMock implements HttpCalloutMock {
  global static HttpResponse respond(HttpRequest request){
       System.assertEquals('https://th-superbadge-apex.herokuapp.com/equipment',
request.getEndpoint());
        System.assertEquals('GET', request.getMethod());
        HttpResponse response = new HttpResponse();
        response.setHeader('Content-Type', 'application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity"
```

```
        response.setStatusCode(200);
        return response;
    }
}
```

6. Test scheduling logic

Apex Class

```
global class WarehouseSyncSchedule implements Schedulable {
    global void execute(SchedulableContext ctx) {

        WarehouseCalloutService.runWarehouseEquipmentSync();
    }
}
```

test class

```
@isTest
public class WarehouseSyncScheduleTest {

    @isTest static void WarehousescheduleTest(){
        String scheduleTime = '00 00 01 * * ?';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class, new WarehouseCalloutServiceMock());
        String jobID=System.schedule('Warehouse Time To Schedule to Test',
scheduleTime, new WarehouseSyncSchedule());
        Test.stopTest();

        CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];
        System.assertEquals(jobID, a.Id,'Schedule ');


    }
}

```

Process Automation Specialist - SuperBadge

Formula and Validation:

1. Use Formula Field

formula:

```
Day Remaining (number) = EndDate - Today()
```

2. Create Validation Rules

error condition formula

```
AND (
NOT(ISBLANK( AccountId )),
MailingPostalCode <> Account.ShippingPostalCode
)
```

Process Automation Specialist:

1. Automate Leads

Error Condition Formula:

```
OR(AND(LEN(State) > 2,
NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:TA:KS:KY:LA:ME:MD:MA:MI:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI:WY"
```

```
3  ,State))),NOT(OR(Country="US",Country="USA",Country="USA",Country="United States",
4  ISBLANK(Country))))
```

2. Automate Account

Formula 1 :

Deal win percent (Percent) =

```
1  Number_of_won_deals__c /Number_of_deals__c
```

Formula 2 :

Call for Service (Text) =

```
1  IF(DATE(YEAR(Last_won_deal_date__c)+2,MONTH(Last_won_deal_date__c),
2  DAY(Last_won_deal_date__c))<=TODAY(),"Yes","No")
```

Error Condition Formula 1 :

```
1  OR(AND(LEN(BillingState) > 2,
   NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:ME:MD:MA
2  MI:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA:WA:WV:WI
   :WY",
3  BillingState ))),AND(LEN(ShippingState) > 2,
4  NOT(CONTAINS("AL:AK:AZ:AR:CA:CO:CT:DE:DC:FL:GA:HI:ID:IL:IN:IA:KS:KY:LA:
5  ME:MD:MA:MI:MN:MS:MO:MT:NE:NV:NH:NJ:NM:NY:NC:ND:OH:OK:OR:PA:RI:SC:SD:TN:TX:UT:VT:VA
   :WA:WV:WI:WY", ShippingState))
6  ),NOT(OR(BillingCountry ="US",BillingCountry ="USA",
7  BillingCountry ="United States", ISBLANK(BillingCountry))),
8  NOT(OR(ShippingCountry ="US",ShippingCountry ="USA",
9  ShippingCountry ="United States", ISBLANK(ShippingCountry))))
```

Error Condition Formula 2 :

```
1  ISCHANGED( Name ) && ( OR( ISPICKVAL( Type ,'Customer - Direct') ,ISPICKVAL( Type
   ,'Customer - Channel') ))
```

3. Create Sales Process and Validate Opportunities

Error Condition Formula

```
AND( Amount > 100000, ISPICKVAL(StageName, 'Closed Won'), Approved__c = false )
```

4. Automate Setups

Due Date Only Formula

```
Today()+7
```

Date Formula

```
CASE(MOD([Opportunity].CloseDate + 180 - DATE(1900, 1, 7),7), 0,
[Opportunity].CloseDate + 181, 6, [Opportunity].CloseDate + 182,
[Opportunity].CloseDate + 180)
```