Apex Triggers

- **Get Started with Apex Triggers**

```
trigger AccountAddressTrigger on Account (before insert, before update) {

    for(Account a: Trigger.New){

        if(a.Match_Billing_Address__c == true && a.BillingPostalCode!= null){

            a.ShippingPostalCode=a.BillingPostalCode;

        }

    }

}
```

- **Bulk Apex Triggers**

```
trigger ClosedOpportunityTrigger on Opportunity (after insert, after update) {
    List<Task> taskList = new List<Task>();
    for(Opportunity opp : [SELECT Id, StageName FROM Opportunity WHERE StageName='Closed Won' AND Id IN : Trigger.New]){
        taskList.add(new Task(Subject='Follow Up Test Task', WhatId = opp.Id));
    }

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

Apex Testing

- **Get Started with Apex Unit Tests**

1. verifyData

```
1 public class VerifyDate {
2
3
4
5   //method to handle potential checks against two
  dates
6
7   public static Date CheckDates(Date date1, Date
  date2) {
8
9     //if date2 is within the next 30 days of date1,
  use date2.  Otherwise use the end of the month
10
11    if(DateWithin30Days(date1,date2)) {
12
13      return date2;
14
15    } else {
16
17      return SetEndOfMonthDate(date1);
18
19    }
20
21  }
22
23
```

```
24
25  //method to check if date2 is within the next 30
  days of date1
26
27  private static Boolean DateWithin30Days(Date
  date1, Date date2) {
28
29    //check for date2 being in the past
30
31          if( date2 < date1) { return false; }
32
33
34
35          //check that date2 is within (>=) 30 days
  of date1
36
37          Date date30Days = date1.addDays(30);
  //create a date 30 days away from date1
38
39    if( date2 >= date30Days ) { return false; }
40
41    else { return true; }
42
43  }
44
45
46
47  //method to return the end of the month of a
  given date
48
49  private static Date SetEndOfMonthDate(Date date1)
```

```
  {
50
51      Integer totalDays =
  Date.daysInMonth(date1.year(), date1.month());
52
53      Date lastDay = Date.newInstance(date1.year(),
  date1.month(), totalDays);
54
55      return lastDay;
56
57   }
58
59
60
61}
62
```

2.TestVerifyDate

```
1 @isTest
2
3 public class TestVerifyDate
4
5 {
6
7      static testMethod void testMethod1()
8
9      {
10
11                  Date d =
```

```
  VerifyDate.CheckDates(System.today(),System.t

12
13                  Date d1 =
  VerifyDate.CheckDates(System.today(),System.t

14
15              }
16
17          }
```

- **Test Apex Triggers**

1.restrictcontactbyname

```
1 trigger RestrictContactByName on Contact (before
  insert, before update) {
2
3
4
5  //check contacts prior to insert or update for
  invalid data
6
7  For (Contact c : Trigger.New) {
8
9       if(c.LastName == 'INVALIDNAME') {
  //invalidname is invalid
10
11          c.AddError('The Last Name
  ed for DML');
```

```
12
13      }
14
15
16
17 }
18
19
20
21}
22
```

2.testrestrictcontactname

```
1 @isTest
2
3 private class TestRestrictContactByName {
4
5
6
7     static testMethod void  metodoTest()
8
9     {
10
11
12
13        List<Contact> listContact= new List<Contact>();
14
15        Contact c1 = new Contact(FirstName='Francesco',
  LastName='Riggio' , email='Test@test.com');
16
17        Contact c2 = new Contact(FirstName='Francesco1',
  LastName = 'INVALIDNAME',email='Test@test.com');
18
19        listContact.add(c1);
```

```
20
21          listContact.add(c2);
22
23
24
25          Test.startTest();
26
27              try
28
29              {
30
31                  insert listContact;
32
33              }
34
35              catch(Exception ee)
36
37              {
38
39              }
40
41
42
43          Test.stopTest();
44
45      }
46
47  }
```

- Create Test Data for Apex Tests

```
1  //@isTest
2  public class RandomContactFactory {
3
4      public static List<Contact> generateRandomContacts(Integer
   numContactsToGenerate, String FName) {
5
```

```
        List<Contact> contactList = new List<Contact>();


    for(Integer i=0;i<numContactsToGenerate;i++) {

            Contact c = new Contact(FirstName=FName + ' ' + i,
LastName = 'Contact '+i);

            contactList.add(c);

            System.debug(c);

        }

        //insert contactList;

        System.debug(contactList.size());

        return contactList;

    }


}
```

Asynchronous Apex

- **Use Future Methods**

1.1 AccountProcessor

```
public class AccountProcessor {

    @future

    public static void countContacts(List<Id> accountIds){
```

```
6
7         List<Account> accounts = [Select Id, Name from
  Account Where Id IN : accountIds];
8
9         List<Account> updatedAccounts = new
  List<Account>();
10
11        for(Account account : accounts){
12
13           account.Number_of_Contacts__c = [Select count()
  from Contact Where AccountId =: account.Id];
14
15            System.debug('No Of Contacts = ' +
  account.Number_of_Contacts__c);
16
17            updatedAccounts.add(account);
18
19        }
20
21        update updatedAccounts;
22
23    }
24
25
26
27 }
```

1.2 AccountProcessorTest

```
1 @isTest
2
3 public class AccountProcessorTest {
4
5     @isTest
6
```

```
7       public static void testNoOfContacts(){
8
9           Account a = new Account();
10
11          a.Name = 'Test Account';
12
13          Insert a;
14
15
16
17          Contact c = new Contact();
18
19          c.FirstName = 'Bob';
20
21          c.LastName =  'Willie';
22
23          c.AccountId = a.Id;
24
25
26
27          Contact c2 = new Contact();
28
29          c2.FirstName = 'Tom';
30
31          c2.LastName = 'Cruise';
32
33          c2.AccountId = a.Id;
34
35
36
37          List<Id> acctIds = new List<Id>();
```

```
38
39          acctIds.add(a.Id);
40
41
42
43          Test.startTest();
44
45          AccountProcessor.countContacts(acctIds);
46
47          Test.stopTest();
48
49      }
50
51
52
53}
```

- **Use Batch Apex**

3.1 LeadProcessor

```
1 public class LeadProcessor implements
  Database.Batchable<sObject> {
2
3
4
5       public Database.QueryLocator
  start(Database.BatchableContext bc) {
6
7           // collect the batches of records or
  objects to be passed to execute
```

```
8
9           return Database.getQueryLocator([Select
  LeadSource From Lead ]);
10
11    }
12
13    public void execute(Database.BatchableContext
  bc, List<Lead> leads){
14
15         // process each batch of records
16
17            for (Lead Lead : leads) {
18
19                lead.LeadSource = 'Dreamforce';
20
21            }
22
23        update leads;
24
25    }
26
27    public void finish(Database.BatchableContext
  bc){
28
29      }
30
31
32
33}
```

3.2 LeadProcessorTest

```
@isTest

public class LeadProcessorTest {



        @testSetup

    static void setup() {

        List<Lead> leads = new List<Lead>();

        for(Integer counter=0 ;counter
  <200;counter++){

            Lead lead = new Lead();

            lead.FirstName ='FirstName';

            lead.LastName
  ='LastName'+counter;

            lead.Company ='demo'+counter;

            leads.add(lead);

```

```
25          }
26
27          insert leads;
28
29      }
30
31
32
33      @isTest static void test() {
34
35          Test.startTest();
36
37          LeadProcessor leadProcessor = new
  LeadProcessor();
38
39          Id batchId =
  Database.executeBatch(leadProcessor);
40
41          Test.stopTest();
42
43      }
44
45
46
47  }
```

- **Control Processes with Queueable Apex**

AddPrimaryContact

```
1 public class AddPrimaryContact implements Queueable
2
3 {
4
5     private Contact c;
6
7     private String state;
8
9     public  AddPrimaryContact(Contact c, String
  state)
10
11    {
12
13        this.c = c;
14
15        this.state = state;
16
17    }
18
19    public void execute(QueueableContext context)
20
21    {
22
23          List<Account> ListAccount = [SELECT ID,
  Name ,(Select id,FirstName,LastName from contacts )
  FROM ACCOUNT WHERE BillingState = :state LIMIT
  200];
24
```

```
25          List<Contact> lstContact = new
  List<Contact>();
26
27          for (Account acc:ListAccount)
28
29          {
30
31                  Contact cont =
  c.clone(false,false,false,false);
32
33                  cont.AccountId =  acc.id;
34
35                  lstContact.add( cont );
36
37          }
38
39
40
41          if(lstContact.size() >0 )
42
43          {
44
45              insert lstContact;
46
47          }
48
49
50
51      }
52
53
```

```
54
55}
56
```

AddPrimaryContactTest

```
1 @isTest
2
3 public class AddPrimaryContactTest
4
5 {
6
7       @isTest static void TestList()
8
9       {
10
11          List<Account> Teste = new List <Account>();
12
13          for(Integer i=0;i<50;i++)
14
15          {
16
17              Teste.add(new Account(BillingState =
  'CA', name = 'Test'+i));
18
19          }
20
21          for(Integer j=0;j<50;j++)
22
23          {
24
25              Teste.add(new Account(BillingState =
```

```
  'NY', name = 'Test'+j));
26
27          }
28
29          insert Teste;
30
31
32
33          Contact co = new Contact();
34
35          co.FirstName='demo';
36
37          co.LastName ='demo';
38
39          insert co;
40
41          String state = 'CA';
42
43
44
45           AddPrimaryContact apc = new
  AddPrimaryContact(co, state);
46
47           Test.startTest();
48
49             System.enqueueJob(apc);
50
51           Test.stopTest();
52
53       }
54
```

```
55 }
```

- **Schedule Jobs Using the Apex Scheduler**

DailyLeadProcessor

```
1  public class DailyLeadProcessor implements Schedulable  {
2
3      Public void execute(SchedulableContext SC){
4
5          List<Lead> LeadObj=[SELECT Id from Lead where
   LeadSource=null limit 200];
6
7           for(Lead l:LeadObj){
8
9               l.LeadSource='Dreamforce';
10
11              update l;
12
13          }
14
15      }
16
17 }
18
```

DailyLeadProcessorTest

```
1 @isTest
2
3 private class DailyLeadProcessorTest {
4
5  static testMethod void
  testDailyLeadProcessor() {
6
```

```
7       String CRON_EXP = '0 0 1 * * ?';
8
9       List<Lead> lList = new List<Lead>();
10
11          for (Integer i = 0; i < 200; i++) {
12
13              lList.add(new
  Lead(LastName='Dreamforce'+i, Company='Test1
14
15          }
16
17          insert lList;
18
19
20
21          Test.startTest();
22
23          String jobId =
  System.schedule('DailyLeadProcessor',
  CRON_EXP, new DailyLeadProcessor());
24
25      }
26
27 }
```

Apex Integration Services

- **Apex REST Callouts**

2.1 AnimalLocator

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

10
11         req.setMethod('GET');
12
13         Map<String, Object> animal= new Map<String, Object>();
14
15         HttpResponse res = http.send(req);
16
17             if (res.getStatusCode() == 200) {
18
19         Map<String, Object> results = (Map<String,
   Object>)JSON.deserializeUntyped(res.getBody());
20
21       animal = (Map<String, Object>) results.get('animal');
22
23         }
24
25 return (String)animal.get('name');
26
27     }
```

```
28
29 }
30
```

2.2 AnimalLocatorTest

```
1 @isTest
2
3 private class AnimalLocatorTest{
4
5     @isTest static void AnimalLocatorMock1()
  {
6
7         Test.setMock(HttpCalloutMock.class,
  new AnimalLocatorMock());
8
9         string result =
  AnimalLocator.getAnimalNameById(3);
10
11         String expectedResult = 'chicken';
12
13
  System.assertEquals(result,expectedResult );
14
15     }
16
17 }
18
```

2.3 AnimalLocatorMock

```
@isTest

global class AnimalLocatorMock implements HttpCalloutMock {

      // Implement this interface method

    global HTTPResponse respond(HTTPRequest request) {

        // Create a fake response

        HttpResponse response = new HttpResponse();

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

        response.setBody('{"animals": ["majestic



        response.setStatusCode(200);

        return response;

    }

```

```
23}
```

- **Apex SOAP Callouts**

3.1 ParkLocator

```
1  public class ParkLocator {
2
3      public static string[] country(string theCountry) {
4
5          ParkService.ParksImplPort  parkSvc = new
   ParkService.ParksImplPort(); // remove space
6
7          return parkSvc.byCountry(theCountry);
8
9      }
10
11 }
12
```

3.2 ParkLocatorTest

```
1@isTest
2
3private class ParkLocatorTest {
4
5    @isTest static void testCallout() {
6
7        Test.setMock(WebServiceMock.class, new ParkServiceMock ());
8
9        String country = 'United States';
10
11         List<String> result = ParkLocator.country(country);
12
13         List<String> parks = new List<String>{'Yellowstone',
   'Mackinac National Park', 'Yosemite'};
14
15          System.assertEquals(parks, result);
```

```
16
17     }
18
19 }
```

3.3 ParkServiceMock

```
1  @isTest
2
3  global class ParkServiceMock implements WebServiceMock {
4
5     global void doInvoke(
6
7              Object stub,
8
9              Object request,
10
11             Map<String, Object> response,
12
13             String endpoint,
14
15             String soapAction,
16
17             String requestName,
18
19             String responseNS,
20
21             String responseName,
22
23             String responseType) {
24
25         // start - specify the response you want to send
26
27         ParkService.byCountryResponse response_x = new
   ParkService.byCountryResponse();
28
29         response_x.return_x = new List<String>{'Yellowstone', 'Mackinac

30
31         // end
32
33         response.put('response_x', response_x);
34
35     }
```

```
36
37 }
```

- **Apex Web Services**

4.1 AccountManager

```
1 @RestResource(urlMapping='/Accounts/*/contacts')
2
3 global class AccountManager {
4
5     @HttpGet
6
7     global static Account getAccount() {
8
9         RestRequest req = RestContext.request;
10
11        String accId =
  req.requestURI.substringBetween('Accounts/',
  '/contacts');
12
13        Account acc = [SELECT Id, Name, (SELECT Id,
  Name FROM Contacts)
14
15                       FROM Account WHERE Id =
  :accId];
16
17        return acc;
18
19    }
```

```
20
21}
22
```

4.2 AccountManagerTest

```
1  @isTest
2
3  private class AccountManagerTest {
4
5
6
7      private static testMethod void getAccountTest1() {
8
9          Id recordId = createTestRecord();
10
11         // Set up a test request
12
13         RestRequest request = new RestRequest();
14
15         request.requestUri =
   'https://na1.salesforce.com/services/apexrest/Accounts/'+ recordId
   +'/contacts' ;
16
17         request.httpMethod = 'GET';
18
19         RestContext.request = request;
20
21         // Call the method to test
22
23         Account thisAccount = AccountManager.getAccount();
24
25         // Verify results
26
27         System.assert(thisAccount != null);
28
29         System.assertEquals('Test record', thisAccount.Name);
30
31
32
33     }
```

```
34
35
36
37      // Helper method
38
39          static Id createTestRecord() {
40
41          // Create test record
42
43          Account TestAcc = new Account(
44
45            Name='Test record');
46
47          insert TestAcc;
48
49          Contact TestCon= new Contact(
50
51          LastName='Test',
52
53          AccountId = TestAcc.id);
54
55          return TestAcc.Id;
56
57      }
58
59  }
```

Apex Specialist

- **Automate record creation**

2.1 MaintenanceRequestHelper

```
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);









                }

            }

        }
```

```
28
29
30
31          if (!validIds.isEmpty()){
32
33              List<Case> newCases = new List<Case>();
34
35              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)
36
37
    FROM Case WHERE Id IN :validIds]);
38
39              Map<Id,Decimal> maintenanceCycles = new
    Map<ID,Decimal>();
40
41              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];
42
43
44
45          for (AggregateResult ar : results){
46
47              maintenanceCycles.put((Id)
    ar.get('Maintenance_Request__c'), (Decimal)
    ar.get('cycle'));
48
49          }
50
51
```

```
52
53              for(Case cc : closedCasesM.values()){
54
55                  Case nc = new Case (
56
57                      ParentId = cc.Id,
58
59                  Status = 'New',
60
61                      Subject = 'Routine Maintenance',
62
63                      Type = 'Routine Maintenance',
64
65                      Vehicle__c = cc.Vehicle__c,
66
67                      Equipment__c =cc.Equipment__c,
68
69                      Origin = 'Web',
70
71                      Date_Reported__c = Date.Today()
72
73
74
75                  );
76
77
78
79                  If (maintenanceCycles.containskey(cc.Id)){
80
81                      nc.Date_Due__c =
  Date.today().addDays((Integer)
  maintenanceCycles.get(cc.Id));
82
83                  } else {
84
85                      nc.Date_Due__c =
  Date.today().addDays((Integer)
```

```
   cc.Equipment__r.maintenance_Cycle__c);
86
87                 }
88
89
90
91                 newCases.add(nc);
92
93              }
94
95
96
97             insert newCases;
98
99
100
101             List<Equipment_Maintenance_Item__c> clonedWPs =
   new List<Equipment_Maintenance_Item__c>();
102
103             for (Case nc : newCases){
104
105                 for (Equipment_Maintenance_Item__c wp :
   closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__

106
107                     Equipment_Maintenance_Item__c wpClone
   = wp.clone();
108
109                     wpClone.Maintenance_Request__c =
   nc.Id;
110
111                     ClonedWPs.add(wpClone);
112
113
114
115                 }
116
```

```
            }

            insert ClonedWPs;

        }

    }
}

```

2.2 MaitenanceRequest

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



    if(Trigger.isUpdate && Trigger.isAfter){



MaintenanceRequestHelper.updateWorkOrders(Trigger.New, Trigger.OldMap);


    }


}
```

- **Synchronize Salesforce data with an external system**

  3.1 WarehouseCalloutService

```
1 public with sharing class
  WarehouseCalloutService implements Queueable
  {
2
3     private static final String
  WAREHOUSE_URL = 'https://th-superbadge-


4
5
6
7     //class that makes a REST callout to an
  external warehouse system to get a list of
  equipment that needs to be updated.
8
9     //The callout's JSON response returns
  the equipment records that you upsert in
  Salesforce.
10
11
12
13     @future(callout=true)
14
15     public static void
  runWarehouseEquipmentSync(){
16
```

```
17          Http http = new Http();
18
19          HttpRequest request = new
  HttpRequest();
20
21
22
23          request.setEndpoint(WAREHOUSE_URL);
24
25          request.setMethod('GET');
26
27          HttpResponse response =
  http.send(request);
28
29
30
31          List<Product2> warehouseEq = new
  List<Product2>();
32
33
34
35          if (response.getStatusCode() ==
  200){
36
37              List<Object> jsonResponse =
  (List<Object>)JSON.deserializeUntyped(respons
```

```
38
39
  System.debug(response.getBody());
40
41
42
43              //class maps the following
  fields: replacement part (always true),
  cost, current inventory, lifespan,
  maintenance cycle, and warehouse SKU
44
45              //warehouse SKU will be external
  ID for identifying which equipment records
  to update within Salesforce
46
47              for (Object eq : jsonResponse){
48
49                  Map<String,Object> mapJson =
  (Map<String,Object>)eq;
50
51                  Product2 myEq = new
  Product2();
52
53                  myEq.Replacement_Part__c =
  (Boolean) mapJson.get('replacement');
54
55                  myEq.Name = (String)
```

```
  mapJson.get('name');
56
57                  myEq.Maintenance_Cycle__c =
  (Integer) mapJson.get('maintenanceperiod');
58
59                  myEq.Lifespan_Months__c =
  (Integer) mapJson.get('lifespan');
60
61                  myEq.Cost__c = (Integer)
  mapJson.get('cost');
62
63                  myEq.Warehouse_SKU__c =
  (String) mapJson.get('sku');
64
65                  myEq.Current_Inventory__c =
  (Double) mapJson.get('quantity');
66
67                  myEq.ProductCode = (String)
  mapJson.get('_id');
68
69                  warehouseEq.add(myEq);
70
71              }
72
73
74
75              if (warehouseEq.size() > 0){
```

```
76
77                  upsert warehouseEq;
78
79                  System.debug('Your equipment

80
81              }
82
83          }
84
85      }
86
87
88
89      public static void execute
  (QueueableContext context){
90
91          runWarehouseEquipmentSync();
92
93      }
94
95
96
97 }
98
```

- **Schedule synchronization**

4.1 WarehouseSyncShedule

```
global with sharing class WarehouseSyncSchedule implements Schedulable{

    global void execute(SchedulableContext ctx){

        System.enqueueJob(new WarehouseCalloutService());

    }

}
```

- **Test automation logic**

5.1 MaintenanceRequestHelperTest

```
@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';

```

```
13      private static final string REPAIR = 'Repair';
14
15      private static final string REQUEST_ORIGIN = 'Web';
16
17      private static final string REQUEST_TYPE = 'Routine

18
19      private static final string REQUEST_SUBJECT = 'Testing

20
21
22
23      PRIVATE STATIC Vehicle__c createVehicle(){
24
25          Vehicle__c Vehicle = new Vehicle__C(name =
    'SuperTruck');
26
27          return Vehicle;
28
29      }
30
31
32
33      PRIVATE STATIC Product2 createEq(){
34
35          product2 equipment = new product2(name =
    'SuperEquipment',
36
37                                              lifespan_months__C
    = 10,
38
39
    maintenance_cycle__C = 10,
40
41
    replacement_part__c = true);
```

```
42
43          return equipment;
44
45      }
46
47
48
49      PRIVATE STATIC Case createMaintenanceRequest(id
    vehicleId, id equipmentId){
50
51          case cs = new case(Type=REPAIR,
52
53                              Status=STATUS_NEW,
54
55                              Origin=REQUEST_ORIGIN,
56
57                              Subject=REQUEST_SUBJECT,
58
59                              Equipment__c=equipmentId,
60
61                              Vehicle__c=vehicleId);
62
63          return cs;
64
65      }
66
67
68
69      PRIVATE STATIC Equipment_Maintenance_Item__c
    createWorkPart(id equipmentId,id requestId){
70
71          Equipment_Maintenance_Item__c wp = new
    Equipment_Maintenance_Item__c(Equipment__c = equipmentId,
72
73
    Maintenance_Request__c = requestId);
```

```
74
75          return wp;
76
77      }
78
79
80
81
82
83      @istest
84
85      private static void testMaintenanceRequestPositive(){
86
87          Vehicle__c vehicle = createVehicle();
88
89          insert vehicle;
90
91          id vehicleId = vehicle.Id;
92
93
94
95          Product2 equipment = createEq();
96
97          insert equipment;
98
99          id equipmentId = equipment.Id;
100
101
102
103           case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
104
105           insert somethingToUpdate;
106
107
108
109           Equipment_Maintenance_Item__c workP =
```

```
  createWorkPart(equipmentId,somethingToUpdate.id);
110
111         insert workP;
112
113
114
115         test.startTest();
116
117         somethingToUpdate.status = CLOSED;
118
119         update somethingToUpdate;
120
121         test.stopTest();
122
123
124
125         Case newReq = [Select id, subject, type,
  Equipment__c, Date_Reported__c, Vehicle__c, Date_Due__c
126
127                         from case
128
129                         where status =:STATUS_NEW];
130
131
132
133         Equipment_Maintenance_Item__c workPart = [select
  id
134
135                                                 from
  Equipment_Maintenance_Item__c
136
137                                                 where
  Maintenance_Request__c =:newReq.Id];
138
139
140
141         system.assert(workPart != null);
```

```
142
143         system.assert(newReq.Subject != null);
144
145         system.assertEquals(newReq.Type, REQUEST_TYPE);
146
147         SYSTEM.assertEquals(newReq.Equipment__c,
  equipmentId);
148
149         SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
150
151         SYSTEM.assertEquals(newReq.Date_Reported__c,
  system.today());
152
153     }
154
155
156
157     @istest
158
159     private static void testMaintenanceRequestNegative(){
160
161         Vehicle__C vehicle = createVehicle();
162
163         insert vehicle;
164
165         id vehicleId = vehicle.Id;
166
167
168
169         product2 equipment = createEq();
170
171         insert equipment;
172
173         id equipmentId = equipment.Id;
174
175
```

```
176
177         case emptyReq =
  createMaintenanceRequest(vehicleId,equipmentId);
178
179         insert emptyReq;
180
181
182
183         Equipment_Maintenance_Item__c workP =
  createWorkPart(equipmentId, emptyReq.Id);
184
185         insert workP;
186
187
188
189         test.startTest();
190
191         emptyReq.Status = WORKING;
192
193         update emptyReq;
194
195         test.stopTest();
196
197
198
199         list<case> allRequest = [select id
200
201                                  from case];
202
203
204
205         Equipment_Maintenance_Item__c workPart = [select
  id
206
207                                                   from
  Equipment_Maintenance_Item__c
```

```
208
209                                                    where
   Maintenance_Request__c = :emptyReq.Id];
210
211
212
213         system.assert(workPart != null);
214
215         system.assert(allRequest.size() == 1);
216
217     }
218
219
220
221     @istest
222
223     private static void testMaintenanceRequestBulk(){
224
225         list<Vehicle__C> vehicleList = new
   list<Vehicle__C>();
226
227         list<Product2> equipmentList = new
   list<Product2>();
228
229         list<Equipment_Maintenance_Item__c> workPartList =
   new list<Equipment_Maintenance_Item__c>();
230
231         list<case> requestList = new list<case>();
232
233         list<id> oldRequestIds = new list<id>();
234
235
236
237         for(integer i = 0; i < 300; i++){
238
239            vehicleList.add(createVehicle());
```

```
240
241             equipmentList.add(createEq());
242
243         }
244
245         insert vehicleList;
246
247         insert equipmentList;
248
249
250
251         for(integer i = 0; i < 300; i++){
252
253
  requestList.add(createMaintenanceRequest(vehicleList.get(i)
  .id, equipmentList.get(i).id));
254
255         }
256
257         insert requestList;
258
259
260
261         for(integer i = 0; i < 300; i++){
262
263
  workPartList.add(createWorkPart(equipmentList.get(i).id,
  requestList.get(i).id));
264
265         }
266
267         insert workPartList;
268
269
270
271         test.startTest();
```

```
272
273         for(case req : requestList){
274
275             req.Status = CLOSED;
276
277             oldRequestIds.add(req.Id);
278
279         }
280
281         update requestList;
282
283         test.stopTest();
284
285
286
287         list<case> allRequests = [select id
288
289                                   from case
290
291                                   where status =:
    STATUS_NEW];
292
293
294
295         list<Equipment_Maintenance_Item__c> workParts =
    [select id
296
297
    from Equipment_Maintenance_Item__c
298
299
    where Maintenance_Request__c in: oldRequestIds];
300
301
302
303         system.assert(allRequests.size() == 300);
```

```
304
305     }
306
307 }
```

5.2MaintenanceRequestHelper

```
1  public with sharing class MaintenanceRequestHelper {
2
3      public static void updateworkOrders(List<Case> updWorkOrders,
   Map<Id,Case> nonUpdCaseMap) {
4
5          Set<Id> validIds = new Set<Id>();
6
7
8
9
10
11         For (Case c : updWorkOrders){
12
13             if (nonUpdCaseMap.get(c.Id).Status != 'Closed' &&
   c.Status == 'Closed'){
14
15                 if (c.Type == 'Repair' || c.Type == 'Routine

16
17                     validIds.add(c.Id);
18
19
20
21
22
23                 }
24
25             }
26
27         }
28
29
30
```

```
31          if (!validIds.isEmpty()){
32
33              List<Case> newCases = new List<Case>();
34
35              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)
36
37                                                              FROM
    Case WHERE Id IN :validIds]);
38
39              Map<Id,Decimal> maintenanceCycles = new
    Map<ID,Decimal>();
40
41              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];
42
43
44
45          for (AggregateResult ar : results){
46
47              maintenanceCycles.put((Id)
    ar.get('Maintenance_Request__c'), (Decimal) ar.get('cycle'));
48
49          }
50
51
52
53              for(Case cc : closedCasesM.values()){
54
55                  Case nc = new Case (
56
57                      ParentId = cc.Id,
58
59                  Status = 'New',
60
```

```
61                      Subject = 'Routine Maintenance',
62
63                      Type = 'Routine Maintenance',
64
65                      Vehicle__c = cc.Vehicle__c,
66
67                      Equipment__c =cc.Equipment__c,
68
69                      Origin = 'Web',
70
71                      Date_Reported__c = Date.Today()
72
73
74
75                  );
76
77
78
79                  If (maintenanceCycles.containskey(cc.Id)){
80
81                      nc.Date_Due__c =
   Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
82
83                  }
84
85
86
87                  newCases.add(nc);
88
89              }
90
91
92
93              insert newCases;
94
95
96
97              List<Equipment_Maintenance_Item__c> clonedWPs = new
   List<Equipment_Maintenance_Item__c>();
98
```

```
99            for (Case nc : newCases){
100
101                for (Equipment_Maintenance_Item__c wp :
   closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
102
103                    Equipment_Maintenance_Item__c wpClone =
   wp.clone();
104
105                    wpClone.Maintenance_Request__c = nc.Id;
106
107                    ClonedWPs.add(wpClone);
108
109
110
111                }
112
113            }
114
115            insert ClonedWPs;
116
117        }
118
119    }
120
121 }
```

5.3 MaintenanceRequest

```
1  trigger MaintenanceRequest on Case (before update, after update)
   {
2
3      if(Trigger.isUpdate && Trigger.isAfter){
4
5          MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
   Trigger.OldMap);
6
7      }
8
9  }
```

- **Test callout logic**

  6.1 WarehouseCalloutService

```
public with sharing class WarehouseCalloutService {



    private static final String WAREHOUSE_URL =
'https://th-superbadge-


    //@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);

```

```
27
28
29
30
31          List<Product2> warehouseEq = new
  List<Product2>();
32
33
34
35          if (response.getStatusCode() == 200){
36
37              List<Object> jsonResponse =
  (List<Object>)JSON.deserializeUntyped(response.getB

38
39              System.debug(response.getBody());
40
41
42
43              for (Object eq : jsonResponse){
44
45                  Map<String,Object> mapJson =
  (Map<String,Object>)eq;
46
47                  Product2 myEq = new Product2();
48
49                  myEq.Replacement_Part__c =
  (Boolean) mapJson.get('replacement');
50
51                  myEq.Name = (String)
  mapJson.get('name');
```

```
52
53                 myEq.Maintenance_Cycle__c =
  (Integer) mapJson.get('maintenanceperiod');
54
55                 myEq.Lifespan_Months__c = (Integer)
  mapJson.get('lifespan');
56
57                 myEq.Cost__c = (Decimal)
  mapJson.get('lifespan');
58
59                 myEq.Warehouse_SKU__c = (String)
  mapJson.get('sku');
60
61                 myEq.Current_Inventory__c =
  (Double) mapJson.get('quantity');
62
63                 warehouseEq.add(myEq);
64
65             }
66
67
68
69             if (warehouseEq.size() > 0){
70
71                 upsert warehouseEq;
72
73                 System.debug('Your equipment was

74
75                 System.debug(warehouseEq);
76
```

```
            }


        }

    }

}

```

6.2 WarehouseCalloutServiceTest

```apex
@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]);

    }

}
```

6.3 WarehouseCalloutServiceMock

```
1 @isTest
2
3 global class WarehouseCalloutServiceMock
  implements HttpCalloutMock {
4
5     // implement http mock callout
6
7     global static HttpResponse
  respond(HttpRequest request){
8
9
10
11         System.assertEquals('https://th-
  ,
  request.getEndpoint());
12
13         System.assertEquals('GET',
  request.getMethod());
14
15
16
17         // Create a fake response
18
19         HttpResponse response = new
  HttpResponse();
20
```

```
21          response.setHeader('Content-Type',
  'application/json');
22
23
  response.setBody('[{"_id":"55d66226726b611100
```

```
24
25          response.setStatusCode(200);
26
27          return response;
28
29      }
30
31 }
```

- **Test callout logic**

  6.1 WarehouseCalloutService

```
1 public with sharing class WarehouseCalloutService {
2
3
4
5     private static final String WAREHOUSE_URL =
  'https://th-superbadge-
```

```
6
7
8
9       //@future(callout=true)
10
11      public static void runWarehouseEquipmentSync(){
12
13
14
15          Http http = new Http();
16
17          HttpRequest request = new HttpRequest();
18
19
20
21          request.setEndpoint(WAREHOUSE_URL);
22
23          request.setMethod('GET');
24
25          HttpResponse response = http.send(request);
26
27
28
29
30
31          List<Product2> warehouseEq = new
  List<Product2>();
32
33
34
35          if (response.getStatusCode() == 200){
```

```
36
37              List<Object> jsonResponse =
  (List<Object>)JSON.deserializeUntyped(response.getB

38
39              System.debug(response.getBody());
40
41
42
43              for (Object eq : jsonResponse){
44
45                  Map<String,Object> mapJson =
  (Map<String,Object>)eq;
46
47                  Product2 myEq = new Product2();
48
49                  myEq.Replacement_Part__c =
  (Boolean) mapJson.get('replacement');
50
51                  myEq.Name = (String)
  mapJson.get('name');
52
53                  myEq.Maintenance_Cycle__c =
  (Integer) mapJson.get('maintenanceperiod');
54
55                  myEq.Lifespan_Months__c = (Integer)
  mapJson.get('lifespan');
56
57                  myEq.Cost__c = (Decimal)
  mapJson.get('lifespan');
58
```

```
59                  myEq.Warehouse_SKU__c = (String)
  mapJson.get('sku');
60
61                  myEq.Current_Inventory__c =
  (Double) mapJson.get('quantity');
62
63                  warehouseEq.add(myEq);
64
65              }
66
67
68
69              if (warehouseEq.size() > 0){
70
71                  upsert warehouseEq;
72
73                  System.debug('Your equipment was

74
75                  System.debug(warehouseEq);
76
77              }
78
79
80
81          }
82
83      }
84
85}
```

6.2 WarehouseCalloutServiceTest

```
@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]);

    }

}
```

6.3 WarehouseCalloutServiceMock

```
@isTest

global class WarehouseCalloutServiceMock implements
HttpCalloutMock {

    // implement http mock callout

    global static HttpResponse respond(HttpRequest request){


        System.assertEquals('https://th-superbadge-
));

        System.assertEquals('GET', request.getMethod());


        // Create a fake response

        HttpResponse response = new HttpResponse();

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

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




        response.setStatusCode(200);

        return response;

    }

}
```

Test scheduling logic

7.1 WarehouseSyncSchedule

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

        WarehouseCalloutService.runWarehouseEquipmentSync();
    }
}
```

7.2 WarehouseSyncScheduleTest

```
@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

WarehouseSyncSchedule());
        Test.stopTest();
        //Contains schedule information for a scheduled
job. CronTrigger is similar to a cron job on UNIX systems.
        CronTrigger a=[SELECT Id FROM CronTrigger where
NextFireTime > today];
        System.assertEquals(jobID, a.Id,'Schedule ');


    }
}
```