Apex Specialist Superbadge:

In this superbadge, initial step is to create a new playground. Now the steps which are mentioned in 'set up development org' has to be done. Then according to the given process, write the code for each step mentioned below:

Step 1 : Answering the multiple choice questions.

Step 2 - Automate Record Creation :

Automate record creation using apex triggers.
Go to developer console and edit the apex class and the triggers for below:

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

                    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();
53                      wpClone.Maintenance_Request__c = nc.Id;
54                      ClonedWPs.add(wpClone);
55
56                  }
57              }
58              insert ClonedWPs;
59          }
60      }
61  }
```

MaintenanceRequestHelperTest

```
1   @istest
2   public with sharing class MaintenanceRequestHelperTest {
3
4       private static final string STATUS_NEW = 'New';
5       private static final string WORKING = 'Working';
6       private static final string CLOSED = 'Closed';
7       private static final string REPAIR = 'Repair';
8       private static final string REQUEST_ORIGIN = 'Web';
9       private static final string REQUEST_TYPE = 'Routine

10      private static final string REQUEST_SUBJECT = 'Testing

11
12      PRIVATE STATIC Vehicle__c createVehicle(){
13          Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
14          return Vehicle;
15      }
16
17      PRIVATE STATIC Product2 createEq(){
18          product2 equipment = new product2(name =
    'SuperEquipment',
19                                            lifespan_months__C = 10,
20                                            maintenance_cycle__C =
    10,
21                                            replacement_part__c =
    true);
```

```
22          return equipment;
23      }
24
25      PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id
    equipmentId){
26          case cs = new case(Type=REPAIR,
27                             Status=STATUS_NEW,
28                             Origin=REQUEST_ORIGIN,
29                             Subject=REQUEST_SUBJECT,
30                             Equipment__c=equipmentId,
31                             Vehicle__c=vehicleId);
32          return cs;
33      }
34
35      PRIVATE STATIC Equipment_Maintenance_Item__c
    createWorkPart(id equipmentId,id requestId){
36          Equipment_Maintenance_Item__c wp = new
    Equipment_Maintenance_Item__c(Equipment__c = equipmentId,
37
    Maintenance_Request__c = requestId);
38          return wp;
39      }
40
41
42      @istest
43      private static void testMaintenanceRequestPositive(){
44          Vehicle__c vehicle = createVehicle();
45          insert vehicle;
46          id vehicleId = vehicle.Id;
47
48          Product2 equipment = createEq();
49          insert equipment;
50          id equipmentId = equipment.Id;
51
52          case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
53          insert somethingToUpdate;
54
55          Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId,somethingToUpdate.id);
```

```
56          insert workP;
57
58          test.startTest();
59          somethingToUpdate.status = CLOSED;
60          update somethingToUpdate;
61          test.stopTest();
62
63          Case newReq = [Select id, subject, type, Equipment__c,
    Date_Reported__c, Vehicle__c, Date_Due__c
64                         from case
65                         where status =:STATUS_NEW];
66
67          Equipment_Maintenance_Item__c workPart = [select id
68                                                    from
    Equipment_Maintenance_Item__c
69                                                    where
    Maintenance_Request__c =:newReq.Id];
70
71          system.assert(workPart != null);
72          system.assert(newReq.Subject != null);
73          system.assertEquals(newReq.Type, REQUEST_TYPE);
74          SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
75          SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
76          SYSTEM.assertEquals(newReq.Date_Reported__c,
    system.today());
77      }
78
79      @istest
80      private static void testMaintenanceRequestNegative(){
81          Vehicle__C vehicle = createVehicle();
82          insert vehicle;
83          id vehicleId = vehicle.Id;
84
85          product2 equipment = createEq();
86          insert equipment;
87          id equipmentId = equipment.Id;
88
89          case emptyReq =
    createMaintenanceRequest(vehicleId,equipmentId);
90          insert emptyReq;
```

```
91
92          Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId, emptyReq.Id);
93          insert workP;
94
95          test.startTest();
96          emptyReq.Status = WORKING;
97          update emptyReq;
98          test.stopTest();
99
100                                 list<case> allRequest = [select
    id
101                                                         from
    case];
102
103                                 Equipment_Maintenance_Item__c
    workPart = [select id
104
    from Equipment_Maintenance_Item__c
105
    where Maintenance_Request__c = :emptyReq.Id];
106
107                                 system.assert(workPart != null);
108                                 system.assert(allRequest.size()
    == 1);
109                             }
110
111                             @istest
112                             private static void
    testMaintenanceRequestBulk(){
113                                 list<Vehicle__C> vehicleList =
    new list<Vehicle__C>();
114                                 list<Product2> equipmentList =
    new list<Product2>();
115
    list<Equipment_Maintenance_Item__c> workPartList = new
    list<Equipment_Maintenance_Item__c>();
116                                 list<case> requestList = new
    list<case>();
117                                 list<id> oldRequestIds = new
```

```
    list<id>();
118
119                                 for(integer i = 0; i < 300;
    i++){
120
    vehicleList.add(createVehicle());
121
    equipmentList.add(createEq());
122                                 }
123                                 insert vehicleList;
124                                 insert equipmentList;
125
126                                 for(integer i = 0; i < 300;
    i++){
127
    requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
    equipmentList.get(i).id));
128                                 }
129                                 insert requestList;
130
131                                 for(integer i = 0; i < 300;
    i++){
132
    workPartList.add(createWorkPart(equipmentList.get(i).id,
    requestList.get(i).id));
133                                 }
134                                 insert workPartList;
135
136                                 test.startTest();
137                                 for(case req : requestList){
138                                     req.Status = CLOSED;
139                                     oldRequestIds.add(req.Id);
140                                 }
141                                 update requestList;
142                                 test.stopTest();
143
144                                 list<case> allRequests = [select
    id
145                                                       from
    case
146                                                       where
```

```
    status =: STATUS_NEW];

    list<Equipment_Maintenance_Item__c> workParts = [select id
    from Equipment_Maintenance_Item__c
    where Maintenance_Request__c in: oldRequestIds];

                                    system.assert(allRequests.size()
    == 300);
                                }
                            }
```

Step 3 - Synchronize the salesforce data with an external system:

Modify the Apex Classes as below, save and run all.

WarehouseCalloutService

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


    //class that makes a REST callout to an external warehouse
system to get a list of equipment that needs to be updated.
    //The callout's JSON response returns the equipment records
that you upsert in Salesforce.

    @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);

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

```
17
18          if (response.getStatusCode() == 200){
19              List<Object> jsonResponse =
    (List<Object>)JSON.deserializeUntyped(response.getBody());
20              System.debug(response.getBody());
21
22              //class maps the following fields: replacement part
    (always true), cost, current inventory, lifespan, maintenance
    cycle, and warehouse SKU
23              //warehouse SKU will be external ID for identifying
    which equipment records to update within Salesforce
24              for (Object eq : jsonResponse){
25                  Map<String,Object> mapJson =
    (Map<String,Object>)eq;
26                  Product2 myEq = new Product2();
27                  myEq.Replacement_Part__c = (Boolean)
    mapJson.get('replacement');
28                  myEq.Name = (String) mapJson.get('name');
29                  myEq.Maintenance_Cycle__c = (Integer)
    mapJson.get('maintenanceperiod');
30                  myEq.Lifespan_Months__c = (Integer)
    mapJson.get('lifespan');
31                  myEq.Cost__c = (Integer) mapJson.get('cost');
32                  myEq.Warehouse_SKU__c = (String)
    mapJson.get('sku');
33                  myEq.Current_Inventory__c = (Double)
    mapJson.get('quantity');
34                  myEq.ProductCode = (String) mapJson.get('_id');
35                  warehouseEq.add(myEq);
36              }
37
38              if (warehouseEq.size() > 0){
39                  upsert warehouseEq;
40                  System.debug('Your equipment was synced with the

41              }
42          }
43      }
44
45      public static void execute (QueueableContext context){
```

```
        runWarehouseEquipmentSync();
    }

}
```

Step 4 - Schedule Synchronization:

Modify the Apex Classes as below, save and run all.

WarehouseSyncSchdeule

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

Step 5 - Test automation logic :

Modify the Apex Classes as below, save and run all.

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

                    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;
        }
    }
}
```

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';
    private static final string REPAIR = 'Repair';
    private static final string REQUEST_ORIGIN = 'Web';
    private static final string REQUEST_TYPE = 'Routine

    private static final string REQUEST_SUBJECT = 'Testing


    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;
```

```
46          id vehicleId = vehicle.Id;
47
48          Product2 equipment = createEq();
49          insert equipment;
50          id equipmentId = equipment.Id;
51
52          case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
53          insert somethingToUpdate;
54
55          Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId,somethingToUpdate.id);
56          insert workP;
57
58          test.startTest();
59          somethingToUpdate.status = CLOSED;
60          update somethingToUpdate;
61          test.stopTest();
62
63          Case newReq = [Select id, subject, type, Equipment__c,
    Date_Reported__c, Vehicle__c, Date_Due__c
64                         from case
65                         where status =:STATUS_NEW];
66
67          Equipment_Maintenance_Item__c workPart = [select id
68                                                    from
    Equipment_Maintenance_Item__c
69                                                    where
    Maintenance_Request__c =:newReq.Id];
70
71          system.assert(workPart != null);
72          system.assert(newReq.Subject != null);
73          system.assertEquals(newReq.Type, REQUEST_TYPE);
74          SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
75          SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
76          SYSTEM.assertEquals(newReq.Date_Reported__c,
    system.today());
77      }
78
79      @istest
```

```
80      private static void testMaintenanceRequestNegative(){
81          Vehicle__C vehicle = createVehicle();
82          insert vehicle;
83          id vehicleId = vehicle.Id;
84
85          product2 equipment = createEq();
86          insert equipment;
87          id equipmentId = equipment.Id;
88
89          case emptyReq =
    createMaintenanceRequest(vehicleId,equipmentId);
90          insert emptyReq;
91
92          Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId, emptyReq.Id);
93          insert workP;
94
95          test.startTest();
96          emptyReq.Status = WORKING;
97          update emptyReq;
98          test.stopTest();
99
100                                 list<case> allRequest = [select
    id
101                                                         from
    case];
102
103                                 Equipment_Maintenance_Item__c
    workPart = [select id
104
    from Equipment_Maintenance_Item__c
105
    where Maintenance_Request__c = :emptyReq.Id];
106
107                                 system.assert(workPart != null);
108                                 system.assert(allRequest.size()
    == 1);
109                             }
110
111                             @istest
```

```
112                                 private static void
   testMaintenanceRequestBulk(){
113                                     list<Vehicle__C> vehicleList =
   new list<Vehicle__C>();
114                                     list<Product2> equipmentList =
   new list<Product2>();
115
   list<Equipment_Maintenance_Item__c> workPartList = new
   list<Equipment_Maintenance_Item__c>();
116                                     list<case> requestList = new
   list<case>();
117                                     list<id> oldRequestIds = new
   list<id>();
118
119                                     for(integer i = 0; i < 300;
   i++){
120
   vehicleList.add(createVehicle());
121
   equipmentList.add(createEq());
122                                     }
123                                     insert vehicleList;
124                                     insert equipmentList;
125
126                                     for(integer i = 0; i < 300;
   i++){
127
   requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
   equipmentList.get(i).id));
128                                     }
129                                     insert requestList;
130
131                                     for(integer i = 0; i < 300;
   i++){
132
   workPartList.add(createWorkPart(equipmentList.get(i).id,
   requestList.get(i).id));
133                                     }
134                                     insert workPartList;
135
```

```
136                                     test.startTest();
137                                     for(case req : requestList){
138                                         req.Status = CLOSED;
139                                         oldRequestIds.add(req.Id);
140                                     }
141                                     update requestList;
142                                     test.stopTest();
143
144                                     list<case> allRequests = [select
    id
145                                                               from
    case
146                                                               where
    status =: STATUS_NEW];
147
148
    list<Equipment_Maintenance_Item__c> workParts = [select id
149
    from Equipment_Maintenance_Item__c
150
    where Maintenance_Request__c in: oldRequestIds];
151
152                                     system.assert(allRequests.size()
    == 300);
153                                 }
154                             }
```

Step 6 - Test callout logic :

Modify the Apex Classes as below, save and run all.

WarehouseCalloutServiceTest

```
1  @isTest
2
3  private class WarehouseCalloutServiceTest {
4      @isTest
5      static void testWareHouseCallout(){
6          Test.startTest();
7          // implement mock callout test here
8          Test.setMock(HTTPCalloutMock.class, new
```

```
    WarehouseCalloutServiceMock());
9           WarehouseCalloutService.execute(null);
10          test.stopTest();
11
12          System.assertEquals(1, [SELECT count() FROM Product2]);
13      }
14  }
```

WarehouseCalloutServiceMock

```
1   @isTest
2   global class WarehouseCalloutServiceMock implements
    HttpCalloutMock {
3       // implement http mock callout
4       global static HttpResponse respond(HttpRequest request){
5           System.assertEquals('https://th-superbadge-
    ));
6           System.assertEquals('GET', request.getMethod());
7           // Create a fake response
8           HttpResponse response = new HttpResponse();
9           response.setHeader('Content-Type', 'application/json');
10
    response.setBody('[{"_id":"55d66226726b611100aaf741","replacement




11          response.setStatusCode(200);
12          return response;
13      }
14  }
```

Step 7 - Test scheduling logic :

Modify the Apex Classes as below, save and run all.

WarehouseSyncSchedule

```
1   global with sharing class WarehouseSyncSchedule implements
    Schedulable{
2       global void execute(SchedulableContext ctx){
```

```
        System.enqueueJob(new WarehouseCalloutService());
    }
}
```

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 Schedule

        Test.stopTest();
        //Contains schedule information for a scheduled job.
CronTrigger is similar to a cron job on UNIX systems.
        // This object is available in API version 17.0 and
later.
        CronTrigger a=[SELECT Id FROM CronTrigger where
NextFireTime > today];
        System.assertEquals(jobID, a.Id,'Schedule ');


    }
}
```