16056 Salesforce Developer Catalyst

Self Learning Super Badges

```
3   1
4   APEX SPECIALIST SUPER BADGE CODES
5   APEX TRIGGERS
6   AccountAddressTrigger.axpt:
7   trigger AccountAddressTrigger on Account (before insert,before update) {
8   for(Account account:Trigger.New){
9   if(account.Match_Billing_Address__c == True){
10account.ShippingPostalCode = account.BillingPostalCode;
11}
12}
13}
14ClosedOpportunityTrigger.axpt:
15trigger ClosedOpportunityTrigger on Opportunity (after insert,after update) {
16List<Task> tasklist = new List<Task>();
17for(Opportunity opp: Trigger.New){ 18if(opp.StageName == 'Closed Won'){
19tasklist.add(new Task(Subject = 'Follow Up Test Task',WhatId = opp.Id));
20}
21}
22if(tasklist.size() > 0){
23insert tasklist;
24}
25}
26APEX TESTING
27VerifyData.apxc:
28public class VerifyDate {
29public static Date CheckDates(Date date1, Date date2) {
30if(DateWithin30Days(date1,date2)) {
31return date2;
```

```
} else {
return SetEndOfMonthDate(date1);
}
}
@TestVisible private static Boolean DateWithin30Days(Date date1,
```

```
  Date
//check for date2 being in the past
if                    return false
```

```
39SPSGP-16056-Salesforce Developer Catalyst
40Self-Learning & Super Badges
412
42APEX SPECIALIST SUPER BADGE CODES
43//check that date2 is within (>=) 30 days of date1
44Date date30Days = date1.addDays(30); //create a date 30 days away from date1
45if( date2 >= date30Days ) { return false; }
46else { return true; }
47}
48//method to return the end of the month of a given date
49@TestVisible private static Date SetEndOfMonthDate(Date date1) {
50Integer totalDays = Date.daysInMonth(date1.year(), date1.month());
51Date lastDay = Date.newInstance(date1.year(), date1.month(), totalDays);
52return lastDay;
53}
54}
55TestVerifyData.apxc:
56@isTest
57private class TestVerifyDate {
58@isTest static void Test_CheckDates_case1(){
59Date D =
    VerifyDate.CheckDates(date.parse('01/01/2022'),date.parse('01/05/

60System.assertEquals(date.parse('01/05/2022'), D);
61}
62@isTest static void Test_CheckDates_case2(){
63Date D = VerifyDate.CheckDates(date.parse('01/01/2022'), date.parse('05/05/2022'));
64System.assertEquals(date.parse('01/31/2022'), D);
65}
```

```
66@isTest static void Test_Within30Days_case1(){
67Boolean flag =
    VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
```

'12/30/2021'

System false

@isTest static void Test_Within30Days_case2

```
Boolean flag =
    VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
date.parse('02/02/2021'));
System.assertEquals(false, flag);
}
@isTest static void Test_Within30Days_case3(){
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
3
APEX SPECIALIST SUPER BADGE CODES
Boolean flag =
    VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
date.parse('01/15/2022'));
System.assertEquals(true, flag);
}
@isTest static void Test_SetEndOfMonthDate(){
Date returndate =
    VerifyDate.SetEndOfMonthDate(date.parse('01/01/2022'));
}
}
RestrictContactByName.apxt:
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


}
}
```

```
97}
98TestRestrictContactByName.apxc:
99@isTest
100    private class TestRestrictContactByName {
101    @isTest static void Test_insertupdateContact(){
102    Contact cnt = new Contact();
103    cnt.LastName = 'INVALIDNAME';
```

Test

Database SaveResult Database false

```
Test.stopTest();
System.assert(!result.isSuccess());
System.assert(result.getErrors().size() > 0);
System.assertEquals('The Last Name "INVALIDNAME" is not allowed

result.getErrors()[0].getMessage());
}
}
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
4
APEX SPECIALIST SUPER BADGE CODES
RandomContactFactory.apxc:
public class RandomContactFactory {
public static List<Contact> generateRandomContacts(Integer num_cnts, string lastname) {
List<Contact> contacts = new List<Contact>();
for(Integer i = 0; i < num_cnts; i++) {
Contact cnt = new Contact(FirstName = 'Test' +i,LastName = lastname);
contacts.add(cnt);
}
return contacts;
}
}
ASYNCHRONOUS APEX
AccountProcessor.apxc:
public class AccountProcessor {
@future
public static void countContacts(List<Id> accountIds){
List<Account> accountsToUpdate = new List<Account>();
```

```
134   List<Account> accounts = [Select Id, Name, (Select Id from
      Contacts)from Account Where Id in
135   :accountIds];
136   For(Account acc: accounts) {
137   List<Contact> contactList = acc.contacts;
138   acc.Number_Of_Contacts__c = contactList.size();
```

```
}
}
AccountProcessorTest.apxc:
@isTest
public class AccountProcessorTest {
@isTest
private static void testCountContacts() {
Account newAccount = new Account(Name = 'Test Account');
insert newAccount;
Contact newContact1 = new Contact(FirstName = 'John',LastName = 'Doe',AccountId =
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
5
APEX SPECIALIST SUPER BADGE CODES
newAccount.Id);
insert newContact1;
Contact newContact2 = new Contact(FirstName = 'John',LastName = 'Doe',AccountId =
newAccount.Id);
insert newContact2;
List<Id> accountIds = new List<Id>();
accountIds.add(newAccount.Id);
Test.startTest();
AccountProcessor.countContacts(accountIds);
Test.stopTest();
}
}
LeadProcessor.apxc:
global class LeadProcessor implements Database.Batchable<sObject>{
global Integer count = 0;
```

```
171   global Database.QueryLocator start(Database.BatchableContext bc)
    {
172   return Database.getQueryLocator('SELECT ID,LeadSource FROM

173   }
174   global void execute(Database.BatchableContext bc, List<Lead>
```

List<lead> new List<lead>

```
176     for(lead L: L_list){
177     L.leadSource = 'Dreamforce';
178     L_list_new.add(L);
179     count += 1;
180     }
181     update L_list_new;
182     }
183     global void finish(Database.BatchableContext bc){
184     system.debug('count = ' + count);
185     }
186     }
187     LeadProcessorTest.apxc:
188     @isTest
189     public class LeadProcessorTest {
190     @isTest
191     public static void testit() {
192     SPSGP-16056-Salesforce Developer Catalyst
193     Self-Learning & Super Badges
194     6
195     APEX SPECIALIST SUPER BADGE CODES
196     List<lead> L_list = new List<lead>();
197     for(Integer i = 0; i < 200; i++) {
198     Lead L = new Lead();
199     L.LastName = 'name' + i;
200     L.Company = 'Company';
201     L.Status = 'Random Status';
202     L_list.add(L); 203 }
204     insert L_list;
```

```
205     Test.startTest();
206     LeadProcessor lp = new LeadProcessor(); 207 Id batchId = Database.executeBatch(lp);
208   Test.stopTest();
209   }
210   }
211   AddPrimaryContact.apxc:
212   public class AddPrimaryContact implements Queueable{
```

```
private Contact
private String
```

public AddPrimaryContact Contact String

```
216     this.con = con;
217     this.state = state;
218     }
219     public void execute(QueueableContext context) {
220     List<Account> accounts = [Select Id,Name,(Select
    FirstName,LastName, Id from contacts)
221     from Account where BillingState = :state Limit 200]; 222 List<Contact> primaryContacts =
new List<Contact>();
223  for(Account acc : accounts) {
224  Contact c = con.clone();
225  c.AccountId = acc.Id;
226  primaryContacts.add(c);
227  }
228  if(primaryContacts.size() > 0) {
229  insert primaryContacts;
230  }
231  }
232  }
233  SPSGP-16056-Salesforce Developer Catalyst
234  Self-Learning & Super Badges
235  7
236  APEX SPECIALIST SUPER BADGE CODES
237  AddPrimaryContactTest.apxc:
238  @isTest
239  public class AddPrimaryContactTest {
240  static testmethod void testQueueable() {
241  List<Account> testAccounts = new List<Account>();
242  for(Integer i = 0; i < 50; i++) {
```

```
243    testAccounts.add(new Account (Name = 'Account' + i,BillingState
       = 'CA'));
```

```
244    }
245    for(Integer j = 0; j < 50; j++) {
246    testAccounts.add(new Account(Name = 'Account'+ j, BillingState =
       'NY'));
247    }
248    insert testAccounts;
249    Contact testContact = new Contact(FirstName = 'John', LastName =
```

```
'Doe'

  AddPrimaryContact          new
```

```
AddPrimaryContact(testContact,'CA');
252 Test.startTest();
253 system.enqueueJob(addit);
254 Test.stopTest();
255 System.assertEquals(50, [Select count() from Contact where accountId in (Select Id from
256 Account where BillingState = 'CA')]);
257 }
258 }
259 DailyLeadProcessor.apxc:
260 global class DailyLeadProcessor implements Schedulable{
261 global void execute(SchedulableContext ctx) {
262 List<Lead> leadstoupdate = new List<Lead>();
263 List<Lead> leads = [Select id From Lead Where LeadSource = NULL Limit 200];
264 for(Lead l: leads) {
265 l.LeadSource = 'Dreamforce';
266 leadstoupdate.add(l); 267 }
268 update leadstoupdate;
269 }
270 }
271 SPSGP-16056-Salesforce Developer Catalyst
272 Self-Learning & Super Badges
273 8
274 APEX SPECIALIST SUPER BADGE CODES
275 DailyLeadProcessorTest.apxc:
276 @isTest
277 private class DailyLeadProcessorTest {
278 public static String CRON_EXP = '0 0 0 15 3 ? 2024';
279 static testmethod void testScheduledJob() {
```

```
List<Lead> leads = new List<Lead>();
for(Integer i = 0; i < 200; i++) {
```

```
282    Lead l = new Lead(
283    FirstName = 'First' + i,
284    LastName = 'LastName',
285    Company = 'The Inc'
```

```
insert leads;
Test.startTest();
String jobId = System.schedule('ScheduledApexTest',CRON_EXP,new DailyLeadProcessor());
Test.stopTest();
List<Lead> checkleads = new List<Lead>();
checkleads = [Select Id From Lead Where LeadSource =
'Dreamforce' and Company = 'The Inc'];
System.assertEquals(200,checkleads.size(),'Leads were not

}
}
APEX INTEGRATION SERVICES
AnimalLocator.apxc:
public class AnimalLocator{
public static String getAnimalNameById(Integer x){
Http http = new Http();
HttpRequest req = new HttpRequest();
req.setEndpoint('https://th-apex-http-

req.setMethod('GET');
Map<String, Object> animal= new Map<String, Object>();
HttpResponse res = http.send(req);
if (res.getStatusCode() == 200) {
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
9
APEX SPECIALIST SUPER BADGE CODES
Map<String, Object> results = (Map<String,
Object>)JSON.deserializeUntyped(res.getBody());
```

```
    animal = (Map<String, Object>) results.get('animal');
}
```

```
316   return (String)animal.get('name');
317   }
318   }
319   AnimalLocatorTest.apxc:
320   @isTest
321   private class AnimalLocatorTest{
322   @isTest static void AnimalLocatorMock1() {
```

```
Test                HttpCalloutMock class  new AnimalLocatorMock
```

```
324   string result = AnimalLocator.getAnimalNameById(3);
325   String expectedResult = 'chicken';
```

```
System.assertEquals(result,expectedResult );
}
}
AnimalLocatorMock.apxc:
@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 badger", "fluffy bunny", "scary bear", "chicken",
"mighty
moose"]}');
response.setStatusCode(200);
return response;
}
}
ParkLocator.apxc:
public class ParkLocator {
public static string[] country(string theCountry) {
ParkService.ParksImplPort parkSvc = new
ParkService.ParksImplPort(); // remove space
return parkSvc.byCountry(theCountry);
}
}
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
```

352 10

353 APEX SPECIALIST SUPER BADGE CODES

```
ParkLocatorTest.apxc:
@isTest
private class ParkLocatorTest {
@isTest static void testCallout() {
Test.setMock(WebServiceMock.class, new ParkServiceMock ());
String country = 'United States';
List<String> result = ParkLocator.country(country);
```

```
List String                new List String  'Yellowstone'  'Mackinac
```

```
System.assertEquals(parks, result);
}
}
ParkServiceMock.apxc:
@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) {
// start - specify the response you want to send
ParkService.byCountryResponse response_x = new
ParkService.byCountryResponse();
response_x.return_x = new List<String>{'Yellowstone', 'Mackinac

// end
response.put('response_x', response_x);
}
}
AccountManager.apxc:
@RestResource(urlMapping='/Accounts/*/contacts')
global class AccountManager {
```

```
388   @HttpGet
389   global static Account getAccount() {
```

```
RestRequest req = RestContext.request;
String accId = req.requestURI.substringBetween('Accounts/', '/contacts');
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
11
APEX SPECIALIST SUPER BADGE CODES
Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)
```

```
        Account        Id
return
```

```
401     AccountManagerTest.apxc:
402     @isTest
403     private class AccountManagerTest {
404     private static testMethod void getAccountTest1() {
405     Id recordId = createTestRecord();
406     // Set up a test request
407     RestRequest request = new RestRequest(); 408 request.requestUri =
    'https://na1.salesforce.com/services/apexrest/Accounts/'+ recordId
409     +'/contacts' ;
410     request.httpMethod = 'GET';
411     RestContext.request = request;
412     // Call the method to test
413     Account thisAccount = AccountManager.getAccount();
414     // Verify results
415     System.assert(thisAccount != null);
416     System.assertEquals('Test record', thisAccount.Name);
417     }
418     // Helper method
419     static Id createTestRecord() { 420 // Create test record
421   Account TestAcc = new Account(
422   Name='Test record');
423   insert TestAcc;
424   Contact TestCon= new Contact(
425   LastName='Test',
426   AccountId = TestAcc.id);
427   return TestAcc.Id;
428   }
429   }
```

```
430  SPSGP-16056-Salesforce Developer Catalyst
431  Self-Learning & Super Badges
```

Challenge 1

MaintenanceRequestHelper

public with class MaintenanceRequestHelper

```
438  public static void updateworkOrders(List<Case> updWorkOrders, Map<Id,Case>
     nonUpdCaseMap) {
439  Set<Id> validIds = new Set<Id>();
440  For (Case c : updWorkOrders){
441  if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status == 'Closed'){
442  if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
443  validIds.add(c.Id);
444  }
445  }
446  }
447  if (!validIds.isEmpty()){
448  List<Case> newCases = new List<Case>();
449  Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id, Vehicle__c, Equipment__c,
450  Equipment__r.Maintenance_Cycle__c,(SELECT
     Id,Equipment__c,Quantity__c FROM
451  Equipment_Maintenance_Items__r)
452  FROM Case WHERE Id IN :validIds]);
453  Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
454  AggregateResult[] results = [SELECT Maintenance_Request__c,
455  MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
     Equipment_Maintenance_Item__c WHERE
456  Maintenance_Request__c IN :ValidIds GROUP BY
     Maintenance_Request__c];
457  for (AggregateResult ar : results){
458  maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'),
     (Decimal) ar.get('cycle'));
459  }
460  for(Case cc : closedCasesM.values()){
461  Case nc = new Case (
```

```
462    ParentId = cc.Id,
463    Status = 'New',
464    SPSGP-16056-Salesforce Developer Catalyst
```

Self Learning Super Badges

13

```
Subject    'Routine Maintenance'
Type    'Routine Maintenance'
Vehicle__c        Vehicle__c
Equipment__c       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;
}
}
}
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
14
APEX SPECIALIST SUPER BADGE CODES
MaintenanceRequest.apxt:
trigger MaintenanceRequest on Case (before update, after update)
{
if(Trigger.isUpdate && Trigger.isAfter){
MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
Trigger OldMap
```

```
MaintenanceRequestHelperTest
@istest
public with          class MaintenanceRequestHelperTest
```

```
508   private static final string STATUS_NEW = 'New';
509   private static final string WORKING = 'Working';
510   private static final string CLOSED = 'Closed';
```

```
511  private static final string REPAIR = 'Repair';
512  private static final string REQUEST_ORIGIN = 'Web';
513  private static final string REQUEST_TYPE = 'Routine

514  private static final string REQUEST_SUBJECT = 'Testing subject';
515  PRIVATE STATIC Vehicle__c createVehicle(){
516  Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
517  return Vehicle;
518  }
519  PRIVATE STATIC Product2 createEq(){
520  product2 equipment = new product2(name = 'SuperEquipment',
521  lifespan_months__C = 10,
522  maintenance_cycle__C = 10,
523  replacement_part__c = true);
524  return equipment;
525  }
526  PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id equipmentId){
527  case cs = new case(Type=REPAIR,
528  Status=STATUS_NEW,
529  Origin=REQUEST_ORIGIN,
530  Subject=REQUEST_SUBJECT,
531  Equipment__c=equipmentId,
532  SPSGP-16056-Salesforce Developer Catalyst
533  Self-Learning & Super Badges
534  15
535  APEX SPECIALIST SUPER BADGE CODES
536  Vehicle__c=vehicleId);
537  return cs;
538  }
```

```
539    PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id equipmentId,id
       requestId){
540    Equipment_Maintenance_Item__c wp = new
     Equipment_Maintenance_Item__c(Equipment__c =
```

```
Maintenance_Request__c
return
```

```
@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];
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
16
APEX SPECIALIST SUPER BADGE CODES
Equipment_Maintenance_Item__c workPart = [select id
from Equipment_Maintenance_Item__c
```

```
571     where Maintenance_Request__c =:newReq.Id];
572     system.assert(workPart != null);
573     system.assert(newReq.Subject != null);
574     system.assertEquals(newReq.Type, REQUEST_TYPE);
575     SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
576     SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
577     SYSTEM.assertEquals(newReq.Date_Reported__c, system.today());
```

```
@istest
private static void
```

```
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; 594 test.stopTest();
   list<case> allRequest = [select id
   from case];
   Equipment_Maintenance_Item__c workPart = [select id
   from Equipment_Maintenance_Item__c
   SPSGP-16056-Salesforce Developer Catalyst
   Self-Learning & Super Badges
   17
   APEX SPECIALIST SUPER BADGE CODES
   where Maintenance_Request__c = :emptyReq.Id];
   system.assert(workPart != null);
   system.assert(allRequest.size() == 1);
   }
   @istest
   private static void testMaintenanceRequestBulk(){
```

```
609      list<Vehicle__C> vehicleList = new list<Vehicle__C>(); 610 list<Product2> equipmentList =
new list<Product2>();
611   list<Equipment_Maintenance_Item__c> workPartList = new
612   list<Equipment_Maintenance_Item__c>();
613   list<case> requestList = new list<case>();
614   list<id> oldRequestIds = new list<id>();
615   for(integer i = 0; i < 300; i++){
```

```
for(integer i = 0; i < 300; i++){
requestList.add(createMaintenanceRequest(vehicleList.get(i).id, equipmentList.get(i).id));
}
insert requestList;
for(integer i = 0; i < 300; i++){
workPartList.add(createWorkPart(equipmentList.get(i).id, requestList.get(i).id));
}
insert workPartList;
test.startTest();
for(case req : requestList){
req.Status = CLOSED;
oldRequestIds.add(req.Id);
}
update requestList;
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
18
APEX SPECIALIST SUPER BADGE CODES
test.stopTest();
list<case> allRequests = [select id
from case
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);
}
}
Challenge-2
WarehouseCalloutService.apxc:
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>();
if (response.getStatusCode() == 200){ 665 List<Object> jsonResponse =
(List<Object>)JSON.deserializeUntyped(response.getBody());
   SPSGP-16056-Salesforce Developer Catalyst
   Self-Learning & Super Badges
   19
   APEX SPECIALIST SUPER BADGE CODES 670 System.debug(response.getBody());
//class maps the following fields: replacement part (always true), cost, current inventory,
lifespan, maintenance cycle, and warehouse SKU
//warehouse SKU will be external ID for identifying which equipment records to update within
Salesforce
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 = (Integer) mapJson.get('cost');
myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
myEq.ProductCode = (String) mapJson.get('_id');
```

```
if (warehouseEq.size() > 0){
upsert warehouseEq;
```

System 'Your equipment was synced with the warehouse

```
}
}
}
public static void execute (QueueableContext context){
runWarehouseEquipmentSync();
}
}
WarehouseCalloutServiceMock.apxc:
@isTest
global class WarehouseCalloutServiceMock implements
HttpCalloutMock {
// implement http mock callout
global static HttpResponse respond(HttpRequest request) {
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
20
APEX SPECIALIST SUPER BADGE CODES
HttpResponse response = new HttpResponse();
response.setHeader('Content-Type', 'application/json');
response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
t":false,"quantity":5,"name":"Gene
rator 1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10

af742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
"},{"_id":"55d66226726b611100aaf743
","replacement":true,"quantity":143,"name":"Fuse

20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005" }]');
```

```
716   response.setStatusCode(200);
717   return response;
718   }
719   }
```

WarehouseCalloutServiceTest

```
722  private class WarehouseCalloutServiceTest {
723  // implement your mock callout test here
724  @isTest
725  static void testWarehouseCallout() {
726  test.startTest();
727  test.setMock(HttpCalloutMock.class, new
     WarehouseCalloutServiceMock());
728  WarehouseCalloutService.execute(null);
729  test.stopTest();
730  List<Product2> product2List = new List<Product2>();
731  product2List = [SELECT ProductCode FROM Product2];
732  System.assertEquals(3, product2List.size());
733  System.assertEquals('55d66226726b611100aaf741', product2List.get(0).ProductCode);
734  System.assertEquals('55d66226726b611100aaf742', product2List.get(1).ProductCode);
735  System.assertEquals('55d66226726b611100aaf743', product2List.get(2).ProductCode);
736  }
737  }
738  Challenge-3
739  WarehouseSyncSchedule.apxc:
740  global with sharing class WarehouseSyncSchedule implements Schedulable{
741  SPSGP-16056-Salesforce Developer Catalyst
742  Self-Learning & Super Badges
743  21
744  APEX SPECIALIST SUPER BADGE CODES
745  global void execute(SchedulableContext ctx){
746  System.enqueueJob(new WarehouseCalloutService());
747  }
748  }
749  WarehouseSyncScheduuleTest.apxc:
750  @isTest
751  public class WarehouseSyncScheduleTest {
752  @isTest static void WarehousescheduleTest(){
753  String scheduleTime = '00 00 01 * * ?';
754  Test.startTest();
755  Test.setMock(HttpCalloutMock.class, new
```

```
    WarehouseCalloutServiceMock());
756 String jobID=System.schedule('Warehouse Time To Schedule to

757 WarehouseSyncSchedule());
758 Test.stopTest();
759 //Contains schedule information for a scheduled job. CronTrigger is similar to a cron job on
    UNIX
760 systems.
761 // This object is available in API version 17.0 and later.
762 CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime > today];
763 System.assertEquals(jobID, a.Id,'Schedule ');
764 }
765 }
766 Challenge-4
767 MaintenanceRequestHelperTest.apxc:
768 @istest
769 public with sharing class MaintenanceRequestHelperTest {
770 private static final string STATUS_NEW = 'New';
771 private static final string WORKING = 'Working';
772 private static final string CLOSED = 'Closed';
773 private static final string REPAIR = 'Repair';
774 private static final string REQUEST_ORIGIN = 'Web';
775 private static final string REQUEST_TYPE = 'Routine

776 private static final string REQUEST_SUBJECT = 'Testing subject';
777 PRIVATE STATIC Vehicle__c createVehicle(){
778 SPSGP-16056-Salesforce Developer Catalyst
779 Self-Learning & Super Badges
780 22
```

```
781  APEX SPECIALIST SUPER BADGE CODES
782  Vehicle__c Vehicle = new Vehicle__C(name = 'SuperTruck');
783  return Vehicle;
784  }
785  PRIVATE STATIC Product2 createEq(){
786  product2 equipment = new product2(name = 'SuperEquipment',
787  lifespan_months__C = 10,
788  maintenance_cycle__C = 10,
789  replacement_part__c = true);
790  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;
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
23
APEX SPECIALIST SUPER BADGE CODES
```

```
818  case somethingToUpdate = createMaintenanceRequest(vehicleId,equipmentId);
819  insert somethingToUpdate;
820  Equipment_Maintenance_Item__c workP =
     createWorkPart(equipmentId,somethingToUpdate.id);
821  insert workP;
```

Case Select Equipment__c

```
    Date_Reported__c, Vehicle__c,
827     Date_Due__c
828     from case
829     where status =:STATUS_NEW];
830     Equipment_Maintenance_Item__c workPart = [select id
831     from Equipment_Maintenance_Item__c
832     where Maintenance_Request__c =:newReq.Id];
833     system.assert(workPart != null);
834     system.assert(newReq.Subject != null);
835     system.assertEquals(newReq.Type, REQUEST_TYPE);
836     SYSTEM.assertEquals(newReq.Equipment__c, equipmentId);
837     SYSTEM.assertEquals(newReq.Vehicle__c, vehicleId);
838     SYSTEM.assertEquals(newReq.Date_Reported__c, system.today()); 839 }
840   @istest
841   private static void testMaintenanceRequestNegative(){
842   Vehicle__C vehicle = createVehicle();
843   insert vehicle;
844   id vehicleId = vehicle.Id;
845   product2 equipment = createEq();
846   insert equipment;
847   id equipmentId = equipment.Id;
848   SPSGP-16056-Salesforce Developer Catalyst
849   Self-Learning & Super Badges
850   24
851   APEX SPECIALIST SUPER BADGE CODES
852   case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);
853   insert emptyReq;
854   Equipment_Maintenance_Item__c workP = createWorkPart(equipmentId, emptyReq.Id);
```

```
855  insert workP;
856  test.startTest();
857  emptyReq.Status = WORKING;
858  update emptyReq; 859 test.stopTest();
860  list<case> allRequest = [select id
861  from case];
862  Equipment_Maintenance_Item__c workPart = [select id
```

```
from Equipment_Maintenance_Item__c
where Maintenance_Request__c                    Id
```

```
865     system.assert(workPart != null);
866     system.assert(allRequest.size() == 1);
867     }
868     @istest
869     private static void testMaintenanceRequestBulk(){
870     list<Vehicle__C> vehicleList = new list<Vehicle__C>(); 871 list<Product2> equipmentList =
new list<Product2>();
872     list<Equipment_Maintenance_Item__c> workPartList = new
873     list<Equipment_Maintenance_Item__c>();
874     list<case> requestList = new list<case>();
875     list<id> oldRequestIds = new list<id>();
876     for(integer i = 0; i < 300; i++){ 877 vehicleList.add(createVehicle());
878     equipmentList.add(createEq());
879     }
880     insert vehicleList;
881     insert equipmentList;
882     SPSGP-16056-Salesforce Developer Catalyst
883     Self-Learning & Super Badges
884     25
885     APEX SPECIALIST SUPER BADGE CODES 886 for(integer i = 0; i < 300; i++){
887  requestList.add(createMaintenanceRequest(vehicleList.get(i).id, equipmentList.get(i).id));
888  }
889  insert requestList;
890  for(integer i = 0; i < 300; i++){
891  workPartList.add(createWorkPart(equipmentList.get(i).id, requestList.get(i).id));
892  }
893  insert workPartList;
894  test.startTest();
```

```
895   for(case req : requestList){
896   req.Status = CLOSED;
897   oldRequestIds.add(req.Id);
```

<case> select

from case

where

```
list<Equipment_Maintenance_Item__c> workParts = [select id
from Equipment_Maintenance_Item__c
where Maintenance_Request__c in: oldRequestIds];
system.assert(allRequests.size() == 300);
}
}
MaintenanceRequestHelper.apxc:
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'){
SPSGP-16056-Salesforce Developer Catalyst
Self-Learning & Super Badges
26
APEX SPECIALIST SUPER BADGE CODES
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
```

```
934    Maintenance_Request__c IN :ValidIds GROUP BY
```

```
Maintenance_Request__c
for AggregateResult Id
get 'Maintenance_Request__c'
```

```
    (Decimal) ar.get('cycle'));
937 }
938 for(Case cc : closedCasesM.values()){
939 Case nc = new Case (
940 ParentId = cc.Id,
941 Status = 'New',
942 Subject = 'Routine Maintenance',
943 Type = 'Routine Maintenance',
944 Vehicle__c = cc.Vehicle__c,
945 Equipment__c =cc.Equipment__c,
946 Origin = 'Web',
947 Date_Reported__c = Date.Today()
948 );
949 If (maintenanceCycles.containskey(cc.Id)){
950 nc.Date_Due__c = Date.today().addDays((Integer) maintenanceCycles.get(cc.Id));
951 SPSGP-16056-Salesforce Developer Catalyst
952 Self-Learning & Super Badges
953 27
954 APEX SPECIALIST SUPER BADGE CODES
955 }
956 newCases.add(nc);
957 }
958 insert newCases;
959 List<Equipment_Maintenance_Item__c> clonedWPs = new
960 List<Equipment_Maintenance_Item__c>();
961 for (Case nc : newCases){
962 for (Equipment_Maintenance_Item__c wp :
963 closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
```

```
964   Equipment_Maintenance_Item__c wpClone = wp.clone();
965   wpClone.Maintenance_Request__c = nc.Id;
966   ClonedWPs.add(wpClone);
967   }
968   }
969   insert ClonedWPs;
```

Challenge 5

WarehouseCalloutService

```
public with          class WarehouseCalloutService implements
```

```
    Queueable {
976   private static final String WAREHOUSE_URL = 'https://th-
```

```
977  //class that makes a REST callout to an external warehouse system to get a list of equipment
     that
978  needs to be updated.
979  //The callout's JSON response returns the equipment records that you upsert in Salesforce.
980  @future(callout=true)
981  public static void runWarehouseEquipmentSync(){
982  Http http = new Http();
983  HttpRequest request = new HttpRequest();
984  request.setEndpoint(WAREHOUSE_URL);
985  SPSGP-16056-Salesforce Developer Catalyst
986  Self-Learning & Super Badges
987  28
988  APEX SPECIALIST SUPER BADGE CODES
989  request.setMethod('GET');
990  HttpResponse response = http.send(request);
991  List<Product2> warehouseEq = new List<Product2>();
992  if (response.getStatusCode() == 200){ 993 List<Object> jsonResponse =
    (List<Object>)JSON.deserializeUntyped(response.getBody());
994  System.debug(response.getBody());
995  //class maps the following fields: replacement part (always true), cost, current inventory,
996  lifespan, maintenance cycle, and warehouse SKU
997  //warehouse SKU will be external ID for identifying which equipment records to update within
998  Salesforce
999  for (Object eq : jsonResponse){
1000Map<String,Object> mapJson = (Map<String,Object>)eq;
1001Product2 myEq = new Product2();
1002myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
1003myEq.Name = (String) mapJson.get('name');
1004myEq.Maintenance_Cycle__c = (Integer) mapJson.get('maintenanceperiod');
```

```
1005myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
```

```
Cost__c     Integer                 get 'cost'
```

```
1007myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
1008myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
1009myEq.ProductCode = (String) mapJson.get('_id');
1010warehouseEq.add(myEq);
1011}
1012if (warehouseEq.size() > 0){
1013upsert warehouseEq;
1014System.debug('Your equipment was synced with the warehouse

1015}
1016}
1017}
1018public static void execute (QueueableContext context){
1019runWarehouseEquipmentSync();
1020}
1021SPSGP-16056-Salesforce Developer Catalyst
1022Self-Learning & Super Badges
102329
1024APEX SPECIALIST SUPER BADGE CODES 1025}
1026WarehouseCalloutServiceMock.apxc:
1027@isTest
1028global class WarehouseCalloutServiceMock implements HttpCalloutMock {
1029// implement http mock callout
1030global static HttpResponse respond(HttpRequest request) {
1031HttpResponse response = new HttpResponse();
1032response.setHeader('Content-Type', 'application/json');
1033response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
    t":false,"quantity":5,"name":"Gene
1034rator 1000
1035kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10

1036af742","replacement":true,"quantity":183,"name":"Cooling
1037Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
    "},{"_id":"55d66226726b611100aaf743
1038","replacement":true,"quantity":143,"name":"Fuse
103920A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"
    }]');
```

```
        return
1042}
1043}
1044WarehouseCalloutServiceTest.apxc:
1045@isTest
1046global class WarehouseCalloutServiceMock implements
HttpCalloutMock {
1047// implement http mock callout
1048global static HttpResponse respond(HttpRequest request) {
1049HttpResponse response = new HttpResponse();
1050response.setHeader('Content-Type', 'application/json');
1051response.setBody('[{"_id":"55d66226726b611100aaf741","replacemen
    t":false,"quantity":5,"name":"Gene
1052rator 1000
1053kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10

1054af742","replacement":true,"quantity":183,"name":"Cooling
1055Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
"},{"_id":"55d66226726b611100aaf743
1056","replacement":true,"quantity":143,"name":"Fuse
105720A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005" }]');
1058SPSGP-16056-Salesforce Developer Catalyst
1059Self-Learning & Super Badges
106030
1061APEX SPECIALIST SUPER BADGE CODES
1062response.setStatusCode(200);
1063return response;
1064}
1065}
1066Challenge-6
1067WarehouseSyncSchedule.apxc:
1068global with sharing class WarehouseSyncSchedule implements Schedulable{
1069global void execute(SchedulableContext ctx){
1070System.enqueueJob(new WarehouseCalloutService());
```

```
}
}
WarehouseSyncScheduleTest.apxc:
```

```
@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

WarehouseSyncSchedule());
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 ');
}

```