

```
1 SPSGP-15972-Salesforce Developer Catalyst
2 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){
10      account.ShippingPostalCode = account.BillingPostalCode;
11    }
12  }
13 }
14 ClosedOpportunityTrigger.axpt:
15 trigger ClosedOpportunityTrigger on Opportunity (after
  insert,after update) {
16   List<Task> tasklist = new List<Task>();
17   for(Opportunity opp: Trigger.New){
18     if(opp.StageName == 'Closed Won'){
19       tasklist.add(new Task(Subject = 'Follow Up Test Task',WhatId =
        opp.Id));
20     }
21   }
22   if(tasklist.size() > 0){
23     insert tasklist;
24   }
25 }
26 APEX TESTING
27 VerifyData.apxc:
28 public class VerifyDate {
29   public static Date CheckDates(Date date1, Date date2) {
30     if(DateWithin30Days(date1,date2)) {
31       return date2;
32     } else {
33       return SetEndOfMonthDate(date1);
34     }
35   }
36   @TestVisible private static Boolean DateWithin30Days(Date date1,
    Date date2) {
```

```

37 //check for date2 being in the past
38 if( date2 < date1) { return false; }
39 SPSGP-15972-Salesforce Developer Catalyst
40 Self-Learning & Super Badges
41 2
42 APEX SPECIALIST SUPER BADGE CODES
43 //check that date2 is within (>=) 30 days of date1
44 Date date30Days = date1.addDays(30); //create a date 30 days away
    from date1
45 if( date2 >= date30Days ) { return false; }
46 else { return true; }
47 }
48 //method to return the end of the month of a given date
49 @TestVisible private static Date SetEndOfMonthDate(Date date1) {
50 Integer totalDays = Date.daysInMonth(date1.year(),
    date1.month());
51 Date lastDay = Date.newInstance(date1.year(), date1.month(),
    totalDays);
52 return lastDay;
53 }
54 }
55 TestVerifyData.apxc:
56 @isTest
57 private class TestVerifyDate {
58 @isTest static void Test_CheckDates_case1(){
59 Date D =
    VerifyDate.CheckDates(date.parse('01/01/2022'),date.parse('01/05/
60 System.assertEquals(date.parse('01/05/2022'), D);
61 }
62 @isTest static void Test_CheckDates_case2(){
63 Date D = VerifyDate.CheckDates(date.parse('01/01/2022'),
    date.parse('05/05/2022'));
64 System.assertEquals(date.parse('01/31/2022'), D);
65 }
66 @isTest static void Test_Within30Days_case1(){
67 Boolean flag =
    VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
68 date.parse('12/30/2021'));
69 System.assertEquals(false, flag);

```

```

70 }
71 @isTest static void Test_Within30Days_case2(){
72 Boolean flag =
    VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
73 date.parse('02/02/2021'));
74 System.assertEquals(false, flag);
75 }
76 @isTest static void Test_Within30Days_case3(){
77 SPSGP-15972-Salesforce Developer Catalyst
78 Self-Learning & Super Badges
79 3
80 APEX SPECIALIST SUPER BADGE CODES
81 Boolean flag =
    VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
82 date.parse('01/15/2022'));
83 System.assertEquals(true, flag);
84 }
85 @isTest static void Test_SetEndOfMonthDate(){
86 Date returndate =
    VerifyDate.SetEndOfMonthDate(date.parse('01/01/2022'));
87 }
88 }
89 RestrictContactByName.apxt:
90 trigger RestrictContactByName on Contact (before insert, before
    update) {
91 //check contacts prior to insert or update for invalid data
92 For (Contact c : Trigger.New) {
93 if(c.LastName == 'INVALIDNAME') { //invalidname is invalid
94 c.AddError('The Last Name "' + c.LastName + '" is not allowed for
95 }
96 }
97 }
98 TestRestrictContactByName.apxc:
99 @isTest
100 private class TestRestrictContactByName {
101 @isTest static void Test_insertupdateContact(){
102 Contact cnt = new Contact();
103 cnt.LastName = 'INVALIDNAME';
104 Test.startTest();

```

```

105 Database.SaveResult result = Database.insert(cnt,false);
106 Test.stopTest();
107 System.assert(!result.isSuccess());
108 System.assert(result.getErrors().size() > 0);
109 System.assertEquals('The Last Name "INVALIDNAME" is not allowed

110 result.getErrors()[0].getMessage());
111 }
112 }
113 SPSGP-15972-Salesforce Developer Catalyst
114 Self-Learning & Super Badges
115 4
116 APEX SPECIALIST SUPER BADGE CODES
117 RandomContactFactory.apxc:
118 public class RandomContactFactory {
119     public static List<Contact> generateRandomContacts(Integer
        num_cnts, string lastname) {
120         List<Contact> contacts = new List<Contact>();
121         for(Integer i = 0; i < num_cnts; i++) {
122             Contact cnt = new Contact(FirstName = 'Test' +i,LastName =
                lastname);
123             contacts.add(cnt);
124         }
125         return contacts;
126     }
127 }
128 ASYNCHRONOUS APEX
129 AccountProcessor.apxc:
130 public class AccountProcessor {
131     @future
132     public static void countContacts(List<Id> accountIds){
133         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();
139             accountsToUpdate.add(acc);
140         }

```

```

141 update accountsToUpdate;
142 }
143 }
144 AccountProcessorTest.apxc:
145 @isTest
146 public class AccountProcessorTest {
147     @isTest
148     private static void testCountContacts() {
149         Account newAccount = new Account(Name = 'Test Account');
150         insert newAccount;
151         Contact newContact1 = new Contact(FirstName = 'John', LastName =
            'Doe', AccountId =
152             SPSGP-15972-Salesforce Developer Catalyst
153             Self-Learning & Super Badges
154             5
155             APEX SPECIALIST SUPER BADGE CODES
156             newAccount.Id);
157         insert newContact1;
158         Contact newContact2 = new Contact(FirstName = 'John', LastName =
            'Doe', AccountId =
159             newAccount.Id);
160         insert newContact2;
161         List<Id> accountIds = new List<Id>();
162         accountIds.add(newAccount.Id);
163         Test.startTest();
164         AccountProcessor.countContacts(accountIds);
165         Test.stopTest();
166     }
167 }
168 LeadProcessor.apxc:
169 global class LeadProcessor implements
    Database.Batchable<sObject>{
170     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>
        L_list){

```

```

175 List<lead> L_list_new = 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-15972-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{
213 private Contact con;
214 private String state;

```

```

215 public AddPrimaryContact(Contact con, String state) {
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-15972-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');
250         insert testContact;

```

```

251 AddPrimaryContact addit = 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-15972-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() {
280 List<Lead> leads = new List<Lead>();
281 for(Integer i = 0; i < 200; i++) {
282 Lead l = new Lead(
283 FirstName = 'First' + i,
284 LastName = 'LastName',
285 Company = 'The Inc'
286 );
287 leads.add(l);

```



```

288 }
289 insert leads;
290 Test.startTest();
291 String jobId = System.schedule('ScheduledApexTest',CRON_EXP,new
    DailyLeadProcessor());
292 Test.stopTest();
293 List<Lead> checkleads = new List<Lead>();
294 checkleads = [Select Id From Lead Where LeadSource =
    'Dreamforce' and Company = 'The Inc'];
295 System.assertEquals(200,checkleads.size(),'Leads were not

296 }
297 }
298 APEX INTEGRATION SERVICES
299 AnimalLocator.apxc:
300 public class AnimalLocator{
301     public static String getAnimalNameById(Integer x){
302     Http http = new Http();
303     HttpRequest req = new HttpRequest();
304     req.setEndpoint('https://th-apex-http-

305     req.setMethod('GET');
306     Map<String, Object> animal= new Map<String, Object>();
307     HttpResponse res = http.send(req);
308     if (res.getStatusCode() == 200) {
309     SPSGP-15972-Salesforce Developer Catalyst
310     Self-Learning & Super Badges
311     9
312     APEX SPECIALIST SUPER BADGE CODES
313     Map<String, Object> results = (Map<String,
        Object>)JSON.deserializeUntyped(res.getBody());
314     animal = (Map<String, Object>) results.get('animal');
315     }
316     return (String)animal.get('name');
317     }
318     }
319 AnimalLocatorTest.apxc:
320 @isTest
321 private class AnimalLocatorTest{
322 @isTest static void AnimalLocatorMock1() {

```

```

323 Test.setMock(HttpCalloutMock.class, new AnimalLocatorMock());
324 String result = AnimalLocator.getAnimalNameById(3);
325 String expectedResult = 'chicken';
326 System.assertEquals(result, expectedResult );
327 }
328 }
329 AnimalLocatorMock.apxc:
330 @isTest
331 global class AnimalLocatorMock implements HttpCalloutMock {
332 // Implement this interface method
333 global HTTPResponse respond(HTTPRequest request) {
334 // Create a fake response
335 HTTPResponse response = new HTTPResponse();
336 response.setHeader('Content-Type', 'application/json');
337 response.setBody('{ "animals": ["majestic badger", "fluffy
    bunny", "scary bear", "chicken", "mighty
338 moose"]}');
339 response.setStatusCode(200);
340 return response;
341 }
342 }
343 ParkLocator.apxc:
344 public class ParkLocator {
345 public static String[] country(String theCountry) {
346 ParkService.ParksImplPort parkSvc = new
    ParkService.ParksImplPort(); // remove space
347 return parkSvc.byCountry(theCountry);
348 }
349 }
350 SPSGP-15972-Salesforce Developer Catalyst
351 Self-Learning & Super Badges
352 10
353 APEX SPECIALIST SUPER BADGE CODES
354 ParkLocatorTest.apxc:
355 @isTest
356 private class ParkLocatorTest {
357 @isTest static void testCallout() {
358 Test.setMock(WebServiceMock.class, new ParkServiceMock ());
359 String country = 'United States';
360 List<String> result = ParkLocator.country(country);

```

```

361 List<String> parks = new List<String>{'Yellowstone', 'Mackinac

362 System.assertEquals(parks, result);
363 }
364 }
365 ParkServiceMock.apxc:
366 @isTest
367 global class ParkServiceMock implements WebserviceMock {
368 global void doInvoke(
369 Object stub,
370 Object request,
371 Map<String, Object> response,
372 String endpoint,
373 String soapAction,
374 String requestName,
375 String responseNS,
376 String responseName,
377 String responseType) {
378 // start - specify the response you want to send
379 ParkService.byCountryResponse response_x = new
    ParkService.byCountryResponse();
380 response_x.return_x = new List<String>{'Yellowstone', 'Mackinac

381 // end
382 response.put('response_x', response_x);
383 }
384 }
385 AccountManager.apxc:
386 @RestResource(urlMapping='/Accounts/*/contacts')
387 global class AccountManager {
388 @HttpGet
389 global static Account getAccount() {
390 RestRequest req = RestContext.request;
391 String accId = req.requestURI.substringBetween('Accounts/',
    '/contacts');
392 SPSGP-15972-Salesforce Developer Catalyst
393 Self-Learning & Super Badges
394 11
395 APEX SPECIALIST SUPER BADGE CODES
396 Account acc = [SELECT Id, Name, (SELECT Id, Name FROM Contacts)

```

```

397 FROM Account WHERE Id = :accId];
398 return acc;
399 }
400 }
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-15972-Salesforce Developer Catalyst
431 Self-Learning & Super Badges
432 12
433 APEX SPECIALIST SUPER BADGE CODES
434 APEX SPECIALIST SUPER BADGE

```

```

435 Challenge-1
436 MaintenanceRequestHelper.apxc:
437 public with sharing class MaintenanceRequestHelper {
438     public static void updateWorkOrders(List<Case> updWorkOrders,
439         Map<Id,Case> nonUpdCaseMap) {
440         Set<Id> validIds = new Set<Id>();
441         For (Case c : updWorkOrders){
442             if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status ==
443                 'Closed'){
444                 if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
445                     validIds.add(c.Id);
446                 }
447             }
448             if (!validIds.isEmpty()){
449                 List<Case> newCases = new List<Case>();
450                 Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id,
451                     Vehicle__c, Equipment__c,
452                     Equipment__r.Maintenance_Cycle__c,(SELECT
453                         Id,Equipment__c,Quantity__c FROM
454                         Equipment_Maintenance_Items__r)
455                     FROM Case WHERE Id IN :validIds]);
456                 Map<Id,Decimal> maintenanceCycles = new Map<Id,Decimal>();
457                 AggregateResult[] results = [SELECT Maintenance_Request__c,
458                     MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
459                     Equipment_Maintenance_Item__c WHERE
460                     Maintenance_Request__c IN :ValidIds GROUP BY
461                     Maintenance_Request__c];
462                 for (AggregateResult ar : results){
463                     maintenanceCycles.put((Id) ar.get('Maintenance_Request__c'),
464                         (Decimal) ar.get('cycle'));
465                 }
466                 for(Case cc : closedCasesM.values()){
467                     Case nc = new Case (
468                         ParentId = cc.Id,
469                         Status = 'New',
470                         SPSGP-15972-Salesforce Developer Catalyst
471                         Self-Learning & Super Badges
472                     )
473                 }
474             }
475         }
476     }
477 }
478
479 APEX SPECIALIST SUPER BADGE CODES

```

```

468 Subject = 'Routine Maintenance',
469 Type = 'Routine Maintenance',
470 Vehicle__c = cc.Vehicle__c,
471 Equipment__c = cc.Equipment__c,
472 Origin = 'Web',
473 Date_Reported__c = Date.Today()
474 );
475 If (maintenanceCycles.containsKey(cc.Id)){
476 nc.Date_Due__c = Date.today().addDays((Integer)
    maintenanceCycles.get(cc.Id));
477 }
478 newCases.add(nc);
479 }
480 insert newCases;
481 List<Equipment_Maintenance_Item__c> clonedWPs = new
482 List<Equipment_Maintenance_Item__c>();
483 for (Case nc : newCases){
484 for (Equipment_Maintenance_Item__c wp :
485 closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items__r){
486 Equipment_Maintenance_Item__c wpClone = wp.clone();
487 wpClone.Maintenance_Request__c = nc.Id;
488 ClonedWPs.add(wpClone);
489 }
490 }
491 insert ClonedWPs;
492 }
493 }
494 }
495 SPSGP-15972-Salesforce Developer Catalyst
496 Self-Learning & Super Badges
497 14
498 APEX SPECIALIST SUPER BADGE CODES
499 MaintenanceRequest.apxt:
500 trigger MaintenanceRequest on Case (before update, after update)
    {
501 if (Trigger.isUpdate && Trigger.isAfter){
502 MaintenanceRequestHelper.updateWorkOrders(Trigger.New,
    Trigger.OldMap);
503 }
504 }

```

```

505 MaintenanceRequestHelperTest.apxc:
506 @istest
507 public with sharing 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-15972-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 =

```

```

541 equipmentId,
542 Maintenance_Request__c = requestId);
543 return wp;
544 }
545 @istest
546 private static void testMaintenanceRequestPositive(){
547 Vehicle__c vehicle = createVehicle();
548 insert vehicle;
549 id vehicleId = vehicle.Id;
550 Product2 equipment = createEq();
551 insert equipment;
552 id equipmentId = equipment.Id;
553 case somethingToUpdate =
    createMaintenanceRequest(vehicleId,equipmentId);
554 insert somethingToUpdate;
555 Equipment_Maintenance_Item__c workP =
    createWorkPart(equipmentId,somethingToUpdate.id);
556 insert workP;
557 test.startTest();
558 somethingToUpdate.status = CLOSED;
559 update somethingToUpdate;
560 test.stopTest();
561 Case newReq = [Select id, subject, type, Equipment__c,
    Date_Reported__c, Vehicle__c,
562 Date_Due__c
563 from case
564 where status =:STATUS_NEW];
565 SPSGP-15972-Salesforce Developer Catalyst
566 Self-Learning & Super Badges
567 16
568 APEX SPECIALIST SUPER BADGE CODES
569 Equipment_Maintenance_Item__c workPart = [select id
570 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());

```



```

578 }
579 @istest
580 private static void testMaintenanceRequestNegative(){
581     Vehicle__C vehicle = createVehicle();
582     insert vehicle;
583     id vehicleId = vehicle.Id;
584     Product2 equipment = createEq();
585     insert equipment;
586     id equipmentId = equipment.Id;
587     case emptyReq = createMaintenanceRequest(vehicleId,equipmentId);
588     insert emptyReq;
589     Equipment_Maintenance_Item__c workP =
        createWorkPart(equipmentId, emptyReq.Id);
590     insert workP;
591     test.startTest();
592     emptyReq.Status = WORKING;
593     update emptyReq;
594     test.stopTest();
595     list<case> allRequest = [select id
596     from case];
597     Equipment_Maintenance_Item__c workPart = [select id
598     from Equipment_Maintenance_Item__c
599     SPSGP-15972-Salesforce Developer Catalyst
600     Self-Learning & Super Badges
601     17
602     APEX SPECIALIST SUPER BADGE CODES
603     where Maintenance_Request__c = :emptyReq.Id];
604     system.assert(workPart != null);
605     system.assert(allRequest.size() == 1);
606 }
607 @istest
608 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++){
616         vehicleList.add(createVehicle());

```

```

617 equipmentList.add(createEq());
618 }
619 insert vehicleList;
620 insert equipmentList;
621 for(integer i = 0; i < 300; i++){
622 requestList.add(createMaintenanceRequest(vehicleList.get(i).id,
        equipmentList.get(i).id));
623 }
624 insert requestList;
625 for(integer i = 0; i < 300; i++){
626 workPartList.add(createWorkPart(equipmentList.get(i).id,
        requestList.get(i).id));
627 }
628 insert workPartList;
629 test.startTest();
630 for(case req : requestList){
631 req.Status = CLOSED;
632 oldRequestIds.add(req.Id);
633 }
634 update requestList;
635 SPSGP-15972-Salesforce Developer Catalyst
636 Self-Learning & Super Badges
637 18
638 APEX SPECIALIST SUPER BADGE CODES
639 test.stopTest();
640 list<case> allRequests = [select id
641 from case
642 where status =: STATUS_NEW];
643 list<Equipment_Maintenance_Item__c> workParts = [select id
644 from Equipment_Maintenance_Item__c
645 where Maintenance_Request__c in: oldRequestIds];
646 system.assert(allRequests.size() == 300);
647 }
648 }
649 Challenge-2
650 WarehouseCalloutService.apxc:
651 public with sharing class WarehouseCalloutService implements
    Queueable {
652 private static final String WAREHOUSE_URL = 'https://th-

```

```

653 //class that makes a REST callout to an external warehouse
    system to get a list of equipment that
654 needs to be updated.
655 //The callout's JSON response returns the equipment records that
    you upsert in Salesforce.
656 @future(callout=true)
657 public static void runWarehouseEquipmentSync(){
658     Http http = new Http();
659     HttpRequest request = new HttpRequest();
660     request.setEndpoint(WAREHOUSE_URL);
661     request.setMethod('GET');
662     HttpResponse response = http.send(request);
663     List<Product2> warehouseEq = new List<Product2>();
664     if (response.getStatusCode() == 200){
665         List<Object> jsonResponse =
            (List<Object>)JSON.deserializeUntyped(response.getBody());
666         SPSGP-15972-Salesforce Developer Catalyst
667         Self-Learning & Super Badges
668         19
669         APEX SPECIALIST SUPER BADGE CODES
670         System.debug(response.getBody());
671         //class maps the following fields: replacement part (always
            true), cost, current inventory,
672         lifespan, maintenance cycle, and warehouse SKU
673         //warehouse SKU will be external ID for identifying which
            equipment records to update within
674         Salesforce
675         for (Object eq : jsonResponse){
676             Map<String,Object> mapJson = (Map<String,Object>)eq;
677             Product2 myEq = new Product2();
678             myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
679             myEq.Name = (String) mapJson.get('name');
680             myEq.Maintenance_Cycle__c = (Integer)
                mapJson.get('maintenanceperiod');
681             myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');
682             myEq.Cost__c = (Integer) mapJson.get('cost');
683             myEq.Warehouse_SKU__c = (String) mapJson.get('sku');
684             myEq.Current_Inventory__c = (Double) mapJson.get('quantity');
685             myEq.ProductCode = (String) mapJson.get('_id');
686             warehouseEq.add(myEq);

```

```

687 }
688 if (warehouseEq.size() > 0){
689 upsert warehouseEq;
690 System.debug('Your equipment was synced with the warehouse

691 }
692 }
693 }
694 public static void execute (QueueableContext context){
695 runWarehouseEquipmentSync();
696 }
697 }
698 WarehouseCalloutServiceMock.apxc:
699 @isTest
700 global class WarehouseCalloutServiceMock implements
    HttpCalloutMock {
701 // implement http mock callout
702 global static HttpResponse respond(HttpRequest request) {
703 SPSGP-15972-Salesforce Developer Catalyst
704 Self-Learning & Super Badges
705 20
706 APEX SPECIALIST SUPER BADGE CODES
707 HttpResponse response = new HttpResponse();
708 response.setHeader('Content-Type', 'application/json');
709 response.setBody('["_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Gene
710 rator 1000
711 kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10
712 af742","replacement":true,"quantity":183,"name":"Cooling
713 Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
714 ","replacement":true,"quantity":143,"name":"Fuse
715 20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"
716 }]]');
716 response.setStatusCode(200);
717 return response;
718 }
719 }
720 WarehouseCalloutServiceTest.apxc:

```

```

721 @IsTest
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-15972-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 WarehouseSyncScheduleTest.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-15972-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;

```

```

791 }
792 PRIVATE STATIC Case createMaintenanceRequest(id vehicleId, id
    equipmentId){
793     case cs = new case(Type=REPAIR,
794     Status=STATUS_NEW,
795     Origin=REQUEST_ORIGIN,
796     Subject=REQUEST_SUBJECT,
797     Equipment__c=equipmentId,
798     Vehicle__c=vehicleId);
799     return cs;
800 }
801 PRIVATE STATIC Equipment_Maintenance_Item__c createWorkPart(id
    equipmentId,id requestId){
802     Equipment_Maintenance_Item__c wp = new
        Equipment_Maintenance_Item__c(Equipment__c =
803     equipmentId, Maintenance_Request__c = requestId);
804     return wp;
805 }
806 @istest
807 private static void testMaintenanceRequestPositive(){
808     Vehicle__c vehicle = createVehicle();
809     insert vehicle;
810     id vehicleId = vehicle.Id;
811     Product2 equipment = createEq();
812     insert equipment;
813     id equipmentId = equipment.Id;
814     SPSGP-15972-Salesforce Developer Catalyst
815     Self-Learning & Super Badges
816     23
817     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;
822     test.startTest();
823     somethingToUpdate.status = CLOSED;
824     update somethingToUpdate;
825     test.stopTest();

```

```

826 Case newReq = [Select id, subject, type, 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-15972-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
863 from Equipment_Maintenance_Item__c

```



```

864 where Maintenance_Request__c = :emptyReq.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-15972-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);
898 }
899 update requestList;
900 test.stopTest();
901 list<case> allRequests = [select id

```

```

902 from case
903 where status =: STATUS_NEW];
904 list<Equipment_Maintenance_Item__c> workParts = [select id
905 from Equipment_Maintenance_Item__c
906 where Maintenance_Request__c in: oldRequestIds];
907 system.assert(allRequests.size() == 300);
908 }
909 }
910 MaintenanceRequestHelper.apxc:
911 public with sharing class MaintenanceRequestHelper {
912 public static void updateWorkOrders(List<Case> updWorkOrders,
    Map<Id,Case> nonUpdCaseMap) {
913 Set<Id> validIds = new Set<Id>();
914 For (Case c : updWorkOrders){
915 if (nonUpdCaseMap.get(c.Id).Status != 'Closed' && c.Status ==
    'Closed'){
916 SPSGP-15972-Salesforce Developer Catalyst
917 Self-Learning & Super Badges
918 26
919 APEX SPECIALIST SUPER BADGE CODES
920 if (c.Type == 'Repair' || c.Type == 'Routine Maintenance'){
921 validIds.add(c.Id);
922 }
923 }
924 }
925 if (!validIds.isEmpty()){
926 List<Case> newCases = new List<Case>();
927 Map<Id,Case> closedCasesM = new Map<Id,Case>([SELECT Id,
    Vehicle__c, Equipment__c,
928 Equipment__r.Maintenance_Cycle__c,(SELECT
    Id,Equipment__c,Quantity__c FROM
929 Equipment_Maintenance_Items__r)
930 FROM Case WHERE Id IN :validIds]);
931 Map<Id,Decimal> maintenanceCycles = new Map<ID,Decimal>();
932 AggregateResult[] results = [SELECT Maintenance_Request__c,
933 MIN(Equipment__r.Maintenance_Cycle__c)cycle FROM
    Equipment_Maintenance_Item__c WHERE
934 Maintenance_Request__c IN :ValidIds GROUP BY
    Maintenance_Request__c];
935 for (AggregateResult ar : results){

```

```

936 maintenanceCycles.put((Id) ar.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-15972-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;
970 }
971 }
972 }
973 Challenge-5

```

```

974 WarehouseCalloutService.apxc:
975 public with sharing 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-15972-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){
1000 Map<String, Object> mapJson = (Map<String, Object>)eq;
1001 Product2 myEq = new Product2();
1002 myEq.Replacement_Part__c = (Boolean) mapJson.get('replacement');
1003 myEq.Name = (String) mapJson.get('name');
1004 myEq.Maintenance_Cycle__c = (Integer)
    mapJson.get('maintenanceperiod');
1005 myEq.Lifespan_Months__c = (Integer) mapJson.get('lifespan');

```

```

1006myEq.Cost__c = (Integer) mapJson.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-15972-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"
    }]]');

```

```

1040response.setStatusCode(200);
1041return response;
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","replacement":false,"quantity":5,"name":"Generator 1000
1052kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"10
1053af742","replacement":true,"quantity":183,"name":"Cooling
1054Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004
1055","_id":"55d66226726b611100aaf743
1056","replacement":true,"quantity":143,"name":"Fuse
105720A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"
1058SPSGP-15972-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());
1071}
1072}
1073WarehouseSyncScheduleTest.apxc:

```

```
1074@Test
1075public class WarehouseSyncScheduleTest {
1076@Test static void WarehousescheduleTest(){
1077String scheduleTime = '00 00 01 * * ?';
1078Test.startTest();
1079Test.setMock(HttpCalloutMock.class, new
    WarehouseCalloutServiceMock());
1080String jobID=System.schedule('Warehouse Time To Schedule to

1081WarehouseSyncSchedule());
1082Test.stopTest();
1083//Contains schedule information for a scheduled job. CronTrigger
    is similar to a cron job on UNIX
1084systems.
1085// This object is available in API version 17.0 and later.
1086CronTrigger a=[SELECT Id FROM CronTrigger where NextFireTime >
    today];
1087System.assertEquals(jobID, a.Id,'Schedule ');
1088}
1089}
```