APEX SPECIALIST SUPER BADGE CODES

APEX TRIGGERS

AccountAddressTrigger.axpt :

```
triggerAccountAddressTriggeronAccount(beforeinsert,beforeupdate){
for(Accountaccount:Trigger.New){
  if(account.Match_Billing_Address_c==True){
    account.ShippingPostalCode=account.BillingPostalCode;
  }
 }
}
```

ClosedOpportunityTrigger.axpt:

```
triggerClosedOpportunityTriggeronOpportunity(afterinsert,afterupdate){
List<Task>tasklist=newList<Task>();
for(Opportunityopp:Trigger.New){
  if(opp.StageName=='ClosedWon'){
    tasklist.add(newTask(Subject='FollowUpTestTask',WhatId=opp.Id));
  }
}
if(tasklist.size()>0){
  inserttasklist;
}
}
```

APEX TESTING

VerifyData.apxc:

```
publicclassVerifyDate{

    publicstaticDateCheckDates(Datedate1,Datedate2){
        if(DateWithin30Days(date1,date2)){
            returndate2;
        }else{
            returnSetEndOfMonthDate(date1);
        }
    }

    @TestVisibleprivatestaticBooleanDateWithin30Days(Datedate1,Datedate2){
        //checkfordate2beinginthepast
    if(date2<date1){returnfalse;}
```

APEX SPECIALIST SUPER BADGE CODES

```
        //checkthatdate2iswithin(>=)30daysofdate1
        Datedate30Days=date1.addDays(30);//createadate30daysawayfromdate1
            if(date2>=date30Days){returnfalse;}
            else{returntrue;}
        }

        //methodtoreturntheendofthemonthofagivendate
        @TestVisibleprivatestaticDateSetEndOfMonthDate(Datedate1){
            IntegertotalDays=Date.daysInMonth(date1.year(),date1.month());
            DatelastDay=Date.newInstance(date1.year(),date1.month(),totalDays);
            returnlastDay;
        }

}
```

TestVerifyData.apxc:

```
@isTest
privateclassTestVerifyDate{

  @isTeststaticvoidTest_CheckDates_case1(){
    DateD=VerifyDate.CheckDates(date.parse('01/01/2022'),date.parse('01/05/2022'));
    System.assertEquals(date.parse('01/05/2022'),D);
}
  @isTeststaticvoidTest_CheckDates_case2(){
    DateD=VerifyDate.CheckDates(date.parse('01/01/2022'),date.parse('05/05/2022'));
    System.assertEquals(date.parse('01/31/2022'),D);
  }
  @isTeststaticvoidTest_Within30Days_case1(){
    Booleanflag=VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
date.parse('12/30/2021'));
    System.assertEquals(false,flag);
  }
@isTeststaticvoidTest_Within30Days_case2(){
    Booleanflag=VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
date.parse('02/02/2021'));
    System.assertEquals(false,flag);
  }
@isTeststaticvoidTest_Within30Days_case3(){
```

APEX SPECIALIST SUPER BADGE CODES

```
    Booleanflag=VerifyDate.DateWithin30Days(date.parse('01/01/2022'),
date.parse('01/15/2022'));
    System.assertEquals(true,flag);
  }
  @isTeststaticvoidTest_SetEndOfMonthDate(){
    Datereturndate=VerifyDate.SetEndOfMonthDate(date.parse('01/01/2022'));
}
}
```

RestrictContactByName.apxt:

```
triggerRestrictContactByNameonContact(beforeinsert,beforeupdate){

        //checkcontactspriortoinsertorupdateforinvaliddata
        For(Contactc:Trigger.New){
            if(c.LastName=='INVALIDNAME'){ //invalidnameisinvalid
                c.AddError('TheLastName"'+c.LastName+'"isnotallowedforDML');
            }

        }
}
```

TestRestrictContactByName.apxc:

```
@isTest
privateclassTestRestrictContactByName{
  @isTeststaticvoidTest_insertupdateContact(){
    Contactcnt=newContact();
    cnt.LastName='INVALIDNAME';
    Test.startTest();
    Database.SaveResultresult=Database.insert(cnt,false);
    Test.stopTest();
    System.assert(!result.isSuccess());
    System.assert(result.getErrors().size()>0);
    System.assertEquals('TheLastName"INVALIDNAME"isnotallowedfor DML',
result.getErrors()[0].getMessage());
  }
}
```

APEX SPECIALIST SUPER BADGE CODES

RandomContactFactory.apxc :

```
publicclassRandomContactFactory{
   publicstaticList<Contact>generateRandomContacts(Integernum_cnts,stringlastname){
     List<Contact>contacts=newList<Contact>();
      for(Integeri=0;i<num_cnts;i++){
         Contactcnt=newContact(FirstName='Test'+i,LastName=lastname);
         contacts.add(cnt);
      }
       returncontacts;
              }
}
```

ASYNCHRONOUS APEX

AccountProcessor.apxc:

```
publicclassAccountProcessor{
              @future
   publicstaticvoidcountContacts(List<Id>accountIds){
      List<Account>accountsToUpdate=newList<Account>();

      List<Account>accounts=[SelectId,Name,(SelectIdfromContacts)fromAccountWhereIdin
:accountIds];
      For(Accountacc:accounts){
                                List<Contact>contactList=acc.contacts;
         acc.Number_Of_Contacts_c=contactList.size();
         accountsToUpdate.add(acc);
      }
      updateaccountsToUpdate;
   }
}
```

AccountProcessorTest.apxc:

```
@isTest
publicclassAccountProcessorTest{
              @isTest
   privatestaticvoidtestCountContacts(){
      AccountnewAccount=newAccount(Name='TestAccount');
      insertnewAccount;
      ContactnewContact1=newContact(FirstName='John',LastName='Doe',AccountId=
```

APEX SPECIALIST SUPER BADGE CODES

```
newAccount.Id);
        insertnewContact1;

        ContactnewContact2=newContact(FirstName='John',LastName='Doe',AccountId=
newAccount.Id);
        insertnewContact2;
        List<Id>accountIds=newList<Id>();
        accountIds.add(newAccount.Id);
        Test.startTest();
        AccountProcessor.countContacts(accountIds);
        Test.stopTest();
    }
}
```

LeadProcessor.apxc
:

```
globalclassLeadProcessorimplementsDatabase.Batchable<sObject>{
                globalIntegercount=0;

    globalDatabase.QueryLocatorstart(Database.BatchableContextbc){
    returnDatabase.getQueryLocator('SELECTID,LeadSourceFROMLead');
    }
    globalvoidexecute(Database.BatchableContextbc,List<Lead>L_list){
        List<lead>L_list_new=newList<lead>();
         for(leadL:L_list){
            L.leadSource='Dreamforce';
            L_list_new.add(L);
            count+=1;
        }
        updateL_list_new;
    }
    globalvoidfinish(Database.BatchableContextbc){
        system.debug('count='+count);
    }
}
```

LeadProcessorTest.apxc:

```
@isTest
publicclassLeadProcessorTest{
                @isTest
    publicstaticvoidtestit(){
```

APEX SPECIALIST SUPER BADGE CODES

```
        List<lead>L_list=newList<lead>();
        for(Integeri=0;i<200;i++){
            LeadL=newLead();
            L.LastName='name'+i;
            L.Company='Company';
            L.Status='RandomStatus';
            L_list.add(L);
        }
        insertL_list;
        Test.startTest();
        LeadProcessorlp=newLeadProcessor();
        IdbatchId=Database.executeBatch(lp);
        Test.stopTest();
    }
}
```

AddPrimaryContact.apxc

:

```
    publicclassAddPrimaryContactimplementsQueueable{
                privateContactcon;
    privateStringstate;
    publicAddPrimaryContact(Contactcon,Stringstate){
        this.con=con;
        this.state=state;
    }
    publicvoidexecute(QueueableContextcontext){
        List<Account>accounts=[SelectId,Name,(SelectFirstName,LastName,Idfromcontacts)
                        fromAccountwhereBillingState=:stateLimit200];
        List<Contact>primaryContacts=newList<Contact>();
        for(Accountacc:accounts){
            Contactc=con.clone();
            c.AccountId=acc.Id;
            primaryContacts.add(c);
        }
        if(primaryContacts.size()>0){
            insertprimaryContacts;
        }
    }
}
```

APEX SPECIALIST SUPER BADGE CODES

AddPrimaryContactTest.apxc

:

```
@isTest
publicclassAddPrimaryContactTest{
   statictestmethodvoidtestQueueable(){
      List<Account>testAccounts=newList<Account>();
      for(Integeri=0;i<50;i++){
         testAccounts.add(newAccount(Name='Account'+i,BillingState='CA'));
      }
      for(Integerj=0;j<50;j++){
         testAccounts.add(newAccount(Name='Account'+j,BillingState='NY'));
      }
      inserttestAccounts;
      ContacttestContact=newContact(FirstName='John',LastName='Doe');
      inserttestContact;
      AddPrimaryContactaddit=newAddPrimaryContact(testContact,'CA');
      Test.startTest();
      system.enqueueJob(addit);
      Test.stopTest();
      System.assertEquals(50,[Selectcount()fromContactwhereaccountIdin(SelectIdfrom
AccountwhereBillingState='CA')]);
   }
}
```

DailyLeadProcessor.apxc

:

```
globalclassDailyLeadProcessorimplementsSchedulable{
   globalvoidexecute(SchedulableContextctx){
      List<Lead>leadstoupdate=newList<Lead>();
      List<Lead>leads=[SelectidFromLeadWhereLeadSource=NULLLimit200];
      for(Leadl:leads){
         l.LeadSource='Dreamforce';
         leadstoupdate.add(l);
      }
      updateleadstoupdate;
   }
}
```

APEX SPECIALIST SUPER BADGE CODES

DailyLeadProcessorTest.apxc:

```
@isTest

  privateclassDailyLeadProcessorTest{
              publicstaticStringCRON_EXP='000153?2024';
   statictestmethodvoidtestScheduledJob(){
      List<Lead>leads=newList<Lead>();
      for(Integeri=0;i<200;i++){
         Leadl=newLead(
            FirstName='First'+i,
            LastName='LastName',
             Company='TheInc'
         );
         leads.add(l);
      }
      insertleads;
      Test.startTest();
      StringjobId=System.schedule('ScheduledApexTest',CRON_EXP,newDailyLeadProcessor());
               Test.stopTest();
      List<Lead>checkleads=newList<Lead>();
      checkleads=[SelectIdFromLeadWhereLeadSource='Dreamforce'andCompany='TheInc'];
      System.assertEquals(200,checkleads.size(),'Leadswerenotcreated');
   }
}
```

APEX INTEGRATION SERVICES

```
publicclassAnimalLocator{
```

AnimalLocator.apxc:

```
   publicstaticStringgetAnimalNameById(Integerx){
      Httphttp=newHttp();
      HttpRequestreq=newHttpRequest();
      req.setEndpoint('https://th-apex-http-callout.herokuapp.com/animals/'+x);
      req.setMethod('GET');
      Map<String,Object>animal=newMap<String,Object>();
      HttpResponseres=http.send(req);
           if(res.getStatusCode()==200){
```

APEX SPECIALIST SUPER BADGE CODES

```
      Map<String,Object>results=(Map<String,Object>)JSON.deserializeUntyped(res.getBody());
     animal=(Map<String,Object>)results.get('animal');
      }
return(String)animal.get('name');
   }
}
```

AnimalLocatorTest.apxc:

```
@isTest
privateclassAnimalLocatorTest{

   @isTeststaticvoidAnimalLocatorMock1(){
      Test.setMock(HttpCalloutMock.class,newAnimalLocatorMock());
      stringresult=AnimalLocator.getAnimalNameById(3);
      StringexpectedResult='chicken';
      System.assertEquals(result,expectedResult);
   }
}
```

AnimalLocatorMock.apxc:

```
@isTest
globalclassAnimalLocatorMockimplementsHttpCalloutMock{
    //Implementthisinterfacemethod
   globalHTTPResponserespond(HTTPRequestrequest){
      //Createafakeresponse
      HttpResponseresponse=newHttpResponse();
      response.setHeader('Content-Type','application/json');
      response.setBody('{"animals":["majesticbadger","fluffybunny","scarybear","chicken","mighty
moose"]}');
      response.setStatusCode(200);
      returnresponse;
   }
}
```

ParkLocator.apxc:

```
publicclassParkLocator{
   publicstaticstring[]country(stringtheCountry){
      ParkService.ParksImplPort parkSvc=new ParkService.ParksImplPort();//removespace
      returnparkSvc.byCountry(theCountry);
   }
}
```

APEX SPECIALIST SUPER BADGE CODES

ParkLocatorTest.apxc :

```
@isTest
privateclassParkLocatorTest{
   @isTeststaticvoidtestCallout(){
      Test.setMock(WebServiceMock.class,newParkServiceMock());
      Stringcountry='UnitedStates';
      List<String>result=ParkLocator.country(country);
      List<String>parks=newList<String>{'Yellowstone','MackinacNationalPark','Yosemite'};
       System.assertEquals(parks,result);
   }
}
```

ParkServiceMock.apxc:

```
@isTest
globalclassParkServiceMockimplementsWebServiceMock{
  globalvoiddoInvoke(
        Objectstub,
        Objectrequest,
        Map<String,Object>response,
        Stringendpoint,
        StringsoapAction,
        StringrequestName,
        StringresponseNS,
        StringresponseName,
        StringresponseType){
      //start-specifytheresponseyouwanttosend
      ParkService.byCountryResponseresponse_x=newParkService.byCountryResponse();
      response_x.return_x=newList<String>{'Yellowstone','MackinacNationalPark','Yosemite'};
      //end
      response.put('response_x',response_x);
  }
}
```

AccountManager.apxc:

```
@RestResource(urlMapping='/Accounts/*/contacts')
globalclassAccountManager{
   @HttpGet
   globalstaticAccountgetAccount(){
      RestRequestreq=RestContext.request;
      StringaccId=req.requestURI.substringBetween('Accounts/','/contacts');
```

APEX SPECIALIST SUPER BADGE CODES

```
    Accountacc=[SELECTId,Name,(SELECTId,NameFROMContacts)
            FROMAccountWHEREId=:accId];
    returnacc;
  }
}
```

AccountManagerTest.apxc:

```
@isTest
privateclassAccountManagerTest{

  privatestatictestMethodvoidgetAccountTest1(){
    IdrecordId=createTestRecord();
    //Setupatestrequest
    RestRequestrequest=newRestRequest();
    request.requestUri='https://na1.salesforce.com/services/apexrest/Accounts/'+recordId
+'/contacts';
    request.httpMethod='GET';
    RestContext.request=request;
    //Callthemethodtotest
    AccountthisAccount=AccountManager.getAccount();
    //Verifyresults System.assert(thisAccount!=null);
    System.assertEquals('Testrecord',thisAccount.Name);

  }

  //Helpermethod
    staticIdcreateTestRecord(){
    //Createtestrecord
    AccountTestAcc=newAccount(
     Name='Testrecord');
    insertTestAcc;
    ContactTestCon=newContact(
    LastName='Test',
    AccountId=TestAcc.id);
    returnTestAcc.Id;
  }
}
```

APEX SPECIALIST SUPER BADGE CODES

APEX SPECIALIST SUPER BADGE

Challenge - 1

MaintenanceRequestHelper.apxc:

```
publicwithsharingclassMaintenanceRequestHelper{
   publicstaticvoidupdateworkOrders(List<Case>updWorkOrders,Map<Id,Case>nonUpdCaseMap){
      Set<Id>validIds=newSet<Id>();


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



           }
        }
      }

      if(!validIds.isEmpty()){
         List<Case>newCases=newList<Case>();
         Map<Id,Case>closedCasesM=newMap<Id,Case>([SELECTId,Vehicle_c,Equipment_c,
Equipment_r.Maintenance_Cycle_c,(SELECTId,Equipment_c,Quantity_cFROM
Equipment_Maintenance_Items_r)
                                          FROMCaseWHEREIdIN:validIds]);
         Map<Id,Decimal>maintenanceCycles=newMap<ID,Decimal>();
         AggregateResult[]results=[SELECTMaintenance_Request_c,
MIN(Equipment_r.Maintenance_Cycle_c)cycleFROMEquipment_Maintenance_Item_cWHERE
Maintenance_Request_cIN:ValidIdsGROUPBYMaintenance_Request_c];

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

         for(Casecc:closedCasesM.values()){
            Casenc=newCase(
               ParentId=cc.Id,
            Status='New',
```

APEX SPECIALIST SUPER BADGE CODES

```
            Subject='RoutineMaintenance',
            Type='RoutineMaintenance',
            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);
      }

      insertnewCases;

      List<Equipment_Maintenance_Item_c>clonedWPs=new
List<Equipment_Maintenance_Item_c>();
      for(Casenc:newCases){
        for(Equipment_Maintenance_Item_cwp:
closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items_r){
            Equipment_Maintenance_Item_cwpClone=wp.clone();
            wpClone.Maintenance_Request_c=nc.Id;
            ClonedWPs.add(wpClone);

        }
      }
      insertClonedWPs;
    }
  }
}
```

APEX SPECIALIST SUPER BADGE CODES

MaintenanceRequest.apxt :

```
triggerMaintenanceRequestonCase(beforeupdate,afterupdate){
   if(Trigger.isUpdate&&Trigger.isAfter){
      MaintenanceRequestHelper.updateWorkOrders(Trigger.New,Trigger.OldMap);
   }
}
```

MaintenanceRequestHelperTest.apxc:

```
@istest

publicwithsharingclassMaintenanceRequestHelperTest{

   privatestaticfinalstringSTATUS_NEW='New';
   privatestaticfinalstringWORKING='Working';
   privatestaticfinalstringCLOSED='Closed';
   privatestaticfinalstringREPAIR='Repair';
   privatestaticfinalstringREQUEST_ORIGIN='Web';
   privatestaticfinalstringREQUEST_TYPE='RoutineMaintenance';
   privatestaticfinalstringREQUEST_SUBJECT='Testingsubject';

   PRIVATESTATICVehicle_ccreateVehicle(){
      Vehicle_cVehicle=newVehicle_C(name='SuperTruck');
      returnVehicle;
   }

   PRIVATESTATICProduct2createEq(){
      product2equipment=newproduct2(name='SuperEquipment',
                                    lifespan_months_C=10,
                                    maintenance_cycle_C=10,
                                    replacement_part_c=true);
      returnequipment;
   }

   PRIVATESTATICCasecreateMaintenanceRequest(idvehicleId,idequipmentId){
      casecs=newcase(Type=REPAIR,
                     Status=STATUS_NEW,
                     Origin=REQUEST_ORIGIN,
                     Subject=REQUEST_SUBJECT,
                     Equipment_c=equipmentId,
```

APEX SPECIALIST SUPER BADGE CODES

```
                Vehicle_c=vehicleId);
    returncs;
  }

  PRIVATESTATICEquipment_Maintenance_Item_ccreateWorkPart(idequipmentId,idrequestId){
    Equipment_Maintenance_Item_cwp=newEquipment_Maintenance_Item_c(Equipment_c=
equipmentId,
                                                Maintenance_Request_c=requestId);
    returnwp;
  }



  @istest
  privatestaticvoidtestMaintenanceRequestPositive(){
    Vehicle_cvehicle=createVehicle();
    insertvehicle;
    idvehicleId=vehicle.Id;

    Product2equipment=createEq();
    insertequipment;
    idequipmentId=equipment.Id;

    casesomethingToUpdate=createMaintenanceRequest(vehicleId,equipmentId);
    insertsomethingToUpdate;

    Equipment_Maintenance_Item_cworkP=createWorkPart(equipmentId,somethingToUpdate.id);
    insertworkP;

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

    CasenewReq=[Selectid,subject,type,Equipment_c,Date_Reported_c,Vehicle_c, Date_Due
c
            fromcase
            wherestatus=:STATUS_NEW];
```

APEX SPECIALIST SUPER BADGE CODES

```
    Equipment_Maintenance_Item_cworkPart=[selectid
                                         fromEquipment_Maintenance_Item_c
                                         whereMaintenance_Request_c=:newReq.Id];

    system.assert(workPart!=null); system.assert(newReq.Subject!=null);
    system.assertEquals(newReq.Type,REQUEST_TYPE);
    SYSTEM.assertEquals(newReq.Equipment_c,equipmentId);
    SYSTEM.assertEquals(newReq.Vehicle_c,vehicleId);
    SYSTEM.assertEquals(newReq.Date_Reported_c,system.today());
}

@istest
privatestaticvoidtestMaintenanceRequestNegative(){
    Vehicle_Cvehicle=createVehicle();
    insertvehicle;
    idvehicleId=vehicle.Id;

    product2equipment=createEq();
    insertequipment;
    idequipmentId=equipment.Id;

    caseemptyReq=createMaintenanceRequest(vehicleId,equipmentId);
    insertemptyReq;

    Equipment_Maintenance_Item_cworkP=createWorkPart(equipmentId,emptyReq.Id);
    insertworkP;

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

    list<case>allRequest=[selectid
                          fromcase];

    Equipment_Maintenance_Item_cworkPart=[selectid
                                         fromEquipment_Maintenance_Item_c
```

APEX SPECIALIST SUPER BADGE CODES

```
whereMaintenance_Request_c=:emptyReq.Id];

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

  @istest
  privatestaticvoidtestMaintenanceRequestBulk(){ list<Vehicle
    C>vehicleList=newlist<Vehicle_C>();
    list<Product2>equipmentList=newlist<Product2>();
    list<Equipment_Maintenance_Item_c>workPartList=new
list<Equipment_Maintenance_Item_c>();
    list<case>requestList=newlist<case>();
    list<id>oldRequestIds=newlist<id>();

    for(integeri=0;i<300;i++){
      vehicleList.add(createVehicle());
      equipmentList.add(createEq());
    }
    insertvehicleList;
    insertequipmentList;

    for(integeri=0;i<300;i++){
       requestList.add(createMaintenanceRequest(vehicleList.get(i).id,equipmentList.get(i).id));
    }
    insertrequestList;

    for(integeri=0;i<300;i++){
       workPartList.add(createWorkPart(equipmentList.get(i).id,requestList.get(i).id));
    }
    insertworkPartList;

    test.startTest();
    for(casereq:requestList){
       req.Status=CLOSED;
       oldRequestIds.add(req.Id);
    }
    updaterequestList;
```

APEX SPECIALIST SUPER BADGE CODES

```
    test.stopTest();

    list<case>allRequests=[selectid
                    fromcase
                    wherestatus=:STATUS_NEW];

    list<Equipment_Maintenance_Item_c>workParts=[selectid
                                fromEquipment_Maintenance_Item_c
                                whereMaintenance_Request_cin:oldRequestIds];

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

Challeng - 2

WarehouseCalloutSer vice.apxc:

```
publicwithsharingclassWarehouseCalloutServiceimplementsQueueable{
  privatestaticfinalStringWAREHOUSE_URL='https://th-superbadge-
apex.herokuapp.com/equipment';

  //classthatmakesaRESTcallouttoanexternalwarehousesystemtogetalistofequipmentthat
needstobeupdated.
  //Thecallout'sJSONresponsereturnstheequipmentrecordsthatyouupsertinSalesforce.

  @future(callout=true)
  publicstaticvoidrunWarehouseEquipmentSync(){
    Httphttp=newHttp();
    HttpRequestrequest=newHttpRequest();

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

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

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

APEX SPECIALIST SUPER BADGE CODES

```
        System.debug(response.getBody());

        //classmapsthefollowingfields:replacementpart(alwaystrue),cost,currentinventory,
lifespan,maintenancecycle,andwarehouseSKU
        //warehouseSKUwillbeexternalIDforidentifyingwhichequipmentrecordstoupdatewithin
Salesforce
        for(Objecteq:jsonResponse){
            Map<String,Object>mapJson=(Map<String,Object>)eq;
            Product2myEq=newProduct2();
            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');
            warehouseEq.add(myEq);
        }

        if(warehouseEq.size()>0){
            upsertwarehouseEq;
            System.debug('Yourequipmentwassyncedwiththewarehouseone');
        }
      }
   }

   publicstaticvoidexecute(QueueableContextcontext){
      runWarehouseEquipmentSync();
   }

}
```

WarehouseCalloutSer viceMock.apxc:

```
@isTest
globalclassWarehouseCalloutServiceMockimplementsHttpCalloutMock{
   //implementhttpmockcallout
   globalstaticHttpResponserespond(HttpRequestrequest){
```

APEX SPECIALIST SUPER BADGE CODES

```
        HttpResponseresponse=newHttpResponse();
        response.setHeader('Content-Type','application/json');

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Gene
rator                                    1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226726b611100
a af742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b611100aaf74
3 ","replacement":true,"quantity":143,"name":"Fuse
20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
        response.setStatusCode(200);
        returnresponse;
    }
}
```

WarehouseCalloutSer viceTest.apxc:

```
@IsTest                                                    ___
privateclassWarehouseCalloutServiceTest{
    //implementyourmockcallouttesthere
            @isTest
    staticvoidtestWarehouseCallout(){
        test.startTest();
        test.setMock(HttpCalloutMock.class,newWarehouseCalloutServiceMock());
        WarehouseCalloutService.execute(null);
        test.stopTest();

        List<Product2>product2List=newList<Product2>();
        product2List=[SELECTProductCodeFROMProduct2];

        System.assertEquals(3,product2List.size());
        System.assertEquals('55d66226726b611100aaf741',product2List.get(0).ProductCode);
        System.assertEquals('55d66226726b611100aaf742',product2List.get(1).ProductCode);
        System.assertEquals('55d66226726b611100aaf743',product2List.get(2).ProductCode);
    }
}
```

Challeng - 3

WarehouseSyncSchedule.apxc :

```
globalwithsharingclassWarehouseSyncScheduleimplementsSchedulable{
```

APEX SPECIALIST SUPER BADGE CODES

```
    globalvoidexecute(SchedulableContextctx){
        System.enqueueJob(newWarehouseCalloutService());
    }
}
```

WarehouseSyncScheduuleTest.apxc:

```
@isTest
publicclassWarehouseSyncScheduleTest{

    @isTeststaticvoidWarehousescheduleTest(){
        StringscheduleTime='000001**?';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class,newWarehouseCalloutServiceMock());
        StringjobID=System.schedule('WarehouseTimeToScheduletoTest',scheduleTime,new
WarehouseSyncSchedule());
        Test.stopTest();
        //ContainsscheduleinformationforascheduledjobCronTriggerissimilartoacronjobonUNIX
systems.
        //ThisobjectisavailableinAPIversion17.0andlater.
        CronTriggera=[SELECTIdFROMCronTriggerwhereNextFireTime>today];
        System.assertEquals(jobID,a.Id,'Schedule');


    }
}
```

Challeng - 4

MaintenanceRequestHelperTest.apxc:

```
@istest

publicwithsharingclassMaintenanceRequestHelperTest{

    privatestaticfinalstringSTATUS_NEW='New';
    privatestaticfinalstringWORKING='Working';
    privatestaticfinalstringCLOSED='Closed';
    privatestaticfinalstringREPAIR='Repair';
    privatestaticfinalstringREQUEST_ORIGIN='Web';
    privatestaticfinalstringREQUEST_TYPE='RoutineMaintenance';
    privatestaticfinalstringREQUEST_SUBJECT='Testingsubject';

    PRIVATESTATICVehicle_ccreateVehicle(){
```

APEX SPECIALIST SUPER BADGE CODES

```
    Vehicle_cVehicle=newVehicle_C(name='SuperTruck');
    returnVehicle;
  }

  PRIVATESTATICProduct2createEq(){
    product2equipment=newproduct2(name='SuperEquipment',
                          lifespan_months_C=10,
                          maintenance_cycle_C=10,
                          replacement_part_c=true);
    returnequipment;
  }

  PRIVATESTATICCasecreateMaintenanceRequest(idvehicleId,idequipmentId){
    casecs=newcase(Type=REPAIR,
                Status=STATUS_NEW,
                Origin=REQUEST_ORIGIN,
                Subject=REQUEST_SUBJECT,
                Equipment_c=equipmentId,
                Vehicle_c=vehicleId);
    returncs;
  }

  PRIVATESTATICEquipment_Maintenance_Item_ccreateWorkPart(idequipmentId,idrequestId){
    Equipment_Maintenance_Item_cwp=newEquipment_Maintenance_Item_c(Equipment_c=
equipmentId,Maintenance_Request_c=requestId);
    returnwp;
  }


  @istest
  privatestaticvoidtestMaintenanceRequestPositive(){
    Vehicle_cvehicle=createVehicle();
    insertvehicle;
    idvehicleId=vehicle.Id;

    Product2equipment=createEq();
    insertequipment;
    idequipmentId=equipment.Id;
```

APEX SPECIALIST SUPER BADGE CODES

```
    casesomethingToUpdate=createMaintenanceRequest(vehicleId,equipmentId);
    insertsomethingToUpdate;

    Equipment_Maintenance_Item_cworkP=createWorkPart(equipmentId,somethingToUpdate.id);
    insertworkP;

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

    CasenewReq=[Selectid,subject,type,Equipment_c,Date_Reported_c,Vehicle_c, Date_Due
c
            fromcase
            wherestatus=:STATUS_NEW];

    Equipment_Maintenance_Item_cworkPart=[selectid
                        fromEquipment_Maintenance_Item_c
                        whereMaintenance_Request_c=:newReq.Id];

    system.assert(workPart!=null); system.assert(newReq.Subject!=null);
    system.assertEquals(newReq.Type,REQUEST_TYPE);
    SYSTEM.assertEquals(newReq.Equipment_c,equipmentId);
    SYSTEM.assertEquals(newReq.Vehicle_c,vehicleId);
    SYSTEM.assertEquals(newReq.Date_Reported_c,system.today());
  }

  @istest
  privatestaticvoidtestMaintenanceRequestNegative(){
    Vehicle_Cvehicle=createVehicle();
    insertvehicle;
    idvehicleId=vehicle.Id;

    product2equipment=createEq();
    insertequipment;
    idequipmentId=equipment.Id;
```

APEX SPECIALIST SUPER BADGE CODES

```
    caseemptyReq=createMaintenanceRequest(vehicleId,equipmentId);
    insertemptyReq;

    Equipment_Maintenance_Item_cworkP=createWorkPart(equipmentId,emptyReq.Id);
    insertworkP;

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

    list<case>allRequest=[selectid
                fromcase];

    Equipment_Maintenance_Item_cworkPart=[selectid
                        fromEquipment_Maintenance_Item_c
                        whereMaintenance_Request_c=:emptyReq.Id];

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

  @istest
  privatestaticvoidtestMaintenanceRequestBulk(){ list<Vehicle
    C>vehicleList=newlist<Vehicle_C>();
    list<Product2>equipmentList=newlist<Product2>();
    list<Equipment_Maintenance_Item_c>workPartList=new
list<Equipment_Maintenance_Item_c>();
    list<case>requestList=newlist<case>();
    list<id>oldRequestIds=newlist<id>();

    for(integeri=0;i<300;i++){
      vehicleList.add(createVehicle());
      equipmentList.add(createEq());
    }
    insertvehicleList;
    insertequipmentList;
```

APEX SPECIALIST SUPER BADGE CODES

```
    for(integeri=0;i<300;i++){
        requestList.add(createMaintenanceRequest(vehicleList.get(i).id,equipmentList.get(i).id));
    }
    insertrequestList;

    for(integeri=0;i<300;i++){
      workPartList.add(createWorkPart(equipmentList.get(i).id,requestList.get(i).id));
    }
    insertworkPartList;

    test.startTest();
    for(casereq:requestList){
      req.Status=CLOSED;
      oldRequestIds.add(req.Id);
    }
    updaterequestList;
    test.stopTest();

    list<case>allRequests=[selectid
                           fromcase
                           wherestatus=:STATUS_NEW];

    list<Equipment_Maintenance_Item_c>workParts=[selectid
                                        fromEquipment_Maintenance_Item_c
                                        whereMaintenance_Request_cin:oldRequestIds];

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

MaintenanceRequestHelper.apxc:

```
publicwithsharingclassMaintenanceRequestHelper{
  publicstaticvoidupdateworkOrders(List<Case>updWorkOrders,Map<Id,Case>nonUpdCaseMap){
    Set<Id>validIds=newSet<Id>();


    For(Casec:updWorkOrders){
      if(nonUpdCaseMap.get(c.Id).Status!='Closed'&&c.Status=='Closed'){
```

APEX SPECIALIST SUPER BADGE CODES

```
            if(c.Type=='Repair'||c.Type=='RoutineMaintenance'){
                validIds.add(c.Id);


            }
        }
    }

    if(!validIds.isEmpty()){
        List<Case>newCases=newList<Case>();
        Map<Id,Case>closedCasesM=newMap<Id,Case>([SELECTId,Vehicle_c,Equipment_c,
Equipment_r.Maintenance_Cycle_c,(SELECTId,Equipment_c,Quantity_cFROM
Equipment_Maintenance_Items_r)
                                        FROMCaseWHEREIdIN:validIds]);
        Map<Id,Decimal>maintenanceCycles=newMap<ID,Decimal>();
        AggregateResult[]results=[SELECTMaintenance_Request_c,
MIN(Equipment_r.Maintenance_Cycle_c)cycleFROMEquipment_Maintenance_Item_cWHERE
Maintenance_Request_cIN:ValidIdsGROUPBYMaintenance_Request_c];

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

        for(Casecc:closedCasesM.values()){
            Casenc=newCase(
                ParentId=cc.Id,
            Status='New',
                Subject='RoutineMaintenance',
                Type='RoutineMaintenance',
                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));
```

APEX SPECIALIST SUPER BADGE CODES

```
            }

            newCases.add(nc);
        }

        insertnewCases;

        List<Equipment_Maintenance_Item_c>clonedWPs=new
List<Equipment_Maintenance_Item_c>();
        for(Casenc:newCases){
            for(Equipment_Maintenance_Item_cwp:
closedCasesM.get(nc.ParentId).Equipment_Maintenance_Items_r){
                Equipment_Maintenance_Item_cwpClone=wp.clone();
                wpClone.Maintenance_Request_c=nc.Id;
                ClonedWPs.add(wpClone);

            }
        }
        insertClonedWPs;
    }
  }
}
```

Challeng -5

WarehouseCalloutSer vice.apxc:

```
publicwithsharingclassWarehouseCalloutServiceimplementsQueueable{
  privatestaticfinalStringWAREHOUSE_URL='https://th-superbadge-
apex.herokuapp.com/equipment';

  //classthatmakesaRESTcallouttoanexternalwarehousesystemtogetalistofequipmentthat
needstobeupdated.
  //Thecallout'sJSONresponsereturnstheequipmentrecordsthatyouupsertinSalesforce.

  @future(callout=true)
  publicstaticvoidrunWarehouseEquipmentSync(){
    Httphttp=newHttp();
    HttpRequestrequest=newHttpRequest();

    request.setEndpoint(WAREHOUSE_URL);
```

APEX SPECIALIST SUPER BADGE CODES

```
        request.setMethod('GET');
        HttpResponseresponse=http.send(request);

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

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

            //classmapsthefollowingfields:replacementpart(alwaystrue),cost,currentinventory,
lifespan,maintenancecycle,andwarehouseSKU
            //warehouseSKUwillbeexternalIDforidentifyingwhichequipmentrecordstoupdatewithin
Salesforce
            for(Objecteq:jsonResponse){
                Map<String,Object>mapJson=(Map<String,Object>)eq;
                Product2myEq=newProduct2();
                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');
                warehouseEq.add(myEq);
            }

            if(warehouseEq.size()>0){
                upsertwarehouseEq;
                System.debug('Yourequipmentwassyncedwiththewarehouseone');
            }
        }
    }

    publicstaticvoidexecute(QueueableContextcontext){
        runWarehouseEquipmentSync();
    }
```

APEX SPECIALIST SUPER BADGE CODES

```
}
```

WarehouseCalloutSer viceMock.apxc:

```
@isTest
globalclassWarehouseCalloutServiceMockimplementsHttpCalloutMock{
   //implementhttpmockcallout
      globalstaticHttpResponserespond(HttpRequestrequest){

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

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Gene
rator1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226726b611100
a af742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b611100aaf74
3 ","replacement":true,"quantity":143,"name":"Fuse
20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
      response.setStatusCode(200);
      returnresponse;
   }
}
```

WarehouseCalloutSer viceTest.apxc:

```
@isTest
globalclassWarehouseCalloutServiceMockimplementsHttpCalloutMock{
   //implementhttpmockcallout
      globalstaticHttpResponserespond(HttpRequestrequest){

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

response.setBody('[{"_id":"55d66226726b611100aaf741","replacement":false,"quantity":5,"name":"Gene
rator1000
kW","maintenanceperiod":365,"lifespan":120,"cost":5000,"sku":"100003"},{"_id":"55d66226726b611100
a af742","replacement":true,"quantity":183,"name":"Cooling
Fan","maintenanceperiod":0,"lifespan":0,"cost":300,"sku":"100004"},{"_id":"55d66226726b611100aaf74
3 ","replacement":true,"quantity":143,"name":"Fuse
20A","maintenanceperiod":0,"lifespan":0,"cost":22,"sku":"100005"}]');
```

APEX SPECIALIST SUPER BADGE CODES

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

Challeng -6

WarehouseSyncSchedule.apxc:

```
globalwithsharingclassWarehouseSyncScheduleimplementsSchedulable{
    globalvoidexecute(SchedulableContextctx){
        System.enqueueJob(newWarehouseCalloutService());
    }
}
```

WarehouseSyncScheduleTest.apxc:

```
@isTest
publicclassWarehouseSyncScheduleTest{

    @isTeststaticvoidWarehousescheduleTest(){
        StringscheduleTime='000001**?';
        Test.startTest();
        Test.setMock(HttpCalloutMock.class,newWarehouseCalloutServiceMock());
        StringjobID=System.schedule('WarehouseTimeToScheduletoTest',scheduleTime,new
WarehouseSyncSchedule());
        Test.stopTest();
        //Containsscheduleinformationforascheduledjob.CronTriggerissimilartoacronjobonUNIX
systems.
        //ThisobjectisavailableinAPIversion17.0andlater.
        CronTriggera=[SELECTIdFROMCronTriggerwhereNextFireTime>today];
        System.assertEquals(jobID,a.Id,'Schedule');


    }
}
```