Premium Learnings

WISDOM.ON DEMAND

APEX TRIGGER EXAMPLES

www.premiumlearnings.com

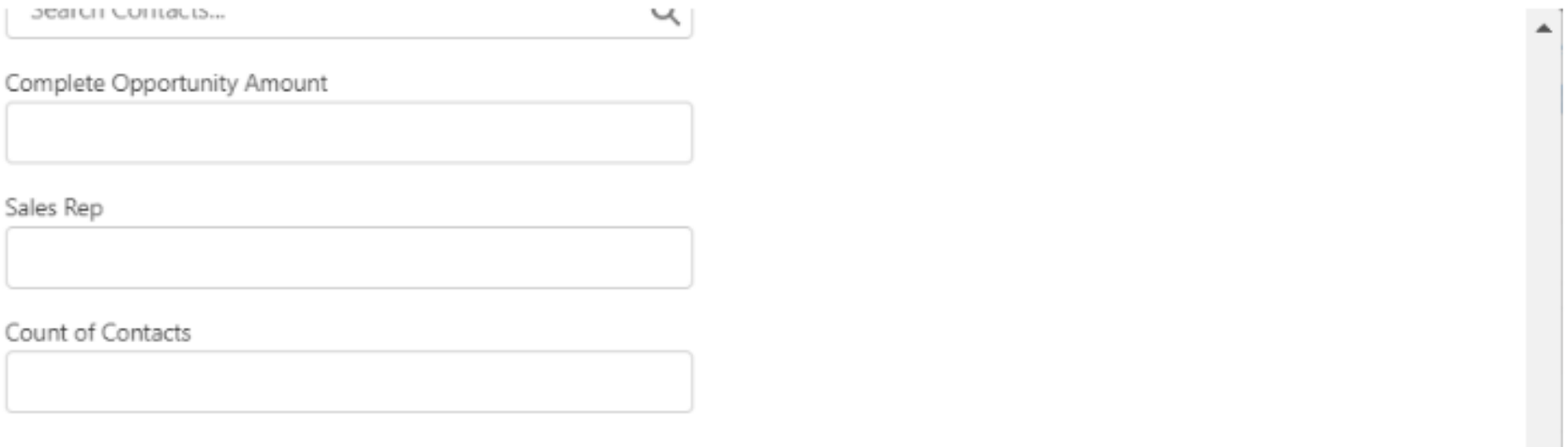
Example 1

Write a trigger on Account, when an account is inserted, automatically account billing address should populate into the account shipping address.

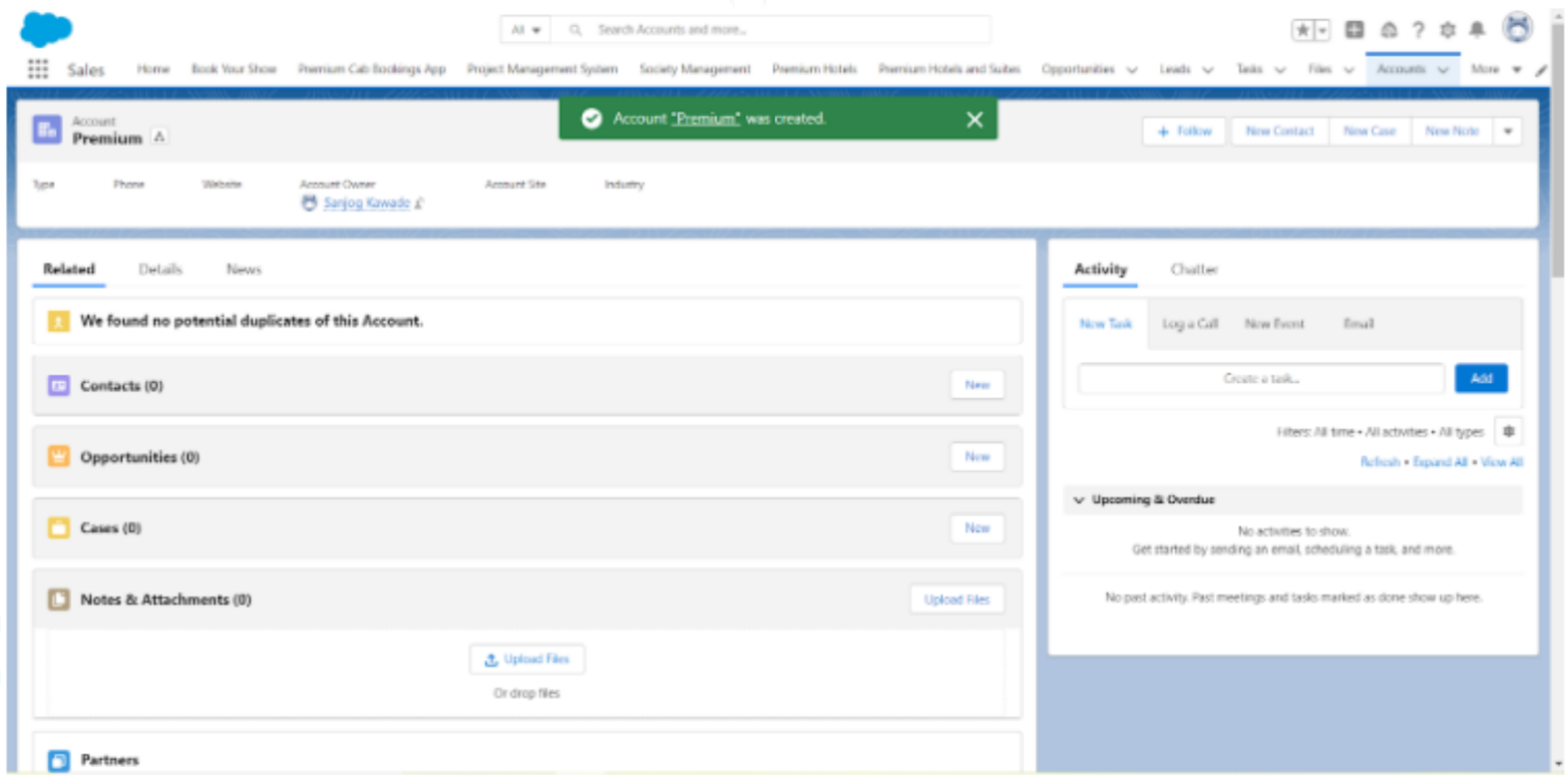
```
trigger changeshippingaddress on Account (before insert){

 for(Account a : Trigger.new){

//We should first check whether the user is passing
//null data

        if(a.BillingStreet!=null){
         a.ShippingStreet=a.BillingStreet;
          }

         if(a.BillingCity!=null){
         a.ShippingCity=a.BillingCity;
          }

        if(a.BillingState!=null){
        a.ShippingState=a.BillingState;
          }
        if(a.BillingPostalCode!=null){
        a.ShippingPostalCode=a.BillingPostalCode;
          }

        if(a.BillingCountry!=null){
        a.ShippingCountry=a.BillingCountry;

        }
    }
}
```

Test Case:

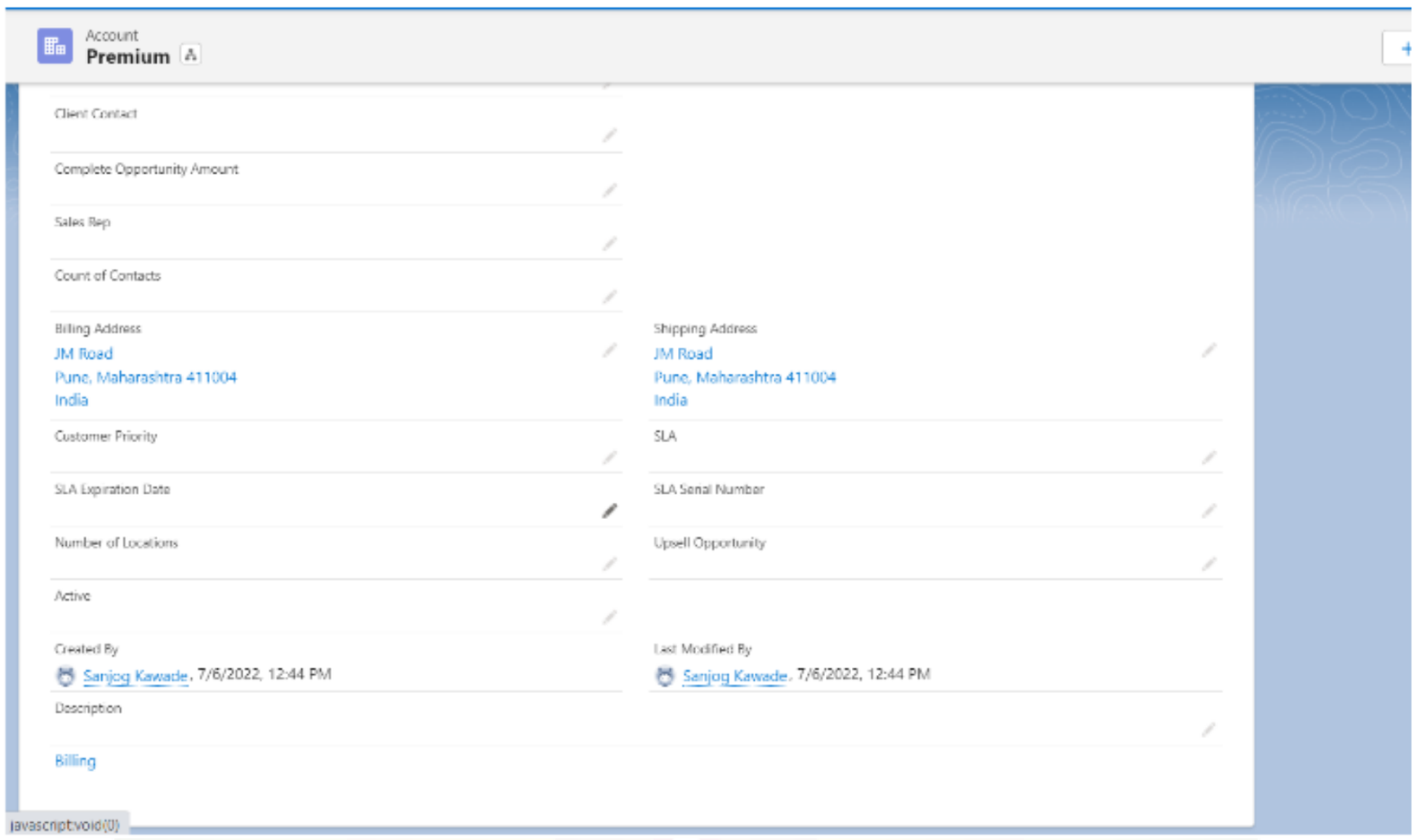
1. Go to Accounts and create a new account. Populate the billing address fields as shown in the below image and leave the shipping address fields blank as shown in the below image. Click save.



2. The record is now saved as shown below image. Click on the details page and check the shipping address which we kept blank while creating the record.



3. The account billing address we entered is populated in the shipping address as well.



Example 2

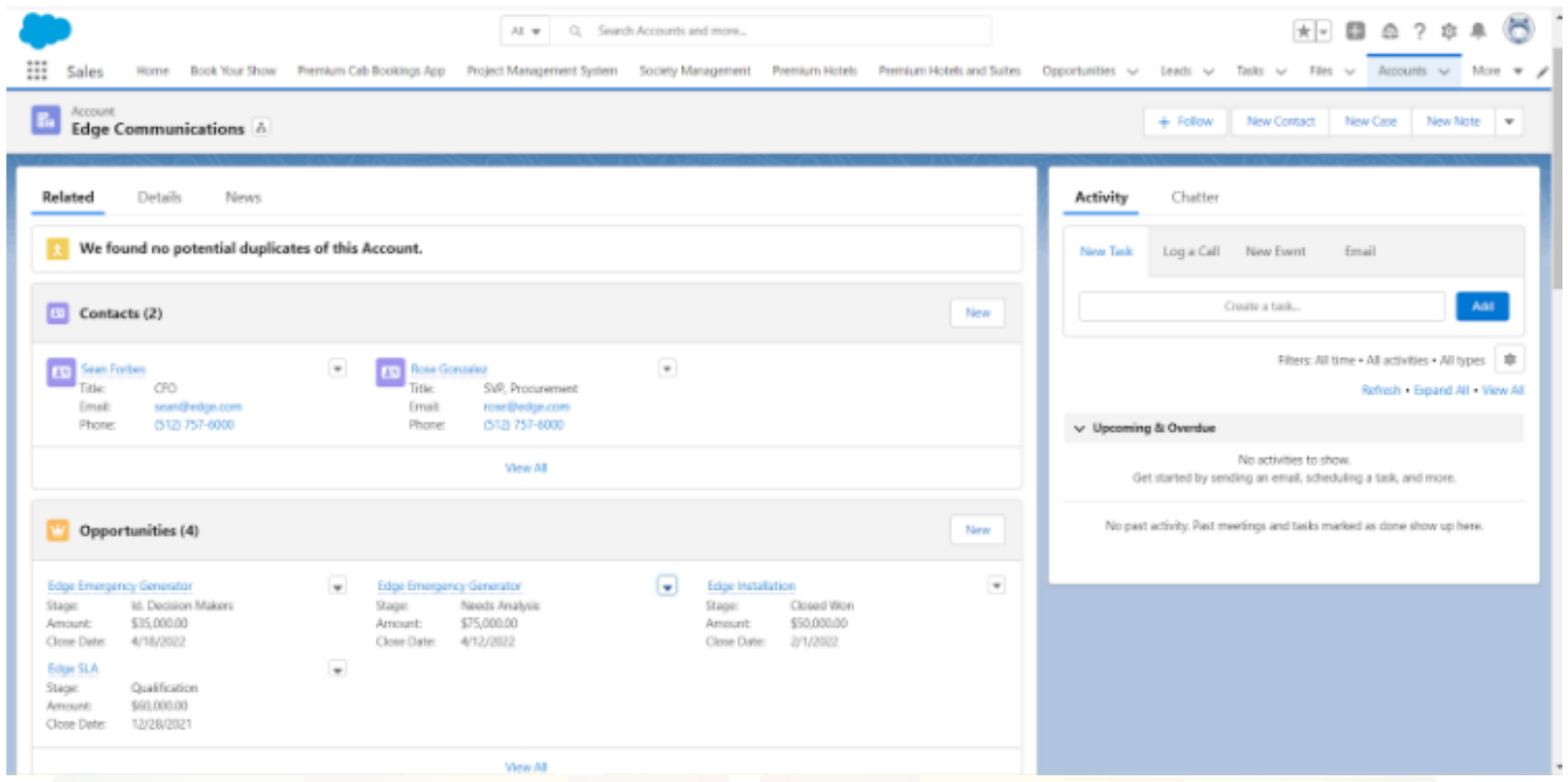
Write a trigger on the Account when the Account is updated check all opportunities related to the account. Update all Opportunities Stage to close lost if an opportunity created date is greater than 30 days from today and stage not equal to close won.

```
trigger OppoStageUpdate on Account (after update){

 Set<Id> accountIds = new Set<Id>();

for(Account a:Trigger.new){
accountIds.add(a.Id);
 }

//day30 is the date which is 30 days less than
//today
DateTime day30=system.now()-30;
List<Opportunity> oppListToUpdate=new List<Opportunity>();

//getting the opportunities whose account has
//been updated
List<Opportunity> oppList = [Select Id, AccountId,
   StageName,CreatedDate, CloseDate from Opportunity where
   AccountId in :accountIds];

if(oppList.size()>0)
{
   for(Opportunity opp : oppList)
   {

//checking for condition if created date is
//greater than 30 days from today and stage not equal to
//close won

    if(opp.CreatedDate<day30 && opp.StageName!=
       'Closed Won')
    {
    opp.StageName='Closed Lost';
```

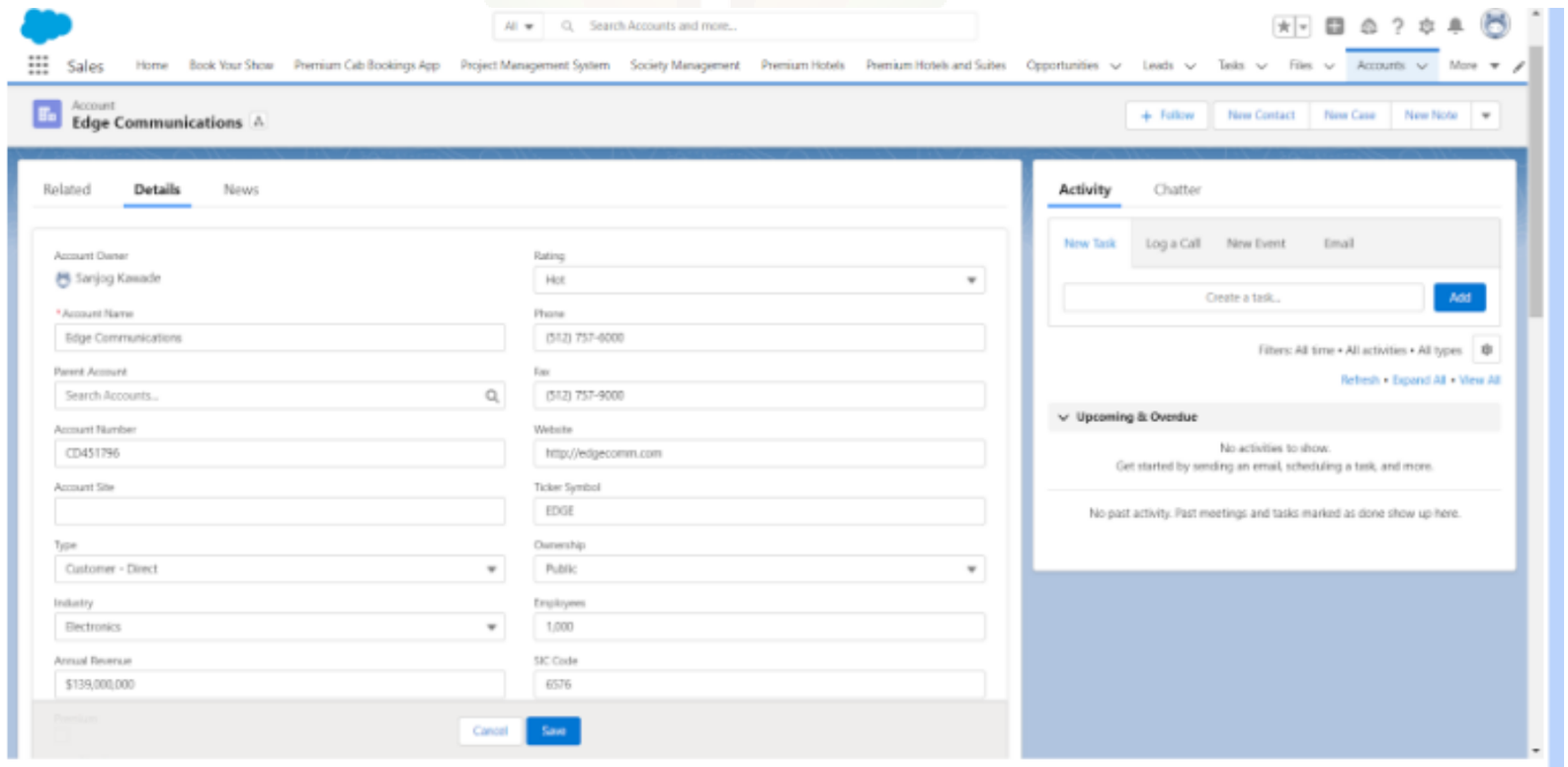
```
    //This is a mandatory field when we update the
    //CloseDate

          opp.CloseDate=system.today();
          oppListToUpdate.add(opp);
    //putting the changed opportunity to separate list
    //to update later
          }
       }
    }
     //checking if the opportunity list which has
     //changed is having records or not

      if(oppListToUpdate.size()>0){
      update oppListToUpdate;
       }
    }
```

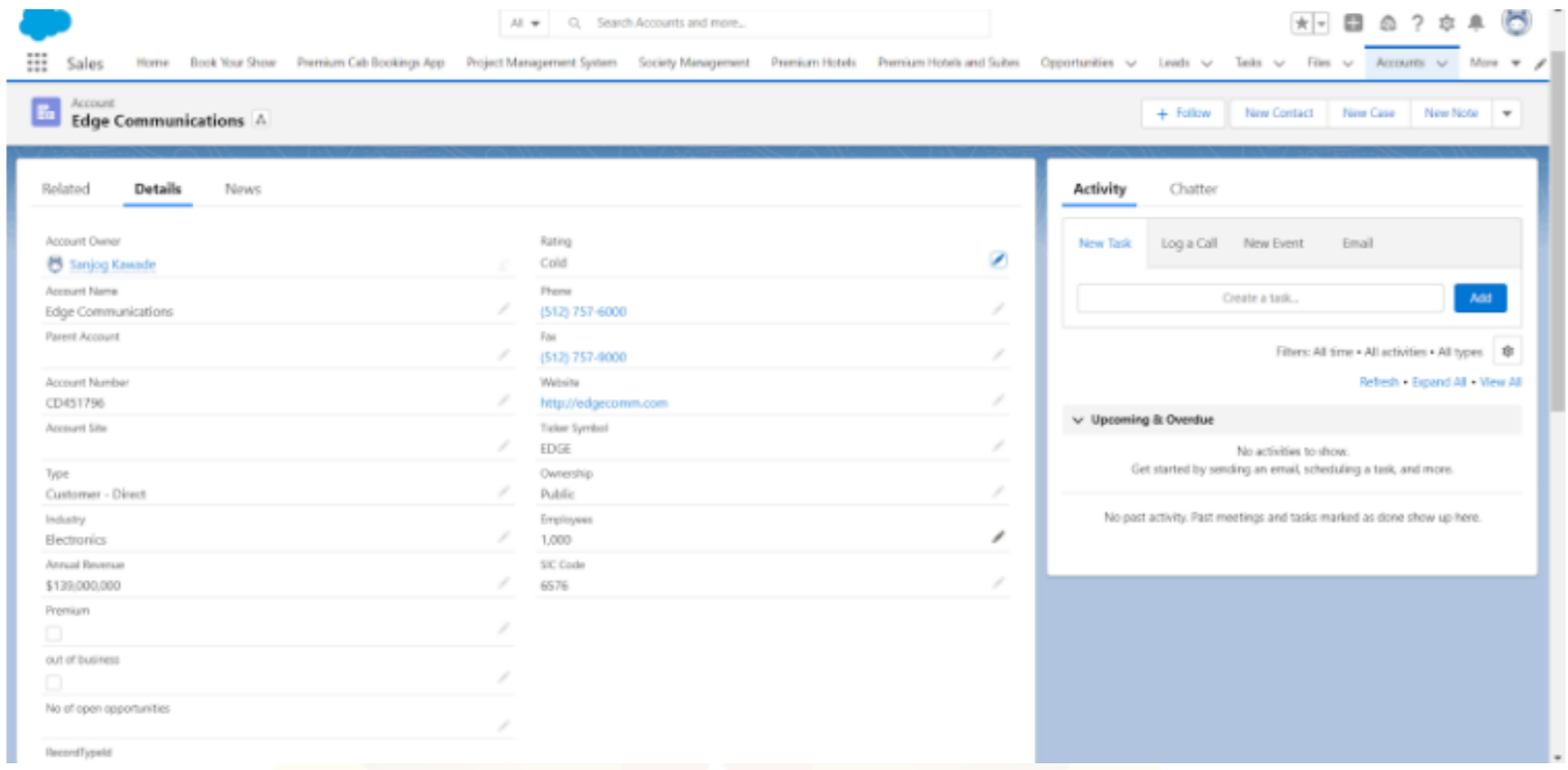
Test Case:

1. Select an account with opportunities created date greater than 30 days.
2. Make sure at least one related opportunity of the account is not closed won or closed lost.

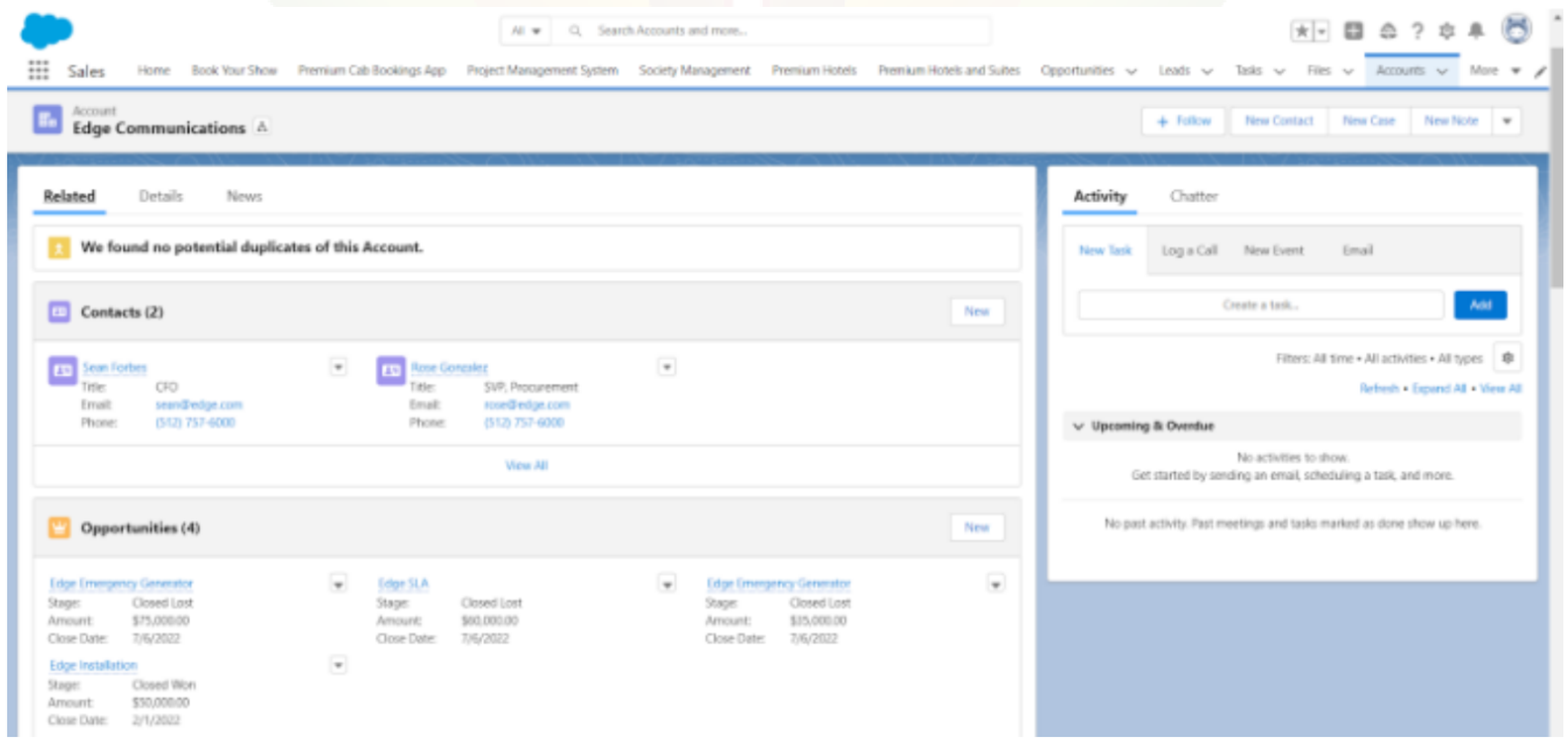


3. Now update the account details with your choice of update (here we change the rating from hot to cold)





4. Now check the related list again, and all the related opportunities (which are not closed won or closed lost) will become closed lost. As shown below image.



Example 3

Once an Account is inserted an email should go to the System Admin user with specified text below.
An account has been created and the name is “Account Name”.

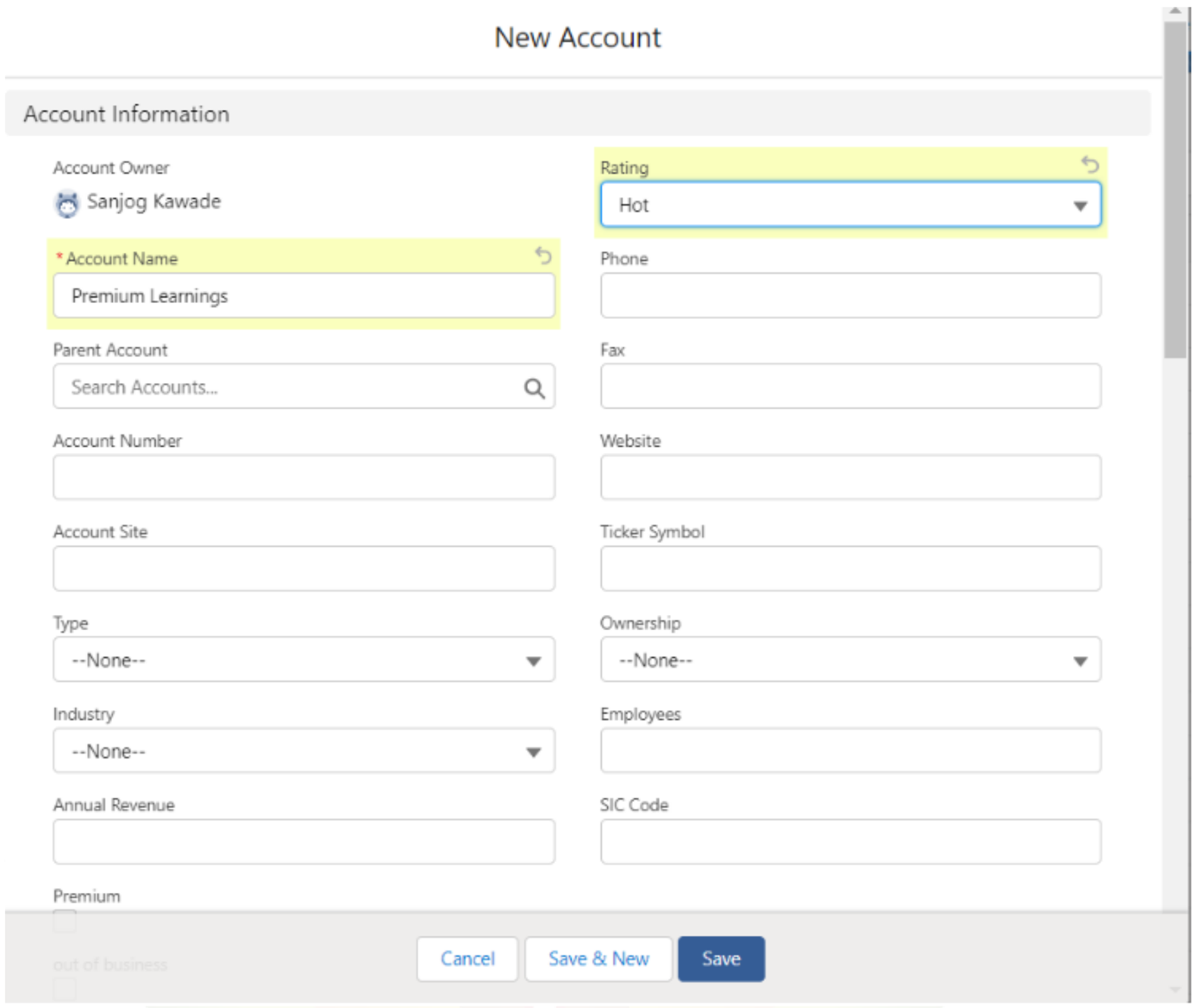
```
trigger sendEmailToAdmin on Account (after insert){
 //To send list of mails when there are insertion
 //of list of Accounts

List<Messaging.SingleEmailMessage> mails = new
List<Messaging.SingleEmailMessage>();

//Query to get the Email of a System
//Administrator

User userObj=[select Id,Profile.Name,Email from user where
              Profile.Name='System Administrator'];
for(Account accObj:Trigger.new)
{

//Checking if the user email is not null
if(userObj.Email!=null)
{
//Assigning a single Mail to send
  Messaging.SingleEmailMessage mail = new
  Messaging.SingleEmailMessage();

//Assigning the Sender Name for Mail
  mail.setSenderDisplayName('Salesforce');

//We are make all this below fields as false
//because those are not needed for now
    mail.setUseSignature(false);
    mail.setBccSender(false);
    mail.setSaveAsActivity(false);
```

```
    //We are make all this below fields as false
    //because those are not needed for now
      mail.setUseSignature(false);
      mail.setBccSender(false);
      mail.setSaveAsActivity(false);

    //Assigning the receiver Mail Address
      mail.toAddresses = new String[]{userObj.Email};

    //Assign the Subject of the Mail
      mail.setSubject('New Account was Created.');

    //A variable to write the body of the Mail
      String body = 'Dear System Administrator,<br/>';
      body += 'An account has been created and name is
            '+accObj.Name+'.';

    //Assigning the variable in which we wrote the
    //body to the Mail Body
      mail.setHtmlBody(body);

    //Adding each single mail to be sent to the list
    //of mails
      mails.add(mail);
      }
    }

    //Checking if the list of mails is not empty
    if(mails.size()>0){
    //"Messaging.sendEmail(mails) " is used to send
    //the list of mails
```

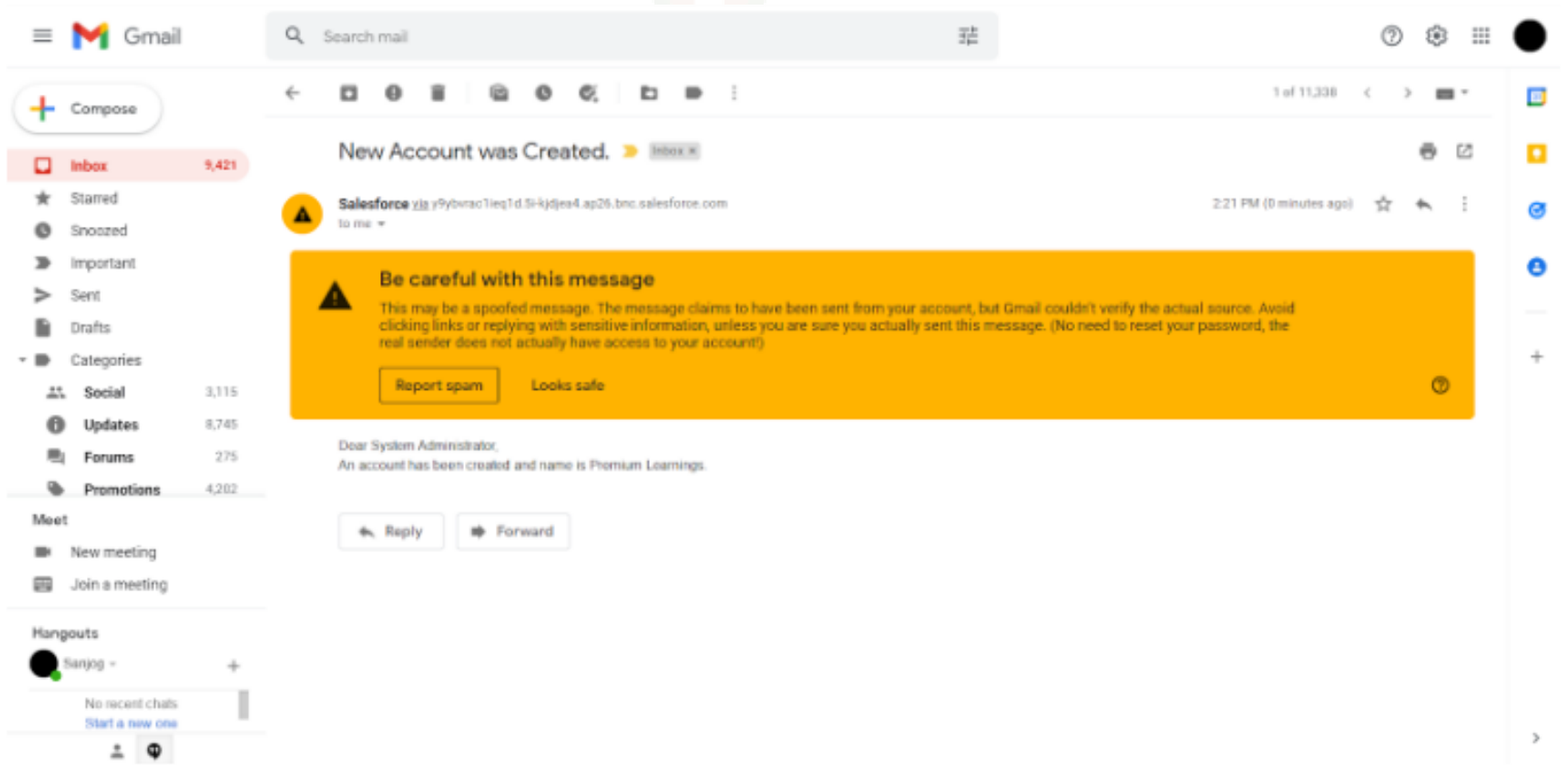
```
        Messaging.SendEmailResult[] results =
        Messaging.sendEmail(mails);
    //we are checking if the mails are sent or not.
        if (results[0].success)
        {
            System.debug('The email was sent successfully.');
        }
         else
         {
           System.debug('The email failed to send:
           '+ results[0].errors[0].message);
         }
      }
}
```

Test Case:

1. Create a new account, give it a suitable name and save.



2. Now check your email(you being the system administrator of your salesforce org). you will receive a mail with the format as shown below.



Example 4

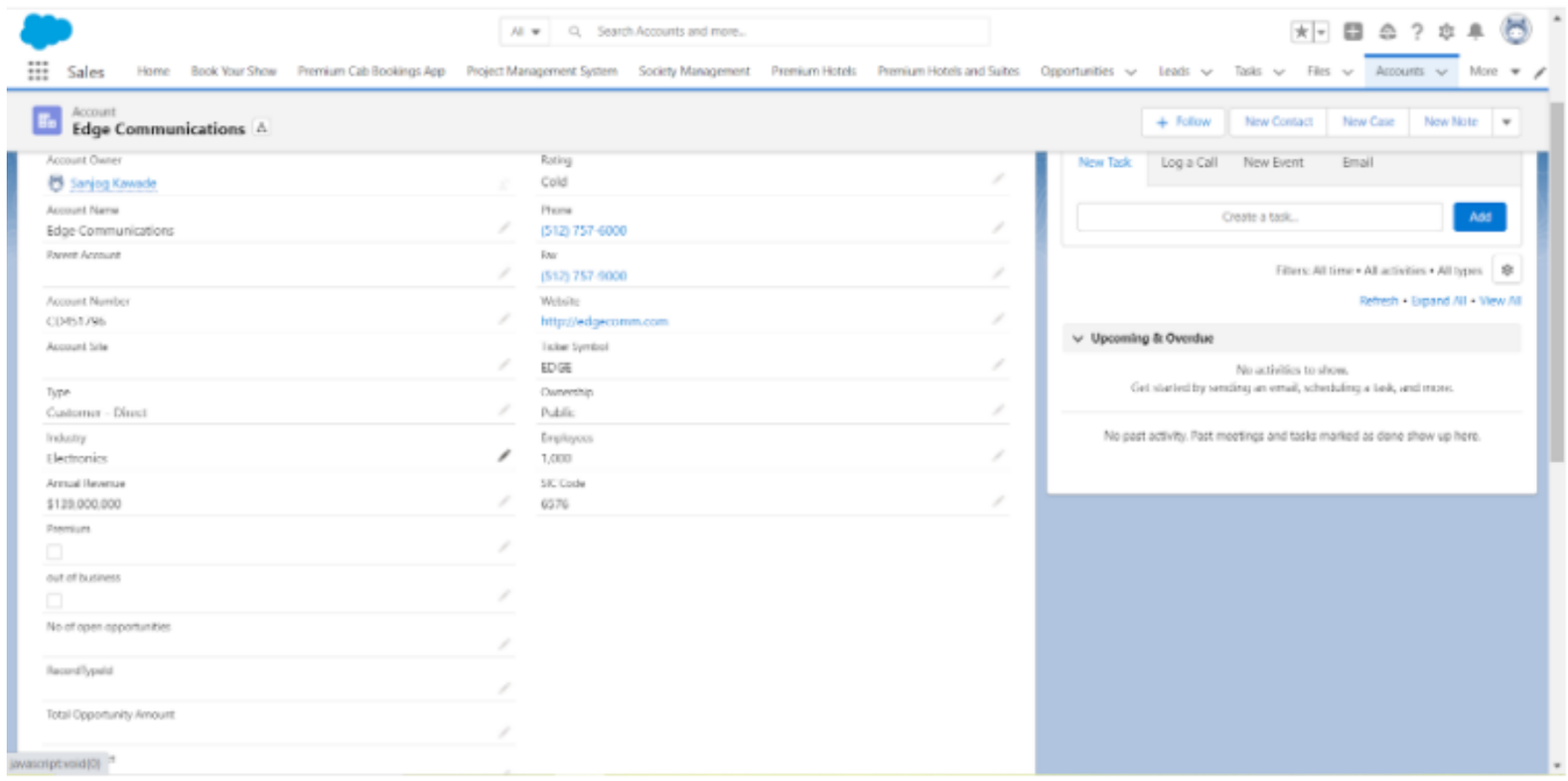
Once an Account will update then that Account will update with the total amount from All its Opportunities on the Account Level. The account field name would be " Total Opportunity Amount ".create a field named Total Opportunity Amount of datatype number

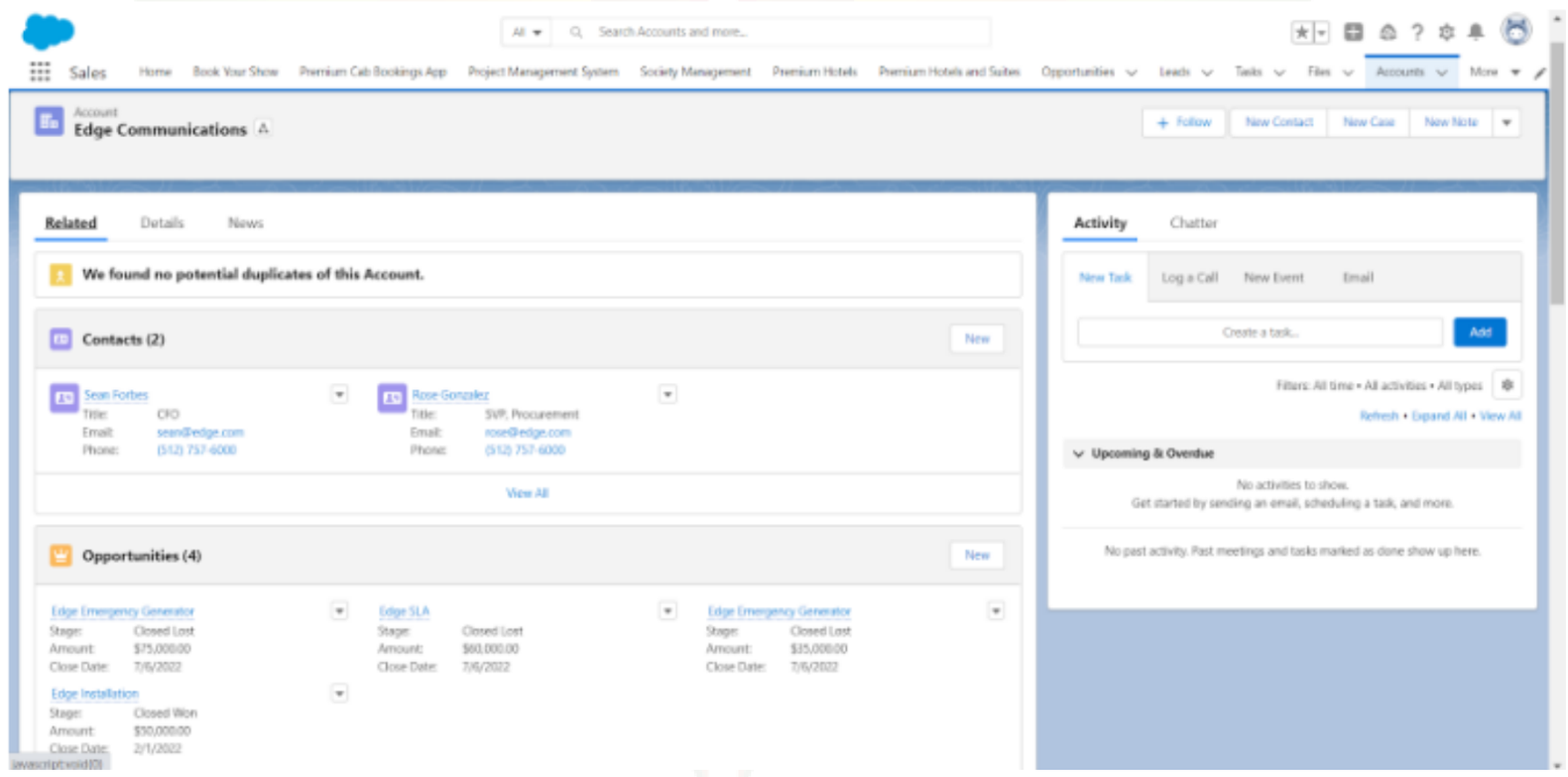
```
trigger totalAmountofRelatedOpp on Account (before update)
{
  //Get the Account Id's of the List of Accounts
  //getting updated to get its related Opportunities

  Set<Id> accId=new Set<Id>();
  for(Account acc:Trigger.new)
  {
  //before making changes see to that "Total
  //Opportunity Amount" field is 0
    acc.Total_Opportunity_Amount__c=0;
    accId.add(acc.Id);
   }

  //map to get the Account Id and the sum of its
  //related opportunities amount to insert later
  Map<Id,Double> amountMap = new Map<Id,Double>();

  //AggregateResult to get the sum of opportunities
  //amount grouped by Account Id
  List<AggregateResult> results=[select
  AccountId,sum(Amount)TotalAmount from opportunity where
  AccountId in :accId group by AccountId];
  if(results.size()>0){
     for(AggregateResult a: results){
```

```
//getting the AccountId and sum(Amount) in
//separate variables and putting it to map
  Id accountId = (Id)a.get('AccountId');
  double TotalAmount = (double)a.get('TotalAmount');
  amountMap.put(accountId,TotalAmount);
  }
 }

//Again looping the accounts which are getting
//updated and making changes in "Total
//Opportunity Amount" field
for(Account acc:Trigger.new)
{
  if(amountMap.containskey(acc.Id))
  {
    acc.Total_Opportunity_Amount__c=amountMap.get(acc.Id);
  }
 }
}
```

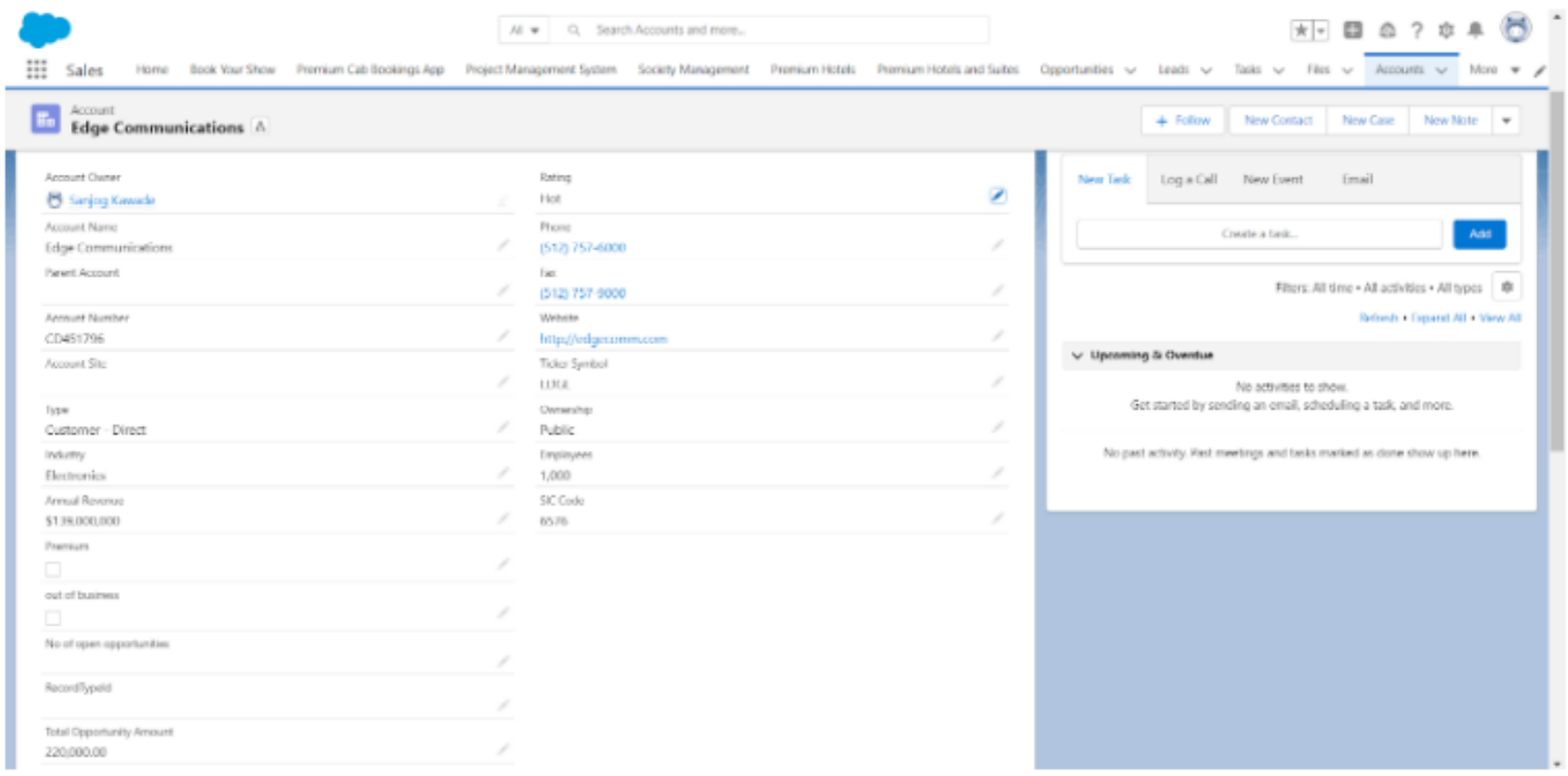
Test Case:

1. Select an account with existing related opportunities (or create one). Notice that the total opportunity amount field is blank.





2. Now update the account (change rating field). The apex trigger will be fired and total opportunity amount field will be populated with amount of all opportunities added as shown in figure.



Example 5

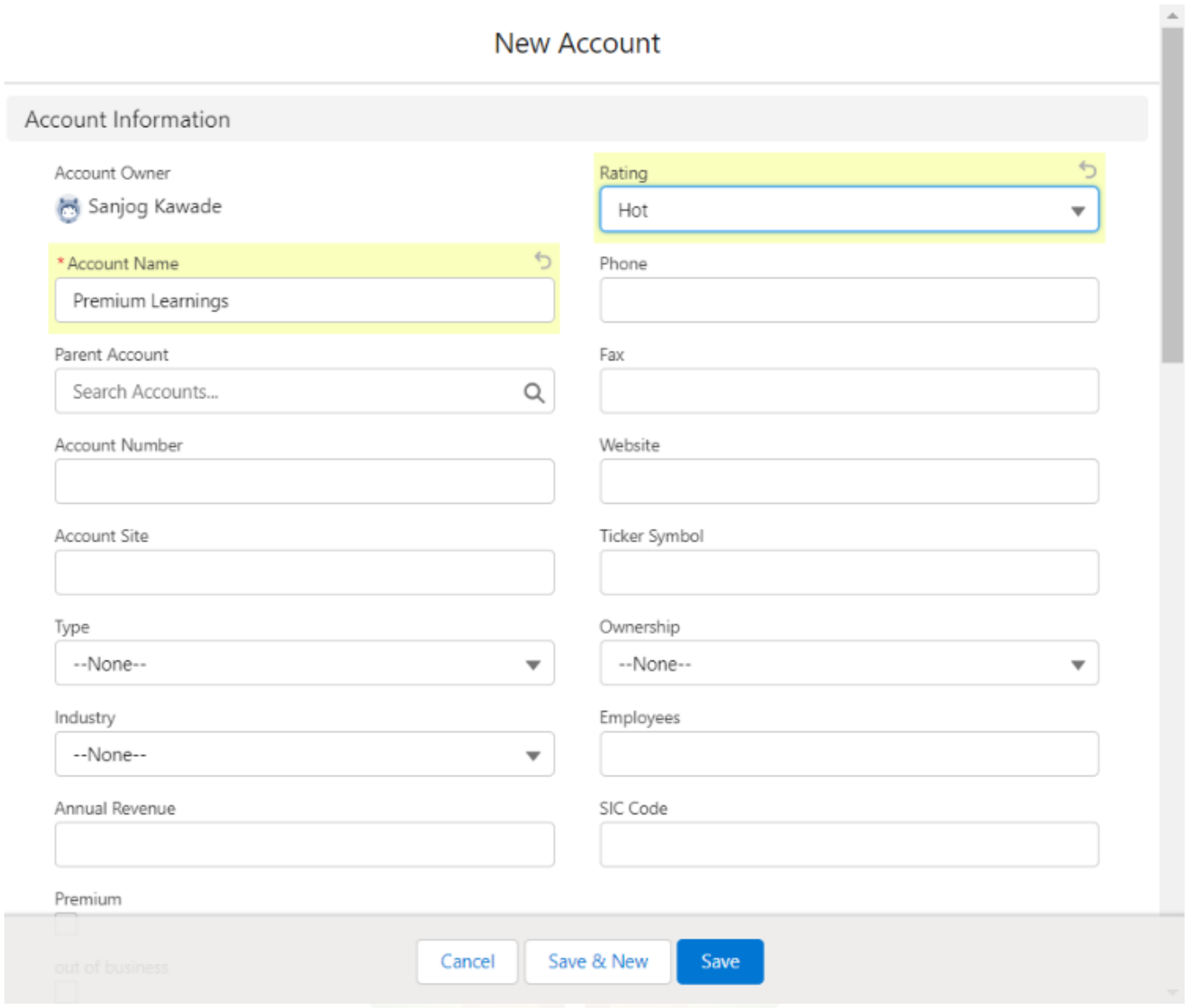
Create a field on Account Named (Client Contact lookup to Contact). Once an Account is inserted a Contact will create with the name of the Account and that Contact will be the Client Contact on the Account.

```
trigger updateClientContactFieldOnAccount on Account
(after insert){

 //Account list to update after the doing changes
List<Account> accListUpdate = new List<Account>();

//Contacts to insert which has same name as
//Account Name
List<Contact> conList = new List<Contact>();

//Account Id list to update later after changes
//are done
Set<Id> accId = new Set<Id>();

//Map of Account Id and the Axxount record
Map<Id,Account> mapVariable = new Map<Id,Account>();

for(Account accObj : Trigger.new)
{

//Contact which has to be inserted with same name
//as Account and putting that Contact as related
//Contact
  Contact conObj =new Contact();
  conObj.LastName = accObj.Name;
  conObj.AccountId = accObj.Id;
  conList.add(conObj);
  accId.add(conObj.AccountId);
}
if(conList.size()>0){
insert conList;
}
```

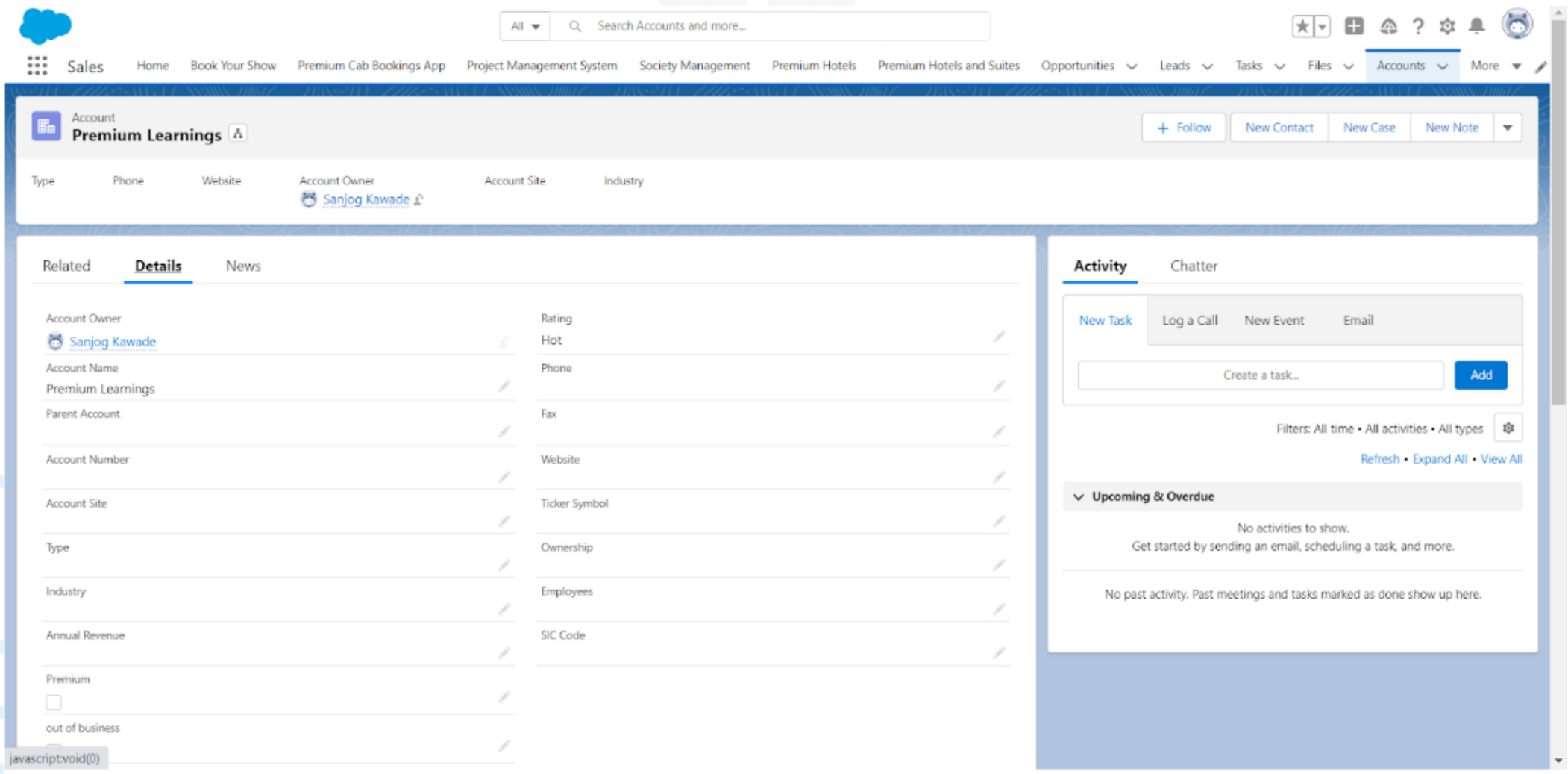
```
//Getting list of Accounts to put into Map
List<Account> accList =[Select Id,Client_Contact__c from
                        Account where Id=:accId];

if(accList.size()>0){
    for(Account acc:accList){
        mapVariable.put(acc.Id,acc);
    }
}

if(conList.size()>0){
/*Looping inserted Contact and Check whether the Account
  Id is related and get its Account record from map and
  change the value of the Client Contact with the current
  looping Contact*/

   for(Contact cObj:conList){
      if(mapVariable.containsKey(cObj.AccountId)){
         Account aObj=mapVariable.get(cObj.AccountId);
         aObj.Client_Contact__c=cObj.Id;
         accListUpdate.add(aObj);
      }
    }
  }

//Update the Account in which chnages has been done
if(accListUpdate.size()>0){
   update accListUpdate;
  }
}
```

Test Case:

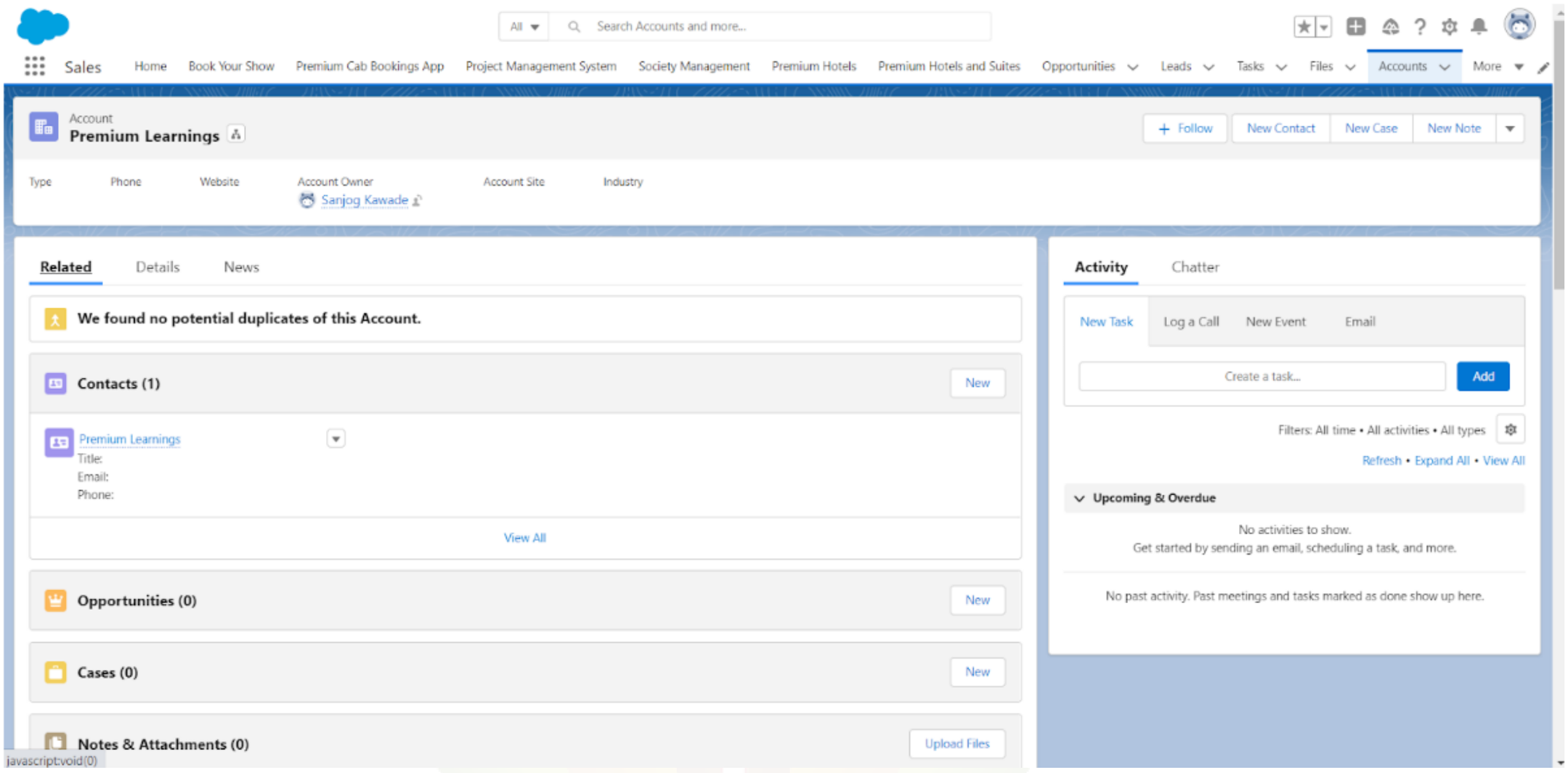
1. Create an account record as shown in figure. Save it



2. The record is saved.



3. Check the related lists of the record. You will see a contact record created with the same name as the account.

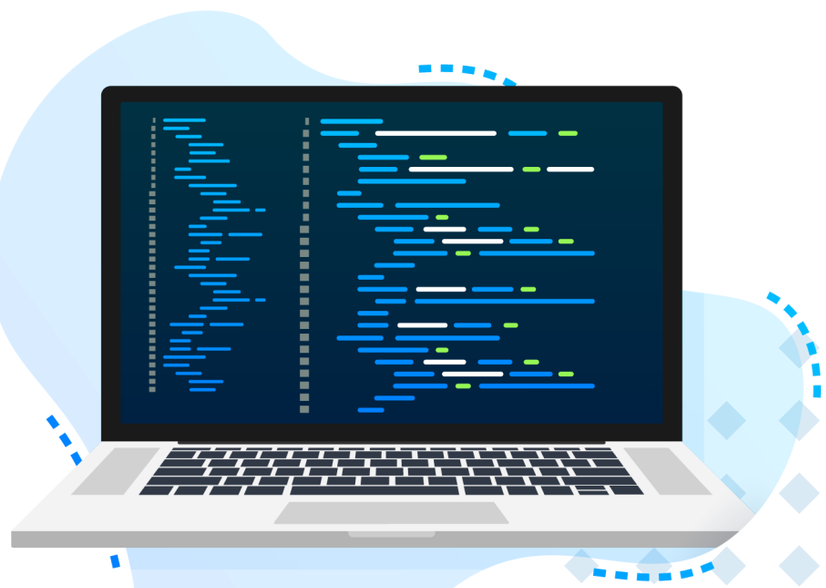


Example 6

Write a trigger on contact to prevent duplicate records based on Contact Email & Contact Phone

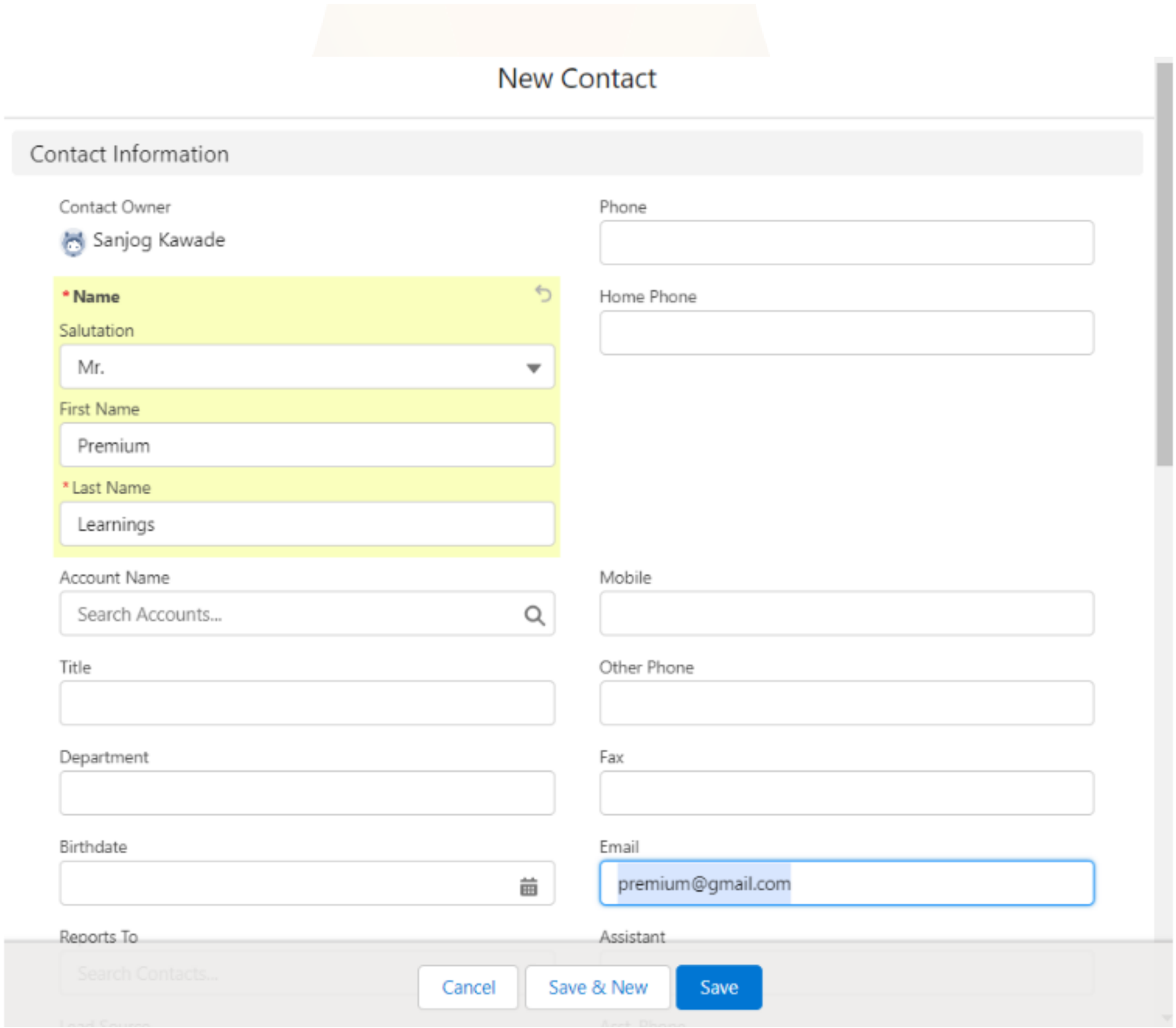
```
trigger dupEmailPhone on Contact (before insert,
before update)
{
 Map<String, Contact> emailMap = new Map<String, Contact>();
Map<String, Contact> phoneMap = new Map<String, Contact>();

for(Contact contact : trigger.new){
    if(trigger.isInsert){
        emailMap.put(contact.Email, contact);
        phoneMap.put(contact.Phone, contact);
    }
    if(trigger.isUpdate){
     if(trigger.oldMap.get(contact.Id).Email!=contact.Email)
     {
        emailMap.put(contact.Email,contact);
     }
     if(trigger.oldMap.get(contact.Id).Phone!=contact.Phone)
     {
        phoneMap.put(contact.Phone,contact);
     }
   }
}

String errorMessage ='';
```

```
    //getting the Contacts whose Email or Phone
    //already exists

    List<Contact> existingContactList = [Select Id, Email,
        Phone FROM Contact Where Email IN:emailMap.keySet() OR
        Phone IN:phoneMap.keySet()];

    if(existingContactList.size()> 0){
       for(Contact contactRec : existingContactList){
           if(contactRec.Email!=null){
              if(emailMap.get(contactRec.Email) != null){
                 errorMessage='Email ';
              }
           }
           if(contactRec.Phone!=null){
              if(phoneMap.get(contactRec.Phone) != null) {
                errorMessage = errorMessage +(errorMessage!='' ?
                'and Phone ' : 'Phone ');
              }
           }
           if(errorMessage!=''){
              trigger.new[0].addError('Your Contact'+errorMessage
              +'already exists in system.');
           }
       }
    }
}
```

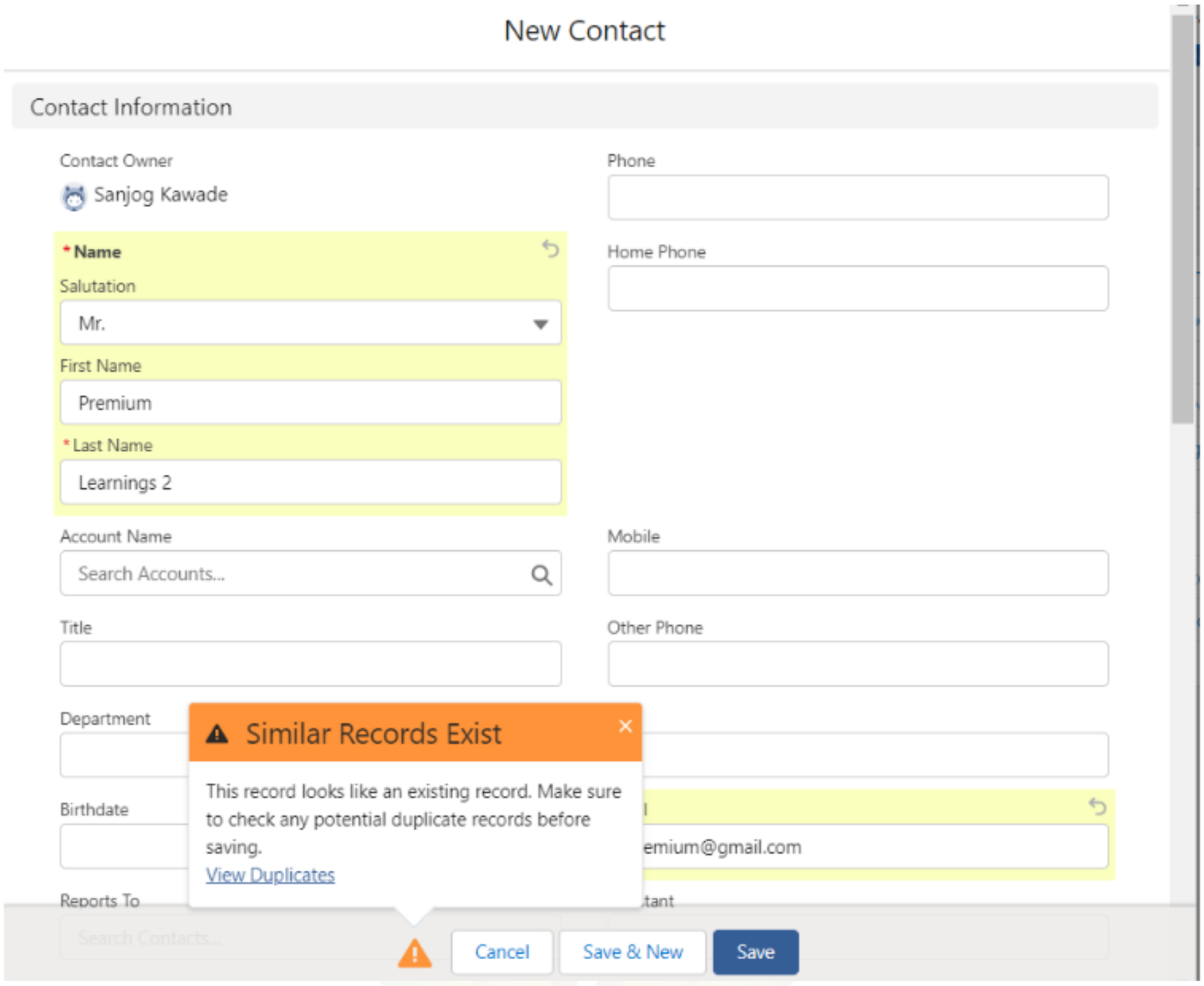


Test Case:

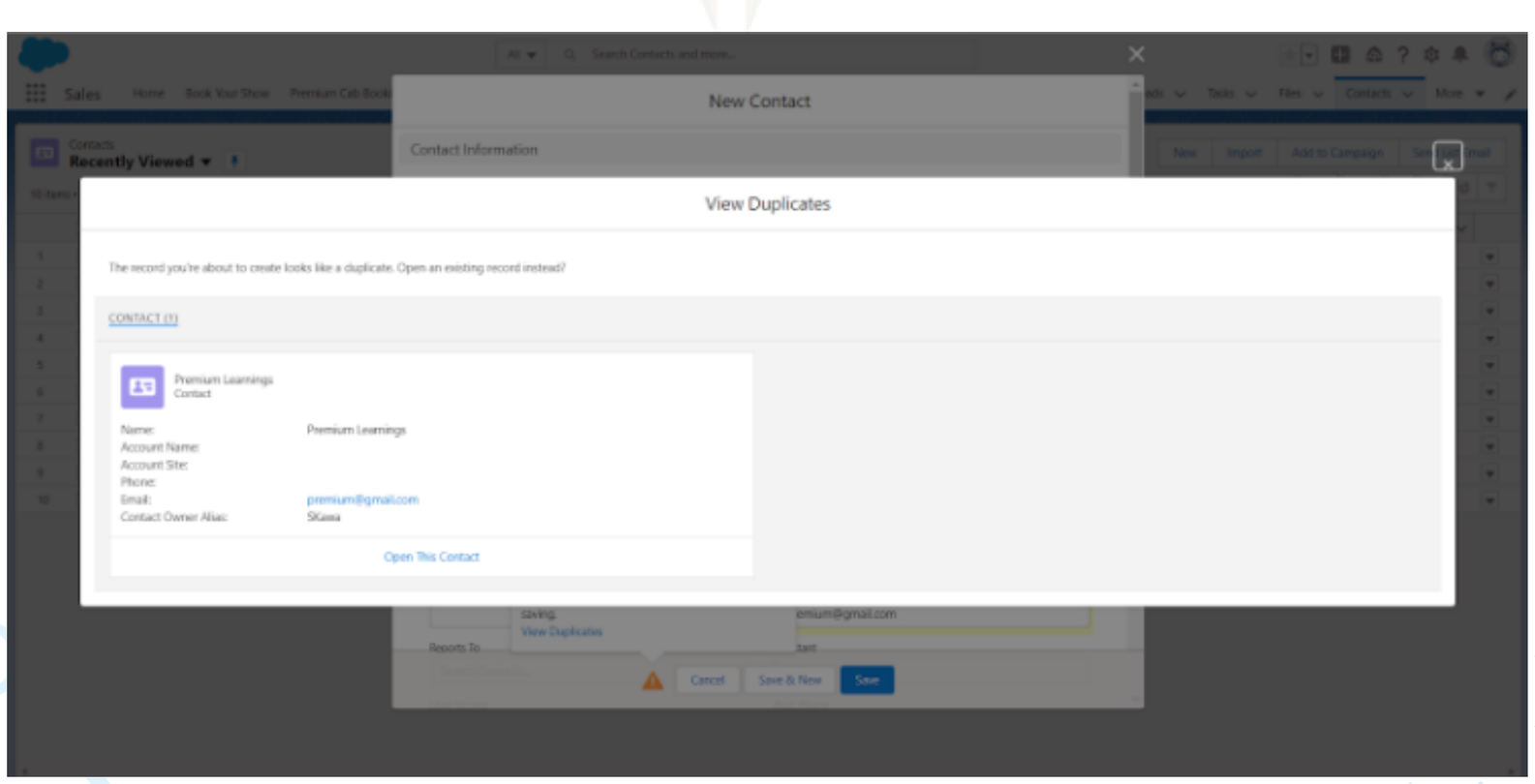
1. Create a contact record, with the email field populated(remember this email). Click save.



2.Now create another contact with same email. When you click save, the following error message is displayed and record fails to save.



3. When you click on View Duplicates, the previously created contact record with the same email is shown.



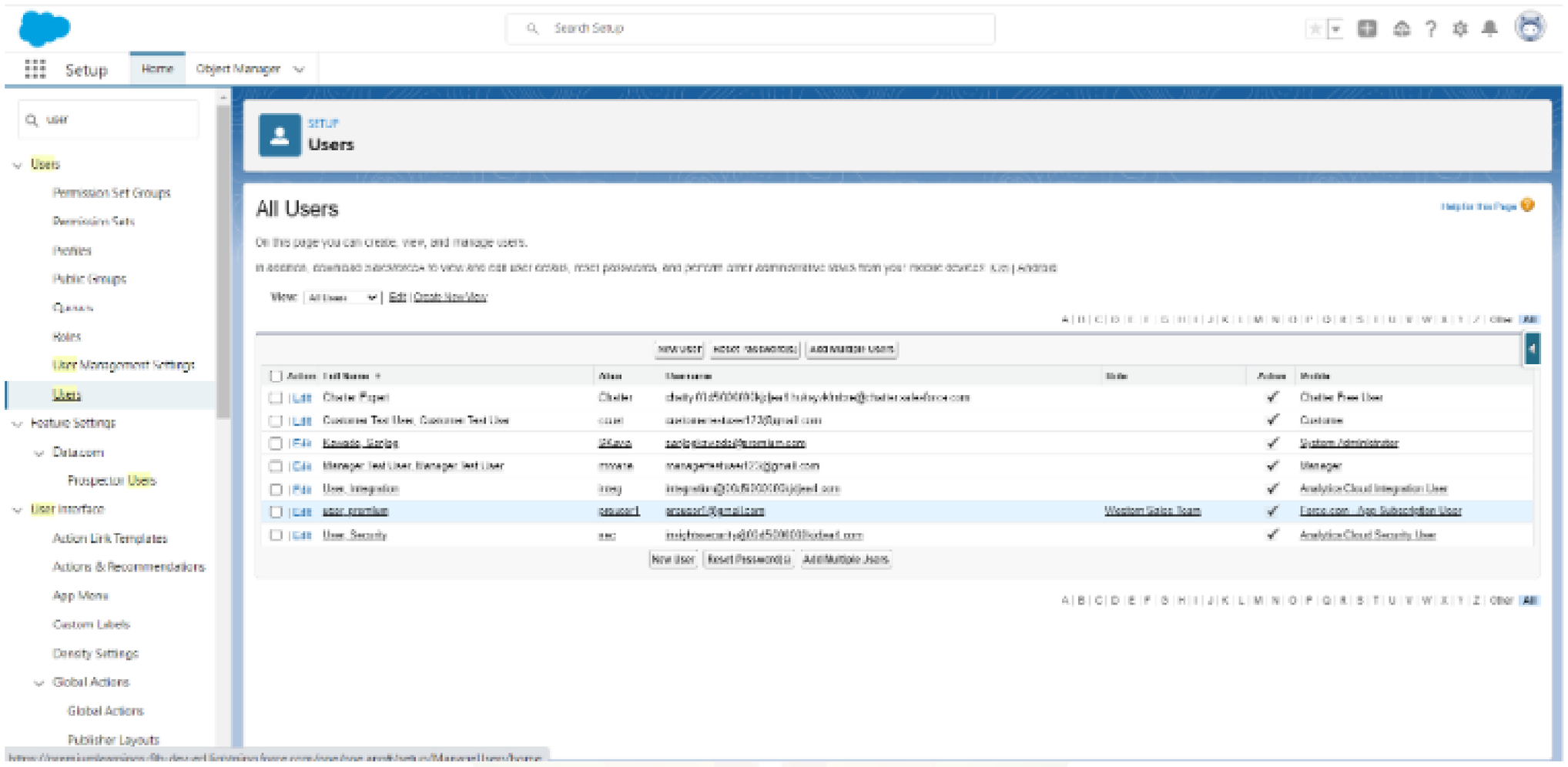
Example 7

Write a trigger, only the system admin user should be able to delete the task. you have to assign a task to the other user to make it visible in their org

```
trigger taskThatCanBeDeletedBySA on Task (before delete)
{
    Id pid = Userinfo.getProfileId();
    Profile pname=[select Name from Profile where id=:pid];

    for(Task taskObj:Trigger.old){
        if(pname.Name!='System Administrator'){
            taskObj.addError('No Access for Deletion');
        }
    }
}
```

Test Case:

1. Create a new user
   (setup > home > quick find > users > new user)make sure the new user is not of system administrator profile and email id field is your working email id. we created a ‘premium user’ for this example.



2. Now create a new task and assign(from your system admin org) it to the newly created user(premium user).

New Task

Task Information

Assigned To: premium user

1 Total Task

*Status: In Progress

*Subject: Test Task

Name: Search Contacts...

Due Date

Related To: Search Accounts...

*Priority: High

Description Information

Comments

Save & New | Cancel | Save

Task
Test Task

Task Test Task was created.

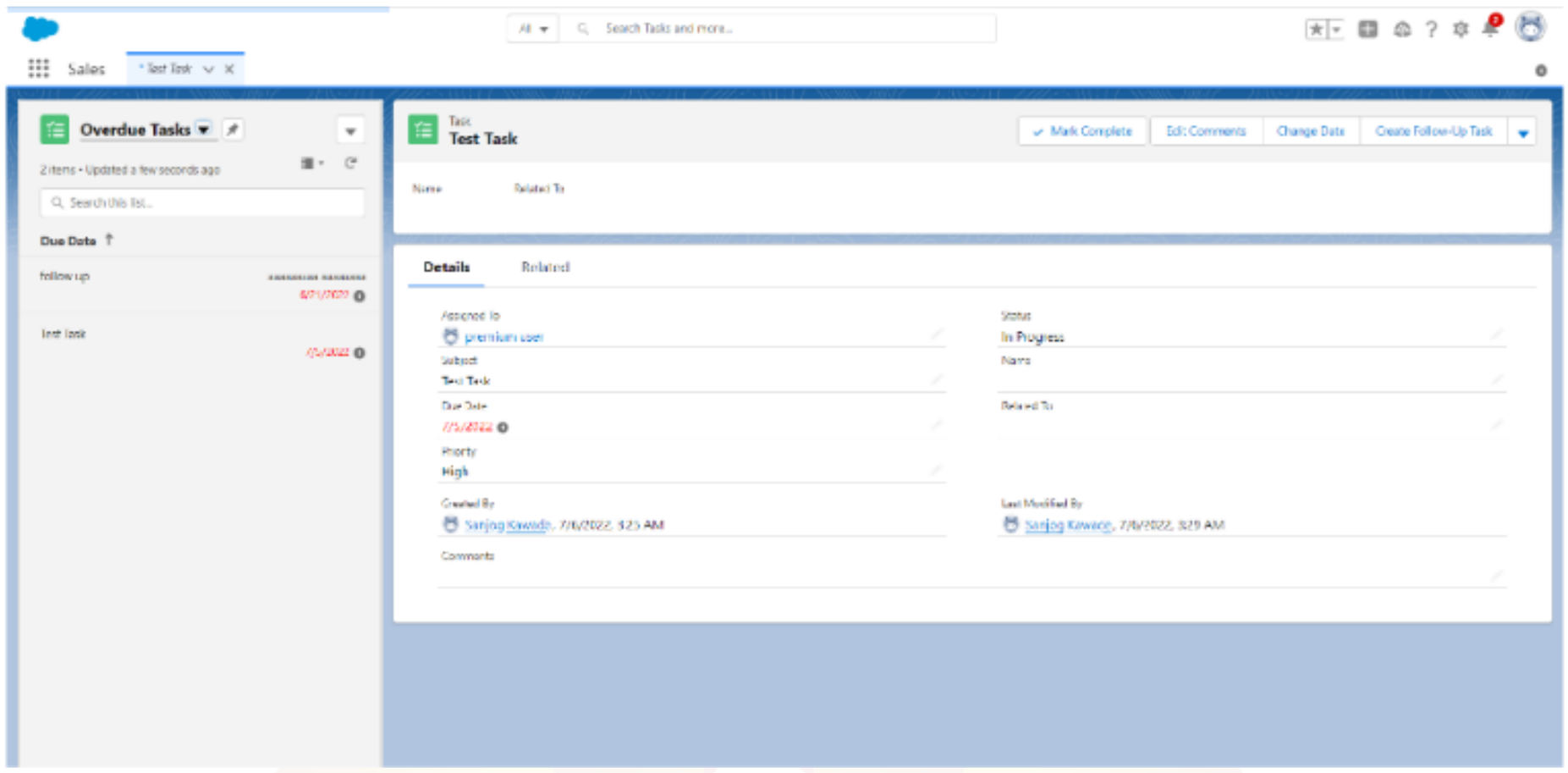
Mark Complete | Edit Comments | Change Date | Create Follow-Up Task

Name | Related To

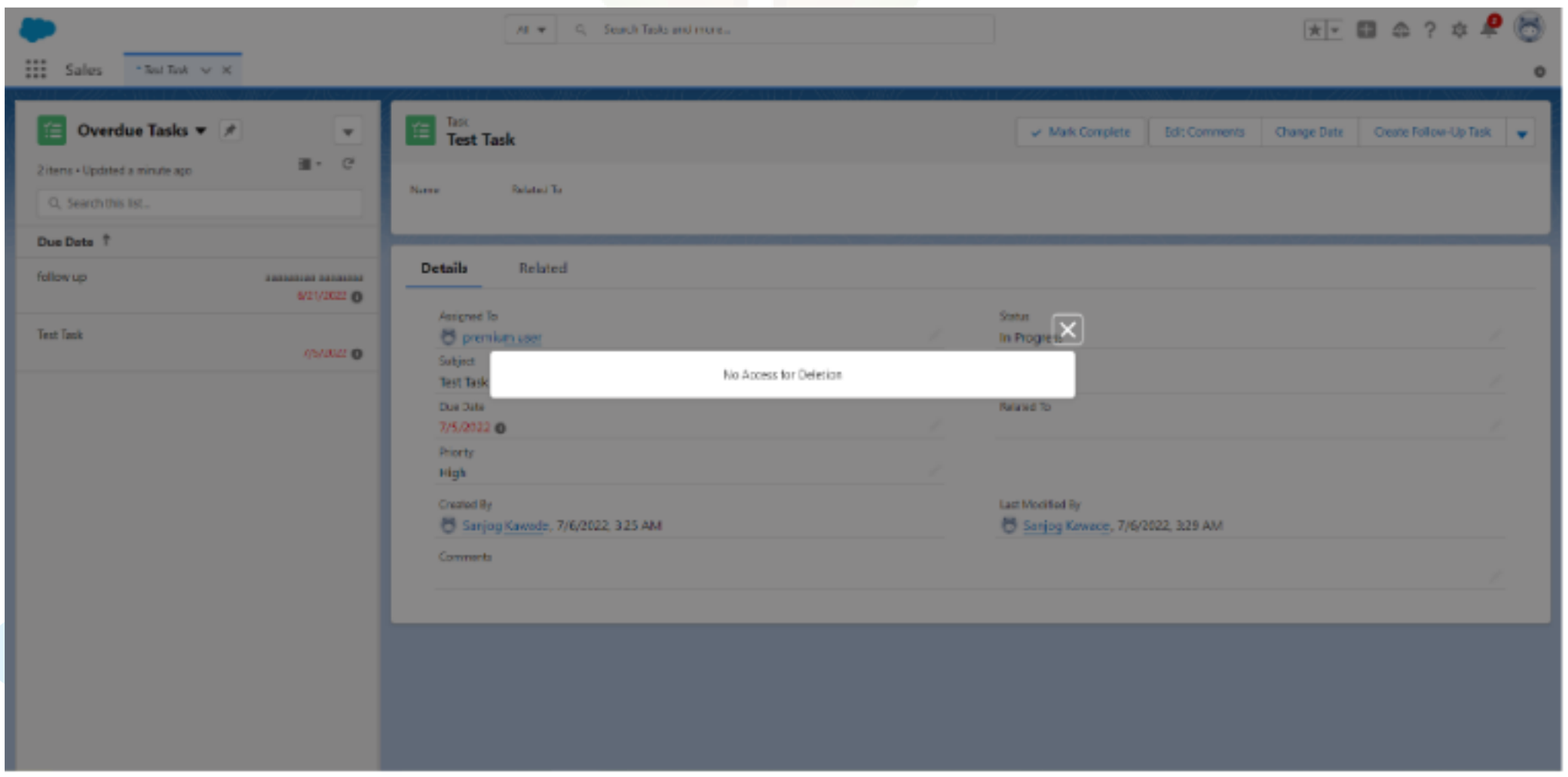
Details | Related

Assigned To: premium user
Status: In Progress
Subject: Test Task
Name:
Due Date:
Related To:
Priority: High
Created By: Sanjog Kawade, 7/6/2022, 3:55 PM
Last Modified By: Sanjog Kawade, 7/6/2022, 3:55 PM
Comments:

3.Log in as the newly created user. You can view the assigned tasks in that org.



4. Try to delete the assigned task. The following error message is displayed. Since the user is not a system administrator, the premium user cannot delete the task.



Example 8

Create a duplicate lead when a lead is inserted.

CheckRecursive (apex class)

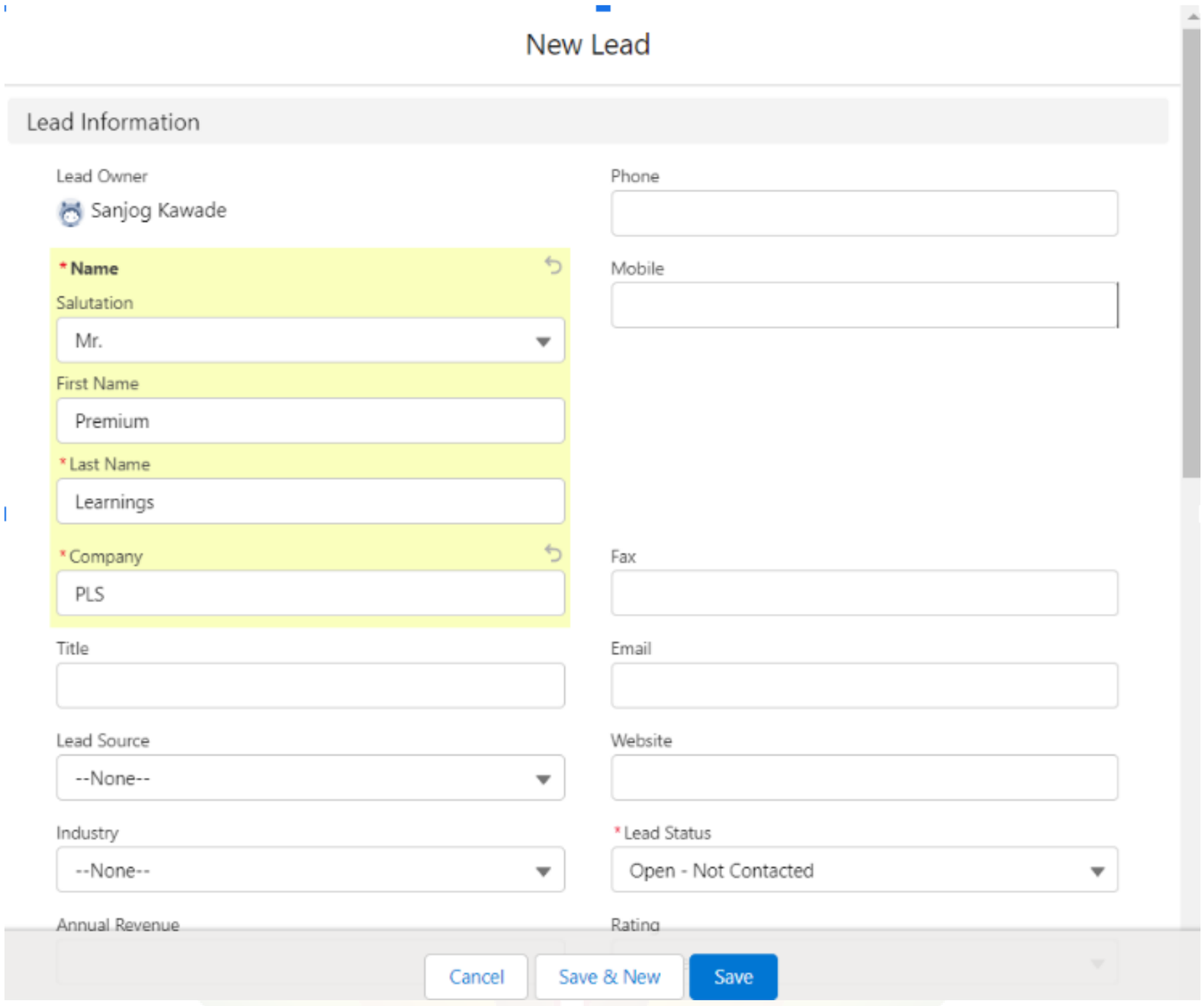
```
public class checkRecursive {
    private static boolean run = true;

    public static boolean runOnce()
    {
        if(run)
        {
            run=false;
            return true;
        }
        else
        {
            return run;
        }
    }
}
```

DuplicateLead (apex trigger)

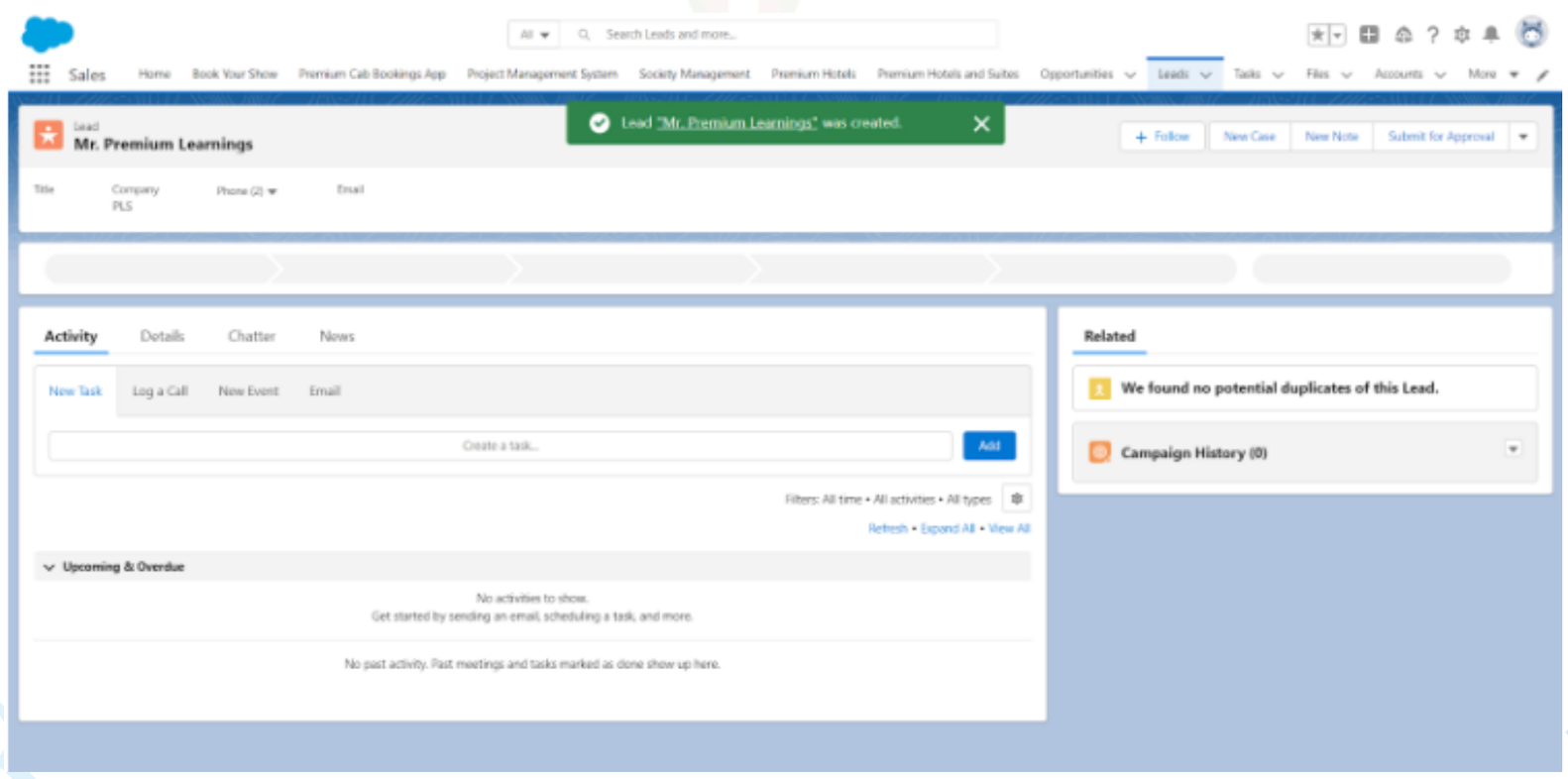
```
trigger duplicateLead on Lead (after insert) {

    if(checkRecursive.runOnce()){

     List<Lead> leadList = new List<Lead>();  leadList
    = Trigger.new.deepClone();
     insert leadList;
    }

}
```

Test Case:

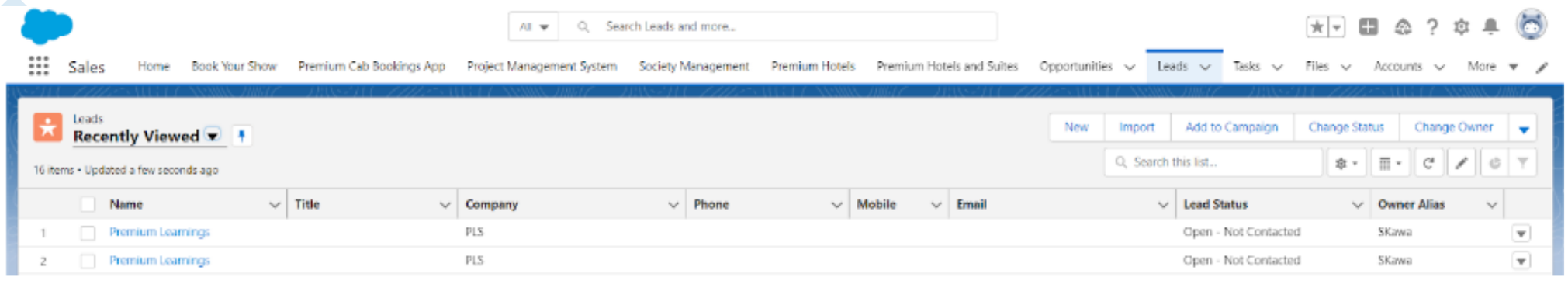
1. Create a new Lead. Click Save.



2. The record is created



3. you can see a duplicate record with the same details is created.



Example 9

Upload any pdf file into the Document first. Send an email as an attachment to the lead email Id when a lead is created.
Email Subject: Welcome
Body: Please find the attached PDF.

On App Launcher search for Email Template then Create a new Email Template with Name=" Pdf Attached Email for Lead", Subject="Welcome", Body= "Please find the attached PDF." and Related Entity type= Lead and Save it.
Later go to its related tab and add the file which u want to attach to the mail.

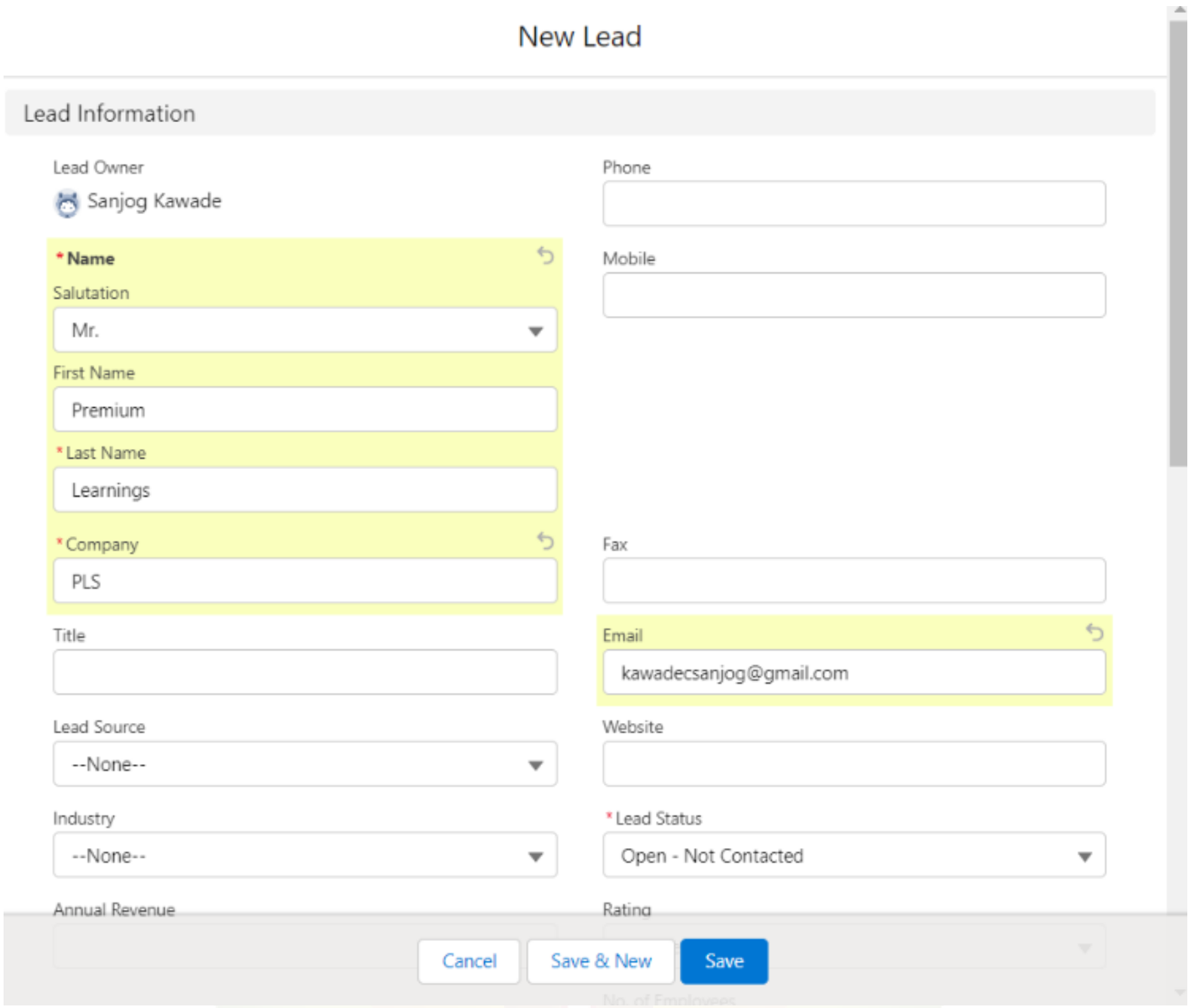
Note: When u send single mail the actual file will be sent.
But if u send a list of emails then the files will be sent in the link format where if they click on that link then the file opens.

```
trigger sendEmailWithPdfFileAttached on Lead (after insert)
{
 List<Messaging.SingleEmailMessage> emailList= new
 List<Messaging.SingleEmailMessage>();

 /*On App Launcher search for Email Template then Create a
new
Email Template with Name="Pdf Attached Email for Lead",
Subject="Welcome", Body= "Please find the attached PDF."
and
Related Entity type= Lead and Save it.
Later go to its related tab and add the file which u want
to attach to
the mail.
Note : When u send single mail the actual file will be
send.
But if u send a list of mails then the files will be send
in the
link format where if they click on that link then the file
opens.*/
```

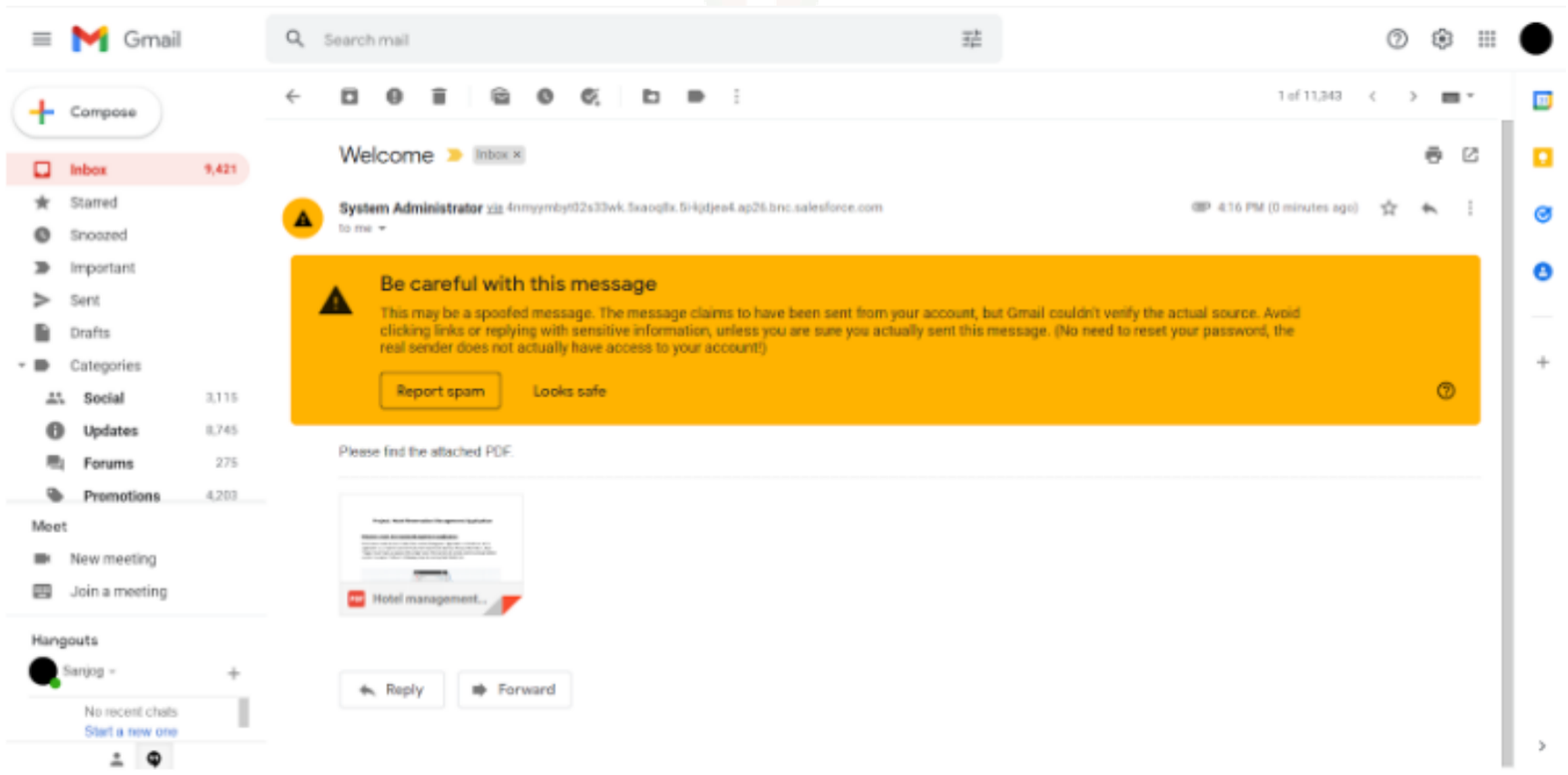
```
EmailTemplate emailTemplate = [Select
  Id,Subject,Description,HtmlValue,DeveloperName,Body
  from EmailTemplate where name='Pdf Attached Email
  for Lead'];

for(Lead lObj:Trigger.new){
    if(lObj.Email!=null){
       Messaging.SingleEmailMessage mail = new
                              Messaging.SingleEmailMessage();
       mail.setTargetObjectId(lObj.Id);
       mail.setSenderDisplayName('System Administrator');
       mail.setUseSignature(false);
       mail.setBccSender(false);
       mail.setSaveAsActivity(false);
       mail.setTemplateID(emailTemplate.Id);
       mail.toAddresses = new String[]{lObj.Email};
       emailList.add(mail);
    }
}

    if(emailList.size()>0){
       Messaging.SendEmailResult[] results =
                              Messaging.sendEmail(emailList);
       if (results[0].success)
       {
         System.debug('The email was sent successfully.');
       }
       else {
           System.debug('The email failed to send:
                    '+ results[0].errors[0].message);
       }
    }
}
```

Test Case:

1. Create a new lead. Make sure email id is populated



2. You will receive an email with attachment as shown



Example 10

Write a trigger on Contact, when contact is inserted an email should be sent to
the contact email id with a specified template. So first you need to create a
text template. The template design is below. Template name: Email Template
to send After Contact is Inserted Subject: A new Contact was Created.
Body: Dear “Contact Name”,

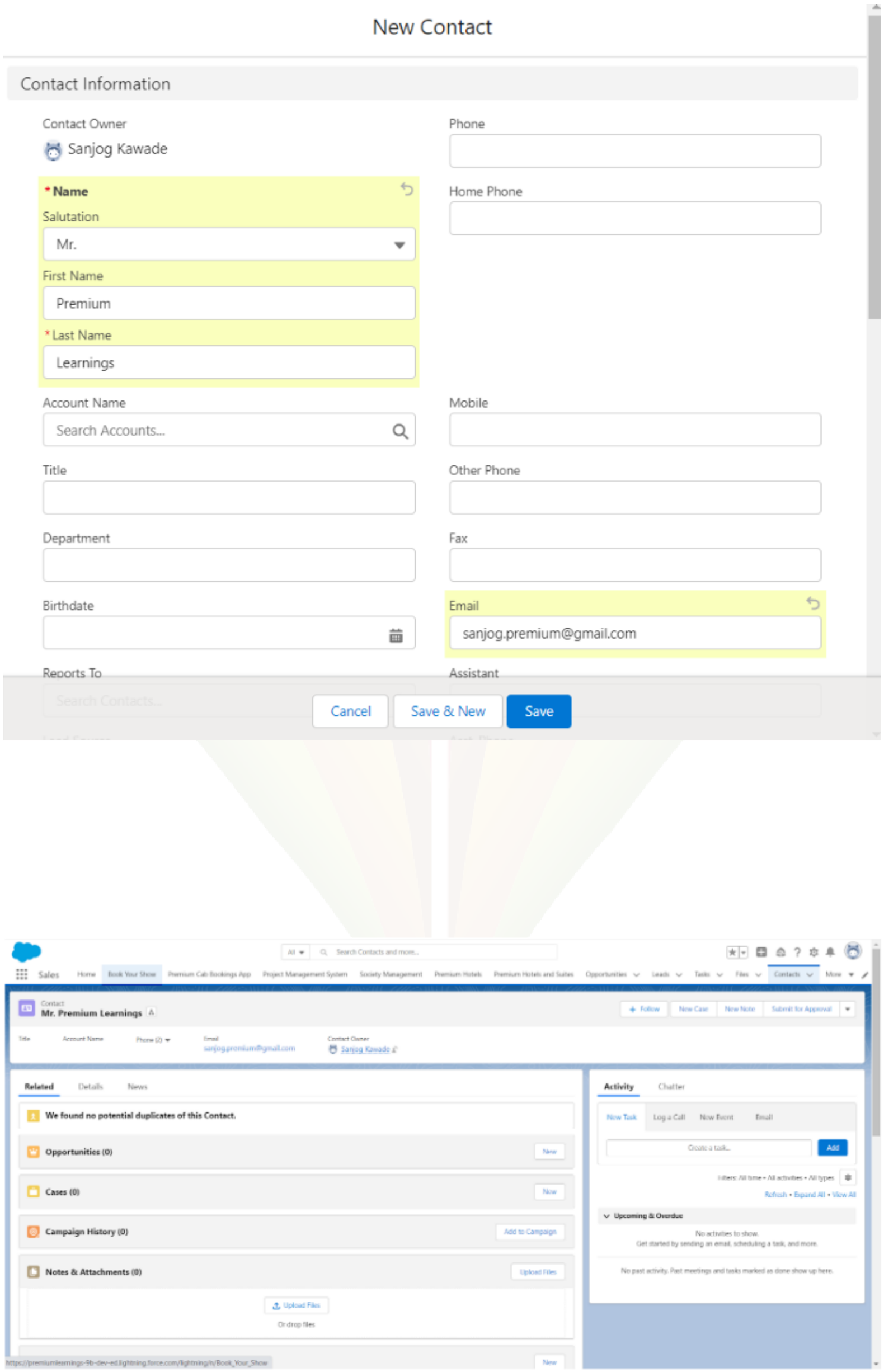
A new contact was created with Name: “Contact Name” on “Contact CreatedDate”. Contact your System Administrator if it was not created by you.
Thank You
Regards System Administrator.

```
trigger sendingMailAfterInsert on Contact (after insert) {

  List<Messaging.SingleEmailMessage> emailList= new
  List<Messaging.SingleEmailMessage>();
  EmailTemplate emailTemplate = [Select
    Id,Subject,Description,HtmlValue,DeveloperName,Body
    from EmailTemplate where name='Email Template to
    send After Contact is Inserted'];

  for(Contact conObj:Trigger.new){
      if (conObj.Email != null) {
          Messaging.SingleEmailMessage mail = new
                      Messaging.SingleEmailMessage();
          mail.setTargetObjectId(conObj.Id);
          mail.setSenderDisplayName('System Administrator');
          mail.setUseSignature(false);
          mail.setBccSender(false);
          mail.setSaveAsActivity(false);
          mail.setTemplateID(emailTemplate.Id);
          mail.toAddresses = new String[]{conObj.Email};
          emailList.add(mail);
      }
  }
```

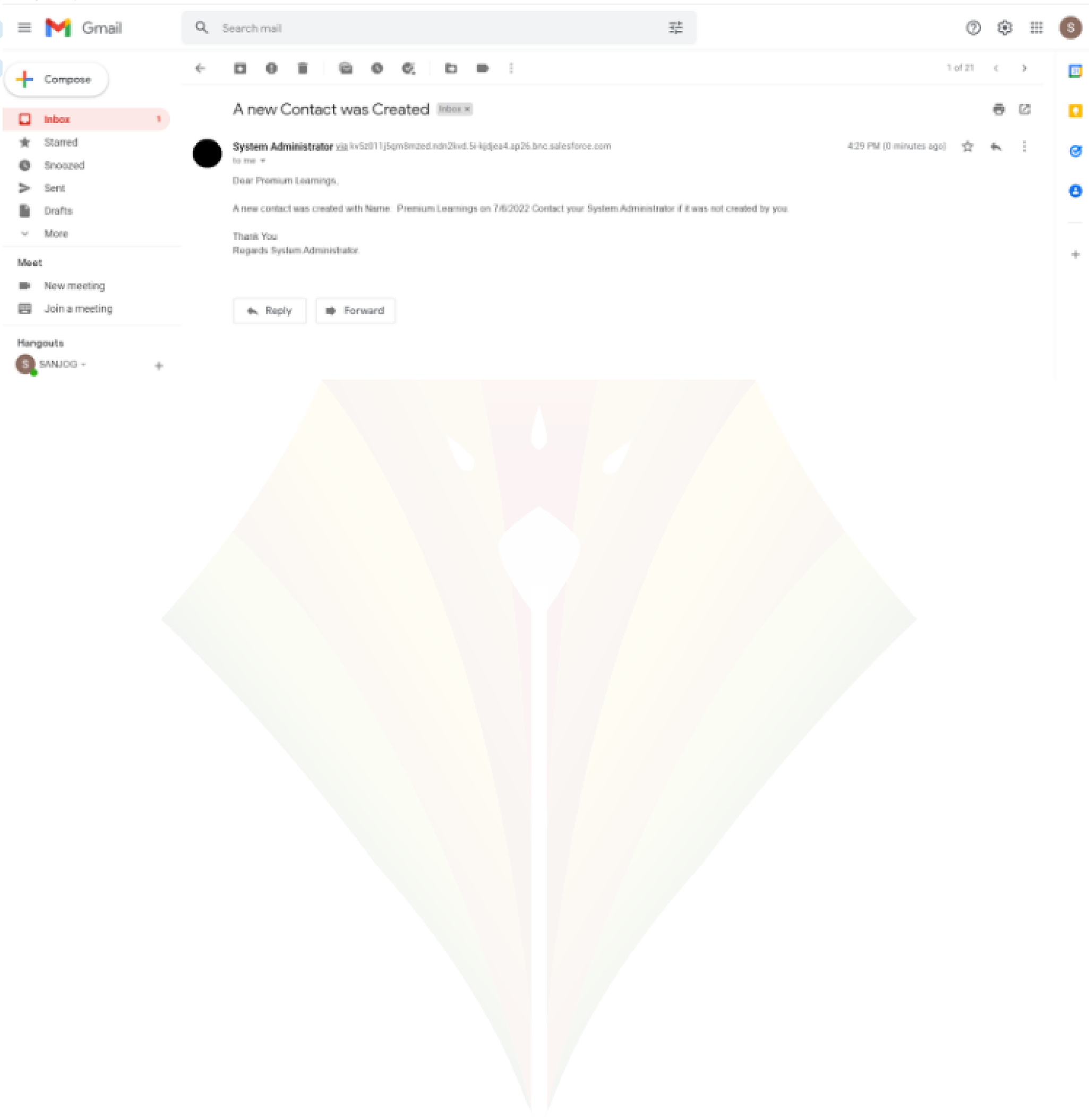
```
        if(emailList.size()>0){
            Messaging.SendEmailResult[] results =
                                    Messaging.sendEmail(emailList);
            if (results[0].success)
            {
              System.debug('The email was sent successfully.');
            }
            else {
                 System.debug('The email failed to send: '+
                                    results[0].errors[0].message);
            }
        }
    }
```

Test Case:

1. Create a new contact as shown below. Make sure email field is populated.



2. Email is sent to the email id as shown below



Example 11

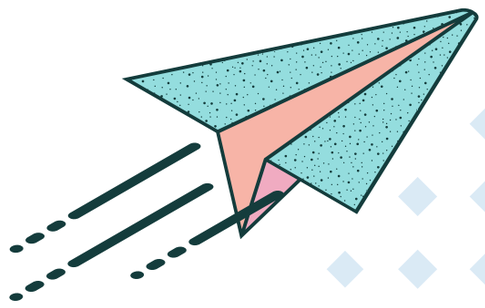
When an opportunity line item is created an email should go to Opportunity

Account Client Contact — Client Contact would be a field on Account lookup to contact.

```
trigger sendEmailToContact on OpportunityLineItem (after
insert)
{

Set<Id> oppId =new Set<Id>();
String userName;
Map<Id,Opportunity> oppIdConEmailMap = new
Map<Id,Opportunity>();

for(OpportunityLineItem oppProd:Trigger.new){
    oppId.add(oppProd.OpportunityId);
    userName=oppProd.CreatedBy.Name;
}

 List<Opportunity> oppList=[Select Id,
                    Account.Client_Contact__r.Email,
                    Account.Client_Contact__r.Last
                    Name,Account.Client_Contact__c From
                    Opportunity where Id=:oppId];

if(oppList.size()>0){
   for(Opportunity opp:oppList){
       oppIdConEmailMap.put(opp.Id,opp);
   }
}
 List<Messaging.SingleEmailMessage> emailList= new
 List<Messaging.SingleEmailMessage>();
```

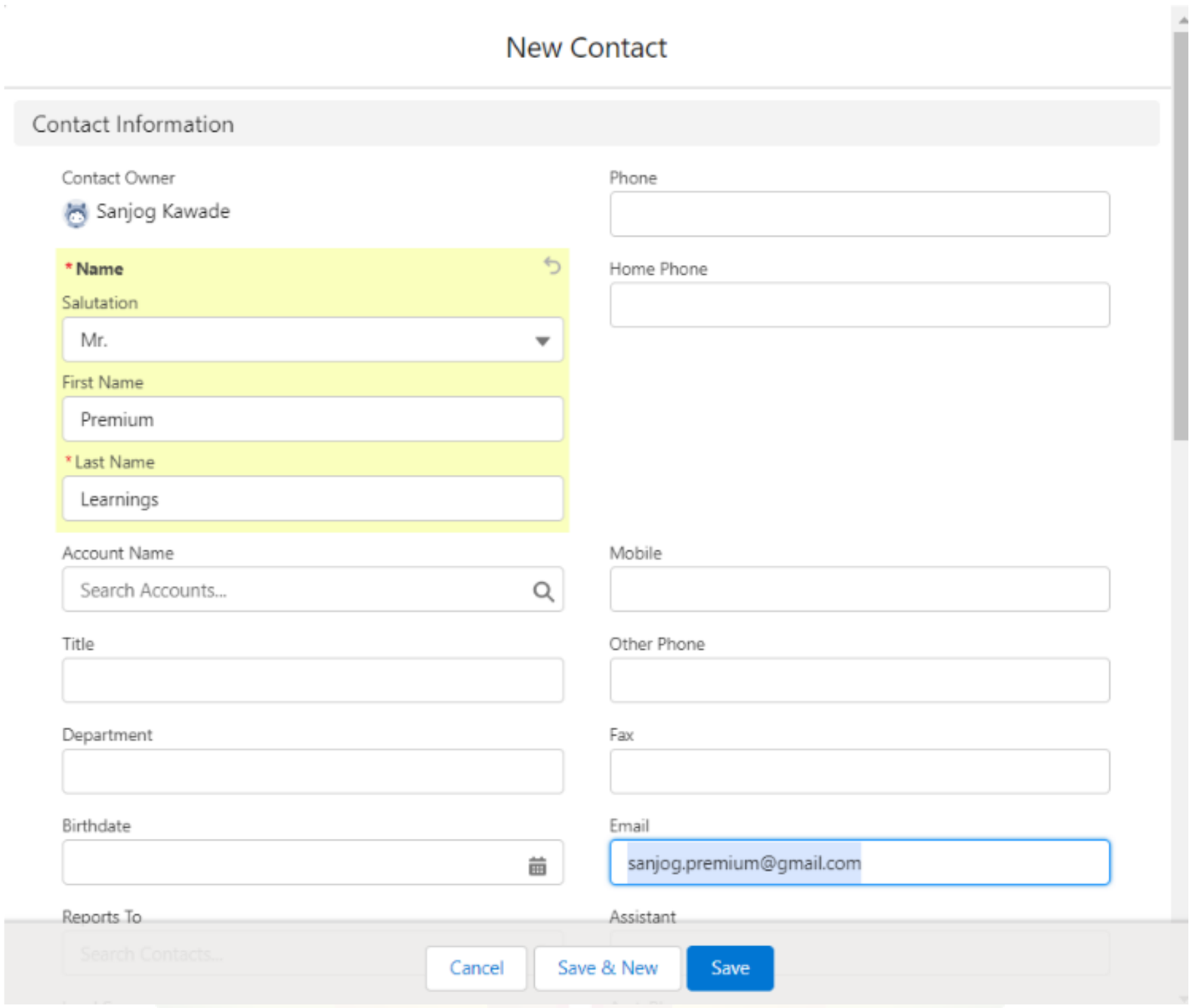
```
for(OpportunityLineItem oppProd:Trigger.new){

    if(oppIdConEmailMap.containsKey(oppProd.OpportunityId))
    {
      Opportunity oppObj =
                  oppIdConEmailMap.get(oppProd.OpportunityId);
     String sendTo=oppObj.Account.Client_Contact__r.Email;

if(sendTo!=null){
  Messaging.SingleEmailMessage mail = new
  Messaging.SingleEmailMessage();
 mail.setTargetObjectId(oppObj.Account.Client_Contact__c);
 mail.setSenderDisplayName('System Administrator');
 mail.setUseSignature(false);
 mail.setBccSender(false);
 mail.setSaveAsActivity(false);
 mail.setSubject('New Opportunity Product was Created.');
 String body = 'Dear' +
       oppObj.Account.Client_Contact__r.LastName+', <br/>';
       body += 'Your Order has been proceed.<br/>';
       body +='Order detail is below.<br/>';
       body +='Product Name : '+oppProd.Name+'.<br/>';
       body +='Product Code : '+oppProd.ProductCode+'.<br/>
       body +='Unit Price : '+oppProd.UnitPrice+'.<br/>';
       body +='List Price : '+oppProd.ListPrice+'.<br/>';
       body +='Thanks <br/>'+userName;
       mail.setHtmlBody(body);
       mail.toAddresses = new String[]{sendTo};
       emailList.add(mail);
    }
    }
}
```

```
if(emailList.size()>0){
    Messaging.SendEmailResult[] results =
                        Messaging.sendEmail(emailList);
    if (results[0].success)
    {
        System.debug('The email was sent successfully.');
     }
    else {
        System.debug('The email failed to send: '+
        results[0].errors[0].message);
     }
   }
}
```

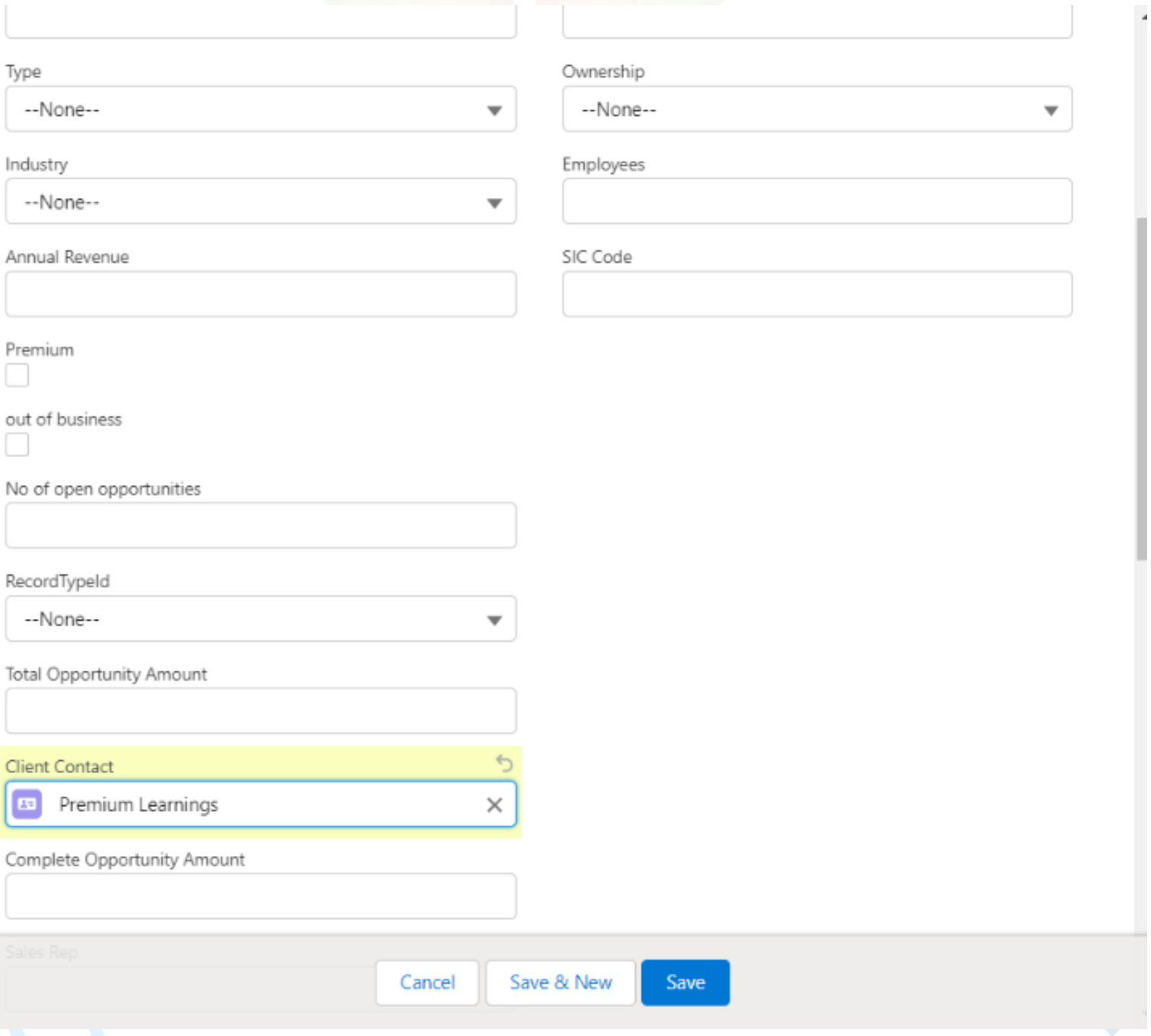



Test Case:

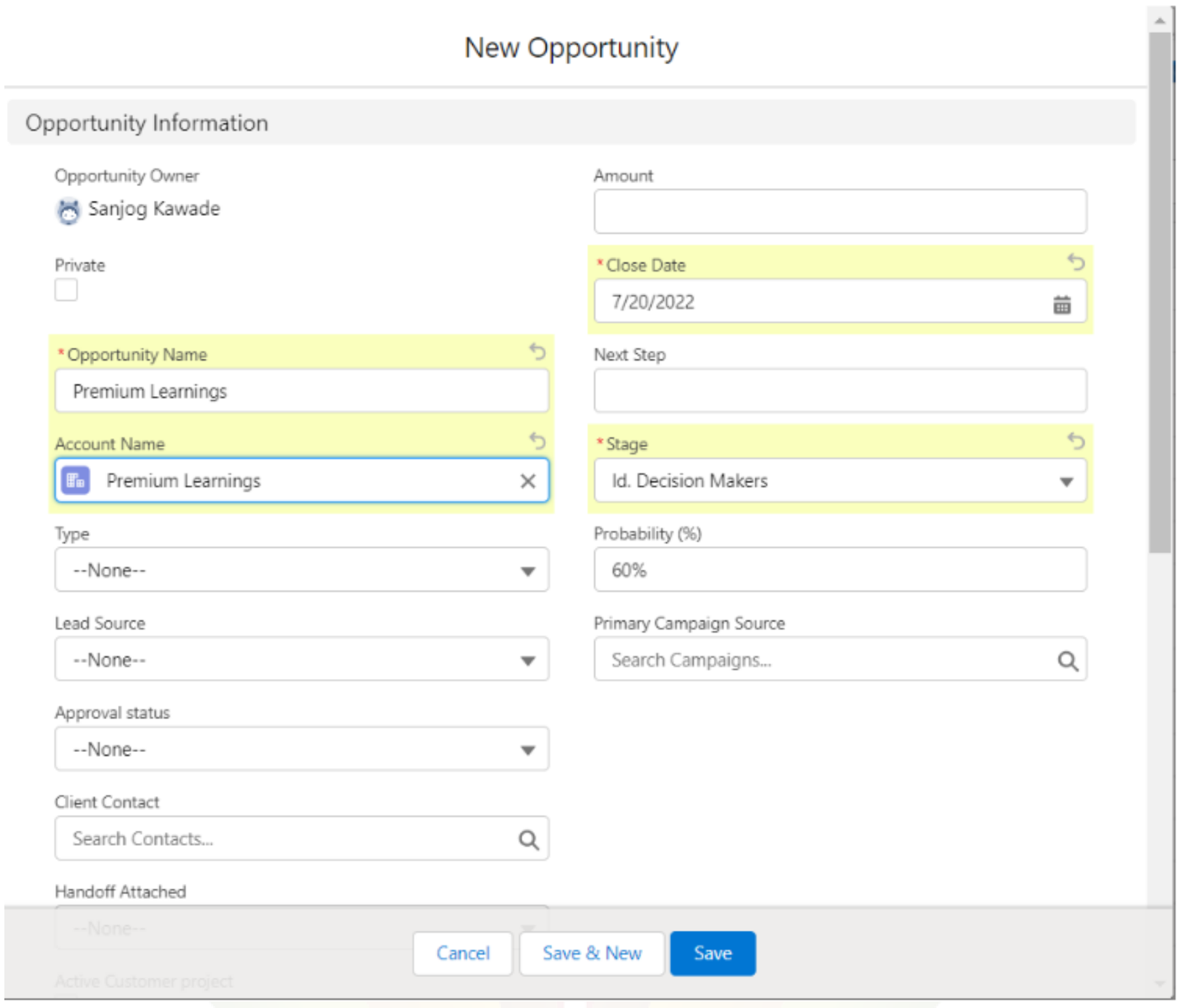
1. Create contact and make sure the email field is populated



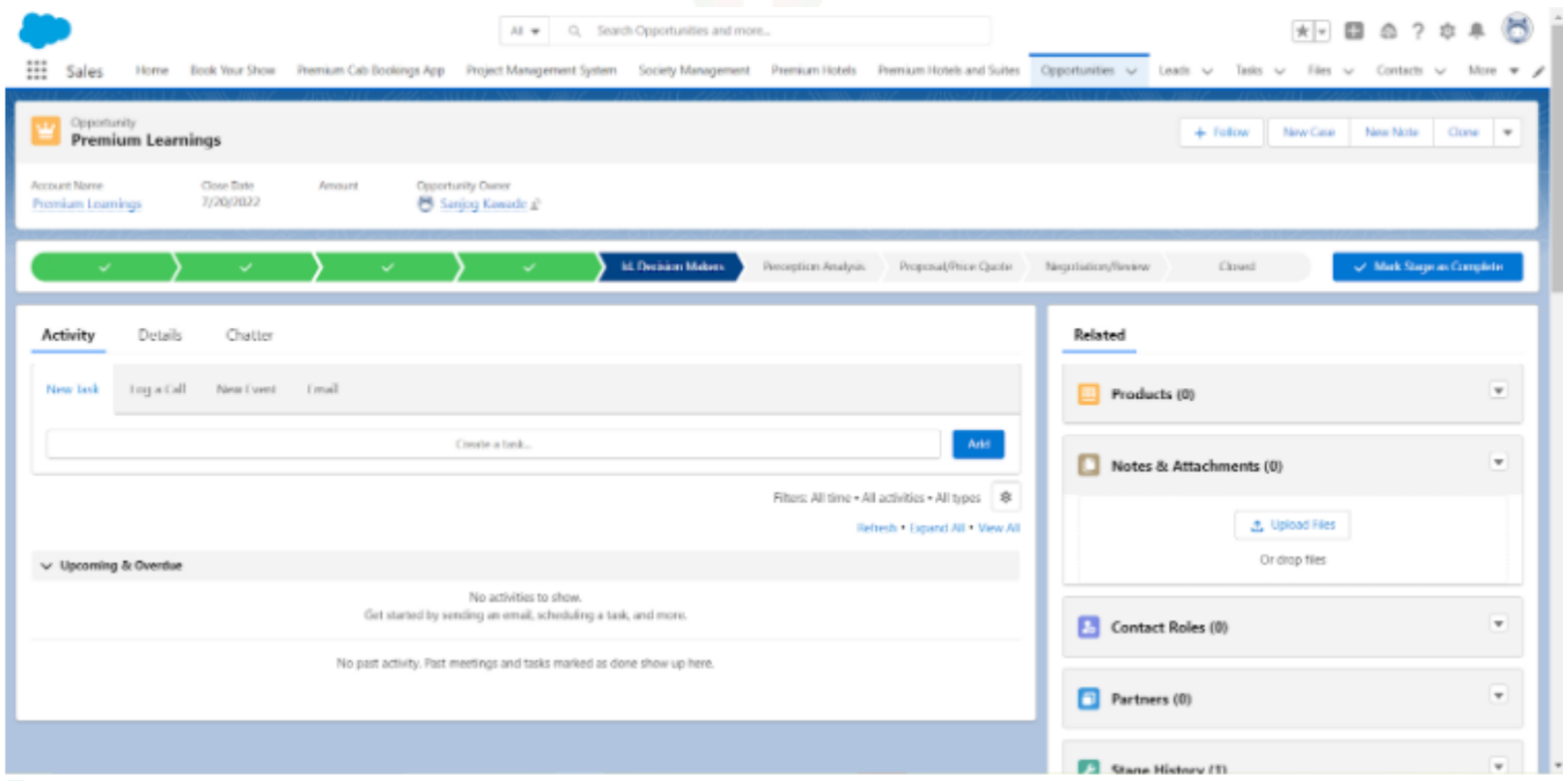
2. Create a new account and make sure the client contact field is populated with the above created contact



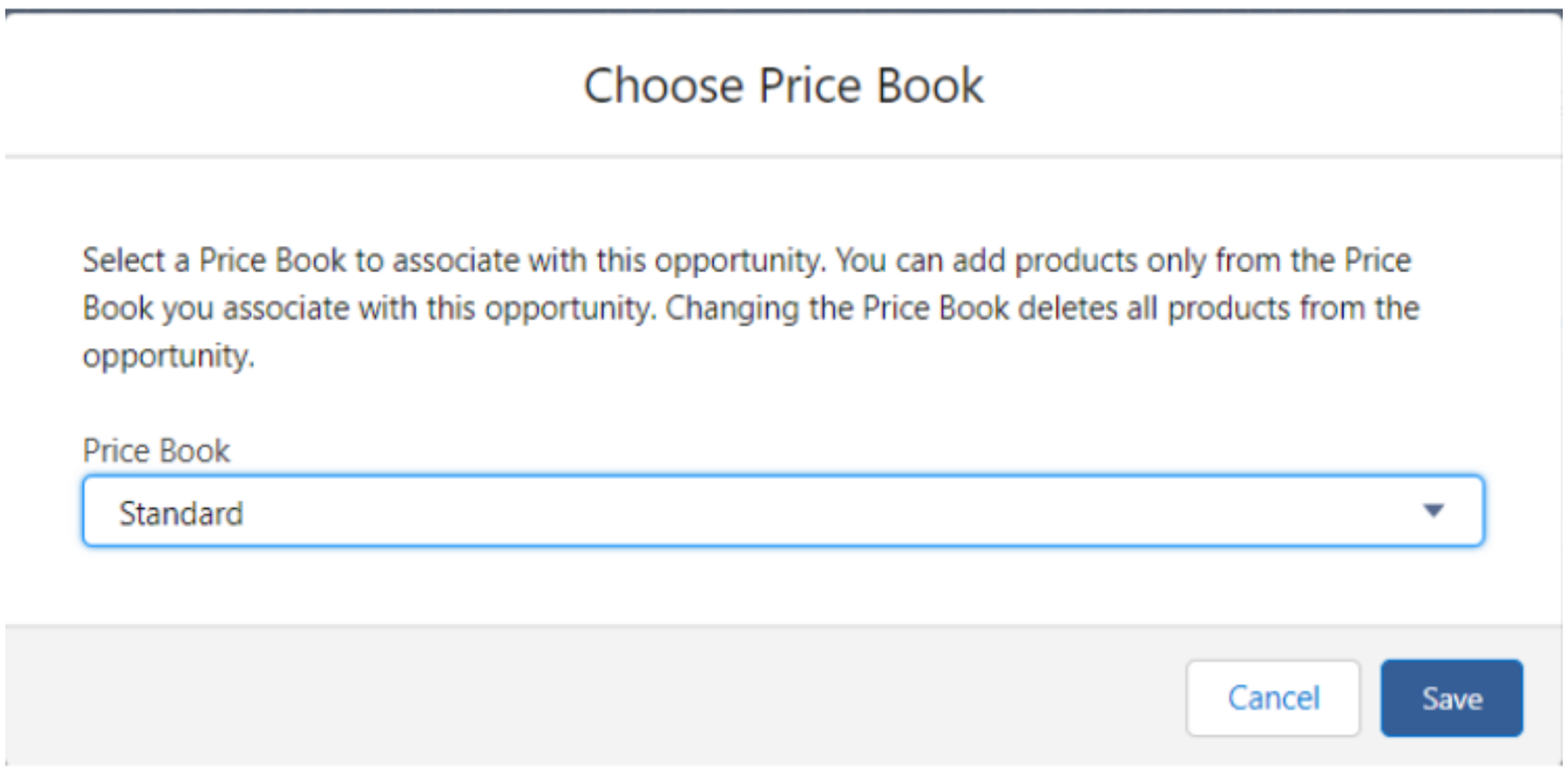
3. Now create a new opportunity and make sure the related the account is the above-created account.



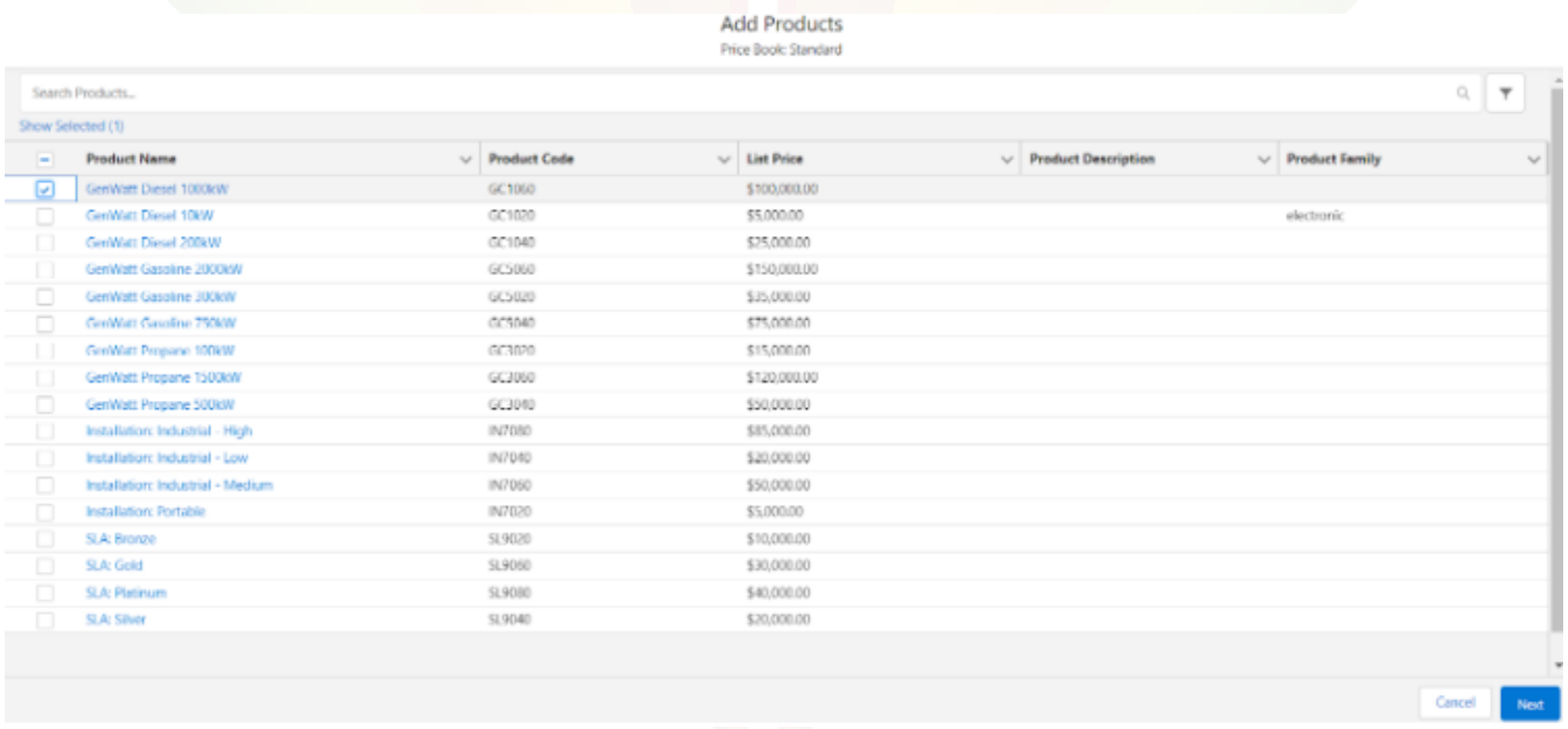
Notice the ‘products’(also called opportunity line item) in related lists of opportunity.



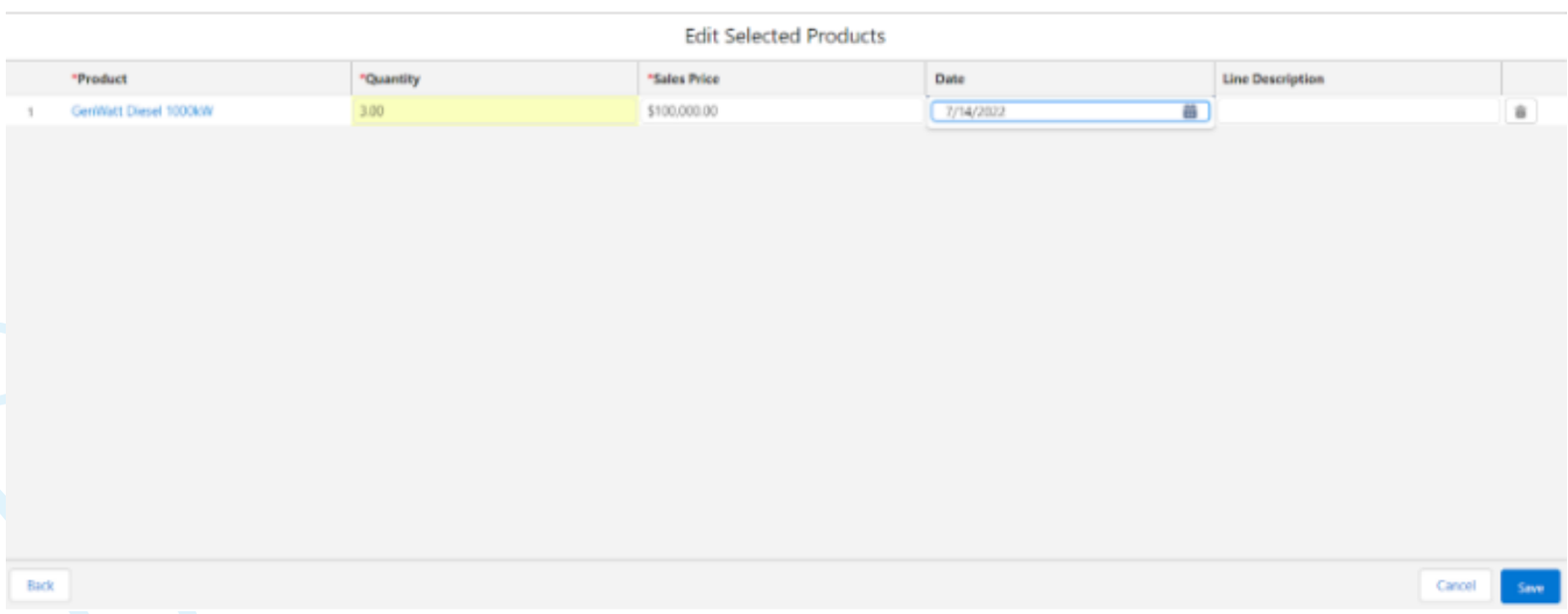
a. Create a new product(click on arrow next to products as visible in above image and click add products) as shown below



Choose any product and click next



Fill in the fields and click save.

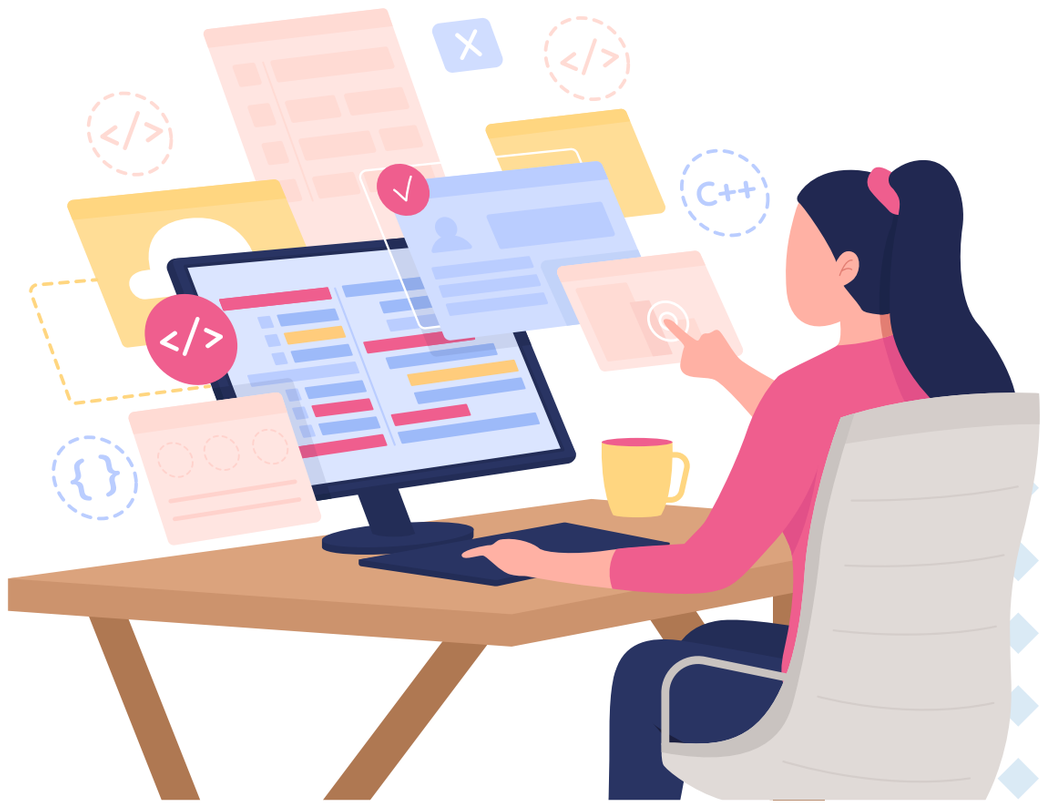


Example 12

Create a fields on Opportunity as Client Contact, Once an Opportunity Client Contact updates then update Account Client Contact with same on Opportunity Client Contact.

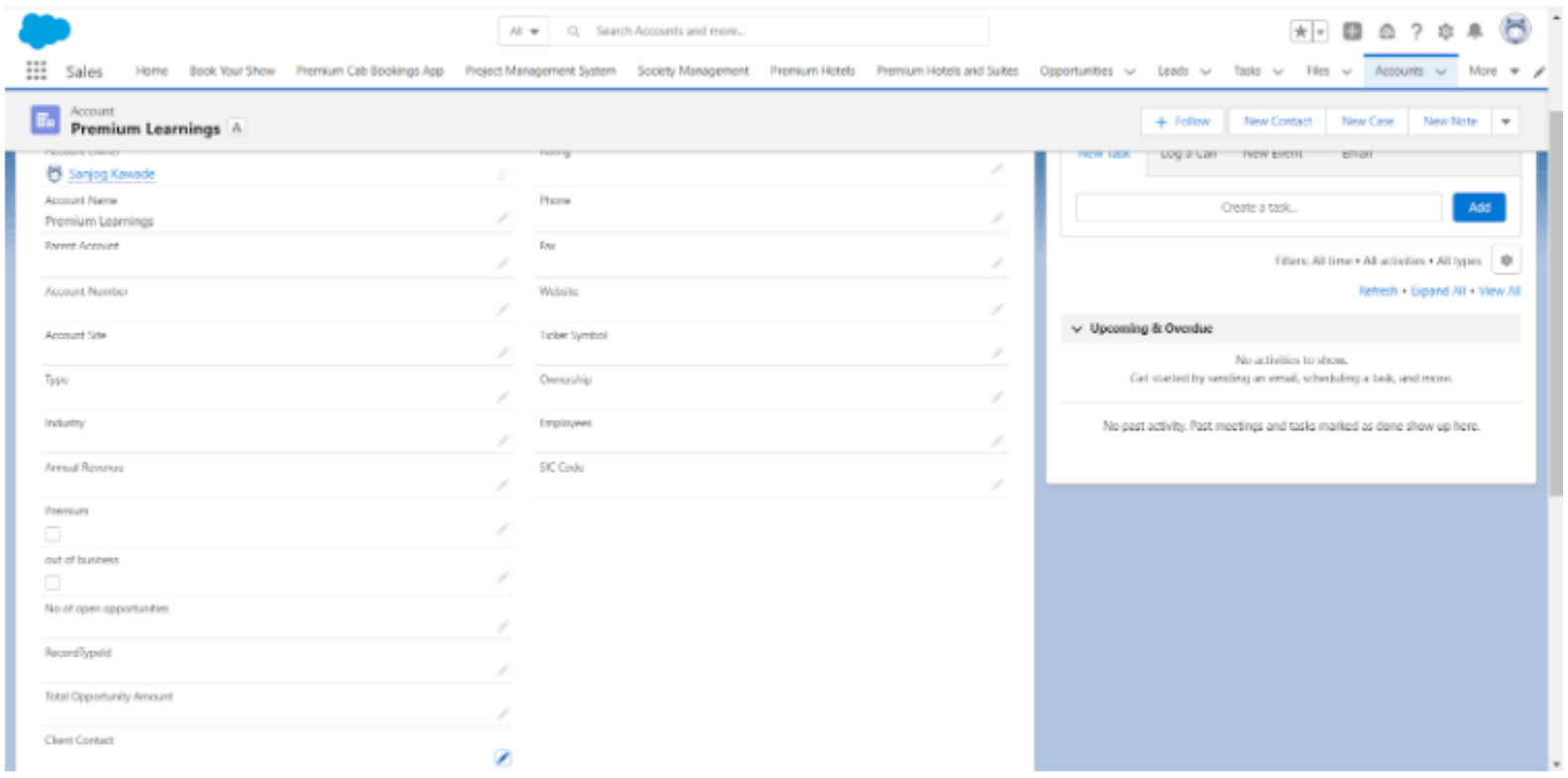
```
trigger accClientConOnUpdateOfOppClientCon on Opportunity
(after update)
{
 Set<Id> oppId= new Set<Id>();

 for(Opportunity opp : Trigger.new) {

 //if the Client Contact of Opportunity is null then we
 //are not changing the Account Client Contact
     if(opp.Client_Contact__c!=null && opp.AccountId!=null)
     {
       if(opp.Client_Contact__c !=
             Trigger.oldMap.get(opp.Id).Client_Contact__c)
       {
           oppId.add(opp.Id);
       }
     }
 }

 List<Account> accList = new List<Account>();
 List<Opportunity> oppList = [Select Id,
     Client_Contact__c,AccountId,Account.Client_Contact__c
     from Opportunity where Id=:oppId];

if(oppList.size()>0){
    for(Opportunity opp:oppList){
```

```
        if(opp.Client_Contact__c !=
                                  opp.Account.Client_Contact__c)
        {
         Account accObj=new Account();
         accObj.Id=opp.AccountId;
            accObj.Client_Contact__c = opp.Client_Contact__c;
            accList.add(accObj);
          }
        }
    }
     if(accList.size()>0){
        update accList;
     }
}
```

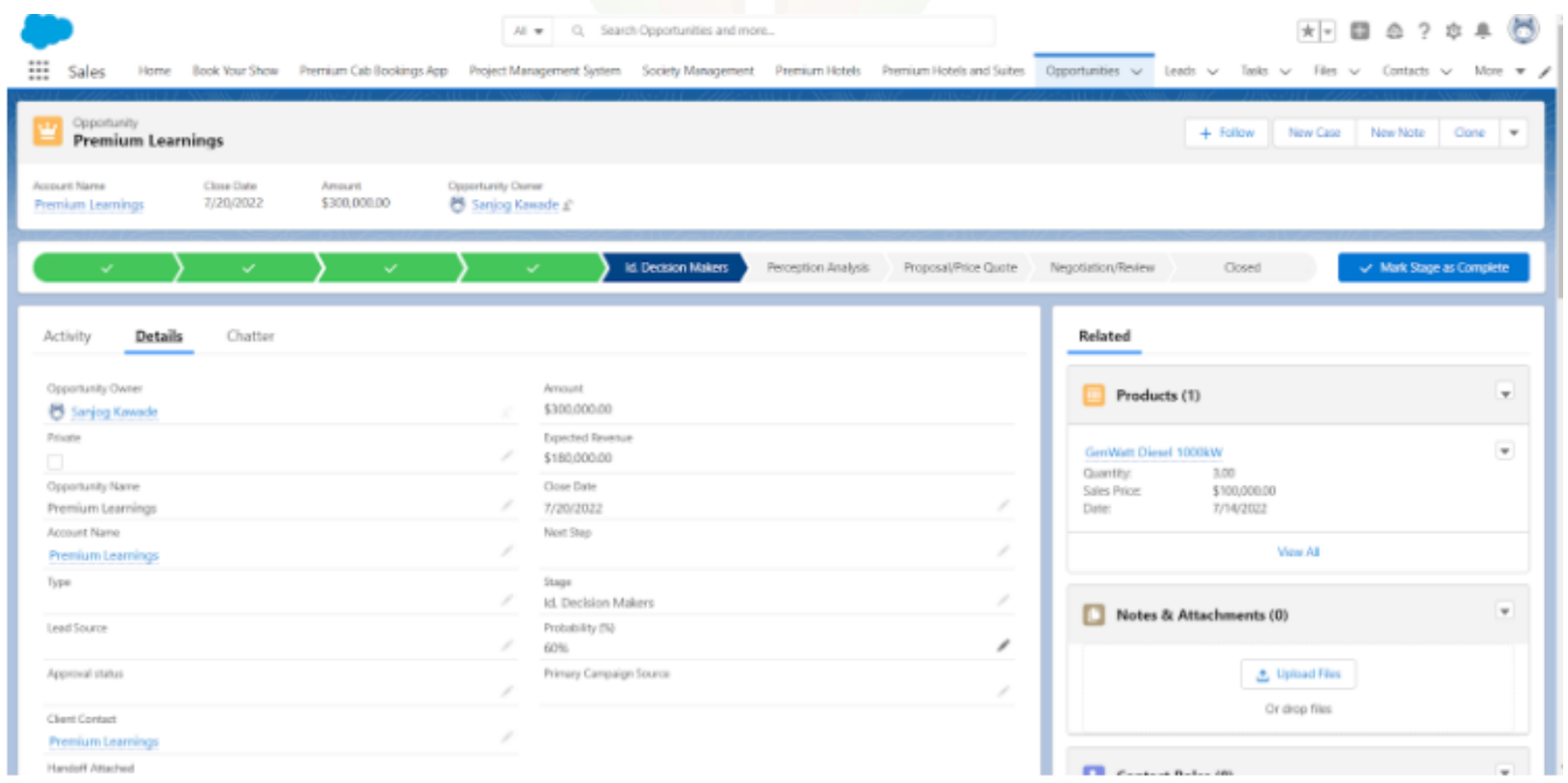



Test Case:

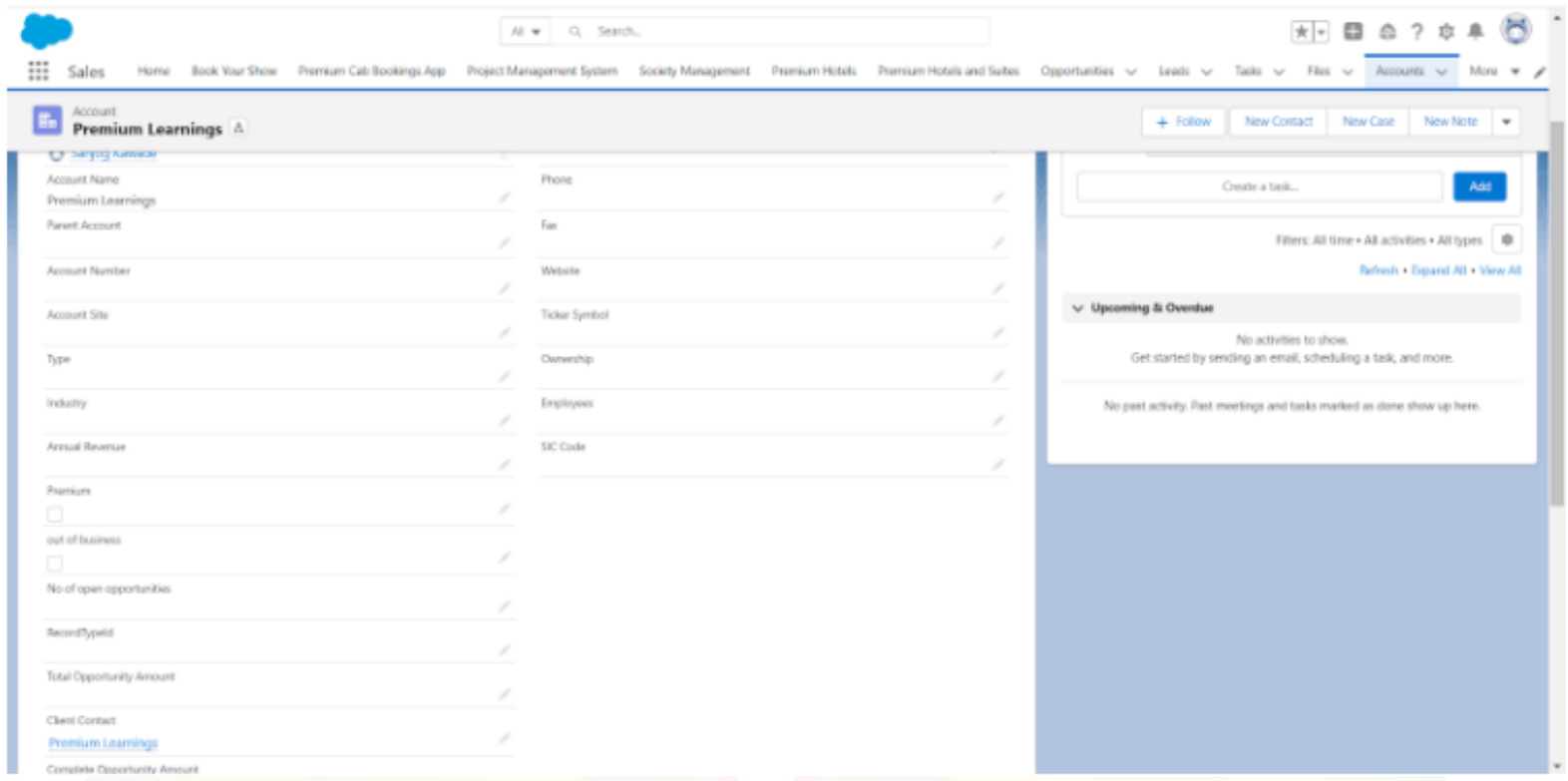
1. Create an account and keep the client contact custom field blank.



2. Create an opportunity and save it. Now update the client contact field of the just created opportunity(select any contact) and select the above created account as the parent account.



3. Now go to the account record we created in step 1 and check the client contact field, it was blank earlier and is now populated with client contact field we selected in step 2, as shown in below image.



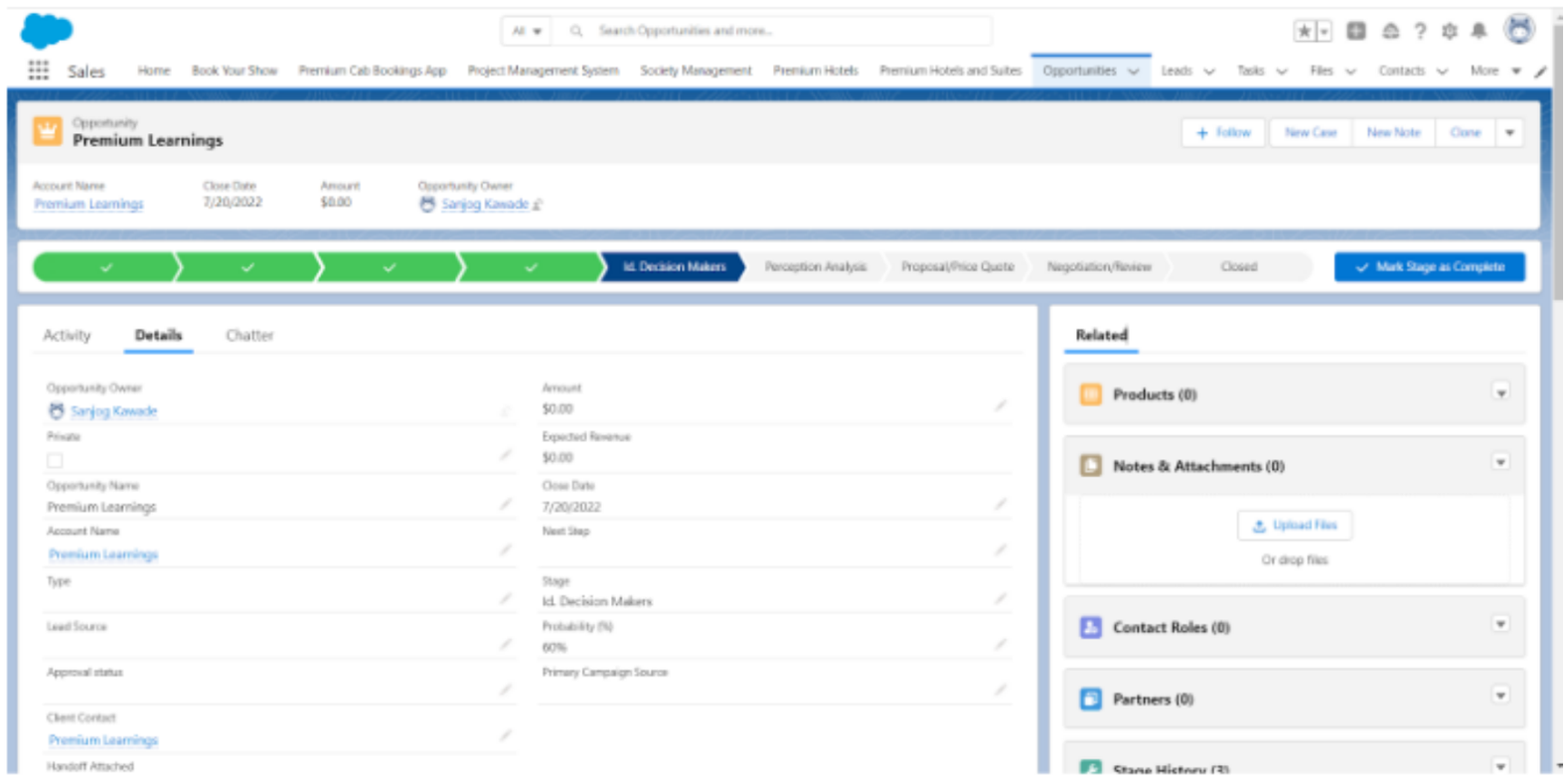
Example 13

Create an asset when create an OpportunityLineItem with associated Account. Create asset custom object with lookup relationship with account

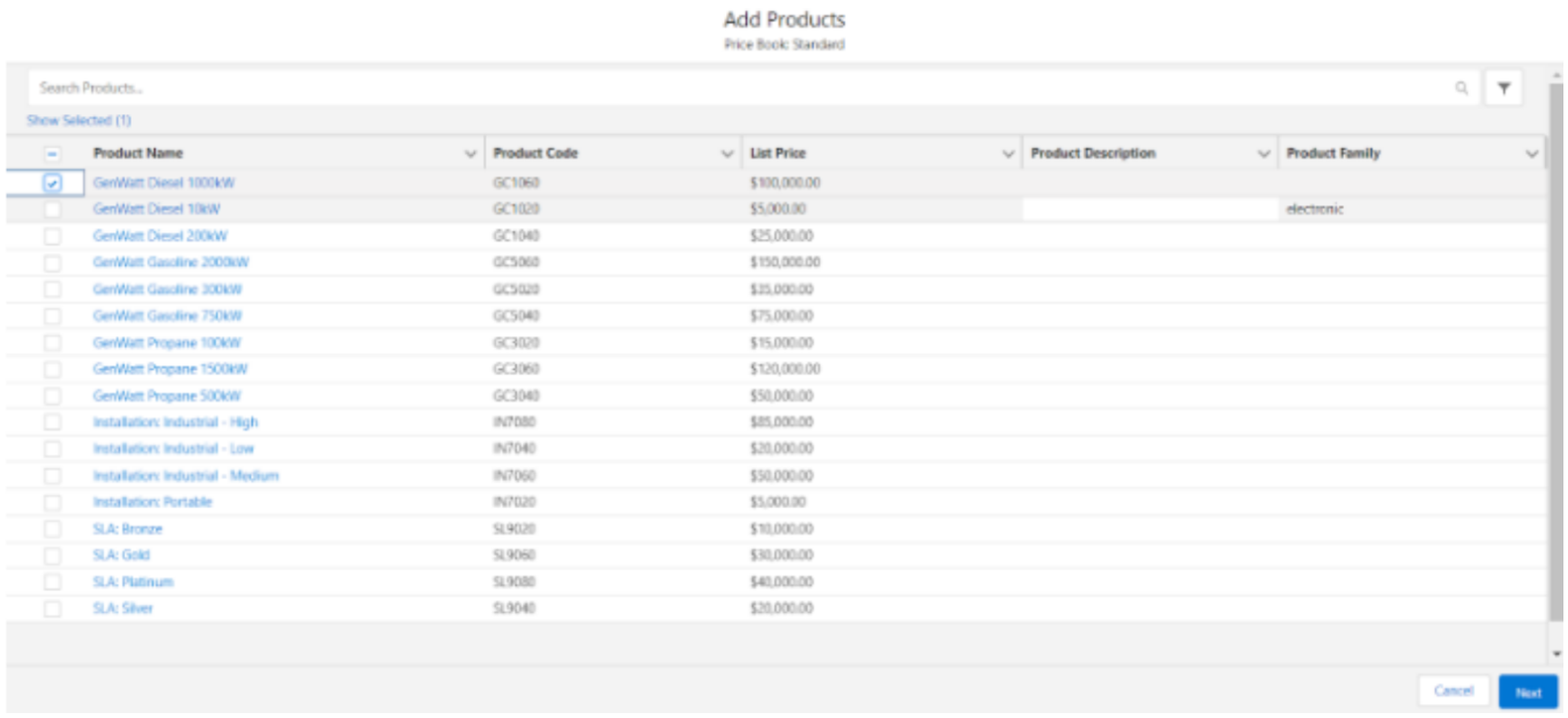
```
trigger accountRelatedAsset on OpportunityLineItem
(after insert)
{
 Set<Id> oppId=new Set<Id>();

  for(OpportunityLineItem oppProd:Trigger.new){
     oppId.add(oppProd.OpportunityId);
  }

 List<Asset> assList=new List<Asset>();
 List<Opportunity> oppList =[Select Id,AccountId,
        Account.Name from Opportunity where Id=:oppId];

  if(oppList.size()>0){
    for(Opportunity opp:oppList){
       if(opp.AccountId!=null){
          Asset assObj=new Asset();
         assObj.Name=opp.Account.Name+
                              'OpportunityLineItem';
         assObj.AccountId=opp.AccountId;
         assList.add(assObj);
       }
    }
 }

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

Test Case:

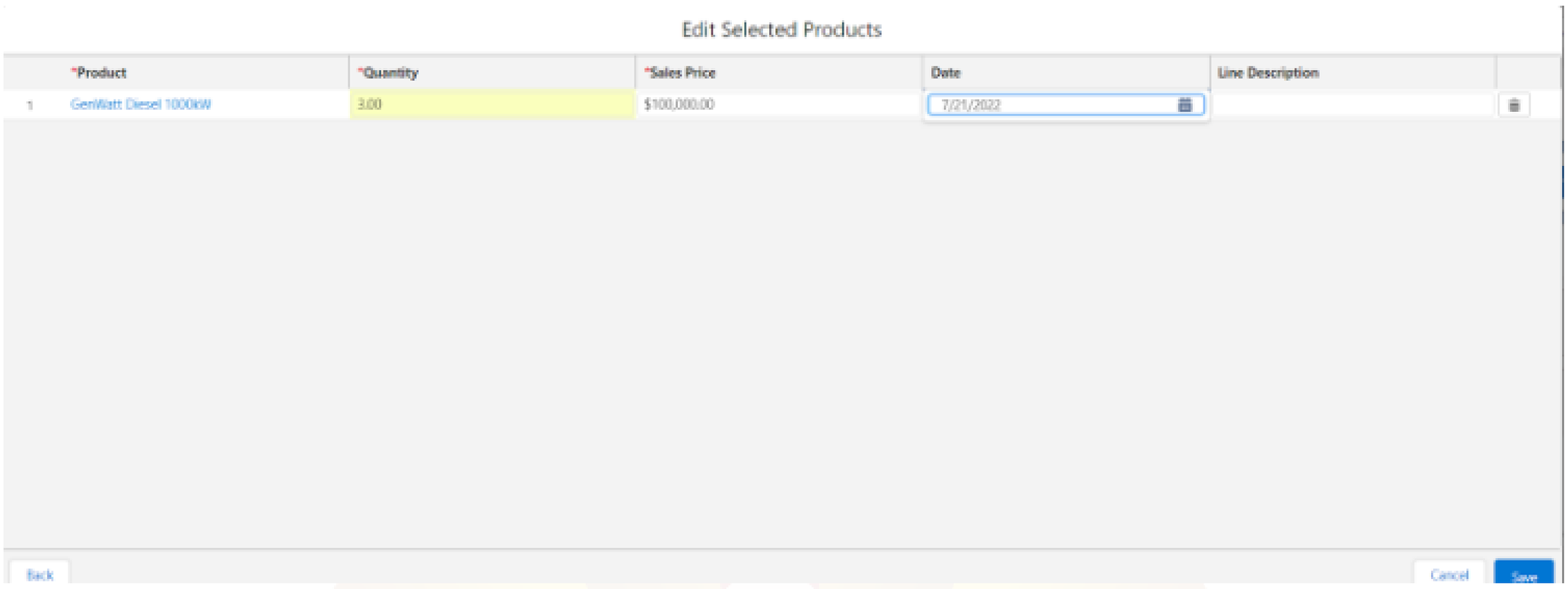
1. create an opportunity with related account field populated as shown in below image



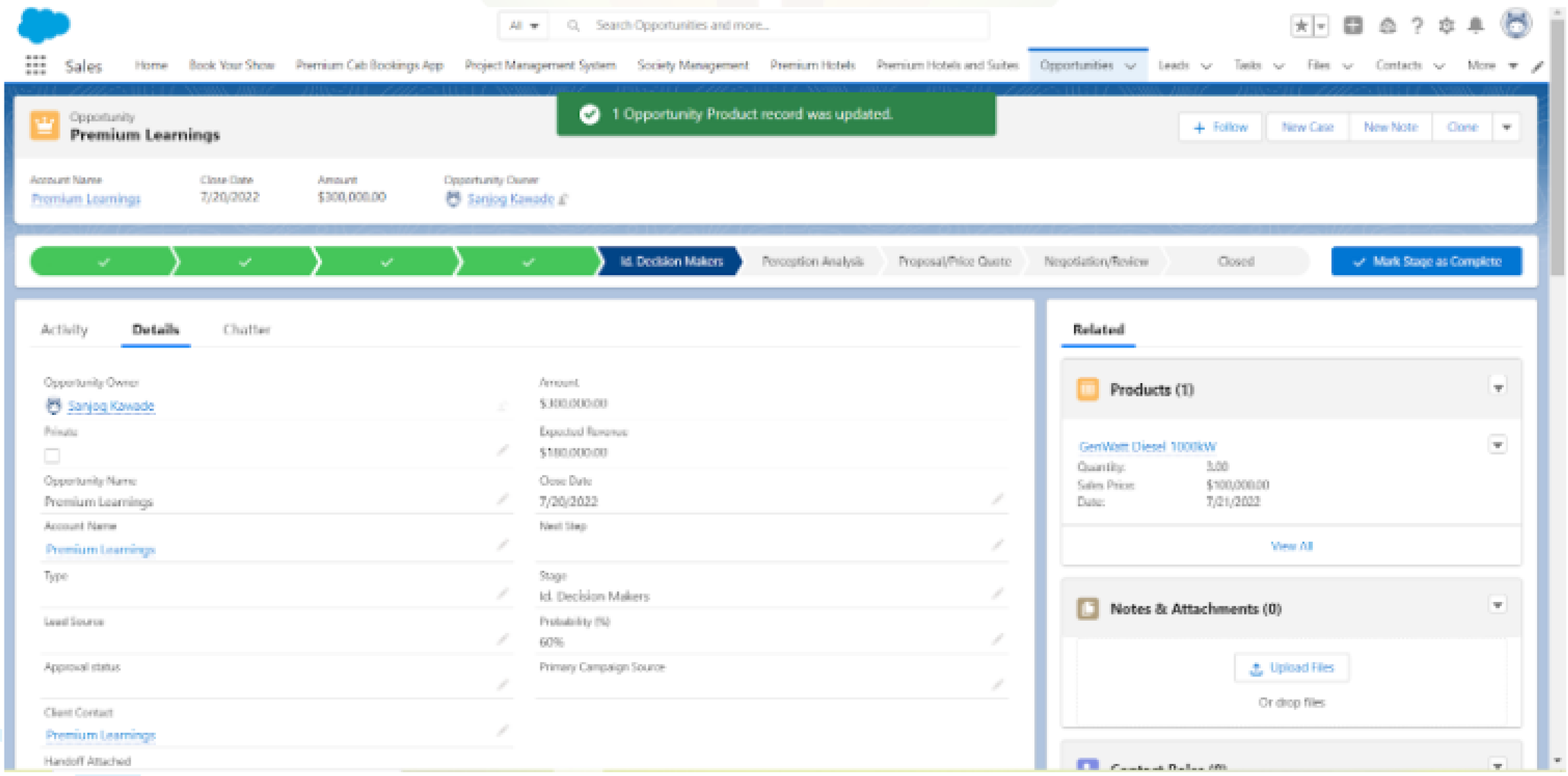
2. Now create a opportunity line item/opportunity product (product in the related list of opportunity, visible in the right of the above image) click on the arrow next to products > add products > price book – standard (if a window appears to select price book) > select any product as shown below. Click Next.



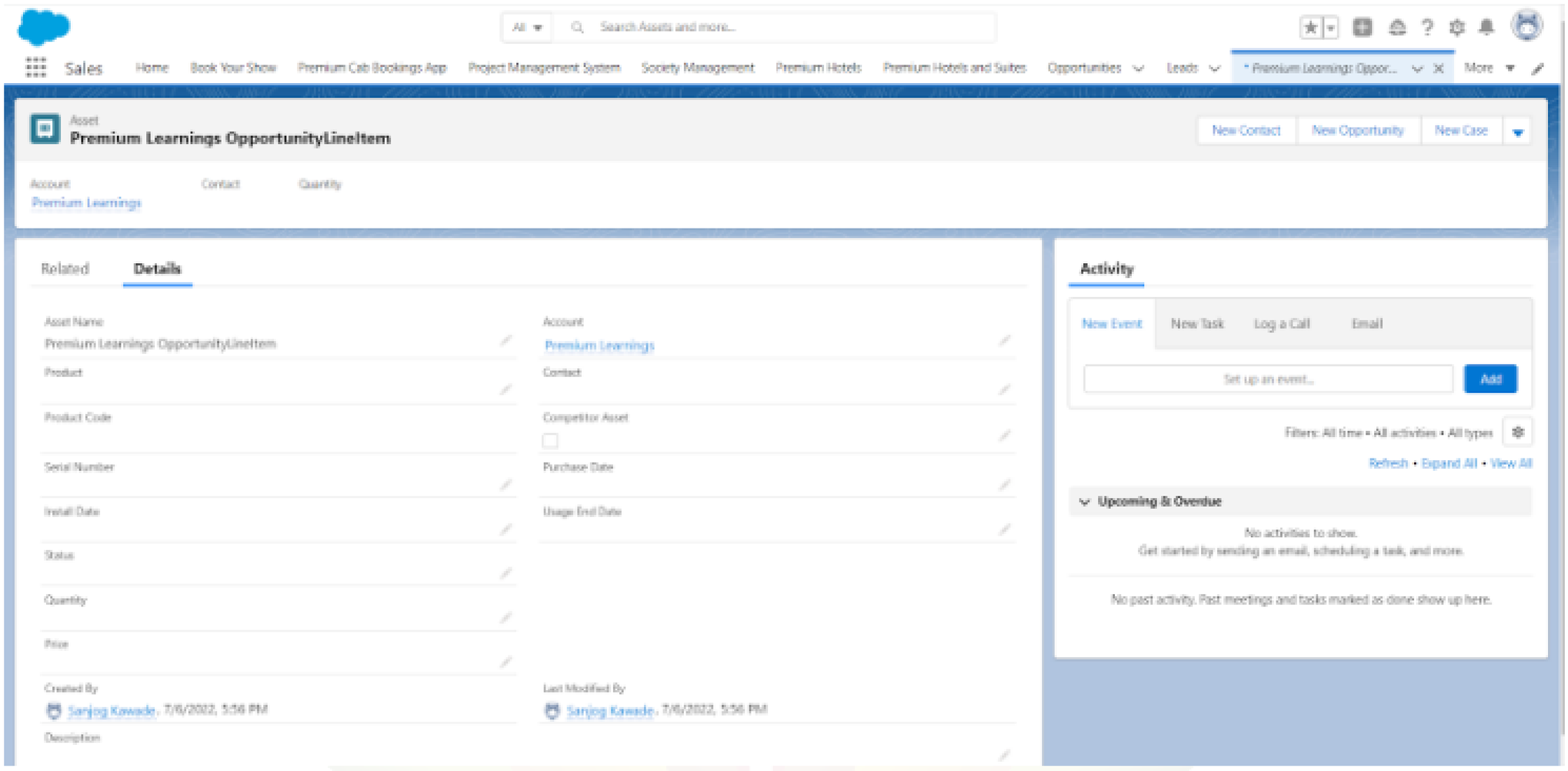
3. Add a desired quantity and date. click save



4.You can now see the product updated (in the right of below image) in the related list of opportunities.



5. Asset record is now created as the apex trigger is fired.



Example 14

Once Opportunity Line Item is added to Opportunity with the specified product, then Product Total Quantities must be deduct from the Product Object

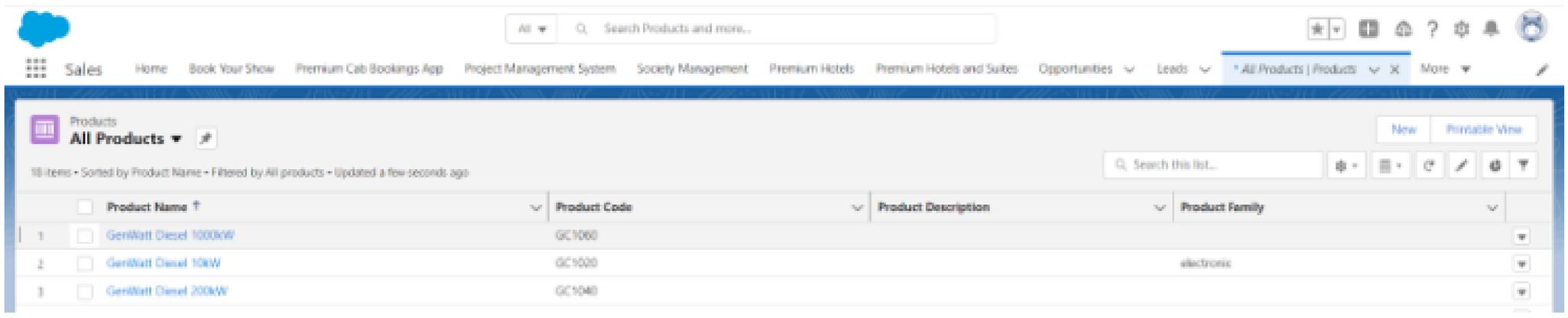
e.g. If we have Total Quantity 100 on Product object and we have added 50 in
line items , then product must be update 100-50 = 50
Create Total Quantity, Available Quantity Fields on Product2 object.

```
trigger updateProductTotalQuantities on OpportunityLineItem
(after insert)
{
 Set<Id> proId = new Set<Id>();

  for(OpportunityLineItem oppProd:Trigger.new){
     proId.add(oppProd.Product2Id);
  }

   Map<Id,Double> proIdQuantitySum =new Map<Id,Double>();
   List<AggregateResult> results=[select Product2Id,
              sum(Quantity)quantity from OpportunityLineItem
              where Product2Id in :proId group by Product2Id];

    if(results.size()>0){
       for(AggregateResult a: results){
          Id prodId = (Id)a.get('Product2Id');
          double TotalQuantity = (double)a.get('quantity');

//Adding ProductId and sum(Quantity) to Map
       proIdQuantitySum.put(prodId,TotalQuantity);
        }
      }
```

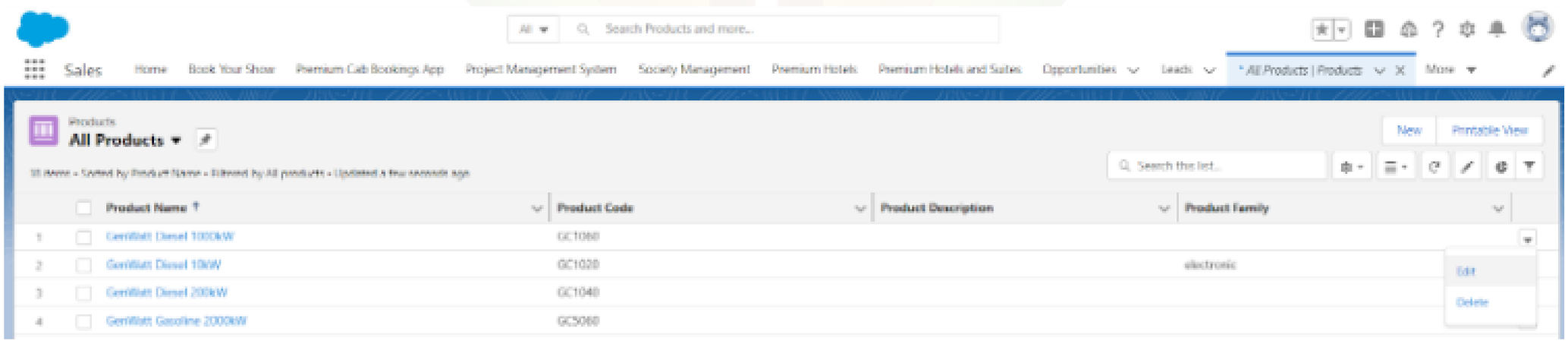
```
List<Product2> proListToUpdate = new List<Product2>();
List<Product2> proList=[Select Id,Total_Quantities__c
                        from Product2 where Id=:proId];

if(proList.size()>0){
   for(Product2 pro : proList){

//Checking if the ProductId is present in Map
      if(proIdQuantitySum.containsKey(pro.Id)){
        if(pro.Total_Quantities__c!=null){
//Subtracting the Quantity
            pro.Total_Quantities__c =
                        pro.Total_Quantities__c-
                        proIdQuantitySum.get(pro.Id);
            proListToUpdate.add(pro);
          }
        }
    }
   if(proListToUpdate.size()>0){
      update proListToUpdate;
   }
}
```

Test Case:

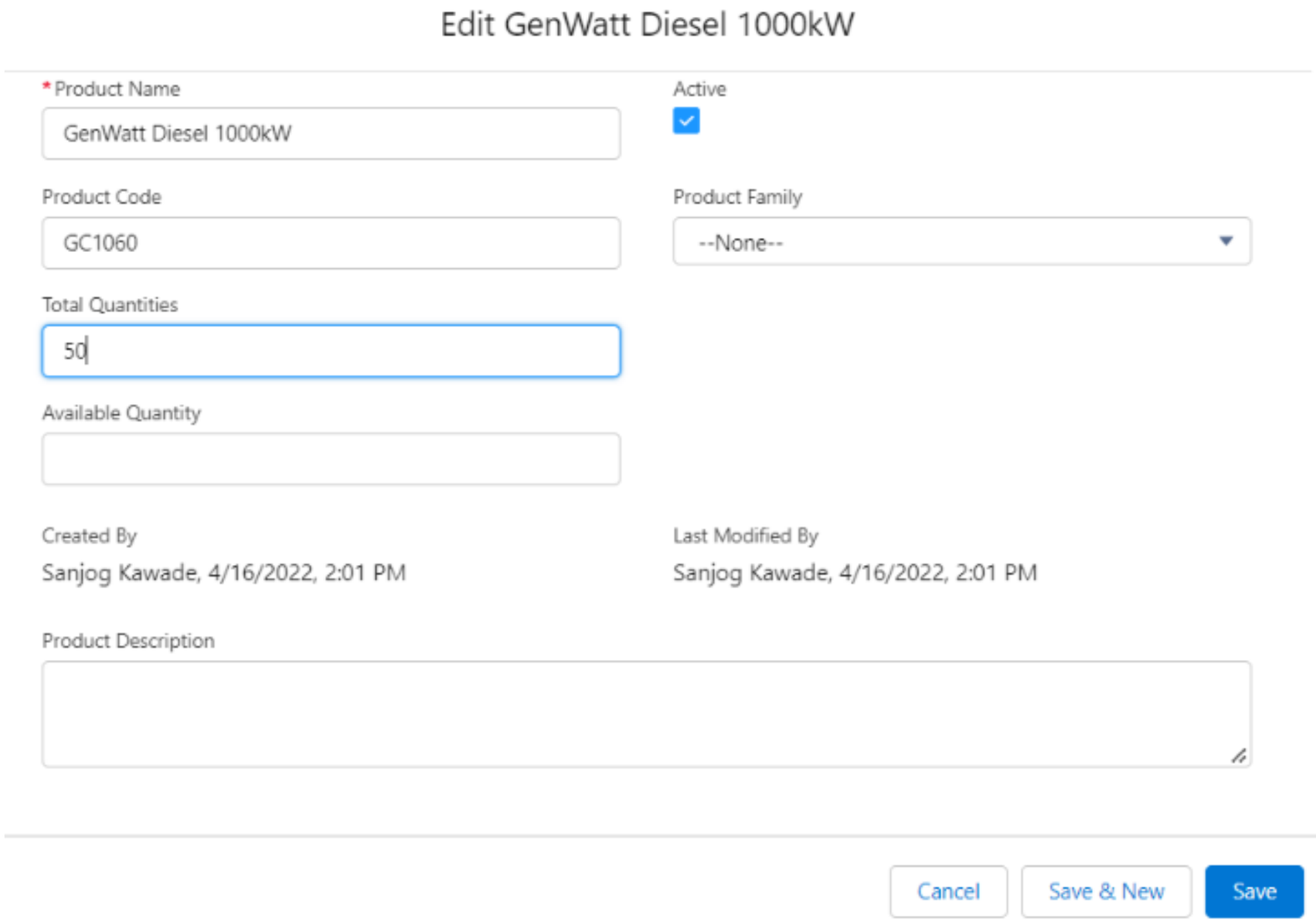
1. Select any record from products object and enter the Total Quantity field with value ‘50’.



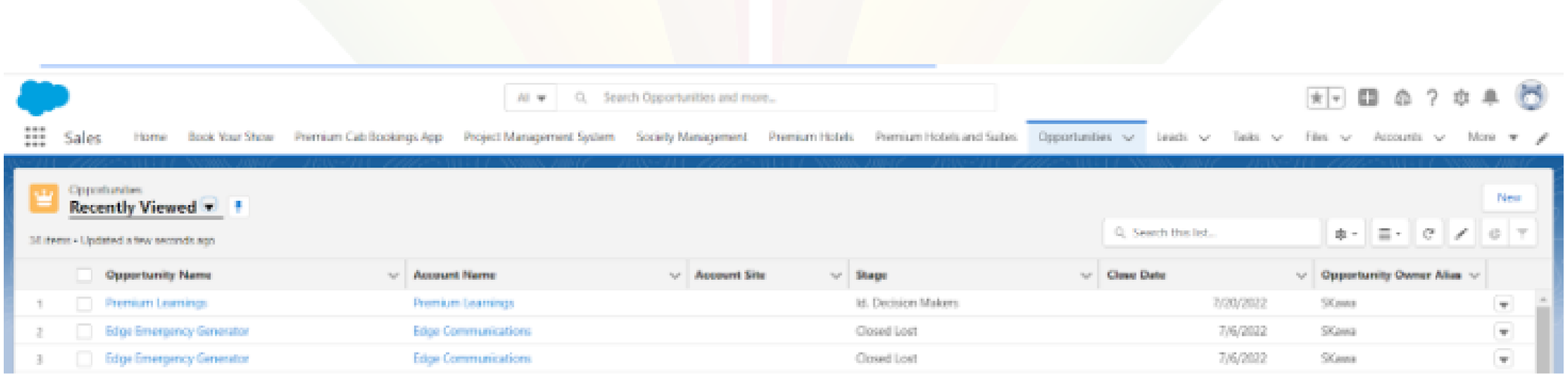
2. We use GenWatt Diesel 1000KW for testing. Click on the arrow next to its name and click edit.



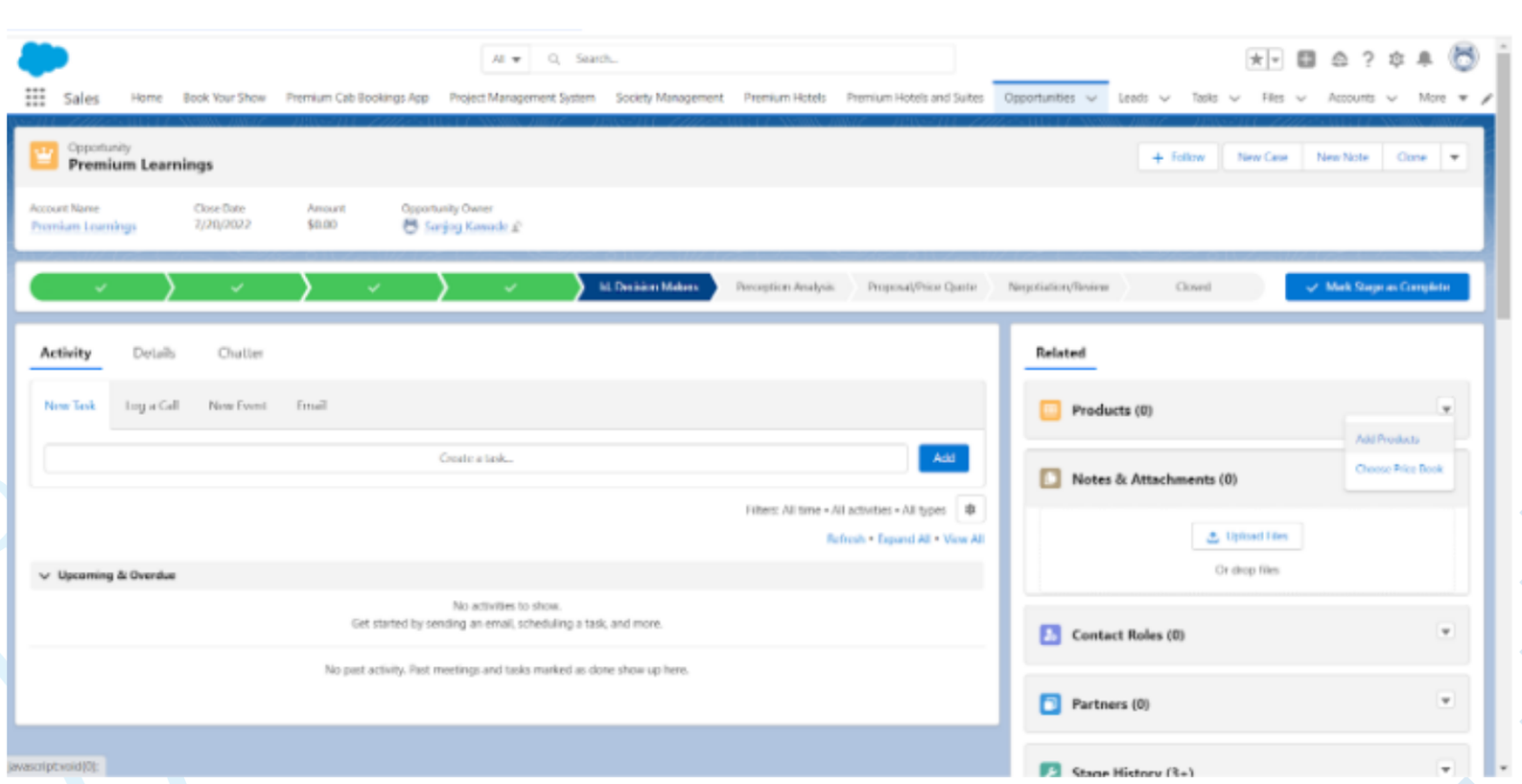
3. Add total quantities as 50



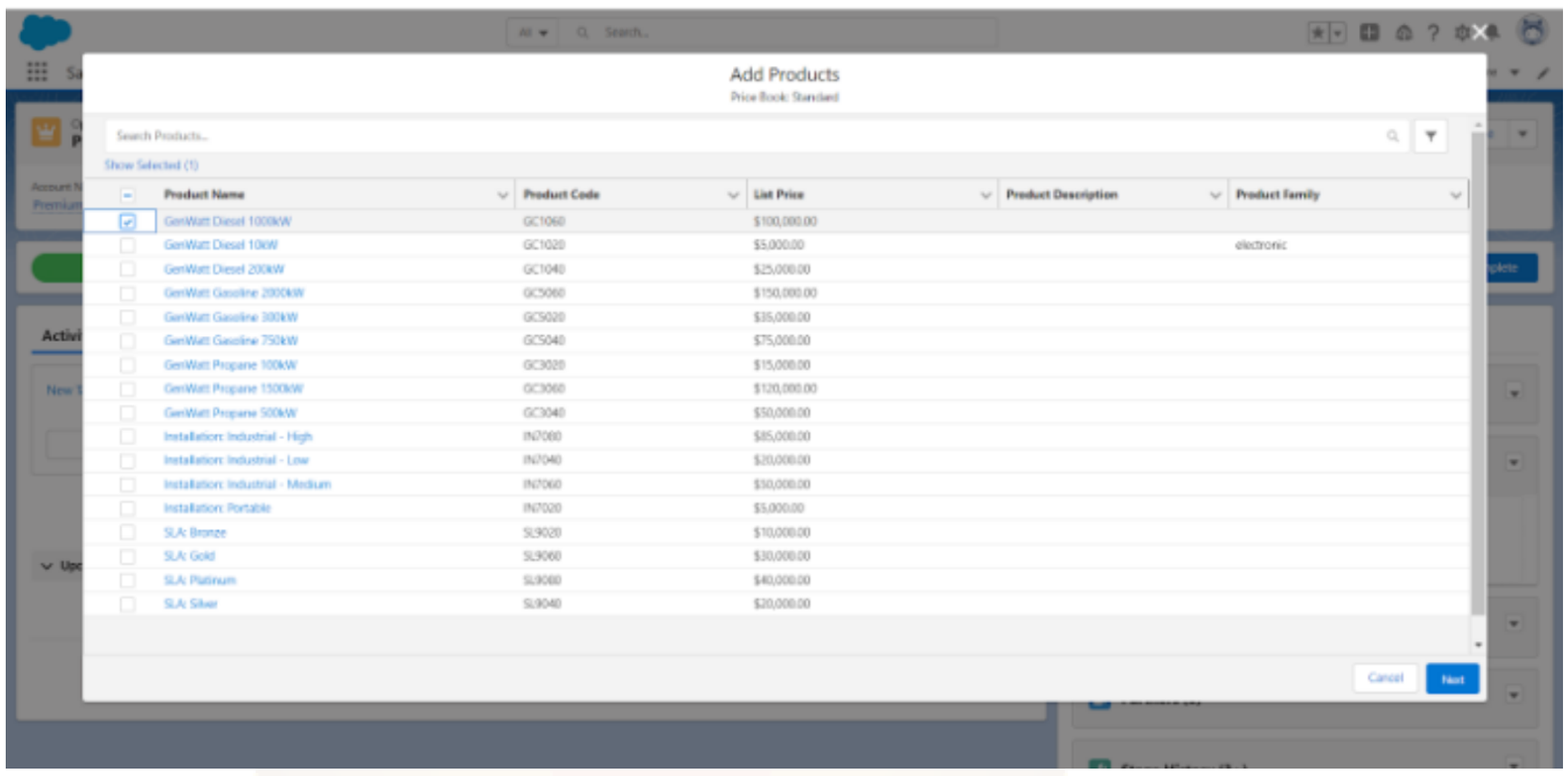
4. select any opportunity (or create one).



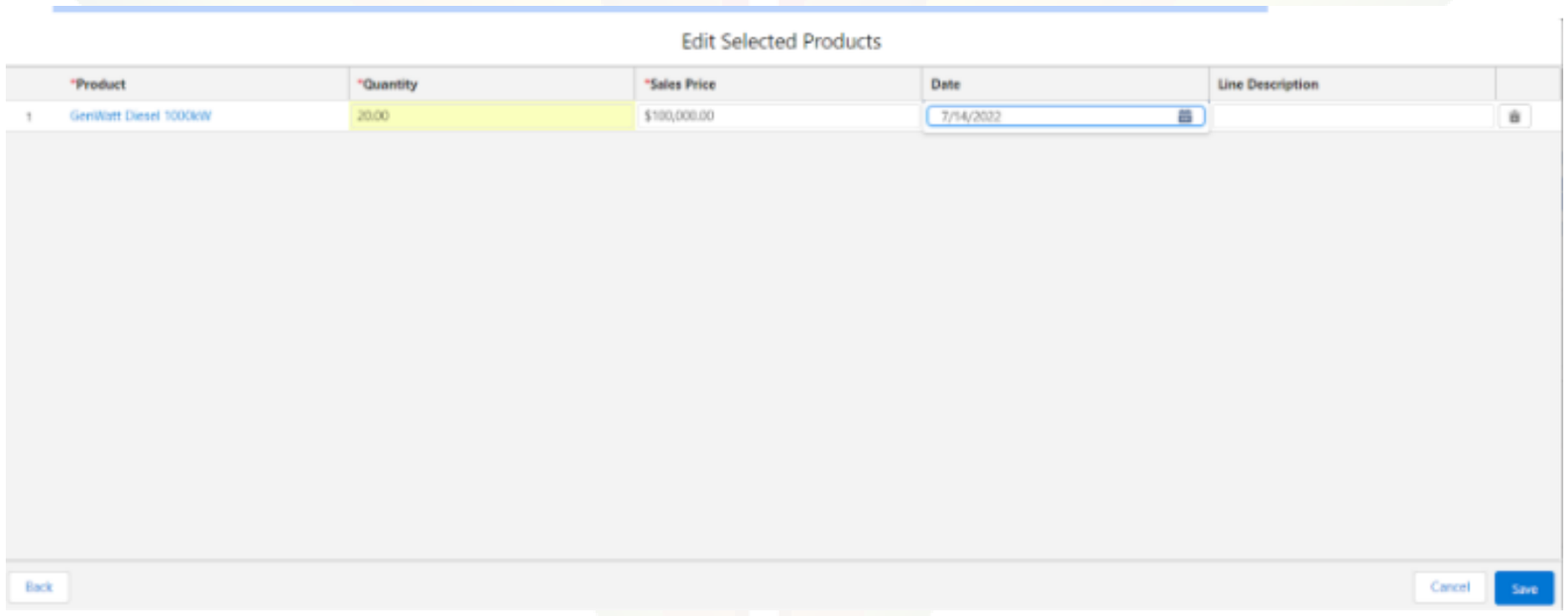
5. Add a product to that opportunity(you can see products in the opportunity related lists)



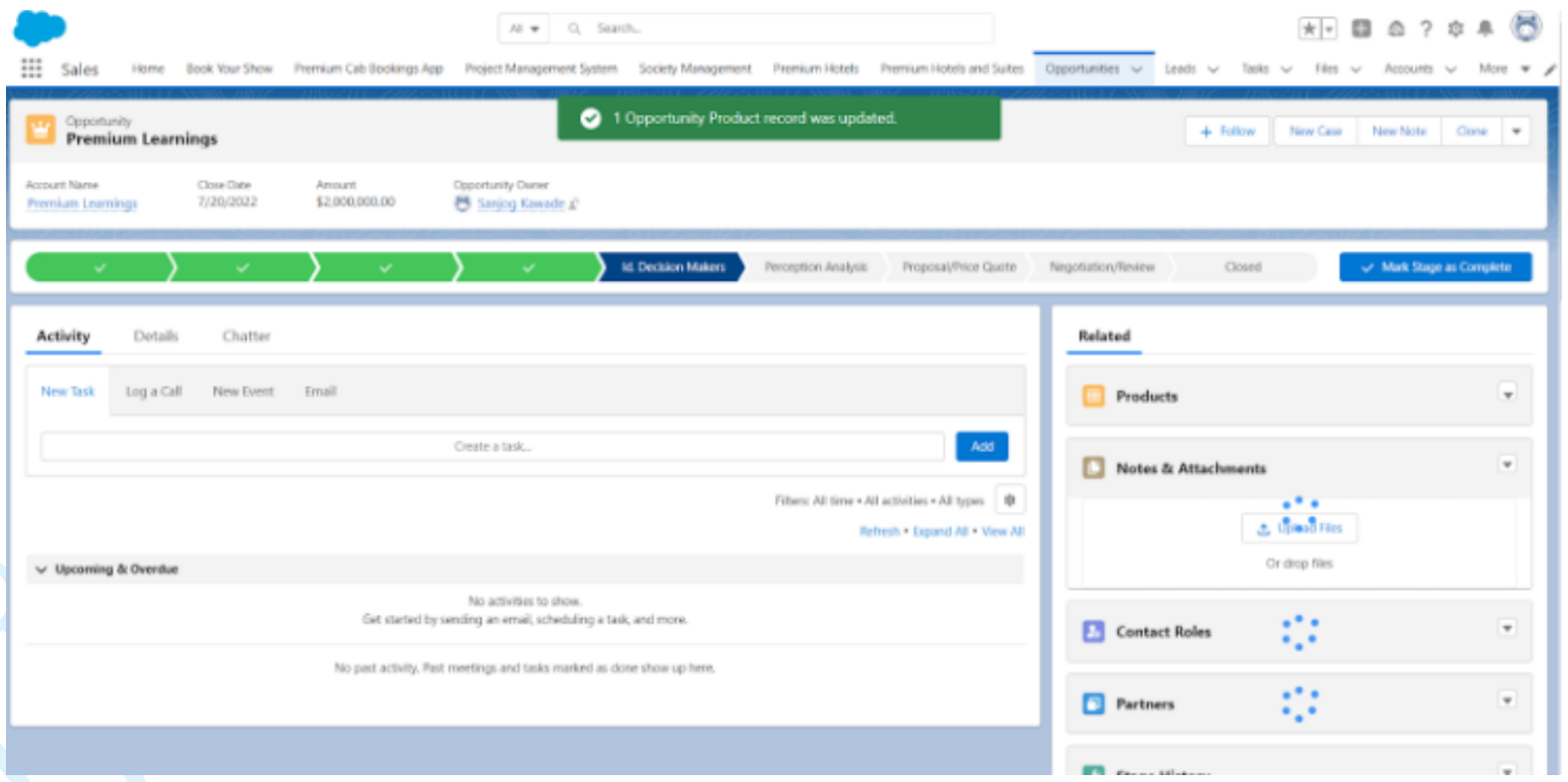
6. Select GenWatt Diesel 1000Kw and click next

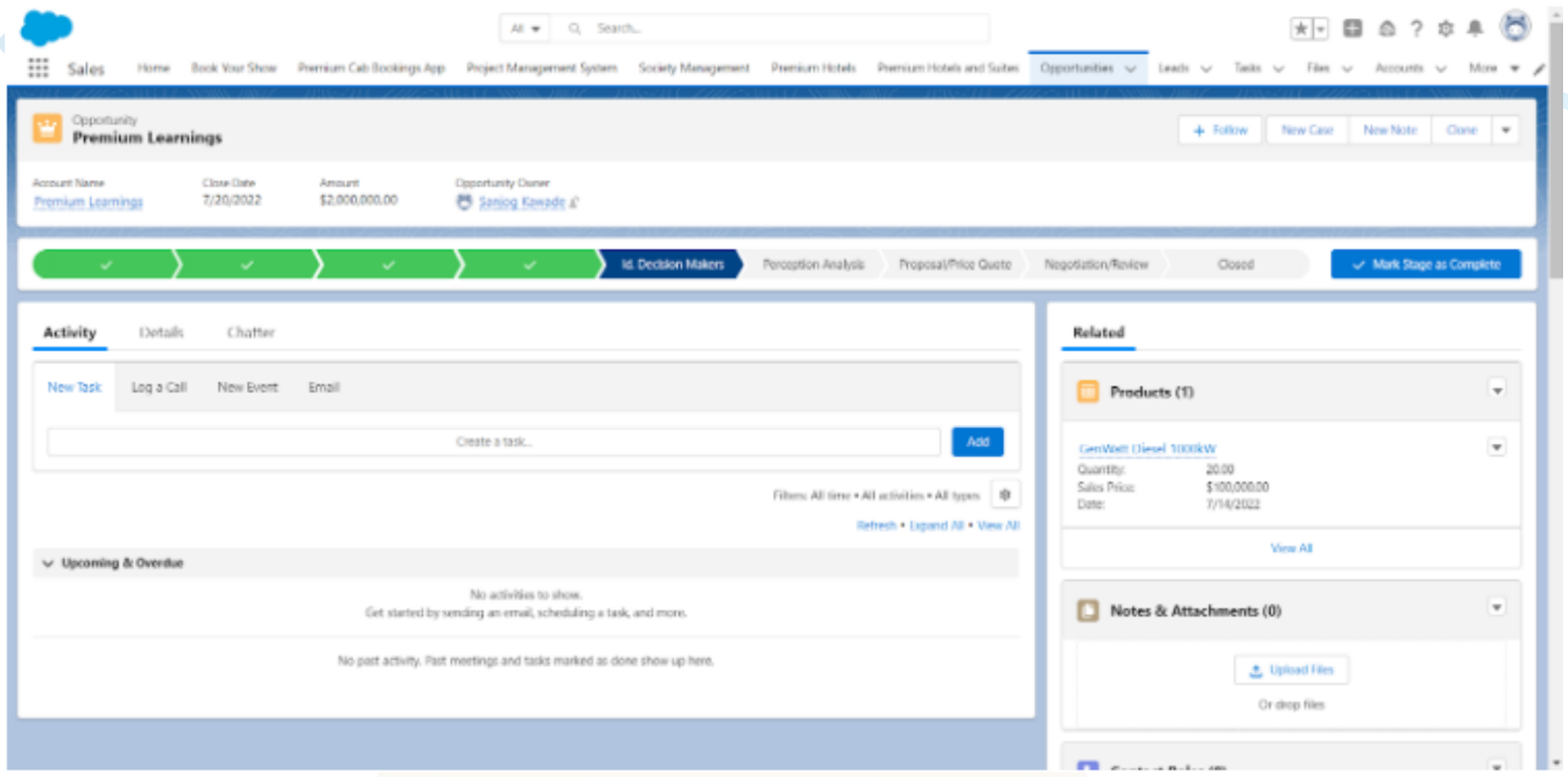


7. set quantity as 20 and set any date in field. Click save

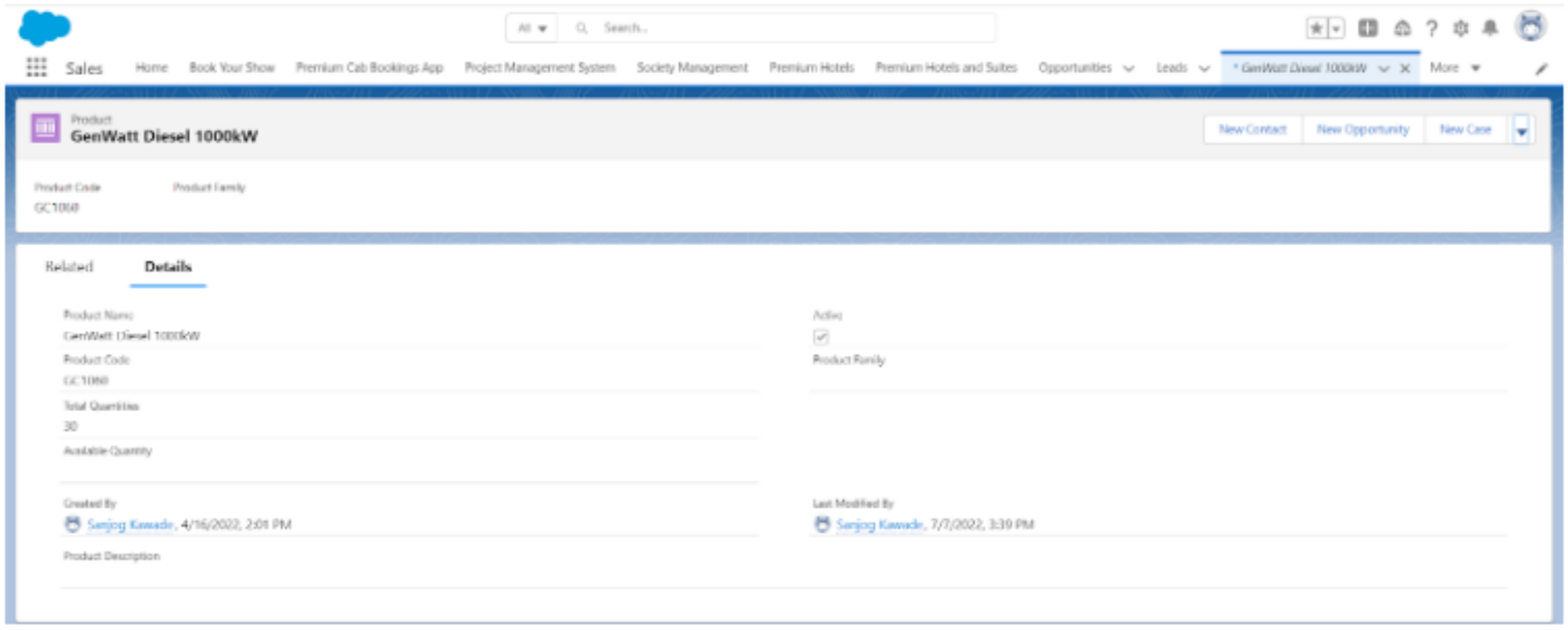


8. Opportunity product is created





9. if we check the product details of genWatt diesel 1000kW product, we can see the total quantities field updated now to 30.(you may need to refresh the page)



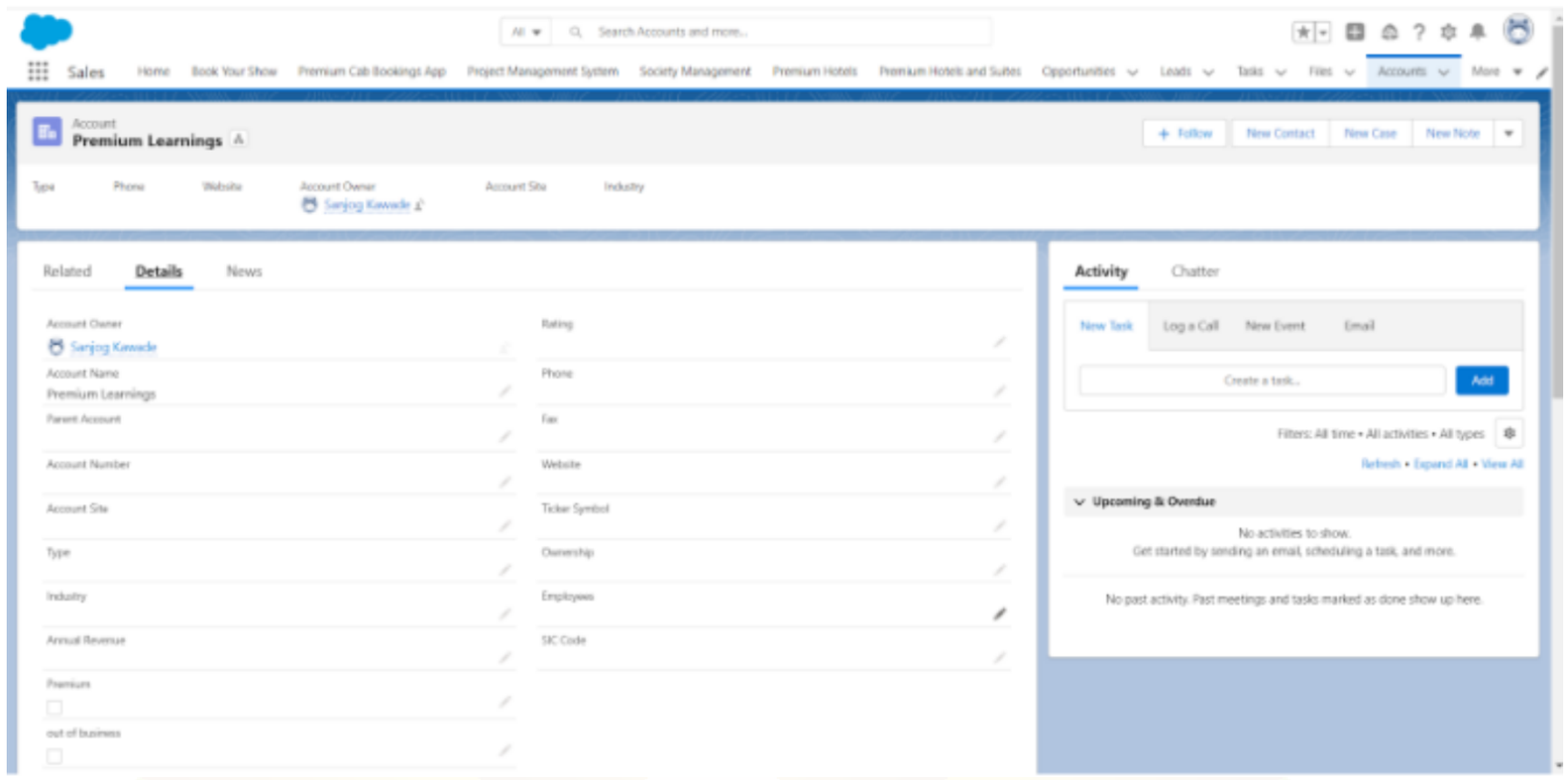
Example 15

The following Trigger will fires when we try to create the account with the the same name i.e. Preventing the users to create Duplicate Accounts

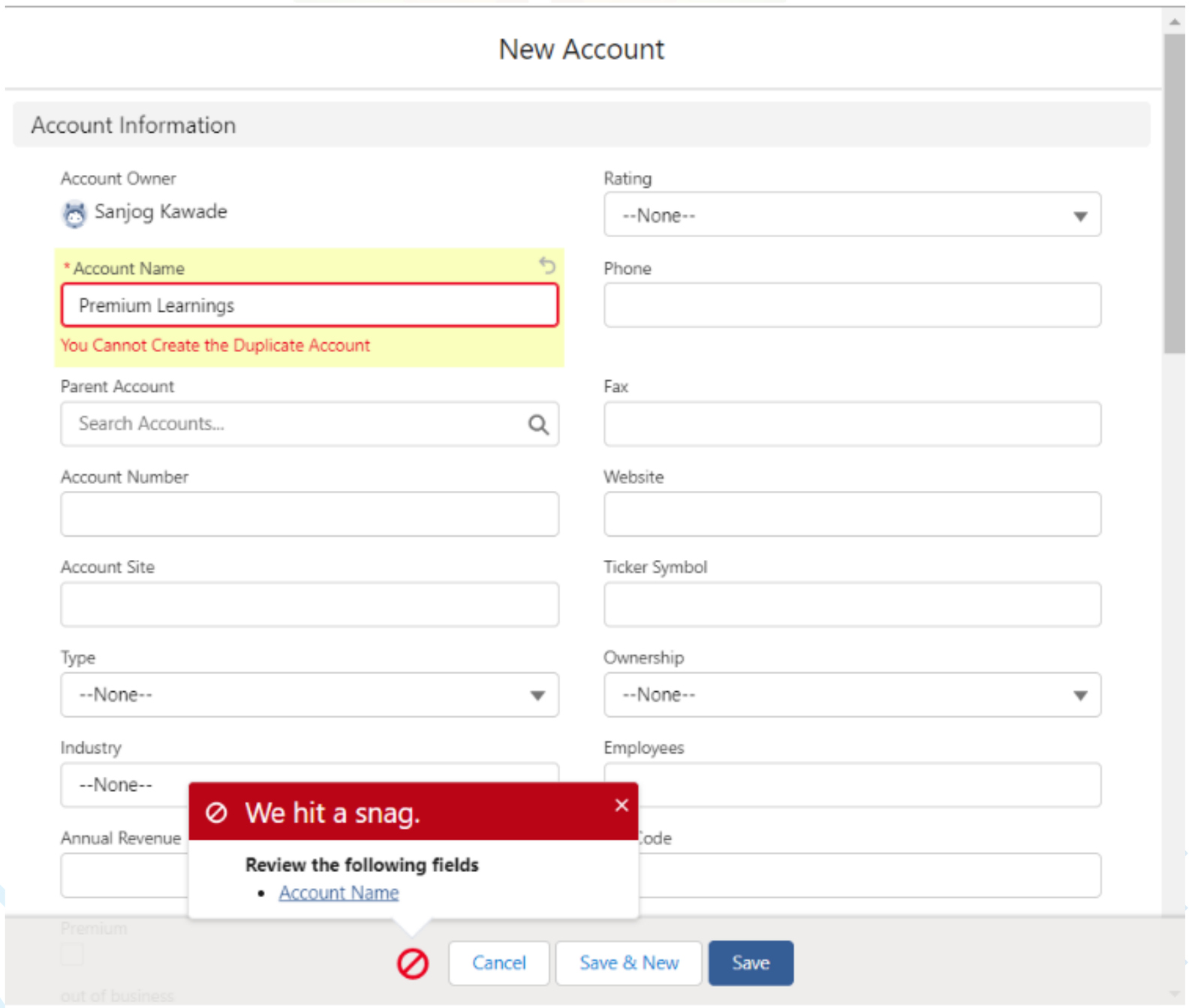
```
trigger AccountDuplicateTrigger on Account (before insert,
before update){

    for(Account a:Trigger.new)
    {
        List<Account> acc=[Select id, Name from Account where
        Name=:a.Name and Rating=:a.Rating];
          if(acc.size()>0)
          {
              a.Name.addError('You Cannot Create the Duplicate
                                Account');
          }
    }
}
```

Test Case:

1. Create an account with name Premium Learnings and save it



2. Create another account with same name and try to save it. You will get the following error on screen.



Example 16

Create the object called “Books” and create the field “Price”(data type is Currency) under this object. Whenever we enter some amount of money in the Price field and once we click on save button, the value we entered in the Price field is 10% less than the actual price. This is applicable for while both inserting and updating records.

DiscountTrigger (apex trigger)

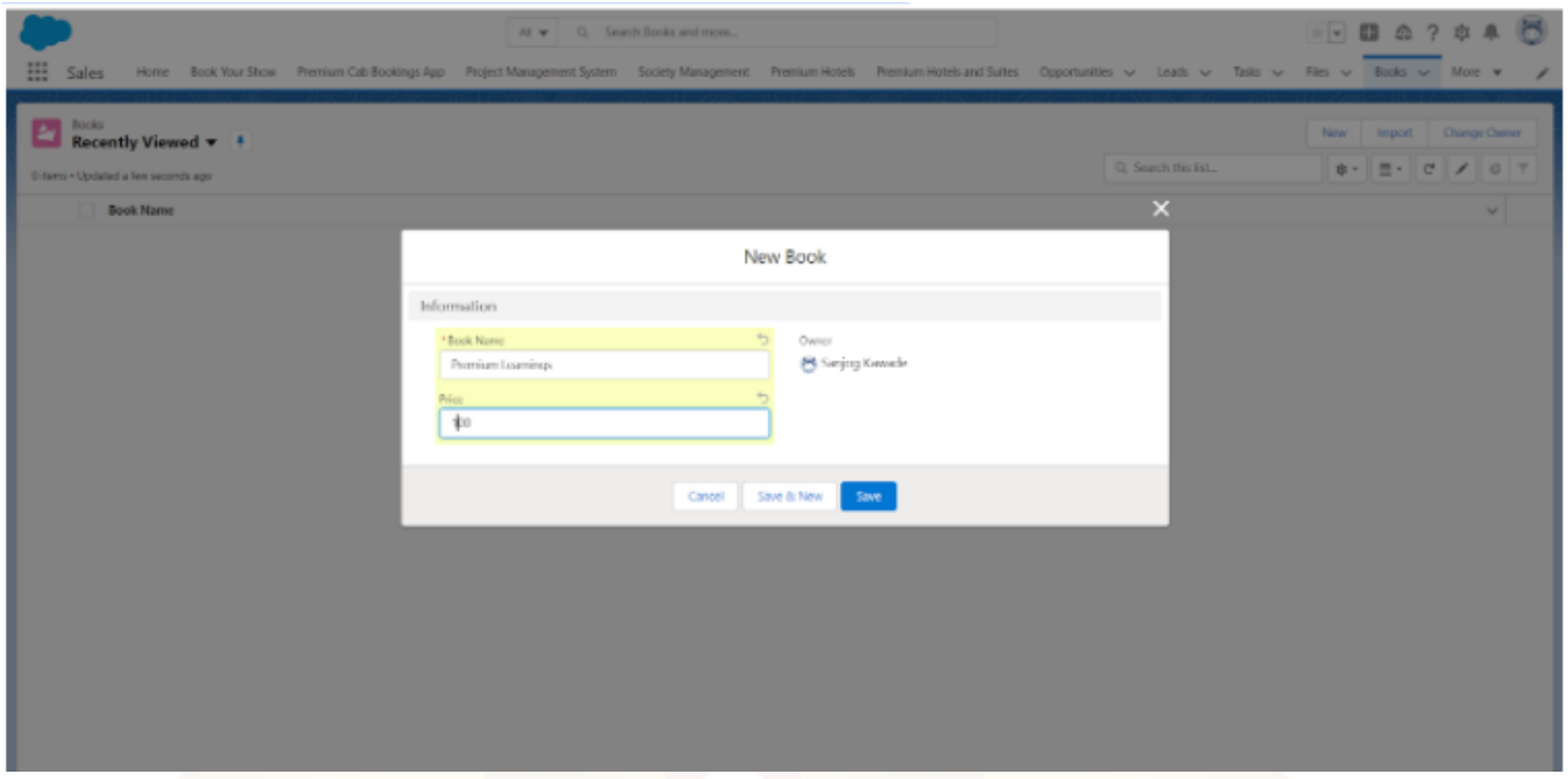
```
trigger DiscountTrigger on Book__c (before insert,
before update)
{
 List<Book__c> books = Trigger.new;
 PriceDiscount.applyDiscount(books);
}
```

PriceDiscount (apex class)

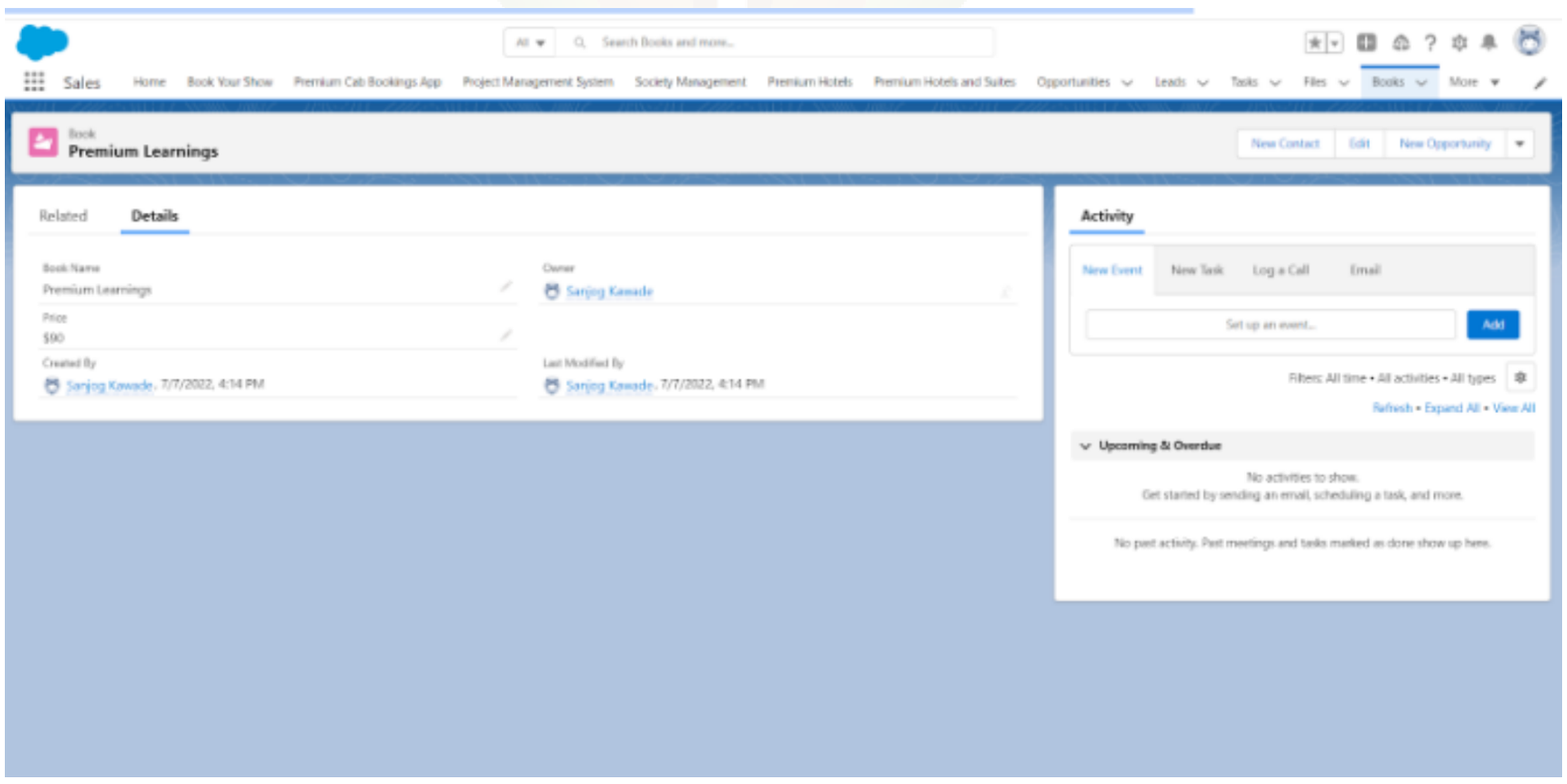
```
public class PriceDiscount {
 public static void applyDiscount(List<Book__c> books)
 {
  for (Book__c b :books){
       b.Price__c *= 0.9;
  }
 }
}
```

Test Case:

1. Create a new ‘book’ record as shown below and click save



2. After the record is saved, the price we entered above(100) gets changed to 90



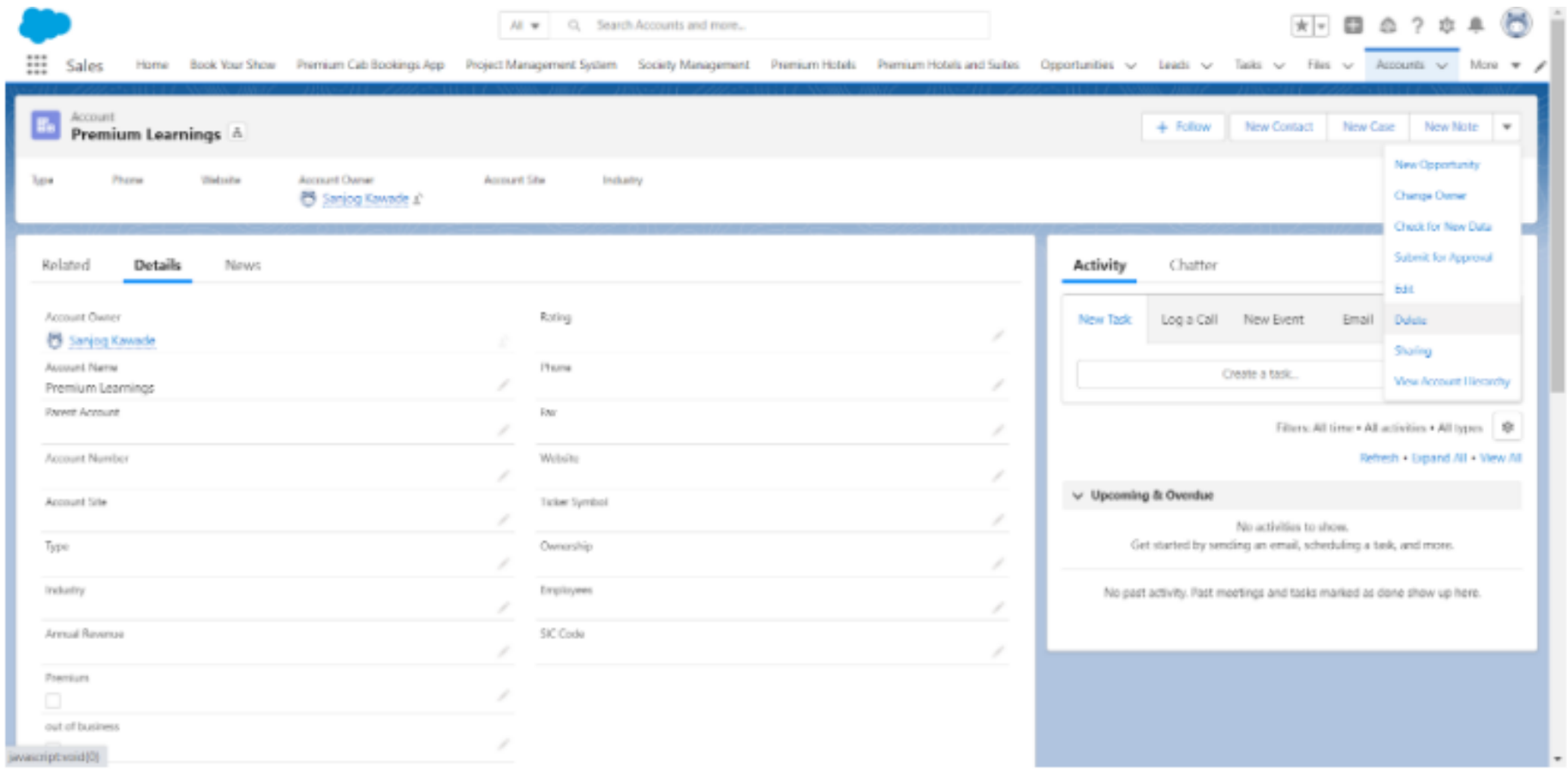
Example 17

The following trigger will prevent the users from deleting the Accounts. This is because System Administrator has all the permissions, we cannot change the permissions.

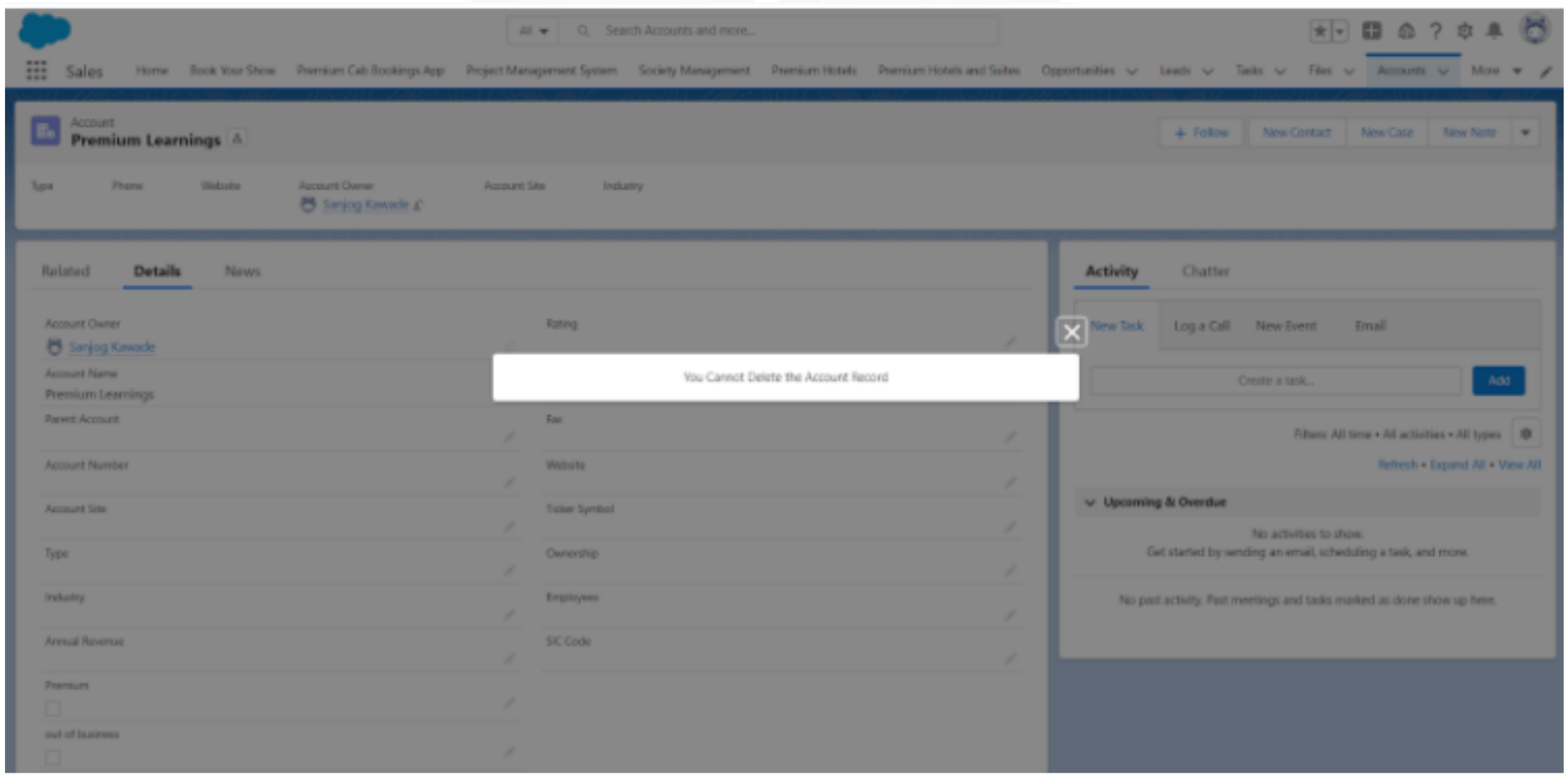
```
trigger AccountDelete on Account (before delete)
{
   for(Account Acc:trigger.old)
   {
     acc.adderror('You Cannot Delete the Account Record');
   }
}
```

Test Case:

1. Open any account record(or create one) and try to delete it



2. The following error message will be displayed



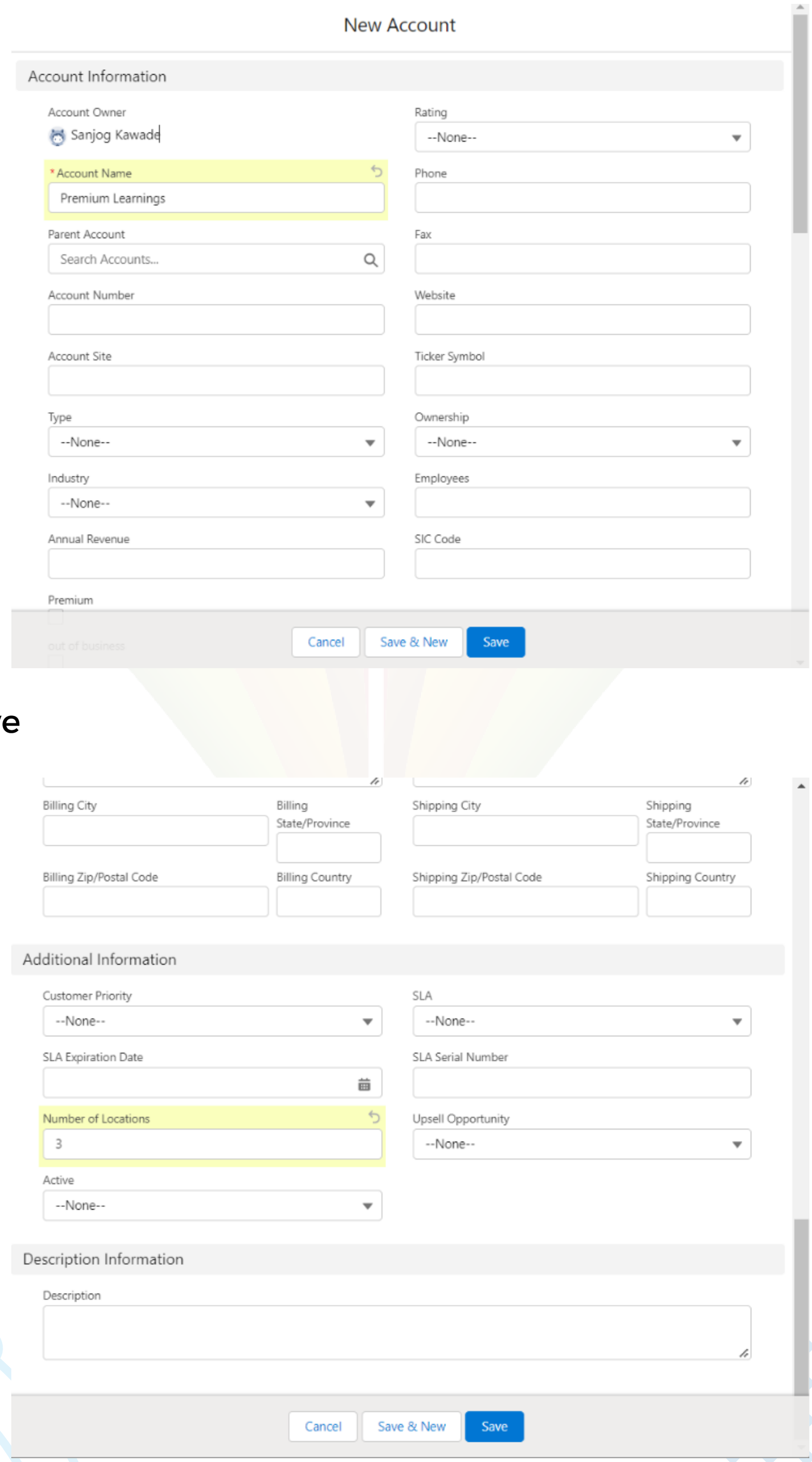
Example 18

Create a Custom field called “Number of Locations” on the Account Object (Data Type=Number)The following trigger creates the number of contacts that are equal to the number which we will enter in the Number of Locations field on the Account Object

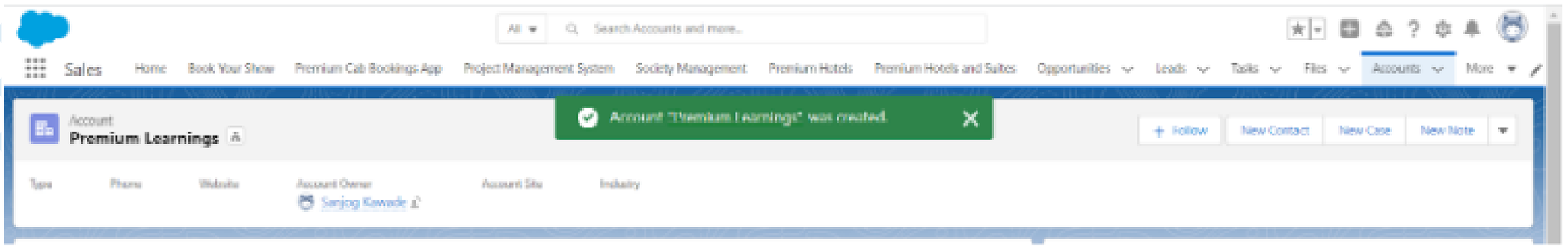
```
trigger ContactsCreation on Account (after insert) {

 list<contact> listContact = new list<contact>();
 map<id,decimal> mapAcc=new map<id,decimal>();

 for(Account acc:trigger.new){
    mapAcc.put(acc.id,acc.Number_of_Locations__c);
 }
 if(mapAcc.size()>0 && mapAcc!=null){
   for(Id accId:mapAcc.keyset()){
      for(integer i=0;i<mapAcc.get(accId);i++){
          contact newContact=new contact();
          newContact.accountid=accId;
          newContact.lastname='contact'+i;
          listContact.add(newContact);
      }
   }
  }
 if(listContact.size()>0 && listContact!=null)
    insert listContact;
}
```

Test Case:

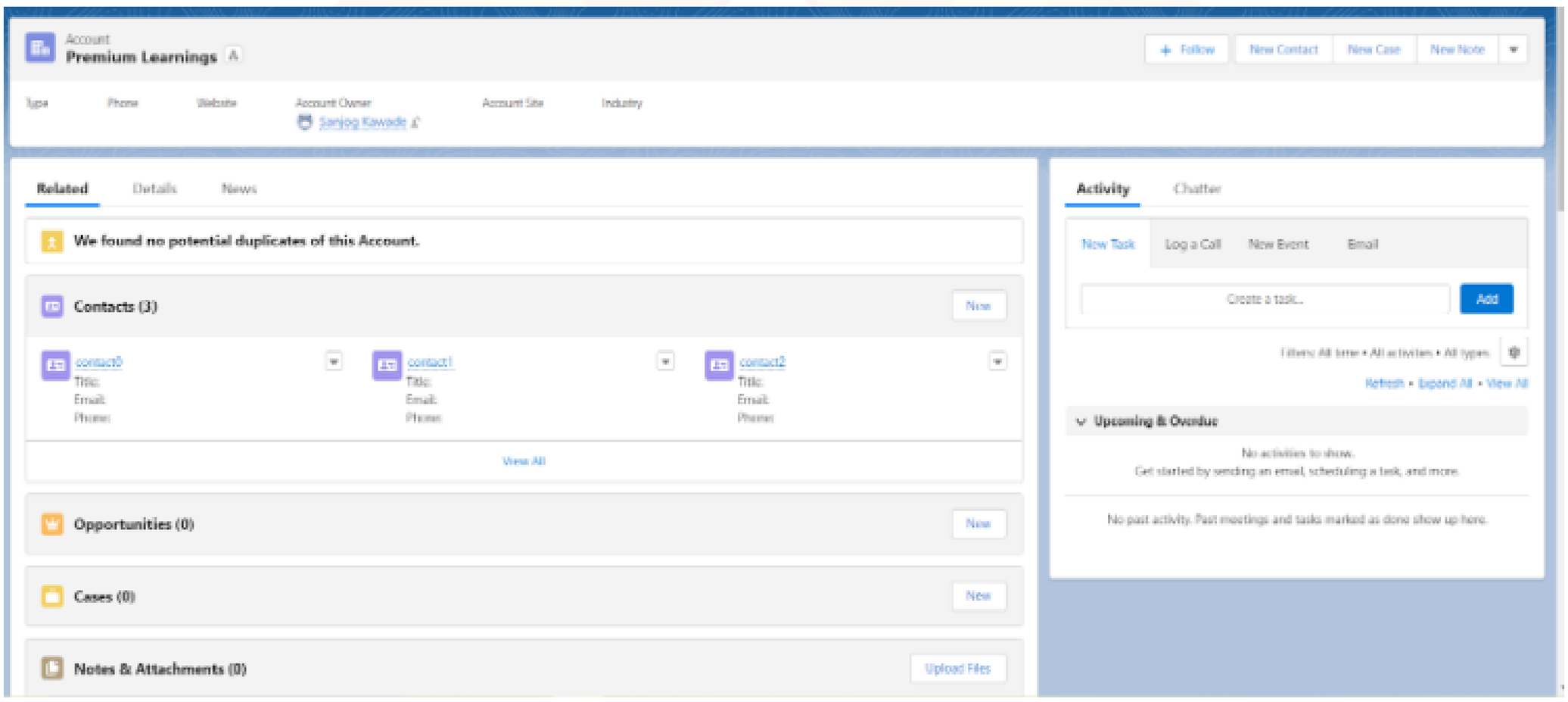
1. Create an account record. And input the number of locations field with a number(for example 3)



Click save



2. we can see three contacts created automatically in related lists as shown below



Example 19

The logic is when we create Customer Project for an Opportunity with the Status=Active, then Active Customer project check box on the Opportunity Detail page is automatically checked.

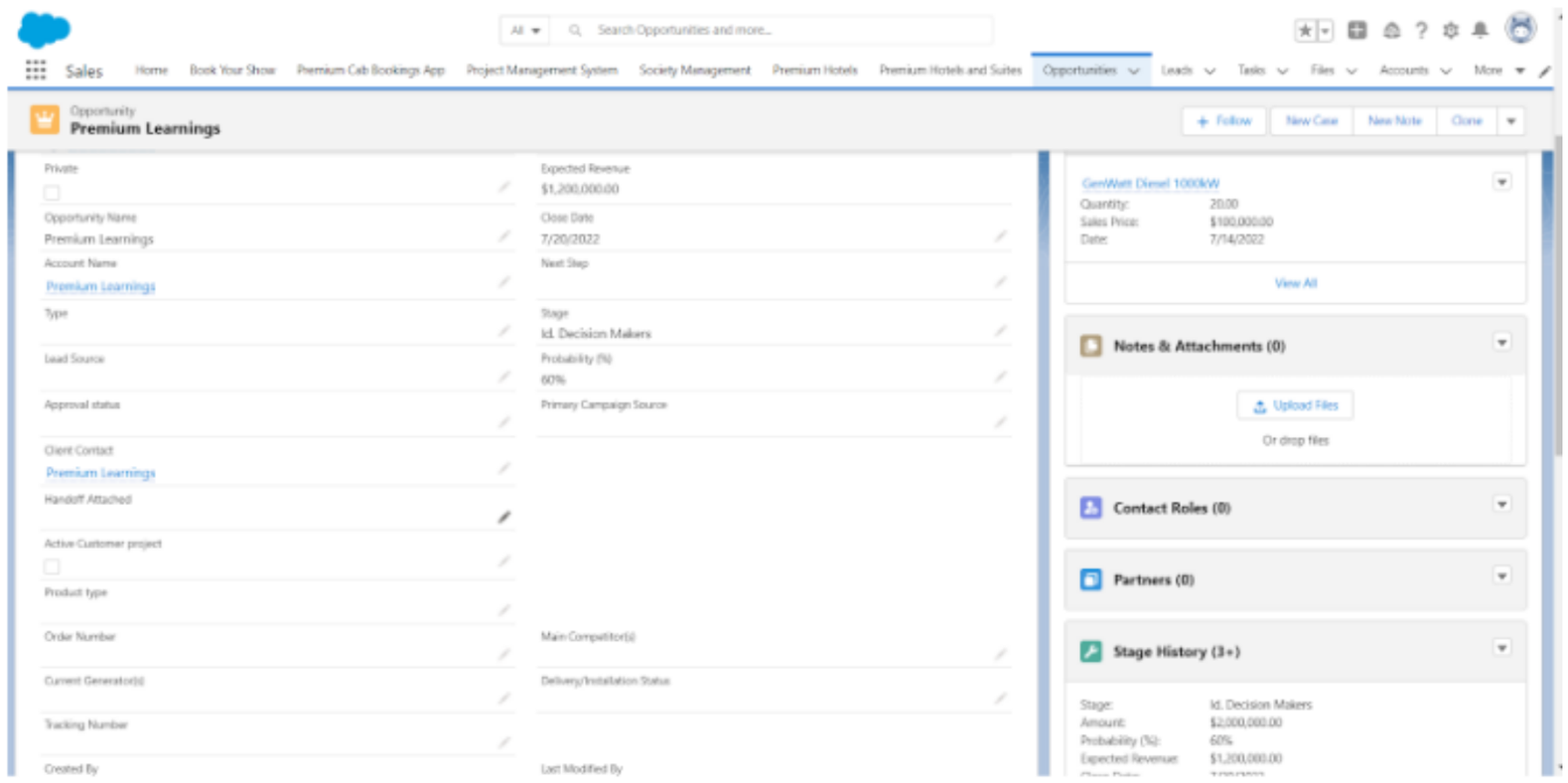
Create the object called “Customer Project” and create a Status field under this object with the picklist data type (Values=Active, Inactive). Create the relationship between this custom object and Opportunity so that the Customer Project is a related list to the Opportunity. Create Active Customer project

- field on Opportunity object (Data Type=Checkbox)

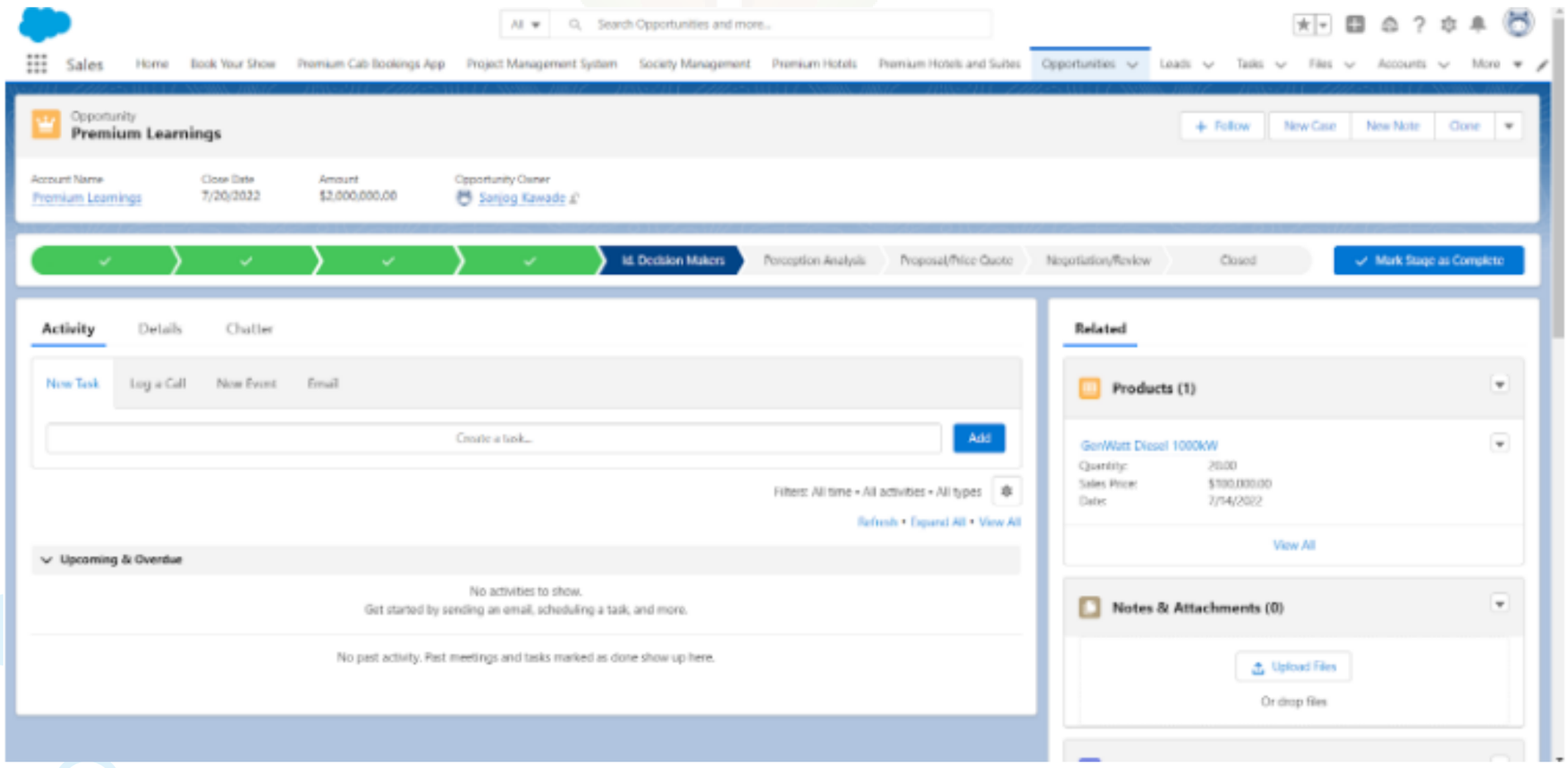
```
trigger UpdateCPActivenOnOppty on Customer_Project__c
(after insert){

 List<Opportunity> opps=new List<Opportunity>();
 for(Customer_Project__c cp:Trigger.New){
    if(cp.Status__c=='Active'){
      Opportunity opp=new Opportunity(id=cp.Opportunity__c);
      opp.Active_Customer_Project__c = True;
      opps.add(opp);
    }
  }

 update opps;
}
```

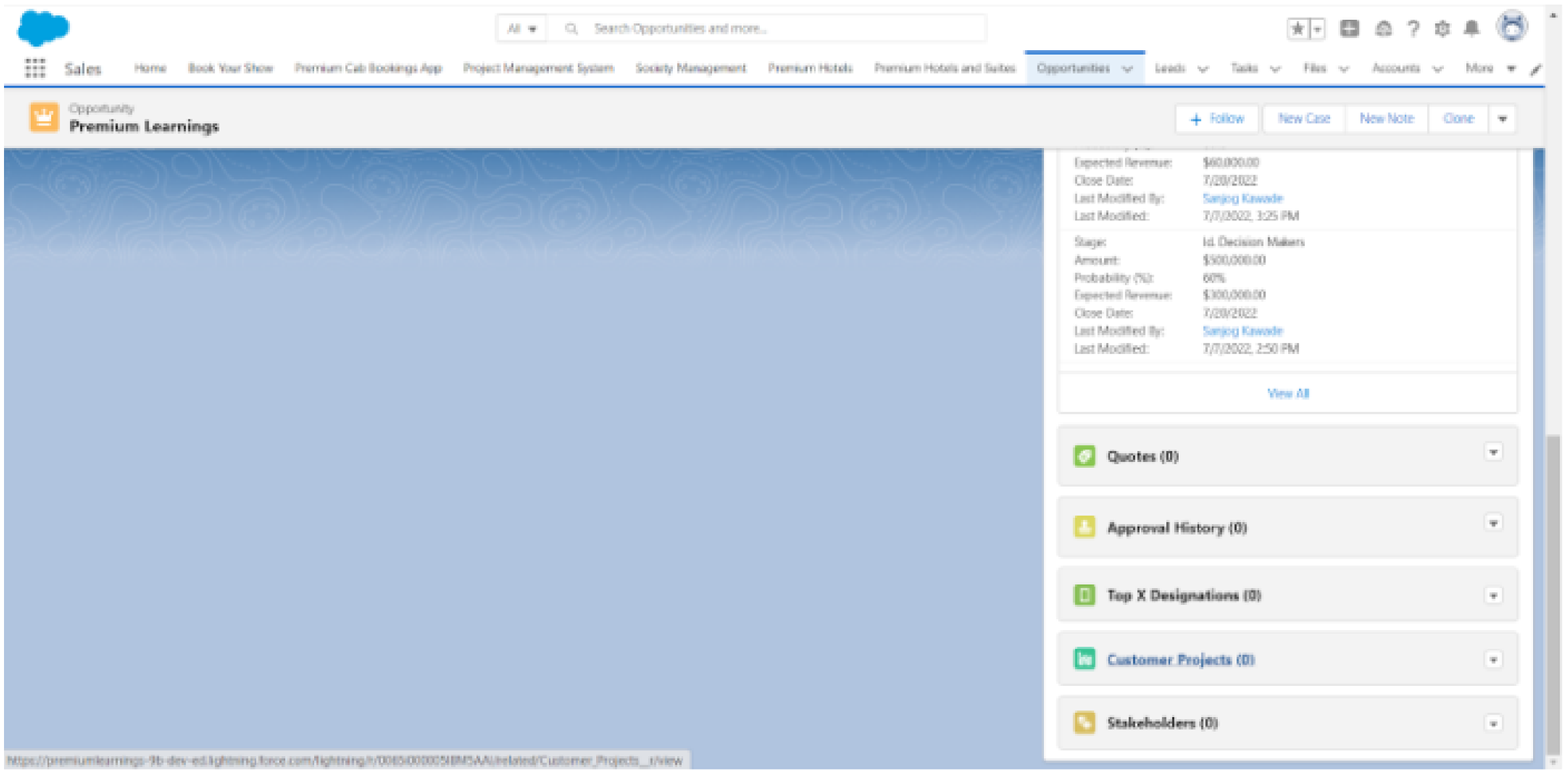
Test Case:

1. Open any opportunity record(or create one) and make sure the Active Customer Project Checkbox is Unchecked.

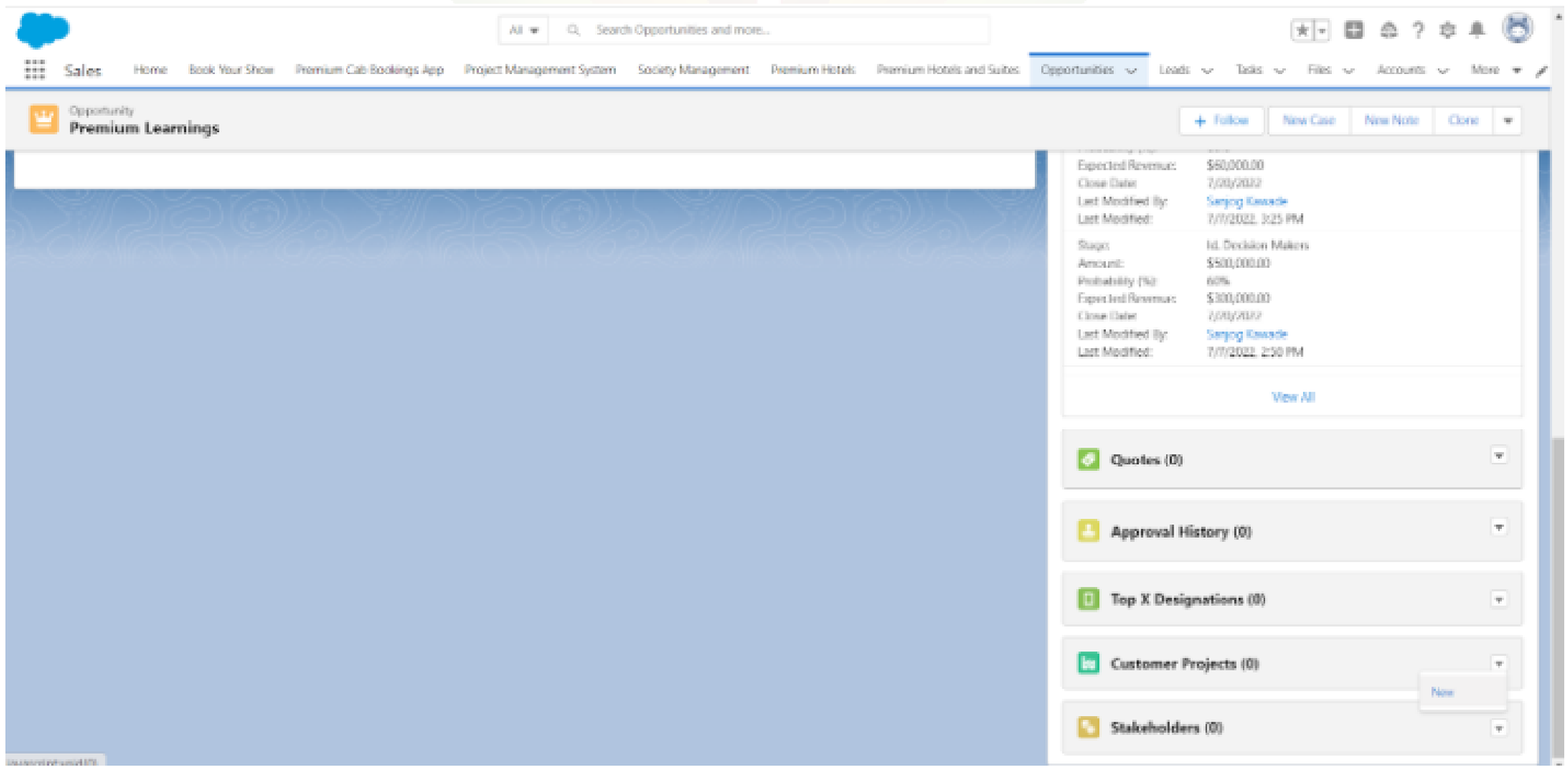


2. scroll down in the related lists to find ‘customer project’ related object

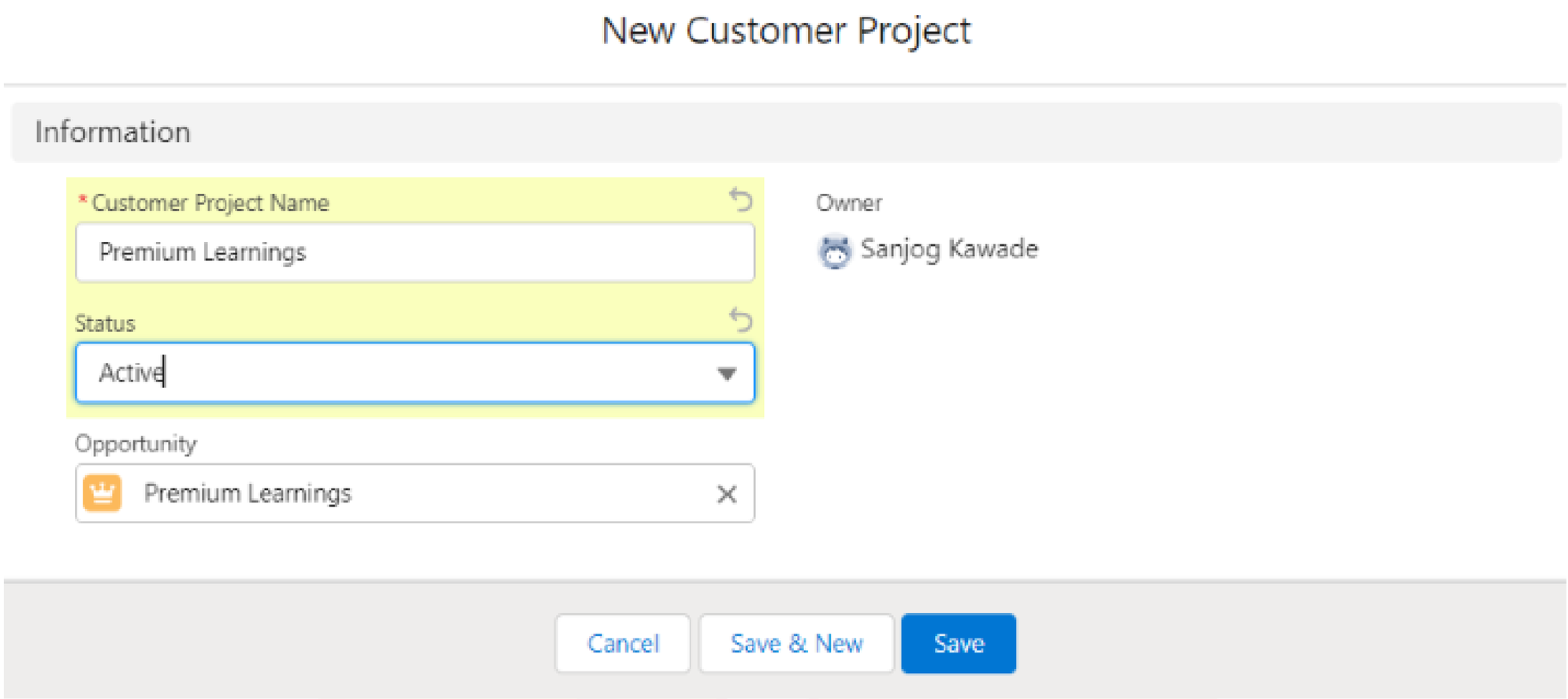




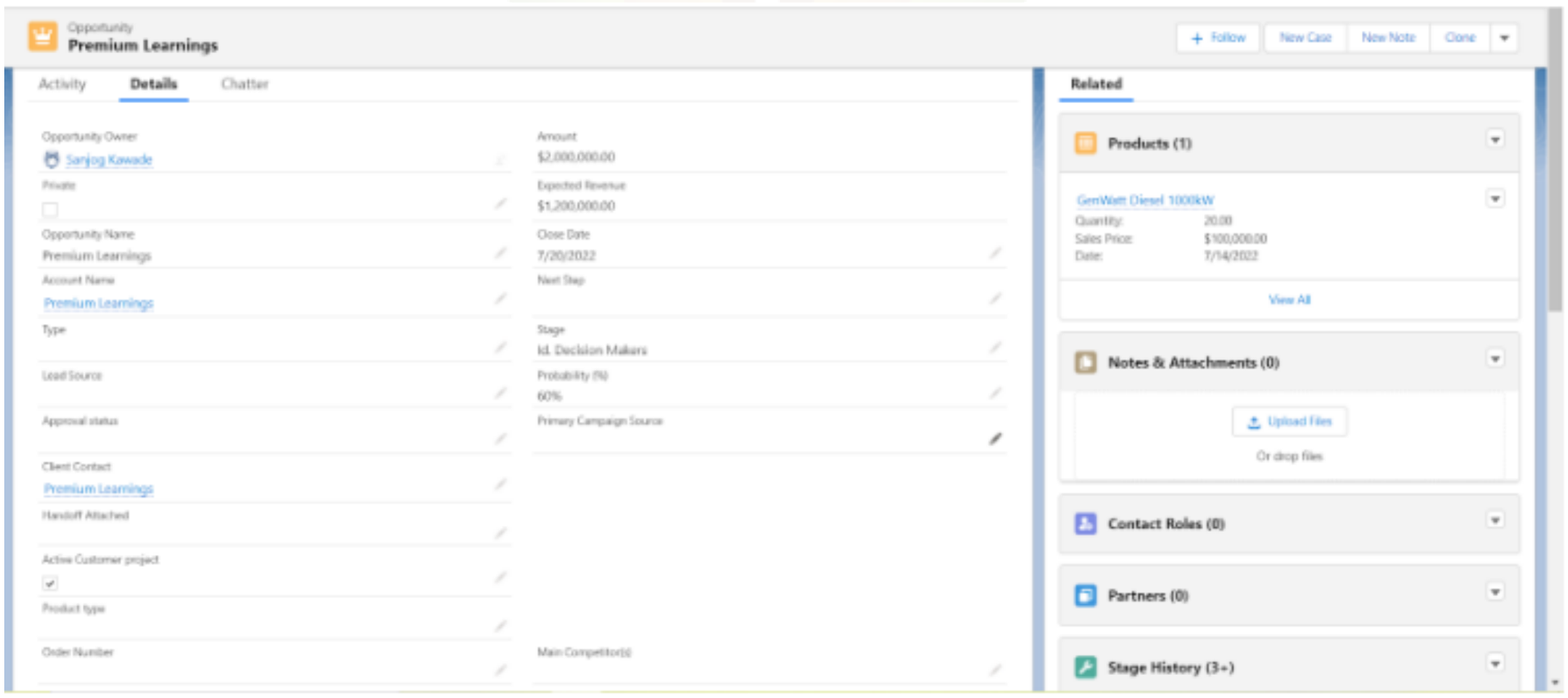
3. Create a new customer project from related lists as shown below



4. Create a new Customer Project record and save it. (make sure Status field is active)



5. Now check the opportunity record again to see the active customer project checklist is now checked.



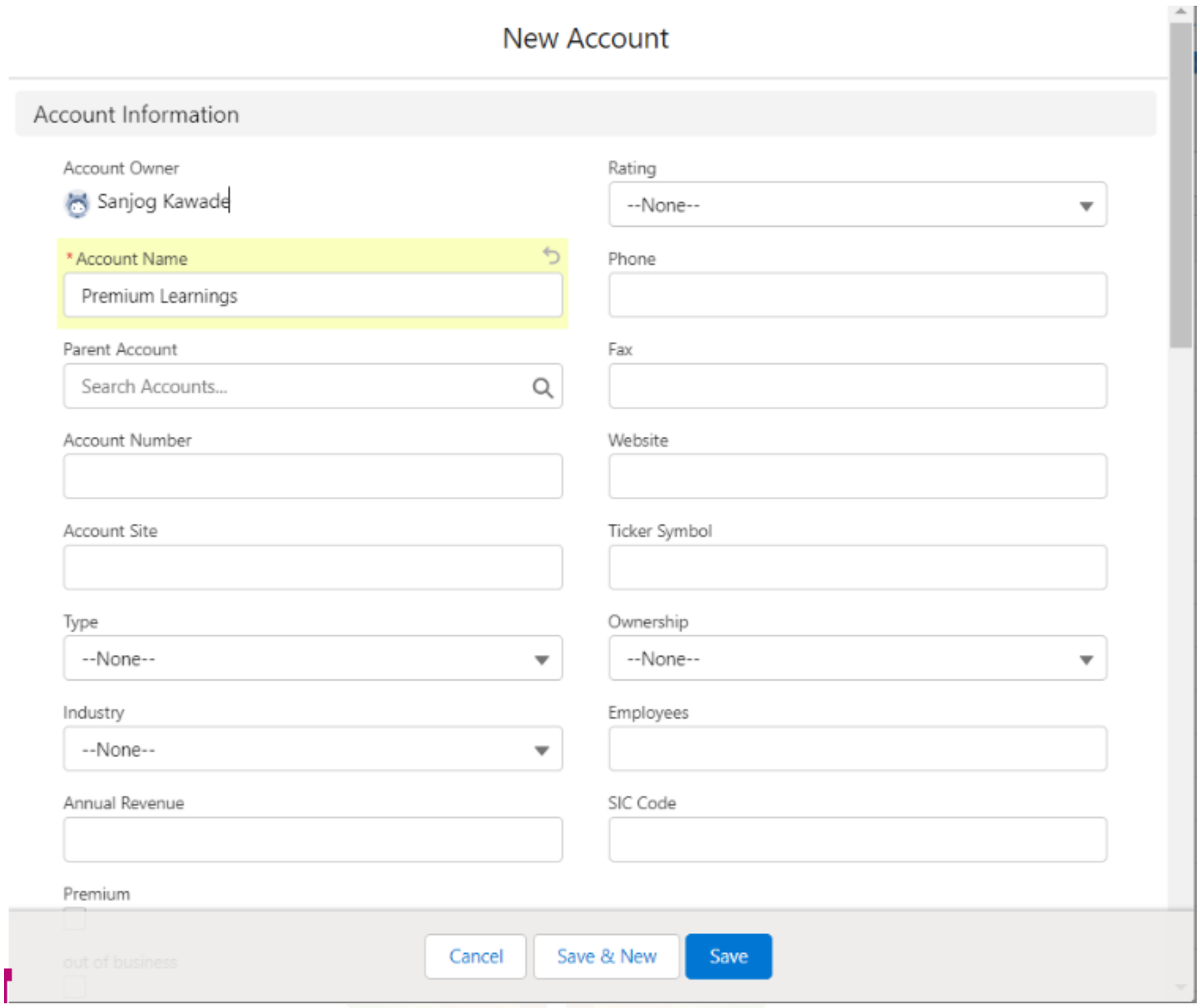
Example 20

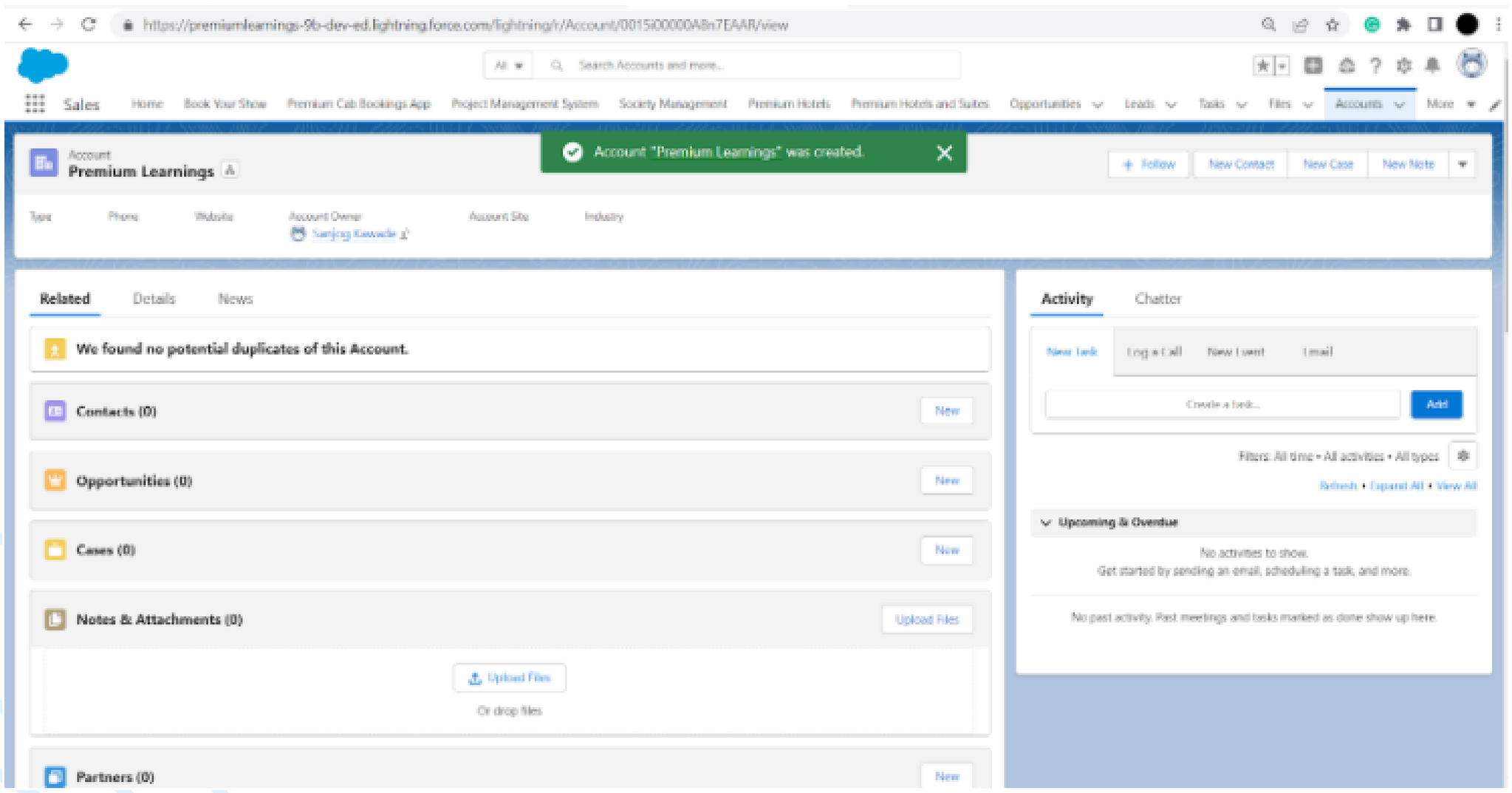
Create “Sales Rep” field with the data type (Text) on the Account Object. When we create the Account record, the Account Owner will be automatically added to Sales Rep field. When we update the Account owner of the record, then also

```
trigger UpdateSalesRep on Account (Before insert,
Before Update)
{

 Set<Id>setAccOwner=new Set<Id>();
 for(Account Acc: trigger.new)
 {
   setAccOwner.add(Acc.OwnerId);
 }

 Map<Id,User>User_map = new Map<Id,User>([select Name
                        from User where id in:setAccOwner]);

      for(Account Acc: Trigger.new)
      {
         User usr=User_map.get(Acc.OwnerId);
         Acc.Sales_Rep__c=usr.Name;
      }
}
```

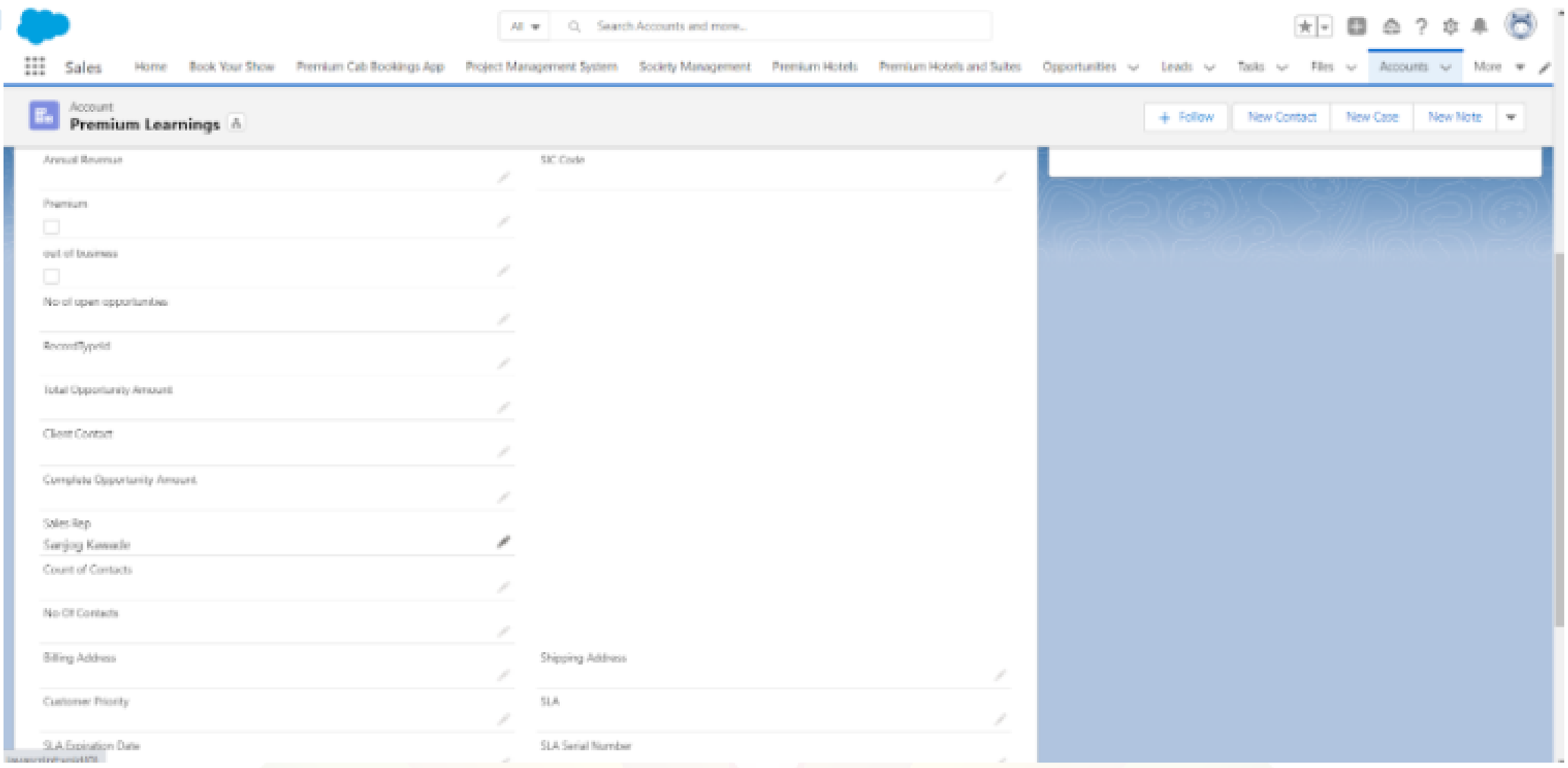
Test Case:

1. Create a new account with only name field populated as shown below





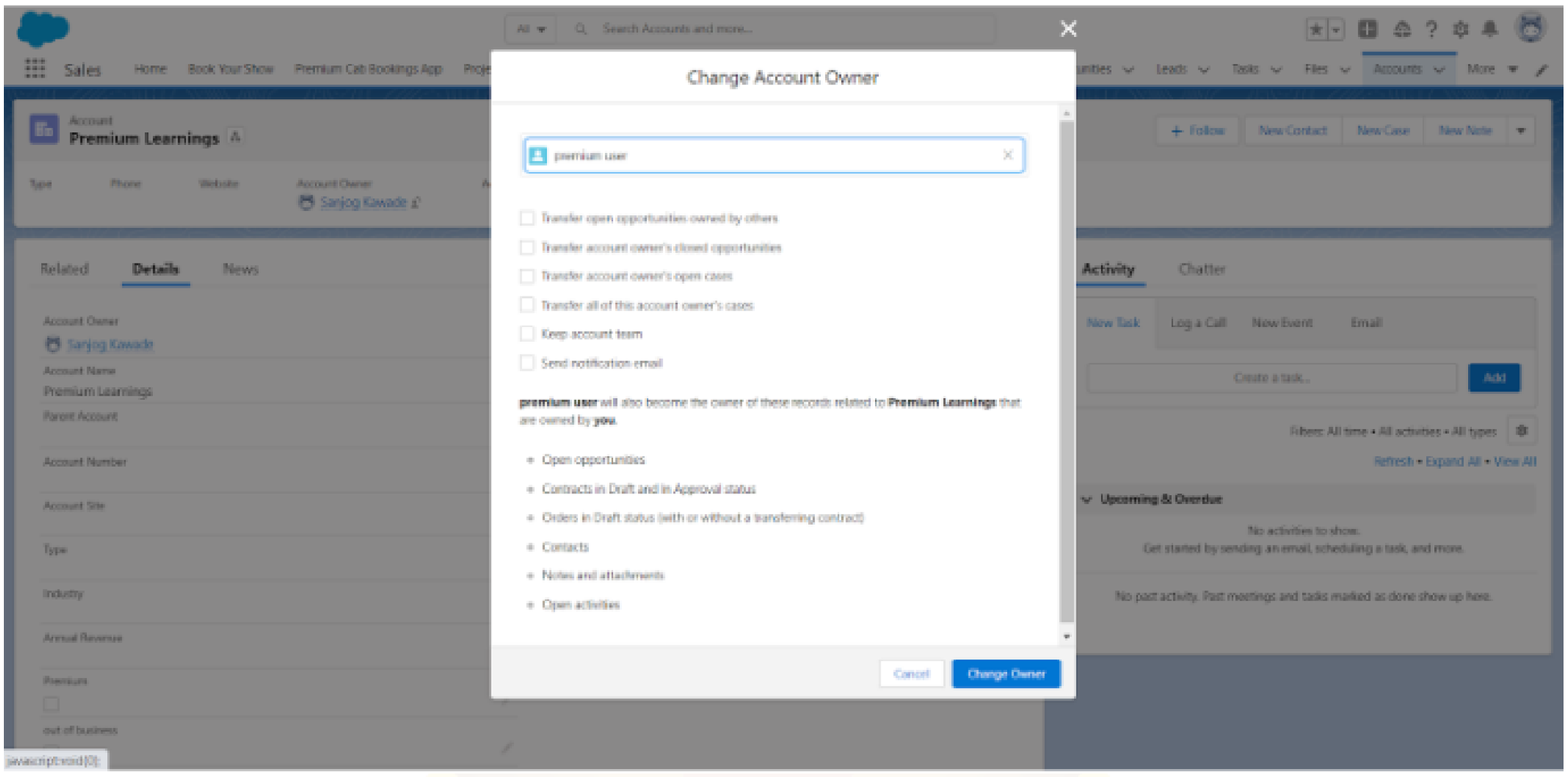
2. Now check the sales rep custom field and you will see your name populated in it.



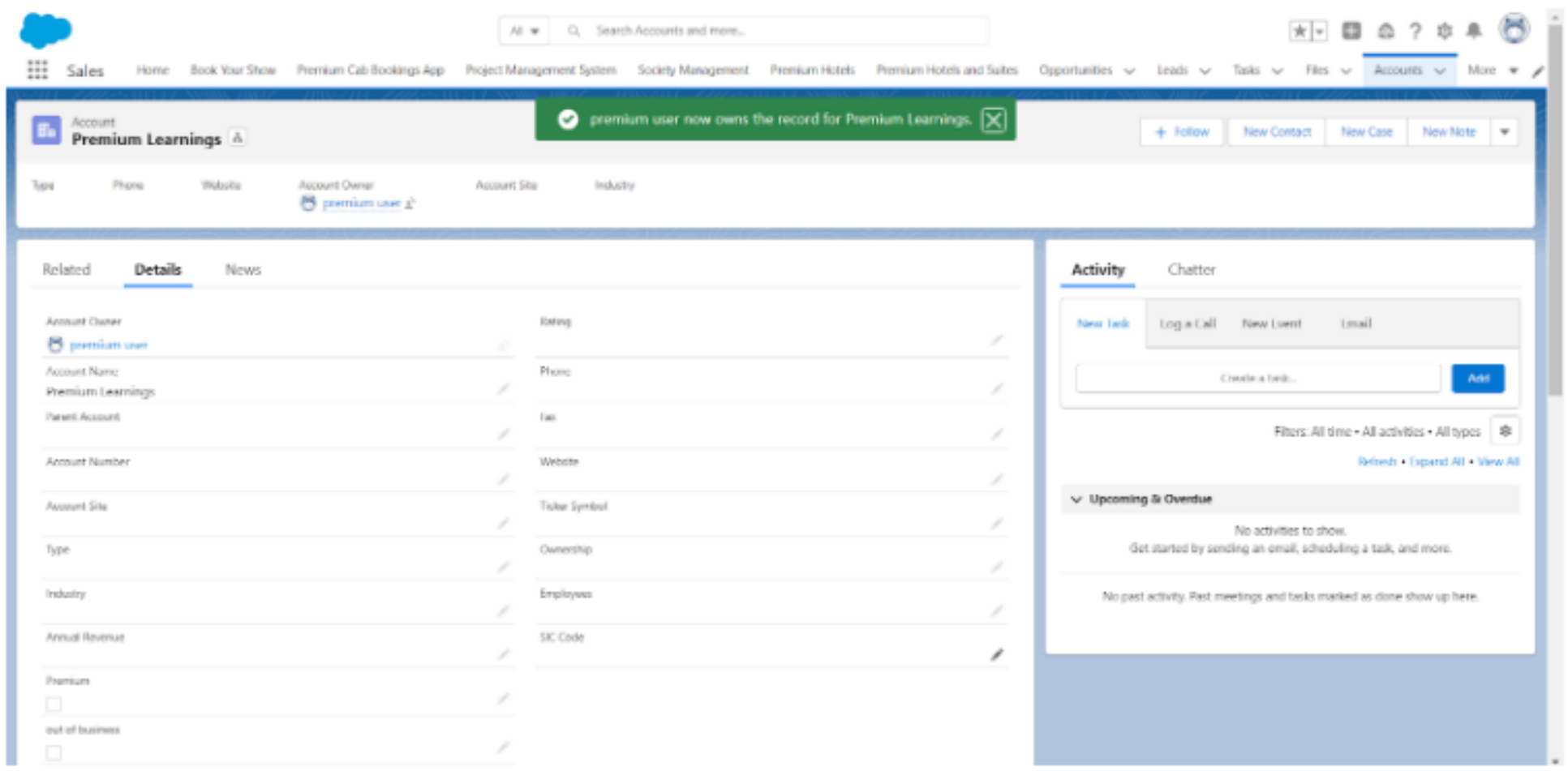
3. Now change the account owner of the same account record, and make any other user owner of the record (click on the account owner field and a pop up will appear to change owner)

select any other user apart from you and click change owner.

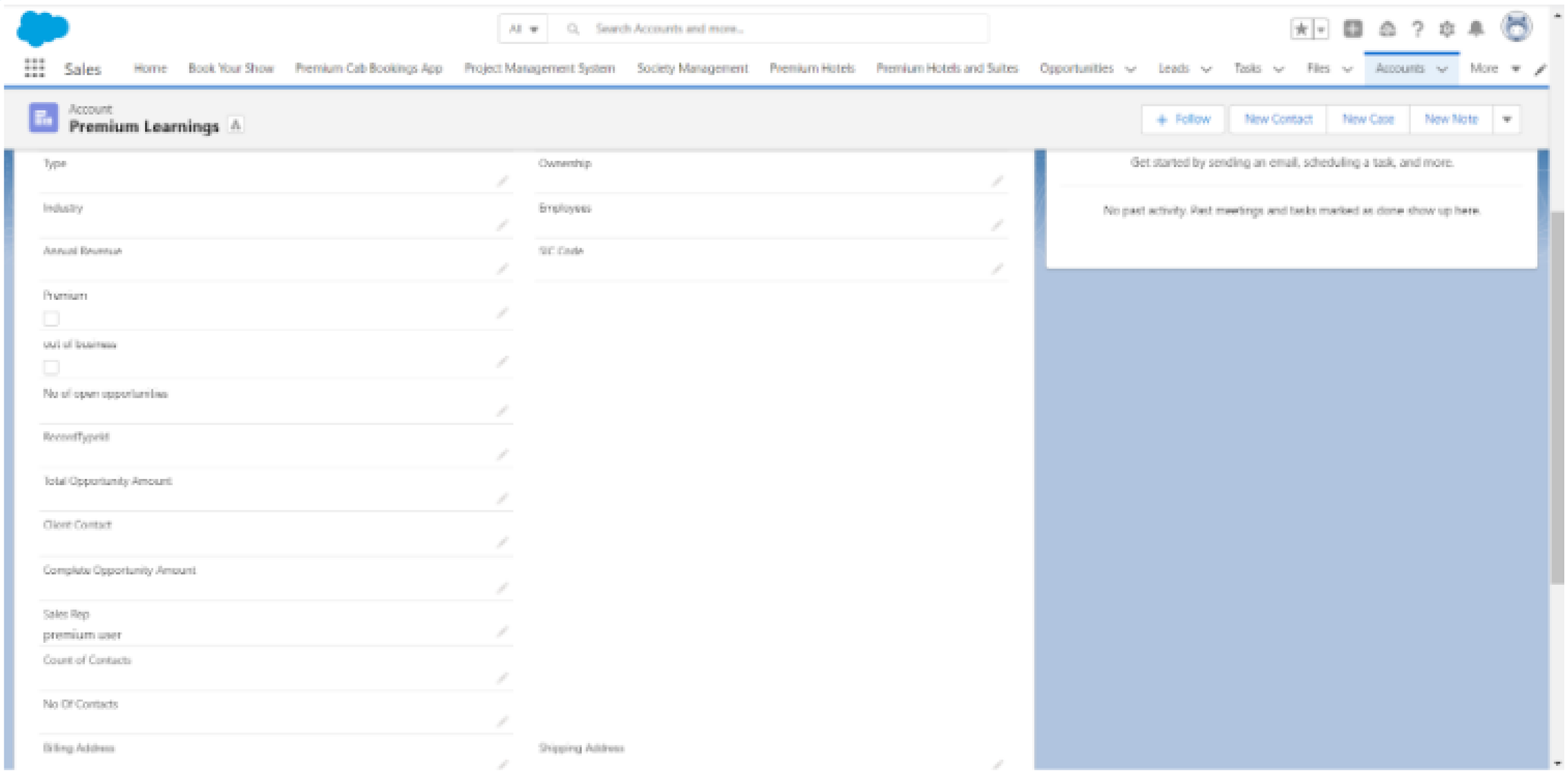
(If you cant see any other user in drop down list, create new user set up > home > quick find > users > new user > input the required fields and create a new user)



4. Now the new user is owner of record



4. Now check the Sales rep field and it will be updated with the newly assigned account owner as shown below



Example 21

Create the field called the “Contact Relationship” checkbox on the Contact Object and Create the related object called “Contact Relationship” which is a related list to the Contacts.(Look Up Relationship). Now logic is when we create contact by checking the Contact Relationship checkbox, then Contact Relationship will be created automatically for that contact.

CreateCRonContactCreation(apex trigger)

```
trigger CreateCRonContactCreation on Contact
(after insert){

 if(trigger.isAfter) {
  if(trigger.isInsert) {
   ContactMasterHandler ConIns = New
                          ContactMasterHandler();
ConIns.createContactRelationshipByContact(trigger.New);
  }
 }
}
```

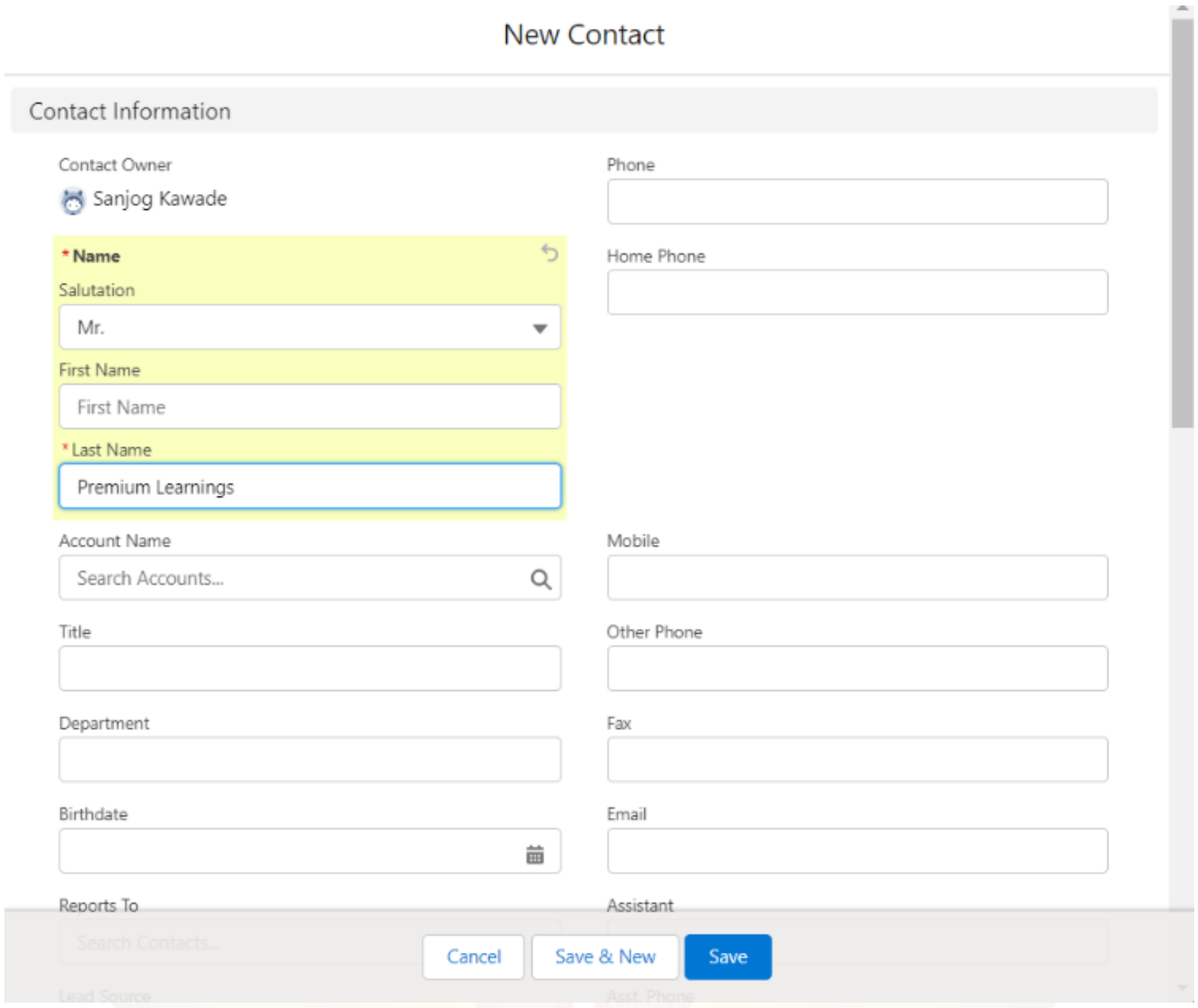
ContactMasterHandler(apex class)

```
Public Class ContactMasterHandler_Undelete{
 public void undeleteContactRelationshipsByContact
                          (list<Contact>List_Contacts)
{
set<Id> ContactIds = New set<Id>();
```

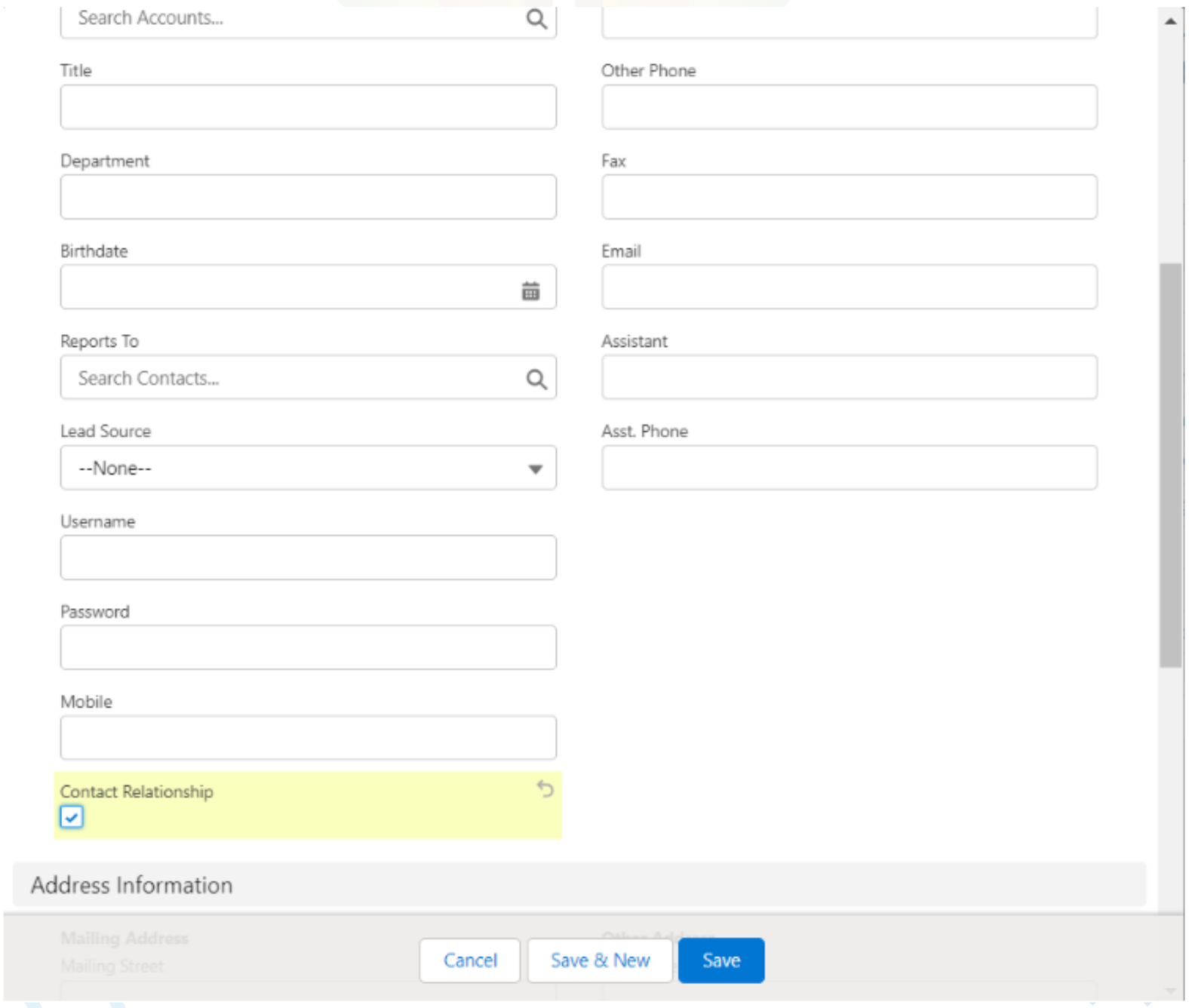
```
if(List_Contacts!=null && List_Contacts.size()>0)
 {
    list<Contact_Relationship__c> List_ConRels= new
                        list<Contact_Relationship__c>();
    List_ConRels= [select id from Contact_
                   Relationship__c where isDeleted=TRUE and
                   Contact__c in:
                   GlobalUtility.getUniqueIds(List_Contacts)];

                   undelete List_ConRels;

  }
 }
}
```

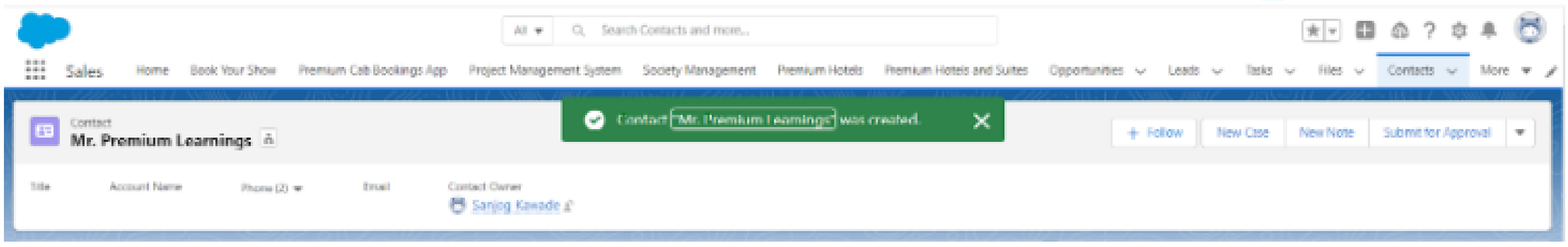

Test Case:

1. Create a new contact as shown below

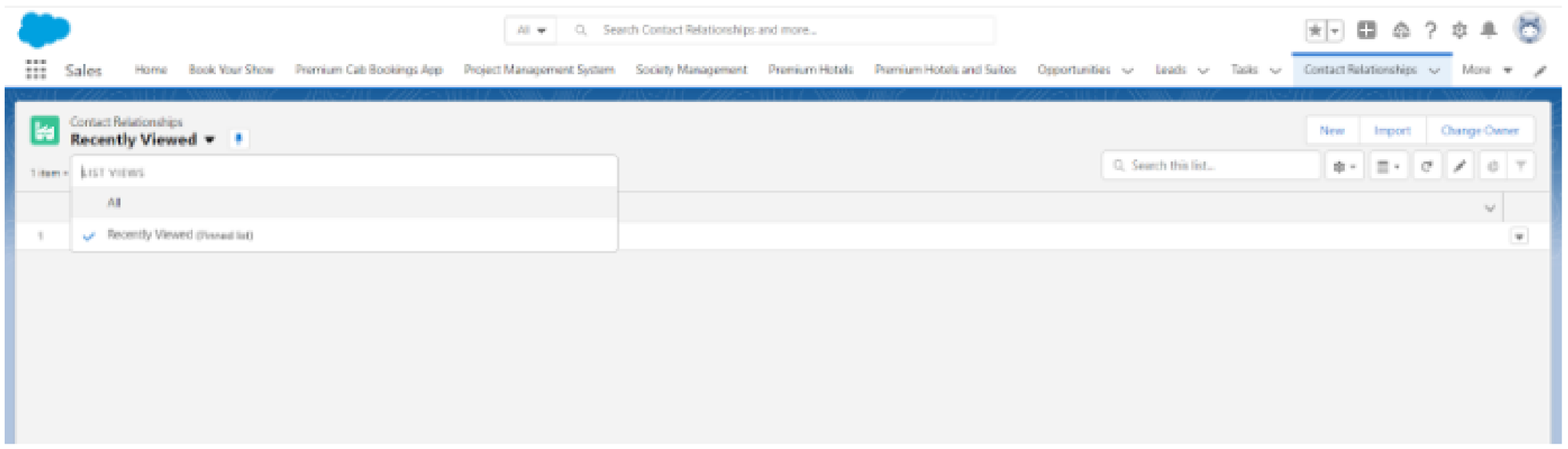


2. make sure the contact relationship checkbox is checked and save.

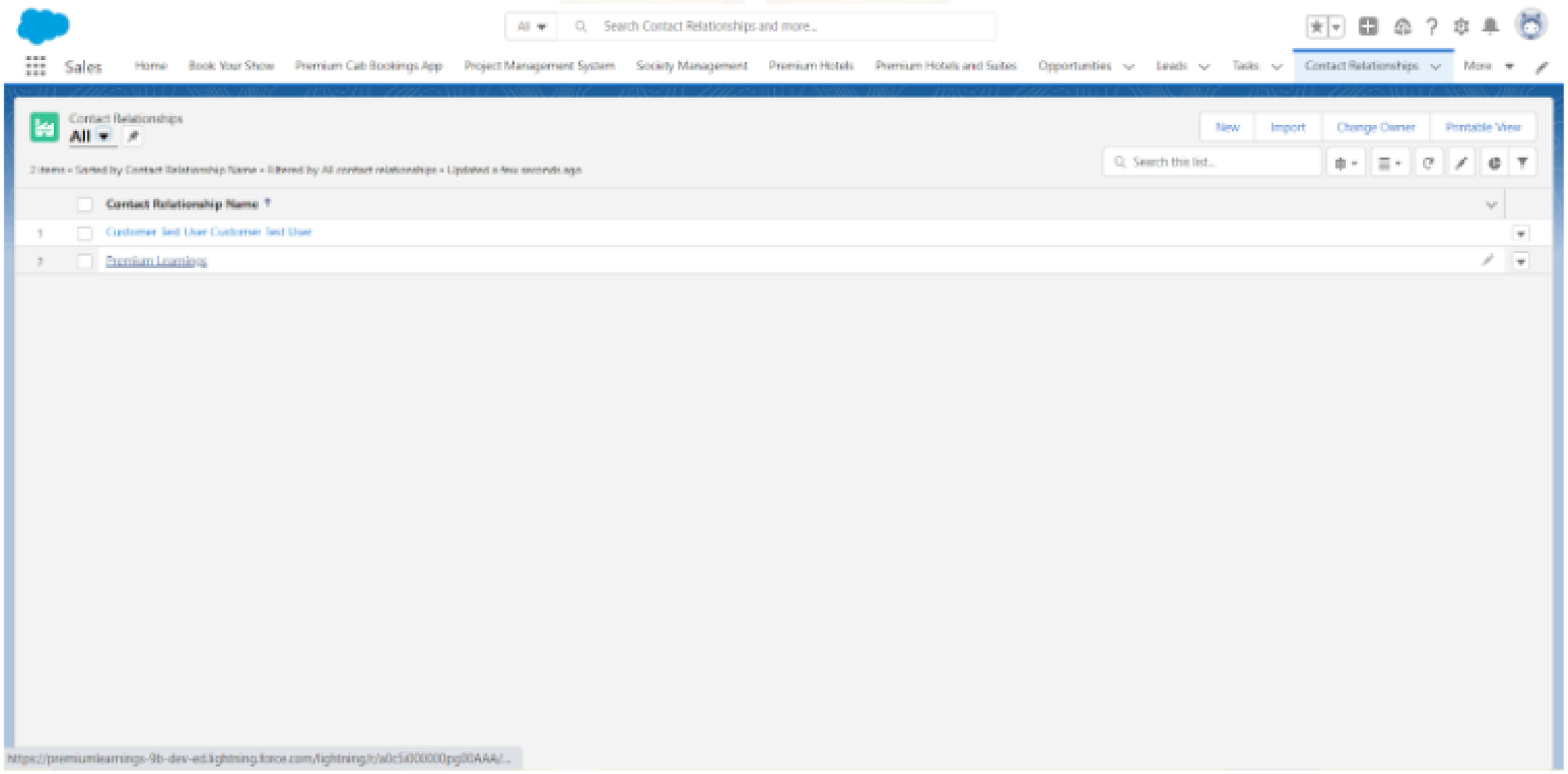




3. Now check the contact relationship object and click to view “all” records



4. You will see a record with same name as contact we created above, automatically created



Example 22

When we change the Owner of the Contact Relationship, then the Owner name will be automatically populated in the Contact Relationship Name field.

ContactRelationshipMasterTrigger(apex trigger)

```
trigger ContactRelationshipMasterTrigger on
Contact_Relationship__c(before update){

if(trigger.isBefore){
 if(trigger.isUpdate){
 //call the handler for the before update trigger event
 updateCROwnerName ConRelUpd = New updateCROwnerName();
 ConRelUpd.updateContactRelationshipNameByOwner(trigger.New);
 }
}
}
```

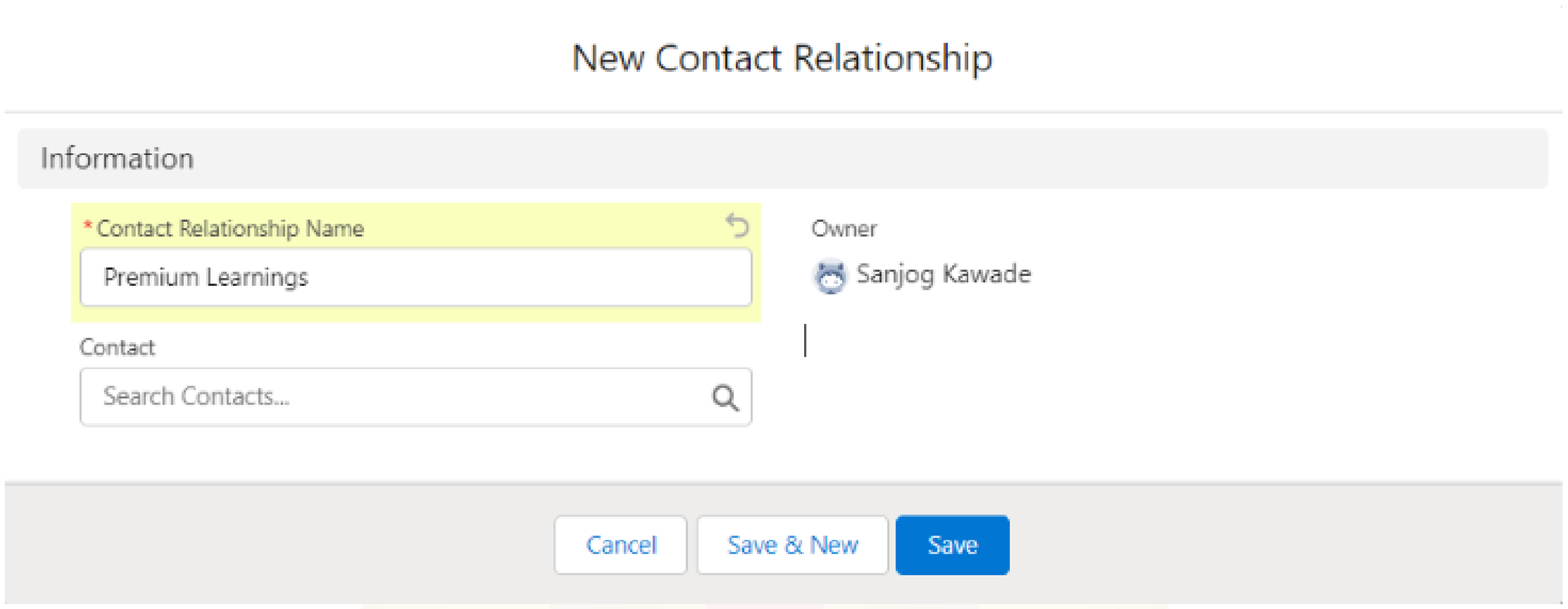
updateCROwnerName (apex class)

```
Public Class updateCROwnerName{
 public void updateContactRelationshipNameByOwner
             (list<Contact_Relationship__c> cont_Rel) {
 map<Id,Id> map_Id_Own = new map<id,id>();
 map<Id,string> map_id_Name = new map<id,string>();
 set<id> Idset = new set<id>();
   for(Contact_Relationship__c List_recs:cont_Rel)
  {
     Idset.add(List_recs.Ownerid);
  }
list<user> u=[select id,Name from user where id in:Idset];
  for(user list_users:u)
  {
    map_id_Name.put(list_users.Id,list_users.Name);
  }
```

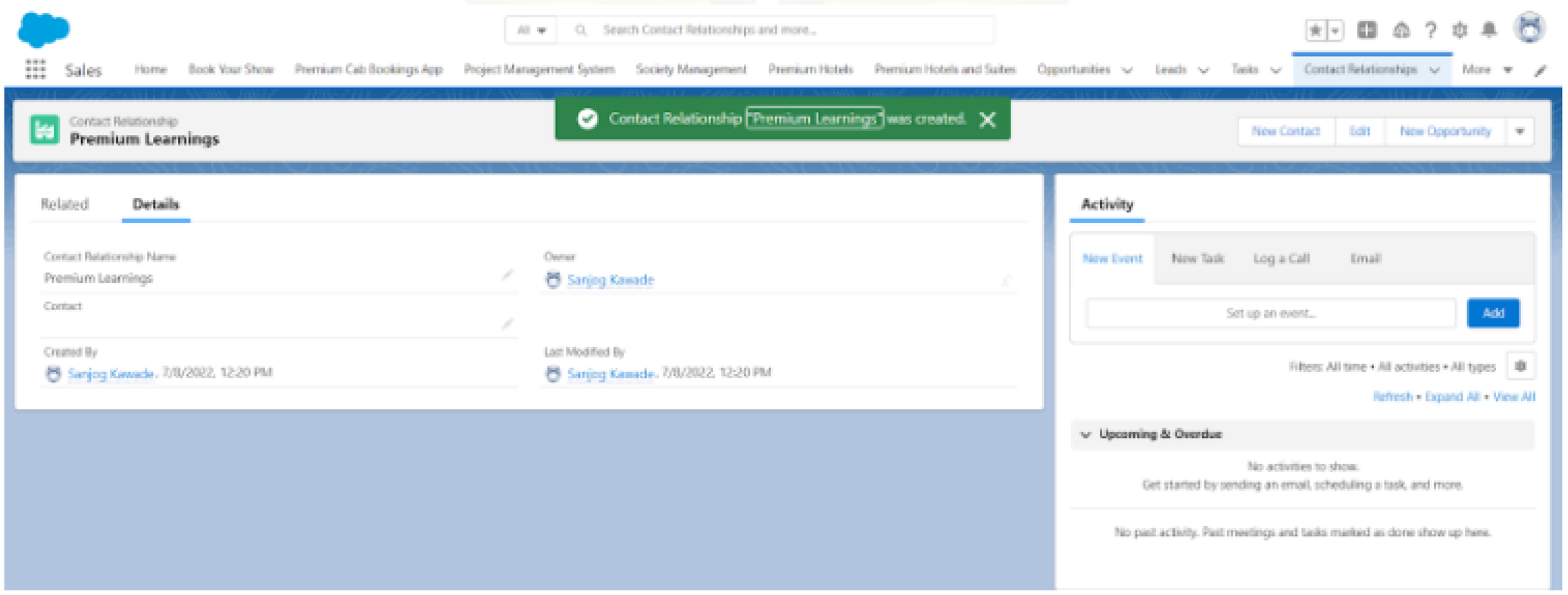
```
if(u!=null && u.size()>0)
{
    for(Contact_Relationship__c List_recs:cont_Rel)
    {
        if (List_recs.Ownerid!=null)
        {
            List_recs.Name =
                map_id_Name.get(List_recs.Ownerid);
        }
    }
}
}
```

Test Case:

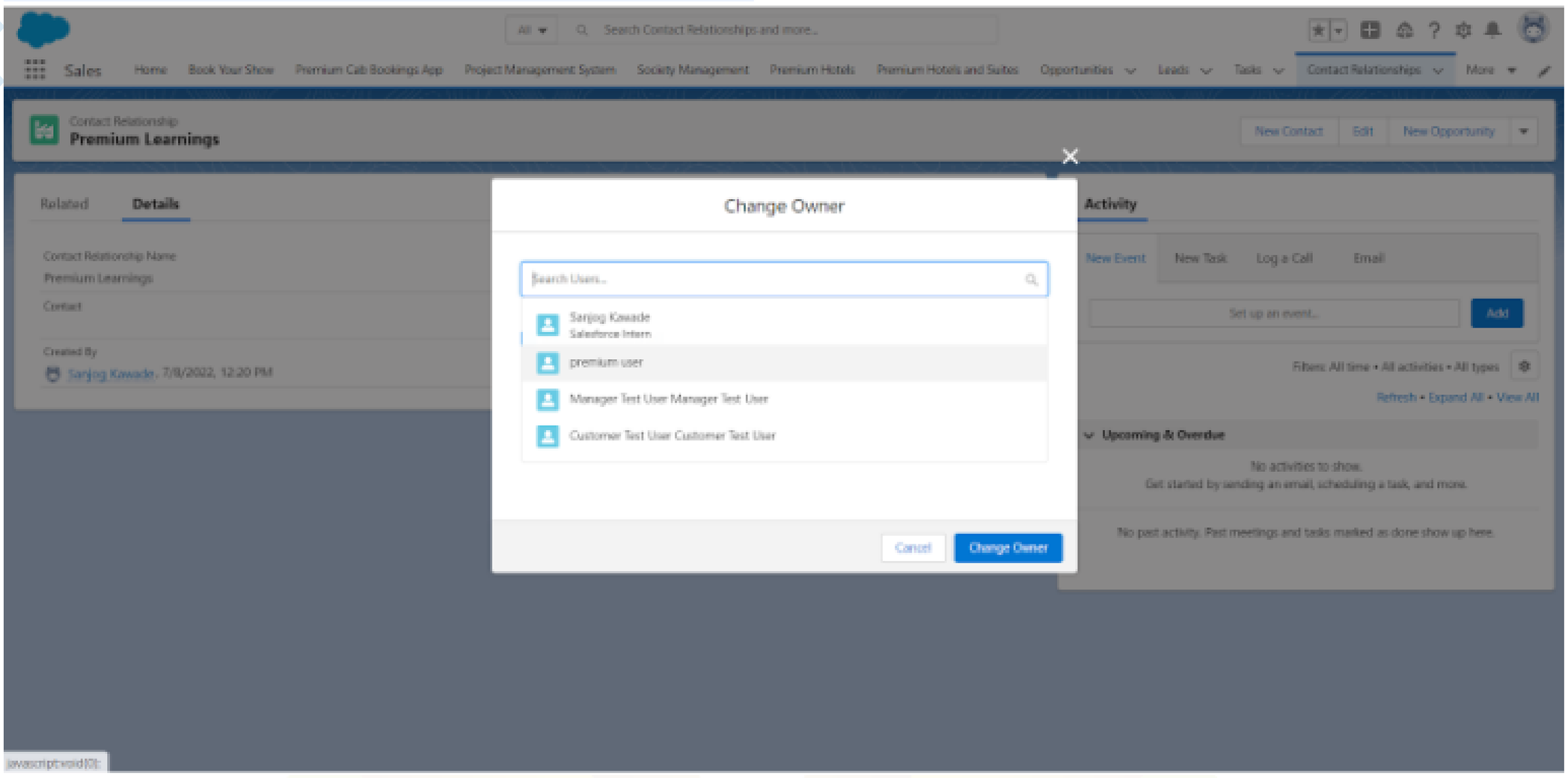
1. After you create a contact relationship object, create a new record for it. And save



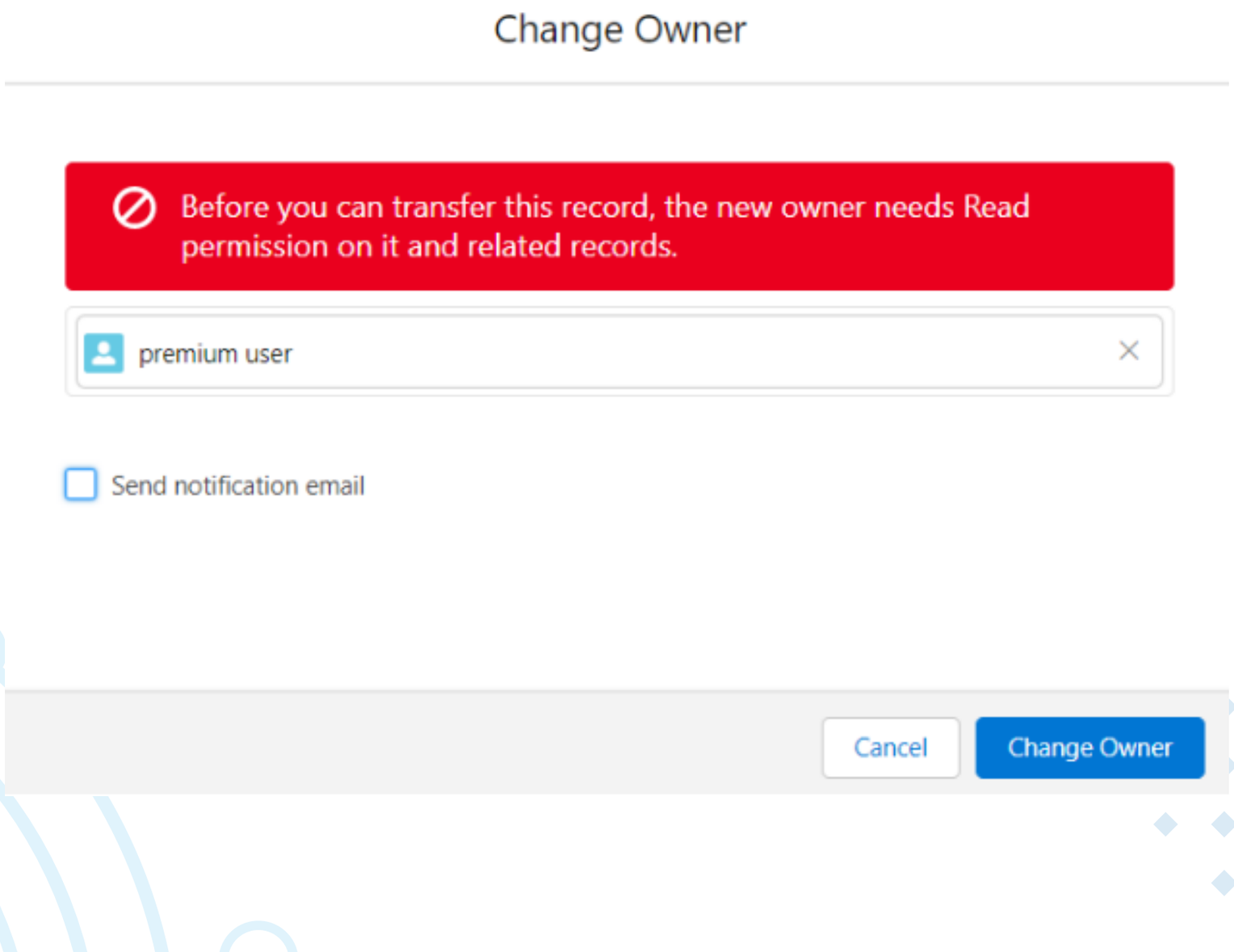
2. The record is saved.(notice owner name and contact relationship name)

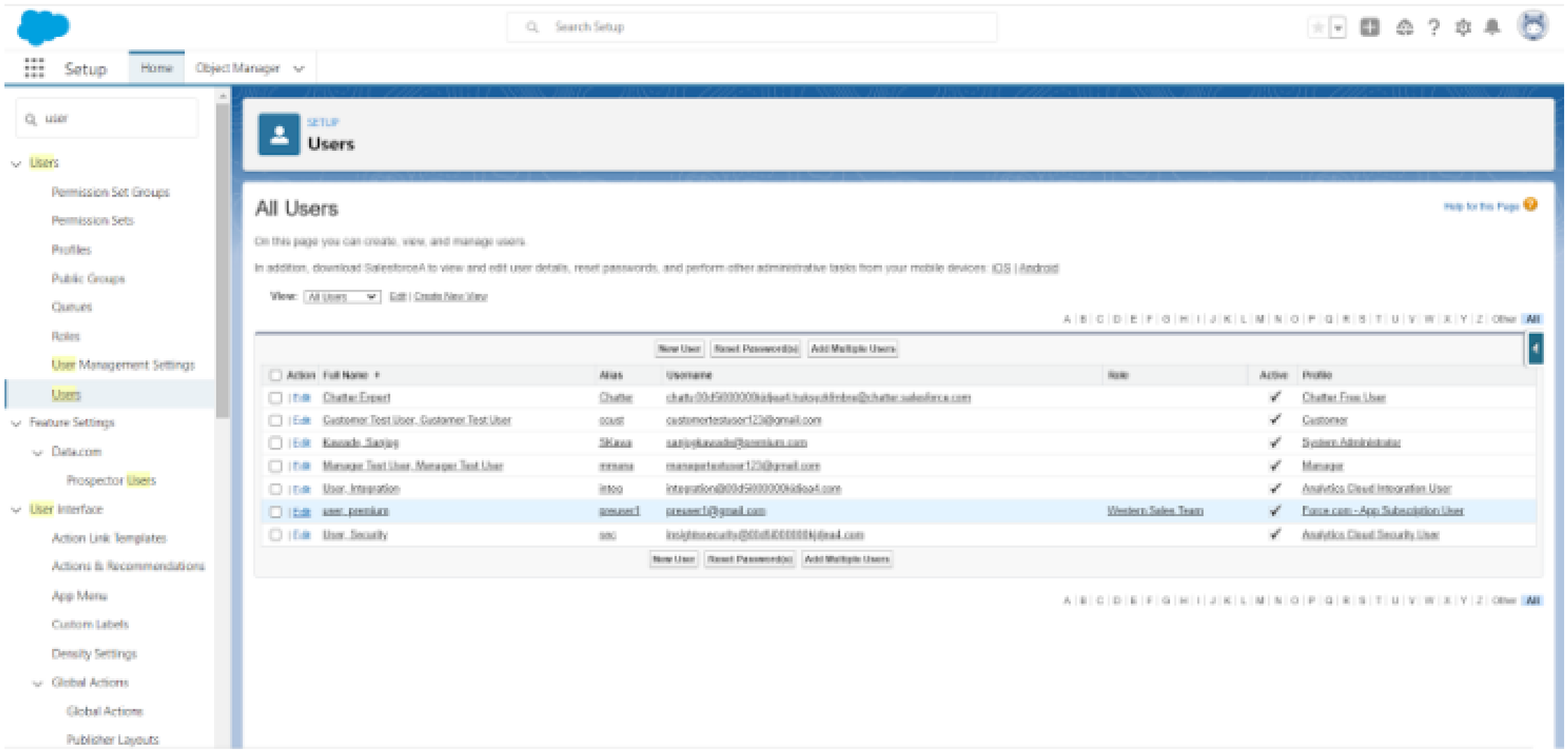


3. Now edit the record to change the owner name and make any other user owner of the record.

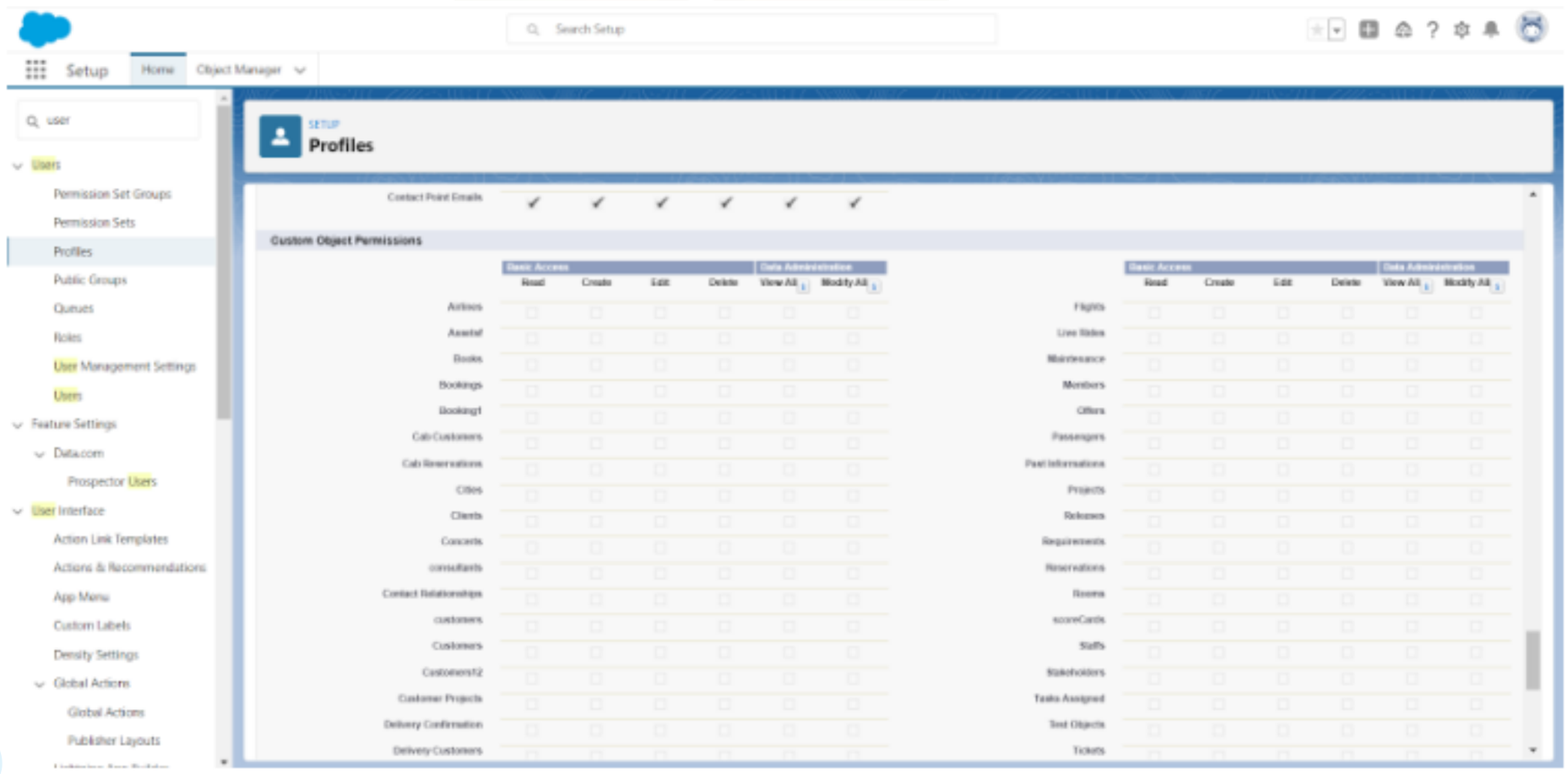


4. If you get the following error, go to setup > home > quick find > users > select user name that you wish to make owner of this record(customer test user in our case) > check the profile name that user is associated with(in our case it is customer)

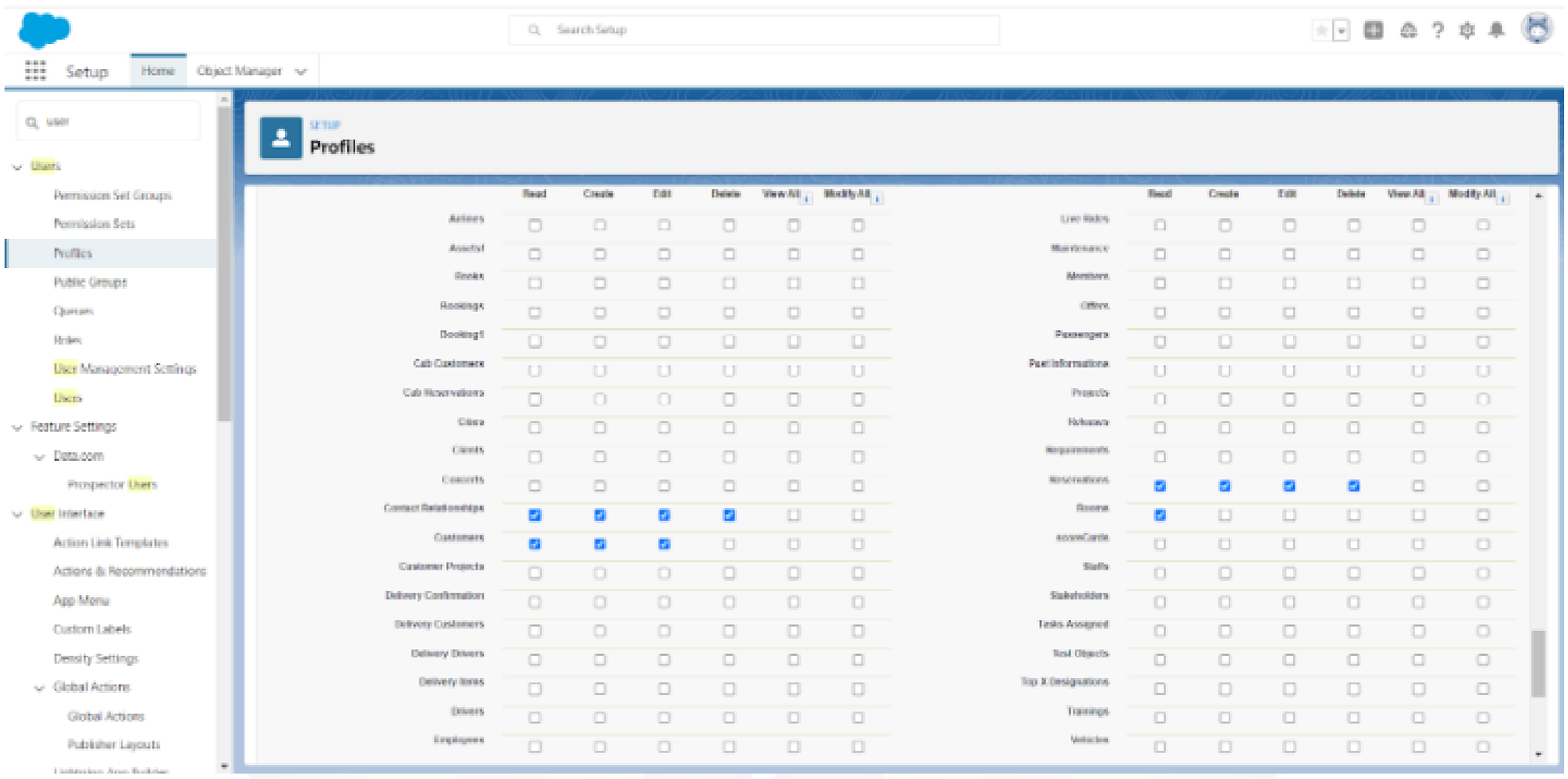




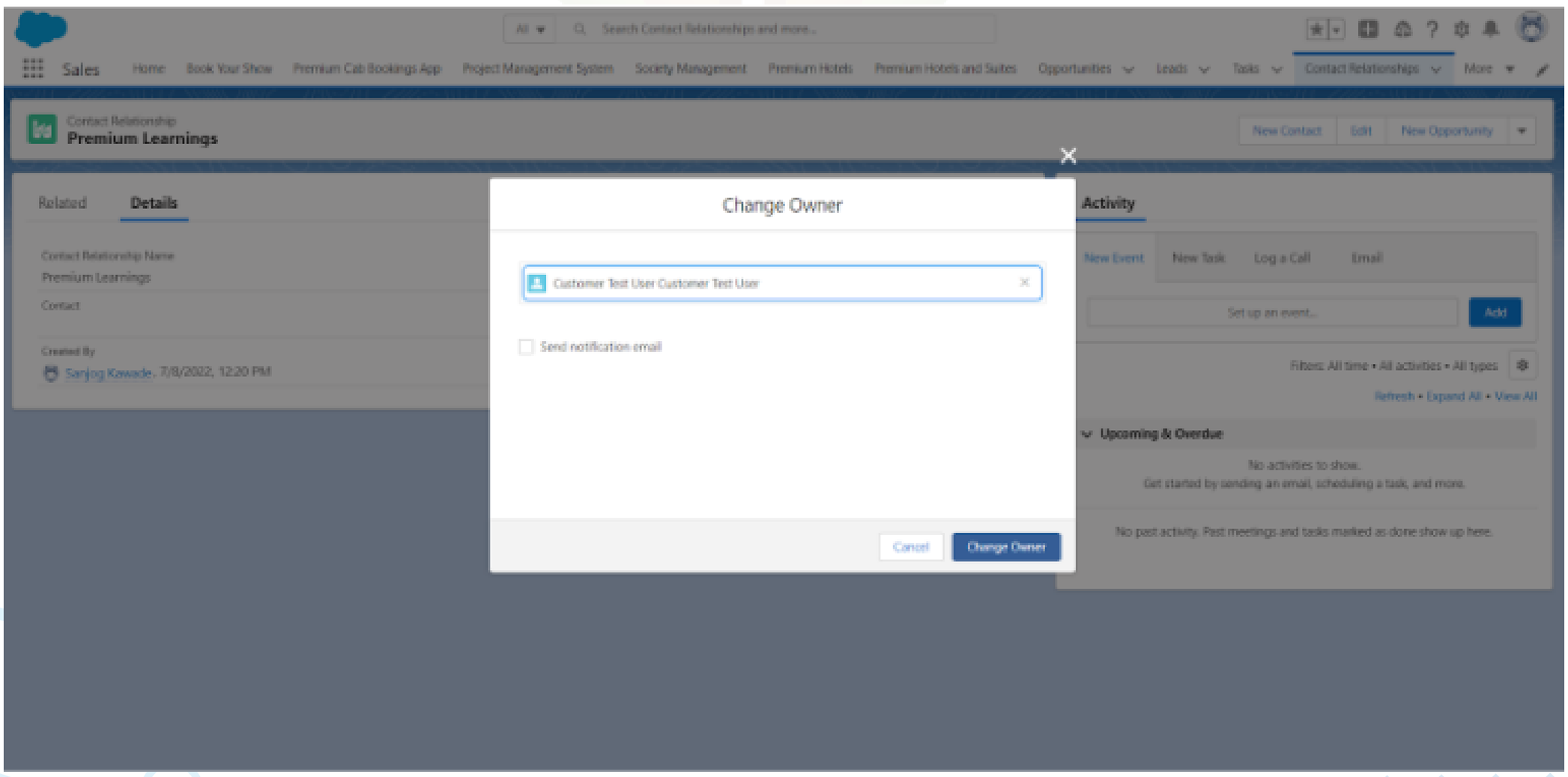
5. Now go to setup > home > quick find > profile > search the profile customer. Scroll down to find custom object permissions



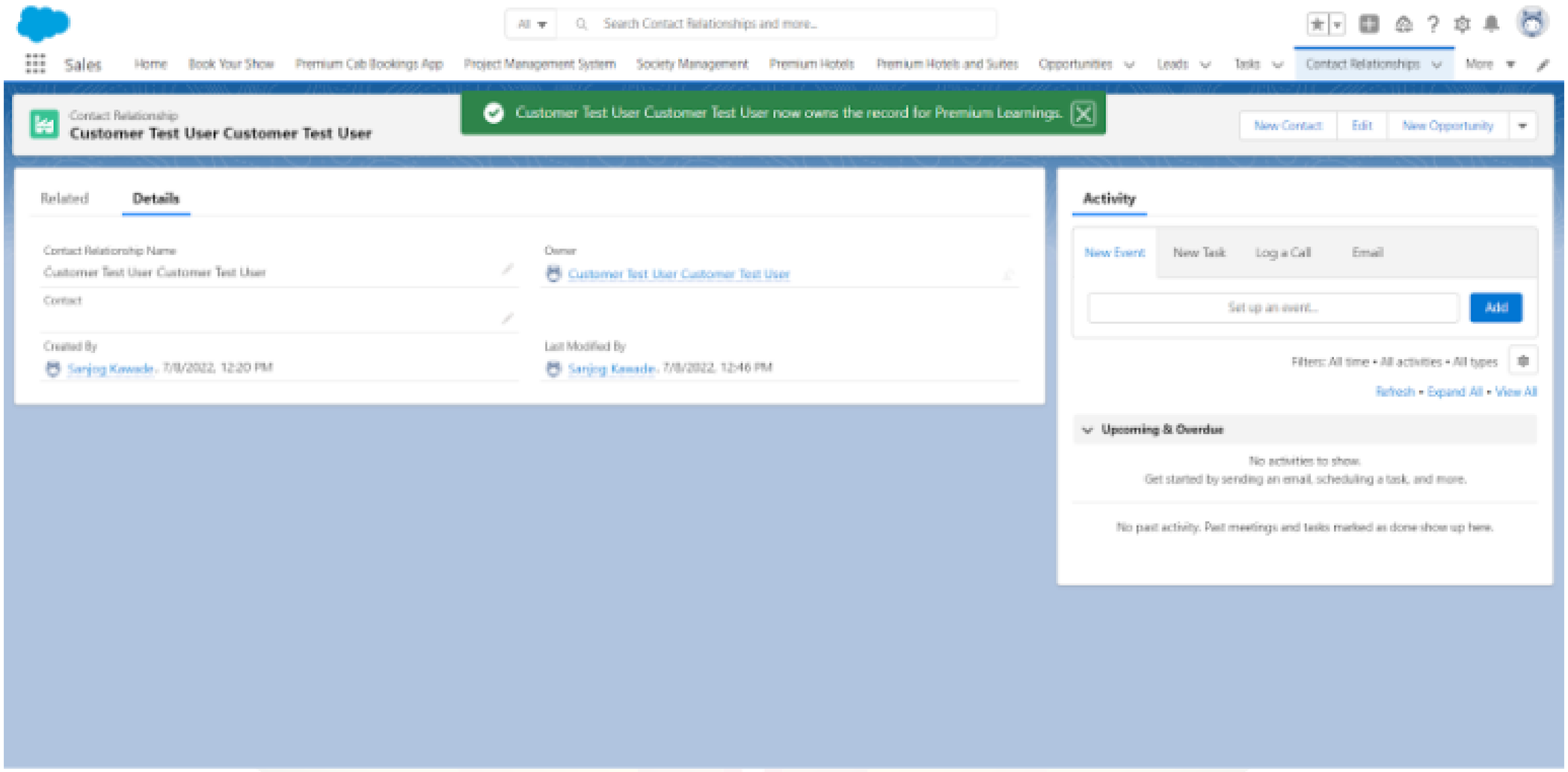
6. Find contact relationship object and give read create edit and delete permissions(tick the respective boxes) and click save.



7. Now select the user as owner as explained in step 3 and change owner



8. Now notice the owner name and customer relationship name



Example 23

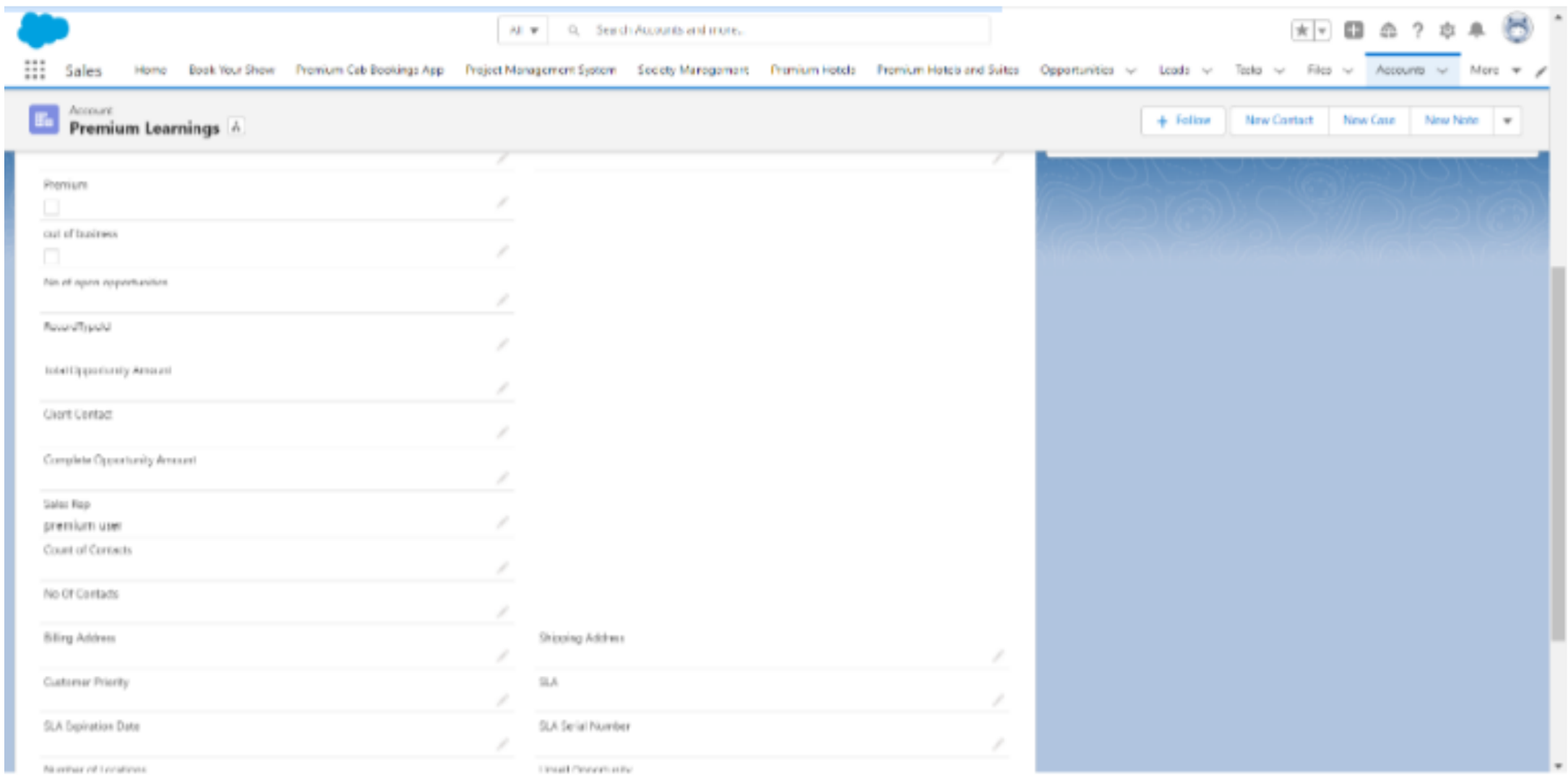
Create field called “Count of Contacts” on Account Object. When we add the Contacts for that Account then the count will populate in the field on the Account details page. When we delete the Contacts for that Account, the Count will update automatically.

```
trigger CountOfContacts on Contact (after insert,
after delete){

 set<id> accid=new set<id>();
 list<contact> contactlist =new list<contact>();
 list<contact> listcon=new list<contact>();
 list<account> acclist=new list<account>();
 list<account> listAcc=new list<account>();
 map<id,integer> mapCount=new map<id,integer>();

 if(trigger.isinsert){
   for(contact con:trigger.new){
       accid.add(con.accountid);
   }
 }

 if(trigger.isdelete){
   for(contact con:trigger.old)
  {
      accid.add(con.accountid);
   }
 }

 acclist=[select id,name from account where id in:accid];
 contactlist = [select id,name,accountid from contact
                where accountid in:accid];
```

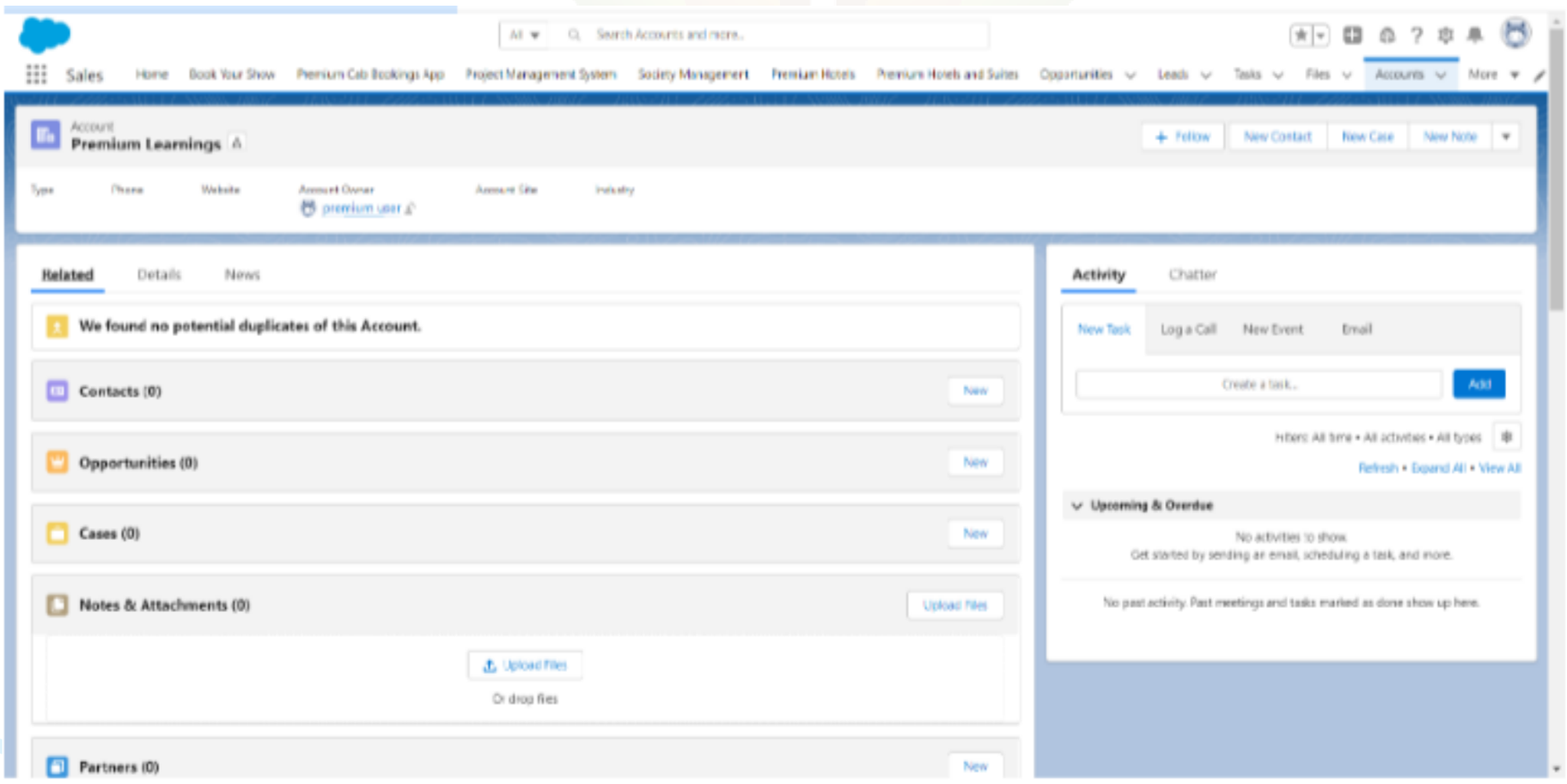
```
    for(account acc:acclist)
    {
        listcon.clear();
        for(contact c:contactlist){
            if(c.accountid==acc.id){
                listcon.add(c);
                mapCount.put(c.accountid,listcon.size());
            }
        }
    }

if(acclist.size()>0){
    for(Account a:acclist){
        if(mapCount.get(a.id)==null)
          a.Count_Of_Contacts__c=0;
    else
    {
        a.Count_Of_Contacts__c=mapCount.get(a.id);
        listAcc.add(a);
    }
    }
  }

  if(listAcc.size()>0)
    update listAcc;
}
```

Test Case:

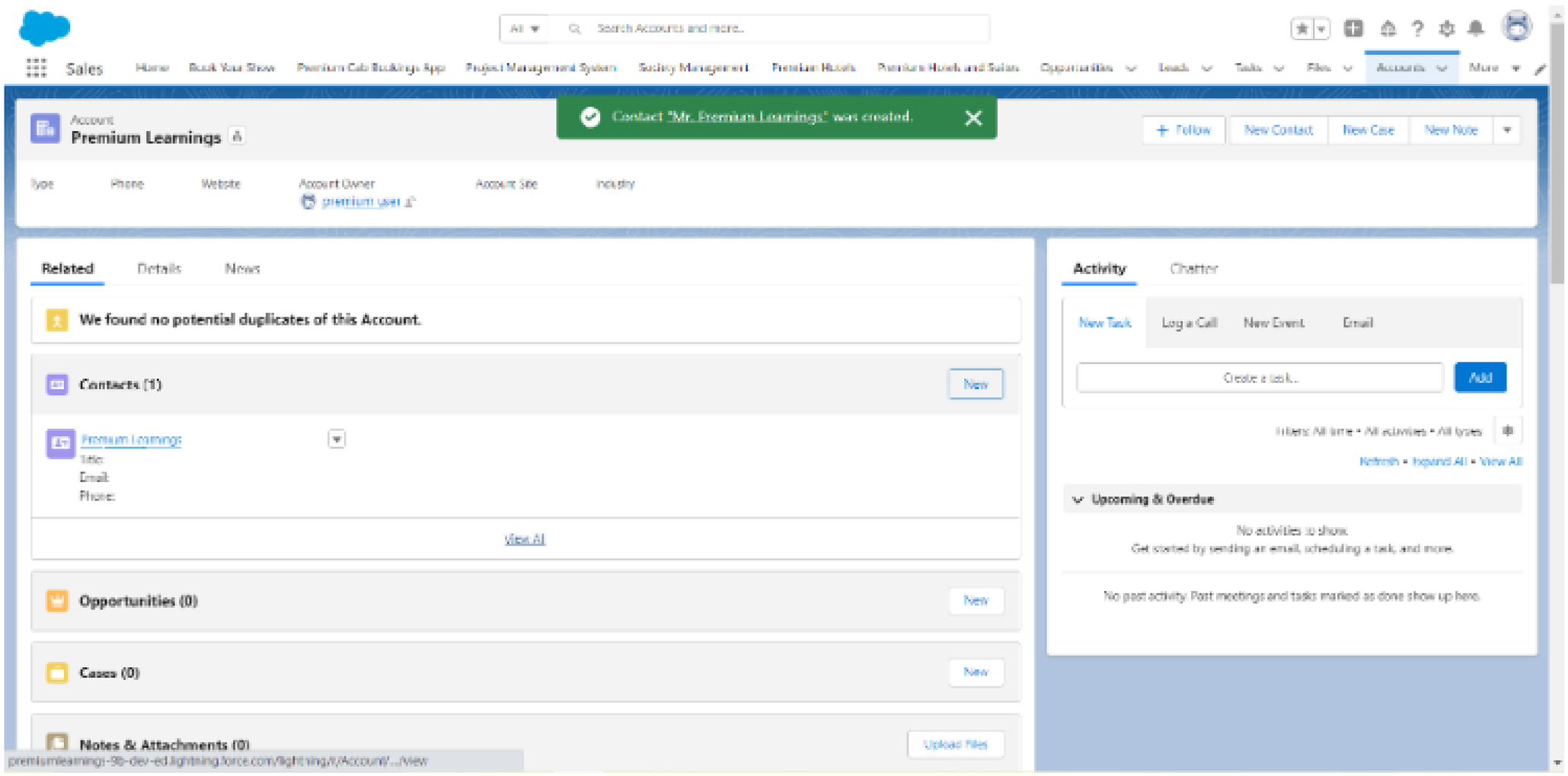
1. Create an account record without any contacts related to the record and click save. Notice the no of contacts field is blank.



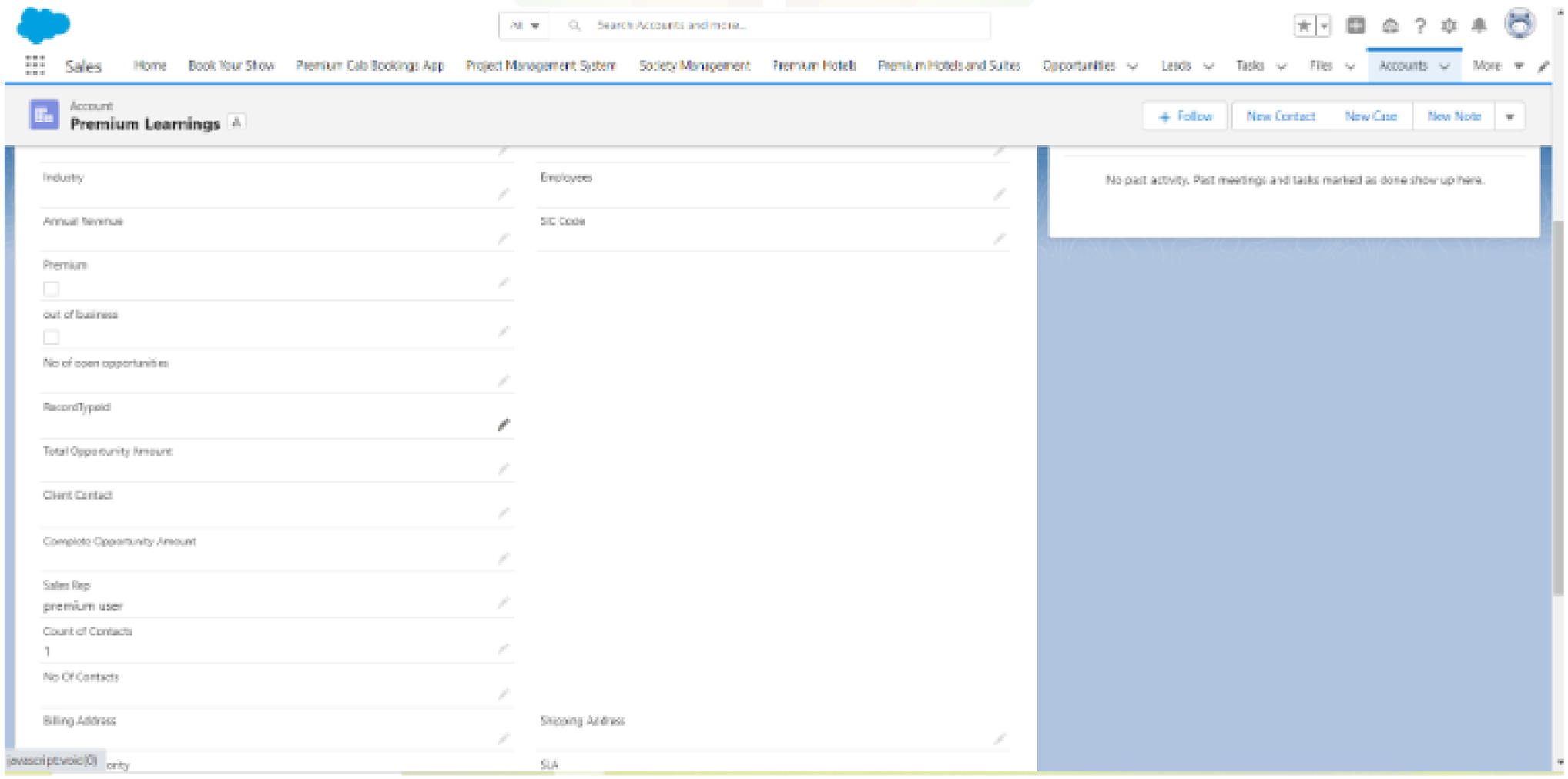
2. Now go to the related lists of the account and notice there are no contacts related to this account



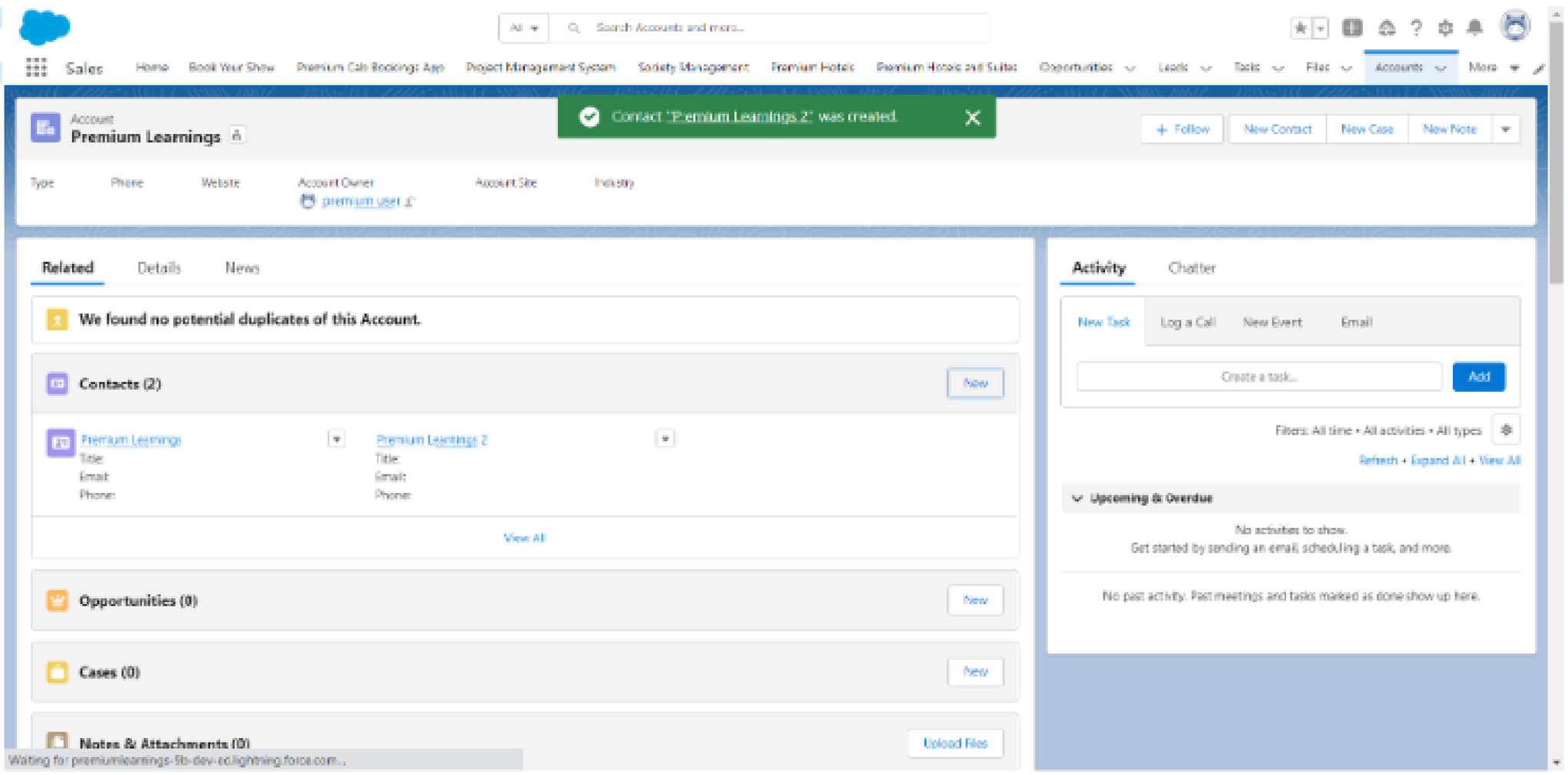
3. Add a contact in the related lists by clicking new button; fill in all the details and click save.

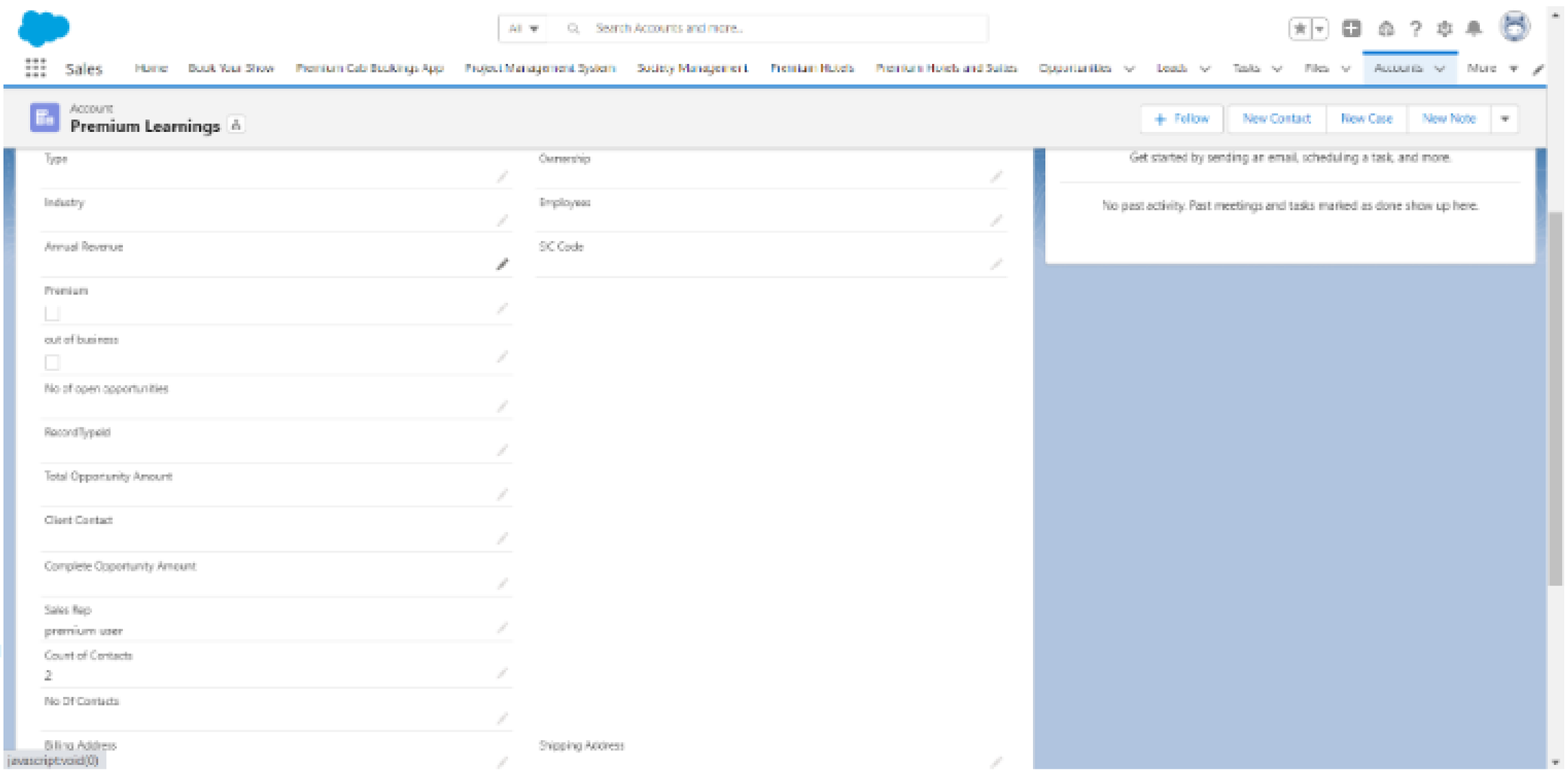


4. Now check the account details, and the field count of contacts. It will show the value 1

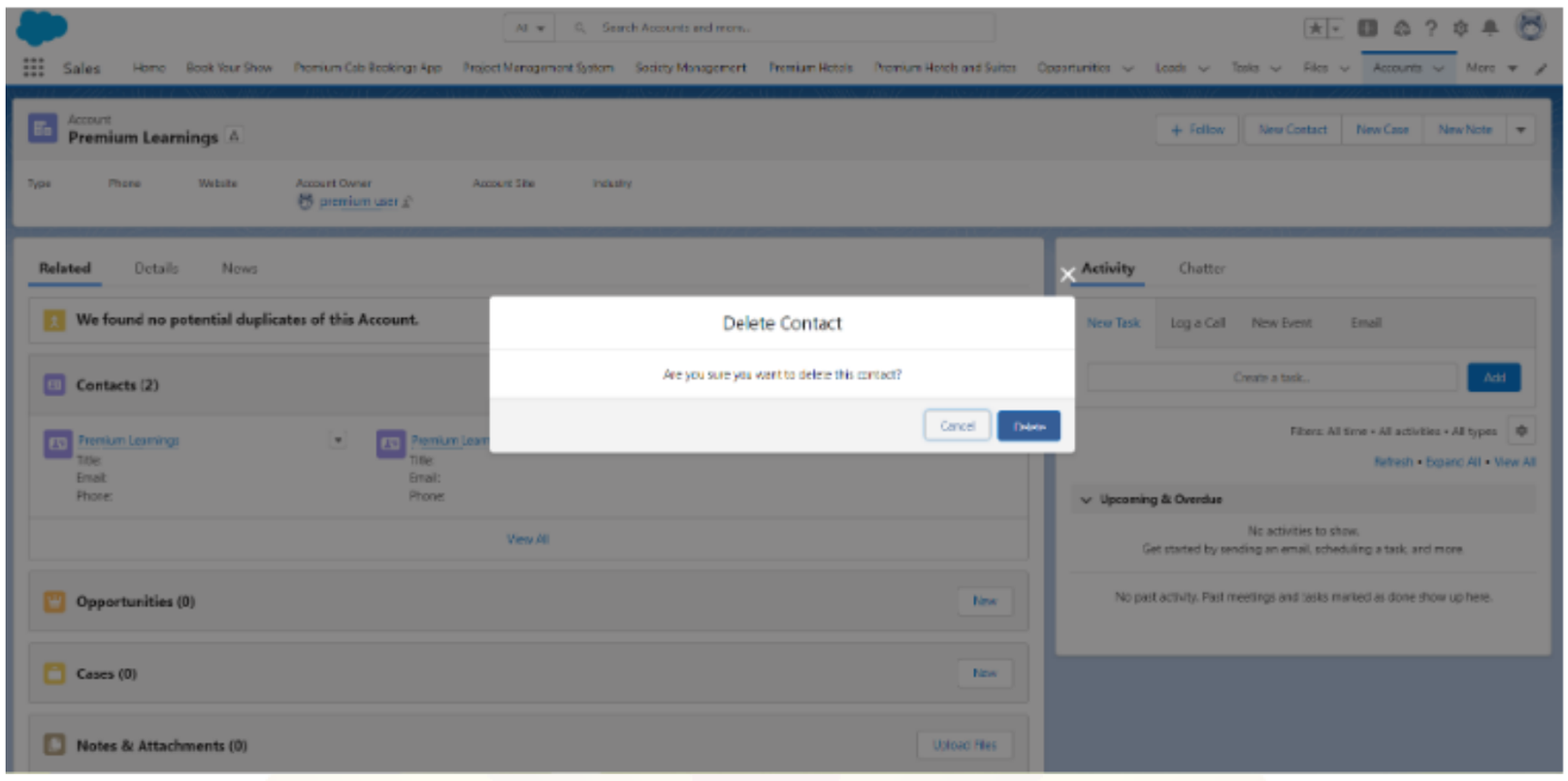


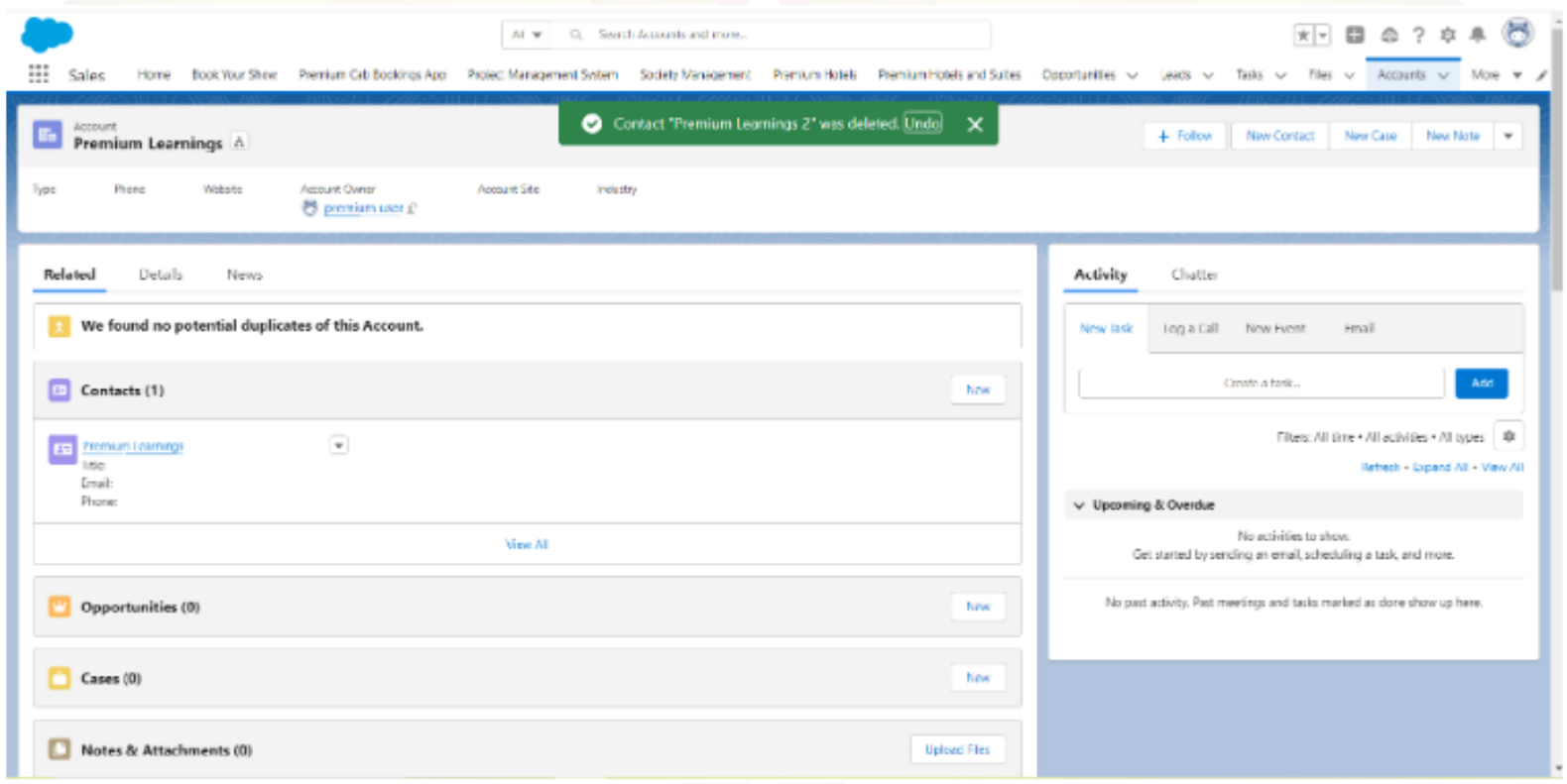
5. repeat step 3 and add another contact, check the count of contactsfield now getting updated to 2

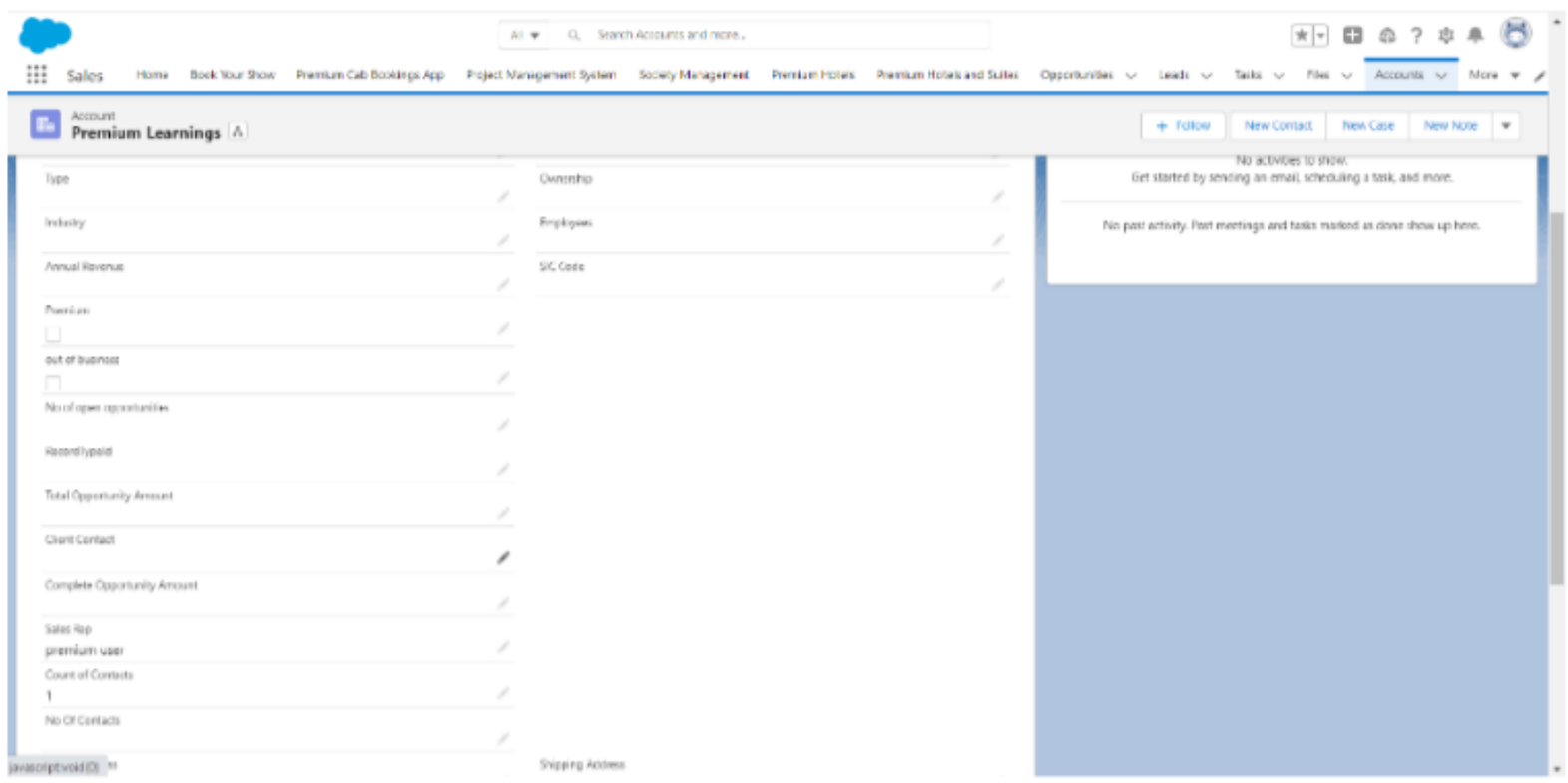




6. Now delete one of the just created contacts and check the count of contacts field getting updated to 1 again







Example 24

Create the object called “Customer12” and create the Master-Detail Relationship on Customer12 object so that Customer12 will be the related list to the Account record. Create the field called “Account Manager” on the Customer12 object which is a lookup to the user object. Now Logic is when we create a Customer12 record for an account record, then the user in the Account Manager field will be automatically added to the Account Team of that associated account.

pre-requisite:

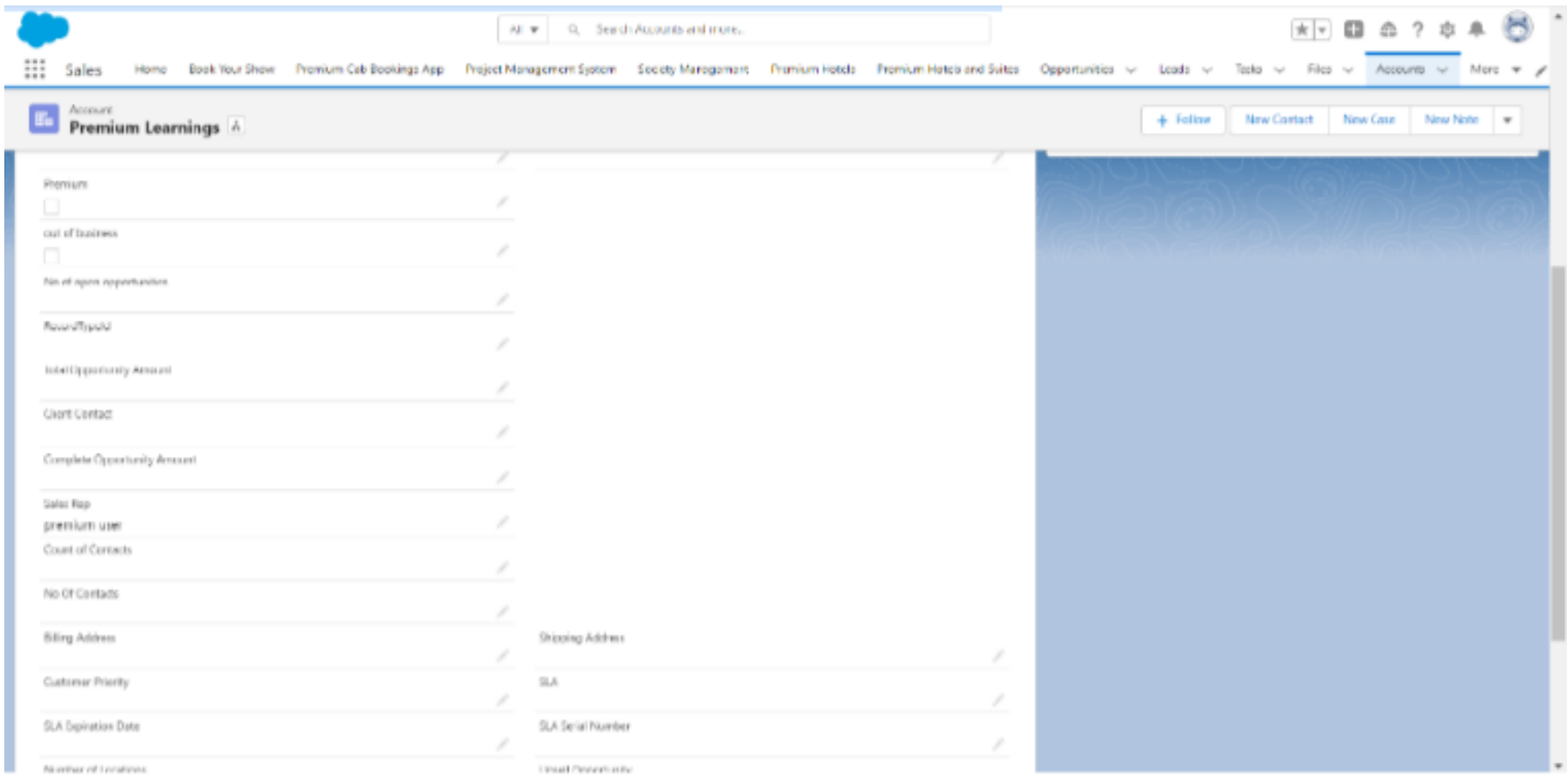
From Setup, enter Account Teams in the Quick Find box, and select Account Teams. Click Enable Account Teams. Select the Account Teams Enabled checkbox and click Save. Select the Account Layout checkbox to add the Account Team related list to the page layout. Select the Add to users' customized related lists checkbox and click Save.

```
trigger InsertAccountTeam on Customer12__c (after insert) {

 List<AccountTeamMember> atm_list=new
                                  List<AccountTeamMember>();
 AccountTeamMember atm = new  AccountTeamMember();
 List<AccountShare> newShare = new List<AccountShare>();

 if(trigger.isInsert) {
   For(Customer12__c c:Trigger.new)
   {
     if(c.AccountManager__c!=null)
{
    atm = new AccountTeamMember();
    atm.accountid=c.Account__c;
    atm.teamMemberRole='Account Manager';
    atm.UserId=c.AccountManager__c;
    AccountShare shares = new AccountShare();
    shares.AccountId=c.Account__c;
```

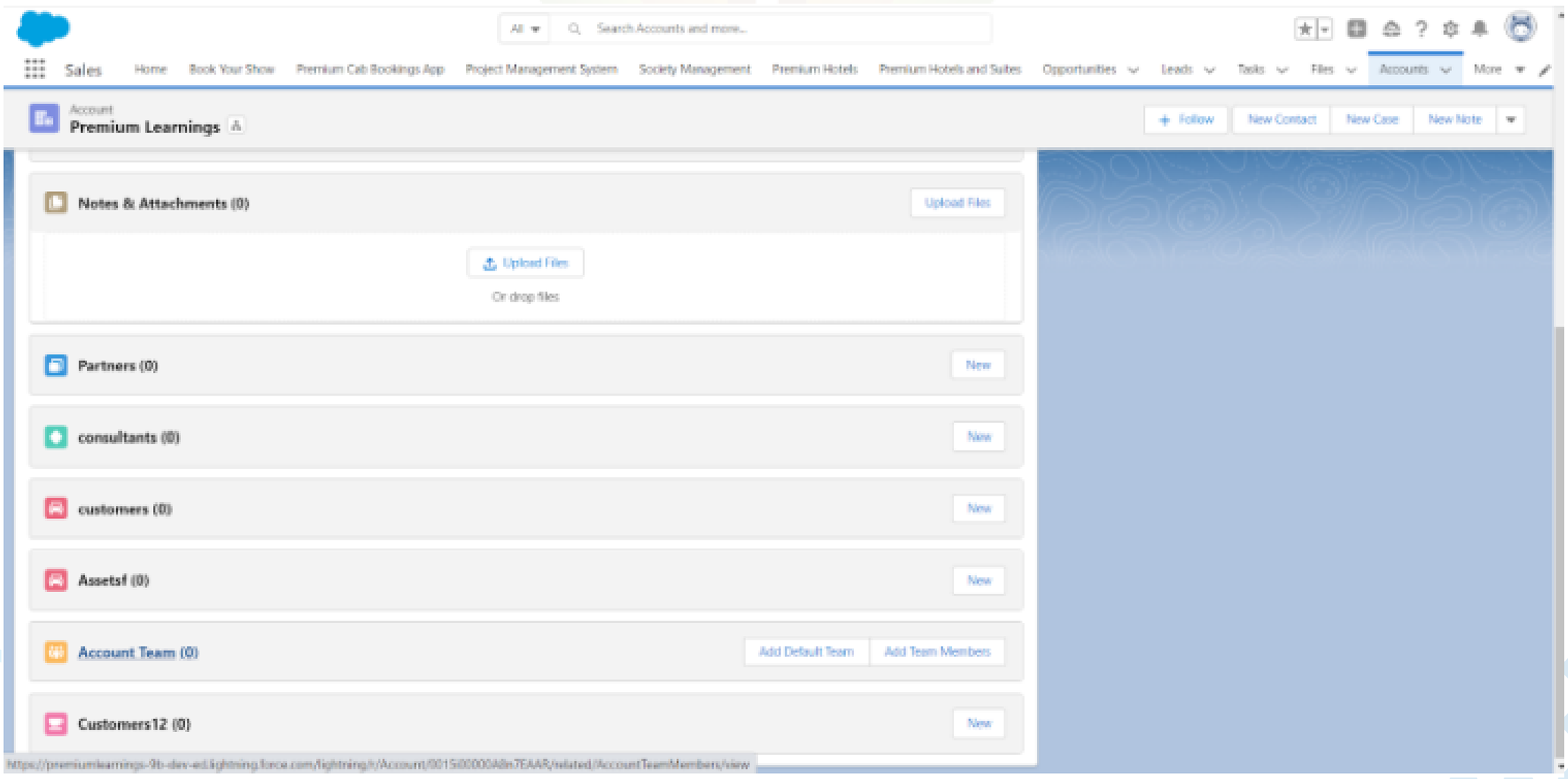
```
            shares.UserOrGroupId = c.AccountManager__c;
            shares.AccountAccessLevel='Read/Write';
            shares.OpportunityAccessLevel = 'Read Only';
            shares.CaseAccessLevel='Read Only';
            newShare.add(shares);
             atm_list.add(atm);
          }
       }
      if(atm_list!=null)
          insert atm_list;
      if(newShare!=null && newShare.size()>0)
          List<Database.saveresult>sr=
                              Database.insert(newShare,false);
      }
}
```

Test Case:

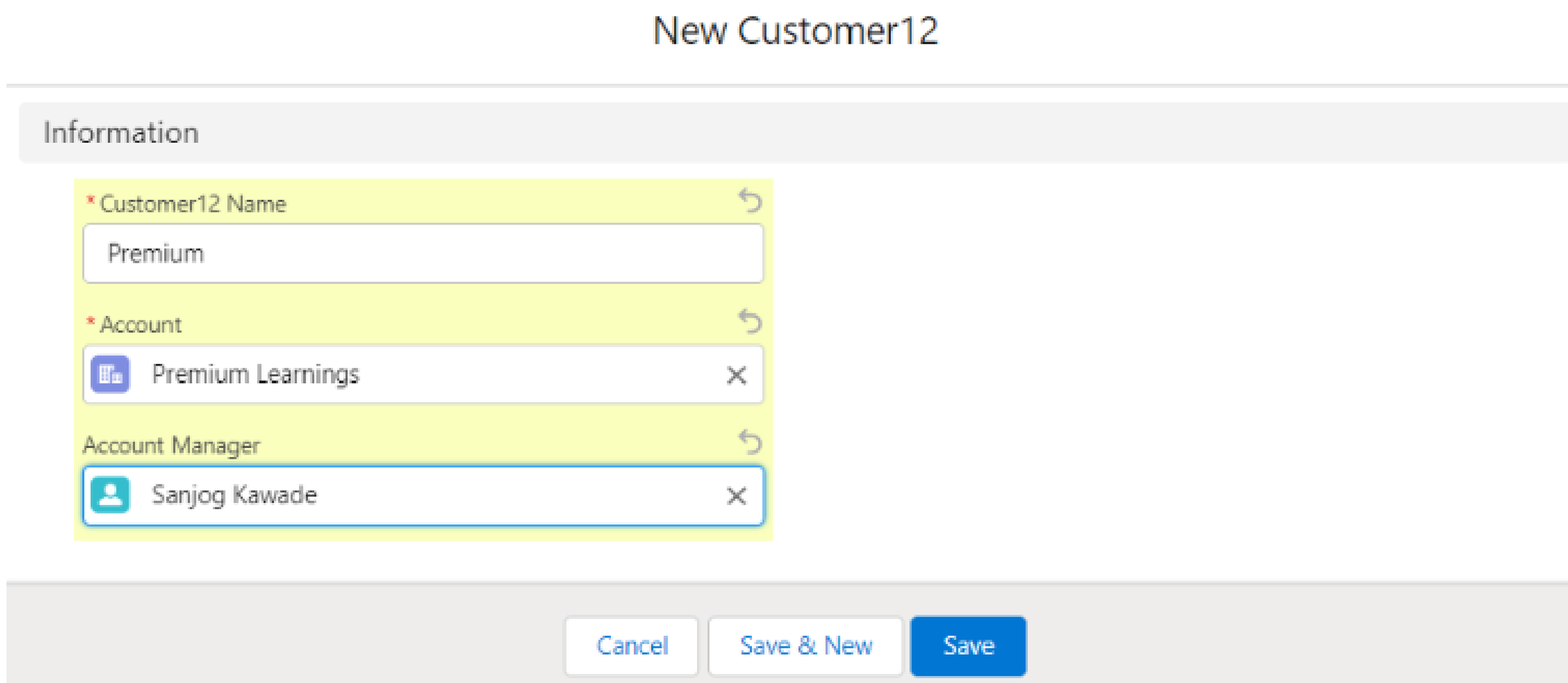
1.Create an account named ‘Premium Learnings’ and save it.



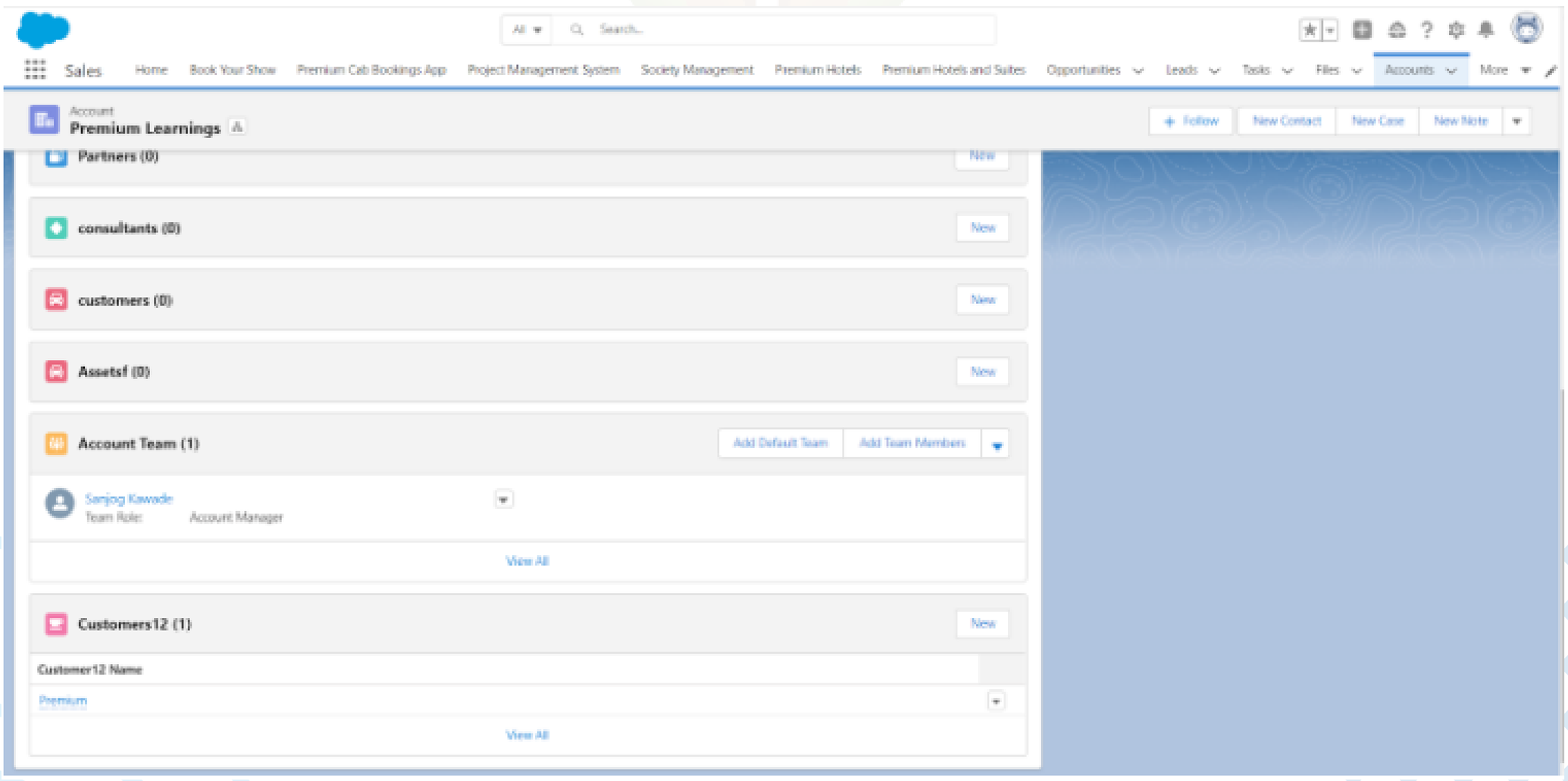
2. Now check the related lists to see the account teams, which will be blank.



3.Now create a Customer12 record with name ‘Premium’. In the account field, select the premium learnings account we created In above step. And in account manager field select your name and click save.



4. Now see the related lists of premium learnings account again, and see the account team; you will find your name(or the account manager you selected in step 3) being populated I the account team.



Example 25

The above trigger Logic(Example 24) is when we create a Customer12 record for an account record, then the user in the Account Manager field will be automatically added to the Account Team of that associated account. Now the following trigger logic is when we update the user in the “Account Manager”, the Account team will be updated automatically.

pre-requisite:

From Setup, enter Account Teams in the Quick Find box, and select Account Teams. Click Enable Account Teams. Select the Account Teams Enabled checkbox and click Save. Select the Account Layout checkbox to add the Account Team related list to the page layout. Select the Add to users' customized related lists checkbox and click Save.

```
trigger UpdateAccountTeam on Customer12__c (before update) {

 List<AccountTeamMember> atm_list=new
                                  List<AccountTeamMember>();
AccountTeamMember atm = new AccountTeamMember();
List<AccountShare> newShare = new List<AccountShare>();

if(trigger.isupdate)
{
  if(trigger.isbefore)
  {
     Set<Id> setAccIds1=new Set<Id>();
     Set<Id> setDelATM=new Set<Id>();
     Map<id,Set<Id>> mapAccMgrs=new Map<id,Set<Id>>();
for(Customer12__c c:Trigger.new)
{
 if(trigger.oldmap.get(c.Id).AccountManager__c !=
       c.AccountManager__c && c.AccountManager__c!=null)
 {
   setAccIds1.add(c.Account__c);
 }
}
```

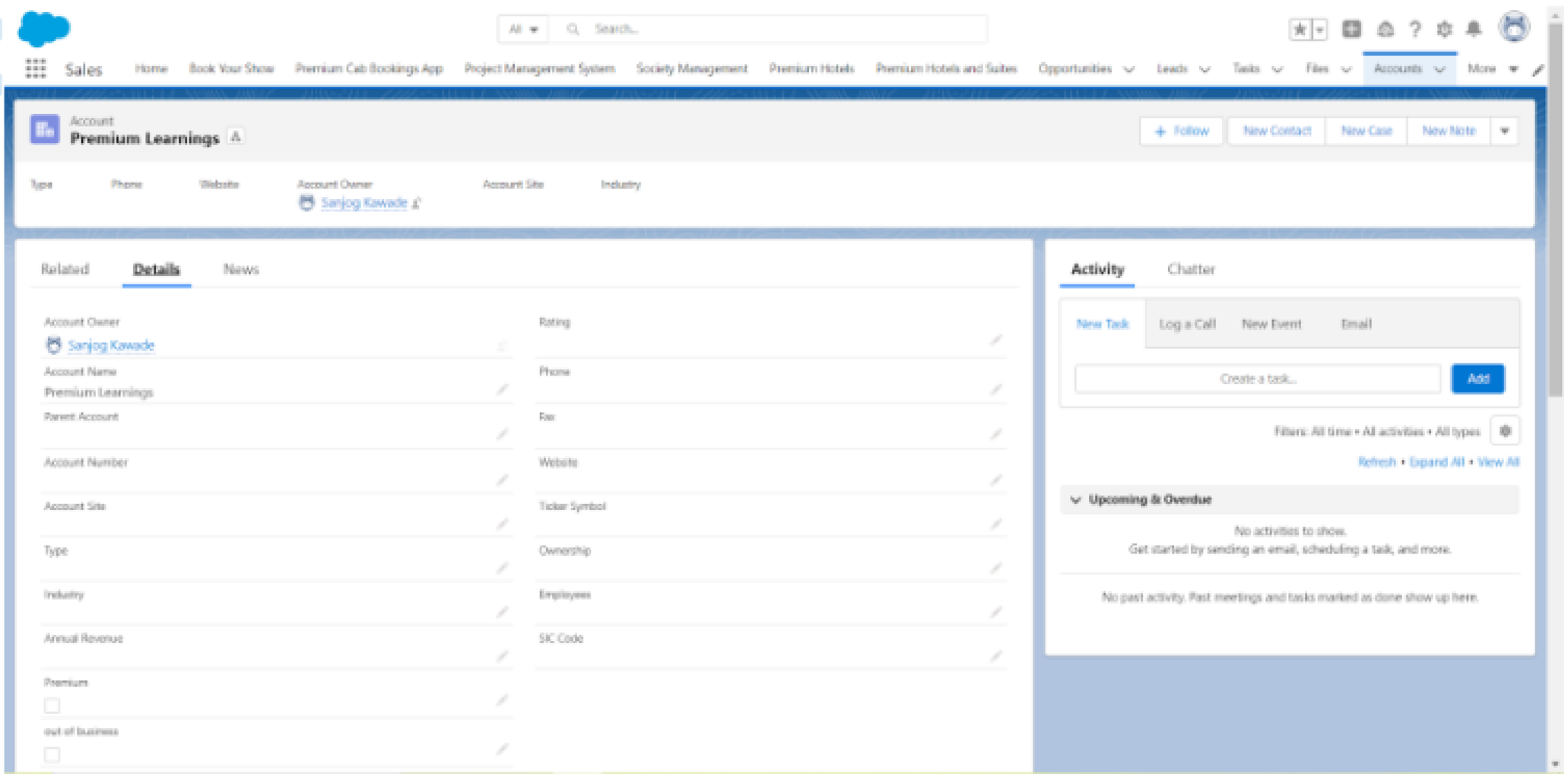
```
List<Customer12__c> listPLAccMgrs=[select id,
        AccountManager__c,Account__c from Customer12__c
        where Account__c in:setAccIds1 and id not
        in:trigger.newmap.keyset()];

if(listPLAccMgrs!=null && listPLAccMgrs.size()>0)
{
 for(Customer12__c c:listPLAccMgrs)
 {
  Set<Id> idMgrs=mapAccMgrs.get(c.Account__c);
    if(null==idMgrs){
       idMgrs=new set<Id>();
       mapAccMgrs.put(c.Account__c,idMgrs);
    }
    idMgrs.add(c.AccountManager__c);
 }
}

Map<id,List<AccountTeamMember>> mapStdAccTeam=new
Map<id,List<AccountTeamMember>>();
List<AccountTeamMember> listStdAccTeam=[select id,
                UserId,AccountId from AccountTeamMember
                where AccountId in:setAccIds1];

if(listStdAccTeam!=null && listStdAccTeam.size()>0){
  for(AccountTeamMember recAccTeam :listStdAccTeam)
  {
   List<AccountTeamMember> listStdAccTeamMap=
                    mapStdAccTeam.get(recAccTeam.AccountId);
  if(null==listStdAccTeamMap)
  {
  listStdAccTeamMap=new List<AccountTeamMember>();
  mapStdAccTeam.put(recAccTeam.AccountId,listStdAccTeamMap);
 }
```

```
 listStdAccTeamMap.add(recAccTeam);
  }
}
system.debug('**********'+mapAccMgrs);

for(Customer12__c c:Trigger.new)
{
 if(trigger.oldmap.get(c.Id).AccountManager__c !=
        c.AccountManager__c && c.AccountManager__c!=null )
  {
List<AccountTeamMember>listAccTeam=
                        mapStdAccTeam.get(c.Account__c);
Set<Id> idMgrs=mapAccMgrs.get(c.Account__c);

if(listAccTeam!=null && listAccTeam.size()>0 )
{
 if(idMgrs!=null && idMgrs.size()>0 && !
   (idMgrs.Contains(trigger.oldmap.get(c.Id)
                                    .AccountManager__c)))
{
  for(AccountTeamMember recATM:listAccTeam)
  {
    if(recATM.UserId==trigger.oldmap.get(c.Id)
                                    .AccountManager__c)
    setDelATM.add(recATM.Id);
   }
}
 else if(idMgrs==null)
  {
    for(AccountTeamMember recATM:listAccTeam)
       setDelATM.add(recATM.Id);
  }
}
```

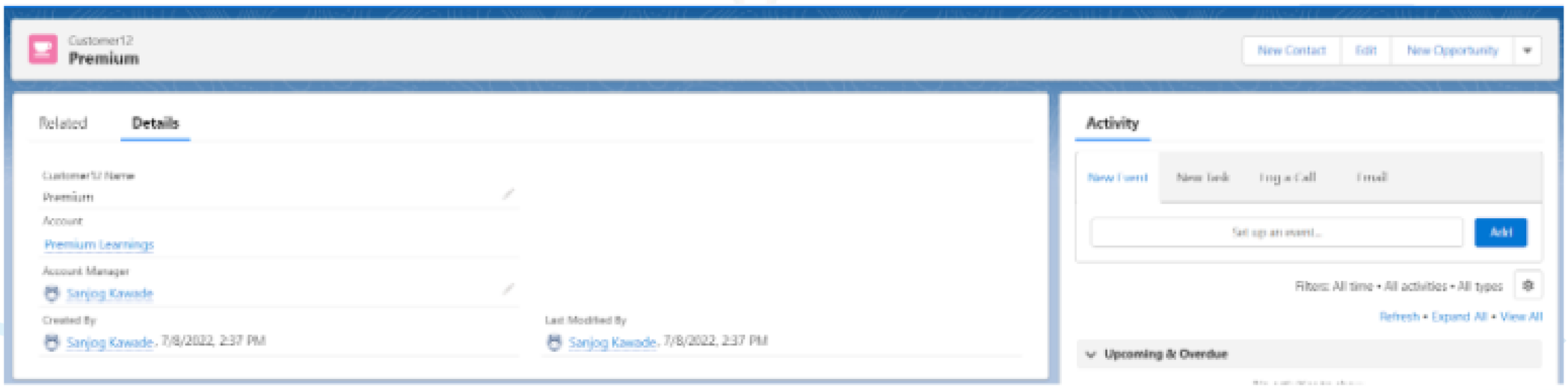
```
    atm = new AccountTeamMember
                (accountid=c.Account__c,teamMemberRole='Account
                 Manager',UserId=c.AccountManager__c);
  AccountShare shares = new AccountShare();
  shares.AccountId=c.Account__c;
  shares.UserOrGroupId = c.AccountManager__c;
  shares.AccountAccessLevel='Edit';
  shares.OpportunityAccessLevel = 'None';
  newShare.add(shares);
  atm_list.add(atm);
    }
  }

  List<AccountTeamMember> listDelATM=[select id from
                        AccountTeamMember where id in:setDelATM];

  if(listDelATM!=null && listDelATM.size()>0 )
  {
    delete listDelATM;
    if(atm_list!=null)
    {
      insert atm_list;
     if(newShare!=null && newShare.size()>0)
      List<Database.saveresult> sr=
                                  Database.insert(newShare,false);
    }
  }
}
```

Test Case:

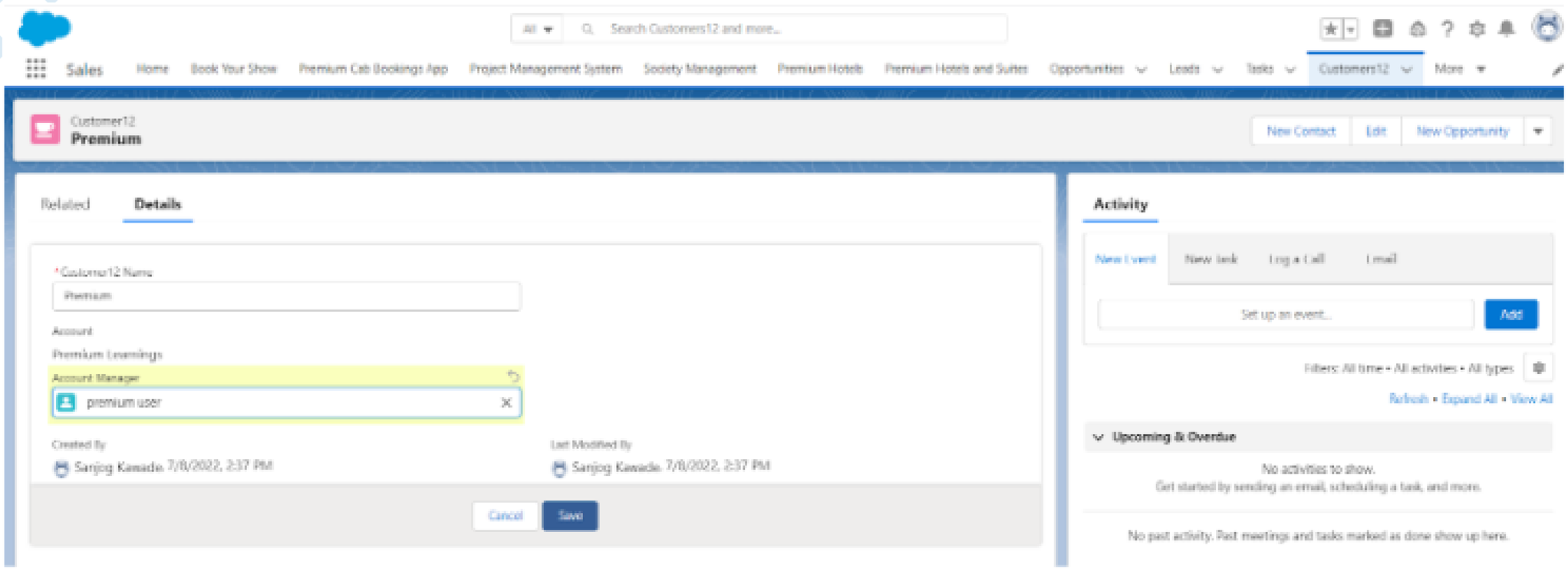
1.Create an account record named ‘Premium Learnings’



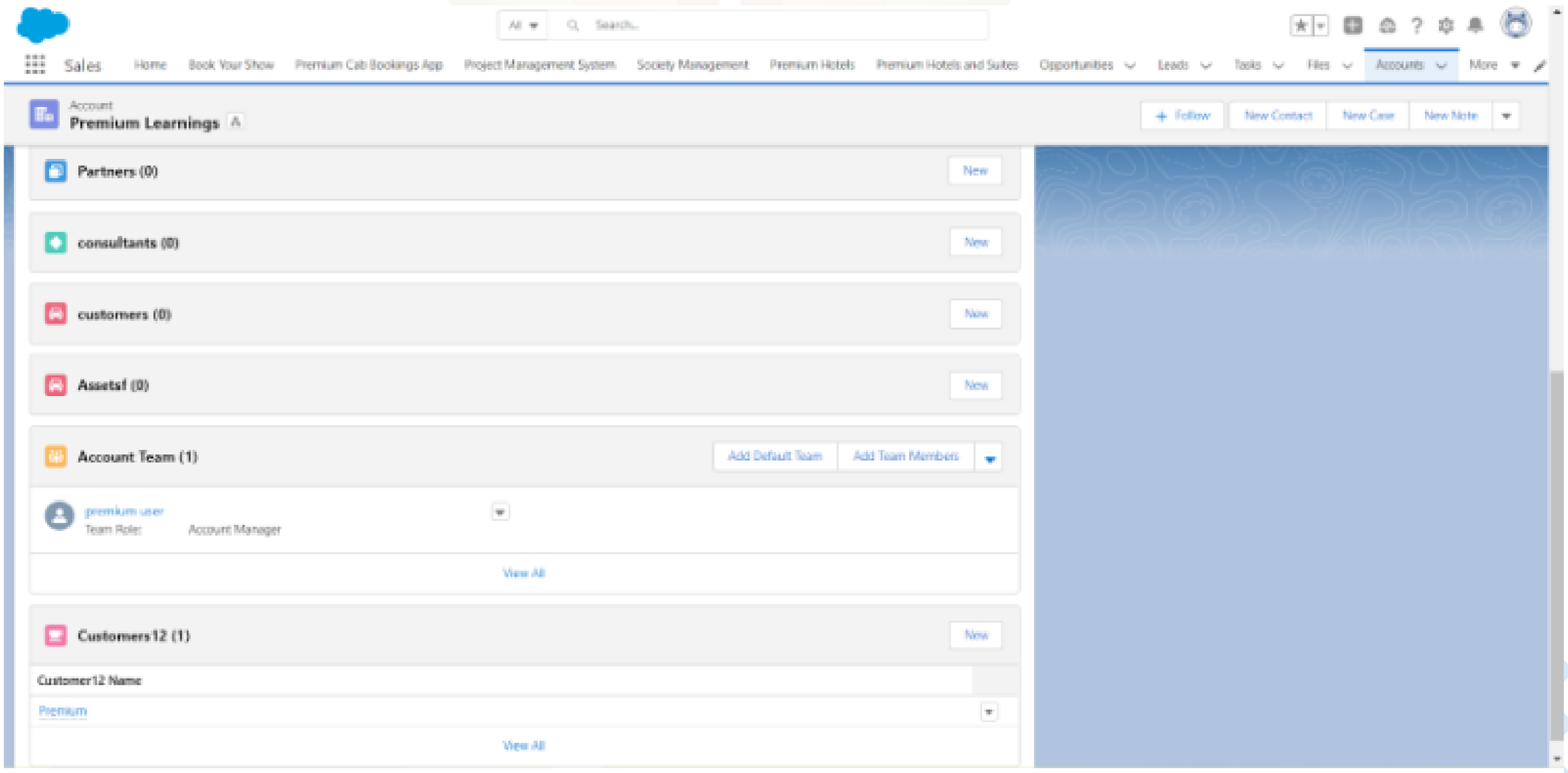
2. create a Customer12 record as shown below. (Put your name/your orgs system administrator in account manager field). Select the account we created above in the account field of the record.



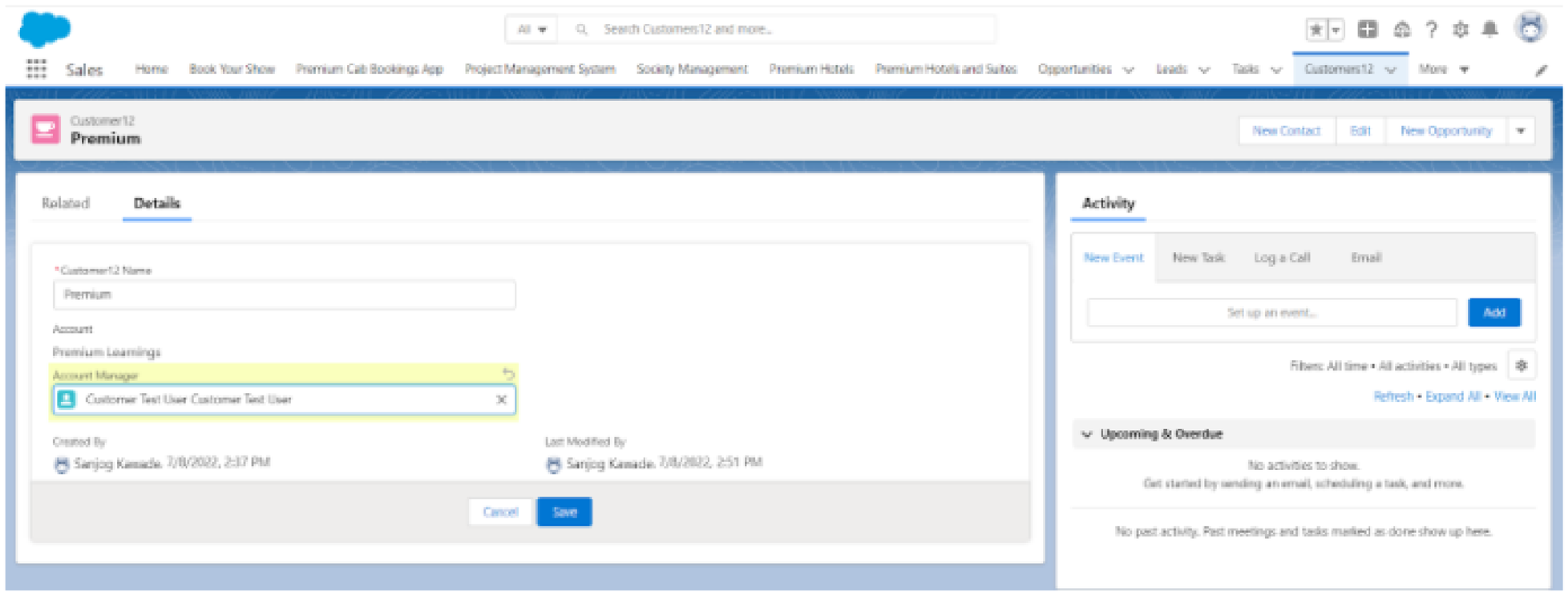
3. Now Update the account manager field and select any other user instead of the current one. Click Save.

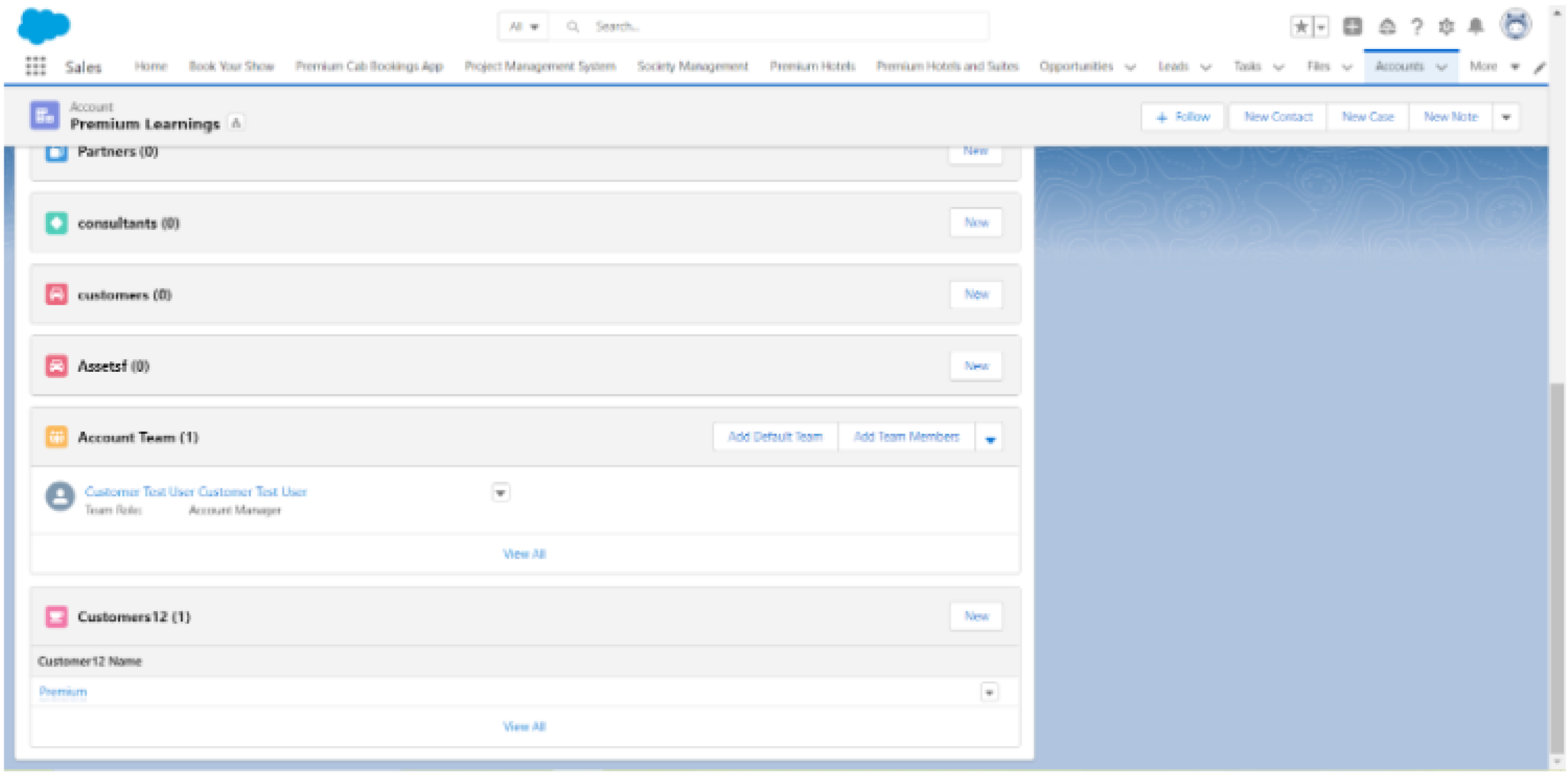


4. Now check related list of account we created in step one, and find account team in it. We will see the account manager(premium user) we updated above in it.



5. Repeat steps 3 and 4 again with a different user as account manager and see the account team field is updated as well.





Example 26

The trigger scenario 35 Logic is when we create a Customer12 record for the account record, then the user in the Account Manager field will be automatically added to the Account Team of that associated account. Now the following trigger gives the logic about when we delete the “Customer12” of that account, then the user will be deleted automatically from the Account Team of that account.

pre-requisite:

From Setup, enter Account Teams in the Quick Find box, and select Account Teams. Click Enable Account Teams. Select the Account Teams Enabled checkbox and click Save. Select the Account Layout checkbox to add the Account Team related list to the page layout. Select the Add to users' customized related lists checkbox and click Save.

```
trigger DeleteAccountTeam on Customer12__c (before delete)
{

List<AccountTeamMember> atm_list=new
List<AccountTeamMember>();
AccountTeamMember atm = new AccountTeamMember();
List<AccountShare> newShare = new List<AccountShare>();

 if(trigger.isdelete)
 {
  set<id> setAccids = new set<id>();
  Set<Id> setDelATM=new Set<Id>();
  Map<id,Set<Id>> mapAccMgrs=new Map<id,Set<Id>>();
 for(Customer12__c c:Trigger.old)
{
 setAccids.add(c.Account__c);
}
 List<Customer12__c> listPLAccMgrs=[select id,
        AccountManager__c,Account__c from Customer12__c
        where Account__c in:setAccids and id not
        in:trigger.oldmap.keyset()];
```

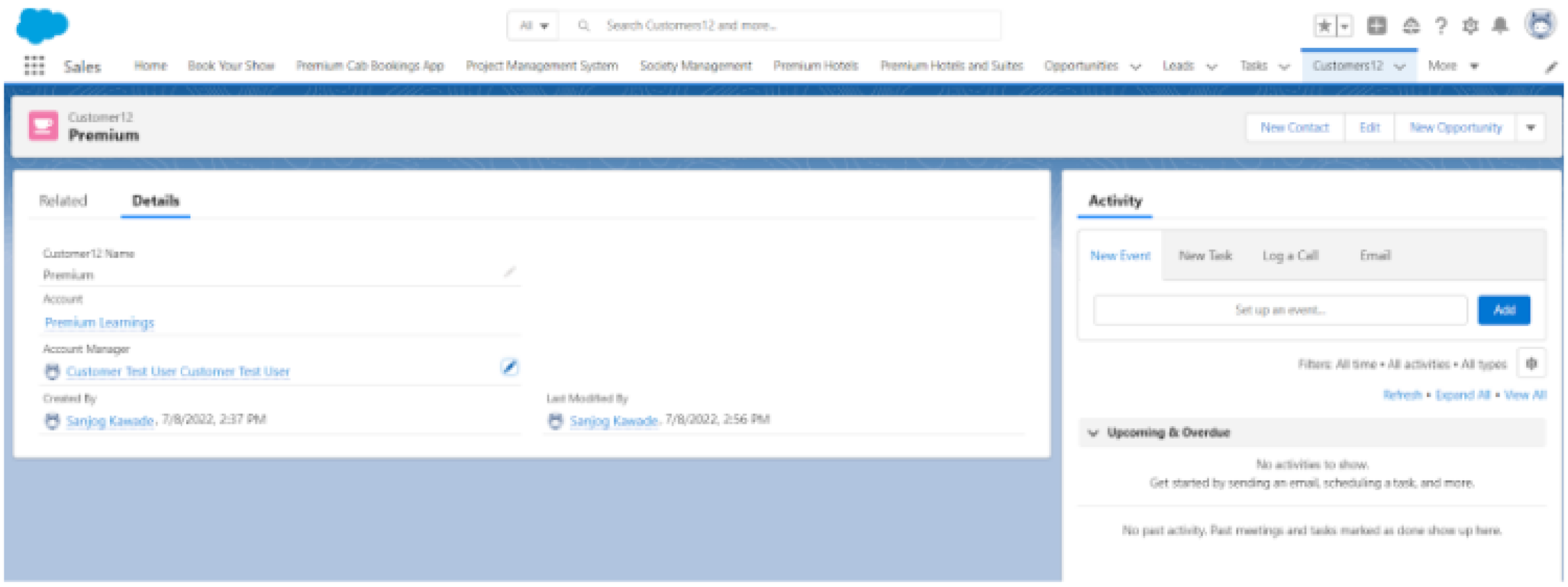
```
 if(listPLAccMgrs!=null && listPLAccMgrs.size()>0)
 {
    for(Customer12__c c:listPLAccMgrs)
    {
    Set<Id> idMgrs=mapAccMgrs.get(c.Account__c);
    if(null==idMgrs){
      idMgrs=new set<Id>();
      mapAccMgrs.put(c.Account__c,idMgrs);
    }
    idMgrs.add(c.AccountManager__c);
    }
  }

Map<id,List<AccountTeamMember>> mapStdAccTeam=new
                        Map<id,List<AccountTeamMember>>();
List<AccountTeamMember> listStdAccTeam=[select id,
                  UserId,AccountId from AccountTeamMember
                  where AccountId in:setAccids];

if(listStdAccTeam!=null && listStdAccTeam.size()>0)
{
  for(AccountTeamMember recAccTeam :listStdAccTeam)
  {
    List<AccountTeamMember>
    listStdAccTeamMap =
                  mapStdAccTeam.get(recAccTeam.AccountId);
```

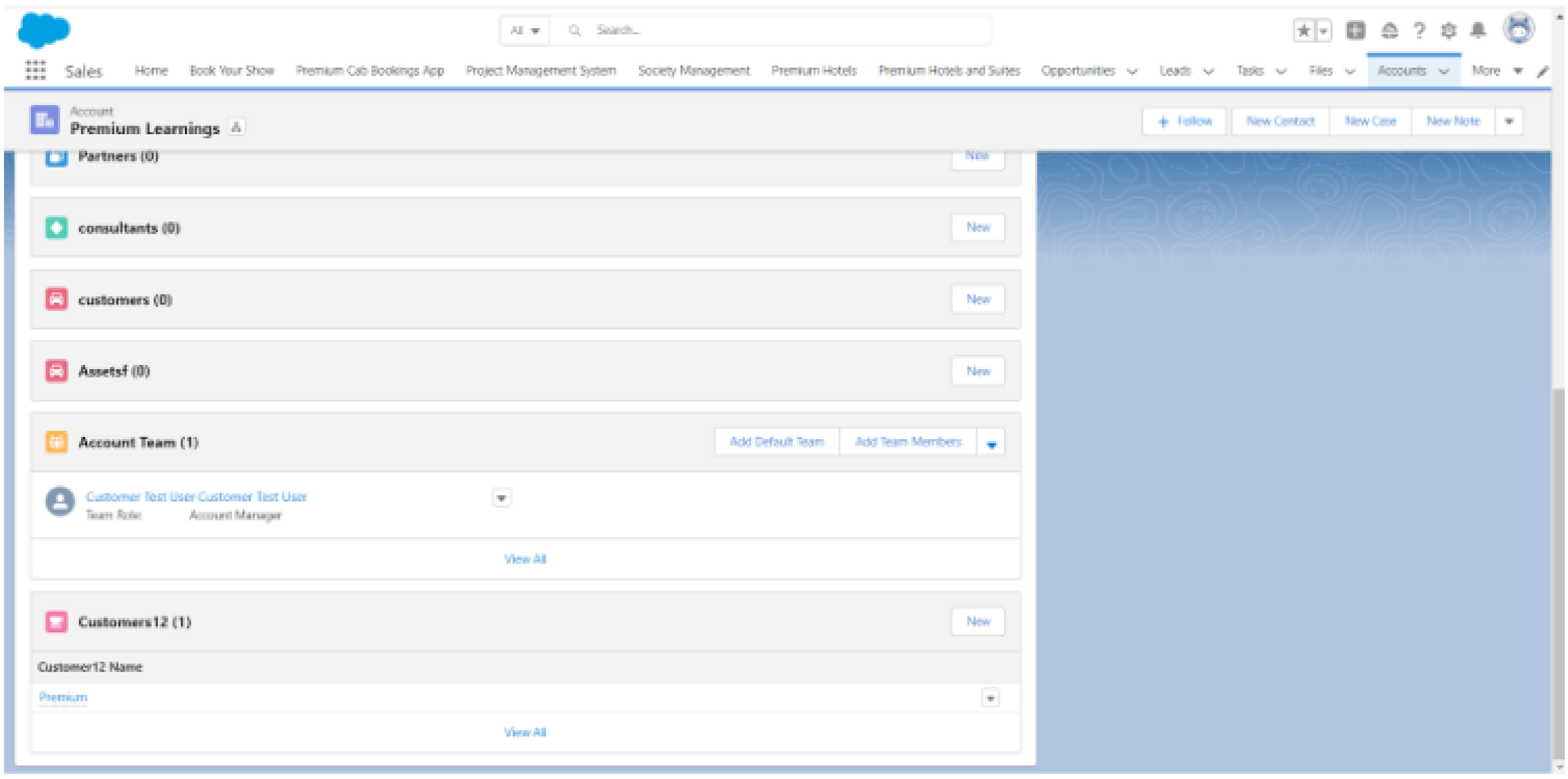
```
        if(null==listStdAccTeamMap)
        {
            listStdAccTeamMap=new List<AccountTeamMember>();
            mapStdAccTeam.put(recAccTeam.AccountId.
                                        listStdAccTeamMap);
        }
        listStdAccTeamMap.add(recAccTeam);
    }
}
   for(Customer12__c c:Trigger.old)
   {
     List<AccountTeamMember>
     listAccTeam=mapStdAccTeam.get(c.Account__c);
     Set<Id> idMgrs=mapAccMgrs.get(c.Account__c);

     if(listAccTeam!=null && listAccTeam.size()>0 )
     {
      if(idMgrs!=null && idMgrs.size()>0 &&
              !(idMgrs.Contains(trigger.oldmap.get(c.Id)
                                  .AccountManager__c)))
      {
        for(AccountTeamMember recATM:listAccTeam)
        {
         if(recATM.UserId ==
              trigger.oldmap.get(c.Id).AccountManager__c)
           setDelATM.add(recATM.Id);
        }
      }
   }
```

```
        else if(idMgrs==null)
        {
          for(AccountTeamMember recATM:listAccTeam)
          setDelATM.add(recATM.Id);
        }
    }
     List<AccountTeamMember> listDelATM=[select id from
                AccountTeamMember where id in:setDelATM];

   if(listDelATM!=null && listDelATM.size()>0 )
   delete listDelATM;
   }
}
```

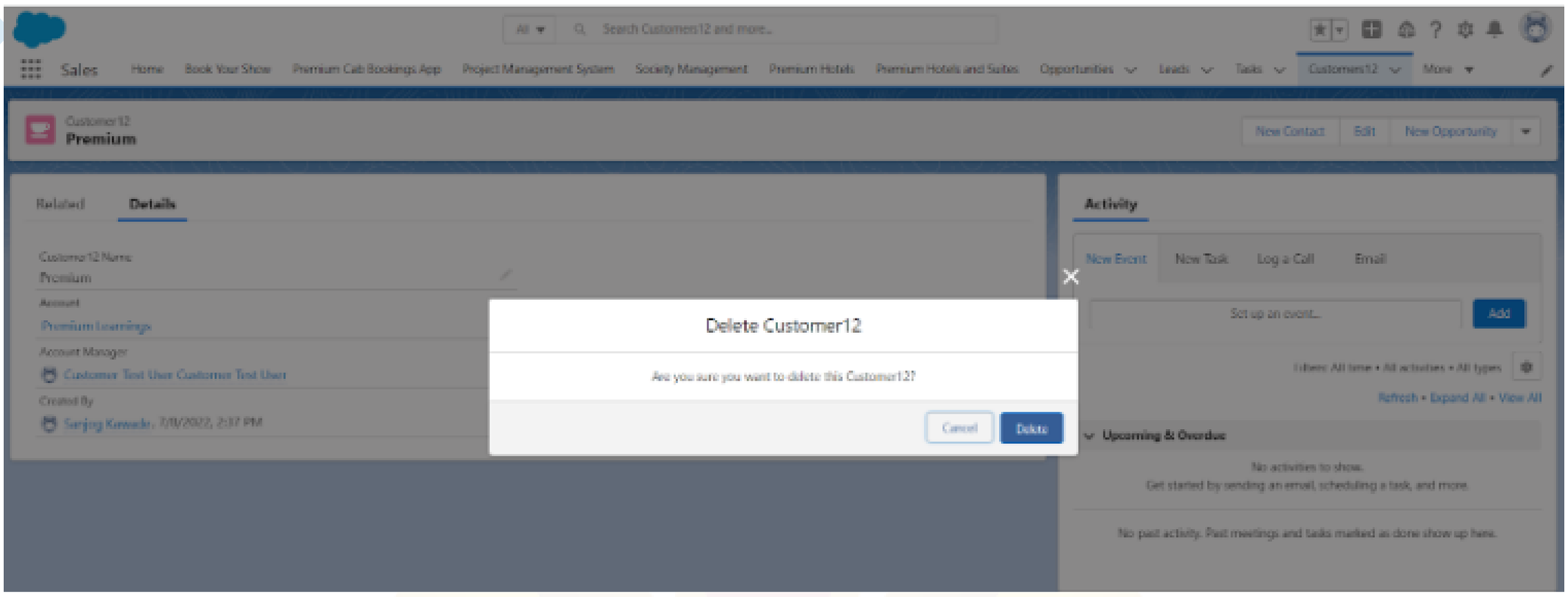

Test Case:

1. In the previous examples we created a customer12 record named ‘premium’ with account manager field populated with a user. The same user will be populated in the account team (in related list) of the lookup account of customer12.

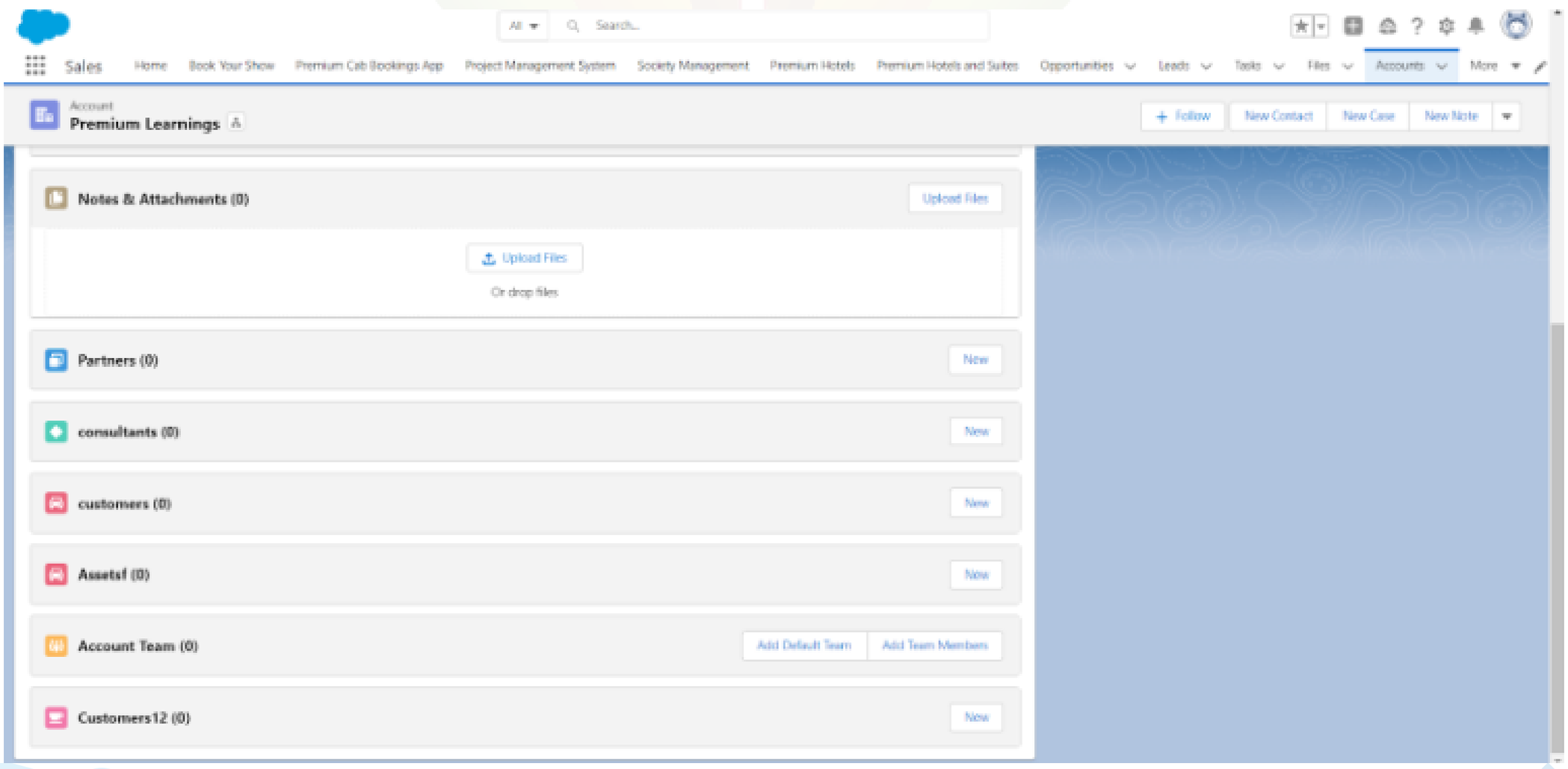




2. Now delete the customer12 record named ‘Premium’ we created



3. The account team field of the account will now be blank.



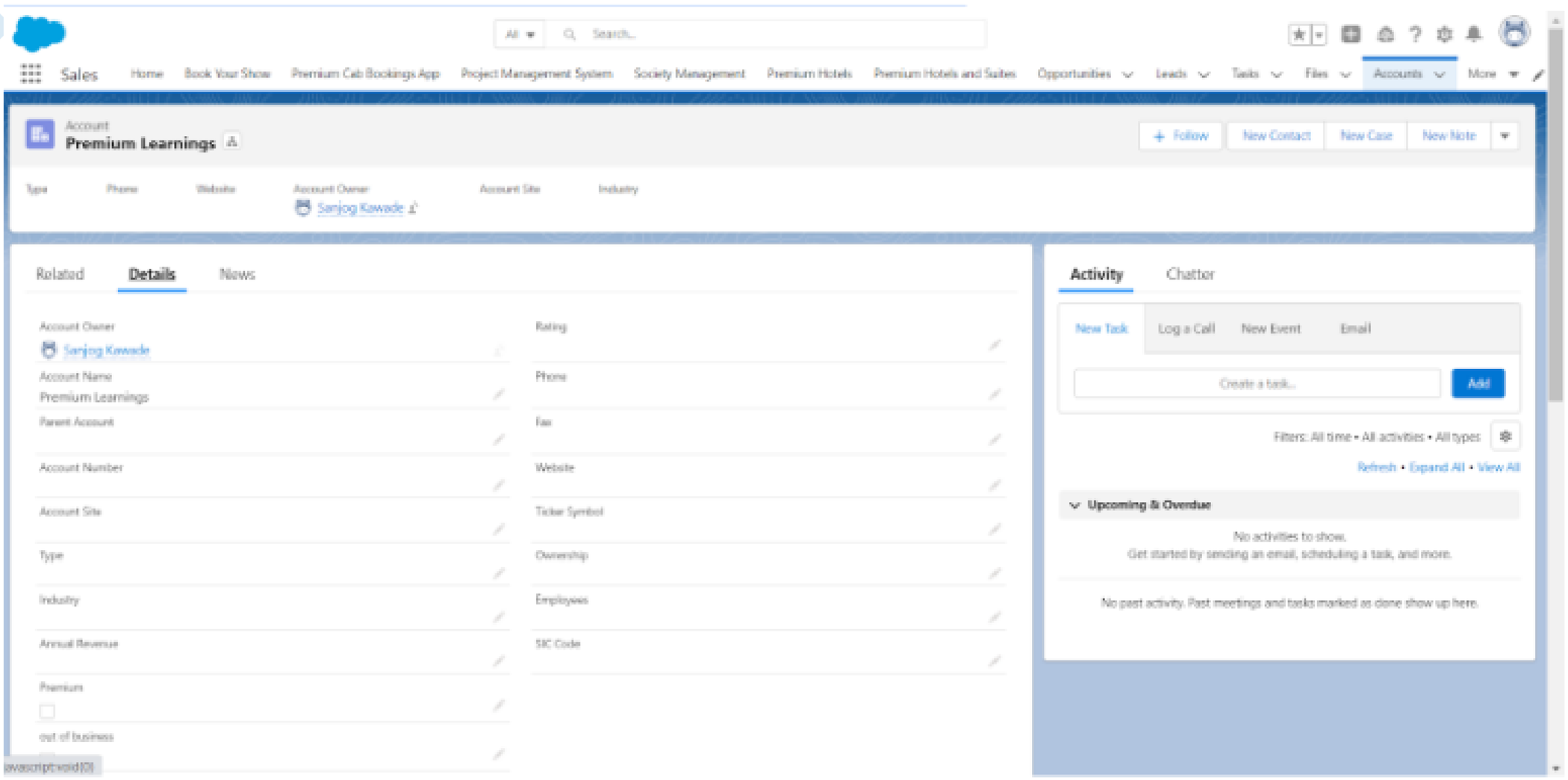
Example 27

When we create the Opportunity with the Probability=20, then the opportunity owner will be automatically added to the Account Team of the associated account for that Opportunity.

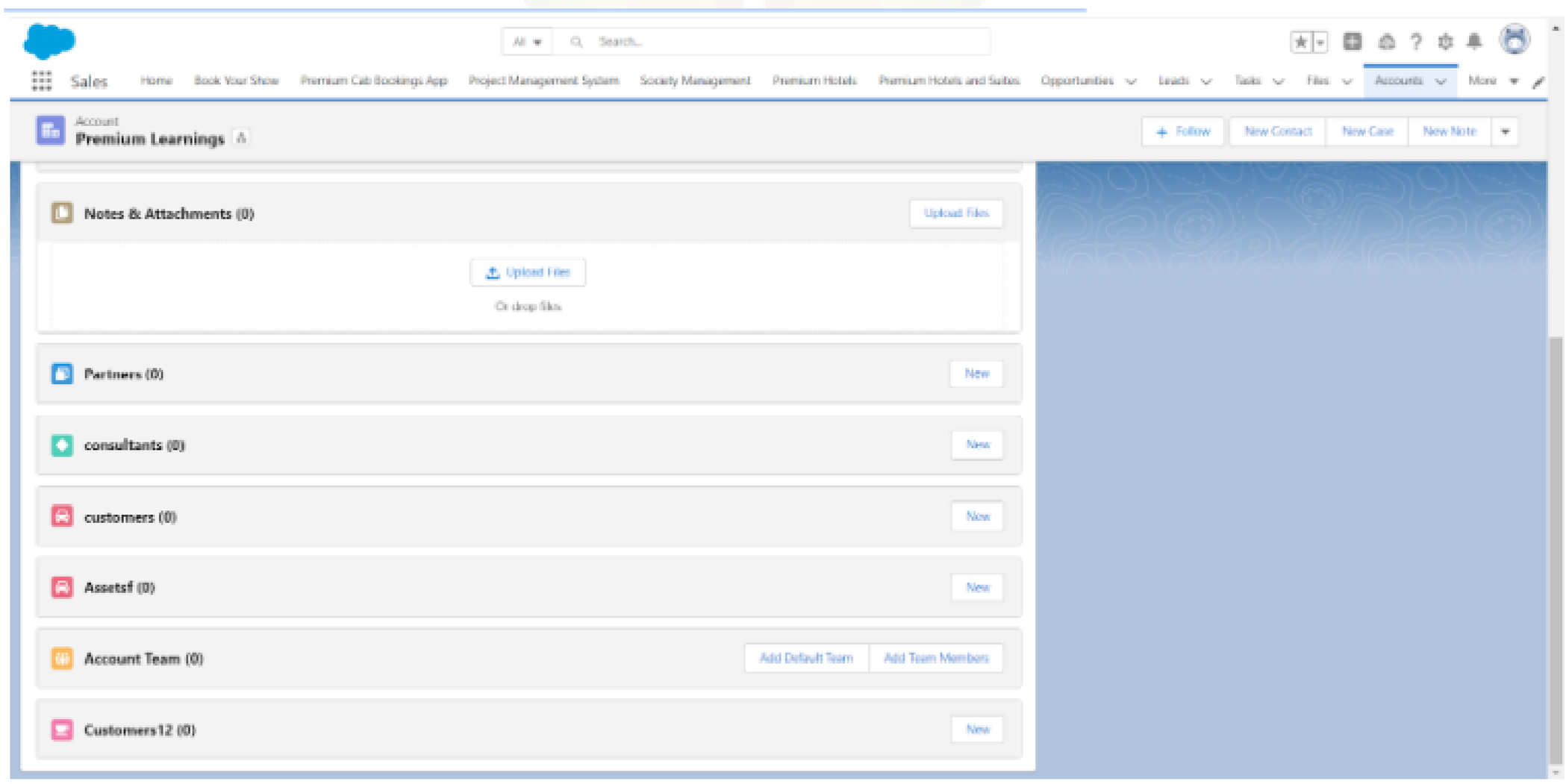
```
trigger UpdateATMwithOwneronOptyCreate on Opportunity
(after insert,after update) {

 List<AccountShare> list_share= new List<AccountShare>();
List<AccountTeamMember> list_atm=new
                                    List<AccountTeamMember>();
 for(Opportunity opp:Trigger.New)
 {
    if(opp.Probability==20)
    {
     AccountTeamMember atm=new AccountTeamMember();
     atm.accountid=opp.accountid;
     atm.teamMemberRole='Account Manager';
     atm.UserId=opp.Ownerid;
     AccountShare share = new AccountShare();
     share.AccountId=opp.Accountid;
     share.UserOrGroupId = opp.OwnerId;
     share.AccountAccessLevel='Read/Write';
     share.OpportunityAccessLevel = 'Read Only';
     share.CaseAccessLevel='Read Only';
    list_atm.add(atm);
    list_share.add(share);
    }
}
   if(list_atm!=null)
      insert list_atm;

   if(list_share!=null && list_share.size()>0)
      List<Database.saveresult> sr=
                  Database.insert(list_share,false);
}
```

Test Case:

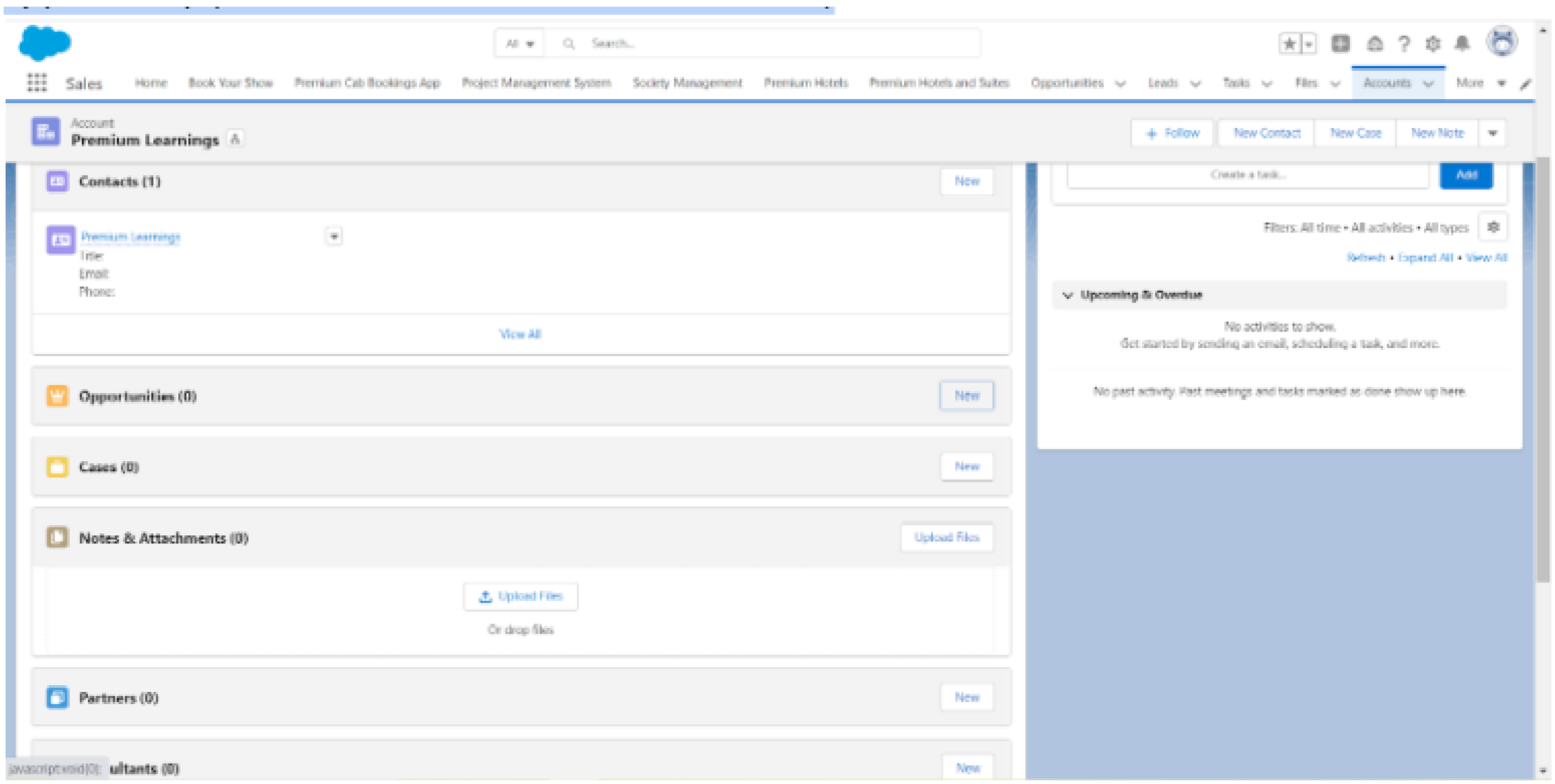
1. Create an account named premium learnings and save it



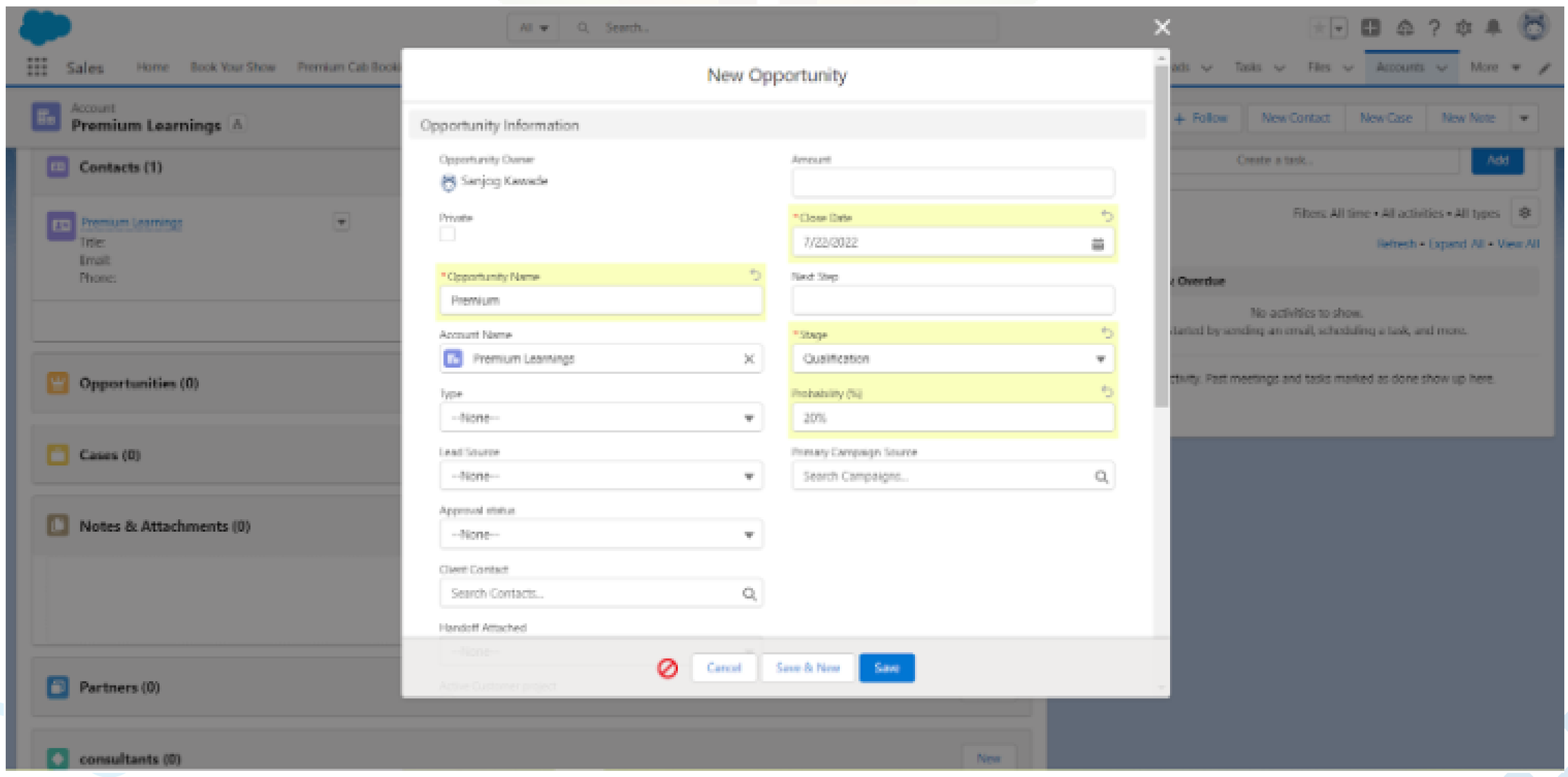
2. Now go to the related lists and check the account team is empty.

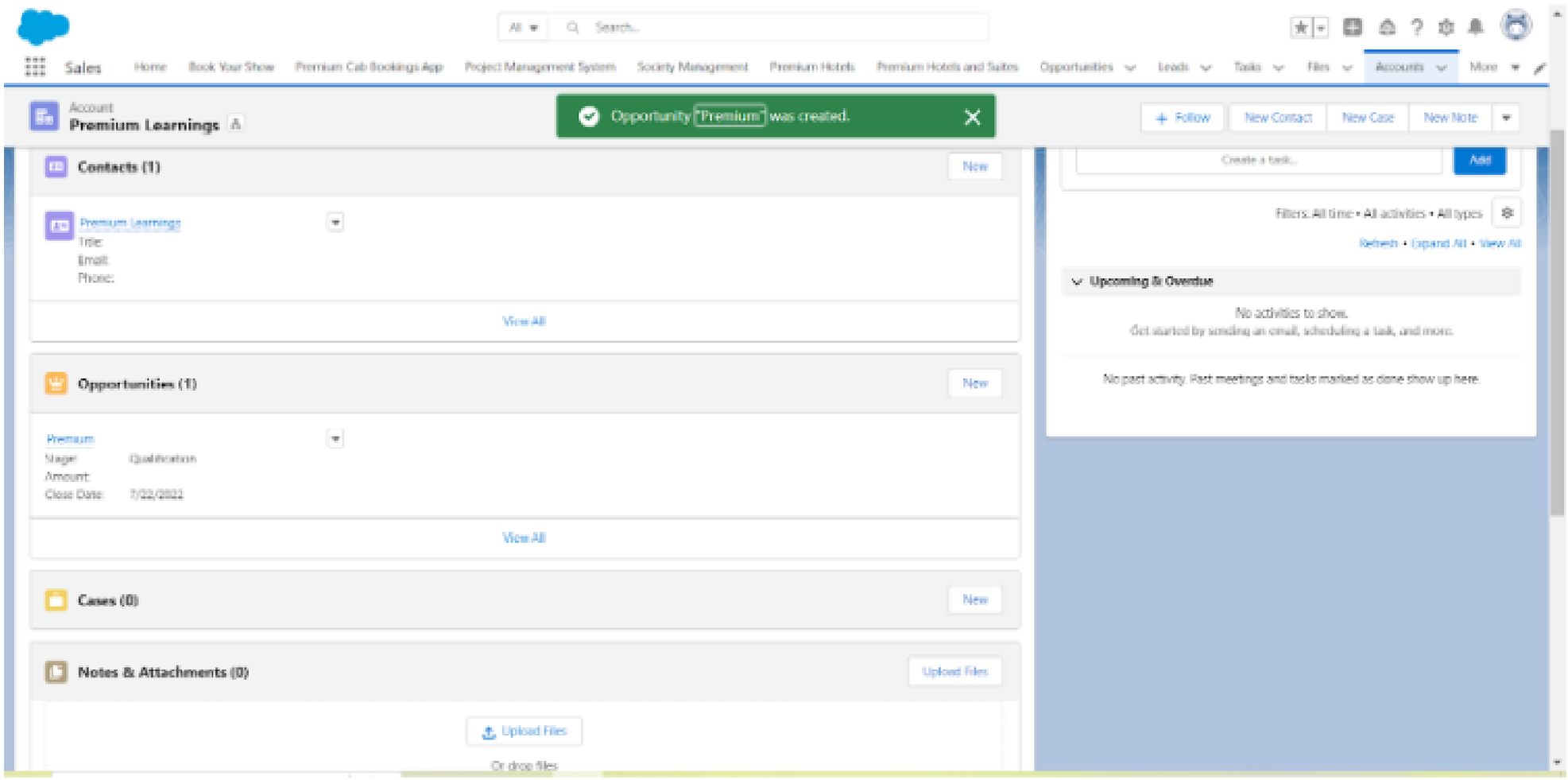


3. Now go to the related list of the same account, find opportunities, and create a new opportunity (which will be related to this account)

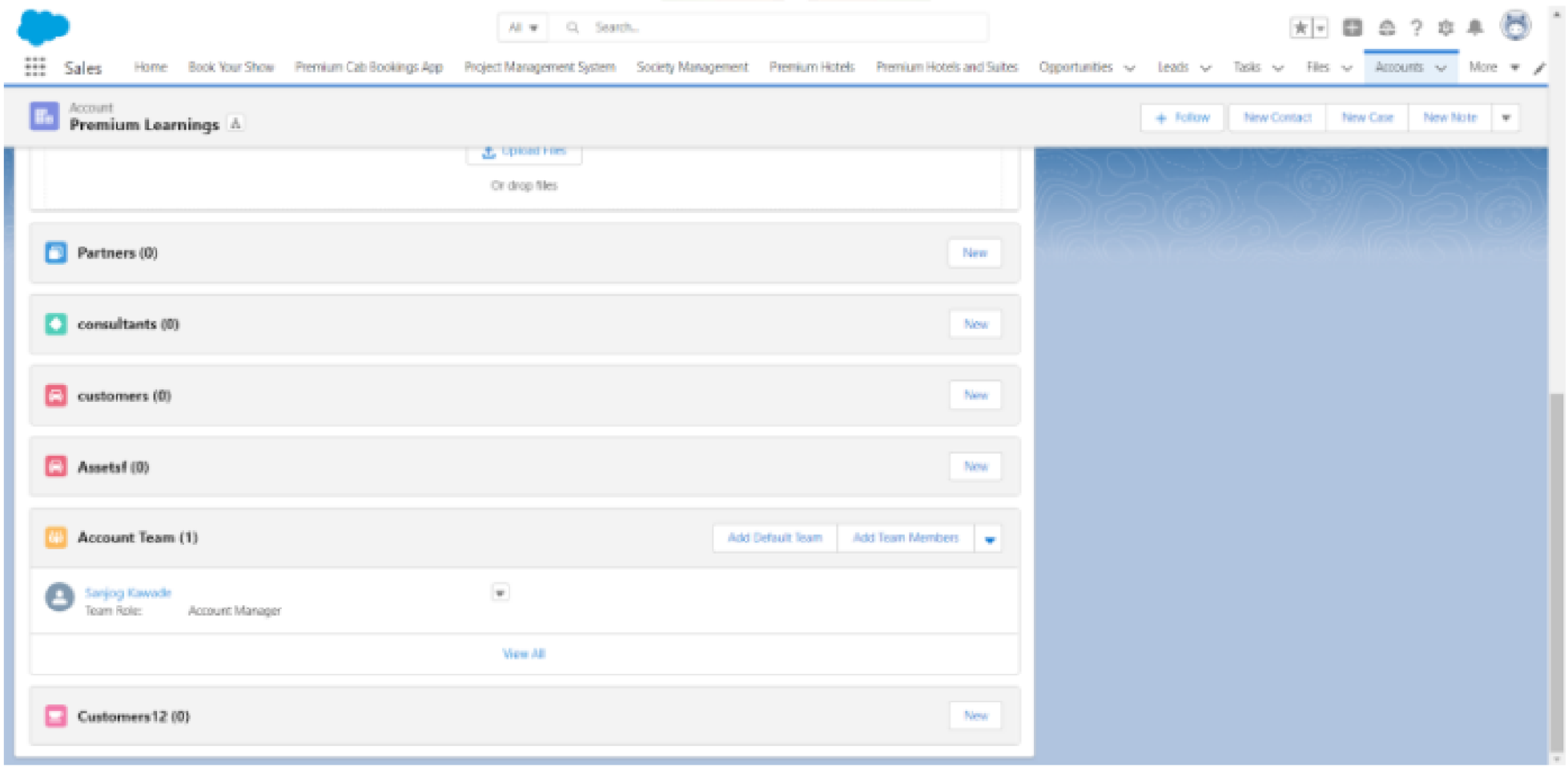


Make sure the probability is 20. Click save





4. Now refresh the related lists page of the account and check account team in related lists. It will be populated with opportunity owner of the opportunity record we created



Eample 28

Trigger to update parent records field when child record is updated | update parent field from child using apex trigger

CheckboxStatus

Create two custom objects, both are associated with a lookup relationship with each other

1) Custom Object:- **Consultant__c** custom field:- Status__c (Checkbox boolean type)
2) Custom Object:- **Leave_Reques__c** custom field:- Status__c (Checkbox boolean type)

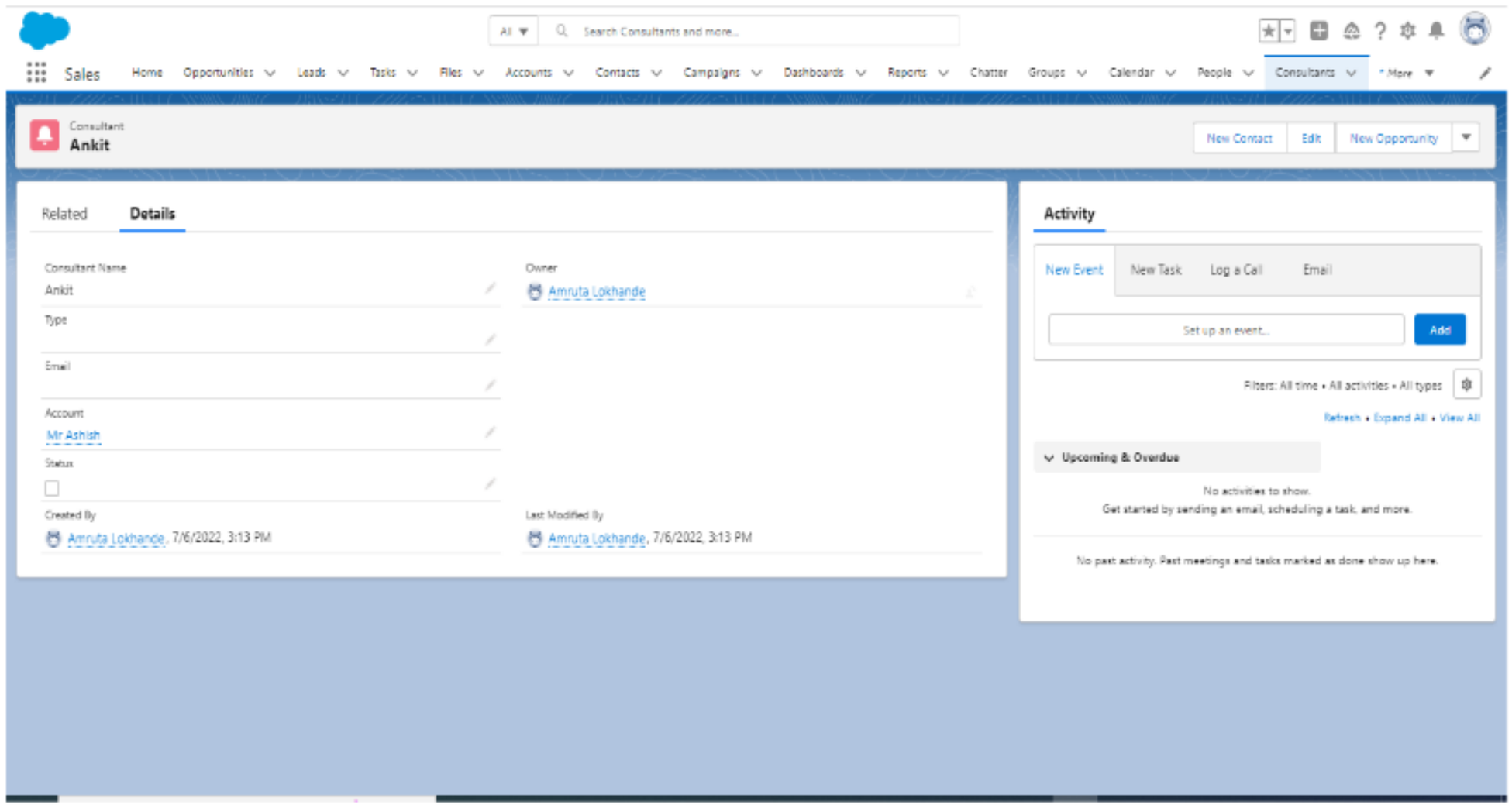
If the status field of child object marked true then should be status field of parent object automatic
marked as true.

```
trigger CheckboxStatus on Leave_Request__c(before Insert,
before update, after insert, after
update, before delete, after delete)
{
 if(trigger.isBefore){
 //system.debug('I am inside before event');
 }
 else if(trigger.isAfter){
 //system.debug('I am inside after event');

   if(trigger.isUpdate){
     for(Leave_Request__c myStatus: Trigger.new){
        Consultant__c getParentObj = [SELECT Id,
                Status__c FROM Consultant__c WHERE Id =
                :myStatus.Consultant__c];
      Leave_Request__c getChildObj = [SELECT Id,
             Status__c, Consultant__r.Status__c FROM
             Leave_Request__c WHERE Id = :myStatus.Id];
      getParentObj.Status__c= getChildObj.Status__c;
      insert getParentObj;
    }
}
```

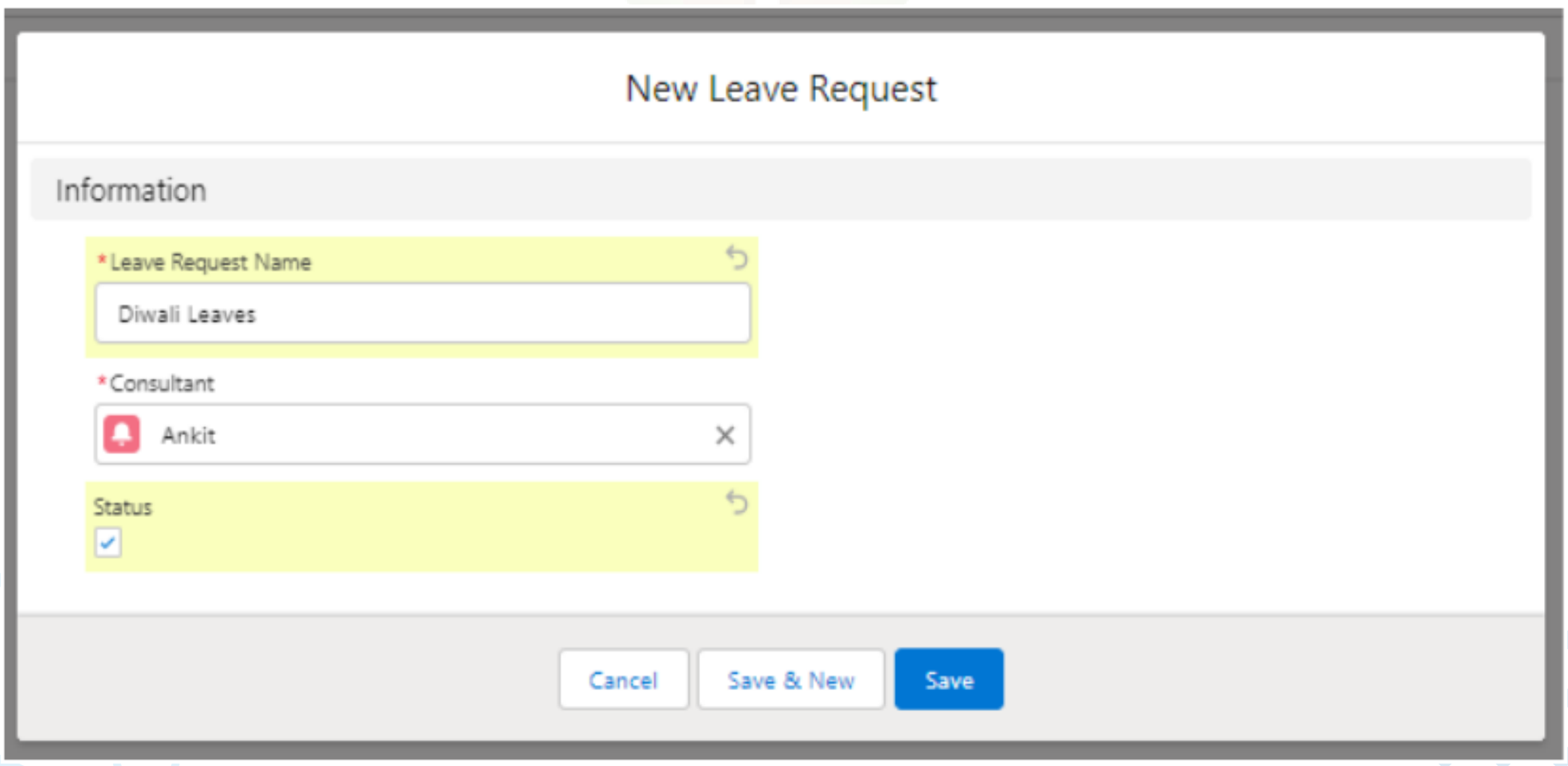
```
if(trigger.isInsert && (trigger.isAfter)){
  for(Leave_Request__c myStatus: Trigger.new){
      Consultant__c getParentObj = [SELECT Id,
            Status__c FROM Consultant__c WHERE Id =
            :myStatus.Consultant__c ];
     Leave_Request__c getChildObj = [SELECT Id,
           Status__c, Consultant__r.Status__c FROM
           Leave_Request__c WHERE Id = :myStatus.Id];
     getParentObj.Status__c= getChildObj.Status__c;
     update getParentObj;
  }
}


if(Trigger.isDelete && (Trigger.isBefore)){
  for(Leave_Request__c myStatus: Trigger.old){
   Consultant__c getParentObj = [SELECT Id, Status__c
                        FROM Consultant__c WHERE Id =
                        :myStatus.Consultant__c ];
   Leave_Request__c getChildObj = [SELECT Id,
             Status__c, Consultant__r.Status__c FROM
             Leave_Request__c WHERE Id = :myStatus.Id];
  delete getParentObj;
  }
 }
}
}
```

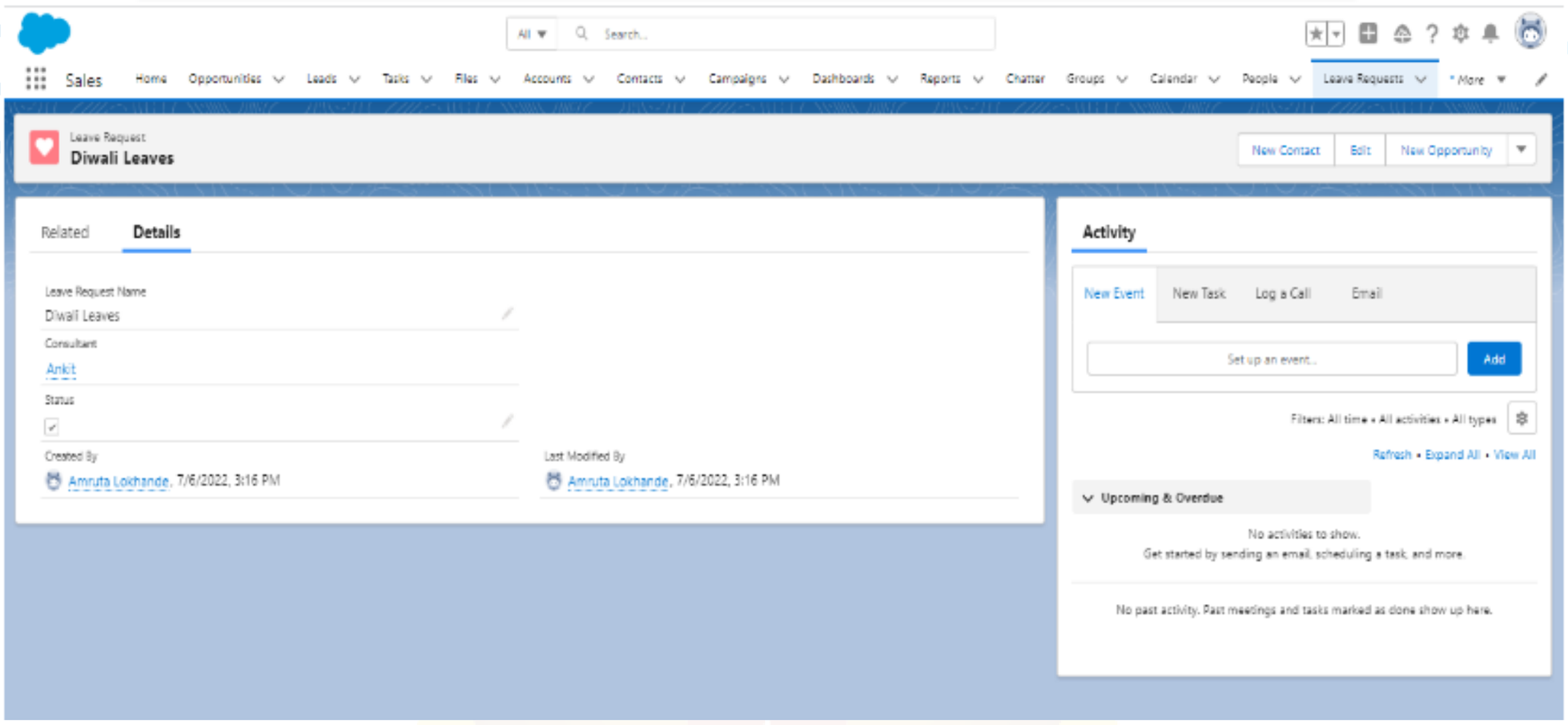
Test Case:

1. Create new record of Consultant and (Keep the Status checkbox unchecked).

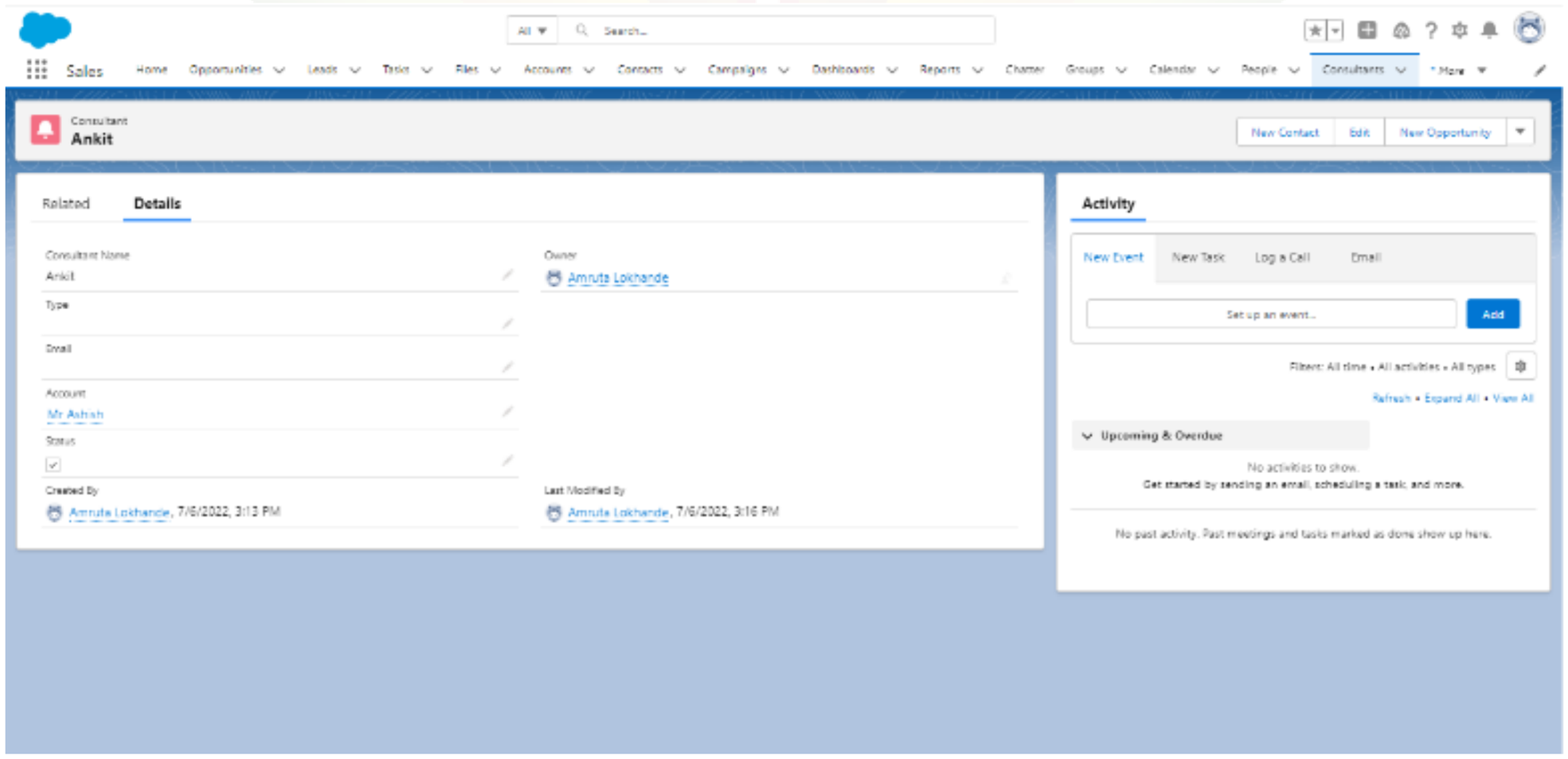


2. Create Child Leave_Request for the same consultant (check the Status checkbox).





3. Check the Status checkbox of parent consultant will automatically become checked.



Example 29

Write a trigger on Account to Prevent the user to create duplicate Account based on Phone if Phone number is already exist in Salesforce

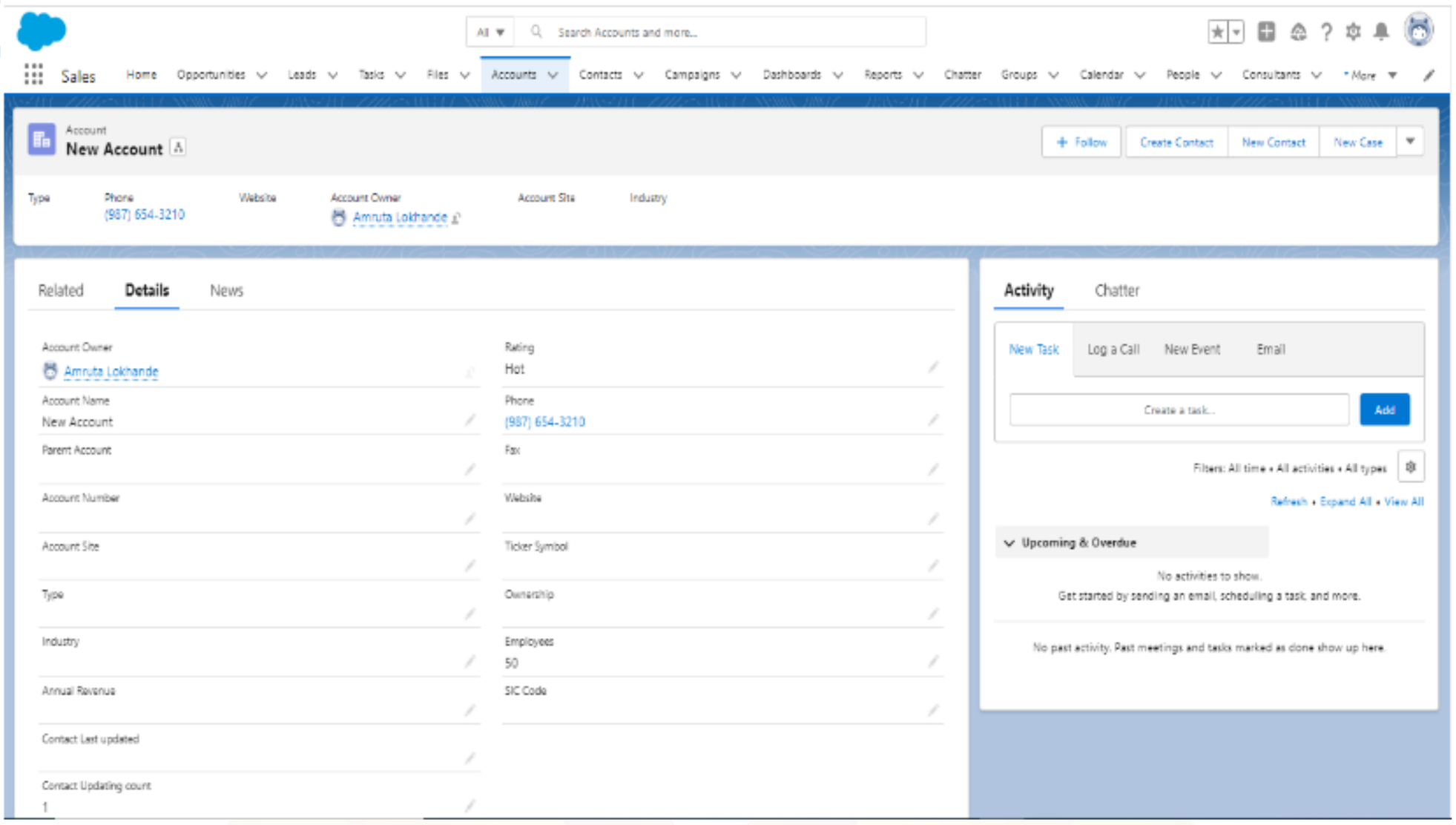
Trigger :

checkDuplicationPhoneAccTrigger.apxt

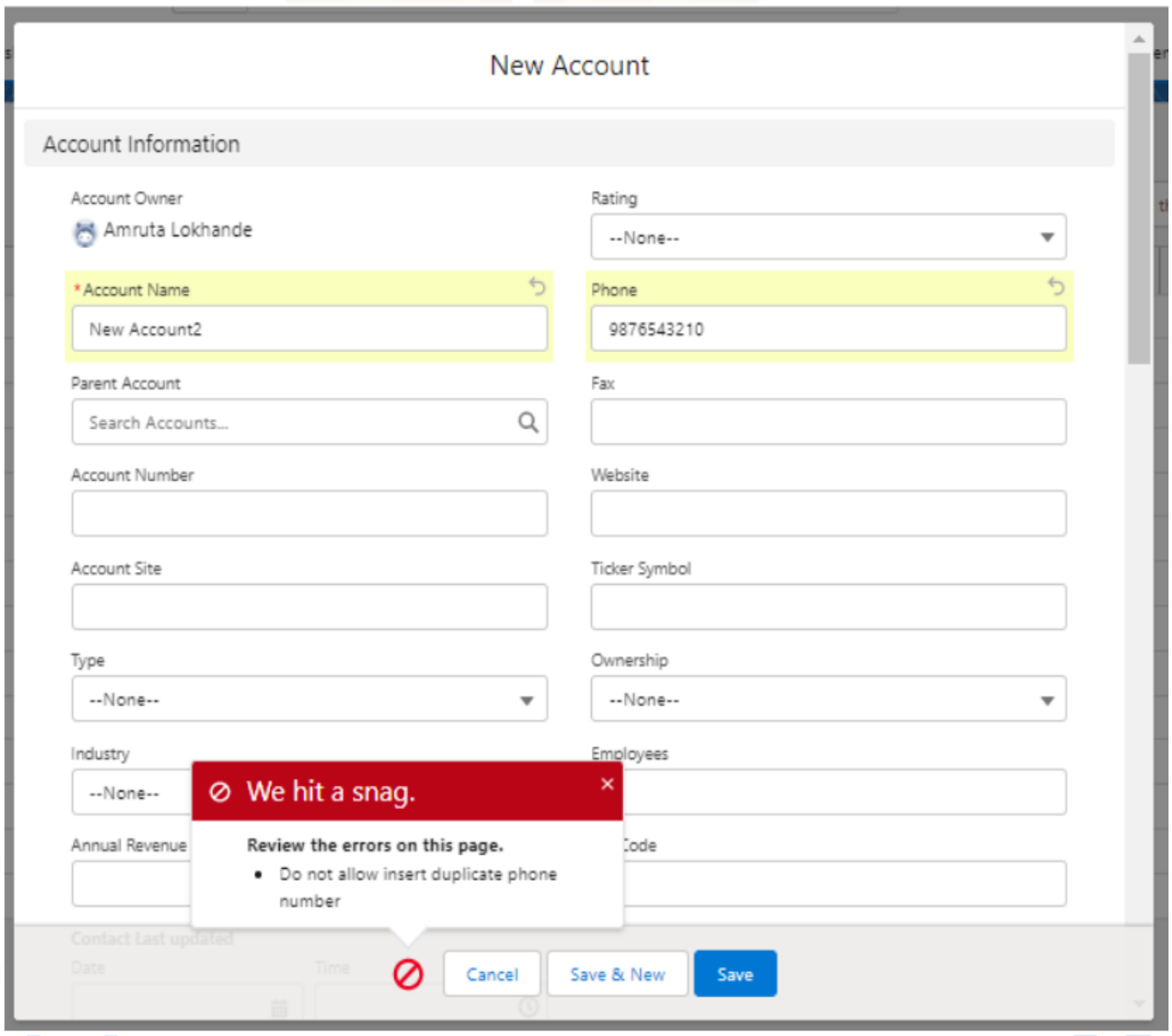
```
trigger checkDuplicationPhoneAccTrigger on Account
(before insert,after insert)
{
 if(trigger.isBefore){
    system.debug('trigger before trigger');

    if(trigger.isInsert){
      set<String> accPhone=new set<String>();
      for(Account a:Trigger.new){
          accPhone.add(a.Phone);
          System.debug('Account phone number is '
                  + accPhone);
    }
    set<String> phoneAccData=new set<String>();
    for(Account a:[select Id, Name, Phone from Account
                  where Phone in:accPhone])
    {
        phoneAccData.add(a.Phone);
    }
    for(Account a:Trigger.new){
        if(phoneAccData.contains(a.Phone)){
           a.addError('Do not allow insert duplicate
                  phone number');
            }
          }
        }
    }
    else if(trigger.isAfter){
          system.debug('trigger after trigger');
    }
  }
```

Test Case:

1. Create new Account record .



2. Try to create another Account record with same Phone. And see it will display an error message as shown in below screenshot.



Example 30

Trigger to check duplicate name to custom object in Salesforce | how to prevent duplicate records based on multiple fields through apex trigger in Salesforce

Trigger : childObjTriggerHandler.apxt

Create Custom Objects.
1.Parent object: Project__c
2.Child Object: Employee__c, Field: Employee_Name__c
Create look relationship between them.

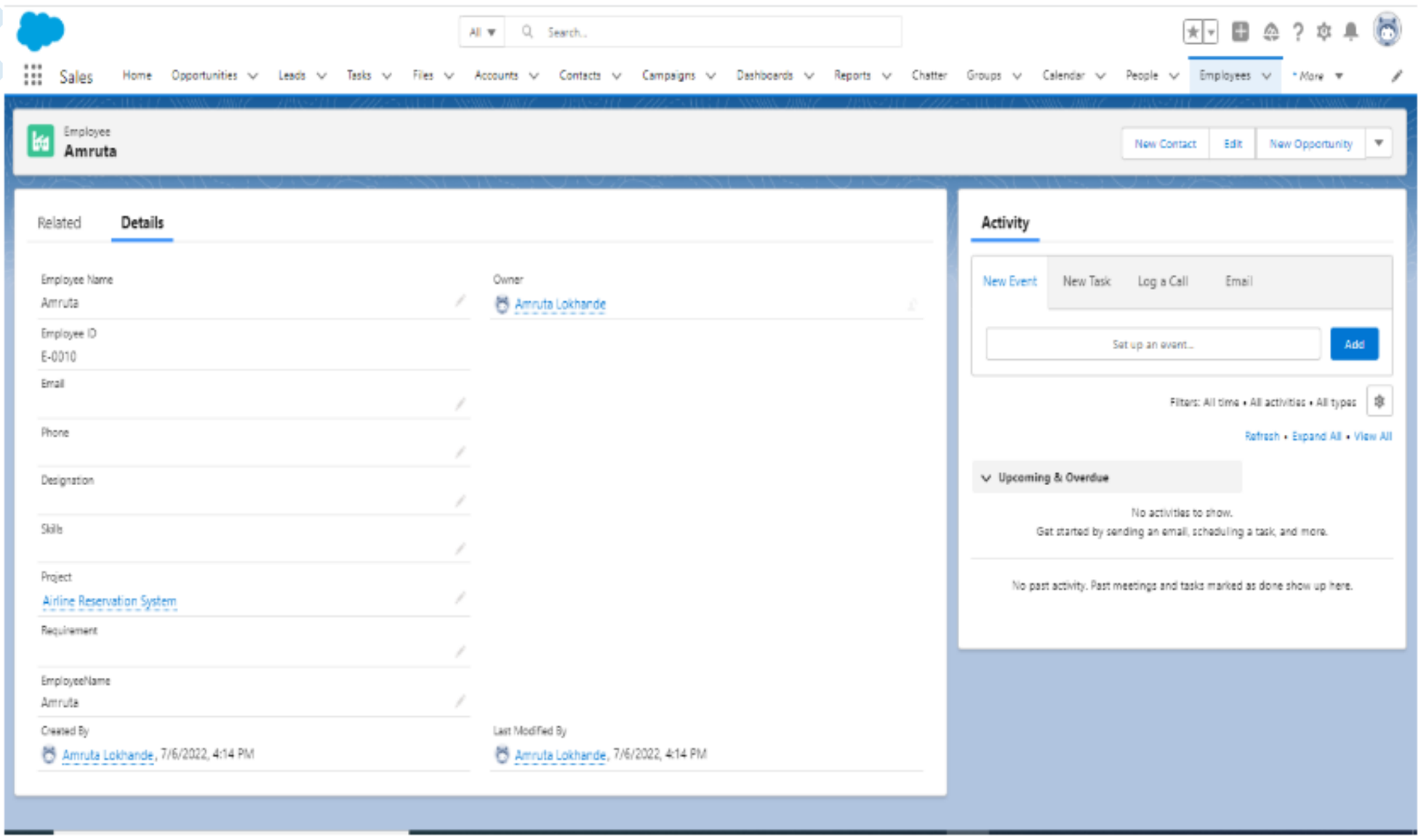
```
trigger childObjTriggerHandler on Employee__c
(before insert, after insert)
{

 if(trigger.isBefore){
  if(trigger.isInsert){
        system.debug('I am inside before insert');
        childObjHandler.childDuplicate(trigger.new);
  }
 }
  else if(trigger.isAfter){
        if(trigger.isInsert){
   //system.debug('I am inside after insert');
         system.debug('Record inserted successfully');
        }
  }
}
```

Class: :ChildObjHandler

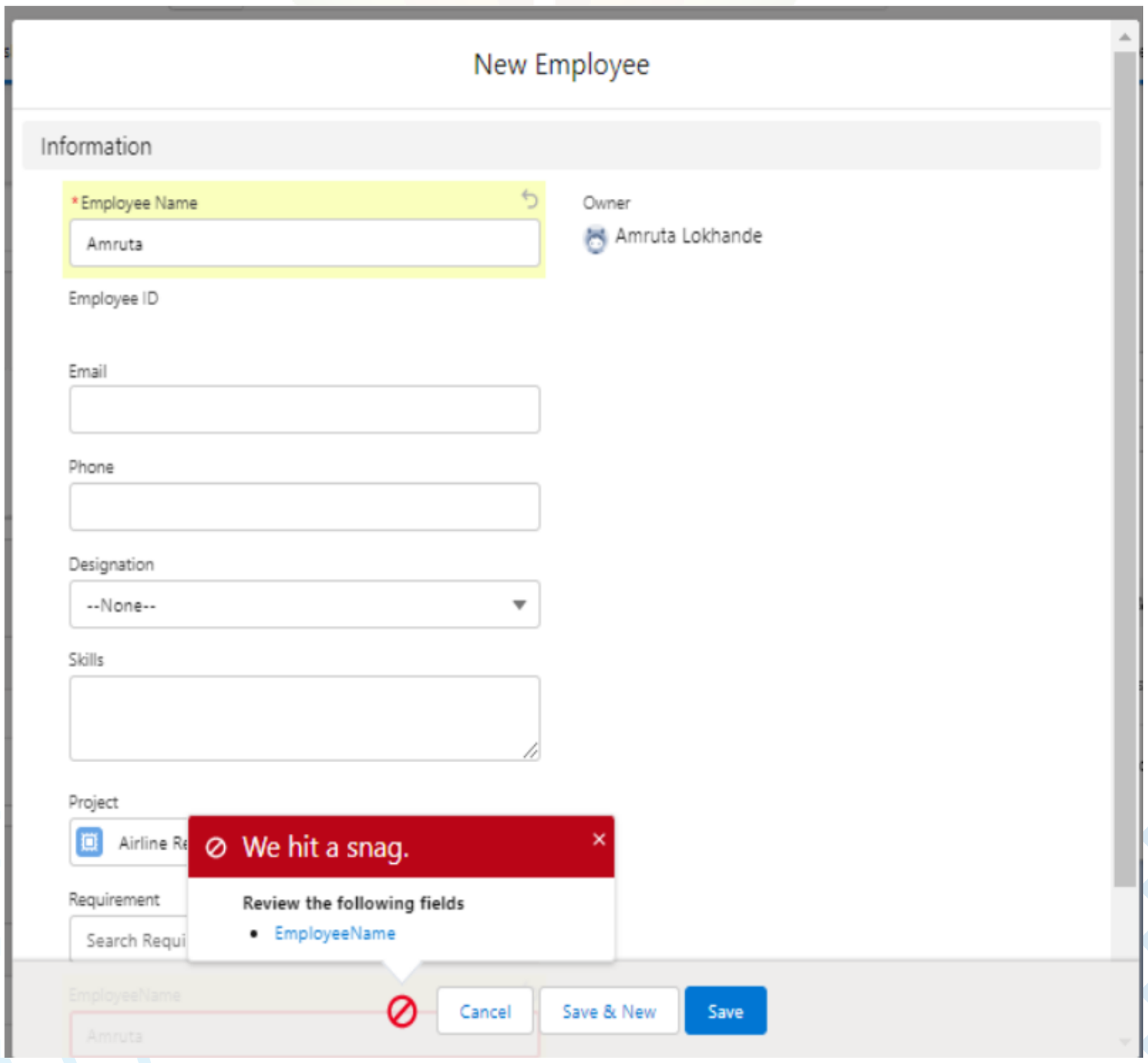
```
public class ChildObjHandler {

    public static void childDuplicate
                                (List<Employee__c> dupChildRec)
{
    Set<String> empExtStr = new Set<String>();
    List<Employee__c> empList = [Select Id, Name,
                  EmployeeName__c, Project__c From
                  Employee__c where EmployeeName__c !=null];

    for(Employee__c empStr:empList){
        empExtStr.add(empStr.EmployeeName__c)
    }

        for(Employee__c childTr:dupChildRec){
            if(empExtStr.contains(childTr.EmployeeName__c))
            {
             childTr.EmployeeName__c.addError
                                ('Do not allow duplicate');
            }
        }
  }
}
```

Test Case:

1. Create Child Employee record .



2. Try to create new Employee record with same Name it will give you an error as shown in below screenshot.



Example 31

Create rollup summary using Apex trigger on custom

Trigger: childObjTriggerRollUp

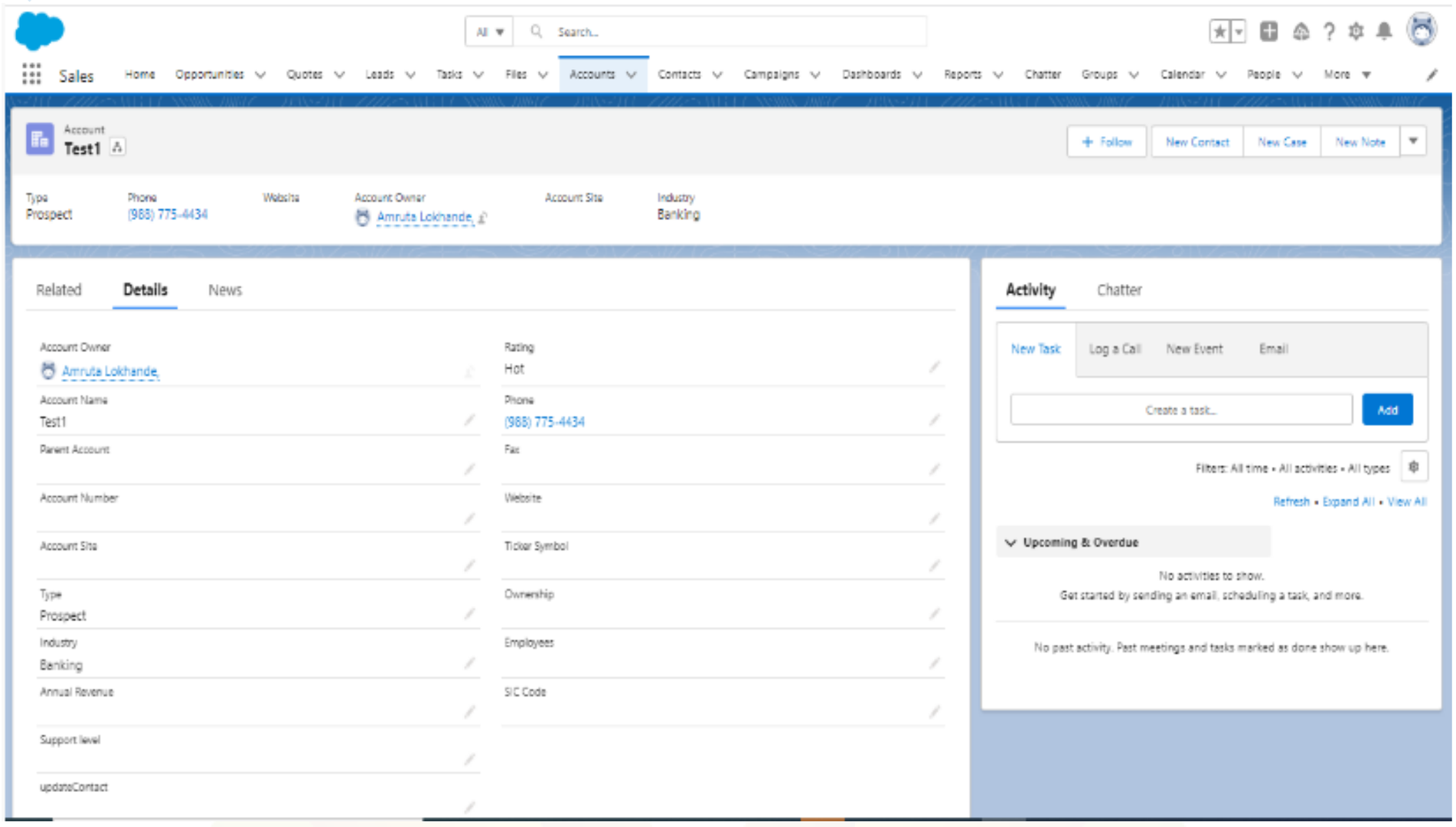
Create a custom field on parent Object name as childRollUp__c (Number Type). Create a lookup relationship field on parent Object

```
trigger childObjTriggerRollUp on Contact (before insert,
before update, before delete, after insert, after update,
after delete, after undelete) {

 List<Contact> childObjList = new List<Contact>();
 Set<Id> accSetId = new Set<Id>();
 if(trigger.isBefore){
    system.debug('trigger before#');
 }
 else if(trigger.isAfter){
   system.debug('trigger after#');
   if(trigger.isDelete){
     childObjList = trigger.old;
   }
 else{
     childObjList = trigger.new;
 }
 List<Account> updateParent = new List<Account>();

 for(Contact childObjNewList:childObjList){
   if(childObjNewList.AccountId != null){
     accSetId.add(childObjNewList.AccountId);
    if(trigger.isUpdate){
       Contact childObjOld =
                 trigger.oldMap.get(childObjNewList.Id);
```

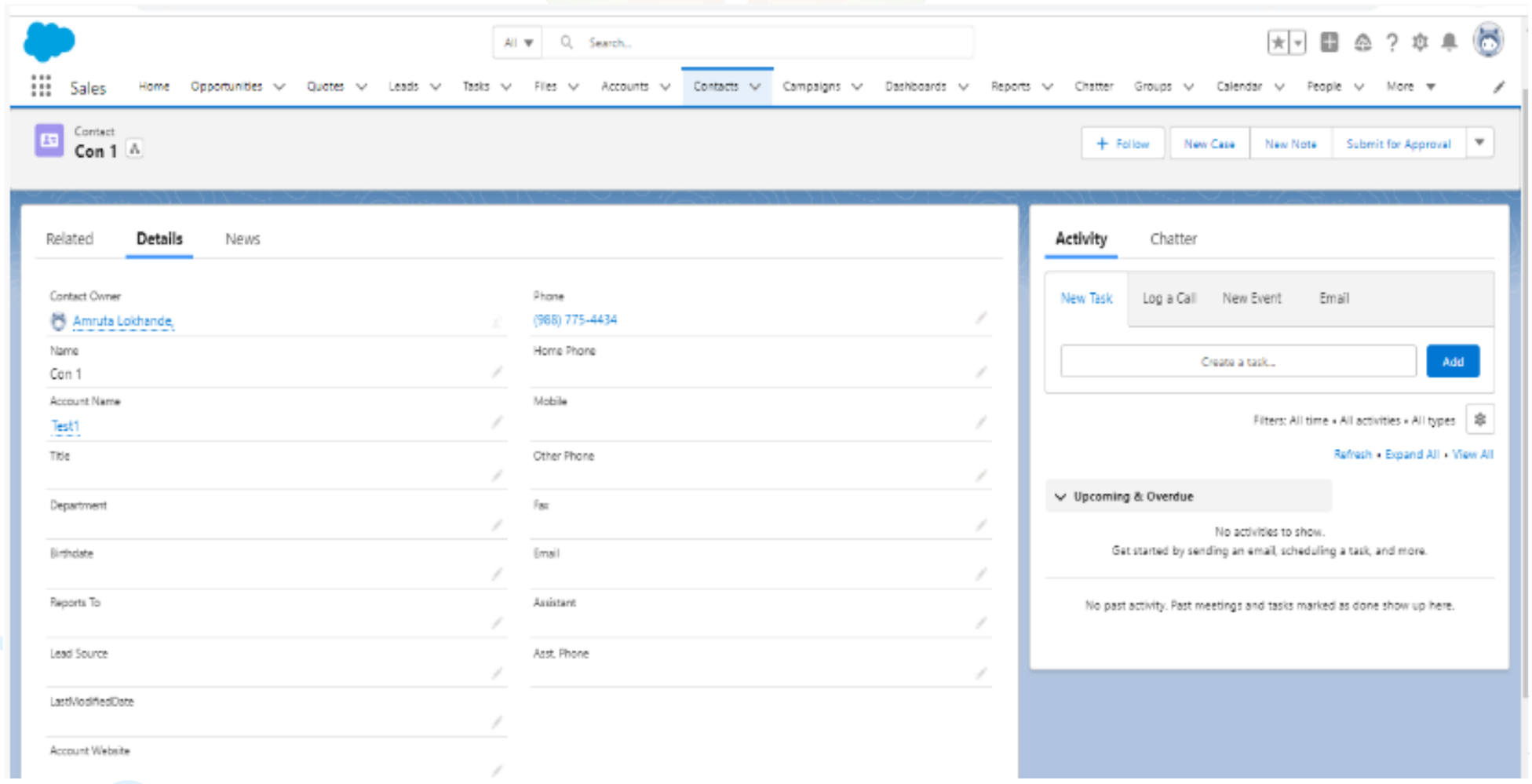
```
        system.debug('childObjOld ' + childObjOld);
        if(childObjOld.AccountId !=
                                childObjNewList.AccountId)
        {
            accSetId.add(childObjOld.AccountId);
        }
        else if(childObjOld.AccountId == null)
        {
            accSetId.add(childObjNewList.AccountId);
        }
      }
    }
}

 for(Account parentSetObj:[Select Id, Name,ChildRollup__c
                          From Account Where Id IN:accSetId])
 {
     List<Contact> childSize = parentSetObj.Contact__r;
     parentSetObj.ChildRollup__c=childSize.size();
     updateParent.add(parentSetObj);
 }
   update updateParent;
 }
}
```

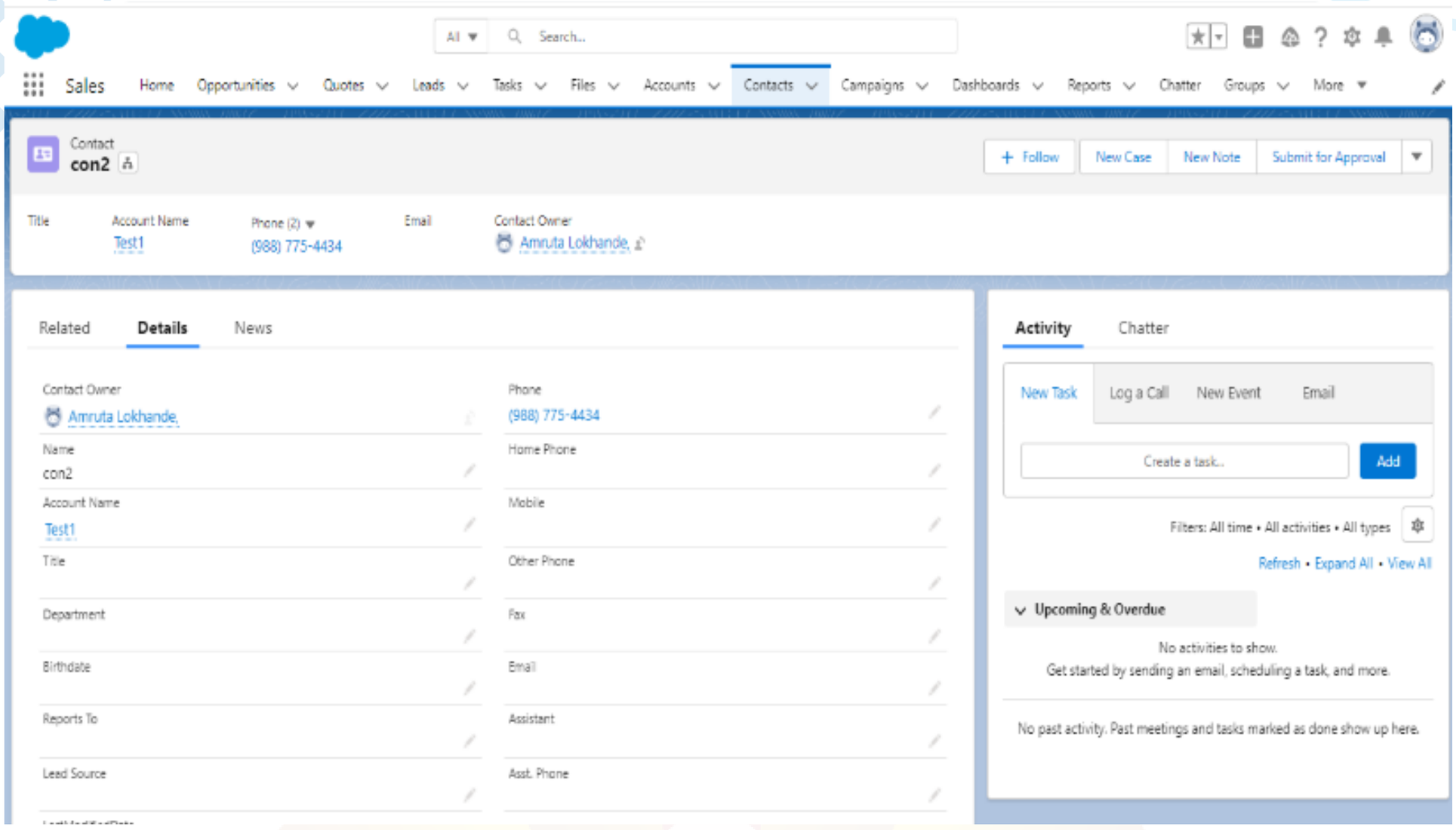
Test Case:

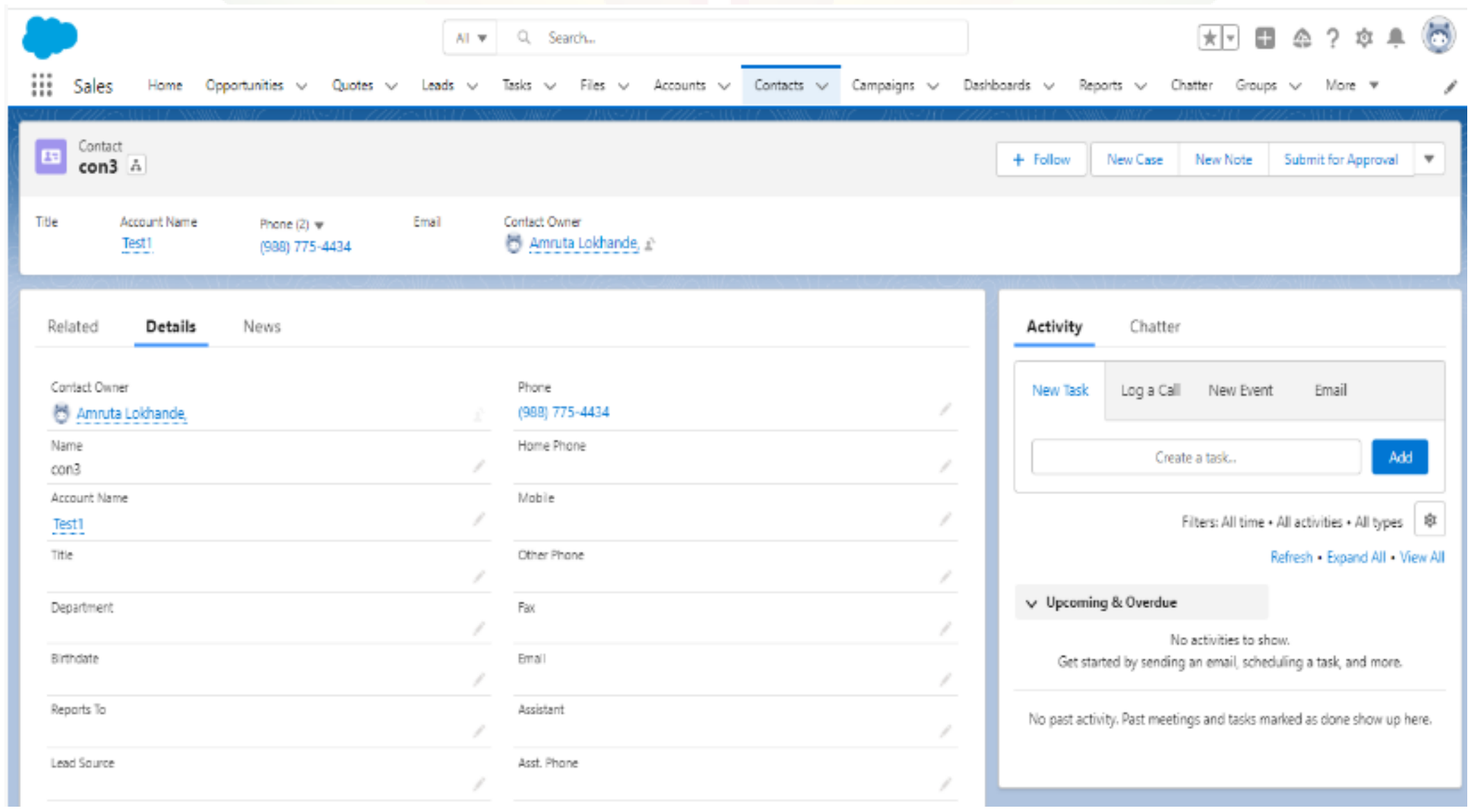
1. Create new Account Record.



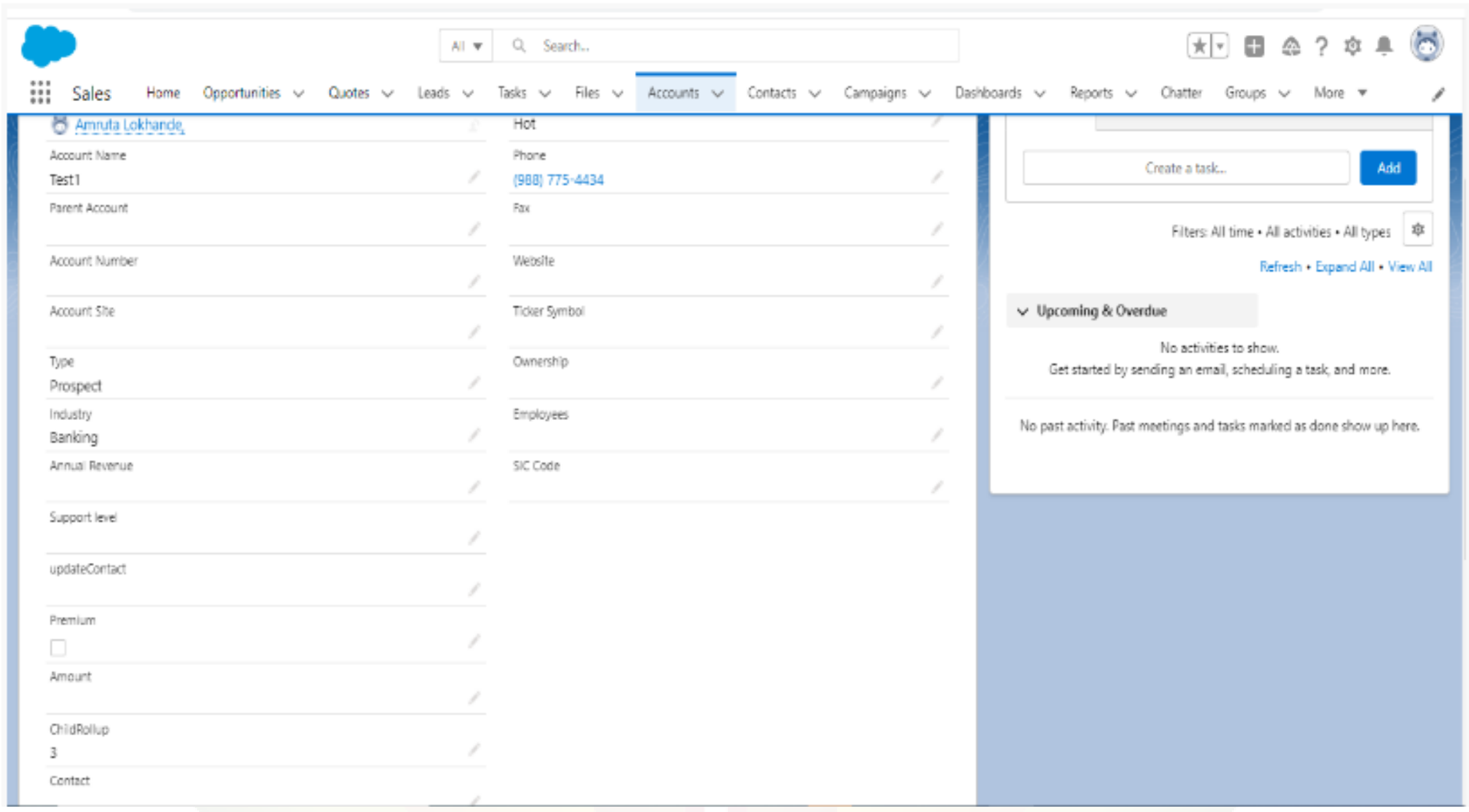
2. Create three child Contact records of same Account.







2. On Account record check the childRollup field. It shows the number of child contacts that account has. As shown in below screenshot



Example 32

Write a trigger on Contact and Add First Name and Last Name of Contact to Associated Account Custom Field Whenever Contact record is Inserted or Updated.

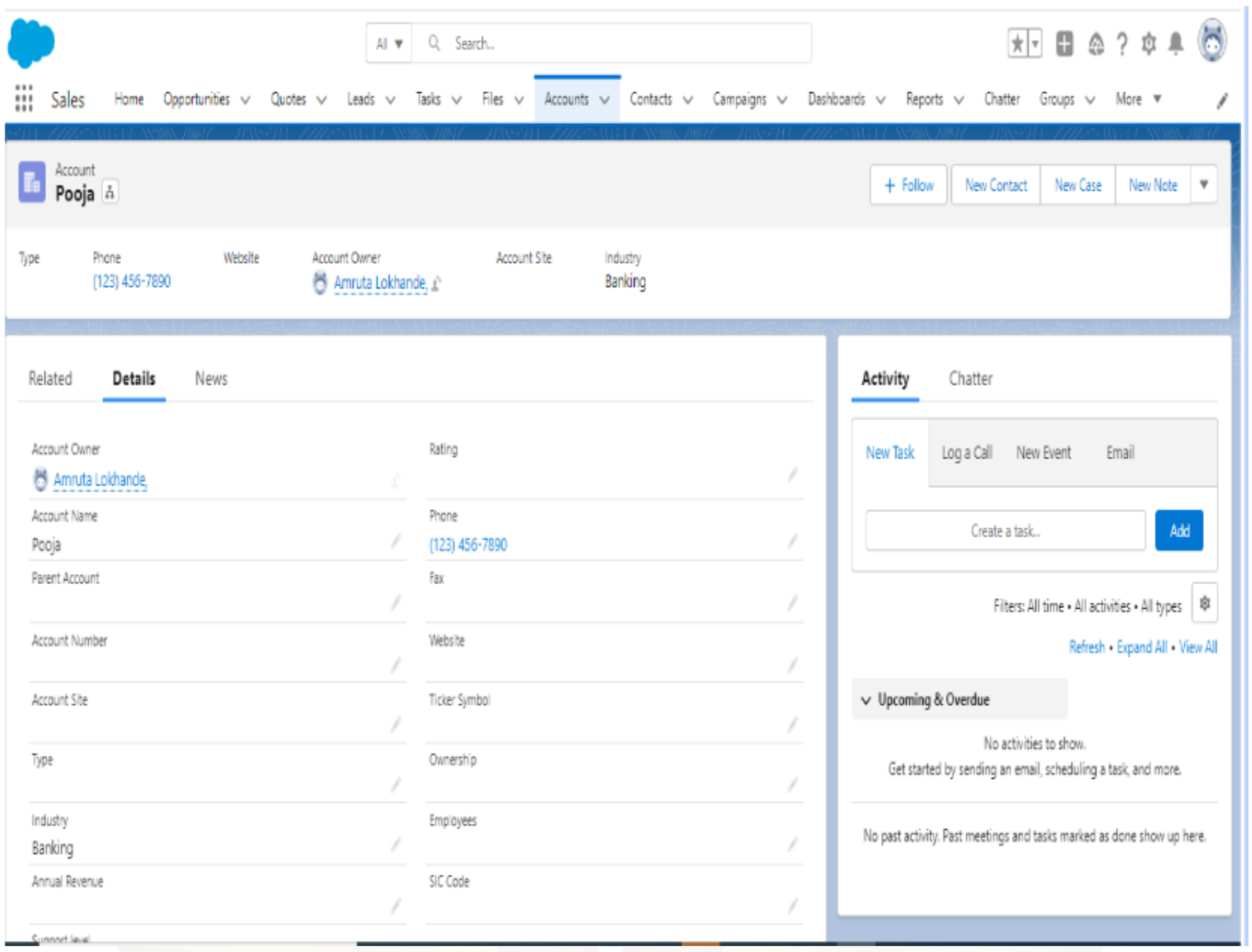
Trigger: ContactAccountTrigger

```
trigger ContactAccountTrigger on Contact (before insert,
before update, after insert, after update) {

  List<Contact> conList =new List<Contact>();
  Set<Id> setid = new  Set<Id>();

  if(trigger.isBefore){
     system.debug('trigger before event');
     conList = trigger.new;
  }

    else if(trigger.isAfter){
          conList=trigger.new;
        for(Contact con:conList){
            setid.add(con.AccountId);
        }
        system.debug('setid ' + setid);

List<Account> accList = [Select Id, Name, updateContact__c
                          From Account  Where Id=:setid];
```

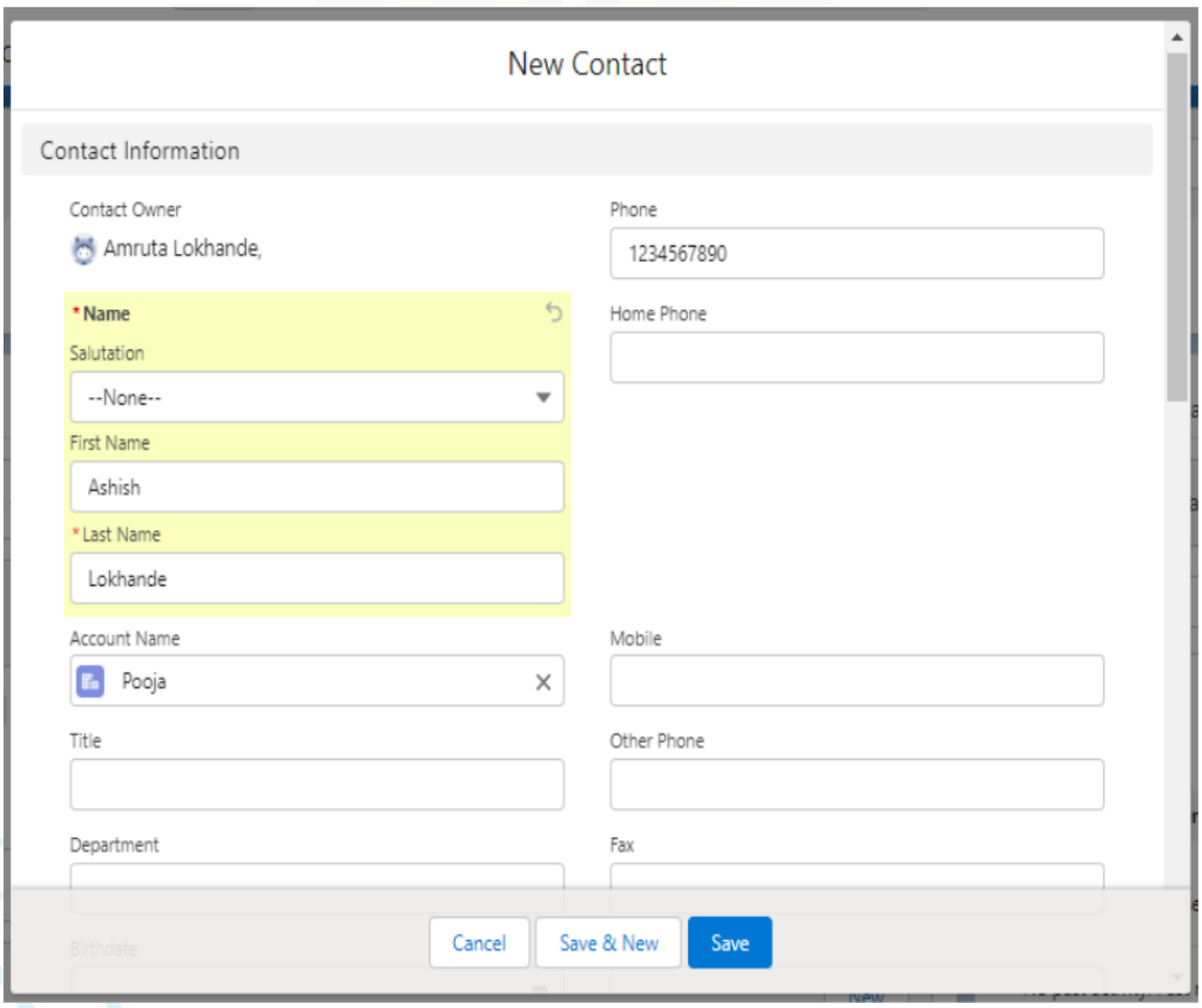
```
if(trigger.isInsert){
    for(Contact c1:trigger.new){
        for(Account a1:accList){
            a1.updateContact__c= c1.FirstName + ''
                                    + c1.LastName;

            update  a1;
        }
    }
}

 else if(trigger.isUpdate){
      for(Contact c2:trigger.new){
         for(Account a2:accList){
            a2.updateContact__c= c2.FirstName +
                                    ' ' + c2.LastName;

            update a2;
         }
      }
   }
}
```

Test Case:

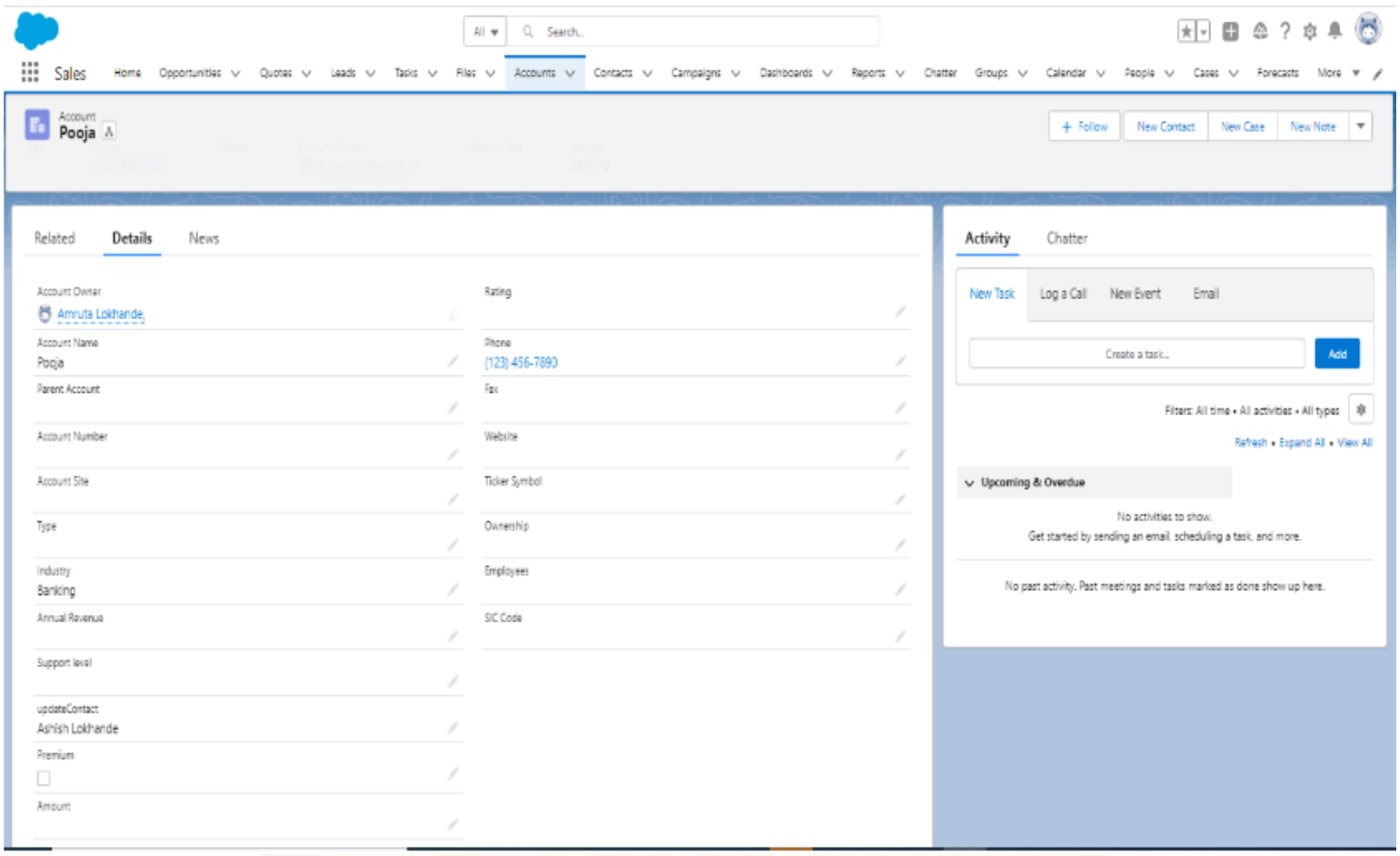
1. Create new Account record.



2. Create new child contact record with contact Last name and First name..



3: Check Account custom field that is Updated_Contact , Contact first name and Last name is associated with it.



Example 33

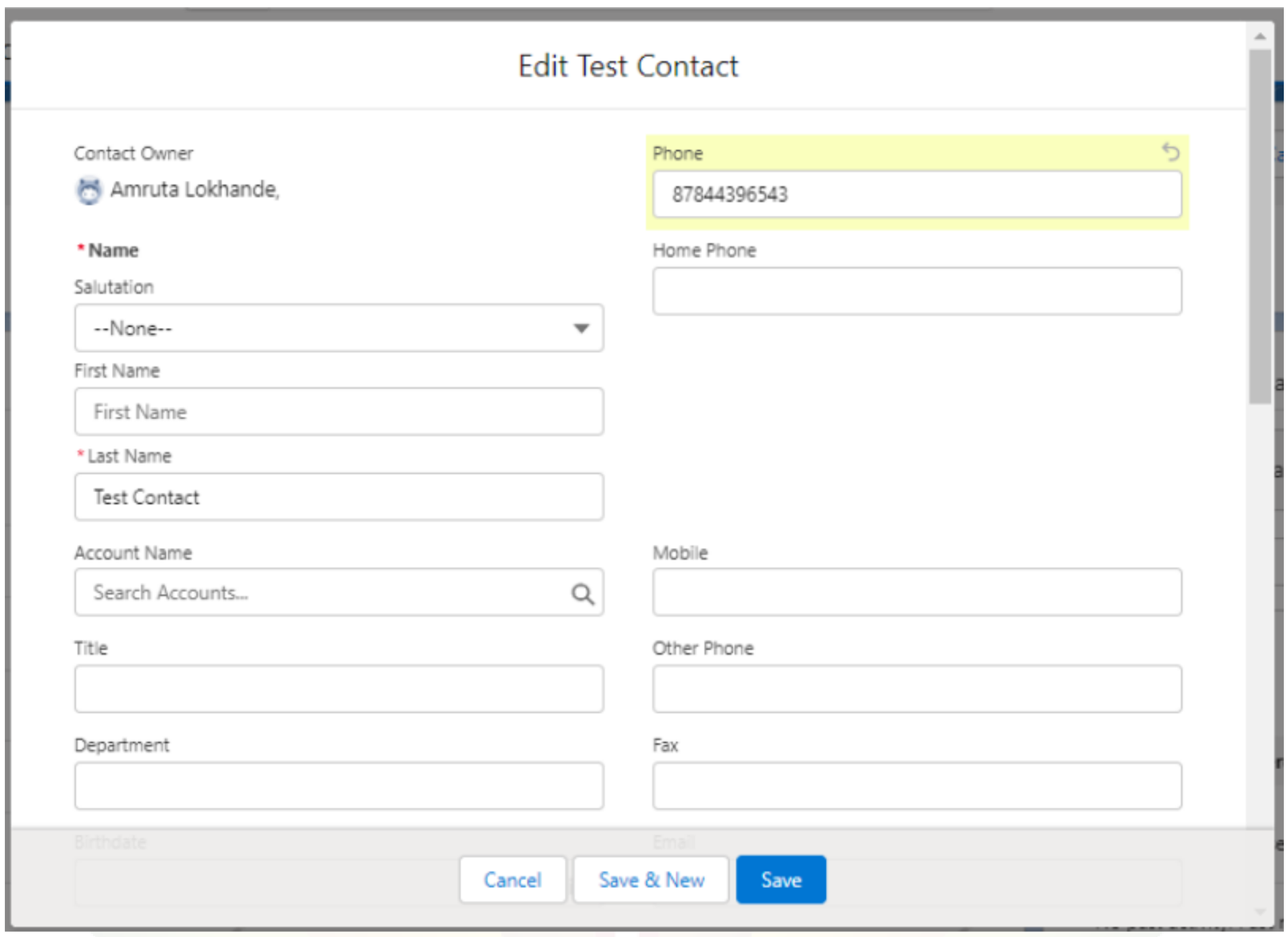
trigger on contact which will create Account record whenever contact is created without an account.

Trigger: ContactCustomTriggerExample.apxt

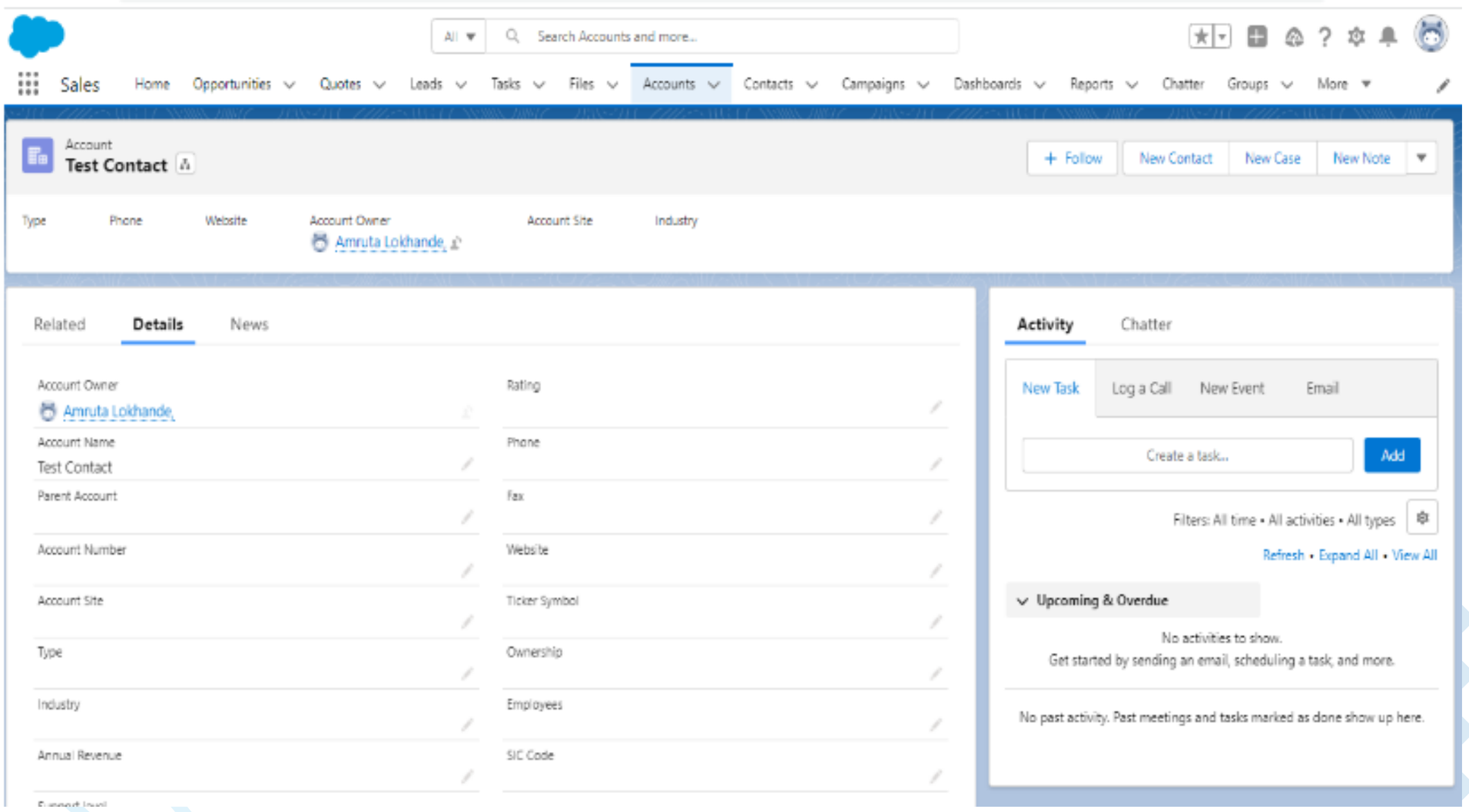
```
trigger ContactCustomTriggerExample on Contact(after insert)
{
 List<Account> accListToInsert = new List<Account>();
  for(Contact con : Trigger.New) {
 //check if account is null on contact
     if(con.AccountId == null ) {
        Account acc = new Account();
 //Add all required field on Account
        acc.Name = con.LastName;
        acc.Phone = con.Phone;
        accListToInsert.add(acc);
     }
   }
   if(!accListToInsert.isEmpty()){
   insert accListToInsert;
   }
}
```

Test Case:

1. Create new contact record without parent account.



2. Check in All Accounts new account record will create automatically with name same as contact last name .



Example 34

Write a trigger to update parent account phone number whenever the contact phone number is updated using trigger handler and helper class in Salesforce

Trigger: contactTrigger

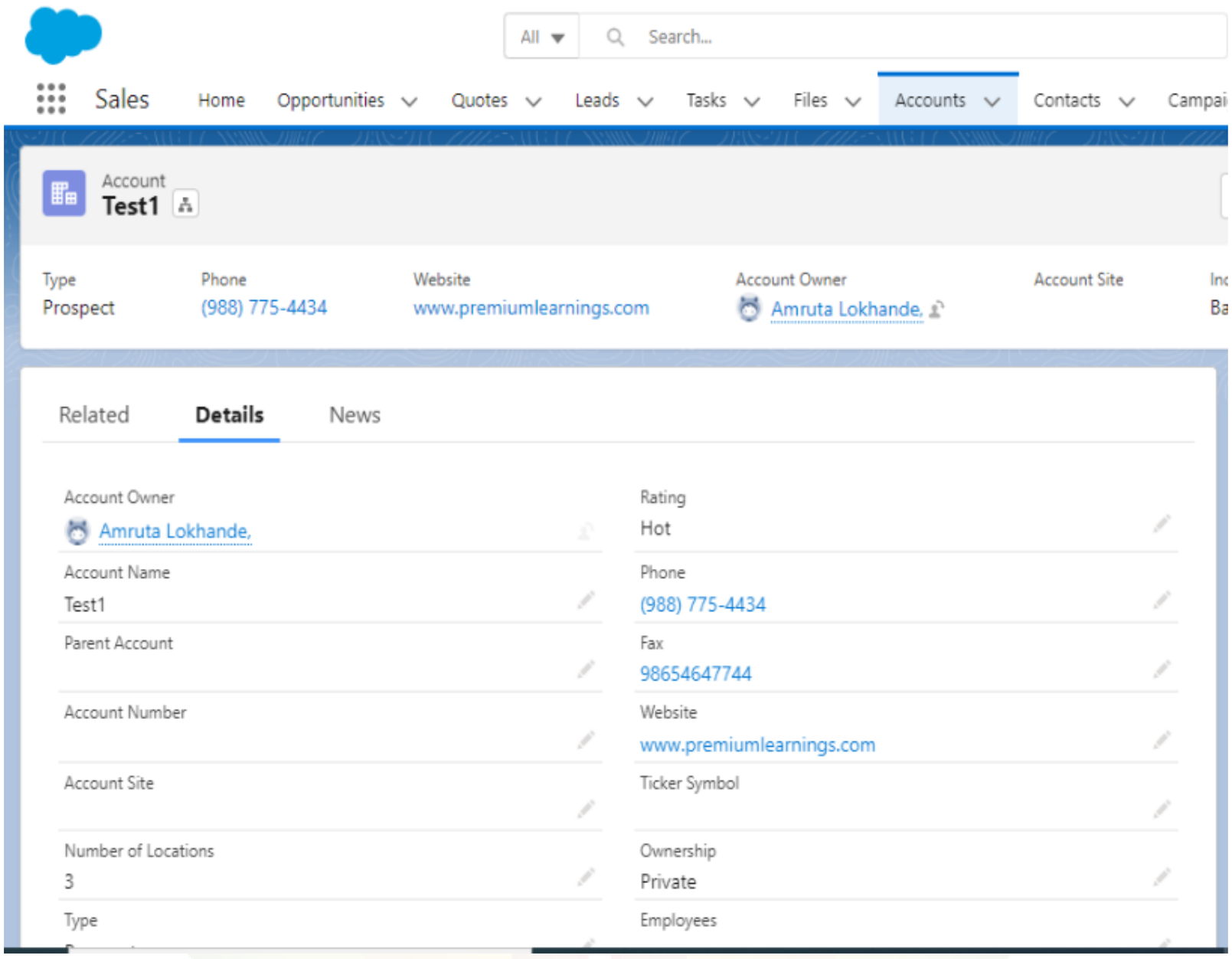
```
trigger contactTrigger on Contact (before insert, before upda
te, before delete, after insert, after update, after delete,
after undelete)
{

    if(trigger.isBefore ){
        system.debug('I am before trigger ');
    }
    else if(trigger.isAfter){
      system.debug('I am after trigger ');
       if(trigger.isUpdate){
        contactTriggerHandler.afterUpdateHelper(trigger.new);
       }
    }
}
```

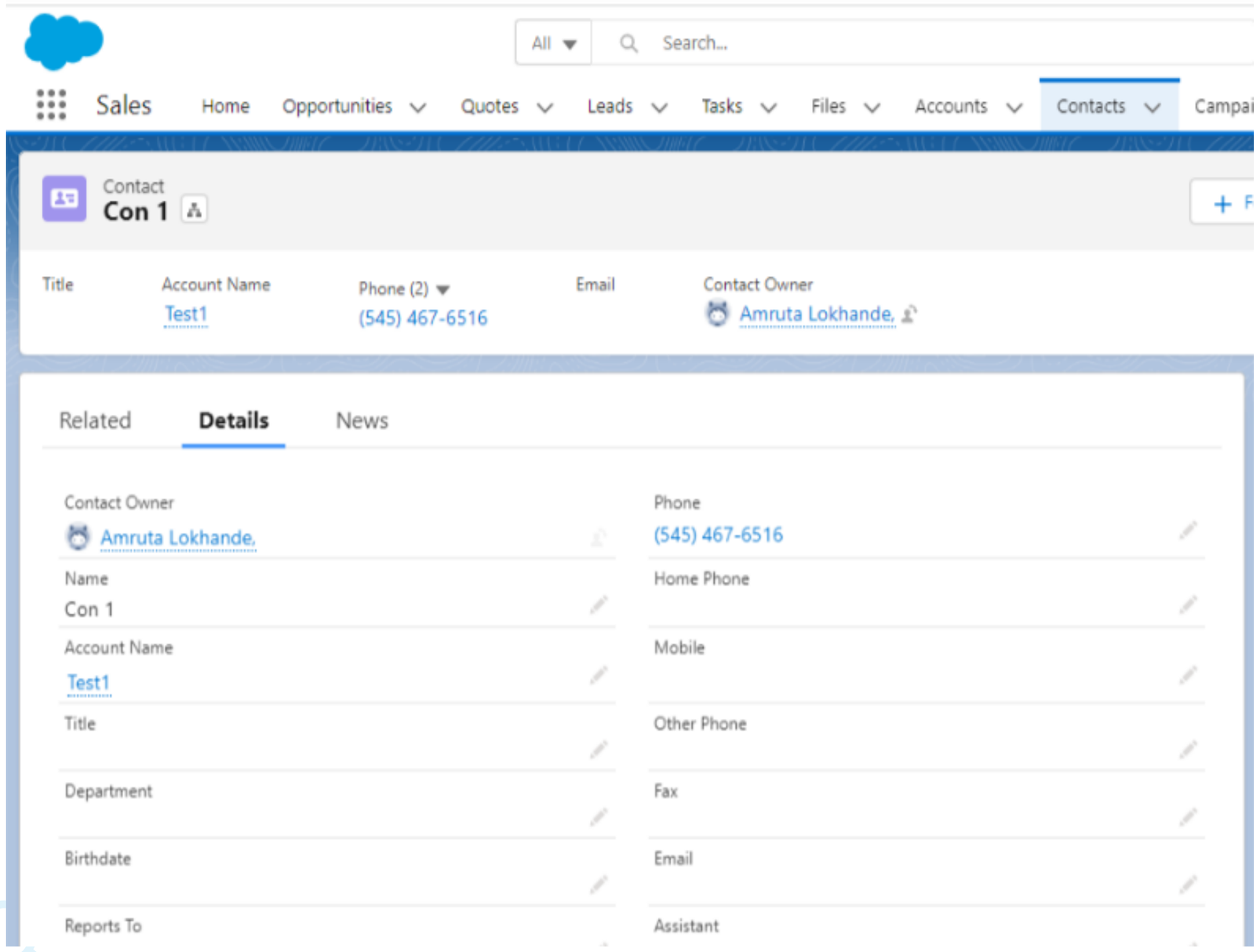
Class :

```
public class contactTriggerHandler {

    public static void afterUpdateHelper(List<Contact> conList)
    {

     Set<Id> setId = new Set<Id>();

    for(Contact con:conList){

        setId.add(con.AccountId);
    }

     system.debug('setId ' + setId);

     List<Account> accItemList = [Select Id, Name, Phone,
                        (Select Id, FirstName, LastName, Phone,
                        AccountId From Contacts) From Account
                        Where Id IN:setId];

    for(Account a1:accItemList){
        for(Contact c1:a1.Contacts){
            if(c1.Phone !=null){
                a1.Phone = c1.Phone;
                update accItemList;
            }
        }
    }
}
```

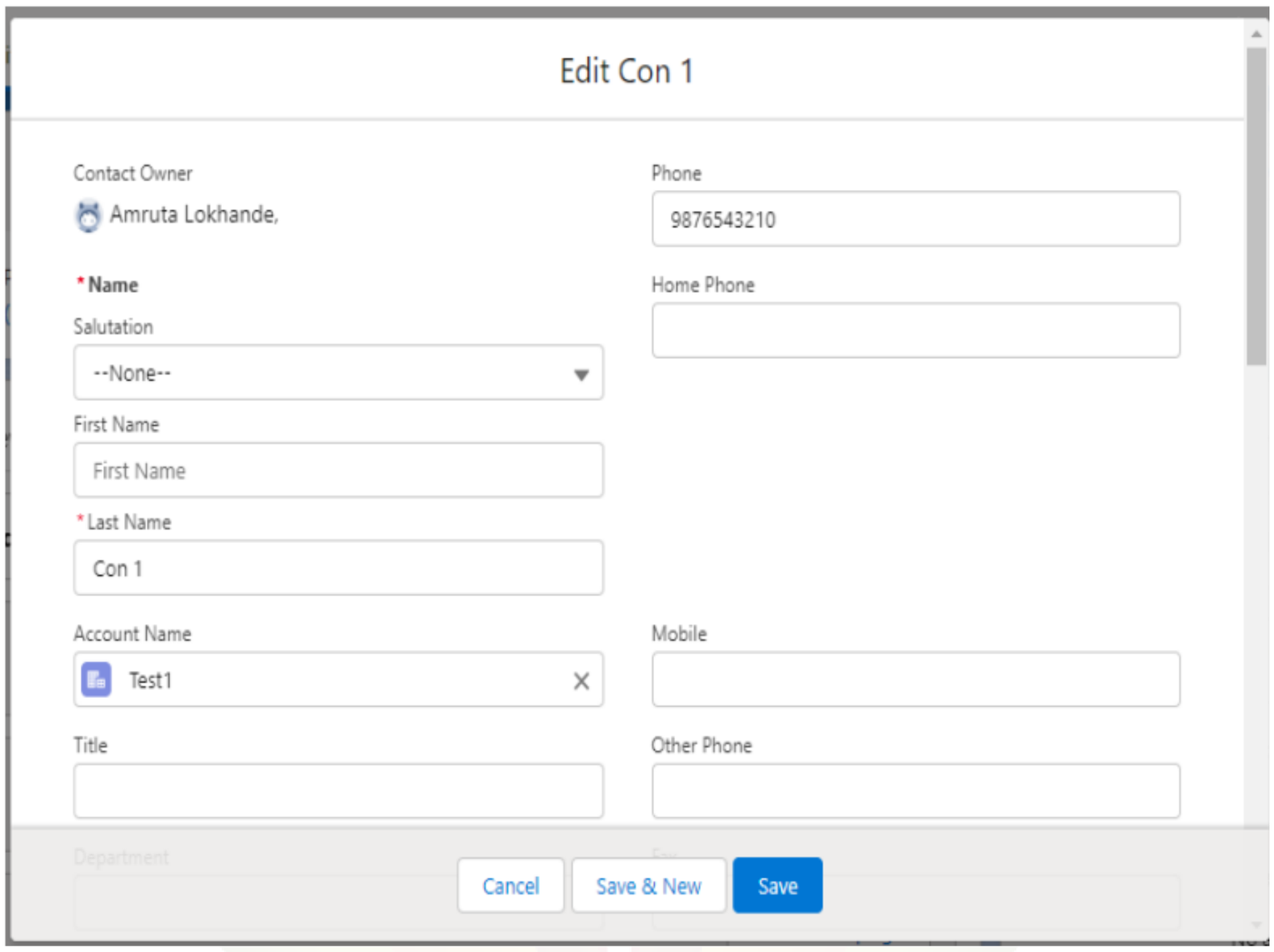
Test Case:

Create Account and contact record with phone number as shown below.

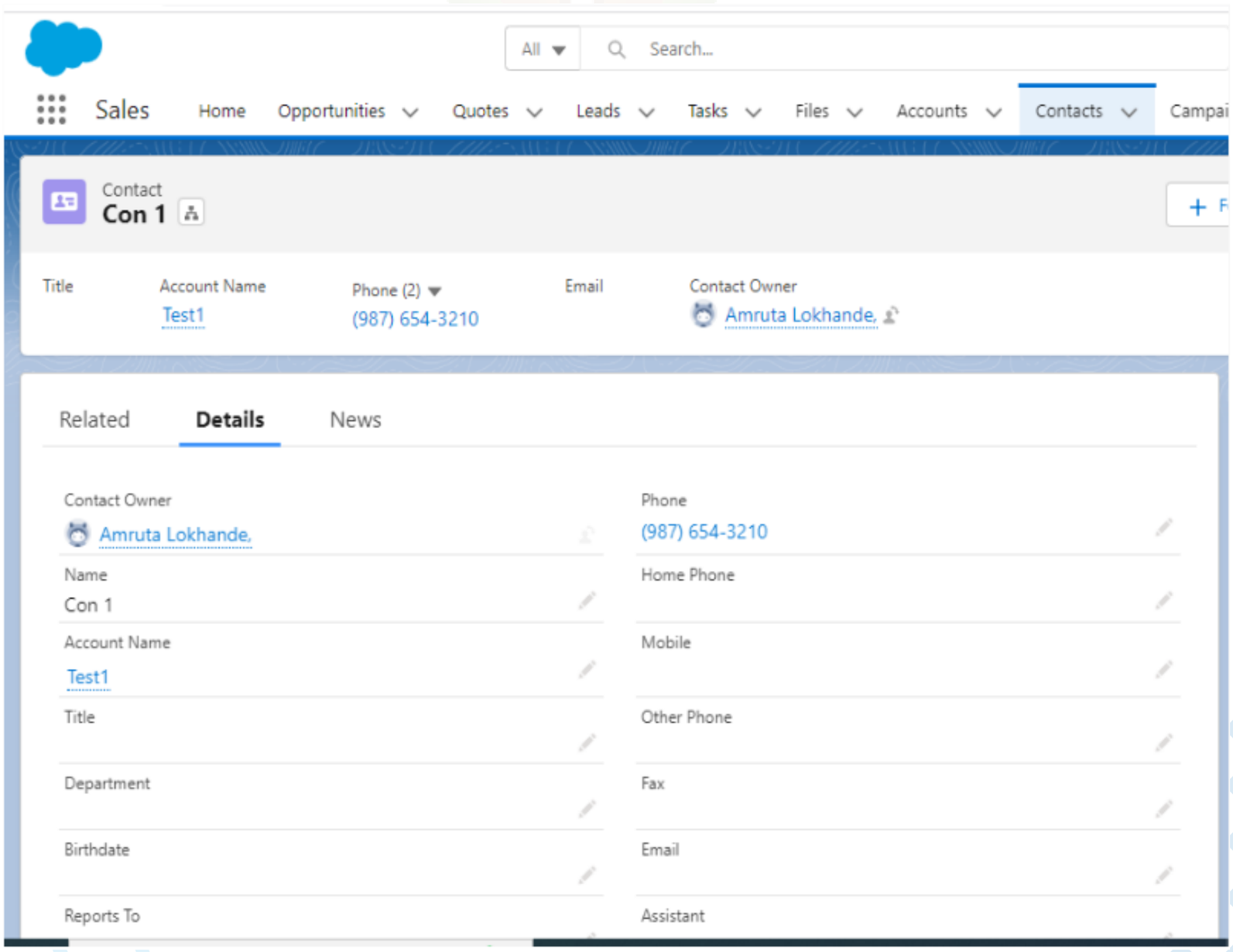




2: Now update the phone field of contact and you can see the parent account phone number will update whenever the contact phone number is updated



Now check phone field of parent account.



Example 35

When ever the Account is created with Industry as Banking then create a contact for account, Contact Lastname as Account name and contact phone as account phone.

Trigger: CreateAccountContact.apxt

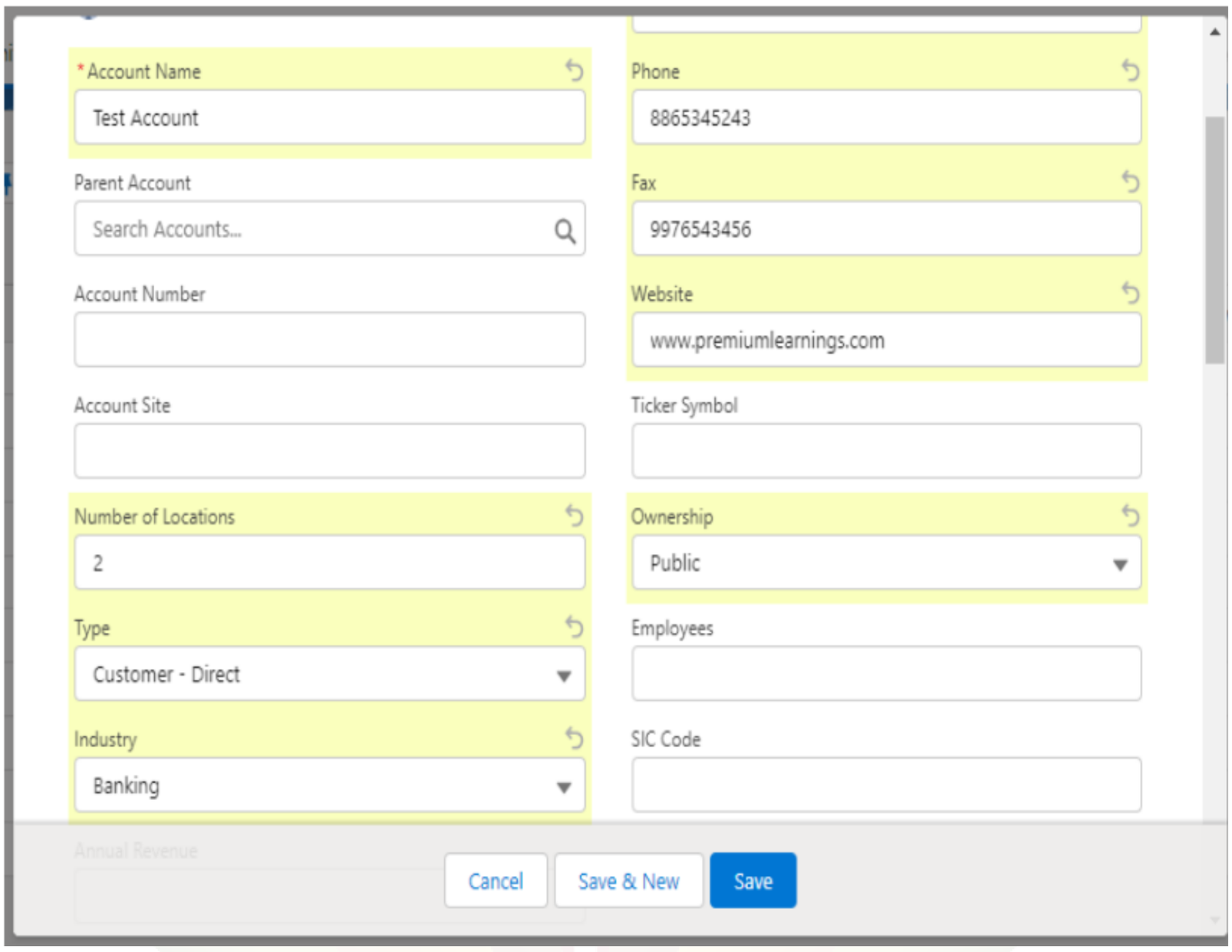
Object : Account

Trigger: After Insert

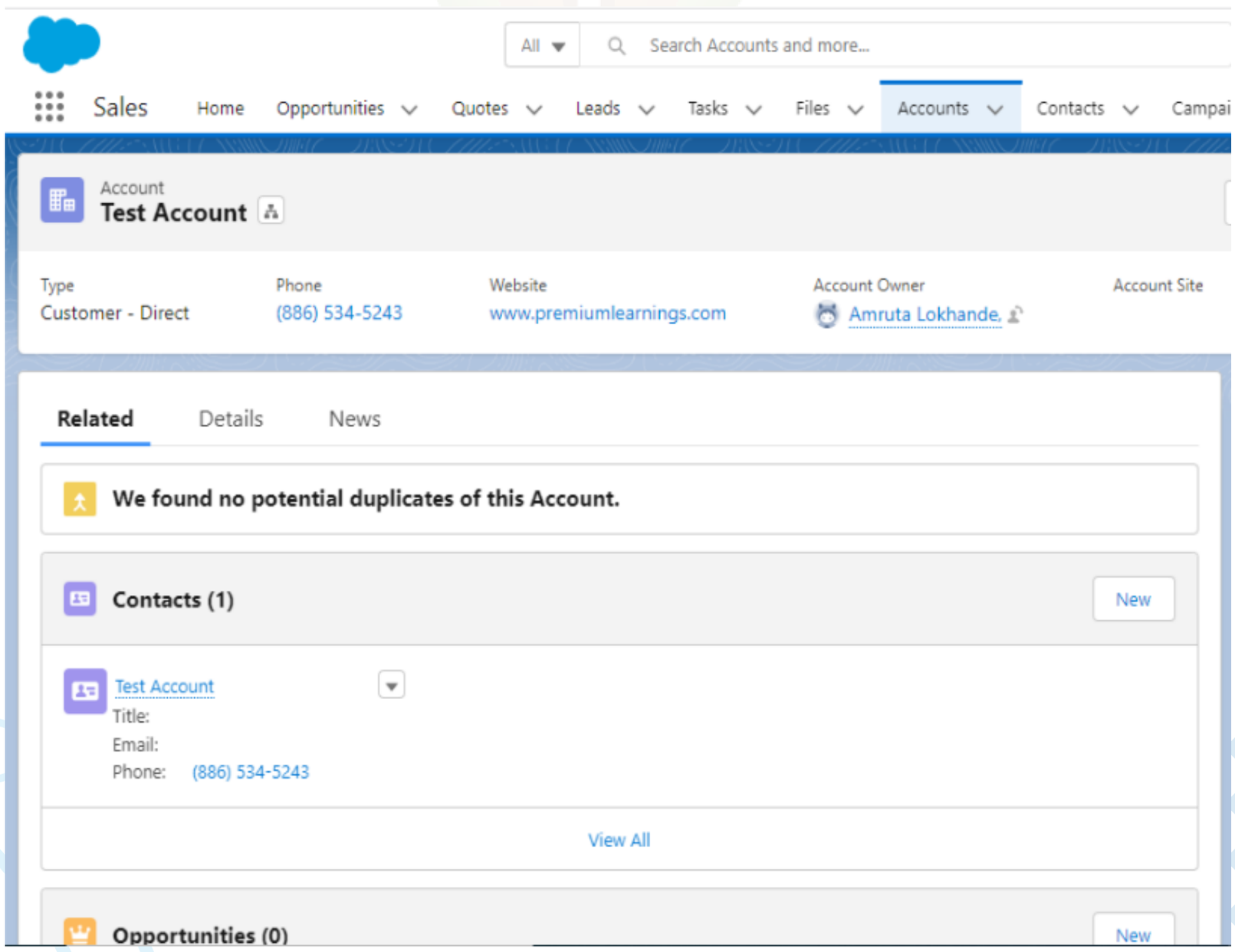
```
trigger CreateAccountContact on Account (after insert)
{
 list<contact> contacts = new list<contact>();
  for(account acc : trigger.new){
    if(acc.Industry == 'Banking'){
       contact con = new contact();
       con.LastName = acc.name;
       con.Phone = acc.Phone;
       con.AccountId = acc.Id;
       contacts.add(con);
    }
  }
   insert contacts;
}
```

Test Case:

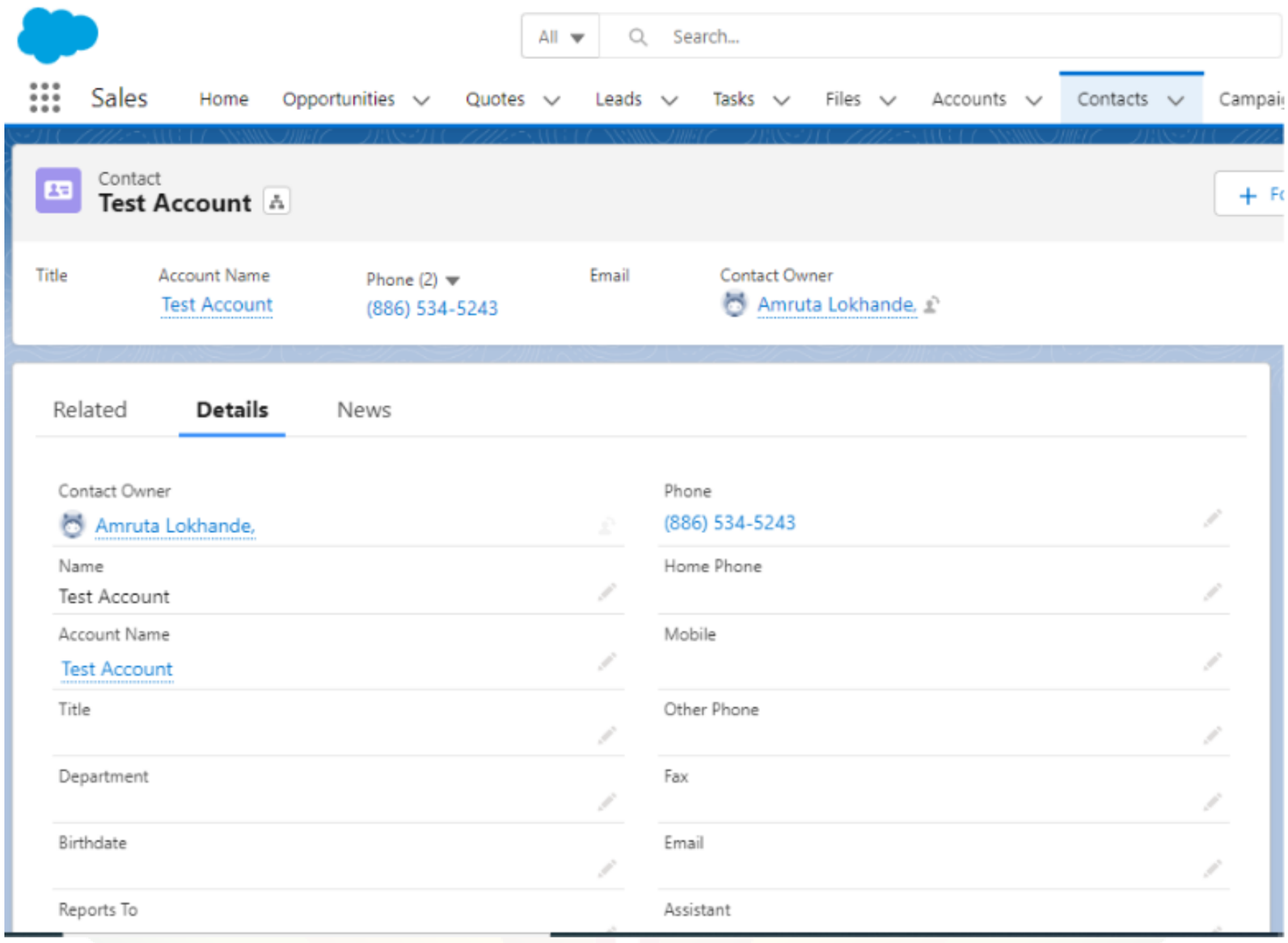
1. Create Account record and industry must be Banking as shown in below screenshot.



2. Check the new child Contact record will create automatically with same name as like Account Last name and with same Phone.



Contact Last name as Account name and contact phone as account phone.



Example 36

Apex Trigger on Contact to Create Account Automatically and map to related account Id to Contact Whenever New Contact is Created in Salesforce | trigger to create account automatically whenever contact is created in salesforce.

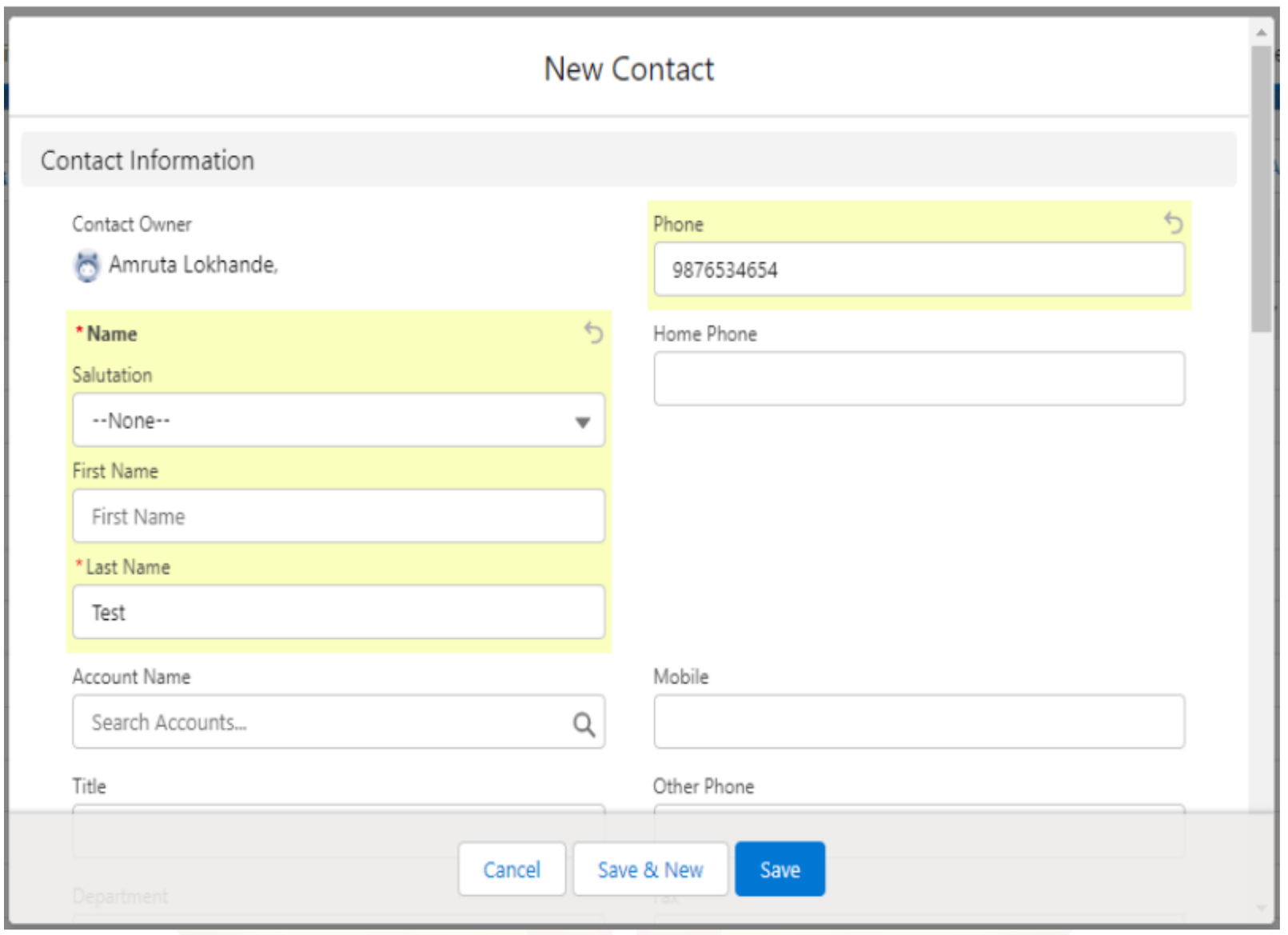
Trigger: CreateAutoAccountFromContact

```
trigger CreateAutoAccountFromContact on Contact (before
insert, after insert) {
 if(trigger.isBefore){
    system.debug('trigger before event');
      if(trigger.isInsert){
        List<Contact> listOfContact=new List<Contact>();

        for (Contact conObj : trigger.new)
        {
           if(String.isBlank(conObj.accountid))
           {
              listOfContact.add(conObj);
           }
      }

system.debug('listOfContact_1 '+ listOfContact);

  if (listOfContact.size() > 0)
  {
    List<Account> createNewAcc=new List<Account>();
    Map<String,Contact> conNameKeys = new
                        Map<String,Contact>();
```

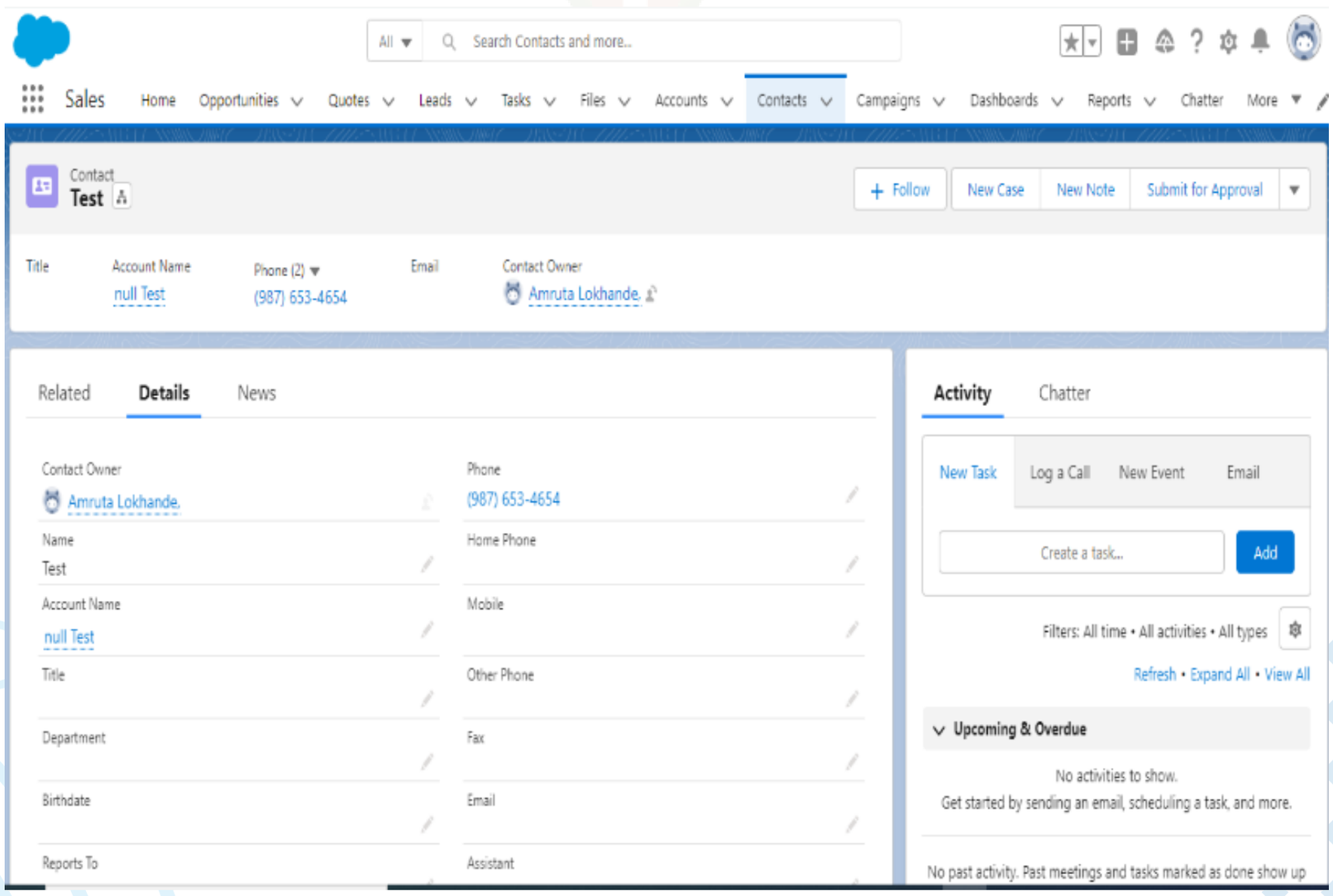
```
        for (Contact con : listOfContact)
        {
          String accountName = con.firstname + ' ' +
                                                con.lastname;
          conNameKeys.put(accountName,con);
          Account accObj = new Account();
          accObj.Name = accountName;
          accObj.Phone= con.MobilePhone;
          createNewAcc.add(accObj);
        }
        Insert createNewAcc;
        for (Account acc : createNewAcc)
        {
          System.debug('mapContainsKey ' +
          conNameKeys.containsKey(acc.Name));

         if (conNameKeys.containsKey(acc.Name))
         {
                conNameKeys.get(acc.Name).accountId = acc.Id;
                }
            }
          }
        }
          else if(trigger.isAfter){
                system.debug('trigger after event');
          }
    }
```

Test Case:

1. Create new Contact record without Account as shown in below screenshot.



2. Check related account record will create automatically whenever contact is created.k the new child Contact record will create automatically with same name as like Account Last name and with same Phone.



Example 37

The number of contacts which are equal to the number which we will enter in the Number of Locations field on the Account Object.

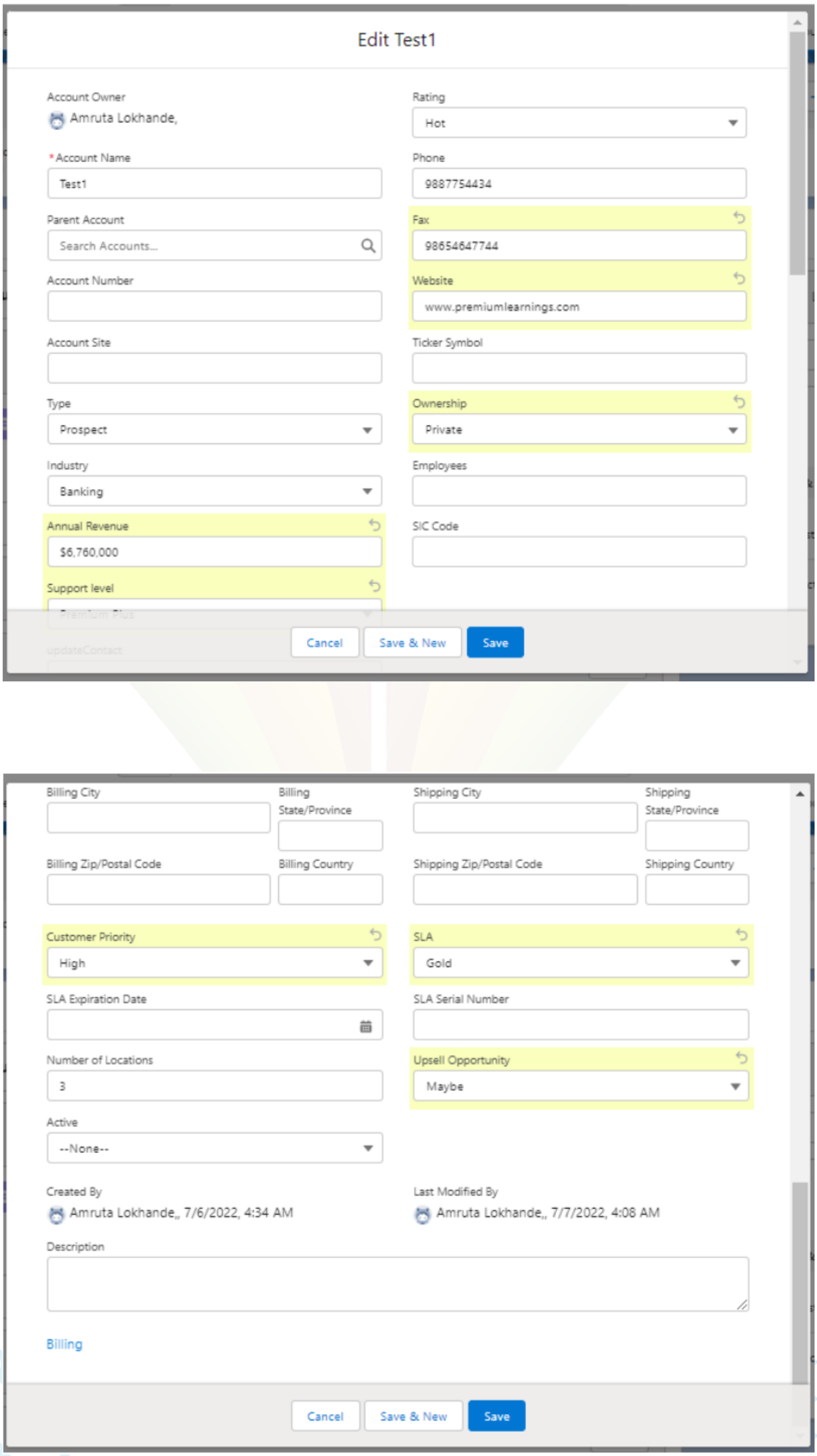
Trigger: CreateNoOfContacts

Create a custom field on parent Object name as NumberofLocations__c (Number Type).

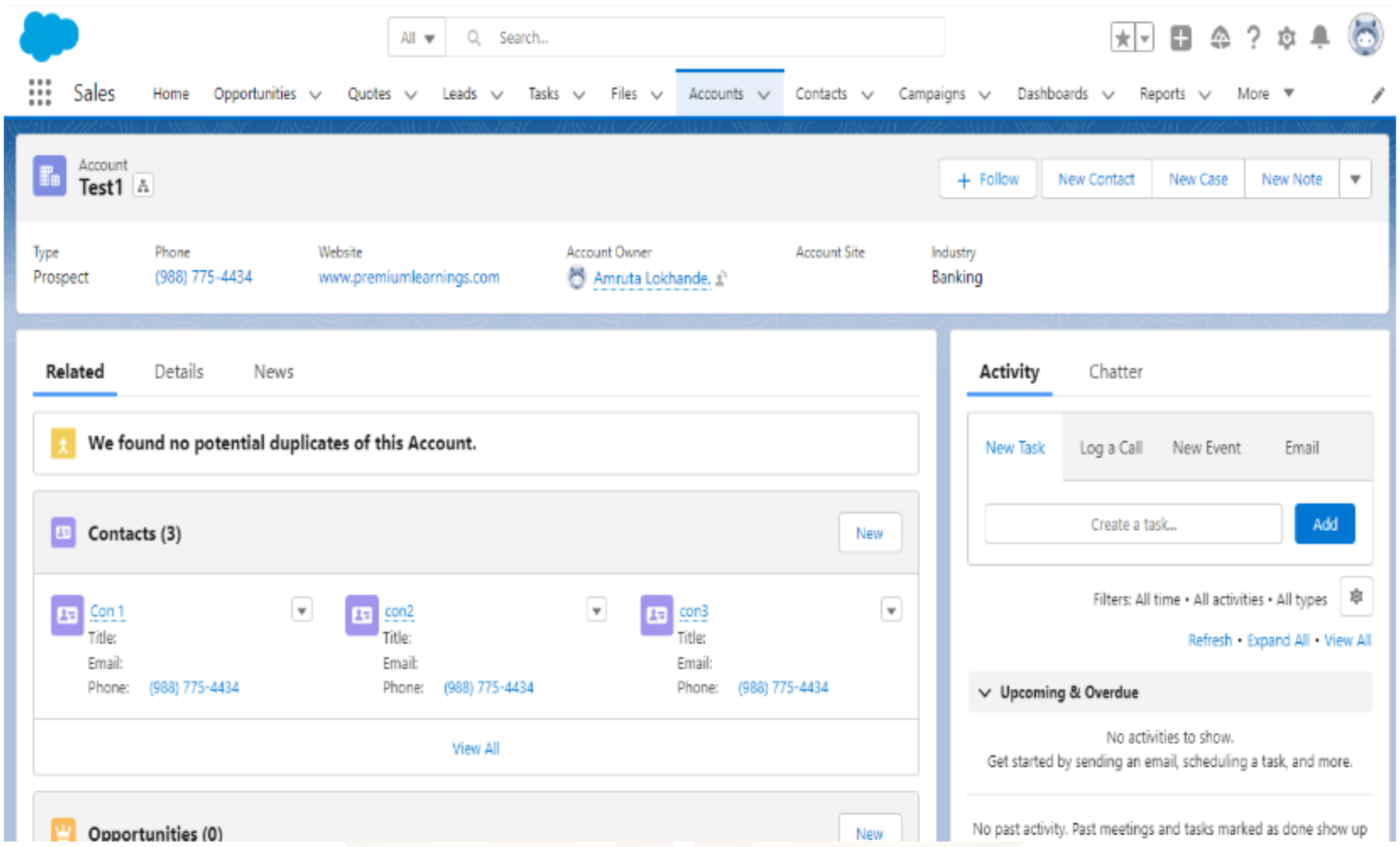
```
trigger CreateNoOfContacts on Account (after insert)
{

  list<contact> listContact = new list<contact>();
  map<id,decimal> mapAcc=new map<id,decimal>();

  for(Account acc:trigger.new){
      mapAcc.put(acc.id,acc.NumberofLocations__c);
  }
  if(mapAcc.size()>0 && mapAcc!=null){
     for(Id accId:mapAcc.keyset()){
        for(integer i=0;i<mapAcc.get(accId);i++){
            contact newContact=new contact();
            newContact.accountid=accId;
            newContact.lastname='contact'+i;
            listContact.add(newContact);
        }
     }
}
if(listContact.size()>0 && listContact!=null)
    insert listContact;
}
```

Test Case:

1. Create Account record and must enter Number of locations as shown in below screenshot.



2. Check the number of contacts will create which are equal to the number which we will enter in the Number of Locations field on the Account Object.



Example 38

Create the object called “Books” and create field “Price”(data type is Currrency) under this object. Whenever we enter some amount of money in the Price field and once we click on save button, the value we entered in the Price field is 10% less than the actual price. This is applicable for while both inserting and updating records.

Trigger: DiscountTrigger.apxt

Create custom object.

1)Custom Object:- Book__c custom field:- Price__c (Currency type)

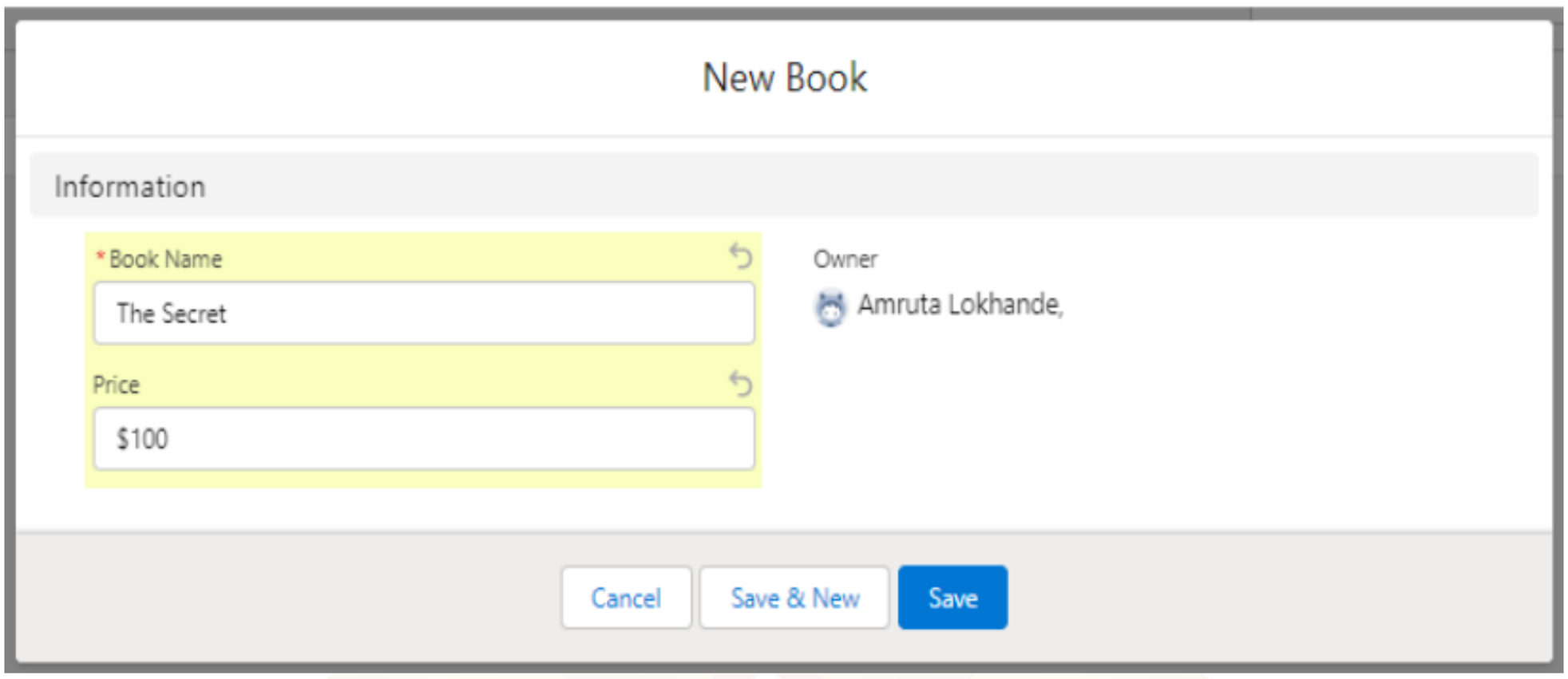
```
trigger DiscountTrigger on Book__c (before insert,
before update)
{
   List<Book__c> books = Trigger.new;
   PriceDiscount.applyDiscount(books);
}
```

Class : PriceDiscount.apxc

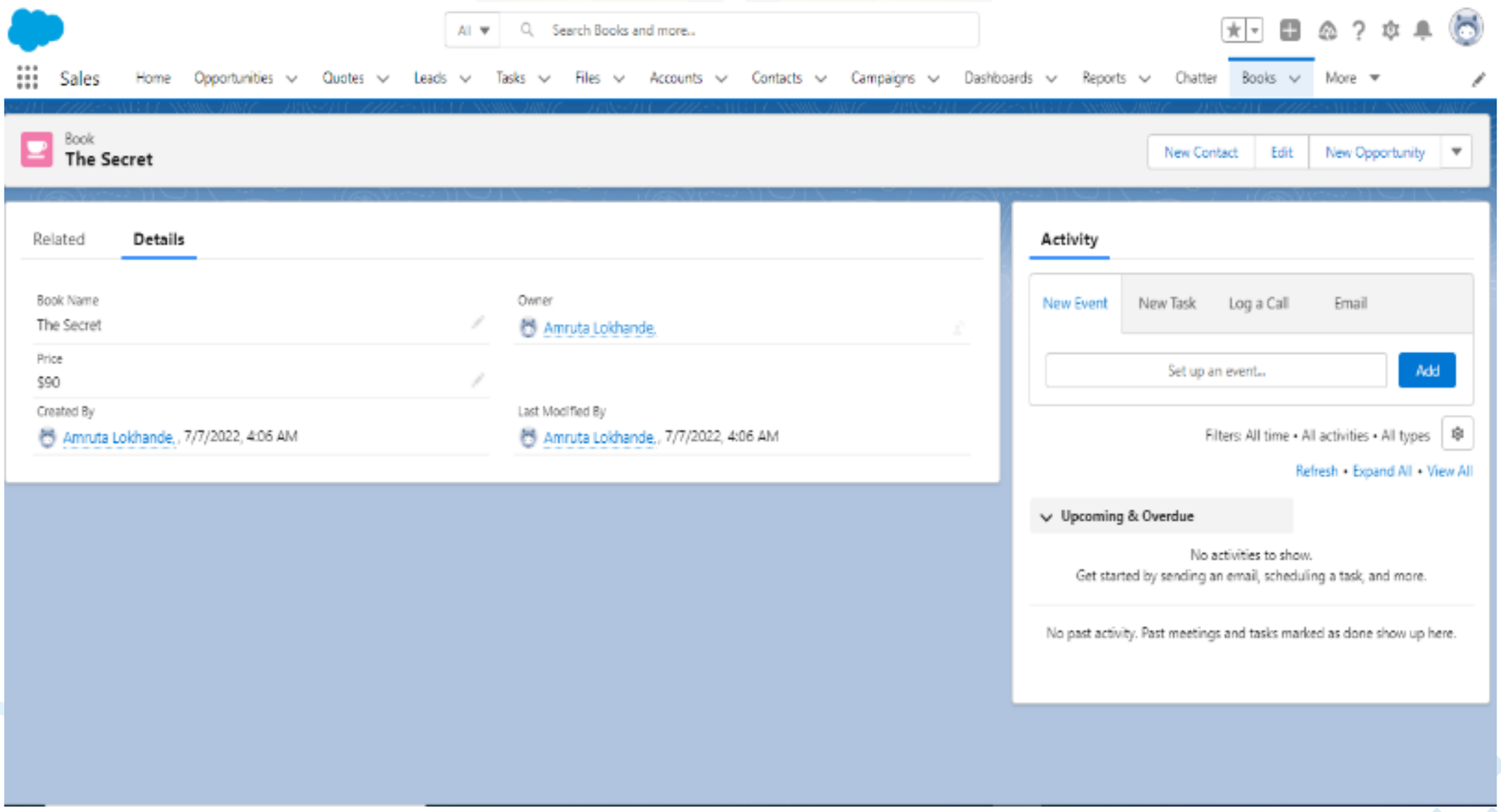
```
public class PriceDiscount
{
   public static void applyDiscount(List<Book__c> books)
   {
      for (Book__c b :books)
     {
       b.Price__c *= 0.9;
      }
   }
}
```

Test Case:

1. Create record of Book object.as shown in the below screenshot.



2. Check the value we entered in the Price field is 10% less than the actual price.



Example 39

Write a trigger whenever Opportunity is deleted the corresponding Account and Contact should be deleted uses of Apex trigger in Salesforce.

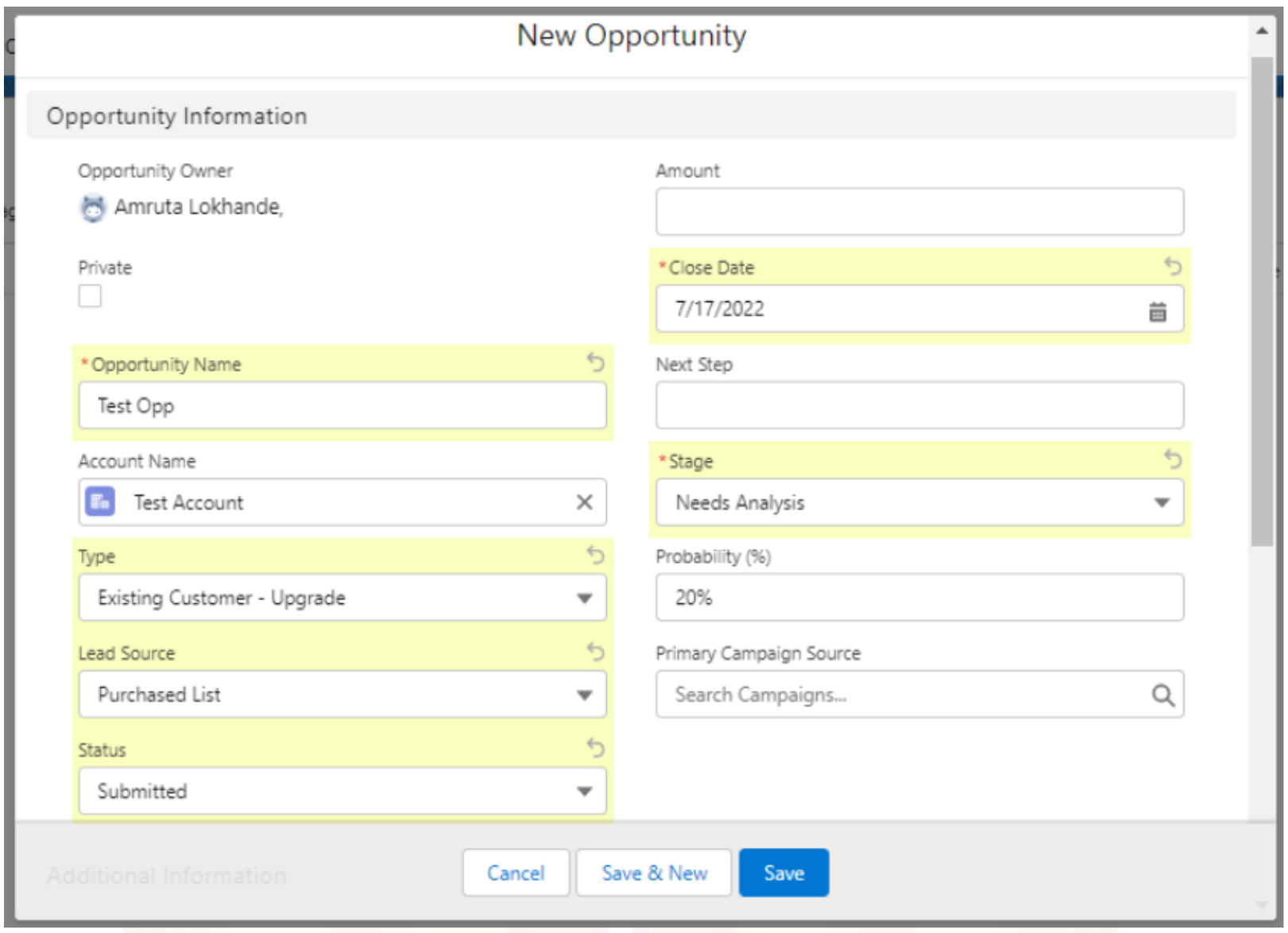
Trigger: OpportunityDeleteTrigger

```
trigger OpportunityDeleteTrigger on Opportunity (
before delete, after delete)
{
 if(trigger.isBefore){
 system.debug('trigger before trigger');
 }
  else if(trigger.isAfter){
      system.debug('trigger after trigger');
       if(trigger.isDelete){
          Map<Id,Opportunity> oppItemMap=new
          Map<Id,Opportunity>();
          for(Opportunity opp:Trigger.old){
              oppItemMap.put(opp.AccountId,opp);
          }

List<Account> oppRelListDelete=new List<Account>();
List<Account> accObjlist=[select Id,Name, Industry,
            Website, Type, Phone, Description From
            Account where Id in:oppItemMap.keyset()];

          for(Account acc:accObjlist){
              oppRelListDelete.add(acc);

      }
```

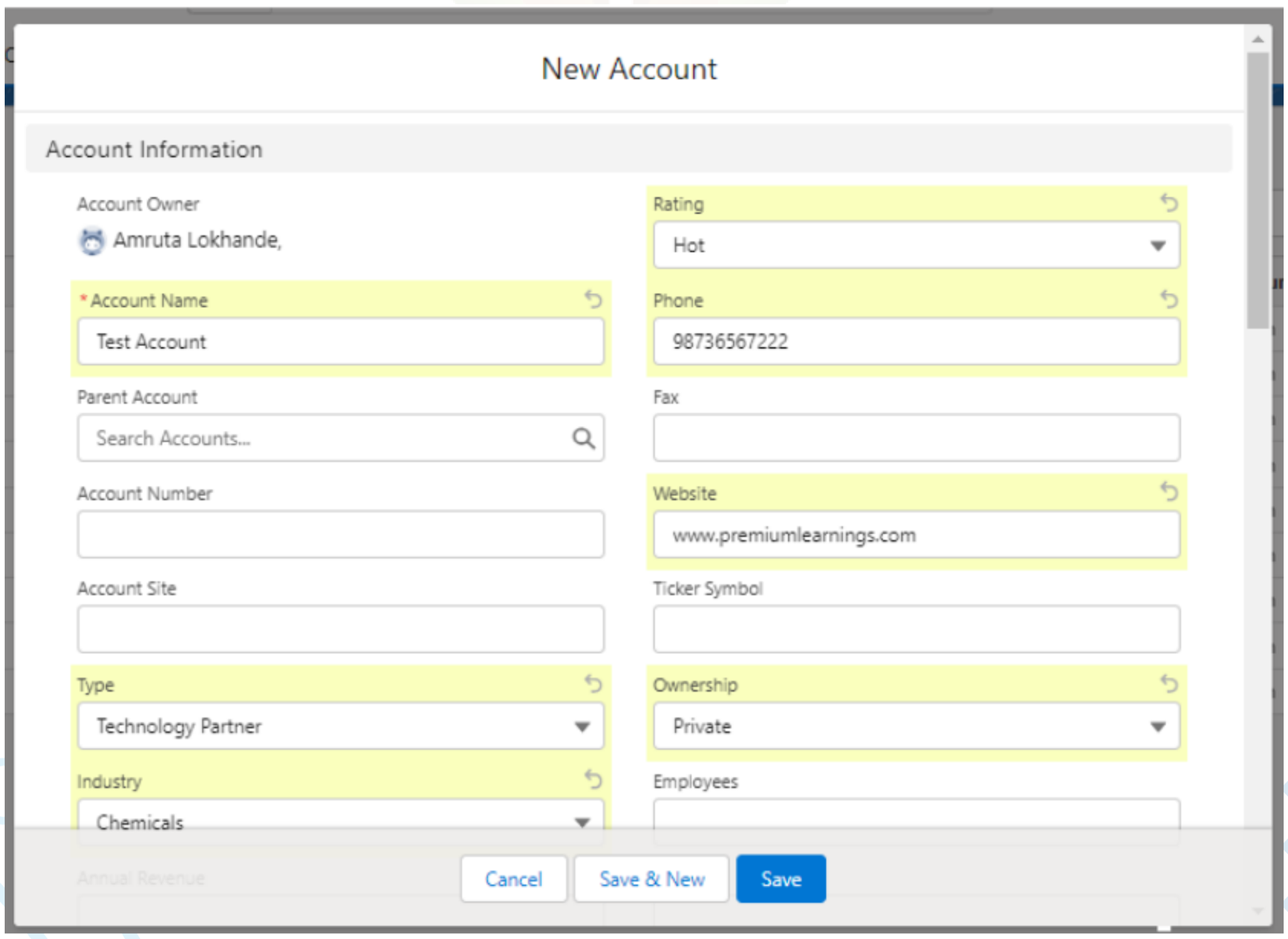
```
        if(!oppRelListDelete.isEmpty() &&
                                    oppRelListDelete.size()>0)
        {

            delete oppRelListDelete;
        }

    }
  }
}
```

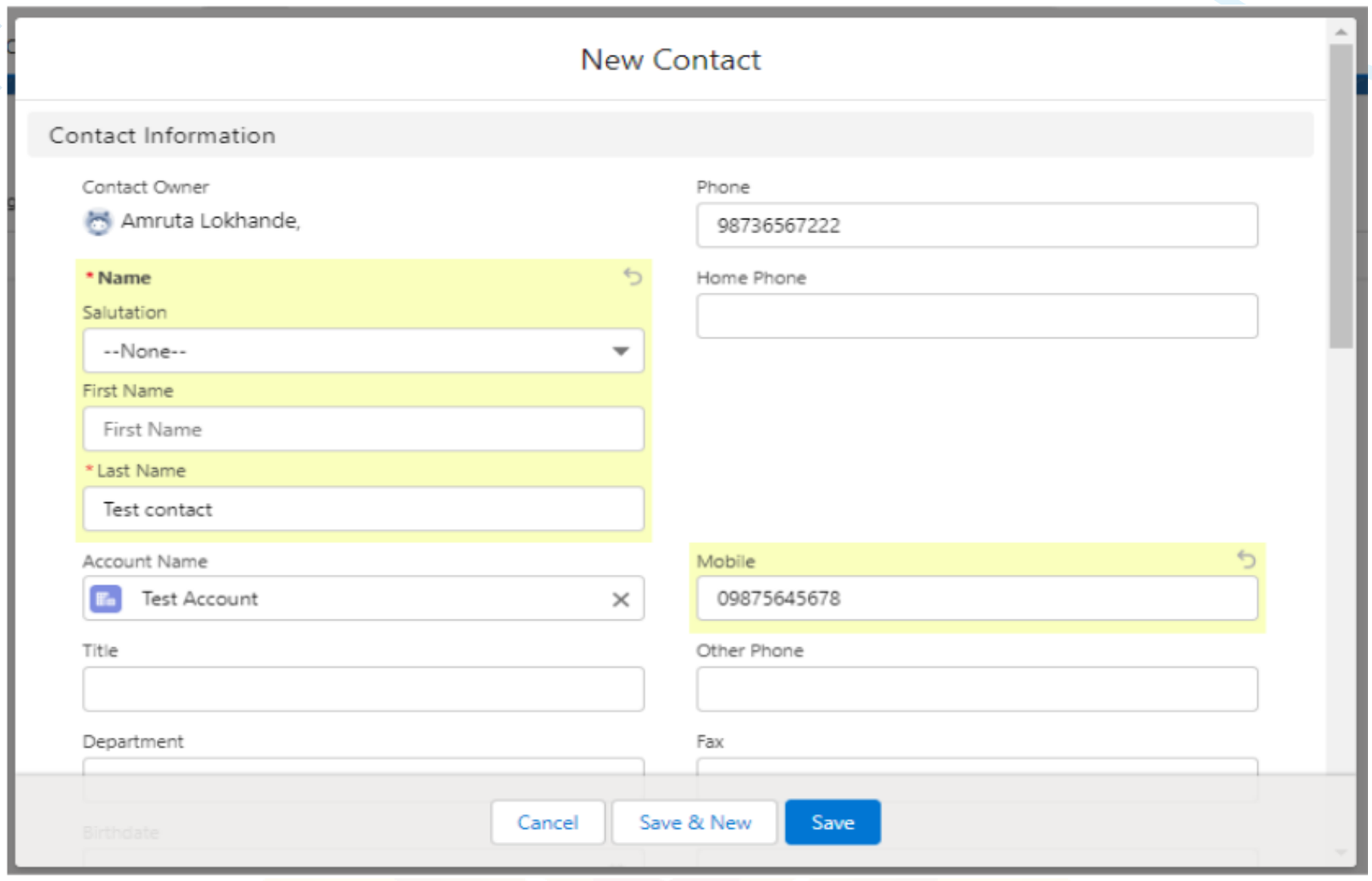
Test Case:

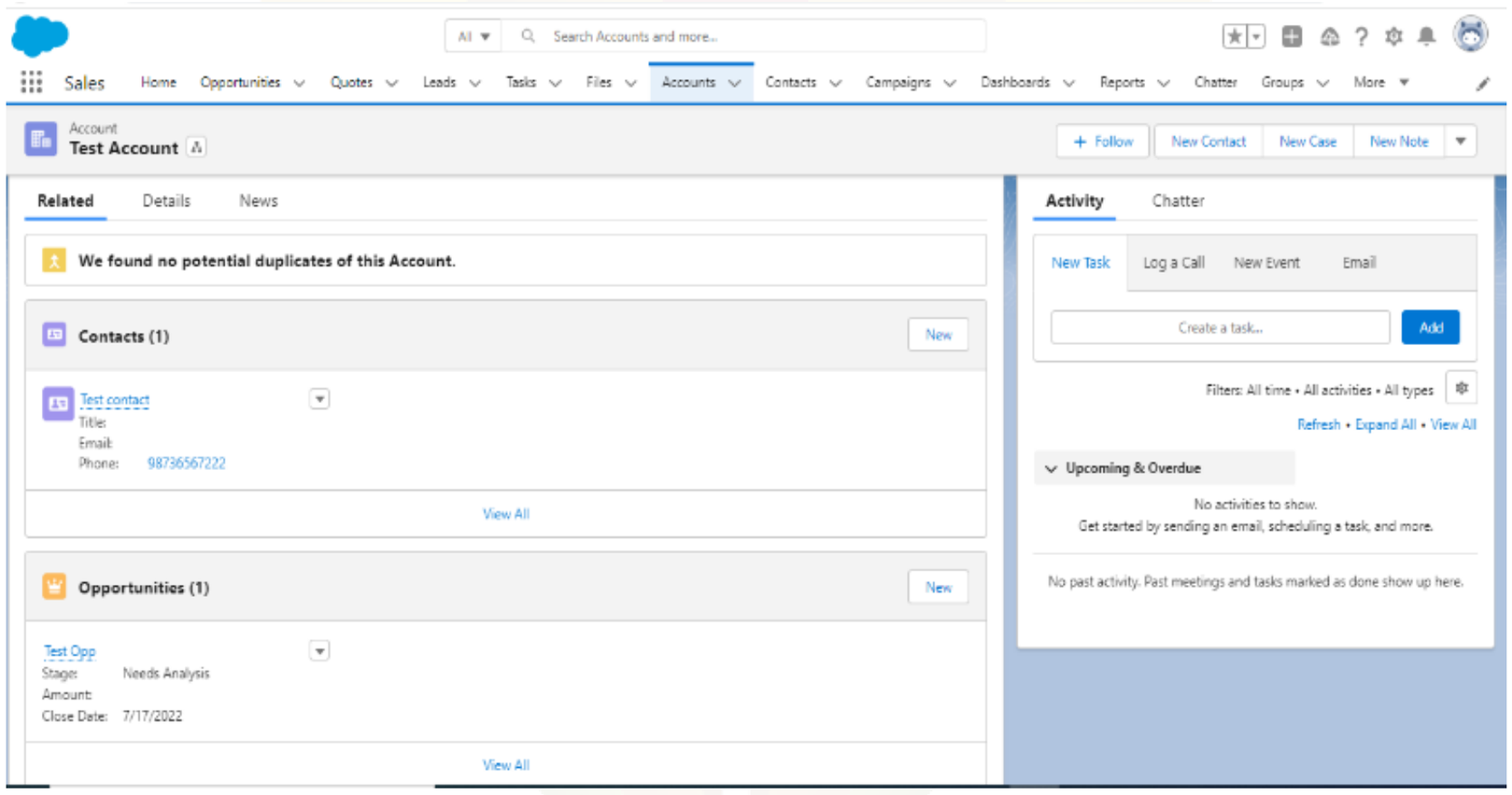
1. Create new Opportunity record as shown in below screenshot.



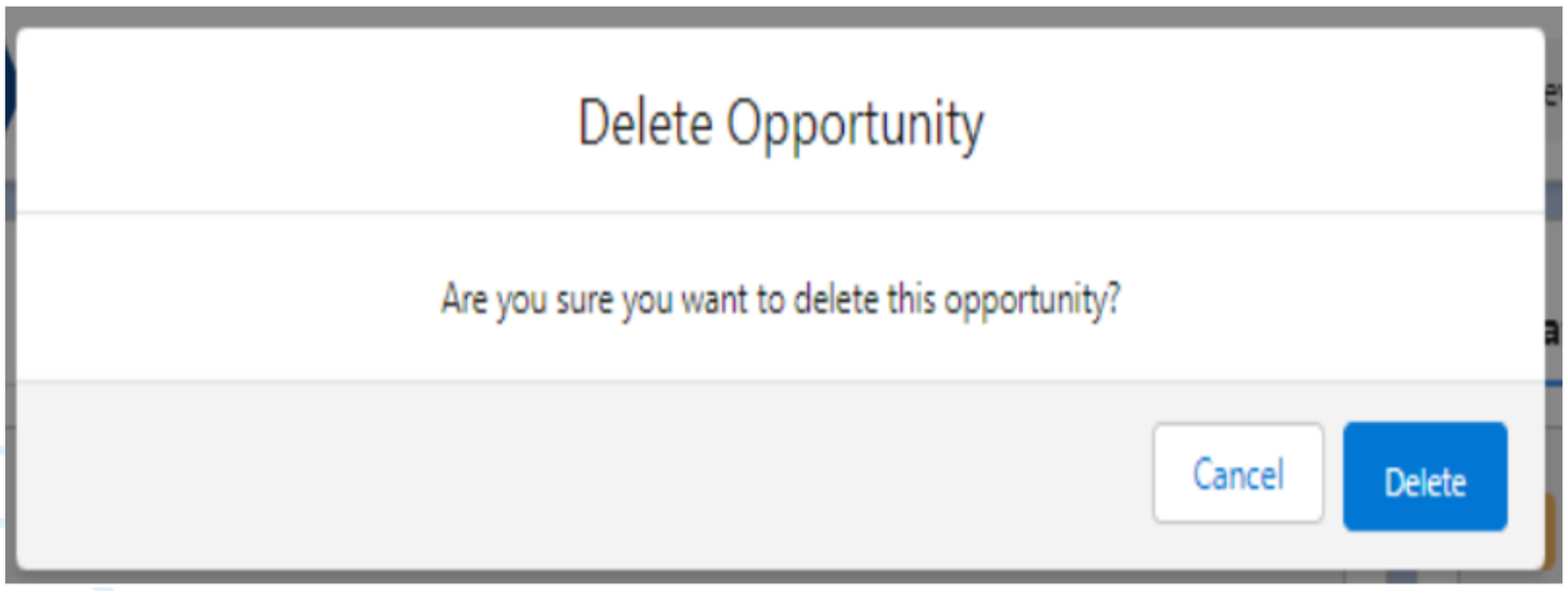
2. Create new Account and contact record related to opportunity.

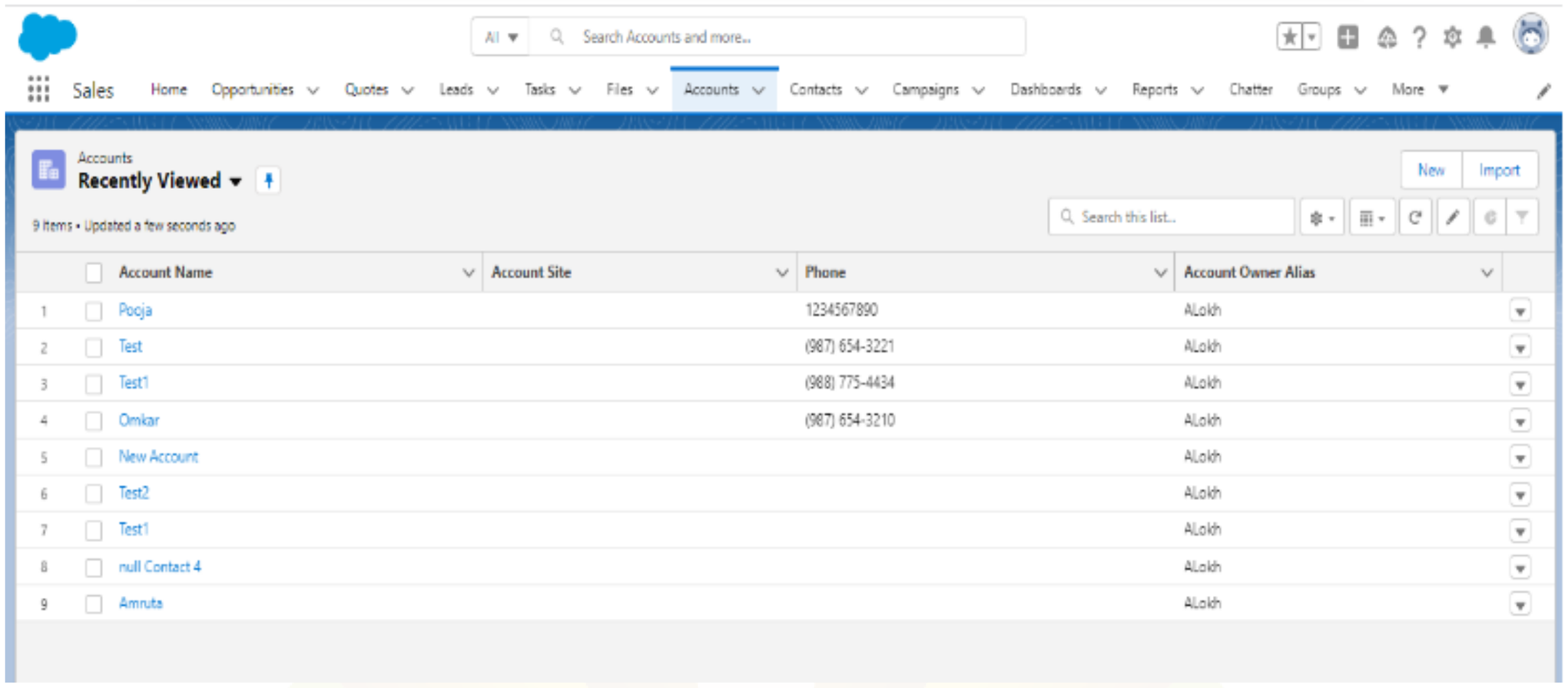


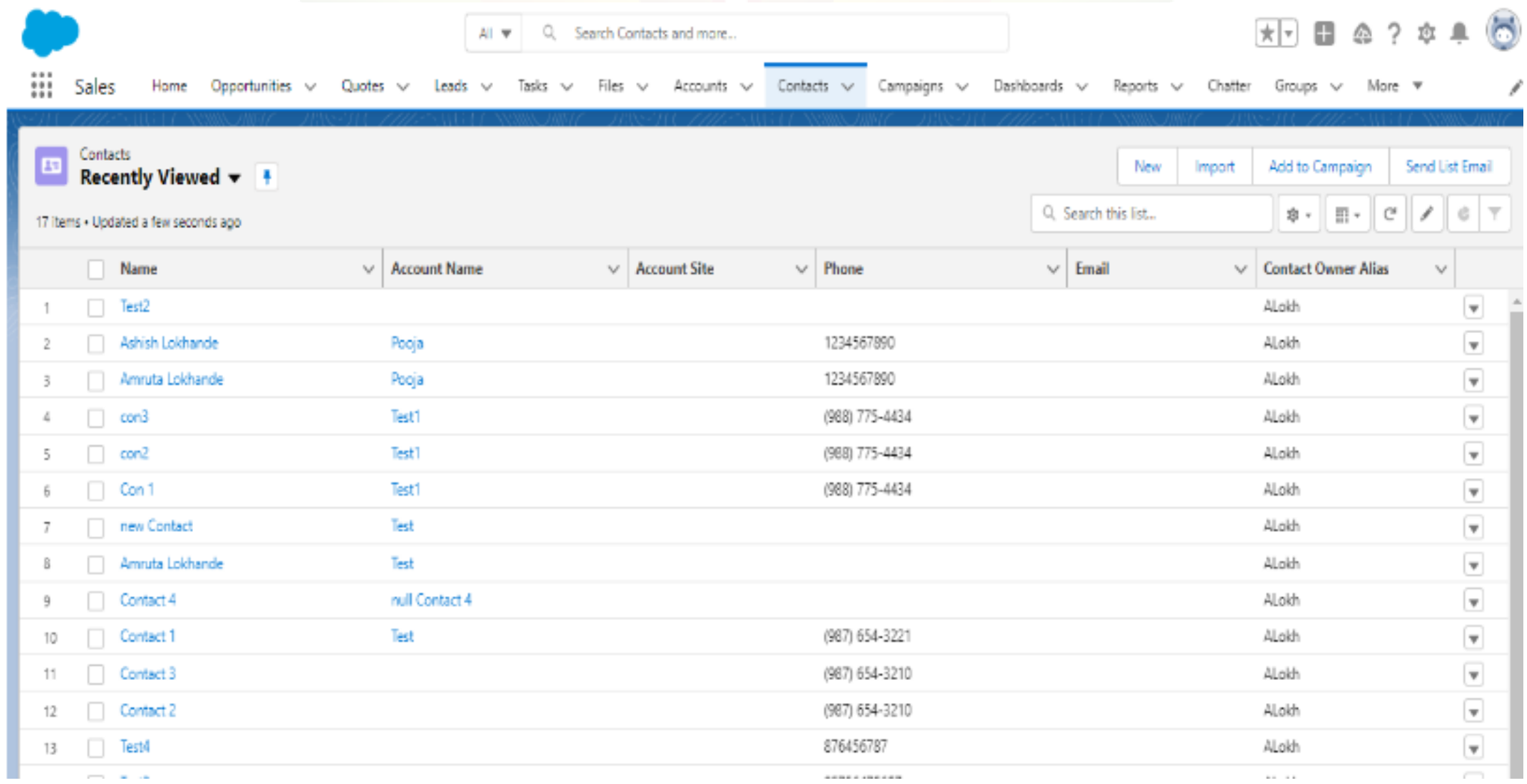




3.When you delete Opportunity corresponding Account and Contact will be deleted.







Example 40

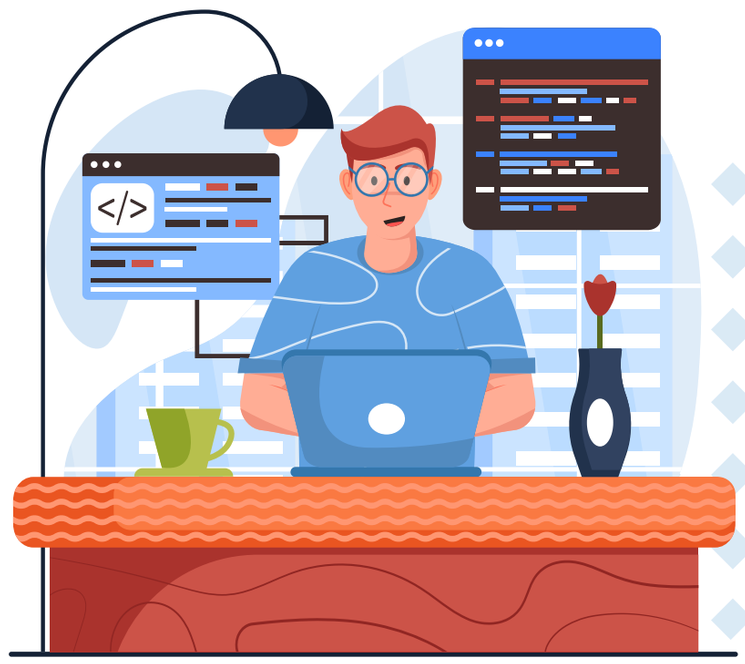
Write a trigger on Contact to Prevent the user to create duplicate Contact based on Phone if Phone number is already exist on related account in Salesforce

Trigger: RestrictDuplicatePhoneOnContactTrigger.apxt

```
trigger RestrictDuplicatePhoneOnContactTrigger on Contact
(before insert,before updat e) {

 if(trigger.isBefore){
    system.debug('Event before trigger');
     if(trigger.isInsert){
        duplicatePhoneOnContactCtrl.
        checkDuplicatePhone(Trigger.new);
     }
     else if(trigger.isUpdate){
       duplicatePhoneOnContactCtrl.
                          checkDuplicatePhone(Trigger.new);
     }
  }
      else if(trigger.isAfter){
            system.debug('trigger after event');

      }

}
```

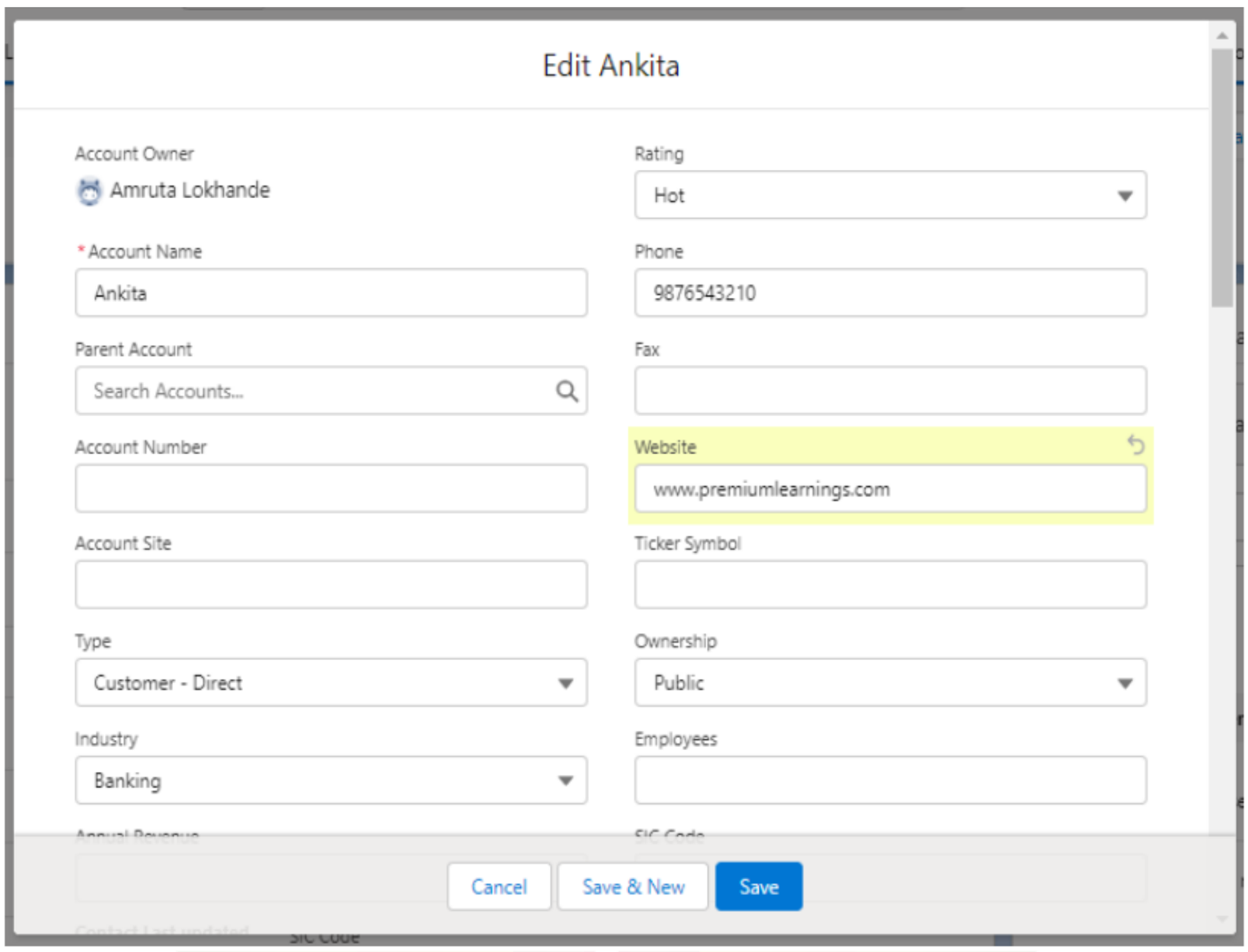
Class: duplicatePhoneOnContactCtrl.apxc

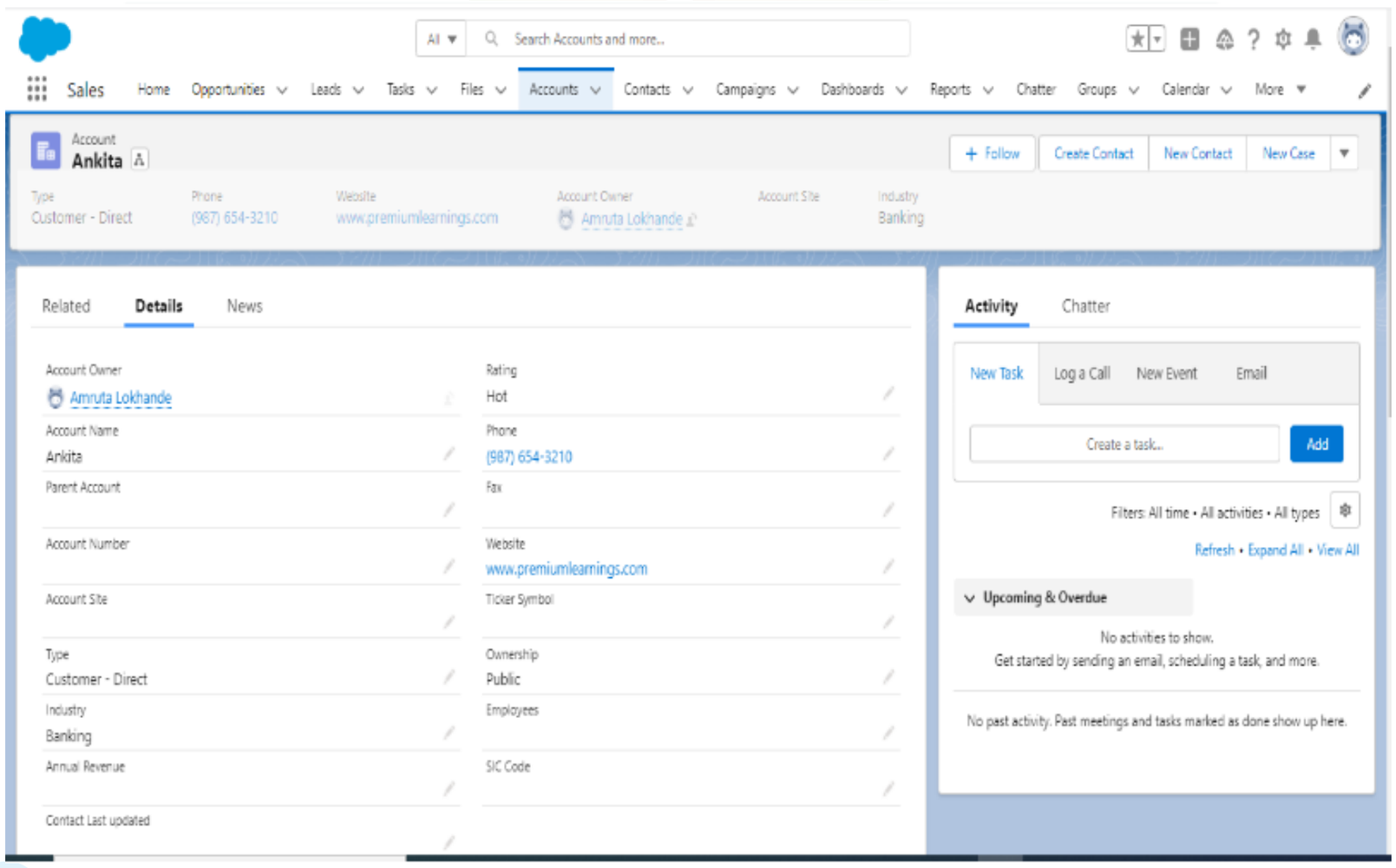
```
public class duplicatePhoneOnContactCtrl
{
 public static void checkDuplicatePhone(List<Contact>
                                                conObjItem)
  {

  Map<String,String> conMapItem=new Map<String,String>();
  set<String> accStr=new set<String>();

   for(Contact con:conObjItem){
      if(con.AccountId!=null)
         accStr.add(con.AccountId);
   }
   for(Contact con:[select Id,Name,Phone,AccountId from
                          Contact where AccountId=:accStr])
   {
      conMapItem.put(con.Phone,con.AccountId);
   }
  for(Contact con:conObjItem)
  {
      if(!conMapItem.isEmpty() &&
                        conMapItem.containskey(con.Phone))
      {
         con.addError('Do not allow duplicate Phone
               number, this phone number is alread exist
               on related account');
       }
  }
}
```



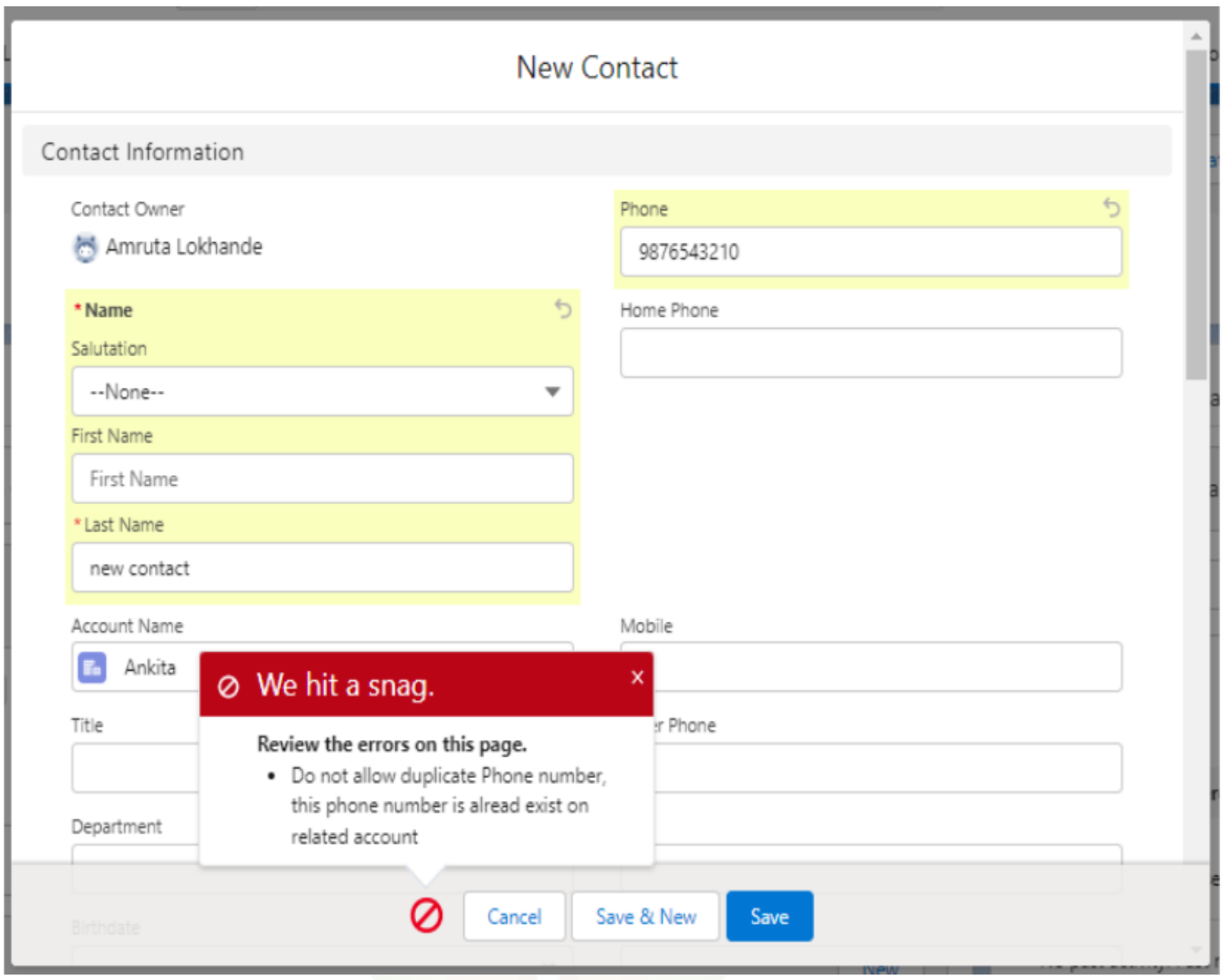
Test Case:

1. Create new Account record and must fill the phone field As shown in below screenshot.





2. try to create child contact record of Account with same phone number, you will get an error as shown in following screenshot.



Example 41

Trigger to Update the Custom Last Modified Date Field in the Parent Object When the Records in the Child Object is Updated Using Apex Trigger in Salesforce

Trigger: UpdateLastModifiedDateOnLr

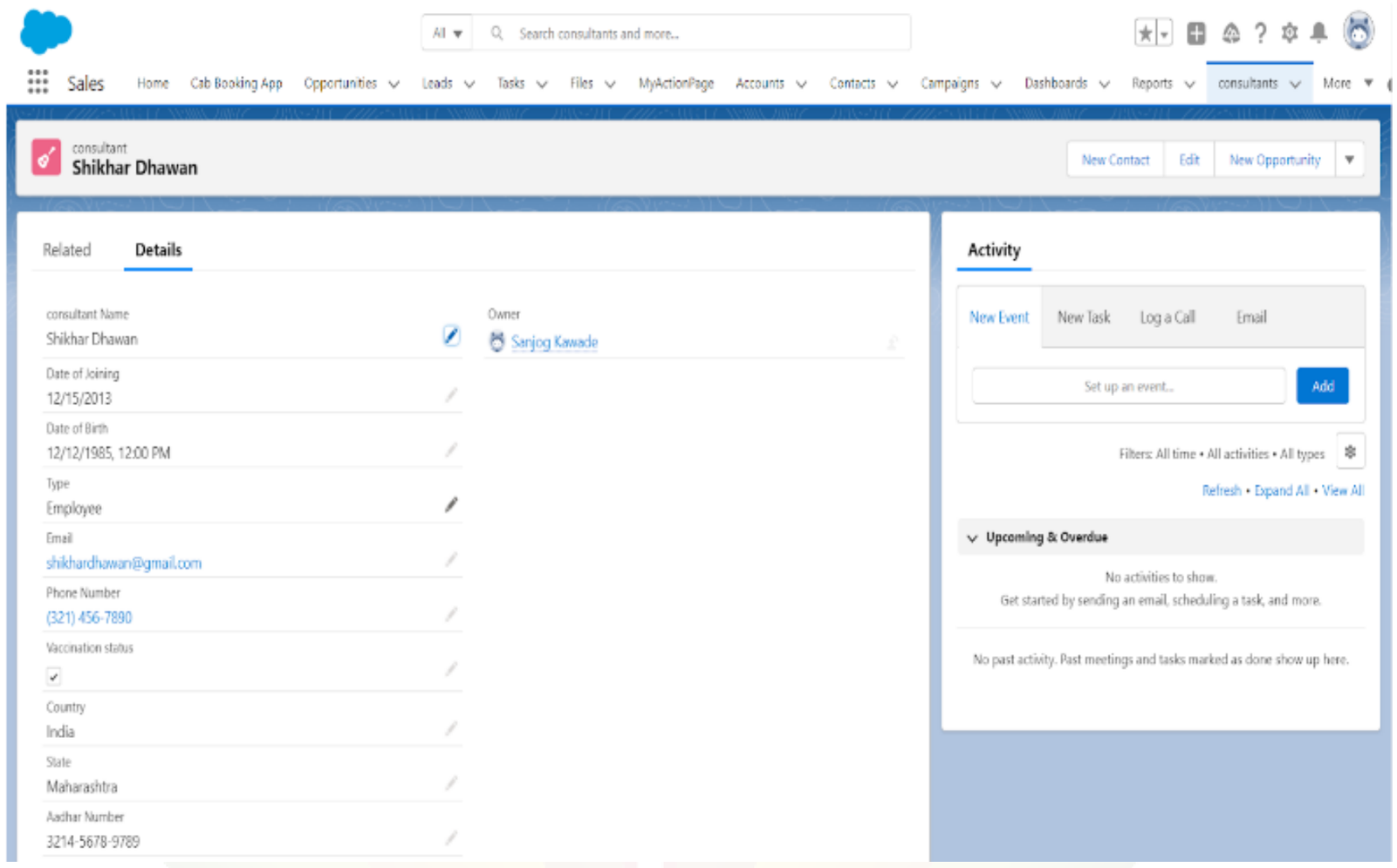
- Create a custom field on parent Object name as LastModifiedDate__c (Date/Time Type).
- Create a lookup relationship field on child Object

```
trigger UpdateLastModifiedDateOnLr on Leave_Request__c
(after insert, after update) {

 Map<Id,Datetime> idLastModifiedDateMap = new
                                          Map<Id,Datetime>();
 List<Datetime> childIdList = new List<Datetime>();
 Set<Id> setIdList = new Set<Id>();

if(trigger.isBefore)
{
  system.debug('trigger before event');
}
else if(trigger.isAfter){
        system.debug('trigger after event');

 if(trigger.isUpdate){
    for(Leave_Request__c temp : trigger.new)
    {
        childIdList.add(temp.LastModifiedDate);
    }
```

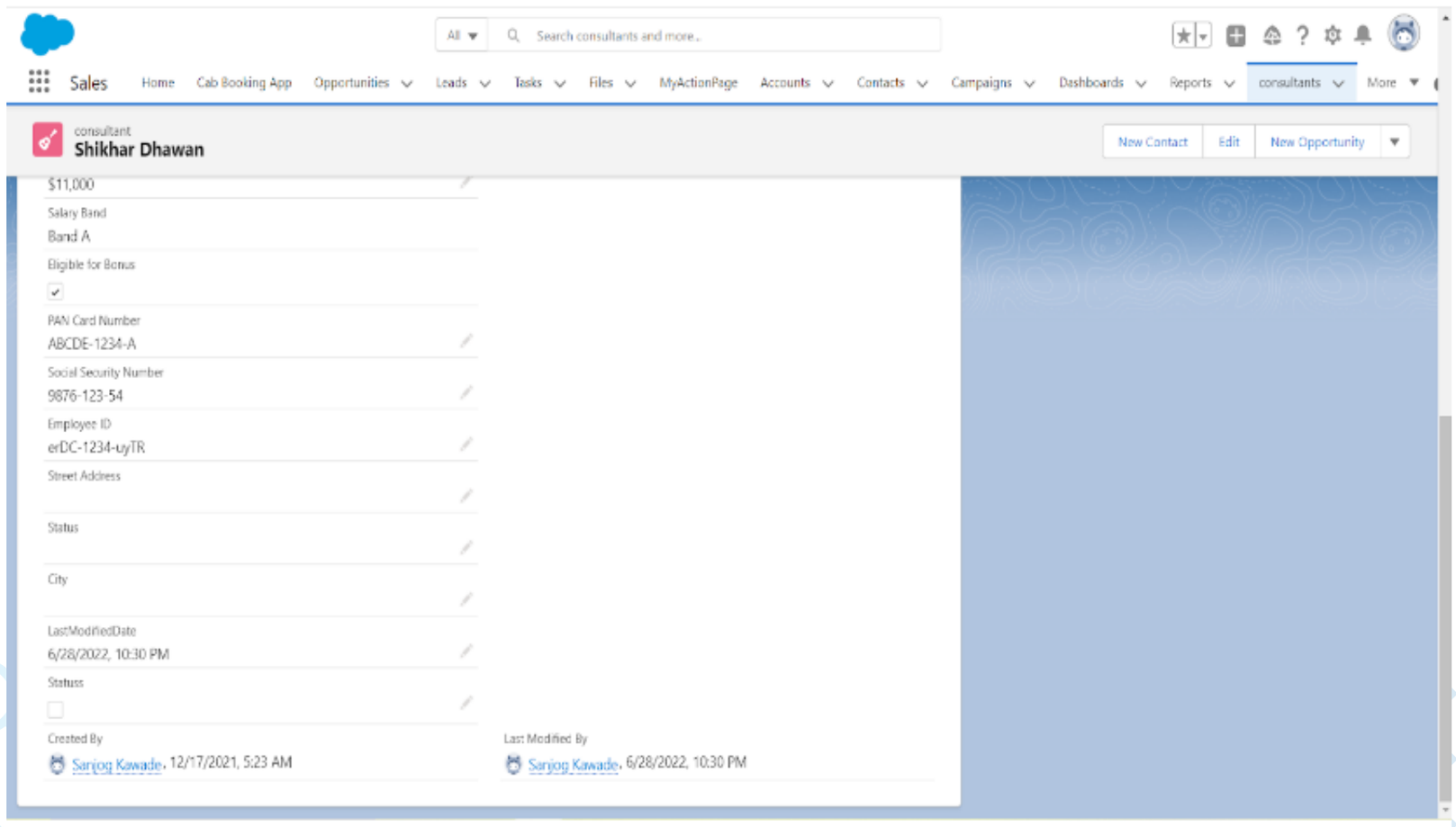
```
    system.debug('childIdList ' + childIdList);

    List<Consultant__c> prntList= [SELECT Id,
                        LastModifiedDate__c From Consultant__c];
    List<Consultant__c> lastDate = new List<Consultant__c>();

    for(Consultant__c p1:prntList){
        for(Leave_Request__c c1: trigger.new){
            system.debug('c1 ' + c1.LastModifiedDate);
            p1.LastModifiedDate__c=c1.LastModifiedDate;
            lastDate.add(p1);
        }
    }

       update prntList;
       system.debug('lastDate1### ' + lastDate);
     }
    }
}
```

Test Case:

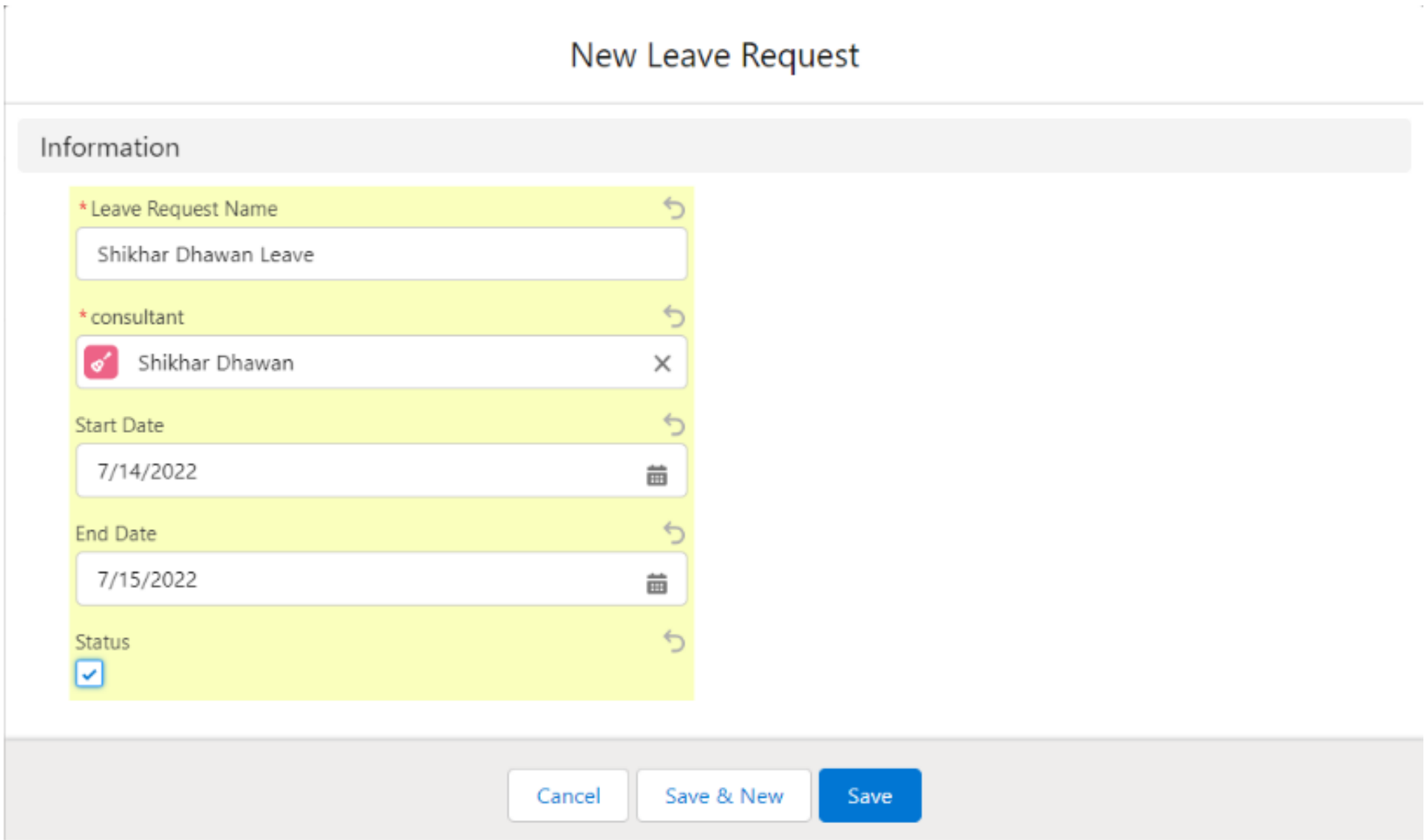
1. take any consultant record which is previously created (Last modified date should be less than today) as shown in below screenshot.

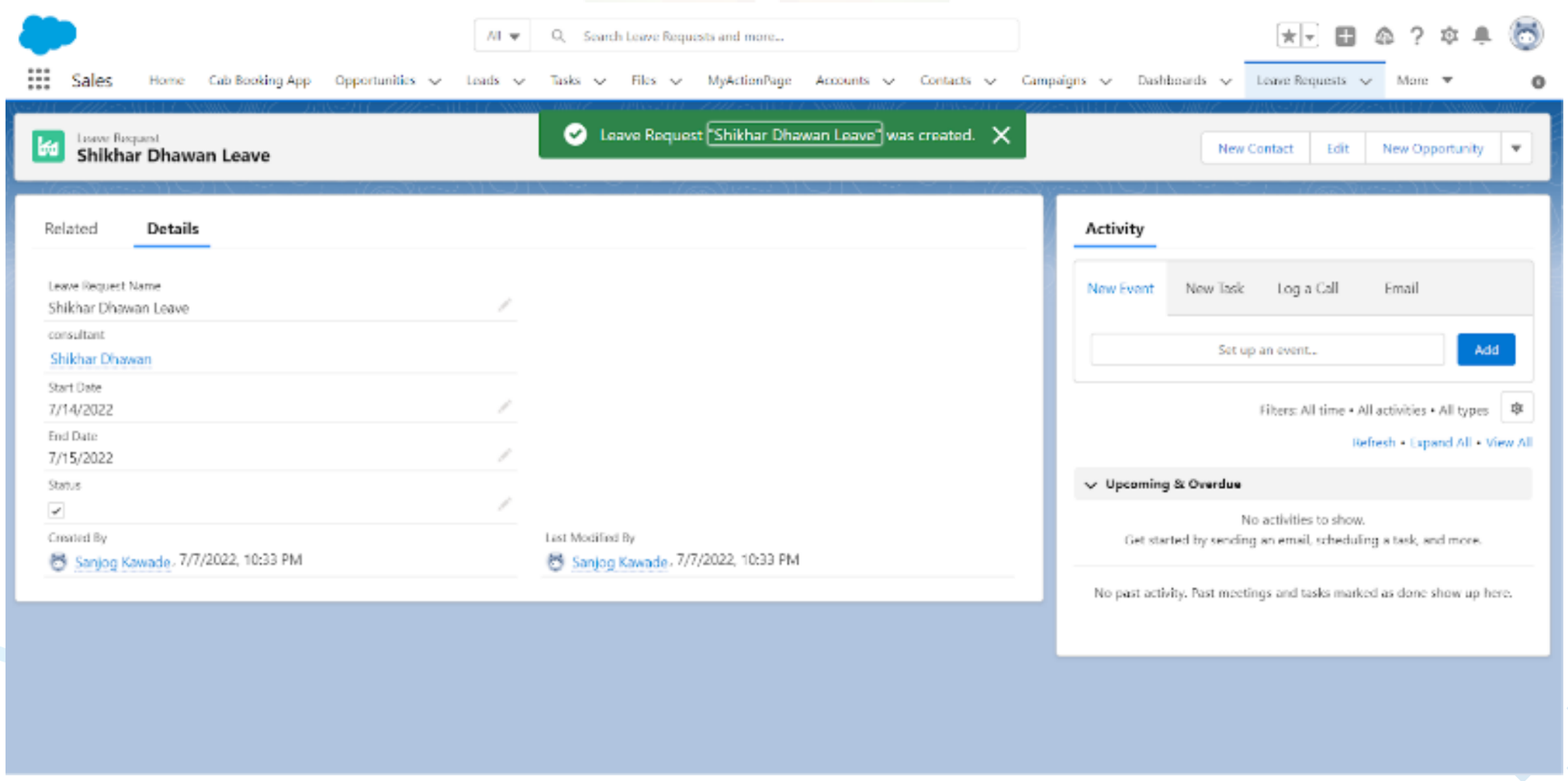


In below screenshot you can see the last modified date of record.

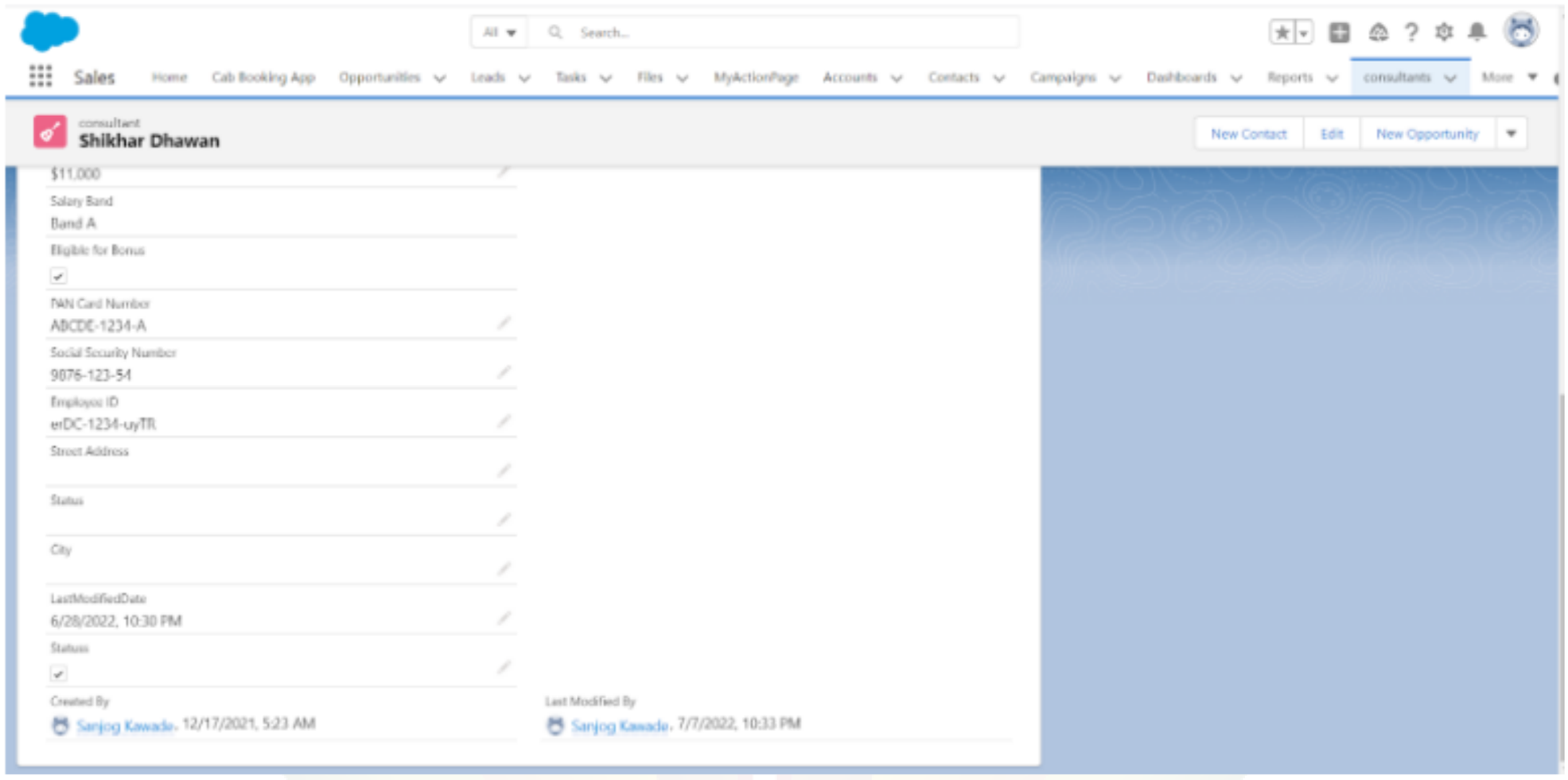


2. Now create new child leave_Request record related to the parent consultant as shown in following screenshot.





3. Now check the Custom Last Modified Date Field on the Parent Consultant Object will update When the Records in the Child Leave_Request Object is Updated.



Example 42

When we create the Account record, the Account Owner will be automatically added to Sales Repr field. When we update the Account owner of the record, then also the Sales Rep will be automatically updated.

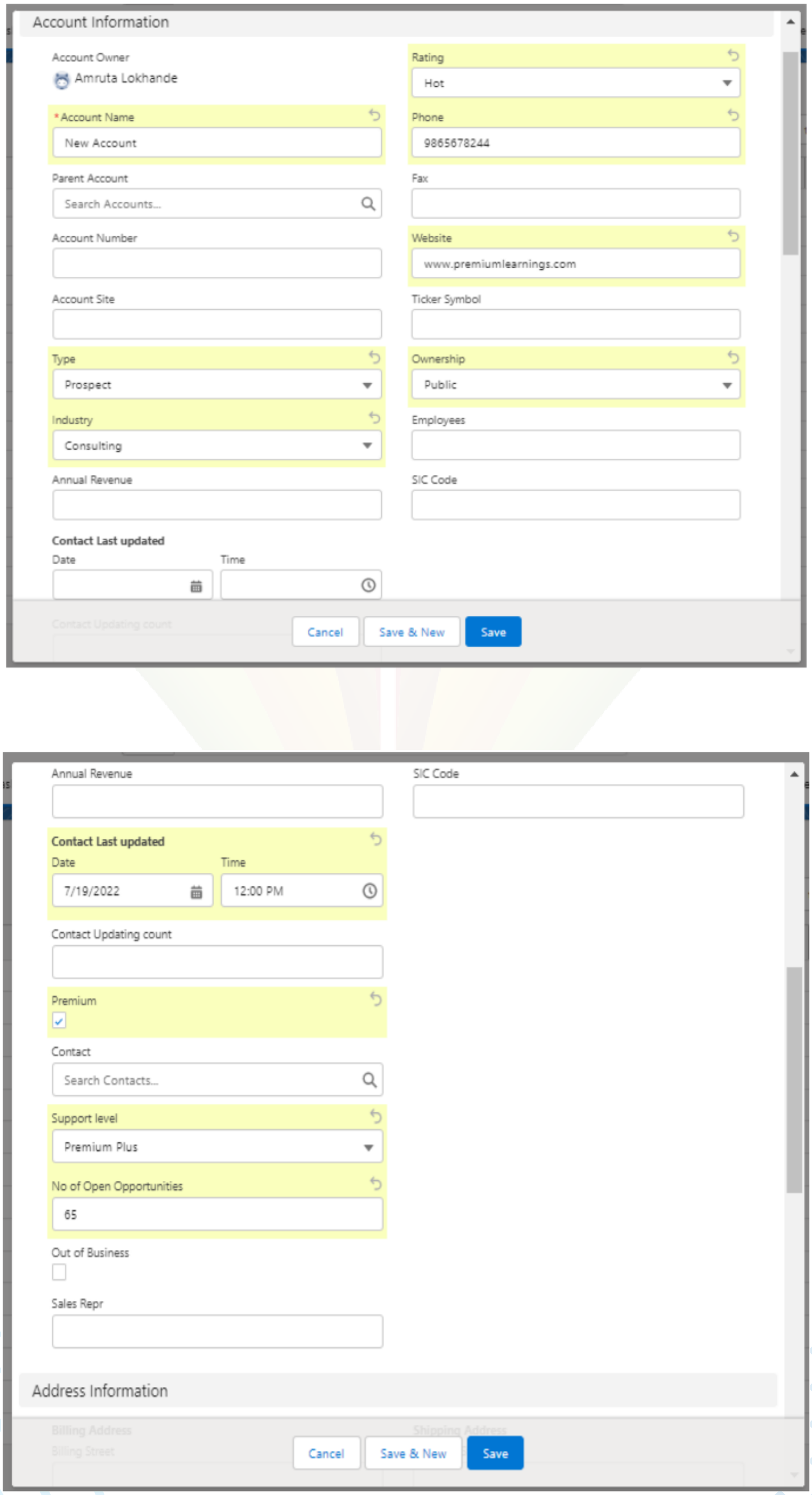
Trigger: UpdateSalesRep.apxt

- Create custom field on Account named Sales_Repr__c(Type Text).

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

 Set<Id>setAccOwner=new Set<Id>();
 for(Account Acc: trigger.new)
 {
   setAccOwner.add(Acc.OwnerId);
 }

 Map<Id,User> User_map = new Map<Id,User>([select Name
                      from User where id in:setAccOwner]);
 for(Account Acc: Trigger.new)
 {
   User usr=User_map.get(Acc.OwnerId);
   Acc.Sales_Repr__c=usr.Name;
 }
}
```

Test Case:

1. Create new Account record.



2. check Account Owner will be automatically added to Sales Repr field.