**Welcome** to module 7, Searching in NetSuite. In this module, we're going to look at how we can get more information through searching. We're either going to do that from the user interface or from our script.

So here, we want to learn about the NetSuite search architecture. We also want to know how we can create a saved search or a search in the NetSuite UI, and also pretty much the same thing in our script. So we want to see if we can execute a saved search or a search that's created in the UI from our script, and also compose our own search entirely from the script.

So that's what we're going to cover in this module. So in terms of searching, this is a quick comparison on a SQL query and comparing that with a search that you have in NetSuite. Now, NetSuite does not really allow us to directly access the database tables from our script, but this is just an example, in case you are already familiar with SQL, so that we can have that comparison.

So anyway, whenever you are searching in NetSuite, or in this example, we have a SQL query, the thing that you have to make sure we include would be the results that you expect out of that search. For example, in this SELECT statement, we can see that we are trying to return the date, the PO number, the vendor that is selling the item. We also have the PO total and also the vendor credit limit.

Now, aside from that, we also want to add in a condition so that it doesn't return all of the records. So the condition that we have here is the total should be greater than 5,000, and that the category of the vendor is equals the design. Now in this case, we're getting the data from two separate tables, so we have the purchase order table, and then we also have the vendor table.

Now in that case, we are going to make sure that we join those two tables together, so we have the vendor ID as our key. Now, comparing that with NetSuite, if you want to insert certain fields off of your search, you need to make sure that you nominate search columns. Now, search columns can be set either from the UI or from your script, and that's something that we are going to look at later.

Now, aside from search column, your conditions, similar to this WHERE clause-- we have our search filter, so that'll be your condition. And of course, if you want to get information from another record, you can perform a join. Now, going to the NetSuite UI, so if you want to create a saved search-- and by the way, when I say saved search, I mean that's a search that's created in the NetSuite UI. So we have two kinds of searches that we will be dealing with in this module, and that will be your saved search and your script search-- save search being in the UI, and script search being in the code.

Now, for your search columns, you can do that by going through the Result subtab of your saved search. I'm going to show you where you can find that in a bit, but basically, on the result subtab, you will be looking at the fields there. You can add all of the fields that is related to the record. So the fields that are native to the record will be listed at the top of your list, and at the bottom of your list, that's where you would find a join.

Now, the illustration is a little bit small, but I hope you can see it. But here, you can see that I have the word vendor and then a colon, and then I have credit limit. Now, whenever you see a colon in your search result or in your search criteria. That means that would be a join.

So the credit limit field is being accessed from our vendor records, so that's a vendor record field. If you don't see that colon, that means that field is native to the table or the record that you're trying to search, so date, number, name, amount-- that is part of your order search, and then credit limit is part of your vendor. Again, this is the search column. So you're going to see that again in the results subtab, and if you want to specify a search filter, you will be seeing that inside the search criteria or the criteria subtab. So if you hear the terminology result, that will also refer to search columns, and if you hear the terminology criteria, that would refer to your search filter.

Now, with your search filter, you have to include more information. Unlike with a search column where you just select the field, and that's that, with a search criteria or a search filter, you have three important elements. So the first element would be your filter or the field that you are trying to search for. So here, for the record, I'm trying to search for a criteria of type, and then aside from that, you also have an operator.

So your operator would depend on the field, so if you're dealing with a free form text, then you can have fields like Contains, Is Equal To, Not, things like that. And then here, in this case, this is a list of records fields, so the operator that we have is Is. And then we're comparing that to the third important part, which is the value. So the first part is your field, second would be your operator, and then the third would be your value, and it's always like that.

Now, there are certain fields that would allow you to have a second value, so this could be value 1, and you can have also value 2. Now, those fields would be if you are comparing dates. So if you have a date comparison, then you can have value 1 for date before and then value 2 for date after, and then your operator could be between. So that's something that you can have, but those are the three things that you would see.

So for example, here, I have mainline. That's my field, and then my operator is Is, and then my value is true, and so on. So I have an amount and also, as you can see here on our criteria, we're also comparing the value of category. And category is not native to our order, but is actually native to the vendor record. So we can see that this is a join.

So what we're going to do in this walkthrough is that we're going to create a saved search. So the saved search would be a search that is on the user interface. Later, we're going to access that search from our script and try to recreate that search directly from our script. So what we're going to search for here would be, we're looking for escalated cases where the assigned employee to that escalated case has support in their job title.

Now, what we want to learn here is that we want to learn how we can create and execute saved searches, specifically how we can add criteria or search filters to our search and also how we can perform a join. Now, regarding our goal here is that we want to return fields from our searched record, our support case search. We want to return the subject, incident date, status, assigned to, and also we want to perform a join so that we can return the department of the assigned employee and also the job title. So let's go ahead and do our walkthrough.

To start creating our search, we're going to our user interface, and there are several ways of creating a saved search. So one way is by going to Lists and then go to Search, Saved Searches, and then go to New. So again that's Lists, Search, Saved Searches, and New.

You can also go to Reports if you have access to that, so it really depends on your NetSuite setting. So if you're on the classic center, or if you're logged on as an administrator, then you have those options. But of course, if you're logged in using a different role, you might not even have Lists or Reports, or you might just have one of those. So anyway, on Reports, if you go to Reports, and then if you go to Saved Searches, you can go to All Saved Searches and the New. So that's very much the same, so Reports, Saved Searches, All Saved Searches, and New.

So let's go ahead and build our search. The first thing that you need to do if you're going to build a search is determine the record type of the data that you want to search for. In our case, we want to look for support cases, so here, I'm going to choose Case for my saved search, so let me click on that.

Then after that, it's going to ask me for the configuration of my search. Now, for my search, I'm going to-- just make sure you name it something that is descriptive of the search that you're trying to do. For example, in this case, I'm looking for escalated support cases, so that's exactly the name that I'm going to use-- Escalated Support Cases.

Now, for the ID, same naming convention, and again, this is something that you have to remind everybody to put in the proper ID for their searches. So underscore sdr underscore escalated underscore searches. And then here at the bottom, under the criteria subtab, this is where we're going to add all of our search criteria or search filter.

First off, we're looking-- well, actually, we only have two criteria here, and then the criteria that we're looking for would be, first, the status. If I scroll down here, I'm going to select the status of my support case, and I only want escalated. Now, if you are going to define a search filter, there are three things that you need to set. First off, the field that you want to search for, in this case Status.

The next thing would be your operator, so for a list or record field, you only have two operators, and that will be Any Of and None Of. You can think of Any Of as a multiselect Equal To, and then None Of would be a multiselect Not Equal To. So for example, if I want to look for escalated and reopened cases, I can select both of them, so I can hold a Shift or a Control key and just select the values that I need. Now in this case, I'm only interested in escalated cases, so that's the one I'm going to select. So click Set here, and then that's our first criteria.

And then the other criteria that we have is that we're looking for support cases where the assigned employee has the word Support in their job title. So for that one, we don't really have job title for a support case because why would a support case have a job title. Employees have a job title, so that means that we need to join to my employee record so that we can look for that job title.

Now, what we're going to do here is that, on our search, if you look at the list, the first few fields that you're going to see would be fields that is related to the record that you're searching for. So in this case, these are the fields of my support case record. Now, if I scroll all the way down, you would see here that there would be entries that has three dots or ellipses, so that means that this would be a join to this particular record. For example, for my activities, I can join to an activity record, a company record, a contact record, a customer record.

Now, in our case, we're looking for employee records or employee values, so I need to join my employee records. I'm going to use that, so join to my employee. And then here, it's going to give me the same set of filters, so instead of giving me a list of fields related to my case record, it'll give me a list of fields related to my employee record because I'm joining to my employee.

So on my employee, I'm looking for a job title. Let me jump over to the Job Title here. So after I click on Job Title, I have my search operator now. So since I'm looking for values on a free form text, you can see here that I have other operators aside from Any Of and None Of. Later, we're going to see what kind of operator is available to us once we start building our search.

So anyway, here in our operator list-- let's see here, I'm looking for something that contains, and the key word I'm looking for is support. That's it. So when I click on Set, as you can see here, like what we talked about, whenever you see a colon in your search criteria or search filter and your search result, what that tells you is that this would be a join. So employee would be the record, and we're joining to a job title.

Now, moving on here to the results, so that we can display the information that we need-- now by default, you will be given the common things that you have for a case search. Now, we don't really need all of these, so I'm going to remove everything, and what I'm going to do is just add in the things that we need. So I'll just click on Add Multiple, so I can add several records at a time.

So we're looking for my subject, so let's start with that. So we have subject. We also have status, since we're already there.

We're also looking for our date, so incident date. And what else do we have-- the assigned employee, so we want to know who this case is assigned to. So let me click on Add, and that's it. So we added all of our four fields.

Now, we also need to add fields from our employee records, so again, we are going to perform a join. So if I open my list, similar to my search criteria, if I scroll all the way down, I should see some entries here that has three dots or an ellipsis. So we also want to join to our employee fields, so here, if I join to our employee record, I can select one at a time, or I can also add multiple, so that's also something I can do.

So here, we're looking for the department. It starts with department here. Let's look for department and also for my job title, job title. There you go, then click on Add. By the way, I'm holding the Control key when I'm selecting the options so that I can select multiple values at the same time.

So here, you can see that I have two joins. I have a join to my employee record for the department, then another join for my job title, and that's pretty much it. So for me to be able to see my search, I can just click on Save and Run. So when I click on Save and Run, it'll save our saved search and then display the information.

So we have four different cases, so we have two insulation cases, one enhancement request for denying input form, and then we have a shipping cost calculation issue. Now as you can see, it's assigned to two different employees, and if I look at the job title, looks like we do have the right search because the job title has support, and the status is escalated. So whenever you're creating a saved search, I do recommend that you add in the criteria fields that you're using so that you can verify, or at least visually verify, if your search is correct because if you don't, you might be thinking that your search was correct when, in fact, you might have made a mistake creating the criteria. By at least adding the fields that you use in the criteria, you can visually verify.

Now, going back to our search, let me click on Edit Search here, and one more thing that I want to mention before we head on over to the next topic is the Preview button. Now, there are two ways of you getting the search results from your search, and that would be by using Save and Run or Preview. The difference between, Save and Run and Preview is that, if you click on Preview, it'll show you the results without saving your research.

Now, if you're using somebody else's search, like maybe you're using that as a base search, and then you've added a few modifications to that search, maybe added a few search result fields, or maybe you've added a few criteria, then instead of saving that search or instead of clicking on Save and Run, which would modify or finalize the changes that you've made, and since that search doesn't really belong to you or you're just going off of somebody else's search, you can click on Preview so you can see the results that you have. So again, those are two options that you can use. If you don't want to save the changes that you've made, you can just click on Preview, but if you do want to save the changes that you've made, you can click on Save and Run. Be careful if you are using Preview, though, because since you're not saving your search, and if your session expires, you might potentially lose all of the changes that you've made on your search.

Now, if you're confident of the changes that you've made on the search, I would recommend to mostly use Save and Run. Now if you're just still testing, maybe you made some modifications, and then you're not really sure of the change, you can click on Preview. But I would recommend that you keep using Save and Run to make sure that all of the changes that you've made would be persistent on the database.

So yeah, that's pretty much it. So that's how you would build a saved search in NetSuite. Just make sure you in a criteria. Then after adding your criteria, specify results, and then after that, save and execute your search.

Now that we know how to create a saved search, we want to know how we can apply that to our script. So moving on here, the fun part-- we are actually going to start scripting our search. Now, to script our search, we need to make sure we get the proper IDs. And we already know where to get the IDs, and that will be from our SuiteScript records browser.

So to get an ID, you need to go to the record, and the one thing that you have to remember if you're searching is that make sure that you're getting the ID from the right section. There are separate IDs for your search join, your search filter, and also for your search column. Now, for the most part, especially for your search filter and your search column, the fields are the same. I would say about 80% to 90% of the time, the search fields would be the same, but they're not always the same. So instead of guessing what the ID is, make sure that you go to the records browser.

You would see the search join, search filter, and search column section by scrolling all the way down on your list. So after your fields, so that'll be the top part, you have your sublist, and then below that, you have your join search filter and then search columns. And make sure, again, you're looking at the right section. Otherwise, you might accidentally use the wrong ID.

What we're going to do in this walkthrough is that we are going to access the saved search that we've created on our script. So instead of recreating our search, we're just going to grab an existing saved search in the NetSuite account and then execute that so that we can get the results. We're also going to look at how we can compose our own search from our script.

So after trying to load that search, we're going to try to recreate it with the same criteria. So we're still going to search for support cases. We're still looking for escalated cases where the assigned employee has support in their job title, and we're also still going to return the same set of values, a subject, incident date status, and the assigned employee, as well as the department and the job title of the assigned employee.

Now, the things that we want to learn here is that we want to learn how we can execute a saved search from our script, and also we want to learn how we can manually create our search on the script, because those are two things that you might potentially do. If you are going to reuse an existing same search, maybe a search that was created by somebody else, then you want to know how to load that. And also, if you are creating your own search specifically for the script, then that's something that we also want to learn.

To start building our search, let's go ahead and create our scripts. Let me just close this. We don't need that anymore, and let me create a new script file.

Now, for this one, the search that we're going to do would be executed on our debugger console. Now, for the script type, since I will be using this later as a scheduled script, I'll just use the scheduled script template. So for this one, let me just change this to sdr underscore ss, for scheduled script. Again, don't worry about that at this point because we are going to create that in the next module.

And then underscore. I'll just call this escalated underscore cases. Also, what I need here would be my search module. Let me create that and then click on Finish, and that's it. Change this to SuiteScript 2.0, and that's it.

So I've already created my search. And another thing that I have to do here is that, whenever I am performing a search on the debugger or performing a debug session on the debugger, we have to change our defined statement to require, so we're going to look at that in a bit. But anyway, let's go ahead and build our search.

Now, first off, what we need to do is we need to get or grab our search from the NetSuite user interface. You can do that by using the load method of your search module. So here, I'll just call this Case Search, and then I have my search module. And then on my search module, I can just use load.

And then for my load, I only have one property that I need to set, which would be the internal ID of the search that I want to load in my script. So here, I'm going to my saved search, and then just copy that ID. So it has to be the complete ID with the prefix, which is custom search. If I go back to my IDE, I'll just pass that there, and that's that.

So as you can see, it's very simple to load an existing saved search. If you already have a saved search that's on the system, and you're happy with the results that you're going to get on that saved search, then just go ahead and reuse that. It's easier that way.

Now, another thing that we're going to do here is that we're going to run the search so that we can look at the results. Now, we're not going to parse the results just yet. We're going to do that in another walkthrough, but here, we just want to run the search so that we can inspect the search results.

So here, I'm going to create my search results objects, so search results. I have case search, and then execute that case search. Now, whenever I run my search, there are two functions or two kinds of search results that I can get.

One, I can get a page range, which gives me the search result in sets. So if I use a page range, I can say I can get a maximum of 1,000 records per execution. If I have 10,000 results, what I can do is that I can just look through my page range to get 1,000 results each execution.

Remember that if you are going to use a page range, you can only get a maximum of 1,000 records. You can make that shorter. Now, in our case, we only have four results because we've already seen that in the NetSuite user interface, so I'm just going to use a lower value. But yeah, you can use up to 1,000 records.

Another search result object that you can get is the result set. Now, result set is a little bit different because that is something that you would use to loop through all of the results. Now, instead of getting a set of results, you're going to get one particular result if you're going to use result set.

Now, with the result set, if you want to loop through all the values inside a result set, you have to use the Each method. So the syntax is a little bit different because you have to loop through the results, unlike here, where you get all of the results in bulk. Now, if you are going to use the result set, the limitation is different. So instead of having 1,000 records, you actually would be using what we call governance units, so for each method, you have 10 units per execution.

Now we haven't talked about governance units yet, but we are going to talk about that in a future module. The governance units or SuiteScript Governance is basically a way for NetSuite to implement a check so that there would be no script that would be using up all of the resources for one particular server. Again, we are going to talk about that in detail, so you don't have to worry about that now. Just do know that there are two results that you can get when performing a search. You can either use a page range, which would give you a set of results, or you can get a result set where you can use that to loop through all of the results.

Moving on here, so for this particular search, we're going to use a pitch range. So for that, I'm going to use the Get Range method. So for that Get Range method, I can get that result by using the Run method, and then for that Get Range, we're just going to say, what are the records that I want to get.

So here, I have two properties, so that'll be where I'm going to start, and that'll be my starting index of 0. And then my end index would be how many records do I want to get. Now, in this case, I only want to get 10 records, so my INT index would be, say, 9. Again, I'm just using a small value, because we only have four results. We know that we have four results because we have already executed our search.

And that is pretty much it. Again, we haven't really done anything to our search, but we are going to parse that later. So if you want to load a search, just go to your search module, and then use the load method, pass in the ID of the search that you want to get, and if you want to run the search, just use that search object. Use the Run method, and then from that Run method, use your Get Range method.

So that is pretty much it. Let me save this. And again, instead of uploading to the server, I'm going to just copy my codes because I'm going to run this on my debugger. So let me go here and then open my script debugger. I'll just use this other tab here.

I'm already logged into the debugger, so I can use that. So whenever I'm debugging, I just need to make sure that I use the right API version. Otherwise, it's not going to work.

Now, instead of debugging an existing script-- which in this case, it's not really existing yet because I haven't created a script record yet. But I'm going to put it here, paste our code inside our script, and there are a few modifications that I need to make. So remember, when you are debugging ad hoc scripts, you can't use the DEFINE statement. What you need to use is the REQUIRE statement, which syntax is pretty much the same. So it's not exactly the same, but it's very similar.

So first thing that I need to change is I need to change DEFINE to REQUIRE, so you can see, it's very easy. Definitions for your dependencies, like any modules that you've added, would be the same. You still have that array, and you still have that callback function, which would be executing your search.

Another thing that you have to do here is that you have to remove your function definition, so here, what we're doing is that we're creating a function. You actually don't need to remove that function, but you have to call that function to execute it. So instead of having a function definition here, you can remove that, or you can just put in Execute, something like that, so that it would actually execute your function.

Now, for us, we don't really want that extra information there, so I'll just remove that. I really don't need that. Then also remove this at the bottom, and also the Execute call that I've added.

I'll just remove that, and I also need to remove my Return statement. So I really don't need that Return statement. I can leave it there, but it's cleaner if we remove it. And that's pretty much it.

That's our search, so to debug our search, I'll just click on debug script. So if you remember previously, when we were debugging our server side script, we click on Debug Existing because we already deployed that. But since we're debugging hoc scripts, I'll just click on Debug Script here.

Then after that, it's going to go through our debugging session, and I'll just say that-- I'll put a breakpoint here. And by the way, if you are debugging your script ad hoc, make sure that you add in a breakpoint because if you don't add in a breakpoint, it'll just treat your REQUIRE statement as one line. So instead of going to the next line, it's going to just complete everything. So make sure you add in a breakpoint.

So anyway, let's move on to our debugging session here, and we have our search results. If I go to my local variables, you can see here that I have my case search. My search result is still undefined, meaning it's still empty. We haven't really gotten any information about that yet. So let me just go to the next line so that we can execute that.

If I click on that, you can see here, if I go back to my local variables, it looks like the last line that I have on my script-- so I was not able to verify if I got the right results. Now, one thing that you can do-- let me click and switch back to editor here-- is that, if this is something that you need to inspect-- so I need to inspect my search results object. But since this is the last line, once this line gets executed, then it'll complete your debugging session. What you can do is that you can add in a stopper line, and that stopper line can be anything.

For example, if I add in var x equals 0, this could be a valid stopper line. So any valid JavaScript statement that would not give you an error message can be used so that when you're debugging, it'll stop on this line instead of ending the execution on your last line. So if I try to debug the script again, let me add in a breakpoint here, and then I'll just click on Continue. Now, if I go to my local variables, I would be able to inspect my search results, and as expected we have four results.

So the first result that we have here-- let's see if we open my values, so the title or the subject of my search is Shipping Cost Calculation Issue. And then the next one should be an Enhancement, so if I open that, you can see an enhancement at a time input form. And then the third and the fourth should be the same, so these are the two installation issues-- so one installation issue there, and then another installation issue. Now, the results that you're going to get here would depend on your saved search, so if you remove any search columns or add any search columns, it would also appear here because you're not creating your search. You're just loading an existing search on the system.

So that's it. So that's how you would create a search. And by the way, one other thing that you might want to think about whenever you're using a stopper line, because once you use this-- if I copy all of my code and then paste that on my script file-- so if I copy that, go back to here, and then just say, you know what, that's OK. So that's working. I'll paste that.

I might accidentally paste this back, and when I do paste this back, what's going to happen is that might potentially affect my code, especially if I'm using a variable x for anything. So instead of using var x equal to 0 as your stopper line, you can do something that would really show that this is a stopper line, and it's not part of my code. So for example, I can just say stop and maybe just say "this is a stopper line--" something that would stand out so that you don't accidentally copy that back to your code.

Anyway, so that's it. So that's it for my definition. As you can see, I was able to easily get the information on my search. Let me just change the formatting of this a little bit. Now that we know how to do that, the next thing that we want to learn is how can I now create my own search, so that's what we're going to do next.

So if I want to create my own search-- first thing, I want to comment this out because I don't need that anymore. For this one, I'm still going to use that, so I'll put that below here, maybe format that a little bit. And then if I want to use my search, I can create my search using the Create method of my search module. So it's very similar. So I have case search, and then I have search.create.

Now, if I am going to create my own search, then I need to specify the things that I've specified in my saved search. Now, first thing I've specified, if you remember, is we chose our type. So for this one, let's see here, I have my type property. Doesn't look like it's loading the type, so I'll just type that in. So search that type, and then I'm looking for a support case.

And that's what I'm looking for. Now, if you want to know more information about the different types, again, you can always go to the definition of your search module. So you'd be able to see all of the properties there.

So after specifying the type, the next thing that I need to specify would be my search filters or my search criteria. So I can set that using the filters. So this would be an array because I need to specify multiple filters sometimes. There you go, so that's my filter. And then after creating my filter, or at least creating my filter array, the next thing that I need to do is create the actual filter.

If you want to create the filter, there is a method that you can use inside your search object, which is the Create Filter method. So with a Create Filter method, it's very similar to what you've done in the saved search, where you're going to select the field that you're searching for and then the operator, and then after the operator, you have the value that you want to search for. Now, in our case, if I am looking for, say, the status-- so Name would be the name of the field that I'm looking for, so in this case, I'm looking for status.

If you're not sure about the search field that you're going to look for, then always go to the records browser for information. So if I go back here and say, I'm not really sure of the status, then I can just go to the records browser. I'm looking for a case search. Let's go to c here, and whoops, refresh that. Looks like that didn't load properly.

There you go. If I go to my case record definition, if I scroll down, I have three sections over here, with these sections for my search. So one section for my join, it'll tell me these are the records that I can join to. If you remember our saved search, these are things that we saw-- for example, contact, customer, employee. So employee, that's something that we were joining to.

We also have a list of search filters, so these are the IDs that you need to use if you're creating a search filter. And then if I scroll down, I have a list of IDs that I can use for creating a search column. Now, for the most part, the IDs for your search filter and the search columns are the same, but they're not always the same. For example, if I look at the Assign To Employee, I can see that my ID is assigned. If I scroll down here, at search columns, you can see that the Assign To is also assigned.

It's mostly the same. I would say about 80% to 90% of the time, it's the same, but it's not always the same. So make sure that you're looking at the right section if you're getting the ID. So for my filter, I'm using status, so as I can see here, the ID of my status is a status.

Now, another thing here-- so going back to my search here, status-- is my operator. Let me just get my operator property here. Now, for my operator, I have an enum value that I can use. If you remember our search-- so here, search.operator, so I have an enum value, and then we use Any Of operator for our search. Now, you might be thinking, how would I know what operator I'm going to use if I did not create a saved search? Because in our saved search, we know that it's Any Of because we actually used Any Of in our saved search.

Now, for this one, if you want to know the information about the operator that you can use for a particular field, you can go to the Help Center. Open the Help Center here, and just look for search operators. So there's an article called "Search Operators" which would be very helpful for us. If I open that "Search Operators" article, you can see here that it gives me the type of the field that I can use and the different operators I can use for that field type.

For example, if I'm dealing with a list or record field similar to our status, I can see here that I can use Any Of, and if I scroll down here, I can use None Of, and those are basically the only two fields that I can use. If I'm going to search for, say, free form text, for example, for our job title, I can see here that I can use any. I can use Between, Does Not Contain, Does Not Start With, and so on. Again, you can just use this as a reference, so this is a very good tool that you can use as a reference if you're starting out in SuiteScript development. So just look for "Search Operators."

So going back here, we have our operator Any Of, and then for the values, for this one, this would be a List Search. So I need to use the internal ID. Remember, if I'm using either a record or list, I would prefer to use the internal ID because the risk of me making a mistake and typing the value would be less.

But in this case, if you're dealing with a search, you really have to use the internal ID. You can't use text. In other field types, for example, a free form text, you can use a text. But if you're dealing with a list or record field, you always have to use an ID.

Now, to get the internal ID of my support cases, that's actually a list that I can go to. So for example here, if I go to-- I believe it is Setup. Then on Setup, I can go to Support, and then under Support, I have Case Statuses. So that's Setup, Support, Case Statuses.

Let's open that in a new subtab, move that right over here, and as you can see, these are the different statuses I have. I have Not Started, In Progress, Escalated, Closed. Now, in my case, I am looking for escalated cases, so I'll just use the 3 as the internal ID. So I have 3 there, and that's it.

Now, if I'm looking for multiple values-- let me go back to my list here, I'm looking for escalated and reopened. I can use an array which contains those two values, 3 and 4. So going back to my search here, I can just add that array definition and say I'm looking for escalated and open cases.

I can also add that in and say 3 is equals escalated, and then 4 is equals to reopen. So that's a guide that I can use, but in our case, we don't really need a reopen case. I only am interested in escalated cases, so that's it.

So that's our first search result. The second search result-- I'm just going to copy this to make it easier. There you go.

So next thing I'm looking for would be my job title. So for my job title, as we've seen in our exercise, the field ID is Title, and then Title can also be your subject. To distinguish the job title from the subject, I need to add in a join.

I'm going to add a join here. Then my join would be a join to my employee record. Again, you can get this information from your SuiteScript records browser, so make sure you use the records browser as a reference.

Now, for the operator, this is something that can change a little bit. You have two options regarding the operator. The operator that we use in our search is Contains, but you can use either Contains, or you can use Has Keywords.

Now, what is the difference between Contains and Has Keywords? The difference is the behavior of the search. If you use Contains, what would happen is that it would be a substring search, meaning it'll try to look for the particular substring that you search for in your main query string. Now, if you use Has Keywords, what's going to happen is that it's going to look for a specific keyword that matches the string that you're looking for.

Now, a keyword would be a separate word, so you can know that something is a separate word if it's in between a space, so for example, you have a space, your keyword, and then another space. Or if it's in between two special characters or a combination of the two. So for example, if you have a period, your keyword, then an AT sign, or maybe you have a space, your word, and then a period. Or maybe if your string is at the beginning of your main string, or is at the end of the main string, so for example, your keywords, so nothing before that, and then a space. Or you have a period, and then your keyword, and then that's the end of your main string, so that will be considered as a word.

So you can just think of the Has Keywords as a separate word. If it's a separate word, then that would be included in your search. Now we would recommend that, if you want to optimize your search, use Has Keywords if you can use keywords instead of Contains, because if you are going to use Contains, that can potentially be slower because it's going to go through each individual check to make sure that the substring matches.

So for example, if you have a 10,000 character string, what's going to happen is that it's going to search for that string one at a time so that until you get a result or complete the search. So that means that you can potentially search for 10,000 times or 9,999 times because you have to move one character at a time. But if you're dealing with Has Keywords, if you have a 10,000 character string, same set of strings, but you only have thousand words in that 10,000 character string, so instead of doing that 10,000 times search, then you only have like 1,000 search because you have to just check each individual word.

Check this word. There's a match. Move to the next word. Move to the next word, until you are done. So Has Keywords is a lot faster-- or can potentially be a lot faster than Contains.

If you can, again, but sometimes, you really can't use Has Keywords. If you're really looking for a substring, it doesn't really matter if it's part of a word or the whole word, then that's fine. I mean, just use Contains because you really can't use Has Keywords in that case. But in our case, we are looking for the word support, so it's a separate word, so that's why Has Keywords would be better.

That's it for the values. Very simple here because this is just a string. We're just looking for status. There you go.

And that's it. We've already created our search. The next thing that we need to do is we need to specify our column.

Here, let me build our columns, and similar to our search filter, this would also be an array. And if you want to build your search column, you would be building that using the Create Columns method. So here, search and then Create Column, and then similar to what we did in our saved search-- we didn't really specify operators and values. All we did was just selected the field that we want to return on our saved search.

Now, in this case, I'll just say I want to return, say, name, and I want to return the subject, which is title. And just copy all of those. And then I have my name. I also have my incident date and the ID of my incident date is start date. So if you're not sure of that, you can always check the records browser.

Then after that start date, I also have-- what else do we have here? Assigned, which is the assigned employee. We also have our status. Have to remember that. Let's add one more here.

And then for my status-- let me see what it looks like. I have not a-- let me just change that. Not going to use a semicolon, but I'm going to use a comma because this is an array definition.

So anyway, these are the four fields that we have on our case record. Now, we also want to get the fields on my employee record because we are doing a join. So what we're going to do here is that we're going to get the department. Aside from our department, we need to make sure that we add in a join property, and that join property would be joined to my employee record. Then I'll just copy this since the joins already there, and I'll change this one to title because I want to get the job title.

Now, notice my format here. You don't need to break it down into several lines, what I did here, where I am splitting all of the different lines. Since I only have one parameter-- or in this case, two parameters, then I'll just put that in one line. I think it's still easy to read.

But if you think it's just going to be really long, it's going to be harder to read, then you can break that down to multiple lines. The formatting is up to you. It doesn't really affect the execution of your script.

That's pretty much it, so we've already created our search. We use the search.create. We created our support case, search, and then we added two filters-- one for my escalated case, and another one where the status of my assigned employee-- or no, not the status. That's support. Good thing I checked.

So we have the support, the job title of our assigned employee is support, and then we are getting some information here, like the title, start date, assigned to, status, and then the department and the job title of our assigned employee. And then for getting the results, we're just using pretty much the same thing here. So the same Get Range, we're going to get 10 results because we only have four results. We don't really need to get everything. If I want to get the whole 1,000, then I can just put in 999 here.

Anyway, let me just copy that. Copy everything and go back to my debugger. Then on my debugger, we space that right over here and then perform my search. I'll click on Debug Script, and I'll add in a breakpoint here at our stopper line because we don't really need to check each individual line.

Then click on Continue, and then here, if I go to my search results, if I expand it, as you can see, same four results because the criteria that we used on our saved search is pretty much the same. So if I expand that, I have same results here as the shipping cost issue. We have the enhancement issue, as you can see here, so that's correct. And we should have two installation issues if I that, so one installation issue here, and then another installation issue.

So that's how you would be building a saved search. It looks like everything is working fine. Let me stop the execution of that. Going back to our script, the system for creating a search on the user interface or on a saved search is pretty much the same as creating a search on your script. It's just that you have to type in all of the commands you need.

So that's about it for this walkthrough. So far, we've created a search using a standard record, and if you want to get the IDs of a standard record, all you need to do is just go to the SuiteScript records browser. If you are dealing with a custom record, then the ID that you will be using would be based on the definition of that custom record.

Now, unlike a sublist, you would actually be using the record ID if you are going to search for a custom record, so for this one, you don't really need to add anything, add any prefix to your IDs. Just use the regular ID. So for example, if I'm going to create a search for my performance review, that's going to be search.create, and then ID would be customrecord underscore sdr underscore perf underscore review.

Now, if I'm going to get the values off my fields, for example, if I want to get my subordinate field, then I'm going to use this as my field ID. So that's custrecord underscore sdr underscore perf underscore subordinate. Also, if I'm going to use a join, then for my join ID, I'm going to use my records or the field that I'm going to use for the join. For example, if I'm joining to an employee record through my subordinate field, then that's going to be my ID.

Now, if I'm going to join to an employee record using another field, then I can use that other field because it is possible to have multiple fields of the same type. So maybe you have a subordinate field, and you also have a supervisor field. Now, instead of joining to the subordinate field, you want to join to a supervisor field, then of course, you need to make sure that your join ID is based on that supervisor ID.

Now, if you are going to perform a join, make sure that you are only going to use the join ID and not the field ID for your custom record. So this can be a little bit confusing, again, for people who are starting out with searches, but just remember, for your type, once you create your search, that will be your record ID. For your fields, you're going to use whatever field is available, similar to your sublist.

And then if you want to perform a join, just use the field that you've used-- or that you're planning to use for the join. And then for the ID, just use the ID of your target record or your parent record. So if you are linked to an employee record, just use the field ID of that employee record, or if you're linked to a customer record, use the ID for that customer records. So that's something that you have to do.

In this walkthrough, what we're going to do is that we're going to log the support cases that we have created. Now, we have created our search, but we haven't really done anything to our search. Now, what we're going to do is just we want to learn how we can parse information on our search. So how can we get data from our search and then do something with that data?

So that's it. So what we want to learn here is we want to know how we can loop through our search results. And also we want to know the different methods that we can use to extract the value from our search.

Going back to our search result here, so we have our search result object. We're still going to use that stopper line-- or actually, we don't need to stopper line anymore because we're just going to execute our search in one go. And then here at the bottom, I'm going to parse my result.

Now, the search results that I'm going to get from my Get Range would be an array of search results. So here, this would be my page range, and then get range would be the actual values. And if I want to get the results off of my page range, the only thing that I need to do is just loop through the results. You can use any looping construct that you want.

Now, in my case, I'm just going to use a FOR loop, so for my FOR loop, I'll just create my counter here. Then I'm just going to say, search results, my search result object here. So I put in searchResults, and then since this is an array, I can use the link property-- if I can spell that right. So I can use the link.

So var i equals to 0, and then I can just say counter i. I have my length here, and then I also need to increment through my counter. That's it.

So again, this is just a simple search-- or a simple looping construct. You can just use whatever looping construct that you want, so you can use a FOR loop or a WHILE loop. But I think, in this case, a FOR loop would be perfect because you need that index number if you want to extract the value off of your array.

So anyway, on our search result, let's get some of the information. We don't really need to extract all of it because we just need to know how that's done. So let's try this subject first. So we have our subject.

So for our subject, we have our search results object, search results. And then to get the information or to extract the element off of our array, you need to put in this identifier. So I have my search results, and then i, so that would be my index number, and then dot, and then the method that you're going to use is Get Value.

And for this one, this is kind of the same as your Get Value and Get Text for a record object. So it's the same function, so if you want to get the value off of your case record-- so here, I have my title. And also, let's say if I want to get the assigned employee-- again, we're not going to get everything. So assigned-- I'll just say Assigned To. I think that's a good enough name.

So I'll just copy this. So for this one, instead of getting the internal ID, because if I just get value, remember Assigned To is an employee record, so you can either get the internal ID or the name of the employee. So instead of using Get Value, I'm going to Get Text. So as you can see, it's pretty much the same.

So the value I have here is assigned. And then what else can we get? Let's say get department, so here var department. So search result-- again, I'll just copy that and paste that right over there.

For the department-- now, for this one, I need to perform a join, so instead of putting in a string as a parameter, I'm going to put in a payload object. So for my payload object, I have my name. Then here, I'll just say department because I also need to perform a join.

So this is where it's a little bit different, but the definition is practically the same as your search column. In fact, what you can do is that you can just copy whatever you have on your search column and then paste that right over here. That would also work.

So we have the Department, then let's also get the status. I think that's important, the var status. Let's change this to status. And then what else do we have here?

The job title-- job title equals to, and then our search. And again, I'm going to use my payload object so that I can define my name, title, and also my join. So this would be joined to my employee record.

And that is pretty much it. So to extract information, just use the search result object. Make sure that you extract the element because your search results would be an array if you're extracting that from a page range.

Then once you have that element, then just use the Get Value or Get Text, similar to what you have been doing with a record object. Now, the thing that would be different would be if you are getting a join. So if you're doing a join, you have to specify the name of the field and where you're joining to.

So that's it, and now, last thing that I need to do here is I'll just put in a debug log, and then on my debug log, I'll just say this would be my case information. Then for my case info, I'll just log in a few things. Maybe subject-- I need my subject there, and then I have my subject already specified in a variable. I'll also add in a line break so that it'll be nicely formatted.

And I'll also add a few more things. Just copy this. Put that right over there, and you just duplicate that. By the way, let's remove this. Fix that later.

So anyway, I have my subject, status maybe. I have status here, and I have-- what else should we get? Let's try to get department. Department-- or job title, I think, because we're using that as a search.

And then here, I'll just say job title, and now we can just use those three. I don't really need to display everything. And close that out, and that's it.

So the only thing different for getting your search result is the FOR loop. That's the thing that we did here, and then we tried to extract that using Get Value and Get Text. And then once you get that value, you can do whatever. Now, in our case, we're just logging the information, so really simple. But if you want to process that, maybe load the additional records, combine certain values, compute for certain values, you can also do that.

So let's try that. Let me copy this and go back to our debugger. Let's go into our debugger here. Let's paste all of the values and then click on Debug Script.

And again, I don't really need to add breakpoints. I will just click on Continue. And if everything is OK, if I go to my execution log, we should have logged four information. It looks like we have logged four information.

So the first information here is the shipping cost issue, so that's correct. We have also another one for the time input form, so that's also correct. Then we have our installation issues, so two installation issues-- so one and then another installation issue. And that's about it, so that's how you would parse the information off of your search.

So that's pretty much it. A few things to remember here-- remember that you can only get up to 1,000 records on a search result, and that is if you're going to use a page range object. If you're going to use a result set object, then the dynamic is a little bit different. Instead of a limitation on the number of records, you have a limitation on the number of units. Now, again, we haven't talked about the number of units or the governance yet, but we are going to talk about that in the future module.

Also, be mindful of the script IDs that you are going to use because if you use the wrong ID, you will get an error message, and your search will not work. If you are going to use a standard record, always go to the SuiteScript records browser and look at the right section. Make sure that you're looking either at the search join, search filter, or the search columns. Do not look at the fields section, or if you're creating a search filter, don't look at the search column section, or if you're creating a search column, don't look at the search filter section. Make sure that you're using the right ID.

Also, it is possible to combine a saved search in a script search if necessary. For example, you can load a search, and once you have a reference to that search, you can add additional criteria, or you can add additional columns if you want to. That's something that you can leverage in case you already have a pre-built save search created by somebody, but there are a few things that you want to modify. Instead of recreating everything on the script, just reuse that and add the things that you want to add on your script.

Now, it's your turn. Please complete the following activity. This concludes module 7, Searching in NetSuite.