DATA REPOSITORY

looking at the data repository. This training will consist of an overview of data repository, seeing how we can add new data sources to the system, searching through our data sources, exploring data source details, and exploring the data snapshot. To start, we're going to load the training project. To do this, we navigate to the getting started screen, find our training project, double clicking on this, clicking yes to confirm that we want to load this project, and then confirming the training project has been successfully loaded, looking both that the training is in green and underlined, as well as it says training at the top of the screen to the left of getting started. Next, we're going to navigate to the data repository. We're going to hover over the left panel until it expands, expand the section for data repository, and then click where it says manage data sources. On our manage data sources screen, you'll notice three panels on the screen. On the left side, we have the ability to connect data to the system through a variety of data sources. In the center panel, we have the data sources already connected to the system, and if we click on any of these on the right side of the screen, we get sources about details about this data source, as well as the data snapshot at the bottom. The first thing we're going to look at in today's training is uploading our own data sources to the system. To do that, we simply need to select the type of data source that we are adding and click on that source type on the connect data panel. I'm going to be uploading a Microsoft Excel file, so I will click where it says Microsoft Excel, which will open a window to configure details about the file. The first thing I'm going to do from here is to point the system to the file I'm uploading. I'll click on the upload file button on the right side of the screen. which will open a new window where I can browse and select the file from my computer. I'm going to click on the browse button, find the file I want to upload, click OK, click save, and then confirm that the file was successfully uploaded to the system. From here, I'm going to check its box and then click close. And then on the original window that opened, I'm going to confirm that the file name being shown is the file I selected. Above that, I have the ability to change the name of the data source as it will appear in the application if I want it to be shown under a different name, as well as change the visibility of this data source, private when I want it only accessible to me or public when other users from my organization should be able to use it as well. I'm going to set mine to public and then come down to the bottom and click on the save button. The application will give me a message on the right side of the screen, letting me know that my data source is being saved, and then it will refresh the current data sources being shown. And we can see that my sample provider file is now on the list. In the center panel, we have a few options to make it easier to find the files that we're looking for. We have the ability to search for the file by the name. So if I want to find my sample provider file, I could begin typing that in the text bubble at the top and notice how the system will filter to only

show the data sources that match the keywords I'm typing in. I can click the X icon to clear my search and I can again see all the different data sources on the system. I also have the ability to change the sort method for the data sources. I can sort this either by the name of the data sources or by the date they are created. To the right of this, I have an option for creating folders if I wanted to group certain data sources together. as well as a button to actually refresh the data source list to see if any new sources have been created or uploaded to the system. Finally on the right side I have an expand and collapse all button for the cases when I do have folders of these data source grouped together so that I can either expand the folders to see all the contents underneath or collapse them when I want to minimize the view. When I click on any data source you'll notice on the right side of the screen the source details pop up as well as the data snapshot actually showing the data in this data source at the bottom. Focusing on just the data source details on the right we can see information such as all the column names as well as the types of these columns and KPIs such as the number of nulls and missing as well as the minimum and maximum value of these columns in the table. I have the ability to refresh the metadata to make sure that these details at the bottom are as up-to-date as possible as well as the ability to edit my data source if I want to rename it or change the visibility or point it to a different data source on my system. I also have the ability to remove data sources from the system as well as to export them to either a CSV or Excel to use outside of the system. Again clicking on any data source will open the data snapshot at the bottom where I can explore the actual contents of this data source. With my data snapshot I have the ability to click the maximize button so that the data snapshot will take up my entire screen and while I'm in this view I will instead have a minimize button to return it to the previous view where it is only shown at the bottom. I have the ability to manually drag and drop it to actually expand it that way and we have an option to close it or collapse it when we no longer wish to see the data snapshot and additionally we have the ability to export the data from the data snapshot itself. You'll notice on the left side here we have the actual name and row count of the data source that we're looking at. So we're looking at our medical service claims sample file with a little bit over 140,000 rows. Underneath that, we can see the actual columns and values in this data source. So if we want to expand any of these columns, we can simply double click on the bar between two columns and it will automatically expand to match the length of the column name or data. You'll notice we also have drop down carrots on each of our columns where I can do things such as filtering it. So for this, for instance, I could type in values that I want to be less than, equal to or greater than. I also have the ability to sort by the different column values here. I can click it once to sort ascending and then I can click it a second time to instead sort descending. And you'll notice on the right side, the system will show us what column we are sorting on and what method we are sorting. We also have the ability to sort on multiple columns at

once. So if I also wanted to sort on another column, we'll see there are now multiple columns that we are sorting on and the system will let us know about all of these. For filtering, if we were only interested in say a certain line number, we'd be able to type that into the filter here. And you'll notice on the left side of the screen, the system is showing us what columns we are filtering on. And just like with sorting, we can filter on more than one column at a time. And additionally, we can sort and filter at the same time. We can remove filters either one at a time by clicking on the X icon on the right side. of the window or by clicking the clear all button to remove all filters. This training includes a few practice questions for understanding, which we will go through now. The first question asks us what is the highest number of lines on a single claim? To do that, I want to get the highest claim line number in the system. So we can simply sort descending by our claim line number. Again, I can click this one time disorder ascending. I can click it a second time disorder descending, and we can see the highest claim line number in the system is 95. Our second question asks us how old is the oldest member? To begin to answer this question, I'm going to clear my previous filter, and then going to scroll to the right until I find a column of showing me member age. Here we are member age, and to find the oldest similarly, I'm just going to click it once to sort ascending, click it a second time to sort descending, and we can see here the oldest member in our data is 104 years old. The third question asks us how many claims are on members over the age of 90? This time, instead of sorting, I'm going to use a filter. So I'm going to clear my sort, I'm going to come back to member age, and I'm going to go to my filters. For this, I'm going to filter to only show claims where the member age is greater than 90. You'll notice when we enter this filter, the data snapshot will update to show us the live number of rows that we're looking at, which is 351, the answer to our question. The fourth question asks us how many claims were paid for exactly $100. I'm going to start off by clearing my current filter. I will then find the column for amount paid, which should be to the left. Here we are amount paid, and we're looking for claims that were paid for exactly $100. So again, I'm going to use a filter, but this time I'm going to filter for equaling $100. Again, the data snapshot will update the live row count, so we can see that there are 87 claims that were paid for exactly $100. And finally, it asks us what is the highest amount that one of these claims was charged for. So in this question, we want to maintain our current filter of amount paid, and simply now sort descending for amount charged to see what the highest amount charged is. I'm going to expand my amount charged column, click it once to sort ascending, and then click it twice to sort descending. And we can see here the highest amount charged for a claim that was paid $100 is $400.

QUERY BUILDER

In today's training, we will be exploring edit development, looking

at an overview of edits, as well as rule creation and scheduling. Typically, sequences of queries or models which address larger business questions, such as which providers are billing for healthcare acquired conditions, which providers are up coding their lab tests with the highest frequency, and are there any providers unbundling outpatient services can be done using edits. In today's training, we will be creating an edit which is based on an OIG audit topic, which looks for radiation therapy, 3D CRT, planning services which were unbundled. The business rule indicates that if the bundled radiation therapy service was billed within 14 days following the planning service, the planning service should have been included as part of that bundle. We will be building an edit that requires a series of queries to be planned out. First, we will identify all radiation therapy planning services. Next, we will cross reference planning services with bundled radiation therapy to identify at-risk planning services. And finally, we will use at-risk services to export on at-risk claims table. In today's training, we will be using the training project. So let's load up the training project on the getting started screen. And then let's navigate to the query builder. First, we're going to identify all radiation therapy planning services. Radiation therapy planning services are defined as being outpatient claims only, meaning that their build type starts with a 0. with one three, have an amount paid greater than $0, and is in a specific list of procedure codes. Let's begin building this query. First, let's select our medical service claims sample table. And let's add in our criteria. First, for outpatient only claims, we wanna specify that the bill type starts with 13. So let's find our claim bill type. Set this like, and then one three with a wild card, specifying that we want it to start with one three. Next, let's add our criteria for amount paid. Switch the criteria to and. We want our amount paid to be greater than $0. And finally, we want our procedure code to be out of the following list. So let's add and. Procedure code is in our list. And these procedure codes are 77014, 77280, 285, 290, 77305, 306, 307, 310, 315, 316, 317, 318, 326, 328, and 337. Now, let's add the following columns to our query. We'll be adding six of them, so I'm just going to go ahead and make six columns here. We'll start with Claim Seek. Then add Claim ICN, Claim Line Number, Billing Provider Analytics ID, Member Analytics ID, and finally the Service From date. Next, we're going to add one more column which uses a more advanced query builder feature using an SQL Server function called Date Add. This column will add 14 days to the service date so we can cross reference the state range in our next query to identify any radiation therapy bundles which occurred within 14 days of the planning service for the same member and provider. So let's go ahead and add another column here. For this one, we will type in Date Add, the name of the function. The first parameter here is the unit of time we'll be adding. In this case, we'll be doing day, comma, the number of days we'll be adding, in this case 14. And finally, the column we will be adding this to, which is Service From Date. Let's go ahead and change the aliases of our two dates to Start Date Range and End Date Range. Let's go ahead

and run our query. This is our first edit component. Let's go ahead and save this edit component as a rule. To do so, I'm first going to close my data snapshot. And then I'll come up to the top for the save icon, but click the drop down carrot next to it and then choose the second option save as a new rule. This will prompt a new window to pop up. So the first thing we're going to want to do is to name our component. In this case, I will do find 3D CRT planning services. After we do that, we'll have to create a new rule group using the create new rule group button. When we click this, a new window will pop up to actually create this rule group. Let's name this rule group our 3D CRT unbundling. And then we can give it any description that we would like. Once we're done with that, we can go ahead and click save. And then from our rule group drop down, we can now choose this new rule group that we created. After we've done this, we can go ahead and click save. And the application will give us a success message letting us know that it was saved successfully. Next, we're going to be doing our second step, which is cross referencing the planning services with bundled radiation therapy to identify. at-risk planning services. Let's start by clearing our canvas using the icon in the top right, and then let's reselect our data source for the medical service claims sample. Let's build a query which filters for the following criteria. Again, we want outpatient claims only with an amount paid greater than zero dollars and only where the procedure code is 77295. Let's go ahead and add our three criteria. So again, our bill type is like 1-3 with our wildcard and our amount paid is greater than zero dollars and finally and our procedure code is equal to 77295. Now we need to cross reference these bundled payments with the previous planning services. For this we can use rule chaining which will enable us to use a previous rules results as a subsequent input. To enable rule chaining we simply check the box up here at the top that says enable rule chaining. After we do that we need to select our rule group. Let's select the group we just created our 3D CRT on bundling. Once we've done that we can actually select our previous rule as a data source for this query. Usually drop down here I can select my find 3d CRT planning services as the second data source for my query. Just like with data sources, adding a rule will require us to configure a join. Let's add an inner join with the following criteria. The planning service member is the same as the bundled service member. The planning service billing provider is the same as the bundled service billing provider. And finally the bundled service occurred within 14 days following the planning service. Let's go ahead and just change our aliases before we're doing these joins. For a medical service claims table I'm going to do the alias B for our bundled service. For a rule that we're chaining with I will do P for our planning services. Now let's go ahead and put together our join. Remember we want to do an inner join. First inner joining on the billing provider analytics ID from the bundled service is equal to the billing provider analytics ID from the planning services. We also want to be joining on the member being the same on both services using the member analytics ID. And finally we want to join on the service date of the bundling service.

The service from date being between the start and end date of the range of our first rule. At this point we will have identified any planning services which meet the criteria for at-r- risk. But what happens if they match more than one bundled service? If they match more than one bundled services, duplicate planning records will be created. To resolve this, we can group by each at-risk claim to ensure that we only have one row per claim in our output. So let's go ahead and add some columns to our query. Let's first do the planning service claim seek, which will be what we'll be grouping by. But let's also additionally get the claim ICN and claim line number from our planning services. And because we need to do aggregation For both of these, let's just do the minimum. So the first of these services. We can go ahead and run this query and we'll see that two claims have been identified as at-risk based on this criteria. Let's go ahead and save this as another rule. Again, we can go up to the top right for the save icon, click save as new rule. And now let's name this rule, identify at-risk planning services. Notice that the rule group has already been selected as we had enabled rule chaining and selected this rule group. Now let's go ahead and save this rule. Now we're ready to move on to our third and final step. Using the at-risk services to export our at-risk claims table. First, let's clear our canvas, and then reselect the medical service claims table. We're going to, again, enable rule chaining and configure the following. A join between the medical service claim and the previous rule output to filter the medical service claim source for the at-risk planning services. To do this, we'll be using ClaimSeek, which is a unique combination of ClaimICN and LINE. Next, we will select all columns we'd like to see in our claims export. At minimum, we should include the claim information, the amount paid, the procedure information, and billing provider details. And remember that this is what we can use to create charts with later. Let's go ahead and create this query. Let's start by enabling rule chaining, choosing our rule group, again, the 3D CRT unbundling. For our second data source, let's do our second rule, identify at-risk planning services. And let's update our aliases. I'll do CL for my medical service claims table, and I will do AR for my at-risk planning services rule. Let's go ahead and add a few columns to our query. Let's start with information on the billing provider, such as the billing provider analytics ID, as well as the billing provider name. Let's get information on the member, such as the member analytics ID, as well as the member name. Let's also get information on the date, so the year quarter. Let's get the claim seek of our claims. I'm going to choose the claim seek from the medical service claims table. Let's also get the claim ICN and line number. Let's get the amount paid. Let's also add a few more columns for information on our procedure and diagnoses. So let's get our procedure code as well as procedure description. Let's also add information on the primary diagnosis code and the primary diagnosis description. Finally, we need to configure our join. So let's add in a join and let's enter join where the two claim seeks match. We can go ahead and run this query. And once again, we're still getting two

claims, but now we have all this additional information on these two claims that were flagged as at risk. Let's go ahead and save this query as another rule. You can save icon. We can select save as new rule. Let's name this rule export at risk claims. This time, let's explore the advanced options. Let's check the box for force rerun chain rule to produce fresh results as well as the box to maintain. all execution results the box for saving results into a data source which we can then select a data source name for let's just do example edit 3d CRT unbundling let's go ahead and click save we've now successfully saved three steps of a rule group. Expand the left panel we have the ability to go into the rules library and we should see our newly created rule group as well as the three rules we created. Now we're going to explore scheduling. Scheduling rules allows them to be run periodically or on an ad hoc basis. Filters can be applied to limit the data which is used for the execution and rules can be combined with various models and chained together within the scheduler. Let's first navigate to the scheduler by hovering over the left panel and going to the second option from the bottom for the scheduler. Let's begin by creating a new process. To do so we can click the add process button at the top and then we can enter a name for our process. Let's name this example edit 3d CRT unbundling. Next let's switch over to the tasks tab to add the different rules we want to run for our new process. For our task type, we want to select rule from the drop down. And then from our task items, let's select the three new tasks that we created. So we have our find 3D CRT planning services, which we can add to the list. Then we have our identify at rich planning services. Again, let's go ahead and add that to the list. And finally, we have our export at risk claims, which we can add to the list. To indicate to the platform the execution sequence, let's mark the dependencies as follows. For the find 3D CRT planning services, we can leave that dependency blank because we do not need it to wait for any of the other tasks to run. For the identify at risk planning services, let's put a dependency of one, meaning that this task will not run until the first task finishes. For the export at risk claims, let's put a dependency of two, so this will not run until the second task finishes. After we've done that, we can go ahead and click Save to save our process. And we can now see it in our process list on the left side of the screen. We can run our process using the play button on the right side of this panel. Let's start by giving our run an execution alias. Let's call this first run. Also on this page, we have the ability to configure the different filters for our run. We could filter on the dates of the claims we would actually be including, or we could filter on any other column on the table. If we only want to look at certain procedure codes or lines of business, we'd be able to configure that into Scheduler here. For this, let's not use any filters and just run this on all of our claims. We can go ahead and click the Execute Now button at the bottom right to run our process. We can monitor the status of our process on the Scheduler. By clicking on the process, we can see the execution history, though right now we're just doing our first run with the status of running.

When it finishes, this status will update to let us know if the run was successful. We can now see the success message next to our process, letting us know that it ran successfully. Let's navigate back to the Manage Data Sources tab within the data repository to look at our output table. Hovering over the left panel of the screen, we can click on Data Repository and then go to Manage Data Sources. Let's refresh our data source list, and we can now see the output table from our process, the Example Edit 3D CRT Unbundling. If we click on this, we can see the output created by our new process.

In today's training, we will continue to explore the query builder, this time exploring criteria. Criteria help us when only specific claims are wanted. For example, if we only wanted the claims for a specific provider, or specific members, specific service state, specific provider type, we can use criteria to limit our output to only those that match the condition previously mentioned. For today's training, we will continue to use the training project. The first thing that I'm going to do is begin putting a query together to return certain columns I'm interested in. Namely, the billing provider name, the member name, procedure code and description, rendering provider name, and service from date. I'm going to start by selecting the medical service claims table, and then adding in the columns of interest to myself. Again, that was the billing provider name, the member name, the procedure code, procedure description, rendering provider name, and finally the service from date. I'm also going to update the alias of my table to be CL for my claims table, rather than the default. When only specific claims are wanted, we can use the criteria button to add a criteria to our query. I'm simply going to come up here and click on the criteria button. And notice the new format of the row at the bottom. The first bubble says where the second is our criteria column one. So this could be a column value such as service from date. We then have an option for our comparator. So this could be equal to this could be greater than less than in not in between not between like not like the same full list you would get in actual SQL. And then finally our criteria column number two which could be a different column or could be a value or values that we type in. So for example the format of this could be where service from date is equal to January 1st or where amount paid is greater than $100. So let's begin by adding a criteria for our service from date and we want our service from date to equal April 1st 2018. The first thing that I'm going to do is for my criteria column one I'm going to add service from date for my comparator. I will leave this as equal to and then finally I'm going to put in 04-01-2018 for April 1st 2018. I can go ahead and run this query and notice I get 142 rows as a result and if we look at the service from date of all of these claims you'll notice they're all April 1st 2018 exactly what I implemented in my criteria. Going back to our query builder let's try another practice example. This time we're going to add a new column to our query. query, namely member age. So I'm going to add column for member age. I'm just going to put this down by my member name to keep my member information

together. Next, for our criteria, we're going to select all claims where the billing provider taxonomy class is equal to pediatrics. I can either adjust my current criteria line to reflect this, or I can use the trash icon on the right side to delete it, and then add a new criteria column to start fresh. Again, we're going to be looking for our billing provider taxonomy class, taxonomy class, and we want this to be equal to pediatrics. After we do this, we can go ahead and run our query, get all the claims back where the billing provider taxonomy class was pediatrics. Let's sort this table descending by member age to see the oldest members that had claims that matched this criteria. So I'll click member age once to sort ascending, a second time to sort descending, and notice I have a few members over the age of 60 with claims for billing providers whose taxonomy class is pediatrics. So maybe this is something we want to investigate further. Let's close our data snapshot and add member ID to the query we ran before. So again, we can just add column, type in member ID. I'm going to put this with my other member information and rerun it. the same thing sorting by member age. So now I can see the actual IDs of my members who are above the age of 60 who are seeing pediatricians. From here, what I can do is I can actually copy the member ID in the data snapshot here, and then I can create a new query to pull all the claims for this specific member using criteria. So I'm going to select this value here, I can right click on the screen and then select copy to copy this member ID to my clipboard. And then I'm going to close my data snapshot, and I'm going to update my criteria to get me all the claims for this member. To do that, I'm going to change the first criteria column here to member ID. And then I'm going to set this equal to the member ID I just copied from my data snapshot. Now when we run this query, we're going to be getting all the claims for this member. I'm also going to add another column for the rendering provider taxonomy class in my query. Rendering provider taxonomy class. So I can have that information in my output table as well. After I've done this, I'm going to go ahead and run my query. See all the claims for this member. Notice that this patient did see a pediatrician, but the services performed do not seem to be age specific. Let's pause for a moment to do some practice questions for understanding. Our first question is, how many claims are there where the amount charged is equal to zero dollars. To answer this question, I'll come back to my Query Builder page, and for my criteria, I'm going to look for amount charged equal to zero dollars. I can go ahead and run this, and notice at the top our row count is 6,300 such claims where the amount charged was zero dollars. Our second question asks, how many claims are there on babies less than a year old or whose age is equal to zero? To answer this question, I can update my criteria to now look at member age, and I can either do equal to zero or I can do less than one, but I'm going to leave mine as equal to zero. I can then go ahead and run this query, and we can see there are 2,594 claims for members less than a year old. Our third question asks us, how many claims are there on members above the age of 60? So I'm going to close my data snapshot, and rather than using equal to, this time

I'm going to switch to greater than. And for my second criteria column, I'm going to do 60. So now we're looking at all claims for members greater than the age of 60. Go ahead and run this, and we can see that there are 14,786 claims for members above the age of 60. The fourth question asks us, how many claims are there with the rendering provider taxonomy class of nurse practitioner? To answer this, we can update our criteria. to look for the rendering provider taxonomy class. Set this back to equals two. And then for a second value nurse practitioner. We can go ahead and run this and we'll see that we have 13,226 claims where the rendering provider taxonomy class was nurse practitioner. Finally, our last question asks how many claims are there with a service from date of July 4th, 2018? To answer this, we can simply change our criteria to look at the service from date. Set this as equal to and then put in 07-01-2018. Oh, sorry, that is 07-04. July 4th, 2018. We can go ahead and run this and we'll see that there are 131 claims with a service from date of July 4th, 2018. For criteria, our options are not just equal to, less than, or greater than, but there are some different types of comparators that we can use, namely like, in, and between. Let's explore a few examples of these. First, let's look at like. So for this example, let's switch our criteria column to look at procedure code. Next, let's change our comparator to be like. What like enables us to do... is to put in a value that is not a full match, but a simply a partial match. For example, if we wanted all procedure codes that started with 99, whether that be 99201, 202, 203, 213, 214, 215, 99999. As long as starts with 99, we'd want to be able to capture that in our criteria. We can use like for this example. All we need to do is put in the string we're trying to match, in this case 99, and then we can use what's called a wildcard, which essentially tells the query builder that we don't care what happens after our string, as long as our pattern is in the column. For the wildcard, we use the percent sign. So what this criteria is saying is we want all claims where the procedure code starts with 99, but then can be followed with anything. We can run this query. And if we focus on the procedure code column, you'll notice all of the procedure codes returned start with 99. Alternatively, instead of getting procedure codes that start with 99, we could put the wildcard before our pattern to get all procedure codes that end with 99. We don't care what proceeds our pattern, as long as the procedure code ends with our pattern. And again, when we run this and focus on the procedure code column, we'll notice that all of the procedure codes end with 99. And if we just want all procedure codes that have a 99 in it, whether they be at the beginning, middle, or end, we could put wildcards on both sides. the 99. So as long as 99 exists somewhere in the procedure code, it will be returned by our query. So in the top row, we can see this procedure code begins with 99. If we scroll down, we can find an example where the procedure code ends with 99. And if we scroll down a little bit further, we can also find procedure codes that have the 99 in the middle. So the opposite of like is not like. Not like will do the same thing except it will return all the columns that are not like the pattern specified. So for

example, if we wanted all the procedure codes that did not start with 99, we could put in not like 99 with a wild card. And we'll get all procedure codes that do not start with 99. Similarly, we could do examples like we did previously using not like for procedure codes not ending with 99 or not having 99 anywhere in them using not like. The query builder also supports the comparator of in and not in. What in enables us to do is pass a list to the second value column. And if the procedure code were to match any of the values passed in the second column, it would return. So let's say for example, I wanted any established patient E&M visit code. So 99212 sorry 99211212213214215. What I could do is I could do work procedure code in 99211, 99212, 99213. 99214, 99215. And if the procedure code is any of the ones in this list that I've provided, it will be returned by my criteria. Notice that I only put a comma, but no space between them. I can go ahead and click execute, and we can see the only claims returned have a procedure code between 99211 and 215. If I instead wanted any claim with a procedure code besides these five, I could use not in to get the opposite set. All procedure codes except the five I've listed here using not in. Instead of writing out all of these procedure codes independently, what we could do is use it between to specify a range between two numbers where we'd want to get all of the matching values. So for example, when I switch my comparator to between, you'll notice that we now have two options on the right side between the first or lower option and the second or higher option. So again, to get all of my established patient E&M visits, what I could do is get all procedure codes between 99211 and 99215. So this will also capture my 213, 212 and 214s because they are between the lower and upper values here. I can go ahead and run this and again, we'll see we're getting all 99211, 212, 213, 214 and 215s. Between will capture the two values we listed here. So it will capture my 211s and 214s. 215 inclusive. Finally, the opposite of between is not between, where we will capture all the claims except those where the procedure code is between this lower and upper bounds. So this will capture all my claims that are not established patient E&M visits using not between. And just a quick note that between works with both numbers as well as dates too. So it can be a very helpful way of filtering for service date ranges. If we wanted all claims between January 1st and December 31st, 2021, we would be able to do that with a between.

In today's training, we will continue exploring the Query Builder, this time looking at aggregations. Aggregations enable us to ask bigger questions that perhaps cannot be answered with just a list of claims. If instead we wanted to know metrics at the provider or member level, we could use aggregations to answer these questions. For example, we could use group by the billing provider NPI to see a list of providers with some information about each. To answer the question for each billing provider NPI, what was their total amount paid, we could group by all our claims by the billing provider NPI and then calculate the sum of the amount paid. And our result will be a list of billing provider NPI's with the total paid for each one. To see this

in action, let's first load our training project and then navigate to the Query Builder. In today's training, we're going to be focused on the medical service claims table. And for example, of getting the total amount paid by billing provider, let's go ahead and add two columns. First, let's get our billing provider NPI. And then let's get our amount paid. When doing aggregations, what we need to do is go to the left bubble where it says select, open the drop down menu, and then choose one of the other options. For our example, we're interested in information at the billing provider NPI level. So let's go ahead and group by our billing provider NPI. For amount paid, as we're interested in total amount paid, let's select sum so we can see the sum across all. of claims for these provider MPIs. When running aggregations, the aliases for columns can be very useful as sometimes the meaning of the output is different than the column name we start with. Let's go ahead and change amount paid to now say total amount paid in our output. We can go ahead and run our query and we'll see that our resulting output table contains the billing provider MPIs as well as the total amount paid for each of these MPIs. If instead we're interested in rendering provider MPIs, we simply go back to our query and instead of grouping by the billing provider MPI, group by the rendering provider MPI. We run this, we can see our resulting output. This time showing the total amount paid by our rendering provider MPI. To look at a practice question, let's think about how we would find the total amount paid for each member ID as well as figuring out what member has the highest amount paid. To do this, rather than grouping on the rendering provider MPI, we can instead group by our member ID and then get the sum of the amount paid to see how much was paid for each member. Here instead of seeing the provider MPI, we see group by member ID with the corresponding total amount paid for each member. If you want to know what the highest total amount paid was, we can simply sort descending by total amount paid to see that this member at the top here. had $72,693.22 paid for them. Next up, we have some questions for understanding. Our first question asks us which non-null place of service description had the highest total paid? To answer this, we can group by our place of service description. And then sum the amount paid. We can run this, sort descending, and we can see that home is the non-null place of service description with the highest total amount paid. Our next question asks us which billing provider ID had the highest amount charged for January 2018? To do this, we can group by our billing provider ID. We can now sum over our amount charged. I'm going to change my alias to total amount charged. And then we can add a criteria to limit the scope of our query to only consider January 2018. So I'm going to select my service from date. And I'm going to set this to be between January 1, 2018 and January 31, 2018. And then go ahead and run this, sort it descending by total amount charged. And we can see that billing provider ID 10021176 had the highest total amount charged. Question three asks us which primary diagnosis had the highest total paid? To do this, we can group by our primary diagnosis code and then sum over the amount paid. And again, I'm going to change my alias to total amount paid. We can run this

sorter table and see the primary diagnosis code Z7689 had the highest total amount paid. Our follow-up question asks, what does this diagnosis code mean? To do this, we can add another column to our query for primary diagnosis description. Notice at the bottom, the query builder is giving us a warning message that we cannot have group by and select together. When we're running a group by, the other columns we're selecting also need some aggregate function performed on them. As we would expect every primary diagnosis description to match exactly to a single primary diagnosis code, we could simply select this to be the min or max, as we expect all the values to be the same. We can go ahead and run this and again, sorter snapshot and see that the primary diagnosis description for this diagnosis code is persons encountering health services in other specified circumstances. In addition to doing sums, we can also use aggregations to look at counts, distinct counts, averages, minimums and maximums. For instance, if we were interested in how many claims were billed on a given date of service, we could create a query that looks like this. Start by grouping by our service from date And then next we count the number of claims on that service state using our count function When we run this we'll see for each service from date we have a corresponding number of claims count From this we could answer questions like what service from date had the most claims Where we can store it by our claim count and see that August 1st 2018 had the most with 1739 claims In our next practice question, let's look at how many rendering provider NPIs have billed each procedure code So for this we can group by our procedure code and then count the distinct rendering provider NPIs To see how many different rendering provider NPIs have billed for this procedure code And we change our alias to number of rendering providers And we can change our alias to number of rendering providers When we go ahead and run this we'll see for each procedure code the number of distinct rendering providers that have billed it And again we can sort this table to see that procedure code 99213 has the most number of distinct rendering providers with 2078 Up next in our training are questions for understanding The first question asks us which procedure code had the highest average amount charged So again, let's group by our procedure code. And this time, get our average amount charged. So we'll get amount charged, and then change my aggregate to average. I can go ahead and run this. Sort my data snapshot. And we see that procedure code J 9042 had the highest average amount charged. Our follow up question asks, what was the total amount paid for this code? To answer this, we can simply add another column to our query, where this time, we get the sum of the amount paid. And I can put in total amount paid. And let's change our alias for amount charged to average amount charged. Again, run this sort by our average amount charged. And we see that procedure code J 9042 had a total amount paid of $18,048.30. Our second question asks, which billing provider ID was associated with the most rendering providers? To answer this question, we can group by our billing provider IDs. And then we can do a count distinct on our rendering provider IDs. Let's change our alias to number of rendering providers.

Run this query. And again, sort our output to see that billing provider ID 10033413 was associated with the most rendering providers at 492. Our third question asks, who was the top paid rendering provider ID? for procedure code 99215. To answer this question, we can group by our rendering provider ID. We can do a sum over the amount paid. Let's change our alias to total amount paid, and then we can add a criteria on our procedure code to only get procedure code equal to 99215. We can go ahead and run this sorter output. And we can see that rendering provider ID 10017623 had the highest total amount paid at $5,350.59 for procedure code 99215. Our follow up question is who would be the top provider if only pediatricians were considered? To answer this, we can go back to our query and add in a second criteria. This time looking at our rendering provider taxonomy class and setting this equal to pediatrics. Note the error message at the bottom where this says only one where per query is allowed. So we can simply set this to an end to require that both the procedure code is equal to 99215 and the rendering provider taxonomy class is equal to pediatrics. We can go ahead and run this again sort by the total amount paid and see that rendering provider ID 10016820 with a total amount paid of $713.44 was the highest paid when we only consider pediatricians. Our next question asks what if all established patient office visits, so 99211 through 215, were considered, again only looking at pediatricians? To answer this we can go back to our query and we can modify our first criteria. Instead of equal to 99215, set this to between 99211 and 99215. We can go ahead and run this and again sort by the total amount paid and see that rendering provider ID 10052033 had the highest total amount paid for all established patient office visits. Our fourth question asks what was the top paid rendering provider referring provider relationship? To answer this let's go ahead and let's group by our rendering provider ID but let's also group by our referring provider ID. So now we're grouping by the pairs of rendering and referring providers. We're interested in which pair is our highest paid so let's do the sum of amount paid and again we'll alias this to be total amount paid. Finally let's add a criteria where the referring provider is not null just to make sure that we're only getting pairs where both providers exist. We do this by setting our criteria to where rendering provider ID is not null. We can go ahead run this query, we can sort by the total amount paid. And see this pair of rendering provider 10090938 and referring provider 10016116 had the highest total amount paid. And we can answer a follow-up question about how much was paid for this combination as $2,868.05.

 returning, we will continue exploring the query builder. Now looking at joins. Joins allow combining information from two different tables. For example, if we wanted to find the authorized official for the billing provider on each claim, we could use joins. Imagine we have a claims table with information such as the claim ID, the member ID, the billing provider NPI, and the amount paid. Well we have a separate table such as a provider file that has information like the primary

NPI, the provider entity type, individual organization, the authorized official name, and the effective date. Today we'll be seeing how we can join these two tables to get information from both returned in a single query builder query. To do so, we're first going to choose a column that they both have that is also the focus of this question. In this example, we'll be using the billing provider NPI from the claims table and the primary NPI from the provider file. We will then join on that column. And finally, we will choose other information to see from either of these two tables. Such as the claim ICN, the member ID, the amount paid, the provider entity type, and the authorized official name. Before doing this in the query builder, let's explore a few samples here. Let's imagine our claims table has the information such as that shown here on the left side of the screen, and our provider file has information such as that shown on the right side of the screen. You can see here our billing provider NPI 1090109 exists three times in the claims table, and we have one associated row in the provider file. Well our billing provider file is 1090109. The provider NPI 1441441 has one row in the claims table and one row in the provider file. When we run a join on these two columns, we will match the three columns from the claim table to the one column from the provider file with the matching NPI, as well as the one row from the claims table and the other row from the provider file with our other NPI. When we run this query, we'll get a result like that shown here at the bottom where we have information from the claims table, such as claim ICN, member ID, billing provider NPI, and amount paid, while also getting information from the provider file, such as the entity type and the authorized official name. There are different types of joins that return different results depending on if the row has a match or not in the other table. For most cases, we will be using a left or an inner join. Left joins return all rows from the left table regardless of if they have a match in the right table or not, whereas inner joins will only return rows if there is a match in both tables. Let's take a look at an example of this. Let's imagine we have an additional row in our claims table with billing provider NPI 1551555, but there is no match in the provider file for that NPI. In the case that we do a left join, it does not matter that there is no match in the right table. We will still carry over that row from the left table, so we will still have a row associated with this billing provider NPI, though the information pulled from the right table will be blank. If, however, instead we did an inner join on the same tables, we will not get a row return for billing provider MPI 1551555 as we will only return matches that exist in both tables. Joins together. Let's first start by adding our medical service claims table to our query. First, let's start by changing the ALA's for medical service claims to CL. And then let's add the columns for the claim ICN, the member ID, the billing provider MPI, and the amount paid. Claim ICN, member ID, billing provider MPI, and finally amount paid. Then we're going to need to add a second table to our query that we will join on. To add an additional table to our query, we come to this plus icon to the right of our current table. And from here, we'll have

the ability to either add a rule or add a data source. Let's go ahead and add a data source, which will enable us to join on another table in the application. Let's join on our provider file. Let's go ahead and start typing in provider file and then select our sample provider file. Let's change the ALA's in this table to P. From this table, let's add in the provider entity type, authorized official first name, and authorized official last name. Again, that is the provider. entity type, the authorized official first name, and the authorized official last name. Note the error message at the bottom saying that the joins are not configured. When we select more than one data source, we also need to tell the system what column or columns to join on. To do that, we push the add join button here in between add column and add criteria. Notice the format of the new row at the bottom. The structure of the join row is the left inner right joining on the column from the first table, the comparator between that generally equal to and the column from our second table. In this case, we want to join on the billing provider NPI from our claims table to the NPI from the provider file. So let's go ahead and fill that in. Billing provider NPI from our claims table is equal to the primary NPI from our provider table. After we do this, we can go ahead and press play to run our query. Note that the claims with an organization as the billing provider NPI have an authorized official and those with an individual as the billing provider NPI do not have an authorized official. Next up, we have some practice questions for understanding. First, how would you structure join between the claims table and the provider table to find information? about the rendering provider for each claim. To do so, we could simply change our join from being on our billing provider MPI to being on our rendering provider MPI from the claims table. This will now be matching information for a provider file on the rendering provider. We can go ahead and run this and see information on the authorized official for our rendering providers. Our next question asks us to write this join to find the last update date for the rendering provider MPI for each claim. To do this, we can go ahead and add another column from our provider file for the last update date. We can go ahead and run this query. And you can see we now have the last update date in our output. Our final question asks us what was the amount paid on the claim with the most recently updated rendering provider MPI. To do this, we can simply sort on our last update date. We can see here the most recently updated date was August 1, 2019. And this claim was paid $62.65.

CHARTS

In today's training, we will be looking at the charting module, exploring the different types of charts that you can create in the charting module. Let's start by loading the training project and then navigate into the charting module by hovering over the left panel of the screen, expanding the analysis dropdown, and then going to charting. In FTVB Finder, we can create a variety of different chart types to help us visualize information and recognize patterns.

Creating a chart is as simple as selecting a data source from the left panel, then selecting the rows and columns that we'd like to include in our chart by dragging and dropping them into the right window. Let's go ahead and select our medical service claims data set. Notice when we do that, that the descriptors and values section will now show the different columns in our data set. Let's search for our billing provider taxonomy group from the list. We can do that manually or by using our search bubble here. Let's drag and drop our billing provider taxonomy group into the rows section in the right panel. Next, let's search for amount paid in the descriptors and values box. We can also search for the amount paid in the search box. Let's go ahead and select the amount paid in the search box. We can also select the amount paid in the search box. Let's go ahead and select the amount paid in the search box. We can also select the amount paid in the descriptors and values box. Let's first clear our search and then search for amount paid. Let's drag this into our column section on the right panel. Let's go ahead and click the arrow to the right of our descriptor to see some of the charting options. We can do different things such as change the alias of this descriptor on our chart, as well as the color, the aggregation, the sorting, or we could remove this descriptor from our chart. Let's go ahead and change the alias from amount paid with no spaces, to now include a space to make it more visually clean on our chart. Let's go ahead and change our color to a pink color as well using the color option. With the color here, we can either manually select it from the grid, type in the hex decimal representation, or the RGB or the HSVA representation of our color. When we're happy with our color choice, we can go ahead and click OK, and notice how the bars in the chart update to reflect the color selected. Next, let's hover over any of our bars to see some of the tooltip information. If instead of using a vertical bar chart, we want to use a horizontal bar chart, we go up to our chart palette in the top left and see the different types of charts and representations we could visualize our data with. Notice some of the options are grayed out and are not selectable. This is because the application is smart enough to know which visualizations work with the current descriptors selected and which do not. Let's go ahead and select our horizontal bar chart. Now, let's go ahead and sort our data. Let's go to Amount Paid and click Sort Descending so that we can see the highest amount paid taxonomy groups at the top and the lowest paid taxonomy groups at the bottom. Finally, let's update the ALS for Billing Provider Taxonomy Group to again put spaces in between the words to make it a little bit visually cleaner. We can hover over any of the bars and see the inner tooltip Billing Provider Taxonomy Group now has spaces between the words. When we're happy with our chart settings, we can save our chart by going up into the top right, clicking the Save icon and saving the chart as. Let's go ahead and save this as Amount Paid by Billing Taxonomy Group. We're happy with the name we entered. We can go ahead and click OK. And in just a second, the application will give us a success message that our chart was saved successfully. Let's practice charting by doing an example

together. Let's make a bar chart showing the average amount paid per place of service description, sorting this chart in descending order. And then let's use this chart to answer the question, which place of service has the highest average amount paid? First let's use the button to clear the canvas. Then we can select our medical service claim sample. First let's get our amount paid. Let's drag this into our row section. Next let's get our place of service description. Let's drag this into our column section. Let's change the aggregation on amount paid to average to get the average amount paid. And let's sort this descending by the average amount paid. Let's go ahead and change our aliases as well. Say average amount paid. And then let's put spaces in our place of service description. Click OK. Going over the tallest bar here on the left, we can see that the highest average amount paid is $2024.24. And the place of service description corresponds with is ambulance, air or water. Let's try changing our chart palette to a pie chart. Let's click on the chart palette button in the top left and find the pie chart option here on the right. Let's go ahead and save our chart. Let's call this one average amount paid by place of service. And then click OK. Let's create another chart. Let's start by clearing our canvas. And then let's find the descriptor year quarter. I'll begin typing in year quarter. When the object pops up, I will drag and drop it into my rows section. Let's go ahead and select the line chart option. So we go to our chart palette and then on the left side of the screen here, we have our line chart option. And then for other descriptor, let's find our amount paid. Let's drag and drop this into the column section of our charting palette. Let's go ahead and change this chart to a blue color. We can go to the bar to the right of amount paid, expand the drop down menu and choose color. And then let's pick a blue color for this chart and click OK. Let's also go ahead and update our aliases again, putting a space between the words year and quarter as well as a space between the words amount and paid. And then let's go ahead and save our chart and name it amount paid by year quarter. Let's do another practice example together. This time, let's make a line chart showing the amount paid per service from date, sorting by the service from date in ascending order. And let's use this chart to answer the question, which service from date shows the highest spike in amount paid. Let's go ahead and clear our canvas and let's drag and drop amount paid into the column section. And then let's also get our service from date. Let's do another practice example. Let's go ahead and change the chart to a green color. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the color to green. Let's go ahead and change the

color to green. into the row section and then let's sort ascending by our service front date. We go ahead to our chart palette and change this to a line graph. And from this chart we can see that this day here, June 1st, 2018 has the largest spike with a total amount paid of $175,784.89. Let's again update our aliases, putting spaces between the words. And after we do that, let's save our chart. We're calling this amount paid by service front date. Next, let's look at creating a drill down chart. Let's start by clearing our canvas and then selecting our medical service claims data source. Let's first add the service category to our chart. We're going to drag and drop this into the rows section. Next, let's add service type. With service type, let's also add this to the rows section of our chart configuration. And then finally, let's add amount paid. And let's add amount paid to the columns section. From our chart palette, notice some charts at the bottom that have green arrows going across them. This signifies that these charts are drill down charts, meaning that we can start at one category and drill down to a lower category, though filtering for the value we selected in the first category. In this example, let's select the drill down bar chart. After we do this, let's go ahead and sort by our amount paid descending. And let's update our aliases. For service category, let's put a space between the words. Likewise for service type and likewise for amount paid. Notice on the chart right now that we only see the service category by the amount paid, even though service type is part of our configuration. This is because this is a drill down chart. If we were to click on one of these bars, for instance, our medicine services and procedure, we would then see the amount paid by the service type, but only the service types with the service category we clicked on. This enables us to look at our data at different levels and drill down to explore how one category behaves at a lower level. We can go ahead and click back to go back to our original view, looking at the service category. Let's go ahead and save our chart as amount paid by service type. Let's look at another example of creating a drill down chart. This time using service category and billing provider. taxonomy group and class. Let's first start by selecting service category, drag and dropping that into our rows section. Next, let's get our billing provider taxonomy group and drag and drop that also in the rows section. Finally, let's also drag and drop our billing provider taxonomy class into again the rows section. After this, let's go ahead and add amount paid to the column section. And then let's sort by amount paid descending. And in our chart palette, let's select the drill down bar chart. And notice we're seeing amount paid by the different service categories. But if we were to drill down by clicking any of the bars, we can then see the amount paid by the billing provider taxonomy group. And then we can drill down again and see the amount paid by the billing provider taxonomy class. Let's go ahead and update our aliases and then save our chart. Again, I'm just going to be putting spaces between all of the words in my column names. And let's also update our color for amount paid to be maybe a purple rather than gray. Let's go ahead and save our chart as amount paid drill down. by billing provider taxonomy group. And then

click okay. For practice, let's do one more drill down chart where we do amount paid by the rendering provider taxonomy group that then drills down to the rendering provider taxonomy class. Let's start off by dragging amount paid to our column section. And then let's find our rendering provider taxonomy group. Drag and drop that into the rows section. And let's have this drill down to our rendering provider taxonomy class. Let's configure our chart pout to be a drill down bar chart. And let's sort by amount paid descending. Let's go ahead and change the aliases for our different descriptors. Again, just putting spaces between all the words. Let's also add a filter to our chart to filter out any claims without a billing provider on them. To do this, we can search for our billing provider ID. Let's drag and drop this into the filters window at the bottom. Notice when we do this, the application will have a window pop-up that will enable us to include or exclude any of the billing provider IDs found in the table. Let's go ahead and check all the boxes, but then use the search window to find boxes that have error as a value. And let's uncheck this box to not include any claims with the value error as the billing provider ID. From here, we can click Apply, and our chart will automatically update to reflect this filter that we've just applied. Let's go ahead and save this chart. It has amount paid drill down by rendering provider taxonomy group. Let's next look at a social network graph. Let's start by clearing our canvas. Let's also use our filters to filter out any claims that don't have a billing provider ID. Again, let's drag and drop our billing provider ID to our filter section. And let's check all the boxes except forever. Click Apply. Notice that the application has a window pop-up that will enable us to include or exclude any of the billing provider IDs found in the table. Notice that the application has a window pop-up that will enable us to include or exclude any of the billing provider IDs found in the table. gives us an error message as we have not correctly configured a chart. It will continue to do so until we have proper chart configurations. However, if we have some steps we want to do, such as setting up filters or tooltips, before actually setting up the chart, we can click the play button up here to actually pause the live generation of the charts to prevent these messages from popping up. Next, let's add a filter for the rendering provider taxonomy group. Let's go ahead and find our rendering provider taxonomy group. Let's drag and drop that into our filters. For this, let's only include the behavioral health and social service providers group by checking the box next to that taxonomy group and then clicking apply. Now let's begin adding rows and columns to our chart. Let's add the rendering provider name. Let's add this to the rows section. And now at this point, our chart will be able to configure and render properly so we can unpause the rendering. Next, let's add the billing provider name to our chart. Let's drag and drop this also in the rows section. Let's go ahead to our chart palette and select the machine that's got our payments. Now let's add the social network map, which is this one here at the bottom. Notice the different network structures that appear in the window. We have the ability to zoom out or zoom in on these structures, as well as

navigate around by dragging and dropping. Let's go ahead and add a mount paid to this chart. Let's drag and drop this into our column section. We can push the stop in the top right here to disable any motion in our graph. For practice, let's edit the rendering provider taxonomy group filter to include the rendering provider taxonomy group filter to include only student health care. To do so, we can simply come down to our filter section and click on the open filter button. Then we can uncheck the behavioral health and social service providers and find the new group we're looking for, which is student health care. Check that box and click apply. Let's go ahead and save our chart. Let's save this as rendering and billing provider social network.

DASHBOARDS

In today's training, we're going to be looking at dashboards. Dashboards are a group of charts that can be organized and filtered at different levels. We can drill between dashboards to navigate from a high-level dashboard to a low-level dashboard, seeing different aspects of the same topic. On the left-hand panel, let's navigate to the Dashboard section. We're going to start by creating a dashboard. To add items to the dashboard, we come up to the top and click on the plus icon, and then we'll have the ability to add different types of items, including title and note items, data source items, chart items, and KPI items. Let's start by adding chart items. Let's begin by looking for one of the charts we created during the charting training. Let's look at the service category by amount paid chart. Let's drag and drop this item into the main screen. Next, let's add some of the other items we created during the charting tutorial. Let's get our amount paid by billing provider taxonomy group. Let's also get the amount paid by year quarter. And finally, let's add our average amount paid by place of service. To quickly align these charts and make it more visually appealing, we can use the free-floating button here. By clicking Disable Free Floating, the application will automatically align the charts and size them to create a 2x2 grid. If we want to rearrange the layout of the charts, it's as simple as dragging and dropping any of the charts into the position that we want them in. After building our dashboard, let's go ahead and save it by clicking the button in the top right and then clicking Save Dashboard. Let's name our dashboard Service Category Dashboard. At the top, under Dashboard Options, we have the ability to show or hide the headers of our charts if we want to only look at the charts themselves. We also have the ability to filter some of our different charts. We can use local filters on specific charts by clicking on the icon on the header for that chart and then choosing the column that we'd like to filter on. Let's choose Service Category. Click on our Edit Filter option and then choose the categories we want to include or exclude. Let's exclude the box Metasyn Services and Procedures. And then click Apply. and then click apply again. Notice when we create a local filter it only applies to the specific chart we configured the filter for. If we

want to reset the filter we can simply click the local filter button again and click the reset filters button. We confirm by clicking yes, click apply, and then all of our local filters will be removed. Let's next expand the left panel by clicking the expand panel button here in the top left corner of the screen. And then let's expand the section for global filters. Global filters will apply filters across all of the different items in our dashboard, not just one specific chart or table. Let's create a global filter by clicking on the add global filter button, and then choosing the type of column that we want to filter on. Let's select text and then from here let's select the column we'll be filtering on, the rendering provider taxonomy group. Let's check the box next to this and then let's give our filter a name. Let's name this physicians. And then we can click apply. Next we need to actually set the parameters for our global filter. Let's double click on our filter on the left side of the screen here. And now a window pops up allowing us to select the different values that we'd like to filter on. Let's go ahead and unselect all of the options and then only reselect allopathic and osteopathic physicians and then click apply. Notice how all of the charts on our dashboard update to reflect the global filter. To remove this filter, we can simply click the trash can icon to the right of it on the global filter section on the left side of the screen. The application will ask us to confirm, and after we click yes, the global filter will be removed. Next, let's look at linking dashboards together using drill downs. Let's go to our paid by service category chart and click the gear icon on the header panel. A new panel will appear on the screen on the right side. Click add. We will have the option to link dashboards using the link dashboards button. When we click on this button, we will have the ability to add a new link. And then we can configure this link to look at our desired dashboard to drill down to, our lower level dashboard. After we do this, we can click apply and then save the changes we've made to our dashboard. Let's start by left clicking on the bar for medicine services and procedures. From here, let's hover over the first bar, allopathic and osteopathic physicians, and then right click on this bar to see our drill down options. When we hover over the lower level dashboard tab, you can see we can drill down to this lower level dashboard by our billing provider taxonomy group. When we click on this, notice how we're drilled down to the lower level dashboard. only for the specific taxonomy group that we were selecting in the higher level dashboard.

EDIT DEVELOPMENT

In today's training, we will be exploring edit development, looking at an overview of edits, as well as rule creation and scheduling. Typically, sequences of queries or models which address larger business questions, such as which providers are billing for healthcare acquired conditions, which providers are up coding their lab tests with the highest frequency, and are there any providers unbundling outpatient services can be done using edits. In today's training, we

will be creating an edit which is based on an OIG audit topic, which looks for radiation therapy, 3D CRT, planning services which were unbundled. The business rule indicates that if the bundled radiation therapy service was billed within 14 days following the planning service, the planning service should have been included as part of that bundle. We will be building an edit that requires a series of queries to be planned out. First, we will identify all radiation therapy planning services. Next, we will cross reference planning services with bundled radiation therapy to identify at-risk planning services. And finally, we will use at-risk services to export on at-risk claims table. In today's training, we will be using the training project. So let's load up the training project on the getting started screen. And then let's navigate to the query builder. First, we're going to identify all radiation therapy planning services. Radiation therapy planning services are defined as being outpatient claims only, meaning that their build type starts with a 0. with one three, have an amount paid greater than $0, and is in a specific list of procedure codes. Let's begin building this query. First, let's select our medical service claims sample table. And let's add in our criteria. First, for outpatient only claims, we wanna specify that the bill type starts with 13. So let's find our claim bill type. Set this like, and then one three with a wild card, specifying that we want it to start with one three. Next, let's add our criteria for amount paid. Switch the criteria to and. We want our amount paid to be greater than $0. And finally, we want our procedure code to be out of the following list. So let's add and. Procedure code is in our list. And these procedure codes are 77014, 77280, 285, 290, 77305, 306, 307, 310, 315, 316, 317, 318, 326, 328, and 337. Now, let's add the following columns to our query. We'll be adding six of them, so I'm just going to go ahead and make six columns here. We'll start with Claim Seek. Then add Claim ICN, Claim Line Number, Billing Provider Analytics ID, Member Analytics ID, and finally the Service From date. Next, we're going to add one more column which uses a more advanced query builder feature using an SQL Server function called Date Add. This column will add 14 days to the service date so we can cross reference the state range in our next query to identify any radiation therapy bundles which occurred within 14 days of the planning service for the same member and provider. So let's go ahead and add another column here. For this one, we will type in Date Add, the name of the function. The first parameter here is the unit of time we'll be adding. In this case, we'll be doing day, comma, the number of days we'll be adding, in this case 14. And finally, the column we will be adding this to, which is Service From Date. Let's go ahead and change the aliases of our two dates to Start Date Range and End Date Range. Let's go ahead and run our query. This is our first edit component. Let's go ahead and save this edit component as a rule. To do so, I'm first going to close my data snapshot. And then I'll come up to the top for the save icon, but click the drop down carrot next to it and then choose the second option save as a new rule. This will prompt a new window to pop up. So the first thing we're going to want to do is to name our

component. In this case, I will do find 3D CRT planning services. After we do that, we'll have to create a new rule group using the create new rule group button. When we click this, a new window will pop up to actually create this rule group. Let's name this rule group our 3D CRT unbundling. And then we can give it any description that we would like. Once we're done with that, we can go ahead and click save. And then from our rule group drop down, we can now choose this new rule group that we created. After we've done this, we can go ahead and click save. And the application will give us a success message letting us know that it was saved successfully. Next, we're going to be doing our second step, which is cross referencing the planning services with bundled radiation therapy to identify. at-risk planning services. Let's start by clearing our canvas using the icon in the top right, and then let's reselect our data source for the medical service claims sample. Let's build a query which filters for the following criteria. Again, we want outpatient claims only with an amount paid greater than zero dollars and only where the procedure code is 77295. Let's go ahead and add our three criteria. So again, our bill type is like 1-3 with our wildcard and our amount paid is greater than zero dollars and finally and our procedure code is equal to 77295. Now we need to cross reference these bundled payments with the previous planning services. For this we can use rule chaining which will enable us to use a previous rules results as a subsequent input. To enable rule chaining we simply check the box up here at the top that says enable rule chaining. After we do that we need to select our rule group. Let's select the group we just created our 3D CRT on bundling. Once we've done that we can actually select our previous rule as a data source for this query. Usually drop down here I can select my find 3d CRT planning services as the second data source for my query. Just like with data sources, adding a rule will require us to configure a join. Let's add an inner join with the following criteria. The planning service member is the same as the bundled service member. The planning service billing provider is the same as the bundled service billing provider. And finally the bundled service occurred within 14 days following the planning service. Let's go ahead and just change our aliases before we're doing these joins. For a medical service claims table I'm going to do the alias B for our bundled service. For a rule that we're chaining with I will do P for our planning services. Now let's go ahead and put together our join. Remember we want to do an inner join. First inner joining on the billing provider analytics ID from the bundled service is equal to the billing provider analytics ID from the planning services. We also want to be joining on the member being the same on both services using the member analytics ID. And finally we want to join on the service date of the bundling service. The service from date being between the start and end date of the range of our first rule. At this point we will have identified any planning services which meet the criteria for at-r- risk. But what happens if they match more than one bundled service? If they match more than one bundled services, duplicate planning records will be created. To resolve this, we can group by each at-risk claim to ensure

that we only have one row per claim in our output. So let's go ahead and add some columns to our query. Let's first do the planning service claim seek, which will be what we'll be grouping by. But let's also additionally get the claim ICN and claim line number from our planning services. And because we need to do aggregation For both of these, let's just do the minimum. So the first of these services. We can go ahead and run this query and we'll see that two claims have been identified as at-risk based on this criteria. Let's go ahead and save this as another rule. Again, we can go up to the top right for the save icon, click save as new rule. And now let's name this rule, identify at-risk planning services. Notice that the rule group has already been selected as we had enabled rule chaining and selected this rule group. Now let's go ahead and save this rule. Now we're ready to move on to our third and final step. Using the at-risk services to export our at-risk claims table. First, let's clear our canvas, and then reselect the medical service claims table. We're going to, again, enable rule chaining and configure the following. A join between the medical service claim and the previous rule output to filter the medical service claim source for the at-risk planning services. To do this, we'll be using ClaimSeek, which is a unique combination of ClaimICN and LINE. Next, we will select all columns we'd like to see in our claims export. At minimum, we should include the claim information, the amount paid, the procedure information, and billing provider details. And remember that this is what we can use to create charts with later. Let's go ahead and create this query. Let's start by enabling rule chaining, choosing our rule group, again, the 3D CRT unbundling. For our second data source, let's do our second rule, identify at-risk planning services. And let's update our aliases. I'll do CL for my medical service claims table, and I will do AR for my at-risk planning services rule. Let's go ahead and add a few columns to our query. Let's start with information on the billing provider, such as the billing provider analytics ID, as well as the billing provider name. Let's get information on the member, such as the member analytics ID, as well as the member name. Let's also get information on the date, so the year quarter. Let's get the claim seek of our claims. I'm going to choose the claim seek from the medical service claims table. Let's also get the claim ICN and line number. Let's get the amount paid. Let's also add a few more columns for information on our procedure and diagnoses. So let's get our procedure code as well as procedure description. Let's also add information on the primary diagnosis code and the primary diagnosis description. Finally, we need to configure our join. So let's add in a join and let's enter join where the two claim seeks match. We can go ahead and run this query. And once again, we're still getting two claims, but now we have all this additional information on these two claims that were flagged as at risk. Let's go ahead and save this query as another rule. You can save icon. We can select save as new rule. Let's name this rule export at risk claims. This time, let's explore the advanced options. Let's check the box for force rerun chain rule to produce fresh results as well as the box to maintain.

all execution results the box for saving results into a data source which we can then select a data source name for let's just do example edit 3d CRT unbundling let's go ahead and click save we've now successfully saved three steps of a rule group. Expand the left panel we have the ability to go into the rules library and we should see our newly created rule group as well as the three rules we created. Now we're going to explore scheduling. Scheduling rules allows them to be run periodically or on an ad hoc basis. Filters can be applied to limit the data which is used for the execution and rules can be combined with various models and chained together within the scheduler. Let's first navigate to the scheduler by hovering over the left panel and going to the second option from the bottom for the scheduler. Let's begin by creating a new process. To do so we can click the add process button at the top and then we can enter a name for our process. Let's name this example edit 3d CRT unbundling. Next let's switch over to the tasks tab to add the different rules we want to run for our new process. For our task type, we want to select rule from the drop down. And then from our task items, let's select the three new tasks that we created. So we have our find 3D CRT planning services, which we can add to the list. Then we have our identify at rich planning services. Again, let's go ahead and add that to the list. And finally, we have our export at risk claims, which we can add to the list. To indicate to the platform the execution sequence, let's mark the dependencies as follows. For the find 3D CRT planning services, we can leave that dependency blank because we do not need it to wait for any of the other tasks to run. For the identify at risk planning services, let's put a dependency of one, meaning that this task will not run until the first task finishes. For the export at risk claims, let's put a dependency of two, so this will not run until the second task finishes. After we've done that, we can go ahead and click Save to save our process. And we can now see it in our process list on the left side of the screen. We can run our process using the play button on the right side of this panel. Let's start by giving our run an execution alias. Let's call this first run. Also on this page, we have the ability to configure the different filters for our run. We could filter on the dates of the claims we would actually be including, or we could filter on any other column on the table. If we only want to look at certain procedure codes or lines of business, we'd be able to configure that into Scheduler here. For this, let's not use any filters and just run this on all of our claims. We can go ahead and click the Execute Now button at the bottom right to run our process. We can monitor the status of our process on the Scheduler. By clicking on the process, we can see the execution history, though right now we're just doing our first run with the status of running. When it finishes, this status will update to let us know if the run was successful. We can now see the success message next to our process, letting us know that it ran successfully. Let's navigate back to the Manage Data Sources tab within the data repository to look at our output table. Hovering over the left panel of the screen, we can click on Data Repository and then go to Manage Data Sources. Let's

refresh our data source list, and we can now see the output table from our process, the Example Edit 3D CRT Unbundling. If we click on this, we can see the output created by our new process.