Cyber Seminar Transcript

Date: 10/31/2016

Series: VIREC CDW

Session: Data Management in SQL: Select Intermediate SQL Skills

Presenter: Margaret Gonsoulin

*This is an unedited transcript of this session. As such, it may contain omissions or errors due to sound quality or misinterpretation. For clarification or verification of any points in the transcript, please refer to the audio version posted at* [*www.hsrd.research.va.gov/cyberseminars/catalog-archive.cfm*](http://www.hsrd.research.va.gov/cyberseminars/catalog-archive.cfm)

Moderator: Welcome, everyone, to VIReC Corporate Data Warehouse Cyber Seminar series. Thank you to Sider for providing technical and promotional support. Today’s session is titled Data Management in SQL: Select Intermediate SQL Skills. This talk is aimed at an audience that has some basic familiarity with SQL and the CDW and is interested in learning additional SQL commands for data management.

Today’s speaker is Dr. Margaret Gonsoulin. Dr. Gonsoulin is a sociologist who studied at the University of Virginia and taught for eight and a half years in the California State University system before she joined the VA Information Resource Center in 2014. Her work at the VA has primarily focused on the CDW. If you have any questions for Dr. Gonsoulin during the presentation, please send them in using the chat box and I will present them to her at the end of the session.

Now, I am pleased to welcome today’s speaker, Dr. Margaret Gonsoulin.

Dr. Margaret Gonsoulin: Thanks, Hara. Thanks, Molly. Thanks, everyone, for joining us today. As you know, we’ll be going through a few selective intermediate data management skills using Structured Query Language. Before I begin, I would like to give great thanks and appreciation to Richard Pham, Mark Dean, Andy Kelly and Hans Nielsen for all their help, guidance and mentorship.

I would also ask a question about the audience. If you would be willing to participate, I would like for you to tell us a little bit about your level of experience with CDW and how you would represent it on this scale, from not having worked with it at all to very experienced.

Moderator: Thank you. So, as our attendees can see on your screen, we do have that first poll question open. The answer options are: have not worked with it at all, have minimal experience working with it, have worked closely with it for less than six months, have worked closely with it for six months to two years, or you feel very experienced with CDW. It looks like we’ve got a nice responsive audience. Three-quarters of our attendees have already replied. I see a pretty clear trend, so I’m going to go ahead and close this poll out and share those results. Fourteen percent have not worked with it, about a third of our respondents have minimal experience, twenty percent less than six months, twenty-five percent six months to two years, and seven percent feel very experienced with it. Thank you once again. Margaret, I will give you back the screen share.

Dr. Margaret Gonsoulin: Thank you so much. Thanks, everyone, for participating in the poll. For today’s talk, I hope that you in the audience, especially the people who have been working with CDW for a little while who answered this poll, will be able to walk away with a few new skills in your skillset. These include just feeling better prepared to incorporate some best practices that we’ve been building throughout this cyberseminar series over the last couple of years; also, to feel comfortable using temporary tables as you test out new logic in SQL, so that you don’t end up with a whole bunch of permanent tables out there taking up your workspace. Also, we’re going to learn to recode columns using SQL, so hopefully that will become part of your repertoire if it isn’t already. That you will be able to use what are referred to as “partitions” to select desired records in those cases where multiple records are present in the data.

I would like to begin this cyberseminar by going over six best practices, some of which we’ve covered before and some of which are new. In general, it’s good when you’re just wondering whether your query is doing what you want it to do to begin by working with a small amount of data so that you don’t overtax the system for a simple problem-solving part of your process. So, we’ll review some best practices in how to do that. Also, we’ll take a quick look at using partition dates and index columns where possible to make our queries run better. This is a newly introduced element to the cyberseminar series, but it’s pretty straightforward in how to find these dates and columns, and if you use them in your queries things tend to go better. Also, we’ll talk a bit about the best practice of converting your date/time fields that were introduced in the CDW insight today and can help your queries run more smoothly and efficiently. We’ll talk about the temporary tables quickly in our best practices and more in depth later on in this cyberseminar. Of course, we’ll review one more time as we discussed in our previous cyberseminar last Monday, the use of execution plans to look at the estimated cost of your query and also look for potential signs of trouble with the query that you’ve written so that you can go and optimize your query better before you hit the execute button. Finally, as always, just a reminder that before you write any query, it’s always really helpful to make sure that you are working with the data that you need and that you understand the data that you’re working with by using the available documentation before you get started.

Our first best practice, starting with that small amount of data, can be accomplished in numerous ways. Some of the ways we’ve already talked about such as taking the top 10,000 rows or the top 100 rows or whatever number is relatively small but that would still help you look at your data. Then by working with that small number of rows you can see whether you have any large flaws in your logic or your query and fix them before you try and run it against the table as a whole or a larger cohort. Also, remember that you can reduce the amount of data you’re working with by selecting only specific columns in your SELECT statement rather than catching the whole of the table at one time. So, just get your columns of interest only and you’ll start to save processing time.

Also, as you’ll remember, you’ve seen many WHERE statements along the way in this cyberseminar series because that WHERE statement allows you to reduce the data that you’re requesting from the warehouse in a multitude of ways based on any column of interest. Some of the more common choices are only looking at a specific station or only looking at a specific date range, but it might also be only looking at a specific diagnosis or any number of other conditions.

Let’s move on to the best practice of looking at partition keys or index columns. We’ll work the partition key first. Partition key in a kind of general overall way can be understood as a key that takes a single large table and kind of splits it into smaller subtables with some little markers, if you will, in there. So, it’s like allowing you to look at a chunk of the table rather than going through every single row of the table. How do you figure out what are the partition keys in a CDW? Pardon me. I’m recovering from a cold. There is a very straightforward way to look at these partition keys and identify which of them may be helpful given the data that you’re working with at the moment.

A place to start is the CDW SharePoint sites metadata report. I put the URL here at the bottom of this slide. On the left, you can see that I’ve expanded two domains, the HealthFactors domain and the Immunization domain, so that I could see the list of tables in the metadata report that are associated with these two domains. If I focus my attention to the right-hand side under the column called “Relevant dates,” I can see the information I need about which keys in the tables, if they are present, would serve as partition keys for the query that I might like. So, for the HealthFactors domain, you can see that the fact table HF (HealthFactors) has a partition key “HealthFactorDateTime,” and if I use that key in my query when I’m running against this HealthFactors table, it will process more quickly. I can pick out a certain chunk of time and it wouldn’t have to go through the whole HealthFactors table in order to find that chunk of time that I’m interested in. The same thing for the Immunization fact table. Immunization has the partition key “VisitDateTime.” If I use that key, my query will run more efficiently.

Let’s take a look at the other kind of helpful column called an Indexed Column. An indexed column is a column that acts as a pointer to data in a table, much the same way that an index in the back of the book will point to a particular page. You don’t have to scan the whole book looking for information on your topic of interest. There is that page number there that will point you right to the right page. Then you can only scan that small section of the book. That’s what an indexed column does for you in the Corporate Data Warehouse.

So, how do you find the indexed columns for the tables that you’re interested in using? Well, inside SQL Server Management Studio, inside the CDW Work folders as I’ve mentioned in previous cyber seminars, there are views that begin with the schema META or meta and these are metadata views. So, they are views that contain documentation about the data themselves and it’s stored right inside the database itself, as we’ve looked at several examples of this so far in the series. This is the new one that is called the Meta.DW (That stands for data warehouse) index. It’s going to give us information about which columns are indexed. Here, I just took a shortcut by right mouse clicking on meta.DWViewIndex and I have a little automatic query popup in the query window that you can see here. Select top 1000. That’s an automatic query. Then, at the bottom of my automatic query, I added a WHERE statement so that I would be able to focus in on particular schema, particular domain or particular table. You could choose whatever you want to focus on and say “only give me output related to this area of the warehouse.” Then, on the bottom right, you can see the automatic output and a column that tells you which of the columns in each of these tables will serve as an index. If you use these, it will point you to specific points of the table and make your query run more efficiently.

Our third best practice comes to us from Andy Kelly’s talk on the CDW Insights which was given on July 27th, 2016. The URL for all the Insights talks are here at the bottom of this slide. He tells us how to convert our date/time functions in CDW to make them process more efficiently. So, in this example, you’re looking at the language that Andy Kelly said to use with the example of VisitDateTime, but you could put in any date/time that you happen to be using in the warehouse and write out this same language and it will help your query run more smoothly.

Our fourth best practice for review today is using temporary tables until you’re sure you’re ready to make a permanent table. We’ll talk about this in more detail later on in the cyberseminar, but just to give you a preliminary look at it, what you’re going to do is add a hashtag sign to the beginning of the table name in order to indicate that you mean it to be temporary. That will allow SQL Server Management Studio to erase that temporary table when you close out of your query window. Thus, you’re not using up any of your storage space until you want to, and it will look something like here what you see in red into temporary table name. Put whatever name you want there. As I mentioned, we’ll go into this in more detail shortly.

Our fifth best practice to review today was the one we introduced in the last cyberseminar, the use of execution plans. So, I’m just going to give a quick reminder of the two main things you might be looking for as a sign of potential trouble in your query. So, the top left you see the red X. If you see a red X on one of the icons in the execution plan, take some time to try to figure out what might be going wrong with your query and whether or not you can write a more efficient query. Also on the bottom, you see that thick arrow indicating that you are working with a lot of rows of data and that when you run your query it may take a long time and a lot of resources. So, unless you’re ready to take that step and you’ve practiced and you know you have the query that you want and you’re ready to go for a large number of rows, you might want to back off of that size until you’re sure your query is going to lead you to the results you want and it’s worth the expenditure. Also, I mentioned last time that you can learn more about execution plans by downloading a free online book that you see here. It was also shown to you in the last cyberseminar last Monday. I’m just now learning about these execution plans as well and I find them really helpful, even if you’re only looking for some very basic things such as the wide arrow or the red X. They can really help you avoid trouble.

Lastly but not least, it’s always important to use documentation before beginning to use the CDW data. As we know from the second talk in the series, CDW: Locating its documentation, that documentation can be found in multiple sources. It can be at times somewhat complex to use for people who are relatively new to the warehouse. The obvious place to start every time is in the CDW’s own metadata. That URL is listed here. You would want to click on either one of the links you see circled to go into the metadata pages. The primary metadata page is as you see here, and the place that you are most likely to go is to execute that metadata report. Also, right below this link to execute the metadata report, you can find very helpful relief documents associated with selected domains from the CDW. So, you may spend some time reading through these documents as well as going into the metadata report itself.

Just to remind you, this is what the metadata report looks like once you execute that metadata report. It lists all of the production domains of the Corporate Data Warehouse. You may need to scroll down to find your domains of interest. Once you find a domain of interest, in this case I’m looking at the patient domain, you might do one of two main things. 1) You might click on the name of the domain itself to open up the entity relationship diagram or 2) you might click the plus sign next to the name of the domain so that you see what you are looking at on this slide, the list of tables in that domain. From there, you can start to drill down to see what’s in a table or what’s in a column and look at the available metadata for each.

If you had clicked on the name of the domain, it would open up this ER diagram that you’ve seen multiple times now. Of course, it would look small like this until you clicked on the image to enlarge it. Then you would be able to scroll around and see the tables. Each table or view becomes a box. You would be able to use the lines between them to know that they were linked to each other, and you would be able to see all of the columns in each table listed there.

Or you might come to VIReC for documentation such as the VIReC CDW Factbooks that we have begun creating. So, you can see the selected list that we’ve completed so far here. This documentation may provide additional information of a variety of sorts.

Also, you may spend some time reading through data quality reports. There are some really great tips about the CDW data, all kinds of validation studies and so forth. Now, the plan is that most of the data quality reports will probably be moved to the VHA Data Portal. That can be found under Resources and then Data Reports as you see here. Once you arrive at that page, you’ll see VHA Data Quality Reports at the top and View Reports listed underneath it. You would want to press the plus sign next to that and you’ll start to see a list of their reports drop down underneath that.

Those were our best practices that I wanted to remind you of or tell you about that are new for today’s purposes. Now, we’ll move into the practice problems for data management using SQL.

The first example for today is about creating a temporary table for you to practice your logic on. In this case, I’m going to create a temporary table of patients. I am going to have them meet certain criteria before I want to include them in my actual query results. I am going to incorporate a number of the best practices I’ve discussed just in these past few slides. First, I’m only going to select certain columns from the patient table. Then, I’m only going to work with a small number of rows until I’m sure my query is what I want it to be. Then, I’m going to sue a temporary table to store these results until I’m sure I have the cohort I want to work with, and I’m going to have a WHERE clause that eliminates some records that I’m not interested in so that I’ve reduced the number of records to only those I find meet my criteria for inclusion. In order to accomplish this goal, I used information from the Factbook for the patient domain. I open it up and I can scan through all the content of this domain and all of its tables and columns. Also, I used several reports from Data Quality which were extremely helpful, such as a CDW Possible Test Patient Flag Analysis so I can understand who was being flagged as not being a real patient. I also looked at the ones about the identifiers for patients that you find in the CDW and what their meaning and purpose are. That was extremely helpful.

Based on all of that, I wrote my query. It is what you see here, and I’m going to break it down step by step. So, taking a look at the SELECT statements, here in the SELECT statement, I’m choosing the top 10,000 rows of patient data, and I am only selecting three columns. Those three columns are patientSID, which is the CDW patient idetntifier; patientICN, which a national unique identifier for a patient and marital status because I’m going to use it in examples to come in today’s cyberseminar.

Now, taking a look at the INTO clause, this INTO clause allows me to tell SQL Server Management Studio that the name of the table will follow and that I want SSMS to put the results of my query into that table. The hashtag sign in front of the table name indicates that I want that table to be temporary so that when I close down my query window I want it to erase the table because I’m not ready to make a permanent table yet. Then, practice cohort is just the name I happened to choose for my temporary table. Just as a little piece of extra information, you can use this INTO clause to create tables that are not temporary, but you cannot use it to insert data into a table that already exists. If you didn’t want a temporary table, you wouldn’t have that hashtag sign. You would have a schema such as the one given as an example at the bottom of the slide.

Let’s break down the FROM statement which should be pretty familiar at this point. The FROM statement specifies which table the columns in the SELECT Statement and anywhere else in the query should come from. So, SQL Server Management Studio knows to look in the CDW Work Database to find the table that is called patient.patient. The first of those patients represents the schema and the second instance of the word patient indicates the name of the table. So, it’s saying to look at this table and that’s where SQL Server Management Studio or SSMS will find the columns we listed in the SELECT statement.

Let’s break down our WHERE clause which now should also be pretty familiar. The WHERE clause is, as you know, used to specify the conditions under which a row of data will be selected for use in the query and, in this case inserted into our temporary table. Using those data quality reports and the Factbook, I chose to exclude any record of patientICN (that’s that national identifier) with null or had the word “missing” or “unknown” in it. If that occurred, I don’t want it. So, I’m saying I want it only if patientICN is not null, is not containing the phrase “missing”, is not containing the phrase “unknown” and does not have a possible TestPatientFlag equal to 'y'. Because if the person is just fictitious and used to train people in how to use CPRS, they would be labeled Test Patient and certainly not any information I would want to use in my real-life study. Also, you will see around “missing” and “unknown” percent signs, and these are wild cards in SQL that allow for you to say if the field has the phrase “missing” in it or “unknown” in it with any character surrounding them, I don’t want to include that record.

Once you come into SSMS, you would first want to open up a new query window. This is after you’re at your correct server with your data in it. Then you would want to type in your query as you planned. This is after you run the test of the Execution Plan. I didn’t know if I would have time to include that in today’s talk, so I went ahead and exclude it. I am going to assume you ran your test of your Execution Plan. Then you execute your query. The only results you see when you create a temporary table like this are the number of rows affected as the output. That’s saying to you that it created a temporary table like you asked and it has 10,000 rows in it, and that’s what we would expect because we asked for the top 10,000 rows.

Before we proceed to practice problem number two, I would like to ask you one more question about yourself. How would you describe your role here in the VA?

Moderator: Thank you. So, for our attendees, you do have that second poll question up on your screen and you can check all that apply. So, do you identify as Research Investigators / PI, Data Management/Analyst, Project Coordinator, Operations / Partnered Research / QI or Other. Please note, if you are selecting Other, we will have a more extensive list of job titles available in the feedback survey at the end of the sessions. So, you might find your exact title there to be able to indicate what role you have. It looks like we have just about an 80% response rate, so that’s great. I’m going to go ahead and close out the poll and share those results. As you can see, 11% of our respondents identified as Research Investigator/PI, two-thirds Data Management/Analyst, 11% each for Project Coordinator and Operations/Partnered Research/QI, and 30% selected other. Thank you again, and I will turn the screen back over to you, Margaret.

Dr. Margaret Gonsoulin: Thank you, Molly, and thanks everyone for answering. We have a great mix today.

So, let’s move on to our second practice problem. In this problem, we will use SQL Server Management Studio and SQL to recode the values of a column in the Corporate Data Warehouse. In this case, we are going to recode the values of marital status.

First, I go to the documentation, of course. In this case, I happen to be looking at the Factbook entry for the column called Marital Status. I can find this in the Patient Domain Factbook. As I look at the values, I think that for my purposes today it might be acceptable to combine the values of common law and married into a single category of married (That may not work for all cases), also, to possibly combine the line widowed/widower with widowed into a single widowed category, and also to collapse those values that indicate a lack of information such as “questionable”, “unknown” and “zzdo not use it”. These things indicate to me that we just don’t know what the patient’s marital status is. So, I would collapse them into “missing” for my purposes.

This is what the query looks like that I came up with to run. Of course, we’ll break this down, but as you can see, we have a new clause, quite large, right in the middle of our SELECT statement, the CASE WHEN, ELSE, END AS clause. So, we’ll break that down separately from the rest of the query overall because that’s our new element.

First, taking a kind of general look at the SELECT clause in our query, we can see some of the basic patterns that we’ve been discussing. We have a selection of our three columns, our patientSID, our CDW identifier; our patientICN, the national identifier for the patient; the marital status that we discussed already; and then at the end of our case statement we’re actually creating a fourth column in our SELECT statement called “marital status recode.” Let’s break down that case expression and go through it step by step.

In this case, we see our case expression being used in our SELECT clause, and this is only one of the possible options for where in your query you can use a case expression. I put a note at the bottom of this slide about the other places. For our purposes today, we’re using it here in the SELECT statement and we’re carrying out the plan we laid out when we saw the documentation. We’re combining common law and married and we’re combining all of the values related to being a window into a single value. Also, we’re collapsing all of the values that indicated missing information into a single category of “missing” and, using the END AS marital status recode, we are creating a new column that will hold the recoded values of marital status for us.

Finally, let’s break down the INTO and FROM sections of our query. When we take INTO, we know that the use of SELECT and INTO together, which is a SELECT/INTO statement, indicates that the result set is going to be stored in a new temporary table that we are calling “marital recode” in this CASE. In our FROM statement, we see that we are using that temporary table called “practice cohort” where we cleaned up our records and selected only good records for our purposes from the patient table that we wanted to analyze or practice our queries on.

Once again, we would open up SSMS. We would press the “new query” button from the menu to open up our query window. We would type in our query in that query window. After we tested it using our Execution Plan (just timesaving today by not including that step), we would press the execute button. Finally, we would see our familiar output of just the number of rows. This is what we would expect to see because we had 10,000 rows of patient data that made it into Practice Cohort, and for each one of them we recoded “marital status.” We could take a moment to run a quick check of our recode and make sure that our plan was carried out the way that we intended it be. In this case, we are just looking at our old column, “marital status” in the SELECT statement and our new recoded column “marital status recode.” We’re collapsing them by saying let’s add a count there. Then we’re taking that information from our newly created temporary table, marital recode” and, of course because of our count, we’re going to group by marital status and marital recode. Basically, this is creating something like just a little cross-tab that allows us to collapse the values of the two columns in conjunction with each other, and you can see that output on the bottom left-hand side, so that the old value “marital status” is the first column, the recode is in the second column and then a count and the number of instances is in the third column. By looking at rows 4 and 5, we can see evidence that our plan was carried out according to what we anticipated. So “unknown” became “missing”; “missing” became “missing.” So, we’re seeing that collapse there.

Let’s move on to our third and final data management strategy using SQL that we’ll cover today. In this case, we are working with the SQL commands or language that will allow us to select a single record in instances where multiple records are available. The example that I put together today finds the most recent visit for our cohort of patients. Most patients have more than one visit on record. So, how do you find the one that is the most recent? That’s the question.

First, of course, you would again, by looking into your documentation—In this case you might come to the CDW SharePoint site and look up information about visits. Once I scroll down, I find the main Fact table of the outpatient domain, Outpat.Visit. I’m here, in particular, to look at what the partition key is. In this case, the partition key is VisitDateTime, and that’s really great for me because I want to find the most recent visit. This partition key, VisitDateTime, is the date that will help me and it will do so very efficiently. So, I am hoping to find that one. Also, I used information from the Outpatient Factbook from that same visit’s table and perhaps potentially, depending on your purposes, other parts of the domain may be helpful, too. The URL for these Factbooks is, again, at the bottom of the slide.

Let’s get started finding that most recent visit. First off, I have my cohort within my temporary practice cohort, right? I need to get the visits connected to them. So, I’m going to connect visits from Outpat.Visit, and in the process, I’m also going to specifically connect that date, VisitDateTime, because that’s how I’ll know which one is more recent. My query looks like this to be able to join in my visits. So, I have my PatientICN. That’s coming from my practice cohort in the FROM statement which you can see I renamed A. As we discussed last time, I chose to use the LEFT JOIN in this case because I wanted to keep everybody in my cohort, whether they have a visit or not. I want to keep them all, and I want to go gather any visits related to them. So, I LEFT JOIN’d my OutPat.Visit table and I renamed it B. Then, in the SELECT statement, you can see that the second column is “B.Visit/Date/Time.” So, that’s the information I collected from the visit table. I joined these two tables on the linking key patientSID, which is the primary key of my patient table, which is what is in practice cohort, and a form key in Outpat.Visit. So, I run this after I check my Execution Plan, and I see this laid out. I kept every one of my patients in my cohort which I know to be 10,000 or fewer because one patient may have more than one record if they went to more than one station. Then I added in each of their visits and I stored everything in a temporary table called OutpatVisits. If you look at the bottom left-hand side, you see the output telling me I have over 277,000 records in my new temporary table called OutpatVisits. That’s to be expected because I would expect more than one visit per patient.

Now, how do I get the most recent visit only? Well, you can do this by creating a subset, otherwise known as a partition, for each patient that will show all of their visits in that little subset of the table, if you will. It’s also important when we do this that we take note of the order and timing of the visits because we’re after the most recent visit. So, we want to tell SSMS to order these dates. Then we need to keep track of how we order them and where we will expect to find the most recent visit record in that lineup. So, with all of that in mind, I wrote this query here, and we will break this down. Most of the parts of this query should now be pretty familiar to everyone. In our SELECT statement, basically we name our columns. In our INTO we are naming our temporary tables since we are still testing out our logic. Our FROM statement says where to go to get our columns.

The new part is this ROW\_NUMBER, OVER, PARTITION BY, ORDER BY, etc. So, let’s break down that section. I’m going to work from the middle of this clause. The PARTITION BY is telling SQL Server Management Studio to break our table OutpatVisits into separate segments, one segment for each patient ICM. So, we PARTITION BY patientICN and it’s going to create little individual tables inside our data, one for each unique patientICN. The ORDER BY part of the clause tells SQL Server Management Studio to organize each row of data in those little subtables or partitions by VisitDateTime. That’s the date and time of the visit. That DESC at the end is telling SSMS to put the most recent visit in the first line of the partition, the first row of data in the partition. So, this was allowing us to say how we want the visits organized by the time and allowing us to realize that by declaring that order, we are putting that most recent visit in the first line of the partition.

ROW\_NUMBER creates a row number for each row of data in each segment or partition. So, for each patientICN, however many visits they’ve had, Row number one will be given the number one. We know now based on our own decisions about the logic that that will be the most recent visit. Then at the end of this clause you see AS. Basically, it’s giving that new column we created to hold the row numbers a name, and that name is RowNumber.

I can’t really show these results without PHI, so I took some results from a single partition and gave the PatientICN just a fake number one and entered in for PatientICN number one. You can see that on row one of this partition we find the most recent visit. So, what this allows us to do is to know that each partition will have that visit in row number one, and we can write a very simple query against our newly created table, Ordered Visits, that’s holding these partitions. We say we want to select the patient identifier and the Visit/Date/Time from our ordered partitioned table when RowNumber = 1, because this is our most recent visit.

So, to summarize, today I hope that you learned a bit more about using temporary tables for partition dates, converted functions and other best practices, that you will feel comfortable recoding a column to meet your research or data needs, and that you will be able to implement the use of partitions to sort out the many instances where you will have multiple records in the CDW. For more tips such as the ones we discussed today, you might want to look at this book, “T-SQL Fundamentals.” It’s a great book and has lots of good information.

Here is my contact information, and I will take any questions now.

Moderator: Thank you, Margaret. We only have a few questions so far. So, if the audience still has anything you would like to ask, please send your questions in. We still have ten minutes before the end of the session.

All right, Margaret, the first question refers to the second practice example you had. Can you recode as you go in one step, or do you have to create a new temporary table?

Dr. Margaret Gonsoulin: I think you can recode as you go. So, you know, it just depends on your purposes, if you want to keep it or you only want to use it in the moment. If you want to keep it around for future use in other purposes, you would want to put it either in a temporary or permanent table depending on your stage. If you were just using it in the moment for your purposes in your query and once you got your output you didn’t need it anymore, you don’t have any need to write any table, temporary or permanent.

Moderator: Thank you. Next question. Why do you have a wildcard with that single for that example?

Dr. Margaret Gonsoulin: In the recode, okay. I’m trying to find the right—I think it’s here. Because in the slide above, the value—Actually, no. I just got overzealous. I think maybe when—There used to be more values of marital status because sometimes different stations will enter things differently. I don’t really know what made me do it. I feel that there used to be more than one way of stating “single;” like “single/never married” was an entry that I somehow remembered. I thought that there were instances where there were characters after “single,” but it might just be me being overzealous and I forgot. Sorry.

Moderator: I’ve got a couple questions here about when a temporary table is created, where can you find it in SSMS?

Dr. Margaret Gonsoulin: You know, honestly, I’ve never really gone to look for where it is. I just trust that it’s gone when I shut down. But I’m sure that there is a folder somewhere on the server somewhere that’s holding it temporarily for you. I apologize. I’m not sure where it goes.

Moderator: Okay. I still have a few more questions and we have plenty of time. In the last example, would it be possible to also pull the procedure associated with the visit date?

Dr. Margaret Gonsoulin: Yes, I’m sure it would. You would probably need to get over to the OutpatV Procedure table and make a few more joins and select your columns and bring them in. But, yeah, I can’t see why not. Then it would probably extend the number of records you have because a visit record may be associated with multiple procedures and so forth. So, you would just have to keep all that in mind.

Moderator: Okay. How much performance benefit is realized with a temporary table versus a subquery?

Dr. Margaret Gonsoulin: Well, I don’t know. I mean I don’t have any sort of numbers around that. I think it is considered better practice to use a subquery in a general way, like less burdensome than a temporary table. Because obviously, you are storing information in that temporary table. The number one reason why I tend to use temporary tables is because it’s a learning tool for me. Just like many people in this audience, I’m still learning and have a great deal to learn. When I use that temporary table, I can go in and see my result sets, just select top ten or select 10,000 records from my temporary table, depending on your size of your temporary table. Then I like to look at the results because I am still learning even after near to three years exactly what data are in these tables and how the records behave. It’s my shortcoming still that I want to see the results. Whenever you have that subquery, I feel like they’re great and I hope I can get to that place, but I still want to see what my intermediate step is. That’s the reason why I’ve been using those is more to learn. I think as time goes on you become much more comfortable with the content of the table and, as you become more comfortable with the records, it’s a wise move to move toward the subquery and not have so many temporary tables and things like that. I’m still working toward that goal.

Moderator: Thank you. Going back to this question, question number two that you have in your slides, can you elaborate on the zz values and what the mean in CDW?

Dr. Margaret Gonsoulin: Yes. Typically, inside Vista, which is where much of the data inside the warehouse comes from, people in coding in CPRS, they had a lot of dropdown menus for people to select from as they’re entering data about the patient and the patient care setting. So, as I understand it, when they stopped using some of these entries that the clerks or the providers or whoever is in the information have to choose options from, let’s say they don’t want to use the name of some drug or whatever is in the dropdown boxes or the selection menu anymore, they can’t really get rid of it because they used it in the past and it’s part of records from the past. So, they have just put the letter z or zz at the beginning of the entry so that it’s all to the bottom of the list and people don’t have to continuously scroll through options that they aren’t using anymore. It’s just a time saver for data entry in the electronic health record.

Moderator: Okay, thank you. In the slide that’s titled, “A quick check of the recode,” I noticed that “single” is not listed. One would think that at least one of the Veterans would have been single. Is this a mistake?

Dr. Margaret Gonsoulin: Well, I mean it’s always possible to have a mistake. I guess it’s also always possible that you have a selection of people where no one was single. I would have to go and investigate is it true? Was no one in my group single? I didn’t do that because it was just for illustration purposes, but I think it would be wise to always spend some time checking as to whether or not something has gone wrong in the recode or in your data or any number of other sources of problems. It’s a good point.

Moderator: Okay. A couple more questions and I think we’ll wrap it up. Where can I get more information on using indexes? I’m interested in speeding up my queries.

Dr. Margaret Gonsoulin: They’ve got this information all over the place. All sorts of offices have been putting out great educational material. I would say first and foremost to go to CDW Insights Days. They have really great tips on how to advance in SQL. The book that I mentioned is also a great resource. That’s at the end of the slide set. Just talking to people who are more advanced than yourself, too, and how they use them might also be a good resource. I think lots of places—I could kind of put together a list to go in search of that particular issue if people thought it was worth it. I would be willing to do that.

Moderator: Thank you. One last question. This is about the Execution Plan. What if you can’t run the Execution Plan? Who controls the Showplan access?

Dr. Margaret Gonsoulin: Oh, I just thought it was part of SQL Server Management Studio. I’m not sure why someone would not be able to run the Execution Plan. I think that’s beyond my technical expertise, but I’d be happy to search out the answers for you on the Help Desk if you’d like.

Moderator: Thank you so much for your time, Margaret, and for answering all these questions. Those are all the questions we had today. If anyone else has additional questions, you can contact us at the VIReC Help Desk at VIReC@VA.gov.

VIReC’s next session is for our Database \_\_\_\_\_ [00:59:44] series. It is scheduled for Monday, November 7th at 1 p.m. Eastern with Linda Coke. She will talk about requesting approval for access to VA data. We hope you can join us. Molly will be posting the evaluation shortly. Molly, can I turn it over to you?

Molly: Yes, thank you, Hara, and thank you, Margaret, for coming on and lending your expertise to the field. For our attendees, I’m going to close out the session now. Please wait just a moment while the feedback survey populates on your screen. It’s just a few questions but we do look closely at your answers and it helps us improve our sessions as well as improve the program over all. Thank you for joining us today, and this does conclude today’s HSR&D cyber seminar presentation. Have a great day.

[End of audio].