Nested CTE's in SQL Server

If you have multiple CTE's in a query then you can do what's called a nested CTE. This is what Steve will go over in this video.

Transcription of Video:

So next, if you've got multiple CTE's, we can do what are considered nested CTE's, where the previous multiples we looked at, it was just two CTE's that were being or three CTE's that were being referenced from the outer query. to nest it, it's like the Russian dolls that are pronounced with Trotsky's, where you have one that's inside of another that's inside of another that's inside of another, and so on. And basically, a nested CTE query is one that it calls a previous CTE that's been declared in the list. And it can only reference itself or queries declared earlier in the query, you can't reference a CTE that's further down in the list.

Here's an example. We just have a real simple query, where we're simply selecting one as a number. We then have a second CTE, that references are the CTE one referencing CTE zero, where it just grabs the number plus one, and then CTE two references CTE one, which adds another number to it. And then if we say select star from CTE two, what's that going to give us? CTE select star from CTE two is going to select the number from CTE one and add one to it. But CTE one is going to select the number from CTE zero and add one to it. CTE zero just returns one. So the output on this is going to be three.

Let's take a look at some examples there. Here's the one we just looked at. And if we wanted to test this as we go, we could say select star from CTE zero. And we can run that and confirm that CTE zero simply returns the number one. Then we can slide this down and say select from CTE one. And confirm that Yep, that does return two references CTE zero, and then that reference, and then that starts with one and adds another one to it. And then select star from CTE to at the end, returns the number three. So these are nested. If we wanted to do something like say CTE one was going to select from CTE two, it's not going to let us do that because CTE two, you can already see the squiggly lines underneath it that say it's invalid. It doesn't exist at the time, until

after CTE one has been run.

So next, I want to take a look at a CTE here that is a little bit more complicated, but a little bit more real world than what we're looking at with some of the examples above. And what we've got here is a CTE called unused CTE's. And when we run it, it goes and looks up any unused indexes, which doesn't return any because we don't have them yet in the demo. So this is this little test database. So what I'm gonna do is I'm going to create three indexes here. So it'll show up in the list on the table departments, and they're kind of bad practice to create three indexes that have the exact same column definitions, but it's for an example. So we'll just go with it. And we're going to do is say select from all the unused C indexes. And it shows that there's two that are unused.

We have this other one that is index CTE that selects all of the indexes. And then it's going to go and look for duplicate indexes. And what this is going to do is it's going to grab the table name columns, and whatnot from selecting from tables, getting index columns, and so on. That's not going to work. I just realized that I say that, because we've got it referencing index CTE, which is up above. So the way we want to test just that dupe indexes, CTE is we can uncomment that highlight up, up up through the top of where the CTE starts. And when we run it, we can see that there's a table name called departments, that has duplicates on the ID and parent columns. But if we replace the duplicate index and CTE, we're selecting from the unused CTE and the dupe indexes CTE, and we're just joining those into two columns that will return the number of indexes and the number of duplicate indexes.

Kind of cleaning that query up into one query, you can easily run this as multiple queries the way we're using it here, but depends on how you want your code to look. So here, it gives us a table that shows us for this database. There are two unused indexes and one duplicate index shown there.