Hey, welcome to another Essential SQL Minute. Here, I'm going to show you how to code like a boss by using comments in your code. Yes, comments are important. We should be commenting our SQL, so we don't forget what it does, and we can help explain it to others.

So there's two types of comments that I want to talk about. The first is a block comment. So a block comment lets me put information inside multiple lines of text, inside a commented block. So here I can say, "Here is a block of text that serves as a comment." And then I can end it with another */. So the block comments would be /*. I can put whatever I'd like in it. And then, again, the */. So the one way to remember this, is you can start it out with a /* and then the * kind of bookends it and then a /, right?

The other type of comment's a in line comment. So if I, as you can see here, I'm declaring a table. I could say, right here, I could say, "These are the fields." And I can have a inline comment that shows me here, the comment at the end of the statement. And notice it's just in the middle of the statement. In fact, I could even put the comment in between lines like this. There are many fields to define. There's a lot of flexibility with the comments. And I think what you'll see is that as we get into more complex SQL, this becomes a good thing to use to document your code. Not only for yourself, but for others.

So I want to run this. We're going to create this product table real quickly here. And I'm showing a list of products for various cities and really what I want to be able to do here with this is not just show this list, but I want to go in and bring in a query where I would show which products are from the same city. So, comb and see Dallas and pencil, and show those on the same line. And to do that, I can use an inner join with a non-equi join clause, like so. I explained how to do this in some of my courses or on my blog. So I'm not going to go into the join, but I do want to explain how this works in the code.

So let's first run it and then check it out. I get the result I was talking about. The thing is though, is down the road if this was in a script, stored procedure, even your program, you may be looking at this going, "Now, what's this query doing again?" This is where comments come into play. So, let's write a block comment just to say what we're going to do with this code. And in here, we'll say, "Generate a list of products that are from the same city." All right. So now we have our intent of our procedure.

Now, the other thing I'd like to do is comment on this non-equi join clause here and what its purpose is. And really I can say here, "Avoid duplicate," I got to spell duplicate right though, "Entries for the same city." So what I'm avoiding here is the notion where I get comb and pencil, and then I'd see pencil and comb. I just want to list them once, right? So this will ensure that comb is always listed first and then pencil. And then it won't allow pencil to be in table one and comb in table two, since the name is always less.

So now when I run this, you'll see that I do still get the result because the comments are in the way. And one really cool thing is here is I can also use comments to kind of code out a piece of my query, just to see what happens. So you guys might be going, "Well, what were you talking about combining and getting the duplicates from the comb and pencil?" Well, now I can show you. Because if I run this now, you'll see where there's a lot more entries. And this is because I'm getting comb combined with comb, which isn't cool. Because we're trying to show not the same product in the same city, but you know, products that are from the same city. And then there's pencil and comb from Dallas. And then if you go up further into the query, you're going to see comb and pencil. So this non-equi join is serves the purpose of eliminating those duplicates.

So hopefully now you're starting to see where comments come in handy A, to help explain what our query is because down the road, we're going to forget, we're human. Secondly, you can go in and actually explain a little piece of your query because sometimes just that one little clause can be tricky and you need to explain it. And then third, you've seen how I can use it to comment out a piece of my SQL, run, see what the output is, and then take that comment out and run it again, and just see the difference. So you can see how, if that clause you wrote is actually working the way you want it to. It's a good way of helping to troubleshoot your SQL.

So hopefully you've now understand that comments aren't drudge work and that it's actually a good feature of SQL and I hope you use it so that you also can code like a boss.