On-Demand Training: Extract API Connecting Transcript

Hello everybody, welcome to the second video on the Extract API. In the first video we downloaded and installed Python, and the Python module for the Extract API. So if you haven't done those things please go back and watch the first video. The purpose of this specific video is to create more of a real world example than the script we built in the first video. We'll be connecting to real data, in this case in the form of a CSV file, and spitting out a TDE using Python.

Besides having Python and the Extract API module installed you'll also need to have these files, which if you didn't download from the first video you can download from this webpage. They should have these three files, the TrivialExample_Finished is the one we created in the first video. csv2TDE is the finished example of what we'll create today, so you can use that if you get lost along the way in today's video. And the SuperStoreCSV is what we'll actually be reading from to convert to a Tableau data extract. Ok let's go ahead and get started.

Just as we did in our first video we'll be using the application idle to create our script. So if you have Python installed you should be able to open the Start menu, type idle and open that application. To create a new script you'll need to go to File New, Control+N, and then file save as. We'll call this our csv2TDE.py.

Now be sure to save this in the same directory where SuperstoreCSV lives. Now we've created our new file we're ready to get started. The first thing we do in almost every Python script is to import the modules that you'll need to use. In this case of course we need the Extract API module Data Extract, and I like to give it an alias as tde. We'll be doing manipulation of files again so we'll need to import the os module. In one of our data types is the datetime so this module helps you manipulate those. Finally because we are connecting to a CSV we need to import the module called csv which will aid us in reading from the CSV file. This csv module is included in Python's library but very often is the case that you'll be connecting to a data source that does not have a built-in module. For example if I was connecting to Microsoft SQL Server, there's no built-in module for that. So it would just be a matter of opening up my browser and searching for something like python module SQL server. Here is a Python module called pymssql which will allow you to read from a SQL server. Or if you were reading from something like MongoDB. There is a module called pymongo. Pretty much any data source out there is going to have some module probably created by the Python community that will aid you in reading from it. In this case, once again, we don't need to do that because the CSV module is already included. Ok we've imported all of the required modules. Now we'll get started in actually connecting the data and creating our extract.

I'll follow pretty similar steps to the first video. Step one is to actually create the extract. Also as part of the first step we'll go ahead and get this CSV ready and open it up. On the first video what we did is we wanted to simulate a full refresh. The way we did that is if the extract already existed we'd simply delete it and start from scratch. However you can also treat your Extract API scripts as an incremental refresh so I already have an extract. I want to add rows to it. The way you do that is you simply open the existing extract and then insert rows just as we did normally, those will be appended to the bottom of the extract. So because we don't need to delete the file if it already exists, we'll just go ahead and open it whether it exists or not. If it doesn't exist it'll be created for us and if it does exist it will just be opened.

So tde.Extract dataextract.Extract will create a new extract file for us, or open the existing one. We also need an object for us to read from the CSV, so we'll called that csv.reader. Specifically we want to open the CSV and we need to pass in the name of the CSV which in this case is SuperStoreCSV.csv. These are all case sensitive so make sure you get all of the capitalizationcorrect. What we're going to be doing to this is reading from the CSV, we need to tell

it what kind of mode we're opening it in, that's rb mode which is just to read from. There's optional arguments here, there's delimiter, and just to be safe we'll be sure to specify explicitly the delimiters. In case that's hard to see, that's single quote, then inside a double quote and then closed with a single quote.

Now earlier what we did is we went ahead and created the table inside the extract. However what we need to do in this case is check if the extract already exists. That is if it already has a table and in that case we can skip the part of the script where we define the table definition and add the table. So essentially what we'll do is we'll check does the TDE file that we just opened or created have a table. If this extract already exists then this will return true, in which case we don't need to define the table definition. We can simply grab the table with tdefile.openTable('Extract'). We'll also be able to grab the table definition which we'll need to define new rows with table.getTableDefinition. As always these are always case sensitive.

If this returns false, if tde.hasTable returns false then essentially this is a new extract and we need to both define the table definition, all of the metadata inside and create the new table. That is step 2 from the previous video, create the table definition. Now to start with we'll create a blank table definition with tde.TableDefinition(). One thing to note before I get too far past this is that Python has significant widespace that means this needs to be a tab and this needs to be un-indented. So if, and then you have a colon, everything indented under the if, falls under the if, and then we create an else and everything indented under the else will be evaluated with the tde file that has the table returns false.

So now we just need to go through one-by-one each of our columns and define what type, what's the name of that column and what data type it is? The first column we'll create is called Row ID. And the type of it is tde.Type.CHAR_STRING. This line is actually repeated in the previous video. The next column is Order Date. The type of it is DATE. Next is sales. Which is a DOUBLE. So is profit. Next we have our Customer Name. Then we have our Zip Code which is an INTEGER. Lastly we'll include the Product Category which also of course is a CHAR_STRING.

Now in this specific script we know exactly what columns are in there and we're manually inserting them so we know what they are and we know what data types they are. Often it would be the case that you'd want to do this part dynamically, that is you don't know when you're writing the script which columns are going to be in there or data types. There's a variety of ways to do that dynamically. One way would be to do some detection of the data type, and the other would be to read a schema.ini for CSVs and XLSs which will define essentially as a schema for the file and you can read from that and loop through it and define the columns based on that. If you look in the documentation for the Extract API which is in the zip file when you download the Extract API there is an example of a CSV TDE where we don't define the columns manually as you see here but instead we define them by reading through the schema.ini and looping through and adding the columns.

The first step after we define the tableDef is to create a table in the image of the tableDef and stored in a variable called table. As always with Extract API the first argument to add a table is just the word Extract and then the second argument is the table definition variable that we just filled out up above. Now note that if TDE file.hasTable has returned true, that is if the extract already existed we would have skipped all of that stuff and we would have already had our table and table definition without having to define them as we did in steps 2 and 3.

Ok we have everything defined. Step 4 whether the extract exists and we're appending to it or whether it's new, either way is to loop through the CSV, to grab all of the data, put it into rows and insert those rows into the table variable. So the first thing to do that step is to create this newrow variable. We'll be using this over and over through each iteration of our loop. We create essentially a blank row but it serves as a skeleton for a row defined by the table definition so we pass in the tableDef into the argument of that row creator. Now the way we'll read through this CSV and grab the data we need is through this csvReader object. The only thing we need to do to prepare that is to go to the second row because the first row in this CSV is the headers, we don't want to include those when we read them, when we read the CSV. Now we're ready to start looping through the CSV reader and adding the data to our extract. So we'll loop through each line in the CSV reader and grab all of the data. Now the way we grab all of the data is… or the way we put all of the data into our Tableau Data Extract is first by putting it into this new row object with newrow.set and then the name of the data type that we're adding. So the 0th column is a charstring so we need to call it newrow. setCharString. The first argument is the index of the column so this is the 0th so we pass 0. We want this to be a string so we'll be sure to cast it to a string, and what we want to actually pass in is a string representation of the 0th column from the CSV so line is our current line in our CSV, we grab our 0th column which we happen to know is Row ID, and then we insert it into the 0th column of this newrow variable. Then we'll repeat the same step for each of the other 6 columns. The next column, the first column is type date. Now we need to do a little bit of extra work to get this from a timestamp, sorry a datestamp, 12, 31, 19, 82 or whatever and put that into a Python date object so that we can then put that into the newrow object. So we imported this module datetime and that has an object called datetime which allows us to call this function strptime. That's a way to kind of parse the datestamp into the correct version. What datestamp do we want to grab that's the first column of the current line in the CSV. Then strptime allows you to specify the format which looks like this, so two-digit month and a two-digit day and then a four-digit year. So we'll grab that and then strptime will then return a Python date object and put it into this variable called date. And what will we do with that our date object, we'll insert it into our TDE with newrow.setDate. First argument is the column index which is 1. The way that you do setDate is by specifying the year the month and the day. If you were doing setdatetime then you would need two more arguments. In this case we just need the year which is from this date object, date.year, date. month and date.day. The rest of the columns are a little more straight forward we just grab them from the CSV and put them into the TDE. So the next one is the Sales, which is a double, so we'll do a newrow.setDouble 2. We want to cast this to a floating point. Line, set to the second column of the current line. The next column is Profit which is also a double so I'll just copy and paste newrow.setDouble and I'll change both of these 2s to 3s.

Next column is our Customer Name which is a string so that's tableDef. Sorry newrow.setCharString. Column number 4. And no casting needed because CSV reader will just read this as a string. Next is our Zip Code which is an integer. In this case we will cast what CSV reader to an integer just in case it reads it as a string or if it's stored as a string. The last one is another string product category.

So we've created our newrow variable, we defined it by our table definition and then we inserted the actual data into each of those columns. That newrow now is ready to be inserted into the extract, specifically into the table. So table. insert. Insert that new row. So this will loop through one line at a time and grab all of the data from the CSV and insert

it into the TDE. That's all we need to do in the loop, so after this loop is done executing the table it should be ready to go. Once again always be sure to un-indent when you're done with your loops. Then the last step is a simple one, but it's an important one, close the TDE. You'll notice that Python, or idle by default had indented. Just make sure that you're un-indented otherwise it'll try to close the TDE file in every iteration of the loop.

So I'll go ahead and save that. Then I'll navigate to it in my file system, here it is csv2TDE. Just make sure it's there. The easiest way to run it is to in idle, go to run, run module. If it executed correctly I shouldn't see any errors or anything like that. Then I can go back to my file system, see that the SuperStoreCSVExtract is there. Now I can double-click on it. Open it up in Tableau, we can see that it looks like the data is correct. Then I can open it up and then start to do some analysis. Just to show the idea that if we ran this script again it wouldn't create it from scratch, I'll put on labels here and just keep these numbers in mind, 5.1 million, 3.7 million, 5.9 million. Now I'll come back to idle, and I'll hit run, run module, I'll go ahead and run through that again. Now if I open up SuperStoreCSVExtract we can go back to Tableau and bring in our product categories again and as we can see, they're all double because what we did is we just read the data again. Of course in real life you wouldn't just duplicate the data but what you might do is have some logic where you're only grabbing new data or maybe grabbing this month's data that happens to live in its own CSV and only putting in those new rows of data into the extract.

So that's pretty much it. Hopefully this example taught you how to create more of a real life example of using the Extract API. Actually connecting to data, whether it's using the built-in CSV module or whatever module you're able to find on the internet to fit the needs of your data source. Then creating the table definition either manually or dynamically and finally adding the data as you read from it into the Tableau Data Extract. Also I hope you understand how it works if you want to insert rows instead of just deleting the extract and starting from scratch. That's it from this video. Hope you've learned a little about the Extract API. Have a good day, bye.