SQL Query Advanced Topics

Conference 2020

Session Description: For Database Administrators (DBA). Topic include multi-table queries, methods for exporting data, SELECT INTO, and tips for finding duplicates / orphaned records.

1. **SQL Query**
   Using SELECT Sub-Queries
2. **Multi-Table Joins**
3. **Finding Duplicate / Orphaned Records**
4. **Joining Across Databases**
5. **Using Variables**
6. **SQL Views**
7. **sp_who**
8. **Data Export Methods**
   Excel
   SQL Agent Job
   PowerShell

Using SELECT Sub-Queries

There are many uses for SELECT queries, but most don't realize that it can go beyond just displaying a simple list of data. When you combine multiple SELECT queries into one, you get a powerful tool that can help find related data in a multitude of tables. For starters, SELECT sub-queries can assist with comparing lists of data quickly and easily.

```
SELECT STU.SC, STU.ID, STU.LN, STU.FN
FROM STU
WHERE STU.DEL = 0 AND STU.ID IN (SELECT CSE.ID FROM CSE WHERE CSE.DEL = 0 AND CSE.DI <> '')
```

Depending on the complexity of your query, you may find that running a multi-table join can accomplish the same result.

```
SELECT STU.SC, STU.ID, STU.LN, STU.FN
FROM STU INNER JOIN CSE ON STU.ID = CSE.ID
WHERE STU.DEL = 0 AND CSE.DEL = 0 AND CSE.DI <> ''
```

Multi-Table Joins

Quick reminder of all our common Join types:

- **INNER JOIN**: Returns data where rows exist in both left and right tables
- **LEFT JOIN**: Returns all rows in the left table with any records that match from the right table
- **RIGHT JOIN**: Returns all rows in the right table with any records that match from the left table
- **FULL JOIN**: Returns all rows in both the left and right tables.

A commonly used query is to list all a student's classes that include the teacher name, section number and period. This requires data from more than 2 tables which adds a new level of complexity to our SQL query. In this statement, we will be pulling data from STU, SEC, MST, and TCH.

```
SELECT STU.ID, STU.LN, STU.FN, TCH.TE, MST.PD, SEC.SE
FROM STU INNER JOIN SEC ON STU.SC = SEC.SC AND STU.SN = SEC.SN
         INNER JOIN MST ON MST.SC = SEC.SC AND MST.SE = SEC.SE
         INNER JOIN TCH ON MST.SC = TCH.SC AND MST.TN = TCH.TN
WHERE STU.DEL = 0 AND SEC.DEL = 0
ORDER BY STU.LN, MST.PD
```

NOTE: Keep in mind that, for the most part, you can change the format of the query to make it easier to read for you. This is only one of many examples to do multiple table joins.

Finding Duplicate or Orphaned Records

The most common way to find duplicate records is by using the **HAVING** command with **COUNT**. In the example below, you can see that we are looking for any specific duplicate PWA records.

```
SELECT EM, COUNT(EM)
FROM PWA
WHERE DEL = 0
GROUP BY EM
HAVING COUNT(EM) > 1
```

Be careful of what you're "counting". Most of the time, when you're querying for duplicate records, you're actually looking for records with duplicate primary keys, not necessarily the entire record. In the example, we are only looking for duplicate EM values.

Another way to find duplicate records is to use a self-join. A self-join looks like a typical INNER JOIN, but joins the same tables together using alias names.

```
SELECT STU1.ID, STU2.ID, STU1.SC, STU2.SC, STU1.SN, STU2.SN
FROM STU AS STU1 INNER JOIN STU AS STU2 ON STU1.ID = STU2.ID
WHERE STU1.DEL = 0 AND STU2.DEL = 0 AND STU1.SC <> STU2.SC
```

Joining Across Databases

Staging Databases

Having a separate database outside of your Aeries database can be very beneficial. You can use this database to load data from another system and manipulate it before importing into your live Aeries database. You can also use this database to develop functions that can become part of your data management process.

Another benefit is the use of cross database views or synonyms that can link data from your live Aeries database within your staging database.

Self-Joins can help you update data between more than one school and even update separate databases. In the example below, we update the student record in the future year with current year NIT, NTD, and NRS fields:

```
SELECT S1.SC, S1.SN, S1.ID, S1.TG, S1.NIT, S1.NTD, S1.NRS, S2.SC, S2.SN, S2.ID, S2.TG, S2.NIT,
S2.NTD, S2.NRS
FROM DST14000AERIESDEMO_ID.dbo.STU AS S1
  INNER JOIN DST15000AERIESDEMO_ID.dbo.STU AS S2 ON S1.ID = S2.ID
WHERE S1.DEL = 0 AND S2.DEL = 0 AND (S1.NIT <> S2.NIT OR S1.NTD <> S2.NTD OR S1.NRS <> S2.NRS)
```

If the data returned from this **SELECT** statement is correct, we can easily modify the first two lines with an **UPDATE** statement:

```
UPDATE S2
SET S2.NIT = S1.NIT, S2.NTD = S1.NTD, S2.NRS = S1.NRS
FROM DST12000AERIESDEMO_ID.dbo.STU AS S1
  INNER JOIN DST13000AERIESDEMO_ID.dbo.STU AS S2 ON S1.ID = S2.ID
WHERE S1.DEL = 0 AND S2.DEL = 0 AND (S1.NIT <> S2.NIT OR S1.NTD <> S2.NTD OR S1.NRS <> S2.NRS)
```

Using Variables

Variables can make your queries more dynamic and lessen the occurrence of typos. Below is a simple script to demonstrate the concept of using variables. The first step in using variables is to name the variable and declare a data type:

```
DECLARE @TCHID AS INT
```

Here we have created a variable named @TCHID.

The second step is to set our variable to either a static value, or initialize it to a default value (like 0).

```
SET @TCHID = 12345
```

We've now set our variable to specific value. Now that it has been declared and set, we can now utilize the variable name without having to retype the actual data:

```
SELECT * FROM TCH
WHERE ID = @TCHID
```

```
SELECT * FROM STF
WHERE ID = @TCHID

SELECT * FROM UGN
WHERE [SID] = @TCHID

SELECT * FROM MST
WHERE DEL = 0
 AND TN IN (SELECT TN FROM TCH WHERE TCH.DEL = 0 AND MST.SC = TCH.SC AND TCH.ID = @TCHID)
```

The results would return any TCH, STF, or UGN record where our variable ID matched TCH.ID, STF.ID, or UGN.SID. In the database, these records should be related, particularly of Aeries.net Teacher Portal Accounts. The result set would be something we could use to compare these records without having to type out the ID repeatedly.

Variables can be used in increasingly difficult queries that iterate through multiple databases and tables at a time. Once you are more comfortable using variables, you can begin to use them to do complex change queries and other more advanced statements.

SQL Views

A SQL View is a virtual "table" of data that can be queried like a regular table. In most cases, the view contains specific columns of data from multiple tables. However, after the view has been completed, a simple query will allow for the display of its result set.

In this example, we create a view called "Teacher_Portal_Accounts". It includes data from the STF, TCH, and UGN tables. After the view is created, we can query "Teacher_Portal_Accounts" to get the **current** data in those tables. Whenever the view is queried, the data that is returned is "real-time" data.

In the example, if data is returned from the given SELECT statement, then it runs the following command. If no data is returned, the following command is skipped.

```
CREATE VIEW [dbo].[Teacher_Portal_Accounts]
AS
SELECT STF.ID AS 'Perm ID', TCH.SC AS 'School Code', TCH.TN AS 'Teacher #', TCH.TE AS 'Teacher
Name', UGN.[UID] AS 'Portal Acct ID', UGN.UN AS 'Username', UGN.XD AS 'Expiration Date', UGN.PLC AS
'Password Last Changed Date'
FROM TCH INNER JOIN STF ON TCH.ID = STF.ID
         INNER JOIN UGN ON STF.ID = UGN.[SID]
WHERE STF.DEL = 0 AND TCH.DEL = 0 AND UGN.DEL = 0
```

Two of the most common uses of a view for an Aeries database is for providing limited access to 3rd party vendors and running queries that are complex or run often.

sp_who2

The stored procedure sp_who2 can provide valuable troubleshooting information in diagnosing SQL server performance and identifying processes for a particular user. If the base command is run using the following command:

EXEC sp_who2

the result would show all current users, sessions, and processes currently on the database engine.

A login name or session can be specified to provide more detailed information for a particular user or session by supplying the login name or session ID after the previous command as follows:

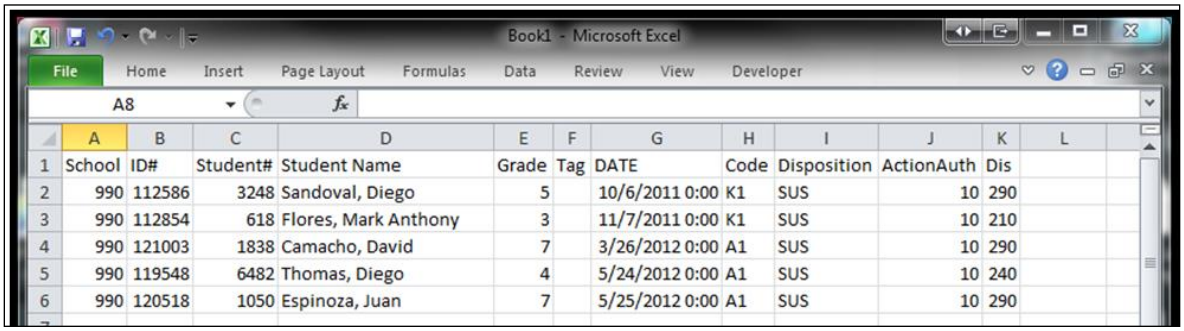
EXEC sp_who2 bob.smith

This command would display all active connections to the SQL database engine for user bob.smith.

Data Export Methods

Microsoft Excel:

One of the easiest ways to export data quickly is to use Microsoft Excel. Write an SQL statement that uses alias names (stu.sc AS 'School'). This will cause the column name to be the alias name.



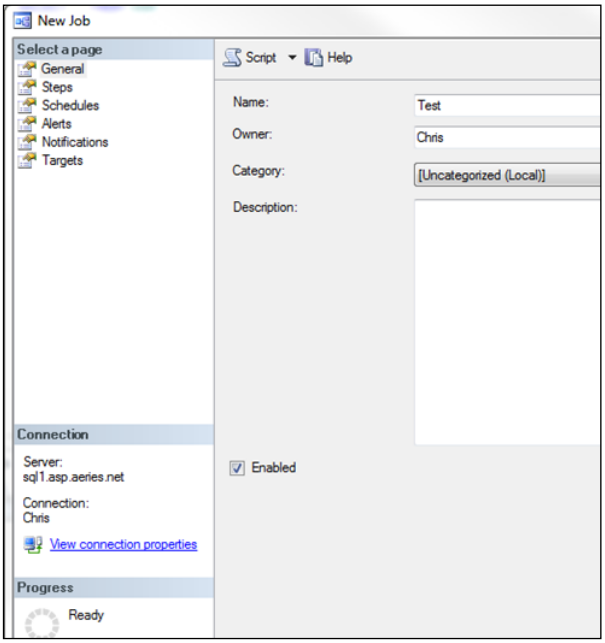
	A	B	C	D	E	F	G	H	I	J	K	L
1	School	ID#	Student#	Student Name	Grade	Tag	DATE	Code	Disposition	ActionAuth	Dis	
2	990	112586	3248	Sandoval, Diego	5		10/6/2011 0:00	K1	SUS	10	290	
3	990	112854	618	Flores, Mark Anthony	3		11/7/2011 0:00	K1	SUS	10	210	
4	990	121003	1838	Camacho, David	7		3/26/2012 0:00	A1	SUS	10	290	
5	990	119548	6482	Thomas, Diego	4		5/24/2012 0:00	A1	SUS	10	240	
6	990	120518	1050	Espinoza, Juan	7		5/25/2012 0:00	A1	SUS	10	290	

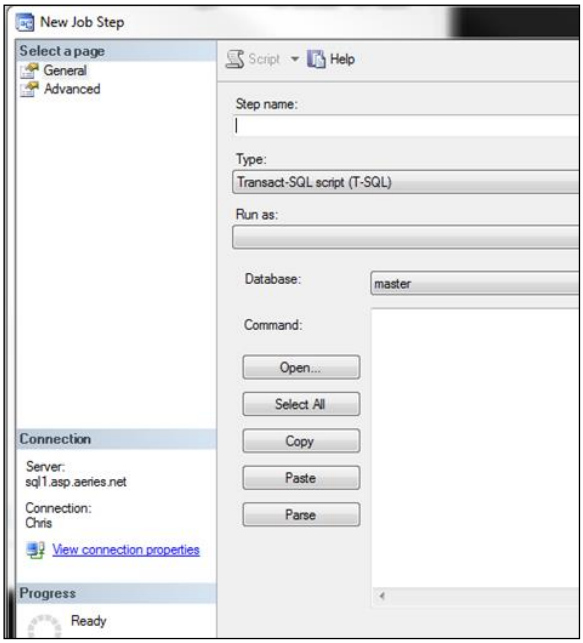
```
SELECT
       STU.SC AS 'School',
       STU.ID AS 'ID#',
       STU.SN AS 'Student#',
       (STU.LN + ', ' + STU.FN) AS 'Student Name',
       STU.GR AS 'Grade',
       STU.TG AS 'Tag',
       ADS.DT AS 'DATE',
       ADS.CD AS 'Code',
       DSP.DS AS 'Disposition',
       ADS.AA AS 'ActionAuth',
       CSE.DI AS 'Dis'
INTO ADS_EXTRACT     --TABLE NAME
FROM STU INNER JOIN ADS ON STU.ID = ADS.PID
         INNER JOIN DSP ON DSP.PID = ADS.PID AND DSP.SQ = ADS.SQ
         INNER JOIN CSE ON STU.ID = CSE.ID
WHERE STU.DEL = 0 AND ADS.DEL = 0 AND CSE.DEL = 0 AND STU.TG NOT IN ('*', 'N') AND STU.SC <> 999
AND CSE.DI > 0 AND DSP.DS IN ('SUS', 'EXP', 'SUSP') AND
ADS.DT > '08/01/2017'
```

Note: You can add an INTO statement between the end of the SELECT fields and the FROM statement to put the data into a temporary table for further use. Take the results in SQL and copy them to excel.

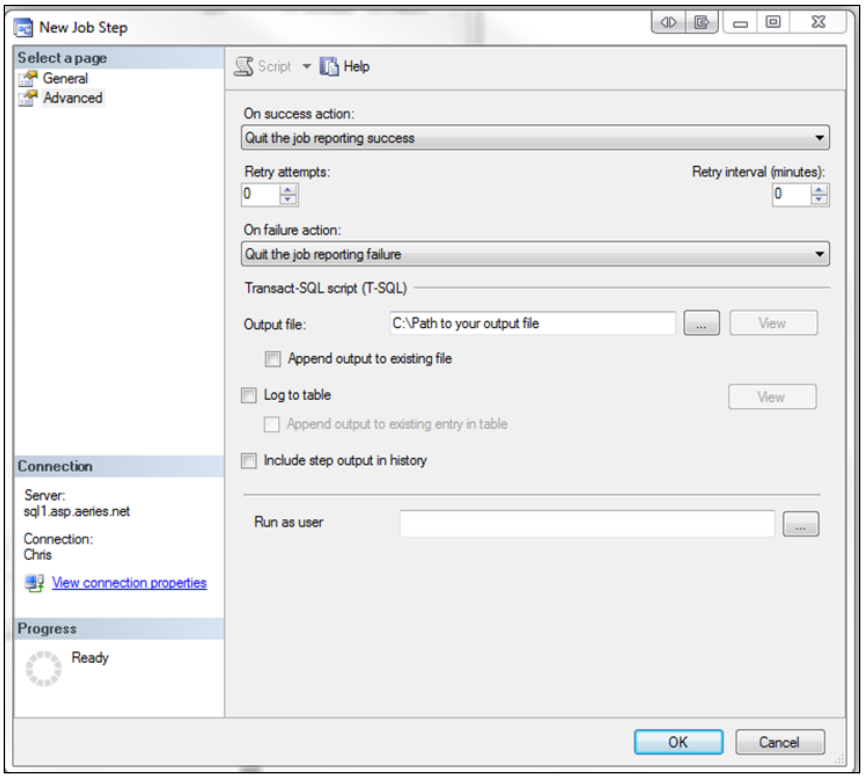
SQL Agent Job:

- Create a job in SQL Management Studio (below left).
- On the general tab, add a step that will execute an SQL statement (below right).





On the Advanced tab set the path to the output file



This is a quick and easy way to export data automatically, but you are limited to the format of the exported file.

Microsoft PowerShell:

Now one of the most commonly used methods for extracting data is to use PowerShell. After downloading a few quick tools from Microsoft (below) you can extract data from your database and place it into an export file in a single step.

- Powershell version 5, run $PSVersionTable to check your version
- Run Install-Module SqlServer

Here is an example of a script that would export a csv formatted file:

Invoke-Sqlcmd -serverinstance localhost -database DST16000AeriesDemo_ID -username test -password test -query "SELECT SC, ID, LN, FN, BD FROM STU" | Export-Csv C:\directoryname\filename.csv -NoTypeInformation

The highlighted sections show the areas where you would need to supply your specific information for login, query, and output path.

If you want to add even more power behind your script, you can add functionality to send your files via SFTP, copy them to another location, or zip the files before sending them.