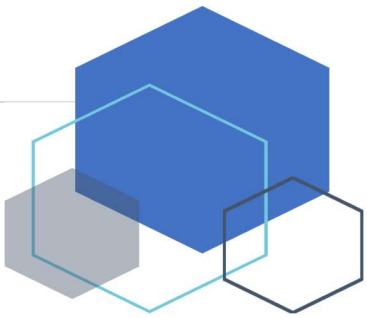

Table Validation

Advanced SQL Puzzles

Scott Peters

https://advancedsqlpuzzles.com/

Last Updated: 10/06/2022

This script compares two identical tables using a FULL OUTER JOIN and returns a table with comparison results. This can be used to audit the differences between two datasets and display all columns which do not match.

Link to the GitHub repository below
AdvancedSQLPuzzles/Database Tips and Tricks/Full Outer Join at main

Inside the GitHub repository you will find the following SQL scripts:

1) **Table Validation Demo Tables.sql**
   - A script that creates the test data for demo purposes.

2) **Table Validation Part 1.sql**
   - A script that inserts the table information to be audited.

3) **Table Validation Part 2.sql**
   - A script that creates a dynamic SQL statement and executes.

To execute a quick demo, execute the above scripts in order. **Note, the tables must have the exact same columns for this script to work.**

Step 1

Run the script: Full Outer Join Create Sample Data (Testing).sql

The following tables will be created for our demo; **dbo.MyTable1** and **dbo.MyTable2**.

dbo.MyTable1

TableID	CustID	Region	City	Sales	ManagerID	AwardStatus
1	453	Central	Chicago	$ 71,967.99	1324	Gold
2	532	Central	Des Moines	$ 65,232.42	6453	Gold
3	643	West	Spokane	$ 44,981.23	7542	Silver
4	643	West	Spokane	$ 44,981.23	7542	Silver
5	732	West	Helena	$ 15,232.19	8123	Bronze
6	732	West	Helena	$ 15,232.19	8123	Bronze

dbo.MyTable2

TableID	CustID	Region	City	Sales	ManagerID	AwardStatus
1	453	Central	Chicago	$ 71,967.99	1324	Gold
2	532	Central	Des Moines	$ 65,232.00	6453	Gold
3	643	West	Spokane	$ 44,981.23	7542	Bronze
4	643	West	Spokane	$ 44,981.23	7542	Bronze
5	898	East	Toledo	$ 53,432.78	9242	Silver

Looking at this data we can see several differences between the tables; some values are different, the tables have different records, and the tables have duplicate rows. The Full Outer Join script will account for these scenarios.

Step 2

Next, run the script Full Outer Join Insert Table Information (Part 1).sql

This script will populate the table **##TableInformation** with the following values from our Test script.

ColumnName	Value
LookupID	1
Schema1Name	dbo
Schema2Name	dbo
Table1Name	MyTable1
Table2Name	MyTable2
Exists1	CONCAT(t1.CustID, t1.Region, t1.City)
Exists2	CONCAT(t2.CustID, t2.Region, t2.City)

The table **##TableInformation** is used to store the schemas, table names, and the join conditions between the two tables.

You will need to modify this table according to the tables you want to audit.

Notes:

- The join between the two tables is created in the fields **Exists1** and **Exists2**, where a **CONCAT** function is used to group the values together.
- If the join criteria is on multiple columns, use a **CONCAT** function to join the columns together. You can remove the **CONCAT** if the join criteria is on one column.
- Note the **t1** and **t2** difference between the fields **Exists1** and **Exists2.** This can be easily overlooked when inserting your join criteria.
- You can store multiple reconciliation scenarios in this table, simply input the values and increment the LookupID value.

Step 3

Lastly, run the script Full Outer Join Create Dynamic SQL (Part 2).sql

This script will generate an SQL statement that compares the tables using a FULL OUTER JOIN and then execute it dynamically.

Notes:

- Modify the **@vLookupID** variable appropriately. This variable is used to lookup the table information in **##TableInformation.**
- The dynamic SQL statement will be saved in the table **#SQLStatementFinal**. You can review this SQL statement and modify as needed.
- The results from the dynamic SQL statement will be saved in the temporary table ##<@vTableName1>_TemporaryTemp (which will be **##MyTable1_TemporaryTable** in our example).
- For each column there is a "_Compare" field created that will be populated with a 1 if the fields are unequal, and a 0 if they are equal.
- The field "Compare_Summary" in the first column of the result set and gives an overall summary if the record matches between the two tables.
- The result set also has information on which fields do not exist in its partner table, distinct counts, etc…. Please review the SQL statement in **#SQLStatementFinal** to get a full understanding of all fields created.

To best understand the SQL generated from the script, review the SQL statement in the table **#SQLStatementFinal.**

```
WITH CTE_SQLStatement AS (
SELECT 'Start Compare-->' AS CompareStart
,'dbo,MyTable1' AS TableName1
,'dbo,MyTable2' AS TableName2
,'VDSPETERS\SQLEXPRESS' AS ServerName
,'CONCAT(t1.CustID, t1.Region, t1.City)' AS JoinSyntax_Table1
,'CONCAT(t2.CustID, t2.Region, t2.City)' AS JoinSyntax_Table2
,CASE WHEN CONCAT(t1.CustID, t1.Region, t1.City) = '' THEN 1 ELSE 0 END AS NotExists_Table1
,CASE WHEN CONCAT(t2.CustID, t2.Region, t2.City) = '' THEN 1 ELSE 0 END AS NotExists_Table2
,(SELECT COUNT(*) FROM dbo.MyTable1) AS Count_Table1
,(SELECT COUNT(*) FROM dbo.MyTable2) AS Count_Table2
,(SELECT COUNT(DISTINCT CONCAT(t1.CustID, t1.Region, t1.City)) FROM dbo.MyTable1 AS t1) AS DistinctCount_Table1
,(SELECT COUNT(DISTINCT CONCAT(t2.CustID, t2.Region, t2.City)) FROM dbo.MyTable2 AS t2) AS DistinctCount_Table2
,(SELECT COUNT(*) FROM dbo.MyTable1 AS t1 INNER JOIN dbo.MyTable2 AS t2 ON CONCAT(t1.CustID, t1.Region, t1.City)
= CONCAT(t2.CustID, t2.Region, t2.City)) AS DistinctIntersect
,CASE WHEN t1.[AwardStatus] IS NULL AND t2.[AwardStatus] IS NULL THEN 0 WHEN t1.[AwardStatus] = t2.[AwardStatus]
THEN 0 ELSE 1 END AS [AwardStatus_Compare]
,t1.[AwardStatus] AS [AwardStatus_Table1]
,t2.[AwardStatus] AS [AwardStatus_Table2]
,CASE WHEN t1.[City] IS NULL AND t2.[City] IS NULL THEN 0 WHEN t1.[City] = t2.[City] THEN 0 ELSE 1 END AS
[City_Compare]
,t1.[City] AS [City_Table1]
,t2.[City] AS [City_Table2]
,CASE WHEN t1.[CustID] IS NULL AND t2.[CustID] IS NULL THEN 0 WHEN t1.[CustID] = t2.[CustID] THEN 0 ELSE 1 END AS
[CustID_Compare]
,t1.[CustID] AS [CustID_Table1]
,t2.[CustID] AS [CustID_Table2]
,CASE WHEN t1.[ManagerID] IS NULL AND t2.[ManagerID] IS NULL THEN 0 WHEN t1.[ManagerID] = t2.[ManagerID] THEN 0
ELSE 1 END AS [ManagerID_Compare]
,t1.[ManagerID] AS [ManagerID_Table1]
,t2.[ManagerID] AS [ManagerID_Table2]
,CASE WHEN t1.[Region] IS NULL AND t2.[Region] IS NULL THEN 0 WHEN t1.[Region] = t2.[Region] THEN 0 ELSE 1 END AS
[Region_Compare]
,t1.[Region] AS [Region_Table1]
,t2.[Region] AS [Region_Table2]
,CASE WHEN t1.[Sales] IS NULL AND t2.[Sales] IS NULL THEN 0 WHEN t1.[Sales] = t2.[Sales] THEN 0 ELSE 1 END AS
[Sales_Compare]
,t1.[Sales] AS [Sales_Table1]
,t2.[Sales] AS [Sales_Table2]
,CASE WHEN t1.[TableID] IS NULL AND t2.[TableID] IS NULL THEN 0 WHEN t1.[TableID] = t2.[TableID] THEN 0 ELSE 1
END AS [TableID_Compare]
,t1.[TableID] AS [TableID_Table1]
,t2.[TableID] AS [TableID_Table2]
FROM
dbo.MyTable1 t1 FULL OUTER JOIN
dbo.MyTable2 t2 ON  CONCAT(t1.CustID, t1.Region, t1.City) = CONCAT(t2.CustID, t2.Region, t2.City)
) SELECT
CASE WHEN
AwardStatus_Compare = 1 OR City_Compare = 1 OR CustID_Compare = 1 OR ManagerID_Compare = 1 OR Region_Compare = 1
OR Sales_Compare = 1 OR TableID_Compare = 1
 THEN 1 ELSE 0 END AS [Compare_Summary]
,*
INTO ##MyTable1_TemporaryTable
FROM CTE_SQLStatement;
```

Above is the statement generated from our demo. If you are receiving errors, you can review the generated SQL statement and modify as needed.

THE END