Julia York
09-NOV-2025
Foundations of Databases and SQL Programming
Assignment 05

SQL Joins

Introduction

SQL JOINs are used to combine data from two or more tables based on a match condition established in the JOIN clause that specifies how the tables are related. Three types of JOINs are INNER JOINs, OUTER JOINs, and CROSS JOINs. The JOIN type determines which rows are returned from the tables within the JOIN clause. INNER JOINs return all matching rows, OUTER JOINs return both matching and un-matching rows, and CROSS JOINs return all possible combinations of rows across both tables. A self-join is used to join one table to itself. Self-joins can be INNER JOINs, OUTER JOINs, or CROSS JOINs.

SQL Joins

In SQL, a JOIN clause is used to connect and extract data from different tables. The relationship between the tables is established in the JOIN clause with the ON keyword, which defines the match condition connecting the tables. For example, the Microsoft Northwind database contains a Categories table and a Products table (Figure 1). Both tables have a CategoryID column. The CategoryID columns can be used in a JOIN clause as the match condition to define the relationship between the two tables (Figures 2 and 3).

	CategoryID	CategoryName
1	1	Beverages
2	2	Condiments
3	3	Confections
4	4	Dairy Products
5	5	Grains/Cereals
6	6	Meat/Poultry
7	7	Produce
8	8	Seafood

	ProductID	ProductName	CategoryID	UnitPrice
1	1	Chai	1	18.00
2	2	Chang	1	19.00
3	3	Aniseed Syrup	2	10.00
4	4	Chef Anton's Cajun Seasoning	2	22.00
5	5	Chef Anton's Gumbo Mix	2	21.35
6	6	Grandma's Boysenberry Spread	2	25.00
7	7	Uncle Bob's Organic Dried Pears	7	30.00

Figure 1. Categories table (top) and Products table (bottom) from the Microsoft Northwind database. Both tables contain a CategoryID column.

```
Select * From Categories
    Join Products on Categories.CategoryID = Products.CategoryID;
    Go
```

Figure 2. SELECT statement with JOIN clause connecting the Categories table and Products table from the Microsoft Northwind database. The CategoryID column from each table is specified as the connecting column using the ON keyword.

	CategoryID	CategoryName	ProductID	ProductName	CategoryID	UnitPrice
1	1	Beverages	1	Chai	1	18.00
2	1	Beverages	2	Chang	1	19.00
3	2	Condiments	3	Aniseed Syrup	2	10.00
4	2	Condiments	4	Chef Anton's Cajun Seasoning	2	22.00
5	2	Condiments	5	Chef Anton's Gumbo Mix	2	21.35
6	2	Condiments	6	Grandma's Boysenberry Spread	2	25.00
7	7	Produce	7	Uncle Bob's Organic Dried Pears	7	30.00
8	2	Condiments	8	Northwoods Cranberry Sauce	2	40.00
9	6	Meat/Poultry	9	Mishi Kobe Niku	6	97.00
10	8	Seafood	10	Ikura	8	31.00
11	4	Dairy Products	11	Queso Cabrales	4	21.00
12	4	Dairy Products	12	Queso Manchego La Pastora	4	38.00
13	8	Seafood	13	Konbu	8	6.00
14	7	Produce	14	Tofu	7	23.25

Figure 3. The first 14 rows returned using the SELECT statement in Figure 2. All columns in the Categories table and Products table are displayed together.

The JOIN clause is used to combine data from two or more tables. The resulting dataset can be queried to select desired columns and/or rows. Also, a variety of functions can be performed using the joined dataset, such as calculating maximum, minimum, and average values.

JOINs are important in SQL because the normalization process used when creating a database separates data into unique tables. While important for database functionality, it is often difficult to identify patterns, perform mathematical operations, or display data in an easily understandable way using normalized data from a single table. JOINs allow the data within unique tables to be connected and queried as a dataset in order to perform operations that would not be possible using individual tables.

Inner Joins, Outer Joins, and Cross Joins

Different types of JOINs can be used to connect tables in different ways. The default JOIN type in SQL Server Management Studio is an INNER JOIN (or JOIN). An INNER JOIN returns all matching rows from the tables referenced in the JOIN clause. If either table has a null value for a particular row, that row will not be included in the results. Figure 2 shows an INNER JOIN clause.

A FULL OUTER JOIN (or FULL JOIN) returns matching and non-matching rows from both tables referenced in the JOIN clause. In Figure 4, a FULL OUTER JOIN is used to connect the Territories and EmployeeTerritories tables from the Microsoft Northwind database. All rows from both tables are returned in the query results for a total of 53 rows.

```
Select * From Northwind.dbo.Territories as T
   Full Outer Join Northwind.dbo.EmployeeTerritories as ET
   On T.TerritoryID = ET.TerritoryID;
Go
```

	TerritoryID	TerritoryDescription	RegionID	EmployeeID	TerritoryID
1	01581	Westboro	1	2	01581
2	01730	Bedford	1	2	01730
3	01833	Georgetow	1	2	01833
4	02116	Boston	1	2	02116
5	02139	Cambridge	1	2	02139
6	02184	Braintree	1	2	02184
7	02903	Providence	1	5	02903
8	03049	Hollis	3	9	03049
9	03801	Portsmouth	3	9	03801
10	06897	Wilton	1	1	06897
11	07960	Morristown	1	5	07960
12	08837	Edison	1	5	08837
13	10019	New York	1	5	10019
14	10038	New York	1	5	10038
15	11747	Mellvile	1	5	11747
16	14450	Fairport	1	5	14450
17	19428	Philadelphia	3	8	19428
18	19713	Neward	1	1	19713
19	20852	Rockville	1	4	20852
20	27403	Greensboro	1	4	27403
21	27511	Cary	1	4	27511
22	29202	Columbia	4	NULL	NULL
23	30346	Atlanta	4	3	30346
24	31406	Savannah	4	3	31406
25	32859	Orlando	4	3	32859
26	33607	Tampa	4	3	33607
27	40222	Louisville	1	2	40222

Query executed successfully. uwc-studentsql.continuum.uw... ITFdn130 (64) pubs 00:00:00 53 rows

Figure 4. Example of a SELECT statement with a FULL OUTER JOIN clause between the Territories table and EmployeeTerritories table in the Microsoft Northwind database and a truncated version of the query result. Fifty-three rows were returned.

RIGHT OUTER JOINs (or RIGHT JOINs) and LEFT OUTER JOINs (or LEFT JOINs) can be used to return all the rows from one of the tables referenced in the JOIN clause and only the matching rows from the other table. Right and left are used to specify which table will return all rows. In a RIGHT JOIN, all rows from the table listed on the right side of the JOIN clause will be returned. In Figure 5, a RIGHT JOIN is

performed between the Territories and EmployeeTerritories tables from the Microsoft Northwind database. In this example, the EmployeeTerritories table is on the right side of the JOIN clause, so all rows in the EmployeeTerritories table are returned. Only the matching rows in the Territories table are returned. This query results in 49 rows.

```
Select * From Northwind.dbo.Territories as T
   Right Join Northwind.dbo.EmployeeTerritories as ET
   On T.TerritoryID = ET.TerritoryID;
Go
```

	TerritoryID	TerritoryDescription	RegionID	EmployeeID	TerritoryID
1	06897	Wilton	1	1	06897
2	19713	Neward	1	1	19713
3	01581	Westboro	1	2	01581
4	01730	Bedford	1	2	01730
5	01833	Georgetow	1	2	01833
6	02116	Boston	1	2	02116
7	02139	Cambridge	1	2	02139
8	02184	Braintree	1	2	02184
9	40222	Louisville	1	2	40222
10	30346	Atlanta	4	3	30346
11	31406	Savannah	4	3	31406
12	32859	Orlando	4	3	32859
13	33607	Tampa	4	3	33607
14	20852	Rockville	1	4	20852
15	27403	Greensboro	1	4	27403
16	27511	Cary	1	4	27511
17	02903	Providence	1	5	02903
18	07960	Morristown	1	5	07960
19	08837	Edison	1	5	08837
20	10019	New York	1	5	10019
21	10038	New York	1	5	10038
22	11747	Mellvile	1	5	11747
23	14450	Fairport	1	5	14450
24	85014	Phoenix	2	6	85014
25	85251	Scottsdale	2	6	85251
26	98004	Bellevue	2	6	98004
27	98052	Redmond	2	6	98052

Query executed successfully. uwc-studentsql.continuum.uw... ITFdn130 (64) pubs 00:00:00 49 rows

Figure 5. Example of a SELECT statement with a RIGHT OUTER JOIN clause between the Territories table and EmployeeTerritories table in the Microsoft Northwind database and a truncated version of the query result. Forty-nine rows were returned.

In a LEFT JOIN, all rows from the table listed on the left side of the JOIN clause will be returned. In Figure 6, a LEFT JOIN is performed between the Territories and EmployeeTerritories tables. The Territories table is on the left side of the JOIN clause, so all rows in the Territories table are returned and only the matching rows in the EmployeeTerritories table are returned. This query results in 53 rows, which is greater than the 49 rows returned using a RIGHT JOIN, indicating there are four territories without assigned employees.

```
Select * From Northwind.dbo.Territories as T
   Left Join Northwind.dbo.EmployeeTerritories as ET
   On T.TerritoryID = ET.TerritoryID;
Go
```

	TerritoryID	TerritoryDescription	RegionID	EmployeeID	TerritoryID
1	01581	Westboro	1	2	01581
2	01730	Bedford	1	2	01730
3	01833	Georgetow	1	2	01833
4	02116	Boston	1	2	02116
5	02139	Cambridge	1	2	02139
6	02184	Braintree	1	2	02184
7	02903	Providence	1	5	02903
8	03049	Hollis	3	9	03049
9	03801	Portsmouth	3	9	03801
10	06897	Wilton	1	1	06897
11	07960	Morristown	1	5	07960
12	08837	Edison	1	5	08837
13	10019	New York	1	5	10019
14	10038	New York	1	5	10038
15	11747	Mellvile	1	5	11747
16	14450	Fairport	1	5	14450
17	19428	Philadelphia	3	8	19428
18	19713	Neward	1	1	19713
19	20852	Rockville	1	4	20852
20	27403	Greensboro	1	4	27403
21	27511	Cary	1	4	27511
22	29202	Columbia	4	NULL	NULL
23	30346	Atlanta	4	3	30346
24	31406	Savannah	4	3	31406
25	32859	Orlando	4	3	32859
26	33607	Tampa	4	3	33607
27	40222	Louisville	1	2	40222

Query executed successfully. | uwc-studentsql.continuum.uw... | ITFdn130 (64) | pubs | 00:00:00 | 53 rows

Figure 6. Example of a SELECT statement with a LEFT OUTER JOIN clause between the Territories table and EmployeeTerritories table in the Microsoft Northwind database and a truncated version of the query result. Fifty-three rows were returned.

A CROSS JOIN returns all possible combinations of the rows within both tables in the JOIN clause. This means both matching and un-matching results will be returned. The syntax of a CROSS JOIN differs from that of INNER and OUTER JOINs because the connection between the two tables is not specified using the ON keyword. Since all possible combinations of the rows will be returned, the ON keyword is not needed. In Figure 7, a CROSS JOIN is performed between the Territories and EmployeeTerritories tables. This query returns 2,597 rows. A CROSS JOIN will always return a greater number of results than an INNER or OUTER JOIN performed on the same tables.

```
Select * From Northwind.dbo.Territories
Cross Join Northwind.dbo.EmployeeTerritories;
Go
```

	TerritoryID	TerritoryDescription	RegionID	EmployeeID	TerritoryID
1	01581	Westboro	1	1	06897
2	01730	Bedford	1	1	06897
3	01833	Georgetow	1	1	06897
4	02116	Boston	1	1	06897
5	02139	Cambridge	1	1	06897
6	02184	Braintree	1	1	06897
7	02903	Providence	1	1	06897
8	03049	Hollis	3	1	06897
9	03801	Portsmouth	3	1	06897
10	06897	Wilton	1	1	06897
11	07960	Morristown	1	1	06897
12	08837	Edison	1	1	06897
13	10019	New York	1	1	06897
14	10038	New York	1	1	06897
15	11747	Mellvile	1	1	06897
16	14450	Fairport	1	1	06897
17	19428	Philadelphia	3	1	06897
18	19713	Neward	1	1	06897
19	20852	Rockville	1	1	06897
20	27403	Greensboro	1	1	06897
21	27511	Cary	1	1	06897
22	29202	Columbia	4	1	06897
23	30346	Atlanta	4	1	06897
24	31406	Savannah	4	1	06897
25	32859	Orlando	4	1	06897
26	33607	Tampa	4	1	06897
27	40222	Louisville	1	1	06897

Query executed successfully. uwc-studentsql.continuum.uw... ITFdn130 (64) pubs 00:00:00 2,597 rows

Figure 7. Example of a SELECT statement with a CROSS JOIN clause between the Territories table and EmployeeTerritories table in the Microsoft Northwind database and a truncated version of the query result. Two thousand five hundred ninety-seven rows were returned.

Self-Joins

A self-join is used to join a table to itself. A self-join may be an INNER JOIN, OUTER JOIN, or CROSS JOIN. Self-joins are used to "perform comparisons or retrieve data from a single table that has some form of hierarchy, relationships, or dependencies." (Hightouch, https://hightouch.com/sql-dictionary/sql-self-join, 2025) (External Site) When writing a self-join in SQL, the table is assigned two different table aliases to distinguish between the two instances of the table being referenced. In Figure 8, a self-join is performed to join the Employees table from the Microsoft Northwind database to itself. The table aliases "E" and "M" are assigned to distinguish the Employee ID and Last Name returned when referring to managers from the Employee ID and Last Name returned when referring to employees. A self-join is necessary in this example because the Employees table contains employees who are managers and all managers within the table are also employees.

```
Select M.EmployeeID, M.LastName, E.EmployeeID, E.LastName
From Northwind.dbo.Employees as E
    Left Join Northwind.dbo.Employees as M
    On E.ReportsTo = M.EmployeeID;
Go
```

Figure 8. Example of a self-join using the Employees table from the Microsoft Northwind database. The table is assigned two table aliases (E and M) to reference different instances of the table in the JOIN clause.

Summary

JOINs are an important tool within SQL used to combine and query data from multiple tables at the same time. The type of JOIN used specifies which rows will be returned from the tables being joined. The default JOIN in SQL is the INNER JOIN, which returns all matching rows from the tables being joined. OUTER JOINs return matching and un-matching rows from the tables being joined. A FULL OUTER JOIN returns all rows from both tables. A RIGHT OUTER JOIN returns all rows from the table on the right side of the JOIN clause and only the matched rows from the table on the left side of the JOIN clause. A LEFT OUTER JOIN returns all rows from the table on the left side of the JOIN clause and only the matched rows from the table on the right side. A CROSS JOIN returns all possible combination of rows across the two tables. A self-join is used to connect a table to itself. A self-join may be an INNER JOIN, OUTER JOIN, or CROSS JOIN. Self-joins are often used to combine data that are hierarchical, related, or interdependent.