9. Joins Explained — Inner, Outer, Cross, Self

Joins are the magic glue that lets you combine tables into richer stories. They are where your relational thinking meets set-based execution.

9.1 INNER JOIN — the meeting of equals

An INNER JOIN returns rows where the join condition matches in both tables.

Example: Customers and Orders

```
SELECT c.CustomerID, c.Name, o.OrderID, o.OrderDate
FROM dbo.Customers c
INNER JOIN dbo.Orders o ON c.CustomerID = o.CustomerID;
```

- Only customers who placed orders appear.
- Rows without a match in either table are discarded.

Visual:

```
Customers ∩ Orders = matched rows only
```

9.2 LEFT OUTER JOIN — all from the left, some from the right

Returns all rows from the **left table**, and matching rows from the right. If no match, right-side columns are NULL.

Example:

```
SELECT c.CustomerID, c.Name, o.OrderID
FROM dbo.Customers c
LEFT JOIN dbo.Orders o ON c.CustomerID = o.CustomerID;
```

- All customers appear.
- Customers without orders have NULL for `OrderID`.

Visual:

```
Customers ─┐
           └── Orders (matches) + NULLs
```

9.3 RIGHT OUTER JOIN — symmetric to LEFT

All rows from the right table, matched left-side rows, else NULLs.

```sql
SELECT c.CustomerID, c.Name, o.OrderID
FROM dbo.Customers c
RIGHT JOIN dbo.Orders o ON c.CustomerID = o.CustomerID;
```

- Often less used; same effect can usually be achieved by swapping table positions in a LEFT JOIN.

9.4 FULL OUTER JOIN — preserve all

Returns **all rows from both tables**. Matching rows are combined; unmatched rows have NULLs for missing side.

```sql
SELECT c.CustomerID, c.Name, o.OrderID
FROM dbo.Customers c
FULL OUTER JOIN dbo.Orders o ON c.CustomerID = o.CustomerID;
```

- Rows without matches in either table appear.
- Useful when auditing or merging datasets where every row counts.

Visual:

```
Customers ∪ Orders = all matched + unmatched with NULLs
```

9.5 CROSS JOIN — Cartesian chaos

Returns every combination of rows from both tables.

```sql
SELECT c.Name, p.ProductName
FROM dbo.Customers c
CROSS JOIN dbo.Products p;
```

- Rows = #Customers × #Products
- Rarely used in production, mostly for generating test sets or combinatorial possibilities.

9.6 SELF JOIN — the table meets itself

A table can join itself. Often used for hierarchical or sequential data.

Example — Employee hierarchy

```
SELECT e.EmployeeID, e.Name AS EmployeeName, m.EmployeeID AS ManagerID, m.Name AS ManagerName
FROM dbo.Employees e
LEFT JOIN dbo.Employees m ON e.ManagerID = m.EmployeeID;
```

- `e` is the employee.
- `m` is the manager.
- LEFT JOIN ensures employees without managers still appear.

9.7 Practical Tips

- Always use table aliases for readability.
- Explicitly specify join conditions; implicit joins in the WHERE clause are legacy style.
- INNER JOIN filters rows; OUTER JOIN preserves unmatched rows.
- Be cautious with CROSS JOIN — row explosion is real.
- SELF JOINs are powerful for hierarchical queries but can be tricky with multiple levels.

Joins are the relational bridge. Understanding them fully makes aggregation, filtering, and reporting far more intuitive. Next chapter dives into anti-joins, where the question is "what doesn’t exist?".