10. Anti-Joins and NOT EXISTS vs NOT IN (MS SQL)

Anti-joins are the elegant SQL way of asking, "Which rows **don't have a match**?" Unlike normal joins that find connections, anti-joins spotlight absence.

10.1 Conceptual Understanding

- **Anti-join** = rows from table A that **do not** match rows in table B according to some condition.
- Common use cases:
  - Find customers who never placed an order.
  - Products that were never sold.
  - Employees without managers.

10.2 Using NOT EXISTS

`NOT EXISTS` checks for the absence of rows in a correlated subquery.

Example: Customers without orders

```
SELECT c.CustomerID, c.Name
FROM dbo.Customers c
WHERE NOT EXISTS (
    SELECT 1
    FROM dbo.Orders o
    WHERE o.CustomerID = c.CustomerID
);
```

- Correlated subquery references the outer table.
- SQL stops scanning as soon as it finds a match for each outer row (efficient).

Key Notes:

- Returns true if the subquery yields no rows.
- Safe with NULLs in the inner table.

10.3 Using NOT IN

`NOT IN` checks a value against a set of values.

Example:

```
SELECT c.CustomerID, c.Name
FROM dbo.Customers c
WHERE c.CustomerID NOT IN (
    SELECT CustomerID FROM dbo.Orders
);
```

Important caveat:

- If `CustomerID` in `Orders` contains NULL, the entire `NOT IN` comparison can **return zero rows**.
- Always filter out NULLs in the inner query:

```
WHERE CustomerID NOT IN (
  SELECT CustomerID FROM dbo.Orders WHERE CustomerID IS NOT NULL
);
```

10.4 Comparison Table

Feature	NOT EXISTS	NOT IN
Handles NULLs safely	☑	❌ (careful!)
Optimizer-friendly for correlated subqueries	☑	Sometimes
Typical use	Anti-join style	Anti-join with set

10.5 LEFT JOIN / IS NULL as Anti-Join

Alternative anti-join using LEFT JOIN

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

- Left table preserved.
- Where no match exists, right-side columns are NULL.
- Works reliably even with NULLs in the right table.

10.6 Performance Tips

- Prefer `NOT EXISTS` for correlated anti-joins — SQL Server handles this efficiently.
- Be cautious with `NOT IN` when the inner table may contain NULLs.
- LEFT JOIN / IS NULL is often readable and intuitive; for large datasets, test performance.
- Indexes on the join column (CustomerID, ProductID) drastically improve anti-join speed.

Anti-joins answer the “who’s missing?” questions in your data world. They are subtle, but mastering them prevents logical gaps in your analysis.

Next chapter: **11. Subqueries — scalar, correlated, and lateral (APPLY)**.