bite-sized.sql

SQL: INTRO TO LATERAL JOINS

Let’s find the three most recent orders per customer

One solution is using a CTE with the ROW_NUMBER window function

bite-sized.sql

E.g. CTE / ROW_NUMBER()

The ROW_NUMBER() window function in the RecentOrders CTE calculates the row number within each customer, sorted by order date.

```sql
WITH RecentOrders AS
  (
      SELECT
        O.CustomerID,
        O.OrderDate,
        ROW_NUMBER() OVER (PARTITION BY O.CustomerID
                           ORDER BY O.OrderDate DESC) AS RowNum
      FROM Orders AS O
  )
SELECT C.CustomerName, R.OrderDate
FROM Customers AS C
    INNER JOIN RecentOrders AS R
      ON C.CustomerID = R.CustomerID
WHERE R.RowNum <= 3;
```

We join the Customers table to the RecentOrders CTE and apply a WHERE condition such that the row number is less than or equal to 3

We can also use a lateral join for this query

A lateral join lets us use a query's output as input to another query in the same SELECT

bite-sized.sql

E.g. CROSS JOIN LATERAL

The comma here means (and can be replaced with) CROSS JOIN

```
SELECT C.CustomerName, O.OrderDate
FROM Customers AS C,
  LATERAL (SELECT *
           FROM Orders AS O
           WHERE O.CustomerID = C.CustomerID
           ORDER BY OrderDate DESC LIMIT 3) AS O;
```

Similar to a correlated sub-query, we can reference columns from the preceding table in the join

Because we're using a lateral join, we can specify an ORDER BY and LIMIT clause in the sub-query.

SQL Server uses CROSS APPLY instead of CROSS JOIN LATERAL

```
SELECT C.CustomerID, O.OrderDate
FROM Customers AS C
CROSS APPLY (SELECT TOP 3 *
             FROM Orders AS O
             WHERE O.CustomerID = C.CustomerID
             ORDER BY OrderDate DESC) AS O;
```

In additional to CROSS JOIN, lateral joins can also be used with INNER, LEFT and RIGHT joins

bite-sized.sql

E.g. LEFT JOIN LATERAL

The left join ensures that all customers are returned, even those with no orders

```
SELECT C.CustomerName, O.OrderDate
FROM Customers AS C
LEFT JOIN LATERAL
    (SELECT *
    FROM Orders AS O
    WHERE O.CustomerID = C.CustomerID
    ORDER BY OrderDate DESC LIMIT 3) AS O ON TRUE;
```

ON TRUE is necessary because LEFT JOIN requires an ON predicate

There are many other uses for lateral joins. This is just one small example

As with any SQL feature, performance is dependent on many factors

Always ensure your approach is appropriate for your data environment

LATERAL joins:

1. Reference columns from preceding tables

2. Can return multiple rows per join key

3. Can be used with CROSS, INNER, LEFT and RIGHT joins