Advanced SQL Tools: Lateral Joins

Advanced SQL and Cloud

Objectives

1. Explain what a lateral join is and identify situations where it is useful compared to subqueries
2. Use LATERAL joins to perform row-wise operations such as top-N per group
3. Combine string functions and lateral joins to solve practical analysis problems

Lateral Joins

A LATERAL join allows a subquery in the FROM clause to reference columns from previous tables in the FROM.

Think: correlated subquery in the FROM clause.

One major difference from correlated subqueries is that lateral joins can return multiple columns and multiple rows.

Lateral Joins

Example: Say we want to find the first order per customer. We can do this with a lateral join, which allows us to reference the first table inside the second.

```
SELECT c.customer_id, o.order_date, o.amount
FROM customers c
JOIN LATERAL (
    SELECT order_date, amount
    FROM orders o
    WHERE o.customer_id = c.customer_id
    ORDER BY order_date
    LIMIT 1
) o ON TRUE;
```

(Note that the “ON TRUE” condition means all rows from the customers table are included in the results)

Lateral Joins

Common use cases:

- First/last event per entity
- Splitting strings into multiple rows, for example, after applying unnest to string_to_array or other json expansion
- Row-wise lookups
    - Eg. give the top-n related rows for each row in an outer query

Implicit vs. Explicit LATERAL

We have seen how we can split a json array into multiple rows using **JSON_ARRAY_ELEMENTS**.

```
SELECT
  asteroid_id, name,
  jsonb_array_elements(data->'close_approach_data') AS approach
FROM neows_lookup;
```

asteroid_id text	name text	approach jsonb
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "122.5074468577", "miles": "29
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "183.4078760325", "miles": "43
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "194.211053134", "miles": "464
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "139.9569441821", "miles": "33
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "67.7144537162", "miles": "161

Implicit vs. Explicit LATERAL

But what if we want to filter based on values in the expanded column? For example, what if we wanted to filter to only close approach values for Earth?

```
SELECT
  asteroid_id,
  name,
  JSONB_ARRAY_ELEMENTS(data->'close_approach_data') AS approach
FROM neows_lookup
WHERE JSONB_ARRAY_ELEMENTS(data->'close_approach_data')->>'orbiting_body' = 'Earth';
```

```
ERROR:  set-returning functions are not allowed in WHERE
LINE 6: WHERE jsonb_array_elements(data->'close_approach_data')->>'o...
              ^
```

Implicit vs. Explicit LATERAL

Instead, we can do an implicit lateral join. This happens when you use a comma join (a join where you just separate the tables with a comma) if there is a reference to the left table in the right one.

```
SELECT
  asteroid_id,
  name,
  approach
FROM neows_lookup,
  JSONB_ARRAY_ELEMENTS(data->'close_approach_data') AS approach
WHERE approach->>'orbiting_body' = 'Earth';
```

asteroid_id text	name text	approach jsonb
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "122.5074468577", "miles
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "183.4078760325", "miles
2000433	433 Eros (A898 PA)	{"miss_distance": {"lunar": "194.211053134", "miles":