**Lab Exercise 14– Semi-Structure (JSON) Data in Snowflake**

**Objective:**

In Snowflake, **semi-structured data** like JSON is stored in VARIANT columns. For your scenario, you have a table called customer\_data with VARIANT column customer\_info, which stores JSON data.

**Recursive CTEs for JSON**

You can use recursive CTEs to iterate over arrays inside the JSON data, such as the orders array in your example. Recursive CTEs are useful for breaking down complex or hierarchical JSON structures.

Here’s how to approach this with your existing customer\_data table:

**Step 1: Create the Table with VARIANT Column**

You already have this step in place:

CREATE OR REPLACE TABLE raw\_db.raw\_data.customer\_data (

customer\_info VARIANT

);

**Step 2: Insert Data Using PARSE\_JSON**

You also already have this data insertion step:

INSERT INTO raw\_db.raw\_data.customer\_data(customer\_info)

SELECT PARSE\_JSON('{"name": "John Doe", "email": "john.doe@example.com", "orders": [{"order\_id": 101, "amount": 250}, {"order\_id": 102, "amount": 150}]}')

UNION ALL

SELECT PARSE\_JSON('{"name": "Jane Smith", "email": "jane.smith@example.com", "orders": [{"order\_id": 103, "amount": 300}, {"order\_id": 104, "amount": 200}]}');

**Approach for Extracting Order Details**

Instead of using a recursive CTE, we can utilize **Snowflake's FLATTEN function** to extract and unroll the JSON array (orders) into individual rows.

**Correct Query Using FLATTEN**

-- Unroll the orders array into individual rows

SELECT

customer\_info:name AS customer\_name,



customer\_info:email AS customer\_email,

value:order\_id AS order\_id,

value:amount AS order\_amount

FROM raw\_db.raw\_data.customer\_data,

LATERAL FLATTEN(input => customer\_info:orders);

**Step 2: model ( demo\_json.sql)**

SELECT

    customer\_info:name AS customer\_name,

    customer\_info:email AS customer\_email,

    value:order\_id AS order\_id,

    value:amount AS order\_amount

FROM raw\_db.raw\_data.customer\_data,

LATERAL FLATTEN(input => customer\_info:orders)

**Run this model:**

dbt run -m demo\_json

**Explanation of the Query:**

1. **FLATTEN Function**:
   * This function is used to unroll an array (or nested structure) into individual rows.
   * input => customer\_info:orders specifies the JSON array (orders) to be unrolled.
2. **Dot Notation (value:order\_id, value:amount)**:
   * The FLATTEN function returns the array elements as VALUE. You can then use dot notation to extract individual fields from the JSON objects within the array.
3. **LATERAL Join**:
   * The LATERAL keyword allows FLATTEN to reference the customer\_info column for each row in customer\_data.

**Expected Output:**

The output will list all customer orders as individual rows:

| **customer\_name** | **customer\_email** | **order\_id** | **order\_amount** |
| --- | --- | --- | --- |
| John Doe | john.doe@example.com | 101 | 250 |
| John Doe | john.doe@example.com | 102 | 150 |
| Jane Smith | jane.smith@example.com | 103 | 300 |
| Jane Smith | jane.smith@example.com | 104 | 200 |