**Understanding RESULT\_SCAN Behavior with Duplicate Column Aliases in Snowflake**

In Snowflake, when you execute a query like:

sql

SELECT 1 AS a, 2 AS a\_1, 3 AS a;

The result set displays duplicate column aliases:

| A | A\_1 | A |

|---+-----+---|

| 1 | 2 | 3 |

This is allowed in direct query output, even though the alias a is used twice.

However, when you use RESULT\_SCAN(LAST\_QUERY\_ID()) to retrieve the result of that query:

sql

SELECT \* FROM TABLE(RESULT\_SCAN(LAST\_QUERY\_ID()));

Snowflake automatically **renames duplicate column aliases** to ensure uniqueness in the result set:

Code

| A | A\_1 | A\_2 |

|---+-----+-----|

| 1 | 2 | 3 |

This renaming is necessary because RESULT\_SCAN() treats the result as a **virtual table**, and table columns must have **unique names** to avoid ambiguity in downstream operations.

-- Step 1: Run the target query

SELECT \* FROM MYSNOW.MY\_TRANSIENT\_SCHEMA.sample\_table;

-- Step 2: Capture the query ID right after

SET my\_query\_id = LAST\_QUERY\_ID();

-- Step 3: Use RESULT\_SCAN with the captured ID

SELECT \* FROM TABLE(RESULT\_SCAN($my\_query\_id));

CREATE OR REPLACE PROCEDURE audit\_high\_value\_sales()

RETURNS STRING

LANGUAGE SQL

AS

$$

DECLARE

query\_id STRING;

BEGIN

-- Step 1: Filter high-value sales

SELECT \* FROM sales.transactions

WHERE amount > 100000;

-- Step 2: Capture the query ID

LET query\_id = LAST\_QUERY\_ID();

-- Step 3: Insert the result into audit table using RESULT\_SCAN

INSERT INTO audit.high\_value\_sales\_log

SELECT \* FROM TABLE(RESULT\_SCAN(query\_id));

-- Step 4: Return confirmation

RETURN 'High-value sales successfully audited.';

END;

$$;

In our audit pipeline, we use RESULT\_SCAN() to capture and reuse filtered data sets without re-execution. This ensures performance efficiency and guarantees consistency between analysis and logging steps.”