□ SCD Type 2 Logic Explanation (Line by Line)

This document explains the logic of a **Slowly Changing Dimension Type 2 (SCD2)** implementation using SQL `MERGE` and `INSERT` statements. It is broken into 4 clear steps.

□ Step 1: Create the SCD2 Table

```
CREATE TABLE scd_cat.scd_schema.scd2_table (
  id INT,
  name STRING,
  value STRING,
  valid_from TIMESTAMP,
  valid_to TIMESTAMP,
  is_current BOOLEAN
);
```

□ **Explanation:**
We create a versioned history table that will store all changes over time.
Each row has:

- `valid_from` : when the record became valid
- `valid_to` : when it stopped being valid (future date means it's current)
- `is_current` : TRUE if this is the current record

□ Step 2: Initial Insert from Source Table

```
 INSERT INTO scd_cat.scd_schema.scd2_table
SELECT
  id,
  name,
  value,
  updated_at AS valid_from,
  TIMESTAMP '9999-12-31 23:59:59' AS valid_to,
  TRUE AS is_current
FROM scd_cat.scd_schema.source_table;
```

□ **Explanation:**
We load the first set of data into the SCD2 table from the source, marking all as current with a far-future `valid_to`.

□ Step 3: MERGE – Expire Old Records (if Changed)

```
 MERGE INTO scd_cat.scd_schema.scd2_table AS target
USING scd_cat.scd_schema.source_table AS source
ON target.id = source.id AND target.is_current = TRUE
WHEN MATCHED AND (
    target.name <> source.name OR target.value <> source.value
)
THEN UPDATE SET
  target.valid_to = source.updated_at,
  target.is_current = FALSE;
```

□ **Explanation:**
We match current records in the target table with incoming source records.
If any changes are found (like `name` or `value`), we expire the old record:

- Set `valid_to` to the update time
- Mark `is_current = FALSE`

✅ Step 4: INSERT – Add New Current Version of Records

```
 INSERT INTO scd_cat.scd_schema.scd2_table
SELECT
  s.id,
  s.name,
  s.value,
  s.updated_at AS valid_from,
  TIMESTAMP '9999-12-31 23:59:59' AS valid_to,
  TRUE AS is_current
FROM scd_cat.scd_schema.source_table s
LEFT JOIN scd_cat.scd_schema.scd2_table t
  ON s.id = t.id AND t.is_current = FALSE AND t.valid_to = s.updated_at
WHERE t.id IS NOT NULL;
```

✅ **Explanation:**
After expiring the old version, we insert the new version:

- New values from source
- `valid_from` = updated_at
- `valid_to` = far future (means still valid)
- `is_current = TRUE`

We join only those rows that just expired (`valid_to = updated_at`) to insert their updated version.

✅ Summary

This full SCD2 logic allows you to:

- Keep track of data changes over time
- Retain full history by marking old records as expired
- Always know the current active record (`is_current = TRUE`)
- Use timestamps (`valid_from` / `valid_to`) to query data as of any point in time

Perfect for **auditing**, **time travel**, **analytics**, and **historical reporting** in a data warehouse or lakehouse.