Basic MERGE Example

In this example, a merge operation involves two tables:

- `visits_daily` logs daily restaurant traffic, and is updated with each customer visit. Data in this table is refreshed every 24 hours.
- `visits_history` stores the history of customer visits to various restaurants, accumulated over an indefinite time span.

Each night, you merge the daily visit count from `visits_daily` into `visits_history`. The merge operation modifies the target table in two ways:

- Updates existing customer data.
- Inserts new rows of data for first-time customers.

One MERGE statement executes both operations as a single (upsert) transaction.

Source and Target Tables

The source and target tables `visits_daily` and `visits_history` are defined as follows:

```
CREATE TABLE public.visits_daily
(
    customer_id int,
    location_name varchar(20),
    visit_time time(0) DEFAULT (now())::timetz(6)
);

CREATE TABLE public.visits_history
(
    customer_id int,
    location_name varchar(20),
    visit_count int
);
```

Table `visits_history` contains rows of three customers who between them visited two restaurants, Etoile and LaRosa:

```
=> SELECT * FROM visits_history ORDER BY customer_id, location_name;
 customer_id | location_name | visit_count
-------------+---------------+-------------
        1001 | Etoile        |           2
        1002 | La Rosa       |           4
        1004 | Etoile        |           1
(3 rows)
```

By close of business, table `visits_daily` contains three rows of restaurant visits:

```
=> SELECT * FROM visits_daily ORDER BY customer_id, location_name;
 customer_id | location_name | visit_time
-------------+---------------+------------
        1001 | Etoile        | 18:19:29
        1003 | Lux Cafe      | 08:07:00
        1004 | La Rosa       | 11:49:20
(3 rows)
```

Table Data Merge

The following MERGE statement merges `visits_daily` data into `visits_history`:

- For matching customers, MERGE updates the occurrence count.
- For non-matching customers, MERGE inserts new rows.

```
=> MERGE INTO visits_history h USING visits_daily d
    ON (h.customer_id=d.customer_id AND h.location_name=d.location_name)
    WHEN MATCHED THEN UPDATE SET visit_count = h.visit_count  + 1
    WHEN NOT MATCHED THEN INSERT (customer_id, location_name, visit_count)
    VALUES (d.customer_id, d.location_name, 1);
 OUTPUT
--------
      3
(1 row)
```

MERGE returns the number of rows updated and inserted. In this case, the returned value specifies three updates and inserts:

- Customer `1001`'s third visit to Etoile
- New customer `1003`'s first visit to new restaurant Lux Cafe
- Customer `1004`'s first visit to La Rosa

If you now query table `visits_history`, the result set shows the merged (updated and inserted) data. Updated and new rows are highlighted:

```
=> SELECT * FROM visits_history ORDER BY customer_id, location_name
 customer_id | location_name | visit_count
-------------+---------------+-------------
        1001 | Etoile        |           3
        1002 | La Rosa       |           4
        1003 | Lux Cafe      |           1
        1004 | Etoile        |           1
        1004 | La Rosa       |           1
(5 rows)
```