**TASK 1**

**Data Loading Procedures for Dimensions**

This part outlines the procedures for loading data into various dimensions within the **BL\_DM** schema. Each dimension has a corresponding stored procedure that handles the insertion and updating of data, ensuring data integrity and consistency.

**1. Customers Dimension**

**Procedure: bl\_cl.load\_dim\_customers()**

**Description:** This procedure loads customer data into **BL\_DM.DIM\_CUSTOMERS**. It inserts new records and updates existing ones based on **Customer\_SRC\_ID** and **SOURCE\_System**. A surrogate key is generated using **NEXTVAL('BL\_DM.seq\_dim\_customers')**.

**Key Steps:**

* Log procedure execution.
* Retrieve the latest records from **BL\_3NF.CE\_CUSTOMERS**.
* Perform an **UPSERT** operation.
* Log success or failure messages.

**Execution:**

CALL bl\_cl.load\_dim\_customers();

**2. Sales Channels Dimension**

**Procedure: bl\_cl.load\_dim\_sales\_channels()**

**Description:** This procedure loads sales channel data into **BL\_DM.DIM\_SALES\_CHANNELS**. It retrieves the latest records and inserts or updates them accordingly.

**Key Steps:**

* Retrieve distinct latest records from **BL\_3NF.CE\_SALES\_CHANNELS**.
* Construct dynamic SQL for **UPSERT** operation.
* **Execute** the **dynamic SQL**.
* Usage of cursor For loop.
* **Cursor Variables** – Stores column values from the cursor.
* Log success messages.

**Execution:**

CALL bl\_cl.load\_dim\_sales\_channels();

**3. Employees Dimension (Slowly Changing Dimension - SCD)**

**Procedure: bl\_cl.load\_dim\_employees\_scd()**

**Description:** This procedure loads employee data into **BL\_DM.DIM\_EMPLOYEES\_SCD**, handling historical changes for employees by marking previous records as inactive.

**Key Steps:**

* Insert new employee records.
* Update existing records.
* Handle **SCD Type 2** changes by marking previous records as inactive when new records arrive.
* Log success messages.

**Execution:**

CALL bl\_cl.load\_dim\_employees\_scd();

**4. Locations Dimension**

**Procedure: bl\_cl.load\_dim\_locations()**

**Description:** This procedure loads location data into **BL\_DM.DIM\_LOCATIONS**. It ensures data consistency by updating existing records when needed.

**Key Steps:**

* Join **CE\_LOCATIONS** with **CE\_CITIES** and **CE\_STATES**.
* Perform an **UPSERT** operation.
* Log success messages.

**Execution:**

CALL bl\_cl.load\_dim\_locations();

**5. Vehicles Dimension**

**Procedure: bl\_cl.load\_dim\_vehicles()**

**Description:** This procedure loads vehicle data into **BL\_DM.DIM\_VEHICLES**, maintaining consistency across vehicle models and manufacturers.

**Key Steps:**

* Join **CE\_VEHICLES** with **CE\_MODELS** and **CE\_PRODUCERS**.
* Perform an **UPSERT** operation.
* Log success messages.

**Execution:**

CALL bl\_cl.load\_dim\_vehicles();

**6. Fact Transactions**

**Procedure: bl\_cl.load\_fact\_transactions()**

**Description:** This procedure loads transactional data into **BL\_DM.FACT\_TRANSACTIONS**, linking all necessary dimensions.

**Key Steps:**

* Join transactional data with relevant dimension tables.
* Perform an **UPSERT** operation.
* Log success or failure messages.

**Execution:**

CALL bl\_cl.load\_fact\_transactions();

**6. Date Dimension**

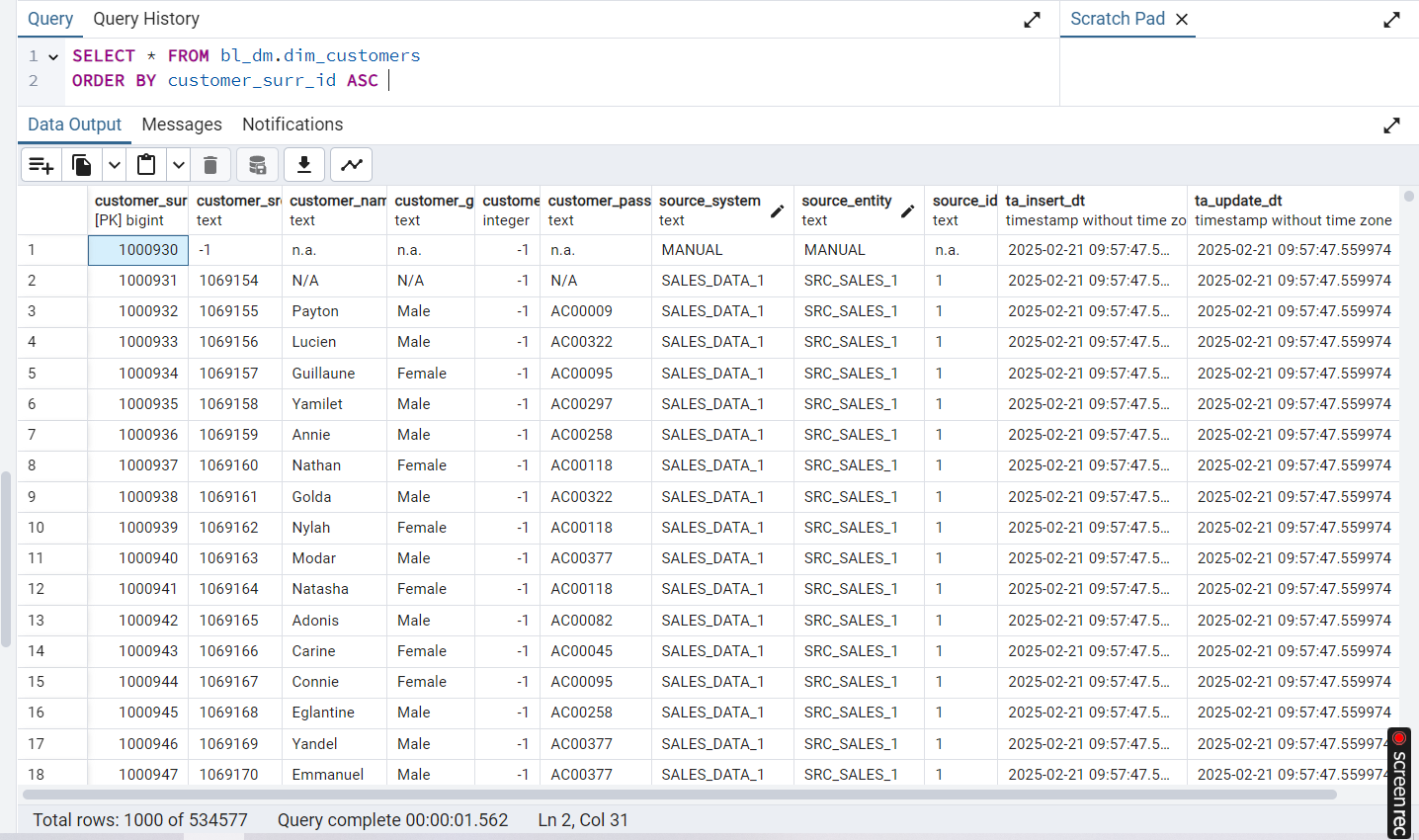
This function bl\_cl.insert\_dates() populates the BL\_DM.DIM\_DATES table with a continuous range of dates from January 1, 2010 to December 31, 2030. It ensures that the table contains all necessary date-related attributes to support time-based analysis.

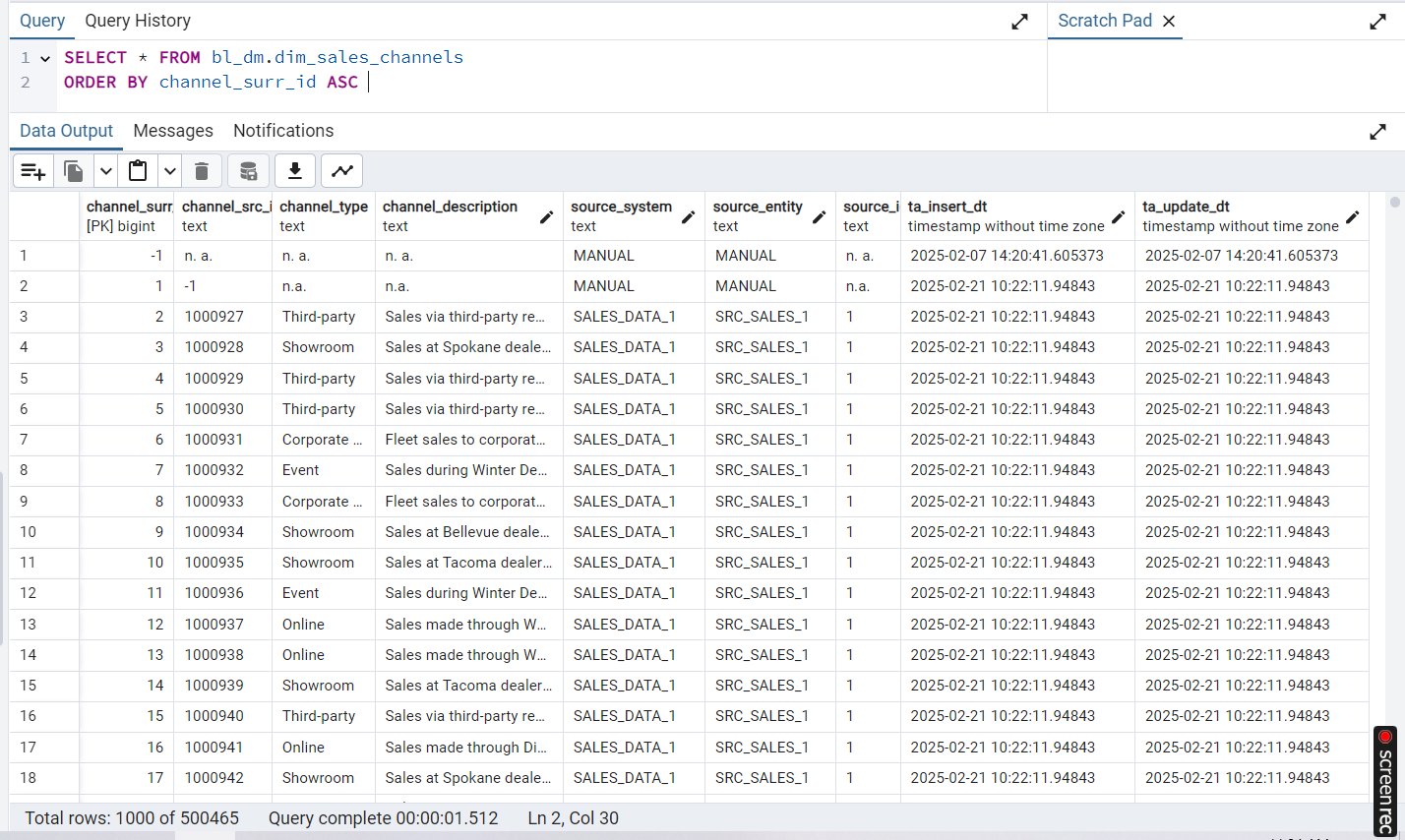
**Key Features:**

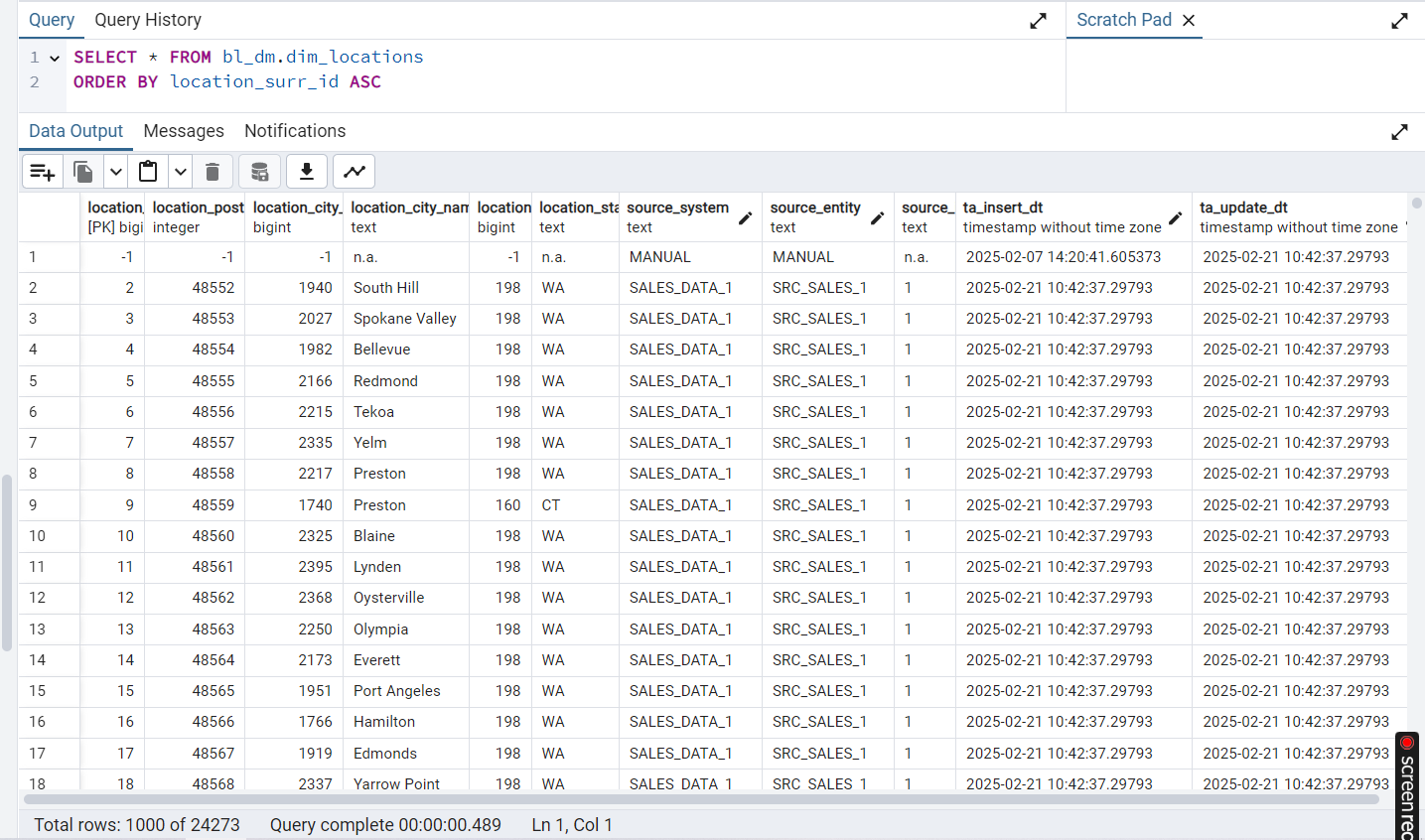
* **Date Range Generation:** Uses GENERATE\_SERIES() to create a sequential list of dates from 2010 to 2030.
* **Extracting Date Components:** The procedure extracts key components such as year, quarter, month, week, day, and weekday name from each generated date.

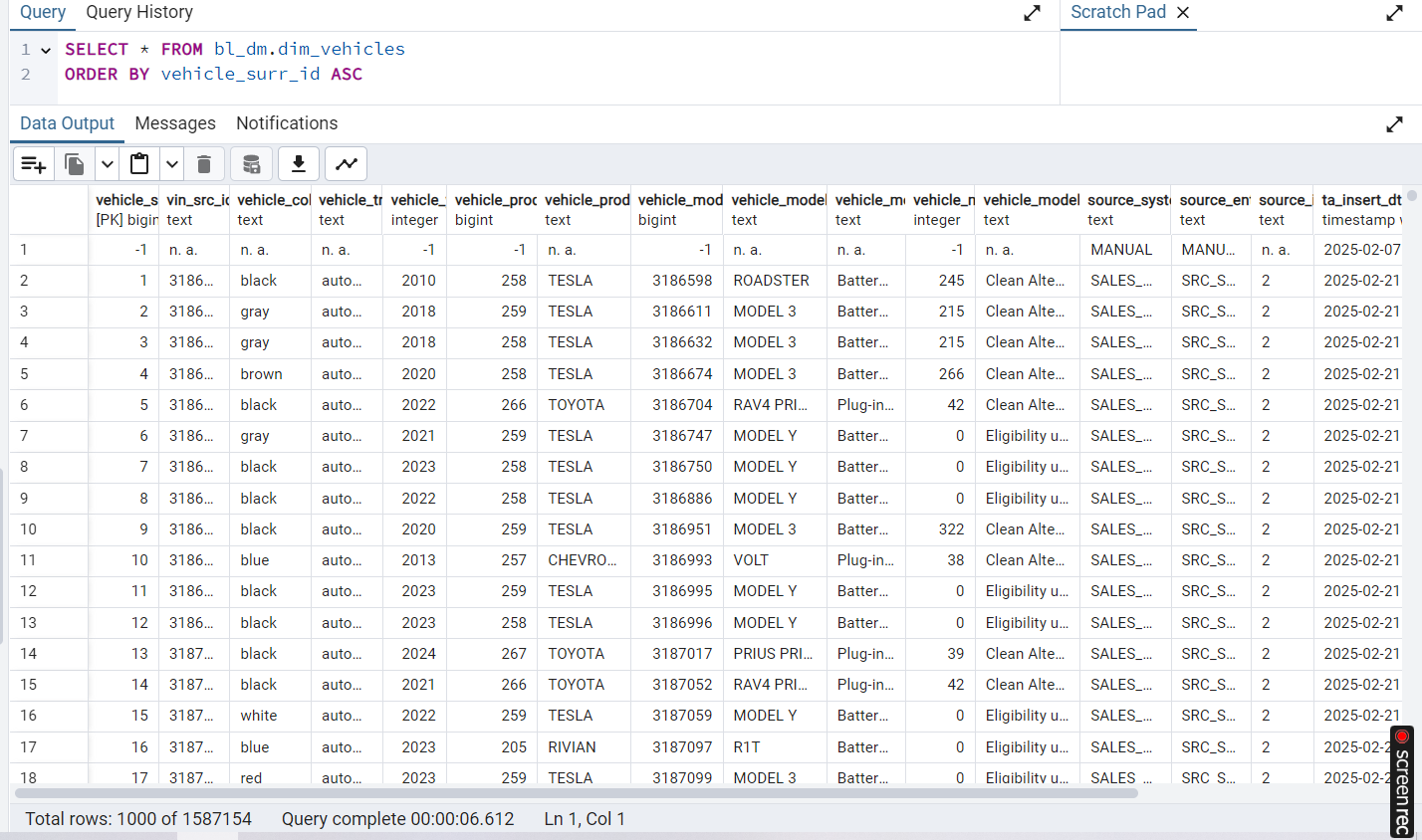
**Execution:** SELECT bl\_cl.insert\_dates();

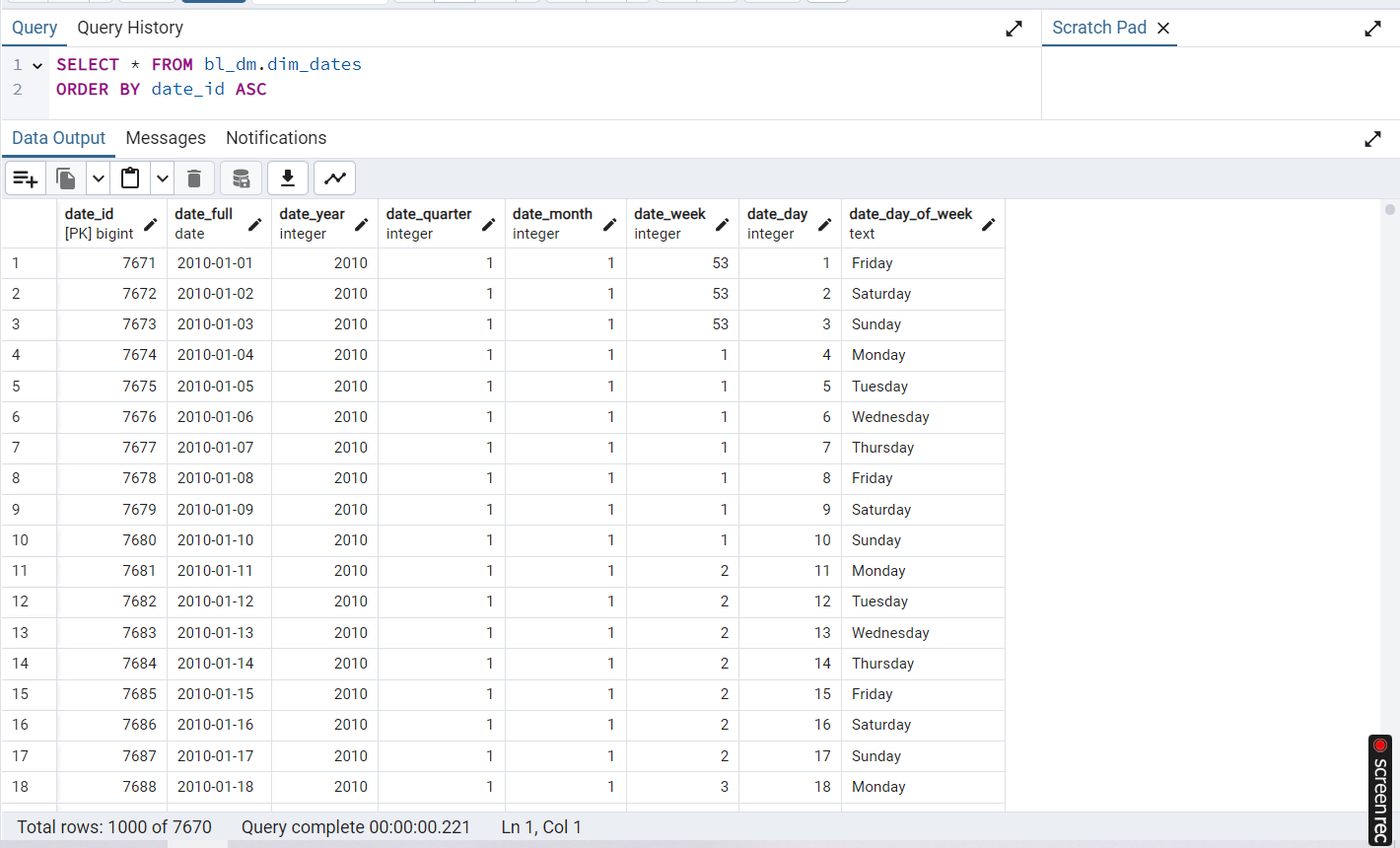
Add screenshots showcasing the data inserted into the DM layer.

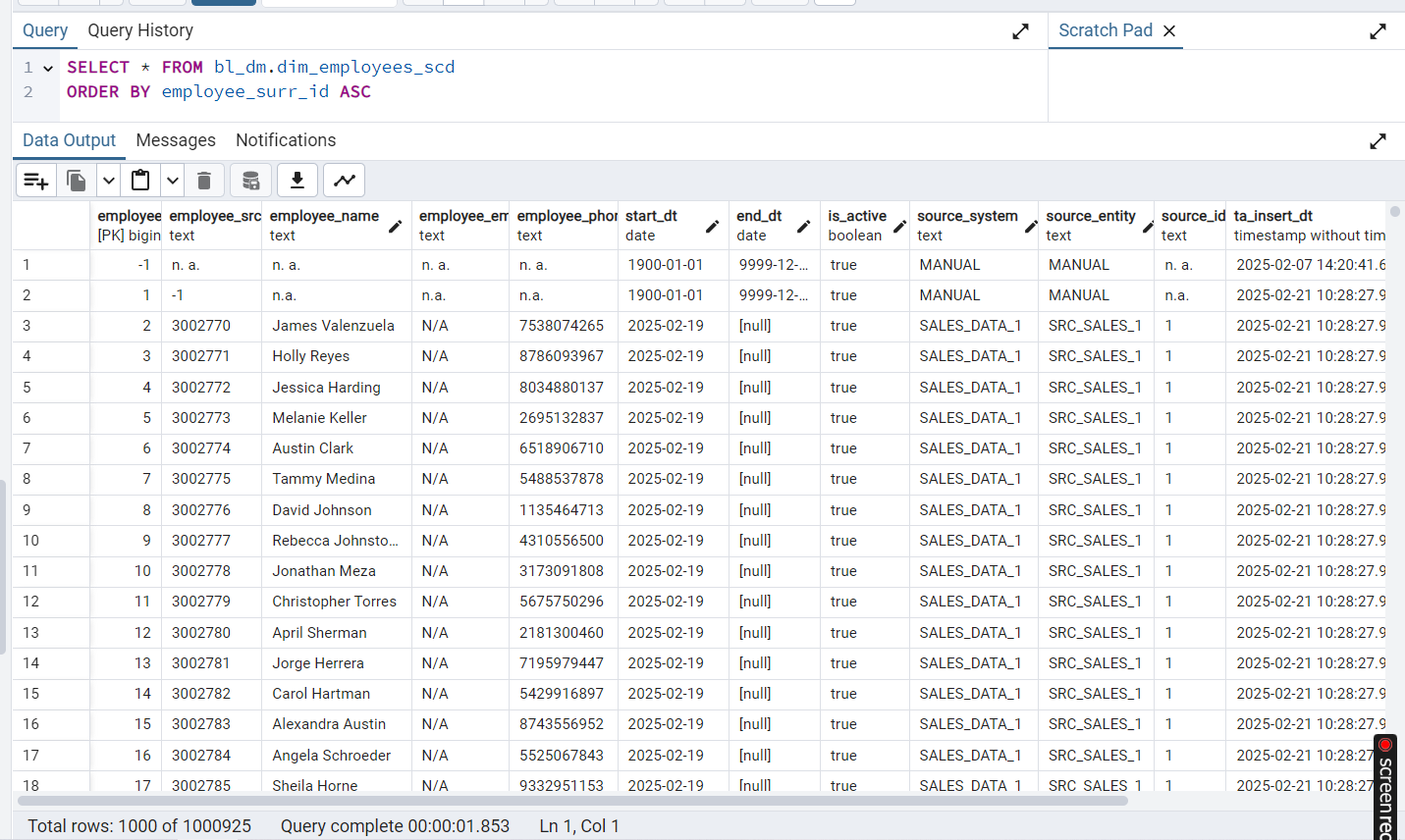


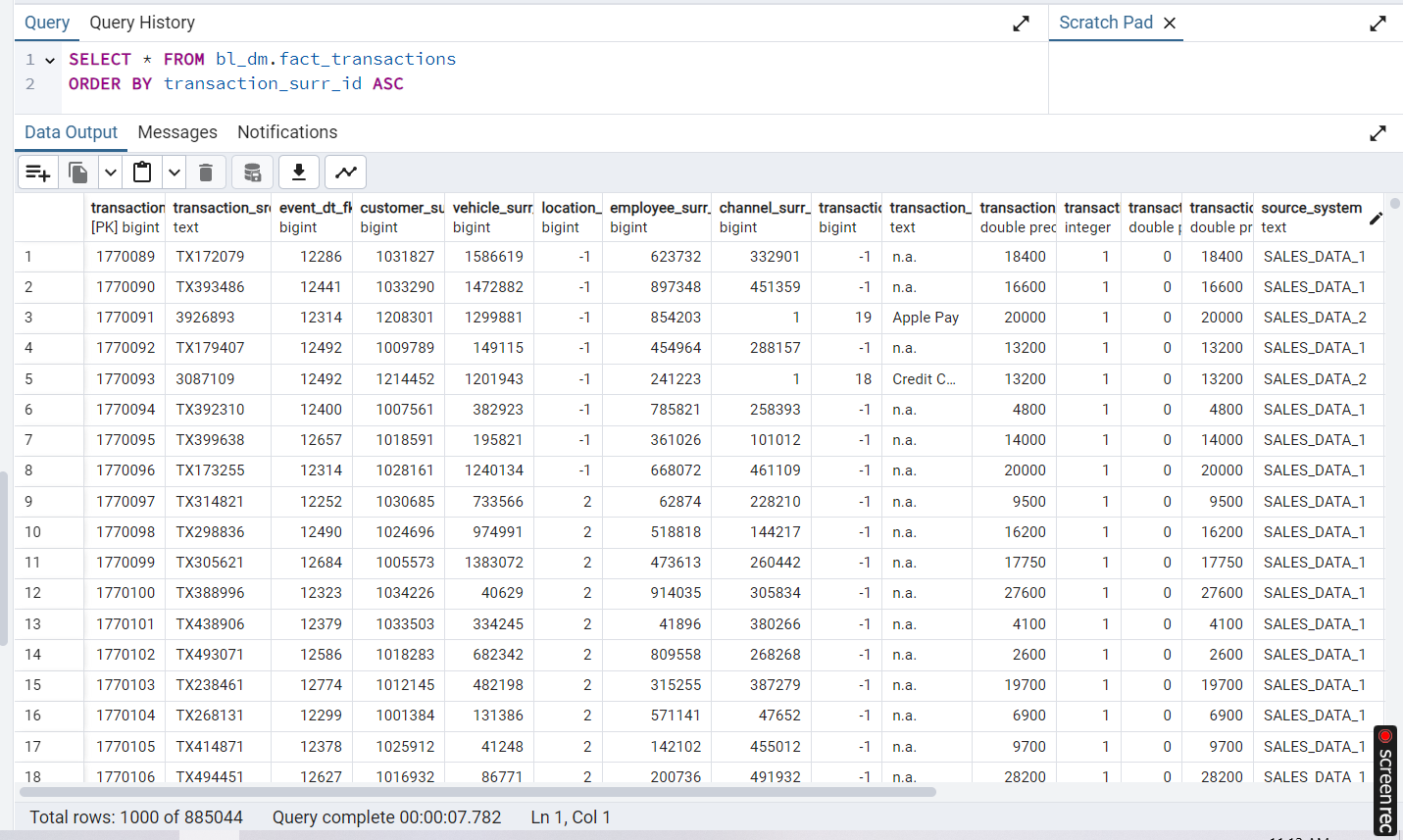




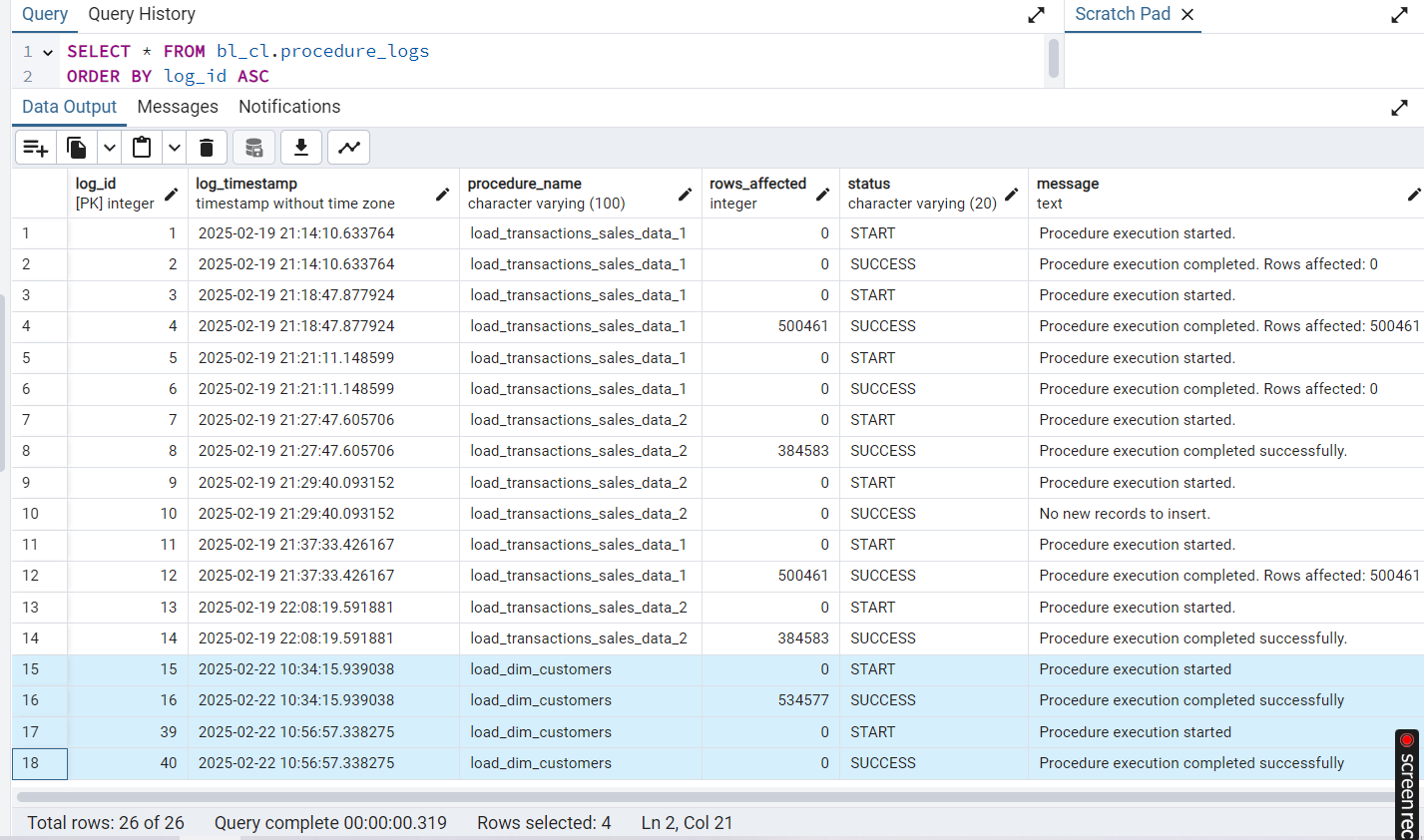






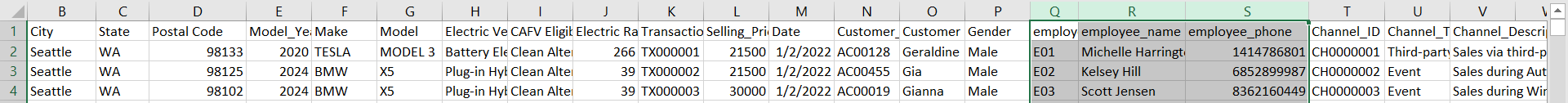


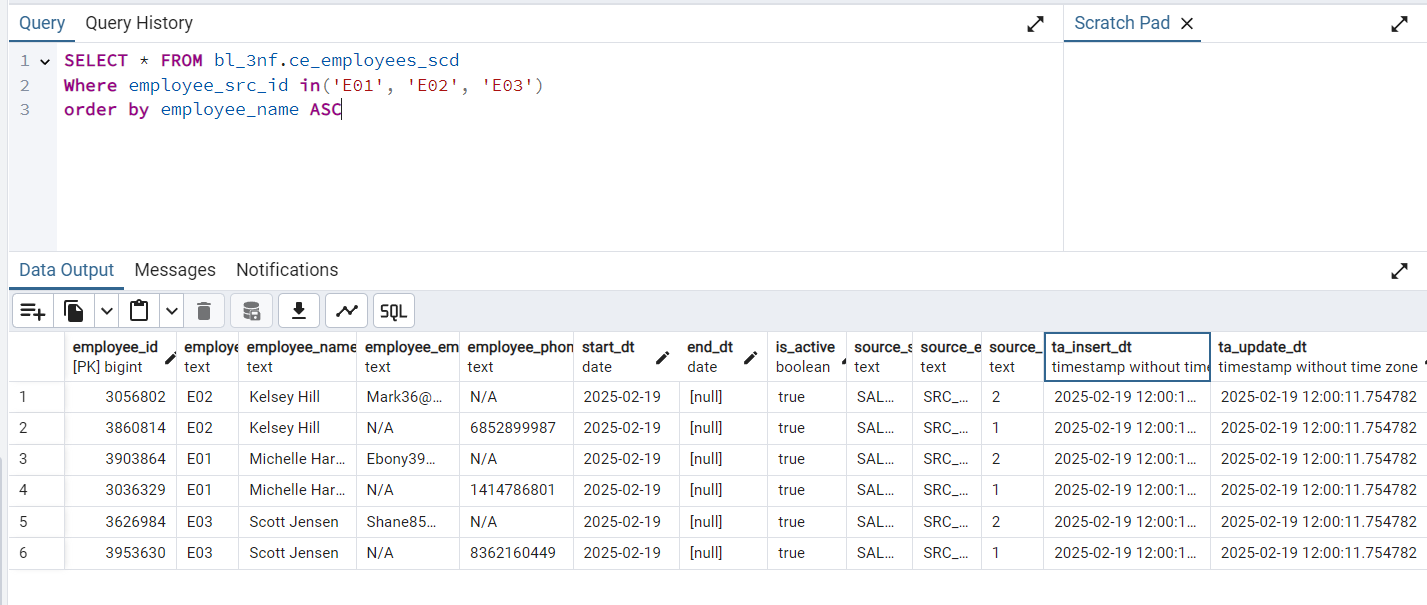
**TASK 2**

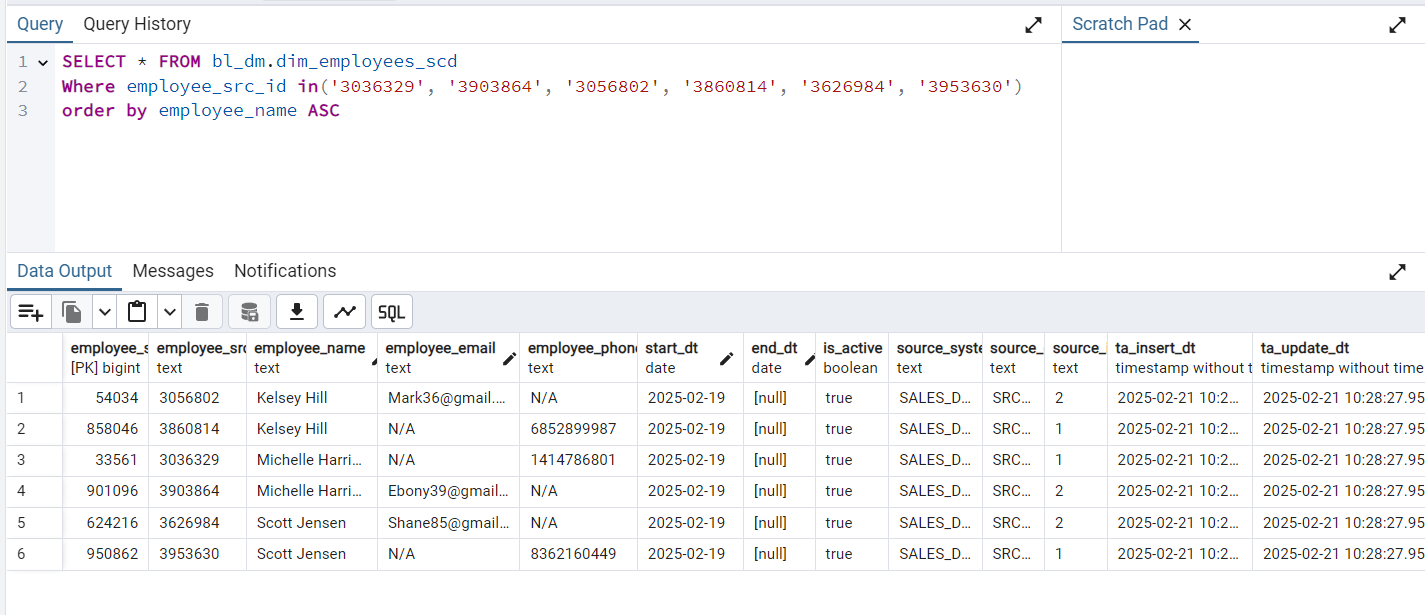


**TASK 3**

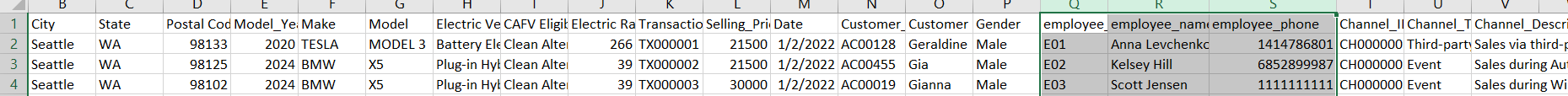
I made changes in sales\_data\_1, this is what I had before:



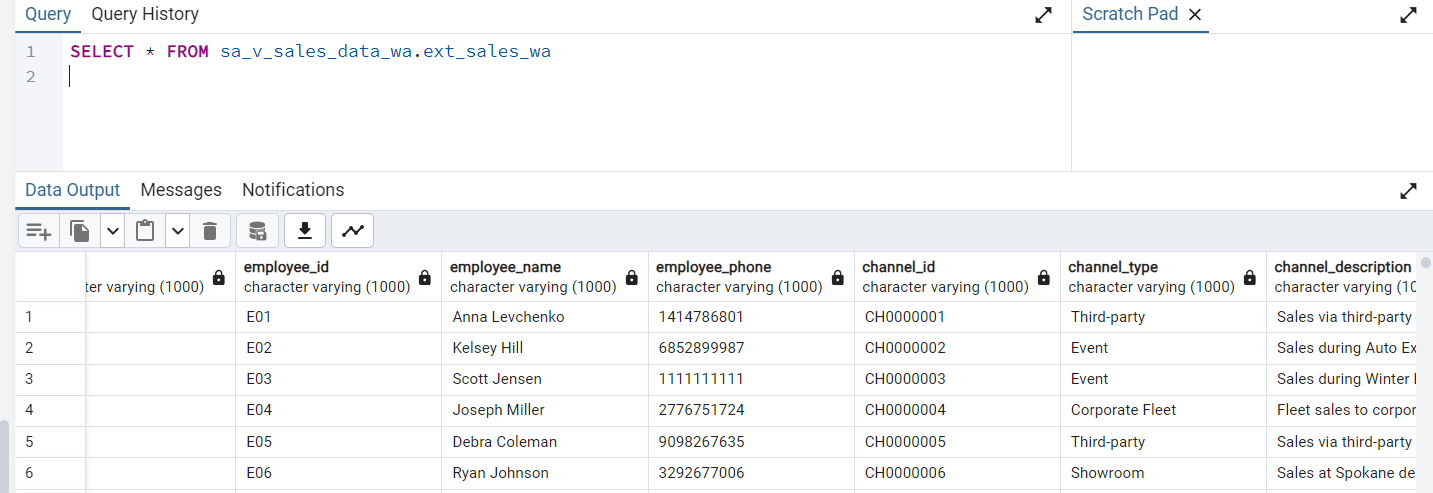




This after I made changes: for E01 set new name, and for E03 changed phone.

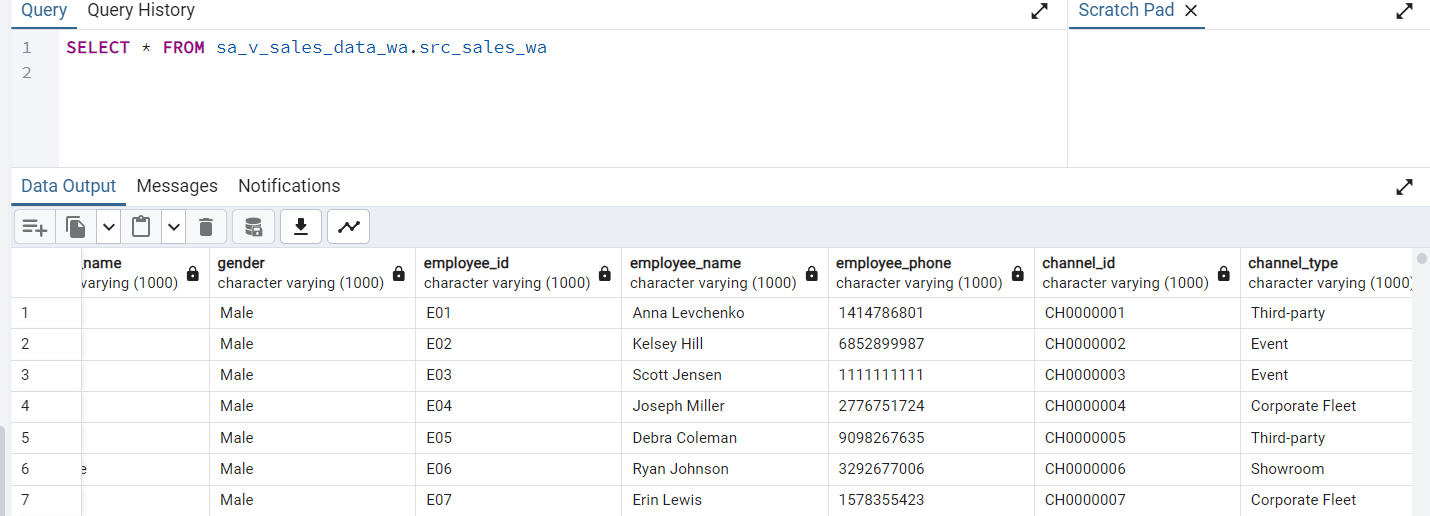


So firstly, I looked into external table:



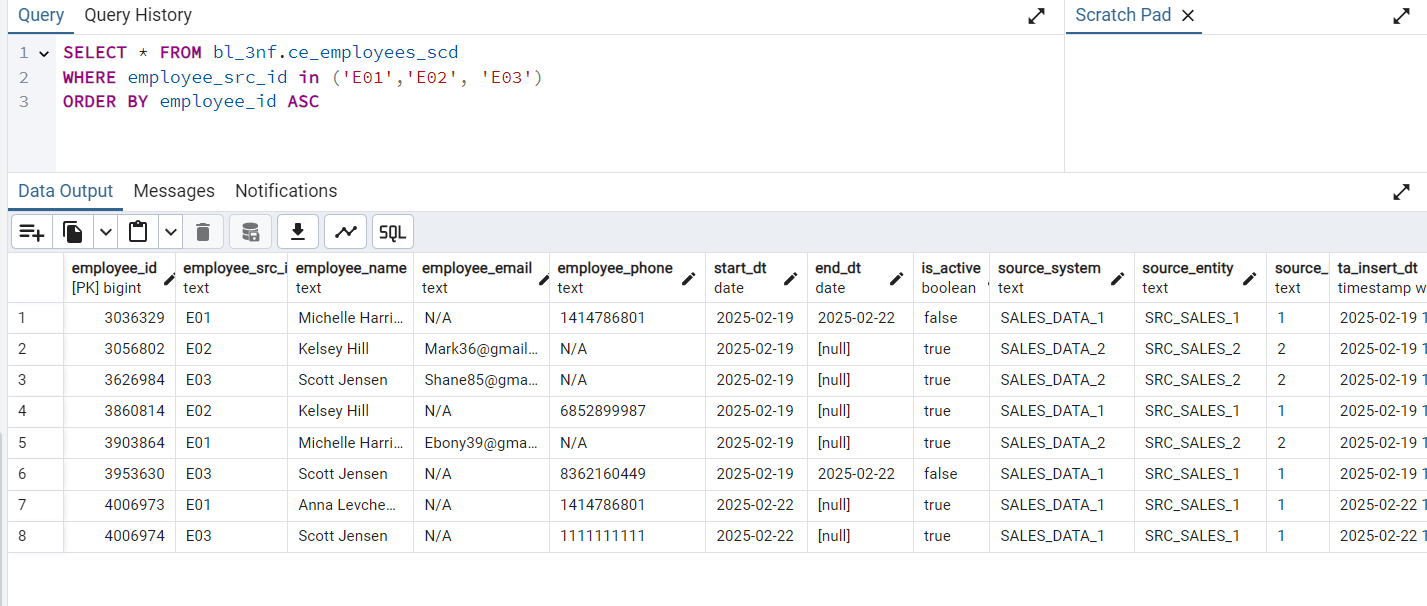
Fine, data was changed automatically.

Then I used a script from some previous hw for inserting into src table and got next result. So here data was changed successfully also.



Then I tried to insert data to 3nf but my query from previous hw was not working, then I did a new one, but it runs too long, so that’s why I created only for specific employee\_id. You can find new query in emoloyee\_3nf\_new.sql

And it gives me correct result:



Then I was loading to dim, that takes values from 3nf, so result seems correct:

