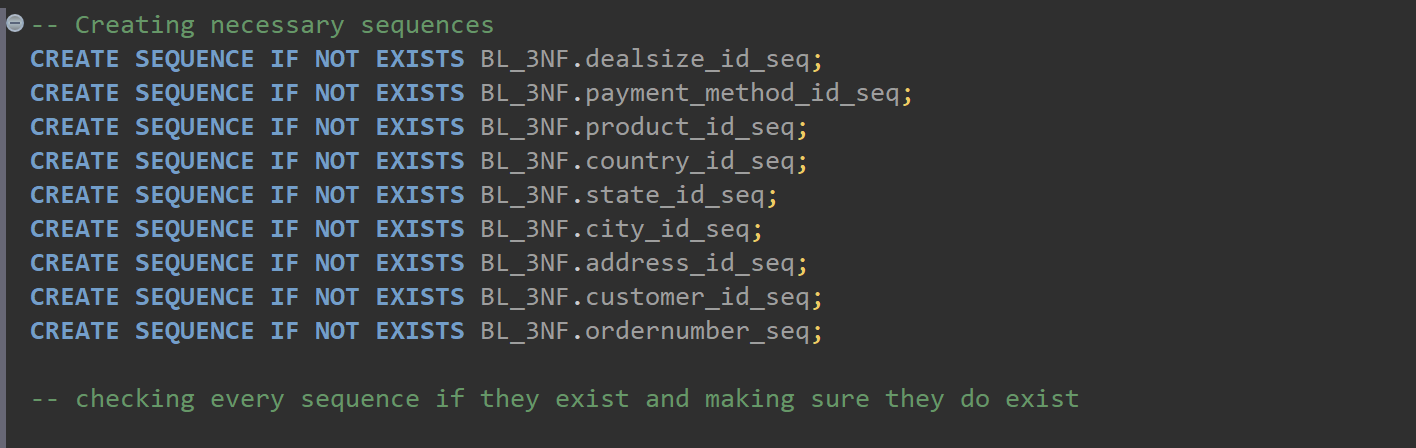
Here I write the process for loading data into a 3NF schema using PL/pgSQL functions. The idea is to ensure data integrity and track procedure executions using a logging mechanism.





**Creating Logging Table**

A logging table is created to keep track of the execution of data loading procedures. This table stores information about the procedure name, rows affected, log message, and timestamp.

This table is used to store log entries for each data loading procedure.

log\_id: This is like the entry number in our journal. It’s a unique number that automatically increases with each new log entry.

procedure\_name: This tells the name of the procedure that is ran, so it known what operation the log entry is about.

rows\_affected: This shows how many rows of data were processed by the procedure. It's like noting down how many tasks were completed.

log\_message: This is a short description of what happened. It could be something like "Dealsizes loaded successfully"

log\_timestamp: This records the exact time when the log entry was made. It helps know when each procedure was run.

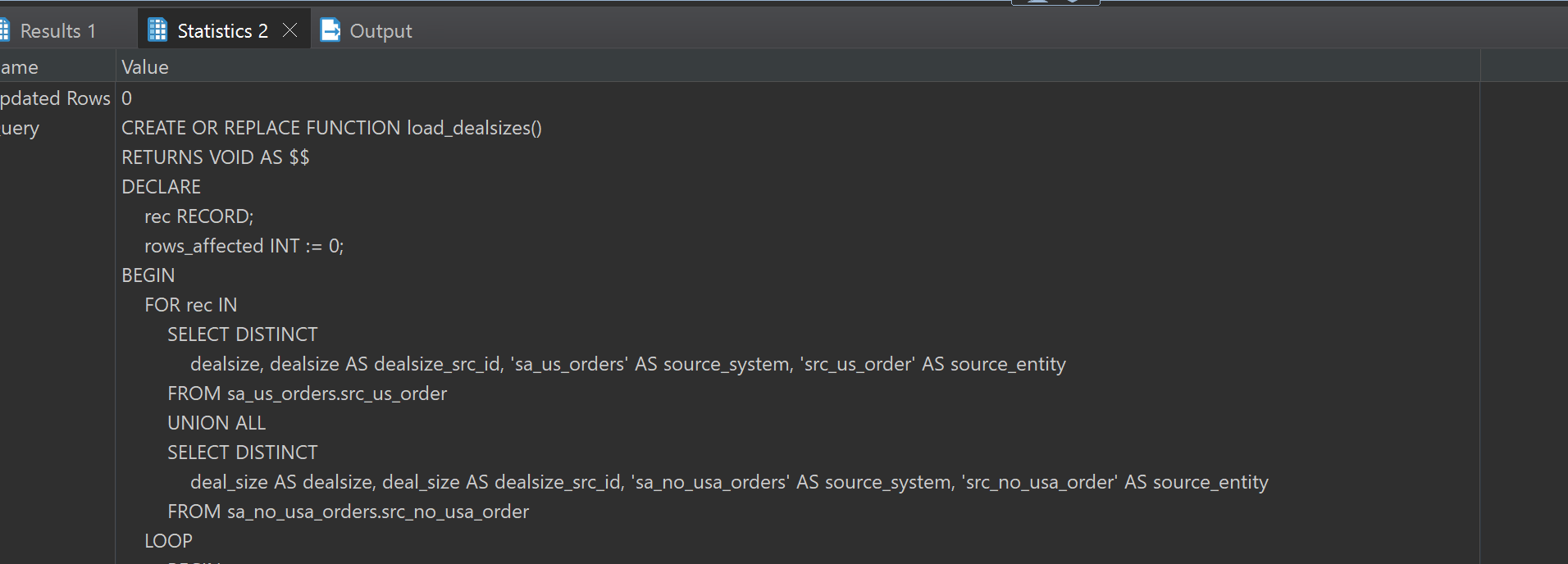
And then next function writing entries into it. We call this function at the end of each data loading procedure to record what happened.

proc\_name is name of the procedure where I am logging. Rows is number of rows the procedure affected. Message is describing what the procedure did.

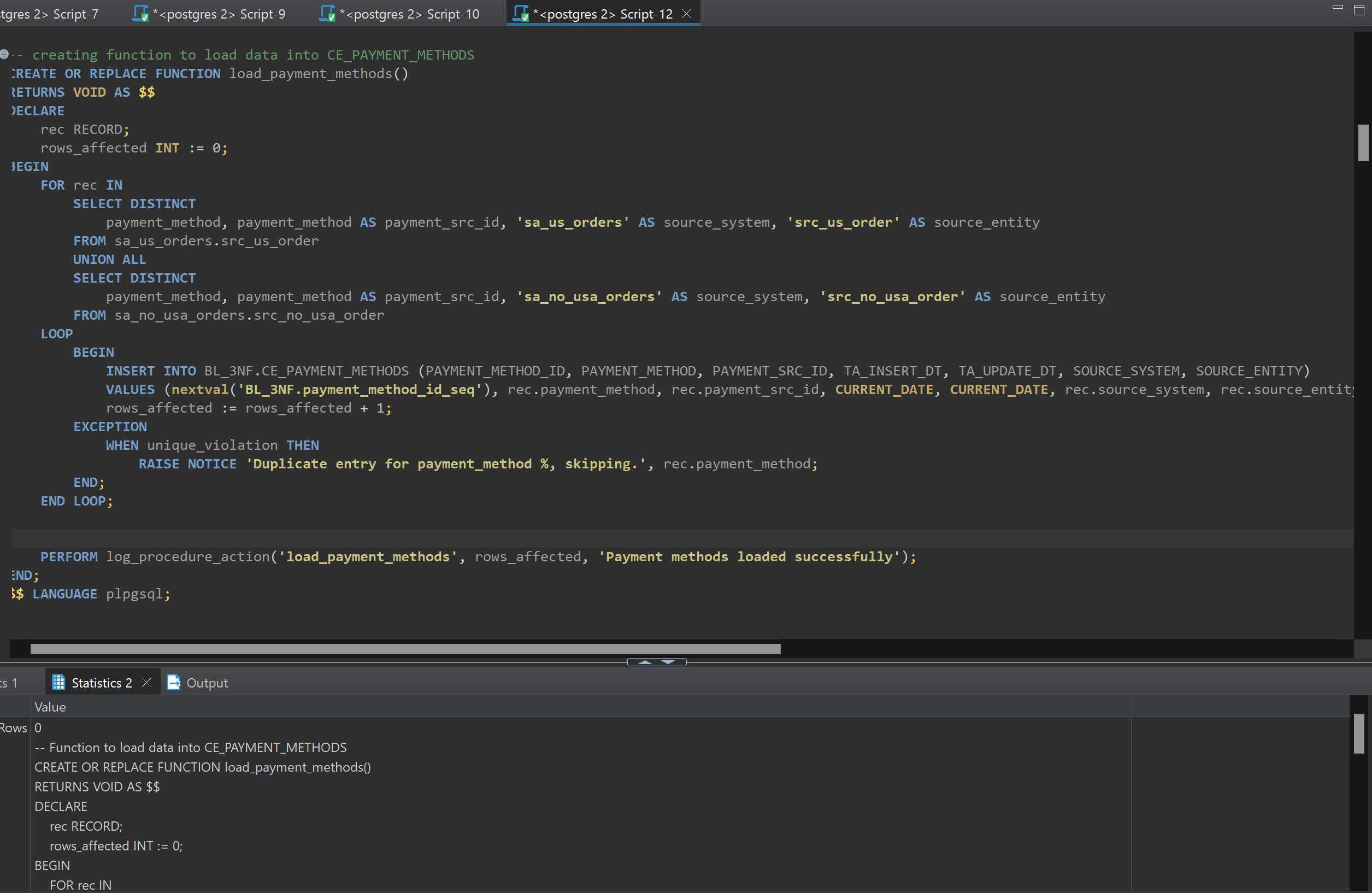
Data Loading Functions:

This function loads data into the CE\_DEALSIZES table. It selects distinct deal sizes from the source tables and inserts them into the target table, handling duplicates appropriately. And basically all the tables are basically same.

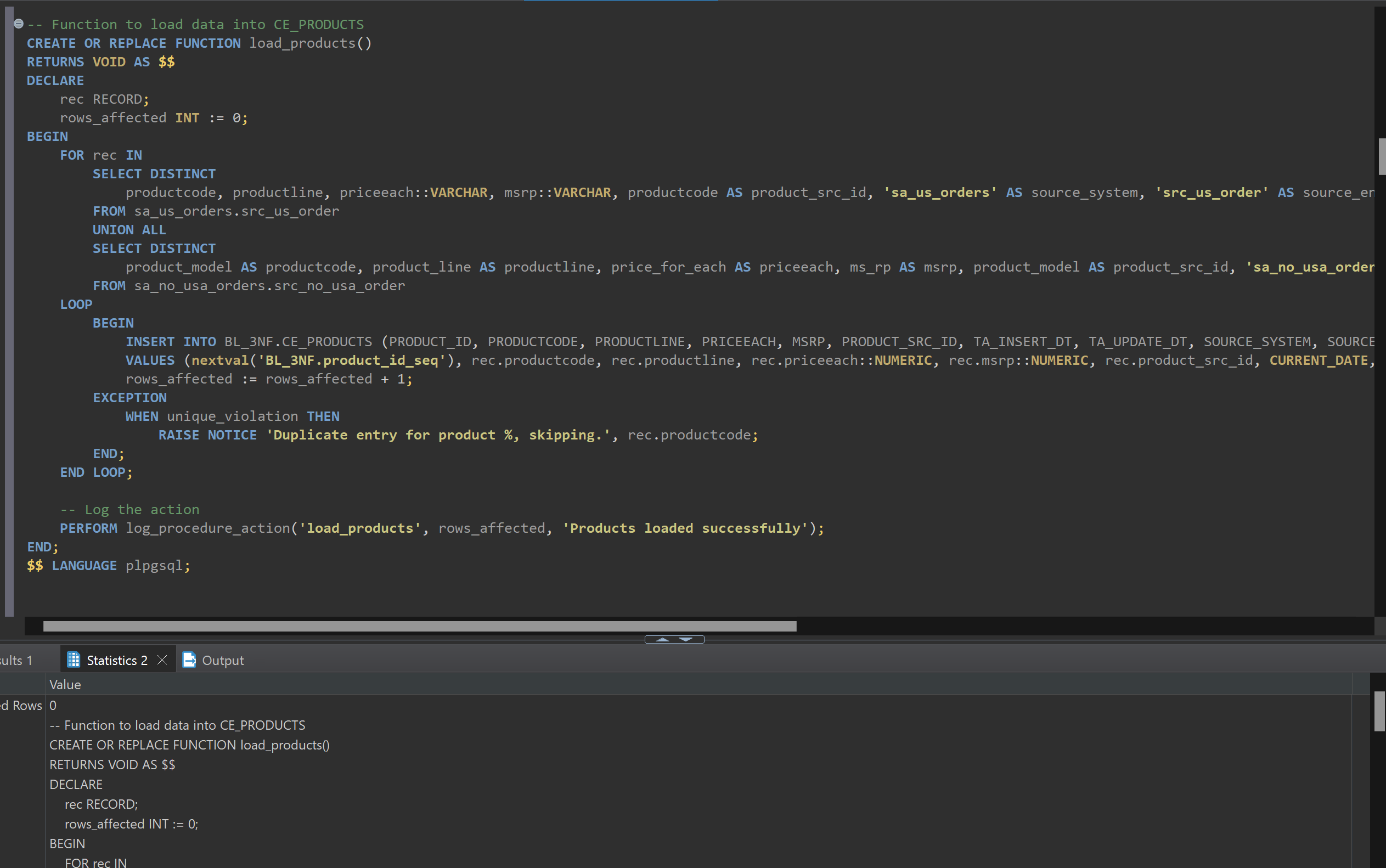




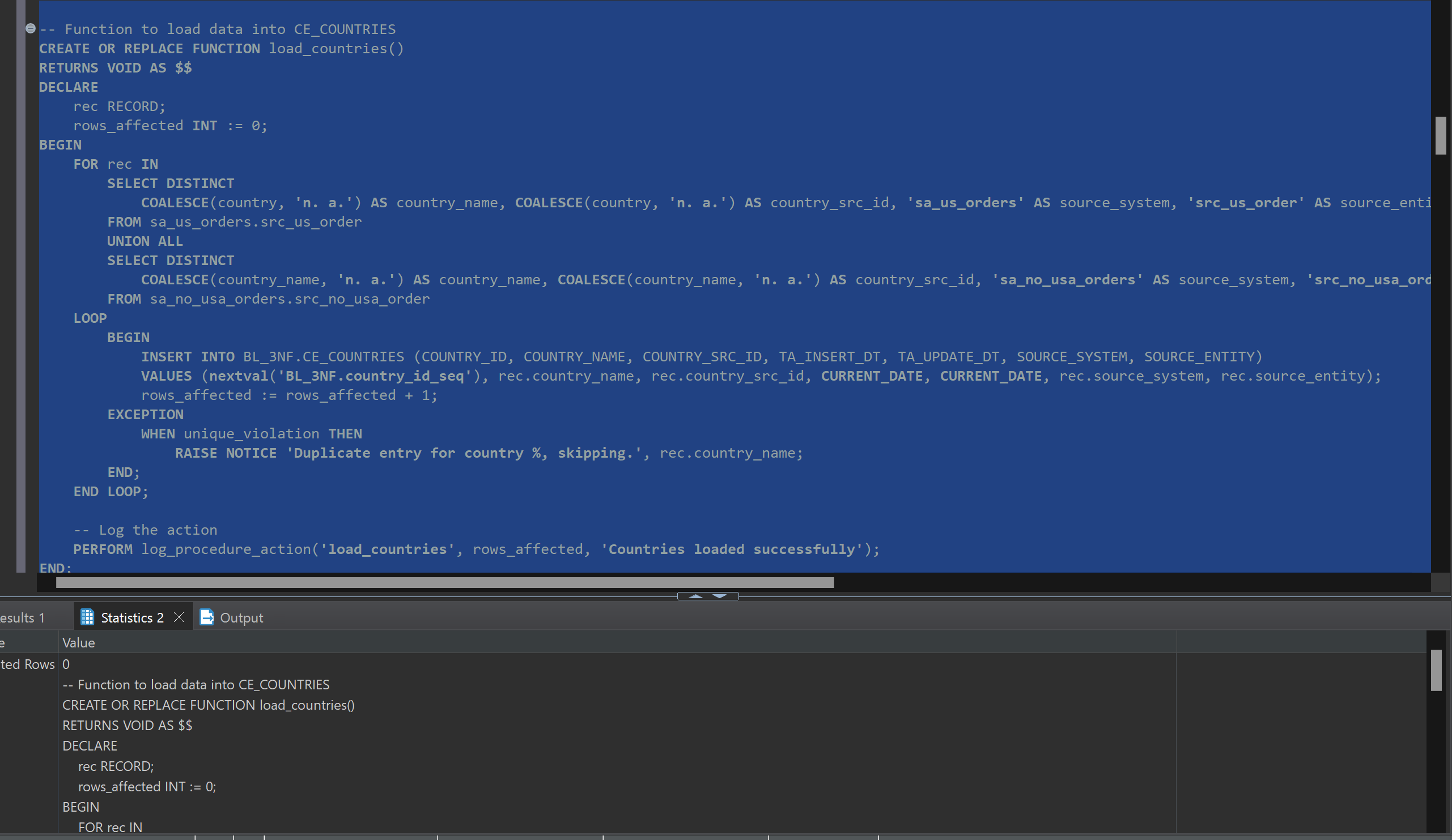
Payments:



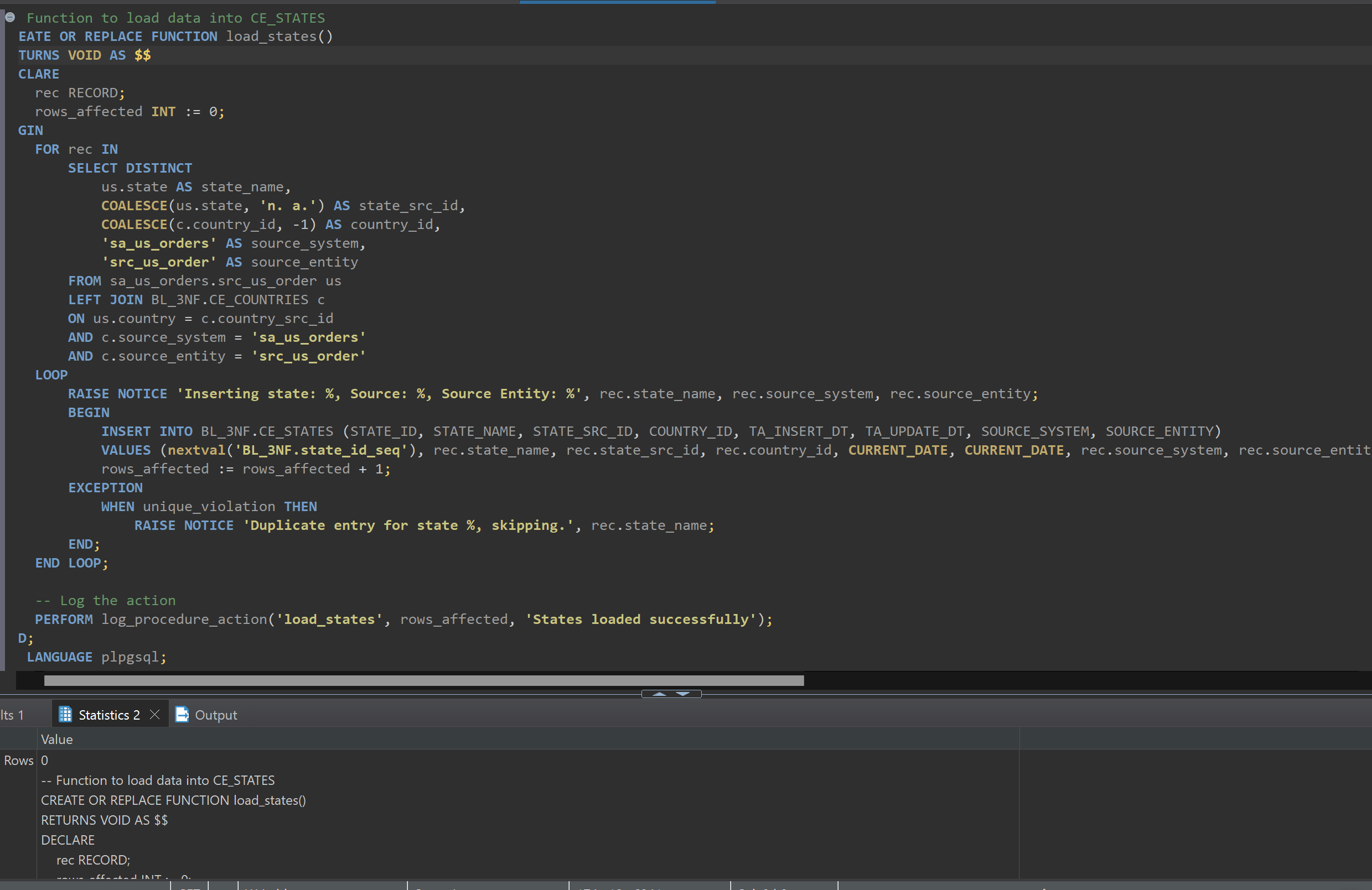
Products:

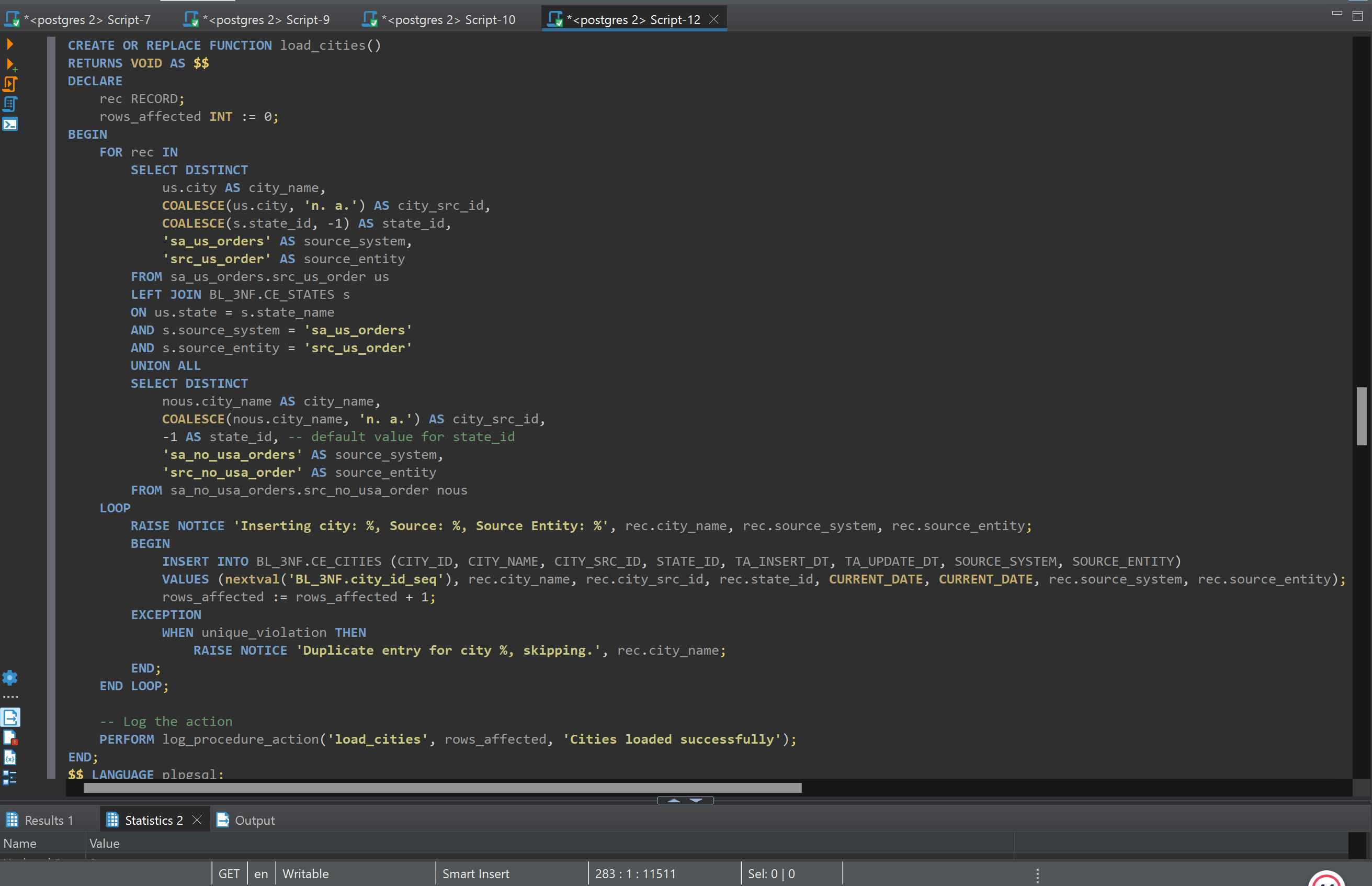


Countries:



States



Cities: 

Address :



Customer: **CREATE** **OR** **REPLACE** **FUNCTION** load\_customers()

**RETURNS** **VOID** **AS** **$$**

**DECLARE**

rec RECORD;

rows\_affected **INT** := 0;

**BEGIN**

**FOR** rec **IN**

**SELECT** **DISTINCT**

**COALESCE**(us.customer\_id::**BIGINT**, -1) **AS** customer\_id,

**COALESCE**(us.customername, **'n. a.'**) **AS** customername,

**COALESCE**(us.contactfirstname, **'n. a.'**) **AS** contactfirstname,

**COALESCE**(us.contactlastname, **'n. a.'**) **AS** contactlastname,

**COALESCE**(us.phone, **'n. a.'**) **AS** phone,

**COALESCE**(us.customer\_id::**VARCHAR**, **'n. a.'**) **AS** customer\_src\_id,

**COALESCE**(a.address\_id, -1) **AS** address\_id,

**'sa\_us\_orders'** **AS** source\_system,

**'src\_us\_order'** **AS** source\_entity

**FROM** sa\_us\_orders.src\_us\_order us

**LEFT** **JOIN** BL\_3NF.CE\_ADDRESSES a

**ON** us.addressline1 = a.addressline1

**AND** a.source\_system = **'sa\_us\_orders'**

**AND** a.source\_entity = **'src\_us\_order'**

**UNION** **ALL**

**SELECT** **DISTINCT**

**COALESCE**(nous.customers\_id::**BIGINT**, -1) **AS** customer\_id,

**COALESCE**(nous.cust\_name, **'n. a.'**) **AS** customername,

**COALESCE**(nous.cust\_firstname, **'n. a.'**) **AS** contactfirstname,

**COALESCE**(nous.cust\_lastname, **'n. a.'**) **AS** contactlastname,

**COALESCE**(nous.phone\_number, **'n. a.'**) **AS** phone,

**COALESCE**(nous.customers\_id::**VARCHAR**, **'n. a.'**) **AS** customer\_src\_id,

**COALESCE**(a.address\_id, -1) **AS** address\_id,

**'sa\_no\_usa\_orders'** **AS** source\_system,

**'src\_no\_usa\_order'** **AS** source\_entity

**FROM** sa\_no\_usa\_orders.src\_no\_usa\_order nous

**LEFT** **JOIN** BL\_3NF.CE\_ADDRESSES a

**ON** nous.address\_line = a.addressline1

**AND** a.source\_system = **'sa\_no\_usa\_orders'**

**AND** a.source\_entity = **'src\_no\_usa\_order'**

**LOOP**

**RAISE** **NOTICE** **'Inserting customer: %, Source: %, Source Entity: %'**, rec.customername, rec.source\_system, rec.source\_entity;

**BEGIN**

**INSERT** **INTO** BL\_3NF.CE\_CUSTOMERS\_SCD (CUSTOMER\_ID, CUSTOMERNAME, CONTACTFIRSTNAME, CONTACTLASTNAME, PHONE, CUSTOMER\_SRC\_ID, ADDRESS\_ID, START\_DT, END\_DT, IS\_ACTIVE, TA\_INSERT\_DT, SOURCE\_SYSTEM, SOURCE\_ENTITY)

**VALUES** (**nextval**(**'BL\_3NF.customer\_id\_seq'**), rec.customername, rec.contactfirstname, rec.contactlastname, rec.phone, rec.customer\_src\_id, rec.address\_id, **CURRENT\_DATE**, **'9999-12-31'**, **'Y'**, **CURRENT\_DATE**, rec.source\_system, rec.source\_entity);

rows\_affected := rows\_affected + 1;

**EXCEPTION**

**WHEN** unique\_violation **THEN**

**RAISE** **NOTICE** **'Duplicate entry for customer %, skipping.'**, rec.customername;

**END**;

**END** **LOOP**;

-- log

**PERFORM** log\_procedure\_action(**'load\_customers'**, rows\_affected, **'Customers loaded successfully'**);

**END**;

**$$** **LANGUAGE** plpgsql;

* Orders
* **CREATE** **OR** **REPLACE** **FUNCTION** load\_orders()
* **RETURNS** **VOID** **AS** **$$**
* **DECLARE**
* rec RECORD;
* rows\_affected **INT** := 0;
* **BEGIN**
* **FOR** rec **IN**
* **SELECT** **DISTINCT**
* us.ordernumber::**BIGINT** **AS** ordernumber,
* us.quantityordered::**INTEGER** **AS** quantityordered,
* us.sales::**NUMERIC** **AS** sales,
* pm.PAYMENT\_METHOD\_ID **AS** payment\_method\_id,
* ds.DEALSIZE\_ID **AS** dealsize\_id,
* p.PRODUCT\_ID **AS** product\_id,
* c.CUSTOMER\_ID **AS** customer\_id,
* us.orderdate::**DATE** **AS** event\_dt,
* **'sa\_us\_orders'** **AS** source\_system,
* **'src\_us\_order'** **AS** source\_entity,
* us.ordernumber::**VARCHAR** **AS** order\_src\_id
* **FROM** sa\_us\_orders.src\_us\_order us
* **LEFT** **JOIN** BL\_3NF.CE\_PAYMENT\_METHODS pm
* **ON** us.payment\_method = pm.payment\_method
* **AND** pm.source\_system = **'sa\_us\_orders'**
* **AND** pm.source\_entity = **'src\_us\_order'**
* **LEFT** **JOIN** BL\_3NF.CE\_DEALSIZES ds
* **ON** us.dealsize = ds.dealsize
* **AND** ds.source\_system = **'sa\_us\_orders'**
* **AND** ds.source\_entity = **'src\_us\_order'**
* **LEFT** **JOIN** BL\_3NF.CE\_PRODUCTS p
* **ON** us.productcode = p.productcode
* **AND** p.source\_system = **'sa\_us\_orders'**
* **AND** p.source\_entity = **'src\_us\_order'**
* **LEFT** **JOIN** BL\_3NF.CE\_CUSTOMERS\_SCD c
* **ON** us.customer\_id::**BIGINT** = c.customer\_id
* **AND** c.source\_system = **'sa\_us\_orders'**
* **AND** c.source\_entity = **'src\_us\_order'**
* **UNION** **ALL**
* **SELECT** **DISTINCT**
* nous.order\_id::**BIGINT** **AS** ordernumber,
* nous.quantity::**INTEGER** **AS** quantityordered,
* nous.sales\_amount::**NUMERIC** **AS** sales,
* pm.PAYMENT\_METHOD\_ID **AS** payment\_method\_id,
* ds.DEALSIZE\_ID **AS** dealsize\_id,
* p.PRODUCT\_ID **AS** product\_id,
* c.CUSTOMER\_ID **AS** customer\_id,
* nous.date\_of\_order::**DATE** **AS** event\_dt,
* **'sa\_no\_usa\_orders'** **AS** source\_system,
* **'src\_no\_usa\_order'** **AS** source\_entity,
* nous.order\_id::**VARCHAR** **AS** order\_src\_id
* **FROM** sa\_no\_usa\_orders.src\_no\_usa\_order nous
* **LEFT** **JOIN** BL\_3NF.CE\_PAYMENT\_METHODS pm
* **ON** nous.payment\_method = pm.payment\_method
* **AND** pm.source\_system = **'sa\_no\_usa\_orders'**
* **AND** pm.source\_entity = **'src\_no\_usa\_order'**
* **LEFT** **JOIN** BL\_3NF.CE\_DEALSIZES ds
* **ON** nous.deal\_size = ds.dealsize
* **AND** ds.source\_system = **'sa\_no\_usa\_orders'**
* **AND** ds.source\_entity = **'src\_no\_usa\_order'**
* **LEFT** **JOIN** BL\_3NF.CE\_PRODUCTS p
* **ON** nous.product\_model = p.productcode
* **AND** p.source\_system = **'sa\_no\_usa\_orders'**
* **AND** p.source\_entity = **'src\_no\_usa\_order'**
* **LEFT** **JOIN** BL\_3NF.CE\_CUSTOMERS\_SCD c
* **ON** nous.customers\_id::**BIGINT** = c.customer\_id
* **AND** c.source\_system = **'sa\_no\_usa\_orders'**
* **AND** c.source\_entity = **'src\_no\_usa\_order'**
* **LOOP**
* -- Check for NULL values in mandatory fields
* **IF** rec.payment\_method\_id **IS** **NULL** **OR** rec.dealsize\_id **IS** **NULL** **OR** rec.product\_id **IS** **NULL** **OR** rec.customer\_id **IS** **NULL** **THEN**
* **RAISE** **NOTICE** **'Skipping order % due to NULL values in mandatory fields.'**, rec.ordernumber;
* **CONTINUE**;
* **END** **IF**;
* **RAISE** **NOTICE** **'Inserting order: %, Source: %, Source Entity: %'**, rec.ordernumber, rec.source\_system, rec.source\_entity;
* **BEGIN**
* **INSERT** **INTO** BL\_3NF.CE\_ORDERS (ORDERNUMBER, QUANTITYORDERED, SALES, PAYMENT\_METHOD\_ID, DEALSIZE\_ID, PRODUCT\_ID, CUSTOMER\_ID, EVENT\_DT, TA\_INSERT\_DT, TA\_UPDATE\_DT, ORDER\_SRC\_ID, SOURCE\_SYSTEM, SOURCE\_ENTITY)
* **VALUES** (**COALESCE**(rec.ordernumber, **nextval**(**'BL\_3NF.ordernumber\_seq'**)), rec.quantityordered, rec.sales, rec.payment\_method\_id, rec.dealsize\_id, rec.product\_id, rec.customer\_id, rec.event\_dt, **CURRENT\_DATE**, **CURRENT\_DATE**, rec.order\_src\_id, rec.source\_system, rec.source\_entity);
* rows\_affected := rows\_affected + 1;
* **EXCEPTION**
* **WHEN** unique\_violation **THEN**
* **RAISE** **NOTICE** **'Duplicate entry for order %, skipping.'**, rec.ordernumber;
* **END**;
* **END** **LOOP**;
* -- loggingn
* **PERFORM** log\_procedure\_action(**'load\_orders'**, rows\_affected, **'Orders loaded successfully'**);
* **END**;
* **$$** **LANGUAGE** plpgsql;

The load\_something function is designed to efficiently load data into the CE\_something table from the sa\_us\_orders.src\_us\_order table. It starts by setting up variables to handle incoming data and count the number of rows inserted. This preparation ensures smooth data processing and accurate tracking of inserted records.

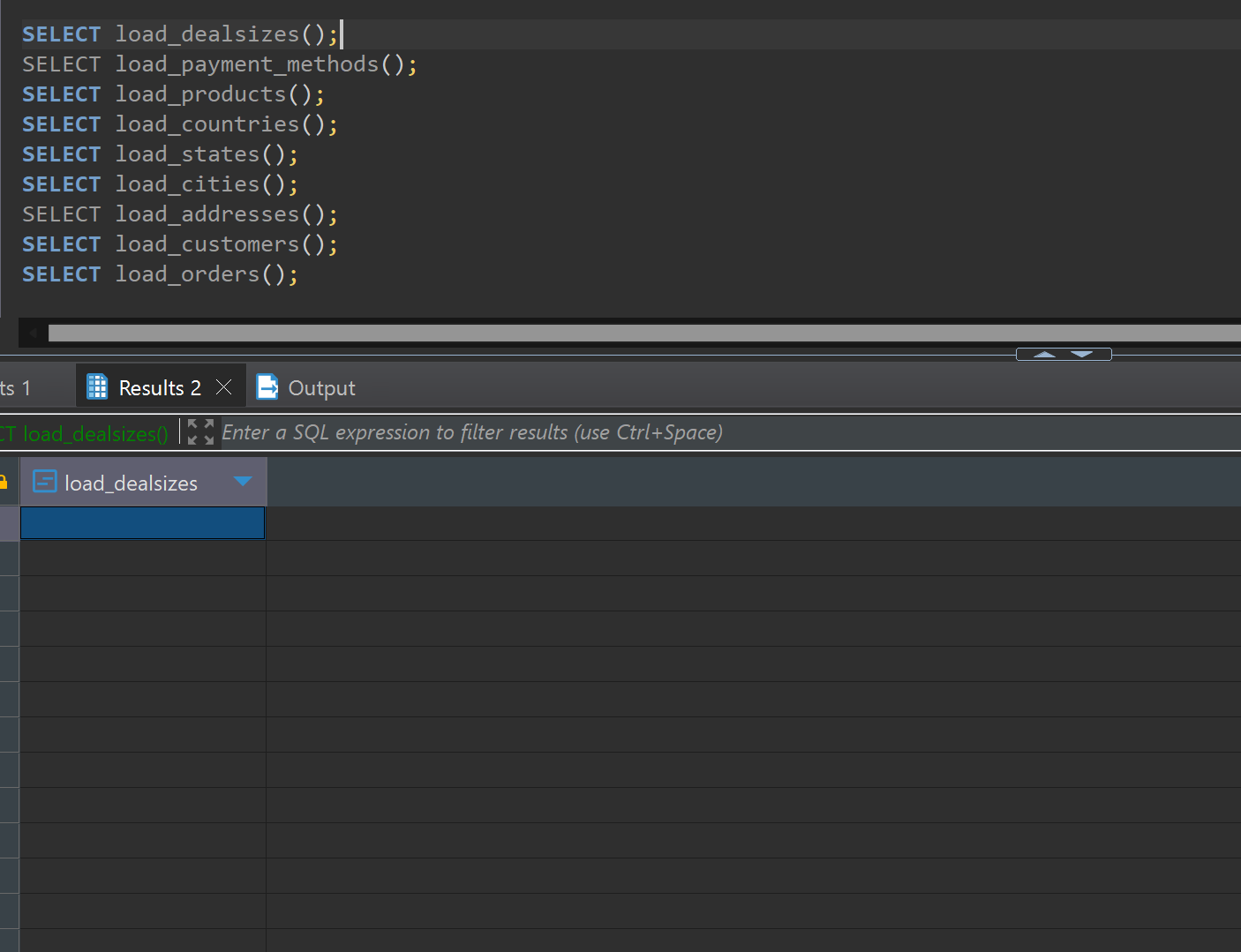
The function uses a FOR loop to go through distinct states from the source table. During this process, it performs a LEFT JOIN with the CE\_something table to get the corresponding something\_id for each state. If no match is found, it uses -1 as a default something\_id, ensuring every state has an associated ID.

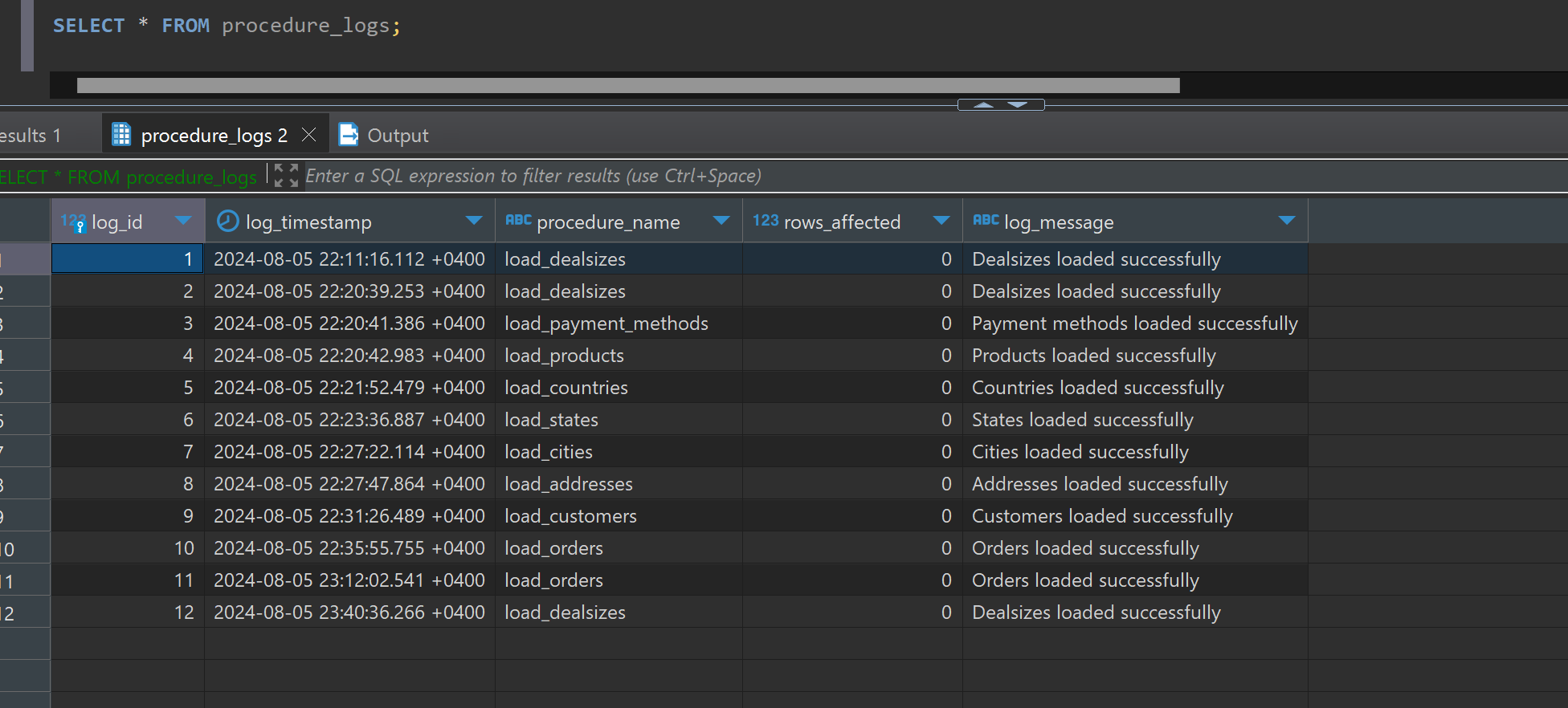
Within the loop, the function tries to insert each state record into the CE\_something table, generating a unique identifier using the sequence nextval('BL\_3NF.something\_id\_seq'). If a duplicate entry is encountered, it raises a notice and skips the duplicate, ensuring only unique data is inserted.

After processing all records, the function logs the execution details by calling the log\_procedure\_action function. This log entry includes the procedure name, the number of rows affected, and a success message, providing traceability and aiding in debugging.

In summary, the load\_something function efficiently loads data into the CE\_something table, handles duplicates, and logs the process. This approach ensures data integrity and provides a reliable record of data loading operations, supporting transparency and troubleshooting.

Calling functions



Checking log tables: 

Now verifying the bl 3nf tables

