# Task D: Procedures

## 1. \*\*`sp\_apply\_category\_discount(p\_category TEXT, p\_percent NUMERIC)`\*\*     - Reduce `unit\_price` of \*\*active\*\* products in a category by `p\_percent` (e.g., 10 = 10%). Prevent negative or zero prices using a `CHECK` at update time.

**CREATE** **OR** **REPLACE** **PROCEDURE** training\_ecom.sp\_apply\_category\_discount(

p\_category **TEXT**,

p\_percent **NUMERIC**

)

**LANGUAGE** plpgsql **AS** **$$**

**BEGIN**

**UPDATE** training\_ecom.products

**SET** unit\_price = unit\_price \* (1 - p\_percent / 100)

**WHERE** category = p\_category

**AND** active = **TRUE**

**AND** unit\_price \* (1 - p\_percent / 100) > 0;

**IF** **NOT** **FOUND** **THEN**

**RAISE** **NOTICE** 'No active products found in category % or discount would result in non-positive prices.', p\_category;

**END** **IF**;

**END**;

**$$**;

**ALTER** **TABLE** training\_ecom.products

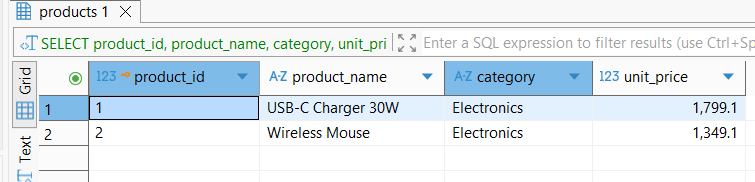
**ADD** **CONSTRAINT** chk\_positive\_unit\_price **CHECK** (unit\_price > 0);

**CALL** training\_ecom.sp\_apply\_category\_discount('Electronics', 10);

**SELECT** product\_id, product\_name, category, unit\_price

**FROM** training\_ecom.products

**WHERE** category = 'Electronics';



## 2. \*\*`sp\_cancel\_order(p\_order\_id INT)`\*\*     - Set order `status` to `cancelled` \*\*only if\*\* it is not already `delivered`.     - (Optional) Delete unpaid `payments` if any exist for that order (there shouldn’t be, but handle defensively).

**CREATE** **OR** **REPLACE** **PROCEDURE** training\_ecom.sp\_cancel\_order(p\_order\_id **INT**)

**LANGUAGE** plpgsql **AS** **$$**

**BEGIN**

**UPDATE** training\_ecom.orders

**SET** status = 'cancelled'

**WHERE** order\_id = p\_order\_id

**AND** status != 'delivered';

**IF** **NOT** **FOUND** **THEN**

**RAISE** **NOTICE** 'Order % not found or already delivered.', p\_order\_id;

**END** **IF**;

-- Delete any unpaid payments (defensive, should not exist)

**DELETE** **FROM** training\_ecom.payments

**WHERE** order\_id = p\_order\_id

**AND** paid\_at **IS** **NULL**;

**END**;

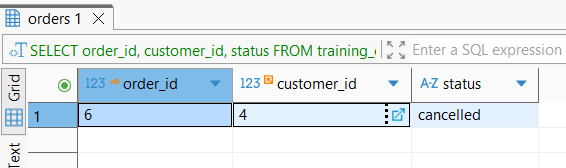
**$$**;

**CALL** training\_ecom.sp\_cancel\_order(6);

**SELECT** order\_id, customer\_id, status

**FROM** training\_ecom.orders

**WHERE** order\_id = 6;



## 3. \*\*`sp\_reprice\_stale\_products(p\_days INT, p\_increase NUMERIC)`\*\*     - For products \*\*not ordered\*\* in the last `p\_days`, increase `unit\_price` by `p\_increase` percent.

**CREATE** **OR** **REPLACE** **PROCEDURE** training\_ecom.sp\_reprice\_stale\_products(p\_days **INT**, p\_increase **NUMERIC**)

**LANGUAGE** plpgsql **AS** **$$**

**BEGIN**

**UPDATE** training\_ecom.products p

**SET** unit\_price = unit\_price \* (1 + p\_increase / 100)

**WHERE** **NOT** **EXISTS** (

**SELECT** 1

**FROM** training\_ecom.order\_items oi

**JOIN** training\_ecom.orders o **ON** oi.order\_id = o.order\_id

**WHERE** oi.product\_id = p.product\_id

**AND** o.order\_date >= CURRENT\_DATE - **INTERVAL** '1 day' \* p\_days

)

**AND** p.active = **TRUE**;

**IF** **NOT** **FOUND** **THEN**

**RAISE** **NOTICE** 'No stale products found in the last % days.', p\_days;

**END** **IF**;

**END**;

**$$**;

**CALL** training\_ecom.sp\_reprice\_stale\_products(30, 10);

**SELECT** product\_id, product\_name, unit\_price, active

**FROM** training\_ecom.products

**ORDER** **BY** product\_id;

