**Створення таблиць orders та orders\_logs**

CREATE TABLE orders (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

status varchar(50) NOT NULL,

product\_id int NOT NULL

);

CREATE TABLE orders\_logs (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

order\_id int NOT NULL,

status varchar(50) NOT NULL,

changed\_on TIMESTAMP(6) NOT NULL

);

**Створення функції**

CREATE OR REPLACE FUNCTION log\_order\_status\_changes()

RETURNS TRIGGER

LANGUAGE PLPGSQL

AS

$$

BEGIN

IF NEW.status <> OLD.status THEN

INSERT INTO orders\_logs(order\_id,status,changed\_on)

VALUES(OLD.id,OLD.status,now());

END IF;

RETURN NEW;

END;

$$

**Створення тригера**

CREATE TRIGGER order\_status\_changes

BEFORE UPDATE

ON orders

FOR EACH ROW

EXECUTE PROCEDURE log\_order\_status\_changes();

**Заповнення таблиці orders даними**

INSERT INTO orders (product\_id, status)

VALUES (1, new);

INSERT INTO orders (product\_id, status)

VALUES (3, new);

SELECT \* FROM orders;

UPDATE orders SET status = 'done' WHERE ID = 2;

SELECT \* FROM orders\_logs;

**Перейменування тригера**

ALTER TRIGGER order\_status\_status\_changes ON orders RENAME TO order\_status;

**Створення нової функціїї**

CREATE OR REPLACE FUNCTION check\_order\_status\_changes()

RETURNS TRIGGER

LANGUAGE PLPGSQL

AS

$$

BEGIN

IF NEW.status <> OLD.status THEN

INSERT INTO orders\_logs(order\_id,status,changed\_on)

VALUES(OLD.id,OLD.status,now());

END IF;

RETURN NEW;

END;

$$

**Присвоєння тригеру нової функції**

BEGIN;

DROP TRIGGER IF EXISTS order\_status ON orders;

CREATE TRIGGER order\_status

BEFORE UPDATE ON orders

FOR EACH ROW

EXECUTE PROCEDURE check\_order\_status\_chenges();

COMMIT;

**Вмикання та вимикання тригера**

ALTER TABLE orders DISABLE TRIGGER order\_status | ALL;

ALTER TABLE orders ENABLE TRIGGER order\_status | ALL;

**Видалення тригера**

DROP TRIGGER IF EXISTS order\_status ON orders [CASCADE | RISTRICT];

Спрощений варіант

DROP TRIGGER order\_status ON orders;