Міністерство освіти і науки України

Івано-Франківський національний технічний

університет нафти і газу

кафедра ІПЗ

**Лабораторна робота №3**

Тема: «Тригери та тригерні функції в PostgreSQL. Тригери в MySQL»

Варіант №40

Виконав:

студент групи ІП 21-3

Деренько О.В.

Перевірила:

Саманів Л.В.

м. Івано-Франківськ

2023 р.

**Мета:** Навчитися створювати тригерні функції в БД

**Завдання:** Створити 10 тригерних функцій в БД “Відвідування школи”

**Хід роботи**

1. Тригерна функція для логування змін використовується для відстеження та реєстрації змін, які відбуваються у таблиці children

-- Створення таблиці для зберігання журналу змін

CREATE TABLE children\_log (

log\_id SERIAL PRIMARY KEY,

log\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

operation CHAR(1), -- 'I' для вставки, 'U' для оновлення

childid INT,

class INT,

performancelevel VARCHAR(255),

parentcode INT

);

-- Створення тригера для таблиці children

CREATE OR REPLACE FUNCTION log\_children\_changes()

RETURNS TRIGGER AS $$

BEGIN

-- Логування вставок

IF TG\_OP = 'INSERT' THEN

INSERT INTO children\_log (operation, childid, class, performancelevel, parentcode)

VALUES ('I', NEW.childid, NEW.class, NEW.performancelevel, NEW.parentcode);

-- Логування оновлень

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO children\_log (operation, childid, class, performancelevel, parentcode)

VALUES ('U', NEW.childid, NEW.class, NEW.performancelevel, NEW.parentcode);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Створення тригера для таблиці children

CREATE TRIGGER children\_log\_trigger

AFTER INSERT OR UPDATE ON children

FOR EACH ROW

EXECUTE PROCEDURE log\_children\_changes();

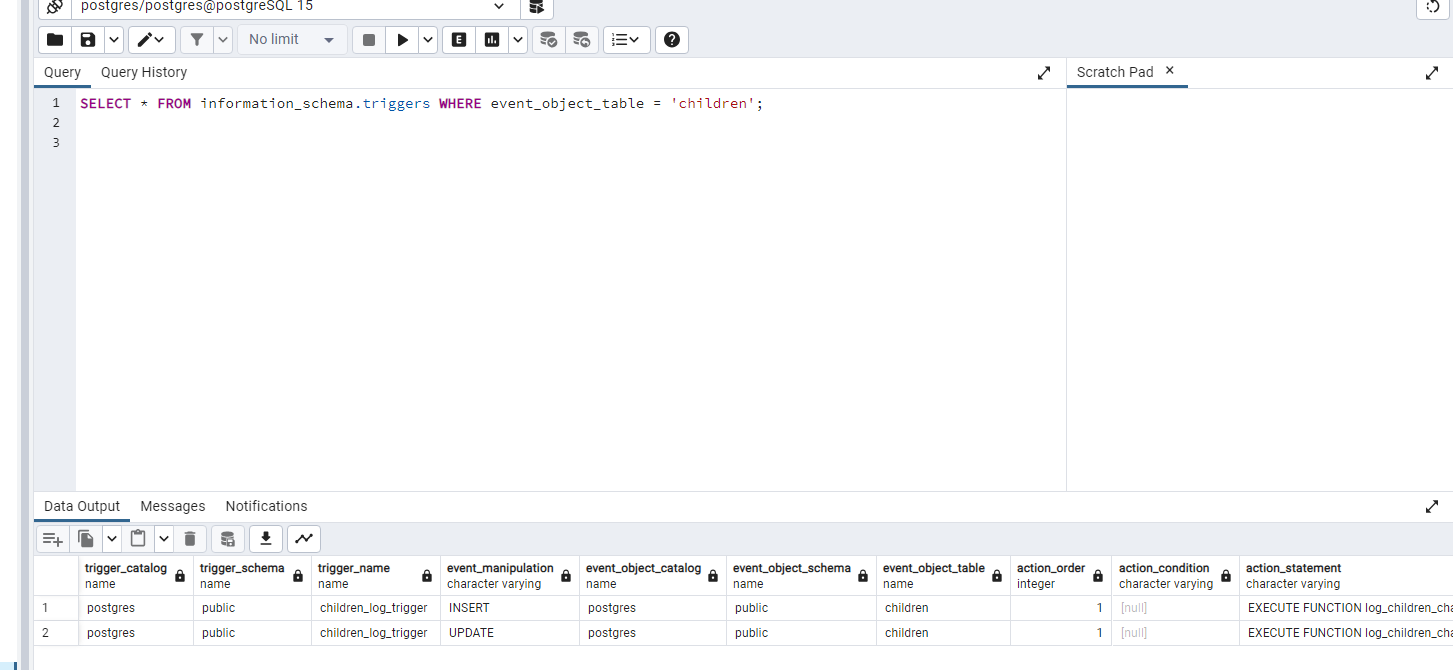


Рисунок 1.1 – Результат виконаної роботи

2. Тригерна функція для логування змін використовується для відстеження та реєстрації змін, які відбуваються у таблиці parents.

-- Створення функції для логування змін в таблиці parents

CREATE OR REPLACE FUNCTION log\_parents\_changes()

RETURNS TRIGGER AS $$

BEGIN

IF TG\_OP = 'INSERT' THEN

INSERT INTO parents\_log (action, parentcode, mothername, fathername, address, phonenumber)

VALUES ('INSERT', NEW.parentcode, NEW.mothername, NEW.fathername, NEW.address, NEW.phonenumber);

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO parents\_log (action, parentcode, mothername, fathername, address, phonenumber)

VALUES ('UPDATE', NEW.parentcode, NEW.mothername, NEW.fathername, NEW.address, NEW.phonenumber);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Створення тригера для таблиці parents

CREATE TRIGGER parents\_log\_trigger

AFTER INSERT OR UPDATE ON parents

FOR EACH ROW

EXECUTE PROCEDURE log\_parents\_changes();

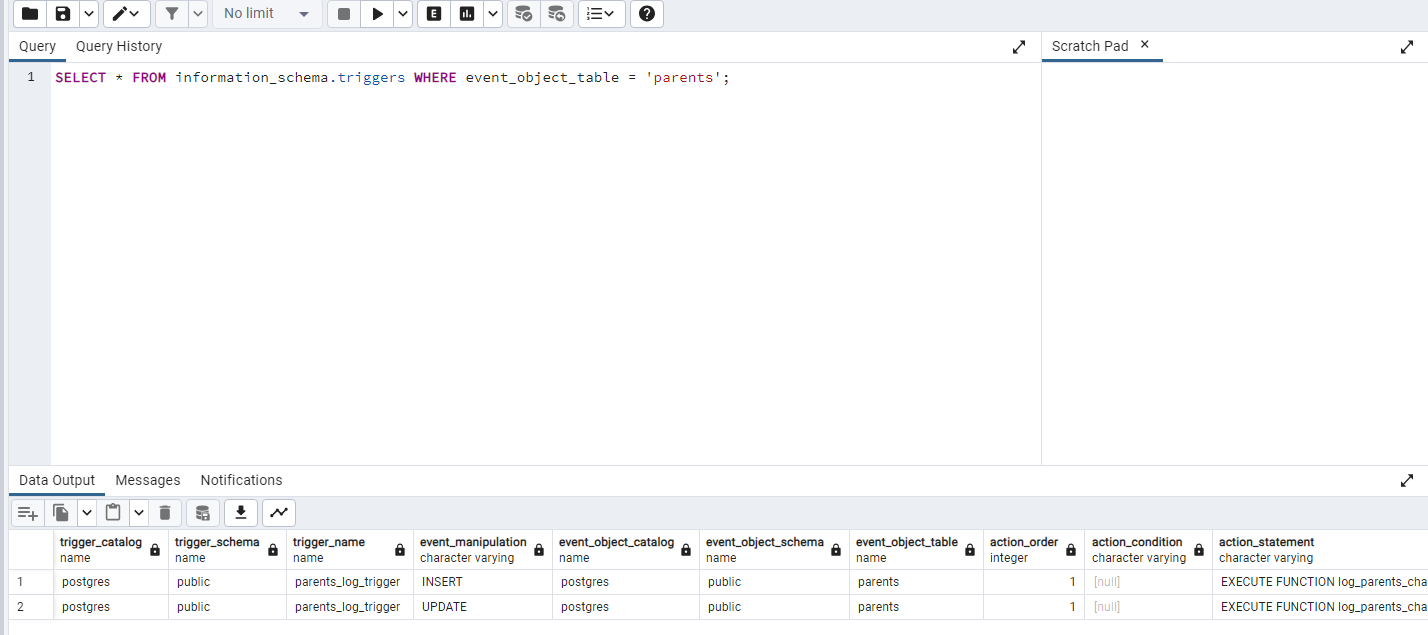


Рисунок 1.2 – Результат виконаної роботи

3. Тригерна функція для логування змін використовується для відстеження та реєстрації змін, які відбуваються у таблиці schoolvisits.

-- Створення лог-таблиці для зберігання інформації про зміни в таблиці schoolvisits

CREATE TABLE schoolvisits\_log (

log\_id SERIAL PRIMARY KEY,

action VARCHAR(10) NOT NULL,

timestamp TIMESTAMPTZ DEFAULT CURRENT\_TIMESTAMP,

visitid INTEGER,

visitdate DATE,

childid INTEGER,

childname VARCHAR(255),

absent CHAR(1)

);

-- Створення функції для логування змін в таблиці schoolvisits

CREATE OR REPLACE FUNCTION log\_schoolvisits\_changes()

RETURNS TRIGGER AS $$

BEGIN

-- Якщо запис вставлено

IF TG\_OP = 'INSERT' THEN

INSERT INTO schoolvisits\_log (action, visitid, visitdate, childid, childname, absent)

VALUES ('INSERT', NEW.visitid, NEW.visitdate, NEW.childid, NEW.childname, NEW.absent);

-- Якщо запис оновлено

ELSIF TG\_OP = 'UPDATE' THEN

INSERT INTO schoolvisits\_log (action, visitid, visitdate, childid, childname, absent)

VALUES ('UPDATE', OLD.visitid, OLD.visitdate, OLD.childid, OLD.childname, OLD.absent);

-- Якщо запис видалено

ELSIF TG\_OP = 'DELETE' THEN

INSERT INTO schoolvisits\_log (action, visitid, visitdate, childid, childname, absent)

VALUES ('DELETE', OLD.visitid, OLD.visitdate, OLD.childid, OLD.childname, OLD.absent);

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

-- Налаштування тригера для таблиці schoolvisits

CREATE TRIGGER schoolvisits\_changes\_trigger

AFTER INSERT OR UPDATE OR DELETE ON schoolvisits

FOR EACH ROW

EXECUTE PROCEDURE log\_schoolvisits\_changes();

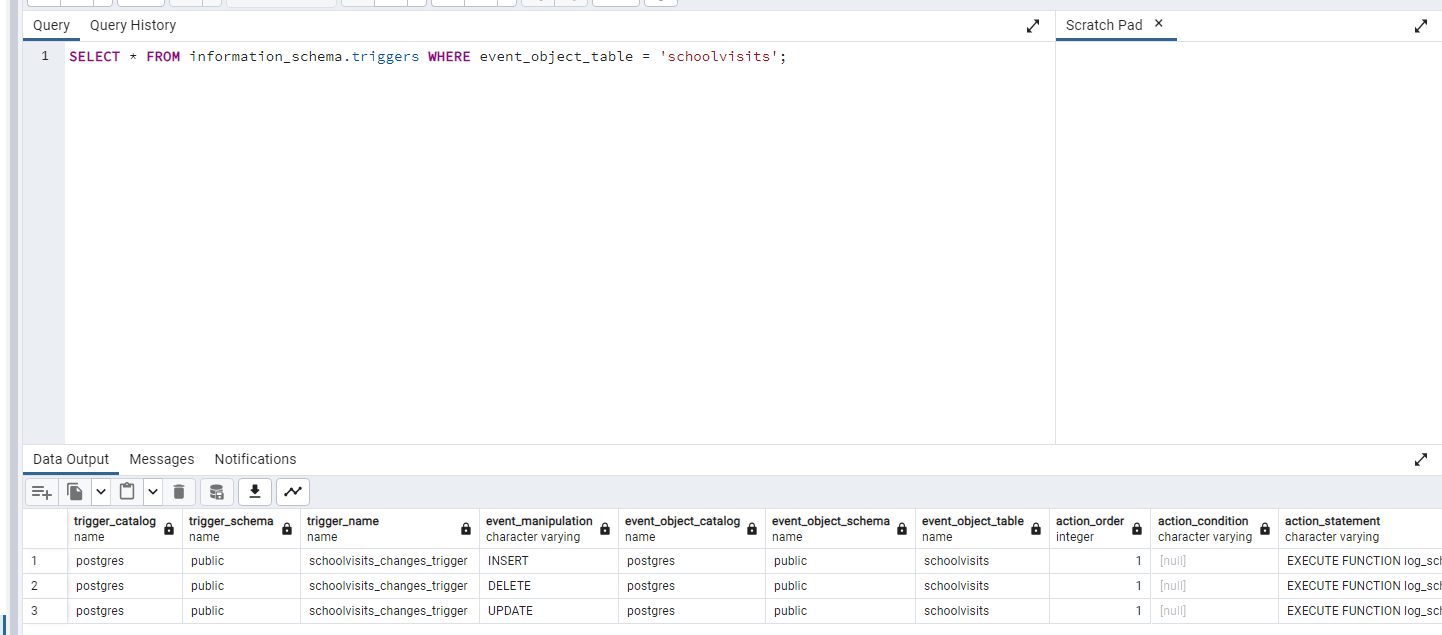


Рисунок 1.3 – Результат виконаної роботи

4. Тригерна функція яка автоматично оновлює рівень успішності учня в таблиці children на основі даних у таблиці schoolvisits при оновленні статусу відвідування. Якщо відвідування відзначено як ('t'), то рівень виконання стає 'Задовільно', якщо ('f') 'Добре', в іншому випадку - 'Відмінно'.

-- Створення функції для оновлення рівня ефективності дитини на основі відвідуваності школи

CREATE OR REPLACE FUNCTION update\_child\_performance()

RETURNS TRIGGER AS $$

BEGIN

-- Оновлення рівня ефективності у таблиці "children" на основі відвідуваності школи

UPDATE children

SET performancelevel =

CASE

WHEN NEW.absent = 't' THEN 'Задовільно'

WHEN NEW.absent = 'f' THEN 'Добре'

WHEN NEW.absent = 'e' THEN 'Відмнінно'

END

WHERE childid = NEW.childid;

-- Повернення нового запису

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

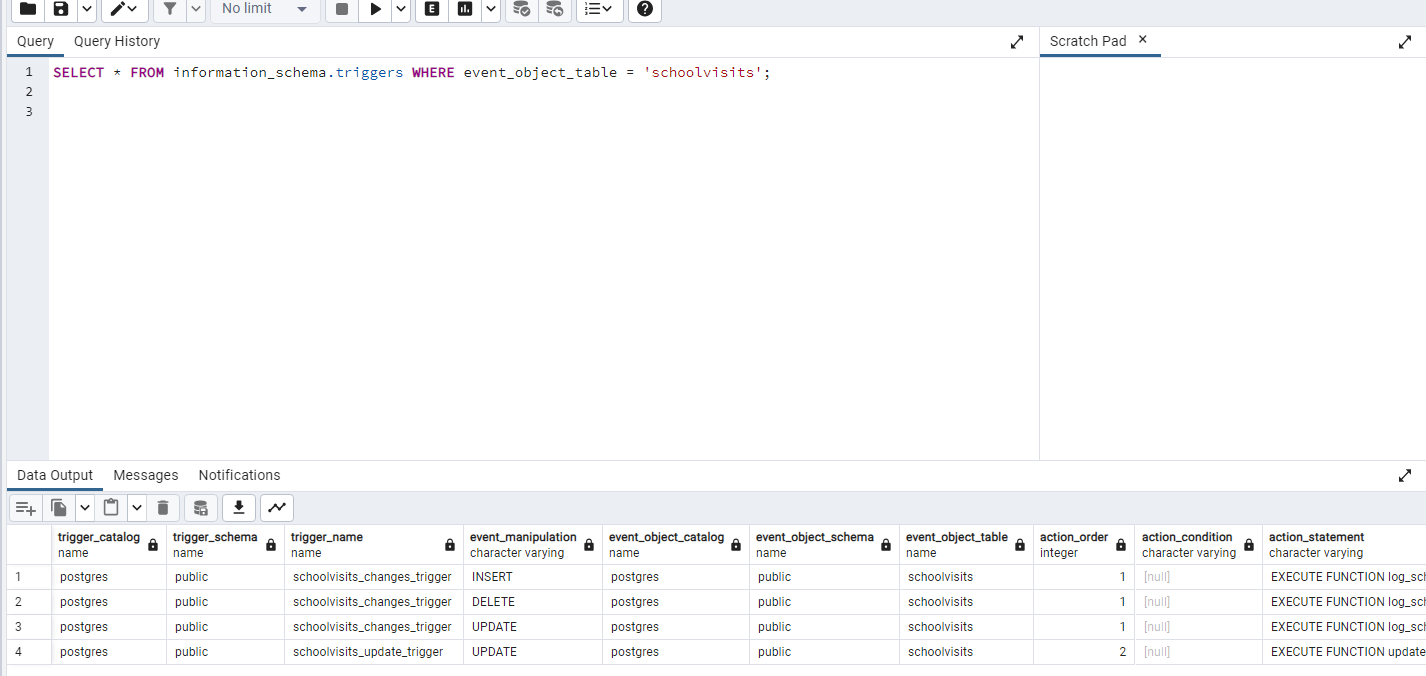
-- Створення тригера, який викликає функцію після оновлення запису в таблиці "schoolvisits"

CREATE TRIGGER schoolvisits\_update\_trigger

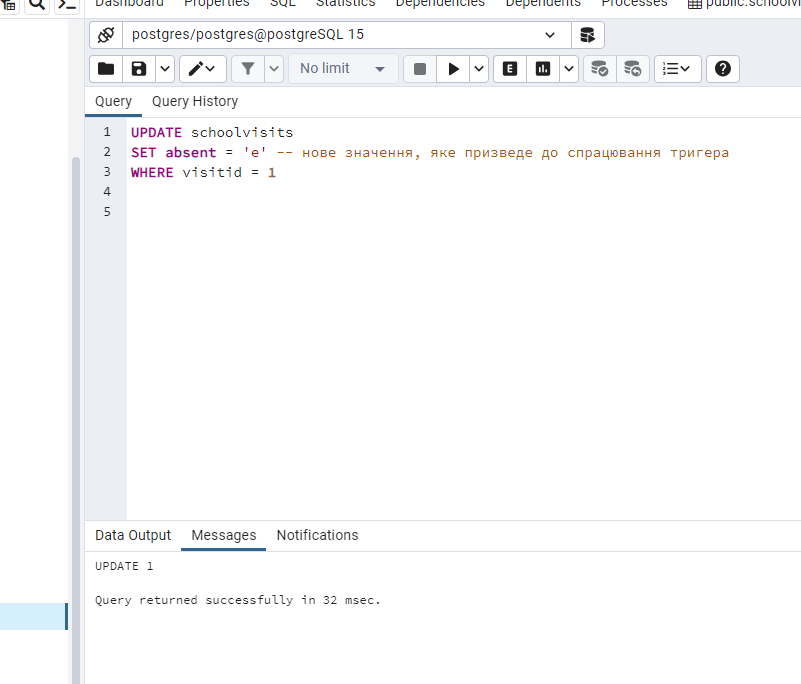
AFTER UPDATE ON schoolvisits

FOR EACH ROW

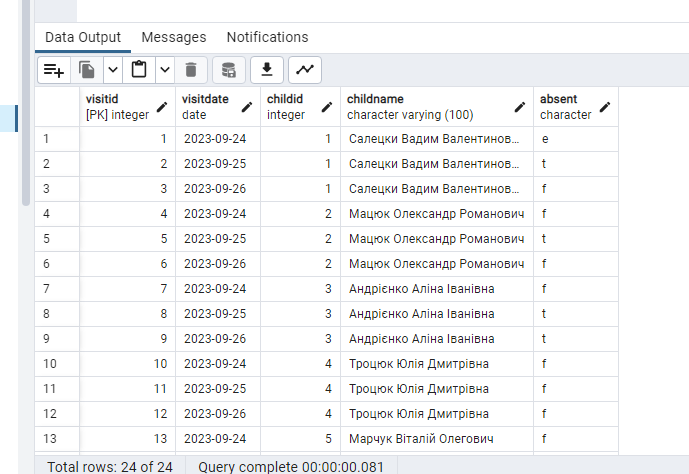
EXECUTE PROCEDURE update\_child\_performance();



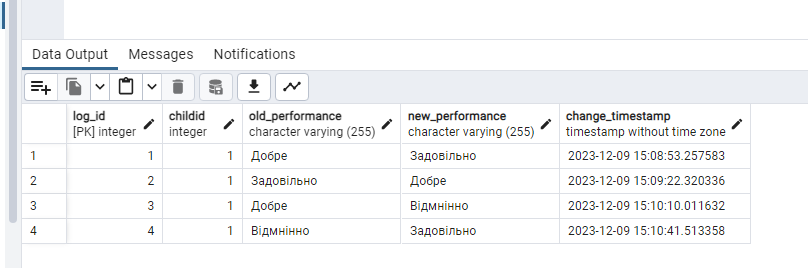
a)



б)



в)



г)

Рисунок 1.4 (а-г) – Результат виконаної роботи

5. Тригерна функція log\_deleted\_children викликається перед видаленням запису з таблиці children і додає відповідний запис в лог deleted\_children\_log.

-- Створення функції для журналювання видалених дочірніх записів

CREATE OR REPLACE FUNCTION log\_deleted\_children()

RETURNS TRIGGER AS $$

BEGIN

-- Перевірка, чи існує старий (видаляється) запис

IF OLD IS NOT NULL THEN

-- Вставка інформації про видалений запис в журнал

INSERT INTO deleted\_children\_log (childid, class, performancelevel, parentcode, deletion\_timestamp)

VALUES (OLD.childid, OLD.class, OLD.performancelevel, OLD.parentcode, NOW());

END IF;

-- Повернення старого запису

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

-- Створення тригера, який викликає функцію при видаленні запису в таблиці "children"

CREATE TRIGGER children\_delete\_trigger

BEFORE DELETE ON children

FOR EACH ROW

EXECUTE PROCEDURE log\_deleted\_children();

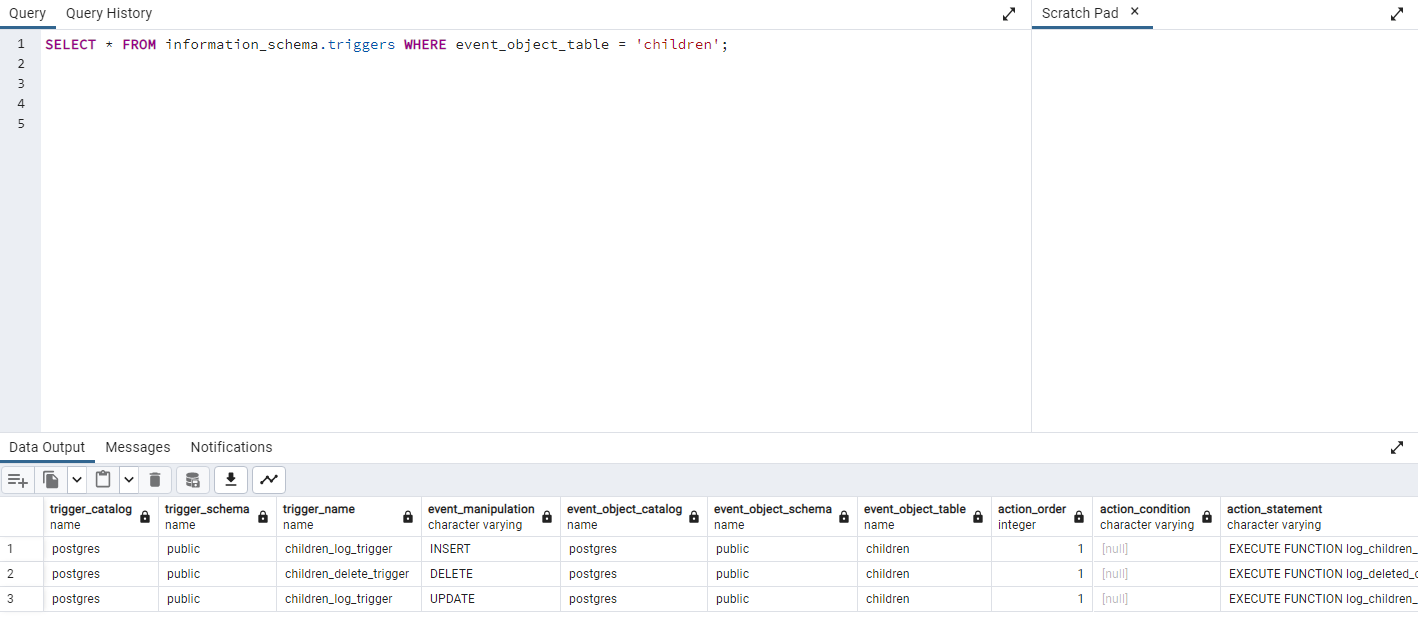


Рисунок 1.5 – Результат виконаної роботи

6. Тригерна функція для логування оновлення рівня успішності учнів у таблиці children:

-- Створення функції для журналювання змін у рівні ефективності дитини

CREATE OR REPLACE FUNCTION log\_child\_performance\_changes()

RETURNS TRIGGER AS $$

BEGIN

-- Перевірка, чи відбулася зміна у рівні ефективності

IF NEW.performancelevel IS DISTINCT FROM OLD.performancelevel THEN

-- Вставка інформації про зміни у журнал

INSERT INTO child\_performance\_log (childid, old\_performance, new\_performance, change\_timestamp)

VALUES (NEW.childid, OLD.performancelevel, NEW.performancelevel, NOW());

END IF;

-- Повернення нового запису

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

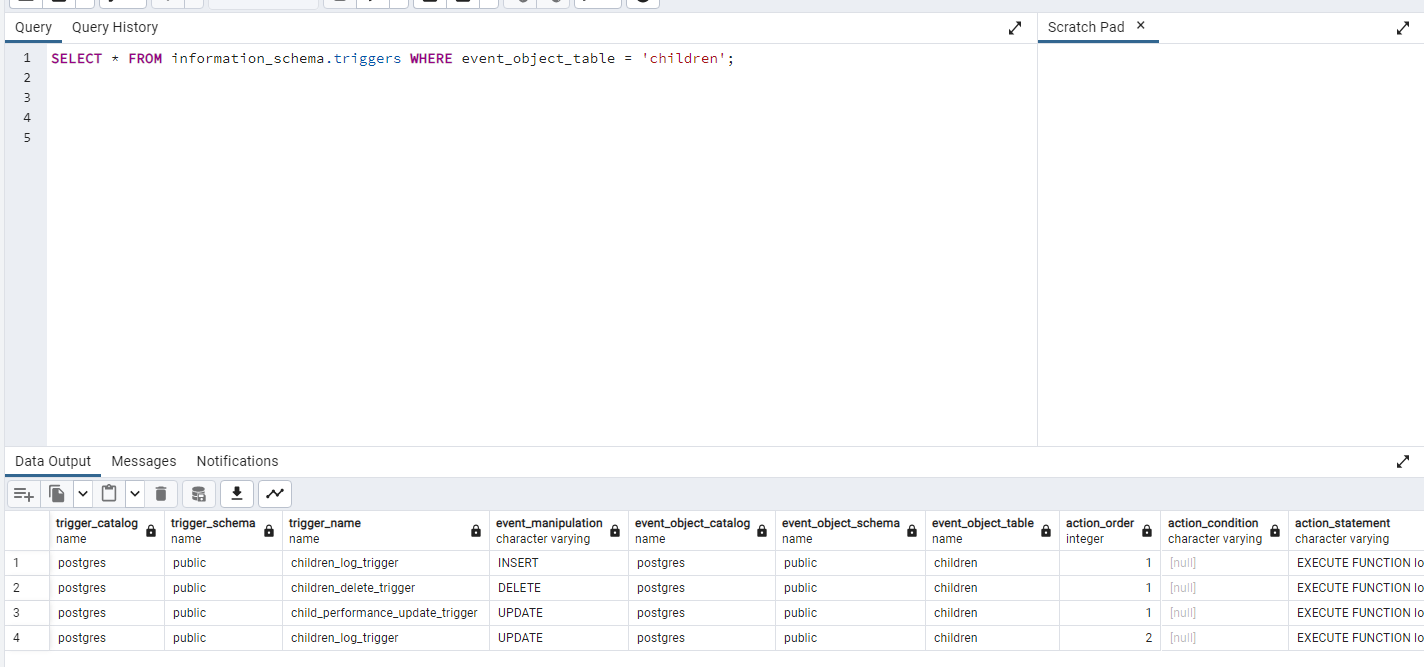
-- Створення тригера, який викликає функцію після оновлення запису в таблиці "children"

CREATE TRIGGER child\_performance\_update\_trigger

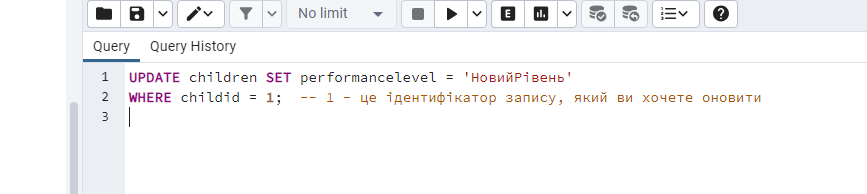
AFTER UPDATE ON children

FOR EACH ROW

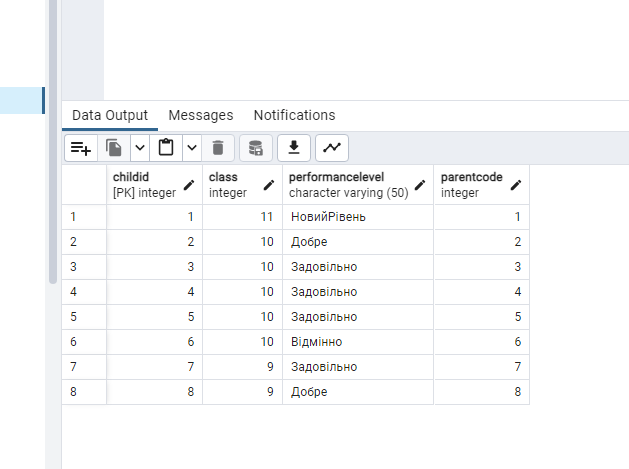
EXECUTE PROCEDURE log\_child\_performance\_changes();



а)



б)



в)

Рисунок 1.6 (а-б) – Результат виконаної роботи

7. Тригер, який буде реагувати на вставку нових записів у таблицю children і автоматично встановлювати рівень успішності (performancelevel) в залежності від класу (class) учнів.

-- Створення тригеру

CREATE OR REPLACE FUNCTION set\_performance\_level()

RETURNS TRIGGER AS $$

BEGIN

-- Перевірка класу та встановлення рівня успішності

IF NEW.class >= 10 THEN

NEW.performancelevel = 'Відмінно';

ELSIF NEW.class >= 8 THEN

NEW.performancelevel = 'Добре';

ELSE

NEW.performancelevel = 'Задовільно';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

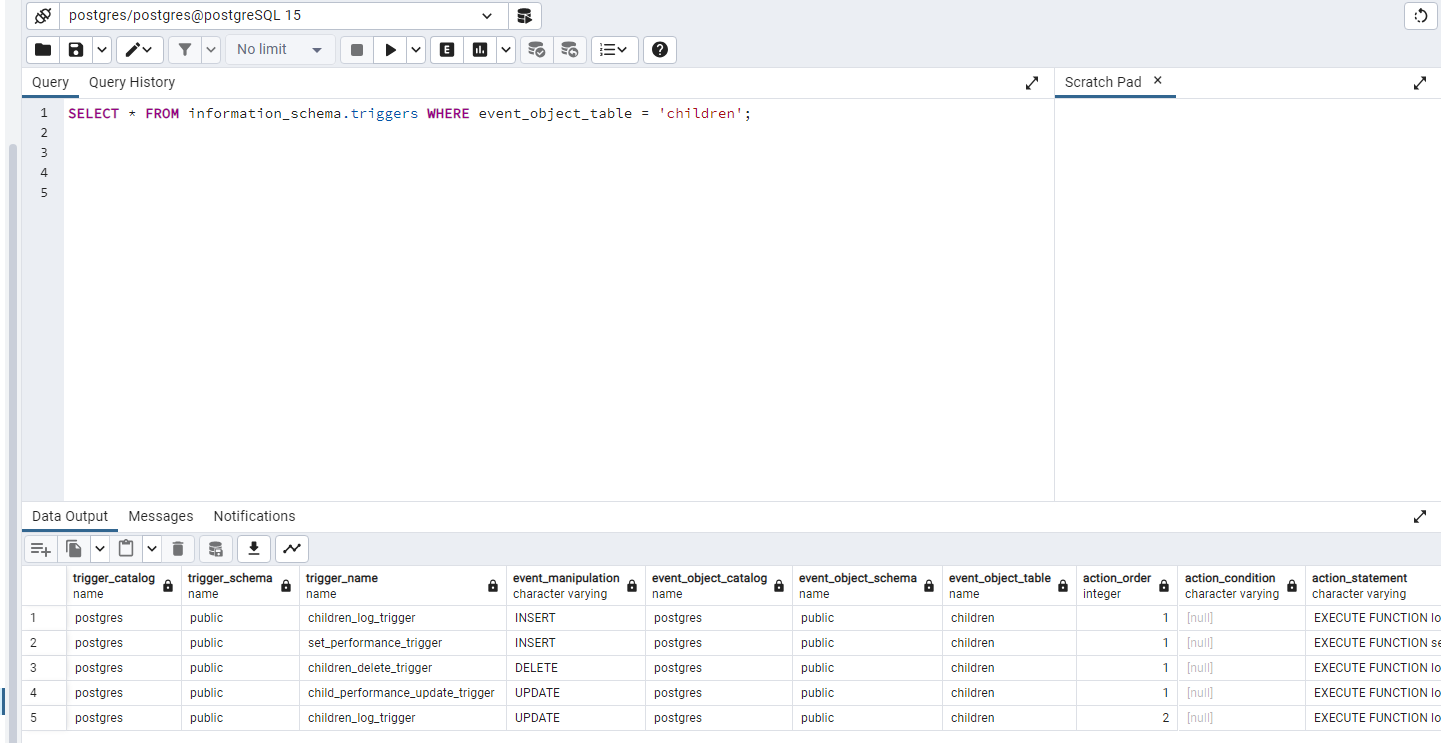
-- Створення тригеру для таблиці children

CREATE TRIGGER set\_performance\_trigger

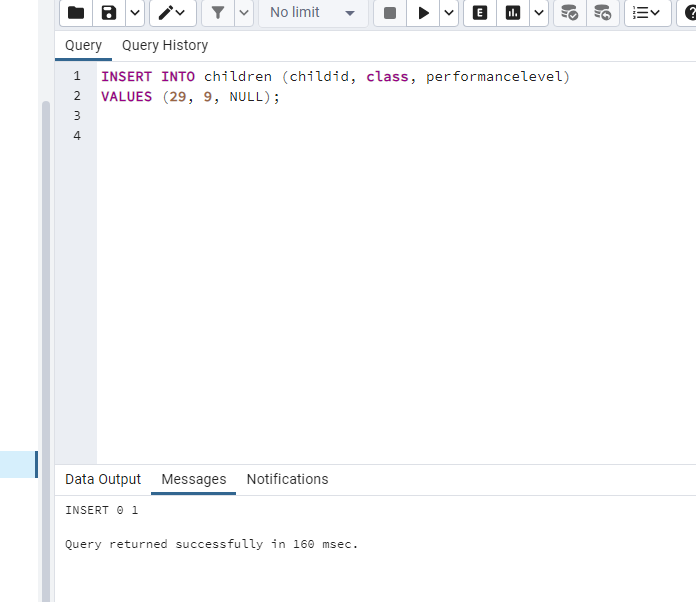
BEFORE INSERT ON children

FOR EACH ROW

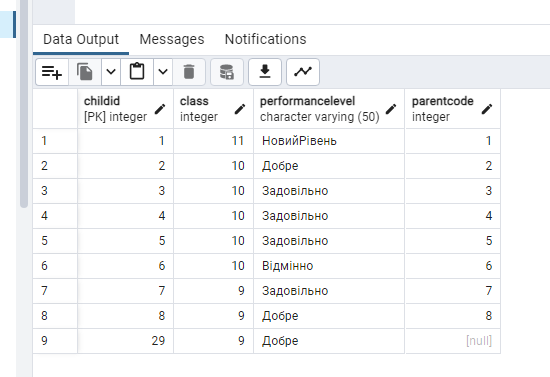
EXECUTE PROCEDURE set\_performance\_level();



а)



б)



в)

Рисунок 1.7(а-в) – Результат виконаної роботи

8. Створиення EVENT TRIGGER на ddl\_command\_start, який вказує про виконання будь-якої DDL команди.

-- Створення тригерної функції

CREATE OR REPLACE FUNCTION notice\_any\_command()

RETURNS event\_trigger

LANGUAGE plpgsql

AS $$

BEGIN

RAISE NOTICE 'команда % спрацювала', tg\_tag;

END;

$$;

-- Тригер, який вказує про виконання будь-якої DDL команди

CREATE EVENT TRIGGER notice\_ddl ON ddl\_command\_start

EXECUTE FUNCTION notice\_any\_command();

-- Перевірка роботи тригеру

CREATE TABLE tab\_1 (id int, name varchar(10));

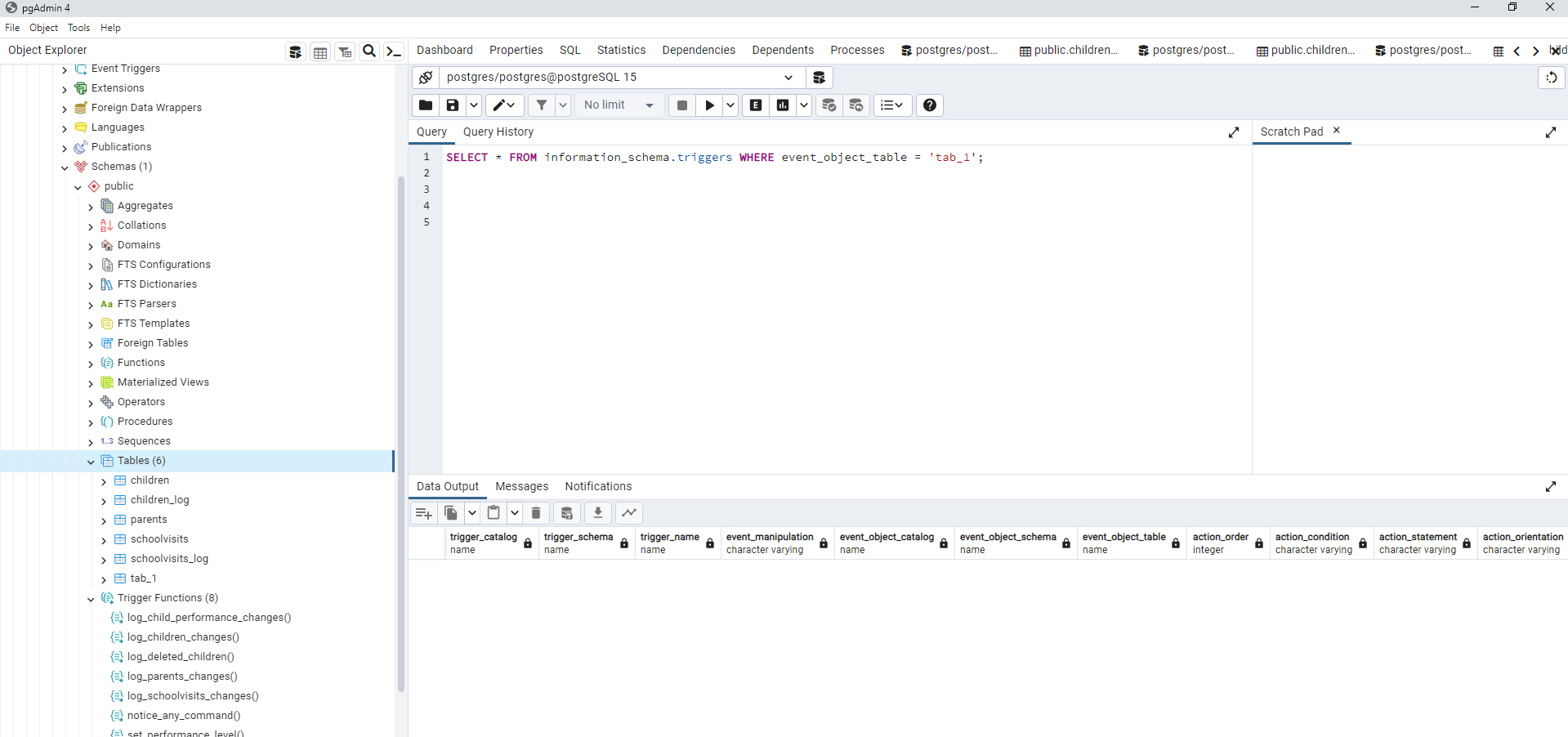


Рисунок 1.8 – Результат виконаної роботи

9. Створення тригеру на подію який видає інформацію про видалений об’єкт БД. EVENT TRIGGER на sql\_drop.

-- Створення тригерної функції

CREATE OR REPLACE FUNCTION

func\_event\_trigger\_drop()

RETURNS event\_trigger AS $$

DECLARE

object record;

BEGIN

FOR object IN SELECT \* FROM

pg\_event\_trigger\_dropped\_objects()

LOOP

RAISE NOTICE '% видалено обєкт - id: %

type: % name: %',

tg\_tag, -- подія

object.objid, -- id об’єкта

object.object\_type, -- тип об’єкта

object.object\_name; -- назва об’єкта

END LOOP;

END;

$$

LANGUAGE plpgsql;

-- Створення тригера sql\_drop

CREATE EVENT TRIGGER event\_trigger\_drop

ON sql\_drop

EXECUTE PROCEDURE func\_event\_trigger\_drop();

-- Перевірка роботи тригеру

DROP TABLE tab\_1;

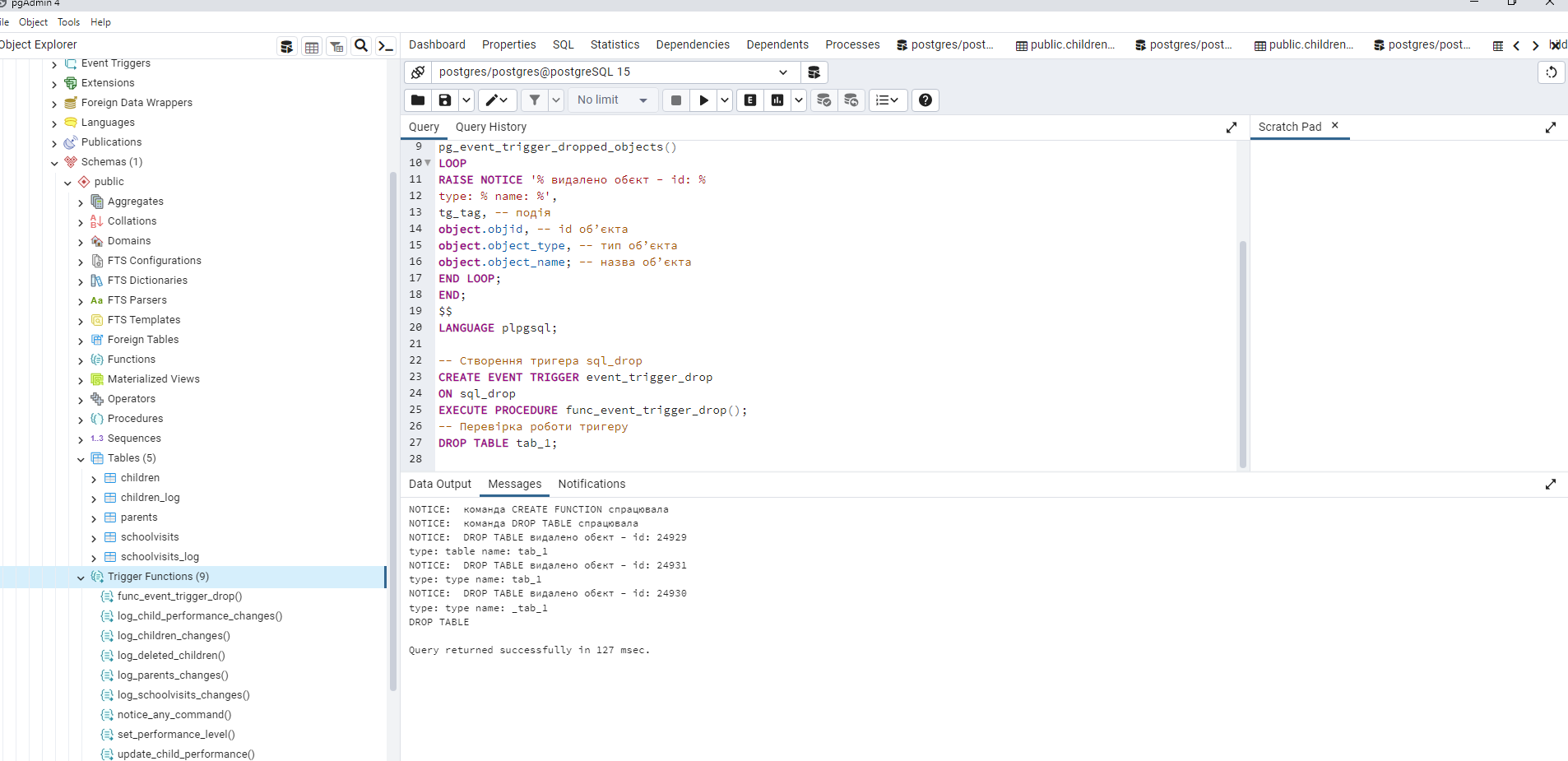


Рисунок 1.9 – Результат виконаної роботи

10. Створення EVENT TRIGGER на ddl\_command\_start із WHEN,який забороняє виконання DDL команди ALTER TABLE.

-- Створення тригерної функції

CREATE OR REPLACE FUNCTION abort\_alter\_table()

RETURNS event\_trigger

LANGUAGE plpgsql

AS $$

BEGIN

RAISE EXCEPTION 'команда % не дозволена', TG\_TAG;

END;

$$;

-- Створення тригера, який забороняє ALTER TABLE

CREATE EVENT TRIGGER abort\_alter\_table\_event\_trigger

ON ddl\_command\_start

WHEN TAG IN ('ALTER TABLE')

EXECUTE FUNCTION abort\_alter\_table();

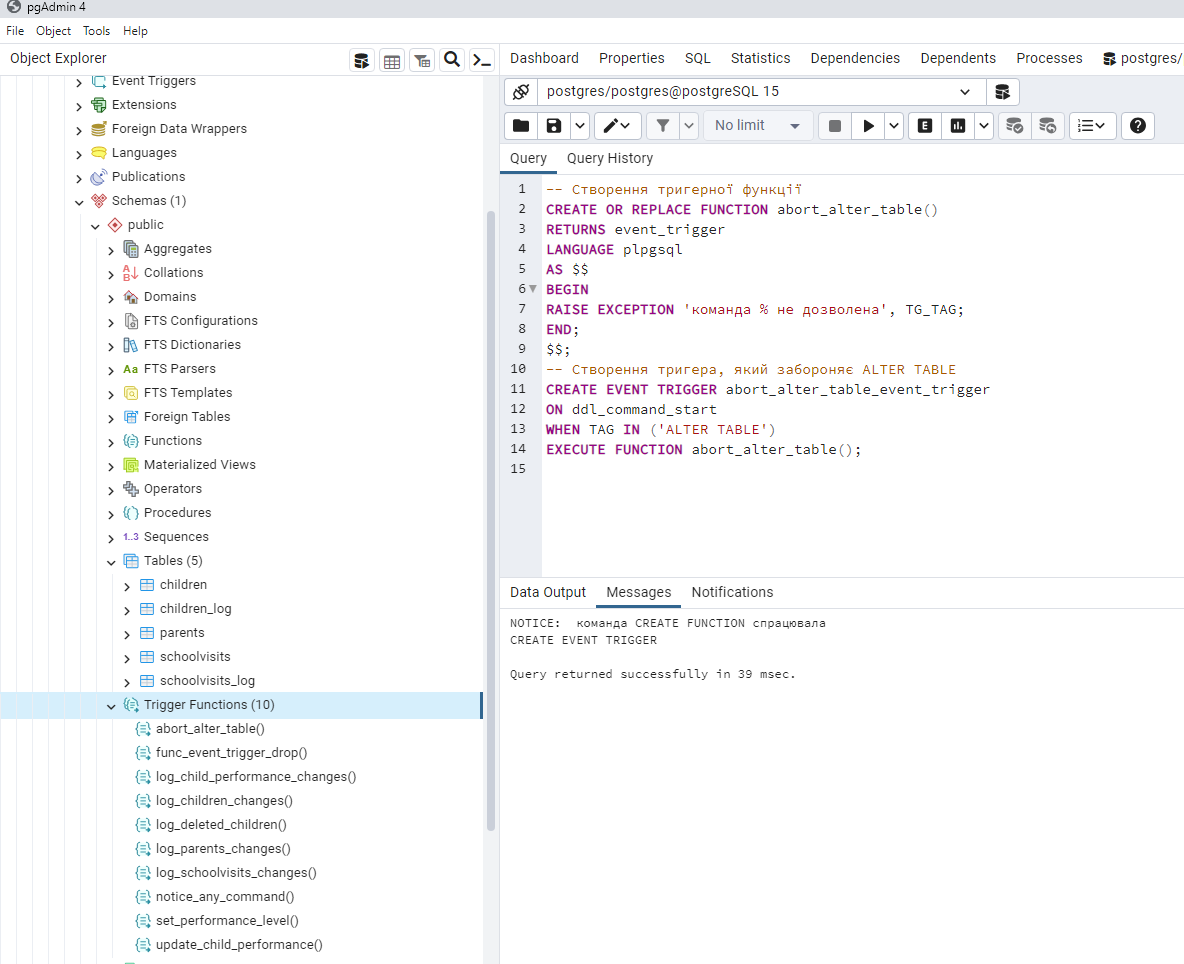


Рисунок 1.9 – Результат виконаної роботи

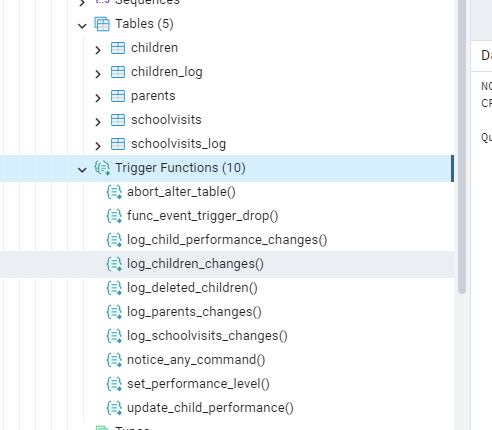


Рисунок 1.10 – Результат виконаної роботи

**Висновок:** під час виконання даної лабораторної роботи я навчився створювати тригерні функції.