

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

# Лабораторна робота №1

***з дисципліни «Бази даних»***

**«Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL»**

Виконав студент групи: КВ-22

ПІБ: Марчук Дмитро Андрійович

Перевірив: Павловський В. І.

**Київ 2024**

**Мета:** здобуття вмінь проектування бази даних та практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

**Завдання:**

1. Розробити модель «сутність-зв’язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі».
2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.
3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ).
4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

# *Завдання №1*

**Опис предметної галузі:**

*Тема обраної бази даних:* Електронний довідник для зберігання технічних характеристик товарів.

## Перелік сутностей з описом їх призначення

Предметна галузь включає в себе 4 сутності , в кожної сутності є атрибут/ти:

1. Category (Category\_id (PK) , Name)
2. Subcategory (Subcategory\_id (PK) , Name , Category\_id (FK))
3. Brand (Brand\_id (PK) , Name , Subcategory\_id (FK))
4. Product (Product\_id (PK) , Name , color, Width, Height, Deepth, Energy Consumption , Weight, Brand\_id (FK))

Сутність Category описує категорії товарів. Наприклад: «Ноутбуки та комп’ютери». Має атрибут «Ім’я».

Сутність Subcategory описує підкатегорію категорії. Наприклад:

Підкатегорія «Монітори» в категорії : «Ноутбуки та комп’ютери». Має атрибути «Ім’я».

Сутність Brand описує бренд/фірму/виробника певного товару. Наприклад бренд «msi». Має атрибути «Ім’я».

Сутність Product описує сам товар. Має атрибути: Ім’я, колір, енергоспоживання, габарити, вага.

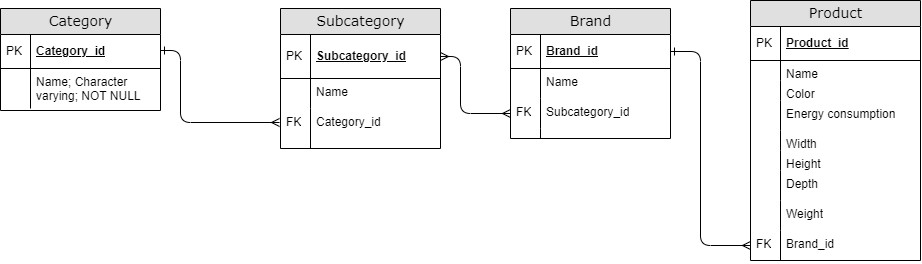
## Зв’язки між сутностями

Зв’язок між Category і Subcategory: В одній категорії може міститися багато підкатегорій, але одна підкатегорія може міститися тільки в одній категорії. 1:N

Зв’язок між Subcategory і Brand: В одній підкатегорії може знаходитися багато різних брендів , але і один і той же бренд може знаходитися в різних підкатегоріях. N:M

Зв’язок між Brand і Product: Бренд може випускати багато продуктів , але у одного продукту може бути тільки один виробник. 1:N

Модель «сутність-зв’язок» для створення електронного довідника для зберігання технічних характеристик товарів.



ER-діаграма побудована за нотацією «Crow`s foot»

# Завдання №2

## Опис процесу перетворення

Сутність Category перетворена в таблицю Category з первинним ключем (ідентифікатором) Category\_id та атрибутом Name.

Сутність Subcategory перетворена в таблицю Subcategory з первинним ключем (ідентифікатором) Subcategory \_id та атрибутом Name.

Сутність Brand перетворена в таблицю Brand з первинним ключем

(ідентифікатором) Brand\_id та атрибутом Name.

Сутність Product перетворена в таблицю Product з первинним ключем (ідентифікатором) Product \_id та атрибутами Name, color, Energy consumption, Height, Width, Deepth, Weight.

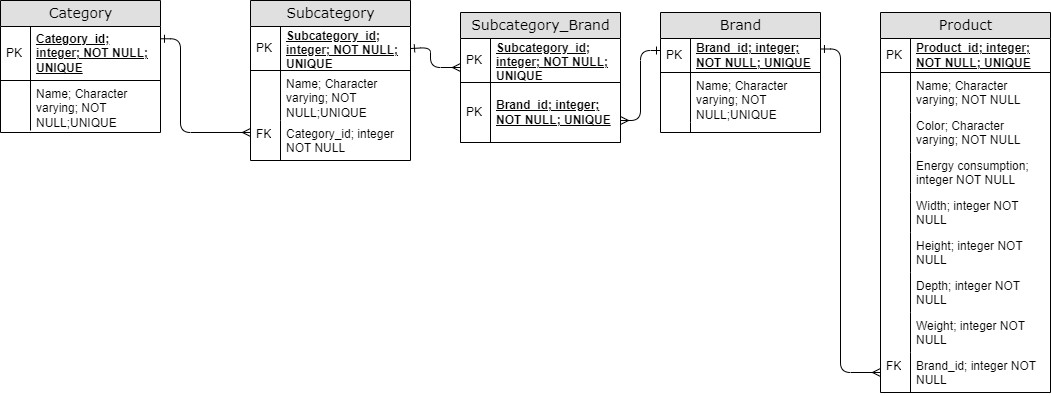
Кожна категорія містить якусь кількість підкатегорій, тому маємо зв’язок 1:N між Category і Category, який зумовив появу зовнішнього ключа FK Category\_id у таблиці Subcategory.

В кожній підкатегорії може міститися велика кількість товарів різних виробників , але і виробники можуть виготовляти товари які відносяться до різних підкатегорій , тому маємо зв’язок N:M між Subcategory і Brand , що зумовлює появу додаткової таблиці Subcategory\_Brand що має 2 зовнішніх ключа для посилання на таблицю Subcategory і Brand.

Кожний бренд може виробляти велику кількість товарів а у товара може бути лише один виробник , тому маємо зв’язок 1:N між Brand і Product, що зумовлює появу зовнішнього ключа FK Brand\_id в табилці Product.

Поля “name” таблиць Category, Category і Brand мають значення UNIQUE тому , що кожна назва категорії та підкатегорії не може повторюватися, так само немає сенсу в повторенні назви брендів. Назви товаір повторюватися можуть.

Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.



# Завдання №3

## Функціональні залежності

**Category** (*Category\_id*, name)

Primary key is Product\_id

Candidate key is name Category\_id → name

Name → Category\_id

**Subcategory** (*Subcategory\_id* , name)

Primary key is Subcategory\_id

Candidate key is name

Subcategory\_id → name

Name → Subcategory\_id

**Brand** (*Brand\_id* , name)

Primary key is Brand\_id

Candidate key is name

Brand\_id → name

Name → Brand\_id

**Product** (*Product\_id* , name , color, Width, Height, Deepth, Energy

Consumption , Weight )

Primary key is Product\_id

Candidate key ----

Product\_id → name

Product\_id → color

Product\_id → Energy Consumption

Product\_id → Width

Product\_id → Height

Product\_id → Deepth

Product\_id → Weight

Name ↮ Product\_id

Name ↮ color

Name ↮ Width

Name ↮ Height

Name ↮ Deepth

Name ↮ Energy Consumption

Name ↮ Weight

Будемо для спрощення вважати, що неможливо однозначно за назвою продукти визначити інші її характеристики , оскільки у однієї компанії можуть бути декілька версій одного і того ж самого продукту (розмір , колір, пам’ять і тд)

**Відповідність схеми нормальним формам**

1. Щоб задовільнити умови 1НФ кожен атрибут в таблиці має бути атомарним, тобто:

* Кожна клітинка містить єдине значення;
* Кожен запис є унікальним.

Таблиці (*Category, Subcategory, Brand, Product*) містять атрибути, які відповідають цим вимогам. Наприклад, кожен продукт має унікальний *Product\_id*, і значення в кожній клітинці таблиць є атомарними (немає масивів чи множин значень).

Дана схема відповідає 1НФ.

1. Щоб схема відповідала 2НФ повинні виконуватись умови:

* Схема перебуває в 1НФ;
* Кожний неключовий атрибут функціонально залежить від цілого ключа.

Схема перебуває в 1НФ. У кожній таблиці є простий первинний ключ (наприклад, *Product\_id* у таблиці *Product*), і всі атрибути залежать від цього ключа. Наприклад, у таблиці *Product*, атрибути *name, color, Energy Consumption, Width, Height, Deepth, Weight* залежать від *Product\_id*, а не від іншого ключа чи атрибута. Таким чином, схема також відповідає **2НФ.**

1. Щоб схема відповідала 3НФ повинні виконуватись умови:

* Схема перебуває в 2НФ;
* Кожен неключовий атрибут функціонально залежить від усього ключа, а не його частини.

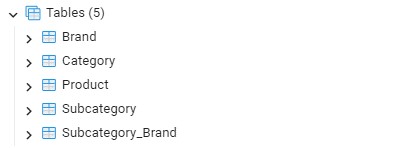
Схема відповідає 2НФ, і жоден з неключових атрибутів не залежить від інших неключових атрибутів. Це означає, що немає транзитивних залежностей, наприклад, у таблиці *Product*, атрибути *color, Width, Height, Energy Consumption, Weight* залежать безпосередньо від *Product\_id*, а не від інших атрибутів, таких як name чи color.

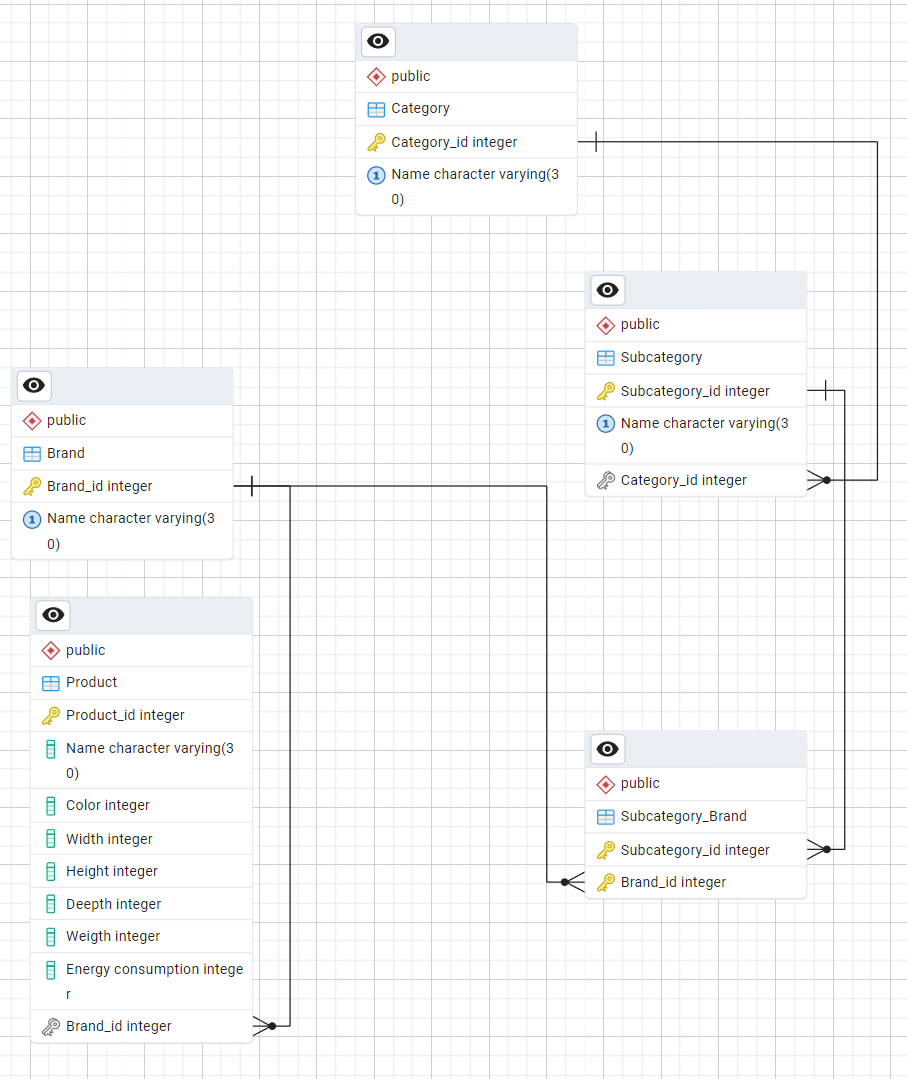
Висновок:

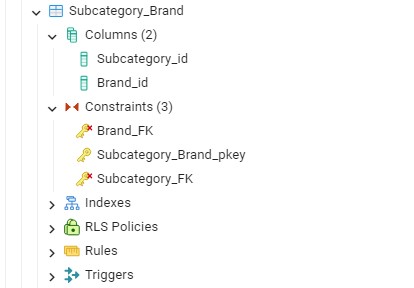
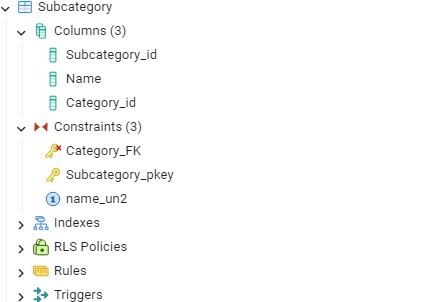
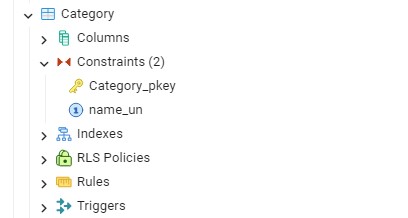
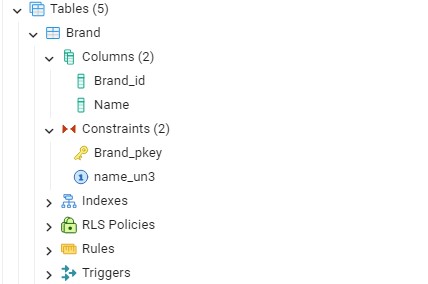
Схема бази даних відповідає нормальним формам 1НФ, 2НФ та 3НФ. Вона добре структурована і нормалізована, що сприяє ефективному та надійному зберіганню та обробці даних. Використання нормалізації допомагає уникнути аномалій вставки, оновлення та видалення, а також забезпечує цілісність даних у базі.

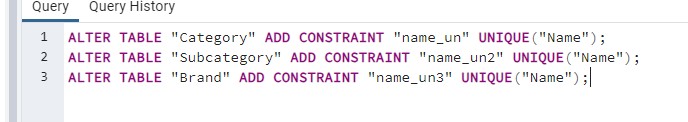
# Завдання №4

Ознайомлення із інструментарієм PostgreSQL та pgAdmin 4 та внесення даних у кожну з таблиць засобами pgAdmin 4.

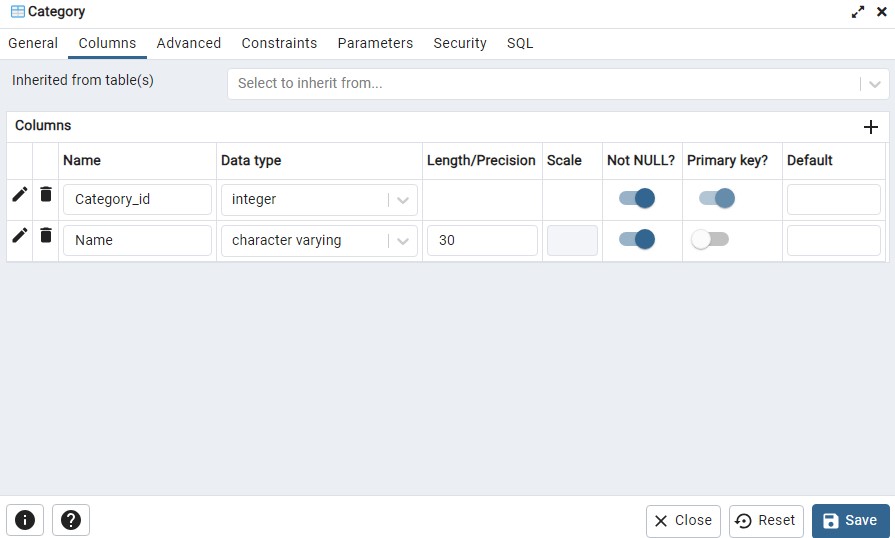
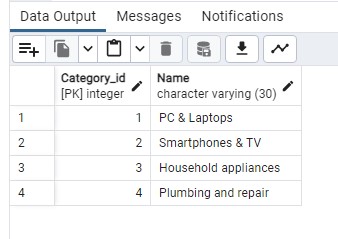




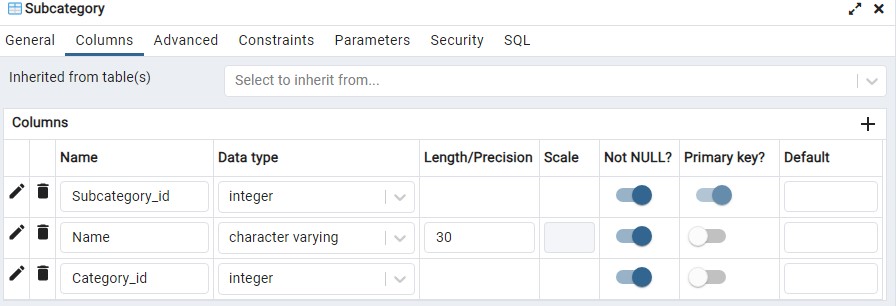


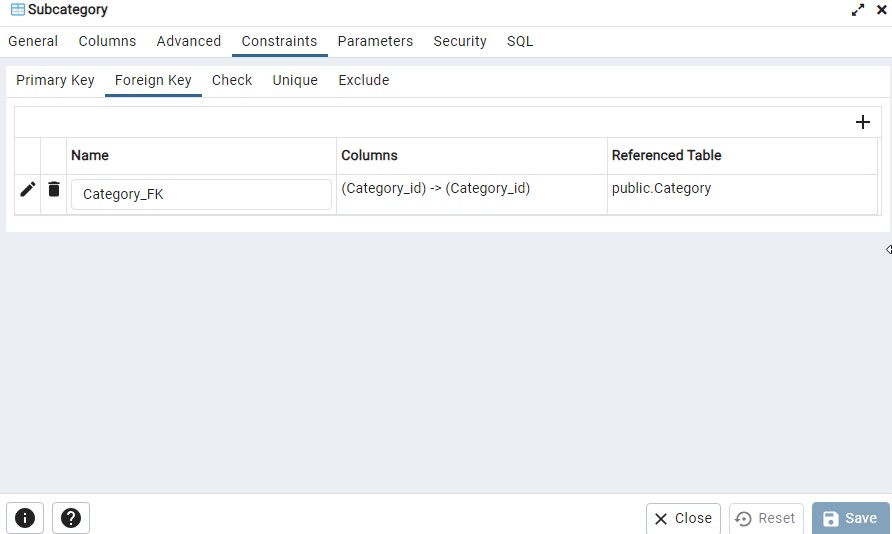


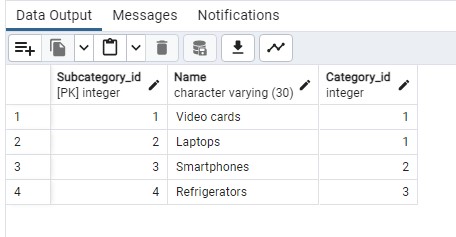
## Category



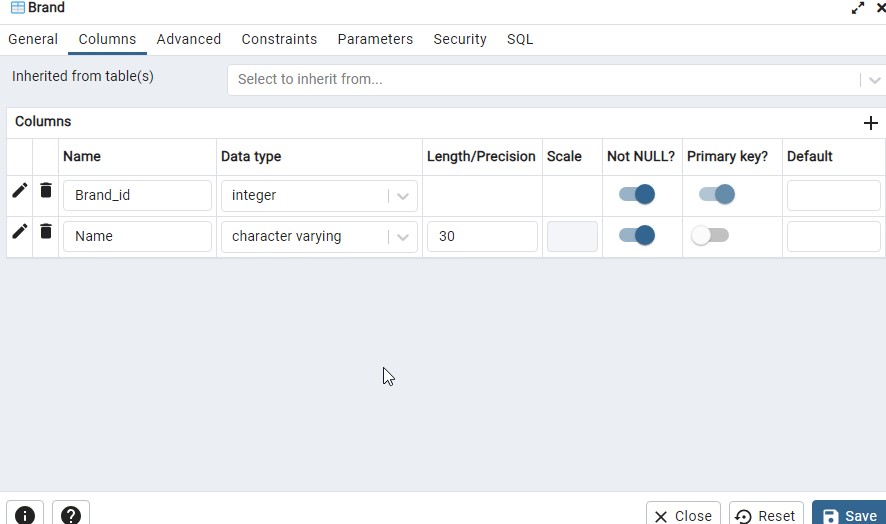
## Subcategory

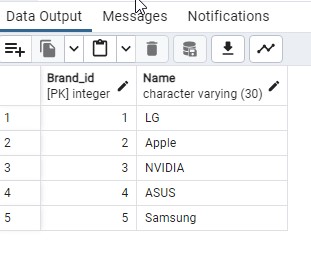




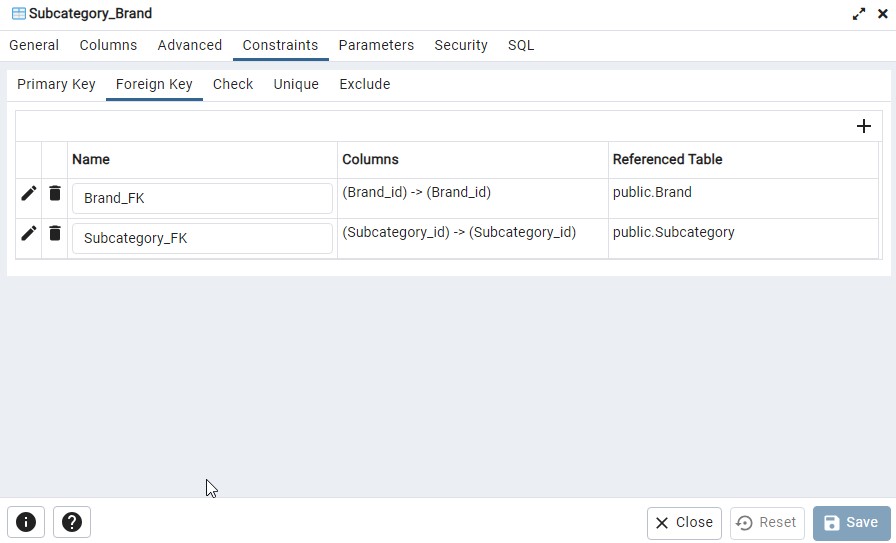
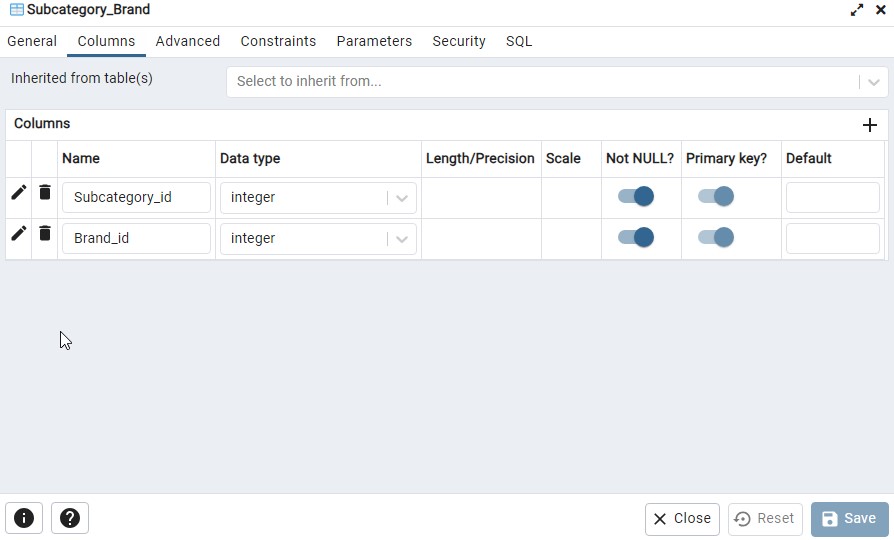


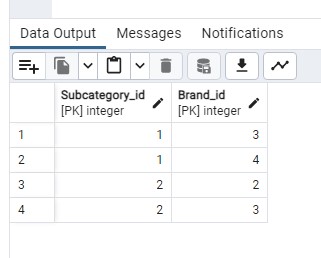
## Brand



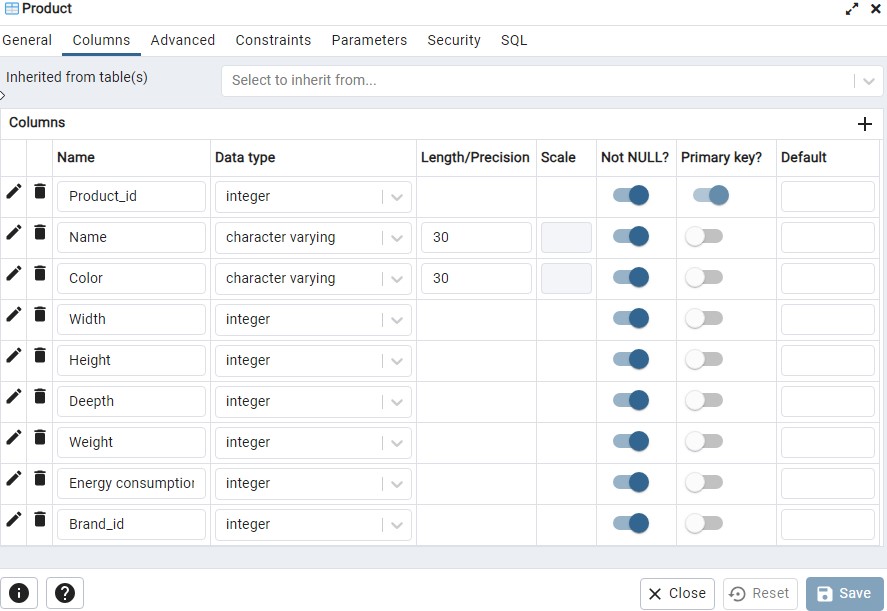


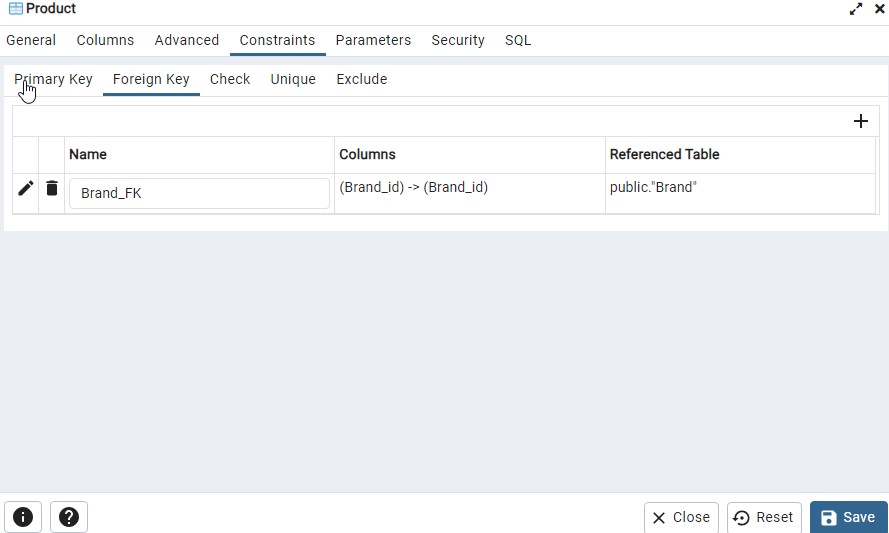
## Subcategory\_Brand

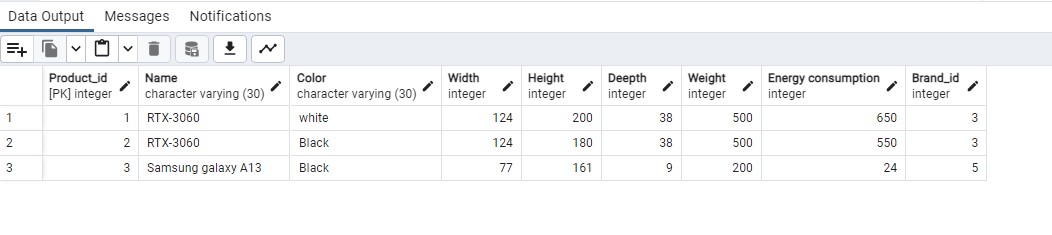




## Product







**Таблиці в коді SQL**

BEGIN;

CREATE TABLE IF NOT EXISTS public."Brand"

(

"Brand\_id" integer NOT NULL,

"Name" character varying(30) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Brand\_pkey" PRIMARY KEY ("Brand\_id"),

CONSTRAINT name\_un3 UNIQUE ("Name")

);

CREATE TABLE IF NOT EXISTS public."Category"

(

"Category\_id" integer NOT NULL,

"Name" character varying(30) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT "Category\_pkey" PRIMARY KEY ("Category\_id"),

CONSTRAINT name\_un UNIQUE ("Name")

);

CREATE TABLE IF NOT EXISTS public."Product"

(

"Product\_id" integer NOT NULL,

"Name" character varying(30) COLLATE pg\_catalog."default" NOT NULL,

"Color" integer NOT NULL,

"Width" integer NOT NULL,

"Height" integer NOT NULL,

"Deepth" integer NOT NULL,

"Weigth" integer NOT NULL,

"Energy consumption" integer NOT NULL,

"Brand\_id" integer NOT NULL,

CONSTRAINT "Product\_pkey" PRIMARY KEY ("Product\_id")

);

CREATE TABLE IF NOT EXISTS public."Subcategory"

(

"Subcategory\_id" integer NOT NULL,

"Name" character varying(30) COLLATE pg\_catalog."default" NOT NULL,

"Category\_id" integer NOT NULL,

CONSTRAINT "Subcategory\_pkey" PRIMARY KEY ("Subcategory\_id"),

CONSTRAINT name\_un2 UNIQUE ("Name")

);

CREATE TABLE IF NOT EXISTS public."Subcategory\_Brand"

(

"Subcategory\_id" integer NOT NULL,

"Brand\_id" integer NOT NULL,

CONSTRAINT "Subcategory\_Brand\_pkey" PRIMARY KEY ("Subcategory\_id", "Brand\_id")

);

ALTER TABLE IF EXISTS public."Product"

ADD CONSTRAINT "Brand\_FK" FOREIGN KEY ("Brand\_id")

REFERENCES public."Brand" ("Brand\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Subcategory"

ADD CONSTRAINT "Category\_FK" FOREIGN KEY ("Category\_id")

REFERENCES public."Category" ("Category\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Subcategory\_Brand"

ADD CONSTRAINT "Brand\_FK" FOREIGN KEY ("Brand\_id")

REFERENCES public."Brand" ("Brand\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."Subcategory\_Brand"

ADD CONSTRAINT "Subcategory\_FK" FOREIGN KEY ("Subcategory\_id")

REFERENCES public."Subcategory" ("Subcategory\_id") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

END;

*Посилання на GitHub:*

<https://github.com/Devded86/DB_KPI.git>