

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-33

ПІБ: Малойван В. Р.

Перевірив: Павловський В. І.

**Київ 2025**

**Постановка задачі:**

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 бази даних: Мобільний додаток для фітнес-трекінгу та здоров’я.

Коротка характеристика: Додаток призначений для відстеження фізичної активності користувачів, контролю стану здоров’я, а також для взаємодії з іншими користувачами у соціальному форматі. Система дозволяє реєструвати тренування, зберігати дані про вправи та показники здоров’я, підтримувати соціальні зв’язки.

**Розробка концептуальної моделі**

**При створенні даної бази даних виділено такі сутності:**

1. **Користувач (User)** – представляє клієнтів мобільного додатку:

* Атрибути: ім'я, прізвище, електронна пошта, дата реєстрації, телефон.

1. **Тренування (Workout)** – представляє заняття спортом, які виконує користувач:

* Атрибути: тип тренування (біг, силові вправи, йога тощо), дата, час.

1. **Здоров’я (Health Metrics)** –представляє дані про фізичний стан користувача:

* Атрибути: кількість кроків, пульс, витрачені калорії, дата вимірювання.

**Зв’язки:**

Зв’язок «Користувач» - «Тренування»:

* Тип зв'язку: 1 до N (один користувач може мати багато тренувань; одне тренування належить лише одному користувачу).

Зв’язок «Користувач» - «Здоров’я»:

* Тип зв'язку: 1 до N (один користувач може мати багато записів стану здоров’я; один запис належить лише одному користувачу).

Зв’язок «Тренування» - «Здоров’я»:

* Тип зв'язку: 1 до N (один користувач може мати багато друзів, але кожен запис дружби відноситься лише до одного користувача — тобто «один користувач → багато записів дружби»).

Зв’язок «Користувач» - «Користувач»:

* Тип зв'язку: 1 до N (один користувач може мати багато друзів, і кожен інший користувач також може мати багато зв’язків дружби).
* Атрибут зв’язку: статус (підтверджено/очікує), дата (дата, коли стали друзями).

Графічне подання концептуальної моделі «Сутність-зв’язок» зображено на рисунку 1

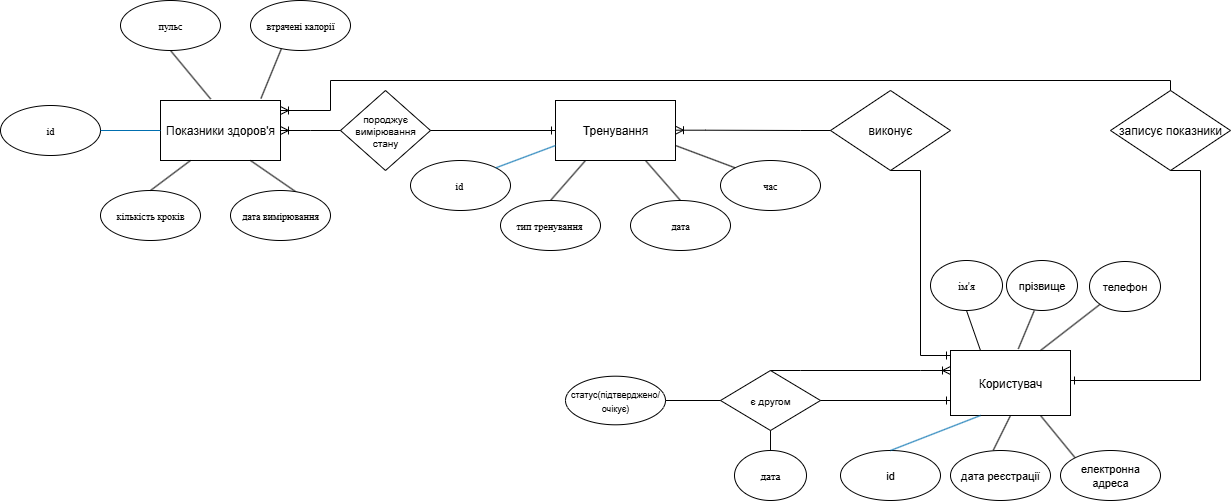


Рисунок 1 – Графічне подання концептуальної моделі «Сутність-зв’язок»

***Завдання №2:***

**Перетворення концептуальної моделі у логічну модель та схему бази даних**

Сутність “Користувач” перетворено в таблицю User з первинним ключем user\_id.

Атрибути таблиці: first\_name (ім’я), last\_name (прізвище), email (електронна пошта), phone (номер телефону), date\_registration (дата реєстрації користувача).

Сутність “Тренування” перетворено в таблицю Workout з первинним ключем workout\_id.

Атрибути таблиці: type\_workout (тип тренування), date (дата проведення), time (час проведення).

У таблиці визначено зовнішній ключ user\_id, який посилається на User (user\_id) і відображає належність кожного тренування конкретному користувачу.

Сутність “Здоров’я” перетворено в таблицю Health Metrics з первинним ключем metrics\_id.

Атрибути таблиці: steps (кількість кроків), pulse (пульс), calories (витрачені калорії), measurement\_date (дата вимірювання).

У таблиці визначено два зовнішні ключі:

• user\_id, який посилається на User (user\_id) і забезпечує належність показників конкретному користувачу;

• workout\_id, який посилається на Workout (workout\_id) та дозволяє фіксувати показники здоров’я у контексті певного тренування.

Зв’язок “Дружба” реалізовано через таблицю Friendship, яка моделює соціальні відносини між користувачами.

У таблиці використовується складений первинний ключ (user\_id1, user\_id2), що ідентифікує пару користувачів.

Додаткові атрибути: status (стан дружби: підтверджено/очікує), date (дата встановлення дружби).

Обидва зовнішні ключі user\_id1 та user\_id2 посилаються на User (user\_id), забезпечуючи унарний зв’язок “Користувач ↔ Користувач”.

Графічне подання логічної моделі «Сутність-зв’язок» зображено на рисунку 2

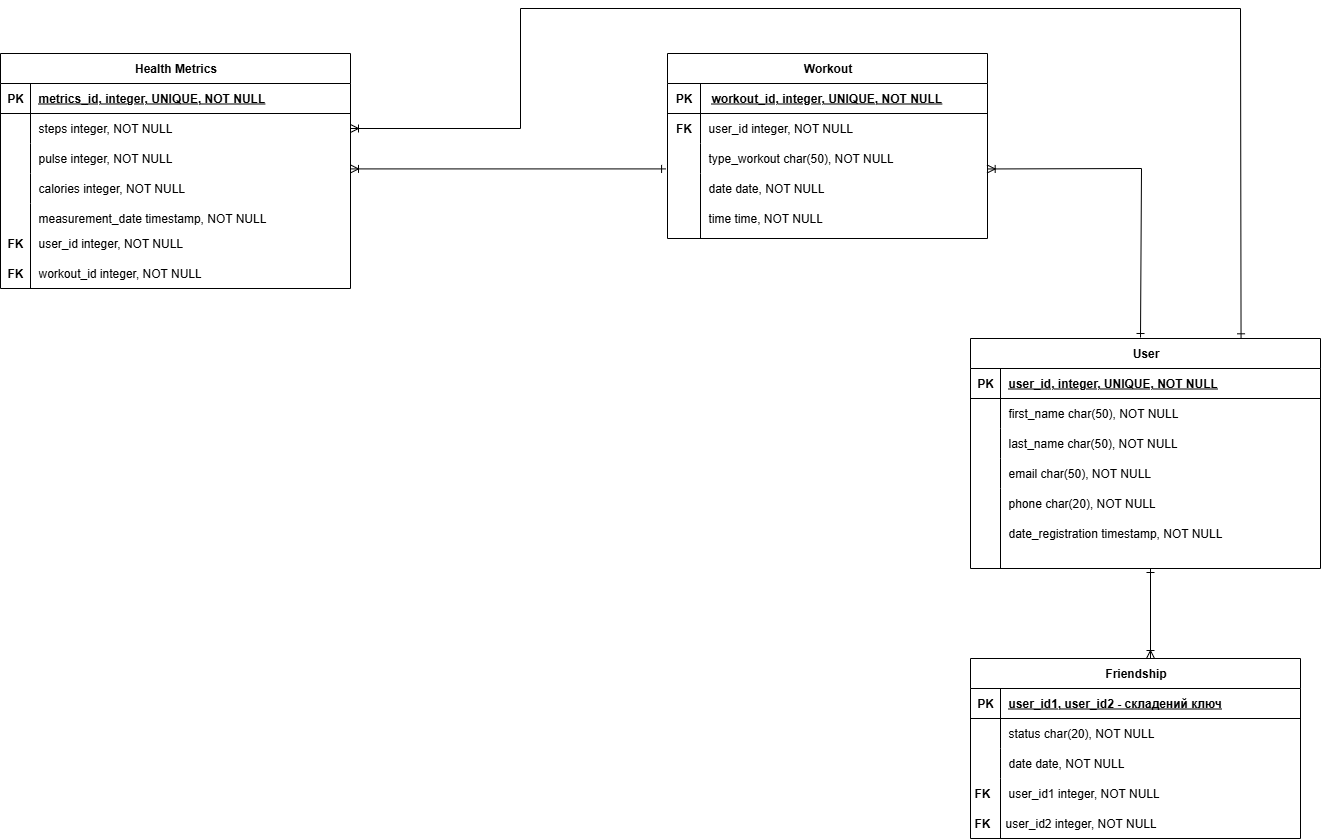


Рисунок 2 – Графічне подання логічної моделі «Сутність-зв’язок»

Таблиця 1 ілюструє детальний перехід від однієї моделі до іншої.

Таблиця 1 – Опис об’єктів бази даних.

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| User – містить дані про користувача | *user\_id* – унікальний ідентифікатор користувача | integer (числовий) |
| *first\_name* – ім’я користувача | character varying (рядок) |
| *last\_name* – прізвище користувача | character varying (рядок) |
| *email –* електронна пошта користувача | character varying (рядок) |
| *phone –* номер телефону користувача | character varying (рядок) |
| *date\_registration –* дата реєстрації користувача | timestamp (дата та час) |
| Workout– містить дані про тренування | *workout\_id* – унікальний ідентифікатор тренування | integer (числовий) |
| *type\_workout* – тип тренування | character varying (рядок) |
| *time* – час проведення тренування | time (час) |
| *date –* дата проведення тренування | date (дата) |
| *user\_id* – ідентифікатор користувача | integer (зовнішній ключ до Користувача) |
| Health Metrics– містить дані про фізичний стан користувача | *metrics\_id* – унікальний ідентифікатор вимірювання | integer (числовий) |
| steps – кількість кроків | integer (числовий) |
| *pulse* – пульс користувача | integer (числовий) |
| *calories* – витрачені калорії | integer (числовий) |
| *measurement\_date* – дата вимірювання | timestamp (дата та час) |
| *user\_id* – ідентифікатор користувача | integer (зовнішній ключ до Користувача) |
| *workout\_id* – ідентифікатор тренування | integer (зовнішній ключ до Тренування) |

|  |  |  |
| --- | --- | --- |
| Зв’язок | Атрибут | Тип атрибуту |
| Friendship – містить дані про дружбу | *user\_id1, user\_id2* – унікальний ідентифікатор дружби(складений первинний ключ) | integer (числовий) |
| *user\_id1* – ідентифікатор першого користувача | integer (зовнішній ключ до Користувача) |
| *user\_id2* – ідентифікатор другого користувача | integer (зовнішній ключ до Користувача) |
| *date –* дата(дата, коли стали друзями) | date (дата) |
| *status* – статус дружби (підтверджено/очікує) | character varying (рядок) |

**Функціональні залежності для кожної таблиці**

1. User:

*user\_id* → first\_name, last\_name, email, phone, date\_registration

2. Workout:

*workout\_id* → workout\_id, user\_id, type\_workout, date, time

3. Health Metrics:

*metrics\_id* → steps, pulse, calories, measurement\_date, user\_id, workout\_id

4. Friendship:

*user\_id1, user\_id2* → status, date, user\_id1, user\_id2

Функціональні залежності відображають співвідношення між ключовими та неключовими атрибутами в таблицях бази даних. Вони визначають, які саме поля однозначно встановлюються на основі значень ключа. Правильне визначення таких залежностей є необхідною умовою для нормалізації схеми, оскільки дозволяє уникнути дублювання даних, зменшує ризик аномалій під час оновлення та забезпечує цілісність інформації.

Транзитивна залежність виникає у випадку, коли один атрибут визначає інший не безпосередньо, а через проміжний атрибут (наприклад, якщо A → B і B → C, то маємо A → C).

У розробленій моделі всі атрибути напряму залежать від своїх ключів:

• у User *–* від user\_id;

• у Workout *–* від workout\_id;

• у Health Metrics *–* від metrics\_id;

• у Friendship *–* від складеної комбінації (user\_id1, user\_id2).

Жодного випадку транзитивної залежності немає. Це означає, що структура не створює надмірності даних і відповідає вимогам нормалізації.

**Відповідність нормальним формам**

*Перша нормальна форма (1НФ)*

Усі атрибути мають атомарний характер (одне поле *–* одне значення). Кожен рядок унікальний завдяки первинним ключам (user\_id, workout\_id, metrics\_id, або складеному ключу (user\_id1, user\_id2)).

Отже, схема задовольняє умови 1НФ.

*Друга нормальна форма (2НФ)*

Модель перебуває в 1НФ. Усі неключові атрибути повністю залежать від первинних ключів:

• у таблицях User, Workout і Health Metrics кожен атрибут визначається відповідним ключем повністю;

• у таблиці Friendship атрибути status і date визначаються лише повною комбінацією (user\_id1, user\_id2). Часткових залежностей немає.

Таким чином, схема відповідає умовам 2НФ.

*Третя нормальна форма (3НФ)*

Модель перебуває в 2НФ. Жоден з неключових атрибутів не визначає інший, а всі вони безпосередньо залежать від первинного ключа своєї таблиці. Транзитивних залежностей не виявлено.

Отже, схема відповідає умовам 3НФ.

**Висновок**

Розроблена модель бази даних відповідає вимогам 1НФ, 2НФ та 3НФ.

• Вона забезпечує атомарність і унікальність записів (1НФ).

• Виключає часткові залежності та повністю підпорядковує атрибути первинним ключам (2НФ).

• Не містить транзитивних залежностей, що гарантує відсутність надмірності даних (3НФ).

Така структура є академічно правильною, логічно узгодженою та стійкою до аномалій при вставці, оновленні й видаленні даних. Вона придатна до масштабування та подальшого розвитку без ризику втрати цілісності.

Схема бази даних у pgAdmin 4 зображено на рисунку 3.

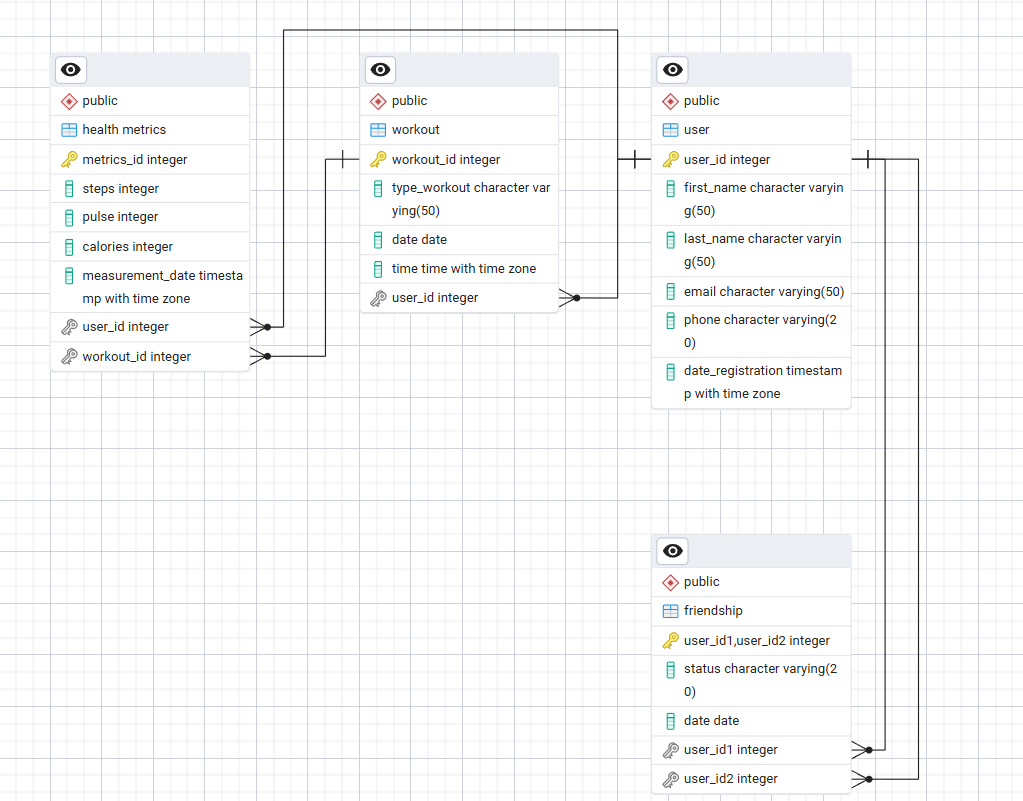
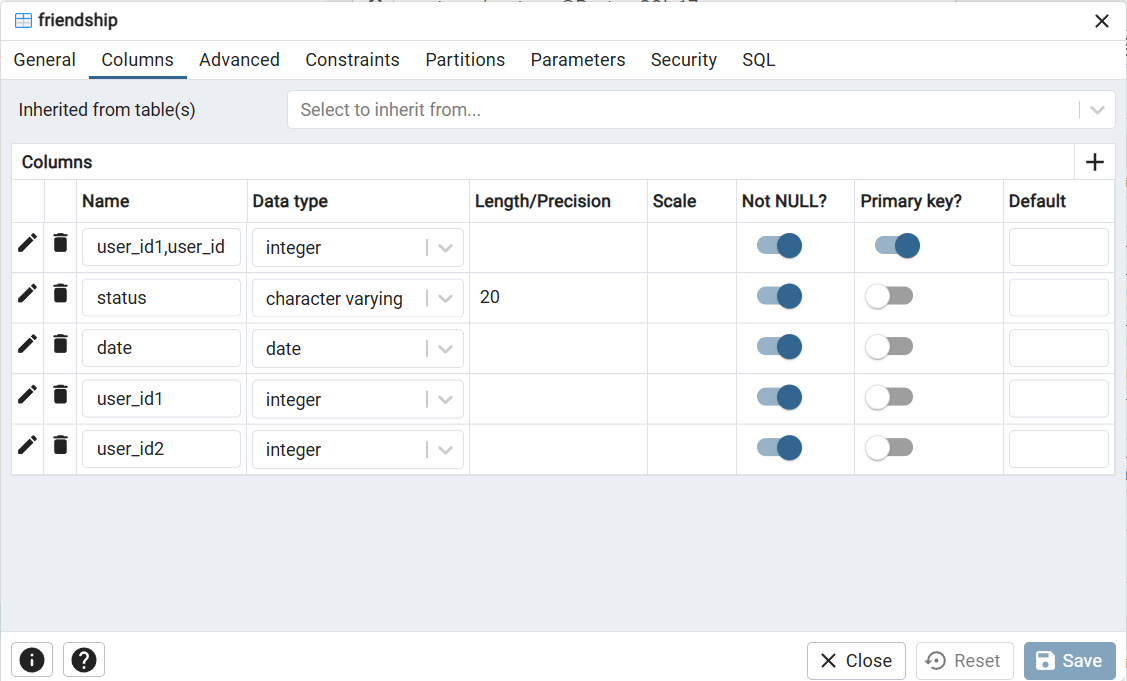


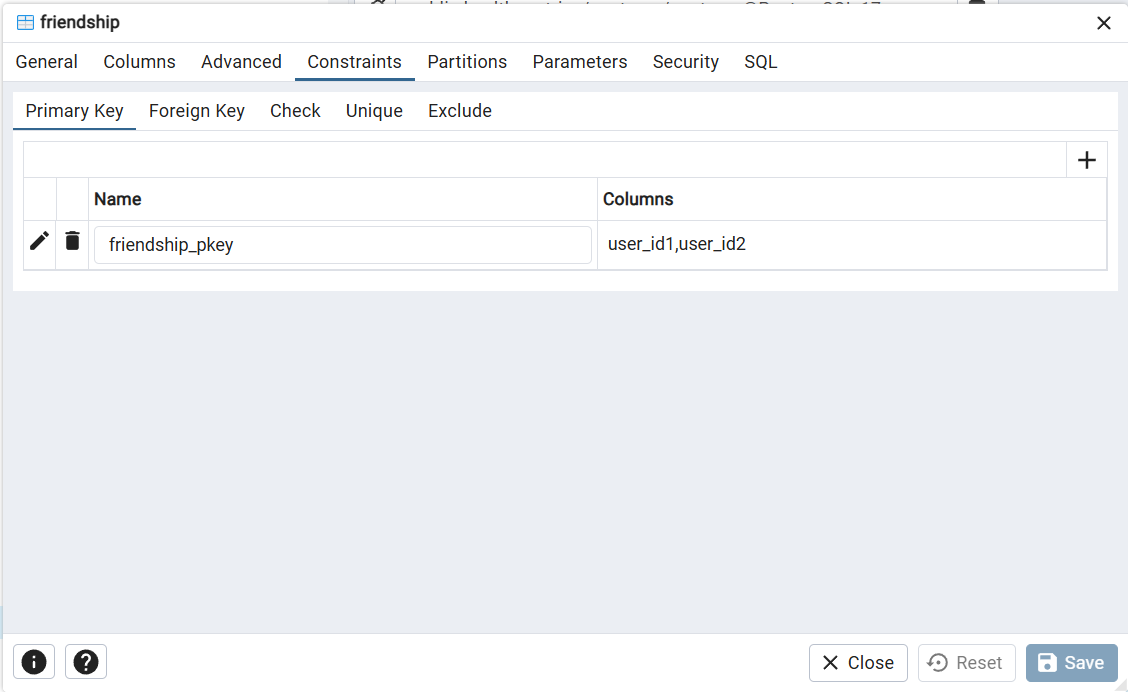
Рисунок 3 *–* Схема бази даних у pgAdmin 4

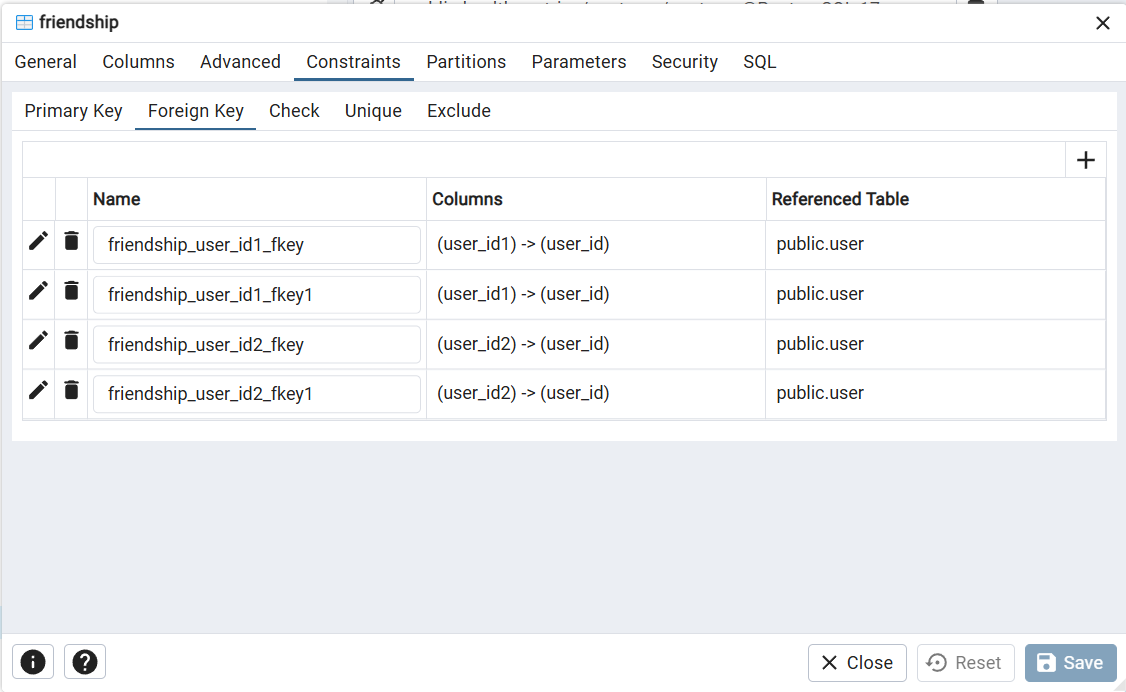
**Таблиці бази даних у pgAdmin4**

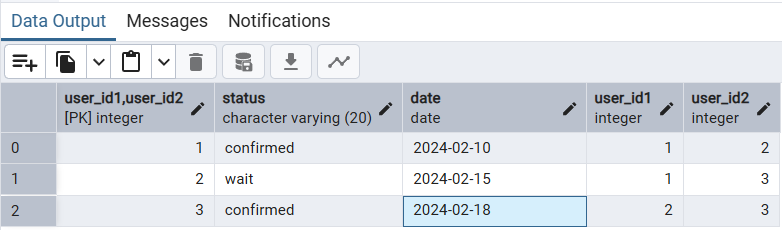


Friendship

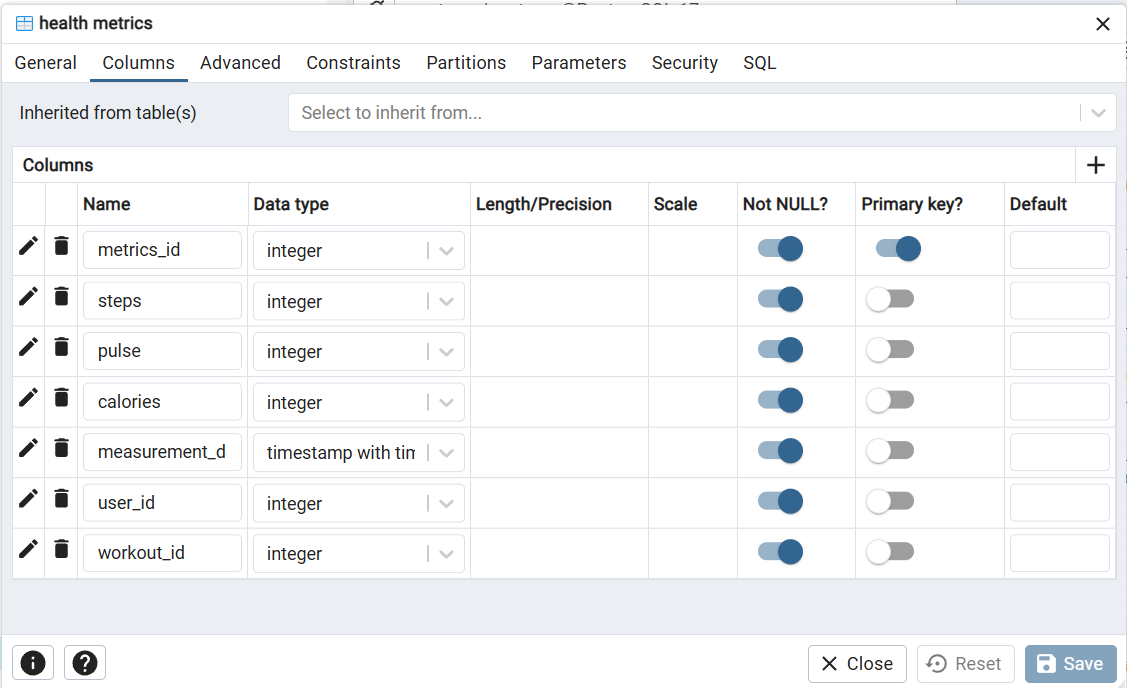


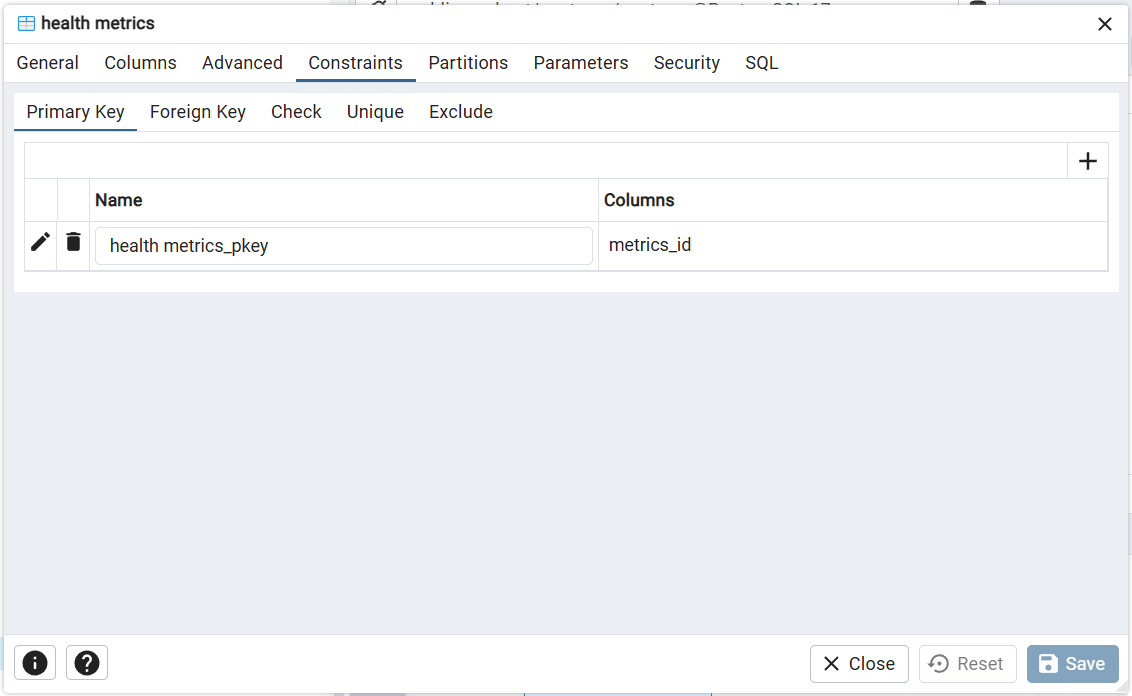


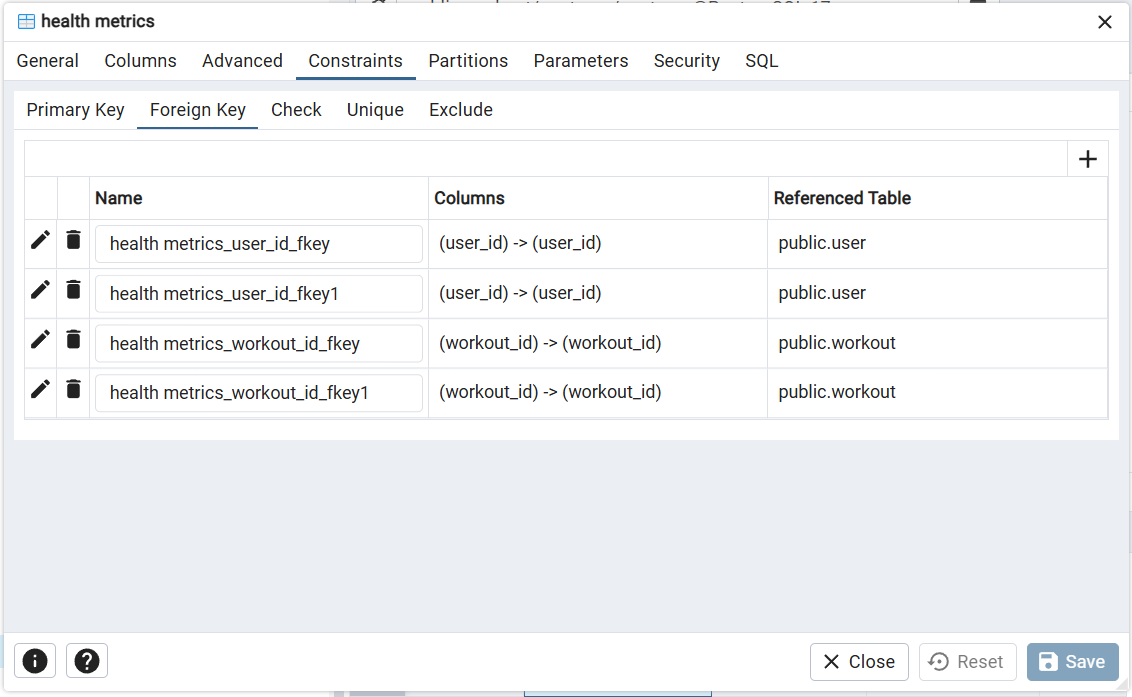


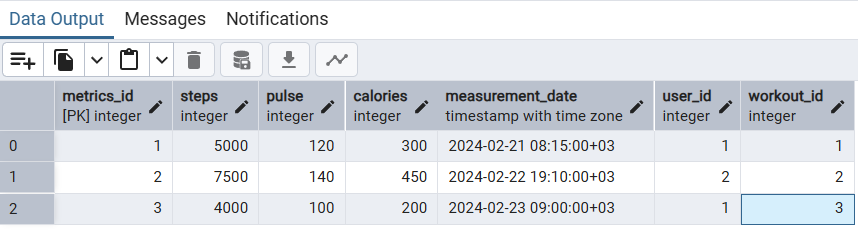


Healthmetrics

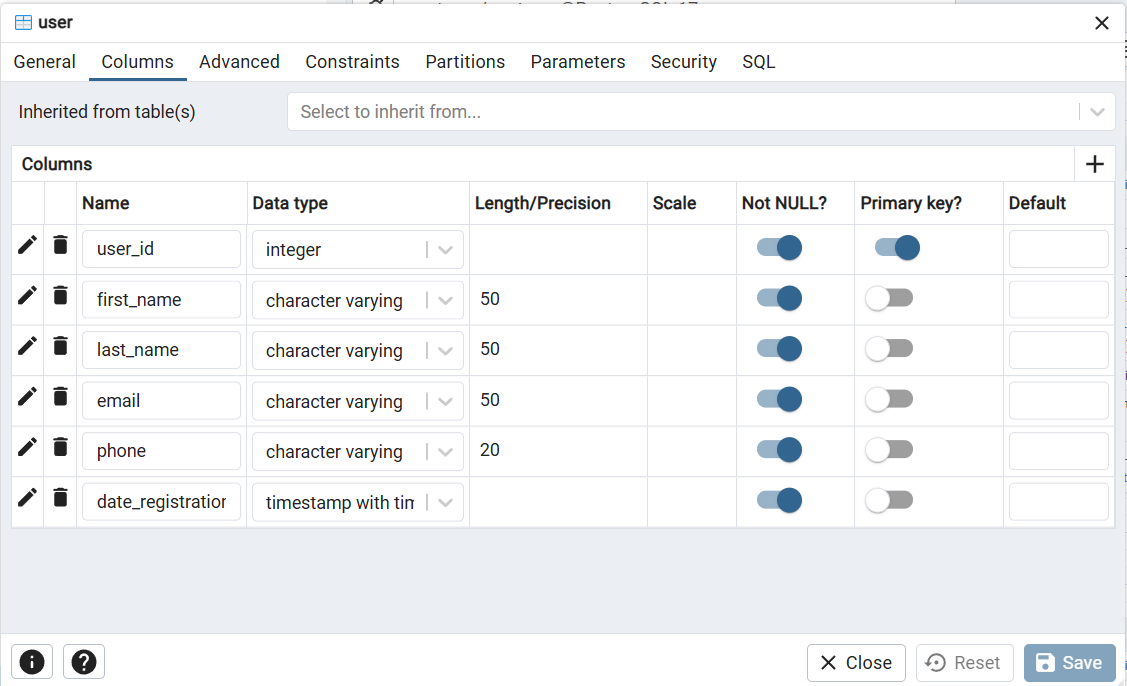


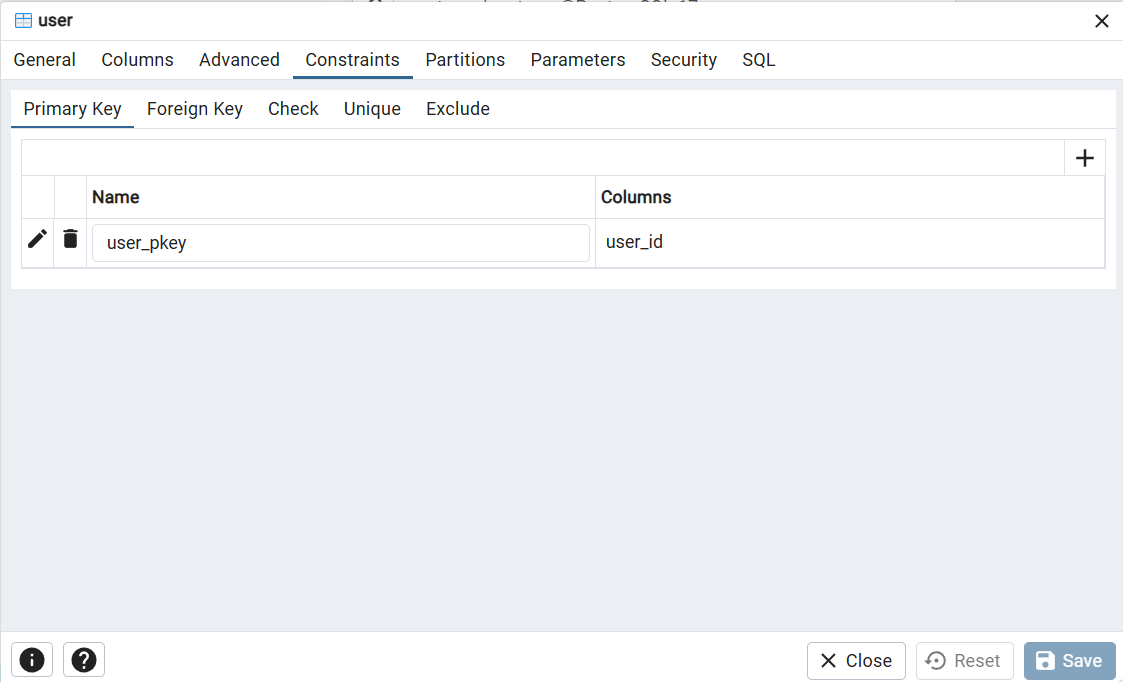




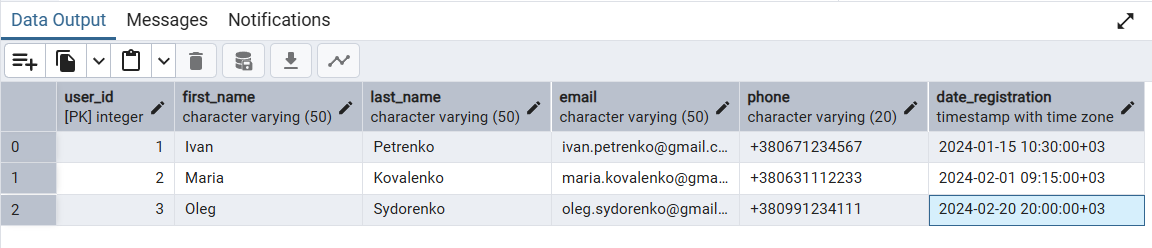


User

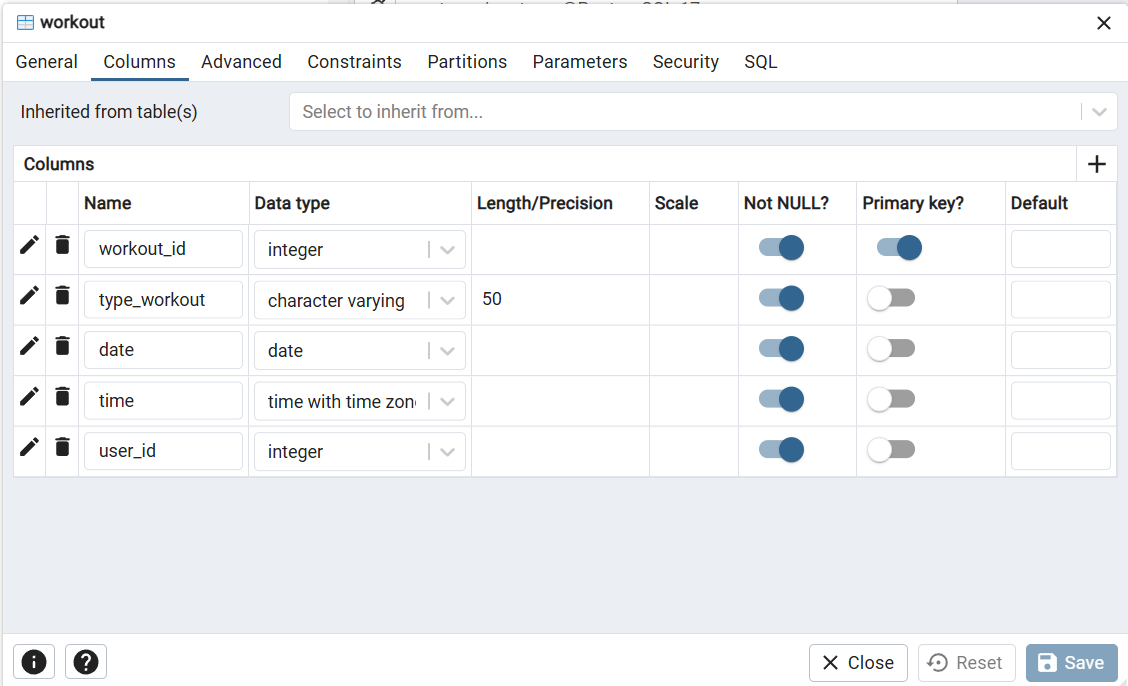


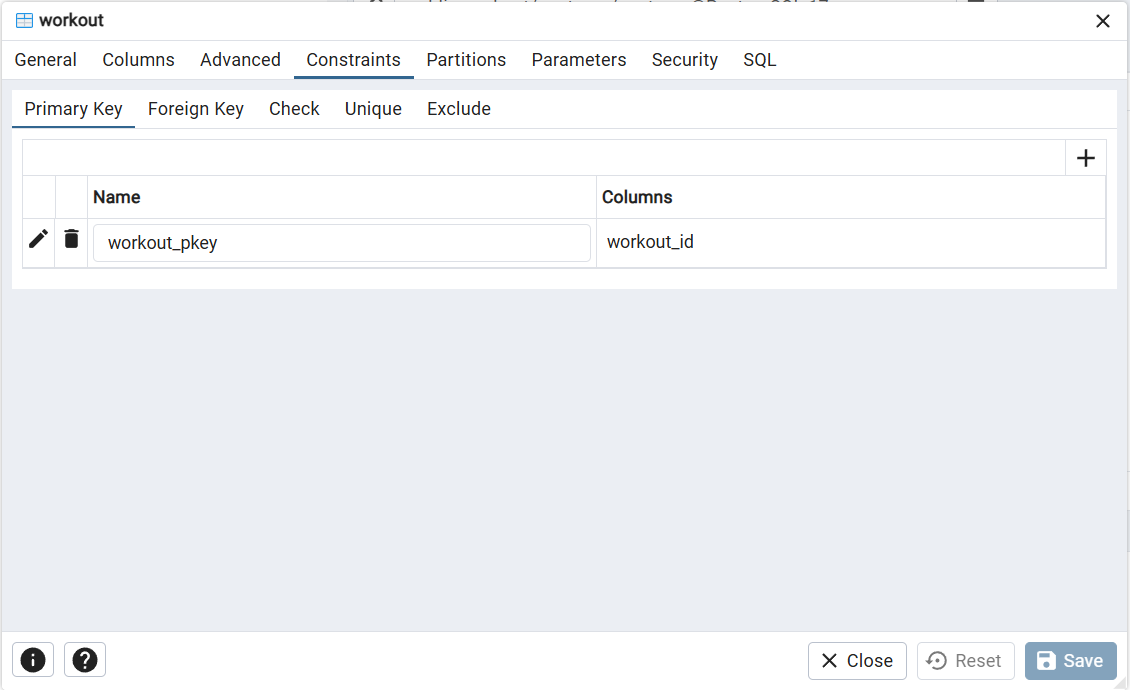


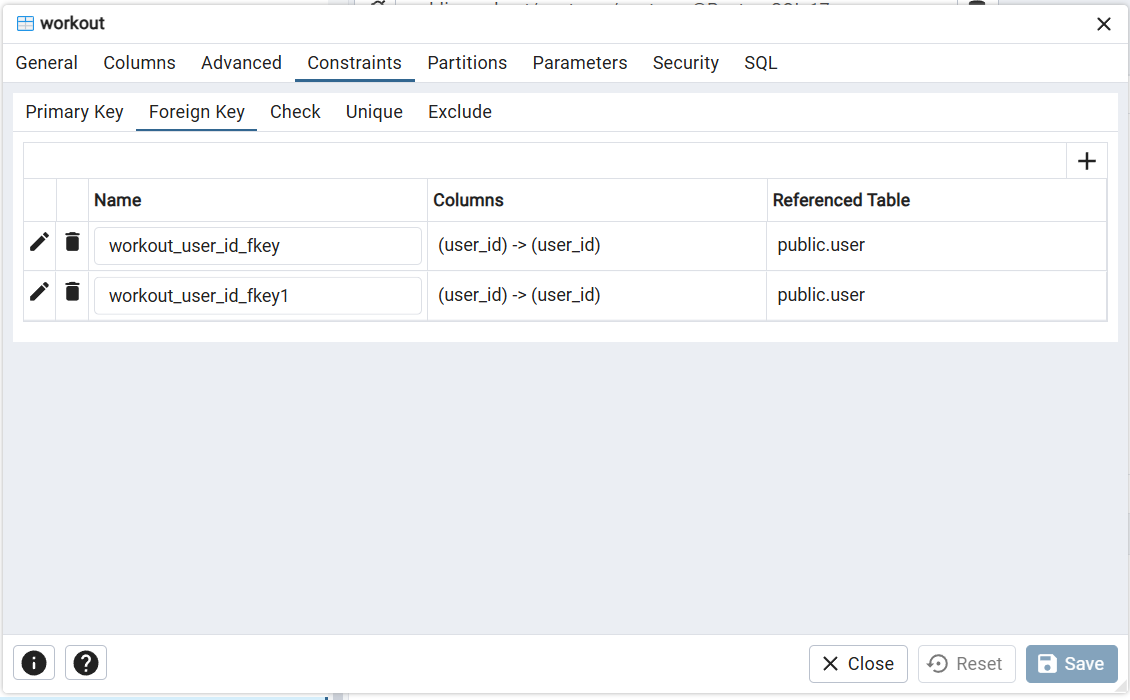
У таблиці User немає зовнішніх ключів(FK)

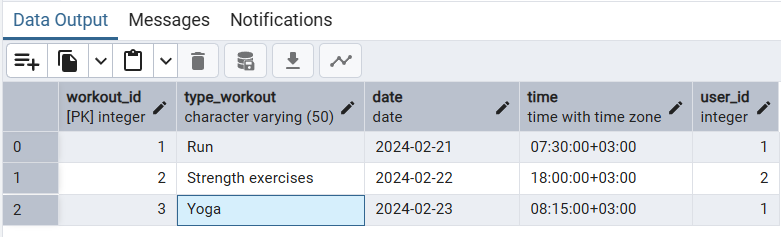


Workout









**Таблиці в коді SQL**

-- This script was generated by the ERD tool in pgAdmin 4.

-- Please log an issue at https://github.com/pgadmin-org/pgadmin4/issues/new/choose if you find any bugs, including reproduction steps.

BEGIN;

CREATE TABLE IF NOT EXISTS public."user"

(

user\_id integer NOT NULL,

first\_name character varying(50) NOT NULL,

last\_name character varying(50) NOT NULL,

email character varying(50) NOT NULL,

phone character varying(20) NOT NULL,

date\_registration timestamp with time zone NOT NULL,

PRIMARY KEY (user\_id)

);

CREATE TABLE IF NOT EXISTS public.friendship

(

"user\_id1,user\_id2" integer NOT NULL,

status character varying(20) NOT NULL,

date date NOT NULL,

user\_id1 integer NOT NULL,

user\_id2 integer NOT NULL,

PRIMARY KEY ("user\_id1,user\_id2")

);

CREATE TABLE IF NOT EXISTS public.workout

(

workout\_id integer NOT NULL,

type\_workout character varying(50) NOT NULL,

date date NOT NULL,

"time" time with time zone NOT NULL,

user\_id integer NOT NULL,

PRIMARY KEY (workout\_id)

);

CREATE TABLE IF NOT EXISTS public."health metrics"

(

metrics\_id integer NOT NULL,

steps integer NOT NULL,

pulse integer NOT NULL,

calories integer NOT NULL,

measurement\_date timestamp with time zone NOT NULL,

user\_id integer NOT NULL,

workout\_id integer NOT NULL,

PRIMARY KEY (metrics\_id)

);

ALTER TABLE IF EXISTS public.friendship

ADD FOREIGN KEY (user\_id1)

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

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public.friendship

ADD FOREIGN KEY (user\_id2)

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

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public.workout

ADD FOREIGN KEY (user\_id)

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

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."health metrics"

ADD FOREIGN KEY (user\_id)

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

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

ALTER TABLE IF EXISTS public."health metrics"

ADD FOREIGN KEY (workout\_id)

REFERENCES public.workout (workout\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID;

END;

**Посилання на GitHub:**