

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ**

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №1**

**з дисципліни Бази даних і засоби управління**

**на тему: “Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL”**

**Виконав:**

**студент ІII курсу**

**групи КВ-01**

**Свисюк Б. О.**

**Перевірив:**

**Київ – 2022**

Метою роботи є здобуття практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

Завдання роботи полягає у наступному:

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.

Зміст звіту

1. Опис проблемного середовища;

2. Концептуальна модель предметної області;

3. Логічна модель (схема) даних БД;

4. Склад СУБД PostgreSQL;

5. Список обмежень цілісності в термінах СУБД PostgreSQL;

6. Фізична модель (схема) даних БД в pgAdmin 4;

7. Приклад вмісту БД

**Опис предметної області «IT-компанія»**

Обрана предметна область передбачає існування фірм та їх власників, працівників які працюють на певну фірму, проектів які належать певній фірмі, а також інформацію про те, який працівник на яких проектах працює.

**Опис сутностей предметної області**

1) Company (компанія), з атрибутами:код фірми**,** назва фірми,дата заснування фірми**.** Компанія, має власника, та містить проєкти з працівниками на них.

2) Owner (власник), з атрибутами: Код власника, Код компанії, якою він володіє, Ім’я власника, Вік власника. Описує власника компанії.

3) Project (проект), з атрибутами: Код проекту, Код компанії, до якої належить, Ім’я проекту, Скільки проект існує (його вік). Описує проект компанії.

4) Employee(працівник), з атрибутами: Код працівника, Код компанії, до якої належить, Ім’я працівника, Вік працівника. Описує працівника компанії.

**Опис зв’язків між сутностями предметної області**

1) Сутність Company має зв’язок 1 до 1 з таблицею Owner, а також 1(Company) до багатьох з таблицями Project та Employee, для всіх є головною.

2) Сутність Owner має зв’язок 1 до 1 з таблицею Company (за допомогою того, що у сутності Owner ід компанії є унікальним і не дозволяє повторів), є залежною від Company.

3) Сутність Project має зв’язок багато до одного з таблицею Company та є залежною від Company, а також має зв’язок багато до багатьох з сутністю Employee.

4) Сутність Employee має зв’язок багато до одного з таблицею Company, є залежною від Company, а також має зв’язок багато до багатьох з сутністю Project.

**Концептуальна модель предметної області «IT-компанія»**

Концептуальна модель наведена на рисунку 1.

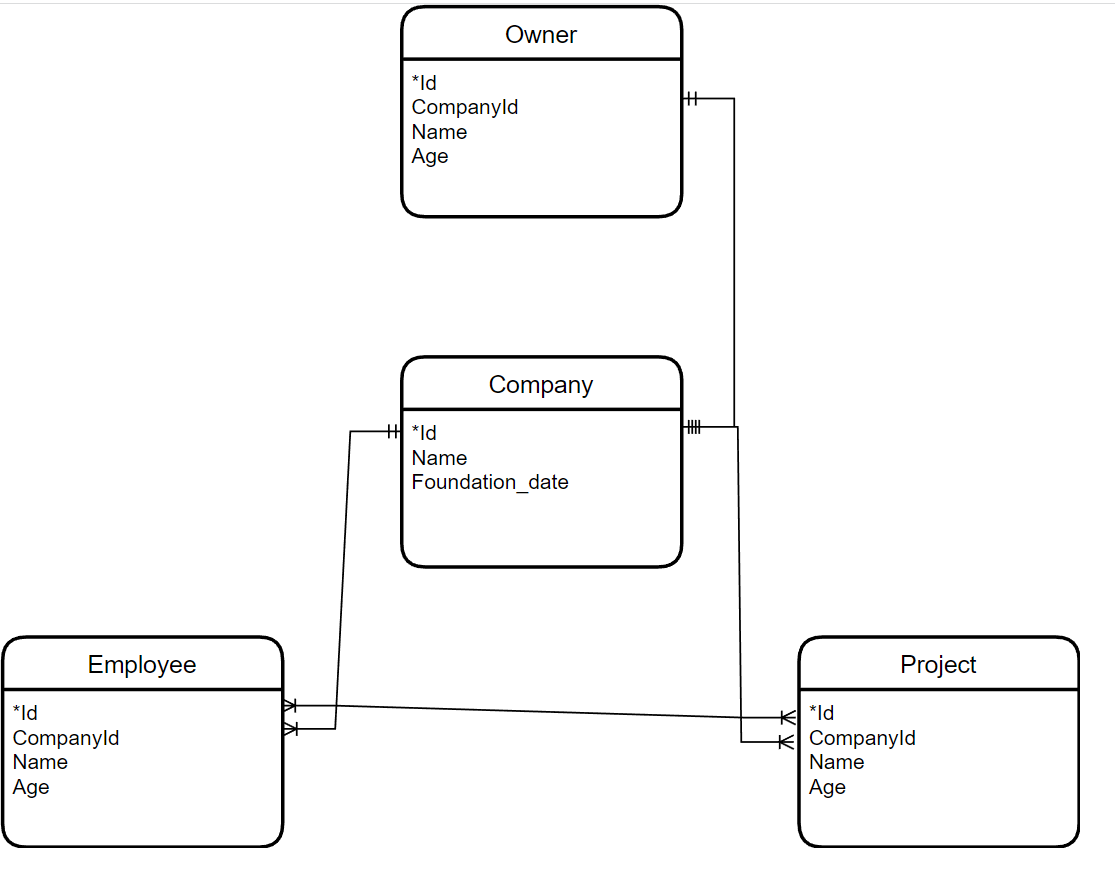


Рисунок 1 - ER-діаграма, побудована за нотацією Crow’s feet (інструмент: draw.io)

**Логічна модель (схема) БД «IT-компанія»**

Логічну модель (схему бази даних наведено на рисунку 2.

1)Сутність Company було перетворено в таблицю з відповідною назвою Company.

2)Сутність Owner було перетворено в таблицю з відповідною назвою Owner.

3)Сутність Project було перетворено в таблицю з відповідною назвою Project.

4)Сутність Employee було перетворено в таблицю з відповідною назвою Employee.

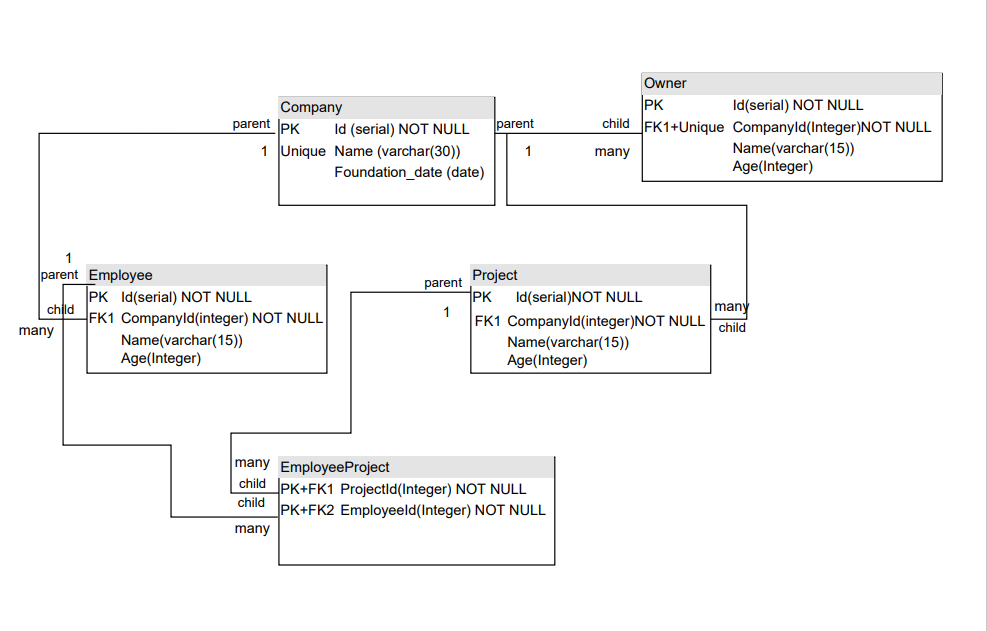
5)Була створена таблиця EmployeeProject зумовив її появу зв’язок M:N між сутностями Project та Employee, є допоміжною для реалізацію цього зв’язку.

Рисунок 2 - Схема бази даних (інструмент: draw.io)

**Опис об’єктів бази даних**

***Company***

|  |  |  |
| --- | --- | --- |
| *ID* | *Serial* | *PK (Not null)* |
| *Name* | *nvarchar(30)* | *Unique* |
| *Foundation\_date* | *date* |  |

***Owner***

|  |  |  |
| --- | --- | --- |
| *ID* | *Serial* | *PK (Not null)* |
| *CompanyId* | *integer* | *FK (Not null, on delete Cascade) Unique* |
| *Name* | *nvarchar(15)* |  |
| *Age* | *integer* |  |

***Employee***

|  |  |  |
| --- | --- | --- |
| *ID* | *Serial* | *PK (Not null)* |
| *CompanyId* | *integer* | *FK (Not null, on delete Cascade)* |
| *Name* | *nvarchar(15)* |  |
| *Age* | *integer* |  |

***Project***

|  |  |  |
| --- | --- | --- |
| *ID* | *Serial* | *PK (Not null)* |
| *CompanyId* | *integer* | *FK (Not null, on delete Cascade)* |
| *Name* | *nvarchar(15)* |  |
| *Age* | *integer* |  |
|  |  |  |

***EmployeeProject***

|  |  |  |
| --- | --- | --- |
| ProjectId | *Serial* | *PK FK (Not null, on delete cascade)* |
| EmployeeId | *Serial* | *PK FK (Not null, on delete cascade)* |

**Відповідність схеми бази даних до третьої нормальної форми**

**Схема відповідає 1НФ, тому що:**

Кожна клітинка таблиць може містити лише одне значення, а кожен новий запис не може повторювати жодний з попередніх бо є унікальним.

**Схема відповідає 2НФ, тому що:**

Схема відповідає 1НФ і в схемі відсутні композитні первинні ключі, а отже первинні ключі складаються з одного атрибуту.

**Схема відповідає 3НФ, тому що:**

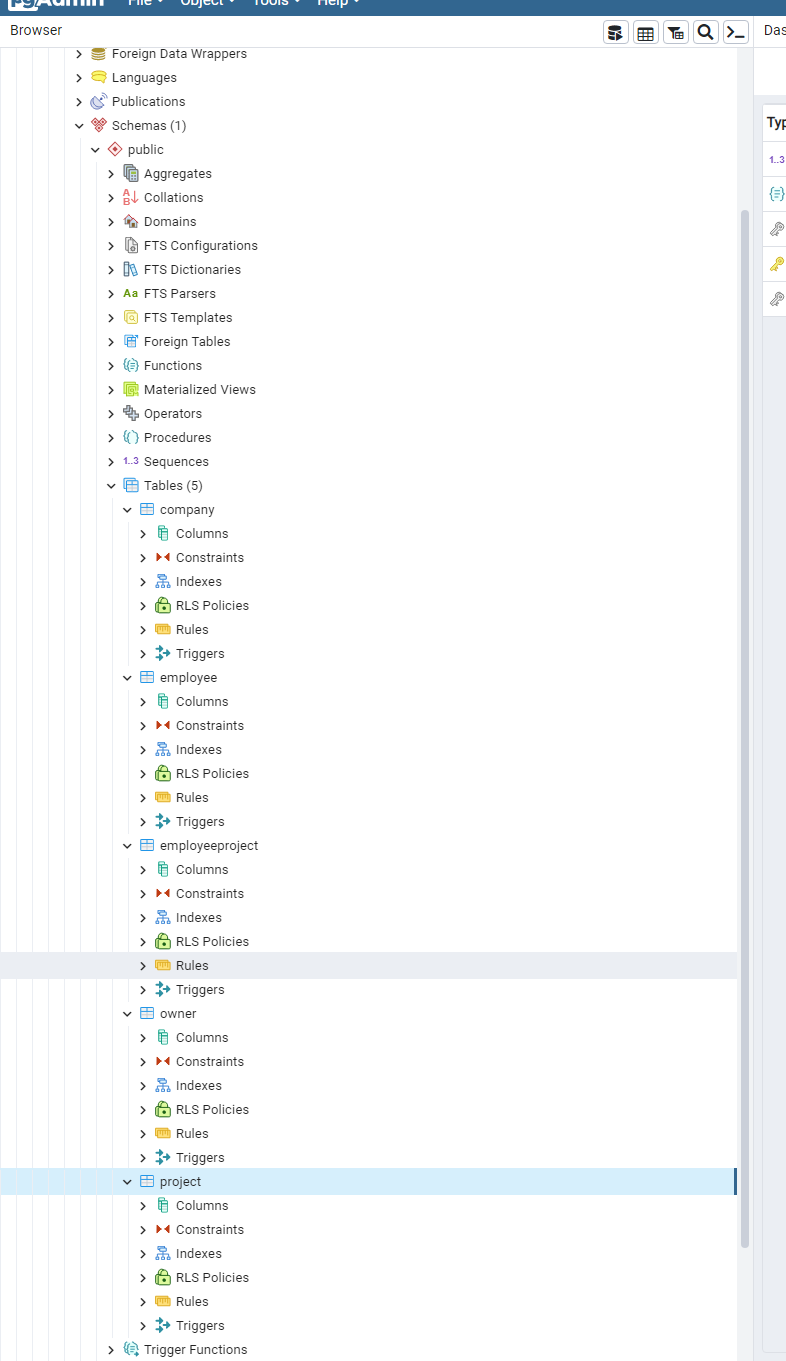
Схема відповідає 2НФ і кожен атрибут, що не являє собою первинний ключ, є функціонально залежним від первинного ключа.

**З прикладами з ДБ:**

Таблиця не містить наборів значень у окремих клітинках (на перехресті стовпчика та рядка), тобто значення які містить таблиця є скалярними, а також таблиці не містять повторюваних рядків, цього ми досягаємо за допомогою використання PRIMARY KE, що доводить що таблиця відповідає правилам НФ1.

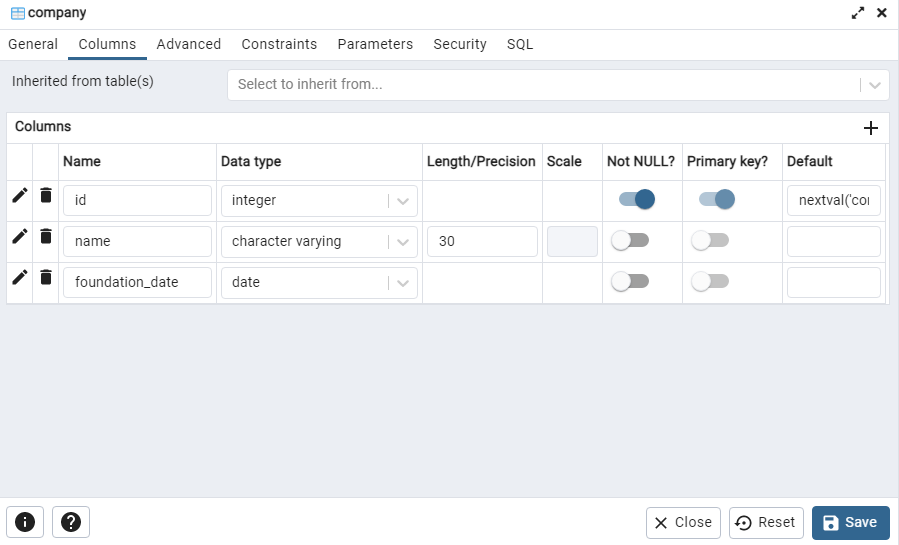
Так як більшість таблиць містить не компонований ключ, ми можемо стверджувати що всі інші атрибути автоматично знаходятся в повній функціональній залежності від первичного ключа (Розберемо таблиці Owner,Company,Project та Employee. Усі таблиці крім Company мають однаковий набір атрибутів а саме age,name,compayid та id, в усіх випадках крім таблиці owner, для всіх атрибутів можна вилаштовувати транзит лише від ключа, тобто дізнатись ім’я, вік та компанію до якої належить ця сутність, в інших випадках я не отримаю унікальну інформацію, будь то фірма(окрім випадку овнер, де фірма це потенційний ключ),вік або ім’я – я отримаю вибірку. Щодо таблиці Овнер – в ній я маю потенційний ключ у виглдяі ід компанії,на який проте не розпоширюється ні правила про 2НФ, ні про 3, те ж саме з Company, де унікальним є ім’я. Щодо таблиці EmployeeProject, то фактично таблиця складається з 2 атрибутів які і є компонованим ключем, тобто в них немає залежних атрибутів, а отже Таблиці відповідають НФ2.( Передбачуючи питання скажу, що в цю таблицю можна було б добавити поле, яке оцінювало б успішність певного працівника на певному проекті( якщо в працівника багато проектів), тоді оцінка залежала б, як і від проекту, так і від того, що це за працівник, тобто цей атрибут був би функціонально залежним від повного ключа, а не якоїсь його частини, будь то, що це за працівник, або ж що це за проект(в такому випадку, таблиця б не відповідала 2НФ) ).  
Всі атрибути розроблених таблиць залежать лише від первичного ключа, тобто у атрибутів ми не можемо прослідкувати транзитивність, а отже таблиці відповідають 3НФ.

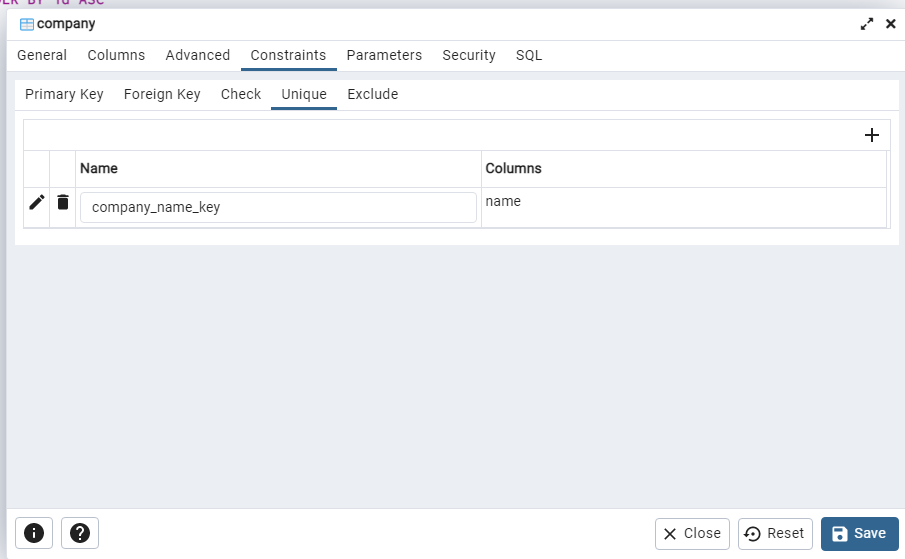
**Фізична модель БД «IT-компанія» у pgAdmin4**

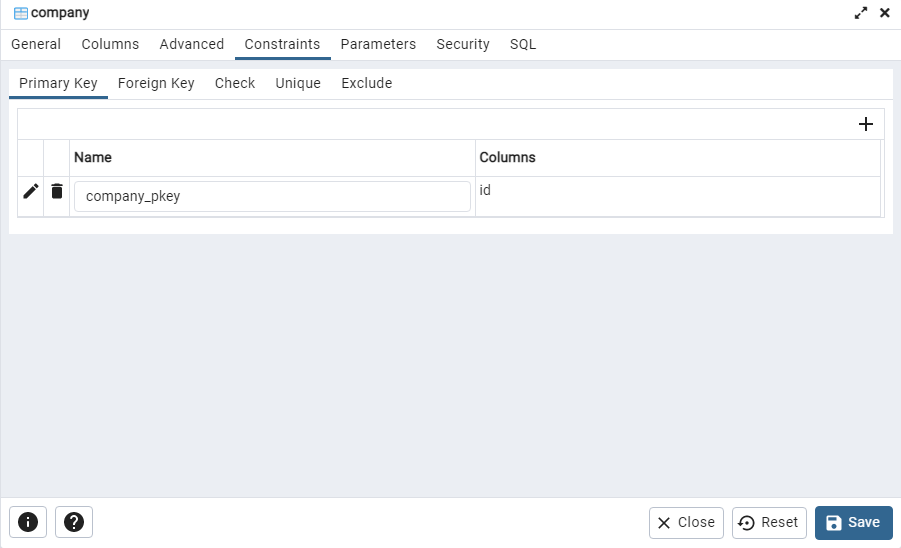


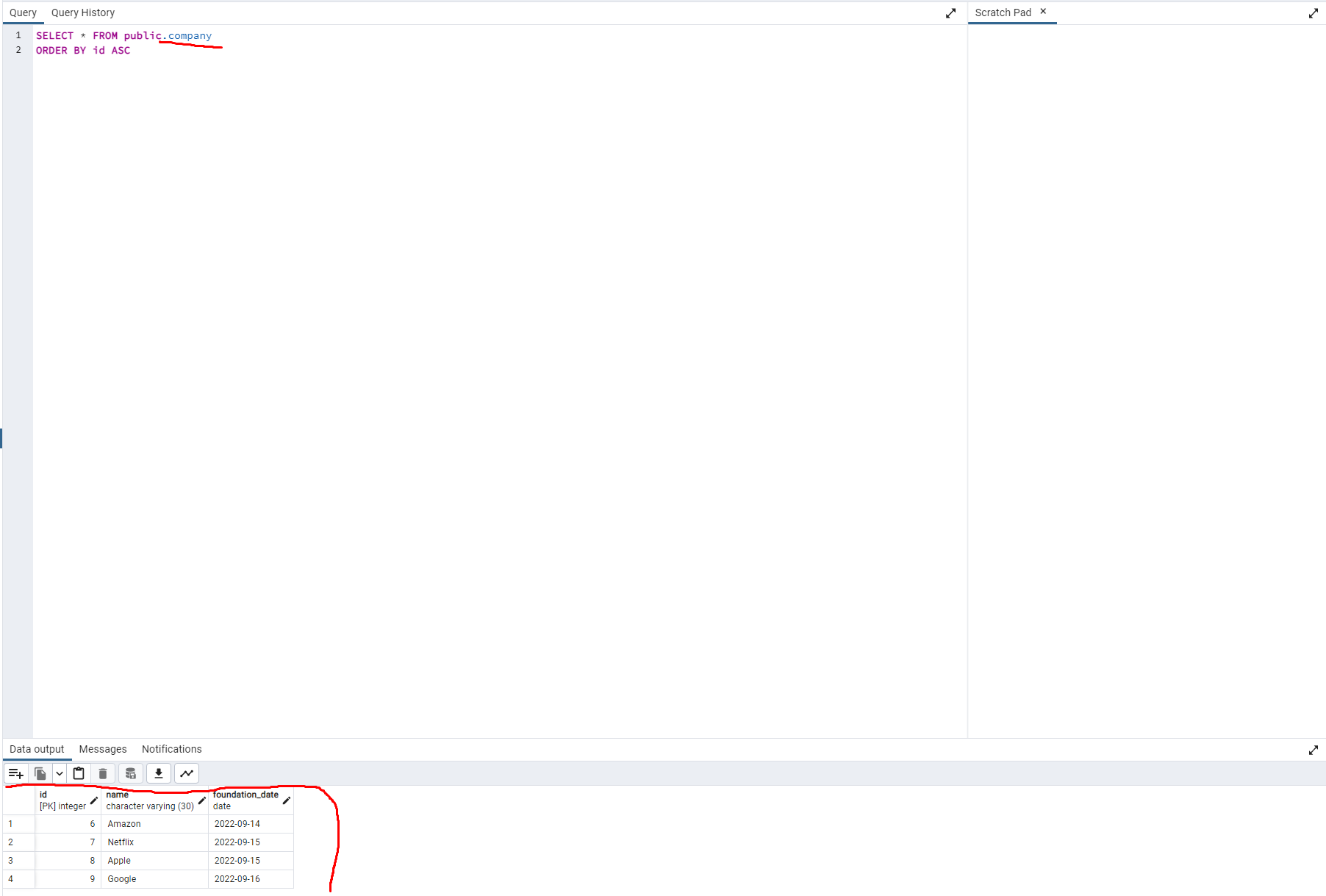
**Фотографії вмісту та обмежень таблиць**

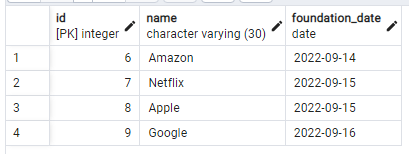
**Company:**



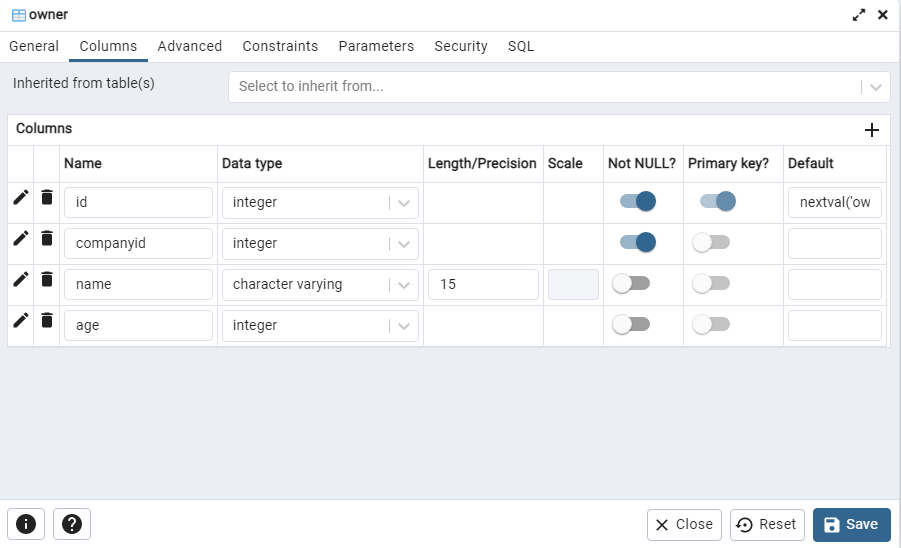


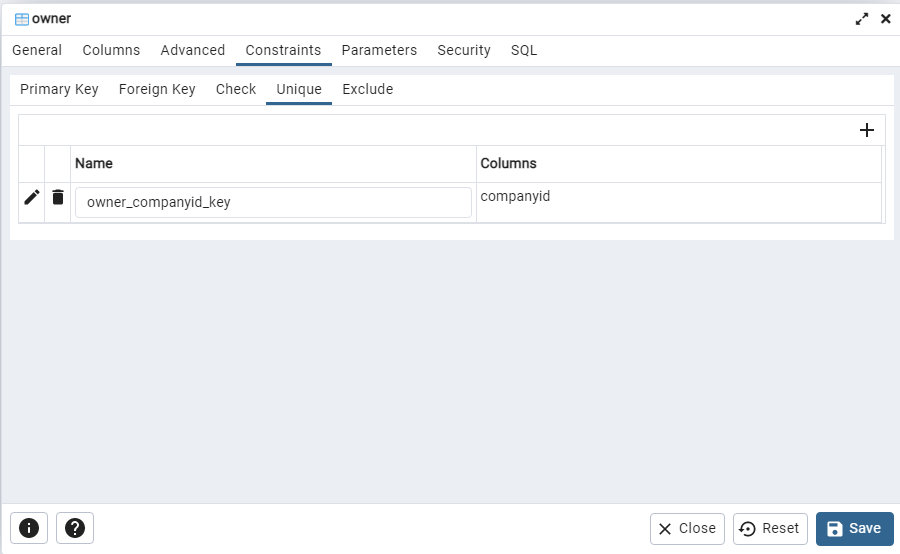


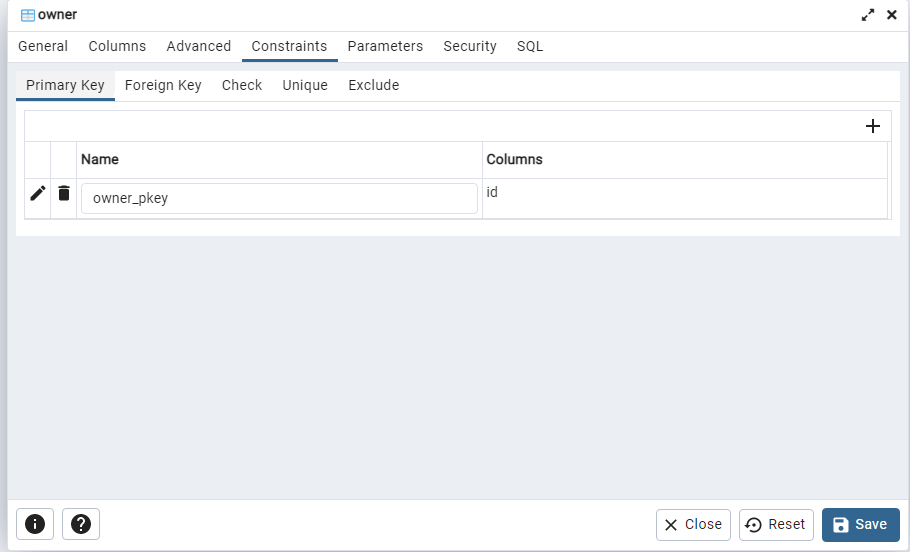


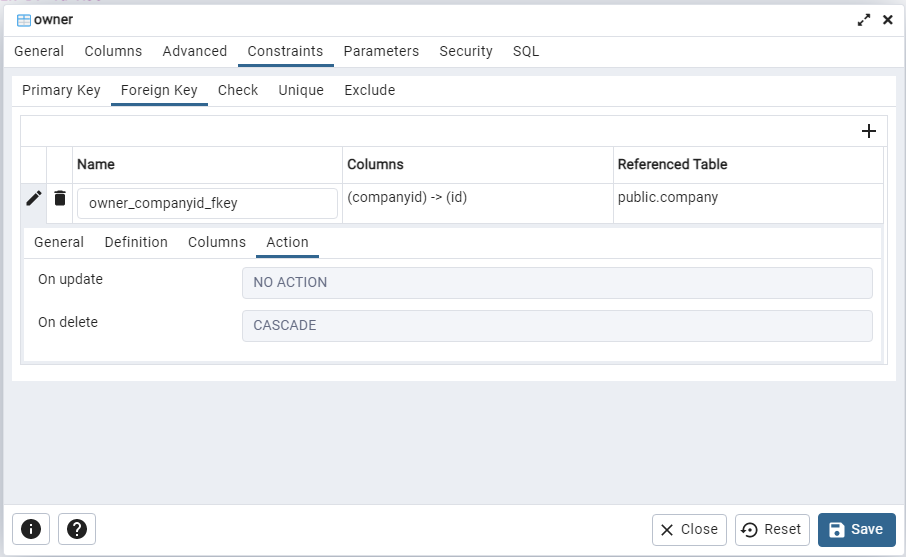


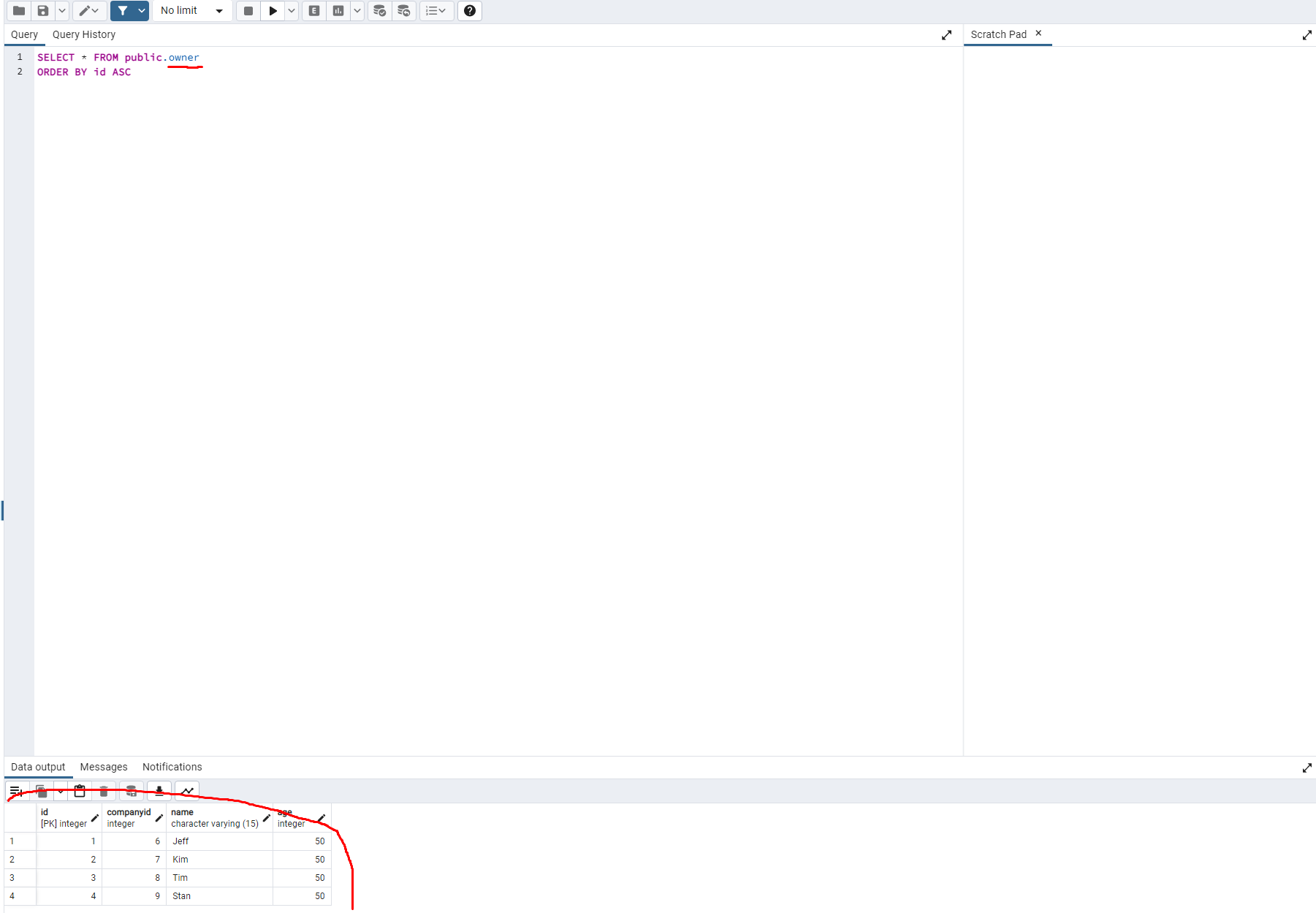
**Owner:**

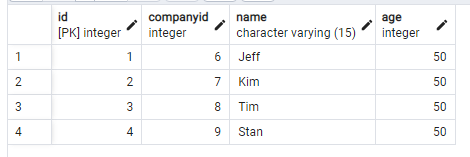




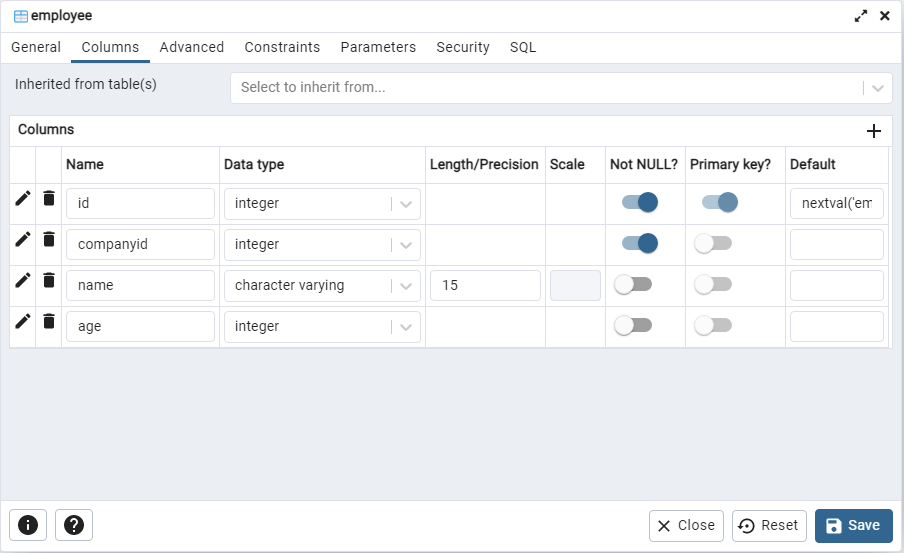


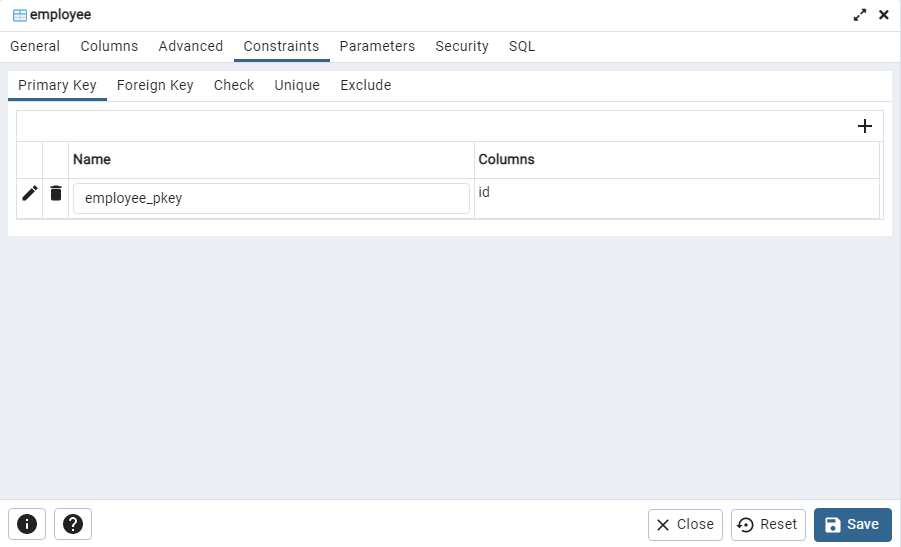


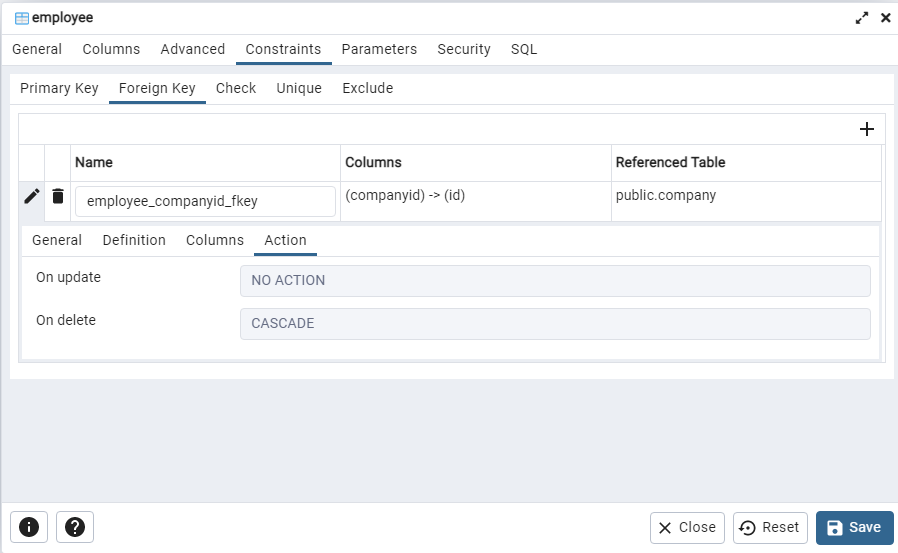


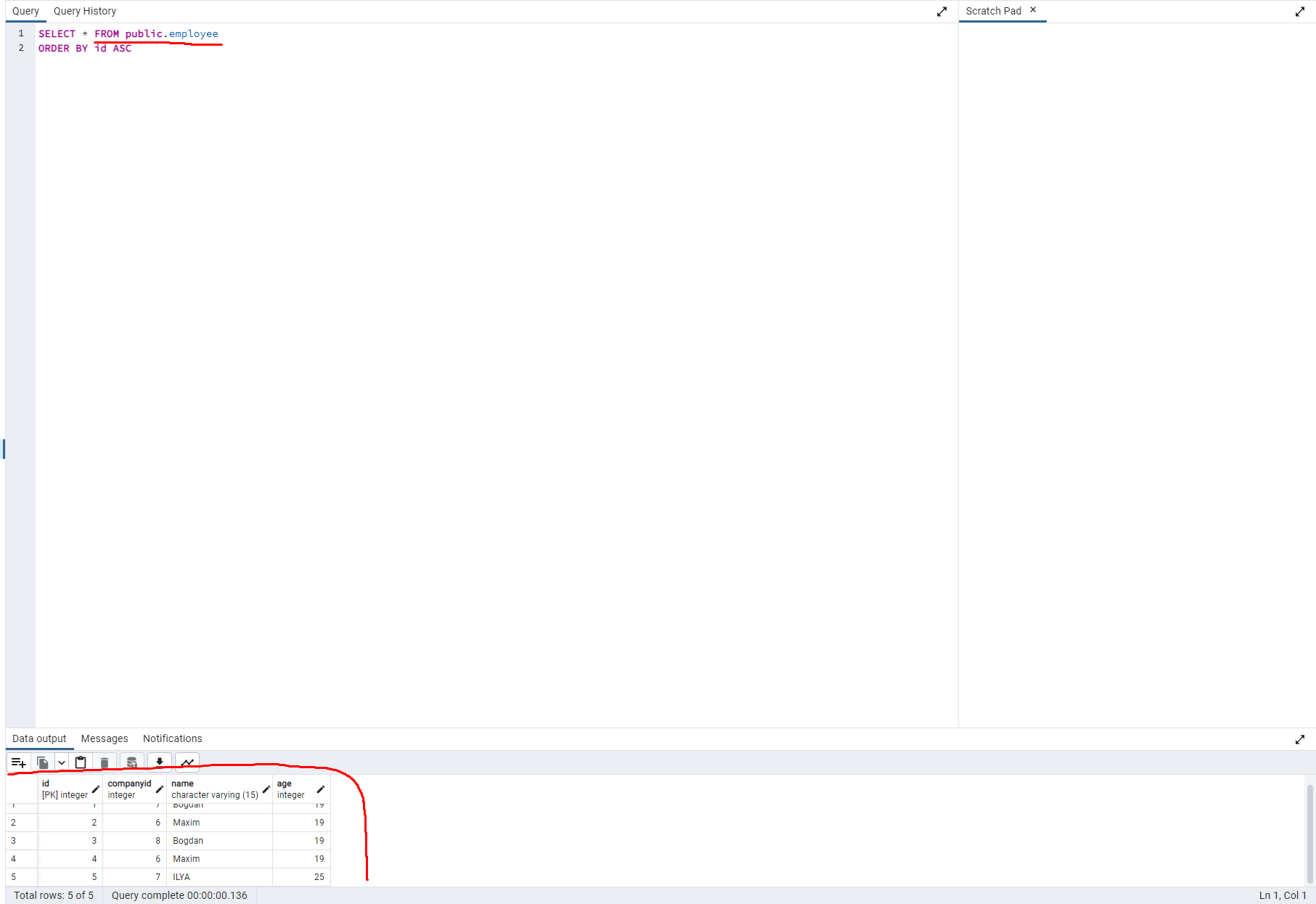


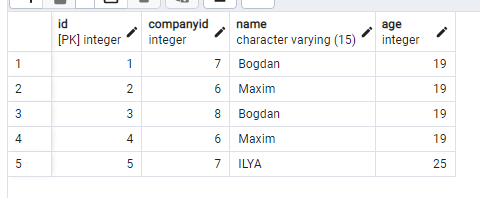
**Employee:**



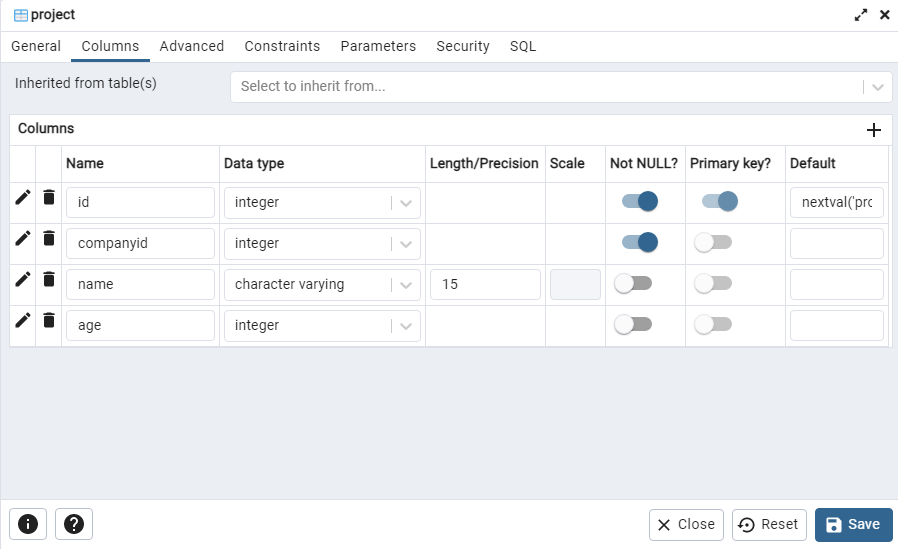


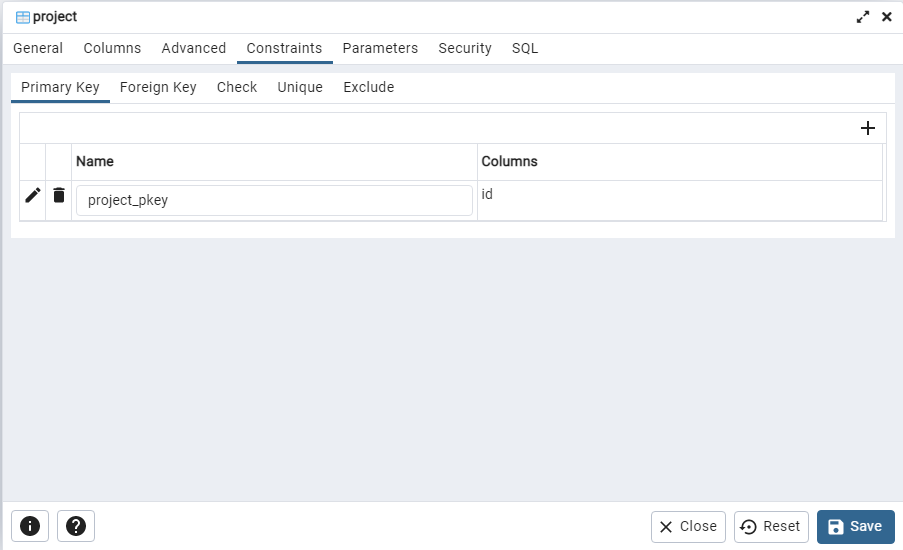


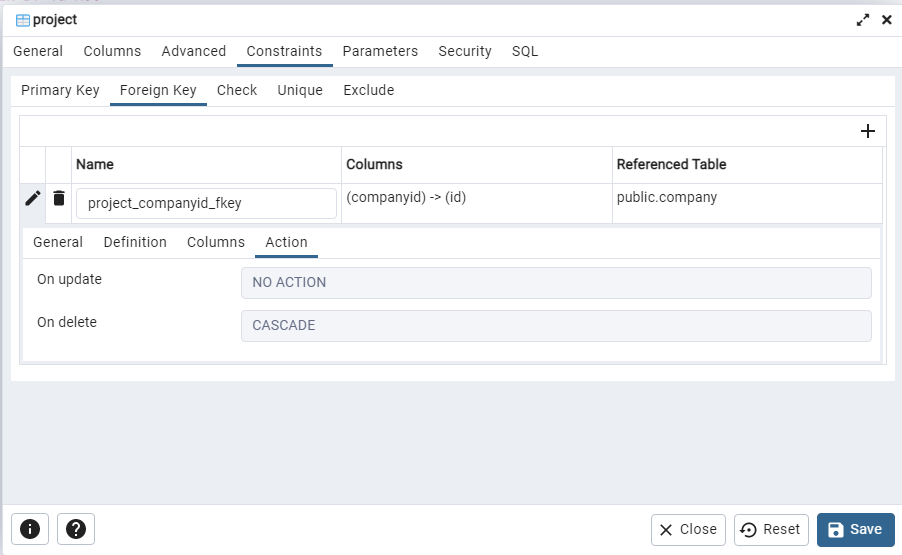


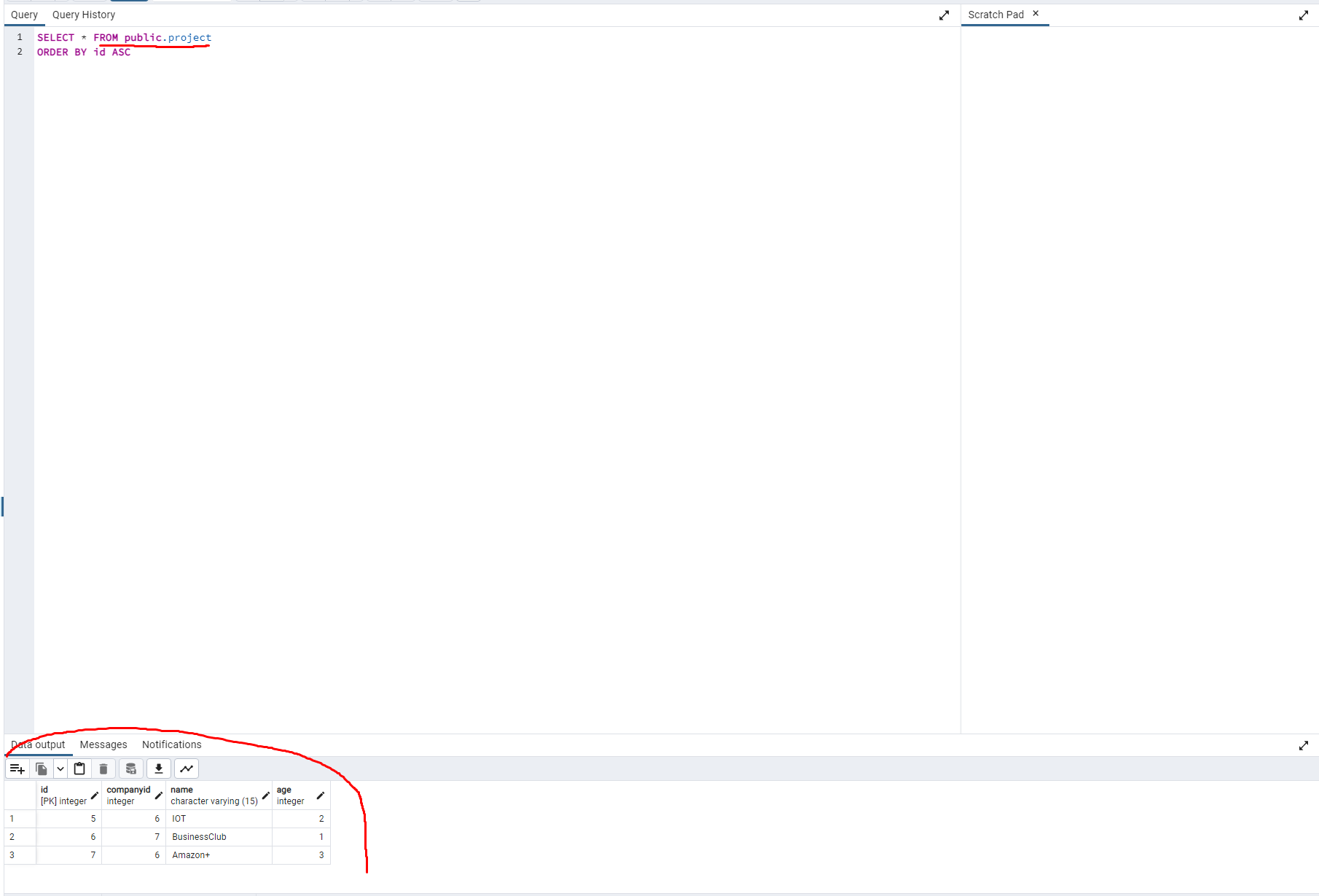


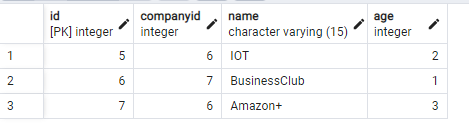
**Project:**



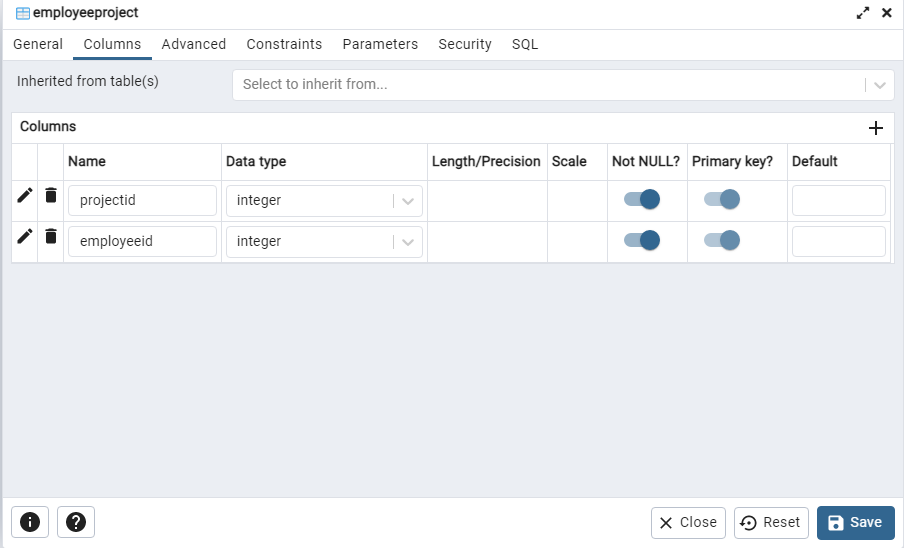


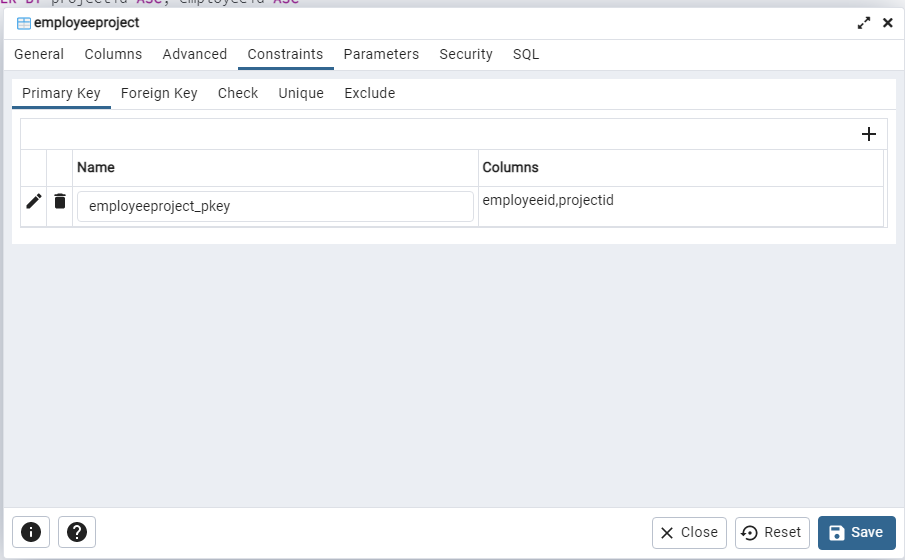


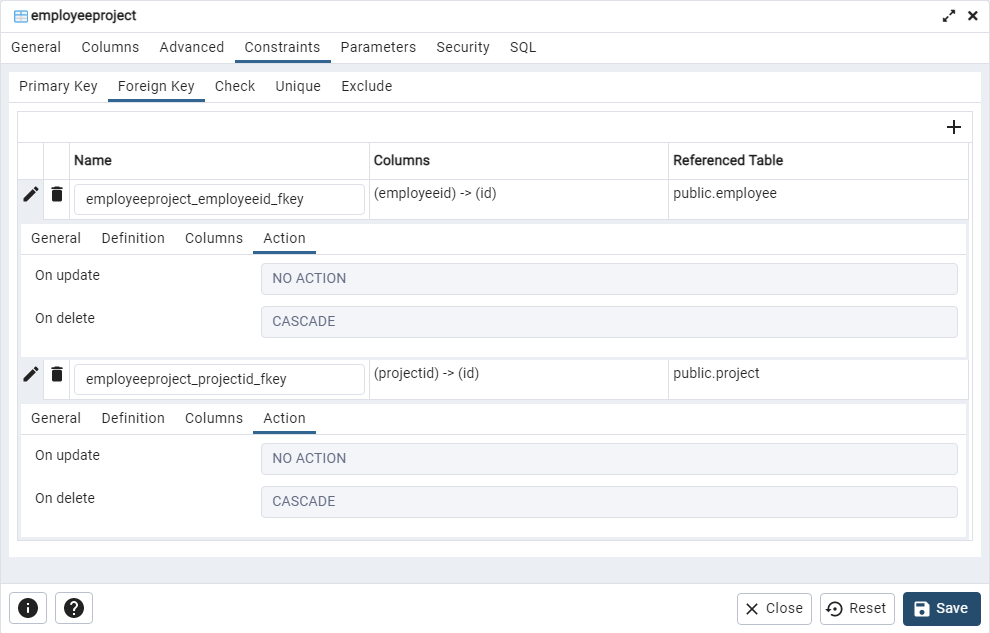


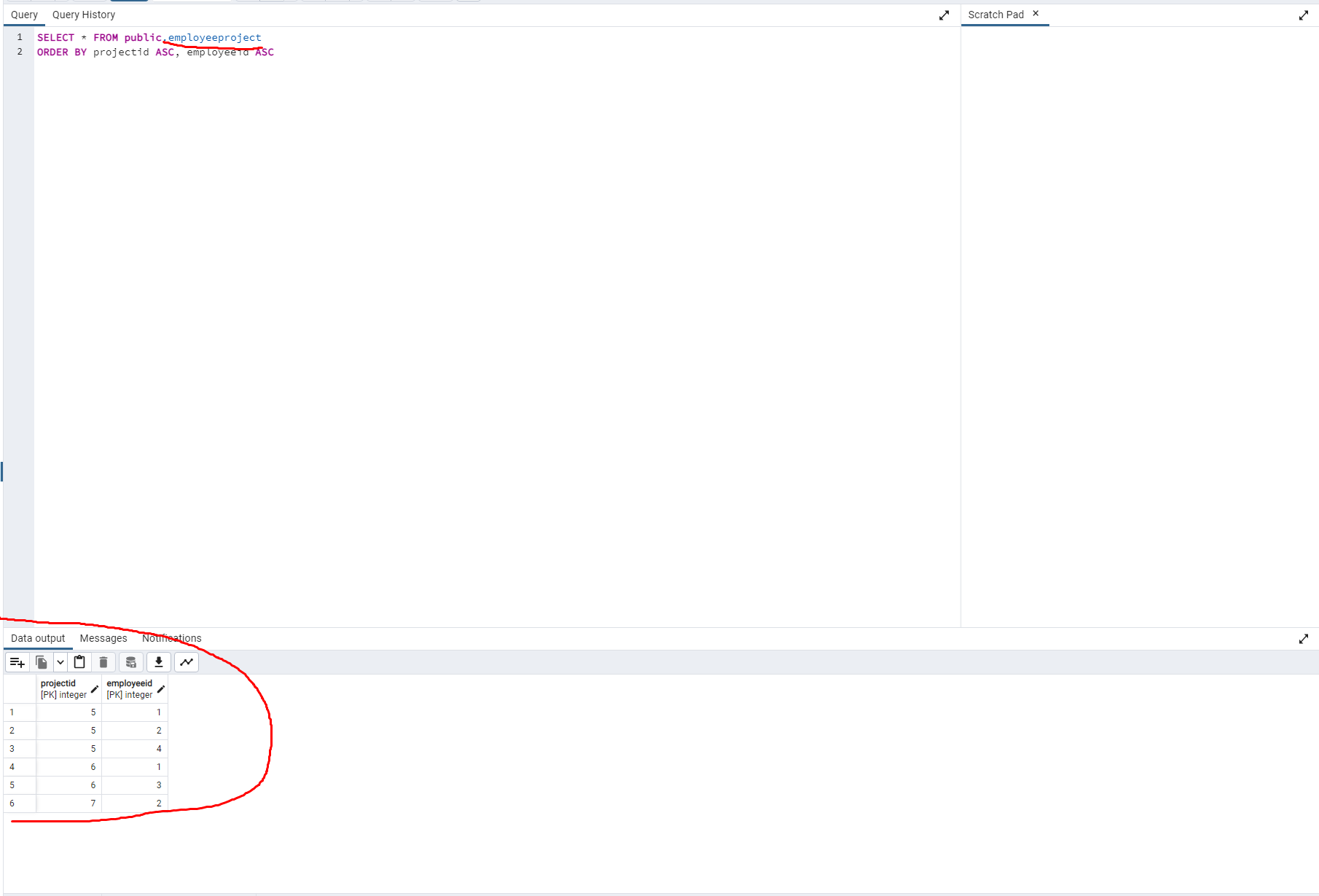


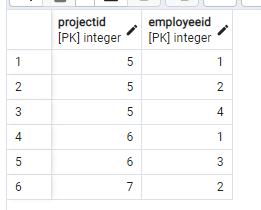
**EmployeeProject:**











**SQL:**

CREATE TABLE Company(

  Id SERIAL PRIMARY KEY,

  Name CHARACTER VARYING(15) UNIQUE,

  foundationDate date

);

CREATE TABLE Owner

(

  Id SERIAL PRIMARY KEY,

  CompanyId INTEGER UNIQUE,

  Name CHARACTER VARYING(15),

  Age INTEGER,

  FOREIGN KEY (CompanyId) REFERENCES Company (Id) ON DELETE CASCADE

);

CREATE TABLE Employee

(

  Id SERIAL PRIMARY KEY,

  CompanyId INTEGER,

  Name CHARACTER VARYING(15),

  Age INTEGER,

  FOREIGN KEY (CompanyId) REFERENCES Company (Id) ON DELETE CASCADE

);

CREATE TABLE EmployeeProject

(

  ProjectId INTEGER PRIMARY KEY,

  EmployeeId INTEGER PRIMARY KEY,

  FOREIGN KEY (ProjectId) REFERENCES Project (Id) ON DELETE CASCADE,

  FOREIGN KEY (EmployeeId) REFERENCES Employee (Id) ON DELETE CASCADE

);

CREATE TABLE Project

(

  Id SERIAL PRIMARY KEY,

  CompanyId INTEGER,

  Name CHARACTER VARYING(15),

  Age INTEGER,

  FOREIGN KEY (CompanyId) REFERENCES Company (Id) ON DELETE CASCADE

);

**Посилання на гітхаб:**

<https://github.com/bogdan213114/Lab1/tree/Lab1>