

Міністерство освіти і науки України Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп’ютерних систем

Розрахунково-графічна робота

*з дисципліни*

«Бази даних та засоби управління»

**Тема: «Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL»**

Виконав студент групи:

КВ-11 Чебан М. Д.

Перевірив: Петрашенко А. В.

Оцінка:

**Київ – 2023**

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

*Завдання* роботи полягає у наступному:

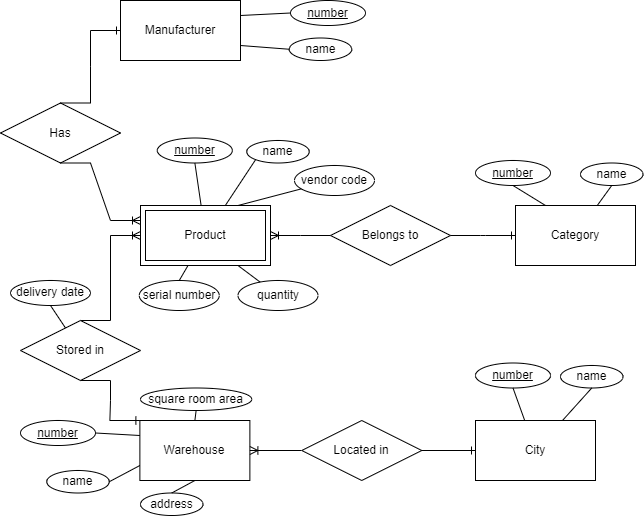
*Загальне* *завдання* роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Дана робота була виконана за допомогою мови програмування **C#**, структурованої мови запитів **SQL**, СУБД **pgAdmin4**, а також з використанням фреймворку на платформі .NET під назвою **Entity Framework Core**. Це фреймворк для взаємодії додатків на платформі .NET із базами даних будь-якого типу.

Посилання на Github репозиторій: https://github.com/m0ksemm/CSharp-.NET-tasks/tree/master/RGRDatabaseTheoryEntityFrameworkPostgreSQL/RGR

Модель «сутність-зв’язок» предметної галузі для проектування бази даних «Inventory of warehouse accounting». Предметна галузь - «Інвентаризація складського обліку», яка була використана під час виконання РГР.



Малюнок 1. ER-діаграма побудована за нотацією «Crow`s foot»

**Сутності з описом призначення:**

Предметна галузь «Inventory of warehouse accounting» включає в себе 5 сутностей, кожна сутність містить декілька атрибутів:

1. Product (Id, name, vendor code, quantity, serial number).
2. Manufacturer (Id, name).
3. Category (Id, name).
4. Warehouse (id, name, address, square room area).
5. City (Id, name).

Сунтість Product описує продукти, якиі зберігається на складі. Кожний продукт має свій ідентифікатор Id, а також містить інформацію про свою назву, код постачальника, серійний номер та кількість продукту(товару).

Сутність Manufacturer описує виробника продукту. Кожний виробник має свій ідентифікатор та назву.

Сутність Category описує категорію продукту. Кожна категорія має свій ідентифікатор та назву.

Сутність Warehouse описує склади, на яких будуть зберігатись продукти. Кожний склад має свій ідентифікатор, назву, адресу, розмір площі у м.кв.

Сутність City відповідає за міста, в яких розташовані склади. Кожне місто має свій ідентифікатор та назву.

**Зв’язки між сутностями:**

Зв’язок між Product та Category:

Кожний товар відноситься до певної категорії. Наприклад: «Cell phones» – мобільні телефони, «Appliances» - побутова техніка, «Furniture» - меблі і т.д. Це може бути потрібно, наприклад, для того, щоб сортувати та розподіляти товар по певних групах на складах. Зв'язок 1:N – до однієї категорії може належати багато різних товарів.

Зв’язок між Product та Manufacturer:

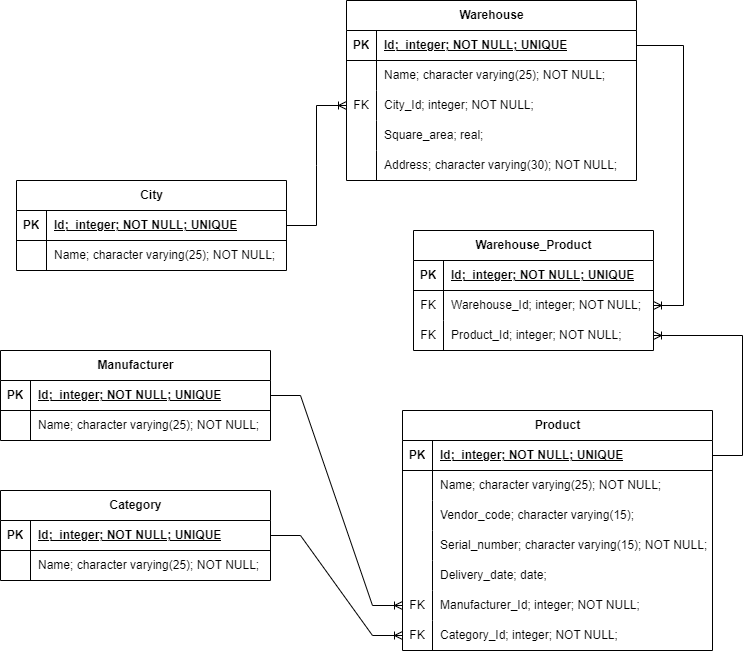
Кожний товар має свого виробника. Простіше кажучи, свою фірму. Зв'язок 1:N – один виробник може виготовляти багато різних товарів. Товар без виробника бути не може.

Зв’язок між Product та Warehouse:

Продукти зберігаються на складі. Оскільки на одному складі зберігаються багато продуктів, зв’язок 1:N.

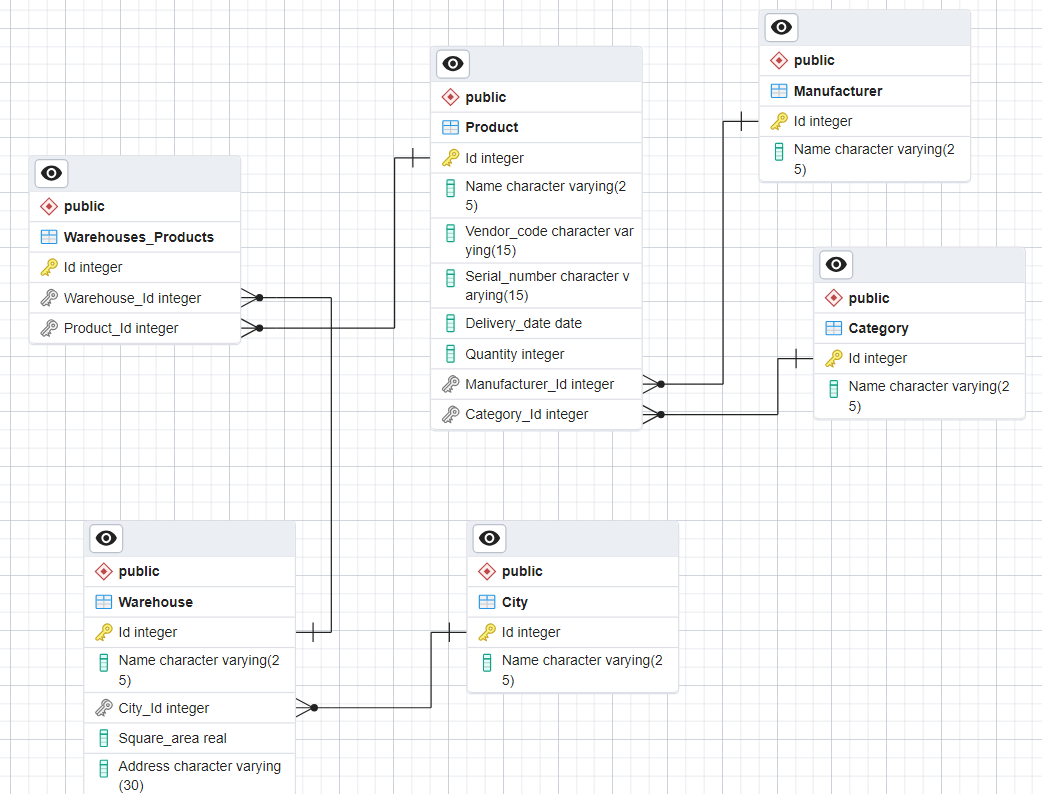
Зв’язок між Warehouse та City:

Склади розташовані в певних містах або селищах. Зв'язок 1:N – в одному місті може бути багато складів. Склад має обов'язково бути розташованим в якомусь місті.

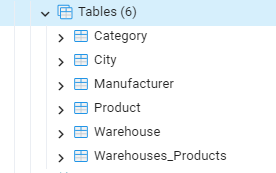
Перетворена модель «сутність-зв’язок» у схему бази даних PostgreSQL

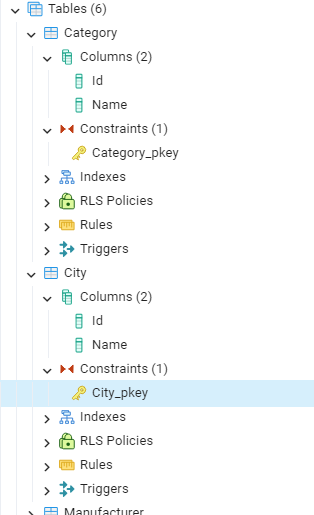
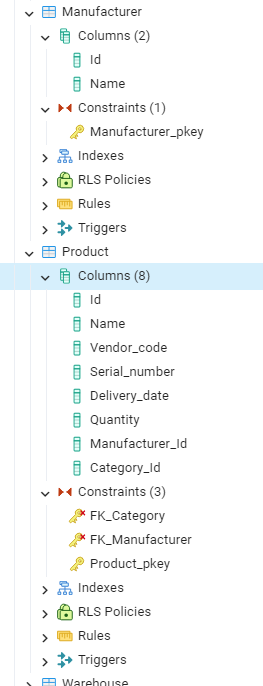
Малюнок 2. Схема бази даних у графічному вигляді

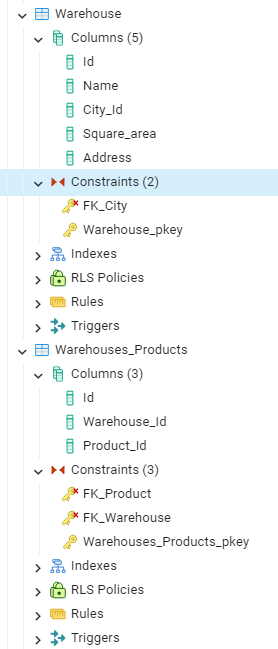
Структури таблиць бази даних в PGAdmin4:



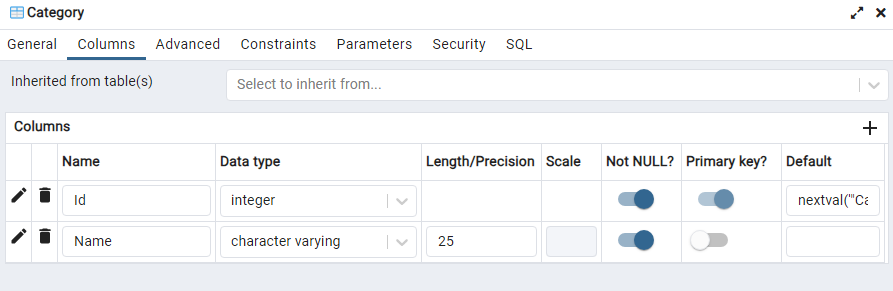
Малюнок 3. Схема бази даних у pgAdmin4

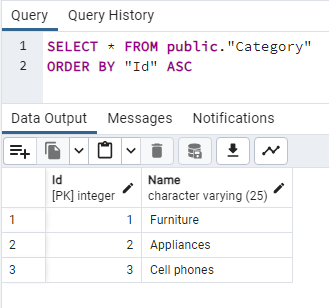


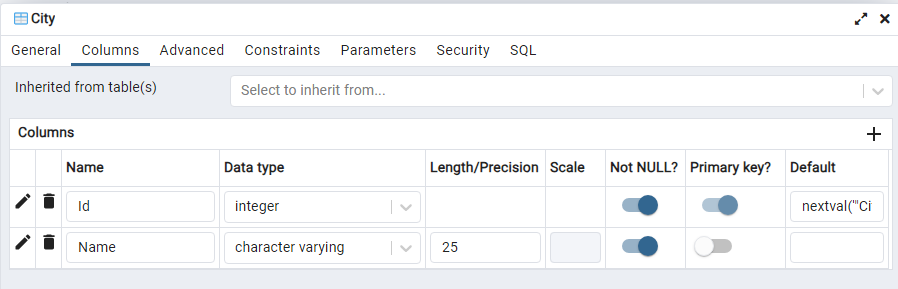


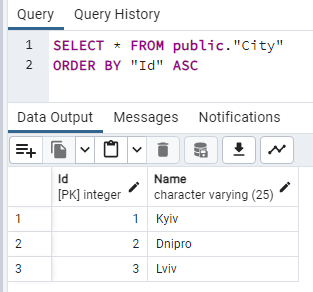
**Таблиця Category:**



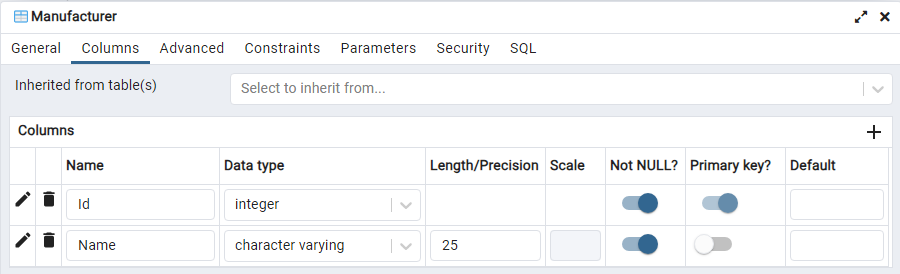


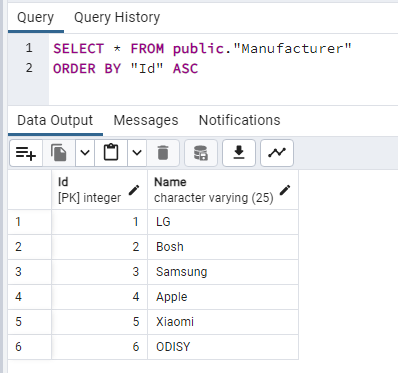
**Таблиця City:**



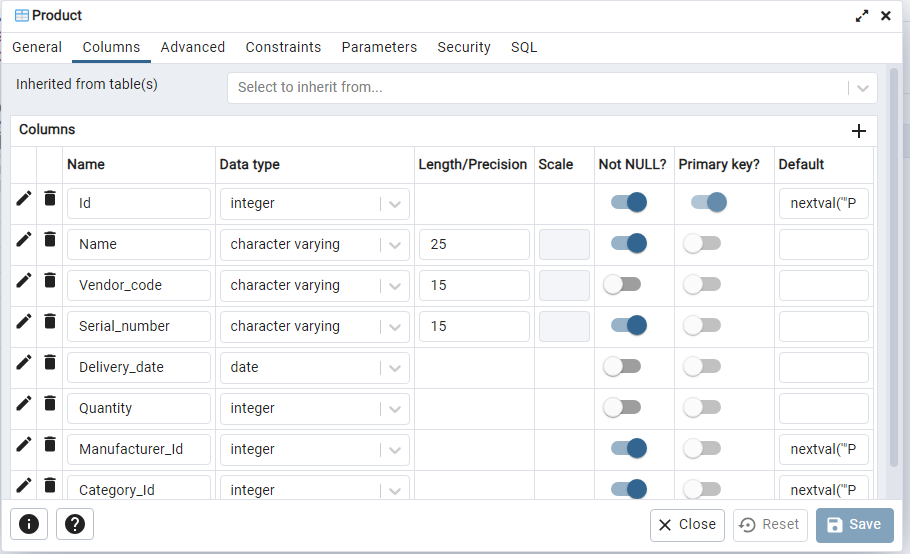


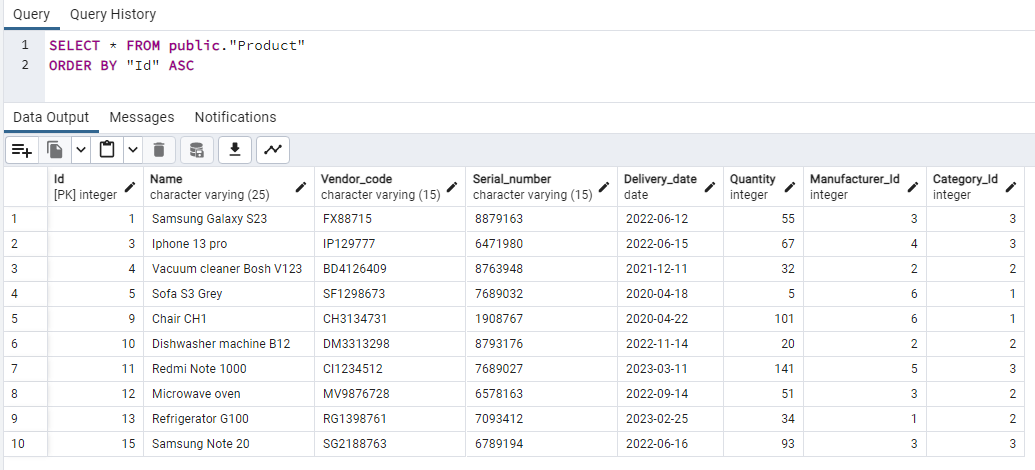
**Таблиця Manufacturer:**



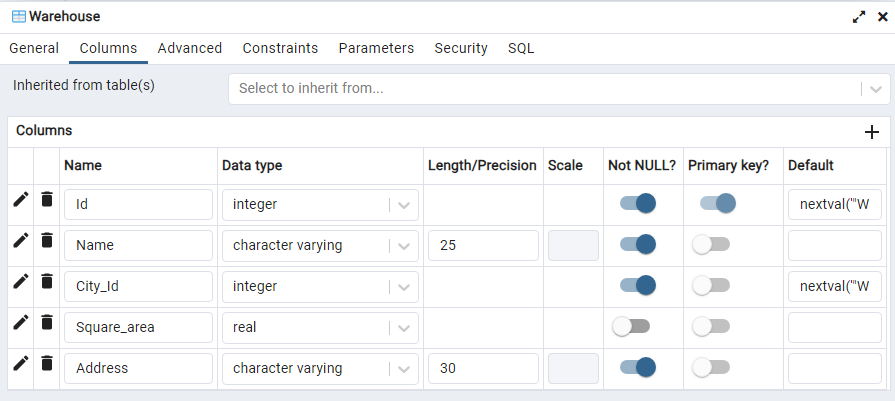


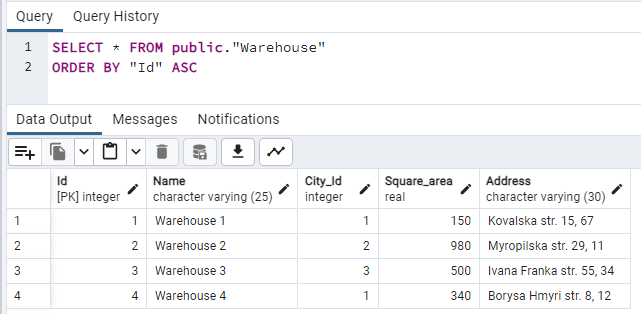
**Таблиця Product:**



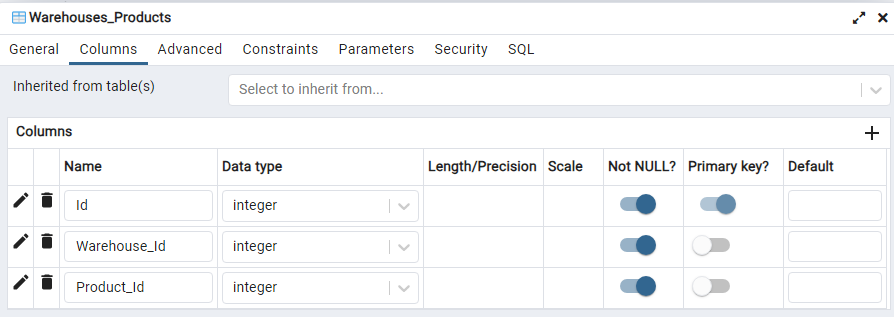


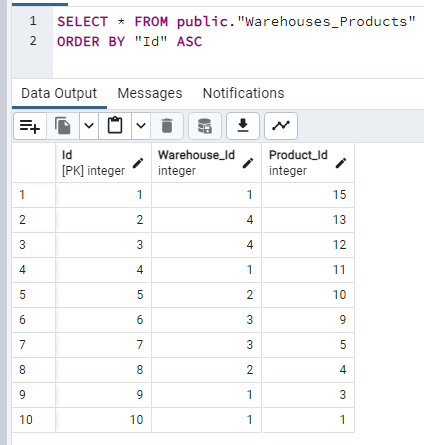
**Таблиця Warehouse:**





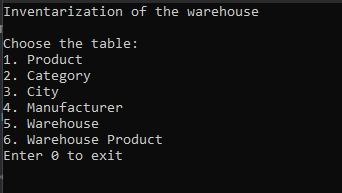
**Таблиця Warehouses\_Products:**



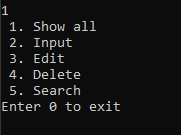


Cхема меню користувача з описом функціональності кожного пункту

Головне меню програми:



В цьому меню користувачу пропонується обрати таблицю, з якою він хоче працювати. БД складається з шістьох таблиць. Відповідно, пунктів в таблиці також шість. Кожний пункт меню відповідає назві таблиці.



Після вибору необхідного пункту меню відкривається наступний пункт – вибір необхідної дії, яку треба застосувати до таблиці. Пропонуються наступні дії:

1. Показати всі записи таблиці

Після вибору цього меню буде виведено всі записи таблиці.

1. Ввести нові дані

Буде запропоновано ввести нові дані до кожного поля запису таблиці

1. Редагувати вже існуючі дані

Спочатку будуть виведені всі записи таблиці для того, щоб користувач обрав той запис(обрав його індекс), який він хоче відредагувати. Після цього буде запропоновано ввести нові дані до обраного запису.

1. Видалення запису

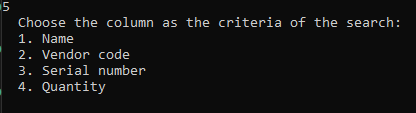
Спочатку знову будуть виведені всі записи таблиці для того, щоб користувач обрав той запис(обрав його індекс), який він хоче видалити. Після цього цей запису буде видалено, якщо від цього рядка не залежать інші рядки в інших табличках. В іншому випадку користувачеві в консолі буде видане повідомлення, що він не може видалити цей запис.

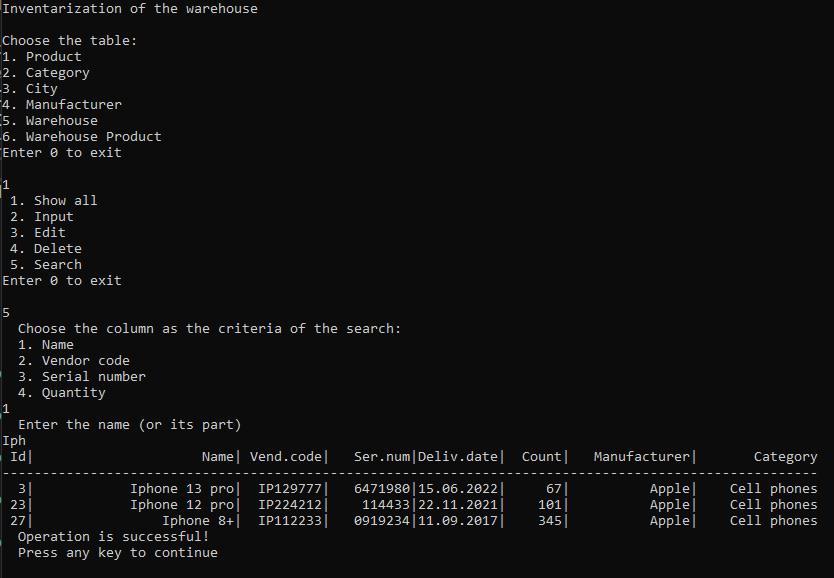
1. Пошук запису в таблиці.

Після обрання цього пункту буде запропоновано обрати поле запису, по якому буде вестись пошук. Якщо це буде поле строкового типу, то буде запропоновано ввести строку, яку треба відшукати (або частину цієї строки). Якщо це буде числове поле, то треба буде ввести мінімальне значення і максимальне значення для відсортування полів.

Після обрання одного з цих пунктів буде проведено пошук SQL-запитом, отримані дані будуть виведені в консоль. Треба зазначити, що дані виводяться у вигляді таблички (відформатованої).

Треба зазначити, що аналогічна логіка застосовується до будь-якого пункту головного меню. Така ж сама послідовність дій передбачена для будь-якої таблиці.

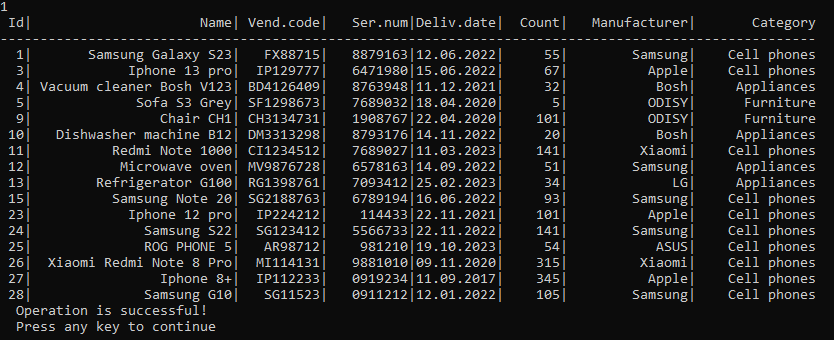




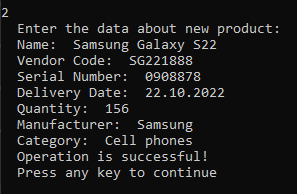
***Пункт №1***

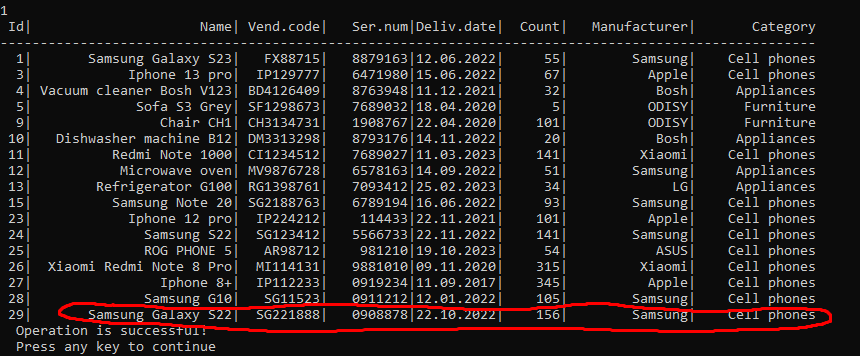
**Таблиця Product**

**Виведення всіх даних:**

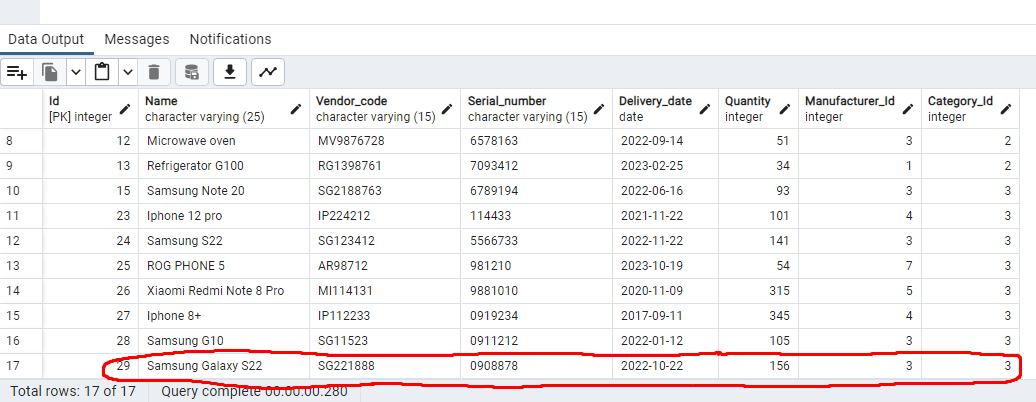


**Додавання нового продукту:**





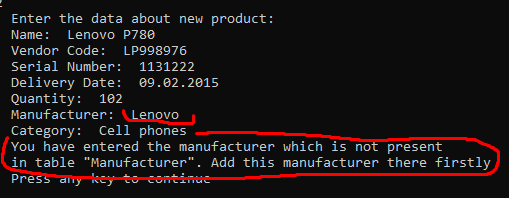
Можна також переконатись, що запис в табличку був дійсно доданий, якщо подивитись всі записи через PgAdmin4:

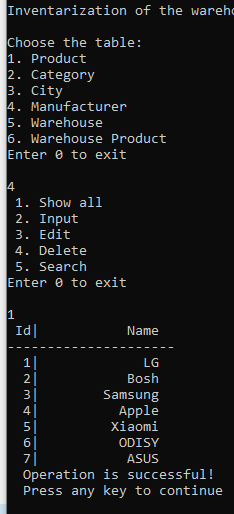


Хочу також зауважити, що програмно передбачено генерування нового ID для кожного запису. Для цього робиться SQL select запит всіх записів таблиці, потім поля ID кожного запису додаються в колекцію. Після чого обирається найбільше значення ID. Після чого береться це значення, збільшується на 1, а потім це значення буде використано під час операції INPUT мовою SQL.

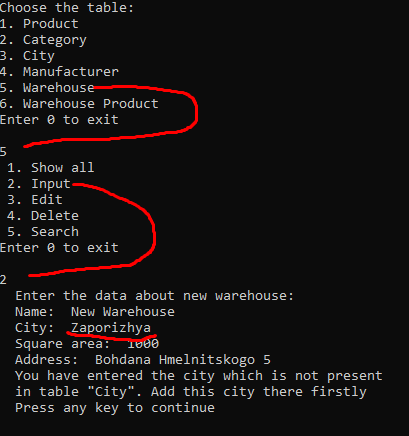
Також під час вставки проводиться валідація даних. Справа в тому, що, коли користувач обирає виробника та категорію товару, то він вводить виробника у вигляді строки, як і категорію. В таблиці ж за виробника та категорію відповідають поля, які є зовнішніми ключами: Manufacturer\_Id та Category\_Id. Отже, для того, що вставка здійснилася, треба вставити значення ID таблиць Manufacturer та Category. Для цього в методі вставки виконуються запити SQL та алгоритм, який визначає чи є взагалі виробник та категорія, яких ввів користувач, в цих таблицях, чи ні. Якщо їх нема, програма виведе в консоль повідомлення, що треба спочатку вставити цього виробника або цю категорію у відповідні таблиці.

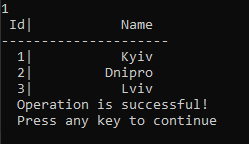
Це буде виглядати наступним чином:





Як можна помітити, виробника Lenovo в таблиці Manufacturer дійсно немає. Отже, вставка не відбулася, помилка не трапилась. Аналогічна валідація відбувається під час вставки даних в будь-яку іншу таблицю, в якій присутні зовнішні ключі.(наприклад, таблиця Warehouse, яка пов’язана із таблицею City). Продемонструємо це ще раз:

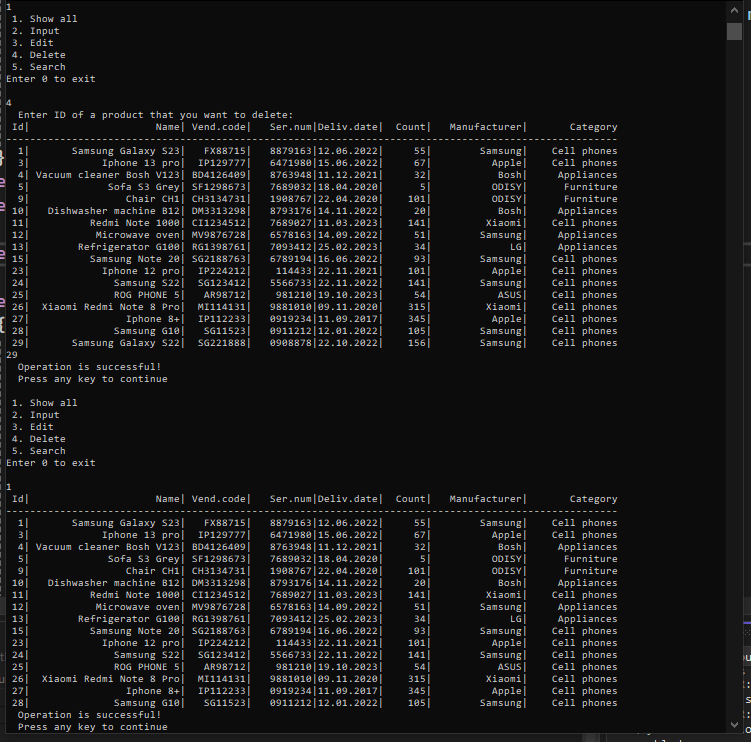




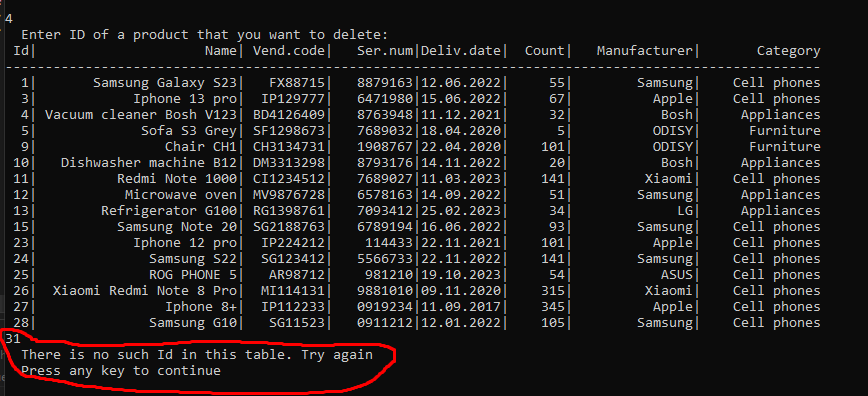
Як можна побачити, міста Запоріжжя дійсно немає в таблиці. Отже, вставка не відбулася.

**Видалення даних**

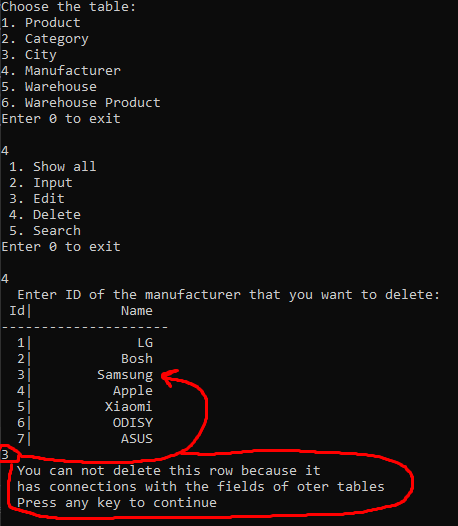
Видалимо із таблиці Product останній товар, який ми додали:

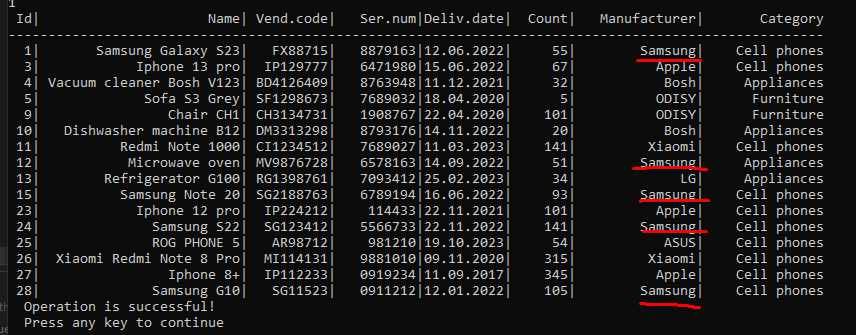


Як можна помітити, обраний продукт був видалений. Якщо ж спробувати видалити продукт, якого немає в таблиці, то програма передбачить цю спробу і видведе повідомлення, що це не є можливим. Помилка в програмі не відбудеться:



Оскільки таблиця Product є однією з головних таблиць, в неї алгоритм видалення і перевірки можливості видалення відрізняється від більш простих таблиць. Розглянемо, наприклад, таблицю Manufacturer, в якій є всього лише 2 поля: Id та Name. В цій таблиці немає зовнішніх ключів, проте на цю таблицю посилається таблиця Product. Отже, не можна видалити ті поля з таблиці Manufacturer, на які посилаються таблиці Product. Інакше трапиться помилка. Недопущення цієї ситуації передбачено програмою:

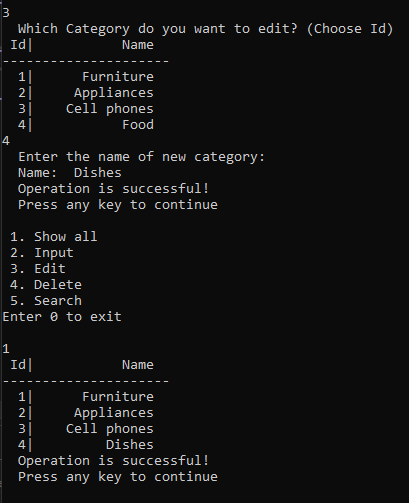




Як можна помітити, видалення виробника Samsung не відбулося, оскільки з цим виробником пов’язана велика кількість продуктів на складах.

**Редагування даних**

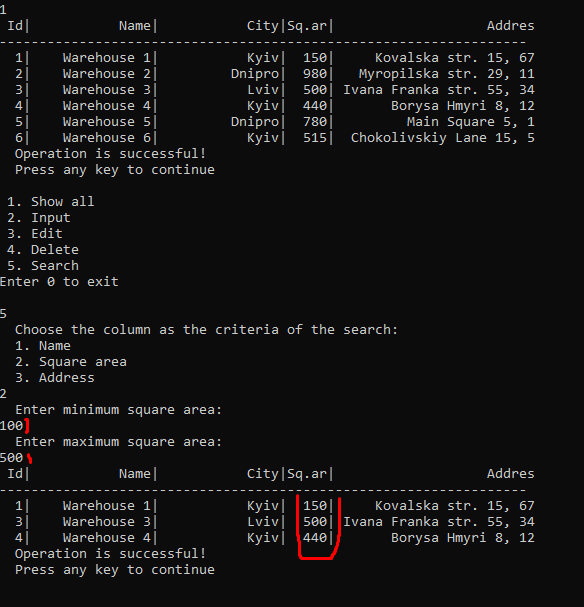
Редагування даних відбувається за схожим на видалення алгоритмом. Так само спочатку треба обрати індекс запису, який користувач хоче відредагувати, а вже потім вводити нові дані:



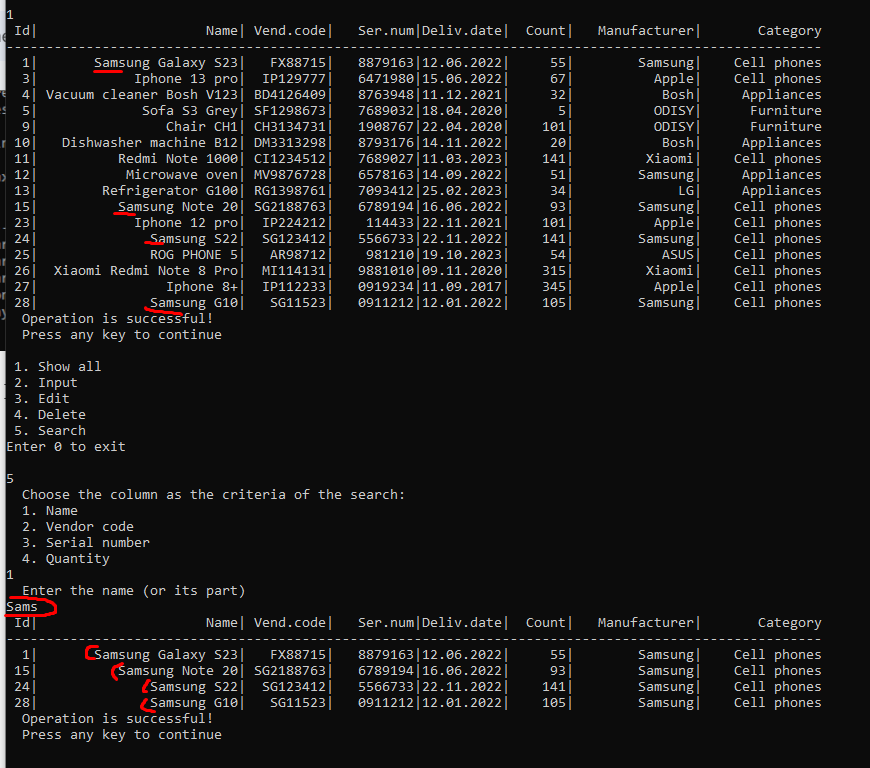
**Пошук даних:**

Пошук даних виконується після того, як користувач введе “ключ” пошуку. Для строкових полів таблиці ключем є, відповідно, строка, або підстрока, яку шукає користувач. Для числових значень користувач має задати мінімальну величину та максимальну. Приклади пошуку показано на наступних скріншотах:

Таблиця Warehouse:



Таблиця Product:



Як можна помітити, пошук відбувається.

***Пункт №2***

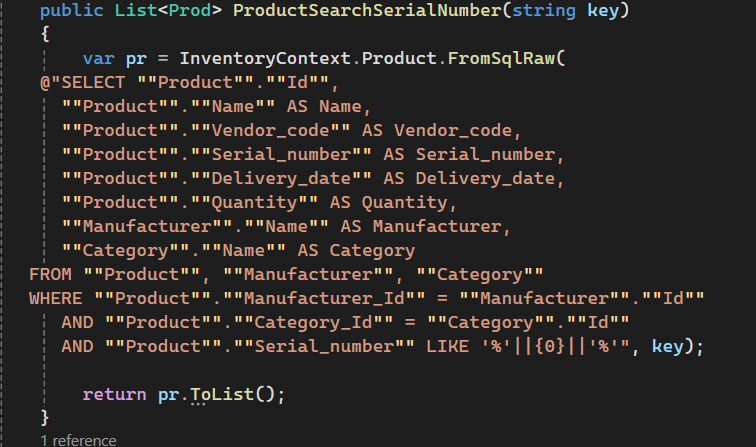
Генерація даних не була передбачена в даній РГР у зв’язку з відносно великим обсягом бази даних. На це пішло б дуже багато часу, якого наразі дуже мало лишилось. Проте, попри це, був зроблений великий акцент на валідацію даних в програмі. Програму майже неможливо вивести з ладу некоректними значеннями. Для тестування програми дані були введені вручну, ще під час виконання 1 лабораторної роботи.

Але все має бути справедливо, якщо за відсутність цього пункту бали будуть зняті, питань не буде. Погоджуюсь на це.

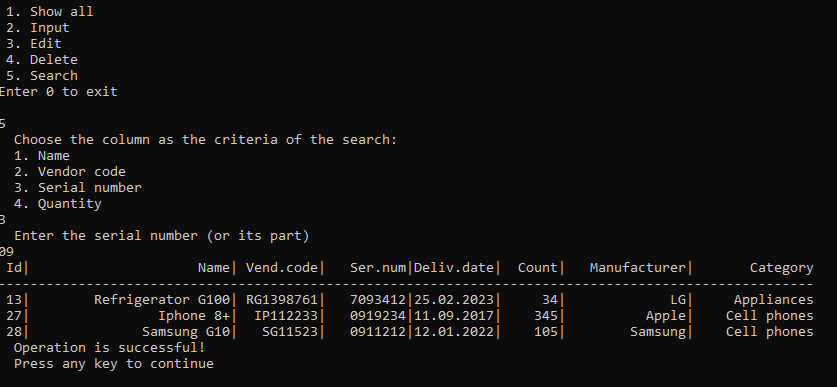
***Пункт №3***

**Приклад 1.**

Пошуковий запит в коді:

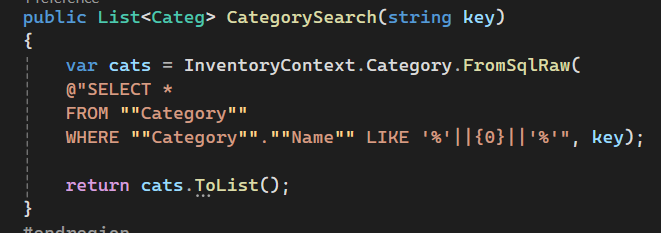


Пошук в консолі:

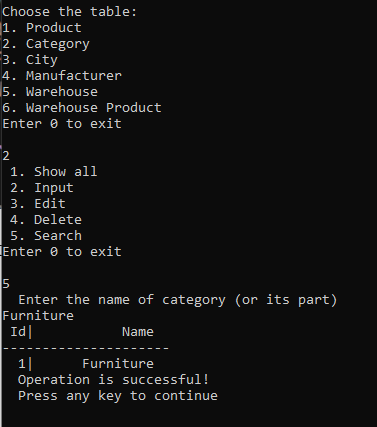


**Приклад 2**

Пошуковий запит в коді:

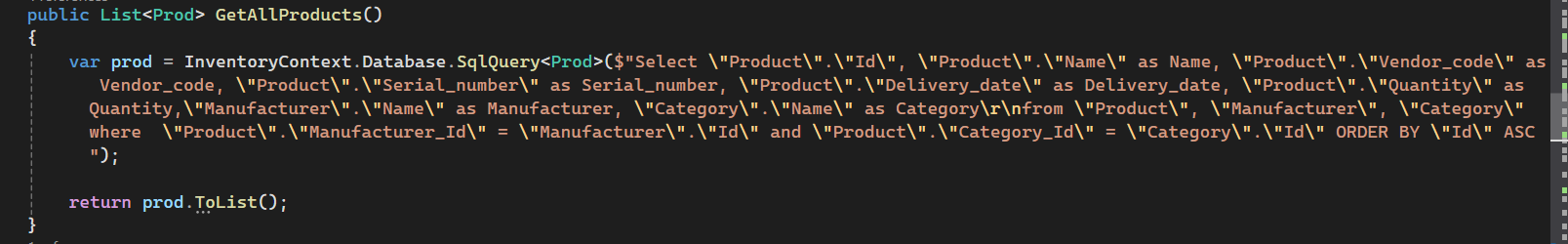


Запит в консолі

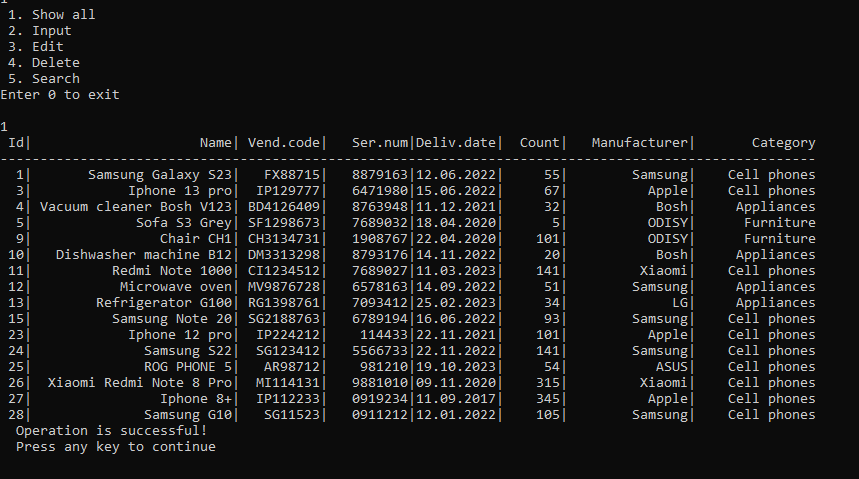


**Приклад 3**

Запит в коді:

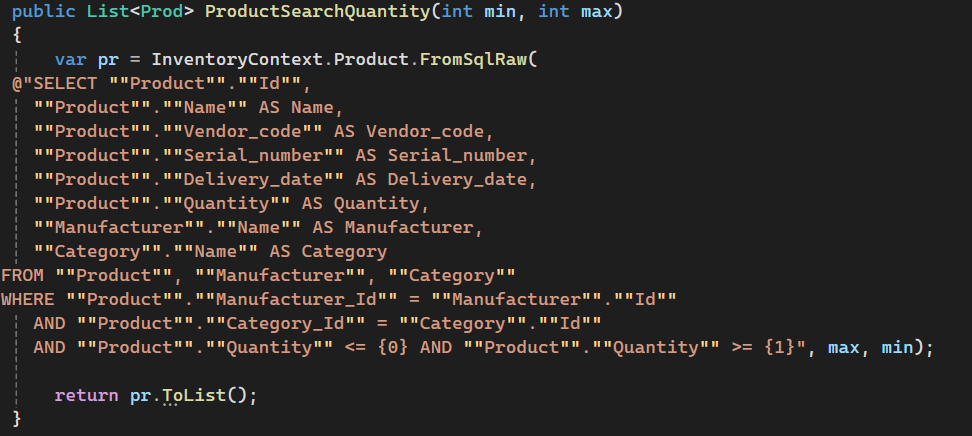
****

Запит в консолі:

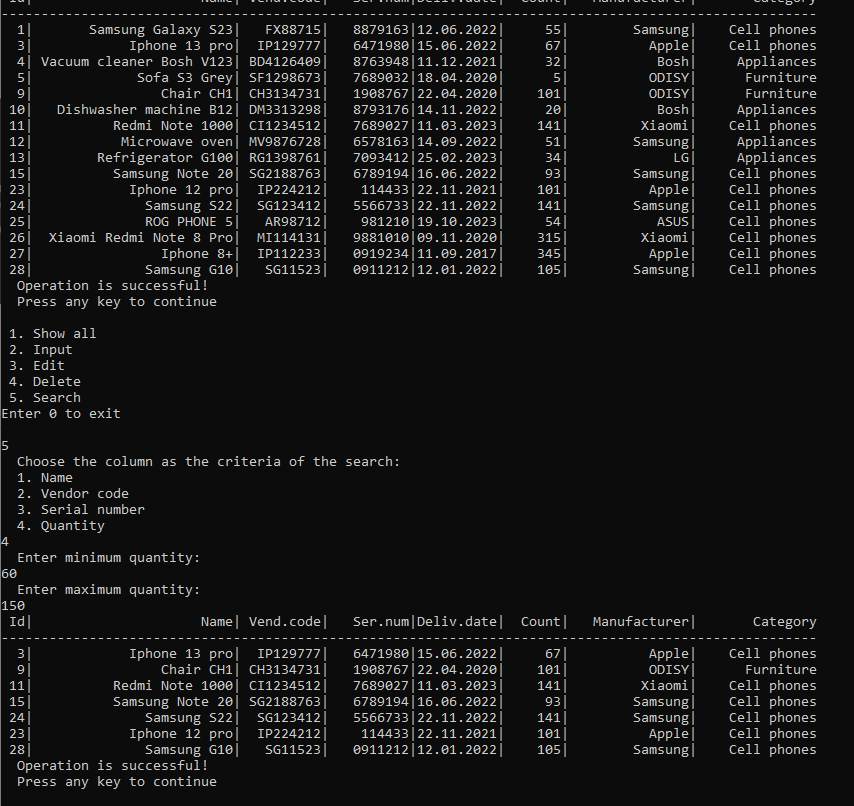


**Приклад 4**

Запит в коді:

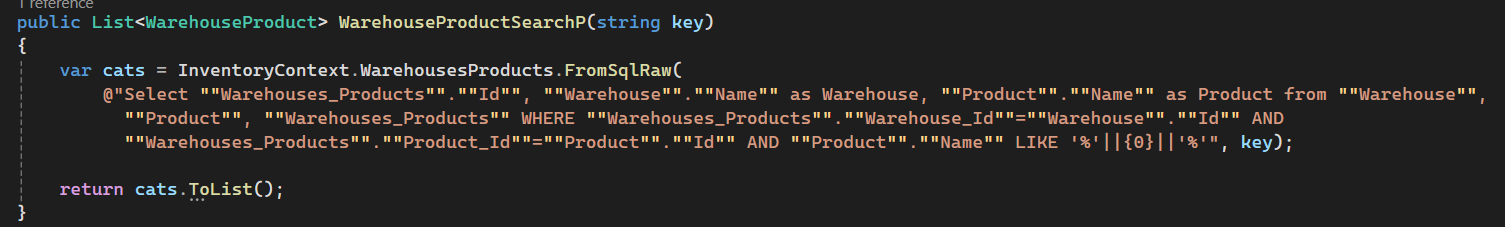


Запит в консолі:

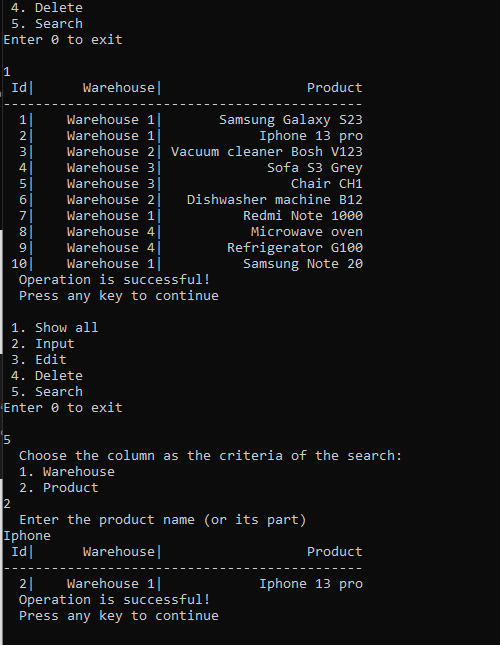


**Приклад 5**

Запит в коді:



Запит в консолі:



Всі пошуки були виконані через SQL-запити, хоча EntityFramework Core передбачає також можливість робити ці пошуки методами мови C#.

***Пункт №4***

Із програмним кодом класу Model\_ можна ознайомитись в файлі проекту. Оскільки цей файл вийшов великим, (700+ строк коду) в цьому пункті наведено структуру цього модуля:

**public class InventoryContext : DbContext** – клас, який створює з’єднання із базою даних та містить необхідні датасети, щоб отримувати дані з запитів.

**public class Model\_** – безпосередньо сам клас моделі. Його методи:

**Методи для роботи з таблицею Product:**

* public List<Prod> GetAllProducts() – метод отримання всіх продуктів
* public int InputProduct(Prod prod) – метод вставки нового продукту
* public int DeleteProduct(int id) – метод видалення продукту
* public int EditProduct(Prod prod, int id) – метод зміни даних про продукт
* public List<Prod> ProductSearchName(string key) – метод пошуку продукту за ім’ям
* public List<Prod> ProductSearchVendorCode(string key) – метод пошуку продукту за кодом продукту
* public List<Prod> ProductSearchSerialNumber(string key) – метод пошуку продукту за серійним кодом
* public List<Prod> ProductSearchQuantity(int min, int max) – метод пошуку продукту за кількістю

Далі всі методи мають аналогічний сенс, але для інших таблиць:

**Методи для роботи з таблицею Manufacturer:**

* public List<Manuf> GetAllManufacturers()
* public int InputManufacturer(Manuf man)
* public int EditManufacturer(Manuf manuf, int id)
* public int DeleteManufacturer(int id)
* public List<Manuf> ManufacturerSearch(string key)

**Методи для роботи з таблицею Category:**

* public List<Categ> GetAllCategories()
* public int InputCategory(Categ cat)
* public int EditCategory(Categ cat, int id)
* public int DeleteCategory(int id)
* public List<Categ> CategorySearch(string key)

**Методи для роботи з таблицею City:**

* public List<Cities> GetAllCities()
* public int InputCity(Cities cts)
* public int EditCity(Cities cat, int id)
* public int DeleteCity(int id)
* public List<Cities> CitiesSearch(string key)

**Методи для роботи з таблицею Warehouse:**

* public List<Warehouse> GetAllWarehouses()
* public int InputWarehouse(Warehouse warehouse)
* public int DeleteWarehouse(int id)
* public int EditWarehouse(Warehouse warehouse, int id)
* public List<Warehouse> WarehouseSearchName(string key)
* public List<Warehouse> WarehouseSearchAddress(string key)
* public List<Warehouse> WarehouseSearchQuantity(double min, double max)

**Методи для роботи з таблицею Warehouse\_Product:**

* public List<WarehouseProduct> GetAllWarehousesProducts()
* public int InputWarehouseProduct(WarehouseProduct prod)
* public int DeleteWarehouseProduct(int id)
* public int EditWarehouseProduct(WarehouseProduct prod, int id)
* public List<WarehouseProduct> WarehouseProductSearchW(string key)
* public List<WarehouseProduct> WarehouseProductSearchP(string key)

Програма написана з використанням архітектури MVC – Model, View, Controller.

Клас моделі відповідає за взаємодію з базою даних, за бізнес логіку. Тільки клас Model має доступ до БД.

Клас View відповідає за подачу даних користувачу. Тільки він має доступ до виводу даних користувачу.

Клас Controller композиціонує в собі View та Model, даючи їм команди. Цей клас відповідає за меню, послідовність команд, отримання даних від клієнта, передачу даних в Model, передачу даних з Model до View.

**Код програми:**

**file Program.cs**

**using** System.Collections.Generic;

**using** Microsoft.EntityFrameworkCore;

**using** System.Collections.Generic;

**using** System.Reflection.Metadata;

**using** Microsoft.EntityFrameworkCore.Metadata.Internal;

**using** RGR.DataModels;

**using** RGR;

**class** Program

{

**public** **static** **void** Main()

{

Controller\_ controller = **new** Controller\_();

controller.Run();

}

}

**File View\_.cs**

**using** Microsoft.EntityFrameworkCore;

**using** RGR.DataModels;

**using** System;

**using** System.Collections.Generic;

**using** System.ComponentModel;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**namespace** RGR

{

**public** **class** View\_

{

#region Base

**public** **void** Menu()

{

Console.WriteLine("Inventarization of the warehouse\n ");

Console.WriteLine("Choose the table: ");

**string**[] tables = **new** **string**[6] { "Product", "Category", "City", "Manufacturer", "Warehouse", "Warehouse Product" };

**for** (**int** i = 0; i < 6; i++)

{

Console.WriteLine(i + 1 + ". " + tables[i]);

}

Console.WriteLine("Enter 0 to exit\n ");

}

**public** **void** TableMenu()

{

**string**[] options = **new** **string**[5] { "Show all", "Input", "Edit", "Delete", "Search" };

**for** (**int** i = 0; i < 5; i++)

{

Console.WriteLine(" " + (i + 1).ToString() + ". " + options[i]);

}

Console.WriteLine("Enter 0 to exit\n ");

}

**public** **void** ErrorMessage(**string** message)

{

Console.WriteLine(" " + message);

Console.WriteLine(" Press any key to continue\n");

Console.ReadKey();

}

**public** **void** Message(**string** message)

{

Console.WriteLine(" " + message);

}

**public** **void** Done()

{

Console.WriteLine(" Operation is successful! \n Press any key to continue\n");

Console.ReadKey();

}

#endregion

#region Products

**public** **void** ShowAllProducts(List<Prod> Products)

{

Console.WriteLine("{0,3}|{1,25}|{2,10}|{3,10}|{4,10}|{5,7}|{6,15}|{7,15}", "Id", "Name", "Vend.code", "Ser.num", "Deliv.date", "Count", "Manufacturer", "Category");

Console.WriteLine("------------------------------------------------------------------------------------------------------");

**foreach** (Prod p **in** Products)

{

Console.WriteLine("{0,3}|{1,25}|{2,10}|{3,10}|{4,10}|{5,7}|{6,15}|{7,15}", p.Id, p.Name, p.Vendor\_code, p.Serial\_number, Convert.ToString(p.Delivery\_date), p.Quantity, p.Manufacturer, p.Category);

}

}

**public** Prod ProductInput()

{

Prod prod = **new** Prod();

Console.WriteLine(" Enter the data about new product: ");

Console.Write(" Name: ");

prod.Name = Console.ReadLine();

//

Console.Write(" Vendor Code: ");

prod.Vendor\_code = Console.ReadLine();

//

Console.Write(" Serial Number: ");

prod.Serial\_number = Console.ReadLine();

//

Console.Write(" Delivery Date: ");

prod.Delivery\_date = DateOnly.FromDateTime(Convert.ToDateTime(Console.ReadLine()));

//

Console.Write(" Quantity: ");

prod.Quantity = Convert.ToInt32(Console.ReadLine());

//

Console.Write(" Manufacturer: ");

prod.Manufacturer = Console.ReadLine();

//

Console.Write(" Category: ");

prod.Category = Console.ReadLine();

//

**return** prod;

}

**public** **void** ProductSearchMenu()

{

Console.WriteLine(" Choose the column as the criteria of the search:");

Console.WriteLine(" 1. Name");

Console.WriteLine(" 2. Vendor code");

Console.WriteLine(" 3. Serial number");

Console.WriteLine(" 4. Quantity");

}

**public** **string** ProductNameSearch()

{

Console.WriteLine(" Enter the name (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

**public** **string** ProductSearchVendorCode()

{

Console.WriteLine(" Enter the vendor code (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

**public** **string** ProductSearchSerialNumber()

{

Console.WriteLine(" Enter the serial number (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

#endregion

#region Manufacturers

**public** **void** ShowAllManufacturers(List<Manuf> manufs)

{

Console.WriteLine("{0,3}|{1,15}", "Id", "Name");

Console.WriteLine("---------------------");

**foreach** (Manuf m **in** manufs)

{

Console.WriteLine("{0,3}|{1,15}", m.Id, m.Name);

}

}

**public** Manuf ManufacturerInput()

{

Manuf man = **new** Manuf();

Console.WriteLine(" Enter the name of new manufacturer: ");

Console.Write(" Name: ");

man.Name = Console.ReadLine();

**return** man;

}

**public** **string** ManufacturerSearch()

{

Console.WriteLine(" Enter the name of manufacturer (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

#endregion

#region Categories

**public** **void** ShowAllCategories(List<Categ> cats)

{

Console.WriteLine("{0,3}|{1,15}", "Id", "Name");

Console.WriteLine("---------------------");

**foreach** (Categ c **in** cats)

{

Console.WriteLine("{0,3}|{1,15}", c.Id, c.Name);

}

}

**public** Categ CategoryInput()

{

Categ cat = **new** Categ();

Console.WriteLine(" Enter the name of new category: ");

Console.Write(" Name: ");

cat.Name = Console.ReadLine();

**return** cat;

}

**public** **string** CategorySearch()

{

Console.WriteLine(" Enter the name of category (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

#endregion

#region Cities

**public** **void** ShowAllCities(List<Cities> cts)

{

Console.WriteLine("{0,3}|{1,15}", "Id", "Name");

Console.WriteLine("---------------------");

**foreach** (Cities c **in** cts)

{

Console.WriteLine("{0,3}|{1,15}", c.Id, c.Name);

}

}

**public** Cities CityInput()

{

Cities cts = **new** Cities();

Console.WriteLine(" Enter the name of new city: ");

Console.Write(" Name: ");

cts.Name = Console.ReadLine();

**return** cts;

}

**public** **string** CitySearch()

{

Console.WriteLine(" Enter the name of city (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

#endregion

#region Warehouses

**public** **void** ShowAllWarehouses(List<Warehouse> warehouses)

{

Console.WriteLine("{0,3}|{1,15}|{2,15}|{3,5}|{4,25}", "Id", "Name", "City", "Sq.ar", "Addres");

Console.WriteLine("------------------------------------------------------------------");

**foreach** (Warehouse w **in** warehouses)

{

Console.WriteLine("{0,3}|{1,15}|{2,15}|{3,5}|{4,25}", w.Id, w.Name, w.City, w.Square\_area, w.Address);

}

}

**public** Warehouse WarehouseInput()

{

Warehouse w = **new** Warehouse();

Console.WriteLine(" Enter the data about new warehouse: ");

Console.Write(" Name: ");

w.Name = Console.ReadLine();

//

Console.Write(" City: ");

w.City = Console.ReadLine();

//

Console.Write(" Square area: ");

w.Square\_area = Convert.ToDouble(Console.ReadLine());

//

Console.Write(" Address: ");

w.Address = Console.ReadLine();

//

**return** w;

}

**public** **void** WarehouseSearchMenu()

{

Console.WriteLine(" Choose the column as the criteria of the search:");

Console.WriteLine(" 1. Name");

Console.WriteLine(" 2. Square area");

Console.WriteLine(" 3. Address");

}

**public** **string** WarehouseNameSearch()

{

Console.WriteLine(" Enter the name of warehouse (or its part)");

**string** name = Console.ReadLine();

**return** name;

}

**public** **string** WarehouseAddressSearch()

{

Console.WriteLine(" Enter the address of warehouse (or its part)");

**string** add = Console.ReadLine();

**return** add;

}

#endregion

#region WarehousesProducts

**public** **void** ShowAllWarehousesProducts(List<WarehouseProduct> wp)

{

Console.WriteLine("{0,3}|{1,15}|{2,25}", "Id", "Warehouse", "Product");

Console.WriteLine("---------------------------------------------");

**foreach** (WarehouseProduct w **in** wp)

{

Console.WriteLine("{0,3}|{1,15}|{2,25}", w.Id, w.Warehouse, w.Product);

}

}

**public** WarehouseProduct WarehouseProductInput()

{

WarehouseProduct w = **new** WarehouseProduct();

Console.WriteLine(" Enter the data about warehouse and corresponding product: ");

Console.Write(" Warehouse: ");

w.Warehouse = Console.ReadLine();

//

Console.Write(" Product: ");

w.Product = Console.ReadLine();

**return** w;

}

**public** **void** WarehouseProductSearchMenu()

{

Console.WriteLine(" Choose the column as the criteria of the search:");

Console.WriteLine(" 1. Warehouse");

Console.WriteLine(" 2. Product");

}

**public** **string** WarehouseProductSearchW()

{

Console.WriteLine(" Enter the warehouse name (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

**public** **string** WarehouseProductSearchP()

{

Console.WriteLine(" Enter the product name (or its part)");

**string** str = Console.ReadLine();

**return** str;

}

#endregion

}

}

**File Model.cs**

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Net.Http.Headers;

**using** System.Text;

**using** System.Threading.Tasks;

**using** Microsoft.EntityFrameworkCore;

**using** Microsoft.EntityFrameworkCore.Metadata;

**using** Microsoft.EntityFrameworkCore.Metadata.Internal;

**using** Npgsql;

**using** RGR.DataModels;

**namespace** RGR

{

**public** **class** InventoryContext : DbContext

{

**public** DbSet<Cities> City { **get**; **set**; }

**public** DbSet<Manuf> Manufacturer { **get**; **set**; }

**public** DbSet<Categ> Category { **get**; **set**; }

**public** DbSet<Prod> Product { **get**; **set**; }

**public** DbSet<WarehouseProduct> WarehousesProducts { **get**; **set**; }

**public** DbSet<Warehouse> Warehouses { **get**; **set**; }

//public DbSet<Post> Posts { get; set; }

**protected** **override** **void** OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseNpgsql(

"Host=localhost;Port=5432;Database=Inventory;Username=postgres;Password=011427223");

}

}

**public** **class** Model\_

{

InventoryContext InventoryContext { **get**; **set**; }

**public** Model\_()

{

InventoryContext = **new** InventoryContext();

}

#region ProductMethods

**public** List<Prod> GetAllProducts()

{

**var** prod = InventoryContext.Database.SqlQuery<Prod>($"Select \"Product\".\"Id\", \"Product\".\"Name\" as Name, \"Product\".\"Vendor\_code\" as Vendor\_code, \"Product\".\"Serial\_number\" as Serial\_number, \"Product\".\"Delivery\_date\" as Delivery\_date, \"Product\".\"Quantity\" as Quantity,\"Manufacturer\".\"Name\" as Manufacturer, \"Category\".\"Name\" as Category\r\nfrom \"Product\", \"Manufacturer\", \"Category\" where \"Product\".\"Manufacturer\_Id\" = \"Manufacturer\".\"Id\" and \"Product\".\"Category\_Id\" = \"Category\".\"Id\" ORDER BY \"Id\" ASC ");

**return** prod.ToList();

}

**public** **int** InputProduct(Prod prod)

{

**int** cat\_id = -1;

**int** man\_id = -1;

**var** cat = InventoryContext.Database.SqlQuery<Categ>($"Select \* from \"Category\"");

**foreach** (Categ c **in** cat)

{

**if** (c.Name == prod.Category)

cat\_id = c.Id;

}

**var** man = InventoryContext.Database.SqlQuery<Manuf>($"Select \* from \"Manufacturer\"");

**foreach** (Manuf m **in** man)

{

**if** (m.Name == prod.Manufacturer)

man\_id = m.Id;

}

**if** (cat\_id == -1)

**return** 3;

**if** (man\_id == -1)

**return** 2;

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Prod>($"Select \"Product\".\"Id\", \"Product\".\"Name\" as Name, \"Product\".\"Vendor\_code\" as Vendor\_code, \"Product\".\"Serial\_number\" as Serial\_number, \"Product\".\"Delivery\_date\" as Delivery\_date, \"Product\".\"Quantity\" as Quantity,\"Manufacturer\".\"Name\" as Manufacturer, \"Category\".\"Name\" as Category\r\nfrom \"Product\", \"Manufacturer\", \"Category\" where \"Product\".\"Manufacturer\_Id\" = \"Manufacturer\".\"Id\" and \"Product\".\"Category\_Id\" = \"Category\".\"Id\"");

**foreach** (Prod p **in** pr)

{

ids.Add(p.Id);

}

**int** maxId = ids.Max(id => id) + 1;

**var** sqlQuery = $"INSERT INTO \"Product\" VALUES ('{maxId}', '{prod.Name}', '{prod.Vendor\_code}','{prod.Serial\_number}','{prod.Delivery\_date}', {prod.Quantity}, {man\_id}, {cat\_id})";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**public** **int** DeleteProduct(**int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Prod>($"Select \"Product\".\"Id\", \"Product\".\"Name\" as Name, \"Product\".\"Vendor\_code\" as Vendor\_code, \"Product\".\"Serial\_number\" as Serial\_number, \"Product\".\"Delivery\_date\" as Delivery\_date, \"Product\".\"Quantity\" as Quantity,\"Manufacturer\".\"Name\" as Manufacturer, \"Category\".\"Name\" as Category\r\nfrom \"Product\", \"Manufacturer\", \"Category\" where \"Product\".\"Manufacturer\_Id\" = \"Manufacturer\".\"Id\" and \"Product\".\"Category\_Id\" = \"Category\".\"Id\"");

**foreach** (Prod p **in** pr)

{

ids.Add(p.Id);

}

**try**

{

**if** (ids.Contains(id))

{

**var** sqlQuery = $"DELETE from \"Product\" WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**catch**(Exception ex)

{

**return** 2;

}

}

**public** **int** EditProduct(Prod prod, **int** id)

{

**int** cat\_id = -1;

**int** man\_id = -1;

**var** cat = InventoryContext.Database.SqlQuery<Categ>($"Select \* from \"Category\"");

**foreach** (Categ c **in** cat)

{

**if** (c.Name == prod.Category)

cat\_id = c.Id;

}

**var** man = InventoryContext.Database.SqlQuery<Manuf>($"Select \* from \"Manufacturer\"");

**foreach** (Manuf m **in** man)

{

**if** (m.Name == prod.Manufacturer)

man\_id = m.Id;

}

**if** (cat\_id == -1)

**return** 3;

**if** (man\_id == -1)

**return** 2;

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Prod>($"Select \"Product\".\"Id\", \"Product\".\"Name\" as Name, \"Product\".\"Vendor\_code\" as Vendor\_code, \"Product\".\"Serial\_number\" as Serial\_number, \"Product\".\"Delivery\_date\" as Delivery\_date, \"Product\".\"Quantity\" as Quantity,\"Manufacturer\".\"Name\" as Manufacturer, \"Category\".\"Name\" as Category\r\nfrom \"Product\", \"Manufacturer\", \"Category\" where \"Product\".\"Manufacturer\_Id\" = \"Manufacturer\".\"Id\" and \"Product\".\"Category\_Id\" = \"Category\".\"Id\"");

**foreach** (Prod p **in** pr)

{

ids.Add(p.Id);

}

**if** (ids.Contains(id))

{

**var** sqlQuery = $"UPDATE \"Product\" SET \"Name\"='{prod.Name}', \"Vendor\_code\"='{prod.Vendor\_code}', \"Serial\_number\"='{prod.Serial\_number}', \"Delivery\_date\"='{prod.Delivery\_date}', \"Quantity\"={prod.Quantity}, \r\n\"Manufacturer\_Id\"={man\_id}, \"Category\_Id\"={cat\_id} where \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**public** List<Prod> ProductSearchName(**string** key)

{

**var** pr = InventoryContext.Product.FromSqlRaw(

@"SELECT ""Product"".""Id"",

""Product"".""Name"" AS Name,

""Product"".""Vendor\_code"" AS Vendor\_code,

""Product"".""Serial\_number"" AS Serial\_number,

""Product"".""Delivery\_date"" AS Delivery\_date,

""Product"".""Quantity"" AS Quantity,

""Manufacturer"".""Name"" AS Manufacturer,

""Category"".""Name"" AS Category

FROM ""Product"", ""Manufacturer"", ""Category""

WHERE ""Product"".""Manufacturer\_Id"" = ""Manufacturer"".""Id""

AND ""Product"".""Category\_Id"" = ""Category"".""Id""

AND ""Product"".""Name"" LIKE '%'||{0}||'%'", key);

**return** pr.ToList();

}

**public** List<Prod> ProductSearchVendorCode(**string** key)

{

**var** pr = InventoryContext.Product.FromSqlRaw(

@"SELECT ""Product"".""Id"",

""Product"".""Name"" AS Name,

""Product"".""Vendor\_code"" AS Vendor\_code,

""Product"".""Serial\_number"" AS Serial\_number,

""Product"".""Delivery\_date"" AS Delivery\_date,

""Product"".""Quantity"" AS Quantity,

""Manufacturer"".""Name"" AS Manufacturer,

""Category"".""Name"" AS Category

FROM ""Product"", ""Manufacturer"", ""Category""

WHERE ""Product"".""Manufacturer\_Id"" = ""Manufacturer"".""Id""

AND ""Product"".""Category\_Id"" = ""Category"".""Id""

AND ""Product"".""Vendor\_code"" LIKE '%'||{0}||'%'", key);

**return** pr.ToList();

}

**public** List<Prod> ProductSearchSerialNumber(**string** key)

{

**var** pr = InventoryContext.Product.FromSqlRaw(

@"SELECT ""Product"".""Id"",

""Product"".""Name"" AS Name,

""Product"".""Vendor\_code"" AS Vendor\_code,

""Product"".""Serial\_number"" AS Serial\_number,

""Product"".""Delivery\_date"" AS Delivery\_date,

""Product"".""Quantity"" AS Quantity,

""Manufacturer"".""Name"" AS Manufacturer,

""Category"".""Name"" AS Category

FROM ""Product"", ""Manufacturer"", ""Category""

WHERE ""Product"".""Manufacturer\_Id"" = ""Manufacturer"".""Id""

AND ""Product"".""Category\_Id"" = ""Category"".""Id""

AND ""Product"".""Serial\_number"" LIKE '%'||{0}||'%'", key);

**return** pr.ToList();

}

**public** List<Prod> ProductSearchQuantity(**int** min, **int** max)

{

**var** pr = InventoryContext.Product.FromSqlRaw(

@"SELECT ""Product"".""Id"",

""Product"".""Name"" AS Name,

""Product"".""Vendor\_code"" AS Vendor\_code,

""Product"".""Serial\_number"" AS Serial\_number,

""Product"".""Delivery\_date"" AS Delivery\_date,

""Product"".""Quantity"" AS Quantity,

""Manufacturer"".""Name"" AS Manufacturer,

""Category"".""Name"" AS Category

FROM ""Product"", ""Manufacturer"", ""Category""

WHERE ""Product"".""Manufacturer\_Id"" = ""Manufacturer"".""Id""

AND ""Product"".""Category\_Id"" = ""Category"".""Id""

AND ""Product"".""Quantity"" <= {0} AND ""Product"".""Quantity"" >= {1}", max, min);

**return** pr.ToList();

}

#endregion

#region ManufacturerMethods

**public** List<Manuf> GetAllManufacturers()

{

**var** prod = InventoryContext.Database.SqlQuery<Manuf>($"Select \* from \"Manufacturer\" ORDER BY \"Id\" ASC ");

**return** prod.ToList();

}

**public** **int** InputManufacturer(Manuf man)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Manuf>($"SELECT \* FROM \"Manufacturer\"");

**foreach** (Manuf m **in** pr)

{

**if** (m.Name == man.Name)

**return** 0;

}

**foreach** (Manuf m **in** pr)

{

ids.Add(m.Id);

}

**int** maxId = ids.Max(id => id) + 1;

**var** sqlQuery = $"INSERT INTO \"Manufacturer\" VALUES ('{maxId}', '{man.Name}')";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**public** **int** EditManufacturer(Manuf manuf, **int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** man = InventoryContext.Database.SqlQuery<Manuf>($"Select \* FROM \"Manufacturer\"");

**foreach** (Manuf m **in** man)

{

**if** (m.Name == manuf.Name)

**return** 2;

ids.Add(m.Id);

}

**if** (ids.Contains(id))

{

**var** sqlQuery = $"UPDATE \"Manufacturer\" SET \"Name\"='{manuf.Name}' WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**public** **int** DeleteManufacturer(**int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Manuf>($"SELECT \* FROM \"Manufacturer\"");

**foreach** (Manuf m **in** pr)

{

ids.Add(m.Id);

}

**try**

{

**if** (ids.Contains(id))

{

**var** sqlQuery = $"DELETE from \"Manufacturer\" WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**catch** (Exception ex)

{

**return** 2;

}

}

**public** List<Manuf> ManufacturerSearch(**string** key)

{

**var** cats = InventoryContext.Manufacturer.FromSqlRaw(

@"SELECT \*

FROM ""Manufacturer""

WHERE ""Manufacturer"".""Name"" LIKE '%'||{0}||'%'", key);

**return** cats.ToList();

}

#endregion

#region CategoryMethods

**public** List<Categ> GetAllCategories()

{

**var** cats = InventoryContext.Database.SqlQuery<Categ>($"Select \* from \"Category\" ORDER BY \"Id\" ASC ");

**return** cats.ToList();

}

**public** **int** InputCategory(Categ cat)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Categ>($"SELECT \* FROM \"Category\"");

**foreach** (Categ c **in** pr)

{

**if** (c.Name == cat.Name)

**return** 0;

}

**foreach** (Categ c **in** pr)

{

ids.Add(c.Id);

}

**int** maxId = ids.Max(id => id) + 1;

**var** sqlQuery = $"INSERT INTO \"Category\" VALUES ('{maxId}', '{cat.Name}')";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**public** **int** EditCategory(Categ cat, **int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** ct = InventoryContext.Database.SqlQuery<Categ>($"Select \* FROM \"Category\"");

**foreach** (Categ c **in** ct)

{

**if** (c.Name == cat.Name)

**return** 2;

ids.Add(c.Id);

}

**if** (ids.Contains(id))

{

**var** sqlQuery = $"UPDATE \"Category\" SET \"Name\"='{cat.Name}' WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**public** **int** DeleteCategory(**int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Categ>($"SELECT \* FROM \"Category\"");

**foreach** (Categ c **in** pr)

{

ids.Add(c.Id);

}

**try**

{

**if** (ids.Contains(id))

{

**var** sqlQuery = $"DELETE from \"Category\" WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**catch**(Exception ex)

{

**return** 2;

}

}

**public** List<Categ> CategorySearch(**string** key)

{

**var** cats = InventoryContext.Category.FromSqlRaw(

@"SELECT \*

FROM ""Category""

WHERE ""Category"".""Name"" LIKE '%'||{0}||'%'", key);

**return** cats.ToList();

}

#endregion

#region CityMethods

**public** List<Cities> GetAllCities()

{

**var** cts = InventoryContext.Database.SqlQuery<Cities>($"Select \* from \"City\" ORDER BY \"Id\" ASC ");

**return** cts.ToList();

}

**public** **int** InputCity(Cities cts)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Cities>($"SELECT \* FROM \"City\"");

**foreach** (Cities c **in** pr)

{

**if** (c.Name == cts.Name)

**return** 0;

}

**foreach** (Cities c **in** pr)

{

ids.Add(c.Id);

}

**int** maxId = ids.Max(id => id) + 1;

**var** sqlQuery = $"INSERT INTO \"City\" VALUES ('{maxId}', '{cts.Name}')";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**public** **int** EditCity(Cities cat, **int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** ct = InventoryContext.Database.SqlQuery<Cities>($"Select \* FROM \"City\"");

**foreach** (Cities c **in** ct)

{

**if** (c.Name == cat.Name)

**return** 2;

ids.Add(c.Id);

}

**if** (ids.Contains(id))

{

**var** sqlQuery = $"UPDATE \"City\" SET \"Name\"='{cat.Name}' WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**public** **int** DeleteCity(**int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Cities>($"SELECT \* FROM \"City\"");

**foreach** (Cities c **in** pr)

{

ids.Add(c.Id);

}

**try**

{

**if** (ids.Contains(id))

{

**var** sqlQuery = $"DELETE from \"City\" WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**catch** (Exception ex)

{

**return** 2;

}

}

**public** List<Cities> CitiesSearch(**string** key)

{

**var** cats = InventoryContext.City.FromSqlRaw(

@"SELECT \*

FROM ""City""

WHERE ""City"".""Name"" LIKE '%'||{0}||'%'", key);

**return** cats.ToList();

}

#endregion

#region WarehouseMethods

**public** List<Warehouse> GetAllWarehouses()

{

**var** wrh = InventoryContext.Database.SqlQuery<Warehouse>($"Select \"Warehouse\".\"Id\", \"Warehouse\".\"Name\" as Name, \"City\".\"Name\" as City, \"Warehouse\".\"Square\_area\" as Square\_area, \"Warehouse\".\"Address\" as Address from \"Warehouse\", \"City\" \nwhere \"Warehouse\".\"City\_Id\" = \"City\".\"Id\" \nORDER BY \"Id\" ASC ");

**return** wrh.ToList();

}

**public** **int** InputWarehouse(Warehouse warehouse)

{

**int** city\_id = -1;

**var** ct = InventoryContext.Database.SqlQuery<Cities>($"Select \* from \"City\" ORDER BY \"Id\" ASC ");

**foreach** (Cities c **in** ct)

{

**if** (c.Name == warehouse.City)

city\_id = c.Id;

}

**if** (city\_id == -1)

**return** 2;

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Warehouse>($"Select \"Warehouse\".\"Id\", \"Warehouse\".\"Name\" as Name, \"City\".\"Name\" as City, \"Warehouse\".\"Square\_area\" as Square\_area, \"Warehouse\".\"Address\" as Address from \"Warehouse\", \"City\" \nwhere \"Warehouse\".\"City\_Id\" = \"City\".\"Id\" \nORDER BY \"Id\" ASC");

**foreach** (Warehouse w **in** pr)

{

ids.Add(w.Id);

}

**int** maxId = ids.Max(id => id) + 1;

**var** sqlQuery = $"INSERT INTO \"Warehouse\" VALUES ('{maxId}', '{warehouse.Name}', '{city\_id}','{warehouse.Square\_area}','{warehouse.Address}')";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**public** **int** DeleteWarehouse(**int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Warehouse>($"Select \"Warehouse\".\"Id\", \"Warehouse\".\"Name\" as Name, \"City\".\"Name\" as City, \"Warehouse\".\"Square\_area\" as Square\_area, \"Warehouse\".\"Address\" as Address from \"Warehouse\", \"City\" \nwhere \"Warehouse\".\"City\_Id\" = \"City\".\"Id\" \nORDER BY \"Id\" ASC");

**foreach** (Warehouse w **in** pr)

{

ids.Add(w.Id);

}

**try**

{

**if** (ids.Contains(id))

{

**var** sqlQuery = $"DELETE from \"Warehouse\" WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**catch** (Exception ex)

{

**return** 2;

}

}

**public** **int** EditWarehouse(Warehouse warehouse, **int** id)

{

**int** city\_id = -1;

**var** ct = InventoryContext.Database.SqlQuery<Cities>($"Select \* from \"City\"");

**foreach** (Cities c **in** ct)

{

**if** (c.Name == warehouse.City)

city\_id = c.Id;

}

**if** (city\_id == -1)

**return** 2;

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<Warehouse>($"Select \"Warehouse\".\"Id\", \"Warehouse\".\"Name\" as Name, \"City\".\"Name\" as City, \"Warehouse\".\"Square\_area\" as Square\_area, \"Warehouse\".\"Address\" as Address from \"Warehouse\", \"City\" \nwhere \"Warehouse\".\"City\_Id\" = \"City\".\"Id\" \nORDER BY \"Id\" ASC");

**foreach** (Warehouse w **in** pr)

{

ids.Add(w.Id);

}

**if** (ids.Contains(id))

{

**var** sqlQuery = $"UPDATE \"Warehouse\" SET \"Name\"='{warehouse.Name}', \"City\_Id\"='{city\_id}', \"Square\_area\"='{warehouse.Square\_area}', \"Address\"='{warehouse.Address}' where \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**public** List<Warehouse> WarehouseSearchName(**string** key)

{

**var** cats = InventoryContext.Warehouses.FromSqlRaw(

@"SELECT ""Warehouse"".""Id"",

""Warehouse"".""Name"" as Name, ""City"".""Name"" as City,

""Warehouse"".""Square\_area"" as Square\_area,

""Warehouse"".""Address"" as Address FROM ""Warehouse"",

""City"" WHERE ""Warehouse"".""City\_Id"" = ""City"".""Id""

AND ""Warehouse"".""Name"" LIKE '%'||{0}||'%'", key);

**return** cats.ToList();

}

**public** List<Warehouse> WarehouseSearchAddress(**string** key)

{

**var** cats = InventoryContext.Warehouses.FromSqlRaw(

@"SELECT ""Warehouse"".""Id"",

""Warehouse"".""Name"" as Name, ""City"".""Name"" as City,

""Warehouse"".""Square\_area"" as Square\_area,

""Warehouse"".""Address"" as Address FROM ""Warehouse"",

""City"" WHERE ""Warehouse"".""City\_Id"" = ""City"".""Id""

AND ""Warehouse"".""Address"" LIKE '%'||{0}||'%'", key);

**return** cats.ToList();

}

**public** List<Warehouse> WarehouseSearchQuantity(**double** min, **double** max)

{

**var** cats = InventoryContext.Warehouses.FromSqlRaw(

@"SELECT ""Warehouse"".""Id"",

""Warehouse"".""Name"" as Name, ""City"".""Name"" as City,

""Warehouse"".""Square\_area"" as Square\_area,

""Warehouse"".""Address"" as Address FROM ""Warehouse"",

""City"" WHERE ""Warehouse"".""City\_Id"" = ""City"".""Id""

AND ""Warehouse"".""Square\_area"" <= {1}

AND ""Warehouse"".""Square\_area"" >= {0}", min, max);

**return** cats.ToList();

}

#endregion

#region WarehouseProductMethods

**public** List<WarehouseProduct> GetAllWarehousesProducts()

{

**var** prod = InventoryContext.Database.SqlQuery<WarehouseProduct>($"Select \"Warehouses\_Products\".\"Id\", \"Warehouse\".\"Name\" as Warehouse, \"Product\".\"Name\" as Product from \"Warehouse\", \"Product\", \"Warehouses\_Products\" where \"Warehouses\_Products\".\"Warehouse\_Id\"=\"Warehouse\".\"Id\" and \"Warehouses\_Products\".\"Product\_Id\"=\"Product\".\"Id\" ORDER BY \"Id\" ASC ");

**return** prod.ToList();

}

**public** **int** InputWarehouseProduct(WarehouseProduct prod)

{

**int** war\_id = -1;

**int** prod\_id = -1;

///

**var** war = InventoryContext.Database.SqlQuery<Warehouse>($"Select \"Warehouse\".\"Id\", \"Warehouse\".\"Name\" as Name, \"City\".\"Name\" as City, \"Warehouse\".\"Square\_area\" as Square\_area, \"Warehouse\".\"Address\" as Address from \"Warehouse\", \"City\" \nwhere \"Warehouse\".\"City\_Id\" = \"City\".\"Id\" \nORDER BY \"Id\" ASC ");

**foreach** (Warehouse w **in** war)

{

**if** (w.Name == prod.Warehouse)

war\_id = w.Id;

}

**var** prd = InventoryContext.Database.SqlQuery<Prod>($"Select \"Product\".\"Id\", \"Product\".\"Name\" as Name, \"Product\".\"Vendor\_code\" as Vendor\_code, \"Product\".\"Serial\_number\" as Serial\_number, \"Product\".\"Delivery\_date\" as Delivery\_date, \"Product\".\"Quantity\" as Quantity,\"Manufacturer\".\"Name\" as Manufacturer, \"Category\".\"Name\" as Category\r\nfrom \"Product\", \"Manufacturer\", \"Category\" where \"Product\".\"Manufacturer\_Id\" = \"Manufacturer\".\"Id\" and \"Product\".\"Category\_Id\" = \"Category\".\"Id\" ORDER BY \"Id\" ASC");

**foreach** (Prod p **in** prd)

{

**if** (p.Name == prod.Product)

prod\_id = p.Id;

}

**if** (war\_id == -1)

**return** 2;

**if** (prod\_id == -1)

**return** 3;

List<**int**> ids = **new** List<**int**>();

**var** wpr = InventoryContext.Database.SqlQuery<WarehouseProduct>($"Select \"Warehouses\_Products\".\"Id\", \"Warehouse\".\"Name\" as Warehouse, \"Product\".\"Name\" as Product from \"Warehouse\", \"Product\", \"Warehouses\_Products\" where \"Warehouses\_Products\".\"Warehouse\_Id\"=\"Warehouse\".\"Id\" and \"Warehouses\_Products\".\"Product\_Id\"=\"Product\".\"Id\" ORDER BY \"Id\" ASC");

**foreach** (WarehouseProduct wp **in** wpr)

{

ids.Add(wp.Id);

}

**int** maxId = ids.Max(id => id) + 1;

**var** sqlQuery = $"INSERT INTO \"Warehouses\_Products\" VALUES ('{maxId}', '{war\_id}', '{prod\_id}')";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**public** **int** DeleteWarehouseProduct(**int** id)

{

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<WarehouseProduct>($"Select \"Warehouses\_Products\".\"Id\", \"Warehouse\".\"Name\" as Warehouse, \"Product\".\"Name\" as Product from \"Warehouse\", \"Product\", \"Warehouses\_Products\" where \"Warehouses\_Products\".\"Warehouse\_Id\"=\"Warehouse\".\"Id\" and \"Warehouses\_Products\".\"Product\_Id\"=\"Product\".\"Id\" ORDER BY \"Id\" ASC");

**foreach** (WarehouseProduct p **in** pr)

{

ids.Add(p.Id);

}

**try**

{

**if** (ids.Contains(id))

{

**var** sqlQuery = $"DELETE from \"Warehouses\_Products\" WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**catch** (Exception ex)

{

**return** 2;

}

}

**public** **int** EditWarehouseProduct(WarehouseProduct prod, **int** id)

{

**int** war\_id = -1;

**int** prod\_id = -1;

///

**var** war = InventoryContext.Database.SqlQuery<Warehouse>($"Select \"Warehouse\".\"Id\", \"Warehouse\".\"Name\" as Name, \"City\".\"Name\" as City, \"Warehouse\".\"Square\_area\" as Square\_area, \"Warehouse\".\"Address\" as Address from \"Warehouse\", \"City\" \nwhere \"Warehouse\".\"City\_Id\" = \"City\".\"Id\" \nORDER BY \"Id\" ASC ");

**foreach** (Warehouse w **in** war)

{

**if** (w.Name == prod.Warehouse)

war\_id = w.Id;

}

**var** prd = InventoryContext.Database.SqlQuery<Prod>($"Select \"Product\".\"Id\", \"Product\".\"Name\" as Name, \"Product\".\"Vendor\_code\" as Vendor\_code, \"Product\".\"Serial\_number\" as Serial\_number, \"Product\".\"Delivery\_date\" as Delivery\_date, \"Product\".\"Quantity\" as Quantity,\"Manufacturer\".\"Name\" as Manufacturer, \"Category\".\"Name\" as Category\r\nfrom \"Product\", \"Manufacturer\", \"Category\" where \"Product\".\"Manufacturer\_Id\" = \"Manufacturer\".\"Id\" and \"Product\".\"Category\_Id\" = \"Category\".\"Id\" ORDER BY \"Id\" ASC");

**foreach** (Prod p **in** prd)

{

**if** (p.Name == prod.Product)

prod\_id = p.Id;

}

**if** (war\_id == -1)

**return** 2;

**if** (prod\_id == -1)

**return** 3;

List<**int**> ids = **new** List<**int**>();

**var** pr = InventoryContext.Database.SqlQuery<WarehouseProduct>($"Select \"Warehouses\_Products\".\"Id\", \"Warehouse\".\"Name\" as Warehouse, \"Product\".\"Name\" as Product from \"Warehouse\", \"Product\", \"Warehouses\_Products\" where \"Warehouses\_Products\".\"Warehouse\_Id\"=\"Warehouse\".\"Id\" and \"Warehouses\_Products\".\"Product\_Id\"=\"Product\".\"Id\" ORDER BY \"Id\" ASC");

**foreach** (WarehouseProduct p **in** pr)

{

ids.Add(p.Id);

}

**if** (ids.Contains(id))

{

**var** sqlQuery = $"UPDATE \"Warehouses\_Products\" SET \"Warehouse\_Id\"='{war\_id}', \"Product\_Id\"='{prod\_id}' WHERE \"Id\" = {id};";

**int** numberOfRowInserted = InventoryContext.Database.ExecuteSqlRaw(sqlQuery);

**return** 1;

}

**else** **return** 0;

}

**public** List<WarehouseProduct> WarehouseProductSearchW(**string** key)

{

**var** cats = InventoryContext.WarehousesProducts.FromSqlRaw(

@"Select ""Warehouses\_Products"".""Id"", ""Warehouse"".""Name"" as Warehouse, ""Product"".""Name"" as Product from ""Warehouse"", ""Product"", ""Warehouses\_Products"" WHERE ""Warehouses\_Products"".""Warehouse\_Id""=""Warehouse"".""Id"" AND ""Warehouses\_Products"".""Product\_Id""=""Product"".""Id"" AND ""Warehouse"".""Name"" LIKE '%'||{0}||'%'", key);

**return** cats.ToList();

}

**public** List<WarehouseProduct> WarehouseProductSearchP(**string** key)

{

**var** cats = InventoryContext.WarehousesProducts.FromSqlRaw(

@"Select ""Warehouses\_Products"".""Id"", ""Warehouse"".""Name"" as Warehouse, ""Product"".""Name"" as Product from ""Warehouse"", ""Product"", ""Warehouses\_Products"" WHERE ""Warehouses\_Products"".""Warehouse\_Id""=""Warehouse"".""Id"" AND ""Warehouses\_Products"".""Product\_Id""=""Product"".""Id"" AND ""Product"".""Name"" LIKE '%'||{0}||'%'", key);

**return** cats.ToList();

}

#endregion

}

}

**File View\_cs**

**using Microsoft.EntityFrameworkCore;**

**using RGR.DataModels;**

**using System;**

**using System.Collections.Generic;**

**using System.ComponentModel;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace RGR**

**{**

**public class View\_**

**{**

**#region Base**

**public void Menu()**

**{**

**Console.WriteLine("Inventarization of the warehouse\n ");**

**Console.WriteLine("Choose the table: ");**

**string[] tables = new string[6] { "Product", "Category", "City", "Manufacturer", "Warehouse", "Warehouse Product" };**

**for (int i = 0; i < 6; i++)**

**{**

**Console.WriteLine(i + 1 + ". " + tables[i]);**

**}**

**Console.WriteLine("Enter 0 to exit\n ");**

**}**

**public void TableMenu()**

**{**

**string[] options = new string[5] { "Show all", "Input", "Edit", "Delete", "Search" };**

**for (int i = 0; i < 5; i++)**

**{**

**Console.WriteLine(" " + (i + 1).ToString() + ". " + options[i]);**

**}**

**Console.WriteLine("Enter 0 to exit\n ");**

**}**

**public void ErrorMessage(string message)**

**{**

**Console.WriteLine(" " + message);**

**Console.WriteLine(" Press any key to continue\n");**

**Console.ReadKey();**

**}**

**public void Message(string message)**

**{**

**Console.WriteLine(" " + message);**

**}**

**public void Done()**

**{**

**Console.WriteLine(" Operation is successful! \n Press any key to continue\n");**

**Console.ReadKey();**

**}**

**#endregion**

**#region Products**

**public void ShowAllProducts(List<Prod> Products)**

**{**

**Console.WriteLine("{0,3}|{1,25}|{2,10}|{3,10}|{4,10}|{5,7}|{6,15}|{7,15}", "Id", "Name", "Vend.code", "Ser.num", "Deliv.date", "Count", "Manufacturer", "Category");**

**Console.WriteLine("------------------------------------------------------------------------------------------------------");**

**foreach (Prod p in Products)**

**{**

**Console.WriteLine("{0,3}|{1,25}|{2,10}|{3,10}|{4,10}|{5,7}|{6,15}|{7,15}", p.Id, p.Name, p.Vendor\_code, p.Serial\_number, Convert.ToString(p.Delivery\_date), p.Quantity, p.Manufacturer, p.Category);**

**}**

**}**

**public Prod ProductInput()**

**{**

**Prod prod = new Prod();**

**Console.WriteLine(" Enter the data about new product: ");**

**Console.Write(" Name: ");**

**prod.Name = Console.ReadLine();**

**//**

**Console.Write(" Vendor Code: ");**

**prod.Vendor\_code = Console.ReadLine();**

**//**

**Console.Write(" Serial Number: ");**

**prod.Serial\_number = Console.ReadLine();**

**//**

**Console.Write(" Delivery Date: ");**

**prod.Delivery\_date = DateOnly.FromDateTime(Convert.ToDateTime(Console.ReadLine()));**

**//**

**Console.Write(" Quantity: ");**

**prod.Quantity = Convert.ToInt32(Console.ReadLine());**

**//**

**Console.Write(" Manufacturer: ");**

**prod.Manufacturer = Console.ReadLine();**

**//**

**Console.Write(" Category: ");**

**prod.Category = Console.ReadLine();**

**//**

**return prod;**

**}**

**public void ProductSearchMenu()**

**{**

**Console.WriteLine(" Choose the column as the criteria of the search:");**

**Console.WriteLine(" 1. Name");**

**Console.WriteLine(" 2. Vendor code");**

**Console.WriteLine(" 3. Serial number");**

**Console.WriteLine(" 4. Quantity");**

**}**

**public string ProductNameSearch()**

**{**

**Console.WriteLine(" Enter the name (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**public string ProductSearchVendorCode()**

**{**

**Console.WriteLine(" Enter the vendor code (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**public string ProductSearchSerialNumber()**

**{**

**Console.WriteLine(" Enter the serial number (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**#endregion**

**#region Manufacturers**

**public void ShowAllManufacturers(List<Manuf> manufs)**

**{**

**Console.WriteLine("{0,3}|{1,15}", "Id", "Name");**

**Console.WriteLine("---------------------");**

**foreach (Manuf m in manufs)**

**{**

**Console.WriteLine("{0,3}|{1,15}", m.Id, m.Name);**

**}**

**}**

**public Manuf ManufacturerInput()**

**{**

**Manuf man = new Manuf();**

**Console.WriteLine(" Enter the name of new manufacturer: ");**

**Console.Write(" Name: ");**

**man.Name = Console.ReadLine();**

**return man;**

**}**

**public string ManufacturerSearch()**

**{**

**Console.WriteLine(" Enter the name of manufacturer (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**#endregion**

**#region Categories**

**public void ShowAllCategories(List<Categ> cats)**

**{**

**Console.WriteLine("{0,3}|{1,15}", "Id", "Name");**

**Console.WriteLine("---------------------");**

**foreach (Categ c in cats)**

**{**

**Console.WriteLine("{0,3}|{1,15}", c.Id, c.Name);**

**}**

**}**

**public Categ CategoryInput()**

**{**

**Categ cat = new Categ();**

**Console.WriteLine(" Enter the name of new category: ");**

**Console.Write(" Name: ");**

**cat.Name = Console.ReadLine();**

**return cat;**

**}**

**public string CategorySearch()**

**{**

**Console.WriteLine(" Enter the name of category (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**#endregion**

**#region Cities**

**public void ShowAllCities(List<Cities> cts)**

**{**

**Console.WriteLine("{0,3}|{1,15}", "Id", "Name");**

**Console.WriteLine("---------------------");**

**foreach (Cities c in cts)**

**{**

**Console.WriteLine("{0,3}|{1,15}", c.Id, c.Name);**

**}**

**}**

**public Cities CityInput()**

**{**

**Cities cts = new Cities();**

**Console.WriteLine(" Enter the name of new city: ");**

**Console.Write(" Name: ");**

**cts.Name = Console.ReadLine();**

**return cts;**

**}**

**public string CitySearch()**

**{**

**Console.WriteLine(" Enter the name of city (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**#endregion**

**#region Warehouses**

**public void ShowAllWarehouses(List<Warehouse> warehouses)**

**{**

**Console.WriteLine("{0,3}|{1,15}|{2,15}|{3,5}|{4,25}", "Id", "Name", "City", "Sq.ar", "Addres");**

**Console.WriteLine("------------------------------------------------------------------");**

**foreach (Warehouse w in warehouses)**

**{**

**Console.WriteLine("{0,3}|{1,15}|{2,15}|{3,5}|{4,25}", w.Id, w.Name, w.City, w.Square\_area, w.Address);**

**}**

**}**

**public Warehouse WarehouseInput()**

**{**

**Warehouse w = new Warehouse();**

**Console.WriteLine(" Enter the data about new warehouse: ");**

**Console.Write(" Name: ");**

**w.Name = Console.ReadLine();**

**//**

**Console.Write(" City: ");**

**w.City = Console.ReadLine();**

**//**

**Console.Write(" Square area: ");**

**w.Square\_area = Convert.ToDouble(Console.ReadLine());**

**//**

**Console.Write(" Address: ");**

**w.Address = Console.ReadLine();**

**//**

**return w;**

**}**

**public void WarehouseSearchMenu()**

**{**

**Console.WriteLine(" Choose the column as the criteria of the search:");**

**Console.WriteLine(" 1. Name");**

**Console.WriteLine(" 2. Square area");**

**Console.WriteLine(" 3. Address");**

**}**

**public string WarehouseNameSearch()**

**{**

**Console.WriteLine(" Enter the name of warehouse (or its part)");**

**string name = Console.ReadLine();**

**return name;**

**}**

**public string WarehouseAddressSearch()**

**{**

**Console.WriteLine(" Enter the address of warehouse (or its part)");**

**string add = Console.ReadLine();**

**return add;**

**}**

**#endregion**

**#region WarehousesProducts**

**public void ShowAllWarehousesProducts(List<WarehouseProduct> wp)**

**{**

**Console.WriteLine("{0,3}|{1,15}|{2,25}", "Id", "Warehouse", "Product");**

**Console.WriteLine("---------------------------------------------");**

**foreach (WarehouseProduct w in wp)**

**{**

**Console.WriteLine("{0,3}|{1,15}|{2,25}", w.Id, w.Warehouse, w.Product);**

**}**

**}**

**public WarehouseProduct WarehouseProductInput()**

**{**

**WarehouseProduct w = new WarehouseProduct();**

**Console.WriteLine(" Enter the data about warehouse and corresponding product: ");**

**Console.Write(" Warehouse: ");**

**w.Warehouse = Console.ReadLine();**

**//**

**Console.Write(" Product: ");**

**w.Product = Console.ReadLine();**

**return w;**

**}**

**public void WarehouseProductSearchMenu()**

**{**

**Console.WriteLine(" Choose the column as the criteria of the search:");**

**Console.WriteLine(" 1. Warehouse");**

**Console.WriteLine(" 2. Product");**

**}**

**public string WarehouseProductSearchW()**

**{**

**Console.WriteLine(" Enter the warehouse name (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**public string WarehouseProductSearchP()**

**{**

**Console.WriteLine(" Enter the product name (or its part)");**

**string str = Console.ReadLine();**

**return str;**

**}**

**#endregion**

**}**

**}**

**File Categ.cs :**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace RGR.DataModels**

**{**

**public class Categ**

**{**

**public int Id { get; set; }**

**public string Name { get; set; }**

**}**

**}**

**File Cities.cs :**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace RGR.DataModels**

**{**

**public class Categ**

**{**

**public int Id { get; set; }**

**public string Name { get; set; }**

**}**

**}**

**File Manuf.cs :**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace RGR.DataModels**

**{**

**public class Manuf**

**{**

**public int Id { get; set; }**

**public string Name { get; set; }**

**}**

**}**

**File Prod.cs :**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace RGR.DataModels**

**{**

**public class Prod**

**{**

**public int Id { get; set; }**

**public string Name { get; set; }**

**public string Vendor\_code { get; set; }**

**public string Serial\_number { get; set; }**

**public DateOnly Delivery\_date { get; set; }**

**public int Quantity { get; set; }**

**public string Manufacturer { get; set; }**

**public string Category { get; set; }**

**}**

**}**

**File Warehouse.cs:**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace RGR.DataModels**

**{**

**public class Warehouse**

**{**

**public int Id { get; set; }**

**public string Name { get; set; }**

**public string City { get; set; }**

**public double Square\_area { get; set; }**

**public string Address { get; set; }**

**}**

**}**

**File WarehouseProduct.cs:**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace RGR.DataModels**

**{**

**public class WarehouseProduct**

**{**

**public int Id { get; set; }**

**public string Warehouse {get; set;}**

**public string Product { get; set; }**

**}**

**}**