

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

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

**Розрахунково-графічна робота**

з дисципліни **Бази даних і засоби управління**

*на тему: “* *Створення додатку бази даних, орієнтованого на взаємодію з СУБД PostgreSQL”*

Виконала: студентка ІІI курсу

ФПМ групи КВ-11

Зубенко Марія

Перевірив:

Київ – 2023

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

*Завдання* роботи полягає у наступному:

1. Реалізувати функції 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).

**Виконання роботи**

**Логічна модель предметної області «Магазин»**

Логічну модель (схему бази даних) наведено на рисунку 1.

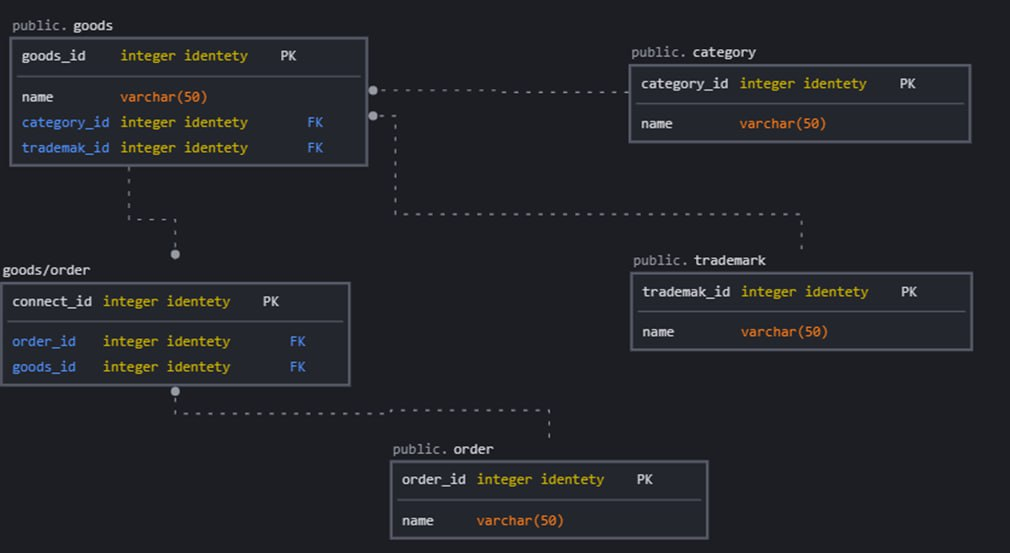


Рисунок 1 -Логічна модель бази даних

Зміни у порівнянні з першою лабораторною роботою відсутні. (інструмент: sqldbm.com)

**Середовище та компоненти розробки**

Для розробки використовувалась мова програмування С#, середовище розробки Visual Studio, а також стороння бібліотека, що надає API для доступу до PostgreSQL – Npgsql.

**Шаблон проектування**

MVC - шаблон проектування, який використаний у програмі.

Model – представляє клас, що описує логіку використовуваних даних. Згідно компоненту моделі, у моїй програмі відповідають всі компоненти які знаходяться у папці Models.

View – в нашому випадку консольний інтерфейс з яким буде взаємодіяти наш користувач. Згідно компоненту представлення, то їй відповідають такі компоненти, згідно яким користувач бачить необхідні дані, що є представленням даних у вигляді консольного інтерфейсу.

Controller – представляє клас, що забезпечує зв'язок між користувачем і системою, поданням і сховищем даних. Він отримує вводяться користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді подання.

**Структура програми та її опис**



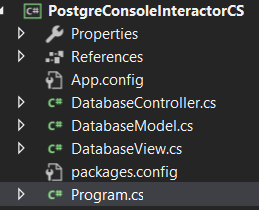


Рисунок 2 - Структура програми

Програма умовно розділена на 4 модулі: файл DatabaseController.cs, файл DatabaseModel.cs та файл DatabaseView.cs та головний файл Program.cs

Класи, як видно з їх назв, повністю відповідають використаному патерну MVC.

У файлі DatabaseModel описаний клас моделі, що займається регулюванням підключення до бази даних, та виконанням запитів до неї.

У файлі DatabaseController описаний інтерфейс взаємодії з користувачем, запит бажаної дії, виконання пошуку, тощо.

У файлі DatabaseView описаний клас, що виводить результати виконання тієї чи іншої дії на екран консолі.

**Структура меню програми**

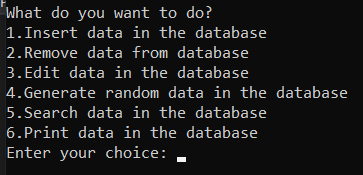


Рисунок 3 - Меню програми

Меню користувача складається з шести пунктів (Рисунок 3).

Перший пункт пропонує введення даних в таблицю

Другий пункт пропонує видалення даних з таблиці

Третій пункт пропонує оновлення даних в таблиці

Четвертий пункт пропонує генерування даних в таблиці

П’ятий пункт пропонує пошук даних у таблиці

Шостий пункт пропонує отримання імен та типів стовпчиків таблиці

**Фрагменти програм внесення, редагування та вилучення даних у базі даних**

Фрагмент програми для введення даних в таблицю:

public void Insert() {

List<string> tables = GetTables("insert");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

List<string> columns = GetColumns(tables[selected\_index]);

List<string> values = new List<string>();

foreach (string column in columns) {

Console.Write("Enter value for column - " + column + " - ");

values.Add(Console.ReadLine());

}

string insert\_query = "INSERT INTO " + tables[selected\_index] + " (" + ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";

NpgsqlCommand insert\_command = new NpgsqlCommand(insert\_query, connection);

Console.WriteLine(insert\_query);

insert\_command.ExecuteNonQuery();

Console.WriteLine("Data succesfull added!");

}

Фрагмент програми для видалення даних з таблиці:

public void Delete() {

List<string> tables = GetTables("delete");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

Console.WriteLine("Input row id for delete");

int row\_id = int.Parse(Console.ReadLine());

NpgsqlCommand delete\_command = new NpgsqlCommand("DELETE FROM " + tables[selected\_index] + " WHERE " + tables[selected\_index] + "\_id = " + row\_id, connection);

delete\_command.ExecuteNonQuery();

Console.WriteLine("Data succesfull deleted!");

}

Фрагмент програми для оновлення даних в таблиці:

public void Update() {

List<string> tables = GetTables("update");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

List<string> columns = GetColumns(tables[selected\_index]);

List<string> values = new List<string>();

Console.WriteLine("Input row id for update");

int row\_id = int.Parse(Console.ReadLine());

foreach (string column in columns) {

Console.Write("Enter new value for column - " + column + " - ");

values.Add(Console.ReadLine());

}

string update\_query = "UPDATE " + tables[selected\_index] + " SET " + UpdatePartialString(columns, values) + " WHERE " + tables[selected\_index] + "\_id = " + row\_id;

NpgsqlCommand update\_command = new NpgsqlCommand(update\_query, connection);

update\_command.ExecuteNonQuery();

Console.WriteLine("Data succesfull updated! " + update\_query);

}

Фрагмент програми для генерування даних в таблиці:

public void Generate() {

List<string> tables = GetTables("generate");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

Console.WriteLine("Input rows count for generate");

int rows\_count = int.Parse(Console.ReadLine());

List<string> columns = GetColumns(tables[selected\_index]);

Random random = new Random();

for (int i = 0; i < rows\_count; i++) {

while (true) {

try {

List<string> values = new List<string>();

foreach (string column in columns) {

values.Add(random.Next(0, 10).ToString());

}

//values[values.Count - 1] = "1";

string insert\_query = "INSERT INTO " + tables[selected\_index] + " (" + ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";

NpgsqlCommand insert\_command = new NpgsqlCommand(insert\_query, connection);

insert\_command.ExecuteNonQuery();

break;

}

catch (Exception ex) {

//Console.WriteLine(ex.Message);

}

}

}

Console.WriteLine("Generated was succesfull!");

}

Фрагмент програми для пошуку даних у таблиці:

public void Search() {

Console.WriteLine("Choose what to search:");

Console.WriteLine("1 - select goods by trademark\_id");

Console.WriteLine("2 - select goods by categoty\_id");

Console.WriteLine("3 - select name by trademark\_id");

Console.Write("Your choice - ");

int search = int.Parse(Console.ReadLine());

switch (search) {

case 1:

Console.Write("Input trademark\_id - ");

string trademark\_id = Console.ReadLine();

NpgsqlCommand command1 = new NpgsqlCommand("SELECT name FROM category WHERE department\_id = " + trademark\_id, connection);

NpgsqlDataReader reader1 = command1.ExecuteReader();

List<string> data1 = new List<string>();

while (reader1.Read()) {

data1.Add(reader1.GetValue(0).ToString());

}

PrintRow(data1);

reader1.Close();

break;

case 2:

Console.Write("Input category\_id - ");

string category\_id = Console.ReadLine();

NpgsqlCommand command2 = new NpgsqlCommand("SELECT name FROM goods WHERE category\_id = " + category\_id, connection);

NpgsqlDataReader reader2 = command2.ExecuteReader();

List<string> data2 = new List<string>();

while (reader2.Read()) {

data2.Add(reader2.GetValue(0).ToString());

}

PrintRow(data2);

reader2.Close();

break;

case 3:

Console.Write("Input trademark\_id - ");

string trademark\_id3 = Console.ReadLine();

NpgsqlCommand command3 = new NpgsqlCommand("SELECT name FROM department WHERE department\_id = " + trademark\_id3, connection);

NpgsqlDataReader reader3 = command3.ExecuteReader();

List<string> data3 = new List<string>();

while (reader3.Read()) {

data3.Add(reader3.GetValue(0).ToString());

}

PrintRow(data3);

reader3.Close();

break;

}

}

Фрагмент програми для отримання імен та типів стовпчиків таблиці:

public void Print() {

List<string> tables = GetTables("print");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

List<string> columns = GetColumns(tables[selected\_index]);

PrintRow(columns);

NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM " + tables[selected\_index], connection);

NpgsqlDataReader reader = command.ExecuteReader();

List<string> data = new List<string>();

while (reader.Read()) {

for (int i = 0; i < columns.Count; i++) {

data.Add(reader.GetValue(i).ToString());

}

PrintRow(data);

data.Clear();

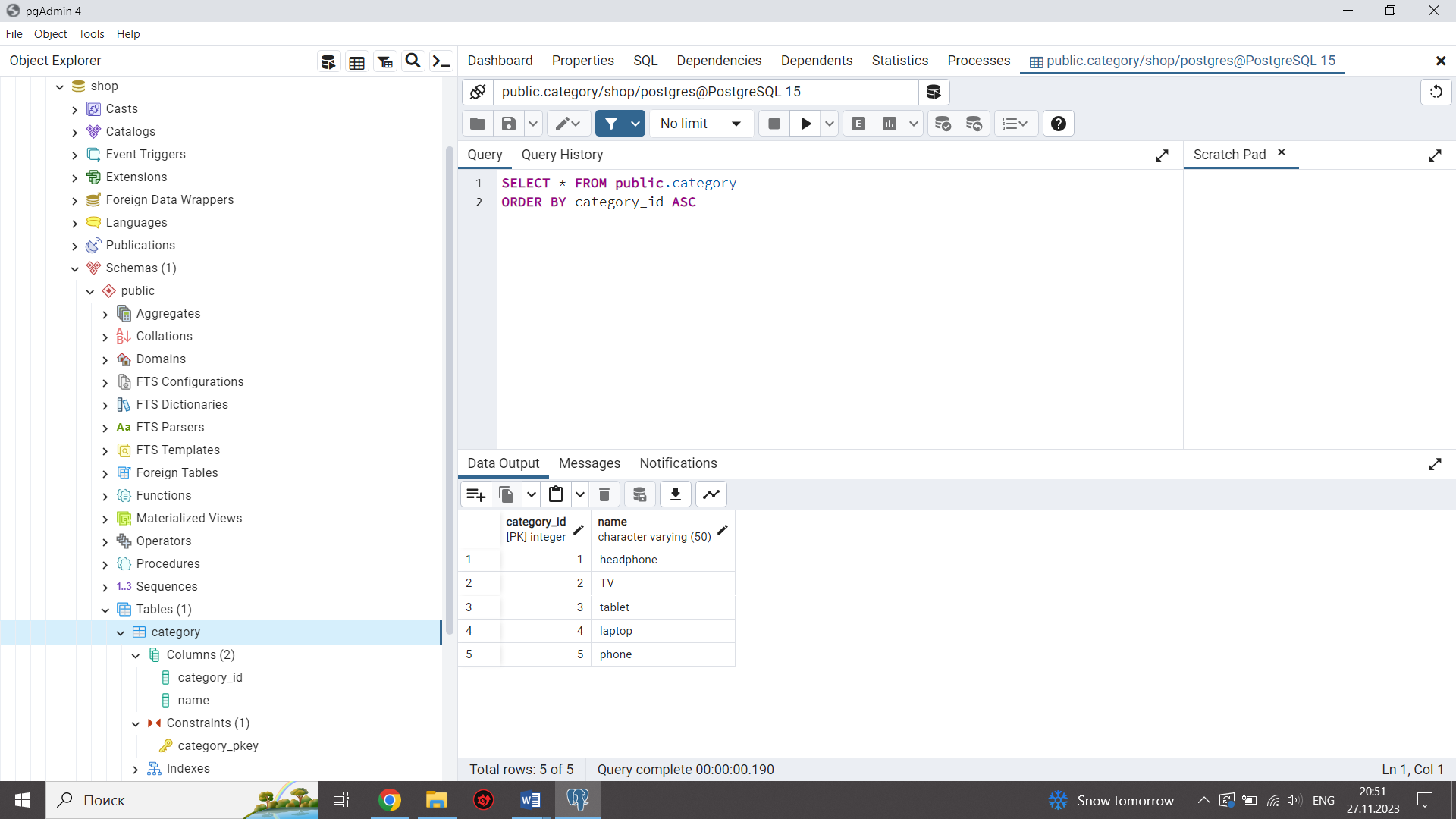
}

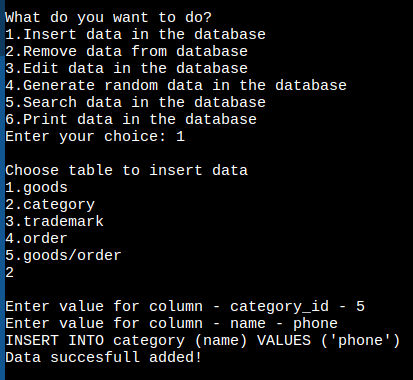
reader.Close();

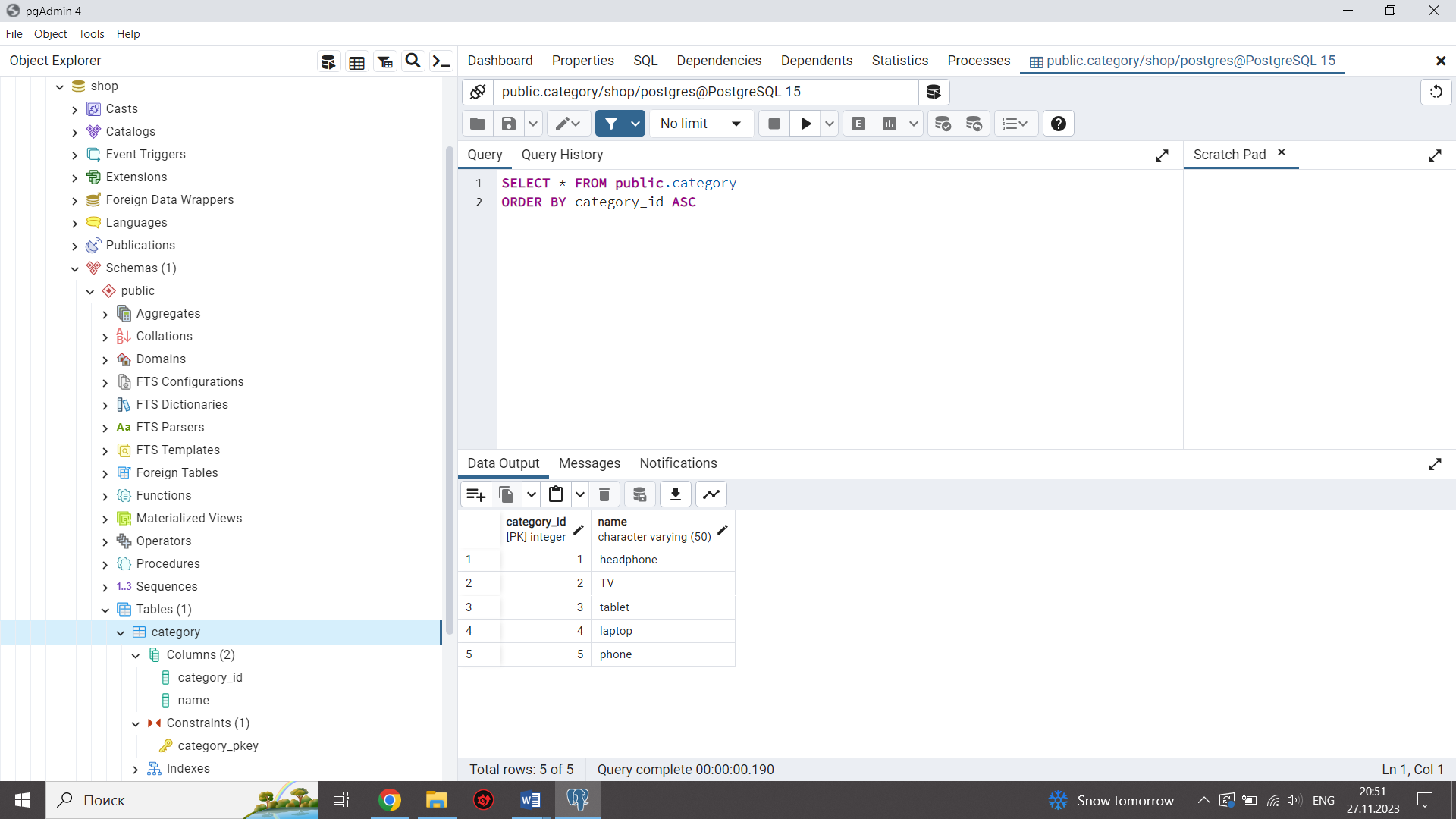
}

Дані фрагменти програми, які наведені вижче, відповідають за функціонал додвання даних, редагування та вилучення даних у базі даних.  
Взаємодія відбувається через клас Model, який займається підключенням до БД, а самі функції знаходяться у файлі Controller.

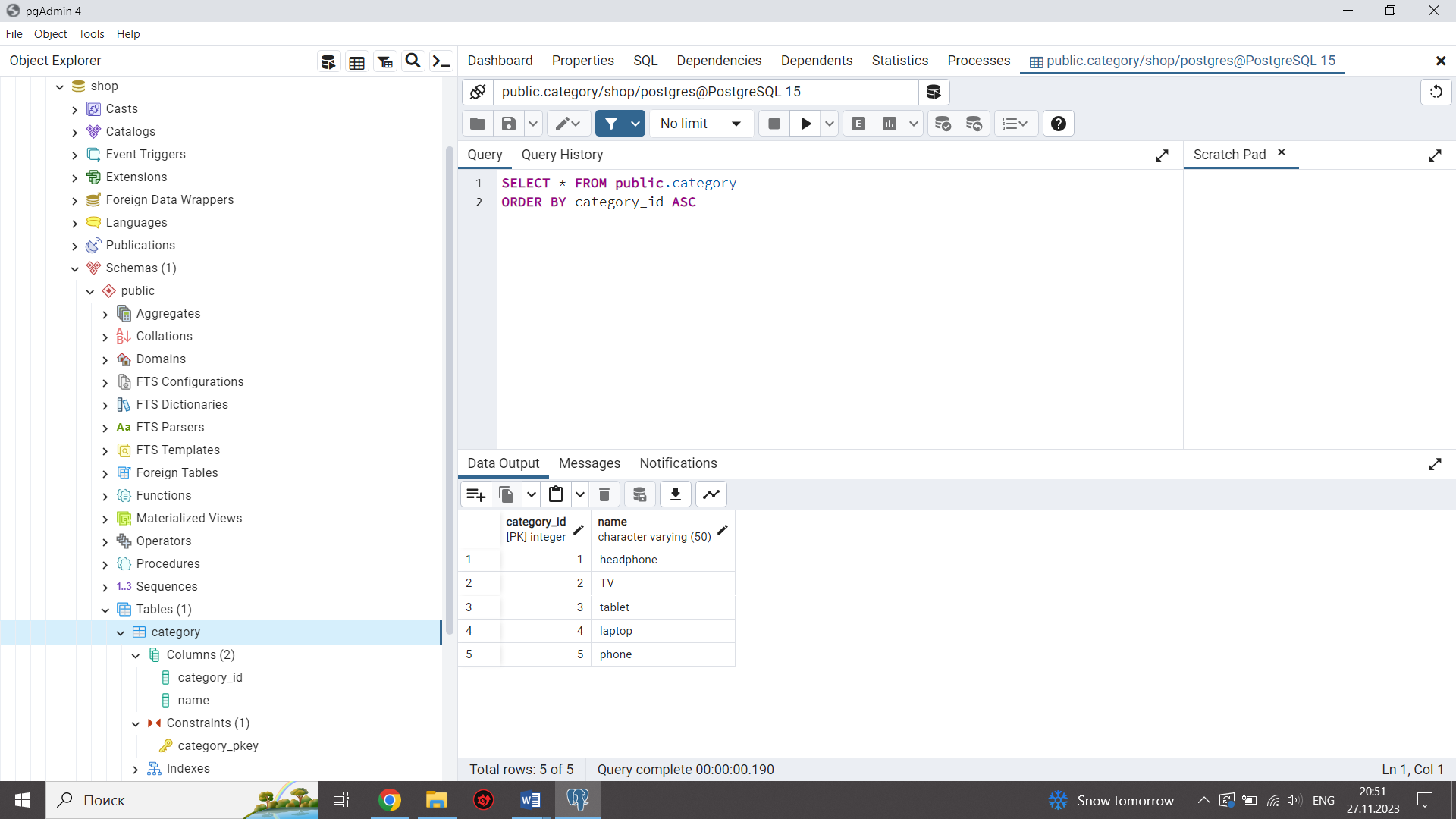
**Скріншоти результатів виконання операції вставки запису в дочірню таблицю**

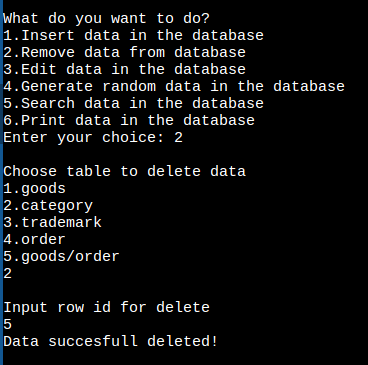


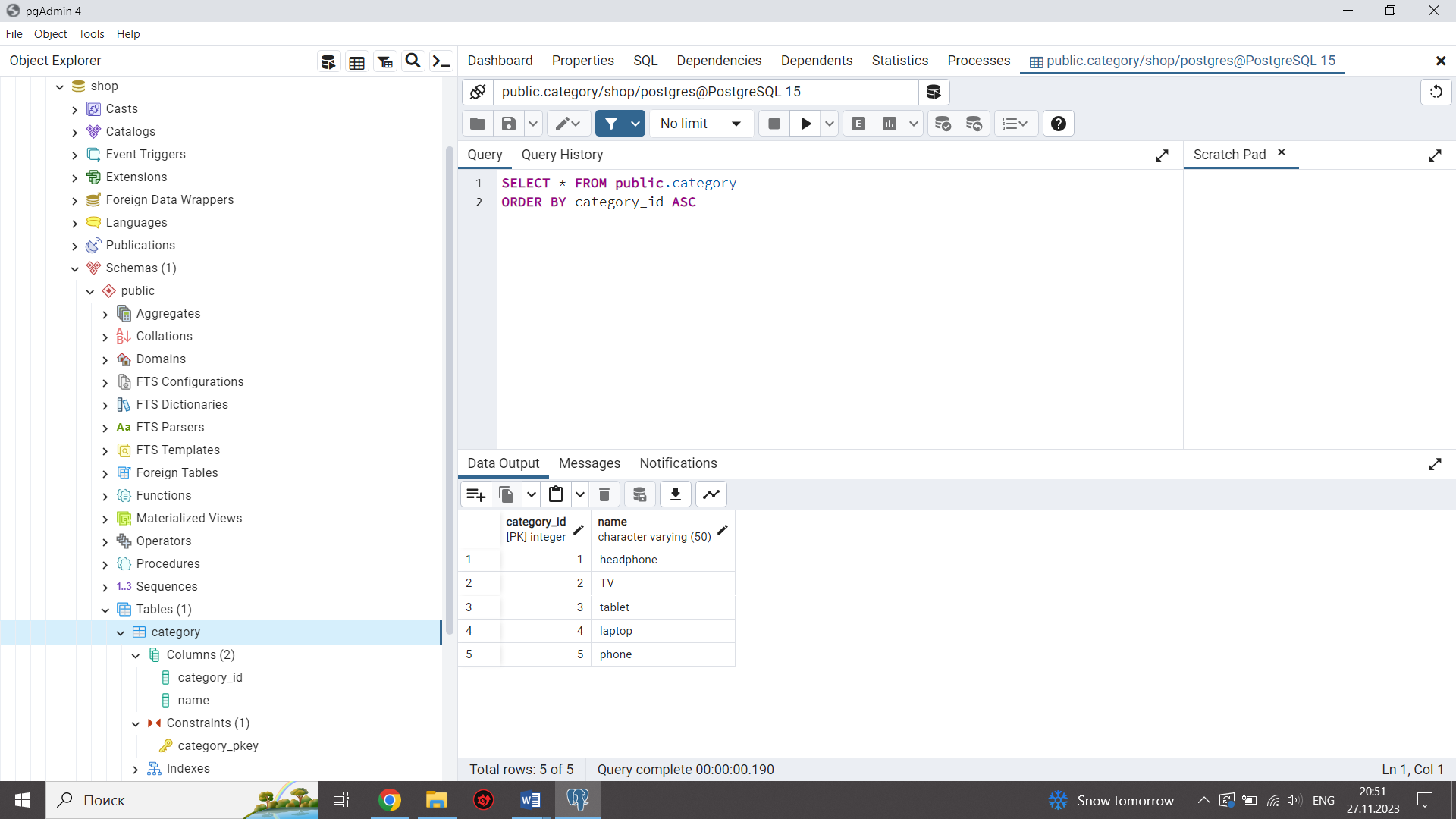




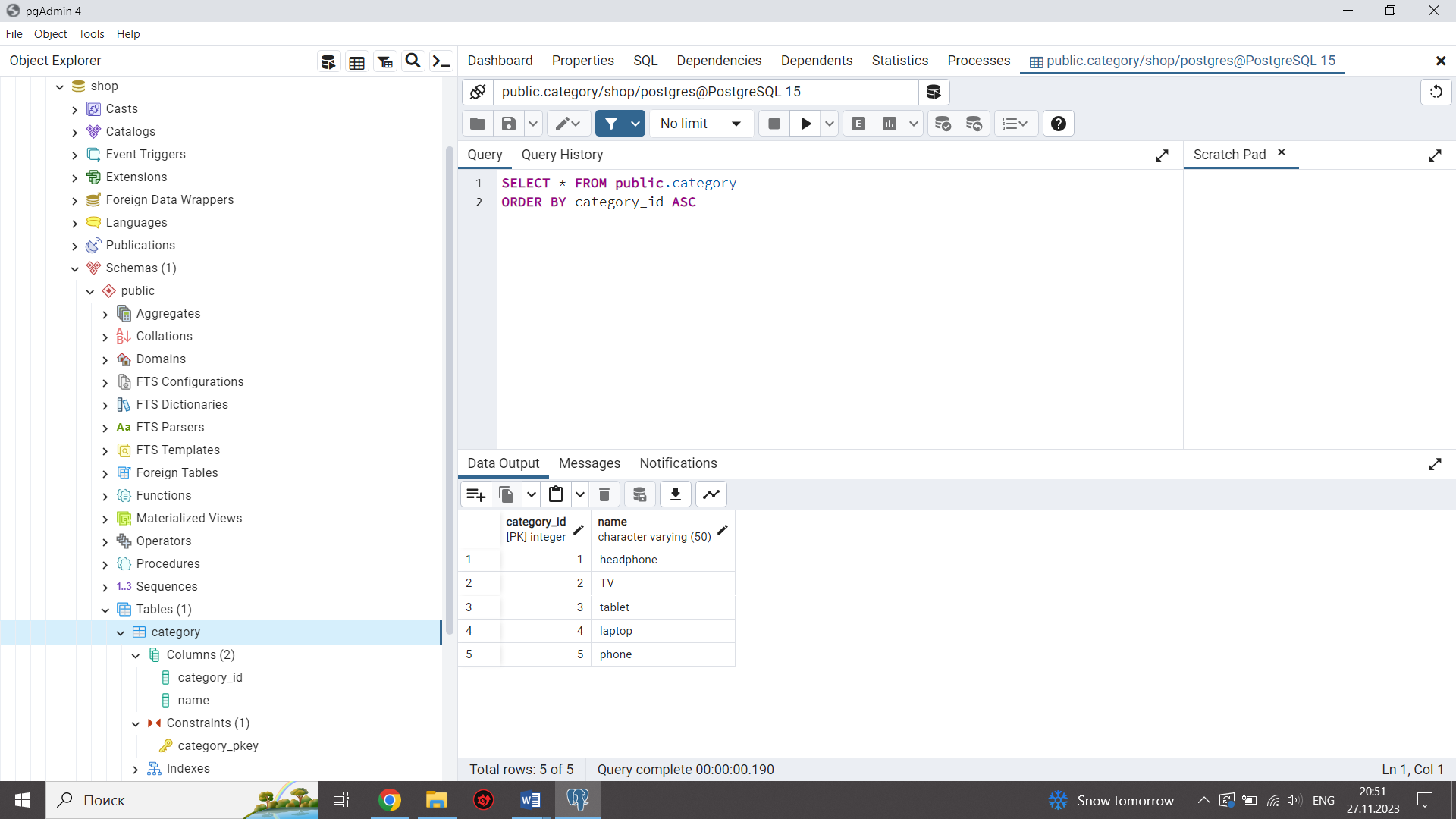
**Скріншоти результатів виконання операції видалення даних з таблиці**

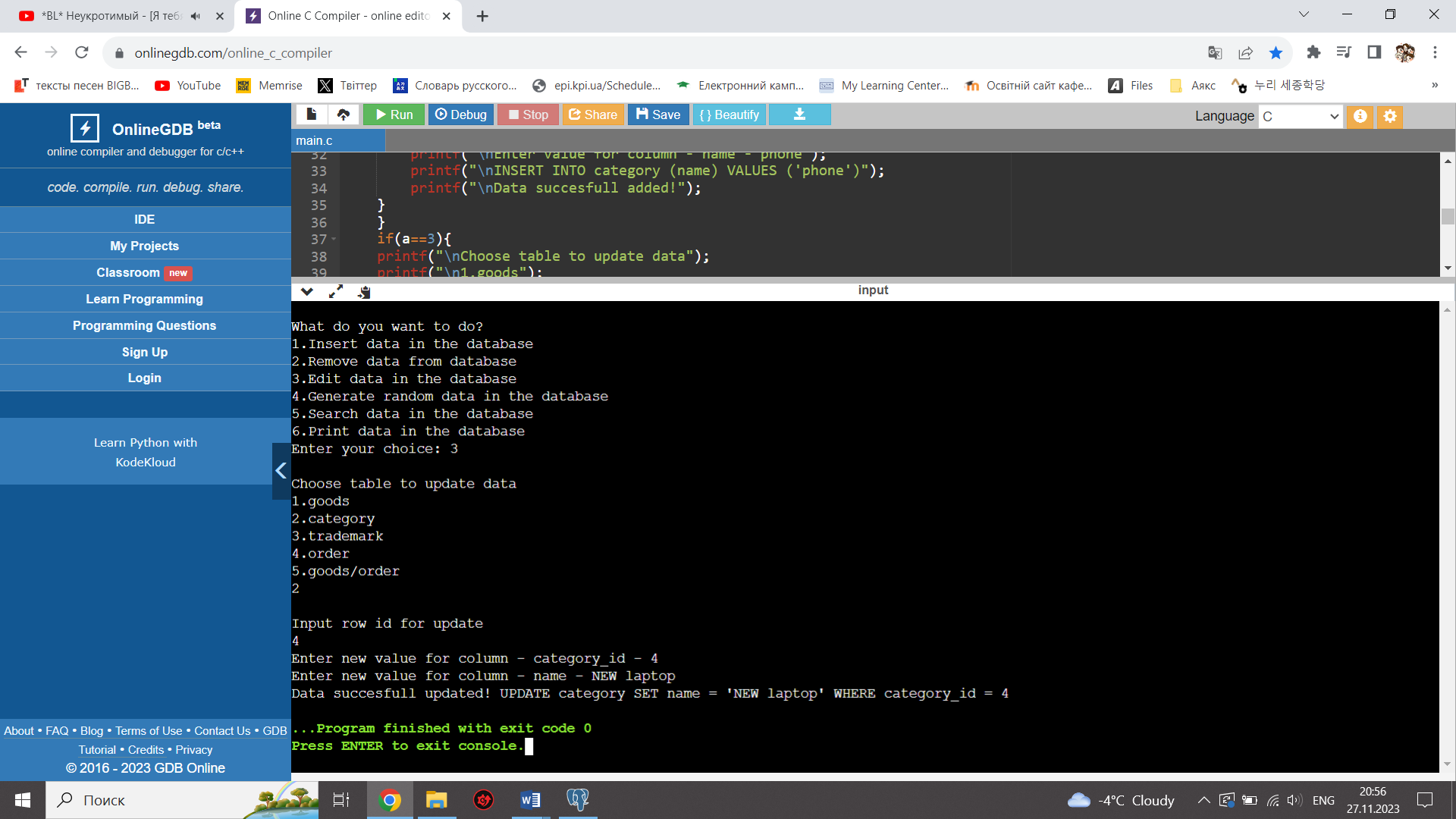


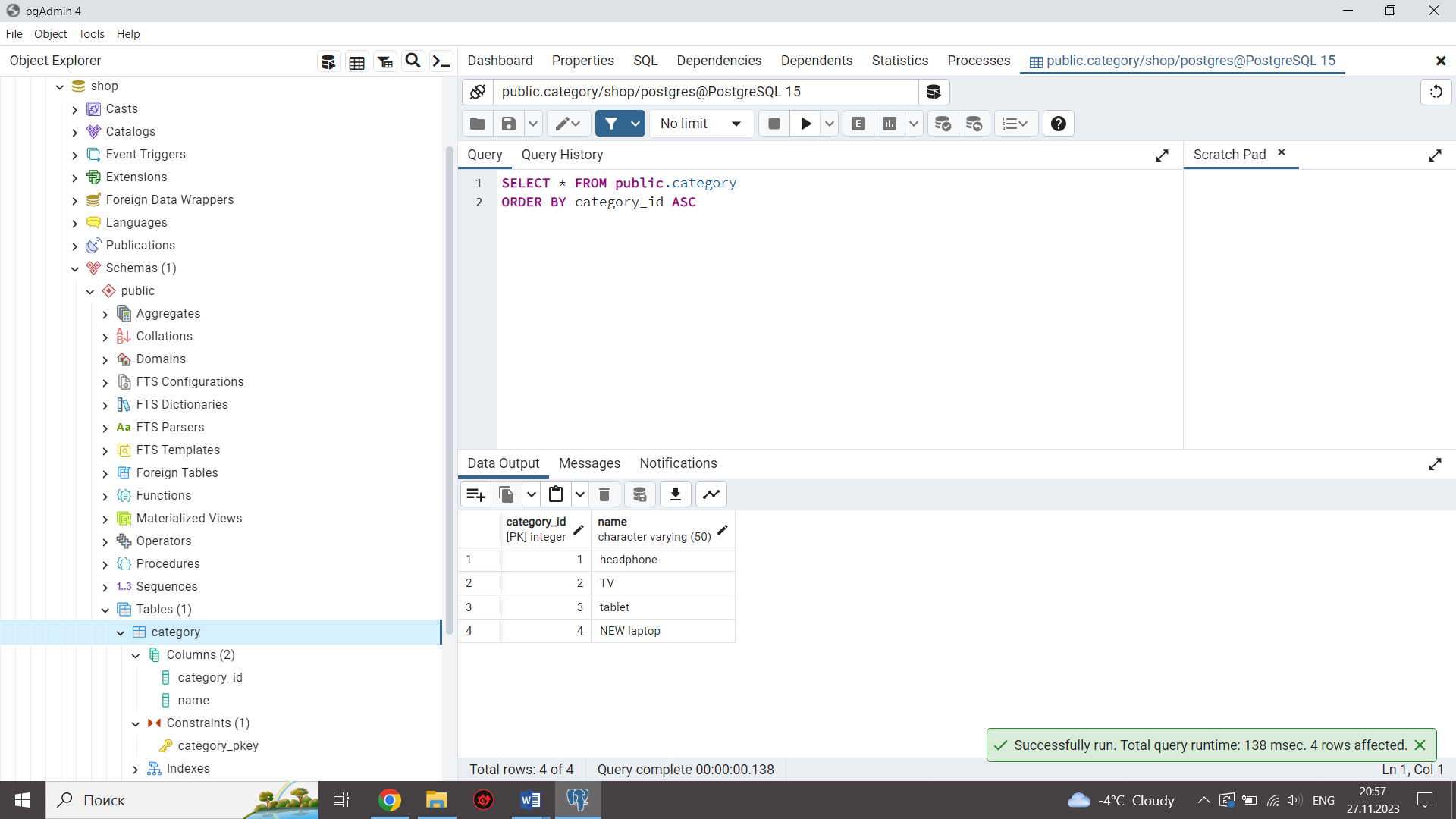
****



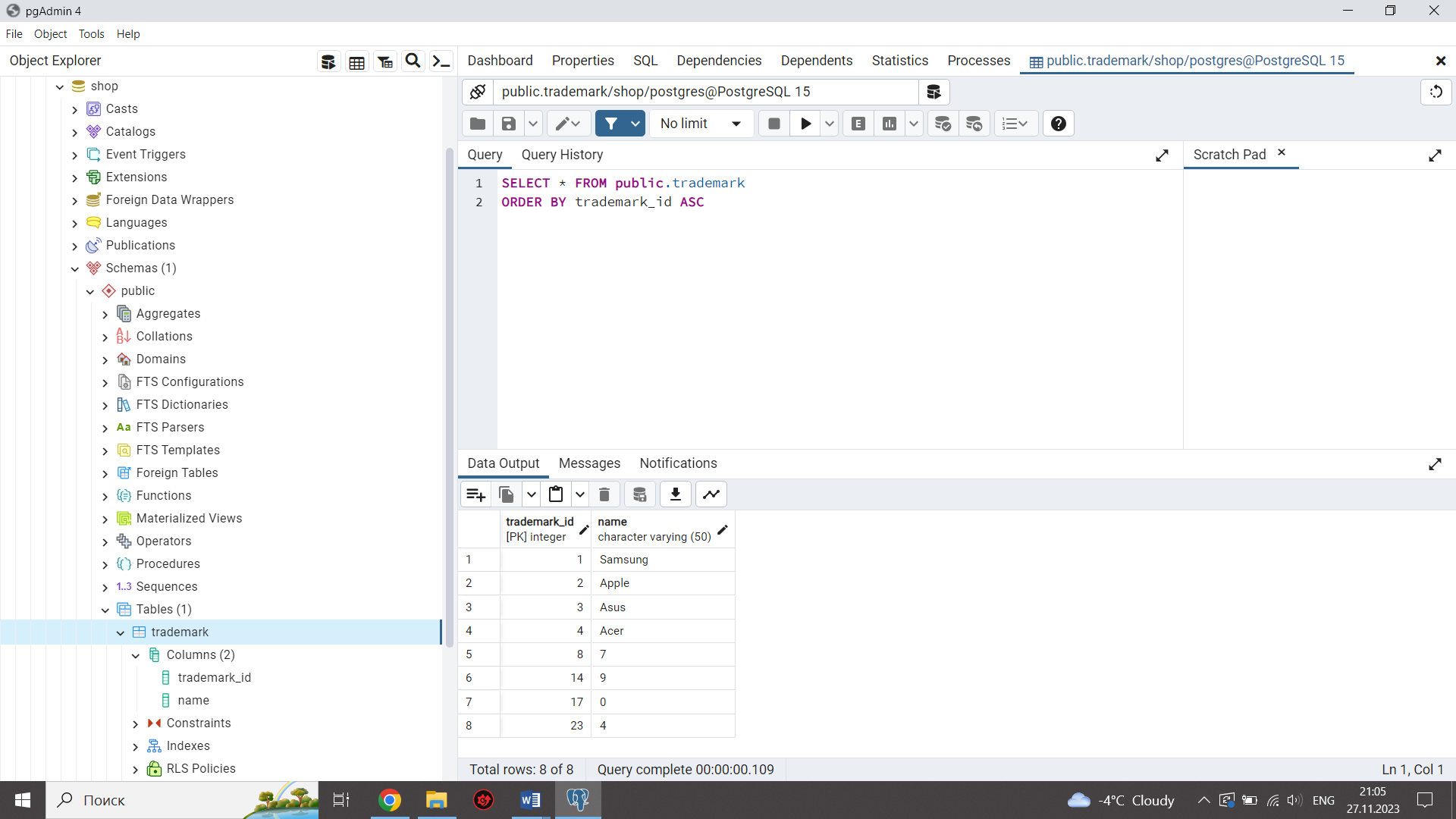
**Скріншоти результатів виконання операції оновлення даних в таблиці**

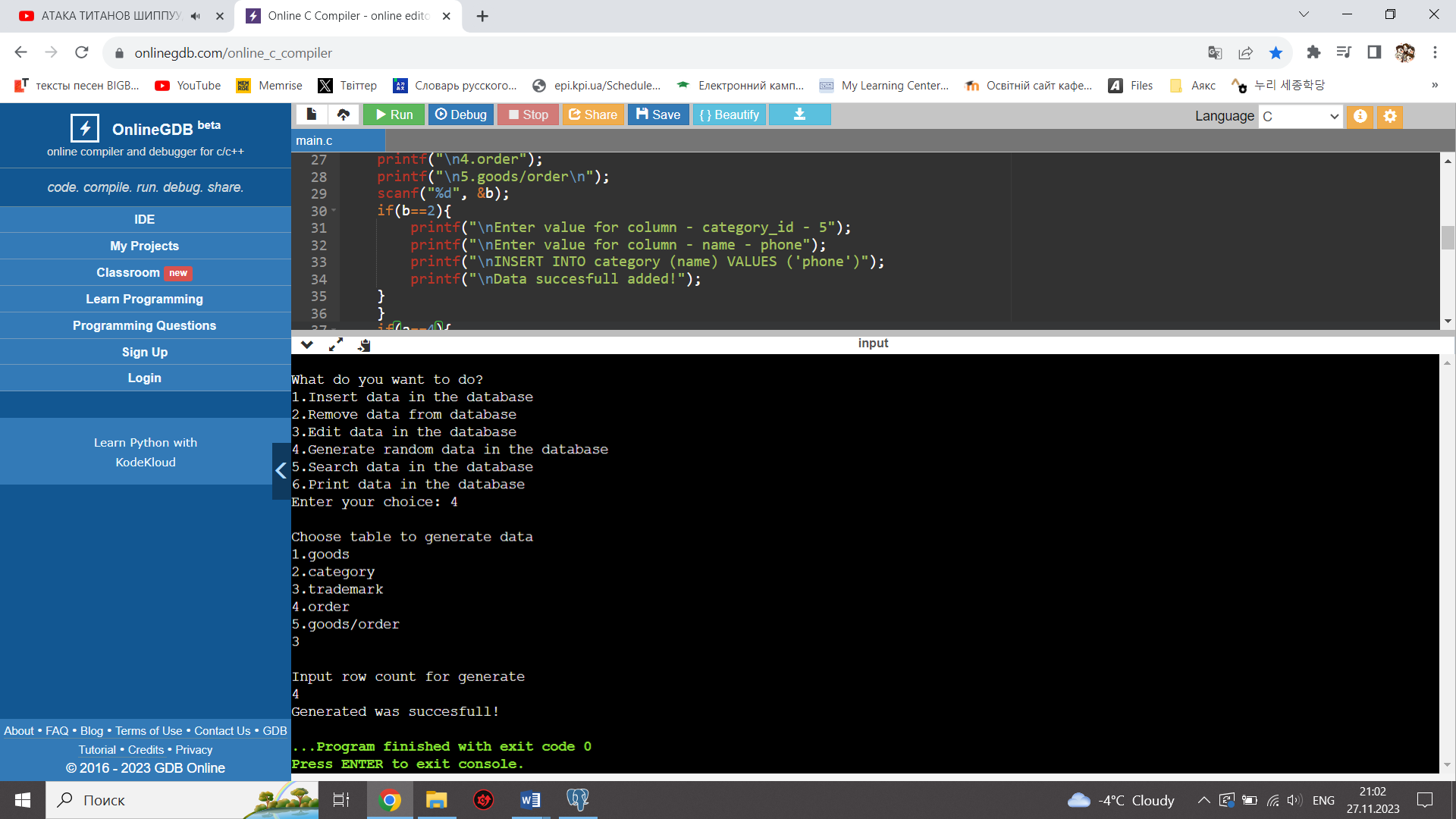


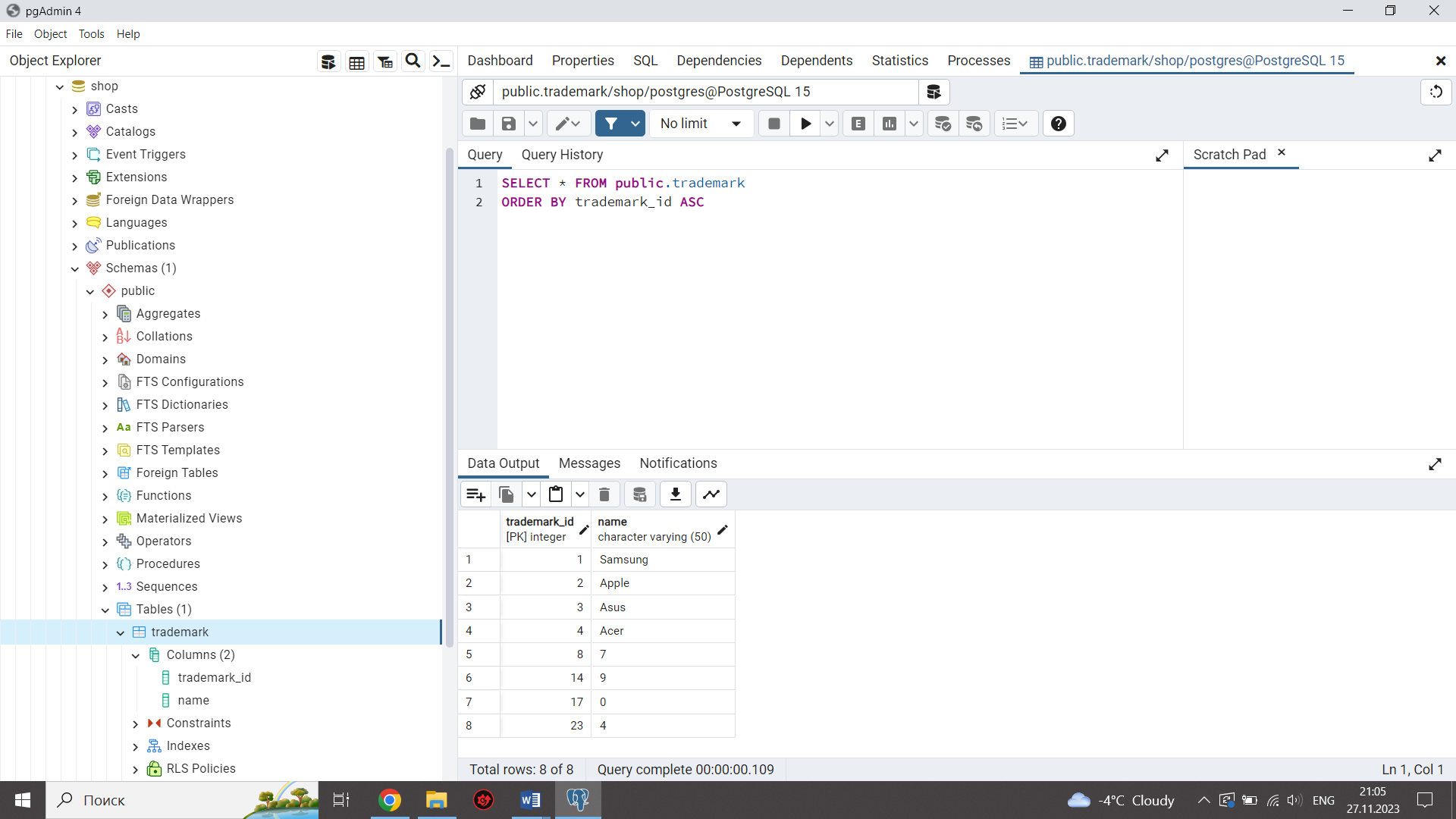




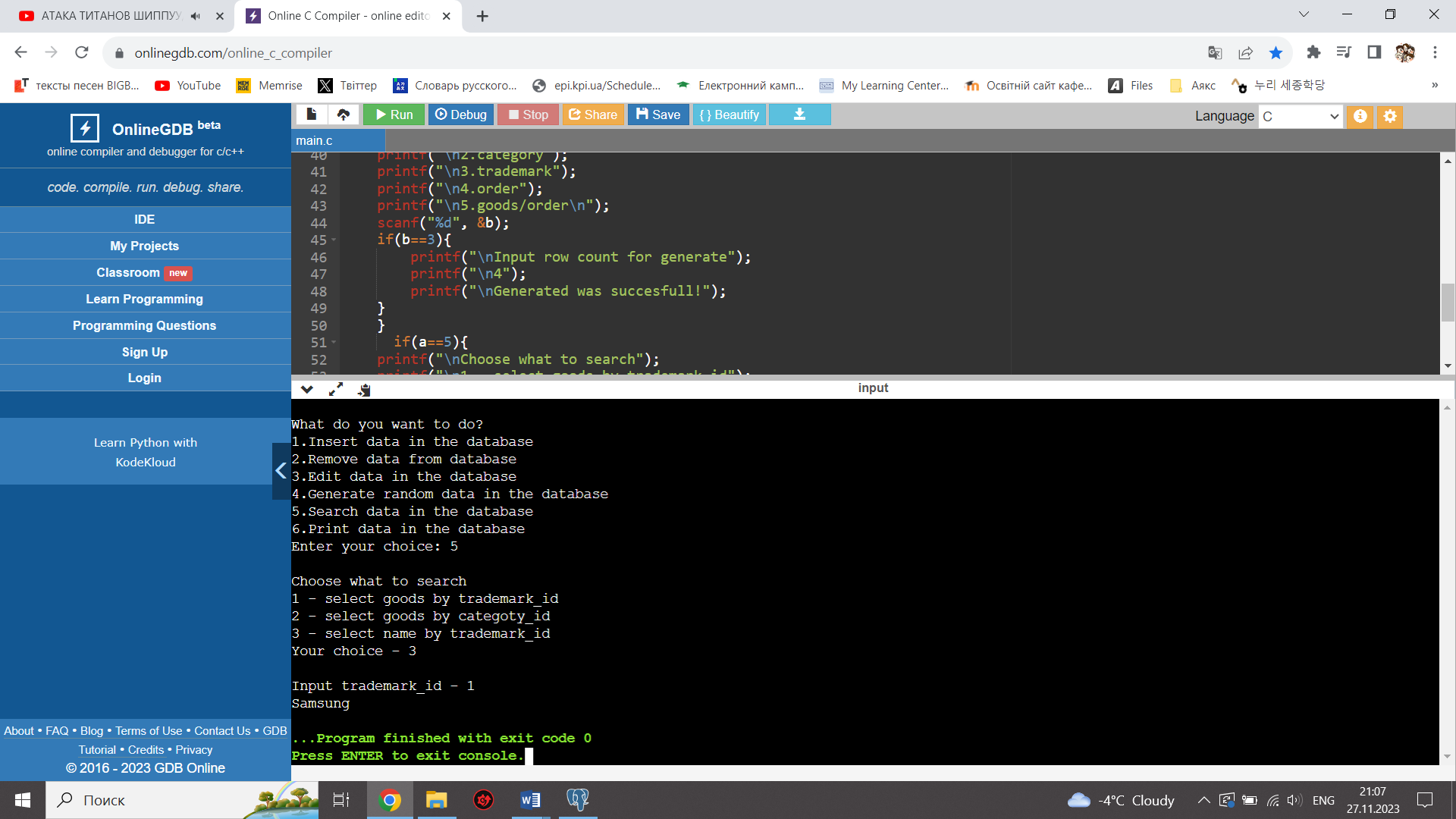
**Скріншоти результатів виконання операції генерування даних в таблиці**



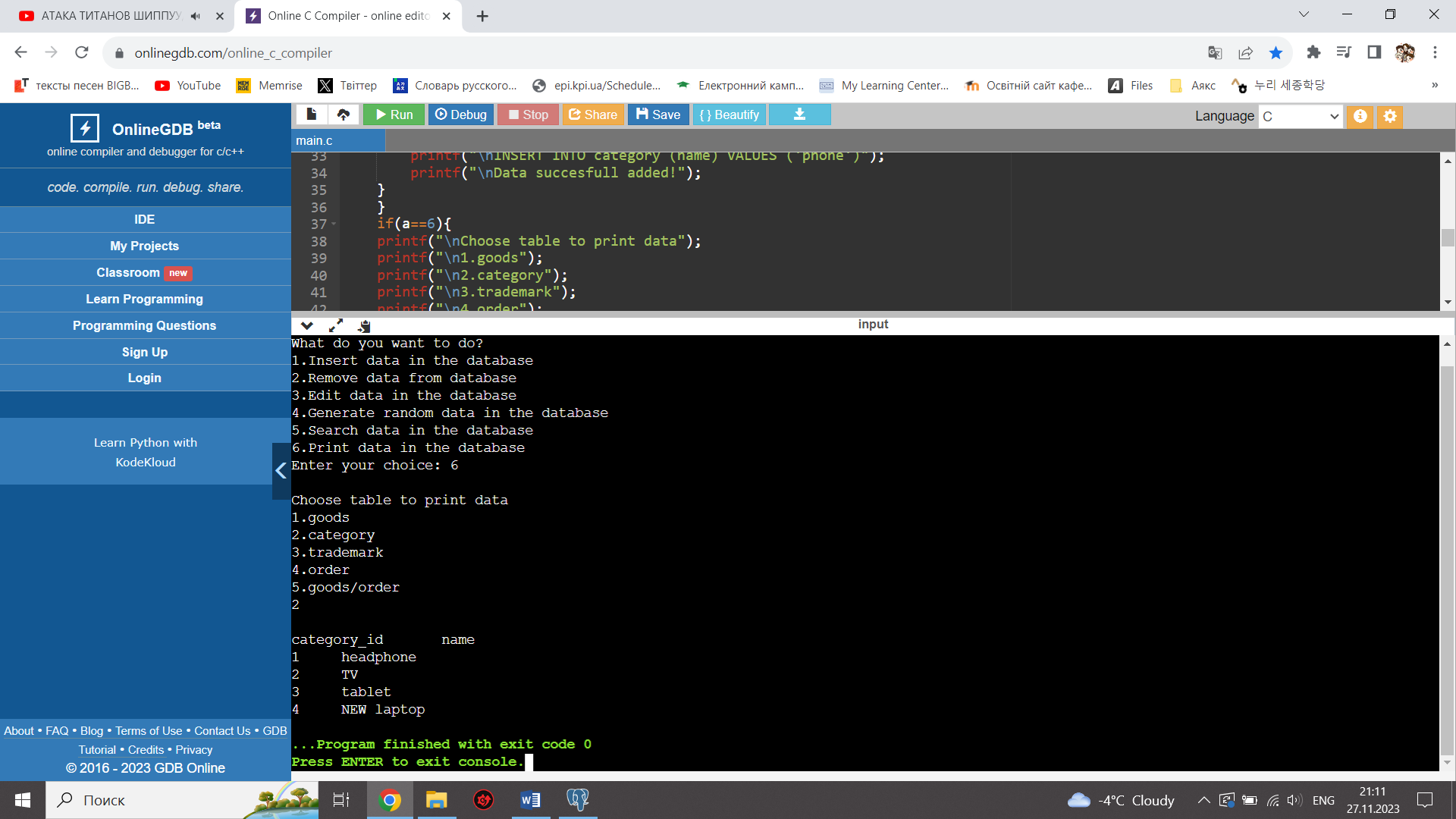




**Скріншоти результатів виконання операції пошуку даних у таблиці**



**Скріншоти результатів виконання операції отримання імен та типів стовпчиків таблиці**



**Текст програми:**

***Program.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Npgsql;

namespace PostgreConsoleInteractorCS {

class Program {

static void Main(string[] args) {

DatabaseView view = new DatabaseView();

view.Process();

}

}

}

***DatabaseView.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PostgreConsoleInteractorCS {

public class DatabaseView {

DatabaseController controller;

public void Process() {

string connection\_string = "Server=localhost;Port=5432;User ID=postgres;Password=123;Database=shop;";

controller = new DatabaseController(connection\_string);

while (true) {

int action = controller.ShowActionList();

controller.Execute(action);

}

}

}

}

***DatabaseController.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Npgsql;

namespace PostgreConsoleInteractorCS {

public class DatabaseController {

DatabaseModel db = null;

public DatabaseController(string connection\_string) {

db = new DatabaseModel(connection\_string);

}

public int ShowActionList() {

int choose;

Console.WriteLine("\nWhat do you want to do?");

Console.WriteLine("1.Insert data in the database");

Console.WriteLine("2.Remove data from database");

Console.WriteLine("3.Edit data in the database");

Console.WriteLine("4.Generate random data in the database");

Console.WriteLine("5.Search data in the database");

Console.WriteLine("6.Print data in the database");

Console.Write("Enter your choice: ");

choose = int.Parse(Console.ReadLine());

return choose;

}

public void Execute(int action) {

if (db != null) {

switch (action) {

case 1:

db.Insert();

break;

case 2:

db.Delete();

break;

case 3:

db.Update();

break;

case 4:

db.Generate();

break;

case 5:

db.Search();

break;

case 6:

db.Print();

break;

default:

Console.WriteLine("Not correct operation. Try again!");

break;

}

}

}

}

}

***DatabaseModel.cs***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Npgsql;

namespace PostgreConsoleInteractorCS {

public class DatabaseModel {

NpgsqlConnection connection;

public DatabaseModel(string connection\_string) {

connection = new NpgsqlConnection(connection\_string);

connection.Open();

Console.WriteLine("Connection was opened succesfull!");

}

public void Insert() {

List<string> tables = GetTables("insert");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

List<string> columns = GetColumns(tables[selected\_index]);

List<string> values = new List<string>();

foreach (string column in columns) {

Console.Write("Enter value for column - " + column + " - ");

values.Add(Console.ReadLine());

}

string insert\_query = "INSERT INTO " + tables[selected\_index] + " (" + ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";

NpgsqlCommand insert\_command = new NpgsqlCommand(insert\_query, connection);

Console.WriteLine(insert\_query);

insert\_command.ExecuteNonQuery();

Console.WriteLine("Data succesfull added!");

}

public void Delete() {

List<string> tables = GetTables("delete");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

Console.WriteLine("Input row id for delete");

int row\_id = int.Parse(Console.ReadLine());

NpgsqlCommand delete\_command = new NpgsqlCommand("DELETE FROM " + tables[selected\_index] + " WHERE " + tables[selected\_index] + "\_id = " + row\_id, connection);

delete\_command.ExecuteNonQuery();

Console.WriteLine("Data succesfull deleted!");

}

public void Update() {

List<string> tables = GetTables("update");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

List<string> columns = GetColumns(tables[selected\_index]);

List<string> values = new List<string>();

Console.WriteLine("Input row id for update");

int row\_id = int.Parse(Console.ReadLine());

foreach (string column in columns) {

Console.Write("Enter new value for column - " + column + " - ");

values.Add(Console.ReadLine());

}

string update\_query = "UPDATE " + tables[selected\_index] + " SET " + UpdatePartialString(columns, values) + " WHERE " + tables[selected\_index] + "\_id = " + row\_id;

NpgsqlCommand update\_command = new NpgsqlCommand(update\_query, connection);

update\_command.ExecuteNonQuery();

Console.WriteLine("Data succesfull updated! " + update\_query);

}

public void Generate() {

List<string> tables = GetTables("generate");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

Console.WriteLine("Input rows count for generate");

int rows\_count = int.Parse(Console.ReadLine());

List<string> columns = GetColumns(tables[selected\_index]);

Random random = new Random();

for (int i = 0; i < rows\_count; i++) {

while (true) {

try {

List<string> values = new List<string>();

foreach (string column in columns) {

values.Add(random.Next(0, 10).ToString());

}

//values[values.Count - 1] = "1";

string insert\_query = "INSERT INTO " + tables[selected\_index] + " (" + ListToString(columns, false) + ") VALUES (" + ListToString(values, true) + ")";

NpgsqlCommand insert\_command = new NpgsqlCommand(insert\_query, connection);

insert\_command.ExecuteNonQuery();

break;

}

catch (Exception ex) {

//Console.WriteLine(ex.Message);

}

}

}

Console.WriteLine("Generated was succesfull!");

}

public void Search() {

Console.WriteLine("Choose what to search:");

Console.WriteLine("1 - select goods by trademark\_id");

Console.WriteLine("2 - select goods by categoty\_id");

Console.WriteLine("3 - select name by trademark\_id");

Console.Write("Your choice - ");

int search = int.Parse(Console.ReadLine());

switch (search) {

case 1:

Console.Write("Input trademark\_id - ");

string trademark\_id = Console.ReadLine();

NpgsqlCommand command1 = new NpgsqlCommand("SELECT name FROM category WHERE department\_id = " + trademark\_id, connection);

NpgsqlDataReader reader1 = command1.ExecuteReader();

List<string> data1 = new List<string>();

while (reader1.Read()) {

data1.Add(reader1.GetValue(0).ToString());

}

PrintRow(data1);

reader1.Close();

break;

case 2:

Console.Write("Input category\_id - ");

string category\_id = Console.ReadLine();

NpgsqlCommand command2 = new NpgsqlCommand("SELECT name FROM goods WHERE category\_id = " + category\_id, connection);

NpgsqlDataReader reader2 = command2.ExecuteReader();

List<string> data2 = new List<string>();

while (reader2.Read()) {

data2.Add(reader2.GetValue(0).ToString());

}

PrintRow(data2);

reader2.Close();

break;

case 3:

Console.Write("Input trademark\_id - ");

string trademark\_id3 = Console.ReadLine();

NpgsqlCommand command3 = new NpgsqlCommand("SELECT name FROM department WHERE department\_id = " + trademark\_id3, connection);

NpgsqlDataReader reader3 = command3.ExecuteReader();

List<string> data3 = new List<string>();

while (reader3.Read()) {

data3.Add(reader3.GetValue(0).ToString());

}

PrintRow(data3);

reader3.Close();

break;

}

}

public void Print() {

List<string> tables = GetTables("print");

int selected\_index = int.Parse(Console.ReadLine()) - 1;

List<string> columns = GetColumns(tables[selected\_index]);

PrintRow(columns);

NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM " + tables[selected\_index], connection);

NpgsqlDataReader reader = command.ExecuteReader();

List<string> data = new List<string>();

while (reader.Read()) {

for (int i = 0; i < columns.Count; i++) {

data.Add(reader.GetValue(i).ToString());

}

PrintRow(data);

data.Clear();

}

reader.Close();

}

private List<string> GetTables(string operation) {

Console.WriteLine("Choose table to " + operation + " data");

NpgsqlCommand command = new NpgsqlCommand("SELECT \* FROM information\_schema.tables WHERE table\_schema = 'public';", connection);

NpgsqlDataReader reader = command.ExecuteReader();

List<string> tables = new List<string>();

while (reader.Read()) {

tables.Add(reader.GetString(2));

}

reader.Close();

int index = 1;

foreach (string table in tables) {

Console.WriteLine(index.ToString() + '.' + table);

index++;

}

return tables;

}

private List<string> GetColumns(string table) {

string query = "SELECT \* FROM information\_schema.columns WHERE table\_schema = 'public' AND table\_name = '" + table + "'";

NpgsqlCommand command = new NpgsqlCommand(query, connection);

NpgsqlDataReader reader = command.ExecuteReader();

List<string> columns = new List<string>();

while (reader.Read()) {

columns.Add(reader.GetString(3));

}

reader.Close();

return columns;

}

private string ListToString(List<string> list, bool is\_value) {

string result = string.Empty;

for (int i = 1; i < list.Count; i++) {

if (is\_value) result += "'" + list[i] + "',";

else result += list[i] + ',';

}

result = result.Remove(result.Length - 1, 1);

return result;

}

private void PrintRow(List<string> list) {

foreach (string s in list) {

Console.Write(s + " ");

}

Console.Write('\n');

}

private string UpdatePartialString(List<string> columns, List<string> values) {

string partial = string.Empty;

for (int i = 1; i < columns.Count; i++) {

partial += columns[i] + " = '" + values[i] + "'";

if (i < columns.Count - 1) partial += ", ";

}

return partial;

}

}

}