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"

Виконав:
студент III курсу
групи КВ-23
Марінченко М. О.

Київ – 2024

Мета: здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

Посилання на телеграм студента
Посилання на репозиторій

Виконання роботи
Сутності предметної області

1. Клієнт(Client), з атрибутами: код клієнта, ім’я, прізвище,електронна пошта, номер телефона, дата реєстрації. Призначена для збереження інформації про клієнта;

2. Бронювання (Bookings), з атрибутами: код бронювання, код клієнта, код сесії, дата бронювання, статус бронювання. Призначена для збереження інформації про бронювання відповідним клієнтом на відповідну сесію.

3. Сесія (Sessions), з атрибутами: код сесії, код тренера, код локації, час початку, час закінчення, максимальна кількість учасників, ціна. Призначена для збереження інформації про сесію тренування;

4. Локації (Locations), з атрибутами: код локації, назва, адресса, місто, вмістивність. Призначена для збереження інформації про місце проведення тренування.

5. Тренери(Instructors), з атрибутами: код тренера, досвід роботи, опис(біографія), рейтинг. Призначена для збереження інформації про тренерів.

Зв’язки між сутностями предметної області

Сутність “Клієнт” має зв’язок 1:N по відношенню до сутності “Бронювання ” оскільки клієнт може містити декілька бронювань.

Сутність “Сесія” має зв'язок 1:N по відношенню до “Бронювань”, оскільки одна сесія може бути забронюванна декілька разів.

Сутність “Локація” має зв'язок 1:N по відношенню до “Сесія”, оскільки одна локація може бути місцем проведенням для багатьох сесій, але сесія має лише одне місце проведення.

Сутність “Тренер” має зв’язок 1:N по відношенню до “Сесія”, оскільки один тренер може проводити декілька тренувальних сесій, але у сесії завжди один тільки тренер.

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

рисунку 1.

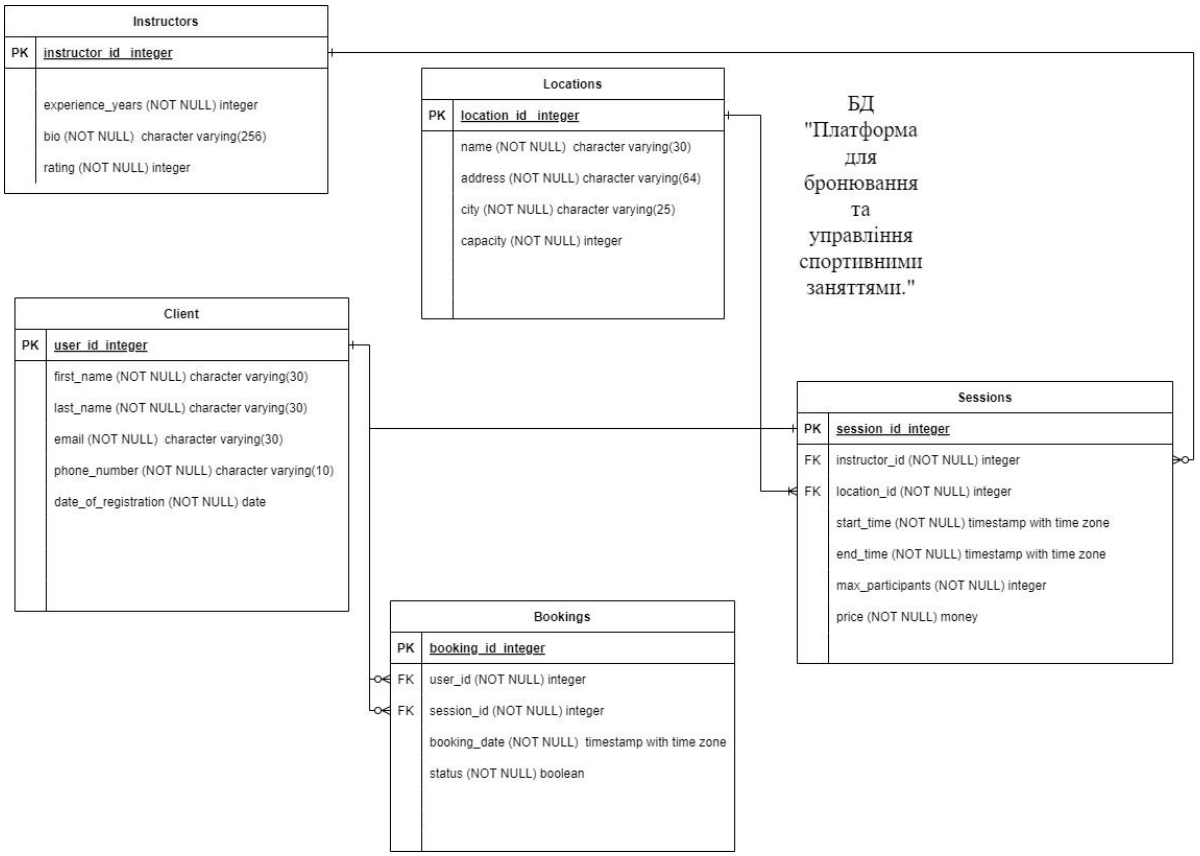


Рисунок 1 – Логічна модель

Середовище та компоненти розробки

У процесі розробки була використана мова програмування C#, інтегроване середовище розробки Visual Studio 2022, а також була використана бібліотека Npgsql v8.0.5, яка надає API для взаємодії з базою даних PostgreSQL.

Шаблон проектування

Модель-представлення-контролер (MVC) – це шаблон проектування, що використовується у програмі. Кожен компонент відповідає за певну функціональну частину:

1. Модель (Model) – це клас, що відображає логіку роботи з даними, обробляє всі операції з даними, такі як додавання, оновлення, вилучення.
2. Представлення (View) – це клас, через який користувач взаємодіє з програмою. У даному випадку, консольний інтерфейс, який відображає дані для користувача та зчитує їх з екрану.
3. Контролер (Controller) – це клас, який відповідає за зв’язок між користувачем і системою. Він приймає введені користувачем дані та обробляє їх. В залежності від результатів, викликає відповідні дії з Model або View.

Даний підхід дозволяє розділити логіку програми на логічні компоненти, що полегшує розробку, тестування і підтримку продукту.

Структура програми та її опис

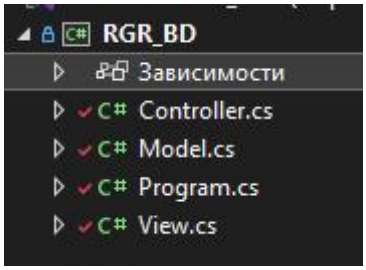


Рисунок 2 – Структура програми

З файлу Program.cs відбувається виклик контролера та передача йому управління.

У файлі Model.cs описаний клас моделі, який відповідає за управління підключенням до бази даних і виконанням низькорівневих запитів до неї.

У файлі Controller.cs реалізовано інтерфейс взаємодії з користувачем, включаючи обробку запитів користувача, а також інші дії, необхідні для взаємодії з моделлю та представленням.

У файлі View.cs описаний клас, який відображає результати виконання різних дій користувача на екрані консолі. Цей компонент відповідає за представлення даних користувачу в зручному для сприйняття вигляді.

Отже, структура програми відповідає патерну MVC.

Структура меню програми

На рисунку 3 зображено меню користувача, яке складається з семи пунктів.

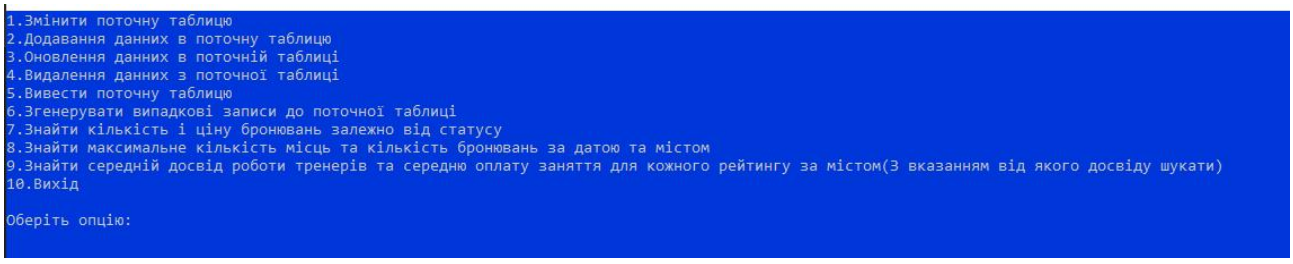


Рисунок 3 – Структура меню користувача

Фрагмент коду (файл Controller.cs), в якому наведено головний цикл роботи програми

```
public static void Run()
{
    bool cycle_run = true;
    while (cycle_run)
    {
        int choice = view.ShowMenu();
        if (current_table == "None" && choice < 7 && choice != 1)
        {
            Console.WriteLine("Оберіть таблицю для роботи!!!");
            Thread.Sleep(1000);
        }
        else
        {
            switch (choice)
            {
                case 1:
                    ChangeCurTable();
                    break;
                case 2:
                    AddDataToTable();
                    break;
                case 3:
                    UpdateTableData();
                    break;
                case 4:
                    DeleteData();
                    break;
                case 5:
                    ShowCurTable();
                    break;
                case 6:
                    GenerateData();
                    break;
                case 7:
                    Search(1);
                    break;
                case 8:
                    Search(2);
                    break;
                case 9:
                    Search(3);
                    break;
                case 10:
                    cycle_run = !cycle_run;
                    break;
            }
        }

    }
}
```

Фрагмент коду (файл Model.cs), в якому наведено функції внесення, редагування, вилучення та генерації даних у базі даних

Функція внесення даних:

```
public bool AddDataToTableModel(List<(string Column, string Value)> values,string
table_name)
{
    try
    {
        connection.Open();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при підключенні до бази данних");
        return true;
    }
    Dictionary<string, string> columnTypes = GetColumnTypes(table_name);
    try
    {
        List<string> setClauses = new List<string>();
        List<string> setValues = new List<string>();
        foreach (var column in values)
        {
            setClauses.Add($"{column.Column}");
            setValues.Add($"@{column.Column}");
        }
        string setClause_str = string.Join(",", setClauses);
        string setValues_str = string.Join(",", setValues);
        string query = $"INSERT INTO {table_name} ({setClause_str}) VALUES
({setValues_str});";
        using (var cmd = new NpgsqlCommand(query, connection))
        {
            foreach (var (Column, Value) in values)
            {
                cmd.Parameters.AddWithValue($"@{Column}", GetConvertedValues(columnTypes,
Column, Value));
            }
            cmd.ExecuteNonQuery();
        }

    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при додаванні данних " + ex.Message);
        return true;
    }
    try
    {
        connection.Close();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при закритті з'єднання з базою данних");
        return true;
    }

    return false;
       }
```

Дана функція вставляє нове значення до відповідної таблиці в БД, генеруючи для неї відповідний запрос, також приводить значення введені користувачем до відповідного типу.Для відслітковування помилок використовується try-catch.

Функція оновлення даних:

```
public bool UpdateDataInTable(List<(string Column, string Value)> values_res, string
table_name, int pk)
{
    string pk_str_column = GetPrimaryKeyColumn(table_name);
    try
    {
        connection.Open();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при підключенні до бази данних");
        return true;
    }
    Dictionary<string, string> columnTypes = GetColumnTypes(table_name);
    try
    {
        List<string> setClauses = new List<string>();
        foreach (var column in values_res) {
            setClauses.Add($"{column.Column} = @{column.Column}");
        }
        string setClause_str = string.Join(",", setClauses);
        string query = $"UPDATE {table_name} SET {setClause_str} WHERE {pk_str_column}
= {pk};";
        using (var cmd = new NpgsqlCommand(query, connection))
        {
            foreach (var (Column, Value) in values_res)
            {
                cmd.Parameters.AddWithValue($"@{Column}", GetConvertedValues(columnTypes,
Column, Value));
            }
            cmd.ExecuteNonQuery();
        }

    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при зміненні данних " + ex.Message);
        return true;
    }
    try
    {
        connection.Close();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при закритті з'єднання з базою данних");
        return true;
    }

    return false;
       }
```

Дана функція знаходить назву колонки де знаходиться pk таблиці для генерування строки запросу до відповідної таблиці в БД, після чого приводить отриманні дані від користувача і змінює відповідні дані.Для відслітковування помилок використовується try-catch.

Функція видалення значень:

```
public bool DeleteDataOfTable(string table_name, int pk, string pk_str)
{
    try
    {
        connection.Open();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при підключенні до бази данних");
        return (true);
    }
    try
    {
        string query = $"DELETE FROM {table_name} WHERE {pk_str} = {pk}";
        using (var cmd = new NpgsqlCommand(query, connection))
        {
            cmd.ExecuteNonQuery();
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при видаленні рядка");
        return (true);
    }
    try
    {
        connection.Close();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при закритті з'єднання з базою данних");
        return (true);
    }
    return false;
       }
```

Дана функція видаляє відповідний запис з обраної таблиці за pk, також у БД налаштоване CASCADE видалення, за рахунок цього будуть видалені відповідні записи в дочірніх таблицях.Для відслітковування помилок використовується try-catch.

Функція генерування даних:

```
 public bool GenerateDataToCurrentTable(string proc_name, int count_rows)
{
    try
    {
        connection.Open();
    }
    catch (Exception ex)
    {
        Console.WriteLine("Помилка при підключенні до бази данних");
        return true;
    }
    try
    {
        string query = $"CALL {proc_name}({count_rows})";
        using (var cmd = new NpgsqlCommand(query, connection))
        {
            cmd.CommandTimeout = 0;
            cmd.ExecuteNonQuery();
```

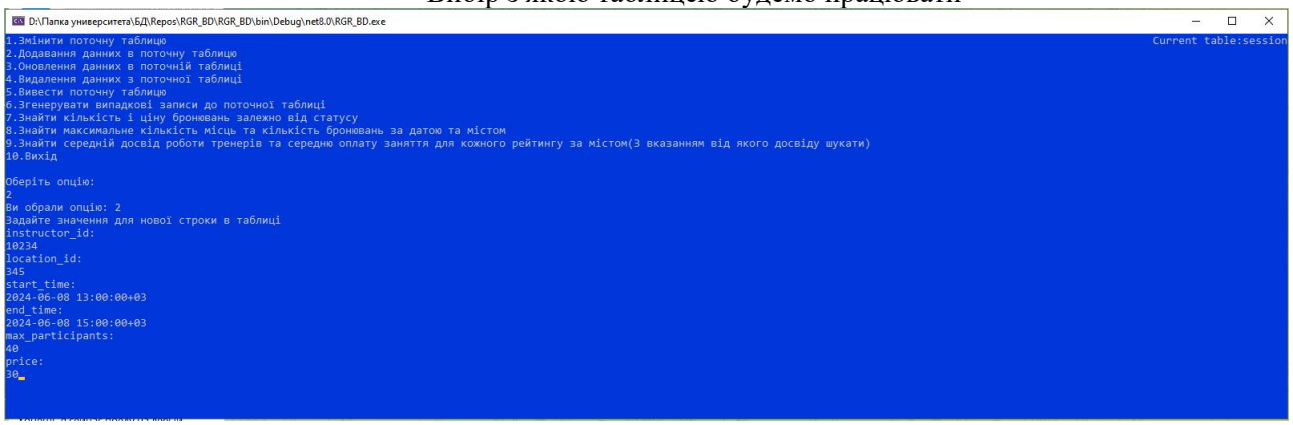
```
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при генерації випадкових данних" + ex.Message);
            return true;
        }
        try
        {
            connection.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при закритті з'єднання з базою данних");
            return true;
        }
        return false;
    }
```

Дана функція викликає відповідну процедуру з БД для генерування рандомних записів для відповідної таблиці.Для відслітковування помилок використовується try-catch.

Приклади виконання вище наведених функцій

Виконання вставки даних

Вибір з якою таблицею будемо працювати



Введення даних до таблиці session(Без помилок)

Query Query History

```
1 SELECT * FROM public.session
2 ORDER BY session_id DESC LIMIT 100
3
```

Data Output Messages Notifications

	session_id [PK] integer	instructor_id integer	location_id integer	start_time timestamp with time zone	end_time timestamp with time zone	max_participants integer	price money
1	120501	10234	345	2024-06-08 13:00:00+03	2024-06-08 15:00:00+03	40	30,00 ?

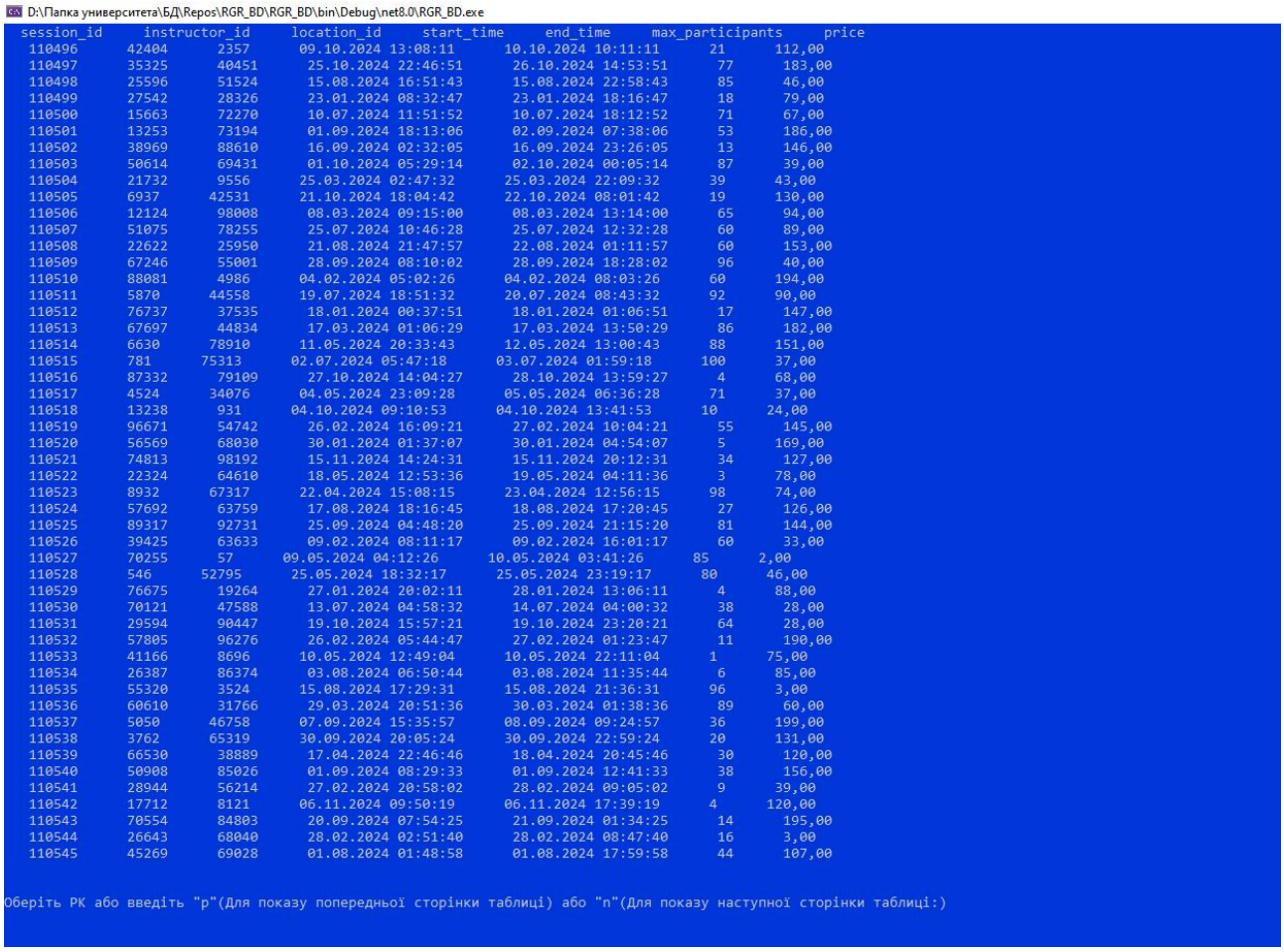
Вставленні дані в таблицю session

```
D:\Папка университета\БД\Repos\RGR_BD\RGR_BD\bin\Debug\net8.0\RGR_BD.exe
3.Оновлення данних в поточній таблиці
4.Видалення данних з поточної таблиці
5.Вивести поточну таблицю
6.Згенерувати випадкові записи до поточної таблиці
7.Знайти кількість і ціну бронювань залежно від статусу
8.Знайти максимальне кількість місць та кількість бронювань за датою та містом
9.Знайти середній досвід роботи тренерів та середню оплату заняття для кожного рейтингу за містом(З вказанням від якого досвіду шукати)
10.Вихід

Оберіть опцію:
2
Ви обрали опцію: 2
Задайте значення для нової строки в таблиці
instructor_id:
120000
location_id:
345
start_time:
2024-06-08 12:00:00+03
end_time:
2024-06-08 13:00:00+03
max_participants:
20
price:
10
Помилка при додаванні данних 23503: INSERT или UPDATE в таблице "session" нарушает ограничение внешнего ключа "fk2"

DETAIL: Detail redacted as it may contain sensitive data. Specify 'Include Error Detail' in the connection string to include this information.
Continue y/n?
```

Запис помилкових даних(Некоректний ключ instructor_id)

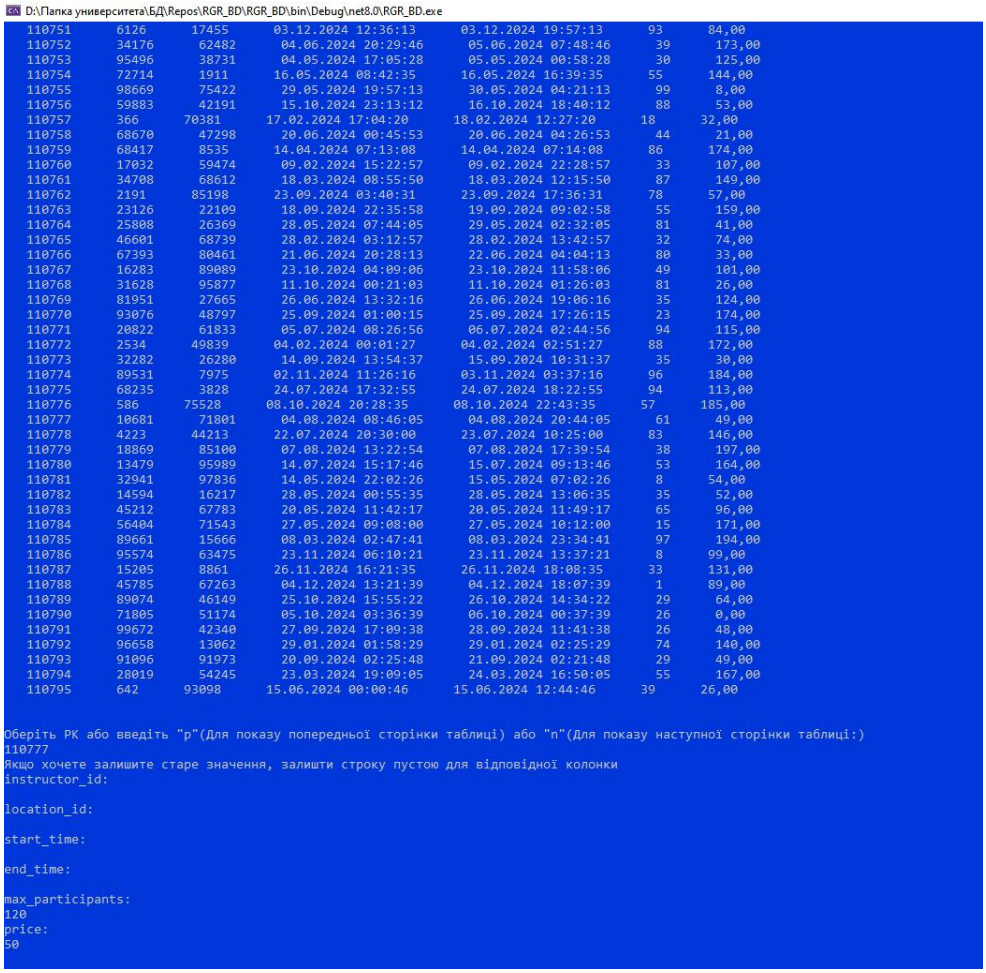
Программа дає вибір продовжити або ні.
Помилка відбулася через, неіснуючий FK(instructor_id)
Після введення y(yes) программа поверне користувача в головне меню.
Після введення n(no) программа завершить свою роботу.

Виконання оновлення даних



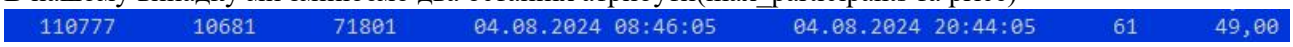
Вибір строки для зміни

Программа виводить посторінково 50 записів для обраної таблиці, між ними можливо переходити за допомогою p(previous) або n(next).
Після вводу PK користувачем йому буде запропоновано надати нові значення



Введення нових значень для запису з PK 110777

В нашому випадку ми змінюємо два останніх атрибути(max_participants та price)



Старі значення

Нові значення

Виконання видалення даних

```
D:\Папка университета\БД\Repos\RGR_BD\RGR_BD\bin\Debug\net8.0\RGR_BD.exe
 user_id    first_name    last_name    email    phone_number    date_of_registration
  901      ZHQFNSHXM      SWWOBPFGOJUQPSR      zhqfnshxm.swwobpfgojuqpsr@gmail.com      0957012377      12.07.2024 00:00:00
  902      XYDCWTSPS      CGQRWHYUXP      xydcwtsps.cgqrwhyuxp@gmail.com      0674535933      27.05.2024 00:00:00
  903      DMRL      DRKH      dmrl.drkh@gmail.com      0959361479      14.05.2024 00:00:00
  904      DIGBC      FGPFIFMZ      digbc.fgpfifmz@gmail.com      0999614542      18.12.2024 00:00:00
  905      SWDSBSGNHF      QHFNSBWU      swdsbsgnhf.qhfnsbwu@gmail.com      0993704730      28.06.2024 00:00:00
  906      TMFKNJ      BBWDLEPTFSK      tmfknj.bbwdleptfsk@gmail.com      0993585241      07.12.2024 00:00:00
  907      JQPEXA      YKKZYVFTCLD      jqpexa.ykkzyvftcld@gmail.com      0994812427      25.02.2024 00:00:00
  908      TSLLHABK      WRNEGMUNWMWSF      tsllhabk.wrnegmunwmwsf@gmail.com      0968195226      23.10.2024 00:00:00
  909      CEZJZGMZO      SINTPZ      cezjzgmzo.sintpz@gmail.com      0954134685      30.06.2024 00:00:00
  910      HYUUNBIJ      ZHJK      hyuunbij.zhjk@gmail.com      0957619148      21.09.2024 00:00:00
  911      ORFHQ      LCLVWLOFNWV      orfhq.lclvwlofnwv@gmail.com      0964131376      09.06.2024 00:00:00
  912      TEP      QJVTKNYOBNHYFSX      tep.qjvtknyobnhyfsx@gmail.com      0951282764      06.03.2024 00:00:00
  913      ILUSM      QBTPJPCBFBL      ilusm.qbtpjpcbfbl@gmail.com      0998147062      14.06.2024 00:00:00
  914      KLCEGO      VTGWSWFPVKG      klcego.vtgwswfpvkg@gmail.com      0673897720      02.01.2024 00:00:00
  915      MAPRMQDOQ      RKNMMAPJRUN      maprmqdoq.rknmmapjrun@gmail.com      0996462000      17.09.2024 00:00:00
  916      RENKJY      WRSXG      renkjy.wrsxg@gmail.com      0961679820      03.11.2024 00:00:00
  917      KYOKUT      UNIVQPFSHQR      kyokut.univqpfshqr@gmail.com      0674553179      28.07.2024 00:00:00
  918      HGIFTNIBAC      MWQAHVOZWRZRGCX      hgiftnibac.mwqahvozwrzrgcx@gmail.com      0969108536      10.09.2024 00:00:00
  919      THAHFEAP      SGMIGFFMYHIL      thahfeap.sgmigffmyhil@gmail.com      0961848611      03.06.2024 00:00:00
  920      BRSWQLZ      XJGJYWHWZBHV      brswqlz.xjgjywhwzbhv@gmail.com      0968329436      21.06.2024 00:00:00
  921      UGZ      CHMGKOBLFJVRXZ      ugz.chmgkoblfjvrxz@gmail.com      0964527092      05.11.2024 00:00:00
  922      SYT      JACROEXV      syt.jacroexv@gmail.com      0992493245      22.05.2024 00:00:00
  923      ZKUGER      XDXKNROZWVPT      zkuger.xdxknrozwvpt@gmail.com      0995424717      06.05.2024 00:00:00
  924      MRFCYKNX      EOZCNFHACGY      mrfcyknx.eozcnfhacgy@gmail.com      0968061321      21.10.2024 00:00:00
  925      QTVD      OFKUPJLWTYO      qtvd.ofkupjlwtyo@gmail.com      0965242367      26.12.2024 00:00:00
  926      QSIVL      LUCKCYGQJZM      qsivl.luckcygqjzm@gmail.com      0964442212      11.10.2024 00:00:00
  927      EDVCV      FXBBGF      edvcv.fxbbgf@gmail.com      0996882010      15.09.2024 00:00:00
  928      VQJUNATXJV      GFHAEFDPFWJLNG      vqjunatxjv.gfhaefdpfwjlng@gmail.com      0966202630      20.12.2024 00:00:00
  929      BOL      PKWRRMH      bol.pkwrrmh@gmail.com      0965520742      10.01.2024 00:00:00
  930      OMQIOLFEOI      FMJSGRLOOLXFH      omqiolfeoi.fmjsgrloolxfh@gmail.com      0678752943      11.08.2024 00:00:00
  931      IUPJ      XAWBMOWH      iupj.xawbmowh@gmail.com      0955576178      15.10.2024 00:00:00
  932      QMU      AKSFN      qmu.aksfn@gmail.com      0999107560      23.06.2024 00:00:00
  933      GYLCIT      HSFAOPF      gylcit.hsfaopf@gmail.com      0969084150      08.04.2024 00:00:00
  934      CTOWHT      OOTQQXWIU      ctowht.ootqqxwiu@gmail.com      0969197457      08.08.2024 00:00:00
  935      VCTNE      QOXDUDT      vctne.qoxdudt@gmail.com      0670911779      10.12.2024 00:00:00
  936      NZDXLGRK      OZJPRIV      nzdxlgrk.ozjpriv@gmail.com      0955016838      27.06.2024 00:00:00
  937      UPAP      RSNGYFJKVL      upap.rsngyfjkvl@gmail.com      0673633343      18.06.2024 00:00:00
  938      OLMZWY      LOBQMEYBXUAMJZU      olmzwy.lobqmeybxuamjzu@gmail.com      0673617288      10.11.2024 00:00:00
  939      SREPSXW      KGOZCUKQIOCDMAE      srepsxw.kgozcukqiocdmae@gmail.com      0956119633      10.07.2024 00:00:00
  940      CSYLHNFFH      SJCCDPCC      csylhnffh.sjccdpcc@gmail.com      0992797854      09.02.2024 00:00:00
  941      NDPNYQN      LCJHKPUG      ndpnyqn.lcjhkpug@gmail.com      0675244714      27.01.2024 00:00:00
  942      KBM      QOUSOGEHD      kbm.qousogehd@gmail.com      0997228797      28.10.2024 00:00:00
  943      RQMWJDAQE      SBIFYAMRPAYEH      rqmwjdaqe.sbifyamrpayeh@gmail.com      0950537016      03.04.2024 00:00:00
  944      IAWXSC      NRBYPTKYCVGAQRB      iawxsc.nrbyptkycvgaqrb@gmail.com      0967291525      04.05.2024 00:00:00
  945      IAUBQHVN      WDOZVKMEEN      iaubqhvn.wdozvkmeen@gmail.com      0678576327      28.12.2024 00:00:00
  946      BFMUSIA      ZLSDRFPBJKKQGM      bfmusia.zlsdrfpbjkkqgm@gmail.com      0964938637      23.06.2024 00:00:00
  947      NONVQWAPDP      NEYZFPSTER      nonvqwapdp.neyzfpster@gmail.com      0672407779      17.10.2024 00:00:00
  948      HMJCF      HYMXWPUHUM      hmjcf.hymxwpuhum@gmail.com      0956193283      06.04.2024 00:00:00
  949      OKVDCGZW      LTYEUTBLBMKBBB      okvdcgzw.ltyeutblbmkbbb@gmail.com      0679398511      20.07.2024 00:00:00
  950      TTRIOPO      FYDJDIF      ttriopo.fydjdif@gmail.com      0963868699      24.10.2024 00:00:00

Оберіть РК або введіть "p"(Для показу попередньої сторінки таблиці) або "n"(Для показу наступної сторінки таблиці:)
```

Видалення запису з user_id 927

```
D:\Папка университета\БД\Repos\RGR_BD\RGR_BD\bin\Debug\net8.0\RGR_BD.exe
 booking_id    user_id    session_id    booking_date    status
  1      927      856      07.11.2024 02:46:36      False
```

Запис з таблиці bookings, який видалиться разом с верхнім видалення через CASCADE

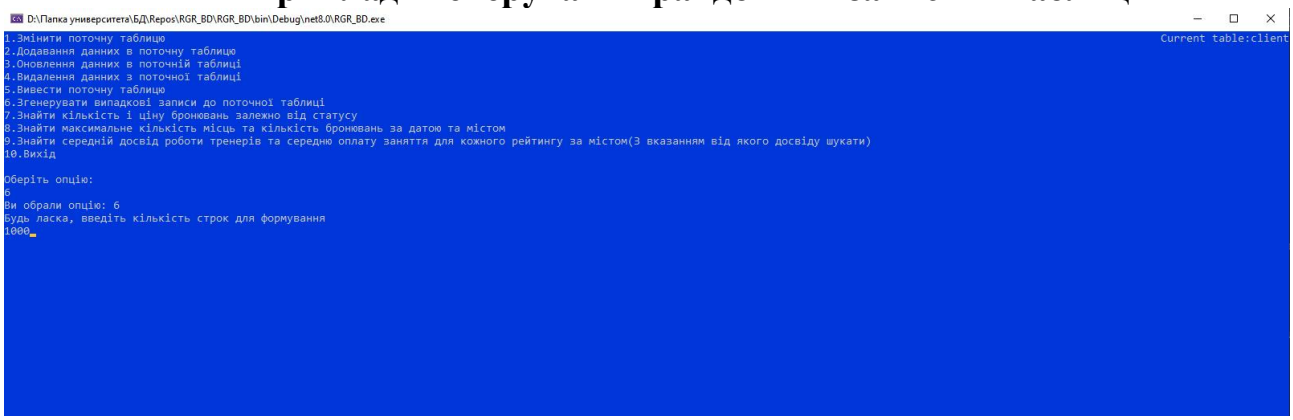
```
 926      QSIVL      LUCKCYGQJZM      qsivl.luckcygqjzm@gmail.com      0964442212      11.10.2024 00:00:00
 928      VQJUNATXJV      GFHAEFDPFWJLNG      vqjunatxjv.gfhaefdpfwjlng@gmail.com      0966202630      20.12.2024 00:00:00
```

Демонстрація видалення користувача за id 927

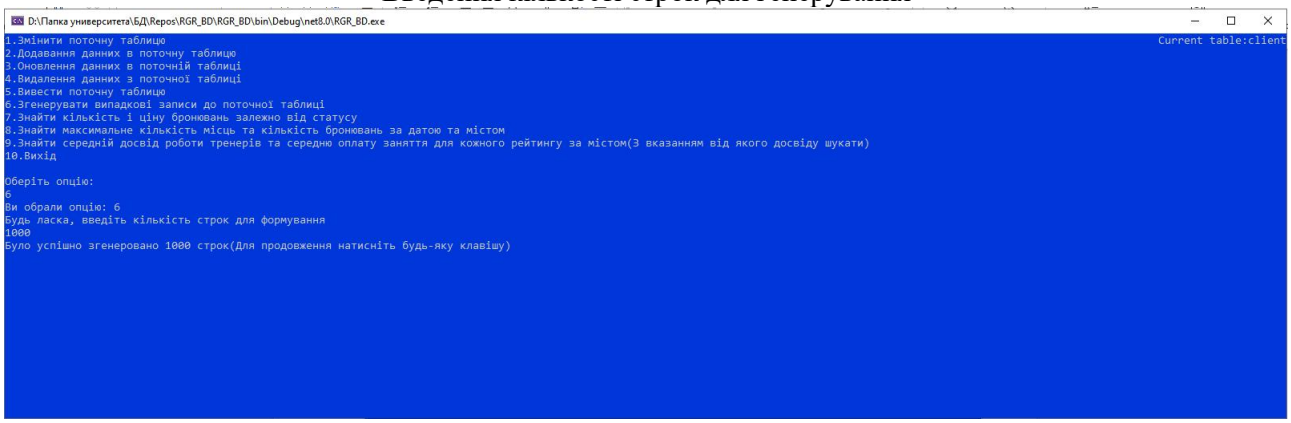
```
D:\Папка университета\БД\Repos\RGR_BD\RGR_BD\bin\Debug\net8.0\RGR_BD.exe
 booking_id    user_id    session_id    booking_date    status
  2      594      856      07.03.2024 07:08:01      False
  3      853      776      08.08.2024 17:33:47      False
  4      665      760      25.05.2024 10:33:23      True
  5      514      988      07.10.2024 07:28:46      True
  6      399      559      04.12.2024 21:00:06      False
  7      681      313      25.04.2024 00:20:52      False
  8      82      83      27.03.2024 20:56:56      False
  9      391      293      28.01.2024 01:33:39      False
  10      567      721      07.08.2024 17:44:48      False
  11      893      260      04.06.2024 07:00:38      True
  12      110      212      08.08.2024 01:33:16      False
  13      916      780      30.10.2024 12:54:58      True
  14      750      155      16.04.2024 22:46:48      False
  15      427      727      15.01.2024 17:32:07      False
  16      385      64      23.10.2024 20:14:31      False
  17      672      2      27.05.2024 18:15:58      False
```

Ми бачимо, що перший запис з таблиці bookings також видалився

Приклади генерування рандомних записів в таблиці



Введення кількості строк для генерування



Вивід про успішну генерацію

Data Output | Messages | Notifications

	user_id [PK] integer	first_name character varying (30)	last_name character varying (30)	email character varying (60)	phone_number character varying (10)	date_of_registration date
1	1	JDO	WYJUPJ	jdo.wyjupj@gmail.com	0999923683	2024-03-06
2	2	EKKHFIQTIR	SEPHCP	ekkhfiqtir.sephcp@gmail.com	0956664110	2024-07-23
3	3	IBQ	DQONPRQASFQ	ibq.dqonprqasfq@gmail.com	0993683246	2024-07-26
4	4	FTU	MWBLI	ftu.mwbli@gmail.com	0952095681	2024-05-12
5	5	KNCV	CGHZYBKRBPIPQ	kncv.cghzybkrbpipq@gmail.com	0958186454	2024-10-31
6	6	TKF	JLNM	tkf.jlnm@gmail.com	0670724961	2024-11-03
7	7	VSKZXSAF	SOWHYDHSRDKKFXO	vskzxsaf.sowhydhsrdkkfxo@gmail.com	0958680011	2024-06-26
8	8	WJE	GDJJJPZVB	wje.gdjjjpzvb@gmail.com	0955793788	2024-08-17
9	9	GYHWBNQ	MKYAFQGPGLFGFK	gyhwbnq.mkyafqgpglfgfk@gmail.com	0675814735	2024-10-21
10	10	ISA	NLDTEYN	isa.nldteyn@gmail.com	0969933103	2024-06-30
11	11	HGUVY	OESQCETCDLHEXT	hguvy.oesqcetcdlhext@gmail.com	0993387876	2024-06-16
12	12	OKOHECTFR	VZJECIGJF	okohectfr.vzjecigjf@gmail.com	0678095033	2024-04-01
13	13	OHKADEQSK	IZUBPGH	ohkadeqsk.izubpgh@gmail.com	0967139024	2024-07-26
14	14	BNTUJIW	UYEZOBLCLHZL	bntujiw.uyezoblclhzl@gmail.com	0958232147	2024-04-07
15	15	EJNSSFLID	REIPRLUVSFGS	ejnssflid.reiprluvsfgs@gmail.com	0963733474	2024-10-02
16	16	RSWBSK	XPAIJNJ	rswbsk.xpaijnj@gmail.com	0990581314	2024-01-17
17	17	OPBUWIBT	MTJGKNPLHSRLMXK	opbuwibt.mtjgknplhsrlmxk@gmail.com	0953553744	2024-09-18

Приклади рандомно згенерованих записів для таблиці client

Data Output Messages Notifications

	instructor_id [PK] integer	experiance_years integer	bio character varying (256)	rating integer
1	1	13	TQWULIJLNWFVGGTXXEN	2
2	2	41	MPGHFYCOPIDXMFYHNQGXEVYPMBBITOFTDYRXOAHHMRNFXFQOXQMDZOOGYGYOEAADHVYBCWDEIWAQTVLMQNLTHMZFLPGMWNRRKLNISTXPVXOBNUTOELWEZ	8
3	3	20	HLQCJARWBHALKJMUENQRCTOVXSEHPJWMKEZMNWMJFDIFCVYNBGFLRHPJYMDALMZJJJBQHTYMOSNCYDRMTBFMOOGNIAKSLWRYNDEHUDWTSFMCHQCFPHLRLICWTF...	1
4	4	26	ZQDZDSEUTAJWCXIEDRGS	3
5	5	42	BNZSORGZBOVDHVEPOJDPHFGSIGYCYFHCHLFKRZRV	6
6	6	39	WEFVKXNOZYTUETUCGL	6
7	7	45	TTJAVCUXHLULGCRVBJEGWCFZJELWBYTWXEPGRYVHDXTHRHYMDQLMVMORDTFINDZBFSKYRXBYTFWV	6
8	8	4	JUESDSALWCPVODKNNADOYYUFSISRIXZQVIZSNKAQDJXZIORFFLTMHQZRESKAAZZCFJDFABIYMCUZWKOL	2
9	9	31	PTVANUGAGQJGFQFGWFKGBTVKFEPPGBEVXYIFAKLPCQLMXYEPXXNVEOEDSTFYOLGWCORBVWRKHPWGKSYPOCPZQMOYDAIZXKQJVBEANRUPKWIOMYYFQOUSXSFVE	4
10	10	5	YWLTTANEDHPQMGVLGSZEVLQTOBGLGHBGRDXKRQUABRBARWCTAXULQYLPOQFDXU	10
11	11	20	WQJFTYRIBYZKEJEAWRLBWRHNBGEVIFAKWSPDAKXJKWYZTRDMWHUFPSLNIBABBXEIGGERBUTK	7
12	12	9	PMDLUHUMRKVBXQNEXMNKEYJRZPWMIQDU	2
13	13	35	KIQLOUQPOYKLKEPBBCGOFISSKHHDUXVGHHNN	10
14	14	12	SRPBUXRUPTLQKFOLQYF	3
15	15	44	XMRTOEVKJFFGOEEDEIHGUOSZFRCUXBXDIZIEBBPOJIJRUOROYHRIDVHPUSRINTDMXT	10
16	16	22	MBZIIUZHFOCLTPJOKHRT	8
17	17	27	QLNEQAHXSJJFRIXIFMFHPDPABLPRCKXALGHFLPFORXHPFIVCXYORCGBYICF	9

Приклади рандомно згенерованих записів для таблиці instructors

Data Output Messages Notifications

	name character varying (30)	address character varying (64)	city character varying (25)	capacity integer	location_id [PK] integer
1	LYJARWIFHR	WXO	Zaporizhzhia	97	1
2	MWASEGSWF	MJNYMFKMXRUWSVHWIL	Dnipro	46	2
3	XKADHVFAKLTVCB	FROUFCKAMO	Zaporizhzhia	55	3
4	KZN	HHFXPXL	Zaporizhzhia	11	4
5	URDEXDJ	ZXUMR	Kyiv	29	5
6	QZALOI	OWGGCWBKZER	Lviv	37	6
7	PUYADOFSGEWKO	WZJVNMATJXZFAEWP	Kyiv	40	7
8	PQNYDEYTLV	KSMCYQAKS	Dnipro	59	8
9	RYV	TNKTDHGQGFETVTWCGBJ	Zaporizhzhia	45	9
10	PNEGLRDWP	YFTBBFTMPBUXVJROF	Lviv	16	10
11	IMNXH	VUGCCUQHAARAWOQG	Zaporizhzhia	21	11
12	NQLCIVLSOWHRPJT	JFL	Kyiv	79	12
13	DBCJ	JXCPAGXUOVEGOSBC	Lviv	99	13
14	WYGMAA	PGVQQAJLXSDBVIYAE	Lviv	94	14
15	UIFQOONJBDR	PGZYSTZINCCQXBWLYSGF	Kyiv	34	15
16	BJPVDRABWNEQI	UDHDT	Kyiv	83	16
17	WBX	JKXDBEGJRFFRTIMCQHL	Dnipro	65	17
18	ANZIXU	KEIWUMNGJMRMADZHQ	Lviv	33	18

Приклади рандомно згенерованих записів для таблиці locations

Data Output | Messages | Notifications

	session_id [PK] integer	instructor_id integer	location_id integer	start_time timestamp with time zone	end_time timestamp with time zone	max_participants integer	price money
1	1	887	119	2024-06-02 22:58:36.603834+03	2024-06-03 07:25:36.603834+03	48	184,00 ?
2	2	958	620	2024-07-07 01:09:41.499634+03	2024-07-07 12:35:41.499634+03	23	37,00 ?
3	3	665	461	2024-04-02 23:49:58.879733+03	2024-04-03 09:59:58.879733+03	27	104,00 ?
4	4	756	748	2024-10-20 01:30:19.097631+03	2024-10-20 23:28:19.097631+03	62	89,00 ?
5	5	304	892	2024-03-31 00:12:56.49624+02	2024-03-31 07:10:56.49624+03	18	90,00 ?
6	6	291	947	2024-10-17 03:07:12.726384+03	2024-10-18 01:50:12.726384+03	12	111,00 ?
7	7	181	174	2024-08-13 12:12:39.089829+03	2024-08-14 10:26:39.089829+03	21	110,00 ?
8	8	178	359	2024-09-23 10:24:32.695166+03	2024-09-23 19:25:32.695166+03	39	3,00 ?
9	9	942	837	2024-03-07 03:54:28.240214+02	2024-03-07 17:26:28.240214+02	23	91,00 ?
10	10	32	764	2024-04-26 17:14:50.296736+03	2024-04-27 11:41:50.296736+03	29	166,00 ?
11	11	842	357	2024-02-10 13:06:44.849295+02	2024-02-10 14:29:44.849295+02	98	93,00 ?
12	12	204	549	2024-10-10 19:56:55.388431+03	2024-10-10 22:08:55.388431+03	53	17,00 ?
13	13	848	959	2024-11-28 22:22:33.428838+02	2024-11-29 10:51:33.428838+02	40	87,00 ?
14	14	569	847	2024-09-18 06:21:04.682287+03	2024-09-18 12:28:04.682287+03	20	151,00 ?
15	15	578	486	2024-09-12 00:59:41.747629+03	2024-09-12 10:04:41.747629+03	8	176,00 ?
16	16	14	21	2024-04-17 01:52:41.270371+03	2024-04-17 11:39:41.270371+03	12	42,00 ?
17	17	430	402	2024-11-11 12:00:11.277249+02	2024-11-12 03:59:11.277249+02	69	178,00 ?

Приклади рандомно згенерованих записів для таблиці session

Data Output | Messages | Notifications

	booking_id [PK] integer	user_id integer	session_id integer	booking_date timestamp with time zone	status boolean
1	2	594	856	2024-03-07 09:08:01.011005+02	false
2	3	853	776	2024-08-08 20:33:47.150349+03	false
3	4	665	760	2024-05-25 13:33:23.864843+03	true
4	5	514	988	2024-10-07 10:28:46.524412+03	true
5	6	399	559	2024-12-04 23:00:06.331443+02	false
6	7	681	313	2024-04-25 03:20:52.814384+03	false
7	8	82	83	2024-03-27 22:56:56.948623+02	false
8	9	391	293	2024-01-28 03:33:39.741676+02	false
9	10	567	721	2024-08-07 20:44:48.986174+03	false
10	11	893	260	2024-06-04 10:00:38.502064+03	true
11	12	110	212	2024-08-08 04:33:16.510649+03	false
12	13	916	780	2024-10-30 14:54:58.083029+02	true
13	14	750	155	2024-04-17 01:46:48.660987+03	false
14	15	427	727	2024-01-15 19:32:07.515642+02	false
15	16	385	64	2024-10-23 23:14:31.503577+03	false
16	17	672	2	2024-05-27 21:15:58.423128+03	false
17	18	634	779	2024-10-18 08:45:15.126217+03	true
18	19	745	207	2024-07-16 01:51:00.133437+03	true
19	20	408	757	2024-04-28 11:18:19.854778+03	false

Приклади рандомно згенерованих записів для таблиці bookings

Код процедур для генерації даних

generate_bookings

```
DECLARE
    i INT := 0;
v_random_user_id INT;
v_random_session_id INT;
v_random_booking_date TIMESTAMP WITH TIME ZONE;
v_random_status BOOLEAN;
BEGIN
    LOOP
        EXIT WHEN i >= count_i;

SELECT user_id INTO v_random_user_id
        FROM client
        ORDER BY random()
        LIMIT 1;

SELECT session_id INTO v_random_session_id
        FROM session
        ORDER BY random()
        LIMIT 1;

SELECT(timestamp '2024-01-10 20:00:00' +
   random() * (timestamp '2024-12-01 23:59:59' - timestamp '2024-01-02 00:00:00'))
             INTO v_random_booking_date;

SELECT(RANDOM() < 0.5) INTO v_random_status;

INSERT INTO bookings (user_id, session_id, booking_date, status)
        VALUES (v_random_user_id, v_random_session_id, v_random_booking_date,
v_random_status);

i:= i + 1;
END LOOP;
        END;
```

generate_client

```
        DECLARE
            i INT := 0;
            v_random_first_name TEXT;
            v_random_last_name TEXT;
               v_random_email TEXT;
               v_random_phone TEXT;
            v_random_registration_date DATE;
        BEGIN
            LOOP
                EXIT WHEN i >= count_i;

                     SELECT
                string_agg(chr(trunc(65    +    random()    *    26)::int),    '')    INTO
v_random_first_name
                   FROM
                generate_series(1, (floor(random() * (10 - 3 + 1)) + 3)::int) AS length;

                     SELECT
                string_agg(chr(trunc(65    +    random()    *    26)::int),    '')    INTO
v_random_last_name
                   FROM
                generate_series(1, (floor(random() * (15 - 3 + 1)) + 3)::int) AS length;

                     v_random_email    :=    lower(v_random_first_name    ||    '.'    ||
v_random_last_name || '@gmail.com');
```

```
            v_random_phone :=
        CASE trunc(random() * 4)
           WHEN 0 THEN '095'
           WHEN 1 THEN '099'
           WHEN 2 THEN '096'
           WHEN 3 THEN '067'
        END ||
        (SELECT string_agg(trunc(random() * 10)::text, '')
        FROM generate_series(1, 7));

            v_random_registration_date := (timestamp '2024-01-01 00:00:00' +
        random() * (timestamp '2024-12-30 00:00:00' - timestamp '2024-01-01
00:00:00'))::DATE;


        INSERT INTO client (first_name, last_name, email, phone_number,
date_of_registration)
        VALUES (v_random_first_name, v_random_last_name, v_random_email,
v_random_phone, v_random_registration_date);

        i := i + 1;
    END LOOP;
END;
```

generate_instructors

```
DECLARE
    i INT := 0;
    v_random_experiance_years INT;
    v_random_bio TEXT;
       v_random_rating INT;
BEGIN
    LOOP
        EXIT WHEN i >= count_i;

            SELECT
        string_agg(chr(trunc(65 + random() * 26)::int), '') INTO v_random_bio
           FROM
        generate_series(1, (floor(random() * (128 - 3 + 1)) + 3)::int) AS length;

            v_random_experiance_years := floor(random() * 45) + 1;

            v_random_rating := floor(random() * 10) + 1;

        INSERT INTO instructors (experiance_years, bio, rating)
        VALUES (v_random_experiance_years, v_random_bio, v_random_rating);

        i := i + 1;
    END LOOP;
END;
```

generate_locations

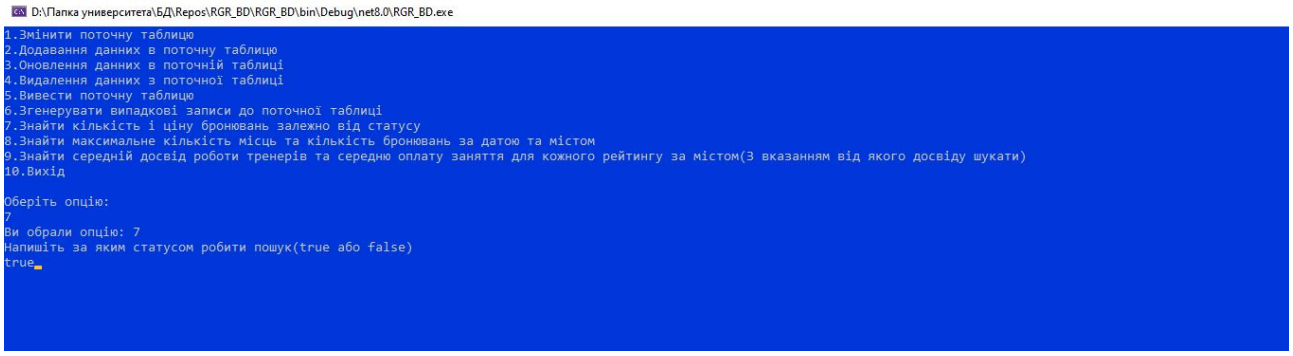
```
DECLARE
    i INT := 0;
    v_random_name TEXT;
    v_random_address TEXT;
       v_random_city TEXT;
       v_random_capacity INT;
BEGIN
    LOOP
        EXIT WHEN i >= count_i;

              SELECT
        string_agg(chr(trunc(65 + random() * 26)::int), '') INTO v_random_name
           FROM
        generate_series(1, (floor(random() * (15 - 3 + 1)) + 3)::int) AS length;

              SELECT
        string_agg(chr(trunc(65 + random() * 26)::int), '') INTO v_random_address
           FROM
        generate_series(1, (floor(random() * (20 - 3 + 1)) + 3)::int) AS length;

              v_random_capacity := floor(random() * 100) + 1;

              v_random_city :=
        CASE trunc(random() * 4)
              WHEN 0 THEN 'Kyiv'
              WHEN 1 THEN 'Lviv'
              WHEN 2 THEN 'Zaporizhzhia'
              WHEN 3 THEN 'Dnipro'
                    WHEN 4 THEN 'Khmelnytskyi'
        END;

        INSERT INTO locations (name, address, city, capacity)
        VALUES (v_random_name,v_random_address,v_random_city,v_random_capacity);

        i := i + 1;
    END LOOP;
END;
```

generate_session

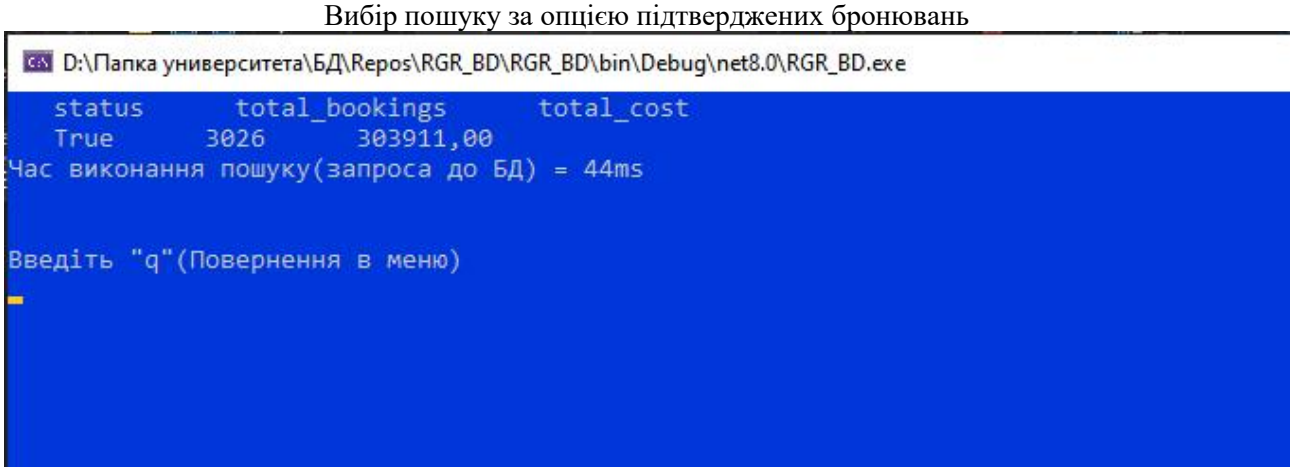
```
DECLARE
    i INT := 0;
    v_random_instructor_id INT;
    v_random_location_id INT;
       v_random_session_id INT;
    v_random_start_time TIMESTAMP WITH TIME ZONE;
       v_random_end_time TIMESTAMP WITH TIME ZONE;
    v_random_max_participants INT;
       v_random_price MONEY;
BEGIN
    LOOP
        EXIT WHEN i >= count_i;

        SELECT instructor_id INTO v_random_instructor_id
        FROM instructors
        ORDER BY random()
        LIMIT 1;

        SELECT location_id INTO v_random_location_id
        FROM locations
        ORDER BY random()
        LIMIT 1;

              v_random_max_participants := floor(random() * 100) + 1;
```

```
                v_random_price := (round(random() * 200)::numeric)::money;

                SELECT (timestamp '2024-01-10 20:00:00' +
               random() * (timestamp '2024-12-01 23:59:59' - timestamp '2024-01-02
00:00:00'))
                INTO v_random_start_time;

                v_random_end_time := v_random_start_time + (floor(random() * (24 *
60))) * interval '1 minute';


            INSERT INTO session (instructor_id, location_id, start_time, end_time,
max_participants, price)
            VALUES                                          (v_random_instructor_id,
v_random_location_id,v_random_start_time,v_random_end_time,v_random_max_participants,v
_random_price );

            i := i + 1;
        END LOOP;
    END;
```

Приклади пошукових запитів
Пошук перший(Опція - 7)



Вибір пошуку за опцією підтверджених бронювань



Виведення результатів пошуку та часу в мілісекундах

*Копія **SQL**–запиту*

```
 SELECT
    b.status,
    COUNT(b.booking_id) AS total_bookings,
    SUM(s.price) AS total_cost
FROM
    bookings b
JOIN
    session s ON b.session_id = s.session_id
WHERE
```

```
    b.status = {status}
GROUP BY
    b.status
ORDER BY
    total_bookings DESC;
```

Пошук другий(Опція - 8)

```
D:\Папка университета\БД\Repos\RGR_BD\RGR_BD\bin\Debug\net8.0\RGR_BD.exe
1.Змінити поточну таблицю
2.Додавання данних в поточну таблицю
3.Оновлення данних в поточній таблиці
4.Видалення данних з поточної таблиці
5.Вивести поточну таблицю
6.Згенерувати випадкові записи до поточної таблиці
7.Знайти кількість і ціну бронювань залежно від статусу
8.Знайти максимальне кількість місць та кількість бронювань за датою та містом
9.Знайти середній досвід роботи тренерів та середню оплату заняття для кожного рейтингу за містом(З вказанням від якого досвіду шукати)
10.Вихід

Оберіть опцію:
8
Ви обрали опцію: 8
Напишіть від якою дати шукати(Приклад: '2024-01-01')
2024-01-15
```

Введення дати від якої шукати кількість місць та кількість бронювань групуючи за містами

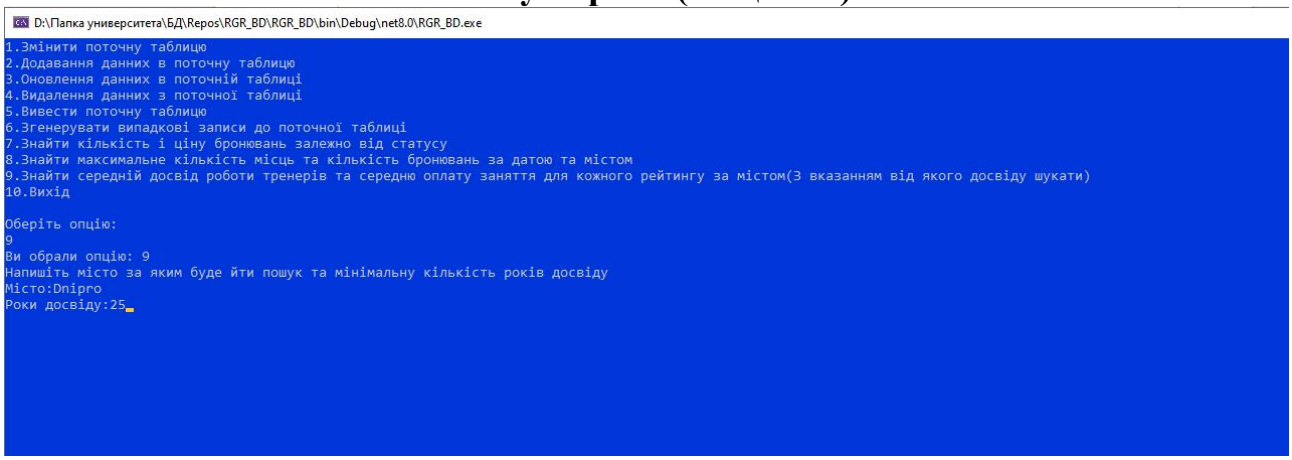
```
D:\Папка университета\БД\Repos\RGR_BD\RGR_BD\bin\Debug\net8.0\RGR_BD.exe
   count_max_participants      city      count_total_bookings
   83704      Dnipro      1625
   74791      Zaporizhzhia      1467
   73491      Kyiv      1533
   65999      Lviv      1311
Час виконання пошуку(запроса до БД) = 4ms


Введіть "q"(Повернення в меню)
```

Виведення результатів пошуку та часу в мілісекундах

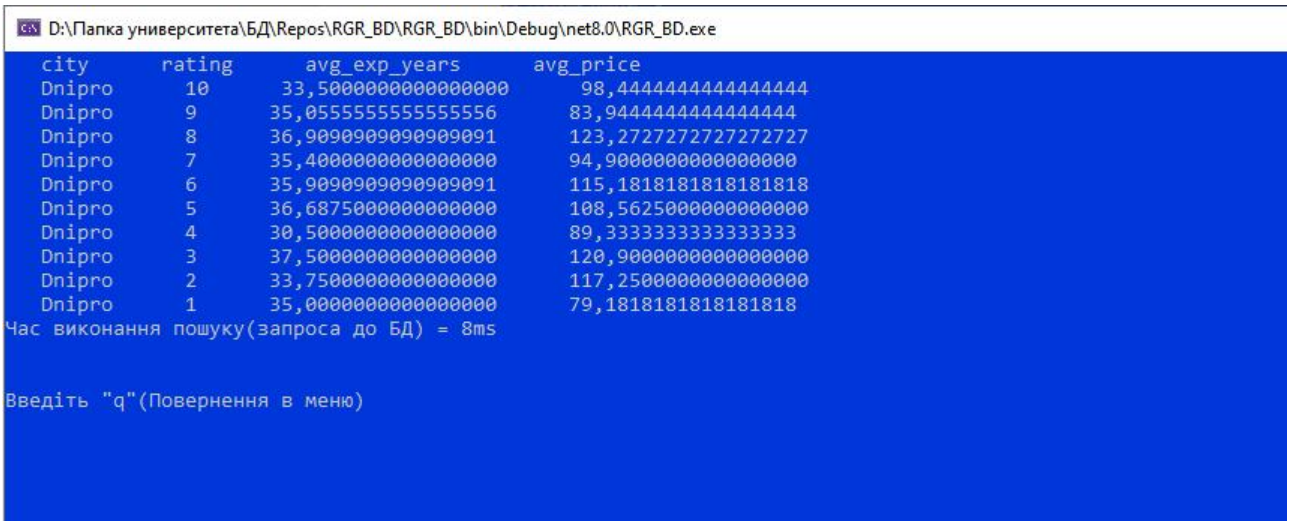
Копія SQL–запиту

```
SELECT
     SUM(s.max_participants) AS count_max_participants,
     l.city,
        count(b.booking_id) as count_total_bookings
  FROM
     session s
  JOIN
     locations l ON s.location_id = l.location_id
  JOIN
     bookings b ON s.session_id = b.session_id
  WHERE
     s.start_time >= '{start_time}'
  GROUP BY
     l.city
  ORDER BY
                        count_max_participants DESC;
```

Пошук третій(Опція - 9)



Введення міста та мінімального досвіду роботи тренерів



Виведення результатів пошуку та часу в мілісекундах

Копія SQL-запиту

```
SELECT
     l.city,
     i.rating,
     avg(i.experiance_years) as av_exp_yars,
        avg(s.price::numeric) as avg_price
  FROM
     instructors i
  JOIN
     session s ON i.instructor_id = s.instructor_id
  JOIN
     locations l ON s.location_id = l.location_id
  WHERE
        l.city = '{city}'
        and i.experiance_years >= {exp_years}
  GROUP BY
        l.city, i.rating
  ORDER BY
      i.rating DESC;
```

Програмний код модулю “Model”

Опис функцій знаходиться в файлі README.md

КОД

```
using Npgsql;
using System.Data;
using System.Diagnostics;

namespace RGR_BD
{
    public class Model
    {
        private NpgsqlConnection connection;
        public Model()
        {
            string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=Miha2004.;Database=Lab_1";
            connection = new NpgsqlConnection(connectionString);
        }
        public void CloseConnection()
        {
            try
            {   if(connection != null && connection.State == ConnectionState.Open)
                {
                    connection.Close();
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                Thread.Sleep(1000);
            }
        }
        public bool AddDataToTableModel(List<(string Column, string Value)> values,string table_name)
        {
            try
            {
                connection.Open();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при підключенні до бази данних");
                return true;
            }
            Dictionary<string, string> columnTypes = GetColumnTypes(table_name);
            try
            {
                List<string> setClauses = new List<string>();
                List<string> setValues = new List<string>();
                foreach (var column in values)
                {
                    setClauses.Add($"{column.Column}");
                    setValues.Add($"@{column.Column}");
                }
                string setClause_str = string.Join(",", setClauses);
```

```
                    string setValues_str = string.Join(",", setValues);
                    string query = $"INSERT INTO {table_name} ({setClause_str}) VALUES
({setValues_str});";
                    using (var cmd = new NpgsqlCommand(query, connection))
                    {
                        foreach (var (Column, Value) in values)
                        {
                            cmd.Parameters.AddWithValue($"@{Column}",
GetConvertedValues(columnTypes, Column, Value));
                        }
                        cmd.ExecuteNonQuery();
                    }

                }
                catch (Exception ex)
                {
                    Console.WriteLine("Помилка при додаванні данних " + ex.Message);
                    return true;
                }
                try
                {
                    connection.Close();
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                    return true;
                }

                return false;
            }
            private Dictionary<string, string> GetColumnTypes(string tableName)
            {
                var columnTypes = new Dictionary<string, string>();

                string query = @$"
                SELECT column_name, data_type
                FROM information_schema.columns
                WHERE table_name = '{tableName}'";

                using (var cmd = new NpgsqlCommand(query, connection))
                {
                    using (var reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            string columnName = reader.GetString(0);
                            string dataType = reader.GetString(1);
                            columnTypes[columnName] = dataType;
                        }
                    }
                }

                return columnTypes;
            }
            public string GetPrimaryKeyColumn(string table_name)
            {
                string pk_str = string.Empty;
                try
                {
                    connection.Open();
                }
                catch (Exception ex)
```

```
            {
                Console.WriteLine("Помилка при підключенні до бази данних");
                return pk_str;
            }
            try
            {
                string query = @"
                SELECT kcu.column_name
                FROM information_schema.table_constraints AS tc
                JOIN information_schema.key_column_usage AS kcu
                    ON kcu.constraint_name = tc.constraint_name
                WHERE tc.table_name = @TableName
                  AND tc.constraint_type = 'PRIMARY KEY';";
                using (var cmd = new NpgsqlCommand(query, connection))
                {
                    cmd.Parameters.AddWithValue("@TableName", table_name);
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            pk_str = reader["column_name"].ToString();
                        }
                        else
                        {
                            pk_str = string.Empty;
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при находженні первинного ключа ");
                return pk_str;
            }
            try
            {
                connection.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                return pk_str;
            }
            return pk_str;
        }
        private object GetConvertedValues(Dictionary<string, string> columnTypes, string column,
string value)
        {
            object convertedValue = null;
            if (columnTypes.TryGetValue(column, out var dataType))
            {
                switch (dataType.ToLower())
                {
                    case "integer":
                        convertedValue = Convert.ToInt32(value);
                        break;
                    case "character varying":
                    case "text":
                        convertedValue = value;
                        break;
                    case "money":
                    case "numeric":
                        convertedValue = Convert.ToDecimal(value);
```

```
                        break;
                    case "boolean":
                        convertedValue = Convert.ToBoolean(value);
                        break;
                    case "date":
                        convertedValue = Convert.ToDateTime(value);
                        break;
                    case "timestamp with time zone":
                        if (DateTimeOffset.TryParse(value, out var parsedValue))
                        {
                            convertedValue = parsedValue.ToUniversalTime();
                        }
                        else
                        {
                            throw new FormatException($"Невірний формат дати: {value}");
                        }
                        break;
                    default:
                        throw new ArgumentException($"Невідомий тип данних: {dataType}");
                }
            }
            return convertedValue;
        }
        public bool UpdateDataInTable(List<(string Column, string Value)> values_res, string table_name, int pk)
        {
            string pk_str_column = GetPrimaryKeyColumn(table_name);
            try
            {
                connection.Open();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при підключенні до бази данних");
                return true;
            }
            Dictionary<string, string> columnTypes = GetColumnTypes(table_name);
            try
            {
                List<string> setClauses = new List<string>();
                foreach (var column in values_res) {
                    setClauses.Add($"{column.Column} = @{column.Column}");
                }
                string setClause_str = string.Join(",", setClauses);
                string query = $"UPDATE {table_name} SET {setClause_str} WHERE {pk_str_column} = {pk};";
                using (var cmd = new NpgsqlCommand(query, connection))
                {
                    foreach (var (Column, Value) in values_res)
                    {
                        cmd.Parameters.AddWithValue($"@{Column}", GetConvertedValues(columnTypes, Column, Value));
                    }
                    cmd.ExecuteNonQuery();
                }

            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при зміненні данних " + ex.Message);
                return true;
            }
            try
```

```
            {
                connection.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                return true;
            }

            return false;
        }
        public (bool error, List<string> ColumnsName) GetColumnNameOfTable(string table_name)
        {
            List<string> columnsname = new List<string>();
            try
            {
                connection.Open();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при підключенні до бази данних");
                return (true, columnsname);
            }
            try
            {
                string query = $"SELECT COLUMN_NAME FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = '{table_name}' ORDER BY
ORDINAL_POSITION;";
                using (var cmd = new NpgsqlCommand(query, connection))
                {
                    using (var reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            string columnName = reader.GetString(0);
                            columnsname.Add(columnName);
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при отриманні списку колонок");
                return (true, columnsname);
            }
            try
            {
                connection.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                return (true, columnsname);
            }
            return (false, columnsname);
        }
        public (bool error, List<List<string>> rows) GetRowsOfTable(string table_name, int
page_num)
        {
            List<List<string>> rows = new List<List<string>>();
            int pageSize = 50;
            int startRow = (page_num - 1) * pageSize;
            string pk_str_column = GetPrimaryKeyColumn(table_name);
```

```
                try
                {
                    connection.Open();
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Помилка при підключенні до бази данних");
                    return (true, rows);
                }
                try
                {
                    string query = $"SELECT * FROM {table_name} ORDER BY {pk_str_column} LIMIT {pageSize} OFFSET {startRow};";
                    using (var cmd = new NpgsqlCommand(query, connection))
                    {
                        using (var reader = cmd.ExecuteReader())
                        {
                            while (reader.Read())
                            {
                                List<string> row = new List<string>();
                                for(int i = 0; i < reader.FieldCount; i++)
                                {
                                    row.Add(reader.GetValue(i).ToString());
                                }
                                rows.Add(row);
                            }
                        }
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Помилка при отриманні списку строк" + ex.Message);
                    return (true, rows);
                }
                try
                {
                    connection.Close();
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                    return (true, rows);
                }
                return (false, rows);
            }
            public (bool error, List<string> tables) GetAllTables()
            {
                List<string> tables = new List<string>();
                try
                {
                    connection.Open();
                }
                catch (Exception ex) {
                    Console.WriteLine("Помилка при підключенні до бази данних");
                    return (true, tables);
                }
                try
                {
                    string query = "SELECT table_name FROM information_schema.tables WHERE table_schema = 'public';";
                    using (var cmd = new NpgsqlCommand(query, connection))
                    {
                        using (var reader = cmd.ExecuteReader())
```

```
                {
                    while (reader.Read())
                    {
                        tables.Add(reader.GetString(0));
                    }
                }
            }
        }
        catch (Exception ex) {
            Console.WriteLine("Помилка при отриманні списку таблиць");
            return (true, tables);
        }
        try
        {
            connection.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при закритті з'єднання з базою данних");
            return (true, tables);
        }
        return (false, tables);
    }
    public bool DeleteDataOfTable(string table_name, int pk, string pk_str)
    {
        try
        {
            connection.Open();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при підключенні до бази данних");
            return (true);
        }
        try
        {
            string query = $"DELETE FROM {table_name} WHERE {pk_str} = {pk}";
            using (var cmd = new NpgsqlCommand(query, connection))
            {
                cmd.ExecuteNonQuery();
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при видаленні рядка");
            return (true);
        }
        try
        {
            connection.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при закритті з'єднання з базою данних");
            return (true);
        }
        return false;
    }
    public bool GenerateDataToCurrentTable(string proc_name, int count_rows)
    {
        try
        {
            connection.Open();
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при підключенні до бази данних");
            return true;
        }
        try
        {
            string query = $"CALL {proc_name}({count_rows})";
            using (var cmd = new NpgsqlCommand(query, connection))
            {
                cmd.CommandTimeout = 0;
                cmd.ExecuteNonQuery();
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при генерації випадкових данних" + ex.Message);
            return true;
        }
        try
        {
            connection.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при закритті з'єднання з базою данних");
            return true;
        }
        return false;
    }
    public (bool error, List<List<string>> str_res, long time) SearchFirst(string status)
    {
        List<List<string>> rows = new List<List<string>>();
        long executionTimeMs = 0;
        try
        {
            connection.Open();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при підключенні до бази данних");
            return (true, rows, 0);
        }
        try
        {
            string query = $@"SELECT
                                    b.status,
                                    COUNT(b.booking_id) AS total_bookings,
                                    SUM(s.price) AS total_cost
                                FROM
                                    bookings b
                                JOIN
                                    session s ON b.session_id = s.session_id
                                WHERE
                                    b.status = {status}
                                GROUP BY
                                    b.status
                                ORDER BY
                                    total_bookings DESC;";
            Stopwatch stopwatch = Stopwatch.StartNew();
            using (var cmd = new NpgsqlCommand(query, connection))
            {
```

```
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        List<string> row = new List<string>();
                        for (int i = 0; i < reader.FieldCount; i++)
                        {
                            row.Add(reader.GetValue(i).ToString());
                        }
                        rows.Add(row);
                    }
                }
            }
            stopwatch.Stop();
            executionTimeMs = stopwatch.ElapsedMilliseconds;
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при пошукову запиті №1" + ex.Message);
            return (true, rows, 0);
        }
        try
        {
            connection.Close();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при закритті з'єднання з базою данних");
            return (true, rows, 0);
        }
        return (false, rows, executionTimeMs);
    }
    public (bool error, List<List<string>> str_res, long time) SearchSecond(string start_time)
    {
        List<List<string>> rows = new List<List<string>>();
        long executionTimeMs = 0;
        try
        {
            connection.Open();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Помилка при підключенні до бази данних");
            return (true, rows, 0);
        }
        try
        {
            string query = $@"SELECT
                                    SUM(s.max_participants) AS count_max_participants,
                                    l.city,
                                        count(b.booking_id) as count_total_bookings
                                FROM
                                    session s
                                JOIN
                                    locations l ON s.location_id = l.location_id
                                JOIN
                                    bookings b ON s.session_id = b.session_id
                                WHERE
                                    s.start_time >= '{start_time}'
                                GROUP BY
                                    l.city
                                ORDER BY
                                    count_max_participants DESC;";
```

```
                Stopwatch stopwatch = Stopwatch.StartNew();
                using (var cmd = new NpgsqlCommand(query, connection))
                {
                    using (var reader = cmd.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            List<string> row = new List<string>();
                            for (int i = 0; i < reader.FieldCount; i++)
                            {
                                row.Add(reader.GetValue(i).ToString());
                            }
                            rows.Add(row);
                        }
                    }
                }
                stopwatch.Stop();
                executionTimeMs = stopwatch.ElapsedMilliseconds;
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при пошукову запиті №1" + ex.Message);
                return (true, rows, 0);
            }
            try
            {
                connection.Close();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                return (true, rows, 0);
            }
            return (false, rows, executionTimeMs);
        }
        public (bool error, List<List<string>> str_res, long time) SearchThird(string city, string
exp_years)
        {
            long executionTimeMs = 0;
            List<List<string>> rows = new List<List<string>>();
            try
            {
                connection.Open();
            }
            catch (Exception ex)
            {
                Console.WriteLine("Помилка при підключенні до бази данних");
                return (true, rows, 0);
            }
            try
            {
                string query = $@"SELECT
                                        l.city,
                                        i.rating,
                                        avg(i.experiance_years) as av_exp_yars,
                                            avg(s.price::numeric) as avg_price
                                    FROM
                                        instructors i
                                    JOIN
                                        session s ON i.instructor_id = s.instructor_id
                                    JOIN
                                        locations l ON s.location_id = l.location_id
                                    WHERE
```

```
                                        l.city = '{city}'
                                        and i.experiance_years >= {exp_years}
                                    GROUP BY
                                        l.city, i.rating
                                    ORDER BY
                                        i.rating DESC;";
                        Stopwatch stopwatch = Stopwatch.StartNew();
                        using (var cmd = new NpgsqlCommand(query, connection))
                        {
                            using (var reader = cmd.ExecuteReader())
                            {
                                while (reader.Read())
                                {
                                    List<string> row = new List<string>();
                                    for (int i = 0; i < reader.FieldCount; i++)
                                    {
                                        row.Add(reader.GetValue(i).ToString());
                                    }
                                    rows.Add(row);
                                }
                            }
                        }
                        stopwatch.Stop();
                        executionTimeMs = stopwatch.ElapsedMilliseconds;
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine("Помилка при пошукову запиті №1" + ex.Message);
                        return (true, rows, 0);
                    }
                    try
                    {
                        connection.Close();
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine("Помилка при закритті з'єднання з базою данних");
                        return (true, rows, 0);
                    }
                    return (false, rows, executionTimeMs);
                }
            }
        }
```