МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

КУРСОВА РОБОТА

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: “Система аналізу рентабельності товарів інтернет-магазинів взуття”

Студент
групи КП-02 **Слободзян Максим Вікторович** _____
(підпис)

Викладач
к.т.н, доцент кафедри
СПіСКС **Петрашенко А.В.** _____
(підпис)

Захищено з оцінкою _____

Київ – 2021

Анотація

Метою розробки даного курсового проєкту є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проєктної документації.

Темою курсового проєкту є розроблення системи аналізу рентабельності товарів інтернет-магазинів взуття.

Курсова робота складається з бази даних онлайн-магазину взуття (товарів, клієнтів, відгуків та замовлень) та консольного додатку для взаємодії з нею, який дозволяє виконувати певні операції над нею (запис, видалення, оновлення, читання, псевдовипадкова генерація, аналіз даних тощо).

Результатами розробки проєкту став консольний додаток для роботи з базою даних інтернет-магазину, який також забезпечує її оновлення та підтримання. Також було здобуто цінні навички розробки програмного забезпечення, що орієнтоване на взаємодію з реляційними базами даних.

Зміст

Вступ

В сучасному світі, підприємцям, що мають інтернет-магазини, дуже важливо отримувати інформацію про споживання товарів, вподобання клієнтів, свою цільову категорію. Доволі очевидно, що кожен онлайн-магазин зараз має свою електронну базу даних. Проте, інколи інформація, що у ній зберігається, жодним чином не аналізується, а це може призвести до втрати прибутків та неефективного ведення бізнесу.

Для обробки такої інформації було розроблено додаток, який має на меті вирішення проблеми аналізу, керування та фільтрації даних, що містяться в базах даних інтернет-магазинів.

Окрім розробленого додатку, було створено відповідну базу даних, у якій була зібрана основна інформація. База має такі сутності:

- Предмети взуття;
- Клієнти;
- Замовлення;
- Відгуки.

База даних була спроєктована відповідно до третьої нормальної форми (3 НФ), швидкість її роботи була оптимізована шляхом створення індексів. Безпеку даних забезпечують доданий механізм реплікації та резервне копіювання даних. Розроблений консольний додаток надає змогу управління інформацією та аналізу даних у базі.

Крім цього, було створено підсистему попередньої обробки даних, що включає в себе засоби генерації даних. Таким чином, з’являється можливість одразу протестувати розроблений додаток на псевдовипадково згенерованих сутностях та детально дослідити процеси аналізу даних.

1. Аналіз інструментарію для виконання курсової роботи

1.1 *Обґрунтування вибору СУБД*

Для виконання даної роботи у якості системи керування базами даних було обрано PostgreSQL. Такий вибір був зроблений у зв'язку з наступними перевагами:

- Об'єктно-реляційна СУБД;
- Підтримка бази даних необмеженого розміру;
- Підтримка великої кількості типів даних;
- Надійні та потужні механізми реплікацій;
- Легка розширюваність.

1.2 *Обґрунтування вибору бібліотек*

Для взаємодії з базою даних було обрано бібліотеку Npgsql.EntityFramework, адже вона:

- Підходить для зручного використання у мові програмування C#;
- Пришвидшує процес написання запитів;
- Надає простий та зрозумілий синтаксис;
- Має чітку та зрозумілу документацію з хорошими прикладами;
- Розроблена спеціально для PostgreSQL.

Для візуалізації результатів аналізу даних було обрано бібліотеку ScottPlot, оскільки:

- Вона надає зручний інтерфейс для побудови графічних об'єктів;
- Більшість графіків створюються незначними фрагментами коду;
- Наявна чітка та зрозуміла документація з хорошими прикладами побудови різних графіків та діаграм;
- Є можливість вибору серед різноманітних типів графіків та діаграм.

2. Структура бази даних

База даних проєкту включає в себе 5 таблиць з наступними полями:

1. Footwear - предмети взуття:
    - id (integer) - первинний ключ;
    - name (text) - назва продукту;
    - brand (text) - назва бренду, що характеризує продукт;
    - cost (integer) - ціна продукту;
    - country_of_origin (text) - назва країни-виробника продукту.
2. Clients - клієнти інтернет-магазину:
    - id (integer) - первинний ключ;
    - fullname (text) - ім’я клієнта;
    - birthday_date (date) - дата народження клієнта;
    - email (text) - електронна скринька клієнта.
3. Orders - замовлення клієнтів інтернет-магазину:
    - id (integer) - первинний ключ;
    - client_id (integer) - зовнішній ключ до таблиці Clients, який вказує на замовника, прив’язаний до поля id;
    - checkout_date (date) - дата замовлення продуктів клієнтом;
    - payment_method (text) - спосіб оплати замовлення.
4. Order_items - спеціальна таблиця, створена для приведення бази даних до третьої нормальної форми (3 НФ), є допоміжною таблицею для реалізації зв’язку Many to Many між таблицями Orders та Footwear:
    - id (integer) - первинний ключ;
    - product_id (integer) - зовнішній ключ до таблиці Footwear, який вказує на річ замовлення, прив’язаний до поля id;

- order_id (integer) - зовнішній ключ до таблиці Orders, який вказує на замовлення, прив’язаний до поля id;

5. Reviews - відгуки клієнтів онлайн-магазину:
    - id (integer) - первинний ключ;
    - content (text)
    - rating (integer)
    - product_id (integer) - зовнішній ключ до таблиці Footwear, який вказує на рецензійний продукт, прив’язаний до поля id;
    - client_id (integer) - зовнішній ключ до таблиці Clients, який вказує на автора, прив’язаний до поля id;
    - created_at (date) - дата створення відгуку;

3. Опис програмного забезпечення

3.1 Загальна структура програмного забезпечення

Розроблене програмне забезпечення складається з таких компонентів:

1. База даних, що зберігає інформацію про онлайн-магазин;
2. Засоби генерації даних до таблиць бази даних, при чому генерація на основі датасетів або ж псевдовипадкова;
3. Засоби CRUD-функціоналу;
4. Засоби пошуку, фільтрації та валідації;
5. Засоби реплікації та резервування даних;
6. Засоби статистичного аналізу даних;
7. Засоби візуального зображення даних.

3.2 Опис модулів програмного забезпечення

Розроблене програмне забезпечення розбите на наступні модулі:

1. Model

   Цей модуль напряму взаємодіє з базою даних. В цьому модулі знаходяться ORM класи сутностей, клас контексту, що відповідає за ORM взаємодію з базою та її схему.
2. View

   Даний модуль відповідає за мінімалістичний консольний інтерфейс взаємодії з користувачем, а саме за вивід запрошеної інформації у консоль.
3. Controller

Цей модуль пов’язує попередні модулі Model та View. Він є ніби “мостом” між цими модулями, передаючи дані в модуль Model, приймаючи та передаючи інформацію на вивід до консолі з Model до View.

4. DataGenerator

   Це модуль генерації сутностей у базу даних. Генерація відбувається як псевдовипадкова, так і на основі датасетів. Виступає окремою утилітою.

5. Visualization

   Модуль, який генерує статистичні таблиці. Методи модуля Visualization викликаються модулем View. Також модуль Visualization має додатковий метод генерації звіту про певну пару взуття.

3.3 Опис основних алгоритмів роботи

Процес генерація даних був влаштований так, щоб генерувались максимально відповідні дані для обраної предметної галузі. Генерація здійснювалась на основі датасетів, взятих з Інтернету. Також присутня псевдовипадкова генерація сутностей.

Для полегшення взаємодії з базою даних та структуруванням запитів до неї було розроблено спеціальні репозиторії, які були об’єднані у клас сервісу.

Дані статистичного аналізу зображуються на спеціальних графіках та діаграмах, які потім зберігаються у відповідну директорію.

Генерація звіту про певний товар відбувається програмним заповненням зразка документу, після чого він також зберігається у певну директорію.

4. Аналіз функціонування засобів реплікації

Для реплікації було використано механізм логічної реплікації. Він полягає у схемі публікація-підписник. Основна база даних надає публікацію для всіх таблиць разом з реплікаційним слотом. Додаткова, або резервна, база даних створює підписку до публікації вказаної вище та прив’язується до створеного реплікаційного слоту.

Перевагами такого виду реплікації є:

- Реплікація між різними основними версіями PostgreSQL;
- Реплікація між екземплярами PostgreSQL на різних платформах (Linux, Windows)
- Об’єднання декількох баз даних в одну
- Розділення підмножини бази даних між декількома базами даних
- Передача підписниками інкрементальних змін в одній базі даних чи підмножині баз даних, коли вони відбуваються.

База даних підписника функціонує так само, як і будь-який інший екземпляр бази PostgreSQL, і може стати публікуючою, якщо створити публікації в ній.

5. Аналіз функціонування засобів резервування/відновлення

Резервне копіювання необхідне для забезпечення безпечного та швидкого відновлення даних у разі втрати їх із бази даних. Тип резервного копіювання, який було реалізовано - повне (щоразу копіюються повністю всі дані). Перевага такого різновиду резервного копіювання полягає у тому, що не потрібно об’єднувати різні файли для відновлення, натомість відновлюється все з одного файлу, за рахунок чого відновлення є помітно швидшим порівняно з іншими видами резервного копіювання. Окрім цього, копія не залежить від версії PostgreSQL. Було обрано стандартну утиліту PostgreSQL pg_dump. (Приклади команд резервації та відновлення у Додатках)

6. Аналіз результатів підвищення швидкодії виконання запитів.

Для підвищення швидкодії пошукових запитів було обрано індекси Hash та B-tree. Індексування Hash було застосовано до таких полів: fullname таблиці Clients, name таблиці Footwear. Індексування B-tree було застосовано до таких полів: rating, created_at таблиці Reviews, checkout_date таблиці Orders.

У разі невеликої кількості сутності індекси можуть ставати неефективними, адже їх алгоритми будуть довшими, ніж звичайний лінійний пошук. У зв’язку з цим індекси і застосовуються для великих баз даних.

Команди, що ініціалізують створення індексів:

```
CREATE INDEX IF NOT EXISTS cl_hash ON clients USING hash(fullname)
```

```
CREATE INDEX IF NOT EXISTS fw_hash ON footwear USING hash(name)
```

```
CREATE INDEX IF NOT EXISTS rw_btr ON reviews USING btree(rating, created_at)
```

```
CREATE INDEX IF NOT EXISTS or_btr ON orders USING btree(checkout_date)
```

Результати підвищення швидкодії виконання запитів розміщені у Додатках - відповідні графіки звичайних пошуків та пошуків з індексами.

7. Опис результатів аналізу предметної галузі

У розробленому консольному додатку наявний наступний аналіз даних, що містяться у базі:

- Для аналізу загального рейтингу продукту серед оцінок клієнтів було використано запит, який допомагає сформувати графік розподілу оцінки продукту клієнтами. Такий графік дозволяє виявити нерентабельні товари а також звернути увагу на слабкі місця позицій з асортименту;
- Для аналізу вікової категорії товару було використано запит, який у вигляді секторної діаграми демонструє зацікавленість клієнтів кожної з 5-ти вікових груп, які були обрані майже у аналогії з реальним світом. Така діаграма дозволяє визначитись з цільовою аудиторією товару та продумати маркетингові ходи на майбутнє;
- Для аналізу доходів онлайн-магазину було використано запит, який збирає інформацію про всі продажі товарів за певний рік та формує ламану доходів. За допомогою такого графіку можна приблизно спрогнозувати подальший розвиток ситуації - будуть доходи магазину зростати чи спадати;
- Додатково було реалізовано формування звіту про певну позицію з асортиментів товару онлайн-магазину. Звіт містить у собі повну інформацію про пару взуття, її середній рейтинг серед клієнтів, відгуки з найбільшою та найменшою оцінкою а також графік розподілу оцінки продукту клієнтами. Звіт заповнюється програмно шляхом архівування та розархівуванням документу-зразка та заміною відповідних значень у ньому.

Приклади графіків, діаграм та звіту наведені у Додатках.

Висновки

Під час виконання даної курсової роботи виконано таку роботу та отримано такі результати:

- Розроблено базу даних, яка відповідає 3-ій нормальній формі та організована максимально зручно та просто;
- Налаштовано реплікацію виду logical replication;
- Реалізовано швидке повне резервування даних та подальше їх відновлення шляхом створення файлу з SQL-командами, які при виконанні на сервері створюють БД у тому самому вигляді, яка і була на момент створення резервної копії;
- Розроблено генерацію даних на основі датасетів та псевдовипадкову генерацію, значної уваги було приділено максимальній відповідності сутностей до реальних аналогій сучасного світу;
- Підвищено швидкодію запитів до бази даних шляхом індексування деяких полів таблиць БД;
- Розроблено засоби статистичного аналізу даних із бази, які представлені у вигляді графіків, діаграм та звітів для наочної демонстрації результатів аналізу та полегшення встановлення відповідних висновків щодо проведеного дослідження.

У результаті виконання даної курсової роботи було досягнуто поставленої мети, а саме: набуто практичні навички розробки сучасного програмного забезпечення, що взаємодіє з реляційними базами даних, а також здобуто навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. Також я оволодів основами використання СУБД та інструментальними засобами підтримки розробки додатків для подібних баз даних.

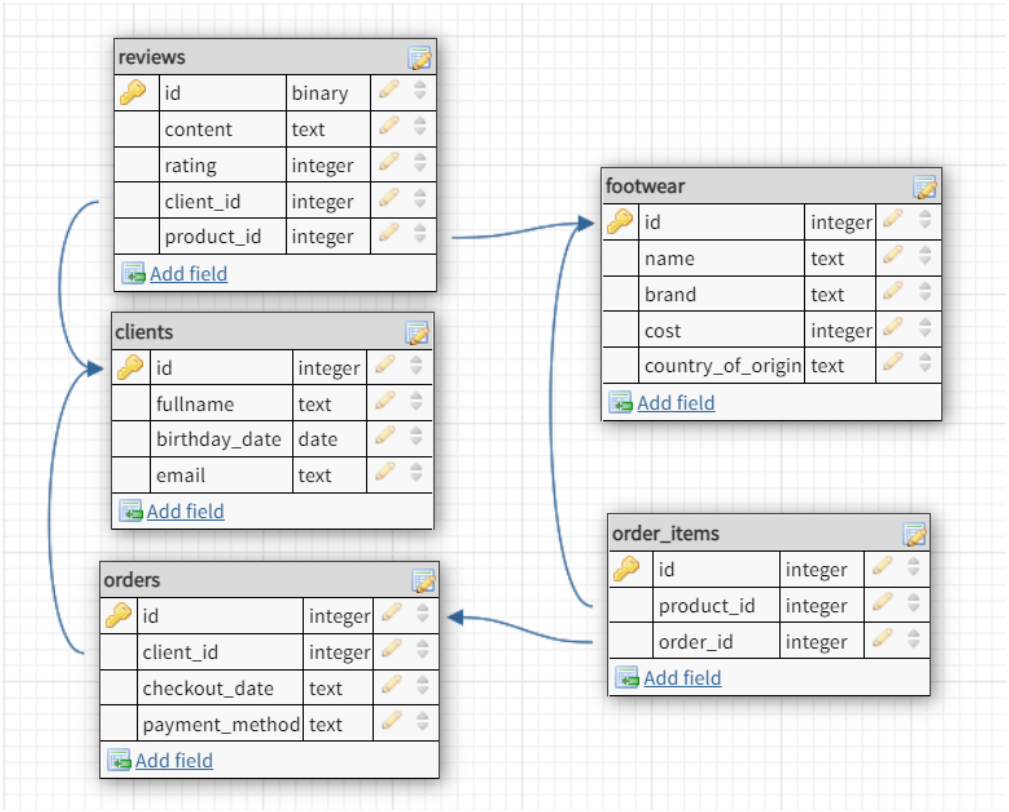
Література

1. PostgreSQL 10.19 Documentation [Електронний ресурс]
   Режим доступу: https://www.postgresql.org/docs/10/index.html
2. Npgsql documentation [Електронний ресурс]
   Режим доступу: https://www.npgsql.org/
3. Entity Framework Documentation [Електронний ресурс]
   Режим доступу: https://docs.microsoft.com/en-us/ef/
4. ScottPlot.NET documentation [Електронний ресурс]
   Режим доступу: https://scottplot.net/
5. Logical replication documentation [Електронний ресурс]
   Режим доступу:
   https://www.postgresql.org/docs/10/logical-replication.html
6. pg_dump documentation [Електронний ресурс]
   Режим доступу: https://postgrespro.ru/docs/postgresql/9.6/app-pgdump
7. Hash індексація [Електронний ресурс]
   Режим доступу: https://habr.com/ru/company/postgrespro/blog/442776/
8. B-Tree індексація [Електронний ресурс]
   Режим доступу: https://www.postgresql.org/docs/11/btree.html

Додатки

А. Графічні матеріали

1. Структура бази даних



2. Команда резервування даних:

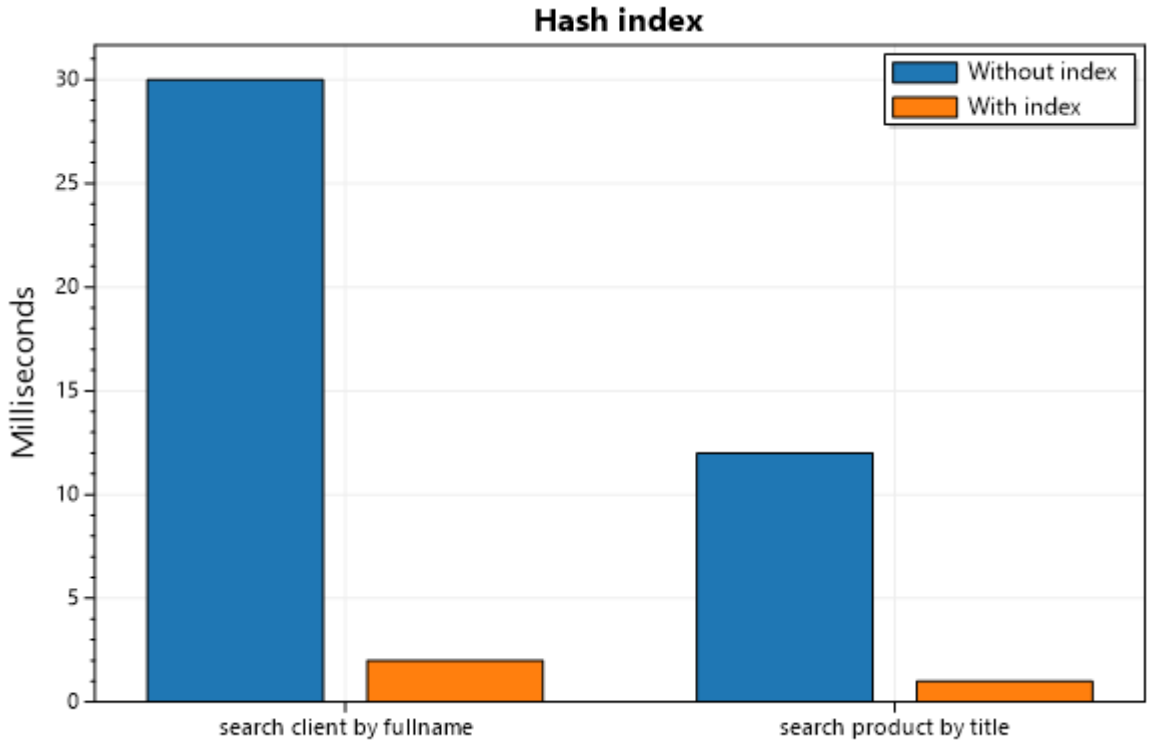
```
pg_dump -U postgres -W online_shop >
C:\Users\Макс\myprojects\db_coursework\data\backup\backup.s
ql
```

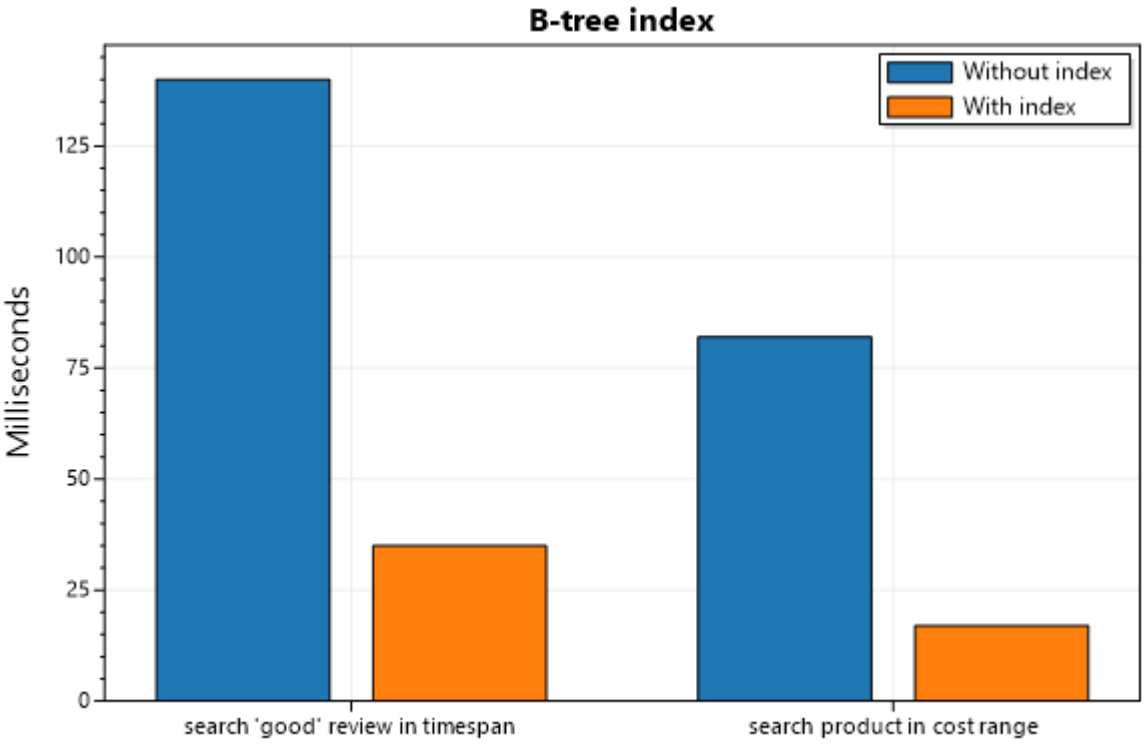
3. Команди відновлення даних:

```
psql -U postgres -c “UPDATE pg_database SET datallowconn =
'false' WHERE datname = 'online_shop'”

psql -U postgres -c “SELECT pg_terminate_backend(pid) FROM
pg_stat_activity WHERE datname = 'online_shop'”

psql -U postgres -c “DROP DATABASE online_shop”
psql -U postgres -c “CREATE DATABASE online_shop”
psql -U postgres -W online_shop <
```

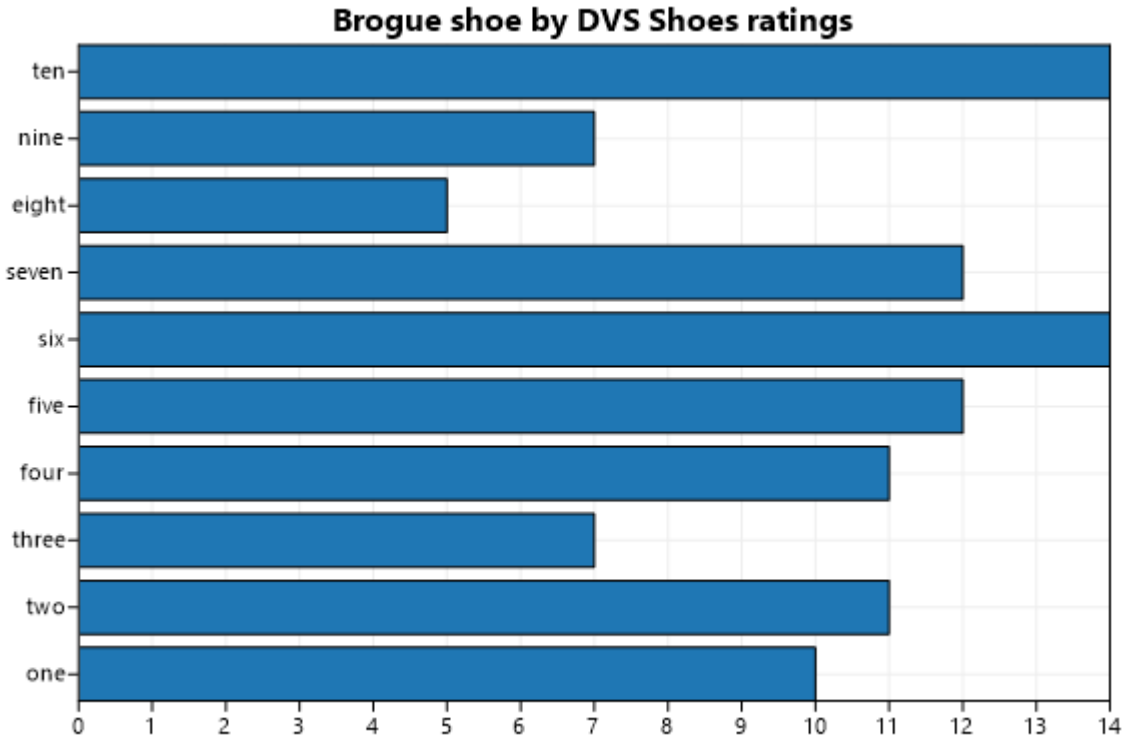
```
C:\Users\Макс\myprojects\db_coursework\data\backup\backup.sql
```

4. Порівняння часу виконання запитів з індексуванням та без:

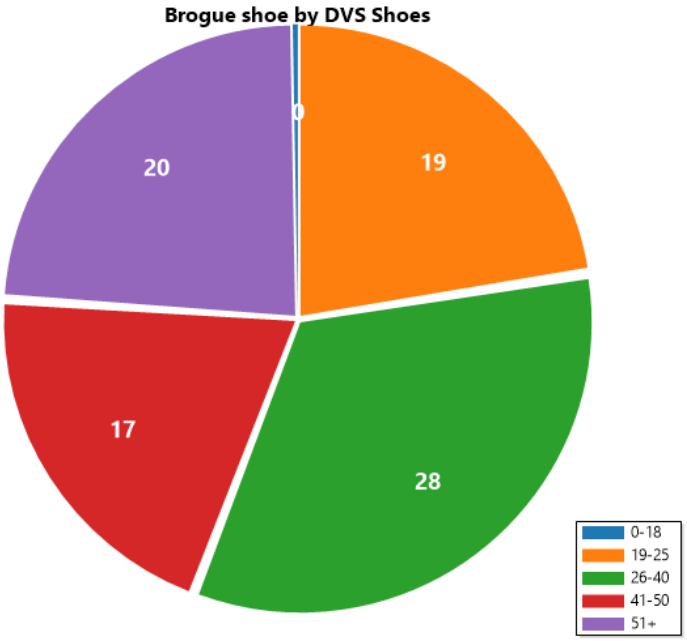




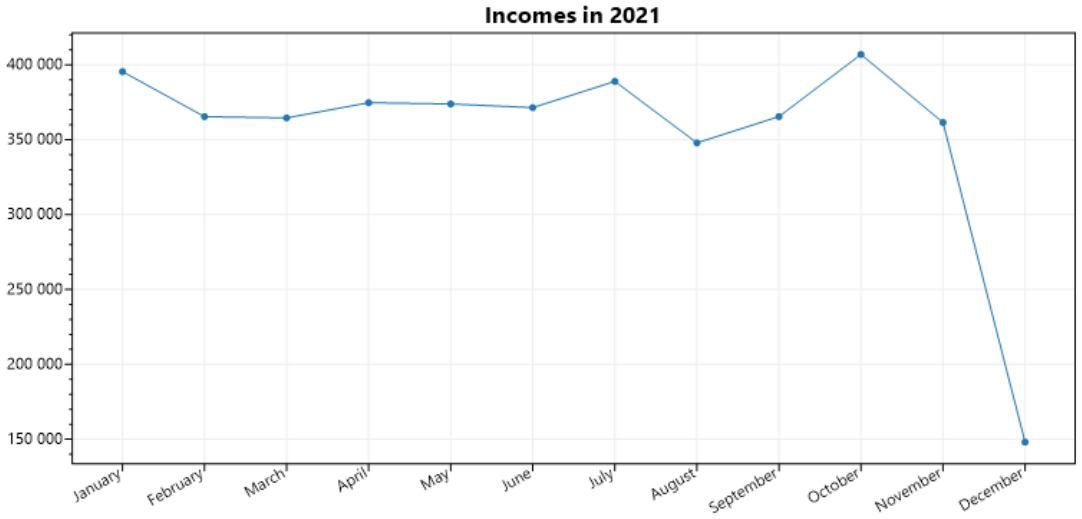
5. Результат аналізу загального рейтингу продукту серед оцінок клієнтів:



6. Результат аналізу вікової категорії товару:



7. Результат аналізу доходів інтернет-магазину:



8. Результат генерації звіту певного товару:

Report

Footwear name: Brogue shoe

Footwear brand: DVS Shoes

Footwear cost: 73$

Footwear country of origin: Korea North

Average footwear rating: 5,51

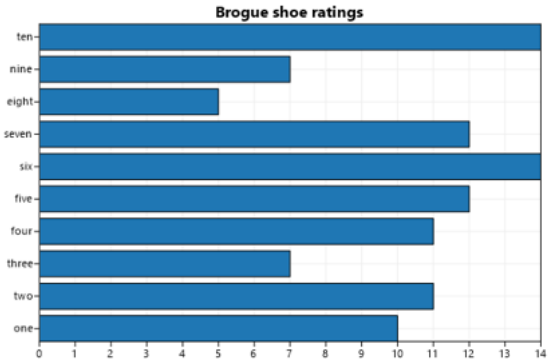
Review with the highest rating:

- Content: jcfwvsfkkkqvkzyc
- Rating: 10
- Created At: 22.09.2021
- By: Fredrick Frierson

Review with the lowest rating:

- Content: fmzpspfnprygajbs
- Rating: 1
- Created At: 29.04.2016
- By: Jason Prow

Graphics of footwear ratings:



Б. Фрагменти програмного коду

Псевдовипадкова генерація сутностей

```
public void GenerateOrders(int amount)
        {
            string query = @$"CREATE TRIGGER order_trigger
                              AFTER INSERT ON orders
                              FOR EACH ROW EXECUTE PROCEDURE
order_items_gen();";
            context.Database.ExecuteSqlRaw(query);

            int[] clientIds = context.clients.Select(o =>
o.id).ToArray();
            Random rand = new Random();

            for(int i = 0; i < amount; i++)
```

```
            {
                query = @$"INSERT INTO orders (client_id,
checkout_date, payment_method)
                           SELECT

{clientIds[rand.Next(0,clientIds.Length)]},
                           timestamp '2016-01-01' + random() *
(timestamp '2021-12-12' - timestamp '2016-01-01'),
                           random_choice(array['cash', 'check',
'debit card', 'credit card', 'mobile payment'])";
                context.Database.ExecuteSqlRaw(query);
            }

            query = "DROP TRIGGER order_trigger ON orders";
            context.Database.ExecuteSqlRaw(query);
        }
```

Генерація на основі датасетів

```
public static void ProcessFootwearGen(DatasetsFilePathes dataFiles,
FootwearRepository repo)
        {
            int amount = GetAmountOfEntities();
            Console.WriteLine("Generating footwear...");
            string[] names =
File.ReadAllText(dataFiles.fwnames).Split("\r\n");
            string[] brands =
File.ReadAllText(dataFiles.brands).Split("\r\n");
            string[] countries =
File.ReadAllText(dataFiles.countries).Split("\r\n");
            Random rand = new Random();
            System.Diagnostics.Stopwatch sw = new
System.Diagnostics.Stopwatch();
            sw.Start();
            List<Footwear> entities = new List<Footwear>();
            for(int i = 0; i < amount; i++)
            {
                entities.Add(new Footwear()
                {
                    name = names[rand.Next(0,names.Length)],
```

```
                    brand = brands[rand.Next(0,brands.Length)],
                    cost = rand.Next(10,700),
                    country_of_origin =
countries[rand.Next(0,countries.Length)]
                });
            }
            repo.AddBulkData(entities);
            Console.WriteLine("[Gen. Footwear] Elapsed:
"+sw.Elapsed);
            sw.Stop();
        }
```

Приклад CRUD

```
 public int Insert(Footwear fw)
        {
            context.footwear.Add(fw);
            context.SaveChanges();
            return fw.id;
        }
        public void DeleteById(int id)
        {
            context.footwear.Remove(context.footwear.Find(id));
            context.SaveChanges();
        }
        public void Update(Footwear fw)
        {
            int id = fw.id;
            var local = context.footwear.Find(id);
            if (local != null)
            {
                context.Entry(local).State = EntityState.Detached;
            }
            context.Entry(fw).State = EntityState.Modified;
            context.SaveChanges();
        }
        public Footwear GetById(int id)
        {
            return context.footwear.Find(id);
        }
```

Аналіз річних доходів

```
public List<double> GetIncomesByYear(int year)
        {
            List<double> result = new List<double>();
            var connection = context.Database.GetDbConnection();
            try
            {
                connection.Open();
                using (var command = connection.CreateCommand())
                {
                    for(int i = 1; i<12; i++)
                    {
                        command.CommandText = @$"SELECT
SUM(footwear.cost) FROM footwear,orders,order_items
                        WHERE orders.checkout_date >= timestamp
'{year}-{i}-1' AND orders.checkout_date < timestamp '{year}-{i+1}-1'
                        AND orders.id = order_items.order_id AND
order_items.product_id = footwear.id";

result.Add(Convert.ToDouble(command.ExecuteScalar()));
                    }
                    command.CommandText = @$"SELECT
SUM(footwear.cost) FROM footwear,orders,order_items
                    WHERE orders.checkout_date >= timestamp
'{year}-12-1' AND orders.checkout_date < timestamp '{year+1}-1-1'
                    AND orders.id = order_items.order_id AND
order_items.product_id = footwear.id";

result.Add(Convert.ToDouble(command.ExecuteScalar()));
                }
            }
            finally { connection.Close(); }

            return result;

        }
```

Створення графіка для аналізу річних доходів

```
public static void CreateIncomesStatisticsChart(List<double> incomes,
int year)
        {
            var plt = new ScottPlot.Plot(800, 400);
            double[] values = incomes.ToArray();
            double[] xs = DataGen.Consecutive(12);
            string[] labels = new string[12]{"January","February",

"March","April","May","June","July","August","September","October","N
ovember","December"};
            plt.AddScatter(xs, values);
            plt.XTicks(xs, labels);
            plt.XAxis.TickLabelStyle(rotation: 30);
            plt.Title($"Incomes in {year}");

plt.SaveFig(@$"C:\Users\Макс\myprojects\db_coursework\data\IncomesPer
{year}.png");
        }
```

Фільтрація даних

```
public List<int> GetRatingsByProductId(int id)
        {return context.reviews.Where(x => x.product_id ==
id).Select(o => o.rating).ToList();}
```

Валідація даних

```
public void UpdateClient(int id, string name, DateTime date, string
email){
            Client c = _service.clientRepository.GetById(id);
            if(c == null)
            {
                View.PrintError($"Cannot update unexisting client");
                return;}
            if(date.Year > DateTime.Now.Year - 16 || date < new
```

```
DateTime(1950,1,1))
            {
                Console.WriteLine("Something is wrong with the client
age");
                return;}
            Client cl = new Client()
            {
                id = id,
                fullname = name,
                birthday_date = date,
                email = email
            };
            try
            {_service.clientRepository.Update(cl);
             View.ShowUpdateResult(typeof(Client).Name, id);}catch
            {
                View.PrintError($"Cannot update client");
            }
        }
```