Лабораторна робота 4: Аналітичні SQL-запити (OLAP)

Контекст

У лабораторних роботах 1–3 ви розробили та заповнили схему бази даних (наприклад, систему реєстрації студентів на курси або інтернет-магазин) і завантажили зразки даних. Тепер, у **лабораторній роботі 4,** ви будете використовувати цю існуючу схему та дані для написання аналітичних SQL-запитів. Ці запити іноді називаються запитами OLAP (онлайн-аналітична обробка), і вони корисні для підсумовування тенденцій та створення звітів. Ви будете використовувати PostgreSQL (у pgAdmin або подібному інструменті) для виконання запитів до створеної вами бази даних.

Цілі

Після завершення цієї лабораторної роботи ви повинні вміти:

- Використати агрегатні функції, такі як COUNT, SUM, AVG, MIN та MAX, для обчислення зведеної статистики з ваших даних.
- Написати запити GROUP BY для групування рядків за одним або кількома стовпцями та обчислення агрегатів для кожної групи.
- Використати HAVING для фільтрації результатів згрупованих запитів на основі агрегованих умов.
- Виконувати операції JOIN (принаймні INNER JOIN та LEFT JOIN), щоб об'єднати дані з кількох таблиць.
- Створювати об'єднані запити на агрегацію для кількох таблиць, які об'єднують таблиці та створюють згрупований, агрегований вивід.
- Інтерпретувати результати ваших запитів та поясніть, що робить кожен з них.

Інструкції

Напишіть мінімум 4 запити, що містять агрегаційні функції (SUM, AVG, COUNT, MIN, MAX, GROUP BY), мінімум 3 запити, що використовують різні типи джоінів (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN), мінімум 3 запити з використанням підзапитів (вибірка з підзапитом у SELECT, WHERE, або HAVING).

Виконайте наступні кроки для завершення лабораторної роботи. Кожен крок описує тип аналітичного запиту, який потрібно написати. Для кожного елемента напишіть SQL-запит у скрипті.`sql` та запустіть його в pgAdmin для вашої бази даних. Перевірте правильність виводу та переконайтеся, що ви розумієте, що робить запит.

Базова агрегація: Напишіть прості запити `SELECT`, використовуючи агрегатні функції для однієї таблиці.

Приклад: У базі даних студентів знайдіть загальну кількість студентів:

```
SELECT COUNT(*) AS total_students
FROM Students;
```

Спробуйте використати схожі запити у вашій схемі. Наприклад, підрахуйте загальну кількість замовлень, знайдіть середню оцінку або підсумуйте загальну кількість продажів. Використовуйте `COUNT`, `SUM`, `AVG`, `MIN` та `MAX` відповідно.

Групування даних: Використовуйте `GROUP BY` для групування рядків за одним або кількома стовпцями та обчислення агрегатів для кожної групи.

Приклад: Знайдіть кількість студентів за кожною спеціальністю:

```
SELECT major, COUNT(*) AS num_students
FROM Students
GROUP BY major;
```

Адаптуйте цей запит до своєї бази даних. Наприклад, групуйте продажі за місяцями, рахуйте студентів за курсами або обчислюйте середню оцінку за кафедрами.

Фільтрування груп: Використовуйте речення `HAVING` для фільтрації агрегованих результатів.

Приклад: Пошук курсів, на яких навчається більше 10 студентів:

```
SELECT course_id, COUNT(*) AS enrolled_count
FROM Enrollments
GROUP BY course_id
HAVING COUNT(*) > 10;
```

Візьміть один із запитів `GROUP BY` з кроку 2 та додайте речення `HAVING`, щоб відфільтрувати групи, які не відповідають умові (наприклад, лише відділи із середнім балом вище 3,0).

Запити JOIN: Об'єднання даних з кількох таблиць за допомогою `JOIN`.

Приклад (INNER JOIN): Перелічіть ім'я кожного студента з назвами курсів, на яких він навчається:

```
SELECT s.student_name, c.course_title
FROM Students s
JOIN Enrollments e ON s.id = e.student_id
JOIN Courses c ON e.course_id = c.id;
```

Напишіть у своїй схемі як запит INNER JOIN, так і запит LEFT JOIN. Наприклад, об'єднуйте таблиці для виведення об'єднаної інформації та використовуйте LEFT JOIN, коли потрібно включити рядки без відповідних записів.

Багатотаблична агрегація: написання запитів, які об'єднують кілька таблиць та агрегують результати.

Приклад: Знайдіть загальну суму, витрачену кожним клієнтом (об'єднання Клієнтів, Замовлень та Елементів Замовлення):

```
SELECT c.customer_name, SUM(oi.quantity * oi.unit_price) AS total_spent
FROM Customers c
JOIN Orders o ON c.id = o.customer_id
JOIN OrderItems oi ON o.id = oi.order_id
GROUP BY c.customer_name;
```

Адаптуйте ці ідеї до своєї бази даних. Використовуйте стільки об'єднань, скільки потрібно, та групуйте за відповідним стовпцем, щоб обчислити потрібний агрегований результат.

Якщо вам потрібні нагадування щодо синтаксису, скористайтеся офіційною документацією PostgreSQL або довідкою pgAdmin. Обов’язково перевірте кожен запит і зрозумійте, чому він видає саме такі результати.

Результати

- **Файли скриптів SQL (`.sql`)** : що містять усі запити, які ви написали для цієї лабораторної роботи (ви можете використовувати один файл на запит або групувати пов'язані запити у файли).
- **Файл Markdown (`.md`)** : короткий опис кожного запиту (що він робить і чому).
- **Надсилання** : Збережіть та надішліть файли `.sql` та `.md` до **Git-репозиторію вашої групи**.

Критерії оцінювання

Ваша лабораторна робота буде оцінена за такими критеріями:
- **Коректність виконання запитів:** усі запити повертають очіквані результати.
- **Синтаксис SQL:** оператори `SELECT` виконуються без помилок у PostgreSQL при використанні з операторами `WHERE`, `JOIN`, `GROUP BY`, `ORDER BY`.
- **Узгодженість даних:** вибіркові дані узгоджуються зі схемою (наприклад, значення зовнішнього ключа відповідають існуючим первинним ключам, немає null-значень, де це заборонено, значення задовольняють умови `CHECK`).
- **Документація:** пояснення схеми є чітким та точним. (Таблиці та ключі повинні бути чітко описані.)

Додаткові ресурси:

Лекційний матеріал до лабораторної

https://github.com/ZheniaTrochun/db-intro-course/tree/master/lectures/5%20-%20JOINs%20and%20set%20operations

https://github.com/ZheniaTrochun/db-intro-course/tree/master/lectures/6%20-%20GROUP%20BY%20and%20window%20functions

PostgreSQL Documentation

https://www.postgresql.org/docs/current/sql.html

Mode SQL Tutorial (Advanced SQL & Analytics)

https://mode.com/sql-tutorial/

PostgreSQL Tutorial (Comprehensive Guides & Examples)

https://www.postgresqltutorial.com/