***Лабораторна робота 6: Агрегатні функції, представлення***

Група: АІ-243

Здобувач: Гаврилов О.В.

Варіант: 22

**Мета:**

Навчити студентів виконувати запити до таблиць бази даних з використанням

агрегатних функцій, виконувати сортування та групування даних, а також створювати

представлення.

**Завдання:**

1. Вибірка з використанням агрегатних функцій:

Виконати запити на вибірку даних, які повертають єдине значення для підмножини

кортежів.

2. Вибірка з використанням групових операцій:

Виконати запити на вибірку даних, які повертають результат обробки в рамках

певних підгруп.

3. Вибірка з використанням операції сортування:

Виконати запити на вибірку даних, які повертають кортежі результату вибірки у

впорядкованому вигляді.

4. Вибірка з результатів роботи групових операцій:

Виконати запити на вибірку даних, які повертають результат обробки тільки в

заданих підгрупах.

5. Створення представлень:

Виконати запити на створення представлення (view) згідно завдання та предметної

області.

**Результат:**

Студенти повинні подати SQL-скрипти, що відображають запити на вибірку з

таблиць бази даних з використанням агрегатних функцій, операцій сортування та

групування, запити на створення представлень згідно завдання та предметної області, їх

опис, а також звіт з результатами тестування.

***1) Виконати запити на вибірку даних, які повертають***

***1.1) максимальне значення атрибуту типу дата:***

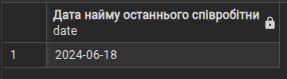
***Словесна постановка задачі:***

*Знайти* ***найпізнішу дату*** *(максимальне значення) найму співробітника (hired\_date) у ресторані, щоб визначити, хто був найнятий останнім.*

SQL-код рішення:

|  |
| --- |
| **SELECT**  MAX(hired\_date) **AS** "Дата найму останнього співробітника"  **FROM**  Співробітник; |

**Скриншот отриманого результату:**



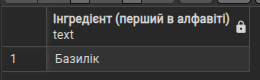
***1.2) Мінімальне значення атрибуту символьного типу:***

***Словесна постановка задачі:***

*Визначити* ***перше в алфавітному порядку*** *ім'я (мінімальне значення) інгредієнта (ingredient\_name), що є в меню.*

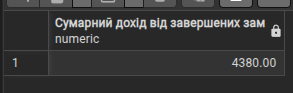
SQL-код рішення:

|  |
| --- |
| **SELECT**  MIN(ingredient\_name) **AS** "Інгредієнт (перший в алфавіті)"  **FROM**  Інгредієнт; |

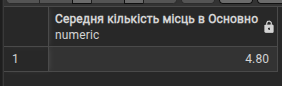
**Скриншот отриманого результату:**

1. ***1.3) Мінімальне значення атрибуту символьного типу:***
2. ***Словесна постановка задачі:***
3. *Розрахувати* ***загальну суму*** *всіх замовлень (total\_amount), які були* ***успішно завершені*** *(status = 'Завершено').*
4. SQL-код рішення:

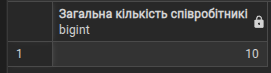
|  |
| --- |
| 1. **SELECT**   SUM(total\_amount) **AS** "Сумарний дохід від завершених замовлень"  **FROM**  Замовлення  **WHERE**  **status** = 'Завершено'; |

1. **Скриншот отриманого результату:**
2. 
3. ***1.4) Середнє значення атрибуту цілого типу:***
4. ***Словесна постановка задачі:***
5. *Розрахувати* ***середню кількість місць*** *(seats) за всіма столиками, що знаходяться в* ***Основному залі****.*
6. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   ROUND(AVG(T.seats), 2) **AS** "Середня кількість місць в Основному залі"  **FROM**  Столик **AS** T  **JOIN**  Зал **AS** H **ON** T.hall\_id = H.hall\_id  **WHERE**  H.hall\_name = 'Основний'; |

1. **Скриншот отриманого результату:**
2. 
3. ***1.5) Кількість рядків будь-якої сутності:***
4. ***Словесна постановка задачі:***
5. *Підрахувати* ***загальну кількість*** *усіх записів (рядків) у таблиці* ***Співробітник****, щоб знати загальний штат персоналу.*
6. SQL-код рішення:

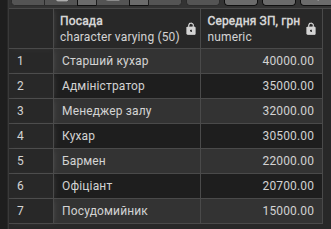
|  |
| --- |
| 1. **SELECT**   COUNT(\*) **AS** "Загальна кількість співробітників"  **FROM**  Співробітник; |

1. **Скриншот отриманого результату:**

## 2. Виконати запити на вибірку даних, які повертають:

1. ***2.1) середнє значення атрибуту цілого типу для кожної групи за значенням атрибуту***
2. ***символьного типу:***
3. ***Словесна постановка задачі:***
4. *Для кожної посади (position\_name — символьний тип) розрахувати* ***середній розмір заробітної плати*** *(salary — цілочисельний/десятковий тип, що є його аналогом для агрегації) співробітників, які обіймають цю посаду.*
5. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   P.position\_name **AS** "Посада",  ROUND(AVG(S.salary), 2) **AS** "Середня ЗП, грн"  **FROM**  Співробітник **AS** S  **JOIN**  Посада **AS** P **ON** S.position\_id = P.position\_id  **GROUP** **BY**  P.position\_name  **ORDER** **BY**  "Середня ЗП, грн" **DESC**; |

1. **Скриншот отриманого результату:**
2. ***2.2) кількість рядків в кожній групі (на вибір студента):***
3. ***Словесна постановка задачі:***
4. *Визначити* ***кількість столиків*** *(COUNT(\*)) для* ***кожного залу*** *(hall\_name — символьний тип), щоб оцінити завантаженість або розмір кожного залу.*
5. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   H.hall\_name **AS** "Назва Залу",  COUNT(T.table\_id) **AS** "Кількість столиків"  **FROM**  Зал **AS** H  **JOIN**  Столик **AS** T **ON** H.hall\_id = T.hall\_id  **GROUP** **BY**  H.hall\_name  **ORDER** **BY**  "Кількість столиків" **DESC**; |

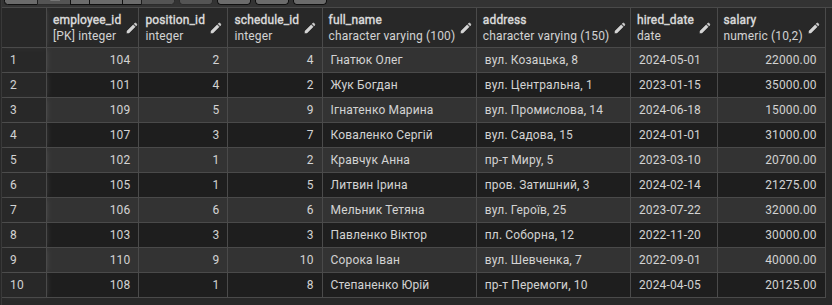
1. **Скриншот отриманого результату:**

## 3. Виконати запити на вибірку даних, які повертають:

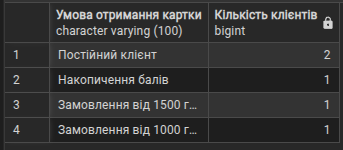
***3.1) всі дані таблиці, впорядковані за зростанням за будь-яким символьним***

1. ***атрибутом:***
2. ***Словесна постановка задачі:***
3. *Отримати* ***всі дані*** *про співробітників, відсортувавши список за* ***повним ім'ям*** *(full\_name) у* ***прямому (за зростанням) алфавітному порядку****.*
4. SQL-код рішення:

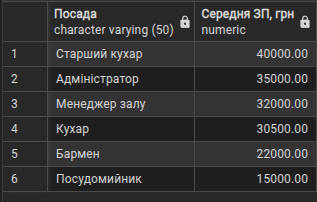
|  |
| --- |
| 1. **SELECT** 2. \* 3. **FROM** 4. Співробітник 5. **ORDER** **BY** 6. full\_name **ASC**; |

1. **Скриншот отриманого результату:**
2. ***3.2) згруповані та відсортовані в зворотньому порядку за значенням символьного***
3. ***поля кортежі таблиці, в яких значення цілого атрибуту більше заданого значення (тобто***
4. ***відібрати кортежі за умовою, згрупувати за символьним полем та відсортувати за цим же***
5. ***символьним полем):***
6. ***Словесна постановка задачі:***
7. *Показати кількість клієнтів, які мають на картці лояльності* ***більше 100 балів*** *(current\_points > 100). Результат* ***згрупувати*** *за* ***умовою отримання картки*** *(obtaining\_condition) та* ***відсортувати*** *ці умови в* ***зворотному алфавітному порядку****.*
8. SQL-код рішення:

|  |
| --- |
| 1. **SELECT** 2. obtaining\_condition **AS** "Умова отримання картки", 3. COUNT(client\_id) **AS** "Кількість клієнтів" 4. **FROM** 5. Картка\_лояльності 6. **WHERE** 7. loyalty\_points > 100 -- Використовуємо перейменоване поле loyalty\_points з попередньої ЛР 8. **GROUP** **BY** 9. obtaining\_condition 10. **ORDER** **BY** 11. obtaining\_condition **DESC**; |

1. (Примітка: У цьому коді використовується ім'я стовпця loyalty\_points, яке було встановлено в завданні 3.3 попередньої частини. Якщо Ви не виконали ALTER TABLE, замініть його на current\_points.)
2. **Скриншот отриманого результату:**
3. 
4. ***4. Виконати запити на вибірку даних, які повертають:***
5. ***4.1) середнє значення атрибуту цілого типу для підгруп за значенням атрибуту***
6. ***символьного типу, які недорівнюють заданому значенню (тобто виділити підгрупи за***
7. ***атрибутом символьного типу, взяти середнє значення атрибуту цілого типу в кожній***
8. ***підгрупі, вивести тільки ті підгрупи, які недорівнюють заданому значенню).***
9. ***Словесна постановка задачі:***
10. *Розрахувати* ***середню заробітну плату*** *(salary) для* ***кожної посади*** *(position\_name), за винятком посад, назва яких* ***не дорівнює 'Офіціант'****. Тобто, ми виключаємо з фінального звіту групу, пов'язану з офіціантами.*
11. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   P.position\_name **AS** "Посада",  ROUND(AVG(S.salary), 2) **AS** "Середня ЗП, грн"  **FROM**  Співробітник **AS** S  **JOIN**  Посада **AS** P **ON** S.position\_id = P.position\_id  **GROUP** **BY**  P.position\_name  **HAVING**  P.position\_name <> 'Офіціант' -- Виключаємо групу 'Офіціант'  **ORDER** **BY**  "Середня ЗП, грн" **DESC**; |

2. **Скриншот отриманого результату:**
3. **5. Виконати запити на:**

**5.1. Створення представлення для завдання 3.2:**

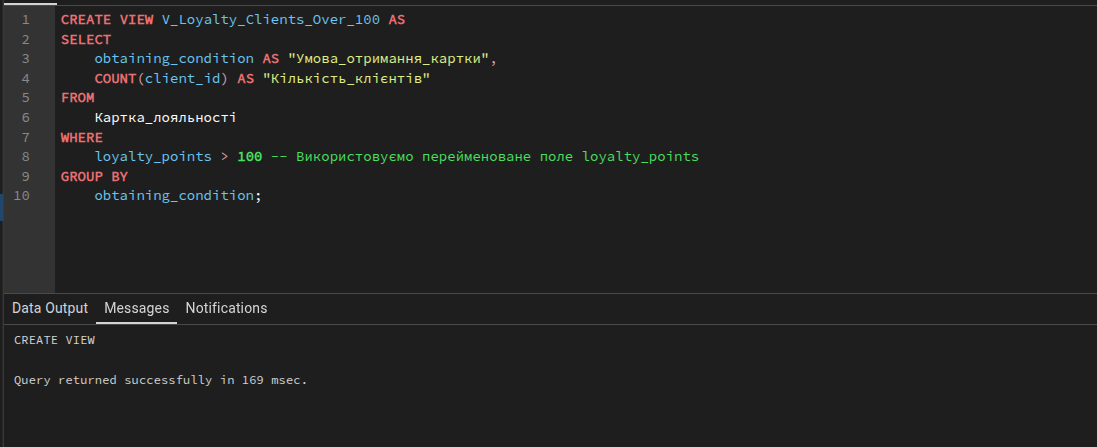
***Словесна постановка задачі:***

*Створити віртуальну таблицю (представлення) на основі результату запиту з пункту 3.2, який підраховує кількість клієнтів за умовою отримання картки лояльності, якщо клієнт має більше 100 балів.*

1. **SQL-код рішення:**

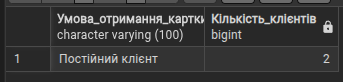
|  |
| --- |
| 1. **CREATE** VIEW V\_Loyalty\_Clients\_Over\_100 **AS**   **SELECT**  obtaining\_condition **AS** "Умова\_отримання\_картки",  COUNT(client\_id) **AS** "Кількість\_клієнтів"  **FROM**  Картка\_лояльності  **WHERE**  loyalty\_points > 100 -- Використовуємо перейменоване поле loyalty\_points  **GROUP** **BY**  obtaining\_condition; |

1. **Скриншот отриманого результату:**



1. **5.2. виконати запит до представлення створеного в пункті 5.1, який повертає**
2. **значення тільки однієї підгрупи:**
3. ***Словесна постановка задачі:***
4. *Виконати запит до щойно створеного представлення V\_Loyalty\_Clients\_Over\_100 та відфільтрувати результат, щоб показати кількість клієнтів лише для однієї підгрупи — тих, хто отримав картку за умовою* ***'Постійний клієнт'****.*
5. **SQL-код рішення:**

|  |
| --- |
| 1. **SELECT**   \*  **FROM**  V\_Loyalty\_Clients\_Over\_100  **WHERE**  "Умова\_отримання\_картки" = 'Постійний клієнт'; |

1. **Скриншот отриманого результату:**
2. 

## Висновок:

Протягом виконання лабораторної роботи було успішно застосовано та закріплено навички використання **агрегатних функцій** (MAX, MIN, SUM, AVG, COUNT), оператора **GROUP BY** для аналізу даних за групами, а також створення та використання **представлень (VIEW)** як інструменту для спрощення складних запитів.

### Основні результати та засвоєні концепції:

1. **Застосування Агрегатних Функцій (Розділ 1):** Була продемонстрована здатність отримувати узагальнену інформацію з бази даних (наприклад, загальний дохід від завершених замовлень, найпізніша дата найму) без необхідності обробки кожного окремого запису.
2. **Аналіз даних за групами (Розділ 2 та 4):**
   * Успішно використано **GROUP BY** для розрахунку середньої заробітної плати для кожної посади та кількості столиків для кожного залу.
   * За допомогою **HAVING** продемонстровано вміння фільтрувати результати після групування (наприклад, виключаючи звіт по офіціантах), що є критичним для формування цільових аналітичних звітів.
3. **Сортування та Фільтрація (Розділ 3):** Підтверджено навички впорядкування вибірок за різними атрибутами (ORDER BY ASC/DESC) та комбінування фільтрації (WHERE) з подальшим групуванням та сортуванням.
4. **Створення та використання Представлень (Розділ 5):**
   * Створено віртуальну таблицю (V\_Loyalty\_Clients\_Over\_100), яка інкапсулює складну логіку групування та фільтрації.
   * Продемонстровано, що **представлення можна використовувати так само, як і звичайну таблицю**, застосовуючи до нього додаткову фільтрацію (SELECT ... WHERE...), що значно спрощує подальші запити та підвищує модульність коду.

### Загальний підсумок:

Лабораторна робота підтвердила глибоке розуміння того, як **SQL переходить від простої вибірки до складного аналізу даних**. Засвоєння цих інструментів є ключовим для формування бізнес-звітів, аналізу продуктивності та ефективного управління даними в реальних корпоративних системах.