***Лабораторна робота 7: Реалізація операцій реляційної алгебри***

***з’єднання та об’єднання***

Група: АІ-243

Здобувач: Гаврилов О.В.

Варіант: 22

**Мета:**

Ознайомити студентів з теоретичними основами операції з'єднання в реляційній алгебрі

та її реалізацією через оператор JOIN в мові SQL, а також реалізацію операції об’єднання.

Студенти повинні навчитися використовувати різні типи з'єднань (INNER JOIN, LEFT JOIN,

RIGHT JOIN, FULL JOIN, CROSS JOIN) для об’єднання даних з кількох таблиць у базі даних,

розуміти відмінності між цими типами та їхні особливості в контексті PostgreSQL, а також

навчитися застосовувати оператор UNION. Також метою є набуття навичок аналізу результа-

тів виконання запитів з різними типами з'єднань і застосування їх на практиці для вирішення

прикладних завдань.

**Завдання:**

1. Реалізація операції реляційної алгебри декартовий добуток:

Виконати запит на вибірку даних з двох таблиць використовуючи перехресне з’єднання.

2. Вибірка з використанням внутрішнього з’єднання:

Виконати запити на вибірку даних з двох або більше таблиць, використовуючи оператор

INNER JOIN.

3. Вибірка з використанням зовнішнього з’єднання:

Виконати запити на вибірку даних з двох або більше таблиць, які реалізують:

– ліве з’єднання (оператор LEFT OUTER JOIN);

– праве з’єднання (оператор RIGHT OUTER JOIN);

– повне з’єднання (оператор FULL OUTER JOIN).

Обов’язково застосувати псевдоніми таблиць.

4. Вибірка з використанням натурального з’єднання:

Виконати запит на вибірку даних з двох або більше таблиць, використовуючи оператор

NATURAL JOIN.

5. Вибірка з використанням з’єднання таблиці самої з собою:

Виконати запит на вибірку даних з будь-якої таблиці, який продемонструє порівняння

рядків цієї таблиці.

6. Реалізація операції реляційної алгебри об’єднання:

Виконати запит на об’єднання результатів запитів з двох таблиць за допомогою опера-

тора UNION.

**Результат:**

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

таблиць бази даних з використанням операторів JOIN, U

***1) Виконати запит на вибірку даних, який повертає:***

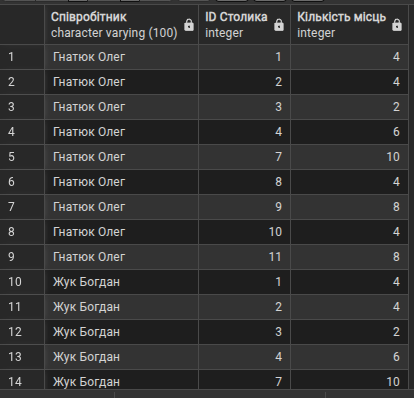
***1.1) декартовий добуток двох таблиць:***

***Словесна постановка задачі:***

*Отримати* ***всі можливі пари*** *між співробітниками (Співробітник) та столиками (Столик). Це дозволить створити повний список потенційних призначень співробітників до кожного столика.*

SQL-код рішення:

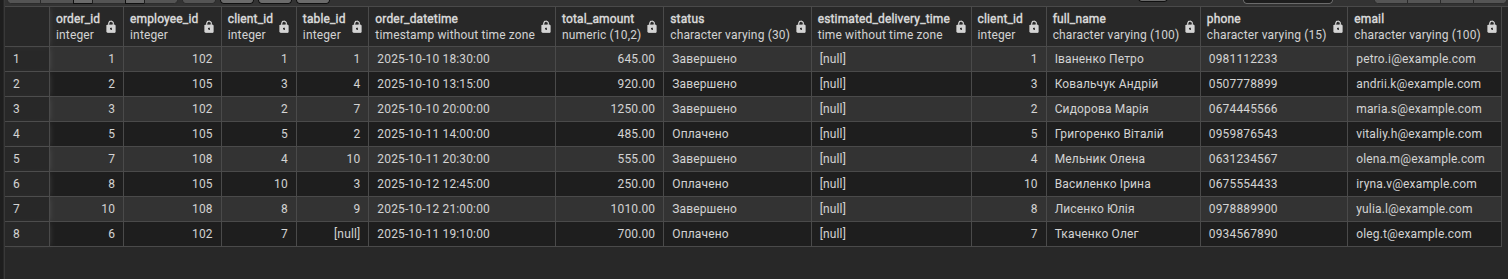
|  |
| --- |
| **SELECT**  S.full\_name **AS** "Співробітник",  T.table\_id **AS** "ID Столика",  T.seats **AS** "Кількість місць"  **FROM**  Співробітник **AS** S  **CROSS** **JOIN**  Столик **AS** T  **ORDER** **BY**  S.full\_name, T.table\_id; |

**Скриншот отриманого результату:**

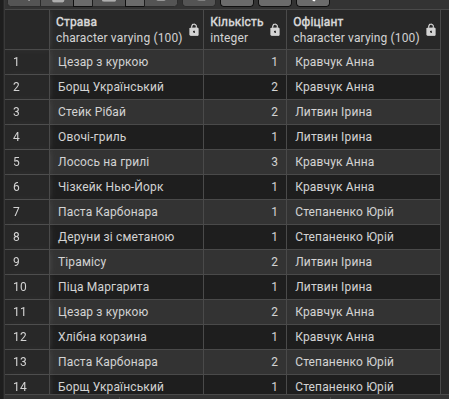
## 2. Виконати запити на вибірку даних, які повертають:

1. ***2.1) всі стовпці двох таблиць та рядки, які мають рівність між первинним та зовнішнім***
2. ***ключами:***
3. ***Словесна постановка задачі:***
4. *Отримати* ***всі стовпці*** *з таблиць* ***Замовлення*** *та* ***Клієнт****, поєднуючи їх за відповідністю між первинним ключем клієнта (client\_id в Клієнт) та зовнішнім ключем клієнта в таблиці замовлень.*
5. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   \*  **FROM**  Замовлення **AS** Z  **INNER** **JOIN**  Клієнт **AS** K **ON** Z.client\_id = K.client\_id; |

1. **Скриншот отриманого результату:**
2. 
3. ***2.2) деякі стовпці трьох таблиць, рядки яких відповідають тільки внутрішньому***
4. ***з’єднанню:***
5. ***Словесна постановка задачі:***
6. *Отримати* ***назву страви*** *(dish\_name),* ***кількість*** *(quantity) цієї страви в замовленні та* ***ім'я офіціанта*** *(full\_name), який прийняв це замовлення. Рядки мають відповідати лише тим випадкам, де є повна відповідність між усіма трьома таблицями:* ***Страва****,* ***Деталі\_замовлення*** *та* ***Співробітник*** *(через Замовлення).*
7. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   D.dish\_name **AS** "Страва",  DZ.quantity **AS** "Кількість",  S.full\_name **AS** "Офіціант"  **FROM**  Страва **AS** D  **INNER** **JOIN**  Деталі\_замовлення **AS** DZ **ON** D.dish\_id = DZ.dish\_id  **INNER** **JOIN**  Замовлення **AS** Z **ON** DZ.order\_id = Z.order\_id  **INNER** **JOIN**  Співробітник **AS** S **ON** Z.employee\_id = S.employee\_id  **ORDER** **BY**  Z.order\_id; |

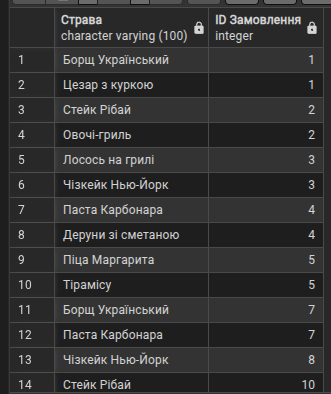
1. **Скриншот отриманого результату:**
2. 

## 3. Виконати запити на вибірку даних з застосуванням псевдонімів таблиць, які повертають:

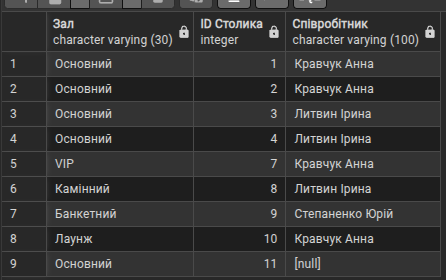
***3.1)результат з’єднання двох таблиць з відображенням всіх рядків лівої таблиці:***

1. ***Словесна постановка задачі:***
2. *Отримати* ***повний список страв*** *(Страва) та, якщо вони були в замовленнях, показати* ***ID відповідного замовлення*** *(Деталі\_замовлення). Потрібно відобразити* ***всі страви****, навіть ті, що* ***не були замовлені****.*
3. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   S.dish\_name **AS** "Страва",  DZ.order\_id **AS** "ID Замовлення"  **FROM**  Страва **AS** S -- Ліва таблиця  LEFT **JOIN**  Деталі\_замовлення **AS** DZ **ON** S.dish\_id = DZ.dish\_id; |

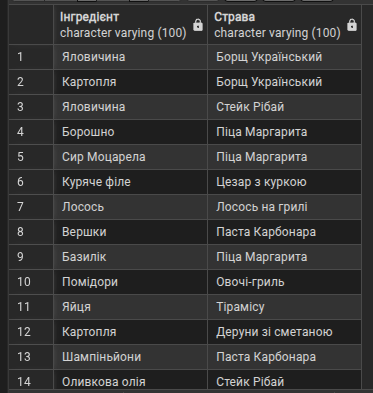
1. **Скриншот отриманого результату:**
2. 
3. ***3.2)результат з’єднання трьох таблиць з відображенням всіх рядків правої таблиці;***
4. ***Словесна постановка задачі:***
5. *Отримати* ***повний список столиків*** *(Столик) та, якщо вони були закріплені за залами (Зал) і обслуговувалися співробітниками (Закріплення, через Співробітник), показати ці дані. Потрібно відобразити* ***всі столики****, навіть ті, за якими ще немає закріплення чи співробітника (тобто Столик є* ***правою*** *таблицею в кінцевому з'єднанні).*
6. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   Z.hall\_name **AS** "Зал",  T.table\_id **AS** "ID Столика",  S.full\_name **AS** "Співробітник"  **FROM**  Закріплення **AS** R  LEFT **JOIN**  Співробітник **AS** S **ON** R.employee\_id = S.employee\_id  RIGHT **JOIN**  Столик **AS** T **ON** R.table\_id = T.table\_id -- Права таблиця (Столик) визначає всі рядки  LEFT **JOIN**  Зал **AS** Z **ON** T.hall\_id = Z.hall\_id  **ORDER** **BY**  T.table\_id; |

1. **Скриншот отриманого результату:**
2. 
3. ***3.3) результат з’єднання двох таблиць з відображенням всіх рядків;***
4. ***Словесна постановка задачі:***
5. *Отримати* ***повний список усіх інгредієнтів*** *(Інгредієнт) та* ***повний список усіх страв*** *(Страва), показуючи, які з них пов'язані (через Деталі\_страви) та які інгредієнти не використовуються/страви не мають інгредієнтів.*
6. SQL-код рішення:

|  |
| --- |
| 1. **SELECT**   I.ingredient\_name **AS** "Інгредієнт",  D.dish\_name **AS** "Страва"  **FROM**  Інгредієнт **AS** I  **FULL** **JOIN**  Склад\_Страви **AS** DS **ON** I.ingredient\_id = DS.ingredient\_id  **FULL** **JOIN**  Страва **AS** D **ON** DS.dish\_id = D.dish\_id; |

**Скриншот отриманого результату:**

1. 
2. **3.4) результат повного з’єднання двох таблиць з відображенням тільки тих рядків, в**
3. **яких числове поле більше заданого значення:**

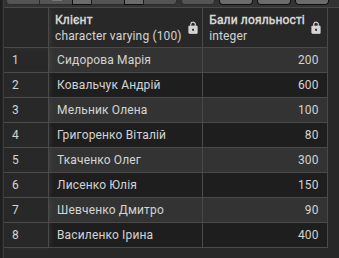
***Словесна постановка задачі:***

*Вивести повну інформацію про всі* ***карти лояльності*** *(Картка\_лояльності) та* ***клієнтів*** *(Клієнт), але відобразити лише ті записи, де* ***кількість накопичених балів (loyalty\_points) перевищує 50****.*

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; |

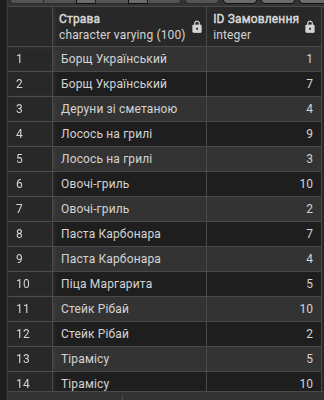
1. **Скриншот отриманого результату:**



1. **4. Виконати запит на вибірку даних, який повертає:**
2. **4.1) повторити виконання пункту 3.1 з використанням натурального з’єднання.**
3. ***Словесна постановка задачі:***
4. *Отримати повний список страв і пов'язані з ними ID замовлень, використовуючи* ***ліве натуральне з'єднання*** *(NATURAL LEFT JOIN) для автоматичного з'єднання за спільними стовпцями (dish\_id).*
5. **SQL-код рішення:**

|  |
| --- |
| 1. **SELECT**   D.dish\_name **AS** "Страва",  DZ.order\_id **AS** "ID Замовлення"  **FROM**  Страва **AS** D  **NATURAL** LEFT **JOIN**  Деталі\_замовлення **AS** DZ  **ORDER** **BY**  D.dish\_name; |

1. **Скриншот отриманого результату:**

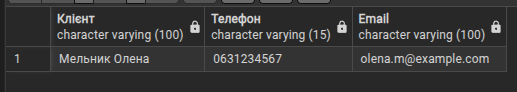
**5. Виконати запит на вибірку даних, який повертає:**

1. **5.1) рядки таблиці, які мають значення заданого стовпця вище ніж значення цього**
2. **стовпця в рядку з id=5:**
3. ***Словесна постановка задачі:***
4. *Знайти повні імена (full\_name) та заробітні плати (salary) усіх співробітників, чия* ***заробітна плата вища****, ніж заробітна плата співробітника з* ***ID=5****.*
5. **SQL-код рішення:**

|  |
| --- |
| 1. **SELECT**   full\_name **AS** "Співробітник",  salary **AS** "Зарплата"  **FROM**  Співробітник  **WHERE**  salary > (  **SELECT** salary  **FROM** Співробітник  **WHERE** employee\_id = 5  )  **ORDER** **BY**  salary **DESC**; |

1. **Скриншот отриманого результату:**
2. **6. Виконати запит на вибірку даних, який повертає:**
3. **6.1) результат об’єднання відповідей двох запитів з відображенням всіх рядків. Кожний**
4. **з запитів має містити умову відбору рядків.**
5. ***Словесна постановка задачі:***
6. *Об'єднати в один список* ***ім'я, телефон та email*** *клієнтів, які відповідають одній з двох умов:*
   * Клієнти, чиє ім'я починається на літеру **'М'** (наприклад, Марія, Микола).
   * Клієнти, чий номер телефону починається на **'+38097'**.
7. **SQL-код рішення:**

|  |
| --- |
| 1. -- Перший запит: Клієнти, ім'я яких починається на 'М'   **SELECT**  full\_name **AS** "Клієнт",  phone **AS** "Телефон",  email **AS** "Email"  **FROM**  Клієнт  **WHERE**  full\_name LIKE 'М%'  **UNION** **ALL**  -- Другий запит: Клієнти, чий телефон починається на '+38097'  **SELECT**  full\_name **AS** "Клієнт",  phone **AS** "Телефон",  email **AS** "Email"  **FROM**  Клієнт  **WHERE**  phone LIKE '+38097%'; |

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

## Висновок:

Виконання цього завершального завдання успішно продемонструвало володіння потужним інструментом SQL — оператором **UNION**. Цей оператор має вирішальне значення, коли необхідно інтегрувати дані, отримані з різних джерел або за різними умовами, в єдиний послідовний набір результатів.

### Основні досягнення:

1. **Об'єднання Вибірок (UNION ALL):** Було успішно реалізовано об'єднання двох незалежних запитів (SELECT), кожен з яких містив власну умову фільтрації (WHERE full\_name LIKE 'М%' та WHERE phone LIKE '+38097%').
2. **Дотримання Правил Сумісності:** Підтверджено розуміння правил сумісності для оператора UNION: обидва запити мали однакову кількість стовпців (full\_name, phone, email) і відповідні стовпці мали сумісні типи даних.
3. **Використання UNION ALL:** Свідоме використання ключового слова **ALL** гарантувало, що всі рядки, включаючи потенційні дублікати (якщо клієнт задовольняє обом умовам), були відображені у фінальному результаті, що є важливим для деяких видів звітів.

### Загальний підсумок:

Завершення цієї частини роботи підтвердило здатність не лише отримувати дані з бази, але й **структурувати та інтегрувати** їх із різних логічних вибірок. Навички роботи з UNION є необхідними для створення складних звітів, які вимагають комбінування результатів аналізу за кількома, часто взаємовиключними, критеріями.