Київський національний університет
Імені Тараса Шевченка.
Кафедра: Мережевих та інтернет-технологій.

Лабораторна робота №3

з дисципліни: Бази даних та інформаційні системи
На тему:«База даних для управління студентськими оцінками”

Студента 3 курсу:
Групи МІТ-31
Мулико Володимира

Київ - 2024р.

Мета роботи

Закріплення теоретичних знань та набуття практичних навичок у складанні складних SQL-запитів до реляційної бази даних. Робота передбачає використання операторів SELECT, WHERE, логічних операторів, агрегатних функцій, JOIN, підзапитів, CTE, віконних функцій тощо.

Завдання

1. **Розробити 40 SQL-запитів** до створеної у попередній роботі бази даних, що охоплюють наступні аспекти:

    - Логічні оператори
        - COUNT(), SUM(), AVG(), MIN(), MAX()
        - Усі типи JOIN (INNER, LEFT, RIGHT, FULL, CROSS, SELF)
    - **Складні запити:**
        - Підзапити (Subqueries) у WHERE, IN, NOT EXISTS, EXISTS
        - Операції над множинами (UNION, INTERSECT, EXCEPT)
        - Common Table Expressions (CTE)
        - Віконні функції (Window Functions)

Код SQL:

-- Додавання тестових складів

INSERT INTO warehouses (id, location, capacity) VALUES

(1, 'Київ, склад №1', 1000),

(2, 'Львів, склад №2', 500),

(3, 'Одеса, склад №3', 800),

(4, 'Харків, склад №4', 1200);

-- Додавання тестових товарів

INSERT INTO products (id, name, description, price, stock_quanity, warehouse_id) VALUES

(1, 'Ноутбук', 'Потужний ноутбук для роботи', 1500.00, 10, 1),

(2, 'Смартфон', 'Новий смартфон з великим екраном', 800.00, 15, 2),

(3, 'Навушники', 'Бездротові навушники', 150.00, 20, 3),

(4, 'Рюкзак', 'Зручний рюкзак для ноутбука', 50.00, 25, 4);

-- Додавання тестових клієнтів

INSERT INTO customers (id, name, contact_info, addres, created_at) VALUES

(1, 'Іван Петренко', 'ivan@example.com', 'Київ, вул. Шевченка, 10', NOW()),

(2, 'Марія Іванова', 'maria@example.com', 'Львів, вул. Франка, 15', NOW()),

(3, 'Олег Сидоренко', 'oleg@example.com', 'Одеса, вул. Дерибасівська, 5', NOW()),

(4, 'Анна Коваленко', 'anna@example.com', 'Харків, вул. Сумська, 20', NOW());

-- Додавання тестових замовлень

INSERT INTO orders (id, customer_id, total_amount, status, created_at) VALUES

(1, 1, 250.50, 'Оплачено', NOW()),

(2, 2, 120.00, 'Очікує оплату', NOW()),

(3, 3, 310.75, 'Доставлено', NOW()),

(4, 4, 95.30, 'В обробці', NOW());

-- Додавання тестових кур'єрів

INSERT INTO couriers (id, name, contact_info, vehicle) VALUES

(1, 'Павло Андрійович', '+380971234567', 'Авто'),

(2, 'Ірина Олександрівна', '+380981234568', 'Скутер'),

(3, 'Олексій Сергійович', '+380991234569', 'Велосипед'),

(4, 'Світлана Миколаївна', '+380951234570', 'Авто');

-- Додавання тестових платежів

INSERT INTO payments (id, order_id, payment_method, status, created_at) VALUES

(1, 1, 'Картка', 'Підтверджено', NOW()),

(2, 2, 'Готівка', 'Очікує оплату', NOW()),

(3, 3, 'Картка', 'Підтверджено', NOW()),

(4, 4, 'PayPal', 'Очікує оплату', NOW());

-- 1. Використання логічних операторів

SELECT * FROM products WHERE price > 100 AND stock_quanity < 20;

-- 2. Використання OR

SELECT * FROM customers WHERE name LIKE 'Іван%' OR contact_info LIKE '%example.com';

-- 3. Використання NOT

SELECT * FROM orders WHERE status NOT IN ('Оплачено', 'Доставлено');

-- 4. COUNT() - підрахунок кількості товарів

SELECT COUNT(*) FROM products;

-- 5. SUM() - загальна вартість товарів

SELECT SUM(price * stock_quanity) AS total_inventory_value FROM products;

-- 6. AVG() - середня вартість товарів

SELECT AVG(price) FROM products;

-- 7. MIN() - мінімальна ціна товару

SELECT MIN(price) FROM products;

-- 8. MAX() - максимальна ціна товару

SELECT MAX(price) FROM products;

```
-- 9. INNER JOIN (замовлення і клієнти)
SELECT orders.id, customers.name, orders.total_amount FROM orders
INNER JOIN customers ON orders.customer_id = customers.id;

-- 10. LEFT JOIN (усі замовлення, навіть якщо немає клієнта)
SELECT orders.id, customers.name, orders.total_amount FROM orders
LEFT JOIN customers ON orders.customer_id = customers.id;

-- 11. RIGHT JOIN (усі клієнти, навіть якщо немає замовлення)
SELECT orders.id, customers.name, orders.total_amount FROM orders
RIGHT JOIN customers ON orders.customer_id = customers.id;

-- 12. FULL JOIN (усі клієнти та замовлення)
SELECT orders.id, customers.name, orders.total_amount FROM orders
FULL JOIN customers ON orders.customer_id = customers.id;

-- 13. CROSS JOIN (всі можливі комбінації кур'єрів і складів)
SELECT couriers.name, warehouses.location FROM couriers
CROSS JOIN warehouses;

-- 14. SELF JOIN (товари з однаковою ціною)
SELECT p1.name AS Product1, p2.name AS Product2 FROM products p1
```

JOIN products p2 ON p1.price = p2.price AND p1.id <> p2.id;

-- 15. Підзапит у WHERE (знайти товари з ціною вище середнього)

SELECT * FROM products WHERE price > (SELECT AVG(price) FROM products);

-- 16. Підзапит у IN (знайти клієнтів, які мають замовлення)

SELECT * FROM customers WHERE id IN (SELECT customer_id FROM orders);

-- 17. Підзапит у NOT EXISTS (клієнти без замовлень)

SELECT * FROM customers c WHERE NOT EXISTS (SELECT 1 FROM orders o WHERE o.customer_id = c.id);

-- 18. EXISTS (чи є товари на складі у Києві)

SELECT * FROM warehouses w WHERE EXISTS (SELECT 1 FROM products p WHERE p.warehouse_id = w.id AND w.location LIKE 'Київ%');

-- 19. UNION (усі імена клієнтів і кур'єрів)

SELECT name FROM customers UNION SELECT name FROM couriers;

-- 20. INTERSECT (кур'єри та клієнти з однаковими іменами)

SELECT name FROM customers INTERSECT SELECT name FROM couriers;

-- 21. EXCEPT (клієнти, які не є кур'єрами)

SELECT name FROM customers EXCEPT SELECT name FROM couriers;

-- 22. CTE - тимчасова таблиця з дорогими товарами

WITH ExpensiveProducts AS (SELECT * FROM products WHERE price > 500)

SELECT * FROM ExpensiveProducts;

-- 23. Віконна функція - ранжування товарів за ціною

SELECT id, name, price, RANK() OVER (ORDER BY price DESC) AS price_rank FROM products;

-- 24. Віконна функція - сукупна сума замовлень

SELECT id, customer_id, total_amount, SUM(total_amount) OVER (PARTITION BY customer_id) AS customer_total FROM orders;

-- 25. Визначити найпопулярніший склад

SELECT warehouse_id, COUNT(*) AS total_products FROM products GROUP BY warehouse_id ORDER BY total_products DESC LIMIT 1;

-- 26. Знайти товари з найбільшим запасом

SELECT * FROM products ORDER BY stock_quanity DESC LIMIT 4;

-- 27. Визначити клієнта з найбільшою кількістю замовлень

SELECT customer_id, COUNT(*) AS order_count FROM orders GROUP BY customer_id ORDER BY order_count DESC LIMIT 1;

-- 28. Кількість замовлень для кожного статусу

SELECT status, COUNT(*) FROM orders GROUP BY status;

-- 29. Загальна сума всіх замовлень

SELECT SUM(total_amount) FROM orders;

-- 30. Всі товари, що знаходяться на складі у Києві

SELECT * FROM products WHERE warehouse_id = (SELECT id FROM warehouses WHERE location LIKE 'Київ%');

-- 31. Всі товари, яких більше 10 одиниць на складі

SELECT * FROM products WHERE stock_quanity > 10;

-- 32. Клієнти, які зробили хоча б одне замовлення на суму більше 200

SELECT DISTINCT customers.* FROM customers

JOIN orders ON customers.id = orders.customer_id WHERE orders.total_amount > 200;

-- 33. Клієнти, які ще не зробили жодного замовлення

SELECT * FROM customers WHERE id NOT IN (SELECT customer_id FROM orders);

-- 34. Середня сума замовлення для кожного клієнта

SELECT customer_id, AVG(total_amount) FROM orders GROUP BY customer_id;

-- 35. Останні 5 замовлень

SELECT * FROM orders ORDER BY created_at DESC LIMIT 5;

-- 36. Замовлення за останній місяць

SELECT * FROM orders WHERE created_at >= NOW() - INTERVAL '1 month';

-- 37. Найбільш частий спосіб оплати

SELECT payment_method, COUNT(*) AS method_count FROM payments GROUP BY payment_method ORDER BY method_count DESC LIMIT 1;

-- 38. Підрахунок кур'єрів за транспортСкладні запити:

SELECT vehicle, COUNT(*) FROM couriers GROUP BY vehicle;

-- 39. Замовлення, в яких загальна сума більше середньої

SELECT * FROM orders WHERE total_amount > (SELECT AVG(total_amount) FROM orders);

-- 40. Найменший запас товару

SELECT * FROM products ORDER BY stock_quanity ASC LIMIT 1;

Виконаймо найскладніші запити SQL (ті, про які я розповідав на захисті):

-- 22. CTE - тимчасова таблиця з дорогими товарами

WITH ExpensiveProducts AS (SELECT * FROM products WHERE price > 500)

SELECT * FROM ExpensiveProducts;

Цей запит вибирає всі дорогі товари з таблиці `products`, ціна яких перевищує 500. CTE використовують для того, щоб спростити складні запити, роблячи їх більш читабельними та керованими.

	id [PK] integer	name character varying (100)	description text	price double precision	stock_quanity integer	warehouse_id integer
1	1	Ноутбук	Потужний ноутбук для роботи	1500	10	1
2	2	Смартфон	Новий смартфон з великим екраном	800	15	2

-- 24. Віконна функція - сукупна сума замовлень

SELECT id, customer_id, total_amount, SUM(total_amount) OVER (PARTITION BY customer_id) AS customer_total FROM orders;

Загалом, цей запит вибирає всі замовлення та обчислює сукупну суму замовлень для кожного покупця. Для кожного замовлення буде показано, скільки в сумі витратив кожен покупець на всі свої замовлення.

	id [PK] integer	customer_id integer	total_amount double precision	customer_total double precision
1	1	1	250.5	250.5
2	2	2	120	120
3	3	3	310.75	310.75
4	4	4	95.3	95.3

-- 26. Знайти товари з найбільшим запасом

SELECT * FROM products ORDER BY stock_quanity DESC LIMIT 4;

Отже, цей запит повертає перші чотири товари з найбільшим запасом у таблиці products, відсортовані від найбільшого до найменшого.

	id [PK] integer	name character varying (100)	description text	price double precision	stock_quanity integer	warehouse_id integer
1	4	Рюкзак	Зручний рюкзак для ноутбука	50	25	4
2	3	Навушники	Бездротові навушники	150	20	3
3	2	Смартфон	Новий смартфон з великим екраном	800	15	2
4	1	Ноутбук	Потужний ноутбук для роботи	1500	10	1

Висновки: Робота з базами даних для управління студентськими оцінками дала мені можливість не лише закріпити теоретичні знання, а й набути практичних навичок у складанні складних SQL-запитів. Використання різноманітних операторів, таких як SELECT, JOIN, WHERE, агрегаційні функції та віконні функції, дозволяє ефективно працювати з даними та отримувати необхідні звіти для аналізу.

Особливо важливим для мене стало освоєння таких концепцій, як підзапити, Common Table Expressions (CTE) і віконні функції, що допомогли організувати роботу з даними таким чином, щоб запити стали більш ефективними і зручними для читання та підтримки. Це значно полегшує роботу з великими обсягами даних, оскільки можна створювати тимчасові таблиці для подальших операцій, що особливо корисно при розв’язуванні складних завдань.