# Аналітичні SQL-запити (OLAP)

### 1. Агрегації (SUM, AVG, COUNT, MIN, MAX, GROUP BY)

### 1) Кількість замовлень кожного покупця

```
SELECT
    customers.id AS customer_id,
    customers.name AS customer_name,
    COUNT(orders.id) AS order_count
FROM customers
LEFT JOIN orders ON customers.id = orders.customer_id
GROUP BY customers.id, customers.name;
```

**Що робить запит:**
Підраховує, скільки замовлень зробив кожен клієнт.
`LEFT JOIN` дає можливість побачити навіть клієнтів без жодного замовлення (у них буде 0).

### 2) Середня ціна товарів у кожній категорії

```
SELECT
    categories.name AS category_name,
    AVG(products.price) AS average_price
FROM categories
JOIN products ON products.category_id = categories.id
GROUP BY categories.name;
```

**Що робить:**
Для кожної категорії товарів рахує середню ціну товарів, що до неї належать.

### 3) Мінімальна і максимальна кількість товарів у замовленнях

```
SELECT
    MIN(order_items.quantity) AS minimum_quantity,
    MAX(order_items.quantity) AS maximum_quantity
FROM order_items;
```

**Що робить:**
Надає загальні мінімальне та максимальне значення поля `quantity` серед усіх позицій замовлень.

### 4) Загальна виручка з кожного продукту

```sql
SELECT
    products.name AS product_name,
    SUM(order_items.quantity * products.price) AS total_revenue
FROM order_items
JOIN products ON order_items.product_id = products.id
GROUP BY products.name;
```

**Що робить:**
Підраховує суму зароблених грошей з кожного товару: (кількість × ціна) для всіх замовлень.

## 2. JOIN-запити (INNER JOIN, LEFT JOIN, CROSS JOIN)

### 5) Список замовлень з покупцем і працівником

```sql
SELECT
    orders.id AS order_id,
    customers.name AS customer_name,
    employees.name AS employee_name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.id
INNER JOIN employees ON orders.employee_id = employees.id;
```

**Що робить:**
Показує всі замовлення разом із покупцем та співробітником, що їх оформлював.

### 6) Усі товари разом з категоріями

```sql
SELECT
    products.name AS product_name,
    categories.name AS category_name
FROM products
LEFT JOIN categories ON products.category_id = categories.id;
```

**Що робить:**
Показує всі продукти і їхні категорії.
`LEFT JOIN` дозволяє бачити навіть ті товари, що не мають категорії.

### 7) Усі можливі пари працівник–покупець

```sql
SELECT
    employees.name AS employee_name,
    customers.name AS customer_name
```

```
FROM employees
CROSS JOIN customers;
```

**Що робить:**
Генерує всі можливі комбінації «співробітник × покупець».
Використовує CROSS JOIN (декартовий добуток).

# 3. Підзапити (в SELECT, WHERE, HAVING)

### 8) Кількість товарів у кожному замовленні

```
SELECT
    orders.id AS order_id,
    (SELECT COUNT(*) FROM order_items WHERE order_items.order_id =
orders.id) AS items_count
FROM orders;
```

**Що робить:**
Використовує підзапит у SELECT, щоб підрахувати, скільки продуктів міститься в кожному замовленні.

### 9) Покупці, які зробили хоча б одне замовлення

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

**Що робить:**
Повертає тільки тих клієнтів, які є в таблиці `orders` (тобто зробили хоча б одне замовлення).

### 10) Категорії, де середня ціна товарів понад 1000

```
SELECT
    categories.name AS category_name,
    AVG(products.price) AS average_price
FROM categories
JOIN products ON products.category_id = categories.id
GROUP BY categories.name
HAVING AVG(products.price) > 1000;
```

**Що робить:**
Показує тільки ті категорії, у яких середня ціна всіх товарів більша за 1000.
`HAVING` використовується для фільтрації уже агрегованих значень.