1a

CREATE SCHEMA LibraryManagement

1b

CREATE TABLE authors (

author\_id INT AUTO\_INCREMENT PRIMARY KEY,

author\_name VARCHAR(45)

);

1c

CREATE TABLE genres (

genre\_id INT AUTO\_INCREMENT PRIMARY KEY,

genre\_name VARCHAR(45)

);

1d

CREATE TABLE books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

publication\_year YEAR,

author\_id INT,

FOREIGN KEY(author\_id) REFERENCES authors(author\_id),

genre\_id INT,

FOREIGN KEY(genre\_id) REFERENCES genres(genre\_id)

);

1e

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(45),

email VARCHAR(45)

);

1f

CREATE TABLE borrowed\_books (

borrow\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

FOREIGN KEY(book\_id) REFERENCES books(book\_id),

user\_id INT,

FOREIGN KEY(user\_id) REFERENCES users(user\_id),

borrow\_date DATE,

return\_date DATE

);

2.

INSERT INTO authors (author\_id, author\_name)

VALUES (1, 'Stephen King'), (2, 'J.K Rowling');

INSERT INTO genres (genre\_id, genre\_name)

VALUES (101, 'Horror'), (102, 'Fantasy');

INSERT INTO users (user\_id, username, email)

VALUES (201, 'Mark', 'mark@gmail.com'), (202, 'Ken', 'ken@gmail.com'), (203, 'Lisy', '[lisy@gmail.com](mailto:lisy@gmail.com)');

INSERT INTO books (book\_id, title, publication\_year, author\_id, genre\_id)

VALUES (301, 'The Shining','2002', 1, 101),

(302, 'Harry Potter. Part 1', '1997', 2, 102),

(303, 'Harry Potter. Part 2', '1999', 2, 102);

INSERT INTO borrowed\_books (borrow\_id, book\_id, user\_id, borrow\_date, return\_date)

VALUES (400, 301, 203, '2024-01-13', '2024-02-13'),

(401, 302, 202, '2024-02-20', '2024-03-20'),

(402, 303, 201, '2024-03-15', '2024-04-15');

3.

Можна просто зробити SELECT \* FROM … але якщо вказати колонки, які відображати, то таблиця виглядає правильніше.

SELECT o.id, o.date, p.name as product\_name, p.price, od.quantity,

cat.name as category\_name, sh.name as shipper\_name, s.contact as suplier\_contact,

e.last\_name as emploee\_name, c.name as customer\_name, c.address, c.city, c.postal\_code FROM order\_details od

INNER JOIN orders o on od.order\_id = o.id

INNER JOIN customers c on c.id = o.customer\_id

INNER JOIN products p on p.id = od.product\_id

INNER JOIN categories cat on cat.id = p.category\_id

INNER JOIN employees e on e.employee\_id = o.employee\_id

INNER JOIN shippers sh on sh.id = o.shipper\_id

INNER JOIN suppliers s on p.supplier\_id

4.1

SELECT COUNT(\*) FROM order\_details od

INNER JOIN orders o on od.order\_id = o.id

INNER JOIN customers c on c.id = o.customer\_id

INNER JOIN products p on p.id = od.product\_id

INNER JOIN categories cat on cat.id = p.category\_id

INNER JOIN employees e on e.employee\_id = o.employee\_id

INNER JOIN shippers sh on sh.id = o.shipper\_id

INNER JOIN suppliers s on p.supplier\_id

Результат: 15022

4.2.

SELECT COUNT(\*) FROM order\_details od

LEFT JOIN orders o on od.order\_id = o.id

LEFT JOIN customers c on c.id = o.customer\_id

LEFT JOIN products p on p.id = od.product\_id

LEFT JOIN categories cat on cat.id = p.category\_id

RIGHT JOIN employees e on e.employee\_id = o.employee\_id

LEFT JOIN shippers sh on sh.id = o.shipper\_id

LEFT JOIN suppliers s on p.supplier\_id;

Результат: 15023

Кількість змінюється. Aле це залежить чи є в таблицях, що порівнюються, значення які не використовуються в іншій таблиці. Наприклад, в таблиці employees один з employee\_id ніколи не зустрічається в таблиці orders й тому при UNION він не враховувався, а при RIGHT JOIN зʼявився в записі, що змінює кількість рядків.

4.3.

SELECT COUNT(\*) FROM order\_details od

LEFT JOIN orders o on od.order\_id = o.id

LEFT JOIN customers c on c.id = o.customer\_id

LEFT JOIN products p on p.id = od.product\_id

LEFT JOIN categories cat on cat.id = p.category\_id

RIGHT JOIN employees e on e.employee\_id = o.employee\_id

LEFT JOIN shippers sh on sh.id = o.shipper\_id

LEFT JOIN suppliers s on p.supplier\_id

WHERE e.employee\_id >3 and e.employee\_id < 10

4.4.

SELECT cat.name as category, COUNT(cat.name) as count, AVG(od.quantity) as avarage\_quantity

FROM order\_details od

LEFT JOIN orders o on od.order\_id = o.id

LEFT JOIN customers c on c.id = o.customer\_id

LEFT JOIN products p on p.id = od.product\_id

LEFT JOIN categories cat on cat.id = p.category\_id

RIGHT JOIN employees e on e.employee\_id = o.employee\_id

LEFT JOIN shippers sh on sh.id = o.shipper\_id

LEFT JOIN suppliers s on p.supplier\_id

WHERE e.employee\_id >3 and e.employee\_id < 10

GROUP BY cat.name

4.5.

SELECT cat.name as category, COUNT(cat.name) as count, AVG(od.quantity) as avarage\_quantity

FROM order\_details od

LEFT JOIN orders o on od.order\_id = o.id

LEFT JOIN customers c on c.id = o.customer\_id

LEFT JOIN products p on p.id = od.product\_id

LEFT JOIN categories cat on cat.id = p.category\_id

RIGHT JOIN employees e on e.employee\_id = o.employee\_id

LEFT JOIN shippers sh on sh.id = o.shipper\_id

LEFT JOIN suppliers s on p.supplier\_id

WHERE e.employee\_id >3 and e.employee\_id < 10

GROUP BY cat.name

HAVING avarage\_quantity > 21;

4.6.

SELECT cat.name as category, COUNT(cat.name) as count, AVG(od.quantity) as avarage\_quantity

FROM order\_details od

LEFT JOIN orders o on od.order\_id = o.id

LEFT JOIN customers c on c.id = o.customer\_id

LEFT JOIN products p on p.id = od.product\_id

LEFT JOIN categories cat on cat.id = p.category\_id

RIGHT JOIN employees e on e.employee\_id = o.employee\_id

LEFT JOIN shippers sh on sh.id = o.shipper\_id

LEFT JOIN suppliers s on p.supplier\_id

WHERE e.employee\_id >3 and e.employee\_id < 10

GROUP BY cat.name

HAVING avarage\_quantity > 21

ORDER BY COUNT(cat.name) DESC;

4.7

SELECT cat.name as category, COUNT(cat.name) as count, AVG(od.quantity) as avarage\_quantity

FROM order\_details od

LEFT JOIN orders o on od.order\_id = o.id

LEFT JOIN customers c on c.id = o.customer\_id

LEFT JOIN products p on p.id = od.product\_id

LEFT JOIN categories cat on cat.id = p.category\_id

RIGHT JOIN employees e on e.employee\_id = o.employee\_id

LEFT JOIN shippers sh on sh.id = o.shipper\_id

LEFT JOIN suppliers s on p.supplier\_id

WHERE e.employee\_id >3 and e.employee\_id < 10

GROUP BY cat.name

HAVING avarage\_quantity > 21

ORDER BY COUNT(cat.name) DESC

LIMIT 4 OFFSET 1