| Завдання | Код |  |
| --- | --- | --- |
| 1 | |  |
| 1а | create database LibraryManagement;  use LibraryManagement; |  |
| 1b | create table authors (  author\_id INT primary key auto\_increment,  author\_name VARCHAR (255)  ); |  |
| 1c | create table genres (  genre\_id INT primary key auto\_increment,  genre\_name VARCHAR (255)  ); |  |
| 1d | create table books (  book\_id INT primary key auto\_increment,  title VARCHAR (255),  publication\_year YEAR,  author\_id INT,  genre\_id INT,  FOREIGN KEY (author\_id) references authors (author\_id),  FOREIGN KEY (genre\_id) references genres (genre\_id)  ); |  |
| 1e | create table users (  user\_id INT primary key auto\_increment,  username VARCHAR (255),  email VARCHAR (255)  ); |  |
| 1f | create table borrowed\_books (  borrow\_id INT primary key auto\_increment,  book\_id int,  user\_id int,  borrow\_date date,  return\_date date,  FOREIGN KEY (book\_id) references books (book\_id),  FOREIGN KEY (user\_id) references users (user\_id)  ); |  |
| 2 | |  |
| 2.1 | insert into authors (author\_name)  values ( 'J.R.R. Tolkien'), ( 'Stephen King'), ('Dan Brown'); |  |
| 2.2 | insert into genres (genre\_name)  values ('Horror'), ('Thriller'), ('Fantasy'); |  |
| 2.3 | insert into books (title,publication\_year,author\_id,genre\_id) values  ('The Hobbit', 1937, 1, 3),  ('The Da Vinci Code', 2003, 3, 2),  ('The Shining', 1977, 2, 1); |  |
| 2.4 | insert into users (username, email) values  ('Oleg', 'oleg@gmail.com'), ('Lidia', 'lidia@gmail.com'), ('Alla', 'alla@gmail.com'); |  |
| 2.5 | insert into borrowed\_books (book\_id, user\_id,borrow\_date, return\_date) values  (1, 1, '2020-01-02', '2020-03-05'),  (3, 2, '2021-05-02', '2021-07-09'),  (2, 3, '2023-01-04', '2023-03-07'),  (1, 3, '2021-02-02', '2021-04-08'); |  |
| 3 | |  |
| 3 | SELECT \* FROM order\_details  INNER JOIN orders on order\_details.order\_id = orders.id  INNER JOIN customers on orders.customer\_id = customers.id  INNER JOIN products on order\_details.product\_id = products.id  INNER JOIN categories on products.category\_id = categories.id  INNER JOIN employees on orders.employee\_id = employees.employee\_id  INNER JOIN shippers on orders.shipper\_id = shippers.id  INNER JOIN suppliers on products.supplier\_id = suppliers.id |  |
| 4 | |  |
| 4.1 | SELECT count(\*) as total\_rows  FROM order\_details  INNER JOIN orders on order\_details.order\_id = orders.id  INNER JOIN customers on orders.customer\_id = customers.id  INNER JOIN products on order\_details.product\_id = products.id  INNER JOIN categories on products.category\_id = categories.id  INNER JOIN employees on orders.employee\_id = employees.employee\_id  INNER JOIN shippers on orders.shipper\_id = shippers.id  INNER JOIN suppliers on products.supplier\_id = suppliers.id |  |
| 4.2 | SELECT count(\*) as total\_rows  FROM order\_details  RIGHT JOIN orders on order\_details.order\_id = orders.id  RIGHT JOIN customers on orders.customer\_id = customers.id  LEFT JOIN products on order\_details.product\_id = products.id  LEFT JOIN categories on products.category\_id = categories.id  LEFT JOIN employees on orders.employee\_id = employees.employee\_id  LEFT JOIN shippers on orders.shipper\_id = shippers.id  LEFT JOIN suppliers on products.supplier\_id = suppliers.id | Оскільки змінюється тип з'єднання на LEFT JOIN або RIGHT JOIN, кількість рядків може змінитися, бо тепер включаються всі рядки з однієї з таблиць навіть у випадку, якщо вони не мають збігів з іншою таблицею. |
| 4.3 | SELECT employees.employee\_id as employee\_3\_9  FROM order\_details  INNER JOIN orders on order\_details.order\_id = orders.id  INNER JOIN customers on orders.customer\_id = customers.id  INNER JOIN products on order\_details.product\_id = products.id  INNER JOIN categories on products.category\_id = categories.id  INNER JOIN employees on orders.employee\_id = employees.employee\_id  INNER JOIN shippers on orders.shipper\_id = shippers.id  INNER JOIN suppliers on products.supplier\_id = suppliers.id  where employees.employee\_id > 3 and employees.employee\_id <= 10 |  |
| 4.4 | SELECT categories.name,  count(\*) as total\_rows,  avg (order\_details.quantity) as quantity  FROM order\_details  RIGHT JOIN orders on order\_details.order\_id = orders.id  RIGHT JOIN customers on orders.customer\_id = customers.id  LEFT JOIN products on order\_details.product\_id = products.id  LEFT JOIN categories on products.category\_id = categories.id  LEFT JOIN employees on orders.employee\_id = employees.employee\_id  LEFT JOIN shippers on orders.shipper\_id = shippers.id  LEFT JOIN suppliers on products.supplier\_id = suppliers.id  where employees.employee\_id > 3 and employees.employee\_id <= 10  GROUP BY categories.name |  |
| 4.5 | SELECT categories.name,  count(\*) as total\_rows,  avg (order\_details.quantity) as quantity  FROM order\_details  RIGHT JOIN orders on order\_details.order\_id = orders.id  RIGHT JOIN customers on orders.customer\_id = customers.id  LEFT JOIN products on order\_details.product\_id = products.id  LEFT JOIN categories on products.category\_id = categories.id  LEFT JOIN employees on orders.employee\_id = employees.employee\_id  LEFT JOIN shippers on orders.shipper\_id = shippers.id  LEFT JOIN suppliers on products.supplier\_id = suppliers.id  where employees.employee\_id > 3 and employees.employee\_id <= 10  GROUP BY categories.name  having avg (order\_details.quantity) > 21 |  |
| 4.6 | SELECT categories.name,  count(\*) as total\_rows,  avg (order\_details.quantity) as quantity  FROM order\_details  RIGHT JOIN orders on order\_details.order\_id = orders.id  RIGHT JOIN customers on orders.customer\_id = customers.id  LEFT JOIN products on order\_details.product\_id = products.id  LEFT JOIN categories on products.category\_id = categories.id  LEFT JOIN employees on orders.employee\_id = employees.employee\_id  LEFT JOIN shippers on orders.shipper\_id = shippers.id  LEFT JOIN suppliers on products.supplier\_id = suppliers.id  where employees.employee\_id > 3 and employees.employee\_id <= 10  GROUP BY categories.name  having avg (order\_details.quantity) > 21  order by total\_rows desc |  |
| 4.7 | SELECT categories.name,  count(\*) as total\_rows,  avg (order\_details.quantity) as quantity  FROM order\_details  RIGHT JOIN orders on order\_details.order\_id = orders.id  RIGHT JOIN customers on orders.customer\_id = customers.id  LEFT JOIN products on order\_details.product\_id = products.id  LEFT JOIN categories on products.category\_id = categories.id  LEFT JOIN employees on orders.employee\_id = employees.employee\_id  LEFT JOIN shippers on orders.shipper\_id = shippers.id  LEFT JOIN suppliers on products.supplier\_id = suppliers.id  where employees.employee\_id > 3 and employees.employee\_id <= 10  GROUP BY categories.name  having avg (order\_details.quantity) > 21  order by total\_rows desc  limit 4 offset 1 |  |