Group 16: Lab Session 3 Report

Begenjov, Agamyrat
Parayno, Gail Rayla Emanuelle
Saparbai, Aidana Saparbai kyzy

How to use?

You can load the database by executing the sql script in lab5.sql.

We have separate text files for more distinction between the parts (ddl.txt → data.txt → query.txt)

Definition for each file:

ddl.txt:

This SQL script defines multiple tables for a comprehensive relational database schema. The Customer table holds information about customers, including unique identifiers, contact details, and categorization. The Bank and CreditCard tables manage financial data, establishing relationships between users, banks, and credit cards. The Orders table captures order details, linked to customers, while the Transaction, Invoice, and Payment tables handle transactional and invoicing data. Logistics are tracked through the Shipment table, and the Shop table manages information about different shops. Product-related data is organized through the ProductType, Product, Photo, and OrderItem tables, specifying details about products, their types, images, and order-related information. Lastly, the CanSell table establishes relationships between product types and shops, indicating which products are available in each shop. The script utilizes various data types and foreign key constraints to maintain data integrity and relational coherence.

data.txt

The provided SQL script includes sample data inserts for various tables in the database schema. It populates the Customer table with information about five customers, such as usernames, emails, addresses, and categories. The Bank and CreditCard tables are populated with data representing different banks and credit cards. The Orders table contains sample order information, including order IDs, dates, statuses, and user IDs. The Transaction, Invoice, and Payment tables are populated with sample data related to transactions, invoices, and payments, respectively. The Shipment table is filled with data about shipments, including IDs, dates, and tracking numbers. The Shop table includes information about five different shops. Sample data is also added to the ProductType and Product tables, detailing product types

and their attributes. The Photo table includes file names for product photos. Lastly, the OrderItem table is populated with sample order items, indicating details about unit prices, quantities, statuses, and payment statuses for each order. The provided data inserts are essential for testing and validating the functionality of the database schema.

Data Types

The database definition script encompasses a variety of data types carefully selected to accurately represent the nature of the data stored within each column. These data types play a crucial role in defining the structure of the relational database, ensuring proper storage and retrieval of information. Below is an overview of the key data types employed in the script:

INT (Integer):

- **Example:** customer_id INT NOT NULL
- Usage: Primarily used for unique identifiers, such as customer IDs, order IDs, and product IDs. This data type is essential for storing whole numbers.

VARCHAR(n) (Variable-Length Character String):

- Example: username VARCHAR(30)
- Usage: This variable-length character string is utilized for columns that store textual data, such as usernames, emails, and names. The 'n' parameter specifies the maximum length of the string.

DATE:

- Example: expiration_date DATE
- Usage: Specifically designed for storing date values, the DATE data type is employed for columns like expiration_date **in the** CreditCard **table.**

DATETIME:

- **Examples:** order_date DATETIME**,** shipment_date DATETIME**,** payment_date DATETIME
- Usage: Combining date and time information, the DATETIME data type is applied to columns where both temporal details are relevant, such as order dates, shipment dates, and payment dates.

VARCHAR without length:

- Example: status VARCHAR(30)
- Usage: A generic VARCHAR data type without a specified length is employed for columns storing various status information, including order status, invoice status, and transaction methods.

INT (for Foreign Keys):

- Example: user_id INT**,** type INT
- Usage: Similar to its use for primary keys, the INT data type is used for columns referencing the primary key of another table, establishing foreign key relationships.

DOUBLE (for Price):

- Example: price INT
- Usage: The DOUBLE data type is used for the price **column in the** Product **table to represent numeric values with decimal points, specifically the price of products.**

VARCHAR for Color, Size, and Description:

- **Examples:** color VARCHAR(30)**,** size VARCHAR(30)**,** description VARCHAR(100)
- Usage: VARCHAR data type is applied to descriptive columns such as color, size, and product descriptions, allowing flexibility in storing variable-length text.

query.txt

The accompanying SQL queries showcase a range of operations, including hierarchy retrieval of product types, random customer selection, aggregation of paid order items by product type, identification of frequently co-purchased products, and retrieval of customer emails for shipped orders. These queries demonstrate the flexibility and utility of the database.

Sample Queries

All outcomes from the sample queries align precisely with our expectations based on the data section.

Sample query (1)

```
--Could you list the descriptions of product types that are categorised at the second
tier? Note:

--Product types without a parent are deemed first-tier, and their immediate offspring
types are

--considered the second tier.

SELECT DISTINCT T.id, P.description

FROM ProductType AS T

JOIN Product AS P ON T.id = P.type

WHERE T.parent IS NULL;
```

Result:

```
SELECT DISTINCT T.id, P.description
FROM ProductType AS T
JOIN Product AS P ON T.id = P.type
WHERE T.parent IS NULL;  30ms
```

Result(RO)

Search results

	id int	description varchar
1	1	High-end smartphone
2	2	Cotton T-shirt
3	3	Bestseller novel
4	4	Comfortable sofa
5	5	Durable basketball

Sample query (2)

```
--Can you randomly pick three customers and share their email addresses?

SELECT customer_id, email

FROM Customer

ORDER BY RAND()

LIMIT 3;
```

Result:

```
--Can you randomly pick three customers and share their email addresses?
▷ Execute
SELECT customer_id, email
FROM Customer
ORDER BY RAND()
LIMIT 3;  6ms
```

omer ×

Search results Cost: 16ms 1 2 Total

customer_i int	email varchar(50)
1	user1@example.com
3	user3@example.com
2	user2@example.com

Sample query (3)

```
--Can you pinpoint the three product type IDs that have the highest sales quantities?
Only

--consider products that have been ordered and paid for, disregarding their shipment
status.

SELECT product_type_id, SUM(quantity) AS total

FROM OrderItem

WHERE status_of_payment = 'paid'

GROUP BY product_type_id

ORDER BY total DESC

LIMIT 3;
```

Result:

```sql
--Can you pinpoint the three product type IDs that have the highest sales quantities? Only
--consider products that have been ordered and paid for, disregarding their shipment status.
SELECT product_type_id, SUM(quantity) AS total
FROM OrderItem
WHERE status_of_payment = 'paid'
GROUP BY product_type_id
ORDER BY total DESC
LIMIT 3;  4ms
```

OrderItem

Search results Cost: 5ms 1 Total 3

	product_type_id int	total newdecim
1	4	4
2	2	3
3	1	2

Sample query (4)

```sql
-- a query that finds the customers who have placed the most

-- orders along with the total amount they have spent. Here's the query:

SELECT C.customer_id, C.name, COUNT(O.order_id) AS total_orders, SUM(OI.quantity *
OI.unit_price) AS total_spent

FROM Customer AS C

JOIN Orders AS O ON C.customer_id = O.user_id

JOIN OrderItem AS OI ON O.order_id = OI.order_id

WHERE OI.status_of_payment = 'Paid'

GROUP BY C.customer_id

ORDER BY total_orders DESC, total_spent DESC

LIMIT 3;
```

Result:

```sql
-- Certainly! Let's create a query that finds the customers who have placed the most
-- orders along with the total amount they have spent. Here's the query:
SELECT C.customer_id, C.name, COUNT(O.order_id) AS total_orders, SUM(OI.quantity * OI.unit_pr
FROM Customer AS C
JOIN Orders AS O ON C.customer_id = O.user_id
JOIN OrderItem AS OI ON O.order_id = OI.order_id
WHERE OI.status_of_payment = 'Paid'
GROUP BY C.customer_id
ORDER BY total_orders DESC, total_spent DESC
LIMIT 3; 3ms
```

Result(RO)

Cost: 3ms Total 3

	customer_id int	name varchar	total_orders bigint	total_spent newdecimal
1	4	Alice Johnson	1	240
2	1	John Doe	1	100
3	2	Jane Doe	1	75

Query 5:

retrieves the product types and the corresponding products that have the highest average price:

```sql
SELECT PT.id AS product_type_id, AVG(P.price) AS average_price
FROM ProductType AS PT
JOIN Product AS P ON PT.id = P.type
GROUP BY PT.id
ORDER BY average_price DESC
LIMIT 3; 48ms
```

Result(RO)

Cost: 48ms

	product_type_id int	average_price newdecimal
1	4	799.0000
2	1	499.0000
3	7	120.0000

Query 6:

--the top 3 most popular product colors based on the total quantity ordered:

```
SELECT P.color, SUM(OI.quantity) AS total_quantity_ordered
FROM Product AS P
JOIN OrderItem AS OI ON P.id = OI.product_id
GROUP BY P.color
ORDER BY total_quantity_ordered DESC
LIMIT 3;  12ms
```

esult(RO)

Search results

Cost: 12ms

	color varchar	total_quantity_ordered newdecimal
1	Red	9
2	Purple	8
3	Denim Blue	7

Name and Signature	**Individual Contribution for Lab Session 3 Submission**	**Percentage of Contribution (100% in total)**
Agamyrat Begenjov	Primarily created the queries.	33.33
Parayno, Gail Rayla Emanuelle	Focused on creating final report and polishing all required submissions.	33.33
Aidana Saparbai kyzy	Created data and defined data, created the video.	33.33