Aim of Practical 5: Write and execute SQL queries subqueries, joins.

SHIVANSH LOHANI
23070521141

Practical 5 part 1

SQL Subqueries

What is a Subquery?

A **subquery** is a query inside another query, used to retrieve intermediate results before executing the main query.

Types of Subqueries:

1. **Single-row subqueries** → Return a single value.
2. **Multi-row subqueries** → Return multiple values.
3. **Correlated subqueries** → Reference columns from the outer query.

Example Database: Supermarket

Use a **SupermarketDB** with the following tables:

Customer (customer_id, name, email, phone, address)
Product (product_id, name, category, price, stock_quantity)
Order_Details (order_id, customer_id, order_date, total_amount)
Order_Item (order_id, product_id, quantity, subtotal)
Employeee (employee_id, name, role, salary, hire_date)

Customer Table

customer_id	name	email	phone	address
1	Alice Johnson	alice@gmail.com	9876543210	New York
2	Bob Smith	bob@yahoo.com	9123456789	Los Angeles
3	Charlie Brown	charlie@outlook.com	9998887776	Chicago
4	David Miller	david@gmail.com	8765432109	Miami
5	Emily Davis	emily@hotmail.com	7654321098	New York

Product Table

product_id	name	category	price ($)	stock_quantity
1	Milk	Dairy	2.50	50
2	Bread	Bakery	1.80	30
3	Eggs	Dairy	3.20	40
4	Chicken	Meat	7.50	20
5	Apples	Fruit	1.20	60
6	Orange Juice	Beverage	3.50	25

Order_Details Table

order_id	customer_id	order_date	total_amount ($)
101	1	2024-01-10	10.50
102	2	2024-01-12	55.20
103	3	2023-12-01	40.80
104	4	2023-11-05	30.00
105	5	2024-02-10	25.50

Order_Item Table

order_id	product_id	quantity	subtotal ($)
101	1	2	5.00
101	2	3	5.40
102	3	1	3.20
102	4	2	15.00
103	5	5	6.00
104	6	2	7.00

Employee Table

employee_id	name	role	salary ($)	hire_date
1	Michael Scott	Manager	75000.00	2020-05-10
2	Jim Halpert	Cashier	30000.00	2021-08-15
3	Pam Beesly	Sales Associate	28000.00	2022-02-20
4	Dwight Schrute	Supervisor	50000.00	2019-11-30
5	Kevin Malone	Cashier	29000.00	2023-03-10

Examples of Subqueries

Find customers who placed orders over $50.00

```
SELECT * FROM Customer
WHERE customer_id IN (SELECT customer_id FROM Order_Details
WHERE total_amount > 50);
```

Retrieve products that cost more than the average product price

```
SELECT * FROM Product
WHERE price > (SELECT AVG(price) FROM Product);
```

Find employees earning more than the lowest manager's salary

```
SELECT * FROM Employeee
WHERE salary > (SELECT MIN(salary) FROM Employeee WHERE role =
'Manager');
```

Find employees hired after the most recent hire date of a cashier

```
SELECT * FROM Employeee
WHERE hire_date > (SELECT MAX(hire_date) FROM Employeee WHERE
role = 'Cashier');
```

Find customers who haven't placed any orders

```
SELECT * FROM Customer
WHERE customer_id NOT IN (SELECT customer_id FROM
Order_Details);
```

Find the name of the highest-paid employee

```
SELECT name FROM Employeee
WHERE salary = (SELECT MAX(salary) FROM Employeee);
```

Retrieve the total revenue generated from orders placed in January 2024

```
SELECT SUM(total_amount)
FROM Order_Details
WHERE order_date BETWEEN TO_DATE('2024-01-01', 'YYYY-MM-DD')
AND TO_DATE('2024-01-31', 'YYYY-MM-DD');
```

Find the most ordered product

```
SELECT name FROM Product
WHERE product_id = (SELECT product_id FROM (
 SELECT product_id, SUM(quantity) AS total_sold  FROM
Order_Item
 GROUP BY product_id
 ORDER BY total_sold DESC
 )
 WHERE ROWNUM = 1
);
```

🔹 Step-by-Step Execution

1️⃣ Analyze the `Order_Item` Table

Before writing the query, let's check the data from `Order_Item`:

order_id	product_id	quantity	subtotal ($)
101	1	2	5.00
101	2	3	5.40
102	3	1	3.20
102	4	2	15.00
103	5	5	6.00
104	6	2	7.00

2️⃣ Query Breakdown

The **inner subquery** calculates the **total quantity** sold for each product:

```sql
SELECT product_id, SUM(quantity) AS total_sold
FROM Order_Item
GROUP BY product_id
ORDER BY total_sold DESC;
```

✅ This returns:

product_id	total_sold
5	5
2	3
1	2
4	2
6	2
3	1

3️⃣ Select the Top 1 Product

Now, we **limit the results** to only the **top product** using `WHERE ROWNUM = 1`:

```sql
SELECT product_id FROM (
    SELECT product_id, SUM(quantity) AS total_sold
    FROM Order_Item
    GROUP BY product_id
    ORDER BY total_sold DESC
)
WHERE ROWNUM = 1;
```

✅ This gives us:

product_id
5

4️⃣ Retrieve the Product Name

Now, we use this `product_id` to **fetch the product name** from the `Product` table:

```sql
SELECT name FROM Product
WHERE product_id = (SELECT product_id FROM (
    SELECT product_id, SUM(quantity) AS total_sold
    FROM Order_Item
    GROUP BY product_id
    ORDER BY total_sold DESC
)
WHERE ROWNUM = 1);
```

✅ This gives us:

name
Apples

Retrieve employees who earn above the average salary of all employees

```
SELECT * FROM Employeee
WHERE salary > (SELECT AVG(salary) FROM Employeee);
```

Find customers who placed orders only in 2023 but not in 2024

```
SELECT * FROM Customer
WHERE customer_id IN (
 SELECT customer_id FROM Order_Details
  WHERE  order_date  BETWEEN  TO_DATE('2023-01-01',  'YYYY-MM-DD')
AND TO_DATE('2023-12-31', 'YYYY-MM-DD') )
AND customer_id NOT IN (
 SELECT customer_id FROM Order_Details
  WHERE  order_date  BETWEEN  TO_DATE('2024-01-01',  'YYYY-MM-DD')
AND TO_DATE('2024-12-31', 'YYYY-MM-DD') );
```

Subquery Tasks

1. Find customers who placed orders over **$50**.

```
mysql> SELECT * FROM Customer
    -> WHERE customer_id IN (
    ->     SELECT customer_id FROM Order_Details
    ->     WHERE total_amount > 50
    -> );
+-------------+---------+---------------------+------------+-------------+
| customer_id | name    | email               | phone      | address     |
+-------------+---------+---------------------+------------+-------------+
|           1 | Alice   | alice@example.com   | 1234567890 | 123 Main St |
|           3 | Charlie | charlie@example.com | 5678901234 | 789 Oak St  |
+-------------+---------+---------------------+------------+-------------+
2 rows in set (0.01 sec)
```

2. Retrieve products that cost more than the **average product price**. 3.

```
mysql> SELECT * FROM Product
    -> WHERE price > (SELECT AVG(price) FROM Product);
+------------+--------+-------------+--------+----------------+
| product_id | name   | category    | price  | stock_quantity |
+------------+--------+-------------+--------+----------------+
|          1 | Laptop | Electronics | 800.00 |             10 |
+------------+--------+-------------+--------+----------------+
1 row in set (0.01 sec)
```

Find employees hired after the **most recent hire date of a cashier**.

```
mysql> SELECT * FROM Employeee
    -> WHERE hire_date > (
    ->     SELECT MAX(hire_date) FROM Employeee WHERE role = 'Cashier'
    -> );
Empty set (0.00 sec)
```

4. List customers who **haven't placed any orders**.

```
mysql> SELECT * FROM Customer
    -> WHERE customer_id NOT IN (
    ->     SELECT customer_id FROM Order_Details
    -> );
Empty set (0.00 sec)
```

5. Retrieve employees who earn **above the average salary**.

```
mysql> SELECT * FROM Employeee
    -> WHERE salary > (SELECT AVG(salary) FROM Employeee);
+-------------+-------+---------+----------+------------+
| employee_id | name  | role    | salary   | hire_date  |
+-------------+-------+---------+----------+------------+
|           1 | David | Manager | 50000.00 | 2023-01-15 |
+-------------+-------+---------+----------+------------+
1 row in set (0.00 sec)
```