Experiment No.7
Nested queries and Complex queries
Date of Performance:07/03/25
Date of Submission:21/03/25

Aim: Nested queries and Complex queries

Objective: In nested queries, a query is written inside a query. The result of inner query is used in execution of outer query

Theory:

Sample table: Salesman

salesman_id name city commission

Sample table: Orders

ord_no purch_amt ord_date customer_id salesman_id

Questions

1. Write a query to display all the orders from the orders table issued by the salesman 'Paul Adam'.

2 Write a query to display all the orders for the salesman who belongs to the city London.

3 Write a query to find all the orders issued against the salesman who may works for customer whose id is 3007

4 Write a query to display all the orders which values are greater than the average order value for 10th October 2012

5 Write a query to find all orders attributed to a salesman in New york.

6 Write a query to display the commission of all the salesmen servicing customers in Paris

Implementation:

```
CREATE DATABASE SalesDB;
USE SalesDB;

CREATE TABLE Salesman (
```

```
    salesman_id INT PRIMARY KEY,
    name VARCHAR(255),
    city VARCHAR(255),
    commission DECIMAL(10, 2)
);

CREATE TABLE Orders (
    ord_no INT PRIMARY KEY,
    purch_amt DECIMAL(10, 2),
    ord_date DATE,
    customer_id INT,
    salesman_id INT
);

CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    customer_city VARCHAR(255),
    ord_no INT
);

INSERT INTO Salesman (salesman_id, name, city, commission) VALUES
(1, 'Paul Adam', 'London', 2500.50),
(2, 'John Doe', 'New York', 1800.75),
(3, 'Jane Smith', 'London', 2000.00),
(4, 'Chris Green', 'Paris', 2200.00),
(5, 'Alice Brown', 'New York', 1900.25),
(6, 'David White', 'New York', 2100.00);

INSERT INTO Orders (ord_no, purch_amt, ord_date, customer_id, salesman_id) VALUES
(101, 500.00, '2012-10-10', 1001, 1),
(102, 800.00, '2012-10-10', 1002, 4),
(103, 1200.00, '2012-10-11', 1003, 2),
(104, 450.00, '2012-10-10', 1004, 1),
(105, 700.00, '2012-10-12', 1005, 4),
(106, 1000.00, '2012-10-10', 1006, 5),
(107, 1200.00, '2012-10-11', 1007, 6),
(108, 950.00, '2012-10-10', 3007, 1),
(109, 550.00, '2012-10-11', 1008, 3);
```

```
INSERT INTO Customers (customer_id, customer_name, customer_city, ord_no) VALUES
(1001, 'Customer A', 'London', 101),
(1002, 'Customer B', 'Paris', 102),
(1003, 'Customer C', 'New York', 103),
(1004, 'Customer D', 'London', 104),
(1005, 'Customer E', 'Paris', 105),
(1006, 'Customer F', 'New York', 106),
(1007, 'Customer G', 'Paris', 108),
(1008, 'Customer H', 'Paris', 109),
(3007, 'Customer X', 'New York', 108);

ALTER TABLE Customers
ADD CONSTRAINT fk_ord_no FOREIGN KEY (ord_no) REFERENCES Orders(ord_no);

SELECT o.*
FROM Orders o
JOIN Salesman s ON o.salesman_id = s.salesman_id
WHERE s.name = 'Paul Adam';

SELECT o.*
FROM Orders o
JOIN Salesman s ON o.salesman_id = s.salesman_id
WHERE s.city = 'London';

SELECT o.*
FROM Orders o
JOIN Salesman s ON o.salesman_id = s.salesman_id
JOIN Customers c ON o.customer_id = c.customer_id
WHERE c.customer_id = 3007;

SELECT *
FROM Orders
WHERE purch_amt > (
   SELECT AVG(purch_amt)
   FROM Orders
   WHERE ord_date = '2012-10-10'
);

SELECT o.*
FROM Orders o
```

```
JOIN Salesman s ON o.salesman_id = s.salesman_id
WHERE s.city = 'New York';

SELECT DISTINCT s.salesman_id, s.name, s.commission
FROM Salesman s
JOIN Orders o ON s.salesman_id = o.salesman_id
JOIN Customers c ON o.customer_id = c.customer_id
WHERE c.customer_city = 'Paris';
```

Output:

1.
```
SELECT o.*
FROM Orders o
JOIN Salesman s ON o.salesman_id = s.salesman_id
WHERE s.name = 'Paul Adam';
```

ord_no	purch_amt	ord_date	customer_id	salesman_id
101	500.00	2012-10-10	1001	1
104	450.00	2012-10-10	1004	1
108	950.00	2012-10-10	3007	1

2.
```
SELECT o.*

FROM Orders o
JOIN Salesman s ON o.salesman_id = s.salesman_id
WHERE s.city = 'London';
```

ord_no	purch_amt	ord_date	customer_id	salesman_id
101	500.00	2012-10-10	1001	1
104	450.00	2012-10-10	1004	1
108	950.00	2012-10-10	3007	1
109	550.00	2012-10-11	1008	3

3.
```
SELECT o.*
FROM Orders o
JOIN Salesman s ON o.salesman_id = s.salesman_id
JOIN Customers c ON o.customer_id = c.customer_id
```

WHERE c.customer_id = 3007;

ord_no	purch_amt	ord_date	customer_id	salesman_id
108	950.00	2012-10-10	3007	1

4. SELECT *
FROM Orders
WHERE purch_amt > (
SELECT AVG(purch_amt)
FROM Orders
WHERE ord_date = '2012-10-10'
);

ord_no	purch_amt	ord_date	customer_id	salesman_id
102	800.00	2012-10-10	1002	4
103	1200.00	2012-10-11	1003	2
106	1000.00	2012-10-10	1006	5
107	1200.00	2012-10-11	1007	6
108	950.00	2012-10-10	3007	1
NULL	NULL	NULL	NULL	NULL

5. SELECT o.*
FROM Orders o
JOIN Salesman s ON o.salesman_id = s.salesman_id
WHERE s.city = 'New York';

ord_no	purch_amt	ord_date	customer_id	salesman_id
103	1200.00	2012-10-11	1003	2
106	1000.00	2012-10-10	1006	5
107	1200.00	2012-10-11	1007	6

6. SELECT DISTINCT s.salesman_id, s.name, s.commission
FROM Salesman s
JOIN Orders o ON s.salesman_id = o.salesman_id
JOIN Customers c ON o.customer_id = c.customer_id
WHERE c.customer_city = 'Paris';

salesman_id	name	commission
4	Chris Green	2200.00
6	David White	2100.00
3	Jane Smith	2000.00

Conclusion: The experiment successfully demonstrated the use of nested and complex queries in SQL to retrieve meaningful data from relational databases. By leveraging inner and outer queries, critical insights such as orders based on specific criteria, commission details, and comparison of order values were efficiently extracted. This highlights the versatility and power of SQL in handling intricate database operations.