Database Systems
Practical 5
Subquery and Set Operations
Version 1.1

Subquery/Nested query is a query in a query. A subquery is usually added in the WHERE Clause of the sql statement. A Subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value.

Subqueries are an alternate way of returning data from multiple tables. Subqueries can be used with the following sql statements along with the comparison operators like =, <, >, >=, <= etc.

```
SELECT
INSERT
UPDATE
DELETE
```

Run the script named **3.sql**. This script will create a table called students with 3 columns- roll, name and subject and it will create another table called english_group. It will insert some data into students table as well. Take a look at the table with a SELECT * FROM students command. You see- there are two students taking Science as their subject. Statically, if you want to know students who are taking Science, you can execute the following command-

```
SELECT roll, name
FROM students
WHERE name IN ('Ratul', 'Nafee');
```

But, to execute such query dynamically, you can run the following-

```
SELECT roll, name
FROM students
WHERE name IN (     SELECT name
             FROM students
             WHERE subject= 'Science');
```

In this case, the query within first bracket will output ‘Ratul’ and ‘Nafee’- is that similar to the first query then?

Now, we will insert data into english_group table with a subquery. We will insert those students of students table who are taking English as their subject into English_group table. Execute the following-

```
INSERT INTO english_group(roll, name)
SELECT roll, name
FROM students WHERE subject= 'English';
```

In this particular session of the practical, we will dig into set operations in Oracle. Firstly, run **"lab5table.sql" then "lab5data1.sql" and then "lab5data2.sql"** with START command.

Firstly, see what the output is for the following query. You don't have to know the relations between the tables. Just focus on the output.

```
SELECT cust_nbr, name
       FROM customer
       WHERE region_id = 5;
```

The output is as follows.

CUST_NBR	NAME
1	Cooper Industries
2	Emblazon Corp.
3	Ditech Corp.
4	Flowtech Inc.
5	Gentech Industries

Next, see what the output is for the following query.

```
SELECT c.cust_nbr, c.name
       FROM customer c
       WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                                   FROM cust_order o, employee e
                                   WHERE o.sales_emp_id = e.emp_id
                                   AND e.lname = 'MARTIN');
```

It actually retrieves all customers with sales representative named MARTIN.

The output is as follows.

CUST_NBR	NAME
4	Flowtech Inc.
8	Zantech Inc.

UNION ALL

The UNION ALL operator merges the result sets of two component queries. This operation returns rows retrieved by either of the component queries, without eliminating duplicates. The following example illustrates the UNION ALL operation:

```
       SELECT cust_nbr, name
       FROM customer
       WHERE region_id = 5
       UNION ALL
       SELECT c.cust_nbr, c.name
       FROM customer c
       WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                                   FROM cust_order o, employee e
                                   WHERE o.sales_emp_id = e.emp_id
                                   AND e.lname = 'MARTIN');
```

What is the output?

UNION
The UNION operator returns all distinct rows retrieved by two component queries. The UNION operation eliminates duplicates while merging rows retrieved by either of the component queries. The following example illustrates the UNION operation:

```
SELECT cust_nbr, name
FROM customer
WHERE region_id = 5
UNION
SELECT c.cust_nbr, c.name
FROM customer c
WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                                  FROM cust_order o, employee e
                            WHERE o.sales_emp_id = e.emp_id
                     AND e.lname = 'MARTIN');
```

What is the output? What is the difference with previous one?

INTERSECT
INTERSECT returns only the rows retrieved by both component queries. Compare this with UNION, which returns the rows retrieved by any of the component queries. If UNION acts like "OR," INTERSECT acts like "AND." For example:

```
SELECT cust_nbr, name
FROM customer
WHERE region_id = 5
INTERSECT
SELECT c.cust_nbr, c.name
FROM customer c
WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                                  FROM cust_order o, employee e
                                  WHERE o.sales_emp_id = e.emp_id
                                  AND e.lname = 'MARTIN');
```

What is the result of the query?

MINUS
MINUS returns all rows from the first SELECT that are not also returned by the second SELECT.

```
SELECT cust_nbr, name
FROM customer
WHERE region_id = 5
MINUS
SELECT c.cust_nbr, c.name
FROM customer c
WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                     FROM cust_order o, employee e
                                  WHERE o.sales_emp_id = e.emp_id
                                  AND e.lname = 'MARTIN');
```

Note the effect of minus. What is the difference with INTERSECT?

The results of UNION, UNION ALL, and INTERSECT will not change if you alter the ordering of component queries. However, the result of MINUS will be different if you alter the order of the component queries. If you rewrite the previous query by switching the positions of the two SELECTs, you get a completely different result:

```
SELECT c.cust_nbr, c.name
FROM customer c
WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                                FROM cust_order o, employee e
                                WHERE o.sales_emp_id = e.emp_id
                                AND e.lname = 'MARTIN')
MINUS
SELECT cust_nbr, name
FROM customer
WHERE region_id = 5;
```

Precedence of Set Operators

If more than two component queries are combined using set operators, then Oracle evaluates the set operators from left to right. In the following example, the UNION is evaluated before the INTERSECT:

```
SELECT cust_nbr, name
FROM customer
WHERE region_id = 5
UNION
SELECT c.cust_nbr, c.name
FROM customer c
WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                                FROM cust_order o, employee e
                                WHERE o.sales_emp_id = e.emp_id
                                AND e.lname = 'MARTIN')
INTERSECT
SELECT cust_nbr, name
FROM customer
WHERE region_id = 6;
```

To influence a particular order of evaluation of the set operators, you can use parentheses. Looking at the preceding example, if you want the INTERSECT to be evaluated before the UNION, you should introduce parentheses into the query such that the component queries involving the INTERSECT are enclosed in parentheses, as shown in the following example:

```
SELECT cust_nbr, name
FROM customer
WHERE region_id = 5
UNION
(
SELECT c.cust_nbr, c.name
FROM customer c
WHERE c.cust_nbr IN        (SELECT o.cust_nbr
                                FROM cust_order o, employee e
                                WHERE o.sales_emp_id = e.emp_id
                                AND e.lname = 'MARTIN')
INTERSECT
SELECT cust_nbr, name
FROM customer
WHERE region_id = 6
);
```