

Set-operations, Subquery and correlated Subquery

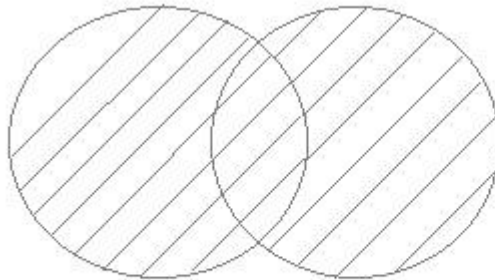
Set Operations in SQL

SQL supports few Set operations which can be performed on the table data. These are used to get meaningful results from data stored in the table, under different special conditions.

- UNION
- UNION ALL
- INTERSECT
- MINUS

UNION Operation:

- **UNION** is used to combine the results of two or more **SELECT** statements. However, it will eliminate duplicate rows from its result set. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.



Example

Table1

ID	Name
1	abhi
2	adam

Table2

ID	Name
2	adam
3	Chester

```
SELECT * FROM Table1 UNION SELECT * FROM Table2;
```

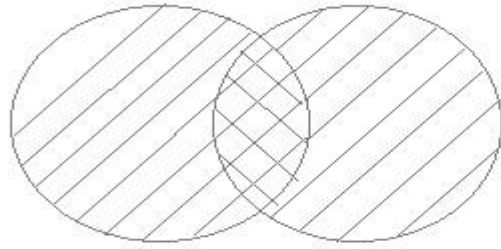
OUTPUT

ID	NAME
1	abhi
2	adam
3	Chester

Union All

UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.



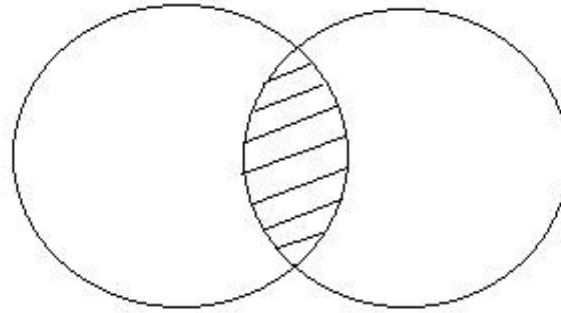
```
SELECT * FROM Table1 UNION ALL SELECT * FROM Table2;
```

OUTPUT

ID	NAME
1	abhi
2	adam
2	adam
3	Chester

Intersection

Intersect operation is used to combine two **SELECT** statements, but it only returns the records which are common from both **SELECT** statements. In case of **Intersect** the number of columns and datatype must be same.



```
SELECT * FROM Table1 INTERSECT SELECT * FROM Table2;
```

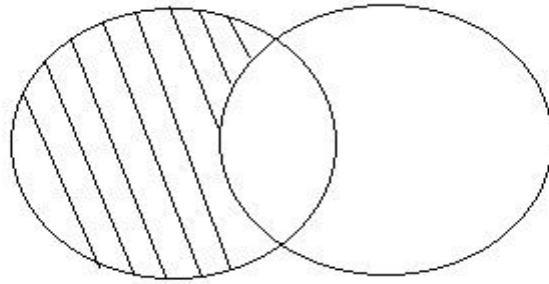
OUTPUT

ID	NAME
2	adam

```
SELECT ..... FROM table1 ;  
WHERE col IN ( SELECT col  
FROM table2 )
```

Minus

The Minus operation combines results of two **SELECT** statements and return only those in the final result, which belongs to the first set of the result.



```
SELECT * FROM Table1 MINUS SELECT * FROM Table2;
```

OUTPUT

ID	NAME
1	abhi

```
SELECT ..... FROM table1 ;  
WHERE col NOT IN ( SELECT  
                  col FROM table2 )
```

Sub-query

- ❑ A sub-query is a SELECT statement that is embedded in a clause of another SELECT statement.
- ❑ The inner query or the sub-query returns value that is used by the outer query or the main query.
- ❑ Using a sub-query is equivalent to performing two sequential queries and using the result of the inner query as the search value for the outer query.
- ❑ They can very useful when we need to select rows from a table with a condition that depends on the data in the table itself.

Using a Subquery to Solve a Problem

Who has a salary greater than Abel's?

Main Query:



Which employees have salaries greater than Abel's salary?

Subquery



What is Abel's salary?



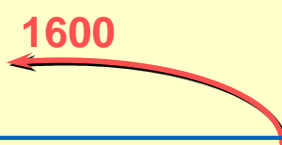
Subquery Syntax

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT      select_list
         FROM         table);
```

- The subquery (inner query) executes once before the main query.
- The result of the subquery is used by the main query (outer query).

Using a Subquery

```
SELECT ENAME
FROM   emp
WHERE  salary >
      (select sal from emp where
       ename= 'ALLEN' ) ;
```



A red arrow points from the value '1600' to the subquery result, indicating that the main query's WHERE clause is filtering for salaries greater than 1600.

ENAME
JONES
BLAKE
CLARK

Guidelines for Using Subqueries

- Enclose subqueries in parentheses.
- Place subqueries on the right side of the comparison condition.
- The `ORDER BY` clause in the subquery is not needed unless you are performing Top-N analysis.
- Use single-row operators with single-row subqueries and use multiple-row operators with multiple-row subqueries.

Types of Subqueries

- **Single-row subquery**



- **Multiple-row subquery**



Single-Row Subqueries

- Return only one row
- Use single-row comparison operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Executing Single-Row Subqueries

Display the details of those employees whose dept-id is same as the dept-id of employee id 7521 and salary more than the salary of employee id 7876.

```
select ename, sal from  
emp where  
deptno =
```

ST CLERK

```
(select deptno from emp  
where empno = 7521)
```

```
AND salary >
```

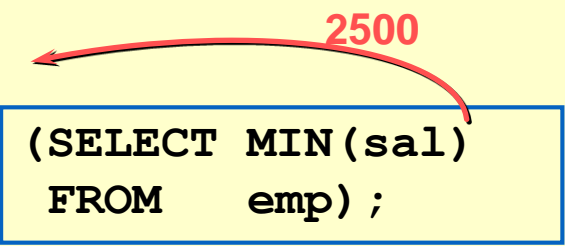
2600

```
(select sal from emp  
where empno = 7876);
```

ENAME	SAL
ALLEN	1600
WARD	1250
MARTIN	1250
BLAKE	2850
TURNER	1500

Display the employee details
whose salary is equal to the
minimum salary

```
SELECT  ename, sal
FROM    emp
WHERE   sal =
```



The diagram illustrates the execution of the SQL query. A red curved arrow originates from the value '2500' and points to the 'sal =' part of the WHERE clause. The subquery '(SELECT MIN(sal) FROM emp);' is enclosed in a blue box, and the arrow indicates that its result, 2500, is used to filter the employees.

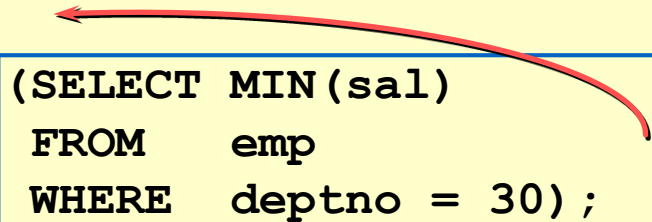
```
(SELECT MIN(sal)
FROM    emp);
```

The HAVING Clause with Subqueries

- The Oracle server executes subqueries first.
- The Oracle server returns results into the HAVING clause of the main query.

Query: To display all the departments that have a minimum salary greater than that of the minimum salary of department 30.

```
SELECT deptno, MIN(sal)
FROM emp
GROUP BY deptno
HAVING MIN(sal) >
    (SELECT MIN(sal)
     FROM emp
     WHERE deptno = 30);
```

A red curved arrow points from the subquery '(SELECT MIN(sal) FROM emp WHERE deptno = 30);' to the 'MIN(sal)' part of the 'HAVING MIN(sal) >' clause in the main query.

What is Wrong with this Statement?

```
SELECT empno, ename
FROM emp
WHERE sal =
      (SELECT MIN(sal)
       FROM emp
       GROUP BY deptno);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

Single-row operator with multiple-row subquery

Multiple-Row Subqueries

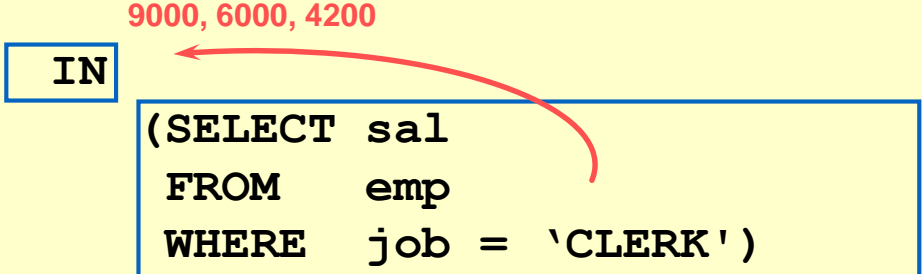
- Queries that return more than one row from the inner SELECT statement.
- Use multiple-row comparison operators.

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

Using the IN Operator in Multiple-Row Subqueries

Display the details of those employees whose salary is EQUAL TO any 'CLERK' and who are not 'CLERK'

```
SELECT ename, empno, sal
FROM emp
WHERE sal IN
      (SELECT sal
       FROM emp
       WHERE job = 'CLERK')
AND job <> 'CLERK';
```

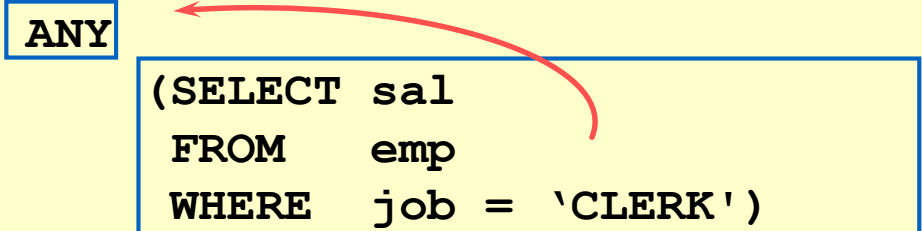


ENAME	EMPNO	SAL
SCOTT	7788	3000
FORD	7902	3000

Using the ANY Operator in Multiple-Row Subqueries

Display the details of those employees whose salary is less than any 'CLERK' and who are not 'CLERK'

```
SELECT empno, ename, job, sal
FROM emp
WHERE sal < ANY
      (SELECT sal
       FROM emp
       WHERE job = 'CLERK')
AND job <> 'CLERK';
```

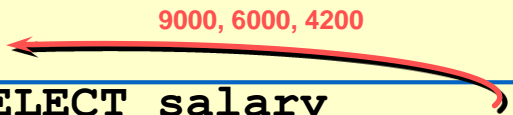


EMPNO	ENAME	JOB	SAL
7521	WARD	SALESMAN	1250
7654	MARTIN	SALESMAN	1250
7844	TURNER	SALESMAN	1500
7499	ALLEN	SALESMAN	1600
7782	CLARK	MANAGER	2450

Using the ALL Operator in Multiple-Row Subqueries

Display the details of those employees whose salary is less than all
'ANALYST' and who are not 'ANALYST',

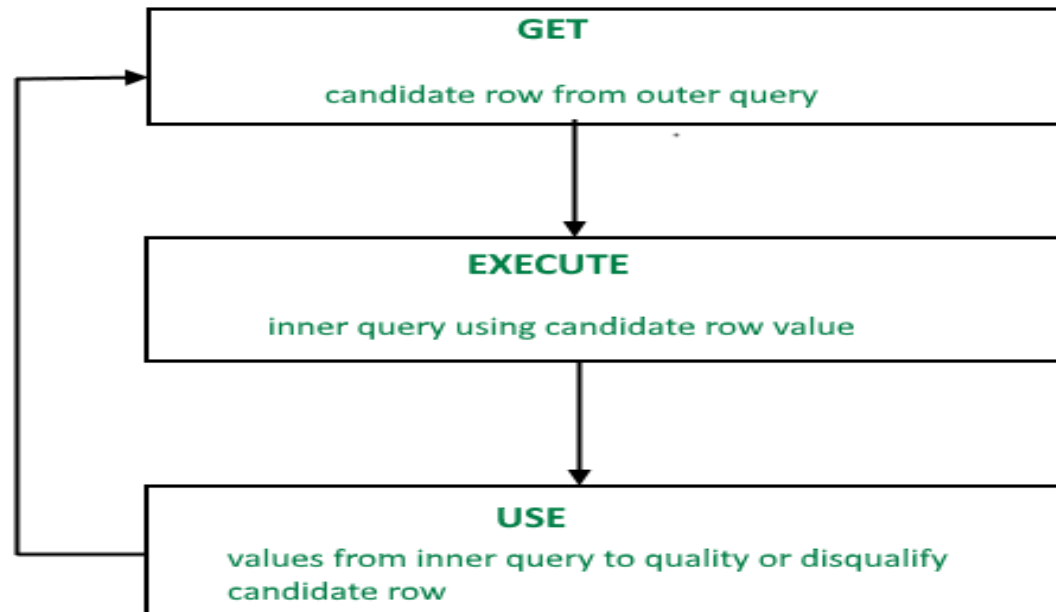
```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ALL
      (SELECT salary
       FROM employees
       WHERE job_id = 'ANALYST')
AND job_id <> 'ANALYST';
```



EMPNO	ENAME	JOB	SAL
7566	JONES	MANAGER	2975
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7499	ALLEN	SALESMAN	1600
7844	TURNER	SALESMAN	1500

Correlated Subquery

- Oracle performs a correlated sub-query when the sub-query references a column from a table referred in the parent statement.
- Each subquery is executed once for every row of the outer query.



Correlated Subquery

- A correlated subquery is evaluated once for each row processed by the parent statement.
- The parent statement can be a **SELECT**, **UPDATE**, or **DELETE** statement.

```
SELECT column1, column2, .... FROM table1 outer WHERE column1  
operator (SELECT column1, column2 FROM table2 WHERE expr1 =  
outer.expr2);
```

Correlated Subquery Example

- finds all employees whose salary is higher than the average salary of the employees in their departments
- **SELECT** empno, ename, sal, deptno **FROM** emp e **WHERE** sal > (SELECT **AVG**(sal) **FROM** emp **WHERE** deptno = e.deptno) **ORDER BY** deptno;

Correlated Subquery

- A correlated subquery is one way of reading every row in a table and comparing values in each row against related data.
- It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query.
- In other words, you can use a correlated subquery to answer a multipart question whose answer depends on the value in each row processed by the parent statement.
- **Nested Subqueries Versus Correlated Subqueries**

Correlated Subquery

- **Nested Subqueries Versus Correlated Subqueries :**
- With a normal nested subquery, the inner **SELECT** query runs first and executes once, returning values to be used by the main query.
- A correlated subquery, however, executes once for each candidate row considered by the outer query.
- In other words, the inner query is driven by the outer query.
- **NOTE :** You can also use the **Exist**, **ANY** and **ALL** operator in a correlated subquery.

The EXISTS Operator

Allows you to determine if the value in a correlated subquery exists. If exist, it returns true, else returns false.

Display those customer name from customer table whose customer id is found in the order table.

correlated subquery:

```
SELECT CustomerName  
FROM Customers C  
WHERE EXISTS  
(SELECT * FROM ORDERS O  
WHERE C.CustomerID=O.CustomerID) ;
```

Examples

- Find all the employees who earn more than the average salary in their department.

```
SELECT ename, salary, department_id FROM employees e WHERE salary > (SELECT  
AVG(salary) FROM employees WHERE department_id = e.department_id);
```

- Find employees who have at least one person reporting to them.

```
SELECT emp_id, ename, job_id, department_id FROM employees e WHERE EXISTS  
(SELECT emp_id FROM employees WHERE mgr_id = e.emp_id);
```

- Find the highest-paid employee of each department.

```
SELECT department_id, ename, salary from employees e WHERE salary = (SELECT  
max(salary) from employees WHERE department_id = e. department_id);
```

- List the employee names whose salary is greater than the lowest salary of employee belonging to department number 20;

```
SELECT ename FROM emp WHERE sal>ANY (SELECT sal FROM emp WHERE deptno=20);
```

- List the employee names whose salary is greater than the highest salary of all employees belonging to department number 20;

```
SELECT ename FROM emp WHERE sal> ALL (SELECT sal FROM emp WHERE deptno=20);
```