



Merit-Quality-Excellence

Sukkur IBA University

DATABASE MANAGEMENT SYSTEM

LAB No: 06

Objective of Lab No. 6:

After performing lab 6, students will be able to:

- Subqueries
- Single Row Subquery
- Multiple Row Subquery
- Multiple column subquery
- Correlated Subquery
- Nested Subqueries
- Using the EXISTS Operator

Subquery

A Subquery or Inner query or a Nested query is a query within another SQL query.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, WHERE, FROM clause along with the operators like =, <, >, >=, <=, IN, BETWEEN, etc. It can be use with SELECT, UPDATE DELETE INSERT statements or inside another subquery.

There are a few rules that subqueries must follow –

- Subqueries must be enclosed within parentheses.
- A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.
- An ORDER BY command cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY command can be used to perform the same function as the ORDER BY in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN, ANY, ALL, SOME operator.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.
- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

Syntax:

```
SELECT    select_list
FROM      table
WHERE     expr operator

          (SELECT    select_list
           FROM      table);
```

Example 1: Subqueries with the WHERE Statement

Show all students who get better marks than that of the student whose StudentID is 'V002'.

To solve the problem, we require two queries. One query returns the marks (stored in Total marks field) of 'V002' and a second query identifies the students who get better marks than the result of the first query.

First Query:

```
1  SELECT *
2  FROM `marks`
3  WHERE studentid = 'V002';
```

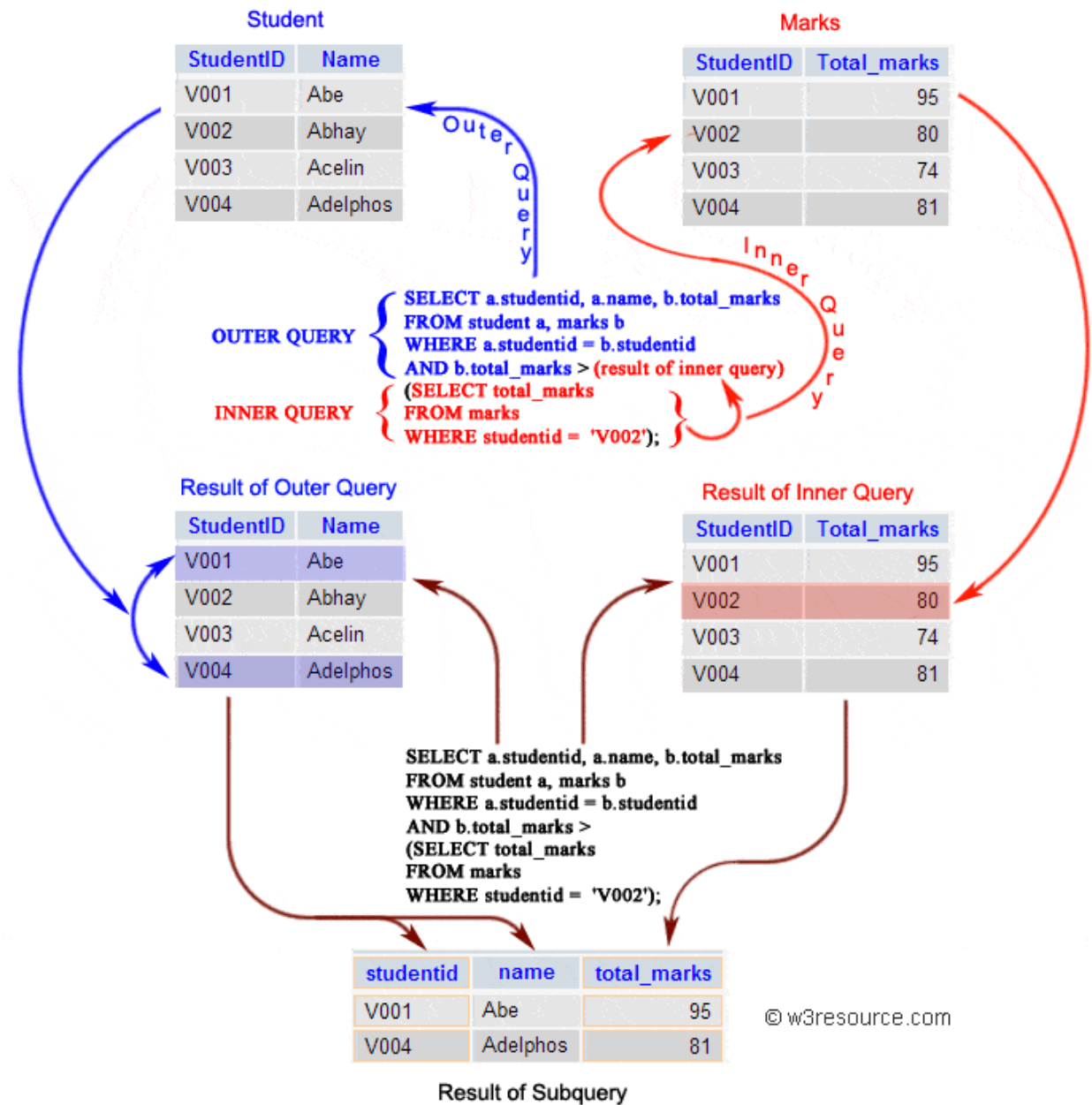
You can combine the above two queries by placing one query inside the other. The subquery (also called the 'inner query') is the query inside the parentheses. See the following code and query result:

Second Query:

```
1  SELECT a.studentid, a.name, b.total_marks
2  FROM student a, marks b
3  WHERE a.studentid = b.studentid
4  AND b.total_marks > 80;
```

Combine:

```
1  SELECT a.studentid, a.name, b.total_marks
2  FROM student a, marks b
3  WHERE a.studentid = b.studentid AND b.total_marks >
4  (SELECT total_marks
5  FROM marks
6  WHERE studentid = 'V002');
```



Example 2: Subqueries with in the FROM Statement

From clause can be used to specify a sub-query expression in SQL. The relation produced by the sub-query is then used as a new relation on which the outer query is applied.

```
SELECT department_ID, salary
FROM (SELECT *
FROM employees
WHERE last_name like "A%" AND hire_date between "1990-01-01" AND "2000-12-31") AS MyTable;
```

Example 3: Subqueries with in the SELECT Statement

Subqueries can also be included in a select statement to bring summary values in to a detailed dataset. Subquery with Select statement are used to return single aggregate value in return.

When a subquery is placed within the column list it is used to return single values. In this case, you can think of the subquery as a single value expression. The result returned is no different than the expression “2 + 2.”

```
Select department_ID, avg(salary),  
(SELECT AVG(salary)  
FROM employees) as "All department AVG Salary"  
FROM employees group by department_id;
```

Comparisons Operator in Subqueries

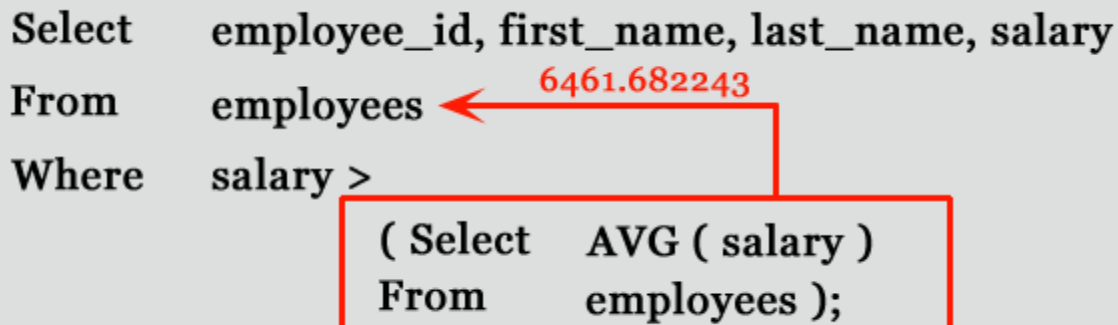
A subquery can be used before or after any of the comparison operators. The subquery can return at most one value. The value can be the result of an arithmetic expression or a column function. SQL then compares the value that results from the subquery with the value on the other side of the comparison operator. You can use the following comparison operators:

Operator	Description
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
!=	Not equal to
<>	Not equal to
<=>	NULL-safe equal to operator

MySQL **null safe equal** (<=>) to operator performs an equality comparison like the equal to (=) operator, but returns 1 rather than NULL if both operands are NULL, and 0 rather than NULL if one operand is NULL.

For example, suppose you want to find the employee id, first_name, last_name, and salaries for employees whose average salary is higher than the average salary throughout the company.

```
Select    employee_id, first_name, last_name, salary
From      employees
Where     salary >
          ( Select    AVG ( salary )
            From      employees );
```



MySQL Subqueries with ALL, ANY, IN, or SOME

You can use a subquery after a comparison operator, followed by the keyword ALL, ANY, or SOME.

The ALL operator compares value to every value returned by the subquery. Therefore ALL operator (which must follow a comparison operator) returns TRUE if the comparison is TRUE for ALL of the values in the column that the subquery returns.

The ALL operator returns true if all of the subquery values meet the condition.

ALL Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
      (SELECT column_name FROM table_name WHERE condition);
```

NOT IN is an alias for <> ALL. Thus, these two statements are the same:

```
SELECT c1 FROM t1 WHERE c1 <> ALL (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 NOT IN (SELECT c1 FROM t2);
```

The following query selects the department with the highest average salary. The subquery finds the average salary for each department, and then the main query selects the department with the highest average salary.

```
SELECT department_id, AVG(SALARY)
FROM EMPLOYEES GROUP BY department_id
HAVING AVG(SALARY) >= ALL
      (SELECT AVG(SALARY) FROM EMPLOYEES GROUP BY department_id);
```

Note: Here we have used ALL keyword for this subquery as the department selected by the query must have an average salary greater than or equal to all the average salaries of the other departments.

The ANY operator compares the value to each value returned by the subquery. Therefore ANY keyword (which must follow a comparison operator) returns TRUE if the comparison is TRUE for ANY of the values in the column that the subquery returns.

The ANY operator returns true if any of the subquery values meet the condition.

ANY Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
(SELECT column_name FROM table_name WHERE condition);
```

The following query selects any employee who works in the location 1700. The subquery finds the department id in the 1700 location, and then the main query selects the employees who work in any of these departments.

```
SELECT first_name, last_name, department_id
FROM employees WHERE department_id= ANY
(SELECT DEPARTMENT_ID FROM departments WHERE location_id=1700);
```

Note: We have used ANY keyword in this query because it is likely that the subquery will find more than one departments in 1700 location. If you use the ALL keyword instead of the ANY keyword, no data is selected because no employee works in all departments of 1700 location

When used with a subquery, the word IN (equal to any member of the list) is an alias for = ANY. Thus, the following two statements are the same:

```
SELECT c1 FROM t1 WHERE c1 = ANY (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 IN (SELECT c1 FROM t2);
```

The word SOME is an alias for ANY. Thus, these two statements are the same:

```
SELECT c1 FROM t1 WHERE c1 <> ANY (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 <> SOME (SELECT c1 FROM t2);
```

Type of Subqueries

1. Single row subquery : Returns zero or one row.
2. Multiple row subquery : Returns one or more rows.
3. Multiple column subqueries : Returns one or more columns.

4. Correlated subqueries : Reference one or more columns in the outer SQL statement.
The subquery is known as a correlated subquery because the subquery is related to the outer SQL statement.
5. Nested subqueries : Subqueries are placed within another subquery.

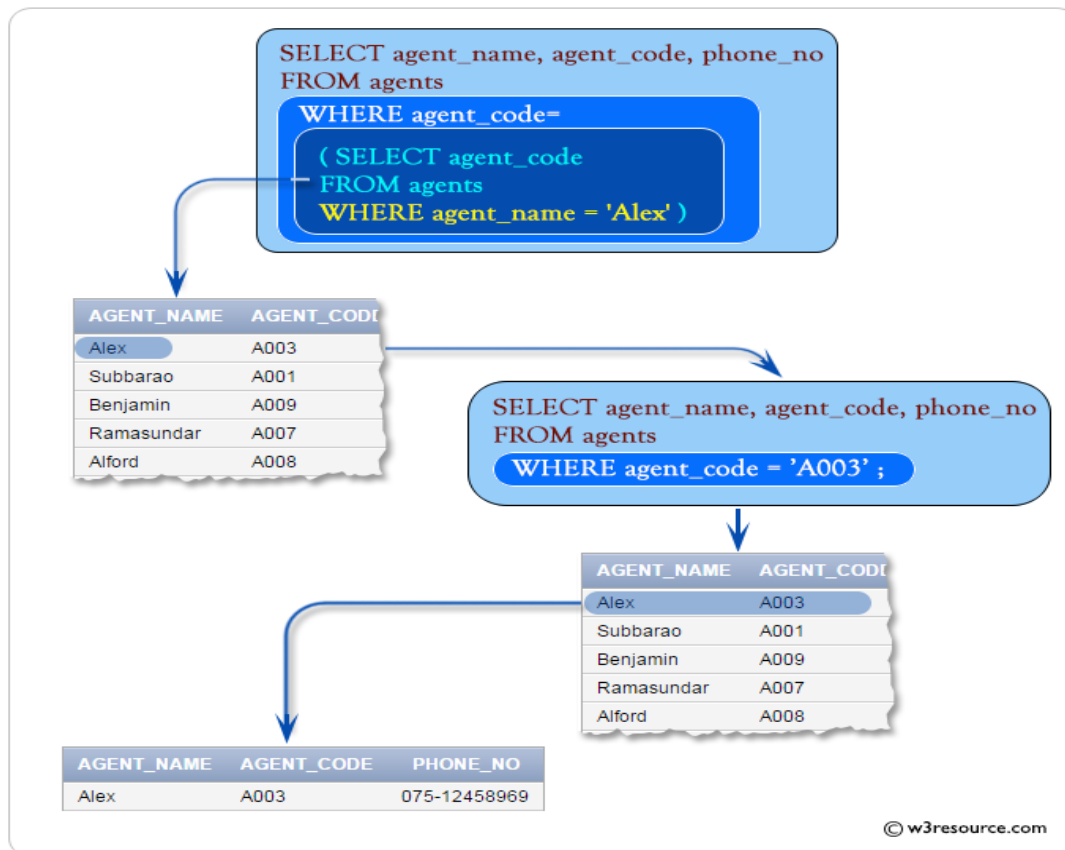
1. Single Row Subqueries

A single row subquery returns zero or one row to the outer SQL statement. You can place a subquery in a WHERE clause, a HAVING clause, or a FROM clause of a SELECT statement.

You can use = , > , < , >= , <= , <> , != , <=> comparison operators.

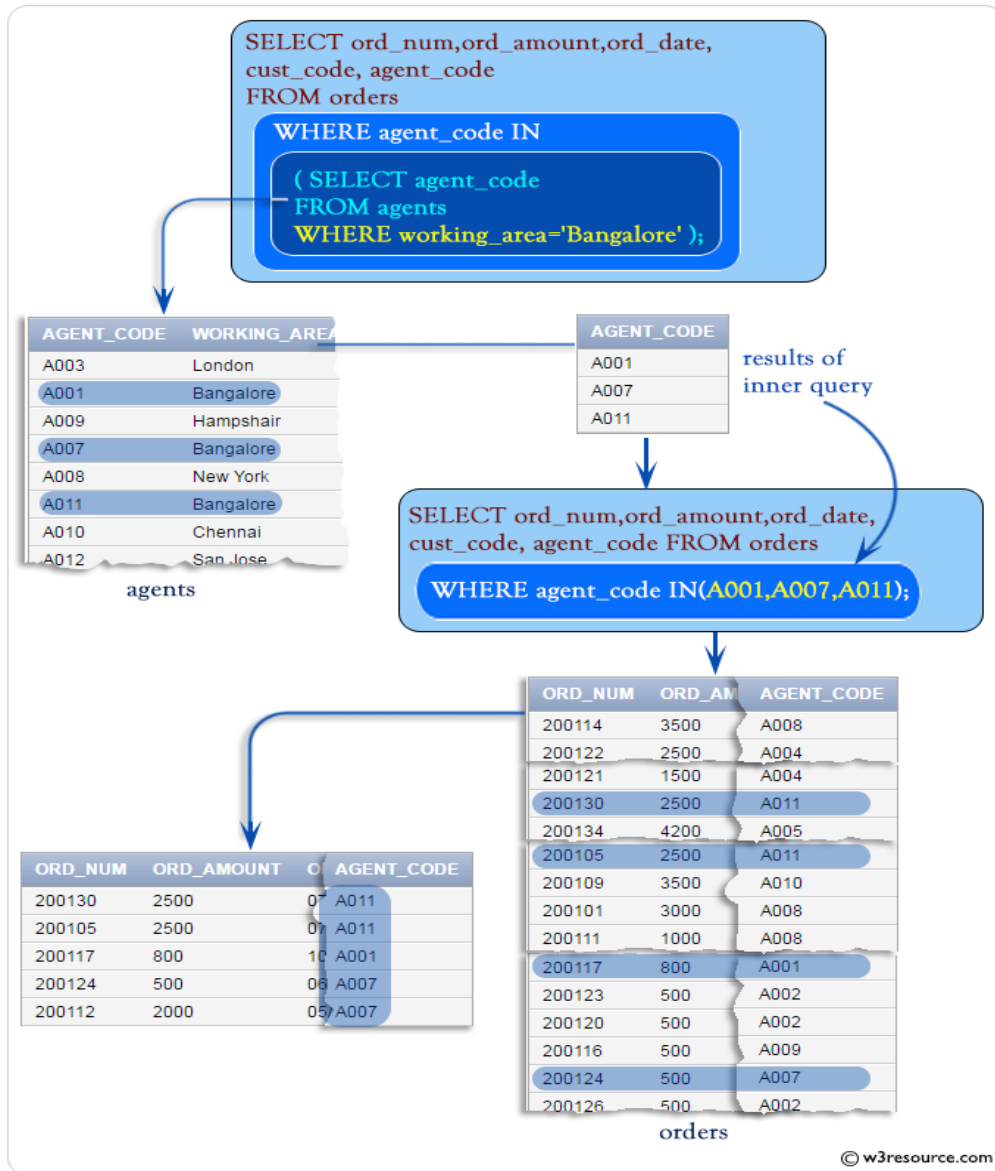
EXAMPLE: Retrieves the agent_name, agent_code, phone_no from the agents table whose agent_name is 'Alex'.

```
1 SELECT agent_name, agent_code, phone_no
2 FROM agents
3 WHERE agent_code =
4 (SELECT agent_code
5 FROM agents
6 WHERE agent_name = 'Alex');
```



2. Multiple Row Subqueries

Multiple row subquery returns one or more rows to the outer SQL statement. You may use the IN, ANY, or ALL operator in outer query to handle a subquery that returns multiple rows.



3. Multiple Column Subqueries

You can write subqueries that return multiple columns. The following example retrieves the order amount with the lowest price, group by agent code.


```
SELECT ord_num, agent_code, ord_date,
ord_amount FROM orders
```

```
WHERE (agent_code, ord_amount) IN
```

```
(SELECT agent_code,
MIN(ord_amount) FROM orders
GROUP BY agent_code);
```

ORD_NUM	ORD_AMOUNT	AGENT_CODE
200114	3500	A008
200122	2500	A004
200118	500	A006
200119	4000	A010
200121	1500	A004
200130	2500	A011
200134	4200	A005
200115	2000	A013
200108	4000	A004
200103	1500	A005
200105	2500	A011
200109	3500	A010

orders

results of
inner query

AGENT_CODE	MIN(ORD_AMOUNT)
A004	1500
A002	500
A007	500
A009	500
A011	2500
A012	900
A010	2000
A013	2000
A001	800
A008	1000

```
SELECT ord_num, agent_code, ord_date,
ord_amount FROM orders
```

```
WHERE (agent_code, ord_amount) IN ( results
from inner query );
```

ORD_NUM	AGENT_CODE	ORD_AMOUNT
200114	A008	3500
200122	A004	2500
200119	A010	4000
200121	A004	1500
200130	A011	2500
200134	A005	4200
200115	A013	2000
200105	A011	2500
200109	A010	3500
200101	A008	3000
200111	A008	1000
200104	A004	1500

ORD_NUM	AGENT_CODE	ORD_AMOUNT
200104	A004	1500
200121	A004	1500
200126	A002	500
200120	A002	500
200123	A002	500
200124	A007	500
200116	A009	500
200105	A011	2500
200130	A011	2500
200131	A012	900
200135	A010	2000
200115	A013	2000

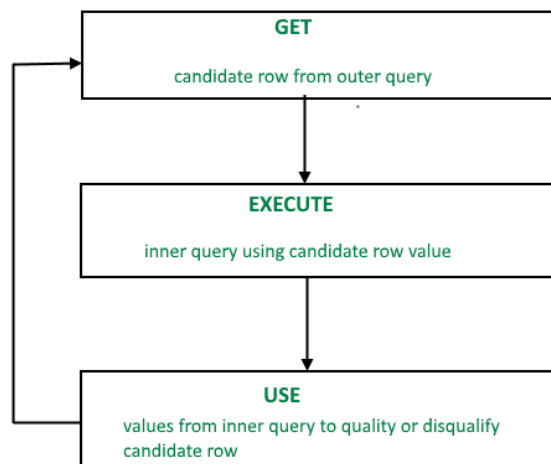
results

4. Correlated Subquery

Correlated subqueries are subqueries that reference one or more columns in the main query. Correlated subqueries depend on information in the main query to run, and thus, cannot be executed on their own.

Correlated subqueries are evaluated in SQL once per row of data retrieved – a process that takes a lot more computing power and time than a simple subquery.

Moreover, a correlated subquery is executed repeatedly, once for each row evaluated by the outer query. The correlated subquery is also known as a repeating subquery.



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

```
SELECT last_name, salary, department_id
FROM employees outer
WHERE salary >
      (SELECT AVG(salary)
       FROM employees
       WHERE department_id =
         outer.department_id);
```

5. Nested Subquery

Just like a simple subquery, a nested subquery's components can be executed independently of the outer query, while a correlated subquery requires both the outer and inner subquery to run and produce results.

Query execution starts from innermost query to outermost queries. The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query.

Example: Find Name, Salary, department_id of all employees who earn more than the average salary in their department and those employees were hired in between year 1990 to 2000.

```
SELECT last_name, salary, department_id
FROM employees e1
WHERE salary >
    (SELECT AVG(salary)
     FROM employees e2
     WHERE e2.department_id =
           e1.department_id AND e1.department_id in
           (SELECT department_id
            FROM employees
            Where hire_date between "1990-01-01" AND "2000-12-31"
           ));
```

Exist Operator:

The EXISTS operator tests for existence of rows in the results set of the subquery. If a subquery row value is found the condition is flagged TRUE and the search does not continue in the inner query, and if it is not found then the condition is flagged FALSE and the search continues in the inner query.

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

```
SELECT employee_id, last_name, job_id, department_id
FROM employees e1
WHERE EXISTS ( SELECT *
FROM employees e2
WHERE e2.manager_id = e1.employee_id);
```

Example: MySQL Subqueries with NOT EXISTS

NOT EXISTS subquery almost always contains correlations.

Find all departments (department_id, department_name) that do not have any employees.

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT * FROM employees WHERE department_id =
d.department_id);
```

Lab Task

1. Display the first name and salary for all employees who earn more than employee number 103.
2. Display all the information of employees who are working in Sales or IT department.
3. Write a query to display the first name and last name, salary, department id for those employees whose salary in average salary of any of departments.
4. Write a query to display the first name, last name and hiredate for all employees, who are working in the same department as an employee whose last name is Fox. Exclude Fox.
5. Display the employee first name, last name and employee id, for all employees whose department location is London.
6. Display the employee ID and Full name of all employees who works in same department where the employees having first name containing a letter 'Z'.
7. Find out the names of all employees whose salary is greater than 50% of their department's total salary bill.
8. Write a query to get the details of employees who are managers.
9. Display the employee id, name, salary, department name and city for all the employees who gets the salary as the salary earn by the employee which is maximum within the joining person January 1st, 1990 and December 31st, 1991.
10. Find all departments that do not have any employees.
11. Write a query in SQL to show the details of employees of job type ST_CLERK, SA-REP, AD_ASST whose working location is Seattle.
12. Find out the employees whose salaries are greater than the salaries of their managers.
13. List the highest paid employees working under DEN.
14. Display the detail information of departments which starting salary is at least 8000.
15. Display the full name of manager who is supervising 4 or more employees.