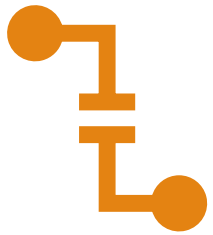
DATABASE SYSTEMS

FACULTY OF AI & MMG

SQL Joins

AGENDA



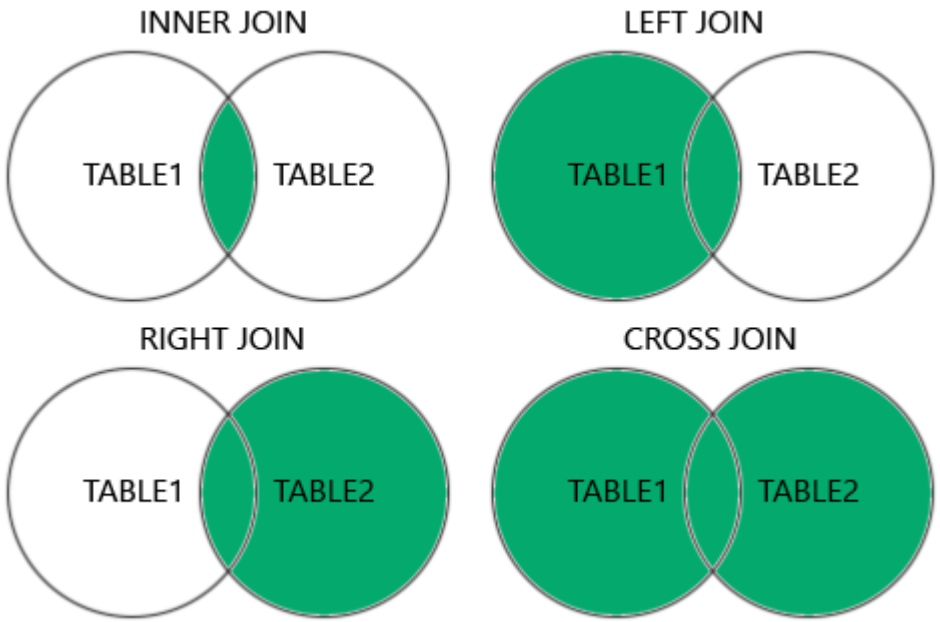
SQL Joins

Types of SQL Joins

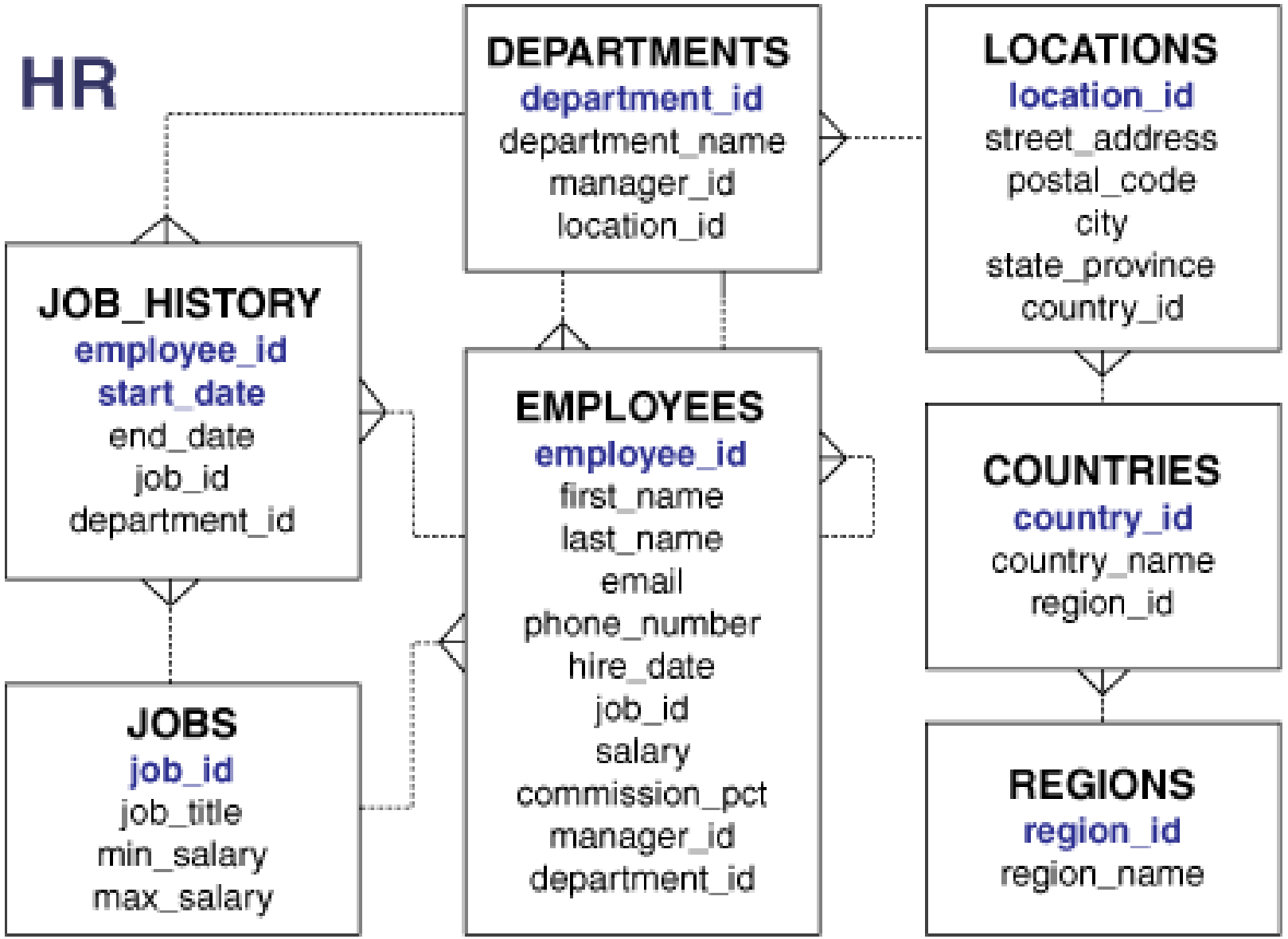
- Inner Join
- Left Join (Left Outer Join)
- Right Join (Right Outer Join)
- Cross Join

SQL Sub-quires

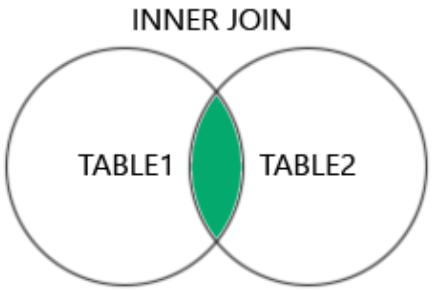
SQL Joins



- SQL joins are used to combine rows from two or more tables based on a related column between them.
- **Types of SQL Joins**
  1. **INNER JOIN** – Returns records that have matching values in both tables.
  2. **LEFT JOIN (LEFT OUTER JOIN)** – Returns all records from the left table and matching records from the right table. If there is no match, NULL values are returned for columns from the right table.
  3. **RIGHT JOIN (RIGHT OUTER JOIN)** – Returns all records from the right table and matching records from the left table. If there is no match, NULL values are returned for columns from the left table.
  4. **CROSS JOIN** – Returns a Cartesian product, meaning all possible combinations of rows between the tables.


HR
DEPARTMENTS
department_id
department_name
manager_id
location_id
LOCATIONS
location_id
street_address
postal_code
city
state_province
country_id
JOB_HISTORY
employee_id
start_date
end_date
job_id
department_id
EMPLOYEES
employee_id
first_name
last_name
email
phone_number
hire_date
job_id
salary
commission_pct
manager_id
department_id
COUNTRIES
country_id
country_name
region_id
JOBS
job_id
job_title
min_salary
max_salary
REGIONS
region_id
region_name

Inner Join



- The INNER JOIN keyword selects records that have matching values in both tables.
- Syntax:

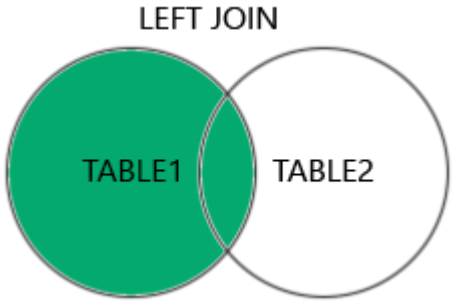
```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT e.employee_id, e.first_name, e.last_name, e.department_id, d.department_name
FROM employees e
INNER JOIN departments d ON e.department_id = d.department_id;
```

Result:

employee_id	first_name	last_name	department_id	department_name
100	Steven	King	90	Executive
101	Neena	Kochhar	90	Executive
102	Lex	De Haan	90	Executive

Left Outer Join



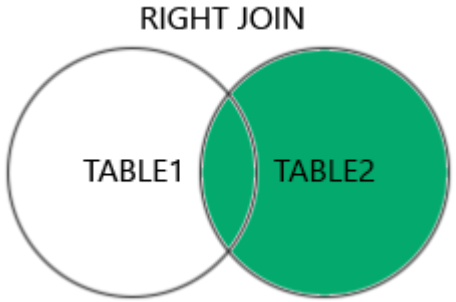
- The LEFT JOIN keyword returns all records from the left table (table1), and the matching records (if any) from the right table (table2).
- Syntax:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT e.employee_id, e.first_name, e.last_name, e.department_id, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

✅ **Result**: Includes employees without a department (i.e., NULL department_name values).

Right Outer Join



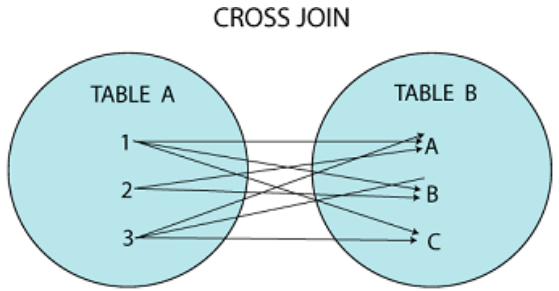
- The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records (if any) from the left table (table1).
- Syntax:

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT e.employee_id, e.first_name, e.last_name, e.department_id, d.department_name
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.department_id;
```

✅ **Result:** Includes departments without employees (NULL values for employee_id, first_name, last_name).

Cross Join



- The CROSS JOIN is also known as CARTESIAN JOIN, which provides the Cartesian product of all associated tables.
- Syntax:

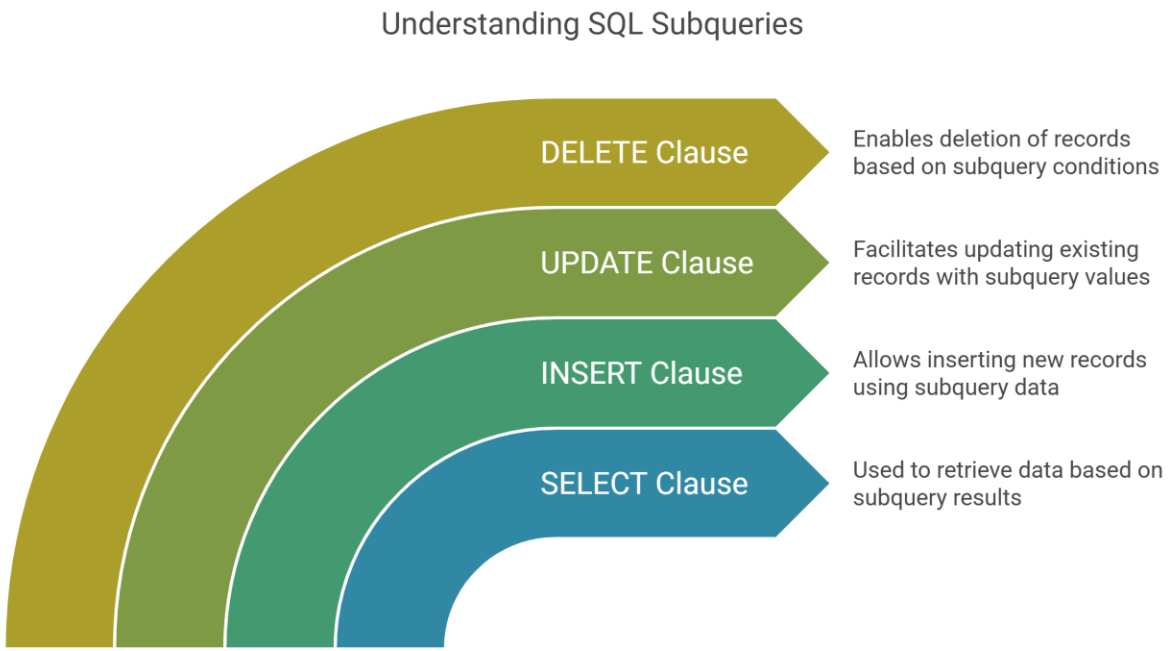
```
SELECT column_name(s)
FROM table1
CROSS JOIN table2;
```

```
SELECT e.employee_id, e.first_name, e.last_name, j.job_title
FROM employees e
CROSS JOIN jobs j;
```

✅ **Result:** Returns a Cartesian product, listing all possible employee-job combinations.

SUB-QUIRES

SQL Subquires



- A subquery (also called an inner query or nested query) is a query nested inside another SQL query.
- A subquery is a query embedded within another SQL query.
- It is used to retrieve data that will be used in the main query (also called the outer query).
- It is often used for:
  - Filtering (WHERE)
  - Aggregation (HAVING)
  - Data retrieval in SELECT, FROM, or WHERE clauses.

Syntax & Examples of Subquires

```
SELECT column1, column2, ...
FROM table1
WHERE column_name OPERATOR (SELECT column_name FROM table2 WHERE condition);
```

- The subquery is enclosed in parentheses `()`.
- The result of the subquery is used by the outer query.

```
SELECT employee_id, first_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

- The subquery `(SELECT AVG(salary) FROM employees)` calculates the average salary, and the outer query uses this value to filter employees.

1. Subquery in WHERE Clause
   - Find employees whose salary is greater than the average salary.

```
SELECT employee_id, first_name, salary,
       (SELECT AVG(salary) FROM employees) AS avg_salary
FROM employees;
```

- The subquery `(SELECT AVG(salary) FROM employees)` is executed once, and its result is displayed as a column in the output.

2. Subquery in SELECT Clause
   - Display employee details along with the average salary of all employees.

Syntax & Examples of Subquires

3. Subquery in FROM Clause
   - Use a subquery as a derived table (temporary table).

```sql
SELECT dept_id, AVG(salary) AS avg_salary
FROM (SELECT department_id AS dept_id, salary FROM employees) AS temp_table
GROUP BY dept_id;
```

- The subquery creates a temporary table (`temp_table`), and the outer query groups the data by `dept_id` .

4. Subquery with IN Operator
   - Find employees who work in departments located in a specific city.

```sql
SELECT employee_id, first_name, department_id
FROM employees
WHERE department_id IN (SELECT department_id FROM departments WHERE location_id = 1700);
```

- The subquery returns a list of department IDs located in `location_id = 1700` , and the outer query uses this list to filter employees.

Examples of Subquires

5. Correlated Subquery
   - Find employees whose salary is greater than the average salary of their department.

```sql
SELECT employee_id, first_name, salary, department_id
FROM employees e1
WHERE salary > (SELECT AVG(salary) FROM employees e2 WHERE e2.department_id = e1.department_id);
```

- The subquery depends on the outer query (`e1.department_id`) and is executed for each row in the outer query.

6. Subquery with EXISTS
   - Find departments that have at least one employee.

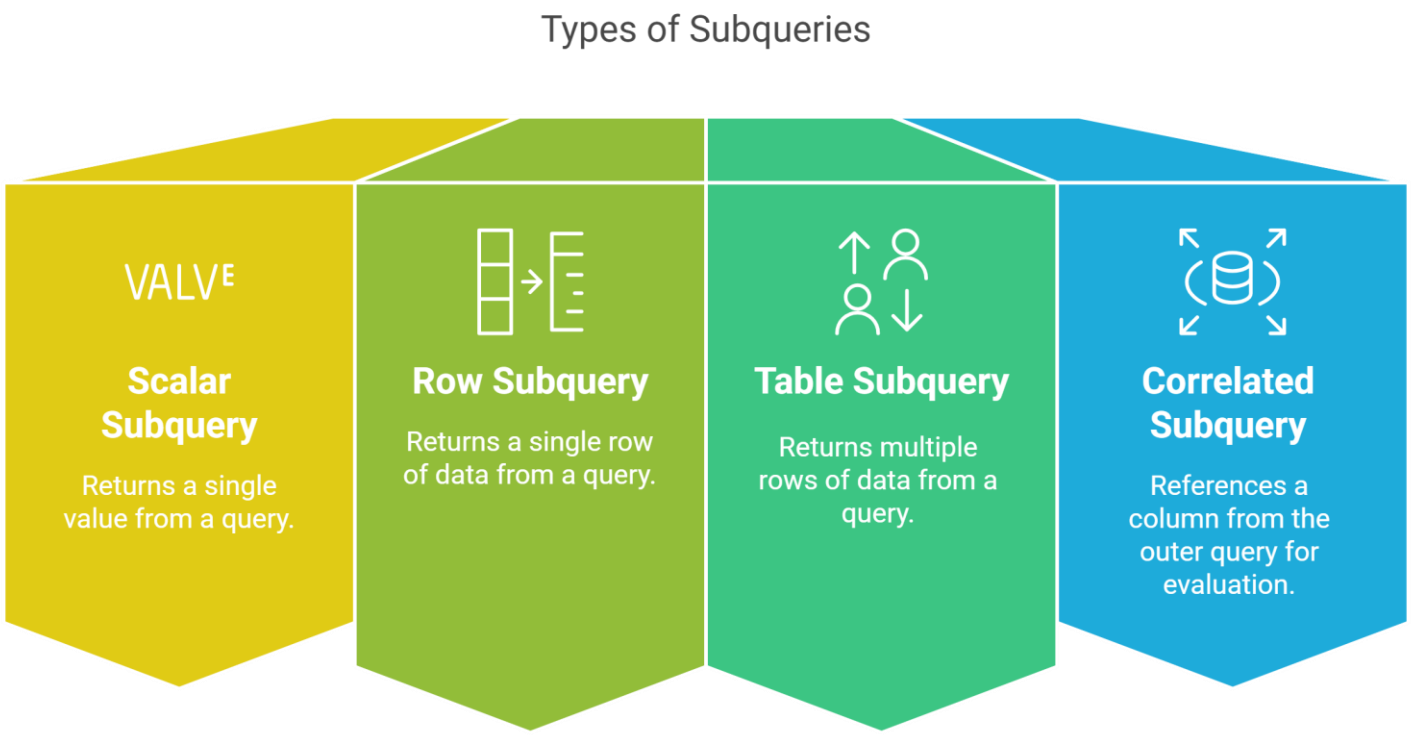
```sql
SELECT department_id, department_name
FROM departments d
WHERE EXISTS (SELECT 1 FROM employees e WHERE e.department_id = d.department_id);
```

- The subquery checks if there is at least one employee in each department, and the outer query returns the departments that satisfy this condition.

Use Cases

- Example Use Cases
- **Filtering Data:** Use subqueries in WHERE or HAVING clauses to filter data based on conditions.
- **Calculations:** Use subqueries in SELECT to calculate derived values.
- **Data Comparison:** Compare data between tables using subqueries.
- **Existence Checks:** Use EXISTS or NOT EXISTS to check for the existence of related data.

Types of Subquries



1. Scalar Subquery:

- Returns a single value (one row and one column).
- Can be used wherever a single value is expected (e.g., in SELECT, WHERE, HAVING).

2. Single or Multi Row Subquery:

- **Single row:**
- Returns one row (one value).
- Used with single-row comparison operators like =, >, <, >=, <=, <>.

- **Multi row:**
- Returns multiple rows.
- Used with operators like IN, ANY, ALL.

3. Correlated Subquery:

- A subquery that depends on the outer query for its values.
- Executed repeatedly, once for each row processed by the outer query.

1. Scalar Subquery

- Returns a single value (one row, one column).
- Usage: Can be used in SELECT, WHERE, HAVING.
- **Example: Find Employees Who Earn More Than the Average Salary**

```sql
SELECT first_name, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

✅ **How it works?**

- The **subquery** computes **one value** (average salary).
- The **outer query** selects employees earning more than this value.

✅ **Expected Output:**

first_name	last_name	salary
Steven	King	24000
Neena	Kochhar	17000

```sql
SELECT first_name, last_name, job_id, salary
FROM employees
WHERE (job_id, salary) = (
    SELECT job_id, salary FROM employees WHERE employee_id = 101
);
```

✅ **How it works?**

- The **subquery** fetches one row with **two columns** (job_id and salary) for employee **101**.
- The **outer query** selects employees with **the same job and salary**.

✅ **Expected Output:**

first_name	last_name	job_id	salary
Neena	Kochhar	AD_VP	17000
Lex	De Haan	AD_VP	17000

Find Employees Who Earn More Than Employee 101

```sql
SELECT first_name, last_name, salary
FROM employees
WHERE salary > (
    SELECT salary FROM employees WHERE employee_id = 101
);
```

2. Single / Multiple Rows Subquery

- **Single Row:**
- A Single-Row Subquery returns one row but may contain multiple columns.
- Usage: Used with single-row operators (=, >, <, >=, <=, <>).
- **Example: Find Employees with the Same Job and Salary as Employee 101**

```sql
SELECT first_name, last_name, department_id
FROM employees
WHERE department_id IN (
    SELECT department_id
    FROM employees
    GROUP BY department_id
    HAVING COUNT(*) > 5
);
```

✅ How it works?

- The **subquery** returns **multiple department IDs** where employee count > 5.
- The **outer query** selects employees who belong to those departments.

✅ Expected Output:

first_name	last_name	department_id
Neena	Kochhar	90
Lex	De Haan	90
Steven	King	90

2. Single / Multiple Row Subquery

- **Multiple row:**
- Returns multiple rows.
- Usage: Used with multi-row operators (IN, ANY, ALL).

- **Example: Find Employees Who Work in Departments With More Than 5 Employees**

Using ANY Operator

- Find Employees with Salary Greater Than At Least One Employee in Department 60

```sql
SELECT first_name, last_name, salary
FROM employees
WHERE salary > ANY (
    SELECT salary FROM employees WHERE department_id = 60
);
```

✅ **How it works?**

- The **subquery** retrieves all salaries in **department 60**.
- The **outer query** finds employees earning **more than at least one of them**.

✅ **Expected Output**: Employees earning more than the **lowest salary** in department 60.

Using ALL Operator

- Find Employees with Salary Higher Than All Employees in Department 30

```sql
SELECT first_name, last_name, salary
FROM employees
WHERE salary > ALL (
    SELECT salary FROM employees WHERE department_id = 30
);
```

✅ **How it works?**

- The **subquery** retrieves all salaries from **department 30**.
- The **outer query** selects employees who earn **more than the highest salary** in department 30.

✅ **Expected Output**: Employees earning more than the **highest salary** in department 30.

```sql
SELECT e1.first_name, e1.last_name, e1.salary, e1.department_id
FROM employees e1
WHERE salary = (
    SELECT MAX(e2.salary)
    FROM employees e2
    WHERE e2.department_id = e1.department_id
);
```

✅ **How it works?**

- The **subquery is executed once per row.**
- It retrieves the **highest salary in each employee's department.**
- The **outer query** selects employees whose salary matches the highest salary.

✅ **Expected Output:**

first_name	last_name	salary	department_id
Steven	King	24000	90
Neena	Kochhar	17000	90

3. Correlated Subquery

- A subquery that depends on the outer query for its values.
- Usage: The subquery is executed once for each row in the outer query.

- **Example: Find Employees Who Earn the Highest Salary in Their Department**