Types of JOINS in SQL

INNER JOIN

OUTER JOIN

- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

SELF JOIN

THETA JOIN

CROSS JOIN

NATURAL JOIN

INNER JOIN

- Definition
  - Combines rows from two tables only where the join condition is evaluated to true
- Syntax

```
SELECT col1, col2, .... FROM
table1 JOIN table2
ON table1.some_column = table2.some_column;
```

- Scenario
  - You want to find employees and their respective department names from two different tables **employee** and **department**

Assume that you have the following two tables

employee

emp_id	name	salary	dept_id
1	Alice	50000	101
2	Bob	45000	102
3	Charlie	60000	NULL
4	Diana	48000	101
5	Eve	70000	103
6	Frank	65000	105

department

dept_id	dept_name
101	HR
102	IT
103	Finance
104	Marketing

Writing INNER JOIN as follows will result in

```
SELECT e.emp_id, e.name,
d.dept_name
FROM
employee e
JOIN
department d
ON e.dept_id = d.dept_id;
```

emp_id	name	dept_name
1	Alice	HR
2	Bob	IT
4	Diana	HR
5	Eve	Finance

LEFT OUTER JOIN

- Definition
  - Retrieves all rows from the left table and matching rows from the right table. If there is no match, NULL is returned for the right table
- Syntax

  SELECT col1, col2, ... FROM
  table1 **LEFT JOIN** table2
  ON table1.some_column = table2.some_column;
- Scenario
  - You want to find all employees, including those who are not assigned to any department from two different tables **employee** and **department**

Assume that you have the following two tables

employee

emp_id	name	salary	dept_id
1	Alice	50000	101
2	Bob	45000	102
3	Charlie	60000	NULL
4	Diana	48000	101
5	Eve	70000	103
6	Frank	65000	105

department

dept_id	dept_name
101	HR
102	IT
103	Finance
104	Marketing

Writing LEFT OUTER JOIN as follows will result in

```
SELECT e.emp_id, e.name,
d.dept_id, d.dept_name
FROM
employee e
LEFT JOIN
department d
ON e.dept_id = d.dept_id;
```

emp_id	name	dept_id	dept_name
1	Alice	101	HR
2	Bob	102	IT
3	Charlie	NULL	NULL
4	Diana	101	HR
5	Eve	103	Finance
6	Frank	NULL	NULL

RIGHT OUTER JOIN

- Definition
  - Retrieves all rows from the right table and matching rows from the left table. If there is no match, NULL is returned for the left table
- Syntax

  SELECT col1, col2, ... FROM

  table1 **RIGHT JOIN** table2

  ON table1.some_column = table2.some_column;
- Scenario
  - You want to find all departments, including those with no employees from two different tables **employee** and **department**

Assume that you have the following two tables

employee

emp_id	name	salary	dept_id
1	Alice	50000	101
2	Bob	45000	102
3	Charlie	60000	NULL
4	Diana	48000	101
5	Eve	70000	103
6	Frank	65000	105

department

dept_id	dept_name
101	HR
102	IT
103	Finance
104	Marketing

Writing RIGHT OUTER JOIN as follows will result in

```
SELECT e.emp_id, e.name,
d.dept_id, d.dept_name
FROM
employee e
RIGHT JOIN
department d
ON e.dept_id = d.dept_id;
```

emp_id	name	dept_id	dept_name
1	Alice	101	HR
2	Bob	102	IT
4	Diana	101	HR
5	Eve	103	Finance
NULL	NULL	104	Marketing

FULL OUTER JOIN

- Definition
  - Combines the results of both left and right outer joins. All Rows from both tables are included, even if they don't match.
- Syntax

  SELECT col1, col2, ... FROM
  table1 **FULL OUTER JOIN** table2
  ON table1.some_column = table2.some_column;
- Scenario
  - You want to find all employees and departments, including **employees without departments** and **departments without employees** from two different tables **employee** and **department**

Assume that you have the following two tables

employee

emp_id	name	salary	dept_id
1	Alice	50000	101
2	Bob	45000	102
3	Charlie	60000	NULL
4	Diana	48000	101
5	Eve	70000	103
6	Frank	65000	105

department

dept_id	dept_name
101	HR
102	IT
103	Finance
104	Marketing

Writing FULL OUTER JOIN as follows will result in

```
SELECT e.emp_id, e.name,
d.dept_id, d.dept_name
FROM
employee e
FULL OUTER JOIN
department d
ON e.dept_id = d.dept_id;
```

emp_id	name	dept_id	dept_name
1	Alice	101	HR
2	Bob	102	IT
3	Charlie	NULL	NULL
4	Diana	101	HR
5	Eve	103	Finance
6	Frank	NULL	NULL
NULL	NULL	104	Marketing

SELF JOIN

- Definition
  - A table is joined with itself. Useful for hierarchical or parent-child relationships. SELF JOIN is written with normal JOIN keyword only, but instead of using two tables we use one table with different alias names
- Syntax

  SELECT col1, col2, ... FROM

  table alias_1 **JOIN** table alias_2

  ON alilas_1.some_column = alias_2.some_column;
- Scenario
  - You want to find all **professor** and their **hod** (assuming the **hod** is also a **professor**) from only one table **professor**.

Assume that you have the following table

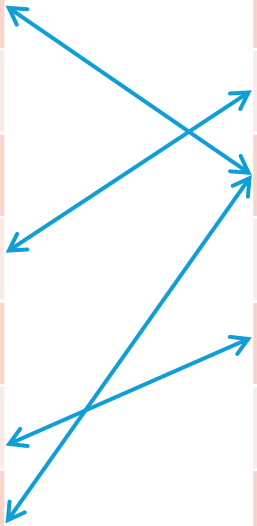
professor

prof_id	name	salary	hod_id
1	Alice	50000	3
2	Bob	45000	NULL
3	Charlie	60000	NULL
4	Diana	48000	2
5	Eve	70000	NULL
6	Frank	65000	5
7	Henry	55000	3

You need to think like combining the table with itself as shown below

professor (p1)

prof_id	name	salary	hod_id
1	Alice	50000	3
2	Bob	45000	NULL
3	Charlie	60000	NULL
4	Diana	48000	2
5	Eve	70000	NULL
6	Frank	65000	5
7	Henry	55000	3



professor (p2)

prof_id	name	salary	hod_id
1	Alice	50000	3
2	Bob	45000	NULL
3	Charlie	60000	NULL
4	Diana	48000	2
5	Eve	70000	NULL
6	Frank	65000	5
7	Henry	55000	3

Writing SELF JOIN (INNER) as follows will result in

```
SELECT
p1.prof_id AS professor_id,
p1.prof_name AS professor_name,
p2.prof_id AS hod_id,
p2.prof_name AS hod_name
FROM
professor p1
JOIN
professor p2
ON p1.hod_id = p2.prof_id
```

professor_id	professor_name	hod_id	hod_name
1	Alice	3	Charlie
4	Diana	2	Bob
6	Frank	5	Eve
7	Henry	3	Charlie

THETA JOIN

- Definition
  - A **THETA JOIN** is a type of join in SQL where the condition between the two tables uses a **comparison operator other than equality (=)**.
  - **It can involved other operators like (<, <=, >, >= , ….)**
- Syntax

  SELECT col1, col2, …. FROM

  table1 **JOIN** table2

  ON **<some_other_comparison>** on columns of two tables
- Scenario
  - Determining worker's pay_grade based on grading criteria.

Assume that you have the following two tables and we have to figure out into which **pay_grade each employee falls....?**

worker

worker_id	name	salary
1	Alice	24500
2	Bob	16900
3	Charlie	40000
4	Diana	35650
5	Eve	12000
6	Frank	29990
7	Henry	47670

payment

min_salary	max_salary	grade
40000	49999	A
30000	39999	B
20000	29999	C
10000	19999	D

Writing THETA JOIN as follows will result in

```
SELECT
w.worker_id, w.name, w.salary, p.grade
FROM
worker w JOIN payment p
ON
w.salary >= p.min_salary
AND
w.salary <= p.max_salary;
```

worker_id	name	salary	grade
1	Alice	24500	C
2	Bob	16900	D
3	Charlie	40000	A
4	Diana	35650	B
5	Eve	12000	C
6	Frank	29990	D
7	Henry	47670	A