MY SQL JOINS

MySQL JOINS :

1. INNER JOIN
2. LEFT JOIN (or LEFT OUTER JOIN)
3. RIGHT JOIN (or RIGHT OUTER JOIN)
4. FULL JOIN (or FULL OUTER JOIN)
5. CROSS JOIN
6. SELF JOIN
7. NATURAL JOIN

---

1. INNER JOIN :

Returns only the rows that have matching values in both tables (according to join condition is satisfied.)

=> Syntax:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Use Case: When you need data that exists in both tables.

=> Tables

1. Employees

EmpID	EmpName	DeptID
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103
5	Eva	101

2. Departments

DeptID	DeptName
101	HR
102	IT
104	Marketing

Example :

```
SELECT Employees.EmpName, Departments.DeptName
FROM Employees
INNER JOIN Departments
ON Employees.DeptID = Departments.DeptID;
```

Output:

EmpName	DeptName
Alice	HR
Bob	IT
Eva	HR

Explanation: Only employees with a matching DeptID in both tables are returned.

---

2. LEFT JOIN (or LEFT OUTER JOIN) :

Returns all rows from the left table and the matched rows from the right table. If there's no match, NULL values are returned for columns from the right table.

=> Syntax:

```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.column = table2.column;
```

Use Case: When you want to keep all records from the left table regardless of matching data in the right table.

=> Tables

1. Employees

EmpID	EmpName	DeptID
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103
5	Eva	101

2. Departments

DeptID	DeptName
101	HR
102	IT
104	Marketing

Example :

```
SELECT Employees.EmpName, Departments.DeptName
FROM Employees
LEFT JOIN Departments
ON Employees.DeptID = Departments.DeptID;
```

Output:

EmpName	DeptName
Alice	HR
Bob	IT
Charlie	NULL
David	NULL
Eva	HR

---

3. RIGHT JOIN (or RIGHT OUTER JOIN) :

Definition: Returns all rows from the right table and the matched rows from the left table. If there's no match, NULL values are returned for columns from the left table.

=> Syntax:

```
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.column = table2.column;
```

Use Case: When you want to keep all records from the right table regardless of matching data in the left table.

=> Tables

1. Employees

EmpID	EmpName	DeptID
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103
5	Eva	101

2. Departments

DeptID	DeptName
101	HR
102	IT
104	Marketing

Example :

```
SELECT Employees.EmpName, Departments.DeptName
FROM Employees
RIGHT JOIN Departments
ON Employees.DeptID = Departments.DeptID;
```

Output:

EmpName	DeptName
Alice	HR
Bob	IT
NULL	Marketing
Eva	HR

---

4. FULL JOIN :

Combines rows from both tables, returning all rows from both sides. Unmatched rows from either table result in NULL.

NOTE : MySQL doesn't support FULL JOIN directly, but you can achieve it using a UNION):

=> Syntax :

```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.column = table2.column
UNION
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.column = table2.column;
```

Use Case: When you want to include all records from both tables.

=> Tables

1. Employees

EmpID	EmpName	DeptID
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103
5	Eva	101

2. Departments

DeptID	DeptName
101	HR
102	IT
104	Marketing

Example :

```
SELECT Employees.EmpName, Departments.DeptName
FROM Employees
LEFT JOIN Departments
ON Employees.DeptID = Departments.DeptID
```

```
UNION
SELECT Employees.EmpName, Departments.DeptName
FROM Employees
RIGHT JOIN Departments
ON Employees.DeptID = Departments.DeptID;
```

Output:

EmpName	DeptName
Alice	HR
Bob	IT
Charlie	NULL
David	NULL
Eva	HR
NULL	Marketing

---

5. CROSS JOIN :

Returns the Cartesian product of both tables, meaning every row in the first table is combined with every row in the second table.

=> Syntax :

```
SELECT columns
FROM table1
CROSS JOIN table2;
```

Use Case: When you want every combination of rows from the two tables.

=> Tables

1. Employees

EmpID	EmpName	DeptID
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103
5	Eva	101

2. Departments

DeptID	DeptName
101	HR
102	IT
104	Marketing

Example :

```
SELECT Employees.EmpName, Departments.DeptName
FROM Employees
CROSS JOIN Departments;
```

Output:

EmpName	DeptName
Alice	HR
Alice	IT
Alice	Marketing
Bob	HR
Bob	IT
Bob	Marketing
Charlie	HR
Charlie	IT
Charlie	Marketing
David	HR
David	IT
David	Marketing
Eva	HR
Eva	IT
Eva	Marketing

Explanation: Every employee is combined with every department, resulting in the Cartesian product of the two tables.

---

6. SELF JOIN :

Joins a table with itself, used to compare rows within the same table.

=> Syntax:

```
SELECT a.columns, b.columns
FROM table a, table b
WHERE condition;
```

Use Case: When you want to compare rows within the same table or create a relationship within the same table.

=> Tables

1. Employees

EmpID	EmpName	DeptID
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103
5	Eva	101

Example :

```
SELECT e1.EmpName AS Employee, e2.EmpName AS
Manager
FROM Employees e1, Employees e2
WHERE e1.DeptID = e2.DeptID AND e1.EmpID != e2.EmpID;
```

Output:

Employee	Manager
Alice	Eva

7. NATURAL JOIN :

Automatically joins tables based on all columns with the same name and data type in both tables.

=> Syntax:

```
SELECT columns
FROM table1
NATURAL JOIN table2;
```

Use Case: When tables have columns with the same name and you want to join on all of those columns automatically.

NOTE :

=> NATURAL JOIN and INNER JOIN can produce similar outputs because both retrieve rows where there is a match between the tables.

=> Key Differences:
In an INNER JOIN, you explicitly specify the columns to join on,
whereas in a NATURAL JOIN, the database automatically determines the join condition based on column names.

Column Control: INNER JOIN offers more control, allowing you to join on specific columns even if other columns have the same name. NATURAL JOIN doesn't provide this flexibility.