

Video-14

Topics to cover:

- Joins in SQL

Joins

In MySQL, joins are used to combine rows from two or more tables based on related columns.

customer			
<i>customer_id</i>	<i>first_name</i>	<i>last_name</i>	<i>email_address</i>
1	Bob	Bobbinson	bob@email.com
2	Mark	Markson	mark@email.com
3	Jack	Jackson	jack@email.com
4	Jane	Janeson	jane@email.com

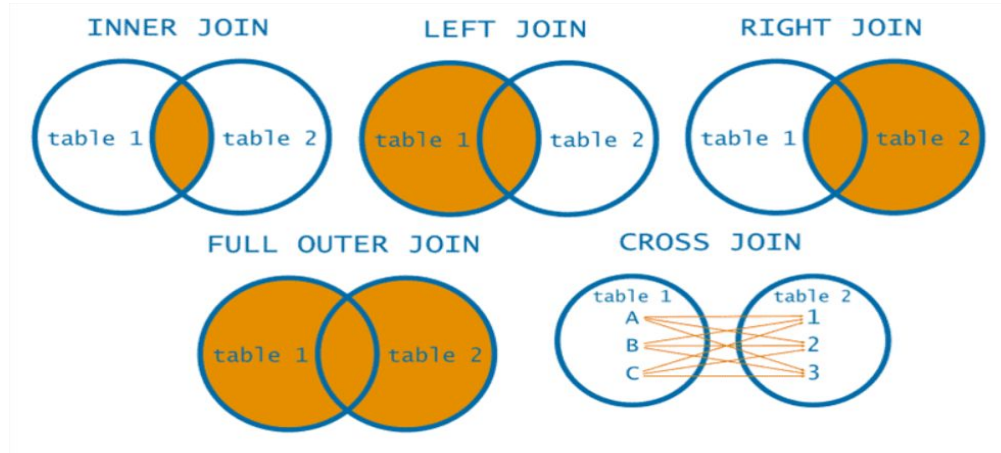
sale			
<i>sale_id</i>	<i>customer_id</i>	<i>date_of_sale</i>	<i>total_amount</i>
1	1	01.01.2022	50
2	1	05.01.2022	75
3	2	07.01.2022	20
4	3	01.02.2022	30
5	3	03.02.2022	90

Output Table

<i>first_name</i>	<i>last_name</i>	<i>date_of_sale</i>	<i>total_amount</i>
Bob	Bobbinson	01.01.2022	50
Bob	Bobbinson	05.01.2022	75
Mark	Markson	07.01.2022	20
Jack	Jackson	01.02.2022	30
Jack	Jackson	03.02.2022	90

Main Types of Joins in MySQL

1. INNER JOIN - Returns rows that have matching values in both tables.
2. LEFT JOIN - Returns all rows from the left table, and matching rows from the right table.
3. RIGHT JOIN - Returns all rows from the right table, and matching rows from the left table.
4. FULL JOIN - Returns all rows when there is a match in either left or right table.
5. CROSS JOIN - Returns the Cartesian product of both tables (every row from table A combined with every row from table B).
6. SELF JOIN - A table joined with itself.



Creating Department & Employees table to understand Join

-- Departments table

```
CREATE TABLE departments (  
    dept_id INT PRIMARY KEY,  
    dept_name VARCHAR(50) NOT NULL  
);
```

-- Employees table

```
CREATE TABLE employees (  
    emp_id INT PRIMARY KEY,  
    name VARCHAR(50) NOT NULL,  
    dept_id INT,  
    FOREIGN KEY (dept_id) REFERENCES departments(dept_id)  
);
```

```
INSERT INTO departments
```

```
VALUES
```

```
(1,'Human Resources'),  
(2,'Information Technology'),  
(3,'Finance'),  
(4,'Sales');
```

```
INSERT INTO employees
```

```
VALUES
```

```
(101,'Amit', 1), -- HR  
(102,'Priya', 2), -- IT  
(103,'Ravi', 3), -- Finance  
(104,'Sneha', NULL), -- No department  
(105,'Arjun', 2), -- IT  
(106,'Meera', 4); -- Sales
```

```
SELECT * FROM departments;
```

```
SELECT * FROM employees;
```

1. INNER JOIN: Returns rows where there is a match in both tables. Excludes non-matching rows.

Syntax:

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

```
SELECT employees.emp_id, employees.name, departments.dept_name  
FROM employees  
INNER JOIN departments  
ON employees.dept_id = departments.dept_id;
```

-- With Alias

```
SELECT e.emp_id, e.name, d.dept_name  
FROM employees e  
INNER JOIN departments d  
ON e.dept_id = d.dept_id;
```

2. LEFT JOIN: Returns all rows from the left table, and matched rows from the right table.

- Non-matching rows from the right table show as NULL.

Syntax:

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

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


3. RIGHT JOIN: Returns all rows from the right table, and matched rows from the left table.

- Non-matching rows from the left table show as NULL.

Syntax:

```
SELECT column_list
```

```
FROM table1
```

```
RIGHT JOIN table2
```

```
ON table1.column = table2.column;
```

```
SELECT e.emp_id, e.name, d.dept_name
```

```
FROM employees e
```

```
RIGHT JOIN departments d
```

```
ON e.dept_id = d.dept_id;
```

4. FULL JOIN: MySQL doesn't support FULL OUTER JOIN directly.

- We can emulate it using UNION of LEFT and RIGHT joins:

Syntax:

LEFT JOIN

UNION

RIGHT JOIN

```
SELECT e.emp_id, e.name, d.dept_name
```

```
FROM employees e
```

```
LEFT JOIN departments d
```

```
ON e.dept_id = d.dept_id;
```

```
UNION
```

```
SELECT e.emp_id, e.name, d.dept_name
```

```
FROM employees e
```

```
RIGHT JOIN departments d
```

```
ON e.dept_id = d.dept_id;
```