CIS 11051 – PRACTICAL FOR DATABASE DESIGN

MYSQL JOINS

MYSQL JOINS

- A JOIN in MySQL is used to **combine rows** from **two or more tables based on a related column between them**.
- Why use JOINS?
  - To retrieve meaningful data that is spread across multiple tables.
  - To follow the principles of database normalization
  - To maintain data consistency.
- You **don’t need to define primary key–foreign key relationships to use JOINs,** but having them improves data integrity and structure.

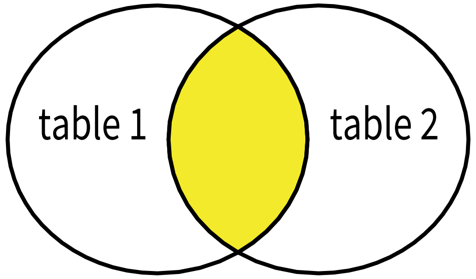
DIFFERENT TYPES OF JOINS IN MYSQL

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- CROSS JOIN

INNER JOIN

- The **INNER JOIN** keyword returns **only the rows that have matching values in both tables**.
- syntax:

```
SELECT Column_Name(s)
FROM table 1
INNER JOIN table 2
ON table1.column_name = table 2. column_name;
```



Example

- Employees Table:

EmployeeID	Name	DepartmentID
1	Alice	10
2	Bob	20
3	Charlie	30

- Department Table:

DepartmentID	DepartmentName
10	HR
20	Sales
40	IT

Inner Join Query & Result:

- Query:

```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID;
```

- Result:

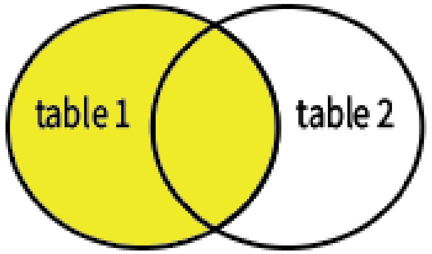
Name	DepartmentName
Alice	HR
Bob	Sales

- Note: Charlie is not included because there's no matching DepartmentID = 30 in the Departments table.

LEFT JOIN

- The **LEFT JOIN** in MySQL is used to combine rows from two tables. **It returns all rows from the left table (the first table), and the matching rows from the right table (the second table).** If there is **no match, the result will contain NULL values** for columns from the right table.
- syntax:

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



Example

- Customers Table:

customer_id	customer_name	country
1	James	USA
2	Maria	Canada
3	Leo	UK
4	Zoe	Australia

- Orders Table:

order_id	customer_id	order_date
101	1	2025-05-01
102	2	2025-05-02
103	1	2025-05-03

Left Join Query & Result:

- Query:

```
SELECT customers.customer_id, customers.customer_name, customers.country, orders.order_id, orders.order_date
FROM customers
LEFT JOIN orders
ON customers.customer_id = orders.customer_id;
```

- Result:

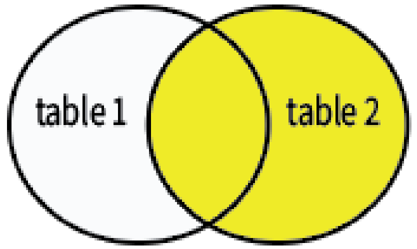
customer_id	customer_name	country	order_id	order_date
1	James	USA	101	2025-05-01
1	James	USA	103	2025-05-03
2	Maria	Canada	102	2025-05-02
3	Leo	UK	NULL	NULL
4	Zoe	Australia	NULL	NULL

- Note: Leo and Zoe have no matching orders, so the order_id and order_date are NULL.

RIGHT JOIN

- The **RIGHT JOIN** in MySQL is like the LEFT JOIN. **It returns all rows from the right table (the second table), and the matching rows from the left table (the first table)**. If there is **no match, the result will contain NULL values** for columns from the left table.
- **Syntax:**

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



Example

- Products Table:

product_id	product_name	category_id
1	Laptop	101
2	Smartphone	102
3	Tablet	103
4	Headphones	104

- Categories Table:

category_id	category_name
101	Electronics
102	Mobile Devices
103	Gadgets
105	Home Appliances

Right Join Query & Result:

- Query:

```
SELECT products.product_id, products.product_name, products.category_id, categories.category_name
FROM products
RIGHT JOIN categories
ON products.category_id = categories.category_id;
```

- Result:

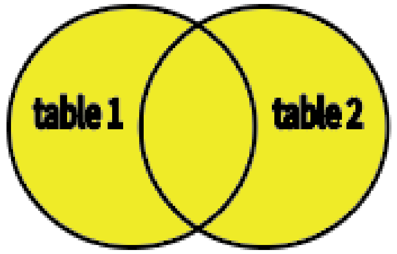
product_id	product_name	category_id	category_name
1	Laptop	101	Electronics
2	Smartphone	102	Mobile Devices
3	Tablet	103	Gadgets
NULL	NULL	105	Home Appliances

- Note: Home Appliances (category_id 105) has no matching product, so the product_id and product_name are NULL.

CROSS JOIN

- The **CROSS JOIN** in MySQL combines each row from the first table with every row from the second table. **It gives you all possible combinations of rows from both tables**.
- If table 1 has m rows and table 2 has n rows, the result will have **m** × **n** rows.
- There’s **no need for a condition** (ON clause).
- **Syntax:**

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



Example

- Colors Table:

color_id	color_name
1	Red
2	Blue

- Sizes Table:

size_id	size_label
1	Small
2	Medium
3	Large

Cross Join Query & Result:

- Query:

```
SELECT colors.color_id, colors.color_name, sizes.size_id, sizes.size_label
FROM colors
CROSS JOIN sizes;
```

- Result:

color_id	color_name	size_id	size_label
1	Red	1	Small
1	Red	2	Medium
1	Red	3	Large
2	Blue	1	Small
2	Blue	2	Medium
2	Blue	3	Large

- Note: 2 colors × 3 sizes = 6 total rows

Summary of SQL Joins

- **INNER JOIN**
  - Returns only rows with matching values in both tables
  - No NULLs unless the columns themselves contain NULL
- **LEFT JOIN**
  - Returns all rows from the left table, matched with right table rows
  - If there's no match, right table columns show NULL
- **RIGHT JOIN**
  - Returns all rows from the right table, matched with left table rows
  - If there's no match, left table columns show NULL
- **CROSS JOIN**
  - Returns all combinations of rows from both tables
  - Result set = total rows in table1 × total rows in table2
  - Can produce many duplicate-looking rows if not filtered later

Additional Key Points

- JOINs **do not automatically remove duplicates**. Use the DISTINCT keyword if needed.
- **NULLs in the result will occur** when a row in **one table has no matching row in the other table** (LEFT/RIGHT JOIN).
- You can use **MySQL aliases** to write **shorter versions** of code.
  - ✓ **Example:**

```
SELECT e.name, p.project_name
FROM Employees e
INNER JOIN Projects p ON e.emp_id = p.emp_id;
```

Code Breakdown

FROM Employees e : Use the **Employees table** and **refer to it using the letter e from now on in this query**.

- **e** is a shortcut for **Employees**
- **p** is a shortcut for **Projects**

THANK YOU !