ALL JOINs

INNER JOIN – Full Notes with Examples

🔷 **Definition:**

INNER JOIN returns only the rows that have **matching values in both tables** based on the join condition.

- 🔸 If there is **no match**, the row is **excluded** from the result.

📘 **Syntax:**

sql

CopyEdit

SELECT table1.column, table2.column

FROM table1

INNER JOIN table2

ON table1.common_column = table2.common_column;

🧪 **Example:**

Table 1: Students

StudentID	Name	DeptID
1	Riya	101
2	Aman	102
3	Sneha	NULL
4	Ravi	104

Table 2: Departments

DeptID	DeptName
101	Computer

DeptID	DeptName
102	Mechanical

✅ **Query:**

```
sql

CopyEdit

SELECT s.Name, d.DeptName

FROM Students s

INNER JOIN Departments d

ON s.DeptID = d.DeptID;
```

✅ **Output:**

Name	DeptName
Riya	Computer
Aman	Mechanical

🔍 **How It Works:**

- Matches where s.DeptID = d.DeptID
- Only rows where match exists in both tables are included
- Sneha (NULL) and Ravi (104) are excluded as they don't match any department

---

📌 **Points to Remember:**

Feature	INNER JOIN Behavior
Matching only	✅ Only returns rows with a match
NULL in join column	❌ Row skipped (NULL ≠ any value)
Most common JOIN	✅ Default for many queries

Feature	INNER JOIN Behavior
Used in MCQs	✅ Yes, often with NULL trap

LEFT JOIN (LEFT OUTER JOIN) – Full Notes with Examples

🔷 **Definition:**

LEFT JOIN returns:

✅ **All rows from the left table**,
✅ and the **matched rows** from the right table,
❌ If there's **no match**, the right side shows NULL.

Also called: **LEFT OUTER JOIN** (same meaning)

📘 **Syntax:**

sql

CopyEdit

SELECT table1.column, table2.column

FROM table1

LEFT JOIN table2

ON table1.common_column = table2.common_column;

🧪 **Example:**

Table 1: Students (Left Table)

StudentID	Name	DeptID
1	Riya	101
2	Aman	102
3	Sneha	NULL

StudentID	Name	DeptID
4	Ravi	104

Table 2: Departments (Right Table)

DeptID	DeptName
101	Computer
102	Mechanical

✅ **Query:**

```
sql
CopyEdit
SELECT s.Name, d.DeptName
FROM Students s
LEFT JOIN Departments d
ON s.DeptID = d.DeptID;
```

✅ **Output:**

Name	DeptName
Riya	Computer
Aman	Mechanical
Sneha	NULL
Ravi	NULL

🔍 **How It Works:**

- First takes **all rows from left (Students)**
- Then **tries to match** with right (Departments)
- If match found → shows dept name

- If not → DeptName = NULL

📌 **Points to Remember:**

Feature	LEFT JOIN Behavior
Keeps unmatched left rows	✅ Yes
NULL in join column (like Sneha)	✅ Still included, right = NULL
RIGHT table unmatched rows	❌ Not included
Output length	Same as or more than left table

✅ **LEFT JOIN vs INNER JOIN**

Concept	INNER JOIN	LEFT JOIN
Matching Rows	Only matched rows	All from left, + match
Unmatched Left	❌ Removed	✅ Kept (right = NULL)
Unmatched Right	❌ Removed	❌ Removed
Use Case	Need only valid pairs	Need full left info

RIGHT JOIN (RIGHT OUTER JOIN) – Full Notes with Examples

🔷 **Definition:**

RIGHT JOIN returns:

✅ **All rows from the right table**,
✅ and the **matched rows** from the left table,
❌ If there's **no match**, the left side shows NULL.

Also called: **RIGHT OUTER JOIN**

📘 **Syntax:**

```
sql

CopyEdit

SELECT table1.column, table2.column

FROM table1

RIGHT JOIN table2

ON table1.common_column = table2.common_column;
```

🧪 **Example:**

Table 1: Students (Left Table)

StudentID	Name	DeptID
1	Riya	101
2	Aman	102
3	Sneha	NULL

Table 2: Departments (Right Table)

DeptID	DeptName
101	Computer
102	Mechanical
103	Civil

✅ **Query:**

```
sql

CopyEdit

SELECT s.Name, d.DeptName

FROM Students s

RIGHT JOIN Departments d

ON s.DeptID = d.DeptID;
```

✅ **Output:**

Name	DeptName
Riya	Computer
Aman	Mechanical
NULL	Civil

🔍 **How It Works:**

- Keeps all rows from **Departments** (right table)
- Matches with Students if DeptID = DeptID
- Civil had no student → Name = NULL

📌 **Points to Remember:**

Feature	RIGHT JOIN Behavior
Keeps unmatched right rows	✅ Yes
Left table NULLs	✅ If no match, left = NULL
Use case	Keep all right-side reference info

✅ **LEFT vs RIGHT JOIN (Quick View)**

Feature	LEFT JOIN	RIGHT JOIN
All from...	Left table	Right table
Unmatched side	Right = NULL	Left = NULL
Direction	From left to right	From right to left
Swappable?	Yes, just **switch table positions**	

FULL OUTER JOIN – Full Notes with Examples

🔷 **Definition:**

A FULL OUTER JOIN returns:

✅ **All rows from both tables**,
✅ If a match is found → row is combined
❌ If no match → unmatched side is filled with **NULL**

📘 **Syntax:**

sql

CopyEdit

SELECT table1.column, table2.column

FROM table1

FULL OUTER JOIN table2

ON table1.common_column = table2.common_column;

⚠️ Note: Not supported in all databases (e.g., **MySQL doesn't support it directly**)
You can simulate it using UNION of LEFT + RIGHT joins if needed.

🧪 **Example:**

Table 1: Students

StudentID	Name	DeptID
1	Riya	101
2	Aman	102
3	Sneha	NULL
4	Ravi	105

Table 2: Departments

DeptID	DeptName
101	Computer
102	Mechanical
103	Civil

✅ **Query:**

sql

CopyEdit

```
SELECT s.Name, d.DeptName
FROM Students s
FULL OUTER JOIN Departments d
ON s.DeptID = d.DeptID;
```

✅ **Output:**

Name	DeptName
Riya	Computer
Aman	Mechanical
Sneha	NULL
Ravi	NULL
NULL	Civil

🔍 **How It Works:**

- Matches rows where DeptID exists in both tables
- Includes **all unmatched students** and **all unmatched departments**
- Fills unmatched columns with **NULL**

📌 **Points to Remember:**

Feature	FULL OUTER JOIN Behavior
Includes all rows	✅ From both left and right
Unmatched → NULL	✅ On whichever side is missing
Good for audit/reporting	✅ When you want everything
NULL join columns	❌ Still won't match

✅ **Simulating in MySQL:**

sql

CopyEdit

-- Not directly supported in MySQL

SELECT s.Name, d.DeptName

FROM Students s

LEFT JOIN Departments d ON s.DeptID = d.DeptID

UNION

SELECT s.Name, d.DeptName

FROM Students s

RIGHT JOIN Departments d ON s.DeptID = d.DeptID;

CROSS JOIN (Cartesian Product) – Full Notes with Examples

🔷 **Definition:**

A CROSS JOIN returns the **Cartesian product** of two tables:

✅ Every row from the first table is paired with **every row** from the second table.

🔸 It **does not require any join condition**.

📘 **Syntax:**

sql

CopyEdit

SELECT *

FROM table1

CROSS JOIN table2;

OR (in many DBMS):

sql

CopyEdit

SELECT *

FROM table1, table2;

⚠️ This older comma syntax behaves like a CROSS JOIN **only if no WHERE clause is used**.

🧪 **Example:**

Table 1: Colors

Color
Red
Blue

Table 2: Sizes

Size
Small
Medium
Large

✅ **Query:**

sql

CopyEdit

```
SELECT Color, Size
FROM Colors
CROSS JOIN Sizes;
```

✅ **Output:**

Color	Size
Red	Small
Red	Medium
Red	Large
Blue	Small
Blue	Medium
Blue	Large

✅ Total rows = **2 × 3 = 6**

📌 **Points to Remember:**

Feature	CROSS JOIN Behavior
Join condition	❌ Not used
Output size	RowsA × RowsB
Use case	To generate combinations (e.g., inventory, testing)
Performance risk	⚠️ Very large output with big tables
Confused with	❌ INNER JOIN if ON clause is missing

❗ **Important Exam/MCQ Trap:**

sql

CopyEdit

SELECT * FROM A, B;

🔸 If there's **no WHERE/ON clause**, it behaves like a **CROSS JOIN**.

✅ If there's a WHERE clause, it becomes **filtered** (not full Cartesian product).

SELF JOIN – Full Notes with Example

🔷 **Definition:**

A **SELF JOIN** is a join where:

✅ A **table is joined to itself** as if it were two separate tables.

📌 Used to compare rows **within the same table** — typically used for:

- Hierarchical relationships (manager-employee)
- Finding duplicates
- Comparing rows

📘 **Syntax:**

sql

CopyEdit

SELECT A.column, B.column

FROM table_name A

JOIN table_name B

ON A.common_column = B.common_column;

⚠️ **You must use aliases (A, B)** to distinguish the two copies of the same table.

🧪 **Example:**

Table: Employees

EmpID	Name	ManagerID
1	Riya	NULL
2	Aman	1
3	Sneha	1
4	Ravi	2

✅ **Goal:**

Show: Employee Name + Manager Name

✅ **Query:**

sql

CopyEdit

```
SELECT E.Name AS Employee, M.Name AS Manager
FROM Employees E
LEFT JOIN Employees M
ON E.ManagerID = M.EmpID;
```

✅ **Output:**

Employee	Manager
Riya	NULL
Aman	Riya
Sneha	Riya
Ravi	Aman

🔍 **How It Works:**

- We use the table **twice**:

    - E → Employee
    - M → Manager
- Join condition: E.ManagerID = M.EmpID

📌 **Points to Remember:**

Feature	SELF JOIN Behavior
Uses same table twice	✅ Required
Must use aliases	✅ To distinguish each role
Common in hierarchies	✅ e.g., employee–manager, category–parent
Can use INNER or LEFT	✅ LEFT JOIN lets you keep top-most entries (like CEO)