

ChatGPT ~

I am a new comer to mssql server and you are experieced teacher and who teaches from scratch gives comprehensive notes. so explain me from scratch all the joins concepts from basics 1. self 2. right 3. left 4.Inner 5. Cartesion join . what are they ?why used ? use emp and dept example . Also include driven and driver table while explaining left and right joins and whenever requiered.

Alright! Let's go step by step and learn **SQL Joins** in **MS SQL Server** from scratch with clear explanations and examples using emp (Employee) and dept (Department) tables.

 $\downarrow$ 

## 1. What is a JOIN in SOL?

A JOIN is used to combine rows from two or more tables based on a related column between them. It helps retrieve meaningful data from multiple tables in a database.

 $\downarrow$ 

#### Why use Joins?

- To avoid data redundancy by splitting related data into multiple tables.
- To fetch meaningful data by combining information from multiple tables.
- To perform data analysis efficiently.

#### 2. Sample Tables for Examples

We will use Employee (emp) and Department (dept) tables.

## Employee Table (emp)

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

#### Department Table (dept)

dept_id	dept_name	
10	HR $\psi$	
2	Bob	20
3	Charlie	30
3		
4	David	NULL
5	Eve	10

## Department Table ( dept )

dept_id		dept_name	
10		HR $\psi$	
2	Bob		20
3	Charlie		30
4	David		NULL
5	Eve		10

## Department Table ( dept )

dept_id	dept_name	
10	HR ↓	
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

## **Department Table (** dept )

dept_id		dept_name	
10		HR $\psi$	
2	Bob		20
3	Charlie		30
4	David		NULL
5	Eve		10

## Department Table ( dept )

dept_name		
HR •	HR •	
Bob	20	
	30	
	NULL	
	10	
	4	

## Department Table ( dept )

dept_id	dept_name
10	HR $\Psi$

#### Result

emp_id	emp_name	dept_name
1	Alice	HR
2	Bob	IT
3	Charlie	Sales
4	David	NULL
5	Eve	HR

#### **Driver and Driven Table**

- Left Table ( emp ) → Driver table
- Right Table ( dept )  $\rightarrow$  Driven table

#### Explanation:

- $\bullet \quad \text{Employee} \quad \text{David} \quad \text{is included even though he has } \textbf{no department} \; (\; \texttt{NULL} \quad \text{in} \quad \texttt{dept\_name} \; ).$
- Department Finance is **not included** since it has no employees.



## 3. RIGHT JOIN (or RIGHT OUTER JOIN)

- ullet Retrieves all records from the right table ( dept ) and matching records from the left table ( emp ).
- If no match is found, NULL is returned for columns from the left table.

#### Query

#### Result

emp_id	emp_name	dept_name
1	Alice	HR
2	Bob	IT
3	Charlie	Sales
5	Eve	HR
NULL	NULL	Finance

#### **Driver and Driven Table**

- Right Table ( dept )  $\rightarrow$  Driver table
- Left Table ( emp )  $\rightarrow$  Driven table

#### Explanation:

- Department Finance is included even though it has no employees (NULL in emp\_name).
- Employee David is not included since his dept\_id is NULL.

#### 4. SELF JOIN

- A **SELF JOIN** is when a table joins itself.
- Used to compare rows within the same table.

#### **Use Case**

Suppose we have an emp table where employees have a manager\_id referring to another employee in the same table

## **Modified Employee Table**

emp_id	emp_name	manager_id
1	Alice	NULL
2	Bob	1
3	Charlie	1
4	David	2
5	Eve	3

#### Query



#### Result

Employee	Manager
Alice	NULL
Bob	Alice
Charlie	Alice



#### Explanation:

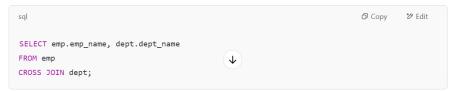
- Alice has **no manager** ( NULL ).
- Bob and Charlie report to Alice.
- David reports to Bob.
- Eve reports to Charlie.

## $(\downarrow)$

## 5. CROSS JOIN (Cartesian Join)

- Produces all possible combinations of rows from both tables.
- No need for a JOIN condition.

#### Query



#### Result

emp_name	dept_name
Alice	HR
Alice	IT
Alice	Sales
Alice	Finance
Bob	→ dR
Bob	IT
Bob	Sales
Bob	Finance

#### **Explanation:**

- If emp has 5 employees and dept has 4 departments, the result will have  $5 \times 4 = 20$  rows.
- This is mostly used in testing or when generating all possible combinations.  $(\cup \psi)$



Join Type	What it Does
INNER JOIN	Returns only matching records from both tables
LEFT JOIN	Returns all records from the left table and matching ones from the right
RIGHT JOIN	Returns all records from the right t. and matching ones from the left
SELF JOIN	Joins a table with itself
CROSS JOIN	Returns all possible combinations of rows from both tables

## **Final Thoughts**

- Use INNER JOIN when you want only matching data.
- Use LEFT JOIN when you need all records from the left table and matching ones from the right.
- Use RIGHT JOIN when you need all records from the right table and matching ones from the left.
- Use SELF JOIN for hierarchical or recursive relationships.
- Use CROSS JOIN when you need all possible row combinations.

Let me know if you need more clarifications! 🚀



Good . now explain me 1. Joining 3 or more tables 2.use of subqueries for joins 3. set operators

Glad you found the explanation useful! Now, let's move forward with:

- 1. Joining Three or More Tables
- 2. Using Subqueries for Joins
- 3. Set Operators



I'll explain each in detail with examples.

# 1. Joining Three or More Tables

When working with databases, you often need to fetch data from more than two tables. The concept of joins remains the same—you simply extend the join conditions to more tables.

## **Example: Employees, Departments, and Projects**

Let's assume we have the following three tables:

#### Employee Table (emp)

emp_id	emp_name	dept_id
1	Alice	10
2	Bob	20
3	Charlie	30
4	David	NULL
5	Eve	10

#### Department Table ( dept )

dept_id	dept_name
10	HR
20	ІТ
30	Sales
40	Finance

#### Project Table (project)

project_id	project_name	emp_id
101	Alpha	1
102	Beta	2
103	Gamma	3
104	Delta	5

105 Omega V

## Joining Three Tables: emp, dept, project

Let's say we want to retrieve employee names, their department names, and the projects they are working on.

#### Query:

```
sql

SELECT emp.emp_name, dept.dept_name, project.project_name

FROM emp

LEFT JOIN dept ON emp.dept_id = dept.dept_id

LEFT JOIN project ON emp.emp_id = project.emp_id;
```

#### **Result:**

emp_name	dept_name	project_name
Alice	HR ↓	Alpha
Bob	IT	Beta
Charlie	Sales	Gamma
David	NULL	NULL
Eve	HR	Delta

#### **Explanation:**

- LEFT JOIN ensures that even employees without projects ( David ) or departments ( Finance ) are included.
- Employees with multiple projects will have multiple rows.

# 2. Using Subqueries for Joins

A **subquery** is a query inside another query. Instead of using **JOINS**, you can sometimes use **subqueries** to fetch data.

## Example 1: Subquery Instead of INNER الد

If we want **employees and their department names**, we can use either a join or a subquery.

#### Using INNER JOIN:

```
sql

SELECT emp.emp_name, dept.dept_name

FROM emp

INNER JOIN dept ON emp.dept_id = dept.dept_id;
```

#### Using SUBQUERY:



#### **Result:**



Alice	TIK
Bob	IΤ
Charlie	Sales
David	NULL
Eve	HR

#### When to Use a Subquery Instead of a Join?

- If you need **only one column** from the joined table.
- If the join is not complex (subqueries can be s for large datasets).
- If you don't need to fetch multiple columns from the related table.

## 3. SQL Set Operators

Set operators combine the results of two or more queries. Unlike joins (which merge columns), set operators combine **rows**.

 $\downarrow$ 

## **Types of Set Operators**

Operator	Description
UNION	Combines results from two queries and removes duplicates.
UNION ALL	Combines results from two queries and keeps duplicates.
INTERSECT	Returns only the <b>common rows</b> between two queries.
EXCEPT	Returns rows from the <b>first query</b> that are not in the second query.

 $\downarrow$ 



Assume we have another table storing employees who have attended training programs.

#### Training Table (training)

emp_name	training_completed
Alice	Yes
Bob	No $igspace$
Charlie	Yes
David	Yes

## 1. UNION (Removes Duplicates)

Fetch a list of all employees and training attendees (without duplicates).



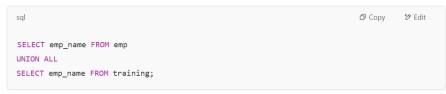
#### **Result:**

emp_name		
Alice		
Bob	<b>\(\psi\)</b>	
Charlie		

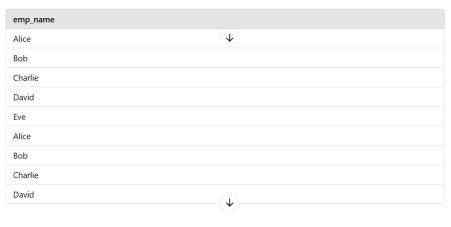


## 2. UNION ALL (Keeps Duplicates)

If we want duplicates to be included, we use  $UNION \stackrel{\checkmark}{\downarrow}$ .

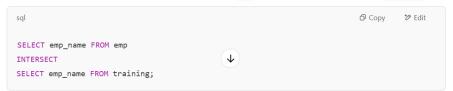


## Result (With Duplicates):



## 3. INTERSECT (Common Records)

Find employees who are both in the employee table (  $_{\text{emp}}$  ) and have attended training ( training ).

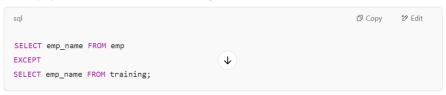


#### Result:



## 4. EXCEPT (Records Present in First Query but Not in Second)

Find employees who have NOT attended training.



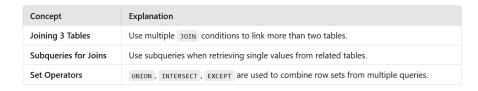


#### **Result:**



 $\downarrow$ 

# **Final Summary**



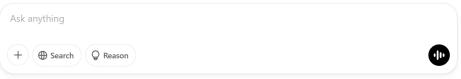
 $\downarrow$ 

#### Which One to Use?

- Use Joins when merging columns from multiple tables.
- Use **Subqueries** when extracting **single values** from related tables.
- Use **Set Operators** when merging **rows** from multiple queries.

## Hope this clears everything! Let me know if you have any doubts! 🖋





 ${\it ChatGPT\ can\ make\ mistakes.\ Check\ important\ info.}$