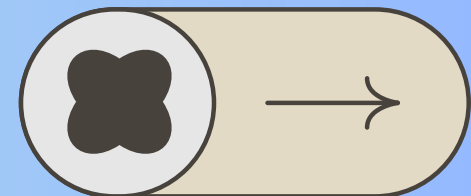


# SQL Joins Are Really Easy & Fun

Ganesh R

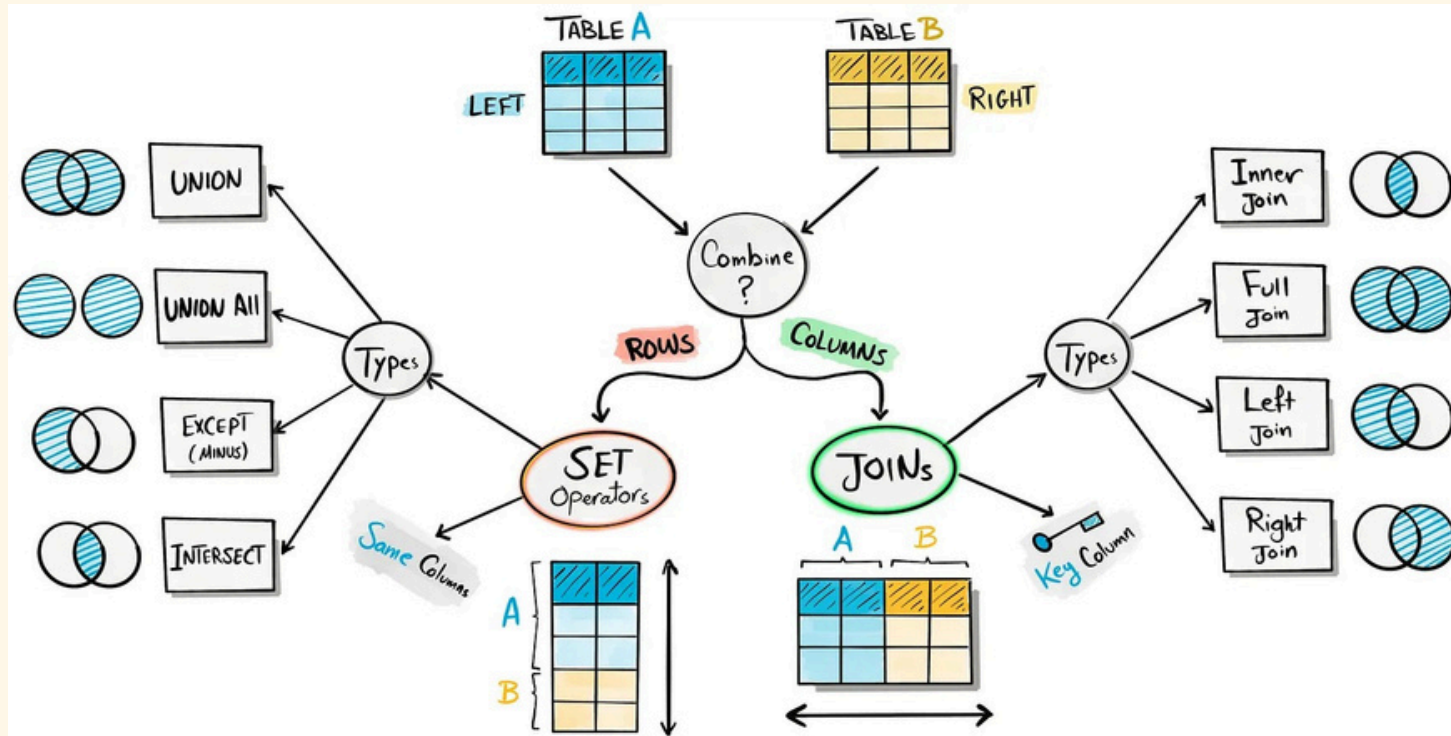
Azure Data Engineer





**Happy Friday!**

Before you head into the weekend, let's talk about something... SQL JOINS !! Writing SQL is always fun—especially when it comes to **joining tables**. But let's be honest, **many people struggle with it**. That's why I thought, **let's make a full video** covering **everything I know** about joins. I'll show you **my point of view**, breaking it down in a way that makes sense. Joins are **the core of SQL**, the **heart of how data connects**. Understanding them is a must if you want to write efficient queries. So, let's go through **all the techniques** you need to know.



Imagine you have two tables, **Table A** and **Table B**. At some point, you'll need to bring their data together. But here's the big question: **Do you want to combine rows or columns?**

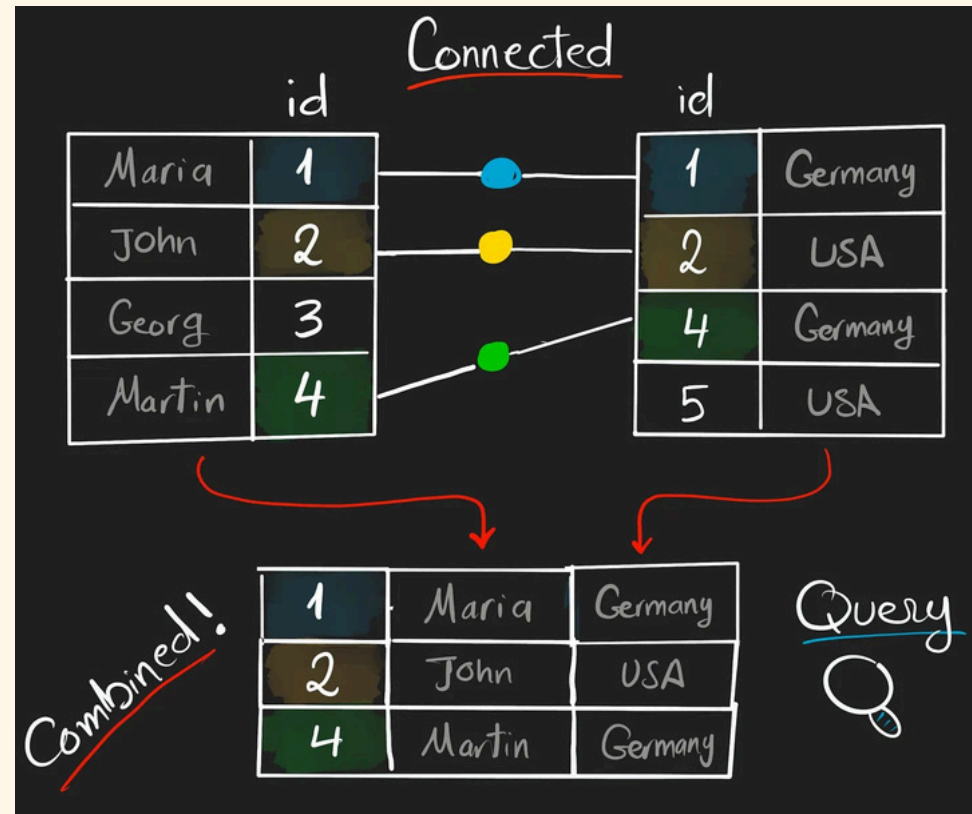
This is where SQL gives you two powerful tools:

**Set Operators** (to stack rows together)

**Joins** (to merge columns based on matching values)



## What is SQL Join?



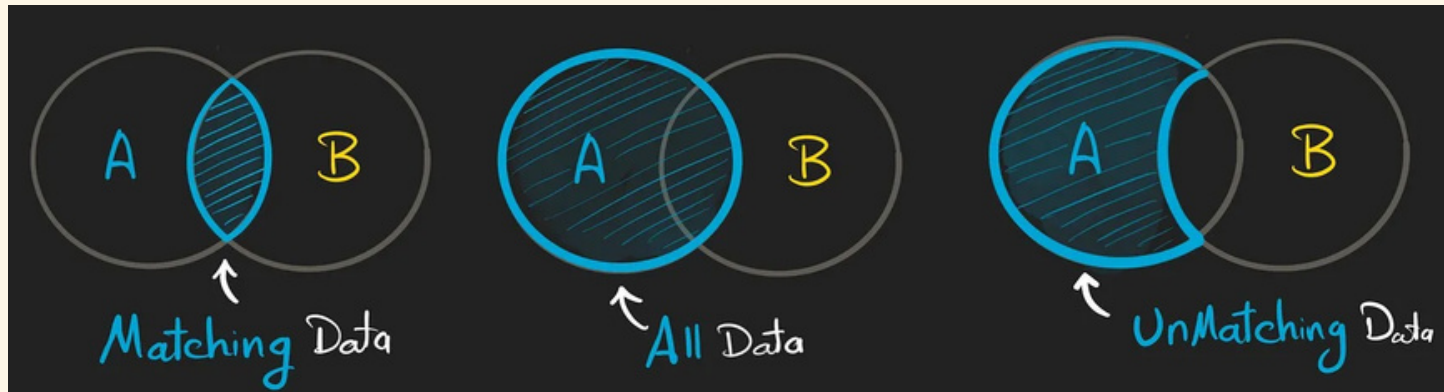
An **SQL JOIN** connects rows from **two tables** based on a common column. If you want to query from **two tables** into **one result**, you first need to **connect** them.

Tables store related data separately, but SQL lets you combine them using a JOIN.



## The Three Possibilities When Joining Tables

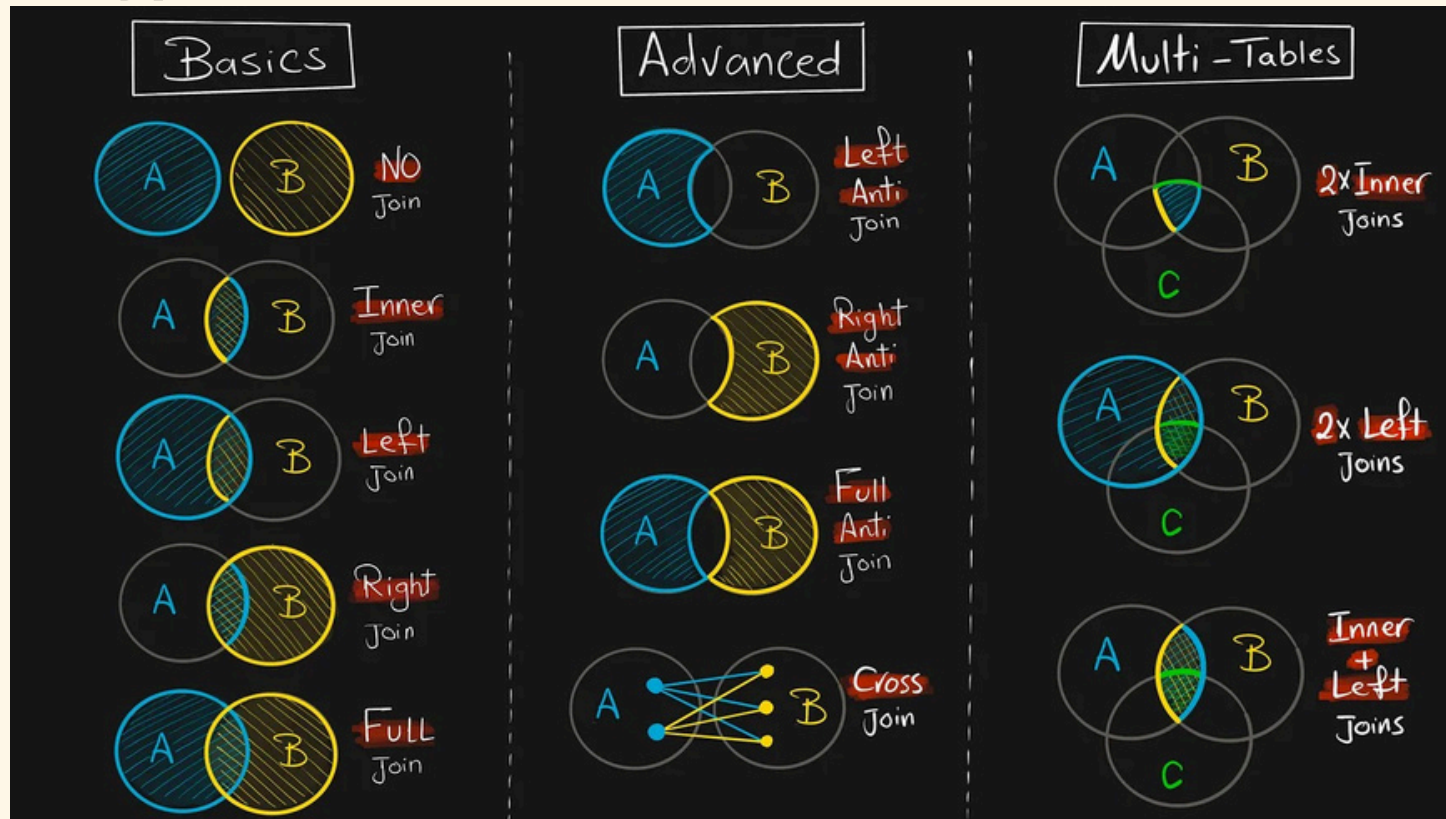
When joining tables, your result set depends on **which data you want to keep**. There are three main possibilities:



- Matching Data (Intersection): You only keep rows where both tables have a match.
- All Data from One Table: You keep all rows from one table, whether they match the second table or not.
- Unmatching Data (Exclusion): You keep only the rows that don't have a match in the second table.



## Join Types



SQL provides different **JOIN types** depending on how you want to combine data from tables.



## Basic Joins

- **Inner Join** → Returns only matching rows.
- **Left Join** → Returns all rows from Table A and matching rows from B.
- **Right Join** → Returns all rows from Table B and matching rows from A.
- **Full Join** → Returns all rows from both tables.

## Advanced Joins

- **Left Anti Join** → Rows in A **not** in B.
- **Right Anti Join** → Rows in B **not** in A.
- **Full Anti Join** → Rows that don't match in either table.
- **Cross Join** → Combines every row from A with every row from B.

## Multi-Table Joins

- **Multiple Inner Joins** → Joining more than two tables with matches.
- **Multiple Left Joins** → Keeping all rows from one main table.
- **Inner + Left Joins** → Mixing join types for complex queries.

Each JOIN type serves a different purpose, helping you get the right data for your queries!

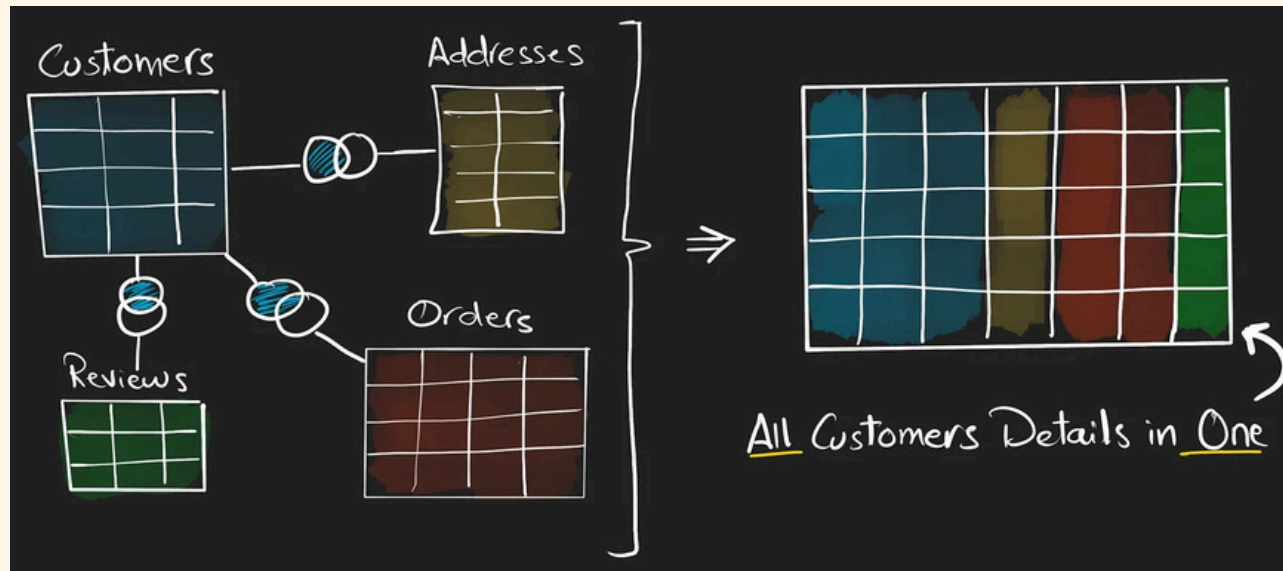




## Why We Join Tables?

Joins aren't just about merging tables—they serve different purposes based on what you need.

### 1- Recombine Data (Big Picture)



Data is often stored across multiple tables to keep it organized. But when you need a **full view**, you have to bring them back together.

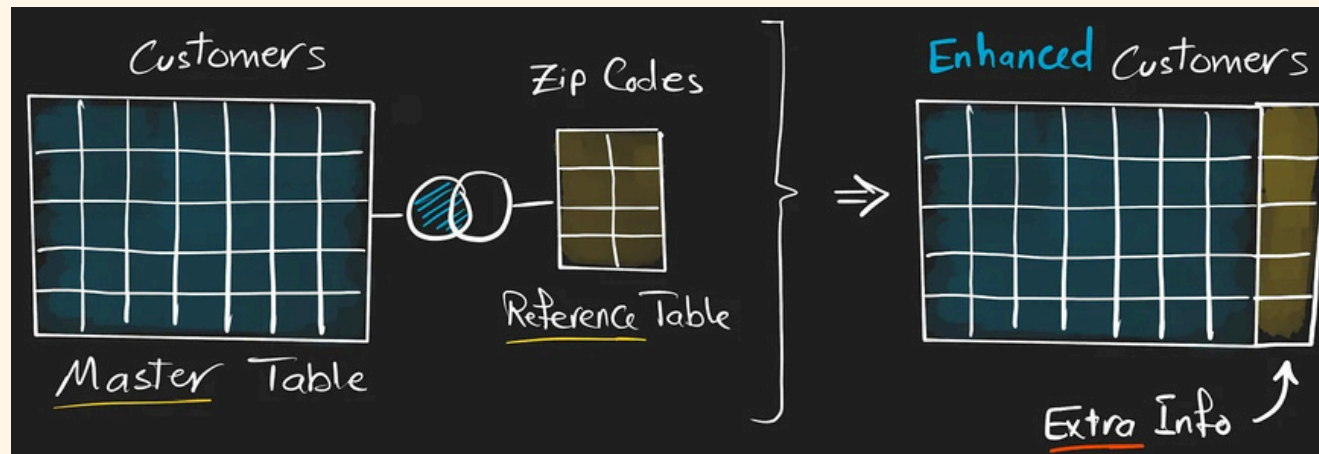
Example: A company stores employee details in one table and salaries in another. To generate a full employee report, you need to connect them.





Use **INNER**, **LEFT**, or **FULL JOIN** to merge related data.

## 2- Data Enrichment (Extra Info)

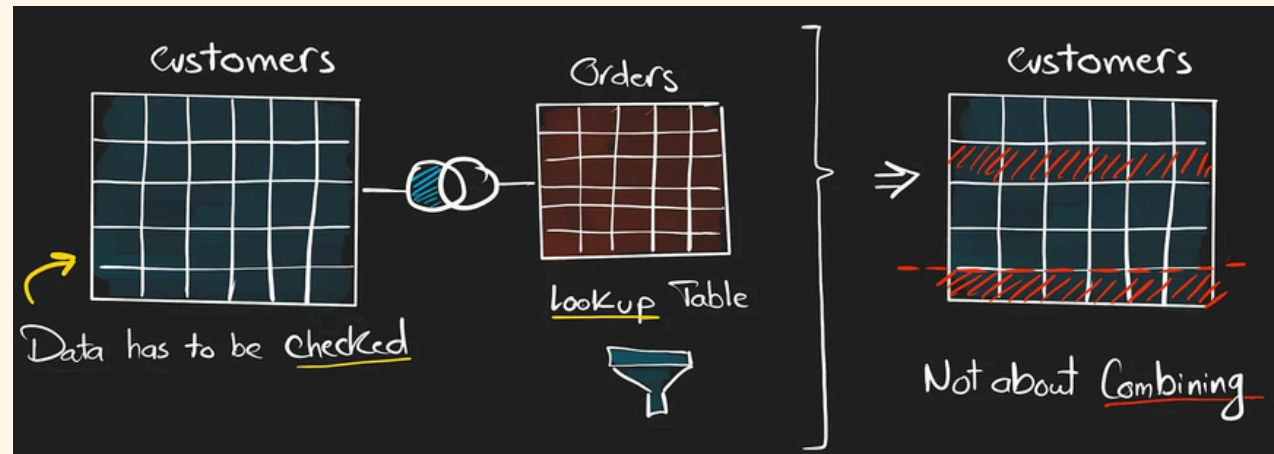


Sometimes, you already have your main dataset, but you need to **add extra details** from another table.

- Example: You have a list of customers and want to include their latest purchase details. Instead of duplicating data, you just link their purchase records when needed. A LEFT JOIN ensures you keep all records from the main table while pulling extra information.



### 3- Check Existence (Filtering)

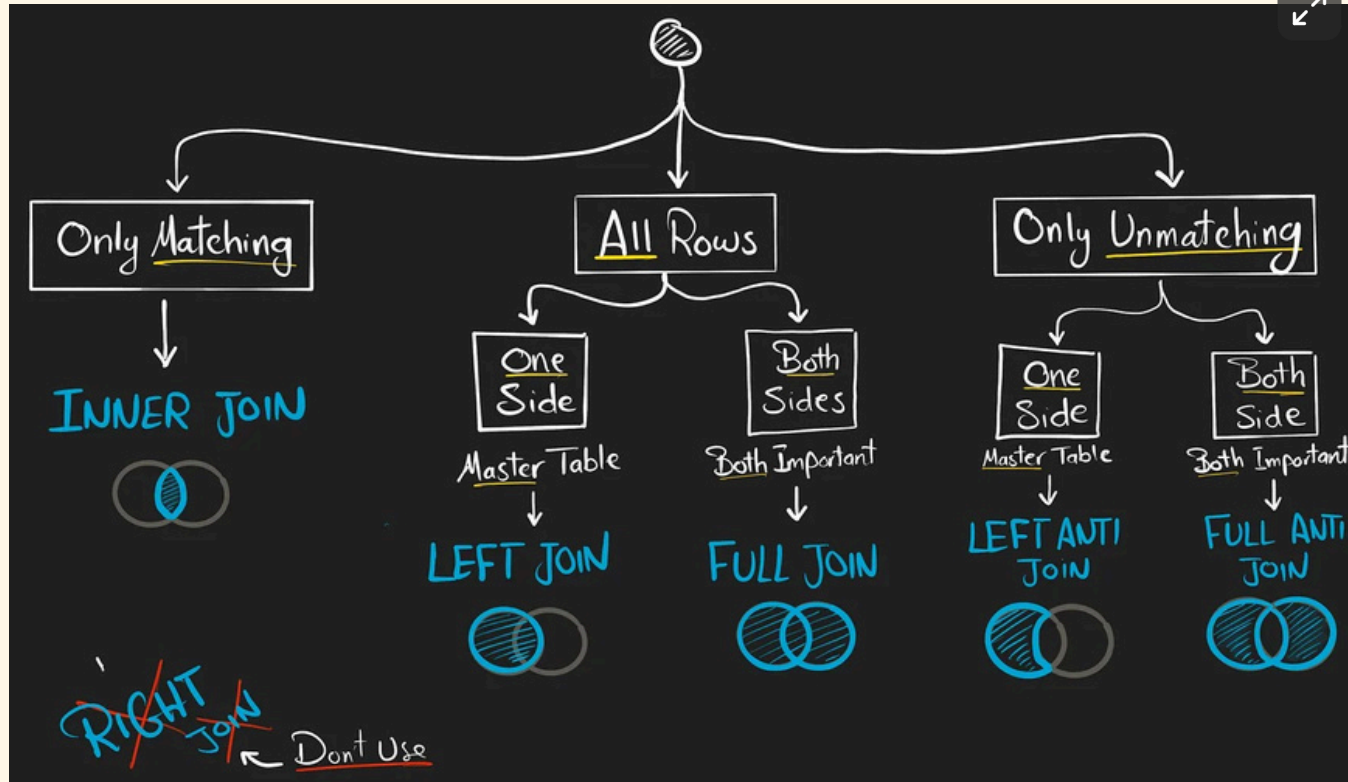


You may need to check if a record exists in another table or filter based on missing data.

- Example: You have a list of registered users but want to find out who has never made a purchase. By checking their presence in the sales table, you can identify them.
- Use **INNER JOIN** to return only matches.
- Use **LEFT** or **FULL JOIN with WHERE** to filter records missing from another table.



## SQL Join Decision Tree



- Picking the right SQL JOIN depends on what data you need. Here's a simple decision tree to guide you:

### 1. Need Only Matching Data?

- Use **INNER JOIN** to return rows present in both tables.




## 2. Need All Rows from One or Both Tables?

- One Side Matters? → Use LEFT JOIN to keep all rows from the main table.
- Both Sides Important? → Use FULL JOIN to keep everything from both tables.

## 3. Need Only Unmatched Data?

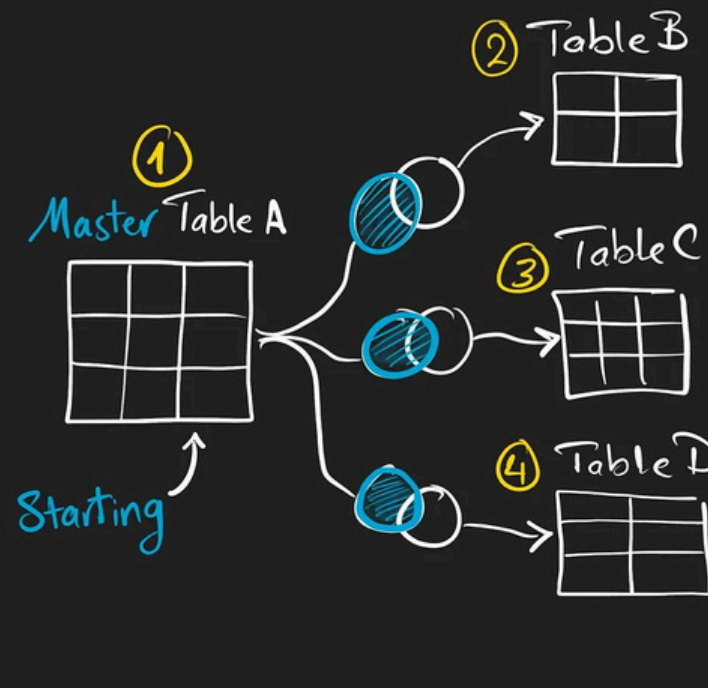
- One Side Matters? → Use LEFT ANTI JOIN to keep only rows without a match.

 **Right Join?** Just don't. You can always flip the tables and use a **LEFT JOIN** instead—it keeps queries more readable.



## How to Join My Tables?

```
SELECT *  
FROM A  
LEFT B ON...  
LEFT C ON...  
LEFT D ON...  
WHERE Control what  
      to keep
```



When working with multiple tables, I follow a structured approach to keep things clear and efficient.

### 1- Start with the Master Table

The **main table (A)** contains the core data I need. All joins will be based on this starting point.



## **2- Add Related Tables One by One**

I join each additional table (B, C, D, etc.) based on a common key. This keeps the query readable and avoids unnecessary complexity.

## **3- Use the WHERE Clause to Control the Output**

After joining, I apply a WHERE clause to filter the results and keep only the necessary data. This step is important for performance and ensuring the query returns only relevant records. This approach keeps joins organized, efficient, and easy to debug.

## **Weekly Wisdom**

*"Do what you can, with what you have, where you are"*

If you have questions or tips to share, I'd love to hear from you.

**Happy querying!**

**Follow for more content like this Azure  
Cloud for Data Engineering**



**Ganesh R**

**Azure Data Engineer**