It looks like you’re working on understanding SQL joins, particularly **INNER JOIN**. Here’s a concise breakdown of what you covered:

**Key Takeaways on SQL Joins**

1. **Why Join Tables?**
   * When data is spread across multiple tables, joining them allows us to combine relevant information.
   * Example: Linking a security vulnerabilities table with a machines table helps identify vulnerable devices.
2. **Syntax for Referring to Columns from Multiple Tables**
   * When a column exists in multiple tables, use **table\_name.column\_name** to specify which one you mean.
3. **INNER JOIN Basics**
   * **INNER JOIN** returns only the rows where there is a match in both tables.
   * It requires a **common column** (often a primary key in one table and a foreign key in another).
   * Example:
   * SELECT employees.username, employees.office, machines.operating\_system
   * FROM employees
   * INNER JOIN machines ON employees.employee\_id = machines.employee\_id;
   * Here, the **employees** table is the left table, and the **machines** table is the right table.
4. **Handling NULL Values**
   * If a machine isn’t assigned to an employee, it won’t appear in the INNER JOIN results because there’s no match.
5. **Result of INNER JOIN**
   * Only the rows where employee\_id exists in **both** tables appear in the result set.
   * The result includes only the **columns specified in SELECT**, even though the join merges all columns from both tables.

**Next Steps**

You’ll likely explore **other types of joins** soon, such as:

* **LEFT JOIN** (returns all rows from the left table and matching ones from the right)
* **RIGHT JOIN** (returns all rows from the right table and matching ones from the left)
* **FULL OUTER JOIN** (returns all rows from both tables, with NULL where no match exists)

Let me know if you need more explanations or examples!

### ****Summary of Outer Joins in SQL****

Outer joins allow us to combine tables **without requiring a match between them**. They help when we need all records from one or both tables, even if some values don’t have a corresponding match.

#### **1. LEFT JOIN (or LEFT OUTER JOIN)**

* Returns **all records from the first (left) table**.
* Returns **matching records from the second (right) table**.
* If no match is found, columns from the right table contain **NULL** values.

✅ **Example SQL Query**:

SELECT employees.username, employees.office, machines.operating\_system

FROM employees

LEFT JOIN machines ON employees.employee\_id = machines.employee\_id;

✅ **Use Case**:  
Retrieve **all employees** even if they don’t have an assigned machine.

#### **2. RIGHT JOIN (or RIGHT OUTER JOIN)**

* Returns **all records from the second (right) table**.
* Returns **matching records from the first (left) table**.
* If no match is found, columns from the left table contain **NULL** values.

✅ **Example SQL Query**:

SELECT employees.username, employees.office, machines.operating\_system

FROM employees

RIGHT JOIN machines ON employees.employee\_id = machines.employee\_id;

✅ **Use Case**:  
Retrieve **all machines**, even those **not assigned to an employee**.

#### **3. FULL OUTER JOIN**

* Returns **all records from both tables**.
* If there’s a match, values from both tables appear.
* If no match exists, **NULL values** appear for missing columns.

✅ **Example SQL Query**:

SELECT employees.username, employees.office, machines.operating\_system

FROM employees

FULL OUTER JOIN machines ON employees.employee\_id = machines.employee\_id;

✅ **Use Case**:  
Retrieve **all employees and all machines**, showing NULL where there’s no match.

### ****Key Takeaways****

* **INNER JOIN** → Returns only matching records.
* **LEFT JOIN** → Returns all records from the left table, plus matching ones from the right.
* **RIGHT JOIN** → Returns all records from the right table, plus matching ones from the left.
* **FULL OUTER JOIN** → Returns all records from both tables, filling in NULLs when no match exists.

Would you like to see more real-world use cases or practice queries?

* **FULL OUTER JOIN** returns all records from both tables, with NULL values where there are no matches in either table.

The syntax is similar to INNER JOIN, but with **LEFT JOIN, RIGHT JOIN, or FULL OUTER JOIN** keywords.

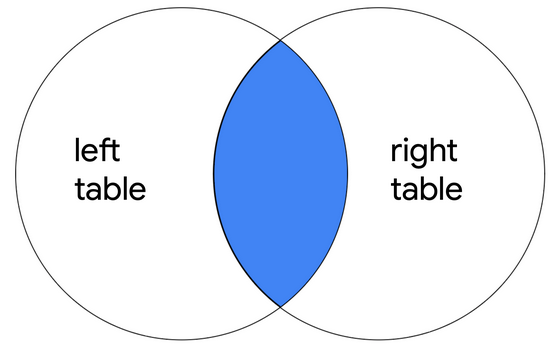
As a security analyst, you do not need to memorize all of these, but understanding their use cases will help you efficiently query databases. You can always look up specific syntax as needed. These join types are essential for working with SQL as a security analyst.

# Compare types of joins

Previously, you explored SQL joins and how to use them to join data from multiple tables when these tables share a common column. You also examined how there are different types of joins, and each of them returns different rows from the tables being joined. In this reading, you'll review these concepts and more closely analyze the syntax needed for each type of join.

## Inner joins

The first type of join that you might perform is an inner join.INNER JOIN returns rows matching on a specified column that exists in more than one table.



It only returns the rows where there is a match, but like other types of joins, it returns all specified columns from all joined tables. For example, if the query joins two tables with SELECT \*, all columns in both of the tables are returned.

**Note:** If a column exists in both of the tables, it is returned twice when SELECT \* is used.

### The syntax of an inner join

To write a query using INNER JOIN, you can use the following syntax:

SELECT \*

FROM employees

INNER JOIN machines ON employees.device\_id = machines.device\_id;

You must specify the two tables to join by including the first or left table after FROM and the second or right table after INNER JOIN.

After the name of the right table, use the ON keyword and the = operator to indicate the column you are joining the tables on. It's important that you specify both the table and column names in this portion of the join by placing a period (.) between the table and the column.

In addition to selecting all columns, you can select only certain columns.  For example, if you only want the join to return the username, operating\_system and device\_id columns, you can write this query:

SELECT username, operating\_system, employees.device\_id

FROM  employees

INNER JOIN machines ON employees.device\_id = machines.device\_id;

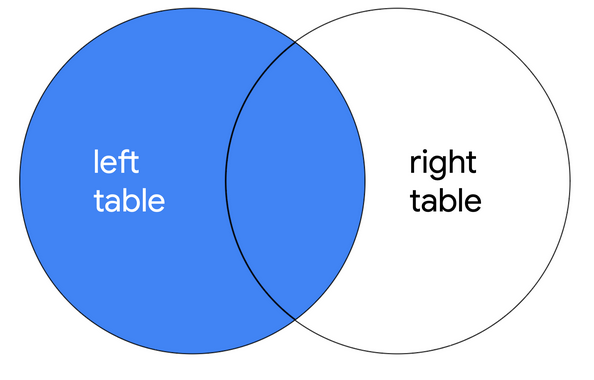
**Note**: In the example query, username and operating\_system only appear in one of the two tables, so they are written with just the column name. On the other hand, because device\_id appears in both tables, it's necessary to indicate which one to return by specifying both the table and column name (employees.device\_id).

## Outer joins

Outer joins expand what is returned from a join. Each type of outer join returns all rows from either one table or both tables.

### Left joins

When joining two tables, LEFT JOINreturns all the records of the first table, but only returns rows of the second table that match on a specified column.



The syntax for using LEFT JOIN is demonstrated in the following query:

SELECT \*

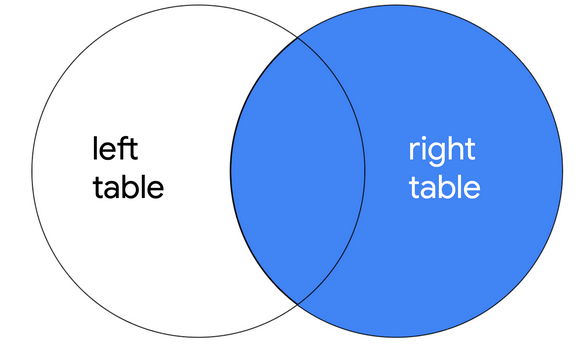
FROM employees

LEFT JOIN machines ON employees.device\_id = machines.device\_id;

As with all joins, you should specify the first or left table as the table that comes after FROM and the second or right table as the table that comes after LEFT JOIN. In the example query, because employees is the left table, all of its records are returned. Only records that match on the device\_id column are returned from the right table, machines.

### Right joins

When joining two tables, RIGHT JOIN returns all of the records of the second table, but only returns rows from the first table that match on a specified column.



The following query demonstrates the syntax for RIGHT JOIN:

SELECT \*

FROM employees

RIGHT JOIN machines ON employees.device\_id = machines.device\_id;

RIGHT JOIN has the same syntax as LEFT JOIN, with the only difference being the keyword RIGHT JOIN instructs SQL to produce different output. The query returns all records from machines, which is the second or right table. Only matching records are returned from employees, which is the first or left table.

**Note:**  You can use LEFT JOIN and RIGHT JOIN and return the exact same results if you use the tables in reverse order. The following RIGHT JOIN query returns the exact same result as the LEFT JOIN query demonstrated in the previous section:

SELECT \*

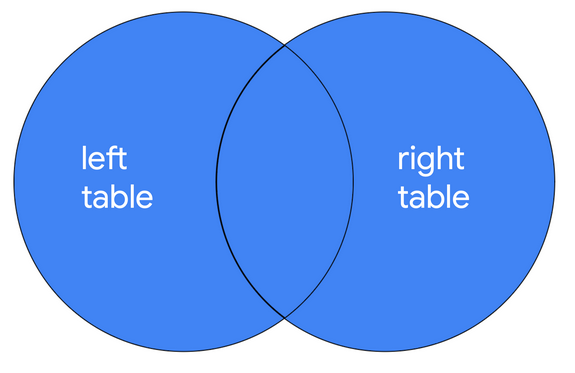
FROM machines

RIGHT JOIN employees ON employees.device\_id = machines.device\_id;

All that you have to do is switch the order of the tables that appear before and after the keyword used for the join, and you will have swapped the left and right tables.

### Full outer joins

FULL OUTER JOIN returns all records from both tables. You can think of it as a way of completely merging two tables.



You can review the syntax for using FULL OUTER JOIN in the following query:

SELECT \*

FROM employees

FULL OUTER JOIN machines ON employees.device\_id = machines.device\_id;

The results of a FULL OUTER JOIN query include all records from both tables. Similar to INNER JOIN, the order of tables does not change the results of the query.

## Key takeaways

When working in SQL, there are multiple ways to join tables.  All joins return the records that match on a specified column. INNER JOIN will return only these records. Outer joins also return all other records from one or both of the tables. LEFT JOIN returns all records from the first or left table, RIGHT JOIN returns all records from the second or right table, and FULL OUTER JOIN returns all records from both tables.