JOINs

by Sophia

WHAT'S COVERED

In this lesson, you will learn how JOIN clauses work and how you can use them to find meaningful facts and interesting associations in the data. Specifically, this lesson will cover:

1. JOIN Basics

JOINs are one of the most fundamental concepts in SQL. They enable you to retrieve data from multiple related tables in a single query. A JOIN enables you to combine rows from several tables, pulling data from different sources into a cohesive dataset. JOINs can help you avoid data duplication and maintain a well-structured schema.

KEY CONCEPT

One common mistake is to confuse a JOIN with a relationship. Relationships are permanent connections between tables based on common fields. A JOIN is an operation used in a query to specify how data from multiple related tables will be included in a query. JOINs rely on existing relationships within the database and are specific to the query in which they are referenced.

PostgreSQL supports several types of JOINs, each serving a specific purpose. Common JOIN types include INNER JOINs, LEFT JOINs (or LEFT OUTER JOINs), RIGHT JOINs (or RIGHT OUTER JOINs), and FULL JOINs (or FULL OUTER JOINs). You will work with these in upcoming lessons, as well as NATURAL JOINs, CROSS JOINs, and self JOINs. Each JOIN type has its own unique behaviors, as you will discover.

2. INNER JOINs

The INNER JOIN is a JOIN in SQL in which rows from multiple tables are combined based on a condition. In this way, only rows matching the condition are included in the result set. The purpose of this type of JOIN is to retrieve data from the participating tables with matching values in the specified columns.

HINT

An INNER JOIN returns only rows from both tables with corresponding values, effectively removing the nonmatching records.

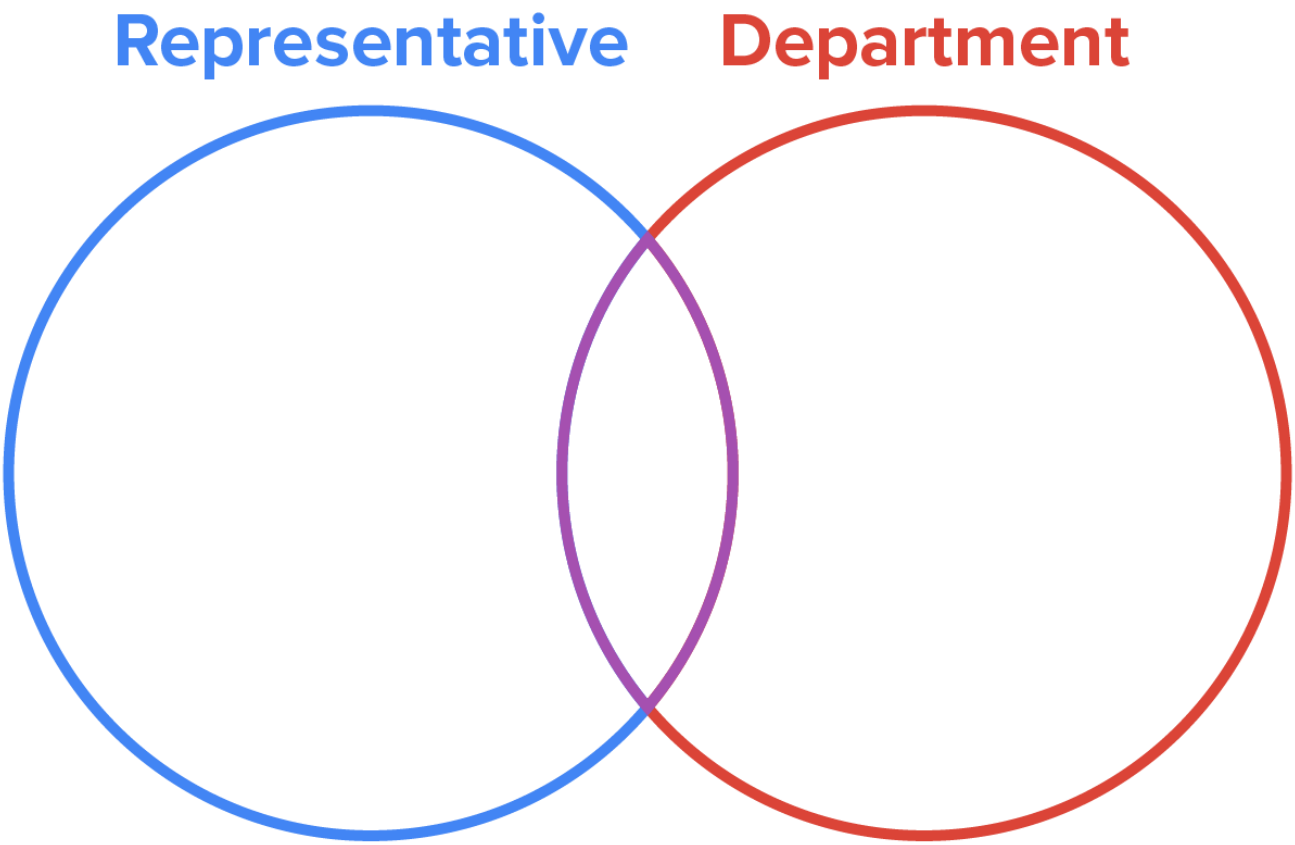
An INNER JOIN is commonly used when combining data from related tables, where the columns being joined have a defined relationship. A database with two tables, such as "Customers" and "Orders," can be used to fetch records indicating which customers have placed orders, and their details. By reducing the result set to only the data that meets the specified criteria, the INNER JOIN operation enhances query efficiency. As a result, complex datasets can be queried in a precise and organized manner with its help.

EXAMPLE First, we will create two tables for representatives and departments. The common attribute between the two tables is the representative_id from the representative table and the manager_id in the manager table.

Notice that in this case, the two column names are different from one another but linked through the primary key.

```
CREATE TABLE representative ( representative_id INT PRIMARY KEY, first_name VARCHAR (30) NOT NULL, last_name VARCHAR (30) NOT NULL );
CREATE TABLE department ( department_id INT PRIMARY KEY, department_name VARCHAR (100) NOT NULL, manager_id INT, constraint fk_manager FOR
);
INSERT INTO representative (representative_id, first_name, last_name)
VALUES (1, 'Bob','Evans'), (2, 'Tango','Rushmore'), (3, 'Danika','Arkane'), (4, 'Mac','Anderson');
INSERT INTO department (department_id, department_name,manager_id)
VALUES (1, 'Sales', 1), (2, 'Marketing', 3), (3, 'IT', 4), (4, 'Finance', null), (5, 'Support', null);
```

We can represent this data through the use of a **Venn diagram**, where the left circle represents the data from the representative table, and the right circle represents the data from the department table:



By running a SELECT statement for each table individually, we can see that Tango Rushmore is not a department manager (because his ID does not appear in the manager_id column) and that the Finance and Support departments still need a manager (because the manager_id column is null for their records).

Individually, the data looks like the following:

Query Results

Row count: 4

representative_id	first_name	last_name
1	Bob	Evans
2	Tango	Rushmore
3	Danika	Arkane
4	Mac	Anderson

Query Results

Row count: 5

department_id	department_name	manager_id
1	Sales	1
2	Marketing	3
3	IT	4
4	Finance	
5	Support	

We can complete an INNER JOIN by identifying the column name. The basic INNER JOIN between the two tables should look like the following:

```
SELECT *
FROM representative
JOIN department
ON representative.representative_id = department.manager_id;
```

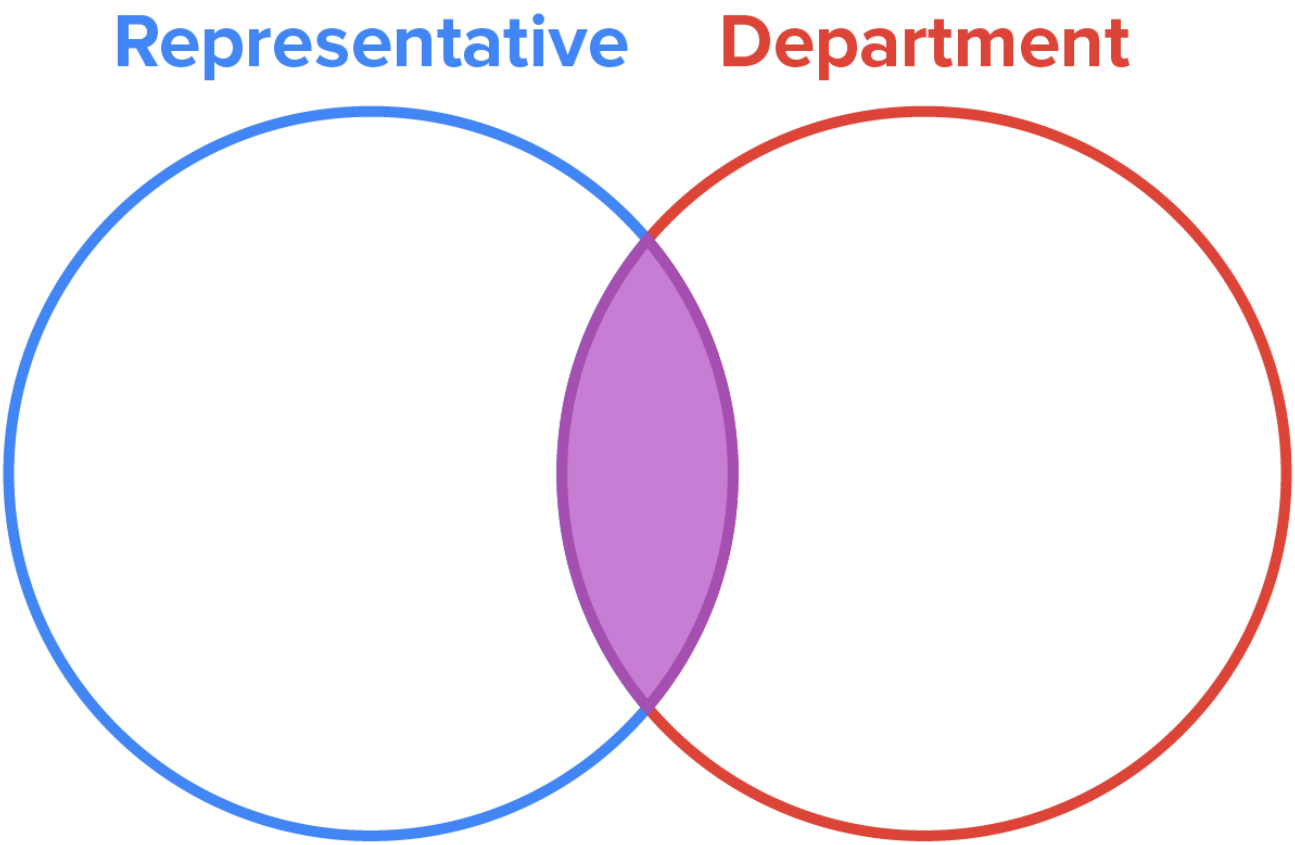
With this INNER JOIN, we find that the data in the representative_id column in the representative table matches the department's manager_id. There should be three that match:

Query Results

Row count: 3

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4

The Venn diagram below illustrates the INNER JOIN:



This is the most common type of JOIN, as we typically want to be able to identify where data between two or more tables matches.

TERM TO KNOW

Venn Diagram

A diagram that illustrates logical relationships between two or more sets of items by using overlapping circles or shapes. In many cases, they serve as a visual way to summarize similar and different aspects of items.

3. OUTER JOINs

OUTER JOINs include nonmatching rows from one or both of the participating tables, extending INNER JOIN capabilities. An OUTER JOIN includes rows where the condition is not met, unlike an INNER JOIN, which retrieves only rows with matching values. This ensures that the result set contains data from both tables. OUTER JOINs come in three varieties: LEFT OUTER JOIN, RIGHT OUTER JOIN, and FULL OUTER JOIN. This lesson explains the basic concepts behind OUTER JOINs; you will learn their specifics and how to create them in later lessons.

KEY CONCEPT

LEFT OUTER JOINs retrieve all rows from the left table and the matching rows from the right table. In the event that there is no match in the right table, the result will still show the corresponding row from the left table along with NULL values for the columns in the right table.

KEY CONCEPT

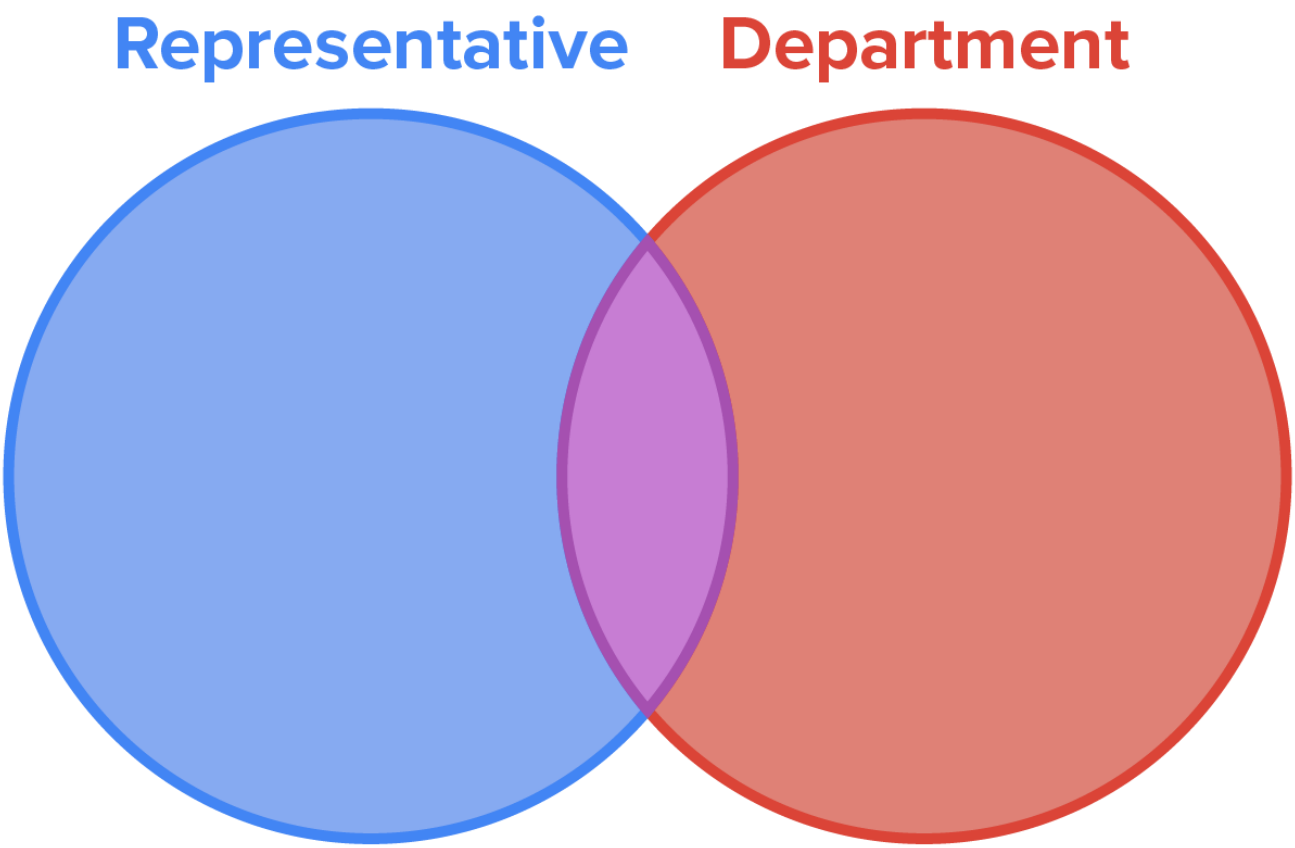
In contrast, a RIGHT OUTER JOIN includes all rows from the right table and the matching rows from the left table. NULL values are also included in the left table columns when there is no match.

KEY CONCEPT

In a FULL OUTER JOIN, both LEFT and RIGHT OUTER JOINs are combined. All rows from both tables are included, and NULL values are filled in where there are no matches. Using an OUTER JOIN provides a comprehensive view of data relationships in your PostgreSQL database, even when there are no matches in the specified JOIN condition.

If you think about your left hand and right hand, the LEFT OUTER JOIN and RIGHT OUTER JOIN make more sense. If you have your customer table in your left hand and your orders table in your right hand, you can join those tables using the OUTER JOIN, LEFT, RIGHT or FULL to determine what data you want to get. You want the match in the

customer table (LEFT OUTER JOIN), the match in the orders table (RIGHT OUTER JOIN), and then the data between the two of them (FULL OUTER JOIN).

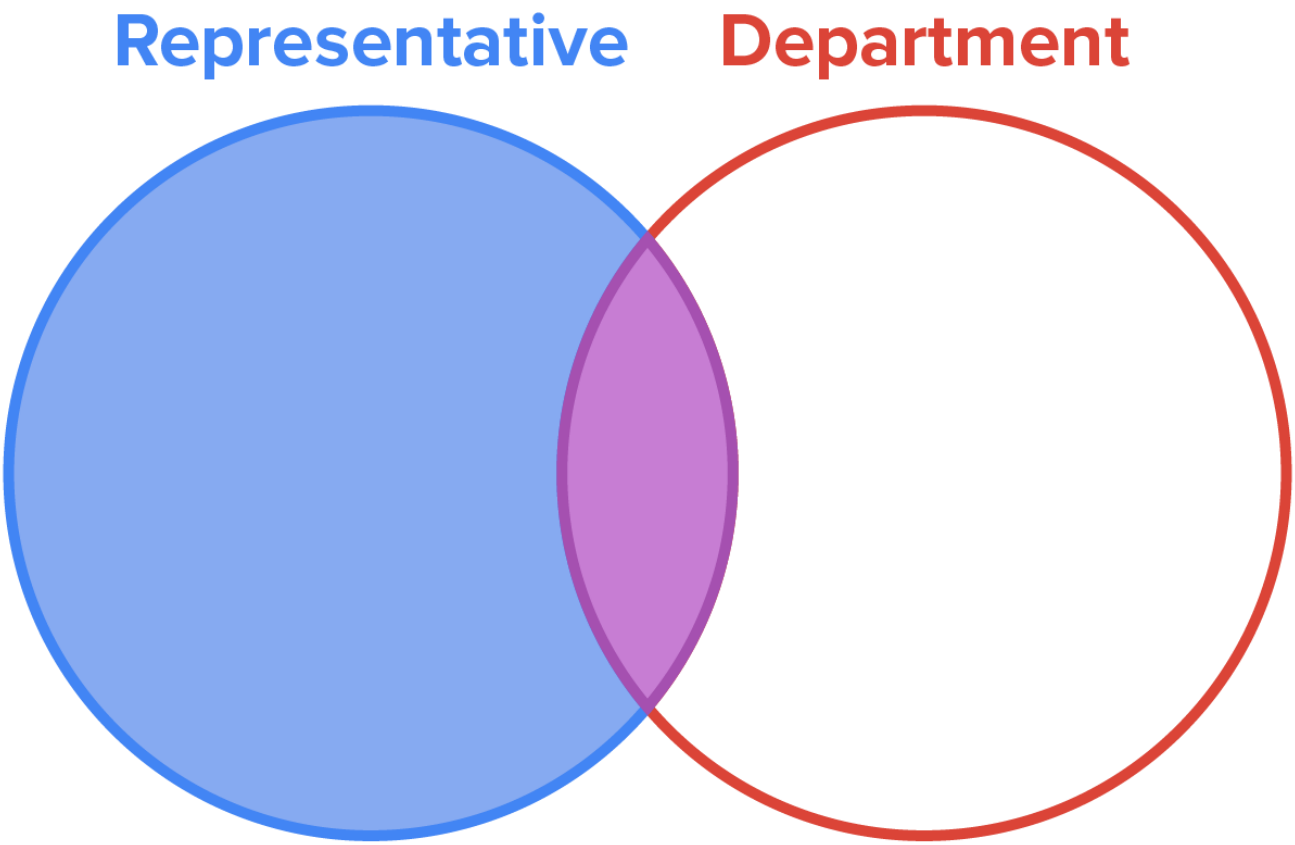


Query Results

Row count: 6

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4
			4	Finance	
			5	Support	
2	Tango	Rushmore			

We may use a LEFT JOIN to find data that exists in both tables and also data in the left table that does not match. This Venn diagram illustrates a LEFT JOIN.

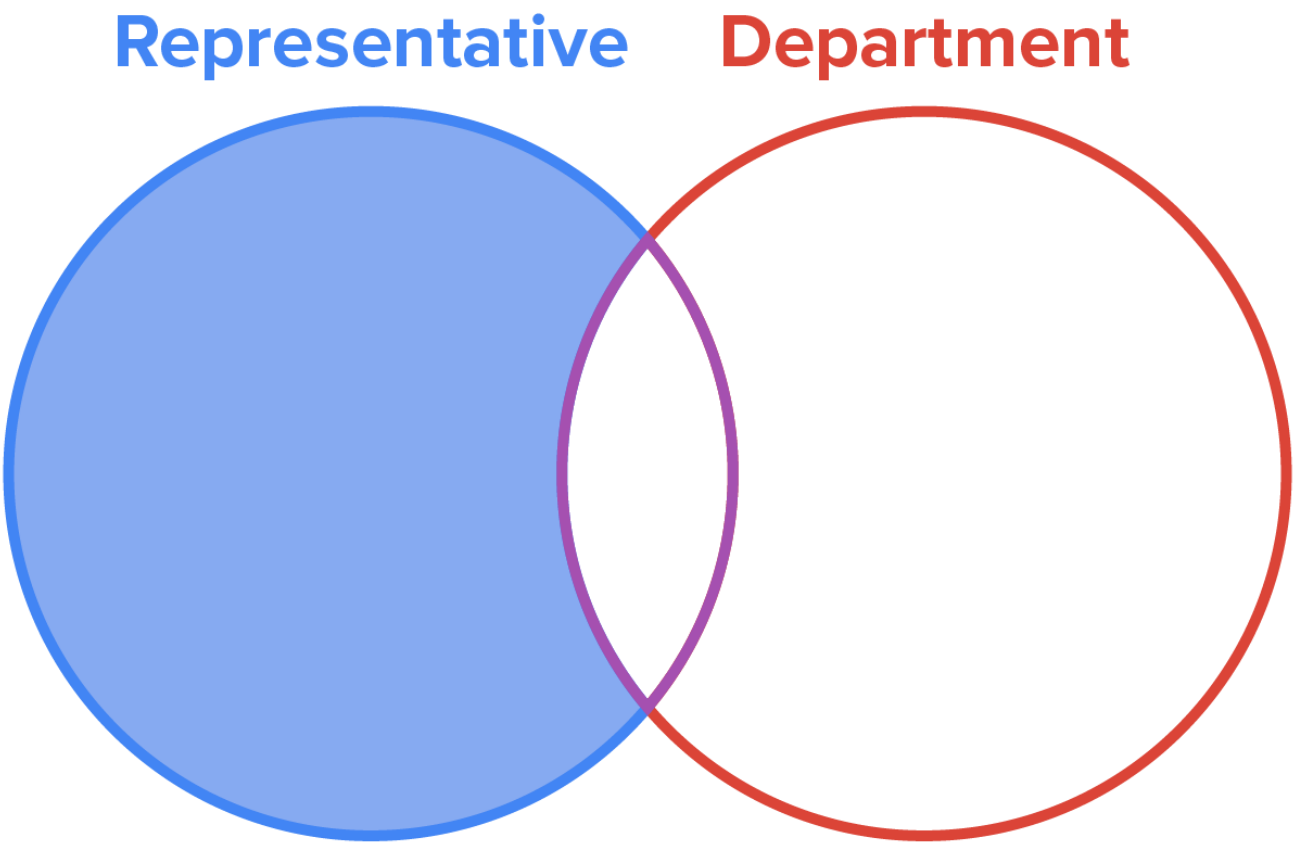


Query Results

Row count: 4

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4
2	Tango	Rushmore			

We may also want to find data that exists in the left table but does not match with data in the right table. This would be considered a LEFT OUTER JOIN:

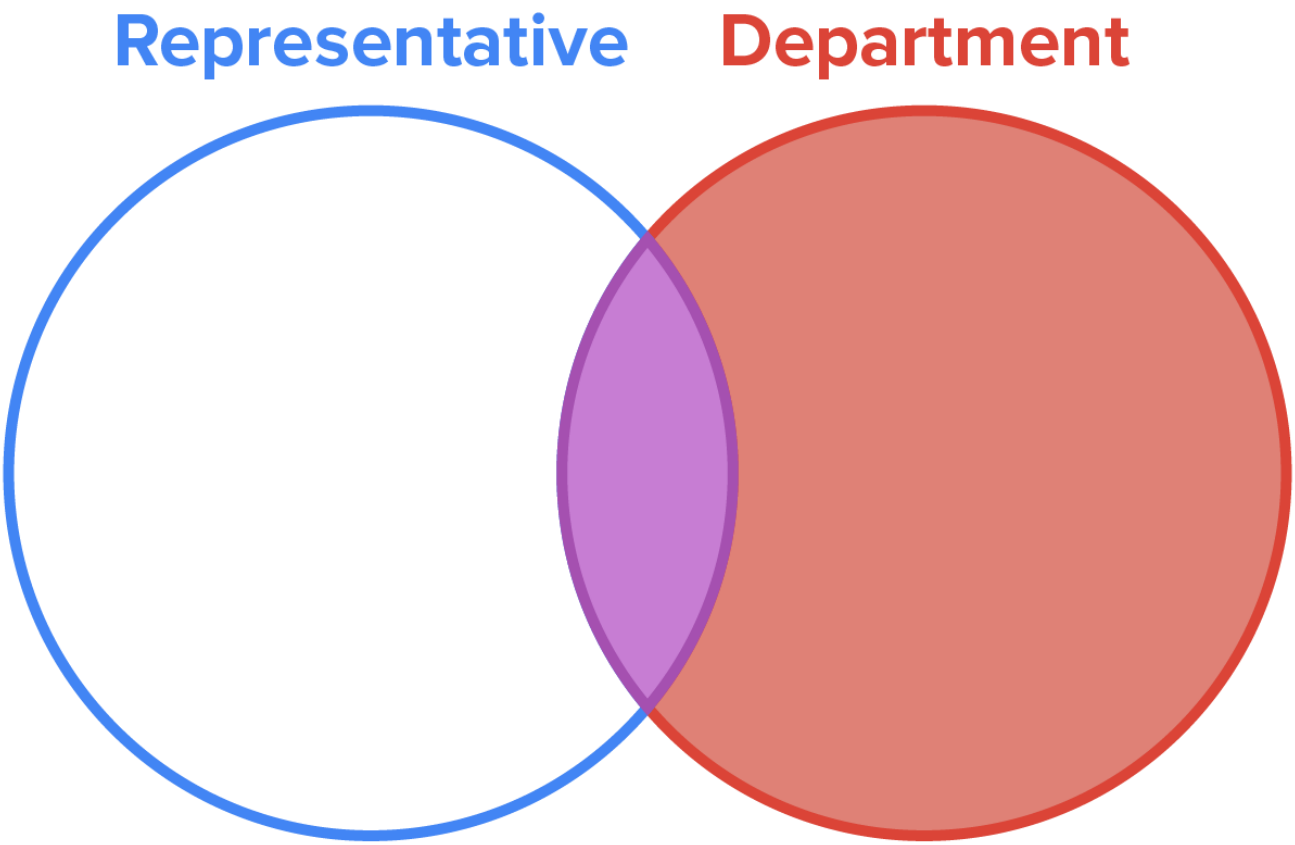


Query Results

Row count: 1

representative_id	first_name	last_name	department_id	department_name	manager_id
2	Tango	Rushmore			

Using a RIGHT JOIN, we can get the data that exists in both tables, along with the data that does not match from the right table:

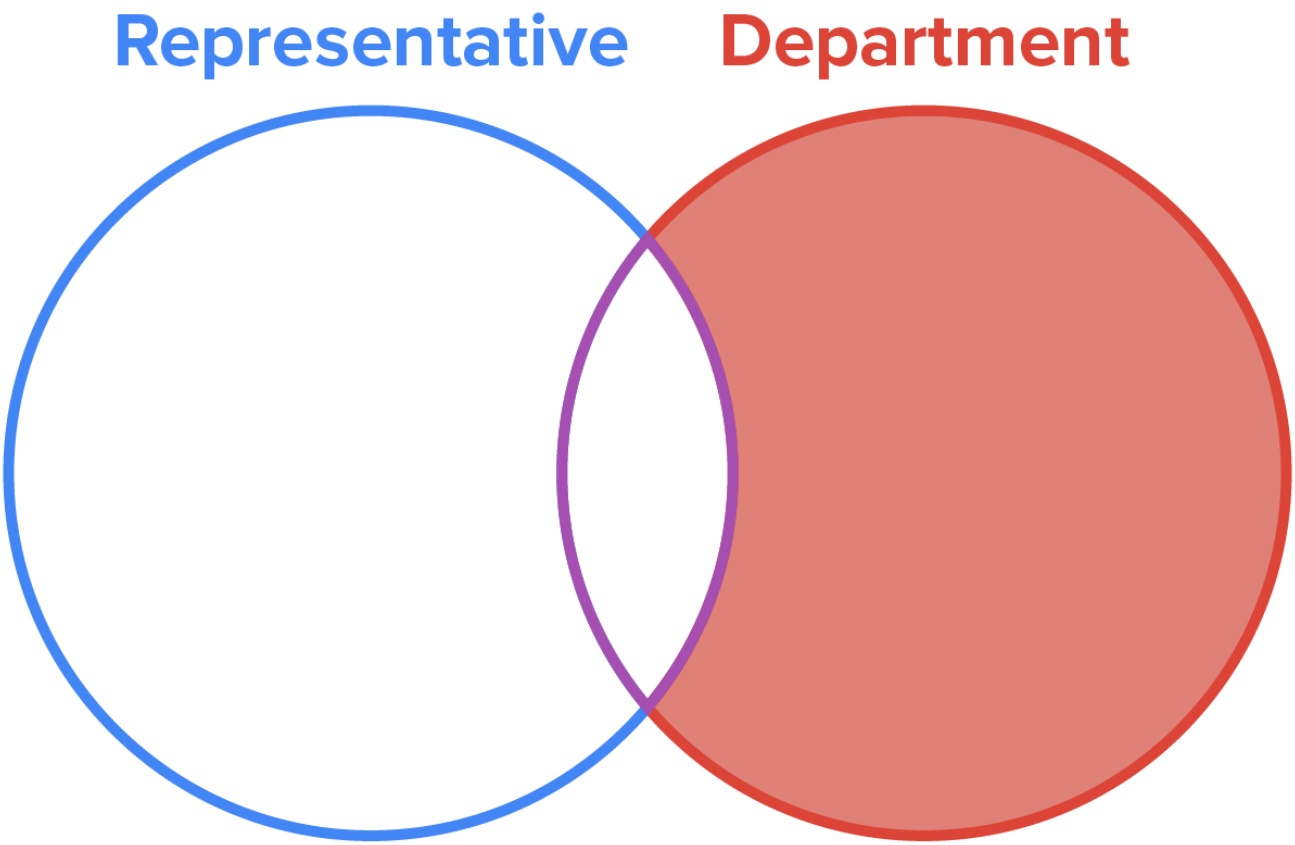


Query Results

Row count: 5

representative_id	first_name	last_name	department_id	department_name	manager_id
1	Bob	Evans	1	Sales	1
3	Danika	Arkane	2	Marketing	3
4	Mac	Anderson	3	IT	4
			4	Finance	
			5	Support	

We can also find data that exists in the right table but does not match with data in the left table, using a RIGHT OUTER JOIN:



WATCH

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

SUMMARY

In this lesson, you were introduced to the **basics of JOINs**, which are query operations that enable you to retrieve data from multiple related tables based on specified criteria. This enables you to create complex queries that generate meaningful insights. There are several kinds of JOINs, including INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.

An **INNER JOIN** combines rows from two or more tables, returning only those rows that match the condition. Using this type of JOINs allows you to retrieve data that have matching values in the specified columns of the participating tables. **OUTER JOINs**, including LEFT JOIN, RIGHT JOIN, or FULL JOIN, include nonmatching rows from one or both tables as well as matching rows from one or both tables. FULL JOIN retrieves all rows from both tables, filling in NULL values where there are no matches in either table. LEFT JOIN retrieves all rows from both tables, while RIGHT JOIN does the opposite. In each of the upcoming lessons in this challenge, you will learn about the potential uses of each JOIN type and how to create them.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND FAITHE WEMPEN (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.

TERMS TO KNOW

Venn Diagram

Venn diagrams illustrate logical relationships between two or more sets of items by using overlapping circles or shapes. In many cases, they serve as a visual way to summarize similar and different aspects of items.