RIGHT JOINs

by Sophia

WHAT'S COVERED

In this lesson, you will explore using the RIGHT JOIN to query data from tables, in two parts. Specifically, this lesson will cover:

1. RIGHT JOINs

A RIGHT JOIN is the complement of a LEFT JOIN. It uses the same command syntax and options, but the right (or second) table is the one from which all records are retrieved, whereas the left (or first) table is the one from which appears only matching records.

Most data analysis scenarios prioritize preserving data from the left table, so RIGHT JOINs are rarely used in practice. Just remember that they are available to you, in case you encounter a situation where you need to focus on the right-hand table due to the data structure or query requirements.

The structure of the query looks like this:

```
SELECT <columnlist>
FROM <table1>
RIGHT JOIN <table2>
ON <table1>.<table1column1> = <table2>.<table2column1>
```

TERM TO KNOW

RIGHT JOIN
A clause that combines data from two tables based on a specified condition. It retrieves all records from the right (or second) table and matching records from the left (or first) table. In the result set, only matching rows from the left and right tables are included, and any unmatched rows from the right table will have NULL values.

2. Examples

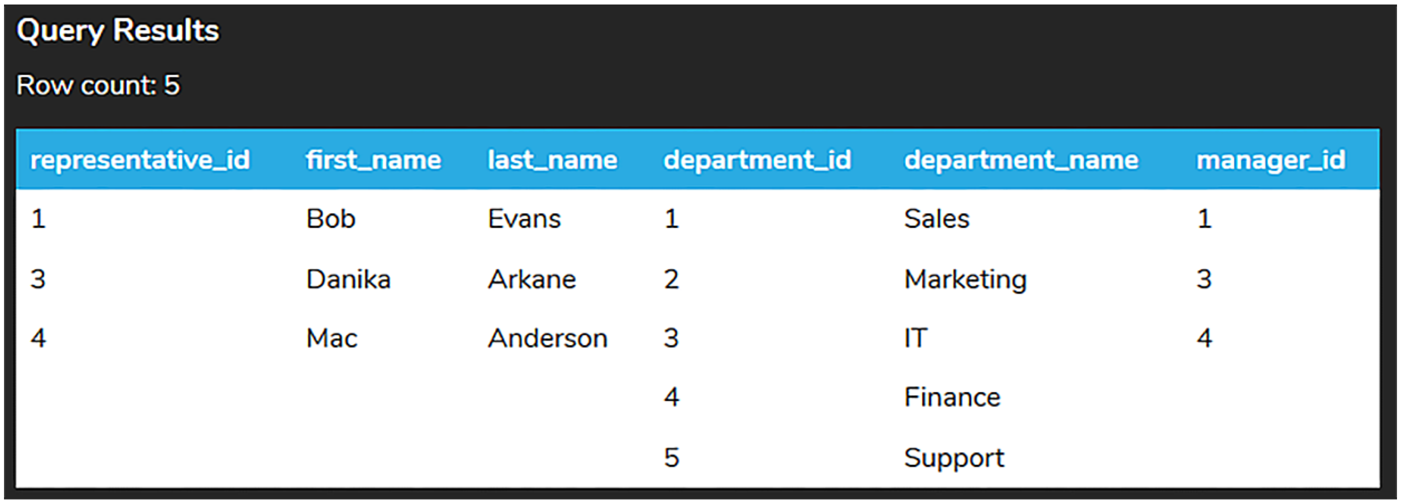
Let's again revisit our data set with the representatives and departments:

```
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 FOREIGN KEY (manager_id) REFERENCES representative(representative_id)
);
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);
```

Let's see what they would look like with the RIGHT JOIN:

```
SELECT *
```

```
FROM representative
RIGHT JOIN department ON representative.representative_id = department.manager_id;
```



Query Results

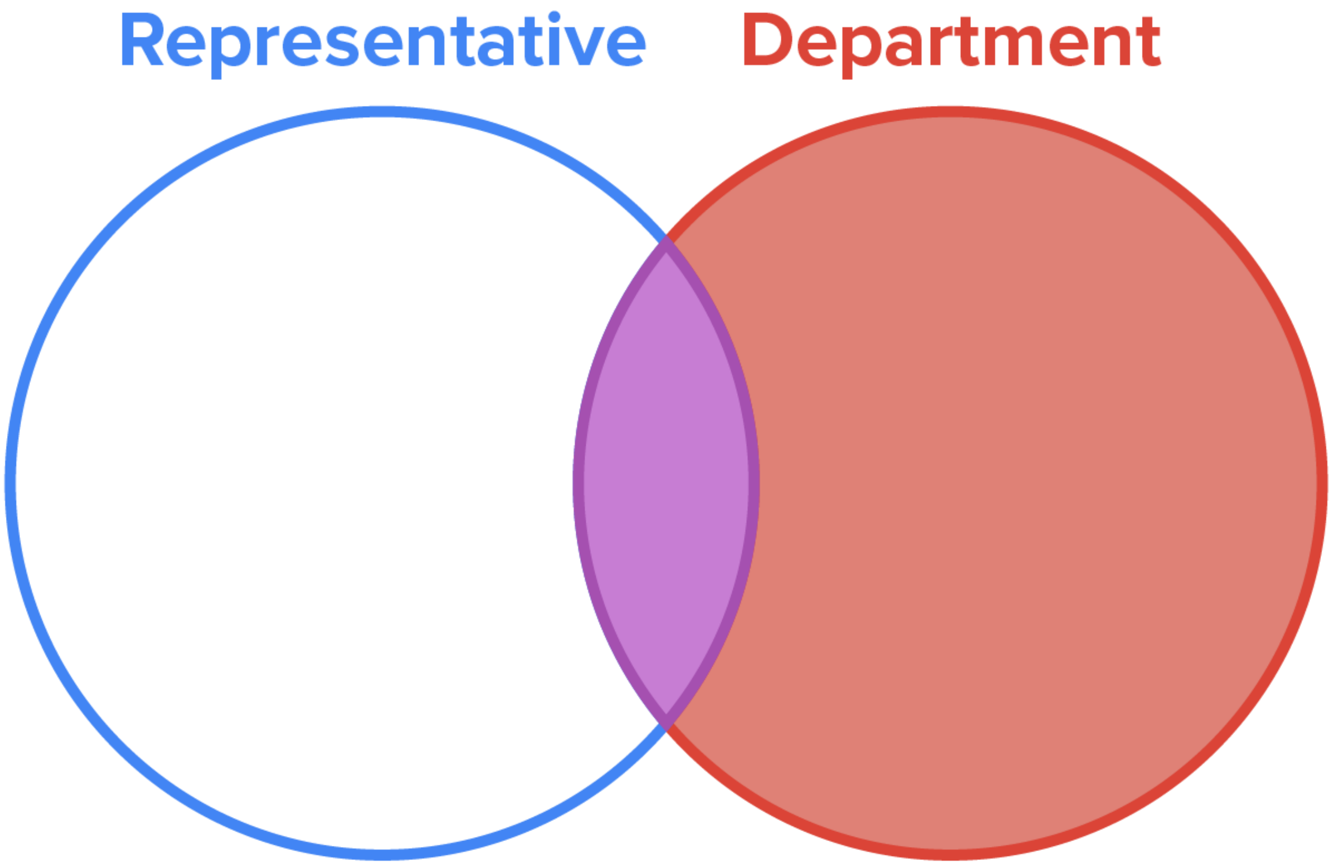
Row count: 5

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	

The query itself isn't that different from the LEFT JOIN, other than specifying RIGHT instead of LEFT. Notice that the last two rows are from the department table, where there is not a match with the representative table.

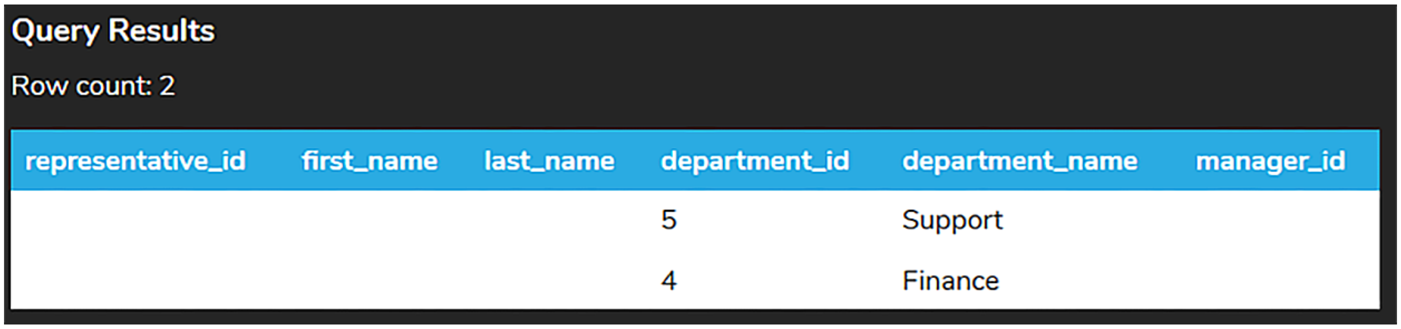
Recall that the table listed in the FROM clause (representative) is considered to be the left table, and the table after the JOIN clause (department) is considered to be the right table. The RIGHT JOIN starts by selecting data from the right table (department). It compares the manager_id from the department table to the representative_id in the representative table. If those values are equal, the RIGHT JOIN creates a row containing both tables' columns and adds the new row in the result set. You can see that in the first three rows in the table. If the values are not equal, the RIGHT JOIN also creates a row containing the columns from both of the tables but fills the columns from the left table (representative) with NULL values.

The Venn diagram for the RIGHT JOIN looks like the following:



As with the LEFT JOIN in the previous lesson, you can add a WHERE clause to retrieve the rows from the right table that do not match any of the rows in the left table.

```
SELECT *
FROM representative
RIGHT JOIN department ON representative.representative_id = department.manager_id
WHERE manager_id IS NULL;
```

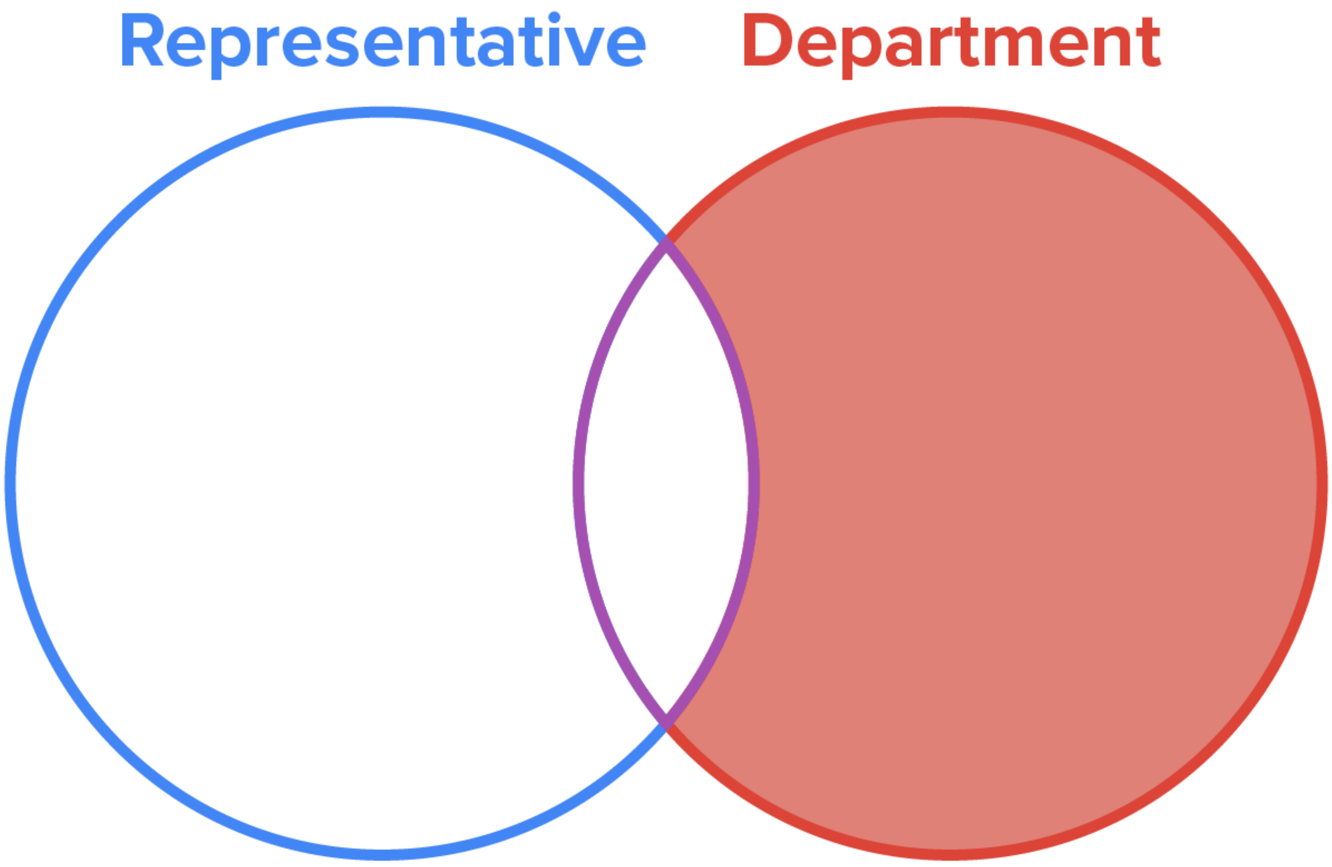


Query Results

Row count: 2

representative_id	first_name	last_name	department_id	department_name	manager_id
			5	Support	
			4	Finance	

The following Venn diagram illustrates the result of the RIGHT JOIN with the WHERE clause that specifies only unmatched rows from the right-hand table should be included. This is sometimes called a RIGHT OUTER JOIN (including in this lesson's video below), but it's created using a WHERE clause; there is not a separate RIGHT OUTER JOIN clause in SQL.



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 learned how to use a **RIGHT JOIN** to retrieve all the records from the right table and only the records from the left table that match them. This type of query can help analyze data based on a specific condition.

In the **examples** section, you learned the syntax for two types of RIGHT JOINs. In the first example, a RIGHT JOIN is used to include all records from the right-hand table and the data from any corresponding records from the left-hand table. In the second example, you learned how to add a WHERE clause to show only records from the right-hand table that have no corresponding records on the left. This is sometimes called a RIGHT OUTER JOIN, and it is referenced as such in this lesson's video.

TERMS TO KNOW

RIGHT JOIN

A clause that combines data from two tables based on a specified condition. It retrieves all records from the right (or second) table and matching records from the left (or first) table. In the result set, only matching rows from the left and right tables are included, and any unmatched rows from the right table will have NULL values.