Unlock the Power of
INNER, LEFT, RIGHT, and
FULL JOINS

Mastering SQL Joins in MySQL: Connecting Data Across Tables

WHAT ARE SQL JOINS?

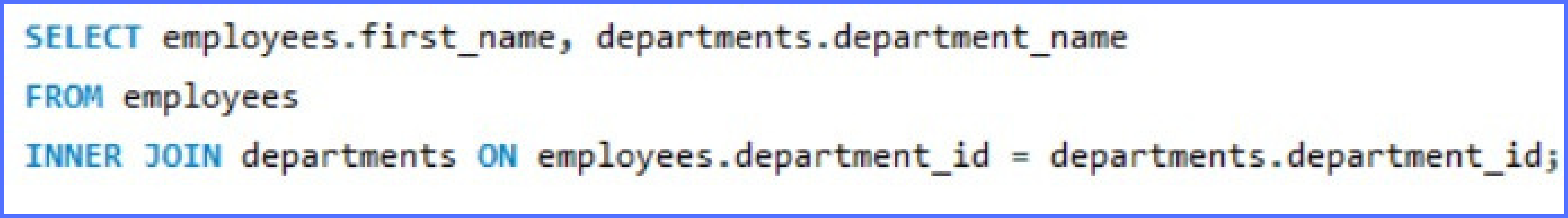
- Joins allow you to combine rows from two or more tables based on a related column.
- Why Use Joins?
- In relational databases, data is often stored across multiple tables. Joins help in querying data from these tables as if they were one.

INNER JOIN – RETURNING ONLY MATCHING ROWS

An INNER JOIN retrieves only the rows that have matching values in both tables. It’s like asking the database: "Show me the data where these two tables intersect."

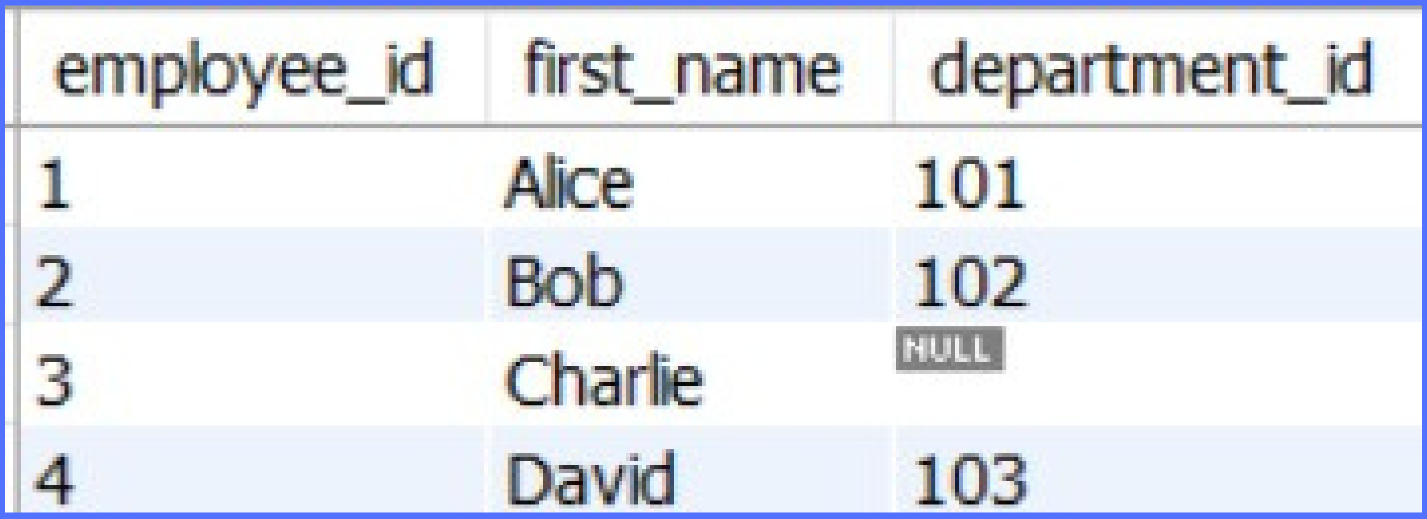
- Use Case:

You want to see a list of employees and their respective departments. If an employee is not assigned to a department, they won’t be included in the result.



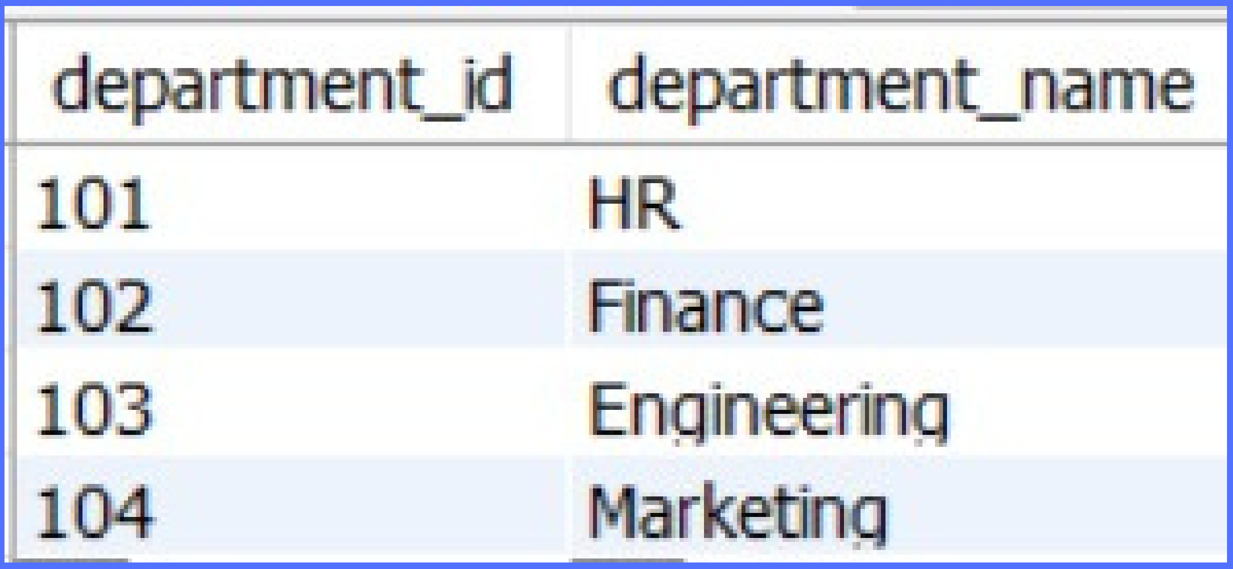
```
SELECT employees.first_name, departments.department_name
FROM employees
INNER JOIN departments ON employees.department_id = departments.department_id;
```

INNER JOIN – RETURNING ONLY MATCHING ROWS



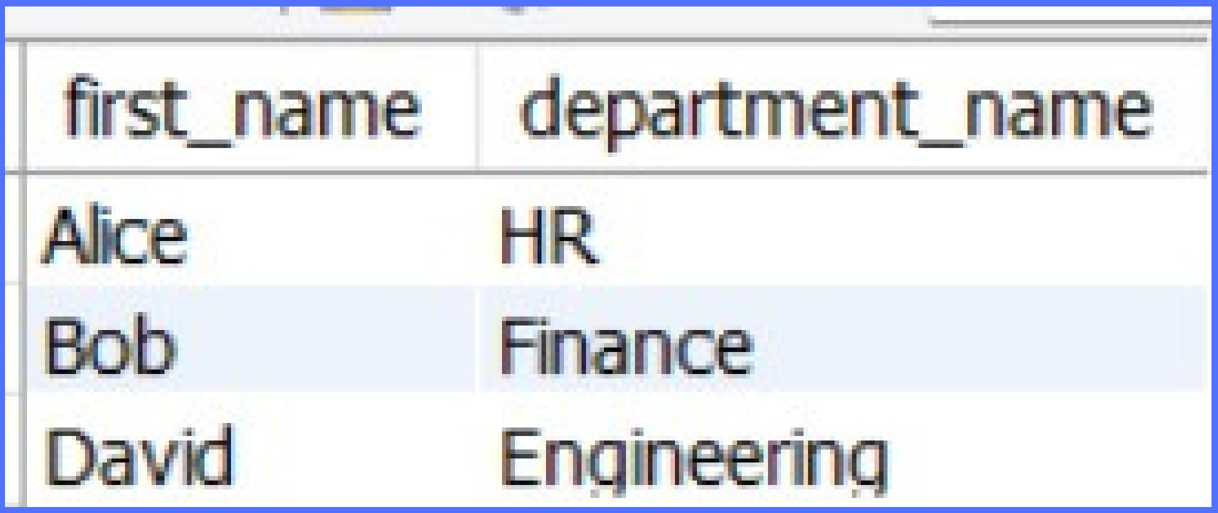
employee_id	first_name	department_id
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103

+



department_id	department_name
101	HR
102	Finance
103	Engineering
104	Marketing

=

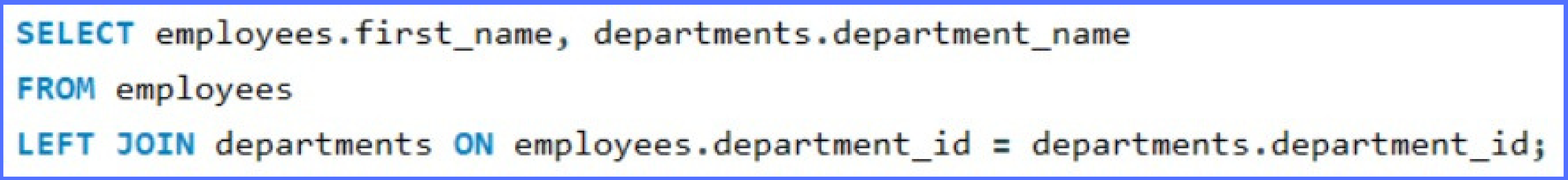


first_name	department_name
Alice	HR
Bob	Finance
David	Engineering

LEFT JOIN – ALL ROWS FROM THE LEFT TABLE

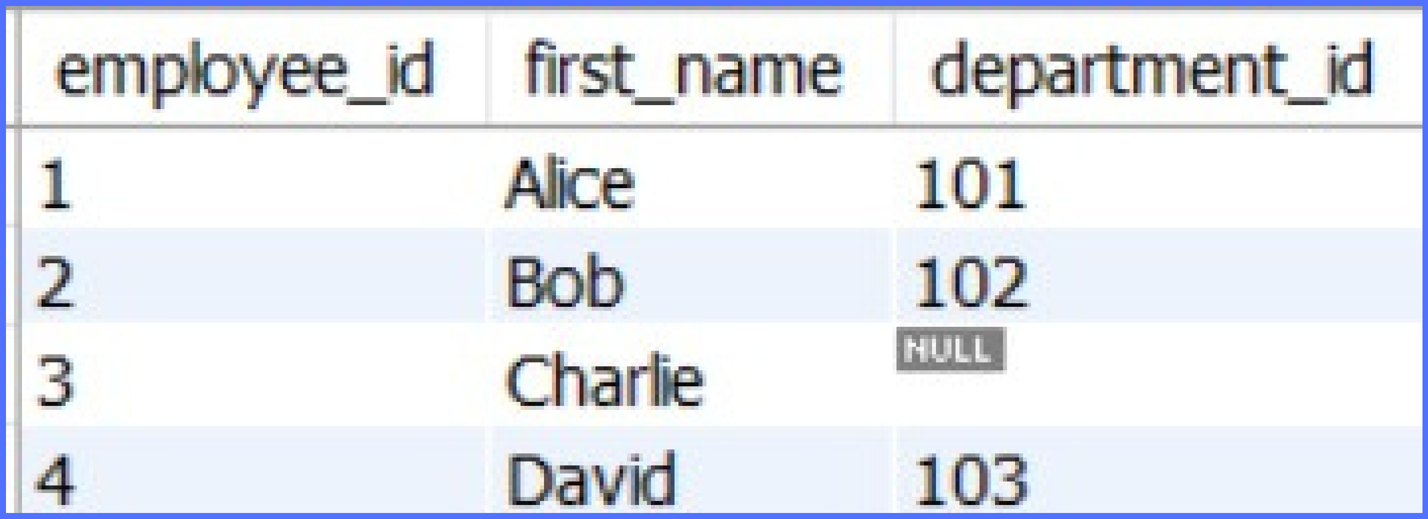
A LEFT JOIN returns all rows from the left table, along with the matching rows from the right table. If there’s no match, the result will still include the left table’s rows, but the columns from the right table will have NULL values.

- Use Case:
- You want a full list of employees, including those who haven’t been assigned to a department yet. The result will show employees without departments as NULL in the department column.



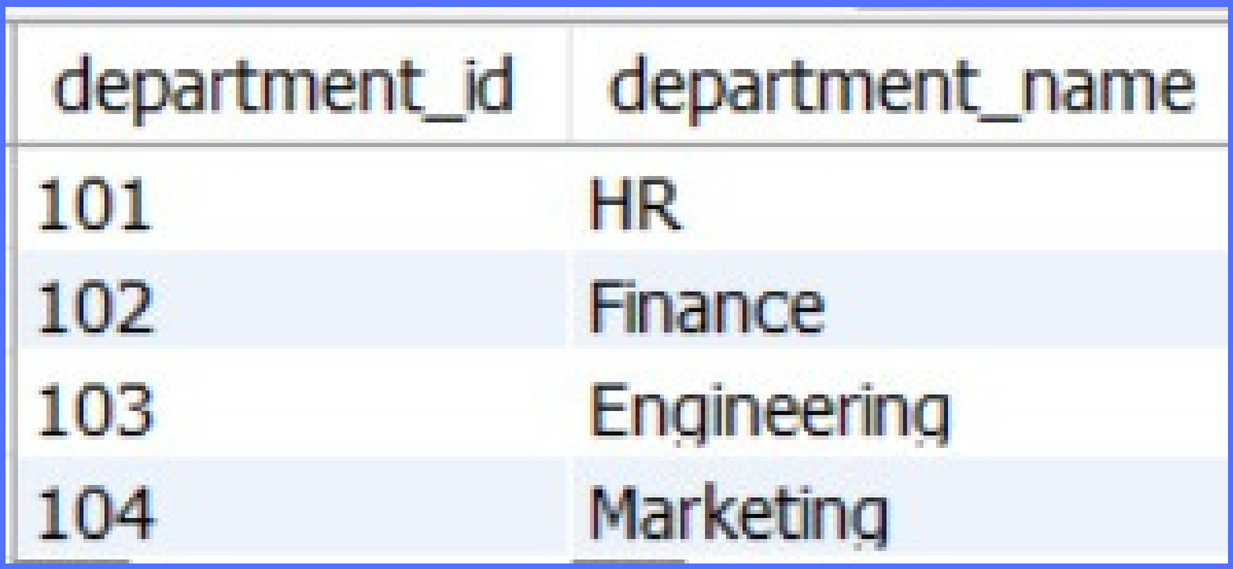
```sql
SELECT employees.first_name, departments.department_name
FROM employees
LEFT JOIN departments ON employees.department_id = departments.department_id;
```

LEFT JOIN – ALL ROWS FROM THE LEFT TABLE



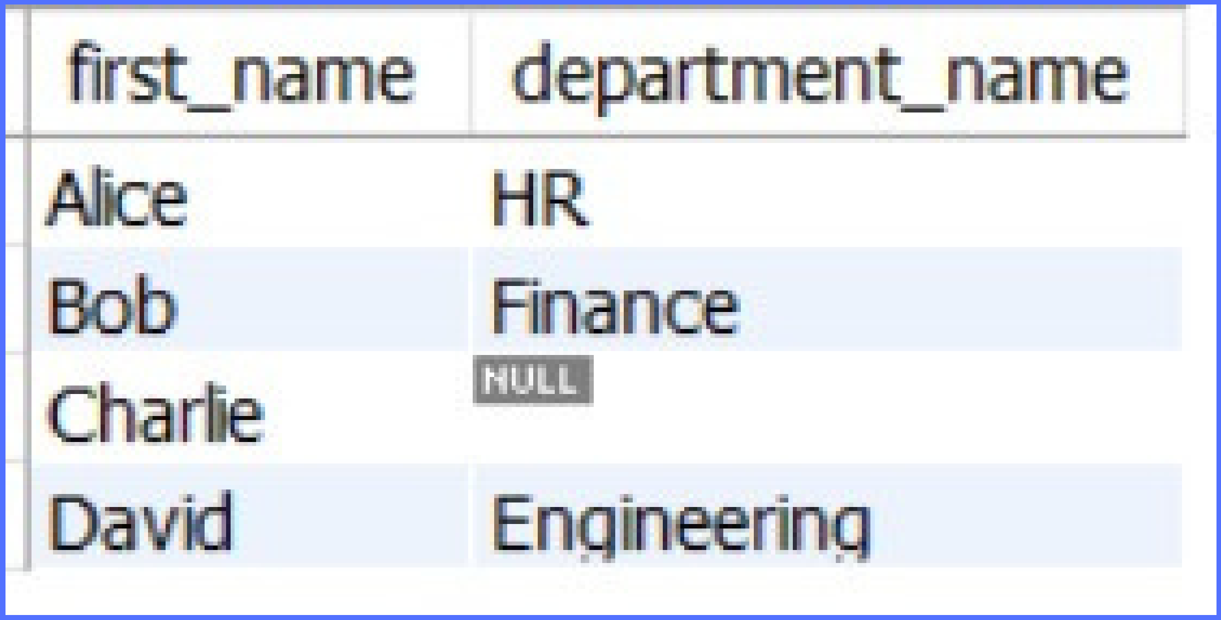
employee_id	first_name	department_id
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103

+



department_id	department_name
101	HR
102	Finance
103	Engineering
104	Marketing

=

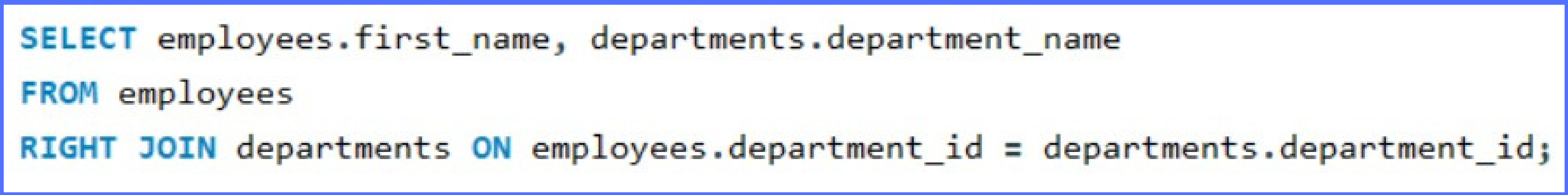


first_name	department_name
Alice	HR
Bob	Finance
Charlie	NULL
David	Engineering

RIGHT JOIN – ALL ROWS FROM THE RIGHT TABLE

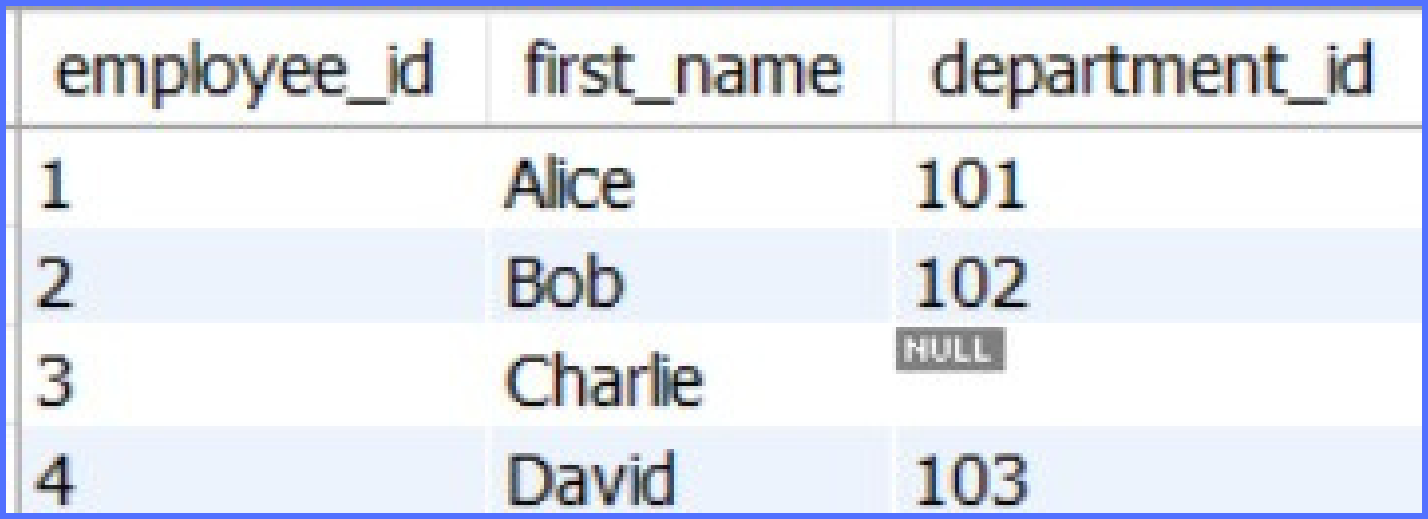
A RIGHT JOIN works similarly to a LEFT JOIN, but it returns all rows from the right table and the matching rows from the left. If there's no match, the result includes all rows from the right, with NULLs from the left table for missing matches.

- Use Case:
- You want a list of all departments, even if some departments have no employees. The result will include departments without employees, with NULL in the employee columns.



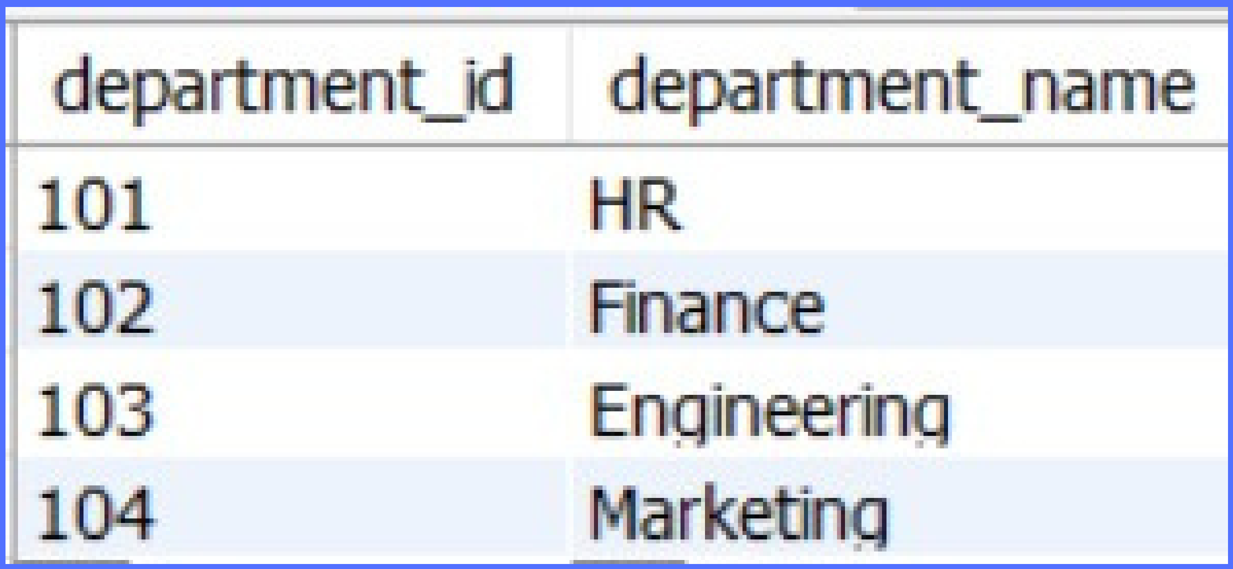
```sql
SELECT employees.first_name, departments.department_name
FROM employees
RIGHT JOIN departments ON employees.department_id = departments.department_id;
```

RIGHT JOIN – ALL ROWS FROM THE RIGHT TABLE



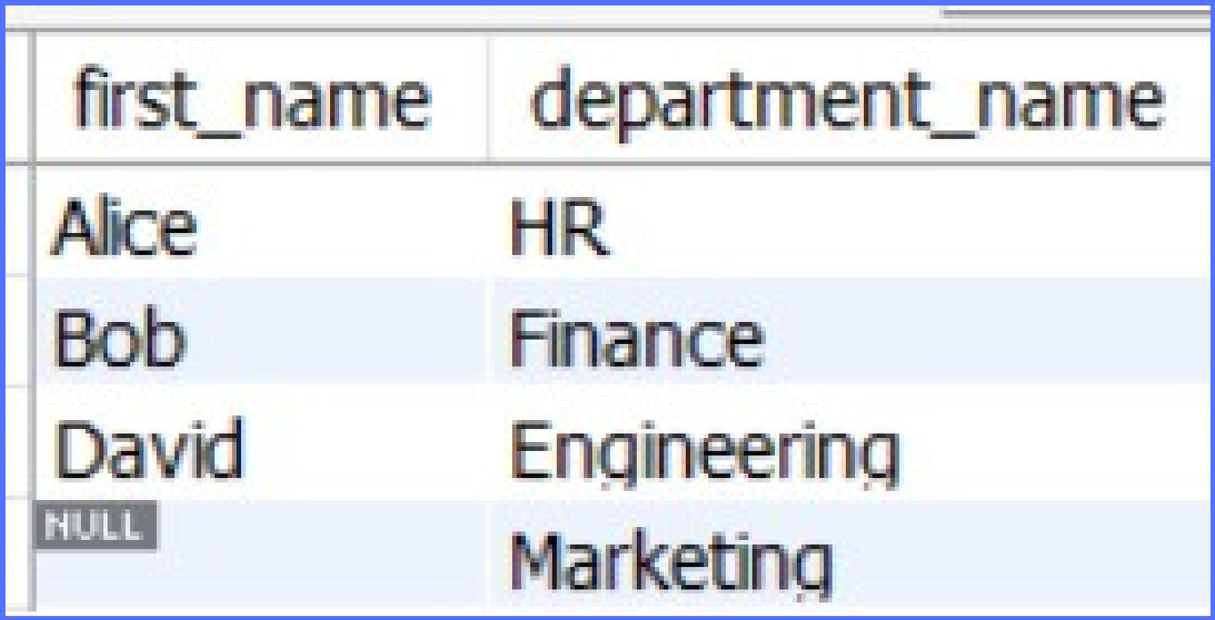
employee_id	first_name	department_id
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103

+



department_id	department_name
101	HR
102	Finance
103	Engineering
104	Marketing

=



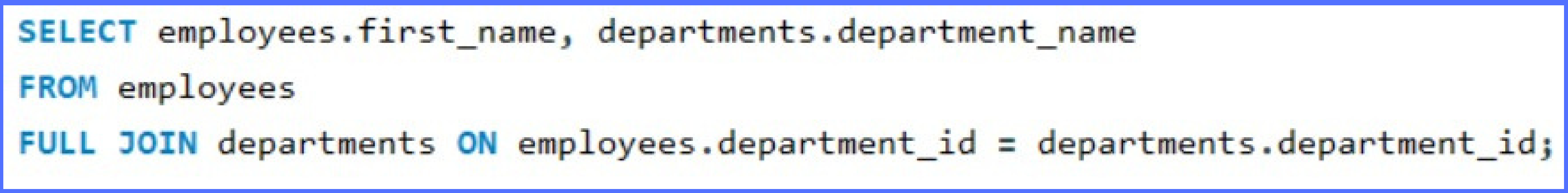
first_name	department_name
Alice	HR
Bob	Finance
David	Engineering
NULL	Marketing

FULL JOIN – ALL ROWS FROM BOTH TABLES

A FULL JOIN returns all rows from both tables. If there's a match, it combines the rows. If there's no match, it fills in NULLs for the missing parts.

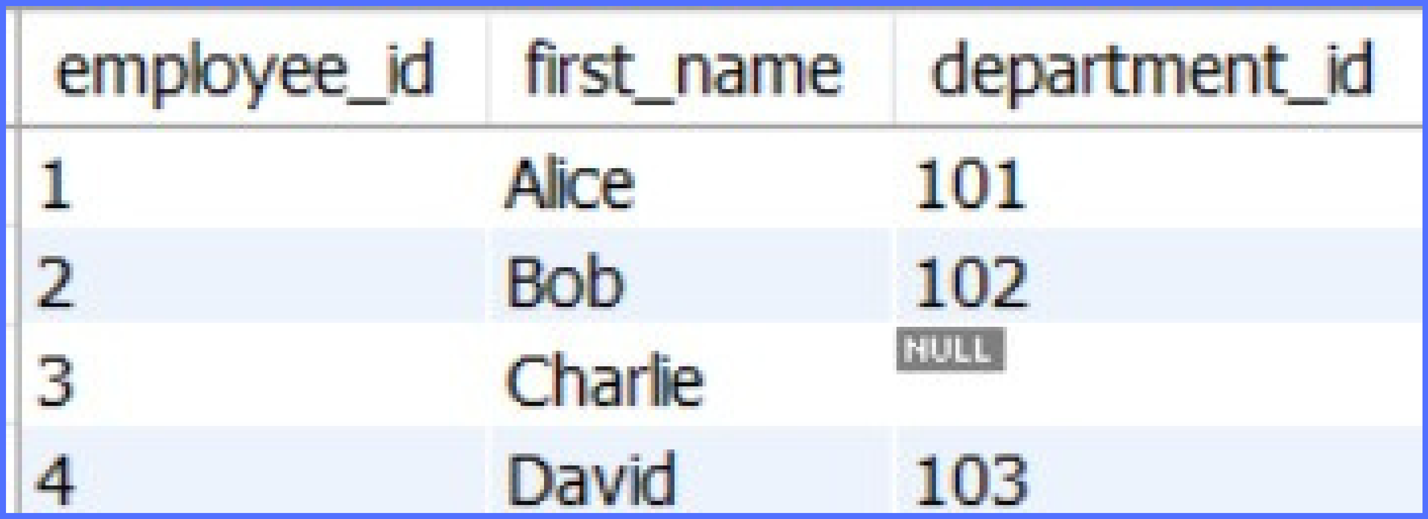
- Use Case:
- You want a complete list of employees and departments, showing which employees are assigned to departments and which employees or departments are unmatched.

NOTE: MySQL does not natively support the FULL OUTER JOIN clause. However, you can emulate it using a combination of LEFT JOIN, RIGHT JOIN and UNION.



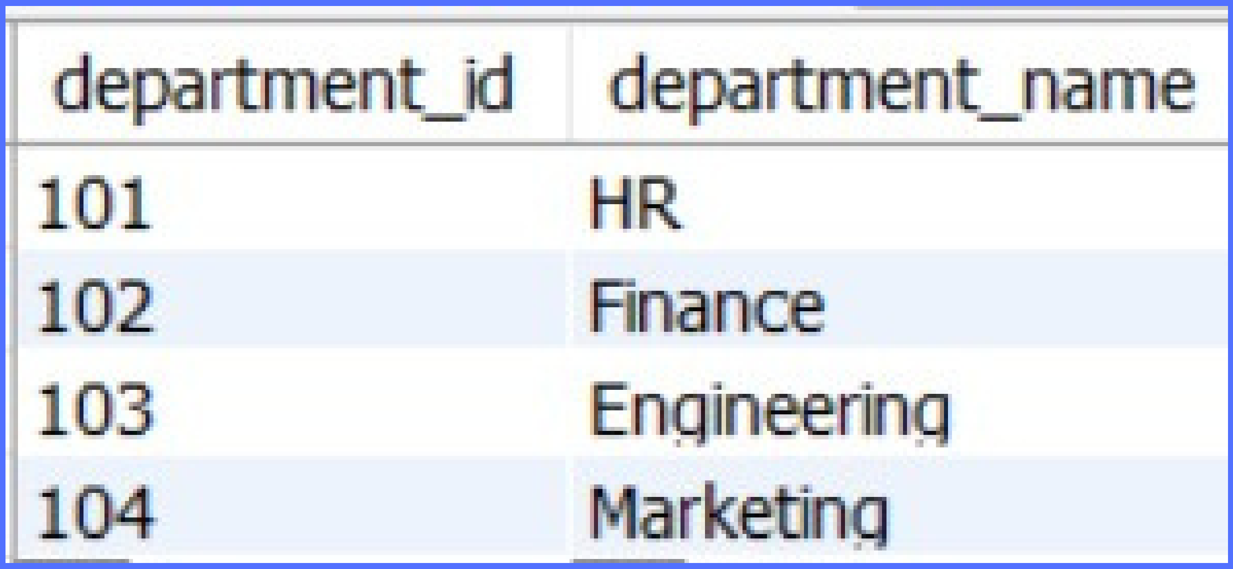
```
SELECT employees.first_name, departments.department_name
FROM employees
FULL JOIN departments ON employees.department_id = departments.department_id;
```

FULL JOIN – ALL ROWS FROM BOTH TABLES



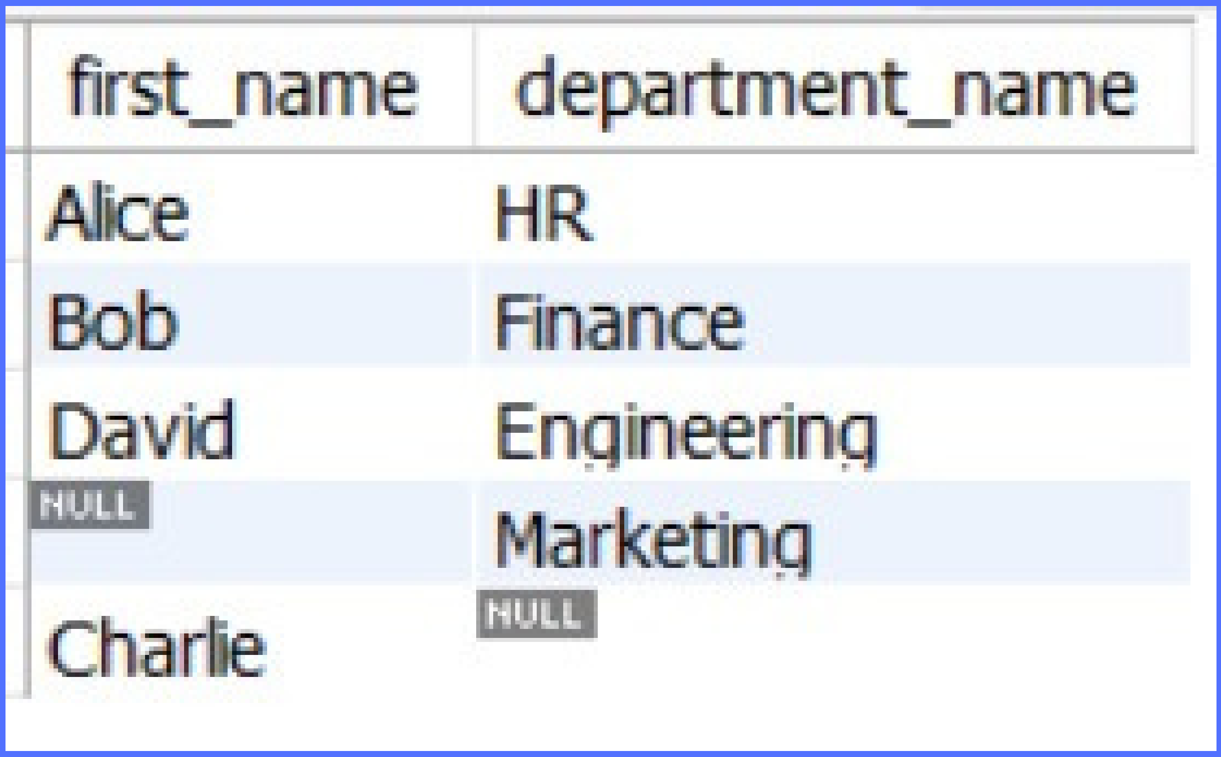
employee_id	first_name	department_id
1	Alice	101
2	Bob	102
3	Charlie	NULL
4	David	103

+



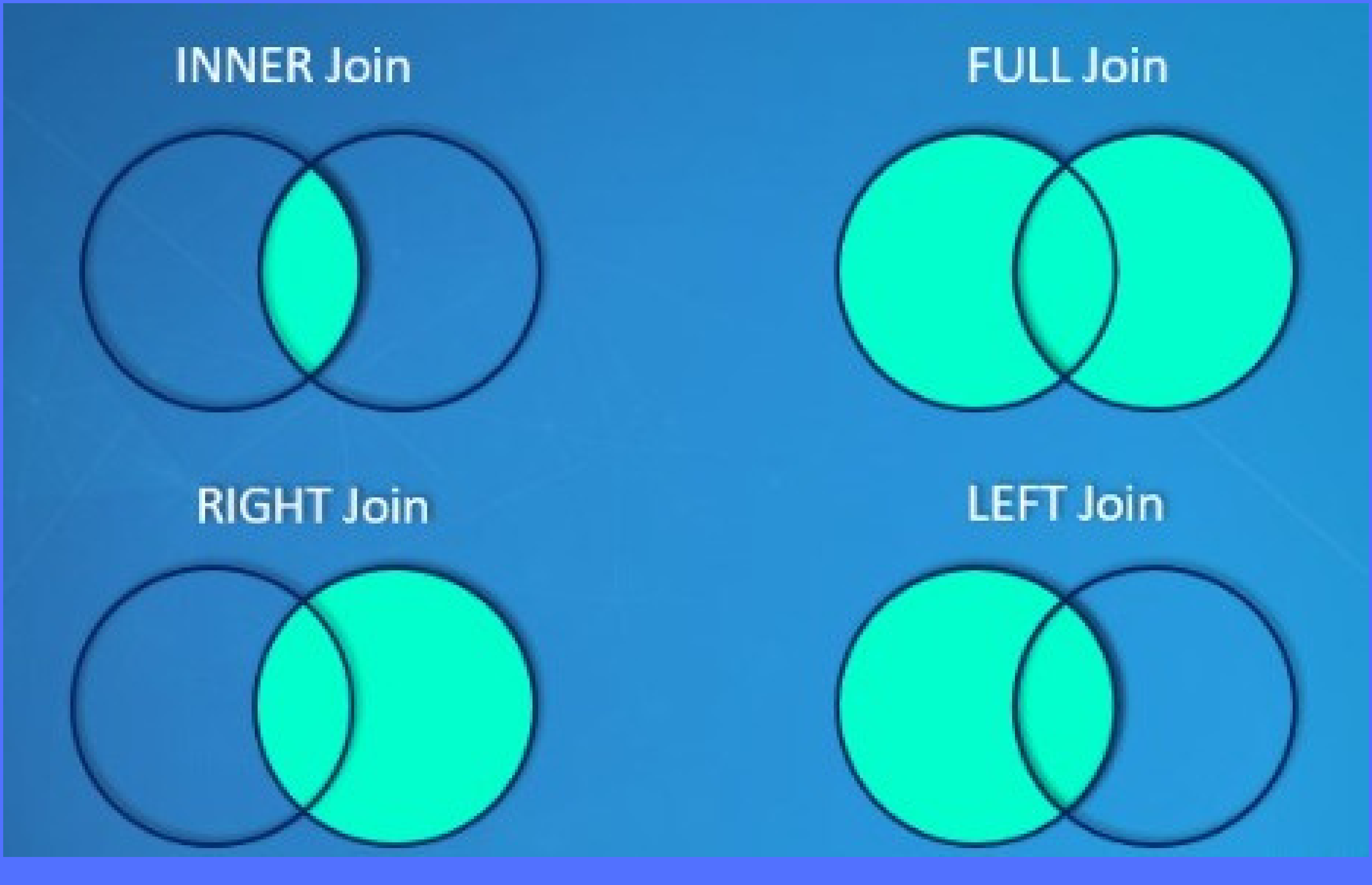
department_id	department_name
101	HR
102	Finance
103	Engineering
104	Marketing

=



first_name	department_name
Alice	HR
Bob	Finance
David	Engineering
NULL	Marketing
Charlie	NULL

VISUALIZING SQL JOINS



“Mastering SQL joins is essential for effective data analysis and decision-making. The more you practice, the more proficient you'll become in combining data from multiple tables to extract meaningful insights.”