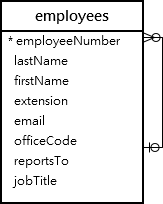
MySQL Self Join

The self join is often used to query hierarchical data or to compare a row with other rows within the same table.

To perform a self join, you must use table aliases to not repeat the same table name twice in a single query. Note that referencing a table twice or more in a query without using table aliases will cause an error.

## **MySQL self join examples**

Let’s take a look at the employees table.



The table employees stores not only employees data but also the organization structure data. The reportsto column is used to determine the manager id of an employee.

### **1) MySQL self join using INNER JOIN clause**

To get the whole organization structure, you can join the employees table to itself using the employeeNumber and reportsTo columns. The table employees has two roles: one is the Manager and the other is Direct Reports.

SELECT

CONCAT(m.lastName, ', ', m.firstName) AS Manager,

CONCAT(e.lastName, ', ', e.firstName) AS 'Direct report'

FROM

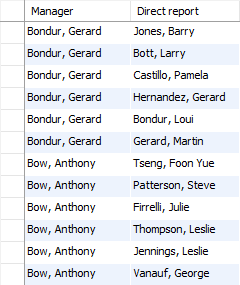
employees e

INNER JOIN employees m ON

m.employeeNumber = e.reportsTo

ORDER BY

Manager;



The output only shows the employees who have a manager. However, you don’t see the President because his name is filtered out due to the INNER JOIN clause.

### **2) MySQL self join using LEFT JOIN clause**

The President is the employee who does not have any manager or value in the reportsTo column is NULL .

The following statement uses the LEFT JOIN clause instead of INNER JOIN to include the President:

SELECT

IFNULL(CONCAT(m.lastname, ', ', m.firstname),

'Top Manager') AS 'Manager',

CONCAT(e.lastname, ', ', e.firstname) AS 'Direct report'

FROM

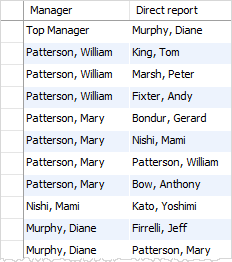
employees e

LEFT JOIN employees m ON

m.employeeNumber = e.reportsto

ORDER BY

manager DESC;



### **3) Using MySQL self join to compare successive rows**

By using the MySQL self join, you can display a list of customers who locate in the same city by joining the customers table to itself.

SELECT

c1.city,

c1.customerName,

c2.customerName

FROM

customers c1

INNER JOIN customers c2 ON

c1.city = c2.city

AND c1.customername > c2.customerName

ORDER BY

c1.city;



In this example, the table customers is joined to itself using the following join conditions:

* c1.city = c2.city  makes sure that both customers have the same city.
* c1.customerName > c2.customerName ensures that no same customer is included.

**Self Join in mysql**

A self-join in MySQL (or any other relational database management system) is a type of join operation where a table is joined with itself.

In other words, it's a way to combine rows from the same table based on a related column within that same table.

Self-joins are particularly useful when you have hierarchical or nested data structures stored within a single table.

To perform a self-join, you typically need to use table aliases to differentiate between the two instances of the same table.

Here's the basic syntax for a self-join in MySQL:

SELECT

t1.column1,

t2.column2

FROM table\_name t1

JOIN table\_name t2 ON t1.related\_column = t2.related\_column

WHERE condition;

Here's a breakdown of the key components:

1. **table\_name**: This is the name of the table you are joining with itself.
2. **t1** and **t2**: These are table aliases (short names) assigned to the two instances of the same table to differentiate between them.
3. **related\_column**: This is the column in the table that establishes the relationship between the rows you want to join.
4. **condition**: This is an optional condition that specifies additional criteria for the join. It can be used to filter the rows that are included in the result set.

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(255),

manager\_id INT

);

INSERT INTO employees (employee\_id, employee\_name, manager\_id)

VALUES

(1, 'John Doe', 0), -- John Doe is the CEO (manager\_id 0)

(2, 'Jane Smith', 1), -- Jane Smith reports to John Doe

(3, 'Bob Johnson', 1), -- Bob Johnson also reports to John Doe

(4, 'Eva Davis', 2), -- Eva Davis reports to Jane Smith

(5, 'Michael Wilson', 2), -- Michael Wilson reports to Jane Smith

(6, 'Sarah White', 3), -- Sarah White reports to Bob Johnson

(7, 'David Lee', 3), -- David Lee reports to Bob Johnson

(8, 'Lisa Harris', 4), -- Lisa Harris reports to Eva Davis

(9, 'Paul Anderson', 4), -- Paul Anderson reports to Eva Davis

(10, 'Emily Clark', 5), -- Emily Clark reports to Michael Wilson

(11, 'Daniel Baker', 5), -- Daniel Baker reports to Michael Wilson

(12, 'Olivia Turner', 6), -- Olivia Turner reports to Sarah White

(13, 'James Hall', 6), -- James Hall reports to Sarah White

(14, 'Ava Scott', 7), -- Ava Scott reports to David Lee

(15, 'William Green', 7), -- William Green reports to David Lee

(16, 'Sophia Adams', 8), -- Sophia Adams reports to Lisa Harris

(17, 'Joseph King', 8), -- Joseph King reports to Lisa Harris

(18, 'Charlotte Taylor', 9), -- Charlotte Taylor reports to Paul Anderson

(19, 'Benjamin Martinez', 9), -- Benjamin Martinez reports to Paul Anderson

(20, 'Chloe Rodriguez', 10); -- Chloe Rodriguez reports to Emily Clark

INSERT INTO employees (employee\_id, employee\_name, manager\_id)

VALUES

(21, 'Liam Smith', 10), -- Liam Smith reports to Emily Clark

(22, 'Mia Martinez', 10), -- Mia Martinez reports to Emily Clark

(23, 'Noah Wilson', 11), -- Noah Wilson reports to Daniel Baker

(24, 'Ava Davis', 11), -- Ava Davis reports to Daniel Baker

(25, 'Oliver Miller', 12), -- Oliver Miller reports to Olivia Turner

(26, 'Emma Turner', 12), -- Emma Turner reports to Olivia Turner

(27, 'Lucas White', 13), -- Lucas White reports to James Hall

(28, 'Harper Harris', 13), -- Harper Harris reports to James Hall

(29, 'Matthew Davis', 14), -- Matthew Davis reports to Ava Scott

(30, 'Abigail Wilson', 14), -- Abigail Wilson reports to Ava Scott

(31, 'Amelia Martin', 15), -- Amelia Martin reports to William Green

(32, 'Elijah Taylor', 15), -- Elijah Taylor reports to William Green

(33, 'Evelyn King', 16), -- Evelyn King reports to Sophia Adams

(34, 'Logan Baker', 16), -- Logan Baker reports to Sophia Adams

(35, 'Grace Hall', 17), -- Grace Hall reports to Joseph King

(36, 'Jackson Adams', 17), -- Jackson Adams reports to Joseph King

(37, 'Sofia Clark', 18), -- Sofia Clark reports to Charlotte Taylor

(38, 'Samuel Turner', 18), -- Samuel Turner reports to Charlotte Taylor

(39, 'Lily Miller', 19), -- Lily Miller reports to Benjamin Martinez

(40, 'Henry Brown', 19); -- Henry Brown reports to Benjamin Martinez

Here's a simple example to illustrate a self-join:

Let's say you have a table called **employees** with the following columns: **employee\_id**, **employee\_name**, and **manager\_id**.

In this case, you could use a self-join to find the employees and their respective managers

SELECT

e1.employee\_name AS employee,

e2.employee\_name AS manager

FROM employees e1

LEFT JOIN employees e2 ON e1.manager\_id = e2.employee\_id;

drop table departments;

drop table managers;

drop table projects;

drop table employees;

CREATE TABLE departments (

department\_id INT PRIMARY KEY,

department\_name VARCHAR(255) NOT NULL

);

CREATE TABLE managers (

manager\_id INT PRIMARY KEY,

manager\_name VARCHAR(255) NOT NULL

);

CREATE TABLE projects (

project\_id INT PRIMARY KEY,

project\_name VARCHAR(255) NOT NULL

);

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(255) NOT NULL,

manager\_id INT,

department\_id INT,

project\_id INT,

FOREIGN KEY (manager\_id) REFERENCES employees(employee\_id),

FOREIGN KEY (department\_id) REFERENCES departments(department\_id),

FOREIGN KEY (project\_id) REFERENCES projects(project\_id)

);

-- Insert 20 rows into the 'managers' table with example names

INSERT INTO managers (manager\_id, manager\_name)

VALUES

(1, 'John Smith'),

(2, 'Jane Johnson'),

(3, 'Robert Brown'),

(4, 'Emily Davis'),

(5, 'Michael Wilson'),

(6, 'Sarah White'),

(7, 'David Lee'),

(8, 'Lisa Harris'),

(9, 'Paul Anderson'),

(10, 'Emma Turner'),

(11, 'Olivia Turner'),

(12, 'James Hall'),

(13, 'Ava Scott'),

(14, 'William Green'),

(15, 'Sophia Adams'),

(16, 'Joseph King'),

(17, 'Charlotte Taylor'),

(18, 'Benjamin Martinez'),

(19, 'Chloe Rodriguez'),

(20, 'Elijah Taylor');

-- Insert 20 rows into the 'departments' table

INSERT INTO departments (department\_id, department\_name)

VALUES

(101, 'HR Department'),

(102, 'Finance Department'),

(103, 'Marketing Department'),

(104, 'Sales Department'),

(105, 'IT Department'),

(106, 'Operations Department'),

(107, 'Research Department'),

(108, 'Customer Support Department'),

(109, 'Legal Department'),

(110, 'Product Development Department'),

(111, 'Quality Assurance Department'),

(112, 'Supply Chain Department'),

(113, 'Public Relations Department'),

(114, 'Administration Department'),

(115, 'Facilities Management Department'),

(116, 'Training and Development Department'),

(117, 'Internal Audit Department'),

(118, 'Safety and Security Department'),

(119, 'Purchasing Department'),

(120, 'Shipping and Receiving Department');

-- Insert 20 rows into the 'projects' table with example names

INSERT INTO projects (project\_id, project\_name)

VALUES

(1001, 'Marketing Campaign for Product Launch'),

(1002, 'Website Redesign and Optimization'),

(1003, 'Customer Engagement Survey'),

(1004, 'New Product Development: Phase 1'),

(1005, 'Sales Training Program'),

(1006, 'Operations Efficiency Improvement'),

(1007, 'Market Research for Expansion Strategy'),

(1008, 'Customer Support Chatbot Implementation'),

(1009, 'Legal Compliance Audit'),

(1010, 'Mobile App Development for Android'),

(1011, 'Product Packaging Redesign'),

(1012, 'Quality Control Process Optimization'),

(1013, 'Supply Chain Management Overhaul'),

(1014, 'Public Relations Campaign'),

(1015, 'Office Space Renovation Project'),

(1016, 'Employee Onboarding and Training'),

(1017, 'Financial Reporting System Upgrade'),

(1018, 'Safety and Security Assessment'),

(1019, 'Vendor Evaluation and Selection'),

(1020, 'Shipping Process Streamlining');

-- Insert 20 rows into the 'employees' table

INSERT INTO employees (employee\_id, employee\_name, manager\_id, department\_id, project\_id)

VALUES

(1, 'John Doe', 1, 101, 1001),

(2, 'Jane Smith', 1, 101, 1001),

(3, 'Robert Johnson', 2, 102, 1002),

(4, 'Emily Brown', 2, 102, 1002),

(5, 'Michael Wilson', 3, 103, 1003),

(6, 'Sarah White', 3, 103, 1003),

(7, 'David Lee', 4, 104, 1004),

(8, 'Lisa Harris', 4, 104, 1004),

(9, 'Paul Anderson', 5, 105, 1005),

(10, 'Emma Turner', 5, 105, 1005),

(11, 'Olivia Turner', 6, 106, 1006),

(12, 'James Hall', 6, 106, 1006),

(13, 'Ava Scott', 7, 107, 1007),

(14, 'William Green', 7, 107, 1007),

(15, 'Sophia Adams', 8, 108, 1008),

(16, 'Joseph King', 8, 108, 1008),

(17, 'Charlotte Taylor', 9, 109, 1009),

(18, 'Benjamin Martinez', 9, 109, 1009),

(19, 'Chloe Rodriguez', 10, 110, 1010),

(20, 'Elijah Taylor', 10, 110, 1010);

-- 1. Find Employees and Their Managers:

-- This query retrieves a list of employees along with their respective managers.

SELECT

e.employee\_name AS employee,

m.employee\_name AS manager

FROM employees e

LEFT JOIN employees m ON e.manager\_id = m.employee\_id;

-- 2. List Employees and Their Colleagues in the Same Department:

-- This query finds employees and their colleagues (employees in the same department but excluding themselves).

SELECT e.employee\_name AS employee, c.employee\_name AS colleague

FROM employees e

JOIN employees c ON e.department\_id = c.department\_id

WHERE e.employee\_id <> c.employee\_id;

-- 3. Retrieve Employees with No Direct Reports:

-- This query identifies employees who do not manage

-- any other employees (i.e., employees with no direct reports).

SELECT e.employee\_name

FROM employees e

LEFT JOIN employees m ON e.employee\_id = m.manager\_id

WHERE m.manager\_id IS NULL;