UnCodemy

SQL Joins Examples PDF

A SQL join combines rows from two tables using a matching column. Use INNER JOIN for rows that match in both tables, LEFT JOIN to keep every row from the left table, and a LEFT JOIN with a NULL test to find rows with no match. This page shows each pattern on small tables you can run yourself.

Key takeaways

- INNER JOIN keeps only rows that match in both tables.
- LEFT JOIN keeps all rows from the left table and fills missing matches with NULL.
- A LEFT JOIN with WHERE right.id IS NULL finds rows that have no match.
- Joining on a non-unique key multiplies rows, so check counts after every join.
- Put filters on the right table in the ON clause when you must keep unmatched left rows.

What is SQL joins examples?

A SQL join is an operation that combines rows from two or more tables based on a related column, such as a customer ID that appears in both a customers table and an orders table.

What tables do the examples use?

The examples use four small tables: `customers(customer_id, name, city, signup_date)`, `orders(order_id, customer_id, order_date, status, order_total)`, `order_items(order_id, product_id, quantity, unit_price)` and `products(product_id, product_name, category)`. The data is made up. Customer 4 (Dev) has no orders, and order 3 is cancelled.

How does an inner join work?

An inner join returns only rows where the key exists in both tables.

```
SELECT c.name, o.order_id, o.order_total
FROM customers c
INNER JOIN orders o ON o.customer_id = c.customer_id
ORDER BY o.order_id;
```

Dev does not appear because he has no order. Use an inner join when you only care about matched rows.

How does a left join work?

A left join keeps every row of the left table.

```
SELECT c.name, o.order_id, o.order_total
FROM customers c
LEFT JOIN orders o ON o.customer_id = c.customer_id
ORDER BY c.customer_id, o.order_id;
```

Dev now appears with NULL in the order columns. Use it when the left table is your main list.

How do I find rows with no match?

Add a NULL test on the right key. This is called an anti join.

```
SELECT c.name
FROM customers c
LEFT JOIN orders o ON o.customer_id = c.customer_id
WHERE o.order_id IS NULL;
```

This returns customers who never ordered. `NOT EXISTS` gives the same result and is often clearer.

How do I filter without losing unmatched rows?

A condition on the right table in WHERE turns a left join into an inner join, because NULL fails the test. Put the condition in ON instead.

```
SELECT c.name, COUNT(o.order_id) AS delivered_orders
FROM customers c
LEFT JOIN orders o
  ON o.customer_id = c.customer_id AND o.status = 'delivered'
GROUP BY c.name
ORDER BY c.name;
```

Dev shows 0 delivered orders instead of disappearing.

How do I join three or more tables?

Chain the joins and keep the aliases short.

```
SELECT c.name, p.product_name, oi.quantity
FROM customers c
JOIN orders o ON o.customer_id = c.customer_id
JOIN order_items oi ON oi.order_id = o.order_id
JOIN products p ON p.product_id = oi.product_id
ORDER BY c.name, p.product_name;
```

What is a self join?

A self join joins a table to itself, useful for pairs and hierarchies. This example finds pairs of customers in the same city.

```
SELECT a.name AS customer_a, b.name AS customer_b, a.city
FROM customers a
JOIN customers b ON a.city = b.city AND a.customer_id < b.customer_id;
```

The `<` condition avoids listing a pair twice or pairing a person with themselves.

What about right and full joins?

A right join is a left join with the tables swapped, so most people use left joins only. A full join keeps unmatched rows from both sides; some databases, including MySQL, do not support it directly and need a UNION of two left joins.

What are the common mistakes?

Joining on the wrong column, forgetting that a one-to-many join repeats the parent row, filtering in WHERE after a left join, using SELECT * and getting duplicate column names, and not checking row counts. A quick habit is to run `COUNT(*)` before and after each join and explain any change.

How do I decide which join to use?

Ask which table is your main list. If you need all rows from it, use a left join. If you only want rows that exist in both tables, use an inner join. If you want rows with no match, use a left join and test for NULL. When in doubt, run both on a small sample and compare the counts; the difference shows you what each join keeps or drops.

How do I check a join result?

Count the rows in each table, join them and count again. For a one-to-many join the row count should match the many side. If it is larger, a key is duplicated. Also check that totals still match: the sum of order amounts after a join should equal the sum in the orders table, unless you filtered rows on purpose. These two checks take a minute and prevent most wrong reports.

What about performance?

Join on indexed key columns, select only the columns you need, filter early and avoid joining on expressions such as functions applied to a column, which can stop the database from using an index. Read the query plan when a join is slow.

Frequently asked questions

What is the difference between inner join and left join?

An inner join keeps only rows that match in both tables, while a left join keeps all rows from the left table and fills missing matches with NULL.

How do I find records with no match?

Use a left join and test the right key for NULL, or use NOT EXISTS. Both return rows from the left table with no partner.

Why did my left join lose rows?

A WHERE condition on the right table removes NULL rows. Move that condition into the ON clause to keep unmatched rows.

Why do I get duplicate rows after a join?

The key is probably not unique on one side, so each match repeats the other row. Check uniqueness and counts.

Does MySQL support full outer join?

Not directly. Combine a left join and a right join with UNION to get the same result.

Sources and further reading

- SQLite documentation
- MySQL reference manual
- PostgreSQL documentation

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550