

SQL JOINS

Column colors show where the data comes from.

Lavender: data from invites

Roast: data from arrivals

L = Lavender = Left · R = Roast = Right

Explained with a party example

A free visual guide from SQL Sip · Cozy, bite-sized SQL practice · sqlsip.com



1 THE PARTY

It's **Bear-ista's** birthday party!

Maya and **Sam** came to celebrate.

Leo was invited, but couldn't make it.

Penny, the postal carrier, stopped by with Bear-ista's birthday mail and joined the celebration.

invites table
People Bear-ista invited

guest_name	favorite_drink
Maya	Iced Chai
Sam	Cortado
Leo	Cold Brew

arrivals table
Who attended and which treat they chose

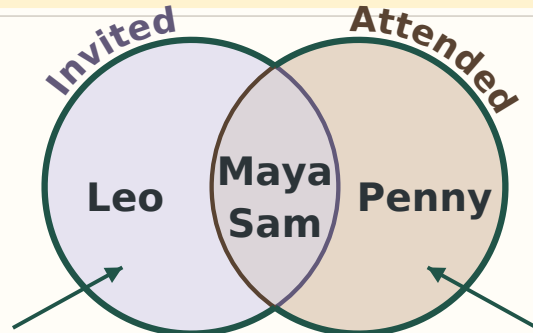
guest_name	treat_chosen
Maya	Cupcake
Sam	Brownie
Penny	Cookie

2 FULL OUTER JOIN

Bear-ista asks:

"Who was **invited** or **attended**, and what do we know about their **drinks** and **treats**?"

invites = who was invited · **arrivals** = who attended



Keeps both whole circles

Everyone on either list.

```
SELECT invites.guest_name AS invited_guest,
       invites.favorite_drink,
       arrivals.guest_name AS arrived_guest,
       arrivals.treat_chosen
FROM invites
FULL OUTER JOIN arrivals
  ON invites.guest_name = arrivals.guest_name;
```

Results · 4 rows

invited_guest	favorite_drink	arrived_guest	treat_chosen
Maya	Iced Chai	Maya	Cupcake
Sam	Cortado	Sam	Brownie
Leo	Cold Brew	NULL	NULL
NULL	NULL	Penny	Cookie

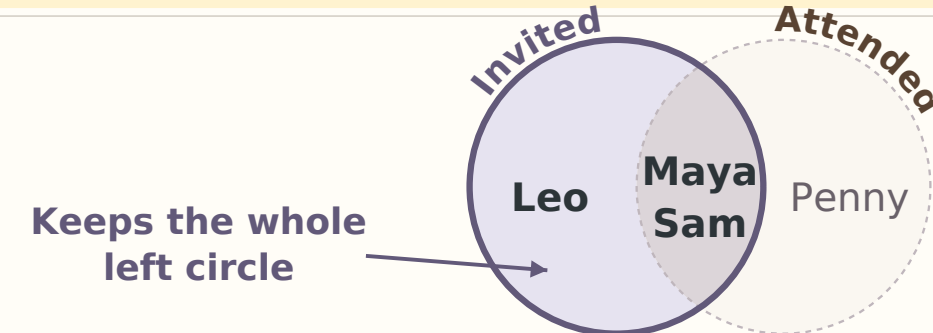
Why NULL? Leo has no matching row in **arrivals**. Penny has no matching row in **invites**.

3 LEFT JOIN

Bear-ista asks:

"Who did I **invite**, and what **treat** did they choose if they attended?"

invites = who was invited · **arrivals** = who attended



Keeps the whole left circle

Everyone who was invited, whether they attended or not.

```
SELECT invites.guest_name AS invited_guest,
       invites.favorite_drink,
       arrivals.guest_name AS arrived_guest,
       arrivals.treat_chosen
FROM invites
LEFT JOIN arrivals
  ON invites.guest_name = arrivals.guest_name;
```

Results · 3 rows

invited_guest	favorite_drink	arrived_guest	treat_chosen
Maya	Iced Chai	Maya	Cupcake
Sam	Cortado	Sam	Brownie
Leo	Cold Brew	NULL	NULL

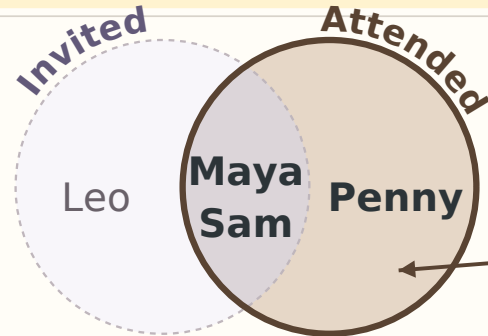
Why NULL? Leo was invited but couldn't attend, so no matching arrival row supplied his arrival values.

4 RIGHT JOIN

Bear-ista asks:

"Who **attended**, which **treat** did they choose, and what **favorite drink** was recorded on their invitation?"

invites = who was invited · **arrivals** = who attended



Keeps the whole right circle

Everyone who attended, whether they were invited or not.

```
SELECT invites.guest_name AS invited_guest,
       invites.favorite_drink,
       arrivals.guest_name AS arrived_guest,
       arrivals.treat_chosen
FROM invites
RIGHT JOIN arrivals
  ON invites.guest_name = arrivals.guest_name;
```

Results · 3 rows

invited_guest	favorite_drink	arrived_guest	treat_chosen
Maya	Iced Chai	Maya	Cupcake
Sam	Cortado	Sam	Brownie
NULL	NULL	Penny	Cookie

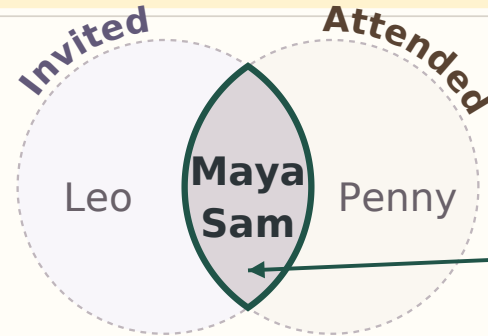
Why NULL? Penny attended but wasn't on the invitation list, so no matching invitation row supplied her invitation values.

5 INNER JOIN

Bear-ista asks:

"Which **invited** guests **attended**, and what **treats** did they choose?"

invites = who was invited · **arrivals** = who attended



Keeps only the overlap

Only invited guests who attended.

```
SELECT invites.guest_name AS invited_guest,
       invites.favorite_drink,
       arrivals.guest_name AS arrived_guest,
       arrivals.treat_chosen
FROM invites
INNER JOIN arrivals
  ON invites.guest_name = arrivals.guest_name;
```

Results · 2 rows

invited_guest	favorite_drink	arrived_guest	treat_chosen
Maya	Iced Chai	Maya	Cupcake
Sam	Cortado	Sam	Brownie

Only matching row pairs appear here: Maya and Sam each have an invitation and an arrival.

6 JOIN Tips & Tricks

INNER vs. **OUTER**

INNER keeps matching row pairs.

OUTER also keeps *unmatched* rows from the chosen side or sides.

WHAT COUNTS AS A MATCH?

In these queries, rows match when **invites.guest_name** equals **arrivals.guest_name**, as specified by the **ON** condition.

In these two-table queries:

FROM invites LEFT JOIN arrivals

Left table: invites - the table named after **FROM**.

Right table: arrivals - the table named after **JOIN**.

LEFT or RIGHT tells SQL which table's unmatched rows to keep.

SQL JOIN NAMES

LEFT, **RIGHT**, and **FULL** are **outer** joins. The keyword **OUTER** is optional - it does not change the result.

LEFT OUTER JOIN = LEFT JOIN

RIGHT OUTER JOIN = RIGHT JOIN

FULL OUTER JOIN = FULL JOIN

If no JOIN type is specified, **JOIN** means **INNER JOIN**.

DIALECT NOTE

MySQL has no native **FULL OUTER JOIN**. A workaround combines a **LEFT JOIN** with only the **unmatched right-side rows** using **UNION ALL**.

Guest names are unique in this simplified example. Real databases commonly match records using stable IDs, such as `guest_id`, rather than a name.

A row can match more than one row in the other table; each matching pair produces a result row. Use **ORDER BY** to guarantee row order.

Ready to practice? Play today's SQL puzzle → sqlsip.com

On a phone? Read the mobile guide at sqlsip.com/guides/joins

Stay curious, sip slowly, and know Bear-ista is beary proud of you for learning your JOINS.

There's always a seat for you at SQL Sip.

— Bear-ista 🐻

SQL Sip · sqlsip.com · Edition 1.0 (October 2026)

