Joins

SQL Reference

Keys & relationships

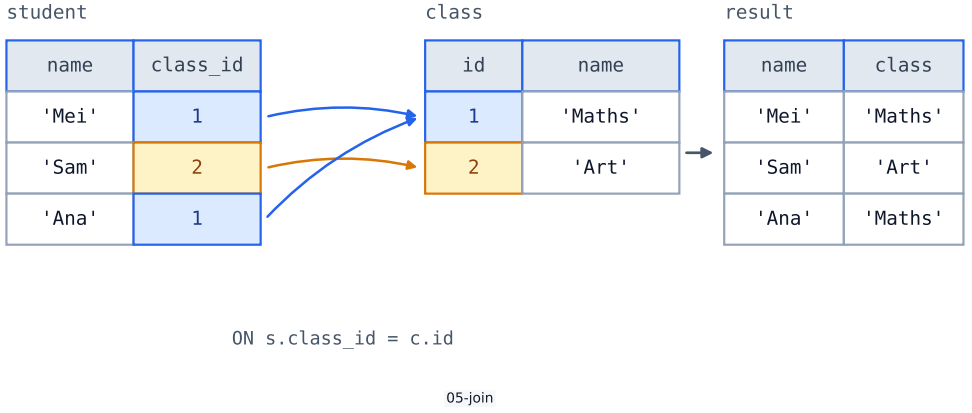
A primary key uniquely names each row in a table. A foreign key in one table points to the primary key of another – that builds a relationship between them.

```
CREATE TABLE class (id INTEGER PRIMARY KEY, name TEXT);
CREATE TABLE student (id INTEGER PRIMARY KEY, name TEXT, class_id
↪  INTEGER REFERENCES class(id));
INSERT INTO class VALUES (1, 'Maths'), (2, 'Art');
INSERT INTO student VALUES (1, 'Mei', 1), (2, 'Sam', 2);
SELECT * FROM student;
```

INNER JOIN

A join combines rows from two tables. `INNER JOIN ... ON ...` keeps rows where the keys match. A short alias (`s`, `c`) keeps the query readable.

```
CREATE TABLE class (id INTEGER PRIMARY KEY, name TEXT);
CREATE TABLE student (id INTEGER PRIMARY KEY, name TEXT, class_id
↪  INTEGER REFERENCES class(id));
INSERT INTO class VALUES (1, 'Maths'), (2, 'Art');
INSERT INTO student VALUES (1, 'Mei', 1), (2, 'Sam', 2);
SELECT s.name, c.name AS class
FROM student s INNER JOIN class c ON s.class_id = c.id;
```



INNER JOIN matches each foreign key to a primary key and combines the matched rows

Joining with grouping

Join first, then `GROUP BY` to summarise across the joined rows.

```
CREATE TABLE class (id INTEGER PRIMARY KEY, name TEXT);
CREATE TABLE student (id INTEGER PRIMARY KEY, name TEXT, class_id
↪  INTEGER REFERENCES class(id));
INSERT INTO class VALUES (1, 'Maths'), (2, 'Art');
INSERT INTO student VALUES (1, 'Mei', 1), (2, 'Sam', 2), (3, 'Ana', 1);
SELECT c.name AS class, COUNT(*) AS n
FROM student s INNER JOIN class c ON s.class_id = c.id
GROUP BY c.id, c.name;
```

Group by the key `c.id` as well as the name: two classes that happen to share a name would otherwise be merged into one row.

LEFT JOIN

An INNER JOIN keeps only matched rows. A `LEFT JOIN` keeps **every** row of the left (first) table; where there is no match, the right table's columns come back as `NULL`.

```
CREATE TABLE class (id INTEGER PRIMARY KEY, name TEXT);
CREATE TABLE student (id INTEGER PRIMARY KEY, name TEXT, class_id
↪  INTEGER REFERENCES class(id));
INSERT INTO class VALUES (1, 'Maths'), (2, 'Art');
INSERT INTO student VALUES (1, 'Mei', 1), (2, 'Sam', NULL);
SELECT s.name, c.name AS class
FROM student s LEFT JOIN class c ON s.class_id = c.id;
```

- Sam has no class, but the row still appears – with `class` as `NULL`.
- To find only the unmatched rows, add `WHERE c.id IS NULL`.

Common mistakes

- A join with no `ON` condition pairs every row with every row (a cross join).
- Match the foreign key to the primary key: `ON orders.customer_id = customers.id`.
- An INNER JOIN drops rows that have no match on the other side; use a LEFT JOIN to keep them.