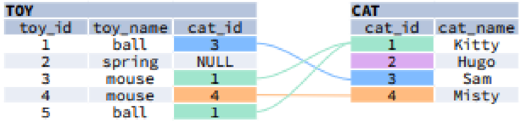
# SQL JOINs Cheat Sheet

LearnSQL.com

## JOINING TABLES

JOIN combines data from two tables.



TOY

| toy_id | toy_name | cat_id |
|---|---|---|
| 1 | ball | 3 |
| 2 | spring | NULL |
| 3 | mouse | 1 |
| 4 | mouse | 4 |
| 5 | ball | 1 |

CAT

| cat_id | cat_name |
|---|---|
| 1 | Kitty |
| 2 | Hugo |
| 3 | Sam |
| 4 | Misty |

JOIN typically combines rows with equal values for the specified columns. **Usually**, one table contains a **primary key**, which is a column or columns that uniquely identify rows in the table (the cat_id column in the cat table).
The other table has a column or columns that **refer to the primary key columns** in the first table (the cat_id column in the toy table). Such columns are **foreign keys**. The JOIN condition is the equality between the primary key columns in one table and columns referring to them in the other table.

## JOIN

JOIN returns all rows that match the ON condition. JOIN is also called INNER JOIN.

```
SELECT *
FROM toy
JOIN cat
  ON toy.cat_id = cat.cat_id;
```

| toy_id | toy_name | cat_id | cat_id | cat_name |
|---|---|---|---|---|
| 5 | ball | 1 | 1 | Kitty |
| 3 | mouse | 1 | 1 | Kitty |
| 1 | ball | 3 | 3 | Sam |
| 4 | mouse | 4 | 4 | Misty |

There is also another, older syntax, but it **isn't recommended**.
List joined tables in the FROM clause, and place the conditions in the WHERE clause.

```
SELECT *
FROM toy, cat
WHERE toy.cat_id = cat.cat_id;
```

## JOIN CONDITIONS

The JOIN condition doesn't have to be an equality – it can be any condition you want. JOIN doesn't interpret the JOIN condition, it only checks if the rows satisfy the given condition.

To refer to a column in the JOIN query, you have to use the full column name: first the table name, then a dot (.) and the column name:

```
  ON cat.cat_id = toy.cat_id
```

You can omit the table name and use just the column name if the name of the column is unique within all columns in the joined tables.

## NATURAL JOIN

If the tables have columns with **the same name**, you can use NATURAL JOIN instead of JOIN.

```
SELECT *
FROM toy
NATURAL JOIN cat;
```

| cat_id | toy_id | toy_name | cat_name |
|---|---|---|---|
| 1 | 5 | ball | Kitty |
| 1 | 3 | mouse | Kitty |
| 3 | 1 | ball | Sam |
| 4 | 4 | mouse | Misty |

The common column appears only once in the result table.
**Note:** NATURAL JOIN is rarely used in real life.

## LEFT JOIN

LEFT JOIN returns all rows from the **left table** with matching rows from the right table. Rows without a match are filled with NULLs. LEFT JOIN is also called LEFT OUTER JOIN.

```
SELECT *
FROM toy
LEFT JOIN cat
  ON toy.cat_id = cat.cat_id;
```

| toy_id | toy_name | cat_id | cat_id | cat_name |
|---|---|---|---|---|
| 5 | ball | 1 | 1 | Kitty |
| 3 | mouse | 1 | 1 | Kitty |
| 1 | ball | 3 | 3 | Sam |
| 4 | mouse | 4 | 4 | Misty |
| 2 | spring | NULL | NULL | NULL |

whole left table

## RIGHT JOIN

RIGHT JOIN returns all rows from the **right table** with matching rows from the left table. Rows without a match are filled with NULLs. RIGHT JOIN is also called RIGHT OUTER JOIN.

```
SELECT *
FROM toy
RIGHT JOIN cat
  ON toy.cat_id = cat.cat_id;
```

| toy_id | toy_name | cat_id | cat_id | cat_name |
|---|---|---|---|---|
| 5 | ball | 1 | 1 | Kitty |
| 3 | mouse | 1 | 1 | Kitty |
| NULL | NULL | NULL | 2 | Hugo |
| 1 | ball | 3 | 3 | Sam |
| 4 | mouse | 4 | 4 | Misty |

whole right table

## FULL JOIN

FULL JOIN returns all rows from the **left table** and all rows from the **right table**. It fills the non-matching rows with NULLs. FULL JOIN is also called FULL OUTER JOIN.

```
SELECT *
FROM toy
FULL JOIN cat
  ON toy.cat_id = cat.cat_id;
```

| toy_id | toy_name | cat_id | cat_id | cat_name |
|---|---|---|---|---|
| 5 | ball | 1 | 1 | Kitty |
| 3 | mouse | 1 | 1 | Kitty |
| NULL | NULL | NULL | 2 | Hugo |
| 1 | ball | 3 | 3 | Sam |
| 4 | mouse | 4 | 4 | Misty |
| 2 | spring | NULL | NULL | NULL |

whole left table whole right table

## CROSS JOIN

CROSS JOIN returns **all possible combinations** of rows from the left and right tables.

```
SELECT *
FROM toy
CROSS JOIN cat;
```

Other syntax:

```
SELECT *
FROM toy, cat;
```

| toy_id | toy_name | cat_id | cat_id | cat_name |
|---|---|---|---|---|
| 1 | ball | 3 | 1 | Kitty |
| 2 | spring | NULL | 1 | Kitty |
| 3 | mouse | 1 | 1 | Kitty |
| 4 | mouse | 4 | 1 | Kitty |
| 5 | ball | 1 | 1 | Kitty |
| 1 | ball | 3 | 2 | Hugo |
| 2 | spring | NULL | 2 | Hugo |
| 3 | mouse | 1 | 2 | Hugo |
| 4 | mouse | 4 | 2 | Hugo |
| 5 | ball | 1 | 2 | Hugo |
| 1 | ball | 3 | 3 | Sam |
| ··· | ··· | ··· | ··· | ··· |