SQL JOINs Cheat Sheet

JOINING TABLES
JOIN combines data from two tables.

<table>
<thead>
<tr>
<th>TOY</th>
<th>cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>toy_id</td>
<td>toy_name</td>
</tr>
<tr>
<td>1</td>
<td>ball</td>
</tr>
<tr>
<td>2</td>
<td>spring</td>
</tr>
<tr>
<td>3</td>
<td>mouse</td>
</tr>
<tr>
<td>4</td>
<td>mouse</td>
</tr>
<tr>
<td>5</td>
<td>ball</td>
</tr>
</tbody>
</table>

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;

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;

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;

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;

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;

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

SELECT * FROM toy CROSS JOIN cat;
SQL JOINs Cheat Sheet

COLUMN AND TABLE ALIASES
Aliases give a temporary name to a table or a column in a table.

A column alias renames a column in the result. A table alias renames a table within the query. If you define a table alias, you must use it instead of the table name everywhere in the query. The AS keyword is optional in defining aliases.

```
SELECT o.name AS owner_name, c.cat_name
FROM cat AS c
JOIN owner AS o
ON c.owner_id = o.id;
```

SELF JOIN
You can join a table to itself, for example, to show a parent-child relationship.

```
SELECT child.cat_name AS child_name, mom.cat_name AS mom_name
FROM cat AS child
JOIN cat AS mom
ON child.cat_id = mom.cat_id;
```

NON-EQUI SELF JOIN
You can use a non-equality in the ON condition, for example, to show all different pairs of rows.

```
SELECT a.toy_name AS toy_a, b.toy_name AS toy_b
FROM toy a
JOIN toy b
ON a.toy_id < b.toy_id;
```

MULTIPLE JOINS
You can join more than two tables together. First, two tables are joined, then the third table is joined to the result of the previous joining.

```
SELECT t.toy_name, c.cat_name, o.name AS owner_name
FROM toy t
JOIN cat c
ON t.cat_id = c.cat_id
JOIN owner o
ON c.owner_id = o.id;
```

JOIN WITH MULTIPLE CONDITIONS
You can use multiple JOIN conditions using the ON keyword once and the AND keywords as many times as you need.

```
SELECT cat_name, o.name AS owner_name, c.age AS cat_age, o.age AS owner_age
FROM cat c
JOIN owner o
ON c.owner_id = o.id
AND c.age < o.age;
```

JOIN & JOIN LEFT JOIN & LEFT JOIN
```
SELECT t.toy_name, c.cat_name, o.name AS owner_name
FROM toy t
JOIN cat c
ON t.cat_id = c.cat_id
LEFT JOIN owner o
ON c.owner_id = o.id;
```

JOIN & LEFT JOIN & LEFT JOIN
```
SELECT t.toy_name, c.cat_name, o.name AS owner_name
FROM toy t
LEFT JOIN cat c
ON t.cat_id = c.cat_id
LEFT JOIN owner o
ON c.owner_id = o.id;
```

Try out the interactive SQL JOINs course at LearnSQL.com, and check out our other SQL courses.