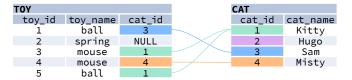
SQL JOINs Cheat Sheet



JOINING TABLES

JOIN combines data from two tables.



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 *	toy_id	toy_name	cat_id	cat_id	cat_name
FROM toy	5	ball	1	1	Kitty
JOIN cat	3	mouse	1	1	Kitty
ON toy.cat_id = cat.cat_id;	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;

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	
<pre>ON toy.cat_id = cat.cat_id;</pre>	

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 ri	ght 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.

SELEC	CT *
FROM	toy
FULL	JOIN cat
ON	<pre>toy.cat_id = cat.cat_id;</pre>

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 ri	ght table

3

. . .

CROSS JOIN

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

		-			
SELECT *	toy_id	toy_name	cat_id	cat_id	cat_name
FROM toy	1	ball	3	1	Kitty
CROSS JOIN cat;	2	spring	NULL	1	Kitty
,	3	mouse	1	1	Kitty
Other syntax:	4	mouse	4	1	Kitty
SELECT *	5	ball	1	1	Kitty
FROM toy, cat;	1	ball	3	2	Hugo
FROM LOY, Cal,	2	spring	NULL	2	Hugo
	3	mouse	1	2	Hugo
	4	mouse	4	2	Hugo
	5	ball	1	2	Hugo

1

. . .

ball

. . .

cat_id toy_id toy_name cat_name

ball

mouse

ball

mouse

Kitty

Kitty

Sam

Mistv

5

3

1

Δ

1 1

3

LearnSQL.com is owned by Vertabelo SA vertabelo.com | CC BY-NC-ND Vertabelo SA

. . .

Sam

. . .

SQL JOINs Cheat Sheet



COLUMN AND TABLE ALIASES

Aliases give a temporary name to a **table** or a **column** in a table.

CAT AS c				C	OWNER AS	0
cat_id	cat_name	mom_id	owner_id		id	name
1	Kitty	5	1	[]	1	John Smith
2	Hugo	1	2		2	Danielle Davis
3	Sam	2	2			
4	Misty	1	NULL			

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,
<pre>c.cat_name</pre>
FROM cat AS c
JOIN owner AS o
<pre>ON c.owner_id = o.id;</pre>

cat_name	owner_name
Kitty	John Smith
Sam	Danielle Davis
Hugo	Danielle Davis
0	

SELF JOIN

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

CAT AS C	hild				CAT AS m	om		
cat_id	cat_name	owner_id	mom_id		cat_id	cat_name	owner_id	mom_id
1	Kitty	1	5		1	Kitty	1	5
2	Hugo	2	1		2	Hugo	2	1
3	Sam	2	2	\vdash	3	Sam	2	2
4	Misty	NULL	1	\sim	4	Misty	NULL	1

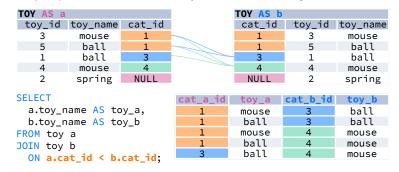
Each occurrence of the table must be given a **different alias**. Each column reference must be preceded with an **appropriate table alias**.

```
SELECT
```

<pre>child.cat_name AS child_name,</pre>	child_name	mom_name
<pre>mom.cat_name AS mom_name</pre>	Hugo	Kitty
FROM cat AS child	Sam	Hugo
JOIN cat AS mom	Misty	Kitty
<pre>ON child.mom_id = mom.cat_id;</pre>		

NON-EQUI SELF JOIN

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



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.

TOY AS t			ī						
	toy_name	cat id		CAT AS c				OWNE	RASO
toy_iu				cat_id	cat_name	mom_id	owner_id		
1	ball	3		1	Kitty	5	1	id	name
2	spring	NULL		2		1	2	1	John
3	mouse	1	$\sim X$	2	Hugo	T	2	-	Smith
5		-		3	Sam	2	2	2	Danielle
4	mouse	4		4	Mistv	1	NULL	~	Davis
5	ball	1		4	misty	1	NULL		

NIOL & NIOL			JC	DIN & LEFT .	JOIN	LEFT	LEFT JOIN & LEFT JOIN			
SELECT			SELECT			SELECT				
t.toy_name,			t.toy_n	ame,		t.toy_name,				
c.cat_n	ame,		c.cat_r	name,		c.cat_r	c.cat_name,			
o.name	AS owner	_name	o.name	AS owner	_name	o.name	o.name AS owner_name			
FROM toy	t		FROM toy	t		FROM toy	FROM toy t			
JOIN cat c			JOIN cat	с		LEFT JOIN cat c				
ON t.cat_id = c.cat_id			ON t.cat_id = c.cat_id			<pre>ON t.cat_id = c.cat_id</pre>				
JOIN owne	er o		LEFT JOIN owner o			LEFT JOIN	LEFT JOIN owner o			
<pre>ON c.owner_id = o.id;</pre>		<pre>ON c.owner_id = o.id;</pre>			ON c.owner_id = o.id;					
toy_name	cat_name	owner_name	toy_name	cat_name	owner_name	toy_name	cat_name	owner_name		
ball	Kitty	John Smith	ball	Kitty	John Smith	ball	Kitty	John Smith		
mouse	Kitty	John Smith	mouse Kitty John Smith		mouse	Kitty	John Smith			
ball	Sam	Danielle Davis	ball	Sam	Danielle Davis	ball	Sam	Danielle Davis		
			mouse	Misty	NULL	mouse	Misty	NULL		
						spring	NULL	NULL		

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.

AT AS c				OWNE	RASO		
cat_id	cat_name	mom_id	owner_id	age	id	name	age
1	Kitty	5	1	17	1	John Smith	18
2	Hugo	1	2	10	2	Danielle Davis	10
3	Sam	2	2	5			
4	Misty	1	NULL	11			

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;</pre>

cat_name	owner_name	age	age
Kitty	John Smith	17	18
Sam	Danielle Davis	5	10