# **SQL JOINs Cheat Sheet**

## LearnSQL • com

### **JOINING TABLES**

JOIN combines data from two tables.

TOY				CAT	
toy_id	toy_name	cat_id		cat_id	cat_name
1	ball	3	$\vdash$	1	Kitty
2	spring	NULL		2	Hugo
3	mouse	1		3	Sam
4	mouse	4		4	Misty
5	ball	1	$\vdash$		

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
<pre>ON toy.cat_id = cat.cat_id;</pre>	1	ball	3	3	Sam
on coy.eac_ia = cac.eac_ia,	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;

1	5	ball	Kitty
1	3	mouse	Kitty
3	1	ball	Sam
4	4	mouse	Misty

cat id toy id toy name cat name

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;
```

J	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.

```
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
L		whole left table	whole ri	ght 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
• • •	• • •	• • •	• • •	• • •

# **SQL JOINs Cheat Sheet**

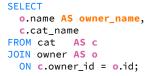
### LearnSQL • com

### **COLUMN AND TABLE ALIASES**

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

CAT AS 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	Mistv	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.



cat_name	owner_name
Kitty	John Smith
Sam	Danielle Davis
Hugo	Danielle Davis

### **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	3	Sam	2	2
4	Misty	NULL	1	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
child.cat_name AS child_name,
mom.cat_name AS mom_name
FROM cat AS child
JOIN cat AS mom
ON child.mom_id = mom.cat_id;
```

mom_name
Kitty
Hugo
Kitty

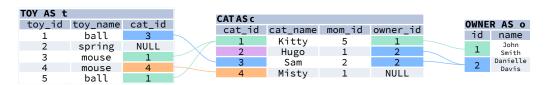
### **NON-EQUI SELF JOIN**

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

cat_id toy_id toy_name									
1 3 mouse									
1 5 ball									
3 1 ball									
4 4 mouse									
NULL 2 spring									
SELECT cat a id toy a cat b id toy b									
d toy_a cat_b_id toy_b									
mouse 3 ball									
ball 3 ball									
mouse 4 mouse									
ball 4 mouse									
ball 4 mouse									
77 ///									

### **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.



#### **JOIN & JOIN** JOIN & LEFT JOIN **LEFT JOIN & LEFT JOIN SELECT SELECT SELECT** t.toy\_name, t.toy\_name, t.toy\_name, c.cat\_name, c.cat\_name, c.cat\_name, o.name AS owner\_name o.name AS owner\_name o.name AS owner\_name FROM tov t FROM tov t FROM tov t JOIN cat c JOIN cat c LEFT JOIN cat c ON t.cat\_id = c.cat\_id ON t.cat\_id = c.cat\_id ON t.cat id = c.cat id JOIN owner o LEFT JOIN owner o LEFT JOIN owner o ON c.owner\_id = o.id; ON c.owner\_id = o.id; ON c.owner\_id = o.id; toy\_name cat\_name owner\_name toy\_name cat\_name owner\_name toy\_name cat\_name owner\_name Kitty John Smith Kitty John Smith ball Kitty Kitty Kitty John Smith Kitty Danielle Davis ball Danielle Davis mouse Misty spring 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					OWNI	ER AS o
cat_id	cat_name	mom_id	owner_id	age	id	name
1	Kitty	5	1	17	1	John Smith
2	Hugo	1	2	10	2	Danielle Davis
3	Sam	2	2	5		
4	Mistv	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;

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