

SQL Subqueries with IN Nested Inside WHERE

- subqueries = inner queries = nested queries = inner select
queries embedded in a query
- they are part of *another* query, called an outer query
= outer select

As their name suggests, subqueries are queries embedded in a query. They are also called inner queries, or nested queries, and they are part of another query called an outer query. Alternative names for these SQL features are inner select and outer select, respectively. Subqueries can be applied in many ways. Nevertheless, the main idea is the same. Most often, a subquery is employed in the where clause of a select statement.

A really simple and interesting take on sub-queries:

```
6 # select the first and last name from the "Employees" table for the same
7 # employee numbers that can be found in the "Department Manager" table
8 • SELECT
9     e.first_name, e.last_name
10 FROM
11     employees e
12 WHERE
13     e.emp_no IN (SELECT
14                 dm.emp_no
15                 FROM
16                 dept_manager dm);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

	first_name	last_name
	Margareta	Markovitch
	Vishwani	Minakawa
	Ebru	Albin
	Isamu	Lecklitner
	Shirish	Ossenbruggen
	Karsten	Slootman
	Kraussir	Wiederle
	Rosine	Cools

employees 2 x

Output

Action Output

#	Time	Action	Message
1	16:00:23	SELECT FROM dept_manager	24 row(s) returned
2	16:01:45	SELECT e.first_name, e.last_name FROM employees e WHERE e.emp_no	24 row(s) returned

```
7 # employee numbers that can be found in the "Department Manager" table
8 • SELECT
9     e.first_name, e.last_name
10 FROM
11     employees e
12 WHERE
13     e.emp_no IN (SELECT
14                 dm.emp_no
15                 FROM
16                 dept_manager dm);
```

outer query

```

11 employees e
12 WHERE
13   e.emp_no IN (SELECT
14     dm.emp_no
15   FROM
16     dept_manager dm);

```

subquery (inner query)

a subquery should **always** be placed within parentheses

SQL Subqueries with IN Nested Inside WHERE

- a subquery may return a single value (a scalar), a single row, a single column, or an entire table
 - you can have a lot more than one subquery in your outer query
 - it is possible to nest inner queries within other inner queries
- in that case, the SQL engine would execute the innermost query first, and then each subsequent query, until it runs the outermost query last

EXISTS() FUNCTION:

SQL Subqueries with EXISTS-NOT EXISTS Nested Inside WHERE

• EXISTS

checks whether certain row values are found within a subquery

- this check is conducted row by row
- it returns a Boolean value

if a row value of a subquery **exists** → **TRUE** → the corresponding record of the outer query is extracted

if a row value of a subquery **doesn't exist** → **FALSE** → no row value from the outer query is extracted

Here's an example: it will deliver all first and last names of the people in the employees table who are also found in the Department Manager table. As a matter of fact, we'll create a whole table, not just a column, as we did with the in operator.

```

1 SELECT
2   e.first_name, e.last_name
3 FROM
4   employees e
5 WHERE
6   EXISTS( SELECT
7            *
8          FROM
9            dept_manager dm
10         WHERE
11           dm.emp_no = e.emp_no);
12
13
14
15
16

```

dept_manager

dept_no	CHAR(4)	(FK)
emp_no	INT	(FK)
from_date	DATE	
to_date	DATE	

SQL Subqueries with EXISTS-NOT EXISTS Nested Inside WHERE

EXISTS	IN
<u>tests</u> row values for existence	<u>searches</u> among values
quicker in retrieving <u>large amounts</u> of data	faster with <u>smaller</u> datasets

```

38 SELECT
39   e.first_name, e.last_name
40 FROM
41   employees e
42 WHERE
43   EXISTS( SELECT
44            *
45          FROM
46            dept_manager dm
47         WHERE
48           dm.emp_no = e.emp_no)
49 ORDER BY emp_no;

```

It is more professional to apply order by in the outer query, so we need to get used to leaving this clause in the outer select. It is more logical to sort the final version of our dataset and not the versions preceding the final one.

JOINS VS Sub-Queries:

We need to be aware that some, though not all, nested queries can be rewritten using joins, which are more efficient and general. This is true particularly for inner queries using the where clause, although sometimes the same answers can be obtained with a join. Nested queries are considered an essential tool, and that's why we must learn them. On certain occasions, subqueries can be bad for performance concerns, but here's why people would still need to use them. First, they allow for better structuring of the outer query. Thus, each inner query can be thought of in isolation, and organising the extraction process can be improved, hence the name of SQL, Structured Query Language. Second, in some situations, the use of subqueries is much more intuitive compared to the use of complex joins and unions. Finally, many users prefer subqueries simply because they offer enhanced code readability.