

# Subqueries

by Sophia



## WHAT'S COVERED

This lesson explores subqueries and how to use them in `SELECT` statements, in three parts.

Specifically, this lesson will cover:

1. [Subquery Basics](#)
2. [Using IN Operator in Clauses in Subqueries](#)
3. [Referring to Multiple Columns](#)

## 1. Subquery Basics

**Subqueries** in SQL are queries nesting within each other. One query can be used in another query as part of the results. Using subqueries, you can retrieve data that will be used for filtering, comparing, or computing in the main query. Combining and manipulating data from multiple tables and sources provides a powerful tool to check data in multiple dimensions in one query.

SQL statements can contain subqueries in various places:

- In the `SELECT` clause, a subquery can retrieve a single value used in the main query. For example, using a subquery, you might calculate an aggregate value and display it alongside each row.
- The `FROM` clause can contain derived tables, which are temporary tables that the main query can reference.
- Subqueries in the `WHERE` clause are commonly used to filter results based on conditions that involve data from other tables.
- The `IN`, `ANY`, and `ALL` operators are often used with subqueries to compare values returned by the subquery with values in the main query.
- In a correlated subquery, columns from the outer query are referenced from the inner query. You can perform operations based on the outer query's values using related subqueries.

↪ **EXAMPLE**

```
SELECT product_name, price FROM products WHERE price > (SELECT AVG(price) FROM products);
```

This example retrieves the names and prices of products whose prices are greater than the average price calculated by the subquery (SELECT AVG(price) FROM products).

By allowing you to operate on multiple tables and conditions simultaneously, subqueries enhance the flexibility of SQL queries. However, subqueries should be used with caution, as not optimizing them can adversely affect performance.

There are many potential uses for subqueries in the PostgreSQL database you have been working with in this class. For example, we could use a subquery to find all of the invoices that have a total amount larger than the average total amount.

We could do this in two separate steps based on what we've learned so far. We could first calculate the average total:

```
SELECT ROUND(AVG(total), 2)
FROM invoice;
```

# Query Results

Row count: 1

round

5.71

Next, we can take that value and query the invoice table to find those that are larger than 5.71.

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE total > 5.71;
```

## Query Results

Row count: 179

invoice_id	invoice_date	customer_id	total
3	2009-01-03T00:00:00.000Z	8	6
4	2009-01-06T00:00:00.000Z	14	9
5	2009-01-11T00:00:00.000Z	23	14
10	2009-02-03T00:00:00.000Z	46	6
11	2009-02-06T00:00:00.000Z	52	9
12	2009-02-11T00:00:00.000Z	2	14
17	2009-03-06T00:00:00.000Z	25	6
18	2009-03-09T00:00:00.000Z	31	9
19	2009-03-14T00:00:00.000Z	40	14
24	2009-04-06T00:00:00.000Z	4	6
25	2009-04-09T00:00:00.000Z	10	9

Although this approach works, we can also do it in a way that can pass the result from the first query (that calculated the average) into the second query by using a subquery. The subquery can be nested in a SELECT, INSERT, DELETE, or UPDATE statement, but we mostly see it in a SELECT statement.

To construct the subquery, we would enclose the second query in round brackets and place it in the WHERE clause as an expression. So, consider our original second statement:

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE total > 5.71;
```

We would remove the 5.71 and replace it with round brackets enclosing the first statement, like this:

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE total > (SELECT ROUND(AVG(total), 2) FROM invoice);
```

The query inside the round brackets is called the subquery, or inner query. The query that contains the subquery is called the **outer query**. Notice that we don't include the semicolon at the end of the inner statement in the subquery.

The database would execute the subquery first, then take the result from that subquery and pass it to the outer query. Then, the database executes the outer query.



## TERMS TO KNOW

### Subquery

Also called an inner query, a query that is nested inside another query, so its results can be used as input for the outer query.

## Outer Query

A query that uses the output of a subquery (inner query) as input.

# 2. Using IN Operator in Clauses in Subqueries

A subquery could potentially return 0 or more results, so it is important to consider the operator used to compare with its results. We can use any valid SQL operator in this type of query. For example, if we use a = operator, we would expect that the subquery would return 0 or 1 row(s) as a result.

Let's look at an obvious subquery that is querying on the customer\_id to return the customer\_id:

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE customer_id =
(SELECT customer_id
FROM customer
WHERE customer_id = 1);
```

Since the primary key of the customer table is the customer\_id, querying on the customer\_id would only return one value.

Query Results			
Row count: 7			
invoice_id	invoice_date	customer_id	total
98	2010-03-11T00:00:00.000Z	1	4
121	2010-06-13T00:00:00.000Z	1	4
143	2010-09-15T00:00:00.000Z	1	6
195	2011-05-06T00:00:00.000Z	1	1
316	2012-10-27T00:00:00.000Z	1	2
327	2012-12-07T00:00:00.000Z	1	14
382	2013-08-07T00:00:00.000Z	1	9

Consider if we tried to select a value that doesn't exist:

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE customer_id =
(SELECT customer_id
FROM customer
WHERE customer_id = 0);
```

The results would still run, but show that 0 rows were displayed:

## Query Results

Query ran successfully. 0 rows to display.

However, consider the case in which we have multiple rows being returned:

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE customer_id =
(SELECT customer_id
FROM customer
WHERE country = 'USA');
```

Since there are 13 customers that live in the country USA, we will end up getting this error:

## Query Results

Query failed because of: error: more than one row returned by a subquery used as an expression

In order to avoid this error, we have to use the IN operator instead of the equal sign. This will allow 0, 1, or many results to be returned:

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE customer_id IN
(SELECT customer_id
FROM customer
WHERE country = 'USA');
```

## Query Results

Row count: 91

invoice_id	invoice_date	customer_id	total
5	2009-01-11T00:00:00.000Z	23	14
13	2009-02-19T00:00:00.000Z	16	1
14	2009-03-04T00:00:00.000Z	17	2
15	2009-03-04T00:00:00.000Z	19	2
16	2009-03-05T00:00:00.000Z	21	4
17	2009-03-06T00:00:00.000Z	25	6
26	2009-04-14T00:00:00.000Z	19	14

### 3. Referring to Multiple Columns

We can also use multiple columns as criteria to compare with the subquery. In order to do so, we must use the round brackets around the columns within the WHERE clause and have them match up with the columns that we want to compare within the subquery. For example, if we wanted to compare the customer\_id and the billing\_country in the invoice table with the customer\_id and country in the customer table, we could do the following:

```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE (customer_id, billing_country) IN
(SELECT customer_id, country
FROM customer
WHERE country = 'USA');
```

#### Query Results

Row count: 91

invoice_id	invoice_date	customer_id	total
5	2009-01-11T00:00:00.000Z	23	14
13	2009-02-19T00:00:00.000Z	16	1
14	2009-03-04T00:00:00.000Z	17	2
15	2009-03-04T00:00:00.000Z	19	2
16	2009-03-05T00:00:00.000Z	21	4
17	2009-03-06T00:00:00.000Z	25	6
26	2009-04-14T00:00:00.000Z	19	14

If we did not put parentheses around the columns named in the first WHERE clause, we would get an error.


```
SELECT invoice_id, invoice_date, customer_id, total
FROM invoice
WHERE customer_id, billing_country IN
(SELECT customer_id, country
FROM customer
WHERE country = 'USA');
```

#### Query Results

Query failed because of: error: syntax error at or near ","




Your turn! Open the SQL tool by clicking on the **LAUNCH DATABASE** button below. Then, enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

 SUMMARY

In this lesson, you learned the **basics of subqueries**, which allow data from multiple tables or data sources to be integrated within a single query. They are nested queries that are embedded within larger queries to retrieve specific data for use in the main query. A SQL subquery can appear in various parts of the statement, such as the **SELECT**, **FROM**, **WHERE**, or **HAVING** clause, and it provides a means of manipulating and analyzing data more intricately.

It is possible to make SQL queries more dynamic and adaptable by leveraging subqueries. With them, you can filter, compare, and compute based on values from other tables or even previous rows in your query. Subqueries can also be used to create derived tables that can be referenced like regular tables, simplifying calculations, or data transformations. You explored **using IN operator in clauses in subqueries** to compare values returned by the subquery with values in the main query. You also learned that SQL subqueries can be used to **refer to multiple columns**. Despite their flexibility, subqueries require optimization for performance, as poorly constructed subqueries can cause slow execution. Using subqueries efficiently and accurately in SQL operations requires careful consideration of their structure and purpose.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND FAITHE WEMPEN (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR [TERMS OF USE](#).

 TERMS TO KNOW

**Outer Query**  
A query that uses the output of a subquery (inner query) as input.

**Subquery**  
Also called an inner query, a query that is nested inside another query, so its results can be used as input for the outer query.