PostgreSQL Subquery

Summary: in this tutorial, you will learn how to use the **PostgreSQL subquery** that allows you to construct complex queries.

Introduction to PostgreSQL subquery

Let's start with a simple example.

Suppose we want to find the films whose rental rate is higher than the average rental rate. We can do it in two steps:

PostgreSQL Quick Start

What is PostgreSQL?

Install PostgreSQL

- Find the average rental rate by using the SELECT statement and average function (AVG).
- Use the result of the first query in the second SELECT statement to find the films that we want.

The following query gets the average rental rate:

```
SELECT
    AVG (rental_rate)
FROM
    film;
```

avg
2.98

The average rental rate is 2.98

Now, we can get films whose rental rate is higher than the average rental rate:

```
SELECT
    film_id,
    title,
    rental_rate
FROM
    film
WHERE
```

Connect to Database

Download PostgreSQL Sample Database

Load Sample Database

Explore Server and Database Objects

PostgreSQL Fundamentals

PostgreSQL Select

PostgreSQL Order By

PostgreSQL Select Distinct

PostgreSQL Where

PostgreSQL IN

PostgreSQL Between

PostgreSQL Like

PostgreSQL Union

PostgreSQL Inner Join

PostgreSQL Left Join

PostgreSQL Group By

PostgreSQL Having

```
    rental_rate > 2.98;
```

film_id	title	rental_rate
133	Chamber Italian	4.99
384	Grosse Wonderful	4.99
8	Airport Pollock	4.99
98	Bright Encounters	4.99
2	Ace Goldfinger	4.99
3	Adaptation Holes	2.99
4	Affair Prejudice	2.99
5	African Egg	2.99

The code is not so elegant, which requires two steps. We want a way to pass the result of the first query to the second query in one query. The solution is subquery.

A subquery is a query nested inside another query such as `SELECT`, `INSERT`, `DELETE` and `UPDATE`. In this tutorial, we are focusing on the `SELECT` statement only.

To construct a subquery, we put the second query in brackets and use it in the WHERE clause as an expression:

```
SELECT
```

```
    film_id,
    title,
    rental_rate
FROM
    film
WHERE
    rental_rate > (
        SELECT
            AVG (rental_rate)
        FROM
            film
    );
```

The query inside the brackets is called a subquery or an inner query. The query that contains the subquery is known as an outer query.

PostgreSQL executes the query that contains a subquery in the following sequence:

- First, executes the subquery.
- Second, gets the result and passes it to the outer query.
- Third, executes the outer query.

PostgreSQL subquery with IN operator

A subquery can return zero or more rows. To use this subquery, you use the IN operator in the `WHERE` clause.

For example, to get films that have return date between `2005-05-29` and `2005-05-30`, you use the following query:

```
SELECT
    inventory.film_id
FROM
    rental
INNER JOIN inventory ON inventory.inventory_id = rental.inventory_id
WHERE
    return_date BETWEEN '2005-05-29'
AND '2005-05-30';
```

film_id
15
19
45
50
52
54
68
73

It returns multiple rows so we can use this query as a subquery in the `WHERE` clause of a query as follows:

```
SELECT
    film_id,
```

```
    title
FROM
    film
WHERE
    film_id IN (
        SELECT
            inventory.film_id
        FROM
            rental
        INNER JOIN inventory ON inventory.inventory_id = rental.inve
        WHERE
            return_date BETWEEN '2005-05-29'
        AND '2005-05-30'
    );
```

film_id	title
120	Caribbean Liberty
480	Jeepers Wedding
681	Pirates Roxanne
227	Details Packer
247	Downhill Enough
347	Games Bowfinger
295	Expendable Stallion
517	Lesson Cleopatra
971	Whale Bikini

PostgreSQL subquery with EXISTS operator

The following expression illustrates how to use a subquery with `EXISTS` operator:

```
EXISTS subquery
```

A subquery can be an input of the `EXISTS` operator. If the subquery returns any row, the `EXISTS` operator returns true. If the subquery return no row, the result of `EXISTS` operator is false.

The `EXISTS` operator only cares about the number of rows returned from the subquery, not the content of the rows therefore the common coding convention of `EXISTS` operator is as follows:

```
EXISTS (SELECT 1 FROM tbl WHERE condition);
```

See the following query:

```
SELECT
    first_name,
    last_name
FROM
    customer
WHERE
    EXISTS (
        SELECT
            1
        FROM
            payment
        WHERE
```

```
            payment.customer_id = payment.customer_id
    );
```

first_name	last_name
Jared	Ely
Mary	Smith
Patricia	Johnson
Linda	Williams
Barbara	Jones
Elizabeth	Brown
Jennifer	Davis
Maria	Miller
Susan	Wilson

The query works like an inner join on the customer id column. However, it returns at most one row for each row in the customer table even through there are some corresponding rows in the payment table.

In this tutorial, you have learned how to use the PostgreSQL subquery to construct complex queries in

« Previous Tutorial:
PostgreSQL HAVING

Next Tutorial:
PostgreSQL INSERT »

About PostgreSQL Tutorial Website

PostgreSQLTutorial.com is a website dedicated to developers and database administrators who are working on PostgreSQL database management system.

We constantly publish useful PostgreSQL tutorials to keep you up-to-date with the latest PostgreSQL features and technologies. All PostgreSQL tutorials are simple, easy-to-follow and practical.

Recent Tutorials

Developing User-defined Functions Using PostgreSQL CREATE FUNCTION Statement

PostgreSQL Stored Procedures

Introduction to PostgreSQL Stored Procedures

PostgresQL Roles Management

PostgreSQL Restore Database

Backing Up Databases Using PostgreSQL Backup Tools

Deleting Tablespaces Using PostgreSQL DROP TABLESPACE Statement

PostgreSQL ALTER TABLESPACE

Site Info

Home

About Us

Contact Us

Privacy Policy