DROP VIEW to Remove Views

by Sophia

WHAT'S COVERED

This lesson explores using the DROP VIEW statement to remove views, in three parts. Specifically, this lesson will cover:

1. Introduction

When a view is no longer needed, or if you want to delete it so you can recreate it, you can use the **DROP VIEW** statement to remove it from the database.

The syntax for dropping a view in PostgreSQL is straightforward:

```
DROP VIEW [IF EXISTS] view_name [, ...] [CASCADE | RESTRICT];
```

The view_name is the name of the view you want to remove.

IF EXISTS is an optional clause that prevents an error from occurring if the view doesn't exist.

CASCADE allows you to drop dependent objects like rules or triggers associated with the view.

RESTRICT prevents the view from being dropped if there are any dependent objects, providing a safety mechanism to avoid accidental deletions.

When you execute the DROP VIEW command, PostgreSQL will permanently delete the specified view and all its associated metadata, including permissions. It's important to exercise caution when using this command, especially in production environments, as it can impact applications relying on the view's existence and structure.

TERM TO KNOW

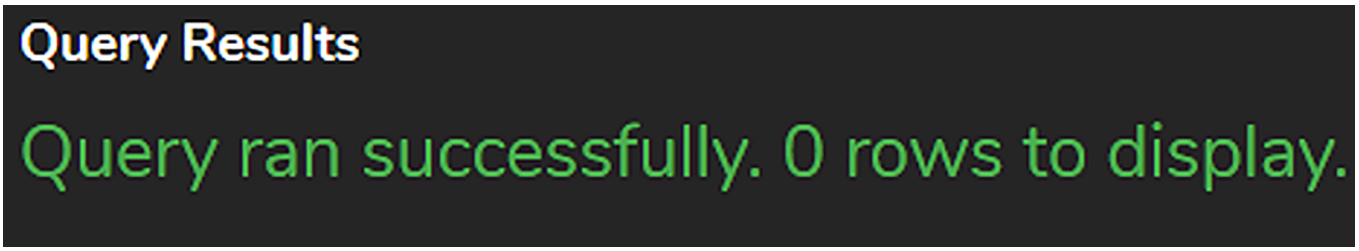
DROP VIEW

A statement that deletes a view.

2. Dropping a View

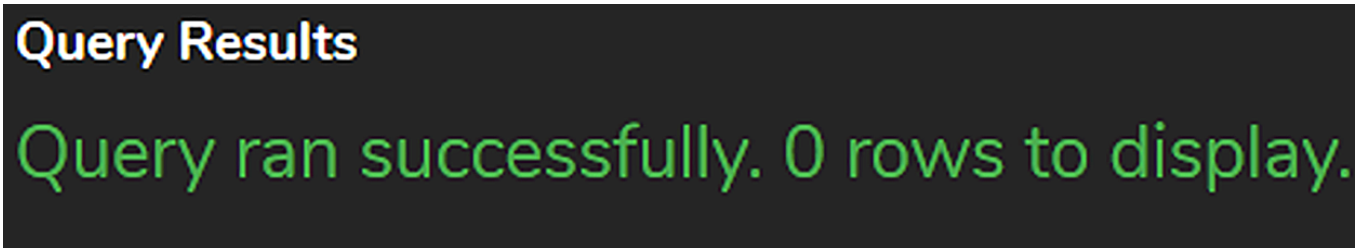
Let's look at an example of dropping a view. First, suppose we create the following view for the album and artist names:

```
CREATE VIEW album_artist_names
AS
SELECT album.title, artist.name
FROM album
INNER JOIN artist ON album.artist_id = artist.artist_id;
```



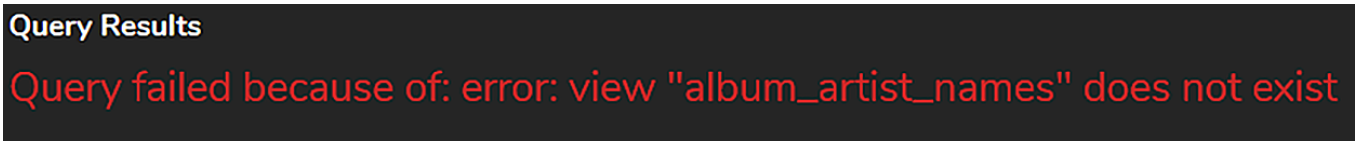
If we wanted to drop the view, it would look like this:

```
DROP VIEW album_artist_names;
```



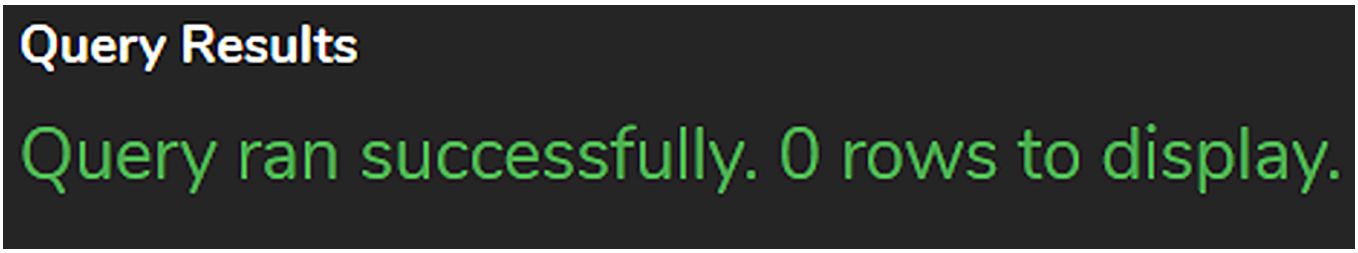
If you try to drop a view that doesn't exist, an error message will appear. For example, if we tried to drop the album_artist_names view a second time, we would see the following:

```
DROP VIEW album_artist_names;
```



However, by using the IF EXISTS parameter, the database will only drop the view if it exists. If not, no error message will be generated:

```
DROP VIEW IF EXISTS album_artist_names;
```



3. Managing the View's Dependencies

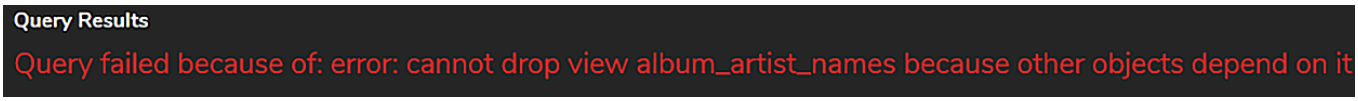
By default, the RESTRICT parameter is in effect in a DROP VIEW statement. RESTRICT prevents the view from being dropped if there are any dependencies upon it. For example, suppose we created the album_artist_names view and then created another view called temp_album_artist_names that queried from the album_artist_names view:

```
CREATE VIEW album_artist_names
AS
SELECT album.title, artist.name
FROM album
INNER JOIN artist ON album.artist_id = artist.artist_id;
CREATE VIEW temp_album_artist_names
AS
SELECT *

FROM album_artist_names;
```

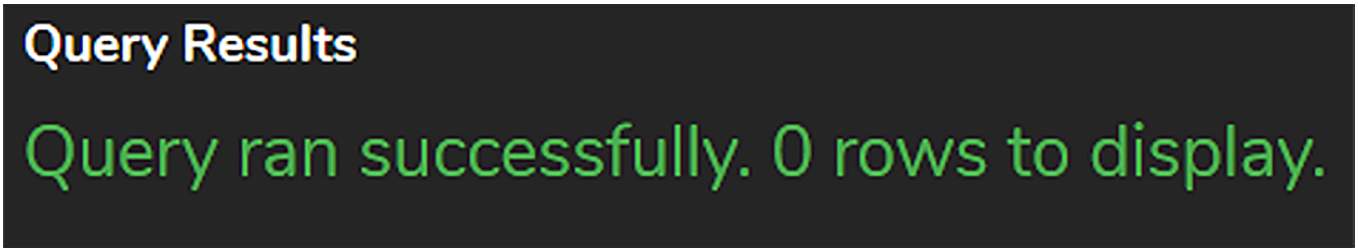
If we tried to drop the album_artist_names view, we would get an error, as the temp_album_artist_names table uses that view:

```
DROP VIEW IF EXISTS album_artist_names
RESTRICT;
```



Alternatively, we could use the CASCADE option to have the statement automatically drop all of the objects that depend on the view, and all objects that also then depend on those other objects. The statement with the cascade looks like this:

```
DROP VIEW IF EXISTS album_artist_names
CASCADE;
```

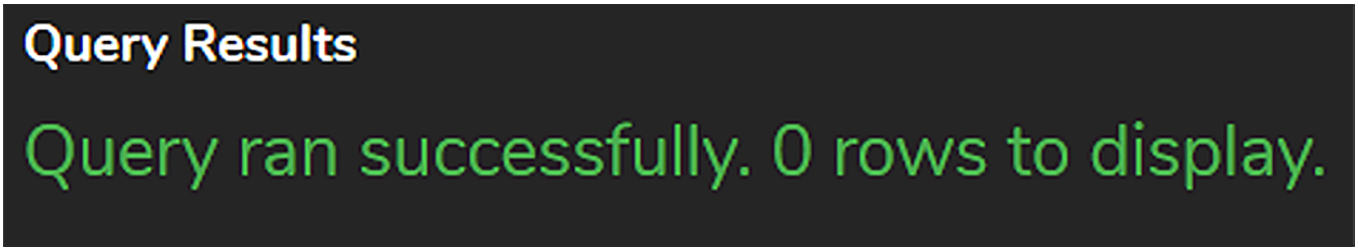


Be careful when using CASCADE, as there could be many unintended items dropped. A lengthier but more cautious approach would be to drop views one at a time:

```
DROP VIEW IF EXISTS temp_album_artist_names;
DROP VIEW IF EXISTS album_artist_names;
```

You can drop multiple views using a single DROP VIEW statement, like this:

```
DROP VIEW IF EXISTS album_artist_names, temp_album_artist_names;
```



Notice that even though there is a dependency between the two views that were dropped in the preceding statement, they could both be dropped together.

WATCH

TRY IT

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 in the **introduction** that the DROP VIEW statement removes views from a database, followed by exploring an example of **dropping a view**. You might remove a view because it's no longer needed, or because you plan to recreate it in a different way.

The IF EXISTS clause can optionally be added to prevent an error from appearing if the view does not exist. You also learned about **managing the view's dependencies**; for example, you can use RESTRICT to prevent the view from being dropped if any dependent objects exist, but it is unnecessary to add it

to the statement because it is the default behavior anyway. The CASCADE option does the opposite: It deletes any objects that are dependent upon the view being deleted.

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

DROP VIEW

A statement that deletes a view.