DROP TABLE to Remove Tables

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores using the DROP TABLE statement to remove a table in three parts:

1. Introduction

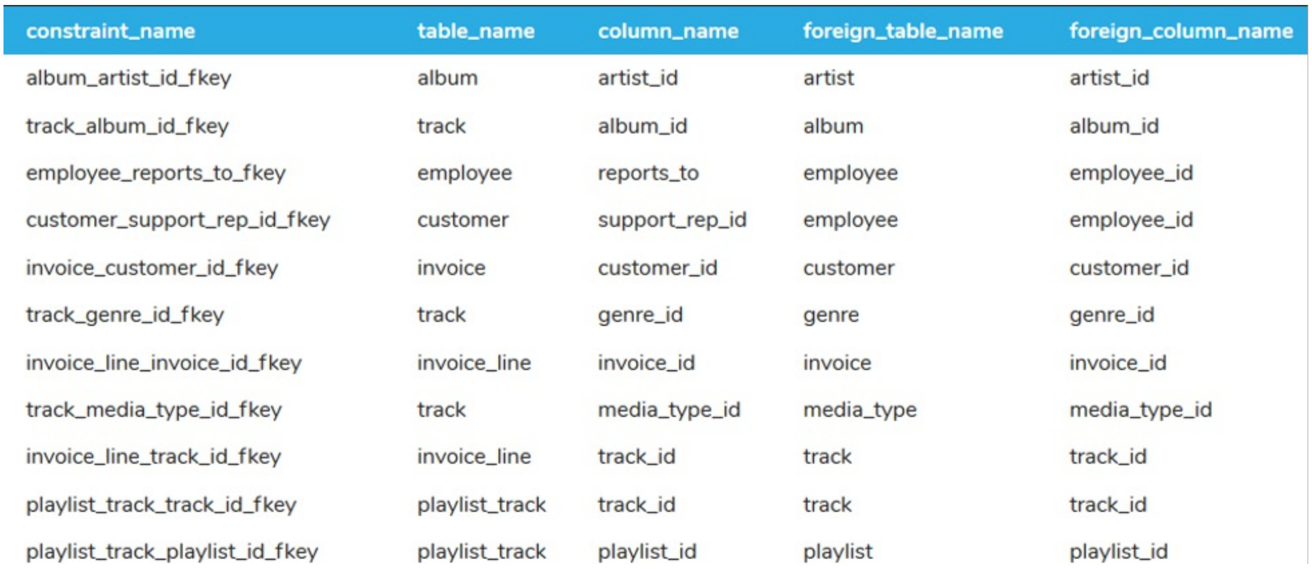
A table can be removed from the database by using the DROP TABLE command. It is one of the easier statements to write but it can get a bit more complex when there are relationships between tables. The structure of the statement looks like the following:

```
DROP TABLE <tablename>;
```

In a relationship between tables, you can only drop a table if it is not the “one” side of a one-to-many relationship. We will get into more specifics about table relationships later in the course, but it is important to be aware of the details. If we try to drop a table that has an existing relationship, we will get an error. The order in which we drop tables depends on the foreign key constraints.

2. Examples

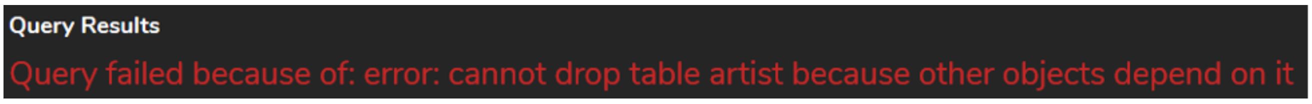
Let's look at our database. We have the following foreign keys set up across the various tables:



constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

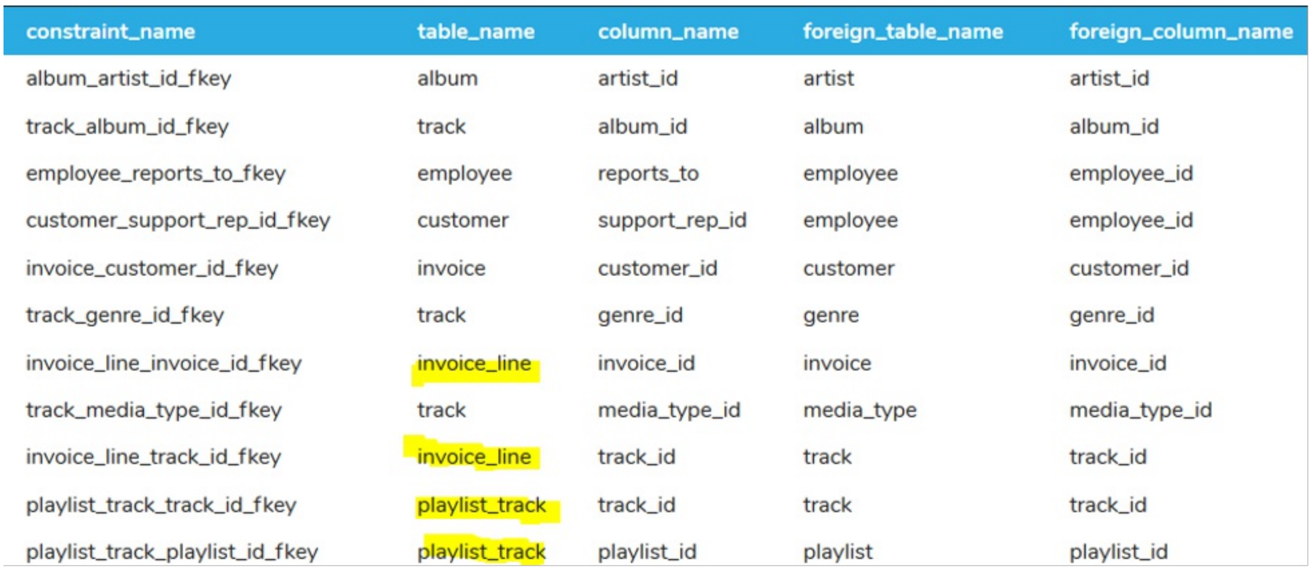
We can see that the album table has the artist_id as a foreign key to the artist table's artist_id. If we tried to run a DROP TABLE on the artist table, we should get the following error:

DROP TABLE artist;



```
Query Results
Query failed because of: error: cannot drop table artist because other objects depend on it
```

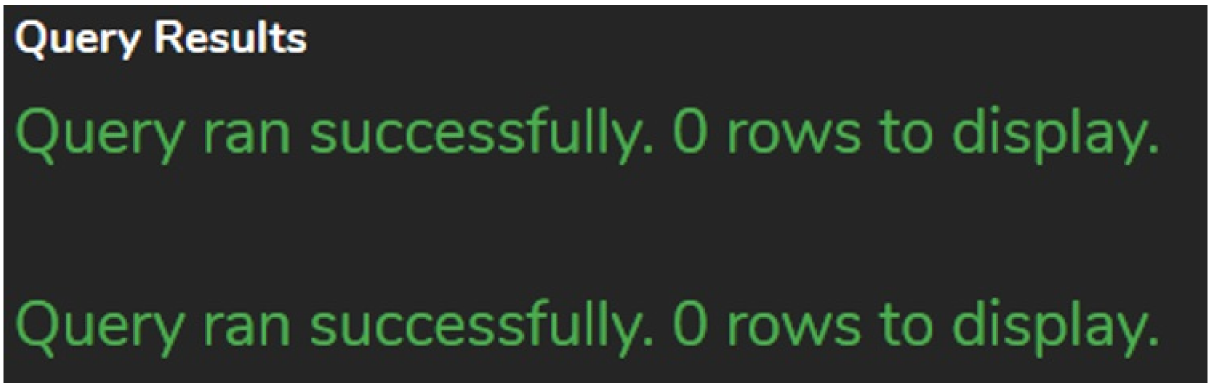
It can be a bit of a puzzle to identify the order in which the tables need to be dropped. To do so, we first want to identify the tables that do not have a foreign key. These include the invoice_line table and the playlist_track table.



constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

We should be able to drop them both. Notice that we have a semicolon at the end of each line; these are separate commands that are run sequentially. As such, the query results will include a successful run command for each individual statement.

DROP TABLE invoice_line;
DROP TABLE playlist_track;

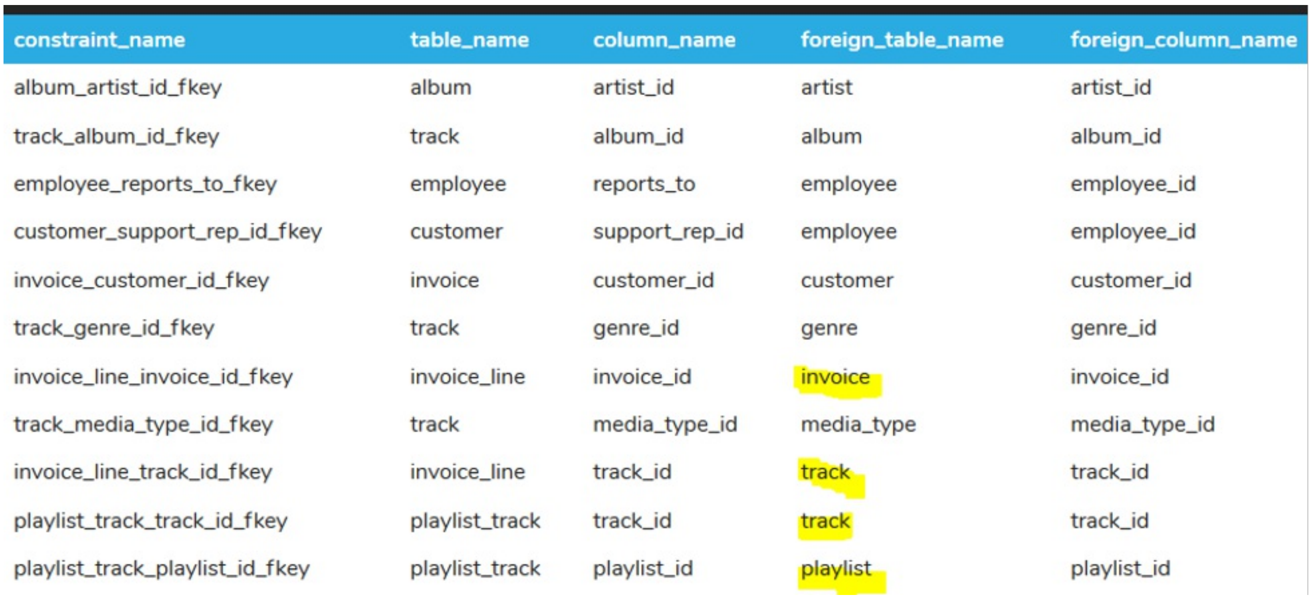


```
Query Results

Query ran successfully. 0 rows to display.

Query ran successfully. 0 rows to display.
```

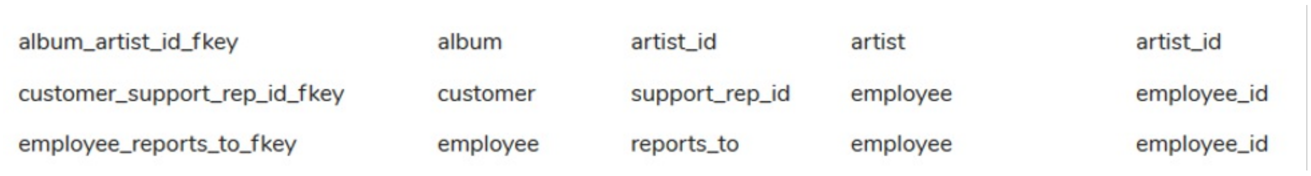
Next, we can look at the tables they had initially referenced to see if they have any relationships that still remain as foreign keys:



constraint_name	table_name	column_name	foreign_table_name	foreign_column_name
album_artist_id_fkey	album	artist_id	artist	artist_id
track_album_id_fkey	track	album_id	album	album_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
invoice_customer_id_fkey	invoice	customer_id	customer	customer_id
track_genre_id_fkey	track	genre_id	genre	genre_id
invoice_line_invoice_id_fkey	invoice_line	invoice_id	invoice	invoice_id
track_media_type_id_fkey	track	media_type_id	media_type	media_type_id
invoice_line_track_id_fkey	invoice_line	track_id	track	track_id
playlist_track_track_id_fkey	playlist_track	track_id	track	track_id
playlist_track_playlist_id_fkey	playlist_track	playlist_id	playlist	playlist_id

Since they do not, we can go ahead and drop those tables as well:

DROP TABLE invoice;
DROP TABLE track;
DROP TABLE playlist;
Then, looking at those tables, we can track down which tables referenced them:



album_artist_id_fkey	album	artist_id	artist	artist_id
customer_support_rep_id_fkey	customer	support_rep_id	employee	employee_id
employee_reports_to_fkey	employee	reports_to	employee	employee_id

Although we have the album, artist, customer, employee, genre, and media_type tables left, if we drop the album and customer tables, we can drop the rest, as the employee table is linked to itself.

We can drop all of the tables by running the following:

DROP TABLE album;
DROP TABLE customer;

```
DROP TABLE employee;
DROP TABLE artist;
DROP TABLE genre;
DROP TABLE media_type;
```

As a recap, we could break down the order of the dropping of the tables into four separate sets of statements, starting with dropping the tables that had no foreign keys linked to them:

```
DROP TABLE invoice_line;
DROP TABLE playlist_track;
```

Next, we drop the tables that only had foreign keys to those tables that were dropped:

```
DROP TABLE invoice;
DROP TABLE track;
DROP TABLE playlist;
```

Then, we drop the tables that had foreign keys to those tables dropped:

```
DROP TABLE album;
DROP TABLE customer;
```

Lastly, we proceed with the rest of the tables as they no longer had any foreign keys that held them back from being dropped:

```
DROP TABLE employee;
DROP TABLE artist;
DROP TABLE genre;
DROP TABLE media_type;
```

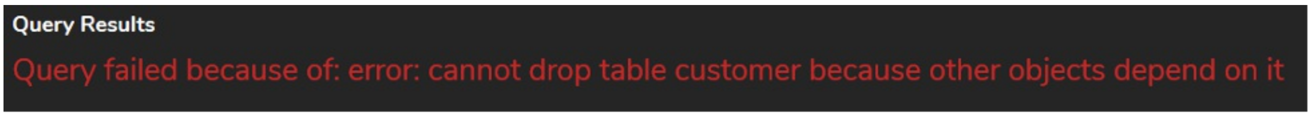
---

2. DROP TABLE Options

Another approach to avoid having to go through each of these steps is to use the DROP TABLE with the CASCADE option. This will drop the table, but also remove any constraints that link to the table. For example, if we tried to drop the CUSTOMER table:

```
DROP TABLE customer;
```

We should get the following error, as the customer_id is used in the invoice table:



```
Query failed because of: error: cannot drop table customer because other objects depend on it
```

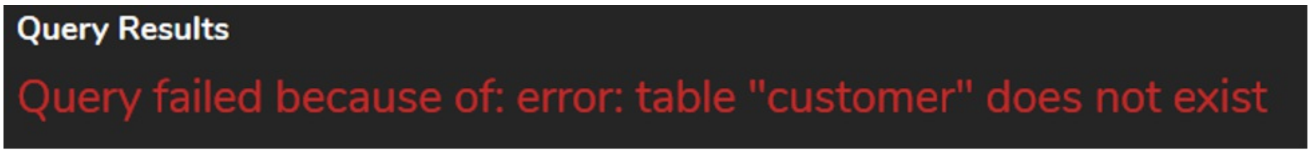
By running this command:

```
DROP TABLE customer CASCADE;
```

The CASCADE option drops the table but also removes the foreign key constraint on the invoice table for the customer_id.

Another option with the DROP TABLE statement is the IF EXISTS option. If you try to drop a table that doesn't exist or has already been dropped, you will get an error:

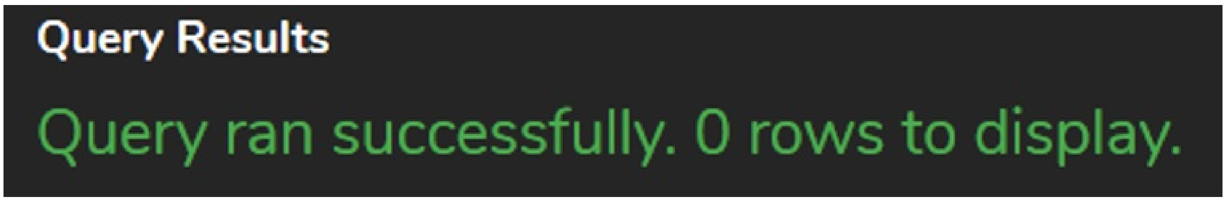
DROP TABLE customer;



```
Query Results
Query failed because of: error: table "customer" does not exist
```

However, adding the IF EXISTS will allow you to still run the command:

DROP TABLE IF EXISTS customer;



```
Query Results
Query ran successfully. 0 rows to display.
```

This can be useful when you include multiple commands together and do not want the database to stop on an error.

Video Transcription

[MUSIC PLAYING] When it comes to the syntax to drop in the table it's quite simple. It just has drop table and then the table name. However, where things get a little bit more difficult is when you think about the order in which you can actually drop the tables because of foreign keys. So for example, if I try to drop the table artist, we will get an error because other objects depended on it.

So if we take a look at the artist table we have the artist ID in which the album table has a foreign key to the artist. If we take a look at the album table with the album id who has the primary key, we'll be able to see that it's actually linked from the track table. And then from the track table, we have it linked in the playlist track table. There's nothing that's actually linked to the playlist track ID, which is not the primary key in this case. So we'd have to drop it based on that order based on what's actually existing in the tables.

So there's two different tables that have no links in this case here within our database. That will be the invoice line and the playlist track. We can go ahead and drop those first. Once we do that, we can go ahead and take a look at the next set of tables, and that'll be the invoice table, the track table, and the playlist table. So if we drop all those, we can go ahead and do that. And because of that order, it doesn't really matter which order that is in the case, but it's important that it is in the correct order.

Once we have that we can drop the other tables that were linked to those ones, which will be the album table, the customer table, the employee table, the artist table, the genre table, and the media type table. And that ends up dropping all the tables that we actually had in this case.

[MUSIC PLAYING]

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

The DROP TABLE command is used to remove tables from the database.

Source: Authored by Vincent Tran