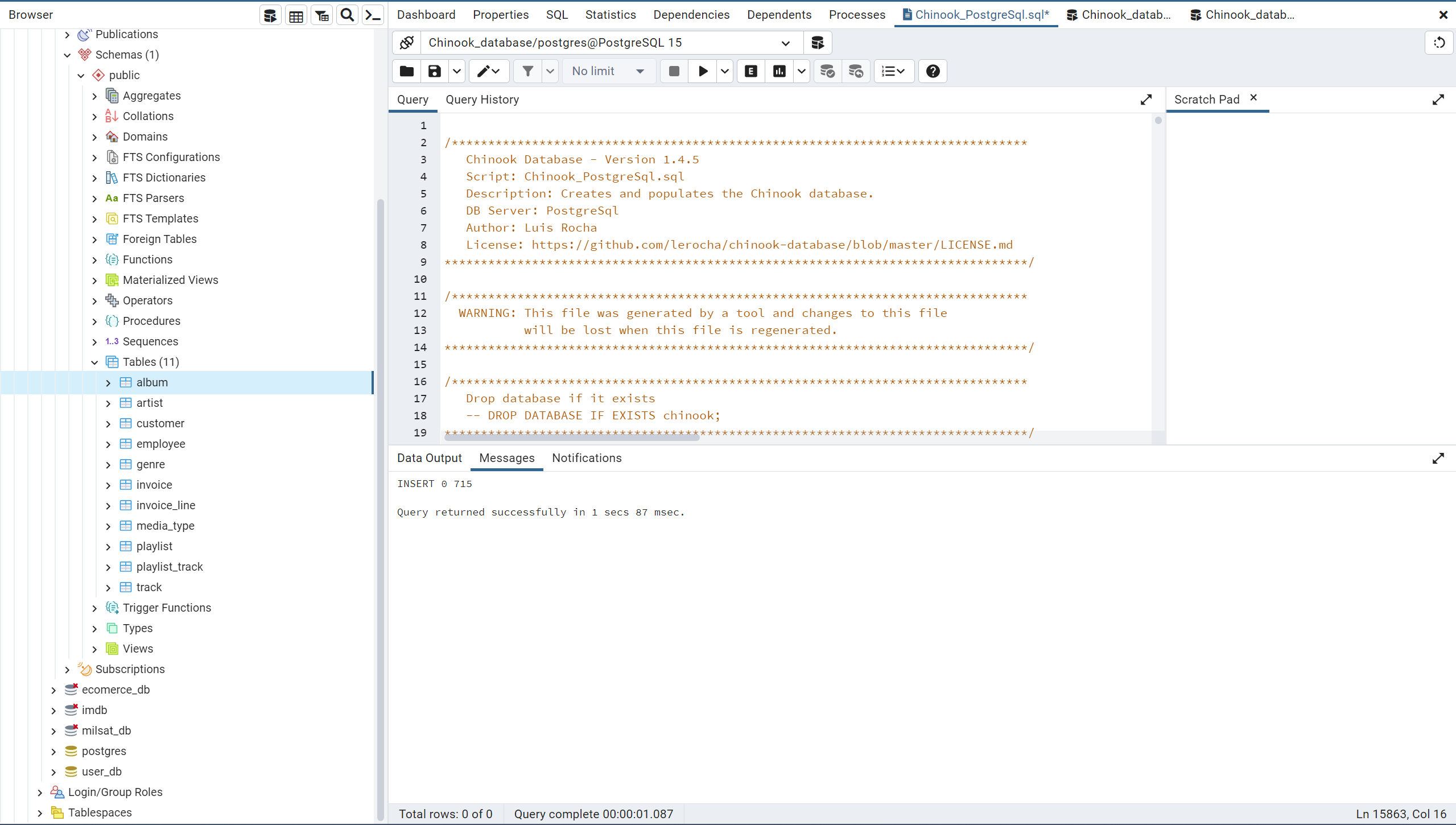
I downloaded the Chinook database SQL script from GitHub and set up a local PostgreSQL environment to work with this Chinook music store dataset. First, I installed PostgreSQL and pgAdmin 4 so I could manage the database through a GUI.

In pgAdmin, I created a new database called **Chinook\_database** to host the Chinook tables. I modified the downloaded SQL script by commenting out the database creation parts while keeping all the table creation and data insertion statements. Then I ran this modified script against my database to create all the tables with their sample data.

To connect to this database from Python in my notebook, I installed the psycopg2 library, which lets Python talk to PostgreSQL databases. I figured out how to write the connection string for a local PostgreSQL server and successfully used pandas' read\_sql\_query function to pull data from my database into a DataFrame.

I'm now able to write SQL queries that join multiple tables together, like combining customer information with their purchased tracks and albums. I can execute these queries either directly in pgAdmin or through Python using psycopg2 and pandas.



A screenshot of a computer

AI-generated content may be incorrect.

**Write a SQL query that joins information across five tables.**

-- query to show invoice details with customer and music information  
SELECT   
 c.customer\_id,  
 c.first\_name,  
 c.last\_name,  
 i.invoice\_id,  
 i.invoice\_date,  
 -- track and album information for each purchase  
 t.name AS track\_name,  
 a.title AS album\_title,  
 il.quantity,  
 il.unit\_price,  
 -- calculate the total price for each line item  
 (il.quantity \* il.unit\_price) AS line\_total  
FROM   
 -- start with main tables and join related information  
 invoice AS i  
JOIN   
 customer AS c ON i.customer\_id = c.customer\_id  
JOIN   
 -- connect to invoice details and music information  
 invoice\_line AS il ON i.invoice\_id = il.invoice\_id  
JOIN   
 track AS t ON il.track\_id = t.track\_id  
JOIN   
 album AS a ON t.album\_id = a.album\_id  
-- organize results by customer and then by date  
ORDER BY   
 c.customer\_id  
LIMIT 100;

**Create a pandas DataFrame that displays Customers’ Last Name and First Name, and each customer’s purchased Track names and Album Titles. The information should be sorted by Customer LastName then Customer FirstName.**

SELECT   
 c.last\_name AS customer\_last\_name,  
 c.first\_name AS customer\_first\_name,  
 t.name AS track\_name,  
 a.title AS album\_title  
FROM   
 customer AS c  
JOIN   
 invoice AS i ON c.customer\_id = i.customer\_id  
JOIN   
 invoice\_line AS il ON i.invoice\_id = il.invoice\_id  
JOIN   
 track AS t ON il.track\_id = t.track\_id  
JOIN   
 album AS a ON t.album\_id = a.album\_id  
ORDER BY   
 c.last\_name,  
 c.first\_name,  
 t.name  
LIMIT 5;