Hands-on Lab: Create Tables and Load Data in PostgreSQL using pgAdmin

Estimated time needed: 20 minutes

In this lab, you will learn how to create tables and load data in the PostgreSQL database service using the pgAdmin graphical user interface (GUI) tool. The pgAdmin GUI provides an alternative to the command line for interacting with a PostgreSQL database using a graphical interface. This GUI provides a number of key features for interacting with a PostgreSQL database in an easy to use format.

Software used in this lab

In this lab, you will use PostgreSQL Database. PostgreSQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.

To complete this lab you will utilize the PostgreSQL relational database service available as part of IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database used in this lab

You will use the Books database in this lab.

The following diagram shows the structure of the "myauthors" table from the Books database:

myauthors	
author_id	int
first_name	varchar(100)
middle_name	varchar(50)
last_name	varchar(100)

Objectives

After completing this lab, you will be able to use pgAdmin with PostgreSQL to:

- Create databases and tables in a PostgreSQL instance
- Load data into tables manually using the pgAdmin GUI
- Load data into tables from a text/script file

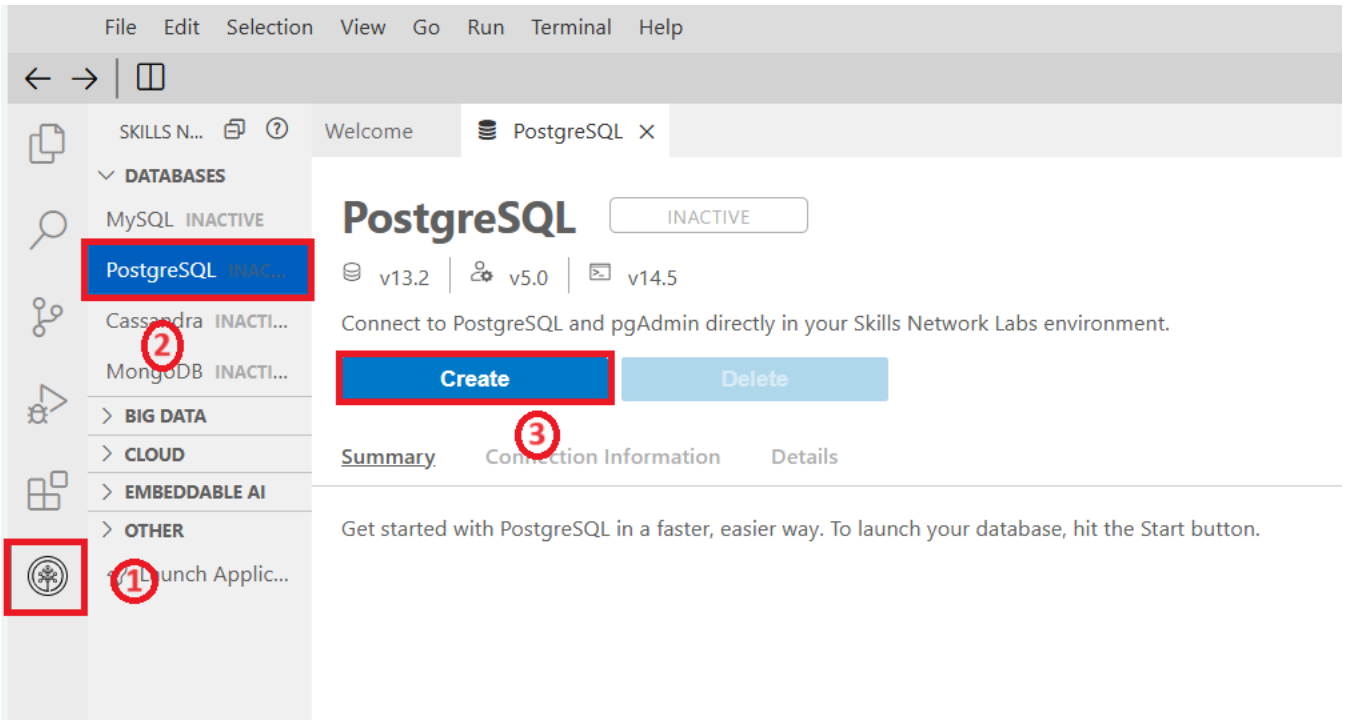
Lab Structure

In this lab, you will complete several tasks in which you will learn how to create tables and load data in the PostgreSQL database service using the pgAdmin graphical user interface (GUI) tool.

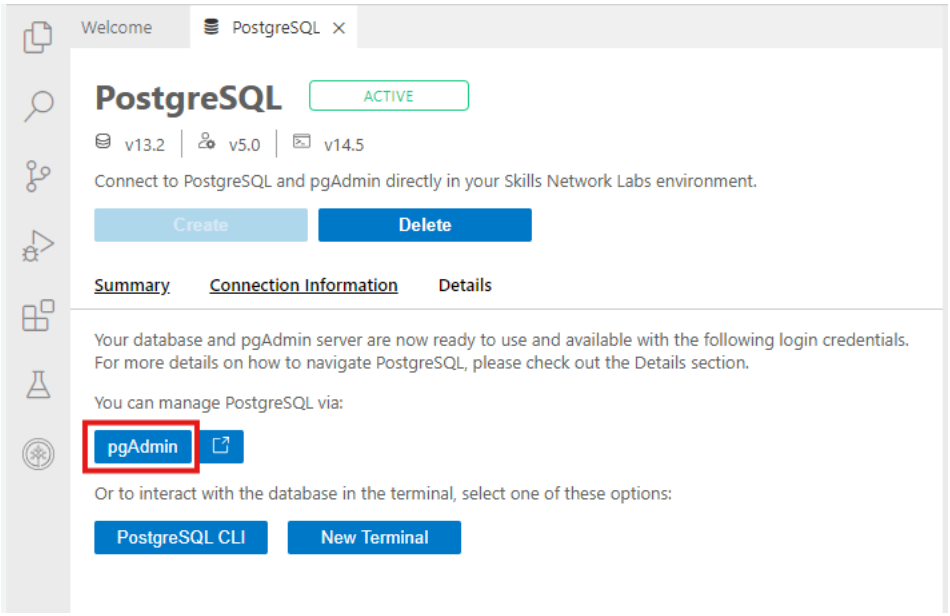
Task A: Create a database

First, to create a database on a PostgreSQL server instance, you'll first launch a PostgreSQL server instance on Cloud IDE and open the pgAdmin Graphical User Interface.

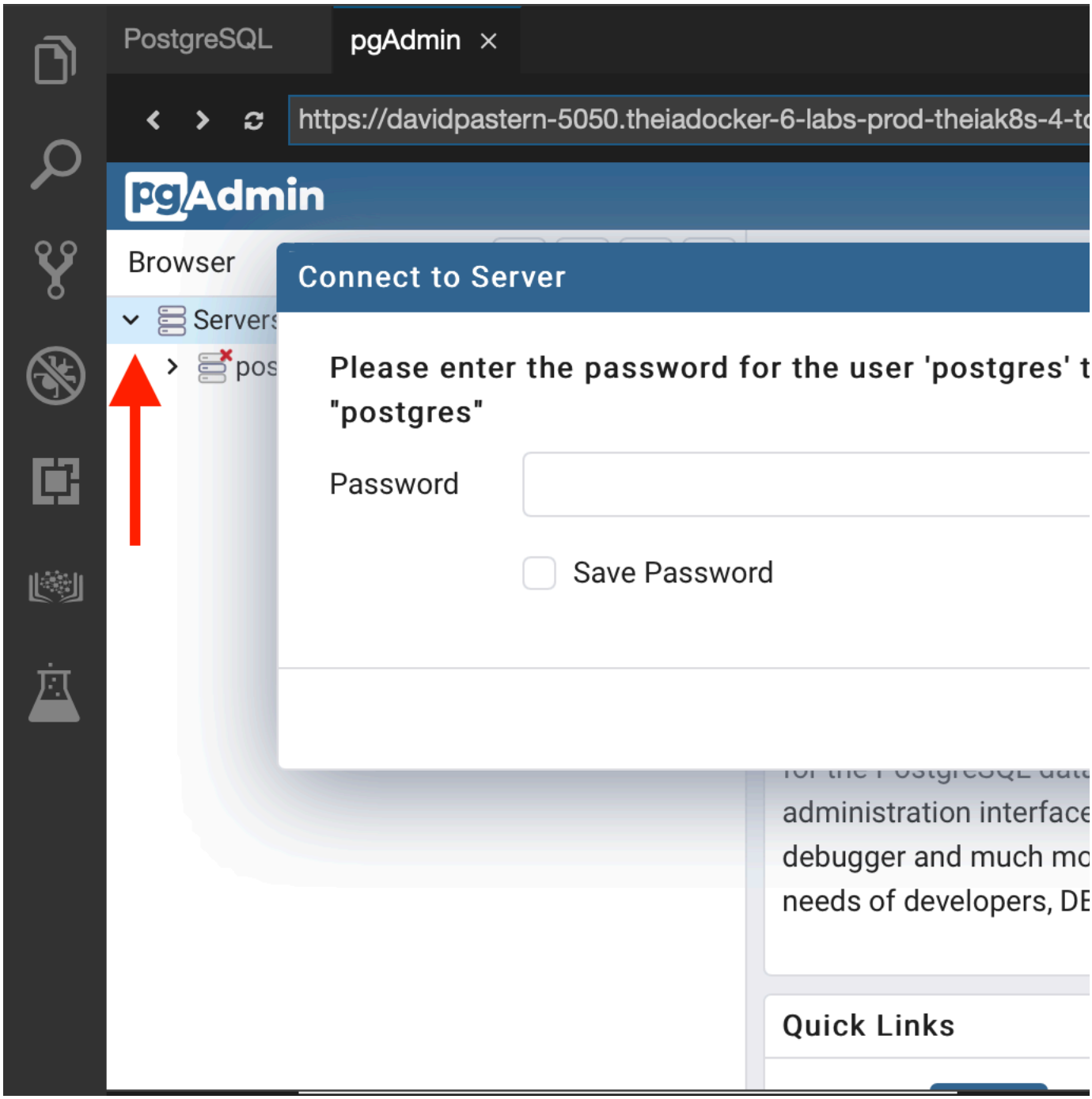
1. Click the Skills Network extension button on the left side of the window.
2. Open the **DATABASES** menu and click **PostgreSQL**.
3. Click **Create**. PostgreSQL may take a few moments to start.



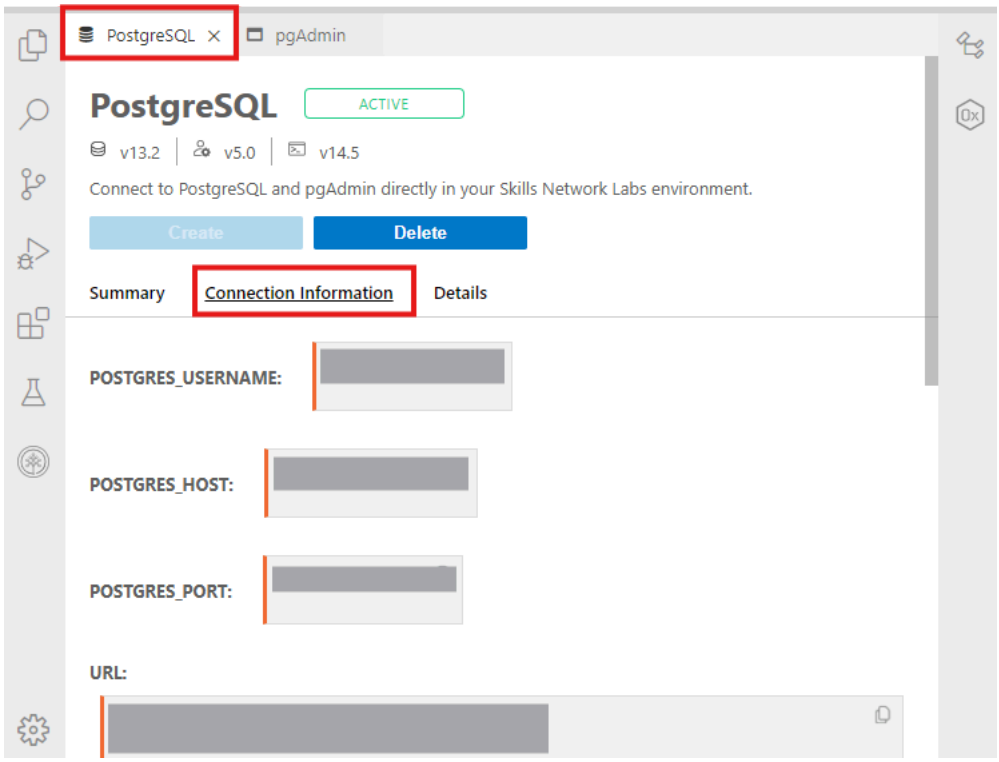
4. Next, open the pgAdmin Graphical User Interface by clicking **pgAdmin** in the Cloud IDE interface.



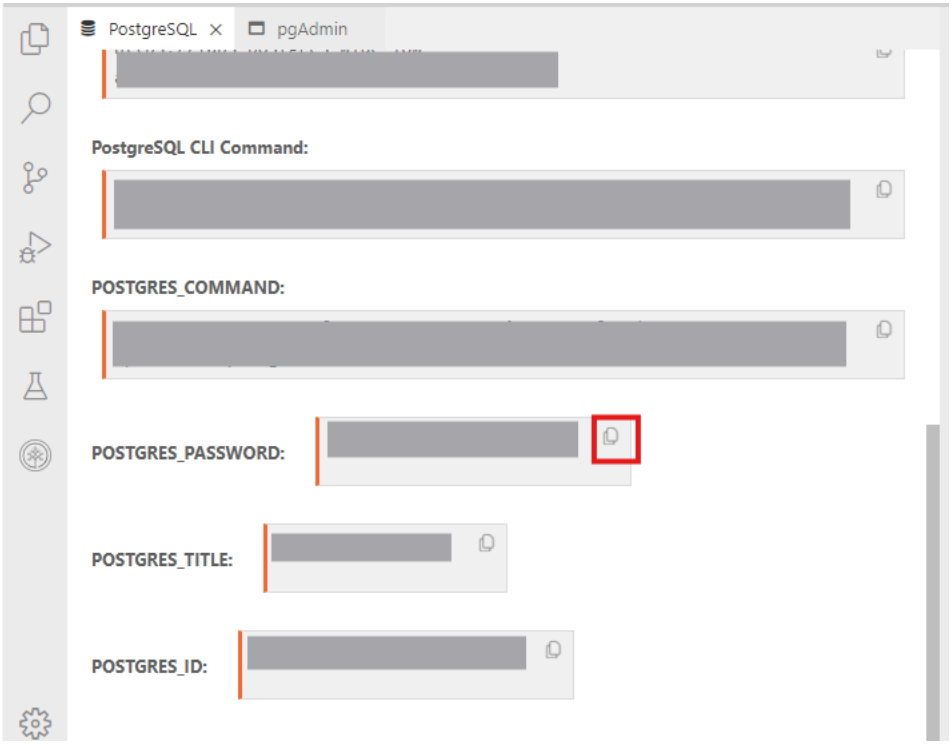
5. Once the pgAdmin GUI opens, click **Servers** tab on the left side of the page. You will be prompted to enter a password.



6. To retrieve your password, click **PostgreSQL** tab near the top of the interface and select **Connection Information** tab.

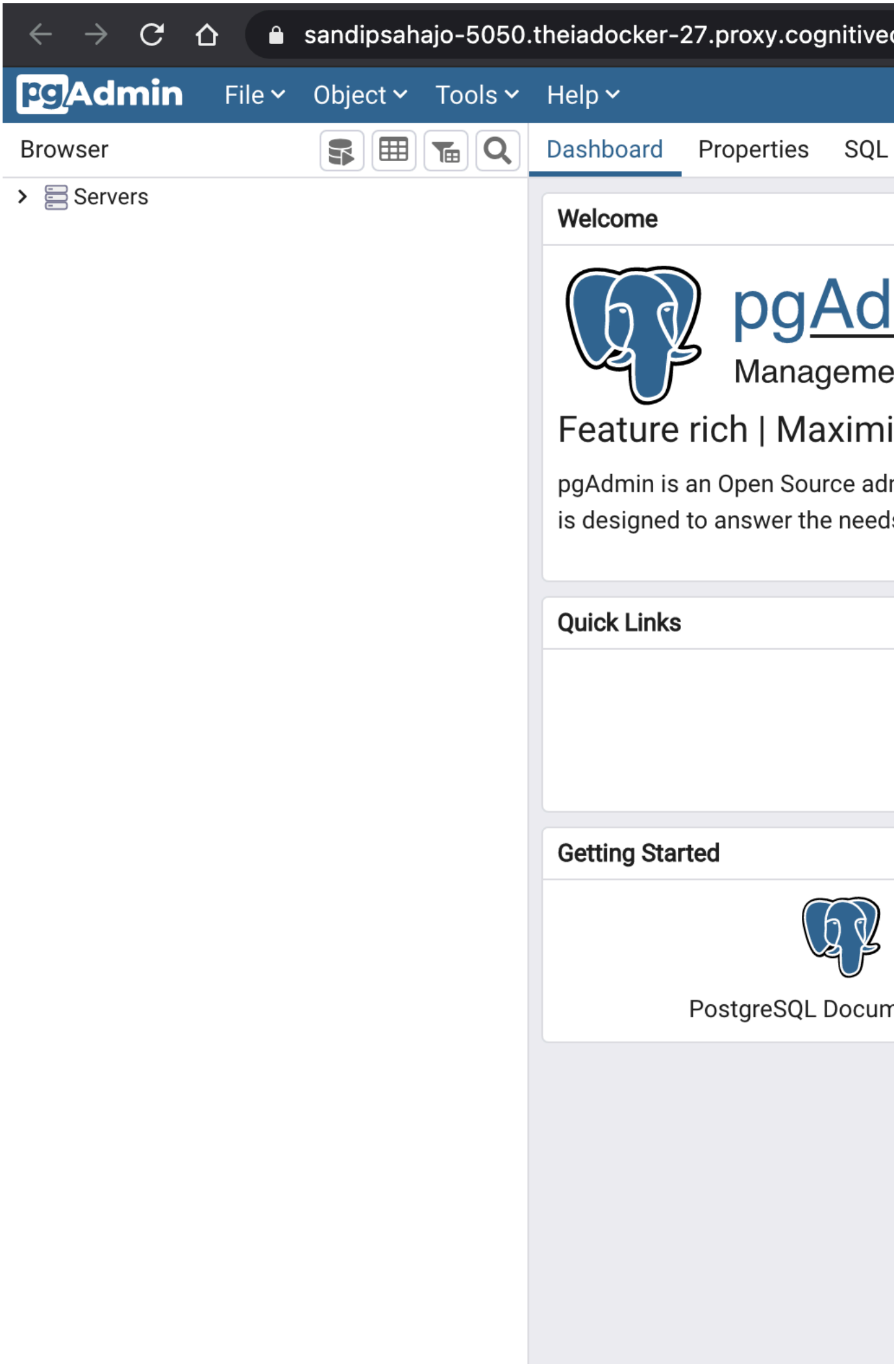


7. Scroll down and click the Copy icon on the left of your password to copy the session password onto your clipboard.

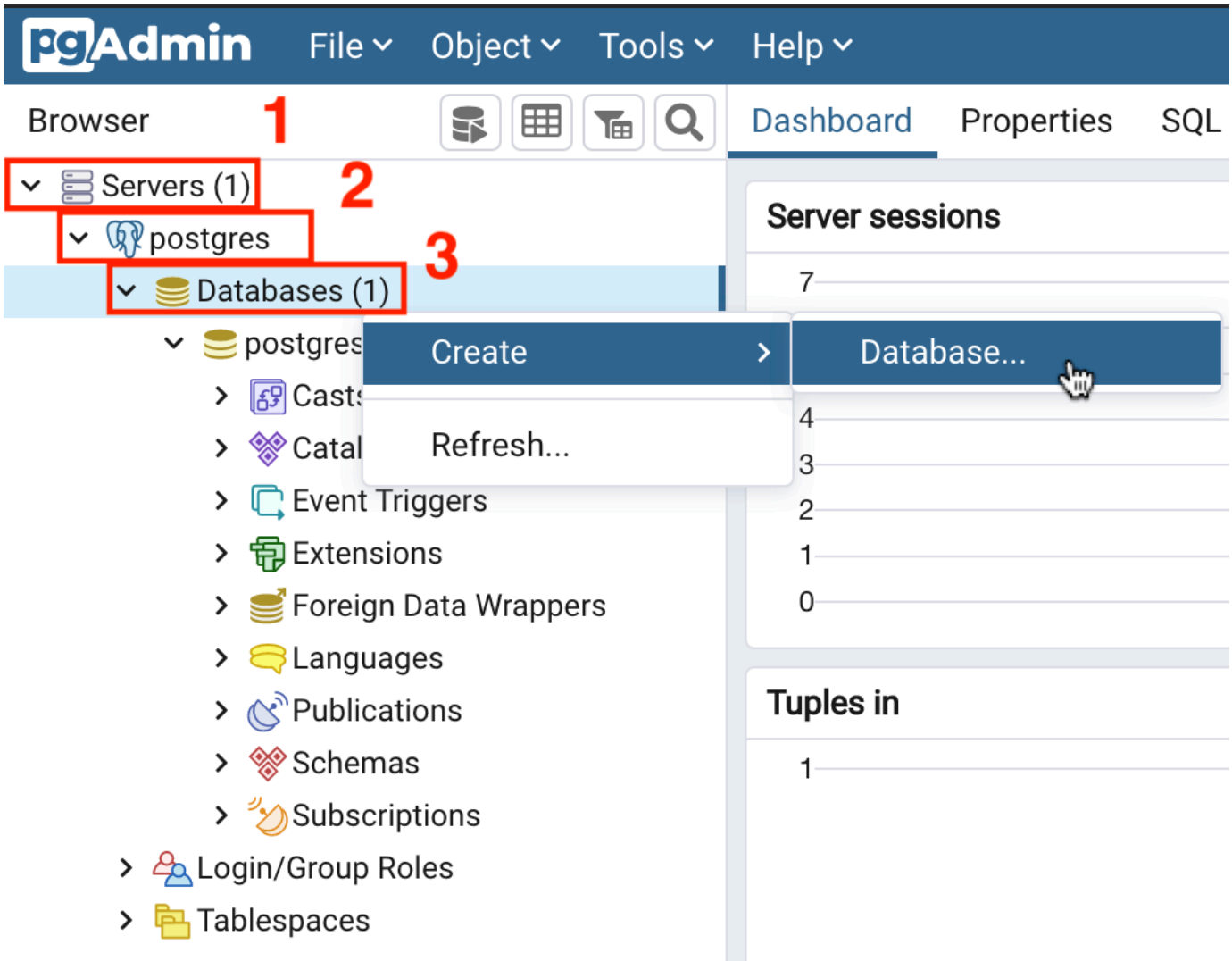


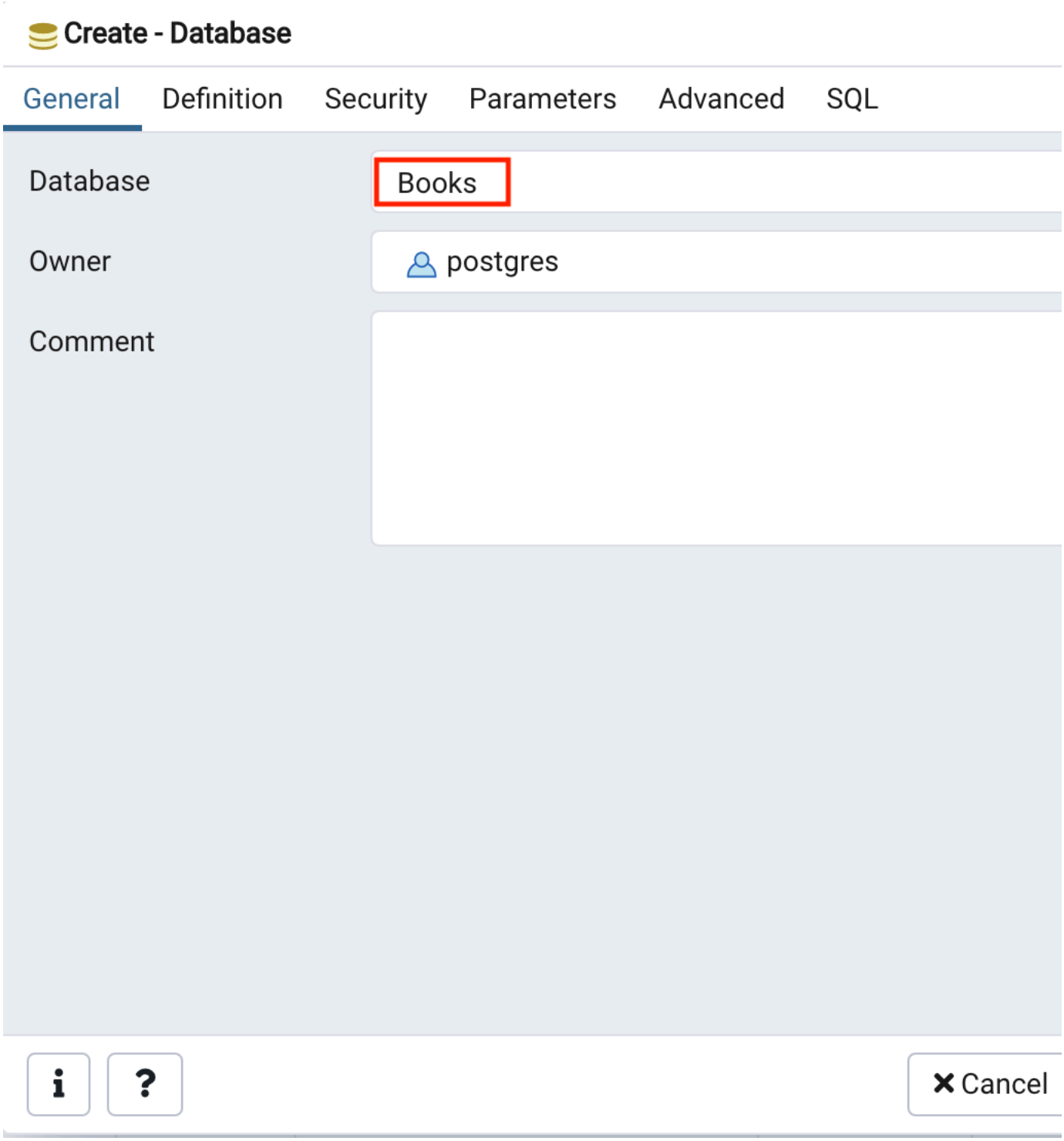
8. Navigate back to the **pgAdmin** tab and paste in your password, then click **OK**.

9. You will then be able to access the pgAdmin GUI tool.


sandipsahajo-5050.theiadocker-27.proxy.cognitive
pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Servers
Welcome
pgAd
Manageme
Feature rich | Maximi
pgAdmin is an Open Source adr
is designed to answer the needs
Quick Links
Getting Started
PostgreSQL Docum

10. In the tree-view, expand **Servers** > **postgres** > **Databases**. If prompted, enter your PostgreSQL service session password. Right-click on **Databases** and go to **Create > Database**. In the **Database** box, type **Books** as the name for your new database, and then click **Save**. Proceed to Task B.

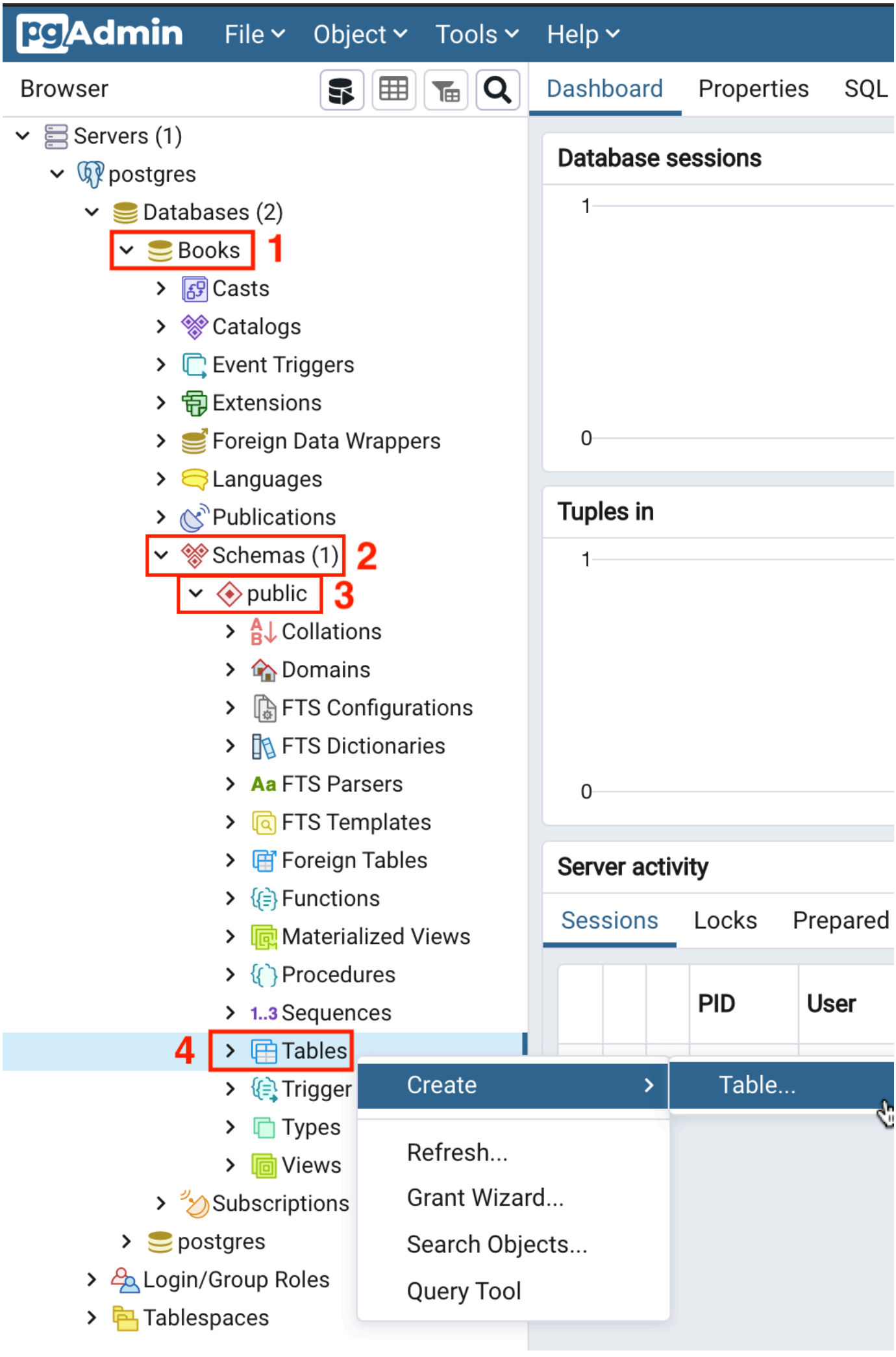




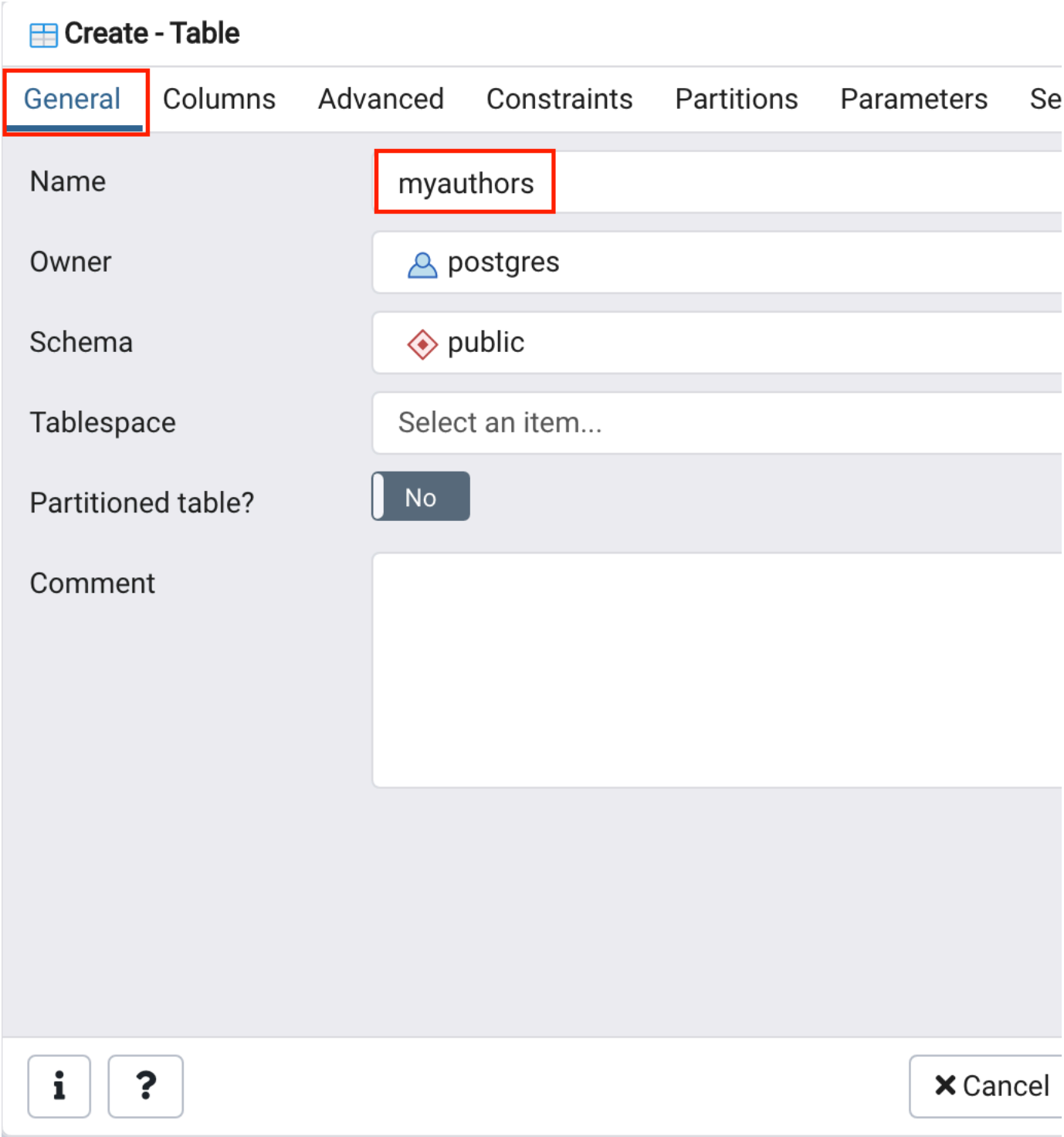
Task B: Create tables

Now that you have your PostgreSQL service active and have created the **Books** database using pgAdmin, let's create a few tables to populate the database and store the data that you wish to eventually upload into it.

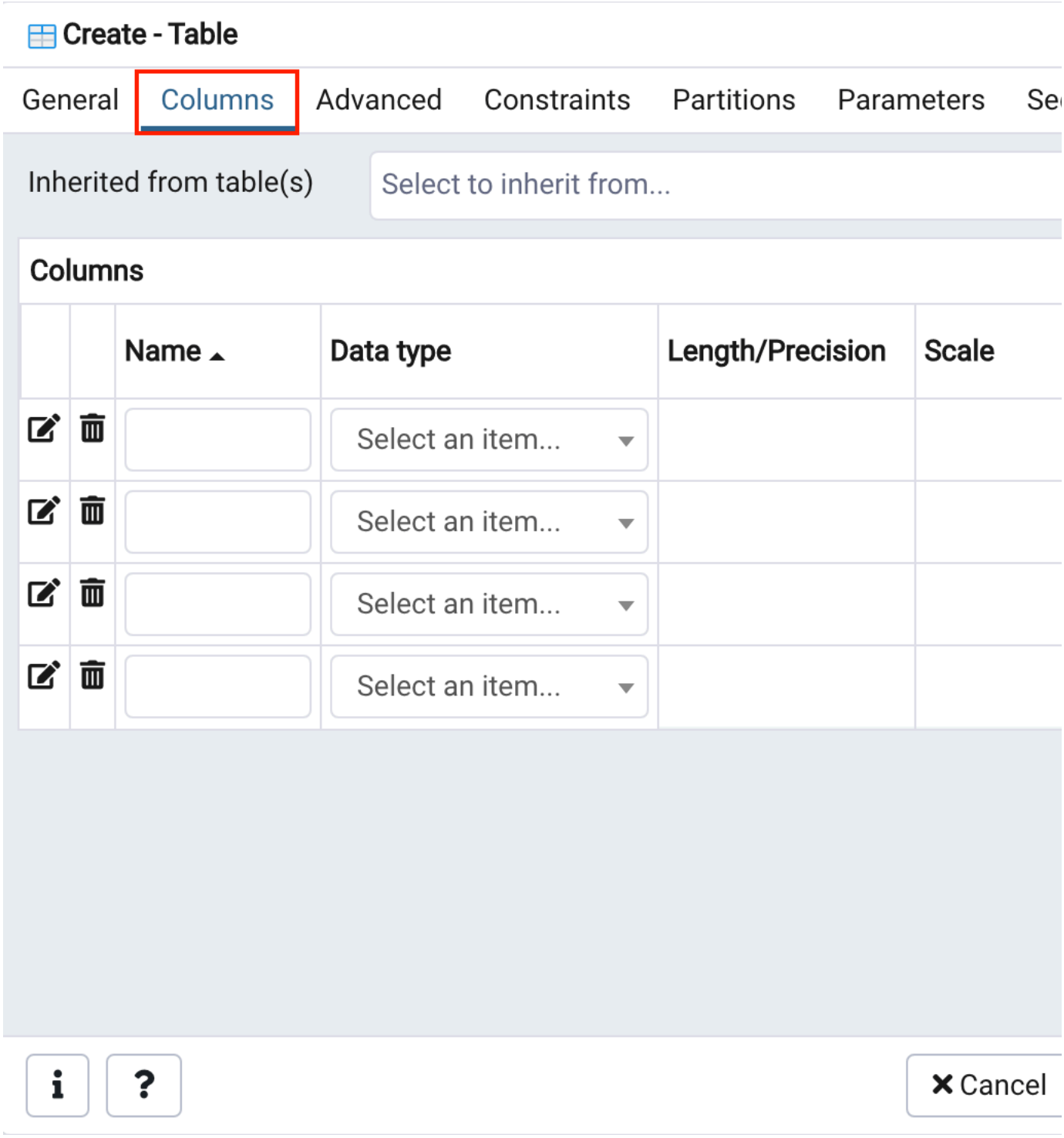
1. In the tree-view, expand **Books** > **Schemas** > **public**. Right-click on **Tables** and go to **Create > Table**.


pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Servers (1)
postgres
Databases (2)
Books
1
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
2
public
3
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Procedures
Sequences
4
Tables
Trigger
Types
Views
Subscriptions
postgres
Login/Group Roles
Tablespaces
Database sessions
1
0
Tuples in
1
0
Server activity
Sessions
Locks
Prepared
PID
User
Create
Table...
Refresh...
Grant Wizard...
Search Objects...
Query Tool

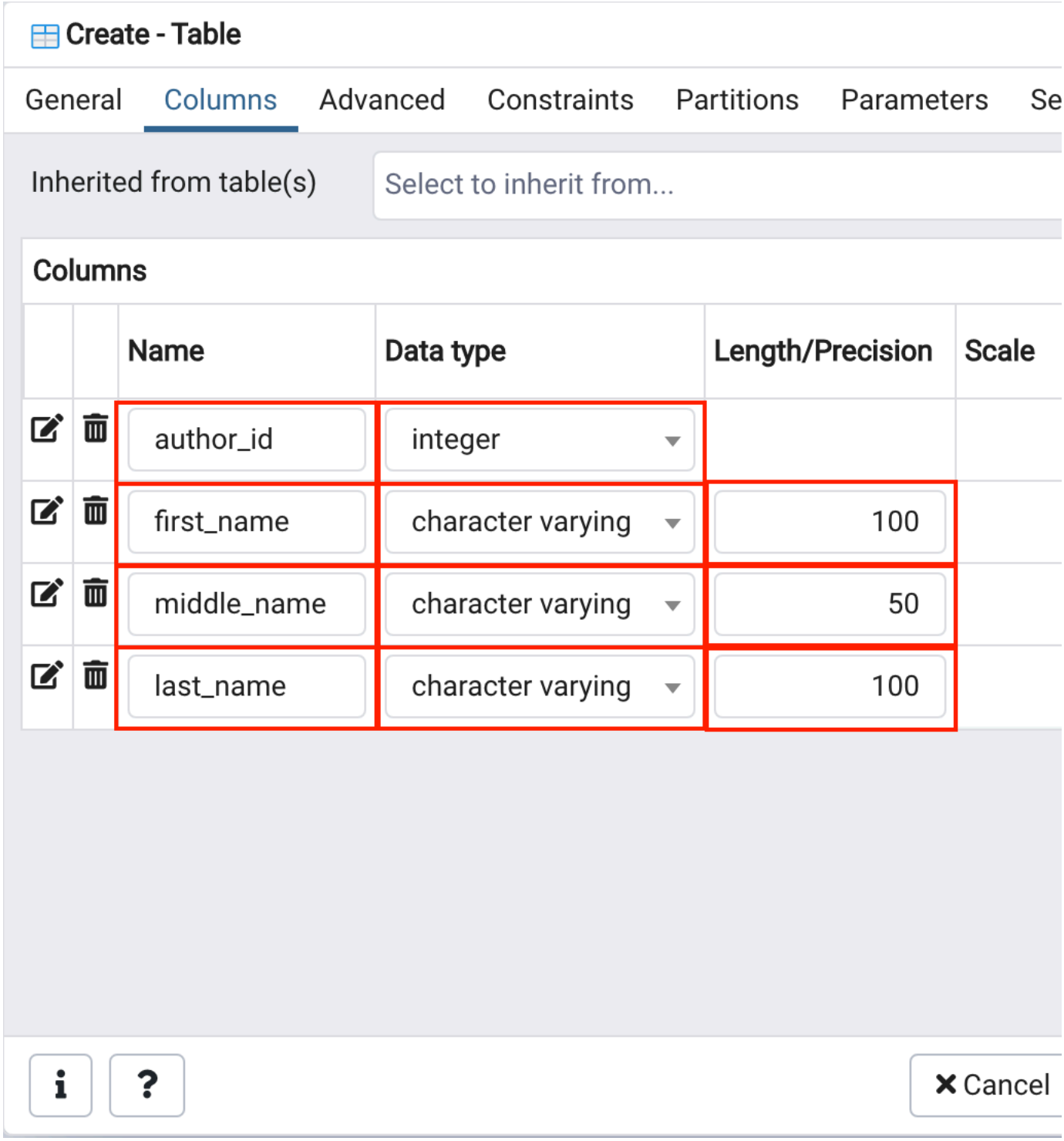
2. On the **General** tab, in the **Name** box, type **myauthors** as name of the table. Don't click **Save**, proceed to the next step.



3. Switch to the tab **Columns** and click the **Add new row** button four times to add 4 column placeholders. Don't click **Save**, proceed to the next step.



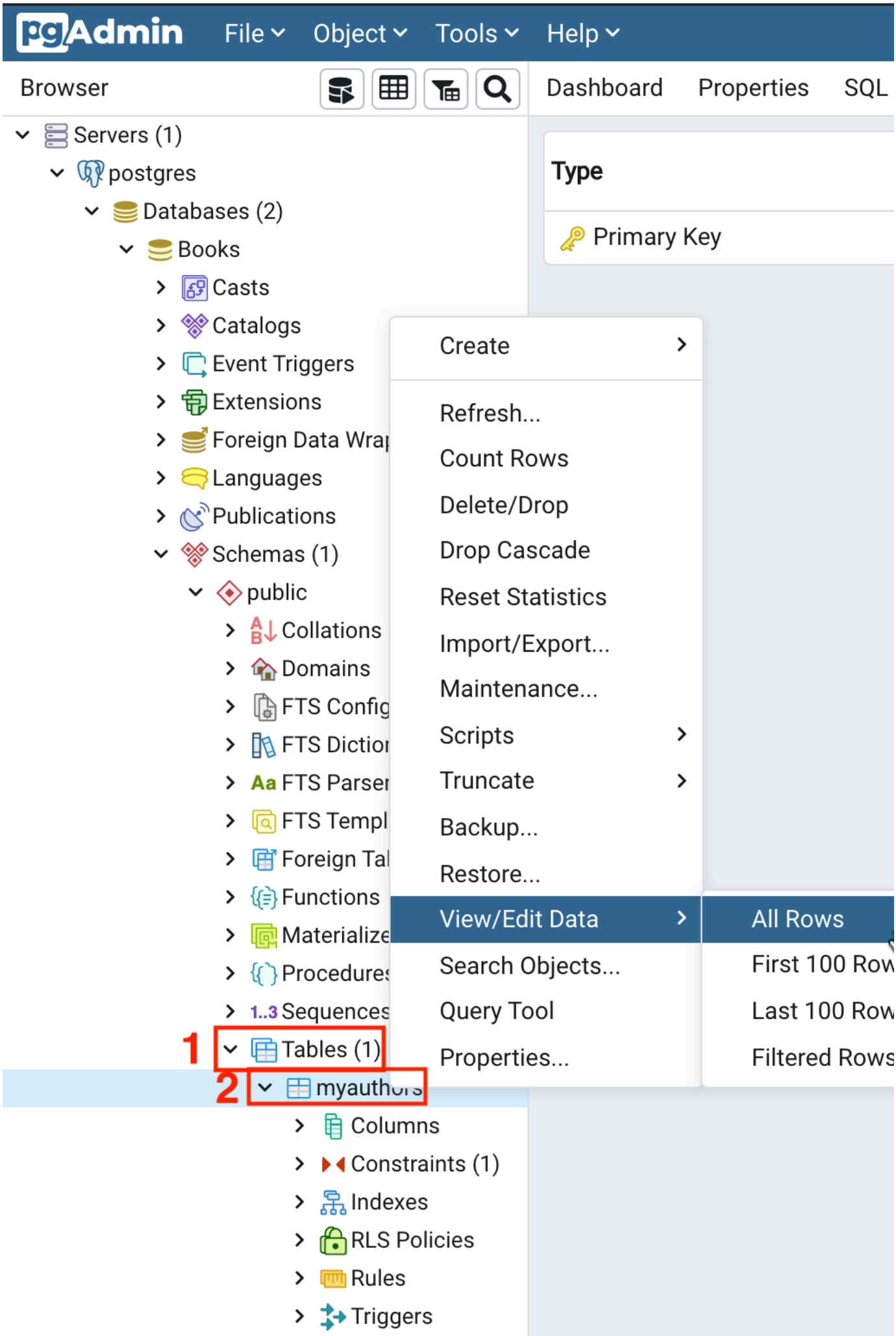
4. Enter the **myauthors** table definition structure information as shown in the image below in the highlighted boxes. Then click **Save**. Proceed to Task C.



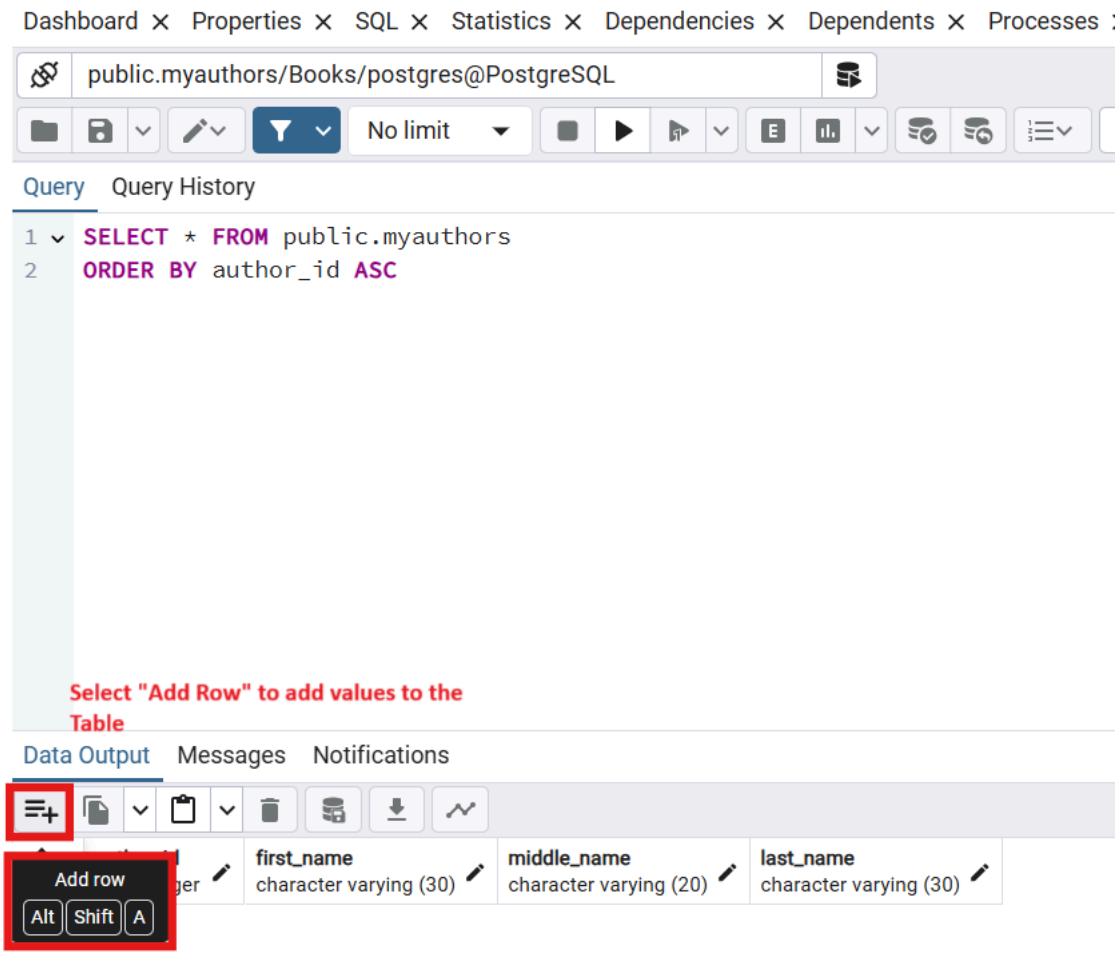
Task C: Load data into tables manually using the pgAdmin GUI

You now have a database and have created tables within it. With the pgAdmin GUI, you can insert values into the tables manually. This is useful if you have a few new entries you wish to add to the database. Let's see how to do it.

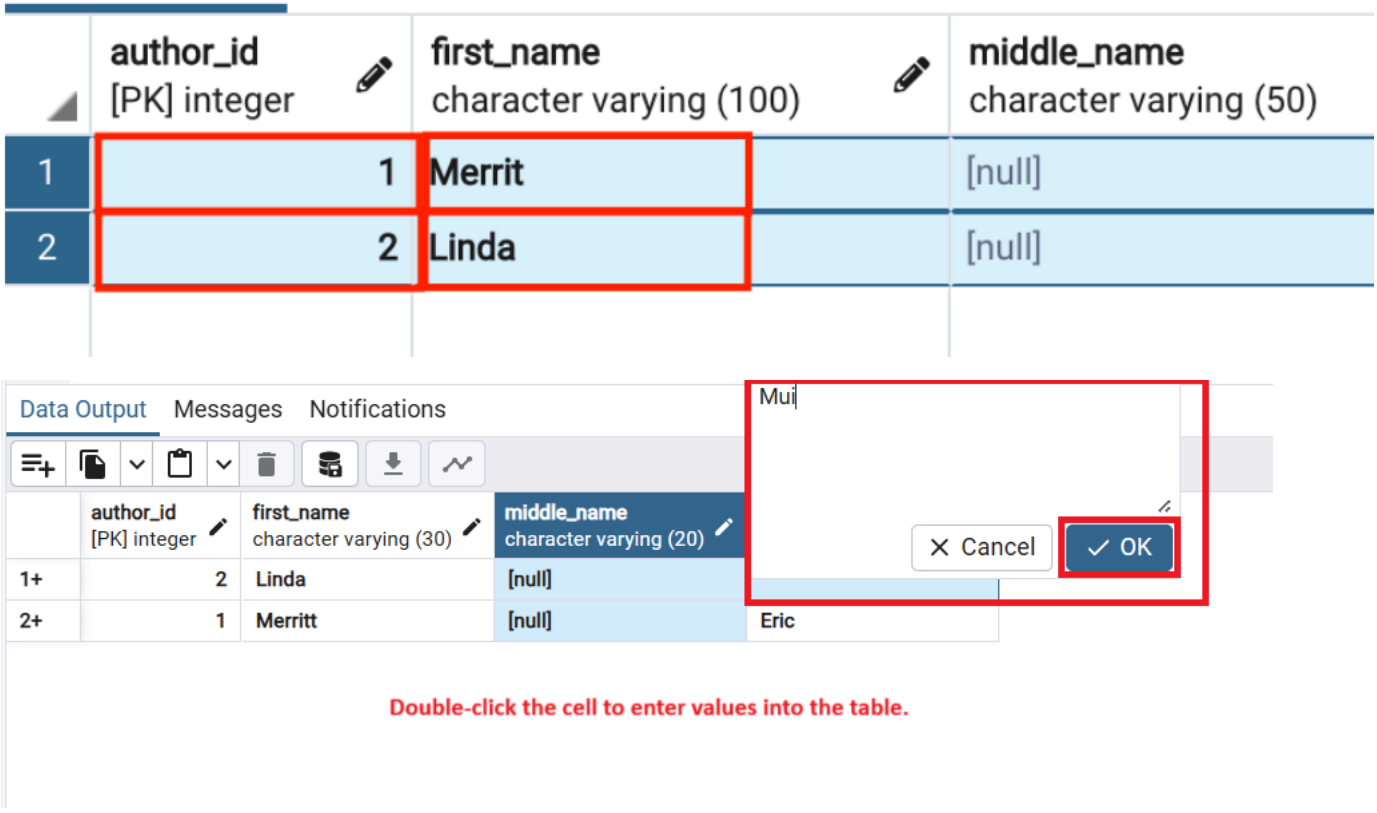
1. In the tree-view, expand **Tables**. Right-click **myauthors** and go to **View/Edit Data > All Rows**.


pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Properties
SQL
Servers (1)
postgres
Databases (2)
Books
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrap
Languages
Publications
Schemas (1)
public
Collations
Domains
FTS Config
FTS Diction
FTS Parser
FTS Templ
Foreign Tal
Functions
Materialize
Procedures
Sequences
1 Tables (1)
2 myauthors
Columns
Constraints (1)
Indexes
RLS Policies
Rules
Triggers
Type
Primary Key
Create
Refresh...
Count Rows
Delete/Drop
Drop Cascade
Reset Statistics
Import/Export...
Maintenance...
Scripts
Truncate
Backup...
Restore...
View/Edit Data
Search Objects...
Query Tool
Properties...
All Rows
First 100 Row
Last 100 Row
Filtered Rows

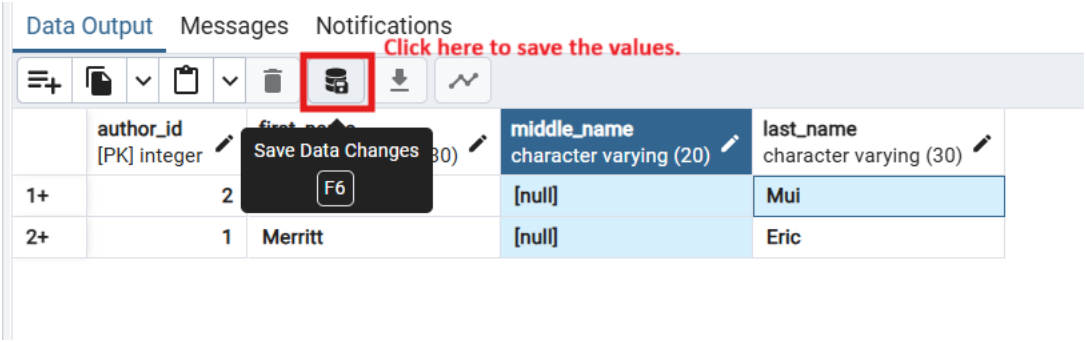
2. You will insert 2 rows of data into the **myauthors** table. In the lower **Data Output** pane, enter **myauthors** table data information for 2 rows as shown in the highlighted boxes in the image below. Then click the **Save Data Changes** icon. Proceed to Task D.



3. Enter the values into the table as shown below:



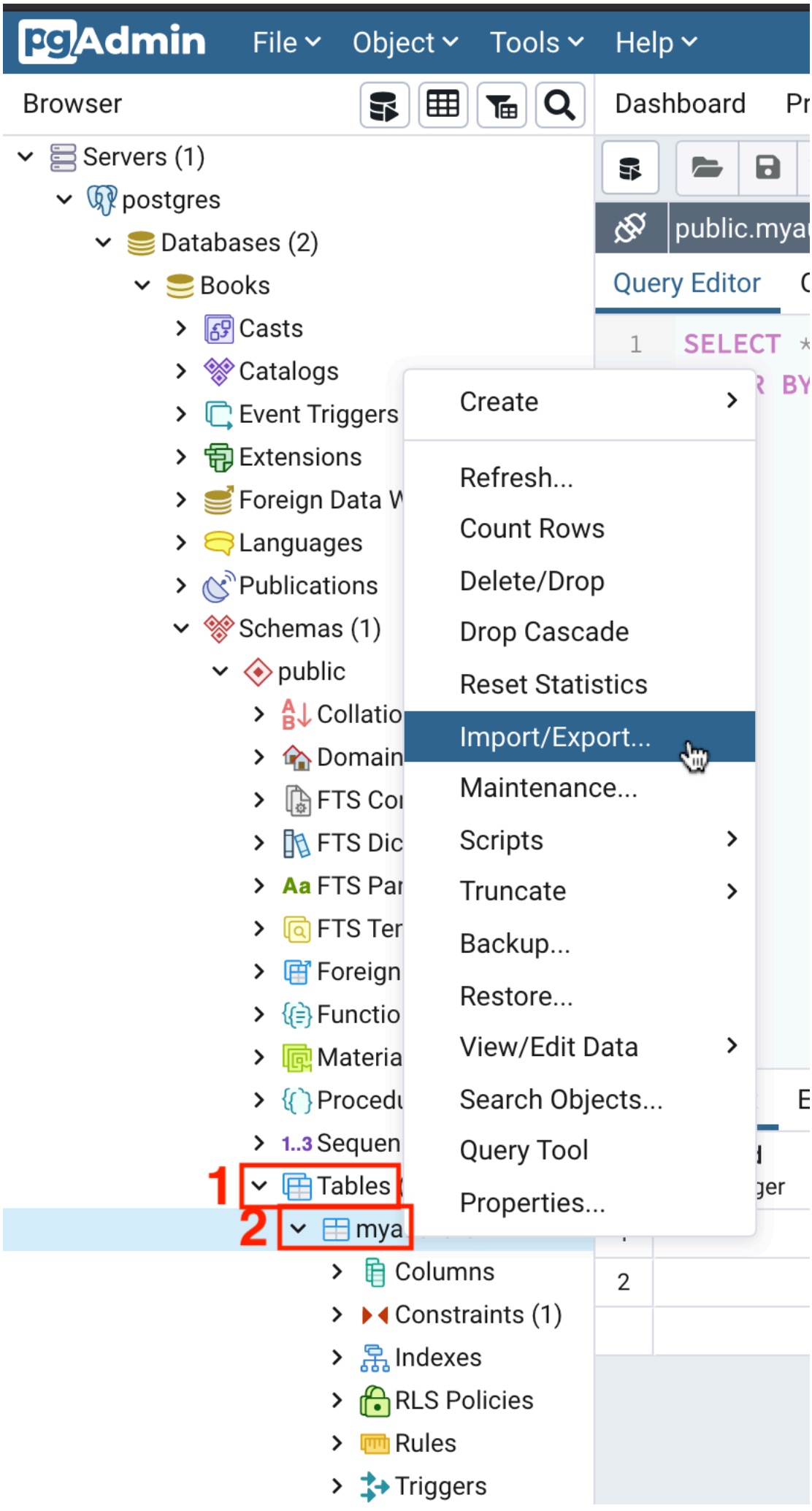
4. Save the values.



Task D: Load data into tables using a text/script file

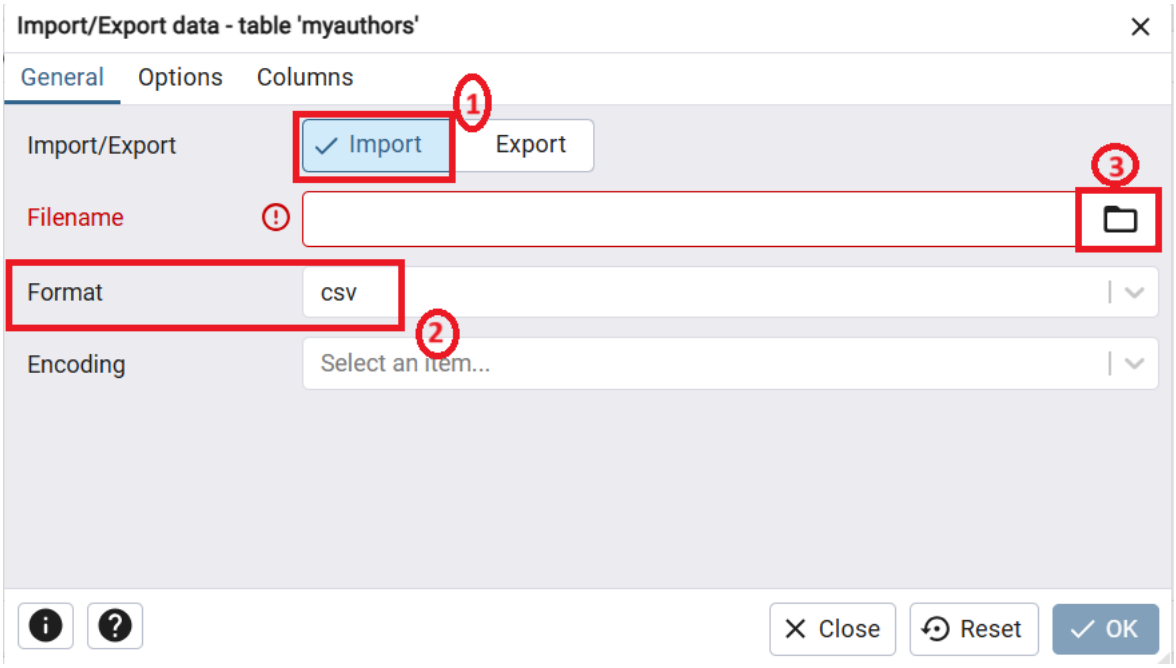
In the previous task, you entered some data entries into a table manually with pgAdmin. While this method can be useful for small additions, if you wish to upload large amounts of data at once, the process becomes tedious. An alternative is to load data into tables from a text or script file containing the data you wish to enter. Let's take a look at how to do this.

1. You will import the remainder of the **myauthors** table data from a csv text file. Download the csv file below to your local computer:
    - myauthors.csv
2. In the tree-view, right-click on **myauthors** and go to **Import/Export**.


pgAdmin
File
Object
Tools
Help
Browser
Dashboard
Servers (1)
postgres
Databases (2)
Books
Casts
Catalogs
Event Triggers
Extensions
Foreign Data W
Languages
Publications
Schemas (1)
public
Collatio
Domain
FTS Co
FTS Dic
FTS Par
FTS Ter
Foreign
Functio
Materia
Procedu
Sequen
1 Tables
2 mya
Columns
Constraints (1)
Indexes
RLS Policies
Rules
Triggers
Create
Refresh...
Count Rows
Delete/Drop
Drop Cascade
Reset Statistics
Import/Export...
Maintenance...
Scripts
Truncate
Backup...
Restore...
View/Edit Data
Search Objects...
Query Tool
Properties...
public.mya
Query Editor
1 SELECT
2

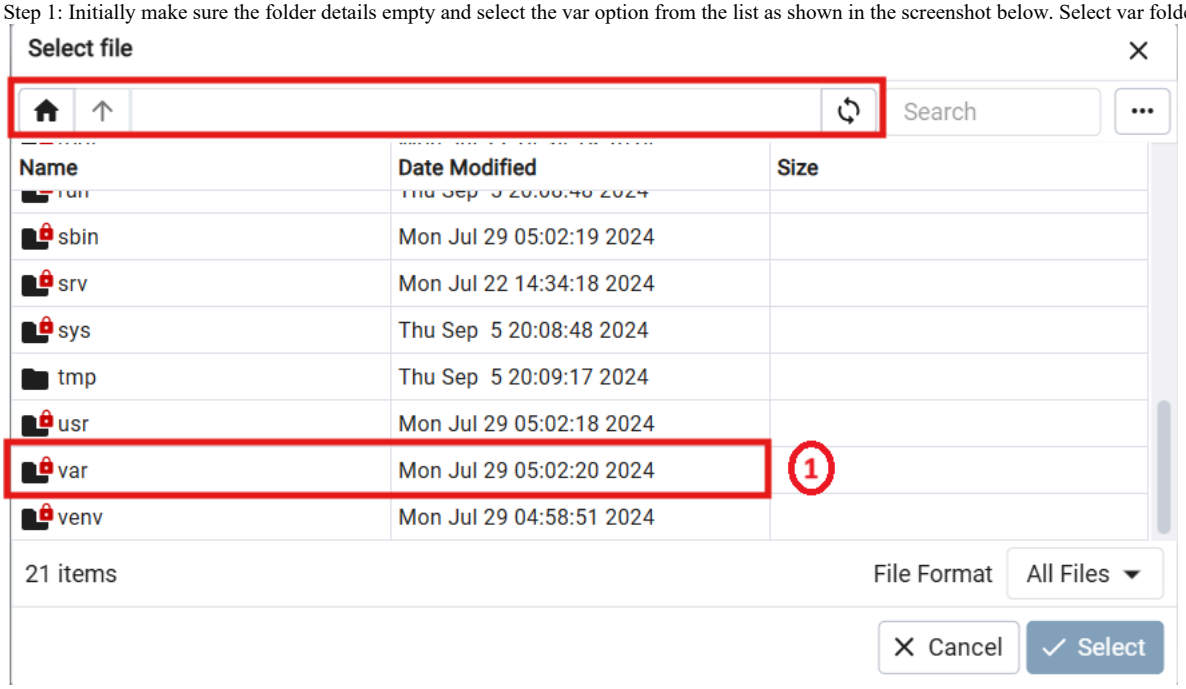
3. Follow the instructions below to import:

    1. Make sure **Import/Export** is set to **Import**,
    2. **Format** = **csv**.
    3. Then click **Select file** icon by the **Filename** box.

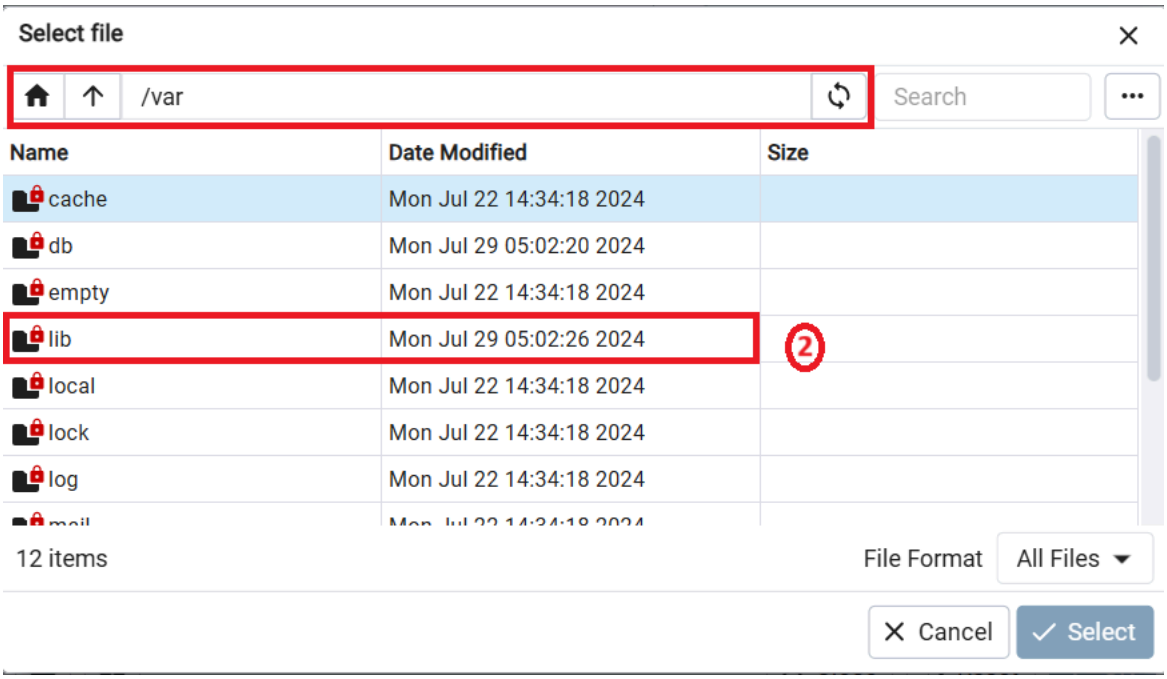


4. Steps to **Upload File**.

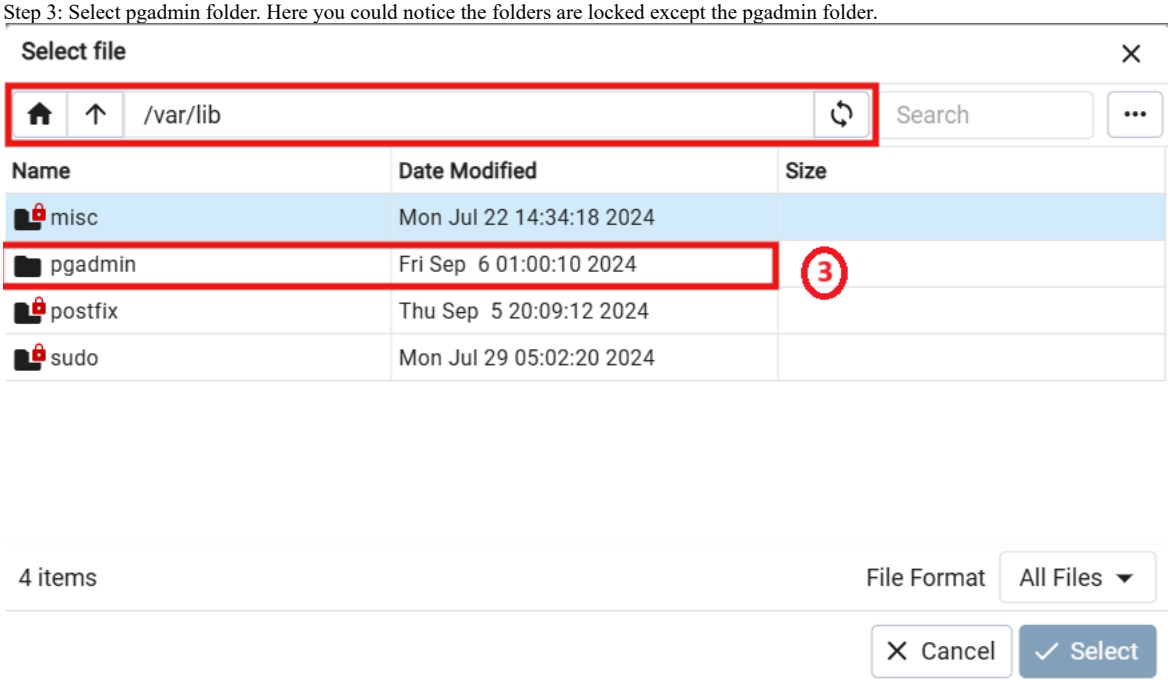
- Step 1: Initially make sure the folder details empty and select the var option from the list as shown in the screenshot below. Select var folder



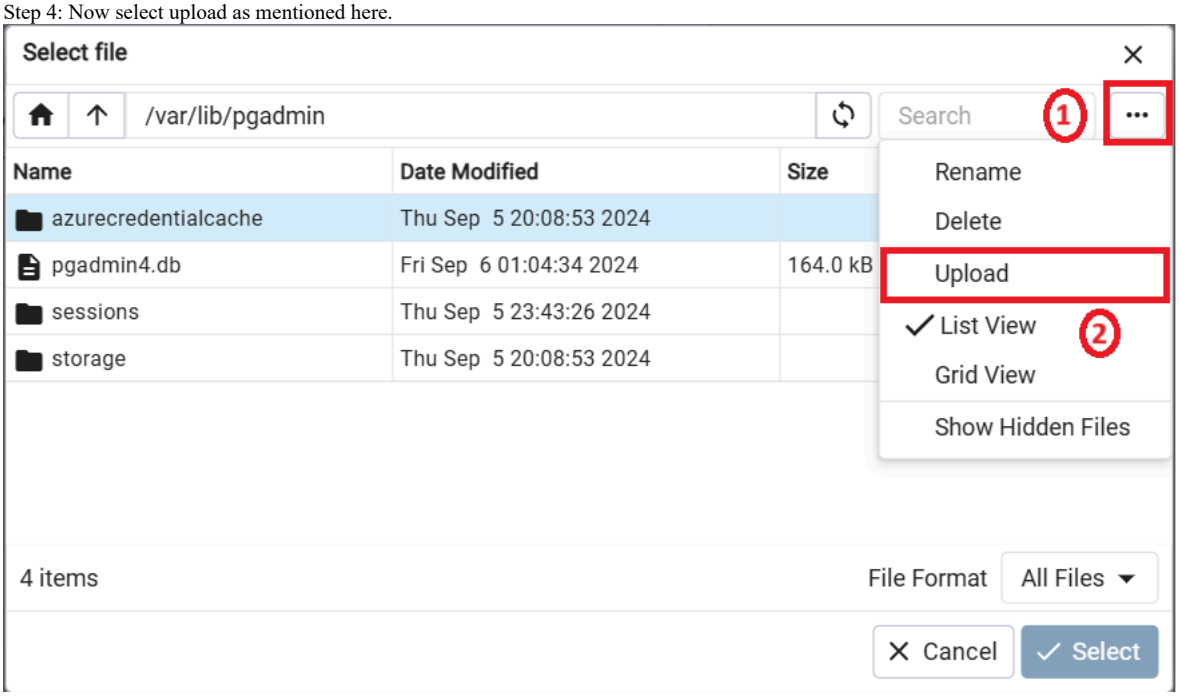
- Step 2: Select lib folder.



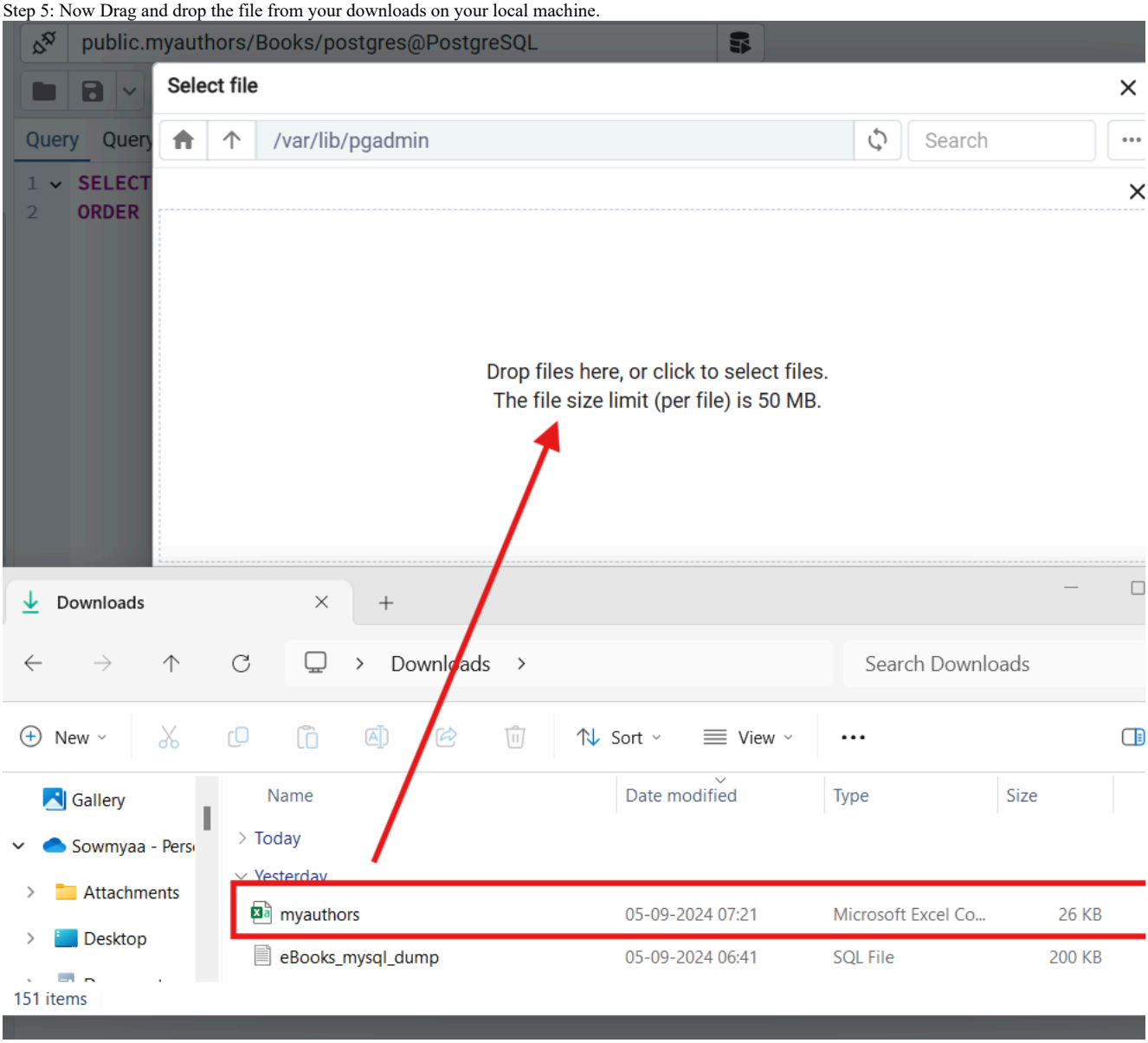
- Step 3: Select pgadmin folder. Here you could notice the folders are locked except the pgadmin folder.



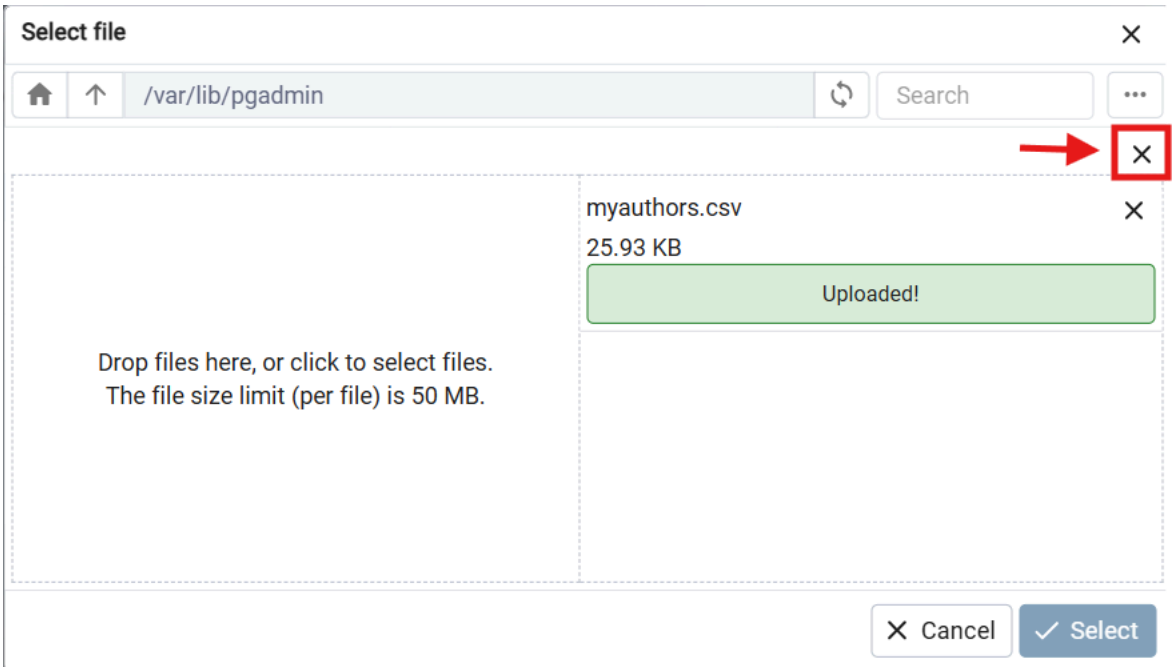
- Step 4: Now select upload as mentioned here.



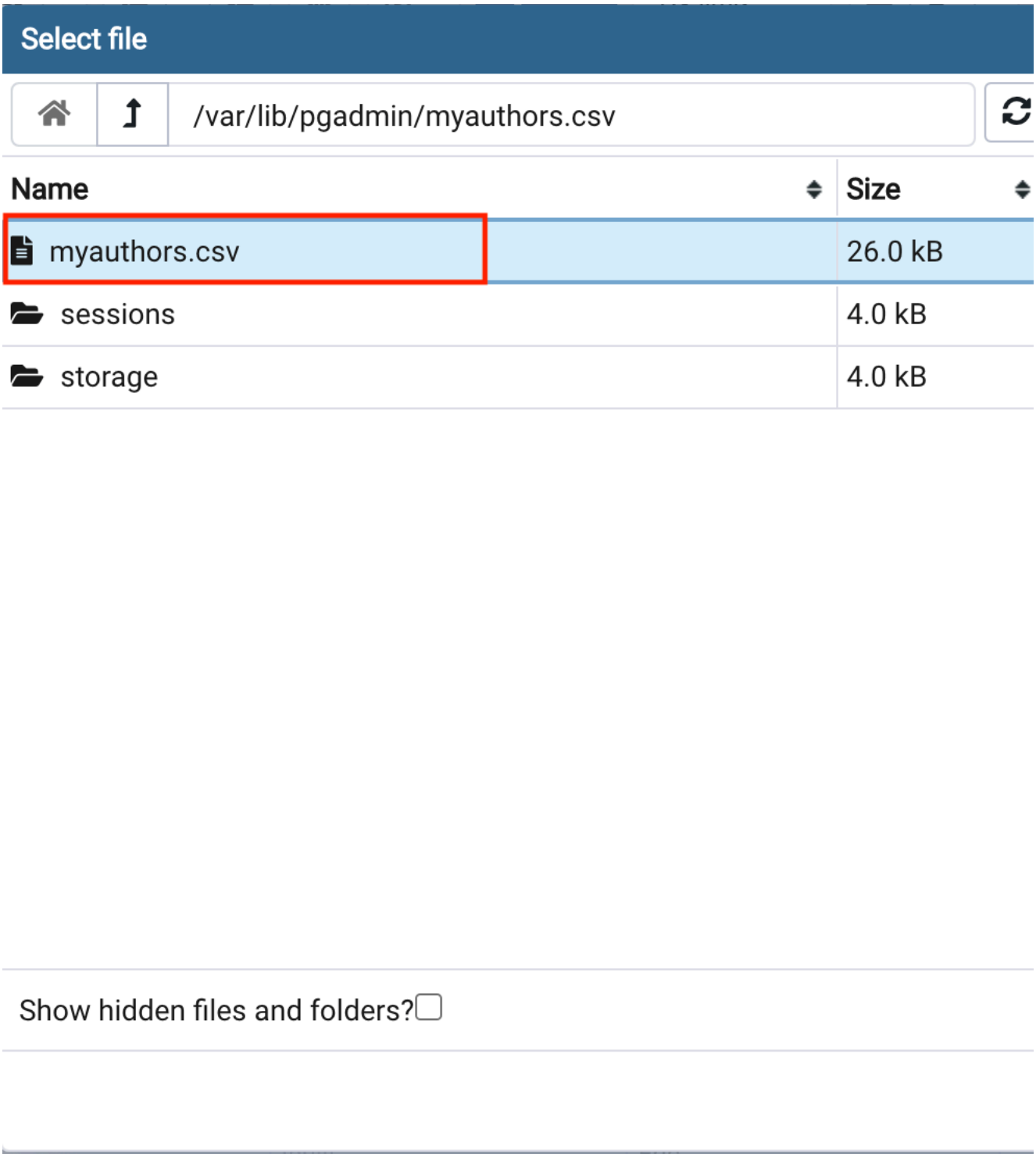
- Step 5: Now Drag and drop the file from your downloads on your local machine.



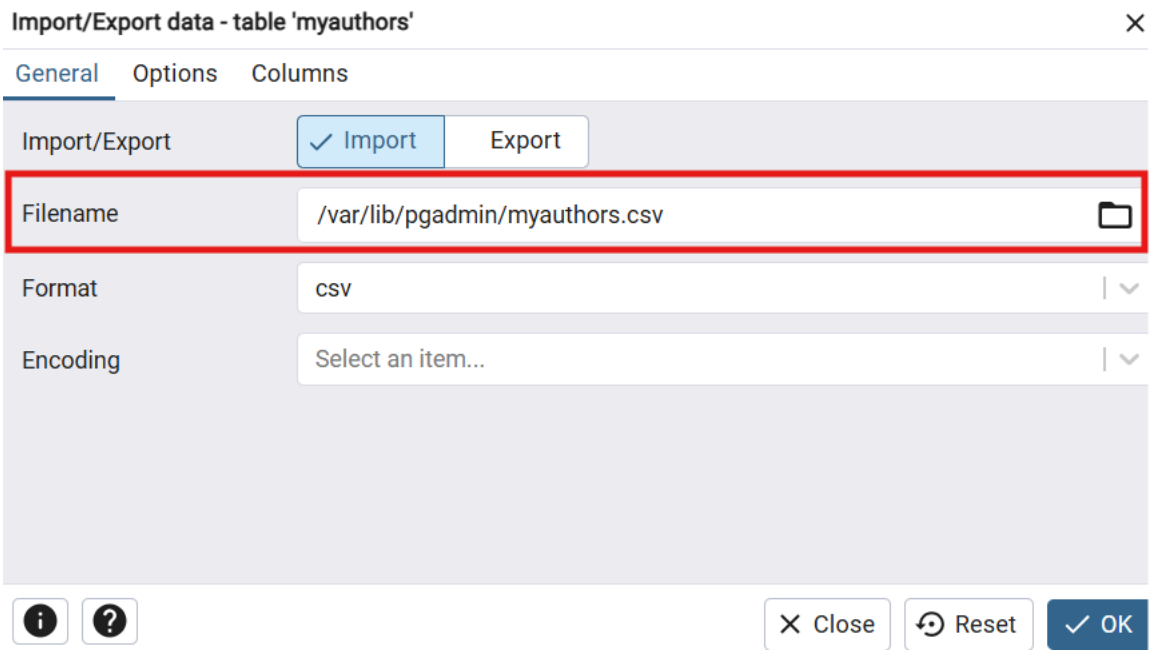
- Step 6: Finally, the upload is successful. When the upload is complete, close the drop files area by clicking **X**.



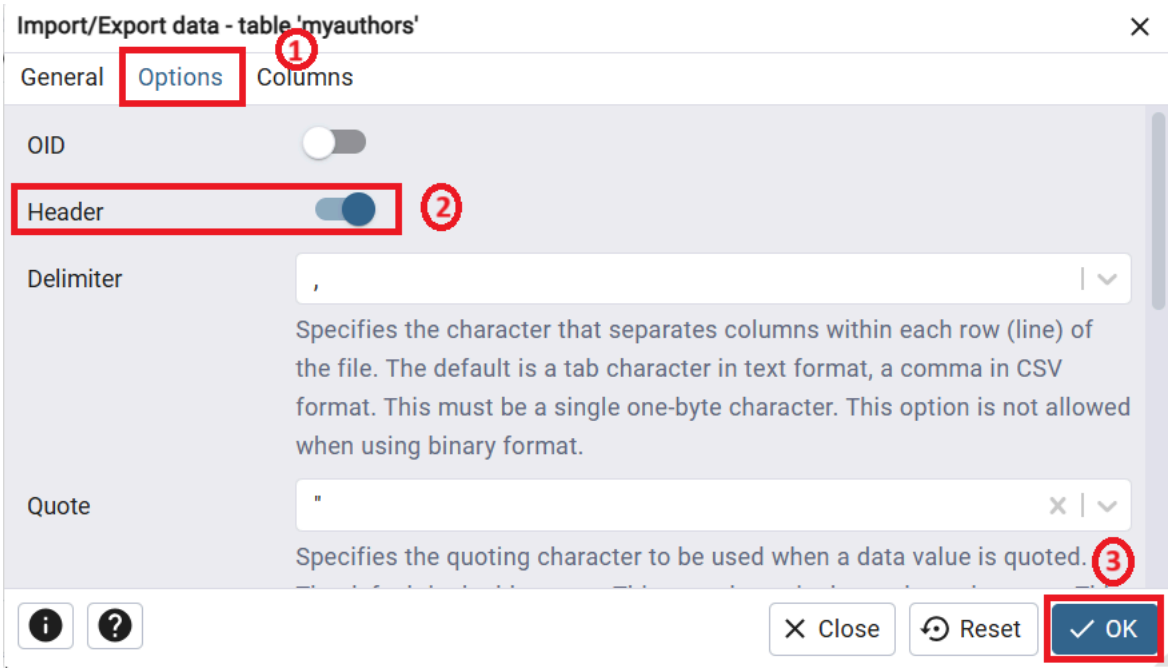
- Select the uploaded **myauthors.csv** file from the list and click **Select**.

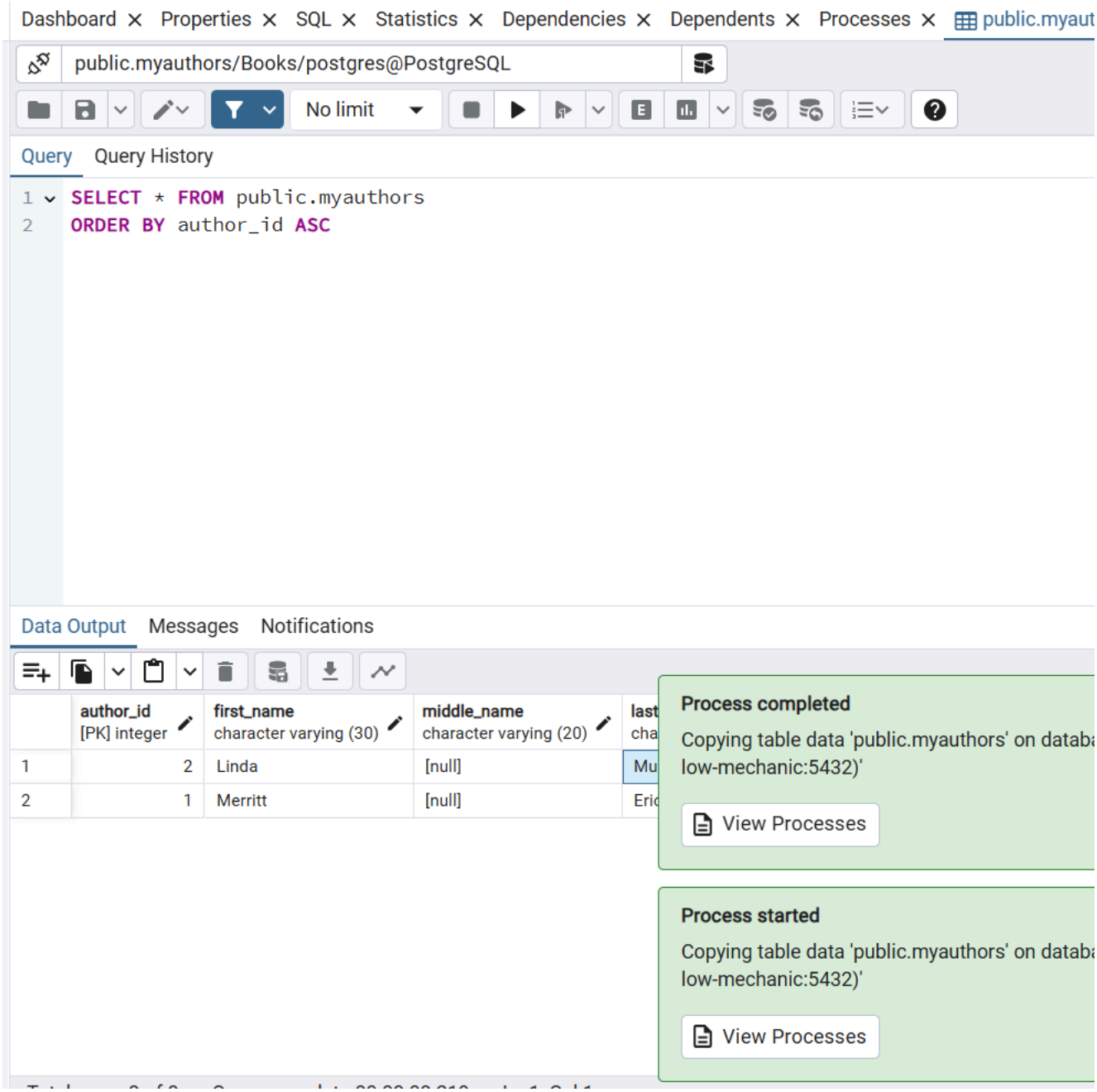

Select file
/var/lib/pgadmin/myauthors.csv
Name
Size
myauthors.csv
26.0 kB
sessions
4.0 kB
storage
4.0 kB
Show hidden files and folders?

- Ensure the file has selected.



- Under **Options** enable **Header** and Click OK and notification of import success will appear.





4. Repeat Task C Step 1 to check that the newly imported data rows appear along with your previously inserted 2 rows.

Dashboard Properties SQL Statistics Dependencies Dependents

public.myauthors/Books/postgres@postgres

Query Editor Query History

```
1  SELECT * FROM public.myauthors
2  ORDER BY author_id ASC
```

Data Output Explain Messages Notifications

	author_id [PK] integer	first_name character varying (100)	middle_name character varying (50)
1	1	Merrit	[null]
2	2	Linda	[null]
3	3	Alecos	[null]
4	4	Paul	C.van
5	5	David	[null]
6	6	Richard	[null]
7	7	Yuval	Noah
8	8	Paul	[null]
9	9	David	[null]
10	10	John	Paul
11	11	Andrew	[null]
12	12	Melanie	[null]
13	13	Neal	[null]
14	14	Nir	[null]
15	15	Tim	[null]
16	16	Mike	[null]
17	17	Brian	P.
18	18	Jean-Philippe	[null]
19	19	Lance	[null]
20	20	Richard	C.
21	21	William	L.

22	22	Magnus	Lie
23	23	Mike	[null]
24	24	Norman	[null]
25	25	John	E.
26	26	S.	[null]

As you can see, the data contained in the **csv** file was successfully uploaded into the table and you did not have to manually input hundreds of entries.

Conclusion

Congratulations! You have completed this lab, and you have learned how to create databases and tables in a PostgreSQL instance, load data into tables manually using the pgAdmin GUI, and load data into tables from a text/script file.

Author

- Sandip Saha Joy

Other Contributors

- David Pasternak