Hands-on Lab: Populating a Data Warehouse using PostgreSQL

Estimated time needed: 1 hour

Purpose of the Lab:

The lab is designed to provide hands-on experience in creating and managing a production database using PostgreSQL within the IBM Skills Network Labs (SN Labs) Cloud IDE. You will learn how to launch a PostgreSQL server instance, utilize the pgAdmin graphical user interface (GUI) for database operations, and execute essential tasks like creating a database, designing tables, and loading data. The lab focuses on building a foundation in database management by guiding learners through the process of setting up a 'Production' database and populating it with data following a star schema design.

Benefits of Learning the Lab:

Engaging in this lab offers significant benefits for learners seeking to deepen their understanding of database management systems, particularly PostgreSQL. By working through the lab, you will gain practical skills in SQL, database creation, table design, and data manipulation, which are crucial for roles in data engineering, database administration, and data science. The hands-on approach helps in consolidating knowledge of database schemas and SQL queries, thereby enhancing the learner's ability to manage and analyze data effectively in real-world scenarios. Additionally, familiarity with tools like pgAdmin and the Cloud IDE environment adds valuable experience to your skill set, preparing you for advanced database projects and tasks.

Software Used in this Lab

To complete this lab you will utilize the PostgreSQL Database 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

Production database is used in this lab.
The production database contains:

- DimCustomer
- DimMonth
- FactBilling

Objectives

In this lab you will:

- Create production related database and tables in a PostgreSQL instance.
- Populate the production data warehouse byloading the tables from Scripts.

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.

Data Used in this Lab

The following are the SQL data files used in this lab.

The production database contains:

- DimCustomer
- DimMonth
- FactBilling
- Star Schema

Task A: Create a database

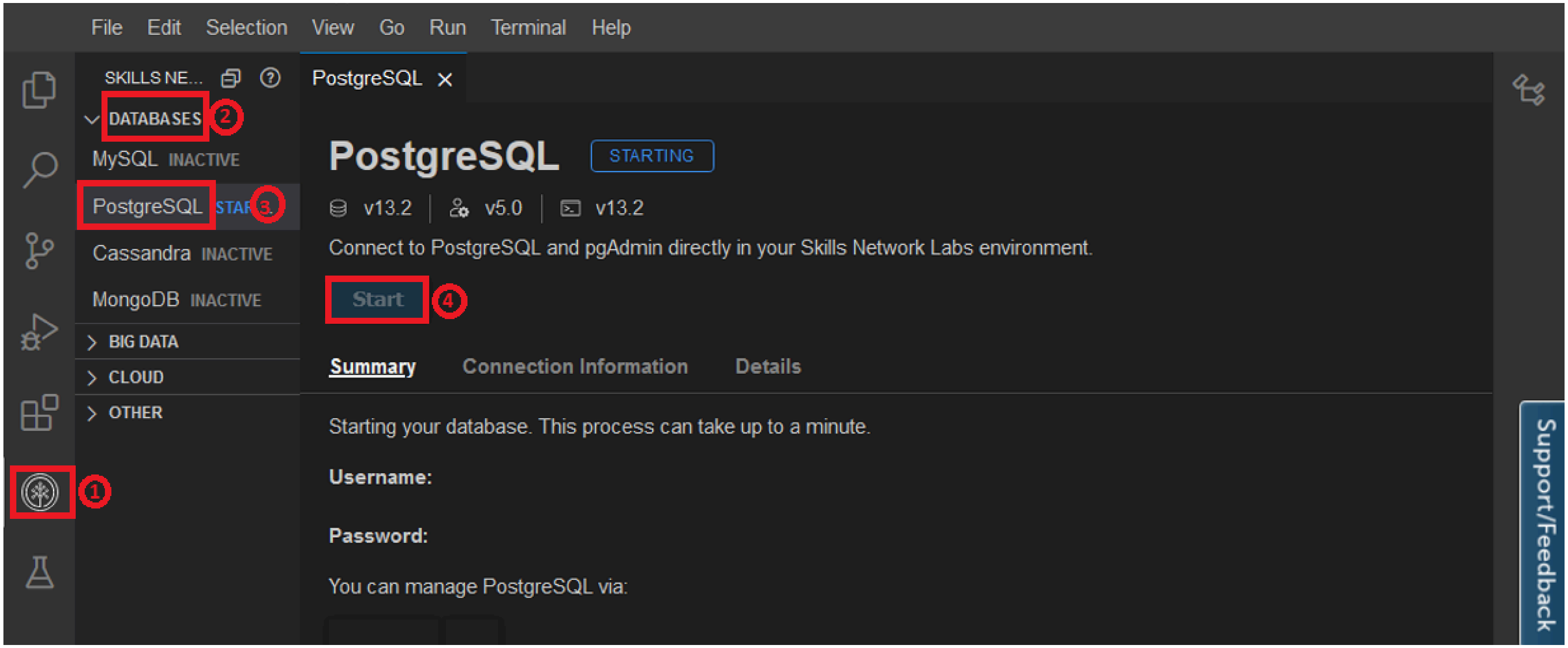
First, to create a database on a PostgreSQL server instance, you'll first want to actually launch a PostgreSQL server instance on Cloud IDE and open up the pgAdmin Graphical User Interface.

1. Click on the Skills Network extension button on the left side of the window.

2. Open the **DATABASES** drop down menu.

3. Click on **PostgreSQL**

4. Click on the **Start** button. PostgreSQL may take a few moments to start.



Connect to PostgreSQL and pgAdmin directly in your Skills Network Labs environment.

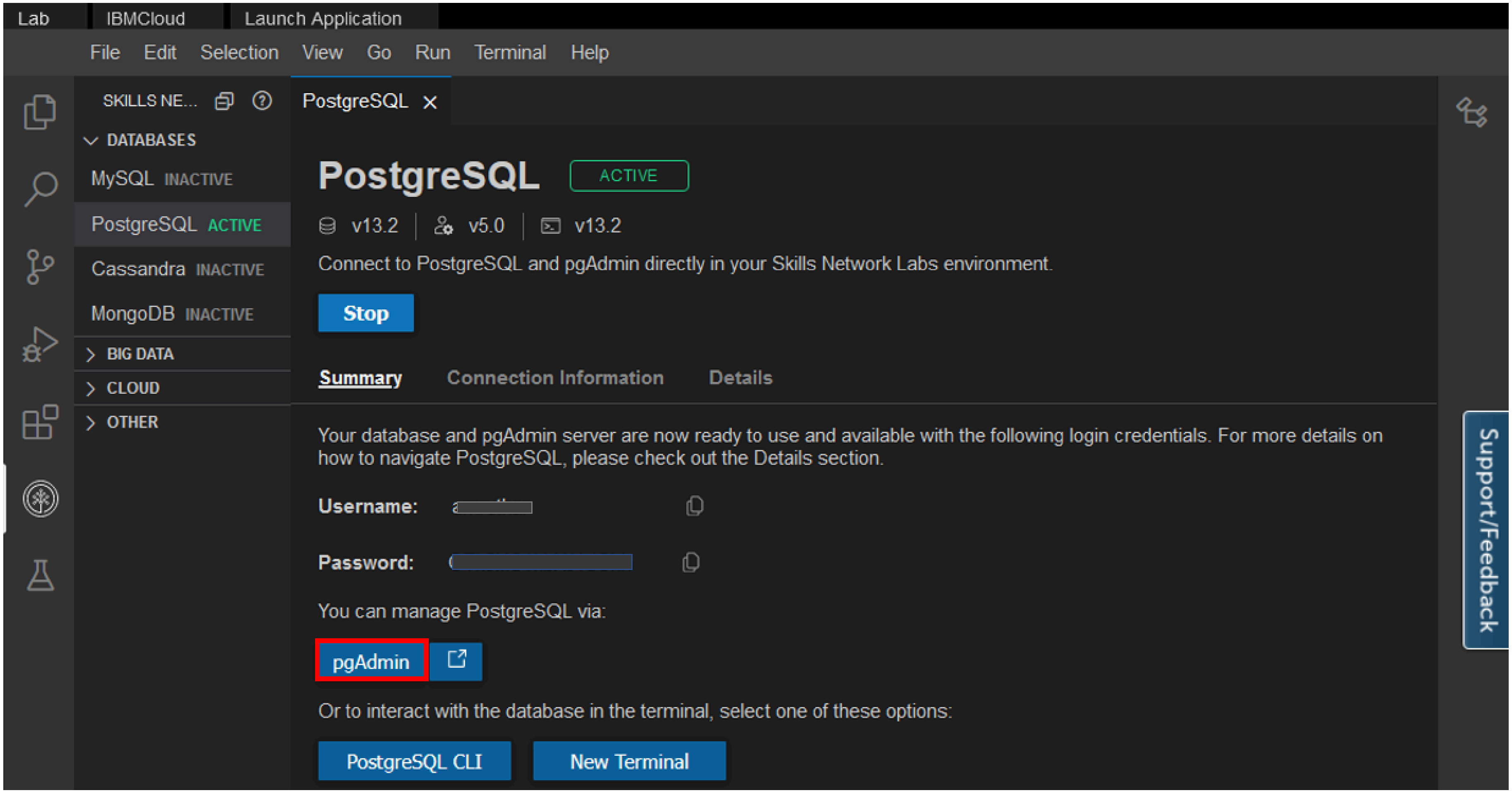
Starting your database. This process can take up to a minute.

Username:

Password:

You can manage PostgreSQL via:

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



PostgreSQL ACTIVE

v13.2 | v5.0 | v13.2

Connect to PostgreSQL and pgAdmin directly in your Skills Network Labs environment.

Stop

Summary Connection Information Details

Your database and pgAdmin server are now ready to use and available with the following login credentials. For more details on how to navigate PostgreSQL, please check out the Details section.

Username: [redacted]

Password: [redacted]

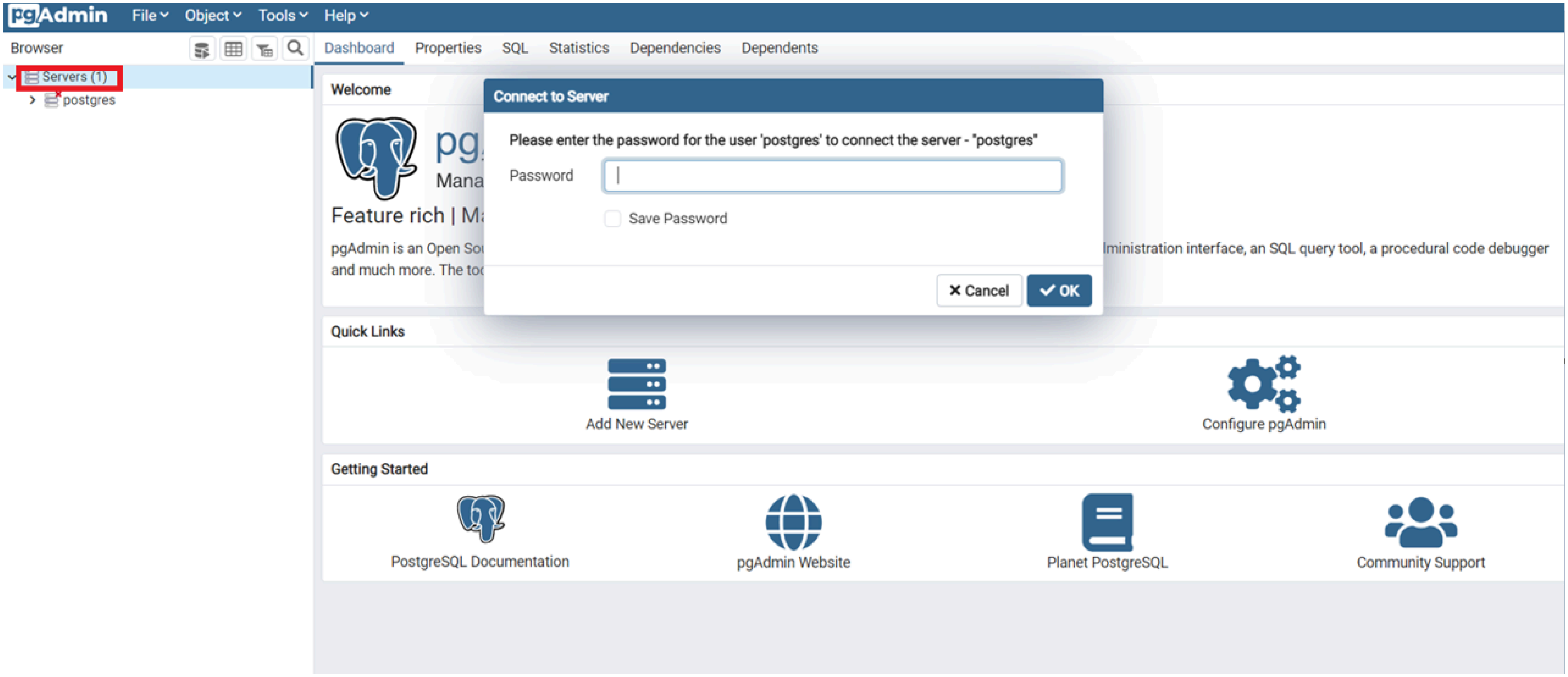
You can manage PostgreSQL via:

pgAdmin

Or to interact with the database in the terminal, select one of these options:

PostgreSQL CLI New Terminal

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



Connect to Server

Please enter the password for the user 'postgres' to connect the server - "postgres"

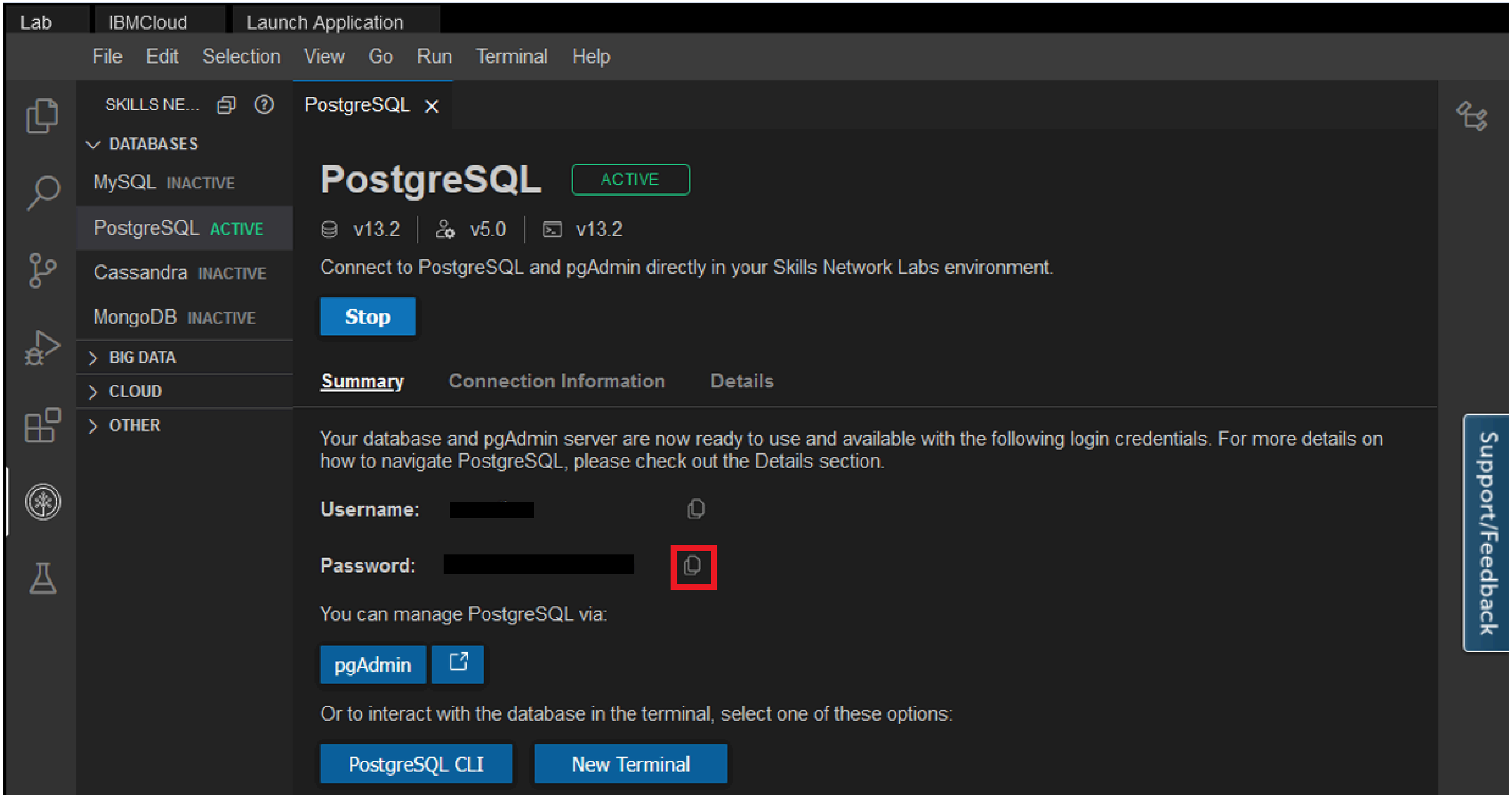
Password

□ Save Password

✕ Cancel ✓ OK

7. To retrieve your password, click on the **PostgreSQL** tab near the top of the interface.

8. Click on the **Copy** icon to the right of your password to copy the session password onto your clipboard.



PostgreSQL

Connect to PostgreSQL and pgAdmin directly in your Skills Network Labs environment.

Your database and pgAdmin server are now ready to use and available with the following login credentials. For more details on how to navigate PostgreSQL, please check out the Details section.

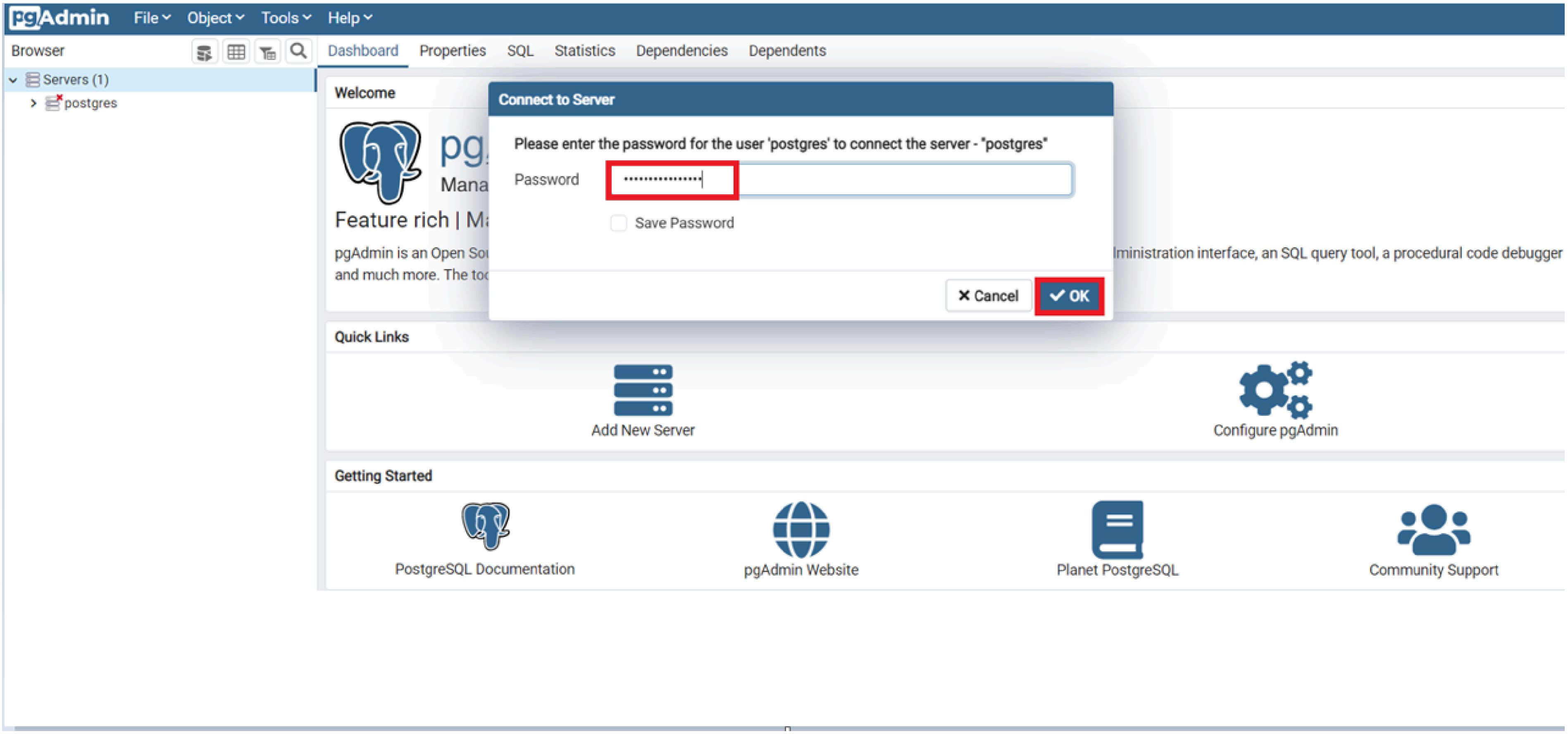
Username: [redacted]

Password: [redacted]

You can manage PostgreSQL via:

Or to interact with the database in the terminal, select one of these options:

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



Connect to Server

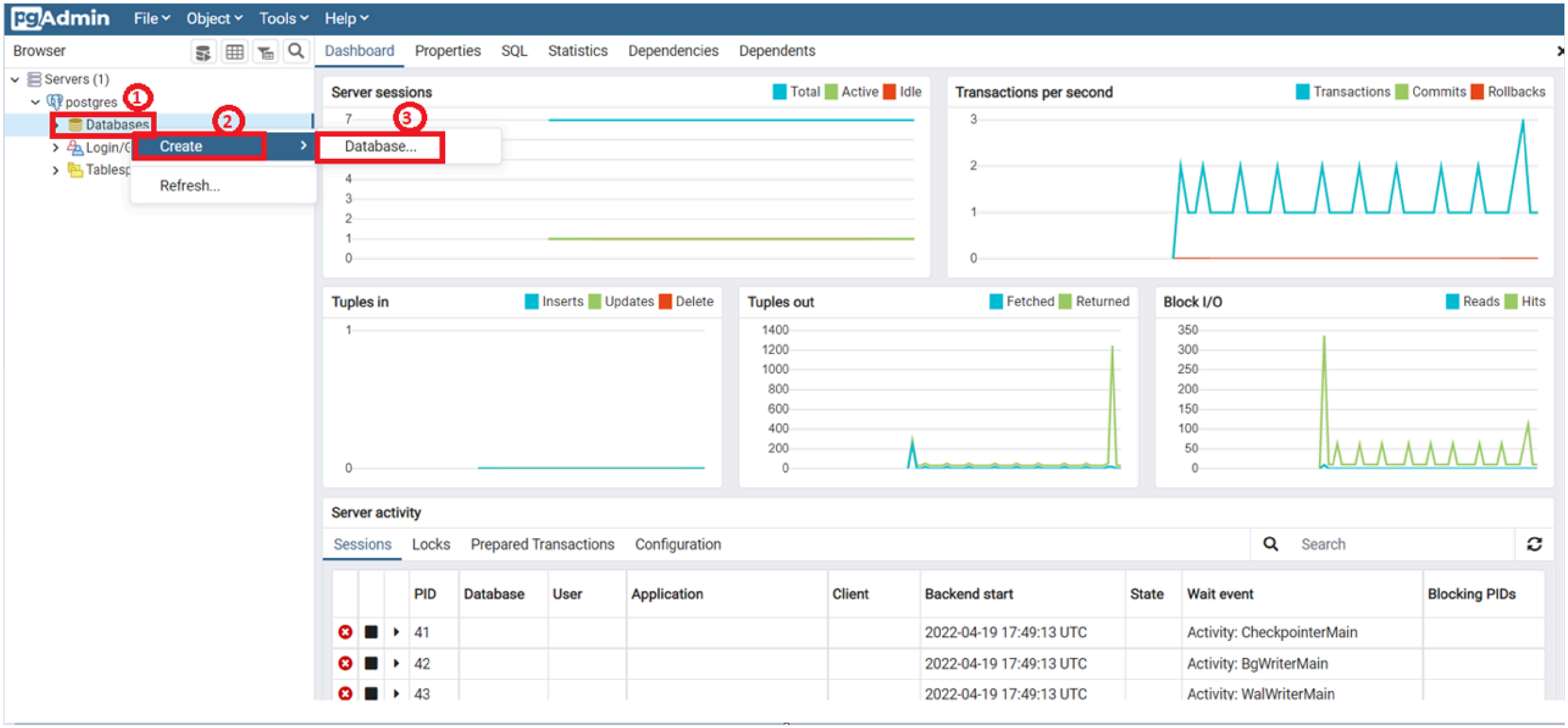
Please enter the password for the user 'postgres' to connect the server - "postgres"

Password ••••••••••••••••

□ Save Password

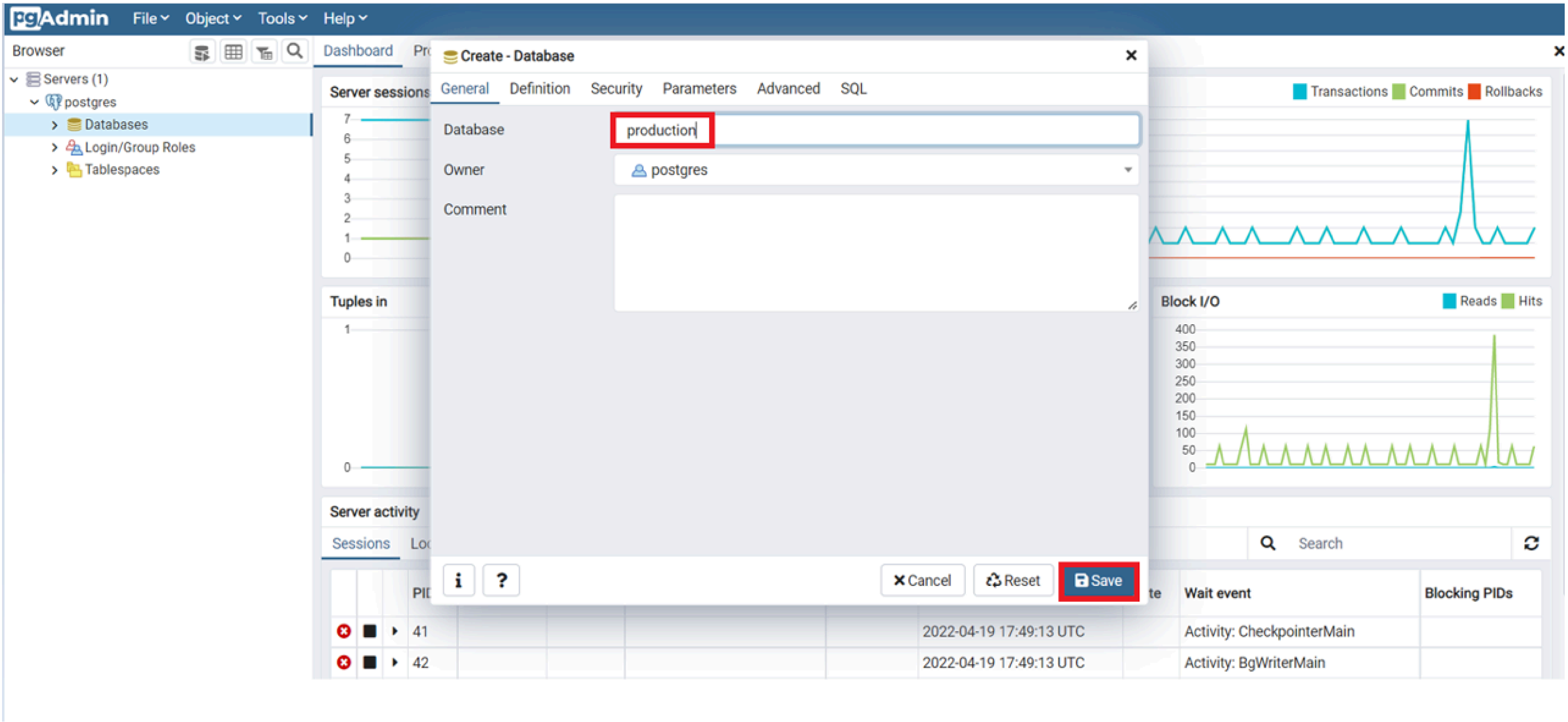
10. You will then be able to access the pgAdmin GUI tool.

11. In the left tree-view, right-click on **Databases> Create > Database**.



			PID	Database	User	Application	Client	Backend start	State	Wait event	Blocking PIDs
⊗	■	▸	41					2022-04-19 17:49:13 UTC		Activity: CheckpointerMain	
⊗	■	▸	42					2022-04-19 17:49:13 UTC		Activity: BgWriterMain	
⊗	■	▸	43					2022-04-19 17:49:13 UTC		Activity: WalWriterMain	

In the Database box, type **Production** as the name for your new database, and then click **Save**. Proceed to Task B.



Create - Database

Database: production

Owner: postgres

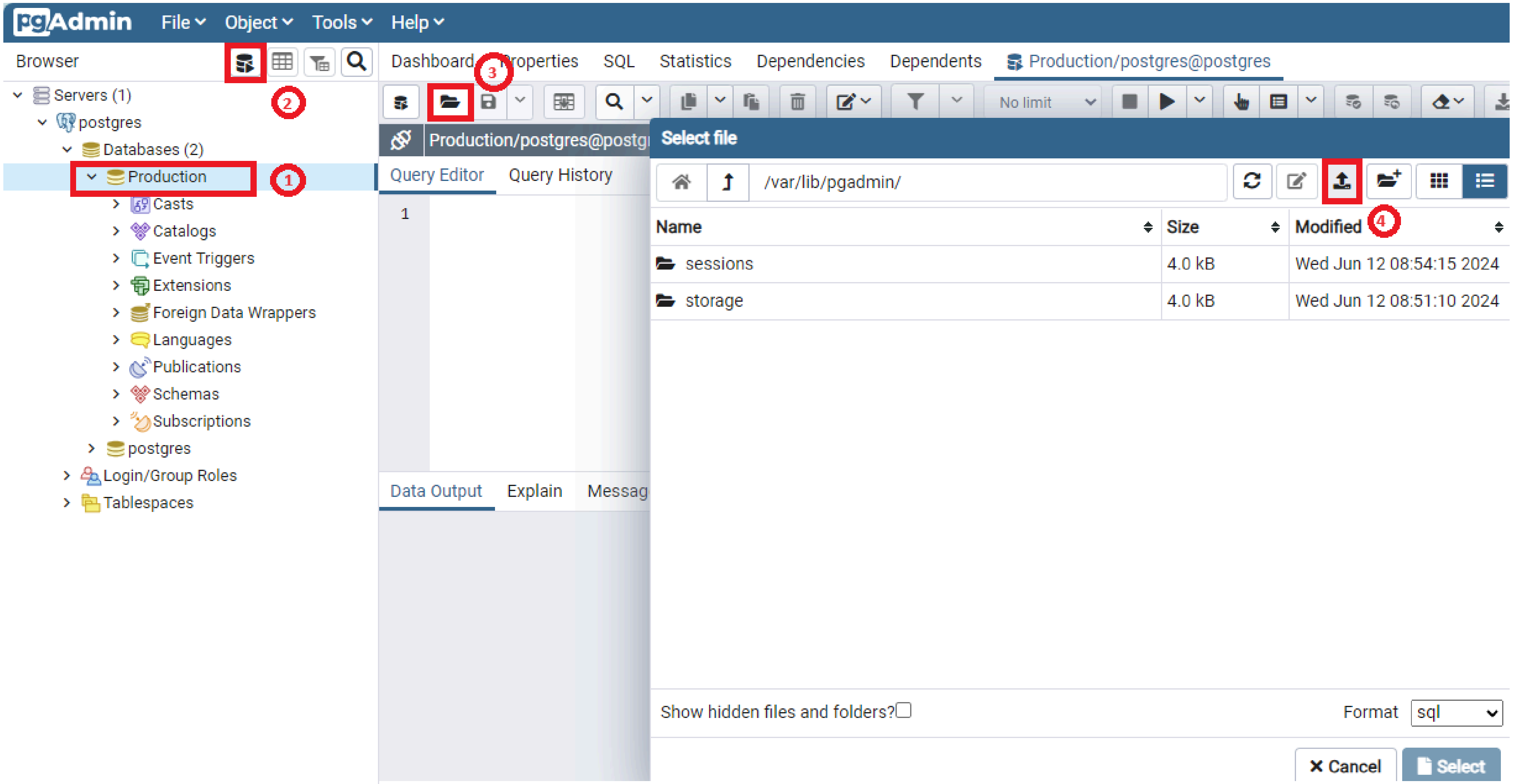
Comment:

Task B: Create tables

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

1. Click on the Production database and in the top of the page go to **Query tool** and then click on **Open File**. Next a new page pops up called **Select File**. Click on **Upload** icon as shown in the screenshot.

   Note: Ensure that you upload the files to this path: /var/lib/pgadmin/



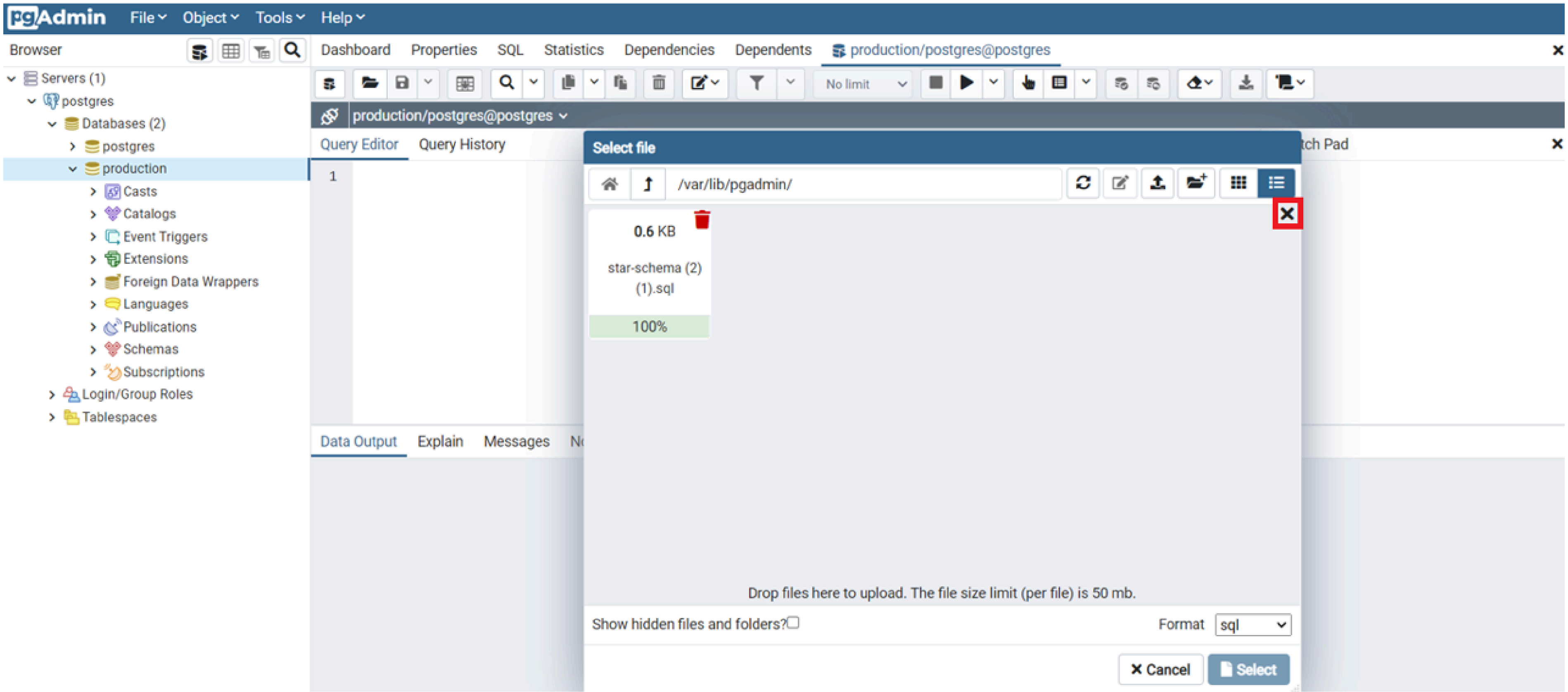
/var/lib/pgadmin/

Name	Size	Modified
sessions	4.0 kB	Wed Jun 12 08:54:15 2024
storage	4.0 kB	Wed Jun 12 08:51:10 2024

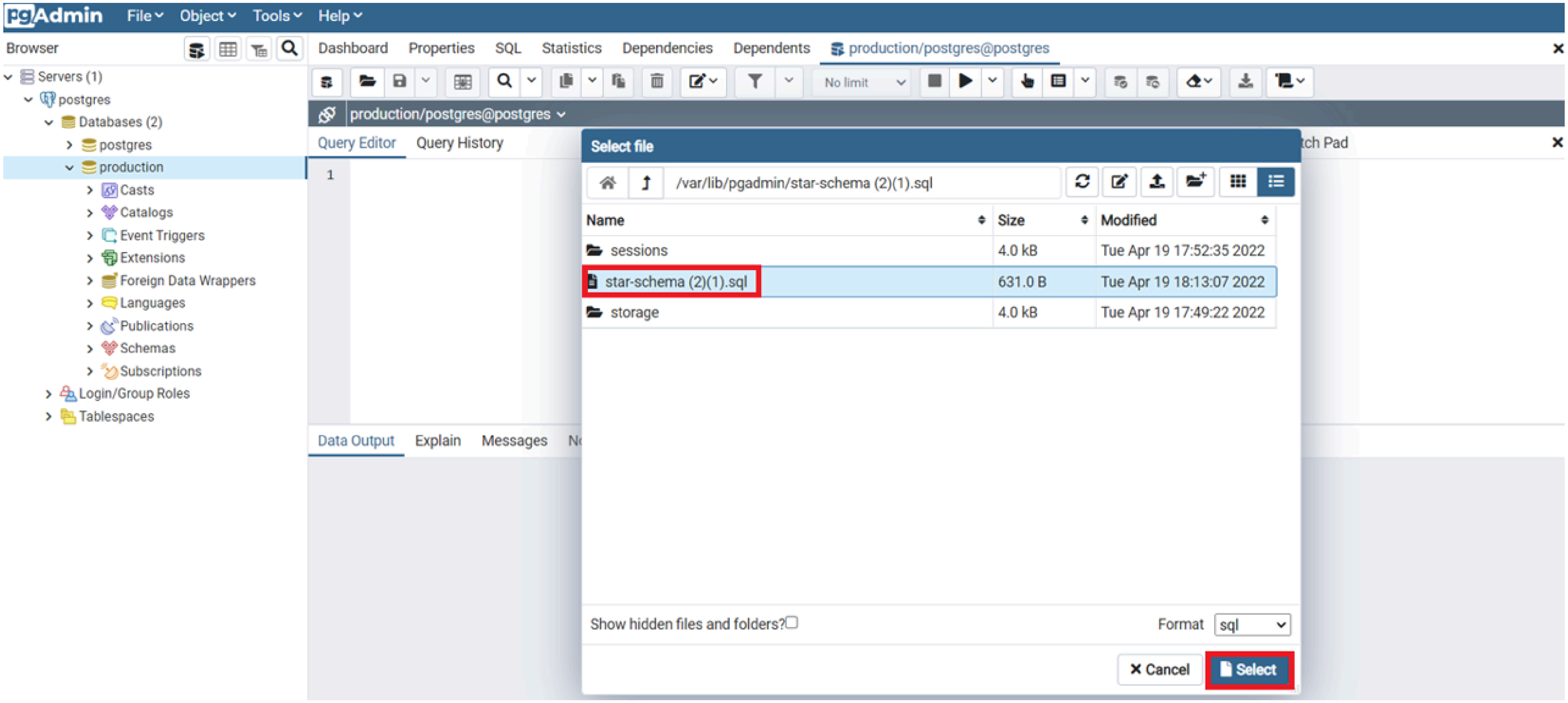
Show hidden files and folders?□

Format sql

2. In the new blank page that appears drag and drop the **star-schema.sql** file inside the blank page. Once the **star-schema.sql** file is successfully loaded, click on the **X** icon on the right hand side of the page as shown in the screenshot.



3. Once you click on the **X** icon a new page appears with the file **star-schema.sql**. Select the **star-schema.sql** file from the list and click the **Select** tab.



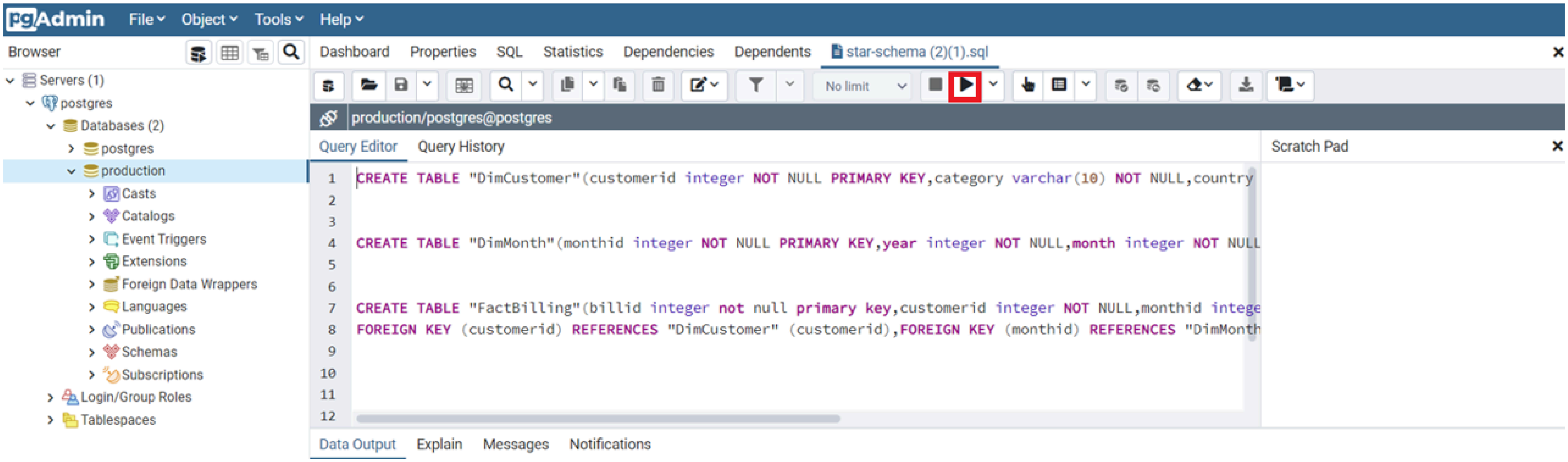
/var/lib/pgadmin/star-schema (2)(1).sql

Name	Size	Modified
sessions	4.0 kB	Tue Apr 19 17:52:35 2022
star-schema (2)(1).sql	631.0 B	Tue Apr 19 18:13:07 2022
storage	4.0 kB	Tue Apr 19 17:49:22 2022

Show hidden files and folders? □

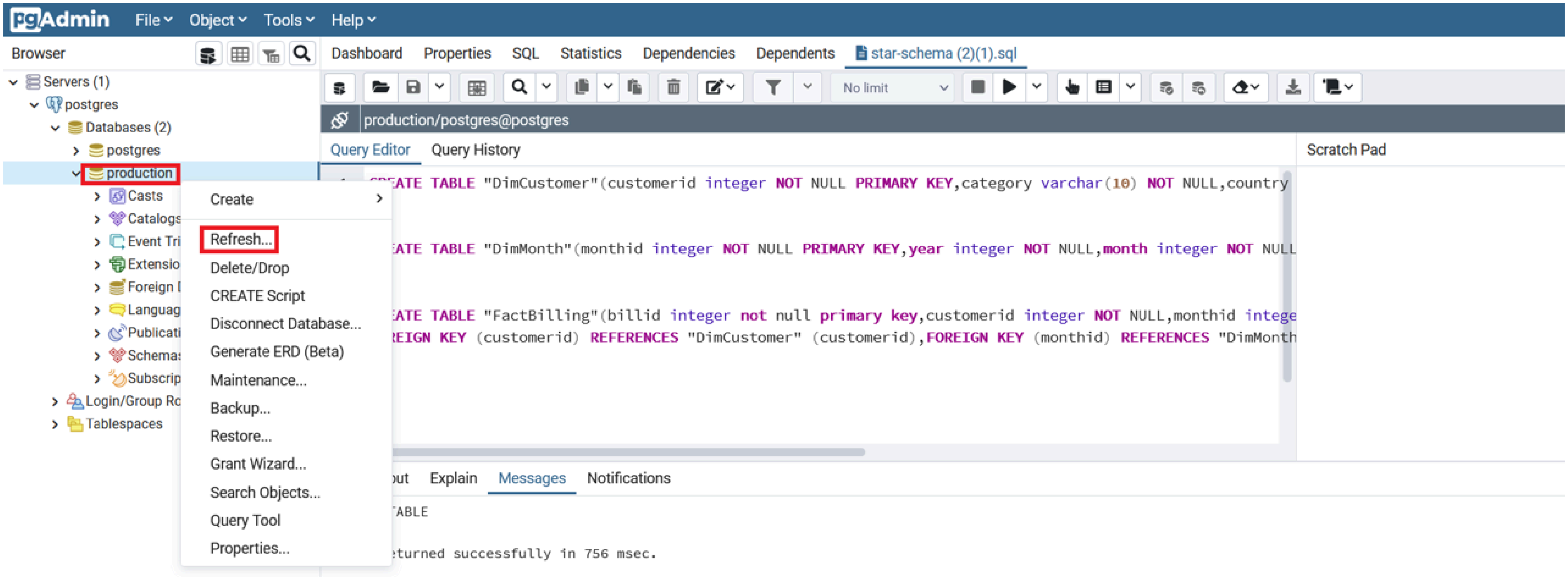
Format: sql

4. Once the file opens up click on the **Run** option to execute the **star-schema.sql** file.



```
CREATE TABLE "DimCustomer"(customerid integer NOT NULL PRIMARY KEY,category varchar(10) NOT NULL,country


CREATE TABLE "DimMonth"(monthid integer NOT NULL PRIMARY KEY,year integer NOT NULL,month integer NOT NULL


CREATE TABLE "FactBilling"(billid integer not null primary key,customerid integer NOT NULL,monthid intege
FOREIGN KEY (customerid) REFERENCES "DimCustomer" (customerid),FOREIGN KEY (monthid) REFERENCES "DimMonth
```

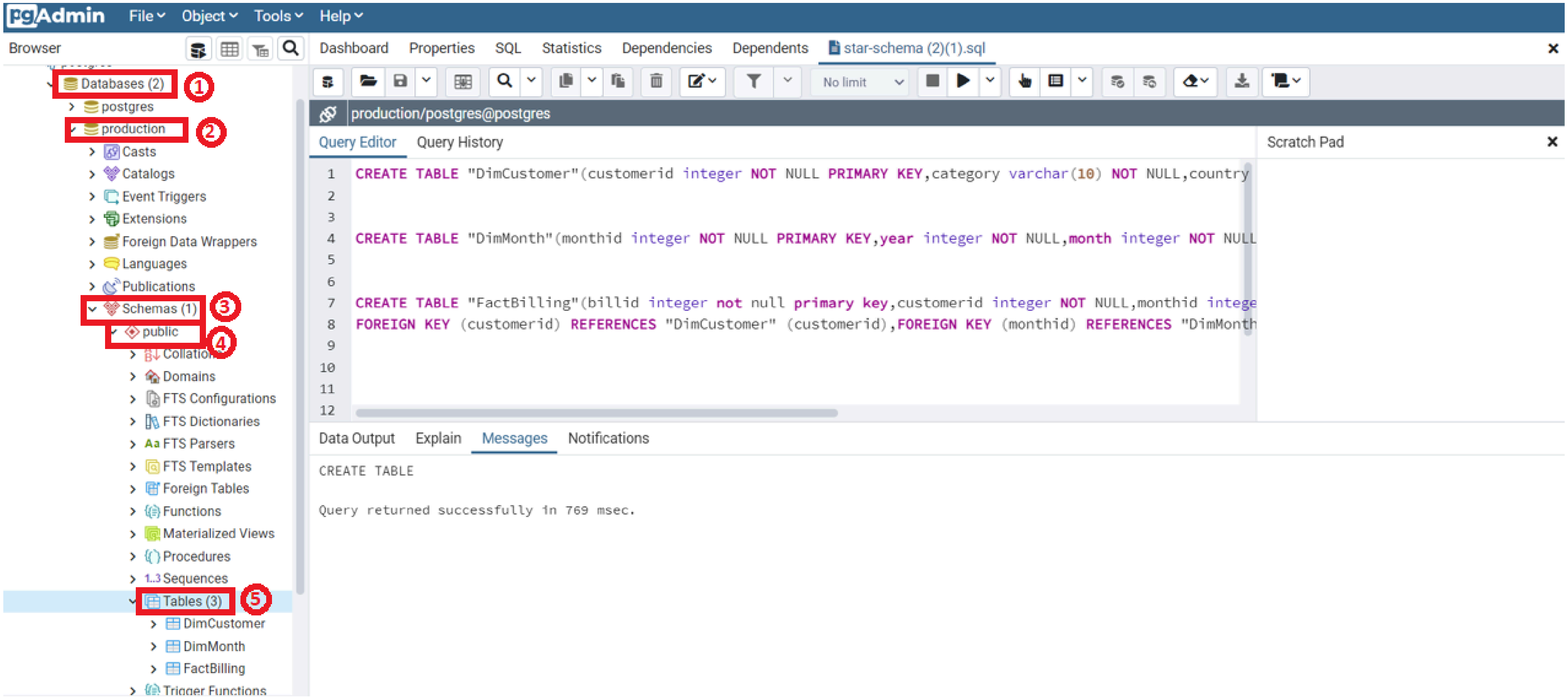
5. Next, right-click on the **Production database** and click on the **Refresh** option from the dropdown.



```
CREATE TABLE "DimCustomer"(customerid integer NOT NULL PRIMARY KEY,category varchar(10) NOT NULL,country

ATE TABLE "DimMonth"(monthid integer NOT NULL PRIMARY KEY,year integer NOT NULL,month integer NOT NULL

ATE TABLE "FactBilling"(billid integer not null primary key,customerid integer NOT NULL,monthid intege
REIGN KEY (customerid) REFERENCES "DimCustomer" (customerid),FOREIGN KEY (monthid) REFERENCES "DimMonth
```

```
ABLE

eturned successfully in 756 msec.
```

After the database is refreshed the 3 tables (DimCustomer, DimMonth,FactBilling) are created under the **Databases > Production > Schemas > Public > Tables**.

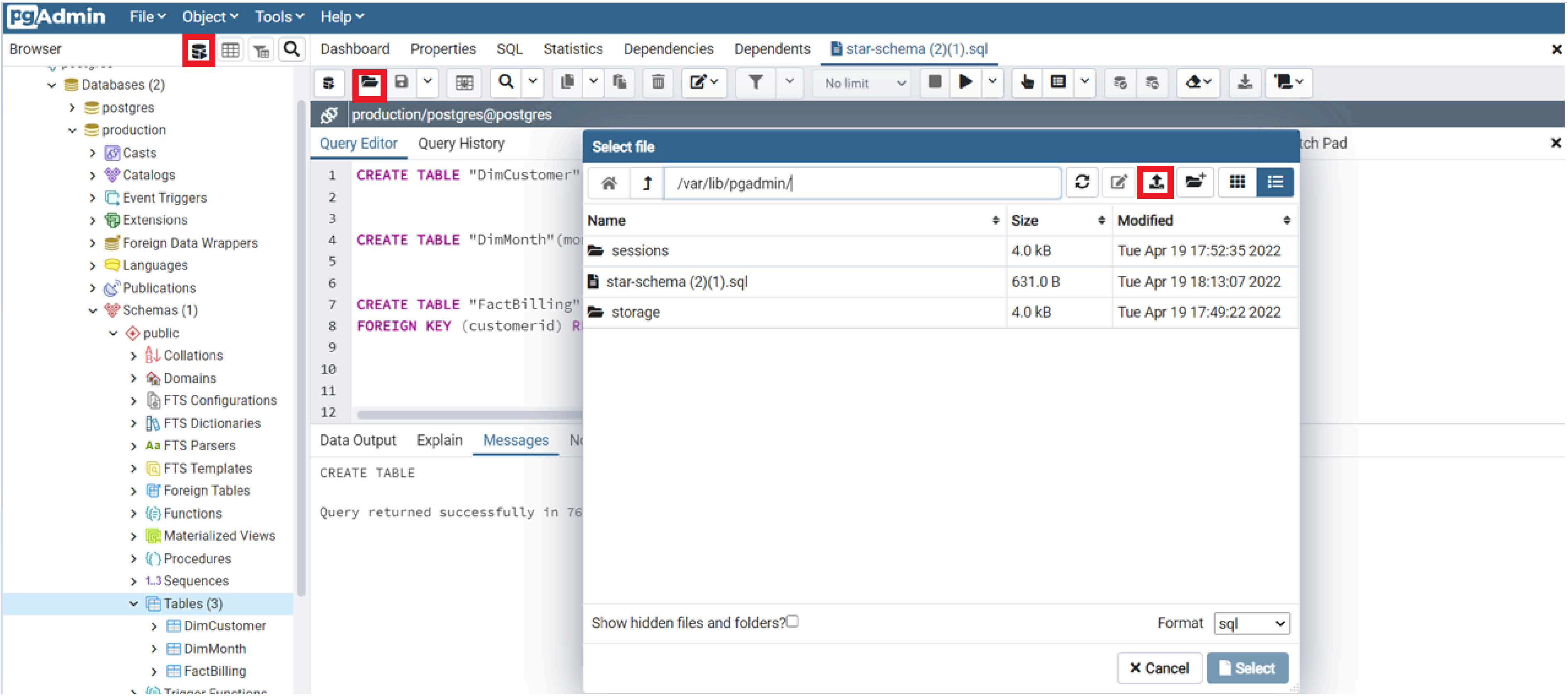


```
CREATE TABLE "DimCustomer"(customerid integer NOT NULL PRIMARY KEY,category varchar(10) NOT NULL,country


CREATE TABLE "DimMonth"(monthid integer NOT NULL PRIMARY KEY,year integer NOT NULL,month integer NOT NULL


CREATE TABLE "FactBilling"(billid integer not null primary key,customerid integer NOT NULL,monthid intege
FOREIGN KEY (customerid) REFERENCES "DimCustomer" (customerid),FOREIGN KEY (monthid) REFERENCES "DimMonth
```

```
CREATE TABLE

Query returned successfully in 769 msec.
```

Task C: Load tables

1. Click on **Query tool** and then click **Open** file and click on **Upload** icon.



```
CREATE TABLE "DimCustomer"


CREATE TABLE "DimMonth"(mo


CREATE TABLE "FactBilling"
FOREIGN KEY (customerid) R
```

```
CREATE TABLE

Query returned successfully in 76
```

/var/lib/pgadmin/

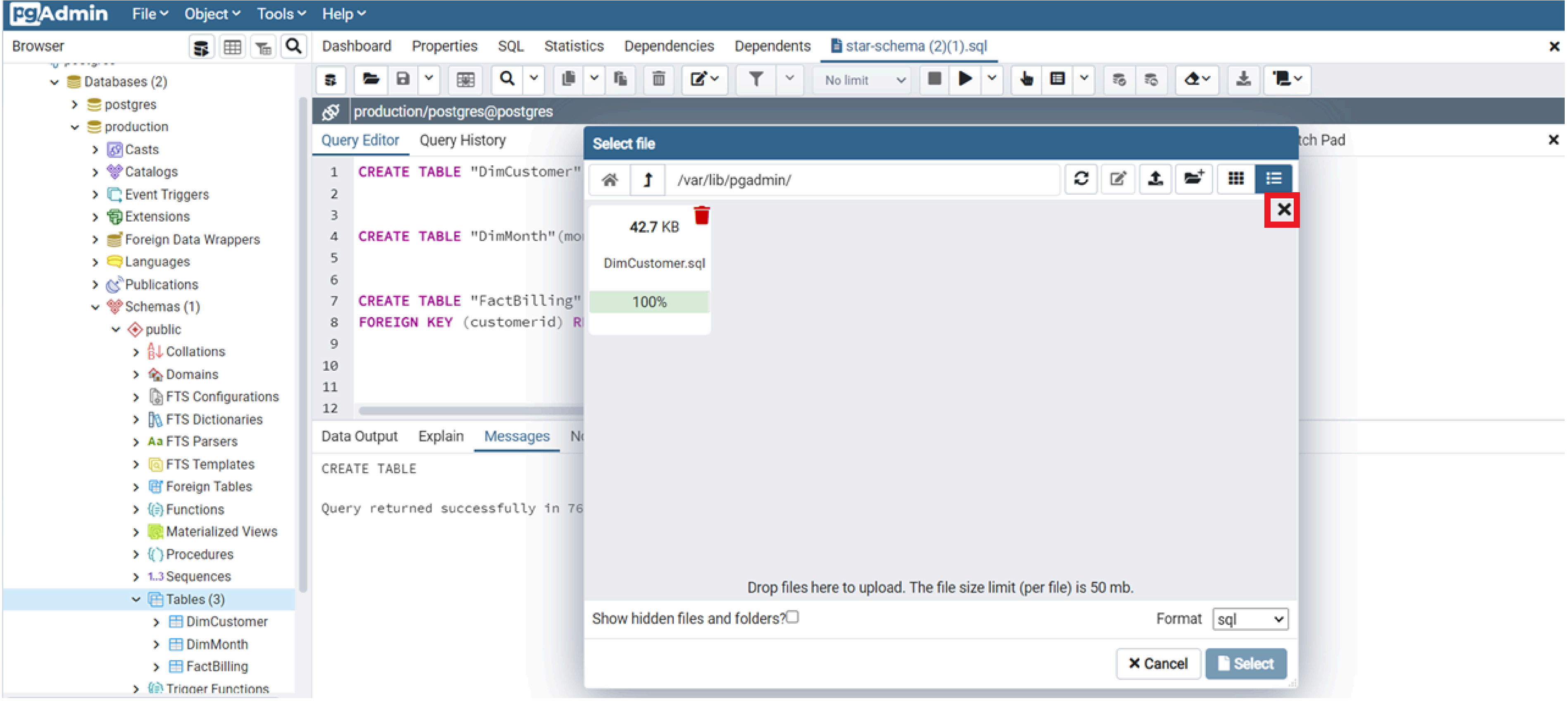
Name	Size	Modified
sessions	4.0 kB	Tue Apr 19 17:52:35 2022
star-schema (2)(1).sql	631.0 B	Tue Apr 19 18:13:07 2022
storage	4.0 kB	Tue Apr 19 17:49:22 2022

Show hidden files and folders? □

Format: sql

2. In the new blank page that appears drag and drop the **DimCustomer.sql** file inside the blank page. Once the **DimCustomer.sql** file is successfully loaded.

Click on the small **X** icon on the right hand side of the page as shown in the screenshot.



```
CREATE TABLE "DimCustomer"


CREATE TABLE "DimMonth"(mo


CREATE TABLE "FactBilling"
FOREIGN KEY (customerid) R
```

```
CREATE TABLE

Query returned successfully in 76
```

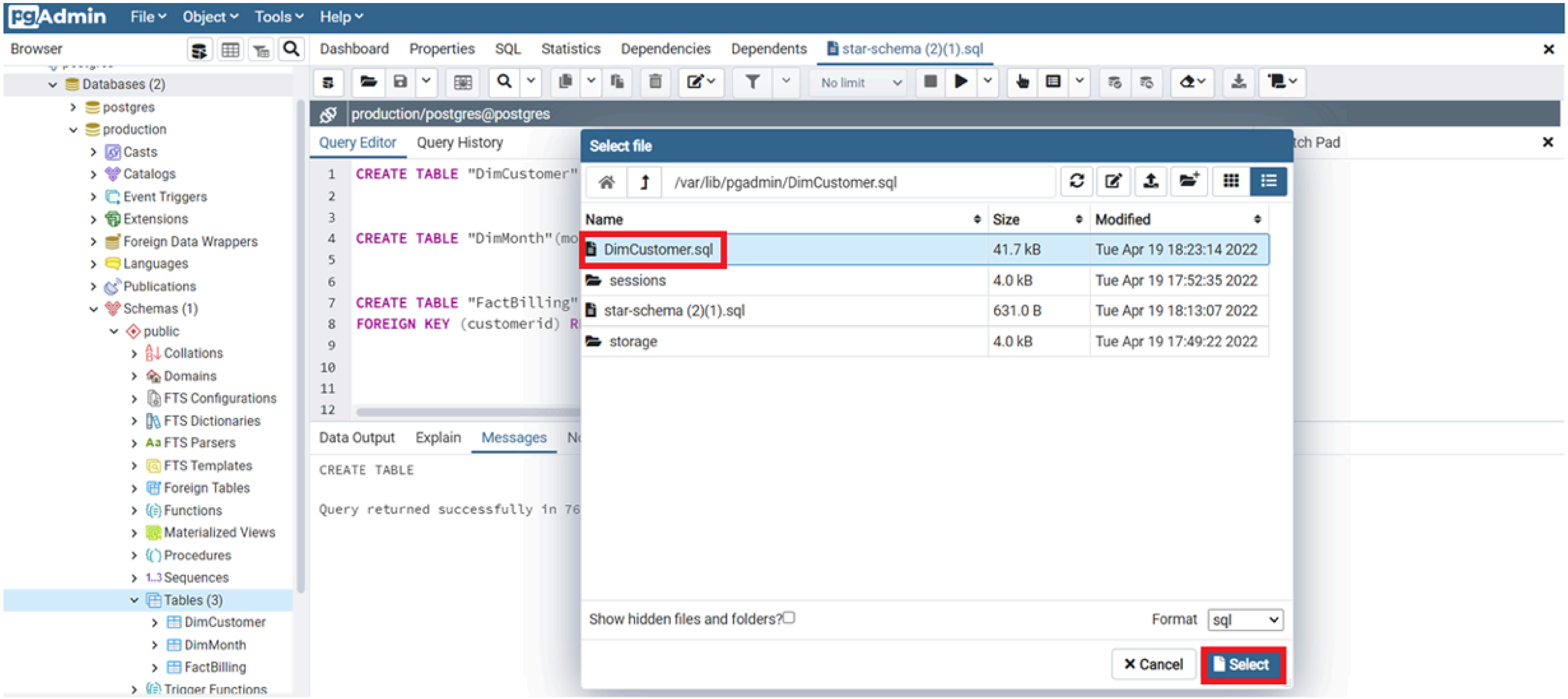
/var/lib/pgadmin/

Drop files here to upload. The file size limit (per file) is 50 mb.

Show hidden files and folders? □

Format sql

3. Once you click on the **X** icon a new page appears with the file **DimCustomer.sql**. Select the **DimCustomer.sql** file from the list and click on **Select** tab.



```
CREATE TABLE "DimCustomer"


CREATE TABLE "DimMonth"(mo


CREATE TABLE "FactBilling"
FOREIGN KEY (customerid) R
```

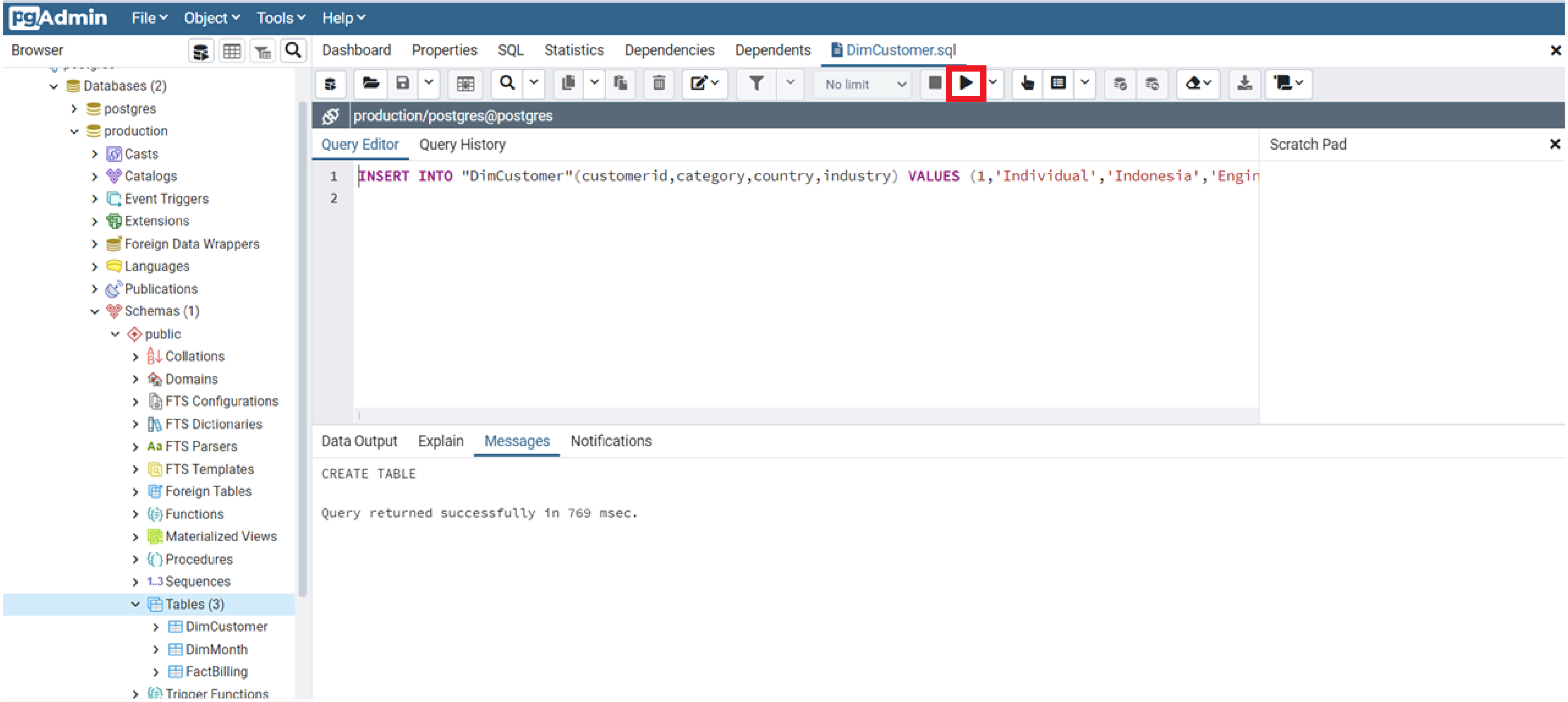
```
CREATE TABLE

Query returned successfully in 76
```

/var/lib/pgadmin/DimCustomer.sql

Name	Size	Modified
DimCustomer.sql	41.7 kB	Tue Apr 19 18:23:14 2022
sessions	4.0 kB	Tue Apr 19 17:52:35 2022
star-schema (2)(1).sql	631.0 B	Tue Apr 19 18:13:07 2022
storage	4.0 kB	Tue Apr 19 17:49:22 2022

Format: sql

4. Once the file opens up, click on the **Run** option to execute the **DimCustomer.sql** file.



```
INSERT INTO "DimCustomer"(customerid,category,country,industry) VALUES (1,'Individual','Indonesia','Engin
```

```
CREATE TABLE

Query returned successfully in 769 msec.
```

Note: Repeat the steps as given in Task C to upload the remaining sql files to insert data in **DimMonth and FactBilling**.

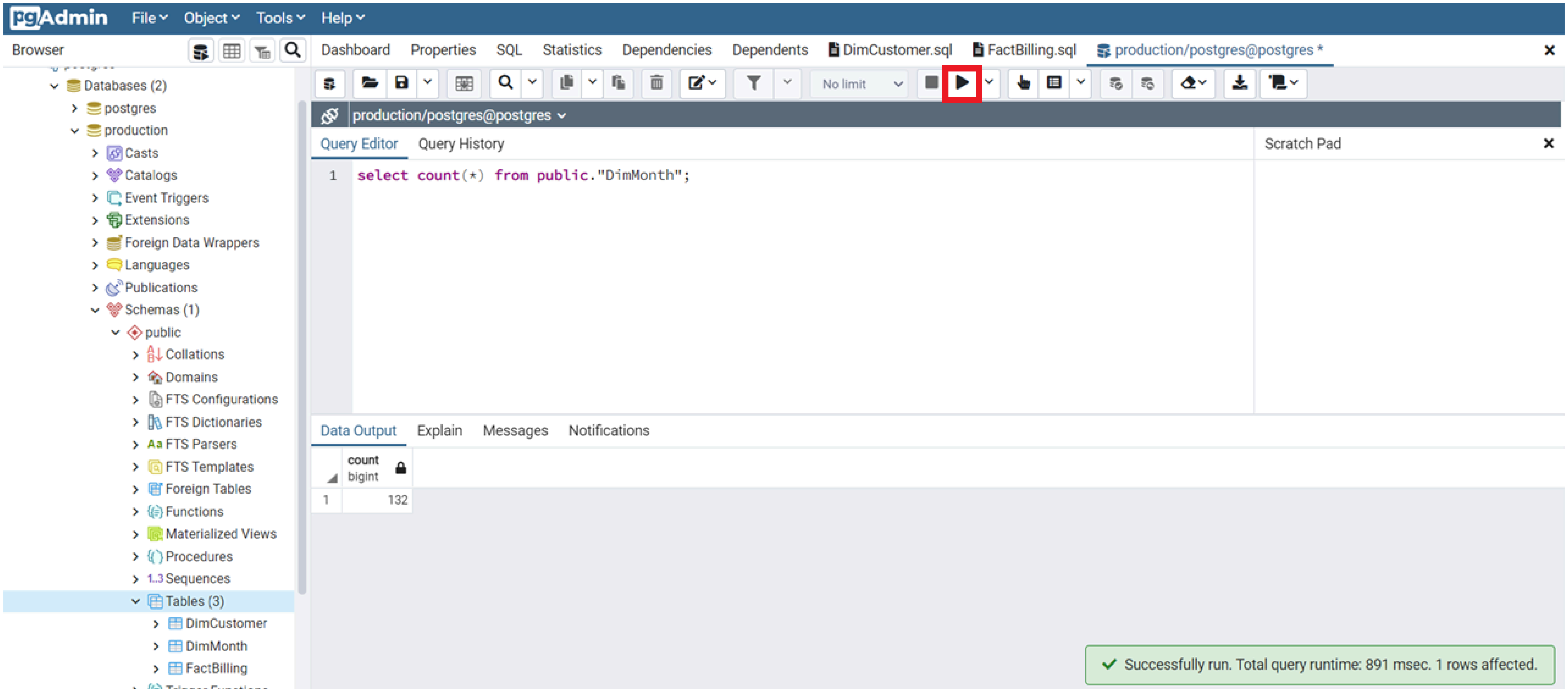
5. Let's run the command below on the PostgreSQL Tool.

```
1. 1
```

```
1. select count(*) from public."DimMonth";
```

Copied!

You should see an output as seen in the image below.



```
select count(*) from public."DimMonth";
```

	count bigint
1	132

✓ Successfully run. Total query runtime: 891 msec. 1 rows affected.

You are encouraged to run more SQL queries.

Practice exercises

Problem 1: Using the PostgreSQL tool, find the count of rows in the table FactBilling

▶ Click here for Hint
▼ Click here for Solution

```
select count(*) from public."FactBilling";
```

Copied!

Problem 2: Using the PostgreSQL tool, create a simple Materialized views named avg_customer_bill with fields customerid and averagebillamount.

▶ Click here for Hint
▼ Click here for Solution

```
4
5
```

```
CREATE MATERIALIZED VIEW  avg_customer_bill (customerid, averagebillamount) AS
(select customerid, avg(billedamount)
from public."FactBilling"
group by customerid
);
```

Copied!

Click the **Run All** Button to run the statement. You should see status as **Success** in the **Result** section.

Problem 3: Refresh the newly created Materialized views

▶ Click here for Hint

▼ Click here for Solution

```
1
```

```
REFRESH MATERIALIZED VIEW avg_customer_bill;
```

Copied!

Problem 4: Using the newly created Materialized views find the customers whose average billing is more than 11000.

▶ Click here for Hint

▼ Click here for Solution

```
1
```

```
select * from avg_customer_bill where averagebillamount > 11000;
```

Copied!

Congratulations! You have successfully finished the Populating a Data Warehouse lab.

Author

Amrutha Rao