

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](#)

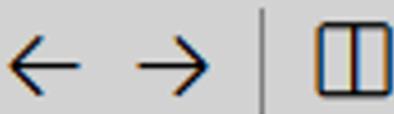
- [Star Schema](#)

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 **Create** button. PostgreSQL may take a few moments to start.

File Edit Selection View Go Run T



SKILLS NE...



PostgreSQL



▼ DATABASES

2

MySQL INACTIVE

PostgreSQL INACTIVE

3

Cassandra INACTIVE

MongoDB INACTIVE

> BIG DATA

> CLOUD

> EMBEDDABLE AI

> OTHER

Launch Applica...



1

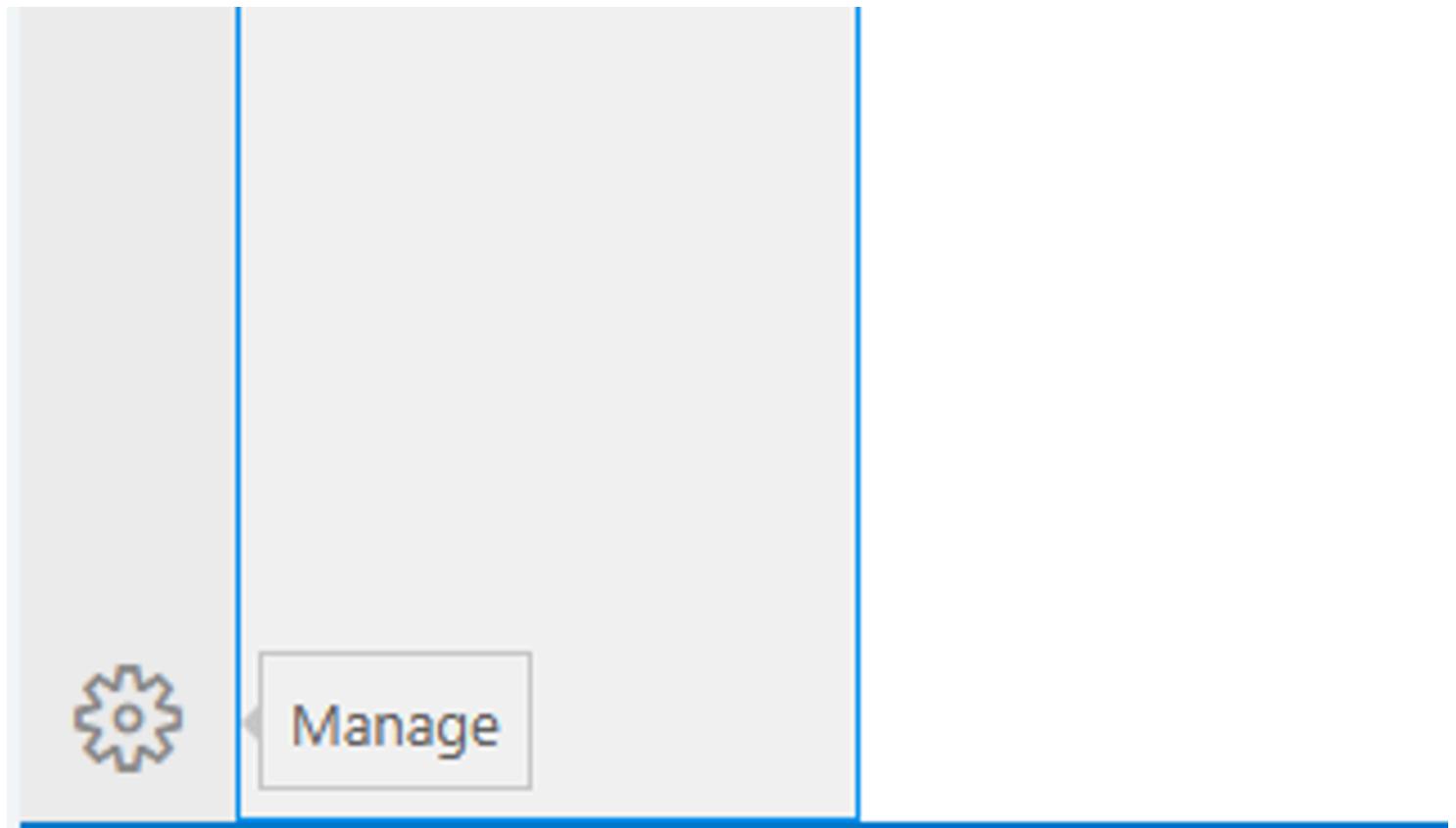
PostgreSe

v13.2 | v5

Connect to Postgres environment.

[Create](#)[Summary](#)[Con](#)

Get started with Pos
the Start button.



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

File Edit Selection View Go Run



PostgreSQL X



PostgreSQL

ACTIVE

v13.2 | v5.0 | v14.5

Connect to PostgreSQL and pgAdmin dire

Create

De

Summary

Connection Information

Your database and pgAdmin server are no
For more details on how to navigate Postg

You can manage PostgreSQL via:

pgAdmin



Or to interact with the database in the terminal.

PostgreSQL CLI

New Terminal



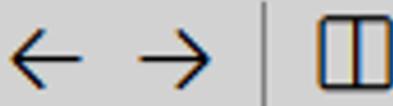
Manage

- 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.

The screenshot shows the pgAdmin 4 graphical user interface. At the top, there is a navigation bar with tabs for File, Object, Tools, and Help. Below the navigation bar, a sidebar on the left has a 'Servers' section with one entry ('Servers (1)') highlighted with a red box. The main content area features a 'Welcome' section with the pgAdmin logo and a 'Feature rich | Manage' message. A 'Connect to Server' dialog box is overlaid on the screen, prompting the user to enter a password for the 'postgres' user to connect to the 'postgres' server. The dialog includes fields for 'Password' and 'Save Password', and buttons for 'Cancel' and 'OK'. Below the dialog, there are 'Quick Links' for 'Add New Server' and 'Configure pgAdmin'. At the bottom, there is a 'Getting Started' section with links to 'PostgreSQL Documentation', 'pgAdmin Website', and 'Planet PostgreSQL'.

7. To retrieve your password, click on the **PostgreSQL** and go to **Conection Information** tab on the top of the interface.

File Edit Selection View Go Run



PostgreSQL X



PostgreSQL

ACTIVE

v13.2 | v5.0 | v14.5

Connect to PostgreSQL and pgAdmin directly

Create

De

Summary

Connection Information



POSTGRES_USERNAME:

postgres

POSTGRES_HOST:

172.21.68.118

5432

POSTGRES_PORT:

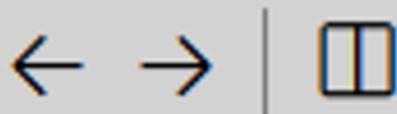
URL:

No Problems

<https://labs-postgres-sharp-m.eventide-postgres.databases.us-east-1.amazonaws.com>

8. Scroll down to the session password and click on the **Copy** icon to the right of your password to copy onto your clipboard.

File Edit Selection View Go Run



PostgreSQL X

eventide.postgres.databases.la



PostgreSQL CLI Command:

```
export PGPASSWORD=U3hEw8LpGIXE  
-p 5432 -U postgres
```



POSTGRES_COMMAND:

```
export PGPASSWORD=U3hEw8LpGIXE  
-p 5432 -U postgres
```



POSTGRES_PASSWORD:

POSTGRES_TITLE:

Postgres Datab

POSTGRES_ID:

labs-postgres-sha



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

The screenshot shows the pgAdmin interface. The top navigation bar includes File, Object, Tools, and Help. Below the menu is a toolbar with icons for Browser, Servers, Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. On the left, a tree view shows 'Servers (1)' with 'postgres' selected. The main area has a 'Welcome' section featuring the pgAdmin logo and text about being a feature-rich management tool. A 'Connect to Server' dialog box is open over the main content. It prompts for the password for the 'postgres' user on the 'postgres' server. The 'Password' field contains several dots and is surrounded by a red box. To the right of the field is a checkbox for 'Save Password'. At the bottom of the dialog are 'Cancel' and 'OK' buttons, with 'OK' also being highlighted by a red box. Below the dialog, there are 'Quick Links' for adding a new server and 'Getting Started' links to PostgreSQL Documentation, pgAdmin Website, and Planet PostgreSQL.

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

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

The screenshot shows the pgAdmin interface with the following steps highlighted:

- Click on the 'Databases' icon under the 'Servers' section.
- Click on the 'Create' button.
- Click on the 'Database...' button.

In the 'Create - Database' dialog box, the 'Database' field is filled with 'production'. The 'Owner' dropdown is set to 'postgres'. The 'Save' button at the bottom right is highlighted with a red box.

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

The screenshot shows the pgAdmin interface with the 'Create - Database' dialog box open. The 'Database' field contains 'production'. The 'Owner' dropdown is set to 'postgres'. The 'Save' button at the bottom right is highlighted with a red box.

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.

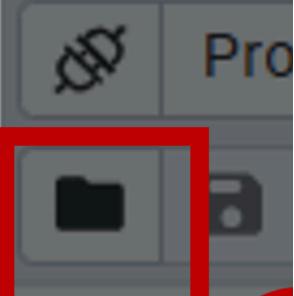
- Click on the Production database and in the top of the page go to **Query tool** and then click on **Open File**. Next a new page pops up called **Select File**. Click on the three dots and choose **Upload** option as shown in the screenshot.

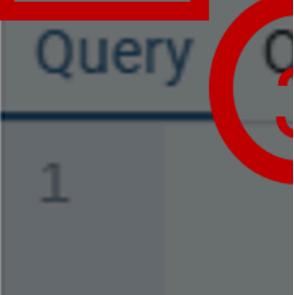
Note: Ensure that you upload the files to this path: /var/lib/pgadmin/

pgAdmin File ▾ Object ▾ Tools ▾

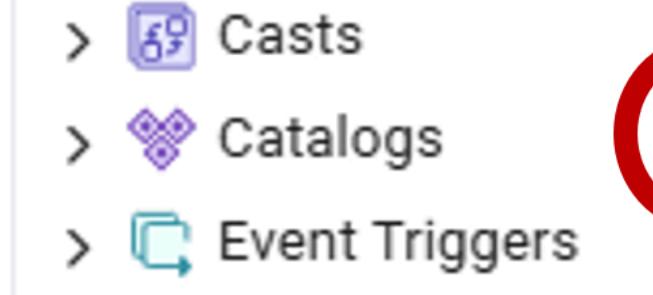
Object Explorer Dashboard

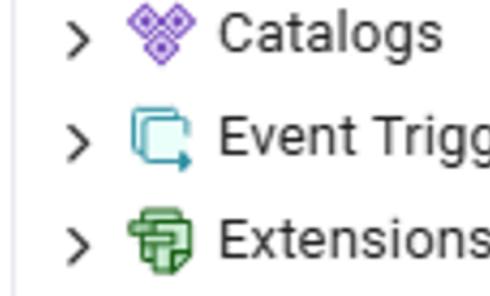
Servers (1)      

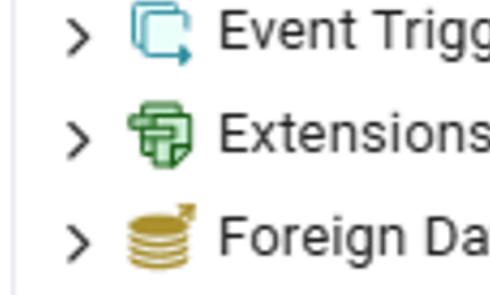
PostgreSQL 

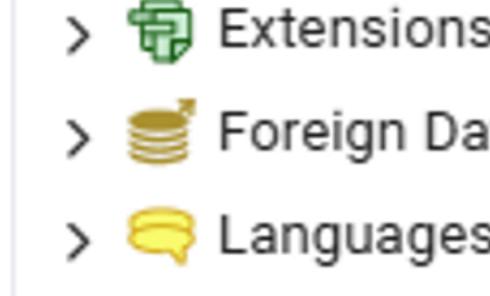
Databases (2) 

Production 

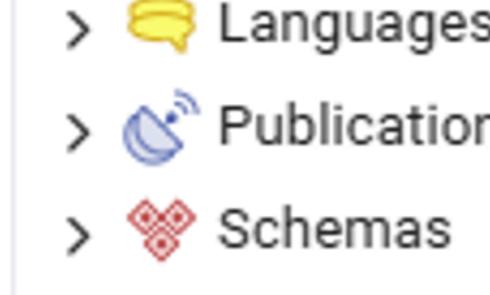
Casts 

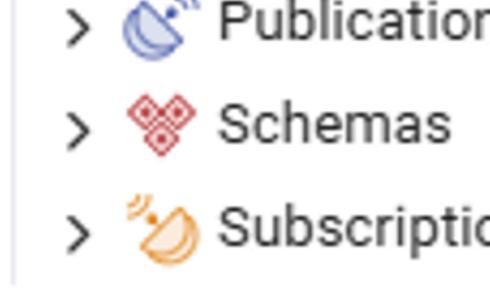
Catalogs 

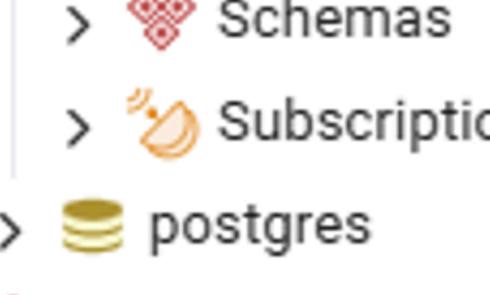
Event Triggers 

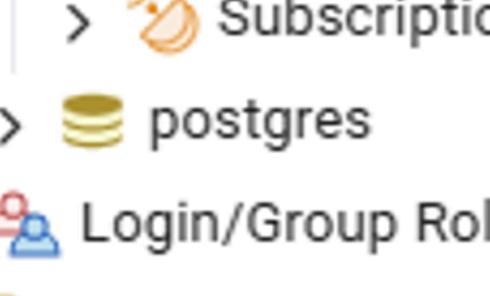
Extensions 

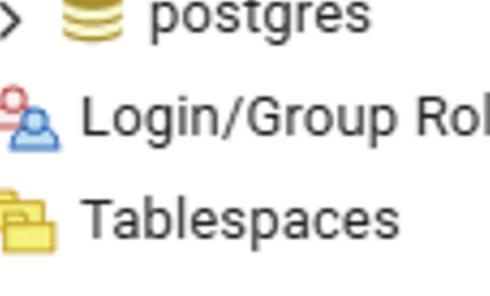
Foreign Data Wrappers 

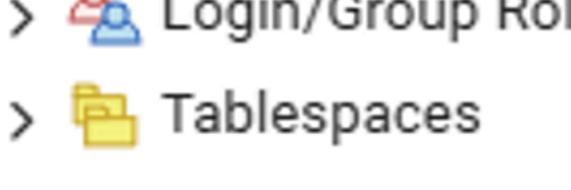
Languages 

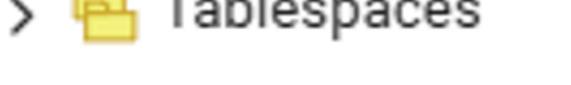
Publications 

Schemas 

Subscriptions 

postgres 

Login/Group Roles 

Tablespaces 



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.

pgAdmin File ▾ Object ▾ Tools ▾

Object Explorer Dashboard

Servers (1) PostgreSQL Databases (2)

Production Casts
 Catalogs
 Event Triggers
 Extensions
 Foreign Data Wrappers
 Languages
 Publications
 Schemas
 Subscriptions
 postgres
 Login/Group Roles
 Tablespaces

Query 1

Data Out



- 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.

pgAdmin File ▾ Object ▾ Tools ▾

Object Explorer

Servers (1)

PostgreSQL

Databases (2)

Production

- > Casts
- > Catalogs
- > Event Triggers
- > Extensions
- > Foreign Data Wrappers
- > Languages
- > Publications
- > Schemas
- > Subscriptions

> postgres

> Login/Group Roles

> Tablespaces

Dashboard

Query

Data Output



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 integer
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.

The screenshot shows the pgAdmin interface. In the left sidebar, under 'Servers (1)', 'postres' is selected, and 'Databases (2)' is expanded to show 'postgres' and 'production'. The 'production' database is selected. A context menu is open over the 'production' database, with the 'Refresh...' option highlighted. The main pane displays the SQL code for creating three 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 integer
FOREIGN KEY (customerid) REFERENCES "DimCustomer" (customerid),FOREIGN KEY (monthid) REFERENCES "DimMonth"

```

The status bar at the bottom right indicates the query was executed successfully in 756 msec.

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

The screenshot shows the pgAdmin interface after refreshing the database. The left sidebar now shows the structure of the 'production' database. Numbered callouts point to specific items:

- ① 'Databases (2)' node
- ② 'production' database node
- ③ 'Schemas (1)' node
- ④ 'public' schema node
- ⑤ 'Tables (3)' node

The main pane shows the same SQL code for creating the three tables. The status bar at the bottom right indicates the query was executed successfully in 769 msec.

Task C: Load tables

1. Click on **Query tool** and then click **Open** file and click on the three dot and choose **Upload** option.

pgAdmin

File ▾ Object ▾ Tools ▾

Object Explorer

Servers (1)

PostgreSQL

Databases (2)

Production

Casts

Catalogs

Event Triggers

Extensions

Foreign Data Wrappers

Languages

Publications

Schemas

Subscriptions

postgres

Login/Group Roles

Tablespaces

Dashboard

Query

Data Out

The screenshot shows the pgAdmin interface with the 'Object Explorer' tab selected. A red box highlights the 'Servers' icon in the toolbar. A large red circle with the number '1' is overlaid on the 'Databases' node in the tree view. The 'Production' database is currently expanded, displaying its child objects: Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, Subscriptions, and the postgres database itself. Other nodes like Login/Group Roles and Tablespaces are also visible. The right side of the screen shows the 'Query' and 'Data Out' panes.



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.

pgAdmin File ▾ Object ▾ Tools ▾

Object Explorer

Properties Query Data Output

> Collations
> Domains
> FTS Configurations
> FTS Dictionaries
> FTS Parsers
> FTS Templates
> Foreign Tables
> Functions
> Materialized Views
> Operators
> Procedures
> Sequences
▼ Tables (3)
> DimCustomer
> DimMonth
> FactBilling

- >  Trigger Functions
- >  Types
- >  Views
- >  Subscriptions
- >  postgres
- >  Login/Group Roles
- >  Tablespaces

Total row

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.

pgAdmin

File ▾ Object ▾ Tools ▾

Object Explorer

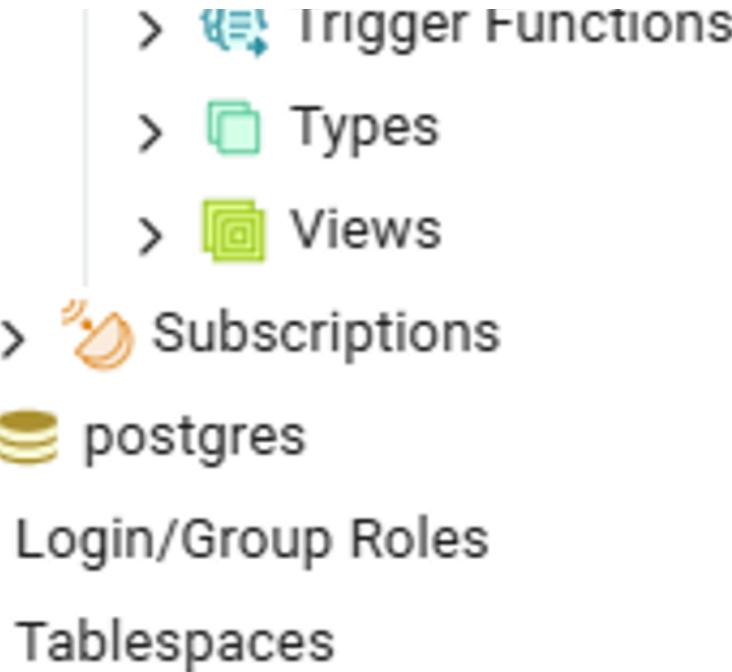
X Prop

- > A↓ Collations
- > Domains
- > FTS Configurations
- > FTS Dictionaries
- > Aa FTS Parsers
- > FTS Templates
- > Foreign Tables
- > Functions
- > Materialized Views
- > Operators
- > Procedures
- > 1..3 Sequences
- Tables (3)
 - > DimCustomer
 - > DimMonth
 - > FactBilling

Query

1

Data Out



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

The screenshot shows the pgAdmin interface with the following details:

- File Bar:** File, Object, Tools, Help
- Toolbar:** Includes icons for Database, Object, Properties, SQL, Statistics, Dependencies, Dependents, and a dropdown for the current connection (DimCustomer.sql).
- Browser:** Shows the database structure under 'production/postgres@postgres' with expanded sections for Databases, Schemas, and Tables.
- Query Editor:** Contains the following SQL code:


```
1 INSERT INTO "DimCustomer"(customerid,category,country,industry) VALUES (1,'Individual','Indonesia','Engineering')
2
```
- Messages Tab:** Shows the output of the query execution:


```
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**.

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

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

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

The screenshot shows the pgAdmin 4 interface. On the left, the Browser pane displays the database structure under 'production'. The central area contains a 'Query Editor' tab with the following SQL query:

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

The results are shown in the 'Data Output' tab:

	count	bigint
1	132	

A green success message at the bottom right states: 'Successfully run. Total query runtime'.

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";
```

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

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

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

```
REFRESH MATERIALIZED VIEW avg_customer_bill;
```

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

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

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

Author

Amrutha Rao

© IBM Corporation 2022. All rights reserved.