Hands-on Lab: Getting Started with the PostgreSQL Command Line

Estimated time needed: 20 minutes

In this lab, you will use the PostgreSQL command line interface (CLI) to create a database and restore the structure and contents of its tables. Then, you will learn how to explore and query tables. Finally, you will learn how to dump/backup tables from a database.

Software used in this lab

In this lab, you will use a PostgreSQL Database. PostgreSQL is a relational database management system (RDBMS) designed to store, manipulate, and retrieve data efficiently.

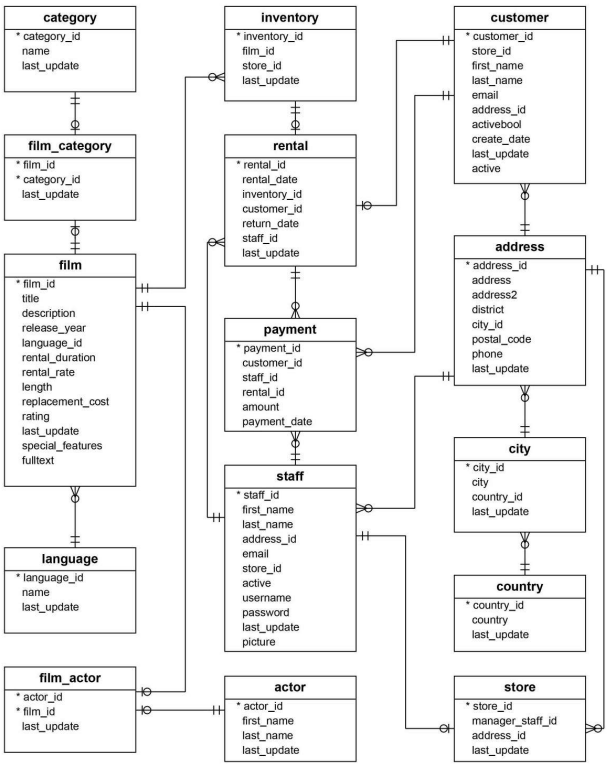
To complete this lab, you will utilize the PostgreSQL relational database service available as part of IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database used in this lab

The Sakila database used in this lab comes from the following source: https://dev.mysql.com/doc/sakila/en/ under New BSD license [Copyright 2021 - Oracle Corporation].

You will use a modified version of the database for the lab. To follow the lab instructions successfully, please use the database provided by the lab rather than the database from the source.

The following entity relation diagram (ERD) shows the structure of the schema of the Sakila database:



Objectives

After completing this lab, you will be able to use the PostgreSQL command line to:

- Create a database
- Restore the structure and data of a table
- Explore and query tables
- Dump/backup tables from a database

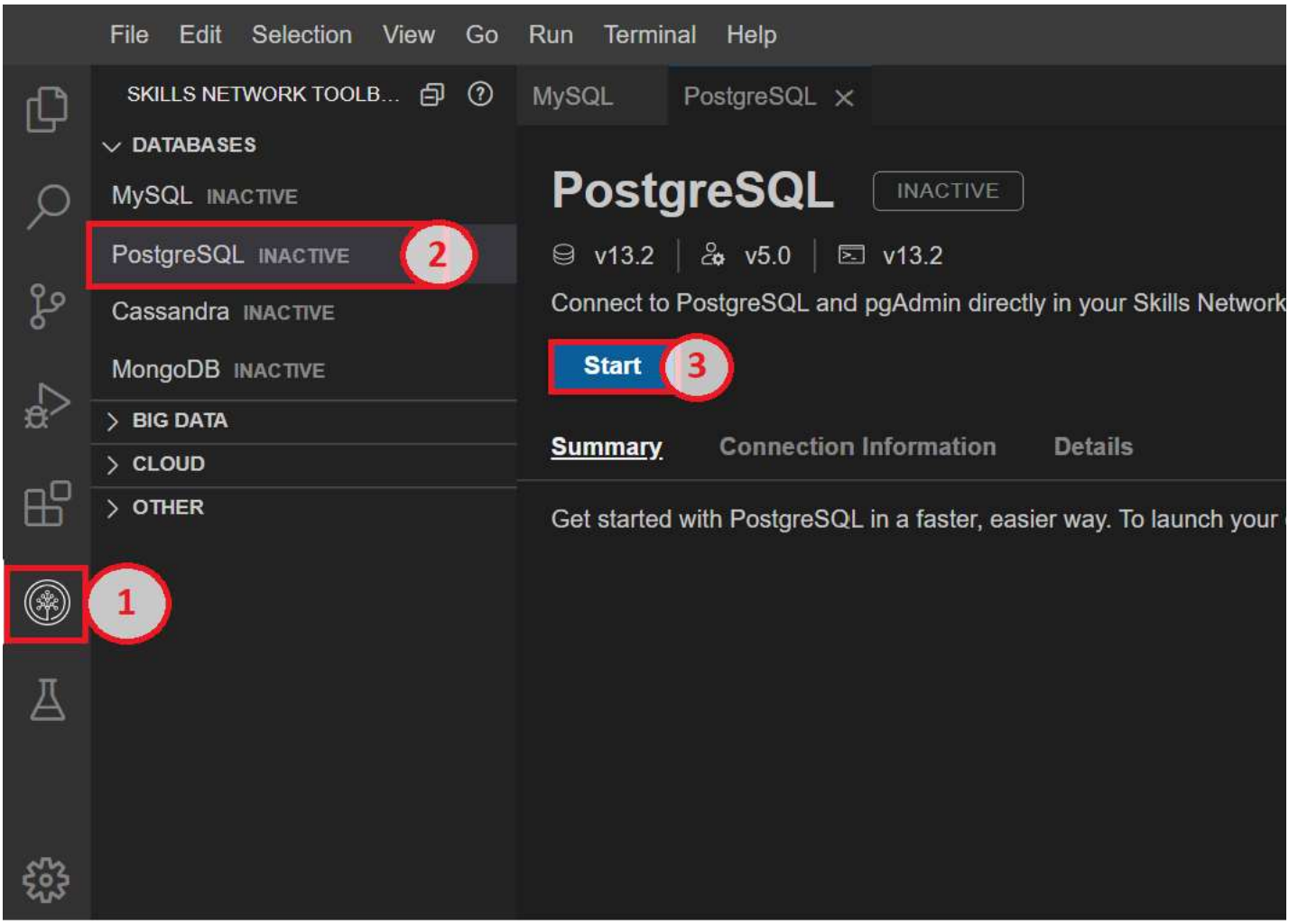
Lab structure

In this exercise, you will go through several subtasks where you will use the PostgreSQL command line interface (CLI) to create a database and restore the structure and contents of tables. Then, you will learn how to explore and query tables. Finally, you will learn how to dump/backup tables from a database.

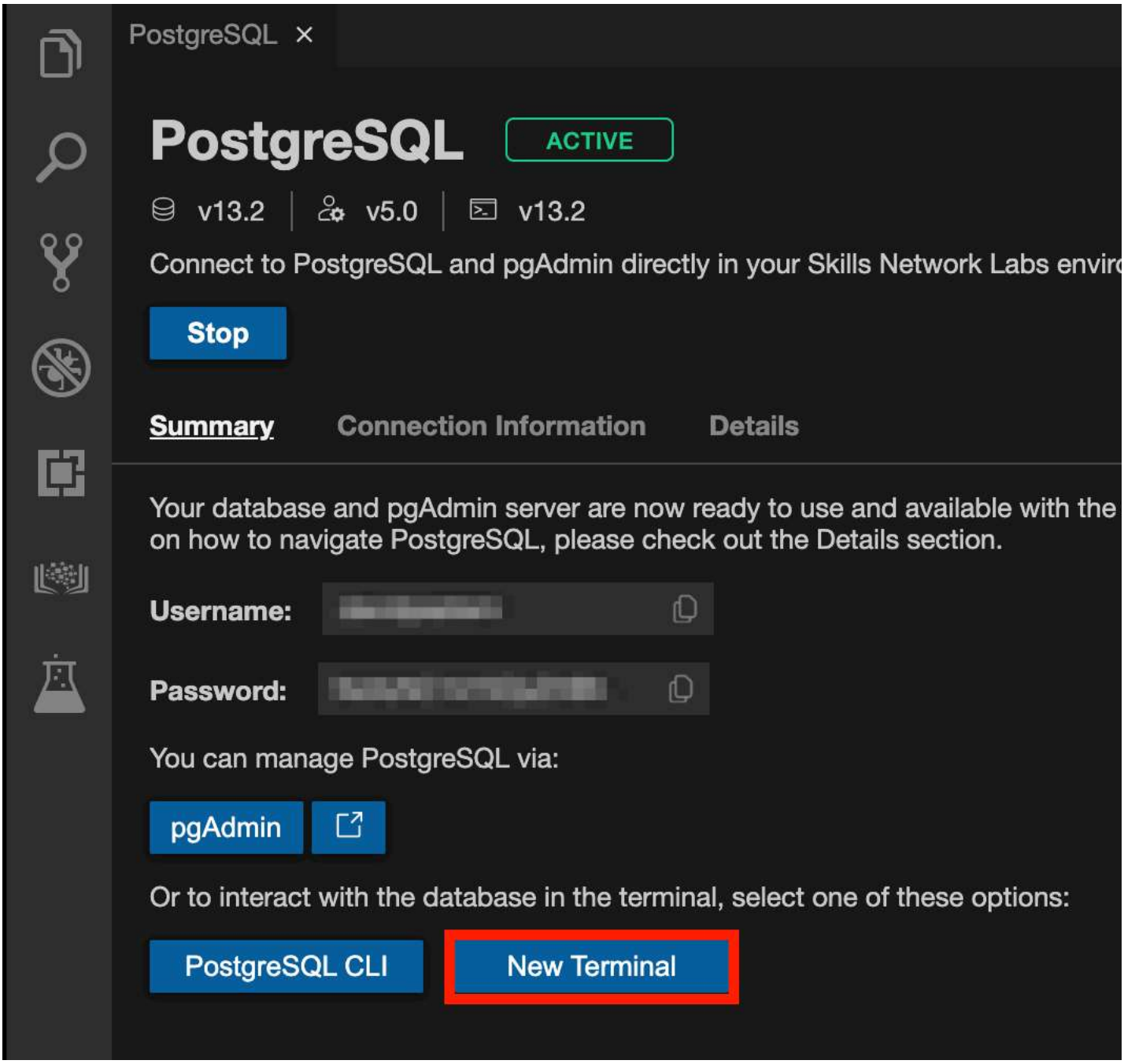
Task A: Create a database

To get started with this lab, launch PostgreSQL using the Cloud IDE. You can do this by following these steps:

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



4. Open a new command terminal by clicking **New Terminal**.



5. Copy the command below by clicking the little copy button on the right of the code block and then paste it into the terminal using **Ctrl + V** (Mac: ⌘ + V) to fetch the sakila_pgsql_dump.sql file to the Cloud IDE.

```
1. 1
1. wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0110EN-SkillsNetwork/datasets/sakila/sakila_pgsql_dump.sql
```

Copied!

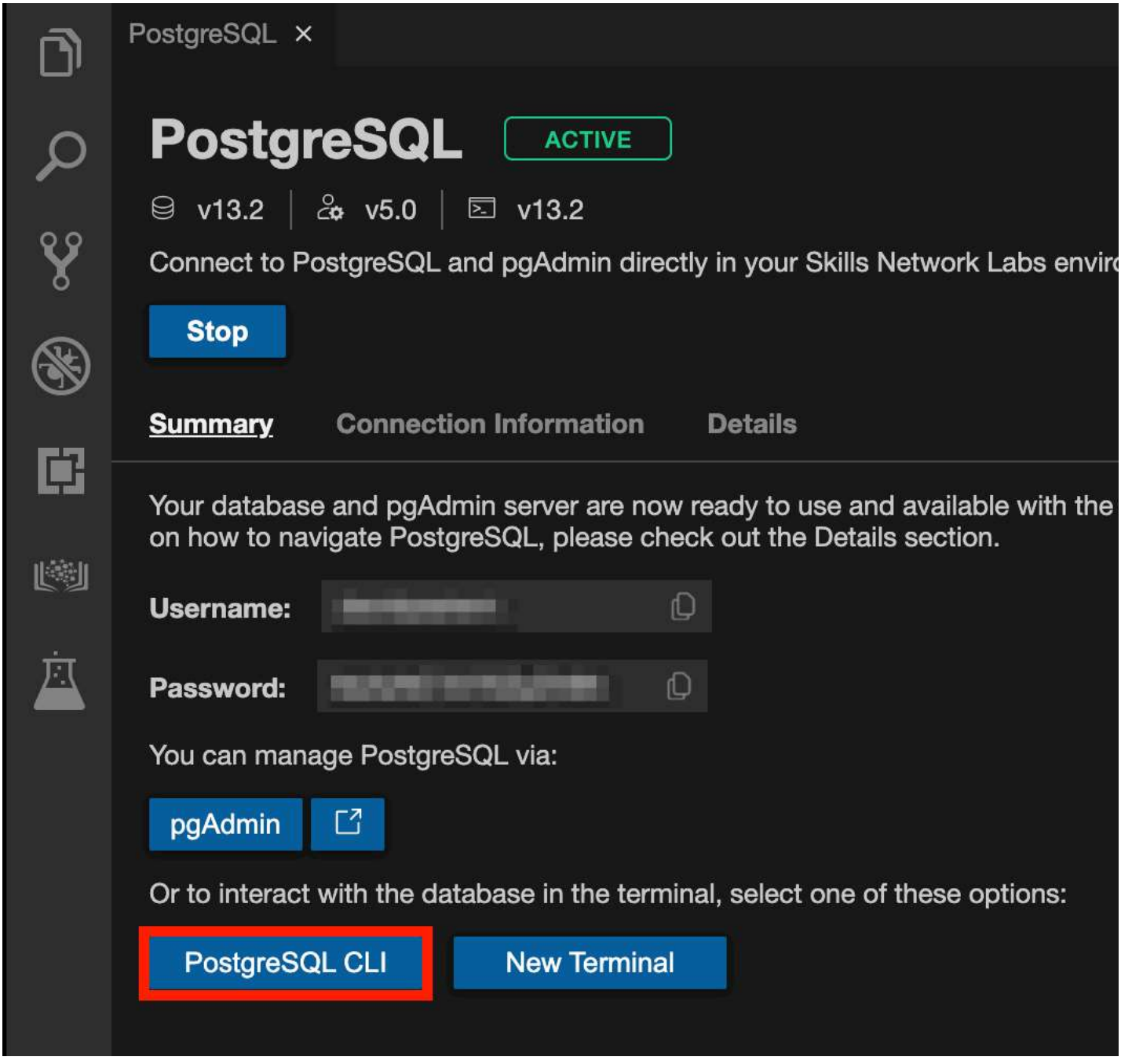
```
theia@theiadocker-sandipsahajo:/home/project$ wget https://cf-courses-d
--2021-03-22 04:19:25--  https://cf-courses-data.s3.us.cloud-object-sto
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (c
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.clou
HTTP request sent, awaiting response... 200 OK
Length: 2764132 (2.6M) [application/x-sql]
Saving to: 'sakila_pgsql_dump.sql'

sakila_pgsql_dump.sql                100%[=========================

2021-03-22 04:19:28 (972 KB/s) - 'sakila_pgsql_dump.sql' saved [2764132

theia@theiadocker-sandipsahajo:/home/project$
```

6. Now, open the PostgreSQL Command Line Interface (CLI) by clicking **PostgreSQL CLI**.



7. Create a new database named **sakila** using the following command in the terminal:

```
1. 1
1. create database sakila;
```
Copied!

```
theia@theiadocker-sandipsahajo:/home/project$ psql --username=postgres
Password:
psql (13.2 (Ubuntu 13.2-1.pgdg18.04+1))
Type "help" for help.

postgres=# create database sakila;
CREATE DATABASE
postgres=#
```

> **Note:** You are using the **create database** command to create a new database within the PostgreSQL CLI. To create a new database named sakila outside the command line interface, you can use the following command directly in a terminal window: `createdb --username=postgres --host=localhost --password sakila` after quitting the psql command prompt session with the command `\q`.

Task B: Restore the structure and data of a table

1. To connect to the newly created empty sakila database, use the following command in the terminal and enter your PostgreSQL service session password:

```
1. 1
1. \connect sakila;
```
Copied!

```
postgres=# \connect sakila;
Password:
You are now connected to database "sakila" as user "postgres".
```

2. Restore the sakila PostgreSQL dump file (containing the sakila database table definitions and data) to the newly created empty sakila database by using the following command in the terminal:

```
1. 1
1. \include sakila_pgsql_dump.sql;
```
Copied!

```
sakila=# \include sakila_pgsql_dump.sql;
```

> **Note:** You are using the **\include** command to restore the database dump file within the PostgreSQL CLI. To restore the database dump file outside of the Command Line Interface, you can use the command `pg_restore --username=postgres --host=localhost --password --dbname=sakila < sakila_pgsql_dump.tar` after quitting the CLI prompt session with the command `\q`. Non-text format **.tar** dumps are restored using the **pg_restore** command. So, before using the **pg_restore** command, first, fetch the .tar version of this dump file using the command `wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0110EN-SkillsNetwork/datasets/sakila/sakila_pgsql_dump.tar`

3. Repeat Step 1 to reconnect to the sakila database after restoring the dump file.

Task C: Explore and query tables

1. To list all the table names from the sakila database, use the following command in the terminal:

```
1. 1
1. \dt
```
Copied!

```
sakila=# \dt;
             List of relations
 Schema |     Name      | Type  |  Owner
--------+---------------+-------+----------
 public | actor         | table | postgres
 public | address       | table | postgres
 public | category      | table | postgres
 public | city          | table | postgres
 public | country       | table | postgres
 public | customer      | table | postgres
 public | film          | table | postgres
 public | film_actor    | table | postgres
 public | film_category | table | postgres
 public | inventory     | table | postgres
 public | language      | table | postgres
 public | payment       | table | postgres
 public | rental        | table | postgres
 public | staff         | table | postgres
 public | store         | table | postgres
(15 rows)

sakila=#
```

2. Explore the structure of the **store** table using the following command in the terminal:

```
1. 1
1. \d store;
```
Copied!

```
sakila=# \d store;
                                        Table "public.store"
      Column      |            Type             | Collation | Nullable
------------------+-----------------------------+-----------+----------
 store_id         | integer                     |           | not null
 manager_staff_id | smallint                    |           | not null
 address_id       | smallint                    |           | not null
 last_update      | timestamp without time zone |           | not null
Indexes:
    "store_pkey" PRIMARY KEY, btree (store_id)
    "idx_unq_manager_staff_id" UNIQUE, btree (manager_staff_id)
Foreign-key constraints:
    "store_address_id_fkey" FOREIGN KEY (address_id) REFERENCES addres
    "store_manager_staff_id_fkey" FOREIGN KEY (manager_staff_id) REFER
Triggers:
    last_updated BEFORE UPDATE ON store FOR EACH ROW EXECUTE FUNCTION

sakila=#
```

3. Retrieve all the records from the **store** table using the following command in the terminal:

```
1. 1
1. SELECT * FROM store;
```
Copied!

```
sakila=# SELECT * FROM store;
 store_id | manager_staff_id | address_id |     last_update
----------+------------------+------------+---------------------
        1 |                1 |          1 | 2006-02-15 09:57:12
        2 |                2 |          2 | 2006-02-15 09:57:12
(2 rows)
```

4. Quit the PostgreSQL command prompt session using the following command in the terminal.

```
1. 1
1. \q
```
Copied!

```
sakila=# \q
theia@theiadocker-sandipsahajo:/home/project$
```

Task D: Dump/backup tables from a database

1. Finally, to dump/backup the **store** table from the database, use the following command in the terminal and enter your PostgreSQL service session password:

```
1. 1
1. pg_dump --username=postgres --host=localhost --password --dbname=sakila --table=store --format=plain > sakila_store_pgsql_dump.sql
```
Copied!

> **Note:** To only dump/backup the table **store** from the database in non-text format **.tar**, you can use the command `pg_dump --username=postgres --host=localhost --password --dbname=sakila --table=store --format=tar > sakila_store_pgsql_dump.tar`

2. To view the dump file within the terminal, use the following command:

```
1. 1
1. cat sakila_store_pgsql_dump.sql
```
Copied!

```
theia@theiadocker-sandipsahajo:/home/project$ pg_dump --username=postg
Password:
theia@theiadocker-sandipsahajo:/home/project$ cat sakila_store_pgsql_d
--
-- PostgreSQL database dump
--

-- Dumped from database version 13.2
-- Dumped by pg_dump version 13.2 (Ubuntu 13.2-1.pgdg18.04+1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- Name: store; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.store (
    store_id integer DEFAULT nextval('public.store_store_id_seq'::regc
    manager_staff_id smallint NOT NULL,
    address_id smallint NOT NULL,
    last_update timestamp without time zone DEFAULT now() NOT NULL
);
```

Conclusion

Congratulations! You have completed this lab, and now you have learned how to create a database, restore the structure and data of a table, explore and query tables, and dump/backup tables from a database.

Author: Sandip Saha Joy

Other Contributors: David Pasternak

Skills Network