

# Hands-on Lab: Setting up a staging area



**Estimated time needed: 30 minutes**

## Purpose of the Lab:

The purpose of the lab is to equip you with practical skills in setting up and managing a staging server for a data warehouse, specifically using PostgreSQL. The lab focuses on teaching how to design and implement a database schema, load data into tables, and run sample queries to interact with the data. This is aimed at providing you with a hands-on understanding of the intricacies involved in preparing and managing a data warehouse environment.

## Benefits of Learning the Lab:

The key benefit of this lab is the hands-on experience it offers in data warehousing concepts and practices. By working in the Skills Network Cloud IDE, an interactive environment, you can practice and hone your database management skills. This practical exposure is crucial for developing a solid foundation in data management and analysis, skills that are highly valued in real-world business scenarios. The lab helps you to understand the operational aspects of data warehouses, making you better prepared for challenges in data management and analysis in professional settings.

## Objectives

In this lab you will:

- Setup a staging server for a data warehouse
- Create the schema to store the data
- Load the data into the tables
- Run a sample query

## About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands on labs for course and project related labs. Theia is an open source IDE (Integrated Development Environment), that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on Theia running in a Docker container.

## Important Notice about this lab environment

Please be aware that sessions for this lab environment are not persistent. A new environment is created for you every time you connect to this lab. Any data you may have saved in an earlier session will get lost. To avoid losing your data, please plan to complete these labs in a single session.

## Exercise 1 - Start the PostgreSQL server

We will be using the PostgreSQL server as our staging server.

Start the PostgreSQL server.

Open a new terminal, by clicking on the menu bar and selecting **Terminal->New Terminal**.

This will open a new terminal at the bottom of the screen.

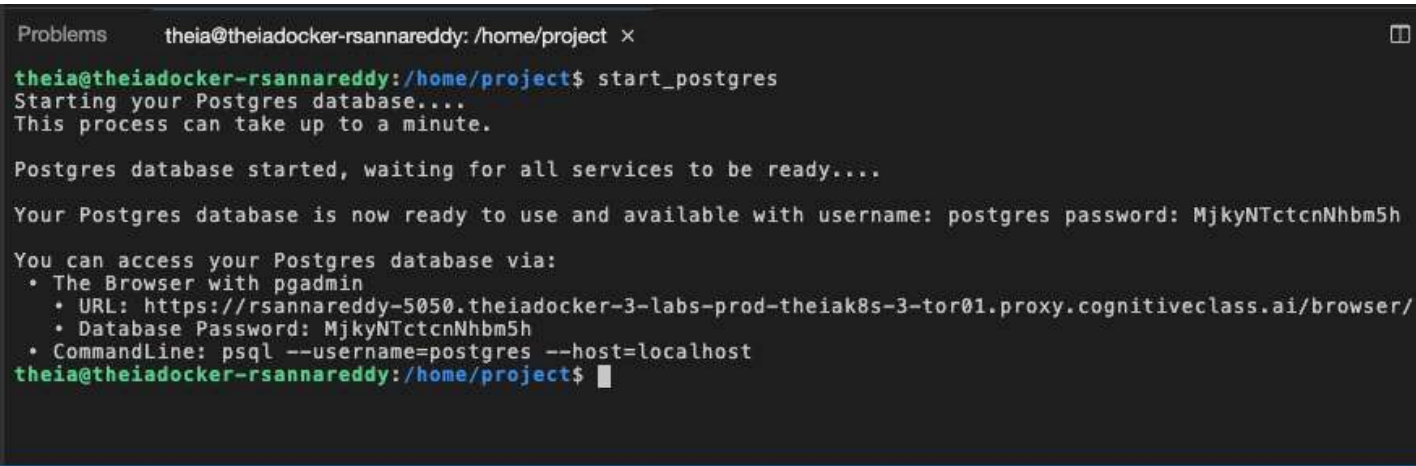
Run the commands below on the newly opened terminal. (You can copy the code by clicking on the little copy button on the bottom right of the codeblock below and then paste it, wherever you wish.)

Start the PostgreSQL server, by running the command below:

- 1
1. start\_postgres

Copied! Executed!

You should see an output similar to the one below.

A terminal window with a dark background. The title bar shows 'Problems' and 'theia@theiadocker-rsannareddy: /home/project'. The terminal content shows the command 'start\_postgres' being executed. The output indicates that the PostgreSQL database is starting, waiting for services to be ready, and is now ready to use with a specific username and password. It also provides instructions on how to access the database via a browser or command line.

```
theia@theiadocker-rsannareddy:/home/project$ start_postgres
Starting your Postgres database....
This process can take up to a minute.

Postgres database started, waiting for all services to be ready....

Your Postgres database is now ready to use and available with username: postgres password: MjkyNTtctcNhbM5h

You can access your Postgres database via:
• The Browser with pgadmin
  • URL: https://rsannareddy-5050.theiadocker-3-labs-prod-theiak8s-3-tor01.proxy.cognitiveclass.ai/browser/
  • Database Password: MjkyNTtctcNhbM5h
• CommandLine: psql --username=postgres --host=localhost
theia@theiadocker-rsannareddy:/home/project$
```

## Exercise 2 - Create Database

Create the database on the data warehouse.

Using the createdb command of the PostgreSQL server, we can directly create the database from the terminal.

Run the command below to create a database named billingDW.

- 1
1. createdb -h postgres -U postgres -p 5432 billingDW

Copied! Executed!

In the above command

- -h mentions that the database server is accessible using the hostname “postgres”
- -U mentions that we are using the user name postgres to log into the database
- -p mentions that the database server is running on port number 5432

You should see an output like this.

```
theia@theiadocker-rsannareddy:/home/project$ createdb -h localhost -U postgres -p 5432 billingDW
theia@theiadocker-rsannareddy:/home/project$
```

## Exercise 3 - Create data warehouse schema

Step 1:

Download the schema files.

The commands to create the schema are available in the file below.

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0260EN-SkillsNetwork/labs/Setting%20up%20a%20staging%20area/billing-datawarehouse.tgz>

Run the commands below to download and extract the schema files.

- 1.
- 2.
- 3.
- 4.

1. `wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0260EN-SkillsNetwork/labs/Setting%20up%20a%20staging%20area/billing-datawarehouse.tgz`
- 2.
3. `tar -xvzf billing-datawarehouse.tgz`
4. `ls *.sql`

Copied! Executed!

You should see 4 .sql files listed in the output

Step 2: Create the schema

Run the command below to create the schema in the billingDW database.

- 1.
1. `psql -h postgres -U postgres -p 5432 billingDW < star-schema.sql`

Copied! Executed!

You should see an output similar to the one below.

```
theia@theiadocker-rsannareddy:/home/project$ psql -h localhost -U postgres -p 5432 billingDW < star-schema.sql
BEGIN
CREATE TABLE
CREATE TABLE
CREATE TABLE
ALTER TABLE
ALTER TABLE
COMMIT
theia@theiadocker-rsannareddy:/home/project$
```

## Exercise 4 - Load data into Dimension tables

When we load data into the tables, it is a good practice to load the data into dimension tables first.

Step 1: Load data into DimCustomer table

Run the command below to load the data into DimCustomer table in billingDW database.

- 1
1. `psql -h postgres -U postgres -p 5432 billingDW < DimCustomer.sql`

Copied! Executed!

Step 2: Load data into DimMonth table

Run the command below to load the data into DimMonth table in billingDW database.

- 1
1. `psql -h postgres -U postgres -p 5432 billingDW < DimMonth.sql`

Copied! Executed!

## Exercise 5 - Load data into Fact table

Load data into FactBilling table

Run the command below to load the data into FactBilling table in billingDW database.

- 1
1. `psql -h postgres -U postgres -p 5432 billingDW < FactBilling.sql`

Copied! Executed!

## Exercise 6 - Run a sample query

Run the command below to check the number of rows in all the tables in the billingDW database.

- 1

1. `psql -h postgres -U postgres -p 5432 billingDW < verify.sql`

Copied! Executed!

You should see an output similar to the one below.

```
theia@theiadocker-rsannareddy:/home/project$ psql -h localhost -U postgres -p 5432 billingDW < verify.sql
"Checking row in DimMonth Table"
count
-----
    132
(1 row)

"Checking row in DimCustomer Table"
count
-----
   1000
(1 row)

"Checking row in FactBilling Table"
count
-----
 132000
(1 row)
```

Your data warehouse staging area is now ready.

## Practice exercises

In this practice session, you will create a database named `practice` and load the data into it.

1. Problem:

*Create a database named `practice`.*

- ▶ [Click here for Hint](#)
- ▶ [Click here for Solution](#)

2. Problem:

*In the `practice` database, create a schema using `star-schema.sql`.*

- ▶ [Click here for Hint](#)
- ▶ [Click here for Solution](#)

3. Problem:

*In the `practice` database, load the data into all tables using the `DimMonth.sql`, `DimCustomer.sql` and `FactBilling.sql`.*

- ▶ [Click here for Hint](#)
- ▶ [Click here for Solution](#)

4. Problem:

*\_Verify that you have correctly loaded the data into the `practice` database.*

- ▶ [Click here for Hint](#)
- ▶ [Click here for Solution](#)

Congratulations!! You have successfully finished the Setting up a staging server lab.

## **Authors**

Ramesh Sannareddy

## **Other Contributors**

Rav Ahuja

© **IBM Corporation 2023. All rights reserved.**