User Guide

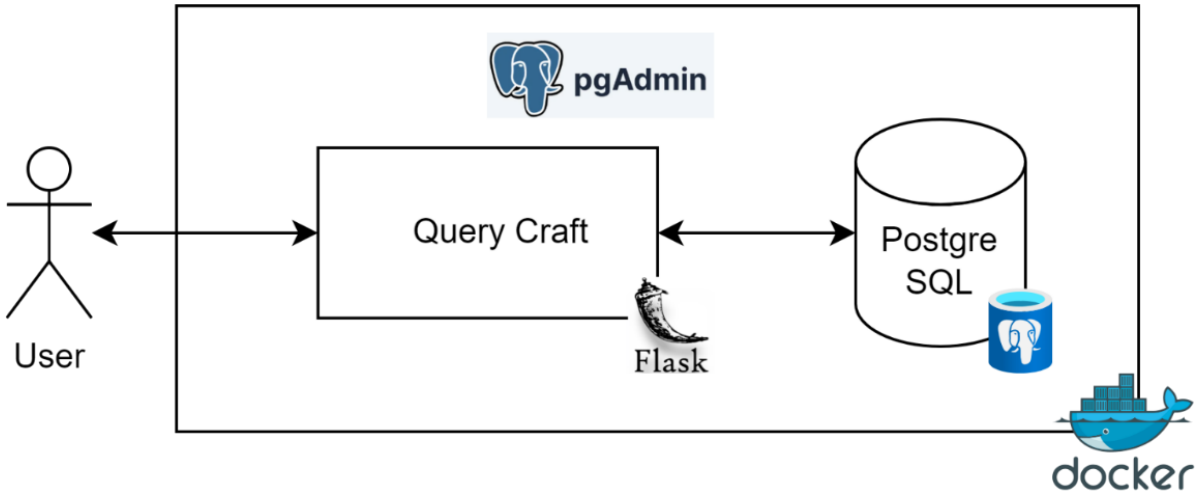
Overview

Query Craft is an educational tool designed to guide students how to craft SQL queries effectively. It provides interactive learning materials, examples, and practice problems to help users understand various SQL concepts and improve their query writing skills.

Key Features

- Interactive Learning Modules:
    - The application has a series of lab materials that cover different aspects of SQL, from basic queries to advanced topics like joins and normalization.
    - Each module includes explanations, examples, and visual aids to help users grasp the concepts.
- Practice Problems:
    - Students can practice their SQL skills with various queries designed to test their understanding of the material.
- Visual Aids:
    - The application includes images and diagrams to help explain complex concepts visually.
- Interactive Query Execution:
    - Users can write and execute SQL queries within the application to get immediate feedback on their code.
- Docker Integration:
    - The application can be easily deployed using Docker, making it simple to set up and run on different systems.

System Design



The architecture diagram illustrates the flow of data and interactions between different components. At the core of the system is the Query Craft application built using Flask, a lightweight Python web framework. Users interact with the Flask-based application, where they can practice queries and learn databases through a series of lab materials covering various aspects of SQL, from basic queries to advanced topics with examples. All the exercises and practice queries written interact with the PostgreSQL instance running. The application is deployed in a Dockerized environment, making it suitable for cloud platforms or any environment supporting Docker.

Technical Overview

The application is built using a combination of Python, Flask, JavaScript, and Docker. Below is a high-level overview of how it works:

- Backend
    - The backend is developed using Python and Flask. Flask handles the routing and serves the HTML templates to the frontend.
    - The app.py file is the main entry point of the application.
- Frontend
    - HTML templates are in the templates directory and are rendered by Flask.

    - JavaScript files in the static/js directory handle interactive elements and query execution.
- Docker:
    - The application includes a Dockerfile and a docker-compose.yml file for containerization.

How to Set Up and Run the Application

- Prerequisites
    - Ensure you have Docker installed on your system.
- Clone the Repository
    - Clone the repository to your local machine.
- Build and Run the Application:
    - Navigate to the project directory and run the following command to build and start the application
        - docker-compose up --build
- Access the Application:
    - Once the application is running, you can access it by navigating to http://localhost:<port number in docker-compose.yml file>in your web browser.
- Exporting the Application
    - Once the docker images are built **run** the **export-query-craft**-mac.sh shell script on a MAC machine or the **export-query-craft**-windows.bat batch file on a windows environment.

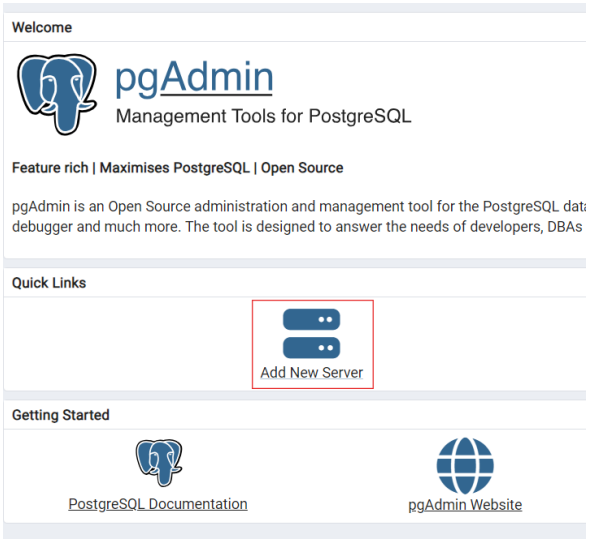
How to setup pgAdmin

- **Open docker-compose.yml**: Locate and open docker-compose.yml file in a text editor.
- **Find pgAdmin Service**: Look for the pgAdmin service section.
- **Identify Port Number**: Note the port number before the colon in the ports section (e.g., "8081:80" - here, 8081 is the port number).
- **Construct URL**: Replace <port number> with the identified port number in http://localhost:<port number>.
  Example: http://localhost:8081. If you haven't made any changes to the Docker Compose file, you can use this url directly.
- **Access pgAdmin**: Open the constructed URL in your web browser to access pgAdmin.

On the Login Page :

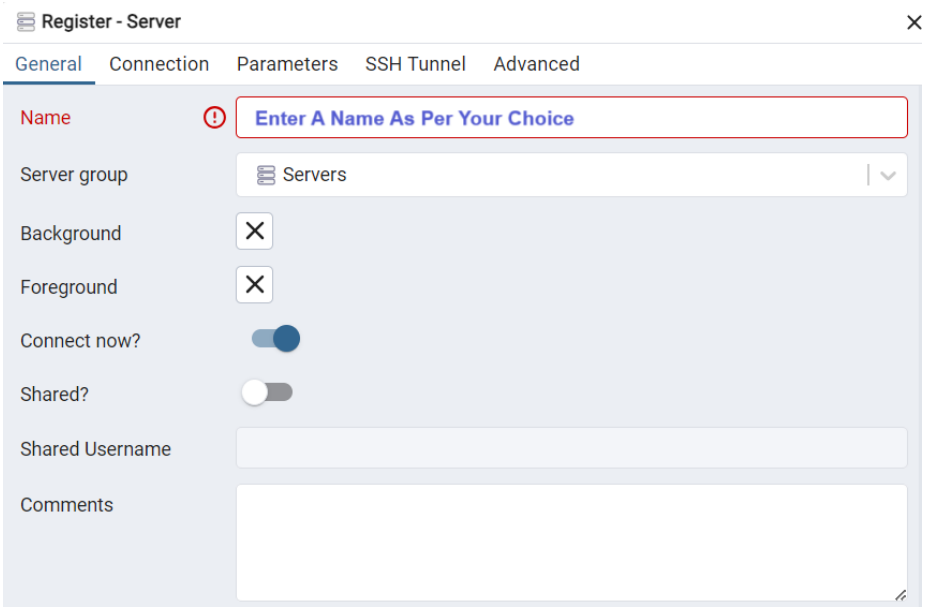
Username: In the Docker Compose File → pgAdmin services section
→ PGADMIN_DEFAULT_EMAIL
Attribute Value (username@mail.com - Do not change this value)
Password: In the Docker Compose File → pgAdmin services section
→ PGADMIN_DEFAULT_PASSWORD
Attribute Value (password - Do not change this value)

Once you login successfully, in the **Home Page** select **Add New Server**.



In the **General Tab** of the **Register Server Window**

Enter a **Name** as per your choice and leave the rest as the defaults.



In the **Connection Tab** of the **Register Server Window**

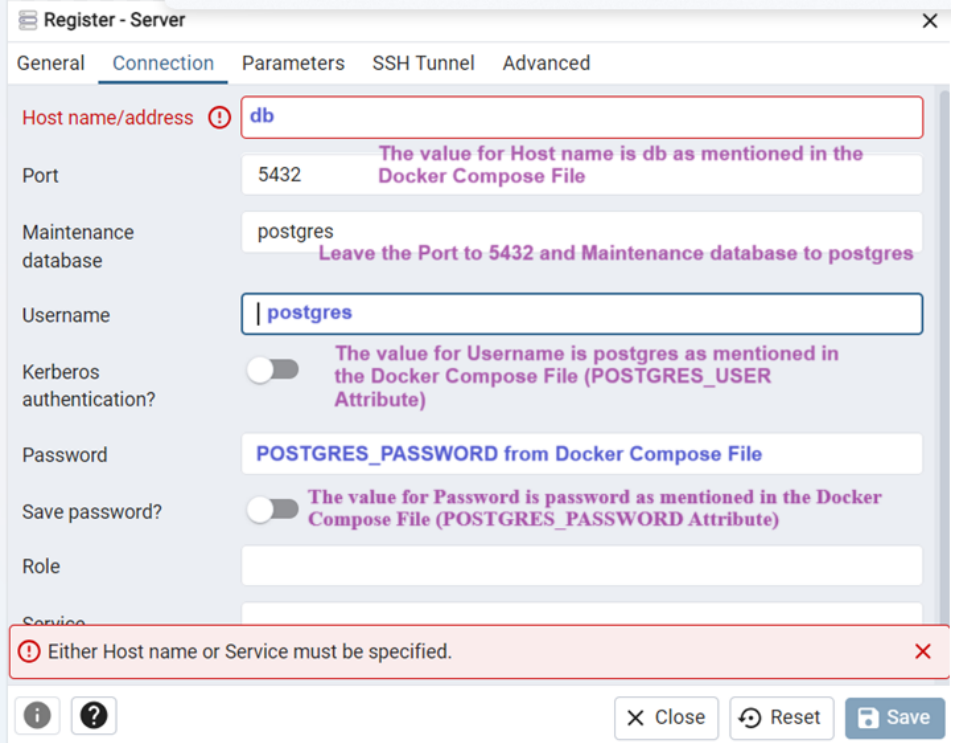
Host name: db (Do not change)

Port: 5432 (Do not change)

Maintenance database: postgres

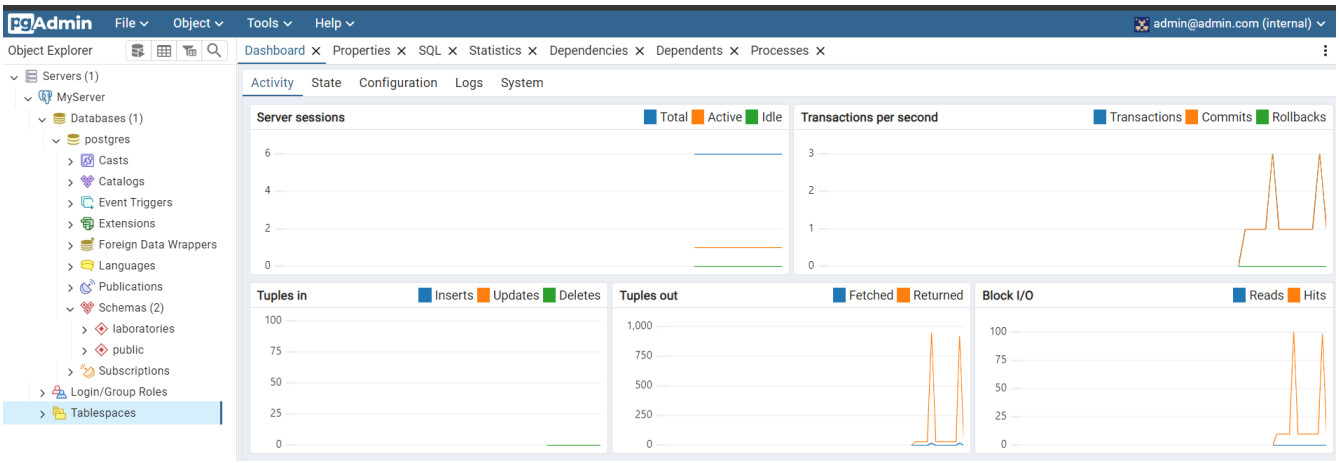
Username: In the Docker Compose File → db services section → POSTGRES_USER Attribute Value (postgres – Do not change this value)

Password: In the Docker Compose File → db services section → POSTGRES_PASSWORD Attribute Value (password - Do not change this value)



Leave the other values as the defaults and Save the Server.

A database will be created with the Name you created.



Use pgAdmin (a management tool for PostgreSQL) from here.

The ingestion_script_files folder contains the ingestion script files for the query practice schema.

Conclusion

Query Craft is a powerful tool for learning SQL in an interactive and engaging way. Its combination of theoretical materials, practical exercises, and visual aids make it an excellent resource for both beginners and advanced users looking to improve their SQL skills.