PostgreSQL Docker Setup and Basic Commands

PostgreSQL Docker Setup

1. Pull Docker Images

docker pull postgres:14

docker pull btholt/complete-intro-to-sql

* + postgres:14: Official PostgreSQL image, version 14.
  + btholt/complete-intro-to-sql: Course-specific image.

1. Run PostgreSQL Container

docker run -e POSTGRES\_PASSWORD=lol --name=pg --rm -d -p 5432:5432 postgres:14

* + -e POSTGRES\_PASSWORD=lol: Set PostgreSQL password to lol.
  + --name=pg: Name the container pg.
  + --rm: Remove the container when it exits.
  + -d: Run in detached mode (background).
  + -p 5432:5432: Map port 5432 of the container to port 5432 on the host.
  + postgres:14: Use PostgreSQL version 14.

1. Verify Container is Running

sudo docker ps

1. Stop PostgreSQL Container

sudo docker kill pg

* + This command stops and removes the container.

1. Access PostgreSQL Container

sudo docker exec -u postgres -it pg psql

* + -u postgres: Use the postgres user.
  + -it: Interactive terminal.

PostgreSQL Basic Commands

1. List Databases

\l

1. Create a New Database

CREATE DATABASE recipeguru;

1. Connect to a Database

\c recipeguru

1. List Tables

\d

1. Create a Table

CREATE TABLE ingredients (

id INTEGER PRIMARY KEY GENERATED ALWAYS AS IDENTITY,

title VARCHAR (255) UNIQUE NOT NULL

);

* + id INTEGER PRIMARY KEY GENERATED ALWAYS AS IDENTITY: Auto-incrementing primary key.
  + title VARCHAR (255) UNIQUE NOT NULL: Text field with a maximum of 255 characters.

1. Alter Table

ALTER TABLE ingredients ADD COLUMN image VARCHAR (255);

ALTER TABLE ingredients DROP COLUMN image;

1. Query Data

SELECT \* FROM ingredients;

1. Comments in SQL

-- This is a single-line comment

1. Handling Conflicts

INSERT INTO ingredients (title) VALUES ('Sugar') ON CONFLICT DO NOTHING;

* + ON CONFLICT DO NOTHING: Skip the insert if a conflict occurs (e.g., duplicate unique key).

**INSERT INTO ingredients (title, image, type)**

VALUES ('watermelon', 'banana.jpg', 'this won''t be updated')

ON CONFLICT (title) DO UPDATE

SET image = EXCLUDED.image;

* + ON CONFLICT (title) DO UPDATE SET image = EXCLUDED.image;: Updates the image field if there is a conflict on the title field. EXCLUDED refers to the row that would have been inserted. (Updates existing row)

Updating Data in PostgreSQL

1. Basic Update Command

UPDATE ingredients

SET image = 'watermelon.jpg'

WHERE title = 'watermelon';

* + Explanation: This command updates the image field for the row where the title is 'watermelon'.
  + Important Note: If you don't include the WHERE clause, all rows in the table will be updated!

1. Returning Updated Data

UPDATE ingredients

SET image = 'watermelon.jpg'

WHERE title = 'watermelon'

RETURNING id, title, image;

* + Explanation: This command updates the image field and returns the id, title, and image of the updated rows.
  + The RETURNING Clause: It tells PostgreSQL to return specified columns of the rows that were updated.

1. Return All Columns for Updated Data

UPDATE ingredients

SET image = 'watermelon.jpg'

WHERE title = 'watermelon'

RETURNING \*;

* + Explanation: This command updates the image field and returns all columns for the updated rows.

Deleting Data in PostgreSQL

1. Basic Delete Command

DELETE FROM ingredients

WHERE image = 'different.jpg';

* + Explanation: This command deletes the row(s) where the image field is 'different.jpg'.

1. Returning Deleted Data

DELETE FROM ingredients

WHERE image = 'different.jpg'

RETURNING \*;

* + Explanation: This command deletes the row(s) where the image is 'different.jpg' and returns all columns for the deleted rows.

Selecting Data in PostgreSQL

1. Basic SELECT Command

SELECT \* FROM ingredients;

* + Explanation: This selects all columns and rows from the ingredients table.
  + Best Practice: It’s better to specify the columns explicitly, especially in large databases, to avoid unnecessary performance overhead.

1. Specify Columns for Selection

SELECT id, title, image, type FROM ingredients;

* + Explanation: This query selects only specific columns: id, title, image, and type from the ingredients table.

1. Limit and Offset Data

SELECT title FROM ingredients LIMIT 10 OFFSET 20;

* + Explanation: This query selects the title column from ingredients, limits the result to 10 rows, and skips the first 20 rows.
  + Note: Using OFFSET may cause items to be missed if new rows are added to the table during paging. It’s generally better to use WHERE clauses for more efficient and reliable pagination.

1. Using WHERE for Efficient Filtering

SELECT title FROM ingredients WHERE id > 30;

* + Explanation: This filters the ingredients table to show only rows where the id is greater than 30. Using WHERE is more optimal than OFFSET in large datasets.

1. Using Not Equal in WHERE Clause

SELECT title FROM ingredients WHERE type <> 'fruits';

* + Explanation: <> means "not equal." This query selects rows where type is not equal to 'fruits'.

1. Range Queries and Limits

SELECT \* FROM ingredients WHERE id >= 10 AND id <= 20 LIMIT 10;

* + Explanation: This selects rows where the id is between 10 and 20, and limits the results to 10 rows.

1. Using OR in WHERE Clause

SELECT \* FROM ingredients WHERE id <= 10 OR id >= 20;

* + Explanation: This returns rows where the id is less than or equal to 10 or greater than or equal to 20.

1. Ordering Results

SELECT \* FROM ingredients ORDER BY id DESC LIMIT 10;

* + Explanation: This query returns all rows ordered by id in descending order, limiting the results to 10 rows.

1. Pattern Matching with LIKE

SELECT \* FROM ingredients WHERE title LIKE 'pota';

* + Explanation: This matches rows where title starts with 'pota'.

SELECT \* FROM ingredients WHERE title LIKE '%pota%';

* + Explanation: This matches rows where 'pota' appears anywhere in the title.

Built-in Functions in PostgreSQL

1. Concatenating Strings

SELECT \* FROM ingredients WHERE CONCAT(title, type) LIKE '%fruit%';

* + Explanation: This concatenates the title and type columns, then checks if the string 'fruit' appears anywhere in the result.

1. Handling Case Sensitivity

SELECT \* FROM ingredients WHERE LOWER(CONCAT(title, type)) LIKE LOWER('%FrUiT%');

* + Explanation: This converts both the concatenated string and the search value to lowercase, making the query case-insensitive.

1. Case-Insensitive LIKE with ILIKE

SELECT \* FROM ingredients WHERE CONCAT(title, type) ILIKE '%FrUiT%';

* + Explanation: ILIKE is a case-insensitive version of LIKE.

Wildcards in LIKE and ILIKE

1. Wildcard Matching with %

SELECT \* FROM ingredients WHERE title ILIKE 'c%';

* + Explanation: This matches rows where the title starts with 'c'.

SELECT \* FROM ingredients WHERE title ILIKE '%berry%';

* + Explanation: This matches rows where 'berry' appears anywhere in the title.

1. Single Character Matching with \_

SELECT \* FROM ingredients WHERE title ILIKE 'ch\_rry';

* + Explanation: This matches rows where the title has any single character in place of the \_, such as 'cherry' or 'churry', but not 'chrry'.