**Propmt : Explain postgresql insert statement in depth for beginners with syntax and 2 detailed examples**

The **INSERT** statement in PostgreSQL is used to add new rows of data into a table. It's the primary way to populate your database with information. Think of it like adding a new entry to a spreadsheet.

**Syntax**

The most common syntax for the INSERT statement is:

SQL

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

* **INSERT INTO table\_name**: This specifies the target table for the new data.
* **(column1, column2, ...)**: This is an optional but highly recommended list of the columns you want to add data to. It's a good practice to include this for clarity and to prevent errors if the table's structure changes.
* **VALUES (value1, value2, ...)**: This provides the actual data to be inserted. The values must correspond to the columns in the order you've listed them, and their data types must be compatible.

**Example 1: Inserting into a Simple users Table**

Let's assume we have a simple table to store user information, with a unique ID, username, and email.

SQL

CREATE TABLE users (

user\_id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL

);

To add a new user named 'johndoe' with an email, you would use this INSERT statement:

SQL

INSERT INTO users (username, email)

VALUES ('johndoe', 'john.doe@example.com');

**Explanation:**

* We're inserting a new record into the users table.
* We specify the username and email columns. We **omit** the user\_id column because it's a **SERIAL** type. PostgreSQL automatically generates a unique, sequential number for this column, so you don't need to provide it.
* The VALUES clause provides the data 'johndoe' for the username and 'john.doe@example.com' for the email. The single quotes are essential for text data.

**Example 2: Inserting Multiple Rows and Using Default Values**

This example demonstrates how to insert multiple rows at once, which is more efficient for bulk data entry. We'll use a products table that has a default value for a column.

First, let's create the products table, which includes a created\_at column with a default value of the current timestamp.

SQL

CREATE TABLE products (

product\_id SERIAL PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT\_TIMESTAMP

);

To insert three products at once, you can provide multiple sets of values separated by commas:

SQL

INSERT INTO products (product\_name, price)

VALUES

('Laptop', 1200.00),

('Mouse', 25.50),

('Keyboard', 75.00);

**Explanation:**

* We're adding three new products to the table in a single command.
* We specify the product\_name and price columns.
* We **omit** the product\_id (a SERIAL column) and the created\_at column. Since created\_at has a **DEFAULT** value, PostgreSQL will automatically set it to the current time for each new row.
* The VALUES clause provides the data for each product in the order specified.