**PROMPT :- Give me briefly description about CREATE TABLE in postgerSQL, some commonly used data types, syntax and constraint. Also, give two example of CREATE TABLE  as works for fresher.**

**CREATE TABLE** is a fundamental command in PostgreSQL used to define and create a new table in a database. It specifies the table's name, the columns it will contain, and the data type for each column. This process essentially structures the database by creating a blueprint for where data will be stored.

**Common Data Types**

PostgreSQL offers a wide array of data types. Here are some of the most common ones:

* **Integer Types**: INTEGER (or INT), SMALLINT, BIGINT. These are used for whole numbers. INTEGER is the most common choice.
* **Floating-Point Types**: REAL, DOUBLE PRECISION, NUMERIC. Used for numbers with decimal points. NUMERIC is ideal for financial data as it provides exact storage.
* **Character Types**: VARCHAR(n), CHAR(n), TEXT.
  + VARCHAR(n) stores strings of varying lengths, up to a specified maximum 'n'. It's widely used for names and addresses.
  + CHAR(n) stores strings of a fixed length 'n', padding with spaces if necessary.
  + TEXT stores strings of any length without a limit.
* **Date/Time Types**: DATE, TIME, TIMESTAMP.
  + DATE stores just the date.
  + TIME stores just the time of day.
  + TIMESTAMP stores both date and time.
* **Boolean Type**: BOOLEAN (or BOOL). Stores TRUE, FALSE, or NULL.

**Syntax**

The basic syntax for the CREATE TABLE command is as follows:

SQL

CREATE TABLE table\_name (

column1\_name datatype [constraints],

column2\_name datatype [constraints],

column3\_name datatype [constraints],

...

);

* **table\_name**: The name of the new table.
* **column\_name**: The name of a column in the table.
* **datatype**: The type of data the column will hold (e.g., INTEGER, VARCHAR).
* **constraints**: Optional rules to enforce data integrity.

**Constraints**

Constraints are rules that enforce the integrity of the data within a table.

* NOT NULL: Ensures that a column cannot have a NULL value.
* UNIQUE: Ensures that all values in a column are different.
* PRIMARY KEY: A combination of NOT NULL and UNIQUE. It uniquely identifies each row in a table. A table can have only one primary key.
* FOREIGN KEY: Enforces a link between two tables. It ensures that a value in a column exists in the primary key of another table.
* CHECK: Ensures that all values in a column satisfy a specific condition.
* DEFAULT: Provides a default value for a column when no value is explicitly specified during an INSERT operation.

**Examples for Freshers 🧑‍🎓**

Here are two straightforward examples to help you understand the concept better.

**Example 1: Creating a simple employees table**

This example creates a table for employee information with a PRIMARY KEY and a NOT NULL constraint.

SQL

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(100) UNIQUE,

hire\_date DATE DEFAULT CURRENT\_DATE

);

* employee\_id is the **primary key**, ensuring each employee has a unique ID.
* first\_name and last\_name are **NOT NULL**, meaning they must contain a value.
* email is **UNIQUE**, preventing two employees from having the same email address.
* hire\_date has a **DEFAULT** constraint, so if a date isn't provided, it will automatically use the current date.

**Example 2: Creating a products table with constraints**

This example demonstrates how to create a table for product inventory with various constraints.

SQL

CREATE TABLE products (

product\_id SERIAL PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

price NUMERIC(10, 2) CHECK (price > 0),

stock\_quantity INT DEFAULT 0

);

* product\_id uses SERIAL, a pseudo-type that automatically generates a unique, sequential integer for the **primary key**. It's a common and convenient practice in PostgreSQL.
* price is defined as a NUMERIC with 10 digits in total and 2 after the decimal point. The CHECK constraint ensures the price is always a positive number.
* stock\_quantity has a **DEFAULT** value of 0, which is useful for new products with no initial stock.