**PostgreSQL: Domains vs. Types**

In PostgreSQL, **domains** and **types** offer ways to define custom data structures beyond the built-in types. Here’s how they can be used:

**-Domains:**

A domain is essentially a user-defined data type based on an existing type, with optional constraints. It helps enforce data integrity while keeping the schema clean

**Example**:

CREATE TABLE customers (

id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

contact phone\_number NOT NULL

);

**-Types:**

PostgreSQL allows defining custom types using CREATE TYPE, which is useful for composite structures or enumerations.

**Example**:

CREATE TYPE address AS (

street VARCHAR(255),

city VARCHAR(255),

state CHAR(2),

zip\_code CHAR(5)

);

**Difference Between Domain and Type in PostgreSQL :**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Domain** | **Type** |
| Definition | A domain is a user-defined data type that is based on an existing data type, with optional constraints. | A type is a classification of data that defines the kind of values that can be stored in a column. |
| Purpose | A type is a classification of data that defines the kind of values that can be stored in a column. | A type is a classification of data that defines the kind of values that can be stored in a column. |
| Constraints | Domains can have constraints (e.g., NOT NULL, CHECK) that apply to all columns using the domain. | Types themselves do not have constraints; constraints are applied at the column level in tables. |
| Reusability | Domains can be reused across multiple tables and columns, ensuring consistency in constraints. | Types can be reused, but they do not inherently carry constraints like domains do. |
| Example | CREATE DOMAIN positive\_integer AS integer CHECK (VALUE > 0);  <br> Usage: column\_name positive\_integer | CREATE TYPE my\_custom\_type AS (field1 integer, field2 text);  <br> Usage: column\_name my\_custom\_type |
| Usage | Can be used in table columns, function parameters, and return types. | Can be used in table columns, function parameters, and return types. |