

# **COURSE NAME: DBMS**

## **COURSE CODE:23AD2102A**

**Topic: DATA TYPES, CONSTRAINTS & DML STATEMENTS**

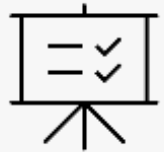
**Session - 9**

## AIM OF THE SESSION



To familiarize students with the basic concept of Data Types in PostgreSQL  
To aware the students how to define the constraints in the queries  
To grasp the students the concept of DML statements

## INSTRUCTIONAL OBJECTIVES



This Session is designed to get the enough knowledge to write the DML queries

## LEARNING OUTCOMES



At the end of this session, you should be able to write the PostgreSQL queries using constraints

**Datatypes** in SQL are used to define the type of data that can be stored in a column of a table, like, INT, CHAR, MONEY, DATETIME etc.

**Defining a Datatype:** Datatypes are defined during the creation of a table in PostgreSQL. While creating a table, it is required to specify its respective datatype and size along with the name of the column.

The syntax to define a datatype on a column of an SQL table is:

```
CREATE TABLE table_name (column1 datatype, column2 datatype...)
```

Ex: `CREATE TABLE Customers (Name VARCHAR (25), Age INT);`

The above query will create a table **Customer**. The query is defined with two datatypes: **VARCHAR** and **INT**

**Note:** PostgreSQL offers the PL/pgSQL procedural programming language. **Additional functionalities** to standard SQL in PostgreSQL include advanced types and user-defined types, extensions and custom modules, JSON support, and additional options for triggers and other functionality.

# BASIC DATA TYPES IN POSTGRESQL

## PostgreSQL Data Types



- 1) Numeric datatype
- 2) Character datatype
- 3) Date/time datatype
- 4) Monetary data type
- 5) Binary data type
- 6) Boolean data type
- 7) Enumerated data type
- 8) Geometric data type
- 9) Text search data type
- 10) UUID data type
- 11) Network address type
- 12) JSON data type
- 13) Bit string type
- 14) XML data type
- 15) Range data type
- 16) Arrays
- 17) Composite data type
- 18) Object identifiers type
- 19) Pseudo data type
- 20) pg-lsn data type

**1. Numeric Datatype:** Numeric datatype is used to specify the numeric data into the table. It contains the following:

name	description	storage size	range
smallint	stores whole numbers, small range.	2 bytes	-32768 to +32767
integer	stores whole numbers.use this when you want to store typical integers.	4 bytes	-2147483648 to +2147483647
bigint	stores whole numbers, large range.	8 bytes	-9223372036854775808 to 9223372036854775807
decimal	user-specified precision, exact	variable	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point.
numeric	user-specified precision, exact	variable	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point.
real	variable-precision, inexact	4 bytes	6 decimal digits precision.
double precision	variable-precision, inexact	8 bytes	15 decimal digits precision
serial	auto incrementing integer	4 bytes	1 to 2147483647
bigserial	large auto incrementing integer	8 bytes	1 to 9223372036854775807

**2. Character Datatype:** The character data types are used to represent the character type values. The below table contains all Character data types that are supported in PostgreSQL:

Datatype	Explanation
char(size)	Here size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
character(size)	Here size is the number of characters to store. Fixed-length strings. Space padded on right to equal size characters.
varchar(size)	Here size is the number of characters to store. Variable-length string.
character varying(size)	Here size is the number of characters to store. Variable-length string.
text	Variable-length string.

**3. Date/Time Datatype:** The character data types are used to represent the character type values. The below table contains all Character data types that are supported in PostgreSQL:

Name	Description	Storage size	Minimum value	Maximum value	Resolution
timestamp [ (p) ] [ without time zone ]	both date and time (no time zone)	8 bytes	4713 bc	294276 ad	1 microsecond / 14 digits
timestamp [ (p) ] with time zone	both date and time, with time zone	8 bytes	4713 bc	294276 ad	1 microsecond / 14 digits
date	date (no time of day)	4 bytes	4713 bc	5874897 ad	1 day
time [ (p) ] [ without time zone ]	time of day (no date)	8 bytes	00:00:00	24:00:00	1 microsecond / 14 digits
time [ (p) ] with time zone	times of day only, with time zone	12 bytes	00:00:00+1459	24:00:00-1459	1 microsecond / 14 digits
interval [ fields ] [ (p) ]	time interval	12 bytes	-178000000 years	178000000 years	1 microsecond / 14 digits



## 4. Monetary Data Type:

Name	Description	Storage size	Range
money	currency amount	8 bytes	-92233720368547758.08 to +92233720368547758.07

## 5. Binary Data Types:

Name	Storage size	Description
bytea	1 or 4 bytes plus the actual binary string	variable-length binary string

## 6. Boolean Data Type:

Name	Description	Storage size
boolean	it specifies the state of true or false.	1 byte

true	false
yes	no
On	off
1	0



**7. Enumerated Data Type:** It is similar to enum types which are compatible with a various programming language. The Enumerated data type is represented in a table with foreign keys to ensure data integrity.

**For Example:**

```
CREATE TYPE fruits_name AS ENUM  
('Mango','Apple','Orange','Strawberry')
```

**8. Geometric Data Types:** Geometric data types represent two-dimensional spatial objects. The most fundamental type, the point, forms the basis for all of the other types.

Name	Storage size	Representation	Description
point	16 bytes	point on a plane	(x,y)
line	32 bytes	infinite line (not fully implemented)	((x1,y1),(x2,y2))
lseg	32 bytes	finite line segment	((x1,y1),(x2,y2))
box	32 bytes	rectangular box	((x1,y1),(x2,y2))
path	16+16n bytes	closed path (similar to polygon)	((x1,y1),...)
path	16+16n bytes	open path	[(x1,y1),...]
polygon	40+16n	polygon (similar to closed path)	((x1,y1),...)
circle	24 bytes	circle	<(x,y),r> (center point and radius)

**9. Text Search Data Type:** In PostgreSQL, the full-text search data type is used to search over a collection of natural language documents. We have two categories of data types that are compatible with full-text search.

Data types	Description
<code>tsvector</code>	It is used to display a document in a form, which enhance text search.
<code>tsquery</code>	It is used to represent a text query.

**10. UUID Data Types:** The UUID stands for Universally Unique identifiers is a 128-bit quantity which is created by an algorithm. It is the best-suited data type for the primary keys. The UUID is written as a group of lower-case hexadecimal digits through multiple sets separated by hyphens.

**For example:**

a0eebc99-9c0b-4ef8-bb6d-6bb9bd380a11

**11. Network Address Data Type:** The UUID stands for Universally Unique identifiers is a 128-bit quantity which is created by an algorithm. It is the best-suited data type for the primary keys. The UUID is written as a group of lower-case hexadecimal digits through multiple sets separated by hyphens. The below table contains all **network address** data types that are supported in PostgreSQL:

Data type	Description	Storage Size
<code>inet</code>	It stores the IPv4 and IPv6 hosts and networks.	7 or 19 bytes
<code>cidr</code>	It is used to store the IPv4 and IPv6 networks.	7 or 19 bytes
<code>macaddr</code>	It stores the MAC addresses.	6 bytes

**12. JSON Data Type:** PostgreSQL provides two kinds of data types for storing the JSON (JavaScript Object Notation) data.

**a) Json** is an extension of a text data type with JSON validation. In this, we can insert the data quickly, but the data retrieval is comparatively slow. It saves inputted data just the way it contains the whitespace. It is also reprocessing on the data retrieval.

**b) Jsonb** is a binary representation of the JSON data. It is also compatible with indexing and also improves the whitespace to make the retrieval quicker. In this, the insertion is slow, but the data retrieval is more rapid, and reprocessing is required on the data retrieval.

**13. Bit string Data Type:** The bit strings data type contains two categories of strings that are 1's and 0's. The bitmasks can be stored with the help of these strings. In this, we have two kinds of SQL bit, such as:

a) bit varying(n) and b) bit (n)

Here, n is a positive integer.

**14. XML Data Type:** In PostgreSQL, the XML data type is used to store the XML data. The function of the XML data type is to check whether that the input XML is well-formed, and also there are support functions to perform type-safe operations on it.

**For Example:**

```
XMLPARSE (DOCUMENT '<?xml version="1.0"?><book><title>PostgreSQL</title><chapter>...</chapter></book>')
```

**15. Range Data Types:** These data types are used to display a range of values of some element types, known as the range's subtype. It also signifies several elements of values in a single range value. In this, we can also create our range types. In PostgreSQL, we have following built-in range types:

Built-in range types	Description
tsrange	Range of timestamp without time zone
tstzrange	Range of timestamp with time zone
daterange	Range of date
int4range	Range of integer
int8range	Range of bigint
numrange	Range of numeric

**16. Array Type:** The PostgreSQL provides a column of tables as a variable-length and the multidimensional array.

We can create any user-defined base type, built-in, composite, and enumerated type arrays.

- ✓ Here, we can perform various operations on arrays such as declare, insert, accessing, modifying, and searching.

**17. Composite Type:** In PostgreSQL, the composite data type is used to signify the structure of a row or record as a list of field names and data types.

**18. Pseudo Type:** In PostgreSQL, the data types are pseudo types, which are used to contain many special-purpose entries. And it is used to declare a result type or the function's argument, but it is not compatible to use as a column data type. The below table contains some of the commonly used pseudo data types in PostgreSQL:

## BASIC DATA TYPES IN POSTGRESQL

Pseudo data types	Description
any	It shows that a function accepts any input data type.
anyelement	It accepts any data type.
anyarray	It shows a function that accepts any array data type
anyenum	It accepts any enum data type
anyrange	It accepts any range of data type
cstring	It is used to specify that a function accepts or returns a null-terminated C string.
language_handler	A procedural language call handler is declared to return language_handler.
fdw_handler	The fdw (foreign-data wrapper) handler is declared to return fdw_handler.
record	It is used to specify a function which is taking or returning an unspecified row type.
trigger	It is declared to return the trigger.
pg_ddl_command	It is used to represent the DDL commands that are available to event triggers.
void	It is used to specify that a function returns no value.



**19. Object Identifier (OIDs) Type:** These types of data types are used as primary keys for several system tables. The oid type represents an object identifier and currently implemented as an unsigned four-byte integer. In huge databases or even in large individual tables, it is not big enough to offer database-wide individuality.

- ✓ Object identifiers are used for references to system tables. Beyond comparison, the oid type itself has few operations that can be cast to integer, and can be manipulated using the standard integer operators.
- ✓ The below table contains all the object identifier data types that are supported in PostgreSQL:

Name	Description	References	Value Example
oid	numeric object identifier	Any	564182
regproc	function name	pg_proc	Sum
regprocedure	function with argument types	pg_proc	sum(int4)
regoper	operator name	pg_operator	+
regoperator	operator with argument types	pg_operator	*(integer,integer) or -(NONE,integer)
regclass	relation name	pg_class	pg_type
regtype	data type name	pg_type	Integer
regnamespace	namespace name	pg_namespace	pg_catalog
regconfig	text search configuration	pg_ts_config	English
regdictionary	text search dictionary	pg_ts_dict	Simple



**20. pg\_lsn Type:** The pg\_lsn data type can be used to store Log Sequence Number (LSN) data, a pointer to a location in the XLOG. It is used to signify the XLogRecPtr and an internal system type of PostgreSQL. The pg\_lsn type is compatible with the standard comparison operators, such as > and =.

## Important Points to Remember:

While using the data types, we can refer to the following points:

- If we have an IEEE 754 data source, we can use the float data type
- For integer data type, we can use int.
- Never use char.
- If we want to limit the input, we can apply a text data type.
- When we have vast numbers, we can use bigint only.

- ❖ The constraints are used to describe the rules for the data columns in a table. If there is any destruction between the constraints and the data action, the action is terminated immediately. The constraints make sure the dependability and the correctness of the data in the database.
- ❖ The constraints can be further divided as column level or table level where the table level constraints are used for the whole table, and the Column level constraints are used only for one column.

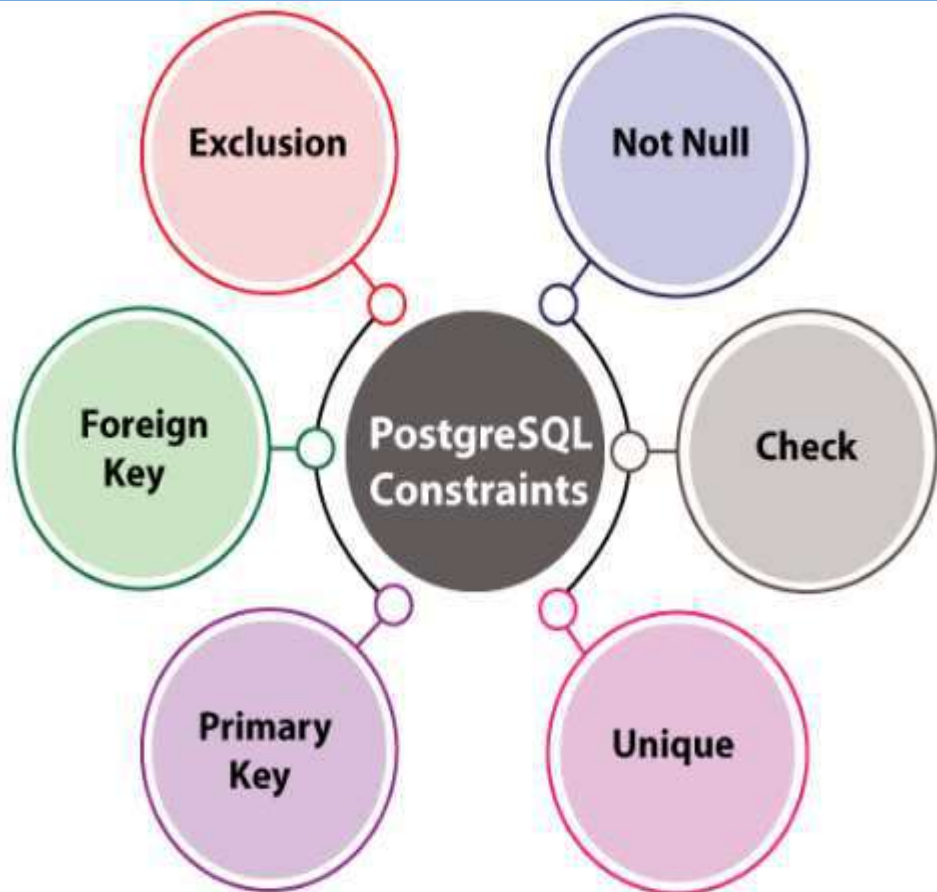
## Where we use the constraints?

Once constraint is created, it can be added to a table, and we can also disable it temporarily. The constraints are most commonly used in below areas:

1. To individual columns, we can use the column constraints.
2. When we are creating a table with the help of creating table command, we can also declare the constraints
3. SQL can discard any value which interrupts the well-defined criteria.
4. All the information related to the constraints is kept in the data dictionary.
5. For one or more columns, we can use the table constraints.
6. All the constraint is allocated a name.

# CONSTRAINTS

## Most commonly used constraints in PostgreSQL

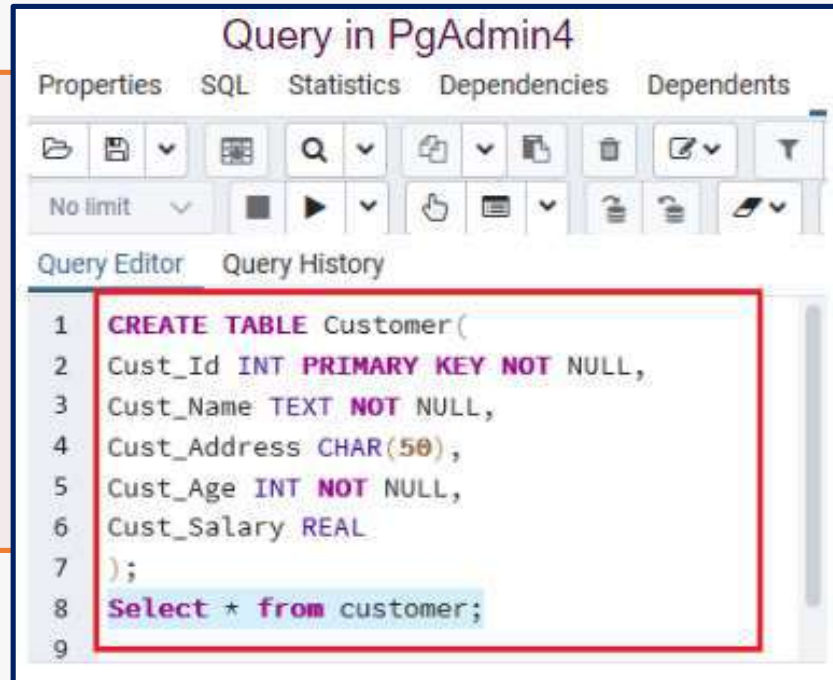


Constraints	Description
<b>Not Null</b>	This type of constraint is used to make sure that a column cannot have a <b>null</b> value. And no name can be further defined to generate a not-null constraint.
<b>Check</b>	It is used to make sure that all values in a column or a field of a table satisfy particular situations such as it must match a Boolean expression. The Check constraint can be defined by a separate name.
<b>Unique</b>	The unique constraints are used to make sure that all values in a column of a table are exclusive.
<b>Primary key</b>	The primary key is to specify each row or record in a database table uniquely and make sure that there is no record duplication in a particular table.
<b>Foreign key</b>	In PostgreSQL, the foreign key is used to define that the value in a column or field of a table is equal to the real value of the primary key of another table.
<b>Exclusion</b>	This constraint is used to make sure that any two rows are linked on the precise columns or statements with the help of defined operators, and one of these operator assessments should return either <b>Null or False</b> .

**1) NOT NULL Constraint:** In not-null constraint, a column can hold the Null values by default. If we don't want a column to have a NULL value, then we need to explain such constraint on this column state that NULL is now not acceptable for that particular column. It is always created as a column constraint, and it represents unknown data but it doesn't mean that the data would be null.

Example: To create a new table called Customer, which has the five columns, such as Cust\_Id, Cust\_ Name, Cust\_Address, Cust\_Age, and Cust\_Salary.

```
CREATE TABLE Customer(  
Cust_Id INT PRIMARY KEY NOT NULL,  
Cust_Name TEXT NOT NULL,  
Cust_Address CHAR(50),  
Cust_Age INT NOT NULL,  
Cust_Salary REAL  
);
```



**Output**

Data Output	Explain	Messages	Notifications	
<div><div>▲</div><div>cust_id</div><div>[PK] integer</div></div>	<div><div></div><div>cust_name</div><div>text</div></div>	<div><div></div><div>cust_address</div><div>character (50)</div></div>	<div><div></div><div>cust_age</div><div>integer</div></div>	<div><div></div><div>cust_salary</div><div>real</div></div>

**2) CHECK Constraint:** In PostgreSQL, the Check constraint can be defined by a separate name. It is used to control the value of columns being inserted. It allows us to verify a condition that the value being stored into a record. If the statement is false, then the data disrupts the constraint and is not saved into the table.

✓ To create a new table called Customer2, and this table contains five columns.

```
CREATE TABLE Customer2(  
Cust_Id INT PRIMARY KEY NOT NULL,  
Cust_Name TEXT NOT NULL,  
Cust_Address CHAR(30),  
Cust_Age INT NOT NULL,  
Cust_Salary REAL check (Cust_Salary>0)  
);
```

## Query in PgAdmin4

Query Editor   Query History

```
8 CREATE TABLE Customer2(  
9 Cust_Id INT PRIMARY KEY NOT NULL,  
10 Cust_Name TEXT NOT NULL,  
11 Cust_Address CHAR(30),  
12 Cust_Age INT NOT NULL,  
13 Cust_Salary REAL Check (Cust_Salary>0)  
14 );
```

## Output

18 **Select** \* **from** Customer2;  
19

Data Output   Explain   Messages   Notifications

cust_id	cust_name	cust_address	cust_age	cust_salary
[PK] integer	text	character (30)	integer	real



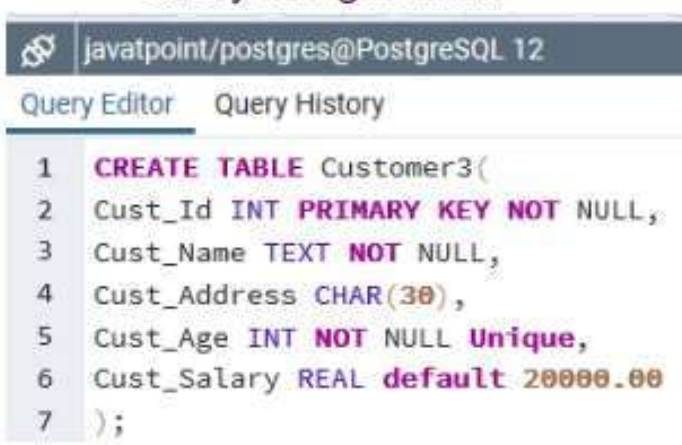
**3) UNIQUE Constraint:** The unique constraint is used to maintain the individuality of the values that we store into a field or a column of the table. It is compatible with a group of column constraints or column constraints and a table constraint.

When we are using the unique constraint, an index on one or more columns generate automatically. If we add two different null values into a column in different rows, but it does not interrupt the specification of the UNIQUE constraint.

✓ To create a new table called Customer3, which has similar five columns as we created in the above tables.

```
CREATE TABLE Customer3(  
Cust_Id INT PRIMARY KEY NOT NULL,  
Cust_Name TEXT NOT NULL,  
Cust_Address CHAR(30),  
Cust_Age INT NOT NULL Unique,  
Cust_Salary REAL default 20000.00  
);
```

## Query in PgAdmin4



```
1 CREATE TABLE Customer3(  
2 Cust_Id INT PRIMARY KEY NOT NULL,  
3 Cust_Name TEXT NOT NULL,  
4 Cust_Address CHAR(30),  
5 Cust_Age INT NOT NULL Unique,  
6 Cust_Salary REAL default 20000.00  
7 );
```

## Output



9	select * from Customer3;				
Data Output Explain Messages Notifications					
	cust_id [PK] integer	cust_name text	cust_address character (30)	cust_age integer	cust_salary real

**4) PRIMARY KEY Constraint:** It is a field in a table that individually identifies each row or the record in a database table, and it contains a unique value. A primary key does not hold any null value. And for this, we can also say that the primary key is a collection of the unique and not-null constraint of a table.

The working of the primary key is similar to a unique constraint. Still, the significant difference between them is one table can have only one primary key; however, the table can have one or more unique and not-null constraints.

- ✓ To create a new table called Employee, which contains the four columns, such as Emp\_Id, Emp\_ Name, Emp\_Address, and Emp\_Age.

```
CREATE TABLE Employee(  
  Emp_Id INT PRIMARY KEY NOT NULL,  
  Emp_Name TEXT NOT NULL,  
  Emp_Address CHAR(30),  
  Emp_Age INT NOT NULL  
);
```

Query in PgAdmin4

javatpoint/postgres@PostgreSQL 12

Query Editor Query History

```
1 CREATE TABLE Employee(  
2   Emp_Id INT PRIMARY KEY NOT NULL,  
3   Emp_Name TEXT NOT NULL,  
4   Emp_Address CHAR(30),  
5   Emp_Age INT NOT NULL  
6 );
```

### Output

8 `Select * from Employee;`

Data Output	Explain	Messages	Notifications								
<table><tr><td>emp_id</td><td>emp_name</td><td>emp_address</td><td>emp_age</td></tr><tr><td>[PK] integer</td><td>text</td><td>character (30)</td><td>integer</td></tr></table>	emp_id	emp_name	emp_address	emp_age	[PK] integer	text	character (30)	integer			
emp_id	emp_name	emp_address	emp_age								
[PK] integer	text	character (30)	integer								



**5) FOREIGN KEY Constraint:** It is a group of columns with values that depend on the primary key benefits from another table. It is used to have the value in a column or group of columns that must be displayed in the same column or combination of columns in another table.

In PostgreSQL, the foreign key's values are parallel to actual values of the primary key in the other table; that's why it is also known as Referential integrity Constraint.

- ✓ To create a new table called Employee1, which contains the four columns that are similar to the previous table.

```
CREATE TABLE Employee1(  
  Emp_Id INT PRIMARY KEY NOT NULL,  
  Emp_Name TEXT NOT NULL,  
  Emp_Address CHAR(30),  
  Emp_Age INT NOT NULL  
);
```

Query in PgAdmin4

Query Editor   Query History

```
1 CREATE TABLE Employee1(  
2   Emp_Id INT PRIMARY KEY NOT NULL,  
3   Emp_Name TEXT NOT NULL,  
4   Emp_Address CHAR(30),  
5   Emp_Age INT NOT NULL  
6 );  
7  
8 CREATE TABLE cust(  
9   Emp_Id INT PRIMARY KEY NOT NULL,  
10  Cust CHAR(50) NOT NULL,  
11  Cust_Id INT references Employee1(Emp_Id)  
12 );
```

## Output

13 **Select** \* **from** cust;

Data Output	Explain	Messages	Notifications
<b>emp_id</b> [PK] integer	<b>cust</b> character (50)	<b>cust_id</b> integer	

**6) EXCLUSION Constraint:** It is used to make sure that any two rows are linked on the specified columns or statements using the defined operators. In any case, one of these operator evaluations will return null or false.

- ✓ To create a new table called Employee, which contains the five columns. And here, we will use an exclude constraint as well.

## ERROR

### Query in PgAdmin4

```
javatpoint/postgres@PostgreSQL 12
Query Editor  Query History
1  CREATE TABLE Employee3(
2  Emp_Id INT PRIMARY KEY NOT NULL,
3  Emp_Name TEXT,
4  Emp_Address CHAR(50),
5  Emp_Age INT,
6  Emp_SALARY REAL,
7  EXCLUDE USING gist (Emp_Name WITH =,Emp_Age WITH <>)
8  );
9  CREATE EXTENSION btree_gist;
10
11 INSERT INTO Employee3 VALUES(101, 'john', 'Newyork',22, 30000.00 );
12 INSERT INTO Employee3 VALUES(102, 'john', 'Florida',22, 30000.00 );
13 INSERT INTO Employee3 VALUES(103, 'john', 'Newyork', 32, 30000.00 );
14
15 Select * from Employee3;
```

```
INSERT INTO Employee3 VALUES(103, 'john', 'New
```

Data Output Explain Messages Notifications

```
ERROR: conflicting key value violates exclusion constraint
"employee3_emp_name_emp_age_excl"
DETAIL: Key (emp_name, emp_age)=(john, 32) conflicts with existing key
(emp_name, emp_age)=(john, 22).
SQL state: 23P01
```

'gist' is the index, and used for creating and implementation of 'exclusion'

'btree\_gist' is a command, for one time in a database. And after that, it will connect the btree\_gist extension that defines the constraints on basic scalar data types.

If we are using the 'exclusion' constraints, we have to run the create extension

## DROPPING CONSTRAINTS

**6) DROPPING Constraints:** If we want to delete a constraint, then we should remember the name of the constraints as it is easier for us to drop the constraints directly by its name. Otherwise, we will require to identify the system-generated name. The following command can be used to find out the names.

```
\d table name
```

The syntax for dropping constraints is as follows:

```
ALTER TABLE table_name DROP CONSTRAINT some_name;
```

**DML** stands for 'Data Manipulation Language' and is typically used to add, retrieve or update data. The commands used for DML in PostgreSQL are INSERT, UPDATE and DELETE.

**1) INSERT:** The INSERT command is used to insert new rows into a table. We can insert a single row or multiple row values at a time into the particular table.

**Syntax:**

```
INSERT INTO TABLE_NAME (column1, column2, ..... columnN) VALUES (value1, value2, ..... valueN);
```

**For example:**

```
INSERT INTO cseblog (Author, Subject) VALUES ("kbr", "DBMS");
```

**2) UPDATE:** The UPDATE command is used to change the present records in a table. To update the selected rows, we have to use the WHERE clause; otherwise, all rows would be updated.

**Syntax:**

```
UPDATE table_name SET column1 = value1, column2 = value2...., columnN = valueN WHERE condition;
```

**For example:**

```
UPDATE students SET User_Name = 'kbr' WHERE Student_Id = '3'
```

**3) DELETE:** The DELETE command is used to delete all existing records from a table and the WHERE clause is used to remove the selected records or else, all the data would be eliminated.

**Syntax:**

```
DELETE FROM table_name WHERE [condition];
```

**For example:**

```
DELETE FROM cseblog WHERE Author="kbr";
```

## ACTIVITIES/ CASE STUDIES/ IMPORTANT FACTS RELATED TO THE SESSION

- 1) Create a table an employee and execute DML operations such as insert, delete and update
- 2) Create a table student and apply the constraints at column level and table level. Use constraints such as not null, unique, primary key, foreign key, check, default and create index.



**1. NOT NULL Constraint:** The NOT NULL constraint in a column means that the column cannot store NULL values. For example,

```
CREATE TABLE Colleges (  
    college_id INT NOT NULL,  
    college_code VARCHAR(20) NOT NULL,  
    college_name VARCHAR(50)  
);
```

Here, the college\_id and the college\_code columns of the Colleges table won't allow NULL values.

**2. UNIQUE Constraint:** The UNIQUE constraint in a column means that the column must have unique value. For example,

```
CREATE TABLE Colleges (  
    college_id INT NOT NULL UNIQUE,  
    college_code VARCHAR(20) UNIQUE,  
    college_name VARCHAR(50)  
);
```

Here, the value of the college\_code column must be unique. Similarly, the value of college\_id must be unique as well as it cannot store NULL values.



- ❖ A data type, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.
- ❖ SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.
- ❖ DML statements allow you to query, edit, add, and remove data stored in database objects. The primary DML commands are SELECT , INSERT , DELETE , and UPDATE . Using DML statements, you can perform powerful actions on the actual data stored in your system.

## SELF-ASSESSMENT QUESTIONS

1. In the database table, data types describe the kind of \_\_\_\_ that it can contain.

- (a) Table
- (b) Data
- (c) Number
- (d) None of the above

## SELF-ASSESSMENT QUESTIONS

1. In the database table, data types describe the kind of \_\_\_\_ that it can contain.

- (a) Table
- (b) Data
- (c) Number
- (d) None of the above

**Answer: B) Data**

## SELF-ASSESSMENT QUESTIONS

1. In the database table, data types describe the kind of \_\_\_\_ that it can contain.

- (a) Table
- (b) Data
- (c) Number
- (d) None of the above

**Answer: B) Data**

2. How many categories of PostgreSQL data types?

- (a) 18
- (b) 17
- (c) 20
- (d) 25

## SELF-ASSESSMENT QUESTIONS

1. In the database table, data types describe the kind of \_\_\_\_ that it can contain.

- (a) Table
- (b) Data
- (c) Number
- (d) None of the above

**Answer: B) Data**

2. How many categories of PostgreSQL data types?

- (a) 18
- (b) 17
- (c) 20
- (d) 25

**Answer: C) 20**

- 1. Describe basic datatypes in PostgreSQL.**
- 2. List out DML commands in PostgreSQL.**
- 3. Analyze constraints with suitable examples.**

### Reference Books:

1. Database System Concepts, Sixth Edition, Abraham Silberschatz, Yale University Henry, F. Korth Lehigh University, S. Sudarshan Indian Institute of Technology, Bombay.
2. Fundamentals of Database Systems, 7<sup>th</sup> Edition, RamezElmasri, University of Texas at Arlington, Shamkant B. Navathe, University of Texasat Arlington.
3. An Introduction to Database Systems by Bipin C. Desai

### Sites and Web links:

1. <https://www.javatpoint.com/postgresql-tutorial>
2. <https://www.tutorialspoint.com/sql/index.htm>
3. [https://www.programiz.com/sql /](https://www.programiz.com/sql/)



THANK YOU



Team – Database Management System