

# PostgreSQL Tutorial For Beginners – All You Need To Know About PostgreSQL

Published on Jul 12, 2019 4.5K Views



Sahiti Kappagantula

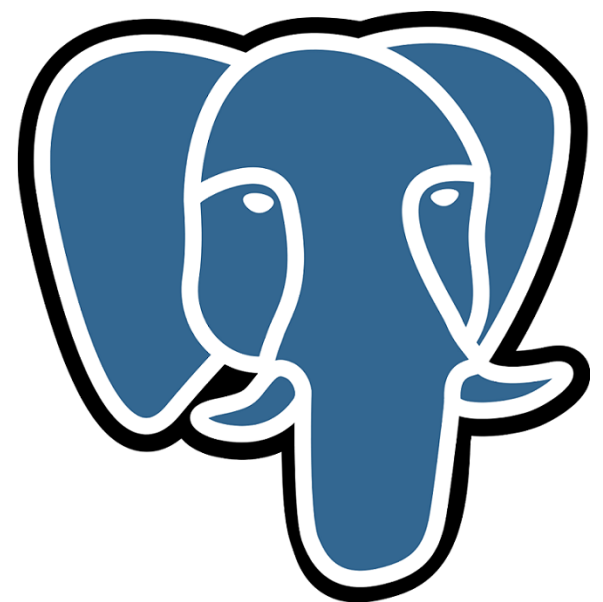
PostgreSQL is an open-source object-relational database system with 30+ years of active development in the industry. In this article on PostgreSQL Tutorial For Beginners, I will introduce you to the different concepts of databases and the commands used in PostgreSQL.

The topics covered in this article are mainly divided into 4 categories: DDL, DML, DCL & TCL.

- The **DDL** (Data Definition Language) commands are used to define the database. Example: CREATE, DROP, ALTER, TRUNCATE, COMMENT, RENAME.
- The **DML** (Data Manipulation Language) commands deal with the manipulation of data present in the database. Example: SELECT, INSERT, UPDATE, DELETE.
- The **DCL** (Data Control Language) commands deal with the permissions, rights and other controls of the database system. Example: GRANT, INVOKE.
- The **TCL** (Transaction Control Language) commands deal with the transaction of the database. Example: BEGIN, COMMIT, ROLLBACK.

Apart from the commands, the following topics will be covered in this article:

- [What is PostgreSQL?](#)
- [Install PostgreSQL on Windows](#)
- [Different Types Of Keys In Database](#)
- [Constraints Used In Database](#)
- [Operators](#)
- [Aggregate Functions](#)
- [Set Operations](#)
- [Nested Queries](#)
- [Joins](#)
- [Views](#)
- [Stored Procedures](#)
- [Triggers](#)
- [UUID Data Type](#)



## What is PostgreSQL? – PostgreSQL Tutorial

PostgreSQL is an object-relational database system which extends and uses the SQL language. It originates in the year 1986 and has been into active development for more than 30 years.

*The features of PostgreSQL are as follows:*

1. **Data Types:** PostgreSQL supports various types of data types such as primitive, structured, document, geometry and customizations. This helps the user to store data in any format.
2. **Data Integrity:** With the help of various constraints and keys in the database, PostgreSQL makes sure that data integrity is satisfied for simple to complex databases.
3. **Performance:** PostgreSQL provides features such as indexing, multi-version concurrency control, JIT compilation of expressions to make sure the concurrency and the performance are kept up to mark.
4. **Reliability:** With the help of Write Ahead Logging(WAL) and Replication, PostgreSQL has proven itself to be one of the most reliable database systems over a period of time.
5. **Security:** PostgreSQL provides powerful mechanisms such as authentication, a robust access-control system to ensure that only authorized users have access to the databases.
6. **Extensibility:** PostgreSQL comes with various extensions to provide additional functionalities. It also has scaled its extensibility features with stored functions, procedural language, and foreign data wrappers.

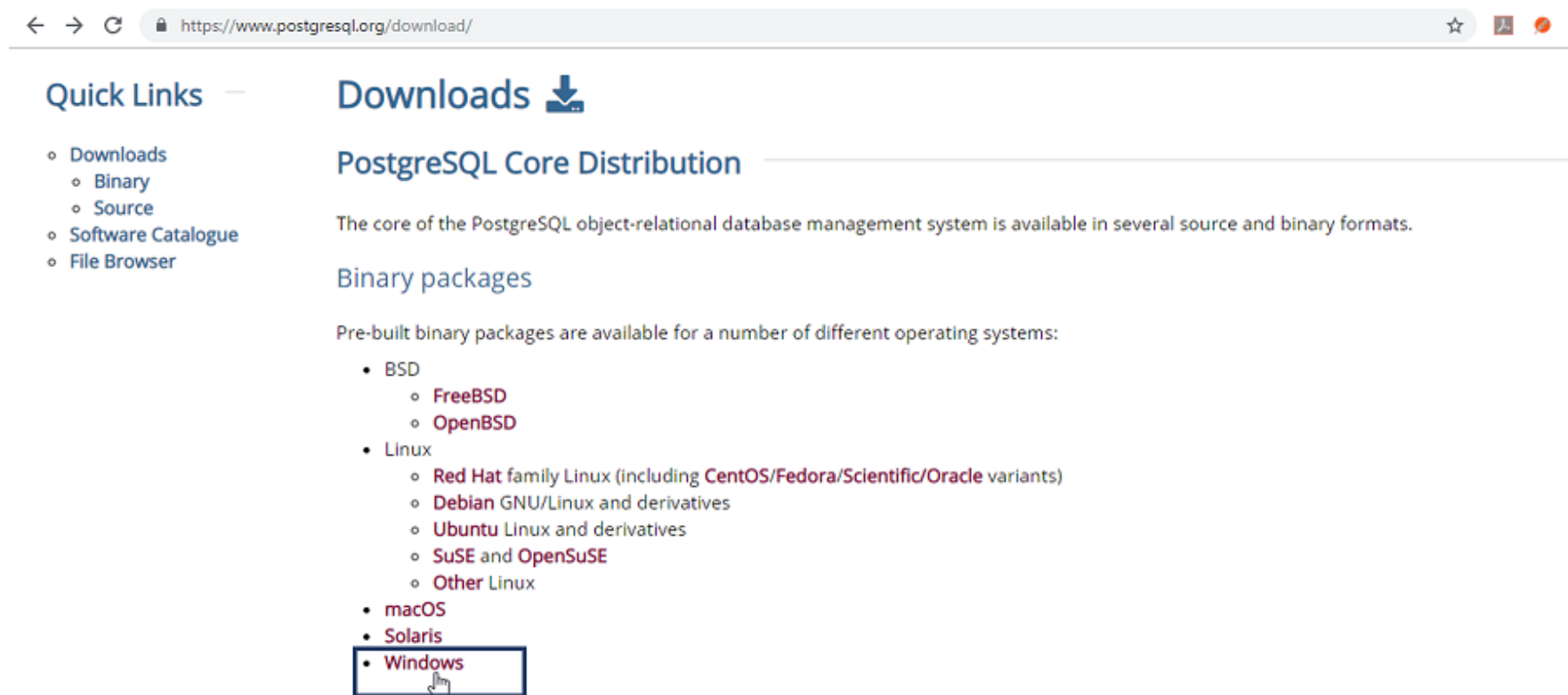
Now, that you know what is PostgreSQL, let us get started by installing PostgreSQL on Windows.

## Install PostgreSQL on Windows – PostgreSQL Tutorial

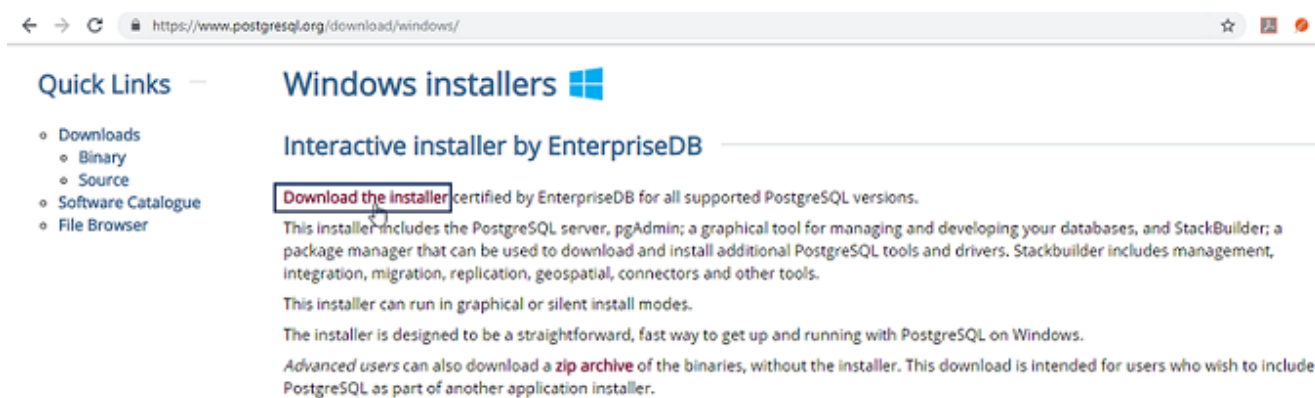
To install PostgreSQL on Windows, you have to follow the below steps:



**Step 1:** Go to the [official website of PostgreSQL](https://www.postgresql.org/download/) and then choose the operating system for which you wish to download. Here I will choose Windows.

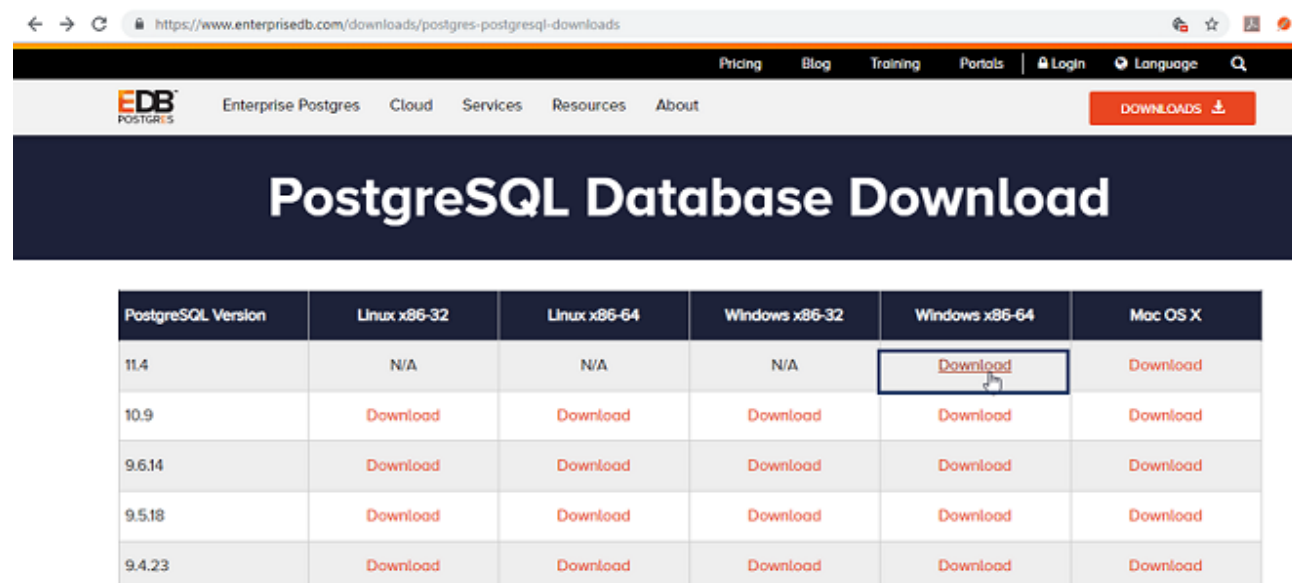


**Step 2:** Once, the operating system is chosen, you will be redirected to a page, where you have to download the installer. To do that click on the option: **Download the installer**. Refer below.



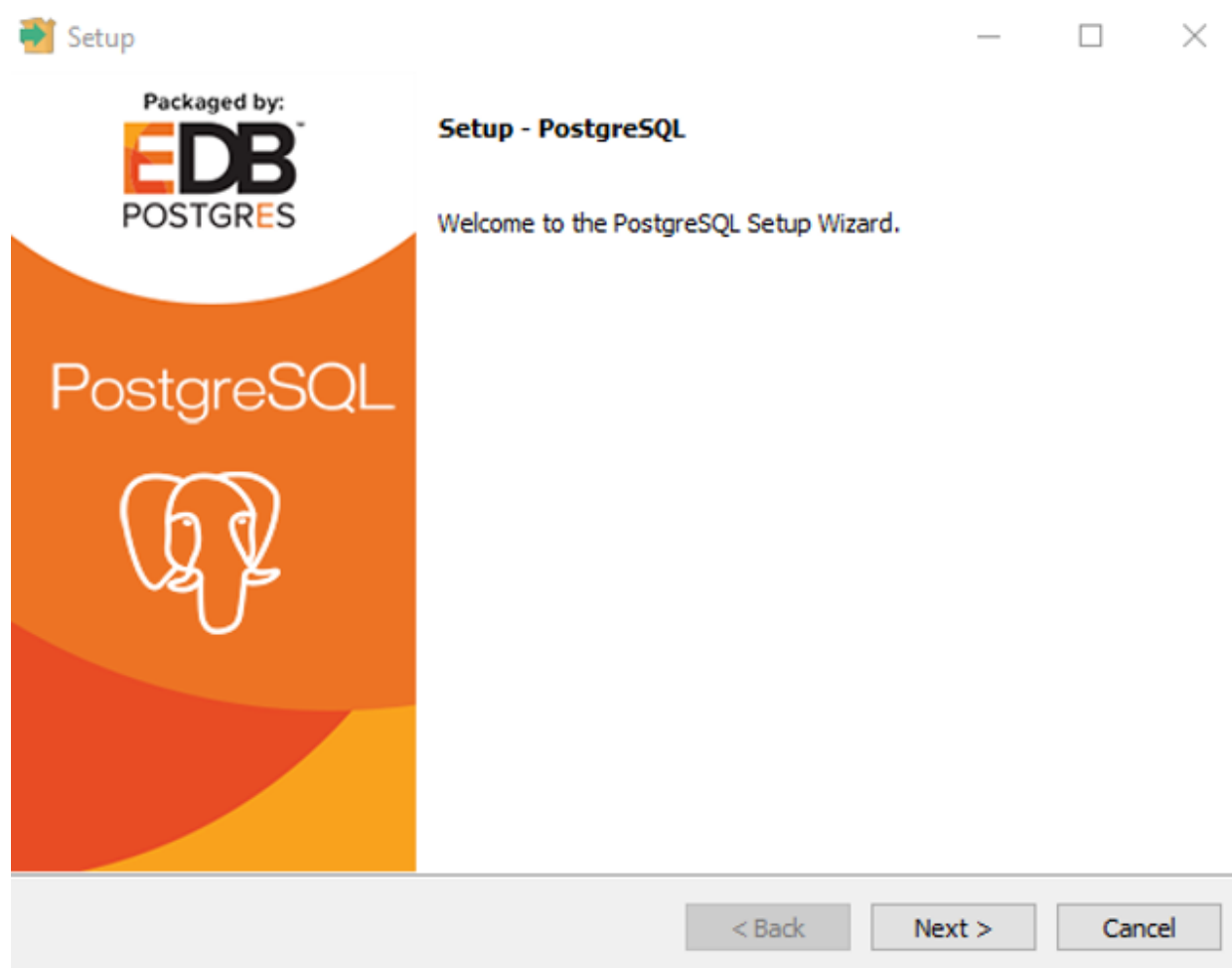
**Step 3:** Then, you will be further redirected to a page, where you have to **choose the installer version based on the Operating System**. Here, i will choose 11.4 version for Windows 64 bit. Refer below.

Once, you **hit on Download**, you will automatically see that PostgreSQL is getting downloaded.

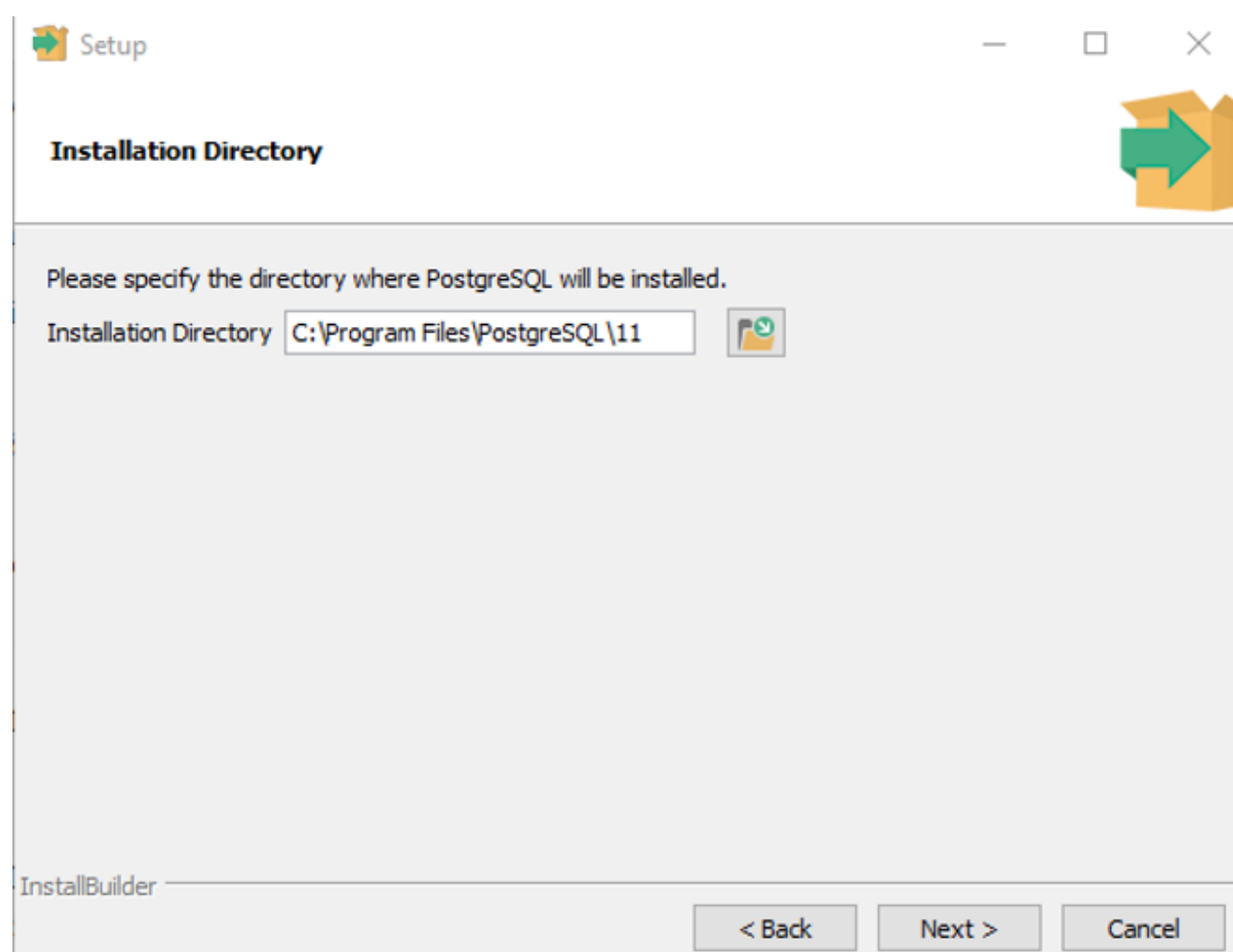


**Step 4:** Now, once the file is downloaded, double click on the file to open it and a wizard will appear on your screen as below. Click on **Next** and proceed further.



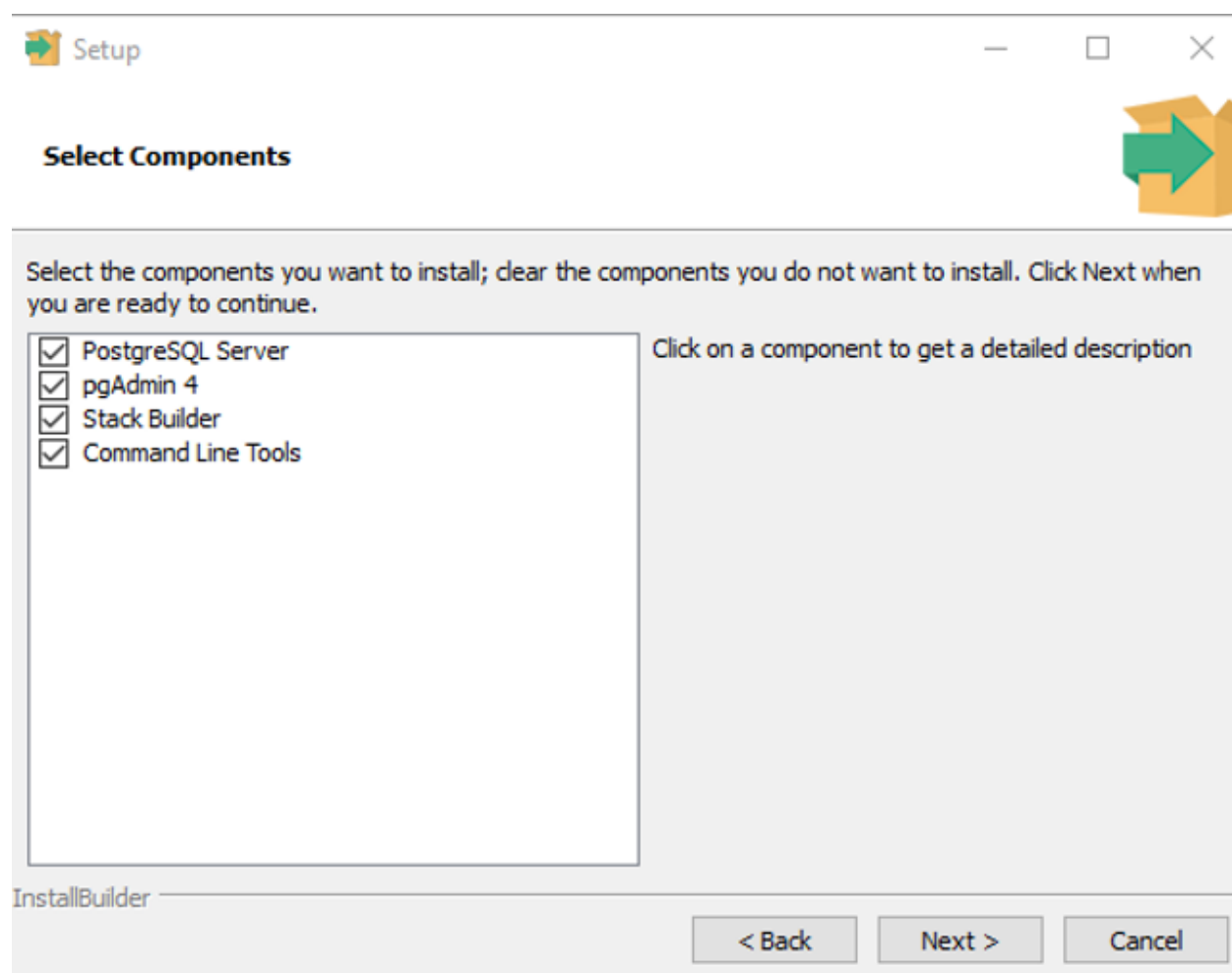


**Step 4.1:** Now, **specify the Installation Directory**. Here, I will leave it as it is, and click on **Next** as below.

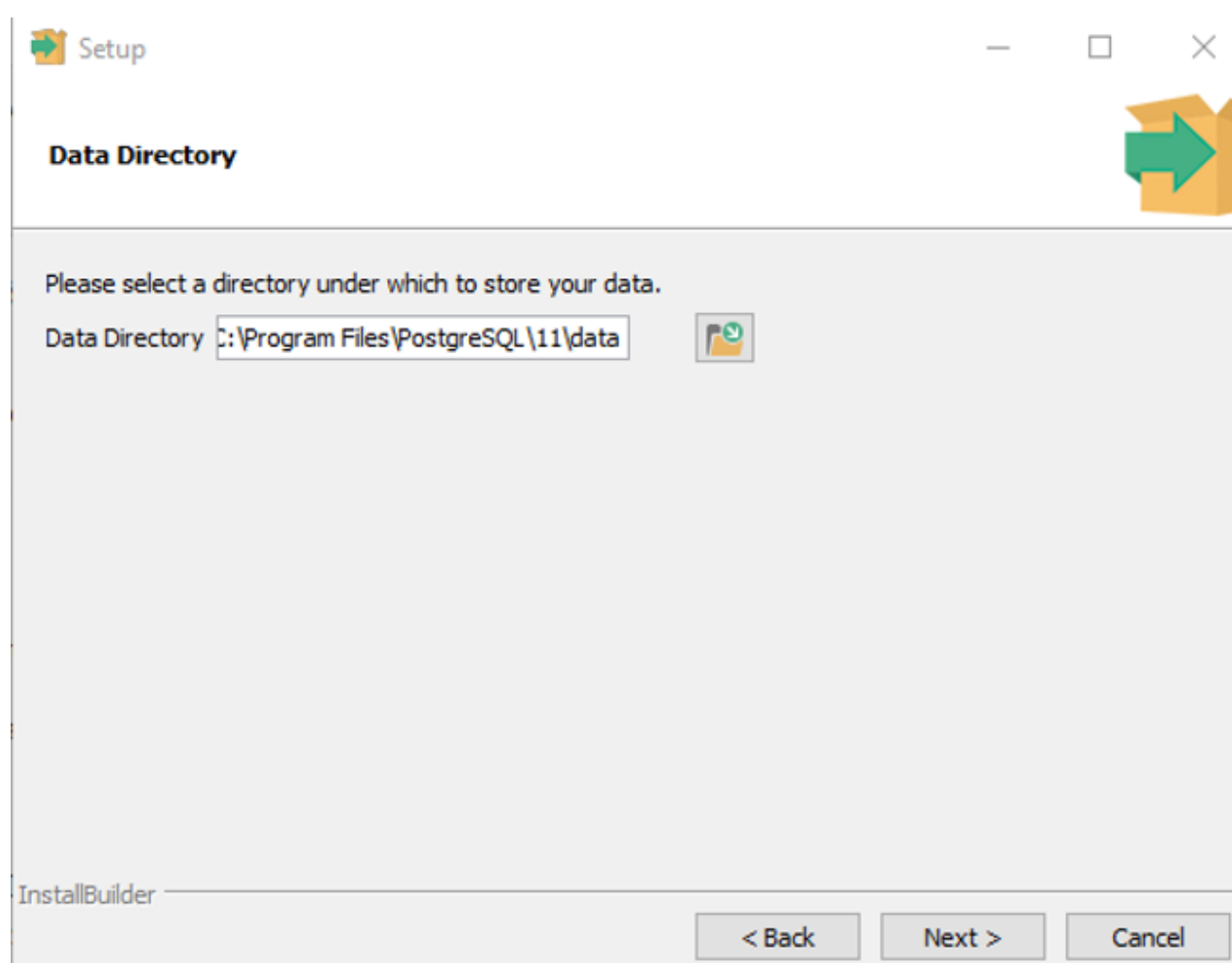


**Step 4.2:** Now, **choose the components which you wish to install** and then click on **Next**. Here, I am selecting all the components.





**Step 4.3:** Next, **select the directory where you want to store data**. Here I am going to leave it as it is. Then, click on **Next**.



**Step 4.4:** In the next dialog box, which comes, you have to **mention the password for the super user**. Then, click on **Next**.



Setup

**Password**

Please provide a password for the database superuser (postgres).

Password

Retype password

InstallBuilder

< Back Next > Cancel

**Step 4.5:** Next, you have to **select the port number** on which server should listen. Here, I will let it be as it is and then click on **Next**.

Setup

**Port**

Please select the port number the server should listen on.

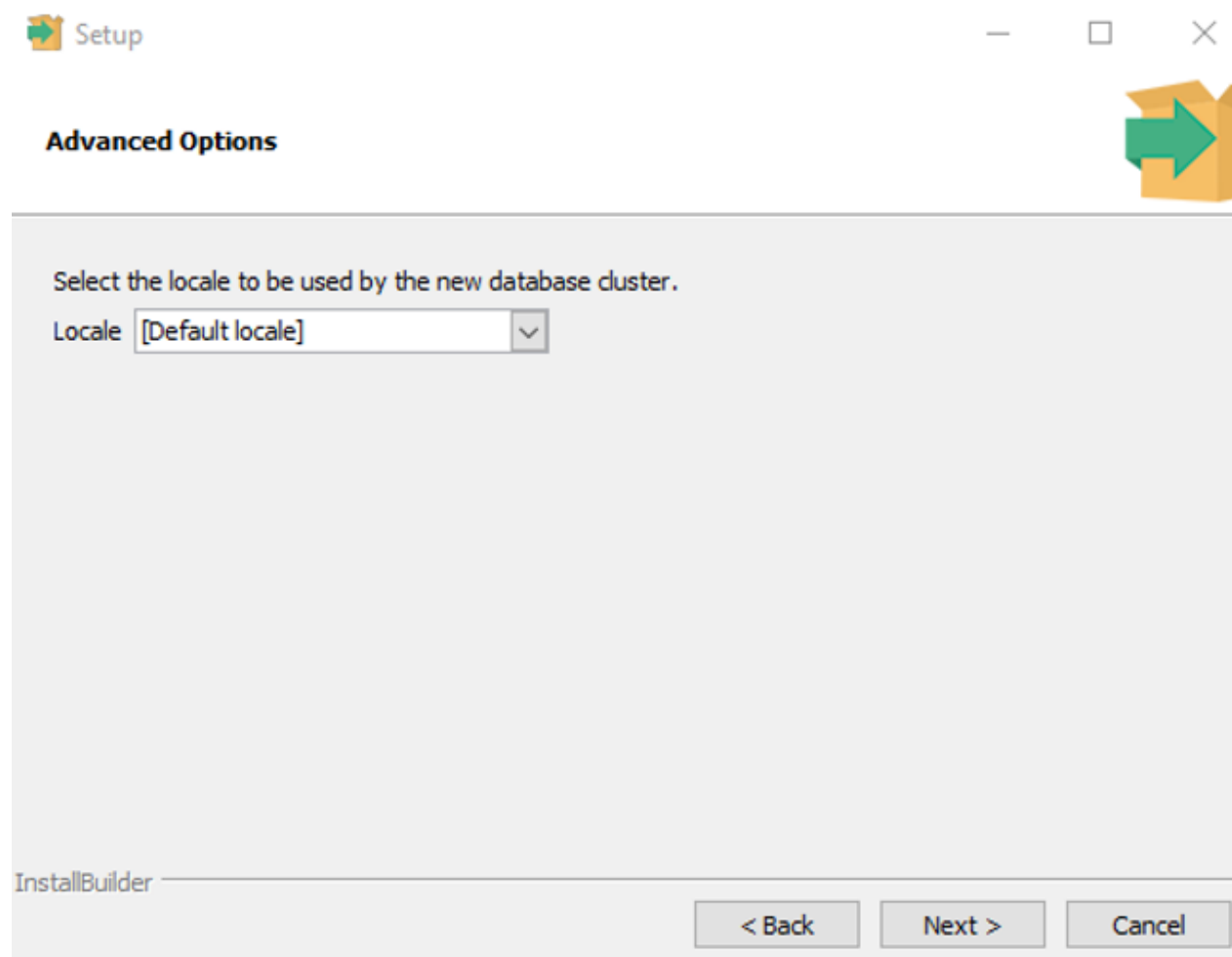
Port

InstallBuilder

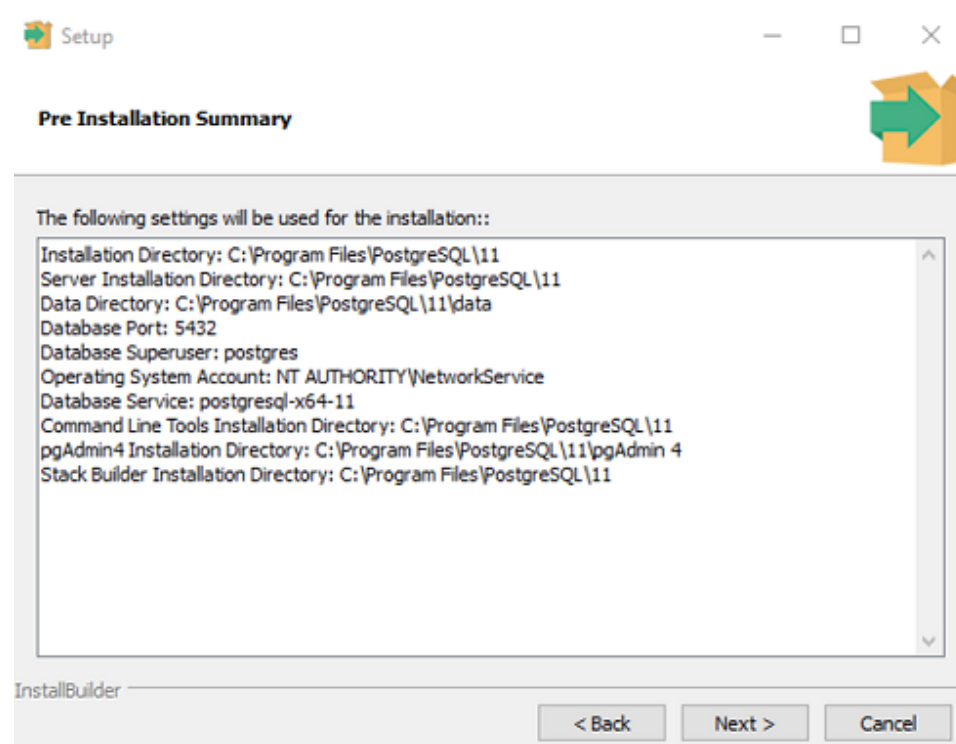
< Back Next > Cancel

**Step 4.6:** Finally, **select the locale** to be used by the new database cluster. I will let it be as it is and then click on **Next**.

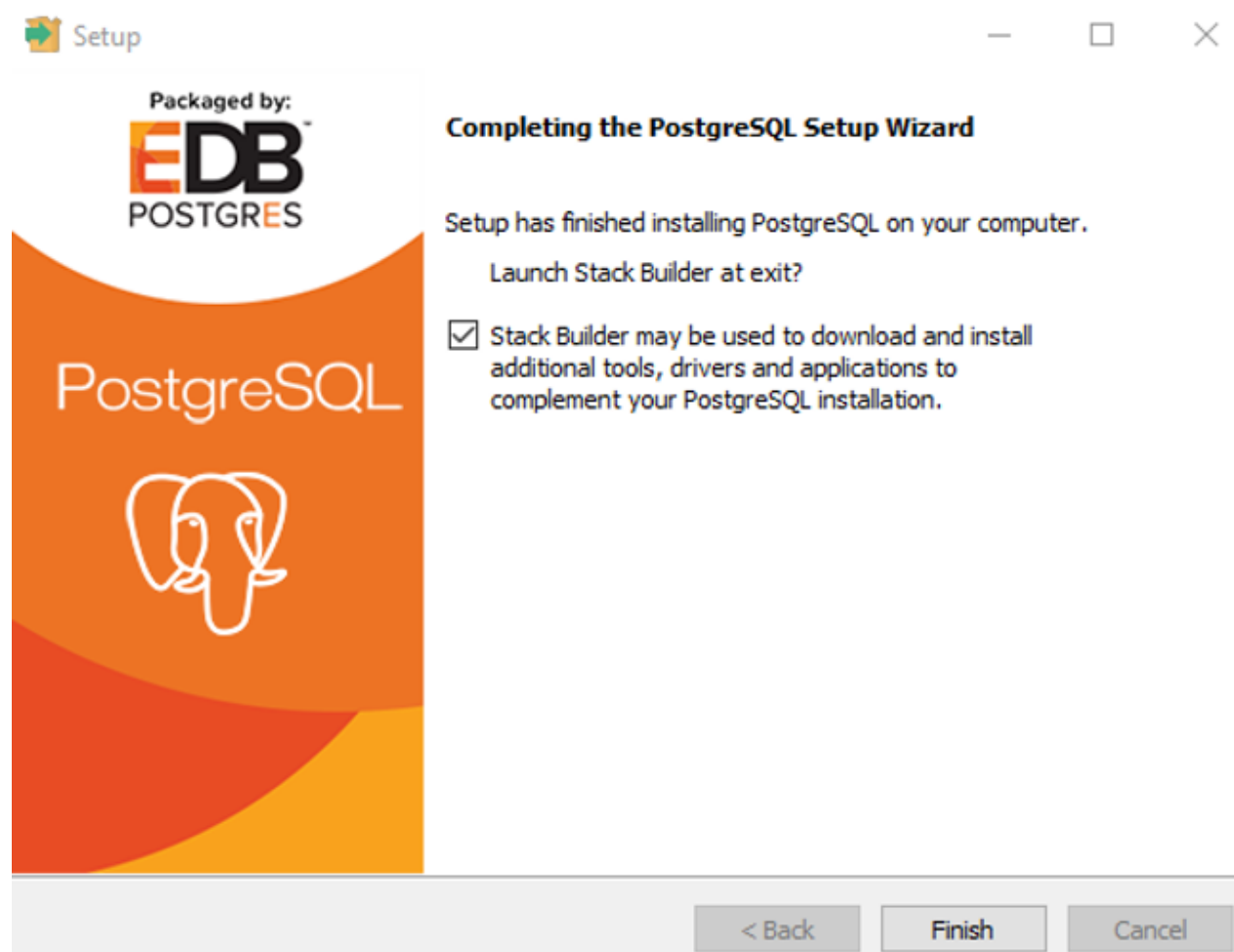




**Step 4.7:** Finally click on **Next** in the wizards which come to begin the installation of PostgreSQL on your computer.



Once, the installation is complete, you will see a dialog box as below on your screen. Click on **Finish**.



**Step 5:** Now, you have to **connect the server to a database**. To do that open pgadmin which is the **official GUI of PostgreSQL**. Once you open pgadmin, you will see a dialog box, which asks you for the password. So, mention the password, and click on **OK**.

Unlock Saved Passwords

Please enter your master password.  
This is required to unlock saved passwords and reconnect to the database server(s).

Password

.....

?

Reset Master Password

✕ Cancel

✓ OK

Now, that you must have installed PostgreSQL, let us get started with commands used in PostgreSQL.

In this article on PostgreSQL Tutorial For Beginners, I am going to consider the below database as an example, to show you how to write commands.

TeacherID	TeacherName	Address	City	PostalCode	Country	Salary
01	Saurav	Gangnam Street	Seoul	06499	South Korea	42000
02	Preeti	Queens Quay	Rio Claro	560001	Brazil	45900
03	Vinod	Kings Road	London	SW6	United Kingdom	65000
04	Akanksha	Mayo Road	Kolkata	700069	India	23000
05	Amit	MG Road	Bengaluru	560001	India	30000

So, let’s get started now!

Data Definition (DDL) Commands – PostgreSQL Tutorial

This section of the article consists of those commands, which you can define your database. The commands are:

- [CREATE](#)
- [ALTER](#)
- [DROP](#)
- [TRUNCATE](#)
- [RENAME](#)

CREATE

This statement is used to either create a schema, tables or an index.

The ‘CREATE SCHEMA’ Statement

The CREATE SCHEMA statement is used to create a database or most commonly known as a schema.

Syntax:

```
CREATE SCHEMA Schema_Name;
```

Example:

```
1 CREATE SCHEMA teachers;
```

The ‘CREATE TABLE’ Statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax:

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
);
```

Example:





```
1 CREATE TABLE TeachersInfo
2 (
3 TeacherID int,
4 TeacherName varchar(255),
5 Address varchar(255),
6 City varchar(255),
7 PostalCode int,
8 Country varchar(255),
9 Salary int
10 );
```

## ALTER

This statement is used to add, modify or delete constraints or columns.

### The 'ALTER TABLE' Statement

The ALTER TABLE statement is used to either add, modify or delete constraints and columns from a table.

#### Syntax:

```
ALTER TABLE table_name
ADD column_name datatype;
```

#### Example:


```
1 ALTER TABLE TeachersInfo
2 ADD DateOfBirth date;
```

## DROP

This command is used to delete the database, tables or columns.

### The 'DROP SCHEMA' Statement

The DROP SCHEMA statement is used to drop the complete schema.



### SQL Essentials Training & Certification

- Course Duration
- Real-life Case Studies
- Assignments
- Lifetime Access

[Explore Curriculum](#)

#### Syntax:

```
DROP SCHEMA schema_name;
```

#### Example:

```
1 DROP SCHEMA teachers;
```

### The 'DROP TABLE' Statement

The DROP TABLE statement is used to drop the entire table with all its values.

#### Syntax:

```
DROP TABLE table_name;
```

#### Example:

```
1 DROP TABLE TeachersInfo;
```

## TRUNCATE

The TRUNCATE statement is used to delete the data which is present inside a table, but the table doesn't get deleted.

#### Syntax:





```
TRUNCATE TABLE table_name;
```

#### Example:

```
1 | TRUNCATE TABLE TeachersInfo;
```

## RENAME

The RENAME statement is used to rename one or more tables or columns.

#### Syntax:

```
ALTER TABLE table_name RENAME TO new_table_name;  --Rename Table name
```

```
ALTER TABLE table_name RENAME COLUMN column_name TO new_column_name;  -- Rename Column name
```

#### Example:

```
1 | ALTER TABLE TeachersInfo RENAME TO InfoTeachers;  
2 |  
3 | ALTER TABLE InfoTeachers RENAME COLUMN dateofbirth TO dob;
```

Now, before I move further in this article on PostgreSQL Tutorial For Beginners, let me tell you the various types of Keys and Constraints that you need to mention while manipulating the databases. The keys and constraints will help you create tables in a much better way, as you can relate each table with the other table.

## Different Types Of Keys In Database – PostgreSQL Tutorial

There are mainly 5 types of Keys, that can be mentioned in the database.

- **Candidate Key** – A Candidate Key is a combination of a minimal set of attributes which can uniquely identify a tuple. Any relation can have more than a single Candidate Key, with the key being a simple or a composite key.
- **Super Key** – A Super Key is the set of attributes which can uniquely identify a tuple. So, a Candidate Key is a Super Key, but vice-versa isn't true.
- **Primary Key** – A Primary Key is a set of attributes that can be used to uniquely identify every tuple. So, if there are 3-4 candidate keys present in a relationship, then out those, one can be chosen as a Primary Key.
- **Alternate Key** – All the Candidate Keys other than the Primary Key is called as an Alternate Key.
- **Foreign Key** – An attribute that can only take the values present as the values of some other attribute, is the foreign key to the attribute to which it refers.

## Constraints Used In Database – PostgreSQL Tutorial

The constraints which you can use in databases are as follows:

- **NOT NULL** – The NOT NULL constraint ensures that a NULL value cannot be stored in a column
- **UNIQUE** – The UNIQUE constraint makes sure that all the values in a column are different
- **CHECK** -The CHECK constraint ensures that all the values in a column satisfy a specific condition.
- **DEFAULT** -The DEFAULT constraint consists of a set of default values for a column when no value is specified.
- **INDEX** – The INDEX constraint is used to create and retrieve data from the database very quickly

Now, that you know the commands in DDL and the various types of keys and constraints, let's move on to the next section i.e Data Manipulation Commands.

## Data Manipulation (DML) Commands – PostgreSQL Tutorial

This section of the article consists of the commands, by which you can manipulate your database. The commands are:

- [SET SEARCH\\_PATH](#)
- [INSERT](#)
- [UPDATE](#)
- [DELETE](#)
- [SELECT](#)

Apart from these commands, there are also other manipulative operators/functions such as:

- [Arithmetic, Bitwise, Compound and Comparison Operators](#)
- [Logical Operators](#)
- [Aggregate Functions](#)
- [Special Operators](#)
- [Set Operations](#)
- [Limit, Offset and Fetch](#)



## SET SEARCH\_PATH

This statement is used to mention which schema has to be used to perform all the operations.

### Syntax:

```
SET search_path TO schema_name;
```

### Example:

```
1 | SET search_path TO teachers;
```

## INSERT

The INSERT statement is used to insert new records in a table.

### Syntax:

The INSERT INTO statement can be written in the following two ways:

```
INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);
```

--You need not mention the column names

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

### Example:

```
1 | INSERT INTO TeachersInfo(TeacherID, TeacherName, Address, City, PostalCode, Country, Salary) VA
2 |
3 | INSERT INTO TeachersInfo VALUES ('02', 'Preeti','Queens Quay', 'Rio Claro', '13500', 'Brazil',
```

## UPDATE

The UPDATE statement is used to modify the existing records in a table.

### Syntax:

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

### Example:

```
1 | UPDATE TeachersInfo
2 | SET TeacherName = 'Alfred', City= 'Frankfurt'
3 | WHERE TeacherID = '01';
```

## DELETE

The DELETE statement is used to delete existing records in a table.

### Syntax:

```
DELETE FROM table_name WHERE condition;
```

### Example:

```
1 | DELETE FROM TeachersInfo WHERE TeacherName='Vinod';
```

## SELECT

The SELECT statement is used to select data from a database and the data returned is stored in a result table, called the **result-set**.

The following are the two ways of using this statement:

### Syntax:



```
SELECT column1, column2, ...
FROM table_name;

--(*) is used to select all from the table

SELECT * FROM table_name;
```

**Example:**

```
1 | SELECT Teachername, City FROM TeachersInfo; SELECT * FROM TeachersInfo;
```

Apart from the individual SELECT keyword, you can use the SELECT keyword with the following statements:

- [DISTINCT](#)
- [ORDER BY](#)
- [GROUP BY](#)
- [HAVING Clause](#)

**The 'SELECT DISTINCT' Statement**

The SELECT DISTINCT statement is used to return only distinct or different values. So, if you have a table with duplicate values, then you can use this statement to list distinct values.

**Syntax:**

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

**Example:**

```
1 | SELECT Country FROM TeachersInfo;
```

**The 'ORDER BY' Statement**

The ORDER BY statement is used to sort the desired results in ascending or descending order. By default, the results would be sorted in ascending order. If you want to sort the records in descending order, then you have to use the **DESC** keyword.

**Syntax:**

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ...ASC|DESC;
```

**Example:**

```
1 | SELECT * FROM TeachersInfo
2 | ORDER BY Country;
3 |
4 | SELECT * FROM TeachersInfo
5 | ORDER BY Country DESC;
6 |
7 | SELECT * FROM TeachersInfo
8 | ORDER BY Country, TeachersName;
9 |
10 | SELECT * FROM TeachersInfo
11 | ORDER BY Country ASC, TeachersName DESC;
```

**The 'GROUP BY' Statement**

This statement is used with the aggregate functions to group the result-set by one or more columns.

**Syntax:**

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

**Example:**

```
1 | SELECT COUNT(TeacherID), Country
2 | FROM TeachersInfo
3 | GROUP BY Country
4 | ORDER BY COUNT(TeacherID) DESC;
```



The 'HAVING' Clause Statement

Since the **WHERE** keyword cannot be used with aggregate functions, the HAVING clause was introduced.

Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

Example:

```
1 | SELECT COUNT(TeacherID), Country
2 | FROM TeachersInfo
3 | GROUP BY Country
4 | HAVING COUNT(Salary) &gt; 40000;
```

Arithmetic, Bitwise, Compound and Comparison Operators – PostgreSQL Tutorial

The arithmetic, bitwise, compound and comparison operators are as follows:

edureka!

ARITHMETIC OPERATORS

+ , - , \* , / , %

BITWISE OPERATORS

& , | , ^

COMPARISON OPERATORS

= , < , > , <= , >= , <>

COMPOUND OPERATORS

+ = , \* = , - = , / = , % = , & = , ^ = , | =

LOGICAL OPERATORS

This set of operators consists of logical operators such as [AND/OR/NOT](#).

AND OPERATOR

This operator displays the records, which satisfy all the conditions separated by AND.

Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

Example:

tabases Training

SQL ESSENTIALS TRAINING & CERTIFICATION

SQL Essentials Training & Certification

Reviews 5(6819)

MYSQL DBA CERTIFICATION TRAINING

MySQL DBA Certification Training

Reviews 5(3756)

MONGODB CERTIFICATION TRAINING

MongoDB Certification Training

Reviews 4(15278)

TERADATA CERTIFICATION TRAINING

Teradata Certification Training

Reviews 5(2414)



```
1 | SELECT * FROM TeachersInfo
2 | WHERE Country='India' AND City='South Korea';
```

## OR OPERATOR

This operator displays those records which satisfy any of the conditions separated by OR.

### *Syntax:*

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition1 OR condition2 OR condition3 ...;
```

### *Example:*

```
1 | SELECT * FROM TeachersInfo  
2 | WHERE Country='India' OR City='South Korea';
```

## NOT OPERATOR

The NOT operator displays a record when the condition (s) is NOT TRUE.

### *Syntax:*

```
SELECT column1, column2, ...  
FROM table_name  
WHERE NOT condition;
```

### *Example:*

```
1 | SELECT * FROM TeachersInfo  
2 | WHERE NOT Country='India';  
3 |  
4 | --You can also combine all the above three operators and write a query like this:  
5 |  
6 | SELECT * FROM TeachersInfo  
7 | WHERE NOT Country='India' AND (City='Bengaluru' OR City='Kolkata');
```

## Aggregate Functions – PostgreSQL Tutorial

The following section of the article will include functions such as:

- [MIN\(\)](#)
- [MAX\(\)](#)
- [COUNT\(\)](#)
- [AVG\(\)](#)
- [SUM\(\)](#)

### MIN() Function

The MIN function returns the smallest value of the selected column in a table.

### *Syntax:*

```
SELECT MIN(column_name)  
FROM table_name  
WHERE condition;
```

### *Example:*

```
1 | SELECT MIN(Salary) AS SmallestSalary  
2 | FROM TeachersInfo;
```

### MAX() Function

The MAX function returns the largest value of the selected column in a table.

### *Syntax:*

```
SELECT MAX (column_name)  
FROM table_name  
WHERE condition;
```

### *Example:*

```
1 | SELECT MAX(Salary) AS LargestSalary  
2 | FROM TeachersInfo;
```



## COUNT() Function

The COUNT function returns the number of rows that match the specified criteria.

### Syntax:

```
SELECT COUNT (column_name)
FROM table_name
WHERE condition;
```

### Example:

```
1 | SELECT COUNT(TeacherID)
2 | FROM TeachersInfo;
```

## AVG() Function

The AVG function returns the average value of a numeric column that you choose.

### Syntax:

```
SELECT AVG (column_name)
FROM table_name
WHERE condition;
```

### Example:

```
1 | SELECT AVG(Salary)
2 | FROM TeachersInfo;
```

## SUM() Function

The SUM function returns the total sum of a numeric column that you choose.

### Syntax:

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

### Example:

```
1 | SELECT SUM(Salary)
2 | FROM TeachersInfo;
```

## Special Operators – PostgreSQL Tutorial

This section of the article will include the following operators:

- [BETWEEN](#)
- [IS NULL](#)
- [LIKE](#)
- [IN](#)
- [EXISTS](#)
- [ALL](#)
- [ANY](#)

### BETWEEN Operator

The BETWEEN operator is an inclusive operator which selects values(numbers, texts or dates) within a given range.

### Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

### Example:

```
1 | SELECT * FROM TeachersInfo
2 | WHERE Fees BETWEEN 30000 AND 45000;
```

### IS NULL Operator

Since, it is not possible to test for NULL values with the comparison operators(=, <, >), we can use IS NULL and IS NOT NULL operators instead.



### ***Syntax:***

```
--Syntax for IS NULL

SELECT column_names
FROM table_name
WHERE column_name IS NULL;

--Syntax for IS NOT NULL

SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

### ***Example:***

```
1 | SELECT TeacherName FROM TeachersInfo
2 | WHERE Address IS NULL;
3 | SELECT TeacherName FROM TeachersInfo
4 | WHERE Address IS NOT NULL;
```

## **LIKE Operator**

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column of a table.

The mentioned below are the two wildcards that are used in conjunction with the LIKE operator:

- % – The percent sign represents zero, one, or multiple characters
- \_ – The underscore represents a single character

### ***Syntax:***

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

### ***Example:***

```
1 | SELECT * FROM TeachersInfo
2 | WHERE TeacherName LIKE 'S%';
```

## **IN Operator**

The IN operator is a shorthand operator and is used for multiple OR conditions.

### ***Syntax:***

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

### ***Example:***

```
1 | SELECT * FROM TeachersInfo
2 | WHERE Country IN ('South Korea', 'India', 'Brazil');
```

**NOTE:** You can also use IN while writing Nested Queries.

## **EXISTS Operator**

The EXISTS operator is used to test if a record exists or not.

### ***Syntax:***

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

### ***Example:***





```

1 | SELECT TeacherName
2 | FROM TeachersInfo
3 | WHERE EXISTS (SELECT * FROM TeachersInfo WHERE TeacherID = 05 AND Salary & & & > 2

```

## ALL Operator

The ALL operator is used with a WHERE or HAVING clause and returns true if all of the sub-query values meet the condition.

### Syntax:

```

SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
(SELECT column_name FROM table_name WHERE condition);

```

### Example:

```

1 | SELECT TeacherName
2 | FROM TeachersInfo
3 | WHERE TeacherID = ALL (SELECT TeacherID FROM TeachersInfo WHERE Salary & & & > 250

```

## ANY Operator

Similar to the ALL operator, the ANY operator is also used with a WHERE or HAVING clause and returns true if any of the sub-query values meet the condition.

### Syntax:

```

SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
(SELECT column_name FROM table_name WHERE condition);

```

### Example:

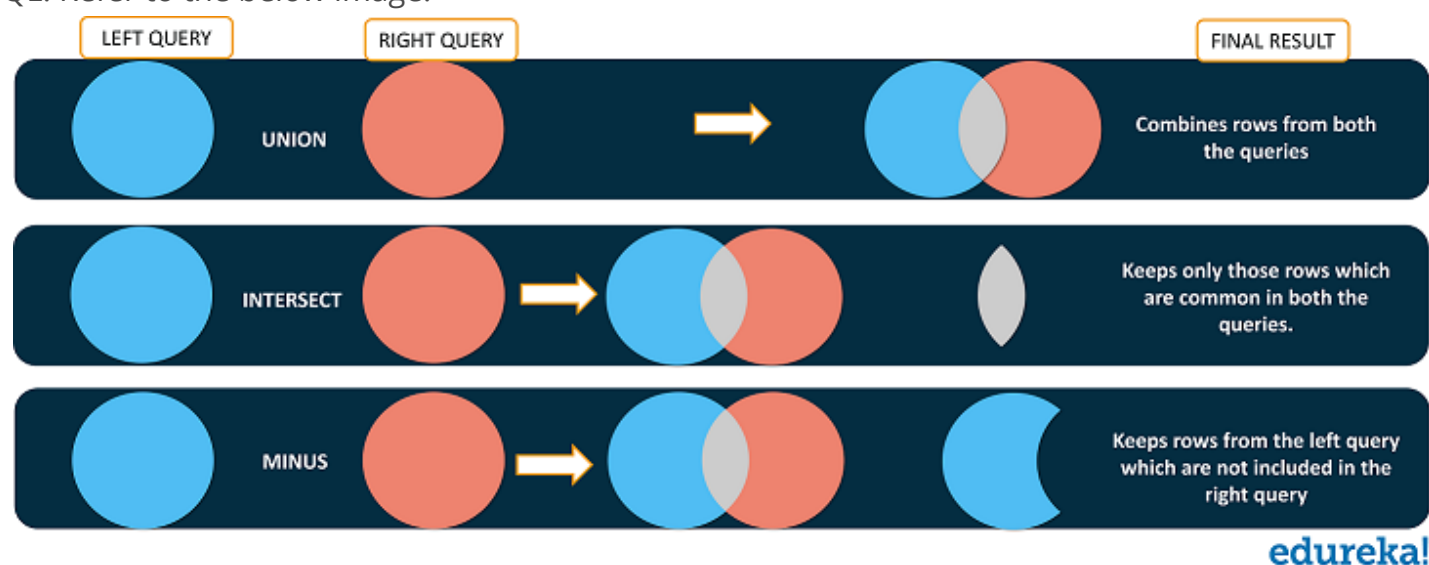
```

1 | SELECT TeacherName
2 | FROM TeachersInfo
3 | WHERE TeacherID = ANY (SELECT TeacherID FROM TeachersInfo WHERE Salary BETWEEN 32000 AND 45000)

```

## Set Operations – PostgreSQL Tutorial

There are mainly three set operations: [UNION](#), [INTERSECT](#), [MINUS](#). You can refer to the image below to understand the set operations in SQL. Refer to the below image:



## UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.

### Syntax

```

SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;

```

## INTERSECT

The INTERSECT clause is used to combine two SELECT statements and return the intersection of the data-sets of both the SELECT statements.

### Syntax



```
SELECT Column1 , Column2 ....
FROM table_name;
WHERE condition

INTERSECT

SELECT Column1 , Column2 ....
FROM table_name;
WHERE condition
```

**EXCEPT**

The EXCEPT operator returns those tuples that are returned by the first SELECT operation, and are not returned by the second SELECT operation.

**Syntax**

```
SELECT column_name
FROM table_name;

EXCEPT

SELECT column_name
FROM table_name;
```

**Limit, Offset and Fetch – PostgreSQL Tutorial**

**LIMIT**

The LIMIT statement is used to retrieve a portion of the rows out of the complete rows present in the table.

**Syntax:**

```
SELECT column_name

FROM table_name LIMIT number;
```

**Example:**

```
1 | SELECT * FROM TeachersInfo LIMIT 5;
```

**OFFSET**

The OFFSET statement omits the number of rows you mention and then retrieves the rest portion of the rows.

**Syntax:**

```
SELECT column_name

FROM table_name OFFSET number LIMIT number;
```

**Example:**

```
1 | --Select 3 rows from TeachersInfo after the 5th row
2 | SELECT * FROM TeachersInfo OFFSET 5 LIMIT 3;
3 |
4 | --Select all rows from TeachersInfo
5 | SELECT * FROM TeachersInfo OFFSET 2;
```

**FETCH**

The FETCH keyword is used to fetch records from a table using a cursor. Here the cursors will be the following:

- NEXT
- PRIOR
- FIRST
- LAST
- RELATIVE Count
- ABSOLUTE Count
- Count
- ALL
- BACKWARD
- BACKWARD Count
- BACKWARD ALL



- FORWARD
- FORWARD Count
- FORWARD ALL

#### Syntax:

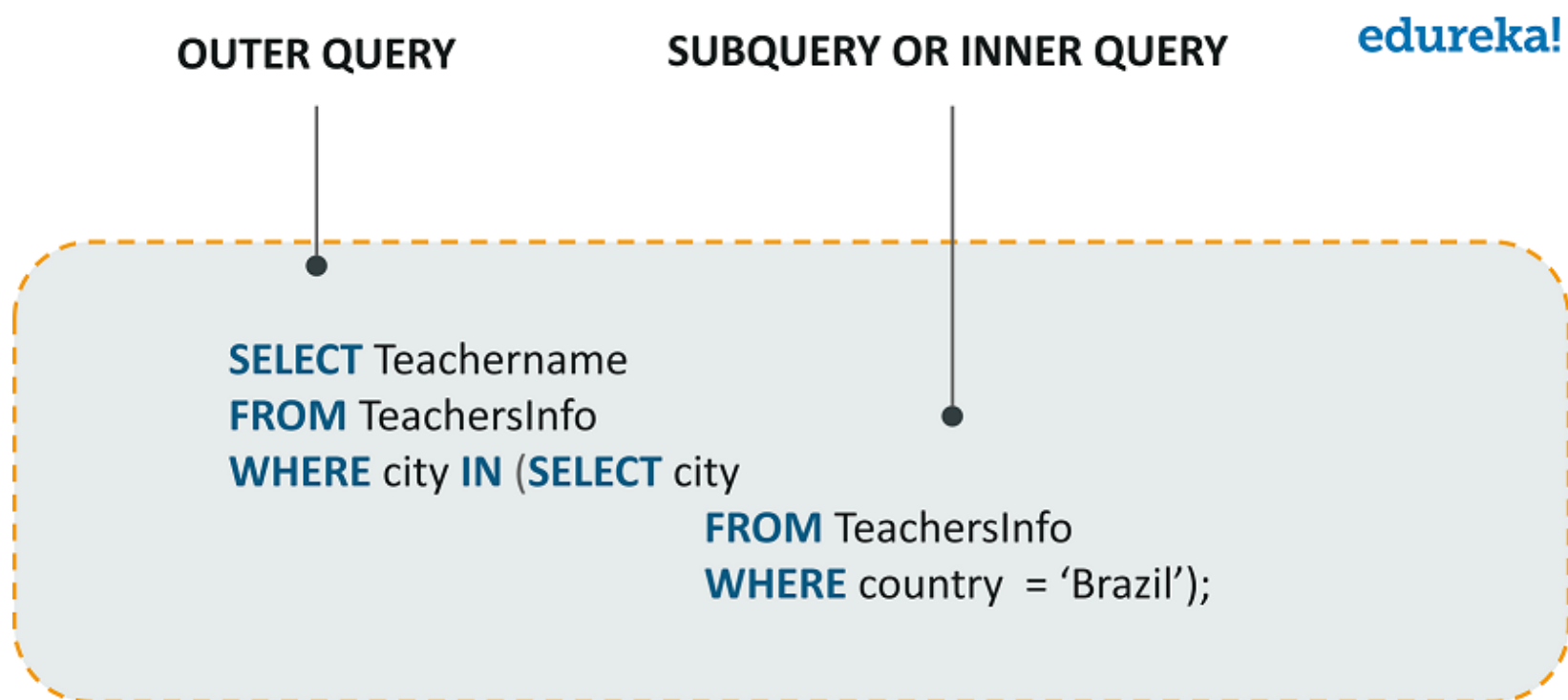
```
FETCH cursorname;
```

#### Example:

```
1 | SELECT * FROM TeachersInfo OFFSET 5 FETCH FIRST 5 ROWS ONLY;
```

## Nested Queries – PostgreSQL Tutorial

**Nested queries** are those queries which have an outer query and inner subquery. So, basically, the subquery is a query which is nested within another query such as SELECT, INSERT, UPDATE or DELETE. Refer to the image below:

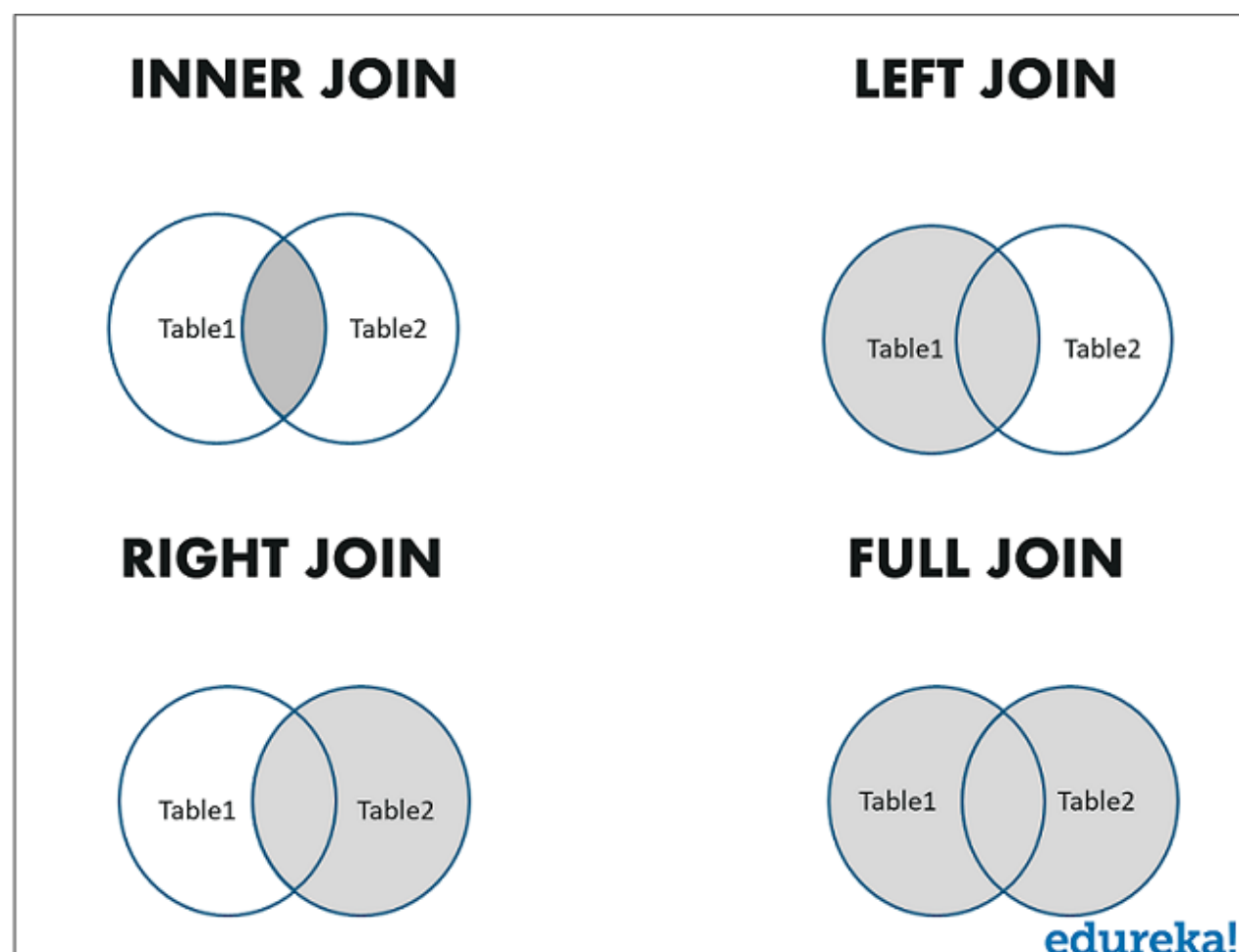


So, when you execute this query, you will see the name of the teacher who is from Brazil.

## Joins – PostgreSQL Tutorial

JOINS in PostgreSQL are used to combine rows from two or more tables, based on a related column between those tables. The following are the types of joins:

- **INNER JOIN:** The INNER JOIN returns those records which have matching values in both the tables.
- **LEFT JOIN:** The LEFT JOIN returns records from the left table, and also those records which satisfy the condition from the right table.
- **RIGHT JOIN:** The RIGHT JOIN returns records from the right table, and also those records which satisfy the condition from the left table.
- **FULL JOIN:** The FULL JOIN returns all those records which either have a match in the left or the right table.



Let's consider the below table apart from the TeachersInfo table, to understand the syntax of joins.

SubjectID	TeacherID	SubjectName
1	10	Maths
2	11	Physics
3	12	Chemistry

**INNER JOIN**

**Syntax:**

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

**Example:**

```
1 | SELECT Subjects.SubjectID, TeachersInfo.TeacherName
2 | FROM Subjects
3 | INNER JOIN TeachersInfo ON Subjects.TeacherID = TeachersInfo.TeacherID;
```

**LEFT JOIN**

**Syntax:**

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2 ON table1.column_name = table2.column_name;
```

**Example:**

```
1 | SELECT TeachersInfo.TeacherName, Subjects.SubjectID
2 | FROM TeachersInfo
3 | LEFT JOIN Subjects ON TeachersInfo.TeacherID = Subjects.TeacherID
4 | ORDER BY TeachersInfo.TeacherName;
```

**RIGHT JOIN**

**Syntax:**

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2 ON table1.column_name = table2.column_name;
```

**Example:**

```
1 | SELECT Subjects.SubjectID
2 | FROM Subjects
3 | RIGHT JOIN TeachersInfo ON Subjects.SubjectID = TeachersInfo.TeacherID
4 | ORDER BY Subjects.SubjectID;
```

**FULL JOIN**

**Syntax:**

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2 ON table1.column_name = table2.column_name;
```

**Example:**

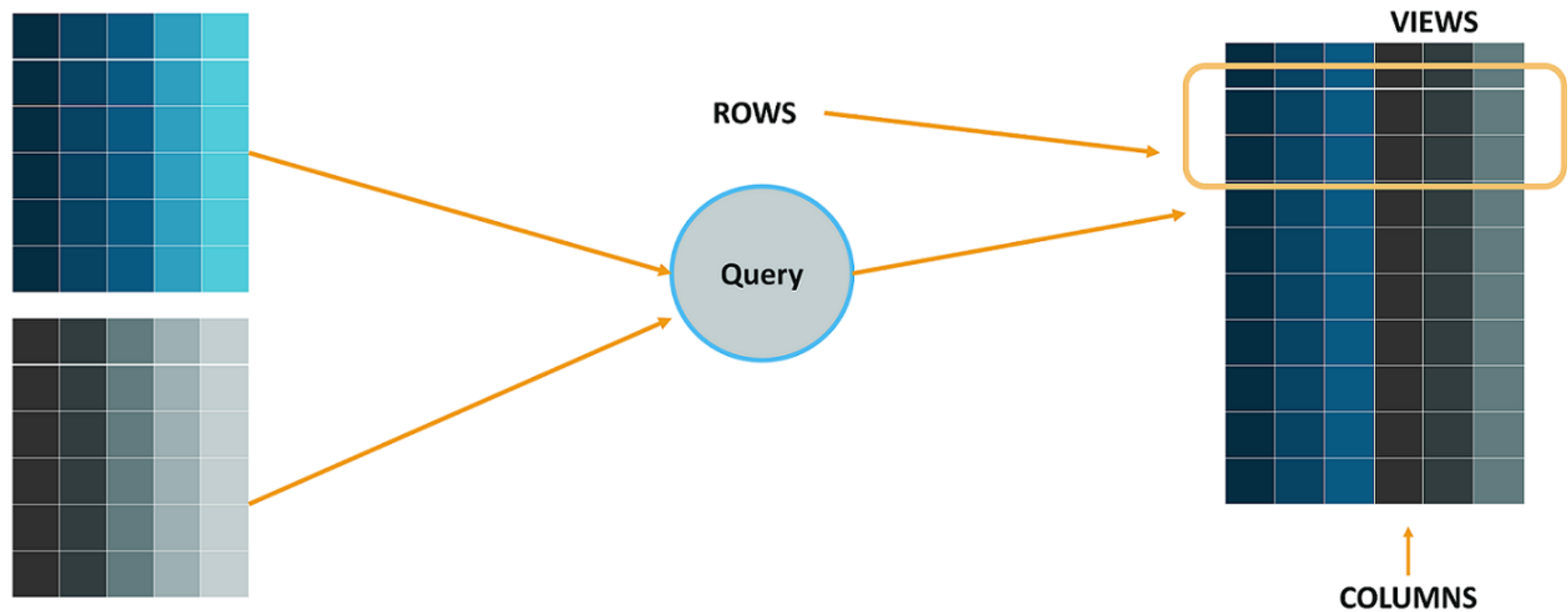
```
1 | SELECT TeachersInfo.TeacherName, Subjects.SubjectID
2 | FROM TeachersInfo
3 | FULL OUTER JOIN Subjects ON TeachersInfo.TeacherID = Subjects.SubjectID
4 | ORDER BY TeachersInfo.TeacherName;
```

Now, next in this article, I will discuss Views, Stored Procedures, and Triggers.

**Views – PostgreSQL Tutorial**

A view is a single table, which is derived from other tables. So, a view contains rows and columns similar to a real table and has fields from one or more table.





### The 'CREATE VIEW' statement

The CREATE VIEW statement is used to create a view from an existing table.

#### Syntax

```
CREATE VIEW view_name AS
SELECT column1, column2, ..., columnN
FROM table_name
WHERE condition;
```

#### Example

```
1 CREATE VIEW teachers_view AS
2 SELECT TeacherName, TeacherID
3 FROM TeachersInfo
4 WHERE City = 'Bengaluru';
```

### The 'DROP VIEW' statement

The DROP VIEW statement is used to delete a view.

#### Syntax

```
DROP VIEW view_name;
```

#### Example

```
1 DROP VIEW teachers_view;
```

## PostgreSQL Tutorial For Beginners: Stored Procedures

Stored Procedures are snippets of codes which can be saved and re-used.

#### Syntax

```
CREATE PROCEDURE procedure_name
LANGUAGE lang_name;
```

#### Example

```
1 --Create two tables
2
3 CREATE TABLE tbl1(tb1id int);
4 CREATE TABLE tbl2(tb2id int);
5
6 --Create Procedure
7 CREATE PROCEDURE insert_data (a1 integer, b1 integer)
8 LANGUAGE SQL
9 AS $$
10 INSERT INTO tbl1 VALUES (a1);
11 INSERT INTO tbl2 VALUES (b1);
12 $$;
13
14 CALL insert_data(4, 5);
```





## SQL Essentials Training & Certification

[Weekday / Weekend Batches](#)

[See Batch Details](#)

Triggers are a set of SQL statements which are stored in the database catalog. These statements are executed whenever an event associated with a table occurs. So, a **trigger** can be invoked either **BEFORE** or **AFTER** the data is changed by **INSERT**, **UPDATE** or **DELETE** statement.

### Syntax

```
CREATE TRIGGER trigger_name [BEFORE|AFTER|INSTEAD OF] event_name
ON table_name
[
--Mention Logic Here
];
```

### Example

```
1 | --CREATE TRIGGER
2 | CREATE TRIGGER example_trigger AFTER INSERT ON TeachersInfo;
```

## Data Control (DCL) Commands – PostgreSQL Tutorial

This section consists of those commands which are used to control privileges in the database. The commands are:

- [GRANT](#)
- [REVOKE](#)

### GRANT

The GRANT command is used to provide user access privileges or other privileges for the schema.

#### Syntax:

```
GRANT privileges ON object TO user;
```

#### Example:

```
1 | GRANT INSERT ON TeachersInfo TO PUBLIC;
```

### REVOKE

The REVOKE command is used to withdraw user's access privileges given by using the GRANT command.

#### Syntax:

```
REVOKE privileges ON object FROM user;
```

#### Example:

```
1 | REVOKE INSERT ON TeachersInfo FROM PUBLIC;
```

Now, let's move on to the last section of this article i.e. the TCL Commands.

## Transaction Control (TCL) Commands – PostgreSQL Tutorial

- [BEGIN](#)
- [COMMIT](#)
- [ROLLBACK](#)
- [SAVEPOINT](#)
  - [RELEASE SAVEPOINT](#)
- [SET TRANSACTION](#)

### BEGIN

The BEGIN TRANSACTION command is used to start the transaction.

#### Syntax:

```
BEGIN;
```



BEGIN TRANSACTION;

**Example:**

```
1 BEGIN;  
2 DELETE * FROM TeachersInfo WHERE Salary = 65000;
```

## COMMIT

The COMMIT command saves all the transactions to the database since the last COMMIT or ROLLBACK command.

**Syntax:**

```
COMMIT;
```

**Example:**

```
1 DELETE * FROM TeachersInfo WHERE Salary = 65000;  
2 COMMIT;
```

## ROLLBACK

The ROLLBACK command is used to undo transactions since the last COMMIT or ROLLBACK command was issued.

**Syntax:**

```
ROLLBACK;
```

**Example:**

```
1 DELETE * FROM TeachersInfo WHERE Salary = 65000;  
2 ROLLBACK;
```

## SAVEPOINT

The SAVEPOINT command defines a new savepoint within the current transaction.

**Syntax:**

```
SAVEPOINT savepoint_name; --Syntax for saving the SAVEPOINT  
ROLLBACK TO savepoint_name --Syntax for rolling back to the SAVEPOINT
```

**Example:**

```
1 SAVEPOINT SP1;  
2 DELETE FROM TeachersInfo WHERE Fees = 65000;  
3 SAVEPOINT SP2;
```

## RELEASE SAVEPOINT

The RELEASE SAVEPOINT command is used to remove a SAVEPOINT that you have created.

**Syntax:**

```
RELEASE SAVEPOINT savepoint_name;
```

**Example:**

```
1 RELEASE SAVEPOINT SP2;
```

## SET TRANSACTION

The SET TRANSACTION command sets the characteristics of the current transaction.

**Syntax:**

```
SET TRANSACTION transaction_mode;
```

## UUID Data Type – PostgreSQL Tutorial

UUID data type stores Universally Unique Identifiers (UUID) with a 128 byte length. It is written as a sequence of lower-case hexadecimal digits and is generated by an algorithm. This algorithm is designed to make sure that the same UUID is not generated by any other person in the universe.

**Example:**

```
1 --Generate a a unique UUID  
2 SELECT uuid_generate_v4();
```





With this, we come to the end of this article on PostgreSQL Tutorial For Beginners. I hope you enjoyed reading this article on PostgreSQL Tutorial For Beginners. We have seen the different commands that will help you write queries and play around with your databases. *If you wish to learn more about SQL and get to know this open source relational database, then check out our [SQL Essentials Training](#). This training will help you understand SQL in depth and help you achieve mastery over the subject.*

Got a question for us? Please mention it in the comments section of "**PostgreSQL Tutorial For Beginners**" and I will get back to you.

Recommended videos for you



Introduction to MongoDB

[Watch Now](#)



Build Application With MongoDB

[Watch Now](#)

Recommended blogs for you



Top 65 SQL Interview Questions You Must Prepare In 2020

[Read Article](#)



What is Normalization in SQL and what are its types?

[Read Article](#)



INSERT Query SQL - All You Need to Know about the INSERT statement

[Read Article](#)



SQL Basics - One Stop for Beginners

[Read Article](#)

Comments

0 Comments

Join the discussion

Enter your comment here...

Trending Courses in Databases



[SQL Essentials Training & Certification](#)

7k Enrolled Learners  
 Weekend/Weekday  
 Self Paced

[Reviews](#)  
★★★★★ 5 (2750)



[MySQL DBA Certification Training](#)

4k Enrolled Learners  
 Weekend  
 Live Class

[Reviews](#)  
★★★★★ 5 (1550)



[MongoDB Certification Training](#)

16k Enrolled Learners  
 Weekend  
 Live Class

[Reviews](#)  
★★★★★ 4 (6150)



[Teradata Certification Training](#)

3k Enrolled Learners  
 Weekend  
 Live Class

[Reviews](#)  
★★★★★ 5 (1000)





Browse Categories

- Artificial Intelligence
- BI and Visualization
- Big Data
- Blockchain
- Cloud Computing
- Cyber Security
- Data Science
- Data Warehousing and ETL
- DevOps
- Digital Marketing
- Enterprise
- Front End Web Development
- Mobile Development
- Operating Systems
- Programming & Frameworks
- Project Management and Methodologies
- Robotic Process Automation
- Software Testing
- Systems & Architecture

edureka!

TRENDING CERTIFICATION COURSES

- [DevOps Certification Training](#)
- [AWS Architect Certification Training](#)
- [Big Data Hadoop Certification Training](#)
- [Tableau Training & Certification](#)
- [Python Certification Training for Data Science](#)
- [Selenium Certification Training](#)
- [PMP® Certification Exam Training](#)
- [Robotic Process Automation Training using UiPath](#)
- [Apache Spark and Scala Certification Training](#)
- [Microsoft Power BI Training](#)
- [Online Java Course and Training](#)
- [Python Certification Course](#)

COMPANY

- [About us](#)
- [News & Media](#)
- [Reviews](#)
- [Contact us](#)
- [Blog](#)
- [Community](#)
- [Sitemap](#)
- [Blog Sitemap](#)
- [Community Sitemap](#)
- [Webinars](#)

TRENDING MASTERS COURSES

- [Data Scientist Masters Program](#)
- [DevOps Engineer Masters Program](#)
- [Cloud Architect Masters Program](#)
- [Big Data Architect Masters Program](#)
- [Machine Learning Engineer Masters Program](#)
- [Full Stack Web Developer Masters Program](#)
- [Business Intelligence Masters Program](#)
- [Data Analyst Masters Program](#)
- [Test Automation Engineer Masters Program](#)
- [Post-Graduate Program in Artificial Intelligence & Machine Learning](#)
- [Post-Graduate Program in Big Data Engineering](#)

WORK WITH US

- [Careers](#)
- [Become an Instructor](#)
- [Become an Affiliate](#)
- [Become a Partner](#)
- [Hire from Edureka](#)

DOWNLOAD APP



CATEGORIES



CATEGORIES

- [Cloud Computing](#) | [DevOps](#) | [Big Data](#) | [Data Science](#) | [BI and Visualization](#) | [Programming & Frameworks](#) | [Software Testing](#) | [Project Management and Methodologies](#) | [Robotic Process Automation](#) | [Frontend Development](#) | [Data Warehousing and ETL](#) | [Artificial Intelligence](#) | [Blockchain](#) | [Databases](#) | [Cyber Security](#) | [Mobile Development](#) | [Operating Systems](#) | [Architecture & Design Patterns](#) | [Digital Marketing](#)

TRENDING BLOG ARTICLES



TRENDING BLOG ARTICLES

- [Selenium tutorial](#) | [Selenium interview questions](#) | [Java tutorial](#) | [What is HTML](#) | [Java interview questions](#) | [PHP tutorial](#) | [JavaScript interview questions](#) | [Spring tutorial](#) | [PHP interview questions](#) | [Inheritance in Java](#) | [Polymorphism in Java](#) | [Spring interview questions](#) | [Pointers in C](#) | [Linux commands](#) | [Android tutorial](#) | [JavaScript tutorial](#) | [jQuery tutorial](#) | [SQL interview questions](#) | [MySQL tutorial](#) | [Machine learning tutorial](#) | [Python tutorial](#) | [What is machine learning](#) | [Ethical hacking tutorial](#) | [SQL injection](#) | [AWS certification career opportunities](#) | [AWS tutorial](#) | [What Is cloud computing](#) | [What is blockchain](#) | [Hadoop tutorial](#) | [What is artificial intelligence](#) | [Node Tutorial](#) | [Collections in Java](#) | [Exception handling in java](#) | [Python Programming Language](#) | [Python interview questions](#) | [Multithreading in Java](#) | [ReactJS Tutorial](#) | [Data Science vs Big Data vs Data Analyt...](#)



