**PostgreSQL Tutorial**

## What is PostgreSQL?

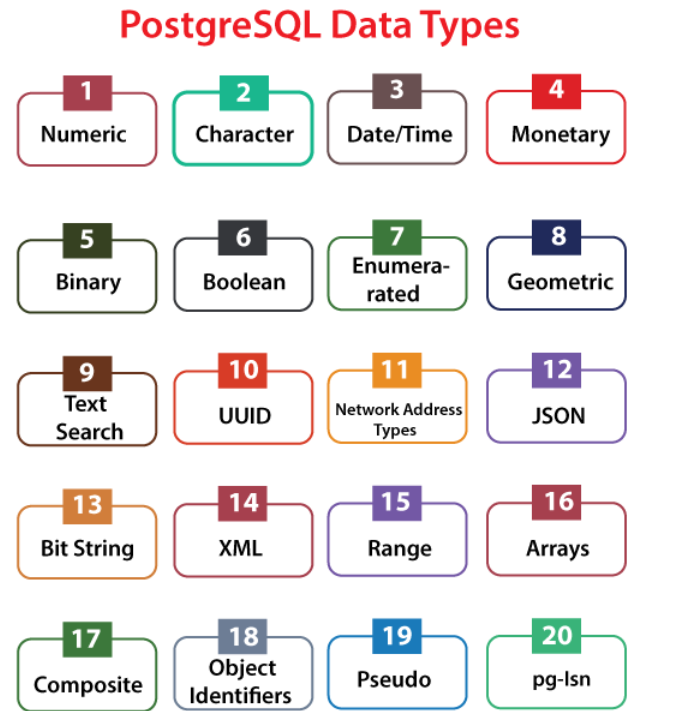
PostgreSQL is an *ORDBMS* [*Open-Source Object-Relational Database Management System*].

It supports both relational (SQL) and non-relational (JSON) queries. It is used to store data securely; supporting best practices, and allow recovering them when the request is processed.

### Features

* It is one of the most popular databases supporting JSON (non-relational) queries and SQL for (relational) queries. PostgreSQL is an *object-relational database management system (ORDBMS)*. It contains the various advanced data types and robust feature sets, which increase the extensibility, reliability, and data integrity of the software.
* It is the first DBMS, which perform *MVCC* [*Multi-Version Concurrency Control*] feature, even before Oracle. The multi-version concurrency control attribute is known as *snapshot* isolation in Oracle.
* **Compatible on several operation systems:** PostgreSQL runs on all major operating systems such as *Microsoft Windows, Linux, MacOS X, UNIX (AIX, BSD, HP-UX, SGI IRIX, Solaris, and Tru64),* etc.
* **Compatible with various programming languages:** It supports multiple programming interfaces such as*C/C++, JAVA, Python, Perl, Ruby, Tcl, and ODBC (Open Database Connectivity)*.
* The primary objective of PostgreSQL is to handle a variety of jobs from single technologies to web service or the data warehouse with several parallel users.
* **Compatible with Data Integrity:** It supports data integrity which includes the following:
  + Primary Keys
  + UNIQUE, NOT NULL
  + Foreign Keys
  + Explicit Locks, Advisory Locks
  + Exclusion Constraints
* **Compatible with multiple data types:** PostgreSQL support various data types such as:
  + *Structured:* Array, Date and Time, UUID (Universally Unique Identifier), Array, Range.
  + *Primitives:* String, Integer, Boolean, Numeric.
  + *Customizations:* Custom Types, Composite.
  + *Geometry:* Polygon, Circle, Line, Point,
  + *Document:* XML, JSON/JSONB, Key-value.
* It is written in *C* programming language.

## Data Types



## PostgreSQL Commands

DDL (Data Definition Language)

| **Description** | **Query** |
| --- | --- |
| To list all database | \l, \l+,  SELECT datname FROM pg\_database; |
| To list all schema | \dn, \dn+  SELECT schema\_name FROM information\_schema.schemata; |
| To list all schema and table | SELECT table\_schema, table\_name FROM information\_schema.tables; |
| To list all table name | \dt, \dt+  SELECT table\_name FROM information\_schema.tables WHERE table\_schema = 'public'; |

### DDL (Data Definition Language)

| **Description** | **Query** |
| --- | --- |
| Create table | **create** **table** Student(Stu\_id **int**, Stu\_Name text, Stu\_Age **int**, Stu\_address **char**(30)); |
| Show table | \dt |
| Describe table | \d <table name> |
| Alter table add column | **ALTER** **TABLE** <table name> **ADD** **COLUMN** <column name> **REAL**; |
| Alter table drop column | **ALTER** **TABLE** <table name> **DROP** **COLUMN** <column name>; |
| Drop table | **drop table <table name>** |
| Truncate table | **truncate table <table name>** |

### DML (Data Manipulation Language)

| **Description** | **Query** |
| --- | --- |
| Insert into table | INSERT INTO table\_name(column1, column2, …) VALUES (value1, value2, …); |
| Update table | UPDATE table\_name SET column1 = value1, column2 = value2 WHERE condition; |
| Delete table | DELETE FROM table\_name WHERE condition; |

### DQL (Data Query Language)

| **Description** | **Query** |
| --- | --- |
| Select | Select \* from table; |

### DCL (Data Control Language)

| **Description** | **Query** |
| --- | --- |
| Grant |  |
| Revoke |  |

### TCL (Transaction Control Language)

| **Description** | **Query** |
| --- | --- |
| Commit |  |
| Rollback |  |
| Savepoint |  |
| Set Transaction |  |

### Aggregate Functions

| **Description** | **Query** |
| --- | --- |
| Count |  |
| Sum |  |
| Average |  |
| Min |  |
| Max |  |

| **Description** | **Query** |
| --- | --- |
| Order By | SELECT  first\_name,  last\_name  FROM  customer  ORDER BY  first\_name ASC or DESC; |
| Group By | SELECT  customer\_id,  SUM (amount)  FROM  payment  GROUP BY  customer\_id; |
| Having By | SELECT  customer\_id,  SUM (amount)  FROM  payment  GROUP BY  customer\_id  HAVING  SUM (amount) > 200; |

## Join

### Inner Join

SELECT

a,

fruit\_a,

b,

fruit\_b

FROM

basket\_a

INNER JOIN basket\_b

ON fruit\_a = fruit\_b;

### Left Join

SELECT

a,

fruit\_a,

b,

fruit\_b

FROM

basket\_a

LEFT JOIN basket\_b

ON fruit\_a = fruit\_b;

### Right Join

SELECT

a,

fruit\_a,

b,

fruit\_b

FROM

basket\_a

RIGHT JOIN basket\_b

ON fruit\_a = fruit\_b;

### Full Outer Join

SELECT

employee\_name,

department\_name

FROM

employees e

FULL OUTER JOIN departments d

ON d.department\_id = e.department\_id;

### Cross Join

SELECT select\_list

FROM T1

CROSS JOIN T2;

### Self Join

SELECT

e.first\_name || ' ' || e.last\_name employee,

m .first\_name || ' ' || m .last\_name manager

FROM

employee e

INNER JOIN employee m ON m .employee\_id = e.manager\_id

ORDER BY manager;

### Natural Join

SELECT \* FROM products

NATURAL JOIN categories;

## Sub Query

### Non-Corelated Subquery

SELECT product\_name

FROM products

WHERE price > (SELECT AVG(price) FROM products);

### Corelated Subquery

SELECT customer\_name

FROM customers c

WHERE EXISTS (

SELECT 1

FROM orders o

WHERE o.customer\_id = c.customer\_id

AND o.order\_date > '2023-01-01'

);