PostgreSQL Privileges

1. Database Level Privileges

These privileges are granted using the `GRANT` command on a database.

Privilege	Description
CONNECT	Allows users to connect to the database.
CREATE	Allows users to create new schemas within the database.
TEMPORARY (or TEMP)	Allows users to create temporary tables.
ALL PRIVILEGES	Grants all database-level privileges (equivalent to CONNECT, CREATE, TEMP).

Example:

```
GRANT CONNECT ON DATABASE mydb TO user1;GRANT CREATE ON DATABASE mydb TO user1;
```

---

2. Schema Level Privileges

These privileges control access to a specific schema within a database.

Privilege	Description
CREATE	Allows users to create tables, views, or other objects within the schema.
USAGE	Allows users to access objects within the schema (needed to use tables, but doesn’t allow creating new ones).
ALL PRIVILEGES	Grants all schema-level privileges (CREATE, USAGE).

Example:

```
GRANT USAGE ON SCHEMA myschema TO user1;GRANT CREATE ON SCHEMA myschema TO user1;
```

---

3. Table Level Privileges

These privileges apply to tables, views, and sequences.

Privilege	Description
SELECT	Allows users to read data from the table.
INSERT	Allows users to insert new rows into the table.
UPDATE	Allows users to modify existing rows in the table.
DELETE	Allows users to remove rows from the table.
TRUNCATE	Allows users to delete all rows from the table.
REFERENCES	Allows users to create foreign keys referencing the table.
TRIGGER	Allows users to create triggers on the table.
ALL PRIVILEGES	Grants all table-level privileges (SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, TRIGGER).

Example:

```
GRANT SELECT, INSERT ON TABLE mytable TO user1;GRANT ALL PRIVILEGES
ON TABLE mytable TO user1;
```

5. Role-Based Privileges

Roles in PostgreSQL act as both **users** and **groups**. You can assign privileges to a role and then grant that role to multiple users.

Privilege	Description
LOGIN	Allows the role to log in to the database.
SUPERUSER	Grants full control over the database.
CREATEDB	Allows the role to create databases.
CREATEROLE	Allows the role to create other roles.
REPLICATION	Grants replication privileges.
BYPASSRLS	Allows the role to bypass row-level security policies.

Example:

```
sql
CopyEdit
CREATE ROLE analyst WITH LOGIN PASSWORD 'securepass';GRANT SELECT ON
ALL TABLES IN SCHEMA public TO analyst;
```

6. Column-Level Privileges

Instead of granting permissions on the entire table, you can restrict access to specific columns.

Privilege	Description
SELECT (column_name)	Allows reading specific columns.
INSERT (column_name)	Allows inserting values into specific columns.
UPDATE (column_name)	Allows modifying values in specific columns.

Example:

```
GRANT SELECT (name, email) ON customers TO analyst;GRANT UPDATE
(email) ON customers TO analyst;
```

7. Sequence-Level Privileges

Sequences are used for generating unique numbers (e.g., auto-incremented IDs). You can control how users interact with them.

Privilege	Description
USAGE	Allows users to use the sequence but not modify it.
SELECT	Allows reading the sequence's current value.
UPDATE	Allows modifying the sequence's current value.
ALL PRIVILEGES	Grants all sequence-related privileges.

Example:

```
GRANT USAGE, SELECT ON SEQUENCE orders_id_seq TO user1;
```

8. Function-Level Privileges

You can allow or restrict execution of stored functions and procedures.

Privilege	Description
EXECUTE	Allows executing the function or procedure.

Example:

```
GRANT EXECUTE ON FUNCTION calculate_discount(integer) TO sales_team;
```

9. View-Level Privileges

Views are virtual tables that display data from one or more tables. Privileges on views work similarly to tables.

Privilege	Description
SELECT	Allows reading data from the view.
INSERT	Allows inserting data into the underlying table via the view (if possible).
UPDATE	Allows updating data through the view (if possible).
DELETE	Allows deleting data through the view (if possible).

Example:

```
GRANT SELECT ON view_sales TO analyst;
```

10. Foreign Data Wrapper (FDW) Privileges

PostgreSQL allows access to external databases via **Foreign Data Wrappers (FDW)**. These require special permissions.

Privilege	Description
USAGE	Allows users to use the FDW but not create foreign tables.
CREATE	Allows users to create foreign tables using the FDW.

Example:

```
GRANT USAGE ON FOREIGN DATA WRAPPER my_fdw TO user1;
```

11. Tablespace-Level Privileges

Tablespaces define where database files are stored. Users need privileges to create objects in them.

Privilege	Description
CREATE	Allows users to create objects (tables, indexes) in the tablespace.

Example:

```
GRANT CREATE ON TABLESPACE mytablespace TO dbuser;
```

12. Row-Level Security (RLS) Policies

PostgreSQL supports **row-level security (RLS)** to control access to individual rows based on a policy.

Privilege	Description
ENABLE ROW LEVEL SECURITY	Enables RLS on a table.
CREATE POLICY	Allows users to define policies for row access.

Example:

```
ALTER TABLE employees ENABLE ROW LEVEL SECURITY;CREATE POLICY
emp_policy ON employees FOR SELECT USING (department = 'IT');
```

13. Event Trigger Privileges

Triggers can be executed before/after certain database events (e.g., `DDL COMMANDS`).

Privilege	Description
EXECUTE	Allows executing the trigger function.

Example:

```
GRANT EXECUTE ON FUNCTION audit_trigger() TO admin_user;
```

14. Procedural Language Privileges

PostgreSQL supports various procedural languages (e.g., `plpgsql`, `plpython`, `plperl`). Users need privileges to use them.

Privilege	Description
USAGE	Allows users to write functions in the procedural language.
CREATE	Allows users to create functions using the procedural language.

Example:

```
GRANT USAGE ON LANGUAGE plpythonu TO user1;
```

15. Publication & Subscription Privileges

In logical replication, **publications** and **subscriptions** require permissions to control which users can replicate data.

Privilege	Description
CREATE	Allows users to create publications or subscriptions.
USAGE	Allows users to access existing publications.

Example:

```
GRANT CREATE ON DATABASE mydb TO replication_user;
```

16. Materialized View Privileges

Materialized views store the result of a query and need to be refreshed periodically.

Privilege	Description
SELECT	Allows reading data from the materialized view.
REFRESH	Allows refreshing the materialized view.

Example:

```
GRANT SELECT, REFRESH ON MATERIALIZED VIEW sales_report TO analyst;
```

17. Operator Class & Operator Family Privileges

PostgreSQL allows users to define custom operator classes and families, which help in indexing.

Privilege	Description
USAGE	Allows using a specific operator class or family in indexes.

Example:

```
GRANT USAGE ON OPERATOR FAMILY pg_catalog.numeric_ops TO user1;
```

18. Domain-Level Privileges

Domains are user-defined data types that add constraints on top of existing data types.

Privilege	Description
USAGE	Allows using the domain in table definitions.

Example:

```
GRANT USAGE ON DOMAIN email_domain TO user1;
```

19. Aggregate Function Privileges

PostgreSQL allows creating **custom aggregate functions** that process multiple rows.

Privilege	Description
EXECUTE	Allows executing the aggregate function.

Example:

```
GRANT EXECUTE ON FUNCTION my_custom_avg(numeric) TO analyst;
```

20. Large Object Privileges

PostgreSQL supports storing binary data as **Large Objects (LOBs)**. These objects require special privileges.

Privilege	Description
SELECT	Allows reading the large object.
UPDATE	Allows modifying the large object.
DELETE	Allows deleting the large object.

Example:

```
GRANT SELECT, UPDATE ON LARGE OBJECT 123456 TO user1;
```

21. Membership & Role Management Privileges

These privileges control **which roles** a user can **manage**.

Privilege	Description
GRANT <ROLE>	Allows assigning a role to a user.
REVOKE <ROLE>	Removes role membership from a user.
ADMIN OPTION	Allows a user to grant/revoke a role to others.
INHERIT	Enables a role to inherit privileges from another role.

Example:

```
GRANT manager_role TO user1 WITH ADMIN OPTION;REVOKE manager_role
FROM user1;
```

22. Table Inheritance Privileges

PostgreSQL allows **table inheritance**, where a child table inherits from a parent.

Privilege	Description
SELECT	Allows selecting from the parent table.
INSERT	Allows inserting into the parent table (propagated to child).
UPDATE	Allows updating data in the parent table.
DELETE	Allows deleting rows in the parent table.

Example:

```
GRANT SELECT ON parent_table TO user1;
```

23. Index-Level Privileges

Indexes improve query performance, but they require underlying **table privileges** to be useful.

Privilege	Description
USAGE	Allows a user to use an index for query optimization.

Example:

```
GRANT USAGE ON INDEX my_index TO user1;
```

24. Database-Wide Default Privileges

Instead of granting privileges **one by one**, you can define **default privileges** for objects created in the future.

Privilege	Description
ALTER DEFAULT PRIVILEGES	Sets default privileges for future objects.

Example:

```
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO
analyst;
```

---

25. Replication Slot Privileges

For physical or logical replication, replication slots require permission.

Privilege	Description
USAGE	Allows a user to use a replication slot.
SELECT	Allows viewing replication slot details.

Example:

```
GRANT USAGE ON REPLICATION SLOT my_slot TO replication_user;
```

---

26. Logical Replication Subscription Privileges

Logical replication subscriptions allow specific table replication.

Privilege	Description
SUBSCRIBE	Allows a user to create a logical replication subscription.

Example:

```
GRANT SUBSCRIBE ON DATABASE mydb TO replication_user;
```

---

27. Custom Data Type Privileges

PostgreSQL allows defining **custom data types**.

Privilege	Description
USAGE	Allows using a custom data type.

Example:

```
GRANT USAGE ON TYPE custom_type TO user1;
```

28. Text Search Configuration Privileges

Text search configurations are used for **full-text search**.

Privilege	Description
USAGE	Allows using a text search configuration.

Example:

```
GRANT USAGE ON TEXT SEARCH CONFIGURATION my_config TO user1;
```

29. Extension-Level Privileges

Extensions allow additional functionality (e.g., `pg_stat_statements`, `postgis`).

Privilege	Description
CREATE	Allows a user to create an extension.
USAGE	Allows using the extension.

Example:

```
GRANT USAGE ON EXTENSION postgis TO gis_user;
```

30. Foreign Server Privileges

If you’re connecting to an external database (via `postgres_fdw`), you can manage access to it.

Privilege	Description
USAGE	Allows using a foreign server.

Example:

```
GRANT USAGE ON FOREIGN SERVER my_remote_server TO remote_user;
```

31. PostgreSQL Built-in Privileges — Column Format

Privilege Role	Purpose
pg_read_all_data	SELECT from all tables, views, sequences in all databases.
pg_write_all_data	INSERT, UPDATE, DELETE on all tables in all databases.
pg_monitor	Access monitoring views and functions (pg_stat*).
pg_read_all_settings	View PostgreSQL settings (SHOW ALL).
pg_read_all_stats	View system statistics (like locks, table stats).
pg_stat_scan_tables	Allow manual ANALYZE execution to scan tables.
pg_execute_server_program	Allow executing server programs with COPY FROM PROGRAM.
pg_signal_backend	Allow sending signals to other backend processes (e.g., terminate queries).
replication	Enable replication (streaming WAL, creating replication slots).

Suppose you want to allow user akash.nishad to **read everything** from every database without manually granting access database by database.

```
GRANT pg_read_all_data TO akash.nishad;
```