********Grant command*******

- PostgreSQL, the **GRANT** command is used to grant privileges or permissions to database objects such as tables, views, functions, and schemas

Commnad : grant < privileges> on <objects> To <user or roles> ;

NOTE:- permissions are typically granted at the schema or object level within a database.

A. OBJECT LEVEL PERMISSION:- 1.select 2.Delete 3. insert 4.Update

Condition 1:- If want to grant read only permission for specific user for specific schema then we use :-

Grant select on all tables in schema <schema_name> to user_name;

Condition 2:- if want grant all permission:-

Grant all privileges on all tables in schema <schema_name> to username;

Condition 3.:- if we want to give multiple permission for multiple user:-

Grant select,update,insert on all tables in schema <schema_name> to username1,username2;

Conditon4:- if we want to grant read only permission on specific table ,specific database:-

Postgres# \c dvd----------(dvd=database name)
Dvd# **grant select on actor to gaurav;**

(table_name=actor ; username:- gaurav)

B. Cluster level:-

Condition1:- Grant permission to user to cretate database:-

Command:- alter user <username> createdb;

Condition2:- Make user superuser

Alter user <username> with superuser

\dp command to obtain information about existing privileges for tables and columns.

```
dvd=> \dp
                                    Access privileges
 Schema |           Name           |   Type   |     Access privileges     | Column privileges | Policies
--------+--------------------------+----------+---------------------------+-------------------+----------
 public | actor                    | table    | postgres=arwdDxt/postgres+|                   |
        |                          |          | gaurav=r/postgres         |                   |
```

r -- SELECT ("read")
w -- UPDATE ("write")
a -- INSERT ("append")
d -- DELETE
D -- TRUNCATE
x -- REFERENCES
t -- TRIGGER
X -- EXECUTE
U -- USAGE
C -- CREATE
c -- CONNECT
T -- TEMPORARY
arwdDxt -- ALL PRIVILEGES (for tables, varies for other objects)
* -- grant option for preceding privilege

```
                                    Access privileges
 Schema |           Name           |   Type   |     Access privileges     | Column privileges | Policies
--------+--------------------------+----------+---------------------------+-------------------+----------
public  | actor                    | table    | postgres=arwdDxt/postgres+|                   |
        |                          |          | gaurav=arwdDxt/postgres   |                   |
```

--ACCESS TABLES

REVOKE ALL ON ALL TABLES **IN SCHEMA** public **FROM** PUBLIC ;
GRANT SELECT ON ALL TABLES **IN SCHEMA** public **TO** read_only ;
GRANT SELECT, **INSERT**, **UPDATE**, **DELETE ON ALL** TABLES **IN SCHEMA** public **TO** read_write ;
GRANT ALL ON ALL TABLES **IN SCHEMA** public **TO ADMIN** ;

Create Read Only User :-

CREATE USER readonly **WITH ENCRYPTED PASSWORD** 'yourpassword' ;
GRANT CONNECT **ON DATABASE** <database_name > **TO** readonly;
GRANT USAGE **ON SCHEMA** public **TO** readonly;
GRANT SELECT ON ALL SEQUENCES **IN SCHEMA** public **TO** readonly;
GRANT SELECT ON ALL TABLES **IN SCHEMA** public **TO** readonly;

Grant access privileges on objects created in the future :-

ALTER DEFAULT PRIVILEGES IN SCHEMA myschema **GRANT SELECT ON** TABLES **TO** read_only;

ALTER DEFAULT PRIVILEGES IN SCHEMA myschema **GRANT SELECT**,**INSERT**,**DELETE**,**UPDATE ON** TABLES **TO** read_write;

ALTER DEFAULT PRIVILEGES IN SCHEMA myschema **GRANT ALL ON** TABLES **TO ADMIN**;

Or, you can set access privileges on objects created in the future by specified user.

ALTER DEFAULT PRIVILEGES FOR ROLE ADMIN GRANT SELECT ON TABLES **TO** read_only;