Postgres

1. Task requirement: Podman.PostgresSQL

2. Environment details:

- Os:- Ubuntu 22.04.3 LTS
- Podman- 3.4.4
- Psql (PostgreSQL) 14.9

3. System configuration:

- CPU - 4
- Storage -16 GB

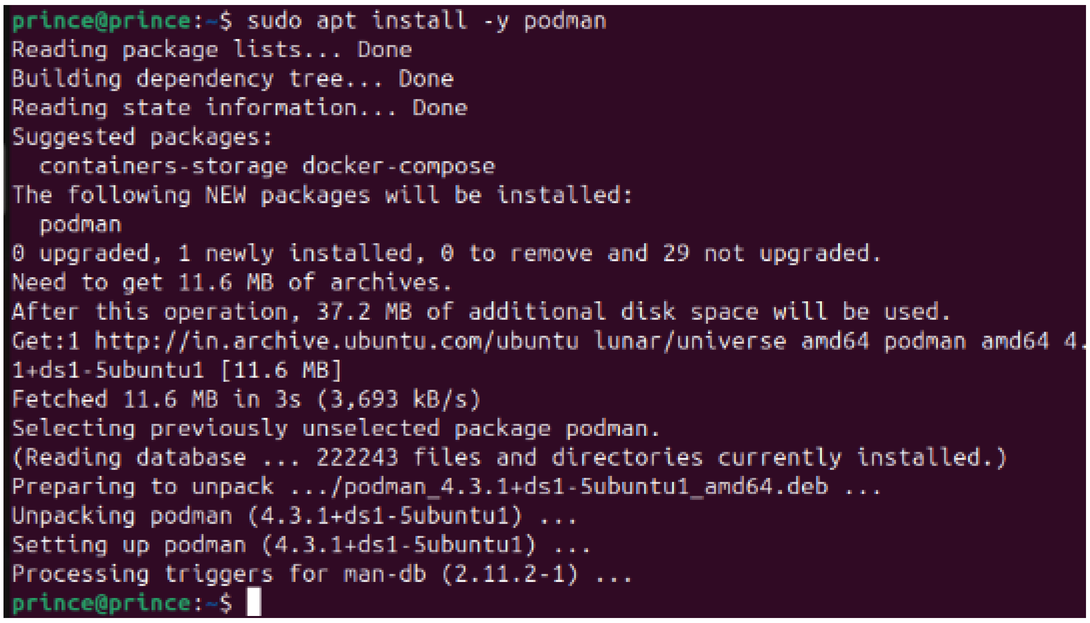
4. Definition of tools:

- Podman is an open-source container management tool used to create, run, and manage containers on Linux systems.

- PostgreSQL ek open-source relational database management system (RDBMS) hai jo data storage aur management ke liye istemal hota hai.

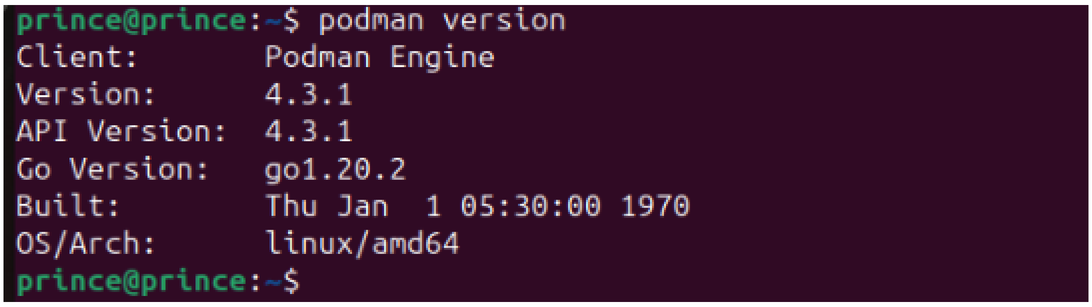
1. Create a postgres instance of postgres 14 with volume mounted.

> sudo apt install -y podman



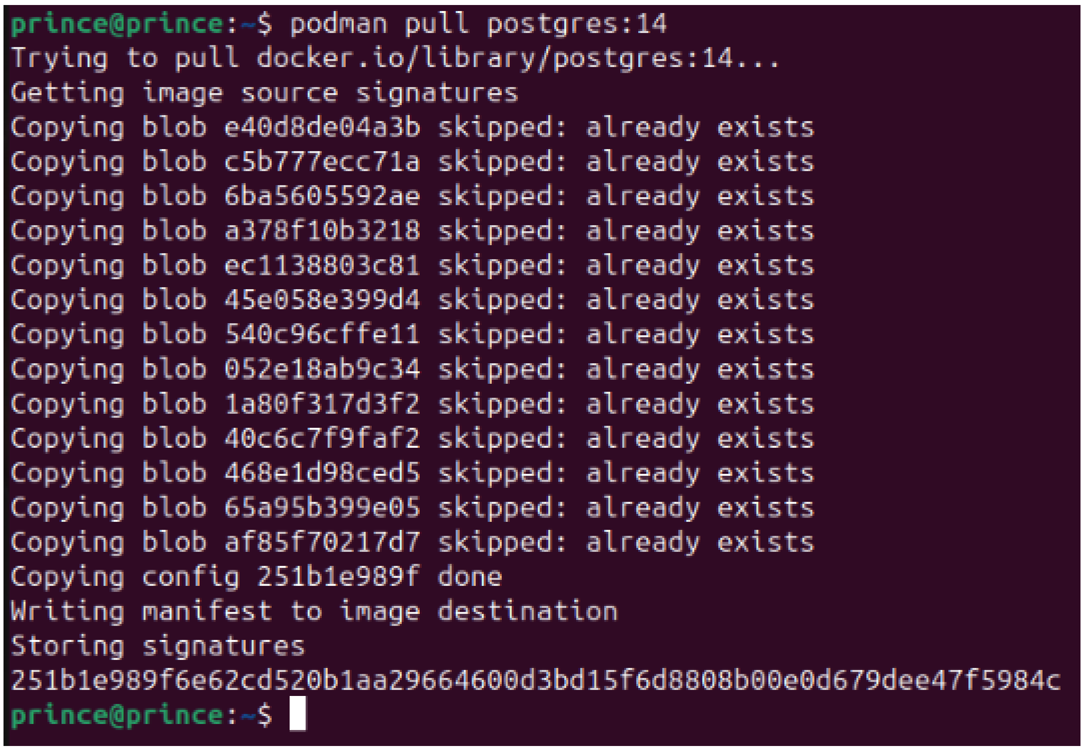
```
prince@prince:~$ sudo apt install -y podman
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  containers-storage docker-compose
The following NEW packages will be installed:
  podman
0 upgraded, 1 newly installed, 0 to remove and 29 not upgraded.
Need to get 11.6 MB of archives.
After this operation, 37.2 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu lunar/universe amd64 podman amd64 4.
1+ds1-5ubuntu1 [11.6 MB]
Fetched 11.6 MB in 3s (3,693 kB/s)
Selecting previously unselected package podman.
(Reading database ... 222243 files and directories currently installed.)
Preparing to unpack .../podman_4.3.1+ds1-5ubuntu1_amd64.deb ...
Unpacking podman (4.3.1+ds1-5ubuntu1) ...
Setting up podman (4.3.1+ds1-5ubuntu1) ...
Processing triggers for man-db (2.11.2-1) ...
prince@prince:~$
```

> podman version



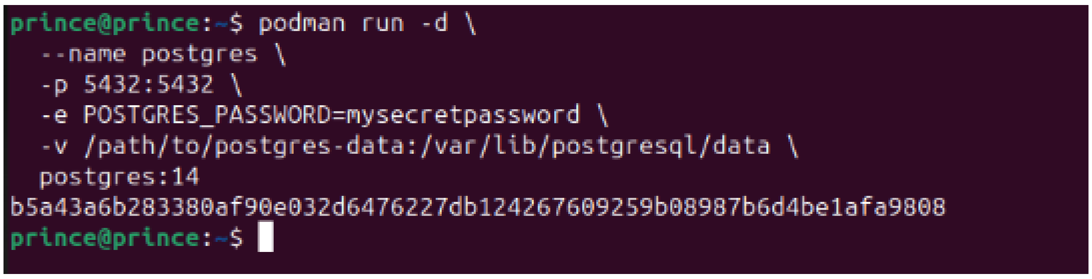
```
prince@prince:~$ podman version
Client:       Podman Engine
Version:      4.3.1
API Version:  4.3.1
Go Version:   go1.20.2
Built:        Thu Jan  1 05:30:00 1970
OS/Arch:      linux/amd64
prince@prince:~$
```

> sudo mkdir -p /home/prince/postgres-data



```
prince@prince:~$ podman pull postgres:14
Trying to pull docker.io/library/postgres:14...
Getting image source signatures
Copying blob e40d8de04a3b skipped: already exists
Copying blob c5b777ecc71a skipped: already exists
Copying blob 6ba5605592ae skipped: already exists
Copying blob a378f10b3218 skipped: already exists
Copying blob ec1138803c81 skipped: already exists
Copying blob 45e058e399d4 skipped: already exists
Copying blob 540c96cffe11 skipped: already exists
Copying blob 052e18ab9c34 skipped: already exists
Copying blob 1a80f317d3f2 skipped: already exists
Copying blob 40c6c7f9faf2 skipped: already exists
Copying blob 468e1d98ced5 skipped: already exists
Copying blob 65a95b399e05 skipped: already exists
Copying blob af85f70217d7 skipped: already exists
Copying config 251b1e989f done
Writing manifest to image destination
Storing signatures
251b1e989f6e62cd520b1aa29664600d3bd15f6d8808b00e0d679dee47f5984c
prince@prince:~$
```

> podman run --name postgres -d -p 5432:5432 -e POSTGRES_PASSWORD=mysecretpassword -v postgres-data:/var/lib/postgresql/data postgres:14



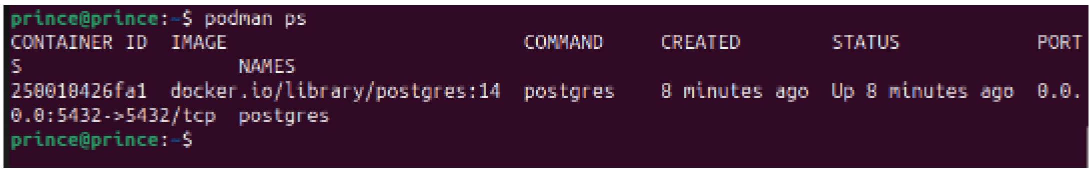
```
prince@prince:~$ podman run -d \
  --name postgres \
  -p 5432:5432 \
  -e POSTGRES_PASSWORD=mysecretpassword \
  -v /path/to/postgres-data:/var/lib/postgresql/data \
  postgres:14
b5a43a6b283380af90e032d6476227db124267609259b08987b6d4be1afa9808
prince@prince:~$
```

- podman run: Ye command ek container ko create aur run karne ke liye use hoti hai.
- -d: Ye option container ko background mode me run karne ke liye use hota hai, matlab container run hoga aur aapka terminal available rahega.
- --name my-postgres: Is option se aap apne container ko "my-postgres" naam se identify kar sakte hain.
- -e POSTGRES_PASSWORD=mysecretpassword: Is option se aap environment variable set kar rahe hain, jiska naam "POSTGRES_PASSWORD" hai aur value "mysecretpassword" hai. Ye password PostgreSQL database access ke liye use hoga.
- -v /mydata:/var/lib/postgresql/data: Is option se aap ek volume mount kar rahe hain. /mydata host system ke path ko /var/lib/postgresql/data container ke path se map karega. Isse aap apne database

  data ko host system me store kar sakte hain, taki wo persist kare, aur aap container ko delete karne ke baad bhi data safe rahe.
- 'postgres:14: Ye image name hai, jise aap container se use kar rahe hain. "postgres:14" PostgreSQL ke version 14 ke official Docker image ko represent karta hai. Isse container PostgreSQL database server ke sath run karega.

Verify that the PostgreSQL container is running:

> podman ps



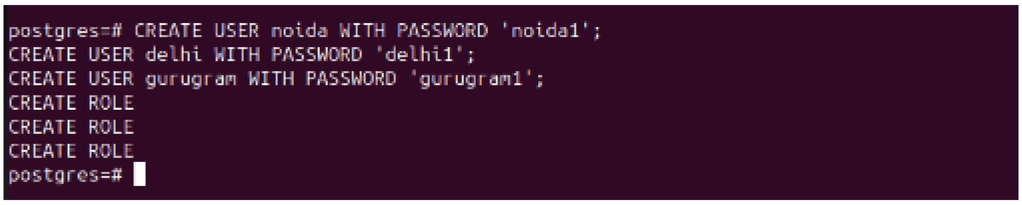
```
prince@prince:~$ podman ps
CONTAINER ID  IMAGE                          COMMAND     CREATED         STATUS             PORT
S             NAMES
250010426fa1  docker.io/library/postgres:14  postgres    8 minutes ago   Up 8 minutes ago   0.0.
0.0:5432->5432/tcp  postgres
prince@prince:~$
```

2.create users,databases,tables,extensions on the same.

> podman exec -it postgres psql -U postgres

(a) Create Users

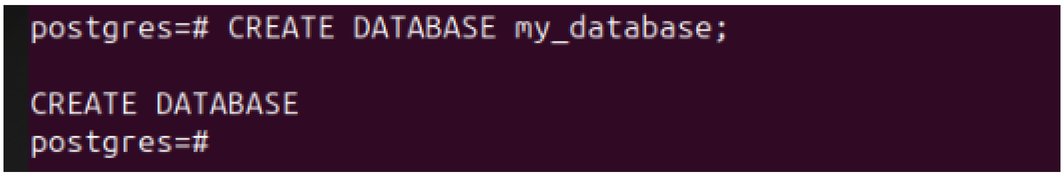
> CREATE USER noida WITH PASSWORD 'noida1'; CREATE USER delhi WITH PASSWORD 'delhi1'; CREATE USER gurugram WITH PASSWORD 'gurugram1'; CREATE ROLE CREATE ROLE CREATE ROLE



```
postgres=# CREATE USER noida WITH PASSWORD 'noida1';
CREATE USER delhi WITH PASSWORD 'delhi1';
CREATE USER gurugram WITH PASSWORD 'gurugram1';
CREATE ROLE
CREATE ROLE
CREATE ROLE
postgres=#
```

(b) Databases

> postgres=# CREATE DATABASE my_database;



```
postgres=# CREATE DATABASE my_database;

CREATE DATABASE
postgres=#
```

(c) Tables

> postgres=# \l (List of databases)

> postgres=# \c (connected to database)

> CREATE TABLE my_table (id SERIAL NOT NULL PRIMARY KEY, name VARCHAR(255) NOT NULL);

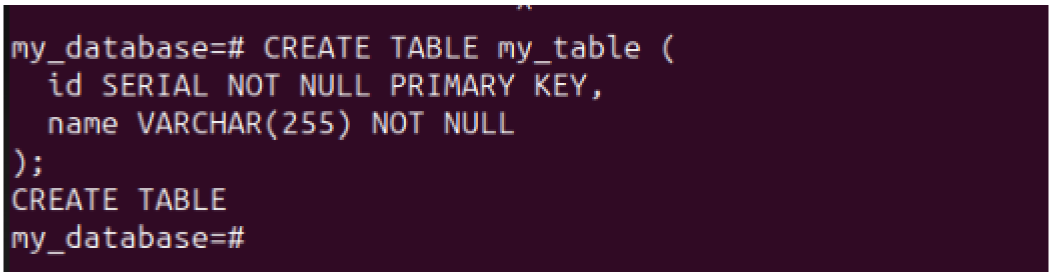
- CREATE TABLE: This keyword tells the database to create a new table.
- my_table: This is the name of the table that is being created.

- (id SERIAL NOT NULL PRIMARY KEY, name VARCHAR(255) NOT NULL): This is the definition of the table, which includes the names and data types of the columns in the table.

Here is a more detailed explanation of each part of the statement:

- id SERIAL NOT NULL PRIMARY KEY: This column will store the unique identifier for each row in the table. The SERIAL data type means that the database will automatically generate a unique integer value for each new row that is inserted into the table. The NOT NULL constraint means that this column cannot be empty. The PRIMARY KEY constraint means that this column uniquely identifies each row in the table.

- name VARCHAR(255) NOT NULL: This column will store the name of each row in the table. The VARCHAR(255) data type means that this column can store up to 255 characters of text. The NOT NULL constraint means that this column cannot be empty.

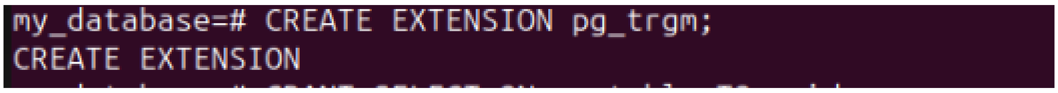


```
my_database=# CREATE TABLE my_table (
  id SERIAL NOT NULL PRIMARY KEY,
  name VARCHAR(255) NOT NULL
);
CREATE TABLE
my_database=#
```

(C) extensions

> CREATE EXTENSION pg_trgm;

> CREATE EXTENSION



```
my_database=# CREATE EXTENSION pg_trgm;
CREATE EXTENSION
```

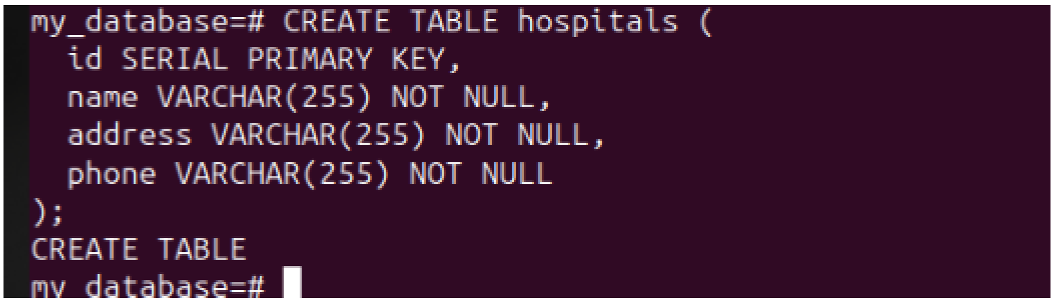
EXTENSION Ye SQL statement PostgreSQL database mein ek extension ko create karne ya activate karne ke liye istemal hota hai. Extension ek prakar ke additional modules ya functions hote hain jo PostgreSQL database functionality ko extend karte hain.

pg_trgm Ye extension PostgreSQL mein full-text search aur trigram similarity capabilities provide karta hai. Full-text search ka use karke, aap apne data mein text ko search kar sakte hain. Trigram similarity ka use karke, aap apne data mein text ke similarity ko calculate kar sakte hain. In capabilities ka use karne ke kai liye hai. For example, aap ine capabilities ka use karke ek website mein contents ko search kar sakte hain, ek database mein data ko search kar sakte hain, ya ek document collection mein documents ko search kar sakte hain.

3.Perform crud operations.

CRUD (Create, Read, Update, Delete)

(a)Create

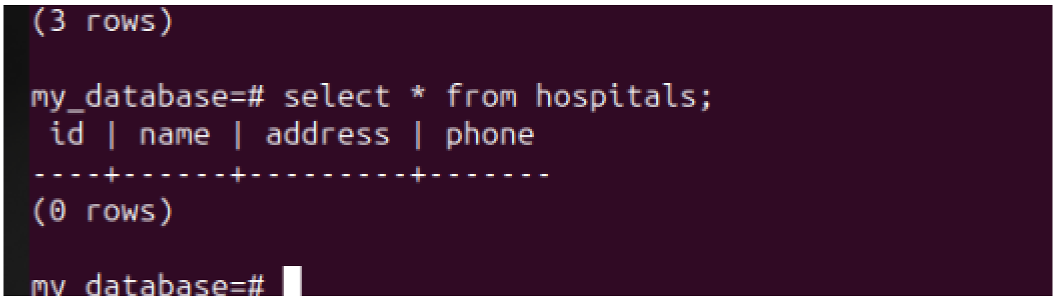


```
my_database=# CREATE TABLE hospitals (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  address VARCHAR(255) NOT NULL,
  phone VARCHAR(255) NOT NULL
);
CREATE TABLE
my_database=#
```

- id: A serial column that is the primary key of the table. This means that each row in the table will have a unique id value.
- name: A VARCHAR column that stores the name of the hospital.
- address: A VARCHAR column that stores the address of the hospital.
- phone: A VARCHAR column that stores the phone number of the hospital.
- NOT NULL: constraint on all of the columns means that each column must have a value. No rows will be inserted into the table if any of the columns are empty.
- VARCHAR : is a variable-length string data type in SQL. It means that it can store strings of any length, up to the maximum length specified when the column is created. It is a good choice for storing strings of variable length, such as names, addresses, and phone numbers.

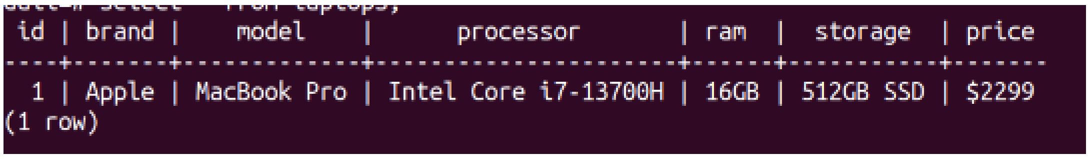
(b) Read

> my_database=# select * from hospitals;



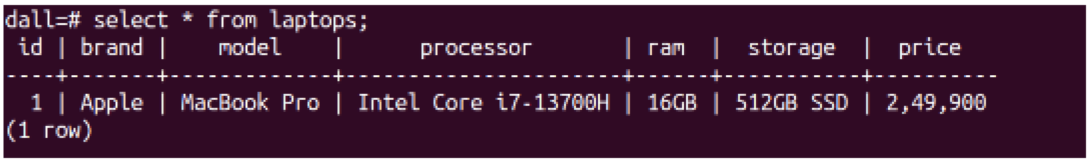
```
(3 rows)

my_database=# select * from hospitals;
 id | name | address | phone
----+------+---------+-------
(0 rows)

my_database=#
```

(C) Update



```
 id | brand |    model     |      processor       | ram  |   storage   | price
----+-------+--------------+----------------------+------+-------------+-------
  1 | Apple | MacBook Pro  | Intel Core i7-13700H | 16GB | 512GB SSD   | $2299
(1 row)
```

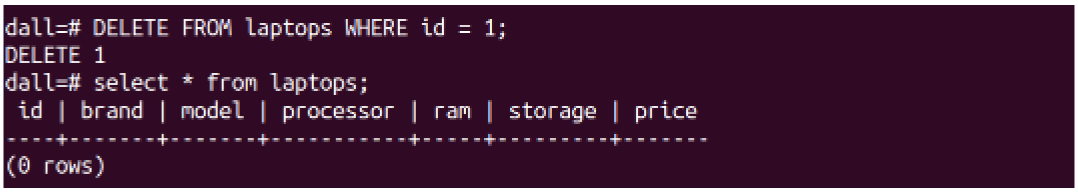
> UPDATE laptops SET price = '2,49,900' WHERE id = 1;



```
dall=# select * from laptops;
 id | brand |    model    |      processor       | ram  |  storage  |  price
----+-------+-------------+----------------------+------+-----------+----------
  1 | Apple | MacBook Pro | Intel Core i7-13700H | 16GB | 512GB SSD | 2,49,900
(1 row)
```

(D) Delete

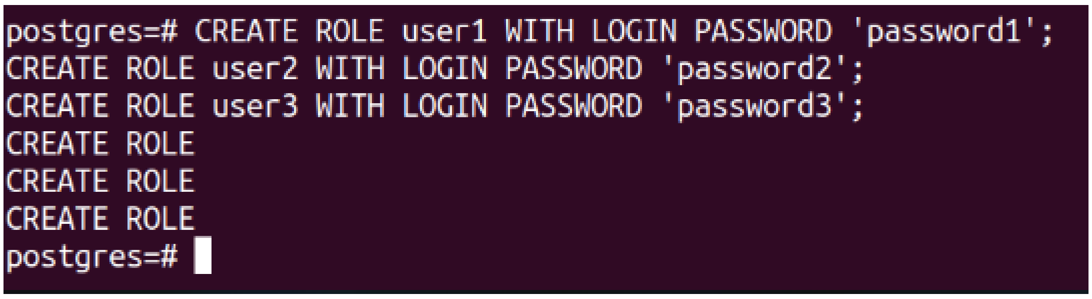
> dall=# DELETE FROM laptops WHERE id = 1;



```
dall=# DELETE FROM laptops WHERE id = 1;
DELETE 1
dall=# select * from laptops;
 id | brand | model | processor | ram | storage | price
----+-------+-------+-----------+-----+---------+-------
(0 rows)
```

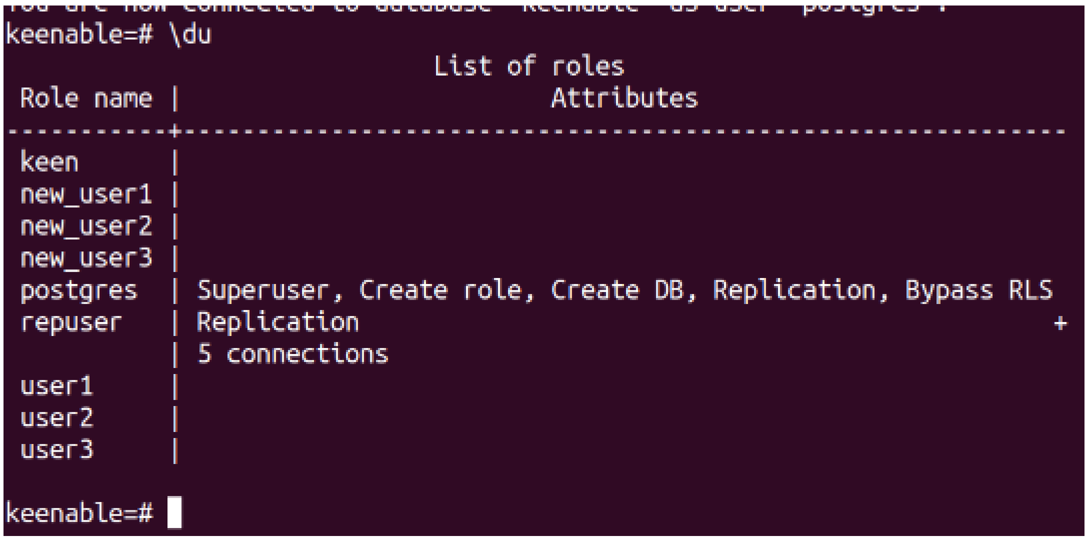
4. Create three users with a password.

> CREATE ROLE user 1 WITH LOGIN PASSWORD 'password1'; CREATE ROLE user 2 WITH LOGIN PASSWORD 'password2'; CREATE ROLE user 3WITH LOGIN PASSWORD 'password3';



```
postgres=# CREATE ROLE user1 WITH LOGIN PASSWORD 'password1';
CREATE ROLE user2 WITH LOGIN PASSWORD 'password2';
CREATE ROLE user3 WITH LOGIN PASSWORD 'password3';
CREATE ROLE
CREATE ROLE
CREATE ROLE
postgres=#
```

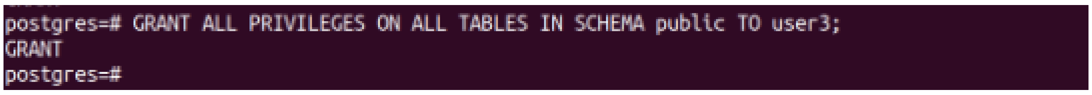
> \du (User show)



```
keenable=# \du
                              List of roles
 Role name |                         Attributes
-----------+------------------------------------------------------------
 keen      |
 new_user1 |
 new_user2 |
 new_user3 |
 postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS
 repuser   | Replication                                               +
           | 5 connections
 user1     |
 user2     |
 user3     |

keenable=#
```

5. Grant select permission for user1,select,insert,delete for user2 and all for user3.

> GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO user3;



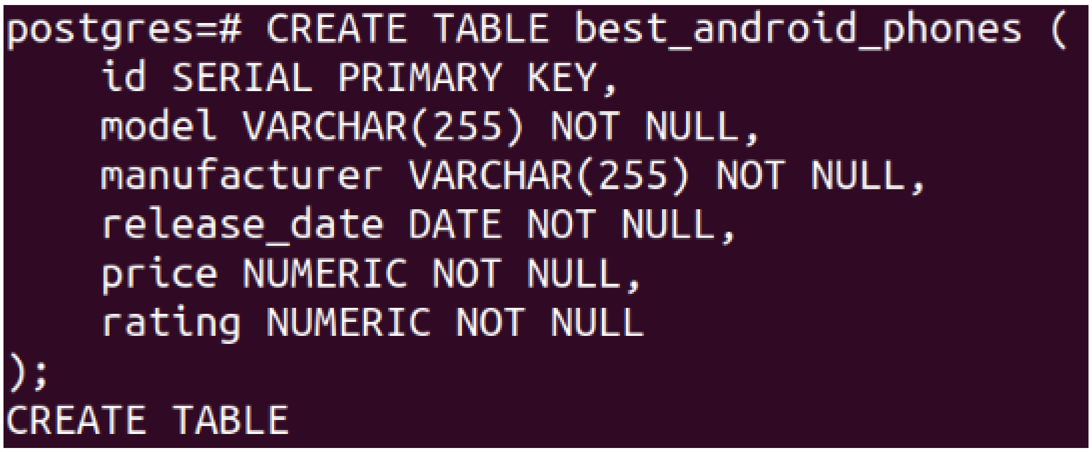
```
postgres=# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO user3;
GRANT
postgres=#
```

6. Create another table and explain about the different joins.

Different types of joins:

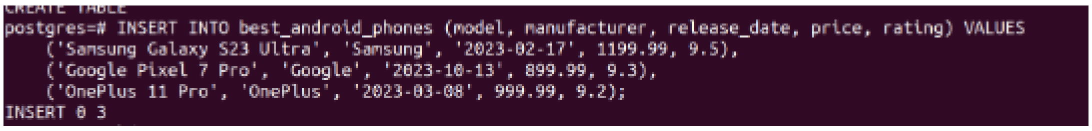
- Inner join: Returns all rows from both tables where the join condition is met.

> CREATE TABLE best_android_phones



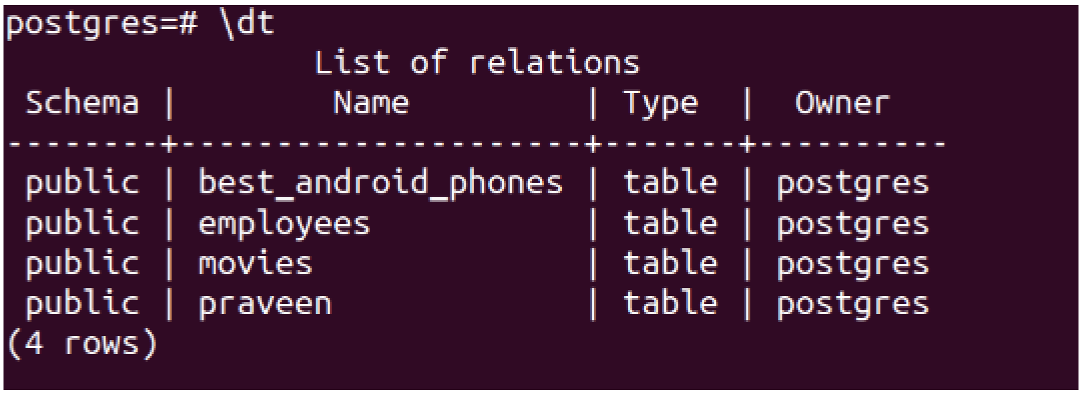
```
postgres=# CREATE TABLE best_android_phones (
    id SERIAL PRIMARY KEY,
    model VARCHAR(255) NOT NULL,
    manufacturer VARCHAR(255) NOT NULL,
    release_date DATE NOT NULL,
    price NUMERIC NOT NULL,
    rating NUMERIC NOT NULL
);
CREATE TABLE
```

> INSERT INTO best_android_phones



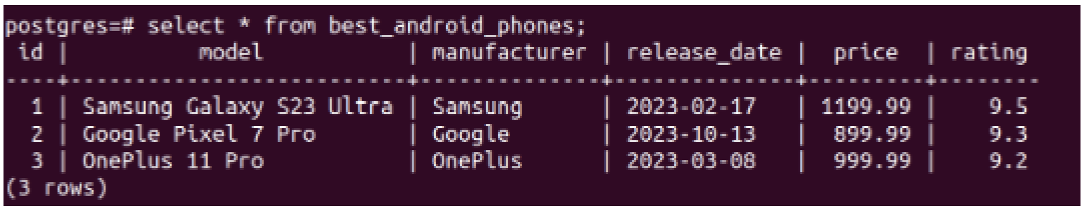
```
CREATE TABLE
postgres=# INSERT INTO best_android_phones (model, manufacturer, release_date, price, rating) VALUES
    ('Samsung Galaxy S23 Ultra', 'Samsung', '2023-02-17', 1199.99, 9.5),
    ('Google Pixel 7 Pro', 'Google', '2023-10-13', 899.99, 9.3),
    ('OnePlus 11 Pro', 'OnePlus', '2023-03-08', 999.99, 9.2);
INSERT 0 3
```

> \dt



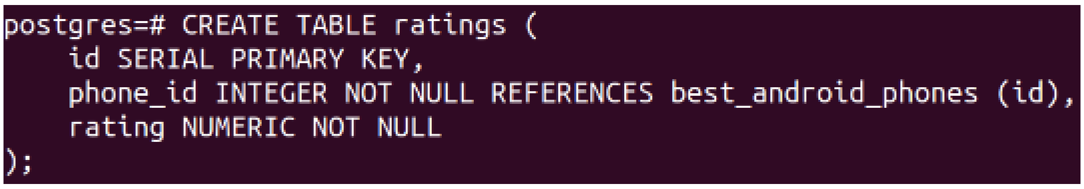
```
postgres=# \dt
                List of relations
 Schema |        Name         | Type  |  Owner
--------+---------------------+-------+----------
 public | best_android_phones | table | postgres
 public | employees           | table | postgres
 public | movies              | table | postgres
 public | praveen             | table | postgres
(4 rows)
```

> select * from best_android_phones;



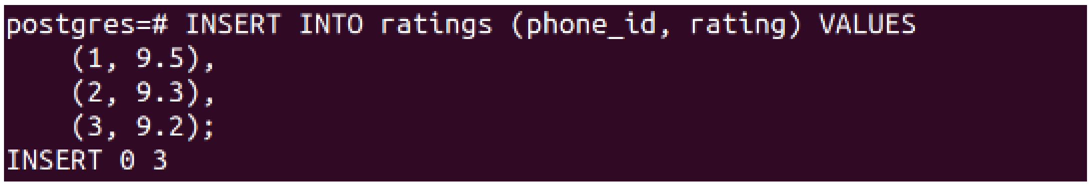
```
postgres=# select * from best_android_phones;
 id |          model           | manufacturer | release_date |  price  | rating
----+--------------------------+--------------+--------------+---------+--------
  1 | Samsung Galaxy S23 Ultra | Samsung      | 2023-02-17   | 1199.99 |    9.5
  2 | Google Pixel 7 Pro       | Google       | 2023-10-13   |  899.99 |    9.3
  3 | OnePlus 11 Pro           | OnePlus      | 2023-03-08   |  999.99 |    9.2
(3 rows)
```

> CREATE TABLE ratings



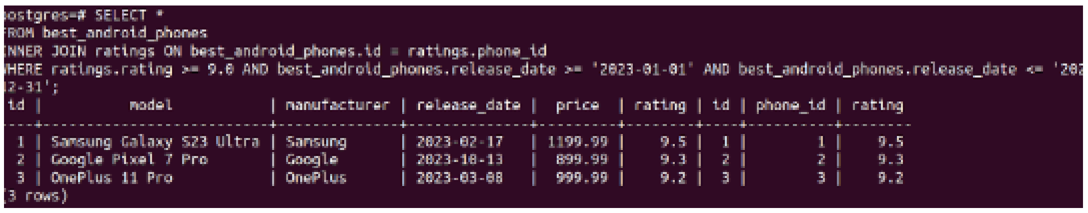
```
postgres=# CREATE TABLE ratings (
    id SERIAL PRIMARY KEY,
    phone_id INTEGER NOT NULL REFERENCES best_android_phones (id),
    rating NUMERIC NOT NULL
);
```

> INSERT INTO ratings



```
postgres=# INSERT INTO ratings (phone_id, rating) VALUES
    (1, 9.5),
    (2, 9.3),
    (3, 9.2);
INSERT 0 3
```

> INNER JOIN

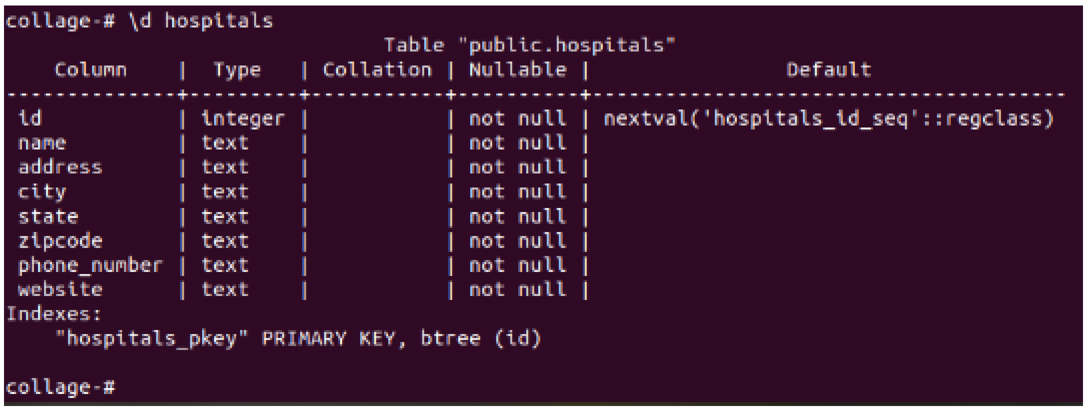


```
ostgres=# SELECT *
ROM best_android_phones
NNER JOIN ratings ON best_android_phones.id = ratings.phone_id
HERE ratings.rating >= 9.0 AND best_android_phones.release_date >= '2023-01-01' AND best_android_phones.release_date <= '202
2-31';
 id |          model           | manufacturer | release_date |  price  | rating | id | phone_id | rating
----+--------------------------+--------------+--------------+---------+--------+----+----------+--------
  1 | Samsung Galaxy S23 Ultra | Samsung      | 2023-02-17   | 1199.99 |    9.5 |  1 |        1 |    9.5
  2 | Google Pixel 7 Pro       | Google       | 2023-10-13   |  899.99 |    9.3 |  2 |        2 |    9.3
  3 | OnePlus 11 Pro           | OnePlus      | 2023-03-08   |  999.99 |    9.2 |  3 |        3 |    9.2
3 rows)
```

7. Understanding the table structure,finding database size, table size etc.

(a) Table structure

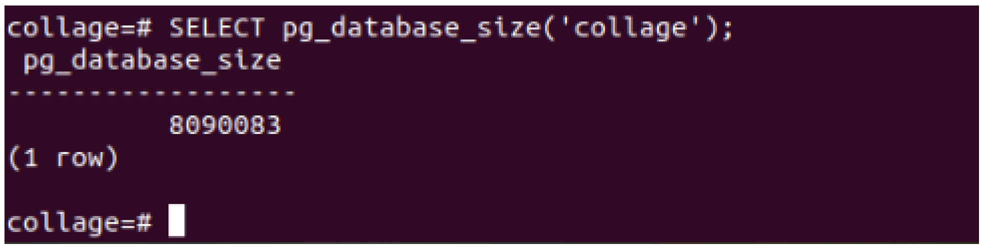
> (\d table name)



```
collage-# \d hospitals
                          Table "public.hospitals"
    Column    |  Type   | Collation | Nullable |                Default
--------------+---------+-----------+----------+---------------------------------------
 id           | integer |           | not null | nextval('hospitals_id_seq'::regclass)
 name         | text    |           | not null |
 address      | text    |           | not null |
 city         | text    |           | not null |
 state        | text    |           | not null |
 zipcode      | text    |           | not null |
 phone_number | text    |           | not null |
 website      | text    |           | not null |
Indexes:
    "hospitals_pkey" PRIMARY KEY, btree (id)

collage-#
```

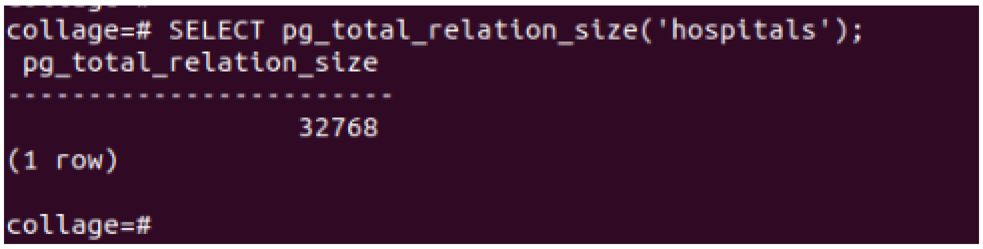
- This will list the name, data type, and other information about each column in the table.

(b) finding database size



```
collage=# SELECT pg_database_size('collage');
 pg_database_size
------------------
          8090083
(1 row)

collage=#
```

(c) table size



```
collage=# SELECT pg_total_relation_size('hospitals');
 pg_total_relation_size
------------------------
                  32768
(1 row)

collage=#
```