

# Tool Presentation

**Presented by:**

20CE009-YASH BHUVA  
20CE010-MADHAV CHAUDHARY  
20CE011-DEVANSHI CHHABHAIYA  
20CE012-SRISHU CHINTAKINDI

---

## PostgreSQL

---

### PostgreSQL

PostgreSQL is an object-relational database management system (ORDBMS) based on POSTGRES, Version 4.2, developed at the University of California at Berkeley Computer Science Department. Many technologies developed by POSTGRES were not available in commercial database systems until much later. PostgreSQL features transactions with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. It is designed to handle a range of workloads, from single machines to data warehouses or Web services with many concurrent users. It is the default database for macOS Server and is also available for Windows, Linux, FreeBSD, and OpenBSD.

The open-source PostgreSQL database is a descendant of the original Berkeley code. It implements a major portion of the SQL standard and has a number of modern features: 1. complex queries

2. foreign keys

3. triggers

4. updatable views

5. transactional integrity

6. multi-version concurrency control

And because of the liberal license, PostgreSQL can be used, modified, and distributed by anyone free of charge for any purpose, be it private, commercial, or academic.

In database jargon, PostgreSQL uses a client/server model. A PostgreSQL session consists of the following cooperating processes (programs):

- A server process, which manages the database files, accepts connections to the database from client applications, and performs database actions on behalf of the clients. The database server program is called postgres.
- The user's client (frontend) application that wants to perform database operations. Client applications can be very diverse in nature: a client could be a text-oriented tool, a graphical application, a web server that accesses the database to display web pages, or a specialized database maintenance tool. Some client applications are supplied with the PostgreSQL distribution, most are developed by users.

As is typical of client/server applications, the client and the server can be on different hosts. In that case they communicate over a TCP/IP network connection. You should keep this in mind, because the files that can be accessed on a client machine might not be accessible on the database server machine.

The PostgreSQL server can handle multiple concurrent connections from clients. To achieve this it starts ("forks") a new process for each connection. From that point on, the client and the new server process communicate without intervention by the original postgres process. Thus, the master server process is always running, waiting for client connections, whereas client and associated server processes come and go.

Traditionally, the configuration and data files used by a database cluster are stored together within the cluster's data directory, commonly referred to as PGDATA (after the name of the environment variable that can be used to define it). A common location for PGDATA is `/var/lib/pgsql/data`. Multiple clusters, managed by different server instances, can exist on the same machine.

The PGDATA directory contains several subdirectories and control files. For each database in the cluster there is a subdirectory within PGDATA/base, named after the database's OID in `pg_database`. This subdirectory is the default location for the database's files; in particular, its system catalogs are stored there.

Every table and index is stored as an array of pages of a fixed size (usually 8 kB, although a different page size can be selected when compiling the server). In a table, all the pages are logically equivalent, so a particular item (row) can be stored in any page. In indexes, the first page is generally reserved as a metapage holding control information, and there can be different types of pages within the index, depending on the index access method.

Each table and index is stored in a separate file. For ordinary relations, these files are named after the table or index's filenode number, which can be found in `pg_class.relfilenode`. But for temporary relations, the file name is of the form `tBBB_FFF`, where BBB is the backend ID of the backend which created the file, and FFF is the filenode number. In either case, in addition to the main file (a/k/a main fork), each table and index has a free space map, which stores information about free space available in the relation.

Tablespaces make the scenario more complicated. Each user-defined tablespace has a symbolic link inside the PGDATA/pg\_tblspc directory, which points to the physical tablespace directory. This symbolic link is named after the tablespace's OID. Inside the physical tablespace directory there is a subdirectory with a name that depends on the

PostgreSQL server version, such as PG\_9.0\_201008051. (The reason for using this subdirectory is so that successive versions of the database can use the same CREATE TABLESPACE location value without conflicts.) Within the version-specific subdirectory,

there is a subdirectory for each database that has elements in the tablespace, named after the database's OID. Tables and indexes are stored within that directory, using the filenode naming scheme. The pg\_default tablespace is not accessed through pg\_tblspc, but corresponds to PGDATA/base. Similarly, the pg\_global tablespace is not accessed through pg\_tblspc, but corresponds to PGDATA/global.

When a table or index exceeds 1 GB, it is divided into gigabyte-sized segments. The first segment's file name is the same as the filenode; subsequent segments are named filenode.1, filenode.2, etc. This arrangement avoids problems on platforms that have file size limitations. (Actually, 1 GB is just the default segment size. The segment size can be adjusted using the configuration option—`with-segsize` when building PostgreSQL.) In principle, free space map and visibility map forks could require multiple segments as well, though this is unlikely to happen in practice.

---

**Now, we'll go through how to install and configure PostgreSQL on Windows 10.**

We can install PostgreSQL on the following operating systems:










































1. Windows
2. Linux
3. Mac OS Server
4. Free BSD and Open BSD

There are three steps to complete the PostgreSQL installation:

1. Download PostgreSQL installer for Windows
2. Install PostgreSQL
3. Verify the installation

## **1) Download PostgreSQL Installer for Windows**

To get started, go to the Enterprise DB's PostgreSQL installers download page.

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
14.2	<a href="https://www.postgresql.org/">postgresql.org</a> 	<a href="https://www.postgresql.org/">postgresql.org</a> 			Not supported
13.6	<a href="https://www.postgresql.org/">postgresql.org</a> 	<a href="https://www.postgresql.org/">postgresql.org</a> 			Not supported
12.10	<a href="https://www.postgresql.org/">postgresql.org</a> 	<a href="https://www.postgresql.org/">postgresql.org</a> 			Not supported
11.15	<a href="https://www.postgresql.org/">postgresql.org</a> 	<a href="https://www.postgresql.org/">postgresql.org</a> 			Not supported
10.20					
9.6.24*					
9.5.25*					
9.4.26*					
9.3.25*					

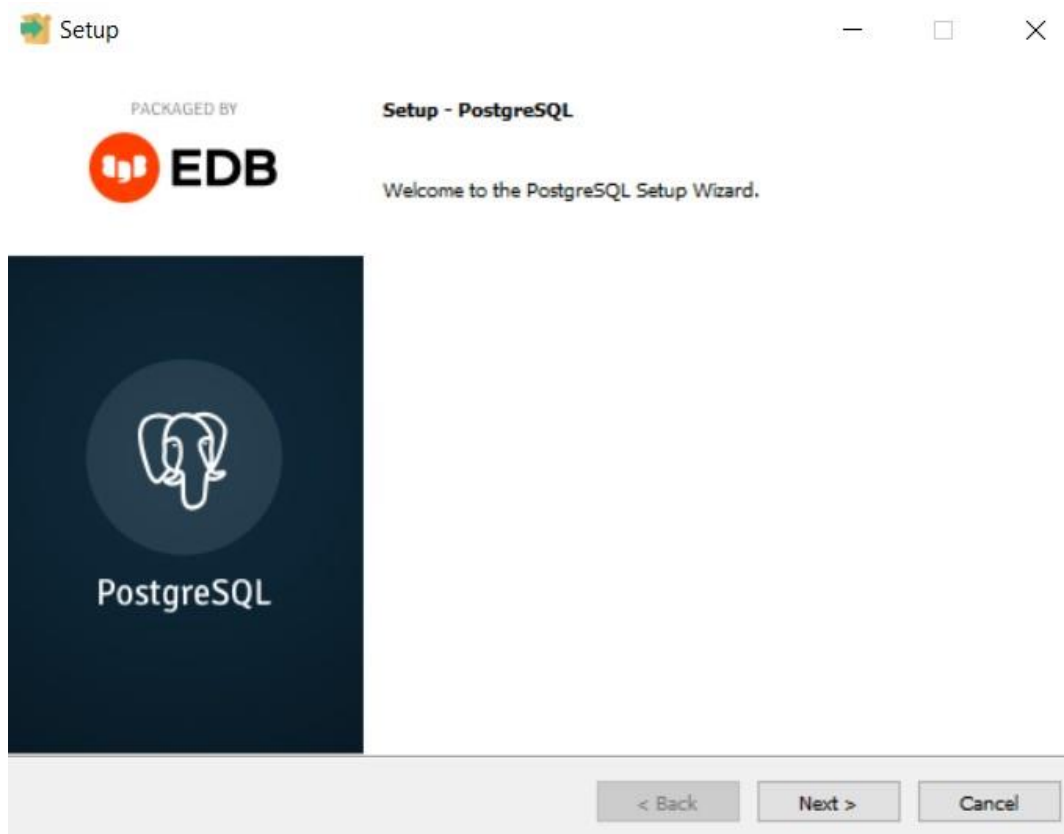
It will take a few minutes to complete the download.

## 2) Install PostgreSQL on Window step by step

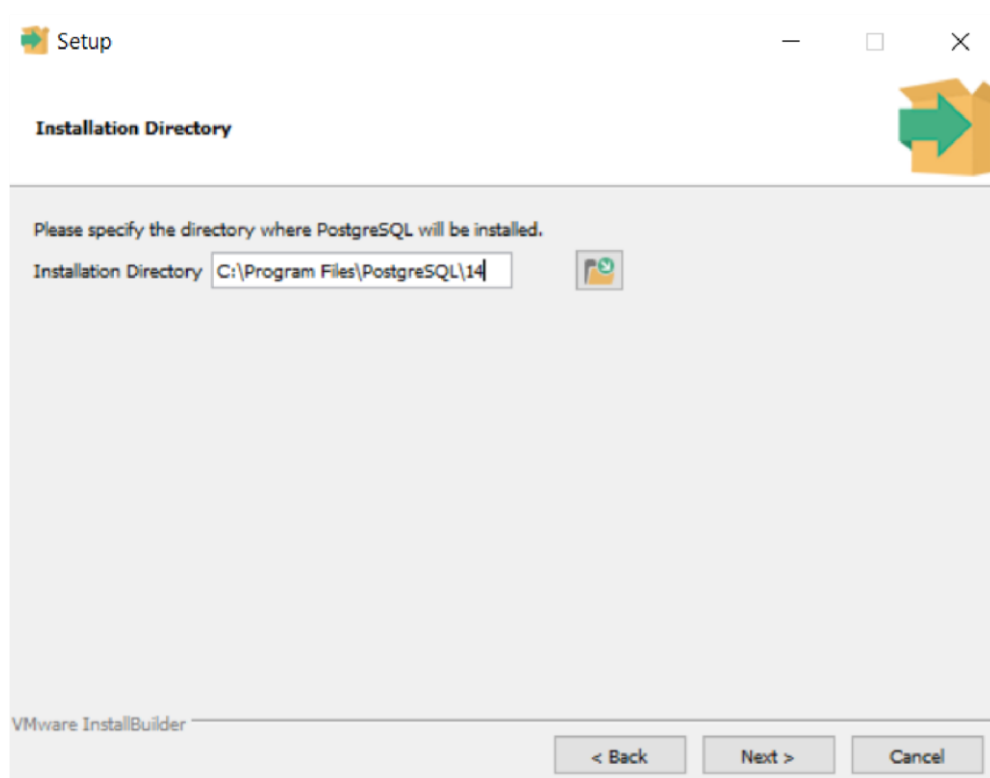
To install PostgreSQL on Windows, you need to have administrator privileges.

Step 1. Double click on the installer file.

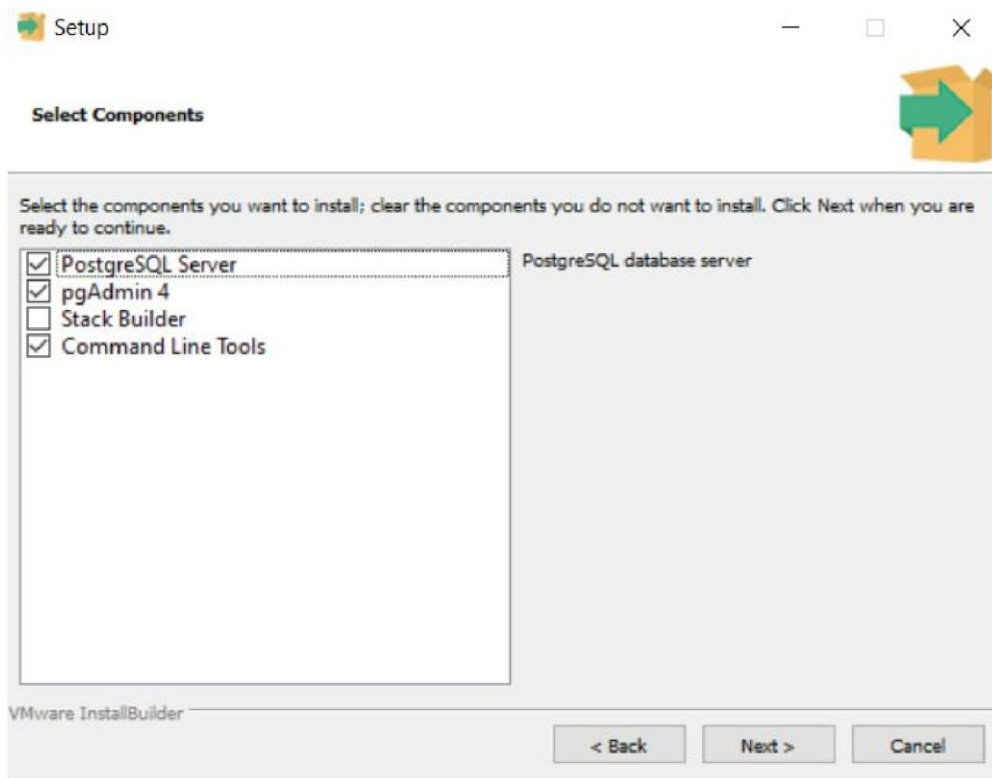
Step 2. Click the Next button.



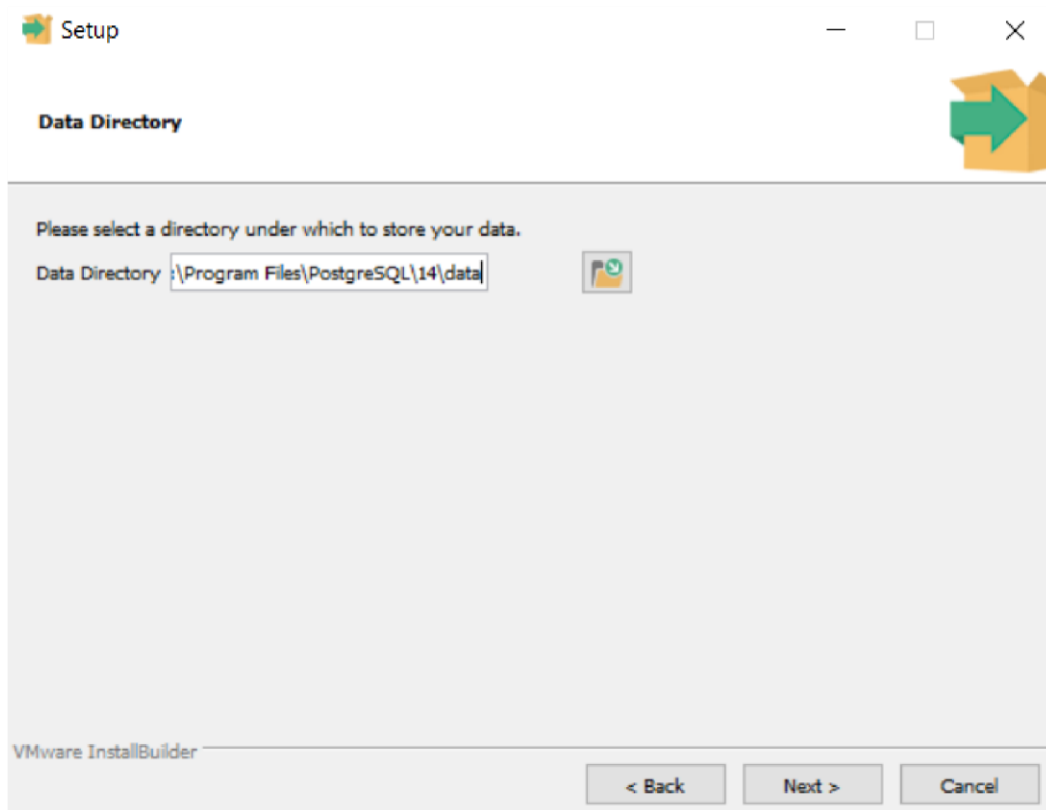
Step 3. Select an installation folder, either your own or the one suggested by the PostgreSQL installer, and then click the Next button.



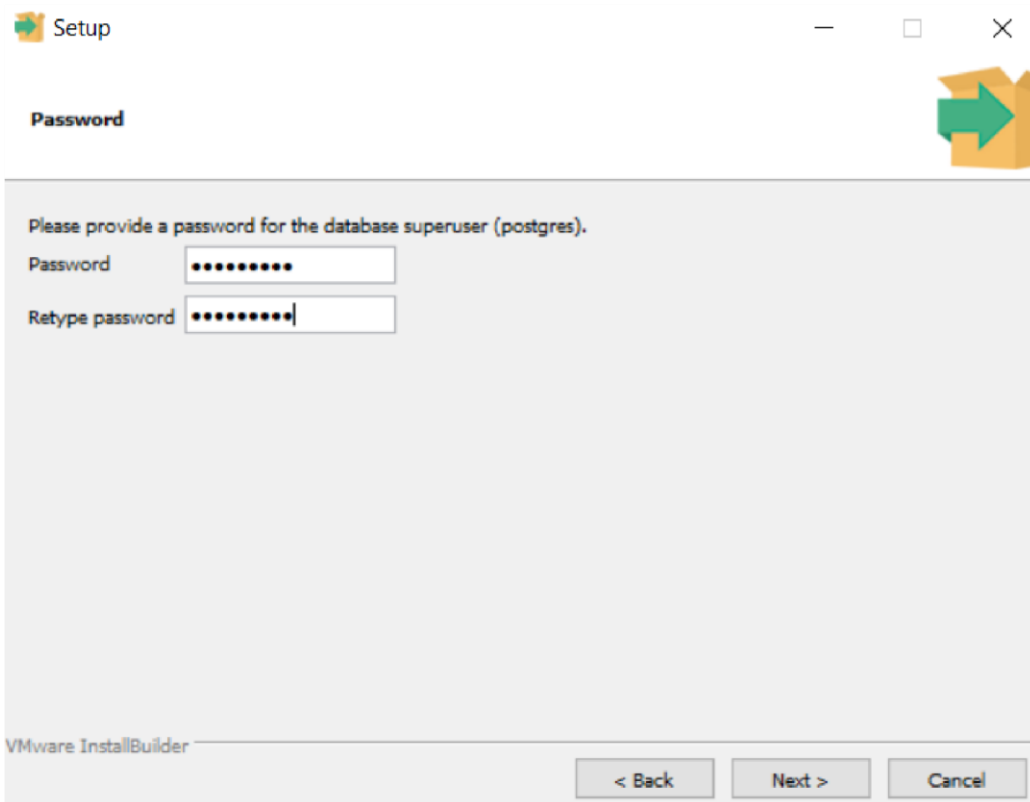
Step 4. Select software components to install.



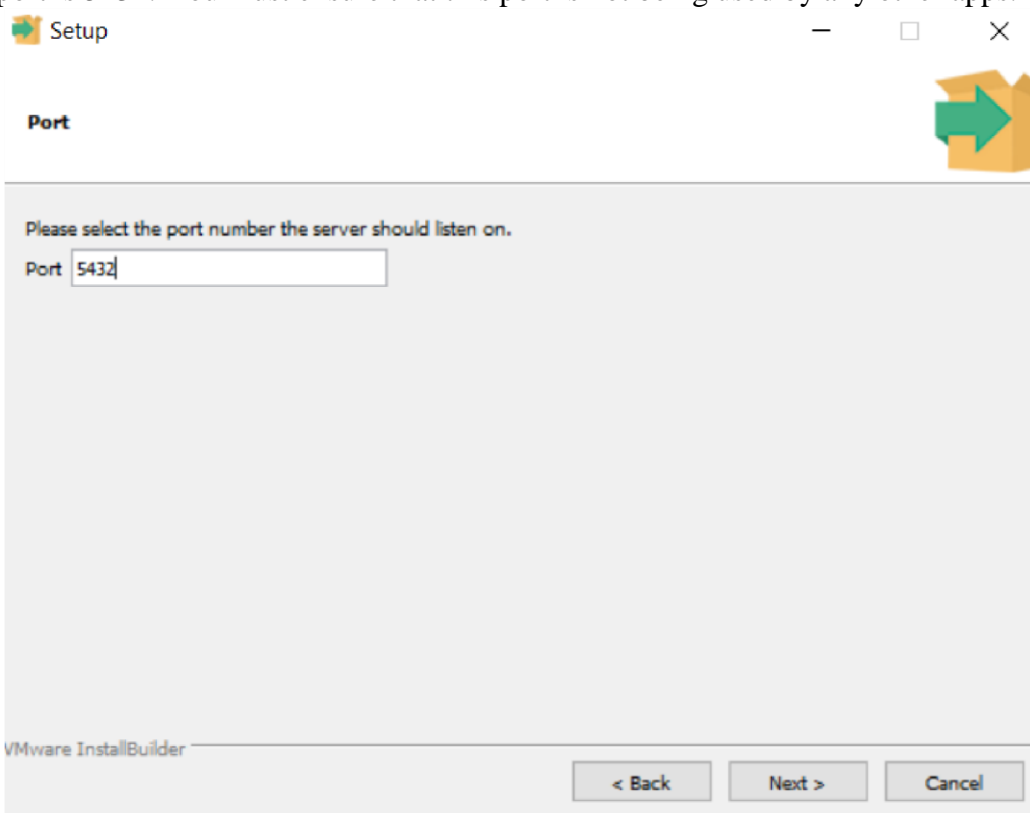
Step 5. Select the database directory to store the data and click Next button to go to the next step.



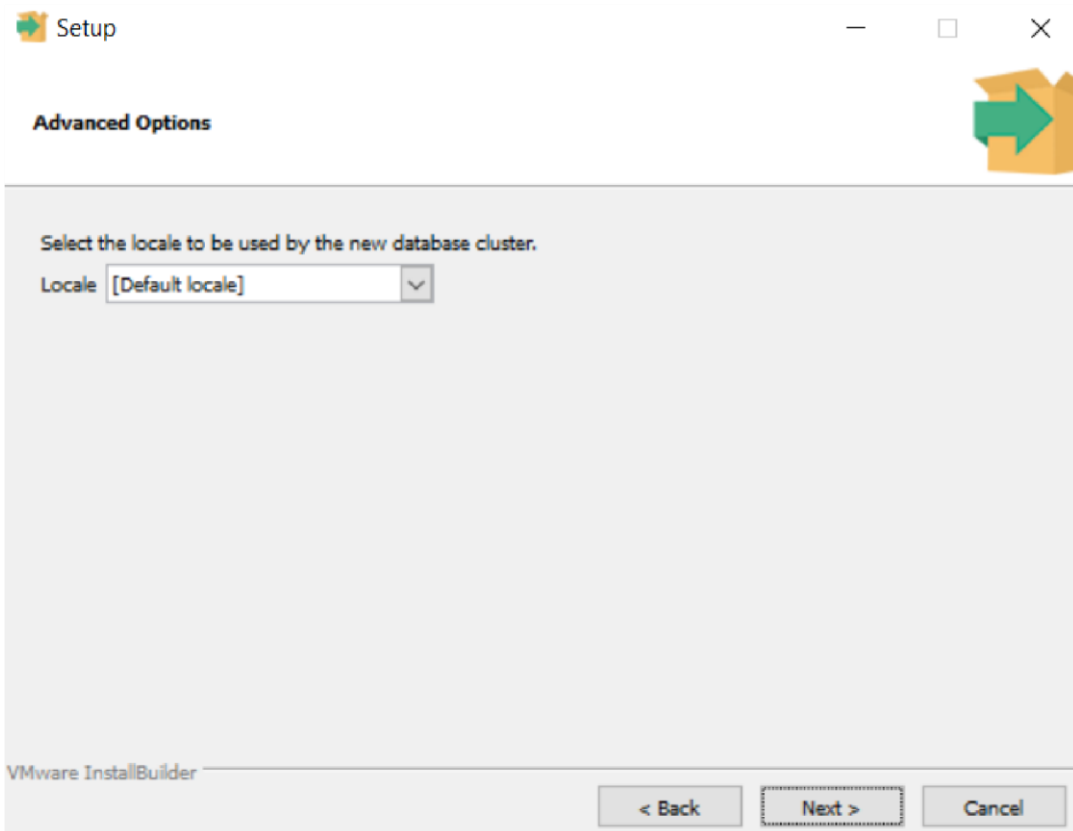
Step 6. Enter the password for the database superuser (postgres). You must retype the password to confirm it before moving on to the next step.



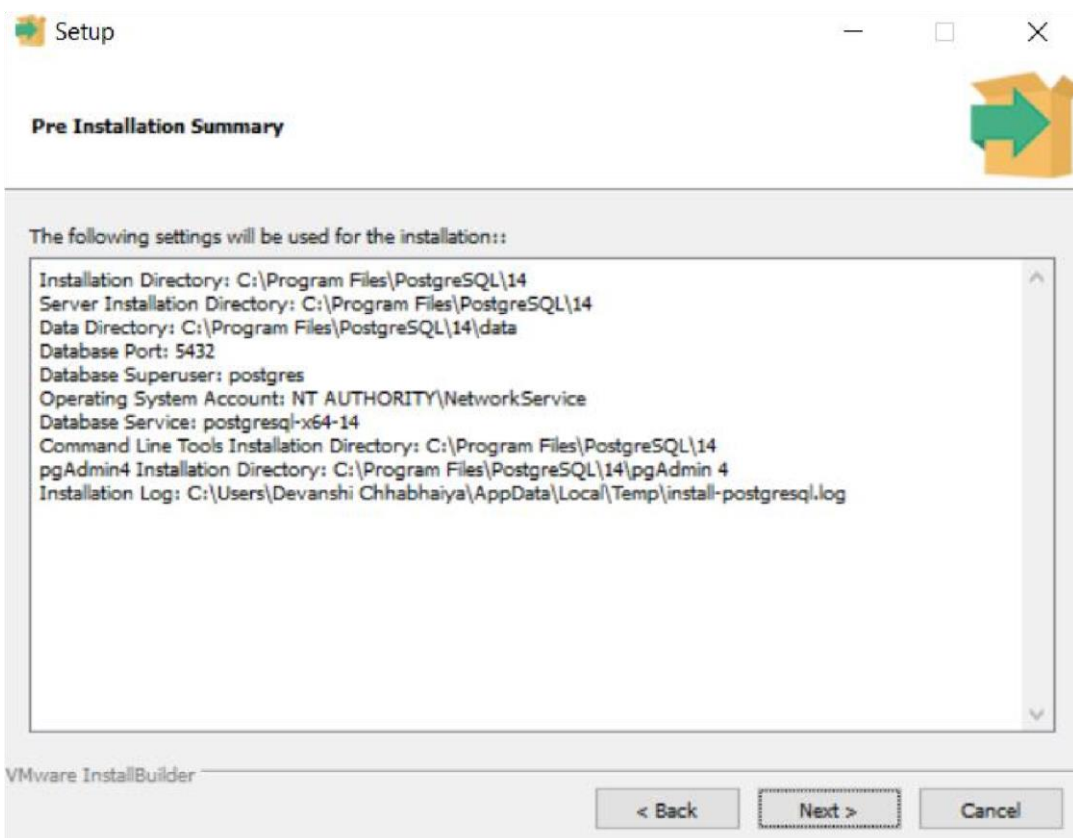
Step 7. Select a port for the PostgreSQL database server to listen on. PostgreSQL's default port is 5432. You must ensure that this port is not being used by any other apps.



Step 8. Choose the default locale used by the PostgreSQL database. PostgreSQL will use the operating system locale if you leave it as default. After that click the Next button.

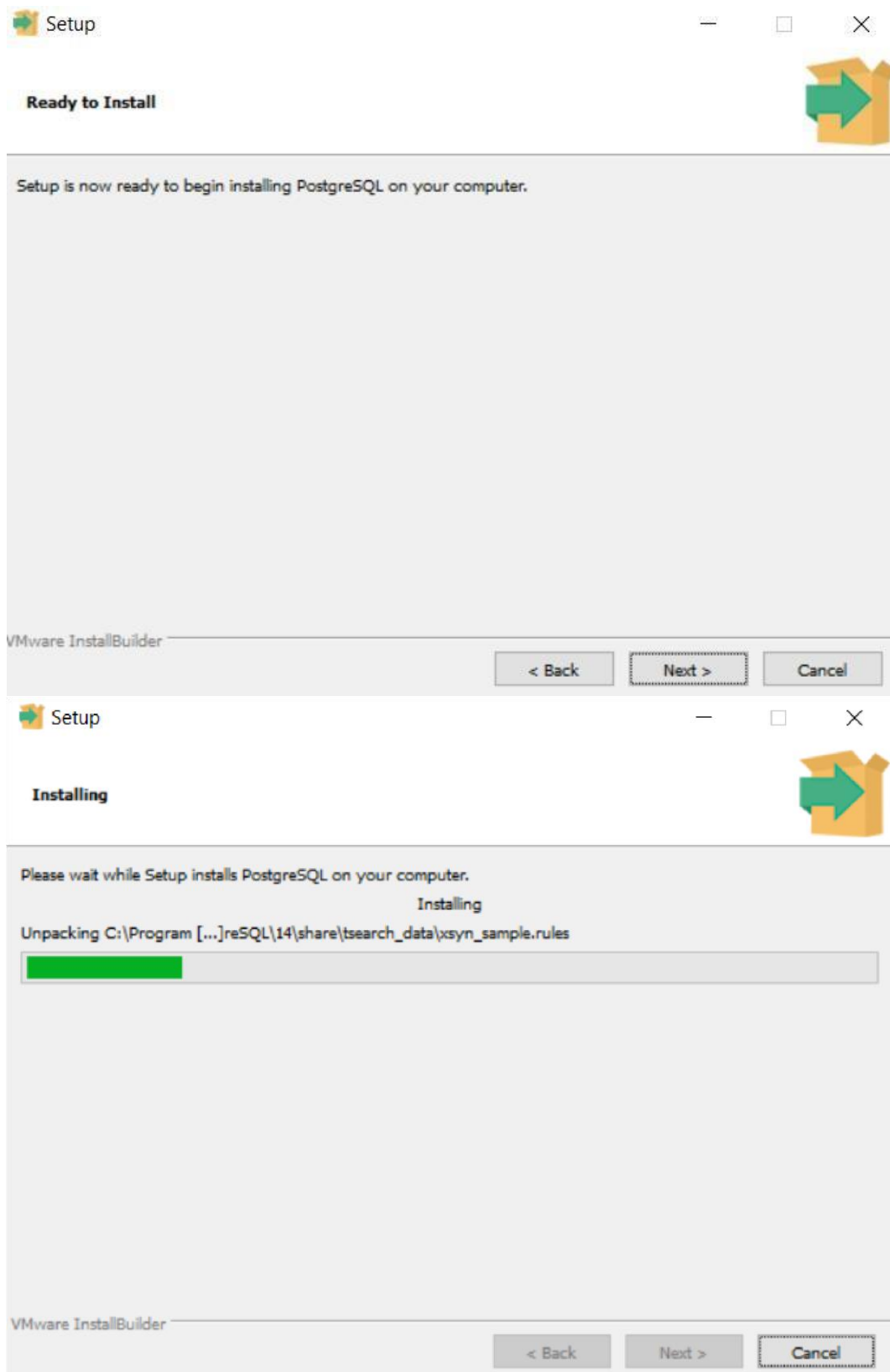


Step 9. The PostgreSQL summary information will be displayed by the setup wizard.

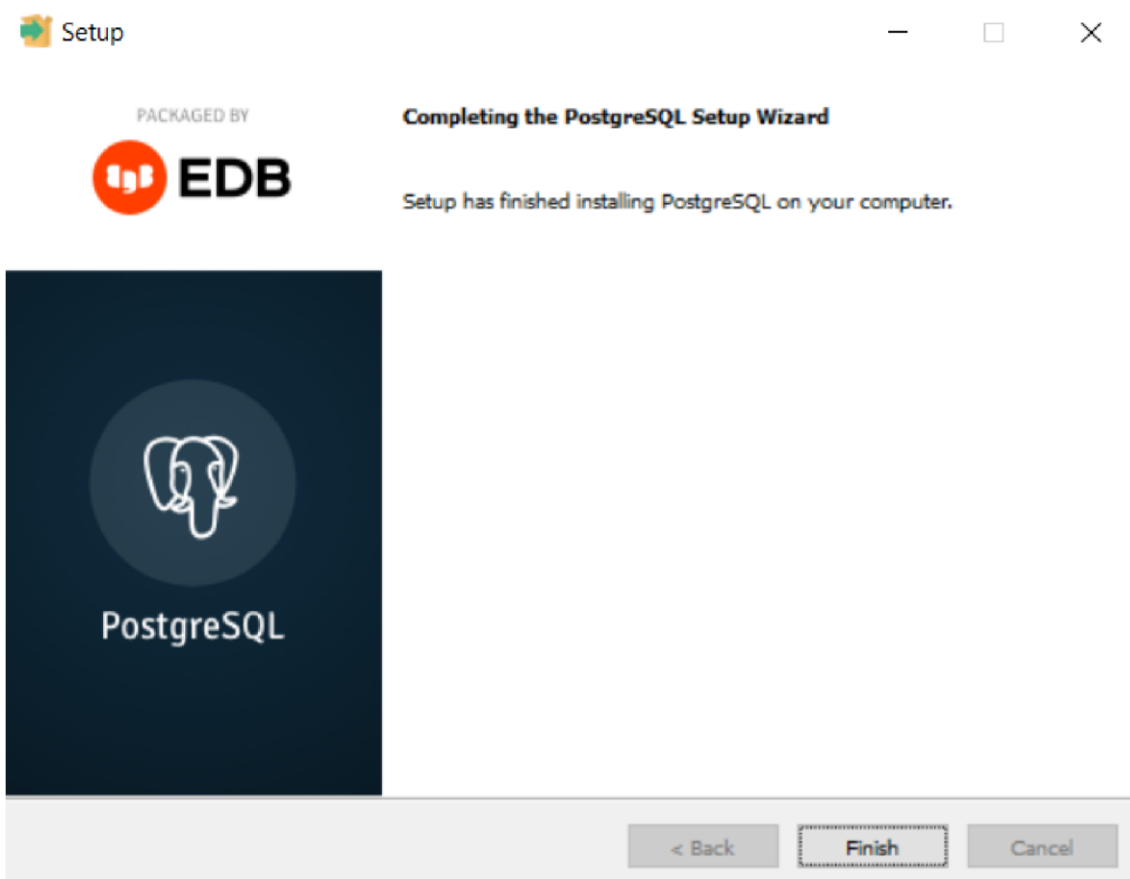


Now your computer is ready to install PostgreSQL, click next and installation will start.





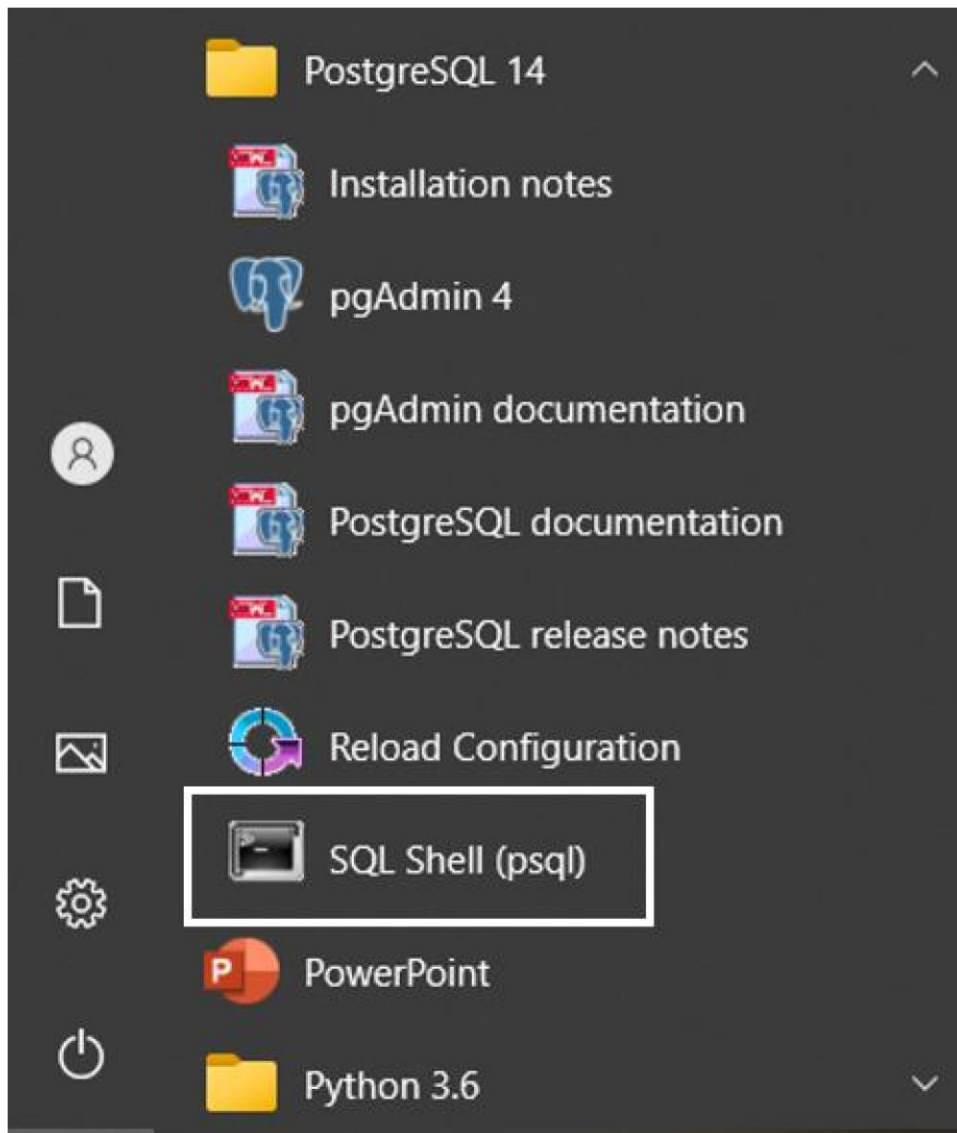
Step 10. Click on finish to complete the installation.



### 3) Verify the Installation

The quick way to verify the installation is through the psql program.

First, click the psql application to launch it. The psql command-line program will display.



You can accept the default by pressing Enter. It's important to note that you should use the same password that you used to install PostgreSQL. use **SELECT version();** to see the version of PostgreSQL.

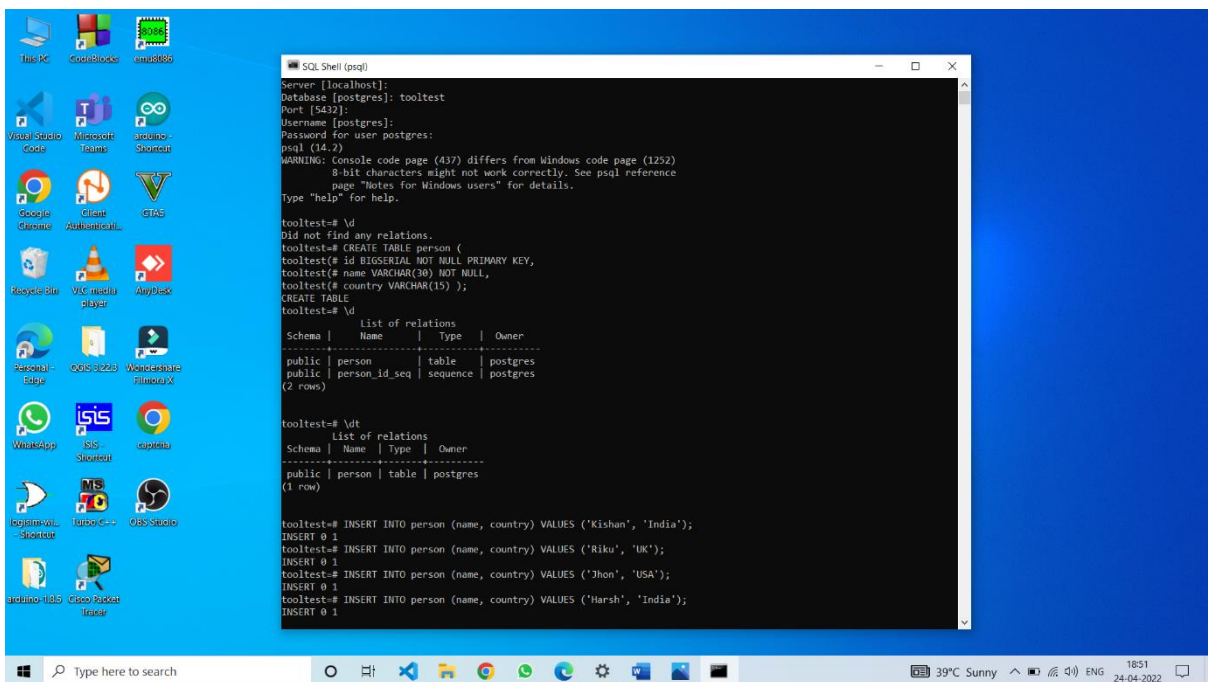
```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (14.2)
WARNING: Console code page (437) differs from Windows code page (1252)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

postgres=# select version();
           version
-----
PostgreSQL 14.2, compiled by Visual C++ build 1914, 64-bit
(1 row)

postgres=#
```

**Congratulation! you've successfully installed the PostgreSQL database server.**

**Login in created Database & CRUD operation using Query:**

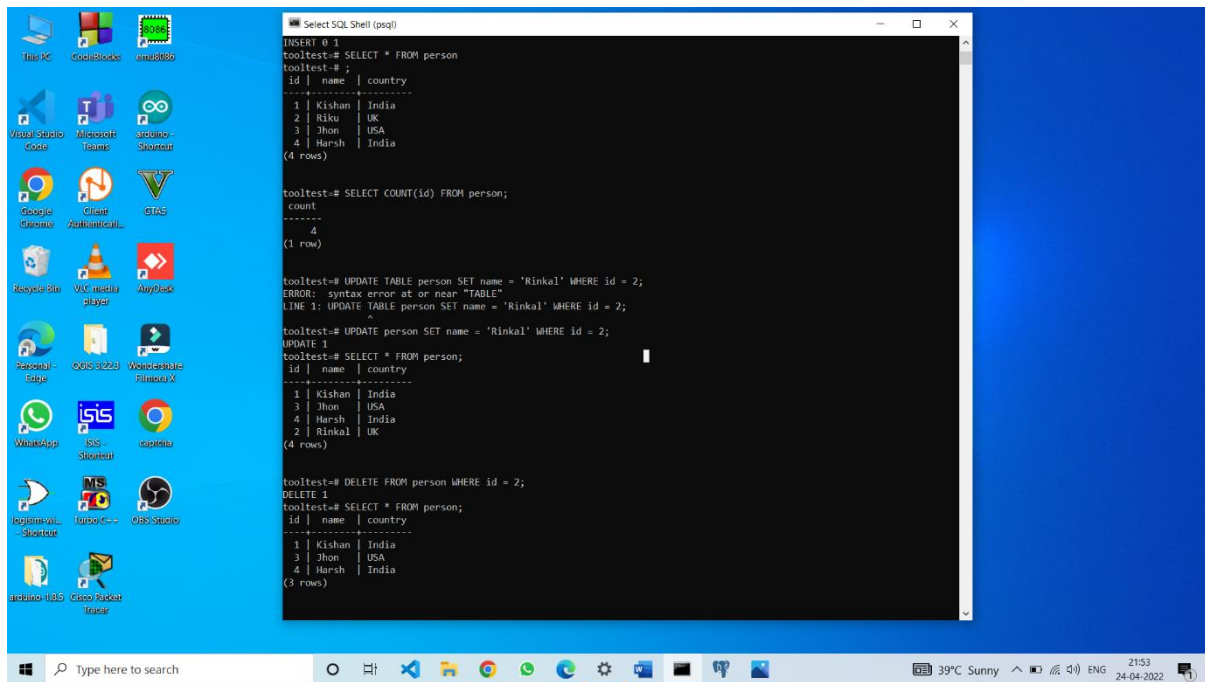


```
SQL Shell (psql)
Server [localhost]:
Database [postgres]: tooltest
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (14.2)
WARNING: Console code page (437) differs from Windows code page (1252)
        8-bit characters might not work correctly. See psql reference
        page "Notes for Windows users" for details.
Type "help" for help.

tooltest=# \d
Did not find any relations.
tooltest=# CREATE TABLE person (
tooltest=# id BIGSERIAL NOT NULL PRIMARY KEY,
tooltest=# name VARCHAR(30) NOT NULL,
tooltest=# country VARCHAR(15) );
CREATE TABLE
tooltest=# \d
      List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | person    | table  | postgres
public | person_id_seq | sequence | postgres
(2 rows)

tooltest=# \dt
      List of relations
Schema | Name | Type | Owner
-----+----+-----+-----
public | person | table | postgres
(1 row)

tooltest=# INSERT INTO person (name, country) VALUES ('Kishan', 'India');
INSERT 0 1
tooltest=# INSERT INTO person (name, country) VALUES ('Riku', 'UK');
INSERT 0 1
tooltest=# INSERT INTO person (name, country) VALUES ('Jhon', 'USA');
INSERT 0 1
tooltest=# INSERT INTO person (name, country) VALUES ('Harsh', 'India');
INSERT 0 1
```



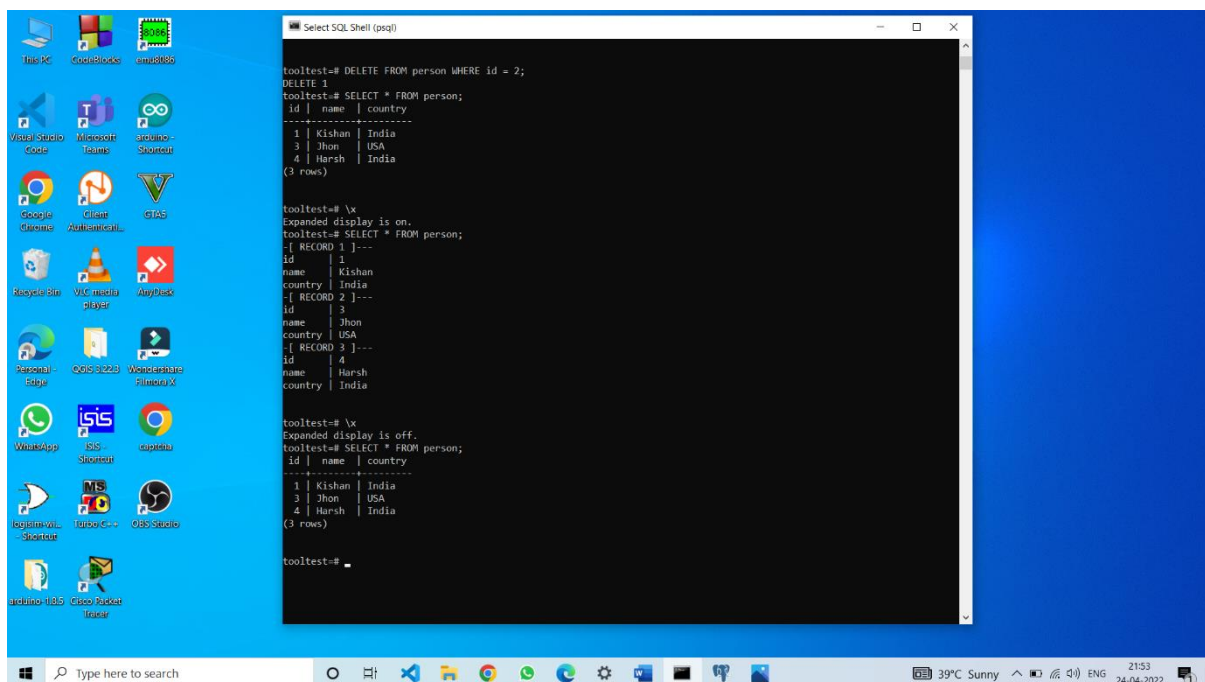
```
INSERT 0 1
tooltest=# SELECT * FROM person
tooltest=#
id | name | country
-----
1 | Kishan | India
2 | Riku | UK
3 | Jhon | USA
4 | Harsh | India
(4 rows)

tooltest=# SELECT COUNT(id) FROM person;
count
-----
4
(1 row)

tooltest=# UPDATE TABLE person SET name = 'Rinkal' WHERE id = 2;
ERROR: syntax error at or near "TABLE"
LINE 1: UPDATE TABLE person SET name = 'Rinkal' WHERE id = 2;
              ^

tooltest=# UPDATE person SET name = 'Rinkal' WHERE id = 2;
UPDATE 1
tooltest=# SELECT * FROM person;
id | name | country
-----
1 | Kishan | India
3 | Jhon | USA
4 | Harsh | India
2 | Rinkal | UK
(4 rows)

tooltest=# DELETE FROM person WHERE id = 2;
DELETE 1
tooltest=# SELECT * FROM person;
id | name | country
-----
1 | Kishan | India
3 | Jhon | USA
4 | Harsh | India
(3 rows)
```



```
tooltest=# DELETE FROM person WHERE id = 2;
DELETE 1
tooltest=# SELECT * FROM person;
id | name | country
-----
1 | Kishan | India
3 | Jhon | USA
4 | Harsh | India
(3 rows)

tooltest=# \x
Expanded display is on.
tooltest=# SELECT * FROM person;
-[ RECORD 1 ]---
id | 1
name | Kishan
country | India
-[ RECORD 2 ]---
id | 3
name | Jhon
country | USA
-[ RECORD 3 ]---
id | 4
name | Harsh
country | India

tooltest=# \x
Expanded display is off.
tooltest=# SELECT * FROM person;
id | name | country
-----
1 | Kishan | India
3 | Jhon | USA
4 | Harsh | India
(3 rows)

tooltest=#
```

**Open pgAdmin Database:**

