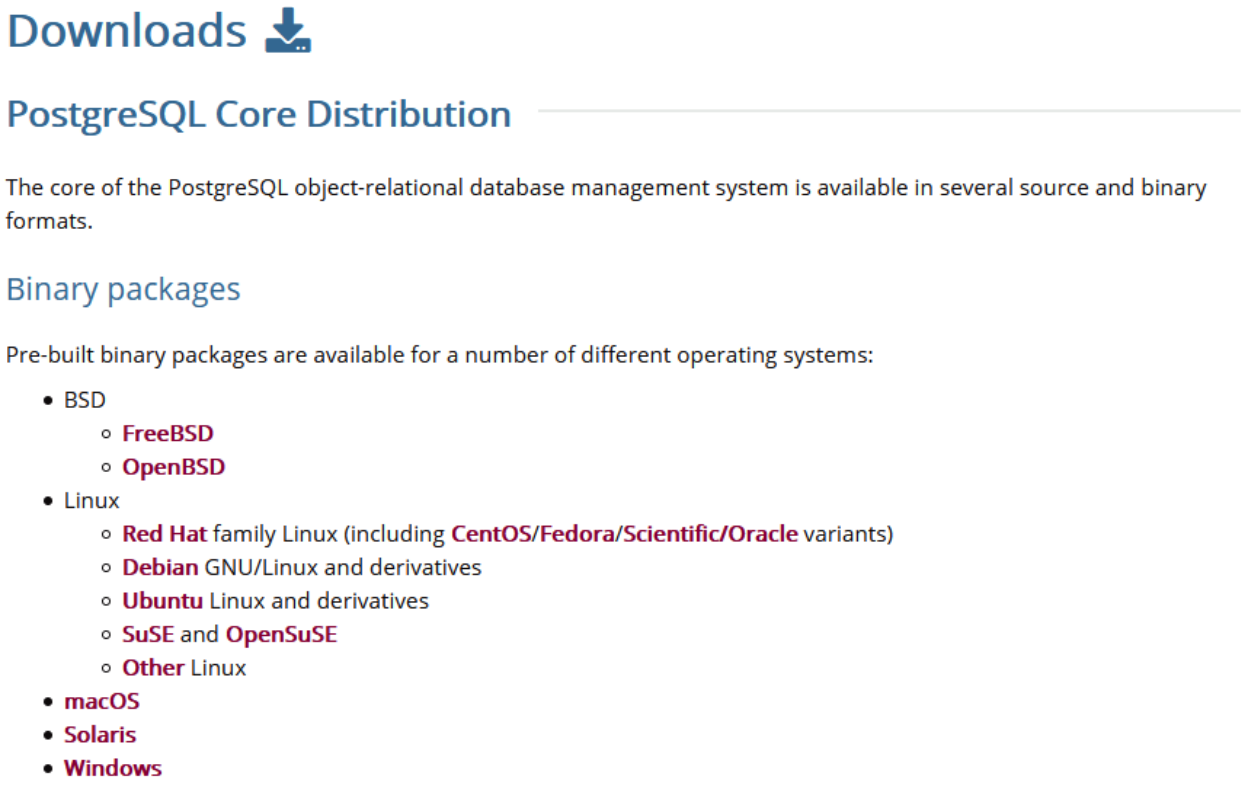
Installing & Using PostgreSQL

Updated: 14 August 2019

This guide explains how to install `PostgreSQL`, an open-source object-relational database management system on your laptop/desktop computer that you will be using for CS2102. It also describes how to use `psql`, `PostgreSQL`'s command-line client program to access a running `PostgreSQL` server.

1. Installing `PostgreSQL`

In the following, we describe a convenient way to install `PostgreSQL` for three main OS platforms. Other options are described at https://www.postgresql.org/download/.



Downloads

PostgreSQL Core Distribution

The core of the PostgreSQL object-relational database management system is available in several source and binary formats.

Binary packages

Pre-built binary packages are available for a number of different operating systems:

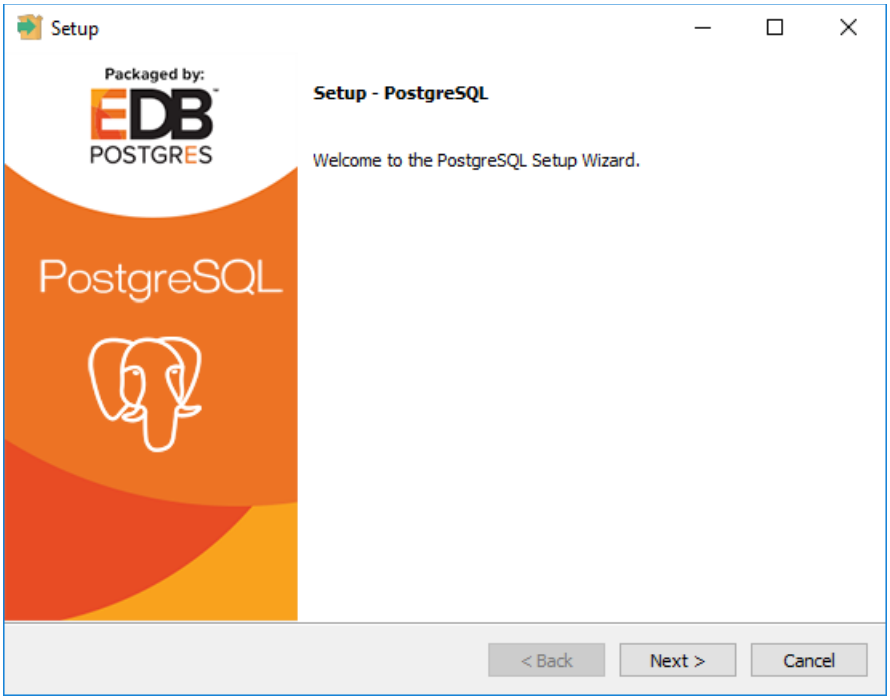
- BSD
  - **FreeBSD**
  - **OpenBSD**
- Linux
  - **Red Hat** family Linux (including **CentOS/Fedora/Scientific/Oracle** variants)
  - **Debian** GNU/Linux and derivatives
  - **Ubuntu** Linux and derivatives
  - **SuSE** and **OpenSuSE**
  - **Other** Linux
- **macOS**
- **Solaris**
- **Windows**

1.1. MacOS/Windows

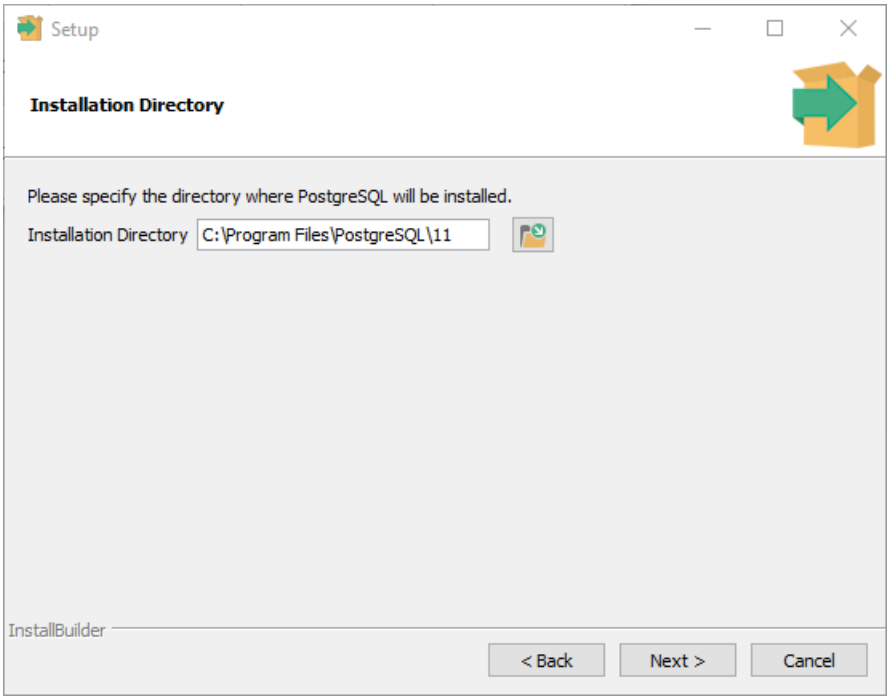
1. **Download** the EnterpriseDB installer for `PostgreSQL` 11.5 from https://www.postgresql.org/download/windows/.

PostgreSQL Version	Windows x86-64	Mac OS X	Linux x86-32	Linux x86-64	Windows x86-32
11.5	Download	Download	N/A	N/A	N/A
10.10	Download	Download	Download	Download	Download
9.6.15	Download	Download	Download	Download	Download
9.5.19	Download	Download	Download	Download	Download
9.4.24	Download	Download	Download	Download	Download
9.3.25 (Not Supported)	Download	Download	Download	Download	Download

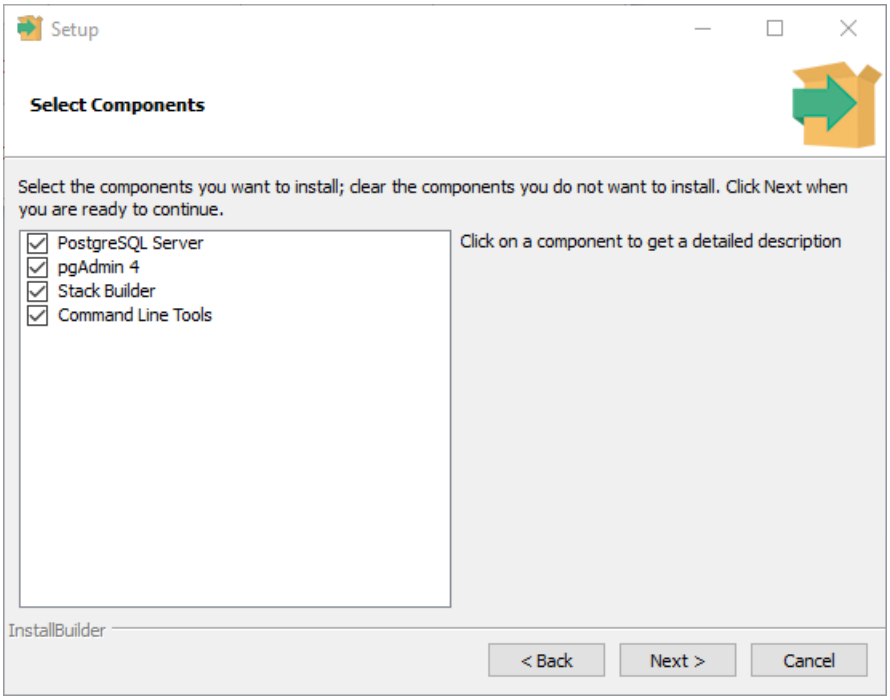
2. Execute the installer file. Click **next** to continue.



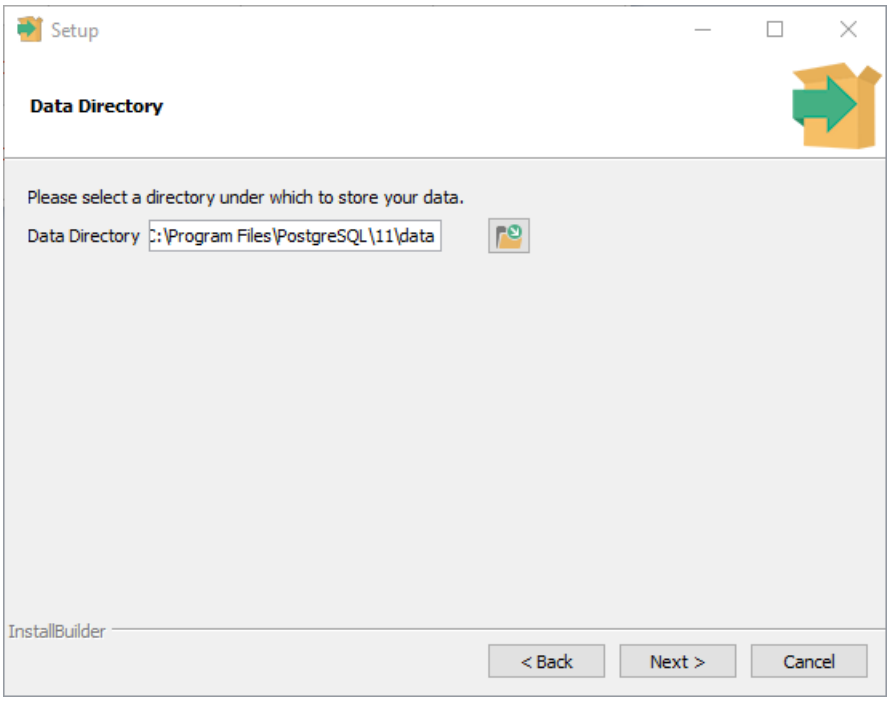
3. Configure the **installation directory**, *if necessary*. You may want to remember this installation directory. Click **Next >** to continue.



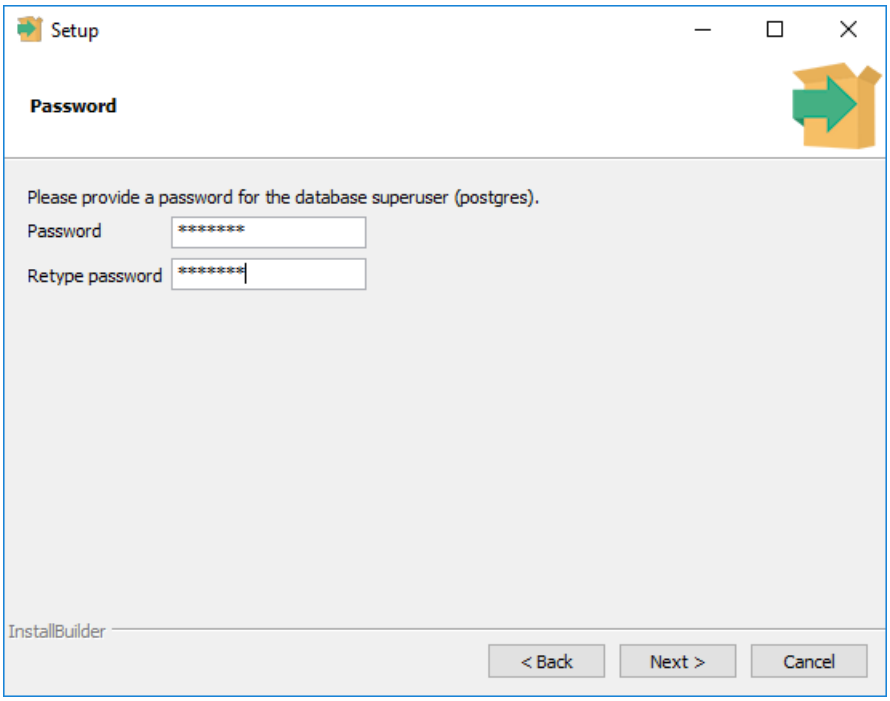
4. Choose the **components** you want, *it is recommended to install everything*. Click **Next >** to continue.



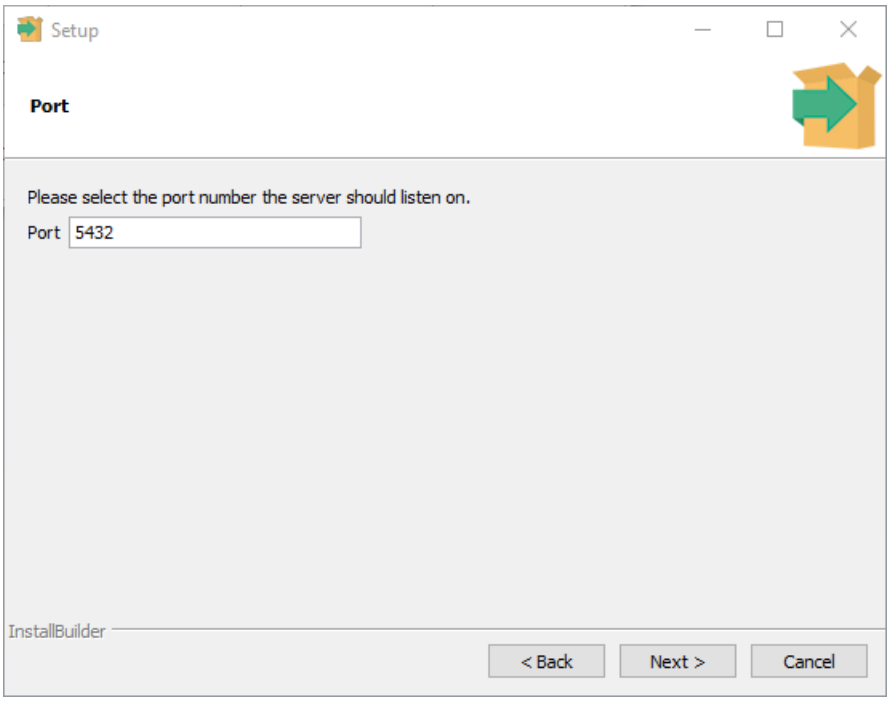
5. Configure the **data directory**, *if necessary*. Click **Next >** to continue.



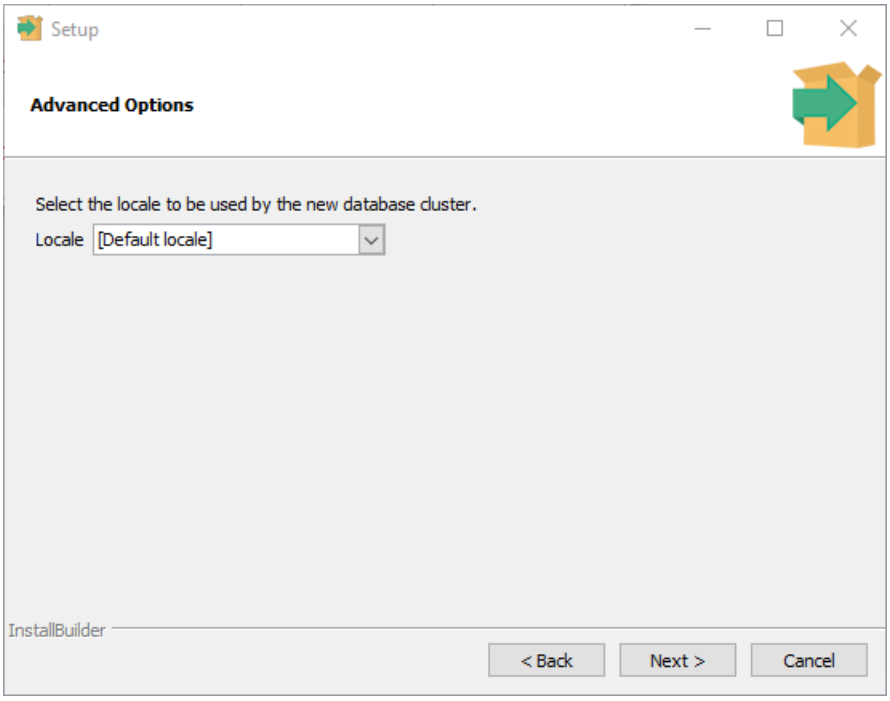
6. Create a **password** for the default database superuser named “postgres”. Retype the **password** and click **Next >** to continue.



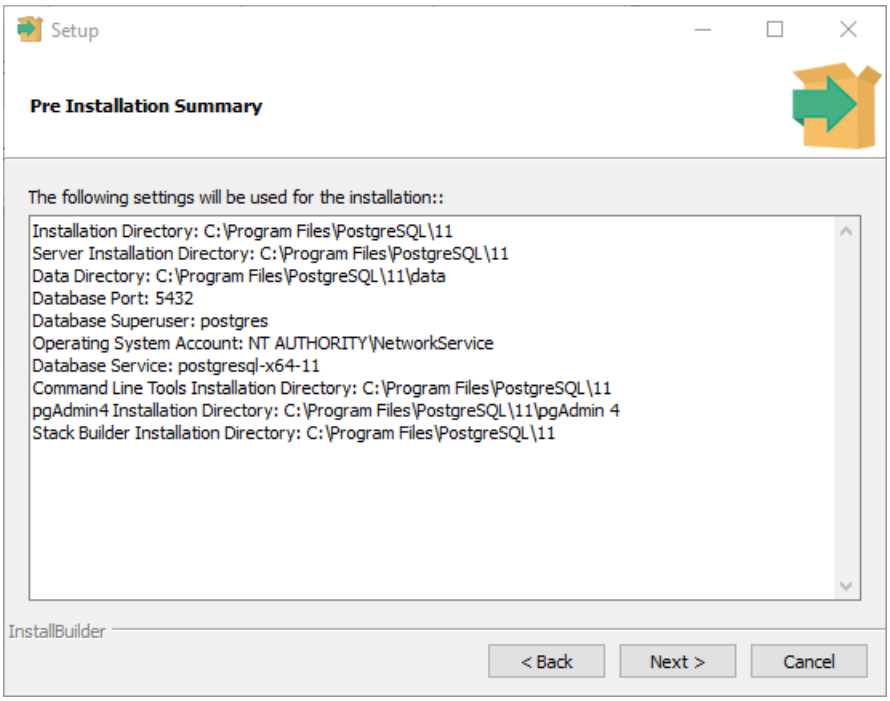
7. Configure the **port number**, *if necessary*. Click **Next >** to continue.



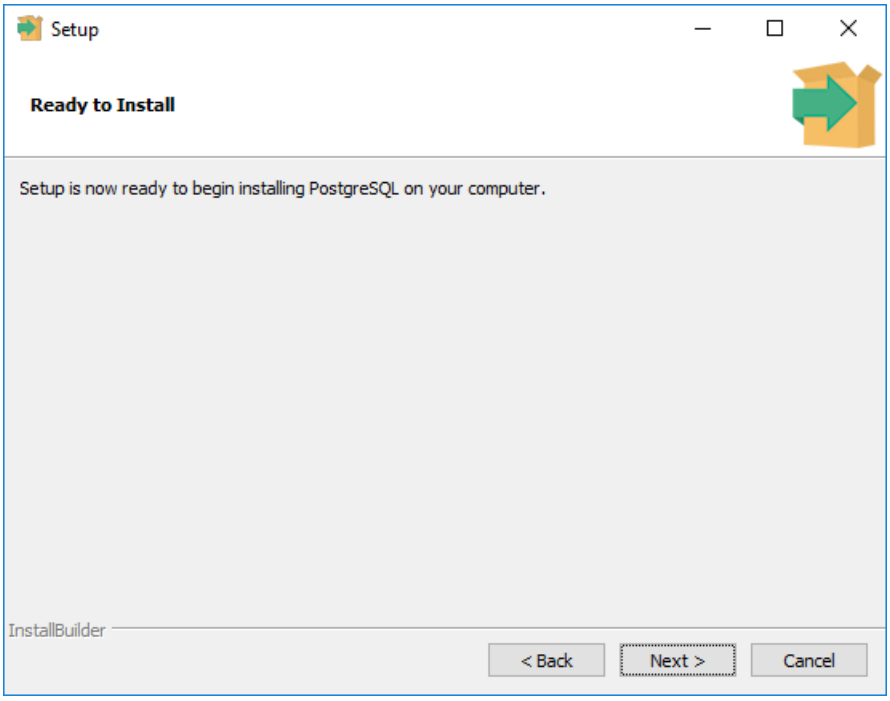
8. Configure the **locale**, *if necessary*. Click **next** to continue.



9. **Verify** that the configuration is OK. Click **Next >** to continue



10. Click **Next >** to install `PostgreSQL` on your computer.



On completion, both the `PostgreSQL` server as well as its command-line tool `psql` will be installed on your computer. **Untick** the box to launch Stack Builder at exit. Click **Finish** to complete.

1.2. Linux

You can install PostgreSQL using your Linux distribution's package manager. User input will be highlighted in `grey` while prompts will be in `monospace`. For example, on Ubuntu Linux, the installation command is:

```
sudo apt update; sudo apt install postgresql postgresql-contrib
```

The installation creates a default database named "postgres" and a `PostgreSQL` user (role in `PostgreSQL` parlance) named "postgres". For convenience, create a new `PostgreSQL` user with the same name as your Linux login id; *the following example assumes that your Linux userid is "alice".*

```
$ sudo -u postgresql psql
[sudo] password for alice: ********
psql (10.6 (Ubuntu 10.6-0ubuntu0.18.14.1))
Type "help" for help.

postgres=# create role alice with superuser login;
CREATE ROLE
postgres=# \q
```

Now you can simply use the command `psql postgres` to access the database named "postgres" as the user "alice" without being prompted for your password[1].

2. Connecting to PostgreSQL

There are two main ways to connect to a running PostgreSQL server. The first approach is to use PostgreSQL's command-line tool named psql which is already installed along with the PostgreSQL server installation. Section 2.1 gives an introduction on how to use psql. The second approach is to use a graphical user interface. One such tool is named DBeaver[2]. We will not teach you how to use DBeaver but you are welcomed to use it. You may refer to the Section 3 for more information about DBeaver. In all our example in the module, we will use psql instead.

2.1. Using psql

psql provides an interactive terminal interface to edit/execute SQL commands/queries and view query results. On a Windows installation, you can find a short-cut to psql in the Start Window. Clicking this application will open a new command window with psql connected to the default database named "postgres".

At psql's prompt "postgres=#", you can type in SQL commands. Input lines that terminate with a semicolon will be sent to the database server for execution. To exit psql, enter `\q` at psql's "postgres=#" prompt. An example run is shown below.

```
postgres=# create table students (
postgres#   sid integer,
postgres#   name varchar(80)
postgres# );
CREATE TABLE
postgres#
```

Besides SQL commands, you can also type in *meta-commands* that will be processed by psql. Each meta-command begins with an unquoted backslash. Any SQL command that has been typed but has not yet been sent for execution is stored in a memory buffer called *query buffer*. The contents of this buffer can be edited by invoking a configurable text editor within psql. The default editor is vim (on Linux[3]).

---

[1] https://www.postgresql.org/docs/current/auth-peer.html
[2] https://dbeaver.io/
[3] The editor program can be configured using the environment variable PGSQL_EDITOR. For more details on configuring psql, refer to https://www.postgresql.org/docs/current/app-psql.html

The table below shows some of the basic meta-commands.

Meta-command	Meaning
\q	Quit psql.
\h	Display all SQL commands with available syntax help.
\h COMMAND	Display syntax of COMMAND (e.g., \h create table).
\d	List all created tables.
\d TABLE	List information on relation named TABLE.
\p	Display the contents of the query buffer; if the current query buffer is empty, display the most recently executed query.
\w FILE	Output the contents of the query buffer to the file named FILE; if the current query buffer is empty, output the most recently executed query to FILE.
\r	Clear the query buffer.
\e	Invoke the text editor to edit the contents of the query buffer; if the current query buffer is empty, edit the most recently executed query.
\e FILE	Invoke the text editor to edit the contents of the file named FILE. The contents of the edited file will be copied to the query buffer at the end of the edit session.
\o FILE	Enable future query results to be saved to the file named FILE.
\g	Send the contents of the current query buffer to the server for execution; if the current query buffer is empty, the most recently sent query is re-executed.
\i FILE	Read the contents from the file named FILE and sent their contents to the server for execution.
\!	Escapes from the psql session to a sub-shell. The psql session resumes when the sub-shell is exited.

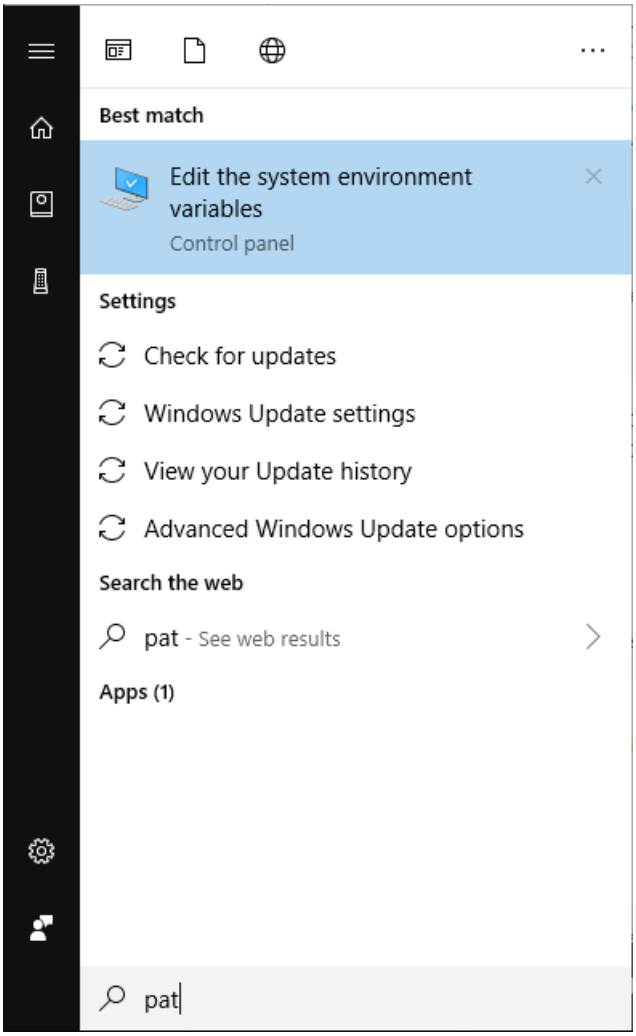
The following example illustrates the execution of some meta-commands. Download the SQL script named pizza.sql from Luminus' SQL workbin.

```
postgres=# \d
          List of relations
 Schema |   Name   | Type  | Owner
--------+----------+-------+----------
 public | students | table | postgres
(1 row)

postgres=# \d students
                    Table "public.students"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 sid    | integer               |           |          |
 name   | character varying(80) |           |          |

postgres=# \i /PATHNAME-OF-DOWNLOADED-SCRIPT/pizza.sql
postgres=#
postgres=# \q
```

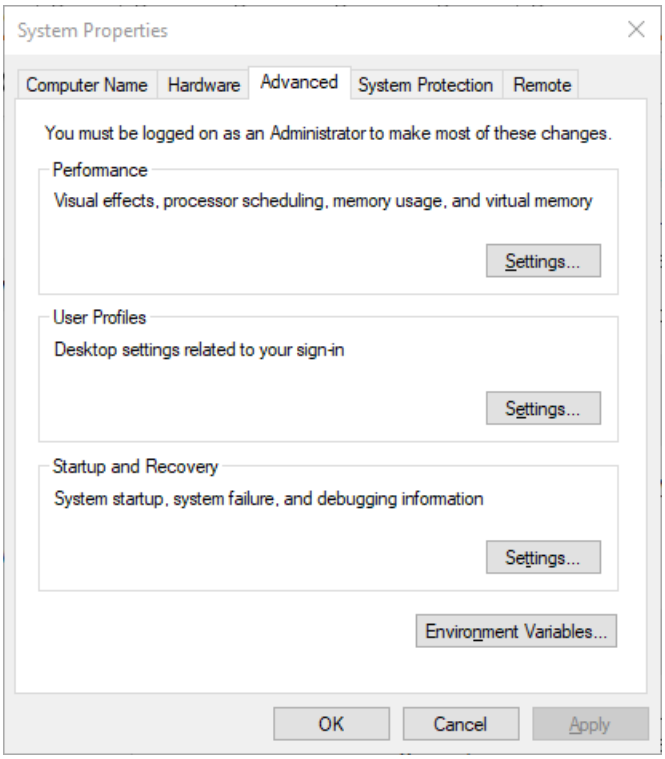
2.2. On Windows

It is likely that the above example does not work for Windows. If you are using Windows, it is advisable to add the `bin` directory of `PostgreSQL` into your `PATH` environment variable. To add `bin` directory of `PostgreSQL` to `PATH` environment variable, follow the steps below:

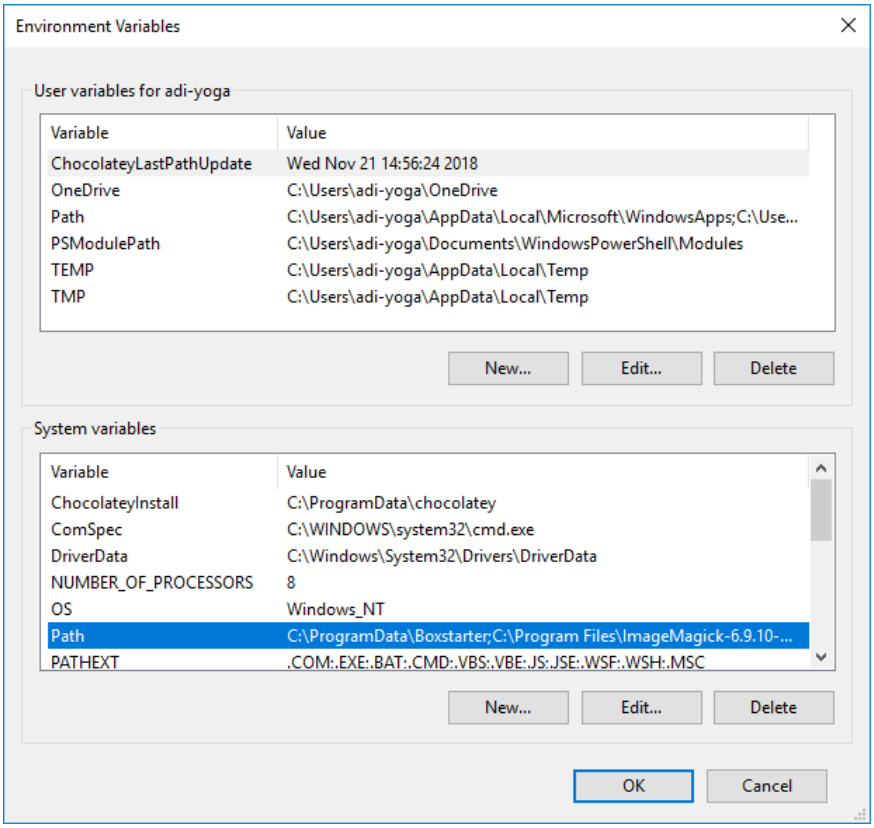
- Search for "**Edit the system environment variables**" in the Start Window.



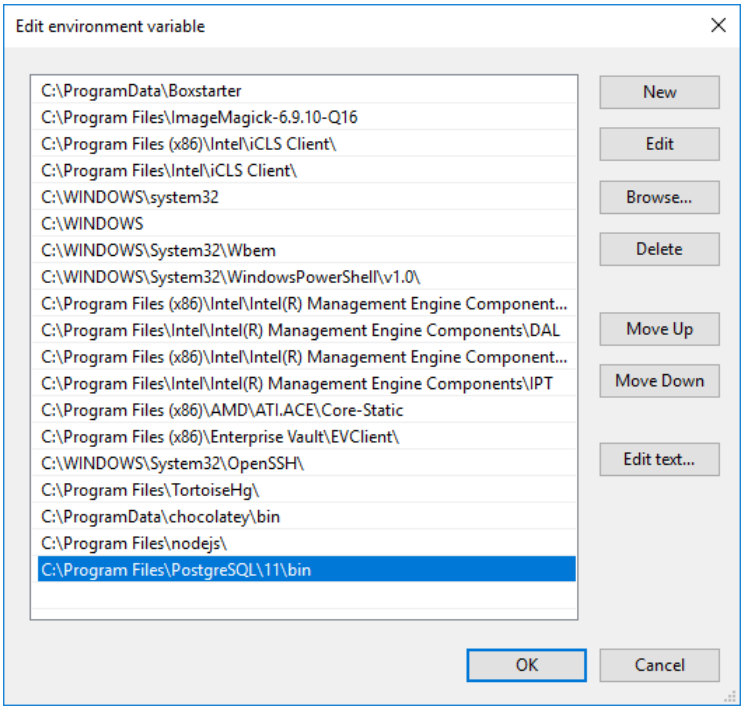
- Click on **Environment Variables...** button.



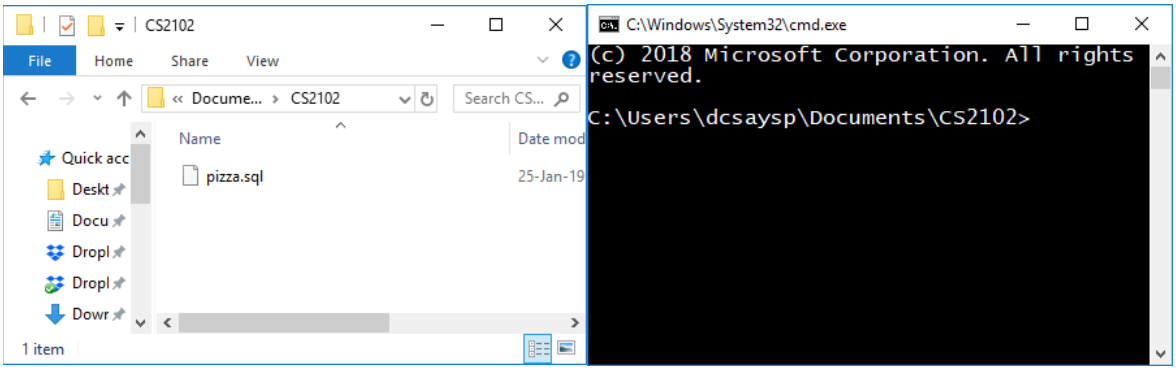
- Select "**Path**" on "**System variables**" panel and click the **Edit...** button.



- Click **New** and add the path to `bin`. If you use the default, it is likely to be `C:\Program Files\PostgreSQL\11\bin`. Otherwise, you should consult the step 3 on Section 1.1. Click **OK** to close.



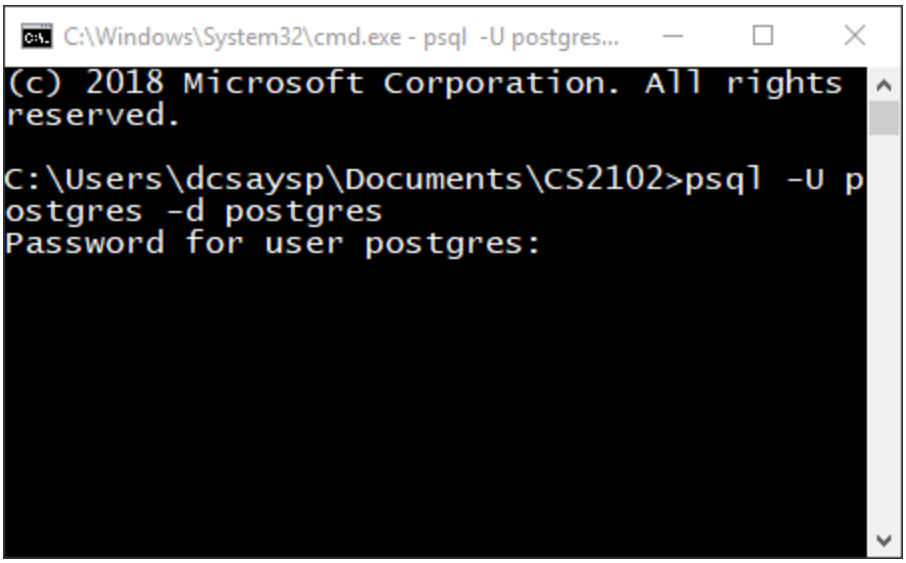
Once you have added the `bin` directory of `PostgreSQL` to `PATH` environment variable, open `CMD` and go to the folder where you have downloaded `pizza.sql`.



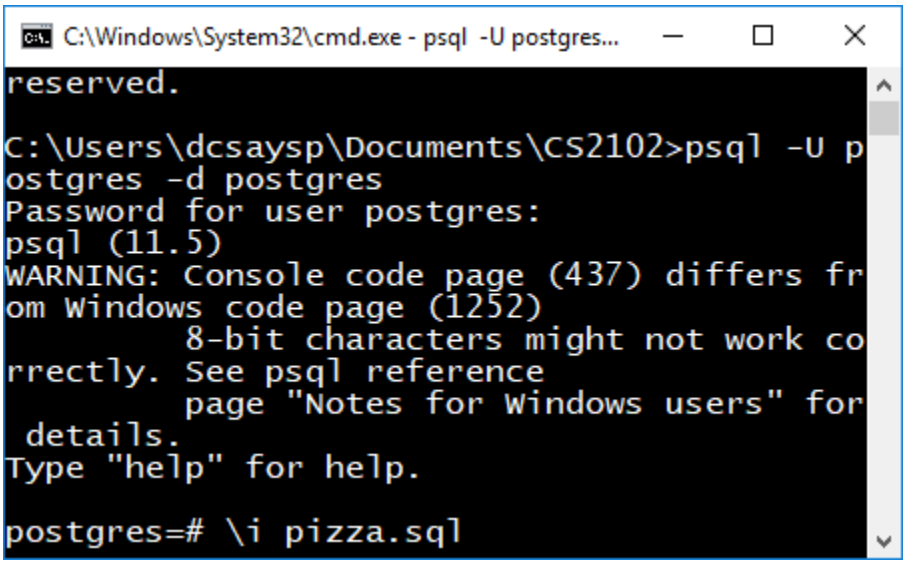
Execute the following command on `CMD`.

```
psql -U username -d database
```

Where username is your `username` (or `postgres` for default) and database is your `database` (or `postgres` for default). You will be prompted for your password. Enter your password.



You can now run the command `\i pizza.sql`.



3. References

- `PostgreSQL` documentation https://www.postgresql.org/docs/current/index.html
- `PostgreSQL` tutorial https://www.postgresql.org/docs/current/tutorial.html
- Manpage of `psql` https://www.postgresql.org/docs/current/app-psql.html
- `DBeaver` documentation https://dbeaver.io/docs/