

Basic Tools Installation and Management, psql



Copyright

© Postgres Professional, 2017–2025

Authors: Egor Rogov, Pavel Luzanov, Ilya Bashtanov, Igor Gnatyuk

Translated by: Liudmila Mantrova, Alexander Meleshko, Elena Sharafutdinova

Photo by: Oleg Bartunov (Phu monastery, Bhrikuti summit, Nepal)

Use of Course Materials

Non-commercial use of course materials (presentations, demonstrations) is allowed without restrictions. Commercial use is possible only with the written permission of Postgres Professional. It is prohibited to make changes to the course materials.

Feedback

Please send your feedback, comments and suggestions to:

edu@postgrespro.ru

Disclaimer

Postgres Professional assumes no responsibility for any damages and losses, including loss of income, caused by direct or indirect, intentional or accidental use of course materials. Postgres Professional company specifically disclaims any warranties on course materials. Course materials are provided “as is,” and Postgres Professional company has no obligations to provide maintenance, support, updates, enhancements, or modifications.

PostgreSQL Installation Types

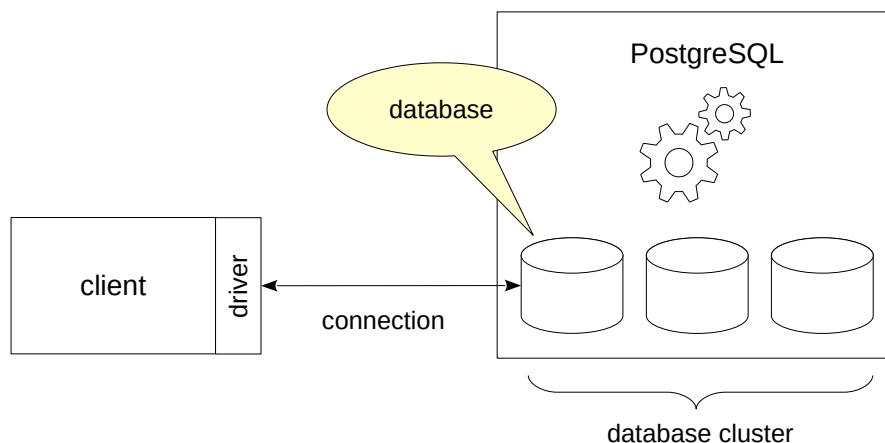
Server Management

Server Message Log

Configuration Parameters

Using psql

Database Cluster



Let's start with the general concepts.

PostgreSQL is a program that belongs to the class of *database management systems*.

When this program is running, we call it a PostgreSQL *server* or a *server instance*. So far, the server seems to be a “black box” for us, but gradually we will get acquainted with how it works.

The data managed by PostgreSQL is stored in *databases*. One instance of PostgreSQL simultaneously manages several databases. This set of databases is called a *database cluster*. We will talk more about databases in the “Data organization. Logical structure” lesson.

The server interacts with clients. Clients are external applications that can connect to server databases and send *queries* for execution.

To summarize: a database cluster is the data in files; a server or a server instance is a program that manages the database cluster; a client is an application that “talks” to the server.

Installation types

- pre-build packages (preferred)
- installation from source code
- without installation (cloud services)

Extensions

- provide additional features
- installed separately
- shipped with the server as modules and programs (~50 extensions)

The preferred option for installing PostgreSQL is via a package manager (such as apt or rpm) using pre-built packages. This gives you a comprehensive installation that is easy to support and upgrade. There are packages available for most operating systems.

Another option is to build PostgreSQL from source code. This may be necessary if you want to set up a non-standard configuration or deploy on an exotic platform.

Pre-built packages and source codes: <http://www.postgresql.org/download/>

Besides, you can work with cloud-based managed databases that do not require any installation at all. Such ready-to-use services are provided by all major cloud platforms (Amazon RDS, Google Cloud SQL, Microsoft Azure).

In this course, we are going to use a virtual machine with Xubuntu 24 OS; PostgreSQL 16 is installed from the package for this OS. In addition, the installation is set up to start and stop PostgreSQL when the OS starts and stops.

There are a lot of PostgreSQL extensions that add new database functionality “on the fly”, without modifying the system core. About 50 extensions are included into the PostgreSQL distribution itself.

<https://postgrespro.com/docs/postgresql/16/contrib>


<https://postgrespro.com/docs/postgresql/16/contrib-prog>

The list of available extensions and their installation status can be accessed using the `pg_available_extensions` view.



Management tools

pg_ctlcluster

 pg_ctl

Main operations

start the server

stop the server

reload configuration parameters

The main server management tasks are: initializing the database cluster, starting and stopping the server, reloading configuration parameters, and a few others. To perform these actions, use the `pg_ctl` utility, which comes together with PostgreSQL.

In the Ubuntu PostgreSQL package, `pg_ctl` is not run directly, but through a special wrapper `pg_ctlcluster`. The help information for `pg_ctlcluster` can be viewed using `man`:

```
$ man pg_ctlcluster
```

Information about existing clusters and their status can be obtained using the following commands:

```
$ pg_lsclusters
```

```
$ pg_ctlcluster status
```

For more details about server management for database administrators, see:

<https://postgrespro.com/docs/postgresql/16/app-pg-ctl>

<https://postgrespro.com/docs/postgresql/16/runtime>

Installation and Management

PostgreSQL in the course VM is installed from packages. PostgreSQL installation directory:

```
student$ ls -l /usr/lib/postgresql/16
```

```
total 8
drwxr-xr-x 2 root root 4096 Nov 27 12:16 bin
drwxr-xr-x 4 root root 4096 Nov 27 13:14 lib
```

The owner of the server software is the root user.

The database cluster is automatically initialized when installed from a package and is located in the /var/lib/postgresql/16/main directory.

This directory will be referred to as PGDATA in subsequent topics.

The owner of the directory is postgres. Here is what the directory contains:

```
student$ sudo ls -l /var/lib/postgresql/16/main
```

```
total 96
drwx----- 1 postgres postgres 4096 Nov 27 14:17 base
drwx----- 1 postgres postgres 4096 Nov 27 14:17 global
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_commit_ts
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_dynshmem
drwx----- 4 postgres postgres 4096 Nov 27 13:16 pg_logical
drwx----- 4 postgres postgres 4096 Nov 27 13:14 pg_multixact
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_notify
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_replslot
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_serial
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_snapshots
drwx----- 1 postgres postgres 4096 Nov 27 14:17 pg_stat
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_stat_tmp
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_subtrans
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_tblspc
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_twophase
-rw----- 1 postgres postgres   3 Nov 27 13:14 PG_VERSION
drwx----- 1 postgres postgres 4096 Nov 27 13:14 pg_wal
drwx----- 2 postgres postgres 4096 Nov 27 13:14 pg_xact
-rw----- 1 postgres postgres   88 Nov 27 13:14 postgresql.auto.conf
-rw----- 1 postgres postgres  130 Nov 27 14:17 postmaster.opts
-rw----- 1 postgres postgres  108 Nov 27 14:17 postmaster.pid
```

With the package installation, PostgreSQL is automatically added to the OS startup list, so there is no need to start it manually.

Use the following commands to control the server as a privileged OS user with sudo:

Stop the server:

```
student$ sudo pg_ctlcluster 16 main stop
```

Start the server:

```
student$ sudo pg_ctlcluster 16 main start
```

Restart:

```
student$ sudo pg_ctlcluster 16 main restart
```

Reload configuration:

```
student$ sudo pg_ctlcluster 16 main reload
```

Get server information:

```
student$ sudo pg_ctlcluster 16 main status
```

```
pg_ctl: server is running (PID: 2836)
/usr/lib/postgresql/16/bin/postgres "-D" "/var/lib/postgresql/16/main" "-c"
"config_file=/etc/postgresql/16/main/postgresql.conf"
```

List of installed instances (works without sudo):

```
student$ pg_lsclusters
```

| Ver | Cluster | Port | Status | Owner | Data directory | Log file |
|-----|---------|------|--------|----------|-----------------------------|----------|
| 16 | main | 5432 | online | postgres | /var/lib/postgresql/16/main | |

/var/log/postgresql/postgresql-16-main.log

Server message log contains

- server messages
- user session messages
- application messages

Log configuration

- log file location
- message format
- events to log

Database operations are tracked in the server message log. The log keeps records about starting and stopping the server, as well as various signal messages, including messages about possible issues.

It can also store records of executed commands, their running time, locks that occur, etc. It can be used to trace user sessions.

Application developers can generate and log their own messages.

PostgreSQL allows you to flexibly configure which messages and in which format should be recorded into the log.

For example, the CSV and JSON output formats are convenient for automating log analysis.

<https://postgrespro.com/docs/postgresql/16/runtime-config-logging>

Server Message Log

The server message log is located here:

```
student$ ls -l /var/log/postgresql/postgresql-16-main.log
```

```
-rw-r----- 1 postgres adm 4532 Nov 27 14:17 /var/log/postgresql/postgresql-16-main.log
```

Let's look at the last messages:

```
student$ tail -n 10 /var/log/postgresql/postgresql-16-main.log
```

```
2025-11-27 14:17:36.340 MSK [812] LOG:  background worker "logical replication launcher"
(PID 837) exited with exit code 1
2025-11-27 14:17:36.344 MSK [829] LOG:  shutting down
2025-11-27 14:17:36.350 MSK [829] LOG:  checkpoint starting: shutdown immediate
2025-11-27 14:17:36.382 MSK [829] LOG:  checkpoint complete: wrote 3 buffers (0.0%); 0
WAL file(s) added, 0 removed, 0 recycled; write=0.009 s, sync=0.005 s, total=0.038 s;
sync files=2, longest=0.003 s, average=0.003 s; distance=0 kB, estimate=0 kB;
lsn=0/1963610, redo lsn=0/1963610
2025-11-27 14:17:36.391 MSK [812] LOG:  database system is shut down
2025-11-27 14:17:37.181 MSK [2836] LOG:  starting PostgreSQL 16.11 (Ubuntu
16.11-1.pgdg24.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu
13.3.0-6ubuntu2~24.04) 13.3.0, 64-bit
2025-11-27 14:17:37.182 MSK [2836] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2025-11-27 14:17:37.186 MSK [2836] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2025-11-27 14:17:37.207 MSK [2839] LOG:  database system was shut down at 2025-11-27
13:16:17 MSK
2025-11-27 14:17:37.243 MSK [2836] LOG:  database system is ready to accept connections
```

Configuration Parameters



For the entire instance

the main configuration file is postgresql.conf

ALTER SYSTEM modifies postgresql.auto.conf

For the current session

SET/RESET

set_config()

View the current setting

SHOW

current_setting()

pg_settings

\dconfig

9

The PostgreSQL server configuration is governed by a variety of parameters that are used to manage resource consumption, tune system processes and user sessions, manage the server log, and handle many other tasks. It's important to know how to check and update the parameter values.

Server configuration is usually stored in configuration files. The main configuration file is postgresql.conf, it has to be edited manually. The second configuration file is postgresql.auto.conf. Its contents are set using the ALTER SYSTEM command. The parameters set via ALTER SYSTEM take precedence over the parameters in postgresql.conf.

File and directory inclusion directives *include* and *include_dir* allow splitting large postgresql.conf into manageable parts. This may come in handy when managing multiple servers with similar configurations.

Most configuration parameters can be adjusted within a session with no server restart. You can define custom parameters and manage them in the same manner as with the system parameters.

Different ways of setting and updating parameters are described here:

<https://postgrespro.com/docs/postgresql/16/config-setting>

And the current parameter values are shown in the pg_settings view.

<https://postgrespro.com/docs/postgresql/16/view-pg-settings>

Configuration Parameters

The main postgresql.conf configuration file is located in this directory:

```
student$ ls -l /etc/postgresql/16/main
```

```
total 60
drwxr-xr-x 2 postgres postgres 4096 Nov 27 13:14 conf.d
-rw-r--r-- 1 postgres postgres 315 Nov 27 13:14 environment
-rw-r--r-- 1 postgres postgres 143 Nov 27 13:14 pg_ctl.conf
-rw-r----- 1 postgres postgres 5743 Nov 27 13:14 pg_hba.conf
-rw-r----- 1 postgres postgres 2640 Nov 27 13:14 pg_ident.conf
-rw-r--r-- 1 postgres postgres 29960 Nov 27 13:14 postgresql.conf
-rw-r--r-- 1 postgres postgres 317 Nov 27 13:14 start.conf
```

Other configuration files are also stored here.

Let's check the value of the work_mem parameter:

```
=> SHOW work_mem;
```

```
work_mem
-----
4MB
(1 row)
```

The work_mem parameter specifies the amount of memory that will be used for internal sorting and hash table allocation operations before using temporary files on disk.

4MB is the default value and it is too small. Let's say we want to increase it to 16MB for the entire instance. There are various ways to do this.

First, you can uncomment and change the line in postgresql.conf where the parameter is defined:

```
student$ grep '#work_mem' /etc/postgresql/16/main/postgresql.conf
```

```
#work_mem = 4MB                                # min 64kB
```

Secondly, you can place the parameter definition in a file with the .conf suffix in the /etc/postgresql/16/main/conf.d directory or in a custom configuration file, the location of which should be set in the include directive of the main configuration file postgresql.conf.

Thirdly, you can change the parameter value using an SQL command, and this is what we will do:

```
=> ALTER SYSTEM SET work_mem TO '16MB';
```

```
ALTER SYSTEM
```

This change does not end up in postgresql.conf, but in postgresql.auto.conf, which is located in the PGDATA directory:

```
student$ sudo cat /var/lib/postgresql/16/main/postgresql.auto.conf
```

```
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
work_mem = '16MB'
```

Reload the configuration files for the change to take effect. To do this, you can use pg_ctlcluster, or use the SQL function:

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

Verify that the new value is applied. Here is another way to check it:

```
=> SELECT current_setting('work_mem');
```

```
current_setting
-----
16MB
(1 row)
```

To restore the default value of a parameter, you can use the RESET command instead of SET (and then reload the configuration, of course):

```
=> ALTER SYSTEM RESET work_mem;
```

ALTER SYSTEM

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

Most parameter values can be set for the current session live. For example, you can increase `work_mem` just for a single query that you expect to sort a large volume of data:

```
=> SET work_mem = '64MB';
```

SET

The new value will remain for the current session or even for the current transaction (with `SET LOCAL`).

Yet another way to check a current value is to query the `pg_settings` view:

```
=> SELECT name, setting, unit FROM pg_settings WHERE name = 'work_mem';
```

```
name | setting | unit
-----+-----+-----
work_mem | 65536 | kB
(1 row)
```

Lastly, there is the `\dconfig` command:

```
=> \dconfig work_mem
```

List of configuration parameters

```
Parameter | Value
-----+-----
work_mem | 64MB
(1 row)
```

The psql Client



Terminal client for working with PostgreSQL

Comes with the DBMS

Used by administrators and developers for interactive work and script execution

11

There are other third-party tools available, but they are not considered in the scope of the course.

In the course, we will use the psql terminal client:

1. psql is the only client supplied with the DBMS.
2. The knowledge of psql will be useful to both developers and DB administrators, regardless of which tool they choose to work with at the end of the day.

For interactivity, psql provides built-in support for readline and pager programs (such as *less* and *pspg*), as well as the option to use external editing tools. Using psql, you can interact with the operating system, browse through the system catalog, and write scripts to automate routine tasks.

<https://postgrespro.com/docs/postgresql/16/app-psql>

Connection

When starting psql, you need to specify the connection parameters. Required parameters include:

- database name, same as the user name by default
- user name (role), same as the OS user name by default
- host, local connection by default
- port, usually 5432 by default

This is how you put them in:

```
student$ psql -d database -U user -h host -p port
```

The VM is set up to connect to PostgreSQL without specifying the parameters:

```
student$ psql
```

Check the connection:

```
=> \conninfo
```

You are connected to database "student" as user "student" via socket in "/var/run/postgresql" at port "5432".

The \connect command creates a new connection without leaving psql. You can use its short form: \c. From here on, the optional parts of commands will be shown in square brackets: \c[onnect].

Help and Documentation

You don't need to open the documentation every time you need to check something. Instead, you can get the information from within the system directly. The command

```
student$ psql --help
```

displays the help information for psql. If PostgreSQL was installed with documentation, then you can get the reference manual with the command

```
student$ man psql
```

psql can execute SQL commands and its own commands which start with a backslash, such as \conninfo. psql commands are always single-line, unlike SQL commands.

Inside psql, you can get a list and a brief description of all psql commands:

- \? displays the list of psql commands
- \h[elp] provides a list of SQL commands that the server supports, as well as the syntax of an SQL command (if specified).

Formatting the Output

The psql client can return output in different formats:

- aligned format
- unaligned format
- extended format

The aligned format is the default:

```
=> SELECT name, setting, unit FROM pg_settings LIMIT 7;
```

| name | setting | unit |
|----------------------------|------------|------|
| allow_in_place_tablespaces | off | |
| allow_system_table_mods | off | |
| application_name | psql | |
| archive_cleanup_command | | |
| archive_command | (disabled) | |
| archive_library | | |
| archive_mode | off | |

(7 rows)

It sets each column's width based on its contents. There's also the header and the footer.

psql commands to switch display modes:

- \a — mode switch: aligned/unaligned.
- \t — toggle the header and footer display.

Let's disable alignment, heading, and summary line:

```
=> \a \t
```

Output format is unaligned.
Tuples only is on.

```
=> SELECT name, setting, unit FROM pg_settings LIMIT 7;
```

```
allow_in_place_tablespaces|off|
allow_system_table_mods|off|
application_name|psql|
archive_cleanup_command||
archive_command|(disabled)|
archive_library||
archive_mode|off|
```

```
=> \a \t
```

Output format is aligned.
Tuples only is off.

This format is not the best for readability, but can be more suitable for automatic processing.

The extended format is best for displaying multiple columns for one or several records. It is switched on for the current command with the \gx flag at the end in place of a semicolon:

```
=> SELECT name, setting, unit, category, context, vartype,
       min_val, max_val, boot_val, reset_val
FROM pg_settings
WHERE name = 'work_mem' \gx
```

```
-[ RECORD 1 ]-----
name      | work_mem
setting   | 4096
unit      | kB
category  | Resource Usage / Memory
context   | user
vartype    | integer
min_val    | 64
max_val    | 2147483647
boot_val   | 4096
reset_val  | 4096
```

Or you can toggle it on and off with the \x flag. The \pset command displays all formatting options.

Interacting with the OS and Executing Scripts

psql can run shell commands:

```
=> \! pwd
```

```
/home/student
```

An SQL query can generate several other SQL queries and write them to a file using the \o[ut] command:

```
=> \a \t \pset fieldsep ' '
```

Output format is unaligned.
Tuples only is on.
Field separator is " ".

```
=> \o dev1_tools.sql
```

```
=> SELECT format('SELECT %L AS tbl, count(*) FROM %I;', tablename, tablename)
FROM pg_tables LIMIT 3;
```

Nothing is printed to the screen (to the standard output). Check the file:

```
=> \! cat dev1_tools.sql
```

```
SELECT 'pg_statistic' AS tbl, count(*) FROM pg_statistic;
SELECT 'pg_type' AS tbl, count(*) FROM pg_type;
SELECT 'pg_foreign_table' AS tbl, count(*) FROM pg_foreign_table;
```

Revert to default output and formatting.

```
=> \o \t \a
```

Tuples only is off.
Output format is aligned.

We can run the commands from the file using \i[nclude]:

```
=> \i dev1_tools.sql
```

```
      tbl      | count
-----+-----
pg_statistic |    409
(1 row)

      tbl      | count
-----+-----
pg_type      |    613
(1 row)

      tbl      | count
-----+-----
pg_foreign_table |      0
(1 row)
```

There are other ways to execute commands, including from files. After executing the commands, the psql session will be terminated:

- psql < filename
 - psql -f < filename
 - psql -c 'command' (one command per call)
-

psql Variables

Like shell, psql has its own variables.

Set a variable:

```
=> \set TEST Hi!
```

To replace a variable with its value, use the colon in front of its name:

```
=> \echo :TEST
```

Hi!

Unset a value:

```
=> \unset TEST
```

```
=> \echo :TEST
```

:TEST

Variables can be used, for example, to store the text of frequently used queries. Here is a query that returns a list of the five largest tables:

```
=> \set top5 'SELECT tablename, pg_total_relation_size(schemaname||'.'||tablename) AS bytes FROM pg_tables ORDER BY bytes DESC LIMIT 5;'
```

Now we can execute the query by just typing:

```
=> :top5
```

| tablename | bytes |
|----------------|---------|
| pg_proc | 1245184 |
| pg_rewrite | 745472 |
| pg_attribute | 720896 |
| pg_description | 630784 |
| pg_statistic | 294912 |

(5 rows)

You can add the command to set the value of top5 to the .psqlrc starter file in the user's home directory. Commands from .psqlrc are executed automatically every time psql starts.

If used with no parameters, \set displays all currently set variables and their values, including the built-in ones. You can check the help for the built-in variables:

```
\? variables
```

To exit psql, either use [quit], quit, or exit, or press Ctrl+D:

```
=> \q
```


Installing PostgreSQL from pre-built packages is the preferred installation type

Pre-packaged distributions bring some OS specifics that you should be aware of:

- how to start and stop the server
- location of configuration files
- location of the server message log

psql is a client application for working with PostgreSQL

1. Set the parameter *work_mem* to 8MB for all sessions.
Reload the configuration and verify that the changes have taken effect.
Restore the default value.
2. Put the command to create a table to the *ddl.sql* file.
Put the commands to insert rows into this table to the *populate.sql* file.
Run both scripts and verify that the table has been created and populated.
3. Find today's records in the server message log.

To launch `psql` in the terminal window, type `psql` without parameters.

```
student:~$ psql
```

It is convenient to have separate databases for tasks that are related to different topics:

```
student/student=# CREATE DATABASE tools_overview;
```

```
CREATE DATABASE
```

```
student/student=# \c tools_overview
```

```
You are now connected to database "tools_overview" as user "student".
```

```
student/tools_overview=#
```

1. Use the `ALTER SYSTEM` command.

2. If the output does not fit into one terminal screen, `psql` forwards it to the tool called `less` for page-by-page output. To return to the prompt, press **q**.

1. Configuration Parameters

Connect to the database:

```
student$ psql
```

Set the parameters:

```
=> ALTER SYSTEM SET work_mem = '8MB';
```

```
ALTER SYSTEM
```

Reload the configuration:

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

Check the parameter value:

```
=> \dconfig work_mem
```

List of configuration parameters

| Parameter | Value |
|-----------|-------|
| work_mem | 8MB |

```
(1 row)
```

Reset to the default value:

```
=> ALTER SYSTEM RESET work_mem;
```

```
ALTER SYSTEM
```

Apply changes:

```
=> SELECT pg_reload_conf();
```

```
pg_reload_conf
-----
t
(1 row)
```

2. Executing Scripts in psql

Write a command into 'ddl.sql' to create a table in PostgreSQL (you can use any text editor):

```
student$ cat > ~/ddl.sql <<EOF
CREATE TABLE keywords (
    word text,
    category text,
    description text
);
EOF
```

Write commands into 'populate.sql' to populate the table:

```
student$ cat > ~/populate.sql <<EOF
INSERT INTO keywords
    SELECT word, catcode, catdesc FROM pg_get_keywords();
EOF
```

Create a new database and connect to it:

```
=> CREATE DATABASE tools_overview;
```

```
CREATE DATABASE
```

```
=> \c tools_overview
```

You are now connected to database "tools_overview" as user "student".

Run the scripts and check the records in the table:

```
=> \i ddl.sql
```

```
CREATE TABLE
```

```
=> \i populate.sql
```

```
INSERT 0 471
```

```
=> SELECT * FROM keywords LIMIT 10;
```

| word | category | description |
|-----------|----------|-------------|
| abort | U | unreserved |
| absent | U | unreserved |
| absolute | U | unreserved |
| access | U | unreserved |
| action | U | unreserved |
| add | U | unreserved |
| admin | U | unreserved |
| after | U | unreserved |
| aggregate | U | unreserved |
| all | R | reserved |

(10 rows)

3. Viewing the Log

The log can be viewed in any text editor. Each log record starts with a date, contains the server process ID (with the packaged installation configuration), and may span several lines. The last records will be at the end of the file.

```
student$ tail /var/log/postgresql/postgresql-16-main.log
```

```
2025-11-27 14:26:39.316 MSK [44254] LOG:  database system is shut down
2025-11-27 14:26:40.321 MSK [46618] LOG:  starting PostgreSQL 16.11 (Ubuntu
16.11-1.pgdg24.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu
13.3.0-6ubuntu2~24.04) 13.3.0, 64-bit
2025-11-27 14:26:40.321 MSK [46618] LOG:  listening on IPv4 address "127.0.0.1", port 5432
2025-11-27 14:26:40.325 MSK [46618] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2025-11-27 14:26:40.341 MSK [46621] LOG:  database system was shut down at 2025-11-27
13:16:17 MSK
2025-11-27 14:26:40.367 MSK [46618] LOG:  database system is ready to accept connections
2025-11-27 14:26:43.184 MSK [46618] LOG:  received SIGHUP, reloading configuration files
2025-11-27 14:26:43.185 MSK [46618] LOG:  parameter "work_mem" changed to "8MB"
2025-11-27 14:26:43.572 MSK [46618] LOG:  received SIGHUP, reloading configuration files
2025-11-27 14:26:43.572 MSK [46618] LOG:  parameter "work_mem" removed from configuration
file, reset to default
```

The database can now be disconnected and deleted (see the Data Organization module for more)

```
=> \c postgres
```

You are now connected to database "postgres" as user "student".

```
=> DROP DATABASE tools_overview;
```

```
DROP DATABASE
```