psql PostgreSQL shell

START HERE

psql -h *<host>* -U *<user>* -d *<db>*
Connect

\x auto
Vertical output if wide

\l
List databases

\timing
Time each query

\dt
List tables

\copy *<table>* to '*<file>*' csv header
Export CSV

\d *<table>*
Describe a table

\q
Quit

Connect flags, URI, env → **Query buffer** SQL ending in ; → **Result** format with \pset → **Output** \o, \g, \copy

CONCEPTS

Backslash commands belong to psql, not SQL: they end at the line break and need no semicolon. SQL statements need a closing ;.

Missing connection flags fall back to PGHOST, PGPORT, PGUSER, PGDATABASE and ~/.pgpass, then to the local socket and your OS user name.

Put \set, \pset and \timing lines in ~/.psqlrc to apply them to every session; -X skips that file.

CONNECT

psql -h *<host>* -p *<port>* -U *<user>* -d *<db>*
Connect with explicit options

psql "postgresql://*<user>*@*<host>*:*<port>*/*<db>*"
Connect with a URI
Add ?sslmode=require for TLS.

psql *<db>*
Connect to a local database

psql -l
List databases and exit

psql -w ...
Never ask for a password
Fails if no password is found; useful in scripts.

psql "service=*<name>*"
Use a named connection service

\c *<db>* [*<user>*]
Switch database in the session

\conninfo
Show the current connection

\password [*<user>*]
Change a password safely
Hashed on the client, so it stays out of logs and history.

\q
Quit
Ctrl+D also quits.

INSPECT

\l
List databases

\dt
List tables
*Only the schemas on your search_path; use \dt *.* for all.*

INSPECT (cont.)

\d *<table>*
Columns, indexes and constraints

\d+ *<table>*
Same plus size and storage
Slower on large catalogs.

\dn
List schemas

\du
List roles

\dv
List views

\di
List indexes

\df *<pattern>*
List functions

\dx
List installed extensions

\dp *<table>*
Show access privileges

\sf *<function>*
Show a function's source

\sv *<view>*
Show a view's definition

\dt *<schema>*.*
Tables in one schema
*Patterns: * and ? are wildcards; quote names to keep case.*

RUN SQL

psql -c "*<sql>*"
Run one statement and exit

psql -f *<file.sql>*
Run a script file

\i *<file>*
Run a file inside psql

\ir *<file>*
Run a file relative to this script

\e
Edit the query in $EDITOR

\g
Run the buffer again

\gx
Run once with vertical output

\p
Print the query buffer

\r
Clear the query buffer

\watch *<seconds>*
Repeat the query every n seconds
Stop with Ctrl+C.

RUN SQL (cont.)

\gexec
Run each result cell as SQL
Generate statements with format(), then execute them.

psql -1 -f *<file.sql>*
Run a script in one transaction
Pair with -v ON_ERROR_STOP=1 to roll back on error.

\ef *<function>*
Edit a function definition

\s
Show command history

OUTPUT

\x auto
Expanded mode only when wide

\x
Toggle vertical output

\timing
Toggle query timing

\pset null '*<text>*'
Show NULL as text

\pset format *<fmt>*
Choose aligned, csv, html or more

\pset pager off
Disable the pager

\t
Toggle header and footer off

psql -A -t -c "*<sql>*"
Bare values for scripts

psql --csv -c "*<sql>*"
CSV output

\o *<file>*
Send results to a file
\o with no argument switches back to the screen.

\crosstabview
Pivot the result

\echo *<text>*
Print text to stdout

IMPORT & EXPORT

\copy *<table>* from '*<file>*' csv header
Load a CSV file
Runs client-side: reads your local file, no superuser needed.

\copy *<table>* to '*<file>*' csv header
Export a table to CSV

\copy (*<query>*) to '*<file>*' csv header
Export a query result

IMPORT & EXPORT (cont.)

\copy *<table>* (*<cols>*) from stdin
Load from standard input

\copy ... with (format csv, delimiter ';')
Set CSV options

\g (format=csv) *<file>*
Write one result as CSV

\copy or COPY?

Form	File lives on
`\copy`	your machine (client)
`COPY`	database server

VARIABLES & SCRIPTING

\set *<name>* *<value>*
Set a psql variable

psql -v *<name>*=*<value>*
Set a variable from the shell

SELECT :'*<name>*';
Use a variable as a quoted literal
Use :name raw and :"name" as an identifier.

\set ON_ERROR_STOP on
Abort a script on first error

\set AUTOCOMMIT off
Open transactions implicitly

\set ECHO queries
Print each statement sent

\gset [*<prefix>*]
Store a one-row result in variables

\if *<cond>* ... \endif
Conditional script parts

\? / \h *<command>*
Help for psql or SQL

RECIPES

Export a query to CSV

```
\copy (SELECT * FROM <table>
WHERE <cond>) to 'out.csv' csv
header
```

Safe migration script

```
psql -X -1 -v ON_ERROR_STOP=1 -f
migrate.sql <db>
```

Bare value into a shell variable

```
n=$(psql -At -c "SELECT count(*)
FROM <table>" <db>)
```

Handy ~/.psqlrc

```
-- ~/.psqlrc
\set QUIET 1
\timing on
\x auto
\pset null '(null)'
\unset QUIET
```