usql Universal SQL CLI

START HERE

`usql <dsn>`
Connect to any database

`usql sq://app.db`
Create or open SQLite

`usql <dsn> -c "<sql>"`
Run one statement

`usql <dsn> -f <file>`
Run a script file

`\dt`
List tables

`\d <name>`
Describe a table

`\x auto`
Expanded output when wide

`\q`
Quit

CONCEPTS

usql is psql's command set for many databases. The scheme of the DSN picks the driver (pg, my, ms, or, sq ...). Which drivers exist depends on the build: check with \drivers.

DSN driver://user:pass@host/db
↓ parses
Driver chosen by scheme or alias
↓ connects
Prompt SQL plus \commands
↓ prints
Output table, CSV, JSON, HTML

Common schemes

Database	Scheme	Alias
PostgreSQL	postgres	pg
MySQL	mysql	my
SQL Server	sqlserver	ms
Oracle	oracle	or
SQLite3	sqlite3	sq
DuckDB	duckdb	dk
ClickHouse	clickhouse	ch

CONNECT

`usql pg://user:pass@host/db`
Connect to PostgreSQL

`usql my://user:pass@host:3306/db`
Connect to MySQL

CONNECT (cont.)

`usql sq://<path>.db`
SQLite, create if missing
Without a scheme the file must already exist; sq: or file: forces creation.

`usql <path>`
Open an existing file or socket
A regular file opens as SQLite or DuckDB, a directory as Postgres, a socket as MySQL.

`\c <dsn or name>`
Switch connection in the prompt

`\drivers`
List drivers built in

RUN AND SCRIPT

`usql <dsn> -c "<sql>"`
Run one command and exit
Also accepts a backslash command, e.g. -c '\dt'.

`usql <dsn> -f <file>`
Run a file and exit

`\i <file>`
Run a file from the prompt

`\g`
Run the query buffer (same as ;)

RUN AND SCRIPT (cont.)

`\p`
Print the query buffer

`\r`
Clear the query buffer

`\e`
Edit the buffer in $EDITOR

`\begin / \commit / \rollback`
Manual transaction

INSPECT THE SCHEMA

`\dt[S+] [<pattern>]`
List tables

`\d[S+] [<name>]`
List relations or describe one

`\dn [<pattern>]`
List schemas

`\l[+]`
List databases

OUTPUT FORMATS

`-C`
CSV output

OUTPUT FORMATS (cont.)

`-J`
JSON output

`-x`
Expanded output (one field per line)

`-t`
Rows only, no headers

`\x [on|off|auto]`
Toggle expanded output

`\G`
Run and print vertically

VARIABLES AND SCRIPTING

`\set <name> <value>`
Set a script variable

`:<name>` / `:'<name>'` / `:"<name>"`
Interpolate raw / as string / as identifier
Not interpolated inside a quoted string literal.

`\gexec`
Run each result value as a statement

`\if <expr> / \elif / \else / \endif`
Conditional blocks in scripts

VARIABLES AND SCRIPTING (cont.)

`` `<command>` ``
Backtick: use shell output as an argument

`\cset <name> <dsn>`
Save a named connection (this session)

`\? [options|variables]`
Help on commands, flags, variables

`help`
Short help at the prompt

COPY AND CONFIG

`\copy <src> <dst> <query> <table>`
Copy query results between databases
Source and destination are DSNs or names. Add (col,...) after the table to name columns.

`~/.config/usql/config.yaml`
Config file (Linux)

`connections: { <name>: <dsn> }`
Named DSNs in config.yaml

`init: |`
Startup script in config.yaml

COPY AND CONFIG (cont.)

`usql --version`
Show version

`usql --help`
Show all flags

RECIPES

SQLite to CSV

```
usql sq://app.db -C -c "select * from users" > users.csv
```

JSON for jq

```
usql pg://localhost/app -J -c "select id, name from users" | jq .
```

Copy a table across engines

```
usql -c "\copy pg://localhost/src my://localhost/dst 'select * from t' t"
```

Safe script run

```
usql pg://localhost/app -1 -v id=42 -f fix.sql
```

Named connections

```
# ~/.config/usql/config.yaml
connections:
  prod: pg://app@db.example.com/app
  dev: sq://dev.db

usql prod -c '\dt'
```