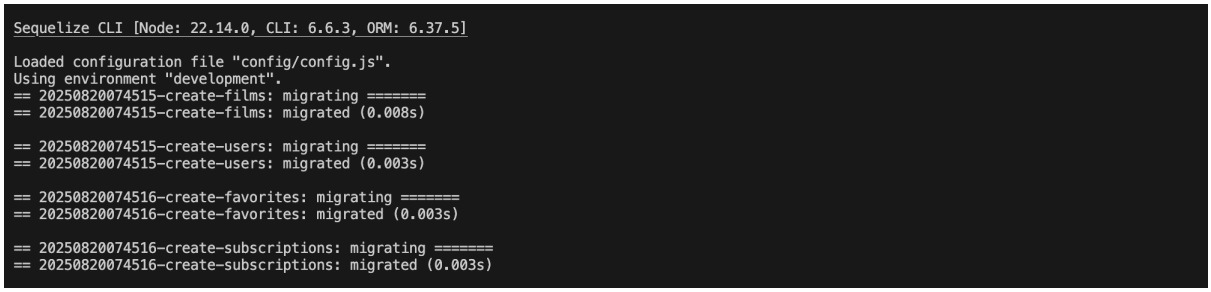
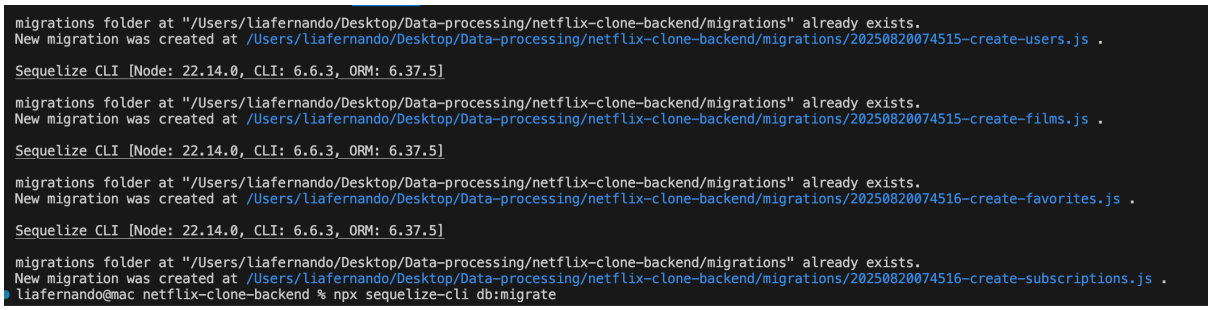
Database Proof Document

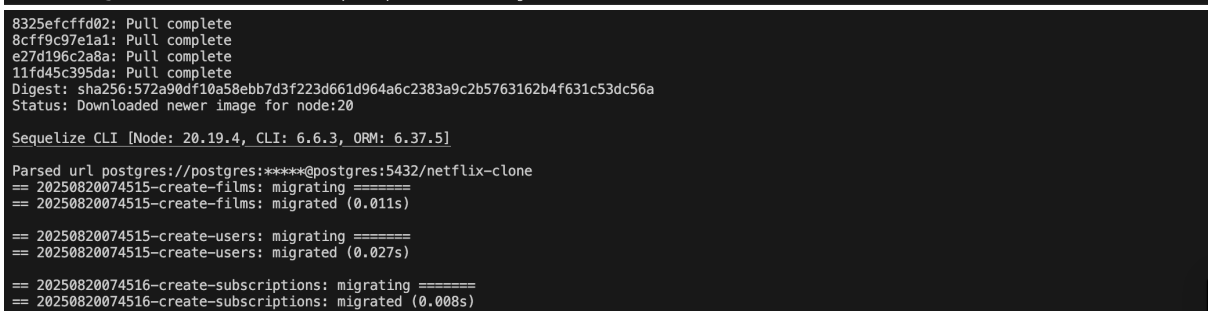
Database Persistence proof with the:

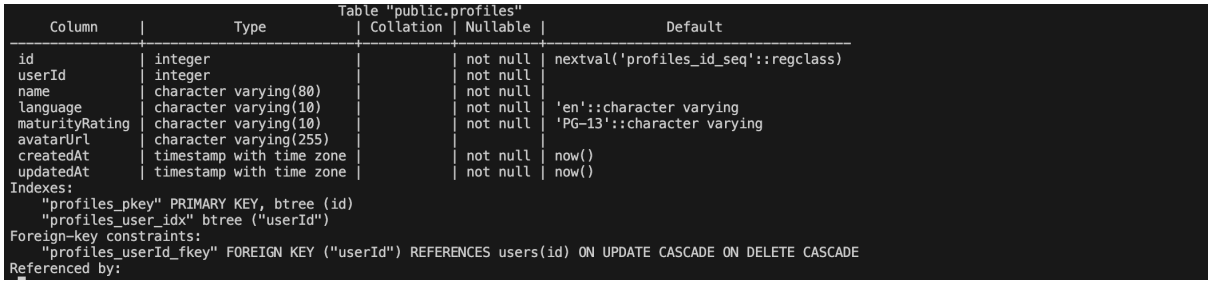
- User saved in postgres
- Login + JWT authentication
- The authenticated routes (current/user)
- Publics routes (/users/:id)

Migrations proof









Datbase tables

```
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.5.2
npm notice To update run: npm install -g npm@11.5.2
npm notice
liafernando@mac netflix-clone-backend % docker exec -it uni-postgres psql -U postgres -d netflix-clone -c '\dt'

             List of relations
 Schema |     Name      | Type  |  Owner
--------+---------------+-------+----------
 public | SequelizeMeta | table | postgres
 public | films         | table | postgres
 public | subscriptions | table | postgres
 public | users         | table | postgres
(4 rows)
```

Seeders for the tables

```
npx sequelize-cli seed:generate --name demo-favorites


Sequelize CLI [Node: 22.14.0, CLI: 6.6.3, ORM: 6.37.5]

seeders folder at "/Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders" already exists.
New seed was created at /Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders/20250820095241-demo-users.js .

Sequelize CLI [Node: 22.14.0, CLI: 6.6.3, ORM: 6.37.5]

seeders folder at "/Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders" already exists.
New seed was created at /Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders/20250820095241-demo-films.js .

Sequelize CLI [Node: 22.14.0, CLI: 6.6.3, ORM: 6.37.5]

seeders folder at "/Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders" already exists.
New seed was created at /Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders/20250820095241-demo-favorites.js .
```

```
seeders folder at "/Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders" already exists.
New seed was created at /Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders/20250822003129-seed-films.js .

Sequelize CLI [Node: 22.14.0, CLI: 6.6.3, ORM: 6.37.5]

seeders folder at "/Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders" already exists.
New seed was created at /Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders/20250822003130-seed-genres.js .

Sequelize CLI [Node: 22.14.0, CLI: 6.6.3, ORM: 6.37.5]

seeders folder at "/Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders" already exists.
New seed was created at /Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders/20250822003130-seed-series.js .

Sequelize CLI [Node: 22.14.0, CLI: 6.6.3, ORM: 6.37.5]

seeders folder at "/Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders" already exists.
New seed was created at /Users/liafernando/Desktop/Data-processing/netflix-clone-backend/seeders/20250822003131-seed-seasons.js .
```

```
Commands:
  sequelize db:migrate                        Run pending migrations
  sequelize db:migrate:schema:timestamps:add  Update migration table to have timestamps
  sequelize db:migrate:status                 List the status of all migrations
  sequelize db:migrate:undo                   Reverts a migration
  sequelize db:migrate:undo:all               Revert all migrations ran
  sequelize db:seed                           Run specified seeder
  sequelize db:seed:undo                      Deletes data from the database
  sequelize db:seed:all                       Run every seeder
  sequelize db:seed:undo:all                  Deletes data from the database
  sequelize db:create                         Create database specified by configuration
  sequelize db:drop                           Drop database specified by configuration
  sequelize init                              Initializes project
  sequelize init:config                       Initializes configuration
  sequelize init:migrations                   Initializes migrations
  sequelize init:models                       Initializes models
  sequelize init:seeders                      Initializes seeders
```

```
liafernando@mac netflix-clone-backend % ls -1 seeders

20250820095241-demo-favorites.js
20250820095241-demo-films.js
20250820095241-demo-users.js
20250822003129-seed-films.js
20250822003130-seed-genres.js
20250822003130-seed-series.js
20250822003131-seed-episodes.js
20250822003131-seed-seasons.js
20250822103946-seed-profiles.js
20250822103947-seed-watch-history.js
```

```
docker exec -it uni-postgres psql -U postgres -d netflix-clone -c 'SELECT * FROM watchlists;'
docker exec -it uni-postgres psql -U postgres -d netflix-clone -c 'SELECT * FROM watch_histories;'

 id | userId | name | language | maturityRating | avatarUrl |          createdAt          |          updatedAt
----+--------+------+----------+----------------+-----------+-----------------------------+-----------------------------
  1 |      1 | Main | en       | PG-13          |           | 2025-08-22 13:24:11.473358+00 | 2025-08-22 13:24:11.473358+00
(1 row)

 profileId | filmId | episodeId | createdAt | updatedAt
-----------+--------+-----------+-----------+-----------
(0 rows)

 id | profileId | filmId | episodeId | progress |        lastWatchedAt        |          createdAt          |          updatedAt
----+-----------+--------+-----------+----------+-----------------------------+-----------------------------+-----------------------------
  1 |         1 |      1 |           |      600 | 2025-08-22 13:24:11.653827+00 | 2025-08-22 13:24:11.653827+00 | 2025-08-22 13:24:11.653827+00
(1 row)
```

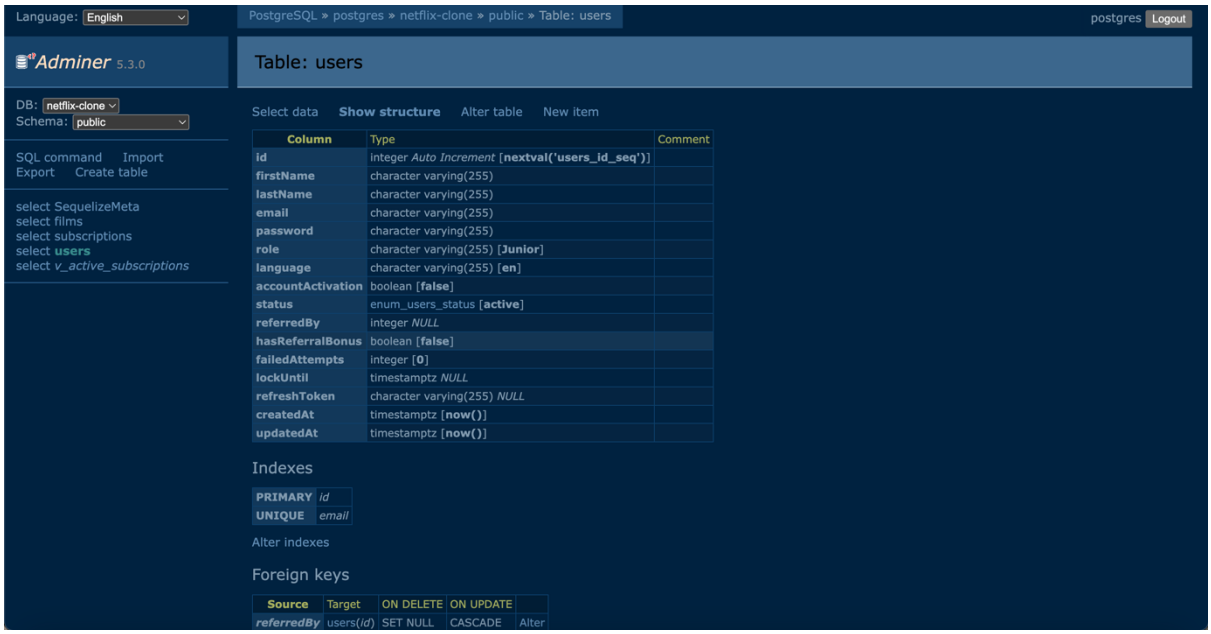
Roles and permissions proof

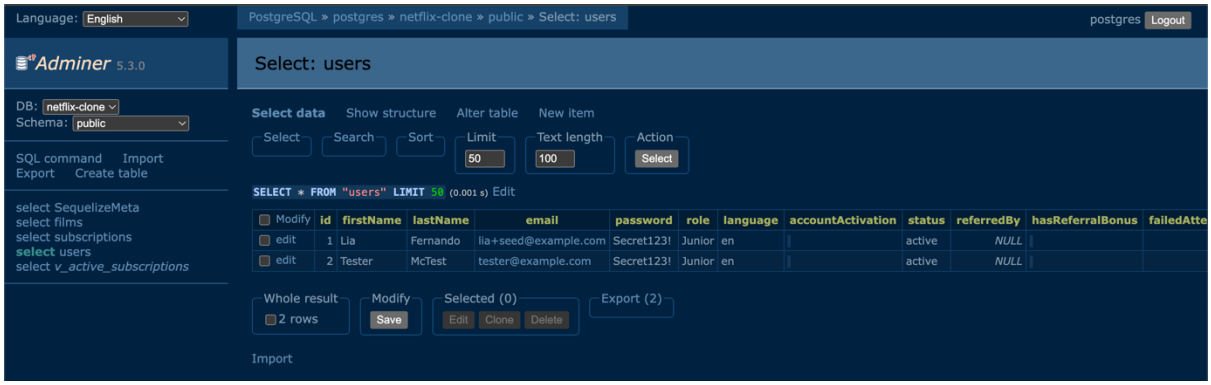
```
netflix-clone=# \du
\z users
                                   List of roles
   Role name    |                         Attributes
----------------+------------------------------------------------------------
 app_user       |
 postgres       | Superuser, Create role, Create DB, Replication, Bypass RLS
 reporting_user |

                                  Access privileges
 Schema | Name  | Type  |     Access privileges     | Column privileges | Policies
--------+-------+-------+---------------------------+-------------------+----------
 public | users | table | postgres=arwdDxt/postgres+|                   |
        |       |       | app_user=arwd/postgres   +|                   |
        |       |       | reporting_user=r/postgres |                   |
(1 row)

netflix-clone=#
```

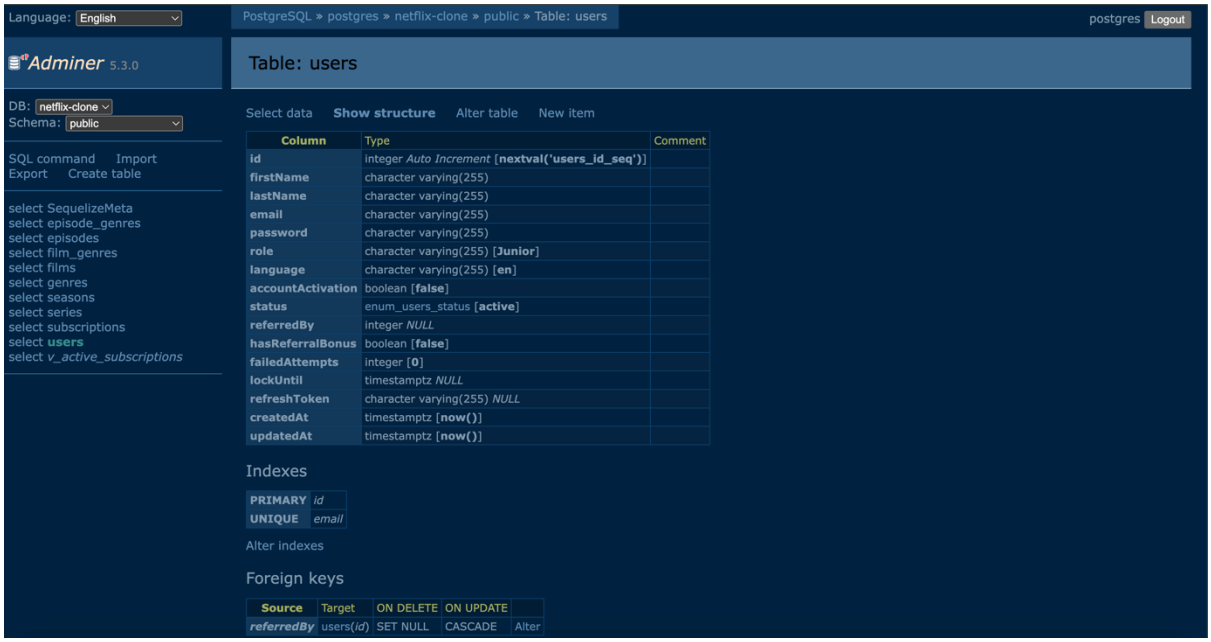
Active_subscriptions proof

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

 subscription_id | user_id | firstName | lastName |       email         | plan  | status |         startDate          |          endDate
-----------------+---------+-----------+----------+---------------------+-------+--------+----------------------------+---------------------------
--
               1 |       1 | Lia       | Fernando | lia+seed@example.com | Basic | active | 2025-08-21 07:49:25.80102+00 | 2025-09-20 07:49:25.80102+0
0
```

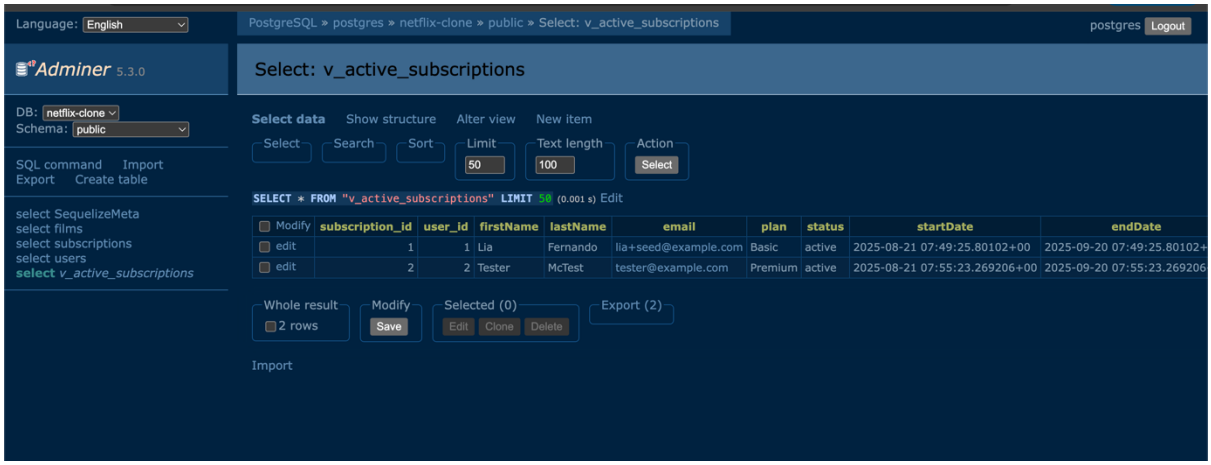
Database tables user







Active Subscriptions



Backup Recovery

```
● liafernando@mac Data-processing % ls -lh ./backups

  total 32
  -rw-r--r--@ 1 liafernando  staff    12K Aug 21 11:01 netflix-clone_2025-08-21_11-01.sql
○ liafernando@mac Data-processing % ▌
```

```
● liafernando@mac Data-processing % head -20 ./backups/netflix-clone_2025-08-21_11-01.sql

  --
  -- PostgreSQL database dump
  --

  \restrict eJKepT22Y9kfU85wmCGLnNtjCpcote7cN2aTsPC929qpUNABL7I8foGSnKl4UhO

  -- Dumped from database version 16.10 (Debian 16.10-1.pgdg13+1)
  -- Dumped by pg_dump version 16.10 (Debian 16.10-1.pgdg13+1)

  SET statement_timeout = 0;
  SET lock_timeout = 0;
  SET idle_in_transaction_session_timeout = 0;
  SET client_encoding = 'UTF8';
  SET standard_conforming_strings = on;
  SELECT pg_catalog.set_config('search_path', '', false);
  SET check_function_bodies = false;
```

```
\restrict eJKepT22Y9kfU85wmCGLnNtjCpcote7cN2aTsPC929qpUNABL7I8foGSnKl4UhO

-- Dumped from database version 16.10 (Debian 16.10-1.pgdg13+1)
-- Dumped by pg_dump version 16.10 (Debian 16.10-1.pgdg13+1)

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;
```

Backup recovery command

```
● liafernando@mac Data-processing % mkdir -p ./backups
● liafernando@mac Data-processing % docker exec uni-postgres pg_dump -U postgres netflix-clone > ./backups/netflix-clone_$(date +%F_%H-%M).sql
```

Triggers

```
 id             | integer                  |           | not null | nextval('films_id_seq'::regclass)
 title          | character varying(255)   |           | not null |
 description    | text                     |           |          |
 category       | character varying(50)    |           |          |
 releaseDate    | date                     |           |          |
 duration       | integer                  |           |          |
 ageLimit       | character varying(20)    |           | not null | 'PG-13'::character varying
 classification | character varying(50)    |           |          |
 quality        | character varying(20)    |           |          |
 genre          | character varying(100)   |           |          |
 createdAt      | timestamp with time zone |           | not null | now()
 updatedAt      | timestamp with time zone |           | not null | now()
Indexes:
    "films_pkey" PRIMARY KEY, btree (id)
Triggers:
    trg_films_updated_at BEFORE UPDATE ON films FOR EACH ROW EXECUTE FUNCTION set_updated_at()
```

```
netflix-clone=# SELECT event_object_table AS table_name,
       trigger_name,
       event_manipulation AS event,
       action_timing AS timing
FROM information_schema.triggers
ORDER BY table_name, trigger_name;
  table_name   |         trigger_name          | event  | timing
---------------+-------------------------------+--------+--------
 films         | trg_films_updated_at          | UPDATE | BEFORE
 subscriptions | trg_subscriptions_updated_at  | UPDATE | BEFORE
 users         | trg_users_updated_at          | UPDATE | BEFORE
(3 rows)
```

```
                                                                          List of functions
 Schema |      Name      | Result data type | Argument data types | Type | Volatility | Parallel |  Owner   | Security | Access privileges | Language
| Internal name | Description
--------+----------------+------------------+---------------------+------+------------+----------+----------+----------+-------------------+----------
+---------------+-------------
 public | set_updated_at | trigger          |                     | func | volatile   | unsafe   | postgres | invoker  |                   | plpgsql
|               |
(1 row)
```

```
      action_timing AS timing
FROM information_schema.triggers
ORDER BY table_name, trigger_name;
  table_name   |         trigger_name         | event  | timing
---------------+------------------------------+--------+--------
 films         | trg_films_updated_at         | UPDATE | BEFORE
 subscriptions | trg_subscriptions_updated_at | UPDATE | BEFORE
 users         | trg_users_updated_at         | UPDATE | BEFORE
(3 rows)

netflix-clone=# \df+ set_updated_at
netflix-clone=# \sf set_updated_at
CREATE OR REPLACE FUNCTION public.set_updated_at()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
BEGIN NEW."updatedAt" = NOW(); RETURN NEW; END; $function$
netflix-clone=#
```

```
-- do an update (no need to change much)
UPDATE films SET title = title || ' ' WHERE id = 1;

-- see updatedAt changed
SELECT id, "updatedAt" FROM films WHERE id = 1;
 id |           updatedAt
----+-------------------------------
  1 | 2025-08-22 00:10:20.377791+00
(1 row)

UPDATE 1
 id |           updatedAt
----+-------------------------------
  1 | 2025-08-22 01:42:11.633806+00
(1 row)
```

CRUD

Create

```
netflix-clone=# -- Create a demo film and store ID into variable
INSERT INTO films
(title, description, category, "releaseDate", duration, "ageLimit", classification, quality, genre)
VALUES
('Demo Film', 'Inserted via CRUD demo', 'movie', '2024-01-01', 120, 'PG-13', 'feature', 'HD', 'Drama')
RETURNING id, title \gset

-- Show what was created
\echo 'Created:' :id :title
\gset: extra argument "Show" ignored
\gset: extra argument "what" ignored
\gset: extra argument "was" ignored
\gset: extra argument "created" ignored
invalid variable name: "--id"
INSERT 0 1
Created: 10 Demo Film
```

```
netflix-clone=# -- C = CREATE (insert a demo film)
INSERT INTO films
(title, description, category, "releaseDate", duration, "ageLimit", classification, quality, genre)
VALUES
('Demo Film', 'Inserted via CRUD demo', 'movie', '2024-01-01', 120, 'PG-13', 'feature', 'HD', 'Drama')
RETURNING id, title \gset
INSERT 0 1
netflix-clone=# \echo :id :title
10 Demo Film
```

Read

```
netflix-clone=# SELECT id, title, category, "releaseDate"
FROM films
ORDER BY id DESC
LIMIT 5;
 id |      title      | category | releaseDate
----+-----------------+----------+-------------
 11 | Demo Film       | movie    | 2024-01-01
 10 | Demo Film       | movie    | 2024-01-01
  9 | Demo Film       | movie    | 2024-01-01
  8 | Stranger Things | show     | 2016-07-15
  7 | Breaking Bad    | show     | 2008-01-20
(5 rows)
```

Update

```
SELECT id, title, "updatedAt"
 FROM films
 WHERE id = :id;
 id |        title        |           updatedAt
----+---------------------+-------------------------------
 10 | Demo Film (Updated) | 2025-08-22 02:01:52.133401+00
(1 row)

UPDATE 1
 id |        title        |           updatedAt
----+---------------------+-------------------------------
 10 | Demo Film (Updated) | 2025-08-22 02:01:52.133401+00
(1 row)
```

Delete

```
-- Verify deletion
SELECT id, title
FROM films
WHERE id = :id;
 id |        title
----+---------------------
 10 | Demo Film (Updated)
(1 row)

DELETE 1
 id | title
----+-------
(0 rows)
```