SQL Assignment

Contents

1 Setup

Nothing to hand in for this section, just do what is described here.

In this assignment, we are going to work with SQL. For practicing SQL, you will be installing the Postgres Relational DBMS on your own computer system and interact with it using a SQL Client Interface. We provide install instructions for both Windows and MacOS-based computers.

PostgreSQL is an open-source relational database system, that also offers support for document and key-value storage. It has been released for the first time more than 20 years ago. It has high conformance with the SQL standard, and it is widely used in a number of industrial and academic projects.

Installing and using Postgres can be somewhat tricky for the unexperienced user, but that makes it an even more valuable exercise. Having interacted and experienced such (admittedly, sometimes not very user-friendly software) is one of the learning goals for this assignment!

Please check out the following:

- Postgres installation instructions ("Database Installation and Import Guide" provided as an additional document as part of this course, see here Database Installation and Data Import Instructions)
- pgAdmin (pgAdmin - PostgreSQL Tools), an open-source GUI Client for PostgreSQL (there are many other SQL GUI Clients; you can use whichever you like, but we have good experiences with PGAdmin.)

2 Data

Nothing to hand in for this section, just do what is described here.

We provide some data sets which are used in this assignment, but which can also be used to play around with. We highly encourage you to just experiment a little, and also read up on some of the many SQL tutorials available on the Web (like https://www.w3schools.com/sql/ or https://www.sqltutorial.org/) .

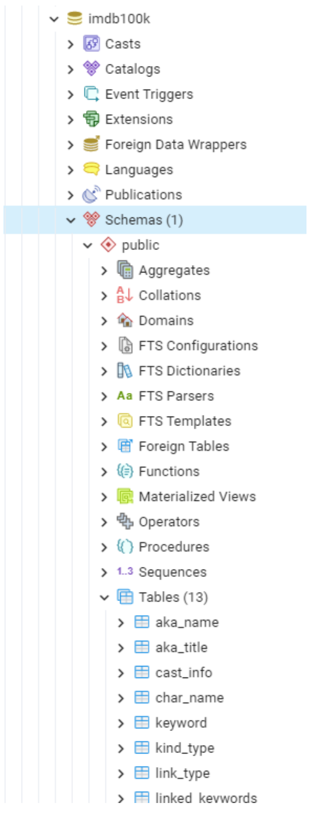
This assignment uses a simplified and very reduced version of the IMDB movie database (found here: Database Dump File). You can download the data dump file from the learning platform and follow the "Database Installation and Import Guide".

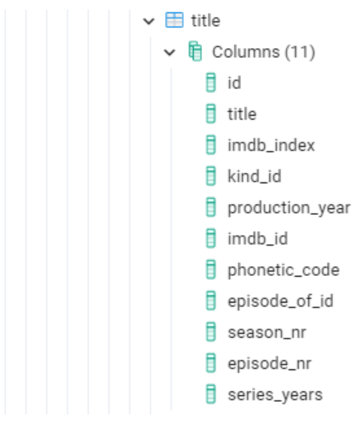
3 Exploring the Schema

Nothing to hand in for this section, just do what is described here.

When faced with a new database, it is a is a good idea to start exploring the schema a bit. We use pgAdmin for that.

You can explore by manually browsing through the table definitions in PostgreSQL (find the Schema/public/Tables entry to see all tables, click on it's columns to see the columns.)





You can also open the Query Tool (right-click in the browser, and select Query Tool from the context menu).

Try a few queries (copied from the Installation guide):

1. Find the names of all people in the database (**Warning**: remember there are 100k movies, which means there is a lot of people in the database, so it's recommended to put a **limit** on the query):

```
SELECT name
FROM person;

SELECT name
FROM person
LIMIT 5;
```

2. Return the imdb_index and production_year of all items called The Matrix

```
SELECT imdb_index, production_year
FROM title
WHERE title = 'The Matrix';
```

3. Return the id, title and millenium_age (the production year minus 2000) of all items called The Matrix

```
SELECT id, title, production_year - 2000 AS millenium_age
FROM title
WHERE title = 'The Matrix';
```

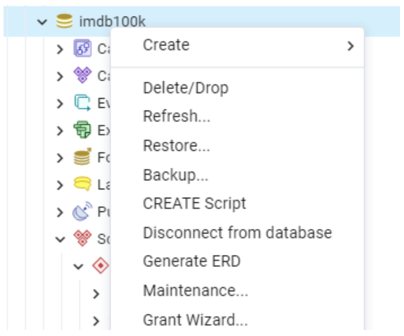
4. Return the title, production_year and kind of all items called The Matrix

```
SELECT t.title, t.production_year, k.kind
FROM title AS t
JOIN kind_type AS k
ON t.kind_id = k.id
WHERE t.title = 'The Matrix';
```

4 Visualizing the Schema

Nothing to hand in for this section, just do what is described here.

You can use pgAdmin to create a graphical representation of the logical schema. Do so by right-clicking on the database in the browser, and select “Generate ERD” (ERD stands for Entity-Relationship Diagram, and is a common technique for database visualization) .



Try to understand how the schema works (it is not uncommon to have a database without documentation; so this task is unfortunately quite representative of real-world scenarios. Also, we are aware that the schema is not very well designed. Unfortunately, also this is normal in many cases.)

Some hints:

- *Person* stores persons involved in making a movie (e.g., actors)
- *Title* stores individual movies/TV Show Episodes.
- *aka_name* stands for “also known as name”
- *Person* is linked to *Title* via the table *cast_info*.
  cast_info also links to additional inforormation like how and for while role that person was involved with that title, and which character they played”

5 Design some SQL Queries

In this section, you design SQL queries fulfilling the given descriptions.

You can create a TXT file using a text editor. In that TXT file, put your query code, and a marker what that query is for. For example:
“
```
5.1:
SELECT * FROM …
```
”

5.1 The Matrix

Write a query which finds all titles which are called “The Matrix” and which are movies (and not TV shows).

Hint:

- Movies are tytles that have a *kind_id* of 1. You can check the content of the *kind_type* table to see what other kind types are used.

5.2 People of the Matrix

Write a query which finds all persons which are involved with the “The Matrix” movie.

Hints:

- You do this by joining the tables *title, cast_info,* and *person.* If you cannot recall how to join, look it up here: SQL Joins (w3schools.com). Selecting “The Matrix” as in the previous task from the join result.
- You can use the diagram generated for exercise 4 to find out which attributes to use for the join. Or check out Foreign Key constraints of the tables by inspecting their properties.
- You can return all attributes for now (`SELECT * FROM`)

5.3 Roles and People

Write a query which lists the names of all persons acting in the “The Matrix” movie and the name of the character they play in ascending order of their appearance on screen.

Hints:

- You can obviously expand the solution of the previous exercise
- ONLY return the person name and character name
- Character names are stored in the *char_name* table
- The attribute *nr_order* in *cast_info* stored the order in which characters appear on screen. Look up online how to order SQL result sets! (Note: By default, SQL results are in random orders unless explicitly ordered)

5.4 What about people who don’t play a role?

Write a query which lists the names of all persons acting in the “The Matrix” movie and the name of the character they play in ascending order of their appearance on screen (as in the last exercise). Also include other people who are involved with that movie and who are not actors acting a character. For each person, also include their role (like actor, writer, director, etc.).

Hints:

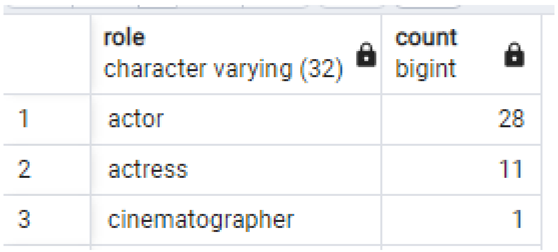
- You can expand the solution of the previous exercise
- ONLY return the person’s name, character name (if applicable), and the role of that person (actor, writer, director, etc. Info on this is found in the *role_type* table)
- In the last exercise, you likely used `JOIN…ON` for including the character names. `JOIN_ON` only includes rows which do have a suitable join partner (in this case: only considers persons who DO play a character). Directors or writers do not play a character. You need an outer join to also include people in your result without a linked entry in the *char_name* table. See SQL LEFT JOIN Keyword (w3schools.com) or SQL FULL OUTER JOIN Keyword (w3schools.com)

5.5 Some simple people stats

Create a list of of all roles (actor, writer, director, etc.) for the movie “The Matrix” with a count statistic of how many people had that role.

Hints:

- You can expand the solution of the previous exercise
- ONLY return the name of roles (actor, writer, composer, etc.), and a count of how many people had that role for the Matrix movie.



	role character varying (32)	count bigint
1	actor	28
2	actress	11
3	cinematographer	1

- You need `GROUP BY` and `COUNT` for this to work! See SQL GROUP BY Statement (w3schools.com)

6 Export Data as CSV

Nothing to hand in for this section, just do what is described here.

Export the result of your favorite SQL exercise in the previous section as a CSV file. Inspect that CSV file in a suitable editor.

Hints:

- You can do that from the data output pane of pgAdmin.