COMP3311 24T1 Database Systems

Assignment 1
SQL Views, Functions,
And The IMDB

Last updated: **Friday 15th March 5:28pm**

Most recent changes are shown in red ... older changes are shown in brown.

[Assignment Spec] [SQL Schema] [SQL Data] [Sample Outputs] [Testing] [Fixes+Updates]

Aims

This assignment aims to give you practice in

- Reading, Understanding, and Manipulating a small relational schema (IMDB)
- Implementing SQL Views to satisfy requests for information
- Implementing SQL functions to satisfy requests for information
- Implementing PLpgSQL functions to satisfy requests for information

The goal is to build some useful data access operations on the Internet Movie Database (IMDB), which contains a wealth of information about movies, actors, etc. One aim of this assignment is to use SQL queries (packaged as views) to extract such information. Another is to build SQL/PlpgSQL functions that can support higher-level activities, such as might be needed in a Web Interface.

Summary

Submission:	Login to Course Web Site > Assignments > Assignment 1 > Submit > upload ass1.sql or on a CSE server run the command `give cs3311 ass1 ass1.sql`
Required Files:	`ass1.sql` (contains both SQL views and PLpgSQL functions)
Deadline:	21:59:59 Friday 22 March
Marks:	**13 marks** toward your total mark for this course
Late Penalty:	0.2 percent off the raw mark for each hour late, for 5 days Any submission after 5 days scores 0 marks This is the UNSW standard late penalty.

How to do this assignment:

- read this specification **carefully and completely**
- create a directory for this assignment
- copy the supplied files into this directory
- login to **vxdb2** and run your PostgreSQL server
- load the database and start exploring

- complete the tasks below by editing ass1.sql
- test your work on vxdb2
- submit ass1.sql via WebCMS or give (you can submit multiple times, only your last submission will be marked)

Note that you can put the files wherever you like; they do not have to be under your /localstorage directory. You also edit your SQL files on hosts other than vxdb2. The only time that you need to use vxdb2 is to manipulate your database.

Introduction

The Internet Movie Database (IMDB) is a huge collection of information about all kinds of video media. It has data about most movies since the dawn of cinema, but also a vast amount of information about TV series, documentaries, short films, etc.
Similarly, it holds information about the people who worked on and starred in these video artefacts. It also hold viewer ratings and critics reviews for video artefacts as well as a host of other trivia (e.g. bloopers).

The full IMDB database is way too large to let you all build copies of it, so we have have created a cut-down version of the database that deals with movies only.

Some comments about the data in our copy of IMDB: Each database dump file contains 2500 random movies plus the people who worked on them.

Assignment Setup

In addition to the database dump file, available on the `SQL Data` page, you are also provided a template SQL file `ass1.sql` which you will need to edit to complete the assignment. You can download the file here or use the command below to copy it into your CSE account.

```
$ cp /web/cs3311/current/assignments/ass1/files/ass1.sql .
```

Your Tasks

Answer each of the following questions by typing SQL or PLpgSQL into the `ass1.sql` file. You may find it convenient to work on each question in a temporary file, so that you don't have to keep loading all of the other views and functions each time you change the one you're working on. Note that you can add as many auxiliary views and functions to `ass1.sql` as you want. However, make sure that everything that's required to make all of your views and functions work is in the `ass1.sql` file before you submit.

Q0 (2 marks) - Style Marks

Given that you've already taken multiple programming courses, we should be able to assume that you'll express your code with good style conventions. But, just in case ...

You must ensure that your SQL queries follow a consistent style. The one I've been using in the lectures is fine. An alternative, where the word JOIN comes at the start of the line, is also OK. The main thing is to choose one style and use it consistently.

Similarly, PLpgSQL should be laid out like you would lay out any procedural programming language. E.g. bodies of loops should be indented from the FOR or WHILE statement that introduces them. E.g. the body of a function should be indented from the BEGIN...END.

Ugly and/or inconsistent layout of SQL queries and PLpgSQL functions will be penalised.

Q1 (1 marks) - SQL View

Write a SQL View, called Q1(Title, Year, Votes), that:
Retrieves the 10 movies with the highest number of votes.
Your view should include the Primary Title, Release Year, and number of Votes for each movie.
Your results should be ordered by the number of votes, from highest to lowest.

You can assume that no two movies will have the same number of votes.

You can define additional helper views if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.
You cannot define SQL functions or PLpgSQL functions in this question.

Example output for this question can be found on the Examples page.

Q2 (2 marks) - SQL View

Write a SQL View, called Q2(Name, Title), that:
Retrieve the names of people who have a year of death recorded in the database and are well known for their work in movies released between 2017 and 2019.
Your view should include the Name of each person, along with the Primary Title of the movie they are well known for.
Your results should be ordered by the name of the person, in alphabetical order.

The Credits and Roles tables record *everyone* who worked on a movie, these are not the tables we are interested in for this question.
The Principals table records the people who are *well known* for their work on a movie

You can define additional helper views if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.

Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.
You cannot define SQL functions or PLpgSQL functions in this question.

Example output for this question can be found on the Examples page.

Q3 (3 marks) - SQL View

Write a SQL View, called Q3(Name, Average), that:
Retrieves the genres with an average rating not less than 6.5 and with more than 60 released movies.
Your view should include the Genre Name and the Average Rating, rounded to two decimal places, for each genre.
Your results should be ordered by the average rating, from highest to lowest.

If multiple genres have the same average rating, order them by their name, in alphabetical order.

You can define additional helper views if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.
You cannot define SQL functions or PLpgSQL functions in this question.

Example output for this question can be found on the Examples page.

Q4 (3 marks) - SQL View

Write a SQL View, called Q4(Region, Average), that:
Retrieves the regions with an average runtime greater than the average runtime of all movies.
Your view should include the Region and the Average Runtime, rounded to an integer, for each region.
Your results should be ordered by the average runtime, from highest to lowest.

If multiple regions have the same average runtime, order them by their name, in alphabetical order.

You can define additional helper views if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.
You cannot define SQL functions or PLpgSQL functions in this question.

Example output for this question can be found on the Examples page.

Q5 (2 marks) - SQL Function

Write a SQL Function, called Q5(Pattern TEXT) RETURNS TABLE (Movie TEXT, Length TEXT), that:
Retrieves the movies whose title matches the given regular expression,

and displays their runtime in hours and minutes.
Your function should return the Primary Title and the runtime of each movie.
Your results should be ordered by the title of the movie, in alphabetical order.

You can define additional helper views and/or helper SQL functions if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.
You cannot define PLpgSQL functions in this question.

Example output for this question can be found on the Examples page.

Q6 (3 marks) - SQL Function

Write a SQL Function, called Q6(GenreName TEXT) RETURNS TABLE (Year Year, Movies INTEGER), that:
Retrieves the years with more than 10 movies released in a given genre.
Your function should return the Release Year and the number of movies released in that year.
Your results should be ordered by the number of movies, from highest to lowest.

If multiple years have the same number of movies, order them by their release year, from newest to oldest.

You can define additional helper views and/or helper SQL functions if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.
You cannot define PLpgSQL functions in this question.

Example output for this question can be found on the Examples page.

Q7 (4 marks) - SQL Function

Write a SQL Function, called Q7(MovieName TEXT) RETURNS TABLE (Actor TEXT), that:
Retrieves the actors who have played multiple different roles within the given movie.
Your function should return the Name of each actor.
Your results should be ordered by the name of the actor, in alphabetical order.

You can assume that MovieName refers to the Primary Title of a movie.
If multiple movies have the same Primary Title, actors for all movies should be returned.

You can define additional helper views and/or helper SQL functions if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used,

otherwise you will get an error when you try to load the file.
You cannot define PLpgSQL functions in this question.

Example output for this question can be found on the Examples page.

Q8 (3 marks) - PLpgSQL Function

Write a PLpgSQL Function, called Q8(MovieName TEXT) RETURNS TEXT, that:
Retrieves the number of releases for a given movie.
If the movie is not found, then an error message should be returned.

You can assume that MovieName refers to the Primary Title of a movie.
If multiple movies have the same Primary Title, the number of releases should be summed and returned as one value.

The output format should be as follows:

- If the movie is found and has >0 releases: `Release count: <N>`
- If the movie is found and has 0 release: `No releases found for "<MoveName>"`
- If the movie is not found: `Movie "<MoveName>" not found`

You can define additional helper views, helper SQL functions, and/or PLpgSQL functions if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.

Example output for this question can be found on the Examples page.

Q9 (4 marks) - PLpgSQL Function

Write a PLpgSQL Function, called Q9(MovieName TEXT) RETURNS SETOF TEXT, that: Prints the Cast and Crew of a given movie. For each cast member (Roles table) return a string containing the name of the person and the character they played. For each crew member (Credits table) return a string containing the name of the person and the job they did.

You can assume that MovieName refers to the Primary Title of a movie.
If multiple movies have the same Primary Title, the cast and crew for all movies should be returned.

The output format should be as follows:

- For each cast member: `"<Persons Name>" played "<Character Name>" in "<Movie Name>"`
- For each crew member: `"<Persons Name>" worked on "<Movie Name>" as a <Job Name>`

The order of the output does not matter, as long as all cast and all crew members are returned.

You can define additional helper views, helper SQL functions, and/or PLpgSQL functions if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.

Example output for this question can be found on the Examples page.

Q10 (5 marks) - PLpgSQL Function

Write a PLpgSQL Function, called Q10(MovieRegion CHAR(4)) RETURNS TABLE (Year INTEGER, Best_Movie TEXT, Movie_Genre Text,Principals TEXT), that:
Retrieves the list of must-watch movies for a given region, year by year.
A must-watch movie is defined as the movie with the highest rating in a given region.
Your function should return the Release Year, the Primary Title of the movie, the list of Genres of the movie, and the names of the Principals for each movie.
The Genres should be returned as a single string, with the names separated by commas.
The Principals should be returned as a single string, with the names separated by commas.
Your results should be ordered by the release year, from newest to oldest.
The list of Genres and Principals should be ordered alphabetically.

If no movies where released in the given region for a given year, then there should be no output for that year.
If multiple movies have a tied top score, all movies should be returned in alphabetical order of title
If no move has a recorded score for that year, no movie shuld be returned
If a move has no recorded genre, or no recorded principals, the movie should still be returned

You can define additional helper views, helper SQL functions, and/or PLpgSQL functions if you wish.
Remember to include all of your views in the `ass1.sql` file before you submit.
Remember that additional views must be defined **before** they are used, otherwise you will get an error when you try to load the file.

Example output for this question can be found on the Examples page.