Assignment 2
Python, Psycopg2 and IMDB

Last updated: **Thursday 7th April 10:31am**
Most recent changes are shown in red ... older changes are shown in brown.

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

Aims

This assignment aims to give you practice in

- manipulating a moderately large database (IMDB)
- implementing SQL views to satisfy requests for information
- implementing PLpgSQL functions to satisfy requests for information
- implementing Python scripts to extract and display data

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. You need to write Python scripts, using the Psycopg2 database connectivity module, to extract and display information from this database.

Summary

Submission:	Submit required files on **moodle**
Required Files:	`q1.py` `q2.py` `q3.py` `q4.py` `xtras.sql` (not necessary)
Deadline:	~~21:00 Friday 15 April~~ 21:00 Friday 19 April
Marks:	**20 marks** toward your total mark for this course
Late Penalty:	0.1 *marks* off the ceiling mark for each hour late, no submission is accepted 5 days (120h) after the deadline

How to do this assignment:

- read this specification carefully and completely
- create a directory for this assignment
- unpack the supplied files into this directory
- run your PostgreSQL server
- set up the supplied database (must be called `imdb`)
- complete the tasks below by editing the files
  - `q1.py` ... Python script to list people who have directed the most movies
  - `q2.py` ... Python script to show world releases for a Movie
  - `q3.py` ... Python script to show cast+crew for a Movie
  - `q4.py` ... Python script to show biography/filmography of a Name
  - `xtras.sql` ... Put any helper views or plpgsql functions here. It is not necessary.
- submit these files on **moodle** (you can submit multiple times)

If there exists `xtras.sql` in your submission, we will first load the file into the database by "psql imdb -f xtras.sql" and then test your python scripts. Details of the above steps are given below.

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 crticis 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 well-rated movies from the last 60 years. You can find more details on the database schema in the [Database Design] page.

Some comments about the data in our copy of IMDB: there seems to be preponderance of recent Bollywood movies; some of the aliases look incorrect (e.g. for "Shawshank Redemption"); the data only goes to mid 2019, so you won't find recent blockbusters.

Assignment Setup

Server Setup. This section describes how to carry out this assignment. Some of the instructions must be followed exactly; others require you to exercise some discretion. The initial instructions are targetted at people doing the assignment on d.cse. For the assignment setup on your home computer, you'll need to adapt instructions in "Local Setup".

The first step in setting up this assignment is to go to your directory and get templates of your python scripts for this assignment. Note that you can create a subfolder to store all assignment files if you like.

```
vxdb$ cd /localstorage/YourZid ... go to your working directory
vxdb$ unzip /home/cs3311/web/22T1/assignments/ass2/ass2.zip ... unzip all template files to your current director
Archive: /home/cs3311/web/22T1/assignments/ass2/ass2.zip
  inflating: q1.py
  inflating: q2.py
  inflating: q3.py
  inflating: q4.py
  inflating: xtras.sql
```

The second step is to load the database. Note that the database dump is quite large. Do not copy into your assignment directory on the CSE servers, because you only need to read it once to build your database (see below).

```
... login to d.cse and source env as usual ...
vxdb$ dropdb imdb  ... if you already had such a database
vxdb$ createdb imdb
vxdb$ bzcat /home/cs3311/web/22T1/assignments/ass2/database/imdb.dump.bz2 | psql imdb
vxdb$ psql imdb
... examine the database contents ...
```

Note the database contains non-ascii characters, and so you will need to make sure that your PostgreSQL server uses `UTF8` encoding (and corresponding collation) before it will load.

Loading the database should take less than 10 seconds on d.cse, assuming that d.cse is not under heavy load. (If you leave your assignment until the last minute, loading the database on d.cse will be considerably slower, thus delaying your work even more. The solution: at least load the database *Right Now*, even if you don't start using it for a while.) (Note that the `imdb.dump` file is 20MB in size; copying the compressed version under your CSE home directory or your `localstorage/` directory is not a good idea).

If you have other large databases under your PostgreSQL server on d.cse or you have large files under your `/localstorage/YOU/` directory, it is possible that you will exhaust your d.cse disk quota. In particular, you will not be able to store a copy of the MyMyUNSW database as well as the IMDB database under your d.cse server. The solution: remove any existing databases before loading your IMDB database. Using remote-ssh extension of VScode may also cause the disk quota issue. So stop using it.

Local Setup. Before downloading assignment files, your local machine should have installed **PostgreSQL**, **Python3** and **Psycopg2**. All files required in the assignment are listed below. You can download them by clicking the file name.

- `ass2.zip` ... templates of python script
- imdb.dump.bz2 ... compressed database dump file.

Download `ass2.zip` and `imdb.dump.bz2` to your home computer, unzip them, and perform commands analogous to the above. You should have a copy of the IMDB database that you can use at home to do this assignment. Think of some questions you could ask on the database and work out SQL queries to answer them. One useful query is

```
imdb=# select * from dbpop();
```

This will give you a list of tables and the number of tuples in each.

Your Tasks

Answer each of the following questions by writing a Python/Psycopg2 script. You can add any SQL views or PLpgSQL functions that are used in your Python scripts into the file `xtras.sql`. If you want to use any other Python modules, make sure that they are available on d.cse; your submitted code will be tested on `d.cse` using the Python installed there. You can change the indentation from the two-space indent in the templates. **Note that hardcoding is strictly forbidden.** Below are several tips to do the assignment.

- **Use PostgreSQL's case-insensitive regular expression pattern matching operator (`~*`) for matching partial names and titles. If you do this, it has the added advantage that your command-line arguments can be in any case you like, and you can even put regular expression chars into them.**
- **Some of the questions below require you to display crew roles. In the database, these are stored as all lower-case with underscores replacing spaces, e.g. "`production_manager`". When these are displayed, the first letter should be capitalised and each underscore should be replaced by a space character.**
- **Several questions require to sort items in the result. These can be achieved by the SQL keyword `ORDER BY`.**

Below is an example to run your scripts:

```
vxdb$ python3 q1.py 5
```

Q1 (2 marks)

Complete the script called "`q1.py`" so that it prints a list of the top *N* people who have directed the most movies (default *N* = 10). The script takes a single command-line argument which specifies how many people should appear in the list. People should be ordered from largest to smallest by the number of movies he/she directed, and should be displayed as, e.g.

```
vxdb$ python3 q1.py 5
48 Woody Allen
40 Takashi Miike
39 Jean-Luc Godard
37 Claude Chabrol
36 Martin Scorsese
```

Within groups of people with the same number of movies, people should be ordered alphabetically by his/her name (`Names.name`). If the user supplies a number less than 1, print the following message and exit.

```
vxdb$ python3 q1.py 0
Usage: q1.py [N]
```

For more examples of how the script behaves, see [Sample Outputs].

Q2 (3 marks)

Complete the script called "`q2.py`" so that it prints a list of the different releases (different regions, different languages) for a movie. The script takes a single command-line argument which gives a part of a movie name (could be the entire name or a pattern). If no argument is given, print the following message and exit.

```
vxdb$ python3 q2.py
Usage: q2.py 'PartialMovieTitle'
```


If there are no movies matching the supplied partial-name, then you should print a message to this effect and quit the program, e.g.


```
vxdb$ python3 q2.py xyzzy
No movie matching 'xyzzy'
```


If the partial-name matches multiple movies, simply print a list of matching movies (rating, title, year of release), ordered by rating (highest to lowest), then year of release (earliest to latest) and then by title (alphabetical order), e.g.

```
vxdb$ python3 q2.py mothra
Movies matching 'mothra'
===============
7.1 Godzilla, Mothra and King Ghidorah: Giant Monsters All-Out Attack (2001)
6.6 Mothra (1961)
6.5 Mothra vs. Godzilla (1964)
6.2 Godzilla and Mothra: The Battle for Earth (1992)
```

If the partial name matches exactly one movie, then print that movie's title and year, and then print a list of all of the other releases (aliases) of the movie. If there are no aliases, print "*Title* (*Year*) has no alternative releases". For each alias, show at least the title. If a region exists, add this, and if a language is specified, add it as well, e.g.,

```
vxdb$ python3 q2.py 2001
2001: A Space Odyssey (1968) was also released as
'2001' (region: XWW, language: en)
'Two Thousand and One: A Space Odyssey' (region: US)
'2001: Odisea del espacio' (region: UY)
'2001: Een zwerftocht in de ruimte' (region: NL)
```

Movie releases should be ordered accoring to the `ordering` attribute in the `Aliases` table.

If an alias has no region or language, then put the string in the `extra_info` field in the parentheses. If it has neither region, language or extra info, just print the local title (without parentheses).

Note that if there are two movies with exactly the same original title, you will not be able to view their releases, since the title alone does not distinguish them. We consider this problem in the next question. Such tests will not be covered in this question.

For more examples of how the script behaves, see [Sample Outputs].

Q3 (5 marks)

Complete the script called "`q3.py`" so that it prints a list of cast and crew for a movie. The script takes a command-line argument which gives a part of a movie name (could be the entire name or a pattern). It also takes an optional command-line argument, which is a year and can be used to distinguish movies with the same title (or, at least, titles which match the partial movie name).

```
vxdb$ python3 q3.py
Usage: q3.py 'MovieTitlePattern' [Year]
```

If there are no movies matching the supplied partial-name, then you should print a message to this effect and quit the program, e.g.,

```
vxdb$ python3 q3.py xyzzy
No movie matching 'xyzzy'
```

If the partial-name matches multiple movies, simply print a list of matching movies, the same as in Q2. If you need to disambiguate, either use a larger partial-name or add a year to the command line. If a longer partial-name and year still doesn't disambiguate, print a list of matching movies as above.

If the command-line arguments identify a single movie, then print the movie details (title and year), followed by a list of the principal actors and their roles, followed by a list of the principal crew members and their roles. The list of actors should be sorted according to the `ordering` attribute in the `Principals` table (i.e the biggest star comes first) and then by the name of the role they played if the `ordering` attribute is equal (i.e. one actor plays multiple roles). The list of crew members should also be sorted according to the `ordering` attribute in the `Principals` table and then by role name, if the `ordering` attribute is the same (e.g. one person has multiple crew roles).

Some movies are not in `Principals` table, they will not be tested. For more examples of how the script behaves, see [Sample Outputs].

Q4 (6 marks)

Complete the script called "`q4.py`" so that it prints a filmography for a given person (`Names`), showing their roles in each of the movies they have been a principal in. The script takes a command-line argument which gives a part of a person's name (could be the entire name or a pattern). It also takes an optional command-line argument, which is a year (their birth year) and which can be used to distinguish people with similar names. If there are no arguments or there are invalid arguments, print the following message and exit.

```
vxdb$ python3 q4.py
Usage: q4.py 'NamePattern' [Year]
```

If the name pattern, optionally combined with a year, doesn't match anyone in the `Names` table, then you should print a message to this effect and quit the program

```
vxdb$ python3 q4.py Smmith
No name matching 'Smmith'
```

If the name pattern, optionally combined with a year, matches more than one person, then print a list of the matching people, with the years they were born and died in parentheses afer the name. If birth year is invalid, e.g., null, regardless of whether death year is valid or not, print (???); if birth year is valid but death year is invalid, print (bbbb-); if both are valid, print (bbbb-dddd). Below is an example.

```
vxdb$: python3 q4.py rooney
Names matching 'rooney'
===============
Darrell Rooney (???)
Frank Rooney (1913-)
Jennie Rooney (???)
Mickey Rooney (1920-2014)
Nancy Rooney (???)
Rooney Mara (1985-)
Sharon Rooney (1988-)
```

Order people by their names alphabetically, then by birth year in chronological order. If two people have the same name and birth year, put them in order of their `Names.id` value (ascending).

If the name pattern, optionally combined with a year, matches a single person, then you should print their name and birth and death years (use the above rules), followed by two parts:

1. The first part
   Two pieces of information, the first one gives their 'personal rating', that is the average rating of all movies they have been a principal in. Use AVG() function to calculate the average, convert the average to decimal, then use ROUND(avg,1) to one decimal place. If no movie is found, set the persoanl rating as 0; the second one gives top 3 movie genres of all movies they have been a principal in. The movie genres should be ordered by the number of such movies they have been a principal in, then the genre name (alphabetically). If there are less than 3 genres in total, just print all of them. If no genre, leave blank. (See examples)
2. The second part
   A list of all the movies (with start years) they have been a principal in. Movies should be in chronological order; if there are multiple movies in a given year, order them by title within the year. For each movie, show first any acting roles they had, including the role they played, and then any production crew roles they had. If a movie has him/her in its principal list but no acting roles or crew roles can be found, only print the movie title and its start year. If no movie is found, leave blank. (See examples)

Note that one person could be both an actor and a director in the same movie. If a person has multiple roles as an actor in one movie, then show the records in order of the role name. Similarly for multiple roles as a crew member, order by role name.

```
vxdb$ python3 q4.py 'spike lee'
Filmography for Spike Lee (1957-)
===============
Personal Rating: 7.1
Top 3 Genres:
 drama
 comedy
 documentary
===============
She's Gotta Have It (1986)
 playing Mars Blackmon     -- acting role
 as Director               -- crew role
 as Writer
School Daze (1988)
 as Director
 as Writer
Do the Right Thing (1989)
 as Director
 as Writer
Mo' Better Blues (1990)
 playing Giant
 as Director
 as Writer
... etc. etc. etc. ...
```

For more examples of how the script behaves, see [Sample Outputs].

Style & Performance (2 marks)

Your programming style will be marked according to the following criteria

- consistent indentation (somewhat forced by Python)
- meaningful variable/function names
- efficient use of the database (see details below)

There is a performance requirement in each task. Any solution that takes longer than 10 seconds on any of our test cases will be penalised for that case, even if it eventually produces the correct result.

Submission & Testing

We will test your submission as follows:

- create a testing subdirectory containing your `xtras.sql` and Python scripts
- create a new database `imdb` and initialise it with `imdb.dump.bz2`
- run the command: `psql imdb -f xtras.sql` (if you provide `xtras.sql`)
- load and run the tests in the `check` script. (Additional test cases will be used for marking)

Your submitted code must be complete so that when we do the above, your python scripts and `xtras.sql` will load without errors, which accounts for 2 marks out of 20. You should thoroughly test your scripts before you submit them. If your Python or SQL scripts generate load-time errors and/or have missing definitions, you will be penalised by a 2 mark administrative penalty.

An example for the test script is provided in [testing]. Before submission, it would be useful to test out whether the submitted files work by following a similar sequence of steps to those noted above.